Shooting for Photorealistic 3DCG with Navara

Our Journey Begins

Piyush Chauhan
Software Engineer, Eukarya Inc.

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Who we are?

About Eukarya

Eukarya Inc. is a Japanese company primarily focused on supporting digital archives and intellectual activities. Our mission is to promote the organization and utilization of data that transcends analog and digital boundaries, aiming to create a world rich in intellectual creativity.

What is a Map Engine?

Map Engine

A map engine is a software component that transforms geographic data into interactive, visual map representations on the web.

Core functionalities

- Data Processing & Visualization
- Coordinate System Management
- Tile System Handling
- Vector/Raster Rendering

User Interactions

- Pan & Zoom Controls
- Layer Management
- Geographic Queries
- Real-time Updates

Popular Examples

CesiumJS (3D), MapLibre GL JS, OpenLayers, Leaflet

The internals of existing web map engines such as CesiumJS and MapLibre GL JS, organized by function, can be divided into four components.

Library
Used by application developers to control the map engine externally

Rendering Engine
Renders the final display based on the map engine's state

Main loop
Changes the internal state of the map engine based on user input

GIS Engine
Performs GIS-specific processing

GIS Engine

A critical component of a Map Engine that processes and prepares geospatial data for rendering and interaction.

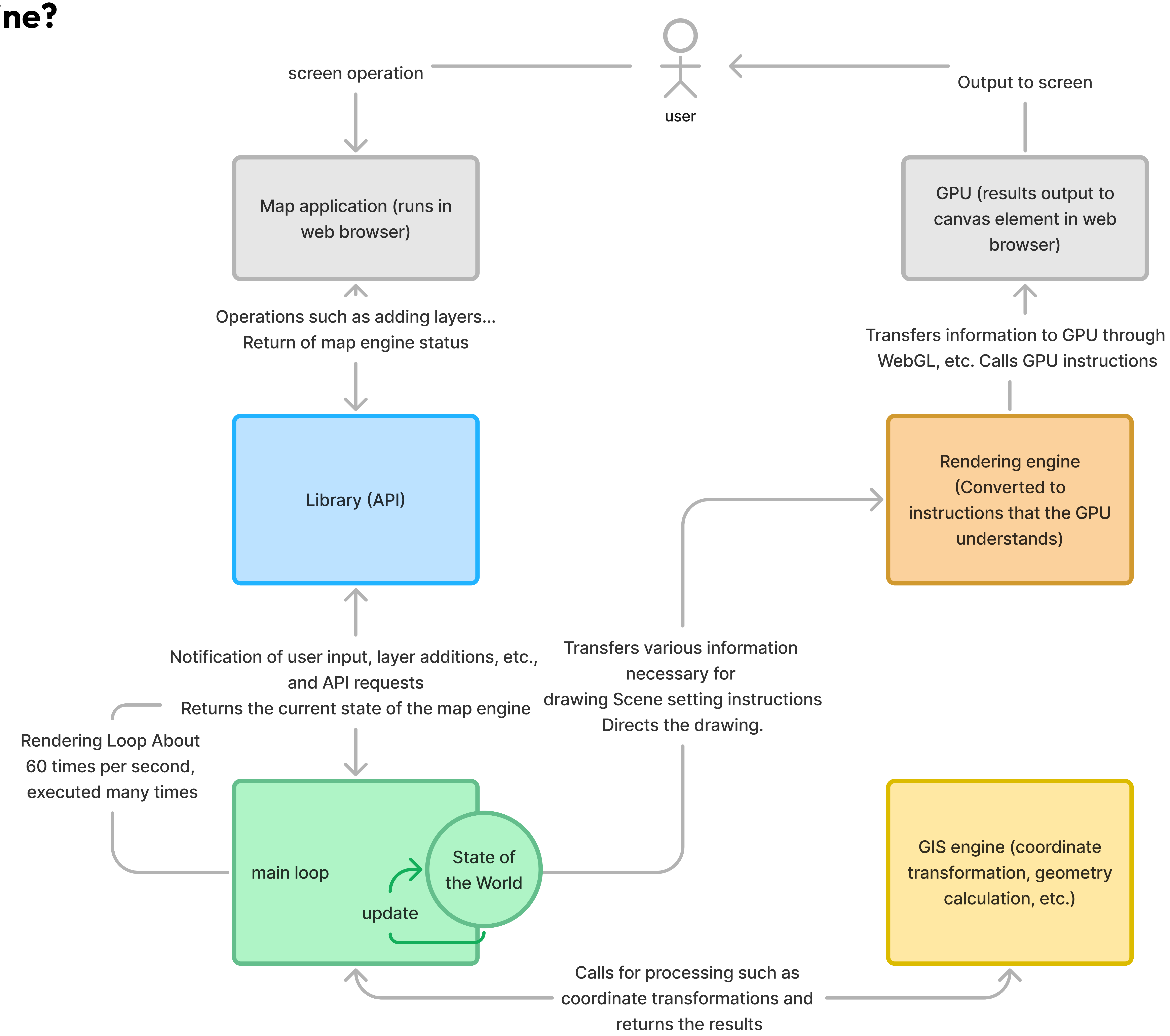
- **Coordinate System Conversion**
  - Converts between Cartesian and Geographic coordinates for accurate mapping.
- **GIS Data Processing**
  - Transforms raw geospatial formats (e.g., GeoJSON, MVT) into displayable models.
- **Large Data Handling**
  - Manages and processes extensive datasets like raster tiles, 3D tiles, and MVT (Mapbox Vector Tiles).
- **Camera & View Calculation**
  - Computes camera positions and adjusts view ranges for precise visualization. Supports dynamic perspectives like zooming, panning, and 3D rotations.

