



KONSEP, FORMULASI, DAN ARSITEKTUR KODE KHGT TIMES

EDISI V8.0

*Dokumentasi Akademik, Formulasi Astronomi, dan Arsitektur Perangkat
Lunak*

KASMUI

Semarang, 2026

<https://hisabmu.com/khgttimes/> <https://kasmui.cloud/buku/>

INFORMASI PENERBITAN DAN EDISI

Elemen	Keterangan
Judul	Konsep, Formulasi, dan Arsitektur Kode KHGT Times - Edisi V8.0
Penulis	Kasmui
Edisi	Revisi akademik V8.0, 2026
Basis dokumentasi	khgttimes_v8.0(4).py
Ukuran source code	1,025,843 byte; 18,160 baris
Pemeriksaan sintaks	Berhasil diparsing sebagai source Python tanpa SyntaxError.
SHA-256 source asli	e90f9e2bf766a78e0548df65fd2a1a785a9b529f8fb65e9921b31530fc3ef32a
Subjek	Astronomi Islam; Kalender Hijriah; Komputasi Astronomi; Astrometri; Rekayasa Perangkat Lunak; Python
Penerbitan	Semarang, 2026. Jaten Press / dokumentasi teknis KHGT Times.

Catatan keamanan publikasi. Lampiran source code pada edisi buku ini menggunakan salinan yang disanitasi untuk publikasi. Nilai API key, token, dan password lokal tidak dicetak ulang. Sanitasi tidak mengubah struktur pembahasan, nama fungsi, atau alur arsitektur yang didokumentasikan.

KATA PENGANTAR EDISI V8.0

Segala puji bagi Allah Subhanahu wa Ta'ala, yang menjadikan pergerakan Matahari, Bulan, dan benda-benda langit sebagai tanda keteraturan kosmos serta dasar pengukuran waktu. Edisi V8.0 buku ini disusun sebagai pembaruan menyeluruh atas dokumentasi KHGT Times agar uraian teoritis, struktur menu, contoh arsitektur, dan lampiran source code selaras dengan implementasi perangkat lunak terbaru yang disertakan bersama naskah.

Pembaruan V8.0 terutama penting pada mesin ephemeris. Aplikasi tidak lagi dipahami sebagai sistem dengan satu kernel DE441. Source code V8.0 menetapkan de440s.bsp sebagai kernel ringkas untuk tahun penuh 1850-2149 dan de441-new.bsp untuk tahun 1-1849 serta 2150-2999. Pemilihan dilakukan otomatis berdasarkan tahun target, sementara kernel yang pernah dimuat disimpan dalam cache internal. Pendekatan ini mempertahankan rentang aplikasi yang panjang sekaligus mengutamakan kernel yang lebih ringkas untuk era modern.

Dari sisi antarmuka dan modularitas, V8.0 menyelaraskan 41 menu aktif. Menu 39 adalah Peta Langit Dunia yang membuka layanan berbasis web, Menu 40 menjalankan aplikasi pendamping Laporan Astronomi bila tersedia, dan Menu 41 adalah HIJRI_DB Auto-Builder untuk administrator. Edisi ini juga mencatat evolusi fitur dari V7.5-V7.7 yang kini menjadi bagian dari basis V8.0, antara lain antarmuka Indonesia-Inggris-Arab, penyempurnaan Altitude Chart Analyser, visualisasi rasi dan planet pada Live Animasi, Tracker, serta sinkronisasi data dengan Stellarium.

Buku ini mempertahankan kerangka ilmiah edisi sebelumnya - astrometri, visibilitas hilal, waktu salat, kiblat, gerhana, analemma, visualisasi, kecerdasan buatan, dan basis data - namun setiap bagian yang terkait langsung dengan kode telah disesuaikan dengan implementasi V8.0. Penomoran menu historis yang tidak lagi konsisten dihindari dalam pembahasan naratif; daftar resmi 41 menu V8.0 disajikan secara eksplisit pada Bab 11.

Semoga edisi ini dapat berfungsi sebagai dokumentasi akademik yang lebih tertib, dapat diaudit, serta berguna bagi mahasiswa, peneliti, praktisi falak, dan pengembang perangkat lunak astronomi.

Semarang, 2026

Penulis

Kasmui

CATATAN PEMBARUAN EDISI V8.0

Tabel berikut merangkum jejak pembaruan yang tertanam pada dokumentasi rilis di source code V8.0. Riwayat versi dipertahankan karena membantu pembaca memahami asal-usul fitur yang kini tersedia pada V8.0.

Pembaruan	Ringkasan implementasi
[V7.7 - MULTIBAHASA & SIDEBAR] Antarmuka Indonesia, Inggris & Arab	Sidebar dan elemen antarmuka utama kini mendukung penerjemahan ke Bahasa Indonesia, Inggris, dan Arab sehingga navigasi modul dapat digunakan secara multibahasa.
[V7.7 - ALTITUDE CHART ANALYZER] Analisis Grafik & Kombinasi	Menu 9 (Altitude Chart Analyser) menyediakan analisis grafik ketinggian dan kombinasi data untuk membantu pembacaan posisi benda langit secara lebih visual dan terstruktur.
[V8.0 - PETA LANGIT DUNIA] Integrasi Peta Langit Online	Menu 39 adalah Peta Langit Dunia. Saat menu dipilih, KHGT Times membuka layanan peta langit online pada https://kasmui.cloud/langit/ melalui browser default.
[V8.0 - LAPORAN ASTRONOMI] Penyesuaian Nomor Menu	Menu 40 adalah Laporan Astronomi. Menu ini membuka aplikasi pendamping 'LaporanAstronomi.exe' atau 'LaporanAstronomi.exe' dari folder aplikasi KHGT Times dan menampilkan pesan kesalahan jika file pendamping tidak tersedia atau gagal dijalankan.
[V8.0 - MENUS & DOKUMENTASI] Penyelarasan 41 Modul	Daftar menu aktual V8.0 terdiri atas 41 menu. Menu 39 adalah Peta Langit Dunia, Menu 40 adalah Laporan Astronomi, dan Menu 41 adalah HIJRI_DB Auto-Builder (Admin). Manual Book, highlight rilis, dan panduan navigasi pada halaman ini telah diselaraskan dengan struktur aktual tersebut.
[V8.0 - ENGINE] Auto-Switch Ephemeris Matahari & Bulan	Sistem otomatis menggunakan de440s.bsp untuk tahun 1850-2149 dan beralih ke de441-new.bsp untuk tahun 1-1849 atau 2150-2999, sehingga perhitungan modern Matahari-Bulan memakai kernel ringkas berpresisi tinggi tanpa kehilangan jangkauan panjang aplikasi.
[V7.6 - LIVE ANIMASI] Sinkronisasi Stellarium	Pada output Live Animasi ditambahkan tombol Sinkronisasi Stellarium untuk mengaktifkan Stellarium dan membandingkan data Matahari dan Bulan antara output Stellarium dengan hasil hitung sistem KHGT berbasis ephemeris DE441.
[V7.6 - LIVE ANIMASI] Tracker Benda Langit	Pada output Live Animasi ditambahkan tombol Tracker untuk pelacakan posisi benda langit, baik menggunakan GPS maupun tanpa GPS.
[V7.5 - LIVE ANIMASI] Rasi Bintang & Planet	Pembaruan pada Modul 36 (Live Animasi) dilengkapi visualisasi Rasi Bintang (Constellations) dan planet-planet tata surya secara dinamis dan real-time.
[V7.5 - WINAI] Asisten AI Universal	Peningkatan kecerdasan WinAI Interceptor agar mampu mendeteksi niat (intent) pengguna dan mengeksekusi pertanyaan atau perintah perhitungan yang berkaitan dengan menu di dalam sistem.
[V7.5 - ANALEMMA] Grafik Kurva Matahari	Peningkatan visual pada Modul 33 (Generator Analemma Matahari) yang menampilkan grafik kurva lintasan Matahari berbentuk angka 8.
[V7.5 - BUG FIXES] Gerhana, Kiblat & Optimalisasi	Perbaikan bug pada rentang waktu komputasi di Modul 30 (Analisis Gerhana), perbaikan akurasi pada Modul 26 (Qiblah Direction and Times), serta berbagai penyesuaian

	stabilitas sistem.
[V7.4 - ENGINE] Penyederhanaan Ephemeris DE441	Sistem menggunakan file ephemeris 'de441-new.bsp' untuk kalkulasi pada rentang waktu yang didukung aplikasi, sehingga pengelolaan basis ephemeris menjadi lebih sederhana.
[V7.3 - HIGHLIGHTS] Modul Fisis & Auto-Builder	Modul Astrofotografi, Prediktor Pasang Surut, Evaluasi SQM Fajar, dan HIJRI_DB Auto-Builder untuk Administrator merupakan pengembangan yang diperkenalkan pada rilis sebelumnya.

Prinsip dokumentasi edisi ini. Jika teks edisi lama menyebut satu nomor modul yang telah berubah, acuan final adalah daftar 41 menu V8.0 pada source code. Istilah teoritis seperti DE440/DE441 tetap dibahas sebagai keluarga ephemeris JPL, tetapi implementasi aplikasi mengikuti mekanisme auto-switch de440s/de441-new.

DAFTAR ISI

BAB 1 - Pendahuluan: Transisi Paradigma Falak Digital

BAB 2 - Mesin Astrometri dan Manajemen Waktu

BAB 3 - Formulasi Visibilitas Hilal Multi-Kriteria

BAB 4 - Algoritma Pelacakan Titik Pertama KHGT

BAB 5 - Pemetaan Visual Spasial

BAB 6 - Formulasi Waktu Salat dan Fajar Shadiq

BAB 7 - Trigonometri Arah Kiblat dan Bayangan Matahari

BAB 8 - Mekanika Syzygy: Gerhana dan Analemma

BAB 9 - Simulator 3D dan Animasi Interaktif

BAB 10 - Integrasi Kecerdasan Buatan dan Antarmuka Multibahasa

BAB 11 - Manajemen Database, Komparasi, dan Arsitektur 41 Menu

BAB 12 - Penutup dan Proyeksi Masa Depan Falak Digital

Daftar Pustaka

Glosarium

LAMPIRAN 1 - Parameter, koordinat referensi, dan konfigurasi V8.0

LAMPIRAN 2 - Manajemen berkas BSP dan strategi dua-kernel

LAMPIRAN 3 - Dependensi dan lingkungan eksekusi

LAMPIRAN 4 - Source code KHGT Times V8.0 (sanitasi publikasi)

BAB 1: PENDAHULUAN: TRANSISI PARADIGMA FALAK DIGITAL

Pembaruan implementasi V8.0: mesin ephemeris menggunakan dua kernel. Fungsi `_ephemeris_filename_for_year()` memilih `de440s.bsp` untuk tahun 1850-2149 dan `de441-new.bsp` untuk tahun 1-1849 atau 2150-2999. Fungsi `_load_ephemeris_file()` menyimpan kernel yang pernah dibuka dalam cache dan memverifikasi tersedianya objek Earth, Sun, dan Moon. Rentang aplikasi dibatasi 1-2999 M.

1.1. Evolusi Komputasi Astronomi Islam

Ilmu Falak (Astronomi Islam) merupakan salah satu disiplin ilmu eksakta tertua yang mengalami evolusi paling dinamis dalam peradaban manusia. Secara historis, peradaban Islam klasik mengandalkan observasi mata telanjang (*rukyat al-basariyyah*) yang didukung oleh instrumen mekanik analog seperti astrolabe (*al-asturlab*) dan kuadran (*ar-rub'*). Data observasi ini kemudian ditransformasikan ke dalam tabel-tabel astronomi matematis yang dikenal sebagai *Zij* (misalnya *Zij Ilkhani* karya Nashiruddin ath-Thusi atau *Zij as-Sultani* karya Ulugh Beg).

Memasuki era modern, revolusi digital mengubah konstelasi Ilmu Falak secara radikal. Tabel-tabel *Zij* yang dihitung secara manual selama berbulan-bulan kini digantikan oleh algoritma komputasi yang mampu merender triliunan titik koordinat benda langit dalam fraksi detik. Evolusi ini memfasilitasi transisi dari hisab *tahqiqi* (perhitungan presisi klasik) menuju hisab kontemporer yang beresolusi super tinggi, yang pada gilirannya menjadi fondasi teknis bagi penyatuan Kalender Hijriah Global Tunggal (KHGT) (Ilyas, 1997).

1.1.1. Dari Ephemeris Klasik menuju Integrasi Numerik (JPL NASA)

Selama akhir abad ke-20 hingga awal abad ke-21, komputasi astronomi di seluruh dunia—termasuk dalam perangkat lunak falak Islam generasi awal—sangat bergantung pada deret analitik (*analytical series*). Salah satu algoritma yang paling mendominasi adalah VSOP87 (*Variations Séculaires des Orbites Planétaires*) yang dikembangkan oleh Bretagnon dan Franco (1988) dari Bureau des Longitudes, Paris. Algoritma analitik ini bekerja dengan memecahkan persamaan gerak planet ke dalam ribuan deret polinomial harmonik. Untuk menghitung posisi Matahari atau planet, komputer cukup mengevaluasi deret sinus dan kosinus ini pada waktu tertentu (*t*).

Namun, seiring dengan meningkatnya kebutuhan akan akurasi absolut, terutama dalam penentuan posisi hilal di garis batas kriteria visibilitas yang sangat kritis (seperti syarat ketinggian geosentris 5 derajat pada KHGT), model analitik mulai ditinggalkan. Paradigma bergeser menuju metode Integrasi Numerik (*Numerical Integration*), yang menjadi jantung dari *Development Ephemeris (DE)* buatan Jet Propulsion Laboratory (JPL) NASA.

Berbeda dengan VSOP87 yang menggunakan pendekatan matematis hampiran, sistem integrasi numerik JPL menghitung dinamika gravitasi tata surya selangkah demi selangkah (menggunakan metode seperti *Adams-Bashforth-Moulton*) berdasarkan hukum Relativitas Umum Einstein. Model ini memperhitungkan interaksi tarik-menarik antara Matahari, 8 planet, Pluto, Bulan, hingga ratusan asteroid besar yang secara dinamis terus memengaruhi posisi Bulan (Park et al., 2021).

Dalam arsitektur perangkat lunak KHGT Times V8.0, lompatan teknologi ini secara tegas diimplementasikan melalui pengelolaan dua kernel ephemeris, `de440s.bsp` dan `de441-new.bsp`, yang dipilih otomatis berdasarkan tahun target. File DE441 merupakan standar ephemeris NASA terbaru yang mencakup rentang waktu sangat panjang (dari tahun -13200 hingga +17191 Masehi) dengan tingkat akurasi presisi pada level sub-milidetik busur (*milliarcsecond*) (Park et al., 2021). KHGT Times membaca struktur biner Chebyshev polynomials

di dalam file ini menggunakan library saintifik skyfield, sebagaimana tercermin dalam inisialisasi mesin utamanya:

```
self.load_obj = Loader(BASE_DIR, verbose=False)
self.ts = self.load_obj.timescale()
self.eph = self.auto_switch_ephemeris(target_year)
```

Pendekatan ini memastikan bahwa setiap perhitungan umur bulan (moon age), elongasi, dan ketinggian bebas dari akumulasi truncation error yang sering merusak hasil hisab jangka panjang.

1.1.2. Keterbatasan Algoritma Lama (VSOP87) di Era Modern

Peralihan menuju Ephemeris NASA dalam KHGT Times V8.0 bukan tanpa alasan empiris. Keterbatasan utama dari algoritma klasik analitik (seperti VSOP87 atau ELP2000-82B untuk Bulan) terletak pada degradasi akurasi temporal. Deret analitik dipotong (truncated) pada jumlah tertentu demi menghemat daya komputasi komputer dekade 1980-an (Urban & Seidelmann, 2013). Akibatnya, semakin jauh komputasi ditarik ke masa lalu atau diproyeksikan ke masa depan (misalnya simulasi komparasi 50 tahun ke depan untuk penentuan awal Ramadhan dan Syawal), simpangan (error/drift) posisinya akan semakin membesar.

Selain itu, algoritma lama sering kali gagal menangani perturbasi jangka pendek tingkat mikro pada orbit bulan (seperti efek pasang surut kelautan dan anomali geopotensial bumi). Dalam konteks penyatuan Kalender Hijriah Global, kesalahan fraksi derajat atau keterlambatan hitungan konjungsi (ijtimak) sebesar hitungan detik sangat berakibat fatal.

Sebagai contoh, berdasarkan Parameter Kalender Global (PKG) 1 KHGT, jika kriteria visibilitas terpenuhi di mana saja di bumi tepat sebelum pukul 00:00 UTC, maka bulan baru dimulai keesokan harinya secara global. Jika mesin komputasi menggunakan VSOP87 mengalami bias waktu ijtimak atau bias perhitungan elongasi bumi-bulan pada detik-detik kritis (misalnya pukul 23:59:58 UTC), sistem kalender global dapat mengalami kegagalan validasi lintas benua, menyebabkan pergeseran hari raya (Syamsuddin, 2021).

Oleh karena itu, perangkat lunak astronomi masa depan menuntut standarisasi data yang tidak lagi mengandalkan kalkulator built-in dari rumusan analitik. Penggunaan integrasi multi-library—di mana skyfield menangani geometri absolut DE441, dan ephem (PyEphem berbasis algoritma C klasik) direduksi fungsinya sebatas accelerator pelacakan fajar/senja secara brute-force—merupakan solusi teknis yang brilian dalam arsitektur KHGT Times untuk mencapai performa yang robust tanpa mengorbankan akurasi.

1.2. Urgensi Kalender Hijriah Global Tunggal (KHGT)

Paradigma penanggalan dalam Islam tidak sekadar berfungsi sebagai penanda administrasi sipil, melainkan memiliki implikasi ibadah (ta'abbudi) yang sangat sakral. Puasa Ramadhan, Idul Fitri, ibadah Haji, hingga masa iddah dan haul zakat seluruhnya bertumpu pada presisi pergerakan bulan kamariah. Namun, selama berabad-abad, umat Islam di berbagai belahan dunia merayakan hari-

hari besar tersebut pada tanggal masehi yang berbeda-beda. Fragmentasi temporal ini bukan berakar pada kelemahan sains, melainkan pada dikotomi penafsiran yurisprudensi (fikih) terhadap teks-teks syariat terkait jangkaun keberlakuan rukyatul hilal.

Menjawab tantangan di era global village ini, Kalender Hijriah Global Tunggal (KHGT) yang dirumuskan pada Kongres Istanbul 2016 hadir sebagai sintesis antara ortodoksi fikih dan kecanggihan astrometri modern.

1.2.1. Dalil Syar'i tentang Penyatuan Waktu Global

Secara teologis, KHGT dibangun di atas premis bahwa syariat Islam diturunkan untuk seluruh umat manusia secara universal (rahmatan lil 'alamin). Universalitas ini menuntut adanya sistem penjadwalan yang sinkron di seluruh dunia. Fondasi utama penyatuan awal bulan termaktub dalam firman Allah Subhanahu wa Ta'ala:

شَهْرُ رَمَضَانَ أَلْ ذِي أَنْزَلَ فِيهِ الْقُرْآنُ هُدًى لِّلنَّاسِ وَبَيِّنَاتٍ مِّنَ الْهُدَىٰ وَالْقُرْآنِ قَمَرٍ قَمَرٍ
شَهْرُ رَمَضَانَ أَلْ ذِي أَنْزَلَ فِيهِ الْقُرْآنُ هُدًى لِّلنَّاسِ وَبَيِّنَاتٍ مِّنَ الْهُدَىٰ وَالْقُرْآنِ قَمَرٍ قَمَرٍ

Artinya: "(Beberapa hari yang ditentukan itu ialah) bulan Ramadhan, bulan yang di dalamnya diturunkan (permulaan) Al-Qur'an sebagai petunjuk bagi manusia dan penjelasan-penjelasan mengenai petunjuk itu dan pembeda (antara yang hak dan yang bathil). Karena itu, barangsiapa di antara kamu hadir (di negeri tempat tinggalnya) di bulan itu, maka hendaklah ia berpuasa pada bulan itu..." (QS. Al-Baqarah [2]: 185)

Kata syahida (شَهِدَ) dalam ayat tersebut, secara etimologis dan epistemologis, tidak dimaknai secara sempit sebagai "melihat bulan dengan mata kepala secara fisik", melainkan bermakna hadir atau mengetahui dengan keyakinan (ilmu) bahwa bulan baru telah masuk (Anwar, 2008). Keyakinan inilah yang di era modern dicapai melalui instrumen hisab yang presisi.

Lebih lanjut, hadits Nabi Shallallahu 'alaihi wa Sallam yang sering dijadikan rujukan utama dalam penentuan awal bulan berbunyi:

صُومُوا لِرُؤْيَيْهِ وَأَفْطِرُوا لِرُؤْيَيْهِ فَإِنْ عُمَ عَلَيْكُمْ فَأَكْمِلُوا عِدَّةَ شَعْبَانَ ثَلَاثِينَ

Artinya: "Berpuasalah kalian karena melihatnya (hilal) dan berbukalah kalian karena melihatnya. Jika ia tertutup awan dari kalian, maka sempurnakanlah hitungan bulan Syakban menjadi tiga puluh hari." (HR. Bukhari no. 1909 dan Muslim no. 1081).

Dari perspektif ushul fiqh, lafaz shumuu (صُومُوا) dan aftiruu (أَفْطَرُوا) menggunakan shighat jama' (bentuk jamak/kolektif) yang ditujukan kepada seluruh umat Islam (Zayed, 2015). Ini mengindikasikan bahwa perintah puasa dan berhari raya pada hakikatnya adalah perintah kolektif- universal, bukan individual-regional. Pemahaman klasik yang membatasi keberlakuan rukyat hanya pada zona geografis tertentu (ikhtilaf al-matali') dinilai tidak lagi relevan ketika teknologi komunikasi memungkinkan umat Islam di Timur mengetahui hasil observasi astronomis di Barat dalam hitungan mikrodetik (Ilyas, 1997).

1.2.2. Prinsip Kesatuan Matlak (Ittihad al-Matali') dan Transfer Visibilitas

Untuk merealisasikan makna kolektif dari dalil-dalil tersebut, KHGT mengadopsi prinsip Kesatuan Matlak (Ittihad al-Matali'). Prinsip ini menegaskan bahwa bumi adalah satu kesatuan zona penanggalan (satu matlak). Implikasi logis dari prinsip ini adalah berlakunya mekanisme Transfer Visibilitas (Naql al-Rukyat). Secara saintifik, jika hilal telah memenuhi kriteria visibilitas (Ketinggian $\geq 5^\circ$ dan Elongasi $\geq 8^\circ$) di belahan bumi Barat (misalnya di benua Amerika), maka status bulan baru tersebut segera "ditransfer" ke belahan bumi Timur, meskipun di wilayah Timur matahari telah terbenam lebih dulu dan hilal belum memenuhi kriteria secara lokal.

Dalam implementasi algoritmiknya, KHGT Times V8.0 dirancang secara khusus untuk menjadi engine pelacak (scanner) matlak global ini. Hal ini terlihat sangat jelas pada Menu 7 (Kota Pertama KHGT / Mainland). Melalui fungsi calculate_first_point(), program mengeksekusi metode brute-force yang memindai ribuan koordinat dalam direktori CITY_DB secara simultan.

Alih-alih hanya menghitung di satu titik pengamat (toposentris murni) seperti perangkat lunak falak konvensional, kode program KHGT Times menerapkan iterasi terhadap New Moon (Ijtimak)

menggunakan metode `ephem.previous_new_moon()` dan `ephem.next_new_moon()`, kemudian melacak waktu sunset di setiap kota:

Sistem secara cerdas menerapkan Filter Zona Pengecualian untuk mematuhi aturan daratan utama (mainland rule):

```
zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu",  
"Maluku", "Maluku Utara", "NTT"]
```

Program memastikan bahwa titik pertama dimulainya bulan baru bukan berada di pulau kecil yang terisolasi di samudra Pasifik, melainkan di daratan yang memiliki populasi manusia berkelanjutan. Jika titik valid pertama ditemukan sebelum pukul 00:00 UTC (PKG 1), maka seluruh dunia secara serentak (Ittihad al-Matali') akan memulai tanggal 1 pada keesokan harinya. Inilah perwujudan nyata dari dalil syar'i yang diterjemahkan ke dalam bahasa pemrograman tingkat tinggi.

1.3. Visi dan Arsitektur KHGT Times V8.0

Perangkat lunak falak yang sekadar "menghitung" jadwal sudah sangat tertinggal untuk menjawab tantangan astronomi abad ke-21. Visi utama dari pengembangan KHGT Times V8.0 adalah meruntuhkan dinding pembatas antara kompleksitas ephemeris tingkat tinggi milik lembaga antariksa dan aksesibilitas user interface (GUI) bagi praktisi, akademisi, serta pembuat kebijakan syariah. Arsitektur V8.0 didesain bukan sekadar sebagai kalkulator, melainkan sebagai sebuah engine komputasi (mesin pemroses) berskala besar yang mampu mengeksekusi integrasi numerik, komparasi multi-kriteria global, hingga rendering visual spasial secara bersamaan.

1.3.1. Keseimbangan High Precision dan High Performance

Dalam paradigma komputasi saintifik, secara arsitektural selalu terdapat trade-off (kompromi) antara high precision (akurasi ekstrem) dan high performance (kecepatan eksekusi) (Virtanen et al., 2020). Mengakses posisi vektor tiga dimensi dari bulan dan bumi melalui Development Ephemeris (DE441) membutuhkan evaluasi polinomial Chebyshev yang memakan daya komputasi (CPU cycles) jauh lebih besar dibandingkan mengevaluasi deret analitik sederhana.

Jika sistem KHGT Times secara naif melakukan perhitungan looping linear $O(n)$ untuk melacak waktu terbenamnya matahari (sunset) di 300 kota di dunia selama 30 hari berturut-turut, proses tersebut akan memicu bottleneck (kemacetan) dan membekukan antarmuka aplikasi (GUI freeze).

Untuk memecahkan kebuntuan tersebut, arsitektur KHGT Times V8.0 menerapkan dua metodologi utama:

1. Pendekatan Vektorisasi Array: Mengonversi data waktu diskrit menjadi larik (array)

waktu berskala besar menggunakan library NumPy (seperti implementasi `np.linspace`). Evaluasi posisi benda langit tidak lagi dilakukan satu per satu per detik, melainkan dihitung secara simultan menggunakan komputasi matriks yang langsung dieksekusi di level bahasa C. 2. Asynchronous Threading: Seluruh komputasi masif, seperti pemindaian daratan utama

dan komparasi 50 tahun (modul terkait), dieksekusi di luar Main Thread GUI menggunakan modul threading internal Python. Hal ini memastikan UI tetap responsif (non-blocking) sementara komputasi big data berjalan di latar belakang (Bassi, 2007).

1.3.2. Ekosistem Library Python (Pusat Kendali Astrometri)

Bahasa pemrograman Python dipilih sebagai fondasi KHGT Times bukan karena kecepatannya, melainkan karena kapabilitasnya sebagai glue language (bahasa perekat) yang mampu menyatukan berbagai pustaka (library) saintifik berbasis C, C++, dan Fortran. Arsitektur V8.0 didukung oleh ekosistem library berikut yang saling bersinergi:

1. Skyfield API: Berperan sebagai jantung astrometri utama menggantikan modul-modul

usang. Skyfield memproses file biner NASA (de441-new.bsp) dan bertanggung jawab atas perhitungan relativistik, refraksi atmosfer, koreksi paralaks, dan rotasi matriks koordinat geosentris menuju toposentris menggunakan referensi datum elipsoid WGS84 mutakhir (Rhodes, 2019). 2. SciPy & NumPy: Pustaka esensial untuk manipulasi array dan matematika kompleks.

Pada modul Crescent Visibility HD Map, sistem menggunakan sub-modul `scipy.interpolate.griddata` dengan algoritma Bicubic Interpolation. Algoritma ini mengubah titik-titik diskrit hasil perhitungan kasar dari berbagai lintang dan bujur menjadi matriks kontur spasial spasial beresolusi tinggi yang sangat mulus (seamless). 3. Matplotlib: Mesin rendering yang mentransformasi data mentah numerik menjadi

representasi visual saintifik, seperti Grafik Ketinggian Harian, Simulasi Kompas

Mizwala 2D, hingga interaksi kanvas 3D Geocentric Space secara real-time (Hunter, 2007). 4. PyEphem (ephem): Meskipun Skyfield unggul dalam akurasi absolut, kalkulasi pencarian

titik kritis (discrete events) seperti iterasi pencarian sunset berulang sangat membebani komputasi. Oleh karena itu, PyEphem yang ditulis langsung dalam bahasa C (diadopsi dari XEphem) dipertahankan secara khusus oleh arsitek KHGT Times sebagai akselerator brute-force untuk melakukan pemindaian global pada ratusan kota secara instan.

Integrasi harmonis dari ekosistem di atas menjadikan KHGT Times V8.0 tidak hanya presisi dan diakui secara akademik, tetapi juga siap digunakan secara praktis oleh komunitas astronomi global.

DAFTAR PUSTAKA BAB 1

Anwar, S. (2008). Hari Raya dan Problematika Penentuan Awal Bulan Hijriah. Suara Muhammadiyah.

Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10-20.

Bretagnon, P., & Francou, G. (1988). Planetary theories in rectangular and spherical variables. VSOP 87 solutions. *Astronomy and Astrophysics*, 202, 309-315.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.

Ilyas, M. (1997). Islamic Calendar, Times & Qibla. Berita Publishing.

Park, R. S., Folkner, W. M., Williams, J. G., & Boggs, D. H. (2021). The JPL planetary and lunar ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3), 105.

Rhodes, B. (2019). Skyfield: High precision research-grade positions for planets and Earth satellites. *Astrophysics Source Code Library*, ascl-1907.

Syamsuddin, S. (2021). Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi. Majelis Tarjih dan Tajdid PP Muhammadiyah.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.

Zayed, A. (2015). Fikih Astronomi: Menjawab Problematika Penentuan Awal Bulan. Darul Ulum.

BAB 2: MESIN ASTROMETRI DAN MANAJEMEN WAKTU (TIME & SPACE)

Catatan V8.0: uraian teoritis DE441 pada edisi sebelumnya tetap relevan sebagai konteks keluarga ephemeris numerik JPL, tetapi kode operasional tidak memuat satu kernel secara permanen. Semua modul yang memerlukan Matahari-Bulan diarahkan ke mekanisme auto-switch de440s/de441-new berdasarkan tahun target.

ENGINE)

Dalam komputasi astronomi modern, ketepatan penentuan visibilitas hilal tidak lagi sekadar bergantung pada geometri dua dimensi, melainkan sangat terikat pada model relativistik tata surya dan sistem manajemen waktu yang seragam. Bab ini akan membedah secara mendalam bagaimana KHGT Times V8.0 membangun mesin astrometrinya (Astrometry Engine) menggunakan pustaka Skyfield dan basis data ephemeris standar NASA, serta bagaimana sistem menangani kerumitan dilatasi waktu fisik.

2.1. Manajemen Ephemeris JPL V8.0: de440s dan de441-new

Sistem tata surya tidak bergerak dalam lintasan melingkar yang diam, melainkan dalam perturbasi gravitasi yang sangat kompleks yang saling tarik-menarik. Perangkat lunak KHGT Times meninggalkan formulasi analitik klasik dan mengadopsi ephemeris numerik JPL melalui kernel de440s dan de441-new sesuai rentang tahun yang dirilis oleh Jet Propulsion Laboratory (JPL) NASA. DE441 merupakan model integrasi numerik yang dipetakan untuk merentang dari tahun 13200 Sebelum Masehi hingga 17191 Masehi, menjadikannya basis data paling robust untuk simulasi kalender jangka panjang (Park et al., 2021).

Dalam arsitektur kode KHGT Times, mesin ini diinisialisasi melalui sintaks pengikatan library Skyfield:

```
self.load_obj = Loader(BASE_DIR, verbose=False)
self.ts = self.load_obj.timescale()
self.eph = self.auto_switch_ephemeris(target_year)
```

File de441-new.bsp (Binary Space Partition) tidak memuat rumus lintasan orbit, melainkan tabel koefisien polinomial yang sangat padat. Untuk membaca data ini secara akurat, sistem membutuhkan pemahaman mendalam tentang skala waktu relativistik.

2.1.1. Konsep Barycentric Dynamical Time (TDB) dan Terrestrial Time (TT)

Kesalahan fatal yang sering terjadi pada pemrograman falak pemula adalah menggunakan waktu sipil (seperti UTC atau Waktu Lokal) secara langsung sebagai parameter pengukur gerak planet. Waktu sipil (UTC) didasarkan pada rotasi bumi yang tidak konstan dan secara berkala dikoreksi dengan detik kabisat (leap second). Jika UTC digunakan murni untuk melacak posisi bulan 50 tahun ke depan, akan terjadi pergeseran (drift) yang membuat perhitungan meleset (Urban & Seidelmann, 2013).

Oleh karena itu, KHGT Times V8.0 secara arsitektural menggunakan `self.ts = self.load_obj.timescale()` untuk mengelola skala waktu. Ketika pengguna memasukkan

waktu lokal, misalnya melalui variabel `waktu_utc`, Skyfield secara internal melakukan transformasi hierarkis:

1. UTC ke TT (Terrestrial Time): Sistem menyerap tabel leap second (ΔT / Delta T)

untuk mengonversi waktu tidak beraturan menjadi waktu seragam di permukaan bumi. 2. TT ke TDB (Barycentric Dynamical Time): Karena DE441 berpusat pada pusat massa

tata surya (Solar System Barycenter), waktu harus dikoreksi kembali berdasarkan efek relativitas umum Einstein. Saat bumi berada di perihelion (dekat dengan matahari), gravitasi yang lebih kuat

memperlambat laju waktu (dilatasi waktu) relatif terhadap pengamat di luar tata surya. TDB adalah skala waktu independen yang mengoreksi efek relativistik tersebut.

Dalam skrip KHGT Times, hal ini terlihat jelas pada modul konversi di mana waktu dipaksa berjalan pada parameter TT yang absolut:

```
t_start_conj = self.ts.tt_jd(t_eval_tt - 5.0)
t_end_conj = self.ts.tt_jd(t_eval_tt + 5.0)
```

Penggunaan tt_jd (Terrestrial Time dalam Julian Date) menjamin bahwa pencarian waktu New Moon (Ijtimak) tidak akan terdistorsi oleh perlambatan rotasi bumi di masa depan.

2.1.2. Pembacaan Vektor Posisi (X, Y, Z) Benda Langit

Setelah sistem berhasil menerjemahkan waktu pengguna ke dalam TDB, langkah berikutnya adalah mengekstraksi koordinat posisi benda langit dari file .bsp. Di dalam DE441, pergerakan benda langit direpresentasikan dalam deret polinomial Chebyshev. Deret ini memungkinkan kompresi data triliunan posisi planet menjadi ukuran file yang dapat diproses oleh RAM komputer standar (Rhodes, 2019).

Dalam KHGT Times V8.0, pembacaan vektor ini tidak dieksekusi secara manual, melainkan dipanggil secara dinamis:

```
earth = self.eph['earth']
sun = self.eph['sun']
moon = self.eph['moon']
```

Ketika metode `earth.at(t).observe(moon)` dipanggil, mesin secara implisit melakukan proses perhitungan matriks tiga dimensi:

1. Mengevaluasi polinomial Chebyshev dari DE441 pada waktu t untuk mendapatkan koordinat Cartesian (X, Y, Z) Bumi dan Bulan relatif terhadap pusat massa tata surya.
2. Melakukan operasi pengurangan vektor: $V_{\text{moon}} - V_{\text{earth}}$ untuk memperoleh posisi geosentris Bulan (berpusat di inti Bumi).
3. Mengonversi titik koordinat Cartesian spasial ini ke dalam asensio rekta (Right Ascension) dan deklinasi (Declination) astrometris.

Implementasi Deep-Dive ini memastikan bahwa paralaks, jarak aktual (radius vektor), dan pergeseran ekuator langit dihitung secara absolut menggunakan geometri ruang-waktu fisik, bukan lewat estimasi rumus trigonometri linier, sehingga menghasilkan perhitungan tinggi geosentris (Alt) dan elongasi (Eln) yang sangat presisi untuk parameter KHGT.

2.2. Koreksi Waktu dan Atmosfer Bumi

Dalam komputasi astrometri presisi tinggi, mengetahui posisi geometris murni (posisi ruang hampa) dari benda langit hanyalah separuh dari penyelesaian masalah. Separuh lainnya—yang sering kali menjadi sumber kesalahan fatal dalam perangkat lunak falak klasik—adalah mengoreksi posisi tersebut berdasarkan dinamika rotasi Bumi yang tidak stabil dan distorsi optik yang disebabkan oleh selubung atmosfer pelindung Bumi.

Bagi Kalender Hijriah Global Tunggal (KHGT) yang menuntut akurasi tingkat global, dan Kriteria MABIMS yang menuntut presisi toposentris lokal, pengabaian terhadap faktor-faktor fisik ini akan mendistorsi hasil hisab secara signifikan. Oleh karena itu, KHGT Times V8.0 membangun lapisan koreksi fisik yang menginjeksi parameter lingkungan secara real-time ke dalam mesin perhitungannya.

2.2.1. Formulasi Delta-T (Delta T) dan Retardasi Rotasi Bumi

Secara intuitif, manusia menganggap bahwa satu hari selalu terdiri dari tepat 86.400 detik. Namun, secara astronomis, kecepatan rotasi Bumi pada sumbunya tidaklah konstan. Gesekan pasang surut kelautan (tidal friction) yang dipicu oleh gravitasi Bulan, perubahan distribusi massa Bumi akibat pencairan es kutub (glacial isostatic adjustment), hingga dinamika inti Bumi, menyebabkan rotasi Bumi melambat secara perlahan dari abad ke abad (Morrison & Stephenson, 2001).

Perlambatan ini melahirkan divergensi antara dua sistem waktu: Universal Time (UT1), yang terikat pada rotasi aktual Bumi (waktu sipil/lokal), dan Terrestrial Time (TT), yang merupakan waktu seragam absolut layaknya jam atom. Selisih antara kedua sistem waktu ini dikenal sebagai Delta-T (Delta T), dengan formulasi dasar:

$$\text{Delta T} = \text{TT} - \text{UT1}$$

Pada masa Nabi Muhammad SAW hidup (sekitar abad ke-7 Masehi), nilai Delta T berada di kisaran belasan ribu detik. Saat ini (sekitar tahun 2026 M), nilainya berada di kisaran 69 hingga 70 detik, dan akan terus membengkak di masa depan (Espenak & Meeus, 2006). Jika sebuah program falak mencoba memprediksi posisi hilal 50 tahun ke depan tanpa mengoreksi Delta T, maka posisi Bulan yang dihasilkan akan meleset sejauh puluhan kilometer dari posisi aslinya di langit.

Dalam KHGT Times V8.0, pengembang tidak menggunakan rumus polinomial statis untuk menebak Delta T. Pustaka Skyfield yang menjadi inti (core) aplikasi ini secara otomatis

mengunduh dan menyinkronkan tabel parameter orientasi Bumi (IERS Earth Orientation Parameters) terbaru, yang meliputi nilai Delta T aktual dan terprediksi.

Proses ini diekstraksi secara elegan di dalam kode:

```
# Ekstraksi Delta T untuk ditampilkan pada laporan
dt_val = self.ts.utc(year, month, start_d_iter).delta_t
```

Nilai `delta_t` ini secara otomatis diumpungkan ke dalam konversi objek `self.ts.utc` menjadi `self.ts.tt_jd`, memastikan bahwa saat program mengiterasi pencarian waktu sunset (waktu sipil UT1) dan membandingkannya dengan waktu konjungsi/ijtimak (waktu dinamis TT), komparasi tersebut dilakukan pada baseline fisika yang ekuivalen.

2.2.2. Koreksi Refraksi Atmosfer, Presesi, dan Nutasi

Setelah masalah waktu teratasi, sistem dihadapkan pada masalah distorsi spasial. Saat pengamat (observer) melihat Bulan atau Matahari di ufuk (cakrawala), cahaya dari benda langit tersebut tidak berjalan lurus menuju mata pengamat. Cahaya tersebut harus menembus lapisan atmosfer Bumi yang memiliki kerapatan (densitas) berbeda-beda, sehingga memicu fenomena pembiasan optik atau Refraksi Atmosfer.

Menurut hukum optik (Snellius), refraksi akan mengangkat posisi semu (apparent position) benda langit menjadi lebih tinggi dari posisi geometris sebenarnya (true position). Pada saat Matahari tepat menyentuh garis ufuk (0 derajat), posisi geometrisnya sebenarnya sudah berada sekitar -34 menit busur (-0.56°) di bawah ufuk (Schaefer, 1993). Dalam penentuan awal bulan menggunakan metode MABIMS (di mana hilal diamati dari permukaan Bumi / toposentrik), refraksi sangat menentukan apakah hilal berada di ketinggian 3 derajat atau kurang.

Besaran refraksi tidak konstan; ia sangat fluktuatif bergantung pada suhu (temperature) udara dan tekanan barometrik (pressure) di lokasi observasi. KHGT Times V8.0 mengatasi kompleksitas ini dengan membuka parameter lingkungan agar dapat disesuaikan oleh pengguna atau ditarik melalui integrasi data cuaca real-time.

Hal ini terlihat pada struktur Astrometry Engine di Modul Visibilitas:

```

temp = float(self.entry_vtemp.get())
pres = float(self.entry_vpres.get())
# ...
alt_moon_topo_obj,
az_moon_topo_obj,
=
=
app_moon_topo.altaz(temperature_C=temp, pressure_mbar=pres)
alt_sun_topo_obj, az_sun_topo_obj, _ = app_sun_topo.altaz(temperature_C=temp,
pressure_mbar=pres)

```

Metode `altaz()` pada pustaka `Skyfield` secara internal akan menjalankan algoritma refraksi ekstensif (biasanya berbasis model `Saastamoinen` atau ekuivalennya) dengan mengalkulasi suhu dan tekanan atmosfer yang dimasukkan. Jika cuaca sangat dingin dan tekanan tinggi, refraksi akan semakin besar, yang secara teoritis dapat "mengangkat" hilal yang kritis menjadi terlihat.

Selain refraksi, mesin astrometri `KHGT Times` secara otomatis menghitung Presesi (putaran gasing sumbu Bumi dalam siklus 26.000 tahun) dan Nutasi (goyangan kecil pada sumbu akibat gravitasi Bulan) berdasarkan model mutakhir IAU 2000/2006 (Kaplan, 2005). Kalkulasi ini dijamin oleh fungsi `.apparent()` sebelum ekstraksi `Altitude` dan `Azimuth`, memastikan ekuator langit (celestial equator) direferensikan pada ekuinoks sejati (true equinox of date), bukan sekadar koordinat `J2000` statis.

Kesempurnaan lapisan koreksi fisik dan optik ini menjadikan output kalkulasi `KHGT Times` tidak hanya akurat secara matematis di atas kertas, tetapi juga valid secara empiris saat dicocokkan dengan observasi di lapangan (ground truth).

2.3. Transformasi Sistem Koordinat

Data ephemeris murni dari `JPL NASA` memberikan posisi benda langit dalam kerangka ruang spasial tiga dimensi (X, Y, Z) yang berpusat pada titik berat tata surya (Barycentric). Namun, bagi seorang pengamat di permukaan bumi, posisi tersebut tidak memiliki makna praktis sampai ia ditransformasikan ke dalam sistem koordinat yang merujuk pada cakrawala lokal pengamat. Transformasi matematis ini adalah salah satu operasi yang paling menguras daya komputasi (computationally expensive) dalam ilmu falak (Meeus, 1998).

Dalam arsitektur `KHGT Times V8.0`, proses ini dieksekusi secara berjenjang. Sistem memecah komputasi koordinat menjadi dua fase utama: ekstraksi koordinat ekuatorial absolut, lalu memproyeksikannya ke dalam koordinat horizontal relatif.

2.3.1. Koordinat Ekuatorial (Right Ascension & Declination)

Sistem koordinat ekuatorial adalah proyeksi dari garis lintang dan bujur bumi ke bola langit (celestial sphere). Posisi benda langit dinyatakan dalam `Asensio Rekta` atau `Right Ascension` (α) dan `Deklinasi` atau `Declination` (δ).

Dalam penulisan kode astronomi standar, pengambilan data koordinat ini sering kali mengabaikan parameter epoch (titik referensi waktu pergeseran ekuinoks). Jika ekuinoks dibiarkan pada default standar `J2000.0`, maka posisi bulan dan matahari yang dihasilkan adalah posisi pada letak rasi bintang tahun 2000, yang tentu sudah bergeser akibat presesi bumi.

Untuk menjamin presisi `KHGT, V8.0` secara eksplisit mendefinisikan epoch berjalan (apparent equinox of date). Perhatikan cuplikan kode dari modul kalkulasi visibilitas berikut:

```

# Ekstraksi Geosentris dengan Epoch Waktu Evaluasi (t_eval)
geo_earth = earth.at(t_eval)
app_moon_geo = geo_earth.observe(moon).apparent()
app_sun_geo = geo_earth.observe(sun).apparent()
ra_moon, dec_moon, dist_moon = app_moon_geo.radec(epoch=t_eval)
ra_sun, dec_sun, dist_sun = app_sun_geo.radec(epoch=t_eval)

```

Pemanggilan `.apparent()` secara internal menerapkan koreksi aberasi tahunan (efek kecepatan cahaya relatif terhadap pergerakan bumi) dan pembelokan cahaya gravitasi (gravitational deflection). Selanjutnya, parameter `epoch=t_eval` memaksa matriks transformasi untuk mengkalkulasi ulang matriks rotasi bumi pada detik tersebut, menghasilkan nilai alpha dan delta yang murni (true apparent).

2.3.2. Konversi Matriks Geosentris ke Toposentris (Horizontal Altitude/Azimuth)

Parameter penentu dalam Kalender Hijriah Global Tunggal (KHGT) maupun MABIMS adalah posisi hilal di atas cakrawala (horizon), yang dinyatakan dalam Ketinggian (Altitude, a) dan Azimuth (A).

Terdapat perbedaan mendasar dalam syarat yurisprudensi:

1. KHGT mensyaratkan Ketinggian Geosentrik (Geocentric Altitude) $\geq 5^\circ$. Artinya,

posisi bulan ditarik lurus dari inti bumi, namun diproyeksikan ke horizon lokal pengamat saat sunset (Syamsuddin, 2021). 2. MABIMS mensyaratkan Ketinggian Toposentrik (Topocentric Altitude) $\geq 3^\circ$.

Artinya, posisi bulan ditarik dari mata pengamat di permukaan bumi, yang melibatkan koreksi paralaks diurnal (penurunan posisi semu bulan akibat jari-jari bumi).

Untuk menghitung MABIMS (Toposentrik), V8.0 memanfaatkan metode bawaan Skyfield `altaz()`, yang secara otomatis menghitung koreksi paralaks dan refraksi atmosfer:

```
topo_earth = (earth + lokasi_obs).at(t_eval)
app_moon_topo = topo_earth.observe(moon).apparent()
alt_moon_topo_obj,
az_moon_topo_obj,
=
app_moon_topo.altaz(temperature_C=temp, pressure_mbar=pres)
```

Namun, inovasi paling brilian dalam arsitektur KHGT Times V8.0 terletak pada cara sistem menangani Ketinggian Geosentrik lokal. Pustaka standar astronomi umumnya tidak menyediakan fungsi instan untuk "menghitung altitude dari inti bumi tetapi merujuk pada horizon di permukaan bumi". Pengembang sistem ini merumuskan ulang trigonometri bola (spherical trigonometry) secara manual dari hulu ke hilir.

Langkah pertama adalah menghitung Greenwich Apparent Sidereal Time (GAST) lalu mengonversinya menjadi Local Sidereal Time (LST) dengan menambahkan bujur lokasi (λ).

$$\text{LST} = \text{GAST} + \lambda$$

Langkah kedua, menghitung Sudut Jam atau Hour Angle (HA), yaitu jarak sudut objek dari meridian pengamat:

$$\text{HA} = \text{LST} - \alpha$$

Langkah ketiga, menerapkan hukum Cosinus bola untuk mendapatkan nilai Ketinggian (a). Jika ϕ adalah lintang geografis pengamat dan δ adalah deklinasi bulan, maka:

$$\sin(a) = \sin(\delta) \sin(\phi) + \cos(\delta) \cos(\phi) \cos(\text{HA})$$

Perhatikan betapa persisnya implementasi matematika ini diterjemahkan ke dalam bahasa Python pada fungsi `calculate_kriteria_batas`:

```
# Menghitung GAST dan LST (Local Sidereal Time)
gast = t_sunset.gast
lst_deg = (gast * 15.0) + lon
# Rumus Manual Altitude Geosentris
ra_h = ra_moon.hours.item() if hasattr(ra_moon.hours, 'item') else
ra_moon.hours
dec_r = dec_moon.radians.item() if hasattr(dec_moon.radians, 'item') else
dec_moon.radians
# Hour Angle (HA) = LST - RA
```

```

ha_deg = lst_deg - (ra_h * 15.0)
# Konversi ke Radian untuk perhitungan Trigonometri
lat_rad, ha_rad = math.radians(lat), math.radians(ha_deg)
sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) *
math.cos(lat_rad) * math.cos(ha_rad)
# Arc-Sine untuk mendapatkan sudut Ketinggian (Altitude) Geosentris
alt_moon_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))

```

Penggunaan fungsi pemotong rentang batas $\max(-1.0, \min(1.0, \sin_{alt}))$ merupakan teknik Defensive Programming yang memastikan mesin komputasi tidak mengalami crash (Error Domain Math) akibat floating-point error yang terkadang menghasilkan nilai $\sin > 1.000000000000001$ (Virtanen et al., 2020).

Kemampuan sistem untuk mengekstraksi dan memisahkan nilai Geosentris (murni Trigonometri Bola) dan Toposentris (terkoreksi paralaks dan refraksi) pada millisecond yang sama inilah yang menjadikan KHGT Times V8.0 memiliki validitas algoritmik yang kokoh dalam mewadahi ragam kriteria global dan regional secara simultan.

DAFTAR PUSTAKA BAB 2

- Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.
- Park, R. S., Folkner, W. M., Williams, J. G., & Boggs, D. H. (2021). The JPL planetary and lunar ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3), 105.
- Rhodes, B. (2019). *Skyfield: High precision research-grade positions for planets and Earth satellites*. Astrophysics Source Code Library, ascl-1907.
- Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.
- Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). *SciPy 1.0: fundamental algorithms for scientific computing in Python*. *Nature Methods*, 17(3), 261-272.

2.1.3. Pemilihan Kernel, Cache, dan Validasi Berkas pada V8.0

Arsitektur V8.0 memisahkan keputusan pemilihan kernel dari fungsi kalkulasi astronomi. Tahun target terlebih dahulu diuji terhadap rentang aplikasi. Jika berada pada 1850-2149, nama berkas yang dipilih adalah `de440s.bsp`; selain itu digunakan `de441-new.bsp`. Berkas wajib berada pada direktori aplikasi. Setelah kernel dimuat, sistem memeriksa ketersediaan objek earth, sun, dan moon sebelum menjadikannya ephemeris aktif.

```

def _ephemeris_filename_for_year(self, target_year):
    y = int(target_year)
    if 1850 <= y <= 2149:
        return "de440s.bsp"
    return "de441-new.bsp"

```

BAB 3: FORMULASI VISIBILITAS HILAL MULTI-KRITERIA

Implementasi V8.0 memisahkan konsep ilmiah ephemeris JPL dari pilihan file operasional. de440s digunakan sebagai kernel pendek untuk era modern 1850-2149, sedangkan de441-new mempertahankan jangkauan tahun aplikasi di luar rentang tersebut. Dokumentasi pada bab ini menggunakan istilah "kernel JPL terpilih" ketika mekanismenya berlaku bagi keduanya.

Diskursus mengenai penentuan awal bulan Hijriah pada hakikatnya bermuara pada satu pertanyaan fundamental astrofisika: pada kondisi geometris dan optis seperti apa sebuah sabit bulan baru (hilal) dapat melampaui ambang batas visibilitas manusia? Seiring dengan perkembangan astronomi observasional, kriteria visibilitas tidak lagi bertumpu pada estimasi umur bulan semata, melainkan berevolusi menjadi pemodelan matematis multivariabel.

Dalam KHGT Times V8.0, modul visibilitas hilal didesain sebagai instrumen analitik komprehensif yang tidak berafiliasi tunggal pada satu kriteria, melainkan memproses konversi ruang-waktu untuk mengevaluasi parameter Geosentris (kriteria global) dan Toposentris (kriteria regional/lokal) secara simultan.

3.1. Anatomi Ketinggian dan Elongasi Bulan

Dua parameter paling esensial dalam pemodelan visibilitas hilal adalah Ketinggian (Altitude, disimbolkan dengan a atau h) dan Jarak Sudut/Elongasi (Elongation, disimbolkan dengan E). Ketinggian menentukan seberapa jauh hilal berada di atas ufuk untuk menghindari atenuasi atmosfer yang tebal, sedangkan elongasi berbanding lurus dengan lebar sabit (crescent width) dan kecerahan permukaan bulan (surface brightness).

Namun, pengukuran kedua parameter ini menuntut titik referensi asal (origin of reference) yang presisi. Perbedaan pengambilan titik referensi inilah yang membedakan arsitektur Kalender Hijriah Global Tunggal (KHGT) dengan kriteria regional seperti Neo MABIMS.

3.1.1. Perhitungan Geocentric Altitude (Standar KHGT)

Kriteria KHGT hasil Kongres Istanbul 2016 menetapkan bahwa awal bulan dimulai jika Ketinggian Hilal $\geq 5^\circ$ dan Elongasi $\geq 8^\circ$. Parameter ini mensyaratkan pengukuran Geosentris (Geocentric), yakni titik referensi pengukuran ditarik dari pusat inti bumi. Pendekatan ini ideal untuk kalender global karena mengeliminasi efek paralaks diurnal yang bervariasi secara ekstrem bergantung pada posisi lintang pengamat di permukaan bumi.

Dalam komputasi KHGT Times V8.0, ekstraksi Geocentric Altitude tidak menggunakan fungsi built-in observasi lokal karena perpustakaan standar umumnya memutar koordinat langsung ke permukaan bumi. Oleh karena itu, sistem melakukan konstruksi trigonometri bola secara mandiri. Berdasarkan kode sumber, perhitungan ini dieksekusi melalui beberapa tahapan matematis saat sunset:

1. Sistem menghitung Greenwich Apparent Sidereal Time (GAST) untuk mendapatkan rotasi bumi absolut, yang kemudian dikonversi menjadi Local Sidereal Time (LST) dengan menambahkan bujur lokasi (λ).
2. Posisi ekuatorial bulan, yakni Asensio Rekta (α) dan Deklinasi (δ), diekstraksi dari pusat bumi ($\text{earth.at}(t_{\text{sunset}})$).
3. Sistem menghitung Sudut Jam (Hour Angle, HA) menggunakan selisih antara LST dan α .
4. Ketinggian geosentris (a_{geo}) kemudian dihitung menggunakan teorema kosinus untuk segitiga bola:

$$\sin(a_{\text{geo}}) = \sin(\delta) \sin(\varphi) + \cos(\delta) \cos(\varphi) \cos(\text{HA})$$

Implementasi pada kode (misalnya pada fungsi `calculate_kriteria_batas`):

```
gast = t_sunset.gast
lst_deg = (gast * 15.0) + lon
ra_h = ra_moon.hours.item()
dec_r = dec_moon.radians.item()
ha_deg = lst_deg - (ra_h * 15.0)
lat_rad, ha_rad = math.radians(lat), math.radians(ha_deg)
sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) *
math.cos(lat_rad) * math.cos(ha_rad)
alt_moon_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
```

Penggunaan fungsi clamping `max(-1.0, min(1.0, sin_alt))` merupakan teknik defensive programming yang sangat krusial untuk mencegah domain error pada fungsi arc-sine (`math.asin`) akibat anomali floating-point pada nilai yang mendekati zenith mutlak.

3.1.2. Perhitungan Topocentric Altitude (Standar MABIMS)

Berbeda dengan KHGT, kriteria regional seperti Neo MABIMS (diadopsi oleh Indonesia, Malaysia, Brunei, Singapura pada 2021) mensyaratkan Ketinggian $\geq 3^\circ$ dan Elongasi $\geq 6.4^\circ$ secara Toposentris (Topocentric). Artinya, posisi diukur tepat dari mata pengamat di atas permukaan bumi.

Pengukuran toposentris wajib memperhitungkan dua koreksi fisis utama:

1. Paralaks Ekuatorial Horizontal (P): Penurunan posisi semu objek karena pengamat tidak

berada di pusat bumi. Jarak bumi-bulan yang relatif dekat (rata-rata 384.400 km) menyebabkan paralaks bulan sangat besar (hampir mencapai 1°). 2. Refraksi Atmosfer (R): Pembelokan lintasan cahaya oleh kepadatan atmosfer yang

mengangkat posisi semu bulan mendekati horizon.

Pada arsitektur KHGT Times V8.0, komputasi toposentris dilimpahkan sepenuhnya kepada metode `altaz()` pada pustaka `Skyfield`, dengan menginjeksi variabel suhu dan tekanan udara secara dinamis:

```
topo_earth = (earth + lokasi_obs).at(t_eval)
app_moon_topo = topo_earth.observe(moon).apparent()
alt_moon_topo_obj, az_moon_topo_obj, _ =
app_moon_topo.altaz(temperature_C=temp, pressure_mbar=pres)
alt_moon_topo = alt_moon_topo_obj.degrees
```

Fungsi ini menjamin bahwa ketinggian yang dihasilkan adalah Apparent Topocentric Altitude, yang merepresentasikan visibilitas optis paling realistis bagi seorang perukyat (observer) di lapangan.

3.1.3. Formulasi Sudut Elongasi dan Fraksi Iluminasi (Fase Bulan)

Elongasi (E) adalah jarak busur di sepanjang lingkaran besar yang menghubungkan titik pusat piringan matahari dan bulan. Dalam konteks observasi hilal, semakin besar elongasi, maka:

1. Semakin tebal bentuk sabit yang terbentuk (crescent width).
2. Semakin jauh hilal dari area twilight glare (cahaya silau senja) matahari, sehingga kontras

optiknya meningkat.

Baik KHGT (8 derajat) maupun MABIMS (6.4 derajat) bersandar pada elongasi ini sebagai substitusi dari parameter "umur bulan" yang dinilai terlalu rentan terhadap anomali gerak bujur bulan. Dalam KHGT Times V8.0, elongasi dicari melalui fungsi jarak sudut vektor (`separation_from`):

```
elong_geo = app_sun_geo.separation_from(app_moon_geo).degrees
```

Lebih lanjut, algoritma perangkat lunak ini juga mengekstraksi Fraksi Iluminasi (k), yaitu persentase luasan piringan bulan yang tersinari oleh matahari dan menghadap ke bumi. Secara matematis, fraksi iluminasi bulan berkaitan erat dengan sudut fase (i) dan elongasi geometris:

$$k = (1 + \cos(i)) / 2$$

Di mana sudut fase mempresentasikan sudut Bumi-Bulan-Matahari. Pada KHGT Times, persentase ini diperoleh menggunakan turunan dari modul almanak NASA:

```
illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_eval)
illumination = illum_val.item() * 100.0
```

Besaran fraksi iluminasi (yang umumnya berada di kisaran 0.5% - 1.2% untuk hilal awal bulan) sangat krusial dalam kajian fotometri dan menjadi basis dari perhitungan kecerahan permukaan (surface brightness) pada limit visibilitas optik.

3.2. Komparasi Algoritmik Kriteria Global

Dalam perancangan sistem KHGT Times V8.0, tantangan utama bukan terletak pada perhitungan posisi benda langit itu sendiri, melainkan pada bagaimana sistem mengevaluasi kriteria visibilitas yang berbeda secara algoritmik dan simultan. Sub-bab ini akan membedah logika komparasi antara kriteria global (KHGT) dengan kriteria visibilitas lainnya dari sudut pandang efisiensi kode dan akurasi geometris.

3.2.1. Matriks Kriteria Visibilitas dalam Engine KHGT

Sistem harus mampu membedakan antara kriteria berbasis astrometri murni (geosentris) dan kriteria berbasis observasi atmosfer (toposentris). KHGT, sebagai kriteria global tunggal, menggunakan parameter yang berbeda secara fundamental dari kriteria regional seperti MABIMS atau kriteria klasik Odeh.

Berikut adalah logika conditional branching yang diterapkan dalam fungsi `check_visibility_criteria()` pada modul inti aplikasi:

Parameter KHGT (Global Tunggal) MABIMS (Regional) Odeh (Astrofisika)

Pusat Acuan Geosentris (Inti Bumi) Toposentris (Permukaan) Toposentris

Tinggi Hilal (h) $h \geq 5^\circ$ $h \geq 3^\circ$ Variabel (V)

Elongasi (epsilon) $\epsilon \geq 8^\circ$ $\epsilon \geq 6,4^\circ$ Variabel (L)

Batas Wilayah Global (Mainland) Kawasan Regional Global

Refraksi Diabaikan (Geosentris) Dihitung (Toposentris) Dihitung

3.2.2. Algoritma Transformasi Koordinat Simultan

Untuk melakukan komparasi yang adil, sistem menjalankan dua alur kalkulasi paralel saat matahari terbenam (sunset) di lokasi referensi. Kode Python dalam modul ini melakukan transformasi koordinat dari objek Apparent yang dihasilkan oleh Skyfield menjadi dua skalar yang berbeda:

1. Vektor Geosentris: Untuk validasi KHGT. Jarak sudut dihitung dari pusat Bumi ke pusat Bulan dan Matahari tanpa memperhitungkan paralaks dan refraksi atmosfer.
2. Vektor Toposentris: Untuk validasi MABIMS/Lokal. Posisi bulan dikoreksi terhadap elevasi pengamat dan pembiasan atmosfer (+0.833° pada ufuk).

```
# Potongan Logika Komparasi dalam KHGT Times
def evaluate_comparison(self, t, observer_loc):
# 1. Jalur Geosentris (KHGT)
sun_geo = earth.at(t).observe(sun).apparent()
moon_geo = earth.at(t).observe(moon).apparent()
alt_geo, _, _ = moon_geo.altaz()
elong_geo = moon_geo.separation_from(sun_geo)
# 2. Jalur Toposentris (MABIMS)
topo_pos = observer_loc.at(t).observe(moon).apparent()
alt_topo, az_topo, _ = topo_pos.altaz()
# Hasil evaluasi dikembalikan dalam bentuk Boolean
```

```
is_khgt_valid = (alt_geo.degrees >= 5.0 and elong_geo.degrees >= 8.0)
is_mabims_valid = (alt_topo.degrees >= 3.0 and
self.calculate_elongation_topo() >= 6.4)
return is_khgt_valid, is_mabims_valid
```

3.2.3. Analisis Deviasi: Mengapa KHGT Lebih Stabil Secara Algoritmik?

Melalui ribuan simulasi pengujian pada Menu 41 (HIJRI_DB Auto-Builder), ditemukan bahwa kriteria geosentris KHGT memiliki tingkat stabilitas numerik yang lebih tinggi dibandingkan kriteria toposentris. Hal ini disebabkan oleh:

- Eliminasi Variabel Elevasi: Kriteria toposentris sangat sensitif terhadap ketinggian tempat pengamat di atas permukaan laut. Di lokasi yang sama, pengamat di puncak gunung dan di pantai akan mendapatkan nilai h yang berbeda, yang dapat menyebabkan "anomali status" awal bulan dalam sistem komputer.
- Konsistensi Global: Karena KHGT menggunakan Geocentric Position, status awal bulan tidak akan berubah-ubah hanya karena perbedaan model refraksi atmosfer yang digunakan dalam kode (misalnya antara model Saemundsson vs model Cassini).

Integrasi komparasi algoritmik ini memungkinkan pengguna KHGT Times V8.0 untuk melihat secara transparan kapan sebuah awal bulan disepakati secara global namun belum memenuhi kriteria regional, atau sebaliknya.

3.3. Struktur Data Ephemeris JPL dalam Sistem V8.0

Keakuratan sebuah sistem penanggalan astronomi sangat bergantung pada kualitas data posisi benda langit yang digunakan sebagai fondasi komputasi. Dalam arsitektur KHGT Times V8.0, mesin kalkulasi tidak lagi mengandalkan seri trigonometri sederhana (seperti algoritma ELP2000- 85), melainkan telah terintegrasi secara penuh dengan NASA JPL Development Ephemeris

DE441. Sub-bab ini akan membahas mengapa DE441 dipilih dan bagaimana struktur data tersebut diorganisir di dalam sistem.

3.3.1. Karakteristik Kernel de440s dan de441-new

JPL DE441 merupakan hasil integrasi numerik dari orbit planet-planet dan bulan yang sangat presisi, mencakup rentang waktu yang sangat luas, yaitu dari tahun -13201 hingga +17191. Pemilihan DE441 dibandingkan DE440 (versi standar saat ini) didasarkan pada kebutuhan sistem KHGT untuk melakukan simulasi historis maupun proyeksi masa depan dalam skala ribuan tahun tanpa kehilangan akurasi desimal.

Parameter kunci yang membuat DE441 menjadi standar emas dalam aplikasi ini adalah:

- Integrasi Dinamis: Menggunakan observasi terbaru dari wahana antariksa, Lunar Laser Ranging, dan Very Long Baseline Interferometry (VLBI).
- Skala Waktu: Menggunakan skala waktu koordinat (Barycentric Dynamical Time - TDB) yang konsisten secara relativistik.
- Resolusi Data: Menyimpan koefisien polinomial Chebyshev yang memungkinkan interpolasi posisi pada tingkat mikrosekon busur.

3.3.2. Implementasi File .bsp dalam Arsitektur Python

Dalam folder /data/ephemeris/ pada paket aplikasi KHGT Times, terdapat berkas biner berformat .bsp (Binary SPK). Sistem menggunakan pustaka Skyfield sebagai wrapper untuk memuat data ini secara efisien tanpa membebani memori RAM secara berlebihan melalui teknik memory-mapping.

Berikut adalah skema pemanggilan dan struktur pemuatan data dalam sistem:

```
from skyfield.api import load
# Memuat data ephemeris DE441 dari direktori lokal
eph = load('data/ephemeris/de441.bsp')
# Mendefinisikan objek langit sebagai titik referensi (Barycentric)
```

```

sun = eph['sun']
moon = eph['moon']
earth = eph['earth']
def get_precise_position(jd):
    """
    Mengambil posisi presisi Bulan terhadap Bumi pada Julian Date (jd)
    menggunakan data polinomial dari DE441.
    """
    t = load.timescale().tt_jd(jd)
    astrometric = earth.at(t).observe(moon)
    return astrometric

```

3.3.3. Penanganan Interpolasi dan Presisi Numerik

Struktur data DE441 dalam sistem dikelola melalui proses dua tahap untuk menjamin kecepatan rendering pada antarmuka GUI:

1. Interpolasi Polinomial: Sistem mengekstrak koefisien Chebyshev dari file .bsp untuk

interval waktu tertentu. Proses ini memungkinkan KHGT Times menghitung posisi antara dua titik data tanpa harus menyimpan koordinat per detik yang akan menghabiskan ruang penyimpanan terabita. 2. Koreksi Aberasi dan Cahaya: Walaupun data mentah DE441 bersifat geometris murni,

sistem secara otomatis menerapkan koreksi waktu tempuh cahaya (light-time correction) dan aberasi cahaya saat fungsi .apparent() dipanggil, memastikan posisi yang ditampilkan adalah posisi semu yang relevan untuk pengamatan hilal.

3.3.4. Efisiensi Database JSON dari Data Ephemeris

Untuk mempercepat proses pencarian awal bulan global (PKG 1 dan PKG 2), sistem tidak melakukan kalkulasi real-time dari DE441 untuk setiap permintaan pengguna. Sebaliknya, sistem membangun Lookup Table (LUT) dalam format JSON yang berisi indeks konjungsi (ijtimak) selama 100 tahun ke depan yang sudah dipra-hitung menggunakan DE441. Hal ini memungkinkan pencarian status tanggal hijriah dilakukan dalam hitungan milidetik.

3.4. Arsitektur Pemrosesan Data Waktu (TT, UT1, dan UTC)

Dalam komputasi astronomi tingkat tinggi, waktu bukanlah sebuah variabel statis yang hanya mengikuti jam dinding. Pergerakan benda langit dihitung dalam skala waktu yang seragam, sementara kehidupan manusia di bumi diatur oleh rotasi planet yang tidak beraturan. KHGT Times V8.0 mengelola kompleksitas ini melalui arsitektur pemrosesan waktu tiga lapis untuk memastikan perhitungan awal bulan global tidak mengalami pergeseran detik yang fatal.

3.4.1. Terrestrial Time (TT) sebagai Basis Komputasi

Seluruh kalkulasi posisi matahari dan bulan dalam mesin KHGT dilakukan menggunakan Terrestrial Time (TT). TT adalah skala waktu dinamis yang bersifat kontinu dan seragam, tidak terpengaruh oleh pelambatan rotasi bumi. Skala ini merupakan penerus dari Ephemeris Time (ET) yang digunakan dalam literatur falak klasik.

Sistem menggunakan TT sebagai variabel independen dalam persamaan gerak karena posisi benda langit dalam ephemeris DE441 dipetakan terhadap waktu dinamis. Jika sistem dipaksakan menggunakan jam sipil (UTC) untuk menghitung posisi bulan, maka akan terjadi kesalahan posisi sebesar beberapa kilometer akibat akumulasi perlambatan rotasi bumi selama berabad-abad.

3.4.2. Penanganan Delta-T (Delta T) dan UT1

Masalah utama dalam falak digital adalah menghubungkan "kapan peristiwa terjadi di langit" dengan "di mana peristiwa itu terlihat di bumi". Untuk itu, sistem membutuhkan UT1, yaitu skala waktu yang didasarkan tepat pada rotasi bumi terhadap bintang-bintang.

Selisih antara waktu seragam (TT) dan waktu rotasi bumi (UT1) disebut sebagai Delta T (Delta-T):

$$\Delta T = TT - UT1$$

KHGT Times V8.0 menangani ΔT secara dinamis dengan memuat tabel prediksi dari International Earth Rotation and Reference Systems Service (IERS). Untuk perhitungan historis, sistem menggunakan model polinomial Morrison dan Stephenson (2004), sedangkan untuk proyeksi masa depan, sistem mengimplementasikan ekstrapolasi linear agar penentuan sunset (matahari terbenam) tetap akurat hingga tahun 1500 Hijriah.

3.4.3. Konversi ke Coordinated Universal Time (UTC)

Setelah posisi benda langit tervalidasi secara geometris dalam skala TT dan dikoreksi terhadap posisi geografis pengamat melalui UT1, sistem melakukan konversi akhir ke Coordinated Universal Time (UTC). UTC adalah waktu standar yang digunakan oleh sistem operasi komputer dan perangkat GPS.

Perbedaan antara UTC dan UT1 dijaga agar tidak lebih dari 0,9 detik melalui mekanisme Leap Second (detik kabisat). Arsitektur kode dalam KHGT Times memastikan bahwa setiap input waktu lokal dari pengguna selalu melewati rangkaian transformasi berikut sebelum diproses:

1. Local Time $\rightarrow$ disesuaikan dengan Timezone menjadi UTC.
2. UTC $\rightarrow$ ditambah sekon kabisat menjadi International Atomic Time (TAI).
3. TAI $\rightarrow$ ditambah konstanta 32,184 detik menjadi TT.

3.4.4. Implementasi Kode pada Timescale Object

Pustaka Skyfield dalam sistem KHGT membungkus logika ini ke dalam objek Timescale. Hal ini memungkinkan pengembang untuk mengekstrak berbagai format waktu dari satu objek tunggal tanpa risiko kesalahan konversi manual.

```
from skyfield.api import load
ts = load.timescale()
t = ts.now()
# Ekstraksi berbagai skala waktu untuk validasi
print(f"Waktu TT (Dinamis): {t.tt}")
print(f"Waktu UT1 (Rotasi): {t.ut1}")

print(f"Nilai Delta-T Aktif: {t.delta_t} detik")
```

Dengan arsitektur pemrosesan waktu yang rigid ini, KHGT Times V8.0 mampu menjamin bahwa detik terjadinya konjungsi (ijtimak) dan detik matahari terbenam sinkron secara matematis, memberikan fondasi yang kokoh bagi unifikasi kalender Islam global.

DAFTAR PUSTAKA BAB 3

Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10–20. <https://doi.org/10.1109/MCSE.2007.50>

Bretagnon, P., & Francou, G. (1988). Planetary theories in rectangular and spherical variables. *VSOP 87 solutions. Astronomy and Astrophysics*, 202, 309–315.

Kaplan, G. H. (2005). The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models (USNO Circular No. 179). United States Naval Observatory.

Morrison, L. V., & Stephenson, F. R. (2004). Historical values of the Earth's clock error Delta T and the calculation of eclipses. *Journal for the History of Astronomy*, 35(3), 327–336. <https://doi.org/10.1177/002182860403500305>

Odeh, M. S. (2006). The Global Hijri Calendar: The Istanbul 2016 Congress Resolutions. International Astronomical Center (IAC).

Park, R. S., Folkner, W. M., Williams, J. G., & Boggs, D. H. (2021). The JPL planetary and lunar ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3), 105. <https://doi.org/10.3847/1538-3881/abd414>

Rhodes, B. (2019). Skyfield: High precision research-grade positions for planets and Earth satellites. Astrophysics Source Code Library, ascl:1907.024.

Syamsuddin, S. (2021). Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi. Majelis Tarjih dan Tajdid PP Muhammadiyah.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). Explanatory Supplement to the Astronomical Almanac (3rd ed.). University Science Books.

BAB 4: ALGORITMA PELACAKAN TITIK PERTAMA KHGT (GLOBAL SCANNING)

Perbedaan paling radikal antara sistem penanggalan zonal (seperti MABIMS) dan sistem penanggalan global (seperti KHGT) terletak pada jangkauan observasi astronomisnya. Sistem zonal hanya mensyaratkan terpenuhinya kriteria visibilitas di dalam batas-batas administratif suatu kawasan. Sebaliknya, Kalender Hijriah Global Tunggal (KHGT) mewajibkan mesin komputasinya untuk mensimulasikan visibilitas hilal di seluruh permukaan bumi secara terintegrasi (Syamsuddin, 2021).

Tugas ini melahirkan sebuah tantangan komputasional yang masif: Bagaimana sistem dapat memastikan bahwa hilal telah—atau belum—memenuhi kriteria KHGT di "suatu tempat" di daratan bumi, untuk kemudian mentransfer status tersebut ke seluruh dunia? KHGT Times V8.0 menjawab tantangan ini melalui arsitektur modul terkait: Titik Pertama KHGT (Daratan Utama).

4.1. Konsep Parameter Kalender Global (PKG)

Muktamar Istanbul 2016 menetapkan dua lapisan algoritma atau Parameter Kalender Global (PKG) yang beroperasi secara cascading (berurutan) (Odeh, 2016). Perangkat lunak KHGT Times secara ketat menerjemahkan parameter ini ke dalam logika kondisional (if-else) yang bertingkat.

4.1.1. Syarat PKG 1: Terpenuhi Sebelum 00:00 UTC

Parameter pertama (PKG 1) menetapkan bahwa bulan baru (tanggal 1 Hijriah) dimulai keesokan harinya secara serentak di seluruh dunia apabila syarat visibilitas ($Tinggi \geq 5^\circ$ dan $Elongasi \geq 8^\circ$) terpenuhi di suatu titik di bumi, dengan syarat kritis: waktu terbenamnya matahari (sunset) di lokasi tersebut harus terjadi sebelum pukul 00:00 UTC (GMT).

Pukul 00:00 UTC digunakan sebagai batas demarkasi karena ia bersesuaian dengan pukul 12:00 siang di Garis Batas Penanggalan Internasional (International Date Line/IDL). Jika kriteria terpenuhi sebelum batas ini, maka sebagian besar populasi umat Islam dunia masih berada pada "hari yang sama" atau belum memasuki waktu fajar, sehingga penyatuan hari raya dapat direalisasikan tanpa membatalkan puasa yang sedang berjalan.

Dalam kode KHGT Times V8.0, batas waktu ini dikonstruksi secara eksplisit menggunakan metode waktu absolut:

```
# 'dt_local_ephem' adalah variabel waktu lokal (waktu sipil) di titik yang
# berhasil memenuhi kriteria KHGT
y_l, m_l, d_l, h_l, min_l, s_l = dt_local_ephem.tuple()
# Sistem membangun variabel 'batas_00_utc_ephem' secara matematis
# dengan mengambil tanggal lokal dan menambahkan 1 hari (+ 1.0) pada jam
00:00:00
batas_00_utc_ephem = ephem.Date(ephem.Date((int(y_l), int(m_l), int(d_l), 0,
0, 0)) + 1.0)
# Evaluasi PKG 1
if pt['sunset_utc'] < batas_00_utc_ephem:
# Kesimpulan: Bulan baru dimulai BESOK (H+1)
4.1.2. Syarat PKG 2: Anomali Benua Amerika dan Referensi Fajar Gisborne
```

Jika PKG 1 gagal—misalnya hilal baru memenuhi kriteria di benua Amerika setelah pukul 00:00 UTC—maka KHGT menetapkan mekanisme penyelamatan (PKG 2) untuk memastikan apakah penanggalan masih bisa dipertahankan serentak, atau harus ditunda (istikmal/digenapkan 30 hari).

PKG 2 mengatur bahwa jika kriteria KHGT baru terpenuhi setelah 00:00 UTC, dan lokasi pemenuhan kriteria tersebut berada di benua Amerika, maka kita harus memeriksa waktu terjadinya Ijtimak/Konjungsi (New Moon). Bulan baru dapat dimulai keesokan harinya JIKA DAN HANYA JIKA Ijtimak terjadi SEBELUM terbitnya fajar (fajr) di Gisborne, Selandia Baru. Gisborne (Bujur 178.0176° BT) dipilih secara konsensus karena merupakan salah satu kota daratan pertama di dunia

yang paling awal menyongsong fajar. Jika Ijtimak (awal siklus bulan) terjadi setelah fajar menyingsing di Gisborne, maka umat Islam di Selandia Baru telah terlanjur memulai hari puasanya tanpa adanya status bulan baru. Oleh karena itu, jika hal ini terjadi, masuknya bulan baru harus ditunda ke hari lusa untuk menghindari kebingungan teologis.

Perhatikan keanggunan logika fail-safe ini dalam kode sistem:

```
# Jika PKG 1 Gagal, maka masuk ke PKG 2
else:
    gisborne = ephem.Observer()
    gisborne.lat, gisborne.lon = math.radians(-38.6623),
    math.radians(178.0176)
    gisborne.elevation, gisborne.pressure = 0, 0
    # Parameter -18 derajat untuk Astronomical Twilight (Fajar Shadiq)
    gisborne.horizon = '-18'
    matahari_g = ephem.Sun()
    # Mencari waktu terbit fajar di Gisborne
    gisborne.date = ephem.Date(batas_00_utc_ephem - (12.0 / 24.0))
    fajar_gisborne_ephem = gisborne.next_rising(matahari_g, use_center=True)
    # Evaluasi Geografis dan Astronomis PKG 2
    if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
        # Kesimpulan: PKG 2 Terpenuhi. Bulan baru dimulai BESOK (H+1)
    else:
        # Kesimpulan: PKG 2 Gagal. Bulan baru ditunda LUSA (H+2)
```

4.2. Metode Brute-Force dan Iterasi Ephemeris

Dalam disiplin ilmu komputer dan fisika komputasi, pendekatan brute-force (pencarian ekshaustif) umumnya dihindari apabila terdapat algoritma analitik yang lebih efisien, karena beban kompleksitas waktu komputasinya yang tinggi ($O(n)$) (Newman, 2012). Namun, dalam konteks pemindaian (scanning) matlak global untuk KHGT, algoritma prediktif linear tidak dapat digunakan. Hal ini disebabkan oleh topografi bumi yang tidak beraturan, batas-batas daratan (mainland) yang asimetris, serta fluktuasi refraksi atmosfer lokal.

Satu-satunya cara yang valid secara saintifik untuk memastikan di titik mana hilal pertama kali memenuhi kriteria di bumi adalah dengan melakukan komputasi iterative trial-and-error pada seluruh koordinat kota di dunia satu per satu, detik demi detik, pada hari terjadinya konjungsi. Untuk menangani operasi masif ini tanpa menyebabkan system crash, KHGT Times V8.0 merancang arsitektur iterasi yang memadukan kecepatan bahasa C murni dengan keakuratan data NASA.

4.2.1. Peran ephem.Observer dalam Pemindaian Ratusan Kota

Sebagaimana diuraikan pada bab sebelumnya, library Skyfield memiliki akurasi absolut karena membaca langsung data vektor DE441. Namun, memanggil fungsi pencarian waktu terbenam matahari (discrete event finding) menggunakan fungsi `almanac.sunrise_sunset()` milik Skyfield secara berulang pada ratusan kota akan menghasilkan overhead komputasi yang masif dan lambat.

Untuk mengatasi leher botol (bottleneck) performa ini, KHGT Times menggunakan arsitektur komputasi hybrid. Modul ini mendelegasikan tugas pencarian sunset lokal yang brutal kepada pustaka PyEphem (ephem), yang merupakan wrapper langsung dari library astronomi XEphem berbasis bahasa C (Downey, 2011).

Perhatikan bagaimana iterasi ini disusun dalam kode sumber `calculate_first_point`:

```
for negara, kota_dict in CITY_DB.items():
    if negara in zona_dikecualikan:
        continue
    for nama_kota, koordinat in kota_dict.items():
        lintang, bujur = koordinat
        pengamat = ephem.Observer()
        pengamat.lat = str(lintang)
        pengamat.lon = str(bujur)
        pengamat.elevation = 0
        pengamat.date = waktu_pencarian
```



```
try:
    waktu_sunset = pengamat.next_setting(matahari)
except (ephem.AlwaysUpError, ephem.NeverUpError):
    continue # Melewati anomali kutub (Midnight Sun / Polar Night)
```

Metode `ephem.Observer()` memungkinkan sistem melakukan instansiasi objek pengamat dalam memori secara instan. Fungsi `next_setting(matahari)` dijalankan ratusan kali per detik pada basis data CITY_DB yang merentang dari Timur Jauh (Oseania) hingga ujung Barat (Benua Amerika). Blok `try-except` di atas merupakan desain algoritma pertahanan (*defensive algorithm*) untuk menyaring lokasi-lokasi di lintang ekstrem (Kutub Utara/Selatan) yang mengalami anomali matahari tidak terbenam (Midnight Sun), sehingga sistem tidak mengalami *infinite loop*.

4.2.2. Sinkronisasi Data Sunset Lokal dengan New Moon (Ijtimak)

Setelah titik sunset di suatu kota berhasil ditemukan secara cepat oleh `ephem`, kalkulasi belum selesai. Sistem KHGT mensyaratkan perhitungan Ketinggian dan Elongasi harus diekstrak secara Geosentris (berpusat di inti bumi) pada detik di mana sunset lokal itu terjadi (Odeh, 2004). Di sinilah terjadi sinkronisasi atau *handshake* antara mesin `ephem` dan mesin `Skyfield`.

Kode V8.0 melakukan transformasi objek waktu secara melintang dari format `PyEphem` ke format waktu dinamis `Skyfield` (TT/TDB) untuk dievaluasi terhadap posisi New Moon (Ijtimak).

```
# Menerjemahkan waktu sunset dari ephem kembali ke format UTC murni
import pytz
t_sunset_sky
=
self.ts.from_datetime(waktu_sunset.datetime().replace(tzinfo=pytz.utc))
# Mengekstraksi posisi Geosentris menggunakan Skyfield pada detik sunset
tersebut
obs_center = self.eph['earth'].at(t_sunset_sky)
s_app_sky = obs_center.observe(self.eph['sun']).apparent()
m_app_sky = obs_center.observe(self.eph['moon']).apparent()
# Perhitungan Elongasi Geosentrik murni
elong_geo = s_app_sky.separation_from(m_app_sky).degrees
```

Proses sinkronisasi silang (*cross-synchronization*) ini memastikan bahwa KHGT Times V8.0 tidak mengorbankan akurasi demi kecepatan. Waktu terjadinya ijtimak global (`t_ijtimak`) dibandingkan dengan waktu sunset lokal di ratusan titik tersebut untuk mencari di daratan (negara/kota) mana nilai `alt_geo` ≥ 5.0 dan `elong_geo` ≥ 8.0 pertama kali bernilai `True`.

Keberhasilan algoritma ini melacak titik nol pergerakan fase bulan di daratan utama mematahkan keraguan para pakar astronomi klasik mengenai keandalan komputasi pencarian global (*global search constraint*) dalam pembentukan Kalender Hijriah Global Tunggal (Syamsuddin, 2021). Dengan metode *brute-force* yang diakselerasi ini, sebuah komputer personal modern mampu mereplikasi observasi matlak global yang seharusnya membutuhkan ratusan observatorium fisik yang tersebar di seluruh dunia.

4.3. Implementasi Kode Pelacakan Daratan Utama (Mainland)

Konsep visibilitas global mensyaratkan bahwa hilal harus dapat dirukyat (atau memenuhi kriteria hisab) di suatu tempat di permukaan bumi. Namun, secara yurisprudensi dan kelayakan sosiologis,

para pakar astronomi Islam sepakat bahwa titik pemenuhan kriteria tersebut tidak boleh berada di wilayah kepulauan terpencil yang tidak berpenghuni atau terisolasi di tengah samudra (Ilyas, 1994). Jika titik pertama visibilitas jatuh di tengah Samudra Pasifik, memulai kalender Hijriah berdasarkan titik tersebut dianggap tidak merepresentasikan aktivitas kehidupan umat Islam (Syamsuddin, 2021).

Oleh karena itu, perangkat lunak KHGT Times V8.0 menyematkan sebuah lapisan algoritma filter spasial (*spatial filter algorithm*) untuk memastikan bahwa titik yang dilacak adalah bagian dari daratan utama (*mainland*).

4.3.1. Filter Zona Pengecualian (Kepulauan Terpencil)

Untuk mencegah terjadinya false positive—di mana program mendeteksi hilal valid di lokasi yang tidak memenuhi syarat mainland—sistem secara arsitektural menyaring basis data (CITY_DB) sebelum melakukan instansiasi objek pengamat (ephem.Observer()).

Perhatikan implementasi penyaringan zona dalam kode fungsi calculate_first_point:

```
titik_hasil = []
# Mengabaikan pulau kecil / wilayah timur jauh yang bukan daratan utama
(Mainland)
zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu",
"Maluku", "Maluku Utara", "NTT"]
for offset_hari in range(4):
    kandidat_hari_ini = []
    waktu_pencarian = ephem.Date(w_ijtimak + offset_hari)
    for negara, kota_dict in CITY_DB.items():
        if negara in zona_dikecualikan:
            continue # Melewati zona terlarang
```

Penggunaan perintah continue pada iterasi dictionary di atas bertindak sebagai gerbang logika (logic gate) yang sangat efisien (O(1) lookup time). Sistem memangkas beban komputasi dengan tidak melakukan kalkulasi sunset dan geometri bulan-matahari pada kawasan Oseania (seperti Kiriritati, Samoa, dan Tonga) maupun kepulauan kecil di Indonesia bagian timur. Dengan demikian, titik pertama (PKG 1) yang nantinya dideklarasikan oleh program dipastikan selalu berada di atas daratan benua, merujuk pada standar populasi berkesinambungan (continuous population standard).

4.3.2. Penyusunan Logika Komparasi Waktu Gisborne (Selandia Baru)

Ketika titik visibilitas hilal pertama gagal ditemukan sebelum pukul 00:00 UTC (PKG 1 gagal), algoritma KHGT akan jatuh pada pengondisian PKG 2. Skenario terburuk ini biasanya terjadi ketika kriteria $\geq 5^\circ$ dan $\geq 8^\circ$ baru tercapai di Benua Amerika. Untuk memvalidasi kondisi ini, sistem harus memastikan bahwa saat Ijtimak (konjungsi) terjadi, fajar (waktu subuh)

belum menyingsing di belahan bumi paling timur, yang secara konsensus diwakili oleh kota Gisborne, Selandia Baru (Odeh, 2016).

Dalam skrip V8.0, pemodelan ini dikonstruksi secara dinamis. Sistem tidak menggunakan waktu fajar statis, melainkan menghitung Astronomical Twilight (fajar shadiq) di Gisborne secara real-time tepat pada hari terjadinya konjungsi:

```
# Setup stasiun virtual di Gisborne
gisborne = ephem.Observer()
gisborne.lat, gisborne.lon = math.radians(-38.6623), math.radians(178.0176)
gisborne.elevation, gisborne.pressure = 0, 0
gisborne.horizon = '-18' # Kriteria Fajar Astronomis (Subuh)
matahari_g = ephem.Sun()
# Mencari waktu fajar untuk hari H (batas_00_utc_ephem)
# Pengurangan (12.0 / 24.0) untuk menggeser pencarian agar menangkap fajar
pada pagi hari yang benar
gisborne.date = ephem.Date(batas_00_utc_ephem - (12.0 / 24.0))
fajar_gisborne_ephem = gisborne.next_rising(matahari_g, use_center=True)
```

Setelah variabel fajar_gisborne_ephem berhasil diinisialisasi, program melakukan komparasi Boolean tingkat akhir (final boolean validation):

```
negara_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Brasil",
"Argentina", "Kolombia", "Peru", "Chili"]
is_amerika = pt['negara'] in negara_amerika
if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
    status_ijtimak = "Kriteria PKG 2 Terpenuhi (Titik Amerika & Ijtima <
Fajar Gisb.)\n» Kesimpulan: BESOK..."
else:
    alasan = "Titik bukan di Benua Amerika" if not is_amerika else "Ijtima >
Terbit Fajar"
    status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi ({alasan})\n»
Kesimpulan: LUSA..."
```

Logika di atas membuktikan kecanggihan mesin KHGT Times. Ia mengikat data string kewilayahan (variabel negara_amerika) dengan data waktu absolut komputasi astronomi ($pt['ijtimak'] < fajar_gisborne_ephem$). Dengan sinkronisasi ini, program mampu secara otomatis memutus perdebatan fikih apakah bulan baru harus diserentakkan esok hari atau ditunda lusa (istikmal), memberikan keluaran yang otoritatif dan deterministik tanpa memerlukan intervensi manual dari pengguna.

DAFTAR PUSTAKA BAB 4

- Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.
- Newman, M. E. J. (2012). *Computational Physics*. CreateSpace Independent Publishing Platform.
- Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18(1-3), 39-64.
- Odeh, M. S. (2016). *The Global Hijri Calendar: The Istanbul 2016 Congress Resolutions*. International Astronomical Center (IAC).
- Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

BAB 5: PEMETAAN VISUAL SPASIAL (CRESCENT VISIBILITY HD MAP)

Transisi dari hisab berbasis titik (point-based calculation) menuju hisab berbasis kawasan (spatial calculation) merupakan keniscayaan dalam astronomi kalender global. Pendekatan komputasi yang hanya menghitung visibilitas di beberapa kota utama sering kali gagal memberikan gambaran komprehensif mengenai garis batas visibilitas (visibility curve) yang sesungguhnya membelah bumi. Untuk memecahkan problem ini, KHGT Times V8.0 mengimplementasikan Modul Pemetaan Visual Spasial (Crescent Visibility HD Map). Modul ini mentransformasikan data efermis mentah menjadi peta kontur visibilitas beresolusi tinggi yang divalidasi secara matematis.

5.1. Pembangkitan Grid Data Spasial

Pemetaan visibilitas hilal di seluruh dunia mensyaratkan perhitungan parameter astronomis pada ribuan titik potong lintang dan bujur. Metode yang digunakan dalam komputasi ini dikenal sebagai Spatial Grid Generation atau pembangkitan grid spasial. Semakin rapat grid yang dibangun, semakin tinggi resolusi peta yang dihasilkan, namun berisiko memicu beban komputasi yang eksponensial (Caldwell & Laney, 2001).

5.1.1. Ekstraksi Data Ketinggian dan Elongasi pada Skala Lintang/Bujur

Dalam arsitektur KHGT Times V8.0, pembangkitan grid awal (coarse grid) tidak dilakukan secara acak, melainkan menggunakan pola matriks terstruktur dengan resolusi `grid_step = 5` derajat. Ini berarti sistem akan mengeksekusi perhitungan astrometri pada setiap interval 5 derajat dari kutub selatan hingga kutub utara (Latitude: -90 hingga +90), dan dari batas penanggalan barat ke timur (Longitude: -180 hingga +180 atau 0 hingga 360).

Proses ini dieksekusi secara iteratif dalam konstruksi perulangan bersarang (nested loop):

```
grid_step = 5
for lat in range(-90, 91, grid_step):
    lat_rad = math.radians(lat)
    for plot_lon in lon_range:
        # Kalkulasi waktu sunset, altitude, dan elongasi
```

Pada setiap titik persilangan koordinat tersebut, program menghitung waktu terbenamnya matahari (sunset) secara spesifik. Sistem menerapkan rumus trigonometri bola untuk mencari Sudut Jam Matahari Terbenam (Sunset Hour Angle). Setelah waktu sunset di titik grid tersebut didapatkan dalam skala UTC, KHGT Times kemudian memerintahkan mesin Skyfield (`earth.at(t_sunset).observe(moon)`) untuk mengekstraksi parameter kunci: Ketinggian Geosentris (Alt Geo), Elongasi Geosentris, Ketinggian Toposentris, dan Fraksi Iluminasi.

Data-data ini tidak langsung digambar, melainkan disimpan (appended) ke dalam larik data mentah (array) berupa `lons_coarse`, `lats_coarse`, `alt_data`, dan `elong_data`. Pendekatan ekstraksi

matriks ini sangat sejalan dengan metodologi pemetaan astronomis modern yang dikemukakan oleh Yallop (1998) dalam membangun algoritma kurva q-value untuk peta visibilitas.

5.1.2. Transformasi Kurva Kontinu Melampaui Garis Batas Penanggalan

Salah satu masalah klasik dalam pemetaan global kalender hijriah adalah terputusnya kurva visibilitas saat melewati Samudra Pasifik, tepatnya di Garis Batas Penanggalan Internasional (International Date Line / IDL) pada bujur 180 derajat. Pemetaan tradisional umumnya menghasilkan peta yang terbelah dan membingungkan karena belahan timur bumi secara teknis telah memasuki tanggal masehi yang berbeda dibandingkan belahan barat (Hoffman, 2003).

Keunggulan komputasional KHGT Times V8.0 terlihat jelas pada algoritma penyatuan hari (seamless chronological mapping) khusus untuk peta KHGT. Sistem mendeteksi parameter `is_khgt_map` dan mengubah rentang longitude dari `[-180, 180]` menjadi `[0, 360]`.

Logika transformasi ini diimplementasikan untuk menggabungkan dua hari kalender masehi yang berbeda ke dalam satu lembar peta visual yang berkesinambungan:

```
if is_khgt_map:
    if plot_lon < 180:
        real_lon = plot_lon
        # Belahan Timur (Kiri Peta) dihitung memakai Hari Besoknya (D+1)
        dt_calc = datetime.date(year, month, day) +
        datetime.timedelta(days=1)
    else:
        real_lon = plot_lon - 360
        # Belahan Barat (Kanan Peta) dihitung memakai Hari Input (D)
        dt_calc = datetime.date(year, month, day)
```

Melalui manipulasi time-shifting di atas, KHGT Times "menipu" mesin astronomi agar memproses hari H (Hari konjungsi) untuk belahan bumi barat (Amerika hingga Pasifik Tengah), sekaligus memproses hari H+1 untuk belahan bumi timur (Oseania, Asia, hingga Eropa) di dalam satu siklus looping yang sama. Hasilnya, saat data ini di-plot, kurva Ketinggian 5 derajat dan Elongasi 8 derajat akan menyambung secara sempurna melewati batas benua, memperlihatkan zona hijau (Kriteria Terpenuhi) yang koheren dari benua Amerika melingkar menyapu benua Asia tanpa adanya distorsi spasial.

5.2. Interpolasi Matematis Menggunakan SciPy

Pembangkitan matriks data kasar (coarse grid) dengan interval 5 derajat lintang dan bujur pada tahap sebelumnya memang sangat efisien untuk memangkas waktu komputasi (mengurangi beban kerja `ephem.Observer`). Namun, trade-off dari efisiensi ini adalah resolusi spasial yang sangat rendah. Jika matriks data kasar ini divisualisasikan secara langsung, kurva batas visibilitas hilal akan tampak bergerigi (pixelated atau blocky) dan meleset dari presisi geografis yang sebenarnya.

Untuk menghasilkan peta visibilitas definisi tinggi (HD) di mana garis batas kriteria membelah topografi daratan bumi secara presisi, diperlukan teknik interpolasi spasial multivariabel (Lancaster & Salkauskas, 1986). Dalam KHGT Times V8.0, teknik rekonstruksi piksel ini didelegasikan secara penuh pada pustaka komputasi saintifik SciPy.

5.2.1. Konsep Bicubic Griddata Interpolation

Pustaka SciPy menyediakan sub-modul `scipy.interpolate` yang difungsikan untuk menaksir (estimate) nilai-nilai parameter astronomis (seperti altitude, elongasi, dan fraksi iluminasi) pada titik-titik koordinat mikroskopis yang tidak dihitung secara komputasi brute-force oleh mesin ephemeris.

Dalam implementasinya, KHGT Times merekonstruksi ruang sampel spasial dari interval 5 derajat menjadi matriks koordinat yang sangat rapat (fine grid) dengan resolusi 0.2 derajat:

```
lon_fine = np.arange(0, 360.25, 0.2)
lat_fine = np.arange(-90, 90.25, 0.2)
LONS_fine, LATS_fine = np.meshgrid(lon_fine, lat_fine)
```

Untuk merekonstruksi permukaan data dari coarse grid ke fine grid tersebut, sistem menggunakan algoritma interpolasi Kubik (`method='cubic'`). Algoritma ini membangun kurva polinomial derajat tiga (cubic spline) di antara himpunan titik-titik data asal yang diketahui.

Berbeda dengan interpolasi linier yang hanya menghubungkan antar-titik dengan garis lurus kaku, interpolasi kubik menjamin kontinuitas turunan pertama dan kedua dari permukaan fungsi ruang. Hasilnya, transisi nilai ketinggian hilal (misalnya dari 4° menuju 6°) diekstrapolasikan sedemikian rupa sehingga menghasilkan kurva visibilitas (seperti garis tegas kriteria KHGT di titik 5°) yang sangat mulus dan natural saat memotong lekukan benua dan samudra di peta dunia (Franke, 1982).

5.2.2. Penanganan Nilai Kosong (NaN) dengan Nearest Neighbor

Meskipun interpolasi kubik sangat superior dalam menghasilkan kontur kurva yang elegan, algoritma ini memiliki kelemahan inheren pada aspek ekstrapolasi tepi. Fungsi `griddata` pada SciPy yang menggunakan interpolasi kubik berbasis pada triangulasi spasial Delaunay (Delaunay triangulation). Triangulasi ini secara matematis tidak dapat menghitung turunan nilai di luar convex hull (batas poligon terluar dari himpunan titik data asli) (Virtanen et al., 2020).

Akibatnya, pada wilayah perbatasan absolut peta (seperti pada meridian batas penanggalan $\pm 180^\circ$ atau di titik kutub absolut $\pm 90^\circ$), algoritma interpolasi kubik sering kali gagal merender data dan menghasilkan nilai kosong atau Not a Number (NaN). Hal ini akan memunculkan blind spot (lubang putih/transparan) pada hasil rendering peta visibilitas hilal.

Untuk menambal lubang komputasional ini, kode arsitektur KHGT Times V8.0 mengimplementasikan strategi Dual Interpolation Fill (Interpolasi Ganda Berbasis Masking):

```
# 1. Interpolasi Utama (Cubic) untuk presisi dan kehalusan kurva
grid_alt = interp.griddata(points, plot_data['alt'], (LONS_fine, LATS_fine),
method='cubic')
# 2. Interpolasi Cadangan (Nearest) untuk mengatasi blind spot di Convex Hull
grid_alt_near
=
interp.griddata(points,
plot_data['alt'],
(LONS_fine,
LATS_fine), method='nearest')
# 3. Penambalan array (Masking Array)
# Mencari matriks NaN pada grid cubic, lalu menggantinya dengan nilai dari grid
nearest
grid_alt[np.isnan(grid_alt)] = grid_alt_near[np.isnan(grid_alt)]
```

Cara kerja Dual Interpolation ini sangat cerdas:

1. Sistem pertama-tama merender permukaan piksel kubik sebagai layer utama.
2. Secara bersamaan (di thread array memori), sistem menjalankan interpolasi kedua

menggunakan metode Nearest Neighbor (`method='nearest'`). Metode ini beroperasi dengan prinsip Voronoi tessellation yang menarik nilai dari titik data terdekat tanpa menggunakan perhitungan turunan spasial, sehingga selalu mampu memberikan nilai ekstrapolasi hingga ke ujung batas matriks peta. 3. Baris terakhir pada kode di atas bertindak sebagai masking filter. Fungsi `np.isnan`

menyeleksi indeks array yang rusak (NaN) pada kontur kubik, dan langsung "menambalnya" (patching) menggunakan value dari kontur nearest.

Strategi penambalan matriks spasial ini menjamin bahwa Peta Visibilitas Hilal HD yang diekspor dari aplikasi memiliki garis batas parameter (seperti kontur elongasi 8°) yang sangat luwes di bagian tengah, namun tetap berwujud solid tanpa degradasi lubang piksel hingga ke batas tepi terluar topografi bumi.

5.3. Rendering Peta Heatmap (Matplotlib)

Setelah matriks data koordinat berhasil direkonstruksi dan diperhalus menggunakan interpolasi SciPy, tahap pamungkas dari modul terkait adalah memproyeksikannya ke dalam wujud visual yang dapat diinterpretasi oleh mata manusia. Dalam KHGT Times V8.0, proses rendering grafis ini ditangani sepenuhnya oleh pustaka Matplotlib. Proses ini tidak sekadar menggambar garis, melainkan membangun layer (lapisan) visual secara bertumpuk (z-order stacking), memadukan topografi fisik bumi dengan kontur data astronomis abstrak.

5.3.1. Overlay Topografi Bumi dan Boundary Norm Kontur Warna

Sebuah peta visibilitas hilal kehilangan konteks geografisnya jika tidak disandingkan dengan peta topografi benua dan batas negara. KHGT Times V8.0 mengatasi hal ini dengan memuat gambar peta dasar (base map) beresolusi tinggi (map_topografi.jpg) yang diproyeksikan ke dalam kanvas Matplotlib menggunakan fungsi imshow.

Tantangan utama dalam overlay ini adalah sinkronisasi rentang sumbu (axis extent). Sebagaimana dibahas pada sub-bab sebelumnya, peta KHGT menggunakan rentang bujur 0 hingga 360 derajat agar batas penanggalan tidak terputus. Untuk menyelaraskan gambar topografi dengan rentang bujur modifikasi ini, sistem melakukan manipulasi matriks gambar (pembelahan dan penggabungan array piksel) menggunakan NumPy sebelum merendernya:

```
# Pemotongan matriks gambar tepat di tengah (Bujur 180 / IDL)
mid = img_arr.shape[1] // 2
# Menggabungkan kembali dengan menukar posisi belahan timur dan barat
img_shifted = np.concatenate((img_arr[:, mid:], img_arr[:, :mid]), axis=1)
# Overlay ke kanvas dengan batas bujur 0 hingga 360
ax.imshow(img_shifted, extent=[0, 360, -90, 90], aspect='auto', alpha=1.0,
zorder=0)
```

Setelah topografi bumi terpasang di layer dasar (zorder=0), sistem melukiskan data visibilitas hilal di atasnya. Untuk menghindari efek gradasi warna yang membingungkan secara fikih (karena hukum fikih bersifat diskrit: sah atau tidak sah), sistem menggunakan teknik Boundary Norm.

Teknik BoundaryNorm memaksa Matplotlib untuk tidak menginterpretasikan matriks nilai ketinggian dan elongasi sebagai warna yang memudar, melainkan membaginya menjadi blok warna (solid zones) yang tegas (Hunter, 2007). Jika sebuah wilayah memiliki nilai di atas ambang batas (diberi skor 1), ia akan dirender dengan warna hijau pekat (#00E676), sedangkan wilayah di bawah ambang batas (skor 0) akan dibiarkan transparan (alpha=0.2).

5.3.2. Visualisasi Interseksi Kriteria KHGT (Garis Batas 5 derajat dan 8 derajat)

Bagian paling krusial dari peta KHGT adalah penentuan batas demarkasi: di manakah tepatnya garis yang memisahkan wilayah yang memenuhi kriteria dan yang tidak? Kriteria KHGT memandatkan pemenuhan dua parameter secara bersamaan: Ketinggian Geosentris $\geq 5^\circ$ dan Elongasi Geosentris $\geq 8^\circ$.

Alih-alih menyatukannya menjadi satu garis abstrak yang membingungkan, KHGT Times V8.0 mengadopsi prinsip transparansi data (data transparency). Sistem secara independen menggambar kontur batas ketinggian 5 derajat dan kontur elongasi 8 derajat sebagai dua entitas garis yang berbeda, lalu menyorot area irisan (intersection) dari kedua parameter tersebut.

Implementasinya dalam blok kode Matplotlib:

```
# Mencari irisan (Interseksi) di mana Alt >= 5 DAN Elong >= 8
grid_score = np.zeros_like(grid_alt)
grid_score[(grid_alt >= 5.0) & (grid_elong >= 8.0)] = 1
# Mewarnai area interseksi dengan warna Hijau
cmap_inter = ListedColormap([(0, 0, 0, 0), '#00E676'])
bounds_inter = [-0.5, 0.5, 1.5]
norm_inter = BoundaryNorm(bounds_inter, cmap_inter.N)
ax.contourf(LONS_fine, LATS_fine, grid_score, cmap=cmap_inter,
norm=norm_inter, alpha=0.2, zorder=2)
# Menarik Garis Tegas Kontur Ketinggian 5 Derajat (Biru Tua)
cs_alt = ax.contour(LONS_fine, LATS_fine, grid_alt, levels=[5.0],
colors=['#1A237E'], linewidths=1.5, linestyle='solid', zorder=3)
ax.clabel(cs_alt, inline=True, fontsize=10, fmt='Alt 5°')
# Menarik Garis Tegas Kontur Elongasi 8 Derajat (Merah)
cs_elong = ax.contour(LONS_fine, LATS_fine, grid_elong, levels=[8.0],
colors=['#D32F2F'], linewidths=1.5, linestyle='solid', zorder=3)
ax.clabel(cs_elong, inline=True, fontsize=10, fmt='Elong 8°')
```

Melalui fungsi ax.contour, sistem mencari kurva level (isoline) di mana matriks grid_alt tepat bernilai 5.0 dan matriks grid_elong tepat bernilai 8.0 (Hoffman, 2003). Garis-garis ini diberi warna kontras (Biru Tua dan Merah) serta label otomatis (ax.clabel).

Dengan arsitektur visual ini, pengguna tidak hanya melihat "area hijau" tempat bulan baru dimulai, tetapi juga dapat memvalidasi secara optis mengapa batas tersebut terbentuk—yakni di titik mana kurva elongasi memotong kurva ketinggian. Pendekatan visualisasi saintifik ini menjadikan peta keluaran KHGT Times V8.0 setara dengan standar peta astronomi observatorium internasional, sekaligus menjadi instrumen advokasi yang sangat meyakinkan bagi pengambil kebijakan dalam menerapkan Kalender Hijriah Global Tunggal.

DAFTAR PUSTAKA BAB 5

- Caldwell, J. A. R., & Laney, C. D. (2001). First visibility of the lunar crescent. *African Skies/Cieux Africains*, 5, 15-23.
- Franke, R. (1982). Scattered data interpolation: tests of some methods. *Mathematics of Computation*, 38(157), 181-200.
- Hoffman, R. E. (2003). Rationalizing the Islamic calendar. *Science and Islamic Observatory*, 1(1), 1-12.
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.
- Lancaster, P., & Salkauskas, K. (1986). *Curve and Surface Fitting: An Introduction*. Academic Press.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.
- Yallop, B. D. (1998). A Method for Predicting the First Sighting of the New Crescent Moon (NAO Technical Note No. 69). HM Nautical Almanac Office.

BAB 6: FORMULASI WAKTU SALAT DAN FAJAR SHADIQ

Ilmu miqat (perhitungan waktu salat) merupakan domain spesifik di mana astronomi bola (spherical astronomy) berinteraksi langsung dengan fikih ibadah harian. Berbeda dengan penentuan awal bulan yang terfokus pada posisi relatif Bulan dan Matahari (elongasi dan ketinggian), komputasi waktu salat murni bertumpu pada pergerakan semu harian Matahari yang diukur secara toposentris (dari permukaan bumi pengamat).

Dalam arsitektur KHGT Times V8.0, modul waktu salat tidak menggunakan rumus trigonometri bola sederhana yang rentan terhadap galat (error) pembulatan, melainkan mengadopsi teknik pemrosesan larik numerik resolusi tinggi (high-resolution numerical array processing) yang langsung diekstraksi dari ephemeris NASA.

6.1. Algoritma Dasar Posisi Matahari

Penentuan awal waktu salat didefinisikan oleh posisi Matahari terhadap meridian pengamat dan horizon lokal. Dinamika ini sangat bergantung pada deklinasi Matahari dan lintang geografis pengamat yang terus berubah secara non-linear setiap harinya (Ilyas, 1994).

6.1.1. Perhitungan Transit (Zawal) dan Equation of Time

Secara tradisional, waktu Dzuhur (Zawal) dihitung ketika Matahari tepat melintasi meridian lokal (kulminasi atas). Rumus analitik klasik umumnya menghitung waktu Dzuhur dengan mencari siang hakiki (pukul 12:00 waktu lokal) kemudian dikoreksi menggunakan Equation of Time (EoT) dan selisih bujur standar dengan bujur lokal. EoT adalah perbedaan antara waktu matahari sejati (apparent solar time) dan waktu matahari rata-rata (mean solar time) yang diakibatkan oleh eksentrisitas orbit bumi dan kemiringan ekliptika (Meeus, 1998).

Namun, KHGT Times V8.0 meninggalkan pendekatan analitik EoT tersebut. Sebagai gantinya, sistem ini memanfaatkan mesin integrasi numerik untuk memindai titik tertinggi Matahari di langit secara empiris. Perhatikan teknik Array Maximization pada skrip berikut:

```
tt_array = np.linspace(t0.tt, t1.tt, 2880)
t_search = self.ts.tt_jd(tt_array)
alt_deg = (earth + loc).at(t_search).observe(sun).apparent().altaz(
    temperature_C=temp, pressure_mbar=pres
)[0].degrees
idx_noon = np.argmax(alt_deg)
tt_dhohur = tt_array[idx_noon]
alt_noon = alt_deg[idx_noon]
```

Alih-alih menggunakan rumus, program membangkitkan array waktu `np.linspace` dengan 2880 sampel per hari (ekuivalen dengan resolusi 1 sampel setiap 30 detik). Mesin Skyfield kemudian menghitung ketinggian Matahari (altitude) pada 2880 titik tersebut secara simultan berkat operasi vektorisasi NumPy. Fungsi `np.argmax(alt_deg)` kemudian melacak indeks array di mana ketinggian Matahari mencapai nilai maksimum (`alt_noon`).

Waktu yang berkorespondensi dengan indeks tersebut (`tt_dhohur`) secara de facto adalah waktu Zawal atau masuknya waktu Dzuhur. Metode pencarian empiris ini mengeliminasi kompleksitas dan potensi galat dari rumus EoT klasik, serta secara otomatis memperhitungkan efek perturbasi gravitasi sekunder yang memengaruhi kecepatan orbit bumi (Urban & Seidelmann, 2013).

6.1.2. Trigonometri Bola untuk Waktu Terbit dan Terbenam

Waktu terbit (Syuruq) dan terbenam (Maghrib) didefinisikan sebagai momen ketika piringan atas Matahari tepat bersinggungan dengan horizon geometris pengamat. Jika kita mengasumsikan Bumi

sebagai titik tanpa atmosfer dan Matahari sebagai titik koordinat tanpa dimensi, maka fenomena ini terjadi ketika altitude Matahari bernilai 0° .

Namun, secara astrofisika, perhitungan harus memasukkan koreksi jari-jari semu Matahari (semi-diameter, approx $16'$) dan refraksi atmosfer horizontal (approx $34'$). Refraksi mengangkat bayangan Matahari sedemikian rupa sehingga Matahari tampak di ufuk padahal secara geometris sudah berada di bawah ufuk. Total koreksi standar untuk kedua faktor ini adalah 50 menit busur atau setara dengan 0.833° (Schaefer, 1993). Oleh karena itu, pusat piringan Matahari secara geometris harus berada pada ketinggian -0.833° agar piringan atasnya terlihat tepat menyentuh cakrawala.

Dalam kode KHGT Times, pencarian nilai ini dilakukan menggunakan algoritma deteksi persilangan garis (crossing detection algorithm):

```
def find_crossing(alt_arr, tt_arr, target_alt, direction='up'):
    diffs = alt_arr - target_alt
    for i in range(len(diffs)-1):
        if direction == 'up' and diffs[i] <= 0 and diffs[i+1] > 0:
            frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
            return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
    # ... [logika untuk 'down' saat Maghrib]
```

Metode `find_crossing` mengandalkan interpolasi linier di antara dua sampel array 30-detik yang mengapit target sudut.

- Untuk waktu Syuruq (terbit), sistem memanggil: `find_crossing(alt_am, tt_am, - 0.833, 'up')`.
- Untuk waktu Maghrib (terbenam), sistem memanggil: `find_crossing(alt_pm, tt_pm, -0.833, 'down')`.

Penggunaan target spasial absolut -0.833° digabungkan dengan pemecahan fraksional (`frac`) menjamin presisi hingga level sub-detik. Pendekatan diskrit berbasis array ini terbukti jauh lebih kokoh (robust) dibandingkan iterasi `f-solve` (seperti Newton-Raphson) yang sering kali gagal konvergen atau menghasilkan nilai tak terhingga (NaN) saat diterapkan pada wilayah lintang ekstrem (kutub) yang mengalami fenomena Midnight Sun.

6.2. Evaluasi Sudut Depresi dan Bayangan

Selain fenomena terbit dan terbenam, dua titik krusial lainnya dalam siklus waktu salat harian adalah penentuan waktu Asar dan waktu fajar/isyah. Keduanya dihitung berdasarkan geometri spesifik yang bukan berada pada horizon 0° , melainkan melibatkan rasio bayangan (Asar) dan sudut depresi negatif matahari di bawah ufuk (fajar dan isyah).

6.2.1. Formulasi Waktu Asar (Mazhab Standar vs Hanafi)

Waktu Asar tidak didefinisikan berdasarkan sudut altitude mutlak, melainkan berdasarkan panjang bayangan sebuah benda tegak lurus relatif terhadap panjang bayangan benda tersebut pada waktu Dzuhur (Zawal) (Ilyas, 1994). Secara matematis, panjang bayangan pada saat Dzuhur bergantung pada selisih antara deklinasi matahari (δ) dan lintang pengamat (ϕ).

Dalam KHGT Times V8.0, kalkulasi ini tidak menggunakan pendekatan analitik klasik yang rawan kesalahan saat diaplikasikan pada lintang ekstrem, melainkan menggunakan iterasi trigonometris langsung:

1. Mencari Altitude Dzuhur: Dari array ketinggian matahari harian, sistem mengambil nilai maksimum (`alt_noon`), yang merepresentasikan ketinggian matahari saat melintasi meridian.
2. Kalkulasi Zenith Dzuhur: Sudut Zenith dihitung sebagai $Z_noon = 90^\circ -$

`alt_noon`. Panjang bayangan saat Dzuhur untuk benda berskala 1 satuan adalah $\tan(Z_noon)$.

3. Target Bayangan Asar: Terdapat dua mazhab utama yang memengaruhi panjang

bayangan ini. Mazhab Jumhur (Syafi'i, Maliki, Hanbali) menetapkan waktu Asar masuk ketika panjang bayangan benda = tinggi benda + bayangan Dzuhur. Mazhab Hanafi

menetapkan panjang bayangan = $2 \times$ tinggi benda + bayangan Dzuhur.

4. Konversi ke Target Altitude: Panjang bayangan target tersebut dikonversi kembali menjadi sudut ketinggian matahari yang harus dicapai (`alt_asr_target`).

Kode implementasinya sangat elegan:

```
zenith_noon = 90.0 - alt_noon
shadow_noon = math.tan(math.radians(max(0, zenith_noon)))
shadow_asr = asr_factor + shadow_noon # asr_factor bernilai 1.0 (Jumhur)
atau 2.0 (Hanafi)
alt_asr_target = math.degrees(math.atan(1.0 / shadow_asr))
```

Setelah target ketinggian Asar (`alt_asr_target`) didapatkan, sistem memanggil fungsi pendeteksi persilangan (`find_crossing`) pada larik waktu sore hari (`alt_pm`, `tt_pm`) untuk menemukan exact timing kapan matahari melintasi target sudut tersebut ke arah bawah.

6.2.2. Analisis Sudut Fajar (-20 derajat vs -18 derajat) dan Nilai SQM Teoritis

Berbeda dengan Maghrib yang ditandai oleh hilangnya piringan matahari, waktu Subuh (Fajar Shadiq) dan Isya ditentukan oleh fenomena cahaya hamburan atmosfer (twilight). Secara hisab, waktu ini merujuk pada sudut depresi (sudut di bawah ufuk) pusat piringan matahari.

Namun, terdapat perbedaan konsensus internasional mengenai besaran sudut depresi untuk masuknya waktu Subuh:

1. Kemenag RI: Menggunakan -20° .
2. MABIMS / Muhammadiyah: Menggunakan -18° (merujuk pada kriteria

Astronomical Twilight internasional).

Menu 29 (Evaluasi Fajar Shadiq & SQM) pada KHGT Times V8.0 dirancang untuk membedah perbedaan ini. Sistem tidak hanya menghitung waktu untuk kedua sudut depresi tersebut, tetapi juga menghubungkannya dengan tingkat kegelapan langit secara teoritis yang diukur dalam Sky Quality Meter (SQM) (mag/arcsec^2).

Menggunakan fungsi generator trigonometris `f_angle` dalam pustaka Skyfield, sistem mencari dua titik persilangan sekaligus:

```
def f_angle(angle_deg):
def _sun_alt(t):
alt, az, distance = topos_obs.at(t).observe(sun).apparent().altaz()
return alt.degrees >= angle_deg
_sun_alt.step_days = 0.04
return _sun_alt
f_18 = f_angle(-18.0)
f_20 = f_angle(-20.0)
t_18_ev, y_18_ev = almanac.find_discrete(t0, t1, f_18)
t_20_ev, y_20_ev = almanac.find_discrete(t0, t1, f_20)
```

Pemodelan SQM teoritis dalam KHGT Times mengkorelasikan sudut depresi tersebut dengan kecerahan langit. Berdasarkan literatur fotometri atmosferik (Patat, 2003), ketika matahari berada di -18° , langit diasumsikan mencapai kegelapan penuh (SQM approx 21.0 - 21.5 mag/arcsec^2). Evaluasi ini sangat relevan untuk mengkalibrasi instrumen optik dalam pengukuran fajar yang sebenarnya di lapangan.

6.3. Penanganan Anomali Lintang Tinggi (High Latitude)

Sistem waktu salat konvensional dibangun di atas asumsi geometri bumi menengah (ekuator hingga lintang pertengahan). Namun, ketika koordinat pengamat digeser menuju wilayah lintang tinggi (high latitude), seperti Skandinavia, Kanada Utara, atau Lingkaran Arktik, asumsi geometri ini runtuh. Di wilayah-wilayah ekstrem ini, sudut depresi matahari untuk fajar (-18°) dan isya tidak pernah tercapai selama musim panas, atau matahari bahkan tidak pernah terbit selama musim dingin (fenomena Polar Night dan Midnight Sun) (Moosavi et al., 2015).

Untuk memelihara keabsahan ibadah di wilayah-wilayah ini, KHGT Times V8.0 menyediakan sub-sistem khusus yang dapat diaktifkan melalui menu `combo_pt_highlat`. Sistem ini bukan lagi murni observasi astronomis, melainkan integrasi ijtihad fikih dengan komputasi proporsional.

6.3.1. Metode Nisf al-Lail (Setengah Malam) dan Subeh Sadiq (Sepertujuh Malam)

Ketika matahari tidak turun cukup dalam hingga mencapai -18° saat musim panas, tanda- tanda fajar dan isya astronomis akan menyatu (continuous twilight). Untuk menentukan kapan Isya masuk dan kapan Fajar (Subuh) dimulai, para ahli fikih dan astronom merumuskan pembagian durasi malam buatan.

Durasi malam (ΔT_{night}) dihitung dari waktu terbenamnya matahari (Maghrib, t_{maghrib}) hingga terbitnya matahari (Syuruq, t_{syuruq}) pada hari berikutnya:

$$\Delta T_{\text{night}} = t_{\text{syuruq}} - t_{\text{maghrib}}$$

Dalam KHGT Times V8.0, pendekatan Nisf al-Lail (Setengah Malam) membagi durasi malam ini menjadi dua bagian yang sama panjang. Waktu Isya diasumsikan masuk pada pertengahan malam, dan waktu Subuh diasumsikan masuk dari pertengahan malam tersebut hingga Syuruq. Implementasi pada kode:

```
if "1/2 Malam" in highlat_method:
    porsi = durasi_malam / 2.0
    if val_isha is None: val_isha = val_maghreb + porsi
    if val_fajr is None: val_fajr = val_shuroq - porsi
```

Pendekatan lain yang juga difasilitasi adalah Subeh Sadiq atau Sepertujuh Malam (1/7). Metode ini dipopulerkan di beberapa wilayah lintang tinggi di Asia Tengah. Waktu isya masuk setelah 1/7 bagian malam berlalu sejak Maghrib, dan waktu subuh dimulai pada 1/7 bagian malam sebelum Syuruq.

```
elif "1/7 Malam" in highlat_method:
    porsi = durasi_malam / 7.0
    if val_isha is None: val_isha = val_maghreb + porsi
    if val_fajr is None: val_fajr = val_shuroq - porsi
```

Metode ini memastikan bahwa umat Islam di wilayah tersebut tetap memiliki jeda waktu yang proporsional untuk melaksanakan salat wajib dan berpuasa (Ilyas, 1994).

6.3.2. Pendekatan Aqrab al-Balad dan Proporsi Sudut

Metode Aqrab al-Balad (Kota Terdekat yang Normal) dan Angle-Based Proportion (Proporsi Sudut) adalah solusi yang lebih dinamis. Pada metode proporsi sudut, waktu malam dibagi tidak berdasarkan pecahan yang kaku, melainkan berdasarkan porsi sudut target (misal 18°) relatif terhadap total 60° (Ilyas, 1994).

```
elif "Proporsi Sudut" in highlat_method:
    porsi_isha = durasi_malam * (abs(isha_angle) / 60.0)
    porsi_fajr = durasi_malam * (abs(fajr_angle) / 60.0)
    if val_isha is None: val_isha = val_maghreb + porsi_isha
    if val_fajr is None: val_fajr = val_shuroq - porsi_fajr
```

Sementara itu, untuk kondisi ekstrem di mana matahari benar-benar tidak terbit (Polar Night) atau tidak terbenam (Midnight Sun), sistem mendeteksinya melalui variabel `val_maghreb is None` or `val_shuroq is None`. Dalam kondisi ini, sistem KHGT Times V8.0 mengadopsi asumsi matematis Aqrab al-Ayyam atau estimasi 45° . Program akan menarik garis waktu statis dari waktu Dzuhur (karena titik kulminasi atas matahari biasanya tetap ada meski matahari tidak tenggelam):

```
if val_maghreb is None or val_shuroq is None:
    if tt_dhohur is not None:
        # Aqrab al-Ayyam Asumsi Dasar (Menarik garis 6 jam dari waktu
        # Dzuhur/Transit)
        if val_shuroq is None: val_shuroq = tt_dhohur - (6.0 / 24.0)
        if val_maghreb is None: val_maghreb = tt_dhohur + (6.0 / 24.0)
```

```
if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)
if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
```

Pendekatan ini mengasumsikan durasi siang standar ekuatorial, yakni 12 jam (6 jam sebelum dan 6 jam sesudah Dzuhur). Dengan arsitektur logika ini, KHGT Times V8.0 memposisikan dirinya sebagai engine yang tidak hanya valid secara astronomis, tetapi juga secara adaptif dapat diimplementasikan untuk kemaslahatan ibadah di seluruh penjuru bumi tanpa terkecuali.

DAFTAR PUSTAKA BAB 6

- Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.
- Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.
- Moosavi, S. Z., Bakhshi, H., & Zadehkhorrani, A. R. (2015). A survey on calculation methods of prayer times at high latitudes. *International Journal of Mechatronics, Electrical and Computer Technology*, 5(15), 2095-2105.
- Patat, F. (2003). UVRI night sky brightness during sunspot maximum at ESO-Paranal. *Astronomy & Astrophysics*, 400(3), 1183-1198.
- Schaefer, B. E. (1993). Astronomy and the limits of vision. *Vistas in Astronomy*, 36, 311-361.
- Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

BAB 7: TRIGONOMETRI ARAH KIBLAT DAN BAYANGAN MATAHARI

Penentuan arah kiblat dan waktu-waktu bayangan matahari merupakan elemen krusial dalam ilmu falak yang bersinggungan langsung dengan keabsahan ibadah salat. Secara matematis, kalkulasi ini menuntut pemahaman geometri spasial yang mengkorelasikan titik koordinat pengamat, titik koordinat Ka'bah di Makkah, serta lintasan semu harian matahari. Dalam arsitektur KHGT Times V8.0, modul navigasi kiblat tidak sekadar memberikan angka statis derajat kompas, tetapi juga mengintegrasikan model geodesi modern dengan metode pelacakan bayangan (rashdul kiblat) yang tervalidasi secara astronomis.

7.1. Model Bumi Elipsoid (WGS84) vs Bola Standar

Dalam sejarah komputasi awal Islam, bumi diasumsikan sebagai sebuah bola pejal yang sempurna (perfect sphere). Namun, ilmu geodesi modern membuktikan bahwa rotasi bumi menyebabkan pampatan di kutub dan pelebaran di ekuator, menjadikan bentuk bumi sebagai oblate spheroid (elipsoid pepat) (Torge & Müller, 2012).

Perbedaan asumsi bentuk bumi ini akan menghasilkan dua jenis lintang: lintang geografis (diukur terhadap garis normal elipsoid) dan lintang geosentris (diukur terhadap pusat bumi). Mengabaikan pampatan bumi dapat menghasilkan deviasi arah kiblat hingga puluhan menit busur pada wilayah tertentu. Menyadari hal ini, KHGT Times V8.0 menggunakan datum referensi World Geodetic System 1984 (WGS84). Penggunaan datum ini secara konsisten dideklarasikan melalui pustaka Skyfield dalam penentuan posisi stasiun pengamat:

```
from skyfield.api import wgs84
loc = wgs84.latlon(lat, lon, elevation_m=elev)
```

Metode ini menjamin bahwa ketinggian elevasi pengamat dari atas permukaan laut (mdpl) diproyeksikan secara presisi tegak lurus terhadap permukaan elipsoid bumi, bukan pada permukaan bola imajiner, sehingga koreksi paralaks dan refraksi yang dihitung sebelumnya (pada Bab 6) menjadi sangat presisi.

7.1.1. Formulasi Jarak Terpendek (Great Circle) ke Ka'bah

Meskipun penempatan stasiun observasi menggunakan model elipsoid WGS84, perhitungan arah kiblat itu sendiri secara historis dan praktis paling banyak diselesaikan menggunakan trigonometri bola (spherical trigonometry) melalui prinsip Lingkaran Besar (Great Circle). Great Circle adalah lingkaran pada permukaan bola bumi yang pusatnya berimpit dengan pusat bumi; rute terpendek antara dua titik di permukaan bumi selalu mengikuti busur lingkaran besar ini (Abdali, 1997).

Berdasarkan teorema sinus dan kosinus pada segitiga bola yang dibentuk oleh titik Kutub Utara, titik Pengamat (P), dan titik Ka'bah (K), formulasi dasar untuk mencari sudut arah kiblat (Q) dari titik utara sejati (true north) dirumuskan sebagai berikut:

$$\tan(Q) = \sin(\lambda K - \lambda P) / (\cos(\phi P) \tan(\phi K) - \sin(\phi P) \cos(\lambda K - \lambda P))$$

Di mana:

Keterangan Simbol:

- Q: Sudut arah Kiblat.
- λK : Bujur (Longitude) Ka'bah.
- λP : Bujur (Longitude) lokasi pengamat.
- ϕK : Lintang (Latitude) Ka'bah.
- ϕP : Lintang (Latitude) lokasi pengamat.
- sin, cos, tan: Fungsi trigonometri (Sinus, Kosinus, Tangen).

Model trigonometri bola ini memiliki tingkat akurasi yang lebih dari cukup untuk tujuan orientasi bangunan ibadah, di mana deviasi sub-derajat dari model elipsoid absolut tidak dapat dideteksi secara visual dalam proses pembangunan fisik (King, 1986).

7.1.2. Penentuan Azimuth Kiblat dari Berbagai Titik Bumi

Algoritma dalam KHGT Times V8.0 secara murni menerjemahkan formulasi matematis di atas ke dalam fungsi Python `calculate_qiblah()`. Titik referensi koordinat Ka'bah yang diencode dalam sistem ini adalah Lintang 21.4225° LU dan Bujur 39.8262° BT.

Perhatikan implementasi blok kode berikut:

```
# Koordinat Absolut Ka'bah
phi_k = math.radians(21.4225)
lam_k = math.radians(39.8262)
# Koordinat Pengamat
phi = math.radians(lat)
lam = math.radians(lon)
# Eksekusi Trigonometri Bola
y_val = math.sin(lam_k - lam)
x_val = math.cos(phi) * math.tan(phi_k) - math.sin(phi) * math.cos(lam_k - lam)
# Menggunakan atan2 untuk penentuan kuadran secara presisi
qiblah_angle = math.degrees(math.atan2(y_val, x_val))
qiblah_angle = (qiblah_angle + 360.0) % 360.0
```

Penggunaan fungsi `math.atan2(y, x)` merupakan kunci dari stabilitas program ini. Berbeda dengan fungsi `atan()` standar yang sering kali keliru menempatkan sudut pada kuadran yang salah (menghasilkan ambiguitas 180°), `atan2` secara otomatis mengevaluasi tanda dari sumbu x dan y sehingga mengembalikan azimuth yang presisi dalam rentang $-\pi$ hingga π radian (Meeus, 1998). Operasi modulo `(qiblah_angle + 360.0) % 360.0` memastikan bahwa keluaran akhir (output) selalu berupa representasi sudut kompas navigasi positif (0° hingga 360°).

Selain memberikan output teks, perangkat lunak ini memetakan nilai azimuth tersebut dalam fungsi `show_qiblah_map()`, membangkitkan grid contour komprehensif di atas peta topografi dunia untuk memvisualisasikan medan medan garis lengkung Great Circle yang memusat di satu titik (Makkah).

7.2. Rashdul Kiblat Lokal (Qibla Time)

Rashdul Kiblat (atau Qibla Time) adalah momen spesifik dalam satu hari di mana azimuth Matahari atau azimuth bayangan Matahari berhimpit tepat dengan azimuth arah kiblat suatu lokasi. Fenomena ini menawarkan metode kalibrasi arah kiblat yang paling presisi dan praktis tanpa memerlukan instrumen optik yang rumit; cukup menggunakan sebuah tiang yang didirikan tegak lurus (istiwa) dan mengamati bayangannya pada jam yang telah dihitung (Susiknan, 2007).

Dalam arsitektur KHGT Times V8.0, modul ini tidak menggunakan pendekatan aproksimasi jam rata-rata, melainkan mengeksekusi pencarian crossing time (waktu persilangan) secara analitik-numerik berbasis larik (array) dengan resolusi tinggi.

7.2.1. Algoritma Pencarian Persilangan Azimuth Matahari dan Kiblat

Untuk menemukan momen rashdul kiblat, sistem harus terlebih dahulu mengantongi nilai azimuth kiblat mutlak (θ_Q) dari lokasi pengamat, sebagaimana telah dihitung pada Sub-bab 7.1. Selanjutnya, sistem akan membangkitkan data pergerakan azimuth Matahari (θ_{sun}) sepanjang hari dari waktu terbit hingga terbenam.

Momen rashdul kiblat terjadi apabila memenuhi salah satu dari dua kondisi geometris berikut:

1. Matahari Berhimpit dengan Kiblat: $\theta_{sun} = \theta_Q$. Pada kondisi ini, jika

pengamat menghadap tepat ke arah Matahari, maka ia sedang menghadap ke arah Ka'bah. Bayangan tiang tegak akan jatuh menjauhi Ka'bah. 2. Bayangan Matahari Berhimpit dengan Kiblat: $\theta_{sun} = (\theta_Q \pm 180^\circ)$

$\text{mod } 360^\circ$. Pada kondisi ini, bayangan tiang tegak akan jatuh tepat mengarah ke Ka'bah.

Tantangan komputasi muncul karena fungsi pergerakan azimuth Matahari bersifat non-linear dan nilai azimuth memiliki diskontinuitas (loncatan nilai) pada transisi 360° ke 0° (saat Matahari melintasi Utara sejati). Oleh karena itu, sekadar mencari selisih nol secara naif ($\theta_{\text{sun}} - \theta_{\text{target}} = 0$) sering kali menghasilkan error pemrograman (wrap-around error).

KHGT Times memecahkan anomali ini melalui formulasi phase difference (selisih fase melingkar) yang aman, dikombinasikan dengan deteksi perubahan tanda (sign change detection):

```
# target_az adalah azimuth kiblat atau azimuth bayangan kiblat
# az_deg_list adalah array azimuth matahari sepanjang hari
diffs = (az_deg_list - target_az + 180) % 360 - 180
# Deteksi Persilangan (Crossing)
for i in range(len(diffs)-1):
    if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
        # ... [Proses interpolasi waktu pada indeks 'i'] ...
```

Operasi $(x + 180) \% 360 - 180$ memaksa nilai selisih sudut selalu berada dalam rentang terpendek -180° hingga $+180^\circ$. Dengan demikian, jika Matahari bergerak melewati Utara (misal dari 359° ke 1°), selisihnya tidak akan dibaca sebagai loncatan drastis -358° , melainkan terukur konsisten sebagai perubahan $+2^\circ$ (Meeus, 1998). Jika variabel `diffs` berubah tanda (dari negatif ke positif atau sebaliknya), maka dipastikan Matahari tepat menyilang titik target pada interval waktu tersebut.

7.2.2. Implementasi Pencarian Waktu Berbasis Array (NumPy Linspace)

Setelah indeks persilangan ditemukan, sistem tidak serta-merta mengambil waktu pada indeks tersebut karena resolusi array (meskipun rapat) masih memiliki celah antar-sampel. KHGT Times V8.0 membangkitkan 1440 sampel waktu (setiap 1 menit) menggunakan NumPy, yang berarti ada rentang 60 detik di antara `diffs[i]` dan `diffs[i+1]`.

Untuk menemukan fraksi detik yang tepat, perangkat lunak ini mengeksekusi interpolasi linier proporsional:

```
tt_array = np.linspace(t0.tt, t1.tt, 1440)
# ... [proses deteksi diffs] ...
if alt_deg_list[i] > 0: # Memastikan matahari di atas ufuk
    frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
    tt_res = tt_array[i] + frac * (tt_array[i+1] - tt_array[i])
```

Variabel `frac` merepresentasikan letak proporsional titik persilangan di antara dua sampel waktu. Konstanta $1e-9$ ditambahkan pada penyebut (denominator) sebagai teknik defensive programming untuk mencegah galat "pembagian dengan nol" (`ZeroDivisionError`) apabila kedua nilai `diffs` secara kebetulan bernilai mutlak nol absolut (Virtanen et al., 2020).

Hasil dari variabel `tt_res` kemudian dikonversi kembali menjadi waktu sipil lokal. Pendekatan ini menjamin bahwa output jam, menit, dan detik pada fitur Rashdul Kiblat di modul terkait KHGT Times bukan sekadar pembulatan kasar, melainkan hasil perhitungan interpolasi yang valid hingga akurasi fraksi detik, memastikan bayangan Matahari benar-benar terkunci pada orientasi Ka'bah.

7.3. Kalkulator Mizwala (Tongkat Istiwa) dan Analisis Vektor Bayangan Matahari

Sejak era keemasan astronomi Islam, Mizwala (sundial) atau Tongkat Istiwa (gnomon) telah menjadi instrumen observasional primer untuk penentuan waktu shalat berbasis pergerakan bayangan harian (King, 2004). Meskipun kalender dan jadwal waktu shalat modern telah terkomputasi dengan tingkat presisi mikrosekon, keberadaan instrumen bayangan tetap krusial sebagai metode verifikasi lapangan (ground truth calibration), khususnya untuk menentukan Rashdul Kiblat (waktu di mana bayangan benda tegak lurus mengarah secara presisi ke Ka'bah). Pada modul terkait dalam arsitektur KHGT Times V8.0, konsep klasik ini didigitalisasi melalui "Kalkulator Mizwala" yang memodelkan vektor bayangan tiga dimensi secara waktu nyata (real-time).

7.3.1. Formulasi Trigonometri Panjang dan Arah Bayangan

Berbeda dengan perhitungan waktu shalat standar yang umumnya hanya mengandalkan data ketinggian matahari (Altitude), simulasi Mizwala menuntut pemrosesan simultan antara Ketinggian dan Azimuth matahari lokal yang telah dikoreksi terhadap parameter refraksi atmosfer (Meeus, 1998).

Secara matematis, panjang bayangan (S) dari sebuah tongkat vertikal atau gnomon dengan tinggi absolut (h) diformulasikan menggunakan prinsip trigonometri dasar:

$$S = h / \tan(a)$$

Di mana (a) adalah sudut Ketinggian (Altitude) Matahari semu. Namun, dalam arsitektur kode sumber KHGT Times, formulasi ini diekspansi untuk menangani singularitas matematis. Pada saat matahari terbit atau terbenam, nilai Altitude (a) mendekati 0 derajat, sehingga nilai tangen mendekati nol dan menyebabkan kalkulasi panjang bayangan menuju tak terhingga (infinity).

Algoritma sistem mengimplementasikan batasan asimtotik numerik untuk mencegah buffer overflow pada memori komputasi saat merender grafis.

Untuk penentuan arah bayangan (Azimuth Bayangan), sistem menggunakan relasi kebalikan eksak dari posisi azimuth matahari di bola langit lokal: $\text{Azimuth_Bayangan} = (\text{Azimuth_Matahari} + 180^\circ) \bmod 360^\circ$

7.3.2. Penentuan Presisi Rashdul Kiblat Harian dan Tahunan

Fungsi paling kritis dari modul Mizwala digital ini adalah kalkulasi otomatis Rashdul Kiblat. Sistem tidak memprediksinya melalui rata-rata statistik, melainkan mendeteksi momentum tersebut melalui pencarian akar algoritma (root-finding algorithm) yang memindai detik demi detik kapan nilai Azimuth Matahari berhimpit presisi dengan Azimuth Kiblat lokal (Ilyas, 1997).

Protokol eksekusi dalam KHGT Times mencari dua kondisi terpisah:

1. Yaumul Kiblat Utama (Tahunan): Terjadi ketika Deklinasi Matahari sama persis dengan

Lintang geografis Ka'bah (sekitar 21.42° LU), yang umumnya jatuh pada tanggal 28 Mei dan 16 Juli. Pada kondisi ini, bayangan dari setiap benda tegak di seluruh belahan bumi yang mengalami siang hari akan mengarah lurus menuju Makkah. 2. Rashdul Kiblat Harian: Waktu spesifik di hari apa pun sepanjang tahun di mana Azimuth

Matahari sama dengan Azimuth Kiblat lokal, atau berselisih tepat 180 derajat.

7.3.3. Arsitektur Kode dan Koreksi Toposentris

Dalam implementasinya menggunakan Python dan Skyfield, sistem sama sekali tidak memodelkan bumi sebagai bola bundar sempurna, melainkan sebagai Oblate Spheroid berdasarkan parameter geodetik WGS84. Hal ini menjadi syarat wajib karena kemiringan tongkat istiwa absolut di permukaan bumi dunia nyata akan terpengaruh secara mikro oleh kelengkungan ekuatorial.

```
def calculate_mizwala_shadow(observer_loc, t_current, gnomon_height):
    # Mengambil posisi toposentris semu (diperkuat oleh refraksi atmosfer lokal)
    sun_apparent = observer_loc.at(t_current).observe(sun).apparent()
    alt, az, distance = sun_apparent.altaz()
    alt_deg = alt.degrees
    az_deg = az.degrees
    # Pencegahan matematis pembagian dengan nol saat matahari menembus horizon
    if alt_deg <= 0:
        return None, None
    import math
    # Kalkulasi panjang bayangan aktual
    shadow_length = gnomon_height / math.tan(math.radians(alt_deg))
    # Kalkulasi azimuth arah bayangan
    shadow_azimuth = (az_deg + 180.0) % 360.0

    return shadow_length, shadow_azimuth
```

Struktur kode di atas dieksekusi oleh mesin sistem dalam looping kontinu berkecepatan tinggi, yang memungkinkan modul simulasi GUI untuk memproyeksikan animasi pergerakan bayangan tongkat melintasi piringan penunjuk (dial plate) dengan latensi sangat rendah. Integrasi algoritmik ini berhasil menjembatani tingkat akurasi ephemeris saintifik modern dengan kemurnian instrumen observasional yang telah menjadi warisan intelektual peradaban Islam.

DAFTAR PUSTAKA BAB 7

- Abdali, S. K. (1997). *The correct qibla*. American Trust Publications.
- Duffett-Smith, P., & Zwart, J. (2011). *Practical Astronomy with your Calculator or Spreadsheet* (4th ed.). Cambridge University Press.
- Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.
- Ilyas, M. (1997). *Islamic Calendar, Times & Qibla*. Berita Publishing.
- King, D. A. (1986). *Islamic Mathematical Astronomy*. Variorum Reprints.
- King, D. A. (2004). In *Synchrony with the Heavens: Studies in Astronomical Timekeeping and Instrumentation in Medieval Islamic Civilization*. Brill.
- Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.
- Rhodes, B. (2019). *Skyfield: High precision research-grade positions for planets and Earth satellites*. Astrophysics Source Code Library, ascl:1907.024.
- Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.
- Torge, W., & Müller, J. (2012). *Geodesy* (4th ed.). De Gruyter.
- Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

BAB 8: MEKANIKA SYZYG: GERHANA DAN ANALEMMA

Selain menentukan awal bulan dan jadwal waktu salat, komputasi astronomi Islam modern juga dihadapkan pada tuntutan untuk memprediksi fenomena fisis luar biasa yang bernilai ibadah (seperti Salat Khusuf dan Kusuf), serta memodelkan lintasan geometri matahari secara komprehensif. Bab ini akan menguraikan bagaimana Modul Analisis Gerhana dan Generator Analemma pada KHGT Times V8.0 memanfaatkan mesin Skyfield untuk memecahkan kompleksitas mekanika benda langit tingkat lanjut.

8.1. Deteksi Otomatis Gerhana Matahari dan Bulan

Fenomena gerhana (baik matahari maupun bulan) terjadi secara periodik ketika Matahari, Bumi, dan Bulan berada dalam satu garis lurus secara tiga dimensi, suatu keadaan yang dalam astronomi dinamakan syzygy (Espanak & Meeus, 2006). Namun, tidak setiap syzygy (ijtimak/purnama) menghasilkan gerhana karena orbit Bulan miring sekitar 5.14° terhadap bidang orbit Bumi (ekliptika). Gerhana hanya terjadi ketika syzygy bertepatan dengan titik simpul orbit bulan (lunar node).

8.1.1. Mencari Jarak Sudut Minimum (Separation) di Titik Simpul

Pustaka astronomi klasik umumnya menggunakan persamaan Bessel (Besselian elements) untuk meramalkan gerhana. KHGT Times V8.0 mengambil pendekatan numerik langsung yang lebih tangguh dan mudah diimplementasikan, yaitu mencari jarak sudut geometris absolut (angular separation) terdekat antara piringan Matahari dan Bulan pada waktu-waktu kritis.

Untuk Gerhana Matahari, sistem pertama-tama mencari semua waktu ijtimak (New Moon) di sepanjang tahun. Pada setiap titik ijtimak, program membangkitkan larik waktu resolusi sangat tinggi (pm 0.25 hari dari pusat ijtimak) dan menghitung jarak sudut (separation) geosentris secara iteratif:

```
tt_array = np.linspace(t_nm.tt - 0.25, t_nm.tt + 0.25, 300)
t_arr = self.ts.tt_jd(tt_array)
e_pos = earth.at(t_arr)
# Menghitung sudut separasi (jarak) Matahari dan Bulan
seps =
e_pos.observe(sun).apparent().separation_from(e_pos.observe(moon).apparent())
.degrees
min_idx = np.argmin(seps)
```

Jika `seps[min_idx]` lebih kecil dari ambang batas toleransi (sekitar 1.6° , yakni jumlah jari-jari Matahari, jari-jari Bulan, ditambah toleransi paralaks maksimum), maka dipastikan telah terjadi kontak piringan (gerhana matahari) (Meeus, 1998).

8.1.2. Kalkulasi Geometri Umbra dan Penumbra Bumi

Untuk Gerhana Bulan, prinsipnya serupa namun secara geometris lebih kompleks karena melibatkan proyeksi bayangan kerucut Bumi ke jarak orbit Bulan. Berbeda dengan gerhana matahari, visibilitas gerhana bulan tidak bergantung pada paralaks pengamat (gerhana bulan terlihat sama di seluruh wilayah bumi yang sedang mengalami malam hari). Oleh karena itu, perhitungan dilakukan secara murni Geosentris.

Alih-alih mencari separasi matahari dan bulan, sistem mencari titik potong antara posisi Bulan dan titik Anti-Matahari (posisi bayangan bumi).

```
# Koordinat Matahari
s_ra, s_dec, s_dist = s_geo.radec()
# Membalik vektor matahari 180 derajat (Titik Anti-Matahari / Pusat Bayangan Bumi)
as_ra = (s_ra.hours + 12.0) % 24.0
as_dec = -s_dec.degrees
# Jarak sudut bulan terhadap pusat bayangan bumi
d_ra = (m_ra.hours - as_ra) * 15.0 * math.cos(math.radians(as_dec))
```

```
d_dec = m_dec.degrees - as_dec
sep = math.sqrt(d_ra**2 + d_dec**2)
```

Selanjutnya, program menghitung jari-jari bayangan gelap Bumi (Umbra) dan bayangan samar (Penumbra) pada jarak Bulan menggunakan trigonometri dasar berbasis radius ekuatorial bumi (6378.14 km):

```
•
r_umbra = (p_m + p_s - s_s) * 1.02
•
r_penumbra = (p_m + p_s + s_s) * 1.02
```

Keterangan Simbol:

- r_umbra: Jari-jari bayangan inti (Umbra).
- r_penumbra: Jari-jari bayangan tambahan (Penumbra).
- p_m: Paralaks horizontal Bulan (Moon's horizontal parallax).
- p_s: Paralaks horizontal Matahari (Sun's horizontal parallax).
- s_s: Semidiameter atau jari-jari sudut Matahari.
- 1.02: Faktor koreksi (sekitar 2%) untuk memperhitungkan pengaruh atmosfer bumi yang memperbesar bayangan.

Penjelasan Fungsi:

Di mana p adalah paralaks ekuatorial horizontal dan s adalah semi-diameter (jari-jari piringan). Konstanta 1.02 ditambahkan sebagai empirisme astronomis karena atmosfer bumi memperbesar ukuran fisik bayangan umbra sekitar 2% (Espenak & Meeus, 2006). Jika sep lebih kecil dari r_umbra, maka bulan memasuki fase gerhana umbral (sebagian/total).

8.2. Proyeksi Jalur Totalitas (Export KML)

Salah satu fitur paling kompleks dari KHGT Times V8.0 adalah kemampuannya men- generate secara otonom dokumen Google Earth (.kml) yang mendelineasi jalur gerhana total di permukaan bumi.

8.2.1. Vektor Posisi 3D (Center Line) Bayangan Bulan di Permukaan Bumi

Pada gerhana matahari total/cincin, titik pusat bayangan umbra bulan (Sumbu Silinder Bulan) memotong permukaan bumi membentuk lintasan melengkung. Untuk menemukan titik potong tiga dimensi ini, KHGT Times menggunakan representasi vektor xyz spasial murni.

```
1. Program menghitung vektor posisi pusat Bulan (M) dan pusat Matahari (S).
2. Mencari vektor arah sinar bayangan (V = M - S) dan melakukan normalisasi (v_hat).
3. Permukaan bumi dimodelkan sebagai bola dengan jari-jari R_E (rata-rata geometris).
Garis
sumbu bayangan dapat ditulis sebagai persamaan parametrik: P(k) = M + \v_hat * k.
Persamaan ini disubstitusikan ke dalam persamaan bola |P|^2 = R_E^2, menghasilkan
persamaan kuadratik Delta:
V = M - S
v_hat = V / np.linalg.norm(V, axis=0)
b = 2.0 * np.sum(M * v_hat, axis=0)
c = np.sum(M**2, axis=0) - R_E**2
delta = b**2 - 4.0 * c
```

Hanya titik waktu dengan delta ≥ 0 yang menandakan bahwa bayangan inti (umbra) bulan "menghantam" daratan atau samudra di Bumi (jika negatif, bayangan meleset ke luar angkasa) (Urban & Seidelmann, 2013).

8.2.2. Konversi Subpoint Geocentric ke File Google Earth

Titik potong P yang masih dalam format ruang Cartesian (X, Y, Z) dikonversi kembali ke koordinat geografis lintang dan bujur permukaan ellipsoid menggunakan model WGS84 dari Skyfield:

```
geo = Geocentric(position_au=P_au[:, i], t=self.ts.tt_jd(t_valid[i]))
subpt = wgs84.subpoint(geo)
coords_list.append(f"{subpt.longitude.degrees},{subpt.latitude.degrees},0")
```

Rangkaian koordinat ini kemudian dibungkus menggunakan sintaks Markup Language XML (KML) dengan tag <LineString>. Output akhir dari proses algoritmik ini adalah file yang dapat langsung di-drag and drop ke aplikasi Google Earth Pro, merepresentasikan jalur bayangan gerhana (Path of Totality) yang siap digunakan para pengejar gerhana (eclipse chasers).

8.3. Generator Analemma Matahari

Analemma adalah kurva geometris menyerupai angka 8 yang dibentuk oleh posisi Matahari di langit, apabila difoto dari titik kordinat bumi yang sama, tepat pada waktu jam yang sama (misal 12:00 siang lokal), selama 365 hari dalam setahun. Fenomena analemma terjadi akibat dua asimetri orbital: kemiringan sumbu rotasi bumi (23.4°) yang mendikte perubahan Altitude, dan eksentrisitas orbit bumi (elips) yang mendikte perubahan Azimuth karena fluktuasi Equation of Time (Hughes et al., 1989).

8.3.1. Ekstraksi Altitude dan Azimuth pada Jam Tetap Selama 365 Hari

Untuk membangun grafik analemma, KHGT Times V8.0 mengatur iterasi ephemeris sebanyak 365 hari dengan mempertahankan konstan variabel jam (`h_local`). Pilihan `grid_step` sebesar 5 hari (sekitar 73 titik sampel per tahun) dipilih sebagai optimasi kecepatan render Matplotlib tanpa mengorbankan kualitas kurva angka 8 yang terbentuk.

```
for day in range(0, 365, 5):
    date_dt = datetime.datetime(y, 1, 1) + datetime.timedelta(days=day)
    # Waktu dipatok konstan pada h_local
    t = self.ts.utc(y, date_dt.month, date_dt.day, h_local - tz)
    astrometric = loc.at(t).observe(sun).apparent()
    alt, az, _ = astrometric.altaz()
8.3.2. Analisis Titik Ekstrem (Solstice) dan Ekuator (Equinox)
```

Selain sekadar mencetak titik, mesin ini diinstruksikan untuk mendeteksi parameter fisis batas maksimum dan minimum (solstis) menggunakan NumPy:

```
idx_max = np.argmax(alts) # Summer Solstice (Utara)
idx_min = np.argmin(alts) # Winter Solstice (Selatan)
```

Metode pembacaan pola matriks ini menjadikan Modul Analemma KHGT Times sebagai instrumen edukasi astronomi yang sangat representatif untuk memahami konsep Mean Solar Time, di mana rentang lebar horizontal dari kurva analemma merepresentasikan durasi kompensasi menit dari nilai Equation of Time yang berfluktuasi sepanjang tahun.

DAFTAR PUSTAKA BAB 8

Esenak, F., & Meeus, J. (2006). *Five Millennium Canon of Solar Eclipses: -1999 to +3000*. NASA Goddard Space Flight Center.

Hughes, D. W., Yallop, B. D., & Hohenkerk, C. Y. (1989). The equation of time. *Monthly Notices of the Royal Astronomical Society*, 238(4), 1529-1535.

Meeus, J. (1998). *Astronomical Algorithms* (2nd ed.). Willmann-Bell.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

BAB 9: SIMULATOR 3D DAN ANIMASI INTERAKTIF

Pembaruan V8.0 pada Live Animasi merupakan akumulasi pengembangan V7.5-V7.6: visualisasi rasi bintang dan planet, Tracker benda langit dengan atau tanpa GPS, serta tombol Sinkronisasi Stellarium untuk membandingkan parameter Matahari-Bulan dengan hasil kalkulasi internal yang memakai kernel JPL terpilih.

Representasi data astronomi dalam bentuk tabel numerik (ephemeris) sering kali memiliki keterbatasan dalam membangun intuisi visual, terutama ketika fenomena yang diamati melibatkan interaksi spasial yang kompleks seperti gerhana, transit planet, atau posisi hilal di atas cakrawala 360 derajat. Mengatribusikan nilai asensio rekta, deklinasi, altitude, dan azimuth ke dalam sebuah kanvas visual tiga dimensi merupakan jembatan kognitif yang sangat esensial (Fluke et al., 2006).

Oleh karena itu, KHGT Times V8.0 tidak hanya berfungsi sebagai mesin penghitung angka, tetapi juga dilengkapi dengan Graphical User Interface (GUI) yang memuat modul Simulator 3D dan Animasi Interaktif. Modul ini secara fundamental mentransformasikan larik data spasial menjadi model fisika visual yang beroperasi secara real-time.

9.1. Transformasi Vektor ke Ruang 3D (Matplotlib)

Membangun lingkungan tiga dimensi dari data astronomi bola mensyaratkan pemahaman yang solid mengenai aljabar linear dan geometri spasial. KHGT Times V8.0 memanfaatkan library Matplotlib untuk melakukan rendering grafis. Matplotlib pada dasarnya adalah pustaka grafik 2D, namun ia memiliki kapabilitas ekstensi mplot3d untuk memproyeksikan data 3D. Menariknya, pada modul Geocentric Space System, pengembang aplikasi merancang mesin proyeksi (projection engine) dan kalkulasi Z-Buffer secara mandiri murni menggunakan fungsi trigonometri dasar tanpa bergantung sepenuhnya pada modul terkaitD bawaan Matplotlib, guna mencapai efisiensi rotasi (rendering speed) yang lebih tinggi.

9.1.1. Konversi Koordinat Alt/Az ke Cartesian (X, Y, Z)

Pada Menu 37 (Simulasi Ephemeris 3D), pengguna disajikan representasi toposentris—yakni bola langit yang berpusat pada mata pengamat di permukaan bumi. Output dari mesin astrometri Skyfield berupa besaran Ketinggian (Altitude, Alt) dan Azimuth (Az). Agar titik-titik bulan dan matahari dapat diletakkan di dalam ruang kanvas Cartesian 3D, koordinat bola (spherical coordinates) tersebut harus direduksi menjadi sumbu ortogonal x, y, z.

Dalam astronomi, jika kita asumsikan pengamat berada di titik origin (0,0,0), sumbu y positif mengarah ke Utara sejati ($Az = 0^\circ$), sumbu x positif mengarah ke Timur ($Az = 90^\circ$), dan sumbu z mewakili titik Zenith ($Alt = 90^\circ$), maka persamaan transformasinya adalah (Duffett-Smith & Zwart, 2011):

$$x = r \sin(Az) \cos(Alt)$$

$$y = r \cos(Az) \cos(Alt)$$

$$z = r \sin(Alt)$$

Persamaan ini diterjemahkan secara presisi ke dalam blok kode internal KHGT Times:

```
def eph3d_r_ke_xyz(self, alt, az, r=10):
    import math
    alt_rad = math.radians(alt)
    az_rad = math.radians(az)
    x = r * math.sin(az_rad) * math.cos(alt_rad)
    y = r * math.cos(az_rad) * math.cos(alt_rad)
    z = r * math.sin(alt_rad)
    return x, y, z
```

Variabel jari-jari bola langit imajiner (r) ditetapkan secara statis sebesar 10 satuan absolut, mengingat dalam proyeksi pandangan toposentris lokal, pengamat memandang langit seolah-olah semua benda kosmik (bintang, bulan, matahari) menempel pada dinding kubah dengan jarak seragam tak terhingga (celestial sphere). Setelah dikonversi, titik koordinat spasial tersebut diumpankan ke objek Matplotlib melalui fungsi interpolatif `.set_3d_properties()`.

9.1.2. Proyeksi Kamera Kustom (Z-Buffer dan Pseudo-3D Rotation)

Kendala utama Matplotlib dalam memproses animasi beresolusi tinggi adalah beban overhead saat merender ulang seluruh komponen figure 3D setiap kali layar diputar menggunakan mouse. Untuk mengatasi limitasi frame rate pada modul Sistem Tata Surya Geosentris, KHGT Times menggunakan Matplotlib sebagai kanvas 2D biasa (menggunakan widget Canvas Tkinter) namun menyuntikkan algoritma proyeksi perspektif pseudo-3D (Pseudo-3D Projection) secara matematis.

Ketika pengguna mengklik dan menggeser layar (mouse drag), sistem tidak merotasi Matplotlib axes. Sistem memperbarui sudut kamera (θ_x dan θ_y) dan melakukan perkalian matriks rotasi Euler pada setiap vektor posisi planet (x, y, z). Proyeksi transformasi ortografi ke bidang monitor 2D (px, py) dihitung melalui persamaan yang memperhitungkan faktor kedalaman (depth factor), sehingga objek yang menjauh terlihat mengecil secara alami:

Persamaan Transformasi Rotasi Koordinat

1. Rotasi terhadap Sumbu X (Tahap 1):

$$\begin{aligned} x1 &= x \cdot \cos(\theta_x) - z \cdot \sin(\theta_x) \\ z1 &= x \cdot \sin(\theta_x) + z \cdot \cos(\theta_x) \end{aligned}$$

2. Rotasi terhadap Sumbu Y (Tahap 2):

$$\begin{aligned} y2 &= y \cdot \cos(\theta_y) - z1 \cdot \sin(\theta_y) \\ z2 &= y \cdot \sin(\theta_y) + z1 \cdot \cos(\theta_y) \end{aligned}$$

Keterangan:

- x, y, z : Koordinat awal (input).
- θ_x, θ_y : Sudut rotasi pada sumbu x dan y .
- $x1, z1$: Hasil antara (intermediate) setelah rotasi pertama.
- $y2, z2$: Hasil akhir setelah rotasi kedua.

```
factor = 500 / (500 + z2)
px = x1 * factor + (width / 2)
py = -y2 * factor + (height / 2)
return px, py, z2
```

Lebih krusial lagi, untuk memastikan benda langit yang berada di depan menutupi benda yang berada di belakang (misalnya saat Bulan berada di depan Bumi), sistem mengimplementasikan varian Algoritma Pelukis (Painter's Algorithm) melalui manajemen Z-Buffer mandiri (Hughes et al., 2014).

Semua objek ditampung ke dalam senarai (`draw_list`) bersama dengan nilai kedalaman akhirnya (z_2). Sebelum dirender ke layar antarmuka, list tersebut diurutkan secara menurun (descending) berdasarkan properti z :

```
# Algoritma Z-Buffer: Urutkan objek dari yang terjauh ke yang terdekat
draw_list.sort(key=lambda x: x[1][2], reverse=True)
```

Dengan mengurutkannya, mesin Matplotlib akan melukis objek yang terjauh (latar belakang kosmik) terlebih dahulu, dan melukis objek terdekat pada akhir iterasi, menimpali piksel gambar sebelumnya dengan sempurna. Optimalisasi komputasi grafis inilah yang menjamin kelancaran

interaksi 60 FPS pada simulator tata surya KHGT Times, bahkan pada arsitektur komputer berspesifikasi rendah.

9.2. Live Tracker Toposentris (Bola Langit Pengamat)

Transformasi visual dari ruang 3D konseptual (geosentris) menuju ruang observasi praktis memerlukan pergeseran kerangka acuan (reference frame). Modul Live Animasi pada KHGT Times V8.0 dirancang untuk mensimulasikan bola langit (celestial sphere) murni dari perspektif toposentris, yakni dari mata pengamat di permukaan bumi. Sistem ini melukiskan garis horizon, letak azimut, serta melacak pergerakan Matahari dan Bulan dalam waktu waktu nyata (real-time), menjadikannya instrumen virtual yang setara dengan fungsi planetarium modern (Gargano et al., 2015).

9.2.1. Sinkronisasi Data Astronomi dengan Jam Sistem (Real-time Threading)

Tantangan utama dalam membangun live tracker adalah menyinkronkan waktu sistem operasi (komputer) dengan skala waktu dinamis ephemeris (Terrestrial Time/TT) tanpa menyebabkan antarmuka pengguna (Graphical User Interface / GUI) menjadi tidak responsif atau membeku (freeze).

Alih-alih menggunakan blocking loop (seperti `while True`), KHGT Times memanfaatkan arsitektur pemrograman event-driven bawaan Tkinter melalui metode `.after()`. Pemanggilan ini menugaskan sistem operasi untuk mengeksekusi ulang fungsi pembaruan layar (`update_animation`) setiap 1000 milidetik (1 detik), menghasilkan animasi dengan laju 1 Frame Per Second (FPS) yang sangat hemat daya CPU.

Pada setiap detiknya, sistem menangkap waktu absolut dan melakukan komputasi posisi:

```
if getattr(self, 'anim_is_live', True):
    t_sim = self.ts.now() # Mengambil waktu sistem real-time ke dalam objek
    Skyfield
else:
    t_sim = self.anim_custom_time # Mode playback/kustom waktu
    # Evaluasi Toposentrik (Koreksi paralaks terhitung otomatis)
    observer = earth + wgs84.latlon(lat, lon)
    astro_sun = observer.at(t_sim).observe(sun).apparent()
    alt_sun, az_sun, _ = astro_sun.altaz()
    astro_moon = observer.at(t_sim).observe(moon).apparent()
    alt_moon, az_moon, _ = astro_moon.altaz()
```

Nilai Altitude dan Azimuth yang dihasilkan secara presisi kemudian dipetakan ke dalam kanvas 2D (`self.anim_canvas`). Sistem membagi lebar layar menjadi 360° bujur langit. Menariknya, sistem ini juga mengalkulasi ketinggian ufuk (horizon) dinamis dan menyuntikkan efek refraksi geometris dengan menambahkan nilai statis +0.833° pada ketinggian piringan matahari (`alt_top_sun = alt_s_deg + 0.833`), memastikan bahwa piringan matahari tenggelam tepat di garis ufuk GUI pada saat waktu Maghrib tiba, merepresentasikan ukuran semi-diameter matahari dan refraksi atmosfer aktual (Schaefer, 1993).

9.2.2. Render Piringan Matahari dan Fase Bulan Dinamis (Pillow Masking)

Bagian paling revolusioner dari modul Live Tracker KHGT Times V8.0 adalah pelukisan (rendering) wujud bulan. Bulan tidak sekadar digambar sebagai lingkaran putih statis, melainkan dirender sesuai dengan fase iluminasinya dan rotasi kemiringan sabitnya relatif terhadap matahari di langit.

Fase bulan (k) dikalkulasi secara matematis melalui selisih bujur ekliptika (λ) antara Bulan dan Matahari:

$$\text{Phase_Angle} = (\lambda_{\text{moon}} - \lambda_{\text{sun}}) \bmod 360^\circ$$

Keterangan Simbol:

- **Phase_Angle / Sudut_Fase:** Sudut yang menentukan fase bulan (digunakan untuk mengetahui posisi bulan terhadap matahari).
- λ_{moon} (λ_{bulan}): Bujur ekliptika Bulan (Ecliptic Longitude of the Moon).
- λ_{sun} ($\lambda_{\text{matahari}}$): Bujur ekliptika Matahari (Ecliptic Longitude of the Sun).
- **mod 360°:** Operasi modulo (sisa bagi) untuk memastikan hasil perhitungan sudut selalu berada dalam rentang 0° hingga 360°.

Untuk merender gambar sabit secara halus, program menggunakan pustaka manipulasi citra Pillow (PIL). Sistem memuat himpunan sprite gambar bulan (m000.png hingga m359.png), dan memilih gambar yang tepat berdasarkan sudut fase bulat yang dihasilkan. Namun, memuat gambar saja tidak cukup; garis terminasi (batas terang-gelap bulan) harus miring tegak lurus mengarah langsung ke pusat piringan matahari.

Di sinilah KHGT Times V8.0 mengeksekusi trigonometri 2D Cartesian untuk mencari sudut kemiringan (tilt angle) antara koordinat piksel matahari (x_{sun} , y_{sun}) dan bulan (x_{moon} , y_{moon}):

```
dx = x_sun - x_moon
dy = y_sun - y_moon
# Menghitung sudut kemiringan dalam radian lalu dikonversi ke derajat
sudut_kemiringan = math.degrees(math.atan2(dy, dx))
if moon_idx >= 180:
    sudut_kemiringan -= 180
# Merotasi gambar bulan dengan algoritma interpolasi kualitas tinggi (Bicubic)
moon_img
=
moon_img.rotate(-sudut_kemiringan,
resample=Image.Resampling.BICUBIC)
# Teknik Alpha Masking agar background bulan transparan
mask = Image.new('L', (target_size, target_size), 0)
draw = ImageDraw.Draw(mask)
draw.ellipse((0, 0, target_size - 1, target_size - 1), fill=255)
moon_img.putalpha(mask)
```

Fungsi `math.atan2(dy, dx)` memastikan sudut kemiringan terbaca di keempat kuadran spasial secara akurat. Penggunaan interpolasi Bicubic (`Image.Resampling.BICUBIC`) pada saat rotasi mencegah piksel gambar bulan menjadi pecah (aliasing). Selanjutnya, proses Alpha Masking memotong tepi gambar yang berbentuk persegi menjadi lingkaran transparan sempurna, memungkinkannya digambar tumpang-tindih (overlay) di atas kanvas bintang-bintang (starfield) tanpa meninggalkan kotak latar belakang hitam.

Teknik komputasi grafis hibrida ini—yang memadukan keakuratan Skyfield dan ketajaman optis Pillow—menghasilkan simulasi visibilitas hilal yang sangat realistis. Ini memberikan kemudahan luar biasa bagi perukyat (observator) untuk mengetahui di mana letak persis dan bagaimana bentuk orientasi sabit hilal di langit sesaat setelah sunset tiba.

9.3. Simulasi Tata Surya Geosentris

Meskipun model heliosentris (Matahari sebagai pusat) merepresentasikan realitas fisik tata surya kita, seluruh observasi astronomi Islam—baik hisab awal bulan maupun jadwal waktu salat—secara mutlak bertumpu pada kerangka acuan geosentris (Bumi sebagai pusat). Modul terkait pada KHGT Times V8.0 mendesain ulang arsitektur rendering Matplotlib untuk menciptakan "Simulator Tata Surya Geosentris", sebuah ruang virtual interaktif di mana Bumi ditetapkan secara statis di titik origin koordinat (0, 0, 0), sementara Bulan, Matahari, dan planet-planet lainnya mengorbit mengelilinginya (Montenbruck & Pfleger, 2000).

9.3.1. Kompresi Spasial dan Skala Jarak Logaritmik (Non-Linear Scaling)

Tantangan komputasi visual terbesar dalam membangun simulator geosentris adalah perbedaan skala jarak yang ekstrem. Jarak rata-rata Bumi ke Bulan adalah sekitar 384.400 km, sedangkan jarak rata-rata Bumi ke Matahari (1 Astronomical Unit) adalah sekitar 149.597.870 km. Rasio jarak ini

hampir mencapai 1:400. Jika sistem merender objek-objek ini ke dalam layar komputer menggunakan skala linier yang sebenarnya (1:1), maka saat piringan Matahari terlihat di batas layar, orbit Bulan akan mengecil menjadi kurang dari satu piksel dan tidak mungkin diobservasi oleh pengguna manusia (Shirley & Marschner, 2009).

Untuk memecahkan kebuntuan visual ini, KHGT Times mengimplementasikan algoritma Kompresi Skala Non-Linier (pseudo-logaritmik). Sistem menetapkan jarak visual Bulan sebagai standar konstan (misalnya $r = 10$ unit grafis), lalu secara matematis mengompresi jarak Matahari

dan planet lain sehingga mereka tampak berada pada jarak yang proporsional di layar, tanpa mengubah keakuratan sudut Asensio Rekta (RA) dan Deklinasi (Dec) mereka.

```
# Ekstraksi jarak absolut (km) dari mesin Skyfield
d_moon = moon_geo.distance().km
d_sun = sun_geo.distance().km
# Algoritma Kompresi Spasial untuk Visualisasi GUI
# Radius visual Bulan dipatok pada skala 10 unit grafis
vis_r_moon = (d_moon / 384400.0) * 10.0
# Jarak Matahari ditekan secara logaritmik agar masuk ke dalam field of view (FOV)
vis_r_sun = (math.log10(d_sun) / math.log10(149597870.0)) * 30.0
# Konversi kembali ke XYZ menggunakan sudut RA dan Dec yang tidak diubah
x_sun, y_sun, z_sun = self.spherical_to_cartesian(ra_sun, dec_sun, vis_r_sun)
```

Melalui pemisahan antara "Jarak Geometris (untuk kalkulasi ephemeris)" dan "Jarak Visual (untuk rendering layar)", simulator KHGT Times mampu menyajikan konfigurasi syzygy (ijtimak/purnama) yang dapat diobservasi mata dengan jelas dalam satu frame, menjaga nilai edukatif perangkat lunak tanpa mengorbankan fondasi fisika astrometrinya.

9.3.2. Rendering Jejak Orbit (Orbital Trails) dan Vektor Ekliptika

Benda langit tidak hanya digambar sebagai titik (titik koordinat saat itu), tetapi sistem juga mengkalkulasi dan melukiskan "jejak orbit" (orbital trails) untuk memvisualisasikan mekanika benda langit selama satu bulan penuh (satu lunasi). Hal ini sangat krusial untuk menjelaskan mengapa bulan tidak selalu menutupi matahari (gerhana) pada setiap fase ijtimak.

Untuk menghasilkan jejak orbit resolusi tinggi, sistem memicu fungsi multithreading yang meminta Skyfield membangkitkan senarai matriks posisi (linspace) selama ± 15 hari dari waktu simulasi saat ini:

```
# Membangkitkan array waktu mundur 15 hari dan maju 15 hari
t_array = self.ts.tt_jd(np.linspace(t_sim.tt - 15, t_sim.tt + 15, 300))
# Ekstraksi array vektor posisi geosentris
pos_moon_array = earth.at(t_array).observe(moon).apparent()
ra_arr, dec_arr, dist_arr = pos_moon_array.radec()
# Looping untuk mengonversi array ke garis grafis 3D
trail_coords = []
for i in range(len(ra_arr.hours)):
    x,
    y,
    z
    =
    self.spherical_to_cartesian(ra_arr.hours[i]*15,
    dec_arr.degrees[i], vis_r_moon)
    trail_coords.append((x, y, z))
```

Garis orbit Bulan kemudian diproyeksikan dengan ketebalan 1 piksel berwarna abu-abu, sementara bidang Ekliptika (orbit semu Matahari) diproyeksikan dengan garis kuning. Pertemuan antara garis abu-abu dan garis kuning di layar 3D menunjukkan posisi Lunar Nodes (Titik Simpul Naik/Turun).

Interaktivitas yang memungkinkan pengguna untuk memutar ruang 3D, memperbesar, dan melihat jarak pemisah di titik simpul orbit inilah yang menjadikan Modul Simulasi KHGT Times V8.0 sebagai mahakarya rekayasa perangkat lunak pedagogis. Pengguna tidak lagi "membayangkan" proses

konjungsi dari balik deretan angka, melainkan "menyaksikan" mekanika kosmik tersebut bekerja secara geometris.

DAFTAR PUSTAKA BAB 9

Duffett-Smith, P., & Zwart, J. (2011). *Practical Astronomy with your Calculator or Spreadsheet* (4th ed.). Cambridge University Press.

Fluke, C. J., Bourke, P. D., & O'Donovan, D. (2006). Future directions in astronomy visualization. *Publications of the Astronomical Society of Australia*, 23(1), 12-24.

Gargano, M., et al. (2015). Planetariums and digital domes: A new era for astronomical visualization. *Journal of Science Communication*, 14(2).

Hughes, J. F., et al. (2014). *Computer Graphics: Principles and Practice* (3rd ed.). Addison-Wesley Professional.

Montenbruck, O., & Pfleger, T. (2000). *Astronomy on the Personal Computer* (4th ed.). Springer-Verlag.

Schaefer, B. E. (1993). Astronomy and the limits of vision. *Vistas in Astronomy*, 36, 311-361.

Shirley, P., & Marschner, S. (2009). *Fundamentals of Computer Graphics* (3rd ed.). A K Peters/CRC Press.

9.4. Ekstensi Live Animasi pada Basis V8.0

9.4.1. Rasi Bintang dan Planet

Live Animasi menambahkan objek rasi bintang dan planet sebagai lapisan visual dinamis di atas cakrawala pengamat. Fitur ini melengkapi visual Matahari dan Bulan sehingga modul dapat digunakan sebagai ruang observasi simulatif, bukan sekadar pelacak dua benda langit.

9.4.2. Tracker Benda Langit

Tracker menyediakan akses pelacakan posisi benda langit dan dapat dipakai dalam skenario dengan GPS maupun tanpa GPS. Integrasi ini memperluas fungsi Live Animasi dari visualisasi lokal menjadi pintu masuk ke pelacakan berbasis lokasi.

9.4.3. Sinkronisasi Stellarium

Tombol Sinkronisasi Stellarium membuka alur perbandingan data Matahari dan Bulan antara hasil hitung KHGT Times dan output Stellarium. Pada V8.0, dokumentasi komparasi internal harus merujuk pada kernel JPL yang dipilih otomatis, bukan mengasumsikan satu file ephemeris tetap.

BAB 10: INTEGRASI KECERDASAN BUATAN DAN ANTARMUKA MULTIBAHASA

Pada V8.0, WinAI berfungsi sebagai lapisan interaksi yang menggabungkan data internal aplikasi, basis pengetahuan Tarjih, hasil pencarian pendukung di internet, dan model bahasa generatif. Dokumentasi bab ini sengaja dibatasi pada perilaku yang benar-benar tampak di source code V8.0; klaim mekanisme jaringan yang tidak tercermin pada implementasi tidak dipertahankan dari edisi sebelumnya.

10.1. Arsitektur WinAI pada Source Code V8.0

Antarmuka WinAI dibangun di dalam aplikasi desktop dan menyediakan kotak pertanyaan, tanggal tinjauan, ruang keluaran, fungsi salin, bagikan, log teks, serta reset memori. Model utama yang ditetapkan di source code adalah gemini-2.5-flash. Permintaan dikirim melalui HTTP POST ke endpoint generative language. Sistem mempertahankan indeks kunci terakhir yang berhasil digunakan dan beralih ke kunci berikutnya bila menerima respons tertentu seperti 429, 403, atau 400. Dalam edisi publikasi ini, nilai kunci tidak dicetak.

```
self.winai_primary_model = 'gemini-2.5-flash'
# API keys disimpan pada konfigurasi aplikasi; nilai disanitasi pada buku.
# Respons 429/403/400 memicu percobaan key berikutnya.
```

10.2. Retrieval Basis Pengetahuan dan Konteks Lokal

Fungsi load_winai_databases() mengambil sejumlah berkas JSON dan teks dari layanan basis data fikih Tarjih. Fungsi retrieve_winai_context() melakukan pencocokan kata kunci sederhana terhadap judul dan isi, memberi skor berdasarkan jumlah kata kunci yang muncul, mengurutkan hasil, lalu menyusun konteks internal hingga batas panjang tertentu. Dengan demikian, implementasi V8.0 lebih tepat disebut retrieval berbasis keyword scoring daripada sistem embedding vektor penuh.

10.3. Smart Interceptor dan Integrasi Hasil Hitung

Fungsi proses_pertanyaan_winai() mencoba mengenali intent melalui kata kerja seperti hitung, kalkulasi, cek, cari, tampilkan, simulasi, dan buka. Intent kemudian dipetakan ke kategori seperti visibilitas hilal, kota pertama, okultasi, gerhana, pasang surut, fajar, animasi, dan simulasi 3D. Untuk permintaan tertentu, aplikasi terlebih dahulu menjalankan kalkulasi internal, kemudian menyertakan laporan hasil hitung ke prompt model agar AI menjelaskan kesimpulan tanpa menggantikan data numerik asli.

Catatan audit V8.0: beberapa string routing internal WinAI masih memuat nomor menu historis dari tahap pengembangan sebelumnya. Oleh karena itu, dokumentasi buku memakai nama fungsi/menu sebagai acuan ilmiah, sedangkan daftar nomor resmi V8.0 mengikuti 41 menu pada Bab 11. Pengujian regresi routing WinAI direkomendasikan pada pengembangan berikutnya.

10.4. Referensi Internet Pendukung

Jika pertanyaan tidak diawali tanda slash, WinAI mencoba mengumpulkan cuplikan referensi melalui permintaan web dan DuckDuckGo Search. Hasil yang diperoleh ditempatkan sebagai konteks pendukung di prompt. Karena data internet dapat berubah dan kualitas cuplikan tidak seragam, hasil ini harus diperlakukan sebagai bahan bantu, bukan sebagai pengganti verifikasi sumber primer.

10.5. Antarmuka Multibahasa Indonesia-Inggris-Arab

Basis V8.0 mewarisi pembaruan V7.7 berupa dukungan antarmuka Indonesia, Inggris, dan Arab. Source code memuat kamus terjemahan untuk menu, label input, status sistem, istilah astronomi, dan bagian informasi rilis. Terjemahan highlight rilis V8.0 juga disediakan secara lokal sehingga tidak bergantung pada layanan daring untuk bagian dokumentasi tersebut.

10.6. Keamanan Kredensial dan Reprodusibilitas

Source code yang menjadi dasar buku masih memuat kredensial API dan password lokal secara hardcoded. Praktik ini tidak direkomendasikan untuk distribusi akademik atau publik. Edisi buku ini meniadakan nilai rahasia dari lampiran source code dan merekomendasikan pemindahan kredensial ke environment variable, file konfigurasi yang tidak ikut dipublikasikan, atau secret store. Langkah ini tidak mengubah algoritma astronomi, tetapi meningkatkan keamanan dan reprodusibilitas pengembangan.

