Seaglass Globe™

Technical Brief

VERSION 1.2.4 · AUGUST 2026

Forged For Speed. Refined by Craft.

SEAGLASS FOUNDRY LLC

A precision WGS-84 virtual globe with an operator workstation, live compute, and an agent surface — the Forge suite's optional 3D view.

Executive Summary

Seaglass Globe™ is a native, GPU-rendered virtual globe built in Rust on wgpu: a WGS-84 ellipsoid held in double precision from orbital altitude to street level, with real elevation above and below the waterline, streaming imagery from any XYZ, WMS, WMTS or MVT service, and geodesy that names the algorithm behind every measurement. Around that engine sits a docked operator workstation — layer stack, catalog, measure and draw tools, a node-graph workflow editor that runs the ForgeGIS™ compute engine and re-drapes results while a parameter is still being dragged, saved views, print-size export, projects that persist on change — and beneath it a Model Context Protocol server through which software can fly the camera, place geometry, drape rasters and run compute on the same globe the operator is looking at.

20 000 km
to street level in one precision-stable descent — world space in f64, relative-to-centre rendering, reverse-Z depth with an infinite far plane

5 + globe
flat projections — equirectangular, Mercator, sinusoidal, orthographic, polar stereographic — switched live and persisted with the view

9
measure modes — distance, path, area, bearing, slant range, circle, square, ellipse, quad — Vincenty distance and Karney area on the WGS-84 ellipsoid, three unit systems

/mcp
one loopback MCP endpoint on the running application, bearer-protected in the suite install — camera, layers, entities, overlays, reports, ForgeGIS compute, ForgeData™ catalog

Within the Forge suite it is the optional, opt-in **3D view**: installed by the Seaglass Control Center™, started first by *Start suite* so ForgeMind™ adopts it as its drawing surface, with its own Assistant panel talking to the same agent. When the Globe is running, answers that produce something visual — contours, elevation profiles, draped rasters, scene entities — are drawn on the globe instead of only described. It runs on one Windows x64 workstation with a real GPU; its API, like the agent's, binds to 127.0.0.1 behind a per-install token.

This brief is a capability reference. After a short account of what the product is and how it is built, it catalogues what it does — rendering, navigation, geodesy and measurement, projections, imagery and layers, vector data, 3D content, live feeds, the workflow editor and compute, the catalog, entities and drawing, the agent surface, the digital twin, export, projects, the shell — then covers measured performance, security posture, deployment, and what is deliberately not claimed.

What Seaglass Globe Is

Three globe binaries — plus a stdio adapter — share one engine. **`seaglass-desktop`** is the product: a docked workstation with a menu bar, side docks, a canvas navigation strip and a status strip. Under **`--studio`** it additionally mounts the Assistant, Entities and Reports panels, spawns the ForgeGIS and ForgeData™ engines as stdio children, and serves the MCP endpoint — this is the globe the Forge suite installs and launches. **`seaglass-globe`** (`sgg-app`) is a lean instrument-cockpit surface with a floating HUD and no MCP dependencies, kept as the reference globe; the showcases share its surface. **`seaglass-mcp`** is the shell-less development and showcase server; it serves the same endpoint and the

same tools but is not staged by the suite. The two interface styles — docked shell and instrument cockpit — are supported permanently; neither is a migration target for the other.

Its role in the suite is precise. The engines are headless and ForgeGIS Studio™ is where an operator works them on a flat map; the Globe is where an answer wants the Earth. It runs its own copy of the GIS engines, which is why the suite asks for 8 GB of RAM with the Globe installed rather than 4 GB without.

It is not a general-purpose GIS desktop: it reads GeoJSON, KML/KMZ and shapefiles as layers with their attributes, but there is no attribute table, no style editor and no WFS. It is not a browser globe: it is a native binary that owns the GPU, which is what lets it hold precision relative to the camera, pin a Vulkan adapter for shared-VRAM interop with the compute engine, and keep rendering while a print-size export streams tiles behind it. And it is not sold separately: it is a component of the Forge suite, whose license is not yet finalized.

Architecture

Crates and process

The workspace is some twenty crates. `sgg-geodesy` is pure `f64` mathematics with no GPU, I/O or async — WGS-84 constants and conversions, Vincenty and Karney, the five projections, split-float precision, tile keys, frustum and picking. `sgg-tile` is the tile data plane (provider traits, the quadtree scheduler, the bundled BlueMarble and a procedural DEM); `sgg-net` fetches XYZ, WMS, WMTS, MVT, Terrarium and 3D Tiles and owns the disk cache and capability discovery; `sgg-render` is the wgpu renderer; `sgg-hud` the instrument cluster and theme tokens; `sgg-shell` the docked shell, command palette and workflow editor; `sgg-workflow` the workflow document — a real DAG of operations with named ports; `sgg-app-core` the runtime and plugin host (`Plugin`, `DataFeed`, `Pickable`, picking, ingest, measure, tours); `sgg-twin-wire` and `sgg-coresident` the digital-twin clients; the `seaglass-mcp-*` crates the tool implementations, the axum server, the server binary and the stdio adapter. Live feeds and demonstrations live in `examples/`, each its own crate, never in the product binary.

The render thread owns the frame loop. A shared multi-thread tokio runtime handles tile fetches (CPU-bound decode on a blocking pool), feed polling and the MCP server; no blocking work is reintroduced on the render thread. Cancellation rides tokens tied to a camera generation counter, so fetches for a view the camera has already left are dropped rather than drawn late.

The frame

There is no render graph. A frame is one hand-written linear encode of about twelve hundred lines with per-section timings: acquire the surface; prepare the visible set and upload budgeted tiles; the terrain pass, clearing to the sky colour and depth to zero; the overlay and decal section — polylines, twin rasters, heatmaps, surface fills, the surface-overlay composite, then in 3D only the walls, extruded polygons and glTF instances, then outlines and markers; egui, loaded rather than cleared and depth-free; one submit; screenshot or readback if requested; present. The renderable pipelines

behind that section — surface fill, heatmap, twin raster, arc tessellation (compute), marker, polyline, wall, extruded polygon and glTF — each keep a registry of what they draw.

Depth is reverse-Z in a `Depth32Float` buffer with an infinite far plane and an altitude-proportional near plane; there is no logarithmic depth buffer and none is needed. Surface overlays — polygon fills, heatmaps and the digital twin's raster epochs — are rasterised into per-tile overlay textures and composited by re-rendering the same tile meshes with a depth compare of *equal*. Because that is a bitwise integer compare, the terrain and overlay vertex shaders are contractually bound to produce bit-identical positions: both are marked invariant, both compute the same mathematics in the same order, and both consume one per-frame draw snapshot. Overlay colour is linear RGBA with premultiplied alpha, converted from sRGB by the piecewise transfer function; the composite pipeline state — equal compare, no depth write, zero bias — is locked.

Precision

ECEF coordinates are about 6.4×10^6 m; at that magnitude one unit in the last place of a 32-bit float is roughly half a metre, which is why single-precision globes shimmer at street level. World space stays in `f64` on the host — camera position and target are double-precision ECEF metres, and nothing is cast before the subtraction. Each frame chooses a render origin at the camera's ground-projection point, so magnitudes drop from megametres to metres; the per-tile translation is folded into the view-projection matrix in `f64`, recomputing only the translation column; and where a value must reach the GPU as a double it travels as a `hi`/`lo` split float that the shader reconstructs with one addition, buying roughly 24 extra bits. Shape rasterisation uses a second, separate relative-to-anchor scheme in lon/lat space. Six scripted camera paths — three 20 000 km → 100 m descents at 0°, 45° N and 75° N, a sixty-second in-and-out oscillation four times over, a one-kilometre pan at 100 m, and a snap from 10 Mm to 500 m with a no-NaN-frames check — remain bound to the number keys as the precision acceptance test.

