A technical journey through 5 years of Google Earth on the Web

John C. Anderson (Google)
Let’s talk about Google Earth!

My name is John. I’m a tech lead manager at Google working on Google Earth clients for a bit over 10 years, including:

- **Earth 9.x** on [Web](http://web.google.com), [Android](http://android.google.com), and [iOS](http://ios.google.com);
- **Earth Pro (7.x)** on Windows, Mac, Linux; and,
- **Earth Studio** on Web.

Earth has a nearly two decade long technical history across multiple platforms.

Today, let’s look at the last 5 years of Google Earth on the Web.
Demo time!

To begin, I’ll set the stage with a quick demo.

I tried to cram in a lot, so here’s the TL;DR of the video so you know the key points:

➢ I search for “Google Earth”;
➢ I click on the link to “earth.google.com” to launch Google Earth;
➢ I spin a 3D globe with imagery, borders, roads, and labels; and,
➢ I use the “Google Earth Timelapse” layer to visualize change over time.

Okay, let’s go!
That video showcased a few key pillars of Google Earth:

- Widely accessible, easy to use product
- High-quality data across the globe, rendered quickly
- The ability for users to create dynamic data visualizations
And abstracting a bit more, a framework for this talk evolves!

1. Core technology
2. Rendering & Optimizations
3. Content
Google Earth for the Web is built from a C++ engine, shared between Web, Android, and iOS. Basically a video game engine for the planet.

While this screen is showing, we’re loading the WASM module and key assets needed to draw the world.

**2017** - Initial Earth Web launch. PNaCL-based and Chrome only. Multi-threaded.

**2019** - Beta of WebAssembly with- and without-threads.

**2020** - Full switch to WebAssembly.
For many years, “earth.google.com” directed to the Earth marketing site which then had a Launch Earth button.

It wasn’t until after switching to WebAssembly that “earth.google.com” could go directly to the Earth application.

2017 - Initial Earth Web launch. PNaCL-based and Chrome only. Multi-threaded.

2019 - Beta of WebAssembly with- and without-threads.

2020 - Full switch to WebAssembly.

2021 - earth.google.com is the app, not the marketing page.
2. Rendering & Optimizations

2013 - Our first attempt to run the Earth 3D renderer in asmjs using Emscripten.

“It crashes pretty quickly citing running out of memory, but it does run...”

< 3 FPS...
2. Rendering & Optimizations

2013 - Our first attempt to run the Earth 3D renderer in asmjs using Emscripten.

2017 - Earth 9 launched with a shared C++ renderer across Web, Android, iOS.
2013 - Our first attempt to run the Earth 3D renderer in asmjs using Emscripten.

2017 - Earth 9 launched with a shared C++ renderer across Web, Android, iOS.

2018 - Enabled brotli compression for large web application files.

7-14% reduced size across various files.
2. Rendering & Optimizations
1. Core tech

2. Rendering & Optimizations

2013 - Our first attempt to run the Earth 3D renderer in asmjs using Emscripten.

2017 - Earth 9 launched with a shared C++ renderer across Web, Android, iOS.

2018 - Enabled brotli compression for large web application files.

2021 - Optimized labeling performance by using direct `<canvas> → texture copy, rather than readback+upload.

Across a large sample of Earth Web sessions:
+2 fps average in multi-threaded build
15% reduction in 60fps jank
1. Core tech

Voyager is a map-based storytelling and education feature built into Earth 9.x (Web, Android, iOS).

It was a premier feature at the launch of New Earth, including stories about national parks, Muppets, and wildlife conservation.

2017 - Initial Earth Web launch with Voyager.

2017+ - Hundreds of individual content launches within Voyager on a diversity of topics.

2018 - Earth Engine tile export directly to Earth.

2021 - “Timelapse in Google Earth” launches in Voyager.

3. Content
1. Core tech

Timelapse is a global video tileset, where each frame of the video is a year of imagery from almost 40 years of Landsat.

2018-2021 - Earth’s 3D renderer was extended to display tiled, geospatial videos in sync as the camera moves and time changes, e.g. for animated clouds and visualizing polar ice.

3. Content

“Regular” Earth app.

Arbitrary user-specified HTML/JS running in a sandboxed iframe

==

Web security fun.
The future...

Looking ahead, there are many exciting developments in web technologies that could impact each of these areas of Google Earth.

1. Core technology
   - Increasing support of multi-threaded WebAssembly.
   - Nearly universal support for single-threaded WebAssembly.

2. Rendering & Optimizations
   - Incremental wasm code loading, e.g. via shared libraries.
   - WebGPU potential for improved performance / visual fidelity, especially if it supports threads.

3. Content
   - Web security evolutions (COOP, COEP, credentialless, oh my!) and their impacts.
   - File system access APIs.
Thanks!