DAFTAR PUSTAKA BAB 10

Gao, Y., et al. (2023). Retrieval-augmented generation for large language models: A survey. arXiv:2312.10997.

Ji, Z., et al. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), 1-38.

Lewis, P., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. NeurIPS, 33, 9459-9474.

Krotov, V., & Silva, L. (2018). Legality and ethics of web scraping. AMCIS.

BAB 11: MANAJEMEN DATABASE DAN KOMPARASI 50 TAHUN

Pembaruan V8.0: HIJRI_DB Auto-Builder berada pada Menu 41. Menu 39 adalah Peta Langit Dunia berbasis web dan Menu 40 adalah peluncur aplikasi pendamping Laporan Astronomi. Daftar resmi 41 menu di bawah ini menggantikan penomoran historis yang tersebar pada edisi lama.

Dalam arsitektur perangkat lunak komputasi astronomi, perhitungan posisi benda langit secara real-time (on-the-fly computation) memakan daya komputasi yang sangat besar. Jika setiap kali pengguna membuka antarmuka kalender aplikasi harus menghitung ulang integrasi numerik orbit bulan sejak konjungsi hingga matahari terbenam di seluruh dunia, aplikasi akan mengalami latensi yang tidak dapat ditoleransi (Richards, 1998).

Untuk mengatasi leher botol (bottleneck) performa ini, KHGT Times V8.0 memperkenalkan paradigma hibrida: menggabungkan komputasi ephemeris real-time untuk observasi harian dengan sistem prapemrosesan (pre-computing) menjadi basis data statis untuk rendering kalender jangka panjang.

11.1. Engine Auto-Builder db_hijriah.json

Fitur HIJRI_DB Auto-Builder merupakan instrumen khusus administrator (developer) yang difungsikan sebagai generator siklus kalender. Alih-alih menulis tabel kalender secara manual, modul ini memerintahkan mesin ephemeris NASA untuk melakukan simulasi waktu berjalan (time-lapse simulation) selama puluhan hingga ratusan tahun secara otonom, mengekstrak data awal bulan, dan menyimpannya ke dalam struktur data JavaScript Object Notation (JSON) yang sangat ringan dan cepat diakses oleh memori antarmuka (GUI).

11.1.1. Ekstraksi Siklus Fase Bulan selama Ratusan Tahun

Pembuatan kalender berjangka panjang dihadapkan pada masalah deret waktu (time series). Perangkat lunak harus melompati ribuan fase lunasi (siklus sinodik bulan) tanpa kehilangan presisi fraksi detik pada setiap waktu ijtimak (konjungsi). KHGT Times V8.0 mengeksekusi proses ini melalui thread terpisah (_build_hijri_db_thread) agar antarmuka tidak membeku (freeze).

Algoritma ini menggunakan rumus konversi empiris untuk memperkirakan tahun Masehi berdasarkan tahun Hijriah yang sedang diiterasi, guna memuat bagian file .bsp ephemeris yang tepat:

```
# Estimasi tahun Masehi dari tahun Hijriah
approx_greg_curr = int(y_h * 0.970224 + 622.54)
self.auto_switch_ephemeris(approx_greg_curr)
```

Konstanta 0.970224 merepresentasikan rasio panjang tahun kamariah rata-rata (354,36 hari) terhadap tahun tropis matahari (365,24 hari), sedangkan 622.54 adalah konstanta kalibrasi epoch Hijriah ke Masehi.

Setelah tahun yang tepat dikunci, sistem tidak mencari fase bulan dari tanggal 1 Januari hingga 31 Desember secara linier, yang mana sangat tidak efisien. Sebaliknya, sistem menggunakan fungsi prediksi waktu ijtimak (get_approx_nm_tt) dan membatasi rentang pencarian Skyfield secara eksklusif pada jendela pm 5 hari dari estimasi tersebut:

```
approx_tt_curr = self.get_approx_nm_tt(y_h, m_h)
nm_list_curr = self.get_new_moons_in_range(approx_tt_curr - 5.0, approx_tt_curr
+ 5.0)
```

Pendekatan narrow-band search ini memangkas beban komputasi evaluasi polinomial Chebyshev secara drastis, memungkinkan pembangunan database kalender 100 tahun diselesaikan hanya dalam hitungan detik hingga menit, sebuah capaian optimasi algoritma pencarian akar (root-finding optimization) tingkat tinggi (Doggett, 2012).

11.1.2. Logika Penentuan Jumlah Hari Berdasarkan Waktu Ijtimak

Sistem kalender Hijriah aritmatik (seperti kalender Urfi atau Fatimiyah) menentukan jumlah hari 29 atau 30 secara selang-seling menggunakan rumus pembagian modulo (Ilyas, 1994). Berbeda secara fundamental dengan metode tersebut, Kalender Hijriah Global Tunggal (KHGT) adalah kalender astronomis hakiki yang penentuan jumlah harinya murni didikte oleh visibilitas hilal di permukaan bumi sesudah terjadinya ijtimak.

Untuk mengotomatisasi penentuan 29 atau 30 hari dalam database JSON, KHGT Times tidak melakukan evaluasi visibilitas harian secara iteratif karena akan memakan waktu berhari-hari. Program ini secara matematis mengukur jarak absolut antar-awal bulan. Algoritmanya bekerja dengan langkah berikut:

1. Sistem menghitung parameter KHGT (PKG 1 dan PKG 2) untuk bulan saat ini (m_h) dan mendapatkan waktu absolut jatuhnya tanggal 1 dalam format Terrestrial Time (TT).
2. Sistem melakukan hal yang sama untuk bulan berikutnya ($m_h + 1$).
3. Jumlah hari dalam bulan saat ini murni merupakan hasil pengurangan (subtraction) antara

titik waktu absolut awal bulan depan dikurangi titik waktu absolut awal bulan saat ini.

```
start_tt_curr = self.calculate_khgt_1st_of_month(nm_curr)
start_tt_next = self.calculate_khgt_1st_of_month(nm_next)
# Hitung Jumlah Hari (Selisih waktu mulai bulan depan dan bulan ini)
jumlah_hari = int(round(start_tt_next - start_tt_curr))
```

Karena nilai `start_tt` yang dihasilkan oleh fungsi `calculate_khgt_1st_of_month` selalu berupa bilangan bulat yang merepresentasikan pukul 00:00 UTC, selisih matematis dari kedua titik ini akan selalu menghasilkan nilai diskrit 29 atau 30. Nilai inilah yang kemudian diinjeksi (append) ke dalam matriks dictionary `year_data` dan disimpan permanen sebagai file `db_hijriah.json`.

Inovasi arsitektur pra-komputasi (pre-computation architecture) ini menggeser beban berat astrometri ke sisi backend administrator, sehingga pengguna akhir (end-user) dapat menikmati modul Kalender Hijriah Berjalan dan Kalender Masehi Berjalan dengan interaktivitas yang mulus (seamless) dan instan.

11.2. Algoritma Komparasi Panjang (Big Data Analysis)

Setelah database kalender statis berhasil dibangun, tantangan selanjutnya dalam KHGT Times V8.0 adalah bagaimana melakukan komparasi analitik antara hasil Kalender Hijriah Global Tunggal (KHGT) dengan sistem regional yang berlaku di Indonesia (Neo MABIMS). Komparasi ini sangat vital untuk mendeteksi kapan akan terjadi "perbedaan" perayaan hari besar (Idul Fitri, Idul Adha, dan awal Ramadhan) di masa depan (Syamsuddin, 2021).

Menganalisis data selama 50 tahun ke depan (misalnya dari 1447 H hingga 1496 H) memerlukan eksekusi algoritma Big Data yang mengevaluasi ribuan parameter fisis dalam satu waktu.

11.2.1. Eksekusi Threading untuk Mencegah GUI Freeze

Operasi iteratif yang panjang (seperti loop dari tahun 1447 hingga 1496) memakan waktu eksekusi (execution time) yang signifikan karena setiap siklus tahun melibatkan perhitungan sunset, moonset, ketinggian, dan elongasi. Jika kalkulasi ini dijalankan di Main Thread (utas utama) yang sama dengan antarmuka grafis (GUI) Tkinter, aplikasi akan berhenti merespons input (freeze atau Not Responding) sampai seluruh proses loop selesai (Bassi, 2007).

KHGT Times V8.0 memecahkan masalah ini dengan mengimplementasikan modul threading bawaan Python. Ketika tombol pemrosesan diklik, GUI hanya bertugas memicu thread latar belakang (background thread):

```
def analisis_komparasi_ramadhan_50_tahun(self):
# Jalankan di thread terpisah agar GUI tidak Not Responding (Freeze)
threading.Thread(target=self._proses_komparasi_ramadhan,
daemon=True).start()
```

Kunci keberhasilan metode ini adalah penggunaan parameter `daemon=True`, yang memastikan thread kalkulasi akan otomatis dihentikan oleh sistem operasi jika jendela utama aplikasi ditutup oleh pengguna. Selanjutnya, untuk memperbarui tampilan teks di layar (yang mana Tkinter tidak mengizinkan pembaruan GUI dari thread selain Main Thread), KHGT Times menggunakan metode penjadwalan `.after(0, fungsi)`:

```
self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun
Ramadhan..."))
```

Pendekatan asynchronous UI update ini memastikan pengalaman pengguna (User Experience) tetap interaktif dan smooth meskipun mesin komputasi sedang bekerja maksimal memproses data puluhan tahun.

11.2.2. Komparasi Historis dan Masa Depan: KHGT vs MABIMS (Ramadhan, Syawal, Zulhijah)

Inti dari algoritma komparasi (modul terkait) adalah menyandingkan dua epistemologi hisab: KHGT (Global) yang bersifat geosentris, melawan Neo MABIMS (Regional) yang bersifat toposentris di ujung barat Indonesia (Kota Sabang, Aceh).

Untuk setiap tahun y dalam rentang 50 tahun, sistem melakukan langkah-langkah analitik berikut:

1. Membaca tanggal jatuh 1 Ramadhan/Syawal/Zulhijah dari HIJRI_DB (hasil prakomputasi KHGT).
2. Menghitung mundur satu hari untuk menetapkan "Hari Rukyat" (misal: 29 Syakban).
3. Mengevaluasi posisi hilal secara toposentrik di Sabang tepat pada waktu sunset di Hari

Rukyat tersebut.

```
# Evaluasi Neo MABIMS di Sabang (Alt >= 3° dan Elong >= 6.4°)
umur_bulan = waktu_sunset - ijtimak
if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
dt_mabims = dt_khgt
status = "Serentak (Sama)"
else:
# Jika gagal MABIMS, bulan sebelumnya istikmal (digenapkan 30 hari)
dt_mabims = dt_khgt + datetime.timedelta(days=1)
status = "Beda (MABIMS Mundur 1 Hari)"
```

Jika kriteria MABIMS terpenuhi di Sabang, maka jatuhnya tanggal 1 versi regional akan Serentak dengan KHGT global. Namun, jika MABIMS Sabang gagal terpenuhi (biasanya karena KHGT tervalidasi di Benua Amerika sementara elongasi di Indonesia belum mencapai 6.4 derajat), maka regional Indonesia akan memaksakan istikmal (penggenapan bulan menjadi 30 hari), sehingga jatuhnya tanggal 1 akan Mundur 1 Hari dibandingkan KHGT.

Laporan komparasi matriks 50 tahun ini menjadi instrumen riset yang tak ternilai bagi Muhammadiyah dan kementerian terkait untuk melakukan mitigasi sosial terhadap potensi perbedaan hari raya di masa depan, meredam polemik jauh sebelum ia terjadi (Odeh, 2016).

11.3. Ekspor Data Mentah (CSV dan PDF)

Perangkat lunak astronomi yang baik tidak hanya menampilkan hasil kalkulasinya di layar, tetapi juga memfasilitasi pengguna untuk menyimpan, mengekstrak, dan menganalisis data tersebut lebih lanjut pada perangkat lunak pengolah data eksternal (seperti Microsoft Excel, SPSS, atau MATLAB). Dalam KHGT Times V8.0, modul ekspor dirancang untuk fleksibilitas akademik

tingkat tinggi, mendukung format Comma Separated Values (CSV) untuk data tabular mentah, dan Portable Document Format (PDF) untuk pelaporan formal.

11.3.1. Parsing String Array menjadi Data Terstruktur (CSV)

Fitur ekspor CSV diaktifkan ketika pengguna berada pada modul yang menghasilkan data Big Data berbentuk tabel (seperti modul terkait). Secara khusus, pada modul "Tabel Ketinggian/Elongasi Hilal 50 Tahun", program menghasilkan data mentah yang sangat masif.

Tantangan teknisnya adalah bahwa pada modul-modul pelaporan teks, data awal berbentuk satu string panjang (multibaris) yang ditampilkan pada layar textbox. Untuk mengubahnya menjadi CSV yang valid, sistem harus melakukan proses parsing (penguraian teks) secara mandiri.

```
# Ekstrak data dari Textbox
report_data = self.textbox.get("1.0", "end-1c")
# Split berdasarkan baris
lines = report_data.split('\n')
for line in lines:
# Memfilter baris yang merupakan bagian dari tabel (mengandung karakter
'|')
# dan mengabaikan baris pembatas dekoratif ('=' atau '-')
if '|' in line and not line.startswith('=') and not line.startswith('-'):
# Memecah baris menjadi kolom-kolom
row = [col.strip() for col in line.split('|')]
writer.writerow(row)
```

Sistem membaca karakter palang vertikal (|) sebagai delimiter virtual dalam tampilan teks GUI, lalu memecahnya menjadi komponen larik (array list) Python. Untuk memastikan kompatibilitas internasional (terutama ketika dibuka di Microsoft Excel regional Eropa/Indonesia yang menggunakan titik koma sebagai pemisah kolom), fungsi csv.writer diinstruksikan menggunakan parameter `delimiter=';'`.

Sistem juga mengimplementasikan teknik `encoding='utf-8-sig'` saat menulis file. Parameter sig (BOM - Byte Order Mark) ini sangat krusial; ia memaksa Excel untuk merender karakter non-ASCII (seperti simbol derajat $^{\circ}$) secara sempurna, sehingga data ketinggian hilal (contoh: "5.45°") tidak berubah menjadi karakter acak (gibberish) saat dianalisis oleh pengguna (VanderPlas et al., 2020).

11.3.2. Pelukisan Ulang Canvas untuk Laporan Akademik (PDF)

Sementara CSV ditujukan untuk analisis kuantitatif lanjutan, PDF ditujukan untuk publikasi, pencetakan (printing), dan pelaporan yudisial. Terdapat dua metodologi ekspor PDF dalam KHGT Times V8.0, bergantung pada modul mana yang sedang aktif.

1. Ekspor Laporan Teks Menggunakan Pustaka FPDF: Jika pengguna sedang membuka modul hisab waktu salat atau visibilitas harian (yang berformat teks standar), program mengonversi log teks menjadi dokumen PDF yang terstruktur dengan header dan waktu cetak:

```
pdf = FPDF()
pdf.add_page()
pdf.set_font("Courier", size=9) # Menggunakan font Monospace agar tabel rata
# ...
clean_line = line.replace('°', ' deg').replace('→', ' ->').replace('►',
'>').replace('☾', '[Bulan]')
safe_line = clean_line.encode('latin-1', 'replace').decode('latin-1')
pdf.cell(0, 4, txt=safe_line, ln=True)
```

Proses sanitization (seperti `replace("°", " deg")`) dilakukan sebagai bentuk penanganan galat (error handling) mengingat library FPDF versi klasik memiliki keterbatasan dalam memproses set karakter Unicode secara native.

2. Ekspor Peta Visual dan Kalender Murni (Konversi Langsung Image): Inovasi visual terbaik terjadi jika pengguna menekan tombol PDF pada modul Kalender Hijriah atau modul Kalender Masehi. Karena GUI kalender dibentuk menggunakan widget Tkinter, tidak ada "teks" yang bisa diubah menjadi PDF secara langsung. Mengeksport screenshot antarmuka GUI akan menghasilkan resolusi yang rendah (buram saat dicetak).

Untuk mensiasatinya, KHGT Times memiliki mesin pelukis (rendering engine) internal tak terlihat (headless) yang dijalankan menggunakan pustaka manipulasi citra Pillow (PIL):

```
# Memanggil fungsi render gambar HD internal (bukan screenshot GUI)
img = self._generate_calendar_image()
# Menyimpan secara langsung ke format PDF dengan resolusi cetak
img.save(filepath, "PDF", resolution=150.0)
```

Fungsi `_generate_calendar_image()` secara otonom menggambar ulang seluruh kotak kalender, menarik fon TrueType resolusi tinggi (arialbd.ttf), memuat fase bulan, dan menyematkan warna indikator peringatan hari besar ke dalam satu kanvas raksasa beresolusi 1400x1000 piksel. Konversi langsung matriks piksel ini ke dalam PDF dengan opsi `resolution=150.0` menjamin hasil export yang sangat tajam (crisp), setara dengan publikasi desain grafis profesional, dan siap dikirimkan sebagai poster atau edaran resmi organisasi.

Rangkaian sistem manajemen basis data dan mekanisme ekspor data ganda inilah yang menyempurnakan KHGT Times V8.0, mentransformasikannya dari sebuah skrip saintifik laboratorium menjadi instrumen perangkat lunak operasional (production-ready) untuk umat Islam di seluruh dunia.

DAFTAR PUSTAKA BAB 11

Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10-20.

Doggett, L. E. (2012). Calendars. In *Explanatory Supplement to the Astronomical Almanac* (pp. 575-608). University Science Books.

Ilyas, M. (1994). *Astronomy of Islamic Times for the Twenty-first Century*. Mansell.

Richards, E. G. (1998). *Mapping Time: The Calendar and Its History*. Oxford University Press.

Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.

VanderPlas, J., et al. (2020). Python for data analysis. *Journal of Data Science*, 14(2), 34-45.

11.4. Arsitektur 41 Menu KHGT Times V8.0

Daftar berikut diekstrak dari struktur `detailed_features` pada informasi rilis V8.0. Ringkasan fungsi diambil dari bagian "Kegunaan" masing-masing menu dan dipadatkan agar dapat dibaca sebagai matriks arsitektur.

No.	Menu V8.0	Fungsi ringkas
1	Kalender Hijriah Berjalan	Kalender visual interaktif yang terintegrasi dengan database KHGT.
2	Kalender Masehi Berjalan	Kalender visual penanggalan sipil internasional. Menampilkan matriks kalender Gregorian dengan penyematan tanggal KHGT sebagai pelengkap. Menghitung rotasi fase bulan visual untuk setiap harinya.
3	Konversi Kalender	Transformasi tanggal antara Masehi (Gregorian/Julian) dan Hijriah (KHGT).
4	Visibility Hilal (KHGT)	Analisis komprehensif visibilitas hilal berdasarkan multi-kriteria global.
5	Analisis Hilal Global	Pelacakan visibilitas hilal di seluruh zona waktu dunia secara kronologis.
6	Status Kriteria Batas	Simulator Anatomi Kriteria Imkanur Rukyat (KHGT). Instrumen edukasi

		untuk membedah parameter fisis kriteria KHGT (5-8). Pengguna dapat melihat bagaimana posisi bulan berinteraksi dengan zona kritis seperti Batas Danjon dan Bias Cahaya Senja.
7	Kota Pertama KHGT (Mainland)	Melacak titik daratan utama (Mainland) pertama di dunia yang masuk bulan baru.
8	Crescent Visibility Map	Pemindaian peta visibilitas hilal global dalam resolusi tinggi (HD).
9	Altitude Chart Analyser	Visualisasi kurva ketinggian harian benda langit. Modul ini merelasikan pergerakan semu harian Matahari dan Bulan dalam sebuah grafik sinusoida. Berguna untuk melihat titik puncak (transit), waktu terbit-terbenam, serta gap ketinggian antara Matahari dan Bulan sepanjang 24 jam penuh di lokasi tertentu.
10	Komparasi 50 Tahun (Ramadhan)	Studi perbandingan penentuan awal bulan antara KHGT dan kriteria lokal (MABIMS).
11	Tabel Tinggi Hilal 50 Thn (Ramadhan)	Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Ramadhan selama 50 tahun.
12	Tabel Elongasi Hilal 50 Thn (Ramadhan)	Ekstraksi data mentah Sudut Elongasi untuk 1 Ramadhan selama 50 tahun.
13	Komparasi 50 Tahun (Syawal)	Analisis potensi beda Hari Raya Idul Fitri umat Islam masa depan.
14	Tabel Tinggi Hilal 50 Thn (Syawal)	Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Syawal selama 50 tahun.
15	Tabel Elongasi Hilal 50 Thn (Syawal)	Ekstraksi data mentah Sudut Elongasi untuk 1 Syawal selama 50 tahun.
16	Komparasi 50 Tahun (Zulhijah)	Analisis potensi beda Hari Raya Idul Adha umat Islam masa depan.
17	Tabel Tinggi Hilal 50 Thn (Zulhijah)	Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Zulhijah selama 50 tahun.
18	Tabel Elongasi Hilal 50 Thn (Zulhijah)	Ekstraksi data mentah Sudut Elongasi untuk 1 Zulhijah selama 50 tahun.
19	Komparasi 50 Tahun (12 Bulan)	Analisis probabilitas sinkronisasi global KHGT vs MABIMS.
20	Fase Bulan (Moonphase)	Prediktor siklus fase bulan bulanan dan tahunan. Menghitung fraksi iluminasi dan sudut fase untuk menentukan bentuk bulan yang tampak dari bumi, mulai dari New Moon, First Quarter, Full Moon, hingga Last Quarter.
21	Moon Times	Jadwal terbit, transit, dan terbenam Bulan selama satu bulan penuh.
22	Sun Times	Jadwal harian Matahari (Sunrise, Transit, Sunset). Menyediakan data presisi mengenai waktu fajar, tengah hari, dan terbenamnya matahari. Modul ini menjadi dasar perhitungan waktu-

		waktu fenomena siang hari.
23	Sun Moon Ephemeris	Ekstraksi data koordinat langit (RA/Dec) secara detail.
24	Sistem Sun Moon Earth	Visualisasi 3D sistem tata surya geosentrik sederhana. Menampilkan model 3D interaktif yang menunjukkan posisi Bulan dan Matahari dalam mengelilingi Bumi sebagai pusat pengamatan (Geosentrik). Termasuk proyeksi kerucut bayangan bumi (Umbra).
25	Prayer Times	Jadwal salat 5 waktu dan fenomena senja (Twilight). Modul ini menghitung waktu Subuh, Terbit, Duha, Zuhur, Asar, Magrib, dan Isya. Mendukung berbagai standar otoritas internasional dan metode khusus untuk wilayah lintang tinggi.
26	Qiblah Direction & Times	Kalkulator arah kiblat dan peta navigasi Ka'bah. Selain menghitung sudut derajat arah kiblat dari arah Utara-Barat menggunakan model Bumi Elipsoid WGS84, modul ini menyediakan visualisasi garis kiblat pada peta dunia untuk memastikan jalur terpendek (Great Circle) menuju Makkah.
27	Qibla Time (Rashdul Lokal)	Menentukan waktu saat matahari berada tepat di atas arah kiblat suatu lokasi.
28	Kalkulator Mizwala (Bayangan)	Simulator instrumen tongkat istiwa' atau Mizwala. Menghitung panjang dan arah bayangan benda tegak sepanjang hari. Sangat berguna untuk penentuan arah kiblat manual dan praktikum falak di lapangan.
29	Evaluasi Fajar Shadiq & SQM	Analisis fajar fisis dan kegelapan langit (Sky Quality).
30	Analisis Gerhana	Pemindaian otomatis fenomena Gerhana Matahari dan Gerhana Bulan.
31	Equinox & Solstice	Menghitung waktu pergantian musim astronomis. Mendeteksi waktu presisi saat matahari melintasi ekuator langit (Equinox) dan saat mencapai deklinasi maksimum/minimum (Solstice).
32	Planetary Times	Jadwal visibilitas 7 planet utama (Merkurius s/d Neptunus).
33	Generator Analemma Matahari	Pemetaan lintasan analemma Matahari dalam satu tahun. Memvisualisasikan lintasan 'angka 8' yang dibentuk oleh posisi matahari di jam yang sama sepanjang tahun akibat kemiringan sumbu bumi dan orbit elips.
34	Prediktor Pasang Surut Gravitasi	Memprediksi fenomena pasang surut air laut akibat gaya gravitasi Bulan dan Matahari.
35	Astrofotografi & Kontras Visibilitas	Analisis teknis untuk pengambilan foto hilal (imaging).

36	Live Animasi	Simulator posisi benda langit secara real-time dan terintegrasi observatorium.
37	Simulasi Ephemeris 3D	Simulator pergerakan benda langit 3D menggunakan mesin Matplotlib.
38	WinAI: Aplikasi AI Windows	Asisten cerdas berbasis kecerdasan buatan (AI) untuk konsultasi astronomi dan fikih.
39	Peta Langit Dunia	Membuka Peta Langit Dunia KHGT Times berbasis web untuk eksplorasi langit secara online.
40	Laporan Astronomi	Membuka aplikasi pendamping Laporan Astronomi dari KHGT Times.
41	HIJRI_DB Auto-Builder (Admin)	Fitur khusus Developer untuk merakit fondasi Kalender Aplikasi (Siklus Pembuatan Data JSON).

11.5. Integrasi Menu 39 dan Menu 40

Menu 39, Peta Langit Dunia, tidak melakukan kalkulasi desktop baru saat dipilih; fungsi membuka layanan <https://kasmui.cloud/langit/> melalui browser default. Menu 40, Laporan Astronomi, mencari file LaporanAstronomi.exe atau LaporanAstronomi.exe pada folder aplikasi dan menjalankannya sebagai aplikasi pendamping. Jika berkas tidak ditemukan atau gagal dieksekusi, aplikasi menampilkan pesan kesalahan.

BAB 12: PENUTUP DAN PROYEKSI MASA DEPAN FALAK DIGITAL

12.1. Sintesis Arsitektur Komputasi KHGT Times V8.0

KHGT Times V8.0 merupakan evolusi dari mesin komputasi falak menuju platform desktop modular yang menggabungkan kalender, analisis hilal, pemetaan, ephemeris, waktu salat, kiblat, fenomena astronomi, visualisasi 3D, Live Animasi, kecerdasan buatan, dan integrasi layanan eksternal. Berdasarkan source code yang dianalisis, arsitektur V8.0 dapat diringkas dalam lima pilar berikut.

1. Mesin ephemeris dua-kernel: de440s.bsp untuk 1850-2149 dan de441-new.bsp di luar rentang tersebut sampai batas aplikasi 1-2999 M, dengan pemilihan otomatis dan cache kernel.
2. Komputasi hibrida: Skyfield digunakan untuk astrometri berbasis ephemeris JPL, sementara PyEphem dan NumPy mendukung kebutuhan iterasi, event, serta pemrosesan numerik pada berbagai modul.
3. Visualisasi dan observasi simulatif: Matplotlib, Pillow, kanvas Tk/CustomTkinter, simulasi 3D, analemma, peta visibilitas, Live Animasi, Tracker, rasi bintang, planet, dan komparasi Stellarium.
4. Antarmuka informasi: 41 menu yang tersusun menurut domain tugas, dukungan bahasa Indonesia-Inggris-Arab, ekspor data/gambar, serta akses ke Peta Langit Dunia dan aplikasi pendamping Laporan Astronomi.
5. Lapisan AI dan basis pengetahuan: WinAI menggabungkan retrieval berbasis kata kunci, data kalender, hasil kalkulasi aplikasi, referensi internet pendukung, dan model generatif untuk membantu interpretasi.

12.2. Implikasi untuk Dokumentasi dan Validasi Akademik

Edisi V8.0 memperjelas bahwa dokumentasi perangkat lunak astronomi harus diperlakukan sebagai bagian dari proses validasi. Nama menu, rentang ephemeris, fungsi pemuatan kernel, dan jalur integrasi eksternal harus sama antara buku dan source code. Dengan demikian, pembaca dapat menelusuri klaim teknis buku ke struktur program yang benar-benar digunakan.

12.3. Keterbatasan Implementasi yang Perlu Dicatat

- Sebagian fungsi bergantung pada koneksi internet, terutama WinAI, basis data daring, Peta Langit Dunia, dan layanan pelacakan tertentu.
- Sejumlah komponen bersifat Windows-spesifik, misalnya win10toast, winsound, dan aplikasi pendamping berformat .exe.
- Source code masih memuat kredensial dan password lokal secara hardcoded; pada publikasi ini nilainya disanitasi.
- Beberapa string routing WinAI masih memakai nomor menu historis, sehingga pengujian regresi terhadap pemetaan 41 menu V8.0 diperlukan.
- Keberhasilan kalkulasi rentang tertentu bergantung pada tersedianya berkas de440s.bsp dan de441-new.bsp yang benar pada folder aplikasi.

12.4. Arah Pengembangan Masa Depan

- Memisahkan kredensial dari source code dan menerapkan konfigurasi keamanan yang aman untuk distribusi publik.

- Menambahkan pengujian otomatis untuk pemetaan menu, auto-switch ephemeris pada batas 1849/1850 dan 2149/2150, serta regresi hasil astrometri.
- Menyediakan metadata versi kernel dan checksum berkas .bsp agar reproduksibilitas hasil lebih kuat.
- Memperluas pengujian multibahasa, khususnya tata letak kanan-ke-kiri untuk bahasa Arab.
- Mendokumentasikan benchmark hasil terhadap referensi astronomi eksternal pada skenario uji yang didefinisikan secara eksplisit.

Penutup

Dengan pembaruan ini, buku tidak lagi mendokumentasikan KHGT Times sebagai V7.4 dengan satu kernel ephemeris, melainkan sebagai KHGT Times V8.0 dengan 41 menu, pemilihan kernel JPL otomatis, antarmuka multibahasa, visualisasi yang diperluas, dan integrasi layanan desktop-web. Pemisahan antara teori astronomi, implementasi perangkat lunak, dan keterbatasan sistem diharapkan membuat buku lebih layak dipakai sebagai rujukan akademik dan dokumentasi pengembangan.

DAFTAR PUSTAKA BAB 12

Park, R. S., Folkner, W. M., Williams, J. G., & Boggs, D. H. (2021). The JPL planetary and lunar ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3), 105.

Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.

Virtanen, P., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.

DAFTAR PUSTAKA

- Abdali, S. K. (1997). The correct qibla. American Trust Publications.
- Anwar, S. (2008). Hari Raya dan Problematika Penentuan Awal Bulan Hijriah. Suara Muhammadiyah.
- Bassi, A. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3), 10-20.
- Bretagnon, P., & Francou, G. (1988). Planetary theories in rectangular and spherical variables. VSOP 87 solutions. *Astronomy and Astrophysics*, 202, 309-315.
- Caldwell, J. A. R., & Laney, C. D. (2001). First visibility of the lunar crescent. *African Skies/Cieux Africains*, 5, 15-23.
- Doggett, L. E. (2012). Calendars. In *Explanatory Supplement to the Astronomical Almanac* (pp. 575-608). University Science Books.
- Downey, A. (2011). Think Python: How to think like a computer scientist. O'Reilly Media.
- Duffett-Smith, P., & Zwart, J. (2011). Practical Astronomy with your Calculator or Spreadsheet (4th ed.). Cambridge University Press.
- Espenak, F., & Meeus, J. (2006). Five Millennium Canon of Solar Eclipses: -1999 to +3000. NASA Goddard Space Flight Center.
- Fluke, C. J., Bourke, P. D., & O'Donovan, D. (2006). Future directions in astronomy visualization. *Publications of the Astronomical Society of Australia*, 23(1), 12-24.
- Franke, R. (1982). Scattered data interpolation: tests of some methods. *Mathematics of Computation*, 38(157), 181-200.
- Garg, S. K., & Buyya, R. (2013). NetworkCloudSim: Modelling parallel applications in cloud simulations. *Fourth International Conference on Utility and Cloud Computing*, 105- 113.
- Gargano, M., et al. (2015). Planetariums and digital domes: A new era for astronomical visualization. *Journal of Science Communication*, 14(2).
- Guessoum, N. (2008). The astrophysics of the Islamic calendar. *Astronomy & Geophysics*, 49(5), 5.16-5.21.
- Haseeb, A. (2020). The Global Islamic Calendar: Prospects and Challenges. Islamic Foundation.
- Hoffman, R. E. (2003). Rationalizing the Islamic calendar. *Science and Islamic Observatory*, 1(1), 1-12.
- Hughes, D. W., Yallop, B. D., & Hohenkerk, C. Y. (1989). The equation of time. *Monthly Notices of the Royal Astronomical Society*, 238(4), 1529-1535.
- Hughes, J. F., et al. (2014). Computer Graphics: Principles and Practice (3rd ed.). Addison-Wesley Professional.
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.
- Ilyas, M. (1994). Astronomy of Islamic Times for the Twenty-first Century. Mansell.
- Ilyas, M. (1997). Islamic Calendar, Times & Qibla. Berita Publishing.
- Kaplan, G. H. (2005). The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models (USNO Circular No. 179). United States Naval Observatory.
- King, D. A. (1986). Astronomy in the Service of Islam. Variorum Reprints.
- Lancaster, P., & Salkauskas, K. (1986). Curve and Surface Fitting: An Introduction. Academic Press.
- Meeus, J. (1998). Astronomical Algorithms (2nd ed.). Willmann-Bell.

- Moosavi, S. Z., Bakhshi, H., & Zadehkorram, A. R. (2015). A survey on calculation methods of prayer times at high latitudes. *International Journal of Mechatronics, Electrical and Computer Technology*, 5(15), 2095-2105.
- Morrison, L. V., & Stephenson, F. R. (2001). Historical values of the Earth's clock error Delta T and the calculation of eclipses. *Journal for the History of Astronomy*, 32(3), 227- 236.
- Newman, M. E. J. (2012). *Computational Physics*. CreateSpace Independent Publishing Platform.
- Odeh, M. S. (2004). New criterion for lunar crescent visibility. *Experimental Astronomy*, 18(1-3), 39-64.
- Odeh, M. S. (2016). *The Global Hijri Calendar: The Istanbul 2016 Congress Resolutions*. International Astronomical Center (IAC).
- Park, R. S., Folkner, W. M., Williams, J. G., & Boggs, D. H. (2021). The JPL planetary and lunar ephemerides DE440 and DE441. *The Astronomical Journal*, 161(3), 105.
- Patat, F. (2003). UBVRI night sky brightness during sunspot maximum at ESO-Paranal. *Astronomy & Astrophysics*, 400(3), 1183-1198.
- Rhodes, B. (2019). *Skyfield: High precision research-grade positions for planets and Earth satellites*. Astrophysics Source Code Library, ascl-1907.
- Richards, E. G. (1998). *Mapping Time: The Calendar and Its History*. Oxford University Press.
- Schaefer, B. E. (1993). Astronomy and the limits of vision. *Vistas in Astronomy*, 36, 311- 361.
- Susiknan, A. (2007). *Ilmu Falak: Perjumpaan Khazanah Islam dan Sains Modern*. Suara Muhammadiyah.
- Syamsuddin, S. (2021). *Kalender Hijriah Global Tunggal: Epistemologi dan Implementasi*. Majelis Tarjih dan Tajdid PP Muhammadiyah.
- Torge, W., & Müller, J. (2012). *Geodesy* (4th ed.). De Gruyter.
- Urban, S. E., & Seidelmann, P. K. (Eds.). (2013). *Explanatory Supplement to the Astronomical Almanac* (3rd ed.). University Science Books.
- VanderPlas, J., et al. (2020). Python for data analysis. *Journal of Data Science*, 14(2), 34- 45.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261-272.
- Wang, J., et al. (2021). Over-the-air updates for machine learning models. *IEEE Internet of Things Journal*, 8(15).
- White, J., et al. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. arXiv preprint arXiv:2302.11382.
- Yallop, B. D. (1998). *A Method for Predicting the First Sighting of the New Crescent Moon* (NAO Technical Note No. 69). HM Nautical Almanac Office.
- Zayed, A. (2015). *Fikih Astronomi: Menjawab Problematika Penentuan Awal Bulan*. Darul Ulum.
- Zheng, L., Chiang, W. L., Ying, J., Shen, S., Huang, C., ... & Xing, E. P. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. arXiv preprint arXiv:2306.05685.

GLOSARIUM

A

- **Analemma:** Kurva berbentuk angka 8 yang menunjukkan posisi Matahari di langit jika diamati pada jam yang sama setiap hari sepanjang tahun.
- **API (Application Programming Interface):** Antarmuka yang memungkinkan dua aplikasi perangkat lunak untuk saling berkomunikasi, digunakan dalam sistem WinAI untuk mengakses LLM.
- **Asensio Rekta (Right Ascension):** Koordinat ekuatorial astronomis yang analog dengan bujur geografis di bumi, diukur ke arah timur dari titik Aries.
- **Azimuth:** Sudut putar arah horizontal suatu benda langit, diukur searah jarum jam dari titik Utara sejati (0°).

B

- **Bicubic Interpolation:** Algoritma matematika dalam SciPy untuk merekonstruksi dan memperhalus matriks data kasar menjadi kurva spasial yang mulus tanpa garis bergerigi (digunakan pada peta visibilitas).
- **Boundary Norm:** Teknik rendering visual pada Matplotlib yang mengubah gradasi warna menjadi blok warna diskrit dan tegas untuk area peta tertentu.

C

- **Celestial Sphere (Bola Langit):** Bola imajiner dengan jari-jari tak terhingga yang berpusat pada bumi, di mana benda-benda langit seolah-olah menempel padanya.
- **CSV (Comma Separated Values):** Format berkas teks sederhana untuk menyimpan data tabular, sering digunakan untuk ekspor data historis KHGT ke spreadsheet.

D

- **Deklinasi:** Koordinat ekuatorial yang analog dengan garis lintang di bumi, mengukur jarak sudut suatu benda langit di utara atau selatan ekuator langit.
- **Delaunay Triangulation:** Metode matematis untuk menghubungkan titik-titik data spasial menjadi jaringan segitiga, digunakan pada interpolasi peta SciPy.

E

- **Elongasi:** Jarak sudut antara pusat piringan Bulan dan pusat piringan Matahari; dalam KHGT disyaratkan minimal 8 derajat (secara geosentris).
- **Ephemeris:** Tabel atau basis data yang memuat posisi benda-benda astronomis di langit pada waktu tertentu secara berurutan.
- **Equation of Time (EoT):** Selisih waktu antara waktu matahari sejati (berdasarkan pengamatan) dengan waktu matahari rata-rata (berdasarkan jam mekanik).

F

- **Fajar Shadiq (Astronomical Twilight):** Waktu ketika cahaya matahari mulai tersebar di atmosfer saat pusat piringan matahari berada pada sudut depresi tertentu (umumnya -18° atau -20°) di bawah ufuk.

G

- **Geosentris:** Sistem koordinat dan titik pengamatan astronomi yang diasumsikan berpusat tepat di inti Bumi.
- **GUI (Graphical User Interface):** Antarmuka pengguna grafis, seperti tombol dan menu visual yang dibangun menggunakan pustaka Tkinter pada Python.

H

- **Hilal:** Bulan sabit paling tipis yang pertama kali dapat diamati sesaat setelah matahari terbenam pada hari terjadinya konjungsi (ijtimak).
- **Hisab:** Perhitungan matematis dan astronomis untuk menentukan posisi benda langit, waktu salat, dan awal bulan Hijriah.

I

- Ijtimak (Konjungsi / New Moon): Peristiwa ketika Bulan dan Matahari berada pada bujur ekliptika yang sama, menandakan akhir siklus lunasi lama dan awal siklus lunasi baru.
- Istikmal: Penggenapan jumlah hari dalam bulan kalender Hijriah menjadi 30 hari apabila hilal tidak terlihat atau kriteria visibilitas tidak terpenuhi.

J

- JSON (JavaScript Object Notation): Format teks ringan untuk pertukaran data, digunakan oleh KHGT Times untuk menyimpan database kalender prakomputasi.
- JPL (Jet Propulsion Laboratory): Pusat penelitian NASA yang menerbitkan data Development Ephemeris presisi tinggi (seperti DE441) yang digunakan dalam aplikasi ini.

K

- KHGT (Kalender Hijriah Global Tunggal): Sistem penanggalan Islam yang bertujuan menyatukan jatuhnya awal bulan di seluruh dunia menggunakan prinsip hisab dan transfer visibilitas global.

L

- LLM (Large Language Model): Model kecerdasan buatan berbasis jaringan saraf dalam yang memproses bahasa alami, digunakan dalam KHGT Times sebagai Asisten Fikih Falak.
- Lunasi: Satu siklus penuh fase bulan, dari satu ijtimak ke ijtimak berikutnya (rata-rata 29,53 hari).

M

- MABIMS: Kriteria visibilitas hilal regional yang disepakati oleh Menteri Agama Brunei Darussalam, Indonesia, Malaysia, dan Singapura.
- Mainland (Daratan Utama): Wilayah daratan benua yang memiliki populasi berkesinambungan, menjadi syarat lokasi terpenuhinya kriteria KHGT (mengabaikan kepulauan terisolasi).
- Matlak: Batas geografis visibilitas; dalam konsep KHGT, matlak bersifat global (satu bumi satu matlak / Ittihad al-Matali').

N

- Nisf al-Lail: Pendekatan fikih untuk wilayah lintang tinggi yang membagi malam menjadi dua bagian sama panjang untuk menentukan masuknya waktu Isya dan Subuh.
- NumPy: Pustaka komputasi saintifik Python yang digunakan untuk memproses array numerik besar dan aljabar linear.

O

- Oblate Spheroid: Bentuk bumi yang sebenarnya, yaitu elipsoid yang agak pepat di bagian kutub dan menggembung di bagian ekuator akibat gaya sentrifugal rotasi bumi.
- Oseania (Oceania): Gugusan kepulauan di Samudra Pasifik (Timur Jauh) yang dikecualikan oleh KHGT Times dari pelacakan visibilitas karena bukan wilayah mainland.

P

- Paralaks (Parallax): Pergeseran posisi semu suatu benda langit jika dilihat dari dua titik pengamatan yang berbeda (misalnya dari inti bumi vs permukaan bumi).
- PyEphem: Pustaka astronomi klasik berbasis bahasa C yang digunakan KHGT Times sebagai mesin brute-force untuk memindai titik sunset dengan cepat.

Q

- Qibla Time (Rashdul Kiblat): Momen spesifik ketika lintasan azimuth matahari (atau bayangannya) sejajar dan berhimpitan secara presisi dengan arah azimuth Ka'bah.

R

- Refraksi Atmosfer: Peristiwa pembiasan cahaya benda langit saat melewati atmosfer bumi, menyebabkan matahari/bulan tampak lebih tinggi dari posisi geometris sebenarnya.
- Round-Robin: Algoritma penjadwalan rotasi yang digunakan dalam KHGT Times untuk merotasi kunci API (API Key) secara bergantian guna menghindari pemblokiran rate-limit.

S

- Skyfield: Pustaka Python modern yang menghitung posisi planet dan satelit menggunakan prinsip vektor mutakhir setara dengan keakuratan observatorium standar IAU.
- Syzygy: Konfigurasi garis lurus antara tiga benda langit (seperti Matahari, Bumi, dan Bulan) yang merupakan syarat mutlak terjadinya gerhana.

T

- Toposentris: Sistem koordinat yang titik referensinya murni bertumpu pada mata pengamat yang berdiri tegak di suatu titik kordinat tertentu di permukaan Bumi.
- Terrestrial Time (TT): Skala waktu dinamis tanpa gangguan yang digunakan dalam komputasi ephemeris, bebas dari anomali pelambatan rotasi bumi (Delta T).

U

- Umbra: Area bayangan inti bumi yang paling pekat; apabila bulan memasuki area ini secara geometris, maka terjadilah gerhana bulan (sebagian atau total).
- UTC (Coordinated Universal Time): Standar waktu sipil global yang menggantikan Greenwich Mean Time (GMT), menjadi dasar waktu dalam pemrograman komputasi antar- zona.

V

- Visibilitas: Parameter atau kriteria astrofisika yang memprediksi apakah hilal secara teoretis dapat dilihat (dirukyat) dengan mata telanjang atau instrumen optik dari permukaan bumi.

W

- WGS84 (World Geodetic System 1984): Standar datum referensi geodetik global yang memodelkan bumi sebagai elipsoid, digunakan KHGT Times untuk memetakan koordinat observasi.
- WinAI: Nama modul Kecerdasan Buatan (AI) berbasis asisten pintar yang disematkan langsung dalam desktop client KHGT Times V8.0.

X

- XEphem: Perangkat lunak kalender astronomi ephemeris historis berkinerja tinggi, yang basis kode C-nya menjadi tulang punggung bagi pustaka PyEphem di Python.

Y

- Y-Axis (Sumbu Ordinat): Sumbu kartesian yang digunakan dalam matematika simulasi 3D KHGT Times untuk merender matriks rotasi proyeksi ke arah vertikal (zenith).

Z

- Z-Buffer (Painter's Algorithm): Algoritma manipulasi kedalaman gambar pada kanvas 3D yang mengatur agar benda astronomi yang dekat selalu digambar menutupi benda yang jauh.
- Zawal: Waktu ketika matahari tepat melintasi garis meridian pengamat (kulminasi atas), menandai masuknya waktu awal salat Dzuhur.
- **Auto-Switch Ephemeris:** Mekanisme V8.0 yang memilih de440s.bsp atau de441-new.bsp berdasarkan tahun target.
- **de440s.bsp:** Kernel ephemeris JPL ringkas yang digunakan KHGT Times V8.0 untuk tahun penuh 1850-2149.

- **de441-new.bsp:** Kernel ephemeris jangka panjang yang digunakan V8.0 untuk tahun 1-1849 dan 2150-2999.
- **Peta Langit Dunia:** Layanan peta langit berbasis web yang dibuka dari Menu 39.
- **Laporan Astronomi:** Aplikasi pendamping Windows yang dijalankan dari Menu 40 bila file executable tersedia.
- **Stellarium Synchronization:** Fitur pada Live Animasi untuk membandingkan data Matahari-Bulan dengan output Stellarium.
- **Multibahasa:** Dukungan antarmuka Indonesia, Inggris, dan Arab yang tersedia pada basis V8.0.

LAMPIRAN 1: PARAMETER, KOORDINAT REFERENSI, DAN KONFIGURASI V8.0

A. Matriks Parameter Visibilitas

Sistem	Sifat	Ketinggian	Elongasi	Referensi	Basis
KHGT (PKG 1)	Global	$h \geq 5^\circ$	$\varepsilon \geq 8^\circ$	Daratan utama	Geosentris
KHGT (PKG 2)	Global	$h \geq 5^\circ$	$\varepsilon \geq 8^\circ$	Benua Amerika / uji fajar Gisborne	Geosentris
Neo MABIMS	Regional	$h \geq 3^\circ$	$\varepsilon \geq 6,4^\circ$	Asia Tenggara	Toposentris
Wujudul Hilal	Regional/Lokal	$h > 0^\circ$	Bebas	Lokasi rujukan	Toposentris

B. Koordinat Referensi

Lokasi	Wilayah	Lintang	Bujur	Fungsi
Makkah (Ka'bah)	Arab Saudi	21.4225° LU	39.8262° BT	Referensi arah kiblat
Greenwich	Inggris Raya	51.4769° LU	0.0005° BB	Acuan UTC
Sabang	Indonesia	5.8943° LU	95.3184° BT	Referensi komparasi MABIMS
Gisborne	Selandia Baru	38.6623° LS	178.0176° BT	Uji fajar PKG 2
Titik Nol IDL	Samudra Pasifik	0.0000°	$\pm 180.0000^\circ$	Rekonstruksi grid garis tanggal

C. Pemilihan Ephemeris V8.0

Tahun target	Berkas	Peran	Catatan
1850-2149	de440s.bsp	Kernel pendek/modern	Batas tahun penuh yang aman pada logika V8.0
1-1849	de441-new.bsp	Kernel panjang	Menjaga jangkauan masa awal aplikasi
2150-2999	de441-new.bsp	Kernel panjang	Menjaga jangkauan masa depan aplikasi

LAMPIRAN 2: MANAJEMEN BERKAS BSP DAN STRATEGI DUA-KERNEL

Edisi lama menyertakan aplikasi pemotong dan penggabung berkas BSP. Pada V8.0, kebutuhan operasional utama justru sederhana: dua berkas ephemeris harus tersedia di direktori aplikasi dengan nama yang tepat, yaitu `de440s.bsp` dan `de441-new.bsp`. Aplikasi menentukan kernel berdasarkan tahun target, memeriksa keberadaan berkas, memuatnya dengan Loader Skyfield, lalu menyimpan objek kernel di cache internal.

```
EPHEMERIS_SHORT_FILE = "de440s.bsp"
EPHEMERIS_LONG_FILE = "de441-new.bsp"
EPHEMERIS_SHORT_MIN_YEAR = 1850
EPHEMERIS_SHORT_MAX_YEAR = 2149
EPHEMERIS_LONG_MIN_YEAR = 1
EPHEMERIS_LONG_MAX_YEAR = 2999
```

A. Pemeriksaan yang Disarankan

- Pastikan kedua berkas `.bsp` berada pada folder yang sama dengan aplikasi atau direktori `BASE_DIR` yang digunakan.
- Verifikasi checksum berkas setelah unduhan atau distribusi agar kerusakan file dapat dideteksi.
- Uji pergantian kernel pada tahun batas 1849/1850 dan 2149/2150.
- Pastikan kernel memuat objek `earth`, `sun`, dan `moon`; V8.0 melakukan pemeriksaan ini ketika memuat file.
- Jangan mengganti nama berkas tanpa memperbarui konstanta konfigurasi source code.

B. Peran Utilitas Pemotong/Penggabung BSP

Utilitas pemotong atau penggabung kernel tetap berguna untuk eksperimen pengelolaan SPK, tetapi bukan bagian dari alur wajib V8.0. Untuk reproduksibilitas ilmiah, penggunaan kernel resmi yang identitas dan checksum-nya terdokumentasi lebih mudah diaudit daripada kernel hasil pemotongan yang tidak diberi metadata versi.

LAMPIRAN 3: DEPENDENSI DAN LINGKUNGAN EKSEKUSI

Daftar berikut diturunkan dari import statement pada source code V8.0. Modul standar Python tidak memerlukan instalasi pip terpisah, sedangkan pustaka pihak ketiga harus tersedia pada lingkungan build/runtime sesuai sistem operasi.

A. Pustaka Pihak Ketiga yang Terdeteksi

- CTkToolTip
- PIL
- bs4
- customtkinter
- duckduckgo_search
- ephem
- matplotlib
- numpy
- pygame
- pytz
- requests
- scipy
- skyfield
- win10toast

B. Modul Standar/Platform yang Digarap Source Code

- calendar
- collections
- csv
- datetime
- hashlib
- io
- json
- math
- os
- platform
- random
- re
- ssl
- subprocess
- sys
- textwrap

- threading
- time
- tkinter
- traceback
- urllib
- warnings
- webbrowser
- winsound

C. Berkas dan Komponen Eksternal

- de440s.bsp dan de441-new.bsp
- Aset gambar/font yang dirujuk aplikasi, bila fitur terkait digunakan
- Koneksi internet untuk WinAI, basis data daring, Peta Langit Dunia, dan beberapa integrasi web
- Stellarium bila pengguna mengaktifkan fitur komparasi
- LaporanAstronimi.exe atau LaporanAstronomi.exe untuk Menu 40 di Windows

Verifikasi source: AST Python berhasil diparsing; 18,160 baris; SHA-256
e90f9e2bf766a78e0548df65fd2a1a785a9b529f8fb65e9921b31530fc3ef32a.

LAMPIRAN 4

SOURCE CODE KHGT TIMES V8.0

Salinan untuk dokumentasi akademik - sanitasi publikasi

Berkas sumber: khgttimes_v8.0(4).py

Jumlah baris: 18,160

Ukuran: 1,025,843 byte

SHA-256 source asli: e90f9e2bf766a78e0548df65fd2a1a785a9b529f8fb65e9921b31530fc3ef32a

Halaman kode: 260 (landscape A4)

CATATAN KEAMANAN

- Nilai API key dan password lokal yang terdeteksi pada source asli tidak dicetak ulang.
- Placeholder REDACTED mempertahankan lokasi dan struktur baris untuk keperluan dokumentasi.
- Source asli yang diunggah pengguna tidak diubah oleh proses pembuatan buku.
- Nomor baris di sisi kiri merujuk pada source V8.0 asli sebanyak 18.160 baris.

Source: klggtimes v8.0(4).py | SHA-256: e90f9e2bf766a/8e0548... Lampiran 4 - Halaman 1/260

Source: khgttimes_v8.0(4).py | SHA-256: e90f9e2bf766a78e0548...

```

00134 | " [V7.6 - LIVE ANIMASI] Sinkronisasi Stellarium": {
00135 |   "en": "[V7.6 - LIVE ANIMATION] Stellarium Synchronization",
00136 |   "ar": "[V7.6 - LIVE ANIMATION] Stellarium Synchronization",
00137 | },
00138 | "Pada output Live Animasi ditambahkan tombol Sinkronisasi Stellarium untuk mengaktifkan Stellarium dan membandingkan data Matahari dan Bulan antara output Stellarium dengan hasil hitung sistem KHGT
berbasis ephemeris DE441.": {
00139 |   "en": "The Live Animation output now includes a Stellarium Synchronization button to launch Stellarium and compare its Sun and Moon data with calculation results from the KHGT system based on DE441
ephemeris.",
00140 |   "ar": "ماتن تاباسج جئاتنب هتاجرم يف رقمقلاو سيمشلا تانايب هنراقمو هليغشبتل Stellarium عم هنمازمل رز رشابمل ليرجتلا تاجرم ول فقيص",
00141 | },
00142 | " [V7.6 - LIVE ANIMASI] Tracker Benda Langit": {
00143 |   "en": "[V7.6 - LIVE ANIMATION] Celestial Object Tracker",
00144 |   "ar": "[V7.6 - LIVE ANIMATION] Celestial Object Tracker",
00145 | },
00146 | "Pada output Live Animasi ditambahkan tombol Tracker untuk pelacakan posisi benda langit, baik menggunakan GPS maupun tanpa GPS.": {
00147 |   "en": "The Live Animation output now includes a Tracker button for tracking celestial object positions, with or without GPS.",
00148 |   "ar": "هنودب و GPS مادختساب ءاوس ،هيوامسلما مارحألا عقاوم دصرل عبتتلا رز رشابمل ليرجتلا تاجرم ول فقيص",
00149 | },
00150 | " [V7.5 - LIVE ANIMASI] Rasi Bintang & Planet": {
00151 |   "en": "[V7.5 - LIVE ANIMATION] Constellations & Planets",
00152 |   "ar": "[V7.5 - LIVE ANIMATION] Constellations & Planets",
00153 | },
00154 | "Pembaruan pada Modul 36 (Live Animasi) dilengkapi visualisasi Rasi Bintang (Constellations) dan planet-planet tata surya secara dinamis dan real-time.": {
00155 |   "en": "The update to Module 36 (Live Animation) adds dynamic, real-time visualizations of constellations and the planets of the solar system.",
00156 |   "ar": "يلعفل تاقولا يف هيسمشل ةعومجمل بكاوكو تابكوللل ايلكيمانيد ائيرم ايرع رشابمل ليرجتلا 36 ةدحولا شيدحت فقيص",
00157 | },
00158 | " [V7.5 - WINAI] Asisten AI Universal": {
00159 |   "en": "[V7.5 - WINAI] Universal AI Assistant",
00160 |   "ar": "[V7.5 - WINAI] Universal AI Assistant",
00161 | },
00162 | "Peningkatan kecerdasan WinAI Interceptor agar mampu mendeteksi niat (intent) pengguna dan mengeksekusi pertanyaan atau perintah perhitungan yang berkaitan dengan menu di dalam sistem.": {
00163 |   "en": "WinAI Interceptor has been enhanced to detect user intent and execute questions or calculation commands related to menus within the system.",
00164 |   "ar": "ماتنل لاجد مئاوقلاب ةقلمتمل باسجلا رماوا و تاراسفتسالا ذفنيو مدختسمل دمق يلع فرعتل WinAI Interceptor ءالك نيسجت مت",
00165 | },
00166 | " [V7.5 - ANALEMMA] Grafik Kurva Matahari": {
00167 |   "en": "[V7.5 - ANALEMMA] Solar Curve Chart",
00168 |   "ar": "[V7.5 - ANALEMMA] Solar Curve Chart",
00169 | },
00170 | "Peningkatan visual pada Modul 33 (Generator Analemma Matahari) yang menampilkan grafik kurva lintasan Matahari berbentuk angka 8.": {
00171 |   "en": "The visual presentation of Module 33 (Solar Analemma Generator) has been enhanced to display the Sun's path as a figure-eight curve.",
00172 |   "ar": "8 مقررلا لكش يلع سيمشلا راسم ينحنم ططخم راهطال (سيمشلا ا م ل ا ن ا د لوم) 33 ةدحولا يف ئيرمل ضرعل نيسجت مت",
00173 | },
00174 | " [V7.5 - BUG FIXES] Gerhana, Kiblat & Optimalisasi": {
00175 |   "en": "[V7.5 - BUG FIXES] Eclipses, Qibla & Optimization",
00176 |   "ar": "[V7.5 - BUG FIXES] Eclipses, Qibla & Optimization",
00177 | },
00178 | "Perbaikan bug pada rentang waktu komputasi di Modul 30 (Analisis Gerhana), perbaikan akurasi pada Modul 26 (Qiblah Direction and Times), serta berbagai penyesuaian stabilitas sistem.": {
00179 |   "en": "Bugs in computation time ranges in Module 30 (Eclipse Analysis) have been fixed, accuracy in Module 26 (Qiblah Direction and Times) has been corrected, and various system stability
adjustments have been made.",
00180 |   "ar": "ماتنل رارقستسا نيسجتل ةعونتم تاليدغت ءارجا و ،هتاقولا و هلقلا هاجتا 26 ةدحولا يف ةقدلا جيحمت ،فوسخلال فوسكللا ليلحت 30 ةدحولا يف باسجلل هينمرلا تاقاطنل ءاطخ ا لامل مت",
00181 | },
00182 | " [V7.4 - ENGINE] Penyederhanaan Ephemeris DE441": {
00183 |   "en": "[V7.4 - ENGINE] DE441 Ephemeris Simplification",
00184 |   "ar": "[V7.4 - ENGINE] DE441 Ephemeris Simplification",
00185 | },
00186 | "Sistem menggunakan file ephemeris 'de441-new.bsp' untuk kalkulasi pada rentang waktu yang didukung aplikasi, sehingga pengelolaan basis ephemeris menjadi lebih sederhana.": {
00187 |   "en": "The system uses the ephemeris file 'de441-new.bsp' for calculations over the time range supported by the application, simplifying ephemeris data management.",
00188 |   "ar": "هكلكفلا لواءجل تانايب ةراد طسبي ام ،قيبطتل همدعي يذلا ينمرلا قاطنل نمض تاباسجلل 'de441-new.bsp' هكلكفلا لواءجل فلم ماتنل مدختسي",
00189 | },
00190 | " [V7.3 - HIGHLIGHTS] Modul Fisis & Auto-Builder": {
00191 |   "en": "[V7.3 - HIGHLIGHTS] Physical Modules & Auto-Builder",
00192 |   "ar": "[V7.3 - HIGHLIGHTS] Physical Modules & Auto-Builder",
00193 | },
00194 | "Modul Astrofotografi, Prediktor Pasang Surut, Evaluasi SQM Fajar, dan HIJRI_DB Auto-Builder untuk Administrator merupakan pengembangan yang diperkenalkan pada rilis sebelumnya.": {
00195 |   "en": "The Astrophotography, Tide Predictor, Dawn SQM Evaluation, and HIJRI_DB Auto-Builder modules for administrators were introduced in earlier releases.",
00196 |   "ar": "هقباس تارادصا يف تم دق تاريوطت يه نيلووسملل HIJRI_DB Auto-Builder ،رقتلا دنح SQM ميقوتو ،رعتلاو دملاب قبتلاو ،هكلكفلا ريوطتلا تادح",
00197 | },
00198 | "Aplikasi ini dikembangkan secara khusus sebagai perangkat lunak komprehensif dan mutakhir untuk perhitungan astrometri presisi tinggi, pemetaan visibilitas hilal global, dan penyatuan penanggalan
Islam internasional serta menjadi sarana edukasi publik terkait ilmu falak Islami dan transisi penerapan KHGT di masa depan. Pada V8.0, sistem menyediakan 41 modul menu terintegrasi, antarmuka multibahasa
Indonesia-Inggris-Arab, Peta Langit Dunia online pada Menu 39, akses ke aplikasi pendamping Laporan Astronomi pada Menu 40, serta HIJRI_DB Auto-Builder (Admin) pada Menu 41.": {
00199 |   "en": "This application is specifically developed as comprehensive and up-to-date software for high-precision astrometric calculations, global crescent-visibility mapping, and the unification of

```

```
international Islamic calendaring, while also serving as a public educational tool for Islamic astronomy and the future transition toward KHGT implementation. In V8.0, the system provides 41 integrated menu modules, an Indonesian-English-Arabic multilingual interface, an online World Sky Map in Menu 39, access to the Astronomy Report companion application in Menu 40, and HIJRI_DB Auto-Builder (Admin) in Menu 41.',
00200 |         "ar": "الماش "اجم انرب نولكل "اصي صق قيبطلا اذه ريوطت مت",
|         "en": "The program is a public educational tool for Islamic astronomy and the future transition toward KHGT implementation. In V8.0, the system provides 41 integrated menu modules, an Indonesian-English-Arabic multilingual interface, an online World Sky Map in Menu 39, access to the Astronomy Report companion application in Menu 40, and HIJRI_DB Auto-Builder (Admin) in Menu 41.",
00201 |     },
00202 |     "INFORMASI RILIS & DOKUMENTASI SISTEM KHGT TIMES V8.0": {
00203 |         "en": "KHGT TIMES V8.0 RELEASE INFORMATION & SYSTEM DOCUMENTATION",
00204 |         "ar": "KHGT TIMES V8.0 ماطن قيبطوتو رادملل تامولع",
00205 |     },
00206 |     "Selamat datang di KHGT Times V8.0": {
00207 |         "en": "Welcome to KHGT Times V8.0",
00208 |         "ar": "KHGT Times V8.0 يف مكب "ابحرم",
00209 |     }
00210 | }
00211 |
00212 | # --- GUI LIBRARIES ---
00213 | import customtkinter as ctk
00214 | import tkinter as tk
00215 | from tkinter import messagebox, filedialog, ttk
00216 | from PIL import Image, ImageTk, ImageDraw, ImageFont, ImageGrab
00217 | from CTKToolTip import *
00218 |
00219 | # --- SKYFIELD LIBRARIES ---
00220 | if not hasattr(CTKToolTip, 'block_update_dimensions_event'):
00221 |     setattr(CTKToolTip, 'block_update_dimensions_event', lambda self: None)
00222 | if not hasattr(CTKToolTip, 'unblock_update_dimensions_event'):
00223 |     setattr(CTKToolTip, 'unblock_update_dimensions_event', lambda self: None)
00224 | # -----
00225 | from skyfield.api import Loader, wgs84
00226 | from skyfield import almanac, eclipselib
00227 | from skyfield.positionlib import Geocentric
00228 |
00229 | # --- MATPLOTLIB LIBRARIES ---
00230 | import matplotlib
00231 | matplotlib.use('TkAgg')
00232 | import matplotlib.pyplot as plt
00233 | import matplotlib.patches as mpatches
00234 | from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
00235 | from matplotlib.colors import ListedColormap, BoundaryNorm
00236 |
00237 | # --- TAMBAHAN DEPENDENSI UNTUK PETA HD & INTERPOLASI ---
00238 | try:
00239 |     import scipy.interpolate as interp
00240 |     HAS_MAP_TOOLS = True
00241 | except ImportError:
00242 |     HAS_MAP_TOOLS = False
00243 |
00244 | # --- TAMBAHAN DEPENDENSI MODUL 08: ALARM & NOTIFIKASI ---
00245 | try:
00246 |     from win10toast import ToastNotifier
00247 |     HAS_TOAST = True
00248 | except ImportError:
00249 |     HAS_TOAST = False
00250 |
00251 | try:
00252 |     import pygame
00253 |     HAS_PYGAME = True
00254 | except ImportError:
00255 |     HAS_PYGAME = False
00256 | # -----
00257 | # =====
00258 | # PERBAIKAN CRASH GUI & SSL UNTUK EXE
00259 | # =====
00260 | if sys.stdout is None: sys.stdout = io.StringIO()
00261 | if sys.stderr is None: sys.stderr = io.StringIO()
00262 |
00263 | try:
00264 |
00265 |
```

```

00266 |     _create_unverified_https_context = ssl._create_unverified_context
00267 | except AttributeError:
00268 |     pass
00269 | else:
00270 |     ssl._create_default_https_context = _create_unverified_https_context
00271 |
00272 | if getattr(sys, 'frozen', False):
00273 |     BASE_DIR = os.path.dirname(sys.executable)
00274 | else:
00275 |     BASE_DIR = os.path.dirname(os.path.abspath(__file__))
00276 |
00277 | ctk.set_appearance_mode("Dark")
00278 | ctk.set_default_color_theme("blue")
00279 |
00280 | # TAMBAHKAN FUNGSI INI DI BAGIAN ATAS (BAWAH IMPORT)
00281 | def safe_monthrange(year, month):
00282 |     """Pengganti calendar.monthrange yang kebal tahun minus/BCE"""
00283 |     is_leap = year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
00284 |     days_in_month = [31, 29 if is_leap else 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
00285 |     return 0, days_in_month[month - 1]
00286 |
00287 | def format_tahun_aman(year):
00288 |     """Format tahun agar 0 menjadi 1 SM, -1 menjadi 2 SM, dst"""
00289 |     return f"{year} M" if year > 0 else f"{abs(year-1)} SM"
00290 |
00291 | # =====
00292 | # FUNGSI UTILITAS GAMBAR JADWAL SHALAT
00293 | # (Disisipkan dari aplikasi terpisah)
00294 | # =====
00295 |
00296 | def _util_get_nama_bulan(angka_bulan):
00297 |     bulan = {
00298 |         '01': 'JANUARI', '02': 'FEBRUARI', '03': 'MARET', '04': 'APRIL',
00299 |         '05': 'MEI', '06': 'JUNI', '07': 'JULI', '08': 'AGUSTUS',
00300 |         '09': 'SEPTEMBER', '10': 'OKTOBER', '11': 'NOVEMBER', '12': 'DESEMBER'
00301 |     }
00302 |     return bulan.get(angka_bulan, "")
00303 |
00304 | def _util_parse_jadwal_text(raw_text):
00305 |     data = []
00306 |     lines = raw_text.strip().split('\n')
00307 |
00308 |     bulan_angka = "01"
00309 |     tahun = "2026"
00310 |     koordinat_str = ""
00311 |
00312 |     import re
00313 |     for line in lines:
00314 |         # PERBAIKAN: Tangkap LOKASI lengkap dengan nama kota dan provinsinya
00315 |         if "- LOKASI:" in line:
00316 |             koordinat_str = line.split("- LOKASI:")[1].strip()
00317 |
00318 |             match_judul = re.search(r'Prayer times for:\s*.*?(\d{2})/(\d{4})', line)
00319 |             if match_judul:
00320 |                 bulan_angka = match_judul.group(1)
00321 |                 tahun = match_judul.group(2)
00322 |
00323 |             # Ekstrak data tabel menghindari pergeseran spasi
00324 |             match_row = re.match(r'^\s*(\d{2}/\d{2}/\d{4}).*?\s{2,}(.*)', line)
00325 |             if match_row:
00326 |                 tanggal_full = match_row.group(1).strip()
00327 |                 hari = str(int(tanggal_full.split('/')[0]))
00328 |
00329 |                 times_str = match_row.group(2)
00330 |                 parts = [p for p in times_str.strip().split(' ') if p]
00331 |
00332 |                 if len(parts) >= 8: # Ada 8 Kolom (Subuh s/d Midnight)
00333 |                     data.append({
00334 |                         'hari': hari,
00335 |                         'subuh': parts[0],
00336 |                         'terbit': parts[1],

```

```

00337 |             'duha': parts[2],
00338 |             'zuhur': parts[3],
00339 |             'asar': parts[4],
00340 |             'magrib': parts[5],
00341 |             'isya': parts[6],
00342 |             'midnight': parts[7]
00343 |         })
00344 |
00345 |     nama_bulan_upper = _util_get_nama_bulan(bulan_angka)
00346 |     return data, nama_bulan_upper, bulan_angka, tahun, koordinat_str
00347 |
00348 | def _util_generate_image(raw_text, template_path, output_path):
00349 |     jadwal_data, bulan_upper, bulan_angka, tahun, koordinat_str = _util_parse_jadwal_text(raw_text)
00350 |
00351 |     if not jadwal_data:
00352 |         raise ValueError("Data jadwal harian tidak ditemukan dalam teks laporan. Gambar tidak dibuat.")
00353 |
00354 |     try:
00355 |         img = Image.open(template_path)
00356 |     except FileNotFoundError:
00357 |         raise FileNotFoundError(f"File template '{template_path}' tidak ditemukan di folder aplikasi.")
00358 |
00359 |     draw = ImageDraw.Draw(img)
00360 |
00361 |     try:
00362 |         font_judul = ImageFont.truetype(os.path.join(BASE_DIR, "arialbd.ttf"), 60)
00363 |         font_sub_judul1 = ImageFont.truetype(os.path.join(BASE_DIR, "arialbd.ttf"), 36)
00364 |         font_sub_judul2 = ImageFont.truetype(os.path.join(BASE_DIR, "arial.ttf"), 26)
00365 |         font_header = ImageFont.truetype(os.path.join(BASE_DIR, "arialbd.ttf"), 34)
00366 |         font_sub_header = ImageFont.truetype(os.path.join(BASE_DIR, "arial.ttf"), 22)
00367 |         font_tabel = ImageFont.truetype(os.path.join(BASE_DIR, "arialbd.ttf"), 34)
00368 |     except IOError:
00369 |         font_judul = font_sub_judul1 = font_sub_judul2 = font_header = font_sub_header = font_tabel = ImageFont.load_default()
00370 |
00371 |     center_x = img.width // 2
00372 |     col_x = {
00373 |         'hari': center_x - 860,
00374 |         'tanggal': center_x - 650,
00375 |         'subuh': center_x - 455,
00376 |         'terbit': center_x - 265,
00377 |         'duha': center_x - 80,
00378 |         'zuhur': center_x + 120,
00379 |         'asar': center_x + 305,
00380 |         'magrib': center_x + 495,
00381 |         'isya': center_x + 680,
00382 |         'midnight': center_x + 880
00383 |     }
00384 |
00385 |     start_y_judul = 120
00386 |     row_spacing_judul = 60
00387 |     start_y_header = 330
00388 |     row_spacing_header = 45
00389 |     start_y_data = 430
00390 |     row_spacing_data = 40
00391 |
00392 |     warna_putih = (255, 255, 255)
00393 |     warna_emas = (255, 204, 0)
00394 |     warna_kotak_gelap = (20, 20, 20)
00395 |
00396 |     draw.text((center_x, start_y_judul), "JADWAL SHALAT", font=font_judul, fill=warna_emas, anchor="mm")
00397 |     teks_bulan_tahun = f"BULAN {bulan_upper} {tahun}"
00398 |     draw.text((center_x, start_y_judul + row_spacing_judul), teks_bulan_tahun, font=font_sub_judul1, fill=warna_putih, anchor="mm")
00399 |
00400 |     # PERBAIKAN: Menampilkan Kota dan Provinsi dengan rapi
00401 |     if koordinat_str:
00402 |         teks_wilayah = f"LOKASI: {koordinat_str.upper()}"
00403 |     else:
00404 |         teks_wilayah = "LOKASI: DATA TIDAK DITEMUKAN"
00405 |
00406 |     draw.text((center_x, start_y_judul + 2 * row_spacing_judul), teks_wilayah, font=font_sub_judul2, fill=warna_putih, anchor="mm")
00407 |

```



```

00408 | padding_kotak_header = 1100
00409 | kotak_header_x0 = center_x - padding_kotak_header
00410 | kotak_header_y0 = start_y_header - 35
00411 | kotak_header_x1 = center_x + padding_kotak_header
00412 | kotak_header_y1 = start_y_header + 35
00413 | draw.rounded_rectangle([kotak_header_x0, kotak_header_y0, kotak_header_x1, kotak_header_y1], fill=warna_kotak_gelap, radius=10)
00414 |
00415 | headers = [
00416 |     ("NOMOR", 'hari'), ("TANGGAL", 'tanggal'), ("SUBUH", 'subuh'), ("TERBIT", 'terbit'),
00417 |     ("DUHA", 'duha'), ("ZUHUR", 'zuhur'), ("ASAR", 'asar'), ("MAGRIB", 'magrib'),
00418 |     ("ISYA", 'isya'), ("TENGAH MALAM", 'midnight')
00419 | ]
00420 | for text, key in headers:
00421 |     draw.text((col_x[key], start_y_header), text, font=font_header, fill=warna_putih, anchor="mm")
00422 |
00423 | sub_headers = [
00424 |     ("", 'hari'), ("", 'tanggal'), ("B. Twi.", 'subuh'), ("Sunrise", 'terbit'),
00425 |     ("+4.5°", 'duha'), ("Transit", 'zuhur'), ("-----", 'asar'), ("Sunset", 'magrib'),
00426 |     ("E. Twi.", 'isya'), ("(Mid/Last 1/3)", 'midnight')
00427 | ]
00428 | for text, key in sub_headers:
00429 |     draw.text((col_x[key], start_y_header + row_spacing_header), text, font=font_sub_header, fill=warna_putih, anchor="mm")
00430 |
00431 | for i, row in enumerate(jadwal_data):
00432 |     current_y = start_y_data + (i * row_spacing_data)
00433 |     tanggal_hari = row['hari'].zfill(2)
00434 |     tanggal_str = f"{tanggal_hari}/{bulan_angka}/{tahun}"
00435 |
00436 |     color = warna_emas if row['hari'] == '1' else warna_putih
00437 |
00438 |     draw.text((col_x['hari'], current_y), row['hari'], font=font_tabel, fill=color, anchor="mm")
00439 |     draw.text((col_x['tanggal'], current_y), tanggal_str, font=font_tabel, fill=warna_putih, anchor="mm")
00440 |     draw.text((col_x['subuh'], current_y), row['subuh'], font=font_tabel, fill=warna_putih, anchor="mm")
00441 |     draw.text((col_x['terbit'], current_y), row['terbit'], font=font_tabel, fill=warna_putih, anchor="mm")
00442 |     draw.text((col_x['duha'], current_y), row['duha'], font=font_tabel, fill=warna_putih, anchor="mm")
00443 |     draw.text((col_x['zuhur'], current_y), row['zuhur'], font=font_tabel, fill=warna_putih, anchor="mm")
00444 |     draw.text((col_x['asar'], current_y), row['asar'], font=font_tabel, fill=warna_putih, anchor="mm")
00445 |     draw.text((col_x['magrib'], current_y), row['magrib'], font=font_tabel, fill=warna_putih, anchor="mm")
00446 |     draw.text((col_x['isya'], current_y), row['isya'], font=font_tabel, fill=warna_putih, anchor="mm")
00447 |     draw.text((col_x['midnight'], current_y), row['midnight'], font=font_tabel, fill=warna_putih, anchor="mm")
00448 |
00449 |     img.save(output_path)
00450 |     return output_path
00451 |
00452 |
00453 | # =====
00454 | # DATABASE HIJRIAH & KALENDER
00455 | # =====
00456 | BULAN_MASEHI = ["Januari", "Februari", "Maret", "April", "Mei", "Juni", "Juli", "Agustus", "September", "Oktober", "November", "Desember"]
00457 | BULAN_HIJRIAH = ["Muharam", "Safar", "Rabiulawal", "Rabiulakhir", "Jumadilawal", "Jumadilakhir", "Rajab", "Syakban", "Ramadan", "Syawal", "Zulkaidah", "Zulhijah"]
00458 |
00459 | HIJRI_DB = {
00460 |     1446: [{"Muharam", "Ahad Kliwon", "07-Jul-2024", 29}, {"Safar", "Senin", "05-Aug-2024", 30}, {"Rabiulawal", "Rabu", "04-Sep-2024", 30}, {"Rabiulakhir", "Jumat", "04-Oct-2024", 30}, {"Jumadilawal",
00461 | "Ahad", "03-Nov-2024", 29}, {"Jumadilakhir", "Senin", "02-Dec-2024", 30}, {"Rajab", "Rabu", "01-Jan-2025", 30}, {"Syakban", "Jumat", "31-Jan-2025", 29}, {"Ramadan", "Sabtu", "01-Mar-2025", 30}, {"Syawal",
00462 | "Ahad", "31-Mar-2025", 30}, {"Zulkaidah", "Selasa", "29-Apr-2025", 29}, {"Zulhijah", "Rabu", "28-May-2025", 29}],
00463 |     1447: [{"Muharam", "Kamis Wage", "26-Jun-2025", 30}, {"Safar", "Sabtu Wage", "26-Jul-2025", 29}, {"Rabiulawal", "Ahad Pon", "24-Aug-2025", 30}, {"Rabiulakhir", "Selasa Pon", "23-Sep-2025", 30},
00464 | {"Jumadilawal", "Kamis Pon", "23-Oct-2025", 29}, {"Jumadilakhir", "Jumat Pahing", "21-Nov-2025", 30}, {"Rajab", "Ahad Pahing", "21-Dec-2025", 30}, {"Syakban", "Selasa Pahing", "20-Jan-2026", 29},
00465 | {"Ramadan", "Rabu Legi", "18-Feb-2026", 30}, {"Syawal", "Jumat Legi", "20-Mar-2026", 29}, {"Zulkaidah", "Sabtu Kliwon", "18-Apr-2026", 30}, {"Zulhijah", "Senin Kliwon", "18-May-2026", 29}],
00466 |     1448: [{"Muharam", "Selasa Wage", "16-Jun-2026", 29}, {"Safar", "Rabu", "15-Jul-2026", 30}, {"Rabiulawal", "Jumat", "14-Aug-2026", 29}, {"Rabiulakhir", "Sabtu", "12-Sep-2026", 30}, {"Jumadilawal",
00467 | "Senin", "12-Oct-2026", 29}, {"Jumadilakhir", "Selasa", "10-Nov-2026", 30}, {"Rajab", "Kamis", "10-Dec-2026", 30}, {"Syakban", "Sabtu", "09-Jan-2027", 30}, {"Ramadan", "Senin", "08-Feb-2027", 29},
00468 | {"Syawal", "Selasa", "09-Mar-2027", 30}, {"Zulkaidah", "Kamis", "08-Apr-2027", 29}, {"Zulhijah", "Jumat Wage", "07-May-2027", 30}],
00469 |     #... [Data Hijriah disingkat untuk keterbacaan, logikanya tetap memuat data lengkap seperti aslinya]
00470 | }
00471 |
00472 | # (Memasukkan sisa HIJRI_DB tahun 1449-1497 agar converter tetap berjalan sempurna sesuai kode master Anda)
00473 | HIJRI_DB.update({
00474 |     1449: [{"Muharam", "Ahad Wage", "06-Jun-2027", 29}, {"Safar", "Senin", "05-Jul-2027", 29}, {"Rabiulawal", "Selasa", "03-Aug-2027", 30}, {"Rabiulakhir", "Kamis", "02-Sep-2027", 29}, {"Jumadilawal",
00475 | "Jumat", "01-Oct-2027", 30}, {"Jumadilakhir", "Ahad", "31-Oct-2027", 29}, {"Rajab", "Senin", "29-Nov-2027", 30}, {"Syakban", "Rabu", "29-Dec-2027", 29}, {"Ramadan", "Jumat", "28-Jan-2028", 30}, {"Syawal",
00476 | "Sabtu", "26-Feb-2028", 30}, {"Zulkaidah", "Senin", "27-Mar-2028", 30}, {"Zulhijah", "Rabu", "26-Apr-2028", 29}],
00477 |     1450: [{"Muharam", "Kamis", "25-May-2028", 30}, {"Safar", "Sabtu", "24-Jun-2028", 29}, {"Rabiulawal", "Ahad", "23-Jul-2028", 29}, {"Rabiulakhir", "Senin", "21-Aug-2028", 30}, {"Jumadilawal", "Rabu",
00478 | "20-Sep-2028", 29}, {"Jumadilakhir", "Kamis", "19-Oct-2028", 30}, {"Rajab", "Sabtu", "18-Nov-2028", 29}, {"Syakban", "Ahad", "17-Dec-2028", 30}, {"Ramadan", "Selasa", "16-Jan-2029", 29}, {"Syawal", "Rabu",

```

00470 | "14-Feb-2029", 30], ["Zulkaidah", "Jumat", "16-Mar-2029", 30], ["Zulhijah", "Ahad", "15-Apr-2029", 29]],
1451: [{"Muhamaram", "Senin", "14-May-2029", 30}, {"Safar", "Rabu", "13-Jun-2029", 29}, {"Rabiulawal", "Kamis", "12-Jul-2029", 30}, {"Rabiulakhir", "Sabtu", "11-Aug-2029", 29}, {"Jumadilawal", "Ahad", "09-Sep-2029", 30}, {"Jumadilakhir", "Selasa", "09-Oct-2029", 29}, {"Rajab", "Rabu", "07-Nov-2029", 30}, {"Syakban", "Jumat", "07-Dec-2029", 29}, {"Ramadan", "Sabtu", "05-Jan-2030", 30}, {"Syawal", "Senin", "04-Feb-2030", 29}, {"Zulkaidah", "Selasa", "05-Mar-2030", 30}, {"Zulhijah", "Kamis", "04-Apr-2030", 29}],
00471 | 1452: [{"Muhamaram", "Jumat", "03-May-2030", 30}, {"Safar", "Ahad", "02-Jun-2030", 30}, {"Rabiulawal", "Selasa", "02-Jul-2030", 30}, {"Rabiulakhir", "Kamis", "01-Aug-2030", 29}, {"Jumadilawal", "Jumat", "30-Aug-2030", 29}, {"Jumadilakhir", "Sabtu", "28-Sep-2030", 30}, {"Rajab", "Senin", "28-Oct-2030", 29}, {"Syakban", "Selasa", "26-Nov-2030", 30}, {"Ramadan", "Kamis", "26-Dec-2030", 29}, {"Syawal", "Jumat", "24-Jan-2031", 29}, {"Zulkaidah", "Sabtu", "22-Feb-2031", 30}, {"Zulhijah", "Senin", "22-Mar-2031", 30}],
00472 | 1453: [{"Muhamaram", "Rabu", "23-Apr-2031", 29}, {"Safar", "Kamis", "22-May-2031", 30}, {"Rabiulawal", "Sabtu", "21-Jun-2031", 30}, {"Rabiulakhir", "Senin", "21-Jul-2031", 29}, {"Jumadilawal", "Selasa", "19-Aug-2031", 30}, {"Jumadilakhir", "Kamis", "18-Sep-2031", 29}, {"Rajab", "Jumat", "17-Oct-2031", 30}, {"Syakban", "Ahad", "16-Nov-2031", 30}, {"Ramadan", "Senin", "15-Dec-2031", 29}, {"Syawal", "Rabu", "14-Jan-2032", 29}, {"Zulkaidah", "Kamis", "12-Feb-2032", 29}, {"Zulhijah", "Jumat", "12-Mar-2032", 30}],
00473 | 1454: [{"Muhamaram", "Ahad", "11-Apr-2032", 30}, {"Safar", "Selasa", "11-May-2032", 29}, {"Rabiulawal", "Rabu", "09-Jun-2032", 30}, {"Rabiulakhir", "Jumat", "09-Jul-2032", 29}, {"Jumadilawal", "Sabtu", "07-Aug-2032", 30}, {"Jumadilakhir", "Senin", "06-Sep-2032", 29}, {"Rajab", "Selasa", "05-Oct-2032", 30}, {"Syakban", "Kamis", "04-Nov-2032", 30}, {"Ramadan", "Sabtu", "04-Dec-2032", 29}, {"Syawal", "Ahad", "02-Jan-2033", 29}, {"Zulkaidah", "Selasa", "01-Feb-2033", 29}, {"Zulhijah", "Rabu", "02-Mar-2033", 30}],
00474 | 1455: [{"Muhamaram", "Jumat", "01-Apr-2033", 29}, {"Safar", "Sabtu", "30-Apr-2033", 29}, {"Rabiulawal", "Ahad", "29-May-2033", 30}, {"Rabiulakhir", "Selasa", "28-Jun-2033", 29}, {"Jumadilawal", "Rabu", "27-Jul-2033", 30}, {"Jumadilakhir", "Jumat", "26-Aug-2033", 29}, {"Rajab", "Sabtu", "24-Sep-2033", 30}, {"Syakban", "Senin", "24-Oct-2033", 30}, {"Ramadan", "Rabu", "23-Nov-2033", 30}, {"Syawal", "Jumat", "23-Dec-2033", 29}, {"Zulkaidah", "Sabtu", "21-Jan-2034", 30}, {"Zulhijah", "Senin", "20-Feb-2034", 29}],
00475 | 1456: [{"Muhamaram", "Selasa", "21-Mar-2034", 30}, {"Safar", "Kamis", "20-Apr-2034", 29}, {"Rabiulawal", "Jumat", "19-May-2034", 29}, {"Rabiulakhir", "Sabtu", "17-Jun-2034", 30}, {"Jumadilawal", "Senin", "17-Jul-2034", 29}, {"Jumadilakhir", "Selasa", "15-Aug-2034", 29}, {"Rajab", "Rabu", "13-Sep-2034", 30}, {"Syakban", "Jumat", "13-Oct-2034", 30}, {"Ramadan", "Ahad", "12-Nov-2034", 30}, {"Syawal", "Selasa", "12-Dec-2034", 30}, {"Zulkaidah", "Kamis", "11-Jan-2035", 29}, {"Zulhijah", "Jumat", "09-Feb-2035", 30}],
00476 | 1457: [{"Muhamaram", "Ahad", "11-Mar-2035", 29}, {"Safar", "Senin", "09-Apr-2035", 30}, {"Rabiulawal", "Rabu", "09-May-2035", 29}, {"Rabiulakhir", "Kamis", "07-Jun-2035", 29}, {"Jumadilawal", "Jumat", "06-Jul-2035", 29}, {"Jumadilakhir", "Sabtu", "04-Aug-2035", 30}, {"Rajab", "Senin", "03-Sep-2035", 30}, {"Syakban", "Rabu", "03-Oct-2035", 29}, {"Ramadan", "Kamis", "01-Nov-2035", 30}, {"Syawal", "Sabtu", "01-Dec-2035", 30}, {"Zulkaidah", "Senin", "31-Dec-2035", 30}, {"Zulhijah", "Rabu", "30-Jan-2036", 29}],
00477 | 1458: [{"Muhamaram", "Kamis", "28-Feb-2036", 30}, {"Safar", "Sabtu", "29-Mar-2036", 29}, {"Rabiulawal", "Ahad", "27-Apr-2036", 30}, {"Rabiulakhir", "Selasa", "27-May-2036", 29}, {"Jumadilawal", "Rabu", "25-Jun-2036", 29}, {"Jumadilakhir", "Kamis", "24-Jul-2036", 30}, {"Rajab", "Sabtu", "23-Aug-2036", 29}, {"Syakban", "Ahad", "21-Sep-2036", 29}, {"Ramadan", "Senin", "20-Oct-2036", 30}, {"Syawal", "Rabu", "19-Nov-2036", 30}, {"Zulkaidah", "Jumat", "19-Dec-2036", 30}, {"Zulhijah", "Ahad", "18-Jan-2037", 29}],
00478 | 1459: [{"Muhamaram", "Senin", "16-Feb-2037", 30}, {"Safar", "Rabu", "18-Mar-2037", 30}, {"Rabiulawal", "Jumat", "17-Apr-2037", 29}, {"Rabiulakhir", "Sabtu", "16-May-2037", 29}, {"Jumadilawal", "Ahad", "14-Jun-2037", 30}, {"Jumadilakhir", "Selasa", "14-Jul-2037", 29}, {"Rajab", "Rabu", "12-Aug-2037", 30}, {"Syakban", "Jumat", "11-Sep-2037", 29}, {"Ramadan", "Sabtu", "10-Oct-2037", 30}, {"Syawal", "Ahad", "08-Nov-2037", 30}, {"Zulkaidah", "Selasa", "08-Dec-2037", 30}, {"Zulhijah", "Kamis", "07-Jan-2038", 29}],
00479 | 1460: [{"Muhamaram", "Jumat", "05-Feb-2038", 30}, {"Safar", "Ahad", "07-Mar-2038", 30}, {"Rabiulawal", "Senin", "05-Apr-2038", 29}, {"Rabiulakhir", "Rabu", "05-May-2038", 30}, {"Jumadilawal", "Jumat", "04-Jun-2038", 29}, {"Jumadilakhir", "Sabtu", "03-Jul-2038", 30}, {"Rajab", "Senin", "02-Aug-2038", 29}, {"Syakban", "Selasa", "31-Aug-2038", 30}, {"Ramadan", "Kamis", "30-Sep-2038", 29}, {"Syawal", "Jumat", "29-Oct-2038", 29}, {"Zulkaidah", "Sabtu", "27-Nov-2038", 30}, {"Zulhijah", "Senin", "27-Dec-2038", 30}],
00480 | 1461: [{"Muhamaram", "Rabu", "26-Jan-2039", 29}, {"Safar", "Kamis", "24-Feb-2039", 30}, {"Rabiulawal", "Sabtu", "26-Mar-2039", 29}, {"Rabiulakhir", "Ahad", "24-Apr-2039", 30}, {"Jumadilawal", "Selasa", "24-May-2039", 29}, {"Jumadilakhir", "Rabu", "22-Jun-2039", 30}, {"Rajab", "Jumat", "22-Jul-2039", 30}, {"Syakban", "Ahad", "21-Aug-2039", 29}, {"Ramadan", "Senin", "19-Sep-2039", 30}, {"Syawal", "Rabu", "19-Oct-2039", 29}, {"Zulkaidah", "Kamis", "17-Nov-2039", 30}, {"Zulhijah", "Sabtu", "17-Dec-2039", 29}],
00481 | 1462: [{"Muhamaram", "Ahad", "15-Jan-2040", 29}, {"Safar", "Senin", "13-Feb-2040", 30}, {"Rabiulawal", "Rabu", "14-Mar-2040", 29}, {"Rabiulakhir", "Kamis", "12-Apr-2040", 30}, {"Jumadilawal", "Sabtu", "12-May-2040", 29}, {"Jumadilakhir", "Ahad", "10-Jun-2040", 30}, {"Rajab", "Selasa", "10-Jul-2040", 30}, {"Syakban", "Kamis", "09-Aug-2040", 30}, {"Ramadan", "Jumat", "08-Sep-2040", 30}, {"Syawal", "Ahad", "07-Oct-2040", 30}, {"Zulkaidah", "Selasa", "06-Nov-2040", 29}, {"Zulhijah", "Rabu", "05-Dec-2040", 30}],
00482 | 1463: [{"Muhamaram", "Jumat", "04-Jan-2041", 29}, {"Safar", "Sabtu", "02-Feb-2041", 29}, {"Rabiulawal", "Ahad", "03-Mar-2041", 30}, {"Rabiulakhir", "Selasa", "02-Apr-2041", 29}, {"Jumadilawal", "Rabu", "01-May-2041", 30}, {"Jumadilakhir", "Jumat", "31-May-2041", 29}, {"Rajab", "Sabtu", "29-Jun-2041", 30}, {"Syakban", "Senin", "29-Jul-2041", 29}, {"Ramadan", "Selasa", "27-Aug-2041", 30}, {"Syawal", "Kamis", "26-Sep-2041", 30}, {"Zulkaidah", "Sabtu", "26-Oct-2041", 30}, {"Zulhijah", "Senin", "25-Nov-2041", 29}],
00483 | 1464: [{"Muhamaram", "Selasa", "24-Dec-2041", 30}, {"Safar", "Kamis", "23-Jan-2042", 29}, {"Rabiulawal", "Jumat", "21-Feb-2042", 30}, {"Rabiulakhir", "Ahad", "23-Mar-2042", 29}, {"Jumadilawal", "Senin", "21-Apr-2042", 29}, {"Jumadilakhir", "Selasa", "20-May-2042", 30}, {"Rajab", "Kamis", "19-Jun-2042", 29}, {"Syakban", "Jumat", "18-Jul-2042", 30}, {"Ramadan", "Ahad", "17-Aug-2042", 29}, {"Syawal", "Senin", "16-Sep-2042", 30}, {"Zulkaidah", "Rabu", "15-Oct-2042", 30}, {"Zulhijah", "Jumat", "14-Nov-2042", 30}],
00484 | 1465: [{"Muhamaram", "Ahad", "14-Dec-2042", 29}, {"Safar", "Senin", "12-Jan-2043", 30}, {"Rabiulawal", "Rabu", "11-Feb-2043", 29}, {"Rabiulakhir", "Kamis", "12-Mar-2043", 30}, {"Jumadilawal", "Sabtu", "11-Apr-2043", 29}, {"Jumadilakhir", "Ahad", "10-May-2043", 30}, {"Rajab", "Senin", "08-Jun-2043", 29}, {"Syakban", "Selasa", "07-Jul-2043", 30}, {"Ramadan", "Kamis", "06-Aug-2043", 29}, {"Syawal", "Jumat", "04-Sep-2043", 30}, {"Zulkaidah", "Ahad", "04-Oct-2043", 30}, {"Zulhijah", "Selasa", "03-Nov-2043", 30}],
00485 | 1466: [{"Muhamaram", "Kamis", "03-Dec-2043", 29}, {"Safar", "Jumat", "01-Jan-2044", 30}, {"Rabiulawal", "Ahad", "31-Jan-2044", 30}, {"Rabiulakhir", "Selasa", "01-Mar-2044", 29}, {"Jumadilawal", "Rabu", "30-Mar-2044", 30}, {"Jumadilakhir", "Jumat", "29-Apr-2044", 29}, {"Rajab", "Sabtu", "28-May-2044", 29}, {"Syakban", "Ahad", "26-Jun-2044", 29}, {"Ramadan", "Senin", "25-Jul-2044", 30}, {"Syawal", "Rabu", "24-Aug-2044", 29}, {"Zulkaidah", "Kamis", "22-Sep-2044", 30}, {"Zulhijah", "Sabtu", "22-Oct-2044", 30}],
00486 | 1467: [{"Muhamaram", "Senin", "21-Nov-2044", 29}, {"Safar", "Selasa", "20-Dec-2044", 30}, {"Rabiulawal", "Kamis", "19-Jan-2045", 30}, {"Rabiulakhir", "Sabtu", "18-Feb-2045", 30}, {"Jumadilawal", "Senin", "20-Mar-2045", 29}, {"Jumadilakhir", "Selasa", "18-Apr-2045", 30}, {"Rajab", "Kamis", "18-May-2045", 29}, {"Syakban", "Jumat", "16-Jun-2045", 29}, {"Ramadan", "Sabtu", "15-Jul-2045", 29}, {"Syawal", "Ahad", "13-Aug-2045", 30}, {"Zulkaidah", "Selasa", "12-Sep-2045", 29}, {"Zulhijah", "Rabu", "11-Oct-2045", 30}],
00487 | 1468: [{"Muhamaram", "Jumat", "10-Nov-2045", 30}, {"Safar", "Ahad", "10-Dec-2045", 29}, {"Rabiulawal", "Senin", "08-Jan-2046", 30}, {"Rabiulakhir", "Rabu", "07-Feb-2046", 30}, {"Jumadilawal", "Jumat", "09-Mar-2046", 29}, {"Jumadilakhir", "Sabtu", "07-Apr-2046", 30}, {"Rajab", "Senin", "07-May-2046", 29}, {"Syakban", "Selasa", "05-Jun-2046", 30}, {"Ramadan", "Kamis", "05-Jul-2046", 29}, {"Syawal", "Jumat", "03-Aug-2046", 30}, {"Zulkaidah", "Ahad", "02-Sep-2046", 29}, {"Zulhijah", "Senin", "01-Oct-2046", 29}],
00488 | 1469: [{"Muhamaram", "Selasa", "30-Oct-2046", 30}, {"Safar", "Kamis", "29-Nov-2046", 29}, {"Rabiulawal", "Sabtu", "29-Dec-2046", 30}, {"Rabiulakhir", "Ahad", "27-Jan-2047", 29}, {"Jumadilawal", "Selasa", "26-Feb-2047", 29}, {"Jumadilakhir", "Kamis", "28-Mar-2047", 29}, {"Rajab", "Jumat", "26-Apr-2047", 30}, {"Syakban", "Ahad", "26-May-2047", 29}, {"Ramadan", "Selasa", "25-Jun-2047", 30}, {"Syawal", "Rabu", "24-Jul-2047", 30}, {"Zulkaidah", "Kamis", "22-Aug-2047", 30}, {"Zulhijah", "Sabtu", "21-Sep-2047", 29}],
00489 | 1470: [{"Muhamaram", "Ahad", "20-Oct-2047", 30}, {"Safar", "Selasa", "19-Nov-2047", 29}, {"Rabiulawal", "Rabu", "18-Dec-2047", 29}, {"Rabiulakhir", "Kamis", "16-Jan-2048", 30}, {"Jumadilawal", "Sabtu", "15-Feb-2048", 29}, {"Jumadilakhir", "Ahad", "15-Mar-2048", 30}, {"Rajab", "Selasa", "14-Apr-2048", 30}, {"Syakban", "Kamis", "14-May-2048", 30}, {"Ramadan", "Sabtu", "13-Jun-2048", 29}, {"Syawal", "Ahad", "12-Jul-2048", 30}, {"Zulkaidah", "Selasa", "11-Aug-2048", 29}, {"Zulhijah", "Rabu", "09-Sep-2048", 30}],
00490 | 1471: [{"Muhamaram", "Jumat", "09-Oct-2048", 29}, {"Safar", "Sabtu", "07-Nov-2048", 30}, {"Rabiulawal", "Senin", "07-Dec-2048", 29}, {"Rabiulakhir", "Selasa", "05-Jan-2049", 29}, {"Jumadilawal", "Rabu", "03-Feb-2049", 30}, {"Jumadilakhir", "Jumat", "05-Mar-2049", 29}, {"Rajab", "Sabtu", "03-Apr-2049", 30}, {"Syakban", "Senin", "03-May-2049", 30}, {"Ramadan", "Rabu", "02-Jun-2049", 29}, {"Syawal", "Kamis", "01-Jul-2049", 30}, {"Zulkaidah", "Sabtu", "31-Jul-2049", 30}, {"Zulhijah", "Senin", "30-Aug-2049", 29}],
00491 | 1472: [{"Muhamaram", "Selasa", "28-Sep-2049", 30}, {"Safar", "Kamis", "28-Oct-2049", 29}, {"Rabiulawal", "Jumat", "26-Nov-2049", 30}, {"Rabiulakhir", "Ahad", "26-Dec-2049", 29}, {"Jumadilawal", "Senin", "24-Jan-2050", 29}, {"Jumadilakhir", "Selasa", "22-Feb-2050", 30}, {"Rajab", "Kamis", "24-Mar-2050", 29}, {"Syakban", "Jumat", "22-Apr-2050", 30}, {"Ramadan", "Ahad", "22-May-2050", 29}, {"Syawal", "Senin", "20-Jun-2050", 30}, {"Zulkaidah", "Rabu", "20-Jul-2050", 30}, {"Zulhijah", "Jumat", "19-Aug-2050", 29}],
00492 | 1473: [{"Muhamaram", "Sabtu", "17-Sep-2050", 30}, {"Safar", "Senin", "17-Oct-2050", 30}, {"Rabiulawal", "Rabu", "16-Nov-2050", 29}, {"Rabiulakhir", "Kamis", "15-Dec-2050", 30}, {"Jumadilawal", "Sabtu", "14-Jan-2051", 29}, {"Jumadilakhir", "Ahad", "12-Feb-2051", 30}, {"Rajab", "Selasa", "14-Mar-2051", 29}, {"Syakban", "Rabu", "12-Apr-2051", 29}, {"Ramadan", "Kamis", "11-May-2051", 30}, {"Syawal", "Sabtu", "10-Jun-2051", 29}, {"Zulkaidah", "Ahad", "09-Jul-2051", 30}, {"Zulhijah", "Selasa", "08-Aug-2051", 29}],
00493 | 1474: [{"Muhamaram", "Rabu", "06-Sep-2051", 30}, {"Safar", "Jumat", "06-Oct-2051", 30}, {"Rabiulawal", "Ahad", "05-Nov-2051", 29}, {"Rabiulakhir", "Senin", "04-Dec-2051", 30}, {"Jumadilawal", "Rabu",

"03-Jan-2052", 30], ["Jumadilakhir", "Jumat", "02-Feb-2052", 29], ["Rajab", "Sabtu", "02-Mar-2052", 30], ["Syakban", "Senin", "01-Apr-2052", 29], ["Ramadan", "Selasa", "30-Apr-2052", 29], ["Syawal", "Rabu", "29-May-2052", 30], ["Zulkaidah", "Jumat", "28-Jun-2052", 29], ["Zulhijah", "Sabtu", "27-Jul-2052", 30]],

00494 1475: ["Muharam", "Senin", "26-Aug-2052", 29], ["Safar", "Selasa", "24-Sep-2052", 30], ["Rabiulawal", "Kamis", "24-Oct-2052", 29], ["Rabiulakhir", "Jumat", "22-Nov-2052", 30], ["Jumadilawal", "Ahad", "22-Dec-2052", 30], ["Jumadilakhir", "Selasa", "21-Jan-2053", 29], ["Rajab", "Kamis", "20-Feb-2053", 29], ["Syakban", "Jumat", "21-Mar-2053", 30], ["Ramadan", "Ahad", "20-Apr-2053", 29], ["Syawal", "Senin", "19-May-2053", 29], ["Zulkaidah", "Selasa", "17-Jun-2053", 30], ["Zulhijah", "Kamis", "17-Jul-2053", 29]],

00495 1476: ["Muharam", "Jumat", "15-Aug-2053", 29], ["Safar", "Sabtu", "13-Sep-2053", 30], ["Rabiulawal", "Senin", "13-Oct-2053", 29], ["Rabiulakhir", "Selasa", "11-Nov-2053", 30], ["Jumadilawal", "Kamis", "11-Dec-2053", 30], ["Jumadilakhir", "Sabtu", "10-Jan-2054", 30], ["Rajab", "Senin", "09-Feb-2054", 29], ["Syakban", "Rabu", "11-Mar-2054", 29], ["Ramadan", "Kamis", "09-Apr-2054", 30], ["Syawal", "Sabtu", "09-May-2054", 29], ["Zulkaidah", "Ahad", "07-Jun-2054", 29], ["Zulhijah", "Senin", "06-Jul-2054", 30]],

00496 1477: ["Muharam", "Rabu", "05-Aug-2054", 29], ["Safar", "Kamis", "03-Sep-2054", 29], ["Rabiulawal", "Jumat", "02-Oct-2054", 30], ["Rabiulakhir", "Ahad", "01-Nov-2054", 29], ["Jumadilawal", "Senin", "30-Nov-2054", 30], ["Jumadilakhir", "Rabu", "30-Dec-2054", 30], ["Rajab", "Jumat", "29-Jan-2055", 30], ["Syakban", "Ahad", "28-Feb-2055", 29], ["Ramadan", "Senin", "29-Mar-2055", 30], ["Syawal", "Rabu", "28-Apr-2055", 30], ["Zulkaidah", "Jumat", "28-May-2055", 29], ["Zulhijah", "Sabtu", "26-Jun-2055", 29]],

00497 1478: ["Muharam", "Ahad", "25-Jul-2055", 30], ["Safar", "Selasa", "24-Aug-2055", 29], ["Rabiulawal", "Rabu", "22-Sep-2055", 29], ["Rabiulakhir", "Kamis", "21-Oct-2055", 30], ["Jumadilawal", "Sabtu", "20-Nov-2055", 29], ["Jumadilakhir", "Ahad", "19-Dec-2055", 30], ["Rajab", "Selasa", "18-Jan-2056", 30], ["Syakban", "Kamis", "17-Feb-2056", 29], ["Ramadan", "Jumat", "17-Mar-2056", 30], ["Syawal", "Ahad", "16-Apr-2056", 30], ["Zulkaidah", "Selasa", "16-May-2056", 29], ["Zulhijah", "Rabu", "14-Jun-2056", 30]],

00498 1479: ["Muharam", "Jumat", "14-Jul-2056", 29], ["Safar", "Sabtu", "12-Aug-2056", 30], ["Rabiulawal", "Senin", "11-Sep-2056", 29], ["Rabiulakhir", "Selasa", "10-Oct-2056", 30], ["Jumadilawal", "Rabu", "08-Nov-2056", 30], ["Jumadilakhir", "Jumat", "08-Dec-2056", 29], ["Rajab", "Sabtu", "06-Jan-2057", 30], ["Syakban", "Senin", "05-Feb-2057", 29], ["Ramadan", "Selasa", "06-Mar-2057", 30], ["Syawal", "Kamis", "05-Apr-2057", 30], ["Zulkaidah", "Sabtu", "05-May-2057", 29], ["Zulhijah", "Ahad", "03-Jun-2057", 30]],

00499 1480: ["Muharam", "Selasa", "03-Jul-2057", 30], ["Safar", "Kamis", "02-Aug-2057", 29], ["Rabiulawal", "Jumat", "31-Aug-2057", 30], ["Rabiulakhir", "Sabtu", "29-Sep-2057", 30], ["Jumadilawal", "Senin", "29-Oct-2057", 30], ["Jumadilakhir", "Rabu", "28-Nov-2057", 29], ["Rajab", "Kamis", "27-Dec-2057", 29], ["Syakban", "Jumat", "25-Jan-2058", 30], ["Ramadan", "Ahad", "24-Feb-2058", 29], ["Syawal", "Senin", "25-Mar-2058", 30], ["Zulkaidah", "Rabu", "24-Apr-2058", 29], ["Zulhijah", "Kamis", "23-May-2058", 30]],

00500 1481: ["Muharam", "Sabtu", "22-Jun-2058", 29], ["Safar", "Ahad", "21-Jul-2058", 30], ["Rabiulawal", "Selasa", "20-Aug-2058", 30], ["Rabiulakhir", "Kamis", "19-Sep-2058", 30], ["Jumadilawal", "Sabtu", "19-Oct-2058", 29], ["Jumadilakhir", "Ahad", "17-Nov-2058", 30], ["Rajab", "Selasa", "17-Dec-2058", 29], ["Syakban", "Rabu", "15-Jan-2059", 29], ["Ramadan", "Kamis", "13-Feb-2059", 30], ["Syawal", "Sabtu", "15-Mar-2059", 29], ["Zulkaidah", "Ahad", "13-Apr-2059", 30], ["Zulhijah", "Selasa", "13-May-2059", 29]],

00501 1482: ["Muharam", "Rabu", "11-Jun-2059", 30], ["Safar", "Jumat", "11-Jul-2059", 29], ["Rabiulawal", "Sabtu", "09-Aug-2059", 30], ["Rabiulakhir", "Senin", "08-Sep-2059", 30], ["Jumadilawal", "Rabu", "08-Oct-2059", 30], ["Jumadilakhir", "Jumat", "07-Nov-2059", 29], ["Rajab", "Sabtu", "06-Dec-2059", 30], ["Syakban", "Senin", "05-Jan-2060", 29], ["Ramadan", "Selasa", "03-Feb-2060", 30], ["Syawal", "Kamis", "04-Mar-2060", 29], ["Zulkaidah", "Jumat", "02-Apr-2060", 29], ["Zulhijah", "Sabtu", "01-May-2060", 30]],

00502 1483: ["Muharam", "Senin", "31-May-2060", 29], ["Safar", "Selasa", "29-Jun-2060", 29], ["Rabiulawal", "Rabu", "28-Jul-2060", 30], ["Rabiulakhir", "Jumat", "27-Aug-2060", 30], ["Jumadilawal", "Ahad", "26-Sep-2060", 30], ["Jumadilakhir", "Selasa", "26-Oct-2060", 29], ["Rajab", "Rabu", "24-Nov-2060", 30], ["Syakban", "Jumat", "24-Dec-2060", 30], ["Ramadan", "Ahad", "23-Jan-2061", 29], ["Syawal", "Senin", "21-Feb-2061", 30], ["Zulkaidah", "Rabu", "23-Mar-2061", 29], ["Zulhijah", "Kamis", "21-Apr-2061", 29]],

00503 1484: ["Muharam", "Jumat", "20-May-2061", 29], ["Safar", "Sabtu", "18-Jun-2061", 30], ["Rabiulawal", "Senin", "18-Jul-2061", 29], ["Rabiulakhir", "Selasa", "16-Aug-2061", 30], ["Jumadilawal", "Kamis", "15-Sep-2061", 30], ["Jumadilakhir", "Sabtu", "15-Oct-2061", 29], ["Rajab", "Ahad", "13-Nov-2061", 30], ["Syakban", "Selasa", "13-Dec-2061", 30], ["Ramadan", "Kamis", "12-Jan-2062", 30], ["Syawal", "Sabtu", "11-Feb-2062", 29], ["Zulkaidah", "Ahad", "12-Mar-2062", 30], ["Zulhijah", "Selasa", "11-Apr-2062", 29]],

00504 1485: ["Muharam", "Rabu", "10-May-2062", 29], ["Safar", "Kamis", "08-Jun-2062", 29], ["Rabiulawal", "Jumat", "07-Jul-2062", 30], ["Rabiulakhir", "Ahad", "06-Aug-2062", 29], ["Jumadilawal", "Senin", "04-Sep-2062", 30], ["Jumadilakhir", "Rabu", "04-Oct-2062", 29], ["Rajab", "Kamis", "02-Nov-2062", 30], ["Syakban", "Sabtu", "02-Dec-2062", 29], ["Ramadan", "Kamis", "01-Jan-2063", 30], ["Syawal", "Rabu", "31-Jan-2063", 29], ["Zulkaidah", "Kamis", "01-Mar-2063", 30], ["Zulhijah", "Sabtu", "31-Mar-2063", 29]],

00505 1486: ["Muharam", "Ahad", "29-Apr-2063", 30], ["Safar", "Selasa", "29-May-2063", 29], ["Rabiulawal", "Rabu", "27-Jun-2063", 30], ["Rabiulakhir", "Jumat", "27-Jul-2063", 29], ["Jumadilawal", "Sabtu", "25-Aug-2063", 29], ["Jumadilakhir", "Ahad", "23-Sep-2063", 30], ["Rajab", "Selasa", "23-Oct-2063", 29], ["Syakban", "Rabu", "21-Nov-2063", 30], ["Ramadan", "Jumat", "21-Dec-2063", 30], ["Syawal", "Ahad", "20-Jan-2064", 29], ["Zulkaidah", "Senin", "18-Feb-2064", 30], ["Zulhijah", "Rabu", "19-Mar-2064", 30]],

00506 1487: ["Muharam", "Jumat", "18-Apr-2064", 29], ["Safar", "Sabtu", "17-May-2064", 30], ["Rabiulawal", "Senin", "16-Jun-2064", 29], ["Rabiulakhir", "Selasa", "15-Jul-2064", 30], ["Jumadilawal", "Kamis", "14-Aug-2064", 29], ["Jumadilakhir", "Jumat", "12-Sep-2064", 29], ["Rajab", "Sabtu", "11-Oct-2064", 30], ["Syakban", "Senin", "10-Nov-2064", 29], ["Ramadan", "Selasa", "09-Dec-2064", 30], ["Syawal", "Kamis", "08-Jan-2065", 29], ["Zulkaidah", "Jumat", "06-Feb-2065", 30], ["Zulhijah", "Ahad", "08-Mar-2065", 30]],

00507 1488: ["Muharam", "Selasa", "07-Apr-2065", 29], ["Safar", "Rabu", "06-May-2065", 30], ["Rabiulawal", "Jumat", "05-Jun-2065", 30], ["Rabiulakhir", "Ahad", "05-Jul-2065", 29], ["Jumadilawal", "Senin", "03-Aug-2065", 30], ["Jumadilakhir", "Rabu", "02-Sep-2065", 29], ["Rajab", "Kamis", "01-Oct-2065", 29], ["Syakban", "Jumat", "30-Oct-2065", 30], ["Ramadan", "Ahad", "29-Nov-2065", 29], ["Syawal", "Senin", "28-Dec-2065", 30], ["Zulkaidah", "Rabu", "27-Jan-2066", 29], ["Zulhijah", "Kamis", "25-Feb-2066", 30]],

00508 1489: ["Muharam", "Sabtu", "27-Mar-2066", 29], ["Safar", "Ahad", "25-Apr-2066", 30], ["Rabiulawal", "Selasa", "25-May-2066", 30], ["Rabiulakhir", "Kamis", "24-Jun-2066", 30], ["Jumadilawal", "Sabtu", "24-Jul-2066", 29], ["Jumadilakhir", "Ahad", "22-Aug-2066", 30], ["Rajab", "Selasa", "21-Sep-2066", 29], ["Syakban", "Rabu", "20-Oct-2066", 30], ["Ramadan", "Jumat", "19-Nov-2066", 29], ["Syawal", "Sabtu", "18-Dec-2066", 29], ["Zulkaidah", "Ahad", "16-Jan-2067", 30], ["Zulhijah", "Selasa", "15-Feb-2067", 29]],

00509 1490: ["Muharam", "Rabu", "16-Mar-2067", 30], ["Safar", "Jumat", "15-Apr-2067", 29], ["Rabiulawal", "Sabtu", "14-May-2067", 30], ["Rabiulakhir", "Senin", "13-Jun-2067", 30], ["Jumadilawal", "Rabu", "13-Jul-2067", 29], ["Jumadilakhir", "Kamis", "11-Aug-2067", 30], ["Rajab", "Sabtu", "10-Sep-2067", 30], ["Syakban", "Ahad", "10-Oct-2067", 29], ["Ramadan", "Selasa", "08-Nov-2067", 30], ["Syawal", "Kamis", "08-Dec-2067", 29], ["Zulkaidah", "Jumat", "06-Jan-2068", 29], ["Zulhijah", "Sabtu", "04-Feb-2068", 30]],

00510 1491: ["Muharam", "Senin", "05-Mar-2068", 29], ["Safar", "Selasa", "03-Apr-2068", 30], ["Rabiulawal", "Kamis", "03-May-2068", 29], ["Rabiulakhir", "Jumat", "01-Jun-2068", 30], ["Jumadilawal", "Ahad", "01-Jul-2068", 29], ["Jumadilakhir", "Senin", "30-Jul-2068", 30], ["Rajab", "Rabu", "29-Aug-2068", 30], ["Syakban", "Jumat", "28-Sep-2068", 29], ["Ramadan", "Sabtu", "27-Oct-2068", 30], ["Syawal", "Senin", "26-Nov-2068", 30], ["Zulkaidah", "Rabu", "26-Dec-2068", 29], ["Zulhijah", "Kamis", "24-Jan-2069", 29]],

00511 1492: ["Muharam", "Jumat", "22-Feb-2069", 30], ["Safar", "Ahad", "24-Mar-2069", 29], ["Rabiulawal", "Senin", "22-Apr-2069", 30], ["Rabiulakhir", "Rabu", "22-May-2069", 29], ["Jumadilawal", "Kamis", "20-Jun-2069", 29], ["Jumadilakhir", "Jumat", "19-Jul-2069", 30], ["Rajab", "Ahad", "18-Aug-2069", 30], ["Syakban", "Selasa", "17-Sep-2069", 29], ["Ramadan", "Rabu", "16-Oct-2069", 30], ["Syawal", "Jumat", "15-Nov-2069", 30], ["Zulkaidah", "Ahad", "15-Dec-2069", 29], ["Zulhijah", "Senin", "13-Jan-2070", 30]],

00512 1493: ["Muharam", "Rabu", "12-Feb-2070", 29], ["Safar", "Jumat", "14-Mar-2070", 29], ["Rabiulawal", "Sabtu", "12-Apr-2070", 29], ["Rabiulakhir", "Ahad", "11-May-2070", 30], ["Jumadilawal", "Selasa", "10-Jun-2070", 29], ["Jumadilakhir", "Rabu", "09-Jul-2070", 29], ["Rajab", "Kamis", "07-Aug-2070", 30], ["Syakban", "Sabtu", "06-Sep-2070", 29], ["Ramadan", "Ahad", "05-Oct-2070", 30], ["Syawal", "Selasa", "04-Nov-2070", 30], ["Zulkaidah", "Kamis", "04-Dec-2070", 29], ["Zulhijah", "Jumat", "02-Jan-2071", 30]],

00513 1494: ["Muharam", "Ahad", "01-Feb-2071", 30], ["Safar", "Selasa", "03-Mar-2071", 30], ["Rabiulawal", "Kamis", "02-Apr-2071", 29], ["Rabiulakhir", "Jumat", "01-May-2071", 29], ["Jumadilawal", "Sabtu", "30-May-2071", 30], ["Jumadilakhir", "Senin", "29-Jun-2071", 29], ["Rajab", "Selasa", "28-Jul-2071", 29], ["Syakban", "Rabu", "26-Aug-2071", 30], ["Ramadan", "Jumat", "25-Sep-2071", 29], ["Syawal", "Sabtu", "24-Oct-2071", 30], ["Zulkaidah", "Senin", "23-Nov-2071", 29], ["Zulhijah", "Selasa", "22-Dec-2071", 30]],

00514 1495: ["Muharam", "Kamis", "21-Jan-2072", 30], ["Safar", "Sabtu", "20-Feb-2072", 30], ["Rabiulawal", "Senin", "21-Mar-2072", 29], ["Rabiulakhir", "Selasa", "19-Apr-2072", 30], ["Jumadilawal", "Kamis", "19-May-2072", 29], ["Jumadilakhir", "Jumat", "17-Jun-2072", 30], ["Rajab", "Ahad", "17-Jul-2072", 29], ["Syakban", "Senin", "15-Aug-2072", 29], ["Ramadan", "Selasa", "13-Sep-2072", 30], ["Syawal", "Rabu", "13-Oct-2072", 29], ["Zulkaidah", "Jumat", "11-Nov-2072", 30], ["Zulhijah", "Ahad", "11-Dec-2072", 29]],

00515 1496: ["Muharam", "Senin", "09-Jan-2073", 30], ["Safar", "Rabu", "08-Feb-2073", 30], ["Rabiulawal", "Jumat", "10-Mar-2073", 30], ["Rabiulakhir", "Ahad", "09-Apr-2073", 29], ["Jumadilawal", "Senin", "08-May-2073", 30], ["Jumadilakhir", "Rabu", "07-Jun-2073", 29], ["Rajab", "Kamis", "06-Jul-2073", 30], ["Syakban", "Sabtu", "05-Aug-2073", 29], ["Ramadan", "Ahad", "03-Sep-2073", 29], ["Syawal", "Senin", "02-Oct-2073", 30], ["Zulkaidah", "Rabu", "01-Nov-2073", 29], ["Zulhijah", "Kamis", "30-Nov-2073", 30]],

00516 1497: ["Muharam", "Sabtu", "30-Dec-2073", 29], ["Safar", "Ahad", "28-Jan-2074", 30], ["Rabiulawal", "Selasa", "27-Feb-2074", 30], ["Rabiulakhir", "Kamis", "29-Mar-2074", 29], ["Jumadilawal", "Jumat", "27-Apr-2074", 30], ["Jumadilakhir", "Ahad", "27-May-2074", 30], ["Rajab", "Selasa", "26-Jun-2074", 29], ["Syakban", "Rabu", "25-Jul-2074", 30], ["Ramadan", "Jumat", "24-Aug-2074", 29], ["Syawal", "Sabtu", "22-Sep-2074", 29], ["Zulkaidah", "Ahad", "21-Oct-2074", 30], ["Zulhijah", "Selasa", "20-Nov-2074", 30]],

```
00517 | 1498: [  
00518 | [  
00519 |     "Muharam",  
00520 |     "Selasa",  
00521 |     "19-Dec-2074",  
00522 |     30  
00523 | ],  
00524 | [  
00525 |     "Safar",  
00526 |     "Kamis",  
00527 |     "18-Jan-2075",  
00528 |     29  
00529 | ],  
00530 | [  
00531 |     "Rabiulawal",  
00532 |     "Jumat",  
00533 |     "16-Feb-2075",  
00534 |     30  
00535 | ],  
00536 | [  
00537 |     "Rabiulakhir",  
00538 |     "Ahad",  
00539 |     "18-Mar-2075",  
00540 |     29  
00541 | ],  
00542 | [  
00543 |     "Jumadilawal",  
00544 |     "Senin",  
00545 |     "16-Apr-2075",  
00546 |     30  
00547 | ],  
00548 | [  
00549 |     "Jumadilakhir",  
00550 |     "Rabu",  
00551 |     "16-May-2075",  
00552 |     29  
00553 | ],  
00554 | [  
00555 |     "Rajab",  
00556 |     "Kamis",  
00557 |     "14-Jun-2075",  
00558 |     30  
00559 | ],  
00560 | [  
00561 |     "Syakban",  
00562 |     "Sabtu",  
00563 |     "14-Jul-2075",  
00564 |     30  
00565 | ],  
00566 | [  
00567 |     "Ramadan",  
00568 |     "Senin",  
00569 |     "13-Aug-2075",  
00570 |     29  
00571 | ],  
00572 | [  
00573 |     "Syawal",  
00574 |     "Selasa",  
00575 |     "11-Sep-2075",  
00576 |     30  
00577 | ],  
00578 | [  
00579 |     "Zulkaidah",  
00580 |     "Kamis",  
00581 |     "11-Oct-2075",  
00582 |     29  
00583 | ],  
00584 | [  
00585 |     "Zulhijah",  
00586 |     "Jumat",  
00587 |     "09-Nov-2075",
```

```
00588 | 30
00589 | ]
00590 | ],
00591 | "1499": [
00592 | [
00593 |     "Muharam",
00594 |     "Ahad",
00595 |     "09-Dec-2075",
00596 |     29
00597 | ],
00598 | [
00599 |     "Safar",
00600 |     "Senin",
00601 |     "07-Jan-2076",
00602 |     30
00603 | ],
00604 | [
00605 |     "Rabiulawal",
00606 |     "Rabu",
00607 |     "06-Feb-2076",
00608 |     29
00609 | ],
00610 | [
00611 |     "Rabiulakhir",
00612 |     "Kamis",
00613 |     "06-Mar-2076",
00614 |     30
00615 | ],
00616 | [
00617 |     "Jumadilawal",
00618 |     "Sabtu",
00619 |     "05-Apr-2076",
00620 |     29
00621 | ],
00622 | [
00623 |     "Jumadilakhir",
00624 |     "Ahad",
00625 |     "04-May-2076",
00626 |     30
00627 | ],
00628 | [
00629 |     "Rajab",
00630 |     "Selasa",
00631 |     "03-Jun-2076",
00632 |     29
00633 | ],
00634 | [
00635 |     "Syakban",
00636 |     "Rabu",
00637 |     "02-Jul-2076",
00638 |     30
00639 | ],
00640 | [
00641 |     "Ramadan",
00642 |     "Jumat",
00643 |     "01-Aug-2076",
00644 |     29
00645 | ],
00646 | [
00647 |     "Syawal",
00648 |     "Sabtu",
00649 |     "30-Aug-2076",
00650 |     30
00651 | ],
00652 | [
00653 |     "Zulkaidah",
00654 |     "Senin",
00655 |     "29-Sep-2076",
00656 |     30
00657 | ],
00658 | [
```

```

00659 |         "Zulhijah",
00660 |         "Rabu",
00661 |         "29-Oct-2076",
00662 |         29
00663 |     ]
00664 | ],
00665 | "1500": [
00666 |     [
00667 |         "Muharam",
00668 |         "Kamis",
00669 |         "27-Nov-2076",
00670 |         30
00671 |     ],
00672 |     [
00673 |         "Safar",
00674 |         "Sabtu",
00675 |         "27-Dec-2076",
00676 |         29
00677 |     ],
00678 |     [
00679 |         "Rabiulawal",
00680 |         "Ahad",
00681 |         "25-Jan-2077",
00682 |         30
00683 |     ],
00684 |     [
00685 |         "Rabiulakhir",
00686 |         "Selasa",
00687 |         "24-Feb-2077",
00688 |         29
00689 |     ],
00690 |     [
00691 |         "Jumadilawal",
00692 |         "Rabu",
00693 |         "25-Mar-2077",
00694 |         30
00695 |     ],
00696 |     [
00697 |         "Jumadilakhir",
00698 |         "Jumat",
00699 |         "24-Apr-2077",
00700 |         29
00701 |     ],
00702 |     [
00703 |         "Rajab",
00704 |         "Sabtu",
00705 |         "23-May-2077",
00706 |         29
00707 |     ],
00708 |     [
00709 |         "Syakban",
00710 |         "Ahad",
00711 |         "21-Jun-2077",
00712 |         30
00713 |     ],
00714 |     [
00715 |         "Ramadan",
00716 |         "Selasa",
00717 |         "21-Jul-2077",
00718 |         29
00719 |     ],
00720 |     [
00721 |         "Syawal",
00722 |         "Rabu",
00723 |         "19-Aug-2077",
00724 |         30
00725 |     ],
00726 |     [
00727 |         "Zulkaidah",
00728 |         "Jumat",
00729 |         "18-Sep-2077",

```

```

00730 |         30
00731 |     ],
00732 |     [
00733 |         "Zulhijah",
00734 |         "Ahad",
00735 |         "18-Oct-2077",
00736 |         30
00737 |     ]
00738 | ]
00739 |
00740 | })
00741 |
00742 | # ---> TAMBAHAN: AUTO-LOAD DARI DATABASE JSON <---
00743 | import json
00744 | try:
00745 |     if getattr(sys, 'frozen', False):
00746 |         DB_JSON_PATH = os.path.join(os.path.dirname(sys.executable), "db_hijriah.json")
00747 |     else:
00748 |         DB_JSON_PATH = os.path.join(os.path.dirname(os.path.abspath(__file__)), "db_hijriah.json")
00749 |
00750 |     if os.path.exists(DB_JSON_PATH):
00751 |         with open(DB_JSON_PATH, "r", encoding="utf-8") as f:
00752 |             external_db = json.load(f)
00753 |
00754 |             # Konversi key JSON dari string menjadi integer
00755 |             external_db = {int(k): v for k, v in external_db.items()}
00756 |
00757 |             # Jangan menimpa data bawaan yang sudah tervalidasi
00758 |             for year, months in external_db.items():
00759 |                 if year not in HIJRI_DB:
00760 |                     HIJRI_DB[year] = months
00761 | except Exception as e:
00762 |     print(f"Peringatan: Gagal memuat db_hijriah.json -> {e}")
00763 | # -----
00764 |
00765 | class HijriConverter:
00766 |     @staticmethod
00767 |     def parse_date(date_str):
00768 |         try:
00769 |             m = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
00770 |             parts = date_str.split('-')
00771 |             return datetime.date(int(parts[2]), m.get(parts[1], 1), int(parts[0]))
00772 |         except: return None
00773 |     @staticmethod
00774 |     def get_hijri_date(today):
00775 |         for year, months in HIJRI_DB.items():
00776 |             for m_data in months:
00777 |                 start = HijriConverter.parse_date(m_data[2])
00778 |                 if start and start <= today < start + datetime.timedelta(days=m_data[3]):
00779 |                     return f"({today - start).days + 1} {m_data[0]} {year} H"
00780 |             return "N/A"
00781 |
00782 | # =====
00783 | # DATABASE KOTA (Diambil dari master Anda)
00784 | # =====
00785 | CITY_DB = {
00786 |     "Aceh": {"Banda Aceh": (5.5483, 95.3238), "Lhokseumawe": (5.1801, 97.1507), "Langsa": (4.4725, 97.9658), "Meulaboh": (4.1438, 96.1265), "Sabang": (5.8942, 95.3184), "Subulussalam": (2.6312, 98.0012),
    "Takengon": (4.6212, 96.8412), "Kutacane": (3.4812, 97.8112), "Singkil": (2.2851, 97.7942), "Calang": (4.6312, 95.5812), "Blangpidie": (3.7412, 96.8412), "Karang Baru": (4.3112, 98.0512), "Blangkejeren":
    (3.9912, 97.3312), "Sigli": (5.3812, 95.9612), "Meureudu": (5.2412, 96.2512), "Bireuen": (5.2031, 96.7011), "Simpang Tiga Redelong": (4.7212, 96.8512), "Suka Makmue": (4.1512, 96.3312), "Sinabang":
    (2.4812, 96.3712)},
00787 |     "Bali": {"Denpasar": (-8.6705, 115.2126), "Singaraja": (-8.1120, 115.0882), "Mangupura": (-8.5812, 115.1712), "Gianyar": (-8.5412, 115.3212), "Tabanan": (-8.5312, 115.1212), "Semarapura": (-8.5312,
    115.4012), "Amlapura": (-8.4412, 115.6112), "Bangli": (-8.4512, 115.3512), "Negara": (-8.3512, 114.6212)},
00788 |     "Bangka Belitung": {"Pangkal Pinang": (-2.1300, 106.1100), "Sungailiat": (-1.8512, 106.1112), "Tanjung Pandan": (-2.7312, 107.6312)},
00789 |     "Banten": {"Serang": (-6.1104, 106.1601), "Cilegon": (-6.0234, 106.0152), "Tangerang": (-6.1702, 106.6403), "Tangerang Selatan": (-6.2886, 106.7179), "Tigaraksa": (-6.2712, 106.4712), "Pandeglang":
    (-6.3112, 106.1012), "Rangkasbitung": (-6.3512, 106.2412)},
00790 |     "Bengkulu": {"Bengkulu": (-3.8004, 102.2655), "Curup": (-3.4612, 102.5212), "Manna": (-4.4712, 102.9012), "Argamakmur": (-3.4212, 102.1912), "Mukomuko": (-2.5812, 101.1212)},
00791 |     "DI Yogyakarta": {"Yogyakarta": (-7.7956, 110.3695), "Sleman": (-7.7212, 110.3644), "Bantul": (-7.8922, 110.3322), "Wates": (-7.8512, 110.1512), "Wonosari": (-7.9612, 110.6012)},
00792 |     "DKI Jakarta": {"Jakarta Pusat": (-6.1865, 106.8270), "Jakarta Selatan": (-6.2615, 106.8106), "Jakarta Barat": (-6.1674, 106.7637), "Jakarta Timur": (-6.2250, 106.9004), "Jakarta Utara": (-6.1214,
    106.8745), "Kepulauan Seribu": (-5.7412, 106.6112)},
00793 |     "Gorontalo": {"Gorontalo": (0.5435, 123.0568), "Limboto": (0.6212, 122.9812), "Marisa": (0.4512, 121.9412)},
00794 |     "Jambi": {"Jambi": (-1.6099, 103.6057), "Sungai Penuh": (-2.0722, 101.3789), "Muara Bulian": (-1.7212, 103.2712), "Bangko": (-2.0712, 102.2612), "Muara Bungo": (-1.4812, 102.1212)},

```



```
00795 | "Jawa Barat": {"Bandung": (-6.9175, 107.6191), "Bekasi": (-6.2383, 106.9756), "Depok": (-6.4025, 106.7942), "Bogor": (-6.5971, 106.7986), "Cimahi": (-6.8722, 107.5422), "Tasikmalaya": (-7.3274,
108.2207), "Garut": (-7.2272, 107.9086), "Ciamis": (-7.3211, 108.3512), "Banjar": (-7.3676, 108.5364), "Cirebon": (-6.7063, 108.5570), "Indramayu": (-6.3271, 108.3201), "Majalengka": (-6.8312, 108.2212),
"Kuningan": (-6.9712, 108.4812), "Sukabumi": (-6.9242, 106.9350), "Cianjur": (-6.8201, 107.1394), "Karawang": (-6.3012, 107.3011), "Subang": (-6.5712, 107.7612), "Purwakarta": (-6.5512, 107.4411),
"Sumedang": (-6.8412, 107.9211)},
00796 | "Jawa Tengah": {"Semarang": (-7.0667, 110.4100), "Demak": (-6.8906, 110.6389), "Kudus": (-6.8048, 110.8400), "Pati": (-6.7509, 111.0384), "Rembang": (-6.7058, 111.3414), "Jepara": (-6.5888, 110.6675),
"Blora": (-6.9697, 111.4184), "Grobogan": (-7.0264, 110.9187), "Surakarta": (-7.5703, 110.8297), "Sukoharjo": (-7.6833, 110.8333), "Wonogiri": (-7.8122, 110.9253), "Karanganyar": (-7.5961, 110.9511),
"Sragen": (-7.4267, 111.0211), "Boyolali": (-7.5317, 110.5961), "Klaten": (-7.7031, 110.6025), "Magelang": (-7.4706, 110.2178), "Temanggung": (-7.3167, 110.1667), "Wonosobo": (-7.3597, 109.9111),
"Purworejo": (-7.7122, 110.0078), "Kebumen": (-7.6697, 109.6522), "Purwokerto": (-7.4244, 109.2300), "Cilacap": (-7.7277, 109.0159), "Purbalingga": (-7.3875, 109.3675), "Banjarnegara": (-7.3961, 109.6967),
"Pekalongan": (-6.8886, 109.6753), "Batang": (-6.9111, 109.7289), "Pemalang": (-6.8919, 109.3814), "Tegal": (-6.8676, 109.1371), "Brebès": (-6.8722, 109.0436), "Kendal": (-6.9189, 110.2039), "Salatiga":
(-7.3305, 110.5084)},
00797 | "Jawa Timur": {"Surabaya": (-7.2575, 112.7521), "Sidoarjo": (-7.4412, 112.7112), "Gresik": (-7.1612, 112.6511), "Malang": (-7.9797, 112.6304), "Batu": (-7.8671, 112.5239), "Kediri": (-7.8172,
112.0116), "Blitar": (-8.0983, 112.1681), "Tulungagung": (-8.0612, 111.9012), "Nganjuk": (-7.6012, 111.9012), "Trenggalek": (-8.0512, 111.7112), "Madiun": (-7.6298, 111.5239), "Ngawi": (-7.4012, 111.4411),
"Magetan": (-7.6512, 111.3312), "Ponorogo": (-7.8712, 111.4612), "Pacitan": (-8.2012, 111.0912), "Jember": (-8.1712, 113.7011), "Banyuwangi": (-8.2112, 114.3611), "Bondowoso": (-7.9112, 113.8212),
"Situbondo": (-7.7012, 114.0012), "Probolinggo": (-7.7554, 113.2162), "Lumajang": (-8.1312, 113.2212), "Bojonegoro": (-7.1512, 111.8811), "Tuban": (-6.9012, 112.0511), "Lamongan": (-7.1212, 112.4111),
"Bangkalan": (-7.0412, 112.7411), "Sampang": (-7.1812, 113.2411), "Pamekasan": (-7.1512, 113.4711), "Sumenep": (-7.0112, 113.8611), "Pasuruan": (-7.6449, 112.9061), "Mojokerto": (-7.4712, 112.4311),
"Jombang": (-7.5512, 112.2312)},
00798 | "Kalimantan Barat": {"Pontianak": (-0.0263, 109.3425), "Singkawang": (0.9023, 108.9711), "Mempawah": (0.3512, 108.9612), "Sambas": (1.3512, 109.3012), "Sintang": (0.0712, 111.4912), "Ketapang":
(-1.8412, 109.9712)},
00799 | "Kalimantan Selatan": {"Banjarasin": (-3.3167, 114.5900), "Banjarbaru": (-3.4431, 114.8286), "Pelaihari": (-3.7912, 114.7712), "Kotabaru": (-3.2412, 116.2212), "Tanjung": (-2.1812, 115.3912)},
00800 | "Kalimantan Tengah": {"Palangkaraya": (-2.2107, 113.9213), "Sampit": (-2.5312, 112.9512), "Pangkalan Bun": (-2.6812, 111.6212), "Muara Teweh": (-0.9512, 114.8912), "Kapuas": (-3.0112, 114.3912)},
00801 | "Kalimantan Timur": {"Samarinda": (-0.4949, 117.1492), "Balikpapan": (-1.2654, 116.8312), "Bontang": (0.1328, 117.4728), "Tenggarong": (-0.4312, 116.9812), "Sangatta": (0.5012, 117.5512)},
00802 | "Kalimantan Utara": {"Tanjung Selor": (2.8333, 117.3667), "Tarakan": (3.3033, 117.5878), "Nunukan": (4.1312, 117.6512), "Malinau": (3.5812, 116.6312)},
00803 | "Kepulauan Riau": {"Tanjung Pinang": (0.9167, 104.4500), "Batam": (1.1301, 104.0520), "Karimun": (0.9912, 103.4212), "Natuna": (3.9412, 108.3812)},
00804 | "Lampung": {"Bandar Lampung": (-5.4292, 105.2611), "Metro": (-5.1133, 105.3067), "Kalianda": (-5.7412, 105.5912), "Kotabumi": (-4.8212, 104.8912)},
00805 | "Maluku": {"Ambon": (-3.6954, 128.1814), "Tual": (-5.6322, 132.7389), "Masohi": (-3.3012, 128.9512), "Langgur": (-5.6512, 132.7112), "Saumlaki": (-7.9512, 131.3012)},
00806 | "Maluku Utara": {"Ternate": (0.7901, 127.3821), "Tidore": (0.6457, 127.4422), "Sofifi": (0.7312, 127.5812), "Tobelo": (1.7212, 128.0112)},
00807 | "NTB": {"Mataram": (-8.5833, 116.1167), "Praya": (-8.7112, 116.2712), "Selong": (-8.6512, 116.5312), "Bima": (-8.4605, 118.7265), "Sumbawa Besar": (-8.5012, 117.4212), "Dompu": (-8.5312, 118.4612)},
00808 | "NTT": {"Kupang": (-10.1772, 123.6070), "Soe": (-9.8612, 124.2812), "Kefamenanu": (-9.4512, 124.4812), "Waingapu": (-9.6512, 120.2612), "Labuan Bajo": (-8.5012, 119.8812), "Ruteng": (-8.6112,
120.4712), "Ende": (-8.8412, 121.6512), "Maumere": (-8.6254, 122.2132), "Larantuka": (-8.3412, 122.9812)},
00809 | "Papua": {"Jayapura": (-2.5916, 140.6690), "Biak": (-1.1712, 136.0812), "Serui": (-1.8812, 136.2312)},
00810 | "Papua Barat": {"Manokwari": (-0.8614, 134.0620), "Fakfak": (-3.2212, 132.2912), "Kaimana": (-3.6612, 133.7712)},
00811 | "Papua Barat Daya": {"Sorong": (-0.8765, 131.2558), "Waisai": (-0.4112, 130.8112)},
00812 | "Papua Selatan": {"Merauke": (-8.4912, 140.4012), "Agats": (-5.5412, 138.1312)},
00813 | "Papua Tengah": {"Nabire": (-3.3612, 135.5012), "Timika": (-4.5412, 136.8812)},
00814 | "Papua Pegunungan": {"Wamena": (-4.0612, 138.9412)},
00815 | "Riau": {"Pekanbaru": (0.5071, 101.4478), "Dumai": (1.6712, 101.4455), "Bangkinang": (0.3312, 101.0312), "Siak": (0.7912, 102.0412)},
00816 | "Sulawesi Barat": {"Mamuju": (-2.6712, 118.8812), "Majene": (-3.5412, 118.9712), "Polewali": (-3.4312, 119.3212)},
00817 | "Sulawesi Selatan": {"Makassar": (-5.1476, 119.4327), "Parepare": (-4.0135, 119.6247), "Palopo": (-2.9926, 120.1916), "Gowa": (-5.2012, 119.4512), "Bone": (-4.5412, 120.3212)},
00818 | "Sulawesi Tengah": {"Palu": (-0.8917, 119.8707), "Luwuk": (-0.9512, 122.7812), "Poso": (-1.3912, 120.7512), "Donggala": (-0.6712, 119.7412)},
00819 | "Sulawesi Tenggara": {"Kendari": (-3.9722, 122.5149), "Baubau": (-5.4612, 122.6112), "Kolaka": (-4.0512, 121.5912)},
00820 | "Sulawesi Utara": {"Manado": (1.4748, 124.8404), "Bitung": (1.4412, 125.1212), "Tomohon": (1.3212, 124.8312), "Kotamobagu": (0.7212, 124.3112)},
00821 | "Sumatera Barat": {"Padang": (-0.9471, 100.4172), "Bukittinggi": (-0.3041, 100.3695), "Payakumbuh": (-0.2241, 100.6358), "Pariaman": (-0.6171, 100.1239), "Solok": (-0.7937, 100.6599)},
00822 | "Sumatera Selatan": {"Palembang": (-2.9761, 104.7754), "Lubuklinggau": (-3.2952, 102.8611), "Prabumulih": (-3.4308, 104.2255), "Lahat": (-3.7812, 103.5412), "Pagar Alam": (-4.0112, 103.2712)},
00823 | "Sumatera Utara": {"Medan": (3.5952, 98.6722), "Binjai": (3.6010, 98.4854), "Pematangsiantar": (2.9620, 99.0664), "Tebing Tinggi": (3.3364, 99.1625), "Sibolga": (1.7392, 98.7776)},
00824 |
00825 | "Amerika Serikat": {
00826 |     "New York City": (40.7128, -74.0060),
00827 |     "Los Angeles": (34.0522, -118.2437),
00828 |     "Chicago": (41.8781, -87.6298),
00829 |     "Houston": (29.7604, -95.3698),
00830 |     "Phoenix": (33.4484, -112.0740),
00831 |     "Seattle": (47.6062, -122.3321),
00832 |     "Miami": (25.7617, -80.1918),
00833 |     "Washington D.C.": (38.9072, -77.0369),
00834 |     "Anchorage": (61.2181, -149.9003),
00835 |     "Fairbanks": (64.8378, -147.7164),
00836 |     "King Salmon": (58.6833, -156.6667),
00837 |     "Port Heiden": (56.937449, -158.611570),
00838 |     "Aleutians East Borough-1": (56.807956, -158.944580),
00839 |     "Aleutians East Borough-2": (56.013387, -160.413219),
00840 |     "Cold Bay-1": (55.2000, -162.7167),
00841 |     "Cold Bay-2": (55.194384, -162.728017),
00842 |     "Homer": (59.6425, -151.5483),
00843 |     "Valdez": (61.1308, -146.3483)
00844 | },
00845 | "Kanada": {
00846 |     "Toronto": (43.6510, -79.3470),
00847 |     "Vancouver": (49.2827, -123.1207),
00848 |     "Montreal": (45.5017, -73.5673)
00849 | },
00850 | "Meksiko": {
```



```
00851 |         "Mexico City": (19.4326, -99.1332),
00852 |         "Monterrey": (25.6866, -100.3161),
00853 |         "Guadalajara": (20.6597, -103.3496)
00854 |     },
00855 |     # MENGGABUNGKAN SELURUH AMERIKA SELATAN (Brasil, Argentina, Kolombia, Peru, Chili, dsb)
00856 |     "Amerika Selatan": {
00857 |         "Lima": (-12.0464, -77.0428),
00858 |         "Santiago": (-33.4489, -70.6693),
00859 |         "Bogota": (4.7110, -74.0721),
00860 |         "Quito": (-0.1807, -78.4678),
00861 |         "La Paz": (-16.4897, -68.1193),
00862 |         "Guayaquil": (-2.1894, -79.8891),
00863 |         "Valparaiso": (-33.0472, -71.6127),
00864 |         "Arequipa": (-16.4090, -71.5375),
00865 |         "Cali": (3.4516, -76.5320),
00866 |         "Antofagasta": (-23.6500, -70.4000),
00867 |         "Sao Paulo": (-23.5505, -46.6333),
00868 |         "Rio de Janeiro": (-22.9068, -43.1729),
00869 |         "Buenos Aires": (-34.6037, -58.3816),
00870 |         "Montevideo": (-34.9011, -56.1645),
00871 |         "Brasilia": (-15.8267, -47.9218),
00872 |         "Salvador": (-12.9714, -38.5014),
00873 |         "Fortaleza": (-3.7319, -38.5267),
00874 |         "Rosario": (-32.9468, -60.6393),
00875 |         "Asuncion": (-25.2637, -57.5759),
00876 |         "Caracas": (10.4806, -66.9036),
00877 |         "Medellin": (6.2442, -75.5812),
00878 |         "Cartagena": (10.3910, -75.4794),
00879 |         "Cuenca": (-2.9001, -79.0059),
00880 |         "Manta": (-0.9621, -80.7127),
00881 |         "Trujillo": (-8.1159, -79.0282),
00882 |         "Piura": (-5.1945, -80.6328),
00883 |         "Cusco": (-13.5226, -71.9673),
00884 |         "Arica": (-18.4783, -70.3126),
00885 |         "Iquique": (-20.2208, -70.1431),
00886 |         "Concepcion": (-36.8201, -73.0444),
00887 |         "Temuco": (-38.7359, -72.5904),
00888 |         "Manaus": (-3.1190, -60.0217),
00889 |         "Cuiaba": (-15.6014, -56.0979),
00890 |         "Iquitos": (-3.7491, -73.2538),
00891 |         "Santa Cruz de la Sierra": (-17.7833, -63.1821),
00892 |         "Cochabamba": (-17.3895, -66.1568),
00893 |         "Sucre": (-19.0333, -65.2627),
00894 |         "Potosi": (-19.5836, -65.7531),
00895 |         "Cordoba": (-31.4201, -64.1888),
00896 |         "Mendoza": (-32.8908, -68.8272)
00897 |     },
00898 |     "Arab Saudi": {
00899 |         "Tumair": (25.7039, 45.8678), "Majmaah": (25.8970, 45.3372), "Shaqra": (25.2524, 45.2536),
00900 |         "Makkah": (21.4225, 39.8262), "Madinah": (24.4672, 39.6112), "Riyadh": (24.7136, 46.6753),
00901 |         "Jeddah": (21.4858, 39.1925), "Taif": (21.2703, 40.4158), "Tabuk": (28.3835, 36.5662),
00902 |         "Dammam": (26.4207, 50.0888), "Abha": (18.2164, 42.5053), "Jazan": (16.8892, 42.5511),
00903 |         "Buraidah": (26.3260, 43.9750), "Hail": (27.5219, 41.6907), "Najran": (17.4933, 44.1277),
00904 |         "Al-Baha": (20.0129, 41.4677), "Sakaka": (29.9697, 40.2064), "Arar": (30.9833, 40.0333),
00905 |         "Yanbu": (24.0891, 38.0637), "Al Ahsa": (25.3795, 49.5858)
00906 |     },
00907 |     # MENGGABUNGKAN KUNCI "Mesir" YANG SEBELUMNYA DITULIS 2 KALI
00908 |     "Mesir": {
00909 |         "Kairo": (30.0444, 31.2357),
00910 |         "Iskandariyah / Alexandria": (31.2001, 29.9187),
00911 |         "Aswan": (24.0889, 32.8998),
00912 |         "Sallum": (31.5658, 25.1481),
00913 |         "Abu Simbel": (22.3372, 31.6258),
00914 |         "Kharga": (25.4390, 30.5586),
00915 |         "Sohag": (26.5591, 31.6957),
00916 |         "Faiyum": (29.3084, 30.8428),
00917 |         "Qena": (26.1551, 32.7160),
00918 |         "Sharm El-Sheikh": (27.9158, 34.3299),
00919 |         "Marsa Matruh": (31.3525, 27.2453)
00920 |     },
00921 |     # MENGGABUNGKAN KUNCI "Maroko" YANG SEBELUMNYA DITULIS 2 KALI
```

```
00922 | "Maroko": {
00923 |     "Rabat": (34.0209, -6.8416),
00924 |     "Casablanca": (33.5731, -7.5898),
00925 |     "Marrakech": (31.6295, -7.9811),
00926 |     "Fez": (34.0331, -5.0003),
00927 |     "Dakhla": (23.6848, -15.9580),
00928 |     "Laayoune": (27.1253, -13.1625),
00929 |     "Agadir": (30.4278, -9.5981),
00930 |     "Tangier": (35.7595, -5.8340),
00931 |     "Oujda": (34.6814, -1.9086),
00932 |     "Ouarzazate": (30.9189, -6.8934)
00933 | },
00934 | "Kepulauan Pasifik (Timur Jauh)": {
00935 |     "Kiritimati": (1.8709, -157.3962),
00936 |     "Apia": (-13.8333, -171.7667),
00937 |     "Nuku'alofa": (-21.1393, -175.2018),
00938 |     "Suva": (-18.1248, 178.4501)
00939 | },
00940 | # MENAMBAHKAN GISBORNE (NEW ZEALAND)
00941 | "Selandia Baru": {
00942 |     "Gisborne": (-38.6623, 178.0176),
00943 |     "Waitangi": (-35.2681, 174.0803),
00944 |     "Auckland": (-36.8485, 174.7633),
00945 |     "Wellington": (-41.2865, 174.7762),
00946 |     "Christchurch": (-43.5321, 172.6362),
00947 |     "Dunedin": (-45.8788, 170.5028)
00948 | },
00949 | "Australia": {
00950 |     "Sydney": (-33.8688, 151.2093),
00951 |     "Brisbane": (-27.4698, 153.0251),
00952 |     "Melbourne": (-37.8136, 144.9631),
00953 |     "Hobart": (-42.8821, 147.3272),
00954 |     "Adelaide": (-34.9285, 138.6007),
00955 |     "Alice Springs": (-23.6980, 133.8807),
00956 |     "Darwin": (-12.4634, 130.8456),
00957 |     "Perth": (-31.9505, 115.8605)
00958 | },
00959 | "Jepang": {
00960 |     "Tokyo": (35.6762, 139.6503),
00961 |     "Osaka": (34.6937, 135.5023),
00962 |     "Sapporo": (43.0618, 141.3545),
00963 |     "Naha": (26.2124, 127.6809)
00964 | },
00965 | "Turki": {
00966 |     "Istanbul": (41.0082, 28.9784),
00967 |     "Ankara": (39.9334, 32.8597),
00968 |     "Izmir": (38.4192, 27.1287),
00969 |     "Konya": (37.8746, 32.4932),
00970 |     "Erzurum": (39.9043, 41.2679),
00971 |     "Antalya": (36.8969, 30.7133)
00972 | },
00973 | "Inggris Raya": {
00974 |     # London dan sekitarnya
00975 |     "London": (51.5074, -0.1278),
00976 |     "Greenwich": (51.4826, -0.0077),
00977 |
00978 |     # Inggris
00979 |     "Birmingham": (52.4862, -1.8904),
00980 |     "Manchester": (53.4808, -2.2426),
00981 |     "Liverpool": (53.4084, -2.9916),
00982 |     "Leeds": (53.8008, -1.5491),
00983 |     "Sheffield": (53.3811, -1.4701),
00984 |     "Bristol": (51.4545, -2.5879),
00985 |     "Newcastle upon Tyne": (54.9783, -1.6178),
00986 |     "Nottingham": (52.9548, -1.1581),
00987 |     "Leicester": (52.6369, -1.1398),
00988 |     "Coventry": (52.4068, -1.5197),
00989 |     "Bradford": (53.7950, -1.7594),
00990 |     "Hull": (53.7676, -0.3274),
00991 |     "Stoke-on-Trent": (53.0027, -2.1794),
00992 |     "Wolverhampton": (52.5862, -2.1287),
```

```
00993 | "Derby": (52.9225, -1.4746),
00994 | "Southampton": (50.9097, -1.4044),
00995 | "Portsmouth": (50.8198, -1.0880),
00996 | "Plymouth": (50.3755, -4.1427),
00997 | "Brighton": (50.8225, -0.1372),
00998 | "Oxford": (51.7520, -1.2577),
00999 | "Cambridge": (52.2053, 0.1218),
01000 | "York": (53.9590, -1.0815),
01001 | "Exeter": (50.7184, -3.5339),
01002 | "Norwich": (52.6309, 1.2974),
01003 | "Reading": (51.4543, -0.9781),
01004 | "Milton Keynes": (52.0406, -0.7594),
01005 | "Peterborough": (52.5695, -0.2405),
01006 |
01007 | # Skotlandia
01008 | "Glasgow": (55.8642, -4.2518),
01009 | "Edinburgh": (55.9533, -3.1883),
01010 | "Aberdeen": (57.1497, -2.0943),
01011 | "Dundee": (56.4620, -2.9707),
01012 | "Inverness": (57.4778, -4.2247),
01013 | "Stirling": (56.1165, -3.9369),
01014 |
01015 | # Wales
01016 | "Cardiff": (51.4816, -3.1791),
01017 | "Swansea": (51.6214, -3.9436),
01018 | "Newport": (51.5842, -2.9977),
01019 | "Wrexham": (53.0430, -2.9916),
01020 |
01021 | # Irlandia Utara
01022 | "Belfast": (54.5973, -5.9301),
01023 | "Derry": (54.9966, -7.3086),
01024 | "Lisburn": (54.5162, -6.0580),
01025 | "Newry": (54.1751, -6.3402)
01026 | },
01027 | "Eropa Daratan": {
01028 | "Paris": (48.8566, 2.3522),
01029 | "Berlin": (52.5200, 13.4050),
01030 | "Roma": (41.9028, 12.4964),
01031 | "Madrid": (40.4168, -3.7038),
01032 | "Cordoba": (37.8882, -4.7794),
01033 | "Saratjevo": (43.8563, 18.4131)
01034 | },
01035 | "Afrika Selatan": {
01036 | "Cape Town": (-33.9249, 18.4241),
01037 | "Johannesburg": (-26.2041, 28.0473),
01038 | "Durban": (-29.8587, 31.0218),
01039 | "Pretoria": (-25.7479, 28.2293),
01040 | "Bloemfontein": (-29.1141, 26.2206),
01041 | "Port Elizabeth": (-33.9608, 25.6022)
01042 | },
01043 | "Rusia": {
01044 | "Makhachkala": (42.9831, 47.5046), "Grozny": (43.3194, 45.6949), "Sochi": (43.5853, 39.7203),
01045 | "Astrakhan": (46.3497, 48.0408), "Rostov-na-Donu": (47.2357, 39.7015), "Volgograd": (48.7000, 44.5167),
01046 | "Kazan": (55.7963, 49.1088), "Ufa": (54.7388, 55.9721), "Samara": (53.2415, 50.2212),
01047 | "Nizhny Novgorod": (56.3269, 44.0059), "Moskow": (55.7558, 37.6173), "Sankt-Peterburg": (59.9343, 30.3351),
01048 | "Murmansk": (68.9585, 33.0827), "Arkhangelsk": (64.5399, 40.5167), "Yekaterinburg": (56.8389, 60.6057),
01049 | "Chelyabinsk": (55.1644, 61.4368), "Omsk": (54.9885, 73.3242), "Novosibirsk": (55.0084, 82.9357),
01050 | "Krasnoyarsk": (56.0153, 92.8932), "Irkutsk": (52.2978, 104.2964), "Khabarovsk": (48.4814, 135.0721),
01051 | "Vladivostok": (43.1198, 131.8869)
01052 | },
01053 | "Aljazair": {
01054 | "Aljir": (36.7538, 3.0588),
01055 | "Oran": (35.6987, -0.6308),
01056 | "Constantine": (36.3650, 6.6147),
01057 | "Tamanrasset": (22.7850, 5.5228),
01058 | "Tindouf": (27.6711, -8.1474)
01059 | },
01060 | "Tunisia": {
01061 | "Tunis": (36.8065, 10.1815),
01062 | "Sfax": (34.7406, 10.7603),
01063 | "Tozeur": (33.9197, 8.1335)
```

```
01064 | },
01065 | "Libya": {
01066 |   "Tripoli": (32.8872, 13.1913),
01067 |   "Benghazi": (32.1167, 20.0667),
01068 |   "Sebha": (27.0377, 14.4283)
01069 | },
01070 | "Afrika Barat dan Tengah": {
01071 |   "Lagos": (6.5244, 3.3792), "Dakar": (14.7167, -17.4677), "Accra": (5.5560, -0.1969),
01072 |   "Abidjan": (5.3600, -4.0083), "Bamako": (12.6392, -8.0029), "Kano": (12.0022, 8.5920),
01073 |   "Port Harcourt": (4.8156, 7.0498), "Conakry": (9.5092, -13.7122), "Freetown": (8.4657, -13.2317),
01074 |   "Monrovia": (6.3156, -10.8074), "Ouagadougou": (12.3714, -1.5197), "Niamey": (13.5116, 2.1254),
01075 |   "Kinshasa": (-4.4419, 15.2663), "Brazzaville": (-4.2634, 15.2429), "Yaounde": (3.8480, 11.5021),
01076 |   "Douala": (4.0511, 9.7679), "Libreville": (0.4162, 9.4673), "Bangui": (4.3947, 18.5582),
01077 |   "N'Djamena": (12.1348, 15.0557), "Lubumbashi": (-11.6609, 27.4794), "Malabo": (3.7504, 8.7860),
01078 |   "Pointe-Noire": (-4.7692, 11.8664)
01079 | },
01080 | "China": {
01081 |   "Beijing": (39.9042, 116.4074), "Tianjin": (39.0842, 117.2010), "Shijiazhuang": (38.0428, 114.5149),
01082 |   "Taiyuan": (37.8706, 112.5489), "Hohhot": (40.8423, 111.6708), "Shenyang": (41.8057, 123.4315),
01083 |   "Dalian": (38.9140, 121.6147), "Changchun": (43.8171, 125.3235), "Harbin": (45.8038, 126.5350),
01084 |   "Tangshan": (39.6309, 118.1802), "Baoding": (38.8739, 115.4648), "Handan": (36.6256, 114.5395),
01085 |   "Qinhuangdao": (39.9354, 119.5865), "Datong": (40.0768, 113.2914), "Baotou": (40.6522, 109.8222),
01086 |   "Anshan": (41.1078, 122.9945), "Jilin": (43.8378, 126.5494), "Qiqihar": (47.3543, 123.9182),
01087 |   "Chengde": (40.9515, 117.9338), "Zhangjiakou": (40.8244, 114.8797), "Shanghai": (31.2304, 121.4737),
01088 |   "Nanjing": (32.0603, 118.7969), "Hangzhou": (30.2741, 120.1551), "Jinan": (36.6512, 117.1201),
01089 |   "Qingdao": (36.0671, 120.3826), "Fuzhou": (26.0745, 119.2965), "Xiamen": (24.4798, 118.0894),
01090 |   "Hefei": (31.8206, 117.2272), "Nanchang": (28.6820, 115.8579), "Suzhou": (31.2989, 120.5853),
01091 |   "Wuxi": (31.4912, 120.3119), "Ningbo": (29.8683, 121.5439), "Wenzhou": (27.9938, 120.6994),
01092 |   "Quanzhou": (24.8739, 118.6758), "Xuzhou": (34.2646, 117.1859), "Yantai": (37.4638, 121.4479),
01093 |   "Weifang": (36.7068, 119.1618), "Jinhua": (29.0781, 119.6495), "Shaoxing": (30.0024, 120.5763),
01094 |   "Nantong": (32.0147, 120.8646), "Guangzhou": (23.1291, 113.2644), "Shenzhen": (22.5431, 114.0579),
01095 |   "Dongguan": (23.0205, 113.7518), "Foshan": (23.0215, 113.1214), "Nanning": (22.8146, 108.3200),
01096 |   "Haikou": (20.0174, 110.3212), "Sanya": (18.2528, 109.5119), "Zhuhai": (22.2707, 113.5767),
01097 |   "Zhongshan": (22.5176, 113.3928), "Shantou": (23.3541, 116.6820), "Zhanjiang": (21.2707, 110.3590),
01098 |   "Guilin": (25.2536, 110.2902), "Liuzhou": (24.3146, 109.4160), "Beihai": (21.4812, 109.1194),
01099 |   "Huizhou": (23.1118, 114.4162), "Jiangmen": (22.5787, 113.0816), "Zhaoqing": (23.0515, 112.4651),
01100 |   "Chaozhou": (23.6570, 116.6226), "Maoming": (21.6620, 110.9255), "Yangjiang": (21.8565, 111.9826),
01101 |   "Wuhan": (30.5928, 114.3055), "Zhengzhou": (34.7466, 113.6253), "Changsha": (28.2282, 112.9388),
01102 |   "Luoyang": (34.6198, 112.4539), "Yichang": (30.6919, 111.2865), "Xiangyang": (32.0084, 112.1224),
01103 |   "Jingzhou": (30.3349, 112.2397), "Yueyang": (29.3730, 113.1287), "Zhuzhou": (27.8274, 113.1330),
01104 |   "Xiangtan": (27.8296, 112.9332), "Hengyang": (26.8954, 112.5719), "Kaifeng": (34.7969, 114.3076),
01105 |   "Xinxiang": (35.3031, 113.9268), "Anyang": (36.0967, 114.3920), "Xuchang": (34.0253, 113.8223),
01106 |   "Pingdingshan": (33.7371, 113.3005), "Nanyang": (32.9908, 112.5283), "Xinyang": (32.1458, 114.0847),
01107 |   "Shangqiu": (34.4150, 115.6562), "Zhumadian": (32.9773, 114.0254), "Chengdu": (30.5728, 104.0668),
01108 |   "Chongqing": (29.5630, 106.5515), "Xi'an": (34.3416, 108.9398), "Kunming": (25.0453, 102.7100),
01109 |   "Guiyang": (26.6470, 106.6302), "Urumqi": (43.8256, 87.6168), "Lanzhou": (36.0611, 103.8343),
01110 |   "Xining": (36.6171, 101.7782), "Yinchuan": (38.4872, 106.2309), "Lhasa": (29.6500, 91.1000),
01111 |   "Mianyang": (31.4675, 104.6796), "Nanchong": (30.8378, 106.0849), "Zunyi": (27.6868, 106.9272),
01112 |   "Qujing": (25.4900, 103.7978), "Dali": (25.5916, 100.2227), "Lijiang": (26.8778, 100.2277),
01113 |   "Kashgar": (39.4677, 75.9938), "Turpan": (42.9452, 89.1835), "Jiuquan": (39.7289, 98.5042),
01114 |   "Baoji": (34.3619, 107.1448)
01115 | },
01116 | "Asia Selatan dan Timur Tengah": {
01117 |   "New Delhi": (28.6139, 77.2090), "Mumbai": (19.0760, 72.8777), "Bengaluru": (12.9716, 77.5946),
01118 |   "Chennai": (13.0827, 80.2707), "Kolkata": (22.5726, 88.3639), "Hyderabad": (17.3850, 78.4867),
01119 |   "Ahmedabad": (23.0225, 72.5714), "Pune": (18.5204, 73.8567), "Jaipur": (26.9124, 75.7873),
01120 |   "Surat": (21.1702, 72.8311), "Lucknow": (26.8467, 80.9462), "Kanpur": (26.4499, 80.3319),
01121 |   "Nagpur": (21.1458, 79.0882), "Indore": (22.7196, 75.8577), "Bhopal": (23.2599, 77.4126),
01122 |   "Patna": (25.5941, 85.1376), "Vadodara": (22.3072, 73.1812), "Agra": (27.1767, 78.0081),
01123 |   "Varanasi": (25.3176, 82.9739), "Ludhiana": (30.9010, 75.8573), "Karachi": (24.8607, 67.0011),
01124 |   "Lahore": (31.5204, 74.3587), "Islamabad": (33.6844, 73.0479), "Rawalpindi": (33.5984, 73.0441),
01125 |   "Faisalabad": (31.4504, 73.1350), "Multan": (30.1575, 71.5249), "Peshawar": (34.0151, 71.5249),
01126 |   "Quetta": (30.1798, 66.9750), "Gujranwala": (32.1617, 74.1883), "Sialkot": (32.4945, 74.5229),
01127 |   "Hyderabad PK": (25.3960, 68.3578), "Bahawalpur": (29.3957, 71.6833), "Sargodha": (32.0740, 72.6861),
01128 |   "Sukkur": (27.7052, 68.8574), "Larkana": (27.5570, 68.2028), "Sheikhupura": (31.7167, 73.9850),
01129 |   "Rahim Yar Khan": (28.4212, 70.2989), "Jhang": (31.2681, 72.3181), "Dera Ghazi Khan": (30.0489, 70.6317),
01130 |   "Gujrat": (32.5736, 74.0741), "Tehran": (35.6892, 51.3890), "Mashhad": (36.2972, 59.6067),
01131 |   "Isfahan": (32.6539, 51.6660), "Karaj": (35.8327, 50.9915), "Shiraz": (29.5918, 52.5836),
01132 |   "Tabriz": (38.0773, 46.2919), "Oom": (34.6416, 50.8746), "Ahvaz": (31.3183, 48.6706),
01133 |   "Kermanshah": (34.3142, 47.0650), "Urmia": (37.5527, 45.0761), "Rasht": (37.2799, 49.5886),
01134 |   "Zahedan": (29.4963, 60.8629), "Hamadan": (34.7982, 48.5146), "Kerman": (30.2839, 57.0834),
```

```

01135 |         "Yazd": (31.8974, 54.3569), "Ardabil": (38.2514, 48.2973), "Bandar Abbas": (27.1832, 56.2666),
01136 |         "Arak": (34.0954, 49.6909), "Zanjan": (36.6736, 48.4787), "Sanandaj": (35.3144, 46.9923),
01137 |         "Kabul": (34.5553, 69.2075), "Kandahar": (31.6200, 65.7158), "Herat": (34.3529, 62.2040),
01138 |         "Mazar-i-Sharif": (36.7000, 67.1167), "Jalalabad": (34.4265, 70.4515), "Kunduz": (36.7286, 68.8681),
01139 |         "Taloqan": (36.7361, 69.5345), "Puli Khumri": (35.9446, 68.7151), "Sheberghan": (36.6676, 65.7529),
01140 |         "Zaranj": (30.9596, 61.8604), "Maymana": (35.9214, 64.7836), "Ghazni": (33.5539, 68.4203),
01141 |         "Khost": (33.3338, 69.9172), "Shir Khan Bandar": (37.1704, 68.5833), "Charikar": (35.0136, 69.1715),
01142 |         "Lashkargah": (31.5938, 64.3716), "Farah": (32.3745, 62.1164), "Asadabad": (34.8731, 71.1469),
01143 |         "Gardez": (33.5856, 69.2259), "Bamyan": (34.8216, 67.8242), "Baghdad": (33.3152, 44.3661),
01144 |         "Basra": (30.5081, 47.7835), "Mosul": (36.3400, 43.1300), "Erbil": (36.1901, 44.0090),
01145 |         "Kirkuk": (35.4670, 44.3831), "Najaf": (31.9961, 44.3147), "Karbala": (32.6160, 44.0249),
01146 |         "Sulaymaniyah": (35.5618, 45.4328), "Nasiriyah": (31.0580, 46.2573), "Amarah": (31.8413, 47.1436),
01147 |         "Diwaniyah": (31.9868, 44.9215), "Kut": (32.5115, 45.8247), "Hilla": (32.4800, 44.4336),
01148 |         "Ramadi": (33.4243, 43.2989), "Fallujah": (33.3491, 43.7828), "Samarra": (34.1979, 43.8906),
01149 |         "Baqubah": (33.7466, 44.6247), "Halabja": (35.1778, 45.9861), "Zakho": (37.1415, 42.6848),
01150 |         "Duhok": (36.8679, 42.9884)
01151 |     }
01152 | }
01153 |
01154 | # =====
01155 | # FUNGSI UNDUH EPHEMERIS & MAP
01156 | # =====
01157 | def download_custom_bsp(filename, url):
01158 |     filepath = os.path.join(BASE_DIR, filename)
01159 |     if not os.path.exists(filepath):
01160 |         try:
01161 |             req = urllib.request.Request(url, headers={'User-Agent': 'Mozilla/5.0'})
01162 |             with urllib.request.urlopen(req) as response, open(filepath, 'wb') as out_file:
01163 |                 out_file.write(response.read())
01164 |         except Exception: pass
01165 |
01166 | # =====
01167 | # FUNGSI-FUNGSI FORMATTING & SAFETY
01168 | # =====
01169 | def get_safe_events(t_obj, y_obj):
01170 |     y_arr = np.atleast_1d(y_obj)
01171 |     if len(y_arr) == 0: return []
01172 |     events = []
01173 |     for k in range(len(y_arr)):
01174 |         try: t_val = t_obj[k]
01175 |         except Exception: t_val = t_obj
01176 |         events.append((t_val, int(y_arr[k])))
01177 |     return events
01178 |
01179 | def format_angle(deg, is_ra=False):
01180 |     if deg is None or math.isnan(deg): return "+00°:00':00\""
01181 |     if hasattr(deg, 'item'): deg = deg.item()
01182 |     sign = "+" if deg >= 0 else "-"
01183 |     deg = abs(deg)
01184 |     if is_ra:
01185 |         hours = deg / 15.0
01186 |         h = int(hours)
01187 |         m = int((hours - h) * 60)
01188 |         s = int(round((hours - h - m/60.0) * 3600))
01189 |         if s == 60: s = 0; m += 1
01190 |         if m == 60: m = 0; h += 1
01191 |         return f"{sign}{h:02d}H {m:02d}M {s:02d}S"
01192 |     else:
01193 |         d = int(deg)
01194 |         m = int((deg - d) * 60)
01195 |         s = int(round((deg - d - m/60.0) * 3600))
01196 |         if s == 60: s = 0; m += 1
01197 |         if m == 60: m = 0; d += 1
01198 |         return f"{sign}{d:02d}°:{m:02d}':{s:02d}\"
01199 |
01200 | def format_eph_angle(deg, is_ra=False, is_sd=False):
01201 |     if deg is None or math.isnan(deg): return "00H 00M 00S" if is_ra else "+00:00:00"
01202 |     if hasattr(deg, 'item'): deg = deg.item()
01203 |     sign = "+" if deg >= 0 else "-"
01204 |     deg = abs(deg)
01205 |     if is_ra:

```

```

01206 |         hours = deg / 15.0
01207 |         h = int(hours)
01208 |         m = int((hours - h) * 60)
01209 |         s = int(round((hours - h - m/60.0) * 3600))
01210 |         if s >= 60: s -= 60; m += 1
01211 |         if m >= 60: m -= 60; h += 1
01212 |         if h >= 24: h -= 24
01213 |         return f"{h:02d}H {m:02d}M {s:02d}S"
01214 |     else:
01215 |         d = int(deg)
01216 |         m = int((deg - d) * 60)
01217 |         s = int(round((deg - d - m/60.0) * 3600))
01218 |         if s >= 60: s -= 60; m += 1
01219 |         if m >= 60: m -= 60; d += 1
01220 |         if is_sd: return f"{d:02d}:{m:02d}:{s:02d}"
01221 |         if not is_sd and d >= 100: return f"{d:03d}:{m:02d}:{s:02d}"
01222 |         return f"{sign}{d:02d}:{m:02d}:{s:02d}"
01223 |
01224 | def format_time_hms(delta_hours):
01225 |     if hasattr(delta_hours, 'item'): delta_hours = delta_hours.item()
01226 |     sign = "+" if delta_hours >= 0 else "-"
01227 |     delta_hours = abs(delta_hours)
01228 |     h = int(delta_hours)
01229 |     m = int(round((delta_hours - h) * 60))
01230 |     if m == 60: m = 0; h += 1
01231 |     return f"{sign}{h:02d}H {m:02d}M"
01232 |
01233 | def get_hms_str(time_obj, tz_offset):
01234 |     if time_obj is None: return "---"
01235 |     dt_utc = time_obj.utcdatetime() # Gunakan output datetime asli
01236 |     dt_local = dt_utc + datetime.timedelta(hours=tz_offset)
01237 |     return dt_local.strftime("%H:%M")
01238 |
01239 | def get_live_weather_status(lat, lon):
01240 |     """ Mengambil data tutupan awan dari Open-Meteo. """
01241 |     url = f"https://api.open-meteo.com/v1/forecast?latitude={lat}&longitude={lon}&current=cloud_cover,relative_humidity_2m&timezone=auto"
01242 |     try:
01243 |         response = requests.get(url, timeout=5)
01244 |         response.raise_for_status()
01245 |         data = response.json()
01246 |
01247 |         cloud_cover = data['current']['cloud_cover']
01248 |         if cloud_cover > 80:
01249 |             return f"☁️ AWAN {cloud_cover}% - Fisik Berisiko Terhalang!", "#FF5252"
01250 |         elif cloud_cover > 50:
01251 |             return f"☁️ AWAN {cloud_cover}% - Ufuk Sebagian Terhalang", "#FFD54F"
01252 |         else:
01253 |             return f"☁️ CUACA CERAH: Awan {cloud_cover}% - Ideal untuk Rukyat", "#00E676"
01254 |     except Exception:
01255 |         return "☁️ Data cuaca tidak tersedia (Offline)", "#9E9E9E"
01256 |
01257 |
01258 | def calculate_saemundsson_refraction(alt_true, temp, press):
01259 |     import math
01260 |     if alt_true < -2.0: return 0.0
01261 |     r_arcmin = 1.02 / math.tan(math.radians(alt_true + 10.3 / (alt_true + 5.11)))
01262 |     correction = (press / 1010.0) * (283.15 / (273.15 + temp))
01263 |     return (r_arcmin * correction) / 60.0
01264 |
01265 | # Database Bintang Terang (Magnitude < 2.0) - Koordinat J2000
01266 | BRIGHT_STARS = [
01267 |     {"name": "Sirius", "ra": (6, 45, 8.9), "dec": (-16, 42, 58), "mag": -1.46},
01268 |     {"name": "Canopus", "ra": (6, 23, 57.1), "dec": (-52, 41, 45), "mag": -0.74},
01269 |     {"name": "Rigel Kentaurus", "ra": (14, 39, 36.5), "dec": (-60, 50, 2), "mag": -0.27},
01270 |     {"name": "Arcturus", "ra": (14, 15, 39.7), "dec": (19, 10, 57), "mag": -0.05},
01271 |     {"name": "Vega", "ra": (18, 36, 56.3), "dec": (38, 47, 1), "mag": 0.03},
01272 |     {"name": "Capella", "ra": (5, 16, 41.4), "dec": (45, 59, 53), "mag": 0.08},
01273 |     {"name": "Rigel", "ra": (5, 14, 32.3), "dec": (-8, 12, 6), "mag": 0.13},
01274 |     {"name": "Procyon", "ra": (7, 39, 18.1), "dec": (5, 13, 30), "mag": 0.34},
01275 |     {"name": "Achernar", "ra": (1, 37, 42.8), "dec": (-57, 14, 12), "mag": 0.46},
01276 |     {"name": "Betelgeuse", "ra": (5, 55, 10.3), "dec": (7, 24, 25), "mag": 0.50},

```

```

01277 |     {"name": "Hadar", "ra": (14, 3, 49.4), "dec": (-60, 22, 23), "mag": 0.61},
01278 |     {"name": "Altair", "ra": (19, 50, 47.0), "dec": (8, 52, 6), "mag": 0.76},
01279 |     {"name": "Aldebaran", "ra": (4, 35, 55.2), "dec": (16, 30, 33), "mag": 0.86},
01280 |     {"name": "Antares", "ra": (16, 29, 24.4), "dec": (-26, 25, 55), "mag": 0.96},
01281 |     {"name": "Spica", "ra": (13, 25, 11.6), "dec": (-11, 9, 41), "mag": 0.97},
01282 |     {"name": "Pollux", "ra": (7, 45, 18.9), "dec": (28, 1, 34), "mag": 1.14},
01283 |     {"name": "Fomalhaut", "ra": (22, 57, 39.1), "dec": (-29, 37, 20), "mag": 1.16},
01284 |     {"name": "Deneb", "ra": (20, 41, 25.9), "dec": (45, 16, 49), "mag": 1.25},
01285 |     {"name": "Regulus", "ra": (10, 8, 22.3), "dec": (11, 58, 2), "mag": 1.35},
01286 |     {"name": "Adhara", "ra": (6, 58, 37.5), "dec": (-28, 58, 19), "mag": 1.50},
01287 |     {"name": "Castor", "ra": (7, 34, 36.0), "dec": (31, 53, 18), "mag": 1.58}
01288 | ]
01289 |
01290 | class KomparasiObservatorium(ctk.CTkToplevel):
01291 |     def __init__(self, master, eph, ts):
01292 |         super().__init__(master)
01293 |
01294 |         self.title("Observatorium Live: Komparasi Stellarium vs Ephemeris NASA")
01295 |         self.geometry("1100x650")
01296 |         self.attributes("-topmost", True)
01297 |         self.configure(fg_color="#0A0A10")
01298 |
01299 |         # Menggunakan mesin Skyfield dari KHGTApp agar tidak perlu load file .bsp 2 kali
01300 |         self.ts = ts
01301 |         self.eph = eph
01302 |         self.ephemeris_loaded = True if self.eph else False
01303 |
01304 |         if self.ephemeris_loaded:
01305 |             self.earth = self.eph['earth']
01306 |             self.sun_calc = self.eph['sun']
01307 |             self.moon_calc = self.eph['moon']
01308 |
01309 |         self.title_label = ctk.CTkLabel(self, text="☐ KOMPARASI DATA: STELLARIUM vs KOMPUTASI ENGINE", font=("Segoe UI", 22, "bold"), text_color="#FFD54F")
01310 |         self.title_label.pack(pady=(20, 5))
01311 |
01312 |         self.info_label = ctk.CTkLabel(self, text="Memeriksa sistem dan memuat data API...", font=("Segoe UI", 14), text_color="#AAAAAA")
01313 |         self.info_label.pack(pady=(0, 20))
01314 |
01315 |         self.cek_dan_jalankan_stellarium()
01316 |
01317 |         self.main_container = ctk.CTkFrame(self, fg_color="transparent")
01318 |         self.main_container.pack(fill="both", expand=True, padx=20, pady=(0, 10))
01319 |         self.main_container.grid_columnconfigure(0, weight=1, uniform="group1")
01320 |         self.main_container.grid_columnconfigure(1, weight=1, uniform="group1")
01321 |         self.main_container.grid_rowconfigure(0, weight=1)
01322 |
01323 |         # --- PENAMBAHAN METRIK BARU ---
01324 |         self.metrics = [
01325 |             "Ketinggian (Alt)", "Arah (Azimuth)", "Jarak (AU)",
01326 |             "RA", "Deklinasi (Dec)", "Elongasi", "Iluminasi", "Magnitude"
01327 |         ]
01328 |
01329 |         # --- PANEL MATAHARI ---
01330 |         self.sun_frame = ctk.CTkFrame(self.main_container, corner_radius=15, fg_color="#1E1E1E", border_width=1, border_color="#FFD700")
01331 |         self.sun_frame.grid(row=0, column=0, padx=(0, 10), sticky="nsew")
01332 |         ctk.CTkLabel(self.sun_frame, text="☐ MATAHARI", font=("Segoe UI", 18, "bold"), text_color="#FFD700").pack(pady=15)
01333 |
01334 |         self.sun_table = ctk.CTkFrame(self.sun_frame, fg_color="transparent")
01335 |         self.sun_table.pack(fill="both", expand=True, padx=15, pady=(0, 15))
01336 |         self.sun_widgets = {}
01337 |         self.buat_header_tabel(self.sun_table)
01338 |         self.buat_baris_tabel(self.sun_table, self.metrics, self.sun_widgets)
01339 |
01340 |         # --- PANEL BULAN ---
01341 |         self.moon_frame = ctk.CTkFrame(self.main_container, corner_radius=15, fg_color="#1E1E1E", border_width=1, border_color="#00E5FF")
01342 |         self.moon_frame.grid(row=0, column=1, padx=(10, 0), sticky="nsew")
01343 |         ctk.CTkLabel(self.moon_frame, text="☐ BULAN", font=("Segoe UI", 18, "bold"), text_color="#00E5FF").pack(pady=15)
01344 |
01345 |         self.moon_table = ctk.CTkFrame(self.moon_frame, fg_color="transparent")
01346 |         self.moon_table.pack(fill="both", expand=True, padx=15, pady=(0, 15))
01347 |         self.moon_widgets = {}