Tiling, level of detail and streaming

Every tile carries its scheme: geographic (EPSG:4326, a 2 × 1 root, south-up rows) for the bundled BlueMarble base, Mercator (EPSG:3857, latitude clamped at ±85.051°) for OpenStreetMap, Esri, CARTO, Terrarium and, by default, every XYZ/WMS/WMTS/MVT source added by URL (XYZ and WMS sources may declare geographic / EPSG:4326). Three independent level-of-detail systems share no code by design: the terrain quadtree, descended by screen-space error to a 256 px target with a falloff beyond ten kilometres; 3D Tiles, driven by their geometric-error metric to a 16 px target; and the flat map, a direct metres-per-pixel zoom pick over a column/row rectangle capped at 1 024 tiles per frame with wrap copies drawn on either side of the antimeridian. Tile bounding radii are sampled on a 5 × 5 grid rather than the four corners; the visible horizon is bounded by $\sqrt{(2Rh + h^2)} \times 1.5$, floored at two tile edges; frustum culling uses a direction-aware swept sphere (9 km up, 11 km down along the surface normal) rather than an isotropic pad.

Streaming is budgeted: at most 64 tiles in flight, closest to the camera first; a provider slot suspended after eight consecutive failures so one dead server cannot stall the loop; at most four GPU uploads and sixteen mesh builds per frame. Texture and mesh caches are LRU with VRAM budgets — 512 MiB and 256 MiB by default, a separate 128

MiB overlay cache, both adjustable — with the three coarsest zoom levels pinned resident and parents prefetched for visible tiles. The disk cache is a plain filesystem hierarchy, `cache/tiles/{provider}/{z}/{x}/{y}`, written temp-then-rename, with no embedded database, no eviction and no expiration; a corrupt tile is re-fetched rather than served forever, and an undecodable-but-complete elevation tile is recorded rather than re-requested per sample. Tile fetches use their own HTTP client with a 15 s read timeout and a 30 s overall ceiling.

Picking

The live path casts a ray from the cursor computed entirely in `f64`, intersects the ellipsoid offset by the current elevation estimate — taking the far root when the camera is inside the offset spheroid, as it legitimately is in a canyon — and refines against terrain until the delta is under half a metre or six iterations. Results resolve `Renderable → Terrain → Ellipsoid → Nothing`, so a marker always wins over the terrain behind it. Every pickable — markers, vector features, extruded polygons, glTF models — implements one trait, and the primary window and every map view resolve through the same `PickView`, so a fix cannot land on one camera and miss another. Picking branches on the projection first: in a flat view a click resolves through orthographic unprojection and the projection's inverse.

One discipline worth stating

The architecture documentation's most important page maps duplicated mathematics: the horizon-cull test exists in three places that must agree — a marker shader, its host-side twin and the camera — plus a deliberately different fourth in the tile scheduler; elevation displacement exists twice, bit-exact, on the GPU; the haze curve exists three times. A missed copy shows up later as a picking failure that looks like a rendering bug. The repository's rule about itself is the one this brief follows: *if a page disagrees with the code, the code wins.*

Capability Catalogue

Rendering and terrain

- **Elevation** arrives as Terrarium-encoded tiles (`h = r·256 + g + b/256 − 32768`, 256 px, Web-Mercator, native zoom 0–15) decoded on the CPU, with heights above the ellipsoid; the mesh is displaced along the geodetic normal on the GPU, negative elevations included, so bathymetry is real geometry. Skirts drop 12 km along the edge normal to hide cracks between levels; tile meshes are 32-byte vertices with 256/128/64/32 segments per side by zoom.
- **Shading**: hill-shading in tangent space from the elevation gradient; sun-direction terrain lighting for the scene time; a **day–night terminator** with adjustable night floor and twilight width, drawn at the same place on the globe and — as of this release — on the flat map; an **ocean depth tint** from shallow cyan at sea level to deep navy at a 6 000 m reference, saturating in the trenches, with an intensity control.
- **Atmosphere**: altitude-adaptive haze and sky colour — a sea-level fog term with a 120 km extinction length decaying on the 8 km barometric scale height, so orbital views see a crisp Earth and ground views see horizon haze;

glTF models integrate the same density along their own ray, so an aircraft and the ground behind it recede together. Fog multiplier 0–4.

- **Imagery**: mip-mapped with 16× anisotropic filtering, so oblique views stay sharp; up to four imagery layers composited in one fixed-slot bind group with per-layer opacity.
- **Vector geometry** on the surface: polylines as screen-space-thick ribbons depth-tested against terrain, surface variants drape-sampled per vertex at 1 000 samples per degree and refreshed as deeper tiles arrive, absolute variants carrying their own altitude for orbits and flight paths; polygon fills triangulated and rasterised into the overlay path with a 0.2-alpha fill; markers and labels sharing one glyph atlas, horizon-culled. Densification is bounded by one 500 k-vertex budget per rebuild.
- **Diagnostics**: F5 hot-reloads the terrain shader with the previous pipeline kept live on a compile error; F8 tints overlay tiles; F9/F10 switch fill textures and patterns.

Camera and navigation

- **Camera model**: longitude, latitude and altitude in `f64`; heading, pitch and field of view in `f32`; pitch measured below horizontal. Default pose 0°, 45° N, 12 Mm, 60° FOV.
- **Keyboard flight** — all state values mutate continuously while keys are held: W/S along the view direction at a speed that scales with altitude; A/D strafe heading-relative and pitch-ignored so the strafe stays level looking straight down; Q/E yaw at ~90°/s; R/F ascend and descend multiplicatively (~+65 % / −39 % per second) so the feel is the same at 100 m and 10 Mm; 0 resets to home.
- **Mouse**: left-drag is an **anchor-preserving pan** — the terrain point under the cursor at drag start stays under the cursor, sensitivity scaled to height above ground; right-drag tilts and rotates; the wheel zooms along the gaze at ~36 % altitude per notch with eased accumulation; left-click picks.
- **Floors and ceilings**: 10 m above the ellipsoid plus a 30 m collision pad above the displaced terrain under the camera footprint — you cannot fly through a mountain; ceiling 200 000 km. The navigation strip clamps zoom at 30 m and 40 000 km.
- **Canvas navigation strip** (`seaglass-desktop`): zoom in (×0.6), zoom out, home, face north looking down, and a 2D/3D toggle whose label shows the destination; it removes itself when the canvas drops below 80 × 120 pt.
- **Views (bookmarks)**: save the current pose under a name; click to fly; persisted per project in `bookmarks.toml`. **`Go to…`** (`Ctrl+G`) searches saved views, loaded vector layers, imagery layers with a declared extent and catalog datasets — your data, not the world — and flies to the pick.
- **Canvas context menu** on right-click: the point's coordinates in the active format and its ground elevation, then *copy coordinates*, *fly to here* (keeping the current altitude), *measure from here*, *bookmark here* (named from the coordinates), *what is under the cursor*, and under `--studio` a sixth row, *load DEM here* (see the workflow section). Every action refers to the point resolved when the menu opened.