Rendering Engine

A core component of a Map Engine responsible for visualizing geographic data by rendering models in 3D Cartesian coordinate space.

- **Receives Processed Data from GIS Engine**
  - Processed GIS data, including coordinates and attributes, is stored in memory for rendering.
- **Handles Geographic Features**
  - Geographic features represent real-world phenomena (e.g., buildings, roads, terrain) at specific locations.
- **Works with Models**
  - Converts GIS-provided coordinates into models for visualization.
  - Ensures features are accurately represented in 3D space.
- **Final Rendering**
  - Draws models using 3D Cartesian coordinates.
  - Integrates lighting, texture, and styling for a realistic or symbolic display.

What is a Map Engine?



What we are a Making?

A Headless Map Engine

Next-generation map engine separate GIS computation from rendering, developing GIS modules as headless engines.

This allows the use of rendering tools like Three.js, enabling the selection of the best rendering engine for specific application needs.

User adds data through library

↓

Engine retrieves data from memory/server

↓

Engine processes data efficiently for display

↓

Engine passes display data to rendering engine

↓

Rendering engine sends instructions to GPU via WebGL

↓

Data displayed through GPU

Why we are Making it?

1. Difficulty in Improving Visual Quality

- Map engines like Cesium and MapLibre have built-in rendering engines
- Visual representation is tightly coupled with their rendering engines
- Improving visuals requires modifying internal rendering engine code
- High complexity due to engine-specific implementations

2. Large-Scale Data Visualization Challenges

- Frame drops with large datasets and user interactions
- Mobile devices face freezing and battery consumption issues
- Requires optimization of:- Processing algorithms- Data management- Multi-threading- Asynchronous I/O
- Web platform limitations in hardware resource utilization

3. Limited Multi-Platform Support

- Web-based libraries have performance constraints
- Native app development requires different:- Programming languages- Libraries- Frameworks- Graphics APIs
- Rendering implementation needs platform-specific development
- Tight coupling makes rendering engine substitution difficult

1. Difficulty in Improving Visual Quality

- Map engines like Cesium and MapLibre have built-in rendering engines
- Visual representation is tightly coupled with their rendering engines
- Improving visuals requires modifying internal rendering engine code
- High complexity due to engine-specific implementations

2. Large-Scale Data Visualization Challenges

- Frame drops with large datasets and user interactions
- Mobile devices face freezing and battery consumption issues
- Requires optimization of:- Processing algorithms- Data management- Multi-threading- Asynchronous I/O
- Web platform limitations in hardware resource utilization

3. Limited Multi-Platform Support

- Web-based libraries have performance constraints
- Native app development requires different:- Programming languages- Libraries- Frameworks- Graphics APIs
- Rendering implementation needs platform-specific development
- Tight coupling makes rendering engine substitution difficult