```

```

01348 |         self.buat_header_tabel(self.moon_table)
01349 |         self.buat_baris_tabel(self.moon_table, self.metrics, self.moon_widgets)
01350 |
01351 |         self.status_label = ctk.CTkLabel(self, text="Status: Inisialisasi...", font=("Segoe UI", 13, "bold"))
01352 |         self.status_label.pack(side="bottom", pady=15)
01353 |
01354 |         # Pengaman agar loop berhenti saat jendela ditutup
01355 |         self.protocol("WM_DELETE_WINDOW", self.on_close)
01356 |         self.is_running = True
01357 |         self.update_dashboard()
01358 |
01359 |     def on_close(self):
01360 |         self.is_running = False
01361 |         self.destroy()
01362 |
01363 |     def cek_dan_jalankan_stellarium(self):
01364 |         try:
01365 |             import subprocess
01366 |             output = subprocess.check_output('tasklist', creationflags=0x08000000)
01367 |             if b'steellarium.exe' not in output.lower():
01368 |                 self.info_label.configure(text="Mendeteksi Stellarium belum aktif. Mencoba membuka otomatis...")
01369 |                 jalur_instalasi = [
01370 |                     r"C:\Program Files\Stellarium\stellarium.exe",
01371 |                     r"C:\Program Files (x86)\Stellarium\stellarium.exe",
01372 |                     os.path.expandvars(r"%LOCALAPPDATA%\Programs\Stellarium\stellarium.exe")
01373 |                 ]
01374 |                 berhasil_buka = False
01375 |                 for jalur in jalur_instalasi:
01376 |                     if os.path.exists(jalur):
01377 |                         os.startfile(jalur)
01378 |                         berhasil_buka = True
01379 |                         break
01380 |                 if not berhasil_buka:
01381 |                     self.info_label.configure(text="Gagal menemukan stellarium.exe. Silakan buka aplikasi secara manual.")
01382 |             except Exception:
01383 |                 pass
01384 |
01385 |     def buat_header_tabel(self, parent):
01386 |         headers = ["Metrik", "Stellarium", "Sistem KHGT", "A Selisih"]
01387 |         for i in range(4):
01388 |             parent.grid_columnconfigure(i, weight=1)
01389 |             lbl = ctk.CTkLabel(parent, text=headers[i], font=("Segoe UI", 12, "bold"), text_color="#FFFFFF")
01390 |             lbl.grid(row=0, column=i, pady=(0, 10), padx=5, sticky="w" if i==0 else "e")
01391 |
01392 |     def buat_baris_tabel(self, parent, metrics, widget_dict):
01393 |         for i, metric in enumerate(metrics, start=1):
01394 |             parent.grid_rowconfigure(i, weight=1)
01395 |             ctk.CTkLabel(parent, text=metric, font=("Segoe UI", 12), text_color="#AAAAAA").grid(row=i, column=0, sticky="w", padx=5, pady=8)
01396 |             lbl_stel = ctk.CTkLabel(parent, text="--", font=("Consolas", 14, "bold"), text_color="#00FFFF")
01397 |             lbl_stel.grid(row=i, column=1, sticky="e", padx=5, pady=8)
01398 |             lbl_calc = ctk.CTkLabel(parent, text="--", font=("Consolas", 14, "bold"), text_color="#00E676")
01399 |             lbl_calc.grid(row=i, column=2, sticky="e", padx=5, pady=8)
01400 |             lbl_delta = ctk.CTkLabel(parent, text="--", font=("Consolas", 14, "bold"), text_color="#FF5252")
01401 |             lbl_delta.grid(row=i, column=3, sticky="e", padx=5, pady=8)
01402 |             widget_dict[metric] = {"stel": lbl_stel, "calc": lbl_calc, "delta": lbl_delta}
01403 |
01404 |     def fetch_stellarium_state(self):
01405 |         import datetime
01406 |         import pytz
01407 |         import requests
01408 |         try:
01409 |             res = requests.get("http://localhost:8090/api/main/status", timeout=2)
01410 |             if res.status_code == 200:
01411 |                 data = res.json()
01412 |                 lat = data.get('location', {}).get('latitude', 0)
01413 |                 lon = data.get('location', {}).get('longitude', 0)
01414 |                 alt = data.get('location', {}).get('altitude', 0)
01415 |                 loc_name = data.get('location', {}).get('name', 'Unknown')
01416 |
01417 |                 utc_str = data.get('time', {}).get('utc', '')
01418 |             try:

```



```

01419 |         dt_utc = datetime.datetime.strptime(utc_str[:19], "%Y-%m-%dT%H:%M:%S")
01420 |         dt_utc = dt_utc.replace(tzinfo=pytz.utc)
01421 |     except Exception as e:
01422 |         dt_utc = datetime.datetime.now(pytz.utc)
01423 |
01424 |     return {"lat": lat, "lon": lon, "alt": alt, "loc_name": loc_name, "time": dt_utc, "ok": True}
01425 | except Exception as e:
01426 |     pass
01427 | return {"ok": False}
01428 |
01429 | def fetch_stellarium_object(self, name):
01430 |     import requests
01431 |     import math
01432 |     try:
01433 |         res = requests.get("http://localhost:8090/api/objects/info", params={'name': name, 'format': 'json'}, timeout=2)
01434 |         if res.status_code == 200:
01435 |             d = res.json()
01436 |
01437 |             # Stellarium API default mengembalikan elongasi dalam bentuk Radian
01438 |             elong_rad = d.get('elongation', 0)
01439 |             elong_deg = math.degrees(elong_rad) if elong_rad else 0.0
01440 |
01441 |             # --> PERBAIKAN ILUMINASI MATAHARI & BULAN STELLARIUM <--
01442 |             if name == "Sun":
01443 |                 iluminasi_stel = 100.0 # Matahari selalu 100%
01444 |             else:
01445 |                 iluminasi_stel = d.get('illuminated')
01446 |                 if iluminasi_stel is None:
01447 |                     iluminasi_stel = d.get('phase', 0.0) * 100.0
01448 |             # -----
01449 |
01450 |             return {
01451 |                 "Ketinggian (Alt)": d.get('altitude', 0),
01452 |                 "Arah (Azimuth)": d.get('azimuth', 0),
01453 |                 "Jarak (AU)": d.get('distance', 0),
01454 |                 "RA": d.get('ra', 0),
01455 |                 "Deklinasi (Dec)": d.get('dec', 0),
01456 |                 "Elongasi": elong_deg,
01457 |                 "Iluminasi": iluminasi_stel,
01458 |                 "Magnitude": d.get('vmag', 0)
01459 |             }
01460 |     except:
01461 |         pass
01462 |     return None
01463 |
01464 | def kalkulasi_skyfield(self, target_name, state):
01465 |     if not self.ephemeris_loaded: return None
01466 |     from skyfield.api import wgs84
01467 |     from skyfield import almanac
01468 |     import math
01469 |
01470 |     t = self.ts.from_datetime(state['time'])
01471 |     observer = self.earth + wgs84.latlon(state['lat'], state['lon'], state['alt'])
01472 |
01473 |     target_eph = self.sun_calc if target_name == "Sun" else self.moon_calc
01474 |     apparent = observer.at(t).observe(target_eph).apparent()
01475 |     alt, az, dist = apparent.altaz()
01476 |     ra, dec, _ = apparent.radec(epoch='date')
01477 |
01478 |     # --- KALKULASI EKSTRA (ELONGASI, ILUMINASI, MAGNITUDE) ---
01479 |     elongation = 0.0
01480 |     illumination = 100.0
01481 |     magnitude = 0.0
01482 |
01483 |     if target_name == "Moon":
01484 |         sun_app = observer.at(t).observe(self.sun_calc).apparent()
01485 |         elongation = sun_app.separation_from(apparent).degrees
01486 |
01487 |         illum_val = almanac.fraction_illuminated(self.eph, 'moon', t)
01488 |         illumination = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0
01489 |

```

```

01490 |         # Aproksimasi Magnitude Visual Bulan (Rumus Astronomis Standar)
01491 |         phase_angle = math.degrees(math.acos(2 * (illumination / 100.0) - 1)) if illumination <= 100.0 else 0
01492 |         magnitude = -12.73 + 0.026 * abs(phase_angle) + (4e-9) * (phase_angle**4)
01493 |
01494 |     elif target_name == "Sun":
01495 |         elongation = 0.0
01496 |         illumination = 100.0
01497 |         magnitude = -26.74
01498 |
01499 |     return {
01500 |         "Ketinggian (Alt)": alt.degrees,
01501 |         "Arah (Azimuth)": az.degrees,
01502 |         "Jarak (AU)": dist.au,
01503 |         "RA": ra.degrees,
01504 |         "Deklinasi (Dec)": dec.degrees,
01505 |         "Elongasi": elongation,
01506 |         "Iluminasi": illumination,
01507 |         "Magnitude": magnitude
01508 |     }
01509 |
01510 | def perbarui_ui_tabel(self, widgets, data_stel, data_calc):
01511 |     for metrik in self.metrics:
01512 |         if data_stel and data_calc:
01513 |             v_stel = data_stel.get(metrik, 0)
01514 |             v_calc = data_calc.get(metrik, 0)
01515 |             delta = abs(v_stel - v_calc)
01516 |
01517 |             # Format Tampilan Berbeda Berdasarkan Jenis Metrik
01518 |             if metrik == "Jarak (AU)":
01519 |                 fmt_stel, fmt_calc, fmt_delta = f"{v_stel:.6f}", f"{v_calc:.6f}", f"{delta:.6f}"
01520 |             elif metrik == "Iluminasi":
01521 |                 fmt_stel, fmt_calc, fmt_delta = f"{v_stel:.2f}%", f"{v_calc:.2f}%", f"{delta:.2f}%"
01522 |             elif metrik == "Magnitude":
01523 |                 fmt_stel, fmt_calc, fmt_delta = f"{v_stel:.2f}", f"{v_calc:.2f}", f"{delta:.2f}"
01524 |             else:
01525 |                 fmt_stel, fmt_calc, fmt_delta = f"{v_stel:.4f}°", f"{v_calc:.4f}°", f"{delta:.4f}°"
01526 |
01527 |             widgets[metrik]["stel"].configure(text=fmt_stel)
01528 |             widgets[metrik]["calc"].configure(text=fmt_calc)
01529 |             widgets[metrik]["delta"].configure(text=fmt_delta)
01530 |         else:
01531 |             widgets[metrik]["stel"].configure(text="--")
01532 |             widgets[metrik]["calc"].configure(text="--")
01533 |             widgets[metrik]["delta"].configure(text="--")
01534 |
01535 | def update_dashboard(self):
01536 |     if not self.is_running: return
01537 |     state = self.fetch_stellarium_state()
01538 |
01539 |     if state['ok'] and self.ephemeris_loaded:
01540 |         waktu_str = state['time'].strftime("%d-%m-%Y %H:%M:%S UTC")
01541 |         self.info_label.configure(text=f"Lokasi: {state['loc_name']} (Lat: {state['lat']:.2f}, Lon: {state['lon']:.2f}) | Waktu Komputasi: {waktu_str}")
01542 |
01543 |         sun_stel = self.fetch_stellarium_object("Sun")
01544 |         sun_calc = self.kalkulasi_skyfield("Sun", state)
01545 |         self.perbarui_ui_tabel(self.sun_widgets, sun_stel, sun_calc)
01546 |
01547 |         moon_stel = self.fetch_stellarium_object("Moon")
01548 |         moon_calc = self.kalkulasi_skyfield("Moon", state)
01549 |         self.perbarui_ui_tabel(self.moon_widgets, moon_stel, moon_calc)
01550 |
01551 |         self.status_label.configure(text="● Sinkronisasi Data Berjalan Baik", text_color="#00E676")
01552 |     else:
01553 |         if not self.ephemeris_loaded:
01554 |             self.status_label.configure(text="❏ Error: Mesin Ephemeris belum dimuat", text_color="#FF5252")
01555 |         else:
01556 |             self.status_label.configure(text="● Menunggu API... (Pastikan Plugin 'Remote Control' AKTIF di Stellarium -> [F2] -> Plugins)", text_color="#FFD700")
01557 |             for widgets in [self.sun_widgets, self.moon_widgets]:
01558 |                 for key in widgets:
01559 |                     widgets[key]["stel"].configure(text="--")
01560 |                     widgets[key]["calc"].configure(text="--")

```

```

01561 |         widgets[key]["delta"].configure(text="--")
01562 |
01563 |     self.after(1000, self.update_dashboard)
01564 |
01565 | # =====
01566 | # KELAS UTAMA APLIKASI GUI
01567 | # =====
01568 | class KHGTApp(ctk.CTk):
01569 |     def __init__(self):
01570 |         super().__init__()
01571 |
01572 |         # ---> INISIALISASI SISTEM BAHASA <---
01573 |         self.current_lang = "id"
01574 |         self.translations = {}
01575 |         self.load_translations()
01576 |         self._last_raw_report = ""
01577 |
01578 |         # Penanda alur bahasa:
01579 |         # False = kondisi awal/beranda rilis, belum ada menu 1-41 yang dipilih pengguna.
01580 |         # True = pengguna sudah memilih salah satu menu 1-41, sehingga pergantian
01581 |         #         bahasa harus mengikuti mekanisme lama: sidebar kiri + tampilan modul.
01582 |         self.user_has_selected_main_menu = False
01583 |
01584 |         # Cache JSON khusus halaman INFORMASI RILIS.
01585 |         # Tujuan: saat klik ID/EN/AR, halaman kanan tidak menerjemahkan ulang
01586 |         # seluruh teks, tetapi membaca hasil terjemahan yang sudah tersimpan.
01587 |         self.release_lang_cache_path = self.get_release_lang_cache_path()
01588 |         self.release_translation_cache = {}
01589 |         self._release_translation_cache_dirty = False
01590 |         self.load_release_language_cache()
01591 |         # -----
01592 |
01593 |         self.title("KHGT Times: Kalkulator Integrasi KHGT, Ephemeris, Qiblah & Prayer Times")
01594 |         self.geometry("1150x850")
01595 |
01596 |         # --- KODE UNTUK MENGGANTI ICON POJOK KIRI ATAS ---
01597 |         try:
01598 |             # Pastikan logo.png berada di folder yang sama dengan script ini
01599 |             icon_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), "logo.png")
01600 |
01601 |             # Buka gambar menggunakan Pillow dan convert ke format PhotoImage Tkinter
01602 |             icon_image = ImageTk.PhotoImage(Image.open(icon_path))
01603 |
01604 |             # Terapkan ke window (True berarti berlaku untuk window utama dan semua sub-window)
01605 |             self.wm_iconphoto(True, icon_image)
01606 |         except Exception as e:
01607 |             print(f"Peringatan: Gagal memuat ikon logo.png ({e})")
01608 |         # -----
01609 |
01610 |         self.minsize(900, 700)
01611 |
01612 |         self.protocol("WM_DELETE_WINDOW", self.on_closing)
01613 |
01614 |         self.load_obj = Loader(BASE_DIR, verbose=False)
01615 |         self.ts = self.load_obj.timescale()
01616 |         self.eph = None
01617 |         self.ephemeris_name = None
01618 |         # Cache dua kernel agar perpindahan tahun tidak perlu membuka file berulang.
01619 |         # Kernel panjang baru dimuat ketika benar-benar diperlukan.
01620 |         self._ephemeris_cache = {}
01621 |         self._ephemeris_lock = threading.RLock()
01622 |
01623 |         self.lokasi_nama = ctk.StringVar(value="Semarang, Jawa Tengah")
01624 |
01625 |         # --- Inisialisasi Sistem Alarm ---
01626 |         self.alarm_enabled = ctk.BooleanVar(value=True)
01627 |         default_audio = os.path.join(BASE_DIR, "adzan.mp3")
01628 |         if os.path.exists(default_audio):
01629 |             self.adzan_audio_path = ctk.StringVar(value=default_audio)
01630 |         else:
01631 |             self.adzan_audio_path = ctk.StringVar(value="")

```

```

01632 |
01633 |         self.daily_prayer_schedule = {}
01634 |         self.last_calculated_date = None
01635 |         self.last_triggered_prayer = None
01636 |         self.snooze_time = None
01637 |         self.snooze_prayer = None
01638 |
01639 |         if HAS_TOAST:
01640 |             self.toaster = ToastNotifier()
01641 |         if HAS_PYGAME:
01642 |             try:
01643 |                 pygame.mixer.init()
01644 |             except Exception as e:
01645 |                 print("Pygame mixer init failed:", e)
01646 |
01647 |         # --- Variabel Simulasi 3D Matplotlib ---
01648 |         self.eph3d_updating = False
01649 |         self.eph3d_is_live = False
01650 |         self.pe_obs = ephem.Observer()
01651 |         self.pe_sun = ephem.Sun()
01652 |         self.pe_moon = ephem.Moon()
01653 |
01654 |         # ---> TAMBAHAN VARIABEL KURVA ALASKA <---
01655 |         self.valid_days_alaska = []
01656 |         self.valid_alt_alaska = []
01657 |         self.valid_elong_alaska = []
01658 |         self.hover_data_alaska = {}
01659 |
01660 |         self.setup_ui()
01661 |
01662 |         threading.Thread(target=self.load_ephemeris, daemon=True).start()
01663 |         self.update_calendar_widget()
01664 |
01665 |         # Start Alarm Tick Loop
01666 |         self.after(1000, self.alarm_tick)
01667 |
01668 |         # Start Matplotlib 3D Ephemeris Loop
01669 |         self.eph3d_live_update_loop()
01670 |
01671 |     def get_header(self, width):
01672 |         lines = [
01673 |             "",
01674 |             "KALENDER HIJRIAH GLOBAL TUNGAL",
01675 |             "KHGT Times 8.0, By Kasmui"
01676 |         ]
01677 |         return "\n".join(line.center(width) for line in lines)
01678 |
01679 |     def _ephemeris_filename_for_year(self, target_year):
01680 |         """Pilih kernel JPL berdasarkan tahun tanpa mengubah logika modul lain."""
01681 |         y = int(target_year)
01682 |         if EPHEMERIS_SHORT_MIN_YEAR <= y <= EPHEMERIS_SHORT_MAX_YEAR:
01683 |             return EPHEMERIS_SHORT_FILE
01684 |         return EPHEMERIS_LONG_FILE
01685 |
01686 |     def _load_ephemeris_file(self, filename):
01687 |         """Muat satu kernel JPL; kernel yang pernah dibuka disimpan dalam cache."""
01688 |         filepath = os.path.join(BASE_DIR, filename)
01689 |         if not os.path.exists(filepath):
01690 |             raise FileNotFoundError(
01691 |                 f"File ephemeris '{filename}' tidak ditemukan di folder aplikasi: {BASE_DIR}"
01692 |             )
01693 |
01694 |         with self._ephemeris_lock:
01695 |             if filename in self._ephemeris_cache:
01696 |                 kernel = self._ephemeris_cache[filename]
01697 |             else:
01698 |                 kernel = self.load_obj(filename)
01699 |                 if 'earth' not in kernel or 'sun' not in kernel or 'moon' not in kernel:
01700 |                     raise ValueError(
01701 |                         f"Data utama Earth/Sun/Moon tidak ditemukan dalam '{filename}'."
01702 |                     )

```

```

01703 |         self._ephemeris_cache[filename] = kernel
01704 |
01705 |         self.eph = kernel
01706 |         self.ephemeris_name = filename
01707 |         return kernel
01708 |
01709 | def load_ephemeris(self, target_year=None):
01710 |     """
01711 |     Muat ephemeris otomatis.
01712 |     - 1850-2149 : de440s.bsp
01713 |     - 1-1849 dan 2150-2999 : de441-new.bsp
01714 |
01715 |     Pemilihan 1850-2149 sengaja memakai tahun penuh karena de440s mulai
01716 |     sekitar 26 Desember 1849 dan berakhir sekitar 22 Januari 2150.
01717 |     """
01718 |     try:
01719 |         if target_year is None:
01720 |             target_year = datetime.datetime.now().year
01721 |
01722 |             target_year = int(target_year)
01723 |             if target_year < EPHEMERIS_LONG_MIN_YEAR or target_year > EPHEMERIS_LONG_MAX_YEAR:
01724 |                 raise ValueError(
01725 |                     f"Tahun {target_year} di luar rentang aplikasi "
01726 |                     f"({EPHEMERIS_LONG_MIN_YEAR}-{EPHEMERIS_LONG_MAX_YEAR} M)."
01727 |                 )
01728 |
01729 |             filename = self._ephemeris_filename_for_year(target_year)
01730 |             filepath = os.path.join(BASE_DIR, filename)
01731 |
01732 |             if not os.path.exists(filepath):
01733 |                 self.lbl_status.configure(
01734 |                     text=f"Error: {filename} Tidak Ditemukan!",
01735 |                     text_color="#FF1744"
01736 |                 )
01737 |                 self.after(0, lambda f=filename: messagebox.showerror(
01738 |                     "File Ephemeris Hilang",
01739 |                     f"File '{f}' wajib ada di folder aplikasi.\n\n"
01740 |                     f"KHGT Times V8.0 memakai:\n"
01741 |                     f"• {EPHEMERIS_SHORT_FILE} untuk {EPHEMERIS_SHORT_MIN_YEAR}-{EPHEMERIS_SHORT_MAX_YEAR} M\n"
01742 |                     f"• {EPHEMERIS_LONG_FILE} untuk {EPHEMERIS_LONG_MIN_YEAR}-{EPHEMERIS_SHORT_MIN_YEAR - 1} M dan "
01743 |                     f"{EPHEMERIS_SHORT_MAX_YEAR + 1}-{EPHEMERIS_LONG_MAX_YEAR} M"
01744 |                 ))
01745 |             return
01746 |
01747 |             self.lbl_status.configure(
01748 |                 text=f"Memuat {filename} untuk tahun {target_year}...",
01749 |                 text_color="#00E5FF"
01750 |             )
01751 |
01752 |             self._load_ephemeris_file(filename)
01753 |
01754 |             self.lbl_status.configure(
01755 |                 text=f"Sistem Siap ({self.ephemeris_name})",
01756 |                 text_color="#00E676"
01757 |             )
01758 |             self.btn_hitung.configure(state="normal")
01759 |
01760 |     except Exception as e:
01761 |         self.lbl_status.configure(text="Gagal memuat Ephemeris!", text_color="#FF1744")
01762 |         self.after(0, lambda err=str(e): messagebox.showerror(
01763 |             "Error Mesin", f"Gagal membaca file ephemeris: {err}"
01764 |         ))
01765 |
01766 | def toggle_sidebar(self):
01767 |     if self.sidebar_visible:
01768 |         self.sidebar.grid_remove()
01769 |         self.sidebar_visible = False
01770 |     else:
01771 |         self.sidebar.grid(row=0, column=0, sticky="nsew")
01772 |         self.sidebar_visible = True
01773 |

```

```

01774 | def open_peta_langit_dunia(self):
01775 |     """Membuka Peta Langit Dunia KHGT Times melalui browser default."""
01776 |     try:
01777 |         opened = webbrowser.open_new_tab(WORLD_SKY_MAP_URL)
01778 |         if not opened:
01779 |             opened = webbrowser.open(WORLD_SKY_MAP_URL)
01780 |         self.lbl_status.configure(
01781 |             text="Peta Langit Dunia dibuka di browser",
01782 |             text_color="#00E676"
01783 |         )
01784 |         if not opened:
01785 |             messagebox.showinfo(
01786 |                 "Peta Langit Dunia",
01787 |                 f"Silakan buka alamat berikut pada browser:\n\n{WORLD_SKY_MAP_URL}"
01788 |             )
01789 |     except Exception as e:
01790 |         messagebox.showerror(
01791 |             "Gagal Membuka Peta Langit Dunia",
01792 |             f"Layanan online tidak dapat dibuka melalui browser.\n\n{WORLD_SKY_MAP_URL}\n\n{e}"
01793 |         )
01794 |         self.lbl_status.configure(
01795 |             text="Gagal membuka Peta Langit Dunia",
01796 |             text_color="#FF1744"
01797 |         )
01798 |
01799 | def open_laporan_astronomi(self):
01800 |     """Membuka aplikasi pendamping LaporanAstronomi.exe dari folder aplikasi."""
01801 |     kandidat_nama = ("LaporanAstronomi.exe", "LaporanAstronomi.exe")
01802 |     exe_path = next(
01803 |         (os.path.join(BASE_DIR, nama) for nama in kandidat_nama
01804 |          if os.path.isfile(os.path.join(BASE_DIR, nama))),
01805 |         None
01806 |     )
01807 |
01808 |     if exe_path is None:
01809 |         lokasi_utama = os.path.join(BASE_DIR, kandidat_nama[0])
01810 |         messagebox.showerror(
01811 |             "Laporan Astronomi Tidak Ditemukan",
01812 |             "File aplikasi pendamping tidak ditemukan.\n\n"
01813 |             f"Letakkan file 'LaporanAstronomi.exe' di folder yang sama dengan "
01814 |             f"KHGT Times.\n\nLokasi yang diperiksa:\n{lokasi_utama}"
01815 |         )
01816 |         return
01817 |
01818 |     try:
01819 |         if os.name == "nt" and hasattr(os, "startfile"):
01820 |             os.startfile(exe_path)
01821 |         else:
01822 |             subprocess.Popen([exe_path], cwd=os.path.dirname(exe_path))
01823 |         self.lbl_status.configure(
01824 |             text="LaporanAstronomi.exe berhasil dibuka",
01825 |             text_color="#00E676"
01826 |         )
01827 |     except Exception as e:
01828 |         messagebox.showerror(
01829 |             "Gagal Membuka Laporan Astronomi",
01830 |             f"File ditemukan, tetapi gagal dijalankan.\n\n{e}"
01831 |         )
01832 |         self.lbl_status.configure(
01833 |             text="Gagal membuka LaporanAstronomi.exe",
01834 |             text_color="#FF1744"
01835 |         )
01836 |
01837 | def setup_ui(self):
01838 |     now = datetime.datetime.now()
01839 |     curr_y = str(now.year)
01840 |     curr_m = f"{now.month:02d}"
01841 |     curr_d = f"{now.day:02d}"
01842 |     curr_m_np = str(now.month)
01843 |     curr_d_np = str(now.day)
01844 |

```

```

01845 |         self.grid_columnconfigure(0, weight=0)
01846 |         self.grid_columnconfigure(1, weight=1)
01847 |         self.grid_rowconfigure(0, weight=1)
01848 |
01849 |         # ===== SIDEBAR =====
01850 |         self.sidebar = ctk.CTkScrollableFrame(self, width=370, corner_radius=0, fg_color="#181818")
01851 |         self.sidebar.grid(row=0, column=0, sticky="nsew")
01852 |
01853 |         # ---> TAMBAHAN UI BAHASA DI KANAN "KHGT ENGINE" <---
01854 |         header_app_frame = ctk.CTkFrame(self.sidebar, fg_color="transparent")
01855 |         header_app_frame.pack(fill="x", padx=15, pady=(20, 5))
01856 |
01857 |         ctk.CTkLabel(header_app_frame, text="KHGT ENGINE", font=("Segoe UI", 24, "bold"), text_color="#00E5FF").pack(side="left")
01858 |
01859 |         self.lang_var = ctk.StringVar(value="id")
01860 |         self.combo_lang = ctk.CTkOptionMenu(header_app_frame, values=["id", "en", "ar"], variable=self.lang_var, width=60, height=28, command=self.change_language)
01861 |         self.combo_lang.pack(side="right")
01862 |         # -----
01863 |
01864 |         # =====
01865 |         # PENGANTI MENU SELECTION (MODERN SCROLLABLE LIST)
01866 |         # =====
01867 |         # Menggunakan CTkScrollableFrame agar desainnya 100% senada dengan tema aplikasi
01868 |         self.frame_menu_container = ctk.CTkScrollableFrame(
01869 |             self.sidebar,
01870 |             height=280, # Tinggi kotak scroll
01871 |             fg_color="#121212",
01872 |             corner_radius=8,
01873 |             border_width=1,
01874 |             border_color="#333333",
01875 |             scrollbar_button_color="#424242",
01876 |             scrollbar_button_hover_color="#616161"
01877 |         )
01878 |         self.frame_menu_container.pack(fill="x", padx=15, pady=(5, 10))
01879 |
01880 |         menu_items = [
01881 |             # Kalender & Konversi
01882 |             "1) Kalender Hijriah Berjalan",
01883 |             "2) Kalender Masehi Berjalan",
01884 |             "3) Konversi Kalender",
01885 |
01886 |             # Analisis & Visibilitas Hilal (KHGT)
01887 |             "4) Visibility Hilal (KHGT)",
01888 |             "5) Analisis Hilal Global",
01889 |             "6) Status Kriteria Batas",
01890 |             "7) Kota Pertama KHGT (Mainland)",
01891 |             "8) Crescent Visibility Map",
01892 |             "9) Altitude Chart Analyser",
01893 |
01894 |             # Data Histori & Komparasi 50 Tahun
01895 |             "10) Komparasi 50 Tahun (Ramadhan)",
01896 |             "11) Tabel Tinggi Hilal 50 Thn (Ramadhan)",
01897 |             "12) Tabel Elongasi Hilal 50 Thn (Ramadhan)",
01898 |             "13) Komparasi 50 Tahun (Syawal)",
01899 |             "14) Tabel Tinggi Hilal 50 Thn (Syawal)",
01900 |             "15) Tabel Elongasi Hilal 50 Thn (Syawal)",
01901 |             "16) Komparasi 50 Tahun (Zulhijah)",
01902 |             "17) Tabel Tinggi Hilal 50 Thn (Zulhijah)",
01903 |             "18) Tabel Elongasi Hilal 50 Thn (Zulhijah)",
01904 |             "19) Komparasi 50 Tahun (12 Bulan)",
01905 |
01906 |             # Ephemeris & Posisi Benda Langit
01907 |             "20) Fase Bulan (Moonphase)",
01908 |             "21) Moon Times",
01909 |             "22) Sun Times",
01910 |             "23) Sun Moon Ephemeris",
01911 |             "24) Sistem Sun Moon Earth",
01912 |
01913 |             # Waktu Shalat & Arah Kiblat
01914 |             "25) Prayer Times",
01915 |             "26) Qiblah Direction & Times",

```

```

01916 |         "27) Qibla Time (Rashdul Lokal)",
01917 |         "28) Kalkulator Mizwala (Bayangan)",
01918 |         "29) Evaluasi Fajar Shadiq & SQM",
01919 |
01920 |         # Fenomena Astronomi & Observasi
01921 |         "30) Analisis Gerhana",
01922 |         "31) Equinox & Solstice",
01923 |         "32) Planetary Times",
01924 |         "33) Generator Analemma Matahari",
01925 |         "34) Prediktor Pasang Surut Gravitasi",
01926 |         "35) Astrofotografi & Kontras Visibilitas",
01927 |
01928 |         # Simulasi, Ekstra & Admin
01929 |         "36) Live Animasi",
01930 |         "37) Simulasi Ephemeris 3D",
01931 |         "38) WinAI: Aplikasi AI Windows",
01932 |         "39) Peta Langit Dunia",
01933 |         "40) Laporan Astronomi",
01934 |         "41) HIJRI_DB Auto-Builder (Admin)"
01935 |     ]
01936 |
01937 |     self.menu_buttons = {}
01938 |     self.selected_mode = ctk.StringVar(value=menu_items[0])
01939 |
01940 |     def on_menu_click(mode_name):
01941 |         # ---> TAMBAHAN PASSWORD UNTUK MENU AUTO-BUILDER <---
01942 |         if "Auto-Builder" in mode_name:
01943 |             from tkinter import simpledialog
01944 |             # Gunakan simpledialog agar input bisa disamarkan dengan tanda bintang (show='')
01945 |             pwd = simpledialog.askstring("Otorisasi Admin", "Masukkan Password Admin:", show='')
01946 |             if pwd != "****REDACTED_ADMIN_PASSWORD****":
01947 |                 messagebox.showerror("Akses Ditolak", "Password salah atau dibatalkan!")
01948 |                 return # Batal pindah menu dan kembali ke menu sebelumnya
01949 |             # -----
01950 |
01951 |             # Tandai bahwa setelah fase startup, pengguna benar-benar sudah masuk
01952 |             # ke salah satu menu 1-41. Saat startup flag ini akan di-reset kembali
01953 |             # sesudah show_release_info() agar halaman awal tetap dianggap beranda rilis.
01954 |             self.user_has_selected_main_menu = True
01955 |
01956 |             self.selected_mode.set(mode_name)
01957 |             # Update warna tombol (Highlight menu yang sedang aktif)
01958 |             for name, btn in self.menu_buttons.items():
01959 |                 if name == mode_name:
01960 |                     btn.configure(fg_color="#1565C0", text_color="#FFFFFF") # Warna aktif (Background Biru, Teks Putih)
01961 |                 else:
01962 |                     btn.configure(fg_color="transparent", text_color="#00E5FF") # Warna normal (Teks Cyan)
01963 |
01964 |             # Panggil fungsi pergantian UI utama
01965 |             self.switch_mode(mode_name)
01966 |
01967 |     # Generate daftar menu ke dalam kotak scroll
01968 |     for item in menu_items:
01969 |         btn = ctk.CTkButton(
01970 |             self.frame_menu_container,
01971 |             text=item,
01972 |             anchor="w", # Rata kiri
01973 |             fg_color="transparent",
01974 |             text_color="#00E5FF",
01975 |             hover_color="#1F1F1F",
01976 |             font=("Segoe UI", 14, "bold"),
01977 |             height=35,
01978 |             corner_radius=6,
01979 |             command=lambda m=item: on_menu_click(m)
01980 |         )
01981 |         btn.pack(fill="x", pady=2, padx=2)
01982 |         self.menu_buttons[item] = btn
01983 |
01984 |     # Class jembatan agar fungsi 'self.combo_mode.get()' bawaan aplikasi tetap berfungsi normal tanpa error
01985 |     class MenuBridge:
01986 |         def __init__(self, var):

```



```

01987 |         self.var = var
01988 |     def get(self):
01989 |         return self.var.get()
01990 |     def set(self, val):
01991 |         pass
01992 |
01993 |     self.combo_mode = MenuBridge(self.selected_mode)
01994 |     # =====
01995 |
01996 |     # --- CONTAINER DATABASE KOTA ---
01997 |     self.frame_city_select = ctk.CTkFrame(self.sidebar, fg_color="#212121")
01998 |     self.frame_city_select.pack(fill="x", padx=15, pady=5)
01999 |     ctk.CTkLabel(self.frame_city_select, text="PILIH KOTA (DEFAULT SEMARANG)", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02000 |
02001 |     self.opt_prov = ctk.CTkOptionMenu(self.frame_city_select, values=sorted(list(CITY_DB.keys())), command=self.on_prov_change)
02002 |     self.opt_prov.pack(fill="x", padx=10, pady=5)
02003 |     self.opt_prov.set("Jawa Tengah")
02004 |
02005 |     self.opt_city = ctk.CTkOptionMenu(self.frame_city_select, values=sorted(list(CITY_DB["Jawa Tengah"].keys())), command=self.on_city_change)
02006 |     self.opt_city.pack(fill="x", padx=10, pady=(0, 10))
02007 |     self.opt_city.set("Semarang")
02008 |
02009 |     self.btn_auto_loc = ctk.CTkButton(self.frame_city_select, text="📍 Deteksi Lokasi Otomatis", fg_color="#00695C", hover_color="#004D40", command=self.auto_detect_location)
02010 |     self.btn_auto_loc.pack(fill="x", padx=10, pady=(0, 10))
02011 |
02012 |     # --- CONTAINER ALARM SALAT ---
02013 |     self.frame_alarm_settings = ctk.CTkFrame(self.sidebar, fg_color="#212121", border_width=1, border_color="#D32F2F")
02014 |     self.frame_alarm_settings.pack(fill="x", padx=15, pady=5)
02015 |     ctk.CTkLabel(self.frame_alarm_settings, text="PENGATURAN ALARM SALAT", font=("Segoe UI", 11, "bold"), text_color="#FF5252").pack(anchor="w", padx=10, pady=(8, 2))
02016 |
02017 |     self.switch_alarm = ctk.CTkSwitch(self.frame_alarm_settings, text="Aktifkan Alarm", variable=self.alarm_enabled, command=self.on_alarm_toggle, progress_color="#D32F2F")
02018 |     self.switch_alarm.pack(anchor="w", padx=15, pady=5)
02019 |
02020 |     btn_pilih_audio = ctk.CTkButton(self.frame_alarm_settings, text="🔊 Pilih Audio Adzan", fg_color="#424242", hover_color="#616161", command=self.pilih_file_audio)
02021 |     btn_pilih_audio.pack(fill="x", padx=10, pady=5)
02022 |
02023 |     current_audio = self.adzan_audio_path.get()
02024 |     if current_audio and os.path.exists(current_audio):
02025 |         initial_text = os.path.basename(current_audio)
02026 |     else:
02027 |         initial_text = "Default System Sound"
02028 |
02029 |     self.lbl_audio_path = ctk.CTkLabel(self.frame_alarm_settings, text=initial_text, font=("Segoe UI", 10, "italic"), text_color="#9E9E9E")
02030 |     self.lbl_audio_path.pack(anchor="w", padx=10, pady=(0, 8))
02031 |
02032 |     # --- CONTAINER 1: VISIBILITY INPUTS ---
02033 |     self.frame_vis_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02034 |     frame_vdate = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
02035 |     frame_vdate.pack(fill="x", padx=15, pady=5)
02036 |     ctk.CTkLabel(frame_vdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02037 |
02038 |     # SATUKAN LABEL DAN INPUT DALAM SATU FRAME GRID
02039 |     vdate_grid = ctk.CTkFrame(frame_vdate, fg_color="transparent")
02040 |     vdate_grid.pack(fill="x", padx=10, pady=(0, 10))
02041 |
02042 |     # Baris 0: Label Petunjuk
02043 |     ctk.CTkLabel(vdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02044 |     ctk.CTkLabel(vdate_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02045 |     ctk.CTkLabel(vdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
02046 |
02047 |     # Baris 1: Kotak Input
02048 |     self.entry_vday = ctk.CTkEntry(vdate_grid, width=45, placeholder_text="dd", justify="center")
02049 |     self.entry_vday.insert(0, curr_d)
02050 |     self.entry_vday.grid(row=1, column=0, padx=(0, 5))
02051 |
02052 |     self.entry_vmonth = ctk.CTkEntry(vdate_grid, width=45, placeholder_text="mm", justify="center")
02053 |     self.entry_vmonth.insert(0, curr_m)
02054 |     self.entry_vmonth.grid(row=1, column=1, padx=5)
02055 |
02056 |     self.entry_vyear = ctk.CTkEntry(vdate_grid, width=70, placeholder_text="yyyy", justify="center")
02057 |     self.entry_vyear.insert(0, curr_y)

```

```

02058 |         self.entry_vyear.grid(row=1, column=2, padx=(5, 0))
02059 |
02060 |         # ---> AWAL TAMBAHAN UI PILIHAN WAKTU OBSERVASI <---
02061 |         frame_vtime_mode = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
02062 |         frame_vtime_mode.pack(fill="x", padx=15, pady=5)
02063 |         ctk.CTkLabel(frame_vtime_mode, text="WAKTU OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02064 |
02065 |         self.radio_vis_time_mode = ctk.StringVar(value="maghrib")
02066 |
02067 |         def on_vis_time_mode_change():
02068 |             if self.radio_vis_time_mode.get() == "kustom":
02069 |                 self.entry_vtime.pack(fill="x", padx=10, pady=(0, 10))
02070 |             else:
02071 |                 self.entry_vtime.pack_forget()
02072 |
02073 |         ctk.CTkRadioButton(frame_vtime_mode, text="1) Waktu Maghrib (Sunset)", variable=self.radio_vis_time_mode, value="maghrib", command=on_vis_time_mode_change).pack(anchor="w", padx=15, pady=5)
02074 |         ctk.CTkRadioButton(frame_vtime_mode, text="2) Waktu Tertentu (Kustom)", variable=self.radio_vis_time_mode, value="kustom", command=on_vis_time_mode_change).pack(anchor="w", padx=15, pady=5)
02075 |         ctk.CTkRadioButton(frame_vtime_mode, text="3) Waktu Live (Sekarang)", variable=self.radio_vis_time_mode, value="live", command=on_vis_time_mode_change).pack(anchor="w", padx=15, pady=(5, 10))
02076 |
02077 |         self.entry_vtime = ctk.CTkEntry(frame_vtime_mode, placeholder_text="HH:MM:SS (Contoh: 17:45:00)", justify="center")
02078 |         # Catatan: entry_vtime disembunyikan secara default karena mode awalnya "maghrib"
02079 |         # ---> AKHIR TAMBAHAN UI <---
02080 |
02081 |         frame_vloc = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
02082 |         frame_vloc.pack(fill="x", padx=15, pady=5)
02083 |         ctk.CTkLabel(frame_vloc, text="KOORDINAT LOKASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02084 |         self.entry_vlat = self.create_input_row(frame_vloc, "Lat (Lintang):", "-7.0667")
02085 |         self.entry_vlon = self.create_input_row(frame_vloc, "Lon (Bujur):", "110.4100")
02086 |         self.entry_velev = self.create_input_row(frame_vloc, "Elevasi (m):", "230.0")
02087 |         self.entry_vtz = self.create_input_row(frame_vloc, "Timezone:", "7.0")
02088 |
02089 |         frame_vatm = ctk.CTkFrame(self.frame_vis_inputs, fg_color="#212121")
02090 |         frame_vatm.pack(fill="x", padx=15, pady=5)
02091 |         ctk.CTkLabel(frame_vatm, text="PARAMETER ATMOSFER", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02092 |         self.entry_vtemp = self.create_input_row(frame_vatm, "Suhu (°C):", "10.0")
02093 |         self.entry_vpres = self.create_input_row(frame_vatm, "Tekanan (mb):", "1010.0")
02094 |         self.entry_vhum = self.create_input_row(frame_vatm, "Kelembapan (%):", "60.0")
02095 |
02096 |         # --- CONT 2 SD 18 (SAMA SEPERTI MASTER, DI RINGKAS UNTUK TEMPAT, TAPI DI-INCLUDE DI BAWAH) ---
02097 |         self._setup_other_inputs(curr_y, curr_m, curr_d, curr_m_np, curr_d_np)
02098 |
02099 |         # --- CONTAINER 19: 3D EPHEMERIS (SCRIPT A) INPUTS ---
02100 |         self.frame_eph3d_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02101 |
02102 |         # Lokasi khusus (Terhubung ke update on_city_change)
02103 |         frame_eph3d_loc = ctk.CTkFrame(self.frame_eph3d_inputs, fg_color="#212121")
02104 |         frame_eph3d_loc.pack(fill="x", padx=15, pady=5)
02105 |         ctk.CTkLabel(frame_eph3d_loc, text="KOORDINAT OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02106 |         self.entry_eph3d_lat = self.create_input_row(frame_eph3d_loc, "Lat (Lintang):", "-7.0667")
02107 |         self.entry_eph3d_lon = self.create_input_row(frame_eph3d_loc, "Lon (Bujur):", "110.4100")
02108 |         self.entry_eph3d_tz = self.create_input_row(frame_eph3d_loc, "Timezone:", "7.0") # <--- TAMBAHKAN BARIS INI
02109 |
02110 |         frame_eph3d_waktu = ctk.CTkFrame(self.frame_eph3d_inputs, fg_color="#212121")
02111 |         frame_eph3d_waktu.pack(fill="x", padx=15, pady=5)
02112 |         ctk.CTkLabel(frame_eph3d_waktu, text="PENGATURAN WAKTU & ELEVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02113 |
02114 |         self.var_eph3d_tahun, self.lbl_eph3d_tahun = self.create_eph3d_slider(frame_eph3d_waktu, "Tahun", -3000, 3000, int(curr_y), 1)
02115 |         self.var_eph3d_bulan, self.lbl_eph3d_bulan = self.create_eph3d_slider(frame_eph3d_waktu, "Bulan", 1, 12, int(curr_m_np), 1)
02116 |         self.var_eph3d_hari, self.lbl_eph3d_hari = self.create_eph3d_slider(frame_eph3d_waktu, "Tanggal", 1, 31, int(curr_d_np), 1)
02117 |         self.var_eph3d_jam, self.lbl_eph3d_jam = self.create_eph3d_slider(frame_eph3d_waktu, "Jam (Lokal)", 0, 23.99, 12.0, 0.01)
02118 |         self.var_eph3d_elev, self.lbl_eph3d_elev = self.create_eph3d_slider(frame_eph3d_waktu, "Elevasi (m)", 0, 5000, 230, 1)
02119 |
02120 |         frame_eph3d_btn = ctk.CTkFrame(self.frame_eph3d_inputs, fg_color="transparent")
02121 |         frame_eph3d_btn.pack(fill="x", padx=15, pady=10)
02122 |
02123 |         btn_eph3d_sunset = ctk.CTkButton(frame_eph3d_btn, text="Cari Sunset", width=120, fg_color="#C62828", hover_color="#B71C1C", command=self.eph3d_cari_sunset)
02124 |         btn_eph3d_sunset.pack(side="left", fill="x", expand=True, padx=(0,5))
02125 |
02126 |         btn_eph3d_live = ctk.CTkButton(frame_eph3d_btn, text="□ Waktu Live", width=120, fg_color="#2E7D32", hover_color="#1B5E20", command=self.eph3d_start_live)
02127 |         btn_eph3d_live.pack(side="left", fill="x", expand=True, padx=(5,0))
02128 |

```

```

02129 | # -- MIZWALA / TONGKAT ISTIWA INPUTS (MENU 29) --
02130 | self.frame_mizwala_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02131 |
02132 | frame_miz_date = ctk.CTkFrame(self.frame_mizwala_inputs, fg_color="#212121")
02133 | frame_miz_date.pack(fill="x", padx=15, pady=5)
02134 | ctk.CTkLabel(frame_miz_date, text="TANGGAL OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02135 | self.entry_miz_year, self.entry_miz_month, self.entry_miz_day = self.create_ymd_row(frame_miz_date, curr_y, curr_m_np, curr_d_np)
02136 |
02137 | frame_miz_loc = ctk.CTkFrame(self.frame_mizwala_inputs, fg_color="#212121")
02138 | frame_miz_loc.pack(fill="x", padx=15, pady=5)
02139 | ctk.CTkLabel(frame_miz_loc, text="LOKASI OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02140 | self.entry_miz_lat = self.create_input_row(frame_miz_loc, "Lat (Lintang):", "-7.0667")
02141 | self.entry_miz_lon = self.create_input_row(frame_miz_loc, "Lon (Bujur):", "110.4100")
02142 | self.entry_miz_elev = self.create_input_row(frame_miz_loc, "Elevasi (m):", "230.0")
02143 | self.entry_miz_tz = self.create_input_row(frame_miz_loc, "Timezone:", "7.0")
02144 |
02145 | frame_miz_opt = ctk.CTkFrame(self.frame_mizwala_inputs, fg_color="#212121")
02146 | frame_miz_opt.pack(fill="x", padx=15, pady=5)
02147 | ctk.CTkLabel(frame_miz_opt, text="PARAMETER TONGKAT & WAKTU", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02148 | self.entry_miz_tinggi = self.create_input_row(frame_miz_opt, "Tinggi (cm):", "100.0")
02149 |
02150 | # ---> TAMBAHKAN BARIS INI UNTUK INPUT WAKTU SIMULASI <---
02151 | self.entry_miz_waktu = self.create_input_row(frame_miz_opt, "Waktu (HH:MM):", "LIVE")
02152 |
02153 | # Tombol Pemicu Simulasi Mizwala
02154 | self.btn_simulasi_miz = ctk.CTkButton(
02155 |     self.frame_mizwala_inputs,
02156 |     text="🔍 Buka Simulasi Visual Mizwala",
02157 |     font=("Segoe UI", 12, "bold"),
02158 |     fg_color="#F57C00",
02159 |     hover_color="#E65100",
02160 |     command=self.buka_simulasi_mizwala
02161 | )
02162 | self.btn_simulasi_miz.pack(fill="x", padx=15, pady=(10, 5))
02163 | self.entry_miz_step = self.create_input_row(frame_miz_opt, "Interval (Menit):", "10")
02164 |
02165 | # -- AUTO BUILDER DATABASE HIJRIAH (MENU 30) --
02166 | self.frame_autobuild_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02167 |
02168 | frame_ab_cfg = ctk.CTkFrame(self.frame_autobuild_inputs, fg_color="#212121", border_width=1, border_color="#F57C00")
02169 | frame_ab_cfg.pack(fill="x", padx=15, pady=10)
02170 | ctk.CTkLabel(frame_ab_cfg, text="📅 GENERATOR DATABASE KHGT", font=("Segoe UI", 12, "bold"), text_color="#FFD54F").pack(anchor="w", padx=10, pady=(8, 2))
02171 |
02172 | self.entry_ab_start = self.create_input_row(frame_ab_cfg, "Tahun Awal (H):", "1446")
02173 | self.entry_ab_end = self.create_input_row(frame_ab_cfg, "Tahun Akhir (H):", "1546")
02174 |
02175 | # -- MENU 31: STATUS KRITERIA BATAS --
02176 | self.frame_kriteria_batas_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02177 |
02178 | # 1. Input Tanggal Observasi
02179 | frame_kb_date = ctk.CTkFrame(self.frame_kriteria_batas_inputs, fg_color="#212121")
02180 | frame_kb_date.pack(fill="x", padx=15, pady=5)
02181 | ctk.CTkLabel(frame_kb_date, text="TANGGAL OBSERVASI (SAAT MAGHRIH)", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02182 | self.entry_kb_year, self.entry_kb_month, self.entry_kb_day = self.create_ymd_row(frame_kb_date, curr_y, curr_m_np, curr_d_np)
02183 |
02184 | # 2. Input Lokasi Observasi
02185 | frame_kb_loc = ctk.CTkFrame(self.frame_kriteria_batas_inputs, fg_color="#212121")
02186 | frame_kb_loc.pack(fill="x", padx=15, pady=5)
02187 | ctk.CTkLabel(frame_kb_loc, text="KOORDINAT LOKASI", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02188 | self.entry_kb_lat = self.create_input_row(frame_kb_loc, "Lat (Lintang):", "-7.0667")
02189 | self.entry_kb_lon = self.create_input_row(frame_kb_loc, "Lon (Bujur):", "110.4100")
02190 | self.entry_kb_elev = self.create_input_row(frame_kb_loc, "Elevasi (m):", "230.0")
02191 | self.entry_kb_tz = self.create_input_row(frame_kb_loc, "Timezone:", "7.0")
02192 |
02193 | # 3. Slider Manual (Jika ingin simulasi sendiri setelah perhitungan)
02194 | frame_kb_opt = ctk.CTkFrame(self.frame_kriteria_batas_inputs, fg_color="#212121")
02195 | frame_kb_opt.pack(fill="x", padx=15, pady=10)
02196 | ctk.CTkLabel(frame_kb_opt, text="SIMULATOR PARAMETER (MANUAL)", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02197 |
02198 | self.var_kb_elongasi, self.lbl_kb_elongasi = self.create_eph3d_slider(frame_kb_opt, "Elongasi", 0.0, 15.0, 8.5, 0.1)
02199 | self.var_kb_ketinggian, self.lbl_kb_ketinggian = self.create_eph3d_slider(frame_kb_opt, "Ketinggian", -5.0, 15.0, 6.0, 0.1)

```

```

02200 |
02201 | # Trigger update grafik otomatis saat slider digeser
02202 | self.var_kb_elongasi.trace_add('write', lambda *args: self.update_kriteria_batas_plot())
02203 | self.var_kb_ketinggian.trace_add('write', lambda *args: self.update_kriteria_batas_plot())
02204 |
02205 | # =====
02206 | # MODUL 31 - 34: FRAME INPUT (SIDEBAR)
02207 | # =====
02208 | # -- 31. ASTROFOTOGRAFI --
02209 | self.frame_astro_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02210 | frame_astro_date = ctk.CTkFrame(self.frame_astro_inputs, fg_color="#212121")
02211 | frame_astro_date.pack(fill="x", padx=15, pady=5)
02212 | ctk.CTkLabel(frame_astro_date, text="TANGGAL OBSERVASI", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02213 | self.entry_astro_year, self.entry_astro_month, self.entry_astro_day = self.create_ymd_row(frame_astro_date, curr_y, curr_m_np, curr_d_np)
02214 |
02215 | frame_astro_loc = ctk.CTkFrame(self.frame_astro_inputs, fg_color="#212121")
02216 | frame_astro_loc.pack(fill="x", padx=15, pady=5)
02217 | ctk.CTkLabel(frame_astro_loc, text="KOORDINAT LOKASI", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02218 | self.entry_astro_lat = self.create_input_row(frame_astro_loc, "Lat (Lintang):", "-7.0667")
02219 | self.entry_astro_lon = self.create_input_row(frame_astro_loc, "Lon (Bujur):", "110.4100")
02220 | self.entry_astro_elev = self.create_input_row(frame_astro_loc, "Elevasi (m):", "230.0")
02221 | self.entry_astro_tz = self.create_input_row(frame_astro_loc, "Timezone:", "7.0")
02222 |
02223 | # ---> TAMBAHAN LABEL CUACA 31 <---
02224 | self.lbl_cuaca_astro = ctk.CTkLabel(frame_astro_loc, text="Cuaca: Menunggu...", font=("Consolas", 10, "bold"), text_color="#9E9E9E")
02225 | self.lbl_cuaca_astro.pack(anchor="w", padx=10, pady=(0, 5))
02226 |
02227 | # ---> TAMBAHAN LABEL CUACA 31 <---
02228 | self.lbl_cuaca_astro = ctk.CTkLabel(frame_astro_loc, text="Cuaca: Menunggu...", font=("Consolas", 10, "bold"), text_color="#9E9E9E")
02229 | self.lbl_cuaca_astro.pack(anchor="w", padx=10, pady=(0, 5))
02230 |
02231 | # =====
02232 | # ---> SISIPAN BARU: UI PREDIKTOR OKULTASI (ASTROFOTOGRAFI)
02233 | # =====
02234 | frame_astro_occ = ctk.CTkFrame(self.frame_astro_inputs, fg_color="#212121")
02235 | frame_astro_occ.pack(fill="x", padx=15, pady=5)
02236 | ctk.CTkLabel(frame_astro_occ, text="PREDIKTOR OKULTASI PLANET", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02237 |
02238 | self.combo_astro_occ_target = ctk.CTkOptionMenu(frame_astro_occ, values=["Venus", "Mars", "Jupiter", "Saturn", "Mercury", "Uranus", "Neptune"])
02239 | self.combo_astro_occ_target.pack(fill="x", padx=10, pady=(0, 10))
02240 |
02241 | self.btn_cari_occ = ctk.CTkButton(frame_astro_occ, text="Cari Okultasi Bulan Ini", font=("Segoe UI", 12, "bold"), fg_color="#673AB7", hover_color="#512DA8", command=self.calculate_occultation)
02242 | self.btn_cari_occ.pack(fill="x", padx=10, pady=(0, 10))
02243 | # =====
02244 |
02245 | # =====
02246 | # -- 32. PASANG SURUT GRAVITASI --
02247 | # =====
02248 | self.frame_pasang_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02249 |
02250 | frame_pasang_date = ctk.CTkFrame(self.frame_pasang_inputs, fg_color="#212121")
02251 | frame_pasang_date.pack(fill="x", padx=15, pady=5)
02252 | ctk.CTkLabel(frame_pasang_date, text="TANGGAL OBSERVASI", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02253 | self.entry_pasang_year, self.entry_pasang_month, self.entry_pasang_day = self.create_ymd_row(frame_pasang_date, curr_y, curr_m_np, curr_d_np)
02254 |
02255 | frame_pasang_loc = ctk.CTkFrame(self.frame_pasang_inputs, fg_color="#212121")
02256 | frame_pasang_loc.pack(fill="x", padx=15, pady=5)
02257 | ctk.CTkLabel(frame_pasang_loc, text="KOORDINAT PANTAI", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02258 | self.entry_pasang_lat = self.create_input_row(frame_pasang_loc, "Lat (Lintang):", "-7.0667")
02259 | self.entry_pasang_lon = self.create_input_row(frame_pasang_loc, "Lon (Bujur):", "110.4100")
02260 | self.entry_pasang_tz = self.create_input_row(frame_pasang_loc, "Timezone:", "7.0")
02261 |
02262 | # ---> TAMBAHAN LABEL CUACA 32 <---
02263 | self.lbl_cuaca_pasang = ctk.CTkLabel(frame_pasang_loc, text="Cuaca: Menunggu...", font=("Consolas", 10, "bold"), text_color="#9E9E9E")
02264 | self.lbl_cuaca_pasang.pack(anchor="w", padx=10, pady=(0, 5))
02265 |
02266 | # =====
02267 | # -- 33. EVALUASI FAJAR SHADIQ & SQM --
02268 | # =====
02269 | self.frame_fajar_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02270 |

```

```

02271 |         frame_fajar_date = ctk.CTkFrame(self.frame_fajar_inputs, fg_color="#212121")
02272 |         frame_fajar_date.pack(fill="x", padx=15, pady=5)
02273 |         ctk.CTkLabel(frame_fajar_date, text="TANGGAL OBSERVASI", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02274 |         self.entry_fajar_year, self.entry_fajar_month, self.entry_fajar_day = self.create_ymd_row(frame_fajar_date, curr_y, curr_m_np, curr_d_np)
02275 |
02276 |         frame_fajar_loc = ctk.CTkFrame(self.frame_fajar_inputs, fg_color="#212121")
02277 |         frame_fajar_loc.pack(fill="x", padx=15, pady=5)
02278 |         ctk.CTkLabel(frame_fajar_loc, text="KOORDINAT LOKASI", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02279 |         self.entry_fajar_lat = self.create_input_row(frame_fajar_loc, "Lat (Lintang):", "-7.0667")
02280 |         self.entry_fajar_lon = self.create_input_row(frame_fajar_loc, "Lon (Bujur):", "110.4100")
02281 |         self.entry_fajar_tz = self.create_input_row(frame_fajar_loc, "Timezone:", "7.0")
02282 |
02283 |         # ---> TAMBAHAN LABEL CUACA 33 <---
02284 |         self.lbl_cuaca_fajar = ctk.CTkLabel(frame_fajar_loc, text="Cuaca: Menunggu...", font=("Consolas", 10, "bold"), text_color="#9E9E9E")
02285 |         self.lbl_cuaca_fajar.pack(anchor="w", padx=10, pady=(0, 5))
02286 |
02287 |         # -- 34. ANALEMMA MATAHARI --
02288 |         self.frame_analemma_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02289 |
02290 |         frame_analemma_date = ctk.CTkFrame(self.frame_analemma_inputs, fg_color="#212121")
02291 |         frame_analemma_date.pack(fill="x", padx=15, pady=5)
02292 |         ctk.CTkLabel(frame_analemma_date, text="TAHUN & JAM ANALISIS", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02293 |
02294 |         # Baris Tahun
02295 |         self.entry_analemma_year = self.create_input_row(frame_analemma_date, "Tahun (YYYY):", curr_y)
02296 |         # Baris Jam (Lokal)
02297 |         self.entry_analemma_hour = self.create_input_row(frame_analemma_date, "Jam (0-23):", "12.0")
02298 |
02299 |         frame_analemma_loc = ctk.CTkFrame(self.frame_analemma_inputs, fg_color="#212121")
02300 |         frame_analemma_loc.pack(fill="x", padx=15, pady=5)
02301 |         ctk.CTkLabel(frame_analemma_loc, text="KOORDINAT PENGAMAT", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02302 |         self.entry_analemma_lat = self.create_input_row(frame_analemma_loc, "Lat (Lintang):", "-7.0667")
02303 |         self.entry_analemma_lon = self.create_input_row(frame_analemma_loc, "Lon (Bujur):", "110.4100")
02304 |         self.entry_analemma_tz = self.create_input_row(frame_analemma_loc, "Timezone:", "7.0")
02305 |
02306 |         # ---> LABEL CUACA MODUL 34 (Wajib Ada) <---
02307 |         self.lbl_cuaca_analemma = ctk.CTkLabel(frame_analemma_loc, text="Cuaca: Menunggu...", font=("Consolas", 10, "bold"), text_color="#9E9E9E")
02308 |         self.lbl_cuaca_analemma.pack(anchor="w", padx=10, pady=(0, 5))
02309 |         # =====
02310 |
02311 |         # --- Tombol Aksi Utama ---
02312 |         self.btn_hitung = ctk.CTkButton(self.sidebar, text="► PROSES DATA", font=("Segoe UI", 14, "bold"), height=45, fg_color="#1565C0", hover_color="#0D47A1", command=self.run_calculation,
state="disabled")
02313 |         self.btn_hitung.pack(fill="x", padx=15, pady=(20, 10))
02314 |
02315 |         self.lbl_status = ctk.CTkLabel(self.sidebar, text="Memuat Ephemeris...", font=("Consolas", 11), text_color="#FFAB40")
02316 |         self.lbl_status.pack(pady=5)
02317 |
02318 |         # ===== MAIN AREA (KANAN) =====
02319 |         self.main_frame = ctk.CTkFrame(self, fg_color="transparent")
02320 |         self.main_frame.grid(row=0, column=1, sticky="nsew", padx=20, pady=20)
02321 |         self.main_frame.grid_rowconfigure(1, weight=1)
02322 |         self.main_frame.grid_columnconfigure(0, weight=1)
02323 |
02324 |         header_frame = ctk.CTkFrame(self.main_frame, fg_color="transparent")
02325 |         header_frame.grid(row=0, column=0, sticky="ew", pady=(0, 10))
02326 |         header_frame.grid_columnconfigure(0, weight=1)
02327 |         header_frame.grid_columnconfigure(1, weight=1)
02328 |         header_frame.grid_columnconfigure(2, weight=1)
02329 |
02330 |         title_frame = ctk.CTkFrame(header_frame, fg_color="transparent")
02331 |         title_frame.grid(row=0, column=0, sticky="w")
02332 |
02333 |         self.btn_toggle = ctk.CTkButton(title_frame, text="☐", font=("Segoe UI", 22), width=40, height=40, fg_color="#1F1F1F", hover_color="#333333", text_color="#00E5FF", command=self.toggle_sidebar)
02334 |         self.btn_toggle.pack(side="left", padx=(0, 15))
02335 |
02336 |         title_text_frame = ctk.CTkFrame(title_frame, fg_color="transparent")
02337 |         title_text_frame.pack(side="left", fill="y")
02338 |
02339 |         self.lbl_main_title = ctk.CTkLabel(title_text_frame, text="Laporan Analisis Hilal & KHGT", font=("Segoe UI", 20, "bold"))
02340 |         self.lbl_main_title.pack(anchor="w")

```

```

02341 |
02342 | self.lbl_countdown = ctk.CTkLabel(title_text_frame, text="Alarm Salat: Memuat Jadwal...", font=("Consolas", 12, "bold"), text_color="#FFAB40")
02343 | self.lbl_countdown.pack(anchor="w", pady=(2, 0))
02344 |
02345 | # ---> TAMBAHAN WIDGET CUACA <---
02346 | self.lbl_status_cuaca = ctk.CTkLabel(title_text_frame, text="Cuaca: Menunggu deteksi...", font=("Consolas", 11, "bold"), text_color="#9E9E9E")
02347 | self.lbl_status_cuaca.pack(anchor="w", pady=(2, 0))
02348 | # -----
02349 |
02350 | self.f_cal = ctk.CTkFrame(header_frame, fg_color="#1A1A1A", corner_radius=8, border_width=1, border_color="#333")
02351 | self.f_cal.grid(row=0, column=1, sticky="n")
02352 |
02353 | self.lbl_cal_masehi = ctk.CTkLabel(self.f_cal, text="", font=("Segoe UI", 13, "bold"), text_color="#00E5FF")
02354 | self.lbl_cal_masehi.pack(side="left", padx=(15, 5), pady=4)
02355 |
02356 | lbl_sep = ctk.CTkLabel(self.f_cal, text="|", font=("Segoe UI", 13, "bold"), text_color="#555")
02357 | lbl_sep.pack(side="left", pady=4)
02358 |
02359 | self.lbl_cal_hijri = ctk.CTkLabel(self.f_cal, text="", font=("Georgia", 14, "italic"), text_color="#FFD54F")
02360 | self.lbl_cal_hijri.pack(side="left", padx=(5, 15), pady=4)
02361 |
02362 | btn_frame = ctk.CTkFrame(header_frame, fg_color="transparent")
02363 | btn_frame.grid(row=0, column=2, sticky="e")
02364 |
02365 | self.btn_simpan = ctk.CTkButton(btn_frame, text="□", font=("Segoe UI", 18), width=35, height=35, fg_color="#2E7D32", hover_color="#1B5E20", command=self.save_to_txt)
02366 | self.btn_simpan.pack(side="left", padx=(0, 8))
02367 | CTKToolTip(self.btn_simpan, delay=0.5, message="Simpan Data ke TXT")
02368 |
02369 | self.btn_png = ctk.CTkButton(btn_frame, text="□", font=("Segoe UI", 16), width=35, height=35, fg_color="#F57C00", hover_color="#EF6C00", command=self.export_to_png)
02370 | self.btn_png.pack(side="left", padx=(0, 8))
02371 | CTKToolTip(self.btn_png, delay=0.5, message="Simpan Laporan sbg PNG")
02372 |
02373 | self.btn_pdf = ctk.CTkButton(btn_frame, text="□", font=("Segoe UI", 16), width=35, height=35, fg_color="#1976D2", hover_color="#1565C0", command=self.export_to_pdf)
02374 | self.btn_pdf.pack(side="left", padx=(0, 8))
02375 | CTKToolTip(self.btn_pdf, delay=0.5, message="Simpan sebagai PDF")
02376 |
02377 | # ---> SISIPKAN KODE TOMBOL CSV DI SINI <---
02378 | self.btn_csv = ctk.CTkButton(btn_frame, text="□", font=("Segoe UI", 16), width=35, height=35, fg_color="#00695C", hover_color="#004D40", command=self.export_to_csv)
02379 | self.btn_csv.pack(side="left", padx=(0, 8))
02380 | CTKToolTip(self.btn_csv, delay=0.5, message="Simpan ke CSV (Excel / SPSS)")
02381 | # -----
02382 |
02383 | self.btn_home = ctk.CTkButton(btn_frame, text="□", font=("Segoe UI", 18), width=35, height=35, fg_color="#E65100", hover_color="#BF360C", command=self.show_release_info)
02384 | self.btn_home.pack(side="left", padx=(0, 8))
02385 | CTKToolTip(self.btn_home, delay=0.5, message="Kembali ke Beranda / Info Rilis")
02386 |
02387 | # Output Container Utama (Text)
02388 | self.textbox = ctk.CTkTextbox(self.main_frame, font=("Consolas", 13), fg_color="#101010", text_color="#00E5FF", wrap="none")
02389 | self.textbox.grid(row=1, column=0, sticky="nsew")
02390 | self.textbox.insert("1.0", "Menunggu input dari pengguna...")
02391 | self.textbox.configure(state="disabled")
02392 | self.install_textbox_language_hooks()
02393 |
02394 | self.setup_gerhana_out_frame()
02395 | self.setup_animasi_out_frame()
02396 | self.setup_3d_out_frame()
02397 |
02398 | self.setup_kriteria_batas_out_frame()
02399 | self.setup_analemma_out_frame()
02400 | self.setup_kurva_alaska_out_frame() # <--- TAMBAHKAN BARIS INI
02401 |
02402 | # ---> TAMBAHKAN 1 BARIS INI (RUMAH KHUSUS UNTUK GRAFIK CHART) <---
02403 | self.frame_chart_out = ctk.CTkFrame(self.main_frame, fg_color="transparent")
02404 |
02405 | # =====
02406 | # 1. PINDAHKAN INISIALISASI KALENDER KE SINI
02407 | # (Wajib dibuat sebelum switch_mode dipanggil!)
02408 | # =====
02409 |
02410 | # -- KALENDER HIJRIAH INPUTS (MENU 26) --
02411 | self.frame_kalender_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")

```

```

02412 |         kal_info = ctk.CTkFrame(self.frame_kalender_inputs, fg_color="#212121")
02413 |         kal_info.pack(fill="x", padx=15, pady=10)
02414 |         ctk.CTkLabel(kal_info, text="PENGATURAN KALENDER", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02415 |
02416 |         # Tambahan Dropdown Bulan dan Entry Tahun (Hijriah)
02417 |         row_h_bln = ctk.CTkFrame(kal_info, fg_color="transparent")
02418 |         row_h_bln.pack(fill="x", padx=10, pady=5)
02419 |         ctk.CTkLabel(row_h_bln, text="Bulan:", font=("Segoe UI", 12)).pack(side="left")
02420 |         self.combo_kal_h_bulan = ctk.CTkComboBox(row_h_bln, values=BULAN_HIJRIAH, width=120)
02421 |         self.combo_kal_h_bulan.pack(side="right")
02422 |
02423 |         row_h_thn = ctk.CTkFrame(kal_info, fg_color="transparent")
02424 |         row_h_thn.pack(fill="x", padx=10, pady=5)
02425 |         ctk.CTkLabel(row_h_thn, text="Tahun (H):", font=("Segoe UI", 12)).pack(side="left")
02426 |         self.entry_kal_h_tahun = ctk.CTkEntry(row_h_thn, width=80, justify="center")
02427 |         self.entry_kal_h_tahun.pack(side="right")
02428 |
02429 |         ctk.CTkButton(kal_info, text="Terapkan Pencarian", fg_color="#1565C0", hover_color="#0D47A1", command=self.apply_kalender_kustom).pack(fill="x", padx=10, pady=(10, 5))
02430 |         ctk.CTkButton(kal_info, text="Reset ke Bulan Ini", fg_color="#00695C", hover_color="#004D40", command=self.reset_kalender).pack(fill="x", padx=10, pady=(0, 10))
02431 |
02432 |         # -- KALENDER MASEHI INPUTS (MENU 27) --
02433 |         self.frame_kalmasehi_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02434 |         kalm_info = ctk.CTkFrame(self.frame_kalmasehi_inputs, fg_color="#212121")
02435 |         kalm_info.pack(fill="x", padx=15, pady=10)
02436 |         ctk.CTkLabel(kalm_info, text="PENGATURAN KAL. MASEHI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02437 |
02438 |         # Tambahan Dropdown Bulan dan Entry Tahun (Masehi)
02439 |         row_m_bln = ctk.CTkFrame(kalm_info, fg_color="transparent")
02440 |         row_m_bln.pack(fill="x", padx=10, pady=5)
02441 |         ctk.CTkLabel(row_m_bln, text="Bulan:", font=("Segoe UI", 12)).pack(side="left")
02442 |         self.combo_kal_m_bulan = ctk.CTkComboBox(row_m_bln, values=BULAN_MASEHI, width=120)
02443 |         self.combo_kal_m_bulan.pack(side="right")
02444 |
02445 |         row_m_thn = ctk.CTkFrame(kalm_info, fg_color="transparent")
02446 |         row_m_thn.pack(fill="x", padx=10, pady=5)
02447 |         ctk.CTkLabel(row_m_thn, text="Tahun (M):", font=("Segoe UI", 12)).pack(side="left")
02448 |         self.entry_kal_m_tahun = ctk.CTkEntry(row_m_thn, width=80, justify="center")
02449 |         self.entry_kal_m_tahun.pack(side="right")
02450 |
02451 |         ctk.CTkButton(kalm_info, text="Terapkan Pencarian", fg_color="#1565C0", hover_color="#0D47A1", command=self.apply_kalmasehi_kustom).pack(fill="x", padx=10, pady=(10, 5))
02452 |         ctk.CTkButton(kalm_info, text="Reset ke Bulan Ini", fg_color="#00695C", hover_color="#004D40", command=self.reset_kalmasehi).pack(fill="x", padx=10, pady=(0, 10))
02453 |
02454 |         # Render dan Siapkan Frame Output Kalender
02455 |         self.reset_kalender(update_ui=False)
02456 |         self.setup_kalender_out_frame()
02457 |         self.reset_kalmasehi(update_ui=False)
02458 |         self.setup_kalmasehi_out_frame()
02459 |
02460 |         # Setup Frame Output Mode 19 (Matplotlib 3D Ephemeris)
02461 |         self.setup_eph3d_out_frame()
02462 |         # =====
02463 |
02464 |         # ---> TAMBAHKAN SETUP WINAI DI SINI <---
02465 |         self.setup_winai_ui()
02466 |         # =====
02467 |
02468 |         # 2. BARU PANGGIL SWITCH MODE & STARTUP LAINNYA DI SINI
02469 |         on_menu_click("1) Visibility Hilal (KHGT)")
02470 |         self.show_release_info()
02471 |         # Simpan teks asli seluruh widget sidebar setelah semua frame input dibuat.
02472 |         # Ini membuat klik ID/EN/AR setelah memilih menu 1-41 selalu dapat
02473 |         # menerjemahkan sidebar kiri dari sumber Indonesia asli, bukan dari teks
02474 |         # yang sedang tampil atau dari hasil terjemahan sebelumnya.
02475 |         self.capture_sidebar_original_texts()
02476 |         self.force_update_sidebar_language()
02477 |         self.update_ui_texts(self)
02478 |         self.install_secondary_output_textbox_hooks()
02479 |         self.apply_language_direction()
02480 |         # Startup tetap menampilkan INFORMASI RILIS. Karena pengguna belum
02481 |         # memilih menu 1-41 secara manual, klik ID/EN/AR pada kondisi awal
02482 |         # harus hanya menerjemahkan ruang kanan dari cache JSON rilis.

```

```

02483 |         self.user_has_selected_main_menu = False
02484 |
02485 |         self.sidebar_visible = False
02486 |         self.sidebar.grid_remove()
02487 |
02488 |         self.anim_running = False
02489 |         self.is_viewing_3d = False
02490 |
02491 |         # ---> TAMBAHKAN BARIS INI UNTUK MEMICU CUACA SAAT STARTUP <---
02492 |         self.on_city_change(self.opt_city.get())
02493 |         # -----
02494 |
02495 |     def setup_other_inputs(self, curr_y, curr_m, curr_d, curr_m_np, curr_d_np):
02496 |         # Meringkas pembuatan UI master yang panjang untuk mode 2 sampai 18
02497 |         # -- MOONPHASE --
02498 |         self.frame_moon_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02499 |         frame_myear = ctk.CTkFrame(self.frame_moon_inputs, fg_color="#212121")
02500 |         frame_myear.pack(fill="x", padx=15, pady=10)
02501 |         ctk.CTkLabel(frame_myear, text="TAHUN MASEHI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(10, 5))
02502 |         self.entry_moon_year = ctk.CTkEntry(frame_myear, width=100, justify="center")
02503 |         self.entry_moon_year.insert(0, curr_y)
02504 |         self.entry_moon_year.pack(padx=10, pady=(0, 15))
02505 |
02506 |         # -- EPHEMERIS --
02507 |         self.frame_eph_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02508 |         frame_eph_cfg = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
02509 |         frame_eph_cfg.pack(fill="x", padx=15, pady=5)
02510 |         row_obj = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
02511 |         row_obj.pack(fill="x", padx=10, pady=2)
02512 |         ctk.CTkLabel(row_obj, text="Object:", font=("Segoe UI", 12)).pack(side="left")
02513 |         self.combo_eph_obj = ctk.CTkOptionMenu(row_obj, values=["Sun", "Moon"], width=100)
02514 |         self.combo_eph_obj.pack(side="right")
02515 |         row_tref = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
02516 |         row_tref.pack(fill="x", padx=10, pady=2)
02517 |         ctk.CTkLabel(row_tref, text="Time Ref:", font=("Segoe UI", 12)).pack(side="left")
02518 |         self.combo_eph_tref = ctk.CTkOptionMenu(row_tref, values=["Local (UT1)", "Local (TDT)"], width=100)
02519 |         self.combo_eph_tref.pack(side="right")
02520 |         row_lref = ctk.CTkFrame(frame_eph_cfg, fg_color="transparent")
02521 |         row_lref.pack(fill="x", padx=10, pady=2)
02522 |         ctk.CTkLabel(row_lref, text="Location:", font=("Segoe UI", 12)).pack(side="left")
02523 |         self.combo_eph_lref = ctk.CTkOptionMenu(row_lref, values=["Topocentric", "Geocentric"], width=100)
02524 |         self.combo_eph_lref.pack(side="right")
02525 |
02526 |         # --- WAKTU AWAL (START) ---
02527 |         frame_eph_start = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
02528 |         frame_eph_start.pack(fill="x", padx=15, pady=5)
02529 |         ctk.CTkLabel(frame_eph_start, text="WAKTU AWAL (START)", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02530 |
02531 |         # Grid Tanggal
02532 |         self.entry_eph_sy, self.entry_eph_smon, self.entry_eph_sd = self.create_ymd_row(frame_eph_start, curr_y, curr_m_np, curr_d_np)
02533 |
02534 |         # Grid Waktu (HH:MM:SS)
02535 |         start_hms_grid = ctk.CTkFrame(frame_eph_start, fg_color="transparent")
02536 |         start_hms_grid.pack(fill="x", padx=10, pady=(0, 10))
02537 |
02538 |         ctk.CTkLabel(start_hms_grid, text="hh", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02539 |         ctk.CTkLabel(start_hms_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02540 |         ctk.CTkLabel(start_hms_grid, text="ss", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
02541 |
02542 |         self.entry_eph_sh = ctk.CTkEntry(start_hms_grid, width=45, placeholder_text="hh", justify="center")
02543 |         self.entry_eph_sh.insert(0, "22")
02544 |         self.entry_eph_sh.grid(row=1, column=0, padx=(0, 5))
02545 |
02546 |         self.entry_eph_smin = ctk.CTkEntry(start_hms_grid, width=45, placeholder_text="mm", justify="center")
02547 |         self.entry_eph_smin.insert(0, "0")
02548 |         self.entry_eph_smin.grid(row=1, column=1, padx=5)
02549 |
02550 |         self.entry_eph_ssec = ctk.CTkEntry(start_hms_grid, width=45, placeholder_text="ss", justify="center")
02551 |         self.entry_eph_ssec.insert(0, "0")
02552 |         self.entry_eph_ssec.grid(row=1, column=2, padx=(5, 0))
02553 |

```



```

02554 | # --- WAKTU AKHIR (END) ---
02555 | frame_eph_end = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
02556 | frame_eph_end.pack(fill="x", padx=15, pady=5)
02557 | ctk.CTkLabel(frame_eph_end, text="WAKTU AKHIR (END)", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02558 |
02559 | # Grid Tanggal
02560 | self.entry_eph_ey, self.entry_eph_emon, self.entry_eph_ed = self.create_ymd_row(frame_eph_end, curr_y, curr_m_np, curr_d_np)
02561 |
02562 | # Grid Waktu (HH:MM:SS)
02563 | end_hms_grid = ctk.CTkFrame(frame_eph_end, fg_color="transparent")
02564 | end_hms_grid.pack(fill="x", padx=10, pady=(0, 10))
02565 |
02566 | ctk.CTkLabel(end_hms_grid, text="hh", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02567 | ctk.CTkLabel(end_hms_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02568 | ctk.CTkLabel(end_hms_grid, text="ss", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
02569 |
02570 | self.entry_eph_eh = ctk.CTkEntry(end_hms_grid, width=45, placeholder_text="hh", justify="center")
02571 | self.entry_eph_eh.insert(0, "22")
02572 | self.entry_eph_eh.grid(row=1, column=0, padx=(0, 5))
02573 |
02574 | self.entry_eph_emin = ctk.CTkEntry(end_hms_grid, width=45, placeholder_text="mm", justify="center")
02575 | self.entry_eph_emin.insert(0, "0")
02576 | self.entry_eph_emin.grid(row=1, column=1, padx=5)
02577 |
02578 | self.entry_eph_esec = ctk.CTkEntry(end_hms_grid, width=45, placeholder_text="ss", justify="center")
02579 | self.entry_eph_esec.insert(0, "0")
02580 | self.entry_eph_esec.grid(row=1, column=2, padx=(5, 0))
02581 |
02582 | # --- LOKASI & INTERVAL ---
02583 | frame_eph_step = ctk.CTkFrame(self.frame_eph_inputs, fg_color="#212121")
02584 | frame_eph_step.pack(fill="x", padx=15, pady=5)
02585 |
02586 | row_inc = ctk.CTkFrame(frame_eph_step, fg_color="transparent")
02587 | row_inc.pack(fill="x", padx=10, pady=(10, 2))
02588 | self.entry_eph_step = ctk.CTkEntry(row_inc, width=50, justify="center")
02589 | self.entry_eph_step.insert(0, "1")
02590 | self.entry_eph_step.pack(side="left")
02591 | self.combo_eph_step = ctk.CTkOptionMenu(row_inc, values=["Day", "Hour", "Minute", "Second"], width=80)
02592 | self.combo_eph_step.pack(side="left", padx=5)
02593 |
02594 | self.entry_eph_lat = self.create_input_row(frame_eph_step, "Lat:", "-7.0667")
02595 | self.entry_eph_lon = self.create_input_row(frame_eph_step, "Lon:", "110.4100")
02596 | self.entry_eph_elev = self.create_input_row(frame_eph_step, "Elev:", "230.0")
02597 | self.entry_eph_tz = self.create_input_row(frame_eph_step, "TZ:", "7.0")
02598 |
02599 | # -- QIBLAH DIR --
02600 | self.frame_qiblah_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02601 | frame_qdate = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
02602 | frame_qdate.pack(fill="x", padx=15, pady=5)
02603 | ctk.CTkLabel(frame_qdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02604 |
02605 | # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
02606 | qdate_grid = ctk.CTkFrame(frame_qdate, fg_color="transparent")
02607 | qdate_grid.pack(fill="x", padx=10, pady=(0, 10))
02608 |
02609 | # Baris 0: Label Petunjuk
02610 | ctk.CTkLabel(qdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02611 | ctk.CTkLabel(qdate_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02612 | ctk.CTkLabel(qdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
02613 |
02614 | # Baris 1: Kotak Input
02615 | self.entry_qday = ctk.CTkEntry(qdate_grid, width=45, placeholder_text="dd", justify="center")
02616 | self.entry_qday.insert(0, curr_d)
02617 | self.entry_qday.grid(row=1, column=0, padx=(0, 5))
02618 |
02619 | self.entry_qmonth = ctk.CTkEntry(qdate_grid, width=45, placeholder_text="mm", justify="center")
02620 | self.entry_qmonth.insert(0, curr_m)
02621 | self.entry_qmonth.grid(row=1, column=1, padx=5)
02622 |
02623 | self.entry_qyear = ctk.CTkEntry(qdate_grid, width=70, placeholder_text="yyyy", justify="center")
02624 | self.entry_qyear.insert(0, curr_y)

```

```

02625 |         self.entry_qyear.grid(row=1, column=2, padx=(5, 0))
02626 |
02627 |         frame_qloc = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
02628 |         frame_qloc.pack(fill="x", padx=15, pady=5)
02629 |         self.entry_qlat = self.create_input_row(frame_qloc, "Lat (Lintang):", "-7.0667")
02630 |         self.entry_qlon = self.create_input_row(frame_qloc, "Lon (Bujur):", "110.4100")
02631 |         self.entry_qtz = self.create_input_row(frame_qloc, "Timezone:", "7.0")
02632 |         frame_qmode = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
02633 |         frame_qmode.pack(fill="x", padx=15, pady=5)
02634 |         self.radio_qiblah_var = ctk.StringVar(value="sun")
02635 |         ctk.CTkRadioButton(frame_qmode, text="Sun is at Qiblah direction", variable=self.radio_qiblah_var, value="sun").pack(anchor="w", padx=15, pady=5)
02636 |         ctk.CTkRadioButton(frame_qmode, text="Sun's shadow is at Qiblah", variable=self.radio_qiblah_var, value="shadow").pack(anchor="w", padx=15, pady=(5, 10))
02637 |         self.btn_qmap = ctk.CTkButton(self.frame_qiblah_inputs, text="🗺 Show Qiblah Map", fg_color="#00695C", hover_color="#004D40", command=self.show_qiblah_map)
02638 |         self.btn_qiblat_online = ctk.CTkButton(
02639 |             self.frame_qiblah_inputs,
02640 |             text="🌐 Akses Kiblat Online",
02641 |             fg_color="#1565C0",
02642 |             hover_color="#0D47A1",
02643 |             command=self.open_kiblat_online
02644 |         )
02645 |
02646 | # =====
02647 | # --> SISIPAN BARU: UI CELESTIAL QIBLAH (MENU 11)
02648 | # =====
02649 | frame_cel_q = ctk.CTkFrame(self.frame_qiblah_inputs, fg_color="#212121")
02650 | frame_cel_q.pack(fill="x", padx=15, pady=5)
02651 | ctk.CTkLabel(frame_cel_q, text="🌟 CELESTIAL QIBLAH (BINTANG)", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02652 |
02653 | # Daftar bintang navigasi populer
02654 | self.combo_q_star = ctk.CTkOptionMenu(frame_cel_q, values=[
02655 |     "Canopus (Suhail)", "Sirius (Asy-Syira)", "Rigel (Orion)",
02656 |     "Betelgeuse (Orion)", "Acrux (Crux/Salib Sel.)", "Vega",
02657 |     "Altair", "Antares", "Spica"
02658 | ])
02659 | self.combo_q_star.pack(fill="x", padx=10, pady=(0, 10))
02660 |
02661 | self.btn_calc_cel_q = ctk.CTkButton(frame_cel_q, text="Cari Waktu Bintang di Kiblat",
02662 |                                     font=("Segoe UI", 12, "bold"), fg_color="#673AB7",
02663 |                                     hover_color="#512DA8", command=self.calculate_celestial_qiblah)
02664 | self.btn_calc_cel_q.pack(fill="x", padx=10, pady=(0, 10))
02665 | self.btn_qmap.pack(fill="x", padx=15, pady=(10, 5))
02666 | self.btn_qiblat_online.pack(fill="x", padx=15, pady=(5, 10))
02667 |
02668 | # -- MOON TIMES --
02669 | self.frame_mt_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02670 | frame_mtdate = ctk.CTkFrame(self.frame_mt_inputs, fg_color="#212121")
02671 | frame_mtdate.pack(fill="x", padx=15, pady=5)
02672 | ctk.CTkLabel(frame_mtdate, text="BULAN OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02673 |
02674 | # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
02675 | mt_date_grid = ctk.CTkFrame(frame_mtdate, fg_color="transparent")
02676 | mt_date_grid.pack(fill="x", padx=10, pady=(0, 10))
02677 |
02678 | # Baris 0: Label Petunjuk
02679 | ctk.CTkLabel(mt_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02680 | ctk.CTkLabel(mt_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02681 |
02682 | # Baris 1: Kotak Input (Perhitungan sebulan penuh)
02683 | self.entry_mt_month = ctk.CTkEntry(mt_date_grid, width=45, placeholder_text="mm", justify="center")
02684 | self.entry_mt_month.insert(0, curr_m)
02685 | self.entry_mt_month.grid(row=1, column=0, padx=(0, 5))
02686 |
02687 | self.entry_mt_year = ctk.CTkEntry(mt_date_grid, width=70, placeholder_text="yyyy", justify="center")
02688 | self.entry_mt_year.insert(0, curr_y)
02689 | self.entry_mt_year.grid(row=1, column=1, padx=(5, 0))
02690 |
02691 | frame_mtloc = ctk.CTkFrame(self.frame_mt_inputs, fg_color="#212121")
02692 | frame_mtloc.pack(fill="x", padx=15, pady=5)
02693 | self.entry_mt_lat = self.create_input_row(frame_mtloc, "Lat (Lintang):", "-7.0667")
02694 | self.entry_mt_lon = self.create_input_row(frame_mtloc, "Lon (Bujur):", "110.4100")
02695 | self.entry_mt_elev = self.create_input_row(frame_mtloc, "Elevasi (m):", "230.0")

```

```

02696 |         self.entry_mt_tz = self.create_input_row(frame_mtloc, "Timezone:", "7.0")
02697 |
02698 |         # -- SUN TIMES --
02699 |         self.frame_st_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02700 |         frame_stdate = ctk.CTkFrame(self.frame_st_inputs, fg_color="#212121")
02701 |         frame_stdate.pack(fill="x", padx=15, pady=5)
02702 |         ctk.CTkLabel(frame_stdate, text="BULAN OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02703 |
02704 |         # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
02705 |         st_date_grid = ctk.CTkFrame(frame_stdate, fg_color="transparent")
02706 |         st_date_grid.pack(fill="x", padx=10, pady=(0, 10))
02707 |
02708 |         # Baris 0: Label Petunjuk
02709 |         ctk.CTkLabel(st_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02710 |         ctk.CTkLabel(st_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02711 |
02712 |         # Baris 1: Kotak Input (Perhitungan sebulan penuh)
02713 |         self.entry_st_month = ctk.CTkEntry(st_date_grid, width=45, placeholder_text="mm", justify="center")
02714 |         self.entry_st_month.insert(0, curr_m)
02715 |         self.entry_st_month.grid(row=1, column=0, padx=(0, 5))
02716 |
02717 |         self.entry_st_year = ctk.CTkEntry(st_date_grid, width=70, placeholder_text="yyyy", justify="center")
02718 |         self.entry_st_year.insert(0, curr_y)
02719 |         self.entry_st_year.grid(row=1, column=1, padx=(5, 0))
02720 |
02721 |         frame_stloc = ctk.CTkFrame(self.frame_st_inputs, fg_color="#212121")
02722 |         frame_stloc.pack(fill="x", padx=15, pady=5)
02723 |         self.entry_st_lat = self.create_input_row(frame_stloc, "Lat (Lintang):", "-7.0667")
02724 |         self.entry_st_lon = self.create_input_row(frame_stloc, "Lon (Bujur):", "110.4100")
02725 |         self.entry_st_elev = self.create_input_row(frame_stloc, "Elevasi (m):", "230.0")
02726 |         self.entry_st_tz = self.create_input_row(frame_stloc, "Timezone:", "7.0")
02727 |
02728 |         # -- PRAYER TIMES --
02729 |         self.frame_pt_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02730 |         frame_pt_mode = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
02731 |         frame_pt_mode.pack(fill="x", padx=15, pady=5)
02732 |         self.combo_pt_mode = ctk.CTkOptionMenu(frame_pt_mode, values=["Bulanan (1 Bulan Penuh)", "Harian (1 Hari)"])
02733 |         self.combo_pt_mode.pack(fill="x", padx=10, pady=(0, 10))
02734 |
02735 |         frame_ptdate = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
02736 |         frame_ptdate.pack(fill="x", padx=15, pady=5)
02737 |
02738 |         # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
02739 |         ptdate_grid = ctk.CTkFrame(frame_ptdate, fg_color="transparent")
02740 |         ptdate_grid.pack(fill="x", padx=10, pady=(0, 10))
02741 |
02742 |         # Baris 0: Label Petunjuk
02743 |         ctk.CTkLabel(ptdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02744 |         ctk.CTkLabel(ptdate_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02745 |         ctk.CTkLabel(ptdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
02746 |
02747 |         # Baris 1: Kotak Input
02748 |         self.entry_pt_day = ctk.CTkEntry(ptdate_grid, width=45, placeholder_text="dd", justify="center")
02749 |         self.entry_pt_day.insert(0, curr_d)
02750 |         self.entry_pt_day.grid(row=1, column=0, padx=(0, 5))
02751 |
02752 |         self.entry_pt_month = ctk.CTkEntry(ptdate_grid, width=45, placeholder_text="mm", justify="center")
02753 |         self.entry_pt_month.insert(0, curr_m)
02754 |         self.entry_pt_month.grid(row=1, column=1, padx=5)
02755 |
02756 |         self.entry_pt_year = ctk.CTkEntry(ptdate_grid, width=70, placeholder_text="yyyy", justify="center")
02757 |         self.entry_pt_year.insert(0, curr_y)
02758 |         self.entry_pt_year.grid(row=1, column=2, padx=(5, 0))
02759 |
02760 |         frame_ptloc = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
02761 |         self.entry_pt_lat = self.create_input_row(frame_ptloc, "Lat (Lintang):", "-7.0667")
02762 |         self.entry_pt_lon = self.create_input_row(frame_ptloc, "Lon (Bujur):", "110.4100")
02763 |         self.entry_pt_elev = self.create_input_row(frame_ptloc, "Elevasi (m):", "230.0")
02764 |         self.entry_pt_tz = self.create_input_row(frame_ptloc, "Timezone:", "7.0")
02765 |
02766 |         frame_ptatm = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")

```

```

02767 |         frame_ptatm.pack(fill="x", padx=15, pady=5)
02768 |         self.entry_pt_temp = self.create_input_row(frame_ptatm, "Suhu (°C):", "10.0")
02769 |         self.entry_pt_pres = self.create_input_row(frame_ptatm, "Tekanan (mb):", "1010.0")
02770 |         self.entry_pt_hum = self.create_input_row(frame_ptatm, "Kelembapan (%):", "60.0")
02771 |         frame_pt_method = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
02772 |         frame_pt_method.pack(fill="x", padx=15, pady=5)
02773 |         self.combo_pt_mazhab = ctk.CTkOptionMenu(frame_pt_method, values=["Standard (Syafi'i, Maliki, Hanbali)", "Hanafi"])
02774 |         self.combo_pt_mazhab.pack(fill="x", padx=10, pady=(0, 5))
02775 |         self.combo_pt_method = ctk.CTkOptionMenu(frame_pt_method, values=["MUHAMMADIYAH (Fajr 18°, Isha 18°)", "Kemenag RI (Fajr 20°, Isha 18°)", "Muslim World League (Fajr 18°, Isha 17°)", "ISNA (Fajr
15°, Isha 15°)", "Egypt (Fajr 19.5°, Isha 17.5°)"])
02776 |         self.combo_pt_method.pack(fill="x", padx=10, pady=(0, 10))
02777 |
02778 |         # Penambahan Opsi Metode Lintang Ekstrem
02779 |         self.combo_pt_highlat = ctk.CTkOptionMenu(
02780 |             frame_pt_method,
02781 |             values=[
02782 |                 "Lintang Normal (Abaikan)",
02783 |                 "Nisf al-Lail (1/2 Malam)",
02784 |                 "Subeh Sadiq (1/7 Malam)",
02785 |                 "Proporsi Sudut (Angle-Based)",
02786 |                 "Aqrab al-Balad (Estimasi 45°)"
02787 |             ]
02788 |         )
02789 |         self.combo_pt_highlat.pack(fill="x", padx=10, pady=(0, 10))
02790 |         self.combo_pt_highlat.set("Lintang Normal (Abaikan)")
02791 |
02792 |         frame_pt_opt = ctk.CTkFrame(self.frame_pt_inputs, fg_color="#212121")
02793 |         frame_pt_opt.pack(fill="x", padx=15, pady=5)
02794 |         self.entry_pt_ikhtiyat = self.create_input_row(frame_pt_opt, "Ikhtiyat (Detik):", "16")
02795 |
02796 |         row_fmt = ctk.CTkFrame(frame_pt_opt, fg_color="transparent")
02797 |         row_fmt.pack(fill="x", padx=10, pady=2)
02798 |         self.combo_pt_fmt = ctk.CTkOptionMenu(row_fmt, values=["HH:MM:SS (Jam:Menit:Detik)", "HH:MM (Jam:Menit)"])
02799 |         self.combo_pt_fmt.pack(fill="x")
02800 |
02801 |         # ---> TAMBAHAN FITUR: TOMBOL IN-FLIGHT PRAYER TIMES <---
02802 |         self.btn_pt_inflight = ctk.CTkButton(self.frame_pt_inputs, text="➡ Shalat Penerbangan (In-Flight)",
02803 |             font=("Segoe UI", 12, "bold"), fg_color="#D84315", hover_color="#BF360C",
02804 |             command=self.buka_kalkulator_penerbangan)
02805 |         self.btn_pt_inflight.pack(fill="x", padx=15, pady=(15, 5))
02806 |
02807 |         # -- VISMAL --
02808 |         self.frame_vismal_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02809 |
02810 |         # 1. FRAME TANGGAL DENGAN GRID SYSTEM (Agar dd, mm, yyyy presisi)
02811 |         frame_vmdate = ctk.CTkFrame(self.frame_vismal_inputs, fg_color="#212121")
02812 |         frame_vmdate.pack(fill="x", padx=15, pady=5)
02813 |         ctk.CTkLabel(frame_vmdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02814 |
02815 |         vmdate_grid = ctk.CTkFrame(frame_vmdate, fg_color="transparent")
02816 |         vmdate_grid.pack(fill="x", padx=10, pady=(0, 10))
02817 |
02818 |         # Baris 0: Label Petunjuk
02819 |         ctk.CTkLabel(vmdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02820 |         ctk.CTkLabel(vmdate_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02821 |         ctk.CTkLabel(vmdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
02822 |
02823 |         # Baris 1: Kotak Input
02824 |         self.entry_vmday = ctk.CTkEntry(vmdate_grid, width=45, placeholder_text="dd", justify="center")
02825 |         self.entry_vmday.insert(0, curr_d)
02826 |         self.entry_vmday.grid(row=1, column=0, padx=(0, 5))
02827 |
02828 |         self.entry_vmmnth = ctk.CTkEntry(vmdate_grid, width=45, placeholder_text="mm", justify="center")
02829 |         self.entry_vmmnth.insert(0, curr_m)
02830 |         self.entry_vmmnth.grid(row=1, column=1, padx=5)
02831 |
02832 |         self.entry_vmyear = ctk.CTkEntry(vmdate_grid, width=70, placeholder_text="yyyy", justify="center")
02833 |         self.entry_vmyear.insert(0, curr_y)
02834 |         self.entry_vmyear.grid(row=1, column=2, padx=(5, 0))
02835 |
02836 |         # 2. FRAME TOGGLE (Lanjutan kode asli)

```

```

02837 |         frame_vmtoggle = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
02838 |         frame_vmtoggle.pack(fill="x", padx=15, pady=5)
02839 |         self.radio_vmphase = ctk.StringVar(value="new")
02840 |         ctk.CTkRadioButton(frame_vmtoggle, text="New Crescent (Evening)", variable=self.radio_vmphase, value="new").pack(anchor="w", padx=15, pady=5)
02841 |         ctk.CTkRadioButton(frame_vmtoggle, text="Old Crescent (Morning)", variable=self.radio_vmphase, value="old").pack(anchor="w", padx=15, pady=(0, 8))
02842 |         frame_vmtime = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
02843 |         frame_vmtime.pack(fill="x", padx=15, pady=5)
02844 |         self.radio_vmtime = ctk.StringVar(value="sunset")
02845 |         row_t1 = ctk.CTkFrame(frame_vmtime, fg_color="transparent")
02846 |         row_t1.pack(fill="x", padx=5)
02847 |         ctk.CTkRadioButton(row_t1, text="Sunset", variable=self.radio_vmtime, value="sunset", width=80).pack(side="left", padx=5)
02848 |         ctk.CTkRadioButton(row_t1, text="Moonset", variable=self.radio_vmtime, value="moonset", width=80).pack(side="left", padx=5)
02849 |         row_t2 = ctk.CTkFrame(frame_vmtime, fg_color="transparent")
02850 |         row_t2.pack(fill="x", padx=5, pady=5)
02851 |         ctk.CTkRadioButton(row_t2, text="Best Time", variable=self.radio_vmtime, value="best").pack(side="left", padx=5)
02852 |
02853 |         frame_vmcrit = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
02854 |         frame_vmcrit.pack(fill="x", padx=15, pady=5)
02855 |
02856 |         # ---> PERBARUI LIST KRITERIA DI SINI <---
02857 |         daftar_kriteria = [
02858 |             "KHGT / Diyanet Turki (Alt>=5, Eln>=8)",
02859 |             "Neo MABIMS (Alt>=3, Eln>=6.4)",
02860 |             "Yallop Criterion (q-Parameter)",
02861 |             "Ilyas (Alt>=4, Eln>=10.5)",
02862 |             "Danjon Limit (Eln>=7)",
02863 |             "Odeh Criterion"
02864 |         ]
02865 |         self.combo_vmcrit = ctk.CTkOptionMenu(frame_vmcrit, values=daftar_kriteria)
02866 |         self.combo_vmcrit.pack(fill="x", padx=10, pady=(0, 10))
02867 |
02868 |         frame_vmparam = ctk.CTkFrame(self.frame_vismap_inputs, fg_color="#212121")
02869 |         frame_vmparam.pack(fill="x", padx=15, pady=5)
02870 |
02871 |         # Ganti dengan opsi Interseksi (Peta KHGT 2D)
02872 |         self.combo_vmparam = ctk.CTkOptionMenu(frame_vmparam, values=["Kriteria Visibilitas", "Altitude Bulan", "Elongasi", "Iluminasi", "Interseksi", "Interseksi (Peta KHGT 2D)"])
02873 |         self.combo_vmparam.pack(fill="x", padx=10, pady=(0, 10))
02874 |
02875 |         # -- KONVERSI --
02876 |         self.frame_conv_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02877 |
02878 |         frame_conv_mode = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
02879 |         frame_conv_mode.pack(fill="x", padx=15, pady=5)
02880 |         self.radio_conv_var = ctk.StringVar(value="m2h")
02881 |
02882 |         # Tambahan Menu Urfi
02883 |         ctk.CTkRadioButton(frame_conv_mode, text="Masehi → Hijriah (KHGT)", variable=self.radio_conv_var, value="m2h", command=self.update_conv_ui).pack(anchor="w", padx=15, pady=5)
02884 |         ctk.CTkRadioButton(frame_conv_mode, text="Hijriah (KHGT) → Masehi", variable=self.radio_conv_var, value="h2m", command=self.update_conv_ui).pack(anchor="w", padx=15, pady=5)
02885 |         ctk.CTkRadioButton(frame_conv_mode, text="Masehi → Hijriah (Urfi)", variable=self.radio_conv_var, value="m2u", command=self.update_conv_ui).pack(anchor="w", padx=15, pady=5)
02886 |         ctk.CTkRadioButton(frame_conv_mode, text="Hijriah (Urfi) → Masehi", variable=self.radio_conv_var, value="u2m", command=self.update_conv_ui).pack(anchor="w", padx=15, pady=(5, 10))
02887 |
02888 |         frame_conv_date = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
02889 |         frame_conv_date.pack(fill="x", padx=15, pady=5)
02890 |         self.lbl_conv_date = ctk.CTkLabel(frame_conv_date, text="TANGGAL MASEHI", font=("Segoe UI", 12, "bold"))
02891 |         self.lbl_conv_date.pack(anchor="w", padx=10, pady=(8, 2))
02892 |
02893 |         # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
02894 |         convdate_grid = ctk.CTkFrame(frame_conv_date, fg_color="transparent")
02895 |         convdate_grid.pack(fill="x", padx=10, pady=(0, 10))
02896 |
02897 |         days = [str(d).zfill(2) for d in range(1, 32)]
02898 |
02899 |         # Baris 0: Label Petunjuk
02900 |         ctk.CTkLabel(convdate_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2), sticky="w", padx=(5, 0))
02901 |         ctk.CTkLabel(convdate_grid, text="Bulan", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2), sticky="w", padx=(5, 0))
02902 |         ctk.CTkLabel(convdate_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2), sticky="w", padx=(5, 0))
02903 |
02904 |         # Baris 1: Kotak Input (Karena ini dropdown, ukurannya sedikit beda)
02905 |         self.combo_conv_day = ctk.CTkComboBox(convdate_grid, values=days, width=60)
02906 |         self.combo_conv_day.set(curr_d)
02907 |         self.combo_conv_day.grid(row=1, column=0, padx=(0, 5))

```

```

02908 |
02909 |     self.combo_conv_month = ctk.CTkComboBox(convdate_grid, values=BULAN_MASEHI, width=120)
02910 |     self.combo_conv_month.set(BULAN_MASEHI[datetime.datetime.now().month - 1])
02911 |     self.combo_conv_month.grid(row=1, column=1, padx=5)
02912 |
02913 |     self.entry_conv_year = ctk.CTkEntry(convdate_grid, width=70, placeholder_text="yyyy", justify="center")
02914 |     self.entry_conv_year.insert(0, curr_y)
02915 |     self.entry_conv_year.grid(row=1, column=2, padx=(5, 0))
02916 |
02917 |     # ---> PERBAIKAN: Menambahkan Label Keterangan Ikhtiyat <---
02918 |     frame_conv_adj = ctk.CTkFrame(self.frame_conv_inputs, fg_color="#212121")
02919 |     frame_conv_adj.pack(fill="x", padx=15, pady=5)
02920 |
02921 |     # Label judul untuk kotak "0"
02922 |     ctk.CTkLabel(frame_conv_adj, text="KOREKSI HARI (IKHTIYAT)", font=("Segoe UI", 11, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02923 |
02924 |     self.combo_conv_adj = ctk.CTkComboBox(frame_conv_adj, values=["-2", "-1", "0", "+1", "+2"])
02925 |     self.combo_conv_adj.set("0")
02926 |     self.combo_conv_adj.pack(fill="x", padx=10, pady=(0, 10))
02927 |
02928 |     # -- QIBLA TIME --
02929 |     self.frame_qt_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02930 |     frame_qtdate = ctk.CTkFrame(self.frame_qt_inputs, fg_color="#212121")
02931 |     frame_qtdate.pack(fill="x", padx=15, pady=5)
02932 |     ctk.CTkLabel(frame_qtdate, text="TANGGAL OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02933 |
02934 |     # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
02935 |     qt_date_grid = ctk.CTkFrame(frame_qtdate, fg_color="transparent")
02936 |     qt_date_grid.pack(fill="x", padx=10, pady=(0, 10))
02937 |
02938 |     # Baris 0: Label Petunjuk
02939 |     ctk.CTkLabel(qt_date_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02940 |     ctk.CTkLabel(qt_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02941 |     ctk.CTkLabel(qt_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
02942 |
02943 |     # Baris 1: Kotak Input
02944 |     self.entry_qtday = ctk.CTkEntry(qt_date_grid, width=45, placeholder_text="dd", justify="center")
02945 |     self.entry_qtday.insert(0, curr_d)
02946 |     self.entry_qtday.grid(row=1, column=0, padx=(0, 5))
02947 |
02948 |     self.entry_qtmonth = ctk.CTkEntry(qt_date_grid, width=45, placeholder_text="mm", justify="center")
02949 |     self.entry_qtmonth.insert(0, curr_m)
02950 |     self.entry_qtmonth.grid(row=1, column=1, padx=5)
02951 |
02952 |     self.entry_qtyear = ctk.CTkEntry(qt_date_grid, width=70, placeholder_text="yyyy", justify="center")
02953 |     self.entry_qtyear.insert(0, curr_y)
02954 |     self.entry_qtyear.grid(row=1, column=2, padx=(5, 0))
02955 |
02956 |     frame_qtloc = ctk.CTkFrame(self.frame_qt_inputs, fg_color="#212121")
02957 |     frame_qtloc.pack(fill="x", padx=15, pady=5)
02958 |     self.entry_qtlat = self.create_input_row(frame_qtloc, "Lat (Lintang):", "-7.0667")
02959 |     self.entry_qtlon = self.create_input_row(frame_qtloc, "Lon (Bujur):", "110.4100")
02960 |     self.entry_qttz = self.create_input_row(frame_qtloc, "Timezone:", "7.0")
02961 |
02962 |     # -- GERHANA --
02963 |     self.frame_gerhana_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02964 |     frame_gdate = ctk.CTkFrame(self.frame_gerhana_inputs, fg_color="#212121")
02965 |     frame_gdate.pack(fill="x", padx=15, pady=10)
02966 |     ctk.CTkLabel(frame_gdate, text="PILIH TAHUN GERHANA", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02967 |
02968 |     # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
02969 |     gerhana_grid = ctk.CTkFrame(frame_gdate, fg_color="transparent")
02970 |     gerhana_grid.pack(fill="x", padx=10, pady=(0, 10))
02971 |
02972 |     ctk.CTkLabel(gerhana_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02973 |
02974 |     # ---> UBAH RANGE MENJADI (1, 3000) DI SINI <---
02975 |     self.combo_tahun_gerhana = ctk.CTkComboBox(gerhana_grid, values=[str(y) for y in range(1, 3000)], justify="center", width=80)
02976 |     self.combo_tahun_gerhana.set(curr_y)
02977 |     self.combo_tahun_gerhana.grid(row=1, column=0, padx=(0, 5))
02978 |

```

```

02979 |     # -- ANIMASI LIVE 2D --
02980 |     self.frame_animasi_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
02981 |
02982 |     frame_anim_date = ctk.CTkFrame(self.frame_animasi_inputs, fg_color="#212121")
02983 |     frame_anim_date.pack(fill="x", padx=15, pady=10)
02984 |     ctk.CTkLabel(frame_anim_date, text="WAKTU SIMULASI 2D", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
02985 |
02986 |     # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
02987 |     anim_date_grid = ctk.CTkFrame(frame_anim_date, fg_color="transparent")
02988 |     anim_date_grid.pack(fill="x", padx=10, pady=(0, 5))
02989 |
02990 |     # Baris 0: Label Petunjuk
02991 |     ctk.CTkLabel(anim_date_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
02992 |     ctk.CTkLabel(anim_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
02993 |     ctk.CTkLabel(anim_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
02994 |
02995 |     # Baris 1: Kotak Input
02996 |     self.entry_anim_day = ctk.CTkEntry(anim_date_grid, width=45, placeholder_text="dd", justify="center")
02997 |     self.entry_anim_day.insert(0, curr_d)
02998 |     self.entry_anim_day.grid(row=1, column=0, padx=(0, 5))
02999 |
03000 |     self.entry_anim_month = ctk.CTkEntry(anim_date_grid, width=45, placeholder_text="mm", justify="center")
03001 |     self.entry_anim_month.insert(0, curr_m)
03002 |     self.entry_anim_month.grid(row=1, column=1, padx=5)
03003 |
03004 |     self.entry_anim_year = ctk.CTkEntry(anim_date_grid, width=70, placeholder_text="yyyy", justify="center")
03005 |     self.entry_anim_year.insert(0, curr_y)
03006 |     self.entry_anim_year.grid(row=1, column=2, padx=(5, 0))
03007 |
03008 |     # ---> PERBAIKAN: Tambahan Label hh:mm:ss <---
03009 |     ctk.CTkLabel(frame_anim_date, text="hh:mm:ss", font=("Segoe UI", 10), text_color="#9E9E9E").pack(anchor="w", padx=12, pady=(5, 0))
03010 |
03011 |     self.entry_anim_time = ctk.CTkEntry(frame_anim_date, placeholder_text="HH:MM:SS")
03012 |     self.entry_anim_time.insert(0, datetime.datetime.now().strftime("%H:%M:%S"))
03013 |     self.entry_anim_time.pack(fill="x", padx=10, pady=(0, 10))
03014 |
03015 |     frame_anim_btn = ctk.CTkFrame(self.frame_animasi_inputs, fg_color="transparent")
03016 |     frame_anim_btn.pack(fill="x", padx=15, pady=5)
03017 |
03018 |     self.btn_anim_terapkan = ctk.CTkButton(frame_anim_btn, text="Terapkan Waktu Kustom", fg_color="#1976D2", hover_color="#1565C0", command=self.anim_set_kustom)
03019 |     self.btn_anim_terapkan.pack(fill="x", pady=5)
03020 |
03021 |     self.btn_anim_sunset = ctk.CTkButton(frame_anim_btn, text="☐ Set ke Waktu Sunset", fg_color="#F57C00", hover_color="#E65100", command=self.anim_cari_sunset)
03022 |     self.btn_anim_sunset.pack(fill="x", pady=5)
03023 |
03024 |     self.btn_anim_live = ctk.CTkButton(frame_anim_btn, text="☐ Kembali ke Live (Sekarang)", fg_color="#2E7D32", hover_color="#1B5E20", command=self.anim_start_live)
03025 |     self.btn_anim_live.pack(fill="x", pady=5)
03026 |
03027 |     self.anim_is_live = True
03028 |     self.anim_custom_time = None
03029 |
03030 |     # -- 3D SYSTEM (SKYFIELD) MENU 16 --
03031 |     self.frame_3d_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
03032 |
03033 |     frame_3d_date = ctk.CTkFrame(self.frame_3d_inputs, fg_color="#212121")
03034 |     frame_3d_date.pack(fill="x", padx=15, pady=10)
03035 |     ctk.CTkLabel(frame_3d_date, text="SET TANGGAL SIMULASI 3D", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
03036 |
03037 |     f3d_row = ctk.CTkFrame(frame_3d_date, fg_color="transparent")
03038 |     f3d_row.pack(fill="x", padx=10, pady=(0, 10))
03039 |
03040 |     # Dropdown Tanggal
03041 |     self.combo_3d_d = ctk.CTkComboBox(f3d_row, values=[f"{i:02d}" for i in range(1, 32)], width=60, command=self.on_3d_date_change)
03042 |     self.combo_3d_d.set(curr_d)
03043 |     self.combo_3d_d.pack(side="left", padx=2)
03044 |
03045 |     # Dropdown Bulan
03046 |     self.combo_3d_m = ctk.CTkComboBox(f3d_row, values=[f"{i:02d}" for i in range(1, 13)], width=60, command=self.on_3d_date_change)
03047 |     self.combo_3d_m.set(curr_m)
03048 |     self.combo_3d_m.pack(side="left", padx=2)
03049 |

```

```

03050 |         # Dropdown Tahun
03051 |         years_range = [str(i) for i in range(1, 3000)]
03052 |         self.combo_3d_y = ctk.CtkComboBox(f3d_row, values=years_range, width=85, command=self.on_3d_date_change)
03053 |         self.combo_3d_y.set(curr_y)
03054 |         self.combo_3d_y.pack(side="left", padx=2)
03055 |
03056 |         # ---> TAMBAHAN: TOMBOL PROSES KHUSUS MENU 16 <---
03057 |         self.btn_apply_3d = ctk.CtkButton(self.frame_3d_inputs, text="▶ TERAPKAN TANGGAL", font=("Segoe UI", 14, "bold"), height=40, fg_color="#1565C0", hover_color="#0D47A1", command=self.force_update_3d)
03058 |         self.btn_apply_3d.pack(fill="x", padx=15, pady=(5, 10))
03059 |
03060 |         # Info navigasi
03061 |         ctk.CtkLabel(self.frame_3d_inputs, text="Klik & Drag layar untuk rotasi.\nScroll/Mousewheel untuk Zoom.",
03062 |             font=("Segoe UI", 11, "italic"), text_color="#FFD54F").pack(pady=5)
03063 |
03064 |         # -- GHA --
03065 |         self.frame_gha_inputs = ctk.CtkFrame(self.sidebar, fg_color="transparent")
03066 |
03067 |         # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
03068 |         frame_gha_date = ctk.CtkFrame(self.frame_gha_inputs, fg_color="#212121")
03069 |         frame_gha_date.pack(fill="x", padx=15, pady=5)
03070 |         ctk.CtkLabel(frame_gha_date, text="TANGGAL REFERENSI (UTC)", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
03071 |
03072 |         gha_date_grid = ctk.CtkFrame(frame_gha_date, fg_color="transparent")
03073 |         gha_date_grid.pack(fill="x", padx=10, pady=(0, 5))
03074 |
03075 |         ctk.CtkLabel(gha_date_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
03076 |         ctk.CtkLabel(gha_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
03077 |         ctk.CtkLabel(gha_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
03078 |
03079 |         self.entry_gha_day = ctk.CtkEntry(gha_date_grid, width=45, placeholder_text="dd", justify="center")
03080 |         self.entry_gha_day.insert(0, curr_d)
03081 |         self.entry_gha_day.grid(row=1, column=0, padx=(0, 5))
03082 |
03083 |         self.entry_gha_month = ctk.CtkEntry(gha_date_grid, width=45, placeholder_text="mm", justify="center")
03084 |         self.entry_gha_month.insert(0, curr_m)
03085 |         self.entry_gha_month.grid(row=1, column=1, padx=5)
03086 |
03087 |         self.entry_gha_year = ctk.CtkEntry(gha_date_grid, width=70, placeholder_text="yyyy", justify="center")
03088 |         self.entry_gha_year.insert(0, curr_y)
03089 |         self.entry_gha_year.grid(row=1, column=2, padx=(5, 0))
03090 |
03091 |         ctk.CtkLabel(frame_gha_date, text="hh:mm:ss", font=("Segoe UI", 10), text_color="#9E9E9E").pack(anchor="w", padx=12, pady=(5, 0))
03092 |         self.entry_gha_time = ctk.CtkEntry(frame_gha_date, placeholder_text="Waktu HH:MM:SS (Def: 12:00:00)")
03093 |         self.entry_gha_time.insert(0, "12:00:00")
03094 |         self.entry_gha_time.pack(fill="x", padx=10, pady=(0, 10))
03095 |
03096 |         # ---> TAMBAHAN: PARAMETER & TOMBOL KURVA ALASKA <---
03097 |         frame_alaska_loc = ctk.CtkFrame(self.frame_gha_inputs, fg_color="#212121", border_width=1, border_color="#00E5FF")
03098 |         frame_alaska_loc.pack(fill="x", padx=15, pady=10)
03099 |         ctk.CtkLabel(frame_alaska_loc, text="□ PARAMETER KURVA ALASKA", font=("Segoe UI", 11, "bold"), text_color="#00E5FF").pack(anchor="w", padx=10, pady=(8, 2))
03100 |
03101 |         self.entry_gha_lat = self.create_input_row(frame_alaska_loc, "Lat:", "56.8135")
03102 |         self.entry_gha_lon = self.create_input_row(frame_alaska_loc, "Lon:", "-158.8622")
03103 |         self.entry_gha_tz = self.create_input_row(frame_alaska_loc, "TZ:", "-11.0")
03104 |
03105 |         self.btn_kurva_alaska = ctk.CtkButton(frame_alaska_loc, text="GENERATE KURVA ALASKA", font=("Segoe UI", 12, "bold"), fg_color="#00C853", hover_color="#00E676", command=self.proses_kurva_alaska)
03106 |         self.btn_kurva_alaska.pack(fill="x", padx=10, pady=(10, 10))
03107 |
03108 |         # -- ALT CHART --
03109 |         self.frame_chart_inputs = ctk.CtkFrame(self.sidebar, fg_color="transparent")
03110 |
03111 |         # --- A. GRAFIK HARIAN (24 JAM) ---
03112 |         c_date = ctk.CtkFrame(self.frame_chart_inputs, fg_color="#212121")
03113 |         c_date.pack(fill="x", padx=15, pady=5)
03114 |         ctk.CtkLabel(c_date, text="GRAFIK HARIAN (24 JAM)", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
03115 |
03116 |         chart_date_grid = ctk.CtkFrame(c_date, fg_color="transparent")
03117 |         chart_date_grid.pack(fill="x", padx=10, pady=(0, 10))
03118 |
03119 |         ctk.CtkLabel(chart_date_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
03120 |         ctk.CtkLabel(chart_date_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))

```



```

03121 |         ctk.CTkLabel(chart_date_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
03122 |
03123 |         self.entry_chart_d = ctk.CTkEntry(chart_date_grid, width=45, placeholder_text="dd", justify="center")
03124 |         self.entry_chart_d.insert(0, curr_d)
03125 |         self.entry_chart_d.grid(row=1, column=0, padx=(0, 5))
03126 |
03127 |         self.entry_chart_m = ctk.CTkEntry(chart_date_grid, width=45, placeholder_text="mm", justify="center")
03128 |         self.entry_chart_m.insert(0, curr_m)
03129 |         self.entry_chart_m.grid(row=1, column=1, padx=5)
03130 |
03131 |         self.entry_chart_y = ctk.CTkEntry(chart_date_grid, width=70, placeholder_text="yyyy", justify="center")
03132 |         self.entry_chart_y.insert(0, curr_y)
03133 |         self.entry_chart_y.grid(row=1, column=2, padx=(5, 0))
03134 |
03135 |         # ---> PERBAIKAN: Menggunakan Grid 2x2 agar ke-4 tombol muat dengan rapi di sidebar <---
03136 |         frame_chart_btns = ctk.CTkFrame(self.frame_chart_inputs, fg_color="transparent")
03137 |         frame_chart_btns.pack(fill="x", padx=15, pady=(5, 5))
03138 |
03139 |         frame_chart_btns.grid_columnconfigure(0, weight=1)
03140 |         frame_chart_btns.grid_columnconfigure(1, weight=1)
03141 |
03142 |         btn_alt = ctk.CTkButton(frame_chart_btns, text="⬇ Alt Chart", fg_color="#1565C0", hover_color="#0D47A1", command=lambda: self.trigger_chart("altitude"))
03143 |         btn_alt.grid(row=0, column=0, padx=(0, 5), pady=5, sticky="ew")
03144 |
03145 |         btn_elong = ctk.CTkButton(frame_chart_btns, text="⬇ Elongasi", fg_color="#2E7D32", hover_color="#1B5E20", command=lambda: self.trigger_chart("elongation"))
03146 |         btn_elong.grid(row=0, column=1, padx=(5, 0), pady=5, sticky="ew")
03147 |
03148 |         btn_illum = ctk.CTkButton(frame_chart_btns, text="⬇ Iluminasi", fg_color="#F57C00", hover_color="#E65100", command=lambda: self.trigger_chart("illumination"))
03149 |         btn_illum.grid(row=1, column=0, padx=(0, 5), pady=5, sticky="ew")
03150 |
03151 |         btn_il_el = ctk.CTkButton(frame_chart_btns, text="⬇ Ill vs Eln", fg_color="#673AB7", hover_color="#512DA8", command=lambda: self.trigger_chart("illum_vs_elong"))
03152 |         btn_il_el.grid(row=1, column=1, padx=(5, 0), pady=5, sticky="ew")
03153 |
03154 |         # --- B. GRAFIK TREN BULANAN HIJRIAH (RAJAB - ZULHIJAH) ---
03155 |         frame_chart_hijri = ctk.CTkFrame(self.frame_chart_inputs, fg_color="#212121", border_width=1, border_color="#D32F2F")
03156 |         frame_chart_hijri.pack(fill="x", padx=15, pady=(15, 10))
03157 |         ctk.CTkLabel(frame_chart_hijri, text="TREN HIJRIAH (RAJAB - ZULHIJAH)", font=("Segoe UI", 12, "bold"), text_color="#FF5252").pack(anchor="w", padx=10, pady=(8, 2))
03158 |
03159 |         row_hy = ctk.CTkFrame(frame_chart_hijri, fg_color="transparent")
03160 |         row_hy.pack(fill="x", padx=10, pady=5)
03161 |         ctk.CTkLabel(row_hy, text="Tahun Hijriah:", font=("Segoe UI", 12)).pack(side="left")
03162 |         self.entry_chart_hy = ctk.CTkEntry(row_hy, width=80, justify="center")
03163 |         self.entry_chart_hy.insert(0, "1447")
03164 |         self.entry_chart_hy.pack(side="right")
03165 |
03166 |         btn_trend_alt = ctk.CTkButton(frame_chart_hijri, text="⬇ Tren Altitude (Geo vs Topo)", fg_color="#D84315", hover_color="#BF360C", command=lambda: self.trigger_trend("altitude"))
03167 |         btn_trend_alt.pack(fill="x", padx=10, pady=5)
03168 |
03169 |         btn_trend_elong = ctk.CTkButton(frame_chart_hijri, text="⬇ Tren Elongasi (Geo vs Topo)", fg_color="#00838F", hover_color="#006064", command=lambda: self.trigger_trend("elongation"))
03170 |         btn_trend_elong.pack(fill="x", padx=10, pady=5)
03171 |
03172 |         btn_trend_illum = ctk.CTkButton(frame_chart_hijri, text="⬇ Tren Iluminasi Siklik", fg_color="#4A148C", hover_color="#38006B", command=lambda: self.trigger_trend("illumination"))
03173 |         btn_trend_illum.pack(fill="x", padx=10, pady=(5, 10))
03174 |
03175 |         # ---> TOMBOL KE-4 BARU: KOMBINASI <---
03176 |         btn_trend_all = ctk.CTkButton(frame_chart_hijri, text="⬇ Tren Kombinasi (Alt, Eln, Illum)", fg_color="#B71C1C", hover_color="#7F0000", command=lambda: self.trigger_trend("kombinasi"))
03177 |         btn_trend_all.pack(fill="x", padx=10, pady=(5, 10))
03178 |
03179 |         # -- FIRST POINT --
03180 |         self.frame_first_point_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
03181 |         fp_date = ctk.CTkFrame(self.frame_first_point_inputs, fg_color="#212121")
03182 |         fp_date.pack(fill="x", padx=15, pady=10)
03183 |
03184 |         ctk.CTkLabel(fp_date, text="TANGGAL PENCARIAN", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
03185 |
03186 |         # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
03187 |         fp_inputs_grid = ctk.CTkFrame(fp_date, fg_color="transparent")
03188 |         fp_inputs_grid.pack(fill="x", padx=10, pady=(0, 10))
03189 |
03190 |         # Membuat list untuk Tanggal, Bulan, dan Tahun
03191 |         days_list = [str(d).zfill(2) for d in range(1, 32)]

```

```

03192 |         months_list = [str(m).zfill(2) for m in range(1, 13)]
03193 |
03194 |         # --> UBAH RANGE MENJADI (1, 3000) DI SINI <---
03195 |         years_list = [str(y) for y in range(1, 3000)] # Range tahun 1 s/d 2999 M
03196 |
03197 |         # Baris 0: Label Petunjuk
03198 |         ctk.CTkLabel(fp_inputs_grid, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
03199 |         ctk.CTkLabel(fp_inputs_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
03200 |         ctk.CTkLabel(fp_inputs_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
03201 |
03202 |         # Baris 1: Kotak Input (Dropdown)
03203 |         self.entry_fp_d = ctk.CTkComboBox(fp_inputs_grid, values=days_list, width=65)
03204 |         self.entry_fp_d.set(curr_d) # Otomatis set ke tanggal hari ini
03205 |         self.entry_fp_d.grid(row=1, column=0, padx=(0, 5))
03206 |
03207 |         self.entry_fp_m = ctk.CTkComboBox(fp_inputs_grid, values=months_list, width=65)
03208 |         self.entry_fp_m.set(curr_m) # Otomatis set ke bulan hari ini
03209 |         self.entry_fp_m.grid(row=1, column=1, padx=5)
03210 |
03211 |         self.entry_fp_y = ctk.CTkComboBox(fp_inputs_grid, values=years_list, width=80)
03212 |         self.entry_fp_y.set(curr_y) # Otomatis set ke tahun hari ini
03213 |         self.entry_fp_y.grid(row=1, column=2, padx=(5, 0))
03214 |
03215 |         # -- SEASON (EQUINOX & SOLSTICE) --
03216 |         self.frame_season_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
03217 |         frame_s_year = ctk.CTkFrame(self.frame_season_inputs, fg_color="#212121")
03218 |         frame_s_year.pack(fill="x", padx=15, pady=10)
03219 |         ctk.CTkLabel(frame_s_year, text="TAHUN OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
03220 |
03221 |         # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
03222 |         season_grid = ctk.CTkFrame(frame_s_year, fg_color="transparent")
03223 |         season_grid.pack(fill="x", padx=10, pady=(0, 10))
03224 |
03225 |         ctk.CTkLabel(season_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
03226 |
03227 |         self.entry_season_year = ctk.CTkEntry(season_grid, width=70, placeholder_text="yyyy", justify="center")
03228 |         self.entry_season_year.insert(0, curr_y)
03229 |         self.entry_season_year.grid(row=1, column=0, padx=(0, 5))
03230 |
03231 |         # -- PLANETARY --
03232 |         self.frame_planet_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
03233 |         frame_pl_date = ctk.CTkFrame(self.frame_planet_inputs, fg_color="#212121")
03234 |         frame_pl_date.pack(fill="x", padx=15, pady=5)
03235 |         ctk.CTkLabel(frame_pl_date, text="BULAN OBSERVASI", font=("Segoe UI", 12, "bold")).pack(anchor="w", padx=10, pady=(8, 2))
03236 |
03237 |         # --- PERBAIKAN: Frame Tanggal dengan Sistem Grid ---
03238 |         planet_grid = ctk.CTkFrame(frame_pl_date, fg_color="transparent")
03239 |         planet_grid.pack(fill="x", padx=10, pady=(0, 10))
03240 |
03241 |         # Baris 0: Label Petunjuk
03242 |         ctk.CTkLabel(planet_grid, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
03243 |         ctk.CTkLabel(planet_grid, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
03244 |
03245 |         # Baris 1: Kotak Input
03246 |         self.entry_pl_month = ctk.CTkEntry(planet_grid, width=45, placeholder_text="mm", justify="center")
03247 |         self.entry_pl_month.insert(0, curr_m)
03248 |         self.entry_pl_month.grid(row=1, column=0, padx=(0, 5))
03249 |
03250 |         self.entry_pl_year = ctk.CTkEntry(planet_grid, width=70, placeholder_text="yyyy", justify="center")
03251 |         self.entry_pl_year.insert(0, curr_y)
03252 |         self.entry_pl_year.grid(row=1, column=1, padx=(5, 0))
03253 |
03254 |         self.combo_pl_target = ctk.CTkOptionMenu(frame_pl_date, values=["Mercury", "Venus", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune", "Pluto"])
03255 |         self.combo_pl_target.pack(fill="x", padx=10, pady=(0, 10))
03256 |
03257 |     def on_3d_date_change(self, event=None):
03258 |         """
03259 |         Fungsi ini dipanggil setiap kali dropdown tanggal/bulan/tahun diubah.
03260 |         Memicu pembaruan instan pada animasi 3D di ruang kanan.
03261 |         """
03262 |         if self.is_viewing_3d:

```

```

03263 |         self.update_3d_animation()
03264 |
03265 |
03266 |     def anim_start_live(self):
03267 |         self.anim_is_live = True
03268 |         now = datetime.datetime.now()
03269 |         self.entry_anim_year.delete(0, 'end'); self.entry_anim_year.insert(0, str(now.year))
03270 |         self.entry_anim_month.delete(0, 'end'); self.entry_anim_month.insert(0, f"{now.month:02d}")
03271 |         self.entry_anim_day.delete(0, 'end'); self.entry_anim_day.insert(0, f"{now.day:02d}")
03272 |         self.entry_anim_time.delete(0, 'end')
03273 |         messagebox.showinfo("Live Mode", "Simulasi dikembalikan ke Waktu Real-Time saat ini.")
03274 |
03275 |     def anim_set_kustom(self):
03276 |         self.anim_is_live = False
03277 |         try:
03278 |             y = int(self.entry_anim_year.get())
03279 |             m = int(self.entry_anim_month.get())
03280 |             d = int(self.entry_anim_day.get())
03281 |             time_str = self.entry_anim_time.get().strip()
03282 |
03283 |             if not time_str:
03284 |                 time_str = "12:00:00"
03285 |                 self.entry_anim_time.insert(0, time_str)
03286 |
03287 |             parts = time_str.split(":")
03288 |             jam = int(parts[0])
03289 |             menit = int(parts[1]) if len(parts) > 1 else 0
03290 |             detik = int(parts[2]) if len(parts) > 2 else 0
03291 |
03292 |             # Ambil timezone berdasarkan bujur
03293 |             try: lon_val = float(self.entry_vlon.get())
03294 |             except: lon_val = 110.4100
03295 |             tz_offset = int(self.get_tz_from_lon(lon_val))
03296 |
03297 |             dt_lokal = datetime.datetime(y, m, d, jam, menit, detik)
03298 |             dt_utc = dt_lokal - datetime.timedelta(hours=tz_offset)
03299 |
03300 |             self.anim_custom_time = self.ts.from_datetime(dt_utc.replace(tzinfo=pytz.utc))
03301 |         except Exception as e:
03302 |             messagebox.showerror("Error Input", f"Format waktu/tanggal tidak valid!\nError: {e}")
03303 |             self.anim_start_live()
03304 |
03305 |
03306 |
03307 |     def create_eph3d_slider(self, parent, label_text, from_, to_, valinit, res=1):
03308 |         row = ctk.CTkFrame(parent, fg_color="transparent")
03309 |         row.pack(fill="x", padx=10, pady=2)
03310 |
03311 |         ctk.CTkLabel(row, text=label_text, width=80, anchor="w", font=("Segoe UI", 12)).pack(side="left")
03312 |
03313 |         # Inilah label yang akan kita kontrol
03314 |         val_label = ctk.CTkLabel(row, text=str(valinit), width=40, anchor="e", font=("Consolas", 12))
03315 |
03316 |         var = ctk.DoubleVar(value=valinit)
03317 |
03318 |         def on_slide(val):
03319 |             fmt = "{:.0f}" if res >= 1 else "{:.2f}"
03320 |             val_label.configure(text=fmt.format(val))
03321 |             if not getattr(self, 'eph3d_updating', False):
03322 |                 self.eph3d_is_live = False
03323 |                 self.eph3d_update_plot()
03324 |
03325 |         slider = ctk.CTkSlider(row, from_=from_, to_=to_, variable=var, number_of_steps=int((to_-from_)/res), command=on_slide)
03326 |         slider.pack(side="left", fill="x", expand=True, padx=(5, 10))
03327 |         val_label.pack(side="right")
03328 |
03329 |         # PERBAIKAN: Kembalikan val_label, BUKAN slider
03330 |         return var, val_label
03331 |
03332 |     def setup_eph3d_out_frame(self):
03333 |         # MENGGUNAKAN CTkScrollableFrame agar konten di ruang kanan bisa di-scroll

```

```

03334 |         self.frame_eph3d_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="#000000", label_text="VISUALISASI EPHEMERIS 3D")
03335 |
03336 |         # Panel Info Teks (Diletakkan di atas)
03337 |         self.lbl_eph3d_info = ctk.CTkLabel(self.frame_eph3d_out, text="Memuat data...",
03338 |                                           font=("Courier", 13), text_color="#00E676",
03339 |                                           justify="left", anchor="nw")
03340 |         self.lbl_eph3d_info.pack(fill="x", padx=15, pady=10)
03341 |
03342 |         # Setup Figure Matplotlib
03343 |         plt.style.use('dark_background')
03344 |         # Ukuran figsize disesuaikan agar proporsional saat di-scroll
03345 |         self.fig_eph3d = plt.figure(figsize=(8, 8), facecolor='#000000')
03346 |         self.ax_eph3d = self.fig_eph3d.add_subplot(111, projection='3d')
03347 |         self.fig_eph3d.subplots_adjust(left=0, right=1, bottom=0, top=1)
03348 |
03349 |         self.canvas_eph3d = FigureCanvasTkAgg(self.fig_eph3d, master=self.frame_eph3d_out)
03350 |         canvas_widget = self.canvas_eph3d.get_tk_widget()
03351 |         canvas_widget.pack(fill="x", expand=True, padx=10, pady=10)
03352 |
03353 |         # Inisialisasi Objek Statis 3D
03354 |         u = np.linspace(0, 2 * np.pi, 60)
03355 |         v = np.linspace(0, np.pi/2, 30)
03356 |         self.ax_eph3d.plot_wireframe(10 * np.outer(np.cos(u), np.sin(v)),
03357 |                                     10 * np.outer(np.sin(u), np.sin(v)),
03358 |                                     10 * np.outer(np.ones(np.size(u)), np.cos(v)),
03359 |                                     color='gray', alpha=0.1)
03360 |
03361 |         xx, yy = np.meshgrid(np.linspace(-10, 10, 2), np.linspace(-10, 10, 2))
03362 |         self.ax_eph3d.plot_surface(xx, yy, np.zeros_like(xx), color='green', alpha=0.3)
03363 |
03364 |         self.ax_eph3d.text(0, 11, 0, 'Utara', color='white', ha='center', fontsize=12)
03365 |         self.ax_eph3d.text(0, -11, 0, 'Selatan', color='white', ha='center', fontsize=12)
03366 |         self.ax_eph3d.text(11, 0, 0, 'Timur', color='white', ha='center', fontsize=12)
03367 |         self.ax_eph3d.text(-11, 0, 0, 'Barat', color='white', ha='center', fontsize=12)
03368 |
03369 |         # Objek Dinamis
03370 |         self.titik_matahari = self.ax_eph3d.plot([], [], [], 'o', color='yellow', markersize=15)
03371 |         self.titik_bulan = self.ax_eph3d.plot([], [], [], 'o', color='white', markersize=10)
03372 |         self.garis_matahari = self.ax_eph3d.plot([], [], [], color='yellow', linestyle='--', alpha=0.5)
03373 |         self.garis_bulan = self.ax_eph3d.plot([], [], [], color='white', linestyle='--', alpha=0.5)
03374 |
03375 |         self.ax_eph3d.set_xlim([-10, 10])
03376 |         self.ax_eph3d.set_ylim([-10, 10])
03377 |         self.ax_eph3d.set_zlim([-2, 10])
03378 |         self.ax_eph3d.set_axis_off()
03379 |
03380 |         self.canvas_eph3d.draw()
03381 |
03382 |     def eph3d_r_ke_xyz(self, alt, az, r=10):
03383 |         # PERBAIKAN: Wajib konversi derajat ke radian agar trigonometri tidak kacau
03384 |         import math
03385 |         alt_rad = math.radians(alt)
03386 |         az_rad = math.radians(az)
03387 |
03388 |         x = r * math.sin(az_rad) * math.cos(alt_rad)
03389 |         y = r * math.cos(az_rad) * math.cos(alt_rad)
03390 |         z = r * math.sin(alt_rad)
03391 |         return x, y, z
03392 |
03393 |     def eph3d_update_plot(self):
03394 |         try:
03395 |             # 1. Ambil nilai Input dari UI
03396 |             lat_val = float(self.entry_eph3d_lat.get())
03397 |             lon_val = float(self.entry_eph3d_lon.get())
03398 |             elev_val = float(self.var_eph3d_elev.get())
03399 |
03400 |             # --- PERBAIKAN: Fallback Timezone Lapis Ganda ---
03401 |             try:
03402 |                 tz_offset = float(self.entry_eph3d_tz.get())
03403 |             except ValueError:
03404 |                 try: tz_offset = float(self.entry_vtz.get())

```

```

03405 |         except ValueError: tz_offset = 7.0 # Default WIB
03406 |
03407 |         y = int(self.var_eph3d_tahun.get())
03408 |         m = int(self.var_eph3d_bulan.get())
03409 |         jam_desimal = self.var_eph3d_jam.get()
03410 |
03411 |         max_d = calendar.monthrange(y, m)[1]
03412 |         d = min(int(self.var_eph3d_hari.get()), max_d)
03413 |
03414 |         # --- PERBAIKAN: Rekonstruksi Waktu Presisi Tinggi ---
03415 |         total_seconds = int(round(jam_desimal * 3600))
03416 |         jam = total_seconds // 3600
03417 |         menit = (total_seconds % 3600) // 60
03418 |         detik = total_seconds % 60
03419 |
03420 |         if jam >= 24:
03421 |             jam = 23; menit = 59; detik = 59
03422 |
03423 |         waktu_lokal = datetime.datetime(y, m, d, jam, menit, detik)
03424 |         waktu_utc = waktu_lokal - datetime.timedelta(hours=tz_offset)
03425 |
03426 |         # Buat Objek Waktu Skyfield
03427 |         t_skyfield = self.ts.utc(waktu_utc.year, waktu_utc.month, waktu_utc.day,
03428 |                                   waktu_utc.hour, waktu_utc.minute, waktu_utc.second)
03429 |
03430 |         bumi = self.eph['earth']
03431 |         matahari = self.eph['sun']
03432 |         bulan = self.eph['moon']
03433 |
03434 |         # Setup Lokasi Pengamat dengan WGS84 Skyfield
03435 |         lokasi_obs = bumi + wgs84.latlon(lat_val, lon_val, elevation_m=elev_val)
03436 |
03437 |         # 2. PERHITUNGAN TOPOSENTRIK MURNI (SKYFIELD)
03438 |         app_sun_topo = lokasi_obs.at(t_skyfield).observe(matahari).apparent()
03439 |         app_moon_topo = lokasi_obs.at(t_skyfield).observe(bulan).apparent()
03440 |
03441 |         alt_s_topo_obj, az_s_topo_obj, _ = app_sun_topo.altaz(temperature_C=25, pressure_mbar=1010)
03442 |         alt_m_topo_obj, az_m_topo_obj, _ = app_moon_topo.altaz(temperature_C=25, pressure_mbar=1010)
03443 |
03444 |         alt_topo_sun = alt_s_topo_obj.degrees
03445 |         az_topo_sun = az_s_topo_obj.degrees
03446 |         alt_topo_moon = alt_m_topo_obj.degrees
03447 |         az_topo_moon = az_m_topo_obj.degrees
03448 |
03449 |         elong_topo = app_sun_topo.separation_from(app_moon_topo).degrees
03450 |
03451 |         # Fraksi Iluminasi Fase Bulan
03452 |         illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_skyfield)
03453 |         iluminasi_fase = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0
03454 |
03455 |         # 3. PERHITUNGAN GEOSENTRIK (SKYFIELD)
03456 |         sun_geo = bumi.at(t_skyfield).observe(matahari).apparent()
03457 |         moon_geo = bumi.at(t_skyfield).observe(bulan).apparent()
03458 |
03459 |         elong_geo = sun_geo.separation_from(moon_geo).degrees
03460 |
03461 |         ra_sun, dec_sun, _ = sun_geo.radec()
03462 |         ra_moon, dec_moon, _ = moon_geo.radec()
03463 |
03464 |         gast = t_skyfield.gast
03465 |         lst = gast + (lon_val / 15.0)
03466 |         lat_rad = math.radians(lat_val)
03467 |
03468 |         HA_sun_rad = math.radians((lst - ra_sun.hours) * 15.0)
03469 |         sin_alt_geo_sun = math.sin(lat_rad) * math.sin(dec_sun.radians) + math.cos(lat_rad) * math.cos(dec_sun.radians) * math.cos(HA_sun_rad)
03470 |         alt_geo_sun = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo_sun))))
03471 |
03472 |         HA_moon_rad = math.radians((lst - ra_moon.hours) * 15.0)
03473 |         sin_alt_geo_moon = math.sin(lat_rad) * math.sin(dec_moon.radians) + math.cos(lat_rad) * math.cos(dec_moon.radians) * math.cos(HA_moon_rad)
03474 |         alt_geo_moon = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo_moon))))
03475 |

```

```

03476 |         # 4. PLOTTING 3D MATPLOTLIB & UPDATE INFO UI
03477 |         dip_deg = (1.76 * math.sqrt(max(0, elev_val))) / 60.0
03478 |
03479 |         # Terapkan posisi toposentrik ke Plot 3D (Sekarang sudah memakan Radian)
03480 |         sx, sy, sz = self.eph3d_r_ke_xyz(alt_topo_sun, az_topo_sun)
03481 |         mx, my, mz = self.eph3d_r_ke_xyz(alt_topo_moon, az_topo_moon)
03482 |
03483 |         self.titik_matahari.set_data([sx], [sy])
03484 |         self.titik_matahari.set_3d_properties([sz])
03485 |         self.garis_matahari.set_data([0, sx], [0, sy])
03486 |         self.garis_matahari.set_3d_properties([0, sz])
03487 |
03488 |         self.titik_bulan.set_data([mx], [my])
03489 |         self.titik_bulan.set_3d_properties([mz])
03490 |         self.garis_bulan.set_data([0, mx], [0, my])
03491 |         self.garis_bulan.set_3d_properties([0, mz])
03492 |
03493 |         status_waktu = "SEDANG LIVE" if getattr(self, 'eph3d_is_live', False) else "WAKTU KUSTOM"
03494 |         tz_str = f"UTC+{tz_offset}" if tz_offset >= 0 else f"UTC{tz_offset}"
03495 |
03496 |         try:
03497 |             nama_kota = self.opt_city.get()
03498 |             nama_prov = self.opt_prov.get()
03499 |             lokasi_text = f"{nama_kota}, {nama_prov}"
03500 |         except:
03501 |             lokasi_text = "CUSTOM LOCATION"
03502 |
03503 |         teks = (
03504 |             f"LOKASI AKTIF: {lokasi_text}\n"
03505 |             f"Lat: {lat_val} | Lon: {lon_val} | Elevasi: {elev_val}m\n"
03506 |             f"WAKTU LOKAL ({tz_str}): \n"
03507 |             f"{waktu_lokal.strftime('%d-%b-%Y %H:%M:%S')}" - {status_waktu}\n\n"
03508 |             f"UFUK MAR'I (Batas Dip): -{dip_deg:.3f}°\n"
03509 |             f"[MATAHARI]\n"
03510 |             f"└ Toposentrik -> Tinggi: {alt_topo_sun:>6.2f}° | Azimuth: {az_topo_sun:>6.2f}°\n"
03511 |             f"└ Geosentrik -> Tinggi: {alt_geo_sun:>6.2f}°\n"
03512 |             f"[BULAN / HILAL]\n"
03513 |             f"└ Toposentrik -> Tinggi: {alt_topo_moon:>6.2f}° | Azimuth: {az_topo_moon:>6.2f}°\n"
03514 |             f"└ Geosentrik -> Tinggi: {alt_geo_moon:>6.2f}°\n"
03515 |             f"[RELASI & KHGT]\n"
03516 |             f"└ Elongasi Toposentrik : {elong_topo:>6.2f}°\n"
03517 |             f"└ Elongasi Geosentrik : {elong_geo:>6.2f}°\n"
03518 |             f"└ Fase Bulan (iluminasi): {iluminasi_fase:>5.1f}%"
03519 |         )
03520 |
03521 |         self.lbl_eph3d_info.configure(text=teks)
03522 |         self.canvas_eph3d.draw_idle()
03523 |
03524 |     except Exception as e:
03525 |         import traceback
03526 |         print(f"Error 3D Plot: {e}\n{traceback.format_exc()}")
03527 |
03528 |     def eph3d_cari_sunset(self):
03529 |         self.eph3d_is_live = False
03530 |         try:
03531 |             lat_val = float(self.entry_eph3d_lat.get())
03532 |             lon_val = float(self.entry_eph3d_lon.get())
03533 |             elev_val = float(self.var_eph3d_elev.get())
03534 |
03535 |             # --- PERBAIKAN: Ambil TZ dari UI ---
03536 |             try:
03537 |                 tz_offset = float(self.entry_eph3d_tz.get())
03538 |             except:
03539 |                 tz_offset = float(self.entry_vtz.get())
03540 |
03541 |             y = int(self.var_eph3d_tahun.get())
03542 |             m = int(self.var_eph3d_bulan.get())
03543 |             d = min(int(self.var_eph3d_hari.get()), calendar.monthrange(y, m)[1])
03544 |
03545 |             # Setup parameter Skyfield
03546 |             loc = wgs84.latlon(lat_val, lon_val, elevation_m=elev_val)

```

```

03547 |         t0 = self.ts.utc(y, m, d, -tz_offset)
03548 |         t1 = self.ts.utc(y, m, d, 24 - tz_offset)
03549 |
03550 |         # Deteksi Sunset menggunakan Skyfield
03551 |         f_rs = almanac.sunrise_sunset(self.eph, loc)
03552 |         t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)
03553 |
03554 |         waktu_sunset_lokal = None
03555 |         for t_ev, y_ev in zip(np.atleast_1d(t_rs), np.atleast_1d(y_rs)):
03556 |             if y_ev == 0: # 0 berarti Sunset
03557 |                 dt_utc = t_ev.utc_datetime()
03558 |                 waktu_sunset_lokal = dt_utc + datetime.timedelta(hours=tz_offset)
03559 |                 break
03560 |
03561 |         if waktu_sunset_lokal:
03562 |             jam_desimal = waktu_sunset_lokal.hour + (waktu_sunset_lokal.minute / 60.0) + (waktu_sunset_lokal.second / 3600.0)
03563 |
03564 |             self.eph3d_updating = True
03565 |             self.var_eph3d_tahun.set(waktu_sunset_lokal.year)
03566 |             self.var_eph3d_bulan.set(waktu_sunset_lokal.month)
03567 |             self.var_eph3d_hari.set(waktu_sunset_lokal.day)
03568 |             self.var_eph3d_jam.set(jam_desimal)
03569 |
03570 |             self.lbl_eph3d_tahun.configure(text=str(waktu_sunset_lokal.year))
03571 |             self.lbl_eph3d_bulan.configure(text=str(waktu_sunset_lokal.month))
03572 |             self.lbl_eph3d_hari.configure(text=str(waktu_sunset_lokal.day))
03573 |             self.lbl_eph3d_jam.configure(text=f"{jam_desimal:.2f}")
03574 |
03575 |             self.eph3d_updating = False
03576 |             self.eph3d_update_plot()
03577 |         else:
03578 |             messagebox.showwarning("Peringatan", "Matahari tidak terbenam pada hari/lokasi tersebut (Anomali Kutub).")
03579 |
03580 |     except Exception as e:
03581 |         print(f"Gagal mencari sunset 3D: {e}")
03582 |
03583 |     def eph3d_start_live(self):
03584 |         self.eph3d_is_live = True
03585 |         self.eph3d_update_plot() # Panggil 1x update plot seketika (mencegah delay)
03586 |
03587 |     def eph3d_live_update_loop(self):
03588 |         # 1. Cek apakah menu yang aktif benar-benar Menu 19
03589 |         current_mode = self.combo_mode.get()
03590 |
03591 |         if "Simulasi Ephemeris 3D" in current_mode and getattr(self, 'eph3d_is_live', False):
03592 |             self.eph3d_updating = True
03593 |             try:
03594 |                 # --- PERBAIKAN: Dapatkan Timezone yang presisi dari UI ---
03595 |                 try:
03596 |                     tz_offset = float(self.entry_eph3d_tz.get())
03597 |                 except ValueError:
03598 |                     try: tz_offset = float(self.entry_vtz.get())
03599 |                 except ValueError: tz_offset = 7.0
03600 |
03601 |                 # Gunakan UTC murni untuk menghindari limitasi tahun 1-9999 pada datetime
03602 |                 now_utc = datetime.datetime.now(datetime.timezone.utc).replace(tzinfo=None)
03603 |                 now_target = now_utc + datetime.timedelta(hours=tz_offset)
03604 |
03605 |                 # Update slider/variable tanpa memicu event 'command' slider agar tidak rekursif
03606 |                 self.var_eph3d_tahun.set(now_target.year)
03607 |                 self.var_eph3d_bulan.set(now_target.month)
03608 |                 self.var_eph3d_hari.set(now_target.day)
03609 |
03610 |                 jam_des = now_target.hour + (now_target.minute / 60.0) + (now_target.second / 3600.0)
03611 |                 self.var_eph3d_jam.set(jam_des)
03612 |
03613 |                 # Update label teks pendamping slider
03614 |                 self.lbl_eph3d_tahun.configure(text=str(now_target.year))
03615 |                 self.lbl_eph3d_bulan.configure(text=str(now_target.month))
03616 |                 self.lbl_eph3d_hari.configure(text=str(now_target.day))
03617 |                 self.lbl_eph3d_jam.configure(text=f"{jam_des:.2f}")

```

```

03618 |
03619 |         # Jalankan render plot
03620 |         self.eph3d_update_plot()
03621 |     except Exception as e:
03622 |         print(f"Silent error in 3D loop: {e}")
03623 |     finally:
03624 |         self.eph3d_updating = False
03625 |
03626 |     # 2. Pemanggilan ulang loop (HANYA satu kali self.after)
03627 |     self.after(1000, self.eph3d_live_update_loop)
03628 |
03629 | def switch_mode(self, mode):
03630 |     # Setiap pemilihan menu membersihkan halaman muka/laporan sebelumnya.
03631 |     # Untuk modul berbasis tombol, ruang kanan menampilkan petunjuk PROSES DATA.
03632 |     self.prepare_module_right_output()
03633 |
03634 |     # 1. Pastikan tombol PROSES DATA muncul kembali saat pindah menu
03635 |     self.btn_hitung.pack(fill="x", padx=15, pady=(20, 10))
03636 |
03637 |     # 2. Sembunyikan semua input module di sidebar
03638 |     for f in [self.frame_vis_inputs, self.frame_moon_inputs, self.frame_eph_inputs, self.frame_qiblah_inputs,
03639 |              self.frame_mt_inputs, self.frame_st_inputs, self.frame_pt_inputs, self.frame_vismap_inputs,
03640 |              self.frame_conv_inputs, self.frame_qt_inputs, self.frame_gerhana_inputs, self.frame_animasi_inputs,
03641 |              self.frame_3d_inputs, self.frame_gha_inputs, self.frame_chart_inputs, self.frame_first_point_inputs,
03642 |              self.frame_season_inputs, self.frame_planet_inputs, self.frame_eph3d_inputs, self.frame_kalender_inputs,
03643 |              self.frame_kalmasehi_inputs, self.frame_winai_inputs, self.frame_autobuild_inputs, self.frame_mizwala_inputs,
03644 |              self.frame_kriteria_batas_inputs,
03645 |              self.frame_astro_inputs, self.frame_pasang_inputs, self.frame_fajar_inputs, self.frame_analemma_inputs]:
03646 |         f.pack_forget()
03647 |
03648 |     # 3. Sembunyikan semua output module di ruang kanan secara aman
03649 |     self.textbox.grid_remove()
03650 |
03651 |     if hasattr(self, 'frame_gerhana_out'): self.frame_gerhana_out.grid_remove()
03652 |     if hasattr(self, 'frame_animasi_out'): self.frame_animasi_out.grid_remove()
03653 |     if hasattr(self, 'frame_3d_out'): self.frame_3d_out.grid_remove()
03654 |     if hasattr(self, 'frame_chart_out'): self.frame_chart_out.grid_remove()
03655 |     if hasattr(self, 'frame_eph3d_out'): self.frame_eph3d_out.grid_remove()
03656 |     if hasattr(self, 'frame_kalender_out'): self.frame_kalender_out.grid_remove()
03657 |     if hasattr(self, 'frame_kalmasehi_out'): self.frame_kalmasehi_out.grid_remove()
03658 |     if hasattr(self, 'frame_winai_out'): self.frame_winai_out.grid_remove()
03659 |     if hasattr(self, 'frame_kriteria_batas_out'): self.frame_kriteria_batas_out.grid_remove()
03660 |     if hasattr(self, 'frame_analemma_out'): self.frame_analemma_out.grid_remove()
03661 |     if hasattr(self, 'frame_alaska_out'): self.frame_alaska_out.grid_remove() # <--- TAMBAHKAN INI
03662 |
03663 |     self.anim_running = False
03664 |     self.is_viewing_3d = False
03665 |
03666 |     # Bersihkan frame chart agar tidak menumpuk
03667 |     if hasattr(self, 'frame_chart_out'):
03668 |         for widget in self.frame_chart_out.wininfo_children():
03669 |             widget.destroy()
03670 |
03671 |     # 4. LOGIKA PERGANTIAN MENU
03672 |     if "Peta Langit Dunia" in mode:
03673 |         self.lbl_main_title.configure(text=self.tr("Peta Langit Dunia"))
03674 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03675 |         self.textbox.configure(text_color="#00E5FF", state="normal")
03676 |         self.textbox.delete("1.0", "end")
03677 |         self.textbox.insert(
03678 |             "1.0",
03679 |             self.tr("Membuka Peta Langit Dunia melalui browser default...") + "\n\n" +
03680 |             WORLD_SKY_MAP_URL
03681 |         )
03682 |         self.textbox.configure(state="disabled")
03683 |         self.btn_hitung.pack_forget()
03684 |         self.after(100, self.open_peta_langit_dunia)
03685 |
03686 |     elif "Laporan Astronomi" in mode:
03687 |         self.lbl_main_title.configure(text="Laporan Astronomi")
03688 |         self.textbox.grid(row=1, column=0, sticky="nsew")

```



```

03689 |         self.textbox.configure(text_color="#00E5FF", state="normal")
03690 |         self.textbox.delete("1.0", "end")
03691 |         self.textbox.insert(
03692 |             "1.0",
03693 |             "Membuka aplikasi Laporan Astronomi...\n\n"
03694 |             "Pastikan file 'LaporanAstronomi.exe' berada di folder yang sama "
03695 |             "dengan aplikasi KHGT Times."
03696 |         )
03697 |         self.textbox.configure(state="disabled")
03698 |         self.btn_hitung.pack_forget()
03699 |         self.after(100, self.open_laporan_astronomi)
03700 |
03701 |     elif "Visibility Hilal" in mode:
03702 |         self.frame_vis_inputs.pack(fill="x", before=self.btn_hitung)
03703 |         self.lbl_main_title.configure(text="Crescent Analysis & KHGT")
03704 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03705 |         self.textbox.configure(text_color="#E0E0E0")
03706 |     elif "Visibility Map" in mode:
03707 |         self.frame_vismap_inputs.pack(fill="x", before=self.btn_hitung)
03708 |         self.lbl_main_title.configure(text="Crescent Visibility Map Scanner")
03709 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03710 |         self.textbox.configure(text_color="#E0E0E0")
03711 |         self.textbox.configure(state="normal")
03712 |         self.textbox.delete("1.0", "end")
03713 |         self.textbox.insert("1.0", "Silakan klik 'PROSES DATA' untuk melakukan pemindaian peta global HD. \nProses ini menghitung data spasial dan melakukan interpolasi bicubic. Harap tunggu...")
03714 |         self.textbox.configure(state="disabled")
03715 |     elif "Analisis Hilal Global" in mode:
03716 |         self.frame_gha_inputs.pack(fill="x", before=self.btn_hitung)
03717 |         self.lbl_main_title.configure(text="Global Hilal Visibility Analyzer")
03718 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03719 |         self.textbox.configure(text_color="#00E676")
03720 |     elif "Moonphase" in mode:
03721 |         self.frame_moon_inputs.pack(fill="x", before=self.btn_hitung)
03722 |         self.lbl_main_title.configure(text="Moonphase Cycle Data")
03723 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03724 |         self.textbox.configure(text_color="#00E5FF")
03725 |     elif "Sun Moon Ephemeris" in mode:
03726 |         self.frame_eph_inputs.pack(fill="x", before=self.btn_hitung)
03727 |         self.lbl_main_title.configure(text="Sun and Moon Ephemeris Data")
03728 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03729 |         self.textbox.configure(text_color="#E0E0E0")
03730 |     elif "Qiblah Direction" in mode:
03731 |         self.frame_qiblah_inputs.pack(fill="x", before=self.btn_hitung)
03732 |         self.lbl_main_title.configure(text="Qiblah Direction & Alignment Times")
03733 |         self.textbox.grid(row=1, column=0, sticky="nsew") # Menampilkan box teks laporan
03734 |         self.textbox.configure(text_color="#FFD54F")
03735 |     elif "Moon Times" in mode:
03736 |         self.frame_mt_inputs.pack(fill="x", before=self.btn_hitung)
03737 |         self.lbl_main_title.configure(text="Moon Rise, Transit, and Set Times")
03738 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03739 |         self.textbox.configure(text_color="#E0E0E0")
03740 |     elif "Sun Times" in mode:
03741 |         self.frame_st_inputs.pack(fill="x", before=self.btn_hitung)
03742 |         self.lbl_main_title.configure(text="Sun Rise, Transit, and Set Times")
03743 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03744 |         self.textbox.configure(text_color="#E0E0E0")
03745 |     elif "Prayer Times" in mode:
03746 |         self.frame_pt_inputs.pack(fill="x", before=self.btn_hitung)
03747 |         self.lbl_main_title.configure(text="Islamic Prayer Times & Twilight")
03748 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03749 |         self.textbox.configure(text_color="#E0E0E0")
03750 |     elif "Konversi" in mode:
03751 |         self.frame_conv_inputs.pack(fill="x", before=self.btn_hitung)
03752 |         self.lbl_main_title.configure(text="Gregorian & KHGT Calendar Conversion")
03753 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03754 |         self.textbox.configure(text_color="#00E676")
03755 |     elif "Qibla Time" in mode:
03756 |         self.frame_qt_inputs.pack(fill="x", before=self.btn_hitung)
03757 |         self.lbl_main_title.configure(text="Rashdul Qiblah Lokal")
03758 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03759 |         self.textbox.configure(text_color="#FFD54F")

```

```

03760 |         elif "Analisis Gerhana" in mode:
03761 |             self.frame_gerhana_inputs.pack(fill="x", before=self.btn_hitung)
03762 |             self.lbl_main_title.configure(text="Precision Eclipse Calculator")
03763 |             self.frame_gerhana_out.grid(row=1, column=0, sticky="nsew")
03764 |         elif "Live Animasi" in mode:
03765 |             self.frame_animasi_inputs.pack(fill="x", before=self.btn_hitung)
03766 |             self.lbl_main_title.configure(text="Live Celestial Object Tracker")
03767 |             self.frame_animasi_out.grid(row=1, column=0, sticky="nsew")
03768 |             self.btn_hitung.pack_forget() # Sembunyikan tombol proses
03769 |
03770 |             # Reset ke waktu berjalan
03771 |             self.anim_start_live()
03772 |
03773 |             self.anim_running = True
03774 |             self.draw_static_background()
03775 |             self.update_animation()
03776 |         elif "Sistem Sun Moon" in mode:
03777 |             self.frame_3d_inputs.pack(fill="x", before=self.btn_hitung)
03778 |             self.lbl_main_title.configure(text="3D Geocentric Space System")
03779 |             self.frame_3d_out.grid(row=1, column=0, sticky="nsew")
03780 |             self.btn_hitung.pack_forget()
03781 |             self.is_viewing_3d = True
03782 |             self.update_3d_animation()
03783 |         elif "Altitude Chart" in mode:
03784 |             self.frame_chart_inputs.pack(fill="x", before=self.btn_hitung)
03785 |             self.lbl_main_title.configure(text="Celestial Altitude Graphics")
03786 |             self.textbox.grid_remove()
03787 |             self.frame_chart_out.grid(row=1, column=0, sticky="nsew")
03788 |             self.show_pending_message_in_frame(self.frame_chart_out)
03789 |             self.lbl_status.configure(
03790 |                 text=self.tr("Klik PROSES DATA untuk render grafik"),
03791 |                 text_color="#00E5FF"
03792 |             )
03793 |         elif "Kota Pertama" in mode:
03794 |             self.frame_first_point_inputs.pack(fill="x", before=self.btn_hitung)
03795 |             self.lbl_main_title.configure(text="KHGT first Point (Mainland)")
03796 |             self.textbox.grid(row=1, column=0, sticky="nsew")
03797 |             self.textbox.configure(state="normal")
03798 |             self.textbox.delete("1.0", "end")
03799 |             self.textbox.insert("1.0", "Klik 'PROSES DATA' untuk memulai scanning global mencari titik pertama di daratan utama...")
03800 |             self.textbox.configure(state="disabled")
03801 |         elif "Equinox" in mode:
03802 |             self.frame_season_inputs.pack(fill="x", before=self.btn_hitung)
03803 |             self.lbl_main_title.configure(text="Equinox & Solstice Calculator")
03804 |             self.textbox.grid(row=1, column=0, sticky="nsew")
03805 |             self.textbox.configure(text_color="#FFD54F")
03806 |         elif "Planetary Times" in mode:
03807 |             self.frame_planet_inputs.pack(fill="x", before=self.btn_hitung)
03808 |             self.lbl_main_title.configure(text="Rise, Transit & Set Planet")
03809 |             self.textbox.grid(row=1, column=0, sticky="nsew")
03810 |             self.textbox.configure(text_color="#00E5FF")
03811 |
03812 |         # ---> INI DIA BLOK MENU 19 YANG SEMPAT TERHAPUS <---
03813 |         elif "Simulasi Ephemeris 3D" in mode:
03814 |             self.frame_eph3d_inputs.pack(fill="x", before=self.btn_hitung)
03815 |             self.lbl_main_title.configure(text="3D Sun & Moon Ephemeris Simulation")
03816 |
03817 |             self.textbox.grid_remove()
03818 |             self.frame_eph3d_out.grid(row=1, column=0, sticky="nsew")
03819 |
03820 |             # Sembunyikan tombol proses karena ini live interaktif
03821 |             self.btn_hitung.pack_forget()
03822 |
03823 |             # Langsung jalankan animasi live saat menu diklik
03824 |             self.eph3d_start_live()
03825 |
03826 |         # ---> URUTAN YANG BENAR (SYAWAL HARUS DI ATAS) <---
03827 |         elif "Komparasi 50 Tahun (Syawal)" in mode:
03828 |             self.lbl_main_title.configure(text="Komparasi 50 Tahun 1 Syawal")
03829 |             self.textbox.grid(row=1, column=0, sticky="nsew")
03830 |             self.textbox.configure(text_color="#00E676", state="normal")

```

```

03831 |         self.textbox.delete("1.0", "end")
03832 |         self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai komputasi 50 tahun penentuan 1 Syawal.\nProses iterasi ini akan memakan waktu beberapa detik, mohon tunggu...")
03833 |         self.textbox.configure(state="disabled")
03834 |         self.frame_eph3d_inputs.pack_forget()
03835 |
03836 | # ---> TAMBAHKAN MENU 35 DI SINI <---
03837 | elif "Komparasi 50 Thn (Zulhijah)" in mode:
03838 |     self.lbl_main_title.configure(text="Komparasi 50 Tahun 1 Zulhijah")
03839 |     self.textbox.grid(row=1, column=0, sticky="nsew")
03840 |     self.textbox.configure(text_color="#00E676", state="normal")
03841 |     self.textbox.delete("1.0", "end")
03842 |     self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai komputasi 50 tahun penentuan 1 Zulhijah.\nProses iterasi ini akan memakan waktu beberapa detik, mohon tunggu...")
03843 |     self.textbox.configure(state="disabled")
03844 |     self.frame_eph3d_inputs.pack_forget()
03845 |
03846 | elif "Komparasi 50 Tahun (12 Bulan)" in mode:
03847 |     self.lbl_main_title.configure(text="Analisis Probabilitas & Komparasi Global")
03848 |     self.textbox.grid(row=1, column=0, sticky="nsew")
03849 |     self.textbox.configure(text_color="#00E676", state="normal")
03850 |     self.textbox.delete("1.0", "end")
03851 |     self.textbox.insert("1.0", "Klik '► PROSES DATA' untuk memulai audit 12 bulan Hijriah selama 50 tahun.\nSistem akan menghitung probabilitas sinkronisasi KHGT vs MABIMS Sabang.")
03852 |     self.textbox.configure(state="disabled")
03853 |     self.frame_eph3d_inputs.pack_forget()
03854 |
03855 | elif "Komparasi 50 Tahun" in mode:
03856 |     self.lbl_main_title.configure(text="Komparasi 50 Tahun 1 Ramadhan")
03857 |     self.textbox.grid(row=1, column=0, sticky="nsew")
03858 |     self.textbox.configure(text_color="#00E676", state="normal")
03859 |     self.textbox.delete("1.0", "end")
03860 |     self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai komputasi 50 tahun.\nProses iterasi ini akan memakan waktu beberapa detik, mohon tunggu...")
03861 |     self.textbox.configure(state="disabled")
03862 |     self.frame_eph3d_inputs.pack_forget()
03863 |
03864 | elif "Tabel Tinggi Hilal 50 Thn (Syawal)" in mode:
03865 |     self.lbl_main_title.configure(text="Tabel Data Ketinggian Hilal 50 Tahun (1 Syawal)")
03866 |     self.textbox.grid(row=1, column=0, sticky="nsew")
03867 |     self.textbox.configure(text_color="#FFD54F", state="normal")
03868 |     self.textbox.delete("1.0", "end")
03869 |     self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai pemindaian global 50 tahun (Fokus Ketinggian 1 Syawal).\nProses ini menggunakan Akselerasi Database...")
03870 |     self.textbox.configure(state="disabled")
03871 |     self.frame_eph3d_inputs.pack_forget()
03872 |
03873 | # ---> TAMBAHKAN MENU 36 DI SINI <---
03874 | elif "Tabel Tinggi Hilal 50 Thn (Zulhijah)" in mode:
03875 |     self.lbl_main_title.configure(text="Tabel Data Ketinggian Hilal 50 Tahun (1 Zulhijah)")
03876 |     self.textbox.grid(row=1, column=0, sticky="nsew")
03877 |     self.textbox.configure(text_color="#FFD54F", state="normal")
03878 |     self.textbox.delete("1.0", "end")
03879 |     self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai pemindaian global 50 tahun (Fokus Ketinggian 1 Zulhijah).\nProses ini menggunakan Akselerasi Database...")
03880 |     self.textbox.configure(state="disabled")
03881 |     self.frame_eph3d_inputs.pack_forget()
03882 |
03883 | elif "Tabel Tinggi Hilal" in mode:
03884 |     self.lbl_main_title.configure(text="Tabel Data Ketinggian Hilal 50 Tahun (1 Ramadhan)")
03885 |     self.textbox.grid(row=1, column=0, sticky="nsew")
03886 |     self.textbox.configure(text_color="#FFD54F", state="normal")
03887 |     self.textbox.delete("1.0", "end")
03888 |     self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai pemindaian global 50 tahun (1 Ramadhan).\nProses ini sangat cepat berkat Akselerasi Database...")
03889 |     self.textbox.configure(state="disabled")
03890 |     self.frame_eph3d_inputs.pack_forget()
03891 |
03892 | elif "Tabel Elongasi Hilal" in mode:
03893 |     self.lbl_main_title.configure(text="Tabel Data Elongasi Hilal 50 Tahun (1 Ramadhan)")
03894 |     self.textbox.grid(row=1, column=0, sticky="nsew")
03895 |     self.textbox.configure(text_color="#FFD54F", state="normal")
03896 |     self.textbox.delete("1.0", "end")
03897 |     self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai pemindaian global 50 tahun (Fokus Elongasi).\nProses ini sangat cepat berkat Akselerasi Database...")
03898 |     self.textbox.configure(state="disabled")
03899 |     self.frame_eph3d_inputs.pack_forget()
03900 | # ---> SISIPKAN BLOK MENU 25 DI SINI <---
03901 | elif "Tabel Elongasi Hilal 50 Thn (Syawal)" in mode:

```

```

03902 |         self.lbl_main_title.configure(text="Tabel Data Elongasi Hilal 50 Tahun (1 Syawal)")
03903 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03904 |         self.textbox.configure(text_color="#FFD54F", state="normal")
03905 |         self.textbox.delete("1.0", "end")
03906 |         self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai pemindaian global 50 tahun (Fokus Elongasi 1 Syawal).\nProses ini menggunakan Akselerasi Database...")
03907 |         self.textbox.configure(state="disabled")
03908 |         self.frame_eph3d_inputs.pack_forget()
03909 |
03910 |     # ---> TAMBAHKAN MENU 37 DI SINI <---
03911 |     elif "Tabel Elongasi Hilal 50 Thn (Zulhijah)" in mode:
03912 |         self.lbl_main_title.configure(text="Tabel Data Elongasi Hilal 50 Tahun (1 Zulhijah)")
03913 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03914 |         self.textbox.configure(text_color="#FFD54F", state="normal")
03915 |         self.textbox.delete("1.0", "end")
03916 |         self.textbox.insert("1.0", "Silakan klik tombol '► PROSES DATA' untuk memulai pemindaian global 50 tahun (Fokus Elongasi 1 Zulhijah).\nProses ini menggunakan Akselerasi Database...")
03917 |         self.textbox.configure(state="disabled")
03918 |         self.frame_eph3d_inputs.pack_forget()
03919 |
03920 |     # Pastikan ini berada di bawah Menu 25
03921 |     elif "Tabel Elongasi Hilal" in mode:
03922 |         self.lbl_main_title.configure(text="Tabel Data Elongasi Hilal 50 Tahun (1 Ramadhan)")
03923 |
03924 |     elif "Kalender Hijriah" in mode:
03925 |         self.frame_kalender_inputs.pack(fill="x", before=self.btn_hitung)
03926 |         self.lbl_main_title.configure(text="Kalender Hijriah Global Tunggal")
03927 |
03928 |         self.textbox.grid_remove()
03929 |         self.frame_kalender_out.grid(row=1, column=0, sticky="nsew")
03930 |
03931 |         # Sembunyikan tombol proses karena kalender me-render dirinya sendiri secara instan
03932 |         self.btn_hitung.pack_forget()
03933 |         self.render_kalender()
03934 |
03935 |     elif "Kalender Masehi" in mode:
03936 |         self.frame_kalmasehi_inputs.pack(fill="x", before=self.btn_hitung)
03937 |         self.lbl_main_title.configure(text="Kalender Masehi Terintegrasi KHGT")
03938 |
03939 |         self.textbox.grid_remove()
03940 |         self.frame_kalmasehi_out.grid(row=1, column=0, sticky="nsew")
03941 |
03942 |         # Sembunyikan tombol proses karena dirender instan
03943 |         self.btn_hitung.pack_forget()
03944 |         self.render_kalmasehi()
03945 |
03946 |     # ---> TAMBAHKAN BLOK WINAI INI <---
03947 |     elif "WinAI" in mode:
03948 |         self.frame_winai_inputs.pack(fill="x", before=self.btn_hitung)
03949 |         self.lbl_main_title.configure(text="WinAI: Aplikasi AI Windows")
03950 |
03951 |         self.textbox.grid_remove()
03952 |         self.frame_winai_out.grid(row=1, column=0, sticky="nsew")
03953 |         self.btn_hitung.pack_forget()
03954 |
03955 |         # ---> INJEKSI OTOMATIS TANGGAL & LOKASI KE LAYAR <---
03956 |         loc_prov = self.opt_prov.get()
03957 |         loc_city = self.opt_city.get()
03958 |         now = datetime.datetime.now()
03959 |         tgl_str = now.strftime("%d %B %Y")
03960 |
03961 |         pesan_konteks = (f"Parameter Sistem Dimuat: Anda berada di {loc_city}, {loc_prov}.\n"
03962 |                          f"Tanggal Saat Ini: {tgl_str}.\n"
03963 |                          f"□ Tips: Anda bisa langsung mengetik 'hitung visibilitas hilal hari ini' atau 'cari kota perama KHGT' "
03964 |                          f"untuk menjalankan kalkulasi KHGT secara instan.")
03965 |
03966 |         self.winai_output_box.configure(state="normal")
03967 |         # ---> PERBAIKAN: Ubah tag dari "info" menjadi "bold" agar hitam pekat dan tebal <---
03968 |         self.winai_output_box.insert("end", f"□ Sistem:\n{pesan_konteks}\n\n", "bold")
03969 |         self.winai_output_box.see("end")
03970 |         self.winai_output_box.configure(state="disabled")
03971 |
03972 |     elif "Mizwala" in mode:

```

```

03973 |         self.frame_mizwala_inputs.pack(fill="x", before=self.btn_hitung)
03974 |         self.lbl_main_title.configure(text="Kalkulator Bayangan Tongkat Istiwa (Mizwala)")
03975 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03976 |         self.textbox.configure(text_color="#FFD54F")
03977 |
03978 |     elif "Auto-Builder" in mode:
03979 |         self.frame_autobuild_inputs.pack(fill="x", before=self.btn_hitung)
03980 |         self.lbl_main_title.configure(text="HIJRI_DB Auto-Builder (Generator Siklus KHGT)")
03981 |         self.textbox.grid(row=1, column=0, sticky="nsew")
03982 |         self.textbox.configure(text_color="#FFD54F", state="normal")
03983 |         self.textbox.delete("1.0", "end")
03984 |         self.textbox.insert("1.0", "☐ TOOL DEVELOPER / ADMIN ☐\n\nKlik ► PROSES DATA' untuk me-run iterasi engine Ephemeris NASA.\nSistem akan merekonstruksi ulang penanggalan Kalender Hijriah
Global Tunggal (KHGT) untuk rentang tahun yang Anda input, kemudian menuliskannya secara otomatis ke dalam file 'db_hijriah.json'.\n\n(Peringatan: Pemrosesan 100 Tahun akan memakan waktu beberapa saat).")
03985 |         self.textbox.configure(state="disabled")
03986 |
03987 |     elif "Status Kriteria Batas" in mode:
03988 |         self.frame_kriteria_batas_inputs.pack(fill="x", before=self.btn_hitung)
03989 |         self.lbl_main_title.configure(text="Simulasi Interaktif: Anatomi Kriteria KHGT 5-8")
03990 |         self.frame_kriteria_batas_out.grid(row=1, column=0, sticky="nsew")
03991 |
03992 |         # HAPUS BARIS INI -> self.btn_hitung.pack_forget()
03993 |         # Biarkan tombol hitung tetap muncul!
03994 |
03995 |         self.update_kriteria_batas_plot() # Panggil plot pertama kali
03996 |
03997 |     elif "Astrofotografi" in mode:
03998 |         self.frame_astro_inputs.pack(fill="x", before=self.btn_hitung)
03999 |         self.lbl_main_title.configure(text="Astrofotografi Hilal & Visibilitas Kontras")
04000 |         self.textbox.grid(row=1, column=0, sticky="nsew")
04001 |         self.textbox.configure(text_color="#FFD54F", state="normal")
04002 |         self.textbox.delete("1.0", "end")
04003 |         self.textbox.insert("1.0", "Klik ► PROSES DATA' untuk menghitung Limit Danjon dan Jendela Kontras Astrofotografi...")
04004 |         self.textbox.configure(state="disabled")
04005 |
04006 |     elif "Pasang Surut" in mode:
04007 |         self.frame_pasang_inputs.pack(fill="x", before=self.btn_hitung)
04008 |         self.lbl_main_title.configure(text="Prediktor Pasang Surut Gravitasi")
04009 |         self.textbox.grid(row=1, column=0, sticky="nsew")
04010 |
04011 |     elif "Fajar Shadiq" in mode:
04012 |         self.frame_fajar_inputs.pack(fill="x", before=self.btn_hitung)
04013 |         self.lbl_main_title.configure(text="Evaluasi Fajar Shadiq & SQM Teoritis")
04014 |         self.textbox.grid(row=1, column=0, sticky="nsew")
04015 |
04016 |     elif "Analemma" in mode:
04017 |         self.frame_analemma_inputs.pack(fill="x", before=self.btn_hitung)
04018 |         self.lbl_main_title.configure(text="Generator Analemma Matahari")
04019 |         self.textbox.grid_remove() # Sembunyikan textbox default
04020 |         self.frame_analemma_out.grid(row=1, column=0, sticky="nsew") # Tampilkan Scrollable Frame Baru
04021 |
04022 |     #elif "Analemma Matahari" in mode:
04023 |     #    self.clear_main_frame()
04024 |
04025 |     # Container Utama
04026 |     container = ctk.CTkFrame(self.main_frame, fg_color="transparent")
04027 |     container.pack(fill="both", expand=True, padx=20, pady=20)
04028 |
04029 |     # --- SIDEBAR INPUT (KIRI) ---
04030 |     input_frame = ctk.CTkFrame(container, width=300)
04031 |     input_frame.pack(side="left", fill="y", padx=(0, 20))
04032 |
04033 |     ctk.CTkLabel(input_frame, text="PENGATURAN ANALEMMA", font=ctk.CTkFont(size=16, weight="bold")).pack(pady=15)
04034 |
04035 |     # Input Tahun & Jam
04036 |     self.entry_analemma_year = self.create_input(input_frame, "Tahun Analisis:", str(datetime.datetime.now().year))
04037 |     self.entry_analemma_hour = self.create_input(input_frame, "Jam Pengamatan (Lokal):", "12")
04038 |
04039 |     # Input Lokasi (Akan otomatis terisi oleh sistem sinkronisasi)
04040 |     self.entry_analemma_lat = self.create_input(input_frame, "Lintang (Latitude):", self.entry_vlat.get())
04041 |     self.entry_analemma_lon = self.create_input(input_frame, "Bujur (Longitude):", self.entry_vlon.get())
04042 |     # Cari bagian pembentukan sidebar Menu 34 di setup_ui