- **Tours**: TOML waypoint files with per-waypoint hold and easing (`Linear`, `EaseInOut`, `Ballistic` — auto-arcing the leg to a continental peak altitude), orbit waypoints, loaded with `--tour=PATH[,loop]`; on the lean globe a Pause/Play/Stop bar appears and T plays, pauses and replays; any input cancels. Descent legs interpolate altitude geometrically, so the apparent zoom rate is constant within and across legs.
- **FOLLOW / PILOT** on live-feed entities from the click-to-detail popup: FOLLOW tracks any entity at the current zoom and tilt; PILOT rides a model entity — aircraft, vessel, station — at its position and heading, pitched just under the horizon; both ease over dead-reckoned pose steps with a ~250 ms time constant and release on any camera input.
- **Extent framing** finds the shortest arc through the antimeridian and clamps altitude at one Earth diameter, so a global dataset frames the disc rather than the antipode.

Geodesy, coordinates and measurement

The datum is WGS-84 with its constants written down: a = 6 378 137 m, e^2 = 0.006 694 379 990 14, b = 6 356 752.314 245 179 m; flattening is deliberately recomputed rather than stored. Geodetic-to-ECEF uses the prime-vertical radius, with altitude as ellipsoidal height along the geodetic normal; ECEF-to-geodetic is Bowring's closed-form latitude with Heikkinen's height and a polar branch. The local frame is right-handed east–north–up. The geodetic normal — not the geocentric radial, from which it differs by up to about 11.5 arcminutes near 45° — is the direction along which elevation is displaced. Distance and azimuth are Vincenty's inverse (a port of the NGS `INVER1` routine, tolerance 0.5×10^{-13}, with an eleven-iteration cap that is part of the algorithm rather than a budget); polygon area is Karney's 2013 method through `geographiclib`, ellipsoidal to better than 0.001 %; slant range is straight-line ECEF and labelled as such.

Coordinate display. F4 cycles the cursor readout: decimal degrees (default), degrees–minutes–seconds with hemisphere letters, UTM by Snyder's forward formulas on WGS-84 (polar caps above ±80° report out of range; the truncation can leave 100–200 m of residual at non-special latitudes), and MGRS, whose 100 km grid-square letters are robust with only the within-square digits carrying that residual.

Measure tool. Reached from `View ▸ Measure`, the palette, or M; choosing a mode starts the tool and clears any half-drawn shape.

Mode	Vertices	Returns
Distance	2	Geodesic distance, forward and back azimuth — Vincenty on the WGS-84 ellipsoid
Path	≥ 2, Enter	Cumulative distance along consecutive vertices
Area	≥ 3, Enter	Closed-polygon area on the WGS-84 ellipsoid — Karney, < 0.001 %
Bearing	2	Initial and final azimuth
Slant range	2	3D ECEF distance between two terrain-elevation samples
Circle · Square · Ellipse · Quad	2–3	Centre-and-extent shapes; ellipse aspect by drag, default 0.6

Three unit systems switch live — metric, US customary, nautical (areas fall back to km²); path type is geodesic (default), rhumb line — always at least as long — or linear. Vertices drag individually with the readout recomputing live; Alt-drag translates the whole shape rigidly; right-click deletes a vertex (path and area); Backspace clears; a **freehand** toggle streams vertices at 12 px cursor spacing for tracing coastlines. Measurements **save and load as GeoJSON FeatureCollections** with the algorithm recorded in a `seaglass:measurement` property, so other consumers see plain geometry and the round trip preserves the method. Vertices render as a crosshair reticle. Escape clears, then stops the tool, and only quits when there is nothing left to cancel.

Projections and map views

Beside the globe are five flat views, cycled by P, set by `--projection=`, persisted with the view: equirectangular; Mercator (conformal, ±85.051°); sinusoidal (equal-area — the precondition for heatmap density to mean anything); orthographic; and north-polar stereographic (conformal, south of −30° cut). **All five are spherical**, using the semi-major axis as a sphere radius per the EPSG:3857 convention; the ellipsoid lives in the geodesy and the measure tool, so a 2D area is not the ellipsoidal answer the tool returns in 3D. One canonical transform maps longitude, latitude and altitude to world space in every mode; each renderable declares how it behaves flat — natively (tiles, markers, polylines), flattened to its footprint (extruded polygons), or hidden (walls, glTF models) — and the default is *hide*, so a renderable that has not considered the flat case disappears rather than drawing at a plausible wrong position. Horizon culling is disabled flat; wrap copies keep the map continuous across the antimeridian.

Map views. `View → New map view` opens another operating-system window onto the same globe with its own camera: drags, the wheel and the flight keys steer it independently; a chip shows its own cursor's latitude, longitude and elevation; a click opens that window's own click-to-detail popup, resolved exactly as the primary resolves it. The projection is renderer-global and stays a primary-window key. Map views are not dockable — a map view wants a monitor — and are closed by opening a project.

Imagery, elevation and the layer stack

Built-in providers: OpenStreetMap, Esri World Imagery and CartoDB Voyager (Mercator) for imagery; Mapzen Terrarium for elevation; an analytic Gaussian-hill DEM (`synthetic`) for offline tests; and — in a repository checkout — the bundled NASA BlueMarble base. Base selection resolves environment variable → the project's `layers.toml` → the workspace default → the built-in.

Sources added by URL — from the Layers panel's `+ add by URL…` form or the command line — persist to `layers.toml` when added through the UI:

Kind	What it takes
XYZ	A `{z}/{x}/{y}` template (`{-y}` for TMS); scheme `mercator` or `geographic`; max zoom (default 19); extension png/jpg/webp; attribution
WMS 1.3.0	GetMap base URL, `LAYERS`, CRS (EPSG:3857 default or 4326), format, styles, transparency, max zoom, attribution
WMTS 1.0.0	GetTile endpoint in KVP or RESTful addressing (RESTful for `{Time}`/`{TileMatrixSet}` templates), layer, tile-matrix set, style, format, version, time
MVT	A vector-tile template; decoded and rasterised client-side; building footprints extruded to 3D automatically; max zoom default 14

Discover for WMS and WMTS fetches `GetCapabilities`, parses it and offers a layer picker inline with its own filter; picking a layer pre-fills the form and the layer's declared extent is retained, so a regional service does not request tiles worldwide.

The layer stack — base at the bottom, overlays composited above, up to four imagery layers: drag-reorder; an eye that hides without losing the slider value; opacity 0.00–1.00 with a numeric readout; a `base` radio down the stack, so any layer can be promoted to base; a `default` pin that makes a provider the base a *new* project starts on, stored at workspace tier; `zoom` to a layer's declared extent where one exists; `×` to remove; per-provider attribution; a `session rasters` section under `--studio` for engine-held handles that can be draped or freed; `+ add layer` for any registered provider (added at 0.6 opacity), with `×` on a source you added that forgets it. Adding a layer whose tiling scheme differs from the base is refused with a toast; the outgoing base is hidden rather than removed; attribution follows the base. The stack — base, overlays, opacity, visibility, custom sources — persists to `layers.toml` on every change and is restored at launch. The runtime can **re-sample an imagery layer in place** for a source whose content changes under a stable URL; the storm-watch showcase uses this for radar every five minutes.

Vector files and feature inspection

Three formats with parity: **GeoJSON** (Point, MultiPoint, LineString, MultiLineString, Polygon, MultiPolygon); **KML 2.2 and KMZ** (Placemark, LineString, Polygon, MultiGeometry; KMZ unzipped in memory); **Esri Shapefile** (point, polyline, polygon with Z and M variants; the companion `.dbf` loaded automatically). Routes in: `Layer → Add vector layer…`, the Layers panel's `+ open file…`, and `--vector=PATH[,PATH…]` at launch — the same command the chooser dispatches, so scripted and clicked routes cannot diverge.