1. Difficulty in Improving Visual Quality

- Map engines like Cesium and MapLibre have built-in rendering engines
- Visual representation is tightly coupled with their rendering engines
- Improving visuals requires modifying internal rendering engine code
- High complexity due to engine-specific implementations

2. Large-Scale Data Visualization Challenges

- Frame drops with large datasets and user interactions
- Mobile devices face freezing and battery consumption issues
- Requires optimization of:- Processing algorithms- Data management- Multi-threading- Asynchronous I/O
- Web platform limitations in hardware resource utilization

3. Limited Multi-Platform Support

- Web-based libraries have performance constraints
- Native app development requires different:- Programming languages- Libraries- Frameworks- Graphics APIs
- Rendering implementation needs platform-specific development
- Tight coupling makes rendering engine substitution difficult

1. Difficulty in Improving Visual Quality

- Map engines like Cesium and MapLibre have built-in rendering engines
- Visual representation is tightly coupled with their rendering engines
- Improving visuals requires modifying internal rendering engine code
- High complexity due to engine-specific implementations

2. Large-Scale Data Visualization Challenges

- Frame drops with large datasets and user interactions
- Mobile devices face freezing and battery consumption issues
- Requires optimization of:- Processing algorithms- Data management- Multi-threading- Asynchronous I/O
- Web platform limitations in hardware resource utilization

3. Limited Multi-Platform Support

- Web-based libraries have performance constraints
- Native app development requires different:- Programming languages- Libraries- Frameworks- Graphics APIs
- Rendering implementation needs platform-specific development
- Tight coupling makes rendering engine substitution difficult

Emerging Technologies

Since Cesium's development began in 2011, hardware and software technologies have evolved significantly, enabling solutions to previously challenging problems.

Rust

Static Typing

Predefined variable types help catch errors early in development

Memory Safety

Static management prevents memory-related bugs

Safe concurrency

Built-in mechanisms for safe parallel processing

Benefits for Map Engines

- Higher performance
- Better reliability
- Efficient resource usage

Emerging Technologies

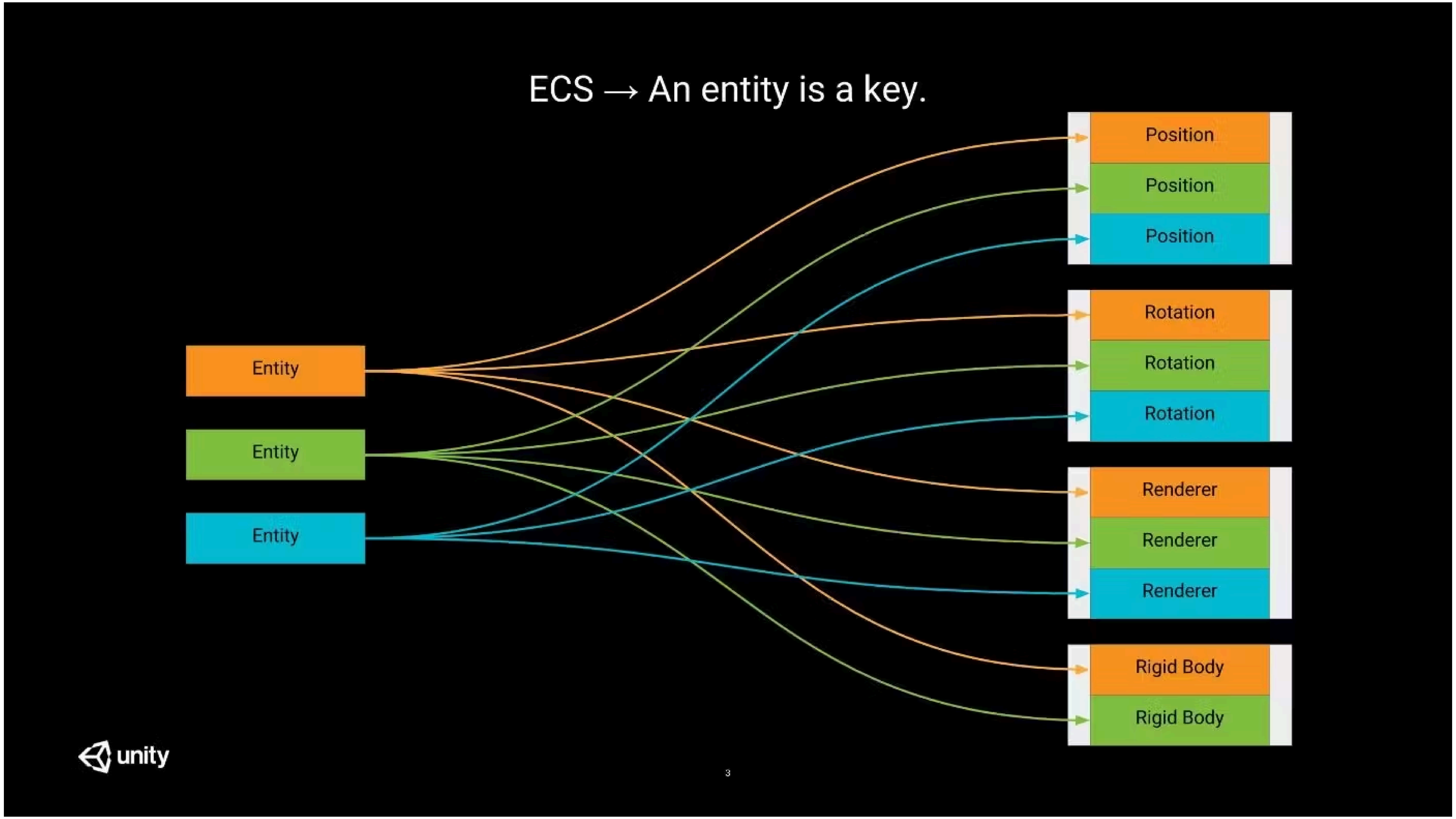
WebAssembly (WASM)