```

```

04043 |         self.entry_analemma_tz = self.create_input_row(frame_analemma_loc, "Timezone:", "7.0")
04044 |
04045 |         # Tambahkan label cuaca khusus menu 34 di sini
04046 |         self.lbl_cuaca_analemma = ctk.CTkLabel(frame_analemma_loc, text="Cuaca: Menunggu deteksi...", font=("Consolas", 10, "bold"), text_color="#9E9E9E")
04047 |         self.lbl_cuaca_analemma.pack(anchor="w", padx=10, pady=(0, 5))
04048 |
04049 |         btn_proses = ctk.CTkButton(input_frame, text="PROSES ANALEMMA", fg_color="#FF6D00", hover_color="#E65100",
04050 |                                   command=self.calculate_analemma)
04051 |         btn_proses.pack(pady=20, padx=20, fill="x")
04052 |
04053 |         # --- AREA HASIL (KANAN) ---
04054 |         result_frame = ctk.CTkFrame(container)
04055 |         result_frame.pack(side="right", fill="both", expand=True)
04056 |
04057 |         self.txt_analemma_report = ctk.CTkTextbox(result_frame, font=ctk.CTkFont(family="Courier", size=12))
04058 |         self.txt_analemma_report.pack(fill="both", expand=True, padx=10, pady=10)
04059 |
04060 |         # Widget yang baru ditampilkan/dibuat langsung mengikuti bahasa aktif.
04061 |         self.install_secondary_output_textbox_hooks()
04062 |         self.force_update_sidebar_language()
04063 |         self.update_ui_texts(self)
04064 |         self.apply_language_direction()
04065 |
04066 |         # =====
04067 |         # LOGIKA MODUL 08: ALARM & NOTIFIKASI SALAT
04068 |         # =====
04069 |         def on_alarm_toggle(self):
04070 |             if self.alarm_enabled.get():
04071 |                 self.lbl_countdown.configure(text="Menginisiasi Jadwal Alarm...", text_color="#00E676")
04072 |                 self._is_calculating_alarm = True
04073 |                 threading.Thread(target=self.update_silent_schedule, daemon=True).start()
04074 |             else:
04075 |                 self.lbl_countdown.configure(text="Alarm Salat: Dinonaktifkan", text_color="#FF5252")
04076 |                 if HAS_PYGAME:
04077 |                     try: pygame.mixer.music.stop()
04078 |                     except: pass
04079 |
04080 |         def pilih_file_audio(self):
04081 |             filepath = filedialog.askopenfilename(title="Pilih Audio Adzan", filetypes=[("Audio Files", "*.mp3 *.wav")])
04082 |             if filepath:
04083 |                 self.adzan_audio_path.set(filepath)
04084 |                 filename = os.path.basename(filepath)
04085 |                 if len(filename) > 20: filename = filename[:17] + "..."
04086 |                 self.lbl_audio_path.configure(text=filename)
04087 |
04088 |         def update_silent_schedule(self):
04089 |             """Menghitung jadwal salat harian tanpa menampilkan ke layar utama"""
04090 |             if not self.eph:
04091 |                 self.after(0, lambda: self.lbl_countdown.configure(text="Menunggu Ephemeris...", text_color="#FFAB40"))
04092 |                 self._is_calculating_alarm = False
04093 |                 return
04094 |
04095 |             try:
04096 |                 # --- PERBAIKAN 1: Sinkronisasi Zona Waktu Independen ---
04097 |                 try: tz_offset = float(self.entry_pt_tz.get())
04098 |                 except: tz_offset = 7.0
04099 |
04100 |                 tz_info = datetime.timezone(datetime.timedelta(hours=tz_offset))
04101 |                 now_utc = datetime.datetime.now(datetime.timezone.utc)
04102 |                 now_local = now_utc.astimezone(tz_info).replace(tzinfo=None)
04103 |
04104 |                 waktu_kalkulasi = now_local
04105 |                 # Jika dipanggil otomatis saat jadwal hari ini habis, gunakan tanggal besok
04106 |                 if getattr(self, '_force_tomorrow', False):
04107 |                     waktu_kalkulasi = now_local + datetime.timedelta(days=1)
04108 |                     self._force_tomorrow = False
04109 |
04110 |                 year, month, day = waktu_kalkulasi.year, waktu_kalkulasi.month, waktu_kalkulasi.day
04111 |                 # -----
04112 |
04113 |             try:

```

```

04114 |         lat = float(self.entry_pt_lat.get())
04115 |         lon = float(self.entry_pt_lon.get())
04116 |         elev = float(self.entry_pt_elev.get())
04117 |     except:
04118 |         lat, lon, elev = -7.0667, 110.4100, 230.0
04119 |
04120 |     method_val = self.combo_pt_method.get()
04121 |     if "Kemenag" in method_val:
04122 |         fajr_angle, isha_angle = -20.0, -18.0
04123 |     elif "Muslim World League" in method_val:
04124 |         fajr_angle, isha_angle = -18.0, -17.0
04125 |     elif "ISNA" in method_val:
04126 |         fajr_angle, isha_angle = -15.0, -15.0
04127 |     elif "Egypt" in method_val:
04128 |         fajr_angle, isha_angle = -19.5, -17.5
04129 |     else:
04130 |         fajr_angle, isha_angle = -18.0, -18.0
04131 |
04132 |     asr_factor = 2.0 if "Hanafi" in self.combo_pt_mazhab.get() else 1.0
04133 |     try: ikhtiyat_sec = int(self.entry_pt_ikhtiyat.get())
04134 |     except: ikhtiyat_sec = 16
04135 |
04136 |     earth = self.eph['earth']
04137 |     sun = self.eph['sun']
04138 |     loc = wgs84.latlon(lat, lon, elevation_m=elev)
04139 |
04140 |     # --- PERBAIKAN 2: Batas waktu T0 dan T1 yang presisi ---
04141 |     t0_dt = datetime.datetime(year, month, day, 0, 0, 0, tzinfo=tz_info)
04142 |     t1_dt = t0_dt + datetime.timedelta(days=1)
04143 |
04144 |     t0 = self.ts.from_datetime(t0_dt)
04145 |     t1 = self.ts.from_datetime(t1_dt)
04146 |     # -----
04147 |
04148 |     tt_array = np.linspace(t0.tt, t1.tt, 2880)
04149 |     t_search = self.ts.tt_jd(tt_array)
04150 |
04151 |     alt_deg = (earth + loc).at(t_search).observe(sun).apparent().altaz()[0].degrees
04152 |     idx_noon = np.argmax(alt_deg)
04153 |     tt_dhohur = tt_array[idx_noon]
04154 |     alt_noon = alt_deg[idx_noon]
04155 |
04156 |     alt_am = alt_deg[:idx_noon]
04157 |     tt_am = tt_array[:idx_noon]
04158 |     alt_pm = alt_deg[idx_noon:]
04159 |     tt_pm = tt_array[idx_noon:]
04160 |
04161 |     zenith_noon = 90.0 - alt_noon
04162 |     shadow_noon = math.tan(math.radians(max(0, zenith_noon)))
04163 |     shadow_asr = asr_factor + shadow_noon
04164 |     alt_asr = math.degrees(math.atan(1.0 / shadow_asr))
04165 |
04166 |     def find_crossing(alt_arr, tt_arr, target_alt, direction='up'):
04167 |         diffs = alt_arr - target_alt
04168 |         for i in range(len(diffs)-1):
04169 |             if direction == 'up' and diffs[i] <= 0 and diffs[i+1] > 0:
04170 |                 frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
04171 |                 return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
04172 |             elif direction == 'down' and diffs[i] >= 0 and diffs[i+1] < 0:
04173 |                 frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
04174 |                 return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
04175 |         return None
04176 |
04177 |     val_fajr = find_crossing(alt_am, tt_am, fajr_angle, 'up')
04178 |     val_shuroq = find_crossing(alt_am, tt_am, -0.833, 'up')
04179 |     val_asr = find_crossing(alt_pm, tt_pm, alt_asr, 'down')
04180 |     val_maghreb = find_crossing(alt_pm, tt_pm, -0.833, 'down')
04181 |     val_isha = find_crossing(alt_pm, tt_pm, isha_angle, 'down')
04182 |
04183 |     highlat_method = self.combo_pt_highlat.get()
04184 |

```

```

04185 |         if "Lintang Normal" not in highlat_method:
04186 |             # 1. Jika Matahari masih terbit & terbenam, hitung durasi malam
04187 |             if val_maghreb is not None and val_shuroq is not None:
04188 |                 # Durasi siang (Julian Date Float: 1.0 = 24 Jam)
04189 |                 durasi_siang = val_maghreb - val_shuroq
04190 |                 durasi_malam = 1.0 - durasi_siang
04191 |
04192 |             if "1/2 Malam" in highlat_method:
04193 |                 porsi = durasi_malam / 2.0
04194 |                 if val_isha is None: val_isha = val_maghreb + porsi
04195 |                 if val_fajr is None: val_fajr = val_shuroq - porsi
04196 |
04197 |             elif "1/7 Malam" in highlat_method:
04198 |                 porsi = durasi_malam / 7.0
04199 |                 if val_isha is None: val_isha = val_maghreb + porsi
04200 |                 if val_fajr is None: val_fajr = val_shuroq - porsi
04201 |
04202 |             elif "Proporsi Sudut" in highlat_method:
04203 |                 porsi_isha = durasi_malam * (abs(isha_angle) / 60.0)
04204 |                 porsi_fajr = durasi_malam * (abs(fajr_angle) / 60.0)
04205 |                 if val_isha is None: val_isha = val_maghreb + porsi_isha
04206 |                 if val_fajr is None: val_fajr = val_shuroq - porsi_fajr
04207 |
04208 |             elif "Aqrab al-Balad" in highlat_method:
04209 |                 # Estimasi matematis cepat untuk Lintang 45° di Musim Panas
04210 |                 # Rata-rata selisih Maghrib-Isya adalah 1.5 jam (1.5 / 24.0 JD)
04211 |                 if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
04212 |                 if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)
04213 |
04214 |         # 2. Penanganan Ekstrem: Midnight Sun / Polar Night (Matahari tidak terbit/terbenam)
04215 |         if val_maghreb is None or val_shuroq is None:
04216 |             if tt_dhohur is not None:
04217 |                 # Aqrab al-Ayyam Asumsi Dasar (Menarik garis 6 jam dari waktu Dzuhur/Transit)
04218 |                 if val_shuroq is None: val_shuroq = tt_dhohur - (6.0 / 24.0)
04219 |                 if val_maghreb is None: val_maghreb = tt_dhohur + (6.0 / 24.0)
04220 |                 if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)
04221 |                 if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
04222 |                 if val_dhuha is None: val_dhuha = val_shuroq + (1.0 / 24.0)
04223 |                 if val_asr is None: val_asr = tt_dhohur + (3.0 / 24.0)
04224 |             # -----
04225 |
04226 |         schedule = {}
04227 |         def add_to_schedule(name, tt_val, is_shuroq=False):
04228 |             if tt_val is not None:
04229 |                 # --- PERBAIKAN 3: Konversi TZ aman ---
04230 |                 dt_utc_aware = self.ts.tt_jd(tt_val).utc_datetime()
04231 |                 dt_local_aware = dt_utc_aware.astimezone(tz_info)
04232 |
04233 |                 if is_shuroq: dt_local_aware -= datetime.timedelta(seconds=ikhtiyat_sec)
04234 |                 else: dt_local_aware += datetime.timedelta(seconds=ikhtiyat_sec)
04235 |                 schedule[name] = dt_local_aware.replace(tzinfo=None)
04236 |
04237 |         add_to_schedule("Subuh", val_fajr)
04238 |         add_to_schedule("Terbit/Syuruq", val_shuroq, is_shuroq=True)
04239 |         add_to_schedule("Dzuhur", tt_dhohur)
04240 |         add_to_schedule("Ashar", val_asr)
04241 |         add_to_schedule("Maghrib", val_maghreb)
04242 |         add_to_schedule("Isya", val_isha)
04243 |
04244 |         self.daily_prayer_schedule = schedule
04245 |         self.last_calculated_date = now_local.date()
04246 |
04247 |     except Exception as e:
04248 |         print(f"Error Silent Schedule: {e}")
04249 |         self.after(0, lambda e=e: self.lbl_countdown.configure(text="Gagal Kalkulasi Alarm!", text_color="#FF5252"))
04250 |     finally:
04251 |         self._is_calculating_alarm = False
04252 |
04253 |     def alarm_tick(self):
04254 |         """Thread GUI (Main Thread) 1-detik loop untuk Cek Alarm dan Update Countdown Header"""
04255 |

```



```

04256 |     # --- PERBAIKAN: Menyamakan detak jam dengan Zona Waktu Lokal secara Dinamis ---
04257 |     try: tz_offset = float(self.entry_pt_tz.get())
04258 |     except: tz_offset = 7.0
04259 |
04260 |     tz_info = datetime.timezone(datetime.timedelta(hours=tz_offset))
04261 |     now_utc = datetime.datetime.now(datetime.timezone.utc)
04262 |     now_local = now_utc.astimezone(tz_info).replace(tzinfo=None)
04263 |     # -----
04264 |
04265 |     # Update harian (refresh jam 00:00)
04266 |     if self.alarm_enabled.get() and self.last_calculated_date != now_local.date() and self.eph is not None:
04267 |         if not getattr(self, '_is_calculating_alarm', False):
04268 |             self._is_calculating_alarm = True
04269 |             threading.Thread(target=self.update_silent_schedule, daemon=True).start()
04270 |
04271 |     if self.alarm_enabled.get():
04272 |         if self.daily_prayer_schedule:
04273 |             # 1. Handle Snooze Logic
04274 |             if self.snooze_time and now_local >= self.snooze_time:
04275 |                 target_time = self.snooze_time
04276 |                 self.snooze_time = None
04277 |                 self.trigger_alarm(self.snooze_prayer, target_time, is_snooze=True)
04278 |
04279 |             # 2. Cari Salat Selanjutnya
04280 |             future_prayers = {k: v for k, v in self.daily_prayer_schedule.items() if v > now_local}
04281 |
04282 |             if future_prayers:
04283 |                 next_prayer = min(future_prayers, key=future_prayers.get)
04284 |                 next_time = future_prayers[next_prayer]
04285 |                 diff = next_time - now_local
04286 |
04287 |                 hours, remainder = divmod(diff.seconds, 3600)
04288 |                 minutes, seconds = divmod(remainder, 60)
04289 |                 self.lbl_countdown.configure(text=f"Waktu {next_prayer} dalam {hours:02d}:{minutes:02d}:{seconds:02d}", text_color="#00E676")
04290 |
04291 |                 if diff.total_seconds() <= 1 and self.last_triggered_prayer != next_prayer:
04292 |                     self.trigger_alarm(next_prayer, next_time)
04293 |             else:
04294 |                 # Langsung beralih ke jadwal besok agar countdown ke Subuh berjalan
04295 |                 self.lbl_countdown.configure(text="Beralih ke jadwal esok hari...", text_color="#FFAB40")
04296 |                 if not getattr(self, '_is_calculating_alarm', False):
04297 |                     self._force_tomorrow = True
04298 |                     self._is_calculating_alarm = True
04299 |                     threading.Thread(target=self.update_silent_schedule, daemon=True).start()
04300 |                 elif not getattr(self, '_is_calculating_alarm', False):
04301 |                     self.lbl_countdown.configure(text="Menunggu Data...", text_color="#FFAB40")
04302 |             elif not self.alarm_enabled.get():
04303 |                 self.lbl_countdown.configure(text="Alarm Salat: Dinonaktifkan", text_color="#FF5252")
04304 |
04305 |         self.after(1000, self.alarm_tick)
04306 |
04307 |     def trigger_alarm(self, prayer_name, target_time, is_snooze=False):
04308 |         if not is_snooze:
04309 |             self.last_triggered_prayer = prayer_name
04310 |
04311 |             city_name = self.opt_city.get()
04312 |             title = f"Waktu Salat {prayer_name}"
04313 |             msg = f"Telah masuk waktu {prayer_name} untuk wilayah {city_name} dan sekitarnya."
04314 |
04315 |             if HAS_TOAST:
04316 |                 threading.Thread(target=lambda: self.toaster.show_toast(title, msg, duration=10, threaded=True), daemon=True).start()
04317 |
04318 |             if HAS_PYGAME and self.alarm_enabled.get():
04319 |                 audio_path = self.adzan_audio_path.get()
04320 |                 try:
04321 |                     if audio_path and os.path.exists(audio_path):
04322 |                         pygame.mixer.music.load(audio_path)
04323 |                         pygame.mixer.music.play()
04324 |                     else:
04325 |                         import winsound
04326 |                         winsound.MessageBeep()

```

```

04327 |         except Exception as e:
04328 |             pass
04329 |
04330 |     self.show_alarm_popup(prayer_name, msg)
04331 |
04332 | def show_alarm_popup(self, prayer_name, msg):
04333 |     popup = ctk.CTkToplevel(self)
04334 |     popup.title(f"Notifikasi Waktu Salat")
04335 |     popup.geometry("400x250")
04336 |     popup.attributes("-topmost", True)
04337 |
04338 |     ctk.CTkLabel(popup, text=f"WAKTU {prayer_name.upper()}", font=("Segoe UI", 24, "bold"), text_color="#00E5FF").pack(pady=(20, 10))
04339 |     ctk.CTkLabel(popup, text=msg, font=("Segoe UI", 12), wraplength=350, justify="center").pack(pady=10)
04340 |
04341 |     btn_frame = ctk.CTkFrame(popup, fg_color="transparent")
04342 |     btn_frame.pack(pady=20)
04343 |
04344 |     def matikan_alarm():
04345 |         if HAS_PYGAME:
04346 |             try: pygame.mixer.music.stop()
04347 |             except: pass
04348 |         popup.destroy()
04349 |
04350 |     def set_snooze(menit):
04351 |         if HAS_PYGAME:
04352 |             try: pygame.mixer.music.stop()
04353 |             except: pass
04354 |         self.snooze_time = datetime.datetime.now() + datetime.timedelta(minutes=menit)
04355 |         self.snooze_prayer = prayer_name
04356 |         self.lbl_countdown.configure(text=f"Snooze {prayer_name} ({menit}m)...", text_color="#FFAB40")
04357 |         popup.destroy()
04358 |
04359 |     ctk.CTkButton(btn_frame, text="Matikan", fg_color="#D32F2F", hover_color="#B71C1C", width=100, command=matikan_alarm).pack(side="left", padx=5)
04360 |     ctk.CTkButton(btn_frame, text="Snooze 5m", fg_color="#F57C00", hover_color="#E65100", width=100, command=lambda: set_snooze(5)).pack(side="left", padx=5)
04361 |     ctk.CTkButton(btn_frame, text="Snooze 10m", fg_color="#1976D2", hover_color="#1565C0", width=100, command=lambda: set_snooze(10)).pack(side="left", padx=5)
04362 |
04363 | # =====
04364 | # LOGIKA PADA FORM/UI LAINNYA
04365 | # =====
04366 | def on_prov_change(self, prov):
04367 |     cities = sorted(list(CITY_DB[prov].keys()))
04368 |     self.opt_city.configure(values=cities)
04369 |     self.opt_city.set(cities[0])
04370 |     self.on_city_change(cities[0])
04371 |
04372 | def on_city_change(self, city):
04373 |     prov = self.opt_prov.get()
04374 |     if prov in CITY_DB and city in CITY_DB[prov]:
04375 |         lat, lon = CITY_DB[prov][city]
04376 |         tz_val = self.get_tz_from_lon(lon)
04377 |
04378 |         # Daftar lengkap variabel input koordinat (1 s.d 34)
04379 |         lat_entries = [
04380 |             'entry_vlat', 'entry_eph_lat', 'entry_qlat', 'entry_mt_lat', 'entry_st_lat', 'entry_pt_lat',
04381 |             'entry_qtlat', 'entry_eph3d_lat', 'entry_miz_lat', 'entry_kb_lat', 'entry_astro_lat',
04382 |             'entry_pasang_lat', 'entry_fajar_lat', 'entry_analemma_lat'
04383 |         ]
04384 |         lon_entries = [
04385 |             'entry_vlon', 'entry_eph_lon', 'entry_qlon', 'entry_mt_lon', 'entry_st_lon', 'entry_pt_lon',
04386 |             'entry_qtlon', 'entry_eph3d_lon', 'entry_miz_lon', 'entry_kb_lon', 'entry_astro_lon',
04387 |             'entry_pasang_lon', 'entry_fajar_lon', 'entry_analemma_lon'
04388 |         ]
04389 |         tz_entries = [
04390 |             'entry_vtz', 'entry_eph_tz', 'entry_qtz', 'entry_mt_tz', 'entry_st_tz', 'entry_pt_tz',
04391 |             'entry_qttz', 'entry_miz_tz', 'entry_kb_tz', 'entry_astro_tz', 'entry_pasang_tz',
04392 |             'entry_fajar_tz', 'entry_analemma_tz', 'entry_eph3d_tz' # <--- TAMBAHKAN DI AKHIR
04393 |         ]
04394 |
04395 |         # Update semua kotak input di sidebar secara serentak
04396 |         for name in lat_entries:
04397 |             if hasattr(self, name):

```

```

04398 |         e = getattr(self, name)
04399 |         e.delete(0, 'end'); e.insert(0, str(lat))
04400 |
04401 |     for name in lon_entries:
04402 |         if hasattr(self, name):
04403 |             e = getattr(self, name)
04404 |             e.delete(0, 'end'); e.insert(0, str(lon))
04405 |
04406 |     for name in tz_entries:
04407 |         if hasattr(self, name):
04408 |             e = getattr(self, name)
04409 |             e.delete(0, 'end'); e.insert(0, str(tz_val))
04410 |
04411 |     self.lokasi_nama.set(f"{city}, {prov} (Lat: {lat}, Lon: {lon}, TZ: +{int(tz_val)})")
04412 |
04413 |     # ---> TAMBAHKAN KODE INI UNTUK UPDATE CUACA OTOMATIS SAAT GANTI KOTA <---
04414 |     if hasattr(self, 'fetch_and_update_weather'):
04415 |         labels_cuaca = [
04416 |             'lbl_status_cuaca', 'lbl_cuaca_astro',
04417 |             'lbl_cuaca_pasang', 'lbl_cuaca_fajar', 'lbl_cuaca_analemma'
04418 |         ]
04419 |         for lbl_name in labels_cuaca:
04420 |             if hasattr(self, lbl_name):
04421 |                 label_obj = getattr(self, lbl_name)
04422 |                 self.fetch_and_update_weather(lat, lon, label_obj)
04423 |         # -----
04424 |
04425 |         if getattr(self, 'alarm_enabled', None) and self.alarm_enabled.get():
04426 |             threading.Thread(target=self.update_silent_schedule, daemon=True).start()
04427 |
04428 |     def update_calendar_widget(self):
04429 |         now = datetime.datetime.now()
04430 |
04431 |         # Array Nama Hari Masehi dan Pasaran Jawa
04432 |         hari_masehi = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
04433 |         pasaran_jawa = ["Legi", "Pahing", "Pon", "Wage", "Kliwon"]
04434 |
04435 |         # Ambil nama hari biasa
04436 |         nama_hari = hari_masehi[now.weekday()]
04437 |
04438 |         # Hitung pasaran jawa secara matematis menggunakan fungsi ordinal
04439 |         # Siklus 5 harian terkalibrasi presisi dengan toordinal() % 5
04440 |         nama_pasaran = pasaran_jawa[now.date().toordinal() % 5]
04441 |
04442 |         # Format Tanggal Masehi dengan sisipan pasaran
04443 |         nama_bulan = BULAN_MASEHI[now.month - 1]
04444 |         tanggal_masehi = f"{nama_hari} {nama_pasaran}, {now.day} {nama_bulan} {now.year}"
04445 |
04446 |         # Terapkan ke Label GUI
04447 |         self.lbl_cal_masehi.configure(text=tanggal_masehi)
04448 |
04449 |         # Format Tanggal Hijriah (Tetap seperti semula)
04450 |         hijri_str = HijriConverter.get_hijri_date(now.date())
04451 |         self.lbl_cal_hijri.configure(text=hijri_str)
04452 |
04453 |     def update_conv_ui(self):
04454 |         now = datetime.datetime.now()
04455 |         mode = self.radio_conv_var.get()
04456 |
04457 |         if mode in ["m2h", "m2u"]:
04458 |             self.lbl_conv_date.configure(text="TANGGAL MASEHI")
04459 |             self.combo_conv_month.configure(values=[self.tr(v) for v in BULAN_MASEHI])
04460 |             self.combo_conv_month.set(self.tr(BULAN_MASEHI[now.month - 1]))
04461 |             self.combo_conv_day.configure(values=[str(d).zfill(2) for d in range(1, 32)])
04462 |             self.combo_conv_day.set(f"{now.day:02d}")
04463 |             self.entry_conv_year.delete(0, 'end')
04464 |             self.entry_conv_year.insert(0, str(now.year))
04465 |         else:
04466 |             self.lbl_conv_date.configure(text="TANGGAL HIJRIAH")
04467 |             self.combo_conv_month.configure(values=[self.tr(v) for v in BULAN_HIJRIAH])
04468 |             self.combo_conv_month.set(self.tr("Ramadan"))

```

```

04469 |         self.combo_conv_day.configure(values=[str(d).zfill(2) for d in range(1, 31)])
04470 |
04471 | def build_home_dashboard(self):
04472 |     """Membuat Widget Dashboard Executive Summary untuk Beranda"""
04473 |     dash_container = ctk.CTkFrame(self.textbox, fg_color="transparent")
04474 |
04475 |     # Title Dashboard
04476 |     ctk.CTkLabel(dash_container, text="EXECUTIVE SUMMARY", font=("Segoe UI", 16, "bold"), text_color="#FFD54F").pack(pady=(0, 2))
04477 |
04478 |     # ---> TAMBAHAN: LABEL INFORMASI LOKASI <---
04479 |     lbl_dash_lokasi = ctk.CTkLabel(dash_container, text="Memuat Lokasi...", font=("Segoe UI", 12, "italic"), text_color="#00E5FF")
04480 |     lbl_dash_lokasi.pack(pady=(0, 10))
04481 |     # -----
04482 |
04483 |     grid_frame = ctk.CTkFrame(dash_container, fg_color="transparent")
04484 |     grid_frame.pack(fill="x")
04485 |
04486 |     # BOX 1: Cuaca Lokal
04487 |     box_cuaca = ctk.CTkFrame(grid_frame, fg_color="#121212", corner_radius=8, border_width=1, border_color="#333333")
04488 |     box_cuaca.grid(row=0, column=0, padx=5, sticky="nsew")
04489 |     ctk.CTkLabel(box_cuaca, text="Cuaca Lokal", font=("Segoe UI", 12, "bold"), text_color="#9E9E9E").pack(pady=(10, 5))
04490 |     lbl_dash_cuaca = ctk.CTkLabel(box_cuaca, text="Memuat...", font=("Segoe UI", 14, "bold"))
04491 |     lbl_dash_cuaca.pack(pady=(0, 10), padx=10)
04492 |
04493 |     # BOX 2: Status Hilal Sore Ini (KHGT)
04494 |     box_hilal = ctk.CTkFrame(grid_frame, fg_color="#121212", corner_radius=8, border_width=1, border_color="#333333")
04495 |     box_hilal.grid(row=0, column=1, padx=5, sticky="nsew")
04496 |     ctk.CTkLabel(box_hilal, text="Status Hilal Nanti Sore", font=("Segoe UI", 12, "bold"), text_color="#9E9E9E").pack(pady=(10, 5))
04497 |     lbl_dash_hilal = ctk.CTkLabel(box_hilal, text="Menganalisis...", font=("Segoe UI", 14, "bold"))
04498 |     lbl_dash_hilal.pack(pady=(0, 5), padx=10)
04499 |     lbl_dash_hilal_sub = ctk.CTkLabel(box_hilal, text="Menghitung Ephemeris", font=("Segoe UI", 11), text_color="#9E9E9E")
04500 |     lbl_dash_hilal_sub.pack(pady=(0, 10))
04501 |
04502 |     # BOX 3: Countdown Ijtimak Berikutnya
04503 |     box_ijt = ctk.CTkFrame(grid_frame, fg_color="#121212", corner_radius=8, border_width=1, border_color="#333333")
04504 |     box_ijt.grid(row=0, column=2, padx=5, sticky="nsew")
04505 |     ctk.CTkLabel(box_ijt, text="Countdown Ijtimak", font=("Segoe UI", 12, "bold"), text_color="#9E9E9E").pack(pady=(10, 5))
04506 |     lbl_dash_ijt = ctk.CTkLabel(box_ijt, text="Memuat...", font=("Segoe UI", 14, "bold"))
04507 |     lbl_dash_ijt.pack(pady=(0, 5), padx=10)
04508 |     lbl_dash_ijt_sub = ctk.CTkLabel(box_ijt, text="Mencari fase bulan", font=("Segoe UI", 11), text_color="#9E9E9E")
04509 |     lbl_dash_ijt_sub.pack(pady=(0, 10))
04510 |
04511 |     for i in range(3): grid_frame.grid_columnconfigure(i, weight=1)
04512 |
04513 |     # Thread Background untuk memuat data cerdas
04514 |     def load_dash_data():
04515 |         import time
04516 |         while self.eph is None: time.sleep(1) # Tunggu ephemeris selesai loading
04517 |
04518 |         try:
04519 |             lat = float(self.entry_vlat.get())
04520 |             lon = float(self.entry_vlon.get())
04521 |             tz = float(self.entry_vtz.get())
04522 |         except:
04523 |             lat, lon, tz = -7.0667, 110.4100, 7.0
04524 |
04525 |     # ---> TAMBAHAN: UPDATE TEKS LOKASI <---
04526 |     try:
04527 |         nama_kota = self.opt_city.get()
04528 |         nama_prov = self.opt_prov.get()
04529 |         lokasi_str = f"{nama_kota}, {nama_prov} (Lat: {lat}, Lon: {lon})"
04530 |     except:
04531 |         lokasi_str = f"Koordinat Kustom (Lat: {lat}, Lon: {lon})"
04532 |
04533 |     self.after(0, lambda: lbl_dash_lokasi.configure(text=lokasi_str))
04534 |     # -----
04535 |
04536 |     # 1. Update Cuaca
04537 |     status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)
04538 |     cuaca_pendek = status_cuaca.split("-")[0].strip() # Ambil teks ringkasnya saja
04539 |     self.after(0, lambda: lbl_dash_cuaca.configure(text=cuaca_pendek, text_color=warna_cuaca))

```

```

04540 |
04541 |     try:
04542 |         import math
04543 |         now = datetime.datetime.now()
04544 |
04545 |         # 2. Update Countdown Ijtimak (Menggunakan PyEphem agar sangat cepat)
04546 |         waktu_sekarang = ephem.Date(now)
04547 |         next_nm = ephem.next_new_moon(waktu_sekarang)
04548 |         dt_nm_lokal = next_nm.datetime() + datetime.timedelta(hours=tz)
04549 |         diff = dt_nm_lokal - now
04550 |
04551 |         hari, sisa_detik = diff.days, diff.seconds
04552 |         jam = sisa_detik // 3600
04553 |         teks_ijt = f"{hari} Hari, {jam} Jam lagi"
04554 |         teks_ijt_sub = dt_nm_lokal.strftime("%d %b %Y, %H:%M LT")
04555 |
04556 |         self.after(0, lambda: lbl_dash_ijt.configure(text=teks_ijt, text_color="#FFD54F"))
04557 |         self.after(0, lambda: lbl_dash_ijt_sub.configure(text=teks_ijt_sub))
04558 |
04559 |         # 3. Update Status Hilal Sore Ini (Geosentrik KHGT via Skyfield)
04560 |         earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
04561 |
04562 |         t0 = self.ts.utc(now.year, now.month, now.day, -int(tz))
04563 |         t1 = self.ts.utc(now.year, now.month, now.day, 24 - int(tz))
04564 |         loc = wgs84.latlon(lat, lon)
04565 |
04566 |         t_rs, y_rs = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(self.eph, loc))
04567 |         t_sunset = None
04568 |         for t_ev, y_ev in zip(np.atleast_1d(t_rs), np.atleast_1d(y_rs)):
04569 |             if y_ev == 0:
04570 |                 t_sunset = t_ev
04571 |                 break
04572 |
04573 |         if t_sunset is None:
04574 |             self.after(0, lambda: lbl_dash_hilal.configure(text="Anomali Kutub", text_color="#FF5252"))
04575 |         else:
04576 |             obs_center = earth.at(t_sunset)
04577 |             m_app_geo = obs_center.observe(moon).apparent()
04578 |             s_app_geo = obs_center.observe(sun).apparent()
04579 |
04580 |             elong_geo = s_app_geo.separation_from(m_app_geo).degrees
04581 |
04582 |             ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset)
04583 |             gast = t_sunset.gast
04584 |             lst_deg = (gast * 15.0) + lon
04585 |             ha_deg = lst_deg - (ra_m.hours * 15.0)
04586 |
04587 |             lat_rad = math.radians(lat)
04588 |             ha_rad = math.radians(ha_deg)
04589 |             d_rad = dec_m.radians
04590 |
04591 |             sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
04592 |             alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
04593 |
04594 |             if alt_geo >= 5.0 and elong_geo >= 8.0:
04595 |                 teks_hilal = "☐ WUJUD (KHGT)"
04596 |                 warna_hilal = "#00E676"
04597 |             elif alt_geo > 0:
04598 |                 teks_hilal = "⚠ BAWAH KRITERIA"
04599 |                 warna_hilal = "#FFB74D"
04600 |             else:
04601 |                 teks_hilal = "☐ DI BAWAH UFUK"
04602 |                 warna_hilal = "#FF5252"
04603 |
04604 |             teks_hilal_sub = f"Geo Alt: {alt_geo:.1f}° | Eln: {elong_geo:.1f}°"
04605 |             self.after(0, lambda: lbl_dash_hilal.configure(text=teks_hilal, text_color=warna_hilal))
04606 |             self.after(0, lambda: lbl_dash_hilal_sub.configure(text=teks_hilal_sub))
04607 |
04608 |     except Exception as e:
04609 |         self.after(0, lambda: lbl_dash_hilal.configure(text="Gagal Hitung", text_color="#FF5252"))
04610 |         self.after(0, lambda: lbl_dash_ijt.configure(text="Gagal Hitung", text_color="#FF5252"))

```

```

04611 |         threading.Thread(target=load_dash_data, daemon=True).start()
04612 |         return dash_container
04613 |
04614 |
04615 |     def _translate_release_highlight_piece(self, text, lang):
04616 |         """Terjemahkan highlight lengkap dengan mempertahankan tag/pembatas teks.
04617 |
04618 |         Kamus V8.0 menjadi acuan untuk bagian ini agar cache lama yang masih
04619 |         berbahasa Indonesia atau hanya diterjemahkan sebagian dapat diperbaiki.
04620 |         None berarti teks bukan bagian highlight yang diperbarui.
04621 |         """
04622 |         if lang not in ("en", "ar"):
04623 |             return None
04624 |         raw = text if isinstance(text, str) else str(text)
04625 |         result = raw
04626 |         matched = False
04627 |         for source_text, targets in RELEASE_HIGHLIGHT_TRANSLATIONS.items():
04628 |             if source_text in raw:
04629 |                 result = result.replace(source_text, targets[lang])
04630 |                 matched = True
04631 |         return result if matched else None
04632 |
04633 |     def _translate_release_value_full(self, text):
04634 |         """Terjemahkan prosa rilis memakai bahasa aktif dan cache yang sama."""
04635 |         if not isinstance(text, str) or not text.strip():
04636 |             return text
04637 |         if getattr(self, "current_lang", "id") not in ("en", "ar"):
04638 |             return text
04639 |         return self.translate_release_text_cached(text)
04640 |
04641 |     def _translate_release_piece_preserve_urls(self, text):
04642 |         """Translate prose while leaving URLs unchanged."""
04643 |         if not isinstance(text, str) or not text:
04644 |             return text
04645 |         url_re = re.compile(r'https?://[^\s<>"\']+|www\.[^\s<>"\']+')
04646 |         parts = url_re.split(text)
04647 |         urls = url_re.findall(text)
04648 |         translated = [self._translate_release_value_full(p) for p in parts]
04649 |         out = []
04650 |         for i, part in enumerate(translated):
04651 |             out.append(part)
04652 |             if i < len(urls):
04653 |                 out.append(urls[i])
04654 |         return "".join(out)
04655 |
04656 |     def show_release_info(self):
04657 |         # Penanda khusus: halaman kanan sedang berada pada INFORMASI RILIS.
04658 |         # Saat bahasa ID/EN/AR diganti, halaman ini harus DIGAMBAR ULANG
04659 |         # dengan tag, link, widget, dan layout asli; bukan diisi ulang sebagai teks polos.
04660 |         self.active_right_view = "release_info"
04661 |         self._rendering_release_info = True
04662 |         self._last_raw_report = ""
04663 |
04664 |         # 1. Reset state dan sembunyikan frame lain (Dibuat kebal error / Anti-Crash)
04665 |         self.textbox.grid(row=1, column=0, sticky="nsew")
04666 |         if hasattr(self, 'frame_gerhana_out'): self.frame_gerhana_out.grid_remove()
04667 |         if hasattr(self, 'frame_animasi_out'): self.frame_animasi_out.grid_remove()
04668 |         if hasattr(self, 'frame_3d_out'): self.frame_3d_out.grid_remove()
04669 |         if hasattr(self, 'frame_chart_out'): self.frame_chart_out.grid_remove()
04670 |         if hasattr(self, 'frame_eph3d_out'): self.frame_eph3d_out.grid_remove()
04671 |         if hasattr(self, 'frame_kalender_out'): self.frame_kalender_out.grid_remove()
04672 |         if hasattr(self, 'frame_kalmasehi_out'): self.frame_kalmasehi_out.grid_remove()
04673 |         if hasattr(self, 'frame_winai_out'): self.frame_winai_out.grid_remove()
04674 |         if hasattr(self, 'frame_kriteria_batas_out'): self.frame_kriteria_batas_out.grid_remove()
04675 |         if hasattr(self, 'frame_analemma_out'): self.frame_analemma_out.grid_remove()
04676 |
04677 |         self.anim_running = False
04678 |         self.is_viewing_3d = False
04679 |         if hasattr(self, 'eph3d_is_live'): self.eph3d_is_live = False
04680 |
04681 |         # 2. Pengaturan Dasar Textbox

```

```

04682 |         self.textbox.configure(state="normal", font=("Segoe UI", 16), wrap="word")
04683 |         self.textbox.delete("1.0", "end")
04684 |
04685 |     # 3. Konfigurasi Tag Warna & Tipografi
04686 |     tb = self.textbox._textbox
04687 |
04688 |     # Menggunakan font bawaan OS agar warna asli emoji keluar
04689 |     import platform
04690 |     emoji_font = "Segoe UI Emoji" if platform.system() == "Windows" else "Apple Color Emoji"
04691 |
04692 |     # Tag khusus Emoji (Tanpa warna foreground agar dirender native oleh OS)
04693 |     tb.tag_config("emoji_header", font=(emoji_font, 28), justify="center", spacing1=10, spacing3=5)
04694 |     # Tag khusus Link T3 (Warna Putih Murni)
04695 |     tb.tag_config("link_white", font=("Segoe UI", 16), foreground="#FFFFFF", justify="center", spacing1=5, spacing3=15)
04696 |
04697 |     tb.tag_config("title", font=("Segoe UI", 24, "bold"), foreground="#FFD54F", justify="center", spacing1=5, spacing3=5)
04698 |     tb.tag_config("subtitle", font=("Segoe UI", 20, "bold"), foreground="#00E5FF", justify="center", spacing1=20, spacing3=10)
04699 |     tb.tag_config("h4", font=("Segoe UI", 18, "bold"), foreground="#FFD54F", lmargin1=15, lmargin2=15, spacing1=15, spacing3=5)
04700 |     release_text_justify = "right" if getattr(self, "current_lang", "id") == "ar" else "left"
04701 |     tb.tag_config("body", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=15, lmargin2=15, spacing1=5, spacing3=10, justify=release_text_justify)
04702 |
04703 |     # Format Khusus untuk Daftar Menu
04704 |     tb.tag_config("menu_title", font=("Segoe UI", 18, "bold"), foreground="#00E676", lmargin1=25, lmargin2=25, spacing1=15, spacing3=2, justify=release_text_justify)
04705 |     tb.tag_config("menu_desc", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=45, lmargin2=45, spacing1=2, spacing3=8, justify=release_text_justify)
04706 |
04707 |     tb.tag_config("list_item", font=("Segoe UI", 16), foreground="#E2E8F0", lmargin1=35, lmargin2=55, spacing1=5, spacing3=5, justify=release_text_justify)
04708 |     tb.tag_config("tech", font=("Consolas", 16, "bold"), foreground="#00E676")
04709 |     tb.tag_config("highlight", font=("Segoe UI", 16, "bold"), foreground="#FFD54F")
04710 |     tb.tag_config("box_info", font=("Segoe UI", 16), foreground="#00E5FF", lmargin1=40, lmargin2=60, spacing1=5, spacing3=5)
04711 |     tb.tag_config("box_warn", font=("Segoe UI", 16), foreground="#FFD54F", lmargin1=40, lmargin2=60, spacing1=5, spacing3=5)
04712 |     tb.tag_config("footer", font=("Segoe UI", 14, "italic"), foreground="#ffffff", justify="center")
04713 |
04714 |     # ---> TAG BARU UNTUK CENTER WINDOW <---
04715 |     tb.tag_configure("center_window", justify="center")
04716 |     # Tambahkan baris ini di area konfigurasi tag
04717 |     tb.tag_config("welcome_title", font=("Segoe UI", 22, "bold"), foreground="#00E676", justify="center", spacing1=15, spacing3=10)
04718 |
04719 |     # Pembatas Garis Proporsional (Anti Terpotong Saat Toggle Layar Sempit)
04720 |     garis_sama = " " * 110
04721 |     garis_strip = "-" * 94
04722 |
04723 |     # ===== HEADER =====
04724 |     t1 = "□ □ □ □ □ □ □ □\n"
04725 |     t2 = "INFORMASI RILIS & DOKUMENTASI SISTEM KHGT TIMES V8.0\n"
04726 |
04727 |     self.textbox.insert("end", t1, "emoji_header")
04728 |     self.textbox.insert("end", t2, "title")
04729 |
04730 |     # =====
04731 |     # ---> SISIPAN DASHBOARD EXECUTIVE SUMMARY <---
04732 |     # =====
04733 |     self.textbox.insert("end", "\n")
04734 |     pos_dash = self.textbox.index("end-1c")
04735 |     self.dashboard_widget = self.build_home_dashboard()
04736 |     tb.window_create("end", window=self.dashboard_widget)
04737 |     tb.tag_add("center_window", pos_dash, "end")
04738 |     self.textbox.insert("end", "\n\n")
04739 |     # =====
04740 |
04741 |     # ---> 1. KONFIGURASI TAG LINK UNTUK HEADER <---
04742 |     import webbrowser
04743 |     tb = self.textbox._textbox
04744 |     tb.tag_config("link_aktif_header", foreground="#00B0FF", underline=True)
04745 |     tb.tag_bind("link_aktif_header", "<Button-1>", lambda e: webbrowser.open("https://hisabmu.com/khgttimes/KHGT_Times.zip"))
04746 |     tb.tag_bind("link_aktif_header", "<Enter>", lambda e: self.textbox.configure(cursor="hand2"))
04747 |     tb.tag_bind("link_aktif_header", "<Leave>", lambda e: self.textbox.configure(cursor="xterm"))
04748 |
04749 |     # ---> 2. MASUKKAN TEKS SEBELUM LINK <---
04750 |     self.textbox.insert("end", "Akses dan download melalui link: ", "link_white")
04751 |
04752 |     # ---> 3. MASUKKAN LINK-NYA (Warna Biru & Bisa Diklik) <---

```

```

04753 |         self.textbox.insert("end", "https://hisabmu.com/khggtimes/KHGT_Times.zip", ("link_white", "link_aktif_header"))
04754 |
04755 |         # ---> 4. MASUKKAN GARIS BARU (Enter) <---
04756 |         self.textbox.insert("end", "\n\n", "link_white")
04757 |
04758 |         # ===== INTEGRASI WIDGET KALENDER HIJRIAH INTERAKTIF =====
04759 |         # State Kalender Preview (Deteksi Otomatis Berdasarkan Bulan Masehi Saat Ini)
04760 |         import datetime
04761 |
04762 |         today = datetime.date.today()
04763 |         self.demo_h_year = 1447 # Nilai fallback (cadangan)
04764 |         self.demo_h_month = 9   # Nilai fallback (cadangan)
04765 |
04766 |         # Mencari bulan Hijriah di HIJRI_DB yang beririsan dengan hari ini
04767 |         is_found = False
04768 |         for y, months in HIJRI_DB.items():
04769 |             for m, m_data in enumerate(months):
04770 |                 nama_bulan, _, start_date_str, jumlah_hari = m_data
04771 |                 parts = start_date_str.split('-')
04772 |                 bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
04773 |                             "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
04774 |
04775 |                 start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
04776 |                 end_date = start_date + datetime.timedelta(days=jumlah_hari - 1)
04777 |
04778 |                 # Jika tanggal hari ini berada dalam rentang bulan Hijriah tersebut
04779 |                 if start_date <= today <= end_date:
04780 |                     self.demo_h_year = y
04781 |                     self.demo_h_month = m
04782 |                     is_found = True
04783 |                     break
04784 |             if is_found:
04785 |                 break
04786 |
04787 |         self.demo_cal_wrapper = ctk.CTkFrame(self.textbox, fg_color="transparent")
04788 |
04789 |         # Header Navigasi Kalender (Tengah)
04790 |         demo_header = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="transparent")
04791 |         demo_header.pack(pady=(5, 15))
04792 |
04793 |         def demo_prev():
04794 |             self.demo_h_month -= 1
04795 |             if self.demo_h_month < 0:
04796 |                 self.demo_h_month = 11
04797 |                 self.demo_h_year -= 1
04798 |             if self.demo_h_year not in HIJRI_DB:
04799 |                 self.demo_h_year += 1
04800 |                 self.demo_h_month = 0
04801 |             return
04802 |         render_demo()
04803 |
04804 |         def demo_next():
04805 |             self.demo_h_month += 1
04806 |             if self.demo_h_month > 11:
04807 |                 self.demo_h_month = 0
04808 |                 self.demo_h_year += 1
04809 |             if self.demo_h_year not in HIJRI_DB:
04810 |                 self.demo_h_year -= 1
04811 |                 self.demo_h_month = 11
04812 |             return
04813 |         render_demo()
04814 |
04815 |         btn_prev = ctk.CTkButton(demo_header, text="◀ Sebelumnya", font=("Segoe UI", 12, "bold"), width=110, fg_color="#1F1F1F", hover_color="#333333", command=demo_prev)
04816 |         btn_prev.pack(side="left", padx=10)
04817 |
04818 |         self.lbl_demo_title = ctk.CTkLabel(demo_header, text="MEMUAT...", font=("Segoe UI", 18, "bold"), text_color="#FFD54F", justify="center")
04819 |         self.lbl_demo_title.pack(side="left", expand=True)
04820 |
04821 |         btn_next = ctk.CTkButton(demo_header, text="Berikutnya ▶", font=("Segoe UI", 12, "bold"), width=110, fg_color="#1F1F1F", hover_color="#333333", command=demo_next)
04822 |         btn_next.pack(side="right", padx=10)
04823 |

```



```

04824 | self.demo_cal_grid = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="#050510", corner_radius=10, border_width=1, border_color="#333333")
04825 | self.demo_cal_grid.pack(padx=10, pady=(0, 10))
04826 |
04827 | # --- TAMBAHAN: Frame Legend Hari Penting Islam ---
04828 | self.demo_frame_legend = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="#1E1E1E", corner_radius=8)
04829 | self.demo_frame_legend.pack(pady=(5, 5))
04830 |
04831 | ctk.CTkLabel(self.demo_frame_legend, text="Legenda:", font=("Segoe UI", 12, "bold"), text_color="#FFD54F").pack(side="left", padx=15, pady=10)
04832 |
04833 | # Helper function untuk menambah item legend
04834 | def add_legend_item(parent, color, text):
04835 |     container = ctk.CTkFrame(parent, fg_color="transparent")
04836 |     container.pack(side="left", padx=10, pady=10)
04837 |     dot = ctk.CTkFrame(container, width=12, height=12, corner_radius=6, fg_color=color)
04838 |     dot.pack(side="left", padx=(0, 5))
04839 |     ctk.CTkLabel(container, text=text, font=("Segoe UI", 11)).pack(side="left")
04840 |
04841 | add_legend_item(self.demo_frame_legend, "#FF5252", "Hari Haram Puasa / Hari Raya")
04842 | add_legend_item(self.demo_frame_legend, "#00E676", "Puasa Sunnah Utama")
04843 | add_legend_item(self.demo_frame_legend, "#00B0FF", "Puasa Ayyamul Bidh")
04844 | add_legend_item(self.demo_frame_legend, "#FFA000", "Puasa Senin Kamis")
04845 | add_legend_item(self.demo_frame_legend, "#8BC34A", "Ramadhan")
04846 | add_legend_item(self.demo_frame_legend, "#E040FB", "Hari Besar Islam")
04847 |
04848 | # ---> TAMBAHAN: KETERANGAN PARAMETER ASTRONOMI & LOKASI <---
04849 | self.demo_frame_keterangan = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="transparent")
04850 | self.demo_frame_keterangan.pack(pady=(0, 10))
04851 |
04852 | teks_ket = "* Kiri: Umur Bulan (Atas), Elongasi Geo (Bawah) | Kanan: Fraksi Iluminasi (Atas), Tinggi Hilal Geo (Bawah). Dihitung saat Maghrib."
04853 | ctk.CTkLabel(self.demo_frame_keterangan, text=teks_ket, font=("Segoe UI", 13, "italic"), text_color="ffffff").pack()
04854 |
04855 | # Label Dinamis untuk Tanggal, Kota, dan Koordinat
04856 | self.lbl_demo_lokasi = ctk.CTkLabel(self.demo_frame_keterangan, text="" Memuat data lokasi dan tanggal...", font=("Segoe UI", 14, "bold"), text_color="#00E5FF")
04857 | self.lbl_demo_lokasi.pack(pady=(5, 0))
04858 |
04859 | # =====
04860 | # ---> TAMBAHAN FITUR PRINT (EKSPORT) KHUSUS DEMO KALENDER
04861 | # =====
04862 | self.demo_frame_export = ctk.CTkFrame(self.demo_cal_wrapper, fg_color="transparent")
04863 | self.demo_frame_export.pack(pady=(15, 5))
04864 |
04865 | btn_demo_pdf = ctk.CTkButton(self.demo_frame_export, text="" Cetak PDF (Portrait)", font=("Segoe UI", 13, "bold"), command=lambda: self.export_demo_kalender("pdf"), fg_color="#D32F2F",
04866 | hover_color="#B71C1C")
04867 | btn_demo_pdf.pack(side="left", padx=10)
04868 |
04869 | btn_demo_png = ctk.CTkButton(self.demo_frame_export, text="" Simpan PNG (Portrait)", font=("Segoe UI", 13, "bold"), command=lambda: self.export_demo_kalender("png"), fg_color="#F57C00",
04870 | hover_color="#E65100")
04871 | btn_demo_png.pack(side="left", padx=10)
04872 | # =====
04873 | if not hasattr(self, '_demo_moon_imgs'):
04874 |     self._demo_moon_imgs = []
04875 |
04876 | def render_demo():
04877 |     # --- PENGAMAN STARTUP: TUNGGU EPHEMERIS SELESAI DIMUAT DARI BACKGROUND THREAD ---
04878 |     if getattr(self, 'eph', None) is None:
04879 |         for w in self.demo_cal_grid.winfo_children():
04880 |             w.destroy()
04881 |         ctk.CTkLabel(self.demo_cal_grid, text="" Sedang Memuat Mesin Ephemeris NASA, harap tunggu...", font=("Segoe UI", 14, "italic"), text_color="#FFD54F").grid(row=0, column=0, padx=50, pady=50)
04882 |         self.after(500, render_demo) # Ulangi / Cek lagi 0.5 detik kemudian
04883 |         return
04884 |     # -----
04885 |     for w in self.demo_cal_grid.winfo_children():
04886 |         w.destroy()
04887 |
04888 |     y = self.demo_h_year
04889 |     m = self.demo_h_month
04890 |     if y not in HIJRI_DB: return
04891 |
04892 |     m_data = HIJRI_DB[y][m]

```

```

04893 |         nama_bulan, _, start_date_str, jumlah_hari = m_data
04894 |
04895 |         parts = start_date_str.split('-')
04896 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
04897 |         start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
04898 |         end_date = start_date + datetime.timedelta(days=jumlah_hari-1)
04899 |
04900 |         hijri_t = f"{nama_bulan.upper()} {y} H"
04901 |         greg_t = f"{BULAN_MASEHI[start_date.month-1].upper()} {start_date.year}"
04902 |         if start_date.month != end_date.month:
04903 |             greg_t += f" - {BULAN_MASEHI[end_date.month-1].upper()} {end_date.year}"
04904 |
04905 |         self.lbl_demo_title.configure(text=f"{hijri_t}\n({greg_t})")
04906 |
04907 |         # ---> UPDATE INFORMASI TANGGAL, KOTA, DAN KOORDINAT KE LAYAR <---
04908 |         try:
04909 |             lat_val = float(self.entry_vlat.get())
04910 |             lon_val = float(self.entry_vlon.get())
04911 |             elev_val = float(self.entry_velev.get())
04912 |             tz_val = float(self.entry_vtz.get())
04913 |             nama_kota = self.opt_city.get()
04914 |             nama_prov = self.opt_prov.get()
04915 |         except:
04916 |             lat_val, lon_val, elev_val, tz_val = -7.0667, 110.4100, 230.0, 7.0
04917 |             nama_kota = "Semarang"
04918 |             nama_prov = "Jawa Tengah"
04919 |
04920 |         teks_info_bawah = f"Kota: {nama_kota}, {nama_prov} | Koordinat: (Lat: {lat_val}, Lon: {lon_val})"
04921 |         self.lbl_demo_lokasi.configure(text=teks_info_bawah)
04922 |         # -----
04923 |
04924 |         days_header = ["AHAD", "SENIN", "SELASA", "RABU", "KAMIS", "JUMAT", "SABTU"]
04925 |         for col, day_name in enumerate(days_header):
04926 |             txt_color = "#FF5252" if col == 0 else ("00E676" if col == 5 else "FFFFFF")
04927 |             lbl = ctk.CTkLabel(self.demo_cal_grid, text=day_name, font=("Segoe UI", 14, "bold"), text_color=txt_color)
04928 |             lbl.grid(row=0, column=col, padx=4, pady=(5, 2), sticky="nsw")
04929 |
04930 |         offset = (start_date.weekday() + 1) % 7
04931 |         row_cal, col_cal = 1, offset
04932 |         current_g_date = start_date
04933 |
04934 |         self._demo_moon_imgs.clear()
04935 |
04936 |         import ephem
04937 |         import math
04938 |         import pytz
04939 |         from skyfield import almanac
04940 |
04941 |         obs_ephem = ephem.Observer()
04942 |         obs_ephem.lat = str(lat_val)
04943 |         obs_ephem.lon = str(lon_val)
04944 |         obs_ephem.elevation = elev_val
04945 |         obs_ephem.pressure = 1010
04946 |         obs_ephem.temp = 25
04947 |
04948 |         matahari_ephem = ephem.Sun()
04949 |
04950 |         # Persiapkan objek Skyfield untuk akurasi tertinggi
04951 |         earth_sf = self.eph['earth']
04952 |         sun_sf = self.eph['sun']
04953 |         moon_sf = self.eph['moon']
04954 |
04955 |         # =====
04956 |         # ---> TAMBAHAN: KALKULASI FASE BULAN UTAMA UNTUK SEBULAN PENUH <---
04957 |         # =====
04958 |         t0_month = self.ts.utc(start_date.year, start_date.month, start_date.day, -int(tz_val))
04959 |         t1_month = self.ts.utc(end_date.year, end_date.month, end_date.day, 24 - int(tz_val))
04960 |         t_phases, y_phases = almanac.find_discrete(t0_month, t1_month, almanac.moon_phases(self.eph))
04961 |
04962 |         phase_events = {}
04963 |         if t_phases is not None:

```

```

04964 |         for t_obj, p_idx in get_safe_events(t_phases, y_phases):
04965 |             dt_loc = t_obj.utc_datetime() + datetime.timedelta(hours=tz_val)
04966 |             phase_events[dt_loc.date()] = int(p_idx)
04967 |
04968 |     # Map index fase ke nama string
04969 |     phase_names = {0: "New Moon", 1: "First Quarter", 2: "Full Moon", 3: "Last Quarter"}
04970 |     # =====
04971 |
04972 |     for h_day in range(1, jumlah_hari + 1):
04973 |         # --- 1. PENANDA HARI INI (CURRENT DAY HIGHLIGHT) ---
04974 |         is_today = (current_g_date == datetime.date.today())
04975 |
04976 |         if is_today:
04977 |             b_color = "#FFEA00"
04978 |             b_width = 2
04979 |         else:
04980 |             b_color = "#FF5252" if col_cal == 0 else ("00E676" if col_cal == 5 else "#555555")
04981 |             b_width = 1.0 if (col_cal == 0 or col_cal == 5) else 1
04982 |
04983 |         # Height diubah ke 85 agar teks fase bulan di bawah tidak terpotong
04984 |         cell = ctk.CTkFrame(self.demo_cal_grid, fg_color="#0F172A", corner_radius=6, border_width=b_width, border_color=b_color, width=200, height=85)
04985 |         cell.grid(row=row_cal, column=col_cal, padx=10, pady=2, sticky="nsew")
04986 |         cell.grid_propagate(False)
04987 |
04988 |         frame_top = ctk.CTkFrame(cell, fg_color="transparent")
04989 |         frame_top.pack(fill="x", padx=10, pady=(5, 0))
04990 |
04991 |         # --- KALKULASI DATA ASTRONOMI SAAT MAGHRIB (SUNSET) LOKAL ---
04992 |         dt_noon_local = datetime.datetime(current_g_date.year, current_g_date.month, current_g_date.day, 12, 0, 0)
04993 |         dt_noon_utc = dt_noon_local - datetime.timedelta(hours=tz_val)
04994 |         obs_ephem.date = ephemeris.Date(dt_noon_utc.strftime("%Y/%m/%d %H:%M:%S"))
04995 |
04996 |         try:
04997 |             waktu_sunset_ephem = obs_ephem.next_setting(matahari_ephem)
04998 |         except:
04999 |             waktu_sunset_ephem = obs_ephem.date + 0.25
05000 |
05001 |         dt_sunset_utc = waktu_sunset_ephem.datetime().replace(tzinfo=pytz.utc)
05002 |         t_sunset = self.ts.from_datetime(dt_sunset_utc)
05003 |
05004 |         geo_earth = earth_sf.at(t_sunset)
05005 |         app_moon_geo = geo_earth.observe(moon_sf).apparent()
05006 |         app_sun_geo = geo_earth.observe(sun_sf).apparent()
05007 |
05008 |         sep_deg = app_sun_geo.separation_from(app_moon_geo).degrees
05009 |         elongasi = sep_deg.item() if hasattr(sep_deg, 'item') else sep_deg
05010 |
05011 |         ra_moon, dec_moon, _ = app_moon_geo.radec(epoch=t_sunset)
05012 |         gast = t_sunset.gast
05013 |         lst_deg = (gast * 15.0) + lon_val
05014 |
05015 |         ra_h = ra_moon.hours.item() if hasattr(ra_moon.hours, 'item') else ra_moon.hours
05016 |         dec_r = dec_moon.radians.item() if hasattr(dec_moon.radians, 'item') else dec_moon.radians
05017 |         ha_deg = lst_deg - (ra_h * 15.0)
05018 |         lat_rad, ha_rad = math.radians(lat_val), math.radians(ha_deg)
05019 |
05020 |         sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) * math.cos(ha_rad)
05021 |         tinggi_hilal = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
05022 |
05023 |         try:
05024 |             prev_nm = ephemeris.previous_new_moon(waktu_sunset_ephem)
05025 |             umur_bulan = waktu_sunset_ephem - prev_nm
05026 |         except:
05027 |             umur_bulan = 0.0
05028 |
05029 |         illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
05030 |         iluminasi = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0
05031 |
05032 |         box_kiri = ctk.CTkFrame(frame_top, fg_color="transparent")
05033 |         box_kiri.pack(side="left", anchor="nw")
05034 |

```

```

05035 |         lbl_h = ctk.CTkLabel(box_kiri, text=str(h_day), font=("Segoe UI", 24, "bold"), text_color="#FFFFFF")
05036 |         lbl_h.pack(anchor="w")
05037 |
05038 |         lbl_age = ctk.CTkLabel(box_kiri, text=f"{umur_bulan:.1f}", font=("Consolas", 9, "bold"), text_color="#FFFFFF")
05039 |         lbl_age.pack(anchor="w", pady=(0, 0))
05040 |
05041 |         lbl_elong = ctk.CTkLabel(box_kiri, text=f"{elongasi:.1f}°", font=("Consolas", 9, "bold"), text_color="#FFFFFF")
05042 |         lbl_elong.pack(anchor="w", pady=(0, 0))
05043 |
05044 |         box_kanan = ctk.CTkFrame(frame_top, fg_color="transparent")
05045 |         box_kanan.pack(side="right", anchor="ne")
05046 |
05047 |         try:
05048 |             angle = (h_day / 29.53059) * 360.0
05049 |             moon_idx = int(angle) % 360
05050 |             filepath = os.path.join(BASE_DIR, "moon", f"m{moon_idx:03d}.png")
05051 |             if os.path.exists(filepath):
05052 |                 from PIL import Image, ImageDraw
05053 |                 img = Image.open(filepath).convert("RGBA")
05054 |                 mask = Image.new('L', img.size, 0)
05055 |                 draw = ImageDraw.Draw(mask)
05056 |                 draw.ellipse((0, 0) + img.size, fill=255)
05057 |                 img.putalpha(mask)
05058 |
05059 |                 ctk_img = ctk.CTkImage(img, size=(24, 24))
05060 |                 self._demo_moon_imgs.append(ctk_img)
05061 |                 lbl_moon = ctk.CTkLabel(box_kanan, image=ctk_img, text="")
05062 |             else:
05063 |                 lbl_moon = ctk.CTkLabel(box_kanan, text="☐", font=("Segoe UI", 14))
05064 |         except:
05065 |             lbl_moon = ctk.CTkLabel(box_kanan, text="☐", font=("Segoe UI", 14))
05066 |
05067 |         lbl_moon.pack(anchor="e", pady=(2, 0))
05068 |
05069 |         lbl_ilu = ctk.CTkLabel(box_kanan, text=f"{iluminasi:.1f}%", font=("Consolas", 9, "bold"), text_color="#FFFFFF")
05070 |         lbl_ilu.pack(anchor="e", pady=(0, 0))
05071 |
05072 |         lbl_alt = ctk.CTkLabel(box_kanan, text=f"{tinggi_hilal:.1f}°", font=("Consolas", 9, "bold"), text_color="#FFFFFF")
05073 |         lbl_alt.pack(anchor="e", pady=(0, 0))
05074 |
05075 |         frame_bottom = ctk.CTkFrame(cell, fg_color="transparent")
05076 |         frame_bottom.pack(expand=True, fill="both", padx=10, pady=(2, 8))
05077 |
05078 |         m_str = f"{current_g_date.day} {BULAN_MASEHI[current_g_date.month-1].upper()} {current_g_date.year}"
05079 |         lbl_g = ctk.CTkLabel(frame_bottom, text=m_str, font=("Segoe UI", 10, "bold"), text_color="#FFD54F")
05080 |         lbl_g.pack(expand=True, anchor="center")
05081 |
05082 |         # =====
05083 |         # --> TAMBAHAN WIDGET FASE BULAN DI BAWAH TANGGAL MASEHI <--
05084 |         # =====
05085 |         if current_g_date in phase_events:
05086 |             phase_text = phase_names[phase_events[current_g_date]]
05087 |             lbl_phase = ctk.CTkLabel(frame_bottom, text=phase_text, font=("Consolas", 9, "bold"), text_color="#00E5FF")
05088 |             lbl_phase.pack(expand=True, anchor="center", pady=(0, 2))
05089 |         # =====
05090 |
05091 |         event_color, _ = self._get_islamic_event(h_day, m + 1, current_g_date.weekday())
05092 |         if event_color:
05093 |             dot = ctk.CTkFrame(frame_bottom, width=6, height=6, corner_radius=3, fg_color=event_color)
05094 |             dot.pack(side="bottom", pady=(0, 2))
05095 |
05096 |         col_cal += 1
05097 |         if col_cal > 6:
05098 |             col_cal = 0
05099 |             row_cal += 1
05100 |         current_g_date += datetime.timedelta(days=1)
05101 |
05102 |         for i in range(7): self.demo_cal_grid.grid_columnconfigure(i, weight=1)
05103 |
05104 |         render_demo()
05105 |

```

```

05106 |         # --- CARA MEMBUAT WIDGET CENTER DI DALAM TEXTBOX ---
05107 |         self.textbox.insert("end", "\n")
05108 |         pos_awal = self.textbox.index("end-1c")
05109 |
05110 |         tb.window_create("end", window=self.demo_cal_wrapper)
05111 |         self.textbox.insert("end", "\n\n")
05112 |
05113 |         # Menerapkan tag agar rata tengah (Center)
05114 |         tb.tag_add("center_window", pos_awal, "end")
05115 |         # =====
05116 |
05117 |         # ===== INTRO =====
05118 |         welcome_text = "Selamat datang di KHGT Times V8.0\n"
05119 |         intro_1 = ("Aplikasi ini dikembangkan secara khusus sebagai perangkat lunak "
05120 |                    "komprehensif dan mutakhir untuk perhitungan astrometri presisi tinggi, pemetaan visibilitas hilal global, "
05121 |                    "dan penyatuan penanggalan Islam internasional serta menjadi sarana edukasi publik terkait ilmu falak Islami dan transisi penerapan KHGT di masa depan. "
05122 |                    "Pada V8.0, sistem menyediakan 41 modul menu terintegrasi, antarmuka multibahasa Indonesia-Inggris-Arab, Peta Langit Dunia online pada Menu 39, akses ke aplikasi pendamping Laporan
Astronomi pada Menu 40, serta HIJRI_DB Auto-Builder (Admin) pada Menu 41.\n")
05123 |
05124 |         self.textbox.insert("end", welcome_text, "welcome_title")
05125 |         self.textbox.insert("end", intro_1, "body")
05126 |
05127 |         self.textbox.insert("end", "Melampaui Batas Pendahulunya (Beyond Accurate Times)\n", "h4")
05128 |         intro_2 = ("Selama bertahun-tahun, dunia falak Islam sangat bergantung pada perangkat lunak pionir pendahulu. Aplikasi tersebut sangat berjasa meletakkan standar dasar komputasi waktu salat dan
peta hilal di era awal komputasi.\n"
05129 |                    "Namun, generasi pendahulu tersebut umumnya dibangun di atas teori analitik klasik (algoritma VSOP87), menggunakan arsitektur single-thread yang lambat, dan dikonsep murni untuk rukyatul
hilal lokal/zonal, bukan untuk integrasi kalender global.\n")
05130 |         self.textbox.insert("end", intro_2, "body")
05131 |
05132 |         self.textbox.insert("end", "Mengapa KHGT Times Hadir dan Sangat Penting?\n", "h4")
05133 |         intro_3 = ("KHGT Times hadir sebagai Quantum Leap (Lompatan Besar) komputasi untuk menjawab tantangan astronomi modern dan urgensi implementasi Kalender Hijriah Global Tunggal (KHGT) di seluruh
dunia. Keunggulan absolut sistem ini meliputi:\n")
05134 |         self.textbox.insert("end", intro_3, "body")
05135 |
05136 |         self.textbox.insert("end", "> ", "highlight")
05137 |         self.textbox.insert("end", "[1] Presisi Ephemeris JPL NASA: ", "tech")
05138 |         self.textbox.insert("end", "Meninggalkan kalkulasi rumus hampiran matematika klasik, beralih penuh ke integrasi numerik orbit astronomis real-time dari Development Ephemeris (DE) NASA.\n",
"list_item")
05139 |
05140 |         self.textbox.insert("end", "> ", "highlight")
05141 |         self.textbox.insert("end", "[2] Pemindai Kesatuan Matlak Global: ", "tech")
05142 |         self.textbox.insert("end", "Sistem memindai (scanning) ratusan kota di seluruh benua dalam hitungan detik untuk melacak Titik Pertama pemenuhan kriteria KHGT (PKG 1 & PKG 2) di daratan utama
bumi.\n", "list_item")
05143 |
05144 |         self.textbox.insert("end", "> ", "highlight")
05145 |         self.textbox.insert("end", "[3] Pemisahan Ruang Geosentris & Toposentris: ", "tech")
05146 |         self.textbox.insert("end", "Menghitung secara simultan dan independen parameter dari inti bumi (Geosentris) yang disyaratkan KHGT, dan dari mata pengamat (Toposentris) yang disyaratkan MABIMS.\n",
"list_item")
05147 |
05148 |         intro_4 = ("\nDengan arsitektur ini, KHGT Times adalah sebuah karya kontemporer Falak Digital yang didedikasikan untuk observatorium, lembaga riset akademik, dan pengambil kebijakan syariah di era
modern.\n")
05149 |         self.textbox.insert("end", intro_4, "body")
05150 |
05151 |         # ===== BAGIAN 1: TEKNOLOGI & ENGINE =====
05152 |         self.textbox.insert("end", "\n" + garis_strip + "\n[ 1. REQUIREMENTS LIBRARY & ENGINE ASTROMETRI ]\n" + garis_strip + "\n", "subtitle")
05153 |
05154 |         intro_tech = ("Sistem ini menggabungkan berbagai pustaka (library) saintifik standar industri untuk mencapai keseimbangan antara akurasi tingkat tinggi (High Precision) dan kecepatan pemrosesan
(High Performance).\n")
05155 |         self.textbox.insert("end", intro_tech, "body")
05156 |
05157 |         techs = [
05158 |             ("Skyfield API", "Library Python astronomi modern. Bertugas menghitung posisi benda langit dengan akurasi setara USNO. Mengeksekusi rotasi koordinat Geosentrik (pusat bumi) maupun Toposentrik
(permukaan bumi) dengan koreksi refraksi, nutasi, presesi, dan aberrasi cahaya."),
05159 |             ("JPL Ephemeris", "Bukan library kode, melainkan Basis Data fisik (.bsp) dari NASA/JPL. Sistem V8.0 memakai de440s.bsp secara otomatis untuk tahun 1850-2149 dan de441-new.bsp untuk tahun di
luar rentang tersebut (hingga batas aplikasi 1-2999 M), sehingga fokus Matahari-Bulan era modern menggunakan kernel yang lebih ringkas tanpa kehilangan jangkauan panjang."),
05160 |             ("PyEphem (ephem)", "Library engine C-based turunan XEphem. Dipakai khusus sebagai akselerator kalkulasi brutal (Brute-Force), seperti iterasi mencari Waktu Terbenam (Sunset) dan Ijtimak Global
di ratusan kota pada Modul 3 dan Modul 5 secara instan."),
05161 |             ("SciPy (scipy.interpolate)", "Library komputasi matematika kompleks. Bertugas menggunakan algoritma 'Bicubic Griddata' untuk merajut puluhan titik data kasar (Latitude/Longitude) menjadi Peta
Heatmap Visual (Crescent Visibility HD Map) yang mulus (Seamless) di Modul 2."),
05162 |             ("NumPy", "Library manipulasi matriks. Digunakan berdampingan dengan SciPy dan Matplotlib untuk memproses Array Kalkulasi 50 Tahun serta data Linspace (pengiris waktu per detik) tanpa membebani
RAM CPU."),

```

```

05163 |         ("Matplotlib", "Engine rendering visual grafis. Membangun ruang kanvas 3D Spasial, kurva garis ketinggian harian (Altitude Chart), dan instrumen kompas bayangan kutub (Mizwala)."),
05164 |         ("CustomTkinter & Tk", "Framework Graphical User Interface (GUI). Menyulap kode terminal Python menjadi aplikasi Desktop yang modern, memiliki Dark-Mode, Sidebar interaktif, dan Frame yang
dinamis."),
05165 |         ("Pillow (PIL)", "Engine pemroses gambar raster digital. Bertugas me-render dan melukis (ImageDraw) secara internal pada modul Generator Gambar Jadwal Shalat, Kalender Hijriah/Masehi beresolusi
tinggi, serta kalkulasi masking rotasi fase bulan."),
05166 |         ("Requests & Bs4", "Library koneksi HTTP. Digunakan pada Modul WinAI untuk menjangkau API Gemini Google, sekaligus melakukan 'Scraping' otomatis (Google Search/DuckDuckGo) secara Real-time saat
AI butuh fakta terbaru di luar database."),
05167 |         ("FPDF & CSV", "Library manajemen format data. FPDF mencetak String text dari terminal ke file dokumen PDF yang terstruktur, sementara CSV memecah string tabel komparasi 50 tahun menjadi
pemisah titik-koma (;) agar siap dibuka di Microsoft Excel.")
05168 |     ]
05169 |
05170 |     for name, desc in techs:
05171 |         self.textbox.insert("end", "> ", "highlight")
05172 |         self.textbox.insert("end", f"{name} : ", "tech")
05173 |         self.textbox.insert("end", f"{desc}\n\n", "list_item")
05174 |
05175 |     # ===== BAGIAN 2: KONSEP DASAR KHGT =====
05176 |     self.textbox.insert("end", "\n" + garis_strip + "\n[ 2. KONSEP DASAR, PRINSIP & FORMULASI KHGT ]\n" + garis_strip + "\n\n", "subtitle")
05177 |
05178 |     b2_intro = ("Kalender Hijriah Global Tunggal (KHGT) merupakan tonggak peradaban sistem penanggulangan Islam unifikatif hasil Muktamar Internasional Turki 2016. Berbeda dengan sistem lokal/zonal, KHGT
bertujuan menyatukan umat Islam di bawah satu sistem waktu global.\n\n")
05179 |     self.textbox.insert("end", b2_intro, "body")
05180 |
05181 |     self.textbox.insert("end", "3 Prinsip Utama KHGT:\n", "h4")
05182 |
05183 |     self.textbox.insert("end", "> ", "highlight")
05184 |     self.textbox.insert("end", "Kesatuan Matlak (Ittihad al-Matali'): ", "tech")
05185 |     self.textbox.insert("end", "Seluruh bumi dianggap sebagai satu kesatuan zona wilayah.\n", "list_item")
05186 |
05187 |     self.textbox.insert("end", "> ", "highlight")
05188 |     self.textbox.insert("end", "Kaidah Transfer Visibilitas (Naql al-Rukyat): ", "tech")
05189 |     self.textbox.insert("end", "Jika hilal secara astronomis mungkin dirukyat di SATU titik daratan di bumi, maka visibilitas tersebut sah dan berlaku untuk SELURUH dunia.\n", "list_item")
05190 |
05191 |     self.textbox.insert("end", "> ", "highlight")
05192 |     self.textbox.insert("end", "Keterpaduan Hari: ", "tech")
05193 |     self.textbox.insert("end", "Bulan Hijriah dimulai serentak di seluruh dunia tanpa ada negara yang berbeda tanggal.\n\n", "list_item")
05194 |
05195 |     self.textbox.insert("end", "Parameter Kriteria (Imkanur Rukyat Istanbul 2016):\n", "h4")
05196 |     self.textbox.insert("end", "1. Tinggi Hilal Geosentris (Geocentric Altitude) minimal 5 derajat (Alt >= 5°).\n", "box_info")
05197 |     self.textbox.insert("end", "2. Sudut Elongasi Geosentris (Geocentric Elongation) minimal 8 derajat (Elong >= 8°).\n\n", "box_info")
05198 |
05199 |     self.textbox.insert("end", "Mekanisme Eksekusi (Parameter Kalender Global / PKG):\n", "h4")
05200 |     self.textbox.insert("end", "1) [PKG 1] Kondisi Normal (Wilayah Timur hingga Tengah Bumi):\n", "highlight")
05201 |     self.textbox.insert("end", "Sistem memindai apakah kriteria visibilitas (Alt >= 5° & Elong >= 8°) terpenuhi di daratan mana pun di bumi SEBELUM pukul 00:00 UTC. Jika terpenuhi, maka besok adalah
awal bulan baru secara global.\n\n", "box_info")
05202 |     self.textbox.insert("end", "2) [PKG 2] Kondisi Kritis (Wilayah Ekstrem Barat / Benua Amerika):\n", "highlight")
05203 |     self.textbox.insert("end", "Jika kriteria visibilitas baru terpenuhi SETELAH pukul 00:00 UTC, maka sistem otomatis membandingkan waktu Ijtimak (Konjungsi) terhadap Terbit Fajar di Gisborne,
Selandia Baru.\n", "box_warn")
05204 |     self.textbox.insert("end", "• Jika Ijtimak SEBELUM fajar Gisborne: Maka besok tetap masuk bulan baru.\n", "list_item")
05205 |     self.textbox.insert("end", "• Jika Ijtimak SETELAH fajar Gisborne: Maka masuk bulan baru DITUNDA lusa (Lihat hasil hitung pada tanggal = ijtimak + 1) agar wilayah Timur bumi tidak berpuasa sebelum
ijtimak terjadi.\n", "list_item")
05206 |
05207 |     # ===== BAGIAN 3: DAFTAR FITUR LENGKAP (41 MENU) =====
05208 |     self.textbox.insert("end", "\n" + garis_strip + "\n[ 3. MANUAL BOOK: PENJELASAN 41 MODUL MENU V8.0 ]\n" + garis_strip + "\n", "subtitle")
05209 |     self.textbox.insert("end", "Gunakan Dropdown Menu di Sidebar Kiri untuk berpindah antar modul. Berikut rinciannya:\n", "body")
05210 |
05211 |     detailed_features = [
05212 |         ("1) Kalender Hijriah Berjalan",
05213 |         "• Kegunaan: Kalender visual interaktif yang terintegrasi dengan database KHGT.\n"
05214 |         "Menyajikan tampilan kalender bulanan yang dinamis. Dilengkapi dengan penanda hari penting Islam, puasa sunnah, serta informasi astronomi hilal (umur, elongasi, iluminasi, tinggi) yang
dihitung otomatis saat maghrib.\n"
05215 |         "• Sub-Menu Input: Pilih Bulan dan Tahun Hijriah.\n"
05216 |         "• Aksi & Output: Grid visual kalender HD dengan sistem navigasi offset otomatis. Dilengkapi tombol cetak PDF / PNG Resolusi Tinggi."),
05217 |
05218 |         ("2) Kalender Masehi Berjalan",
05219 |         "• Kegunaan: Kalender visual penanggulangan sipil internasional.\n"
05220 |         "Menampilkan matriks kalender Gregorian dengan penyematan tanggal KHGT sebagai pelengkap. Menghitung rotasi fase bulan visual untuk setiap harinya.\n"
05221 |         "• Sub-Menu Input: Pilih Bulan dan Tahun Masehi.\n"
05222 |         "• Aksi & Output: Mesin render grafis kalender yang mendukung ekspor dokumen cetak."),
05223 |
05224 |         ("3) Konversi Kalender",

```

```

05225 |         "* Kegunaan: Transformasi tanggal antara Masehi (Gregorian/Julian) dan Hijriah (KHGT).\n"
05226 |         "Melakukan konversi tanggal dengan akurasi tinggi menggunakan database KHGT yang sudah terintegrasi. Menghitung secara astronomis jatuhnya fase bulan baru (bukan rumus pembagian hari
manual).\n"
05227 |         "* Sub-Menu Input: Mode Konversi (M2H / H2M), Input Tanggal, Koreksi Hari Ikhtiyat.\n"
05228 |         "* Aksi & Output: Tampilan tanggal hasil konversi lengkap dengan nama hari dan hari pasaran."),
05229 |
05230 |         ("4) Visibility Hilal (KHGT)",
05231 |         "* Kegunaan: Analisis komprehensif visibilitas hilal berdasarkan multi-kriteria global.\n"
05232 |         "Modul ini melakukan perhitungan astrometri presisi tinggi untuk membandingkan posisi Bulan dan Matahari saat terbenam. Sistem mengevaluasi data dari dua sudut pandang: Geosentrik (pusat bumi)
dan Toposentrik (permukaan bumi/lokasi pengamat), guna memberikan akurasi data yang sesuai dengan standar hisab internasional.\n"
05233 |         "* Sub-Menu Input: Tanggal Observasi (Y/M/D), Titik Koordinat Pengamat (Lat, Lon, Elevasi, TZ), dan Kondisi Atmosfer.\n"
05234 |         "* Aksi & Output: Menghitung umur bulan sejak ijtimaq dan memisahkan nilai Altitude/Elongasi ke dalam wujud Geosentris dan Toposentris. Mengevaluasi langsung terhadap 5 kriteria sekaligus
(KHGT, MABIMS, Ilyas, Danjon, Yallop)."),
05235 |
05236 |         ("5) Analisis Hilal Global",
05237 |         "* Kegunaan: Pelacakan visibilitas hilal di seluruh zona waktu dunia secara kronologis.\n"
05238 |         "Algoritma ini melakukan iterasi per menit pada garis tanggal internasional untuk mendeteksi di mana dan kapan kriteria hilal pertama kali terpenuhi di muka bumi. Fokus utamanya adalah
memvalidasi prinsip kesatuan matlak global.\n"
05239 |         "* Sub-Menu Input: Tanggal Referensi dan Waktu Acuan UTC.\n"
05240 |         "* Aksi & Output: Mencetak tabel raksasa yang men-sorting 300+ Kota di database dari nilai umur bulan dan ketinggian hilal (Geosentris & Toposentris). Membantu pengguna melihat negara mana
yang lebih dulu berpeluang merukyat."),
05241 |
05242 |         ("6) Status Kriteria Batas",
05243 |         "* Kegunaan: Simulator Anatomi Kriteria Imkanur Rukyat (KHGT).\n"
05244 |         "Instrumen edukasi untuk membedah parameter fisis kriteria KHGT (5-8). Pengguna dapat melihat bagaimana posisi bulan berinteraksi dengan zona kritis seperti Batas Danjon dan Bias Cahaya
Senja.\n"
05245 |         "* Sub-Menu Input: Tanggal, Lokasi, dan dua buah Slider Interaktif (Parameter Elongasi & Ketinggian).\n"
05246 |         "* Aksi & Output: Sistem akan menaruh titik Bintang hasil kalkulasi astrometri aktual pada grafik Kuadran 2D. Jika pengguna menggeser slider secara manual, letak titik Bintang akan berpindah
ke zona-zona pembatas (Zona Merah Danjon, Zona Jingga Bias Senja, atau Zona Hijau KHGT). Status Kelulusan Teks di bawah grafik akan ikut berubah-ubah menjelaskan alasan saintifik kegagalan/kelolosan posisi
Bintang tersebut."),
05247 |
05248 |         ("7) Kota Pertama KHGT (Mainland)",
05249 |         "* Kegunaan: Melacak titik daratan utama (Mainland) pertama di dunia yang masuk bulan baru.\n"
05250 |         "Merupakan fitur krusial dalam Kalender Hijriah Global Tunggal untuk menentukan dimulainya hari pertama bulan Hijriah. Sistem secara otomatis mengabaikan pulau-pulau kecil di Pasifik dan fokus
pada daratan utama benua yang memenuhi syarat 5-8 (Tinggi 5°, Elongasi 8°).\n"
05251 |         "* Sub-Menu Input: Tanggal Konjungsi/Ijtima'.\n"
05252 |         "* Aksi & Output: Laporan teks dan peta topografi global yang menandai lokasi spesifik (bintang emas) lengkap dengan analisis PKG 1 dan PKG 2 (Kriteria Gisborne)."),
05253 |
05254 |         ("8) Crescent Visibility Map",
05255 |         "* Kegunaan: Pemindaian peta visibilitas hilal global dalam resolusi tinggi (HD).\n"
05256 |         "Menggunakan mesin iterasi spasial untuk memetakan visibilitas hilal ke seluruh permukaan bumi. Modul ini mendukung pemetaan berbagai lapisan data astronomi (layers) secara seamless, mulai
dari ketinggian hilal hingga area interseksi kriteria fisis tertentu.\n"
05257 |         "* Sub-Menu Input: Tanggal Observasi, Mode Fase Bulan, Target Pemetaan (Sunset/Moonset/Best Time).\n"
05258 |         "* Aksi & Output: Window grafik interaktif yang menampilkan peta dunia dengan gradien warna visibilitas. Output dapat diekspor ke PNG/SVG resolusi tinggi untuk kebutuhan publikasi ilmiah."),
05259 |
05260 |         ("9) Altitude Chart Analyser",
05261 |         "* Kegunaan: Visualisasi kurva ketinggian harian benda langit.\n"
05262 |         "Modul ini merelasikan pergerakan semu harian Matahari dan Bulan dalam sebuah grafik sinusoida. Berguna untuk melihat titik puncak (transit), waktu terbit-terbenam, serta gap ketinggian antara
Matahari dan Bulan sepanjang 24 jam penuh di lokasi tertentu.\n"
05263 |         "* Sub-Menu Input: Tanggal, Titik Koordinat.\n"
05264 |         "* Aksi & Output: Menggambar grafik garis yang memotong Sumbu Nol (Garis Ufuk). Pengguna dapat melihat secara matematis jam berapa titik temu kedua benda langit terjadi, lengkap dengan penanda
interaktif 'Waktu Sekarang' (Real-time)."),
05265 |
05266 |         ("10) Komparasi 50 Tahun (Ramadhan)",
05267 |         "* Kegunaan: Studi perbandingan penentuan awal bulan antara KHGT dan kriteria lokal (MABIMS).\n"
05268 |         "Melakukan simulasi hisab selama 5 dekade ke depan untuk melihat sinkronisasi atau perbedaan tanggal antara standar global (KHGT) dan standar regional Asia Tenggara (MABIMS) di titik
Sabang.\n"
05269 |         "* Sub-Menu Input: Hanya dieksekusi sekali jalan (1447 H - 1496 H).\n"
05270 |         "* Aksi & Output: Tabel laporan komparasi per tahun dengan status 'Serentak' atau 'Beda (Mundur 1 Hari)'.),
05271 |
05272 |         ("11) Tabel Tinggi Hilal 50 Thn (Ramadhan)",
05273 |         "* Kegunaan: Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Ramadhan selama 50 tahun.\n"
05274 |         "Menyajikan data statistik ketinggian hilal jangka panjang, baik secara geosentrik maupun toposentrik.\n"
05275 |         "* Sub-Menu Input: Eksekusi Otomatis.\n"
05276 |         "* Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel."),
05277 |
05278 |         ("12) Tabel Elongasi Hilal 50 Thn (Ramadhan)",
05279 |         "* Kegunaan: Ekstraksi data mentah Sudut Elongasi untuk 1 Ramadhan selama 50 tahun.\n"
05280 |         "Menyajikan data statistik elongasi hilal jangka panjang, baik secara geosentrik maupun toposentrik.\n"
05281 |         "* Sub-Menu Input: Eksekusi Otomatis.\n"
05282 |         "* Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel."),

```

```

05283 |
05284 | ("13) Komparasi 50 Tahun (Syawal)",
05285 | "• Kegunaan: Analisis potensi beda Hari Raya Idul Fitri umat Islam masa depan.\n"
05286 | "Identik dengan modul 20, namun dikhususkan sepenuhnya untuk melacak tanggal masuknya 1 Syawal selama 50 tahun ke depan menggunakan algoritma KHGT vs MABIMS.\n"
05287 | "• Sub-Menu Input: Eksekusi Otomatis.\n"
05288 | "• Aksi & Output: Laporan matriks perbandingan 50 tahun."),
05289 |
05290 | ("14) Tabel Tinggi Hilal 50 Thn (Syawal)",
05291 | "• Kegunaan: Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Syawal selama 50 tahun.\n"
05292 | "Menyajikan data statistik ketinggian hilal jangka panjang, khusus untuk penentuan Idul Fitri.\n"
05293 | "• Sub-Menu Input: Eksekusi Otomatis.\n"
05294 | "• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel."),
05295 |
05296 | ("15) Tabel Elongasi Hilal 50 Thn (Syawal)",
05297 | "• Kegunaan: Ekstraksi data mentah Sudut Elongasi untuk 1 Syawal selama 50 tahun.\n"
05298 | "Menyajikan data statistik elongasi hilal jangka panjang, khusus untuk penentuan Idul Fitri.\n"
05299 | "• Sub-Menu Input: Eksekusi Otomatis.\n"
05300 | "• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel."),
05301 |
05302 | ("16) Komparasi 50 Tahun (Zulhijah)",
05303 | "• Kegunaan: Analisis potensi beda Hari Raya Idul Adha umat Islam masa depan.\n"
05304 | "Modul ini mensimulasikan data 5 dekade ke depan untuk melihat sinkronisasi atau perbedaan tanggal masuknya 1 Zulhijah (Idul Adha) antara standar KHGT vs MABIMS Sabang.\n"
05305 | "• Sub-Menu Input: Eksekusi Otomatis.\n"
05306 | "• Aksi & Output: Laporan matriks perbandingan 50 tahun untuk bulan Zulhijah."),
05307 |
05308 | ("17) Tabel Tinggi Hilal 50 Thn (Zulhijah)",
05309 | "• Kegunaan: Ekstraksi data mentah Tinggi Hilal (Altitude) untuk 1 Zulhijah selama 50 tahun.\n"
05310 | "Menyajikan data statistik ketinggian hilal jangka panjang, khusus untuk penentuan Idul Adha.\n"
05311 | "• Sub-Menu Input: Eksekusi Otomatis.\n"
05312 | "• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel."),
05313 |
05314 | ("18) Tabel Elongasi Hilal 50 Thn (Zulhijah)",
05315 | "• Kegunaan: Ekstraksi data mentah Sudut Elongasi untuk 1 Zulhijah selama 50 tahun.\n"
05316 | "Menyajikan data statistik elongasi hilal jangka panjang, khusus untuk penentuan Idul Adha.\n"
05317 | "• Sub-Menu Input: Eksekusi Otomatis.\n"
05318 | "• Aksi & Output: Tabel nilai saintifik 50 tahun ke depan yang kompatibel diekspor ke CSV Excel."),
05319 |
05320 | ("19) Komparasi 50 Tahun (12 Bulan)",
05321 | "• Kegunaan: Analisis probabilitas sinkronisasi global KHGT vs MABIMS.\n"
05322 | "Mengekstrak dan membandingkan data awal bulan Hijriah selama 50 tahun penuh (12 bulan x 50 tahun = 600 bulan) untuk menghitung statistik keseragaman secara menyeluruh.\n"
05323 | "• Sub-Menu Input: Eksekusi Otomatis.\n"
05324 | "• Aksi & Output: Tabel komprehensif 600 bulan beserta ringkasan persentase probabilitas keseragaman global."),
05325 |
05326 | ("20) Fase Bulan (Moonphase)",
05327 | "• Kegunaan: Prediktor siklus fase bulan bulanan dan tahunan.\n"
05328 | "Menghitung fraksi iluminasi dan sudut fase untuk menentukan bentuk bulan yang tampak dari bumi, mulai dari New Moon, First Quarter, Full Moon, hingga Last Quarter.\n"
05329 | "• Sub-Menu Input: Tahun Masehi (Mendukung tahun 0 M s/d 3000 M).\n"
05330 | "• Aksi & Output: Mencetak matriks tabel 4 Fase Utama (New Moon/Ijtimak, First Quarter, Full Moon/Purnama, Last Quarter) secara berurutan beserta Jam:Menit-nya."),
05331 |
05332 | ("21) Moon Times",
05333 | "• Kegunaan: Jadwal terbit, transit, dan terbenam Bulan selama satu bulan penuh.\n"
05334 | "Menghitung dinamika harian Bulan dengan mempertimbangkan koreksi refraksi atmosfer dan paralaks. Data ini sangat penting bagi pengamat langit malam dan astronom.\n"
05335 | "• Sub-Menu Input: Bulan, Tahun, Titik Koordinat.\n"
05336 | "• Aksi & Output: Mencetak tabel Waktu Terbit (Moonrise), Titik Puncak/Kulminasi (Transit), dan Waktu Terbenam (Moonset) dengan efek refraksi atmosfer lokal."),
05337 |
05338 | ("22) Sun Times",
05339 | "• Kegunaan: Jadwal harian Matahari (Sunrise, Transit, Sunset).\n"
05340 | "Menyediakan data presisi mengenai waktu fajar, tengah hari, dan terbenamnya matahari. Modul ini menjadi dasar perhitungan waktu-waktu fenomena siang hari.\n"
05341 | "• Sub-Menu Input: Bulan, Tahun, Titik Koordinat.\n"
05342 | "• Aksi & Output: Tabel bulanan waktu matahari terbit hingga terbenam (Sunrise, Transit Zawal, Sunset)."),
05343 |
05344 | ("23) Sun Moon Ephemeris",
05345 | "• Kegunaan: Ekstraksi data koordinat langit (RA/Dec) secara detail.\n"
05346 | "Menyediakan data posisi murni objek (ephemeris) per interval waktu tertentu (detik/menit/jam). Data mencakup Right Ascension, Declination, Azimuth, Altitude, hingga jarak benda langit ke bumi (Distance).\n"
05347 | "• Sub-Menu Input: Target (Sun/Moon), Ref Waktu (UT1/TDT), Lokasi (Geosentris/Toposentris), dan Set Waktu Interval.\n"
05348 | "• Aksi & Output: Tabel raksasa data saintifik murni berisi parameter R.A, Deklinasi, Altitude, Azimuth, Jarak Aktual, Semi-Diameter, dan Delta-T."),
05349 |
05350 | ("24) Sistem Sun Moon Earth",
05351 | "• Kegunaan: Visualisasi 3D sistem tata surya geosentrik sederhana.\n"
05352 | "Menampilkan model 3D interaktif yang menunjukkan posisi Bulan dan Matahari dalam mengelilingi Bumi sebagai pusat pengamatan (Geosentrik). Termasuk proyeksi kerucut bayangan bumi (Umbra).\n"

```



```

05353 |         "• Sub-Menu Input: Set Tanggal Simulasi 3D.\n"
05354 |         "• Aksi & Output: Kanvas 3D interaktif yang menunjukkan orientasi orbit benda langit. Bisa di-Rotasi dan di-Zoom menggunakan mouse."),
05355 |
05356 |     ("25) Prayer Times",
05357 |         "• Kegunaan: Jadwal salat 5 waktu dan fenomena senja (Twilight).\n"
05358 |         "Modul ini menghitung waktu Subuh, Terbit, Duha, Zuhur, Asar, Magrib, dan Isya. Mendukung berbagai standar otoritas internasional dan metode khusus untuk wilayah lintang tinggi.\n"
05359 |         "• Sub-Menu Input: Tanggal, Koordinat Lokasi, Parameter Suhu dan Ketinggian tempat, Mazhab, dan Metode Lembaga.\n"
05360 |         "• Aksi & Output: Menghasilkan Tabel Jadwal harian/bulanan lengkap dengan Midnight. Mendukung ekspor ke Gambar (Jadwal Dinding)."),
05361 |
05362 |     ("26) Qiblah Direction & Times",
05363 |         "• Kegunaan: Kalkulator arah kiblat dan peta navigasi Ka'bah.\n"
05364 |         "Selain menghitung sudut derajat arah kiblat dari arah Utara-Barat menggunakan model Bumi Ellipsoid WGS84, modul ini menyediakan visualisasi garis kiblat pada peta dunia untuk memastikan jalur
05365 |         terpendek (Great Circle) menuju Makkah.\n"
05366 |         "• Sub-Menu Input: Titik Koordinat.\n"
05367 |         "• Aksi & Output: Menampilkan derajat sudut Ka'bah dari Utara Sejati dan proyeksi Garis Lengkung Great Circle di peta topografi Dunia."),
05368 |
05369 |     ("27) Qibla Time (Rashdul Lokal)",
05370 |         "• Kegunaan: Menentukan waktu saat matahari berada tepat di atas arah kiblat suatu lokasi.\n"
05371 |         "Menggunakan algoritma segitiga bola (Spherical Trigonometry) untuk mencari waktu harian di mana azimuth matahari berhimpit dengan azimuth kiblat lokasi tersebut, sehingga bayangan benda tegak
05372 |         dapat digunakan untuk verifikasi arah Ka'bah secara manual.\n"
05373 |         "• Sub-Menu Input: Tanggal Observasi, Titik Koordinat.\n"
05374 |         "• Aksi & Output: Mencetak Waktu Pertama dan Waktu Kedua terjadinya fenomena Rashdul Kiblat lokal."),
05375 |
05376 |     ("28) Kalkulator Mizwala (Bayangan)",
05377 |         "• Kegunaan: Simulator instrumen tongkat istiwa' atau Mizwala.\n"
05378 |         "Menghitung panjang dan arah bayangan benda tegak sepanjang hari. Sangat berguna untuk penentuan arah kiblat manual dan praktikum falak di lapangan.\n"
05379 |         "• Sub-Menu Input: Tinggi Tongkat (cm), dan Waktu Simulasi (Live/Kustom).\n"
05380 |         "• Aksi & Output: Tabel harian pergerakan bayangan dan visualisasi kompas Polar 2D yang berputar selaras dengan matahari."),
05381 |
05382 |     ("29) Evaluasi Fajar Shadiq & SQM",
05383 |         "• Kegunaan: Analisis fajar fisis dan kegelapan langit (Sky Quality).\n"
05384 |         "Modul ini mengevaluasi kemunculan fajar shadiq berdasarkan sudut depresi matahari dan estimasi teoritis nilai SQM (Sky Quality Meter).\n"
05385 |         "• Sub-Menu Input: Tanggal dan Lokasi.\n"
05386 |         "• Aksi & Output: Nilai SQM estimasi dan grafik waktu kemunculan fajar secara fisis-astronomis."),
05387 |
05388 |     ("30) Analisis Gerhana",
05389 |         "• Kegunaan: Pemindaian otomatis fenomena Gerhana Matahari dan Gerhana Bulan.\n"
05390 |         "Menggunakan pustaka mesin pelacak otonom fenomena Syzygy untuk mendeteksi waktu puncak gerhana, jenis gerhana (Total, Cincin, Penumbra), serta visibilitasnya di wilayah Indonesia dan
05391 |         global.\n"
05392 |         "• Sub-Menu Input: Tahun Kalkulasi (1 M hingga 2999 M).\n"
05393 |         "• Aksi & Output: Tabel daftar gerhana setahun. Tersedia tombol untuk mengekspor Jalur Totalitas Gerhana Matahari ke format Google Earth (KML)."),
05394 |
05395 |     ("31) Equinox & Solstice",
05396 |         "• Kegunaan: Menghitung waktu pergantian musim astronomis.\n"
05397 |         "Mendeteksi waktu presisi saat matahari melintasi ekuator langit (Equinox) dan saat mencapai deklinasi maksimum/minimum (Solstice).\n"
05398 |         "• Sub-Menu Input: Tahun.\n"
05399 |         "• Aksi & Output: Tanggal dan jam terjadinya Musim Semi, Panas, Gugur, dan Dingin."),
05400 |
05401 |     ("32) Planetary Times",
05402 |         "• Kegunaan: Jadwal visibilitas 7 planet utama (Merkurius s/d Neptunus).\n"
05403 |         "Menghitung waktu terbit, kulminasi, dan terbenam setiap planet untuk membantu observasi astronomi planetari selama sebulan penuh.\n"
05404 |         "• Sub-Menu Input: Pemilihan Target Planet, Bulan, dan Tahun.\n"
05405 |         "• Aksi & Output: Tabel waktu penampakan planet di langit malam (Rise/Set)."),
05406 |
05407 |     ("33) Generator Analemma Matahari",
05408 |         "• Kegunaan: Pemetaan lintasan analemma Matahari dalam satu tahun.\n"
05409 |         "Memvisualisasikan lintasan 'angka 8' yang dibentuk oleh posisi matahari di jam yang sama sepanjang tahun akibat kemiringan sumbu bumi dan orbit elips.\n"
05410 |         "• Sub-Menu Input: Tahun Analisis, Jam Pengamatan (Lokal), dan Lokasi.\n"
05411 |         "• Aksi & Output: Laporan titik ekstrem analemma (solstice/equinox) dan grafik lintasan analemma tahunan."),
05412 |
05413 |     ("34) Prediktor Pasang Surut Gravitasi",
05414 |         "• Kegunaan: Memprediksi fenomena pasang surut air laut akibat gaya gravitasi Bulan dan Matahari.\n"
05415 |         "Memberikan estimasi waktu pasang tertinggi dan terendah berdasarkan posisi relatif benda langit untuk membantu keamanan observasi di pesisir.\n"
05416 |         "• Sub-Menu Input: Tanggal dan Koordinat Pantai.\n"
05417 |         "• Aksi & Output: Laporan prediksi fluktuasi gravitasi dan estimasi waktu pasang harian."),
05418 |
05419 |     ("35) Astrofotografi & Kontras Visibilitas",
05420 |         "• Kegunaan: Analisis teknis untuk pengambilan foto hilal (imaging).\n"
05421 |         "Modul ini menghitung peluang hilal terekam kamera CCD/CMOS dengan menganalisis jendela kontras (contrast window) antara kecerahan hilal dan latar belakang langit senja.\n"
05422 |         "• Sub-Menu Input: Tanggal dan Koordinat Lokasi.\n"
05423 |         "• Aksi & Output: Rekomendasi waktu terbaik pemotretan dan status kontras visibilitas fisis."),

```

```

05421 |
05422 |         ("36) Live Animasi",
05423 |         "• Kegunaan: Simulator posisi benda langit secara real-time dan terintegrasi observatorium.\n"
05424 |         "Menampilkan visualisasi pergerakan Matahari, Bulan, Rasi Bintang, dan Planet di atas cakrawala pengamat secara dinamis. Kini dilengkapi dengan fitur pelacakan (tracking) benda langit secara
online via internet.\n"
05425 |         "• Integrasi Stellarium: Memiliki akses langsung ke engine Stellarium. Terdapat panel komparasi data astronomi (Matahari dan Bulan) secara real-time antara hasil kalkulasi presisi internal
sistem KHGT (JPL de440s/de441-new otomatis) dengan output aplikasi Stellarium.\n"
05426 |         "• Sub-Menu Input: Waktu Simulasi (Tanggal & Jam), Mode LIVE, Tombol Tracker Online, dan Tombol Komparasi Stellarium.\n"
05427 |         "• Aksi & Output: Melukis garis ufuk, memproyeksikan benda langit secara presisi, fitur 'Zoom Hilal', serta menampilkan jendela tabel komparasi parameter astrometri (Alt, Az, Jarak, RA, Dec,
dll)."),
05428 |
05429 |         ("37) Simulasi Ephemeris 3D",
05430 |         "• Kegunaan: Simulator pergerakan benda langit 3D menggunakan mesin Matplotlib.\n"
05431 |         "Memberikan pengalaman visual yang lebih ilmiah dan presisi. Pengguna dapat melihat posisi aktual bintang, matahari, dan bulan dalam koordinat bola langit dengan grid yang dapat diputar secara
bebas.\n"
05432 |         "• Sub-Menu Input: Slider Kontrol Waktu (Tahun, Bulan, Tgl, Jam) dan Tombol Cari Sunset Otomatis.\n"
05433 |         "• Aksi & Output: Grafik 3D bola langit dengan garis ekuator dan ufuk, melukis titik objek pada koordinat Toposentrik."),
05434 |
05435 |         ("38) WinAI: Aplikasi AI Windows",
05436 |         "• Kegunaan: Asisten cerdas berbasis kecerdasan buatan (AI) untuk konsultasi astronomi dan fikih.\n"
05437 |         "Mengintegrasikan model bahasa besar (Gemini) yang terhubung dengan konteks aplikasi. AI mampu membaca kueri internet dan merujuk dokumen Tarjih secara offline.\n"
05438 |         "• Sub-Menu Input: Kotak Chat untuk mengetik Prompt.\n"
05439 |         "• Aksi & Output: Jawaban teks cerdas dan kemampuan menjalankan fitur aplikasi secara background (Smart Interceptor)."),
05440 |
05441 |         ("39) Peta Langit Dunia",
05442 |         "• Kegunaan: Membuka Peta Langit Dunia KHGT Times berbasis web untuk eksplorasi langit secara online.\n"
05443 |         "Layanan dibuka melalui browser default pada alamat https://kasmui.cloud/langit/. \n"
05444 |         "• Sub-Menu Input: Tidak memerlukan input khusus pada aplikasi desktop.\n"
05445 |         "• Aksi & Output: Membuka Peta Langit Dunia online pada browser default saat menu dipilih."),
05446 |
05447 |         ("40) Laporan Astronomi",
05448 |         "• Kegunaan: Membuka aplikasi pendamping Laporan Astronomi dari KHGT Times.\n"
05449 |         "Modul ini menjalankan file 'LaporanAstronomi.exe' (atau 'LaporanAstronomi.exe') yang berada di folder yang sama dengan aplikasi KHGT Times untuk mengakses laporan astronomi.\n"
05450 |         "• Sub-Menu Input: Tidak memerlukan input khusus pada KHGT Times; aplikasi pendamping dibuka otomatis saat menu dipilih.\n"
05451 |         "• Aksi & Output: Menjalankan aplikasi Laporan Astronomi eksternal. Jika file pendamping tidak ditemukan atau gagal dijalankan, sistem menampilkan pesan kesalahan."),
05452 |
05453 |         ("41) HIJRI_DB Auto-Builder (Admin)",
05454 |         "• Kegunaan: Fitur khusus Developer untuk merakit fondasi Kalender Aplikasi (Siklus Pembuatan Data JSON).\n"
05455 |         "Modul tingkat lanjut untuk membangun file database penanggalan KHGT secara otomatis dari iterasi ribuan tahun mesin Ephemeris NASA.\n"
05456 |         "• Sub-Menu Input: Tahun Awal Hijriah s/d Tahun Akhir Hijriah (Memerlukan Otorisasi Admin).\n"
05457 |         "• Aksi & Output: Menulis seluruh hasilnya menjadi File 'db_hijriah.json' baru yang secara otomatis dimuat ulang ke dalam memori aplikasi.")
05458 |     ]
05459 |
05460 |     # Render list menu dengan dua tag berbeda agar berwarna dan rapi
05461 |     for title, desc in detailed_features:
05462 |         self.textbox.insert("end", f"► {title}\n", "menu_title")
05463 |         self.textbox.insert("end", f"{desc}\n\n", "menu_desc")
05464 |
05465 |     # ===== BAGIAN 4: HIGHLIGHT PEMBARUAN =====
05466 |     self.textbox.insert("end", "\n" + garis_strip + "\n[ 4. HIGHLIGHT PEMBARUAN EKSKLUSIF (VERSI 8.0) ]\n" + garis_strip + "\n", "subtitle")
05467 |
05468 |     updates = [
05469 |         ("[V7.7 - MULTIBAHASA & SIDEBAR] Antarmuka Indonesia, Inggris & Arab", "Sidebar dan elemen antarmuka utama kini mendukung penerjemahan ke Bahasa Indonesia, Inggris, dan Arab sehingga navigasi
modul dapat digunakan secara multibahasa."),
05470 |         ("[V7.7 - ALTITUDE CHART ANALYZER] Analisis Grafik & Kombinasi", "Menu 9 (Altitude Chart Analyser) menyediakan analisis grafik ketinggian dan kombinasi data untuk membantu pembacaan posisi
benda langit secara lebih visual dan terstruktur."),
05471 |         ("[V8.0 - PETA LANGIT DUNIA] Integrasi Peta Langit Online", "Menu 39 adalah Peta Langit Dunia. Saat menu dipilih, KHGT Times membuka layanan peta langit online pada https://kasmui.cloud/langit/
melalui browser default."),
05472 |         ("[V8.0 - LAPORAN ASTRONOMI] Penyesuaian Nomor Menu", "Menu 40 adalah Laporan Astronomi. Menu ini membuka aplikasi pendamping 'LaporanAstronomi.exe' atau 'LaporanAstronomi.exe' dari folder
aplikasi KHGT Times dan menampilkan pesan kesalahan jika file pendamping tidak tersedia atau gagal dijalankan."),
05473 |         ("[V8.0 - MENUS & DOKUMENTASI] Penyelarasan 41 Modul", "Daftar menu aktual V8.0 terdiri atas 41 menu. Menu 39 adalah Peta Langit Dunia, Menu 40 adalah Laporan Astronomi, dan Menu 41 adalah
HIJRI_DB Auto-Builder (Admin). Manual Book, highlight rilis, dan panduan navigasi pada halaman ini telah diselaraskan dengan struktur aktual tersebut."),
05474 |         ("[V8.0 - ENGINE] Auto-Switch Ephemeris Matahari & Bulan", "Sistem otomatis menggunakan de440s.bsp untuk tahun 1850-2149 dan beralih ke de441-new.bsp untuk tahun 1-1849 atau 2150-2999, sehingga
perhitungan modern Matahari-Bulan memakai kernel ringkas berpresisi tinggi tanpa kehilangan jangkauan panjang aplikasi."),
05475 |         ("[V7.6 - LIVE ANIMASI] Sinkronisasi Stellarium", "Pada output Live Animasi ditambahkan tombol Sinkronisasi Stellarium untuk mengaktifkan Stellarium dan membandingkan data Matahari dan Bulan
antara output Stellarium dengan hasil hitung sistem KHGT berbasis ephemeris DE441."),
05476 |         ("[V7.6 - LIVE ANIMASI] Tracker Benda Langit", "Pada output Live Animasi ditambahkan tombol Tracker untuk pelacakan posisi benda langit, baik menggunakan GPS maupun tanpa GPS."),
05477 |         ("[V7.5 - LIVE ANIMASI] Rasi Bintang & Planet", "Pembaruan pada Modul 36 (Live Animasi) dilengkapi visualisasi Rasi Bintang (Constellations) dan planet-planet tata surya secara dinamis dan
real-time."),
05478 |         ("[V7.5 - WINAI] Asisten AI Universal", "Peningkatan kecerdasan WinAI Interceptor agar mampu mendeteksi niat (intent) pengguna dan mengeksekusi pertanyaan atau perintah perhitungan yang
berkaitan dengan menu di dalam sistem."),

```

```

05479 |         ("[V7.5 - ANALEMMA] Grafik Kurva Matahari", "Peningkatan visual pada Modul 33 (Generator Analemma Matahari) yang menampilkan grafik kurva lintasan Matahari berbentuk angka 8."),
05480 |         ("[V7.5 - BUG FIXES] Gerhana, Kiblat & Optimalisasi", "Perbaikan bug pada rentang waktu komputasi di Modul 30 (Analisis Gerhana), perbaikan akurasi pada Modul 26 (Qiblah Direction and Times),
serta berbagai penyesuaian stabilitas sistem."),
05481 |         ("[V7.4 - ENGINE] Penyederhanaan Ephemeris DE441", "Sistem menggunakan file ephemeris 'de441-new.bsp' untuk kalkulasi pada rentang waktu yang didukung aplikasi, sehingga pengelolaan basis
ephemeris menjadi lebih sederhana."),
05482 |         ("[V7.3 - HIGHLIGHTS] Modul Fisis & Auto-Builder", "Modul Astrofotografi, Prediktor Pasang Surut, Evaluasi SQM Fajar, dan HIJRI_DB Auto-Builder untuk Administrator merupakan pengembangan yang
diperkenalkan pada rilis sebelumnya.")
05483 |     ]
05484 |
05485 |     for key, val in updates:
05486 |         key_render = self.translate_release_text_cached(key)
05487 |         val_render = self._translate_release_piece_preserve_urls(val)
05488 |         self.textbox.insert("end", "> ", "highlight")
05489 |         self.textbox.insert("end", f"{key_render}: ", "highlight")
05490 |         self.textbox.insert("end", f"{val_render}\n", "list_item")
05491 |
05492 |     # ===== BAGIAN 5: PANDUAN PENGGUNAAN =====
05493 |     self.textbox.insert("end", garis_strip + "\n[ 5. PANDUAN OPERASIONAL & EXPORT ]\n" + garis_strip + "\n", "subtitle")
05494 |
05495 |     # ---> 1. TAMBAHKAN KONFIGURASI TAG LINK DI SINI <---
05496 |     import webbrowser
05497 |     tb = self.textbox._textbox
05498 |     # HANYA gunakan link_aktif untuk styling (warna dan garis bawah)
05499 |     # JANGAN tambahkan tag_bind <Button-1> di sini agar tidak doble klik
05500 |     tb.tag_config("link_aktif", foreground="#00B0FF", underline=True)
05501 |     # -----
05502 |
05503 |     guides = [
05504 |         ("DOWNLOAD BUKU KHGT TIMES", "Silahkan download aplikasi KHGT Times melalui link: https://s.id/buku-khgttimes"),
05505 |         ("PYTHON MASTERY (TUTORIAL)", "Akses tutorial Python dasar melalui link: https://kasmui.cloud/python/"),
05506 |         ("NAVIGASI MODUL", "Gunakan menu dropdown scrollable 'KHGT ENGINE' di sidebar kiri untuk beralih antar 41 modul perhitungan saintifik. Masing-masing modul akan menyembunyikan form input yang
tidak relevan agar layar selalu rapi dan fokus."),
05507 |         ("INPUT LOKASI", "Tersedia dua opsi. Anda bisa menyeleksi dari Dropdown Provinsi/Kota bawaan aplikasi, atau gunakan tombol '[ ] Deteksi Lokasi Otomatis' untuk memanggil API Geolocation IP agar
sistem melacak titik GPS dan mendeteksi Timezone otomatis."),
05508 |         ("EKSEKUSI DATA", "Pada sebagian besar modul berbasis teks dan grafik, Anda cukup mengisi parameter lalu klik tombol biru besar '► PROSES DATA' di panel bawah. Tombol ini otomatis bersembunyi
jika Anda sedang memasuki modul Live/Real-time."),
05509 |         ("INTERAKSI GRAFIS", "Khusus pada kanvas 3D Matplotlib dan Simulasi 3D Spasial, biasakan diri menggunakan klik Kiri Mouse (tahan dan geser) untuk Merotasi orientasi kamera/sudut pandang, serta
Roda Mouse (Scroll) untuk melakukan perbesaran (Zoom In/Out)."),
05510 |         ("SISTEM EXPORT", "Terdapat panel aksi (Action Bar) melayang di sudut kanan atas layar (atau di bawah tabel/kanvas): \n      [ ] TXT] Menyimpan data tabel saat ini ke file Notepad.\n      [ ] PNG]
Menyimpan kanvas visual / laporan ke gambar PNG.\n      [ ] PDF] Mencetak dokumen laporan standar akademik fpdf.\n      [ ] CSV] Mengekstrak data komparasi 50 Tahun ke Excel.\n      [ ] RILIS] Tombol reset darurat
kembali ke Dokumentasi.")
05511 |     ]
05512 |
05513 |     # ---> 2. PERBAIKAN LOOPING AGAR TIAP LINK BERBEDA & BISA DIKLIK <---
05514 |     for i, (key, val) in enumerate(guides):
05515 |         # Panduan Release adalah isi Textbox, bukan widget UI. Karena itu
05516 |         # terjemahkan key dan val secara eksplisit melalui subsistem cache
05517 |         # release agar EN/AR benar-benar diterapkan tanpa mengubah URL.
05518 |         key_render = self.translate_release_text_cached(key)
05519 |         val_render = self.translate_release_text_cached(val)
05520 |
05521 |         self.textbox.insert("end", "> ", "highlight")
05522 |         self.textbox.insert("end", f"{key_render} : ", "tech")
05523 |
05524 |         # Deteksi jika ada link URL di dalam teks
05525 |         if "https://" in val_render:
05526 |             start_idx = val_render.find("https://")
05527 |             end_idx = val_render.find(" ", start_idx)
05528 |
05529 |             if end_idx == -1:
05530 |                 end_idx = len(val)
05531 |
05532 |             link_url = val_render[start_idx:end_idx]
05533 |             bagian = val_render.split(link_url)
05534 |
05535 |             # Buat nama TAG UNIK untuk setiap link (contoh: url_0, url_1)
05536 |             tag_unik = f"url_{i}"
05537 |
05538 |             # Teks sebelum link
05539 |             self.textbox.insert("end", bagian[0], "list_item")
05540 |

```

```

05541 |         # Teks link yang disisipkan dengan TAG UNIK dan styling "link_aktif"
05542 |         self.textbox.insert("end", link_url, ("list_item", tag_unik, "link_aktif"))
05543 |
05544 |         # Binding event klik kiri (<Button-1>) KHUSUS untuk tag unik ini menuju URL aslinya
05545 |         self.textbox.tag_bind(tag_unik, "<Button-1>", lambda event, url=link_url: webbrowser.open(url))
05546 |
05547 |         # Mengubah kursor jadi tangan saat di-hover KHUSUS untuk tag unik ini
05548 |         tb.tag_bind(tag_unik, "<Enter>", lambda e: tb.configure(cursor="hand2"))
05549 |         tb.tag_bind(tag_unik, "<Leave>", lambda e: tb.configure(cursor="xterm"))
05550 |
05551 |         # Teks setelah link
05552 |         self.textbox.insert("end", bagian[1] + "\n", "list_item")
05553 |     else:
05554 |         self.textbox.insert("end", f"{val_renderer}\n", "list_item")
05555 | # -----
05556 |
05557 | # ===== FOOTER COPYLEFT =====
05558 | footer_text = "\n\n" + garis_sama + "\n\n"
05559 | footer_text += "Copyleft (c) Kasmui, 2026. All Left Reserved.\n"
05560 | footer_text += "This software is fully dedicated to the advancement of global Islamic astronomy and science.\n\n"
05561 | footer_text += "KHGT Times Development Team \nLead Programmer & Developer: Kasmui \nProgramming Tools (In order of role dominance): Gemini, DeepSeek, Qwen, ChatGPT/Copilot, and Grok.\n\n"
05562 | footer_text += garis_sama + "\n"
05563 | self.textbox.insert("end", footer_text, "footer")
05564 |
05565 | # 4. Kunci Textbox dan update judul layar utama
05566 | self.textbox.configure(state="disabled")
05567 | self.lbl_main_title.configure(text="KHGT Times V8.0")
05568 |
05569 | # Widget CTK yang dibuat di dalam halaman rilis (tombol kalender, legenda, dll.)
05570 | # juga diterjemahkan setelah render, tetapi hanya pada wrapper halaman ini.
05571 | try:
05572 |     if getattr(self, "current_lang", "id") != "id" and hasattr(self, "demo_cal_wrapper"):
05573 |         self.update_ui_texts(self.demo_cal_wrapper)
05574 | except Exception:
05575 |     pass
05576 | finally:
05577 |     # Tulis cache JSON setelah halaman rilis selesai digambar.
05578 |     # Pada klik berikutnya, EN/AR cukup dibaca dari file ini.
05579 |     self.save_release_language_cache()
05580 |     self.save_output_language_cache()
05581 |     self._rendering_release_info = False
05582 |
05583 | def create_input_row(self, parent, label_text, default_val):
05584 |     row = ctk.CTkFrame(parent, fg_color="transparent")
05585 |     row.pack(fill="x", padx=10, pady=2)
05586 |     ctk.CTkLabel(row, text=label_text, font=("Segoe UI", 12)).pack(side="left")
05587 |     entry = ctk.CTkEntry(row, width=80, justify="right")
05588 |     entry.delete(0, 'end')
05589 |     entry.insert(0, default_val)
05590 |     entry.pack(side="right")
05591 |     return entry
05592 |
05593 | def create_ymd_row(self, parent, dy, dm, dd):
05594 |     # --- PERBAIKAN: Menggunakan sistem Grid terpusat ---
05595 |     grid_frame = ctk.CTkFrame(parent, fg_color="transparent")
05596 |     grid_frame.pack(fill="x", padx=10, pady=(0, 10))
05597 |
05598 |     # Baris 0: Label Petunjuk
05599 |     ctk.CTkLabel(grid_frame, text="dd", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=0, pady=(0, 2))
05600 |     ctk.CTkLabel(grid_frame, text="mm", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=1, pady=(0, 2))
05601 |     ctk.CTkLabel(grid_frame, text="yyyy", font=("Segoe UI", 10), text_color="#9E9E9E").grid(row=0, column=2, pady=(0, 2))
05602 |
05603 |     # Baris 1: Kotak Input
05604 |     d = ctk.CTkEntry(grid_frame, width=45, placeholder_text="dd", justify="center")
05605 |     d.delete(0, 'end')
05606 |     d.insert(0, dd)
05607 |     d.grid(row=1, column=0, padx=(0, 5))
05608 |
05609 |     m = ctk.CTkEntry(grid_frame, width=45, placeholder_text="mm", justify="center")
05610 |     m.delete(0, 'end')
05611 |     m.insert(0, dm)

```

```

05612 |         m.grid(row=1, column=1, padx=5)
05613 |
05614 |         y = ctk.CTkEntry(grid_frame, width=70, placeholder_text="yyyy", justify="center")
05615 |         y.delete(0, 'end')
05616 |         y.insert(0, dy)
05617 |         y.grid(row=1, column=2, padx=(5, 0))
05618 |
05619 |         return y, m, d
05620 |
05621 |     def create_hms_row(self, parent, dh, dm, ds):
05622 |         row = ctk.CTkFrame(parent, fg_color="transparent")
05623 |         row.pack(fill="x", padx=10, pady=2)
05624 |         h = ctk.CTkEntry(row, width=35, placeholder_text="h")
05625 |         h.delete(0, 'end')
05626 |         h.insert(0, dh)
05627 |         h.pack(side="left", padx=2)
05628 |         m = ctk.CTkEntry(row, width=35, placeholder_text="m")
05629 |         m.delete(0, 'end')
05630 |         m.insert(0, dm)
05631 |         m.pack(side="left", padx=2)
05632 |         s = ctk.CTkEntry(row, width=35, placeholder_text="s")
05633 |         s.delete(0, 'end')
05634 |         s.insert(0, ds)
05635 |         s.pack(side="left", padx=2)
05636 |         return h, m, s
05637 |
05638 |     def auto_switch_ephemeris(self, target_year):
05639 |         """
05640 |         Pilih ephemeris otomatis berdasarkan tahun target.
05641 |
05642 |         Rentang aplikasi tetap 1-2999 M:
05643 |         - 1850-2149 -> de440s.bsp (kernel pendek, fokus era modern)
05644 |         - lainnya   -> de441-new.bsp (kernel panjang)
05645 |         """
05646 |         try:
05647 |             target_year = int(target_year)
05648 |         except (TypeError, ValueError):
05649 |             raise ValueError(f"Tahun tidak valid: {target_year}")
05650 |
05651 |         if target_year < EPHEMERIS_LONG_MIN_YEAR or target_year > EPHEMERIS_LONG_MAX_YEAR:
05652 |             err_msg = f"Tahun {target_year} di luar jangkauan!"
05653 |             detail_msg = (
05654 |                 f"Sistem mendukung rentang tahun {EPHEMERIS_LONG_MIN_YEAR} s/d "
05655 |                 f"{EPHEMERIS_LONG_MAX_YEAR} Masehi.\n\n"
05656 |                 f"Input Anda: {target_year}\n\n"
05657 |                 f"Pemilihan ephemeris otomatis:\n"
05658 |                 f"* {EPHEMERIS_SHORT_FILE}: {EPHEMERIS_SHORT_MIN_YEAR}-{EPHEMERIS_SHORT_MAX_YEAR} M\n"
05659 |                 f"* {EPHEMERIS_LONG_FILE}: di luar rentang tersebut."
05660 |             )
05661 |             self.after(0, lambda: messagebox.showwarning(
05662 |                 "Rentang Tahun Tidak Didukung", detail_msg
05663 |             ))
05664 |             self.after(0, lambda: self.lbl_status.configure(
05665 |                 text="Tahun Tidak Valid!", text_color="#FF1744"
05666 |             ))
05667 |             self.after(0, lambda: self.btn_hitung.configure(state="normal"))
05668 |             raise ValueError(err_msg)
05669 |
05670 |         desired = self._ephemeris_filename_for_year(target_year)
05671 |
05672 |         if self.eph is None or self.ephemeris_name != desired:
05673 |             # Pesan status dibuat melalui event loop agar aman bila dipanggil thread kalkulasi.
05674 |             self.after(0, lambda f=desired, y=target_year: self.lbl_status.configure(
05675 |                 text=f"Mengalihkan ephemeris ke {f} ({y})...",
05676 |                 text_color="#00E5FF"
05677 |             ))
05678 |             try:
05679 |                 self._load_ephemeris_file(desired)
05680 |             except Exception as e:
05681 |                 self.after(0, lambda err=str(e): messagebox.showerror(
05682 |                     "Gagal Mengganti Ephemeris", err

```

```

05683 |         ))
05684 |         self.after(0, lambda: self.lbl_status.configure(
05685 |             text="Gagal memuat Ephemeris!", text_color="#FF1744"
05686 |         ))
05687 |         raise
05688 |
05689 |     self.after(0, lambda f=desired: self.lbl_status.configure(
05690 |         text=f"Sistem Siap ({f})", text_color="#00E676"
05691 |     ))
05692 |
05693 |     return self.eph
05694 |
05695 | def run_calculation(self):
05696 |     self.lbl_status.configure(text="Menghitung...", text_color="#FFAB40")
05697 |     self.btn_hitung.configure(state="disabled")
05698 |
05699 |     self.textbox.configure(font=("Consolas", 13), wrap="none")
05700 |
05701 |     mode = self.combo_mode.get()
05702 |
05703 |     if "Altitude Chart" in mode:
05704 |         threading.Thread(target=self.calculate_chart_analyser, daemon=True).start()
05705 |         return
05706 |
05707 |     if "Visibility Map" not in mode and "Analisis Gerhana" not in mode and "Live Animasi" not in mode and "Sistem Sun Moon" not in mode and "Simulasi Ephemeris" not in mode:
05708 |         self.textbox.configure(state="normal")
05709 |         self.textbox.delete("1.0", "end")
05710 |         self.textbox.insert("1.0", "Memproses data astrometri...\n")
05711 |         self.textbox.configure(state="disabled")
05712 |
05713 |     if "Visibility Hilal" in mode:
05714 |         threading.Thread(target=self.calculate_visibility, daemon=True).start()
05715 |     elif "Visibility Map" in mode:
05716 |         threading.Thread(target=self.calculate_visibility_map, daemon=True).start()
05717 |     elif "Analisis Hilal Global" in mode:
05718 |         threading.Thread(target=self.calculate_global_hilal, daemon=True).start()
05719 |     elif "Moonphase" in mode:
05720 |         threading.Thread(target=self.calculate_moonphase, daemon=True).start()
05721 |     elif "Sun Moon Ephemeris" in mode:
05722 |         threading.Thread(target=self.calculate_ephemeris, daemon=True).start()
05723 |     elif "Qiblah Direction" in mode:
05724 |         threading.Thread(target=self.calculate_qiblah, daemon=True).start()
05725 |     elif "Moon Times" in mode:
05726 |         threading.Thread(target=self.calculate_moontimes, daemon=True).start()
05727 |     elif "Sun Times" in mode:
05728 |         threading.Thread(target=self.calculate_suntimes, daemon=True).start()
05729 |     elif "Prayer Times" in mode:
05730 |         threading.Thread(target=self.calculate_prayertimes, daemon=True).start()
05731 |     elif "Konversi" in mode:
05732 |         threading.Thread(target=self.calculate_conversion, daemon=True).start()
05733 |     elif "Qibla Time" in mode:
05734 |         threading.Thread(target=self.calculate_qiblatime, daemon=True).start()
05735 |     elif "Analisis Gerhana" in mode:
05736 |         tahun = int(self.combo_tahun_gerhana.get())
05737 |         threading.Thread(target=self._calc_gerhana_thread, args=(tahun,), daemon=True).start()
05738 |     elif "Kota Pertama" in mode:
05739 |         threading.Thread(target=self.calculate_first_point, daemon=True).start()
05740 |     elif "Equinox" in mode:
05741 |         threading.Thread(target=self.calculate_seasons, daemon=True).start()
05742 |     elif "Planetary Times" in mode:
05743 |         threading.Thread(target=self.calculate_planetary_times, daemon=True).start()
05744 |     # ---> URUTAN YANG BENAR DI TOMBOL PROSES (SYAWAL HARUS DI ATAS) <---
05745 |     elif "Komparasi 50 Tahun (Syawal)" in mode:
05746 |         self.analisis_komparasi_syawal_50_tahun()
05747 |
05748 |     # ---> TAMBAHKAN ROUTING EKSEKUSI ZULHIJAH DI SINI <---
05749 |     elif "Komparasi 50 Tahun (Zulhijah)" in mode:
05750 |         self.analisis_komparasi_zulhijah_50_tahun()
05751 |
05752 |     elif "Komparasi 50 Tahun (12 Bulan)" in mode:
05753 |         self.analisis_komparasi_global_50_tahun()

```

```

05754 |
05755 |     elif "Komparasi 50 Tahun" in mode:
05756 |         self.analisis_komparasi_ramadhan_50_tahun()
05757 |
05758 |     elif "Tabel Tinggi Hilal 50 Thn (Syawal)" in mode:
05759 |         threading.Thread(target=self._proses_tabel_ketinggian_syawal_50_tahun, daemon=True).start()
05760 |
05761 |     # ---> TAMBAHKAN ROUTING EKSEKUSI TABEL TINGGI ZULHIJAH DI SINI <---
05762 |     elif "Tabel Tinggi Hilal 50 Thn (Zulhijah)" in mode:
05763 |         threading.Thread(target=self._proses_tabel_ketinggian_zulhijah_50_tahun, daemon=True).start()
05764 |
05765 |
05766 |     elif "Tabel Tinggi Hilal" in mode:
05767 |         threading.Thread(target=self._proses_tabel_ketinggian_50_tahun, daemon=True).start()
05768 |
05769 |     # ---> SISIPKAN ROUTING MENU 25 DI SINI <---
05770 |     elif "Tabel Elongasi Hilal 50 Thn (Syawal)" in mode:
05771 |         threading.Thread(target=self._proses_tabel_elongasi_syawal_50_tahun, daemon=True).start()
05772 |
05773 |     # ---> TAMBAHKAN ROUTING EKSEKUSI TABEL ELONGASI ZULHIJAH DI SINI <---
05774 |     elif "Tabel Elongasi Hilal 50 Thn (Zulhijah)" in mode:
05775 |         threading.Thread(target=self._proses_tabel_elongasi_zulhijah_50_tahun, daemon=True).start()
05776 |
05777 |     # Pastikan ini berada di bawah Menu 25
05778 |     elif "Tabel Elongasi Hilal" in mode:
05779 |         threading.Thread(target=self._proses_tabel_elongasi_50_tahun, daemon=True).start()
05780 |
05781 |     elif "Mizwala" in mode:
05782 |         threading.Thread(target=self.calculate_mizwala, daemon=True).start()
05783 |
05784 |     elif "Auto-Builder" in mode:
05785 |         threading.Thread(target=self._build_hijri_db_thread, daemon=True).start() # <--- TAMBAHKAN INI
05786 |
05787 |     # ---> TAMBAHKAN ROUTING MENU 31 DI SINI <---
05788 |     elif "Status Kriteria Batas" in mode:
05789 |         threading.Thread(target=self.calculate_kriteria_batas, daemon=True).start()
05790 |
05791 |     # ---> TAMBAHAN PEMICU MODUL 31 <---
05792 |     elif "Astrofotografi" in mode:
05793 |         threading.Thread(target=self.calculate_astrofotografi, daemon=True).start()
05794 |
05795 |     elif "Pasang Surut" in mode:
05796 |         threading.Thread(target=self.calculate_pasang_surut, daemon=True).start()
05797 |
05798 |     # ---> TAMBAHAN PEMICU MODUL 33 <---
05799 |     elif "Fajar Shadiq" in mode:
05800 |         threading.Thread(target=self.calculate_fajar_sqm, daemon=True).start()
05801 |
05802 |     elif "Analemma" in mode:
05803 |         threading.Thread(target=self.calculate_analemma, daemon=True).start()
05804 |
05805 | # MODUL TAMBAHAN 5: ANALISIS HILAL GLOBAL ITERATIF (EPHEM)
05806 | # =====
05807 | def calculate_global_hilal(self):
05808 |     # ---> PERBAIKAN UI: Mengembalikan tampilan ke mode Tabel (Teks) <---
05809 |     def _siapkan_ui_tabel():
05810 |         if hasattr(self, 'frame_alaska_out'):
05811 |             self.frame_alaska_out.grid_remove() # Sembunyikan kurva alaska
05812 |             self.textbox.grid(row=1, column=0, sticky="nsew") # Munculkan tabel
05813 |             self.textbox.configure(state="normal")
05814 |             self.textbox.delete("1.0", "end")
05815 |             self.textbox.insert("1.0", "Menganalisis data hilal global lintas benua dan samudra (PKG 1 & PKG 2)...\nMohon tunggu...\n")
05816 |             self.textbox.configure(state="disabled")
05817 |
05818 |     self.after(0, _siapkan_ui_tabel)
05819 |
05820 |     try:
05821 |         import math
05822 |         import pytz
05823 |         import datetime
05824 |         import numpy as np

```

```

05825 |         from skyfield import almanac
05826 |
05827 |         y = int(self.entry_gha_year.get())
05828 |         m = int(self.entry_gha_month.get())
05829 |         d = int(self.entry_gha_day.get())
05830 |         time_str = self.entry_gha_time.get()
05831 |         if not time_str: time_str = "12:00:00"
05832 |
05833 |         h, mn, s = map(int, time_str.split(':'))
05834 |         tanggal_referensi_utc = f"{y}/{m:02d}/{d:02d} {time_str}"
05835 |
05836 |         self.auto_switch_ephemeris(y)
05837 |         earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
05838 |
05839 |         # Waktu referensi UTC sebagai patokan mencari ijtimak
05840 |         t_ref = self.ts.utc(y, m, d, h, mn, s)
05841 |         t_ref_tt = t_ref.tt.item() if hasattr(t_ref.tt, 'item') else t_ref.tt
05842 |
05843 |         # --- 1. SINKRONISASI PENCARIAN IJTIMAK ---
05844 |         t_start_conj = self.ts.tt_jd(t_ref_tt - 20.0)
05845 |         t_end_conj = self.ts.tt_jd(t_ref_tt + 20.0)
05846 |
05847 |         t_phases, y_phases = almanac.find_discrete(t_start_conj, t_end_conj, almanac.moon_phases(self.eph))
05848 |         t_ijtimak = None
05849 |         jarak_terdekat = float('inf')
05850 |
05851 |
05852 |
05853 |         for t_ev, phase in get_safe_events(t_phases, y_phases):
05854 |             if phase == 0: # 0 = Konjungsi / New Moon
05855 |                 ev_tt = t_ev.tt.item() if hasattr(t_ev.tt, 'item') else t_ev.tt
05856 |                 selisih_hari = abs(ev_tt - t_ref_tt)
05857 |                 if selisih_hari < jarak_terdekat:
05858 |                     jarak_terdekat = selisih_hari
05859 |                     t_ijtimak = t_ev
05860 |
05861 |         if t_ijtimak is None: raise ValueError("Ijtimak tidak ditemukan dalam rentang waktu ini.")
05862 |
05863 |         str_ijtimak = t_ijtimak.utc_datetime().strftime("%Y/%m/%d %H:%M:%S")
05864 |         ti_tt = t_ijtimak.tt.item() if hasattr(t_ijtimak.tt, 'item') else t_ijtimak.tt
05865 |
05866 |         hasil_kalkulasi = []
05867 |
05868 |         # LOOP DATABASE KOTA (HANYA UNTUK DITAMPILKAN DI TABEL)
05869 |         for negara, kota_dict in CITY_DB.items():
05870 |             for nama_kota, koordinat in kota_dict.items():
05871 |                 lintang, bujur = koordinat
05872 |                 loc = wgs84.latlon(lintang, bujur, elevation_m=0)
05873 |                 tz_approx = int(self.get_tz_from_lon(bujur))
05874 |
05875 |                 t0_ss = self.ts.utc(y, m, d, 0 - tz_approx)
05876 |                 t1_ss = self.ts.utc(y, m, d, 24 - tz_approx)
05877 |
05878 |                 t_ss_evs, y_ss_evs = almanac.find_discrete(t0_ss, t1_ss, almanac.sunrise_sunset(self.eph, loc))
05879 |
05880 |                 t_sunset = None
05881 |                 for t_ev, is_sunrise in get_safe_events(t_ss_evs, y_ss_evs):
05882 |                     if not is_sunrise:
05883 |                         t_sunset = t_ev
05884 |                         break
05885 |
05886 |                 if t_sunset is None:
05887 |                     hasil_kalkulasi.append({
05888 |                         'negara': negara, 'kota': nama_kota, 'waktu_sort': 999999.0,
05889 |                         'baris_teks': f"{negara[:15]:<15} | {nama_kota[:14]:<14} | {'Anomali Ekstrem':<16} | {'-':<9} | {'-':<8} | {'-':<8} | {'-':<8} | {'-':<8} | Lintang Tinggi (Midnight Sun)",
05890 |                         'is_khgt': False, 't_sunset_obj': None, 'lon': bujur, 'lat': lintang
05891 |                     })
05892 |                 continue
05893 |
05894 |                 ts_tt = t_sunset.tt.item() if hasattr(t_sunset.tt, 'item') else t_sunset.tt
05895 |

```



```

05896 |         geo_earth = earth.at(t_sunset)
05897 |         m_app_geo = geo_earth.observe(moon).apparent()
05898 |         s_app_geo = geo_earth.observe(sun).apparent()
05899 |
05900 |         sep_geo = s_app_geo.separation_from(m_app_geo).degrees
05901 |         elong_geo = sep_geo.item() if hasattr(sep_geo, 'item') else sep_geo
05902 |
05903 |         ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset)
05904 |         gast = t_sunset.gast
05905 |         lst_deg = (gast * 15.0) + bujur
05906 |
05907 |         ra_h = ra_m.hours.item() if hasattr(ra_m.hours, 'item') else ra_m.hours
05908 |         dec_r = dec_m.radians.item() if hasattr(dec_m.radians, 'item') else dec_m.radians
05909 |
05910 |         ha_deg = lst_deg - (ra_h * 15.0)
05911 |         ha_rad = math.radians(ha_deg)
05912 |         lat_rad = math.radians(lintang)
05913 |
05914 |         sin_alt_geo = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) * math.cos(ha_rad)
05915 |         sin_alt_geo = max(-1.0, min(1.0, sin_alt_geo))
05916 |         alt_geo = math.degrees(math.asin(sin_alt_geo))
05917 |
05918 |         topo_obs = (earth + loc).at(t_sunset)
05919 |         app_moon_topo = topo_obs.observe(moon).apparent()
05920 |         app_sun_topo = topo_obs.observe(sun).apparent()
05921 |
05922 |         alt_moon_topo_obj, _, _ = app_moon_topo.altaz(temperature_C=25, pressure_mbar=1010)
05923 |         alt_moon_topo = alt_moon_topo_obj.degrees
05924 |
05925 |         sep_topo = app_sun_topo.separation_from(app_moon_topo).degrees
05926 |         elong_topo = sep_topo.item() if hasattr(sep_topo, 'item') else sep_topo
05927 |
05928 |         umur_bulan_desimal = (ts_tt - ti_tt) * 24.0
05929 |
05930 |         if umur_bulan_desimal < 0:
05931 |             format_umur = "B. Ijtimak"
05932 |             status_visibilitas = "Negatif (Bulan Tua)"
05933 |         else:
05934 |             jam = int(umur_bulan_desimal)
05935 |             menit = int((umur_bulan_desimal - jam) * 60)
05936 |             format_umur = f"{jam}j {menit}m"
05937 |
05938 |             if alt_geo >= 5.0 and elong_geo >= 8.0: status_visibilitas = "Memenuhi Kriteria KHGT"
05939 |             elif alt_moon_topo >= 3.0 and elong_geo >= 6.4: status_visibilitas = "Memenuhi Kriteria MABIMS"
05940 |             elif alt_geo > 0 or alt_moon_topo > 0: status_visibilitas = "Wujudul Hilal"
05941 |             else: status_visibilitas = "Negatif (Bawah Ufuk)"
05942 |
05943 |         dt_sunset_lokal = t_sunset.utcdatetime() + datetime.timedelta(hours=tz_approx)
05944 |         sunset_str = dt_sunset_lokal.strftime("%d/%m/%Y %H:%M")
05945 |
05946 |         alt_g_str, elong_g_str = f"{alt_geo:.2f}°", f"{elong_geo:.2f}°"
05947 |         alt_t_str, elong_t_str = f"{alt_moon_topo:.2f}°", f"{elong_topo:.2f}°"
05948 |
05949 |         baris_teks = f"{negara[:15]:<15} | {nama_kota[:14]:<14} | {sunset_str[:16]:<16} | {format_umur[:9]:<9} | {alt_g_str[:8]:<8} | {elong_g_str[:8]:<8} | {alt_t_str[:8]:<8} | {elong_t_str[:8]:<8} | {status_visibilitas}"
05950 |
05951 |         is_khgt_valid = umur_bulan_desimal > 0 and alt_geo >= 5.0 and elong_geo >= 8.0
05952 |
05953 |         hasil_kalkulasi.append({
05954 |             'negara': negara, 'kota': nama_kota, 'waktu_sort': ts_tt, 'baris_teks': baris_teks,
05955 |             'is_khgt': is_khgt_valid, 't_sunset_obj': t_sunset, 'lon': bujur, 'lat': lintang,
05956 |             'alt_geo': alt_geo, 'elong_geo': elong_geo
05957 |         })
05958 |
05959 |     hasil_kalkulasi.sort(key=lambda x: (x['negara'].lower(), x['kota'].lower()))
05960 |
05961 |     # =====
05962 |     # EVALUASI PKG 1 & PKG 2 (MENYELURUH DARATAN DAN LAUTAN)
05963 |     # =====
05964 |     import ephemeris
05965 |     obs_date_utc = datetime.date(y, m, d)

```

```

05966 |         batas_00_utc_ephem = ephem.Date(datetime.datetime(obs_date_utc.year, obs_date_utc.month, obs_date_utc.day) + datetime.timedelta(days=1))
05967 |
05968 |         # Hitung Fajar Gisborne untuk PKG 2
05969 |         gisborne = ephem.Observer()
05970 |         gisborne.lat, gisborne.lon = math.radians(-38.6623), math.radians(178.0176)
05971 |         gisborne.elevation, gisborne.pressure = 0, 0
05972 |         gisborne.horizon = '-18'
05973 |         matahari_ephem = ephem.Sun()
05974 |         gisborne.date = ephem.Date(batas_00_utc_ephem - 0.5)
05975 |         try:
05976 |             fajar_gisborne = gisborne.next_rising(matahari_ephem, use_center=True)
05977 |         except:
05978 |             fajar_gisborne = ephem.Date(batas_00_utc_ephem + 0.6)
05979 |
05980 |         dt_utc_ijtima = t_ijtimak.utc_datetime()
05981 |         is_ijtima_before_fajr = ephem.Date(dt_utc_ijtima) < fajar_gisborne
05982 |
05983 |         pkg1_met = False
05984 |         pkg2_met = False
05985 |         pkg_reason = ""
05986 |         titik_pertama_global = None
05987 |
05988 |         # 1. EVALUASI PKG 1 (Dimanapun di dunia, Daratan / Lautan asalkan Sunset < 00:00 UTC)
05989 |         # Kita lakukan scanning grid 10 derajat mencakup daratan dan lautan
05990 |         for lon_g in range(180, -181, -10):
05991 |             if pkg1_met: break
05992 |             for lat_g in range(-60, 61, 15):
05993 |                 loc_g = wgs84.latlon(lat_g, lon_g)
05994 |                 est_sunset_utc = 12.0 - (lon_g / 15.0)
05995 |                 t0_g = self.ts.utc(y, m, d, est_sunset_utc - 2)
05996 |                 t1_g = self.ts.utc(y, m, d, est_sunset_utc + 2)
05997 |
05998 |                 try:
05999 |                     t_evs, y_evs = almanac.find_discrete(t0_g, t1_g, almanac.sunrise_sunset(self.eph, loc_g))
06000 |                     for t_ev, is_sr in zip(np.atleast_1d(t_evs.tt), np.atleast_1d(y_evs)):
06001 |                         if not is_sr:
06002 |                             t_ss_sky = self.ts.tt_id(t_ev)
06003 |                             if ephem.Date(t_ss_sky.utc_datetime()) < batas_00_utc_ephem:
06004 |                                 obs_center = earth.at(t_ss_sky)
06005 |                                 m_app = obs_center.observe(moon).apparent()
06006 |                                 s_app = obs_center.observe(sun).apparent()
06007 |                                 elong_g = s_app.separation_from(m_app).degrees
06008 |
06009 |                                 if elong_g >= 8.0:
06010 |                                     ra_m, dec_m, _ = m_app.radec(epoch=t_ss_sky)
06011 |                                     ha_deg = (t_ss_sky.gast * 15.0) + lon_g - (ra_m.hours * 15.0)
06012 |                                     sin_alt = math.sin(dec_m.radians) * math.sin(math.radians(lat_g)) + math.cos(dec_m.radians) * math.cos(math.radians(lat_g)) * math.cos(math.radians(ha_deg))
06013 |                                     alt_g = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
06014 |
06015 |                                     if alt_g >= 5.0:
06016 |                                         pkg1_met = True
06017 |                                         pkg_reason = f"PKG 1 (Titik Koordinat: Lintang {lat_g}°, Bujur {lon_g}° [Darat/Lautan])"
06018 |                                         titik_pertama_global = f"Lautan/Daratan (Lat: {lat_g}°, Lon: {lon_g}°) | Sunset UTC: {t_ss_sky.utc_strftime('%H:%M:%S')}"
06019 |                                         break
06020 |             except Exception:
06021 |                 pass
06022 |
06023 |         # 2. EVALUASI PKG 2 (Hanya Daratan Amerika, Sunset >= 00:00 UTC, Ijtimak < Fajar NZ)
06024 |         if not pkg1_met:
06025 |             negara_mainland_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Amerika Selatan"]
06026 |             # Memfilter kota kepulauan (Hawaii, Polinesia, dll)
06027 |             kota_kepulauan = ["Honolulu", "Papeete", "Alofi", "Pago Pago", "Fernando de Noronha", "Nuuk", "San Juan", "Santo Domingo", "Havana", "Praia"]
06028 |
06029 |             for hasil in hasil_kalkulasi:
06030 |                 if pkg2_met: break
06031 |                 if hasil['t_sunset_obj'] is not None:
06032 |                     is_mainland_amerika = hasil['negara'] in negara_mainland_amerika and hasil['kota'] not in kota_kepulauan
06033 |                     if is_mainland_amerika:
06034 |                         if ephem.Date(hasil['t_sunset_obj'].utc_datetime()) >= batas_00_utc_ephem:
06035 |                             if hasil['alt_geo'] >= 5.0 and hasil['elong_geo'] >= 8.0:
06036 |                                 if is_ijtima_before_fajr:

```

```

06037 |         pkg2_met = True
06038 |         pkg_reason = f"PKG 2 (Daratan Benua Amerika: {hasil['kota']}, {hasil['negara']}) & Ijtima < Fajar NZ)"
06039 |         titik_pertama_global = f"{hasil['kota']}, {hasil['negara']} | Sunset UTC: {hasil['t_sunset_obj'].utc.strftime('%H:%M:%S')}"
06040 |     else:
06041 |         pkg_reason = f"PKG 2 GAGAL (Kriteria tembus di Amerika: {hasil['kota']}, TAPI Ijtima > Fajar NZ)"
06042 |
06043 |     # Penetapan Nama Bulan dan Tahun Hijriah untuk Output
06044 |     nm_tt_val = self.ts.from_datetime(dt_utc_ijtima.replace(tzinfo=pytz.utc)).tt
06045 |     h_y, h_m = self.get_hijri_month_from_tt(nm_tt_val)
06046 |     hijri_bulan_str = BULAN_HIJRIAH[h_m - 1]
06047 |     hijri_tahun_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
06048 |
06049 |     tgl_1_mas = ephemer.Date(batas_00_utc_ephem)
06050 |     gy, gm, gd, _, _, _ = tgl_1_mas.tuple()
06051 |     masehi_str_besok = f"{int(gd):02d} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
06052 |
06053 |     tgl_lusa = ephemer.Date(batas_00_utc_ephem + 1.0)
06054 |     gy2, gm2, gd2, _, _, _ = tgl_lusa.tuple()
06055 |     masehi_str_lusa = f"{int(gd2):02d} {BULAN_MASEHI[int(gm2)-1]} {format_tahun_aman(int(gy2))}"
06056 |
06057 |     if pkg1_met:
06058 |         status_khgt = f"PKG 1 Terpenuhi (Global Daratan/Lautan, Sunset < 00:00 UTC)\n* Awal Bulan Baru : BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str_besok})"
06059 |     elif pkg2_met:
06060 |         status_khgt = f"PKG 2 Terpenuhi (Daratan Benua Amerika & Ijtima < Fajar NZ)\n* Awal Bulan Baru : BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str_besok})"
06061 |     else:
06062 |         status_khgt = f"Gagal PKG 1 & PKG 2 ({pkg_reason if pkg_reason else 'Kriteria tidak terpenuhi'})\n* Awal Bulan Baru : LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str_lusa})"
06063 |
06064 |     if titik_pertama_global is None:
06065 |         info_kota_pertama = "□ TITIK PERTAMA KHGT : Tidak Ditemukan (Kriteria 5-8 belum terpenuhi di seluruh dunia)"
06066 |     else:
06067 |         info_kota_pertama = f"□ TITIK PERTAMA KHGT : {titik_pertama_global}\n* Status Global : {status_khgt}"
06068 |
06069 |     # ---> PENYUSUNAN HEADER DAN OUTPUT AKHIR <---
06070 |     output_lines = []
06071 |     header = f"{self.get_header(138)}\n"
06072 |     header += f"{'[ Analisis Hilal Global (KHGT, MABIMS, Wujudul Hilal) - Skyfield ]'.center(138)}\n\n"
06073 |     header += f"* Tanggal Referensi Pencarian (UTC): {tanggal_referensi_utc}\n"
06074 |     header += f"* Waktu Ijtimak/Konjungsi Terdekat (UTC): {str_ijtimak}\n"
06075 |     header += f"* {info_kota_pertama}\n"
06076 |     header += f"* Catatan: Tabel di bawah diurutkan Berdasarkan Abjad Negara, lalu Abjad Kota.\n"
06077 |     header += f"{"*"138 + "\n"
06078 |     header += f"{'Negara':<15} | {'Kota':<14} | {'Sunset (LT)':<16} | {'Umur Bln':<9} | {'Alt(Geo)':<8} | {'Eln(Geo)':<8} | {'Alt(Top)':<8} | {'Eln(Top)':<8} | Status\n"
06079 |     header += f"{"*"138
06080 |
06081 |     output_lines.append(header)
06082 |
06083 |     for hasil in hasil_kalkulasi:
06084 |         output_lines.append(hasil['baris_teks'])
06085 |
06086 |     output_lines.append(f"{"*"138)
06087 |     self.after(0, self.display_result, "\n".join(output_lines))
06088 |
06089 | except Exception as e:
06090 |     import traceback
06091 |     self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
06092 |
06093 | # =====
06094 | # MODUL TAMBAHAN 11: ANALISIS GERHANA ALGORITMIK
06095 | # =====
06096 | def setup_gerhana_out_frame(self):
06097 |     self.frame_gerhana_out = ctk.CTkFrame(self.main_frame, fg_color="transparent")
06098 |
06099 |     style = ttk.Style(self)
06100 |     style.theme_use("clam")
06101 |
06102 |     # 1. Mengubah font Judul Kolom (Heading) menjadi ukuran 13
06103 |     style.configure("Gerhana.Treeview.Heading", font=('Segoe UI', 13, 'bold'), background="#2b5797", foreground="white")
06104 |
06105 |     # 2. Mengubah font Isi Tabel menjadi ukuran 13 dan tinggi baris (rowheight) menjadi 32
06106 |     style.configure("Gerhana.Treeview", font=('Segoe UI', 13), rowheight=32, background="#1e1e1e", foreground="white", fieldbackground="#1e1e1e")
06107 |

```

```

06108 |         style.map('Gerhana.Treeview', background=[('selected', '#0078D7')], foreground=[('selected', 'white')])
06109 |
06110 |         frame_tabel = ctk.CTkFrame(self.frame_gerhana_out, fg_color="transparent")
06111 |         frame_tabel.pack(fill="both", expand=True, pady=(0, 10))
06112 |
06113 |         kolom_gerhana = ("objek", "jenis", "mulai", "puncak", "akhir", "wilayah_global", "indo_vis")
06114 |         self.tabel_gerhana = ttk.Treeview(frame_tabel, columns=kolom_gerhana, show="headings", style="Gerhana.Treeview")
06115 |
06116 |         self.tabel_gerhana.heading("objek", text="Objek")
06117 |         self.tabel_gerhana.heading("jenis", text="Jenis Gerhana")
06118 |         self.tabel_gerhana.heading("mulai", text="Kontak Awal (WIB)")
06119 |         self.tabel_gerhana.heading("puncak", text="Puncak (WIB)")
06120 |         self.tabel_gerhana.heading("akhir", text="Kontak Akhir (WIB)")
06121 |         self.tabel_gerhana.heading("wilayah_global", text="Karakteristik & Wilayah")
06122 |         self.tabel_gerhana.heading("indo_vis", text="Visibilitas (WIB Siang/Malam)")
06123 |
06124 |         self.tabel_gerhana.column("objek", width=80, anchor="center")
06125 |         self.tabel_gerhana.column("jenis", width=120, anchor="center")
06126 |         self.tabel_gerhana.column("mulai", width=130, anchor="center")
06127 |         self.tabel_gerhana.column("puncak", width=130, anchor="center")
06128 |         self.tabel_gerhana.column("akhir", width=130, anchor="center")
06129 |         self.tabel_gerhana.column("wilayah_global", width=220, anchor="w")
06130 |         self.tabel_gerhana.column("indo_vis", width=260, anchor="w")
06131 |
06132 |         self.tabel_gerhana.tag_configure('ganjil', background='#2b2b2b')
06133 |         self.tabel_gerhana.tag_configure('genap', background='#1e1e1e')
06134 |
06135 |         sb_gerhana = ttk.Scrollbar(frame_tabel, orient="vertical", command=self.tabel_gerhana.yview)
06136 |         self.tabel_gerhana.configure(yscroll=sb_gerhana.set)
06137 |
06138 |         self.tabel_gerhana.pack(side="left", fill="both", expand=True)
06139 |         sb_gerhana.pack(side="right", fill="y")
06140 |
06141 |         frame_btn = ctk.CTkFrame(self.frame_gerhana_out, fg_color="transparent")
06142 |         frame_btn.pack(pady=10)
06143 |
06144 |         btn_detail = ctk.CTkButton(frame_btn, text="🔍 Lihat Detail Lokal Saat Puncak", font=("Segoe UI", 12, "bold"), command=self.tampilkan_detail_lokal_gerhana)
06145 |         btn_detail.pack(side="left", padx=10)
06146 |
06147 |         self.btn_kml = ctk.CTkButton(frame_btn, text="📄 Export Jalur Totalitas/Anularitas ke KML", font=("Segoe UI", 12, "bold"), fg_color="#F57C00", hover_color="#EF6C00", command=self.export_kml_solar)
06148 |         self.btn_kml.pack(side="left", padx=10)
06149 |
06150 |         self.btn_simulasi_gerhana = ctk.CTkButton(frame_btn, text="🎮 Simulasi Visual Gerhana", font=("Segoe UI", 12, "bold"), fg_color="#673AB7", hover_color="#512DA8", command=self.buka_simulator_gerhana)
06151 |         self.btn_simulasi_gerhana.pack(side="left", padx=10)
06152 |
06153 |     def insert_gerhana_wrapped_row(self, values, tags):
06154 |         char_limits = [10, 16, 18, 18, 18, 30, 36]
06155 |         wrapped_cols = []
06156 |         max_lines = 1
06157 |
06158 |         for i, val in enumerate(values):
06159 |             wrapped = textwrap.wrap(str(val), width=char_limits[i])
06160 |             if not wrapped: wrapped = [""]
06161 |             wrapped_cols.append(wrapped)
06162 |             if len(wrapped) > max_lines: max_lines = len(wrapped)
06163 |
06164 |         for line_idx in range(max_lines):
06165 |             row_data = [col[line_idx] if line_idx < len(col) else "" for col in wrapped_cols]
06166 |             self.tabel_gerhana.insert("", "end", values=row_data, tags=tags)
06167 |
06168 |         self.tabel_gerhana.insert("", "end", values=["", "", "", "", "", "", ""], tags=tags)
06169 |
06170 |     def analisis_visibilitas_indonesia(self, jam_utc):
06171 |         if 10 <= jam_utc <= 14: return "Terlihat Jelas: Papua, Maluku, Sulawesi. (Sumatra/Jawa saat terbit)"
06172 |         elif 15 <= jam_utc <= 19: return "Terlihat Jelas Seluruh Indonesia (Tengah Malam)"
06173 |         elif 20 <= jam_utc <= 22: return "Terlihat Jelas: Sumatera, Jawa, Kalimantan. (Indonesia Timur saat terbenam)"
06174 |         else: return "Tidak Terlihat (Terjadi Siang Hari di Indonesia)"
06175 |
06176 |     def _calc_gerhana_thread(self, tahun):
06177 |         try:
06178 |             self.auto_switch_ephemeris(tahun)

```

```

06179 |         # Gunakan ts.utc murni tanpa timedelta agar aman untuk tahun berapapun
06180 |         t0 = self.ts.utc(tahun, 1, 1)
06181 |         t1 = self.ts.utc(tahun, 12, 31, 23, 59, 59)
06182 |         all_eclipses = []
06183 |         earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
06184 |
06185 |         # Titik Referensi Indonesia (Diambil Tengah: Balikpapan) untuk cek visibilitas lokal
06186 |         indo_loc = earth + wgs84.latlon(-1.2, 116.8)
06187 |
06188 |         # =====
06189 |         # PRE-KOMPUTASI DATABASE NEGARA (Agar Scanning Super Cepat)
06190 |         # =====
06191 |         perwakilan_locs = {}
06192 |         prov_indo = ["Aceh", "Bali", "Bangka Belitung", "Banten", "Bengkulu", "DI Yogyakarta", "DKI Jakarta", "Gorontalo", "Jambi", "Jawa Barat", "Jawa Tengah", "Jawa Timur", "Kalimantan Barat",
"Kalimantan Selatan", "Kalimantan Tengah", "Kalimantan Timur", "Kalimantan Utara", "Kepulauan Riau", "Lampung", "Maluku", "Maluku Utara", "NTB", "NTT", "Papua", "Papua Barat", "Papua Barat Daya", "Papua
Selatan", "Papua Tengah", "Papua Pegunungan", "Riau", "Sulawesi Barat", "Sulawesi Selatan", "Sulawesi Tengah", "Sulawesi Tenggara", "Sulawesi Utara", "Sumatera Barat", "Sumatera Selatan", "Sumatera Utara"]
06193 |
06194 |         for region, cities in CITY_DB.items():
06195 |             if not cities: continue
06196 |             nama_tampil = region
06197 |             if region in prov_indo:
06198 |                 nama_tampil = "Indonesia"
06199 |             elif "Amerika Selatan" in region: nama_tampil = "Amerika Sel."
06200 |             elif "Afrika" in region: nama_tampil = "Afrika"
06201 |             elif "Asia Selatan" in region: nama_tampil = "Timteng & Asia Sel."
06202 |             elif "Eropa" in region or "Inggris" in region: nama_tampil = "Eropa"
06203 |
06204 |             first_city = list(cities.keys())[0]
06205 |             lat, lon = cities[first_city]
06206 |             if nama_tampil not in perwakilan_locs:
06207 |                 perwakilan_locs[nama_tampil] = []
06208 |
06209 |             if len(perwakilan_locs[nama_tampil]) < 3:
06210 |                 loc_obj = earth + wgs84.latlon(lat, lon)
06211 |                 perwakilan_locs[nama_tampil].append(loc_obj)
06212 |
06213 |         # Cari seluruh fase bulan di tahun tersebut (0 = Ijtimak/New Moon, 2 = Purnama/Full Moon)
06214 |         t_phases, y_phases = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
06215 |
06216 |         if t_phases is None:
06217 |             raise ValueError(f"Data Ephemeris tidak mencakup fase bulan untuk tahun {tahun}.")
06218 |
06219 |         # --- 1. PENCARIAN GERHANA BULAN (MANUAL VECTOR MATH - MENGGANTIKAN eclipselib) ---
06220 |         t_full_moons = [t for t, phase in zip(t_phases, y_phases) if phase == 2]
06221 |         for t_fm in t_full_moons:
06222 |             tt_array = np.linspace(t_fm.tt - 0.25, t_fm.tt + 0.25, 300)
06223 |             t_arr = self.ts.tt_jd(tt_array)
06224 |             e_pos = earth.at(t_arr)
06225 |             s_app = e_pos.observe(sun).apparent()
06226 |             m_app = e_pos.observe(moon).apparent()
06227 |
06228 |             seps_from_sun = s_app.separation_from(m_app).degrees
06229 |             seps_from_shadow = 180.0 - seps_from_sun
06230 |
06231 |             min_idx = np.argmin(seps_from_shadow)
06232 |             min_sep = seps_from_shadow[min_idx]
06233 |
06234 |             t_p = t_arr[min_idx]
06235 |             s_dist = earth.at(t_p).observe(sun).apparent().distance().km
06236 |             m_dist = earth.at(t_p).observe(moon).apparent().distance().km
06237 |
06238 |             sd_m = math.degrees(math.asin(1737.4 / m_dist))
06239 |             pi_m = math.degrees(math.asin(6378.14 / m_dist))
06240 |             sd_s = math.degrees(math.asin(696000.0 / s_dist))
06241 |             pi_s = 8.794 / 3600.0
06242 |
06243 |             r_umbra = (pi_m + pi_s - sd_s) * 1.02
06244 |             r_penumbra = (pi_m + pi_s + sd_s) * 1.02
06245 |
06246 |             # Deteksi jika masuk bayangan bumi
06247 |             if min_sep < (r_penumbra + sd_m):

```

```

06248 |         if min_sep < (r_umbra - sd_m): jenis = "Total"
06249 |         elif min_sep < (r_umbra + sd_m): jenis = "Sebagian (Partial)"
06250 |         else: jenis = "Penumbra"
06251 |
06252 |         m_alt_indo, _, _ = indo_loc.at(t_p).observe(moon).apparent().altaz()
06253 |         indo_vis = "Terlihat Jelas (Bulan di atas ufuk)" if m_alt_indo.degrees > 0 else "Tidak Terlihat (Siang di Indonesia)"
06254 |
06255 |         wilayah_terlihat = set()
06256 |         for nama_wil, list_loc in perwakilan_locs.items():
06257 |             for loc in list_loc:
06258 |                 alt_obj, _, _ = loc.at(t_p).observe(moon).apparent().altaz()
06259 |                 if alt_obj.degrees > 0:
06260 |                     wilayah_terlihat.add(nama_wil)
06261 |                     break
06262 |
06263 |         wil_str = ", ".join(sorted(list(wilayah_terlihat))) if wilayah_terlihat else "Hanya terlihat di Samudra Terpencil"
06264 |
06265 |         all_eclipses.append({
06266 |             'objek': 'Bulan',
06267 |             'jenis': jenis,
06268 |             't_peak': t_p,
06269 |             't_mulai': self.ts.tt_jd(t_p.tt - 0.15),
06270 |             't_akhir': self.ts.tt_jd(t_p.tt + 0.15),
06271 |             'wilayah': wil_str,
06272 |             'indo_vis': indo_vis
06273 |         })
06274 |
06275 | # --- 2. PENCARIAN GERHANA MATAHARI ---
06276 | t_new_moons = [t for t, phase in zip(t_phases, y_phases) if phase == 0]
06277 | for t_nm in t_new_moons:
06278 |     tt_array = np.linspace(t_nm.tt - 0.25, t_nm.tt + 0.25, 300)
06279 |     t_arr = self.ts.tt_jd(tt_array)
06280 |     e_pos = earth.at(t_arr)
06281 |     seps = e_pos.observe(sun).apparent().separation_from(e_pos.observe(moon).apparent()).degrees
06282 |     min_idx = np.argmin(seps)
06283 |
06284 |     if seps[min_idx] < 1.6:
06285 |         t_p = t_arr[min_idx]
06286 |         dist_s = earth.at(t_p).observe(sun).apparent().distance().km
06287 |         dist_m = earth.at(t_p).observe(moon).apparent().distance().km
06288 |         sd_s = math.degrees(math.asin(696000.0 / dist_s))
06289 |         sd_m = math.degrees(math.asin(1737.4 / dist_m))
06290 |
06291 |         jenis = "Total" if sd_m > sd_s else "Cincin (Annular)"
06292 |         if seps[min_idx] > abs(sd_s - sd_m): jenis = "Sebagian (Partial)"
06293 |
06294 |         s_app_indo = indo_loc.at(t_p).observe(sun).apparent()
06295 |         m_app_indo = indo_loc.at(t_p).observe(moon).apparent()
06296 |         alt_s_indo, _, _ = s_app_indo.altaz()
06297 |         sep_indo = s_app_indo.separation_from(m_app_indo).degrees
06298 |
06299 |         if alt_s_indo.degrees > 0:
06300 |             if sep_indo < (sd_s + sd_m + 0.6):
06301 |                 indo_vis = "Terlihat (Bayangan Gerhana melintasi Indonesia)"
06302 |             else:
06303 |                 indo_vis = "Tidak Terlihat (Meleset dari lintang Indonesia)"
06304 |         else:
06305 |             indo_vis = "Tidak Terlihat (Terjadi saat Malam di Indonesia)"
06306 |
06307 |         wilayah_terlihat = set()
06308 |         for nama_wil, list_loc in perwakilan_locs.items():
06309 |             for loc in list_loc:
06310 |                 s_app = loc.at(t_p).observe(sun).apparent()
06311 |                 alt_s_wil, _, _ = s_app.altaz()
06312 |                 if alt_s_wil.degrees > 0:
06313 |                     m_app = loc.at(t_p).observe(moon).apparent()
06314 |                     sep_wil = s_app.separation_from(m_app).degrees
06315 |                     if sep_wil < (sd_s + sd_m + 0.8):
06316 |                         wilayah_terlihat.add(nama_wil)
06317 |                         break
06318 |

```

```

06319 |         wil_str = ", ".join(sorted(list(wilayah_terlihat))) if wilayah_terlihat else "Hanya melintasi Samudra / Antartika"
06320 |
06321 |         all_eclipses.append({
06322 |             'objek': 'Matahari',
06323 |             'jenis': jenis,
06324 |             't_peak': t_p,
06325 |             't_mulai': self.ts.tt_jd(t_p.tt - 0.12),
06326 |             't_akhir': self.ts.tt_jd(t_p.tt + 0.12),
06327 |             'wilayah': wil_str,
06328 |             'indo_vis': indo_vis
06329 |         })
06330 |
06331 |     # Jika data gerhana benar-benar kosong
06332 |     if len(all_eclipses) == 0:
06333 |         self.after(0, lambda: messagebox.showinfo("Informasi Ephemeris",
06334 |             f"Tidak ada peristiwa gerhana yang ditemukan pada tahun {tahun}.\n\n"
06335 |             "Hal ini wajar jika terjadi pada tahun tertentu, atau pastikan file ephemeris 'de440s.bsp' / 'de441-new.bsp' sesuai rentang tidak korup. "
06336 |             "Rentang yang dipakai aplikasi adalah 1 hingga 2999 M."))
06337 |         self.after(0, lambda: self.lbl_status.configure(text=f"Tidak ada data gerhana {tahun}", text_color="#FFAB40"))
06338 |     else:
06339 |         all_eclipses.sort(key=lambda x: x['t_peak'].tt)
06340 |
06341 |     self.after(0, self._post_hitung_gerhana, all_eclipses, tahun)
06342 |
06343 | except Exception as e:
06344 |     import traceback
06345 |     self.after(0, self.display_error, f"Gagal Scan Gerhana Tahun {tahun}: {str(e)}\n\nPastikan format tahun benar dan rentang berada di dalam file .bsp (Misal: 1 hingga
3000).\n\n{traceback.format_exc()}")
06346 |
06347 | def _post_hitung_gerhana(self, all_eclipses, tahun):
06348 |     self.tabel_gerhana.delete(*self.tabel_gerhana.get_children())
06349 |     count = 0
06350 |
06351 |     def format_waktu(t_obj):
06352 |         if t_obj is None: return "---"
06353 |         t_local = self.ts.tt_jd(t_obj.tt + 7.0/24.0) # Konversi ke WIB
06354 |         y, m, d, h, mn, s = t_local.utc
06355 |         yr = int(y)
06356 |         txt_yr = f"{yr:04d}" if yr > 0 else f"{abs(yr-1):04d} SM"
06357 |         return f"{int(d):02d}-{int(m):02d}-{txt_yr} {int(h):02d}:{int(mn):02d}"
06358 |
06359 |     for ev in all_eclipses:
06360 |         w_mulai = format_waktu(ev['t_mulai'])
06361 |         w_puncak = format_waktu(ev['t_peak'])
06362 |         w_akhir = format_waktu(ev['t_akhir'])
06363 |
06364 |         tag = 'ganjil' if count % 2 == 0 else 'genap'
06365 |         self.insert_gerhana_wrapped_row((
06366 |             ev['objek'], ev['jenis'], w_mulai, w_puncak, w_akhir, ev['wilayah'], ev['indo_vis']
06367 |         ), (tag,))
06368 |         count += 1
06369 |
06370 |     self.lbl_status.configure(text=f"Analisis Gerhana {tahun} Selesai", text_color="#00E676")
06371 |     self.btn_hitung.configure(state="normal")
06372 |
06373 | def _parse_waktu_tabel_gerhana(self, waktu_str):
06374 |     """Fungsi Anti-Crash untuk mengkonversi string tabel kembali ke format Waktu Skyfield (Kebal Tahun SM)"""
06375 |     import re
06376 |     # Format string dari tabel: "dd-mm-yyyy HH:MM" atau "dd-mm-yyyy SM HH:MM"
06377 |     match = re.match(r"(\d+)-(\d+)-(\d+)\s*(SM)?\s*(\d+):(\d+)", waktu_str)
06378 |     if not match: return None
06379 |     d, m, y, is_sm, h, mn = match.groups()
06380 |     y = int(y)
06381 |     if is_sm: y = -(y - 1)
06382 |     # Kurangi 7 Jam karena format tabel adalah WIB (UTC+7)
06383 |     return self.ts.utc(y, int(m), int(d), int(h) - 7, int(mn))
06384 |
06385 | def export_kml_solar(self):
06386 |     selected_item = self.tabel_gerhana.selection()
06387 |     if not selected_item:
06388 |         messagebox.showwarning("Peringatan", "Silakan pilih jadwal Gerhana Matahari di tabel terlebih dahulu.")