Loaded files sit in a **`vector files`** section of the Layers panel: a checkbox that draws or hides without unloading, the file's palette swatch, its name (full path on hover), a feature count — *"it loaded" and "it loaded something" are different claims* — `zoom` to extent, and `×` to unload with the file on disk untouched. Colours come from a fixed six-colour cycle validated all-pairs for colour-vision-deficiency separation over ocean and land, assigned lowest-free at load and kept across re-open, so colour follows the file rather than its rank. Re-opening a loaded path replaces it, so a file edited on disk is refreshed by opening it again. A file that fails to load gets a **report row** — its name and the parser's own root

cause (`Error while deserializing JSON: EOF while parsing a value at line 2 column 0`), with no controls a layer would have — whichever route named the file. Removing a file removes everything it drew.

Click-to-detail: every feature is pickable; a left-click opens a popup with its GeoJSON `properties` or KML `description`, geodetic location and altitude, re-projected every frame; Esc dismisses it ahead of quit. Not shipped: an attribute table view, styling, MVT-as-features, WFS.

3D content

- **glTF 2.0 / GLB**: binary and text loaders; embedded textures decoded to sRGB RGBA8; unlit and simple PBR materials; UV and flat-normal synthesis when attributes are missing; instanced draws with per-instance frustum culling; node-transform animation with linear, step and cubic interpolation evaluated per frame; screen-space sizing that holds an aircraft at a constant on-screen size while zooming, floored at life size close up, with a distance-falloff exponent per feed; picking through the same trait as markers. Deferred: skinning, morph targets, Draco/Meshopt, `KHR_lights_punctual`, advanced material extensions.
- **3D Tiles**: `tileset.json` parsing (region, box and sphere bounding volumes; replace/add refinement), b3dm and glb/gltf content, per-tile ECEF transforms, a per-frame screen-space-error traversal to a 16 px target with bounded-concurrency fetch, loading-skeleton replacement and LRU eviction; loaded with `--tileset=PATH[,PATH…]` on the lean globe binary.
- **Extrusion**: path walls (a great-circle curtain from a route to the surface) and extruded polygons with picking; MVT building footprints extruded per tile — the tallest by render height — with per-shape frustum and 5 km draw-distance culling and incremental per-tile add/remove.

Live data feeds

The framework is category-blind: it knows about entities appearing, moving and disappearing and nothing about vessels, aircraft or earthquakes. A feed implements one trait — id, display name, licence, spawn — receives the shared keep-alive HTTP client and a watch channel of the current viewport, runs as an async task, and pushes inserts, updates, removals, health and notices down a channel; it never touches renderer state. An update refreshes the entity's staleness clock, and going offline does not clear the map: entities age visibly until `stale_after` and the panel indicator turns amber. Filtering lives in the feed task; distance level of detail lives in the runtime, in pixels. Framework services include per-feed trail ring buffers drawn as lines and altitude curtains; a shared dead-reckoning helper feeds use between polls; and **distant-traffic symbols** — a model entity may declare a far marker, and once its airframe would draw below a pixel threshold the runtime swaps in a heading-rotated icon with a leader-line tag carrying callsign, level and live camera range, size-neutral at the swap, exactly one pickable representation at a time, suppressed in 2D.

Showcases — example crates run from a repository checkout, each labelled for what it is:

Showcase	What it draws
Earthquakes	USGS GeoJSON, polled; markers sized by magnitude and coloured by depth. Live: it shows whatever the feed says that day
Seismic month	The USGS thirty-day catalogue, GPU-splatted and **replayed** as a moving window — play/pause, speed, window length, kernel bandwidth, four colormaps, count-versus-energy weighting. Real data, replayed — not synthetic, not live (offline it falls back to a set labelled synthetic)
Storm watch	Active NWS alerts polled every 90 s as severity-sorted fills and centroid markers in eight hazard families, over the IEM NEXRAD composite as an uncached XYZ overlay re-sampled every five minutes; zone-only alerts counted rather than dropped
AIS	Digitraffic Baltic traffic as type-coloured 3D ships, dead-reckoned every 250 ms, six minutes of wake, moored traffic desaturated
Satellites	A CelesTrak element catalogue with a per-group disk cache of up to seven days, **propagated locally by SGP4 at 5 Hz** — propagated, not live — with a per-category filter and the ISS and CSS as models with along-track heading
Twin replay	The Spec D/E client replaying a recorded hydraulic model over terrain — see *The digital twin*

The workflow editor and ForgeGIS compute

The Workflow panel is a **node canvas over a real DAG document** with named ports and explicit edges, and also a step list; it opens in its own window (1 180 × 760) or docks, drawing the canvas when it has more than 620 pt of width and the step list when it has less. It is not opened at launch; `Processing → Toolbox…`, `View → Panels` and the palette open it, and a restored session reopens the last workflow.

Authoring. Drag the canvas to pan and scroll to zoom; drag cards to arrange (auto layout hands over to manual on the first drag without teleporting); drag port to port to connect, validated by the document — a cycle or an already-bound input is refused with the document's own reason; drag a fed input socket to pick an edge up and rewire or remove it; click an open socket to declare a **document input** named after the port; right-click a card for inspect, run this step, delete, show parameters or preview; right-click empty canvas for *add step here*, auto/manual layout and fit. Cards show a category rail coloured by hue rotation from the theme accent, the op name and category, then a preview image once run or the parameters, and a footer with status and elapsed time coloured against the median on the canvas. Input sockets are every port the op declares, so a join can be drawn before it exists.

The add-step menu lists the live ForgeGIS catalogue grouped by the engine's taxonomy and name-sorted, with a filter, favourites (persisted with the shell state) and recents; an op that cannot run on the document's lane is listed greyed with the reason. Without a compute bridge a small fallback catalogue keeps the editor usable; an empty catalogue never replaces a real one, because it means the engine is still warming.

Parameters. The rail form draws one control per parameter matched to its declared type — drag-value, checkbox, dropdown for a stated vocabulary (listed whole, with `auto` to unset), text — with a constraint line generated from the schema (`number 0 to 90, default 45`) and the description beneath; required parameters are starred and sorted first; overflow past six optional parameters folds. New steps are seeded from declared defaults, and a parameter that declares

none stays absent — *the engine decides* — rather than being seeded with zero. Blank optional parameters are dropped from the request; `0` and `false` are kept.

Scrubbing and nudging. The numeric value on a card is a horizontal drag surface: a parameter with declared bounds sweeps its whole range in about 250 pt of drag and clamps; unbounded values move 0.1 per point, integers 1. A **latest-wins pump** — never a queue — re-runs the freshest value whenever the engine is free (at least 350 ms apart) and always runs the value the drag ends on; the first hot run stages the input into an engine session handle so disk ingest is paid once; on a branching document only the scrubbed node's downstream subgraph re-runs, pinned handles standing in for the untouched prefix, and an independent branch is not re-run at all (measured: full run 817 ms, pin run 485 ms, suffix runs 229–238 ms). Each result re-drapes as it lands, keeping the opacity the operator set. Arrow keys nudge the value under the pointer — or the last one touched — by the parameter's smallest unit, Shift for ten. Booleans and vocabularies change on click; free text stays in the rail.