- Enables non-JavaScript languages (like Rust, C++) in browsers
- Provides:
  - Near-native execution speed
  - Type safety
  - Code strictness
- Cross-platform compatibility with WASM runtime
- Bridge between high-performance languages and web platforms

Entity Component System (ECS)

- Game development architecture applied to map engines
- Benefits:
  - Flexible scene element representation - Reusable components (position, color, etc.) - Easy behavior definition
- Performance advantages:
  - Data-oriented design
  - Improved memory layout
  - Better cache hit rates
  - Faster loading times

Why are we making it?



Modern Graphics Technologies

WebGPU

Evolution from WebGL (2012) to modern graphics API for web browsers

Advantages

- Modern graphics capabilities
- Better performance
- More efficient GPU utilization

wgpu Library

Multi-platform graphics API library for Rust

Key Benefits

- Unified API across platforms
- Automatic backend selection
- Single codebase for all platforms
- Abstraction layer for graphics APIs

How we are Making it?

Development with Rust & WASM

By developing the headless map engine using Rust and WASM, it becomes possible to integrate it into various platforms.

Platform-Dependent Rendering

Rendering engines are fundamentally platform-dependent. For example, Three.js depends on WebGL and can only be used on the web.

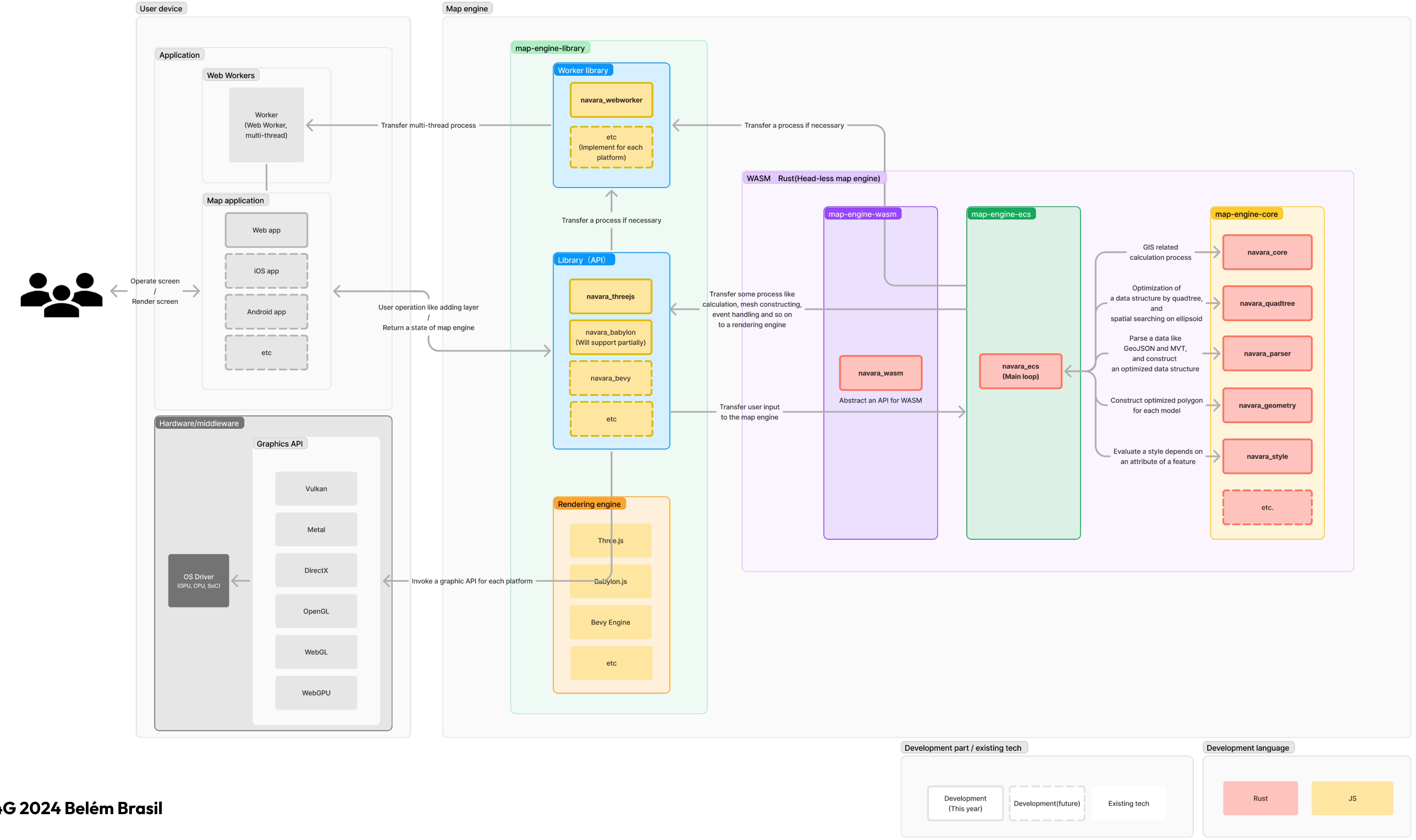
Cross-Platform Strategy