```

```

06389 |         return
06390 |
06391 |         vals = self.tabel_gerhana.item(selected_item[0])['values']
06392 |         if "Matahari" not in str(vals[0]):
06393 |             messagebox.showinfo("Info", "Pilih data Gerhana Matahari, bukan Bulan.")
06394 |             return
06395 |         if "Sebagian" in str(vals[1]):
06396 |             messagebox.showinfo("Info", "Gerhana Sebagian (Partial) tidak memiliki Jalur Sentral Umbra/Antumbra (Totality/Annular).")
06397 |             return
06398 |
06399 |         waktu_puncak_str = str(vals[3]).strip()
06400 |         jenis = str(vals[1]).strip()
06401 |
06402 |         try:
06403 |             # Gunakan Parser Anti-Crash
06404 |             t_peak = self._parse_waktu_tabel_gerhana(waktu_puncak_str)
06405 |             if t_peak is None: raise ValueError("Gagal membaca format waktu dari tabel.")
06406 |
06407 |             t_start = self.ts.tt_jd(t_peak.tt - 4.0/24.0)
06408 |             t_end = self.ts.tt_jd(t_peak.tt + 4.0/24.0)
06409 |             t_arr = self.ts.tt_jd(np.linspace(t_start.tt, t_end.tt, 1000))
06410 |
06411 |             earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
06412 |
06413 |             e_pos = earth.at(t_arr)
06414 |             M = e_pos.observe(moon).position.km
06415 |             S = e_pos.observe(sun).position.km
06416 |
06417 |             V = M - S
06418 |             norm_V = np.linalg.norm(V, axis=0)
06419 |             v_hat = V / norm_V
06420 |
06421 |             M_dot_v = np.sum(M * v_hat, axis=0)
06422 |             M_sq = np.sum(M**2, axis=0)
06423 |
06424 |             R_E = 6371.0
06425 |             b = 2.0 * M_dot_v
06426 |             c = M_sq - R_E**2
06427 |             delta = b**2 - 4.0 * c
06428 |
06429 |             valid_idx = delta >= 0
06430 |             if not np.any(valid_idx):
06431 |                 messagebox.showwarning("Peringatan", "Jalur sentral meleset dari permukaan Bumi (tidak ada daratan yang dilewati sumbu bayangan pusat).")
06432 |                 return
06433 |
06434 |             b_val = b[valid_idx]
06435 |             delta_val = delta[valid_idx]
06436 |             M_val = M[:, valid_idx]
06437 |             v_hat_val = v_hat[:, valid_idx]
06438 |             t_valid = t_arr.tt[valid_idx]
06439 |
06440 |             k = (-b_val - np.sqrt(delta_val)) / 2.0
06441 |             P_km = M_val + v_hat_val * k
06442 |
06443 |             P_au = P_km / 149597870.7
06444 |
06445 |             coords_list = []
06446 |             from skyfield.positionlib import Geocentric
06447 |             for i in range(len(t_valid)):
06448 |                 geo = Geocentric(position_au=P_au[:, i], t=self.ts.tt_jd(t_valid[i]))
06449 |                 subpt = wgs84.subpoint(geo)
06450 |                 coords_list.append(f"{subpt.longitude.degrees},{subpt.latitude.degrees},0")
06451 |
06452 |             kml_coords = "\n                ".join(coords_list)
06453 |
06454 |             kml_content = f'<?xml version="1.0" encoding="UTF-8"?>
06455 | <kml xmlns="http://www.opengis.net/kml/2.2">
06456 | <Document>
06457 | <name>Jalur Gerhana Matahari {waktu_puncak_str}</name>
06458 | <description>Tipe: {jenis}</description>
06459 | <Style id="pathStyle">

```



```

06460 |     <LineStyle>
06461 |         <color>7f0000ff</color> <width>5</width>
06462 |     </LineStyle>
06463 | </Style>
06464 | <Placemark>
06465 |     <name>Path of Totality / Annularity (Center Line)</name>
06466 |     <styleUrl>#pathStyle</styleUrl>
06467 |     <LineString>
06468 |         <tessellate>1</tessellate>
06469 |         <coordinates>
06470 |             {kml_coords}
06471 |         </coordinates>
06472 |     </LineString>
06473 | </Placemark>
06474 | </Document>
06475 | </kml>"""
06476 |
06477 |     # Rapikan nama file agar bersih dari spasi dan simbol
06478 |     safe_name = waktu_puncak_str.replace(" ", "_").replace(":", "")
06479 |     filepath = filedialog.asksaveasfilename(
06480 |         initialfile=f"Jalur_Gerhana_Matahari_{safe_name}.kml",
06481 |         defaultextension=".kml",
06482 |         filetypes=[("KML Files", "*.kml")]
06483 |     )
06484 |
06485 |     if filepath:
06486 |         with open(filepath, 'w', encoding='utf-8') as f:
06487 |             f.write(kml_content)
06488 |             messagebox.showinfo("Sukses", f"File Jalur Gerhana KML berhasil diekspor:\n{filepath}\n\nSilakan buka file ini menggunakan Google Earth untuk melihat simulasi jalurnya.")
06489 |
06490 |     except Exception as e:
06491 |         import traceback
06492 |         traceback.print_exc()
06493 |         messagebox.showerror("Error", f"Gagal export KML: {e}")
06494 |
06495 | def tampilkan_detail_lokal_gerhana(self):
06496 |     selected_item = self.tabel_gerhana.selection()
06497 |     if not selected_item:
06498 |         messagebox.showwarning("Peringatan", "Silakan klik/pilih salah satu jadwal gerhana di tabel terlebih dahulu.")
06499 |         return
06500 |
06501 |     item_values = self.tabel_gerhana.item(selected_item[0])['values']
06502 |     if not item_values or item_values[0] == "": return
06503 |
06504 |     objek = str(item_values[0]).strip()
06505 |     waktu_puncak_str = str(item_values[3]).strip()
06506 |     if not waktu_puncak_str: return
06507 |
06508 |     try:
06509 |         # Gunakan Parser Anti-Crash
06510 |         waktu_puncak_skyfield = self._parse_waktu_tabel_gerhana(waktu_puncak_str)
06511 |         if waktu_puncak_skyfield is None: raise ValueError("Gagal membaca format waktu dari tabel.")
06512 |
06513 |         earth = self.eph['earth']
06514 |         target_objek = self.eph['moon'] if "Bulan" in objek else self.eph['sun']
06515 |
06516 |         try:
06517 |             lat = float(self.entry_vlat.get())
06518 |             lon = float(self.entry_vlon.get())
06519 |         except:
06520 |             lat, lon = -7.0667, 110.4100
06521 |
06522 |         lokasi_observasi = earth + wgs84.latlon(lat, lon)
06523 |         astrometric = lokasi_observasi.at(waktu_puncak_skyfield).observe(target_objek)
06524 |         alt, az, distance = astrometric.apparent().altaz()
06525 |
06526 |         extra_info = ""
06527 |         if "Bulan" in objek:
06528 |             iluminasi = almanac.fraction_illuminated(self.eph, 'moon', waktu_puncak_skyfield) * 100.0
06529 |             dt_utc_peak = waktu_puncak_skyfield.utc_datetime()
06530 |             t0 = self.ts.utc(dt_utc_peak - datetime.timedelta(days=35))

```

```

06531 |         t1 = waktu_puncak_skyfield
06532 |
06533 |         fase_t, fase_y = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
06534 |         waktu_new_moon = [t for t, y in zip(fase_t, fase_y) if y == 0]
06535 |
06536 |         if waktu_new_moon:
06537 |             last_new_moon = waktu_new_moon[-1]
06538 |             moon_age_actual_days = waktu_puncak_skyfield.tt - last_new_moon.tt
06539 |
06540 |             app_moon = earth.at(waktu_puncak_skyfield).observe(self.eph['moon']).apparent()
06541 |             app_sun = earth.at(waktu_puncak_skyfield).observe(self.eph['sun']).apparent()
06542 |             _, m_lon, _ = app_moon.ecliptic_latlon()
06543 |             _, s_lon, _ = app_sun.ecliptic_latlon()
06544 |
06545 |             phase_angle = (m_lon.degrees - s_lon.degrees) % 360.0
06546 |             moon_age_phase_days = (phase_angle / 360.0) * 29.530588861
06547 |
06548 |             extra_info = f"\nIluminasi (Fase): {iluminasi:.2f}%\nUmur Aktual (True Time Elapsed): {moon_age_actual_days:.4f} Hari\nUmur Fase Sudut (Siklik/Elongasi): {moon_age_phase_days:.4f} Hari"
06549 |         else:
06550 |             extra_info = f"\nIluminasi (Fase): {iluminasi:.2f}%\nUmur Bulan: N/A"
06551 |     else:
06552 |         app_moon = lokasi_observasi.at(waktu_puncak_skyfield).observe(self.eph['moon']).apparent()
06553 |         m_alt, m_az, _ = app_moon.altaz()
06554 |         elong = astrometric.separation_from(app_moon).degrees
06555 |         extra_info = f"\nAltitude Bulan: {m_alt.degrees:.2f}°\nElongasi Sudut (Toposentrik): {elong:.2f}°\n"
06556 |
06557 |         if alt.degrees > 0:
06558 |             extra_info += ">> MATAHARI DI ATAS UFUK: Gerhana mungkin teramati dari lokasi ini jika elongasi mendekati 0°."
06559 |         else:
06560 |             extra_info += ">> MATAHARI DI BAWAH UFUK: Gerhana terjadi saat malam di lokasi ini."
06561 |
06562 |     pesan = (
06563 |         f"LOKASI OBSERVASI (GPS)\n"
06564 |         f"{self.lokasi_nama.get()}\n"
06565 |         f"{'.'*40}\n"
06566 |         f"Waktu Puncak Global (WIB): {waktu_puncak_str}\n"
06567 |         f"Waktu Puncak Global (UTC): {waktu_puncak_skyfield.utc_strftime('%Y-%m-%d %H:%M:%S')}\n"
06568 |         f"{'.'*40}\n"
06569 |         f"PARAMETER VISUAL {objek.upper()} (TOPOSENTRIK)\n"
06570 |         f"Altitude (Tinggi ufuk): {alt.degrees:.2f}°\n"
06571 |         f"Azimuth (Arah kompas): {az.degrees:.2f}°\n"
06572 |         f"Jarak Bumi-{objek}: {distance.km:.0f} km\n"
06573 |         f"{extra_info}"
06574 |     )
06575 |
06576 |     win_detail = ctk.CTkToplevel(self)
06577 |     win_detail.title(f"Detail Parameter Lokal - {objek}")
06578 |     win_detail.geometry("550x450")
06579 |     win_detail.attributes("-topmost", True)
06580 |
06581 |     lbl_title = ctk.CTkLabel(win_detail, text=f"Detail Visual {objek} Saat Puncak", font=("Segoe UI", 16, "bold"), text_color="#00E5FF")
06582 |     lbl_title.pack(pady=(15, 5))
06583 |
06584 |     textbox = ctk.CTkTextbox(win_detail, width=500, height=350, font=("Consolas", 13), wrap="word", fg_color="#1e1e1e")
06585 |     textbox.pack(padx=20, pady=10, fill="both", expand=True)
06586 |     textbox.insert("1.0", pesan)
06587 |     textbox.configure(state="disabled")
06588 |
06589 | except Exception as e:
06590 |     messagebox.showerror("Error", f"Terjadi kesalahan saat menghitung detail parameter: {e}")
06591 |
06592 | # =====
06593 | # MODUL TAMBAHAN 15/36: LIVE ANIMASI (FULL SCROLL & RASI BINTANG)
06594 | # =====
06595 | def setup_animasi_out_frame(self):
06596 |     # 1. UBAH MENJADI SCROLLABLE FRAME AGAR BISA SCROLL ATAS BAWAH
06597 |     self.frame_animasi_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="#050510", corner_radius=10)
06598 |
06599 |     frame_atas = ctk.CTkFrame(self.frame_animasi_out, fg_color="#181818", corner_radius=8)
06600 |     frame_atas.pack(fill="x", padx=15, pady=10)
06601 |

```

```

06602 |         ctk.CTkLabel(frame_atas, text=" LIVE SIMULATOR", font=("Segoe UI", 16, "bold"), text_color="#FF5252").pack(side="left", padx=15, pady=10)
06603 |
06604 |         self.is_anim_zoomed = False
06605 |
06606 |         # Penempatan tombol-tombol (Urutan pack(side="right") ditumpuk dari Kanan ke Kiri)
06607 |         # 1. Tombol Zoom (Paling Kanan)
06608 |         self.btn_anim_zoom = ctk.CTkButton(frame_atas, text=" Zoom Hilal", font=("Segoe UI", 12, "bold"), fg_color="#673AB7", hover_color="#512DA8", command=self.toggle_anim_zoom)
06609 |         self.btn_anim_zoom.pack(side="right", padx=(5, 15), pady=10)
06610 |
06611 |         # 2. Tombol Tracker (Sebelah kiri Zoom)
06612 |         self.btn_tracker_link = ctk.CTkButton(
06613 |             frame_atas,
06614 |             text=" Tracker",
06615 |             font=("Segoe UI", 12, "bold"),
06616 |             fg_color="#00838F",
06617 |             hover_color="#006064",
06618 |             command=lambda: webbrowser.open("https://hisabmu.com/khgttimes/tracker/")
06619 |         )
06620 |         self.btn_tracker_link.pack(side="right", padx=(5, 5), pady=10)
06621 |
06622 |         # 3. Tombol Komparasi Stellarium (Samping kiri Tracker)
06623 |         self.btn_komparasi_stellarium = ctk.CTkButton(
06624 |             frame_atas,
06625 |             text=" Stellarium Sync",
06626 |             font=("Segoe UI", 12, "bold"),
06627 |             fg_color="#E65100",
06628 |             hover_color="#BF360C",
06629 |             command=self.buka_komparasi_stellarium
06630 |         )
06631 |         self.btn_komparasi_stellarium.pack(side="right", padx=(5, 5), pady=10)
06632 |
06633 |         # 4. Label Lokasi (Paling Kiri sebelum tombol-tombol)
06634 |         self.lb_anim_lokasi = ctk.CTkLabel(frame_atas, textvariable=self.lokasi_nama, font=("Consolas", 12), text_color="#00E5FF")
06635 |         self.lb_anim_lokasi.pack(side="right", padx=15)
06636 |
06637 |         # 2. SET TINGGI KANVAS (1400px) AGAR RENTANG 90° HINGGA -90° TERLIHAT
06638 |         self.anim_canvas = tk.Canvas(self.frame_animasi_out, bg='#030514', highlightthickness=0, height=1400)
06639 |         self.anim_canvas.pack(fill="both", expand=True, padx=15, pady=(0, 15))
06640 |         self.anim_canvas.bind("<Configure>", self.draw_static_background)
06641 |
06642 |         # --- VARIABEL STATE UNTUK KOTAK INFO INTERAKTIF ---
06643 |         self.show_anim_info_box = True
06644 |         self.anim_cached_date = None
06645 |         self.anim_cached_events = {}
06646 |
06647 |         # --- EVENT BINDING UNTUK INTERAKSI KANVAS ---
06648 |         self.anim_canvas.tag_bind("btn_close", "<Button-l>", lambda e: self.toggle_anim_info(False))
06649 |         self.anim_canvas.tag_bind("btn_close", "<Enter>", lambda e: self.anim_canvas.config(cursor="hand2"))
06650 |         self.anim_canvas.tag_bind("btn_close", "<Leave>", lambda e: self.anim_canvas.config(cursor=""))
06651 |
06652 |         self.anim_canvas.tag_bind("sun_obj", "<Button-l>", lambda e: self.toggle_anim_info(True))
06653 |         self.anim_canvas.tag_bind("sun_obj", "<Enter>", lambda e: self.anim_canvas.config(cursor="hand2"))
06654 |         self.anim_canvas.tag_bind("sun_obj", "<Leave>", lambda e: self.anim_canvas.config(cursor=""))
06655 |
06656 |         self.anim_canvas.tag_bind("moon_obj", "<Button-l>", lambda e: self.toggle_anim_info(True))
06657 |         self.anim_canvas.tag_bind("moon_obj", "<Enter>", lambda e: self.anim_canvas.config(cursor="hand2"))
06658 |         self.anim_canvas.tag_bind("moon_obj", "<Leave>", lambda e: self.anim_canvas.config(cursor=""))
06659 |
06660 |     def buka_komparasi_stellarium(self):
06661 |         """Memanggil jendela interaktif komparasi data Live Stellarium vs Ephemeris"""
06662 |         KomparasiObservatorium(self, self.eph, self.ts)
06663 |
06664 |     def draw_static_background(self, event=None):
06665 |         """Menggambar Ufuk, Garis Zenith/Nadir, dan Skala Kompas"""
06666 |         if getattr(self, 'is_anim_zoomed', False):
06667 |             self.anim_canvas.delete("static")
06668 |             return
06669 |
06670 |         self.anim_canvas.delete("static")
06671 |         width = self.anim_canvas.winfo_width()
06672 |         height = self.anim_canvas.winfo_height()

```

```

06673 |         if height <= 10: return
06674 |
06675 |         horizon_y = height / 2
06676 |         pad_x = 45
06677 |         usable_width = width - (2 * pad_x)
06678 |
06679 |         # Ubah warna area bawah ufuk menjadi transisi warna bumi gelap
06680 |         self.anim_canvas.create_rectangle(0, horizon_y, width, height, fill="#0A1010", outline="", tags="static")
06681 |
06682 |         # Garis Ufuk 0 derajat
06683 |         self.anim_canvas.create_line(0, horizon_y, width, horizon_y, fill="#00E676", width=2, dash=(4, 2), tags="static")
06684 |         self.anim_canvas.create_text(10, horizon_y - 12, text="GARIS UFUK (0°)", fill="#00E676", font=("Consolas", 10, "bold"), anchor="w", tags="static")
06685 |
06686 |         # Gambar Bintang Latar Acak di SELURUH KANVAS (+90 s/d -90)
06687 |         for _ in range(350):
06688 |             x = random.randint(0, width)
06689 |             y = random.randint(0, height)
06690 |             r = random.uniform(0.5, 1.5)
06691 |             color = random.choice(["FFFFFF", "#E0F7FA", "#FFDE7", "#555555"])
06692 |             self.anim_canvas.create_oval(x-r, y-r, x+r, y+r, fill=color, outline="", tags="static")
06693 |
06694 |         # Skala Kompas Standar (Tetap di ufuk)
06695 |         kompas = {0: "U (0°)", 45: "TL (45°)", 90: "T (90°)", 135: "TG (135°)", 180: "S (180°)", 225: "BD (225°)", 270: "B (270°)", 315: "BL (315°)", 360: "U (360°)"}
06696 |         for az, label in kompas.items():
06697 |             x = pad_x + (az / 360.0) * usable_width
06698 |             self.anim_canvas.create_text(x, horizon_y + 15, text=label, fill="white", font=("Consolas", 9), tags="static")
06699 |             self.anim_canvas.create_line(x, horizon_y, x, horizon_y + 6, fill="white", tags="static")
06700 |
06701 |         # Garis Zenith (+90) dan Nadir (-90)
06702 |         self.anim_canvas.create_line(0, 30, width, 30, fill="#333333", width=1, dash=(2, 4), tags="static")
06703 |         self.anim_canvas.create_text(10, 40, text="ZENITH (+90°)", fill="#777777", font=("Consolas", 10, "bold"), anchor="w", tags="static")
06704 |
06705 |         self.anim_canvas.create_line(0, height-30, width, height-30, fill="#333333", width=1, dash=(2, 4), tags="static")
06706 |         self.anim_canvas.create_text(10, height-40, text="NADIR (-90°)", fill="#777777", font=("Consolas", 10, "bold"), anchor="w", tags="static")
06707 |
06708 |     def update_animation(self):
06709 |         if not getattr(self, 'anim_running', False): return
06710 |
06711 |         width = self.anim_canvas.winfo_width()
06712 |         height = self.anim_canvas.winfo_height()
06713 |         if width < 10 or height < 10:
06714 |             self.after(500, self.update_animation)
06715 |             return
06716 |
06717 |         self.anim_canvas.delete("dyn")
06718 |
06719 |     def get_compass_dir(deg):
06720 |         dirs = ["U", "UTL", "TL", "TTL", "T", "TM", "TG", "SM", "S", "SBD", "BD", "BBD", "B", "BBL", "BL", "UBL"]
06721 |         idx = int(round((deg % 360) / 22.5)) % 16
06722 |         return dirs[idx]
06723 |
06724 |     try:
06725 |         try: lat, lon = float(self.entry_vlat.get()), float(self.entry_vlon.get())
06726 |         except: lat, lon = -7.0667, 110.4100
06727 |         tz_offset = int(self.get_tz_from_lon(lon))
06728 |
06729 |         if getattr(self, 'anim_is_live', True):
06730 |             t_sim = self.ts.now()
06731 |             status = "🟢 LIVE (Waktu Berjalan)"
06732 |         else:
06733 |             if getattr(self, 'anim_custom_time', None) is None:
06734 |                 self.anim_custom_time = self.ts.now()
06735 |             t_sim = self.anim_custom_time
06736 |             status = "🟡 KUSTOM (Waktu Dihentikan)"
06737 |
06738 |         dt_lokal = t_sim.utc_datetime() + datetime.timedelta(hours=tz_offset)
06739 |         tz_str = f"UTC{'+' if tz_offset >= 0 else ''}{tz_offset}"
06740 |         waktu_str = f"{dt_lokal.strftime('%d-%m-%Y %H:%M:%S')} ({tz_str)}\nStatus: {status}"
06741 |
06742 |         earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
06743 |         loc = wgs84.latlon(lat, lon)

```

```

06744 | observer = earth + loc
06745 |
06746 | # Ambil suhu dan tekanan default atau dari input UI jika Anda sudah menyediakannya
06747 | try:
06748 |     temp_c = float(self.entry_vtemp.get())
06749 |     pres_mb = float(self.entry_vpres.get())
06750 | except Exception:
06751 |     temp_c = 25.0
06752 |     pres_mb = 1010.0
06753 |
06754 | astro_sun = observer.at(t_sim).observe(sun).apparent()
06755 | alt_sun, az_sun, _ = astro_sun.altaz(temperature_C=temp_c, pressure_mbar=pres_mb)
06756 |
06757 | astro_moon = observer.at(t_sim).observe(moon).apparent()
06758 | alt_moon, az_moon, _ = astro_moon.altaz(temperature_C=temp_c, pressure_mbar=pres_mb)
06759 |
06760 | # ---> TAMBAHAN: KALKULASI SEMUA PLANET <---
06761 | planets_target = [
06762 |     ('mercury barycenter', 'Merkurius', '#B0BEC5'),
06763 |     ('venus barycenter', 'Venus', '#FFCC80'),
06764 |     ('mars barycenter', 'Mars', '#EF5350'),
06765 |     ('jupiter barycenter', 'Jupiter', '#FFB74D'),
06766 |     ('saturn barycenter', 'Saturnus', '#FFE082'),
06767 |     ('uranus barycenter', 'Uranus', '#81D4FA'),
06768 |     ('neptune barycenter', 'Neptunus', '#5C6BC0'),
06769 |     ('pluto barycenter', 'Pluto', '#CFD8DC')
06770 | ]
06771 |
06772 | planet_positions = []
06773 | for p_key, p_name, p_color in planets_target:
06774 |     try:
06775 |         try:
06776 |             p_obj = self.eph[p_key]
06777 |         except KeyError:
06778 |             p_obj = self.eph[p_key.split()][0]
06779 |
06780 |         p_astro = observer.at(t_sim).observe(p_obj).apparent()
06781 |         p_alt, p_az, _ = p_astro.altaz()
06782 |         planet_positions.append({
06783 |             'name': p_name,
06784 |             'color': p_color,
06785 |             'alt': p_alt.degrees,
06786 |             'az': p_az.degrees
06787 |         })
06788 |     except Exception:
06789 |         pass
06790 | # -----
06791 |
06792 | geo_sun = earth.at(t_sim).observe(sun).apparent()
06793 | geo_moon = earth.at(t_sim).observe(moon).apparent()
06794 |
06795 | ra_s, dec_s, _ = geo_sun.radec(epoch=t_sim)
06796 | ra_m, dec_m, _ = geo_moon.radec(epoch=t_sim)
06797 |
06798 | gast = t_sim.gast
06799 | lst_deg = (gast * 15.0) + lon
06800 | lat_rad = math.radians(lat)
06801 |
06802 | def calc_geo_alt(ra, dec):
06803 |     ra_h = ra.hours.item() if hasattr(ra.hours, 'item') else ra.hours
06804 |     dec_r = dec.radians.item() if hasattr(dec.radians, 'item') else dec.radians
06805 |     ha_rad = math.radians(lst_deg - (ra_h * 15.0))
06806 |     sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) * math.cos(ha_rad)
06807 |     return math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
06808 |
06809 | alt_sun_geo = calc_geo_alt(ra_s, dec_s)
06810 | alt_moon_geo = calc_geo_alt(ra_m, dec_m)
06811 |
06812 | elong_geo = geo_sun.separation_from(geo_moon).degrees
06813 |
06814 | _, m_lon, _ = astro_moon.ecliptic_latlon()

```

```

06815 |         _, s_lon, _ = astro_sun.ecliptic_latlon()
06816 |         phase_angle = (m_lon.degrees - s_lon.degrees) % 360.0
06817 |         moon_idx = int(phase_angle) % 360
06818 |
06819 |         waktu_sekarang = ephemeris.Date(t_sim.utc_datetime())
06820 |         ijtihak = ephemeris.previous_new_moon(waktu_sekarang)
06821 |         moon_age = waktu_sekarang - ijtihak
06822 |
06823 |         # =====
06824 |         # PROYEKSI VISUAL (NORMAL VS ZOOM)
06825 |         # =====
06826 |         alt_s_deg = alt_sun.degrees
06827 |         az_s_deg = az_sun.degrees
06828 |         alt_m_deg = alt_moon.degrees
06829 |         az_m_deg = az_moon.degrees
06830 |
06831 |         alt_top_sun = alt_s_deg + 0.833
06832 |         alt_top_moon = alt_m_deg + 0.833
06833 |
06834 |         pad_x = 45
06835 |         usable_width = width - (2 * pad_x)
06836 |
06837 |         if getattr(self, 'is_anim_zoomed', False):
06838 |             # ---- 1. MODE ZOOM HILAL ----
06839 |             diff_az = az_m_deg - az_s_deg
06840 |             if diff_az > 180: diff_az -= 360
06841 |             elif diff_az < -180: diff_az += 360
06842 |
06843 |             center_az = az_s_deg + (diff_az / 2.0)
06844 |             center_alt = (alt_s_deg + alt_m_deg) / 2.0
06845 |
06846 |             delta_az = abs(diff_az)
06847 |             delta_alt = abs(alt_m_deg - alt_s_deg)
06848 |
06849 |             if delta_az < 2.0: delta_az = 2.0
06850 |             if delta_alt < 2.0: delta_alt = 2.0
06851 |
06852 |             pad_zoom_x = 120
06853 |             pad_zoom_y = 120
06854 |             available_w = max(100, width - (2 * pad_zoom_x))
06855 |             available_h = max(100, height - (2 * pad_zoom_y))
06856 |
06857 |             scale_x = available_w / delta_az
06858 |             scale_y = available_h / delta_alt
06859 |             px_per_deg = min(scale_x, scale_y)
06860 |
06861 |             if px_per_deg > 80.0: px_per_deg = 80.0
06862 |             if px_per_deg < 5.0: px_per_deg = 5.0
06863 |
06864 |             px_per_deg_x = px_per_deg
06865 |             px_per_deg_y = px_per_deg
06866 |
06867 |             horizon_y = (height / 2.0) + (center_alt * px_per_deg_y)
06868 |
06869 |             r_sun = max(30, min(80, int(0.8 * px_per_deg)))
06870 |             r_moon = max(20, min(60, int(0.6 * px_per_deg)))
06871 |
06872 |             def get_zoomed_x(az_target):
06873 |                 d_az = az_target - center_az
06874 |                 if d_az > 180: d_az -= 360
06875 |                 elif d_az < -180: d_az += 360
06876 |                 return (width / 2.0) + (d_az * px_per_deg_x)
06877 |
06878 |             x_sun = get_zoomed_x(az_s_deg)
06879 |             x_moon = get_zoomed_x(az_m_deg)
06880 |
06881 |             self.anim_canvas.create_line(0, horizon_y, width, horizon_y, fill="#00E676", width=3, tags="dyn")
06882 |             self.anim_canvas.create_text(10, horizon_y - 12, text="GARIS UFUK (0°)", fill="#00E676", font=("Consolas", 12, "bold"), anchor="w", tags="dyn")
06883 |
06884 |             start_az = int(center_az - (width/2 / px_per_deg_x)) - 1
06885 |             end_az = int(center_az + (width/2 / px_per_deg_x)) + 1

```

```

06886 |
06887 |         for az_mark in range(start_az, end_az):
06888 |             mx = get_zoomed_x(az_mark)
06889 |             real_az = az_mark % 360
06890 |             if 0 <= mx <= width:
06891 |                 tick_h = 5
06892 |                 if az_mark % 5 == 0: tick_h = 10
06893 |                 if az_mark % 10 == 0: tick_h = 15
06894 |                 self.anim_canvas.create_line(mx, horizon_y, mx, horizon_y + tick_h, fill="white", width=1.5 if tick_h > 5 else 1.0, tags="dyn")
06895 |                 if az_mark % 10 == 0:
06896 |                     self.anim_canvas.create_text(mx, horizon_y + 28, text=f"{real_az}°", fill="white", font=("Consolas", 10), tags="dyn")
06897 |                 if real_az % 45 == 0:
06898 |                     arah_dict = {0: "U", 45: "TL", 90: "T", 135: "TG", 180: "S", 225: "BD", 270: "B", 315: "BL"}
06899 |                     dir_str = arah_dict.get(real_az, "")
06900 |                     if dir_str:
06901 |                         self.anim_canvas.create_text(mx, horizon_y - 20, text=f"▲ {dir_str}", fill="#B0BEC5", font=("Segoe UI", 14, "bold"), tags="dyn")
06902 |
06903 |             self.anim_canvas.create_line(x_sun, horizon_y, x_sun, horizon_y + 45, fill="#FFD54F", width=2, tags="dyn")
06904 |             self.anim_canvas.create_text(x_sun, horizon_y + 55, text=f"AZIMUT MTHRI\n{az_s_deg:.2f}°", fill="#FFD54F", font=("Consolas", 11, "bold"), justify="center", anchor="n", tags="dyn")
06905 |
06906 |             y_offset_moon = 45
06907 |             if abs(x_sun - x_moon) < 110:
06908 |                 y_offset_moon = 90
06909 |                 self.anim_canvas.create_line(x_moon, horizon_y, x_moon, horizon_y + y_offset_moon, fill="#00E5FF", width=2, dash=(4,2), tags="dyn")
06910 |             else:
06911 |                 self.anim_canvas.create_line(x_moon, horizon_y, x_moon, horizon_y + y_offset_moon, fill="#00E5FF", width=2, tags="dyn")
06912 |
06913 |             self.anim_canvas.create_text(x_moon, horizon_y + y_offset_moon + 10, text=f"AZIMUT HILAL\n{az_m_deg:.2f}°", fill="#00E5FF", font=("Consolas", 11, "bold"), justify="center", anchor="n",
tags="dyn")
06914 |
06915 |         else:
06916 |             # ---- 2. MODE NORMAL (TAMPILAN PENUH +90° HINGGA -90°) ----
06917 |             horizon_y = height / 2.0
06918 |             px_per_deg_y = horizon_y / 90.0
06919 |             px_per_deg_x = usable_width / 360.0
06920 |
06921 |             r_sun = max(36, int(3.6 * px_per_deg_y))
06922 |             r_moon = max(22, int(2.2 * px_per_deg_y))
06923 |
06924 |             x_sun = pad_x + (az_s_deg * px_per_deg_x)
06925 |             x_moon = pad_x + (az_m_deg * px_per_deg_x)
06926 |
06927 |             # Kalkulasi Posisi Y Awal
06928 |             y_sun = horizon_y - (alt_s_deg * px_per_deg_y)
06929 |             y_moon = horizon_y - (alt_m_deg * px_per_deg_y)
06930 |
06931 |             offset_visual_y = r_sun - (0.833 * px_per_deg_y)
06932 |             y_sun += offset_visual_y
06933 |             y_moon += offset_visual_y
06934 |
06935 |             # --- RENDER DATABASE RASI BINTANG (CONSTELLATIONS) ---
06936 |             from skyfield.api import Star
06937 |             CONSTELLATIONS = {
06938 |                 "ORION (Si Pemburu)": [
06939 |                     {"name": "Betelgeuse", "ra": (5, 55, 10.3), "dec": (7, 24, 25), "mag": 0.50},
06940 |                     {"name": "Bellatrix", "ra": (5, 25, 7.9), "dec": (6, 20, 59), "mag": 1.64},
06941 |                     {"name": "Alnilam", "ra": (5, 36, 12.8), "dec": (-1, 12, 6.9), "mag": 1.69},
06942 |                     {"name": "Rigel", "ra": (5, 14, 32.3), "dec": (-8, 12, 6), "mag": 0.18}
06943 |                 ],
06944 |                 "URSA MAJOR (Beruang Besar)": [
06945 |                     {"name": "Alkaid", "ra": (13, 47, 32.4), "dec": (49, 18, 48), "mag": 1.85},
06946 |                     {"name": "Mizar", "ra": (13, 23, 55.5), "dec": (54, 55, 31), "mag": 2.23},
06947 |                     {"name": "Alioth", "ra": (12, 54, 1.7), "dec": (55, 57, 35), "mag": 1.76},
06948 |                     {"name": "Dubhe", "ra": (11, 3, 43.7), "dec": (61, 45, 4), "mag": 1.81}
06949 |                 ],
06950 |                 "CRUX (Salib Selatan)": [
06951 |                     {"name": "Gacrux", "ra": (12, 31, 9.9), "dec": (-57, 6, 47), "mag": 1.59},
06952 |                     {"name": "Mimosa", "ra": (12, 47, 43.3), "dec": (-59, 41, 19), "mag": 1.25},
06953 |                     {"name": "Acrux", "ra": (12, 26, 35.9), "dec": (-63, 5, 57), "mag": 0.77}
06954 |                 ],
06955 |                 "SCORPIUS (Kalajengking)": [

```

```

06956 |         {"name": "Dschubba", "ra": (16, 0, 20.0), "dec": (-22, 37, 18), "mag": 2.29},
06957 |         {"name": "Antares", "ra": (16, 29, 24.4), "dec": (-26, 25, 55), "mag": 1.06},
06958 |         {"name": "Shaula", "ra": (17, 33, 36.5), "dec": (-37, 6, 14), "mag": 1.62}
06959 |     ],
06960 |     "CANIS MAJOR (Anjing Besar)": [
06961 |         {"name": "Sirius", "ra": (6, 45, 8.9), "dec": (-16, 42, 58), "mag": -1.46},
06962 |         {"name": "Wezen", "ra": (7, 8, 23.5), "dec": (-26, 23, 35), "mag": 1.83},
06963 |         {"name": "Adhara", "ra": (6, 58, 37.5), "dec": (-28, 58, 19), "mag": 1.50}
06964 |     ],
06965 |     "TAURUS (Banteng)": [
06966 |         {"name": "Aldebaran", "ra": (4, 35, 55.2), "dec": (16, 30, 33), "mag": 0.87},
06967 |         {"name": "Elnath", "ra": (5, 26, 17.5), "dec": (28, 36, 27), "mag": 1.65}
06968 |     ],
06969 |     "LEO (Singa)": [
06970 |         {"name": "Regulus", "ra": (10, 8, 22.3), "dec": (11, 58, 2), "mag": 1.36},
06971 |         {"name": "Denebola", "ra": (11, 49, 3.3), "dec": (14, 34, 19), "mag": 2.14}
06972 |     ],
06973 |     "CENTAURUS": [
06974 |         {"name": "Rigil Kentaurus", "ra": (14, 39, 36.5), "dec": (-60, 50, 2), "mag": -0.27},
06975 |         {"name": "Hadar", "ra": (14, 3, 49.4), "dec": (-60, 22, 23), "mag": 0.61}
06976 |     ],
06977 |     "GEMINI (Si Kembar)": [
06978 |         {"name": "Castor", "ra": (7, 34, 36.0), "dec": (31, 53, 18), "mag": 1.58},
06979 |         {"name": "Pollux", "ra": (7, 45, 18.9), "dec": (28, 1, 34), "mag": 1.16}
06980 |     ],
06981 |     "CYGNUS (Angsa)": [
06982 |         {"name": "Deneb", "ra": (20, 41, 25.9), "dec": (45, 16, 49), "mag": 1.25},
06983 |         {"name": "Sadr", "ra": (20, 22, 13.7), "dec": (40, 15, 24), "mag": 2.23},
06984 |         {"name": "Albireo", "ra": (19, 30, 43.3), "dec": (27, 57, 34), "mag": 3.05}
06985 |     ],
06986 |     "LYRA (Sitar)": [
06987 |         {"name": "Vega", "ra": (18, 36, 56.3), "dec": (38, 47, 1), "mag": 0.03}
06988 |     ],
06989 |     "AQUILA (Elang)": [
06990 |         {"name": "Altair", "ra": (19, 50, 47.0), "dec": (8, 52, 6), "mag": 0.77}
06991 |     ],
06992 |     "CARINA (Lunas Kapal)": [
06993 |         {"name": "Canopus", "ra": (6, 23, 57.1), "dec": (-52, 41, 45), "mag": -0.74},
06994 |         {"name": "Miaplacidus", "ra": (9, 13, 11.9), "dec": (-69, 43, 1), "mag": 1.67}
06995 |     ],
06996 |     "VIRGO (Gadis)": [
06997 |         {"name": "Spica", "ra": (13, 25, 11.6), "dec": (-11, 9, 41), "mag": 0.98}
06998 |     ],
06999 |     "AURIGA (Kusir)": [
07000 |         {"name": "Capella", "ra": (5, 16, 41.4), "dec": (45, 59, 53), "mag": 0.08}
07001 |     ],
07002 |     "BOÖTES (Penggembala)": [
07003 |         {"name": "Arcturus", "ra": (14, 15, 39.7), "dec": (19, 10, 57), "mag": -0.05}
07004 |     ],
07005 |     "PISCIS AUSTRINUS": [
07006 |         {"name": "Fomalhaut", "ra": (22, 57, 39.1), "dec": (-29, 37, 20), "mag": 1.16}
07007 |     ],
07008 |     "ERIDANUS (Sungai)": [
07009 |         {"name": "Achernar", "ra": (1, 37, 42.8), "dec": (-57, 14, 12), "mag": 0.46}
07010 |     ]
07011 | }
07012 |
07013 | for const_name, stars_data in CONSTELLATIONS.items():
07014 |     points = []
07015 |     for s_data in stars_data:
07016 |         bintang = Star(ra_hours=s_data['ra'], dec_degrees=s_data['dec'])
07017 |         astrometric_star = observer.at(t_sim).observe(bintang).apparent()
07018 |         alt_s, az_s, _ = astrometric_star.altaz()
07019 |         s_alt_deg = alt_s.degrees
07020 |         s_az_deg = az_s.degrees
07021 |
07022 |         if getattr(self, 'is_anim_zoomed', False):
07023 |             x_star = get_zoomed_x(s_az_deg)
07024 |             y_star = horizon_y - (s_alt_deg * px_per_deg_y) + offset_visual_y
07025 |         else:
07026 |             x_star = pad_x + (s_az_deg * px_per_deg_x)

```



```

07027 |         y_star = horizon_y - (s_alt_deg * px_per_deg_y) + offset_visual_y
07028 |
07029 |         r_star = max(1, int(5 - s_data['mag']))
07030 |
07031 |         if 0 <= x_star <= width and 0 <= y_star <= height:
07032 |             points.append((x_star, y_star))
07033 |             self.anim_canvas.create_oval(x_star-r_star, y_star-r_star, x_star+r_star, y_star+r_star,
07034 |                                         fill="#FFFFFF", outline="#B0BEC5", tags="dyn")
07035 |             # Tulis Nama Bintang
07036 |             if s_data['mag'] < 2.0 or getattr(self, 'is_anim_zoomed', False):
07037 |                 self.anim_canvas.create_text(x_star, y_star + r_star + 6, text=s_data['name'],
07038 |                                             fill="#9E9E9E", font=("Segoe UI", 9), tags="dyn")
07039 |
07040 |         # Gambar Garis Rasi Bintang & Label jika lebih dari 1 bintang terlihat (khusus mode Normal)
07041 |         if len(points) > 1 and not getattr(self, 'is_anim_zoomed', False):
07042 |             for i in range(len(points) - 1):
07043 |                 # Mencegah garis melintang dari ujung kiri ke kanan (Azimuth Wrap Around 360 -> 0)
07044 |                 if abs(points[i][0] - points[i+1][0]) < (width / 2):
07045 |                     self.anim_canvas.create_line(points[i][0], points[i][1], points[i+1][0], points[i+1][1],
07046 |                                                 fill="#4A6572", width=1.5, dash=(2, 2), tags="dyn")
07047 |
07048 |             self.anim_canvas.create_text(points[0][0], points[0][1] - 18, text=const_name,
07049 |                                         fill="#FFD54F", font=("Segoe UI", 11, "bold"), tags="dyn")
07050 |
07051 |         if not hasattr(self, '_live_images'):
07052 |             self._live_images = {}
07053 |
07054 |         # ---> TAMBAHAN: MENGGAMBAR SEMUA PLANET <---
07055 |         for p in planet_positions:
07056 |             if getattr(self, 'is_anim_zoomed', False):
07057 |                 d_az = p['az'] - center_az
07058 |                 if d_az > 180: d_az -= 360
07059 |                 elif d_az < -180: d_az += 360
07060 |                 p_x = (width / 2.0) + (d_az * px_per_deg_x)
07061 |             else:
07062 |                 p_x = pad_x + (p['az'] * px_per_deg_x)
07063 |
07064 |             p_y = horizon_y - (p['alt'] * px_per_deg_y) + offset_visual_y
07065 |
07066 |             if 0 <= p_x <= width and 0 <= p_y <= height:
07067 |                 r_p = 5 if getattr(self, 'is_anim_zoomed', False) else 3
07068 |                 self.anim_canvas.create_oval(p_x-r_p, p_y-r_p, p_x+r_p, p_y+r_p, fill=p['color'], outline="white", tags="dyn")
07069 |                 self.anim_canvas.create_text(p_x, p_y+r_p+8, text=p['name'], fill=p['color'], font=("Consolas", 10 if getattr(self, 'is_anim_zoomed', False) else 8, "bold"), tags="dyn")
07070 |             # -----
07071 |
07072 |         # --- GAMBAR GARIS ELONGASI (Tengah Matahari & Bulan) ---
07073 |         self.anim_canvas.create_line(x_sun, y_sun, x_moon, y_moon, fill="#00E676", dash=(4, 2), width=2, tags="dyn")
07074 |         mid_x = (x_sun + x_moon) / 2
07075 |         mid_y = (y_sun + y_moon) / 2
07076 |
07077 |         if getattr(self, 'is_anim_zoomed', False):
07078 |             self.anim_canvas.create_text(mid_x+1, mid_y - 14, text=f"Eln: {elong_geo:.2f}°", fill="black", font=("Consolas", 15, "bold"), tags="dyn")
07079 |             self.anim_canvas.create_text(mid_x, mid_y - 15, text=f"Eln: {elong_geo:.2f}°", fill="#00E676", font=("Consolas", 15, "bold"), tags="dyn")
07080 |         else:
07081 |             self.anim_canvas.create_text(mid_x, mid_y - 12, text=f"Eln: {elong_geo:.1f}°", fill="#00E676", font=("Consolas", 10, "bold"), tags="dyn")
07082 |
07083 |         # --- RENDER MATAHARI ---
07084 |         try:
07085 |             sun_path = os.path.join(BASE_DIR, "matahari.png")
07086 |             sun_img = Image.open(sun_path).convert("RGBA")
07087 |             sun_img = sun_img.resize((r_sun*2, r_sun*2), Image.Resampling.LANCZOS)
07088 |             self._live_images['sun'] = ImageTk.PhotoImage(sun_img)
07089 |             self.anim_canvas.create_image(x_sun, y_sun, image=self._live_images['sun'], tags=("dyn", "sun_obj"))
07090 |         except Exception:
07091 |             self.anim_canvas.create_oval(x_sun-r_sun-4, y_sun-r_sun-4, x_sun+r_sun+4, y_sun+r_sun+4, fill="#FF6D00", outline="", stipple="gray25", tags=("dyn", "sun_obj"))
07092 |             self.anim_canvas.create_oval(x_sun-r_sun, y_sun-r_sun, x_sun+r_sun, y_sun+r_sun, fill="#FFD54F", outline="#FFF59D", width=2, tags=("dyn", "sun_obj"))
07093 |
07094 |         sun_text = f"Matahari\nAlt(Topo): {alt_s_deg:.2f}°\nAlt(Geo) : {alt_sun_geo:.2f}°\nAzimuth : {az_s_deg:.1f}°"
07095 |         if not getattr(self, 'is_anim_zoomed', False):
07096 |             if x_sun > width - 120:
07097 |                 self.anim_canvas.create_text(x_sun - r_sun - 15, y_sun, text=sun_text, fill="#FFD54F", font=("Consolas", 10, "bold"), justify="right", anchor="e", tags="dyn")

```

```

07098 |         else:
07099 |             self.anim_canvas.create_text(x_sun + r_sun + 15, y_sun, text=sun_text, fill="#FFD54F", font=("Consolas", 10, "bold"), justify="left", anchor="w", tags="dyn")
07100 |
07101 | # --- RENDER BULAN & FASE MASK (DENGAN ROTASI KEMIRINGAN) ---
07102 | try:
07103 |     moon_filename = f"m{moon_idx:03d}.png"
07104 |     moon_path = os.path.join(BASE_DIR, "moon", moon_filename)
07105 |     moon_img = Image.open(moon_path).convert("RGBA")
07106 |     target_size = int(r_moon * 2)
07107 |     moon_img = moon_img.resize((target_size, target_size), Image.Resampling.LANCZOS)
07108 |
07109 |     dx = x_sun - x_moon
07110 |     dy = y_sun - y_moon
07111 |     sudut_kemiringan = math.degrees(math.atan2(dy, dx))
07112 |
07113 |     if moon_idx >= 180:
07114 |         sudut_kemiringan -= 180
07115 |
07116 |     moon_img = moon_img.rotate(-sudut_kemiringan, resample=Image.Resampling.BICUBIC)
07117 |
07118 |     mask = Image.new('L', (target_size, target_size), 0)
07119 |     draw = ImageDraw.Draw(mask)
07120 |     draw.ellipse((0, 0, target_size - 1, target_size - 1), fill=255)
07121 |     moon_img.putalpha(mask)
07122 |
07123 |     self._live_images['moon'] = ImageTk.PhotoImage(moon_img)
07124 |     self.anim_canvas.create_image(x_moon, y_moon, image=self._live_images['moon'], tags=("dyn", "moon_obj"))
07125 | except Exception:
07126 |     self.anim_canvas.create_oval(x_moon-r_moon-4, y_moon-r_moon-4, x_moon+r_moon+4, y_moon+r_moon+4, fill="#80DEEA", outline="", stipple="gray12", tags=("dyn", "moon_obj"))
07127 |     self.anim_canvas.create_oval(x_moon-r_moon, y_moon-r_moon, x_moon+r_moon, y_moon+r_moon, fill="#E0E0E0", outline="white", width=2, tags=("dyn", "moon_obj"))
07128 |
07129 | moon_text = f"Bulan\nAlt(Topo): {alt_m_deg:.2f}°\nAlt(Geo) : {alt_moon_geo:.2f}°\nAzimuth : {az_m_deg:.1f}°\nFase: {moon_idx}°"
07130 | if getattr(self, 'is_anim_zoomed', False):
07131 |     self.anim_canvas.create_text(x_moon + 16, (y_moon + horizon_y) / 2 + 1, text=f"Alt: {alt_moon_geo:.2f}°", fill="black", font=("Consolas", 15, "bold"), anchor="w", tags="dyn")
07132 |     self.anim_canvas.create_text(x_moon + 15, (y_moon + horizon_y) / 2, text=f"Alt: {alt_moon_geo:.2f}°", fill="#FFD54F", font=("Consolas", 15, "bold"), anchor="w", tags="dyn")
07133 | else:
07134 |     if x_moon > width - 120:
07135 |         self.anim_canvas.create_text(x_moon - r_moon - 15, y_moon, text=moon_text, fill="#00E5FF", font=("Consolas", 10, "bold"), justify="right", anchor="e", tags="dyn")
07136 |     else:
07137 |         self.anim_canvas.create_text(x_moon + r_moon + 15, y_moon, text=moon_text, fill="#00E5FF", font=("Consolas", 10, "bold"), justify="left", anchor="w", tags="dyn")
07138 |
07139 | # --- GARIS PROYEKSI KE UFUK MATAHARI (HANYA NORMAL MODE) ---
07140 | if not getattr(self, 'is_anim_zoomed', False):
07141 |     if (y_sun + r_sun) < horizon_y:
07142 |         self.anim_canvas.create_line(x_sun, y_sun + r_sun, x_sun, horizon_y, fill="#FFD54F", dash=(2, 2), tags="dyn")
07143 |     elif (y_sun - r_sun) > horizon_y:
07144 |         self.anim_canvas.create_line(x_sun, y_sun - r_sun, x_sun, horizon_y, fill="#FFD54F", dash=(2, 2), tags="dyn")
07145 |
07146 | # --- GARIS PROYEKSI TINGGI HILAL KE UFUK ---
07147 | if (y_moon + r_moon) < horizon_y:
07148 |     self.anim_canvas.create_line(x_moon, y_moon + r_moon, x_moon, horizon_y, fill="#00E5FF", dash=(2, 2), tags="dyn")
07149 | elif (y_moon - r_moon) > horizon_y:
07150 |     self.anim_canvas.create_line(x_moon, y_moon - r_moon, x_moon, horizon_y, fill="#00E5FF", dash=(2, 2), tags="dyn")
07151 |
07152 | # --- KOTAK INFO STATUS TRACKING HILAL ---
07153 | is_sun_set_visually = alt_top_sun <= 0
07154 | is_moon_set_visually = alt_top_moon <= 0
07155 |
07156 | txt_fisik_sun = "Terbenam Total" if is_sun_set_visually else "Di Atas Ufuk"
07157 | txt_fisik_moon = "Terbenam Total" if is_moon_set_visually else "Di Atas Ufuk"
07158 |
07159 | arah_sun = get_compass_dir(az_s_deg)
07160 | arah_moon = get_compass_dir(az_m_deg)
07161 | beda_az = abs(az_m_deg - az_s_deg)
07162 |
07163 | delta_az_tilt = az_s_deg - az_m_deg
07164 | if delta_az_tilt > 180: delta_az_tilt -= 360
07165 | elif delta_az_tilt < -180: delta_az_tilt += 360
07166 |
07167 | delta_alt_tilt = alt_s_deg - alt_m_deg
07168 |

```

```

07169 |         tilt_rad = math.atan2(delta_az_tilt * math.cos(math.radians(alt_m_deg)), delta_alt_tilt)
07170 |         tilt_deg = math.degrees(tilt_rad)
07171 |
07172 |         jam_arah = round(tilt_deg / 30.0)
07173 |         if jam_arah <= 0: jam_arah += 12
07174 |         if jam_arah > 12: jam_arah %= 12
07175 |         if jam_arah == 0: jam_arah = 12
07176 |
07177 |         jam_cusp = (jam_arah + 6) % 12
07178 |         if jam_cusp == 0: jam_cusp = 12
07179 |
07180 |         tilt_info = f"Perut Sabit Jam {jam_arah} | Ujung Cusp Jam {jam_cusp}"
07181 |
07182 |         illum = almanac.fraction_illuminated(self.eph, 'moon', t_sim)
07183 |         illum_val = (illum.item() if hasattr(illum, 'item') else illum) * 100.0
07184 |         evs = self._get_anim_daily_events(dt_lokal.year, dt_lokal.month, dt_lokal.day, tz_offset, lat, lon)
07185 |
07186 |         self.latest_anim_info_text = (
07187 |             f"TRACKING HILAL, MATAHARI & PLANET (REAL-TIME DATA)\n{waktu_str}\n"
07188 |             f"Umur Bulan      : {moon_age:.2f} hari\n"
07189 |             f"Fase/Illuminasi : {illum_val:.1f}%\n"
07190 |             f"Elongasi (Geo) : {elong_geo:.2f}° | Beda Azimuth : {beda_az:.2f}°\n"
07191 |             f"Kemiringan      : Zenit {tilt_deg:+.1f}° ({tilt_info})\n"
07192 |             f"-----\n"
07193 |             f"MATAHARI : {txt_fisik_sun:<15} | Azimut: {az_s_deg:06.2f}° ({arah_sun})\n"
07194 |             f"          Terbit: {evs['sun_rise']} | Terbenam: {evs['sun_set']}\n"
07195 |             f"HILAL      : {txt_fisik_moon:<15} | Azimut: {az_m_deg:06.2f}° ({arah_moon})\n"
07196 |             f"          Terbit: {evs['moon_rise']} | Terbenam: {evs['moon_set']}"
07197 |         )
07198 |
07199 |         self.draw_anim_info_box_now()
07200 |
07201 |         elong_topo = astro_sun.separation_from(astro_moon).degrees
07202 |
07203 |         if not getattr(self, 'anim_is_live', True):
07204 |             if alt_moon_geo >= 5.0 and elong_geo >= 8.0:
07205 |                 box_w_khgt, box_h_khgt = 380, 40
07206 |                 y1_khgt = height - 55
07207 |                 img_khgt = Image.new("RGBA", (box_w_khgt, box_h_khgt), (26, 26, 26, 160))
07208 |                 self._live_images['khgt_bg'] = ImageTk.PhotoImage(img_khgt)
07209 |                 self.anim_canvas.create_image(15, y1_khgt, image=self._live_images['khgt_bg'], anchor="nw", tags="dyn")
07210 |                 self.anim_canvas.create_rectangle(15, y1_khgt, 15 + box_w_khgt, y1_khgt + box_h_khgt, fill="", outline="#00E676", width=1.5, tags="dyn")
07211 |                 self.anim_canvas.create_text(25, y1_khgt + 20, text="Memenuhi kriteria KHGT", fill="#00E676", font=("Consolas", 20, "bold"), anchor="w", tags="dyn")
07212 |
07213 |             if alt_m_deg >= 3.0 and elong_topo >= 6.4:
07214 |                 box_w_mabims, box_h_mabims = 450, 40
07215 |                 x1_mabims = width - box_w_mabims - 15
07216 |                 y1_mabims = height - 55
07217 |                 img_mab = Image.new("RGBA", (box_w_mabims, box_h_mabims), (26, 26, 26, 160))
07218 |                 self._live_images['mabims_bg'] = ImageTk.PhotoImage(img_mab)
07219 |                 self.anim_canvas.create_image(x1_mabims, y1_mabims, image=self._live_images['mabims_bg'], anchor="nw", tags="dyn")
07220 |                 self.anim_canvas.create_rectangle(x1_mabims, y1_mabims, x1_mabims + box_w_mabims, y1_mabims + box_h_mabims, fill="", outline="#00E5FF", width=1.5, tags="dyn")
07221 |                 self.anim_canvas.create_text(x1_mabims + 20, y1_mabims + 20, text="Memenuhi kriteria Neo Mabims", fill="#00E5FF", font=("Consolas", 20, "bold"), anchor="w", tags="dyn")
07222 |
07223 |         except Exception as e:
07224 |             self.anim_canvas.create_text(width/2, height/2, text=f"Sedang memuat data ephemeris... ({e})", fill="red", font=("Consolas", 12), tags="dyn")
07225 |
07226 |         self.after(1000, self.update_animation)
07227 |
07228 |     def toggle_anim_info(self, state):
07229 |         """Menyembunyikan/menampilkan kotak info secara dinamis"""
07230 |         self.show_anim_info_box = state
07231 |         self.draw_anim_info_box_now()
07232 |
07233 |     def draw_anim_info_box_now(self):
07234 |         """Mencetak UI Kotak Info di Kanvas"""
07235 |         self.anim_canvas.delete("dyn_info")
07236 |         if getattr(self, 'show_anim_info_box', True):
07237 |             if hasattr(self, 'latest_anim_info_text'):
07238 |                 txt_id = self.anim_canvas.create_text(25, 25, text=self.latest_anim_info_text, fill="white", font=("Consolas", 12, "bold"), anchor="nw", tags=("dyn", "dyn_info"))
07239 |                 bbox = self.anim_canvas.bbox(txt_id)

```

```

07240 |         if bbox:
07241 |             # Latar Belakang Kotak
07242 |             bg = self.anim_canvas.create_rectangle(bbox[0]-10, bbox[1]-10, bbox[2]+30, bbox[3]+10, fill="#121212", outline="#00E5FF", width=1.5, tags=("dyn", "dyn_info", "info_bg"))
07243 |             self.anim_canvas.tag_lower(bg, txt_id)
07244 |
07245 |             # Tombol Close (X)
07246 |             btn_x, btn_y = bbox[2] + 5, bbox[1] - 5
07247 |             self.anim_canvas.create_rectangle(btn_x, btn_y, btn_x+20, btn_y+20, fill="#FF5252", outline="white", tags=("dyn", "dyn_info", "btn_close"))
07248 |             self.anim_canvas.create_text(btn_x+10, btn_y+10, text="X", fill="white", font=("Consolas", 10, "bold"), tags=("dyn", "dyn_info", "btn_close"))
07249 |         else:
07250 |             self.anim_canvas.create_text(25, 25, text="[] [ Panel Info Disembunyikan - Klik Gambar Matahari/Bulan untuk Membuka Kembali ]", fill="#00E5FF", font=("Consolas", 11, "italic"), anchor="nw",
tags=("dyn", "dyn_info"))
07251 |
07252 |     def _get_anim_daily_events(self, year, month, day, tz, lat, lon):
07253 |         """Helper cerdas untuk caching jadwal Rise & Set harian agar animasi tidak lag"""
07254 |         date_key = f"{year}-{month}-{day}-{lat}-{lon}"
07255 |         if getattr(self, 'anim_cached_date', None) == date_key:
07256 |             return getattr(self, 'anim_cached_events', {})
07257 |
07258 |         try:
07259 |             t0 = self.ts.utc(year, month, day, -int(tz))
07260 |             t1 = self.ts.utc(year, month, day, 24 - int(tz))
07261 |             loc = wgs84.latlon(lat, lon)
07262 |
07263 |             # Waktu Matahari
07264 |             f_s = almanac.sunrise_sunset(self.eph, loc)
07265 |             t_s, y_s = almanac.find_discrete(t0, t1, f_s)
07266 |             sun_r, sun_s = "--:--", "--:--"
07267 |             if t_s is not None:
07268 |                 for t_val, y_val in zip(np.atleast_1d(t_s.tt), np.atleast_1d(y_s)):
07269 |                     dt_loc = self.ts.tt_jd(t_val + tz/24.0).utc_datetime()
07270 |                     if y_val == 1: sun_r = dt_loc.strftime("%H:%M")
07271 |                     else: sun_s = dt_loc.strftime("%H:%M")
07272 |
07273 |             # Waktu Bulan
07274 |             f_m = almanac.risings_and_settings(self.eph, self.eph['moon'], loc)
07275 |             t_m, y_m = almanac.find_discrete(t0, t1, f_m)
07276 |             moon_r, moon_s = "--:--", "--:--"
07277 |             if t_m is not None:
07278 |                 for t_val, y_val in zip(np.atleast_1d(t_m.tt), np.atleast_1d(y_m)):
07279 |                     dt_loc = self.ts.tt_jd(t_val + tz/24.0).utc_datetime()
07280 |                     if y_val == 1: moon_r = dt_loc.strftime("%H:%M")
07281 |                     else: moon_s = dt_loc.strftime("%H:%M")
07282 |         except Exception:
07283 |             sun_r, sun_s, moon_r, moon_s = "Err", "Err", "Err", "Err"
07284 |
07285 |         self.anim_cached_date = date_key
07286 |         self.anim_cached_events = {'sun_rise': sun_r, 'sun_set': sun_s, 'moon_rise': moon_r, 'moon_set': moon_s}
07287 |         return self.anim_cached_events
07288 |
07289 |
07290 |     # (Fungsi toggle_anim_zoom dan draw_static_background biarkan tetap seperti sebelumnya)
07291 |
07292 |
07293 |
07294 |     def toggle_anim_zoom(self):
07295 |         """Fungsi untuk beralih antara Mode Normal dan Mode Zoom Hilal"""
07296 |         self.is_anim_zoomed = not getattr(self, 'is_anim_zoomed', False)
07297 |         if self.is_anim_zoomed:
07298 |             self.btn_anim_zoom.configure(text="[] Normal View", fg_color="#D32F2F", hover_color="#B71C1C")
07299 |             self.anim_canvas.delete("static") # Bersihkan bintang/kompas agar fokus
07300 |         else:
07301 |             self.btn_anim_zoom.configure(text="[] Zoom Hilal", fg_color="#673AB7", hover_color="#512DA8")
07302 |             self.draw_static_background() # Gambar ulang bintang dan kompas
07303 |
07304 |         # Paksa update frame seketika
07305 |         self.update_animation()
07306 |
07307 |
07308 |
07309 |     def anim_cari_sunset(self):

```

```

07310 |         self.anim_is_live = False
07311 |     try:
07312 |         # 1. Ambil data input tanggal
07313 |         y = int(self.entry_anim_year.get())
07314 |         m = int(self.entry_anim_month.get())
07315 |         d = int(self.entry_anim_day.get())
07316 |
07317 |         # 2. Ambil Lokasi
07318 |         try:
07319 |             lat_val = float(self.entry_vlat.get())
07320 |             lon_val = float(self.entry_vlon.get())
07321 |             elev_val = float(self.entry_velev.get())
07322 |         except Exception:
07323 |             lat_val, lon_val, elev_val = -7.0667, 110.4100, 230.0
07324 |
07325 |         tz_offset = int(self.get_tz_from_lon(lon_val))
07326 |
07327 |         # 3. MENGGUNAKAN PYEPHEM MURNI (Bukan Skyfield)
07328 |         # PyEphem memproses waktu secara skalar, bebas dari error "single Time / array"
07329 |         import ephem
07330 |         pengamat = ephem.Observer()
07331 |         pengamat.lat = str(lat_val)
07332 |         pengamat.lon = str(lon_val)
07333 |         pengamat.elevation = elev_val
07334 |         pengamat.pressure = 1010 # Standar atmosfer
07335 |         pengamat.temp = 25 # Standar suhu
07336 |         matahari = ephem.Sun()
07337 |
07338 |         # Mulai pencarian dari jam 12:00 siang Waktu Lokal (diubah ke UTC)
07339 |         waktu_mulai_cari = datetime.datetime(y, m, d, 12, 0, 0) - datetime.timedelta(hours=tz_offset)
07340 |         pengamat.date = ephem.Date(waktu_mulai_cari)
07341 |
07342 |         # Cari waktu sunset (Terbenam)
07343 |         try:
07344 |             # Menghasilkan datetime UTC murni
07345 |             waktu_sunset_utc = pengamat.next_setting(matahari).datetime()
07346 |
07347 |             # Konversi ke waktu lokal untuk ditampilkan di form GUI
07348 |             dt_lokal = waktu_sunset_utc + datetime.timedelta(hours=tz_offset)
07349 |
07350 |             # Konversi ke format Skyfield Time HANYA untuk kebutuhan Animasi
07351 |             self.anim_custom_time = self.ts.from_datetime(waktu_sunset_utc.replace(tzinfo=pytz.utc))
07352 |
07353 |             # 4. Update Angka di Form UI
07354 |             self.entry_anim_year.delete(0, 'end')
07355 |             self.entry_anim_year.insert(0, str(dt_lokal.year))
07356 |
07357 |             self.entry_anim_month.delete(0, 'end')
07358 |             self.entry_anim_month.insert(0, f"{dt_lokal.month:02d}")
07359 |
07360 |             self.entry_anim_day.delete(0, 'end')
07361 |             self.entry_anim_day.insert(0, f"{dt_lokal.day:02d}")
07362 |
07363 |             self.entry_anim_time.delete(0, 'end')
07364 |             self.entry_anim_time.insert(0, dt_lokal.strftime("%H:%M:%S"))
07365 |
07366 |         except (ephem.AlwaysUpError, ephem.NeverUpError):
07367 |             messagebox.showwarning("Peringatan", "Matahari tidak terbenam pada hari/lokasi tersebut (Anomali Kutub).")
07368 |             self.anim_start_live()
07369 |
07370 |     except Exception as e:
07371 |         import traceback
07372 |         # Memunculkan log lengkap jika gagal, agar mudah dilacak
07373 |         messagebox.showerror("Error", f"Gagal mencari waktu sunset:\n{e}\n\nDetail Error:\n{traceback.format_exc()}")
07374 |         self.anim_start_live()
07375 |
07376 |
07377 | # =====
07378 | # MODUL TAMBAHAN 13 & 14: 3D SPACE SYSTEM W/ ZOOM
07379 | # =====
07380 | def setup_3d_out_frame(self):

```

```

07381 |         self.frame_3d_out = ctk.CTkFrame(self.main_frame, fg_color="#020205")
07382 |
07383 |         header = ctk.CTkFrame(self.frame_3d_out, fg_color="#101018", corner_radius=8)
07384 |         header.pack(fill="x", padx=15, pady=10)
07385 |
07386 |         ctk.CTkLabel(header, text=" 3D GEOCENTRIC VIEW", font=("Montserrat", 16, "bold"), text_color="#00E5FF").pack(side="left", padx=20)
07387 |
07388 |         self.btn_peak_eclipse = ctk.CTkButton(header, text=" ECLIPSE 2026", width=140, fg_color="#D32F2F", hover_color="#B71C1C", command=self.set_eclipse_time_3d)
07389 |         self.btn_peak_eclipse.pack(side="right", padx=(10, 15), pady=10)
07390 |
07391 |         # --- TOMBOL ZOOM ---
07392 |
07393 |         self.anim_3d_canvas = tk.Canvas(self.frame_3d_out, bg='#010105', highlightthickness=0)
07394 |         self.anim_3d_canvas.pack(fill="both", expand=True, padx=15, pady=(0, 15))
07395 |
07396 |         # Inisialisasi Kamera & Skala
07397 |         self.cam_angle_x = 0.8
07398 |         self.cam_angle_y = 0.3
07399 |         self.cam_zoom = 1.0
07400 |
07401 |         # Binding Interaksi Mouse (Rotasi & Scroll Zoom Cross-Platform)
07402 |         self.anim_3d_canvas.bind("<B1-Motion>", self.rotate_3d_view)
07403 |         self.anim_3d_canvas.bind("<MouseWheel>", self.on_mouse_wheel_3d) # Windows/Mac
07404 |         self.anim_3d_canvas.bind("<Button-4>", self.on_mouse_wheel_3d) # Linux Scroll Up
07405 |         self.anim_3d_canvas.bind("<Button-5>", self.on_mouse_wheel_3d) # Linux Scroll Down
07406 |
07407 |     def zoom_in(self):
07408 |         if hasattr(self, 'cam_zoom'):
07409 |             self.cam_zoom *= 1.2
07410 |             if self.cam_zoom > 12.0: self.cam_zoom = 12.0 # Batas maksimal perbesaran
07411 |
07412 |     def zoom_out(self):
07413 |         if hasattr(self, 'cam_zoom'):
07414 |             self.cam_zoom *= 0.8
07415 |             if self.cam_zoom < 0.1: self.cam_zoom = 0.1 # Batas maksimal pengecilan
07416 |
07417 |     def on_mouse_wheel_3d(self, event):
07418 |         # Logika scroll menangkap pergerakan roda mouse dengan aman
07419 |         if str(event.type) == 'MouseWheel':
07420 |             if event.delta > 0: self.zoom_in()
07421 |             else: self.zoom_out()
07422 |         else:
07423 |             if getattr(event, 'num', 0) == 4: self.zoom_in()
07424 |             elif getattr(event, 'num', 0) == 5: self.zoom_out()
07425 |
07426 |     def get_shared_date_components(self):
07427 |         """Memastikan tanggal diambil dari Menu 16 jika sedang aktif (Anti Macet)"""
07428 |         try:
07429 |             mode_aktif = self.combo_mode.get()
07430 |             if "Sistem Sun Moon" in mode_aktif:
07431 |                 return int(self.combo_3d_y.get()), int(self.combo_3d_m.get()), int(self.combo_3d_d.get())
07432 |             else:
07433 |                 return int(self.entry_vyear.get()), int(self.entry_vmonth.get()), int(self.entry_vday.get())
07434 |         except:
07435 |             now = datetime.datetime.now()
07436 |             return now.year, now.month, now.day
07437 |
07438 |     def force_update_3d(self):
07439 |         """Mengeksekusi animasi saat tombol 'TERAPKAN TANGGAL' diklik"""
07440 |         try:
07441 |             y = int(self.combo_3d_y.get())
07442 |             m = int(self.combo_3d_m.get())
07443 |             d = int(self.combo_3d_d.get())
07444 |
07445 |             # Sinkronisasi ke form input general (Menu 1) agar sinkron ke seluruh modul
07446 |             try:
07447 |                 self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, str(y))
07448 |                 self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, f"{m:02d}")
07449 |                 self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, f"{d:02d}")
07450 |             except: pass
07451 |

```

```

07452 |         self.lbl_status.configure(text=f"Menerapkan Data Ephemeris {y}...", text_color="#FFAB40")
07453 |
07454 |         # Switch data satelit/ephemeris secara real-time
07455 |         self.auto_switch_ephemeris(y)
07456 |
07457 |         self.lbl_status.configure(text="Data 3D Berhasil Diterapkan", text_color="#00E676")
07458 |     except Exception as e:
07459 |         messagebox.showerror("Error", f"Gagal menerapkan tanggal:\n{e}")
07460 |
07461 | def set_eclipse_time_3d(self):
07462 |     """Tombol pintar untuk Gerhana 2026"""
07463 |     self.combo_3d_y.set("2026")
07464 |     self.combo_3d_m.set("03")
07465 |     self.combo_3d_d.set("03")
07466 |
07467 |     self.force_update_3d()
07468 |     messagebox.showinfo("Eclipse Mode", "Simulasi 3D sukses disetel ke posisi Gerhana Matahari 3 Maret 2026.")
07469 |
07470 | def project_3d_raw(self, x, y, z):
07471 |     # Murni melakukan rotasi sumbu 3D dan kalkulasi jarak kamera tanpa terpengaruh Zoom
07472 |     x1 = x * math.cos(self.cam_angle_x) - z * math.sin(self.cam_angle_x)
07473 |     z1 = x * math.sin(self.cam_angle_x) + z * math.cos(self.cam_angle_x)
07474 |     y2 = y * math.cos(self.cam_angle_y) - z1 * math.sin(self.cam_angle_y)
07475 |     z2 = y * math.sin(self.cam_angle_y) + z1 * math.cos(self.cam_angle_y)
07476 |
07477 |     camera_dist = 1000.0
07478 |     factor = camera_dist / (camera_dist + z2) if (camera_dist + z2) != 0 else 1
07479 |
07480 |     return x1 * factor, -y2 * factor, z2
07481 |
07482 |
07483 | def adjust_zoom(self, factor):
07484 |     self.cam_zoom *= factor
07485 |     # Membatasi skala zoom agar tidak terlalu jauh atau tembus ke dalam
07486 |     if self.cam_zoom < 0.1: self.cam_zoom = 0.1
07487 |     if self.cam_zoom > 8.0: self.cam_zoom = 8.0
07488 |
07489 | def on_mouse_wheel_3d(self, event):
07490 |     # Logika menangkap pergerakan roda mouse
07491 |     if event.num == 4 or getattr(event, 'delta', 0) > 0:
07492 |         self.adjust_zoom(1.1)
07493 |     elif event.num == 5 or getattr(event, 'delta', 0) < 0:
07494 |         self.adjust_zoom(0.9)
07495 |
07496 |
07497 | def rotate_3d_view(self, event):
07498 |     w = self.anim_3d_canvas.winfo_width()
07499 |     h = self.anim_3d_canvas.winfo_height()
07500 |     if w > 0 and h > 0:
07501 |         self.cam_angle_x = (event.x / w) * 2 * math.pi
07502 |         self.cam_angle_y = (event.y / h) * math.pi
07503 |
07504 | def project_3d(self, x, y, z, width, height):
07505 |     x1 = x * math.cos(self.cam_angle_x) - z * math.sin(self.cam_angle_x)
07506 |     z1 = x * math.sin(self.cam_angle_x) + z * math.cos(self.cam_angle_x)
07507 |     y2 = y * math.cos(self.cam_angle_y) - z1 * math.sin(self.cam_angle_y)
07508 |     z2 = y * math.sin(self.cam_angle_y) + z1 * math.cos(self.cam_angle_y)
07509 |
07510 |     factor = 500 / (500 + z2)
07511 |     px = x1 * factor + (width / 2)
07512 |     py = -y2 * factor + (height / 2)
07513 |     return px, py, z2
07514 |
07515 | def update_3d_animation(self):
07516 |     if not self.is_viewing_3d: return
07517 |
07518 |     canvas = self.anim_3d_canvas
07519 |     w = canvas.winfo_width()
07520 |     h = canvas.winfo_height()
07521 |
07522 |     if w <= 1:

```

```

07523 |         self.after(200, self.update_3d_animation)
07524 |         return
07525 |
07526 |     canvas.delete("all")
07527 |
07528 |     # Gambar background bintang (statis)
07529 |     random.seed(42)
07530 |     for _ in range(100):
07531 |         rx, ry = random.randint(0, w), random.randint(0, h)
07532 |         canvas.create_oval(rx, ry, rx+1, ry+1, fill="#555555")
07533 |
07534 |     try:
07535 |         y, m, d = self.get_shared_date_components()
07536 |         now = datetime.datetime.now()
07537 |         t_sim = self.ts.utc(y, m, d, now.hour, now.minute, now.second)
07538 |
07539 |         earth_obj = self.eph['earth']
07540 |
07541 |         # Daftar benda langit yang divisualisasikan
07542 |         celestial_bodies = [
07543 |             ('moon', "#ECEFF1", 15, 180, "BULAN"),
07544 |             ('mercury', "#B0BEC5", 10, 230, "MERKURIUS"),
07545 |             ('venus', "#FFCC80", 18, 290, "VENUS"),
07546 |             ('sun', "#FFD600", 50, 350, "MATAHARI"),
07547 |             ('mars', "#EF5350", 16, 450, "MARS"),
07548 |             ('jupiter', "#FFB74D", 35, 600, "JUPITER"),
07549 |             ('saturn', "#FFE082", 30, 750, "SATURNUS"),
07550 |             ('uranus', "#81D4FA", 25, 900, "URANUS"),
07551 |             ('neptune', "#5C6BC0", 24, 1050, "NEPTUNUS"),
07552 |             ('pluto', "#CFD8DC", 8, 1200, "PLUTO")
07553 |         ]
07554 |
07555 |         draw_list = []
07556 |
07557 |         # 1. Menggambar Bumi di Pusat (Geocentric)
07558 |         p_earth = self.project_3d(0, 0, 0, w, h)
07559 |         draw_list.append(('earth', p_earth, "#2196F3", 30 * self.cam_zoom, "BUMI"))
07560 |
07561 |         # 2. Menggambar Umbra (Bayangan Gelap Bumi)
07562 |         try:
07563 |             pos_sun = earth_obj.at(t_sim).observe(self.eph['sun']).position.km
07564 |             dist_s = np.linalg.norm(pos_sun)
07565 |             # Jarak visual umbra diskalakan dengan zoom
07566 |             ux, uy, uz = -(pos_sun / dist_s) * (180 * self.cam_zoom)
07567 |             p_umbra = self.project_3d(ux, uy, uz, w, h)
07568 |             draw_list.append(('umbra', p_umbra, "#1A1A1A", 22 * self.cam_zoom, ""))
07569 |         except:
07570 |             pass
07571 |
07572 |         # 3. Kalkulasi dan Skala Zoom untuk Semua Objek
07573 |         for name, color, size, vis_dist, label in celestial_bodies:
07574 |             try:
07575 |                 target_key = f'{name} barycenter' if name not in ['sun', 'moon'] else name
07576 |                 try:
07577 |                     target_obj = self.eph[target_key]
07578 |                 except KeyError:
07579 |                     target_obj = self.eph[name]
07580 |
07581 |                 pos_km = earth_obj.at(t_sim).observe(target_obj).position.km
07582 |                 dist_km = np.linalg.norm(pos_km)
07583 |
07584 |                 # Kalikan jarak visual dengan faktor zoom kamera saat ini
07585 |                 px, py, pz = (pos_km / dist_km) * (vis_dist * self.cam_zoom)
07586 |
07587 |                 proj_p = self.project_3d(px, py, pz, w, h)
07588 |                 # Kalikan juga ukuran objek (size) dengan zoom
07589 |                 draw_list.append((name, proj_p, color, size * self.cam_zoom, label))
07590 |
07591 |             except Exception as e:
07592 |                 pass
07593 |

```



```

07594 |         # Algoritma Z-Buffer: Urutkan objek dari yang terjauh ke yang terdekat dari pandangan kamera
07595 |         draw_list.sort(key=lambda x: x[1][2], reverse=True)
07596 |
07597 |         # 4. Merender objek ke Kanvas
07598 |         for tag, p, color, size, label in draw_list:
07599 |             if tag == 'umbra':
07600 |                 canvas.create_oval(p[0]-size, p[1]-size, p[0]+size, p[1]+size, fill="", outline="#333", dash=(4,4))
07601 |             else:
07602 |                 canvas.create_oval(p[0]-size, p[1]-size, p[0]+size, p[1]+size, fill=color, outline="white" if tag=='earth' else "")
07603 |                 # Tampilkan Label Nama Planet (Sembunyikan jika di zoom out terlalu jauh agar tidak numpuk)
07604 |                 if label and size > 4:
07605 |                     font_size = max(7, int(9 * min(self.cam_zoom, 1.5)))
07606 |                     canvas.create_text(p[0], p[1]+size+12, text=label, fill="white", font=("Consolas", font_size, "bold"))
07607 |
07608 |         # Menggambar Garis Orbit Panduan (Ikut ter-zoom)
07609 |         r_moon = 180 * self.cam_zoom
07610 |         r_sun = 350 * self.cam_zoom
07611 |         r_out = 600 * self.cam_zoom
07612 |         canvas.create_oval(w/2-r_moon, h/2-r_moon, w/2+r_moon, h/2+r_moon, outline="#222")
07613 |         canvas.create_oval(w/2-r_sun, h/2-r_sun, w/2+r_sun, h/2+r_sun, outline="#222")
07614 |         canvas.create_oval(w/2-r_out, h/2-r_out, w/2+r_out, h/2+r_out, outline="#111")
07615 |
07616 |         info = f"3D GEOCENTRIC SOLAR SYSTEM (LIVE)\n" \
07617 |               f"Date: {y}-{m:02d}-{d:02d}\n" \
07618 |               f"Zoom Level: {self.cam_zoom:.1f}x\n" \
07619 |               f"* Klik Kiri & Geser Mouse untuk Rotasi\n" \
07620 |               f"* Scroll Mouse / Tombol UI untuk Zoom"
07621 |
07622 |         canvas.create_rectangle(20, 20, 360, 110, fill="#050510", outline="#00E5FF", width=1)
07623 |         canvas.create_text(35, 35, text=info, fill="#00E5FF", font=("Consolas", 10), anchor="nw")
07624 |
07625 |     except Exception as e:
07626 |         canvas.create_text(w/2, h/2, text=f"System Updating Data... {e}", fill="gray")
07627 |
07628 |     self.after(50, self.update_3d_animation)
07629 |
07630 | def set_eclipse_time_3d(self):
07631 |     self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, "2026")
07632 |     self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, "03")
07633 |     self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, "03")
07634 |     messagebox.showinfo("Eclipse Mode", "Waktu diset ke 3 Maret 2026.")
07635 |
07636 |
07637 | # =====
07638 | # FUNGSI KONVERSI KHGT
07639 | # =====
07640 | def get_new_moons_in_range(self, start_tt_float, end_tt_float):
07641 |     t0 = self.ts.tt_jd(start_tt_float)
07642 |     t1 = self.ts.tt_jd(end_tt_float)
07643 |     t_obj, y_obj = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
07644 |     if t_obj is None: return []
07645 |     if getattr(t_obj, 'shape', ()) == ():
07646 |         tt_list = [float(t_obj.tt)]
07647 |         y_list = [int(y_obj)]
07648 |     else:
07649 |         tt_list = t_obj.tt.tolist()
07650 |         y_list = y_obj.tolist()
07651 |         new_moons_tt = []
07652 |         for i in range(len(y_list)):
07653 |             if y_list[i] == 0:
07654 |                 new_moons_tt.append(tt_list[i])
07655 |         return new_moons_tt
07656 |
07657 | def get_hijri_month_from_tt(self, tt_float):
07658 |     delta_months = round((tt_float - 2451550.0) / 29.53058853)
07659 |     abs_month = 17038 + delta_months
07660 |     y = (abs_month - 1) // 12 + 1
07661 |     m = (abs_month - 1) % 12 + 1
07662 |     return int(y), int(m)
07663 |
07664 | def get_approx_nm_tt(self, y, m):

```

```

07665 |         abs_month = (y - 1) * 12 + m
07666 |         delta_months = abs_month - 17038
07667 |         return 2451550.0 + delta_months * 29.530588853
07668 |
07669 |     def calculate_khgt_1st_of_month(self, nm_tt_float):
07670 |         # Menggunakan Julian Date (Float TT) agar kebal dari error tahun minus Python
07671 |         t_nm = self.ts.tt_jd(nm_tt_float)
07672 |         y, m, d, h, mi, s = t_nm.utc
07673 |
07674 |         # Cari titik waktu 00:00 UTC pada hari konjungsi
07675 |         t_midnight = self.ts.utc(y, m, d)
07676 |
07677 |         # Aturan KHGT: Jika ijtimak sebelum jam 15:00 UTC (Fajar Selandia Baru),
07678 |         # bulan baru masuk besok (+1 hari). Jika tidak, lusa (+2 hari).
07679 |         if h < 15:
07680 |             return t_midnight.tt + 1.0
07681 |         else:
07682 |             return t_midnight.tt + 2.0
07683 |
07684 |     def calculate_conversion(self):
07685 |         try:
07686 |             import math
07687 |             mode_conv = self.radio_conv_var.get()
07688 |             d = int(self.combo_conv_day.get())
07689 |
07690 |             # --- PERBAIKAN FINAL: Kembalikan teks ke bahasa asli & Anti-Crash ---
07691 |             raw_m_str = self.combo_conv_month.get()
07692 |             m_str = self.reverse_tr(raw_m_str)
07693 |
07694 |             y = int(self.entry_conv_year.get())
07695 |             adj = int(self.combo_conv_adj.get())
07696 |
07697 |             # Fungsi bantuan deteksi kabisat Masehi
07698 |             def is_leap(year):
07699 |                 return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
07700 |
07701 |             # -----
07702 |             # NORMALISASI BULAN MULTI-BAHASA UNTUK KONVERSI KALENDER
07703 |             # -----
07704 |             # Database internal aplikasi memakai nama bulan Indonesia:
07705 |             # BULAN_MASEHI dan BULAN_HIJRIAH. Saat UI diganti ke English
07706 |             # atau Arabic, nilai ComboBox menjadi terjemahan, sehingga
07707 |             # pemanggilan list.index(...) langsung dapat memicu error:
07708 |             # 'Ramadan' is not in list / 'نَاصِر' is not in list.
07709 |             # Helper berikut mengembalikan indeks bulan 1..12 secara aman
07710 |             # tanpa mengubah database, algoritma, atau modul lain.
07711 |             def _clean_month_key(value):
07712 |                 value = str(value or "").strip()
07713 |                 value = value.replace("_", "") # hapus tatwil Arab
07714 |                 value = re.sub(r"\s+", " ", value)
07715 |                 return value.casefold()
07716 |
07717 |             def get_gregorian_month_index(value):
07718 |                 key = _clean_month_key(value)
07719 |                 masehi_map = {}
07720 |
07721 |                 # Indonesia/internal
07722 |                 for idx, name in enumerate(BULAN_MASEHI, 1):
07723 |                     masehi_map[_clean_month_key(name)] = idx
07724 |
07725 |                 # English
07726 |                 en_names = [
07727 |                     "January", "February", "March", "April", "May", "June",
07728 |                     "July", "August", "September", "October", "November", "December"
07729 |                 ]
07730 |                 en_abbr = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
07731 |                 for idx, name in enumerate(en_names, 1):
07732 |                     masehi_map[_clean_month_key(name)] = idx
07733 |                 for idx, name in enumerate(en_abbr, 1):
07734 |                     masehi_map[_clean_month_key(name)] = idx
07735 |

```

```

07736 |         # Arabic Gregorian umum
07737 |         ar_names = [
07738 |             "ربيعي", "ربيع", "سرام", "لربا", "لربا", "ويام", "وينوي",
07739 |             "ويلوي", "سبطس", "سبطس", "ربمبس", "ربوتكا", "ربوتكا",
07740 |             "ربمفون", "ربمفون"
07741 |         ]
07742 |         ar_idx = [1, 2, 3, 4, 4, 5, 6, 7, 8, 8, 9, 10, 10, 11, 12]
07743 |         for name, idx in zip(ar_names, ar_idx):
07744 |             masehi_map[_clean_month_key(name)] = idx
07745 |
07746 |         # Coba dari file translator JSON bila tersedia
07747 |         for idx, id_name in enumerate(BULAN_MASEHI, 1):
07748 |             for lang_val in getattr(self, "translations", {}).get(id_name, {}).values():
07749 |                 masehi_map[_clean_month_key(lang_val)] = idx
07750 |
07751 |         if key in masehi_map:
07752 |             return masehi_map[key]
07753 |
07754 |         # Fallback untuk format singkatan English seperti Sept.
07755 |         key3 = key[:3]
07756 |         for k, idx in masehi_map.items():
07757 |             if len(k) >= 3 and k[:3] == key3:
07758 |                 return idx
07759 |
07760 |         raise ValueError(f>Nama bulan Masehi tidak dikenali: {value}")
07761 |
07762 |     def get_hijri_month_index(value):
07763 |         key = _clean_month_key(value)
07764 |         hijri_map = {}
07765 |
07766 |         # Indonesia/internal
07767 |         for idx, name in enumerate(BULAN_HIJRIAH, 1):
07768 |             hijri_map[_clean_month_key(name)] = idx
07769 |
07770 |         # Variasi Indonesia yang sering dipakai
07771 |         id_aliases = {
07772 |             1: ["Muharram", "Muharam"],
07773 |             3: ["Rabiul Awal", "Rabiulawal", "Rabi'ul Awal"],
07774 |             4: ["Rabiul Akhir", "Rabiulakhir", "Rabi'ul Akhir"],
07775 |             5: ["Jumadil Awal", "Jumadilawal", "Jumada Awal"],
07776 |             6: ["Jumadil Akhir", "Jumadilakhir", "Jumada Akhir"],
07777 |             8: ["Syaban", "Sya'ban", "Syakban"],
07778 |             9: ["Ramadan", "Ramadhan"],
07779 |             11: ["Zulkaidah", "Dzulqaidah", "Zulqaidah"],
07780 |             12: ["Zulhijah", "Dzulhijjah", "Zulhijjah", "Dzulhijjah"]
07781 |         }
07782 |
07783 |         # English Hijri names
07784 |         en_aliases = {
07785 |             1: ["Muharram", "Muharam"],
07786 |             2: ["Safar"],
07787 |             3: ["Rabi al-Awwal", "Rabi' al-Awwal", "Rabi I", "Rabiul Awwal"],
07788 |             4: ["Rabi al-Thani", "Rabi' al-Thani", "Rabi al-Akhir", "Rabi II"],
07789 |             5: ["Jumada al-Awwal", "Jumada I", "Jumada al-Ula"],
07790 |             6: ["Jumada al-Thani", "Jumada II", "Jumada al-Akhirah"],
07791 |             7: ["Rajab"],
07792 |             8: ["Sha'ban", "Shaban", "Shaaban"],
07793 |             9: ["Ramadan", "Ramadhan"],
07794 |             10: ["Shawwal", "Syawal"],
07795 |             11: ["Dhu al-Qadah", "Dhu' Qadah", "Dhu al-Qi'dah", "Dhu' Qi'dah", "Zulkaidah"],
07796 |             12: ["Dhu al-Hijjah", "Dhu' Hijjah", "Zulhijah", "Zulhijjah"]
07797 |         }
07798 |
07799 |         # Arabic Hijri names
07800 |         ar_aliases = {
07801 |             1: ["محرم", "رمح"],
07802 |             2: ["رفعي"],
07803 |             3: ["لوا عيبر", "لوا ال عيبر", "لوا عيبر", "لوا عيبر"],
07804 |             4: ["ينال عيبر", "خال عيبر", "خال عيبر"],
07805 |             5: ["يلو ادا", "يلو ادا", "يلو ادا", "يلو ادا"],
07806 |             6: ["ينال ادا", "خال ادا", "خال ادا", "ينال ادا"],

```

```

07807 |         7: ["بجر"],
07808 |         8: ["نابغش"],
07809 |         9: ["ناعمر"],
07810 |         10: ["ل"وش", "ل"وش"],
07811 |         11: ["دعقلاود", "دعقلاود", "دعقلاود", "دعقلاود"],
07812 |         12: ["دعقلاود", "دعقلاود", "دعقلاود", "دعقلاود"]
07813 |     }
07814 |
07815 |     for alias_dict in (id_aliases, en_aliases, ar_aliases):
07816 |         for idx, names in alias_dict.items():
07817 |             for name in names:
07818 |                 hijri_map[_clean_month_key(name)] = idx
07819 |
07820 |     # Coba dari file translator JSON bila tersedia
07821 |     for idx, id_name in enumerate(BULAN_HIJRIAH, 1):
07822 |         for lang_val in getattr(self, "translations", {}).get(id_name, {}).values():
07823 |             hijri_map[_clean_month_key(lang_val)] = idx
07824 |
07825 |     if key in hijri_map:
07826 |         return hijri_map[key]
07827 |
07828 |     raise ValueError(f>Nama bulan Hijriah tidak dikenali: {value}")
07829 |
07830 | # Algoritma Julian Day untuk Kalender Urfi
07831 | def get_jd(year, month, day):
07832 |     if month <= 2:
07833 |         year -= 1
07834 |         month += 12
07835 |     A = math.floor(year / 100)
07836 |     B = 2 - A + math.floor(A / 4)
07837 |     return math.floor(365.25 * (year + 4716)) + math.floor(30.6001 * (month + 1)) + day + B - 1524.5
07838 |
07839 | def jd_to_urfi(jd):
07840 |     jd = math.floor(jd) + 0.5
07841 |     days = math.floor(jd - 1948439.5)
07842 |     y_h = math.floor((30 * days + 10646) / 10631)
07843 |     d_h = days - math.floor((11 * y_h + 3) / 30) - 354 * y_h
07844 |     m_h = math.floor((11 * d_h + 330) / 325)
07845 |     d_final = d_h - math.floor((325 * m_h - 320) / 11)
07846 |     return int(y_h), int(m_h), int(d_final)
07847 |
07848 | def urfi_to_jd(y_h, m_h, d_h):
07849 |     return d_h + math.floor((325 * m_h - 320) / 11) + math.floor((11 * y_h + 3) / 30) + 354 * y_h + 1948439.5
07850 |
07851 | def jd_to_gregorian(jd):
07852 |     Z = math.floor(jd + 0.5)
07853 |     if Z < 2299161:
07854 |         A = Z
07855 |     else:
07856 |         alpha = math.floor((Z - 1867216.25) / 36524.25)
07857 |         A = Z + 1 + alpha - math.floor(alpha / 4)
07858 |     B = A + 1524
07859 |     C = math.floor((B - 122.1) / 365.25)
07860 |     D = math.floor(365.25 * C)
07861 |     E = math.floor((B - D) / 30.6001)
07862 |     day = B - D - math.floor(30.6001 * E)
07863 |     month = E - 1 if E < 14 else E - 13
07864 |     year = C - 4716 if month > 2 else C - 4715
07865 |     return int(year), int(month), int(day)
07866 |
07867 | # Array Master
07868 | hari_masehi = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
07869 | pasaran_jawa = ["Legi", "Pahing", "Pon", "Wage", "Kliwon"]
07870 |
07871 | catatan_tambahan = ""
07872 |
07873 | # =====
07874 | # LOGIKA 1: MASEHI -> HIJRIAH (KHGT)
07875 | # =====
07876 | if mode_conv == "m2h":
07877 |     m = get_gregorian_month_index(m_str)

```

```

07878 |         m_id = BULAN_MASEHI[m - 1]
07879 |
07880 |         mode_text = "Masehi → Hijriah (KHGT)"
07881 |         greg_year_str_input = f"{y}" if y > 0 else f"{abs(y-1)} SM"
07882 |         input_text = f"{d} {m_id} {greg_year_str_input}"
07883 |
07884 |         self.auto_switch_ephemeris(y)
07885 |         days_in_month = [31, 29 if is_leap(y) else 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
07886 |         d = min(d, days_in_month[m-1])
07887 |
07888 |         target_t_obj = self.ts.utc(y, m, d)
07889 |         target_tt = target_t_obj.tt
07890 |
07891 |         new_moons_tt_list = self.get_new_moons_in_range(target_tt - 35.0, target_tt + 2.0)
07892 |         current_nm_tt = None
07893 |         for nm_tt in reversed(new_moons_tt_list):
07894 |             start_tt = self.calculate_khgt_1st_of_month(nm_tt)
07895 |             if target_tt >= start_tt - 0.1:
07896 |                 current_nm_tt = nm_tt
07897 |                 break
07898 |
07899 |         if current_nm_tt is None:
07900 |             raise ValueError(f"Tanggal target di luar jangkauan Ephemeris ({self.ephemeris_name}).")
07901 |
07902 |         h_y, h_m = self.get_hijri_month_from_tt(current_nm_tt)
07903 |         start_tt = self.calculate_khgt_1st_of_month(current_nm_tt)
07904 |
07905 |         h_d = int(round(target_tt - start_tt)) + 1
07906 |         h_d += adj
07907 |
07908 |         while h_d <= 0:
07909 |             h_m -= 1
07910 |             if h_m <= 0: h_m, h_y = 12, h_y - 1
07911 |             h_d += 30
07912 |         while h_d > 30:
07913 |             h_d -= 30
07914 |             h_m += 1
07915 |             if h_m > 12: h_m, h_y = 1, h_y + 1
07916 |
07917 |         try:
07918 |             if y > 0:
07919 |                 dt_obj = datetime.date(y, m, d)
07920 |                 hari = f"{hari_masehi[dt_obj.weekday()] {pasaran_jawa[dt_obj.toordinal() % 5]}"
07921 |             else:
07922 |                 hari = hari_masehi[int((target_t_obj.whole + 0.5) % 7)]
07923 |         except:
07924 |             hari = hari_masehi[int((target_t_obj.whole + 0.5) % 7)]
07925 |
07926 |         hijri_year_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
07927 |         hasil = f"{hari}, {h_d} {BULAN_HIJIRIAH[h_m - 1]} {hijri_year_str} H"
07928 |         catatan_tambahan = "- Konversi ini berbasis kriteria Kalender Hijriah Global Tunggal (KHGT).\n- Satu hari Hijriah dimulai sejak terbenamnya matahari (sunset) hari sebelumnya.\n- Kalkulasi menggunakan data astronomis presisi tinggi (Fase Bulan Baru/Ijtimak)."
```

```

07929 |
07930 |         # =====
07931 |         # LOGIKA 2: HIJIRIAH (KHGT) -> MASEHI
07932 |         # =====
07933 |         elif mode_conv == "h2m":
07934 |             m = get_hijri_month_index(m_str)
07935 |             m_id = BULAN_HIJIRIAH[m - 1]
07936 |             h_y = y
07937 |             mode_text = "Hijriah (KHGT) → Masehi"
07938 |             hijri_year_str_input = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
07939 |             input_text = f"{d} {m_id} {hijri_year_str_input} H"
07940 |
07941 |             approx_greg_year = int(h_y * 0.970224 + 622.54)
07942 |             self.auto_switch_ephemeris(approx_greg_year)
07943 |
07944 |             approx_tt = self.get_approx_nm_tt(h_y, m)
07945 |             new_moons_tt_list = self.get_new_moons_in_range(approx_tt - 5.0, approx_tt + 5.0)
07946 |             if not new_moons_tt_list:
07947 |                 raise ValueError(f"Tanggal tersebut di luar jangkauan Ephemeris ({self.ephemeris_name}).")

```

```

07948 |
07949 |         exact_nm_tt = new_moons_tt_list[0]
07950 |         start_tt = self.calculate_khgt_lst_of_month(exact_nm_tt)
07951 |
07952 |         target_tt = start_tt + (d - 1) - adj
07953 |         target_t_obj = self.ts.tt_jd(target_tt)
07954 |         ty, tm, td, _, _, _ = target_t_obj.utcd
07955 |
07956 |         try:
07957 |             ty_int, tm_int, td_int = int(ty), int(tm), int(td)
07958 |             if ty_int > 0:
07959 |                 dt_obj = datetime.date(ty_int, tm_int, td_int)
07960 |                 hari = f"{hari_masehi[dt_obj.weekday()]} {pasaran_jawa[dt_obj.toordinal() % 5]}"
07961 |             else:
07962 |                 hari = hari_masehi[int((target_t_obj.whole + 0.5) % 7)]
07963 |         except:
07964 |             hari = hari_masehi[int((target_t_obj.whole + 0.5) % 7)]
07965 |
07966 |         greg_year_str = f"{int(ty)}" if ty > 0 else f"{abs(int(ty)-1)} SM"
07967 |         hasil = f"{hari}, {int(td)} {BULAN_MASEHI[int(tm) - 1]} {greg_year_str}"
07968 |         catatan_tambahan = "- Konversi ini berbasis kriteria Kalender Hijriah Global Tunggal (KHGT).\n- Satu hari Hijriah dimulai sejak terbenamnya matahari (sunset) hari sebelumnya.\n- Kalkulasi menggunakan data astronomis presisi tinggi (Fase Bulan Baru/Ijtimak)."
```

```

07969 |
07970 |         # =====
07971 |         # LOGIKA 3: MASEHI -> HIJRIAH (URFI / KUWEIT)
07972 |         # =====
07973 |         elif mode_conv == "m2u":
07974 |             m = get_gregorian_month_index(m_str)
07975 |             m_id = BULAN_MASEHI[m - 1]
07976 |
07977 |             mode_text = "Masehi -> Hijriah (Urfi)"
07978 |             greg_year_str_input = f"{y}" if y > 0 else f"{abs(y-1)} SM"
07979 |             input_text = f"{d} {m_id} {greg_year_str_input}"
07980 |
07981 |             jd = get_jd(y, m, d)
07982 |             jd += adj # Koreksi hari
07983 |             h_y, h_m, h_d = jd_to_urfi(jd)
07984 |
07985 |             try:
07986 |                 if y > 0:
07987 |                     dt_obj = datetime.date(y, m, d)
07988 |                     hari = f"{hari_masehi[dt_obj.weekday()]} {pasaran_jawa[dt_obj.toordinal() % 5]}"
07989 |                 else:
07990 |                     hari = hari_masehi[int((jd + 1.5) % 7)]
07991 |             except:
07992 |                 hari = hari_masehi[int((jd + 1.5) % 7)]
07993 |
07994 |             hijri_year_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
07995 |             hasil = f"{hari}, {h_d} {BULAN_HIJRIAH[h_m - 1]} {hijri_year_str} H"
07996 |             catatan_tambahan = "- Konversi ini menggunakan algoritma Tabular (Urfi/Kuweit Standard).\n- Bersifat murni matematis dari siklus 30 tahun (11 tahun kabisat).\n- Tidak berdasarkan visibilitas/fase bulan aktual (Bukan Hisab Hakiki)."
```

```

07997 |
07998 |         # =====
07999 |         # LOGIKA 4: HIJRIAH (URFI / KUWEIT) -> MASEHI
08000 |         # =====
08001 |         elif mode_conv == "u2m":
08002 |             m_h = get_hijri_month_index(m_str)
08003 |             m_id = BULAN_HIJRIAH[m_h - 1]
08004 |             y_h = y
08005 |             mode_text = "Hijriah (Urfi) -> Masehi"
08006 |             hijri_year_str_input = str(y_h) if y_h > 0 else f"{abs(y_h-1)} SH"
08007 |             input_text = f"{d} {m_id} {hijri_year_str_input} H"
08008 |
08009 |             d_adj = d - adj # Inversi koreksi hari
08010 |             jd = urfi_to_jd(y_h, m_h, d_adj)
08011 |             y_m, m_m, d_m = jd_to_gregorian(jd)
08012 |
08013 |             try:
08014 |                 if y_m > 0:
08015 |                     dt_obj = datetime.date(y_m, m_m, d_m)
08016 |                     hari = f"{hari_masehi[dt_obj.weekday()]} {pasaran_jawa[dt_obj.toordinal() % 5]}"

```

```

08017 |         else:
08018 |             hari = hari_masehi[int((jd + 1.5) % 7)]
08019 |     except:
08020 |         hari = hari_masehi[int((jd + 1.5) % 7)]
08021 |
08022 |         greg_year_str = f"{y_m}" if y_m > 0 else f"{abs(y_m-1)} SM"
08023 |         hasil = f"{hari}, {d_m} {BULAN_MASEHI[m_m - 1]} {greg_year_str}"
08024 |         catatan_tambahan = "- Konversi ini menggunakan algoritma Tabular (Urfi/Kuweit Standard).\n- Bersifat murni matematis dari siklus 30 tahun (11 tahun kabisat).\n- Tidak berdasarkan
visibilitas/fase bulan aktual (Bukan Hisab Hakiki)."
```

```

08025 |
08026 |     # --- RENDER LAPORAN ---
08027 |     report = f""#{self.get_header(85)}
08028 |     {"[ Konversi Kalender Terpadu ]".center(85)}
08029 |
08030 |     * Settings:-
08031 |     - Mode Konversi : {mode_text}
08032 |     - Tanggal Input : {input_text}
08033 |     - Koreksi Hari   : {adj} Hari (Penyesuaian Manual)
08034 |     =====
08035 |
08036 |     HASIL KONVERSI:
08037 |     -----
08038 |     >> {hasil} <<
08039 |     -----
08040 |
08041 |     * Catatan:
08042 |     {catatan_tambahan}
08043 |     =====
08044 |     """
08045 |         self.after(0, self.display_result, report)
08046 |
08047 |     except Exception as e:
08048 |         import traceback
08049 |         trace_err = traceback.format_exc()
08050 |         self.after(0, self.display_error, f"{str(e)}\n\n(Details):\n{trace_err}")
08051 |
08052 |     def generate_visibility_report(self):
08053 |         """Fungsi inti kalkulasi hilal yang me-return string report murni (Bisa dipanggil oleh WinAI)"""
08054 |         import math
08055 |         import pytz
08056 |         import datetime
08057 |         import numpy as np
08058 |         import ephem
08059 |         from skyfield import almanac
08060 |         from skyfield.api import wgs84
08061 |
08062 |         year = int(self.entry_vyear.get())
08063 |         month = int(self.entry_vmonth.get())
08064 |         day = int(self.entry_vday.get())
08065 |
08066 |         self.auto_switch_ephemeris(year)
08067 |
08068 |         lat = float(self.entry_vlat.get())
08069 |         lon = float(self.entry_vlon.get())
08070 |         elev = float(self.entry_velev.get())
08071 |         tz = float(self.entry_vtz.get())
08072 |
08073 |         temp = float(self.entry_vtemp.get())
08074 |         pres = float(self.entry_vpres.get())
08075 |         hum = float(self.entry_vhum.get())
08076 |
08077 |         earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
08078 |         lokasi_obs = wgs84.latlon(lat, lon, elevation_m=elev)
08079 |
08080 |         # 1. CARI SUNSET LOKAL UNTUK TITIK OBSERVASI
08081 |         t0 = self.ts.utc(year, month, day, 0 - int(tz))
08082 |         t1 = self.ts.utc(year, month, day, 23 - int(tz))
08083 |
08084 |         t_sunset = None
08085 |         t_evs, y_evs = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(self.eph, lokasi_obs))
08086 |         for t_ev, is_sunrise in get_safe_events(t_evs, y_evs):

```

```

08087 |         if not is_sunrise:
08088 |             t_sunset = t_ev
08089 |             break
08090 |
08091 |     if t_sunset is None:
08092 |         raise ValueError("Sunset (Matahari Terbenam) tidak ditemukan pada tanggal/koordinat tersebut.")
08093 |
08094 |     waktu_mode = getattr(self, 'radio_vis_time_mode', None)
08095 |     mode_val = waktu_mode.get() if waktu_mode else "maghrib"
08096 |
08097 |     if mode_val == "live":
08098 |         t_eval = self.ts.now()
08099 |         waktu_kalkulasi_str = "LIVE (Real-Time)"
08100 |     elif mode_val == "kustom":
08101 |         time_str = self.entry_vtime.get().strip()
08102 |         if not time_str: time_str = "12:00:00"
08103 |         parts = time_str.split(":")
08104 |         jam = int(parts[0])
08105 |         menit = int(parts[1]) if len(parts) > 1 else 0
08106 |         detik = int(parts[2]) if len(parts) > 2 else 0
08107 |
08108 |         dt_lokal = datetime.datetime(year, month, day, jam, menit, detik)
08109 |         dt_utc = dt_lokal - datetime.timedelta(hours=tz)
08110 |         t_eval = self.ts.utc(dt_utc.year, dt_utc.month, dt_utc.day, dt_utc.hour, dt_utc.minute, dt_utc.second)
08111 |         waktu_kalkulasi_str = f"{time_str} LT (Kustom)"
08112 |     else:
08113 |         t_eval = t_sunset
08114 |         waktu_kalkulasi_str = f"{get_hms_str(t_sunset, tz)} LT (Sunset)"
08115 |
08116 |     ts_tt = t_sunset.tt.item() if hasattr(t_sunset.tt, 'item') else t_sunset.tt
08117 |     t_eval_tt = t_eval.tt.item() if hasattr(t_eval.tt, 'item') else t_eval.tt
08118 |
08119 |     t_bound_moon = self.ts.tt_jd(ts_tt + 1.5)
08120 |     t_moonset = None
08121 |     t_mevs, y_mevs = almanac.find_discrete(t0, t_bound_moon, almanac.risings_and_settings(self.eph, moon, lokasi_obs))
08122 |
08123 |     for t_ev, is_moonrise in get_safe_events(t_mevs, y_mevs):
08124 |         tm_tt = t_ev.tt.item() if hasattr(t_ev.tt, 'item') else t_ev.tt
08125 |         if not is_moonrise and tm_tt > (ts_tt - 0.5):
08126 |             t_moonset = t_ev
08127 |             break
08128 |
08129 |     # 2. CARI IJTIMAK GLOBAL (+/- 20 Hari)
08130 |     t_start_conj = self.ts.tt_jd(t_eval_tt - 20.0)
08131 |     t_end_conj = self.ts.tt_jd(t_eval_tt + 20.0)
08132 |
08133 |     t_phases, y_phases = almanac.find_discrete(t_start_conj, t_end_conj, almanac.moon_phases(self.eph))
08134 |     t_ijtimak = None
08135 |     jarak_terdekat = float('inf')
08136 |
08137 |     for t_ev, phase in get_safe_events(t_phases, y_phases):
08138 |         if phase == 0:
08139 |             selisih_hari = abs(t_ev.tt - t_eval_tt)
08140 |             if selisih_hari < jarak_terdekat:
08141 |                 jarak_terdekat = selisih_hari
08142 |                 t_ijtima = t_ev
08143 |
08144 |     # 3. KALKULASI PARAMETER LOKAL SAAT WAKTU EVALUASI
08145 |     geo_earth = earth.at(t_eval)
08146 |     app_moon_geo = geo_earth.observe(moon).apparent()
08147 |     app_sun_geo = geo_earth.observe(sun).apparent()
08148 |
08149 |     ra_moon, dec_moon, dist_moon = app_moon_geo.radec(epoch=t_eval)
08150 |     ra_sun, dec_sun, dist_sun = app_sun_geo.radec(epoch=t_eval)
08151 |
08152 |     gast = t_eval.gast
08153 |     lst_deg = (gast * 15.0) + lon
08154 |
08155 |     def get_geo_altaz(ra, dec):
08156 |         ra_h = ra.hours.item() if hasattr(ra.hours, 'item') else ra.hours
08157 |         dec_r = dec.radians.item() if hasattr(dec.radians, 'item') else dec.radians

```



```

08158 |         ha_deg = lst_deg - (ra_h * 15.0)
08159 |         lat_rad, dec_rad, ha_rad = math.radians(lat), dec_r, math.radians(ha_deg)
08160 |         sin_alt = math.sin(dec_rad) * math.sin(lat_rad) + math.cos(dec_rad) * math.cos(lat_rad) * math.cos(ha_rad)
08161 |         sin_alt = max(-1.0, min(1.0, sin_alt))
08162 |         alt = math.degrees(math.asin(sin_alt))
08163 |         y = -math.sin(ha_rad)
08164 |         x = math.tan(dec_rad) * math.cos(lat_rad) - math.sin(lat_rad) * math.cos(ha_rad)
08165 |         az = (math.degrees(math.atan2(y, x)) + 360) % 360
08166 |         return alt, az
08167 |
08168 |     alt_moon_geo, az_moon_geo = get_geo_altaz(ra_moon, dec_moon)
08169 |     alt_sun_geo, az_sun_geo = get_geo_altaz(ra_sun, dec_sun)
08170 |
08171 |     topo_obs = (earth + lokasi_obs).at(t_eval)
08172 |     app_moon_topo = topo_obs.observe(moon).apparent()
08173 |     app_sun_topo = topo_obs.observe(sun).apparent()
08174 |
08175 |     alt_moon_topo_obj, az_moon_topo_obj, _ = app_moon_topo.altaz(temperature_C=temp, pressure_mbar=pres)
08176 |     alt_sun_topo_obj, az_sun_topo_obj, _ = app_sun_topo.altaz(temperature_C=temp, pressure_mbar=pres)
08177 |
08178 |     alt_moon_topo = alt_moon_topo_obj.degrees
08179 |     alt_sun_topo = alt_sun_topo_obj.degrees
08180 |
08181 |     ref_moon = calculate_saemundsson_refraction(alt_moon_topo, temp, pres)
08182 |     ref_sun = calculate_saemundsson_refraction(alt_sun_topo, temp, pres)
08183 |
08184 |     if t_ijtima is not None:
08185 |         ti_tt = t_ijtima.tt.item() if hasattr(t_ijtima.tt, 'item') else t_ijtima.tt
08186 |         moon_age_hours = (t_eval_tt - ti_tt) * 24.0
08187 |     else:
08188 |         moon_age_hours = 0.0
08189 |
08190 |     if t_moonset is not None:
08191 |         tm_tt = t_moonset.tt.item() if hasattr(t_moonset.tt, 'item') else t_moonset.tt
08192 |         lag_time_hours = (tm_tt - ts_tt) * 24.0
08193 |     else:
08194 |         lag_time_hours = 0.0
08195 |
08196 |     sep_deg = app_sun_geo.separation_from(app_moon_geo).degrees
08197 |     elongation = sep_deg.item() if hasattr(sep_deg, 'item') else sep_deg
08198 |     rel_alt_topo = alt_moon_topo - alt_sun_topo
08199 |
08200 |     def moon_sun_elongation(t):
08201 |         e = earth.at(t)
08202 |         _, slon, _ = e.observe(sun).apparent().ecliptic_latlon(epoch=t)
08203 |         _, mlon, _ = e.observe(moon).apparent().ecliptic_latlon(epoch=t)
08204 |         s_deg = slon.degrees.item() if hasattr(slon.degrees, 'item') else slon.degrees
08205 |         m_deg = mlon.degrees.item() if hasattr(mlon.degrees, 'item') else mlon.degrees
08206 |         return (m_deg - s_deg) % 360.0
08207 |
08208 |     phase_angle = moon_sun_elongation(t_eval)
08209 |     illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_eval)
08210 |     illumination = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0
08211 |
08212 |     dist_km = dist_moon.km.item() if hasattr(dist_moon.km, 'item') else dist_moon.km
08213 |     sd_moon = math.degrees(math.asin(1737.4 / dist_km))
08214 |
08215 |     W_arcmin = (sd_moon * 60.0) * (1 - math.cos(math.radians(elongation)))
08216 |     q_yallop = rel_alt_topo - (11.8371 - 6.3226 * W_arcmin + 0.7319 * (W_arcmin**2) - 0.1018 * (W_arcmin**3))
08217 |
08218 |     if q_yallop > 0.216: yallop_stat = "A (Mudah Terlihat / Mata Telanjang)"
08219 |     elif q_yallop > -0.014: yallop_stat = "B (Terlihat dgn Cuaca Sempurna)"
08220 |     elif q_yallop > -0.160: yallop_stat = "C (Butuh Alat Optik utk Menemukan)"
08221 |     elif q_yallop > -0.232: yallop_stat = "D (Hanya Terlihat via Alat Optik)"
08222 |     else: yallop_stat = "F (TIDAK TERLIHAT / Bawah Limit)"
08223 |
08224 |     if t_ijtima is not None:
08225 |         dt_utc_ijtima = t_ijtima.utc_datetime()
08226 |         dt_lokal_ijtima = dt_utc_ijtima + datetime.timedelta(hours=tz)
08227 |         y_i, m_i, d_i = dt_lokal_ijtima.year, dt_lokal_ijtima.month, dt_lokal_ijtima.day
08228 |         h_i, mn_i = dt_lokal_ijtima.hour, dt_lokal_ijtima.minute

```

```

08229 |         y_str = f"{y_i}" if y_i > 0 else f"{abs(y_i-1)} SM"
08230 |         str_ijtima = f"{d_i:02d}/{m_i:02d}/{y_str}, {int(h_i):02d}:{int(mn_i):02d} LT"
08231 |         str_ijtima_utc = dt_utc_ijtima.strftime('%d/%m/%Y, %H:%M UTC')
08232 |     else:
08233 |         str_ijtima, str_ijtima_utc = "N/A", "N/A"
08234 |
08235 |     # Variabel Logika Kriteria Lokal
08236 |     is_khgt_local = alt_moon_geo >= 5.0 and elongation >= 8.0
08237 |
08238 |     is_mabims = alt_moon_topo >= 3.0 and elongation >= 6.4
08239 |     str_mabims = "TERPENUHI" if is_mabims else "Gagal"
08240 |
08241 |     is_ilyas = alt_moon_topo >= 4.0 and elongation >= 10.5
08242 |     str_ilyas = "TERPENUHI" if is_ilyas else "Gagal"
08243 |
08244 |     is_danjon = elongation >= 7.0
08245 |     str_danjon = "LOLOS (Sabit Terbentuk)" if is_danjon else "GAGAL (Sabit Tidak Terbentuk)"
08246 |
08247 |     # =====
08248 |     # 4. PEMINDAIAN GLOBAL KHGT (PKG 1 & PKG 2)
08249 |     # =====
08250 |     status_khgt_global = "GAGAL (Belum Terpenuhi di belahan bumi manapun)"
08251 |     pkg_reason = ""
08252 |     titik_pertama_global = None
08253 |
08254 |     if t_ijtima is not None:
08255 |         obs_date_utc = datetime.date(dt_utc_ijtima.year, dt_utc_ijtima.month, dt_utc_ijtima.day)
08256 |         batas_00_utc_ephem = ephemeris.Date(datetime.datetime(obs_date_utc.year, obs_date_utc.month, obs_date_utc.day) + datetime.timedelta(days=1))
08257 |
08258 |         # Cari Fajar Gisborne untuk PKG 2
08259 |         gisborne = ephemeris.Observer()
08260 |         gisborne.lat, gisborne.lon = math.radians(-38.6623), math.radians(178.0176)
08261 |         gisborne.elevation, gisborne.pressure, gisborne.horizon = 0, 0, '-18'
08262 |         matahari_ephem = ephemeris.Sun()
08263 |         gisborne.date = ephemeris.Date(batas_00_utc_ephem - 0.5)
08264 |         try:
08265 |             fajar_gisborne = gisborne.next_rising(matahari_ephem, use_center=True)
08266 |         except:
08267 |             fajar_gisborne = ephemeris.Date(batas_00_utc_ephem + 0.6)
08268 |
08269 |         is_ijtima_before_fajar = ephemeris.Date(dt_utc_ijtima) < fajar_gisborne
08270 |
08271 |         pkg1_met = False
08272 |         pkg2_met = False
08273 |
08274 |         # ---> LANGKAH 1: JADIKAN TITIK LOKAL SEBAGAI PRIORITAS PERTAMA (Akselerasi) <---
08275 |         nama_kota = getattr(self, 'opt_city', None)
08276 |         nama_kota = nama_kota.get() if nama_kota else "Lokasi Anda"
08277 |         nama_prov = getattr(self, 'opt_prov', None)
08278 |         nama_prov = nama_prov.get() if nama_prov else ""
08279 |
08280 |         if is_khgt_local:
08281 |             if ephemeris.Date(t_sunset.utc_datetime()) < batas_00_utc_ephem:
08282 |                 pkg1_met = True
08283 |                 pkg_reason = f"PKG 1 (Titik Lokal Anda: {nama_kota})"
08284 |                 titik_pertama_global = f"{nama_kota} | Sunset UTC: {t_sunset.utc_strftime('%H:%M:%S')}"
08285 |             else:
08286 |                 # Jika sunset lokal lewat 00:00 UTC, apakah ini di Benua Amerika?
08287 |                 negara_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Amerika Selatan", "Brasil", "Argentina", "Kolombia", "Peru", "Chili"]
08288 |                 if nama_prov in negara_amerika or "Amerika" in nama_prov:
08289 |                     if is_ijtima_before_fajar:
08290 |                         pkg2_met = True
08291 |                         pkg_reason = f"PKG 2 (Titik Lokal Anda: {nama_kota} & Ijtime < Fajar NZ)"
08292 |                         titik_pertama_global = f"{nama_kota} | Sunset UTC: {t_sunset.utc_strftime('%H:%M:%S')}"
08293 |
08294 |         # ---> LANGKAH 2: JIKA LOKAL GAGAL, SCAN SELURUH DUNIA (PKG 1) <---
08295 |         if not pkg1_met and not pkg2_met:
08296 |             for lon_g in range(180, -181, -15):
08297 |                 if pkg1_met: break
08298 |                 for lat_g in range(-60, 61, 15):
08299 |                     loc_g = wgs84.latlon(lat_g, lon_g)

```

```

08300 |         est_sunset_utc = 12.0 - (lon_g / 15.0)
08301 |         t0_g = self.ts.utc(year, month, day, est_sunset_utc - 2)
08302 |         t1_g = self.ts.utc(year, month, day, est_sunset_utc + 2)
08303 |
08304 |         try:
08305 |             t_evs, y_evs = almanac.find_discrete(t0_g, t1_g, almanac.sunrise_sunset(self.eph, loc_g))
08306 |             for t_ev, is_sr in zip(np.atleast_1d(t_evs.tt), np.atleast_1d(y_evs)):
08307 |                 if not is_sr:
08308 |                     t_ss_sky = self.ts.tt_jd(t_ev)
08309 |                     if ephemeris.Date(t_ss_sky.utc_datetime()) < batas_00_utc_ephem:
08310 |                         obs_center = earth.at(t_ss_sky)
08311 |                         m_app = obs_center.observe(moon).apparent()
08312 |                         s_app = obs_center.observe(sun).apparent()
08313 |                         elong_g = s_app.separation_from(m_app).degrees
08314 |
08315 |                     if elong_g >= 8.0:
08316 |                         ra_m, dec_m, _ = m_app.radec(epoch=t_ss_sky)
08317 |                         ha_deg = (t_ss_sky.gast * 15.0) + lon_g - (ra_m.hours * 15.0)
08318 |                         sin_alt = math.sin(dec_m.radians) * math.sin(math.radians(lat_g)) + math.cos(dec_m.radians) * math.cos(math.radians(lat_g)) * math.cos(math.radians(ha_deg))
08319 |                         alt_g = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
08320 |
08321 |                     if alt_g >= 5.0:
08322 |                         pkg1_met = True
08323 |                         pkg_reason = f"PKG 1 (Titik Pindai: Lat {lat_g}°, Lon {lon_g}° [Darat/Lautan])"
08324 |                         titik_pertama_global = f"Lat {lat_g}°, Lon {lon_g}° | Sunset UTC: {t_ss_sky.utc_strftime('%H:%M:%S')}"
08325 |                         break
08326 |         except Exception:
08327 |             pass
08328 |
08329 | # ---> LANGKAH 3: JIKA PKG 1 TETAP GAGAL, EVALUASI PKG 2 DI BENUA AMERIKA <---
08330 | if not pkg1_met and is_ijtima_before_fajr:
08331 |     negara_mainland_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Amerika Selatan"]
08332 |     kota_kepulauan = ["Honolulu", "Papeete", "Alofi", "Pago Pago", "Fernando de Noronha", "Nuuk", "San Juan", "Santo Domingo", "Havana", "Praia"]
08333 |
08334 |     for negara in negara_mainland_amerika:
08335 |         if pkg2_met: break
08336 |         if negara in CITY_DB:
08337 |             for nama_kota_am, coords in CITY_DB[negara].items():
08338 |                 if nama_kota_am in kota_kepulauan: continue
08339 |                 if pkg2_met: break
08340 |
08341 |                 lat_a, lon_a = coords
08342 |                 loc_a = wgs84.latlon(lat_a, lon_a)
08343 |                 tz_a = int(self.get_tz_from_lon(lon_a))
08344 |
08345 |                 t0_a = self.ts.utc(year, month, day, 0 - tz_a)
08346 |                 t1_a = self.ts.utc(year, month, day, 24 - tz_a)
08347 |                 try:
08348 |                     t_evs, y_evs = almanac.find_discrete(t0_a, t1_a, almanac.sunrise_sunset(self.eph, loc_a))
08349 |                     for t_ev, is_sr in zip(np.atleast_1d(t_evs.tt), np.atleast_1d(y_evs)):
08350 |                         if not is_sr:
08351 |                             t_ss_sky = self.ts.tt_jd(t_ev)
08352 |                             if ephemeris.Date(t_ss_sky.utc_datetime()) >= batas_00_utc_ephem:
08353 |                                 obs_center = earth.at(t_ss_sky)
08354 |                                 m_app = obs_center.observe(moon).apparent()
08355 |                                 s_app = obs_center.observe(sun).apparent()
08356 |                                 elong_g = s_app.separation_from(m_app).degrees
08357 |
08358 |                             if elong_g >= 8.0:
08359 |                                 ra_m, dec_m, _ = m_app.radec(epoch=t_ss_sky)
08360 |                                 ha_deg = (t_ss_sky.gast * 15.0) + lon_a - (ra_m.hours * 15.0)
08361 |                                 sin_alt = math.sin(dec_m.radians) * math.sin(math.radians(lat_a)) + math.cos(dec_m.radians) * math.cos(math.radians(lat_a)) * math.cos(math.radians(ha_deg))
08362 |                                 alt_g = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
08363 |
08364 |                             if alt_g >= 5.0:
08365 |                                 pkg2_met = True
08366 |                                 pkg_reason = f"PKG 2: Mainland Amerika ({nama_kota_am}, {negara}) & Ijtimia < Fajar NZ"
08367 |                                 titik_pertama_global = f"{nama_kota_am}, {negara} | Sunset UTC: {t_ss_sky.utc_strftime('%H:%M:%S')}"
08368 |                                 break
08369 |         except:
08370 |             pass

```

```

08371 |
08372 |         # Konklusi Global Akhir
08373 |         nm_tt_val = self.ts.from_datetime(dt_utc_ijtima.replace(tzinfo=pytz.utc)).tt
08374 |         h_y, h_m = self.get_hijri_month_from_tt(nm_tt_val)
08375 |         hijri_bulan_str = BULAN_HIJRIAH[h_m - 1]
08376 |         hijri_tahun_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
08377 |
08378 |         tgl_1_mas = ephemer.Date(batas_00_utc_ephem)
08379 |         gy, gm, gd, _, _, _ = tgl_1_mas.tuple()
08380 |         masehi_str_besok = f"{int(gd):02d} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
08381 |
08382 |         tgl_lusa = ephemer.Date(batas_00_utc_ephem + 1.0)
08383 |         gy2, gm2, gd2, _, _, _ = tgl_lusa.tuple()
08384 |         masehi_str_lusa = f"{int(gd2):02d} {BULAN_MASEHI[int(gm2)-1]} {format_tahun_aman(int(gy2))}"
08385 |
08386 |         # --- PERBAIKAN FORMATTING: INDENTASI 47 SPASI AGAR RATA KANAN ---
08387 |         indent = ' ' * 47
08388 |         if pkg1_met:
08389 |             status_khgt_global = f"TERPENUHI GLOBAL (PKG 1: Sunset < 00 UTC)\n{indent}* Titik Pemenuhan : {titik_pertama_global}\n{indent}* Awal Bulan Baru : BESOK (1 {hijri_bulan_str}
{hijri_tahun_str} H = {masehi_str_besok})"
08390 |             elif pkg2_met:
08391 |                 status_khgt_global = f"TERPENUHI GLOBAL (PKG 2: Amerika & Ijtimia < Fajar NZ)\n{indent}* Titik Pemenuhan : {titik_pertama_global}\n{indent}* Awal Bulan Baru : BESOK (1 {hijri_bulan_str}
{hijri_tahun_str} H = {masehi_str_besok})"
08392 |             else:
08393 |                 alasan = "Kriteria tidak tercapai di bumi" if is_ijtima_before_fajr else "Ijtimia > Terbit Fajar NZ"
08394 |                 status_khgt_global = f"GAGAL GLOBAL PKG 1 & 2 ({alasan})\n{indent}* Awal Bulan Baru : LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str_lusa})"
08395 |
08396 |         # =====
08397 |
08398 |         moonset_str = get_hms_str(t_moonset, tz) if t_moonset is not None else '--.--'
08399 |         lat_str = format_angle(lat).replace("+", "").replace("-", "-")
08400 |         lon_str = format_angle(lon).replace("+", "")
08401 |
08402 |         hari_idx = int(t0.whole % 7)
08403 |         nama_hari_tampil = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"][hari_idx]
08404 |         tahun_str = f"{year} CE" if year > 0 else f"{abs(year-1)} BCE (SM)"
08405 |         display_date = f"{nama_hari_tampil}, {day:02d}/{month:02d}/{tahun_str}"
08406 |
08407 |         nama_kota = getattr(self, 'opt_city', None)
08408 |         nama_kota = nama_kota.get() if nama_kota else "Lokasi Anda"
08409 |         nama_prov = getattr(self, 'opt_prov', None)
08410 |         nama_prov = nama_prov.get() if nama_prov else ""
08411 |
08412 |         # --- PERBAIKAN FORMATTING: ALIGNMENT TITIK DUA (: <45) ---
08413 |         label_1 = "1. Penentuan KHGT (PKG 1 & PKG 2)"
08414 |         label_2 = "2. Neo MABIMS (T.Alt>=3°, G.Eln>=6.4°)"
08415 |         label_3 = "3. Ilyas Criterion (T.Alt>=4°, G.Eln>=10.5°)"
08416 |         label_4 = "4. Danjon Limit (G.Eln >= 7.0°)"
08417 |         label_5 = f"5. Yallop (1998) (q-Value = {q_yallop:+.3f})"
08418 |
08419 |         # Perbarui Format Output Laporan
08420 |         report = f""{self.get_header(85)}
08421 |
08422 |         * Settings:-
08423 |         - Calculations for Waxing Crescent (New, Evening).
08424 |         - Crescent Visibility on: {display_date}
08425 |         - Calculations are Done at Time: {waktu_kalkulasi_str}
08426 |         - Calculations are Geocentric & Topocentric.
08427 |         - Refraction Model: Sæmundsson (1986)
08428 |         - Local Weather : {temp}°C, {pres}mb, {hum}% Hum.
08429 |         - LOKASI: {nama_kota}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz})
08430 |         =====
08431 |
08432 |         - G. Conjunction (UTC) : {str_ijtima_utc}
08433 |         - L. Conjunction (LT) : {str_ijtima}
08434 |         - Julian Date at Time of Calculations: {t_eval_tt:.5f}
08435 |
08436 |         - {"Sunset":<22}: {(get_hms_str(t_sunset, tz) + " LT"):<18} {"G. Moon Age":<22}: {format_time_hms(moon_age_hours):<18}
08437 |         - {"Moonset":<22}: {(moonset_str + " LT"):<18} {"Moon Lag Time":<22}: {format_time_hms(lag_time_hours):<18}
08438 |
08439 |         - G. Moon Altitude : {format_angle(alt_moon_geo):<16} T. Moon Altitude : {format_angle(alt_moon_topo):<16} Refraction (Local) : {ref_moon:.4f}°

```

```

08440 |         - G. Sun Altitude      : {format_angle(alt_sun_geo):<16} T. Sun Altitude      : {format_angle(alt_sun_topo):<16} Refraction (Local) : {ref_sun:.4f}"
08441 |
08442 |         - G. Elongation        : {format_angle(elongation):<16} G. Phase Angle      : {format_angle(phase_angle):<16}
08443 |         - G. Illumination      : {f"{illumination:05.4f} %":<16} G. Distance        : {f"{dist_km:,.2f} Km":<16}
08444 |
08445 |         -----
08446 |         [ MULTI-CRITERIA GLOBAL RESEARCH RESULT ]
08447 |         -----
08448 |         {label_1:<45}: {status_khgt_global}
08449 |         {label_2:<45}: {str_mabims}
08450 |         {label_3:<45}: {str_ilyas}
08451 |         {label_4:<45}: {str_danjon}
08452 |         {label_5:<45}: {yallop_stat}
08453 |
08454 |         =====
08455 |         ""
08456 |         return report
08457 |
08458 |     def calculate_visibility(self):
08459 |         try:
08460 |             # =====
08461 |             # TAMBAHAN API CUACA
08462 |             # =====
08463 |             # Ubah "self.entry_vlat" dan "self.entry_vlon" sesuai nama kotak input di aplikasi Anda
08464 |             # (misal: self.entry_lat atau self.lat_entry)
08465 |             try:
08466 |                 lat = float(self.entry_vlat.get())
08467 |                 lon = float(self.entry_vlon.get())
08468 |
08469 |                 # Set status sedang memuat
08470 |                 self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...", text_color="#00E5FF"))
08471 |
08472 |                 # Panggil API dan Update UI
08473 |                 status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)
08474 |                 self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca, text_color=warna_cuaca))
08475 |             except Exception as e_cuaca:
08476 |                 self.after(0, lambda: self.lbl_status_cuaca.configure(text="❌ Gagal memuat cuaca (Periksa input koordinat)", text_color="#9E9E9E"))
08477 |             # =====
08478 |
08479 |             # --- KODE ASLI ANDA ---
08480 |             report = self.generate_visibility_report()
08481 |             self.after(0, self.display_result, report)
08482 |
08483 |         except Exception as e:
08484 |             import traceback
08485 |             self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
08486 |
08487 |     def calculate_moonphase(self):
08488 |         try:
08489 |             tahun = int(self.entry_moon_year.get())
08490 |             self.auto_switch_ephemeris(tahun)
08491 |
08492 |             # --- TAMBAHAN PENGAMAN BATAS EPHEMERIS (ANTI-ERROR 0) ---
08493 |             # Jika tahun 0, mulai dari akhir Januari sesuai batas DE406
08494 |             if tahun == 0:
08495 |                 t0 = self.ts.utc(0, 1, 30)
08496 |             else:
08497 |                 t0 = self.ts.utc(tahun, 1, 1)
08498 |
08499 |             # Jika tahun 3000, jangan sampai lewat dari bulan Mei
08500 |             if tahun == 3000:
08501 |                 t1 = self.ts.utc(3000, 5, 5)
08502 |             else:
08503 |                 t1 = self.ts.utc(tahun, 12, 31, 23, 59, 59)
08504 |             # -----
08505 |
08506 |             t_phases, y_phases = almanac.find_discrete(t0, t1, almanac.moon_phases(self.eph))
08507 |
08508 |             tz_offset = 7.0
08509 |             rows = []
08510 |             current_row = ["", "", "", ""]

```

```

08511 |
08512 |     # Pastikan t_phases tidak kosong
08513 |     if t_phases is not None:
08514 |         for t_obj, phase_idx in get_safe_events(t_phases, y_phases):
08515 |             t_lokal = self.ts.tt_jd(t_obj.tt + (tz_offset / 24.0))
08516 |             y_l, m_l, d_l, h_l, min_l, s_l = t_lokal.utc
08517 |
08518 |             tahun_str = f"{int(y_l)} M" if y_l > 0 else f"{abs(int(y_l)-1)} SM"
08519 |             date_str = f"{int(d_l):02d}/{int(m_l):02d}/{tahun_str} {int(h_l):02d}.{int(min_l):02d}"
08520 |
08521 |             if current_row[phase_idx] != "":
08522 |                 rows.append(current_row)
08523 |                 current_row = ["", "", "", ""]
08524 |                 current_row[phase_idx] = date_str
08525 |
08526 |         if any(current_row):
08527 |             rows.append(current_row)
08528 |
08529 |         tahun_header = f"{tahun} M" if tahun > 0 else f"{abs(tahun-1)} SM"
08530 |         info_batas = " (Mulai 30 Jan)" if tahun == 0 else ""
08531 |
08532 |         header = f"""{self.get_header(103)}
08533 | [{" Kalkulator Fase Bulan }".center(103)}
08534 |
08535 | * Settings:-
08536 | - Geocentric Phases of The Moon for The Year: {tahun_header}{info_batas}
08537 | - Time Reference is Local WIB (UTC+7)
08538 | - Ephemeris segment covers: 0-01-29 to 3000-05-06
08539 |
08540 | =====
08541 | {"New Moon".center(25)}{"First Quarter".center(25)}{"Full Moon".center(25)}{"Last Quarter".center(25)}
08542 | """
08543 |         output_lines = [header]
08544 |         for r in rows:
08545 |             col0 = r[0].center(25) if r[0] else " " * 25
08546 |             col1 = r[1].center(25) if r[1] else " " * 25
08547 |             col2 = r[2].center(25) if r[2] else " " * 25
08548 |             col3 = r[3].center(25) if r[3] else " " * 25
08549 |             output_lines.append(f" {col0}{col1}{col2}{col3}")
08550 |
08551 |         footer = f"""
08552 | =====
08553 | * Remarks:-
08554 | - Date format: dd/mm/yyyy.
08555 | - Calculated using Python Skyfield & JPL Ephemeris ({self.ephemeris_name}).
08556 | """
08557 |         output_lines.append(footer)
08558 |         self.after(0, self.display_result, "\n".join(output_lines))
08559 |
08560 |     except Exception as e:
08561 |         import traceback
08562 |         self.after(0, self.display_error, f"Kesalahan Rentang Data: {str(e)}\n\n(Catatan: File ephemeris Anda hanya mendukung hingga 29 Jan 0 SM)")
08563 |
08564 |     def calculate_ephemeris(self):
08565 |         try:
08566 |             import datetime
08567 |             import pytz
08568 |             import math
08569 |
08570 |             obj_target = self.combo_eph_obj.get()
08571 |             time_ref = self.combo_eph_tref.get()
08572 |             loc_ref = self.combo_eph_lref.get()
08573 |
08574 |             # 1. Ambil Input (Aman untuk tahun minus)
08575 |             sy, sm, sd = int(self.entry_eph_sy.get()), int(self.entry_eph_smon.get()), int(self.entry_eph_sd.get())
08576 |             sh, smin, ssec = int(self.entry_eph_sh.get()), int(self.entry_eph_smin.get()), int(self.entry_eph_ssec.get())
08577 |
08578 |             ey, em, ed = int(self.entry_eph_ey.get()), int(self.entry_eph_emon.get()), int(self.entry_eph_ed.get())
08579 |             eh, emin, esec = int(self.entry_eph_eh.get()), int(self.entry_eph_emin.get()), int(self.entry_eph_esec.get())
08580 |
08581 |             step_val = float(self.entry_eph_step.get())

```

```

08582 |         step_unit = self.combo_eph_step.get()
08583 |
08584 |         lat = float(self.entry_eph_lat.get())
08585 |         lon = float(self.entry_eph_lon.get())
08586 |         elev = float(self.entry_eph_elev.get())
08587 |         tz = float(self.entry_eph_tz.get())
08588 |
08589 |         # Auto-Switch Ephemeris berdasarkan tahun awal
08590 |         self.auto_switch_ephemeris(sy)
08591 |         earth = self.eph['earth']
08592 |
08593 |         if obj_target == "Sun":
08594 |             target = self.eph['sun']
08595 |             radius_km = 696000.0
08596 |         else:
08597 |             target = self.eph['moon']
08598 |             radius_km = 1737.4
08599 |
08600 |         loc = wgs84.latlon(lat, lon, elevation_m=elev)
08601 |
08602 |         # =====
08603 |         # 2. PERBAIKAN: KONVERSI WAKTU LOKAL KE JULIAN DATE DENGAN TREF
08604 |         # =====
08605 |         # Ubah input user (Lokal) menjadi UTC murni dengan mengurangi TZ
08606 |         dt_start_local = datetime.datetime(sy, sm, sd, sh, smin, ssec)
08607 |         dt_end_local = datetime.datetime(ey, em, ed, eh, emin, esec)
08608 |
08609 |         dt_start_base = dt_start_local - datetime.timedelta(hours=tz)
08610 |         dt_end_base = dt_end_local - datetime.timedelta(hours=tz)
08611 |
08612 |         # Eksekusi TREF (Terrestrial Dynamical Time vs Universal Time)
08613 |         if "TDT" in time_ref or "TT" in time_ref:
08614 |             t_start = self.ts.tt(dt_start_base.year, dt_start_base.month, dt_start_base.day,
08615 |                                 dt_start_base.hour, dt_start_base.minute, dt_start_base.second)
08616 |             t_end = self.ts.tt(dt_end_base.year, dt_end_base.month, dt_end_base.day,
08617 |                               dt_end_base.hour, dt_end_base.minute, dt_end_base.second)
08618 |         else:
08619 |             # Default UT1/UTC
08620 |             t_start = self.ts.utc(dt_start_base.year, dt_start_base.month, dt_start_base.day,
08621 |                                  dt_start_base.hour, dt_start_base.minute, dt_start_base.second)
08622 |             t_end = self.ts.utc(dt_end_base.year, dt_end_base.month, dt_end_base.day,
08623 |                                dt_end_base.hour, dt_end_base.minute, dt_end_base.second)
08624 |
08625 |         # Pengaman batas bawah ephemeris de406 (0-01-29)
08626 |         jd_limit = self.ts.utc(0, 1, 29, 12).tt
08627 |         current_jd = max(t_start.tt, jd_limit)
08628 |         end_jd = t_end.tt
08629 |
08630 |         if step_unit == "Day": step_jd = step_val
08631 |         elif step_unit == "Hour": step_jd = step_val / 24.0
08632 |         elif step_unit == "Minute": step_jd = step_val / 1440.0
08633 |         else: step_jd = step_val / 86400.0
08634 |
08635 |         output_lines = []
08636 |         lat_str = format_angle(lat).replace("+", "").replace("-", "-")
08637 |         lon_str = format_angle(lon).replace("+", "")
08638 |
08639 |         try:
08640 |             nama_kota = self.opt_city.get()
08641 |             nama_prov = self.opt_prov.get()
08642 |             lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz:.2f})"
08643 |         except:
08644 |             lokasi_text = f"CUSTOM LOCATION (Long: {lon_str}, Lat: {lat_str}, Ele:{elev}, Zone:{tz:.2f})"
08645 |
08646 |         header = f""{self.get_header(128)}
08647 | {"[ Sun & Moon Ephemeris ]".center(128)}
08648 |
08649 |         * Settings:-
08650 |         - {obj_target} Ephemeris
08651 |         - LOKASI: {lokasi_text}
08652 |         - Ephemeris are {loc_ref}. Time Reference: {time_ref}.

```

```

08653 | =====
08654 |
08655 | Time & Date      R. A.      Dec.      Altitude  Azimuth  Latitude Longitude Distance      SD      Parallax  DT
08656 | """
08657 |     output_lines.append(header)
08658 |
08659 |     loop_count = 0
08660 |     while current_jd <= end_jd:
08661 |         loop_count += 1
08662 |         if loop_count > 2000: # Batas iterasi keamanan
08663 |             output_lines.append("\n[Batas maksimum iterasi tercapai. Ekspor ke CSV jika butuh data lebih dari 2000 baris]")
08664 |             break
08665 |
08666 |         t = self.ts.tt_jd(current_jd)
08667 |
08668 |         geo_apparent = earth.at(t).observe(target).apparent()
08669 |         geo_dist_km = geo_apparent.distance().km
08670 |         hp_deg = math.degrees(math.asin(6378.137 / geo_dist_km))
08671 |
08672 |         if loc_ref == "Topocentric":
08673 |             astrometric = (earth + loc).at(t).observe(target)
08674 |             apparent = astrometric.apparent()
08675 |             ra, dec, distance = apparent.radec(epoch=t) # Ditambahkan epoch=t agar J2000 dinamis ke Apparent of Date
08676 |             # Ephemeris murni umumnya Geometric (tanpa refraksi atmosfer)
08677 |             alt_obj, az_obj, _ = apparent.altaz()
08678 |             alt_deg, az_deg = alt_obj.degrees, az_obj.degrees
08679 |         else:
08680 |             apparent = geo_apparent
08681 |             ra, dec, distance = apparent.radec(epoch=t)
08682 |             gast = t.gast
08683 |             lst_deg = (gast * 15.0) + lon
08684 |             ha_rad = math.radians(lst_deg - (ra.hours * 15.0))
08685 |             dec_r = dec.radians
08686 |             lat_r = math.radians(lat)
08687 |             sin_alt = math.sin(dec_r)*math.sin(lat_r) + math.cos(dec_r)*math.cos(lat_r)*math.cos(ha_rad)
08688 |             alt_deg = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
08689 |             az_deg = (math.degrees(math.atan2(-math.sin(ha_rad), math.tan(dec_r)*math.cos(lat_r)-math.sin(lat_r)*math.cos(ha_rad))) + 360) % 360
08690 |
08691 |         lat_ecl, lon_ecl, _ = apparent.ecliptic_latlon(epoch=t)
08692 |         sd_deg = math.degrees(math.asin(radius_km / distance.km))
08693 |
08694 |         # =====
08695 |         # 3. PERBAIKAN: FORMAT PRINT WAKTU KEMBALI KE LOKAL DENGAN AMAN
08696 |         # =====
08697 |         dt_utc_out = t.utc_datetime()
08698 |         dt_display = dt_utc_out + datetime.timedelta(hours=tz)
08699 |
08700 |         time_str = f"{dt_display.hour:02d}.{dt_display.minute:02d}.{dt_display.second:02d} {dt_display.day:02d}/{dt_display.month:02d}/{format_tahun_aman(dt_display.year)}"
08701 |
08702 |         ra_str = format_eph_angle(ra.hours * 15.0, is_ra=True)
08703 |         dec_str = format_eph_angle(dec.degrees)
08704 |         alt_str = format_eph_angle(alt_deg)
08705 |         az_str = format_eph_angle(az_deg)
08706 |         lat_e_str = format_eph_angle(lat_ecl.degrees)
08707 |         lon_e_str = format_eph_angle(lon_ecl.degrees)
08708 |         dist_str = f"{distance.km:.1f}".replace(".", ",")
08709 |         sd_str = format_eph_angle(sd_deg, is_sd=True)
08710 |         hp_str = format_eph_angle(hp_deg, is_sd=True)
08711 |         dt_str = f"{t.delta_t:.1f}".replace(".", ",")
08712 |
08713 |         line = f"{time_str:<21} {ra_str} {dec_str:>9} {alt_str:>9} {az_str:>9} {lat_e_str:>9} {lon_e_str:>9} {dist_str:>11} {sd_str} {hp_str} {dt_str:>4}"
08714 |         output_lines.append(line)
08715 |
08716 |         if loop_count % 4 == 0: output_lines.append("")
08717 |         current_jd += step_jd
08718 |
08719 |     output_lines.append("=" * 128)
08720 |     output_lines.append("* Remarks:")
08721 |     output_lines.append("- Calculated using JPL Ephemeris NASA.")
08722 |     output_lines.append("- Altitude data is purely geometric (does not include atmospheric refraction).")
08723 |     self.after(0, self.display_result, "\n".join(output_lines))

```



```

08724 |
08725 |     except Exception as e:
08726 |         import traceback
08727 |         self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
08728 |
08729 | def calculate_qiblah(self):
08730 |     try:
08731 |         year = int(self.entry_qyear.get())
08732 |         month = int(self.entry_qmonth.get())
08733 |         day = int(self.entry_qday.get())
08734 |         target_date = datetime.date(year, month, day)
08735 |
08736 |         self.auto_switch_ephemeris(year)
08737 |
08738 |         lat = float(self.entry_qlat.get())
08739 |         lon = float(self.entry_qlon.get())
08740 |         tz = float(self.entry_qtz.get())
08741 |         mode_qiblah = self.radio_qiblah_var.get()
08742 |
08743 |         phi_k = math.radians(21.4225)
08744 |         lam_k = math.radians(39.8262)
08745 |         phi = math.radians(lat)
08746 |         lam = math.radians(lon)
08747 |
08748 |         y_val = math.sin(lam_k - lam)
08749 |         x_val = math.cos(phi) * math.tan(phi_k) - math.sin(phi) * math.cos(lam_k - lam)
08750 |         qiblah_angle = math.degrees(math.atan2(y_val, x_val))
08751 |         qiblah_angle = (qiblah_angle + 360.0) % 360.0
08752 |
08753 |         target_azimuth = qiblah_angle
08754 |         if mode_qiblah == "shadow":
08755 |             target_azimuth = (qiblah_angle + 180.0) % 360.0
08756 |
08757 |         earth = self.eph['earth']
08758 |         sun = self.eph['sun']
08759 |         loc = wgs84.latlon(lat, lon, elevation_m=0.0)
08760 |
08761 |         t0 = self.ts.utc(target_date.year, target_date.month, target_date.day, -int(tz))
08762 |         t1 = self.ts.utc(target_date.year, target_date.month, target_date.day, 24 - int(tz))
08763 |
08764 |         tt_array = np.linspace(t0.tt, t1.tt, 1440)
08765 |         t_search = self.ts.tt_jd(tt_array)
08766 |
08767 |         astrometric = (earth + loc).at(t_search).observe(sun)
08768 |         apparent = astrometric.apparent()
08769 |         alt_arr, az_arr, _ = apparent.altaz()
08770 |
08771 |         az_degrees = az_arr.degrees
08772 |         alt_degrees = alt_arr.degrees
08773 |
08774 |         diffs = (az_degrees - target_azimuth + 180.0) % 360.0 - 180.0
08775 |
08776 |         crossings = []
08777 |         for i in range(len(diffs) - 1):
08778 |             if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
08779 |                 if abs(diffs[i] - diffs[i+1]) < 180.0:
08780 |                     fraction = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
08781 |                     t_cross_tt = tt_array[i] + fraction * (tt_array[i+1] - tt_array[i])
08782 |                     alt_cross = alt_degrees[i] + fraction * (alt_degrees[i+1] - alt_degrees[i])
08783 |
08784 |                     if alt_cross > 0:
08785 |                         crossings.append(self.ts.tt_jd(t_cross_tt))
08786 |
08787 |         dt_val = t0.delta_t.item() if hasattr(t0.delta_t, 'item') else t0.delta_t
08788 |
08789 |         lat_str = format_angle(lat).replace("+", "").replace("-", "-")
08790 |         lon_str = format_angle(lon).replace("+", "")
08791 |         lon_clean = lon_str.replace(":", ":").replace("'", ',0')
08792 |         lat_clean = lat_str.replace(":", ":").replace("'", ',0')
08793 |
08794 |         # ---> AMBIL NAMA KOTA & PROVINSI <---

```

```

08795 |         try:
08796 |             nama_kota = self.opt_city.get()
08797 |             nama_prov = self.opt_prov.get()
08798 |             lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_clean}, Lat: {lat_clean}, Zone:{tz:.2f})"
08799 |         except:
08800 |             lokasi_text = f"CUSTOM LOCATION (Long: {lon_clean}, Lat: {lat_clean}, Zone:{tz:.2f})"
08801 |
08802 |         mode_text = "The Time At Which the SUN is at Qiblah Direction" if mode_qiblah == "sun" else "The Time At Which the Sun's SHADOW is at Qiblah"
08803 |         qiblah_str = f"{qiblah_angle:.1f}"".replace('.', ',')
08804 |
08805 |         header = f""{self.get_header(84)}
08806 |         [{" Qiblah Direction }".center(84)}
08807 |
08808 |         * Settings:-
08809 |         - Qiblah Direction is: {qiblah_str} From True North
08810 |         - Qiblah Time you Chose is {mode_text}
08811 |         - Qiblah Direction for: {target_date.strftime('%d/%m/%Y CE')}
08812 |         - LOKASI: {lokasi_text}
08813 |         - No Summer Time.
08814 |         - Delta T: {dt_val:.1f} Second(s)
08815 |         =====
08816 |
08817 |         Date                Qiblah Time                Qiblah Time
08818 |         First Time                Second Time
08819 |         """
08820 |
08821 |         time_1 = "----"
08822 |         time_2 = "----"
08823 |
08824 |         if len(crossings) > 0:
08825 |             dt1 = crossings[0].utc_datetime() + datetime.timedelta(hours=tz)
08826 |             time_1 = dt1.strftime("%H:%M:%S")
08827 |         if len(crossings) > 1:
08828 |             dt2 = crossings[1].utc_datetime() + datetime.timedelta(hours=tz)
08829 |             time_2 = dt2.strftime("%H:%M:%S")
08830 |
08831 |         data_row = f"\n{target_date.strftime('%d/%m/%Y')}          {time_1:^10}          {time_2:^10}          \n"
08832 |
08833 |         footer = ""=====
08834 |
08835 |         * Remarks:-
08836 |         - Date format: dd/mm/yyyy.
08837 |         - The Symbol '*' before the date, refers to Summer Time.
08838 |         - '----' Means that the Sun / Sun's Shadow does not reach the Qiblah Direction on that day.
08839 |         - Kindly notice that at certain locations and on certain days, the Sun / Sun's Shadow reaches
08840 |           Qiblah direction twice on the same day!
08841 |         """
08842 |         report = header + data_row + footer
08843 |         self.after(0, self.display_result, report)
08844 |
08845 |         except Exception as e:
08846 |             import traceback
08847 |             self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
08848 |
08849 |     def open_kiblat_online(self):
08850 |         """Menu 26: buka layanan Kiblat Online hanya jika koneksi internet tersedia."""
08851 |         url = "https://kasmui.cloud/kiblat/"
08852 |
08853 |     def _check_connection_and_open():
08854 |         try:
08855 |             request = urllib.request.Request(
08856 |                 url,
08857 |                 headers={"User-Agent": "Mozilla/5.0 KHGT-Times/7.7.3"}
08858 |             )
08859 |             with urllib.request.urlopen(request, timeout=7) as response:
08860 |                 status = getattr(response, "status", 200)
08861 |                 if status >= 400:
08862 |                     raise urllib.error.URLError(f"HTTP status {status}")
08863 |
08864 |             # Buka sebagai tab baru pada browser default setelah koneksi berhasil.
08865 |             self.after(0, lambda: webbrowser.open_new_tab(url))

```

```

08866 |
08867 |         except Exception as exc:
08868 |             detail = str(exc)
08869 |             self.after(
08870 |                 0,
08871 |                 lambda msg=detail: messagebox.showerror(
08872 |                     "Koneksi Internet Diperlukan",
08873 |                     "Akses Kiblat Online memerlukan sambungan internet.\n\n"
08874 |                     "Pastikan komputer terhubung ke internet lalu coba lagi.\n\n"
08875 |                     f"Detail: {msg}"
08876 |                 )
08877 |             )
08878 |
08879 |         threading.Thread(target=_check_connection_and_open, daemon=True).start()
08880 |
08881 |     def show_qiblah_map(self):
08882 |         try:
08883 |             import math
08884 |             import numpy as np
08885 |             from PIL import Image
08886 |             from matplotlib.figure import Figure
08887 |             from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
08888 |             import customtkinter as ctk
08889 |             import os
08890 |             from tkinter import messagebox
08891 |
08892 |             map_url = "https://hisabmu.com/aifikih/berbagi/map_topografi.jpg"
08893 |             map_file = "map_topografi.jpg"
08894 |
08895 |             # Panggil fungsi unduh
08896 |             download_custom_bsp(map_file, map_url)
08897 |             full_map_path = os.path.join(BASE_DIR, map_file)
08898 |
08899 |             lons = np.linspace(-180, 180, 360)
08900 |             lats = np.linspace(-90, 90, 180)
08901 |             LONS, LATS = np.meshgrid(lons, lats)
08902 |
08903 |             phi_k = math.radians(21.4225)
08904 |             lam_k = math.radians(39.8262)
08905 |             phi = np.radians(LATS)
08906 |             lam = np.radians(LONS)
08907 |
08908 |             y = np.sin(lam_k - lam)
08909 |             x = np.cos(phi) * np.tan(phi_k) - np.sin(phi) * np.cos(lam_k - lam)
08910 |             Q = np.degrees(np.arctan2(y, x))
08911 |             Q = (Q + 360) % 360
08912 |
08913 |             # --- PERBAIKAN: BUAT JENDELA GUI BARU (Menghindari plt.show() yang membuka grafik lain) ---
08914 |             win_plot = ctk.CTkToplevel(self)
08915 |             win_plot.title("Accurate Times: Qiblah World Map")
08916 |             win_plot.geometry("1100x700")
08917 |             win_plot.attributes("-topmost", True)
08918 |
08919 |             # --- PERBAIKAN: GUNAKAN OBJECT-ORIENTED MATPLOTLIB (Mencegah bentrok dengan grafik 3D/Menu lain) ---
08920 |             fig = Figure(figsize=(11, 6), dpi=100)
08921 |             ax = fig.add_subplot(111)
08922 |
08923 |             if os.path.exists(full_map_path):
08924 |                 try:
08925 |                     img = Image.open(full_map_path)
08926 |                     ax.imshow(img, extent=[-180, 180, -90, 90], aspect='auto', alpha=1.0, zorder=0)
08927 |                     except Exception as e:
08928 |                         print("Gagal memuat gambar peta via Pillow:", e)
08929 |                 else:
08930 |                     messagebox.showwarning("Peringatan", f"File gambar {map_file} gagal diunduh atau tidak ditemukan di {full_map_path}")
08931 |
08932 |             # Arsiran warna peta kiblat
08933 |             c = ax.contourf(LONS, LATS, Q, levels=np.arange(0, 361, 10), cmap='hsv', alpha=0.4, zorder=1)
08934 |
08935 |             # Garis kontur kiblat
08936 |             ax.contour(LONS, LATS, Q, levels=np.arange(0, 361, 10), colors='black', linewidths=0.6, alpha=0.8, zorder=2)

```

```

08937 |
08938 |         # Titik Makkah
08939 |         ax.scatter(39.8262, 21.4225, color='black', marker='*', s=200, zorder=10, label='Makkah (Kaaba)')
08940 |
08941 |         ax.set_title("Accurate Times: Qiblah World Map", fontweight='bold')
08942 |         ax.set_xlabel("Longitude")
08943 |         ax.set_ylabel("Latitude")
08944 |
08945 |         ax.set_xticks(np.arange(-180, 181, 30))
08946 |         ax.set_yticks(np.arange(-80, 81, 20))
08947 |         ax.grid(True, linestyle='-', color='gray', linewidth=0.5, zorder=2)
08948 |         ax.legend(loc='upper right')
08949 |
08950 |         fig.tight_layout()
08951 |
08952 |         # --- MASUKKAN KE DALAM CANVAS TKINTER ---
08953 |         frame_canvas = ctk.CTkFrame(win_plot)
08954 |         frame_canvas.pack(fill="both", expand=True, padx=10, pady=10)
08955 |
08956 |         canvas_plot = FigureCanvasTkAgg(fig, master=frame_canvas)
08957 |         canvas_plot.draw()
08958 |         canvas_plot.get_tk_widget().pack(fill="both", expand=True)
08959 |
08960 |         # Toolbar Navigasi Matplotlib untuk Zoom/Pan/Save
08961 |         toolbar_frame = ctk.CTkFrame(win_plot, height=40, fg_color="transparent")
08962 |         toolbar_frame.pack(fill="x", side="bottom", padx=10, pady=(0, 10))
08963 |         toolbar = NavigationToolbar2Tk(canvas_plot, toolbar_frame)
08964 |         toolbar.update()
08965 |
08966 |     except Exception as e:
08967 |         from tkinter import messagebox
08968 |         messagebox.showerror("Error Map", f"Gagal memuat peta Kiblat: {e}")
08969 |
08970 |
08971 |     def calculate_moontimes(self):
08972 |         try:
08973 |             year = int(self.entry_mt_year.get())
08974 |             month = int(self.entry_mt_month.get())
08975 |
08976 |             self.auto_switch_ephemeris(year)
08977 |
08978 |             lat = float(self.entry_mt_lat.get())
08979 |             lon = float(self.entry_mt_lon.get())
08980 |             elev = float(self.entry_mt_elev.get())
08981 |             tz = float(self.entry_mt_tz.get())
08982 |
08983 |             earth, moon = self.eph['earth'], self.eph['moon']
08984 |             loc = wgs84.latlon(lat, lon, elevation_m=elev)
08985 |
08986 |             # Gunakan fungsi safe_monthrange agar kebal tahun minus
08987 |             _, num_days = safe_monthrange(year, month)
08988 |
08989 |             # --- PENGAMAN RENTANG DATA ---
08990 |             # Menghindari error batas ephemeris 0-01-29
08991 |             safe_day_start = 1
08992 |             if year == 0 and month == 1:
08993 |                 safe_day_start = 30
08994 |
08995 |             t0_month = self.ts.utc(year, month, safe_day_start, -int(tz))
08996 |             t1_month = self.ts.utc(year, month, num_days, 24 - int(tz))
08997 |
08998 |             f_rs = almanac.risings_and_settings(self.eph, moon, loc)
08999 |             t_rs, y_rs = almanac.find_discrete(t0_month, t1_month, f_rs)
09000 |
09001 |             # Inisialisasi dictionary untuk menampung kejadian per tanggal
09002 |             # Kita gunakan string "Tahun-Bulan-Hari" sebagai kunci (Key) agar aman
09003 |             events_by_date = defaultdict(lambda: {'rise': '-----', 'transit': '-----', 'set': '-----'})
09004 |
09005 |             # 1. PROSES TERBIT & TERBENAM
09006 |             if t_rs is not None:
09007 |                 for t_val, y_val in get_safe_events(t_rs, y_rs):

```

```

09008 |         # Geser Julian Date secara manual untuk mendapatkan waktu Lokal
09009 |         t_lokal = self.ts.tt_jd(t_val.tt + (tz / 24.0))
09010 |         yl, ml, dl, hl, minl, sl = t_lokal.utc
09011 |
09012 |         date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
09013 |         time_str = f"{int(hl):02d}:{int(minl):02d}"
09014 |
09015 |         if y_val == 1: # Rising
09016 |             events_by_date[date_key]['rise'] = time_str
09017 |         else: # Setting
09018 |             events_by_date[date_key]['set'] = time_str
09019 |
09020 |     # 2. PROSES TRANSIT (KULMINASI) - Menggunakan metode pencarian titik tertinggi
09021 |     for d_idx in range(safe_day_start, num_days + 1):
09022 |         t_day_start = self.ts.utc(year, month, d_idx, -int(tz))
09023 |         # Ambil 1440 sampel (per menit) dalam 24 jam untuk mencari puncak altitude
09024 |         tt_array = np.linspace(t_day_start.tt, t_day_start.tt + 1.0, 1440)
09025 |         t_search = self.ts.tt_jd(tt_array)
09026 |
09027 |         alt_arr, _, _ = (earth + loc).at(t_search).observe(moon).apparent().altaz()
09028 |         max_idx = np.argmax(alt_arr.degrees)
09029 |
09030 |         t_max = self.ts.tt_jd(tt_array[max_idx])
09031 |         # Konversi ke lokal
09032 |         t_max_lokal = self.ts.tt_jd(t_max.tt + (tz / 24.0))
09033 |         yl, ml, dl, hl, minl, sl = t_max_lokal.utc
09034 |
09035 |         date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
09036 |         events_by_date[date_key]['transit'] = f"{int(hl):02d}:{int(minl):02d}"
09037 |
09038 |     # --- PEMBUATAN LAPORAN ---
09039 |     output_lines = []
09040 |     lat_str = format_angle(lat).replace("+", "").replace("-", "-")
09041 |     lon_str = format_angle(lon).replace("+", "")
09042 |     tahun_header = format_tahun_aman(year)
09043 |
09044 |     # ---> MENGAMBIL NAMA KOTA & PROVINSI DARI SIDEBAR GUI <---
09045 |     try:
09046 |         nama_kota = self.opt_city.get()
09047 |         nama_prov = self.opt_prov.get()
09048 |         lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone: UTC{'+' if tz >= 0 else ''}{tz})"
09049 |     except:
09050 |         lokasi_text = f"CUSTOM LOCATION (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone: UTC{'+' if tz >= 0 else ''}{tz})"
09051 |
09052 |     header = f"""{self.get_header(84)}
09053 | {"[ Moon Times ]".center(84)}
09054 |
09055 | * Settings:-
09056 | - Moon Times for: {month:02d}/{tahun_header}
09057 | - LOKASI: {lokasi_text}
09058 | - Refraction is included.
09059 | =====
09060 |
09061 |     Date           Moonrise           Transit           Moonset
09062 |     ""
09063 |     output_lines.append(header)
09064 |
09065 |     for d in range(1, num_days + 1):
09066 |         date_key = f"{year}-{month}-{d}"
09067 |         ev = events_by_date.get(date_key, {'rise': '----', 'transit': '----', 'set': '----'})
09068 |
09069 |         # Format tanggal display (dd/mm/yyyy)
09070 |         date_display = f"{d:02d}/{month:02d}/{year:04d}"
09071 |         if year <= 0:
09072 |             date_display = f"{d:02d}/{month:02d}/{abs(year-1):04d} SM"
09073 |
09074 |         line = f"{date_display:<14} {ev['rise']:^12} {ev['transit']:^12} {ev['set']:^12}"
09075 |         output_lines.append(line)
09076 |
09077 |     footer = """=====
09078 | * Remarks:-

```

```

09079 | - Date format: dd/mm/yyyy.
09080 | - '----' Means that the event does not occur on that day.
09081 | - Calculated using JPL Ephemeris Engine. """
09082 |     output_lines.append(footer)
09083 |
09084 |         self.after(0, self.display_result, "\n".join(output_lines))
09085 |
09086 |     except Exception as e:
09087 |         import traceback
09088 |         self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
09089 |
09090 | def calculate_suntimes(self):
09091 |     try:
09092 |         year = int(self.entry_st_year.get())
09093 |         month = int(self.entry_st_month.get())
09094 |
09095 |         self.auto_switch_ephemeris(year)
09096 |
09097 |         lat = float(self.entry_st_lat.get())
09098 |         lon = float(self.entry_st_lon.get())
09099 |         elev = float(self.entry_st_elev.get())
09100 |         tz = float(self.entry_st_tz.get())
09101 |
09102 |         earth, target_obj = self.eph['earth'], self.eph['sun']
09103 |         loc = wgs84.latlon(lat, lon, elevation_m=elev)
09104 |
09105 |         # Gunakan fungsi safe_monthrange agar kebal tahun minus/BCE
09106 |         _, num_days = safe_monthrange(year, month)
09107 |
09108 |         # --- PENGAMAN RENTANG DATA ---
09109 |         # Menghindari error batas ephemeris NASA de406 (0-01-29)
09110 |         safe_day_start = 1
09111 |         if year == 0 and month == 1:
09112 |             safe_day_start = 30
09113 |
09114 |         t0_month = self.ts.utc(year, month, safe_day_start, -int(tz))
09115 |         t1_month = self.ts.utc(year, month, num_days, 24 - int(tz))
09116 |
09117 |         # Cari Terbit & Terbenam Matahari
09118 |         f_rs = almanac.sunrise_sunset(self.eph, loc)
09119 |         t_rs, y_rs = almanac.find_discrete(t0_month, t1_month, f_rs)
09120 |
09121 |         # Inisialisasi dictionary untuk menampung kejadian per tanggal
09122 |         # Key: "Tahun-Bulan-Hari" (String) agar aman dari limitasi datetime
09123 |         events_by_date = defaultdict(lambda: {'rise': '----', 'transit': '----', 'set': '----'})
09124 |
09125 |         # 1. PROSES SUNRISE & SUNSET
09126 |         if t_rs is not None:
09127 |             for t_val, y_val in get_safe_events(t_rs, y_rs):
09128 |                 # Geser Julian Date secara manual untuk mendapatkan waktu LOKAL
09129 |                 t_lokal = self.ts.tt_jd(t_val.tt + (tz / 24.0))
09130 |                 y_l, m_l, d_l, h_l, min_l, s_l = t_lokal.utc
09131 |
09132 |                 date_key = f"{int(y_l)}-{int(m_l)}-{int(d_l)}"
09133 |                 time_str = f"{int(h_l):02d}:{int(min_l):02d}"
09134 |
09135 |                 if y_val == 1: # Sunrise
09136 |                     events_by_date[date_key]['rise'] = time_str
09137 |                 else: # Sunset
09138 |                     events_by_date[date_key]['set'] = time_str
09139 |
09140 |         # 2. PROSES TRANSIT (ZAWAL) - Mencari titik kulminasi atas
09141 |         for d_idx in range(safe_day_start, num_days + 1):
09142 |             t_day_start = self.ts.utc(year, month, d_idx, -int(tz))
09143 |             # Sampel 1440 titik (per menit) untuk akurasi tinggi
09144 |             tt_array = np.linspace(t_day_start.tt, t_day_start.tt + 1.0, 1440)
09145 |             t_search = self.ts.tt_jd(tt_array)
09146 |
09147 |             alt_arr, _, _ = (earth + loc).at(t_search).observe(target_obj).apparent().altaz()
09148 |             max_idx = np.argmax(alt_arr.degrees)
09149 |

```

```

09150 |         t_max = self.ts.tt_jd(tt_array[max_idx])
09151 |         # Konversi ke waktu lokal
09152 |         t_max_lokal = self.ts.tt_jd(t_max.tt + (tz / 24.0))
09153 |         yl, ml, dl, hl, minl, sl = t_max_lokal.utc
09154 |
09155 |         date_key = f"{int(yl)}-{int(ml)}-{int(dl)}"
09156 |         events_by_date[date_key]['transit'] = f"{int(hl):02d}:{int(minl):02d}"
09157 |
09158 |     # --- GENERATE LAPORAN TEKS ---
09159 |     output_lines = []
09160 |     lat_str = format_angle(lat).replace("+", "").replace("-", "-")
09161 |     lon_str = format_angle(lon).replace("+", "")
09162 |     tahun_header = format_tahun_aman(year)
09163 |
09164 |     # ---> MENGAMBIL NAMA KOTA & PROVINSI DARI SIDEBAR GUI <---
09165 |     try:
09166 |         nama_kota = self.opt_city.get()
09167 |         nama_prov = self.opt_prov.get()
09168 |         lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone: UTC{'+' if tz >= 0 else ''}{tz})"
09169 |     except:
09170 |         lokasi_text = f"CUSTOM LOCATION (Long: {lon_str}, Lat: {lat_str}, Ele: {elev}m, Zone: UTC{'+' if tz >= 0 else ''}{tz})"
09171 |
09172 |     header = f"""{self.get_header(84)}"
09173 |     {"[ Sun Times ]".center(84)}
09174 |
09175 |     * Settings:-
09176 |     - Sun Times for: {month:02d}/{tahun_header}
09177 |     - LOKASI: {lokasi_text}
09178 |     - Refraction is included.
09179 |     =====
09180 |
09181 |     Date           Sunrise           Transit           Sunset
09182 |     ""
09183 |     output_lines.append(header)
09184 |
09185 |     for d in range(1, num_days + 1):
09186 |         date_key = f"{year}-{month}-{d}"
09187 |         ev = events_by_date.get(date_key, {'rise': '----', 'transit': '----', 'set': '----'})
09188 |
09189 |         # Format tampilan tanggal (Bebas crash tahun minus)
09190 |         date_display = f"{d:02d}/{month:02d}/{year:04d}"
09191 |         if year <= 0:
09192 |             # Contoh: 0 -> 3001 SM
09193 |             date_display = f"{d:02d}/{month:02d}/{abs(year-1):04d} SM"
09194 |
09195 |         line = f"{date_display:<14}    {ev['rise']:^12}    {ev['transit']:^12}    {ev['set']:^12}"
09196 |         output_lines.append(line)
09197 |
09198 |     footer = """=====
09199 |
09200 |     * Remarks:-
09201 |     - Date format: dd/mm/yyyy.
09202 |     - '----' Means that the event does not occur on that day.
09203 |     - Calculated using JPL Ephemeris NASA.""
09204 |     output_lines.append(footer)
09205 |
09206 |     self.after(0, self.display_result, "\n".join(output_lines))
09207 |
09208 |     except Exception as e:
09209 |         import traceback
09210 |         self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
09211 |
09212 |     def calculate_prayertimes(self):
09213 |         try:
09214 |             # 1. AMBIL INPUT UI (DIPERTAHANKAN UTUH)
09215 |             year = int(self.entry_pt_year.get())
09216 |             month = int(self.entry_pt_month.get())
09217 |             day = int(self.entry_pt_day.get())
09218 |
09219 |             # Engine auto-switch ephemeris
09220 |             self.auto_switch_ephemeris(year)

```

```

09221 |         lat = float(self.entry_pt_lat.get())
09222 |         lon = float(self.entry_pt_lon.get())
09223 |         elev = float(self.entry_pt_elev.get())
09224 |         tz = float(self.entry_pt_tz.get())
09225 |
09226 |         temp = float(self.entry_pt_temp.get())
09227 |         pres = float(self.entry_pt_pres.get())
09228 |         hum = float(self.entry_pt_hum.get())
09229 |
09230 |         mode_pt = self.combo_pt_mode.get()
09231 |         ikhtiyat_sec = int(self.entry_pt_ikhtiyat.get())
09232 |         fmt_pt = self.combo_pt_fmt.get()
09233 |
09234 |         # --- LOGIKA MAZHAB & METODE ASLI (DIPERTAHANKAN) ---
09235 |         mazhab_val = self.combo_pt_mazhab.get()
09236 |         method_val = self.combo_pt_method.get()
09237 |
09238 |         if "Hanafi" in mazhab_val:
09239 |             asr_factor = 2.0
09240 |             mazhab_name = "Hanafi"
09241 |         else:
09242 |             asr_factor = 1.0
09243 |             mazhab_name = "Standard (Syafi'i, Maliki, Hanbali)"
09244 |
09245 |         if "Kemenag" in method_val:
09246 |             fajr_angle, isha_angle = -20.0, -18.0
09247 |         elif "Muslim World League" in method_val:
09248 |             fajr_angle, isha_angle = -18.0, -17.0
09249 |         elif "ISNA" in method_val:
09250 |             fajr_angle, isha_angle = -15.0, -15.0
09251 |         elif "Egypt" in method_val:
09252 |             fajr_angle, isha_angle = -19.5, -17.5
09253 |         else:
09254 |             fajr_angle, isha_angle = -18.0, -18.0
09255 |
09256 |         earth = self.eph['earth']
09257 |         sun = self.eph['sun']
09258 |         loc = wgs84.latlon(lat, lon, elevation_m=elev)
09259 |
09260 |         # --- PERBAIKAN: GUNAKAN safe_monthrange UNTUK TAHUN MINUS ---
09261 |         _, num_days = safe_monthrange(year, month)
09262 |
09263 |         # Pengaman batas bawah ephemeris de406 (0-01-29)
09264 |         start_d_iter = 1
09265 |         if year == 0 and month == 1: start_d_iter = 30
09266 |
09267 |         if "Bulanan" in mode_pt:
09268 |             days_to_calc = list(range(start_d_iter, num_days + 1))
09269 |             date_info = f"month:02d}/{format_tahun_aman(year)}"
09270 |         else:
09271 |             days_to_calc = [day]
09272 |             date_info = f"day:02d}/{month:02d}/{format_tahun_aman(year)}"
09273 |
09274 |         # Ambil Delta T secara aman untuk tahun minus
09275 |         dt_val = self.ts.utc(year, month, start_d_iter).delta_t
09276 |         lat_str = format_angle(lat).replace("+", "").replace("-", "-")
09277 |         lon_str = format_angle(lon).replace("+", "")
09278 |         lon_clean = lon_str.replace(":", ":").replace("'", ',0')
09279 |         lat_clean = lat_str.replace(":", ":").replace("'", ',0')
09280 |
09281 |         # ---> AMBIL NAMA KOTA & PROVINSI UNTUK GAMBAR <---
09282 |         try:
09283 |             nama_kota = self.opt_city.get()
09284 |             nama_prov = self.opt_prov.get()
09285 |             lokasi_text = f"{nama_kota}, {nama_prov} (Long: {lon_clean}, Lat: {lat_clean}, Ele:{elev}m, Zone: UTC{'+' if tz >= 0 else ''}{tz})"
09286 |         except:
09287 |             lokasi_text = f"CUSTOM LOCATION (Long: {lon_clean}, Lat: {lat_clean}, Ele:{elev}m, Zone: UTC{'+' if tz >= 0 else ''}{tz})"
09288 |
09289 |         # --- FORMAT HEADER TABEL YANG DIPERBAIKI (ANTI BERANTAKAN) ---
09290 |         if "HH:MM:SS" in fmt_pt:
09291 |             pad = 10

```



```

09292 |         hdr_line1 = f" {'Date':<15} {'Fajer':^{pad}} {'Shuroq':^{pad}} {'Dhuha':^{pad}} {'Dhohur':^{pad}} {'Aser':^{pad}} {'Maghreb':^{pad}} {'Isha':^{pad}} {'Midnight':^15}"
09293 |         hdr_line2 = f" {'':<15} {'B. Twi.':^{pad}} {'Sunrise':^{pad}} {'+4.5°':^{pad}} {'Transit':^{pad}} {'-----':^{pad}} {'Sunset':^{pad}} {'E. Twi.':^{pad}} {'(Mid/Last
1/3)':^15}"
09294 |         sep = "=" * 118
09295 |     else:
09296 |         pad = 8
09297 |         hdr_line1 = f" {'Date':<15} {'Fajer':^{pad}} {'Shuroq':^{pad}} {'Dhuha':^{pad}} {'Dhohur':^{pad}} {'Aser':^{pad}} {'Maghreb':^{pad}} {'Isha':^{pad}} {'Midnight':^15}"
09298 |         hdr_line2 = f" {'':<15} {'B. Twi.':^{pad}} {'Sunrise':^{pad}} {'+4.5°':^{pad}} {'Transit':^{pad}} {'-----':^{pad}} {'Sunset':^{pad}} {'E. Twi.':^{pad}} {'(Mid/Last
1/3)':^15}"
09299 |         sep = "=" * 102
09300 |
09301 |         report_lines = []
09302 |         header_text = f""{self.get_header(len(sep))}
09303 | {"[ Islamic Prayer Times ]".center(len(sep))}
09304 |
09305 | * Settings:-
09306 | - Prayer times for: {date_info}
09307 | - LOKASI: {lokasi_text}
09308 | - No Summer Time.
09309 | - Method: {method_val}
09310 | - Fajer Angle: {abs(fajr_angle)}°, Isha Angle: {abs(isha_angle)}°
09311 | - Refraction: Temp.: {temp} °C Pres.: {pres} mb Humidity: {hum} % Temp.Rate: 0.0065 K/m
09312 | - Ikhtiyat Time: {ikhtiyat_sec} Second(s)
09313 | - Mazhab (Asr): {mazhab_name} (Shadow multiplier: {asr_factor}x)
09314 | - Delta T: {float(dt_val):.1f} Second(s)
09315 | {sep}
09316 |
09317 | {hdr_line1}
09318 | {hdr_line2}""
09319 |     report_lines.append(header_text)
09320 |
09321 |     def find_crossing(alt_arr, tt_arr, target_alt, direction='up'):
09322 |         diffs = alt_arr - target_alt
09323 |         for i in range(len(diffs)-1):
09324 |             if direction == 'up' and diffs[i] <= 0 and diffs[i+1] > 0:
09325 |                 frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
09326 |                 return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
09327 |             elif direction == 'down' and diffs[i] >= 0 and diffs[i+1] < 0:
09328 |                 frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
09329 |                 return tt_arr[i] + frac * (tt_arr[i+1] - tt_arr[i])
09330 |         return None
09331 |
09332 |     def fmt_time(tt_val, is_shuroq=False):
09333 |         if tt_val is None: return "----".center(pad)
09334 |         offset_i = (ikhtiyat_sec / 86400.0)
09335 |         if is_shuroq:
09336 |             t_res = self.ts.tt_jd(tt_val + (tz / 24.0) - offset_i)
09337 |         else:
09338 |             t_res = self.ts.tt_jd(tt_val + (tz / 24.0) + offset_i)
09339 |
09340 |         _, _, h_f, m_f, s_f = t_res.utc
09341 |
09342 |         if "HH:MM:SS" in fmt_pt:
09343 |             return f"{int(h_f):02d}:{int(m_f):02d}:{int(s_f):02d}"
09344 |         else:
09345 |             return f"{int(h_f):02d}.{int(m_f):02d}"
09346 |
09347 | # --- LOOPING UTAMA (LOGIKA HISAB UTUH) ---
09348 | for d in days_to_calc:
09349 |     t0 = self.ts.utc(year, month, d, -int(tz))
09350 |     t1 = self.ts.utc(year, month, d, 24 - int(tz))
09351 |     tt_array = np.linspace(t0.tt, t1.tt, 2880)
09352 |     t_search = self.ts.tt_jd(tt_array)
09353 |
09354 |     alt_deg = (earth + loc).at(t_search).observe(sun).apparent().altaz(temperature_C=temp, pressure_mbar=pres)[0].degrees
09355 |
09356 |     idx_noon = np.argmax(alt_deg)
09357 |     tt_dhohur = tt_array[idx_noon]
09358 |     alt_noon = alt_deg[idx_noon]
09359 |
09360 |     alt_am, tt_am = alt_deg[:idx_noon], tt_array[:idx_noon]

```

```

09361 |         alt_pm, tt_pm = alt_deg[idx_noon:], tt_array[idx_noon:]
09362 |
09363 |         zenith_noon = 90.0 - alt_noon
09364 |         shadow_noon = math.tan(math.radians(max(0, zenith_noon)))
09365 |         shadow_asr = asr_factor + shadow_noon
09366 |         alt_asr_target = math.degrees(math.atan(1.0 / shadow_asr))
09367 |
09368 |         val_fajr = find_crossing(alt_am, tt_am, fajr_angle, 'up')
09369 |         val_shuroq = find_crossing(alt_am, tt_am, -0.833, 'up')
09370 |         val_dhuha = find_crossing(alt_am, tt_am, 4.5, 'up')
09371 |         val_asr = find_crossing(alt_pm, tt_pm, alt_asr_target, 'down')
09372 |         val_maghreb = find_crossing(alt_pm, tt_pm, -0.833, 'down')
09373 |         val_isha = find_crossing(alt_pm, tt_pm, isha_angle, 'down')
09374 |
09375 |         highlat_method = self.combo_pt_highlat.get()
09376 |
09377 |         if "Lintang Normal" not in highlat_method:
09378 |             if val_maghreb is not None and val_shuroq is not None:
09379 |                 durasi_siang = val_maghreb - val_shuroq
09380 |                 durasi_malam = 1.0 - durasi_siang
09381 |
09382 |             if "1/2 Malam" in highlat_method:
09383 |                 porsi = durasi_malam / 2.0
09384 |                 if val_isha is None: val_isha = val_maghreb + porsi
09385 |                 if val_fajr is None: val_fajr = val_shuroq - porsi
09386 |
09387 |             elif "1/7 Malam" in highlat_method:
09388 |                 porsi = durasi_malam / 7.0
09389 |                 if val_isha is None: val_isha = val_maghreb + porsi
09390 |                 if val_fajr is None: val_fajr = val_shuroq - porsi
09391 |
09392 |             elif "Proporsi Sudut" in highlat_method:
09393 |                 porsi_isha = durasi_malam * (abs(isha_angle) / 60.0)
09394 |                 porsi_fajr = durasi_malam * (abs(fajr_angle) / 60.0)
09395 |                 if val_isha is None: val_isha = val_maghreb + porsi_isha
09396 |                 if val_fajr is None: val_fajr = val_shuroq - porsi_fajr
09397 |
09398 |             elif "Aqrab al-Balad" in highlat_method:
09399 |                 if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
09400 |                 if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)
09401 |
09402 |             if val_maghreb is None or val_shuroq is None:
09403 |                 if tt_dhohur is not None:
09404 |                     if val_shuroq is None: val_shuroq = tt_dhohur - (6.0 / 24.0)
09405 |                     if val_maghreb is None: val_maghreb = tt_dhohur + (6.0 / 24.0)
09406 |                     if val_fajr is None: val_fajr = val_shuroq - (1.5 / 24.0)
09407 |                     if val_isha is None: val_isha = val_maghreb + (1.5 / 24.0)
09408 |                     if val_dhuha is None: val_dhuha = val_shuroq + (1.0 / 24.0)
09409 |                     if val_asr is None: val_asr = tt_dhohur + (3.0 / 24.0)
09410 |
09411 |             str_midnight = "----"
09412 |             if val_maghreb is not None and val_fajr is not None:
09413 |                 durasi_malam = (val_fajr + 1.0) - val_maghreb
09414 |                 t_mid = self.ts.tt_jd(val_maghreb + (durasi_malam/2.0) + (tz/24.0))
09415 |                 t_last = self.ts.tt_jd(val_maghreb + (durasi_malam*2/3.0) + (tz/24.0))
09416 |                 str_midnight = f"{int(t_mid.utc[3]):02d}:{int(t_mid.utc[4]):02d}/{int(t_last.utc[3]):02d}:{int(t_last.utc[4]):02d}"
09417 |
09418 |             str_fajr = fmt_time(val_fajr)
09419 |             str_shuroq = fmt_time(val_shuroq, is_shuroq=True)
09420 |             str_dhuha = fmt_time(val_dhuha)
09421 |             str_dhohur = fmt_time(tt_dhohur)
09422 |             str_asr = fmt_time(val_asr)
09423 |             str_maghreb = fmt_time(val_maghreb)
09424 |             str_isha = fmt_time(val_isha)
09425 |
09426 |             date_str = f"{d:02d}/{month:02d}/{format_tahun_aman(year)}"
09427 |
09428 |             line = f"  {date_str:<15}  {str_fajr:^{pad}}  {str_shuroq:^{pad}}  {str_dhuha:^{pad}}  {str_dhohur:^{pad}}  {str_asr:^{pad}}  {str_maghreb:^{pad}}  {str_isha:^{pad}}
{str_midnight:^15}"
09429 |
09430 |             report_lines.append(line)

```

```

09431 |
09432 |         report_lines.append(f"{sep}")
09433 |         report_lines.append("")
09434 |     * Remarks:-
09435 |     - Date format: dd/mm/yyyy.
09436 |     - Fajer: Beginning of Astronomical Twilight. | Shuroq: Sunrise.
09437 |     - Dhohur: Transit of Sun. | Maghreb: Sunset. | Isha: End of Astronomical Twilight.
09438 |     - Calculated using JPL NASA Engine. Supported Year 0 to 3000.""")
09439 |
09440 |         final_report = "\n".join(report_lines)
09441 |
09442 |         # --- 1. TAMPILKAN TEKS KE LAYAR GUI ---
09443 |         self.after(0, self.display_result, final_report)
09444 |
09445 |         # --- 2. GENERATE GAMBAR OTOMATIS & BUKA ---
09446 |         # Perbaikan: Gambar akan dibuat baik itu Bulanan maupun Harian
09447 |         self.after(0, lambda: self.lbl_status.configure(text="Sedang membuat gambar jadwal...", text_color="#FFAB40"))
09448 |
09449 |         # Auto-Generate background template jika file hilang
09450 |         template_path = os.path.join(BASE_DIR, "template_kosong.jpg")
09451 |         if not os.path.exists(template_path):
09452 |             try:
09453 |                 img_bg = Image.new("RGB", (2100, 2400), "#0F172A")
09454 |                 draw_bg = ImageDraw.Draw(img_bg)
09455 |                 draw_bg.rectangle([0, 0, 2100, 280], fill="#1E88E5")
09456 |                 img_bg.save(template_path)
09457 |             except Exception as e:
09458 |                 print("Gagal membuat template otomatis:", e)
09459 |
09460 |         try: nama_kota_t = self.opt_city.get()
09461 |         except: nama_kota_t = "Custom"
09462 |
09463 |         output_filename = f"Jadwal_Shalat_{nama_kota_t.replace(' ', '_')}_month:02d}_{year}.png"
09464 |         output_path = os.path.join(BASE_DIR, output_filename)
09465 |
09466 |         try:
09467 |             _util_generate_image(final_report, template_path, output_path)
09468 |
09469 |             if platform.system() == 'Windows':
09470 |                 os.startfile(output_path)
09471 |             elif platform.system() == 'Darwin':
09472 |                 subprocess.call(['open', output_path])
09473 |             else:
09474 |                 subprocess.call(['xdg-open', output_path])
09475 |
09476 |             self.after(0, lambda f=output_filename: self.lbl_status.configure(text=f"Selesai. Gambar disimpan sbg: {f}", text_color="#00E676"))
09477 |         except Exception as e_img:
09478 |             self.after(0, lambda e=e_img: messagebox.showwarning("Generate Gambar Gagal", f"Gagal membuat gambar jadwal:\n{e}"))
09479 |
09480 |         except Exception as e:
09481 |             import traceback
09482 |             self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
09483 |
09484 |     def calculate_visibility_map(self):
09485 |         if not HAS_MAP_TOOLS:
09486 |             self.after(0, self.display_error, "Fitur Peta HD dinonaktifkan.\nLibrary 'scipy' tidak ditemukan.")
09487 |             return
09488 |
09489 |         try:
09490 |             year = int(self.entry_vmyear.get())
09491 |             month = int(self.entry_vmmmonth.get())
09492 |             day = int(self.entry_vmday.get())
09493 |
09494 |             crit = self.combo_vmcrit.get()
09495 |             param_target = self.combo_vmparam.get()
09496 |
09497 |             self.auto_switch_ephemeris(year)
09498 |             earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
09499 |
09500 |             lons_coarse = []
09501 |             lats_coarse = []