Running. A single chain runs through the engine's `run_pipeline`; anything else — a join, a fork, several sources — runs through `run_dag`; the editor says which. On disk the document is ForgeGIS's own `SavedPipeline` — v1 for a plain raster chain, v2 when the document branches, declares named inputs or runs on another lane — saved as `workflows/<name>.json` with the source path in a sidecar so the document stays portable. Run is enabled exactly when a runnable plan exists, and when it is not the reason sits under the button as a sentence: *read_raster does not say what to read. Select it and set a source path.* A `>` on each step runs that step alone on the source. A `colour` row decides how the result renders — `auto` follows the last op, or a chosen colormap, or `none`. Every named output is rendered, and a chain that ends in geometry draws features as a scene entity.

Batch runs the authored chain over every raster in an input folder, one input per file, writing under a configured write root, with a job list and per-job cancel; per-file failure reasons travel with the row. A branching document disables batch with the reason rather than being flattened. **Contours**: type an interval, press *Contours*, and `generate_contours` runs against the document's source, drawing labelled lines at their true elevation as a scene entity — re-running replaces it. **Load DEM here**: right-click terrain, and the catalog ranks the best elevation dataset covering the point, extracts a window sized to the ground span you were viewing (half of it, clamped 2–200 km, capped 4 096 px) to a local GeoTIFF, and opens the document with that file as its source — root selected, pick announced by name, resolution and tier.

How results land. Prose and stats go to the Reports panel; a GeoJSON feature collection becomes a scene entity, reprojected on the way in; a raster handle is draped through a CRS gate — EPSG:4326 as-is, anything else reprojected by ForgeGIS, never in Rust, and skipped with a note if it cannot be — resampled by the engine above 4 096 px, written as GeoTIFF into the workspace and composited at 0.85 opacity; anything else reports *ran — see report*. Intermediate handles are closed on the way out, success or failure.

The catalogue reachable from the canvas is the attached engine's, fetched at warm-up and grouped by the engine's categories (consolidated here). Categories and representative operations:

Category	Representative operations
Terrain	Slope · Aspect · Hillshade · Curvature · Roughness · Rugosity · TPI · TRI · VRM · Relative elevation · Heat-load index · Surface-area ratio
Hydrology	Sink fill · Flow direction · Flow accumulation · Stream network · Stream order · Watershed · TWI · HAND · Snap pour points · Resolve flats
Visibility · Solar	Viewshed · Batch viewshed · Intervisibility · Fresnel zone · Insolation · Shadow · Horizon angle · Sky-view factor
Filters · Algebra · Statistics	Gaussian, median, majority, focal, anisotropic diffusion · Edge detect · Morphology · Nibble · Sieve · Fill no-data · Band math · Map algebra · Reclassify · Zonal statistics · Histogram · GLCM
Classification	K-means · Max likelihood · Random forest · DBSCAN · HDBSCAN · Spectral angle mapper · Segment · Connected components · Landform · Benthic terrain
Interpolation	IDW · Kriging · Spline · TIN · Natural neighbour · KDE · Trend surface · Gridders (average, count, min, max, range)
Multi-raster · Geometry	Blend · Mosaic · Change detect · Pansharpen · PCA · MNF · Tasseled cap · Spectral unmixing · Cut/fill volume · Clip by mask · Reproject · Resample · Rescale · Vertical-datum convert · Band select/concat
Distance	Euclidean distance and allocation · Cost distance · Least-cost path · Corridor · Buffer
Bathymetry	Bathymetric position index · Benthic irradiance · Depth uncertainty · Isopach · Sound-velocity profile · Tidal current · Wave exposure · Dredge volume · Backscatter classify
Vector · Vector geometry	Polygonize (raster stamp) and polygonize to vector features · Contour (raster) · Footprint · Cutline · Buffer, variable buffer · Clip, union, difference, intersection, symmetric difference · Dissolve · Convex hull · Centroid · Voronoi · Simplify · Make valid · Field calculator · Extract by expression · Select by location · Vector SQL · Reproject
Point cloud · Temporal · Encoding	Ground filter · Classify · Rasterize · Canopy height model · Temporal statistics · Temporal anomaly · RGB encode

Each operation's parameters, constraints, defaults and input ports are read from the engine's schema; the constraint text on screen and in the generated reference come from one function, so they cannot disagree.

The catalog and ForgeData

The **Catalog panel** browses what ForgeData knows about, as a `category > source` tree built by the host with count badges, a case-insensitive name filter, two-line rows (name, then category · resolution · format) and a tier chip — `LOC`, `NET` or `?` — saying where the dataset lives and what it costs. A **single click loads**: a raster drapes on the globe (engine-resampled, band-aware colormap), becomes the workflow document's source and reveals the editor with the root selected; a vector draws its features and reveals Entities. Double-click flies to the dataset's declared extent; right-click offers *zoom to extent*, *use as source* without loading, and *show on globe* for vectors. Its status line reads `warming…`, `unavailable: …`, or `no compute bridge — launch with --studio`.

ForgeData through the bridge — where ForgeGIS computes, ForgeData finds and serves data — lands each capability where it belongs:

Capability	Lands as
Catalog browse and search, dataset detail, windowed raster reads	Report entries; extracted windows become engine handles
Vector features in a bbox; road lookup by name	Scene entities
Elevation at a point, along a profile, over a region	Markers and a 3D profile line
Best-dataset ranking for an operation	Report
`analyze_area` — best dataset → extract → compute → drape	Raster overlay and report; the shortest path from *look at this place* to a rendered analysis
Data-source ingest and sync (local directories, STAC, WCS, PostGIS, ArcGIS REST, Overpass, OpenTopography, NOAA, Copernicus CDS, TIGER, Terrarium)	Report
Publish a file or vector back to the catalog; storage layout	Report — dedup by SHA-256 into managed storage, immediately searchable
Geocoding; LiDAR elevation and density; roads in a region or near a point	Markers and scene entities
LiDAR draped as a surface (DSM, DTM, intensity or density)	Draped surface overlay

ForgeData's roots are merged into ForgeGIS's read roots at startup so a catalog-to-compute flow does not fail inside the engine's path sandbox. Extracts are cached on disk by dataset, bbox, size and format; deleting the cache is always safe. Catalog terrain *serving* is deferred pending a PMTiles writer on the engine side; two catalog tools are not yet bridged.

Entities, drawing and scene

The **draw tool** — `View ▸ Draw`, the Entities panel's buttons, or 1/2/3 while armed — places points, lines and polygons by click, finishes on Enter or double-click, pops one vertex on Backspace, and shows a rubber band; naming the finished shape (64 characters, duplicates refused) commits it as a **scene entity of the same kind the agent places**, listed beside them and referenceable by name in chat. Committed points render as a small disc; committed polygons carry a 0.2-alpha fill (capped at the 64 oldest so a bulk publish degrades to outlines). Draw and measure are mutually exclusive.

The **Entities panel** (`--studio`) leads with the draw buttons, then everything placed in three sections — scene entities, overlays, rasters — sorted by creation time: click a row to fly to its bounding box; double-click a name to rename in place; `×` arms as *sure?* and removes through the same path the MCP tools use; rows wear their catalog state (`#id` published, ~ in flight, `!` failed with click-to-retry); overlays with a TTL show remaining time.

Persistence. When the ForgeData catalog is up, every scene entity is published as a deterministic one-feature GeoJSON tagged `scene-entity`; the row id rides the chat scene context; deletes tombstone through the undo log so an undone delete rebinds the same id; renames patch the row; and on every catalog restart orphaned rows re-adopt — entities survive a restart with the same id. Without a catalog everything degrades to in-memory behaviour, quietly.