By using appropriate rendering engines for each platform, the system can operate not only on the web but potentially also in native applications in the future.

Current Implementation

Currently, Three.js is being adopted and developed as the rendering engine.

How we are making it?



How does it look like right now?


118 FPS (110-120)
Parameters
layer
layer1
Delete Layer
material
point
show
color
#ffffff
size
0.10
height
1.00
clampTo-
Ground
scaleBy-
Distance
Manila
Pilipinas
Malaysia
Indonesia
Micronesia
Papua Niugini
Borneo
Viet Nam
FOSS4G 2024 Belém Brasil


63 FPS (1-122)
Parameters

What to expect?


google maps
apiKey
AIzaSyBlItcTaggdZkHwcr
stats
exposure
color grading
effects
local date
dayOfYear
205
timeOfDay
14.8
atmosphere
correctAl…
correctGe…
photometr…
aerial perspective
enable
sun
sky
transmitt…
inscatter


exposure
local date
dayOfYear
340
timeOfDay
9.0
c

Next Steps

Open-Source It!

Open-Source

We plan to launch a private alpha release in 2025. In 2026, we will release the beta version, making Navara open-source. The official release is scheduled for 2027.

Features in the Planning

1. Implement previously shown Photorealistic view through Navara.
2. Expanded GIS Format Support: Enhance compatibility with more GIS formats.
3. Performance Optimization: Address current system slowness by exploring better algorithms and solutions.
4. Provide Integration with Re:Earth Visualizer

Questions?

Contact us

x.com/eukaryaofficial

github.com/reearth

Thank You!