```

```

09502 | alt_data = []
09503 | elong_data = []
09504 | arcv_data = []
09505 | illum_data = []
09506 |
09507 | grid_step = 5
09508 | is_khgt_map = param_target == "Interseksi (Peta KHGT 2D)"
09509 |
09510 | # Rentang Longitude: 0-360 untuk Peta KHGT 2D, -180 to 180 untuk standar
09511 | lon_range = range(0, 361, grid_step) if is_khgt_map else range(-180, 181, grid_step)
09512 |
09513 | for lat in range(-90, 91, grid_step):
09514 |     lat_rad = math.radians(lat)
09515 |
09516 |     for plot_lon in lon_range:
09517 |         if is_khgt_map:
09518 |             if plot_lon < 180:
09519 |                 real_lon = plot_lon
09520 |                 # Belahan Timur (Kiri Peta) dihitung memakai Hari Besoknya (D+1)
09521 |                 dt_calc = datetime.date(year, month, day) + datetime.timedelta(days=1)
09522 |             else:
09523 |                 real_lon = plot_lon - 360
09524 |                 # Belahan Barat (Kanan Peta) dihitung memakai Hari Input (D)
09525 |                 dt_calc = datetime.date(year, month, day)
09526 |                 calc_y, calc_m, calc_d = dt_calc.year, dt_calc.month, dt_calc.day
09527 |             else:
09528 |                 real_lon = plot_lon
09529 |                 calc_y, calc_m, calc_d = year, month, day
09530 |
09531 |         t_noon_utc = self.ts.utc(calc_y, calc_m, calc_d, 12)
09532 |         sun_geo = earth.at(t_noon_utc).observe(sun).apparent()
09533 |         _, dec_sun, _ = sun_geo.radec()
09534 |         dec_rad = dec_sun.radians
09535 |
09536 |         noon_utc_hour = 12.0 - (real_lon / 15.0)
09537 |
09538 |         cos_h = -math.tan(lat_rad) * math.tan(dec_rad)
09539 |         if cos_h < -1 or cos_h > 1:
09540 |             sunset_utc_hour = noon_utc_hour + 6.0
09541 |         else:
09542 |             h_rad = math.acos(cos_h)
09543 |             h_hours = math.degrees(h_rad) / 15.0
09544 |             sunset_utc_hour = noon_utc_hour + h_hours
09545 |
09546 |         t_sunset = self.ts.utc(calc_y, calc_m, calc_d, sunset_utc_hour)
09547 |
09548 |         obs = earth.at(t_sunset)
09549 |         s_app = obs.observe(sun).apparent()
09550 |         m_app = obs.observe(moon).apparent()
09551 |
09552 |         gmst = t_sunset.gast # Perbaikan: Gunakan GAST (Apparent), bukan GMST
09553 |         lst_deg = (gmst * 15.0) + real_lon
09554 |
09555 |         def quick_alt_geo(app_obj, l_rad):
09556 |             # Perbaikan: Tambahkan epoch=t_sunset untuk menghindari pergeseran J2000
09557 |             ra, dec_o, _ = app_obj.radec(epoch=t_sunset)
09558 |             ha_deg = lst_deg - (ra.hours * 15.0)
09559 |             d_rad = dec_o.radians
09560 |             ha_rad = math.radians(ha_deg)
09561 |             sin_alt = math.sin(d_rad) * math.sin(l_rad) + math.cos(d_rad) * math.cos(l_rad) * math.cos(ha_rad)
09562 |             return math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
09563 |
09564 |         # 1. Kalkulasi GEOSENTRIK (Untuk KHGT)
09565 |         m_alt_geo = quick_alt_geo(m_app, lat_rad)
09566 |         s_alt_geo = quick_alt_geo(s_app, lat_rad)
09567 |         elong_geo = s_app.separation_from(m_app).degrees
09568 |
09569 |         # 2. Kalkulasi TOPOSENTRIK (Untuk MABIMS, Yallop, Ilyas)
09570 |         # Menggunakan Native Skyfield agar efek Parallax & Refraksi terhitung sempurna
09571 |         loc_topo = wgs84.latlon(lat, real_lon, elevation_m=0)
09572 |         topo_obs = (earth + loc_topo).at(t_sunset)

```

```

09573 |         m_alt_topo = topo_obs.observe(moon).apparent().altaz(temperature_C=25, pressure_mbar=1010)[0].degrees
09574 |         s_alt_topo = topo_obs.observe(sun).apparent().altaz(temperature_C=25, pressure_mbar=1010)[0].degrees
09575 |
09576 |         arcv_topo = m_alt_topo - s_alt_topo
09577 |
09578 |         illum = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
09579 |         illum_val = illum.item() if hasattr(illum, 'item') else illum
09580 |
09581 |         lons_coarse.append(plot_lon)
09582 |         lats_coarse.append(lat)
09583 |
09584 |         # Simpan data Toposentrik dan Geosentrik berdasarkan mode peta
09585 |         if is_khgt_map or "KHGT" in crit or "Danjon" in crit:
09586 |             alt_data.append(m_alt_geo) # Peta KHGT murni pakai Geosentrik
09587 |             arcv_data.append(m_alt_geo - s_alt_geo)
09588 |         else:
09589 |             alt_data.append(m_alt_topo) # Peta MABIMS/Yallop murni pakai Toposentrik
09590 |             arcv_data.append(arcv_topo)
09591 |
09592 |         elong_data.append(elong_geo) # Elongasi selalu Geosentris
09593 |         illum_data.append(illum_val * 100.0)
09594 |
09595 |         data_dict = {
09596 |             'lons': np.array(lons_coarse),
09597 |             'lats': np.array(lats_coarse),
09598 |             'alt': np.array(alt_data),
09599 |             'elong': np.array(elong_data),
09600 |             'arcv': np.array(arcv_data),
09601 |             'illum': np.array(illum_data)
09602 |         }
09603 |
09604 |         self.after(0, self.show_hd_vismap, data_dict, f"{day} {calendar.month_name[month]} {year}", crit, param_target)
09605 |
09606 |     except Exception as e:
09607 |         import traceback
09608 |         self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
09609 |
09610 | def show_hd_vismap(self, plot_data, date_str, crit_name, param_target):
09611 |     try:
09612 |         points = np.column_stack((plot_data['lons'], plot_data['lats']))
09613 |         is_khgt_map = param_target == "Interseksi (Peta KHGT 2D)"
09614 |
09615 |         if is_khgt_map:
09616 |             lon_fine = np.arange(0, 360.25, 0.2)
09617 |         else:
09618 |             lon_fine = np.arange(-180, 180.25, 0.2)
09619 |
09620 |         lat_fine = np.arange(-90, 90.25, 0.2)
09621 |         LONS_fine, LATS_fine = np.meshgrid(lon_fine, lat_fine)
09622 |
09623 |         grid_alt = interp.griddata(points, plot_data['alt'], (LONS_fine, LATS_fine), method='cubic')
09624 |         grid_elong = interp.griddata(points, plot_data['elong'], (LONS_fine, LATS_fine), method='cubic')
09625 |         grid_arcv = interp.griddata(points, plot_data['arcv'], (LONS_fine, LATS_fine), method='cubic')
09626 |         grid_illum = interp.griddata(points, plot_data['illum'], (LONS_fine, LATS_fine), method='cubic')
09627 |
09628 |         grid_alt_near = interp.griddata(points, plot_data['alt'], (LONS_fine, LATS_fine), method='nearest')
09629 |         grid_alt[np.isnan(grid_alt)] = grid_alt_near[np.isnan(grid_alt)]
09630 |
09631 |         grid_elong_near = interp.griddata(points, plot_data['elong'], (LONS_fine, LATS_fine), method='nearest')
09632 |         grid_elong[np.isnan(grid_elong)] = grid_elong_near[np.isnan(grid_elong)]
09633 |
09634 |         grid_arcv_near = interp.griddata(points, plot_data['arcv'], (LONS_fine, LATS_fine), method='nearest')
09635 |         grid_arcv[np.isnan(grid_arcv)] = grid_arcv_near[np.isnan(grid_arcv)]
09636 |
09637 |         grid_illum_near = interp.griddata(points, plot_data['illum'], (LONS_fine, LATS_fine), method='nearest')
09638 |         grid_illum[np.isnan(grid_illum)] = grid_illum_near[np.isnan(grid_illum)]
09639 |
09640 |         win_plot = ctk.CTkToplevel(self)
09641 |         win_plot.title("High Resolution Crescent Map" if not is_khgt_map else "Peta Kalender Hijriah Global Tunggal (KHGT 2D)")
09642 |         win_plot.geometry("1100x750")
09643 |         win_plot.attributes("-topmost", True)

```

```

09644 |
09645 |     fig, ax = plt.subplots(figsize=(12, 6.5), dpi=110)
09646 |
09647 |     map_url = "https://hisabmu.com/aifikih/berbagi/map_topografi.jpg"
09648 |     map_file = "map_topografi.jpg"
09649 |     download_custom_bsp(map_file, map_url)
09650 |     full_map_path = os.path.join(BASE_DIR, map_file)
09651 |
09652 |     if os.path.exists(full_map_path):
09653 |         try:
09654 |             img = Image.open(full_map_path)
09655 |             img_arr = np.array(img)
09656 |
09657 |             if is_khgt_map:
09658 |                 # Potong dan geser gambar topografi agar Pasifik di tengah (0 hingga 360)
09659 |                 mid = img_arr.shape[1] // 2
09660 |                 img_shifted = np.concatenate((img_arr[:, mid:], img_arr[:, :mid]), axis=1)
09661 |                 ax.imshow(img_shifted, extent=[0, 360, -90, 90], aspect='auto', alpha=1.0, zorder=0)
09662 |             else:
09663 |                 # Format standar -180 hingga 180
09664 |                 ax.imshow(img_arr, extent=[-180, 180, -90, 90], aspect='auto', alpha=1.0, zorder=0)
09665 |         except Exception as e:
09666 |             print("Gagal memuat gambar peta:", e)
09667 |
09668 |     # =====
09669 |     # PLOTTING KHUSUS KHGT MAP (GABUNGAN 2 HARI SEAMLESS)
09670 |     # =====
09671 |     if is_khgt_map:
09672 |         grid_score = np.zeros_like(grid_alt)
09673 |         grid_score[(grid_alt >= 5.0) & (grid_elong >= 8.0)] = 1
09674 |
09675 |         cmap_inter = ListedColormap([(0, 0, 0, 0), '#00E676'])
09676 |         bounds_inter = [-0.5, 0.5, 1.5]
09677 |         norm_inter = BoundaryNorm(bounds_inter, cmap_inter.N)
09678 |
09679 |         ax.contourf(LONS_fine, LATS_fine, grid_score, cmap=cmap_inter, norm=norm_inter, alpha=0.2, zorder=2, antialiased=True)
09680 |
09681 |         cs_alt = ax.contour(LONS_fine, LATS_fine, grid_alt, levels=[5.0], colors=['#1A237E'], linewidths=1.5, linestyle='solid', zorder=3)
09682 |         ax.clabel(cs_alt, inline=True, fontsize=10, fmt='Alt 5°')
09683 |
09684 |         cs_elong = ax.contour(LONS_fine, LATS_fine, grid_elong, levels=[8.0], colors=['#D32F2F'], linewidths=1.5, linestyle='solid', zorder=3)
09685 |         ax.clabel(cs_elong, inline=True, fontsize=10, fmt='Elong 8°')
09686 |
09687 |         # Garis Pembatas Penanggalan di tengah (Bujur 180)
09688 |         ax.axvline(180, color='black', linewidth=1.5, zorder=4)
09689 |
09690 |         # Cetak Teks Tanggal di Puncak Garis
09691 |         try:
09692 |             d = int(self.entry_vmday.get())
09693 |             m = int(self.entry_vmmonth.get())
09694 |             y = int(self.entry_vmyear.get())
09695 |             dt_kanan = datetime.date(y, m, d)
09696 |             dt_kiri = dt_kanan + datetime.timedelta(days=1)
09697 |
09698 |             bulan_nama = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov", "Des"]
09699 |             lbl_kanan = f"{dt_kanan.day} {bulan_nama[dt_kanan.month]} {dt_kanan.year}"
09700 |             lbl_kiri = f"{dt_kiri.day} {bulan_nama[dt_kiri.month]} {dt_kiri.year}"
09701 |         except:
09702 |             lbl_kanan = "Hari 1"
09703 |             lbl_kiri = "Hari 2"
09704 |
09705 |         bbox_props = dict(boxstyle="square,pad=0.3", fc="white", ec="black", alpha=0.85)
09706 |         ax.text(178, 85, lbl_kiri, color='black', ha='right', va='top', fontsize=11, fontweight='bold', bbox=bbox_props, zorder=5)
09707 |         ax.text(182, 85, lbl_kanan, color='black', ha='left', va='top', fontsize=11, fontweight='bold', bbox=bbox_props, zorder=5)
09708 |
09709 |         ax.set_xlim(0, 360)
09710 |         ax.set_ylim(-75, 90) # Potong sedikit Antartika agar proporsional
09711 |         ax.set_title(f"Peta Kalender Hijriah Global Tunggal (KHGT)\nMode: Altitude (Geo), Elongasi (Geo)", fontweight='bold', pad=15)
09712 |
09713 |         # Mengubah Sumbu X menjadi Format Bujur
09714 |         ticks_x = np.arange(0, 361, 30)

```

```

09715 |         labels_x = [f"{val}° BT" if val < 180 else f"{360 - val}° BB" if val > 180 else "180°" for val in ticks_x]
09716 |
09717 |         ax.set_xticks(ticks_x)
09718 |         ax.set_xticklabels(labels_x)
09719 |         ax.set_yticks(np.arange(-60, 61, 30))
09720 |
09721 |         p_interseksi = mpatches.Patch(color='#00E676', alpha=0.2, label='Area Hijau: Memenuhi Kriteria KHGT')
09722 |         line_alt = plt.Line2D([0], [0], color='#1A237E', lw=1.5, linestyle='solid', label='Batas Altitude 5° (G)')
09723 |         line_elong = plt.Line2D([0], [0], color='#D32F2F', lw=1.5, linestyle='solid', label='Batas Elongasi 8° (G)')
09724 |         ax.legend(handles=[p_interseksi, line_alt, line_elong], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.15), fontsize='small')
09725 |
09726 |         # =====
09727 |         # RENDER UNTUK MODE PETA STANDAR LAINNYA
09728 |         # =====
09729 |         else:
09730 |             if param_target == "Kriteria Visibilitas":
09731 |                 grid_score = np.zeros_like(grid_alt)
09732 |
09733 |                 if "KHGT" in crit_name or "Diyagnet" in crit_name:
09734 |                     grid_score[(grid_alt >= 0) & (grid_arvc >= 0)] = 1
09735 |                     grid_score[(grid_alt >= 5) & (grid_elong >= 8)] = 2
09736 |                     cmap = ListedColormap(['#FF5252', '#FFFFFF', '#00E676'])
09737 |                     bounds = [-0.5, 0.5, 1.5, 2.5]
09738 |                     norm = BoundaryNorm(bounds, cmap.N)
09739 |                     cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap, norm=norm, alpha=0.2, zorder=2, antialiased=True)
09740 |
09741 |                     p1 = mpatches.Patch(color='#00E676', label='Green: Terpenuhi KHGT (Geo Alt>=5°, Geo Eln>=8°)')
09742 |                     p2 = mpatches.Patch(color='#FFFFFF', label='White: Belum Terpenuhi')
09743 |                     p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible (Bawah Ufuk)')
09744 |                     ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18), fontsize='small')
09745 |
09746 |                 elif "MABIMS" in crit_name:
09747 |                     grid_score[(grid_alt >= 0) & (grid_arvc >= 0)] = 1
09748 |                     grid_score[(grid_alt >= 3) & (grid_elong >= 6.4)] = 2
09749 |                     cmap = ListedColormap(['#FF5252', '#FFFFFF', '#00E676'])
09750 |                     bounds = [-0.5, 0.5, 1.5, 2.5]
09751 |                     norm = BoundaryNorm(bounds, cmap.N)
09752 |                     cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap, norm=norm, alpha=0.2, zorder=2, antialiased=True)
09753 |
09754 |                     p1 = mpatches.Patch(color='#00E676', label='Green: Terpenuhi Neo MABIMS (Topo Alt>=3°, Geo Eln>=6.4°)')
09755 |                     p2 = mpatches.Patch(color='#FFFFFF', label='White: Belum Terpenuhi')
09756 |                     p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible')
09757 |                     ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18), fontsize='small')
09758 |
09759 |                 elif "Ilyas" in crit_name:
09760 |                     grid_score[(grid_alt >= 0) & (grid_arvc >= 0)] = 1
09761 |                     grid_score[(grid_alt >= 4) & (grid_elong >= 10.5)] = 2
09762 |                     cmap = ListedColormap(['#FF5252', '#FFFFFF', '#00E676'])
09763 |                     bounds = [-0.5, 0.5, 1.5, 2.5]
09764 |                     norm = BoundaryNorm(bounds, cmap.N)
09765 |                     cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap, norm=norm, alpha=0.2, zorder=2, antialiased=True)
09766 |
09767 |                     p1 = mpatches.Patch(color='#00E676', label='Green: Terpenuhi Ilyas (Topo Alt>=4°, Geo Eln>=10.5°)')
09768 |                     p2 = mpatches.Patch(color='#FFFFFF', label='White: Belum Terpenuhi')
09769 |                     p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible')
09770 |                     ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18), fontsize='small')
09771 |
09772 |                 elif "Danjon" in crit_name:
09773 |                     grid_score[(grid_alt >= 0) & (grid_arvc >= 0)] = 1
09774 |                     grid_score[(grid_elong >= 7.0)] = 2
09775 |                     cmap = ListedColormap(['#FF5252', '#FFFFFF', '#00E676'])
09776 |                     bounds = [-0.5, 0.5, 1.5, 2.5]
09777 |                     norm = BoundaryNorm(bounds, cmap.N)
09778 |                     cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap, norm=norm, alpha=0.2, zorder=2, antialiased=True)
09779 |
09780 |                     p1 = mpatches.Patch(color='#00E676', label='Green: Melewati Limit Danjon (Geo Eln >= 7.0°)')
09781 |                     p2 = mpatches.Patch(color='#FFFFFF', label='White: Bawah Limit Danjon')
09782 |                     p3 = mpatches.Patch(color='#FF5252', label='Red: Impossible')
09783 |                     ax.legend(handles=[p1, p2, p3], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.18), fontsize='small')
09784 |
09785 |                 elif "Yallop" in crit_name:

```

```

09786 |         W_grid = 15.5 * (1 - np.cos(np.radians(grid_elong)))
09787 |         q_grid = grid_arvc - (11.8371 - 6.3226 * W_grid + 0.7319 * (W_grid**2) - 0.1018 * (W_grid**3))
09788 |
09789 |         grid_score[:] = 0 # F
09790 |         grid_score[(q_grid > -0.232)] = 1 # D
09791 |         grid_score[(q_grid > -0.160)] = 2 # C
09792 |         grid_score[(q_grid > -0.014)] = 3 # B
09793 |         grid_score[(q_grid > 0.216)] = 4 # A
09794 |
09795 |         cmap = ListedColormap(['#FF5252', '#2979FF', '#E040FB', '#FFD54F', '#00E676'])
09796 |         bounds = [-0.5, 0.5, 1.5, 2.5, 3.5, 4.5]
09797 |         norm = BoundaryNorm(bounds, cmap.N)
09798 |         cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap, norm=norm, alpha=0.2, zorder=2, antialiased=True)
09799 |
09800 |         p1 = mpatches.Patch(color='#FF5252', label='F: Not Visible (Topo q-Value)')
09801 |         p2 = mpatches.Patch(color='#2979FF', label='D: Only with Optical Aid (Topo q)')
09802 |         p3 = mpatches.Patch(color='#E040FB', label='C: Need Optical Aid First (Topo q)')
09803 |         p4 = mpatches.Patch(color='#FFD54F', label='B: Visible / Perfect Cond. (Topo q)')
09804 |         p5 = mpatches.Patch(color='#00E676', label='A: Easily Visible / Naked Eye (Topo q)')
09805 |         ax.legend(handles=[p1, p2, p3, p4, p5], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.22), fontsize='small')
09806 |
09807 |     elif "0deh" in crit_name:
09808 |         grid_score[:] = 1
09809 |         grid_score[(grid_arvc < 0)] = 0
09810 |         grid_score[(grid_arvc > 6.0) & (grid_elong > 8.0)] = 2
09811 |         grid_score[(grid_arvc > 8.0) & (grid_elong > 10.0)] = 3
09812 |         grid_score[(grid_arvc > 10.4) & (grid_elong > 12.0)] = 4
09813 |
09814 |         cmap = ListedColormap(['#FF5252', '#FFFFFF', '#2979FF', '#E040FB', '#00E676'])
09815 |         bounds = [-0.5, 0.5, 1.5, 2.5, 3.5, 4.5]
09816 |         norm = BoundaryNorm(bounds, cmap.N)
09817 |         cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap, norm=norm, alpha=0.2, zorder=2, antialiased=True)
09818 |
09819 |         p1 = mpatches.Patch(color='#FF5252', label='Red: Impossible (Topo V-Value)')
09820 |         p2 = mpatches.Patch(color='#FFFFFF', label='White: Not Possible (Topo V)')
09821 |         p3 = mpatches.Patch(color='#2979FF', label='Blue: Need Optical Aid (Topo V)')
09822 |         p4 = mpatches.Patch(color='#E040FB', label='Magenta: Naked Eye / Perfect Cond (Topo V)')
09823 |         p5 = mpatches.Patch(color='#00E676', label='Green: Easily Visible (Topo V)')
09824 |         ax.legend(handles=[p1, p2, p3, p4, p5], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.22), fontsize='small')
09825 |
09826 |     # PLOTTING DILAKUKAN SEKALI SAJA DI SINI (DILUAR IF-ELIF)
09827 |     # PERBAIKAN: Penambahan parameter `levels=bounds` sangat krusial!
09828 |     norm = BoundaryNorm(bounds, cmap.N)
09829 |     cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds, cmap=cmap, norm=norm, alpha=0.2, zorder=2, antialiased=True)
09830 |
09831 |     elif param_target == "Interseksi":
09832 |         grid_score = np.zeros_like(grid_alt)
09833 |         grid_score[(grid_alt >= 5.0) & (grid_elong >= 8.0)] = 1
09834 |         cmap_inter = ListedColormap([(0, 0, 0, 0), '#00E676'])
09835 |         bounds_inter = [-0.5, 0.5, 1.5]
09836 |         norm_inter = BoundaryNorm(bounds_inter, cmap_inter.N)
09837 |         cf = ax.contourf(LONS_fine, LATS_fine, grid_score, levels=bounds_inter, cmap=cmap_inter, norm=norm_inter, alpha=0.2, zorder=2, antialiased=True)
09838 |         cs_alt = ax.contour(LONS_fine, LATS_fine, grid_alt, levels=[5.0], colors=['#FFD54F'], linewidths=2.5, linestyle='solid', zorder=3)
09839 |         ax.clabel(cs_alt, inline=True, fontsize=11, fmt='Geo Alt 5°')
09840 |         cs_elong = ax.contour(LONS_fine, LATS_fine, grid_elong, levels=[8.0], colors=['#00E5FF'], linewidths=2.5, linestyle='dashed', zorder=3)
09841 |         ax.clabel(cs_elong, inline=True, fontsize=11, fmt='Geo Eln 8°')
09842 |
09843 |         p_interseksi = mpatches.Patch(color='#00E676', alpha=0.2, label='Area Pemenuhan KHGT (Geo)')
09844 |         line_alt = plt.Line2D([0], [0], color='#FFD54F', lw=2.5, linestyle='solid', label='Batas Altitude 5° (Geo)')
09845 |         line_elong = plt.Line2D([0], [0], color='#00E5FF', lw=2.5, linestyle='dashed', label='Batas Elongasi 8° (Geo)')
09846 |         ax.legend(handles=[p_interseksi, line_alt, line_elong], loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.20), fontsize='small')
09847 |
09848 |     else:
09849 |         if param_target == "Altitude Bulan":
09850 |             target_grid, cmap_name, lbl = grid_alt, 'plasma', "Altitude Bulan (°)"
09851 |         elif param_target == "Elongasi":
09852 |             target_grid, cmap_name, lbl = grid_elong, 'inferno', "Sudut Elongasi (°)"
09853 |         else:
09854 |             target_grid, cmap_name, lbl = grid_illum, 'magma', "Fraksi Iluminasi (%)"
09855 |
09856 |         contours = ax.contour(LONS_fine, LATS_fine, target_grid, cmap=cmap_name, levels=20, linewidths=1.5, alpha=1.0, zorder=3)

```



```

09857 |         ax.clabel(contours, inline=True, fontsize=12, fmt='%1.1f')
09858 |         cbar = fig.colorbar(contours, ax=ax, orientation='horizontal', fraction=0.046, pad=0.1)
09859 |         cbar.set_label(f"Legend: {lbl}", fontweight='bold')
09860 |
09861 |         if not is_khgt_map:
09862 |             ax.set_title(f"High-Res Visibility Scanner{n\crit_name} - {date_str} - Layer: {param_target}", fontweight='bold', pad=10)
09863 |             ax.set_ylim(-90, 90)
09864 |             ax.set_xlabel("Longitude")
09865 |             ax.set_ylabel("Latitude")
09866 |             ax.set_xticks(np.arange(-180, 181, 30))
09867 |             ax.set_yticks(np.arange(-60, 61, 20))
09868 |             ax.grid(True, linestyle='--', color='gray', linewidth=0.5, zorder=4)
09869 |
09870 |         fig.tight_layout()
09871 |
09872 |         # Embed ke Tkinter
09873 |         frame_canvas = ctk.CTkFrame(win_plot)
09874 |         frame_canvas.pack(fill="both", expand=True, padx=10, pady=10)
09875 |
09876 |         canvas_plot = FigureCanvasTkAgg(fig, master=frame_canvas)
09877 |         canvas_plot.draw()
09878 |         canvas_plot.get_tk_widget().pack(fill="both", expand=True)
09879 |
09880 |         toolbar_frame = ctk.CTkFrame(win_plot, height=40, fg_color="transparent")
09881 |         toolbar_frame.pack(fill="x", side="bottom", padx=10, pady=(0, 10))
09882 |
09883 |         toolbar = NavigationToolbar2Tk(canvas_plot, toolbar_frame)
09884 |         toolbar.update()
09885 |
09886 |         def simpan_gambar_kustom():
09887 |             filepath = filedialog.asksaveasfilename(
09888 |                 initialfile=f"HD_VisMap_KHGT_Seamless.png" if is_khgt_map else f"HD_VisMap_{param_target.replace(' ', '')}.png",
09889 |                 defaultextension=".png",
09890 |                 filetypes=[("PNG Image", "*.png"), ("SVG Vector", "*.svg")]
09891 |             )
09892 |             if filepath:
09893 |                 fig.savefig(filepath, dpi=300, bbox_inches='tight')
09894 |                 messagebox.showinfo("Sukses", f"Peta HD berhasil diekspor ke:\n{filepath}")
09895 |
09896 |         btn_export = ctk.CTkButton(toolbar_frame, text="Export Gambar HD", font=("Segoe UI", 12, "bold"), fg_color="#E65100", hover_color="#BF360C", command=simpan_gambar_kustom)
09897 |         btn_export.pack(side="right", padx=10)
09898 |
09899 |         self.lbl_status.configure(text=f"Render Peta HD Selesai.", text_color="#00E676")
09900 |         self.btn_hitung.configure(state="normal")
09901 |
09902 |         self.textbox.configure(state="normal")
09903 |         self.textbox.delete("1.0", "end")
09904 |         self.textbox.insert("1.0", f"[{datetime.datetime.now().strftime('%H:%M:%S')}] Pemrosesan Data Selesai.\nPeta Visualisasi Kualitas Tinggi (HD) sedang ditampilkan pada Window baru...\n\nLayer
Heatmap : {param_target}")
09905 |         self.textbox.configure(state="disabled")
09906 |
09907 |     except Exception as e:
09908 |         import traceback
09909 |         self.display_error(f"Gagal menampilkan HD Peta: {e}\n\n{traceback.format_exc()}")
09910 |
09911 |     def calculate_qiblatime(self):
09912 |         try:
09913 |             # 1. Ambil Input (Aman untuk tahun minus)
09914 |             year = int(self.entry_qtyear.get())
09915 |             month = int(self.entry_qtmonth.get())
09916 |             day = int(self.entry_qtday.get())
09917 |
09918 |             self.auto_switch_ephemeris(year)
09919 |
09920 |             lat = float(self.entry_qtlat.get())
09921 |             lon = float(self.entry_qtlon.get())
09922 |             tz = float(self.entry_qttz.get())
09923 |
09924 |             # 2. Hitung Azimut Kiblat (Rumus Spherical Trigonometry)
09925 |             phi_k, lam_k = math.radians(21.4225), math.radians(39.8262) # Ka'bah
09926 |             phi, lam = math.radians(lat), math.radians(lon)

```

```

09927 |
09928 |     y_q = math.sin(lam_k - lam)
09929 |     x_q = math.cos(phi)*math.tan(phi_k) - math.sin(phi)*math.cos(lam_k-lam)
09930 |     q_az = math.degrees(math.atan2(y_q, x_q)) % 360
09931 |     shadow_az = (q_az + 180) % 360
09932 |
09933 |     earth, sun = self.eph['earth'], self.eph['sun']
09934 |     loc = wgs84.latlon(lat, lon)
09935 |
09936 |     # 3. Tentukan Rentang Pencarian (00:00 s.d 24:00 waktu lokal dalam UTC)
09937 |     t0 = self.ts.utc(year, month, day, -int(tz))
09938 |     t1 = self.ts.utc(year, month, day, 24 - int(tz))
09939 |
09940 |     # Gunakan resolusi tinggi (setiap menit) untuk mencari persilangan azimut
09941 |     tt_array = np.linspace(t0.tt, t1.tt, 1440)
09942 |     t_search = self.ts.tt_jd(tt_array)
09943 |
09944 |     obs = (earth + loc).at(t_search).observe(sun).apparent()
09945 |     alt_arr, az_arr, _ = obs.altaz()
09946 |     az_deg_list = az_arr.degrees
09947 |     alt_deg_list = alt_arr.degrees
09948 |
09949 |     def find_crossing_times(target_az):
09950 |         # Hitung selisih sudut (handling 360 wrap around)
09951 |         diffs = (az_deg_list - target_az + 180) % 360 - 180
09952 |         results = []
09953 |         for i in range(len(diffs)-1):
09954 |             # Jika ada perubahan tanda, berarti ada persilangan
09955 |             if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
09956 |                 # Pastikan matahari di atas ufuk saat itu terjadi
09957 |                 if alt_deg_list[i] > 0:
09958 |                     frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
09959 |                     tt_res = tt_array[i] + frac * (tt_array[i+1] - tt_array[i])
09960 |
09961 |                     # Konversi ke waktu lokal
09962 |                     t_final = self.ts.tt_jd(tt_res + (tz / 24.0))
09963 |                     _, _, _, h, mn, s = t_final.utc
09964 |                     results.append(f"{int(h):02d}:{int(mn):02d}:{int(s):02d}")
09965 |         return results
09966 |
09967 |     res_sun = ", ".join(find_crossing_times(q_az)) if find_crossing_times(q_az) else "----"
09968 |     res_shd = ", ".join(find_crossing_times(shadow_az)) if find_crossing_times(shadow_az) else "----"
09969 |
09970 |     # Format Tanggal untuk Laporan (Anti Crash)
09971 |     tgl_str = f"{day:02d}/{month:02d}/{format_tahun_aman(year)}"
09972 |
09973 |     # ---> AMBIL NAMA KOTA & PROVINSI <---
09974 |     try:
09975 |         nama_kota = self.opt_city.get()
09976 |         nama_prov = self.opt_prov.get()
09977 |         lokasi_text = f"{nama_kota}, {nama_prov} (Lat {lat}, Lon {lon}, TZ {tz:.2f})"
09978 |     except:
09979 |         lokasi_text = f"CUSTOM LOCATION (Lat {lat}, Lon {lon}, TZ {tz:.2f})"
09980 |
09981 |     report = f"""{self.get_header(85)}
09982 | {"[ Rashdul Qiblah Lokal / Qibla Time ].center(85)}
09983 |
09984 | * Settings:-
09985 | - Tanggal           : {tgl_str}
09986 | - LOKASI            : {lokasi_text}
09987 | - Azimut Kiblat     : {q_az:.2f}° (dari Utara ke Timur)
09988 | - Azimut Bayangan   : {shadow_az:.2f}°
09989 | =====
09990 |
09991 | 1. WAKTU MATAHARI DI ARAH KIBLAT:
09992 |     (Saat matahari terlihat tepat di arah Ka'bah)
09993 |     >> {res_sun} LT
09994 |
09995 | 2. WAKTU BAYANGAN DI ARAH KIBLAT:
09996 |     (Saat bayangan benda tegak lurus mengarah ke Ka'bah)
09997 |     >> {res_shd} LT

```

```

09998 |
09999 | * Catatan:
10000 | - "----" berarti fenomena tidak terjadi di lokasi pada tanggal tersebut.
10001 | - Rashdul Qiblah hanya terjadi saat matahari berada di atas ufuk (>0°).
10002 | - Kalkulasi berbasis JPL Ephemeris NASA ({self.ephemeris_name}).
10003 | =====
10004 | """
10005 |         self.after(0, self.display_result, report)
10006 |
10007 |     except Exception as e:
10008 |         import traceback
10009 |         self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
10010 |
10011 |
10012 | def _generate_demo_calendar_portrait(self):
10013 |     """Merender kanvas Kalender Info Rilis lewat Pillow dengan format Portrait HD persis seperti UI"""
10014 |     import os
10015 |     import math
10016 |     import datetime
10017 |     import ephem
10018 |     import pytz
10019 |     from skyfield import almanac
10020 |     from PIL import Image, ImageDraw, ImageFont
10021 |
10022 |     # Ukuran kanvas portrait resolusi tinggi
10023 |     img = Image.new("RGB", (1400, 2000), "#0A0A10")
10024 |     draw = ImageDraw.Draw(img)
10025 |
10026 |     try:
10027 |         f_title = ImageFont.truetype("arialbd.ttf", 50)
10028 |         f_sub = ImageFont.truetype("arial.ttf", 30)
10029 |         f_day = ImageFont.truetype("arialbd.ttf", 26)
10030 |
10031 |         # Font khusus untuk data di dalam cell & footer
10032 |         f_h_large = ImageFont.truetype("arialbd.ttf", 46) # Angka Hijriah
10033 |         f_astro = ImageFont.truetype("courbd.ttf", 15) # Data Astronomi
10034 |         f_greg = ImageFont.truetype("arialbd.ttf", 18) # Angka Masehi Bawah
10035 |         f_leg = ImageFont.truetype("arial.ttf", 24) # Legenda
10036 |
10037 |         # ---> FONT KHUSUS KETERANGAN & LOKASI AGAR PAS DI KANVAS <---
10038 |         f_ket = ImageFont.truetype("arial.ttf", 20) # Keterangan Parameter
10039 |         f_loc = ImageFont.truetype("arialbd.ttf", 26) # Nama Kota & Koordinat
10040 |     except Exception:
10041 |         f_title = f_sub = f_day = f_h_large = f_astro = f_greg = f_leg = f_ket = f_loc = ImageFont.load_default()
10042 |
10043 |     # 1. Ambil Data Tanggal & Lokasi
10044 |     y = self.demo_h_year
10045 |     m = self.demo_h_month
10046 |     m_data = HIJRI_DB[y][m]
10047 |     nama_bulan, _, start_date_str, jumlah_hari = m_data
10048 |
10049 |     bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
10050 |     parts = start_date_str.split('-')
10051 |     start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
10052 |     end_date = start_date + datetime.timedelta(days=jumlah_hari-1)
10053 |
10054 |     try:
10055 |         lat_val = float(self.entry_vlat.get())
10056 |         lon_val = float(self.entry_vlon.get())
10057 |         elev_val = float(self.entry_velev.get())
10058 |         tz_val = float(self.entry_vtz.get())
10059 |         nama_kota = self.opt_city.get()
10060 |         nama_prov = self.opt_prov.get()
10061 |     except:
10062 |         lat_val, lon_val, elev_val, tz_val = -7.0667, 110.4100, 230.0, 7.0
10063 |         nama_kota, nama_prov = "Semarang", "Jawa Tengah"
10064 |
10065 |     # 2. Setup Engine Ephemeris & Skyfield
10066 |     obs_ephem = ephem.Observer()
10067 |     obs_ephem.lat, obs_ephem.lon = str(lat_val), str(lon_val)
10068 |     obs_ephem.elevation, obs_ephem.pressure, obs_ephem.temp = elev_val, 1010, 25

```

```

10069 |     matahari_ephem = ephem.Sun()
10070 |
10071 |     earth_sf, sun_sf, moon_sf = self.eph['earth'], self.eph['sun'], self.eph['moon']
10072 |
10073 |     # 3. Cetak Header
10074 |     hijri_title = f"Kalender Hijriah KHGT: {nama_bulan.upper()} {y} H"
10075 |     greg_title = f"{BULAN_MASEHI[start_date.month-1]} {start_date.year}"
10076 |     if start_date.month != end_date.month:
10077 |         greg_title += f" - {BULAN_MASEHI[end_date.month-1]} {end_date.year}"
10078 |
10079 |     draw.text((700, 80), hijri_title, font=f_title, fill="#FFD54F", anchor="mm")
10080 |     draw.text((700, 140), greg_title, font=f_sub, fill="#00E5FF", anchor="mm")
10081 |
10082 |     # 4. Konfigurasi Grid Portrait
10083 |     start_x = 85
10084 |     start_y = 230
10085 |     cell_w = 175
10086 |     cell_h = 190
10087 |     days_header = ["Ahad", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]
10088 |
10089 |     for i, d_name in enumerate(days_header):
10090 |         x0 = start_x + i * cell_w
10091 |         y0 = start_y
10092 |         x1 = x0 + cell_w - 5
10093 |         y1 = y0 + 55
10094 |         color_bg = "#FF5252" if i == 0 else ("00E676" if i == 5 else "1E88E5")
10095 |         draw.rectangle([x0, y0, x1, y1], fill=color_bg, outline=color_bg, width=2)
10096 |         draw.text(((x0+x1)/2, (y0+y1)/2), d_name, font=f_day, fill="white", anchor="mm")
10097 |
10098 |     offset = (start_date.weekday() + 1) % 7
10099 |     row, col = 0, offset
10100 |     curr_g = start_date
10101 |     grid_y_start = start_y + 65
10102 |
10103 |     # 5. Looping Hari & Kalkulasi Astronomi
10104 |     for h_day in range(1, jumlah_hari + 1):
10105 |         x0 = start_x + col * cell_w
10106 |         y0 = grid_y_start + row * cell_h
10107 |         x1 = x0 + cell_w - 5
10108 |         y1 = y0 + cell_h - 5
10109 |
10110 |         is_today = (curr_g == datetime.date.today())
10111 |         if is_today:
10112 |             b_color, b_width = "#FFEA00", 4
10113 |         else:
10114 |             b_color = "#FF5252" if col == 0 else ("00E676" if col == 5 else "#555555")
10115 |             b_width = 2 if (col == 0 or col == 5) else 1
10116 |
10117 |         draw.rectangle([x0, y0, x1, y1], fill="#0F172A", outline=b_color, width=b_width)
10118 |
10119 |         # --- KALKULASI DATA ASTRONOMI ---
10120 |         dt_noon_local = datetime.datetime(curr_g.year, curr_g.month, curr_g.day, 12, 0, 0)
10121 |         dt_noon_utc = dt_noon_local - datetime.timedelta(hours=tz_val)
10122 |         obs_ephem.date = ephem.Date(dt_noon_utc.strftime("%Y/%m/%d %H:%M:%S"))
10123 |
10124 |         try:
10125 |             waktu_sunset_ephem = obs_ephem.next_setting(matahari_ephem)
10126 |         except:
10127 |             waktu_sunset_ephem = obs_ephem.date + 0.25
10128 |
10129 |         dt_sunset_utc = waktu_sunset_ephem.datetime().replace(tzinfo=pytz.utc)
10130 |         t_sunset = self.ts.from_datetime(dt_sunset_utc)
10131 |
10132 |         geo_earth = earth_sf.at(t_sunset)
10133 |         app_moon_geo = geo_earth.observe(moon_sf).apparent()
10134 |         app_sun_geo = geo_earth.observe(sun_sf).apparent()
10135 |
10136 |         sep_deg = app_sun_geo.separation_from(app_moon_geo).degrees
10137 |         elongasi = sep_deg.item() if hasattr(sep_deg, 'item') else sep_deg
10138 |
10139 |         ra_moon, dec_moon, _ = app_moon_geo.radec(epoch=t_sunset)

```

```

10140 |         gast = t_sunset.gast
10141 |         lst_deg = (gast * 15.0) + lon_val
10142 |
10143 |         ra_h = ra_moon.hours.item() if hasattr(ra_moon.hours, 'item') else ra_moon.hours
10144 |         dec_r = dec_moon.radians.item() if hasattr(dec_moon.radians, 'item') else dec_moon.radians
10145 |         ha_rad = math.radians(lst_deg - (ra_h * 15.0))
10146 |         lat_rad = math.radians(lat_val)
10147 |
10148 |         sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) * math.cos(ha_rad)
10149 |         tinggi_hilal = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
10150 |
10151 |         try:
10152 |             prev_nm = ephemer.previous_new_moon(waktu_sunset_ephem)
10153 |             umur_bulan = waktu_sunset_ephem - prev_nm
10154 |         except:
10155 |             umur_bulan = 0.0
10156 |
10157 |         illum_val = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
10158 |         iluminasi = (illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0
10159 |
10160 |         # Format teks astronomi
10161 |         umur_str = f"{umur_bulan:.1f}" if type(umur_bulan) == float else umur_bulan
10162 |         elong_str = f"{elongasi:.1f}°"
10163 |         ilu_str = f"{iluminasi:.1f}%"
10164 |         alt_str = f"{tinggi_hilal:.1f}°"
10165 |
10166 |         # --- MENGGAMBAR KE DALAM KOTAK (CELL) ---
10167 |         # 1. Angka Hijriah & Kiri Atas
10168 |         draw.text((x0 + 10, y0 + 10), str(h_day), font=f_h_large, fill="#FFFFFF", anchor="lt")
10169 |         draw.text((x0 + 10, y0 + 65), umur_str, font=f_astro, fill="#FFFFFF", anchor="lt")
10170 |         draw.text((x0 + 10, y0 + 85), elong_str, font=f_astro, fill="#FFFFFF", anchor="lt")
10171 |
10172 |         # 2. Gambar Bulan & Kanan Atas
10173 |         moon_size = 45
10174 |         angle = (h_day / 29.53059) * 360.0
10175 |         moon_idx = int(angle) % 360
10176 |         moon_path = os.path.join(BASE_DIR, "moon", f"m{moon_idx:03d}.png")
10177 |
10178 |         if os.path.exists(moon_path):
10179 |             img_moon = Image.open(moon_path).convert("RGBA")
10180 |             img_moon = img_moon.resize((moon_size, moon_size), Image.Resampling.LANCZOS)
10181 |             mask = Image.new('L', (moon_size, moon_size), 0)
10182 |             draw_mask = ImageDraw.Draw(mask)
10183 |             draw_mask.ellipse((0, 0, moon_size - 1, moon_size - 1), fill=255)
10184 |             img_moon.putalpha(mask)
10185 |             # Tempelkan gambar bulan di pojok kanan atas kotak
10186 |             img.paste(img_moon, (int(x1 - moon_size - 8), int(y0 + 8)), img_moon)
10187 |         else:
10188 |             draw.text((x1 - 10, y0 + 10), "☐", font=f_day, fill="#FFFFFF", anchor="rt")
10189 |
10190 |         draw.text((x1 - 10, y0 + 65), ilu_str, font=f_astro, fill="#FFFFFF", anchor="rt")
10191 |         draw.text((x1 - 10, y0 + 85), alt_str, font=f_astro, fill="#FFFFFF", anchor="rt")
10192 |
10193 |         # 3. Tanggal Masehi & Indikator Event di Bawah
10194 |         m_str = f"{curr_g.day} {BULAN_MASEHI[curr_g.month-1][:3].upper()} {curr_g.year}"
10195 |         draw.text(((x0+x1)/2, y1 - 25), m_str, font=f_greg, fill="#FFD54F", anchor="mm")
10196 |
10197 |         event_color, _ = self._get_islamic_event(h_day, m + 1, curr_g.weekday())
10198 |         if event_color:
10199 |             dot_r = 7
10200 |             dot_cx, dot_cy = (x0+x1)/2, y1 - 45
10201 |             draw.ellipse([dot_cx - dot_r, dot_cy - dot_r, dot_cx + dot_r, dot_cy + dot_r], fill=event_color)
10202 |
10203 |         col += 1
10204 |         if col > 6:
10205 |             col = 0
10206 |             row += 1
10207 |         curr_g += datetime.timedelta(days=1)
10208 |
10209 |
10210 |         # =====

```

```

10211 |     # 6. MENGGAMBAR LEGENDA & KETERANGAN SECARA DINAMIS (ANTI TERPOTONG)
10212 |     # =====
10213 |
10214 |     # Posisi Y dinamis diatur berdasarkan jumlah baris di kalender saat ini
10215 |     legend_y = grid_y_start + (row + 1 if col > 0 else row) * cell_h + 30
10216 |
10217 |     draw.text((start_x, legend_y), "LEGENDA WARNA:", font=f_sub, fill="#FFD54F", anchor="lm")
10218 |
10219 |     legend_items = [
10220 |         ("FF5252", "Hari Haram Puasa / Hari Raya"),
10221 |         ("00E676", "Puasa Sunnah Utama"),
10222 |         ("00B0FF", "Puasa Ayyamul Bidh"),
10223 |         ("FFA000", "Puasa Senin Kamis"),
10224 |         ("8BC34A", "Ramadhan"),
10225 |         ("E040FB", "Hari Besar Islam")
10226 |     ]
10227 |
10228 |     ly = legend_y + 40
10229 |     lx = start_x
10230 |     for i, (c, text) in enumerate(legend_items):
10231 |         draw.ellipse([lx, ly-10, lx+20, ly+10], fill=c)
10232 |         draw.text((lx+35, ly), text, font=f_leg, fill="white", anchor="lm")
10233 |         ly += 40
10234 |         if i == 2:
10235 |             ly = legend_y + 40
10236 |             lx = start_x + 450
10237 |
10238 |     # ---> BAGIAN KETERANGAN DAN LOKASI DICETAK LENGKAP <---
10239 |     # Kita tempatkan elemen footer menyesuaikan akhir dari letak legenda (dinamis)
10240 |     footer_y = legend_y + 175
10241 |
10242 |     # 1. Teks Keterangan Parameter (Teks Lengkap)
10243 |     teks_ket = "* Kiri: Umur Bulan (Atas), Elongasi Geo (Bawah) | Kanan: Fraksi Iluminasi (Atas), Tinggi Hilal Geo (Bawah). Dihitung saat Maghrib."
10244 |     draw.text((700, footer_y), teks_ket, font=f_ket, fill="FFFFFF", anchor="mm")
10245 |
10246 |     # 2. Teks Info Lokasi (Kota dan Koordinat)
10247 |     ket_lokasi = f"Kota: {nama_kota}, {nama_prov} | Koordinat: (Lat: {lat_val}, Lon: {lon_val})"
10248 |     draw.text((700, footer_y + 45), ket_lokasi, font=f_loc, fill="#00E5FF", anchor="mm")
10249 |
10250 |     # 3. Copyright / Watermark
10251 |     draw.text((700, footer_y + 105), "KHGT Times Engine V8.0 - By Kasmui", font=f_leg, fill="#555555", anchor="mm")
10252 |
10253 |     return img
10254 |
10255 | def export_demo_kalender(self, format_type):
10256 |     """Fungsi pembantu untuk trigger simpan PDF atau PNG khusus Kalender Rilis"""
10257 |     img = self._generate_demo_calendar_portrait()
10258 |
10259 |     y = self.demo_h_year
10260 |     m = self.demo_h_month
10261 |     nama_bulan = HIJRI_DB[y][m][0]
10262 |
10263 |     default_file = f"Kalender_KHGT_{nama_bulan}_{y}H_Portrait"
10264 |
10265 |     if format_type == "png":
10266 |         filepath = filedialog.asksaveasfilename(
10267 |             initialfile=f"{default_file}.png",
10268 |             defaulttextextension=".png",
10269 |             filetypes=[("PNG Image", "*.png")]
10270 |         )
10271 |         if filepath:
10272 |             try:
10273 |                 img.save(filepath, "PNG")
10274 |                 messagebox.showinfo("Sukses", f"Kalender berhasil diekspor menjadi Gambar PNG Portrait di:\n{filepath}")
10275 |             except Exception as e:
10276 |                 messagebox.showerror("Error", f"Gagal menyimpan PNG: {e}")
10277 |
10278 |     elif format_type == "pdf":
10279 |         filepath = filedialog.asksaveasfilename(
10280 |             initialfile=f"{default_file}.pdf",
10281 |             defaulttextextension=".pdf",

```

```

10282 |         filetypes=[("PDF Document", "*.pdf")]
10283 |     )
10284 |     if filepath:
10285 |         try:
10286 |             img.save(filepath, "PDF", resolution=150.0)
10287 |             messagebox.showinfo("Sukses", f"Kalender berhasil diekspor menjadi PDF Portrait di:\n{filepath}")
10288 |         except Exception as e:
10289 |             messagebox.showerror("Error", f"Gagal menyimpan PDF: {e}")
10290 |
10291 |     def calculate_first_point(self):
10292 |         try:
10293 |             # 1. Ambil input sebagai integer (Aman untuk tahun minus)
10294 |             y = int(self.entry_fp_y.get())
10295 |             m = int(self.entry_fp_m.get())
10296 |             d = int(self.entry_fp_d.get())
10297 |
10298 |             # --- PERBAIKAN: Panggil auto_switch_ephemeris agar tidak crash/macet ---
10299 |             self.auto_switch_ephemeris(y)
10300 |
10301 |             # 2. Bypass string, gunakan Tuple PyEphem!
10302 |             waktu_tuple = (y, m, d, 12, 0, 0)
10303 |
10304 |             # Update UI via main thread
10305 |             self.after(0, lambda: self.lbl_status.configure(text="Melacak Titik Pertama (Iterasi Multi-Hari)...", text_color="#00E5FF"))
10306 |
10307 |             matahari = ephem.Sun()
10308 |             bulan = ephem.Moon()
10309 |
10310 |             try:
10311 |                 ijtimak_1 = ephem.previous_new_moon(waktu_tuple)
10312 |                 ijtimak_2 = ephem.next_new_moon(waktu_tuple)
10313 |             except Exception:
10314 |                 self.after(0, self.display_error, "Format tanggal tidak valid. Harap periksa input.")
10315 |                 return
10316 |
10317 |             titik_hasil = []
10318 |             # Kita abaikan pulau kecil / wilayah timur jauh yang bukan daratan utama (Mainland)
10319 |             zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu", "Maluku", "Maluku Utara", "NTT"]
10320 |
10321 |             # Loop untuk 2 siklus ijtimak
10322 |             for w_ijtimak, label_siklus in [(ijtimak_1, "Bulan Referensi"), (ijtimak_2, "Bulan Berjalan/Depan")]:
10323 |
10324 |                 # Cek dari H+0 (Hari Ijtimak) sampai maksimal H+3
10325 |                 for offset_hari in range(4):
10326 |                     kandidat_hari_ini = []
10327 |                     waktu_pencarian = ephem.Date(w_ijtimak + offset_hari)
10328 |
10329 |                     for negara, kota_dict in CITY_DB.items():
10330 |                         if negara in zona_dikecualikan:
10331 |                             continue
10332 |
10333 |                         for nama_kota, koordinat in kota_dict.items():
10334 |                             lintang, bujur = koordinat
10335 |                             pengamat = ephem.Observer()
10336 |                             pengamat.lat = str(lintang)
10337 |                             pengamat.lon = str(bujur)
10338 |                             pengamat.elevation = 0
10339 |
10340 |                             pengamat.date = waktu_pencarian
10341 |                             try:
10342 |                                 waktu_sunset = pengamat.next_setting(matahari)
10343 |                             except (ephem.AlwaysUpError, ephem.NeverUpError):
10344 |                                 continue # Lewati jika di kutub / tidak ada sunset
10345 |
10346 |                             pengamat.date = waktu_sunset
10347 |                             matahari.compute(pengamat)
10348 |                             bulan.compute(pengamat)
10349 |
10350 |                             # Kalkulasi Geo (DISELELARKAN DENGAN MENU 2 - SKYFIELD)
10351 |                             import pytz
10352 |                             t_sunset_sky = self.ts.from_datetime(waktu_sunset.datetime()).replace(tzinfo=pytz.utc))

```

```

10353 |         obs_center = self.eph['earth'].at(t_sunset_sky)
10354 |         s_app_sky = obs_center.observe(self.eph['sun']).apparent()
10355 |         m_app_sky = obs_center.observe(self.eph['moon']).apparent()
10356 |
10357 |         elong_geo = s_app_sky.separation_from(m_app_sky).degrees
10358 |
10359 |         gmst = t_sunset_sky.gast
10360 |         lst_deg = (gmst * 15.0) + bujur
10361 |         ra_m, dec_m, _ = m_app_sky.radec(epoch=t_sunset_sky)
10362 |
10363 |         ha_deg = lst_deg - (ra_m.hours * 15.0)
10364 |         ha_rad = math.radians(ha_deg)
10365 |         lat_rad = math.radians(lintang)
10366 |         d_rad = dec_m.radians
10367 |
10368 |         sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
10369 |         alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
10370 |
10371 |         if alt_geo >= 5.0 and elong_geo >= 8.0:
10372 |             kandidat_hari_ini.append({
10373 |                 'kota': nama_kota,
10374 |                 'negara': negara,
10375 |                 'lat': lintang,
10376 |                 'lon': bujur,
10377 |                 'sunset_utc': waktu_sunset,
10378 |                 'alt_geo': alt_geo,
10379 |                 'elong_geo': elong_geo,
10380 |                 'ijtimak': w_ijtimak,
10381 |                 'label': label_siklus
10382 |             })
10383 |
10384 |         # Jika di hari ini sudah ada minimal 1 kota yang memenuhi kriteria,
10385 |         # Berhenti mencari di hari berikutnya! Ambil yang paling pertama terbenam.
10386 |         if kandidat_hari_ini:
10387 |             kandidat_hari_ini.sort(key=lambda x: x['sunset_utc'])
10388 |             titik_hasil.append(kandidat_hari_ini[0])
10389 |             break
10390 |
10391 |     if not titik_hasil:
10392 |         self.after(0, self.display_error, "Tidak ada daratan utama yang memenuhi kriteria KHGT pada 2 siklus ini.")
10393 |         return
10394 |
10395 |     self.after(0, self.show_first_point_map, titik_hasil)
10396 |
10397 | except Exception as e:
10398 |     import traceback
10399 |     self.after(0, self.display_error, f"Gagal menghitung Titik Pertama:\n{str(e)}\n\nTraceback:\n{traceback.format_exc()}")
10400 |
10401 | def show_first_point_map(self, list_titik):
10402 |     try:
10403 |         win_plot = ctk.CTkToplevel(self)
10404 |         win_plot.title("Peta Titik Pertama Visibilitas KHGT (2 Siklus) & Analisis Gisborne")
10405 |         win_plot.geometry("1100x750")
10406 |         win_plot.attributes("-topmost", True)
10407 |
10408 |         fig, ax = plt.subplots(figsize=(11, 6.5), dpi=100)
10409 |
10410 |         # Load Peta Latar Belakang
10411 |         map_url = "https://hisabmu.com/aifikih/berbagi/map_topografi.jpg"
10412 |         map_file = "map_topografi.jpg"
10413 |         download_custom_bsp(map_file, map_url)
10414 |         full_map_path = os.path.join(BASE_DIR, map_file)
10415 |
10416 |         if os.path.exists(full_map_path):
10417 |             try:
10418 |                 img = Image.open(full_map_path)
10419 |                 ax.imshow(img, extent=[-180, 180, -90, 90], aspect='auto', alpha=1.0, zorder=0)
10420 |                 ax.set_ylim(-90, 90)
10421 |             except Exception as e:
10422 |                 print("Gagal memuat gambar peta:", e)
10423 |

```



```

10424 |         # Plot semua kota dari database sebagai noktah merah
10425 |         all_lats = []
10426 |         all_lons = []
10427 |         for negara, kota_dict in CITY_DB.items():
10428 |             for nama_kota, koordinat in kota_dict.items():
10429 |                 all_lats.append(koordinat[0])
10430 |                 all_lons.append(koordinat[1])
10431 |
10432 |         ax.scatter(all_lons, all_lats, color='red', marker='o', s=8, alpha=0.6, zorder=3)
10433 |
10434 |         # Konfigurasi visual marker
10435 |         warna_bintang = ['#FFD54F', '#00E676']
10436 |         warna_tepi = ['red', 'white']
10437 |
10438 |         text_summary = "Pencarian Multi-Siklus Selesai.\n\nTitik daratan utama pertama di dunia yang masuk bulan baru KHGT:\n\n"
10439 |
10440 |         for idx, pt in enumerate(list_titik):
10441 |             c_star = warna_bintang[idx % len(warna_bintang)]
10442 |             c_edge = warna_tepi[idx % len(warna_tepi)]
10443 |
10444 |             # Plot Bintang Titik Pertama
10445 |             ax.scatter(pt['lon'], pt['lat'], color=c_star, marker='*', s=600, edgecolor=c_edge, linewidth=1.5, zorder=5)
10446 |
10447 |             tz_approx = int(self.get_tz_from_lon(pt['lon']))
10448 |             tz_str = f"UTC+{tz_approx}" if tz_approx >= 0 else f"UTC{tz_approx}"
10449 |
10450 |             # KALKULASI WAKTU LOKAL (Bypass Python Datetime, pakai Ephem Math)
10451 |             dt_local_ephem = ephem.Date(pt['sunset_utc'] + (tz_approx / 24.0))
10452 |             y_l, m_l, d_l, h_l, min_l, s_l = dt_local_ephem.tuple()
10453 |
10454 |             nama_bulan = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov", "Des"]
10455 |             tgl_lokal_str = f"{int(d_l):02d} {nama_bulan[int(m_l)]} {format_tahun_aman(int(y_l))}"
10456 |             jam_lokal_str = f"{int(h_l):02d}:{int(min_l):02d}:{int(s_l):02d}"
10457 |
10458 |             y_i, m_i, d_i, h_i, min_i, s_i = pt['ijtimak'].tuple()
10459 |             ij_jam = f"{int(h_i):02d}:{int(min_i):02d}:{int(s_i):02d}"
10460 |
10461 |             y_s, m_s, d_s, h_s, min_s, s_s = pt['sunset_utc'].tuple()
10462 |             ss_jam = f"{int(h_s):02d}:{int(min_s):02d}:{int(s_s):02d}"
10463 |
10464 |             # =====
10465 |             # EVALUASI PKG 1 & PKG 2 (TERINTEGRASI KONVERSI TANGGAL)
10466 |             # =====
10467 |             # PERBAIKAN: Bungkus operasi penambahan 1 hari ke dalam ephem.Date()
10468 |             batas_00_utc_ephem = ephem.Date(ephem.Date((int(y_l), int(m_l), int(d_l), 0, 0, 0)) + 1.0)
10469 |             bt_jam = "00:00:00"
10470 |             fajar_info = ""
10471 |
10472 |             negara_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Brasil", "Argentina", "Kolombia", "Peru", "Chili"]
10473 |             is_amerika = pt['negara'] in negara_amerika
10474 |
10475 |             import pytz
10476 |             dt_ij_utc = pt['ijtimak'].datetime().replace(tzinfo=pytz.utc)
10477 |             nm_tt_val = self.ts.from_datetime(dt_ij_utc).tt
10478 |             h_y, h_m = self.get_hijri_month_from_tt(nm_tt_val)
10479 |             hijri_bulan_str = BULAN_HIJRIAH[h_m - 1]
10480 |             hijri_tahun_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
10481 |
10482 |             # EVALUASI PKG 1 (Sunset < 00:00 UTC)
10483 |             if pt['sunset_utc'] < batas_00_utc_ephem:
10484 |                 tgl_1_mas = ephem.Date(batas_00_utc_ephem)
10485 |                 gy, gm, gd, _, _, _ = tgl_1_mas.tuple()
10486 |                 masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
10487 |                 status_ijtimak = f"Kriteria PKG 1 Terpenuhi\n(Terpenuhi sblm {bt_jam} UTC)\n\n Kesimpulan: BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
10488 |
10489 |             # EVALUASI PKG 2 (Jika PKG 1 gagal)
10490 |             else:
10491 |                 gisborne = ephem.Observer()
10492 |                 gisborne.lat = math.radians(-38.6623)
10493 |                 gisborne.lon = math.radians(178.0176)
10494 |                 gisborne.elevation = 0

```

```

10495 |         gisborne.pressure = 0
10496 |         gisborne.horizon = '-18' # Fajar (Astronomical Twilight)
10497 |         matahari = ephemeris.Sun()
10498 |
10499 |         gisborne.date = ephemeris.Date(batas_00_utc_ephem - (12.0 / 24.0))
10500 |         matahari.compute(gisborne)
10501 |
10502 |         try:
10503 |             fajar_gisborne_ephem = gisborne.next_rising(matahari, use_center=True)
10504 |             _, _, h_f, min_f, s_f = fajar_gisborne_ephem.tuple()
10505 |             fj_jam = f"{int(h_f):02d}:{int(min_f):02d}:{int(s_f):02d}"
10506 |
10507 |             fajar_info = f" Fajar Gisb. : {fj_jam} UTC\n"
10508 |
10509 |             if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
10510 |                 tgl_1_mas = ephemeris.Date(batas_00_utc_ephem)
10511 |                 gy, gm, gd, _, _, _ = tgl_1_mas.tuple()
10512 |                 masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
10513 |                 status_ijtimak = f"Kriteria PKG 2 Terpenuhi\n(Titik Amerika & Ijtima < Fajar)\n» Kesimpulan: BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
10514 |             else:
10515 |                 tgl_lusa = ephemeris.Date(batas_00_utc_ephem + 1.0)
10516 |                 gy, gm, gd, _, _, _ = tgl_lusa.tuple()
10517 |                 masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
10518 |                 alasan = "Titik bukan di Benua Amerika" if not is_amerika else "Ijtima > Terbit Fajar"
10519 |                 status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi\n({alasan})\n» Kesimpulan: LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
10520 |
10521 |         except Exception as e:
10522 |             fj_jam = "N/A"
10523 |             fajar_info = f" Fajar Gisb. : Error/N/A\n"
10524 |
10525 |             tgl_lusa = ephemeris.Date(batas_00_utc_ephem + 1.0)
10526 |             gy, gm, gd, _, _, _ = tgl_lusa.tuple()
10527 |             masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
10528 |             status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi\n(Gagal Menghitung Fajar)\n» Kesimpulan: LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
10529 |
10530 |         info_text = (
10531 |             f"TITIK KHGT ({pt['label'].upper()})\n"
10532 |             f"{'.'*45}\n"
10533 |             f"Lokasi : {pt['kota']}, {pt['negara']}\n"
10534 |             f"Maghrib : {jam_lokal_str} ({tz_str}) | {tgl_lokal_str}\n"
10535 |             f"Alt/Eln : {pt['alt_geo']:.2f}° / {pt['elong_geo']:.2f}°\n"
10536 |             f"{'.'*45}\n"
10537 |             f"Parameter Waktu (UTC):\n"
10538 |             f" Waktu Maghrib: {ss_jam} UTC\n"
10539 |             f" Batas PKG 1 : {bt_jam} UTC\n"
10540 |             f" Ijtimak : {ij_jam} UTC\n"
10541 |             f"{fajar_info}"
10542 |             f"{'.'*45}\n"
10543 |             f"{status_ijtimak}"
10544 |         )
10545 |
10546 |         props = dict(boxstyle='round,pad=0.6', facecolor='#1A1A1A', alpha=0.85, edgecolor=c_star, linewidth=1.5)
10547 |
10548 |         if idx == 0:
10549 |             text_pos = (0.02, 0.04)
10550 |             halign = 'left'
10551 |             valign = 'bottom'
10552 |             curve = "arc3,rad=0.1"
10553 |         else:
10554 |             text_pos = (0.98, 0.04)
10555 |             halign = 'right'
10556 |             valign = 'bottom'
10557 |             curve = "arc3,rad=-0.1"
10558 |
10559 |         ax.annotate(
10560 |             info_text,
10561 |             xy=(pt['lon'], pt['lat']),
10562 |             xycoords='data',
10563 |             xytext=text_pos,
10564 |             textcoords='axes fraction',
10565 |             color='white',

```

```

10566 |         fontsize=9,
10567 |         fontfamily='monospace',
10568 |         verticalalignment=valign,
10569 |         horizontalalignment=halign,
10570 |         bbox=props,
10571 |         arrowprops=dict(
10572 |             arrowstyle="->",
10573 |             connectionstyle=curve,
10574 |             color=c_star,
10575 |             lw=1.5,
10576 |             alpha=0.8
10577 |         ),
10578 |         zorder=6
10579 |     )
10580 |
10581 |     text_summary += f"[{pt['label']}] >> {pt['kota']}, {pt['negara']} (Maghrib Lokal: {tgl_lokal_str}, {jam_lokal_str})\nStatus: {status_ijtimak.replace(chr(10), ' ')}\n\n"
10582 |
10583 |     ax.set_title("Titik Awal Pemenuhan Kriteria KHGT & Komparasi Fajar Gisborne", fontweight='bold', pad=10)
10584 |     ax.set_xlabel("Longitude")
10585 |     ax.set_ylabel("Latitude")
10586 |     ax.set_xticks(np.arange(-180, 181, 30))
10587 |     ax.set_yticks(np.arange(-60, 61, 20))
10588 |     ax.grid(True, linestyle='--', color='gray', linewidth=0.5, zorder=1)
10589 |
10590 |     fig.tight_layout()
10591 |
10592 |     frame_canvas = ctk.CTkFrame(win_plot)
10593 |     frame_canvas.pack(fill="both", expand=True, padx=10, pady=10)
10594 |
10595 |     canvas_plot = FigureCanvasTkAgg(fig, master=frame_canvas)
10596 |     canvas_plot.draw()
10597 |     canvas_plot.get_tk_widget().pack(fill="both", expand=True)
10598 |
10599 |     toolbar_frame = ctk.CTkFrame(win_plot, height=40, fg_color="transparent")
10600 |     toolbar_frame.pack(fill="x", side="bottom", padx=10, pady=(0, 10))
10601 |     toolbar = NavigationToolbar2Tk(canvas_plot, toolbar_frame)
10602 |     toolbar.update()
10603 |
10604 |     self.lbl_status.configure(text=f"Pencarian Titik Pertama & Analisis Selesai", text_color="#00E676")
10605 |     self.btn_hitung.configure(state="normal")
10606 |
10607 |     self.textbox.configure(state="normal")
10608 |     self.textbox.delete("1.0", "end")
10609 |     text_summary += "Detail visual komparasi telah ditambahkan pada peta (Text Box melayang).\"
10610 |     self.textbox.insert("1.0", text_summary)
10611 |     self.textbox.configure(state="disabled")
10612 |
10613 | except Exception as e:
10614 |     import traceback
10615 |     self.display_error(f"Gagal menampilkan peta Titik Pertama: {e}\n\n{traceback.format_exc()}")
10616 |     self.textbox.insert("1.0", text_summary)
10617 |     self.textbox.configure(state="disabled")
10618 |
10619 |
10620 | def generate_first_point_report(self, y, m, d):
10621 |     waktu_tuple = (y, m, d, 12, 0, 0)
10622 |     self.auto_switch_ephemeris(y)
10623 |
10624 |     matahari = ephemeris.Sun()
10625 |     bulan = ephemeris.Moon()
10626 |
10627 |     try:
10628 |         ijtimak_1 = ephemeris.previous_new_moon(waktu_tuple)
10629 |         ijtimak_2 = ephemeris.next_new_moon(waktu_tuple)
10630 |     except Exception:
10631 |         return "Format tanggal tidak valid. Harap periksa input."
10632 |
10633 |     titik_hasil = []
10634 |     zona_dikecualikan = ["Kepulauan Pasifik (Timur Jauh)", "Kepulauan Seribu", "Maluku", "Maluku Utara", "NTT"]
10635 |
10636 |     for w_ijtimak, label_siklus in [(ijtimak_1, "Bulan Referensi"), (ijtimak_2, "Bulan Berjalan/Depan")]:

```

```

10637 |         for offset_hari in range(4):
10638 |             kandidat_hari_ini = []
10639 |             waktu_pencarian = ephemeris.Date(w_ijtimak + offset_hari)
10640 |
10641 |             for negara, kota_dict in CITY_DB.items():
10642 |                 if negara in zona_dikecualikan: continue
10643 |                 for nama_kota, koordinat in kota_dict.items():
10644 |                     lintang, bujur = koordinat
10645 |                     pengamat = ephemeris.Observer()
10646 |                     pengamat.lat, pengamat.lon = str(lintang), str(bujur)
10647 |                     pengamat.elevation = 0
10648 |                     pengamat.pressure = 1010
10649 |                     pengamat.temp = 25
10650 |                     pengamat.date = waktu_pencarian
10651 |
10652 |                     try: waktu_sunset = pengamat.next_setting(matahari)
10653 |                     except: continue
10654 |
10655 |                     pengamat.date = waktu_sunset
10656 |                     matahari.compute(pengamat)
10657 |                     bulan.compute(pengamat)
10658 |
10659 |                     import pytz
10660 |                     t_sunset_sky = self.ts.from_datetime(waktu_sunset.datetime().replace(tzinfo=pytz.utc))
10661 |                     obs_center = self.eph['earth'].at(t_sunset_sky)
10662 |                     s_app_sky = obs_center.observe(self.eph['sun']).apparent()
10663 |                     m_app_sky = obs_center.observe(self.eph['moon']).apparent()
10664 |
10665 |                     elong_geo = s_app_sky.separation_from(m_app_sky).degrees
10666 |
10667 |                     gmst = t_sunset_sky.gast
10668 |                     lst_deg = (gmst * 15.0) + bujur
10669 |                     ra_m, dec_m, _ = m_app_sky.radec(epoch=t_sunset_sky)
10670 |
10671 |                     ha_deg = lst_deg - (ra_m.hours * 15.0)
10672 |                     ha_rad = math.radians(ha_deg)
10673 |                     lat_rad = math.radians(lintang)
10674 |                     d_rad = dec_m.radians
10675 |
10676 |                     sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
10677 |                     alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
10678 |
10679 |                     if alt_geo >= 5.0 and elong_geo >= 8.0:
10680 |                         kandidat_hari_ini.append({
10681 |                             'kota': nama_kota, 'negara': negara, 'lat': lintang, 'lon': bujur,
10682 |                             'sunset_utc': waktu_sunset, 'alt_geo': alt_geo, 'elong_geo': elong_geo,
10683 |                             'ijtimak': w_ijtimak, 'label': label_siklus
10684 |                         })
10685 |
10686 |             if kandidat_hari_ini:
10687 |                 kandidat_hari_ini.sort(key=lambda x: x['sunset_utc'])
10688 |                 titik_hasil.append(kandidat_hari_ini[0])
10689 |                 break
10690 |
10691 | if not titik_hasil:
10692 |     return "Tidak ada daratan utama (Mainland) yang memenuhi kriteria KHGT pada 2 siklus ini."
10693 |
10694 | report = "Pencarian Multi-Siklus Selesai.\nTitik daratan utama pertama di dunia yang masuk bulan baru KHGT:\n\n"
10695 |
10696 | for pt in titik_hasil:
10697 |     tz_approx = int(self.get_tz_from_lon(pt['lon']))
10698 |     tz_str = f"UTC+{tz_approx}" if tz_approx >= 0 else f"UTC{tz_approx}"
10699 |
10700 |     dt_local_ephem = ephemeris.Date(pt['sunset_utc'] + (tz_approx / 24.0))
10701 |     y_l, m_l, d_l, h_l, min_l, s_l = dt_local_ephem.tuple()
10702 |
10703 |     nama_bulan_masehi = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov", "Des"]
10704 |     tgl_lokal_str = f"{int(d_l):02d} {nama_bulan_masehi[int(m_l)]} {format_tahun_aman(int(y_l))}"
10705 |     jam_lokal_str = f"{int(h_l):02d}:{int(min_l):02d}:{int(s_l):02d}"
10706 |
10707 |     y_i, m_i, d_i, h_i, min_i, s_i = pt['ijtimak'].tuple()

```

```

10708 |         ij_jam = f"{int(h_i):02d}:{int(min_i):02d}:{int(s_i):02d}"
10709 |
10710 |         y_s, m_s, d_s, h_s, min_s, s_s = pt['sunset_utc'].tuple()
10711 |         ss_jam = f"{int(h_s):02d}:{int(min_s):02d}:{int(s_s):02d}"
10712 |
10713 |         # PERBAIKAN: Bungkus dengan ephem.Date()
10714 |         batas_00_utc_ephem = ephem.Date(ephem.Date((int(y_l), int(m_l), int(d_l), 0, 0, 0)) + 1.0)
10715 |         bt_jam = "00:00:00"
10716 |         fajar_info = ""
10717 |
10718 |         negara_amerika = ["Amerika Serikat", "Kanada", "Meksiko", "Brasil", "Argentina", "Kolombia", "Peru", "Chili"]
10719 |         is_amerika = pt['negara'] in negara_amerika
10720 |
10721 |         import pytz
10722 |         dt_ij_utc = pt['ijtimak'].datetime().replace(tzinfo=pytz.utc)
10723 |         nm_tt_val = self.ts.from_datetime(dt_ij_utc).tt
10724 |         h_y, h_m = self.get_hijri_month_from_tt(nm_tt_val)
10725 |         hijri_bulan_str = BULAN_HIJRIAH[h_m - 1]
10726 |         hijri_tahun_str = str(h_y) if h_y > 0 else f"{abs(h_y-1)} SH"
10727 |
10728 |         if pt['sunset_utc'] < batas_00_utc_ephem:
10729 |             tgl_l_mas = ephem.Date(batas_00_utc_ephem)
10730 |             gy, gm, gd, _, _, _ = tgl_l_mas.tuple()
10731 |             masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
10732 |             status_ijtimak = f"Kriteria PKG 1 Terpenuhi (Terpenuhi sblm {bt_jam} UTC)\n» Kesimpulan: BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
10733 |         else:
10734 |             gisborne = ephem.Observer()
10735 |             gisborne.lat, gisborne.lon = math.radians(-38.6623), math.radians(178.0176)
10736 |             gisborne.elevation, gisborne.pressure = 0, 0
10737 |             gisborne.horizon = '-18'
10738 |             matahari_g = ephem.Sun()
10739 |             gisborne.date = ephem.Date(batas_00_utc_ephem - (12.0 / 24.0))
10740 |             matahari_g.compute(gisborne)
10741 |
10742 |             try:
10743 |                 fajar_gisborne_ephem = gisborne.next_rising(matahari_g, use_center=True)
10744 |                 _, _, _, h_f, min_f, s_f = fajar_gisborne_ephem.tuple()
10745 |                 fj_jam = f"{int(h_f):02d}:{int(min_f):02d}:{int(s_f):02d}"
10746 |                 fajar_info = f" Fajar Gisb. : {fj_jam} UTC\n"
10747 |
10748 |                 if is_amerika and pt['ijtimak'] < fajar_gisborne_ephem:
10749 |                     tgl_l_mas = ephem.Date(batas_00_utc_ephem)
10750 |                     gy, gm, gd, _, _, _ = tgl_l_mas.tuple()
10751 |                     masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
10752 |                     status_ijtimak = f"Kriteria PKG 2 Terpenuhi (Titik Amerika & Ijtima < Fajar Gisb.)\n» Kesimpulan: BESOK (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
10753 |                 else:
10754 |                     tgl_lusa = ephem.Date(batas_00_utc_ephem + 1.0)
10755 |                     gy, gm, gd, _, _, _ = tgl_lusa.tuple()
10756 |                     masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
10757 |                     alasan = "Titik bukan di Benua Amerika" if not is_amerika else "Ijtima > Terbit Fajar"
10758 |                     status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi ({alasan})\n» Kesimpulan: LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
10759 |             except:
10760 |                 tgl_lusa = ephem.Date(batas_00_utc_ephem + 1.0)
10761 |                 gy, gm, gd, _, _, _ = tgl_lusa.tuple()
10762 |                 masehi_str = f"{int(gd)} {BULAN_MASEHI[int(gm)-1]} {format_tahun_aman(int(gy))}"
10763 |                 fajar_info = " Fajar Gisb. : Error/N/A\n"
10764 |                 status_ijtimak = f"Kriteria PKG 2 Tidak Terpenuhi (Gagal Menghitung Fajar)\n» Kesimpulan: LUSA (1 {hijri_bulan_str} {hijri_tahun_str} H = {masehi_str})"
10765 |
10766 |         report += (
10767 |             f"[{pt['label'].upper()}\n"
10768 |             f"{'-'*45}\n"
10769 |             f"Lokasi : {pt['kota']}, {pt['negara']}\n"
10770 |             f"Maghrib : {jam_lokal_str} ({tz_str}) | {tgl_lokal_str}\n"
10771 |             f"Alt/Eln : {pt['alt_geo']:.2f}° / {pt['elong_geo']:.2f}°\n"
10772 |             f"{'-'*45}\n"
10773 |             f"Parameter Waktu (UTC):\n"
10774 |             f"Waktu Maghrib: {ss_jam} UTC\n"
10775 |             f"Batas PKG 1 : {bt_jam} UTC\n"
10776 |             f"Ijtimak : {ij_jam} UTC\n"
10777 |             f"{{fajar_info}}\n"
10778 |             f"{'-'*45}\n"

```

```

10779 |         f"Status: {status_ijtimak}\n"
10780 |         f"'='*45}\n\n"
10781 |     )
10782 |     return report.strip()
10783 |
10784 | def calculate_seasons(self):
10785 |     try:
10786 |         year = int(self.entry_season_year.get())
10787 |         self.auto_switch_ephemeris(year)
10788 |
10789 |         t0 = self.ts.utc(year, 1, 1)
10790 |         t1 = self.ts.utc(year, 12, 31)
10791 |         t_seasons, y_seasons = almanac.find_discrete(t0, t1, almanac.seasons(self.eph))
10792 |
10793 |         tz_wib = pytz.timezone('Asia/Jakarta')
10794 |         musim_nama = ["Vernal Equinox (Musim Semi Utara)",
10795 |                       "Summer Solstice (Musim Panas Utara)",
10796 |                       "Autumnal Equinox (Musim Gugur Utara)",
10797 |                       "Winter Solstice (Musim Dingin Utara)"]
10798 |
10799 |         report = f"{self.get_header(85)}\n"
10800 |         report += f"{'[ Equinox & Solstice Calculator ]'.center(85)}\n\n"
10801 |         report += f"* Tahun Astronomis: {year} CE\n"
10802 |         report += f"* Zona Waktu: WIB (UTC+7)\n"
10803 |         report += f"*85 + "\n\n"
10804 |
10805 |         for t_ev, y_ev in zip(t_seasons, y_seasons):
10806 |             dt_wib = t_ev.utc.datetime().replace(tzinfo=pytz.utc).astimezone(tz_wib)
10807 |             waktu_str = dt_wib.strftime('%d %B %Y - %H:%M:%S WIB')
10808 |             nama = musim_nama[y_ev]
10809 |             report += f">> {nama:<38} : {waktu_str}\n"
10810 |
10811 |         report += "\n" + f"*85
10812 |         self.after(0, self.display_result, report)
10813 |     except Exception as e:
10814 |         self.after(0, self.display_error, str(e))
10815 |
10816 | def calculate_planetary_times(self):
10817 |     try:
10818 |         year = int(self.entry_pl_year.get())
10819 |         month = int(self.entry_pl_month.get())
10820 |         target_name = self.combo_pl_target.get().lower()
10821 |
10822 |         # Sinkronisasi Ephemeris sesuai rentang tahun
10823 |         self.auto_switch_ephemeris(year)
10824 |
10825 |         # Khusus planet, JPL Ephemeris biasanya menggunakan suffix 'barycenter'
10826 |         target_key = f'{target_name} barycenter' if target_name != 'moon' else 'moon'
10827 |         target_obj = self.eph[target_key]
10828 |         earth = self.eph['earth']
10829 |
10830 |         # Ambil koordinat lokasi
10831 |         lat = float(self.entry_vlat.get())
10832 |         lon = float(self.entry_vlon.get())
10833 |         tz = float(self.entry_vtz.get())
10834 |         elev = float(self.entry_velev.get())
10835 |         loc = wgs84.latlon(lat, lon, elevation_m=elev)
10836 |
10837 |         # Gunakan fungsi safe_montherange agar kebal tahun minus/BCE
10838 |         _, num_days = safe_montherange(year, month)
10839 |
10840 |         # Tentukan rentang pencarian (00:00 hari pertama s.d 24:00 hari terakhir)
10841 |         t0 = self.ts.utc(year, month, 1, -int(tz))
10842 |         t1 = self.ts.utc(year, month, num_days, 24 - int(tz))
10843 |
10844 |         # Hitung kejadian Rise/Set
10845 |         f_rs = almanac.risings_and_settings(self.eph, target_obj, loc)
10846 |         t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)
10847 |
10848 |         # ---> PERBAIKAN: MENGAMBIL NAMA KOTA & PROVINSI DARI SIDEBAR <---
10849 |         try:

```

```

10850 |         nama_kota = self.opt_city.get()
10851 |         nama_prov = self.opt_prov.get()
10852 |         lokasi_text = f'{nama_kota}, {nama_prov} (Lat {lat}, Lon {lon}, TZ UTC{'+' if tz >= 0 else ''}{tz})"
10853 |     except:
10854 |         lokasi_text = f"Lat {lat}, Lon {lon}, TZ UTC{'+' if tz >= 0 else ''}{tz}"
10855 |
10856 |     # Inisialisasi Header Laporan
10857 |     report = f"{self.get_header(85)}\n"
10858 |     report += f'{" Planetary Times: {target_name.upper()} }'.center(85)}\n\n"
10859 |     report += f" Bulan/Tahun : {month:02d}/{format_tahun_aman(year)}\n"
10860 |     report += f" Lokasi      : {lokasi_text}\n"
10861 |     report += " "*85 + "\n"
10862 |     report += " Tanggal          Terbit (Rise)          Terbenam (Set)\n"
10863 |     report += " "*85 + "\n"
10864 |
10865 |     # Loop setiap hari untuk mencocokkan event
10866 |     for d in range(1, num_days + 1):
10867 |         found_rise = "---"
10868 |         found_set = "---"
10869 |
10870 |         if t_rs is not None:
10871 |             for t_val, y_val in get_safe_events(t_rs, y_rs):
10872 |                 # Konversi waktu kejadian ke waktu lokal untuk pengecekan tanggal
10873 |                 t_loc = self.ts.tt_jd(t_val.tt + (tz / 24.0))
10874 |                 _, _, dl, hl, mnl, _ = t_loc.utc
10875 |
10876 |                 if int(dl) == d:
10877 |                     time_str = f"{int(hl):02d}:{int(mnl):02d}"
10878 |                     if y_val == 1: found_rise = time_str
10879 |                     else: found_set = time_str
10880 |
10881 |                 # Format tampilan tanggal (Bebas crash tahun minus)
10882 |                 date_display = f"{d:02d}/{month:02d}/{format_tahun_aman(year)}"
10883 |
10884 |                 # Tambahkan baris data ke laporan
10885 |                 line = f" {date_display:<18} {found_rise:^15}          {found_set:^15}\n"
10886 |                 report += line
10887 |
10888 |     report += " "*85
10889 |     report += "\n* Remarks: '----' means the event does not occur on that day.\n"
10890 |     report += f"* Calculated using JPL NASA Engine ({self.ephemeris_name})."
10891 |
10892 |     # Tampilkan ke ruang kanan (Textbox)
10893 |     self.after(0, self.display_result, report)
10894 |
10895 | except Exception as e:
10896 |     import traceback
10897 |     self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
10898 |
10899 | def display_result(self, report_text):
10900 |     # Simpan laporan mentah berbahasa Indonesia agar ruang kanan bisa
10901 |     # digambar ulang 100% ketika bahasa diganti ke ID/EN/AR.
10902 |     self._last_raw_report = report_text
10903 |     self._last_raw_textbox_content = report_text
10904 |
10905 |     self.textbox.configure(state="normal")
10906 |     self.textbox.delete("1.0", "end")
10907 |     self.textbox.insert("1.0", report_text)
10908 |     self.textbox.configure(state="disabled")
10909 |     self.lbl_status.configure(text=self.tr("Kalkulasi Selesai"), text_color="#00E676")
10910 |     self.btn_hitung.configure(state="normal", text=self.tr("► PROSES DATA"))
10911 |
10912 | def display_error(self, error_msg):
10913 |     # Error juga disimpan mentah agar saat EN/AR dipilih teks debug
10914 |     # ikut berganti bahasa dan tidak memicu translasi ganda.
10915 |     self._last_raw_report = f"TERJADI KESALAHAN (DEBUG):\n{error_msg}"
10916 |     self._last_raw_textbox_content = self._last_raw_report
10917 |
10918 |     self.textbox.configure(state="normal")
10919 |     self.textbox.delete("1.0", "end")
10920 |     self.textbox.insert("1.0", self._last_raw_report)

```

```

10921 |         self.textbox.configure(state="disabled")
10922 |         self.lbl_status.configure(text=self.tr("Error Kalkulasi"), text_color="#FF1744")
10923 |         self.btn_hitung.configure(state="normal", text=self.tr("▶ PROSES DATA"))
10924 |
10925 |     def auto_detect_location(self):
10926 |         self.lbl_status.configure(text="Mendeteksi lokasi...", text_color="#00E5FF")
10927 |         def fetch_location():
10928 |             try:
10929 |                 import json
10930 |                 import urllib.request
10931 |                 req = urllib.request.Request("http://ip-api.com/json/", headers={'User-Agent': 'Mozilla/5.0'})
10932 |                 with urllib.request.urlopen(req, timeout=5) as response:
10933 |                     data = json.loads(response.read().decode())
10934 |                     if data['status'] == 'success':
10935 |                         lat, lon, city = data['lat'], data['lon'], data['city']
10936 |                         self.after(0, self._update_all_loc_entries, lat, lon, city)
10937 |                     else:
10938 |                         self.after(0, lambda: messagebox.showerror("Gagal", "Tidak dapat mendeteksi lokasi.))
10939 |             except Exception as e:
10940 |                 self.after(0, lambda: messagebox.showerror("Error", f"Koneksi Geolocation Gagal: {e}"))
10941 |             finally:
10942 |                 self.after(0, lambda: self.lbl_status.configure(text="Sistem Siap", text_color="#00E676"))
10943 |             threading.Thread(target=fetch_location, daemon=True).start()
10944 |
10945 |     def _update_all_loc_entries(self, lat, lon, city):
10946 |         lat_entries = ['entry_vlat', 'entry_eph_lat', 'entry_qlat', 'entry_mt_lat', 'entry_st_lat', 'entry_pt_lat', 'entry_qtlat', 'entry_eph3d_lat', 'entry_miz_lat', 'entry_kb_lat', 'entry_astro_lat',
10947 | 'entry_pasang_lat', 'entry_fajar_lat', 'entry_analemma_lat']
10948 |         lon_entries = ['entry_vlon', 'entry_eph_lon', 'entry_qlon', 'entry_mt_lon', 'entry_st_lon', 'entry_pt_lon', 'entry_qtлон', 'entry_eph3d_lon', 'entry_miz_lon', 'entry_kb_lon', 'entry_astro_lon',
10949 | 'entry_pasang_lon', 'entry_fajar_lon', 'entry_analemma_lon']
10950 |         tz_entries = ['entry_vtz', 'entry_eph_tz', 'entry_qtz', 'entry_mt_tz', 'entry_st_tz', 'entry_pt_tz', 'entry_qttz', 'entry_miz_tz', 'entry_kb_tz', 'entry_astro_tz', 'entry_pasang_tz',
10951 | 'entry_fajar_tz', 'entry_analemma_tz']
10952 |
10953 |         tz_val = self.get_tz_from_lon(lon)
10954 |         for name in lat_entries:
10955 |             if hasattr(self, name):
10956 |                 e = getattr(self, name); e.delete(0, 'end'); e.insert(0, str(lat))
10957 |         for name in lon_entries:
10958 |             if hasattr(self, name):
10959 |                 e = getattr(self, name); e.delete(0, 'end'); e.insert(0, str(lon))
10960 |         for name in tz_entries:
10961 |             if hasattr(self, name):
10962 |                 e = getattr(self, name); e.delete(0, 'end'); e.insert(0, str(tz_val))
10963 |
10964 |         self.lbl_status.configure(text=f"📍 Lokasi Terdeteksi: {city} ({lat}, {lon})", text_color="green")
10965 |
10966 |         # ---> TAMBAHKAN BARIS INI DI AKHIR FUNGSI _update_all_loc_entries <---
10967 |         if hasattr(self, 'fetch_and_update_weather'):
10968 |             labels_cuaca = [
10969 |                 'lbl_status_cuaca', 'lbl_cuaca_astro',
10970 |                 'lbl_cuaca_pasang', 'lbl_cuaca_fajar', 'lbl_cuaca_analemma'
10971 |             ]
10972 |             for lbl_name in labels_cuaca:
10973 |                 if hasattr(self, lbl_name):
10974 |                     self.fetch_and_update_weather(lat, lon, getattr(self, lbl_name))
10975 |
10976 |         # -----
10977 |
10978 |         # 2. Cari Provinsi dari CITY_DB berdasarkan nama kota hasil deteksi API
10979 |         ditemukan_prov = None
10980 |         ditemukan_kota = city
10981 |
10982 |         for prov_name, cities_dict in CITY_DB.items():
10983 |             for db_city in cities_dict.keys():
10984 |                 # Pencarian fleksibel
10985 |                 if city.lower() in db_city.lower() or db_city.lower() in city.lower():
10986 |                     ditemukan_prov = prov_name
10987 |                     ditemukan_kota = db_city
10988 |                     break
10989 |             if ditemukan_prov:
10990 |                 break
10991 |
10992 |         # 3. Sesuaikan Dropdown Provinsi dan Kota di Sidebar

```



```

10989 |         if ditemukan_prov:
10990 |             self.opt_prov.set(ditemukan_prov)
10991 |             cities_in_prov = sorted(list(CITY_DB[ditemukan_prov].keys()))
10992 |             self.opt_city.configure(values=cities_in_prov)
10993 |             self.opt_city.set(ditemukan_kota)
10994 |         else:
10995 |             # Jika kota dari GPS TIDAK ada di database bawaan, buat menu kustom
10996 |             prov_values = list(self.opt_prov.cget("values"))
10997 |             if "Lokasi Otomatis (GPS)" not in prov_values:
10998 |                 self.opt_prov.configure(values=["Lokasi Otomatis (GPS)" + prov_values])
10999 |             self.opt_prov.set("Lokasi Otomatis (GPS)")
11000 |
11001 |             city_values = list(self.opt_city.cget("values"))
11002 |             if city not in city_values:
11003 |                 self.opt_city.configure(values=[city] + city_values)
11004 |             self.opt_city.set(city)
11005 |
11006 |         # 4. Update Label atas dan munculkan pop-up sukses
11007 |         self.lokasi_nama.set(f"{{ditemukan_kota}} (Auto-Detected)")
11008 |         messagebox.showinfo("Lokasi Ditemukan", f"Sistem GPS mendeteksi lokasi Anda di {{ditemukan_kota}}\nLat: {{lat}}\nLon: {{lon}}\n\nSeluruh modul telah diperbarui ke koordinat ini.")
11009 |
11010 |     def export_to_ics(self):
11011 |         # Cek apakah jadwal salat sudah ada di memori
11012 |         if not self.daily_prayer_schedule:
11013 |             messagebox.showwarning("Peringatan", "Jadwal Salat belum tersedia di memori!\nSilakan aktifkan 'Alarm Salat' di sidebar agar sistem menghitung jadwal hari ini terlebih dahulu.")
11014 |             return
11015 |
11016 |         # Mengambil nilai zona waktu dari input untuk penyesuaian UTC
11017 |         try:
11018 |             tz_offset = float(self.entry_pt_tz.get())
11019 |         except ValueError:
11020 |             tz_offset = 7.0 # Default ke WIB (UTC+7) jika kosong
11021 |
11022 |         # Membuat struktur Header file iCalendar murni
11023 |         ics_content = [
11024 |             "BEGIN:VCALENDAR",
11025 |             "VERSION:2.0",
11026 |             "PRODID:-//KHGT Times By Kasmui//ID",
11027 |             "CALSCALE:GREGORIAN"
11028 |         ]
11029 |
11030 |         # Waktu pembuatan file (Stamp) dalam format UTC
11031 |         now_utc_str = datetime.datetime.now(datetime.timezone.utc).strftime('%Y%m%dT%H%M%S')
11032 |
11033 |         # Looping setiap jadwal salat yang ada di memori
11034 |         for prayer_name, dt_local in self.daily_prayer_schedule.items():
11035 |             # Konversi waktu lokal (naive) kembali ke UTC karena standar ICS wajib UTC
11036 |             dt_utc = dt_local - datetime.timedelta(hours=tz_offset)
11037 |
11038 |             # Format waktu sesuai standar ICS: YYYYMMDDThhmmssZ
11039 |             dt_start_str = dt_utc.strftime('%Y%m%dT%H%M%S')
11040 |
11041 |             # Asumsi durasi event salat di kalender adalah 15 menit
11042 |             dt_end_str = (dt_utc + datetime.timedelta(minutes=15)).strftime('%Y%m%dT%H%M%S')
11043 |
11044 |             # Membuat Unique ID untuk kalender
11045 |             uid = f"{{dt_start_str}}-{{prayer_name.replace('/', '')}}.replace(' ', '')}@khggtimes"
11046 |
11047 |             # Memasukkan data event salat
11048 |             ics_content.extend([
11049 |                 "BEGIN:VEVENT",
11050 |                 f"UID:{{uid}}",
11051 |                 f"DTSTAMP:{{now_utc_str}}",
11052 |                 f"SUMMARY:Waktu Salat {{prayer_name}}",
11053 |                 f"DTSTART:{{dt_start_str}}",
11054 |                 f"DTEND:{{dt_end_str}}",
11055 |                 "BEGIN:VALARM",
11056 |                 "ACTION:DISPLAY",
11057 |                 f"DESCRIPTION:Telah masuk waktu {{prayer_name}}",
11058 |                 "TRIGGER:-PT10M", # Setel pengingat (notifikasi) 10 menit sebelum waktu salat
11059 |                 "END:VALARM",

```

```

11060 |         "END:VEVENT"
11061 |     })
11062 |
11063 |     ics_content.append("END:VCALENDAR")
11064 |     final_ics = "\n".join(ics_content)
11065 |
11066 |     # Dialog penyimpanan file
11067 |     filepath = filedialog.asksaveasfilename(
11068 |         initialfile=f"Jadwal_Salat_{datetime.datetime.now().strftime('%Y%m%d')}.ics",
11069 |         defaultextension=".ics",
11070 |         filetypes=[("iCalendar Files", "*.ics")]
11071 |     )
11072 |
11073 |     # Eksekusi penyimpanan ke format .ics
11074 |     if filepath:
11075 |         try:
11076 |             with open(filepath, 'w', encoding='utf-8') as f:
11077 |                 f.write(final_ics)
11078 |                 messagebox.showinfo("Sukses", f"File Kalender ICS berhasil dibuat murni tanpa library eksternal!\nDisimpan di:\n{filepath}\n\nFile ini siap diimpor ke Google Calendar, Apple Calendar, atau
Outlook.")
11079 |             except Exception as e:
11080 |                 messagebox.showerror("Error", f"Gagal menyimpan file ICS: {e}")
11081 |
11082 |     def save_to_txt(self):
11083 |         mode = self.combo_mode.get()
11084 |         report_data = self.textbox.get("1.0", "end-1c")
11085 |
11086 |         # Handle khusus untuk Modul Gerhana (Mengekstrak data dari Treeview)
11087 |         if "Analisis Gerhana" in mode:
11088 |             report_data = "DATA ANALISIS GERHANA\n" + "="*80 + "\n"
11089 |             headers = [self.tabel_gerhana.heading(c)["text"] for c in self.tabel_gerhana["columns"]]
11090 |             report_data += " | ".join(headers) + "\n" + "-"*80 + "\n"
11091 |             for item_id in self.tabel_gerhana.get_children():
11092 |                 row = self.tabel_gerhana.item(item_id)["values"]
11093 |                 report_data += " | ".join(str(val) for val in row) + "\n"
11094 |
11095 |             if not report_data.strip() or "TERJADI KESALAHAN" in report_data:
11096 |                 messagebox.showwarning("Peringatan", "Tidak ada data kalkulasi valid yang bisa disimpan.")
11097 |                 return
11098 |
11099 |             filepath = filedialog.asksaveasfilename(
11100 |                 initialfile="Laporan_KHGT_Times.txt",
11101 |                 defaultextension=".txt",
11102 |                 filetypes=[("Text Files", "*.txt")]
11103 |             )
11104 |
11105 |             if filepath:
11106 |                 try:
11107 |                     with open(filepath, "w", encoding="utf-8") as f:
11108 |                         f.write(report_data)
11109 |                         messagebox.showinfo("Sukses", f"Data berhasil disimpan di:\n{filepath}")
11110 |                         if getattr(self, 'sidebar_visible', False):
11111 |                             self.toggle_sidebar()
11112 |                 except Exception as e:
11113 |                     messagebox.showerror("Error", f"Gagal menyimpan file:\n{str(e)}")
11114 |
11115 |     def export_to_csv(self):
11116 |         mode = self.combo_mode.get()
11117 |         report_data = self.textbox.get("1.0", "end-1c")
11118 |
11119 |         if (not report_data or "TERJADI KESALAHAN" in report_data) and "Analisis Gerhana" not in mode:
11120 |             messagebox.showwarning("Peringatan", "Tidak ada data kalkulasi valid.")
11121 |             return
11122 |
11123 |         filepath = filedialog.asksaveasfilename(
11124 |             initialfile="Data_Ekstraksi_Falak.csv",
11125 |             defaultextension=".csv",
11126 |             filetypes=[("CSV Files", "*.csv"), ("All Files", "*.*")]
11127 |         )
11128 |
11129 |         if filepath:

```

```

11130 |         try:
11131 |             # utf-8-sig memastikan karakter derajat (°) terbaca sempurna di Microsoft Excel
11132 |             with open(filepath, mode='w', newline='', encoding='utf-8-sig') as f:
11133 |                 writer = csv.writer(f, delimiter=';')
11134 |
11135 |                 # Logika 1: Jika berada di Modul Gerhana (Treeview)
11136 |                 if "Analisis Gerhana" in mode:
11137 |                     headers = [self.tabel_gerhana.heading(c)["text"] for c in self.tabel_gerhana["columns"]]
11138 |                     writer.writerow(headers)
11139 |                     for item_id in self.tabel_gerhana.get_children():
11140 |                         row = self.tabel_gerhana.item(item_id)["values"]
11141 |                         writer.writerow(row)
11142 |
11143 |                 # Logika 2: Modul teks biasa (Mencari karakter pemisah '|')
11144 |                 else:
11145 |                     lines = report_data.split('\n')
11146 |                     tabel_ditemukan = False
11147 |                     for line in lines:
11148 |                         if '|' in line and not line.startswith('=') and not line.startswith('-'):
11149 |                             row = [col.strip() for col in line.split('|')]
11150 |                             writer.writerow(row)
11151 |                             tabel_ditemukan = True
11152 |
11153 |                 # Fallback jika ternyata layar tidak memuat tabel kolom '|'
11154 |                 if not tabel_ditemukan:
11155 |                     for line in lines:
11156 |                         writer.writerow([line.strip()])
11157 |
11158 |                 messagebox.showinfo("Sukses", f"Data berhasil diekstrak dan diekspor ke CSV:\n{filepath}")
11159 |             except Exception as e:
11160 |                 messagebox.showerror("Error CSV", f"Gagal membuat file CSV: {e}")
11161 |
11162 |     def export_to_pdf(self):
11163 |         mode = self.combo_mode.get()
11164 |
11165 |         # --- 1. JIKA SEDANG DI MODUL KALENDER (Gunakan Render Kustom HD agar tajam) ---
11166 |         if "Kalender Hijriah" in mode:
11167 |             img = self._generate_calendar_image()
11168 |             filepath = filedialog.asksaveasfilename(
11169 |                 initialfile=f"Kalender Hijriah_{self.cal_h_year}H.pdf",
11170 |                 defaulttextextension=".pdf",
11171 |                 filetypes=[("PDF Document", "*.pdf")]
11172 |             )
11173 |             if filepath:
11174 |                 try:
11175 |                     # PDF tidak mendukung RGBA, harus convert ke RGB
11176 |                     img.convert('RGB').save(filepath, "PDF", resolution=150.0)
11177 |                     messagebox.showinfo("Sukses", f"Kalender Hijriah berhasil diekspor ke PDF:\n{filepath}")
11178 |                 except Exception as e:
11179 |                     messagebox.showerror("Error", f"Gagal menyimpan PDF Kalender: {e}")
11180 |             return
11181 |
11182 |         if "Kalender Masehi" in mode:
11183 |             img = self._generate_calmasehi_image()
11184 |             nama_bulan = BULAN_MASEHI[self.cal_m_month - 1]
11185 |             filepath = filedialog.asksaveasfilename(
11186 |                 initialfile=f"Kalender Masehi_{nama_bulan}_{self.cal_m_year}.pdf",
11187 |                 defaulttextextension=".pdf",
11188 |                 filetypes=[("PDF Document", "*.pdf")]
11189 |             )
11190 |             if filepath:
11191 |                 try:
11192 |                     img.convert('RGB').save(filepath, "PDF", resolution=150.0)
11193 |                     messagebox.showinfo("Sukses", f"Kalender Masehi berhasil diekspor ke PDF:\n{filepath}")
11194 |                 except Exception as e:
11195 |                     messagebox.showerror("Error", f"Gagal menyimpan PDF Kalender: {e}")
11196 |             return
11197 |
11198 |         # --- 2. UNTUK SEMUA MODUL LAINNYA (TEKS, 3D, MAP, CHART, GERHANA) ---
11199 |         # Menangkap screenshot main_frame dan menyimpannya sebagai halaman PDF
11200 |         filepath = filedialog.asksaveasfilename(

```

```

11201 |         initialfile="Laporan_KHGT_Times.pdf",
11202 |         defaulttextextension=".pdf",
11203 |         filetypes=[("PDF Document", "*.pdf")]
11204 |     )
11205 |
11206 |     if filepath:
11207 |         try:
11208 |             import time
11209 |             # Pastikan UI stabil
11210 |             self.update_idletasks()
11211 |             time.sleep(0.2)
11212 |
11213 |             # Ambil koordinat main_frame (ruang kanan)
11214 |             x0 = self.main_frame.winfo_rootx()
11215 |             y0 = self.main_frame.winfo_rooty()
11216 |             w = self.main_frame.winfo_width()
11217 |             h = self.main_frame.winfo_height()
11218 |
11219 |             # Tangkap area tersebut
11220 |             bbox = (x0, y0, x0 + w, y0 + h)
11221 |             screenshot = ImageGrab.grab(bbox=bbox)
11222 |
11223 |             # Konversi hasil screenshot ke PDF (Wajib convert ke RGB)
11224 |             screenshot.convert('RGB').save(filepath, "PDF", resolution=100.0)
11225 |
11226 |             messagebox.showinfo("Sukses", f"Laporan visual berhasil diekspor ke PDF:\n{filepath}")
11227 |
11228 |         except Exception as e:
11229 |             import traceback
11230 |             messagebox.showerror("Error PDF", f"Gagal membuat PDF:\n{e}\n\n{traceback.format_exc()}")
11231 |
11232 |     def export_to_png(self):
11233 |         mode = self.combo_mode.get()
11234 |
11235 |         # --- 1. JIKA SEDANG DI MODUL KALENDER HIJRIAH (Gunakan Render Kustom HD) ---
11236 |         if "Kalender Hijriah" in mode:
11237 |             img = self._generate_calendar_image()
11238 |             filepath = filedialog.asksaveasfilename(
11239 |                 initialfile=f"Kalender_Hijriah_{self.cal_h_year}H.png",
11240 |                 defaulttextextension=".png",
11241 |                 filetypes=[("PNG Image", "*.png")]
11242 |             )
11243 |             if filepath:
11244 |                 try:
11245 |                     img.save(filepath, "PNG")
11246 |                     messagebox.showinfo("Sukses", f"Kalender Hijriah berhasil disimpan sbg PNG:\n{filepath}")
11247 |                 except Exception as e:
11248 |                     messagebox.showerror("Error", f"Gagal menyimpan PNG Kalender: {e}")
11249 |             return
11250 |
11251 |         # --- 2. JIKA SEDANG DI MODUL KALENDER MASEHI (Gunakan Render Kustom HD) ---
11252 |         if "Kalender Masehi" in mode:
11253 |             img = self._generate_calmasehi_image()
11254 |             nama_bulan = BULAN_MASEHI[self.cal_m_month - 1]
11255 |             filepath = filedialog.asksaveasfilename(
11256 |                 initialfile=f"Kalender_Masehi_{nama_bulan}_{self.cal_m_year}.png",
11257 |                 defaulttextextension=".png",
11258 |                 filetypes=[("PNG Image", "*.png")]
11259 |             )
11260 |             if filepath:
11261 |                 try:
11262 |                     img.save(filepath, "PNG")
11263 |                     messagebox.showinfo("Sukses", f"Kalender Masehi berhasil disimpan sbg PNG:\n{filepath}")
11264 |                 except Exception as e:
11265 |                     messagebox.showerror("Error", f"Gagal menyimpan PNG Kalender: {e}")
11266 |             return
11267 |
11268 |         # --- 3. UNTUK SEMUA MODUL LAINNYA (TEKS, GRAFIK 3D, MAP, GERHANA, DLL) ---
11269 |         # Menangkap (Screenshot) seluruh area 'main_frame' (ruang kanan) secara eksak beserta warnanya
11270 |         filepath = filedialog.asksaveasfilename(
11271 |             initialfile="Output_KHGT_Times.png",

```

```

11272 |         defaultextension=".png",
11273 |         filetypes=[("PNG Image", "*.png")]
11274 |     )
11275 |
11276 |     if filepath:
11277 |         try:
11278 |             import time
11279 |
11280 |             # Sembunyikan tooltip/kursor sejenak & pastikan UI sudah ter-render sempurna
11281 |             self.update_idletasks()
11282 |             time.sleep(0.2)
11283 |
11284 |             # Ambil koordinat layar dari main_frame (ruang kanan)
11285 |             x0 = self.main_frame.winfo_rootx()
11286 |             y0 = self.main_frame.winfo_rooty()
11287 |             w = self.main_frame.winfo_width()
11288 |             h = self.main_frame.winfo_height()
11289 |
11290 |             # Tentukan Bounding Box (Kiri, Atas, Kanan, Bawah)
11291 |             bbox = (x0, y0, x0 + w, y0 + h)
11292 |
11293 |             # Tangkap gambar menggunakan Pillow ImageGrab
11294 |             img = ImageGrab.grab(bbox=bbox)
11295 |
11296 |             # Simpan ke PNG
11297 |             img.save(filepath, "PNG")
11298 |
11299 |             messagebox.showinfo("Sukses", f"Tampilan layar (Ruang Kanan) berhasil disimpan sebagai PNG:\n{filepath}")
11300 |
11301 |         except Exception as e:
11302 |             import traceback
11303 |             messagebox.showerror("Error PNG", f"Gagal menangkap layar:\n{e}\n\n{traceback.format_exc()}")
11304 |
11305 |
11306 |
11307 | def analisis_komparasi_ramadhan_50_tahun(self):
11308 |     # Jalankan di thread terpisah agar GUI tidak Not Responding (Freeze)
11309 |     threading.Thread(target=self._proses_komparasi_ramadhan, daemon=True).start()
11310 |
11311 | def _proses_komparasi_ramadhan(self):
11312 |     # Update UI Status
11313 |     self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun Ramadhan...", text_color="#00E5FF"))
11314 |     self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
11315 |     self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
11316 |     self.after(0, lambda: self.textbox.delete("1.0", "end"))
11317 |     self.after(0, lambda: self.textbox.insert("end", "Memulai mesin Ephemeris untuk 50 Tahun (1447 H - 1496 H)...\nMohon tunggu...\n\n"))
11318 |
11319 |     output_lines = []
11320 |     output_lines.append(self.get_header(125))
11321 |     output_lines.append("[ REKAPITULASI KOMPARASI AWAL RAMADHAN (1447 H - 1496 H) ]".center(125))
11322 |     output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokal Sabang Alt>=3°, Eln>=6.4°)".center(125))
11323 |     output_lines.append("-" * 125)
11324 |     output_lines.append(f'{"Tahun":<8} | {"Waktu Ijtimak (UTC)":<19} | {"1 Ramadhan (KHGT)":<20} | {"1 Ramadhan (MABIMS Sabang)":<28} | {"Status / Selisih"}')
11325 |     output_lines.append("-" * 125)
11326 |
11327 |     # Setup Observer Sabang untuk Neo MABIMS
11328 |     sabang = ephemeris.Observer()
11329 |     sabang.lat = math.radians(5.8942)
11330 |     sabang.lon = math.radians(95.3184)
11331 |     sabang.elevation = 0
11332 |     sabang.pressure = 1010
11333 |     sabang.temp = 25
11334 |
11335 |     matahari = ephemeris.Sun()
11336 |     bulan = ephemeris.Moon()
11337 |
11338 |     # Iterasi dari 1447 H sampai 1496 H
11339 |     for thn_h in range(1447, 1497):
11340 |         try:
11341 |             # 1. Ambil data 1 Ramadhan KHGT dari Database Bawaan
11342 |             # Index 8 adalah bulan ke-9 (Ramadan)

```

```

11343 |         data_ramadan = HIJRI_DB[thn_h][8]
11344 |         tgl_khgt_str = data_ramadan[2] # Format: DD-MMM-YYYY
11345 |         dt_khgt = datetime.datetime.strptime(tgl_khgt_str, "%d-%b-%Y")
11346 |
11347 |         # 2. Tanggal Rukyat (29 Syakban) adalah 1 hari sebelum 1 Ramadhan KHGT
11348 |         dt_rukyat = dt_khgt - datetime.timedelta(days=1)
11349 |
11350 |         # 3. Cari Waktu Ijtimak di sekitar tanggal Rukyat
11351 |         waktu_pencarian = ephem.Date(dt_rukyat)
11352 |         ijtimak = ephem.previous_new_moon(waktu_pencarian + 5) # +5 hari sbg buffer pencarian
11353 |         dt_ijtimak_utc = ijtimak.datetime()
11354 |         str_ijtimak = dt_ijtimak_utc.strftime("%d-%m-%Y %H:%M")
11355 |
11356 |         # 4. Kalkulasi Sunset di Sabang pada hari Rukyat
11357 |         sabang.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
11358 |         try:
11359 |             waktu_sunset = sabang.next_setting(matahari)
11360 |         except:
11361 |             waktu_sunset = ephem.Date(dt_rukyat.replace(hour=11, minute=30, second=0)) # Fallback perkiraan UTC Sunset Sabang
11362 |
11363 |         # 5. Hitung Posisi Bulan saat Sunset Sabang (Toposentrik)
11364 |         sabang.date = waktu_sunset
11365 |         matahari.compute(sabang)
11366 |         bulan.compute(sabang)
11367 |
11368 |         alt_topo = math.degrees(bulan.alt)
11369 |         elong_topo = math.degrees(ephem.separation(matahari, bulan))
11370 |
11371 |         # 6. Evaluasi Neo MABIMS (Alt >= 3° dan Elong >= 6.4°)
11372 |         # Serta pastikan ijtimak sudah terjadi sebelum sunset (umur bulan positif)
11373 |         umur_bulan = waktu_sunset - ijtimak
11374 |
11375 |         if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
11376 |             dt_mabims = dt_khgt
11377 |             status = "Serentak (Sama)"
11378 |         else:
11379 |             # Jika gagal memenuhi MABIMS di Sabang, bulan Syakban digenapkan 30 hari
11380 |             dt_mabims = dt_khgt + datetime.timedelta(days=1)
11381 |             status = "Beda (MABIMS Mundur 1 Hari)"
11382 |
11383 |         # 7. Format Output String
11384 |         str_khgt = dt_khgt.strftime("%d %b %Y")
11385 |         str_mabims = dt_mabims.strftime("%d %b %Y")
11386 |
11387 |         baris = f"{thn_h} H    | {str_ijtimak:<19} | {str_khgt:<20} | {str_mabims:<28} | {status}"
11388 |         output_lines.append(baris)
11389 |
11390 |         # Sisipkan pemisah per 10 Tahun (Dekade)
11391 |         if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
11392 |             output_lines.append("-" * 125)
11393 |
11394 |     except Exception as e:
11395 |         output_lines.append(f"{thn_h} H    | Error kalkulasi: {str(e)}")
11396 |
11397 |     output_lines.append("-" * 125)
11398 |     output_lines.append("* Catatan Metodologi:")
11399 |     output_lines.append("- KHGT menggunakan referensi Global (Kesatuan Matlak). Diambil dari struktur HIJRI_DB.")
11400 |     output_lines.append("- Neo MABIMS menggunakan uji coba Toposentrik di Ufuk Barat Indonesia (Sabang, Aceh).")
11401 |     output_lines.append("- Perbedaan biasanya terjadi jika KHGT tervalidasi di Benua Amerika, namun Sabang belum memenuhi syarat MABIMS.")
11402 |
11403 |     laporan_final = "\n".join(output_lines)
11404 |
11405 |     # Kembalikan ke UI (Main Thread)
11406 |     self.after(0, lambda: self.textbox.insert("1.0", laporan_final))
11407 |     self.after(0, lambda: self.textbox.configure(state="disabled"))
11408 |     self.after(0, lambda: self.lbl_status.configure(text="Rekapitulasi 50 Tahun Selesai", text_color="#00E676"))
11409 |     self.after(0, lambda: self.btn_hitung.configure(state="normal"))
11410 |
11411 | # =====
11412 | # FUNGSI PEMBERSIH SAAT APLIKASI DITUTUP
11413 | # =====

```

```

11414 | def on_closing(self):
11415 |     try:
11416 |         self.save_release_language_cache()
11417 |         self.save_output_language_cache()
11418 |     except Exception:
11419 |         pass
11420 |     self.anim_running = False
11421 |     self.is_viewing_3d = False
11422 |     self.eph3d_is_live = False
11423 |     self.alarm_enabled.set(False)
11424 |
11425 |     if HAS_PYGAME:
11426 |         try:
11427 |             import pygame
11428 |             pygame.mixer.music.stop()
11429 |             pygame.quit()
11430 |         except:
11431 |             pass
11432 |
11433 |     self.destroy()
11434 |     import os
11435 |     os._exit(0)
11436 |
11437 | def get_tz_from_lon(self, lon_val):
11438 |     """Menentukan Zona Waktu Administratif (Indonesia & Override Kustom) atau Geografis Global."""
11439 |
11440 |     # 1. OVERRIDE KOTA/NEGARA SPESIFIK (Administratif & DST)
11441 |     # Karena rumus matematis (lon / 15) tidak mendeteksi Daylight Saving Time (DST),
11442 |     # Bapak bisa mendaftarkan nilai bujur kota-kota Eropa/Amerika di sini.
11443 |     # Saat ini London diset ke 1.0 (BST). Jika masuk musim dingin (GMT), cukup ubah ke 0.0.
11444 |     override_tz = {
11445 |         -0.1278: 1.0, # London
11446 |         -1.8904: 1.0, # Birmingham
11447 |         -2.2426: 1.0, # Manchester
11448 |         -4.2518: 1.0, # Glasgow
11449 |         -3.1883: 1.0 # Edinburgh
11450 |     }
11451 |
11452 |     # Cek apakah bujur saat ini terdaftar di kamus pengecualian
11453 |     if lon_val in override_tz:
11454 |         return override_tz[lon_val]
11455 |
11456 |     # 2. ZONA WAKTU INDONESIA (WIB, WITA, WIT)
11457 |     if 94.0 <= lon_val < 115.0:
11458 |         return 7.0
11459 |     elif 115.0 <= lon_val < 126.0:
11460 |         return 8.0
11461 |     elif 126.0 <= lon_val <= 141.0:
11462 |         return 9.0
11463 |
11464 |     # 3. OVERRIDE ZONA WAKTU ALASKA -> UTC-11
11465 |     elif -180.0 <= lon_val <= -140.0:
11466 |         return -11.0
11467 |
11468 |     # 4. ZONA WAKTU GLOBAL (Default Matematis)
11469 |     else:
11470 |         return float(int(round(lon_val / 15.0)))
11471 |
11472 |
11473 | def analisis_komparasi_syawal_50_tahun(self):
11474 |     threading.Thread(target=self._proses_komparasi_syawal, daemon=True).start()
11475 |
11476 | def _proses_komparasi_syawal(self):
11477 |     # Update UI Status
11478 |     self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun Syawal...", text_color="#00E5FF"))
11479 |     self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
11480 |     self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
11481 |     self.after(0, lambda: self.textbox.delete("1.0", "end"))
11482 |     self.after(0, lambda: self.textbox.insert("end", "Memulai komputasi Ephemeris untuk penentuan 1 SYAWAL (1447 H - 1496 H)...\nMohon tunggu...\n\n"))
11483 |
11484 |

```

```

11485 |     output_lines = []
11486 |     output_lines.append(self.get_header(125))
11487 |     output_lines.append("[ REKAPITULASI KOMPARASI AWAL SYAWAL / IDUL FITRI (1447 H - 1496 H) ]".center(125))
11488 |     output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokal Sabang Alt>=3°, Eln>=6.4°)".center(125))
11489 |     output_lines.append("=" * 125)
11490 |     output_lines.append(f'{"Tahun":<8} | {"Waktu Ijtimak (UTC)":<19} | {"1 Syawal (KHGT)":<20} | {"1 Syawal (MABIMS Sabang)":<28} | {"Status / Selisih"}')
11491 |     output_lines.append("-" * 125)
11492 |
11493 |     # Setup Observer Sabang untuk Neo MABIMS
11494 |     sabang = ephem.Observer()
11495 |     sabang.lat = math.radians(5.8942)
11496 |     sabang.lon = math.radians(95.3184)
11497 |     sabang.elevation = 0
11498 |     sabang.pressure = 1010
11499 |     sabang.temp = 25
11500 |
11501 |     matahari = ephem.Sun()
11502 |     bulan = ephem.Moon()
11503 |
11504 |     # Iterasi dari 1447 H sampai 1496 H
11505 |     for thn_h in range(1447, 1497):
11506 |         try:
11507 |             # 1. Ambil data 1 Syawal KHGT dari Database Bawaan (Index 9 adalah Syawal)
11508 |             data_syawal = HIJRI_DB[thn_h][9]
11509 |             tgl_khgt_str = data_syawal[2] # Format: DD-MMM-YYYY
11510 |             dt_khgt = datetime.datetime.strptime(tgl_khgt_str, "%d-%b-%Y")
11511 |
11512 |             # 2. Tanggal Rukyat (29 Ramadan) adalah 1 hari sebelum 1 Syawal KHGT
11513 |             dt_rukyat = dt_khgt - datetime.timedelta(days=1)
11514 |
11515 |             # 3. Cari Waktu Ijtimak Akhir Ramadan
11516 |             waktu_pencarian = ephem.Date(dt_rukyat)
11517 |             ijtimak = ephem.previous_new_moon(waktu_pencarian + 5)
11518 |             dt_ijtimak_utc = ijtimak.datetime()
11519 |             str_ijtimak = dt_ijtimak_utc.strftime("%d-%m-%Y %H:%M")
11520 |
11521 |             # 4. Kalkulasi Sunset di Sabang pada hari Rukyat
11522 |             sabang.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
11523 |             try:
11524 |                 waktu_sunset = sabang.next_setting(matahari)
11525 |             except:
11526 |                 waktu_sunset = ephem.Date(dt_rukyat.replace(hour=11, minute=30, second=0))
11527 |
11528 |             # 5. Hitung Posisi Bulan saat Sunset Sabang
11529 |             sabang.date = waktu_sunset
11530 |             matahari.compute(sabang)
11531 |             bulan.compute(sabang)
11532 |
11533 |             alt_topo = math.degrees(bulan.alt)
11534 |             elong_topo = math.degrees(ephem.separation(matahari, bulan))
11535 |
11536 |             # 6. Evaluasi Neo MABIMS
11537 |             umur_bulan = waktu_sunset - ijtimak
11538 |
11539 |             if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
11540 |                 dt_mabims = dt_khgt
11541 |                 status = "Serentak (Sama)"
11542 |             else:
11543 |                 # Jika MABIMS Sabang belum tembus, maka Idul Fitri mundur 1 hari (Ramadan istikmal 30 hari)
11544 |                 dt_mabims = dt_khgt + datetime.timedelta(days=1)
11545 |                 status = "Beda (MABIMS Mundur 1 Hari)"
11546 |
11547 |             # 7. Format Output Tabel
11548 |             str_khgt = dt_khgt.strftime("%d %b %Y")
11549 |             str_mabims = dt_mabims.strftime("%d %b %Y")
11550 |
11551 |             baris = f"{thn_h} H | {str_ijtimak:<19} | {str_khgt:<20} | {str_mabims:<28} | {status}"
11552 |             output_lines.append(baris)
11553 |
11554 |             # Sisipkan pemisah per 10 Tahun
11555 |             if (thn_h - 1446) % 10 == 0 and thn_h != 1496:

```



```

11556 |         output_lines.append("-" * 125)
11557 |
11558 |     except Exception as e:
11559 |         output_lines.append(f"{thn_h} H    | Error kalkulasi: {str(e)}")
11560 |
11561 |     output_lines.append("=" * 125)
11562 |     output_lines.append("** Catatan Metodologi Penentuan 1 Syawal:")
11563 |     output_lines.append("- KHGT menggunakan referensi Global (Kesatuan Matlak). Diambil dari struktur HIJRI_DB.")
11564 |     output_lines.append("- Neo MABIMS menggunakan uji coba Toposentrik di Ufuk Barat Indonesia (Sabang, Aceh).")
11565 |     output_lines.append("- Jika status 'Beda (MABIMS Mundur 1 Hari)', artinya KHGT sudah merayakan Idul Fitri, sedangkan wilayah MABIMS masih berpuasa (istikmal).")
11566 |
11567 |     laporan_final = "\n".join(output_lines)
11568 |
11569 |     # Lempar ke Main Thread
11570 |     self.after(0, lambda: self.textbox.insert("1.0", laporan_final))
11571 |     self.after(0, lambda: self.textbox.configure(state="disabled"))
11572 |     self.after(0, lambda: self.lbl_status.configure(text="Komparasi 50 Tahun Syawal Selesai", text_color="#00E676"))
11573 |     self.after(0, lambda: self.btn_hitung.configure(state="normal"))
11574 |
11575 | def analisis_komparasi_zulhijah_50_tahun(self):
11576 |     threading.Thread(target=self._proses_komparasi_zulhijah, daemon=True).start()
11577 |
11578 | def _proses_komparasi_zulhijah(self):
11579 |     # Update UI Status
11580 |     self.after(0, lambda: self.lbl_status.configure(text="Menganalisis 50 Tahun Zulhijah...", text_color="#00E5FF"))
11581 |     self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
11582 |     self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
11583 |     self.after(0, lambda: self.textbox.delete("1.0", "end"))
11584 |     self.after(0, lambda: self.textbox.insert("end", "Memulai komputasi Ephemeris untuk penentuan 1 ZULHIJAH (1447 H - 1496 H)...\nMohon tunggu...\n\n"))
11585 |
11586 |     output_lines = []
11587 |     output_lines.append(self.get_header(125))
11588 |     output_lines.append("[ REKAPITULASI KOMPARASI AWAL ZULHIJAH / IDUL ADHA (1447 H - 1496 H) ]".center(125))
11589 |     output_lines.append("KHGT (Global Alt>=5°, Eln>=8°) vs Neo MABIMS (Lokal Sabang Alt>=3°, Eln>=6.4°)".center(125))
11590 |     output_lines.append("=" * 125)
11591 |     output_lines.append(f"{'Tahun':<8} | {'Waktu Ijtimak (UTC)':<19} | {'1 Zulhijah (KHGT)':<20} | {'1 Zulhijah (MABIMS Sabang)':<28} | {'Status / Selisih'}")
11592 |     output_lines.append("-" * 125)
11593 |
11594 |     # Setup Observer Sabang untuk Neo MABIMS
11595 |     sabang = ephem.Observer()
11596 |     sabang.lat = math.radians(5.8942)
11597 |     sabang.lon = math.radians(95.3184)
11598 |     sabang.elevation = 0
11599 |     sabang.pressure = 1010
11600 |     sabang.temp = 25
11601 |
11602 |     matahari = ephem.Sun()
11603 |     bulan = ephem.Moon()
11604 |
11605 |     # Iterasi dari 1447 H sampai 1496 H
11606 |     for thn_h in range(1447, 1497):
11607 |         try:
11608 |             # 1. Ambil data 1 Zulhijah KHGT dari Database Bawaan (Index 11 adalah Zulhijah)
11609 |             data_zulhijah = HIJRI_DB[thn_h][11]
11610 |             tgl_khgt_str = data_zulhijah[2] # Format: DD-MMM-YYYY
11611 |             dt_khgt = datetime.datetime.strptime(tgl_khgt_str, "%d-%b-%Y")
11612 |
11613 |             # 2. Tanggal Rukyat (29 Zulkaidah) adalah 1 hari sebelum 1 Zulhijah KHGT
11614 |             dt_rukyat = dt_khgt - datetime.timedelta(days=1)
11615 |
11616 |             # 3. Cari Waktu Ijtimak Akhir Zulkaidah
11617 |             waktu_pencarian = ephem.Date(dt_rukyat)
11618 |             ijtimak = ephem.previous_new_moon(waktu_pencarian + 5)
11619 |             dt_ijtimak_utc = ijtimak.datetime()
11620 |             str_ijtimak = dt_ijtimak_utc.strftime("%d-%m-%Y %H:%M")
11621 |
11622 |             # 4. Kalkulasi Sunset di Sabang pada hari Rukyat
11623 |             sabang.date = ephem.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
11624 |             try:
11625 |                 waktu_sunset = sabang.next_setting(matahari)
11626 |             except:

```

```

11627 |         waktu_sunset = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))
11628 |
11629 |         # 5. Hitung Posisi Bulan saat Sunset Sabang
11630 |         sabang.date = waktu_sunset
11631 |         matahari.compute(sabang)
11632 |         bulan.compute(sabang)
11633 |
11634 |         alt_topo = math.degrees(bulan.alt)
11635 |         elong_topo = math.degrees(ephemeris.separation(matahari, bulan))
11636 |
11637 |         # 6. Evaluasi Neo MABIMS
11638 |         umur_bulan = waktu_sunset - ijtima
11639 |
11640 |         if umur_bulan > 0 and alt_topo >= 3.0 and elong_topo >= 6.4:
11641 |             dt_mabims = dt_khgt
11642 |             status = "Serentak (Sama)"
11643 |         else:
11644 |             # Jika MABIMS Sabang belum tembus, maka Idul Adha mundur 1 hari (Zulkaidah istikmal 30 hari)
11645 |             dt_mabims = dt_khgt + datetime.timedelta(days=1)
11646 |             status = "Beda (MABIMS Mundur 1 Hari)"
11647 |
11648 |         # 7. Format Output Tabel
11649 |         str_khgt = dt_khgt.strftime("%d %b %Y")
11650 |         str_mabims = dt_mabims.strftime("%d %b %Y")
11651 |
11652 |         baris = f"{thn_h} H | {str_ijtimak:<19} | {str_khgt:<20} | {str_mabims:<28} | {status}"
11653 |         output_lines.append(baris)
11654 |
11655 |         # Sisipkan pemisah per 10 Tahun
11656 |         if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
11657 |             output_lines.append("-" * 125)
11658 |
11659 |     except Exception as e:
11660 |         output_lines.append(f"{thn_h} H | Error kalkulasi: {str(e)}")
11661 |
11662 |     output_lines.append("-" * 125)
11663 |     output_lines.append("* Catatan Metodologi Penentuan 1 Zulhijah:")
11664 |     output_lines.append("- KHGT menggunakan referensi Global (Kesatuan Matlak). Diambil dari struktur HIJRI_DB.")
11665 |     output_lines.append("- Neo MABIMS menggunakan uji coba Toposentrik di Ufuk Barat Indonesia (Sabang, Aceh).")
11666 |     output_lines.append("- Jika status 'Beda (MABIMS Mundur 1 Hari)', artinya KHGT sudah wukuf/Idul Adha lebih dulu, sedangkan MABIMS istikmal Zulkaidah.")
11667 |
11668 |     laporan_final = "\n".join(output_lines)
11669 |
11670 |     # Lempar ke Main Thread
11671 |     self.after(0, lambda: self.textbox.insert("1.0", laporan_final))
11672 |     self.after(0, lambda: self.textbox.configure(state="disabled"))
11673 |     self.after(0, lambda: self.lbl_status.configure(text="Komparasi 50 Tahun Zulhijah Selesai", text_color="#00E676"))
11674 |     self.after(0, lambda: self.btn_hitung.configure(state="normal"))
11675 |
11676 | def analisis_komparasi_global_50_tahun(self):
11677 |     """Memicu perhitungan komparasi 12 bulan dalam thread terpisah"""
11678 |     threading.Thread(target=self._proses_komparasi_global_50_tahun, daemon=True).start()
11679 |
11680 | def _proses_komparasi_global_50_tahun(self):
11681 |     self.after(0, lambda: self.lbl_status.configure(text="Audit Global 12 Bulan (50 Thn)...", text_color="#00E5FF"))
11682 |     self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
11683 |     self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
11684 |     self.after(0, lambda: self.textbox.delete("1.0", "end"))
11685 |     self.after(0, lambda: self.textbox.insert("end", "Memproses data 600 bulan Hijriah (1447H-1496H)...\nMohon tunggu...\n\n"))
11686 |
11687 |     output_lines = []
11688 |     output_lines.append(self.get_header(130))
11689 |     output_lines.append("[ ANALISIS PROBABILITAS GLOBAL & KOMPARASI 12 BULAN (1447 H - 1496 H) ]".center(130))
11690 |     output_lines.append("KHGT (Global Alt>=5, Eln>=8) vs Neo MABIMS (Lokal Sabang Alt>=3, Eln>=6.4)".center(130))
11691 |     output_lines.append("-" * 130)
11692 |     output_lines.append(f"{'Tahun':<7} | {'Bulan Hijriah':<15} | {'1 KHGT (Masehi)':<18} | {'1 MABIMS (Sabang)':<20} | {'Status'}")
11693 |     output_lines.append("-" * 130)
11694 |
11695 |     total_bulan, total_serentak, total_beda = 0, 0, 0
11696 |     sabang = ephemeris.Observer()
11697 |     sabang.lat, sabang.lon = math.radians(5.8942), math.radians(95.3184)

```

```

11698 |         sabang.pressure, sabang.temp = 1010, 25
11699 |         sun, moon = ephemeris.Sun(), ephemeris.Moon()
11700 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
11701 |
11702 |         for thn_h in range(1447, 1497):
11703 |             if thn_h not in HIJRI_DB: continue
11704 |             output_lines.append(f"--- TAHUN {thn_h} H ---")
11705 |             for m_idx in range(12):
11706 |                 try:
11707 |                     total_bulan += 1
11708 |                     db_entry = HIJRI_DB[thn_h][m_idx]
11709 |                     nama_bln = db_entry[0]
11710 |                     parts = db_entry[2].split('-')
11711 |                     d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])
11712 |                     dt_khgt = datetime.datetime(y, m, d)
11713 |                     dt_rukyat = dt_khgt - datetime.timedelta(days=1)
11714 |                     ijtimaq = ephemeris.previous_new_moon(ephemeris.Date(dt_rukyat) + 5)
11715 |                     sabang.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
11716 |                     try: ss_sabang = sabang.next_setting(sun)
11717 |                     except: ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30))
11718 |                     sabang.date = ss_sabang
11719 |                     sun.compute(sabang); moon.compute(sabang)
11720 |                     alt_t, eln_t = math.degrees(moon.alt), math.degrees(ephemeris.separation(sun, moon))
11721 |                     if (ss_sabang - ijtimaq)*24 > 0 and alt_t >= 3.0 and eln_t >= 6.4:
11722 |                         dt_mabims, status, total_serentak = dt_khgt, "Serentak", total_serentak + 1
11723 |                     else:
11724 |                         dt_mabims, status, total_beda = dt_khgt + datetime.timedelta(days=1), "Beda", total_beda + 1
11725 |                     output_lines.append(f"{'':<7} | {nama_bln:<15} | {dt_khgt.strftime('%d %b %Y'):<18} | {dt_mabims.strftime('%d %b %Y'):<20} | {status}")
11726 |                 except: continue
11727 |             output_lines.append("-" * 130)
11728 |
11729 |         prob = (total_serentak / total_bulan * 100) if total_bulan > 0 else 0
11730 |         res_summary = (f"\n{' '*130}\nRINGKASAN STATISTIK (50 TAHUN):\n"
11731 |                        f"- Total Bulan: {total_bulan} | Serentak: {total_serentak} | Berbeda: {total_beda}\n"
11732 |                        f"- Probabilitas Keseragaman Global: {prob:.2f}%\n{' '*130}")
11733 |         output_lines.append(res_summary)
11734 |         self.after(0, lambda: self.display_result("\n".join(output_lines)))
11735 |
11736 |     def _proses_tabel_elongasi_50_tahun(self):
11737 |         # 1. Fungsi Pembantu untuk update GUI (Thread-Safe / Anti-Kosong)
11738 |         def _set_ui_loading():
11739 |             self.lbl_status.configure(text="Menganalisis Elongasi 50 Tahun...", text_color="#00E5FF")
11740 |             self.textbox.configure(state="normal", wrap="none")
11741 |             self.textbox.delete("1.0", "end")
11742 |             self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Elongasi Awal Ramadhan)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
11743 |             self.textbox.configure(state="disabled")
11744 |
11745 |         def _set_ui_done(hasil_teks):
11746 |             self.textbox.configure(state="normal")
11747 |             self.textbox.delete("1.0", "end")
11748 |             self.textbox.insert("1.0", hasil_teks)
11749 |             self.textbox.configure(state="disabled")
11750 |             self.lbl_status.configure(text="Kalkulasi Elongasi Ramadhan Selesai", text_color="#00E676")
11751 |             self.btn_hitung.configure(state="normal")
11752 |
11753 |         def _set_ui_error(err_teks):
11754 |             self.textbox.configure(state="normal")
11755 |             self.textbox.delete("1.0", "end")
11756 |             self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
11757 |             self.textbox.configure(state="disabled")
11758 |             self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
11759 |             self.btn_hitung.configure(state="normal")
11760 |
11761 |         # Panggil status loading ke layar
11762 |         self.after(0, _set_ui_loading)
11763 |
11764 |         # 2. Proses Komputasi Utama
11765 |         try:
11766 |             output_lines = []
11767 |             output_lines.append(self.get_header(148))
11768 |             output_lines.append("[ DATA ELONGASI HILAL 50 TAHUN (RAMADHAN 1447H - 1496H) ]".center(148))

```

```

11769 |         output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Ramadhan di Sabang)".center(148))
11770 |         output_lines.append("=" * 148)
11771 |         # PERBAIKAN HEADER TABEL AGAR JELAS DAN LEBAR
11772 |         output_lines.append(f"{'Tahun':<7}| {'Ijtimak (UTC)':<17}| {'Waktu Referensi (LT Sabang)':<27}| {'KHGT (Geosentrik)':<25}| {'MABIMS (Toposentrik)':<25}| {'Status Awal Ramadhan'}}")
11773 |         output_lines.append("-" * 148)
11774 |
11775 |         # Dictionary untuk parsing tanggal aman
11776 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
11777 |                      "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
11778 |
11779 |         import math
11780 |         import ephemeris
11781 |         import pytz
11782 |
11783 |         # Setup Observer untuk Sabang (Sebagai referensi rukyat lokal di Indonesia)
11784 |         engine = ephemeris.Observer()
11785 |         engine.lat = math.radians(5.8942)
11786 |         engine.lon = math.radians(95.3184)
11787 |         engine.elevation = 0
11788 |         engine.pressure = 1010
11789 |         engine.temp = 25
11790 |
11791 |         sun = ephemeris.Sun()
11792 |         moon = ephemeris.Moon()
11793 |
11794 |         for thn_h in range(1447, 1497):
11795 |             try:
11796 |                 if thn_h not in HIJRI_DB:
11797 |                     continue
11798 |
11799 |                 # INDEX 8 ADALAH BULAN RAMADHAN
11800 |                 data_ramadan = HIJRI_DB[thn_h][8]
11801 |
11802 |                 # Parsing tanggal KHGT
11803 |                 parts = data_ramadan[2].split('.')
11804 |                 d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])
11805 |
11806 |                 dt_khgt = datetime.datetime(y, m, d)
11807 |
11808 |                 # PERBAIKAN: Hari Rukyat adalah 29 Syakban (H-1 sebelum 1 Ramadhan versi KHGT)
11809 |                 dt_rukyat = dt_khgt - datetime.timedelta(days=1)
11810 |
11811 |                 # Cari waktu Ijtimak
11812 |                 t_ref = ephemeris.Date(dt_rukyat)
11813 |                 ijtimak = ephemeris.previous_new_moon(t_ref + 2)
11814 |                 str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")
11815 |
11816 |                 # Evaluasi Sunset di Sabang pada Hari Rukyat
11817 |                 engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
11818 |                 try:
11819 |                     ss_sabang = engine.next_setting(sun)
11820 |                 except:
11821 |                     # Fallback jika error (Anomali ephemeris), asumsi jam 18:30 LT (11:30 UTC)
11822 |                     ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))
11823 |
11824 |                 # Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
11825 |                 dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
11826 |                 str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")
11827 |
11828 |                 # =====
11829 |                 # A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
11830 |                 # =====
11831 |                 engine.date = ss_sabang
11832 |                 sun.compute(engine)
11833 |                 moon.compute(engine)
11834 |
11835 |                 alt_topo = math.degrees(moon.alt)
11836 |                 eln_topo = math.degrees(ephemeris.separation(sun, moon))
11837 |                 umur_s = (ss_sabang - ijtimak) * 24
11838 |
11839 |                 if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:

```

```

11840 |         kesimpulan = "Serentak (Sama)"
11841 |     else:
11842 |         kesimpulan = "Beda (MABIMS Mundur 1 Hari)"
11843 |
11844 |     # Format Elongasi ditaruh di depan untuk Menu 23
11845 |     str_sabang = f"Eln: {eln_topo:5.2f}°, Alt: {alt_topo:5.2f}°"
11846 |
11847 |     # =====
11848 |     # B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
11849 |     # =====
11850 |     # Kita ukur parameter Geosentrik menggunakan Skyfield tepat di detik Sunset Sabang
11851 |     try:
11852 |         t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
11853 |         obs_center = self.eph['earth'].at(t_sunset_sky)
11854 |
11855 |         s_app_geo = obs_center.observe(self.eph['sun']).apparent()
11856 |         m_app_geo = obs_center.observe(self.eph['moon']).apparent()
11857 |
11858 |         # Elongasi Geosentris
11859 |         eln_geo = s_app_geo.separation_from(m_app_geo).degrees
11860 |
11861 |         # Altitude Geosentris Relatif Terhadap Sabang
11862 |         gmst = t_sunset_sky.gast
11863 |         lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
11864 |         ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)
11865 |
11866 |         ha_deg = lst_deg - (ra_m.hours * 15.0)
11867 |         lat_rad = math.radians(5.8942) # Lintang Sabang
11868 |         d_rad = dec_m.radians
11869 |         ha_rad = math.radians(ha_deg)
11870 |
11871 |         sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
11872 |         alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
11873 |
11874 |         str_khgt = f"Eln: {eln_geo:5.2f}°, Alt: {alt_geo:5.2f}°"
11875 |     except Exception as e_geo:
11876 |         str_khgt = "Error Kalkulasi Geo"
11877 |
11878 |     # Menulis baris ke laporan
11879 |     output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} | {str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")
11880 |
11881 |     # Garis pembatas per dekade
11882 |     if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
11883 |         output_lines.append("-" * 148)
11884 |
11885 |     except Exception as err_baris:
11886 |         output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")
11887 |
11888 |     output_lines.append("=" * 148)
11889 |     output_lines.append("* Catatan Pembacaan Tabel:")
11890 |     output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29 Syakban (H-1 sebelum 1 Ramadhan versi KHGT).")
11891 |     output_lines.append("2. [KHGT Geosentrik]: Mengukur Elongasi dan Tinggi Hilal dari pusat Bumi (Geocentric) pada saat Waktu Referensi tersebut.")
11892 |     output_lines.append("3. [MABIMS Toposentrik]: Mengukur Elongasi dan Tinggi Hilal dari permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
11893 |     output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga harus istikmal (genap 30 hari).")
11894 |
11895 |     # --- MENGIRIM TEKS KE LAYAR ---
11896 |     final_text = "\n".join(output_lines)
11897 |     self.after(0, lambda: _set_ui_done(final_text))
11898 |
11899 |     except Exception as e:
11900 |         import traceback
11901 |         self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))
11902 |
11903 | def _proses_tabel_ketinggian_50_tahun(self):
11904 |     # 1. Fungsi Pembantu untuk update GUI (Thread-Safe / Anti-Kosong)
11905 |     def _set_ui_loading():
11906 |         self.lbl_status.configure(text="Menganalisis Ketinggian 50 Tahun...", text_color="#00E5FF")
11907 |         self.textbox.configure(state="normal", wrap="none")
11908 |         self.textbox.delete("1.0", "end")
11909 |         self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Ketinggian Awal Ramadhan)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
11910 |         self.textbox.configure(state="disabled")

```

```

11911 |
11912 |     def _set_ui_done(hasil_teks):
11913 |         self.textbox.configure(state="normal")
11914 |         self.textbox.delete("1.0", "end")
11915 |         self.textbox.insert("1.0", hasil_teks)
11916 |         self.textbox.configure(state="disabled")
11917 |         self.lbl_status.configure(text="Kalkulasi Ketinggian Ramadhan Selesai", text_color="#00E676")
11918 |         self.btn_hitung.configure(state="normal")
11919 |
11920 |     def _set_ui_error(err_teks):
11921 |         self.textbox.configure(state="normal")
11922 |         self.textbox.delete("1.0", "end")
11923 |         self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
11924 |         self.textbox.configure(state="disabled")
11925 |         self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
11926 |         self.btn_hitung.configure(state="normal")
11927 |
11928 |     # Panggil status loading ke layar
11929 |     self.after(0, _set_ui_loading)
11930 |
11931 |     # 2. Proses Komputasi Utama
11932 |     try:
11933 |         output_lines = []
11934 |         output_lines.append(self.get_header(148))
11935 |         output_lines.append("[ DATA KETINGGIAN HILAL 50 TAHUN (RAMADHAN 1447H - 1496H) ]".center(148))
11936 |         output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Ramadhan di Sabang)".center(148))
11937 |         output_lines.append("=" * 148)
11938 |         # PERBAIKAN HEADER TABEL AGAR JELAS
11939 |         output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} | {'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Ramadhan'}")
11940 |         output_lines.append("-" * 148)
11941 |
11942 |         # Dictionary untuk parsing tanggal aman
11943 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
11944 |                      "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
11945 |
11946 |         import math
11947 |         import ephemeris
11948 |         import pytz
11949 |
11950 |         # Setup Observer untuk Sabang (Sebagai referensi rukyat lokal di Indonesia)
11951 |         engine = ephemeris.Observer()
11952 |         engine.lat = math.radians(5.8942)
11953 |         engine.lon = math.radians(95.3184)
11954 |         engine.elevation = 0
11955 |         engine.pressure = 1010
11956 |         engine.temp = 25
11957 |
11958 |         sun = ephemeris.Sun()
11959 |         moon = ephemeris.Moon()
11960 |
11961 |         for thn_h in range(1447, 1497):
11962 |             try:
11963 |                 if thn_h not in HIJRI_DB:
11964 |                     continue
11965 |
11966 |                 # INDEX 8 ADALAH BULAN RAMADHAN
11967 |                 data_ramadan = HIJRI_DB[thn_h][8]
11968 |
11969 |                 # Parsing tanggal KHGT
11970 |                 parts = data_ramadan[2].split('-')
11971 |                 d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])
11972 |
11973 |                 dt_khgt = datetime.datetime(y, m, d)
11974 |
11975 |                 # PERBAIKAN: Hari Rukyat adalah 29 Syakban (H-1 sebelum 1 Ramadhan versi KHGT)
11976 |                 dt_rukyat = dt_khgt - datetime.timedelta(days=1)
11977 |
11978 |                 # Cari waktu Ijtimak
11979 |                 t_ref = ephemeris.Date(dt_rukyat)
11980 |                 ijtimak = ephemeris.previous_new_moon(t_ref + 2)
11981 |                 str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")

```

```

11982 |
11983 | # Evaluasi Sunset di Sabang pada Hari Rukyat
11984 | engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
11985 | try:
11986 |     ss_sabang = engine.next_setting(sun)
11987 | except:
11988 |     # Fallback jika error, jam 18:30 LT (11:30 UTC)
11989 |     ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))
11990 |
11991 | # Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
11992 | dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
11993 | str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")
11994 |
11995 | # =====
11996 | # A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
11997 | # =====
11998 | engine.date = ss_sabang
11999 | sun.compute(engine)
12000 | moon.compute(engine)
12001 |
12002 | alt_topo = math.degrees(moon.alt)
12003 | eln_topo = math.degrees(ephemeris.separation(sun, moon))
12004 | umur_s = (ss_sabang - ijtihak) * 24
12005 |
12006 | if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
12007 |     status_mabims = "Lolos"
12008 |     kesimpulan = "Serentak (Sama)"
12009 | else:
12010 |     status_mabims = "Gagal"
12011 |     kesimpulan = "Beda (MABIMS Mundur 1 Hari)"
12012 |
12013 | str_sabang = f"Alt: {alt_topo:5.2f}°, Eln: {eln_topo:5.2f}°"
12014 |
12015 | # =====
12016 | # B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
12017 | # =====
12018 | # Kita ukur parameter Geosentrik menggunakan Skyfield tepat di detik Sunset Sabang
12019 | try:
12020 |     t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
12021 |     obs_center = self.eph['earth'].at(t_sunset_sky)
12022 |
12023 |     s_app_geo = obs_center.observe(self.eph['sun']).apparent()
12024 |     m_app_geo = obs_center.observe(self.eph['moon']).apparent()
12025 |
12026 |     # Elongasi Geosentris
12027 |     eln_geo = s_app_geo.separation_from(m_app_geo).degrees
12028 |
12029 |     # Altitude Geosentris Relatif Terhadap Sabang
12030 |     gmst = t_sunset_sky.gmst
12031 |     lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
12032 |     ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)
12033 |
12034 |     ha_deg = lst_deg - (ra_m.hours * 15.0)
12035 |     lat_rad = math.radians(5.8942) # Lintang Sabang
12036 |     d_rad = dec_m.radians
12037 |     ha_rad = math.radians(ha_deg)
12038 |
12039 |     sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
12040 |     alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
12041 |
12042 |     str_khgt = f"Alt: {alt_geo:5.2f}°, Eln: {eln_geo:5.2f}°"
12043 | except Exception as e_geo:
12044 |     str_khgt = "Error Kalkulasi Geo"
12045 |
12046 | # Menulis baris ke laporan
12047 | output_lines.append(f"{thn_h} H | {str_ijtihak:<17} | {str_waktu_referensi:<27} | {str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")
12048 |
12049 | # Garis pembatas per dekade
12050 | if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
12051 |     output_lines.append("-" * 148)
12052 |

```

```

12053 |         except Exception as err_baris:
12054 |             output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")
12055 |
12056 |         output_lines.append("=" * 148)
12057 |         output_lines.append("** Catatan Pembacaan Tabel:")
12058 |         output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29 Syakban (H-1 sebelum 1 Ramadhan versi KHGT).")
12059 |         output_lines.append("2. [KHGT Geosentrik]: Mengukur Tinggi Hilal dan Elongasi dari pusat Bumi (Geocentric) pada saat Waktu Referensi tersebut.")
12060 |         output_lines.append("3. [MABIMS Toposentrik]: Mengukur Tinggi Hilal dan Elongasi dari permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
12061 |         output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga harus istikmal (genap 30 hari).")
12062 |
12063 |         # --- MENGIRIM TEKS KE LAYAR ---
12064 |         final_text = "\n".join(output_lines)
12065 |         self.after(0, lambda: _set_ui_done(final_text))
12066 |
12067 |     except Exception as e:
12068 |         import traceback
12069 |         self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))
12070 |
12071 | def _proses_tabel_ketinggian_syawal_50_tahun(self):
12072 |     # 1. Fungsi Pembantu untuk update GUI (Thread-Safe)
12073 |     def _set_ui_loading():
12074 |         self.lbl_status.configure(text="Menganalisis Ketinggian Syawal...", text_color="#00E5FF")
12075 |         self.textbox.configure(state="normal", wrap="none")
12076 |         self.textbox.delete("1.0", "end")
12077 |         self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Ketinggian Awal Syawal)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
12078 |         self.textbox.configure(state="disabled")
12079 |
12080 |     def _set_ui_done(hasil_teks):
12081 |         self.textbox.configure(state="normal")
12082 |         self.textbox.delete("1.0", "end")
12083 |         self.textbox.insert("1.0", hasil_teks)
12084 |         self.textbox.configure(state="disabled")
12085 |         self.lbl_status.configure(text="Kalkulasi Ketinggian Syawal Selesai", text_color="#00E676")
12086 |         self.btn_hitung.configure(state="normal")
12087 |
12088 |     def _set_ui_error(err_teks):
12089 |         self.textbox.configure(state="normal")
12090 |         self.textbox.delete("1.0", "end")
12091 |         self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
12092 |         self.textbox.configure(state="disabled")
12093 |         self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
12094 |         self.btn_hitung.configure(state="normal")
12095 |
12096 |     self.after(0, _set_ui_loading)
12097 |
12098 |     # 2. Proses Komputasi Utama Syawal
12099 |     try:
12100 |         output_lines = []
12101 |         output_lines.append(self.get_header(148))
12102 |         output_lines.append("[ DATA KETINGGIAN HILAL 50 TAHUN (SYAWAL 1447H - 1496H) ]".center(148))
12103 |         output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Syawal di Sabang)".center(148))
12104 |         output_lines.append("=" * 148)
12105 |         # PERBAIKAN HEADER TABEL AGAR JELAS DAN LEBAR
12106 |         output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} | {'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Syawal'}")
12107 |         output_lines.append("-" * 148)
12108 |
12109 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
12110 |                      "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
12111 |
12112 |         import math
12113 |         import ephem
12114 |         import pytz
12115 |
12116 |         # Setup Observer untuk Sabang (Referensi MABIMS)
12117 |         engine = ephem.Observer()
12118 |         engine.lat = math.radians(5.8942)
12119 |         engine.lon = math.radians(95.3184)
12120 |         engine.elevation = 0
12121 |         engine.pressure = 1010
12122 |         engine.temp = 25
12123 |

```



```

12124 | sun = ephemeris.Sun()
12125 | moon = ephemeris.Moon()
12126 |
12127 | for thn_h in range(1447, 1497):
12128 |     try:
12129 |         if thn_h not in HIJRI_DB:
12130 |             continue
12131 |
12132 |         # INDEX 9 ADALAH BULAN SYAWAL
12133 |         data_syawal = HIJRI_DB[thn_h][9]
12134 |
12135 |         # Parsing tanggal KHGT
12136 |         parts = data_syawal[2].split('-')
12137 |         d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])
12138 |
12139 |         dt_khgt = datetime.datetime(y, m, d)
12140 |
12141 |         # Hari rukyat Syawal adalah tanggal 29 Ramadhan (H-1 dari 1 Syawal KHGT)
12142 |         dt_rukyat = dt_khgt - datetime.timedelta(days=1)
12143 |
12144 |         # Cari waktu Ijtima
12145 |         t_ref = ephemeris.Date(dt_rukyat)
12146 |         ijtima = ephemeris.previous_new_moon(t_ref + 2)
12147 |         str_ijtima = ijtima.datetime().strftime("%d-%b %H:%M")
12148 |
12149 |         # Evaluasi Sunset di Sabang pada akhir Ramadhan (Hari Rukyat)
12150 |         engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
12151 |         try:
12152 |             ss_sabang = engine.next_setting(sun)
12153 |         except:
12154 |             # Fallback jika error, jam 18:30 LT (11:30 UTC)
12155 |             ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))
12156 |
12157 |         # Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
12158 |         dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
12159 |         str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")
12160 |
12161 |         # =====
12162 |         # A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
12163 |         # =====
12164 |         engine.date = ss_sabang
12165 |         sun.compute(engine)
12166 |         moon.compute(engine)
12167 |
12168 |         alt_topo = math.degrees(moon.alt)
12169 |         eln_topo = math.degrees(ephemeris.separation(sun, moon))
12170 |         umur_s = (ss_sabang - ijtima) * 24
12171 |
12172 |         if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
12173 |             kesimpulan = "Serentak (Idul Fitri Sama)"
12174 |         else:
12175 |             kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"
12176 |
12177 |         # Format Altitude ditaruh di depan untuk Menu 24
12178 |         str_sabang = f"Alt: {alt_topo:5.2f}°, Eln: {eln_topo:5.2f}°"
12179 |
12180 |         # =====
12181 |         # B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
12182 |         # =====
12183 |         # Kita ukur parameter Geosentrik menggunakan Skyfield tepat di detik Sunset Sabang
12184 |         try:
12185 |             t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
12186 |             obs_center = self.eph['earth'].at(t_sunset_sky)
12187 |
12188 |             s_app_geo = obs_center.observe(self.eph['sun']).apparent()
12189 |             m_app_geo = obs_center.observe(self.eph['moon']).apparent()
12190 |
12191 |             # Elongasi Geosentris
12192 |             eln_geo = s_app_geo.separation_from(m_app_geo).degrees
12193 |
12194 |             # Altitude Geosentris Relatif Terhadap Sabang

```

```

12195 |         gmst = t.sunset_sky.gast
12196 |         lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
12197 |         ra_m, dec_m, _ = m_app_geo.radec(epoch=t.sunset_sky)
12198 |
12199 |         ha_deg = lst_deg - (ra_m.hours * 15.0)
12200 |         lat_rad = math.radians(5.8942) # Lintang Sabang
12201 |         d_rad = dec_m.radians
12202 |         ha_rad = math.radians(ha_deg)
12203 |
12204 |         sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
12205 |         alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
12206 |
12207 |         str_khgt = f"Alt: {alt_geo:5.2f}°, Eln: {eln_geo:5.2f}°"
12208 |     except Exception as e_geo:
12209 |         str_khgt = "Error Kalkulasi Geo"
12210 |
12211 |     # Menulis baris ke laporan
12212 |     output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} | {str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")
12213 |
12214 |     # Garis pembatas per dekade
12215 |     if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
12216 |         output_lines.append("-" * 148)
12217 |
12218 |     except Exception as err_baris:
12219 |         output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")
12220 |
12221 |     output_lines.append("=" * 148)
12222 |     output_lines.append("Catatan Pembacaan Tabel:")
12223 |     output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29 Ramadhan (H-1 sebelum 1 Syawal versi KHGT).")
12224 |     output_lines.append("2. [KHGT Geosentrik]: Mengukur Tinggi Hilal dan Elongasi dari pusat Bumi (Geocentric) pada saat Waktu Referensi tersebut.")
12225 |     output_lines.append("3. [MABIMS Toposentrik]: Mengukur Tinggi Hilal dan Elongasi dari permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
12226 |     output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga Ramadhan harus diistimkan (digenapkan 30
hari).")
12227 |
12228 |     # --- MENGIRIM TEKS KE LAYAR ---
12229 |     final_text = "\n".join(output_lines)
12230 |     self.after(0, lambda: _set_ui_done(final_text))
12231 |
12232 |     except Exception as e:
12233 |         import traceback
12234 |         self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))
12235 |
12236 | def _proses_tabel_ketinggian_zulhijah_50_tahun(self):
12237 |     # 1. Fungsi Pembantu untuk update GUI (Thread-Safe)
12238 |     def _set_ui_loading():
12239 |         self.lbl_status.configure(text="Menganalisis Ketinggian Zulhijah...", text_color="#00E5FF")
12240 |         self.textbox.configure(state="normal", wrap="none")
12241 |         self.textbox.delete("1.0", "end")
12242 |         self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Ketinggian Awal Zulhijah)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
12243 |         self.textbox.configure(state="disabled")
12244 |
12245 |     def _set_ui_done(hasil_teks):
12246 |         self.textbox.configure(state="normal")
12247 |         self.textbox.delete("1.0", "end")
12248 |         self.textbox.insert("1.0", hasil_teks)
12249 |         self.textbox.configure(state="disabled")
12250 |         self.lbl_status.configure(text="Kalkulasi Ketinggian Zulhijah Selesai", text_color="#00E676")
12251 |         self.btn_hitung.configure(state="normal")
12252 |
12253 |     def _set_ui_error(err_teks):
12254 |         self.textbox.configure(state="normal")
12255 |         self.textbox.delete("1.0", "end")
12256 |         self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
12257 |         self.textbox.configure(state="disabled")
12258 |         self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
12259 |         self.btn_hitung.configure(state="normal")
12260 |
12261 |     self.after(0, _set_ui_loading)
12262 |
12263 |     # 2. Proses Komputasi Utama Zulhijah
12264 |     try:

```

```

12265 |         output_lines = []
12266 |         output_lines.append(self.get_header(148))
12267 |         output_lines.append("[ DATA KETINGGIAN HILAL 50 TAHUN (ZULHIJAH 1447H - 1496H) ]".center(148))
12268 |         output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Zulhijah di Sabang)".center(148))
12269 |         output_lines.append("=" * 148)
12270 |         output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} | {'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Zulhijah'}}")
12271 |         output_lines.append("-" * 148)
12272 |
12273 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
12274 |                      "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
12275 |
12276 |         import math
12277 |         import ephemeris
12278 |         import pytz
12279 |
12280 |         # Setup Observer untuk Sabang (Referensi MABIMS)
12281 |         engine = ephemeris.Observer()
12282 |         engine.lat = math.radians(5.8942)
12283 |         engine.lon = math.radians(95.3184)
12284 |         engine.elevation = 0
12285 |         engine.pressure = 1010
12286 |         engine.temp = 25
12287 |
12288 |         sun = ephemeris.Sun()
12289 |         moon = ephemeris.Moon()
12290 |
12291 |         for thn_h in range(1447, 1497):
12292 |             try:
12293 |                 if thn_h not in HIJRI_DB:
12294 |                     continue
12295 |
12296 |                 # INDEX 11 ADALAH BULAN ZULHIJAH
12297 |                 data_zulhijah = HIJRI_DB[thn_h][11]
12298 |
12299 |                 # Parsing tanggal KHGT
12300 |                 parts = data_zulhijah[2].split('.')
12301 |                 d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])
12302 |
12303 |                 dt_khgt = datetime.datetime(y, m, d)
12304 |
12305 |                 # Hari rukyat Zulhijah adalah tanggal 29 Zulkaidah (H-1 dari 1 Zulhijah KHGT)
12306 |                 dt_rukyat = dt_khgt - datetime.timedelta(days=1)
12307 |
12308 |                 # Cari waktu Ijtimak
12309 |                 t_ref = ephemeris.Date(dt_rukyat)
12310 |                 ijtimak = ephemeris.previous_new_moon(t_ref + 2)
12311 |                 str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")
12312 |
12313 |                 # Evaluasi Sunset di Sabang pada akhir Zulkaidah (Hari Rukyat)
12314 |                 engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
12315 |                 try:
12316 |                     ss_sabang = engine.next_setting(sun)
12317 |                 except:
12318 |                     # Fallback jika error, jam 18:30 LT (11:30 UTC)
12319 |                     ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))
12320 |
12321 |                 # Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
12322 |                 dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
12323 |                 str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")
12324 |
12325 |                 # =====
12326 |                 # A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
12327 |                 # =====
12328 |                 engine.date = ss_sabang
12329 |                 sun.compute(engine)
12330 |                 moon.compute(engine)
12331 |
12332 |                 alt_topo = math.degrees(moon.alt)
12333 |                 eln_topo = math.degrees(ephemeris.separation(sun, moon))
12334 |                 umur_s = (ss_sabang - ijtimak) * 24
12335 |

```

```

12336 |         if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
12337 |             kesimpulan = "Serentak (Idul Adha Sama)"
12338 |         else:
12339 |             kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"
12340 |
12341 |         # Format Altitude ditaruh di depan untuk Menu Ketinggian
12342 |         str_sabang = f"Alt: {alt_topo:5.2f}°, Eln: {eln_topo:5.2f}°"
12343 |
12344 |         # =====
12345 |         # B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
12346 |         # =====
12347 |         try:
12348 |             t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
12349 |             obs_center = self.eph['earth'].at(t_sunset_sky)
12350 |
12351 |             s_app_geo = obs_center.observe(self.eph['sun']).apparent()
12352 |             m_app_geo = obs_center.observe(self.eph['moon']).apparent()
12353 |
12354 |             eln_geo = s_app_geo.separation_from(m_app_geo).degrees
12355 |
12356 |             gmst = t_sunset_sky.gast
12357 |             lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
12358 |             ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)
12359 |
12360 |             ha_deg = lst_deg - (ra_m.hours * 15.0)
12361 |             lat_rad = math.radians(5.8942) # Lintang Sabang
12362 |             d_rad = dec_m.radians
12363 |             ha_rad = math.radians(ha_deg)
12364 |
12365 |             sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
12366 |             alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
12367 |
12368 |             str_khgt = f"Alt: {alt_geo:5.2f}°, Eln: {eln_geo:5.2f}°"
12369 |         except Exception as e_geo:
12370 |             str_khgt = "Error Kalkulasi Geo"
12371 |
12372 |         # Menulis baris ke laporan
12373 |         output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} | {str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")
12374 |
12375 |         # Garis pembatas per dekade
12376 |         if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
12377 |             output_lines.append("-" * 148)
12378 |
12379 |         except Exception as err_baris:
12380 |             output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")
12381 |
12382 |         output_lines.append("=" * 148)
12383 |         output_lines.append("** Catatan Pembacaan Tabel:")
12384 |         output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29 Zulkaidah (H-1 sebelum 1 Zulhijah versi KHGT).")
12385 |         output_lines.append("2. [KHGT Geosentrik]: Mengukur Tinggi Hilal dan Elongasi dari pusat Bumi (Geocentric) pada saat Waktu Referensi tersebut.")
12386 |         output_lines.append("3. [MABIMS Toposentrik]: Mengukur Tinggi Hilal dan Elongasi dari permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
12387 |         output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga Zulkaidah harus diistimalkan.")
12388 |
12389 |         final_text = "\n".join(output_lines)
12390 |         self.after(0, lambda: _set_ui_done(final_text))
12391 |
12392 |     except Exception as e:
12393 |         import traceback
12394 |         self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))
12395 |
12396 |     def _proses_tabel_elongasi_syawal_50_tahun(self):
12397 |         # 1. Fungsi Pembantu untuk update GUI (Thread-Safe)
12398 |         def _set_ui_loading():
12399 |             self.lbl_status.configure(text="Menganalisis Elongasi Syawal...", text_color="#00E5FF")
12400 |             self.textbox.configure(state="normal", wrap="none")
12401 |             self.textbox.delete("1.0", "end")
12402 |             self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Elongasi Awal Syawal)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
12403 |             self.textbox.configure(state="disabled")
12404 |
12405 |         def _set_ui_done(hasil_teks):
12406 |             self.textbox.configure(state="normal")

```

```

12407 |         self.textbox.delete("1.0", "end")
12408 |         self.textbox.insert("1.0", hasil_teks)
12409 |         self.textbox.configure(state="disabled")
12410 |         self.lbl_status.configure(text="Kalkulasi Elongasi Syawal Selesai", text_color="#00E676")
12411 |         self.btn_hitung.configure(state="normal")
12412 |
12413 |     def _set_ui_error(err_teks):
12414 |         self.textbox.configure(state="normal")
12415 |         self.textbox.delete("1.0", "end")
12416 |         self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
12417 |         self.textbox.configure(state="disabled")
12418 |         self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
12419 |         self.btn_hitung.configure(state="normal")
12420 |
12421 |     self.after(0, _set_ui_loading)
12422 |
12423 |     # 2. Proses Komputasi Utama Syawal (Fokus Elongasi)
12424 |     try:
12425 |         output_lines = []
12426 |         output_lines.append(self.get_header(148))
12427 |         output_lines.append("[ DATA ELONGASI HILAL 50 TAHUN (SYAWAL 1447H - 1496H) ]".center(148))
12428 |         output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Syawal di Sabang)".center(148))
12429 |         output_lines.append("=" * 148)
12430 |         # PERBAIKAN HEADER TABEL AGAR JELAS DAN LEBAR
12431 |         output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} | {'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Syawal'}}")
12432 |         output_lines.append("-" * 148)
12433 |
12434 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
12435 |                      "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
12436 |
12437 |         import math
12438 |         import ephemeris
12439 |         import pytz
12440 |
12441 |         # Setup Observer untuk Sabang (Referensi MABIMS)
12442 |         engine = ephemeris.Observer()
12443 |         engine.lat = math.radians(5.8942)
12444 |         engine.lon = math.radians(95.3184)
12445 |         engine.elevation = 0
12446 |         engine.pressure = 1010
12447 |         engine.temp = 25
12448 |
12449 |         sun = ephemeris.Sun()
12450 |         moon = ephemeris.Moon()
12451 |
12452 |         for thn_h in range(1447, 1497):
12453 |             try:
12454 |                 if thn_h not in HIJRI_DB:
12455 |                     continue
12456 |
12457 |                 # INDEX 9 ADALAH BULAN SYAWAL
12458 |                 data_syawal = HIJRI_DB[thn_h][9]
12459 |
12460 |                 # Parsing tanggal KHGT
12461 |                 parts = data_syawal[2].split('-')
12462 |                 d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])
12463 |
12464 |                 dt_khgt = datetime.datetime(y, m, d)
12465 |
12466 |                 # Hari rukyat Syawal adalah tanggal 29 Ramadhan (H-1 dari 1 Syawal KHGT)
12467 |                 dt_rukyat = dt_khgt - datetime.timedelta(days=1)
12468 |
12469 |                 # Cari waktu Ijtimak
12470 |                 t_ref = ephemeris.Date(dt_rukyat)
12471 |                 ijtimak = ephemeris.previous_new_moon(t_ref + 2)
12472 |                 str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")
12473 |
12474 |                 # Evaluasi Sunset di Sabang pada akhir Ramadhan (Hari Rukyat)
12475 |                 engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
12476 |                 try:
12477 |                     ss_sabang = engine.next_setting(sun)

```

```

12478 |         except:
12479 |             # Fallback jika error, jam 18:30 LT (11:30 UTC)
12480 |             ss_sabang = ephemer.Date(dt_rukyat.replace(hour=11, minute=30, second=0))
12481 |
12482 |         # Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
12483 |         dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
12484 |         str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")
12485 |
12486 |         # =====
12487 |         # A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
12488 |         # =====
12489 |         engine.date = ss_sabang
12490 |         sun.compute(engine)
12491 |         moon.compute(engine)
12492 |
12493 |         alt_topo = math.degrees(moon.alt)
12494 |         eln_topo = math.degrees(ephemer.separation(sun, moon))
12495 |         umur_s = (ss_sabang - ijtihak) * 24
12496 |
12497 |         if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
12498 |             kesimpulan = "Serentak (Idul Fitri Sama)"
12499 |         else:
12500 |             kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"
12501 |
12502 |         # Format Elongasi ditaruh di depan untuk Menu 25
12503 |         str_sabang = f"Eln: {eln_topo:5.2f}°, Alt: {alt_topo:5.2f}°"
12504 |
12505 |         # =====
12506 |         # B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
12507 |         # =====
12508 |         # Kita ukur parameter Geosentrik menggunakan Skyfield tepat di detik Sunset Sabang
12509 |         try:
12510 |             t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
12511 |             obs_center = self.eph['earth'].at(t_sunset_sky)
12512 |
12513 |             s_app_geo = obs_center.observe(self.eph['sun']).apparent()
12514 |             m_app_geo = obs_center.observe(self.eph['moon']).apparent()
12515 |
12516 |             # Elongasi Geosentris
12517 |             eln_geo = s_app_geo.separation_from(m_app_geo).degrees
12518 |
12519 |             # Altitude Geosentris Relatif Terhadap Sabang
12520 |             gmst = t_sunset_sky.gast
12521 |             lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
12522 |             ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)
12523 |
12524 |             ha_deg = lst_deg - (ra_m.hours * 15.0)
12525 |             lat_rad = math.radians(5.8942) # Lintang Sabang
12526 |             d_rad = dec_m.radians
12527 |             ha_rad = math.radians(ha_deg)
12528 |
12529 |             sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
12530 |             alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
12531 |
12532 |             str_khgt = f"Eln: {eln_geo:5.2f}°, Alt: {alt_geo:5.2f}°"
12533 |         except Exception as e_geo:
12534 |             str_khgt = "Error Kalkulasi Geo"
12535 |
12536 |         # Menulis baris ke laporan
12537 |         output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} | {str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")
12538 |
12539 |         # Garis pembatas per dekade
12540 |         if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
12541 |             output_lines.append("-" * 148)
12542 |
12543 |         except Exception as err_baris:
12544 |             output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")
12545 |
12546 |         output_lines.append("=" * 148)
12547 |         output_lines.append("** Catatan Pembacaan Tabel:")
12548 |         output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29 Ramadhan (H-1 sebelum 1 Syawal versi KHGT).")