The agent surface

The MCP server is an axum application implementing the MCP Streamable HTTP transport, revision 2025-03-26: one `POST /mcp` endpoint accepting JSON-RPC 2.0 for `initialize`, `ping`, `tools/list` and `tools/call`; notifications answer `202 Accepted`; there is no server-sent event stream because every tool is short-lived. It binds `127.0.0.1:7900` by default (`--port=`), is spawned before the shell and the GPU come up so the port is open at about half a second, retries a busy port every 250 ms for ten seconds, and stays non-fatal — a desktop that cannot serve agents is still a globe. The model is **attach, don't spawn**: a stdio-to-HTTP adapter proxies to a running globe for clients that speak stdio — ForgeMind included — because the user owns the window and the agent joins it. Every object-shaped result is also sent as `structuredContent`; request bodies up to 64 MiB; each call bounded by a 30 s timeout and a panic guard.

The tool surface, by group:

Group	Representative tools
Camera	`fly_to` (duration, easing, heading, pitch) · `set_camera` · `get_camera` · `zoom_camera` · `pan_camera` · `fit_bounds`
Cosmetic and diagnostics	`toast` · `screenshot` (path checked up front, capture on the next frame) · `set_projection` · `reload_shaders`
Layers and providers	`set_active_imagery` · `set_active_elevation` · `add_imagery_layer` · `remove_imagery_layer` · `reorder_imagery_layers` · `set_layer_opacity` · `toggle_renderable`
Scene and vector display	`add_overlay` (transient, TTL) · `clear_overlays` · `publish_scene_entity` (durable, named) · `list_scene_entities` · `rename_scene_entity` · `delete_scene_entity` · `clear_scene_entities` · `undo_last`
Reports and raster overlays	`push_report` (markdown) · `add_image_overlay` (an already-colourised raster over a WGS-84 bbox, by path or URL) · `remove_overlay` (and its legacy alias)
ForgeGIS operations and handles	`list_ops` · `get_op` · `apply_op` · `load_raster` · `list_handles` · `close_raster` · `drape_handle`
ForgeGIS pipelines and batch	`run_pipeline` (flat chain) · `run_dag` (SavedPipeline document) · `save_pipeline` · `apply_pipeline` · `list_pipelines` · `get_pipeline` · `delete_pipeline` · `submit_batch` · `get_job_status` · `cancel_batch`
ForgeData catalog, vector, elevation	`list_datasets` · `search_datasets` · `get_dataset` · `read_raster_window` · `vector_features_in_bbox` · `lookup_road_by_name` · `elevation_at_point` · `elevation_profile` · `elevation_in_region` · `find_optimal_for_operation` · `analyze_area`
ForgeData ingest, publish, location, LiDAR	`list_data_sources` · `add_data_source` · `sync_data_source` · `publish_file` · `publish_vector` · `get_storage_layout` · `geocode_address` · `roads_in_region` · `roads_near_point` · `lidar_elevation_at_point` · `lidar_density_in_region` · `drape_lidar`

Compute tools **fire and queue**: they return `{"ok": true, "queued": true}` immediately and the overlay lands a moment later on the UI thread; the outcome is read from `list_handles`, `list_scene_entities` or the report. Tool descriptions are hand-written agent-facing prose, because the description decides whether a tool is picked. Not yet on the surface: reading layer state back (`list_layers`), `pick_at`, and tour/bookmark tools.

Assistant and Reports. The Assistant panel (`--studio`) is the conversation with ForgeMind (default `http://127.0.0.1:8080`), streaming the agent's narration — started, iteration, tool call, tool result, completed, error — while camera and overlay changes arrive separately over MCP; a `new` button clears the transcript and scene. The Reports panel renders markdown the agent pushes, with previous/next paging.

The digital twin

The Globe is the remote consumer of the ForgeGIS digital twin's published wire contract — Spec D, frozen additive-only since 2026-07-21 — and never embeds the engine. The client reads a framed TCP stream of raster and vector layer deltas (base64 ISO-WKB geometry, raw-deflate payloads); unknown header fields are ignored, unknown frame types fail loud, prime frames are excluded from ordering and completeness accounting. It was implemented from the specification text alone, and a live flood rendered on 2026-07-22 with zero client changes. Raster epochs land as overlay textures through the same depth-equal composite as everything else, colormapped on the GPU, and live and replayed runs render identically because a replay is indistinguishable from live by design.

Co-residency (Spec E) is the case where the engine and the globe share one GPU: raster epochs never cross a socket but land in shared device-local VRAM rings and are sampled in place — zero copy, header-only frames. To make that possible the renderer pins Vulkan rather than letting wgpu choose, because the shared-surface path is opaque Win32 external memory between the engine's OpenCL context and the globe's Vulkan device and DirectX 12 has no import path. The mode is opt-in and off by default, and every error, refusal or end-of-stream in it ends in working wire-only rendering, loudly — co-residency is an optimisation over Spec D, never a dependency. Its conformance probe must re-run on every wgpu or driver bump; it last passed on 2026-07-25 on wgpu 29.0.4, six iterations bit-exact. In the twin showcase, the decal's sun-azimuth shading slider — a parameter riding immediates over a shared surface — re-lights the flood surface continuously while the frame and co-residency counters do not move: no re-upload and no wire traffic. The demonstration flood is a hydraulic model, not a real event.

Export and screenshots

`File → Export image…` writes the canvas area as a PNG — exactly what the canvas shows. `Export at resolution ▸` offers 2×, 4× and 8×, each labelled with the pixels it produces; it renders the scene again off-screen — bounded by the GPU rather than the monitor, without in-canvas chrome, attribution composited in — and **fetches the imagery its resolution implies**, with a progress readout and cancel. Two fallbacks are named rather than silent: a 30 s ceiling writes at screen detail, and a set too large for the tile cache is refused up front with the largest size that would fit (*8x needs 4351 tiles, the cache holds 1635 — 4x would fit*). Screenshots are also available from the command line and from the agent surface, with the outcome toasted when the copy has run.

Projects, state and persistence

A **project is a working directory** carrying a `project.toml` marker — a name and a format version, nothing else. There is no *Save project* because nothing in the application has a dirty flag: seven files and a `workflows/` directory, all in the working directory, are written on change rather than at exit — `project.toml`, `shell_state.toml` (look, active tabs,

hidden panels, the placement of only the panels the user moved), `panel_layout.toml`, `view_state.toml` (projection and camera pose, hand-editable), `layers.toml`, `bookmarks.toml`, `settings.toml`, and `workflows/<name>.json`. `New project…`, `Open project…` and `Open recent ▸` (up to ten, recorded at startup, path elided in the middle, `×` to forget) reload the window into the chosen folder, carrying the layer stack, camera pose, arranged frame, torn-off panels and window geometry; recents live at user level in `<user config>/recents/`, one file per project. Naming or renaming a project is a small modal, and a marker written by a newer format version is refused rather than downgraded.

The shell

- **Menus**: File (projects, export, quit), View (Measure ▸, Draw ▸, panel checkboxes, New map view, Panel placement ▸, Reset camera, LOOK presets and the four look axes), Layer (add imagery layer, add vector layer, add vector tiles, zoom to layer extent), Processing (Toolbox…, Run pipeline…, Batch…). Menus close when the action is done, not on click; disabled items state their reason on hover.
- **Command palette** (`Ctrl+K`): the palette and the menu bar render one registry, so a command cannot exist on one and not the other; substring matching over title, group and keywords; a command that cannot run now is listed greyed with its reason; ↑/↓, Enter, Esc.
- **Docks and placement**: left (Layers, Catalog), right (Settings, Views, plus Entities and Assistant under `--studio`), bottom (Reports under `--studio`); members tab rather than split; only the left dock opens on a fresh profile. Every panel can be placed in any dock or torn off into a real operating-system window sharing the GPU device and queue; closing the window re-docks it. Docks resize from their outer edge (180–560 pt sides, 120–480 pt bottom).
- **Look**: four orthogonal axes — palette (Studio Neutral, Cold Cyan, Tactical Amber, Neon Phosphor, Hot Magenta), surface (Solid, Translucent, Glass), density (Compact, Comfortable), typography (Proportional, Monospace) — with Studio and Tactical presets and `--look=` on the command line. Tokens are resolved once per frame into egui's global style, so stock widgets inherit the look. The globe renders across the whole window with docks painted over it; with solid docks the optical axis shifts so the subject stays centred.
- **Status strip**: camera latitude, longitude and altitude and the active look; opt-in performance meters — FPS (green ≥ 58, amber ≥ 30), frame time, VRAM against budget (green under 70 %, amber to 90 %), tiles drawn/visible.
- **Input**: F2–F10 diagnostics; M measure; P projection; 1–9 modes; Ctrl+K and Ctrl+G; WASD/QE/RF/arrows to fly; an Esc ladder that dismisses, cancels and only quits with nothing left to cancel; letter keys skipped while a text field has focus. Right-click on the globe opens the context menu; on the workflow canvas, its own; on panel furniture, reset.
- **Feedback**: toasts (4 s, at most four, identical messages folded with a count) in the tooltip layer above every panel; a scale bar measured through the camera's own ray-cast in the measure tool's active units; the active provider's attribution, never suppressed.

Settings, caches and diagnostics

The Settings panel holds the knobs that genuinely vary by machine, each with a one-line explanation: detail falloff 1.0–5.0; LOD cap 10–22; atmosphere on/off and fog multiplier 0–4; day/night terminator with night floor and twilight width; ocean depth tint and intensity; imagery and mesh VRAM budgets (64–4 096 MiB and 32–2 048 MiB, applied at next launch); the status-bar performance readouts. Sliders under an unchecked master toggle stay visible so the user can see what re-enabling would restore. Frame instrumentation is built in — `SGG_PERF_DUMP=<frames>` dumps a phase-timing breakdown every N frames, and a slow-frame accumulator speaks only when a frame is over 25 ms — and a soak use case verifies that neither VRAM, the tile cache nor the frame time creeps over a long session.

Performance: What Has Been Measured

The performance record is measurements, not budgets, and its own caveat applies: every number came off one machine — an NVIDIA RTX 5070 Laptop GPU under Vulkan at 1600 × 900 on a ~193 Hz panel — and *a profile is a photograph, not a law.*

Scenario (before, ms per frame)	frame	acquire	prep	terrain	overlay	egui	submit
Home view, idle, all panels	5.19	3.26	0.20	0.06	0.06	1.13	0.49
Same, every panel hidden	5.49	4.38	0.11	0.05	0.05	0.45	0.38
Zoom 12, 110 tiles, idle	5.62	3.05	0.43	0.10	0.07	1.14	0.63
4 400 vector features, idle	5.31	2.62	0.23	0.06	0.24	0.68	0.49

With FIFO presentation the wall-clock frame is pinned to the display and *acquire* is waiting, not work; the interesting number was CPU per frame — about 1.9 ms at ~193 fps, roughly 37 % of a core on a still scene. That unconditional repaint is fixed: idle settles to a floor of about 15 Hz, frames in a 22-second idle fell from roughly 4 340 to roughly 395, while a scripted drive holds 180 fps and an agent driving over MCP saw 213. Behind it, in rank order: loading a 4 400-feature GeoJSON went from 233 ms to 44 ms (5.2×) once elevation sampling was bounded to the drawn zoom and a per-line synchronous GPU round trip became a CPU `f64` path; the direction-aware swept-sphere cull took a pitched view over Everest from 283 tiles at 8.0 ms to 169 at 5.5 ms for a pixel-identical picture; a per-frame catalogue rebuild in the `--studio` host loop fell from 295 µs to 3 µs; visible-set caching for a still camera cut prep 2.4–2.8×; surface-polyline tessellation sped up 9.2×; and vertex densification for agent geometry was bounded to a 500 k budget (43.7 s → 0.98 s on 1 173 features). Measured and found fine: 4 400 vector features cost nothing per frame; marker upload is linear at ~2.2 µs each; mesh build, terrain pass and present are noise. Scrub reruns on a DAG: 817 ms full, 485 ms pin, 229–238 ms per suffix. No per-frame budgets are declared anywhere in the shipped code; declared, regression-tested budgets are a stated roadmap item.

Security Posture

Everything runs locally. The MCP endpoint binds `127.0.0.1`; when a token is configured (`--token=` / `SEAGLASS_MCP_TOKEN`), an axum middleware refuses any request without a matching `Authorization: Bearer` header — an empty 401 before body handling, a constant-time comparison, one warning line with the peer address and no token material. In the suite, ForgeMind's HTTP API requires a per-install bearer and the Globe's MCP server enforces a second, distinct token in the opposite direction; neither API is exposed to the LAN, and neither is open once the Control Center has minted its tokens. Tokens reach the processes by environment, never on the command lines ForgeMind parses and spawns; LLM provider keys live in the Windows Credential Manager, never on disk. The bind is the rest of the boundary today: per-tool allow-listing is deferred, and a hand-launched globe with no token configured has none — do not expose the port or put it behind a tunnel without setting a token first.

Blast radius is contained by construction: every tool call runs under a panic guard and a 30 s timeout; the child engines run under ForgeGIS's path sandbox with explicit read and write roots (the Globe's workspace force-added to both); batch output must fall under a write root; a raster the engine cannot reproject is skipped with a note rather than placed wrongly; the chat panel's bearer is attached per request rather than as a client default, so it cannot leak into the fetch that draws whatever URL an agent names. The tile cache and engine workspaces are plain directories under the working directory. Operating with the network fenced: the tile cache serves any region already visited and never expires; the suite ships a small offline elevation sample so slope and elevation work the instant it is installed; any imagery service you host on your own network is added by URL and drawn the same way; your own elevation data enters through the ForgeData catalog. This brief makes no claim about air-gap accreditation; it describes what binds where.

Deployment

The Globe is a component of the Forge suite for **Windows x64**, installed and started by the Seaglass Control Center. The installer checks disk, RAM, GPU and ports before it begins — RAM and GPU are advisories, but the Globe wants a real GPU and the full suite wants 8 GB — lays down a private Temurin JRE 25 for the engines and the two native Globe binaries (`seaglass-desktop.exe`, `seaglass-mcp-stdio.exe`) beside them, and needs no prerequisites. *Start suite* launches the Globe first as `seaglass-desktop.exe --studio --port=7900`, then ForgeMind — which spawns ForgeGIS and ForgeData as its own tool servers — then Studio; start the Globe after ForgeMind and ForgeMind must be restarted, because it picks its drawing surface at launch. Because the Globe spawns its own engine pair, an install with the Globe runs two pairs and the Control Center plans heaps for six services rather than four, reserving headroom for the Globe's tile cache and driver working set. An RDP or remote-desktop session usually cannot drive the GPU; leave the window open and unminimised while the agent works, because a minimised globe stops rendering and drawing requests time out. Every binary answers `--version`.