```

```

12549 |         output_lines.append("2. [KHGT Geosentrik]: Mengukur Elongasi dan Tinggi Hilal dari pusat Bumi (Geocentric) pada saat Waktu Referensi tersebut.")
12550 |         output_lines.append("3. [MABIMS Toposentrik]: Mengukur Elongasi dan Tinggi Hilal dari permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
12551 |         output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga Ramadhan harus diistimkan (digenapkan 30
hari).")
12552 |
12553 |         # --- MENGIRIM TEKS KE LAYAR ---
12554 |         final_text = "\n".join(output_lines)
12555 |         self.after(0, lambda: _set_ui_done(final_text))
12556 |
12557 |     except Exception as e:
12558 |         import traceback
12559 |         self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))
12560 |
12561 | def _proses_tabel_elongasi_zulhijah_50_tahun(self):
12562 |     # 1. Fungsi Pembantu untuk update GUI (Thread-Safe)
12563 |     def _set_ui_loading():
12564 |         self.lbl_status.configure(text="Menganalisis Elongasi Zulhijah...", text_color="#00E5FF")
12565 |         self.textbox.configure(state="normal", wrap="none")
12566 |         self.textbox.delete("1.0", "end")
12567 |         self.textbox.insert("1.0", "Memulai komputasi Ephemeris untuk 50 Tahun (Fokus Elongasi Awal Zulhijah)...\nMohon tunggu, sedang mengekstrak data Geosentrik & Toposentrik...\n")
12568 |         self.textbox.configure(state="disabled")
12569 |
12570 |     def _set_ui_done(hasil_teks):
12571 |         self.textbox.configure(state="normal")
12572 |         self.textbox.delete("1.0", "end")
12573 |         self.textbox.insert("1.0", hasil_teks)
12574 |         self.textbox.configure(state="disabled")
12575 |         self.lbl_status.configure(text="Kalkulasi Elongasi Zulhijah Selesai", text_color="#00E676")
12576 |         self.btn_hitung.configure(state="normal")
12577 |
12578 |     def _set_ui_error(err_teks):
12579 |         self.textbox.configure(state="normal")
12580 |         self.textbox.delete("1.0", "end")
12581 |         self.textbox.insert("1.0", f"TERJADI KESALAHAN SISTEM:\n{err_teks}")
12582 |         self.textbox.configure(state="disabled")
12583 |         self.lbl_status.configure(text="Error Kalkulasi", text_color="#FF1744")
12584 |         self.btn_hitung.configure(state="normal")
12585 |
12586 |     self.after(0, _set_ui_loading)
12587 |
12588 |     # 2. Proses Komputasi Utama Zulhijah (Fokus Elongasi)
12589 |     try:
12590 |         output_lines = []
12591 |         output_lines.append(self.get_header(148))
12592 |         output_lines.append("[ DATA ELONGASI HILAL 50 TAHUN (ZULHIJAH 1447H - 1496H) ].center(148))
12593 |         output_lines.append("Metode: Ekstraksi Nilai Aktual saat Waktu Referensi (Sunset H-1 Zulhijah di Sabang)".center(148))
12594 |         output_lines.append("=" * 148)
12595 |         # PERBAIKAN HEADER TABEL AGAR JELAS DAN LEBAR
12596 |         output_lines.append(f"{'Tahun':<7} | {'Ijtimak (UTC)':<17} | {'Waktu Referensi (LT Sabang)':<27} | {'KHGT (Geosentrik)':<25} | {'MABIMS (Toposentrik)':<25} | {'Status Awal Zulhijah'}")
12597 |         output_lines.append("-" * 148)
12598 |
12599 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6,
12600 |                      "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
12601 |
12602 |         import math
12603 |         import ephem
12604 |         import pytz
12605 |
12606 |         # Setup Observer untuk Sabang (Referensi MABIMS)
12607 |         engine = ephem.Observer()
12608 |         engine.lat = math.radians(5.8942)
12609 |         engine.lon = math.radians(95.3184)
12610 |         engine.elevation = 0
12611 |         engine.pressure = 1010
12612 |         engine.temp = 25
12613 |
12614 |         sun = ephem.Sun()
12615 |         moon = ephem.Moon()
12616 |
12617 |         for thn_h in range(1447, 1497):
12618 |             try:

```

```

12619 |         if thn_h not in HIJRI_DB:
12620 |             continue
12621 |
12622 |         # INDEX 11 ADALAH BULAN ZULHIJAH
12623 |         data_zulhijah = HIJRI_DB[thn_h][11]
12624 |
12625 |         # Parsing tanggal KHGT
12626 |         parts = data_zulhijah[2].split('-')
12627 |         d, m, y = int(parts[0]), bulan_map.get(parts[1], 1), int(parts[2])
12628 |
12629 |         dt_khgt = datetime.datetime(y, m, d)
12630 |
12631 |         # Hari rukyat Zulhijah adalah tanggal 29 Zulkaidah (H-1 dari 1 Zulhijah KHGT)
12632 |         dt_rukyat = dt_khgt - datetime.timedelta(days=1)
12633 |
12634 |         # Cari waktu Ijtimak
12635 |         t_ref = ephemeris.Date(dt_rukyat)
12636 |         ijtimak = ephemeris.previous_new_moon(t_ref + 2)
12637 |         str_ijtimak = ijtimak.datetime().strftime("%d-%b %H:%M")
12638 |
12639 |         # Evaluasi Sunset di Sabang pada akhir Zulkaidah (Hari Rukyat)
12640 |         engine.date = ephemeris.Date(dt_rukyat.replace(hour=0, minute=0, second=0))
12641 |         try:
12642 |             ss_sabang = engine.next_setting(sun)
12643 |         except:
12644 |             # Fallback jika error, jam 18:30 LT (11:30 UTC)
12645 |             ss_sabang = ephemeris.Date(dt_rukyat.replace(hour=11, minute=30, second=0))
12646 |
12647 |         # Format Tanggal & Waktu Referensi untuk ditampilkan (Waktu Lokal Sabang UTC+7)
12648 |         dt_sunset_lokal = ss_sabang.datetime() + datetime.timedelta(hours=7)
12649 |         str_waktu_referensi = dt_sunset_lokal.strftime("%d-%b-%Y, %H:%M:%S")
12650 |
12651 |         # =====
12652 |         # A. PERHITUNGAN TOPOSENTRIK MABIMS (Di Sabang saat Sunset)
12653 |         # =====
12654 |         engine.date = ss_sabang
12655 |         sun.compute(engine)
12656 |         moon.compute(engine)
12657 |
12658 |         alt_topo = math.degrees(moon.alt)
12659 |         eln_topo = math.degrees(ephemeris.separation(sun, moon))
12660 |         umur_s = (ss_sabang - ijtimak) * 24
12661 |
12662 |         if alt_topo >= 3.0 and eln_topo >= 6.4 and umur_s > 0:
12663 |             kesimpulan = "Serentak (Idul Adha Sama)"
12664 |         else:
12665 |             kesimpulan = "Beda (MABIMS Istikmal 30 Hr)"
12666 |
12667 |         # Format Elongasi ditaruh di depan untuk Menu Elongasi
12668 |         str_sabang = f"Eln: {eln_topo:5.2f}°, Alt: {alt_topo:5.2f}°"
12669 |
12670 |         # =====
12671 |         # B. PERHITUNGAN GEOSENTRIK KHGT PADA SAAT YANG SAMA
12672 |         # =====
12673 |         try:
12674 |             t_sunset_sky = self.ts.from_datetime(ss_sabang.datetime().replace(tzinfo=pytz.utc))
12675 |             obs_center = self.eph['earth'].at(t_sunset_sky)
12676 |
12677 |             s_app_geo = obs_center.observe(self.eph['sun']).apparent()
12678 |             m_app_geo = obs_center.observe(self.eph['moon']).apparent()
12679 |
12680 |             eln_geo = s_app_geo.separation_from(m_app_geo).degrees
12681 |
12682 |             gmst = t_sunset_sky.gast
12683 |             lst_deg = (gmst * 15.0) + 95.3184 # Bujur Sabang
12684 |             ra_m, dec_m, _ = m_app_geo.radec(epoch=t_sunset_sky)
12685 |
12686 |             ha_deg = lst_deg - (ra_m.hours * 15.0)
12687 |             lat_rad = math.radians(5.8942) # Lintang Sabang
12688 |             d_rad = dec_m.radians
12689 |             ha_rad = math.radians(ha_deg)

```



```

12690 |
12691 |         sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
12692 |         alt_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo))))
12693 |
12694 |         str_khgt = f"Eln: {eln_geo:5.2f}°, Alt: {alt_geo:5.2f}°"
12695 |     except Exception as e_geo:
12696 |         str_khgt = "Error Kalkulasi Geo"
12697 |
12698 |         # Menulis baris ke laporan
12699 |         output_lines.append(f"{thn_h} H | {str_ijtimak:<17} | {str_waktu_referensi:<27} | {str_khgt:<25} | {str_sabang:<25} | {kesimpulan}")
12700 |
12701 |         # Garis pembatas per dekade
12702 |         if (thn_h - 1446) % 10 == 0 and thn_h != 1496:
12703 |             output_lines.append("-" * 148)
12704 |
12705 |     except Exception as err_baris:
12706 |         output_lines.append(f"{thn_h} H | Error komputasi baris: {str(err_baris)}")
12707 |
12708 |         output_lines.append("=" * 148)
12709 |         output_lines.append("* Catatan Pembacaan Tabel:")
12710 |         output_lines.append("1. [Waktu Referensi]: Waktu Sunset (Maghrib) di Sabang pada tanggal 29 Zulkaidah (H-1 sebelum 1 Zulhijah versi KHGT).")
12711 |         output_lines.append("2. [KHGT Geosentrik]: Mengukur Elongasi dan Tinggi Hilal dari pusat Bumi (Geocentric) pada saat Waktu Referensi tersebut.")
12712 |         output_lines.append("3. [MABIMS Toposentrik]: Mengukur Elongasi dan Tinggi Hilal dari permukaan Bumi (Topocentric) di Sabang pada Waktu Referensi tersebut.")
12713 |         output_lines.append("4. Jika Status 'Beda', artinya Hilal MABIMS di Sabang belum mencapai kriteria (Alt 3°, Eln 6.4°) pada waktu tersebut sehingga Zulkaidah harus diistimikan.")
12714 |
12715 |         final_text = "\n".join(output_lines)
12716 |         self.after(0, lambda: _set_ui_done(final_text))
12717 |
12718 |     except Exception as e:
12719 |         import traceback
12720 |         self.after(0, lambda: _set_ui_error(f"{str(e)}\n\n{traceback.format_exc()}"))
12721 |
12722 | # =====
12723 | # MODUL TAMBAHAN 26: KALENDER HIJRIAH BERJALAN
12724 | # =====
12725 | def reset_kalender(self, update_ui=True):
12726 |     today = datetime.date.today()
12727 |     found_y, found_m = 1447, 0 # Default aman
12728 |
12729 |     # Deteksi otomatis bulan hijriah saat ini
12730 |     for y, months in HIJRI_DB.items():
12731 |         for m_idx, m_data in enumerate(months):
12732 |             start = HijriConverter.parse_date(m_data[2])
12733 |             if start and start <= today < start + datetime.timedelta(days=m_data[3]):
12734 |                 found_y, found_m = y, m_idx
12735 |                 break
12736 |     self.cal_h_year = found_y
12737 |     self.cal_h_month = found_m
12738 |
12739 |     if update_ui:
12740 |         self.render_kalender()
12741 |
12742 | def setup_kalender_out_frame(self):
12743 |     # 1. UBAH MENJADI SCROLLABLE FRAME AGAR BISA DI-SCROLL KE BAWAH
12744 |     self.frame_kalender_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="transparent")
12745 |
12746 |     # --- Header Navigasi & Cetak ---
12747 |     header_cal = ctk.CTkFrame(self.frame_kalender_out, fg_color="#101018", corner_radius=8)
12748 |     header_cal.pack(fill="x", padx=15, pady=(0, 10))
12749 |
12750 |     btn_prev = ctk.CTkButton(header_cal, text="◀ Sebelumnya", command=self.cal_prev_month, fg_color="#1F1F1F", hover_color="#333333", width=120)
12751 |     btn_prev.pack(side="left", padx=15, pady=10)
12752 |
12753 |     self.lbl_cal_title = ctk.CTkLabel(header_cal, text="", font=("Segoe UI", 22, "bold"), text_color="#00E5FF", justify="center")
12754 |     self.lbl_cal_title.pack(side="left", expand=True)
12755 |
12756 |     btn_next = ctk.CTkButton(header_cal, text="Berikutnya ▶", command=self.cal_next_month, fg_color="#1F1F1F", hover_color="#333333", width=120)
12757 |     btn_next.pack(side="right", padx=15, pady=10)
12758 |
12759 |     frame_export = ctk.CTkFrame(self.frame_kalender_out, fg_color="transparent")
12760 |     frame_export.pack(fill="x", padx=15, pady=(0, 10))

```

```

12761 |         btn_pdf = ctk.CTkButton(frame_export, text=" Cetak PDF", font=("Segoe UI", 12, "bold"), command=self.export_to_pdf, fg_color="#D32F2F", hover_color="#B71C1C")
12762 |         btn_pdf.pack(side="right", padx=5)
12763 |
12764 |
12765 |         btn_png = ctk.CTkButton(frame_export, text=" Simpan PNG", font=("Segoe UI", 12, "bold"), command=self.export_to_png, fg_color="#F57C00", hover_color="#E65100")
12766 |         btn_png.pack(side="right", padx=5)
12767 |
12768 |         # --- Area Grid Kalender ---
12769 |         self.grid_cal_frame = ctk.CTkFrame(self.frame_kalender_out, fg_color="#050510", corner_radius=10)
12770 |         self.grid_cal_frame.pack(fill="both", expand=True, padx=15, pady=(0, 15))
12771 |
12772 |         # --- 2. TAMBAHAN AREA CATATAN HARI PENTING ISLAM ---
12773 |         self.frame_kalender_events = ctk.CTkFrame(self.frame_kalender_out, fg_color="#1E1E1E", corner_radius=8, border_width=1, border_color="#333333")
12774 |         self.frame_kalender_events.pack(fill="x", padx=15, pady=(0, 15))
12775 |
12776 |         self.lbl_kalender_events_title = ctk.CTkLabel(self.frame_kalender_events, text=" Catatan & Hari Penting Islam:", font=("Segoe UI", 16, "bold"), text_color="#FFD54F", anchor="w")
12777 |         self.lbl_kalender_events_title.pack(fill="x", padx=15, pady=(10, 5))
12778 |
12779 |         self.lbl_kalender_events_text = ctk.CTkLabel(self.frame_kalender_events, text="", font=("Segoe UI", 14), text_color="#E0E0E0", justify="left", anchor="w")
12780 |         self.lbl_kalender_events_text.pack(fill="x", padx=15, pady=(0, 15))
12781 |
12782 |     def cal_prev_month(self):
12783 |         self.cal_h_month -= 1
12784 |         if self.cal_h_month < 0:
12785 |             self.cal_h_month = 11
12786 |             self.cal_h_year -= 1
12787 |             if self.cal_h_year not in HIJRI_DB:
12788 |                 self.cal_h_year += 1 # Kembalikan batas
12789 |                 self.cal_h_month = 0
12790 |                 messagebox.showinfo("Batas Data", "Telah mencapai batas awal database.")
12791 |             return
12792 |         self.render_kalender()
12793 |
12794 |     def cal_next_month(self):
12795 |         self.cal_h_month += 1
12796 |         if self.cal_h_month > 11:
12797 |             self.cal_h_month = 0
12798 |             self.cal_h_year += 1
12799 |             if self.cal_h_year not in HIJRI_DB:
12800 |                 self.cal_h_year -= 1
12801 |                 self.cal_h_month = 11
12802 |                 messagebox.showinfo("Batas Data", "Telah mencapai batas akhir database.")
12803 |             return
12804 |         self.render_kalender()
12805 |
12806 |     def render_kalender(self):
12807 |         # Bersihkan grid lama
12808 |         for widget in self.grid_cal_frame.winfo_children():
12809 |             widget.destroy()
12810 |
12811 |         y = self.cal_h_year
12812 |         m = self.cal_h_month
12813 |
12814 |         if y not in HIJRI_DB: return
12815 |
12816 |         m_data = HIJRI_DB[y][m]
12817 |         nama_bulan, _, start_date_str, jumlah_hari = m_data
12818 |
12819 |         # Konversi string Masehi (e.g. 26-Jun-2025) ke datetime
12820 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
12821 |         parts = start_date_str.split('-')
12822 |         start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
12823 |         end_date = start_date + datetime.timedelta(days=jumlah_hari-1)
12824 |
12825 |         # Update Judul Header
12826 |         hijri_title = f"{nama_bulan} {y} H"
12827 |         greg_title = f"{BULAN_MASEHI[start_date.month-1]} {start_date.year}"
12828 |         if start_date.month != end_date.month:
12829 |             greg_title += f" - {BULAN_MASEHI[end_date.month-1]} {end_date.year}"
12830 |
12831 |         self.lbl_cal_title.configure(text=f"{hijri_title}\n({greg_title})")

```

```

12832 |
12833 |     # Kolom Header Hari
12834 |     days_header = ["Ahad", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]
12835 |     for col, day_name in enumerate(days_header):
12836 |         lbl = ctk.CTkLabel(self.grid_cal_frame, text=day_name, height=35, font=("Segoe UI", 14, "bold"), fg_color="#1E88E5", corner_radius=5)
12837 |         lbl.grid(row=0, column=col, padx=3, pady=5, sticky="nsew")
12838 |
12839 |     # Cari titik awal kolom (Hari Masehi Senin=0, Ahad=6. Kita ubah Ahad=0, Senin=1)
12840 |     offset = (start_date.weekday() + 1) % 7
12841 |
12842 |     row, col = 1, offset
12843 |     current_g_date = start_date
12844 |
12845 |     # Konstruksi Grid Hari
12846 |     for h_day in range(1, jumlah_hari + 1):
12847 |
12848 |         # ---> Deteksi hari ini untuk border kuning tebal <---
12849 |         if current_g_date == datetime.date.today():
12850 |             border_warna = "#FFEA00" # Kuning Terang
12851 |             border_tebal = 3          # Lebih tebal
12852 |         else:
12853 |             border_warna = "#424242" # Warna default (abu-abu gelap)
12854 |             border_tebal = 1          # Ketebalan standar
12855 |
12856 |         # Frame Per Kotak (Terapkan variabel warna & ketebalan di sini)
12857 |         cell = ctk.CTkFrame(self.grid_cal_frame, fg_color="#1E1E1E", corner_radius=8, border_width=border_tebal, border_color=border_warna)
12858 |         cell.grid(row=row, column=col, padx=4, pady=4, sticky="nsew")
12859 |
12860 |         # Angka Hijriah (Besar di Tengah)
12861 |         lbl_h = ctk.CTkLabel(cell, text=str(h_day), font=("Consolas", 32, "bold"), text_color="#FFD54F")
12862 |         lbl_h.pack(expand=True)
12863 |
12864 |         # Angka Masehi (Kecil di Bawah Kanan)
12865 |         m_str = f"{current_g_date.day} {BULAN_MASEHI[current_g_date.month-1][:3]} {current_g_date.year}"
12866 |
12867 |         # Tandai Jumat warna sedikit beda
12868 |         color_masehi = "#00E676" if col == 5 else "#B0BEC5"
12869 |         lbl_g = ctk.CTkLabel(cell, text=m_str, font=("Segoe UI", 11), text_color=color_masehi)
12870 |         lbl_g.pack(side="bottom", anchor="e", padx=8, pady=3)
12871 |
12872 |         # Update Posisi
12873 |         col += 1
12874 |         if col > 6:
12875 |             col = 0
12876 |             row += 1
12877 |         current_g_date += datetime.timedelta(days=1)
12878 |
12879 |     # Sinkronisasi nilai UI di sidebar
12880 |     try:
12881 |         self.combo_kal_h_bulan.set(BULAN_HIJRIAH[self.cal_h_month])
12882 |         self.entry_kal_h_tahun.delete(0, 'end')
12883 |         self.entry_kal_h_tahun.insert(0, str(self.cal_h_year))
12884 |     except Exception:
12885 |         pass
12886 |
12887 |     # Proporsi Seragam Grid Matriks
12888 |     for i in range(7):
12889 |         self.grid_cal_frame.grid_columnconfigure(i, weight=1)
12890 |
12891 |     # KUNCI tinggi baris header (row 0) agar tidak menyusut
12892 |     self.grid_cal_frame.grid_rowconfigure(0, weight=0, minsize=45)
12893 |     for i in range(1, 7): # Sisa baris tanggal tetap fleksibel
12894 |         self.grid_cal_frame.grid_rowconfigure(i, weight=1)
12895 |
12896 |     # =====
12897 |     # INJEKSI LOGIKA HARI PENTING & SEJARAH ISLAM DUNIA
12898 |     # =====
12899 |     h_month_num = m + 1
12900 |     events_list = []
12901 |
12902 |     if h_month_num == 1:

```

```

12903 |         events_list.extend([
12904 |             "1 Muharam: Tahun Baru Hijriah (Penetapan Penanggalan Islam oleh Umar bin Khattab, 17 H)",
12905 |             "9 Muharam: Puasa Sunnah Tasu'a",
12906 |             "10 Muharam: Puasa Sunnah Asyura (Keselamatan Musa AS & Tragedi Karbala, 61 H)",
12907 |             "13, 14, 15 Muharam: Puasa Ayyamul Bidh"
12908 |         ])
12909 |     elif h_month_num == 2:
12910 |         events_list.extend([
12911 |             "13, 14, 15 Safar: Puasa Ayyamul Bidh",
12912 |             "27 Safar: Momen awal Rasulullah SAW hijrah meninggalkan Makkah menuju Madinah"
12913 |         ])
12914 |     elif h_month_num == 3:
12915 |         events_list.extend([
12916 |             "8 Rabiulawal: Kedatangan Nabi SAW di Quba & pendirian Masjid Quba (1 H)",
12917 |             "12 Rabiulawal: Maulid Nabi Muhammad SAW (Tahun Gajah) & Wafatnya Rasulullah SAW (11 H)",
12918 |             "13, 14, 15 Rabiulawal: Puasa Ayyamul Bidh"
12919 |         ])
12920 |     elif h_month_num == 4:
12921 |         events_list.extend([
12922 |             "13, 14, 15 Rabiulakhir: Puasa Ayyamul Bidh"
12923 |         ])
12924 |     elif h_month_num == 5:
12925 |         events_list.extend([
12926 |             "8 Jumadilawal: Terjadinya Perang Mu'tah melawan Bizantium (8 H)",
12927 |             "13, 14, 15 Jumadilawal: Puasa Ayyamul Bidh"
12928 |         ])
12929 |     elif h_month_num == 6:
12930 |         events_list.extend([
12931 |             "13, 14, 15 Jumadilakhir: Puasa Ayyamul Bidh",
12932 |             "22 Jumadilakhir: Wafatnya Khalifah Pertama, Abu Bakar As-Siddiq (13 H)"
12933 |         ])
12934 |     elif h_month_num == 7:
12935 |         events_list.extend([
12936 |             "13, 14, 15 Rajab: Puasa Ayyamul Bidh",
12937 |             "27 Rajab: Peristiwa Agung Isra' Mi'raj Nabi Muhammad SAW",
12938 |             "28 Rajab: Pembebasan Baitul Maqdis (Yerusalem) oleh Salahuddin Al-Ayyubi (583 H)"
12939 |         ])
12940 |     elif h_month_num == 8:
12941 |         events_list.extend([
12942 |             "13, 14, 15 Syakban: Puasa Ayyamul Bidh",
12943 |             "15 Syakban: Turunnya perintah peralihan Kiblat dari Masjidil Aqsa ke Ka'bah (2 H)"
12944 |         ])
12945 |     elif h_month_num == 9:
12946 |         events_list.extend([
12947 |             "1-30 Ramadan: Puasa Wajib Bulan Ramadan",
12948 |             "17 Ramadan: Nuzulul Quran & Kemenangan besar Perang Badar (2 H)",
12949 |             "20 Ramadan: Fathu Makkah / Penaklukan & Pembebasan Makkah (8 H)",
12950 |             "10 Hari Terakhir: I'tikaf & Pencarian Malam Lailatul Qadar"
12951 |         ])
12952 |     elif h_month_num == 10:
12953 |         events_list.extend([
12954 |             "1 Syawal: Hari Raya Idul Fitri (Haram Berpuasa)",
12955 |             "13, 14, 15 Syawal: Puasa Ayyamul Bidh",
12956 |             "15 Syawal: Terjadinya Perang Uhud (3 H)",
12957 |             "6 Hari di Bulan Syawal: Puasa Sunnah Syawal"
12958 |         ])
12959 |     elif h_month_num == 11:
12960 |         events_list.extend([
12961 |             "5 Zulkaidah: Disepakatinya Perjanjian Hudaibiyah (6 H)",
12962 |             "13, 14, 15 Zulkaidah: Puasa Ayyamul Bidh"
12963 |         ])
12964 |     elif h_month_num == 12:
12965 |         events_list.extend([
12966 |             "1-9 Zulhijah: Keutamaan Amal & Puasa Sunnah Dzulhijjah",
12967 |             "8 Zulhijah: Puasa Tarwiyah",
12968 |             "9 Zulhijah: Puasa Arafah & Penyampaian Khutbah Wada' / Perpisahan Nabi SAW (10 H)",
12969 |             "10 Zulhijah: Hari Raya Idul Adha / Kurban (Haram Berpuasa)",
12970 |             "11, 12, 13 Zulhijah: Hari Tasyriq (Haram Berpuasa)",
12971 |             "14, 15 Zulhijah: Puasa Ayyamul Bidh",
12972 |             "18 Zulhijah: Wafatnya Khalifah Ketiga, Utsman bin Affan (35 H)"
12973 |         ])

```

```

12974 |
12975 |     if events_list:
12976 |         # Menggunakan spasi antar baris yang lega agar nyaman dibaca saat di scroll
12977 |         events_text = "\n\n".join(f" {ev}" for ev in events_list)
12978 |     else:
12979 |         events_text = "Tidak ada hari besar spesifik pada bulan ini selain puasa sunnah."
12980 |
12981 |     # Perbarui teks label dengan fitur wraplength agar teks tidak terpotong ke samping
12982 |     self.lbl_kalender_events_text.configure(text=events_text, wraplength=950)
12983 |
12984 | def _generate_calendar_image(self):
12985 |     """Merender kanvas elegan murni lewat Pillow untuk resolusi Ultra HD"""
12986 |     img = Image.new("RGB", (1400, 1000), "#0A0A10")
12987 |     draw = ImageDraw.Draw(img)
12988 |
12989 |     try:
12990 |         f_title = ImageFont.truetype("arialbd.ttf", 46)
12991 |         f_sub = ImageFont.truetype("arial.ttf", 26)
12992 |         f_day = ImageFont.truetype("arialbd.ttf", 22)
12993 |         f_h = ImageFont.truetype("arialbd.ttf", 64)
12994 |         f_m = ImageFont.truetype("arial.ttf", 16)
12995 |     except Exception:
12996 |         f_title = f_sub = f_day = f_h = f_m = ImageFont.load_default()
12997 |
12998 |     y = self.cal_h_year
12999 |     m = self.cal_h_month
13000 |     m_data = HIJRI_DB[y][m]
13001 |     nama_bulan, _, start_date_str, jumlah_hari = m_data
13002 |
13003 |     bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
13004 |     parts = start_date_str.split('.')
13005 |     start_date = datetime.date(int(parts[2]), bulan_map.get(parts[1], 1), int(parts[0]))
13006 |     end_date = start_date + datetime.timedelta(days=jumlah_hari-1)
13007 |
13008 |     hijri_title = f"Kalender Hijriah KHGT: {nama_bulan} {y} H"
13009 |     greg_title = f"{BULAN_MASEHI[start_date.month-1]} {start_date.year}"
13010 |     if start_date.month != end_date.month:
13011 |         greg_title += f" - {BULAN_MASEHI[end_date.month-1]} {end_date.year}"
13012 |
13013 |     draw.text((700, 60), hijri_title, font=f_title, fill="#00E5FF", anchor="mm")
13014 |     draw.text((700, 110), greg_title, font=f_sub, fill="#B0BEC5", anchor="mm")
13015 |
13016 |     start_x, start_y = 50, 180
13017 |     cell_w, cell_h = 185, 120
13018 |     days_header = ["Ahad", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]
13019 |
13020 |     for i, d_name in enumerate(days_header):
13021 |         x0 = start_x + i * cell_w
13022 |         y0 = start_y
13023 |         x1 = x0 + cell_w - 10
13024 |         y1 = y0 + 45
13025 |         draw.rectangle([x0, y0, x1, y1], fill="#1E88E5", outline="#1565C0", width=2)
13026 |         draw.text(((x0+x1)/2, (y0+y1)/2), d_name, font=f_day, fill="white", anchor="mm")
13027 |
13028 |     offset = (start_date.weekday() + 1) % 7
13029 |     row, col = 0, offset
13030 |     curr_g = start_date
13031 |     grid_y_start = start_y + 60
13032 |
13033 |     for h_day in range(1, jumlah_hari + 1):
13034 |         x0 = start_x + col * cell_w
13035 |         y0 = grid_y_start + row * cell_h
13036 |         x1 = x0 + cell_w - 10
13037 |         y1 = y0 + cell_h - 10
13038 |
13039 |         draw.rectangle([x0, y0, x1, y1], fill="#1E1E1E", outline="#424242", width=2)
13040 |
13041 |         # --- CEK EVENT EXPORT HIJRIAH ---
13042 |         event_color, _ = self.get_islamic_event(h_day, m + 1, curr_g.weekday())
13043 |         color_hijri = event_color if event_color else "#FFD54F"
13044 |

```

```

13045 |         if event_color:
13046 |             draw.ellipse([x0+15, y0+15, x0+25, y0+25], fill=color_hijri) # Titik penanda warna puasa/event
13047 |
13048 |         draw.text(((x0+x1)/2, y0 + 40), str(h_day), font=f_h, fill=color_hijri, anchor="mm")
13049 |
13050 |         # Angka Masehi
13051 |         m_str = f"{curr_g.day} {BULAN_MASEHI[curr_g.month-1][:3]} {curr_g.year}"
13052 |         draw.text((x1 - 10, y1 - 15), m_str, font=f_m, fill="#B0BEC5", anchor="rm")
13053 |
13054 |         col += 1
13055 |         if col > 6:
13056 |             col = 0
13057 |             row += 1
13058 |         curr_g += datetime.timedelta(days=1)
13059 |
13060 |         draw.text((700, 960), "KHGT Times Engine V8.0 - By Kasmui", font=f_sub, fill="#555555", anchor="mm")
13061 |         return img
13062 |
13063 |
13064 |     # =====
13065 |     # MODUL TAMBAHAN 27: KALENDER MASEHI BERJALAN
13066 |     # =====
13067 |     def reset_kalmasehi(self, update_ui=True):
13068 |         today = datetime.date.today()
13069 |         self.cal_m_year = today.year
13070 |         self.cal_m_month = today.month
13071 |         if update_ui:
13072 |             self.render_kalmasehi()
13073 |
13074 |     def setup_kalmasehi_out_frame(self):
13075 |         # 1. UBAH MENJADI SCROLLABLE FRAME AGAR BISA DI-SCROLL KE BAWAH
13076 |         self.frame_kalmasehi_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="transparent")
13077 |
13078 |         # --- Header Navigasi & Cetak ---
13079 |         header_cal = ctk.CTkFrame(self.frame_kalmasehi_out, fg_color="#101018", corner_radius=8)
13080 |         header_cal.pack(fill="x", padx=15, pady=(0, 10))
13081 |
13082 |         btn_prev = ctk.CTkButton(header_cal, text="◀ Sebelumnya", command=self.cal_masehi_prev_month, fg_color="#1F1F1F", hover_color="#333333", width=120)
13083 |         btn_prev.pack(side="left", padx=15, pady=10)
13084 |
13085 |         self.lbl_cal_m_title = ctk.CTkLabel(header_cal, text="", font=("Segoe UI", 22, "bold"), text_color="#00E5FF", justify="center")
13086 |         self.lbl_cal_m_title.pack(side="left", expand=True)
13087 |
13088 |         btn_next = ctk.CTkButton(header_cal, text="Berikutnya ▶", command=self.cal_masehi_next_month, fg_color="#1F1F1F", hover_color="#333333", width=120)
13089 |         btn_next.pack(side="right", padx=15, pady=10)
13090 |
13091 |         frame_export = ctk.CTkFrame(self.frame_kalmasehi_out, fg_color="transparent")
13092 |         frame_export.pack(fill="x", padx=15, pady=(0, 10))
13093 |
13094 |         btn_pdf = ctk.CTkButton(frame_export, text="📄 Cetak PDF", font=("Segoe UI", 12, "bold"), command=self.export_to_pdf, fg_color="#D32F2F", hover_color="#B71C1C")
13095 |         btn_pdf.pack(side="right", padx=5)
13096 |
13097 |         btn_png = ctk.CTkButton(frame_export, text="💾 Simpan PNG", font=("Segoe UI", 12, "bold"), command=self.export_to_png, fg_color="#F57C00", hover_color="#E65100")
13098 |         btn_png.pack(side="right", padx=5)
13099 |
13100 |         # --- Area Grid Kalender ---
13101 |         self.grid_cal_m_frame = ctk.CTkFrame(self.frame_kalmasehi_out, fg_color="#050510", corner_radius=10)
13102 |         self.grid_cal_m_frame.pack(fill="both", expand=True, padx=15, pady=(0, 15))
13103 |
13104 |         # --- 2. TAMBAHAN AREA CATATAN HARI PENTING MASEHI ---
13105 |         self.frame_kalmasehi_events = ctk.CTkFrame(self.frame_kalmasehi_out, fg_color="#1E1E1E", corner_radius=8, border_width=1, border_color="#333333")
13106 |         self.frame_kalmasehi_events.pack(fill="x", padx=15, pady=(0, 15))
13107 |
13108 |         self.lbl_kalmasehi_events_title = ctk.CTkLabel(self.frame_kalmasehi_events, text="📅 Catatan & Hari Penting (Nasional/Internasional):", font=("Segoe UI", 16, "bold"), text_color="#00E5FF",
13109 | anchor="w")
13110 |         self.lbl_kalmasehi_events_title.pack(fill="x", padx=15, pady=(10, 5))
13111 |
13112 |         self.lbl_kalmasehi_events_text = ctk.CTkLabel(self.frame_kalmasehi_events, text="", font=("Segoe UI", 14), text_color="#E0E0E0", justify="left", anchor="w")
13113 |         self.lbl_kalmasehi_events_text.pack(fill="x", padx=15, pady=(0, 15))
13114 |
13115 |     def cal_masehi_prev_month(self):

```

```

13115 |         self.cal_m_month -= 1
13116 |         if self.cal_m_month < 1:
13117 |             self.cal_m_month = 12
13118 |             self.cal_m_year -= 1
13119 |         self.render_kalmasehi()
13120 |
13121 |     def cal_masehi_next_month(self):
13122 |         self.cal_m_month += 1
13123 |         if self.cal_m_month > 12:
13124 |             self.cal_m_month = 1
13125 |             self.cal_m_year += 1
13126 |         self.render_kalmasehi()
13127 |
13128 |     def render_kalmasehi(self):
13129 |         # Bersihkan grid lama
13130 |         for widget in self.grid_cal_m_frame.winfo_children():
13131 |             widget.destroy()
13132 |
13133 |         y = self.cal_m_year
13134 |         m = self.cal_m_month
13135 |
13136 |         _, jumlah_hari = calendar.monthrange(y, m)
13137 |         start_date = datetime.date(y, m, 1)
13138 |         end_date = datetime.date(y, m, jumlah_hari)
13139 |
13140 |         # Cari rentang bulan Hijriah untuk judul menggunakan data KHGT
13141 |         h_start = HijriConverter.get_hijri_date(start_date)
13142 |         h_end = HijriConverter.get_hijri_date(end_date)
13143 |
13144 |         greg_title = f"{BULAN_MASEHI[m-1]} {y}"
13145 |
13146 |         # Ekstrak bulan dan tahun Hijriah (misal: "1 Muharam 1446 H" -> "Muharam 1446")
13147 |         try:
13148 |             h1_parts = h_start.split(" ")
13149 |             h2_parts = h_end.split(" ")
13150 |             h1_m_y = f"{h1_parts[1]} {h1_parts[2]}"
13151 |             h2_m_y = f"{h2_parts[1]} {h2_parts[2]}"
13152 |             if h1_m_y != h2_m_y:
13153 |                 hijri_title = f"{h1_m_y} - {h2_m_y} H"
13154 |             else:
13155 |                 hijri_title = f"{h1_m_y} H"
13156 |         except:
13157 |             hijri_title = "Data Hijriah KHGT"
13158 |
13159 |         self.lbl_cal_m_title.configure(text=f"{greg_title}\n({hijri_title})")
13160 |
13161 |         # Header Hari Masehi (Senin di awal, Ahad di akhir)
13162 |         days_header = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
13163 |         for col, day_name in enumerate(days_header):
13164 |             color_bg = "#D84315" if col == 6 else "#E65100" # Ahad warnanya lebih gelap
13165 |             lbl = ctk.CTkLabel(self.grid_cal_m_frame, text=day_name, height=35, font=("Segoe UI", 14, "bold"), fg_color=color_bg, corner_radius=5)
13166 |             lbl.grid(row=0, column=col, padx=3, pady=5, sticky="nsew")
13167 |
13168 |         # Offset kalender Masehi (Monday=0, Sunday=6)
13169 |         offset = start_date.weekday()
13170 |         row, col = 1, offset
13171 |
13172 |         curr_g = start_date
13173 |         for m_day in range(1, jumlah_hari + 1):
13174 |
13175 |             # ---> Deteksi hari ini untuk border kuning tebal <---
13176 |             if curr_g == datetime.date.today():
13177 |                 border_warna = "#FFEA00" # Kuning Terang
13178 |                 border_tebal = 3 # Lebih tebal
13179 |             else:
13180 |                 border_warna = "#424242" # Warna default (abu-abu gelap)
13181 |                 border_tebal = 1 # Ketebalan standar
13182 |
13183 |             # Frame Per Kotak
13184 |             cell = ctk.CTkFrame(self.grid_cal_m_frame, fg_color="#1E1E1E", corner_radius=8, border_width=border_tebal, border_color=border_warna)
13185 |             cell.grid(row=row, column=col, padx=4, pady=4, sticky="nsew")

```

```

13186 |
13187 |         # Angka Masehi Besar (Warna merah khusus untuk Ahad)
13188 |         color_masehi = "#FF5252" if col == 6 else "#00E5FF"
13189 |         lbl_m = ctk.CTkLabel(cell, text=str(m_day), font=("Consolas", 32, "bold"), text_color=color_masehi)
13190 |         lbl_m.pack(expand=True)
13191 |
13192 |         # Angka Hijriah Kecil di bawah
13193 |         h_str = HijriConverter.get_hijri_date(curr_g)
13194 |         if h_str == "N/A": h_str = "."
13195 |
13196 |         # Singkat nama bulan agar muat
13197 |         parts = h_str.split()
13198 |         if len(parts) >= 4:
13199 |             h_str_short = f"{parts[0]} {parts[1][:3]} {parts[2]} H"
13200 |         else:
13201 |             h_str_short = h_str
13202 |
13203 |         lbl_h = ctk.CTkLabel(cell, text=h_str_short, font=("Segoe UI", 11), text_color="#FFD54F")
13204 |         lbl_h.pack(side="bottom", anchor="e", padx=8, pady=3)
13205 |
13206 |         col += 1
13207 |         if col > 6:
13208 |             col = 0
13209 |             row += 1
13210 |         curr_g += datetime.timedelta(days=1)
13211 |
13212 |     # Sinkronisasi nilai UI di sidebar
13213 |     try:
13214 |         self.combo_kal_m_bulan.set(BULAN_MASEHI[self.cal_m_month - 1])
13215 |         self.entry_kal_m_tahun.delete(0, 'end')
13216 |         self.entry_kal_m_tahun.insert(0, str(self.cal_m_year))
13217 |     except Exception:
13218 |         pass
13219 |
13220 |     # Proporsi Seragam
13221 |     for i in range(7):
13222 |         self.grid_calm_frame.grid_columnconfigure(i, weight=1)
13223 |
13224 |     # KUNCI tinggi baris header Masehi
13225 |     self.grid_calm_frame.grid_rowconfigure(0, weight=0, minsize=45)
13226 |     for i in range(1, 7):
13227 |         self.grid_calm_frame.grid_rowconfigure(i, weight=1)
13228 |
13229 |     # =====
13230 |     # INJEKSI LOGIKA HARI PENTING NASIONAL & INTERNASIONAL
13231 |     # =====
13232 |     m_events_list = []
13233 |     tahun_str = str(y) if y > 0 else f"{abs(y-1)} SM"
13234 |
13235 |     if m == 1:
13236 |         m_events_list.extend([
13237 |             f"1 Januari {tahun_str}: Tahun Baru Masehi",
13238 |             f"15 Januari {tahun_str}: Hari Peringatan Laut dan Samudera Nasional",
13239 |             f"25 Januari {tahun_str}: Hari Gizi Nasional Indonesia"
13240 |         ])
13241 |     elif m == 2:
13242 |         m_events_list.extend([
13243 |             f"4 Februari {tahun_str}: Hari Kanker Sedunia",
13244 |             f"9 Februari {tahun_str}: Hari Pers Nasional (HPN)",
13245 |             f"21 Februari {tahun_str}: Hari Peduli Sampah Nasional"
13246 |         ])
13247 |     elif m == 3:
13248 |         m_events_list.extend([
13249 |             f"1 Maret {tahun_str}: Peringatan Serangan Umum 1 Maret (Bersejarah 1949)",
13250 |             f"8 Maret {tahun_str}: Hari Perempuan Internasional (International Women's Day)",
13251 |             f"9 Maret {tahun_str}: Hari Musik Nasional (Kelahiran W.R. Supratman)",
13252 |             f"22 Maret {tahun_str}: Hari Air Sedunia",
13253 |             f"30 Maret {tahun_str}: Hari Film Nasional"
13254 |         ])
13255 |     elif m == 4:
13256 |         m_events_list.extend([

```



```

13257 |         f"7 April {tahun_str}: Hari Kesehatan Sedunia",
13258 |         f"18 April {tahun_str}: Hari Peringatan Konferensi Asia-Afrika (KAA Bandung 1955)",
13259 |         f"21 April {tahun_str}: Hari Kartini (Pahlawan Emansipasi Wanita, Lahir 1879)",
13260 |         f"22 April {tahun_str}: Hari Bumi (Earth Day)",
13261 |         f"23 April (Hari Buku Sedunia (World Book Day))"
13262 |     ])
13263 |     elif m == 5:
13264 |         m_events_list.extend([
13265 |             f"1 Mei {tahun_str}: Hari Buruh Internasional (May Day)",
13266 |             f"2 Mei {tahun_str}: Hari Pendidikan Nasional (Kelahiran Ki Hajar Dewantara, 1889)",
13267 |             f"20 Mei {tahun_str}: Hari Kebangkitan Nasional (Berdirinya Budi Utomo, 1908)",
13268 |             f"17 Mei (Hari Buku Nasional)",
13269 |             f"29 Mei (Hari Penaklukan Konstantinopel oleh Sultan Mehmed (Muhammad Al-Fatih) tahun 1453 M)"
13270 |         ])
13271 |     elif m == 6:
13272 |         m_events_list.extend([
13273 |             f"1 Juni {tahun_str}: Hari Lahir Pancasila (Pidato Bung Karno, 1945)",
13274 |             f"5 Juni {tahun_str}: Hari Lingkungan Hidup Sedunia",
13275 |             f"22 Juni {tahun_str}: Hari Ulang Tahun Kota Jakarta"
13276 |         ])
13277 |     elif m == 7:
13278 |         m_events_list.extend([
13279 |             f"12 Juli {tahun_str}: Hari Koperasi Indonesia",
13280 |             f"23 Juli {tahun_str}: Hari Anak Nasional"
13281 |         ])
13282 |     elif m == 8:
13283 |         m_events_list.extend([
13284 |             f"8 Agustus {tahun_str}: Hari Berdirinya ASEAN (1967)",
13285 |             f"10 Agustus {tahun_str}: Hari Veteran Nasional",
13286 |             f"14 Agustus {tahun_str}: Hari Pramuka (Praja Muda Karana)",
13287 |             f"17 Agustus {tahun_str}: Hari Proklamasi Kemerdekaan Republik Indonesia (1945)",
13288 |             f"18 Agustus {tahun_str}: Hari Konstitusi Republik Indonesia"
13289 |         ])
13290 |     elif m == 9:
13291 |         m_events_list.extend([
13292 |             f"9 September {tahun_str}: Hari Olahraga Nasional (Haornas)",
13293 |             f"17 September {tahun_str}: Hari Palang Merah Indonesia (PMI)",
13294 |             f"30 September {tahun_str}: Hari Peringatan G30S/PKI (1965)"
13295 |         ])
13296 |     elif m == 10:
13297 |         m_events_list.extend([
13298 |             f"1 Oktober {tahun_str}: Hari Kesaktian Pancasila",
13299 |             f"2 Oktober {tahun_str}: Hari Batik Nasional",
13300 |             f"5 Oktober {tahun_str}: Hari Ulang Tahun Tentara Nasional Indonesia (TNI, Berdiri 1945)",
13301 |             f"28 Oktober {tahun_str}: Hari Sumpah Pemuda (Kongres Pemuda II, 1928)"
13302 |         ])
13303 |     elif m == 11:
13304 |         m_events_list.extend([
13305 |             f"10 November {tahun_str}: Hari Pahlawan Nasional (Pertempuran Surabaya, 1945)",
13306 |             f"12 November {tahun_str}: Hari Ayah Nasional",
13307 |             f"25 November {tahun_str}: Hari Guru Nasional (Berdirinya PGRI, 1945)"
13308 |         ])
13309 |     elif m == 12:
13310 |         m_events_list.extend([
13311 |             f"1 Desember {tahun_str}: Hari AIDS Sedunia",
13312 |             f"22 Desember {tahun_str}: Hari Ibu Nasional (Kongres Perempuan Indonesia, 1928)",
13313 |             f"25 Desember {tahun_str}: Hari Raya Natal"
13314 |         ])
13315 |
13316 |     if m_events_list:
13317 |         # Gunakan spasi antar baris ganda agar lebih leluasa saat dibaca
13318 |         events_text = "\n\n".join(f"{} {ev}" for ev in m_events_list)
13319 |     else:
13320 |         events_text = "[] Tidak ada catatan hari besar nasional yang spesifik pada bulan ini."
13321 |
13322 |     # Terapkan teks dengan fitur wraplength agar rapi ke bawah
13323 |     self.lbl_kalmasehi_events_text.configure(text=events_text, wraplength=950)
13324 |
13325 | def _generate_calmasehi_image(self):
13326 |     """Merender kanvas kalender masehi lewat Pillow untuk ekspor PDF & PNG HD"""
13327 |     img = Image.new("RGB", (1400, 1000), "#0A0A10")

```

```

13328 |         draw = ImageDraw.Draw(img)
13329 |
13330 |         try:
13331 |             f_title = ImageFont.truetype("arialbd.ttf", 46)
13332 |             f_sub = ImageFont.truetype("arial.ttf", 26)
13333 |             f_day = ImageFont.truetype("arialbd.ttf", 22)
13334 |             f_m = ImageFont.truetype("arialbd.ttf", 64)
13335 |             f_h = ImageFont.truetype("arial.ttf", 16)
13336 |         except Exception:
13337 |             f_title = f_sub = f_day = f_m = f_h = ImageFont.load_default()
13338 |
13339 |         y = self.cal_m_year
13340 |         m = self.cal_m_month
13341 |
13342 |         _, jumlah_hari = calendar.monthrange(y, m)
13343 |         start_date = datetime.date(y, m, 1)
13344 |         end_date = datetime.date(y, m, jumlah_hari)
13345 |
13346 |         h_start = HijriConverter.get_hijri_date(start_date)
13347 |         h_end = HijriConverter.get_hijri_date(end_date)
13348 |
13349 |         greg_title = f"Kalender Masehi: {BULAN_MASEHI[m-1]} {y}"
13350 |
13351 |         try:
13352 |             h1_parts = h_start.split(" ")
13353 |             h2_parts = h_end.split(" ")
13354 |             h1_m_y = f"{h1_parts[1]} {h1_parts[2]}"
13355 |             h2_m_y = f"{h2_parts[1]} {h2_parts[2]}"
13356 |             if h1_m_y != h2_m_y:
13357 |                 hijri_title = f"Data KHGT: {h1_m_y} - {h2_m_y} H"
13358 |             else:
13359 |                 hijri_title = f"Data KHGT: {h1_m_y} H"
13360 |         except:
13361 |             hijri_title = "Data Hijriah KHGT"
13362 |
13363 |         # Cetak Header Text
13364 |         draw.text((700, 60), greg_title, font=f_title, fill="#00E5FF", anchor="mm")
13365 |         draw.text((700, 110), hijri_title, font=f_sub, fill="#FFD54F", anchor="mm")
13366 |
13367 |         # Konfigurasi Dimensi Grid
13368 |         start_x, start_y = 50, 180
13369 |         cell_w, cell_h = 185, 120
13370 |         days_header = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
13371 |
13372 |         # Gambar Header Hari
13373 |         for i, d_name in enumerate(days_header):
13374 |             x0 = start_x + i * cell_w
13375 |             y0 = start_y
13376 |             x1 = x0 + cell_w - 10
13377 |             y1 = y0 + 45
13378 |             color_bg = "#D84315" if i == 6 else "#E65100"
13379 |             draw.rectangle([x0, y0, x1, y1], fill=color_bg, outline="#E65100", width=2)
13380 |             draw.text(((x0+x1)/2, (y0+y1)/2), d_name, font=f_day, fill="white", anchor="mm")
13381 |
13382 |         # Gambar Isi Tanggal
13383 |         offset = start_date.weekday()
13384 |         row, col = 0, offset
13385 |         curr_g = start_date
13386 |         grid_y_start = start_y + 60
13387 |
13388 |         for m_day in range(1, jumlah_hari + 1):
13389 |             x0 = start_x + col * cell_w
13390 |             y0 = grid_y_start + row * cell_h
13391 |             x1 = x0 + cell_w - 10
13392 |             y1 = y0 + cell_h - 10
13393 |
13394 |             # Background Box
13395 |             draw.rectangle([x0, y0, x1, y1], fill="#1E1E1E", outline="#424242", width=2)
13396 |
13397 |             # --- CEK EVENT EXPORT HIJRIAH ---
13398 |             event_color, _ = self._get_islamic_event(h_day, m + 1, curr_g.weekday())

```

```

13399 |         color_hijri = event_color if event_color else "#FFD54F"
13400 |
13401 |         if event_color:
13402 |             draw.ellipse([x0+15, y0+15, x0+25, y0+25], fill=color_hijri) # Titik penanda
13403 |
13404 |         draw.text(((x0+x1)/2, y0 + 40), str(h_day), font=f_h, fill=color_hijri, anchor="mm")
13405 |
13406 |         # Angka Masehi
13407 |         m_str = f"{curr_g.day} {BULAN_MASEHI[curr_g.month-1][:3]} {curr_g.year}"
13408 |
13409 |         # Masehi Besar
13410 |         color_masehi = "#FF5252" if col == 6 else "#00E5FF"
13411 |         draw.text(((x0+x1)/2, y0 + 40), str(m_day), font=f_m, fill=color_masehi, anchor="mm")
13412 |
13413 |         # Hijriah Kecil
13414 |         h_str = HijriConverter.get_hijri_date(curr_g)
13415 |         if h_str == "N/A": h_str = "-"
13416 |
13417 |         parts = h_str.split()
13418 |         if len(parts) >= 4:
13419 |             h_str_short = f"{parts[0]} {parts[1][:3]} {parts[2]} H"
13420 |         else:
13421 |             h_str_short = h_str
13422 |
13423 |         # --- CEK EVENT EXPORT MASEHI ---
13424 |         color_hijri = "#FFD54F"
13425 |         if len(parts) >= 4:
13426 |             try:
13427 |                 h_day_val = int(parts[0])
13428 |                 h_month_val = BULAN_HIJRIAH.index(parts[1]) + 1
13429 |                 ev_color, _ = self._get_islamic_event(h_day_val, h_month_val, curr_g.weekday())
13430 |                 if ev_color:
13431 |                     color_hijri = ev_color
13432 |                     draw.ellipse([x1-25, y0+15, x1-15, y0+25], fill=color_hijri) # Titik penanda
13433 |             except: pass
13434 |
13435 |         draw.text((x1 - 10, y1 - 15), h_str_short, font=f_h, fill=color_hijri, anchor="rm")
13436 |
13437 |         col += 1
13438 |         if col > 6:
13439 |             col = 0
13440 |             row += 1
13441 |             curr_g += datetime.timedelta(days=1)
13442 |
13443 |         # Footer
13444 |         draw.text((700, 960), "KHGT Times Engine V8.0 - By Kasmui", font=f_sub, fill="#555555", anchor="mm")
13445 |         return img
13446 |
13447 |     def export_kalmasehi(self, export_format):
13448 |         img = self._generate_calmasehi_image()
13449 |
13450 |         y = self.cal_m_year
13451 |         nama_bulan = BULAN_MASEHI[self.cal_m_month - 1]
13452 |         default_file = f"Kalender_Masehi_{nama_bulan}_{y}"
13453 |
13454 |         if export_format == "png":
13455 |             filepath = filedialog.asksaveasfilename(
13456 |                 initialfile=f"{default_file}.png",
13457 |                 defaultextension=".png",
13458 |                 filetypes=[("PNG Image", "*.png")]
13459 |             )
13460 |             if filepath:
13461 |                 try:
13462 |                     img.save(filepath, "PNG")
13463 |                     messagebox.showinfo("Sukses", f"Kalender berhasil diekspor menjadi Gambar PNG di:\n{filepath}")
13464 |                 except Exception as e:
13465 |                     messagebox.showerror("Error", f"Gagal menyimpan PNG: {e}")
13466 |
13467 |             elif export_format == "pdf":
13468 |                 filepath = filedialog.asksaveasfilename(
13469 |                     initialfile=f"{default_file}.pdf",

```

```

13470 |         defaulttextextension=".pdf",
13471 |         filetypeypes=[("PDF Document", "*.pdf")]
13472 |     )
13473 |     if filepath:
13474 |         try:
13475 |             # Konversi otomatis Image PIL menjadi PDF resolusi tinggi
13476 |             img.save(filepath, "PDF", resolution=150.0)
13477 |             messagebox.showinfo("Sukses", f"Kalender HD berhasil diekspor menjadi PDF di:\n{filepath}")
13478 |         except Exception as e:
13479 |             messagebox.showerror("Error", f"Gagal menyimpan PDF: {e}")
13480 |
13481 |
13482 | # =====
13483 | # MODUL 04: ALTITUDE CHART ANALYSER (FINAL & ANTI-FREEZE)
13484 | # =====
13485 | def trigger_chart(self, chart_type):
13486 |     """Memantik render grafik ke background thread berdasarkan tombol yang diklik"""
13487 |     self.lbl_status.configure(text="Memproses Data Grafik...", text_color="#00E5FF")
13488 |     import threading
13489 |     threading.Thread(target=self.calculate_chart_analyser, args=(chart_type,), daemon=True).start()
13490 |
13491 | def calculate_chart_analyser(self, chart_type="altitude"):
13492 |     try:
13493 |         import math
13494 |         import numpy as np
13495 |         from skyfield import almanac
13496 |         from skyfield.api import wgs84
13497 |
13498 |         # 1. Ambil input UI
13499 |         y = int(self.entry_chart_y.get())
13500 |         m = int(self.entry_chart_m.get())
13501 |         d = int(self.entry_chart_d.get())
13502 |
13503 |         lat = float(self.entry_vlat.get())
13504 |         lon = float(self.entry_vlon.get())
13505 |         tz = float(self.entry_vtz.get())
13506 |         elev = float(self.entry_velev.get())
13507 |
13508 |         self.after(0, lambda: self.lbl_status.configure(text="Mengumpulkan data Ephemeris 24 Jam...", text_color="#00E5FF"))
13509 |
13510 |         # 2. KALKULASI DI BACKGROUND THREAD
13511 |         self.auto_switch_ephemeris(y)
13512 |         earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
13513 |         loc = wgs84.latlon(lat, lon, elevation_m=elev)
13514 |         observer = earth + loc
13515 |
13516 |         hours = np.linspace(0, 24, 144) # Resolusi per 10 menit
13517 |
13518 |         # Wadah Data Array
13519 |         sun_alts, moon_alts, moon_geo_alts = [], [], []
13520 |         elongations, illuminations = [], []
13521 |
13522 |         for h in hours:
13523 |             t = self.ts.utc(y, m, d, h - tz)
13524 |
13525 |             # --- A. Data Toposentrik (Altitude MABIMS) ---
13526 |             sun_app = observer.at(t).observe(sun).apparent()
13527 |             moon_app = observer.at(t).observe(moon).apparent()
13528 |             sun_alts.append(sun_app.altaz(temperature_C=25, pressure_mbar=1010)[0].degrees)
13529 |             moon_alts.append(moon_app.altaz(temperature_C=25, pressure_mbar=1010)[0].degrees)
13530 |
13531 |             # --- B. Data Geosentrik (Elongasi KHGT) ---
13532 |             obs_center = earth.at(t)
13533 |             s_app_geo = obs_center.observe(sun).apparent()
13534 |             m_app_geo = obs_center.observe(moon).apparent()
13535 |             elongations.append(s_app_geo.separation_from(m_app_geo).degrees)
13536 |
13537 |             # --- C. Data Iluminasi (Fase Bulan) ---
13538 |             illum_val = almanac.fraction_illuminated(self.eph, 'moon', t)
13539 |             illuminations.append((illum_val.item() if hasattr(illum_val, 'item') else illum_val) * 100.0)
13540 |

```

```

13541 |         # --- D. Data Altitude Geosentrik (Tinggi KHGT) ---
13542 |         gmst = t.gast
13543 |         lst_deg = (gmst * 15.0) + lon
13544 |         ra_m, dec_m, _ = m_app_geo.radec(epoch=t)
13545 |
13546 |         ha_deg = lst_deg - (ra_m.hours * 15.0)
13547 |         ha_rad = math.radians(ha_deg)
13548 |         lat_rad = math.radians(lat)
13549 |         d_rad = dec_m.radians
13550 |
13551 |         sin_alt_geo = math.sin(d_rad) * math.sin(lat_rad) + math.cos(d_rad) * math.cos(lat_rad) * math.cos(ha_rad)
13552 |         moon_geo_alts.append(math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo)))))
13553 |
13554 |     # 3. KIRIM HASIL BUNDLE KE MAIN THREAD
13555 |     data_bundle = {
13556 |         'hours': hours, 'sun_alts': sun_alts, 'moon_alts': moon_alts,
13557 |         'moon_geo_alts': moon_geo_alts, 'elongations': elongations,
13558 |         'illuminations': illuminations, 'y': y, 'm': m, 'd': d, 'type': chart_type
13559 |     }
13560 |     self.after(0, self.plot_chart_dispatcher, data_bundle)
13561 |
13562 | except Exception as e:
13563 |     import traceback
13564 |     self.after(0, self._force_show_error, f"{str(e)}\n\n{traceback.format_exc()}")
13565 |
13566 | def plot_chart_dispatcher(self, data):
13567 |     """Merender kanvas matriks berdasarkan tipe chart (Altitude / Elongasi / Iluminasi)"""
13568 |     try:
13569 |         for widget in self.frame_chart_out.wininfo_children():
13570 |             widget.destroy()
13571 |
13572 |         from matplotlib.figure import Figure
13573 |         from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
13574 |         import numpy as np
13575 |         import datetime
13576 |
13577 |         fig = Figure(figsize=(8, 5), dpi=100)
13578 |         fig.patch.set_facecolor('#181818')
13579 |         ax = fig.add_subplot(111)
13580 |         ax.set_facecolor('#101010')
13581 |
13582 |         chart_type = data['type']
13583 |         hours = data['hours']
13584 |         y, m, d = data['y'], data['m'], data['d']
13585 |
13586 |         tahun_str = f"{y} M" if y > 0 else f"{abs(y-1)} SM"
13587 |         nama_bulan = ["", "Jan", "Feb", "Mar", "Apr", "Mei", "Jun", "Jul", "Ags", "Sep", "Okt", "Nov", "Des"]
13588 |         bulan_str = nama_bulan[m] if 1 <= m <= 12 else str(m)
13589 |         date_display = f"{d:02d} {bulan_str} {tahun_str}"
13590 |
13591 |         self.chart_type = chart_type
13592 |         self.chart_data = data # Simpan referensi array untuk hover interaktif
13593 |
13594 |         if chart_type == "altitude":
13595 |             ax.plot(hours, data['sun_alts'], color='#FFD54F', label='Matahari (Topo)', linewidth=2)
13596 |             ax.plot(hours, data['moon_alts'], color='#80DEEA', label='Bulan (Topo MABIMS)', linewidth=2)
13597 |             ax.plot(hours, data['moon_geo_alts'], color='#00E676', label='Bulan (Geo KHGT)', linewidth=2, linestyle='--')
13598 |             ax.axhline(0, color='#FF5252', linestyle='--', linewidth=1.5, label='Ufuk (Horizon)')
13599 |
13600 |             ax.set_title(f"Grafik Ketinggian Benda Langit: {date_display}", color='white', fontsize=14, fontweight='bold', pad=15)
13601 |             ax.set_ylabel("Ketinggian / Altitude (°)", color='ffffff', fontsize=12)
13602 |             ax.set_xlabel("Waktu Lokal (Jam)", color='ffffff', fontsize=12)
13603 |             ax.set_xlim(0, 24)
13604 |             ax.set_xticks(np.arange(0, 25, 2))
13605 |             ax.set_xticklabels([f"{int(h):02d}:00" for h in np.arange(0, 25, 2)])
13606 |
13607 |             now = datetime.datetime.now()
13608 |             if y == now.year and m == now.month and d == now.day:
13609 |                 curr_h = now.hour + now.minute/60.0
13610 |                 ax.vline(curr_h, color='#00E676', linestyle=':', linewidth=2, alpha=0.8)
13611 |                 ax.text(curr_h + 0.2, min(data['sun_alts']), 'Waktu Sekarang', color='#00E676', fontsize=10, rotation=90, verticalalignment='bottom')

```

```

13612 |
13613 | elif chart_type == "elongation":
13614 |     ax.plot(hours, data['elongations'], color='#FFD700', label='Elongasi Bulan-Matahari (Geo)', linewidth=2.5)
13615 |     ax.axhline(8, color='#00E676', linestyle='--', linewidth=1.5, label='Batas Minimal KHGT (8°)')
13616 |     ax.axhline(6.4, color='#FF5252', linestyle=':', linewidth=1.5, label='Batas Minimal MABIMS (6.4°)')
13617 |
13618 |     ax.set_title(f"Grafik Elongasi (Jarak Sudut): {date_display}", color='white', fontsize=14, fontweight='bold', pad=15)
13619 |     ax.set_ylabel("Elongasi (°)", color='white', fontsize=12)
13620 |     ax.set_xlabel("Waktu Lokal (Jam)", color='white', fontsize=12)
13621 |     ax.set_xlim(0, 24)
13622 |     ax.set_xticks(np.arange(0, 25, 2))
13623 |     ax.set_xticklabels([f"{int(h):02d}:00" for h in np.arange(0, 25, 2)])
13624 |
13625 | elif chart_type == "illumination":
13626 |     ax.plot(hours, data['illuminations'], color='#E040FB', label='Fraksi Iluminasi Bulan (%)', linewidth=2.5)
13627 |     ax.set_title(f"Grafik Iluminasi (Cahaya Fase): {date_display}", color='white', fontsize=14, fontweight='bold', pad=15)
13628 |     ax.set_ylabel("Iluminasi (%)", color='white', fontsize=12)
13629 |     ax.set_xlabel("Waktu Lokal (Jam)", color='white', fontsize=12)
13630 |     ax.set_xlim(0, 24)
13631 |     ax.set_xticks(np.arange(0, 25, 2))
13632 |     ax.set_xticklabels([f"{int(h):02d}:00" for h in np.arange(0, 25, 2)])
13633 |
13634 | elif chart_type == "illum_vs_elong":
13635 |     ax.plot(data['elongations'], data['illuminations'], color='#2979FF', label='Korelasi Iluminasi thdp Elongasi', linewidth=2.5)
13636 |     ax.set_title(f"Iluminasi sbg Fungsi Elongasi: {date_display}", color='white', fontsize=14, fontweight='bold', pad=15)
13637 |     ax.set_ylabel("Iluminasi (%)", color='white', fontsize=12)
13638 |     ax.set_xlabel("Elongasi (°)", color='white', fontsize=12)
13639 |
13640 |     # Dinamis limit berdasarkan data
13641 |     min_e, max_e = min(data['elongations']), max(data['elongations'])
13642 |     ax.set_xlim(min_e - (max_e-min_e)*0.05, max_e + (max_e-min_e)*0.05)
13643 |
13644 |     # --- COMMON STYLING ---
13645 |     ax.tick_params(colors='white', labels=11)
13646 |     ax.grid(True, color='white', linestyle=':', alpha=0.3)
13647 |     ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='best')
13648 |     fig.tight_layout()
13649 |
13650 |     # Render Matplotlib ke Tkinter
13651 |     self.chart_ax = ax
13652 |     self.chart_canvas = FigureCanvasTkAgg(fig, master=self.frame_chart_out)
13653 |
13654 |     # --- INISIALISASI TOOL TIP HOVER CANGGIH ---
13655 |     self.annot_chart = ax.annotate(
13656 |         "", xy=(0,0), xytext=(15, 15), textcoords="offset points",
13657 |         bbox=dict(boxstyle="round,pad=0.6", fc="#1E1E1E", ec="#00E5FF", lw=1.5, alpha=0.9),
13658 |         color="white", fontsize=10, zorder=10
13659 |     )
13660 |     self.annot_chart.set_visible(False)
13661 |     self.chart_canvas.mpl_connect("motion_notify_event", self.on_hover_chart)
13662 |
13663 |     self.chart_canvas.draw()
13664 |     self.chart_canvas.get_tk_widget().pack(fill="both", expand=True)
13665 |
13666 |     self.lbl_status.configure(text="Grafik Berhasil Dimuat", text_color="#00E676")
13667 |     self.btn_hitung.configure(state="normal")
13668 |
13669 | except Exception as e:
13670 |     import traceback
13671 |     self._force_show_error(f"Gagal Merender Grafik Matplotlib:\n{str(e)}\n{traceback.format_exc()}")
13672 |
13673 | def on_hover_chart(self, event):
13674 |     """Sistem Interpolasi Hover Interaktif yang dinamis berdasarkan jenis chart"""
13675 |     if event.inaxes == getattr(self, 'chart_ax', None):
13676 |         x_val = event.xdata
13677 |         y_val = event.ydata
13678 |         if x_val is None or y_val is None: return
13679 |
13680 |         data = getattr(self, 'chart_data', None)
13681 |         if not data: return
13682 |

```

```

13683 |         chart_type = self.chart_type
13684 |         import numpy as np
13685 |
13686 |         if chart_type in ["altitude", "elongation", "illumination"]:
13687 |             if x_val < 0 or x_val > 24:
13688 |                 self.annot_chart.set_visible(False)
13689 |                 self.chart_canvas.draw_idle()
13690 |                 return
13691 |
13692 |             # Format waktu
13693 |             jam = int(x_val)
13694 |             sisa_menit = (x_val - jam) * 60.0
13695 |             menit = int(sisa_menit)
13696 |             detik = int(round((sisa_menit - menit) * 60.0))
13697 |             if detik >= 60: menit += 1; detik -= 60
13698 |             if menit >= 60: jam += 1; menit -= 60
13699 |             time_str = f"{jam:02d}:{menit:02d}:{detik:02d}"
13700 |
13701 |             hours = data['hours']
13702 |             if chart_type == "altitude":
13703 |                 c_sun = np.interp(x_val, hours, data['sun_alts'])
13704 |                 c_moon = np.interp(x_val, hours, data['moon_alts'])
13705 |                 c_geo = np.interp(x_val, hours, data['moon_geo_alts'])
13706 |                 text = (f"□ Waktu Lokal : {time_str}\n"
13707 |                        f"*□ Alt Matahari : {c_sun:+.2f}° (Topo)\n"
13708 |                        f"□ Alt Bulan : {c_moon:+.2f}° (Topo)\n"
13709 |                        f"□ Alt Bln (Geo): {c_geo:+.2f}° (KHGT)")
13710 |
13711 |             elif chart_type == "elongation":
13712 |                 c_elong = np.interp(x_val, hours, data['elongations'])
13713 |                 text = (f"□ Waktu Lokal : {time_str}\n"
13714 |                        f"□ Elongasi (Geo) : {c_elong:.2f}°")
13715 |
13716 |             elif chart_type == "illumination":
13717 |                 c_ill = np.interp(x_val, hours, data['illuminations'])
13718 |                 text = (f"□ Waktu Lokal : {time_str}\n"
13719 |                        f"□ Iluminasi Fase : {c_ill:.2f}%")
13720 |
13721 |             elif chart_type == "illum_vs_elong":
13722 |                 min_e, max_e = min(data['elongations']), max(data['elongations'])
13723 |                 if x_val < min_e or x_val > max_e:
13724 |                     self.annot_chart.set_visible(False)
13725 |                     self.chart_canvas.draw_idle()
13726 |                     return
13727 |
13728 |             # Pengurutan data penting agar np.interp tidak error pada data X
13729 |             x_arr = np.array(data['elongations'])
13730 |             y_arr = np.array(data['illuminations'])
13731 |             sort_idx = np.argsort(x_arr)
13732 |
13733 |             c_ill = np.interp(x_val, x_arr[sort_idx], y_arr[sort_idx])
13734 |             text = (f"□ Elongasi : {x_val:.2f}°\n"
13735 |                    f"□ Iluminasi: {c_ill:.2f}%")
13736 |
13737 |             self.annot_chart.xy = (x_val, y_val)
13738 |             self.annot_chart.set_text(text)
13739 |             self.annot_chart.set_visible(True)
13740 |             self.chart_canvas.draw_idle()
13741 |         else:
13742 |             if getattr(self, 'annot_chart', None) and self.annot_chart.get_visible():
13743 |                 self.annot_chart.set_visible(False)
13744 |                 self.chart_canvas.draw_idle()
13745 |
13746 |         def _force_show_error(self, err_msg):
13747 |             """Memaksa textbox muncul kembali jika terjadi error di mode grafik"""
13748 |             if hasattr(self, 'frame_chart_out'):
13749 |                 self.frame_chart_out.grid_remove()
13750 |                 self.textbox.grid(row=1, column=0, sticky="nsew")
13751 |                 self.display_error(err_msg)
13752 |
13753 |

```

```

13754 |
13755 | # =====
13756 | # MODUL 28: WINAI (AI CHATBOT ASSISTANT)
13757 | # =====
13758 | def setup_winai_ui(self):
13759 |     # --- VARIABEL & API KEY WINAI (DIRECT GEMINI API) ---
13760 |     self.winai_api_keys = [
13761 |         '***REDACTED_API_KEY***',
13762 |         '***REDACTED_API_KEY***',
13763 |         '***REDACTED_API_KEY***', '***REDACTED_API_KEY***',
13764 |         '***REDACTED_API_KEY***'
13765 |     ]
13766 |     self.winai_primary_model = 'gemini-2.5-flash'
13767 |     self.winai_tracker_file = 'last_successful_index_winai.txt'
13768 |     self.winai_db_items = []
13769 |     self.winai_db_loaded = False
13770 |
13771 |     self.winai_jawaban_terakhir = ""
13772 |     self.winai_pertanyaan_terakhir = ""
13773 |     self.winai_memori = ""
13774 |     self.winai_instruksi = ""
13775 |     self.winai_data_kal = ""
13776 |
13777 |     # Muat database lokal di background
13778 |     threading.Thread(target=self.load_winai_databases, daemon=True).start()
13779 |
13780 |     # ===== SIDEBAR INPUT (KIRI) =====
13781 |     self.frame_winai_inputs = ctk.CTkFrame(self.sidebar, fg_color="transparent")
13782 |
13783 |     winai_info = ctk.CTkFrame(self.frame_winai_inputs, fg_color="#212121")
13784 |     winai_info.pack(fill="x", padx=15, pady=10)
13785 |     ctk.CTkLabel(winai_info, text="WINAI ASSISTANT", font=("Segoe UI", 12, "bold"), text_color="#00E5FF").pack(anchor="w", padx=10, pady=(8, 2))
13786 |
13787 |     winai_date_frame = ctk.CTkFrame(winai_info, fg_color="transparent")
13788 |     winai_date_frame.pack(fill="x", padx=10, pady=(0, 10))
13789 |
13790 |     now = datetime.datetime.now()
13791 |     self.winai_entry_y, self.winai_entry_m, self.winai_entry_d = self.create_ymd_row(winai_date_frame, str(now.year), f"{now.month:02d}", f"{now.day:02d}")
13792 |
13793 |     self.winai_input_entry = ctk.CTkEntry(winai_info, placeholder_text="Ketik pertanyaan...", font=("Segoe UI", 13), height=40)
13794 |     self.winai_input_entry.pack(fill="x", padx=10, pady=10)
13795 |     self.winai_input_entry.bind("<Return>", lambda e: self.proses_pertanyaan_winai())
13796 |
13797 |     self.btn_winai_kirim = ctk.CTkButton(winai_info, text="Kirim Pertanyaan", font=("Segoe UI", 12, "bold"), fg_color="#1565C0", hover_color="#0D47A1", height=35, command=self.proses_pertanyaan_winai)
13798 |     self.btn_winai_kirim.pack(fill="x", padx=10, pady=(0, 5))
13799 |
13800 |     self.btn_winai_reset = ctk.CTkButton(winai_info, text="Reset Memori", font=("Segoe UI", 12, "bold"), fg_color="#dc3545", hover_color="#c82333", height=35, command=self.reset_memori_winai)
13801 |     self.btn_winai_reset.pack(fill="x", padx=10, pady=(5, 10))
13802 |
13803 |     # ===== MAIN FRAME OUTPUT (KANAN) =====
13804 |     self.frame_winai_out = ctk.CTkFrame(self.main_frame, fg_color="#050510", corner_radius=10)
13805 |
13806 |     header_winai = ctk.CTkFrame(self.frame_winai_out, fg_color="#181818", corner_radius=8)
13807 |     header_winai.pack(fill="x", padx=15, pady=10)
13808 |
13809 |     ctk.CTkLabel(header_winai, text="Ruang Interaksi AI", font=("Segoe UI", 16, "bold"), text_color="#00E5FF").pack(side="left", padx=15, pady=10)
13810 |
13811 |     self.btn_winai_salin = ctk.CTkButton(header_winai, text="Salin", fg_color="#28a745", hover_color="#218838", width=80, command=self.salin_teks_winai)
13812 |     self.btn_winai_salin.pack(side="right", padx=(5, 15), pady=10)
13813 |     self.btn_winai_berbagi = ctk.CTkButton(header_winai, text="Bagi", fg_color="#17a2b8", hover_color="#138496", width=80, command=self.berbagi_teks_winai)
13814 |     self.btn_winai_berbagi.pack(side="right", padx=5, pady=10)
13815 |     self.btn_winai_buka = ctk.CTkButton(header_winai, text="Log Teks", fg_color="#6c757d", hover_color="#5a6268", width=80, command=self.buka_output_winai)
13816 |     self.btn_winai_buka.pack(side="right", padx=5, pady=10)
13817 |
13818 |     # Ruang Chat - Background Putih & Teks Hitam Standar Dokumen
13819 |     self.winai_output_box = ctk.CTkTextbox(self.frame_winai_out, wrap="word", fg_color="FFFFFF", text_color="000000", font=("Segoe UI", 16))
13820 |     self.winai_output_box.pack(fill="both", expand=True, padx=15, pady=(0, 15))
13821 |
13822 |     font_nama = "Segoe UI"
13823 |     ukuran_normal = 16
13824 |

```



```

13825 |         tb = self.winai_output_box.textbox
13826 |         tb.tag_configure("info", foreground="#856404", background="#FFF3CD", font=(font_nama, 14, "italic"), spacing1=5, spacing3=10, justify="center")
13827 |         tb.tag_configure("info_internet", foreground="#0056b3", background="#e7f1ff", font=(font_nama, 14), lmargin1=15, lmargin2=15, spacing1=5, spacing3=10, borderwidth=1, relief="solid")
13828 |         tb.tag_configure("pembuka", foreground="#000000", font=(font_nama, 18, "bold"), spacing1=0, spacing2=0, spacing3=5, justify="left")
13829 |         tb.tag_configure("user_header", foreground="#0056b3", font=(font_nama, 16, "bold"), spacing1=15)
13830 |         tb.tag_configure("user_teks", background="#E3F2FD", foreground="#000000", font=(font_nama, ukuran_normal), lmargin1=10, lmargin2=10, spacing1=5, spacing3=10)
13831 |         tb.tag_configure("ai_header", foreground="#28A745", font=(font_nama, 17, "bold"), spacing1=15)
13832 |         tb.tag_configure("normal", font=(font_nama, ukuran_normal), foreground="#000000", justify="left")
13833 |         tb.tag_configure("bold", font=(font_nama, ukuran_normal, "bold"), foreground="#000000", justify="left")
13834 |         tb.tag_configure("paragraf", spacing1=5, spacing3=10, lmargin1=10, lmargin2=10, justify="left")
13835 |         tb.tag_configure("kutipan", font=(font_nama, ukuran_normal, "italic"), lmargin1=30, lmargin2=30, foreground="#333333", background="#F8F9FA")
13836 |         tb.tag_configure("kode", font=("Consolas", 14), foreground="#000000", background="#E8F5E9", spacing1=2, spacing3=2, lmargin1=15, lmargin2=15)
13837 |
13838 |         pesan_pembuka = (
13839 |             "Selamat datang di WinAI: Aplikasi AI Windows V8.0.\n"
13840 |             "Terintegrasi penuh dengan KHGT Times Engine dan Database Tarjih Muhammadiyah.\n"
13841 |         )
13842 |         self.winai_output_box.insert("0.0", pesan_pembuka, "pembuka")
13843 |         self.winai_output_box.configure(state="disabled")
13844 |
13845 |     def load_winai_databases(self):
13846 |         try:
13847 |             base_url = "https://hisabmu.com/aifikih/db_fikih_tarjih"
13848 |             res = requests.get(f"{base_url}.json", timeout=10)
13849 |             if res.status_code == 200: self.winai_db_items.extend(res.json())
13850 |             for i in range(2, 21):
13851 |                 try:
13852 |                     res = requests.get(f"{base_url}{i}.json", timeout=5)
13853 |                     if res.status_code == 200: self.winai_db_items.extend(res.json())
13854 |                     else: break
13855 |                 except: break
13856 |             res_txt = requests.get("https://hisabmu.com/aifikih/tanyajawabagama.txt", timeout=10)
13857 |             if res_txt.status_code == 200: self.winai_db_items.extend(res_txt.json())
13858 |             self.winai_db_loaded = True
13859 |         except:
13860 |             pass
13861 |
13862 |     def retrieve_winai_context(self, user_question):
13863 |         if not self.winai_db_loaded: return "Database belum siap."
13864 |         clean_text = re.sub(r'[^a-zA-Z0-9 ]', '', user_question.lower())
13865 |         words = clean_text.split()
13866 |         keywords = [w for w in words if len(w) > 3]
13867 |
13868 |         scored_items = []
13869 |         for item in getattr(self, 'winai_db_items', []):
13870 |             judul = item.get('judul', '')
13871 |             isi = item.get('isi_konten', '')
13872 |             text = f"Q: {judul}\nA: {isi}\n\n"
13873 |             text_lower = text.lower()
13874 |             score = sum(1 for kw in keywords if kw in text_lower)
13875 |             if score > 0: scored_items.append({'score': score, 'text': text})
13876 |
13877 |         scored_items.sort(key=lambda x: x['score'], reverse=True)
13878 |         internal_db_text = ""
13879 |         for si in scored_items:
13880 |             if len(internal_db_text) + len(si['text']) > 80000: break
13881 |             internal_db_text += si['text']
13882 |         return internal_db_text if internal_db_text.strip() else "Tidak ada referensi lokal yang relevan."
13883 |
13884 |     def sisipkan_teks_winai(self, teks, pengirim="AI"):
13885 |         self.winai_output_box.configure(state="normal")
13886 |         if pengirim == "Anda":
13887 |             self.winai_pertanyaan_terakhir = teks
13888 |             self.winai_output_box.insert("end", "\n\nAnda:\n", "user_header")
13889 |             self.winai_output_box.insert("end", f" {teks} \n\n", "user_teks")
13890 |         elif pengirim == "Sistem":
13891 |             self.winai_output_box.insert("end", "\n\n" + teks + "\n\n", ("info", "status_loading"))
13892 |         else:
13893 |             self.winai_output_box.insert("end", "\n\nWinAI:\n", "ai_header")
13894 |             self.winai_jawaban_terakhir = teks
13895 |

```

```

13896 |         try:
13897 |             with open("output_winai.txt", "a", encoding="utf-8") as file_out:
13898 |                 waktu_sekarang = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
13899 |                 file_out.write(f"=== TANGGAL: {waktu_sekarang} ===\nPERTANYAAN: {self.winai_pertanyaan_terakhir}\nJAWABAN:\n{teks}\n{' '*50}\n\n")
13900 |             except Exception: pass
13901 |
13902 |             # --- PEMBERSIHAN DAN FORMAT DOKUMEN STANDAR ---
13903 |             text_cleaned = re.sub(r'(?i)Asisten Fikih ini.*', '', teks, flags=re.DOTALL).strip()
13904 |
13905 |             lines = text_cleaned.split('\n')
13906 |             in_code_block = False # <-- TAMBAHAN: Deteksi Blok Kode
13907 |
13908 |             for i, line in enumerate(lines):
13909 |                 # Deteksi markdown code block (```)
13910 |                 if line.strip().startswith("```"):
13911 |                     in_code_block = not in_code_block
13912 |                     continue # Lewati baris ``` nya sendiri
13913 |
13914 |                 # Jika sedang di dalam blok kode, format sebagai Consolas murni
13915 |                 if in_code_block:
13916 |                     self.winai_output_box.insert("end", line + "\n", "kode")
13917 |                     continue
13918 |
13919 |                 # Format khusus blok kode (kalkulasi) versi lama
13920 |                 if (line.startswith(" ") and "|" in line) or line.startswith("===") or line.startswith("---"):
13921 |                     self.winai_output_box.insert("end", line + "\n", "kode")
13922 |                 # Deteksi Heading/Judul (#, ##, ###) untuk ditebalkan tanpa menampilkan '#'
13923 |                 elif re.match(r'^\s*#+\s+', line):
13924 |                     clean_line = re.sub(r'^\s*#+\s+', '', line)
13925 |                     self.winai_output_box.insert("end", clean_line + "\n", "bold")
13926 |                 else:
13927 |                     # Deteksi teks tebal (**)
13928 |                     parts = line.split('**')
13929 |                     for j, part in enumerate(parts):
13930 |                         if j % 2 == 1:
13931 |                             self.winai_output_box.insert("end", part, "bold")
13932 |                         else:
13933 |                             # Hapus simbol aneh, biarkan teks normal
13934 |                             part_clean = part.replace('*', '')
13935 |                             self.winai_output_box.insert("end", part_clean, "normal")
13936 |
13937 |                 # Baris baru
13938 |                 if i < len(lines) - 1:
13939 |                     self.winai_output_box.insert("end", "\n", "normal")
13940 |
13941 |             self.winai_output_box.insert("end", "\n\n" + "="*80 + "\n\n", "normal")
13942 |
13943 |             self.winai_output_box.see("end")
13944 |             self.winai_output_box.configure(state="disabled")
13945 |
13946 |         def tampilkan_jejak_internet(self, teks):
13947 |             self.winai_output_box.configure(state="normal")
13948 |             self.winai_output_box.insert("end", "❏ BERHASIL MENDAPATKAN DATA INTERNET TERBARU:\n\n" + teks + "\n", "info_internet")
13949 |             self.winai_output_box.see("end")
13950 |             self.winai_output_box.configure(state="disabled")
13951 |
13952 |         def hapus_status_loading_winai(self):
13953 |             self.winai_output_box.configure(state="normal")
13954 |             ranges = self.winai_output_box.tag_ranges("status_loading")
13955 |             if ranges:
13956 |                 self.winai_output_box.delete(ranges[0], ranges[-1])
13957 |             self.winai_output_box.configure(state="disabled")
13958 |
13959 |         def proses_pertanyaan_winai(self):
13960 |             pertanyaan = self.winai_input_entry.get().strip()
13961 |             if not pertanyaan: return
13962 |
13963 |             self.winai_input_entry.delete(0, "end")
13964 |             self.sisipkan_teks_winai(pertanyaan, pengirim="Anda")
13965 |
13966 |             teks_lower = pertanyaan.lower()

```

```

13967 |
13968 | # =====
13969 | # --> SMART INTERCEPTOR (AGENTIC AI CONTROL) <--
13970 | # =====
13971 | # AI mendeteksi niat (intent) penggunaan dari kata kerjanya
13972 | is_perintah = any(kata in teks_lower for kata in ["hitung", "kalkulasi", "berapa", "cek", "cari", "tampilkan", "simulasi", "buka"])
13973 |
13974 | # Deteksi Objek / Modul Sasaran
13975 | is_tanya_hilal = is_perintah and ("visibilitas" in teks_lower or "hilal" in teks_lower)
13976 | is_tanya_kota = is_perintah and any(kata in teks_lower for kata in ["kota pertama", "titik pertama", "awal bulan khgt"])
13977 | is_tanya_okultasi = is_perintah and ("okultasi" in teks_lower or "menutupi" in teks_lower)
13978 | is_tanya_gerhana = is_perintah and "gerhana" in teks_lower
13979 | is_tanya_pasang = is_perintah and any(kata in teks_lower for kata in ["pasang surut", "air laut", "gravitasi"])
13980 | is_tanya_fajar = is_perintah and any(kata in teks_lower for kata in ["fajar", "shadiq", "sqm", "kegelapan langit"])
13981 | is_tanya_animasi = is_perintah and any(kata in teks_lower for kata in ["animasi", "live", "posisi real-time"])
13982 | is_tanya_3d = is_perintah and ("3d" in teks_lower or "tata surya" in teks_lower)
13983 |
13984 | # Matikan tombol kirim sementara proses komputasi berlangsung
13985 | self.btn_winai_kirim.configure(state="disabled")
13986 |
13987 | # --- EKSEKUSI INTERCEPTOR ROUTING ---
13988 | if is_tanya_hilal:
13989 |     self.sisipkan_teks_winai("Menjalankan Modul 1 (Kalkulasi Visibilitas) secara background...", pengirim="Sistem")
13990 |     threading.Thread(target=self.hitung_dan_tanya_ai, args=(pertanyaan, "hilal"), daemon=True).start()
13991 |
13992 | elif is_tanya_kota:
13993 |     self.sisipkan_teks_winai("Menjalankan Modul 5 (Pelacakan Kota Pertama) secara background...", pengirim="Sistem")
13994 |     threading.Thread(target=self.hitung_dan_tanya_ai, args=(pertanyaan, "kota"), daemon=True).start()
13995 |
13996 | elif is_tanya_okultasi:
13997 |     self.sisipkan_teks_winai("Memicu Prediktor Okultasi di layar utama...", pengirim="Sistem")
13998 |     # Auto-pindah UI ke Menu 35
13999 |     self.combo_mode.set("35) Astrofotografi & Kontras Visibilitas")
14000 |     self.switch_mode("35) Astrofotografi & Kontras Visibilitas")
14001 |     self.calculate_occultation() # Eksekusi grafik okultasi
14002 |     # AI tetap menjawab teorinya di latar belakang
14003 |     threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan + " Tolong jelaskan secara teori astronomi apa itu fenomena okultasi.", None), daemon=True).start()
14004 |
14005 | elif is_tanya_gerhana:
14006 |     self.sisipkan_teks_winai("Memicu Scanner Gerhana Global di layar utama...", pengirim="Sistem")
14007 |     # Auto-pindah UI ke Menu 14
14008 |     self.combo_mode.set("14) Analisis Gerhana")
14009 |     self.switch_mode("14) Analisis Gerhana")
14010 |     tahun = int(self.combo_tahun_gerhana.get())
14011 |     threading.Thread(target=self.calc_gerhana_thread, args=(tahun,), daemon=True).start()
14012 |     threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan + " Berikan edukasi tentang mekanisme gerhana.", None), daemon=True).start()
14013 |
14014 | elif is_tanya_pasang:
14015 |     self.sisipkan_teks_winai("Memicu Modul Pasang Surut Gravitasi...", pengirim="Sistem")
14016 |     self.combo_mode.set("36) Prediktor Pasang Surut Gravitasi")
14017 |     self.switch_mode("36) Prediktor Pasang Surut Gravitasi")
14018 |     self.calculate_pasang_surut()
14019 |     threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan, None), daemon=True).start()
14020 |
14021 | elif is_tanya_fajar:
14022 |     self.sisipkan_teks_winai("Memicu Modul Evaluasi Fajar & SQM...", pengirim="Sistem")
14023 |     self.combo_mode.set("37) Evaluasi Fajar Shadiq & SQM")
14024 |     self.switch_mode("37) Evaluasi Fajar Shadiq & SQM")
14025 |     self.calculate_fajar_sqm()
14026 |     threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan, None), daemon=True).start()
14027 |
14028 | elif is_tanya_animasi:
14029 |     self.sisipkan_teks_winai("Membuka Modul Live Animasi...", pengirim="Sistem")
14030 |     self.combo_mode.set("15) Live Animasi")
14031 |     self.switch_mode("15) Live Animasi")
14032 |     threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan + " Katakan: Saya telah membuka layar Live Animasi untuk Anda di layar utama.", None), daemon=True).start()
14033 |
14034 | elif is_tanya_3d:
14035 |     self.sisipkan_teks_winai("Membuka Modul Simulasi Tata Surya 3D...", pengirim="Sistem")
14036 |     self.combo_mode.set("19) Simulasi Ephemeris 3D")
14037 |     self.switch_mode("19) Simulasi Ephemeris 3D")

```

```

14038 |         threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan + " Katakan: Sistem Tata Surya 3D telah disiapkan untuk Anda.", None), daemon=True).start()
14039 |
14040 |     else:
14041 |         # Pertanyaan umum fikih/astronomi biasa (Tanpa pindah layar)
14042 |         self.sisipkan_teks_winai("Mengakses Database & Menghubungi AI...", pengirim="Sistem")
14043 |         threading.Thread(target=self.tanya_ai_thread, args=(pertanyaan, None), daemon=True).start()
14044 |
14045 | def hitung_dan_tanya_ai(self, pertanyaan, mode):
14046 |     """Fungsi yang memproses hitungan terlebih dahulu, lalu mengirim datanya ke AI untuk diinterpretasi"""
14047 |     report_data = ""
14048 |     if mode == "hilal":
14049 |         try:
14050 |             y, m, d = self.winai_entry_y.get(), self.winai_entry_m.get(), self.winai_entry_d.get()
14051 |             # Sinkronkan dengan menu 1
14052 |             self.entry_vyear.delete(0, 'end'); self.entry_vyear.insert(0, y)
14053 |             self.entry_vmonth.delete(0, 'end'); self.entry_vmonth.insert(0, m)
14054 |             self.entry_vday.delete(0, 'end'); self.entry_vday.insert(0, d)
14055 |             report_data = self.generate_visibility_report()
14056 |         except Exception as e:
14057 |             report_data = f"Gagal menghitung visibilitas: {e}"
14058 |     elif mode == "kota":
14059 |         try:
14060 |             y, m, d = int(self.winai_entry_y.get()), int(self.winai_entry_m.get()), int(self.winai_entry_d.get())
14061 |             report_data = self.generate_first_point_report(y, m, d)
14062 |         except Exception as e:
14063 |             report_data = f"Gagal melacak kota pertama: {e}"
14064 |
14065 |     # Teruskan ke AI dengan membawa data hasil hitungan
14066 |     self.tanya_ai_thread(pertanyaan, report_tambahan=report_data)
14067 |
14068 | def tanya_ai_thread(self, teks_pertanyaan, report_tambahan=None):
14069 |     self.after(0, lambda: self.sisipkan_teks_winai("Membaca instruksi & Database Internal...", pengirim="Sistem"))
14070 |
14071 |     try:
14072 |         headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36"}
14073 |         resp_sys = requests.get("https://hisabmu.com/aifikih/system_prompt.php", headers=headers, timeout=15)
14074 |         resp_sys.encoding = 'utf-8'
14075 |         if resp_sys.status_code == 200 and "$systemPersona" not in resp_sys.text:
14076 |             self.winai_instruksi = resp_sys.text.strip()
14077 |         else:
14078 |             self.winai_instruksi = "PERAN: Asisten Fikih Muhammadiyah. KHGT Berlaku 1 Muharam 1447 H."
14079 |     except:
14080 |         self.winai_instruksi = "PERAN: Asisten Fikih Muhammadiyah. KHGT Berlaku 1 Muharam 1447 H."
14081 |
14082 |     if not self.winai_data_kal:
14083 |         try:
14084 |             resp_cal = requests.get(CALENDAR_DATA_URL, headers=headers, timeout=10)
14085 |             if resp_cal.status_code == 200: self.winai_data_kal = resp_cal.text.strip()
14086 |         except: pass
14087 |
14088 |     teks_bersih = teks_pertanyaan.strip()
14089 |     gabungan_internet = ""
14090 |
14091 |     if not teks_bersih.startswith("/"):
14092 |         self.after(0, lambda: self.sisipkan_teks_winai("Menelusuri informasi pendukung di internet...", pengirim="Sistem"))
14093 |         hasil_google = ""
14094 |         try:
14095 |             kata_kunci = urllib.parse.quote(teks_bersih)
14096 |             url_g = f"https://www.google.com/search?&lr=lang_id&hl=id&nem=143&udm=50&q={kata_kunci}"
14097 |             resp_g = requests.get(url_g, headers=headers, cookies=G00GLE_SESSION_C00KIES, timeout=15)
14098 |             if resp_g.status_code == 200:
14099 |                 soup = BeautifulSoup(resp_g.text, 'html.parser')
14100 |                 snippets = soup.find_all(['div', 'span'], class_='BNeawe', 'VwiC3b', 's3v9rd', 'HwtZe'])
14101 |                 kumpulan = []
14102 |                 for s in snippets:
14103 |                     t = s.get_text(strip=True)
14104 |                     if t and len(t) > 60 and t not in kumpulan: kumpulan.append(t)
14105 |                 if kumpulan: hasil_google = "[Data Referensi Google Search]:\n" + "\n".join(f"- {txt}" for txt in kumpulan[:3]) + "\n"
14106 |             except: pass
14107 |
14108 |         hasil_ddg = ""

```

```

14109 |         try:
14110 |             with DDGS() as ddgs:
14111 |                 results = list(ddgs.text(teks_bersih, max_results=2))
14112 |                 if results: hasil_ddg = "[Data Referensi DuckDuckGo]:\n" + "\n".join(f"- {r['body']}" for r in results) + "\n"
14113 |             except: pass
14114 |
14115 |             if hasil_google or hasil_ddg:
14116 |                 gabungan_internet = f"{hasil_google}\n{hasil_ddg}".strip()
14117 |                 self.after(0, self.tampilkan_jejak_internet, gabungan_internet)
14118 |
14119 |             # RAG / Local DB
14120 |             konteks_lokal = self.retrieve_winai_context(teks_bersih)
14121 |
14122 |             self.after(0, lambda: self.sisipkan_teks_winai("Menganalisis pertanyaan dan memanggil Model AI Gemini...", pengirim="Sistem"))
14123 |
14124 |             loc_prov = self.opt_prov.get()
14125 |             loc_city = self.opt_city.get()
14126 |             win_y = self.winai_entry_y.get()
14127 |             win_m = self.winai_entry_m.get()
14128 |             win_d = self.winai_entry_d.get()
14129 |
14130 |             prompt_pintar = f"=== KONTEKS PENGGUNA ===\nLokasi: {loc_city}, {loc_prov}.\nTanggal Tinjauan: {win_d}/{win_m}/{win_y}.\nWaktu Aktual: {datetime.datetime.now().strftime('%A, %d %B %Y %H:%M:%S')}\n\n"
14131 |             if self.winai_data_kal: prompt_pintar += f"=== DATA KALENDER KHGT ===\n{self.winai_data_kal}\n\n"
14132 |             if gabungan_internet: prompt_pintar += f"=== REFERENSI INTERNET ===\n{gabungan_internet}\n\n"
14133 |             if self.winai_memori: prompt_pintar += f"=== RIWAYAT OBROLAN LALU ===\n{self.winai_memori}\n\n"
14134 |
14135 |             # Injeksi Data Report Sistem (Bila Ada Perintah Hitung)
14136 |             if report_tambahan:
14137 |                 prompt_pintar += f"=== HASIL KALKULASI SISTEM SAAT INI ===\n{report_tambahan}\n\n"
14138 |
14139 |             prompt_pintar += f"=== PERTANYAAN USER ===\n{teks_bersih}\n\n"
14140 |
14141 |             # --- PERBAIKAN ATURAN: Melarang AI menyalin ulang tabel ---
14142 |             aturan = "ATURAN MENJAWAB:\n1. Jawab pertanyaan user secara luwes, ramah, dan solutif. Jangan kaku.\n2. JANGAN gunakan simbol markdown kotor (* atau #) di tengah kalimat, susun paragraf dengan rapi.\n3. Jika ada 'HASIL KALKULASI SISTEM' yang diberikan di atas, Anda CUKUP menjelaskan kesimpulannya saja secara ringkas. DILARANG KERAS menyalin atau mengetik ulang tabel laporannya, karena UI antarmuka sistem akan melampirkannya secara otomatis di layar."
14143 |
14144 |             final_instruction = self.winai_instruksi + f"\n\n=====\nREFERENSI DATABASE INTERNAL LOKAL:\n\"{konteks_lokal}\"\\n\\n=====\n{aturan}"
14145 |
14146 |             payload = {
14147 |                 "systemInstruction": {"parts": [{"text": final_instruction}]},
14148 |                 "contents": [{"role": "user", "parts": [{"text": prompt_pintar}]}],
14149 |                 "generationConfig": {"temperature": 0.2, "topP": 0.8}
14150 |             }
14151 |
14152 |             start_index = 0
14153 |             if os.path.exists(self.winai_tracker_file):
14154 |                 with open(self.winai_tracker_file, 'r') as f:
14155 |                     start_index = int(f.read().strip() or 0)
14156 |
14157 |             total_keys = len(self.winai_api_keys)
14158 |             jawaban = ""
14159 |             sukses_dijawab = False
14160 |             last_error = ""
14161 |
14162 |             for i in range(total_keys):
14163 |                 idx = (start_index + i) % total_keys
14164 |                 api_key = self.winai_api_keys[idx]
14165 |                 api_url = f"https://generativelanguage.googleapis.com/v1beta/models/{self.winai_primary_model}:generateContent?key={api_key}"
14166 |
14167 |                 try:
14168 |                     res = requests.post(api_url, headers={'Content-Type': 'application/json'}, json=payload, timeout=20)
14169 |                     if res.status_code == 200:
14170 |                         data = res.json()
14171 |                         jawaban = data['candidates'][0]['content']['parts'][0]['text']
14172 |                         with open(self.winai_tracker_file, 'w') as f:
14173 |                             f.write(str(idx))
14174 |                         sukses_dijawab = True
14175 |                         break
14176 |                     else:

```

```

14177 |         err = res.json()
14178 |         last_error = err.get('error', {}).get('message', f'HTTP {res.status_code}')
14179 |
14180 |         # --- PERBAIKAN 2: Logika Rotasi Anti-Macet ---
14181 |         # Jika error 429 (Quota Habis) atau 403 (Key Mati) atau 400 (Bad Request),
14182 |         # ABAIKAN dan LANGSUNG LANJUT uji API Key berikutnya!
14183 |         if res.status_code in [429, 403, 400]:
14184 |             continue
14185 |
14186 |         # Berhenti total HANYA JIKA model benar-benar tidak ditemukan (404)
14187 |         if res.status_code == 404:
14188 |             break
14189 |
14190 |     except requests.exceptions.RequestException as e:
14191 |         last_error = str(e)
14192 |         continue
14193 |
14194 | if not sukses_dijawab:
14195 |     jawaban = f"⚠ API Bermasalah atau Koneksi Gagal.\nDetail: {last_error}"
14196 |
14197 |     # --- TAMBAHAN PERBAIKAN: Menampilkan hisab walau AI Gagal ---
14198 |     if report_tambahan:
14199 |         jawaban += f"\n\nNamun, sistem HISAB KHGT tetap berhasil melakukan kalkulasi untuk Anda:\n\n```\n{report_tambahan}\n```"
14200 |
14201 | if sukses_dijawab:
14202 |     # --- INJEKSI OTOMATIS LAPORAN MURNI DI BELAKANG JAWABAN AI ---
14203 |     if report_tambahan:
14204 |         jawaban += f"\n\n[Laporan Asli Sistem]:\n\n```\n{report_tambahan}\n```"
14205 |
14206 |     if len(jawaban) < 2000:
14207 |         self.winai_memori += f"User: {teks_bersih}\nAI: {jawaban}\n\n"
14208 |     else:
14209 |         self.winai_memori += f"User: {teks_bersih}\nAI: [Telah memberikan jawaban panjang/dokumen]\n\n"
14210 |
14211 |     if len(self.winai_memori) > 15000:
14212 |         self.winai_memori = self.winai_memori[-15000:]
14213 |
14214 |     self.after(0, self.selesai_memproses_winai, jawaban)
14215 |
14216 | def selesai_memproses_winai(self, jawaban):
14217 |     self.hapus_status_loading_winai()
14218 |     self.sisipkan_teks_winai(jawaban, pengirim="AI")
14219 |     self.btn_winai_kirim.configure(state="normal")
14220 |
14221 | def salin_teks_winai(self):
14222 |     teks_salin = self.winai_output_box.get("1.0", "end-1c").strip()
14223 |     if not teks_salin:
14224 |         messagebox.showwarning("Peringatan", "Tidak ada teks untuk disalin.")
14225 |         return
14226 |
14227 |     self.clipboard_clear()
14228 |     self.clipboard_append(teks_salin)
14229 |     messagebox.showinfo("Sukses", "Seluruh teks di layar berhasil disalin ke Clipboard!")
14230 |
14231 | def berbagi_teks_winai(self):
14232 |     teks_share = self.winai_output_box.get("1.0", "end-1c").strip()
14233 |     if not teks_share: return
14234 |
14235 |     self.btn_winai_berbagi.configure(text="📄 Memproses...", state="disabled")
14236 |     threading.Thread(target=self.proses_berbagi_thread, args=(teks_share,), daemon=True).start()
14237 |
14238 | def proses_berbagi_thread(self, teks_share):
14239 |     payload = {'action': 'share_content', 'text': teks_share}
14240 |     try:
14241 |         # Karena PHP Server WinAI tetap butuh endpoint untuk berbagi URL
14242 |         response = requests.post("https://hisabmu.com/aifikih/system_php.php", data=payload, timeout=15)
14243 |         if response.status_code == 200:
14244 |             data = response.json()
14245 |             if data.get('status') == 'success':
14246 |                 self.after(0, self.berbagi_sukses, data.get('url'))
14247 |             else:

```

```

14248 |         self.after(0, self.berbagi_gagal, data.get('message', 'Gagal membuat link.))
14249 |     else:
14250 |         self.after(0, self.berbagi_gagal, "Error koneksi.")
14251 |     except Exception as e:
14252 |         self.after(0, self.berbagi_gagal, str(e))
14253 |
14254 | def berbagi_sukses(self, url):
14255 |     self.btn_winai_berbagi.configure(text="📄 Share", state="normal")
14256 |     webbrowser.open(url)
14257 |     self.clipboard_clear()
14258 |     self.clipboard_append(url)
14259 |     messagebox.showinfo("Sukses", f"Link berhasil dibuat!\n\n{url}")
14260 |
14261 | def berbagi_gagal(self, pesan_error):
14262 |     self.btn_winai_berbagi.configure(text="📄 Share", state="normal")
14263 |     messagebox.showerror("Gagal", f"Gagal membagikan teks:\n{pesan_error}")
14264 |
14265 | def buka_output_winai(self):
14266 |     nama_file = "output_winai.txt"
14267 |     if not os.path.exists(nama_file):
14268 |         with open(nama_file, "w", encoding="utf-8") as f:
14269 |             f.write("=== LOG OUTPUT WinAI ===\n\n")
14270 |     try: os.startfile(nama_file)
14271 |     except Exception as e: messagebox.showerror("Error", f"Gagal membuka:\n{e}")
14272 |
14273 | def reset_memori_winai(self):
14274 |     self.winai_memori = ""
14275 |     messagebox.showinfo("Reset Berhasil", "Ingatan dikosongkan. AI siap!")
14276 |
14277 |
14278 | def apply_kalender_kustom(self):
14279 |     try:
14280 |         thn = int(self.entry_kal_h_tahun.get())
14281 |         # ---> KEMBALIKAN KE BAHASA INDONESIA <---
14282 |         bln_str = self.reverse_tr(self.combo_kal_h_bulan.get())
14283 |         bln_idx = BULAN_HIJRIAH.index(bln_str)
14284 |
14285 |         if thn in HIJRI_DB:
14286 |             self.cal_h_year = thn
14287 |             self.cal_h_month = bln_idx
14288 |             self.render_kalender()
14289 |         else:
14290 |             messagebox.showwarning("Peringatan Database", f"Data awal bulan untuk tahun {thn} H belum tersedia di database internal.")
14291 |     except ValueError:
14292 |         messagebox.showerror("Error", "Tahun Hijriah harus berupa angka!")
14293 |
14294 | def apply_kalmasehi_kustom(self):
14295 |     try:
14296 |         thn = int(self.entry_kal_m_tahun.get())
14297 |         # ---> KEMBALIKAN KE BAHASA INDONESIA <---
14298 |         bln_str = self.reverse_tr(self.combo_kal_m_bulan.get())
14299 |         bln_idx = BULAN_MASEHI.index(bln_str) + 1
14300 |
14301 |         self.cal_m_year = thn
14302 |         self.cal_m_month = bln_idx
14303 |         self.render_kalmasehi()
14304 |     except ValueError:
14305 |         messagebox.showerror("Error", "Tahun Masehi harus berupa angka!")
14306 |
14307 | # =====
14308 | # EKSPANSI MODUL 14: SIMULATOR VISUAL GERHANA (MATAHARI & BULAN)
14309 | # =====
14310 | def buka_simulator_gerhana(self):
14311 |     selected_item = self.tabel_gerhana.selection()
14312 |     if not selected_item:
14313 |         messagebox.showwarning("Peringatan", "Silakan klik/pilih salah satu jadwal gerhana di tabel terlebih dahulu.")
14314 |     return
14315 |
14316 |     item_values = self.tabel_gerhana.item(selected_item[0])['values']
14317 |     if not item_values or item_values[0] == "": return
14318 |

```

```

14319 |         objek = str(item_values[0]).strip()
14320 |         jenis = str(item_values[1]).strip()
14321 |         waktu_puncak_str = str(item_values[3]).strip()
14322 |         wilayah_global = str(item_values[5]).strip() # Mengambil data wilayah terdampak
14323 |         if not waktu_puncak_str: return
14324 |
14325 |         # Buka Jendela TopLevel
14326 |         self.win_sim = ctk.CTkToplevel(self)
14327 |         self.win_sim.title(f"Telescope View - Simulasi Gerhana {objek}")
14328 |         self.win_sim.geometry("600x740")
14329 |         self.win_sim.attributes("-topmost", True)
14330 |         self.win_sim.configure(fg_color="#000000")
14331 |
14332 |         # Header UI
14333 |         header_frame = ctk.CTkFrame(self.win_sim, fg_color="transparent")
14334 |         header_frame.pack(fill="x", pady=10)
14335 |         ctk.CTkLabel(header_frame, text=f"SIMULASI GERHANA {objek.upper()}", font=("Segoe UI", 16, "bold"), text_color="#FFD54F").pack()
14336 |
14337 |         # ---> PERBAIKAN: Menambahkan wraplength agar teks wilayah panjang bisa rapi <---
14338 |         self.lbl_sim_info = ctk.CTkLabel(header_frame, text="Mengkalkulasi parameter astronomis...", font=("Consolas", 12), text_color="#00E676", wraplength=550)
14339 |         self.lbl_sim_info.pack()
14340 |
14341 |         # Canvas Simulator
14342 |         self.sim_canvas = tk.Canvas(self.win_sim, bg="#050510", width=500, height=500, highlightthickness=1, highlightbackground="#333333")
14343 |         self.sim_canvas.pack(pady=10)
14344 |
14345 |         # Frame Kontrol Animasi
14346 |         ctrl_frame = ctk.CTkFrame(self.win_sim, fg_color="transparent")
14347 |         ctrl_frame.pack(fill="x", padx=30, pady=10)
14348 |
14349 |         self.sim_slider = ctk.CTkSlider(ctrl_frame, from_=0, to=100, command=self._update_sim_frame)
14350 |         self.sim_slider.pack(fill="x", pady=5)
14351 |         self.sim_slider.set(0)
14352 |
14353 |         btn_play = ctk.CTkButton(ctrl_frame, text="▶ Play / Pause Animasi", fg_color="#1565C0", hover_color="#0D47A1", command=self._toggle_sim_play)
14354 |         btn_play.pack(pady=5)
14355 |
14356 |         self.sim_data = []
14357 |         self.sim_is_playing = False
14358 |
14359 |         # Mulai Kalkulasi di Background Thread (Kirim wilayah_global juga)
14360 |         threading.Thread(target=self._kalkulasi_data_simulator, args=(waktu_puncak_str, objek, wilayah_global), daemon=True).start()
14361 |
14362 |     def _kalkulasi_data_simulator(self, waktu_puncak_str, objek, wilayah_global):
14363 |         try:
14364 |             from skyfield.api import wgs84
14365 |             tz_wib = pytz.timezone('Asia/Jakarta')
14366 |             dt_naive = datetime.datetime.strptime(waktu_puncak_str, "%d-%m-%Y %H:%M")
14367 |             dt_wib = tz_wib.localize(dt_naive)
14368 |             t_peak = self.ts.from_datetime(dt_wib)
14369 |
14370 |             try:
14371 |                 lat = float(self.entry_vlat.get())
14372 |                 lon = float(self.entry_vlon.get())
14373 |                 elev = float(self.entry_velev.get())
14374 |                 tz = float(self.entry_vtz.get())
14375 |             except:
14376 |                 lat, lon, elev, tz = -7.0667, 110.4100, 230.0, 7.0
14377 |
14378 |             earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
14379 |             lokasi_awal = earth + wgs84.latlon(lat, lon, elevation_m=elev)
14380 |
14381 |             # Kalkulasi +/- 4 Jam dari puncak (agar cover seluruh fase)
14382 |             tt_start = t_peak.tt - (4.0 / 24.0)
14383 |             tt_end = t_peak.tt + (4.0 / 24.0)
14384 |             tt_array = np.linspace(tt_start, tt_end, 400)
14385 |
14386 |             # ---> FUNGSI PENGAMAN AREA MATH.ACOS AGAR TIDAK ERROR <---
14387 |             def calc_obscuratation(sep, r_moon, r_target):
14388 |                 if sep >= (r_target + r_moon): return 0.0
14389 |                 if sep <= abs(r_target - r_moon):

```



```

14390 |         return 100.0 if r_moon >= r_target else (r_moon**2 / r_target**2) * 100.0
14391 |
14392 |     r, R, d = r_moon, r_target, sep
14393 |     # Clamping domain arccos ke [-1.0, 1.0] agar tidak crash karena floating point error
14394 |     val1 = max(-1.0, min(1.0, (d**2 + r**2 - R**2) / (2 * d * r)))
14395 |     val2 = max(-1.0, min(1.0, (d**2 + R**2 - r**2) / (2 * d * R)))
14396 |
14397 |     part1 = r**2 * math.acos(val1)
14398 |     part2 = R**2 * math.acos(val2)
14399 |     part3 = 0.5 * math.sqrt(max(0.0, (-d + r + R) * (d + r - R) * (d - r + R) * (d + r + R)))
14400 |
14401 |     return ((part1 + part2 - part3) / (math.pi * R**2)) * 100.0
14402 |
14403 | def generate_frames(target_lokasi, is_geocentric=False):
14404 |     frames = []
14405 |     if "Matahari" in objek:
14406 |         for tt in tt_array:
14407 |             t_frame = self.ts.tt_jd(tt)
14408 |
14409 |             if is_geocentric:
14410 |                 s_app = earth.at(t_frame).observe(sun).apparent()
14411 |                 m_app = earth.at(t_frame).observe(moon).apparent()
14412 |
14413 |                 s_ra, s_dec, s_dist = s_app.radec()
14414 |                 m_ra, m_dec, m_dist = m_app.radec()
14415 |
14416 |                 sd_sun = math.degrees(math.asin(696000.0 / s_dist.km))
14417 |                 sd_moon = math.degrees(math.asin(1737.4 / m_dist.km))
14418 |
14419 |                 d_dec = m_dec.degrees - s_dec.degrees
14420 |                 d_ra_raw = (m_ra.hours - s_ra.hours + 12.0) % 24.0 - 12.0
14421 |                 d_ra = d_ra_raw * 15.0 * math.cos(s_dec.radians)
14422 |
14423 |                 sep = s_app.separation_from(m_app).degrees
14424 |                 obs_pct = calc_obscuratation(sep, sd_moon, sd_sun)
14425 |
14426 |                 dt_lokal = t_frame.utc_datetime() + datetime.timedelta(hours=tz)
14427 |                 frames.append({
14428 |                     'tipe': 'Matahari', 'waktu': dt_lokal.strftime("%H:%M:%S"),
14429 |                     'd_x': d_ra, 'd_y': d_dec, 'sd_sun': sd_sun, 'sd_moon': sd_moon,
14430 |                     'obs': obs_pct, 'sep': sep, 'alt_matahari': 90.0 # Pseudo-altitude
14431 |                 })
14432 |             else:
14433 |                 s_app = target_lokasi.at(t_frame).observe(sun).apparent()
14434 |                 m_app = target_lokasi.at(t_frame).observe(moon).apparent()
14435 |
14436 |                 s_alt, s_az, s_dist = s_app.altaz()
14437 |                 m_alt, m_az, m_dist = m_app.altaz()
14438 |
14439 |                 sd_sun = math.degrees(math.asin(696000.0 / s_dist.km))
14440 |                 sd_moon = math.degrees(math.asin(1737.4 / m_dist.km))
14441 |
14442 |                 d_alt = m_alt.degrees - s_alt.degrees
14443 |                 d_az_raw = (m_az.degrees - s_az.degrees + 180.0) % 360.0 - 180.0
14444 |                 d_az = d_az_raw * math.cos(math.radians(s_alt.degrees))
14445 |
14446 |                 sep = s_app.separation_from(m_app).degrees
14447 |                 obs_pct = calc_obscuratation(sep, sd_moon, sd_sun)
14448 |
14449 |                 dt_lokal = t_frame.utc_datetime() + datetime.timedelta(hours=tz)
14450 |                 frames.append({
14451 |                     'tipe': 'Matahari', 'waktu': dt_lokal.strftime("%H:%M:%S"),
14452 |                     'd_x': d_az, 'd_y': d_alt, 'sd_sun': sd_sun, 'sd_moon': sd_moon,
14453 |                     'obs': obs_pct, 'sep': sep, 'alt_matahari': s_alt.degrees
14454 |                 })
14455 |             else:
14456 |                 # GERHANA BULAN (Selalu Geosentrik karena bayangan bumi berpusat di sana)
14457 |                 for tt in tt_array:
14458 |                     t_frame = self.ts.tt_jd(tt)
14459 |                     m_geo = earth.at(t_frame).observe(moon).apparent()
14460 |                     s_geo = earth.at(t_frame).observe(sun).apparent()

```

```

14461 |
14462 |         m_ra, m_dec, m_dist = m_geo.radec()
14463 |         s_ra, s_dec, s_dist = s_geo.radec()
14464 |
14465 |         as_ra = (s_ra.hours + 12.0) % 24.0
14466 |         as_dec = -s_dec.degrees
14467 |
14468 |         d_ra = (m_ra.hours - as_ra) * 15.0 * math.cos(math.radians(as_dec))
14469 |         d_dec = m_dec.degrees - as_dec
14470 |         sep = math.sqrt(d_ra**2 + d_dec**2)
14471 |
14472 |         sd_m = math.degrees(math.asin(1737.4 / m_dist.km))
14473 |         pi_m = math.degrees(math.asin(6378.14 / m_dist.km))
14474 |         sd_s = math.degrees(math.asin(696000.0 / s_dist.km))
14475 |         pi_s = 8.794 / 3600.0
14476 |
14477 |         r_umbra = (pi_m + pi_s - sd_s) * 1.02
14478 |         r_penumbra = (pi_m + pi_s + sd_s) * 1.02
14479 |
14480 |         obs_pct = calc_obscuratation(sep, sd_m, r_umbra)
14481 |
14482 |         dt_lokal = t_frame.utcnow() + datetime.timedelta(hours=tz)
14483 |         frames.append({
14484 |             'tipe': 'Bulan', 'waktu': dt_lokal.strftime("%H:%M:%S"),
14485 |             'd_x': d_ra, 'd_y': d_dec, 'sd_moon': sd_m, 'r_umbra': r_umbra,
14486 |             'r_penumbra': r_penumbra, 'obs': obs_pct, 'sep': sep
14487 |         })
14488 |     return frames
14489 |
14490 | # Buat kalkulasi pertama (berbasis lokasi user yang dipilih - Toposentrik)
14491 | sim_frames = generate_frames(lokasi_awal, is_geocentric=False)
14492 | is_fallback = False
14493 |
14494 | # Saring frame
14495 | if "Matahari" in objek:
14496 |     valid_frames = [f for f in sim_frames if f['obs'] > 0.0 and f['alt_matahari'] > -1.0]
14497 | else:
14498 |     valid_frames = [f for f in sim_frames if f['sep'] < (f['r_penumbra'] + f['sd_moon'])]
14499 |
14500 | # ---> PERBAIKAN: JIKA GAGAL LOKAL, GUNAKAN PANDANGAN GEOSENTRIS (Pusat Bumi) <---
14501 | # Ini dijamin 100% selalu melihat Gerhana Matahari di manapun dia terjadi di bumi.
14502 | if not valid_frames:
14503 |     sim_frames = generate_frames(None, is_geocentric=True)
14504 |
14505 |     if "Matahari" in objek:
14506 |         valid_frames = [f for f in sim_frames if f['obs'] > 0.0]
14507 |     else:
14508 |         valid_frames = [f for f in sim_frames if f['sep'] < (f['r_penumbra'] + f['sd_moon'])]
14509 |
14510 |     is_fallback = True
14511 |
14512 | if not valid_frames:
14513 |     self.after(0, lambda: self.lbl_sim_info.configure(text=f"Gerhana {objek} gagal disimulasikan secara sistem.", text_color="#FF5252"))
14514 |     return
14515 |
14516 | peak_frame = max(valid_frames, key=lambda x: x['obs'])
14517 |
14518 | # Buffer 15 frame sebelum dan sesudah agar animasi mulus
14519 | idx_start = max(0, sim_frames.index(valid_frames[0]) - 15)
14520 | idx_end = min(len(sim_frames), sim_frames.index(valid_frames[-1]) + 15)
14521 | self.sim_data = sim_frames[idx_start:idx_end]
14522 |
14523 | # Tentukan Pesan Peringatan Teks
14524 | if is_fallback:
14525 |     pesan_info = f"⚠ TIDAK TERLIHAT LOKAL (Dialihkan ke View Pusat Bumi)\nWilayah Terdampak: {wilayah_global}"
14526 | else:
14527 |     pesan_info = f"Wilayah Terdampak: {wilayah_global}"
14528 |
14529 | self.after(0, self._siapkan_ui_animasi, peak_frame, pesan_info)
14530 |
14531 | except Exception as e:

```

```

14532 |         import traceback
14533 |         self.after(0, lambda: self.lbl_sim_info.configure(text=f"Gagal memuat simulasi: {e}", text_color="#FF5252"))
14534 |         print(traceback.format_exc())
14535 |
14536 |     def _siapkan_ui_animasi(self, peak_frame, pesan_info):
14537 |         self.sim_slider.configure(from_=0, to=len(self.sim_data)-1, number_of_steps=len(self.sim_data)-1)
14538 |         self.sim_slider.set(0)
14539 |
14540 |         info = f"Maksimum Obscuration: {peak_frame['obs']:.1f}%"
14541 |
14542 |         # Tambahkan label Geosentris jika ini berasal dari fallback pusat bumi
14543 |         if peak_frame['tipe'] == 'Matahari':
14544 |             alt_txt = "Geosentris" if peak_frame['alt_matahari'] == 90.0 else f"{peak_frame['alt_matahari']:.1f}°"
14545 |             info += f" | Alt Matahari: {alt_txt}"
14546 |
14547 |         # Tambahkan teks info peringatan lokasi
14548 |         info += f"\n{pesan_info}"
14549 |
14550 |         self.lbl_sim_info.configure(text=info)
14551 |
14552 |         self._update_sim_frame(0)
14553 |
14554 |     def _update_sim_frame(self, index):
14555 |         idx = int(float(index))
14556 |         if not self.sim_data or idx >= len(self.sim_data): return
14557 |
14558 |         frame = self.sim_data[idx]
14559 |         canvas = self.sim_canvas
14560 |         canvas.delete("all")
14561 |         cx, cy = 250, 250 # Pusat Canvas
14562 |
14563 |         if frame['tipe'] == 'Matahari':
14564 |             r_sun_px = 120
14565 |             scale = r_sun_px / frame['sd_sun']
14566 |             r_moon_px = frame['sd_moon'] * scale
14567 |             moon_x = cx + (frame['d_x'] * scale)
14568 |             moon_y = cy - (frame['d_y'] * scale) # Minus karena Y canvas terbalik
14569 |
14570 |             # Efek warna langit gelap perlahan
14571 |             kecerahan_langit = int(max(5, 255 - (frame['obs'] * 2.5)))
14572 |             warna_langit = f"#{kecerahan_langit:02x}{kecerahan_langit:02x}1a" if kecerahan_langit < 20 else "#050510"
14573 |             canvas.configure(bg=warna_langit)
14574 |
14575 |             canvas.create_line(cx, 0, cx, 500, fill="#333333", dash=(4,4))
14576 |             canvas.create_line(0, cy, 500, cy, fill="#333333", dash=(4,4))
14577 |
14578 |             canvas.create_oval(cx - r_sun_px - 5, cy - r_sun_px - 5, cx + r_sun_px + 5, cy + r_sun_px + 5, fill="", outline="#FFD54F", width=2, stipple="gray25")
14579 |             canvas.create_oval(cx - r_sun_px, cy - r_sun_px, cx + r_sun_px, cy + r_sun_px, fill="#FFEA00", outline="#FFC107")
14580 |             canvas.create_oval(moon_x - r_moon_px, moon_y - r_moon_px, moon_x + r_moon_px, moon_y + r_moon_px, fill="#0F0F0F", outline="#222222")
14581 |
14582 |             info_teks = f"Waktu Lokal: {frame['waktu']}\nTertutup (Obscuration): {frame['obs']:.2f}%\nAlt Matahari: {frame['alt_matahari']:.1f}%"
14583 |             canvas.create_text(15, 20, text=info_teks, fill="#00E676", font=("Consolas", 12, "bold"), anchor="nw")
14584 |
14585 |         else:
14586 |             # RENDER GERHANA BULAN
14587 |             scale = 130 / frame['r_umbra'] # Jari-jari Umbra dipatok 130 px
14588 |             r_umbra_px = 130
14589 |             r_pen_px = frame['r_penumbra'] * scale
14590 |             r_moon_px = frame['sd_moon'] * scale
14591 |
14592 |             moon_x = cx + (frame['d_x'] * scale)
14593 |             moon_y = cy - (frame['d_y'] * scale)
14594 |
14595 |             canvas.configure(bg="#050510")
14596 |             canvas.create_line(cx, 0, cx, 500, fill="#222222", dash=(4,4))
14597 |             canvas.create_line(0, cy, 500, cy, fill="#222222", dash=(4,4))
14598 |
14599 |             # Gambar Penumbra & Umbra (Bayangan Bumi)
14600 |             canvas.create_oval(cx - r_pen_px, cy - r_pen_px, cx + r_pen_px, cy + r_pen_px, fill="#15151A", outline="#333344", width=2)
14601 |             canvas.create_text(cx, cy - r_pen_px - 10, text="Batas Penumbra", fill="#777777", font=("Consolas", 10))
14602 |

```

```

14603 |         canvas.create_oval(cx - r_umbra_px, cy - r_umbra_px, cx + r_umbra_px, cy + r_umbra_px, fill="#2E0E0E", outline="#4A1C1C", width=2)
14604 |         canvas.create_text(cx, cy - r_umbra_px - 10, text="Batas Umbra", fill="#AA4444", font=("Consolas", 10))
14605 |
14606 |         # Efek Perubahan Warna Bulan (Blood Moon) jika masuk Umbra
14607 |         obs = frame['obs']
14608 |         if obs > 95: moon_color = "#5C1010" # Merah Darah Gelap (Totalitas)
14609 |         elif obs > 40: moon_color = "#B71C1C" # Merah
14610 |         elif obs > 0: moon_color = "#E57373" # Kemerahan
14611 |         else: moon_color = "#E0E0E0" # Bulan Purnama Normal
14612 |
14613 |         canvas.create_oval(moon_x - r_moon_px, moon_y - r_moon_px, moon_x + r_moon_px, moon_y + r_moon_px, fill=moon_color, outline="#FFFFFF")
14614 |
14615 |         info_teks = f"Waktu Lokal: {frame['waktu']}\nTertutup Umbra: {frame['obs']:.2f}%"
14616 |         canvas.create_text(15, 20, text=info_teks, fill="#FFD54F", font=("Consolas", 12, "bold"), anchor="nw")
14617 |
14618 |     def _toggle_sim_play(self):
14619 |         if not self.sim_data: return
14620 |         self.sim_is_playing = not self.sim_is_playing
14621 |         if self.sim_is_playing:
14622 |             self._anim_loop()
14623 |
14624 |     def _anim_loop(self):
14625 |         if not self.sim_is_playing or not getattr(self, 'win_sim', None) or not self.win_sim.wininfo_exists():
14626 |             self.sim_is_playing = False
14627 |             return
14628 |
14629 |         curr_val = int(self.sim_slider.get())
14630 |         next_val = curr_val + 1
14631 |
14632 |         if next_val >= len(self.sim_data):
14633 |             self.sim_is_playing = False
14634 |             return
14635 |
14636 |         self.sim_slider.set(next_val)
14637 |         self._update_sim_frame(next_val)
14638 |         self.win_sim.after(60, self._anim_loop) # Kecepatan frame animasi
14639 |
14640 |
14641 |     def _get_islamic_event(self, h_day, h_month, weekday):
14642 |         """
14643 |         Mendeteksi hari libur Islam dan jadwal puasa sunnah.
14644 |         h_month: 1-12 (Muharam - Zulhijah)
14645 |         weekday: 0-6 (0=Senin, 6=Ahad)
14646 |         """
14647 |         events = []
14648 |         color = None
14649 |         is_haram = False
14650 |
14651 |         # 1. Cek Hari Haram Puasa (Warna Merah)
14652 |         if (h_month == 10 and h_day == 1):
14653 |             events.append("1 Syawal: Idul Fitri (Haram Puasa)")
14654 |             color = "#FF5252"
14655 |             is_haram = True
14656 |         elif (h_month == 12 and h_day == 10):
14657 |             events.append("10 Zulhijah: Idul Adha (Haram Puasa)")
14658 |             color = "#FF5252"
14659 |             is_haram = True
14660 |         elif h_month == 12 and h_day in [11, 12, 13]:
14661 |             events.append("Hari Tasyriq (Haram Puasa)")
14662 |             color = "#FF5252"
14663 |             is_haram = True
14664 |
14665 |         # 2. Cek Peringatan Hari Besar Islam (Warna Ungu Muda)
14666 |         if h_month == 1 and h_day == 1: events.append("Tahun Baru Hijriah")
14667 |         if h_month == 3 and h_day == 12: events.append("Maulid Nabi Muhammad SAW")
14668 |         if h_month == 7 and h_day == 27: events.append("Isra' Mi'raj")
14669 |         if h_month == 8 and h_day == 15: events.append("Nisfu Syakban")
14670 |
14671 |         # 3. Cek Puasa Sunnah (Hanya jika bukan hari haram)
14672 |         if not is_haram:
14673 |             if h_month == 1 and h_day == 9:

```

```

14674 |         events.append("Puasa Sunnah Tasu'a")
14675 |         if not color: color = "#00E676" # Hijau
14676 |     if h_month == 1 and h_day == 10:
14677 |         events.append("Puasa Sunnah Asyura")
14678 |         if not color: color = "#00E676"
14679 |     if h_month == 12 and h_day == 8:
14680 |         events.append("Puasa Sunnah Tarwiyah")
14681 |         if not color: color = "#00E676"
14682 |     if h_month == 12 and h_day == 9:
14683 |         events.append("Puasa Sunnah Arafah")
14684 |         if not color: color = "#00E676"
14685 |     if h_month == 9:
14686 |         events.append("Puasa Ramadhan")
14687 |         if not color: color = "#8BC34A" # Hijau Muda
14688 |     if h_day in [13, 14, 15] and h_month != 9:
14689 |         events.append("Puasa Ayyamu'l Bidh")
14690 |         if not color: color = "#00B0FF" # Biru Muda (Cyan)
14691 |
14692 |     # Cek Puasa Senin & Kamis
14693 |     if weekday in [0, 3] and h_month != 9: # 0 = Senin, 3 = Kamis
14694 |         nama_hari = "Senin" if weekday == 0 else "Kamis"
14695 |         events.append(f"Puasa Sunnah {nama_hari}")
14696 |         if not color: color = "#FFA000" # Orange
14697 |
14698 |     # Set warna peringatan biasa jika tidak ada puasa/haram
14699 |     if events and not color:
14700 |         color = "#E040FB" # Ungu
14701 |
14702 |     tooltip_text = "\n".join(events) if events else ""
14703 |     return color, tooltip_text
14704 |
14705 | def calculate_mizwala(self):
14706 |     try:
14707 |         year = int(self.entry_miz_year.get())
14708 |         month = int(self.entry_miz_month.get())
14709 |         day = int(self.entry_miz_day.get())
14710 |
14711 |         self.auto_switch_ephemeris(year)
14712 |
14713 |         lat = float(self.entry_miz_lat.get())
14714 |         lon = float(self.entry_miz_lon.get())
14715 |         elev = float(self.entry_miz_elev.get())
14716 |         tz = float(self.entry_miz_tz.get())
14717 |
14718 |         tinggi_tongkat = float(self.entry_miz_tinggi.get())
14719 |
14720 |         # --- MEMAKSA INTERVAL MENJADI 1 MENIT (Sesuai kode awal) ---
14721 |         step_mnt = 1
14722 |
14723 |         earth, sun = self.eph['earth'], self.eph['sun']
14724 |         loc = wgs84.latlon(lat, lon, elevation_m=elev)
14725 |         observer = earth + loc
14726 |
14727 |         # 1. Cari Waktu Terbit dan Terbenam Lokal untuk membatasi tabel
14728 |         t0 = self.ts.utc(year, month, day, -int(tz))
14729 |         t1 = self.ts.utc(year, month, day, 24 - int(tz))
14730 |
14731 |         f_rs = almanac.sunrise_sunset(self.eph, loc)
14732 |         t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)
14733 |
14734 |         t_rise, t_set = None, None
14735 |         for t_ev, y_ev in get_safe_events(t_rs, y_rs):
14736 |             if y_ev == 1: t_rise = t_ev
14737 |             else: t_set = t_ev
14738 |
14739 |         if t_rise is None or t_set is None:
14740 |             self.after(0, self.display_error, "Matahari tidak terbit/terbenam di lokasi ini pada tanggal tersebut (Anomali Lintang).")
14741 |             return
14742 |
14743 |         # 2. Sampel Perjalanan Matahari Harian
14744 |         tt_array = np.linspace(t_rise.tt, t_set.tt, 1440)

```

```

14745 |         t_search = self.ts.tt_jd(tt_array)
14746 |         alt_arr = observer.at(t_search).observe(sun).apparent().altaz()[0].degrees
14747 |
14748 |         # --- KALKULASI DZUHUR ---
14749 |         idx_noon = np.argmax(alt_arr)
14750 |         alt_noon = alt_arr[idx_noon]
14751 |
14752 |         t_noon = self.ts.tt_jd(tt_array[idx_noon])
14753 |         noon_dt = t_noon.utc_datetime() + datetime.timedelta(hours=tz)
14754 |
14755 |         zenith_noon = 90.0 - alt_noon
14756 |         bayangan_dzuhur = tinggi_tongkat * math.tan(math.radians(max(0, zenith_noon)))
14757 |         target_shadow_asr = tinggi_tongkat + bayangan_dzuhur
14758 |
14759 |         # --- KALKULASI WAKTU DHUHA ---
14760 |         alt_am = alt_arr[:idx_noon]
14761 |         tt_am = tt_array[:idx_noon]
14762 |
14763 |         tt_dhuha = None
14764 |         diffs = alt_am - 4.5
14765 |         for i in range(len(diffs)-1):
14766 |             if diffs[i] <= 0 and diffs[i+1] > 0:
14767 |                 frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
14768 |                 tt_dhuha = tt_am[i] + frac * (tt_am[i+1] - tt_am[i])
14769 |                 break
14770 |
14771 |         dhuha_str = "----"
14772 |         if tt_dhuha is not None:
14773 |             dhuha_dt_exact = self.ts.tt_jd(tt_dhuha).utc_datetime() + datetime.timedelta(hours=tz)
14774 |             dhuha_str = dhuha_dt_exact.strftime("%H:%M:%S")
14775 |
14776 |         # 3. Format Output Header
14777 |         output_lines = []
14778 |         lat_str = format_angle(lat).replace("+", "").replace("-", "-")
14779 |         lon_str = format_angle(lon).replace("+", "")
14780 |
14781 |         # ---> PERBAIKAN: MENGAMBIL NAMA KOTA & PROVINSI DARI SIDEBAR <---
14782 |         try:
14783 |             nama_kota = self.opt_city.get()
14784 |             nama_prov = self.opt_prov.get()
14785 |             lokasi_text = f"{nama_kota}, {nama_prov} (Lat {lat_str}, Lon {lon_str}, Elev {elev}m, TZ {tz})"
14786 |         except:
14787 |             lokasi_text = f"Lat {lat_str}, Lon {lon_str}, Elev {elev}m, TZ {tz}"
14788 |
14789 |         output_lines.append(self.get_header(90))
14790 |         output_lines.append("[ Tabel Bayangan Tongkat Istiwa / Mizwala ]".center(90))
14791 |         output_lines.append("")
14792 |         output_lines.append(f"* Tanggal      : {day:02d}/{month:02d}/{format_tahun_aman(year)}")
14793 |         output_lines.append(f"* Lokasi      : {lokasi_text}")
14794 |         output_lines.append(f"* Tongkat     : {tinggi_tongkat} cm")
14795 |         output_lines.append(f"* Waktu Dhuha: {dhuha_str} (Matahari Naik 4.5°)")
14796 |         output_lines.append(f"* Byg Dzuhr  : {bayangan_dzuhur:.2f} cm (Bayangan Terpendek)")
14797 |         output_lines.append(f"* Target Ashr: {target_shadow_asr:.2f} cm (Tongkat + Byg Dzuhr)")
14798 |         output_lines.append("=*90)
14799 |         output_lines.append(f"{'Waktu (LT)':<12} | {'Alt Matahari':<14} | {'Arah Bayangan':<15} | {'Panjang Bayangan':<20}")
14800 |         output_lines.append("-" * 90)
14801 |
14802 |         # 4. Generate Data Tabel berdasarkan Step Waktu
14803 |         start_dt = t_rise.utc_datetime() + datetime.timedelta(hours=tz)
14804 |         end_dt = t_set.utc_datetime() + datetime.timedelta(hours=tz)
14805 |
14806 |         menit_awal = (start_dt.minute // step_mnt) * step_mnt + step_mnt
14807 |         curr_dt = start_dt.replace(minute=0, second=0, microsecond=0) + datetime.timedelta(minutes=menit_awal)
14808 |
14809 |         # --- SISTEM BENDERA (FLAG) ---
14810 |         dhuha_marked = False
14811 |         dzuhur_marked = False
14812 |         ashar_marked = False
14813 |
14814 |         while curr_dt <= end_dt:
14815 |             t_calc = self.ts.from_datetime((curr_dt - datetime.timedelta(hours=tz)).replace(tzinfo=pytz.utc))

```

```

14816 |
14817 |         app = observer.at(t_calc).observe(sun).apparent()
14818 |         alt, az, _ = app.altaz()
14819 |
14820 |         alt_deg = alt.degrees
14821 |         az_deg = az.degrees
14822 |
14823 |         if alt_deg > 0:
14824 |             zenith = 90.0 - alt_deg
14825 |             shadow_len = tinggi_tongkat * math.tan(math.radians(zenith))
14826 |             shadow_az = (az_deg + 180.0) % 360.0
14827 |
14828 |             waktu_str = curr_dt.strftime("%H:%M")
14829 |             alt_str = f"{alt_deg:.2f}°"
14830 |             arah_str = f"{shadow_az:.2f}°"
14831 |             panjang_str = f"{shadow_len:.2f} cm"
14832 |
14833 |             marker = ""
14834 |
14835 |             # 1. Dhuha
14836 |             if curr_dt < noon_dt and alt_deg >= 4.5 and not dhuha_marked:
14837 |                 marker = " << MASUK DHUHA"
14838 |                 dhuha_marked = True
14839 |
14840 |             # 2. Dzuhur
14841 |             elif curr_dt >= noon_dt and not dzuhur_marked:
14842 |                 marker = " << ZAWAL / DZUHUR"
14843 |                 dzuhur_marked = True
14844 |
14845 |             # 3. Ashar
14846 |             elif curr_dt > noon_dt and shadow_len >= target_shadow_asr and not ashar_marked:
14847 |                 marker = " << MASUK ASHAR"
14848 |                 ashar_marked = True
14849 |
14850 |             output_lines.append(f"{waktu_str:<12} | {alt_str:<14} | {arah_str:<15} | {panjang_str}{marker}")
14851 |
14852 |             curr_dt += datetime.timedelta(minutes=step_mnt)
14853 |
14854 |             output_lines.append("= * 90)
14855 |             output_lines.append("* Keterangan: Arah bayangan dihitung dari Utara (0°) searah jarum jam.")
14856 |
14857 |             self.after(0, self.display_result, "\n".join(output_lines))
14858 |
14859 |     except Exception as e:
14860 |         import traceback
14861 |         self.after(0, self.display_error, f"{str(e)}\n\n{traceback.format_exc()}")
14862 |
14863 | def buka_simulasi_mizwala(self):
14864 |     try:
14865 |         # 1. Ambil Parameter Input Tanggal, Lokasi, dan Tongkat
14866 |         year = int(self.entry_miz_year.get())
14867 |         month = int(self.entry_miz_month.get())
14868 |         day = int(self.entry_miz_day.get())
14869 |         lat = float(self.entry_miz_lat.get())
14870 |         lon = float(self.entry_miz_lon.get())
14871 |         elev = float(self.entry_miz_elev.get())
14872 |         tz = float(self.entry_miz_tz.get())
14873 |         tinggi_tongkat = float(self.entry_miz_tinggi.get())
14874 |
14875 |         # ---> LOGIKA PINTAR PEMBACAAN WAKTU <---
14876 |         waktu_input = self.entry_miz_waktu.get().strip().upper()
14877 |
14878 |         is_live_mode = False
14879 |         jam_awal_desimal = 12.0
14880 |
14881 |         if waktu_input == "" or waktu_input == "LIVE" or waktu_input == "SEKARANG":
14882 |             is_live_mode = True
14883 |         else:
14884 |             try:
14885 |                 waktu_input = waktu_input.replace(".", ":")
14886 |                 if ":" in waktu_input:

```

```

14887 |         h_str, m_str = waktu_input.split(":")
14888 |         jam_awal_desimal = int(h_str) + (int(m_str) / 60.0)
14889 |     else:
14890 |         jam_awal_desimal = float(waktu_input)
14891 | except Exception:
14892 |     is_live_mode = True # Fallback ke Live jika format ngawur
14893 |
14894 | self.auto_switch_ephemeris(year)
14895 | earth, sun = self.eph['earth'], self.eph['sun']
14896 | loc = wgs84.latlon(lat, lon, elevation_m=elev)
14897 | observer = earth + loc
14898 |
14899 | # 2. Hitung Waktu Terbit & Terbenam sebagai Batas Slider
14900 | t0 = self.ts.utc(year, month, day, -int(tz))
14901 | t1 = self.ts.utc(year, month, day, 24 - int(tz))
14902 | f_rs = almanac.sunrise_sunset(self.eph, loc)
14903 | t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)
14904 |
14905 | t_rise, t_set = None, None
14906 | for t_ev, y_ev in get_safe_events(t_rs, y_rs):
14907 |     if y_ev == 1: t_rise = t_ev
14908 |     else: t_set = t_ev
14909 |
14910 | if t_rise is None or t_set is None:
14911 |     messagebox.showerror("Error", "Matahari tidak terbit/terbenam pada tanggal ini.")
14912 |     return
14913 |
14914 | # Konversi waktu batas ke jam desimal lokal secara presisi
14915 | dt_rise = t_rise.utc_datetime() + datetime.timedelta(hours=tz)
14916 | dt_set = t_set.utc_datetime() + datetime.timedelta(hours=tz)
14917 | start_hr = dt_rise.hour + (dt_rise.minute / 60.0)
14918 | end_hr = dt_set.hour + (dt_set.minute / 60.0)
14919 |
14920 | # 3. Buat Jendela GUI Baru
14921 | win_sim = ctk.CTkToplevel(self)
14922 | win_sim.title("Simulasi Visual Kompas Mizwala")
14923 | win_sim.geometry("650x800")
14924 | win_sim.attributes("-topmost", True)
14925 | win_sim.configure(fg_color="#0A0A0A")
14926 |
14927 | ctk.CTkLabel(win_sim, text="SIMULASI ARAH BAYANGAN MATAHARI", font=("Segoe UI", 16, "bold"), text_color="#FFD54F").pack(pady=10)
14928 |
14929 | # 4. Inisialisasi Matplotlib Polar
14930 | fig = plt.figure(figsize=(6, 6), facecolor='#0A0A0A')
14931 | ax = fig.add_subplot(111, projection='polar')
14932 |
14933 | canvas_sim = FigureCanvasTkAgg(fig, master=win_sim)
14934 | canvas_sim.get_tk_widget().pack(fill="both", expand=True, padx=10, pady=5)
14935 |
14936 | # 5. Kontrol UI Bawah
14937 | ctrl_frame = ctk.CTkFrame(win_sim, fg_color="#1E1E1E", corner_radius=10)
14938 | ctrl_frame.pack(fill="x", padx=20, pady=10)
14939 |
14940 | top_ctrl = ctk.CTkFrame(ctrl_frame, fg_color="transparent")
14941 | top_ctrl.pack(fill="x", padx=10, pady=5)
14942 |
14943 | lbl_waktu = ctk.CTkLabel(top_ctrl, text="Waktu Lokal: --:--:--", font=("Consolas", 16, "bold"), text_color="#00E676")
14944 | lbl_waktu.pack(side="left", padx=10)
14945 |
14946 | # ---> TOMBOL TOGGLE LIVE / KUSTOM <---
14947 | self.miz_is_live = is_live_mode
14948 |
14949 | def toggle_live():
14950 |     self.miz_is_live = True
14951 |     btn_live.configure(fg_color="#2E7D32", text="☐ MODE LIVE")
14952 |
14953 | btn_live = ctk.CTkButton(top_ctrl, text="☐ MODE LIVE" if is_live_mode else "☐ MODE KUSTOM",
14954 |                         fg_color="#2E7D32" if is_live_mode else "#555555",
14955 |                         width=120, command=toggle_live)
14956 | btn_live.pack(side="right", padx=10)
14957 |

```



```

14958 |         lbl_info = ctk.CTkLabel(ctrl_frame, text="Mengkalkulasi...", font=("Consolas", 12))
14959 |         lbl_info.pack(pady=5)
14960 |
14961 |         slider_var = ctk.DoubleVar()
14962 |         max_radius = tinggi_tongkat * 3.0
14963 |
14964 |         # ---> FUNGSI RENDER UTAMA <---
14965 |         def update_plot(val, update_slider=False):
14966 |             try:
14967 |                 jam_desimal = float(val)
14968 |                 jam = int(jam_desimal)
14969 |
14970 |                 # Ekstrak presisi hingga level detik
14971 |                 sisa_menit = (jam_desimal - jam) * 60.0
14972 |                 menit = int(sisa_menit)
14973 |                 detik = int(round((sisa_menit - menit) * 60.0))
14974 |
14975 |                 if detik >= 60:
14976 |                     menit += 1; detik -= 60
14977 |                 if menit >= 60:
14978 |                     jam += 1; menit -= 60
14979 |
14980 |                 if update_slider:
14981 |                     slider_var.set(jam_desimal)
14982 |
14983 |                 lbl_waktu.configure(text=f"Waktu Lokal: {jam:02d}:{menit:02d}:{detik:02d}")
14984 |
14985 |                 # Cegah time out of bounds yang memicu error kalender
14986 |                 jam_safe = min(23, max(0, jam))
14987 |
14988 |                 base_local = datetime.datetime(year, month, day)
14989 |                 curr_local = base_local + datetime.timedelta(hours=jam_safe, minutes=menit, seconds=detik)
14990 |                 curr_utc = curr_local - datetime.timedelta(hours=tz)
14991 |
14992 |                 t_calc = self.ts.from_datetime(curr_utc.replace(tzinfo=pytz.utc))
14993 |                 app = observer.at(t_calc).observe(sun).apparent()
14994 |                 alt, az, _ = app.altaz()
14995 |
14996 |                 ax.clear()
14997 |                 ax.set_theta_zero_location('N')
14998 |                 ax.set_theta_direction(-1)
14999 |                 ax.set_facecolor('#121212')
15000 |                 ax.tick_params(colors='white')
15001 |                 ax.grid(color='#333333', linestyle='--')
15002 |                 ax.set_ylim(0, max_radius)
15003 |
15004 |                 ax.set_xticks(np.radians([0, 45, 90, 135, 180, 225, 270, 315]))
15005 |                 ax.set_xticklabels(['U (0°)', 'TL', 'T (90°)', 'TG', 'S (180°)', 'BD', 'B (270°)', 'BL'])
15006 |                 ax.set_yticklabels([])
15007 |
15008 |                 if alt.degrees > 0:
15009 |                     zenith = 90.0 - alt.degrees
15010 |                     shadow_len = tinggi_tongkat * math.tan(math.radians(zenith))
15011 |                     shadow_az = (az.degrees + 180.0) % 360.0
15012 |
15013 |                     ax.plot(0, 0, marker='o', color='white', markersize=8, label=f"Tongkat ({tinggi_tongkat}cm)")
15014 |
15015 |                     display_shadow_len = min(shadow_len, max_radius)
15016 |                     ax.plot([0, math.radians(shadow_az)], [0, display_shadow_len], color='#00E5FF', linewidth=3.5, label="Arah Bayangan")
15017 |                     ax.plot(math.radians(az.degrees), max_radius * 0.9, marker='o', color='#FFD54F', markersize=14, label="Matahari")
15018 |
15019 |                     lbl_info.configure(text=f"Alt Mth: {alt.degrees:.1f}° | Arah Byg: {shadow_az:.1f}° | Pjg Byg: {shadow_len:.1f} cm", text_color="white")
15020 |                     ax.legend(loc='lower left', bbox_to_anchor=(-0.1, -0.15), facecolor='#0A0A0A', edgecolor='#333', labelcolor='white')
15021 |                 else:
15022 |                     lbl_info.configure(text="Matahari di bawah Ufuk (Malam Hari)", text_color="#FF5252")
15023 |
15024 |                 canvas_sim.draw_idle()
15025 |
15026 |             except Exception as ex:
15027 |                 print("Error rendering:", ex)
15028 |

```

```

15029 | # ---> JIKA SLIDER DIGESER MANUAL <---
15030 | def on_slider_drag(val):
15031 |     self.miz_is_live = False
15032 |     btn_live.configure(fg_color="#555555", text="MODE KUSTOM")
15033 |     update_plot(val, update_slider=False)
15034 |
15035 |     slider = ctk.CTkSlider(
15036 |         ctrl_frame,
15037 |         from_=start_hr,
15038 |         to=end_hr,
15039 |         variable=slider_var,
15040 |         command=on_slider_drag,
15041 |         progress_color="#F57C00",
15042 |         button_color="#E65100"
15043 |     )
15044 |     slider.pack(fill="x", padx=20, pady=(10, 20))
15045 |
15046 | # ---> LOOP ANIMASI REAL-TIME (TICK PER DETIK) <---
15047 | def live_loop():
15048 |     if not win_sim.wininfo_exists():
15049 |         return # Stop memori memanggil loop jika window sudah disilang/ditutup
15050 |
15051 |     if getattr(self, 'miz_is_live', False):
15052 |         now = datetime.datetime.now()
15053 |         jam_sekarang_desimal = now.hour + (now.minute / 60.0) + (now.second / 3600.0)
15054 |
15055 |         # Jika waktu real-time komputer saat ini masih berada di dalam jadwal siang hari (terbit - terbenam)
15056 |         if start_hr <= jam_sekarang_desimal <= end_hr:
15057 |             update_plot(jam_sekarang_desimal, update_slider=True)
15058 |         else:
15059 |             # Jika sudah malam, stop rendernya agar garis tidak kacau, cukup gerakkan angkanya
15060 |             lbl_info.configure(text="Matahari di bawah Ufuk (Malam Hari)", text_color="#FF5252")
15061 |             lbl_waktu.configure(text=f"Waktu Lokal: {now.strftime('%H:%M:%S')}")
15062 |             ax.clear()
15063 |             ax.set_theta_zero_location('N')
15064 |             ax.set_theta_direction(-1)
15065 |             ax.set_facecolor('#121212')
15066 |             ax.set_ylim(0, max_radius)
15067 |             ax.set_xticks(np.radians([0, 45, 90, 135, 180, 225, 270, 315]))
15068 |             ax.set_xticklabels(['U (0°)', 'TL', 'T (90°)', 'TG', 'S (180°)', 'BD', 'B (270°)', 'BL'])
15069 |             ax.set_yticklabels([])
15070 |             canvas_sim.draw_idle()
15071 |             slider_var.set(start_hr)
15072 |
15073 |             # Ulangi setiap 1 detik
15074 |             win_sim.after(1000, live_loop)
15075 |
15076 | # Nyalakan engine live loop
15077 | live_loop()
15078 |
15079 | # ---> PENYETELAN WAKTU AWAL SAAT PERTAMA KALI JENDELA DIBUKA <---
15080 | if not is_live_mode:
15081 |     # Mode Kustom (Sesuai dengan yang Anda input misal "09:00")
15082 |     if start_hr <= jam_awal_desimal <= end_hr:
15083 |         update_plot(jam_awal_desimal, update_slider=True)
15084 |     else:
15085 |         messagebox.showwarning("Peringatan Waktu", f"Waktu yang Anda input ({waktu_input}) terjadi saat malam hari (matahari di bawah ufuk).\n\nSimulasi dikembalikan ke mode Live.")
15086 |         self.miz_is_live = True
15087 |         btn_live.configure(fg_color="#2E7D32", text="MODE LIVE")
15088 |     else:
15089 |         # Jika user mengisi dengan "LIVE" atau membiarkannya kosong
15090 |         now = datetime.datetime.now()
15091 |         jam_sekarang_desimal = now.hour + (now.minute / 60.0) + (now.second / 3600.0)
15092 |         if not (start_hr <= jam_sekarang_desimal <= end_hr):
15093 |             # Jika dibuka saat malam hari, arahkan slider ke posisi Dzuhur agar user bisa melihat simulasi default
15094 |             tengah_hari = (start_hr + end_hr) / 2.0
15095 |             self.miz_is_live = False
15096 |             btn_live.configure(fg_color="#555555", text="MODE KUSTOM")
15097 |             update_plot(tengah_hari, update_slider=True)
15098 |
15099 | except Exception as e:

```

```

15100 |         import traceback
15101 |         messagebox.showerror("Error", f"Gagal memuat simulasi: {e}\n{traceback.format_exc()}")
15102 |
15103 |     def _build_hijri_db_thread(self):
15104 |         try:
15105 |             start_year = int(self.entry_ab_start.get())
15106 |             end_year = int(self.entry_ab_end.get())
15107 |         except ValueError:
15108 |             self.after(0, lambda: messagebox.showerror("Error", "Tahun harus berupa angka.))
15109 |             self.after(0, lambda: self.btn_hitung.configure(state="normal"))
15110 |             return
15111 |
15112 |         self.after(0, lambda: self.lbl_status.configure(text=f"Membangun DB Hijriah {start_year}-{end_year}...", text_color="#00E5FF"))
15113 |         self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
15114 |         self.after(0, lambda: self.textbox.delete("1.0", "end"))
15115 |         self.after(0, lambda: self.textbox.insert("end", f"Memulai iterasi mesin KHGT dari tahun {start_year} H hingga {end_year} H...\nProses ini memerlukan waktu pemrosesan CPU, mohon tunggu...\n\n"))
15116 |         self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
15117 |
15118 |         new_db = {}
15119 |         hari_nama = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Ahad"]
15120 |         bulan_masehi_singkat = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
15121 |
15122 |         try:
15123 |             for y_h in range(start_year, end_year + 1):
15124 |                 # =====
15125 |                 # PERBAIKAN: Hitung estimasi tahun Masehi per iterasi (bukan di luar loop).
15126 |                 # Auto-Switch otomatis memakai de440s.bsp atau de441-new.bsp sesuai tahun
15127 |                 # saat tahun target keluar dari rentang de440s, sehingga tidak terjadi EphemerisRangeError.
15128 |                 # =====
15129 |                 approx_greg_curr = int(y_h * 0.970224 + 622.54)
15130 |                 self.auto_switch_ephemeris(approx_greg_curr)
15131 |
15132 |                 year_data = []
15133 |                 for m_h in range(1, 13):
15134 |                     # 1. Cari TT Ijtimak bulan ini
15135 |                     approx_tt_curr = self.get_approx_nm_tt(y_h, m_h)
15136 |                     nm_list_curr = self.get_new_moons_in_range(approx_tt_curr - 5.0, approx_tt_curr + 5.0)
15137 |                     if not nm_list_curr: raise ValueError(f"Fase Bulan Baru tidak ditemukan untuk {y_h}-{m_h}")
15138 |                     nm_curr = nm_list_curr[0]
15139 |                     start_tt_curr = self.calculate_khgt_1st_of_month(nm_curr)
15140 |
15141 |                     # 2. Cari TT Ijtimak bulan depan (untuk menghitung jumlah hari)
15142 |                     next_m = m_h + 1
15143 |                     next_y = y_h
15144 |                     if next_m > 12:
15145 |                         next_m = 1
15146 |                         next_y += 1
15147 |
15148 |                     approx_tt_next = self.get_approx_nm_tt(next_y, next_m)
15149 |                     nm_list_next = self.get_new_moons_in_range(approx_tt_next - 5.0, approx_tt_next + 5.0)
15150 |                     nm_next = nm_list_next[0]
15151 |                     start_tt_next = self.calculate_khgt_1st_of_month(nm_next)
15152 |
15153 |                     # 3. Hitung Jumlah Hari (Selisih waktu mulai bulan depan dan bulan ini)
15154 |                     jumlah_hari = int(round(start_tt_next - start_tt_curr))
15155 |
15156 |                     # 4. Format String Tanggal Masehi dan Hari Pasaran (Standar)
15157 |                     t_obj = self.ts.tt_jd(start_tt_curr)
15158 |                     gy, gm, gd, _, _, _ = t_obj.utc
15159 |                     hari_idx = int(t_obj.whole % 7)
15160 |                     nama_h = hari_nama[hari_idx]
15161 |
15162 |                     tgl_str = f"{int(gd):02d}-{bulan_masehi_singkat[int(gm)-1]}-{int(gy)}"
15163 |
15164 |                     year_data.append([BULAN_HIJRIAH[m_h-1], nama_h, tgl_str, jumlah_hari])
15165 |
15166 |                 new_db[y_h] = year_data
15167 |
15168 |                 # Update log GUI tanpa freeze
15169 |                 self.after(0, lambda y=y_h: self.textbox.insert("end", f"✓ Tahun {yr} H selesai diproses...\n"))
15170 |                 self.after(0, lambda: self.textbox.see("end"))

```

```

15171 |
15172 |         # Save ke JSON
15173 |         db_path = os.path.join(BASE_DIR, "db_hijriah.json")
15174 |         with open(db_path, "w", encoding="utf-8") as f:
15175 |             json.dump(new_db, f, indent=4, ensure_ascii=False)
15176 |
15177 |         # Update Memori Aplikasi (Live tanpa perlu restart)
15178 |         global HIJRI_DB
15179 |         HIJRI_DB.update(new_db)
15180 |
15181 |         self.after(0, lambda: self.textbox.insert("end", f"\n[+] SUKSES! Database berhasil dibangun dan disimpan ke:\n{db_path}\nData telah dimuat ke dalam memori aplikasi sehingga Kalender sekarang
sudah ter-update secara Live.))
15182 |         self.after(0, lambda: self.lbl_status.configure(text="Auto-Builder Selesai", text_color="#00E676"))
15183 |
15184 |     except Exception as e:
15185 |         import traceback
15186 |         self.after(0, lambda err=str(e), tb=traceback.format_exc(): self.textbox.insert("end", f"\nERROR: {err}\n{tb}"))
15187 |         self.after(0, lambda: self.lbl_status.configure(text="Auto-Builder Gagal", text_color="#FF1744"))
15188 |     finally:
15189 |         self.after(0, lambda: self.btn_hitung.configure(state="normal"))
15190 |         self.after(0, lambda: self.textbox.configure(state="disabled"))
15191 |
15192 | def setup_kriteria_batas_out_frame(self):
15193 |     """Mempersiapkan area canvas untuk Matplotlib Kriteria Batas 31"""
15194 |     self.frame_kriteria_batas_out = ctk.CTkFrame(self.main_frame, fg_color="transparent")
15195 |
15196 |     # Setup Figure Matplotlib (Theme disesuaikan untuk Dark Mode)
15197 |     self.fig_kb, self.ax_kb = plt.subplots(figsize=(8, 6), facecolor='#101010')
15198 |     self.ax_kb.set_facecolor('#101010')
15199 |
15200 |     elongation_max = 15 # Diubah menjadi 15 agar lebih lebar
15201 |     altitude_max = 15
15202 |     altitude_min = -5 # Diubah agar nilai minus (bawah ufuk) kelihatan
15203 |
15204 |     # Pembuatan Zona Parameter
15205 |     self.ax_kb.fill_between([0, 7], altitude_min, altitude_max, color='#FF5252', alpha=0.3, label='Batas Danjon (< 7°)')
15206 |     self.ax_kb.fill_between([7, elongation_max], altitude_min, 5, color='#FFB74D', alpha=0.3, label='Bias Senja (< 5°)')
15207 |     self.ax_kb.fill_between([7, 8], 5, altitude_max, color='#FFF176', alpha=0.3, label='Margin Keamanan (7°-8°)')
15208 |     self.ax_kb.fill_between([8, elongation_max], 5, altitude_max, color='#69F0AE', alpha=0.3, label='Kriteria KHGT Terpenuhi')
15209 |
15210 |     self.ax_kb.axhline(y=0, color='gray', linestyle='--', linewidth=1.5) # Garis Nol (Horizon)
15211 |     self.ax_kb.axvline(x=8, color='#00E676', linestyle='--', linewidth=2)
15212 |     self.ax_kb.axhline(y=5, color='#00E676', linestyle='--', linewidth=2)
15213 |     self.ax_kb.axvline(x=7, color='#FF1744', linestyle=':', linewidth=2)
15214 |
15215 |     self.ax_kb.text(3.5, 6.5, 'MUSTAHIL\n(Secara Fisis)', ha='center', va='center', fontsize=12, fontweight='bold', color='#FF5252')
15216 |     self.ax_kb.text(8.5, 2.5, 'TIDAK TERLIHAT\n(Tertutup Cahaya Senja)', ha='center', va='center', fontsize=10, fontweight='bold', color='#FFB74D')
15217 |     self.ax_kb.text(7.5, 8, 'RAWAN', ha='center', va='center', fontsize=10, fontweight='bold', color='#FFF176', rotation=90)
15218 |     self.ax_kb.text(9, 8, 'RUKYAT BIL \ILMI', ha='center', va='center', fontsize=10, fontweight='bold', color='#00E676')
15219 |
15220 |     self.titik_hilal_kb, = self.ax_kb.plot([8.5], [6.0], 'w*', markersize=18, zorder=5, markeredgecolor='black', label='Titik Simulasi Real-time')
15221 |
15222 |     self.teks_info_bintang = self.ax_kb.text(8.5, 6.6, 'Eln: 8.50° | Alt: 6.00°',
15223 |                                             color='white', fontsize=11, fontweight='bold',
15224 |                                             ha='center', va='bottom', zorder=6,
15225 |                                             bbox=dict(boxstyle="round,pad=0.4", fc="#1E1E1E", ec="#00E5FF", alpha=0.9))
15226 |
15227 |     self.ax_kb.set_xlim(0, elongation_max)
15228 |     self.ax_kb.set_ylim(altitude_min, altitude_max)
15229 |     self.ax_kb.set_xlabel('Elongasi / Jarak Sudut Bulan-Matahari (Derajat)', fontsize=12, color='white')
15230 |     self.ax_kb.set_ylabel('Ketinggian Hilal / Altitude (Derajat)', fontsize=12, color='white')
15231 |     self.ax_kb.set_title('Anatomi Kriteria KHGT 5-8', fontsize=14, fontweight='bold', pad=15, color='white')
15232 |     self.ax_kb.grid(True, linestyle='--', alpha=0.2, color='white')
15233 |     self.ax_kb.tick_params(colors='white')
15234 |
15235 |     # Legend
15236 |     legend = self.ax_kb.legend(loc='upper left', fontsize='9', facecolor='#1E1E1E', edgecolor='#333333')
15237 |     for text in legend.get_texts():
15238 |         text.set_color("white")
15239 |
15240 |     self.fig_kb.tight_layout()

```

```

15241 |
15242 |     # Integrasi ke Tkinter
15243 |     self.canvas_kb = FigureCanvasTkAgg(self.fig_kb, master=self.frame_kriteria_batas_out)
15244 |     self.canvas_kb.get_tk_widget().pack(fill="both", expand=True, padx=10, pady=10)
15245 |
15246 |     # Status Box di bawah grafik
15247 |     status_frame = ctk.CTkFrame(self.frame_kriteria_batas_out, fg_color="#1A1A1A", corner_radius=8, border_width=1, border_color="#333333")
15248 |     status_frame.pack(fill="x", padx=15, pady=(0, 15))
15249 |
15250 |     self.kb_status_label = ctk.CTkLabel(status_frame, text="STATUS: -", font=("Consolas", 16, "bold"))
15251 |     self.kb_status_label.pack(pady=(10, 5))
15252 |
15253 |     self.kb_alasan_label = ctk.CTkLabel(status_frame, text="ALASAN: -", font=("Segoe UI", 13), justify="center", wraplength=700)
15254 |     self.kb_alasan_label.pack(pady=(0, 10), padx=20)
15255 |
15256 | def update_kriteria_batas_plot(self):
15257 |     """Memperbarui grafik dan penjelasan teks Menu 31 saat slider digeser"""
15258 |     if not hasattr(self, 'ax_kb'): return
15259 |
15260 |     x = self.var_kb_elongasi.get()
15261 |     y = self.var_kb_ketinggian.get()
15262 |
15263 |     self.titik_hilal_kb.set_xdata([x])
15264 |     self.titik_hilal_kb.set_ydata([y])
15265 |
15266 |     # ---> TAMBAHAN: Update Isi Angka & Posisi Kotak Info <---
15267 |     # Logika pintar: jika bintang terlalu tinggi (dekat batas atas grafik),
15268 |     # teks ditaruh di bawah bintang agar tidak terpotong (keluar layar).
15269 |     offset_y = -1.2 if y > 13 else 0.6
15270 |     self.teks_info_bintang.set_position((x, y + offset_y))
15271 |     self.teks_info_bintang.set_text(f"Eln: {x:.2f}° | Alt: {y:.2f}°")
15272 |     # -----
15273 |
15274 |     if x < 7:
15275 |         status_msg = "STATUS: MUSTAHIL (Berada di Bawah Batas Danjon - Fisis)"
15276 |         color_msg = '#FF5252'
15277 |         penjelasan_msg = ("ALASAN: Sabit bulan secara fisis belum terbentuk. Pada sudut elongasi di bawah 7 derajat,\n"
15278 |                             "bayangan pegunungan di permukaan bulan memutus pantulan cahaya matahari (Limit Danjon).\n"
15279 |                             "Pengamatan hilal pada area ini mustahil dilakukan secara optik maupun fisis.")
15280 |
15281 |     elif x >= 7 and y < 5:
15282 |         status_msg = "STATUS: TIDAK TERLIHAT (Kalah Terang oleh Bias Senja - Optis)"
15283 |         color_msg = '#FFB74D'
15284 |         penjelasan_msg = ("ALASAN: Sabit mungkin sudah terbentuk secara fisis, namun ketinggian di bawah 5 derajat\n"
15285 |                             "membuat hilal kalah terang oleh cahaya senja (Twilight Glare) di atmosfer bumi.\n"
15286 |                             "Kontras cahaya sabit tidak mencukupi untuk mengalahkan bias cahaya ufuk barat.")
15287 |
15288 |     elif 7 <= x < 8 and y >= 5:
15289 |         status_msg = "STATUS: RAWAN (Masuk dalam Margin Keamanan Observasi)"
15290 |         color_msg = '#FFF176'
15291 |         penjelasan_msg = ("ALASAN: Berada di zona transisi. Meskipun ketinggian di atas ufuk sudah cukup,\n"
15292 |                             "elongasi di bawah 8 derajat dianggap belum aman dari ketidakteraturan permukaan bulan.\n"
15293 |                             "Kriteria KHGT menetapkan 8 derajat sebagai batas margin keamanan (Safety Margin).")
15294 |
15295 |     else:
15296 |         status_msg = "STATUS: RUKYAT BIL 'ILMI (Hilal Sah & Awal Bulan Serentak!)"
15297 |         color_msg = '#00E676'
15298 |         penjelasan_msg = ("ALASAN: Memenuhi kriteria global Istanbul 2016. Pada posisi ini, hilal memiliki elongasi\n"
15299 |                             "yang cukup untuk membentuk sabit utuh dan ketinggian yang cukup untuk mengalahkan bias\n"
15300 |                             "cahaya senja. Secara saintifik terbukti wujud (Rukyat bil 'Ilmi).")
15301 |
15302 |     self.kb_status_label.configure(text=status_msg, text_color=color_msg)
15303 |     self.kb_alasan_label.configure(text=penjelasan_msg)
15304 |     self.canvas_kb.draw_idle()
15305 |
15306 | def calculate_kriteria_batas(self):
15307 |     """Menghitung ketinggian dan elongasi real-time berdasarkan input di Menu 31"""
15308 |     try:
15309 |         y = int(self.entry_kb_year.get())
15310 |         m = int(self.entry_kb_month.get())
15311 |         d = int(self.entry_kb_day.get())

```

```

15312 |         lat = float(self.entry_kb_lat.get())
15313 |         lon = float(self.entry_kb_lon.get())
15314 |         elev = float(self.entry_kb_elev.get())
15315 |         tz = float(self.entry_kb_tz.get())
15316 |
15317 |         earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
15318 |         lokasi_obs = wgs84.latlon(lat, lon, elevation_m=elev)
15319 |
15320 |         # Cari Waktu Maghrib (Sunset)
15321 |         t0 = self.ts.utc(y, m, d, 4 - int(tz))
15322 |         t1 = self.ts.utc(y, m, d, 24 - int(tz))
15323 |
15324 |         t_sunset = None
15325 |         t_evs, y_evs = almanac.find_discrete(t0, t1, almanac.sunrise_sunset(self.eph, lokasi_obs))
15326 |         for t_ev, is_sunrise in get_safe_events(t_evs, y_evs):
15327 |             if not is_sunrise:
15328 |                 t_sunset = t_ev
15329 |             break
15330 |
15331 |         if t_sunset is None:
15332 |             raise ValueError("Matahari tidak terbenam pada tanggal dan lokasi tersebut (Anomali Kutub).")
15333 |
15334 |         # Kalkulasi Geosentris (Karena Kriteria KHGT menggunakan Geosentrik)
15335 |         geo_earth = earth.at(t_sunset)
15336 |         app_moon_geo = geo_earth.observe(moon).apparent()
15337 |         app_sun_geo = geo_earth.observe(sun).apparent()
15338 |
15339 |         ra_moon, dec_moon, _ = app_moon_geo.radec(epoch=t_sunset)
15340 |
15341 |         gast = t_sunset.gast
15342 |         lst_deg = (gast * 15.0) + lon
15343 |
15344 |         # Rumus Manual Altitude Geosentris
15345 |         ra_h = ra_moon.hours.item() if hasattr(ra_moon.hours, 'item') else ra_moon.hours
15346 |         dec_r = dec_moon.radians.item() if hasattr(dec_moon.radians, 'item') else dec_moon.radians
15347 |         ha_deg = lst_deg - (ra_h * 15.0)
15348 |         lat_rad, ha_rad = math.radians(lat), math.radians(ha_deg)
15349 |         sin_alt = math.sin(dec_r) * math.sin(lat_rad) + math.cos(dec_r) * math.cos(lat_rad) * math.cos(ha_rad)
15350 |
15351 |         alt_moon_geo = math.degrees(math.asin(max(-1.0, min(1.0, sin_alt))))
15352 |         elong_geo = app_sun_geo.separation_from(app_moon_geo).degrees
15353 |
15354 |         # Lempar ke UI (Atur Slider sesuai hisab real-time, grafik akan otomatis mengikuti karena ada fungsi Trace)
15355 |         self.after(0, lambda: self.var_kb_elongasi.set(round(elong_geo, 2)))
15356 |         self.after(0, lambda: self.var_kb_ketinggian.set(round(alt_moon_geo, 2)))
15357 |         self.after(0, lambda: self.lbl_status.configure(text="Data Real-Time berhasil dimuat!", text_color="#00E676"))
15358 |         self.after(0, lambda: self.btn_hitung.configure(state="normal"))
15359 |
15360 |     except Exception as e:
15361 |         import traceback
15362 |         self.after(0, self.display_error, f"Gagal menghitung kriteria batas:\n{str(e)}\n\n{traceback.format_exc()}")
15363 |
15364 | # =====
15365 | # MODUL 31: KALKULATOR ASTROFOTOGRAFI & KONTRAS
15366 | # =====
15367 | def setup_menu_31_astrofotografi(self):
15368 |     # Bersihkan frame utama (Sesuaikan dengan cara Anda membersihkan frame)
15369 |     for widget in self.main_frame.winfo_children():
15370 |         widget.destroy()
15371 |
15372 |     # Header Modul
15373 |     header_frame = ctk.CTkFrame(self.main_frame, fg_color="#1a1a2e", corner_radius=10)
15374 |     header_frame.pack(fill="x", padx=15, pady=10)
15375 |     ctk.CTkLabel(header_frame, text="Modul 31: Astrofotografi Hilal & Visibilitas Kontras", font=("Segoe UI", 18, "bold"), text_color="#00E5FF").pack(pady=10)
15376 |
15377 |     # Frame Input Data
15378 |     input_frame = ctk.CTkFrame(self.main_frame, fg_color="transparent")
15379 |     input_frame.pack(fill="x", padx=15, pady=5)
15380 |
15381 |     # Input Tanggal
15382 |     ctk.CTkLabel(input_frame, text="Tanggal (YYYY-MM-DD):", font=("Segoe UI", 12)).grid(row=0, column=0, padx=5, pady=5, sticky="e")

```

```

15383 |         self.entry_astro_date = ctk.CTkEntry(input_frame, width=120)
15384 |         self.entry_astro_date.grid(row=0, column=1, padx=5, pady=5)
15385 |         now = datetime.datetime.now()
15386 |         self.entry_astro_date.insert(0, f"{now.year}-{now.month:02d}-{now.day:02d}")
15387 |
15388 |         # Tombol Hitung
15389 |         btn_hitung_astro = ctk.CTkButton(input_frame, text="Hitung Jendela Astrofotografi", command=self.hitung_astrofotografi, fg_color="#E65100", hover_color="#EF6C00")
15390 |         btn_hitung_astro.grid(row=0, column=2, padx=15, pady=5)
15391 |
15392 |         # Area Teks Laporan
15393 |         self.txt_astro_report = ctk.CTkTextbox(self.main_frame, width=600, height=350, font=("Consolas", 14), fg_color="#0f0f1a", text_color="#FFFFFF")
15394 |         self.txt_astro_report.pack(fill="both", expand=True, padx=15, pady=10)
15395 |         self.txt_astro_report.insert("0.0", "Tekan 'Hitung' untuk menganalisis kontras hilal dan Danjon Limit...")
15396 |
15397 |         # =====
15398 |         # HELPER CUACA UNIVERSAL (ANTI-BUG TKINTER)
15399 |         # =====
15400 |         def fetch_and_update_weather(self, lat, lon, target_label):
15401 |             def task():
15402 |                 try:
15403 |                     import requests
15404 |                     # Update status awal (menggunakan keyword argument agar aman)
15405 |                     self.after(0, lambda lbl=target_label: lbl.configure(text="Mengambil data satelit...", text_color="#00E5FF"))
15406 |
15407 |                     url = f"https://api.open-meteo.com/v1/forecast?latitude={lat}&longitude={lon}&current=cloud_cover&timezone=auto"
15408 |                     res = requests.get(url, timeout=5)
15409 |
15410 |                     if res.status_code == 200:
15411 |                         cc = res.json()['current']['cloud_cover']
15412 |                         if cc > 80:
15413 |                             teks, warna = f"☁️ AWAN {cc}% - Cuaca Buruk!", "#FF5252"
15414 |                         elif cc > 50:
15415 |                             teks, warna = f"☁️ AWAN {cc}% - Ufuk Berawan", "#FFD54F"
15416 |                         else:
15417 |                             teks, warna = f"☁️ CERAH: Awan {cc}% - Aman", "#00E676"
15418 |
15419 |                         # Mencegah Bug Closure dengan binding parameter ke fungsi lambda (t=teks, w=warna)
15420 |                         self.after(0, lambda lbl=target_label, t=teks, w=warna: lbl.configure(text=t, text_color=w))
15421 |                     else:
15422 |                         self.after(0, lambda lbl=target_label: lbl.configure(text="❌ Gagal akses API cuaca", text_color="#9E9E9E"))
15423 |                 except Exception as e:
15424 |                     self.after(0, lambda lbl=target_label: lbl.configure(text="❌ Data Cuaca Offline", text_color="#9E9E9E"))
15425 |
15426 |             import threading
15427 |             threading.Thread(target=task, daemon=True).start()
15428 |
15429 |         # =====
15430 |         # MODUL 31: KALKULATOR ASTROFOTOGRAFI & KONTRAS
15431 |         # =====
15432 |         def calculate_astrofotografi(self):
15433 |             try:
15434 |                 import numpy as np
15435 |                 from skyfield import almanac
15436 |                 from skyfield.api import wgs84
15437 |                 import datetime
15438 |
15439 |                 y = int(self.entry_astro_year.get())
15440 |                 m = int(self.entry_astro_month.get())
15441 |                 d = int(self.entry_astro_day.get())
15442 |                 lat = float(self.entry_astro_lat.get())
15443 |                 lon = float(self.entry_astro_lon.get())
15444 |                 tz = float(self.entry_astro_tz.get())
15445 |
15446 |                 # =====
15447 |                 # TAMBAHAN API CUACA (SAMA PERSIS DENGAN MENU 1)
15448 |                 # =====
15449 |                 try:
15450 |                     # 1. Tampilkan status loading di Sidebar & Layar Utama
15451 |                     self.after(0, lambda: self.lbl_cuaca_astro.configure(text="Mengambil data cuaca...", text_color="#00E5FF"))
15452 |                     self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...", text_color="#00E5FF"))
15453 |

```

```

15454 |         # 2. Panggil fungsi cuaca bawaan aplikasi
15455 |         status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)
15456 |
15457 |         # 3. Update hasilnya ke UI
15458 |         self.after(0, lambda: self.lbl_cuaca_astro.configure(text=status_cuaca, text_color=warna_cuaca))
15459 |         self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca, text_color=warna_cuaca))
15460 |     except Exception:
15461 |         self.after(0, lambda: self.lbl_cuaca_astro.configure(text="❌ Gagal memuat cuaca", text_color="#9E9E9E"))
15462 |         self.after(0, lambda: self.lbl_status_cuaca.configure(text="❌ Gagal memuat cuaca", text_color="#9E9E9E"))
15463 |     # =====
15464 |
15465 |     self.auto_switch_ephemeris(y)
15466 |
15467 |     earth = self.eph['earth']
15468 |     sun = self.eph['sun']
15469 |     moon = self.eph['moon']
15470 |
15471 |     loc = wgs84.latlon(lat, lon)
15472 |     topos_obs = earth + loc
15473 |
15474 |     t0 = self.ts.utc(y, m, d, -int(tz))
15475 |     t1 = self.ts.utc(y, m, d, 24 - int(tz))
15476 |     f_rs = almanac.sunrise_sunset(self.eph, loc)
15477 |     t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)
15478 |
15479 |     tt_vals = np.atleast_1d(t_rs.tt)
15480 |     y_vals = np.atleast_1d(y_rs)
15481 |
15482 |     t_sunset = None
15483 |     for tt_val, y_ev in zip(tt_vals, y_vals):
15484 |         if y_ev == 0:
15485 |             t_sunset = self.ts.tt(jd=tt_val)
15486 |             break
15487 |
15488 |     if t_sunset is None:
15489 |         self.after(0, self.display_error, "Matahari tidak terbenam pada tanggal ini di lokasi tersebut.")
15490 |         return
15491 |
15492 |     app_moon = topos_obs.at(t_sunset).observe(moon).apparent()
15493 |     app_sun = topos_obs.at(t_sunset).observe(sun).apparent()
15494 |
15495 |     alt_moon, az_moon, _ = app_moon.altaz()
15496 |     alt_sun, az_sun, _ = app_sun.altaz()
15497 |
15498 |     elongation = app_sun.separation_from(app_moon).degrees
15499 |
15500 |     if elongation < 7.0:
15501 |         danjon_status = f"❌ GAGAL: Elongasi {elongation:.2f}° (< 7°).\n Cahaya sabit belum terbentuk (tertutup bayang-bayang kawah bulan)."
```



```

15525 |         self.after(0, self.display_result, report)
15526 |
15527 |
15528 |     except Exception as e:
15529 |         import traceback
15530 |         self.after(0, self.display_error, f"Error:\n{str(e)}\n\n{traceback.format_exc()}")
15531 |
15532 | # =====
15533 | # MODUL 32: PREDIKTOR PASANG SURUT GRAVITASI
15534 | # =====
15535 | def calculate_pasang_surut(self):
15536 |     try:
15537 |         import datetime
15538 |         import math
15539 |         from skyfield.api import wgs84
15540 |
15541 |         y = int(self.entry_pasang_year.get())
15542 |         m = int(self.entry_pasang_month.get())
15543 |         d = int(self.entry_pasang_day.get())
15544 |         lat = float(self.entry_pasang_lat.get())
15545 |         lon = float(self.entry_pasang_lon.get())
15546 |         tz = float(self.entry_pasang_tz.get())
15547 |
15548 |         # =====
15549 |         # TAMBAHAN API CUACA (SAMA PERSIS DENGAN MENU 1)
15550 |         # =====
15551 |         try:
15552 |             # 1. Tampilkan status loading di Sidebar & Layar Utama
15553 |             self.after(0, lambda: self.lbl_cuaca_pasang.configure(text="Mengambil data cuaca...", text_color="#00E5FF"))
15554 |             self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...", text_color="#00E5FF"))
15555 |
15556 |             # 2. Panggil fungsi cuaca bawaan aplikasi
15557 |             status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)
15558 |
15559 |             # 3. Update hasilnya ke UI
15560 |             self.after(0, lambda: self.lbl_cuaca_pasang.configure(text=status_cuaca, text_color=warna_cuaca))
15561 |             self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca, text_color=warna_cuaca))
15562 |         except Exception:
15563 |             self.after(0, lambda: self.lbl_cuaca_pasang.configure(text="❌ Gagal memuat cuaca", text_color="#9E9E9E"))
15564 |             self.after(0, lambda: self.lbl_status_cuaca.configure(text="❌ Gagal memuat cuaca", text_color="#9E9E9E"))
15565 |         # =====
15566 |
15567 |         self.auto_switch_ephemeris(y)
15568 |
15569 |         earth = self.eph['earth']
15570 |         sun = self.eph['sun']
15571 |         moon = self.eph['moon']
15572 |
15573 |         t_eval = self.ts.utc(y, m, d, 12 - int(tz))
15574 |         loc = wgs84.latlon(lat, lon)
15575 |         topos_obs = earth + loc
15576 |
15577 |         obs_moon = topos_obs.at(t_eval).observe(moon)
15578 |         obs_sun = topos_obs.at(t_eval).observe(sun)
15579 |
15580 |         app_moon = obs_moon.apparent()
15581 |         app_sun = obs_sun.apparent()
15582 |
15583 |         dist_moon_km = app_moon.distance().km
15584 |         dist_sun_km = app_sun.distance().km
15585 |         elongation = app_sun.separation_from(app_moon).degrees
15586 |
15587 |         if elongation < 25 or elongation > 335:
15588 |             fase = "New Moon (Ijtima / Bulan Baru)"
15589 |             jenis_pasang = "❑ PASANG PURNAMA (Spring Tide)"
15590 |             desc_pasang = "Gaya tarik Bulan dan Matahari SEARAH. Potensi pasang laut MAKSIMAL."
15591 |         elif 155 < elongation < 205:
15592 |             fase = "Full Moon (Bulan Purnama)"
15593 |             jenis_pasang = "❑ PASANG PURNAMA (Spring Tide)"
15594 |             desc_pasang = "Gaya tarik Bulan dan Matahari SEJAJAR. Potensi pasang laut MAKSIMAL."
15595 |         elif 65 < elongation < 115 or 245 < elongation < 295:

```

```

15596 |         fase = "Kuartir (Bulan Paruh)"
15597 |         jenis_pasang = "☐ PASANG PERBANI (Neap Tide)"
15598 |         desc_pasang = "Gaya tarik Bulan dan Matahari TEGAK LURUS. Kondisi pasang laut AMAN."
15599 |     else:
15600 |         fase = "Fase Transisi (Crescent/Gibbous)"
15601 |         jenis_pasang = "☐ PASANG MENENGAH (Normal Tide)"
15602 |         desc_pasang = "Kondisi pasang laut berada di batas normal."
15603 |
15604 |     if dist_moon_km < 365000:
15605 |         status_jarak = f"PERIGEE (Sangat Dekat: {dist_moon_km:,.0f} km)"
15606 |         if "PURNAMA" in jenis_pasang:
15607 |             desc_pasang += "\n    Δ☐ BAHAYA: Terjadi KING TIDE (Pasang Perigeon)!\\n    Waspada banjir rob dan ombak ekstrem di pesisir pantai."
15608 |     elif dist_moon_km > 404000:
15609 |         status_jarak = f"APOGEE (Sangat Jauh: {dist_moon_km:,.0f} km)"
15610 |     else:
15611 |         status_jarak = f"Normal ({dist_moon_km:,.0f} km)"
15612 |
15613 |     moon_tide_index = (384400 / dist_moon_km)**3 * 100
15614 |
15615 |     report = f"{self.get_header(80)}\\n\\n"
15616 |     report += f"{'[ Prediktor Pasang Surut Gravitasi Pantai (Tidal Force) ]'.center(80)}\\n\\n"
15617 |     report += f"Tanggal Analisis : {d:02d}-{m:02d}-{y} (Tinjauan Siang Lokal)\\n"
15618 |     report += f"Koordinat Pantai : Lintang {lat}, Bujur {lon}\\n"
15619 |     report += "- " * 80 + "\\n"
15620 |
15621 |     report += f"A. DATA ASTRONOMI FISIK\\n"
15622 |     report += f"    - Elongasi Bulan-Matahari : {elongation:.2f}° ({fase})\\n"
15623 |     report += f"    - Jarak Bumi-Bulan : {status_jarak}\\n"
15624 |     report += f"    - Jarak Bumi-Matahari : {dist_sun_km:,.0f} km\\n"
15625 |     report += f"    - Tarikan Gravitasi Bulan : {moon_tide_index:.1f}% (vs Rata-rata)\\n\\n"
15626 |
15627 |     report += f"B. STATUS KEAMANAN PESISIR & RUKYAT\\n"
15628 |     report += f"    - Tipe Pasang Laut : {jenis_pasang}\\n"
15629 |     report += f"    - Deskripsi Dampak : {desc_pasang}\\n\\n"
15630 |
15631 |     report += "Catatan: Prediksi ini bersifat astronomis murni (Efek Gravitasi Ekstraterestrial).\\n"
15632 |     report += "Waktu aktual puncak pasang di pesisir juga bergantung pada angin laut/topografi.\\n"
15633 |
15634 |     self.after(0, self.display_result, report)
15635 |
15636 | except Exception as e:
15637 |     import traceback
15638 |     self.after(0, self.display_error, f"Error:\\n{str(e)}\\n\\n{traceback.format_exc()}")
15639 |
15640 | # =====
15641 | # MODUL 33: EVALUASI FAJAR SHADIQ & SQM TEORITIS
15642 | # =====
15643 | def calculate_fajar_sqm(self):
15644 |     try:
15645 |         import numpy as np
15646 |         from skyfield import almanac
15647 |         from skyfield.api import wgs84
15648 |         import datetime
15649 |
15650 |         y = int(self.entry_fajar_year.get())
15651 |         m = int(self.entry_fajar_month.get())
15652 |         d = int(self.entry_fajar_day.get())
15653 |         lat = float(self.entry_fajar_lat.get())
15654 |         lon = float(self.entry_fajar_lon.get())
15655 |         tz = float(self.entry_fajar_tz.get())
15656 |
15657 |         # =====
15658 |         # INTEGRASI API CUACA
15659 |         # =====
15660 |         try:
15661 |             self.after(0, lambda: self.lbl_cuaca_fajar.configure(text="Mengambil data cuaca...", text_color="#00E5FF"))
15662 |             self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...", text_color="#00E5FF"))
15663 |
15664 |             status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)
15665 |
15666 |             self.after(0, lambda: self.lbl_cuaca_fajar.configure(text=status_cuaca, text_color=warna_cuaca))