Flag / variable	Effect
`--studio`	Mount Assistant, Entities, Reports; spawn the engines; serve `/mcp`
`--port=`, `--token=` / `SEAGLASS_MCP_TOKEN`	MCP endpoint port (default 7900) and bearer
`--projection=`, `--look=studio|tactical`, `--tour=PATH[,loop]`	Startup projection, look preset, flythrough
`--vector=PATH[,…]`, `--measurements=PATH`	Open vector files / a saved measurement file at launch
`--screenshot=`, `--ui-script=`, `--window=x,y,w,h`	One-shot capture, scripted drive, window geometry
`SGG_IMAGERY`, `SGG_ELEVATION`, `SGG_CONFIG`	Base providers and config path (override the project)
`FORGEGIS_READ_ROOTS` / `FORGEGIS_WRITE_ROOTS`, `FORGEGIS_RATE_LIMIT`, `FORGEGIS_JAVA_OPTS`	Engine path sandbox, request rate, JVM options
`FORGEDATA_CATALOG` / `FORGEDATA_CATALOG_SET`, `FORGEDATA_DOWNLOAD_DIR`, `FORGEDATA_DATA_ROOT`	Catalog selection (the first wins if both are set), download and data roots
`FORGEMIND_API_TOKEN` / `--forgemind-token=`, `FORGEMIND_URL` / `--forgemind-url=`	The Assistant's bearer and endpoint (default `http://127.0.0.1:8080`)
`FORGEGIS_TWIN_CORESIDENT_PORT`	Spec E co-residency endpoint (the Spec D wire client attaches with `--server=host:port`)
`SGG_TILE_CACHE_DIR`	Tile-cache root (default `cache/` under the working directory)
`SGG_PERF_DUMP=<frames>`, `SGG_RENDER_EPOCH`	Frame-timing dump every N frames; pinned render epoch

Specifications at a glance

Role	Optional 3D view of the Forge suite; the agent's presentation surface
Binaries	`seaglass-desktop --studio` (the product) · `seaglass-mcp-stdio` (adapter) — native Rust, wgpu 29, winit, egui; no JVM for the globe itself
Platform	Windows x64. GPU features `PUSH_CONSTANTS` (immediates in wgpu 29) and `FLOAT32_FILTERABLE` — essentially any modern discrete GPU; backend Vulkan or DirectX 12 as wgpu selects (Vulkan pinned in co-resident twin mode). Validated on an RTX 5070 Laptop GPU under Windows 11; other platforms are not validated for this release
Memory & disk	8 GB RAM for the suite with the Globe installed (4 GB without); ~4 GB disk for a tile cache that grows with use
Endpoint	`POST /mcp` on `127.0.0.1:7900` (configurable), MCP Streamable HTTP rev. 2025-03-26, bearer-authenticated per install; ForgeMind on 8080; engines over stdio
Data in	XYZ · WMS 1.3.0 · WMTS 1.0.0 (KVP and RESTful) · MVT · 3D Tiles (b3dm, glb — lean globe binary) · glTF 2.0/GLB (feeds and models) · Terrarium elevation · GeoJSON · KML 2.2/KMZ · Esri Shapefile (+DBF); the tile cache serves visited regions without a network

Projections	WGS-84 globe; five spherical flat views (equirectangular, Mercator, sinusoidal, orthographic, polar stereographic); independent map-view windows
Coordinates & measure	DD · DMS · UTM · MGRS readouts; nine measure modes; Vincenty inverse and Karney area on the ellipsoid; metric, US, nautical; geodesic, rhumb or linear paths; GeoJSON round-trip
Compute	ForgeGIS operations, chains (`run_pipeline`) and DAGs (`run_dag`) from a node-graph editor with live parameter scrubbing, batch, contours, load-DEM-here; ForgeData catalog, extraction, elevation, publish; results draped, drawn as entities, or reported
Digital twin	Spec D wire client; Spec E shared-VRAM co-residency (opt-in, Vulkan-pinned, wire-only fallback)
State	A project is a working directory; layers, view, panels, bookmarks, settings and workflows are small files written on change; recents at user level
License	Proprietary. Component of the Forge suite, not sold separately; the suite's license is not yet finalized

Known Gaps and What Is Not Claimed

- **No operator, dataset or tool counts** appear in this brief. The compute catalogue and the catalog are the attached engines'; the tool surface is generated from the build; a number printed here could not be re-verified once the document ships.
- **The flat projections are spherical.** "WGS-84 ellipsoid" is true of the geodesy and the measure tool and false of the five map projections.
- **Satellites are propagated, not live**; the seismic showcase is real USGS data replayed; the twin flood is a model; live feeds show whatever their sources said that day. The showcases and the bundled BlueMarble base are example crates and assets in a repository checkout — the suite install stages the two binaries.
- **Platforms.** wgpu can target other backends and operating systems, but macOS, Linux and AMD GPUs are recorded as *should work, not yet tested*; this release is Windows x64 on a validated NVIDIA configuration.
- **Performance budgets** are not declared; the figures above are single-machine measurements.
- **Agent surface gaps**: layer state cannot yet be read back; `pick_at`, tour and bookmark tools are absent; per-tool allow-listing is deferred.
- **Data**: GeoJSON, KML/KMZ and Shapefile only — no MVT-as-features, no WFS, no attribute table, no styling; polygon holes are not yet plumbed through ingest; vector files are not persisted across launches; 3D Tiles load only through the lean globe binary's `--tileset` flag — the docked shell has no tileset or glTF loader of its own. Catalog terrain serving is deferred; two catalog tools are unbridged; the Entities panel lists placed scene entities, not the catalog's dataset list. Map-view windows cannot be docked and are not persisted.

- **Shell**: no splitters inside a dock and no drag-to-rearrange tabs; recents cannot be pinned; the 1× canvas export includes the navigation strip; keybindings are effectively exhausted, which is why there is a palette.
- **Removed in this release cycle and not features**: the Time panel and simulation clock, CZML time-aware layers, sun and moon billboards, the particle system and the launch demonstration; the Ops panel was retired in July, its parts now living in Workflow, Layers and Catalog.
- **Roadmap is direction, not commitment**: annotations, tour authoring, point clouds, a celestial sphere and other bodies, symbology and sensor-feed modules, and web, XR and collaborative targets are listed for orientation.

Licensing and Contact

Seaglass Globe is proprietary software of Seaglass Foundry LLC — no license is granted except under a separate written commercial agreement — distributed as an opt-in component of the Forge suite through the Seaglass Control Center. It is not sold separately, the suite's license is not yet finalized, and nothing in this brief should be read as a grant of rights. Third-party open-source components retain their own licenses and are enumerated in the repository's generated third-party notices — 455 components under permissive or file-level weak-copyleft terms (one MPL-2.0), with a license check that errors if a strong-copyleft dependency ever appears. The application bundles JetBrains Mono under the SIL Open Font License 1.1.

Imagery and data attribution is drawn in the application's attribution strip and repeated here. Imagery: © OpenStreetMap contributors · Esri, Maxar, Earthstar Geographics, and the GIS User Community · © CARTO · Mapzen Terrain Tiles, hosted by AWS Open Data · NASA Earth Observatory · NASA GIBS. Data: USGS · NOAA/NWS · Iowa Environmental Mesonet NEXRAD · Digitraffic (Finnish Transport Infrastructure Agency) · CelesTrak. Their tile-usage and data policies apply when you use them.

Contact. rich@seaglassfoundry.com · seaglassfoundry.com — Seaglass Foundry™ reads every email. Evaluate the Forge suite on one Windows x64 workstation with a real GPU, and watch the agent draw its answers on the globe.