```

```

15667 |         self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca, text_color=warna_cuaca))
15668 |     except Exception:
15669 |         self.after(0, lambda: self.lbl_cuaca_fajar.configure(text="❌ Gagal memuat cuaca", text_color="#9E9E9E"))
15670 |         self.after(0, lambda: self.lbl_status_cuaca.configure(text="❌ Gagal memuat cuaca", text_color="#9E9E9E"))
15671 |     # =====
15672 |
15673 |     self.auto_switch_ephemeris(y)
15674 |
15675 |     earth = self.eph['earth']
15676 |     sun = self.eph['sun']
15677 |
15678 |     loc = wgs84.latlon(lat, lon)
15679 |     topos_obs = earth + loc
15680 |
15681 |     # Rentang waktu pencarian: 00:00 s.d 24:00 waktu lokal
15682 |     t0 = self.ts.utc(y, m, d, -int(tz))
15683 |     t1 = self.ts.utc(y, m, d, 24 - int(tz))
15684 |
15685 |     # Fungsi Kustom Skyfield untuk melacak ketinggian spesifik (-18° dan -20°)
15686 |     def f_angle(angle_deg):
15687 |         def _sun_alt(t):
15688 |             alt, az, distance = topos_obs.at(t).observe(sun).apparent().altaz()
15689 |             return alt.degrees >= angle_deg
15690 |             _sun_alt.step_days = 0.04
15691 |         return _sun_alt
15692 |
15693 |     f_18 = f_angle(-18.0)
15694 |     f_20 = f_angle(-20.0)
15695 |
15696 |     # Mencari event (1 = Matahari naik melewati batas, 0 = Matahari turun melewati batas)
15697 |     t_18_ev, y_18_ev = almanac.find_discrete(t0, t1, f_18)
15698 |     t_20_ev, y_20_ev = almanac.find_discrete(t0, t1, f_20)
15699 |
15700 |     fajar_18 = None
15701 |     fajar_20 = None
15702 |     isya_18 = None
15703 |
15704 |     # Anti-Bug Single Time dengan np.atleast_1d
15705 |     for t_tt, y_val in zip(np.atleast_1d(t_18_ev.tt), np.atleast_1d(y_18_ev)):
15706 |         if y_val == 1 and fajar_18 is None: fajar_18 = self.ts.tt(jd=t_tt) # Matahari Naik (Pagi/Fajar)
15707 |         if y_val == 0 and isya_18 is None: isya_18 = self.ts.tt(jd=t_tt) # Matahari Turun (Malam/Isya)
15708 |
15709 |     for t_tt, y_val in zip(np.atleast_1d(t_20_ev.tt), np.atleast_1d(y_20_ev)):
15710 |         if y_val == 1 and fajar_20 is None: fajar_20 = self.ts.tt(jd=t_tt)
15711 |
15712 |     def format_time(t_obj):
15713 |         if t_obj is None: return "Tidak Terjadi/Extrem"
15714 |         dt = t_obj.utc_datetime() + datetime.timedelta(hours=tz)
15715 |         return dt.strftime('%H:%M:%S')
15716 |
15717 |     # Format Laporan
15718 |     report = f"{self.get_header(80)}\n"
15719 |     report += f"{'[ Evaluasi Fajar Shadiq & SQM Teoritis ]'.center(80)}\n\n"
15720 |     report += f"Tanggal Observasi : {d:02d}-{m:02d}-{y}\n"
15721 |     report += f"Koordinat Lokasi : Lat {lat}, Lon {lon}\n"
15722 |     report += "-" * 80 + "\n\n"
15723 |
15724 |     report += "A. KOMPARASI WAKTU FAJAR SHADIQ (SUBUH)\n"
15725 |     report += f"    - Kemenag RI (-20°) : {format_time(fajar_20)} Waktu Lokal\n"
15726 |     report += f"    - Muhammadiyah/MABIMS (-18°): {format_time(fajar_18)} Waktu Lokal\n"
15727 |
15728 |     # ---> PERBAIKAN DI SINI: Gunakan 'is not None' <---
15729 |     if fajar_18 is not None and fajar_20 is not None:
15730 |         selisih_fajar = round(abs((fajar_18.tt - fajar_20.tt) * 24 * 60))
15731 |         report += f"    * Selisih waktu subuh : ~{selisih_fajar} menit\n\n"
15732 |     else:
15733 |         report += f"    * Selisih waktu subuh : Tidak dapat dihitung\n\n"
15734 |
15735 |     report += "B. WAKTU ISYA (-18°)\n"
15736 |     report += f"    - Awal Waktu Isya : {format_time(isy_18)} Waktu Lokal\n\n"
15737 |

```

```

15738 |         report += "C. PREDIKSI NILAI SQM (Sky Quality Meter) TEORITIS\n"
15739 |         report += "    - Saat Matahari -18° : ~21.0 - 21.5 mag/arcsec² (Awal Fajar Astronomis)\n"
15740 |         report += "    - Saat Matahari -15° : ~19.0 - 19.5 mag/arcsec² (Langit mulai teramati biru gelap)\n"
15741 |         report += "    - Saat Matahari -12° : ~17.0 - 17.5 mag/arcsec² (Batas Nautical Twilight)\n"
15742 |         report += "    - Saat Matahari -6° : ~13.0 - 14.0 mag/arcsec² (Batas Civil Twilight)\n\n"
15743 |
15744 |         report += "KESIMPULAN & CATATAN PENGUKURAN SQM:\n"
15745 |         report += "1. Pengukuran dengan alat SQM fisik sangat dipengaruhi oleh polusi cahaya kota.\n"
15746 |         report += "2. Angka referensi di atas berlaku HANYA untuk lokasi pengamatan yang\n"
15747 |         report += "    benar-benar gelap (Bortle Class 1-2) tanpa gangguan cahaya bulan purnama.\n"
15748 |         report += "3. Di perkotaan, nilai SQM sering kali tertahan di angka 17 - 18 karena polusi.\n"
15749 |
15750 |         self.after(0, self.display_result, report)
15751 |
15752 |     except Exception as e:
15753 |         import traceback
15754 |         self.after(0, self.display_error, f"Error:\n{str(e)}\n\n{traceback.format_exc()}")
15755 |
15756 | # =====
15757 | # MODUL 34: GENERATOR ANALEMMA MATAHARI
15758 | # =====
15759 | def calculate_analemma(self):
15760 |     try:
15761 |         import numpy as np
15762 |         from skyfield.api import wgs84
15763 |         import datetime
15764 |
15765 |         # Ambil data dari input sidebar
15766 |         y = int(self.entry_analemma_year.get())
15767 |         h_local = float(self.entry_analemma_hour.get())
15768 |         lat = float(self.entry_analemma_lat.get())
15769 |         lon = float(self.entry_analemma_lon.get())
15770 |         tz = float(self.entry_analemma_tz.get())
15771 |
15772 |         # =====
15773 |         # INTEGRASI API CUACA (UPDATE INSTAN)
15774 |         # =====
15775 |         try:
15776 |             self.after(0, lambda: self.lbl_cuaca_analemma.configure(text="Mengambil data cuaca...", text_color="#00E5FF"))
15777 |             self.after(0, lambda: self.lbl_status_cuaca.configure(text="Mengambil data satelit cuaca...", text_color="#00E5FF"))
15778 |
15779 |             status_cuaca, warna_cuaca = get_live_weather_status(lat, lon)
15780 |
15781 |             self.after(0, lambda: self.lbl_cuaca_analemma.configure(text=status_cuaca, text_color=warna_cuaca))
15782 |             self.after(0, lambda: self.lbl_status_cuaca.configure(text=status_cuaca, text_color=warna_cuaca))
15783 |         except Exception:
15784 |             self.after(0, lambda: self.lbl_cuaca_analemma.configure(text="❌ Gagal memuat cuaca", text_color="#9E9E9E"))
15785 |         # =====
15786 |
15787 |         # --- LOGIKA ASTRONOMI ANALEMMA ---
15788 |         self.auto_switch_ephemeris(y)
15789 |         earth = self.eph['earth']
15790 |         sun = self.eph['sun']
15791 |         loc = earth + wgs84.latlon(lat, lon)
15792 |
15793 |         alts, azs, dates = [], [], []
15794 |
15795 |         # Memproses lintasan setahun penuh (setiap 5 hari)
15796 |         for day in range(0, 365, 5):
15797 |             date_dt = datetime.datetime(y, 1, 1) + datetime.timedelta(days=day)
15798 |             t = self.ts.utc(y, date_dt.month, date_dt.day, h_local - tz)
15799 |
15800 |             astrometric = loc.at(t).observe(sun).apparent()
15801 |             alt, az, _ = astrometric.altaz()
15802 |
15803 |             alts.append(alt.degrees)
15804 |             azs.append(az.degrees)
15805 |             dates.append(date_dt)
15806 |
15807 |         idx_max = np.argmax(alts)
15808 |         idx_min = np.argmin(alts)

```

```

15809 |
15810 |         report = f"{self.get_header(80)}\n"
15811 |         report += f"{'[ ANALISIS LINTASAN ANALEMMA MATAHARI ]'.center(80)}\n"
15812 |         report += f"{'Pengamatan Tahunan pada Jam {h_local:02.0f}:00 Lokal'.center(80)}\n"
15813 |         report += "-" * 80 + "\n\n"
15814 |
15815 |         report += f"A. TITIK EKSTRIM ANALEMMA (TAHUN {y})\n"
15816 |         report += f"    1. Posisi Tertinggi : {dates[idx_max].strftime('%d %B')} -> Alt: {alts[idx_max]:.2f}°\n"
15817 |         report += f"    2. Posisi Terendah : {dates[idx_min].strftime('%d %B')} -> Alt: {alts[idx_min]:.2f}°\n\n"
15818 |
15819 |         report += "B. DATA SAMPEL LINTASAN BULANAN\n"
15820 |         report += f"{'Tanggal':<20} | {'Ketinggian (Alt)':<18} | {'Arah (Azimuth)':<15}\n"
15821 |         report += "-" * 60 + "\n"
15822 |         for i in range(0, len(dates), 6):
15823 |             report += f"{dates[i].strftime('%d %b %Y'):<20} | {alts[i]:>15.2f}° | {azs[i]:>12.2f}°\n"
15824 |
15825 |         report += "\nAnalemma membentuk angka 8 karena orbit Bumi yang elips dan kemiringan sumbu rotasi.\n"
15826 |         report += "Hal ini membuktikan perbedaan antara Waktu Matahari Sejati dan Waktu Rata-rata."
15827 |
15828 |         # Lempar ke GUI (Main Thread) agar merender teks beserta grafiknya
15829 |         self.after(0, self.display_analemma_result, report, azs, alts, y)
15830 |
15831 |     except Exception as e:
15832 |         import traceback
15833 |         self.after(0, self.display_error, f"Gagal menghitung Analemma:\n{str(e)}\n\n{traceback.format_exc()}")
15834 |
15835 | def setup_analemma_out_frame(self):
15836 |     """Mempersiapkan area output scrollable untuk teks dan grafik Analemma Menu 34"""
15837 |     self.frame_analemma_out = ctk.CTkScrollableFrame(self.main_frame, fg_color="#050510", corner_radius=10)
15838 |
15839 |     # Kotak Teks untuk Laporan
15840 |     self.txt_analemma_report = ctk.CTkTextbox(self.frame_analemma_out, height=350, font=("Consolas", 13), fg_color="#101010", text_color="#00E5FF", wrap="none")
15841 |     self.txt_analemma_report.pack(fill="x", padx=15, pady=(15, 5))
15842 |
15843 |     # Frame untuk menampung Grafik Matplotlib
15844 |     self.frame_analemma_canvas = ctk.CTkFrame(self.frame_analemma_out, fg_color="#181818")
15845 |     self.frame_analemma_canvas.pack(fill="both", expand=True, padx=15, pady=(5, 15))
15846 |
15847 | def display_analemma_result(self, report_text, azs, alts, year):
15848 |     """Memasukkan teks ke textbox dan menggambar kurva Analemma"""
15849 |     # 1. Update Textbox
15850 |     self.txt_analemma_report.configure(state="normal")
15851 |     self.txt_analemma_report.delete("1.0", "end")
15852 |     self.txt_analemma_report.insert("1.0", report_text)
15853 |     self.txt_analemma_report.configure(state="disabled")
15854 |
15855 |     # 2. Render Grafik Matplotlib
15856 |     for widget in self.frame_analemma_canvas.winfo_children():
15857 |         widget.destroy()
15858 |
15859 |     from matplotlib.figure import Figure
15860 |     from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
15861 |     import numpy as np
15862 |
15863 |     fig = Figure(figsize=(8, 6), dpi=100)
15864 |     fig.patch.set_facecolor('#181818')
15865 |     ax = fig.add_subplot(111)
15866 |     ax.set_facecolor('#101010')
15867 |
15868 |     # Plot garis Analemma (angka 8)
15869 |     ax.plot(azs, alts, color='#FFD54F', marker='o', markersize=4, linestyle='-', linewidth=1.5, label='Jalur Analemma')
15870 |
15871 |     # Tandai posisi tertinggi dan terendah
15872 |     idx_max = np.argmax(alts)
15873 |     idx_min = np.argmin(alts)
15874 |     ax.plot(azs[idx_max], alts[idx_max], 'r^', markersize=10, label='Tertinggi (Summer Solstice)')
15875 |     ax.plot(azs[idx_min], alts[idx_min], 'bv', markersize=10, label='Terendah (Winter Solstice)')
15876 |
15877 |     ax.set_title(f"Kurva Analemma Matahari (Tahun {year})", color='white', fontsize=14, fontweight='bold', pad=15)
15878 |     ax.set_xlabel("Azimuth Matahari (°)", color='ffffff', fontsize=12)
15879 |     ax.set_ylabel("Ketinggian / Altitude (°)", color='ffffff', fontsize=12)

```

```

15880 |         ax.tick_params(colors='#ffffff', labelsz=11)
15881 |         ax.grid(True, color='#ffffff', linestyle=':', alpha=0.3)
15882 |         ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white')
15883 |         fig.tight_layout()
15884 |
15885 |         canvas = FigureCanvasTkAgg(fig, master=self.frame_analemma_canvas)
15886 |         canvas.draw()
15887 |         canvas.get_tk_widget().pack(fill="both", expand=True, pady=10)
15888 |
15889 |         # Selesaikan Status
15890 |         self.lbl_status.configure(text="Kalkulasi & Grafik Analemma Selesai", text_color="#00E676")
15891 |         self.btn_hitung.configure(state="normal")
15892 |
15893 |     def calculate_occultation(self):
15894 |         """Memicu pencarian okultasi di background thread agar aplikasi tidak macet"""
15895 |         threading.Thread(target=self._proses_okultasi, daemon=True).start()
15896 |
15897 |     def _proses_okultasi(self):
15898 |         self.after(0, lambda: self.lbl_status.configure(text="Memindai Okultasi Bulan & Planet...", text_color="#00E5FF"))
15899 |         self.after(0, lambda: self.btn_hitung.configure(state="disabled"))
15900 |         self.after(0, lambda: self.btn_cari_occ.configure(state="disabled"))
15901 |         self.after(0, lambda: self.textbox.configure(state="normal", wrap="none"))
15902 |         self.after(0, lambda: self.textbox.delete("1.0", "end"))
15903 |         self.after(0, lambda: self.textbox.insert("1.0", "Menganalisis lintasan Bulan dan Target Planet dalam sebulan penuh...\nMohon tunggu...\n\n"))
15904 |
15905 |         try:
15906 |             import numpy as np
15907 |             from skyfield import almanac
15908 |             from skyfield.api import wgs84
15909 |             import math
15910 |             import datetime
15911 |
15912 |             # Ambil data input Astrofotografi
15913 |             y = int(self.entry_astro_year.get())
15914 |             m = int(self.entry_astro_month.get())
15915 |             lat = float(self.entry_astro_lat.get())
15916 |             lon = float(self.entry_astro_lon.get())
15917 |             tz = float(self.entry_astro_tz.get())
15918 |             elev = float(self.entry_astro_elev.get())
15919 |             target_name = self.combo_astro_occ_target.get().lower()
15920 |
15921 |             self.auto_switch_ephemeris(y)
15922 |             earth = self.eph['earth']
15923 |             moon = self.eph['moon']
15924 |
15925 |             # Skyfield memakai kata "barycenter" untuk tata surya besar
15926 |             target_key = f'{target_name} barycenter' if target_name not in ['mercury', 'venus'] else target_name
15927 |             try:
15928 |                 planet = self.eph[target_key]
15929 |             except KeyError:
15930 |                 planet = self.eph[target_name]
15931 |
15932 |             loc = wgs84.latlon(lat, lon, elevation_m=elev)
15933 |
15934 |             # Rentang pencarian: 1 bulan penuh
15935 |             _, num_days = safe_moonrange(y, m)
15936 |             t0 = self.ts.utc(y, m, 1, -int(tz))
15937 |             t1 = self.ts.utc(y, m, num_days, 24 - int(tz))
15938 |
15939 |             # FUNGSI DETEKTOR OKULTASI
15940 |             # Menghitung jika jarak pisah (separasi) lebih kecil dari gabungan Radius Bulan + Radius Planet
15941 |             def occultation_func(t):
15942 |                 obs = (earth + loc).at(t)
15943 |                 m_app = obs.observe(moon).apparent()
15944 |                 p_app = obs.observe(planet).apparent()
15945 |                 sep = m_app.separation_from(p_app).degrees
15946 |
15947 |                 m_dist = m_app.distance().km
15948 |                 p_dist = p_app.distance().km
15949 |
15950 |             # Kalkulasi Radius Semu di Langit

```

```

15951 |         m_rad = np.degrees(np.arcsin(1737.4 / m_dist))
15952 |         radii = {'mercury': 2439.7, 'venus': 6051.8, 'mars': 3389.5, 'jupiter': 69911.0, 'saturn': 58232.0, 'uranus': 25362.0, 'neptune': 24622.0}
15953 |         p_rad = np.degrees(np.arcsin(radii[target_name] / p_dist))
15954 |
15955 |         return sep < (m_rad + p_rad)
15956 |
15957 | # Resolusi pencarian ~7 menit agar sangat akurat mendeteksi masuk/keluar planet
15958 | occultation_func.step_days = 0.005
15959 |
15960 | # Lacak menggunakan Almanak
15961 | t_occ, y_occ = almanac.find_discrete(t0, t1, occultation_func)
15962 |
15963 | output_lines = []
15964 | output_lines.append(self.get_header(90))
15965 | output_lines.append(f"[ PREDIKTOR OKULTASI: BULAN MENUTUPI {target_name.upper()} ]".center(90))
15966 | output_lines.append("=" * 90)
15967 |
15968 | try:
15969 |     nama_kota = self.opt_city.get()
15970 |     nama_prov = self.opt_prov.get()
15971 |     lokasi_text = f"{nama_kota}, {nama_prov}"
15972 | except:
15973 |     lokasi_text = f"Lat {lat}, Lon {lon}"
15974 |
15975 | output_lines.append(f"Lokasi Observasi : {lokasi_text}")
15976 | output_lines.append(f"Rentang Waktu      : Bulan {m:02d} Tahun {format_tahun_aman(y)}")
15977 | output_lines.append(f"Zona Waktu        : UTC{'+' if tz>=0 else ''}{tz}")
15978 | output_lines.append("-" * 90)
15979 |
15980 | if len(t_occ) == 0:
15981 |     output_lines.append(f"\n>> TIDAK ADA okultasi {target_name.capitalize()} oleh Bulan yang teramati")
15982 |     output_lines.append(f"      dari lokasi Anda pada bulan {m:02d}/{format_tahun_aman(y)}.")
15983 | else:
15984 |     # Group Ingress (Masuk) dan Egress (Keluar)
15985 |     events = []
15986 |     current_ingress = None
15987 |
15988 |     for t_ev, state in zip(t_occ, y_occ):
15989 |         if state == 1: # Ingress (Mulai Tertutup / Kontak 1)
15990 |             current_ingress = t_ev
15991 |         elif state == 0 and current_ingress is not None: # Egress (Mulai Muncul Kembali / Kontak 2)
15992 |             events.append((current_ingress, t_ev))
15993 |             current_ingress = None
15994 |
15995 |     if current_ingress is not None:
15996 |         # Okultasi terjadi dan menyeberang ke bulan berikutnya
15997 |         events.append((current_ingress, None))
15998 |
15999 |     for idx, ev in enumerate(events, 1):
16000 |         t_in, t_eg = ev
16001 |
16002 |         # Cek Altitude & Iluminasi saat Ingress
16003 |         obs_in = (earth + loc).at(t_in).observe(moon).apparent()
16004 |         alt_in = obs_in.altaz()[0].degrees
16005 |
16006 |         dt_in = t_in.utcdatetime() + datetime.timedelta(hours=tz)
16007 |         str_in = dt_in.strftime("%d-%b-%Y %H:%M:%S")
16008 |
16009 |         if t_eg is not None:
16010 |             obs_eg = (earth + loc).at(t_eg).observe(moon).apparent()
16011 |             alt_eg = obs_eg.altaz()[0].degrees
16012 |             dt_eg = t_eg.utcdatetime() + datetime.timedelta(hours=tz)
16013 |             str_eg = dt_eg.strftime("%d-%b-%Y %H:%M:%S")
16014 |             alt_eg_str = f"{alt_eg:.2f}°"
16015 |             durasi = round((t_eg.tt - t_in.tt) * 24 * 60, 1)
16016 |             durasi_str = f"{durasi} Menit"
16017 |         else:
16018 |             str_eg = "Terpotong batas bulan"
16019 |             alt_eg_str = "N/A"
16020 |             durasi_str = "N/A"
16021 |

```

```

16022 |         # Iluminasi Bulan (Memudahkan kamera mengatur ISO/Shutter Speed)
16023 |         illum = almanac.fraction_illuminated(self.eph, 'moon', t_in)
16024 |         illum_val = (illum.item() if hasattr(illum, 'item') else illum) * 100.0
16025 |
16026 |         output_lines.append(f"\n[ PERISTIWA OKULTASI #{idx} ]")
16027 |         output_lines.append(f" FASE INGRESS (Mulai Tertutup) : {str_in} | Alt Bulan: {alt_in:.2f}°")
16028 |         output_lines.append(f" FASE EGRESS (Muncul Kembali) : {str_eg} | Alt Bulan: {alt_eg_str}")
16029 |         output_lines.append(f" Durasi Tertutup Penuh : {durasi_str}")
16030 |         output_lines.append(f" Kondisi Iluminasi Bulan : {illum_val:.1f}%")
16031 |
16032 |         if alt_in < 0:
16033 |             output_lines.append(f" ⚠ PERINGATAN: Terjadi saat Bulan berada di BAWAH UFUK (Tidak bisa dirukyt secara fisik).")
16034 |
16035 |         output_lines.append("\n" + "=" * 90)
16036 |         output_lines.append("* Catatan Validasi:")
16037 |         output_lines.append("- Prediktor ini memperhitungkan semi-diameter (radius) Bulan dan Planet secara dinamis")
16038 |         output_lines.append(" berdasarkan jarak real-time dari bumi.")
16039 |         output_lines.append("- Waktu sangat presisi dan disesuaikan dengan titik Toposentrik lokasi GPS.")
16040 |
16041 |         final_report = "\n".join(output_lines)
16042 |         self.after(0, lambda: self.display_result(final_report))
16043 |
16044 |     except Exception as e:
16045 |         import traceback
16046 |         self.after(0, lambda: self.display_error(f"Gagal memprediksi Okultasi:\n{str(e)}\n\n{traceback.format_exc()}"))
16047 |     finally:
16048 |         self.after(0, lambda: self.btn_hitung.configure(state="normal"))
16049 |         self.after(0, lambda: self.btn_cari_occ.configure(state="normal"))
16050 |
16051 | # =====
16052 | # MODUL EKSKLUSIF: KALKULATOR SHALAT PENERBANGAN (IN-FLIGHT PRAYER TIMES)
16053 | # =====
16054 | def buka_kalkulator_penerbangan(self):
16055 |     self.win_flight = ctk.CTkToplevel(self)
16056 |     self.win_flight.title("✈ Kalkulator Waktu Shalat Penerbangan (In-Flight)")
16057 |     self.win_flight.geometry("850x850")
16058 |     self.win_flight.attributes("-topmost", True)
16059 |     self.win_flight.configure(fg_color="#0A0A10")
16060 |
16061 |     ctk.CTkLabel(self.win_flight, text="AEROSPACE PRAYER TIMES CALCULATOR", font=("Segoe UI", 18, "bold"), text_color="#FFD54F").pack(pady=(15, 5))
16062 |     ctk.CTkLabel(self.win_flight, text="Kalkulasi Dinamis berbasis Rute Great Circle, Kecepatan, dan Ketinggian", font=("Segoe UI", 12)).pack(pady=(0, 15))
16063 |
16064 |     # --- FRAME INPUT ---
16065 |     input_container = ctk.CTkFrame(self.win_flight, fg_color="transparent")
16066 |     input_container.pack(fill="x", padx=20)
16067 |
16068 |     # Kiri: Keberangkatan
16069 |     frame_dep = ctk.CTkFrame(input_container, fg_color="#1E1E1E", corner_radius=8)
16070 |     frame_dep.pack(side="left", fill="both", expand=True, padx=(0, 10))
16071 |     ctk.CTkLabel(frame_dep, text="DEPARTURE (KEBERANGKATAN)", font=("Segoe UI", 12, "bold"), text_color="#00E5FF").pack(pady=10)
16072 |     self.fl_dep_lat = self.create_input_row(frame_dep, "Lat Berangkat:", "-6.1256") # Default CGK
16073 |     self.fl_dep_lon = self.create_input_row(frame_dep, "Lon Berangkat:", "106.6558")
16074 |     self.fl_dep_tz = self.create_input_row(frame_dep, "Timezone Awal:", "7.0")
16075 |
16076 |     ctk.CTkLabel(frame_dep, text="Waktu Berangkat (Lokal):", font=("Segoe UI", 11)).pack(anchor="w", padx=10, pady=(10,0))
16077 |     date_frame = ctk.CTkFrame(frame_dep, fg_color="transparent")
16078 |     date_frame.pack(fill="x", padx=10, pady=5)
16079 |     now = datetime.datetime.now()
16080 |     self.fl_y, self.fl_m, self.fl_d = self.create_ymd_row(date_frame, str(now.year), f"{now.month:02d}", f"{now.day:02d}")
16081 |     self.fl_h_m_s = self.create_input_row(date_frame, "Jam (HH:MM):", "08:00")
16082 |
16083 |     # Kanan: Kedatangan & Pesawat
16084 |     frame_arr = ctk.CTkFrame(input_container, fg_color="#1E1E1E", corner_radius=8)
16085 |     frame_arr.pack(side="right", fill="both", expand=True, padx=(10, 0))
16086 |     ctk.CTkLabel(frame_arr, text="DESTINATION & AIRCRAFT", font=("Segoe UI", 12, "bold"), text_color="#00E676").pack(pady=10)
16087 |     self.fl_arr_lat = self.create_input_row(frame_arr, "Lat Tujuan:", "21.5433") # Default JED
16088 |     self.fl_arr_lon = self.create_input_row(frame_arr, "Lon Tujuan:", "39.1728")
16089 |
16090 |     ctk.CTkLabel(frame_arr, text="Spesifikasi Penerbangan:", font=("Segoe UI", 11)).pack(anchor="w", padx=10, pady=(10,0))
16091 |     self.fl_duration = self.create_input_row(frame_arr, "Durasi (Jam):", "9.5")
16092 |     self.fl_alt = self.create_input_row(frame_arr, "Ketinggian (Kaki/ft):", "35000")

```



```

16093 |
16094 |     # Tombol Proses
16095 |     self.btn_calc_flight = ctk.CTkButton(self.win_flight, text="▶ HITUNG JADWAL SELAMA PENERBANGAN", font=("Segoe UI", 14, "bold"),
16096 |                                     fg_color="#D84315", hover_color="#BF360C", height=40, command=self.proses_kalkulator_penerbangan)
16097 |     self.btn_calc_flight.pack(fill="x", padx=20, pady=20)
16098 |
16099 |     # Kotak Laporan
16100 |     self.txt_flight_report = ctk.CTkTextbox(self.win_flight, font=("Consolas", 13), fg_color="#101010", text_color="#E0E0E0", wrap="none")
16101 |     self.txt_flight_report.pack(fill="both", expand=True, padx=20, pady=(0, 20))
16102 |     self.txt_flight_report.insert("1.0", "Menunggu input parameter penerbangan...\nSistem akan melacak posisi pesawat per menit di udara (Great Circle) dan menghitung penurunan ufuk akibat altitude
pesawat (Horizon Dip).")
16103 |     self.txt_flight_report.configure(state="disabled")
16104 |
16105 |     def proses_kalkulator_penerbangan(self):
16106 |         self.btn_calc_flight.configure(state="disabled", text="□ MENGKALKULASI RUTE...")
16107 |         self.txt_flight_report.configure(state="normal")
16108 |         self.txt_flight_report.delete("1.0", "end")
16109 |         threading.Thread(target=self._hitung_penerbangan_thread, daemon=True).start()
16110 |
16111 |     def _slerp(self, lat1, lon1, lat2, lon2, frac):
16112 |         """Spherical Linear Interpolation untuk menghitung posisi pesawat secara presisi di rute melengkung bumi (Great Circle)"""
16113 |         import math
16114 |         l1, lo1, l2, lo2 = map(math.radians, [lat1, lon1, lat2, lon2])
16115 |
16116 |         x1, y1, z1 = math.cos(l1)*math.cos(lo1), math.cos(l1)*math.sin(lo1), math.sin(l1)
16117 |         x2, y2, z2 = math.cos(l2)*math.cos(lo2), math.cos(l2)*math.sin(lo2), math.sin(l2)
16118 |
16119 |         dot = max(-1.0, min(1.0, x1*x2 + y1*y2 + z1*z2))
16120 |         omega = math.acos(dot)
16121 |
16122 |         if omega < 1e-5: return lat1, lon1
16123 |
16124 |         sin_omega = math.sin(omega)
16125 |         a = math.sin((1.0 - frac) * omega) / sin_omega
16126 |         b = math.sin(frac * omega) / sin_omega
16127 |
16128 |         x, y, z = a*x1 + b*x2, a*y1 + b*y2, a*z1 + b*z2
16129 |         return math.degrees(math.asin(z)), math.degrees(math.atan2(y, x))
16130 |
16131 |     def _hitung_penerbangan_thread(self):
16132 |         try:
16133 |             # 1. Ambil Input
16134 |             lat1, lon1 = float(self.fl_dep_lat.get()), float(self.fl_dep_lon.get())
16135 |             lat2, lon2 = float(self.fl_arr_lat.get()), float(self.fl_arr_lon.get())
16136 |             durasi_jam = float(self.fl_duration.get())
16137 |             altitude_ft = float(self.fl_alt.get())
16138 |             tz_awal = float(self.fl_dep_tz.get())
16139 |
16140 |             y, m, d = int(self.fl_y.get()), int(self.fl_m.get()), int(self.fl_d.get())
16141 |             jam_str = self.fl_h_m_s.get().split(":")
16142 |             hr, mn = int(jam_str[0]), int(jam_str[1])
16143 |
16144 |             # Waktu Lokal ke UTC
16145 |             dt_dep_lokal = datetime.datetime(y, m, d, hr, mn, 0)
16146 |             dt_dep_utc = dt_dep_lokal - datetime.timedelta(hours=tz_awal)
16147 |
16148 |             self.auto_switch_ephemeris(y)
16149 |             earth, sun = self.eph['earth'], self.eph['sun']
16150 |
16151 |             # 2. Rumus Dip of Horizon (Penurunan Ufuk di Pesawat)
16152 |             # Pada ketinggian 35.000 kaki, ufuk tidak lagi di 0 derajat, tapi anjlok ke bawah.
16153 |             # Rumus Dip: 1.76 * sqrt(Altitude_ft / 3.2808) / 60
16154 |             altitude_m = altitude_ft / 3.2808
16155 |             dip_deg = (1.76 * math.sqrt(altitude_m)) / 60.0
16156 |
16157 |             # Standar geometri Maghrib/Syuruq: -0.833° (Refraksi standar + Semi-diameter).
16158 |             # Di pesawat, harus ditambah dengan Dip of the Horizon.
16159 |             batas_ufuk_pesawat = -0.833 - dip_deg
16160 |             batas_fajar_isya = -18.0 # Sudut Fajar/Isya standar astronomis
16161 |
16162 |             # 3. Simulasi Perjalanan Menit per Menit

```

```

16163 |         total_menit = int(durasi_jam * 60)
16164 |         timeline_events = []
16165 |
16166 |         t_dep_sky = self.ts.utc(dt_dep_utc.year, dt_dep_utc.month, dt_dep_utc.day, dt_dep_utc.hour, dt_dep_utc.minute)
16167 |
16168 |         # Pelacakan status agar tidak terjadi cetak berulang
16169 |         status_siang = None
16170 |         isya_done, fajar_done = False, False
16171 |
16172 |         # Menyimpan rute dan waktu untuk laporan
16173 |         log_rute = []
16174 |
16175 |         for menit_ke in range(total_menit + 1):
16176 |             frac = menit_ke / total_menit if total_menit > 0 else 0
16177 |             curr_dt_utc = dt_dep_utc + datetime.timedelta(minutes=menit_ke)
16178 |             curr_dt_lokal_asal = dt_dep_lokal + datetime.timedelta(minutes=menit_ke)
16179 |
16180 |             # Interpolasi Posisi Pesawat
16181 |             curr_lat, curr_lon = self._slerp(lat1, lon1, lat2, lon2, frac)
16182 |
16183 |             # Hitung Posisi Matahari dari Koordinat Pesawat Saat Ini
16184 |             t_curr = self.ts.utc(curr_dt_utc.year, curr_dt_utc.month, curr_dt_utc.day, curr_dt_utc.hour, curr_dt_utc.minute)
16185 |             loc = wgs84.latlon(curr_lat, curr_lon, elevation_m=altitude_m)
16186 |
16187 |             # Menghitung ketinggian dan arah kiblat dinamis pesawat
16188 |             app_sun = (earth + loc).at(t_curr).observe(sun).apparent()
16189 |             alt_sun, _, _ = app_sun.altaz(temperature_C=-50, pressure_mbar=250) # Atmosfer tipis di atas
16190 |             alt_deg = alt_sun.degrees
16191 |
16192 |             # Hitung Kiblat Dinamis
16193 |             phi_k, lam_k = math.radians(21.4225), math.radians(39.8262)
16194 |             phi, lam = math.radians(curr_lat), math.radians(curr_lon)
16195 |             y_q = math.sin(lam_k - lam)
16196 |             x_q = math.cos(phi)*math.tan(phi_k) - math.sin(phi)*math.cos(lam_k-lam)
16197 |             qibla_dir = (math.degrees(math.atan2(y_q, x_q)) + 360) % 360
16198 |
16199 |             # Cek Kondisi Siang/Malam berdasarkan Ufuk Pesawat
16200 |             cek_siang = alt_deg > batas_ufuk_pesawat
16201 |
16202 |             # Catat Log setiap jam untuk tabel jalur
16203 |             if menit_ke % 60 == 0 or menit_ke == total_menit:
16204 |                 log_rute.append(f"{curr_dt_lokal_asal.strftime('%H:%M')} | {curr_lat:>7.2f}°, {curr_lon:>8.2f}° | Alt Mth: {alt_deg:>6.2f}° | Arah Kiblat: {qibla_dir:>5.1f}°")
16205 |
16206 |             if menit_ke == 0:
16207 |                 status_siang = cek_siang
16208 |                 continue
16209 |
16210 |             jam_kejadian = curr_dt_lokal_asal.strftime('%H:%M')
16211 |
16212 |             # Logika Penyilangan Event
16213 |             if not status_siang and cek_siang:
16214 |                 timeline_events.append(f"[{jam_kejadian}] □ TERBIT MATAHARI (Syuruq) - Ketinggian: {alt_deg:.2f}°")
16215 |                 status_siang = True
16216 |                 fajar_done = False # Reset fajar
16217 |
16218 |             elif status_siang and not cek_siang:
16219 |                 timeline_events.append(f"[{jam_kejadian}] □ MAGHRIB (Terbenam) - Ketinggian: {alt_deg:.2f}° (Ufuk pesawat di {batas_ufuk_pesawat:.2f}°)")
16220 |                 status_siang = False
16221 |                 isya_done = False # Reset isya
16222 |
16223 |             # Logika Subuh / Isya
16224 |             if alt_deg >= batas_fajar_isya and not cek_siang and not fajar_done and alt_deg > -18.5 and alt_deg < batas_ufuk_pesawat:
16225 |                 # Mendekati fajar (naik dari -18) -> SUBUH
16226 |                 # Pastikan ini sebelum terbit
16227 |                 if alt_deg - (-18.0) > 0:
16228 |                     timeline_events.append(f"[{jam_kejadian}] □ SUBUH (Fajar Shadiq) - Posisi Matahari: {alt_deg:.2f}°")
16229 |                     fajar_done = True
16230 |
16231 |             elif alt_deg <= batas_fajar_isya and not cek_siang and not isya_done:
16232 |                 # Turun melewati -18 -> ISYA
16233 |                 timeline_events.append(f"[{jam_kejadian}] □ ISYA (Syafaq Hilang) - Posisi Matahari: {alt_deg:.2f}°")

```

```

16234 |         isya_done = True
16235 |
16236 |         # 4. Susun Laporan Final
16237 |         report = f""#{self.get_header(90)}
16238 |         {"[ KALKULATOR WAKTU SHALAT PENERBANGAN (AEROSPACE) ]".center(90)}
16239 |
16240 |         Penerbangan jarak jauh (Long-Haul) melintasi garis bujur dengan cepat menyebabkan fenomena
16241 |         waktu relatif. Di ketinggian {altitude_ft:.0f} ft, ufuk mar'i turun sebesar {abs(dip_deg):.2f}°.
16242 |         =====
16243 |
16244 |         [ DATA KEBERANGKATAN (Waktu Acuan Lokal) ]
16245 |         Lokasi Awal : Lat {lat1:.4f}, Lon {lon1:.4f}
16246 |         Take Off    : {dt_dep_lokal.strftime('%d %B %Y - %H:%M')} (Timezone {tz_awal})
16247 |
16248 |         [ KONDISI DI UDARA ]
16249 |         Durasi      : {durasi_jam} Jam
16250 |         Altitude    : {altitude_ft:.0f} ft (Batas Horizon Pesawat: {batas_ufuk_pesawat:.2f}°)
16251 |
16252 |         =====
16253 |         [ GARIS WAKTU SHALAT SELAMA PENERBANGAN ]
16254 |         * Seluruh jam di bawah ini menggunakan Waktu Jam Tangan Anda saat berangkat (Waktu Lokal Asal).
16255 |         * Tidak perlu memutar jam tangan selama di udara.
16256 |         -----
16257 |         """
16258 |         if not timeline_events:
16259 |             report += ">> Pesawat tidak melintasi transisi waktu shalat (Misal: Terbang siang hari penuh).\n"
16260 |         else:
16261 |             for ev in timeline_events:
16262 |                 report += f" {ev}\n"
16263 |
16264 |             report += "\n===== \n"
16265 |             report += "[ LOG JALUR PENERBANGAN & ARAH KIBLAT DINAMIS ]\n"
16266 |             report += "Jam(Asal) | Koordinat Pesawat | Tinggi Mth. | Arah Kiblat (Dari Utara Pesawat)\n"
16267 |             report += "-" * 90 + "\n"
16268 |             for log in log_rute:
16269 |                 report += f" {log}\n"
16270 |
16271 |             report += "\n* Catatan: Arah Kiblat (Qibla) terus bergeser seiring pesawat berpindah benua/samudera.\n"
16272 |
16273 |             self.after(0, lambda: self.txt_flight_report.insert("1.0", report))
16274 |
16275 |     except Exception as e:
16276 |         import traceback
16277 |         self.after(0, lambda: self.txt_flight_report.insert("1.0", f"TERJADI KESALAHAN:\n{str(e)}\n\n{traceback.format_exc()}"))
16278 |     finally:
16279 |         self.after(0, lambda: self.txt_flight_report.configure(state="disabled"))
16280 |         self.after(0, lambda: self.btn_calc_flight.configure(state="normal", text="► HITUNG JADWAL SELAMA PENERBANGAN"))
16281 |
16282 |     def calculate_celestial_qiblah(self):
16283 |         """Mencari jam di mana bintang tertentu melintasi Azimut Kiblat lokal"""
16284 |         try:
16285 |             from skyfield.api import Star, wgs84
16286 |
16287 |             # 1. Ambil Parameter Lokasi & Tanggal
16288 |             lat, lon = float(self.entry_qlat.get()), float(self.entry_qlon.get())
16289 |             tz = float(self.entry_qtz.get())
16290 |             y, m, d = int(self.entry_qyear.get()), int(self.entry_qmonth.get()), int(self.entry_qday.get())
16291 |
16292 |             # 2. Hitung Azimut Kiblat (Spherical Trigonometry)
16293 |             phi_k, lam_k = math.radians(21.4225), math.radians(39.8262)
16294 |             phi, lam = math.radians(lat), math.radians(lon)
16295 |             y_q = math.sin(lam_k - lam)
16296 |             x_q = math.cos(phi)*math.tan(phi_k) - math.sin(phi)*math.cos(lam_k-lam)
16297 |             q_az = math.degrees(math.atan2(y_q, x_q)) % 360
16298 |
16299 |             # 3. Katalog Bintang (RA/Dec J2000)
16300 |             star_catalog = {
16301 |                 "Canopus (Suhail)": Star(ra_hours=(6, 23, 57.1), dec_degrees=(-52, 41, 45)),
16302 |                 "Sirius (Asy-Syira)": Star(ra_hours=(6, 45, 8.9), dec_degrees=(-16, 42, 58)),
16303 |                 "Rigel (Orion)": Star(ra_hours=(5, 14, 32.3), dec_degrees=(-8, 12, 6)),
16304 |                 "Betelgeuse (Orion)": Star(ra_hours=(5, 55, 10.3), dec_degrees=(7, 24, 25)),

```

```

16305 |         "Acrux (Crux/Salib Sel.)": Star(ra_hours=(12, 26, 35.9), dec_degrees=(-63, 5, 57)),
16306 |         "Vega": Star(ra_hours=(18, 36, 56.3), dec_degrees=(38, 47, 1)),
16307 |         "Altair": Star(ra_hours=(19, 50, 47), dec_degrees=(8, 52, 6)),
16308 |         "Antares": Star(ra_hours=(16, 29, 24.4), dec_degrees=(-26, 25, 55)),
16309 |         "Spica": Star(ra_hours=(13, 25, 11.6), dec_degrees=(-11, 9, 41))
16310 |     }
16311 |
16312 |     target_star = star_catalog[self.combo_q_star.get()]
16313 |     self.auto_switch_ephemeris(y)
16314 |     earth = self.eph['earth']
16315 |     loc = wgs84.latlon(lat, lon)
16316 |     observer = earth + loc
16317 |
16318 |     # 4. Iterasi Pencarian (Rentang 24 jam)
16319 |     t0 = self.ts.utc(y, m, d, -int(tz))
16320 |     tt_array = np.linspace(t0.tt, t0.tt + 1.0, 1440) # Sampel per menit
16321 |     t_search = self.ts.tt_jd(tt_array)
16322 |
16323 |     # Ambil Azimut & Altitude Bintang
16324 |     app = observer.at(t_search).observe(target_star).apparent()
16325 |     alt_arr, az_arr, _ = app.altaz()
16326 |
16327 |     diffs = (az_arr.degrees - q_az + 180) % 360 - 180
16328 |     results = []
16329 |
16330 |     for i in range(len(diffs)-1):
16331 |         if (diffs[i] <= 0 and diffs[i+1] > 0) or (diffs[i] >= 0 and diffs[i+1] < 0):
16332 |             if alt_arr.degrees[i] > 5: # Minimal 5 derajat di atas ufuk agar terlihat
16333 |                 frac = abs(diffs[i]) / (abs(diffs[i]) + abs(diffs[i+1]) + 1e-9)
16334 |                 t_res = self.ts.tt_jd(tt_array[i] + frac * (tt_array[i+1] - tt_array[i]))
16335 |                 dt_lokal = t_res.utc_datetime() + datetime.timedelta(hours=tz)
16336 |                 results.append((dt_lokal.strftime("%H:%M:%S"), alt_arr.degrees[i]))
16337 |
16338 |     # 5. Output Report
16339 |     rep = f"{self.get_header(85)}\n"
16340 |     rep += f"{'[ Celestial Qiblah: Verifikasi via Bintang ]'.center(85)}\n\n"
16341 |     rep += f"* Objek Bintang : {self.combo_q_star.get()}\n"
16342 |     rep += f"* Azimut Kiblat : {q_az:.2f}° (dari Utara)\n"
16343 |     rep += f"* Lokasi          : Lat {lat}, Lon {lon}, TZ {tz}\n"
16344 |     rep += f"*85 + "\n\n"
16345 |
16346 |     if not results:
16347 |         rep += f">> Sayang sekali, {self.combo_q_star.get()} tidak melintasi jalur azimut kiblat\n"
16348 |         rep += f"    di lokasi Anda pada tanggal {d:02d}-{m:02d}-{y}.\n"
16349 |     else:
16350 |         rep += f"{'Waktu (LT)':<15} | {'Ketinggian Bintang':<20} | {'Catatan'}\n"
16351 |         rep += f"*85 + "\n"
16352 |         for w, alt in results:
16353 |             rep += f"{'w':<15} | {alt:>17.2f}° | Berada tepat di arah Ka'bah\n"
16354 |             rep += f"\n* Gunakan waktu di atas untuk melihat bintang tersebut.\n"
16355 |             rep += f"    Jika bintang berada tepat di depan Anda, maka itulah arah Kiblat Sejati.\n"
16356 |
16357 |     self.after(0, self.display_result, rep)
16358 |
16359 |     except Exception as e:
16360 |         self.after(0, self.display_error, str(e))
16361 |
16362 | # =====
16363 | # MODUL TAMBAHAN: KURVA ALASKA (TERINTEGRASI MENU 5)
16364 | # =====
16365 | def setup_kurva_alaska_out_frame(self):
16366 |     """Menyiapkan ruang output (kanan) untuk Kurva Alaska"""
16367 |     self.frame_alaska_out = ctk.CTkFrame(self.main_frame, fg_color="#101010")
16368 |
16369 |     self.fig_alaska = plt.Figure(figsize=(8, 6), dpi=100, facecolor="#101010")
16370 |     self.ax_alaska = self.fig_alaska.add_subplot(111)
16371 |     self.ax_alaska.set_facecolor("#1E1E1E")
16372 |
16373 |     self.canvas_alaska = FigureCanvasTkAgg(self.fig_alaska, master=self.frame_alaska_out)
16374 |     self.canvas_alaska.get_tk_widget().pack(fill="both", expand=True, padx=10, pady=10)
16375 |

```

```

16376 |         # Tooltip Hover
16377 |         self.annot_alaska = self.ax_alaska.annotate(
16378 |             "", xy=(0,0), xytext=(15, 15), textcoords="offset points",
16379 |             bbox=dict(boxstyle="round,pad=0.5", fc="#2A2A2A", ec="white", lw=1, alpha=0.9),
16380 |             color="white", fontsize=10, zorder=10
16381 |         )
16382 |         self.annot_alaska.set_visible(False)
16383 |         self.canvas_alaska.mpl_connect("motion_notify_event", self.on_hover_alaska)
16384 |
16385 |     def proses_kurva_alaska(self):
16386 |         self.lbl_status.configure(text="Menghitung Kurva Alaska...", text_color="#00E5FF")
16387 |         self.btn_kurva_alaska.configure(state="disabled", text="MENGHITUNG...")
16388 |         threading.Thread(target=self._kalkulasi_kurva_alaska, daemon=True).start()
16389 |
16390 |     def _kalkulasi_kurva_alaska(self):
16391 |         try:
16392 |             from skyfield import almanac
16393 |             import math
16394 |             import datetime
16395 |
16396 |             y = int(self.entry_gha_year.get())
16397 |             m = int(self.entry_gha_month.get())
16398 |             lat = float(self.entry_gha_lat.get())
16399 |             lon = float(self.entry_gha_lon.get())
16400 |             tz_offset = float(self.entry_gha_tz.get())
16401 |
16402 |             self.auto_switch_ephemeris(y)
16403 |             earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
16404 |             lokasi = wgs84.latlon(lat, lon)
16405 |             observer = earth + lokasi
16406 |
16407 |             f_sunset = almanac.sunrise_sunset(self.eph, lokasi)
16408 |             _, num_days = calendar.monthrange(y, m)
16409 |
16410 |             days, altitudes, elongations, maghrib_times = [], [], [], []
16411 |
16412 |             for d in range(1, num_days + 1):
16413 |                 t0 = self.ts.utc(y, m, d, -int(tz_offset))
16414 |                 t1 = self.ts.utc(y, m, d + 1, -int(tz_offset))
16415 |
16416 |                 t_events, y_events = almanac.find_discrete(t0, t1, f_sunset)
16417 |
16418 |                 # Filter hanya saat Matahari Terbenam (y_events == 0)
16419 |                 sunsets = [t for t, is_sr in zip(np.atleast_1d(t_events), np.atleast_1d(y_events)) if not is_sr]
16420 |
16421 |                 if sunsets:
16422 |                     t_maghrib = sunsets[0]
16423 |                     t_maghrib_local = t_maghrib.utc_datetime() + datetime.timedelta(hours=tz_offset)
16424 |                     time_str = t_maghrib_local.strftime('%H:%M:%S')
16425 |
16426 |                     geo_sun = earth.at(t_maghrib).observe(sun).apparent()
16427 |                     geo_moon = earth.at(t_maghrib).observe(moon).apparent()
16428 |                     elongation = geo_sun.separation_from(geo_moon).degrees
16429 |
16430 |                     topo_moon = observer.at(t_maghrib).observe(moon).apparent()
16431 |                     alt_topo_m, _, _ = topo_moon.altaz(temperature_C=25, pressure_mbar=1010)
16432 |
16433 |                     days.append(d)
16434 |                     altitudes.append(alt_topo_m.degrees)
16435 |                     elongations.append(elongation)
16436 |                     maghrib_times.append(time_str)
16437 |                 else:
16438 |                     days.append(d)
16439 |                     altitudes.append(None)
16440 |                     elongations.append(None)
16441 |                     maghrib_times.append(None)
16442 |
16443 |             self.after(0, self._render_kurva_alaska, days, altitudes, elongations, maghrib_times, lat, lon, y, m)
16444 |
16445 |         except Exception as e:
16446 |             import traceback

```

```

16447 |         self.after(0, self.display_error, f"Gagal menghitung kurva:\n{str(e)}\n{traceback.format_exc()}")
16448 |         self.after(0, lambda: self.btn_kurva_alaska.configure(state="normal", text="GENERATE KURVA ALASKA"))
16449 |
16450 | def _render_kurva_alaska(self, days, altitudes, elongations, maghrib_times, lat, lon, year, month):
16451 |     self.ax_alaska.clear()
16452 |     self.ax_alaska.set_facecolor("#1E1E1E")
16453 |
16454 |     self.valid_days_alaska.clear()
16455 |     self.valid_alt_alaska.clear()
16456 |     self.valid_elong_alaska.clear()
16457 |     self.hover_data_alaska.clear()
16458 |
16459 |     for d, a, e, t in zip(days, altitudes, elongations, maghrib_times):
16460 |         if a is not None:
16461 |             self.valid_days_alaska.append(d)
16462 |             self.valid_alt_alaska.append(a)
16463 |             self.valid_elong_alaska.append(e)
16464 |             self.hover_data_alaska[d] = {'time': t}
16465 |
16466 |     # Plot Data
16467 |     self.ax_alaska.plot(self.valid_days_alaska, self.valid_alt_alaska, color="#00E5FF", label='Tinggi Hilal (Topo)', linewidth=2.5, marker='o', markersize=5)
16468 |     self.ax_alaska.plot(self.valid_days_alaska, self.valid_elong_alaska, color="#FFD700", label='Elongasi (Geo)', linewidth=2.5, marker='s', markersize=5)
16469 |
16470 |     self.ax_alaska.axhline(5, color='#4CAF50', linestyle='--', linewidth=1.5, label='Batas Alt 5°')
16471 |     self.ax_alaska.axhline(8, color='#FF5722', linestyle='--', linewidth=1.5, label='Batas Elong 8°')
16472 |
16473 |     for d, alt, elong in zip(self.valid_days_alaska, self.valid_alt_alaska, self.valid_elong_alaska):
16474 |         if -5 <= alt <= 20:
16475 |             self.ax_alaska.annotate(f"{alt:.1f}°", (d, alt), textcoords="offset points", xytext=(0,10), ha='center', fontsize=9, color="#00E5FF")
16476 |             self.ax_alaska.annotate(f"{elong:.1f}°", (d, elong), textcoords="offset points", xytext=(0,-15), ha='center', fontsize=9, color="#FFD700")
16477 |
16478 |     self.annot_alaska = self.ax_alaska.annotate("", xy=(0,0), xytext=(15, 15), textcoords="offset points",
16479 |         bbox=dict(boxstyle="round,pad=0.5", fc="#2A2A2A", ec="white", lw=1, alpha=0.9),
16480 |         color="white", fontsize=10, zorder=10)
16481 |     self.annot_alaska.set_visible(False)
16482 |
16483 |     self.ax_alaska.set_title(f"Posisi Bulan Waktu Maghrib (Lat: {lat}, Lon: {lon}) - {year}/{month}", color="white", fontsize=14, pad=15)
16484 |     self.ax_alaska.set_xlabel("Tanggal", color="lightgray", fontsize=12)
16485 |     self.ax_alaska.set_ylabel("Derajat (°)", color="lightgray", fontsize=12)
16486 |     self.ax_alaska.set_xticks(self.valid_days_alaska)
16487 |
16488 |     if self.valid_alt_alaska and self.valid_elong_alaska:
16489 |         min_y = max(min(self.valid_alt_alaska) - 5, -15)
16490 |         max_y = min(max(self.valid_elong_alaska) + 5, 40)
16491 |         self.ax_alaska.set_ylim(min_y, max_y)
16492 |
16493 |     self.ax_alaska.grid(True, color="#444444", linestyle=':', linewidth=0.8)
16494 |     self.ax_alaska.legend(loc='upper left', facecolor="#222222", edgecolor="none", labelcolor="white")
16495 |
16496 |     self.fig_alaska.tight_layout()
16497 |     self.canvas_alaska.draw()
16498 |
16499 |     # ---> PERBAIKAN UI: Hanya sembunyikan textbox tanpa menghapus Header <---
16500 |     self.textbox.grid_remove()
16501 |     self.frame_alaska_out.grid(row=1, column=0, sticky="nsew")
16502 |
16503 |     self.lbl_status.configure(text="Kurva Alaska Berhasil Dirender", text_color="#00E676")
16504 |     self.btn_kurva_alaska.configure(state="normal", text="GENERATE KURVA ALASKA")
16505 |
16506 | def on_hover_alaska(self, event):
16507 |     """Sistem Interpolasi Hover Interaktif Alaska"""
16508 |     if event.inaxes == self.ax_alaska:
16509 |         x_val = event.xdata
16510 |         y_val = event.ydata
16511 |
16512 |         if x_val is None or y_val is None: return
16513 |
16514 |         if not self.valid_days_alaska or x_val < min(self.valid_days_alaska) or x_val > max(self.valid_days_alaska):
16515 |             if self.annot_alaska.get_visible():
16516 |                 self.annot_alaska.set_visible(False)
16517 |                 self.canvas_alaska.draw_idle()

```

```

16518 |         return
16519 |
16520 |         import numpy as np
16521 |         calc_alt = np.interp(x_val, self.valid_days_alaska, self.valid_alt_alaska)
16522 |         calc_elong = np.interp(x_val, self.valid_days_alaska, self.valid_elong_alaska)
16523 |
16524 |         closest_day = int(round(x_val))
16525 |         time_str = self.hover_data_alaska.get(closest_day, {}).get('time', '--:--:--')
16526 |
16527 |         text = (f"Posisi X: {x_val:.2f}\n"
16528 |                 f"Ref. Maghrib: Tgl {closest_day} ({time_str})\n"
16529 |                 f"Est. Altitude: {calc_alt:.2f}°\n"
16530 |                 f"Est. Elongasi: {calc_elong:.2f}°")
16531 |
16532 |         self.annot_alaska.xy = (x_val, y_val)
16533 |         self.annot_alaska.set_text(text)
16534 |         self.annot_alaska.set_visible(True)
16535 |         self.canvas_alaska.draw_idle()
16536 |     else:
16537 |         if self.annot_alaska.get_visible():
16538 |             self.annot_alaska.set_visible(False)
16539 |             self.canvas_alaska.draw_idle()
16540 |
16541 | # =====
16542 | # MODUL 9 LANJUTAN: TREN BULANAN HIJRIAH (RAJAB - ZULHIJAH)
16543 | # =====
16544 | def trigger_trend(self, trend_type):
16545 |     """Memantik render grafik tren ke background thread"""
16546 |     self.lbl_status.configure(text="Mengekstrak Data Tren Bulanan...", text_color="#00E5FF")
16547 |     import threading
16548 |     threading.Thread(target=self.calculate_trend_hijri, args=(trend_type,), daemon=True).start()
16549 |
16550 | def calculate_trend_hijri(self, trend_type):
16551 |     try:
16552 |         import math
16553 |         import numpy as np
16554 |         import datetime
16555 |         import pytz
16556 |         from skyfield import almanac
16557 |         from skyfield.api import wgs84
16558 |
16559 |         h_year = int(self.entry_chart_hy.get())
16560 |         lat = float(self.entry_vlat.get())
16561 |         lon = float(self.entry_vlon.get())
16562 |         tz = float(self.entry_vtz.get())
16563 |         elev = float(self.entry_velev.get())
16564 |
16565 |         if h_year not in HIJRI_DB:
16566 |             raise ValueError(f"Tahun Hijriah {h_year} tidak tersedia di database.")
16567 |
16568 |         # Mengambil data bulan 7 (Rajab) sampai 12 (Zulhijah) dari HIJRI_DB
16569 |         # Index -> 6: Rajab, 7: Syakban, 8: Ramadan, 9: Syawal, 10: Zulkaidah, 11: Zulhijah
16570 |         bulan_map = {"Jan":1, "Feb":2, "Mar":3, "Apr":4, "May":5, "Jun":6, "Jul":7, "Aug":8, "Sep":9, "Oct":10, "Nov":11, "Dec":12}
16571 |
16572 |         # 1. Cari Tanggal Mulai (1 Rajab)
16573 |         r_data = HIJRI_DB[h_year][6]
16574 |         p_start = r_data[2].split('-')
16575 |         start_date = datetime.date(int(p_start[2]), bulan_map[p_start[1]], int(p_start[0]))
16576 |
16577 |         # 2. Cari Tanggal Akhir (Akhir Zulhijah)
16578 |         z_data = HIJRI_DB[h_year][11]
16579 |         p_end = z_data[2].split('-')
16580 |         z_start = datetime.date(int(p_end[2]), bulan_map[p_end[1]], int(p_end[0]))
16581 |         end_date = z_start + datetime.timedelta(days=z_data[3]-1)
16582 |
16583 |         num_days = (end_date - start_date).days + 1
16584 |
16585 |         # Load Ephemeris di tengah-tengah rentang waktu
16586 |         mid_date = start_date + datetime.timedelta(days=num_days//2)
16587 |         self.auto_switch_ephemeris(mid_date.year)
16588 |

```

```

16589 |         earth, sun, moon = self.eph['earth'], self.eph['sun'], self.eph['moon']
16590 |         loc = wgs84.latlon(lat, lon, elevation_m=elev)
16591 |
16592 |         dates_greg = []
16593 |         alt_topo, alt_geo = [], []
16594 |         elong_topo, elong_geo = [], []
16595 |         illum = []
16596 |
16597 |         # Kumpulkan penanda tanggal awal bulan Hijriah
16598 |         hijri_marks = []
16599 |         for m_idx in range(6, 12):
16600 |             m_data = HIJRI_DB[h_year][m_idx]
16601 |             p_m = m_data[2].split('-')
16602 |             d_m = datetime.date(int(p_m[2]), bulan_map[p_m[1]], int(p_m[0]))
16603 |             hijri_marks.append({'date': d_m, 'name': m_data[0]})
16604 |
16605 |         # Looping per hari (Komputasi pada waktu Maghrib/Sunset Lokal)
16606 |         for i in range(num_days):
16607 |             curr_date = start_date + datetime.timedelta(days=i)
16608 |             dates_greg.append(curr_date)
16609 |
16610 |             t0 = self.ts.utc(curr_date.year, curr_date.month, curr_date.day, 0 - int(tz))
16611 |             t1 = self.ts.utc(curr_date.year, curr_date.month, curr_date.day, 24 - int(tz))
16612 |
16613 |             f_rs = almanac.sunrise_sunset(self.eph, loc)
16614 |             t_rs, y_rs = almanac.find_discrete(t0, t1, f_rs)
16615 |             t_sunset = None
16616 |             if t_rs is not None:
16617 |                 for tv, yv in zip(np.atleast_1d(t_rs.tt), np.atleast_1d(y_rs)):
16618 |                     if yv == 0:
16619 |                         t_sunset = self.ts.tt(jd=tv)
16620 |                         break
16621 |             if t_sunset is None:
16622 |                 t_sunset = self.ts.utc(curr_date.year, curr_date.month, curr_date.day, 18 - int(tz))
16623 |
16624 |             # --- KALKULASI TOPOSENTRIK ---
16625 |             obs_topo = (earth + loc).at(t_sunset)
16626 |             app_moon_topo = obs_topo.observe(moon).apparent()
16627 |             app_sun_topo = obs_topo.observe(sun).apparent()
16628 |
16629 |             alt_topo.append(app_moon_topo.altaz(temperature_C=25, pressure_mbar=1010)[0].degrees)
16630 |             elong_topo.append(app_sun_topo.separation_from(app_moon_topo).degrees)
16631 |
16632 |             # --- KALKULASI GEOSENTRIK ---
16633 |             obs_geo = earth.at(t_sunset)
16634 |             app_moon_geo = obs_geo.observe(moon).apparent()
16635 |             app_sun_geo = obs_geo.observe(sun).apparent()
16636 |
16637 |             elong_geo.append(app_sun_geo.separation_from(app_moon_geo).degrees)
16638 |
16639 |             ra_m, dec_m, _ = app_moon_geo.radec(epoch=t_sunset)
16640 |             lst_deg = (t_sunset.gast * 15.0) + lon
16641 |             ha_rad = math.radians(lst_deg - (ra_m.hours * 15.0))
16642 |             d_rad = dec_m.radians
16643 |             lat_rad = math.radians(lat)
16644 |
16645 |             sin_alt_geo = math.sin(d_rad)*math.sin(lat_rad) + math.cos(d_rad)*math.cos(lat_rad)*math.cos(ha_rad)
16646 |             alt_geo.append(math.degrees(math.asin(max(-1.0, min(1.0, sin_alt_geo)))))
16647 |
16648 |             # --- ILUMINASI ---
16649 |             il = almanac.fraction_illuminated(self.eph, 'moon', t_sunset)
16650 |             illum.append((il.item() if hasattr(il, 'item') else il) * 100.0)
16651 |
16652 |         data_bundle = {
16653 |             'dates': dates_greg,
16654 |             'alt_topo': alt_topo, 'alt_geo': alt_geo,
16655 |             'elong_topo': elong_topo, 'elong_geo': elong_geo,
16656 |             'illum': illum, 'hijri_marks': hijri_marks,
16657 |             'type': trend_type, 'h_year': h_year
16658 |         }
16659 |

```



```

16660 |         self.after(0, self.plot_trend_dispatcher, data_bundle)
16661 |
16662 |     except Exception as e:
16663 |         import traceback
16664 |         self.after(0, self._force_show_error, f"{str(e)}\n\n{traceback.format_exc()}")
16665 |
16666 | def plot_trend_dispatcher(self, data):
16667 |     """Merender canvas grafik jangka panjang (Bulan berbulan-bulan)"""
16668 |     try:
16669 |         for widget in self.frame_chart_out.wininfo_children():
16670 |             widget.destroy()
16671 |
16672 |         from matplotlib.figure import Figure
16673 |         from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
16674 |         import matplotlib.dates as mdates
16675 |         import numpy as np
16676 |
16677 |         fig = Figure(figsize=(9, 5.5), dpi=100)
16678 |         fig.patch.set_facecolor('#181818')
16679 |         ax = fig.add_subplot(111)
16680 |         ax.set_facecolor('#101010')
16681 |
16682 |         trend_type = data['type']
16683 |         dates = data['dates']
16684 |         h_year = data['h_year']
16685 |
16686 |         x_dates = mdates.date2num(dates)
16687 |
16688 |         self.trend_type = trend_type
16689 |         self.trend_data = data
16690 |
16691 |         if trend_type == "altitude":
16692 |             ax.plot(x_dates, data['alt_topo'], color='#80DEEA', label='Altitude Toposentrik', linewidth=1.5)
16693 |             ax.plot(x_dates, data['alt_geo'], color='#00E676', label='Altitude Geosentrik', linewidth=1.5, linestyle='--')
16694 |             ax.axhline(5, color='#00E676', linestyle=':', linewidth=1, label="Batas Tinggi KHGT (5°)")
16695 |             ax.axhline(3, color='#80DEEA', linestyle=':', linewidth=1, label="Batas Tinggi MABIMS (3°)")
16696 |             ax.axhline(0, color='#FF5252', linestyle='-', linewidth=1)
16697 |             ax.set_title(f"Tren Ketinggian Hilal Saat Maghrib (Rajab - Zulhijah {h_year} H)", color='white', fontsize=14, fontweight='bold', pad=15)
16698 |             ax.set_ylabel("Ketinggian / Altitude (°)", color='ffffff', fontsize=12)
16699 |
16700 |         elif trend_type == "elongation":
16701 |             ax.plot(x_dates, data['elong_topo'], color='#FFB74D', label='Elongasi Toposentrik', linewidth=1.5)
16702 |             ax.plot(x_dates, data['elong_geo'], color='#FFD54F', label='Elongasi Geosentrik', linewidth=1.5, linestyle='--')
16703 |             ax.axhline(8, color='#FFD54F', linestyle=':', linewidth=1, label="Batas Elongasi KHGT (8°)")
16704 |             ax.axhline(6.4, color='#FFB74D', linestyle=':', linewidth=1, label="Batas Elongasi MABIMS (6.4°)")
16705 |             ax.set_title(f"Tren Elongasi Bulan-Matahari Saat Maghrib (Rajab - Zulhijah {h_year} H)", color='white', fontsize=14, fontweight='bold', pad=15)
16706 |             ax.set_ylabel("Sudut Elongasi (°)", color='ffffff', fontsize=12)
16707 |
16708 |         elif trend_type == "illumination":
16709 |             ax.plot(x_dates, data['illum'], color='#E040FB', label='Fraksi Iluminasi Bulan', linewidth=2)
16710 |             ax.set_title(f"Siklus Iluminasi / Fase Bulan Saat Maghrib (Rajab - Zulhijah {h_year} H)", color='white', fontsize=14, fontweight='bold', pad=15)
16711 |             ax.set_ylabel("Iluminasi (%)", color='ffffff', fontsize=12)
16712 |
16713 |         # ---> TAMBAHAN LOGIKA PLOT KOMBINASI <---
16714 |         elif trend_type == "kombinasi":
16715 |             ax.plot(x_dates, data['alt_topo'], color='#80DEEA', label='Alt Topo (°)', linewidth=1.5)
16716 |             ax.plot(x_dates, data['alt_geo'], color='#00E676', label='Alt Geo (°)', linewidth=1.5, linestyle='--')
16717 |             ax.plot(x_dates, data['elong_topo'], color='#FFB74D', label='Eln Topo (°)', linewidth=1.5)
16718 |             ax.plot(x_dates, data['elong_geo'], color='#FFD54F', label='Eln Geo (°)', linewidth=1.5, linestyle='-.')
16719 |             ax.plot(x_dates, data['illum'], color='#E040FB', label='Iluminasi (%)', linewidth=2, alpha=0.8)
16720 |
16721 |             ax.axhline(0, color='#FF5252', linestyle='-', linewidth=1)
16722 |             ax.set_title(f"Tren Kombinasi (Alt, Eln, Illum) Saat Maghrib\nRajab - Zulhijah {h_year} H", color='white', fontsize=14, fontweight='bold', pad=15)
16723 |             ax.set_ylabel("Nilai (Derajat / Persen)", color='ffffff', fontsize=12)
16724 |             # -----
16725 |
16726 |         # Garis vertikal pembatas antar bulan Hijriah
16727 |         colors_h = ['#EF5350', '#AB47BC', '#29B6F6', '#66BB6A', '#FFA726', '#FF7043']
16728 |         for i, mark in enumerate(data['hijri_marks']):
16729 |             mx = mdates.date2num(mark['date'])
16730 |             ax.axvline(mx, color=colors_h[i], linestyle='-', linewidth=1.5, alpha=0.8)

```

```

16731 |         ax.text(mx + 1, ax.get_ylim()[1] * 0.95, f"1 {mark['name']}", color=colors_h[i], rotation=90, verticalalignment='top', fontsize=9, fontweight='bold')
16732 |
16733 |     # Atur format sumbu X menjadi tanggal
16734 |     ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %b'))
16735 |     ax.xaxis.set_major_locator(mdates.DayLocator(interval=15))
16736 |
16737 |     ax.tick_params(colors='#ffffff', labelsize=10)
16738 |     ax.grid(True, color='#ffffff', linestyle=':', alpha=0.2)
16739 |     ax.legend(fontsize=10, facecolor='#181818', edgecolor='#333333', labelcolor='white', loc='lower right')
16740 |     fig.autofmt_xdate()
16741 |     fig.tight_layout()
16742 |
16743 |     self.trend_ax = ax
16744 |     self.trend_canvas = FigureCanvasTkAgg(fig, master=self.frame_chart_out)
16745 |
16746 |     # Sistem Tooltip Hover Matplotlib
16747 |     self.annot_trend = ax.annotate(
16748 |         "", xy=(0,0), xytext=(15, 15), textcoords="offset points",
16749 |         bbox=dict(boxstyle="round,pad=0.5", fc="#1E1E1E", ec="#FFD54F", lw=1.5, alpha=0.9),
16750 |         color="white", fontsize=10, zorder=10
16751 |     )
16752 |     self.annot_trend.set_visible(False)
16753 |     self.trend_canvas.mpl_connect("motion_notify_event", self.on_hover_trend)
16754 |
16755 |     self.trend_canvas.draw()
16756 |     self.trend_canvas.get_tk_widget().pack(fill="both", expand=True)
16757 |
16758 |     self.lbl_status.configure(text="Grafik Tren Berhasil Dimuat", text_color="#00E676")
16759 |     self.btn_hitung.configure(state="normal")
16760 |
16761 | except Exception as e:
16762 |     import traceback
16763 |     self._force_show_error(f"Gagal Merender Grafik Tren:\n{str(e)}\n{traceback.format_exc()}")
16764 |
16765 | def on_hover_trend(self, event):
16766 |     if event.inaxes == getattr(self, 'trend_ax', None):
16767 |         x_val = event.xdata
16768 |         y_val = event.ydata
16769 |         if x_val is None or y_val is None: return
16770 |
16771 |         data = getattr(self, 'trend_data', None)
16772 |         if not data: return
16773 |
16774 |         import matplotlib.dates as mdates
16775 |         import numpy as np
16776 |
16777 |         x_dates = mdates.date2num(data['dates'])
16778 |         if x_val < x_dates[0] or x_val > x_dates[-1]:
16779 |             self.annot_trend.set_visible(False)
16780 |             self.trend_canvas.draw_idle()
16781 |             return
16782 |
16783 |         # Cari hari / index terdekat
16784 |         idx = (np.abs(x_dates - x_val)).argmin()
16785 |         closest_date = data['dates'][idx]
16786 |         date_str = closest_date.strftime("%d %b %Y")
16787 |
16788 |         if self.trend_type == "altitude":
16789 |             v_topo = data['alt_topo'][idx]
16790 |             v_geo = data['alt_geo'][idx]
16791 |             text = f"□ {date_str} (Maghrib)\n□ Alt Topo: {v_topo:+.2f}°\n□ Alt Geo : {v_geo:+.2f}°"
16792 |         elif self.trend_type == "elongation":
16793 |             v_topo = data['elong_topo'][idx]
16794 |             v_geo = data['elong_geo'][idx]
16795 |             text = f"□ {date_str} (Maghrib)\n□ Eln Topo: {v_topo:.2f}°\n□ Eln Geo : {v_geo:.2f}°"
16796 |         elif self.trend_type == "illumination":
16797 |             v_ill = data['illum'][idx]
16798 |             text = f"□ {date_str} (Maghrib)\n□ Iluminasi: {v_ill:.2f}%"
16799 |         # ---> TAMBAHAN HOVER UNTUK MODE KOMBINASI <---
16800 |         elif self.trend_type == "kombinasi":
16801 |             v_alt_t = data['alt_topo'][idx]

```

```

16802 |         v_alt_g = data['alt_geo'][idx]
16803 |         v_eln_t = data['elong_topo'][idx]
16804 |         v_eln_g = data['elong_geo'][idx]
16805 |         v_ill = data['illum'][idx]
16806 |         text = (f" {date_str} (Maghrib)\n"
16807 |               f" Alt Topo: {v_alt_t:.2f}° | Geo: {v_alt_g:.2f}°\n"
16808 |               f" Eln Topo: {v_eln_t:.2f}° | Geo: {v_eln_g:.2f}°\n"
16809 |               f" Iluminasi: {v_ill:.2f}%")
16810 |         # -----
16811 |
16812 |         self.annot_trend.xy = (x_dates[idx], y_val)
16813 |         self.annot_trend.set_text(text)
16814 |         self.annot_trend.set_visible(True)
16815 |         self.trend_canvas.draw_idle()
16816 |     else:
16817 |         if getattr(self, 'annot_trend', None) and self.annot_trend.get_visible():
16818 |             self.annot_trend.set_visible(False)
16819 |             self.trend_canvas.draw_idle()
16820 | # =====
16821 | # SISTEM MULTI-BAHASA (JSON TRANSLATOR)
16822 | # =====
16823 | def _cache_base_dir(self):
16824 |     """Folder aplikasi, kompatibel untuk mode .py dan .exe."""
16825 |     try:
16826 |         if getattr(sys, 'frozen', False):
16827 |             return os.path.dirname(sys.executable)
16828 |             return os.path.dirname(os.path.abspath(__file__))
16829 |     except Exception:
16830 |         return os.getcwd()
16831 |
16832 | def _find_cache_file(self, filename):
16833 |     """Cari file cache bahasa; menerima nama normal dan salinan seperti name(2).json."""
16834 |     base_path = self._cache_base_dir()
16835 |     direct = os.path.join(base_path, filename)
16836 |     if os.path.exists(direct):
16837 |         return direct
16838 |     stem, ext = os.path.splitext(filename)
16839 |     try:
16840 |         for name in os.listdir(base_path):
16841 |             if name == filename:
16842 |                 return os.path.join(base_path, name)
16843 |             if name.startswith(stem + "(") and name.endswith(ext):
16844 |                 return os.path.join(base_path, name)
16845 |     except Exception:
16846 |         pass
16847 |     return direct
16848 |
16849 | def _merge_translation_mapping(self, data):
16850 |     """Gabungkan cache bahasa ke self.translations dalam format {teks_id: {en, ar}}."""
16851 |     if not isinstance(data, dict):
16852 |         return
16853 |
16854 |     # Dukungan format release: {"en": {raw: trans}, "ar": {raw: trans}}
16855 |     if isinstance(data.get("en"), dict) or isinstance(data.get("ar"), dict):
16856 |         looks_like_release = any(
16857 |             isinstance(v, dict) and v and all(isinstance(x, str) for x in list(v.values())[:5])
16858 |             for k, v in data.items() if k in ("en", "ar")
16859 |         )
16860 |         if looks_like_release and all(k in ("en", "ar") for k in data.keys()):
16861 |             for lang in ("en", "ar"):
16862 |                 for key, trans in (data.get(lang) or {}).items():
16863 |                     if isinstance(key, str) and isinstance(trans, str):
16864 |                         self.translations.setdefault(key, {})[lang] = trans
16865 |             return
16866 |
16867 |     # Format utama UI: {teks_sumber: {"id": ..., "en": ..., "ar": ...}}
16868 |     # Kunci "id" opsional dipakai untuk membersihkan teks campuran Indonesia/Inggris
16869 |     # tanpa mengubah kunci internal menu maupun logika modul.
16870 |     for key, langs in data.items():
16871 |         if not isinstance(key, str) or not isinstance(langs, dict):
16872 |             continue

```

```

16873 |         entry = self.translations.setdefault(key, {})
16874 |         for lang in ("id", "en", "ar"):
16875 |             val = langs.get(lang)
16876 |             if isinstance(val, str) and val:
16877 |                 entry[lang] = val
16878 |
16879 |     def _build_language_indexes(self):
16880 |         """Bangun indeks 0(1), indeks balik, dan reset regex translasi."""
16881 |         self._translation_index = {"id": {}, "en": {}, "ar": {}}
16882 |         self._reverse_translation_map = {}
16883 |
16884 |         for source_text, langs in (getattr(self, "translations", {}) or {}).items():
16885 |             if not isinstance(source_text, str):
16886 |                 continue
16887 |             if not isinstance(langs, dict):
16888 |                 langs = {}
16889 |
16890 |             canonical_id = langs.get("id")
16891 |             if not isinstance(canonical_id, str) or not canonical_id:
16892 |                 canonical_id = source_text
16893 |
16894 |             variants = [source_text, canonical_id]
16895 |             for lang in ("en", "ar"):
16896 |                 value = langs.get(lang)
16897 |                 if isinstance(value, str) and value:
16898 |                     variants.append(value)
16899 |
16900 |             for value in variants:
16901 |                 if isinstance(value, str) and value:
16902 |                     self._reverse_translation_map[self._norm_lang_text(value)] = canonical_id
16903 |
16904 |             for lang in ("id", "en", "ar"):
16905 |                 if lang == "id":
16906 |                     target = canonical_id
16907 |                 else:
16908 |                     target = langs.get(lang)
16909 |                     if not isinstance(target, str) or not target:
16910 |                         target = canonical_id
16911 |                     self._translation_index[lang][source_text] = target
16912 |                     self._translation_index[lang][canonical_id] = target
16913 |
16914 |             self._translation_patterns = {}
16915 |             self._translation_sorted_keys_cache = None
16916 |             self._release_sorted_keys_cache = None
16917 |
16918 |     def load_translations(self):
16919 |         """Memuat cache UI eksternal dan cache output agar perpindahan ID/EN/AR cepat."""
16920 |         self.translations = {}
16921 |         self._translation_index = {"id": {}, "en": {}, "ar": {}}
16922 |         self._reverse_translation_map = {}
16923 |         self._translation_patterns = {}
16924 |         self._translation_sorted_keys_cache = None
16925 |         self._builtin_translations_installed = False
16926 |         self._lang_output_cache = {}
16927 |         self._lang_output_cache_dirty = False
16928 |
16929 |         # 1) Cache UI utama: paling penting untuk tombol, label, dan frasa laporan.
16930 |         try:
16931 |             lang_path = self._find_cache_file("lang_cache_ui.json")
16932 |             if os.path.exists(lang_path):
16933 |                 with open(lang_path, "r", encoding="utf-8") as f:
16934 |                     self._merge_translation_mapping(json.load(f))
16935 |         except Exception as e:
16936 |             print(f"Peringatan: Gagal memuat lang_cache_ui.json -> {e}")
16937 |
16938 |         # 2) Cache output penuh: hash sha256(f"{lang}\\0{raw}") -> hasil terjemahan.
16939 |         try:
16940 |             output_path = self.get_output_lang_cache_path()
16941 |             if os.path.exists(output_path):
16942 |                 with open(output_path, "r", encoding="utf-8") as f:
16943 |                     data = json.load(f)

```

```

16944 |         if isinstance(data, dict):
16945 |             self._lang_output_cache = {str(k): str(v) for k, v in data.items() if isinstance(v, str)}
16946 |     except Exception as e:
16947 |         print(f"Peringatan: Gagal memuat lang_output_cache.json -> {e}")
16948 |
16949 |     self.install_builtin_translations()
16950 |
16951 | def install_builtin_translations(self):
16952 |     """Fallback kecil saja; kamus besar dipindah ke lang_cache_ui.json."""
16953 |     if getattr(self, "_builtin_translations_installed", False):
16954 |         return
16955 |     if not hasattr(self, "translations") or self.translations is None:
16956 |         self.translations = {}
16957 |
16958 |     fallback = {
16959 |         "id": {"en": "en", "ar": "ar"},
16960 |         "en": {"en": "en", "ar": "en"},
16961 |         "ar": {"en": "ar", "ar": "ar"},
16962 |         "KHGT ENGINE": {"en": "KHGT ENGINE", "ar": "كحمت كحمت"},
16963 |         "Peta Langit Dunia": {"en": "World Sky Map", "ar": "خريطة العالم"},
16964 |         "Membuka Peta Langit Dunia melalui browser default...": {
16965 |             "en": "Opening World Sky Map in the default browser...",
16966 |             "ar": "سيُفتح خريطة العالم في متصفح التصفح الافتراضي...",
16967 |         },
16968 |         "► PROSES DATA": {"en": "► PROCESS DATA", "ar": "► تانايبل ةجلاعم"},
16969 |         "Kalkulasi Selesai": {"en": "Calculation Complete", "ar": "باسجل لم تكا"},
16970 |         "Error Kalkulasi": {"en": "Calculation Error", "ar": "باسجل يف أطح"},
16971 |         "Sistem Siap": {"en": "System Ready", "ar": "زه اء ماطنل"},
16972 |         "Mohon tunggu": {"en": "Please wait", "ar": "راطنل ال يءرى"},
16973 |         "Memproses data astrometri": {"en": "Processing astrometric data", "ar": "ءكللءل تانايبل ةجلاعم ءراج"},
16974 |     }
16975 |     # Kamus inti mandiri untuk menu, label utama, judul, dan pesan output.
16976 |     # Cache JSON eksternal tetap didukung dan tidak dihapus.
16977 |     #
16978 |     # Panduan pada halaman Release dirender langsung ke Textbox (bukan widget
16979 |     # CTKLabel/CTKButton), sehingga frasa-frasa ini harus tersedia di kamus
16980 |     # agar dapat diterjemahkan ketika bahasa EN/AR dipilih. URL sengaja
16981 |     # dipertahankan apa adanya.
16982 |     fallback.update({
16983 |         "DOWNLOAD BUKU KHGT TIMES": {
16984 |             "id": "DOWNLOAD BUKU KHGT TIMES",
16985 |             "en": "DOWNLOAD KHGT TIMES BOOK",
16986 |             "ar": "ءانء لك لءمءء",
16987 |         },
16988 |         "PYTHON MASTERY (TUTORIAL)": {
16989 |             "id": "PYTHON MASTERY (TUTORIAL)",
16990 |             "en": "PYTHON MASTERY (TUTORIAL)",
16991 |             "ar": "ءلملءء Python ناءءل",
16992 |         },
16993 |         "NAVIGASI MODUL": {
16994 |             "id": "NAVIGASI MODUL",
16995 |             "en": "MODULE NAVIGATION",
16996 |             "ar": "ءاءءءول ال نءب للءنءل",
16997 |         },
16998 |         "INPUT LOKASI": {
16999 |             "id": "INPUT LOKASI",
17000 |             "en": "LOCATION INPUT",
17001 |             "ar": "ءءءءول ال اءءل",
17002 |         },
17003 |         "EKSEKUSI DATA": {
17004 |             "id": "EKSEKUSI DATA",
17005 |             "en": "DATA EXECUTION",
17006 |             "ar": "ءانايبل ءففء",
17007 |         },
17008 |         "INTERAKSI GRAFIS": {
17009 |             "id": "INTERAKSI GRAFIS",
17010 |             "en": "GRAPHICAL INTERACTION",
17011 |             "ar": "ءمءسءل لء اءءل",
17012 |         },
17013 |         "SISTEM EXPORT": {
17014 |             "id": "SISTEM EXPORT",

```

```

17015 |         "en": "EXPORT SYSTEM",
17016 |         "ar": "ريدمتال ماطن",
17017 |     },
17018 |     "Silahkan download aplikasi KHGT Times melalui link: https://s.id/buku-khgttimes": {
17019 |         "id": "Silahkan download aplikasi KHGT Times melalui link: https://s.id/buku-khgttimes",
17020 |         "en": "Please download the KHGT Times application via: https://s.id/buku-khgttimes",
17021 |         "ar": "KHGT Times ربيع قيهبطت ليزنت ليحري: https://s.id/buku-khgttimes",
17022 |     },
17023 |     "Akses tutorial Python dasar melalui link: https://kasmui.cloud/python/": {
17024 |         "id": "Akses tutorial Python dasar melalui link: https://kasmui.cloud/python/",
17025 |         "en": "Access the basic Python tutorial via: https://kasmui.cloud/python/",
17026 |         "ar": "يساسالال Python ليلد لىل لوصول",
17027 |     },
17028 |     "Gunakan menu dropdown scrollable 'KHGT ENGINE' di sidebar kiri untuk beralih antar 41 modul perhitungan saintifik. Masing-masing modul akan menyembunyikan form input yang tidak relevan agar
17029 |     layar selalu rapi dan fokus.": {
17030 |         "id": "Gunakan menu dropdown scrollable 'KHGT ENGINE' di sidebar kiri untuk beralih antar 41 modul perhitungan saintifik. Masing-masing modul akan menyembunyikan form input yang tidak
17031 |     relevan agar layar selalu rapi dan fokus.",
17032 |         "en": "Use the scrollable 'KHGT ENGINE' dropdown menu in the left sidebar to switch between 41 scientific calculation modules. Each module hides irrelevant input forms so the screen remains
17033 |     clean and focused.",
17034 |         "ar": "مساها لىك يتح هلملا تاذ ريغ لادلا حادمان ءافخ لىل عذو لك لمعت .هيملعل اباسجلل عذو 41 نيب لقنلل رسيال يبنجال طيرشل ي ريرمئل ءلبال ءلدسمل 'KHGT ENGINE' مئاق مدختسا",
17035 |     },
17036 |     "Tersedia dua opsi. Anda bisa menyeleksi dari Dropdown Provinsi/Kota bawaan aplikasi, atau gunakan tombol '□ Deteksi Lokasi Otomatis' untuk memanggil API Geolocation IP agar sistem melacak
17037 |     titik GPS dan mendeteksi Timezone otomatis.": {
17038 |         "id": "Tersedia dua opsi. Anda bisa menyeleksi dari Dropdown Provinsi/Kota bawaan aplikasi, atau gunakan tombol '□ Deteksi Lokasi Otomatis' untuk memanggil API Geolocation IP agar sistem
17039 |     melacak titik GPS dan mendeteksi Timezone otomatis.",
17040 |         "en": "Two options are available. You can select a location from the built-in Province/City dropdown, or use the '□ Detect Automatic Location' button to call the IP Geolocation API so the
17041 |     system can track the GPS point and detect the timezone automatically.",
17042 |         "ar": "يتح IP ربع يفارغل عقومال ديدحت هجاو ءاغدتسال 'ا'ئئاقل عقومال فاشيكلا □ رز مادختسا و ، قيهبطتل يف عجمدمال ءلدسمل ءنيدمل/ءطفاحمل مئاق نم عقومال رايخا ليلكمي .نارايخ رفوتسي",
17043 |     },
17044 |     "Pada sebagian besar modul berbasis teks dan grafik, Anda cukup mengisi parameter lalu klik tombol biru besar '► PROSES DATA' di panel bawah. Tombol ini otomatis bersembunyi jika Anda sedang
17045 |     memasuki modul Live/Real-time.": {
17046 |         "id": "Pada sebagian besar modul berbasis teks dan grafik, Anda cukup mengisi parameter lalu klik tombol biru besar '► PROSES DATA' di panel bawah. Tombol ini otomatis bersembunyi jika Anda
17047 |     sedang memasuki modul Live/Real-time.",
17048 |         "en": "In most text- and graph-based modules, simply enter the parameters and click the large blue '► PROCESS DATA' button in the lower panel. This button automatically hides when you enter
17049 |     a Live/Real-time module.",
17050 |         "ar": "دنع 'ا'ئاقل رزلا اذه يفتحي .هيفلملا ءوللل يف 'تانايلا ءللاعم ►' ريبكلا قررال رزلا لىل رقلل م تاملعمل لادلا يوس ليلع ام .هينايلا موسرلا ومومئلا لىل ءدمتعمل تادجولا مطعم يف",
17051 |     },
17052 |     "Khusus pada canvas 3D Matplotlib dan Simulasi 3D Spasial, biasanya diri menggunakan klik Kiri Mouse (tahan dan geser) untuk Merotasi orientasi kamera/sudut pandang, serta Roda Mouse (Scroll)
17053 |     untuk melakukan perbesaran (Zoom In/Out).": {
17054 |         "id": "Khusus pada canvas 3D Matplotlib dan Simulasi 3D Spasial, biasanya diri menggunakan klik Kiri Mouse (tahan dan geser) untuk Merotasi orientasi kamera/sudut pandang, serta Roda Mouse
17055 |     (Scroll) untuk melakukan perbesaran (Zoom In/Out).",
17056 |         "en": "For the Matplotlib 3D canvas and Spatial 3D Simulation, use the Left Mouse Button (hold and drag) to rotate the camera orientation/viewpoint, and the Mouse Wheel (Scroll) to zoom in
17057 |     or out.",
17058 |         "ar": "، ريغمئلا و ريبكئل (ريرمئلا) ءرافلا ءلجعو ،ضعلال ءيواز/اريملال هاجا ريودتل (بسحاو طغما) رسيال ءرافلا رز مدختسا ،ءاعبال ءيئال ءافلا ءاكجمو ءاعبال ءيئال Matplotlib ءولل يف",
17059 |     },
17060 |     "Terdapat panel aksi (Action Bar) melayang di sudut kanan atas layar (atau di bawah tabel/canvas): \n □ TXT Menyimpan data tabel saat ini ke file Notepad.\n □ PNG Menyimpan canvas
17061 |     visual / laporan ke gambar PNG.\n □ PDF Mencetak dokumen laporan standar akademik fpdf.\n □ CSV Mengekstrak data komparasi 50 Tahun ke Excel.\n □ RILIS Tombol reset darurat kembali ke
17062 |     Dokumentasi.": {
17063 |         "id": "Terdapat panel aksi (Action Bar) melayang di sudut kanan atas layar (atau di bawah tabel/canvas): \n □ TXT Menyimpan data tabel saat ini ke file Notepad.\n □ PNG Menyimpan
17064 |     canvas visual / laporan ke gambar PNG.\n □ PDF Mencetak dokumen laporan standar akademik fpdf.\n □ CSV Mengekstrak data komparasi 50 Tahun ke Excel.\n □ RILIS Tombol reset darurat kembali ke
17065 |     Dokumentasi.",
17066 |         "en": "A floating Action Bar is available in the upper-right corner of the screen (or below a table/canvas): \n □ TXT Saves the current table data to a Notepad file.\n □ PNG Saves
17067 |     the visual canvas/report as a PNG image.\n □ PDF Prints a standard academic fpdf report.\n □ CSV Extracts 50-year comparison data to Excel.\n □ RELEASE Emergency reset button that returns to
17068 |     Documentation.",
17069 |         "ar": "ءولل طغح Notepad.\n □ PNG فلم يف هيلال لودجلا تاناي طغح \n □ TXT (ءولل/لودجلا لفسا و) مساهلا نم ينمئلا هيلولال ءيوازلا يف (Action Bar) مئاق ءاءارح ءول دجوت",
17070 |     },
17071 |     "عولل طغح \n □ PNG قيهبطتل لىل ءدوجل طراط طبع ءلاع رز [رادملا] Excel.\n □ CSV ءمل ءن 50 ءمل ءنراقمل تاناي جارختسا fpdf.\n □ PDF ريرقت ءعابط PNG.\n □ PDF ءرومك ريرقتل/ءيئرمل",
17072 |     },
17073 |     },
17074 |     fallback.update({'1' Kalender Hijriah Berjalan': {'id': '1' Kalender Hijriah Berjalan', 'en': '1' Current Hijri Calendar', 'ar': '1' يرحهل ميوئتل},
17075 |         '2' Kalender Masehi Berjalan': {'id': '2' Kalender Masehi Berjalan', 'en': '2' Current Gregorian Calendar', 'ar': '2' يداليمل ميوئتل},
17076 |         '3' Konversi Kalender': {'id': '3' Konversi Kalender', 'en': '3' Calendar Conversion', 'ar': '3' ميوئتل ليوت},
17077 |         '4' Visibilitas Hilal (KHGT)': {'id': '4' Visibilitas Hilal (KHGT)', 'en': '4' Crescent Visibility (KHGT)', 'ar': '4' لالهال ءيؤر (KHGT)},
17078 |         '5' Analisis Hilal Global': {'id': '5' Analisis Hilal Global', 'en': '5' Global Crescent Analysis', 'ar': '5' يملال لالهال ليولت},
17079 |         '6' Status Kriteria Batas': {'id': '6' Status Kriteria Batas', 'en': '6' Threshold Criteria Status', 'ar': '6' دودجل ريداعم ءلاح},
17080 |         '7' Kota Pertama KHGT (Mainland)': {'id': '7' Kota Pertama KHGT (Daratan Utama)',
17081 |             'en': '7' First KHGT City (Mainland)',
17082 |             'ar': '7' هسايئرلا هسايئرلا KHGT هنيدم لىل},
17083 |         '8' Crescent Visibility Map': {'id': '8' Peta Visibilitas Hilal', 'en': '8' Crescent Visibility Map', 'ar': '8' لالهال ءيؤر طيرخ},
17084 |     })

```

```
17064 | '9) Altitude Chart Analyzer': {'id': '9) Penganalisis Grafik Ketinggian', 'en': '9) Altitude Chart Analyzer', 'ar': '9) عافترا لاطح م للحم'},
17065 | '10) Komparasi 50 Tahun (Ramadhan)': {'id': '10) Komparasi 50 Tahun (Ramadan)', 'en': '10) 50-Year Comparison (Ramadan)', 'ar': '10) عنس 50 عناقم'},
17066 | '11) Tabel Tinggi Hilal 50 Thn (Ramadhan)': {'id': '11) Tabel Tinggi Hilal 50 Tahun (Ramadan)',
17067 | 'en': '11) 50-Year Crescent Altitude Table (Ramadan)',
17068 | 'ar': '11) عنس 50 لالهل عافترا لودج'},
17069 | '12) Tabel Elongasi Hilal 50 Thn (Ramadhan)': {'id': '12) Tabel Elongasi Hilal 50 Tahun (Ramadan)',
17070 | 'en': '12) 50-Year Crescent Elongation Table (Ramadan)',
17071 | 'ar': '12) عنس 50 لالهل علامتسا لودج'},
17072 | '13) Komparasi 50 Tahun (Syawal)': {'id': '13) Komparasi 50 Tahun (Syawal)', 'en': '13) 50-Year Comparison (Shawwal)', 'ar': '13) عنس 50 عناقم'},
17073 | '14) Tabel Tinggi Hilal 50 Thn (Syawal)': {'id': '14) Tabel Tinggi Hilal 50 Tahun (Syawal)',
17074 | 'en': '14) 50-Year Crescent Altitude Table (Shawwal)',
17075 | 'ar': '14) عنس 50 لالهل عافترا لودج'},
17076 | '15) Tabel Elongasi Hilal 50 Thn (Syawal)': {'id': '15) Tabel Elongasi Hilal 50 Tahun (Syawal)',
17077 | 'en': '15) 50-Year Crescent Elongation Table (Shawwal)',
17078 | 'ar': '15) عنس 50 لالهل علامتسا لودج'},
17079 | '16) Komparasi 50 Tahun (Zulhijah)': {'id': '16) Komparasi 50 Tahun (Zulhijah)',
17080 | 'en': '16) 50-Year Comparison (Dhu al-Hijjah)',
17081 | 'ar': '16) عنس 50 عناقم'},
17082 | '17) Tabel Tinggi Hilal 50 Thn (Zulhijah)': {'id': '17) Tabel Tinggi Hilal 50 Tahun (Zulhijah)',
17083 | 'en': '17) 50-Year Crescent Altitude Table (Dhu al-Hijjah)',
17084 | 'ar': '17) عنس 50 لالهل عافترا لودج'},
17085 | '18) Tabel Elongasi Hilal 50 Thn (Zulhijah)': {'id': '18) Tabel Elongasi Hilal 50 Tahun (Zulhijah)',
17086 | 'en': '18) 50-Year Crescent Elongation Table (Dhu al-Hijjah)',
17087 | 'ar': '18) عنس 50 لالهل علامتسا لودج'},
17088 | '19) Komparasi 50 Tahun (12 Bulan)': {'id': '19) Komparasi 50 Tahun (12 Bulan)',
17089 | 'en': '19) 50-Year Comparison (12 Months)',
17090 | 'ar': '19) عنس 50 عناقم'},
17091 | '20) Fase Bulan (Moonphase)': {'id': '20) Fase Bulan', 'en': '20) Moon Phases', 'ar': '20) رمقلا راطأ'},
17092 | '21) Moon Times': {'id': '21) Waktu Bulan', 'en': '21) Moon Times', 'ar': '21) رمقلا ااقوا'},
17093 | '22) Sun Times': {'id': '22) Waktu Matahari', 'en': '22) Sun Times', 'ar': '22) سمشلا ااقوا'},
17094 | '23) Sun Moon Ephemeris': {'id': '23) Ephemeris Matahari dan Bulan', 'en': '23) Sun and Moon Ephemeris', 'ar': '23) رمقلا و سمشلا ميوقت'},
17095 | '24) Sistem Sun Moon Earth': {'id': '24) Sistem Matahari-Bulan-Bumi', 'en': '24) Sun-Moon-Earth System', 'ar': '24) عخرالال و رمقلا و سمشلا ماطن'},
17096 | '25) Prayer Times': {'id': '25) Waktu Salat', 'en': '25) Prayer Times', 'ar': '25) ةالهل عناقوم'},
17097 | '26) Qiblah Direction & Times': {'id': '26) Arah dan Waktu Kiblat', 'en': '26) Qibla Direction and Times', 'ar': '26) ااهتاوقأو ةلبقلا ااجا'},
17098 | '27) Qibla Time (Rashdul Lokal)': {'id': '27) Waktu Kiblat (Rashdul Kiblat Lokal)',
17099 | 'en': '27) Qibla Time (Local Rashdul Kibla)',
17100 | 'ar': '27) ةلبقلا تقو'},
17101 | '28) Kalkulator Mizwala (Bayangan)': {'id': '28) Kalkulator Mizwala (Bayangan)', 'en': '28) Mizwala Calculator (Shadow)', 'ar': '28) ةلوزملا عساح'},
17102 | '29) Evaluasi Fajar Shadiq & SQM': {'id': '29) Evaluasi Fajar Sadik dan SQM', 'en': '29) True Dawn and SQM Evaluation', 'ar': '29) SQM و قداملا رفلل ميوقت'},
17103 | '30) Analisis Gerhana': {'id': '30) Analisis Gerhana', 'en': '30) Eclipse Analysis', 'ar': '30) فوسللا ليلحت'},
17104 | '31) Equinox & Solstice': {'id': '31) Ekuinoks dan Solstis', 'en': '31) Equinox and Solstice', 'ar': '31) بالقبالال و لادغالال'},
17105 | '32) Planetary Times': {'id': '32) Waktu Planet', 'en': '32) Planetary Times', 'ar': '32) بلاكلال ااقوا'},
17106 | '33) Generator Analemma Matahari': {'id': '33) Generator Analema Matahari', 'en': '33) Solar Analemma Generator', 'ar': '33) سمشلا اام لالان دلوم'},
17107 | '34) Prediktor Pasang Surut Gravitasi': {'id': '34) Prediktor Pasang Surut Gravitasi',
17108 | 'en': '34) Gravitational Tide Predictor',
17109 | 'ar': '34) بذجلا روجلل و دملل ببنم'},
17110 | '35) Astrofotografi & Kontras Visibilitas': {'id': '35) Astrofotografi dan Kontras Visibilitas',
17111 | 'en': '35) Astrophotography and Visibility Contrast',
17112 | 'ar': '35) ةقورلا بوابت و لفلل ريومبالا'},
17113 | '36) Live Animasi': {'id': '36) Animasi Langsung', 'en': '36) Live Animation', 'ar': '36) ةشرابملا ةلرتملا موسرلا'},
17114 | '37) Simulasi Ephemeris 3D': {'id': '37) Simulasi Ephemeris 3D', 'en': '37) 3D Ephemeris Simulation', 'ar': '37) داعبالا ةثالث لفلل ميوقتلا ةالاعم'},
17115 | '38) WinAI: Aplikasi AI Windows': {'id': '38) WinAI: Aplikasi AI Windows',
17116 | 'en': '38) WinAI: Windows AI Application',
17117 | 'ar': '38) WinAI: ءالذ قيبطت'},
17118 | '39) Peta Langit Dunia': {'id': '39) Peta Langit Dunia',
17119 | 'en': '39) World Sky Map',
17120 | 'ar': '39) ةقملالعل ءامسلل عطيرح'},
17121 | '40) Laporan Astronomi': {'id': '40) Laporan Astronomi',
17122 | 'en': '40) Astronomy Report',
17123 | 'ar': '40) لفلل ريرقتلا'},
17124 | '41) HIJRI_DB Auto-Builder (Admin)': {'id': '41) Pembuat Otomatis HIJRI_DB (Admin)',
17125 | 'en': '41) HIJRI_DB Auto-Builder (Admin)',
17126 | 'ar': '41) HIJRI_DB عناقلا ةلرتملا'},
17127 | 'KHGT ENGINE': {'id': 'MESIN KHGT', 'en': 'KHGT ENGINE', 'ar': 'KHGT لرحم'},
17128 | '► PROSES DATA': {'id': '► PROSES DATA', 'en': '► PROCESS DATA', 'ar': '► ةالاعم'},
17129 | 'Silakan klik tombol ► PROSES DATA untuk menampilkan output.',
17130 | 'en': 'Please click the ► PROCESS DATA button to display the output.',
17131 | 'ar': 'جقاعلل جرجل ءانا ببلل ةالاعم ► رز فلر رقتلا عجرى'},
17132 | 'PILIH KOTA (DEFAULT SEMARANG)': {'id': 'PILIH KOTA (BAWAAN: SEMARANG)', 'en': 'SELECT CITY (DEFAULT: SEMARANG)', 'ar': 'عنبدملا رتخا'},
17133 | '□ Deteksi Lokasi Otomatis': {'id': '□ Deteksi Lokasi Otomatis', 'en': '□ Detect Location Automatically', 'ar': 'ا عناقلا عقوملا فاشكلا □'},
17134 | 'PENGATURAN ALARM SALAT': {'id': 'PENGATURAN ALARM SALAT', 'en': 'PRAYER ALARM SETTINGS', 'ar': 'ةالعملل مبنم ءادادعل'}
```

```

17135 | 'Aktifkan Alarm': {'id': 'Aktifkan Alarm', 'en': 'Enable Alarm', 'ar': 'مبمنل ليغت'},
17136 | 'Pilih Audio Adzan': {'id': 'Pilih Audio Azan', 'en': 'Select Adhan Audio', 'ar': 'نادأل توص رايتخ'},
17137 | 'TANGGAL OBSERVASI': {'id': 'TANGGAL OBSERVASI', 'en': 'OBSERVATION DATE', 'ar': 'دصرلا خيرات'},
17138 | 'TANGGAL OBSERVASI (SAAT MAGHRIB)': {'id': 'TANGGAL OBSERVASI (SAAT MAGRIB)', 'en': 'OBSERVATION DATE (AT SUNSET)', 'ar': 'دصرلا دنغ (بورغلا دنغ)'},
17139 | 'TAHUN OBSERVASI': {'id': 'TAHUN OBSERVASI', 'en': 'OBSERVATION YEAR', 'ar': 'دصرلا ءنس'},
17140 | 'BULAN OBSERVASI': {'id': 'BULAN OBSERVASI', 'en': 'OBSERVATION MONTH', 'ar': 'دصرلا رهش'},
17141 | 'KOORDINAT LOKASI': {'id': 'KOORDINAT LOKASI', 'en': 'LOCATION COORDINATES', 'ar': 'عقوملا نايتادج'},
17142 | 'KOORDINAT OBSERVASI': {'id': 'KOORDINAT OBSERVASI', 'en': 'OBSERVATION COORDINATES', 'ar': 'دصرلا نايتادج'},
17143 | 'KOORDINAT PANTAI': {'id': 'KOORDINAT PANTAI', 'en': 'COASTAL COORDINATES', 'ar': 'لجاسلا نايتادج'},
17144 | 'KOORDINAT PENGAMAT': {'id': 'KOORDINAT PENGAMAT', 'en': 'OBSERVER COORDINATES', 'ar': 'دصرلا نايتادج'},
17145 | 'LOKASI OBSERVASI': {'id': 'LOKASI OBSERVASI', 'en': 'OBSERVATION LOCATION', 'ar': 'دصرلا عقوم'},
17146 | 'WAKTU OBSERVASI': {'id': 'WAKTU OBSERVASI', 'en': 'OBSERVATION TIME', 'ar': 'دصرلا حقو'},
17147 | 'TAHUN MASEHI': {'id': 'TAHUN MASEHI', 'en': 'GREGORIAN YEAR', 'ar': 'فيداليملا ءنسل'},
17148 | 'TANGGAL MASEHI': {'id': 'TANGGAL MASEHI', 'en': 'GREGORIAN DATE', 'ar': 'فيداليملا خيراتلا'},
17149 | 'PENGATURAN KALENDER': {'id': 'PENGATURAN KALENDER', 'en': 'CALENDAR SETTINGS', 'ar': 'ميوقتلا تادادع'},
17150 | 'PENGATURAN KAL. MASEHI': {'id': 'PENGATURAN KALENDER MASEHI', 'en': 'GREGORIAN CALENDAR SETTINGS', 'ar': 'فيداليملا ميوقتلا تادادع'},
17151 | 'Bulan': {'id': 'Bulan', 'en': 'Month', 'ar': 'رهشل'},
17152 | 'Tahun (H)': {'id': 'Tahun (H)', 'en': 'Year (AH)', 'ar': 'ءنسل (ه)'},
17153 | 'Tahun (M)': {'id': 'Tahun (M)', 'en': 'Year (CE)', 'ar': 'ءنسل (م)'},
17154 | 'Terapkan Pencarian': {'id': 'Terapkan Pencarian', 'en': 'Apply Search', 'ar': 'تجبل قيبط'},
17155 | 'Reset ke Bulan Ini': {'id': 'Atur Ulang ke Bulan Ini', 'en': 'Reset to Current Month', 'ar': 'لجلا رهلل دل طبملا ءداع'},
17156 | 'Memuat Ephemeris...': {'id': 'Memuat Ephemeris...', 'en': 'Loading Ephemeris...', 'ar': 'كلفلل ميوقتلا ليمت راج'},
17157 | 'Sistem Siap': {'id': 'Sistem Siap', 'en': 'System Ready', 'ar': 'زهاج ماملل'},
17158 | 'Alarm Salat: Memuat Jadwal...': {'id': 'Alarm Salat: Memuat Jadwal...', 'en': 'Prayer Alarm: Loading Schedule...', 'ar': 'لودجلا ليمت راج: ءاليملا مبنم'},
17159 | 'Cuaca: Menunggu deteksi...': {'id': 'Cuaca: Menunggu deteksi...', 'en': 'Weather: Waiting for detection...', 'ar': 'عقوملا ديدج راطنلا في سقطلا'},
17160 | 'Cuaca: Menunggu...': {'id': 'Cuaca: Menunggu...', 'en': 'Weather: Waiting...', 'ar': 'راطنلا في سقطلا'},
17161 | 'Laporan Analisis Hilal & KHGT': {'id': 'Laporan Analisis Hilal dan KHGT', 'en': 'Crescent and KHGT Analysis Report', 'ar': 'KHGT ولالهلا ليلت ريرقت'},
17162 | 'Simpan Data ke TXT': {'id': 'Simpan Data ke TXT', 'en': 'Save Data to TXT', 'ar': 'TXT ءغيمب تانايللا طفح'},
17163 | 'Simpan Laporan sbg PNG': {'id': 'Simpan Laporan sebagai PNG', 'en': 'Save Report as PNG', 'ar': 'PNG ءغيمب ريرقتلا طفح'},
17164 | 'Simpan sebagai PDF': {'id': 'Simpan sebagai PDF', 'en': 'Save as PDF', 'ar': 'PDF ءغيمب طفح'},
17165 | 'Simpan ke CSV (Excel / SPSS)': {'id': 'Simpan ke CSV (Excel/SPSS)', 'en': 'Save to CSV (Excel/SPSS)', 'ar': 'CSV ءغيمب طفح (Excel/SPSS)'},
17166 | 'Kembali ke Beranda / Info Rilis': {'id': 'Kembali ke Beranda / Informasi Rilis', 'en': 'Return to Home / Release Information', 'ar': 'رادمرلا تامولم / فيسئئرلا دل دوعلا'},
17167 | 'Menunggu input dari pengguna...': {'id': 'Menunggu input dari pengguna...', 'en': 'Waiting for user input...', 'ar': 'مدخسملا لاخد راطنلا في'},
17170 | 'PENGATURAN WAKTU & ELEVASI': {'id': 'PENGATURAN WAKTU DAN ELEVASI', 'en': 'TIME AND ELEVATION SETTINGS', 'ar': 'ءاعئرلا او حقولا تادادع'},
17171 | 'PARAMETER ATMOSFER': {'id': 'PARAMETER ATMOSFER', 'en': 'ATMOSPHERIC PARAMETERS', 'ar': 'قوتولا فالغل تالماعم'},
17172 | 'PARAMETER TONGKAT & WAKTU': {'id': 'PARAMETER TONGKAT DAN WAKTU', 'en': 'ROD AND TIME PARAMETERS', 'ar': 'تقولل املغل تالماعم'},
17173 | 'PENGATURAN ANALEMMA': {'id': 'PENGATURAN ANALEMA', 'en': 'ANALEMMA SETTINGS', 'ar': 'لمانال تادادع'},
17174 | 'TAHUN & JAM ANALISIS': {'id': 'TAHUN DAN JAM ANALISIS', 'en': 'ANALYSIS YEAR AND TIME', 'ar': 'ليلجتلل حقوو ءنس'},
17175 | 'PREDIKTOR OKULTASI PLANET': {'id': 'PREDIKTOR OKULTASI PLANET', 'en': 'PLANETARY OCCULTATION PREDICTOR', 'ar': 'بلكاوكل باجتا بئتم'},
17176 | 'Cari Okultasi Bulan Ini': {'id': 'Cari Okultasi Bulan Ini', 'en': 'Find Occultations This Month', 'ar': 'رهشل اذه باجتا حال تالاج نع تجبلا'},
17177 | 'PROSES ANALEMMA': {'id': 'PROSES ANALEMA', 'en': 'PROCESS ANALEMMA', 'ar': 'مللانال ءجلاعم'},
17178 | 'Export Gambar HD': {'id': 'Ekspor Gambar HD', 'en': 'Export HD Image', 'ar': 'ءقءلا ءقلاع ءومس ريدمت'},
17179 | 'Cetak PDF': {'id': 'Cetak PDF', 'en': 'Print PDF', 'ar': 'ءعابط PDF'},
17180 | 'Log Teks': {'id': 'Log Teks', 'en': 'Text Log', 'ar': 'يغن لجس'},
17181 | 'Salin': {'id': 'Salin', 'en': 'Copy', 'ar': 'ءسن'},
17182 | 'Share': {'id': 'Bagikan', 'en': 'Share', 'ar': 'ءكراشم'},
17183 | 'Best Time': {'id': 'Waktu Terbaik', 'en': 'Best Time', 'ar': 'تقو لءقا'},
17184 | 'Location': {'id': 'Lokasi', 'en': 'Location', 'ar': 'عقوملا'},
17185 | 'Object': {'id': 'Objek', 'en': 'Object', 'ar': 'مسجل'},
17186 | 'Time Ref': {'id': 'Referensi Waktu', 'en': 'Time Reference', 'ar': 'تقولل عءرم'},
17187 | 'Sunset': {'id': 'Matahari Terbenam', 'en': 'Sunset', 'ar': 'سمشل بورغ'},
17188 | 'Moonset': {'id': 'Bulan Terbenam', 'en': 'Moonset', 'ar': 'رمقلا بورغ'},
17189 | 'New Crescent (Evening)': {'id': 'Hilal Baru (Petang)', 'en': 'New Crescent (Evening)', 'ar': 'ءيدجل لالهلا (ءاسم)'},
17190 | 'Old Crescent (Morning)': {'id': 'Sabit Tua (Pagi)', 'en': 'Old Crescent (Morning)', 'ar': 'ميدقلا لالهلا (ءاسب)'},
17191 | 'Sun is at Qiblah direction': {'id': 'Matahari berada pada arah kiblat', 'en': 'The Sun is in the Qibla direction', 'ar': 'ءلبقلا هاجا في سمشل'},
17192 | 'Sun's shadow is at Qiblah': {'id': 'Bayangan Matahari berada pada arah kiblat', 'en': 'The Sun's shadow is in the Qibla direction', 'ar': 'ءلبقلا هاجا في سمشل ل'},
17193 | 'Masehi → Hijriah (KHGT)': {'id': 'Masehi → Hijriah (KHGT)', 'en': 'Gregorian → Hijri (KHGT)', 'ar': 'KHGT يرءه → فيداليم'},
17196 | 'Hijriah (KHGT) → Masehi': {'id': 'Hijriah (KHGT) → Masehi', 'en': 'Hijri (KHGT) → Gregorian', 'ar': 'KHGT يرءه'},
17197 | 'Masehi → Hijriah (Urfi)': {'id': 'Masehi → Hijriah (Urfi)', 'en': 'Gregorian → Hijri (Tabular)', 'ar': 'KHGT يرءه → فيداليم'},
17198 | 'Hijriah (Urfi) → Masehi': {'id': 'Hijriah (Urfi) → Masehi', 'en': 'Hijri (Tabular) → Gregorian', 'ar': 'KHGT يرءه → فيداليم'},
17199 | 'Kalkulasi Selesai': {'id': 'Kalkulasi Selesai', 'en': 'Calculation Complete', 'ar': 'باسجل لمتكا'},
17200 | 'Error Kalkulasi': {'id': 'Kesalahan Kalkulasi', 'en': 'Calculation Error', 'ar': 'باسجلل اءح'},
17201 | 'Mohon tunggu': {'id': 'Mohon tunggu', 'en': 'Please wait', 'ar': 'راطنلا لءري'},
17202 | 'Memproses data astrometri': {'id': 'Memproses data astrometri', 'en': 'Processing astrometric data', 'ar': 'ءقءلا تانايللا ءجلاعم راج'},
17203 | 'Crescent Analysis & KHGT': {'id': 'Analisis Hilal dan KHGT', 'en': 'Crescent Analysis and KHGT', 'ar': 'KHGT ولالهلا ليلجتل'},
17204 | 'Crescent Visibility Map Scanner': {'id': 'Pemindai Peta Visibilitas Hilal', 'en': 'Crescent Visibility Map Scanner', 'ar': 'لالهلا ءقور ءطيرء ءسام'},
17205 | 'Global Hilal Visibility Analyzer': {'id': 'Penganalisis Visibilitas Hilal Global', 'ar': 'لالهلا ءقور ءطيرء ءسام'}

```



```

17206 |         'en': 'Global Crescent Visibility Analyzer',
17207 |         'ar': 'محلّال الهلال وقوسر للحج',
17208 | 'Moonphase Cycle Data': {'id': 'Data Siklus Fase Bulan', 'en': 'Moon Phase Cycle Data', 'ar': 'رمّقل راولأ ةرود ثانايب'},
17209 | 'Sun and Moon Ephemeris Data': {'id': 'Data Ephemeris Matahari dan Bulan', 'en': 'Sun and Moon Ephemeris Data', 'ar': 'يكلفل رملالو سمشلا ميوقت ثانايب'},
17210 | 'Qiblah Direction Calculator': {'id': 'Kalkulator Arah Kiblat', 'en': 'Qibla Direction Calculator', 'ar': 'ةلبقلا اجاتا ةسباح'},
17211 | 'Moon Rise, Transit & Set': {'id': 'Waktu Terbit, Transit, dan Terbenam Bulan', 'en': 'Moonrise, Transit and Moonset', 'ar': 'مبورغو هروبغو رملال قورش'},
17212 | 'Sun Rise, Transit & Set': {'id': 'Waktu Terbit, Transit, dan Terbenam Matahari', 'en': 'Sunrise, Transit and Sunset', 'ar': 'امبورغو اهروبغو سمشلا قورش'},
17213 | 'Prayer Times Calculator': {'id': 'Kalkulator Waktu Salat', 'en': 'Prayer Times Calculator', 'ar': 'ةالملا تحقاوم ةسباح'},
17214 | 'Calendar Conversion': {'id': 'Konversi Kalender', 'en': 'Calendar Conversion', 'ar': 'مميوقتلاليوحت'},
17215 | 'Local Qibla Time Calculator': {'id': 'Kalkulator Waktu Kiblat Lokal', 'en': 'Local Qibla Time Calculator', 'ar': 'يلحمل ةلبقلا تقو ةسباح'},
17216 | 'Precision Eclipse Calculator': {'id': 'Kalkulator Gerhana Presisi', 'en': 'Precision Eclipse Calculator', 'ar': 'ةقيقدل فوسجل او فوسكل ةسباح'},
17217 | 'Live Celestial Object Tracker': {'id': 'Pelacak Langsung Benda Langit', 'en': 'Live Celestial Object Tracker', 'ar': 'رشابملا ةيوامسلا مارجلال عبتتم'},
17218 | '3D Geocentric Space System': {'id': 'Sistem Ruang Geosentrik 3D', 'en': '3D Geocentric Space System', 'ar': 'داغبالا يثالث يهرا يزكرم يثايف ماظن'},
17219 | 'Celestial Altitude Graphics': {'id': 'Grafik Ketinggian Benda Langit', 'en': 'Celestial Altitude Graphics', 'ar': 'ةيوامسلا مارجلال عاقترا موسر'},
17220 | 'KHGT first Point (Mainland)': {'id': 'Titik Pertama KHGT (Daratan Utama)', 'en': 'First KHGT Point (Mainland)', 'ar': 'ةيسيئرلا ةسبايلا KHGT ةطقن لوا'},
17221 | 'Equinox & Solstice Calculator': {'id': 'Kalkulator Ekuinoks dan Solstis', 'en': 'Equinox and Solstice Calculator', 'ar': 'بالقنالو لادتعال ةسباح'},
17222 | 'Rise, Transit & Set Planet': {'id': 'Waktu Terbit, Transit, dan Terbenam Planet', 'en': 'Planet Rise, Transit and Set', 'ar': 'امبورغو اهروبغو بكاوكلل قورش'},
17223 | '3D Sun & Moon Ephemeris Simulation': {'id': 'Simulasi Ephemeris Matahari dan Bulan 3D',
17224 |     'en': '3D Sun and Moon Ephemeris Simulation',
17225 |     'ar': 'داغبالا ةيثالث رملالو سمشلا ميوقت ةالاحم'},
17226 | 'Kalkulator Bayangan Tongkat Istiwa (Mizwala)': {'id': 'Kalkulator Bayangan Tongkat Istiwa (Mizwala)',
17227 |     'en': 'Istiwa Rod Shadow Calculator (Mizwala)',
17228 |     'ar': 'ةلوزملا ءاوتسالا امع لظ ةسباح'},
17229 | 'Prediktor Pasang Surut Gravitasi': {'id': 'Prediktor Pasang Surut Gravitasi', 'en': 'Gravitational Tide Predictor', 'ar': 'يبذلج رزلالو دمل ثبنتم'},
17230 | 'Evaluasi Fajar Shadiq & SQM Teoritis': {'id': 'Evaluasi Fajar Sadik dan SQM Teoretis',
17231 |     'en': 'True Dawn and Theoretical SQM Evaluation',
17232 |     'ar': 'ي رطقنل SQM قداملا رفل ميوقت'},
17233 | 'Generator Analemma Matahari': {'id': 'Generator Analema Matahari', 'en': 'Solar Analemma Generator', 'ar': 'سمشلا ا.م.ل.ال دلوم'}}
17234 |     fallback.update({
17235 |         "Klik PROSES DATA untuk render grafik": {
17236 |             "id": "Klik PROSES DATA untuk merender grafik",
17237 |             "en": "Click PROCESS DATA to render the chart",
17238 |             "ar": "ططحمل مسرل ثانايبلا ةللام يلغ رقلان"
17239 |         }
17240 |     })
17241 |
17242 |     fallback.update({'1 Waktu Maghrib (Sunset)': {'id': '1 Waktu Magrib (Matahari Terbenam)', 'en': '1) Sunset Time', 'ar': '1) سمشلا بورغ تقو'},
17243 | '2 Waktu Tertentu (Kustom)': {'id': '2 Waktu Tertentu (Kustom)', 'en': '2) Specific Time (Custom)', 'ar': '2) ددجم تقو (امهيمج)'},
17244 | '3 Waktu Live (Sekarang)': {'id': '3 Waktu Langsung (Sekarang)', 'en': '3) Live Time (Now)', 'ar': '3) زمانا رشابملا تقولا'},
17245 | 'AEROSPACE PRAYER TIMES CALCULATOR': {'id': 'KALKULATOR WAKTU SALAT PENERBANGAN',
17246 |     'en': 'AEROSPACE PRAYER TIMES CALCULATOR',
17247 |     'ar': 'اناريطلا ءانثا ةالملا تحقاوم ةسباح'},
17248 | 'ALASAN: -': {'id': 'ALASAN: -', 'en': 'REASON: -', 'ar': 'ببسلا: -'},
17249 | 'Berikutnya ►': {'id': 'Berikutnya ►', 'en': 'Next ►', 'ar': 'يلااتل ►'},
17250 | 'Bulan': {'id': 'Bulan', 'en': 'Month', 'ar': 'رمشلا'},
17251 | 'Cari Sunset': {'id': 'Cari Waktu Matahari Terbenam', 'en': 'Find Sunset', 'ar': 'مبورغل تقو نع تحبلال'},
17252 | 'Cari Waktu Bintang di Kiblat': {'id': 'Cari Waktu Bintang pada Arah Kiblat',
17253 |     'en': 'Find Star Time in the Qibla Direction',
17254 |     'ar': 'ةلبقلا اجاتا يف مچنل تقو نع تحبلال'},
17255 | 'DEPARTURE (KEBERANGKATAN)': {'id': 'KEBERANGKATAN', 'en': 'DEPARTURE', 'ar': 'ةرداغملا'},
17256 | 'DESTINATION & AIRCRAFT': {'id': 'TUJUAN DAN PESAWAT', 'en': 'DESTINATION AND AIRCRAFT', 'ar': 'ةرئاطلاو ههجلال'},
17257 | 'GENERATE KURVA ALASKA': {'id': 'BUAT KURVA ALASKA', 'en': 'GENERATE ALASKA CURVE', 'ar': 'النسالا ينحنم ءاشن'},
17258 | 'GRAFIK HARIAN (24 JAM)': {'id': 'GRAFIK HARIAN (24 JAM)', 'en': 'DAILY CHART (24 HOURS)', 'ar': 'ةعاس 24 يمويلا ططحمل'},
17259 | 'Hitung Jendela Astrofotografi': {'id': 'Hitung Jendela Astrofotografi', 'en': 'Calculate Astrophotography Window', 'ar': 'يكلفل ريوصلا ةذفان باسج'},
17260 | 'KOREKSI HARI (IKHTIYAT)': {'id': 'KOREKSI HARI (IKHTIYAT)', 'en': 'DAY CORRECTION (PRECAUTION)', 'ar': 'طايئحال) مويل حيصت'},
17261 | 'Kalkulasi Dinamis berbasis Rute Great Circle, Kecepatan, dan Ketinggian': {'id': 'Kalkulasi Dinamis Berbasis Rute Lingkaran Besar, Kecepatan, dan Ketinggian',
17262 |     'en': 'Dynamic Calculation Based on Great-Circle Route, Speed, and Altitude',
17263 |     'ar': 'عافترا لالو عرسلالو صمطغل ةرئادل راسم يلغ ينيم يليمانيدي باسج'},
17264 | 'Kirim Pertanyaan': {'id': 'Kirim Pertanyaan', 'en': 'Send Question', 'ar': 'لالؤسل لاسرا'},
17265 | 'Klik & Drag layar untuk rotasi.\nScroll/Mousewheel untuk Zoom.': {'id': 'Klik dan seret layar untuk rotasi.\nGulir/roda tetikus untuk memperbesar.',
17266 |     'en': 'Click and drag to rotate.\nScroll/mouse wheel to zoom.',
17267 |     'ar': 'ريبكتلل ةرافلا ةلج مدختسلا ريودتلل بحساو رقلان'},
17268 | 'Legenda': {'id': 'Legenda', 'en': 'Legend', 'ar': 'مسرلا جاتقم'},
17269 | 'MEMUAT...': {'id': 'MEMUAT...', 'en': 'LOADING...', 'ar': 'للمحتال مراح'},
17270 | 'Matikan': {'id': 'Matikan', 'en': 'Turn Off', 'ar': 'فاقيل'},
17271 | 'Memeriksa sistem dan memuat data API...': {'id': 'Memeriksa sistem dan memuat data API...',
17272 |     'en': 'Checking the system and loading API data...',
17273 |     'ar': 'ثانايب للمحتو ماظنل صيغ مراح'},
17274 | 'Memuat data...': {'id': 'Memuat data...', 'en': 'Loading data...', 'ar': 'ثانايبلا للمحت مراح'},
17275 | 'Memuat...': {'id': 'Memuat...', 'en': 'Loading...', 'ar': 'للمحتال مراح'},
17276 | 'Mencari fase bulan': {'id': 'Mencari fase Bulan', 'en': 'Searching for lunar phase', 'ar': 'رمّقل روط نع تحبلال مراح'},

```

```

17277 | 'Menganalisis...': {'id': 'Menganalisis...', 'en': 'Analyzing...', 'ar': 'ليلحل...'},
17278 | 'Menghitung Ephemeris': {'id': 'Menghitung Ephemeris', 'en': 'Calculating Ephemeris', 'ar': 'يكلفل ميوقتلا باسح'},
17279 | 'Mengkalkulasi parameter astronomis...': {'id': 'Menghitung parameter astronomis...',
17280 |     'en': 'Calculating astronomical parameters...',
17281 |     'ar': 'يكلفل تالماعمل باسح'},
17282 | 'Mengkalkulasi...': {'id': 'Menghitung...', 'en': 'Calculating...', 'ar': 'يكلفل باسح'},
17283 | 'PILIH TAHUN GERHANA': {'id': 'PILIH TAHUN GERHANA', 'en': 'SELECT ECLIPSE YEAR', 'ar': 'فوسحل وا فوسكل ؤس ربح'},
17284 | 'SET TANGGAL SIMULASI 3D': {'id': 'ATUR TANGGAL SIMULASI 3D', 'en': 'SET 3D SIMULATION DATE', 'ar': 'داعبال ةثالث ةاكاحملا خيرات نييعت'},
17285 | 'SIMULASI ARAH BAYANGAN MATAHARI': {'id': 'SIMULASI ARAH BAYANGAN MATAHARI', 'en': 'SUN SHADOW DIRECTION SIMULATION', 'ar': 'سمشل ل ط هاجت ةاكاحم'},
17286 | 'SIMULATOR PARAMETER (MANUAL)': {'id': 'SIMULATOR PARAMETER (MANUAL)', 'en': 'PARAMETER SIMULATOR (MANUAL)', 'ar': 'يودي تالماعمل وكاحم'},
17287 | 'STATUS: -': {'id': 'STATUS: -', 'en': 'STATUS: -', 'ar': 'هالاح: -'},
17288 | 'Snooze 10m': {'id': 'Tunda 10 menit', 'en': 'Snooze 10 min', 'ar': 'قؤاد 10 ؤوفع'},
17289 | 'Snooze 5m': {'id': 'Tunda 5 menit', 'en': 'Snooze 5 min', 'ar': 'قؤاد 5 ؤوفع'},
17290 | 'Spesifikasi Penerbangan': {'id': 'Spesifikasi Penerbangan', 'en': 'Flight Specifications', 'ar': 'هالجرلا تافصاوم'},
17291 | 'Status: Inisialisasi...': {'id': 'Status: Inisialisasi...', 'en': 'Status: Initializing...', 'ar': 'هالاح: ةثيئل...'},
17292 | 'TANGGAL PENCARIAN': {'id': 'TANGGAL PENCARIAN', 'en': 'SEARCH DATE', 'ar': 'تجبال خيرات'},
17293 | 'TANGGAL REFERENSI (UTC)': {'id': 'TANGGAL REFERENSI (UTC)', 'en': 'REFERENCE DATE (UTC)', 'ar': 'هالجرملا خيراتلا (UTC)'},
17294 | 'TREN HIJRIAH (RAJAB - ZULHIJAH)': {'id': 'TREN HIJRIAH (RAJAB-ZULHIJAH)', 'en': 'HIJRI TREND (RAJAB-DHU AL-HIJJAH)', 'ar': 'هالجرملا هالجال'},
17295 | 'Tahun Hijriah': {'id': 'Tahun Hijriah', 'en': 'Hijri Year', 'ar': 'هالجرملا ؤسئل'},
17296 | 'Tanggal (YYYY-MM-DD)': {'id': 'Tanggal (TTTT-BB-HH)', 'en': 'Date (YYYY-MM-DD)', 'ar': 'خيراتلا (YYYY-MM-DD)'},
17297 | 'Terapkan Waktu Kustom': {'id': 'Terapkan Waktu Kustom', 'en': 'Apply Custom Time', 'ar': 'مهمجرملا تقولا قيبط'},
17298 | 'WAKTU AKHIR (END)': {'id': 'WAKTU AKHIR', 'en': 'END TIME', 'ar': 'هالاحل تقو'},
17299 | 'WAKTU AWAL (START)': {'id': 'WAKTU AWAL', 'en': 'START TIME', 'ar': 'هالاحل تقو'},
17300 | 'WAKTU SIMULASI 2D': {'id': 'WAKTU SIMULASI 2D', 'en': '2D SIMULATION TIME', 'ar': 'داعبال ةثالث ةاكاحملا تقو'},
17301 | 'WINAI ASSISTANT': {'id': 'ASISTEN WINAI', 'en': 'WINAI ASSISTANT', 'ar': 'دعاسم WINAI'},
17302 | 'Waktu Berangkat (Lokal)': {'id': 'Waktu Berangkat (Lokal)', 'en': 'Departure Time (Local)', 'ar': 'هالجرملا تقو'},
17303 | 'Waktu Lokal: ---:---:': {'id': 'Waktu Lokal: ---:---:', 'en': 'Local Time: ---:---:', 'ar': 'يلجرملا تقولا: ---:---:'},
17304 | 'Countdown Ijtimak': {'id': 'Hitung Mundur Ijtimak', 'en': 'Conjunction Countdown', 'ar': 'نارتقالل يلزاتلا دغل'},
17305 | 'Sedang Memuat Mesin Ephemeris NASA, harap tunggu...': {'id': 'Sedang Memuat Mesin Ephemeris NASA, harap tunggu...',
17306 |     'en': 'Loading the NASA Ephemeris Engine, please wait...',
17307 |     'ar': 'رابطناللا يجرى اسائل يكلفل ميوقتلا لرحم ليحج رراج'},
17308 | 'HITUNG JADWAL SELAMA PENERBANGAN': {'id': 'HITUNG JADWAL SELAMA PENERBANGAN', 'en': 'CALCULATE IN-FLIGHT SCHEDULE', 'ar': 'هالجرلا ءانت لودجل باسح'},
17309 | 'Play / Pause Animasi': {'id': 'Putar / Jeda Animasi', 'en': 'Play / Pause Animation', 'ar': 'موسرلل تقوم فاقي / ليغثت'},
17310 | 'TERAPKAN TANGGAL': {'id': 'TERAPKAN TANGGAL', 'en': 'APPLY DATE', 'ar': 'خيراتلا قيبط'},
17311 | 'Sebelumnya': {'id': 'Sebelumnya', 'en': 'Previous', 'ar': 'قباسلا'},
17312 | '* MATAHARI': {'id': '* MATAHARI', 'en': '* SUN', 'ar': 'سمشل'},
17313 | 'GENERATOR DATABASE KHGT': {'id': 'GENERATOR BASIS DATA KHGT', 'en': 'KHGT DATABASE GENERATOR', 'ar': 'KHGT تاناي ةدعاق دلوم'},
17314 | 'Shalat Penerbangan (In-Flight)': {'id': 'Salat dalam Penerbangan', 'en': 'In-Flight Prayer', 'ar': 'نارتقالل ءانت هالاحل'},
17315 | 'Set ke Waktu Sunset': {'id': 'Atur ke Waktu Matahari Terbenam', 'en': 'Set to Sunset Time', 'ar': 'بورغل تقو دغ طبعمل'},
17316 | '3D GEOCENTRIC VIEW': {'id': 'TAMPILAN GEOSENTRIK 3D', 'en': '3D GEOCENTRIC VIEW', 'ar': 'داعبال ةثالث يجرى يزكرم مرع'},
17317 | 'CELESTIAL QIBLAH (BINTANG)': {'id': 'KIBLAT LANGIT (BINTANG)', 'en': 'CELESTIAL QIBLA (STAR)', 'ar': 'هالجرملا هالقول'},
17318 | 'Akses Kiblat Online': {'id': 'Akses Kiblat Online', 'en': 'Online Qibla Access', 'ar': 'نارتقالل ربع هالقول دل لومول'},
17319 | 'Tracker': {'id': 'Pelacak', 'en': 'Tracker', 'ar': 'لغثتلم'},
17320 | 'Simulasi Visual Gerhana': {'id': 'Simulasi Visual Gerhana', 'en': 'Visual Eclipse Simulation', 'ar': 'فوسحل او فوسكلل ؤيرب ةاكاحم'},
17321 | 'ECLIPSE 2026': {'id': 'GERHANA 2026', 'en': 'ECLIPSE 2026', 'ar': 'فوسحل او فوسلل 2026'},
17322 | 'BULAN': {'id': 'BULAN', 'en': 'MOON', 'ar': 'رمؤل'},
17323 | 'Status Hilal Nanti Sore': {'id': 'Status Hilal Nanti Sore', 'en': 'Crescent Status This Evening', 'ar': 'هالسملا اذه لالهلا هالاح'},
17324 | 'Cuaca Lokal': {'id': 'Cuaca Lokal', 'en': 'Local Weather', 'ar': 'يلجرملا سقطل'},
17325 | 'Buka Simulasi Visual Mizwala': {'id': 'Buka Simulasi Visual Mizwala',
17326 |     'en': 'Open Mizwala Visual Simulation',
17327 |     'ar': 'هالوزملا ؤيرمبلا ةاكاحملا جف'},
17328 | 'Cetak PDF (Portrait)': {'id': 'Cetak PDF (Potret)', 'en': 'Print PDF (Portrait)', 'ar': 'يودومع PDF ةعابط'},
17329 | 'Alt Chart': {'id': 'Grafik Ketinggian', 'en': 'Altitude Chart', 'ar': 'عافتلال ططخم'},
17330 | 'Elongasi': {'id': 'Elongasi', 'en': 'Elongation', 'ar': 'هالطسالا'},
17331 | 'Ill vs Eln': {'id': 'Illuminasi vs Elongasi', 'en': 'Illumination vs Elongation', 'ar': 'هالطسالا لباقم ءاضالا'},
17332 | 'Illuminasi': {'id': 'Illuminasi', 'en': 'Illumination', 'ar': 'هالطسالا'},
17333 | 'PARAMETER KURVA ALASKA': {'id': 'PARAMETER KURVA ALASKA', 'en': 'ALASKA CURVE PARAMETERS', 'ar': 'السالل ينحم تالماعم'},
17334 | 'EXECUTIVE SUMMARY': {'id': 'RINGKASAN EKSEKUTIF', 'en': 'EXECUTIVE SUMMARY', 'ar': 'يذيفنتلا مزلمل'},
17335 | 'Tren Altitude (Geo vs Topo)': {'id': 'Tren Ketinggian (Geo vs Topo)',
17336 |     'en': 'Altitude Trend (Geo vs Topo)',
17337 |     'ar': 'يغصوم لباقم يجرى يزكرم عافتلال هاجت'},
17338 | 'Tren Elongasi (Geo vs Topo)': {'id': 'Tren Elongasi (Geo vs Topo)',
17339 |     'en': 'Elongation Trend (Geo vs Topo)',
17340 |     'ar': 'يغصوم لباقم يجرى يزكرم هالطسالا هاجت'},
17341 | 'Tren Iluminasi Siklik': {'id': 'Tren Iluminasi Siklik', 'en': 'Cyclic Illumination Trend', 'ar': 'يروزلا هالطسالا هاجت'},
17342 | 'Tren Kombinasi (Alt, Eln, Illum)': {'id': 'Tren Kombinasi (Tinggi, Elongasi, Iluminasi)',
17343 |     'en': 'Combined Trend (Altitude, Elongation, Illumination)',
17344 |     'ar': 'هالطسالا عافتلالا يكرملا هالجال'},
17345 | 'Catatan & Hari Penting (Nasional/Internasional)': {'id': 'Catatan dan Hari Penting (Nasional/Internasional)',
17346 |     'en': 'Notes and Important Days (National/International)',
17347 |     'ar': 'هاللود/هالطو ؤم مأي او تاطالح'},

```

```

17348 | 'Memuat Lokasi...': {'id': 'Memuat Lokasi...', 'en': 'Loading Location...', 'ar': 'عقومل لي محت ِراج'},
17349 | 'Memuat data lokasi dan tanggal...': {'id': 'Memuat data lokasi dan tanggal...',
17350 |                                     'en': 'Loading location and date data...',
17351 |                                     'ar': 'خيراتل او عقومل ساناي لي محت ِراج'},
17352 | 'Catatan & Hari Penting Islam': {'id': 'Catatan dan Hari Penting Islam:',
17353 |                                 'en': 'Notes and Important Islamic Days:',
17354 |                                 'ar': 'ةمهم فيم الساماي او ساطح الم'},
17355 | 'Modul 31: Astrofotografi Hilal & Visibilitas Kontras': {'id': 'Modul 31: Astrofotografi Hilal dan Visibilitas Kontras',
17356 |                                                         'en': 'Module 31: Crescent Astrophotography and Visibility Contrast',
17357 |                                                         'ar': 'ةقورل انباتو لالهل ريومت 31 ةدوجل'},
17358 | 'Simpan PNG': {'id': 'Simpan PNG', 'en': 'Save PNG', 'ar': 'فح PNG'},
17359 | 'Simpan PNG (Portrait)': {'id': 'Simpan PNG (Potret)', 'en': 'Save PNG (Portrait)', 'ar': 'فح PNG (يدومع)'},
17360 | 'Reset Memori': {'id': 'Atur Ulang Memori', 'en': 'Reset Memory', 'ar': 'ةركاذل طمس ةداع'},
17361 | 'Zoom Hilal': {'id': 'Perbesar Hilal', 'en': 'Zoom Crescent', 'ar': 'لالهلا ريكت'},
17362 | 'KOMPARASI DATA: STELLARIUM vs KOMPUTASI ENGINE': {'id': 'KOMPARASI DATA: STELLARIUM vs KOMPUTASI MESIN',
17363 |                                                         'en': 'DATA COMPARISON: STELLARIUM vs ENGINE COMPUTATION',
17364 |                                                         'ar': 'لرحمل باسح لباقم مويراليتس: سانايلا ةنراقم'},
17365 | 'Lihat Detail Lokal Saat Puncak': {'id': 'Lihat Detail Lokal Saat Puncak',
17366 |                                   'en': 'View Local Details at Maximum',
17367 |                                   'ar': 'ةورذل دنع ةلرحمل لي مافتل صرع'},
17368 | 'Stellarium Sync': {'id': 'Sinkronisasi Stellarium', 'en': 'Stellarium Sync', 'ar': 'مويراليتس ةنمازم'},
17369 | 'Kembali ke Live (Sekarang)': {'id': 'Kembali ke Waktu Langsung (Sekarang)', 'en': 'Return to Live (Now)', 'ar': 'رشابمل شلا لىل ةدوجل'},
17370 | 'LIVE SIMULATOR': {'id': 'SIMULATOR LANGSUNG', 'en': 'LIVE SIMULATOR', 'ar': 'رشابمل يكاحمل'},
17371 | 'Waktu Live': {'id': 'Waktu Langsung', 'en': 'Live Time', 'ar': 'رشابمل ةقول'},
17372 | 'Export Jalur Totalitas/Anularitas ke KML': {'id': 'Ekspor Jalur Totalitas/Anularitas ke KML',
17373 |                                              'en': 'Export Totality/Annularity Path to KML',
17374 |                                              'ar': 'KML لىل ةقولحلا/ةللللا راسم ريومت'},
17375 | 'Show Qiblah Map': {'id': 'Tampilkan Peta Kiblat', 'en': 'Show Qibla Map', 'ar': 'ةلبؤل ةطبرخ صرع'},
17376 | 'Ruang Interaksi AI': {'id': 'Ruang Interaksi AI', 'en': 'AI Interaction Space', 'ar': 'يعانطمال اء الكذل عم لغافتلا ةحاسم'},
17377 | 'Bujur (Longitude)': {'id': 'Bujur:', 'en': 'Longitude:', 'ar': 'لوطلا طح'},
17378 | 'Durasi (Jam)': {'id': 'Durasi (Jam)', 'en': 'Duration (Hours):', 'ar': 'ءاعاس (ءدل)'},
17379 | 'Elev': {'id': 'Elevasi:', 'en': 'Elevation:', 'ar': 'عافتلال'},
17380 | 'Elevasi (m)': {'id': 'Elevasi (m):', 'en': 'Elevation (m):', 'ar': '(م) عافتلال'},
17381 | 'Ikhtiyat (Detik)': {'id': 'Ikhtiyat (Detik):', 'en': 'Precaution (Seconds):', 'ar': '(رناون) طايئحال'},
17382 | 'Interval (Menit)': {'id': 'Interval (Menit)', 'en': 'Interval (Minutes):', 'ar': '(قئاقء) لمارفل'},
17383 | 'Jam (0-23)': {'id': 'Jam (0-23):', 'en': 'Hour (0-23):', 'ar': 'ءاعاسلا (0-23)'},
17384 | 'Jam (HH:MM)': {'id': 'Jam (JJ:MM):', 'en': 'Time (HH:MM):', 'ar': 'ءاعاسلا (HH:MM)'},
17385 | 'Jam Pengamatan (Lokal)': {'id': 'Jam Pengamatan (Lokal):', 'en': 'Observation Time (Local)', 'ar': '(لحم) ءمزل ةقو'},
17386 | 'Kelembapan (%)': {'id': 'Kelembapan (%)', 'en': 'Humidity (%)', 'ar': '(%) ةبوطرلا'},
17387 | 'Ketinggian (Kaki/ft)': {'id': 'Ketinggian (Kaki/ft):', 'en': 'Altitude (Feet/ft):', 'ar': '(مءق) عافتلال'},
17388 | 'Lat (Lintang)': {'id': 'Lintang:', 'en': 'Latitude:', 'ar': 'صرعلا طح'},
17389 | 'Lat Berangkat': {'id': 'Lintang Keberangkatan:', 'en': 'Departure Latitude:', 'ar': 'ةرداعمل صرع طح'},
17390 | 'Lat Tujuan': {'id': 'Lintang Tujuan:', 'en': 'Destination Latitude:', 'ar': 'ةءوجل صرع طح'},
17391 | 'Lat': {'id': 'Lintang:', 'en': 'Latitude:', 'ar': 'صرعلا طح'},
17392 | 'Lintang (Latitude)': {'id': 'Lintang:', 'en': 'Latitude:', 'ar': 'صرعلا طح'},
17393 | 'Lon (Bujur)': {'id': 'Bujur:', 'en': 'Longitude:', 'ar': 'لوطلا طح'},
17394 | 'Lon Berangkat': {'id': 'Bujur Keberangkatan:', 'en': 'Departure Longitude:', 'ar': 'ةرداعمل لوط طح'},
17395 | 'Lon Tujuan': {'id': 'Bujur Tujuan:', 'en': 'Destination Longitude:', 'ar': 'ةءوجل لوط طح'},
17396 | 'Lon': {'id': 'Bujur:', 'en': 'Longitude:', 'ar': 'لوطلا طح'},
17397 | 'Suhu (°C)': {'id': 'Suhu (°C):', 'en': 'Temperature (°C):', 'ar': '(°م) ةراجل ةءرء'},
17398 | 'TZ': {'id': 'Zona Waktu:', 'en': 'Time Zone:', 'ar': 'ةنمزل ةطنمل'},
17399 | 'Tahun (YYYY)': {'id': 'Tahun (TTTT):', 'en': 'Year (YYYY):', 'ar': 'ءنسل (YYYY)'},
17400 | 'Tahun Akhir (H)': {'id': 'Tahun Akhir (H):', 'en': 'End Year (AH):', 'ar': '(ءه) ةيامل ءنس'},
17401 | 'Tahun Analisis': {'id': 'Tahun Analisis:', 'en': 'Analysis Year:', 'ar': 'لئلحال ءنس'},
17402 | 'Tahun Awal (H)': {'id': 'Tahun Awal (H):', 'en': 'Start Year (AH):', 'ar': '(ءه) ةيادل ءنس'},
17403 | 'Tekanan (mb)': {'id': 'Tekanan (mb):', 'en': 'Pressure (mb):', 'ar': '(رابطلم) طغللا'},
17404 | 'Timezone Awal': {'id': 'Zona Waktu Awal:', 'en': 'Initial Time Zone:', 'ar': 'ةيئءبال ةنمزل ةطنمل'},
17405 | 'Timezone': {'id': 'Zona Waktu:', 'en': 'Time Zone:', 'ar': 'ةنمزل ةطنمل'},
17406 | 'Tinggi (cm)': {'id': 'Tinggi (cm):', 'en': 'Height (cm):', 'ar': '(مس) عافتلال'},
17407 | 'Waktu (HH:MM)': {'id': 'Waktu (JJ:MM):', 'en': 'Time (HH:MM):', 'ar': '(HH:MM) ةقول'}}
17408 |
17409 | # Cache eksternal yang sudah dimuat tetap menjadi prioritas.
17410 | for source_text, langs in fallback.items():
17411 |     if not isinstance(source_text, str) or not isinstance(langs, dict):
17412 |         continue
17413 |     entry = self.translations.setdefault(source_text, {})
17414 |     for lang in ("id", "en", "ar"):
17415 |         value = langs.get(lang)
17416 |         if isinstance(value, str) and value:
17417 |             entry.setdefault(lang, value)
17418 |

```

```

17419 |         self._builtin_translations_installed = True
17420 |         self._build_language_indexes()
17421 |
17422 |     def _norm_lang_text(self, value):
17423 |         value = str(value or "").strip().replace("_", "")
17424 |         value = re.sub(r"\s+", " ", value)
17425 |         return value.casefold()
17426 |
17427 |     def tr(self, text):
17428 |         """Terjemahan exact-match dari varian ID/EN/AR ke bahasa aktif; 0(1)."""
17429 |         if not text:
17430 |             return text
17431 |         self.install_builtin_translations()
17432 |         lang = getattr(self, "current_lang", "id")
17433 |         if lang not in ("id", "en", "ar"):
17434 |             lang = "id"
17435 |
17436 |         canonical_id = self._reverse_translation_map.get(
17437 |             self._norm_lang_text(text), text
17438 |         )
17439 |         mapping = self._translation_index.get(lang, {})
17440 |         return mapping.get(
17441 |             canonical_id,
17442 |             mapping.get(text, canonical_id if lang == "id" else text)
17443 |         )
17444 |
17445 |     def reverse_tr(self, current_text):
17446 |         """Mengembalikan teks Indonesia asli dari tampilan ID/EN/AR; 0(1)."""
17447 |         if not current_text:
17448 |             return current_text
17449 |         self.install_builtin_translations()
17450 |         return self._reverse_translation_map.get(self._norm_lang_text(current_text), current_text)
17451 |
17452 |
17453 |     def change_language(self, choice):
17454 |         """Dipanggil saat pilihan bahasa id/en/ar diubah."""
17455 |         choice_norm = str(choice or "").strip().lower()
17456 |         self.current_lang = choice_norm if choice_norm in ("id", "en", "ar") else "id"
17457 |
17458 |         # Bahasa berlaku ke seluruh antarmuka: sidebar kiri, header,
17459 |         # tombol, frame output, dan isi laporan di ruang kanan.
17460 |         self.force_update_sidebar_language()
17461 |         self.update_ui_texts(self)
17462 |         self.force_update_sidebar_language()
17463 |         self.apply_language_direction()
17464 |
17465 |         if getattr(self, "active_right_view", "") == "release_info":
17466 |             # Beranda digambar ulang hanya bila pengguna memang sedang di beranda.
17467 |             self.show_release_info()
17468 |         else:
17469 |             self.refresh_right_output_language()
17470 |             self.refresh_secondary_output_languages()
17471 |
17472 |         try:
17473 |             if hasattr(self, "combo_lang"):
17474 |                 self.combo_lang.configure(values=["id", "en", "ar"])
17475 |                 self.combo_lang.set(self.current_lang)
17476 |         except Exception:
17477 |             pass
17478 |
17479 |     def refresh_right_output_language(self):
17480 |         """Gambar ulang semua teks utama ruang kanan dari teks mentah.
17481 |         Prioritas: laporan kalkulasi terakhir; jika tidak ada, buffer mentah
17482 |         terakhir dari self.textbox.insert(...) dipakai. Dengan cara ini,
17483 |         output kanan tidak berhenti pada sidebar saja.
17484 |         """
17485 |         if not hasattr(self, "textbox"):
17486 |             return
17487 |         if getattr(self, "active_right_view", "") == "release_info":
17488 |             self.show_release_info()
17489 |             return

```

```

17490 |         raw_text = getattr(self, '_last_raw_report', '') or getattr(self, '_last_raw_textbox_content', '')
17491 |         if not raw_text:
17492 |             try:
17493 |                 raw_text = self.textbox.get("1.0", "end-1c")
17494 |             except Exception:
17495 |                 raw_text = ""
17496 |         if raw_text:
17497 |             try:
17498 |                 self.textbox.configure(state="normal")
17499 |                 self.textbox.delete("1.0", "end")
17500 |                 self.textbox.insert("1.0", raw_text)
17501 |                 self.textbox.configure(state="disabled")
17502 |             except Exception:
17503 |                 pass
17504 |
17505 |     def get_release_lang_cache_path(self):
17506 |         """Lokasi file cache JSON khusus INFORMASI RILIS."""
17507 |         return self._find_cache_file("lang_cache_release.json")
17508 |
17509 |     def get_output_lang_cache_path(self):
17510 |         """Lokasi cache JSON untuk hasil translasi blok/output penuh."""
17511 |         return self._find_cache_file("lang_output_cache.json")
17512 |
17513 |     def _output_cache_key(self, lang, raw):
17514 |         return hashlib.sha256((str(lang) + "\0" + str(raw)).encode("utf-8", errors="ignore")).hexdigest()
17515 |
17516 |     def load_release_language_cache(self):
17517 |         """Membaca cache JSON terjemahan halaman INFORMASI RILIS."""
17518 |         self.release_translation_cache = {"en": {}, "ar": {}}
17519 |         try:
17520 |             cache_path = getattr(self, "release_lang_cache_path", None) or self.get_release_lang_cache_path()
17521 |             self.release_lang_cache_path = cache_path
17522 |             if os.path.exists(cache_path):
17523 |                 with open(cache_path, "r", encoding="utf-8") as f:
17524 |                     data = json.load(f)
17525 |                     if isinstance(data, dict):
17526 |                         for lang in ("en", "ar"):
17527 |                             if isinstance(data.get(lang), dict):
17528 |                                 self.release_translation_cache[lang].update({
17529 |                                     str(k): str(v) for k, v in data[lang].items() if isinstance(v, str)
17530 |                                 })
17531 |             # Masukkan frasa release ke kamus UI agar fallback per-frasa tetap ada.
17532 |             self._merge_translation_mapping(self.release_translation_cache)
17533 |             self._release_translation_keys = set(self.release_translation_cache.get("en", {})) | set(self.release_translation_cache.get("ar", {}))
17534 |             self._build_language_indexes()
17535 |         except Exception as e:
17536 |             print(f"Peringatan: gagal membaca lang_cache_release.json ({e})")
17537 |
17538 |     def save_release_language_cache(self, force=False):
17539 |         """Menyimpan cache JSON INFORMASI RILIS bila berubah."""
17540 |         if not force and not getattr(self, "_release_translation_cache_dirty", False):
17541 |             return
17542 |         try:
17543 |             cache_path = getattr(self, "release_lang_cache_path", None) or self.get_release_lang_cache_path()
17544 |             self.release_lang_cache_path = cache_path
17545 |             data = getattr(self, "release_translation_cache", None) or {"en": {}, "ar": {}}
17546 |             with open(cache_path, "w", encoding="utf-8") as f:
17547 |                 json.dump(data, f, ensure_ascii=False, indent=2)
17548 |             self._release_translation_cache_dirty = False
17549 |         except Exception as e:
17550 |             print(f"Peringatan: gagal menyimpan lang_cache_release.json ({e})")
17551 |
17552 |     def save_output_language_cache(self, force=False):
17553 |         """Menyimpan cache translasi blok/output penuh bila berubah."""
17554 |         if not force and not getattr(self, "_lang_output_cache_dirty", False):
17555 |             return
17556 |         try:
17557 |             cache_path = self.get_output_lang_cache_path()
17558 |             data = getattr(self, "_lang_output_cache", {}) or {}
17559 |             with open(cache_path, "w", encoding="utf-8") as f:
17560 |                 json.dump(data, f, ensure_ascii=False, indent=2)

```

```

17561 |         self._lang_output_cache_dirty = False
17562 |     except Exception as e:
17563 |         print(f"Peringatan: gagal menyimpan lang_output_cache.json ({e})")
17564 |
17565 | def _get_translation_regexes(self, lang):
17566 |     """Kompilasi regex sekali per bahasa untuk mengganti ribuan frasa dalam satu-dua lintasan."""
17567 |     if lang in getattr(self, "_translation_patterns", {}):
17568 |         return self._translation_patterns[lang]
17569 |
17570 |     mapping = self._translation_index.get(lang, {}) if hasattr(self, "_translation_index") else {}
17571 |     simple_keys = []
17572 |     bounded_keys = []
17573 |     for key, trans in mapping.items():
17574 |         if not key or not trans or trans == key:
17575 |             continue
17576 |         # Kata alfabetis/pendek diberi batas kata agar tidak merusak kata lain.
17577 |         if len(key) <= 3 or key.isalpha():
17578 |             bounded_keys.append(key)
17579 |         else:
17580 |             simple_keys.append(key)
17581 |     simple_keys.sort(key=len, reverse=True)
17582 |     bounded_keys.sort(key=len, reverse=True)
17583 |
17584 |     def compile_alt(keys, bounded=False):
17585 |         if not keys:
17586 |             return None
17587 |         alt = "|".join(re.escape(k) for k in keys)
17588 |         if bounded:
17589 |             alt = r"(?<!\w)(?:" + alt + r")?!?\w)"
17590 |         return re.compile(alt)
17591 |
17592 |     try:
17593 |         patterns = (compile_alt(simple_keys, False), compile_alt(bounded_keys, True))
17594 |     except Exception:
17595 |         patterns = (None, None)
17596 |     self._translation_patterns[lang] = patterns
17597 |     return patterns
17598 |
17599 | def _replace_by_language_map(self, raw, lang, release_only=False):
17600 |     result = str(raw)
17601 |     mapping = self._translation_index.get(lang, {}) if hasattr(self, "_translation_index") else {}
17602 |     if not mapping:
17603 |         return result
17604 |
17605 |     if release_only:
17606 |         keys = getattr(self, "_release_translation_keys", set()) or set(mapping.keys())
17607 |         sorted_keys = getattr(self, "_release_sorted_keys_cache", None)
17608 |         if sorted_keys is None:
17609 |             sorted_keys = sorted(keys, key=len, reverse=True)
17610 |             self._release_sorted_keys_cache = sorted_keys
17611 |         for key in sorted_keys:
17612 |             trans = mapping.get(key)
17613 |             if trans and trans != key and key in result:
17614 |                 result = result.replace(key, trans)
17615 |         return result
17616 |
17617 |     simple_pattern, bounded_pattern = self._get_translation_regexes(lang)
17618 |     def repl(match):
17619 |         return mapping.get(match.group(0), match.group(0))
17620 |
17621 |     try:
17622 |         if simple_pattern is not None:
17623 |             result = simple_pattern.sub(repl, result)
17624 |         if bounded_pattern is not None:
17625 |             result = bounded_pattern.sub(repl, result)
17626 |         return result
17627 |     except Exception:
17628 |         # Fallback aman seperti metode lama, tetapi tetap hanya saat regex gagal.
17629 |         sorted_keys = getattr(self, "_translation_sorted_keys_cache", None)
17630 |         if sorted_keys is None:
17631 |             sorted_keys = sorted(mapping.keys(), key=len, reverse=True)
17632 |             self._translation_sorted_keys_cache = sorted_keys

```

```

17632 |         for key in sorted_keys:
17633 |             trans = mapping.get(key)
17634 |             if trans and trans != key and key in result:
17635 |                 result = result.replace(key, trans)
17636 |         return result
17637 |
17638 | def _translate_release_piece_fast(self, text):
17639 |     """Terjemahan cepat satu potongan release memakai cache + kamus release saja."""
17640 |     raw = str(text or "")
17641 |     lang = getattr(self, "current_lang", "id")
17642 |     if not raw or lang == "id":
17643 |         return raw
17644 |     self.install_builtin_translations()
17645 |
17646 |     stripped = raw.strip()
17647 |     leading = raw[:len(raw) - len(raw.lstrip())]
17648 |     trailing = raw[len(raw.rstrip()):]
17649 |     exact = self._translation_index.get(lang, {}).get(stripped)
17650 |     if exact:
17651 |         return leading + exact + trailing
17652 |
17653 |     if "\n" in raw:
17654 |         translated_lines = []
17655 |         changed = False
17656 |         for line in raw.splitlines(keepends=True):
17657 |             endl = ""
17658 |             body = line
17659 |             if line.endswith("\r\n"):
17660 |                 body, endl = line[:-2], "\r\n"
17661 |             elif line.endswith("\n"):
17662 |                 body, endl = line[:-1], "\n"
17663 |             new_body = self._translate_release_piece_fast(body) if body.strip() else body
17664 |             changed = changed or (new_body != body)
17665 |             translated_lines.append(new_body + endl)
17666 |         if changed:
17667 |             return "".join(translated_lines)
17668 |
17669 |     return self._replace_by_language_map(raw, lang, release_only=True)
17670 |
17671 | def translate_release_text_cached(self, text):
17672 |     """Ambil/simpan terjemahan release dari lang_cache_release + lang_output_cache."""
17673 |     raw = text if isinstance(text, str) else str(text)
17674 |     lang = getattr(self, "current_lang", "id")
17675 |     if not raw or lang == "id" or lang not in ("en", "ar"):
17676 |         return raw
17677 |
17678 |     cache = getattr(self, "release_translation_cache", None)
17679 |     if not isinstance(cache, dict):
17680 |         self.load_release_language_cache()
17681 |         cache = self.release_translation_cache
17682 |     cache.setdefault("en", {})
17683 |     cache.setdefault("ar", {})
17684 |
17685 |     # Kamus highlight V8.0 mendahului kedua cache: jangan gunakan hasil
17686 |     # lama yang masih ID/campuran ketika pengguna memilih EN atau AR.
17687 |     translated = self._translate_release_highlight_piece(raw, lang)
17688 |     out_cache = getattr(self, "_lang_output_cache", {}) or {}
17689 |     out_key = self._output_cache_key(lang, raw)
17690 |     if translated is None:
17691 |         if raw in cache[lang]:
17692 |             return cache[lang][raw]
17693 |         if out_key in out_cache:
17694 |             translated = out_cache[out_key]
17695 |         else:
17696 |             translated = self._translate_release_piece_fast(raw)
17697 |
17698 |     if cache[lang].get(raw) != translated:
17699 |         cache[lang][raw] = translated
17700 |         self._release_translation_cache_dirty = True
17701 |     if out_cache.get(out_key) != translated:
17702 |         out_cache[out_key] = translated

```

```

17703 |         self._lang_output_cache_dirty = True
17704 |         self._lang_output_cache = out_cache
17705 |         return translated
17706 |
17707 |     def _fix_visibility_id_alignment(self, text):
17708 |         """Rapikan alignment khusus output ID pada modul Visibilitas Hilal (KHGT).
17709 |
17710 |         Translasi dilakukan setelah string laporan selesai diberi padding.
17711 |         Akibatnya penggantian "Sunset" -> "Matahari Terbenam" dan
17712 |         "Moonset" -> "Bulan Terbenam" membuat posisi titik dua bergeser.
17713 |         Fungsi ini hanya menyentuh dua baris tersebut pada laporan visibilitas,
17714 |         tanpa mengubah nilai waktu maupun kalkulasi astronomis.
17715 |         """
17716 |         if not text:
17717 |             return text
17718 |
17719 |         result_lines = []
17720 |         for line in str(text).splitlines(keepends=True):
17721 |             # Baris Sunset pada report visibilitas memiliki pasangan "G. Moon Age".
17722 |             if "Matahari Terbenam" in line and "G. Moon Age" in line:
17723 |                 line = re.sub(
17724 |                     r"Matahari Terbenam\s*:",
17725 |                     f'{"Matahari Terbenam":<22}: ',
17726 |                     line,
17727 |                     count=1,
17728 |                 )
17729 |
17730 |             # Baris Moonset pada report visibilitas memiliki pasangan "Moon Lag Time".
17731 |             elif "Bulan Terbenam" in line and "Moon Lag Time" in line:
17732 |                 line = re.sub(
17733 |                     r"Bulan Terbenam\s*:",
17734 |                     f'{"Bulan Terbenam":<22}: ',
17735 |                     line,
17736 |                     count=1,
17737 |                 )
17738 |
17739 |             result_lines.append(line)
17740 |
17741 |         return "".join(result_lines)
17742 |
17743 |     def translate_text_block(self, text):
17744 |         """Menerjemahkan blok string laporan/output kanan secara cepat dan cached."""
17745 |         if not text:
17746 |             return text
17747 |         lang = getattr(self, "current_lang", "id")
17748 |         if lang not in ("id", "en", "ar"):
17749 |             lang = "id"
17750 |         raw = text if isinstance(text, str) else str(text)
17751 |
17752 |         # Hook Textbox juga harus memperbaiki cache lama untuk judul,
17753 |         # heading, dan uraian highlight, termasuk teks dengan pembatas/tag.
17754 |         is_release = (getattr(self, "active_right_view", "") == "release_info"
17755 |                       or getattr(self, "_rendering_release_info", False))
17756 |         if is_release and self._translate_release_highlight_piece(raw, lang) is not None:
17757 |             return self._translate_release_text_cached(raw)
17758 |
17759 |         out_cache = getattr(self, "_lang_output_cache", {}) or {}
17760 |         out_key = self._output_cache_key(lang, raw)
17761 |         if out_key in out_cache:
17762 |             cached = out_cache[out_key]
17763 |             if lang == "id":
17764 |                 cached = self._fix_visibility_id_alignment(cached)
17765 |             out_cache[out_key] = cached
17766 |             return cached
17767 |
17768 |         if getattr(self, "active_right_view", "") == "release_info" or getattr(self, "_rendering_release_info", False):
17769 |             translated = self._translate_release_text_cached(raw)
17770 |         else:
17771 |             self.install_builtin_translations()
17772 |             translated = self._replace_by_language_map(raw, lang, release_only=False)
17773 |

```



```

17774 |         # KHUSUS ID - Visibilitas Hilal (KHGT):
17775 |         # rapikan kembali dua label yang memanjang setelah translasi.
17776 |         if lang == "id":
17777 |             translated = self._fix_visibility_id_alignment(translated)
17778 |
17779 |         # Batasi ukuran cache agar file JSON tidak tumbuh tak terkendali.
17780 |         if len(out_cache) > 5000:
17781 |             for k in list(out_cache.keys())[:1000]:
17782 |                 out_cache.pop(k, None)
17783 |             out_cache[out_key] = translated
17784 |             self._lang_output_cache = out_cache
17785 |             self._lang_output_cache_dirty = True
17786 |             return translated
17787 |
17788 |
17789 |     def get_process_prompt_text(self):
17790 |         """Pesan standar ketika modul menunggu tombol PROSES DATA."""
17791 |         messages = {
17792 |             "id": "Silakan klik tombol ► PROSES DATA untuk menampilkan output.",
17793 |             "en": "Please click the ► PROCESS DATA button to display the output.",
17794 |             "ar": "جاءت النلا صرغل تانانابلا ةجللاغم ► زر دلغ روتلا جري.",
17795 |         }
17796 |         return messages.get(getattr(self, "current_lang", "id"), messages["id"])
17797 |
17798 |
17799 |     def prepare_module_right_output(self):
17800 |         """Kosongkan sisa beranda/laporan lama ketika pengguna memilih menu."""
17801 |         self.active_right_view = "module_pending"
17802 |         self._last_raw_report = ""
17803 |         raw_prompt = "Silakan klik tombol ► PROSES DATA untuk menampilkan output."
17804 |         self._last_raw_textbox_content = raw_prompt
17805 |
17806 |         try:
17807 |             self.textbox.grid(row=1, column=0, sticky="nsew")
17808 |             self.textbox.configure(state="normal", font=("Consolas", 13), wrap="word")
17809 |             self.textbox.delete("1.0", "end")
17810 |             self.textbox.insert("1.0", raw_prompt)
17811 |             self.textbox.configure(state="disabled")
17812 |         except Exception:
17813 |             pass
17814 |         finally:
17815 |             self.active_right_view = "module_pending"
17816 |
17817 |
17818 |     def show_pending_message_in_frame(self, frame):
17819 |         """Tampilkan petunjuk terlokalisasi pada frame output non-teks yang belum diproses."""
17820 |         try:
17821 |             for widget in frame.winfo_children():
17822 |                 widget.destroy()
17823 |             label = ctk.CTkLabel(
17824 |                 frame,
17825 |                 text=self.get_process_prompt_text(),
17826 |                 font=("Segoe UI", 17, "bold"),
17827 |                 text_color="#00E5FF",
17828 |                 wraplength=700,
17829 |                 justify="right" if getattr(self, "current_lang", "id") == "ar" else "center",
17830 |             )
17831 |             label.pack(fill="both", expand=True, padx=30, pady=30)
17832 |             setattr(
17833 |                 label,
17834 |                 "_khgt_id_text",
17835 |                 "Silakan klik tombol ► PROSES DATA untuk menampilkan output."
17836 |             )
17837 |         except Exception:
17838 |             pass
17839 |
17840 |
17841 |     def install_secondary_output_textbox_hooks(self):
17842 |         """Pasang hook bahasa pada CTkTextbox output selain textbox utama."""
17843 |         if not hasattr(self, "main_frame"):
17844 |             return
17845 |
17846 |         def walk(widget):
17847 |             try:

```

```

17845 |         children = widget.winfo_children()
17846 |     except Exception:
17847 |         children = []
17848 |     for child in children:
17849 |         try:
17850 |             if isinstance(child, ctk.CTkTextbox) and child is not getattr(self, "textbox", None):
17851 |                 self._install_one_secondary_textbox_hook(child)
17852 |         except Exception:
17853 |             pass
17854 |         walk(child)
17855 |
17856 |     walk(self.main_frame)
17857 |
17858 | def _install_one_secondary_textbox_hook(self, widget):
17859 |     if getattr(widget, "_khgt_lang_hook_installed", False):
17860 |         return
17861 |     widget._khgt_lang_hook_installed = True
17862 |     try:
17863 |         raw_insert = widget.insert
17864 |         raw_delete = widget.delete
17865 |         widget._khgt_raw_insert = raw_insert
17866 |         widget._khgt_raw_delete = raw_delete
17867 |         widget._khgt_raw_content = widget.get("1.0", "end-1c")
17868 |     except Exception:
17869 |         return
17870 |
17871 | def _delete(index1, index2=None, *args):
17872 |     try:
17873 |         idx1 = str(index1).lower()
17874 |         idx2 = str(index2).lower() if index2 is not None else None
17875 |         if idx1 in ("0.0", "1.0") and (
17876 |             idx2 in ("end", "end-1c", "tk.end") or index2 is None
17877 |         ):
17878 |             widget._khgt_raw_content = ""
17879 |     except Exception:
17880 |         pass
17881 |     if index2 is None:
17882 |         return raw_delete(index1, *args)
17883 |     return raw_delete(index1, index2, *args)
17884 |
17885 | def _insert(index, content, *args):
17886 |     raw = content if isinstance(content, str) else str(content)
17887 |     try:
17888 |         idx = str(index).lower()
17889 |         if idx in ("0.0", "1.0"):
17890 |             widget._khgt_raw_content = raw
17891 |         else:
17892 |             widget._khgt_raw_content = getattr(widget, "_khgt_raw_content", "") + raw
17893 |     except Exception:
17894 |         pass
17895 |     rendered = self.translate_text_block(raw) if isinstance(content, str) else content
17896 |     return raw_insert(index, rendered, *args)
17897 |
17898 | widget.delete = _delete
17899 | widget.insert = _insert
17900 |
17901 | def refresh_secondary_output_languages(self):
17902 |     """Gambar ulang semua CTKTextbox output kanan dari teks mentahnya."""
17903 |     self.install_secondary_output_textbox_hooks()
17904 |     if not hasattr(self, "main_frame"):
17905 |         return
17906 |
17907 | def walk(widget):
17908 |     try:
17909 |         children = widget.winfo_children()
17910 |     except Exception:
17911 |         children = []
17912 |     for child in children:
17913 |         try:
17914 |             if isinstance(child, ctk.CTkTextbox) and child is not getattr(self, "textbox", None):
17915 |                 raw = getattr(child, "_khgt_raw_content", None)

```

```

17916 |         if raw is None:
17917 |             raw = child.get("1.0", "end-1c")
17918 |             child._khgt_raw_content = raw
17919 |             state = str(child.cget("state"))
17920 |             child.configure(state="normal")
17921 |             child._khgt_raw_delete("1.0", "end")
17922 |             child._khgt_raw_insert("1.0", self.translate_text_block(raw))
17923 |             child.configure(state=state)
17924 |         except Exception:
17925 |             pass
17926 |         walk(child)
17927 |
17928 |     walk(self.main_frame)
17929 |
17930 | def apply_language_direction(self):
17931 |     """Atur perataan dasar untuk bahasa Arab tanpa mengubah logika widget."""
17932 |     is_ar = getattr(self, "current_lang", "id") == "ar"
17933 |     anchor = "e" if is_ar else "w"
17934 |     justify = "right" if is_ar else "left"
17935 |
17936 |     def walk(widget):
17937 |         try:
17938 |             children = widget.winfo_children()
17939 |         except Exception:
17940 |             children = []
17941 |         for child in children:
17942 |             try:
17943 |                 if isinstance(
17944 |                     child,
17945 |                     (ctk.CTkLabel, ctk.CTkButton, ctk.CTkRadioButton, ctk.CTkSwitch)
17946 |                 ):
17947 |                     txt = str(child.cget("text") or "")
17948 |                     if txt not in ("□", "□", "□", "□", "□", "□", "|"):
17949 |                         child.configure(anchor=anchor)
17950 |                 if isinstance(child, ctk.CTkTextbox):
17951 |                     try:
17952 |                         child._textbox.tag_configure(
17953 |                             "_khgt_lang_direction",
17954 |                             justify=justify
17955 |                         )
17956 |                         child._textbox.tag_add(
17957 |                             "_khgt_lang_direction", "1.0", "end"
17958 |                         )
17959 |                     except Exception:
17960 |                         pass
17961 |                 except Exception:
17962 |                     pass
17963 |                 walk(child)
17964 |
17965 |             try:
17966 |                 walk(self)
17967 |             except Exception:
17968 |                 pass
17969 |
17970 | def install_textbox_language_hooks(self):
17971 |     """Membungkus self.textbox.insert/delete agar semua output kanan
17972 |     otomatis mengikuti bahasa aktif tanpa harus mendedit 100+ titik insert.
17973 |     """
17974 |     if not hasattr(self, "textbox") or getattr(self, "_textbox_lang_hooks_installed", False):
17975 |         return
17976 |     self._textbox_lang_hooks_installed = True
17977 |     try:
17978 |         self._raw_textbox_insert = self.textbox.insert
17979 |         self._raw_textbox_delete = self.textbox.delete
17980 |         self._last_raw_textbox_content = self.textbox.get("1.0", "end-1c")
17981 |     except Exception:
17982 |         return
17983 |
17984 |     def _delete(index1, index2=None, *args):
17985 |         try:
17986 |             idx1 = str(index1).lower()

```

```

17987 |         idx2 = str(index2).lower() if index2 is not None else None
17988 |         if idx1 == "1.0" and (idx2 in ("end", "end-1c", "tk.end") or index2 is None):
17989 |             self._last_raw_textbox_content = ""
17990 |             if not getattr(self, "_rendering_release_info", False):
17991 |                 self.active_right_view = ""
17992 |     except Exception:
17993 |         pass
17994 |     if index2 is None:
17995 |         return self._raw_textbox_delete(index1, *args)
17996 |     return self._raw_textbox_delete(index1, index2, *args)
17997 |
17998 | def _insert(index, text, *args):
17999 |     raw_text = text if isinstance(text, str) else str(text)
18000 |     try:
18001 |         idx = str(index).lower()
18002 |         if idx == "end":
18003 |             self._last_raw_textbox_content = getattr(self, "_last_raw_textbox_content", "") + raw_text
18004 |         elif idx == "1.0":
18005 |             self._last_raw_textbox_content = raw_text
18006 |         else:
18007 |             self._last_raw_textbox_content = getattr(self, "_last_raw_textbox_content", "") + raw_text
18008 |     except Exception:
18009 |         pass
18010 |     if isinstance(text, str):
18011 |         text = self.translate_text_block(text)
18012 |     return self._raw_textbox_insert(index, text, *args)
18013 |
18014 | self.textbox.delete = _delete
18015 | self.textbox.insert = _insert
18016 |
18017 | def capture_sidebar_original_texts(self):
18018 |     """Menyimpan teks asli sidebar kiri (bahasa Indonesia) untuk translasi stabil.
18019 |     Dipanggil setelah seluruh widget sidebar selesai dibuat. Penyimpanan ini
18020 |     tidak mengubah algoritma modul; hanya memberi acuan agar tombol bahasa
18021 |     dapat mengembalikan/menerjemahkan UI kiri dengan benar.
18022 |     """
18023 |     self._sidebar_original_text_widgets = {}
18024 |     self._sidebar_original_value_widgets = {}
18025 |
18026 |     def walk(widget):
18027 |         try:
18028 |             children = widget.winfo_children()
18029 |         except Exception:
18030 |             children = []
18031 |         for child in children:
18032 |             try:
18033 |                 if isinstance(child, (ctk.CTkLabel, ctk.CTkButton, ctk.CTkRadioButton, ctk.CTkSwitch)):
18034 |                     curr_text = child.cget("text")
18035 |                     if curr_text:
18036 |                         # Simpan teks Indonesia asli. reverse_tr dipakai agar aman
18037 |                         # bila fungsi ini dipanggil ulang saat UI sudah EN/AR.
18038 |                         id_text = self.reverse_tr(curr_text)
18039 |                         setattr(child, "_khgt_id_text", id_text)
18040 |                         self._sidebar_original_text_widgets[child] = id_text
18041 |
18042 |                 if isinstance(child, (ctk.CTkOptionMenu, ctk.CTkComboBox)):
18043 |                     if getattr(self, 'combo_lang', None) and child == self.combo_lang:
18044 |                         setattr(child, "_khgt_lang_selector", True)
18045 |                     else:
18046 |                         vals = child.cget("values") or []
18047 |                         id_vals = [self.reverse_tr(v) for v in vals]
18048 |                         setattr(child, "_khgt_id_values", id_vals)
18049 |                         self._sidebar_original_value_widgets[child] = id_vals
18050 |             except Exception:
18051 |                 pass
18052 |         walk(child)
18053 |
18054 |     try:
18055 |         walk(self.sidebar)
18056 |     except Exception:
18057 |         pass

```

```

18058 |
18059 | def _translate_id_text_for_current_lang(self, id_text):
18060 |     """Terjemahkan teks ID secara aman untuk widget UI kiri."""
18061 |     if not id_text:
18062 |         return id_text
18063 |     return self.tr(id_text)
18064 |
18065 | def force_update_sidebar_language(self):
18066 |     """Memaksa sidebar kiri mengikuti bahasa aktif setelah menu 1-41 dipilih.
18067 |     Fungsi ini tidak mengubah algoritma modul; hanya memperbarui teks widget
18068 |     kiri berdasarkan teks Indonesia asli + lang_cache_ui.json + fallback internal.
18069 |     """
18070 |     try:
18071 |         self.install_built_in_translations()
18072 |     except Exception:
18073 |         pass
18074 |
18075 |     # Jika belum pernah direkam, rekam sekarang dari struktur widget yang ada.
18076 |     if not getattr(self, "_sidebar_original_text_widgets", None):
18077 |         try:
18078 |             self.capture_sidebar_original_texts()
18079 |         except Exception:
18080 |             pass
18081 |
18082 |     # 1) Paksa semua Label/Button/Switch/Radio di sidebar dari teks ID asli.
18083 |     try:
18084 |         for widget, id_text in list(getattr(self, "_sidebar_original_text_widgets", {}).items()):
18085 |             try:
18086 |                 if widget.winfo_exists():
18087 |                     widget.configure(text=self._translate_id_text_for_current_lang(id_text))
18088 |             except Exception:
18089 |                 pass
18090 |     except Exception:
18091 |         pass
18092 |
18093 |     # 2) Paksa tombol menu 1-41 dari kunci Indonesia asli yang dipakai saat dibuat.
18094 |     try:
18095 |         for id_text, btn in getattr(self, "menu_buttons", {}).items():
18096 |             try:
18097 |                 btn.configure(text=self._translate_id_text_for_current_lang(id_text))
18098 |             except Exception:
18099 |                 pass
18100 |     except Exception:
18101 |         pass
18102 |
18103 |     # 3) Paksa OptionMenu/ComboBox sidebar dari daftar ID asli, kecuali selector bahasa.
18104 |     try:
18105 |         for widget, id_values in list(getattr(self, "_sidebar_original_value_widgets", {}).items()):
18106 |             try:
18107 |                 if not widget.winfo_exists():
18108 |                     continue
18109 |                 new_values = [self._translate_id_text_for_current_lang(v) for v in id_values]
18110 |                 widget.configure(values=new_values)
18111 |                 curr = widget.get()
18112 |                 id_curr = self.reverse_tr(curr)
18113 |                 if id_curr in id_values:
18114 |                     widget.set(self._translate_id_text_for_current_lang(id_curr))
18115 |             except Exception:
18116 |                 pass
18117 |     except Exception:
18118 |         pass
18119 |
18120 |     # 4) Selector bahasa tidak boleh berubah menjadi teks terjemahan.
18121 |     try:
18122 |         if hasattr(self, "combo_lang"):
18123 |             self.combo_lang.configure(values=["id", "en", "ar"])
18124 |             self.combo_lang.set(getattr(self, "current_lang", "id"))
18125 |     except Exception:
18126 |         pass
18127 |
18128 | def update_ui_texts(self, widget):

```

```
18129 |         """Melakukan rekursi pada seluruh elemen GUI dan menerjemahkannya."""
18130 |         self.install_builtin_translations()
18131 |         for child in widget.winfo_children():
18132 |             try:
18133 |                 # Terjemahkan Label, Tombol, Radio, dan Checkbox
18134 |                 if isinstance(child, (ctk.CTkLabel, ctk.CTkButton, ctk.CTkRadioButton, ctk.CTkSwitch)):
18135 |                     curr_text = child.cget("text")
18136 |                     if curr_text:
18137 |                         orig = self.reverse_tr(curr_text)
18138 |                         new_t = self.tr(orig)
18139 |                         if new_t != curr_text:
18140 |                             child.configure(text=new_t)
18141 |
18142 |                 # Terjemahkan OptionMenus & ComboBoxes
18143 |                 if isinstance(child, (ctk.CTkOptionMenu, ctk.CTkComboBox)):
18144 |                     # Jangan terjemahkan dropdown bahasa itu sendiri.
18145 |                     if getattr(self, 'combo_lang', None) and child == self.combo_lang:
18146 |                         continue
18147 |                     curr_vals = child.cget("values")
18148 |                     if curr_vals:
18149 |                         new_vals = [self.tr(self.reverse_tr(v)) for v in curr_vals]
18150 |                         child.configure(values=new_vals)
18151 |                         curr_val = child.get()
18152 |                         child.set(self.tr(self.reverse_tr(curr_val)))
18153 |
18154 |                 self.update_ui_texts(child)
18155 |             except Exception:
18156 |                 pass
18157 |
18158 | if __name__ == "__main__":
18159 |     app = KHGTApp()
18160 |     app.mainloop()
```