Hi, thanks for taking the time to check out my slides! If you have any questions or comments, please don’t hesitate to email me or post a question to the Cesium forum: https://groups.google.com/forum/?fromgroups#!forum/cesium-dev
My name is Matt Amato and I am one of the founders of the Cesium project and an active maintainer.

I’m currently the technical lead for Cesium Composer, an up and coming service for authoring and hosting 3D globes and maps on the web.

I have a lot of experience with KML and interoperability of Google Earth with other software projects, most recently I wrote the majority of the KML implementation for Cesium, which is the most feature complete KML implementation of any 3D globe outside of Google.

If you’re a long time Google Earth user, my biggest claim to fame comes from being the creator of the Google Earth Satellite Database, which was a real-time KML that visualized all of the satellites orbiting the earth and was one of the most popular KMLs in the world for a little while as well as being the most popular Google Earth Blog post in history. A picture of it is what you see here.
I assume that anyone reading this is at least familiar with Google Earth Enterprise so I won’t go into a lot of detail on it. Released in 2005, Google Earth Enterprise let’s you build globes and maps from raw terrain, imagery, and vector data and serve them to Google Earth and Google Maps clients on your local network or the internet. It was the forefather of a lot of the hosted mapping services that people take for granted today.
For those of you that may not be familiar with Cesium, it’s a modern open source JavaScript framework for building world class 3D globes and maps. Since this is a Google Earth Enterprise talk, I’ll say it’s like Google Earth client, but on steroids with first class support for time-dynamic data and interoperability with open standards and specifications in mind; no vendor lock-in. The same codebase runs everywhere, desktop, browser, mobile, etc...
So what’s one of the creators of Cesium doing giving a Google Earth talk? Well, a lot of former Google Earth users have really picked up Cesium and ran with it, and here are just a few of the hundreds of applications ported from Google Earth to Cesium. This was something that happened very organically in our community, starting with a few users asking for help on our forum and the high demand for help leading to us to create Google Earth specific on-ramping material to help our users migrate. This included over 30 Google Earth API examples that we ported from Google’s own documentation.

Perhaps the most significant migration is the NGA Map of the World program, which has selected Cesium to replace Google Earth as their web-based client.

Keep in mind all of these initial ports were about migrating the application code to Cesium, and not necessarily doing anything with Google Earth Enterprise. We received a lot of requests to connect Cesium directly to GEE, but we ran into one minor problem:
The dreaded vendor lock-in: as is often the class with a lot of closed source software, there were both technical and licensing restrictions in place that prevented us from connecting Cesium directly to GEE. We could try reverse engineering the protocol, but even if we were somehow successful users would be violating Google’s terms of service by actually using it.
That is until...

GEE was unexpectedly deprecated in 2015 and sent a shockwave through the many industries that relied on it for their day to day operations. This is a classic case of vendor lock-in and customers did not have a clear path forward or any clear instructions for migrating the petabytes of data locked away in their GEE databases. Many users were going to be stuck with their data locked away in a proprietary format that they would eventually lose access to.

Thankfully, Google did the right thing here and decided to open source the brunt of the enterprise code base. They handed community management over to Thermopylae Science and Technology, who I believe is actually presenting more info about the project in this room immediately after me.

The code is up on GitHub.
I noticed that there has been some confusion in the OSS community as to exactly what was open sourced, here are the 3 main pieces:

Fusion Pro - Takes imagery, terrain, and 2D vector data and does tiling, compositing and styling. This is the main part of the application that does all of the heavy lifting.

Enterprise Server - Apache or Tornado-based server that hosts the data created by Fusion.

Enterprise Portable Server – Similar to the server, but meant specifically for hosting subsets of a larger dataset on individual machines for offline use in disconnected environments.
Probably more important, and the point of some confusion I’ve seen, is what was not open sourced.

None of the clients, whether it’s the Google Earth browser plug-in, Google Earth enterprise client, or Google Maps API has been open sourced.

The Google Earth Enterprise client and Google Earth Pro have been made available free of charge in binary form, but without source code. These products are on life support and will only be receiving critical updates. Google has already moved on to a new, internet-only version of Google Earth that can’t connect to Enterprise servers.

None of the data that Google hosted as part of Google Earth was made available. The global terrain and imagery is still tightly controlled by Google and you have to find your own data to populate your own maps with.

Finally, you are probably familiar with Google’s buildings and city support available in Google Earth. The enterprise server has no mechanism for generating 3D building data. It’s only terrain, imagery, and vector data.
We laid out two primary integration strategies

- Two primary integration strategies
  - Reverse engineer the GEE protocol and connect to the server just like the closed-source Google Earth client does.
  - Talk directly to the database (PostgreSQL)
Because it was natural for Cesium, being a front-end JavaScript framework, we opted to implement the client connecting to GEE server first. Turns out that there are actually two protocols, one is the 2D protocol, which is used by Google Maps and serves 2D raster tiles only and the 3D protocol, which is what Google Earth uses. Even though the code was open source, there was minimal documentation about how everything was laid out. We actually had to do some packet sniffing to help figure out the non-standard handshaking that needs to be performed with the server as well as the non-standard metadata layout to discover the tiles. And most importantly, we have the server code! It’s not as good as having the client code, but it certainly helps. In fact, the most critical piece of code is this string here, it’s actually much longer than this so this is just the start. This is literally vendor lock-in in source code form, this is an encoding key that was put into the server before Google Earth was even Google Earth and back when it was Keyhole EarthViewer. Without this key, it’s impossible to decode data sent by the server and ingest it into a client.
Cesium already had the ability to stream terrain and imagery, it was just a matter of writing a plug-in to load in the now-unlocked GEE data. After about a month of work, almost all of it done by Tom Fili, we succeeded. Above is a Cesium client connected directly to a GEE server and streaming in high resolution terrain and imagery into the browser. It was a video in the presentations, but obviously that doesn’t come through on the PDF. We were actually surprised by how well it performs because there’s a bit more overhead involved than modern tileset layouts.
Of course there are limitations. While terrain and imagery work perfectly, vector data is a challenge and is not currently implemented in Cesium. The primary problem is that the vector format used by the server is not ideal for modern browser applications because it’s too computationally expensive. Vector tiles are not what we refer to as “ready-to-render”, specifically, the vector tiles themselves just contain definitions for polygons and polylines but the actual geometry needs to be tessellated on the client for each tile before being sent on the video card.

That being said, we haven’t finished reverse engineering the vector format and more research is needed. We haven’t ruled out implementation of vector data in Cesium in the future, but not many users have asked for it yet and most people seem to use KML instead, which Cesium has supported for a number of years.
The second method I mentioned is direct database access, and it’s not something we have actually tried yet but I did want to discuss it.

The idea would be to export the data from the database into modern formats, such as 3D Tiles and ignore the server completely, of course this is more a migration strategy than client support, but the end result is largely the same.

Another idea would be to extend the server to transcode to other formats on the fly, the main benefit would be existing globes and maps would be able to get GEE support “for free” if they already implement web standards. However, I think this kind of transcoding would be difficult to pull off in the legacy confines of GEE.
Which brings me to my next thought, future innovation. While the code base is now open source, it’s still 15 years old and in need of modernization.

The problem is that since the primary clients are closed source, we can’t update the server to modern protocols because the client only knows how to connect to the legacy one.

We can’t update the tiling capabilities to support modern data, such as photogrammetry models, because there would be no way to update the client so that it could actually view it.

I’m not sure what this means for future innovation in Fusion Pro or if there are any plans to do any significant feature work.
So in summary,

Google did a great thing and they should be commended for it, while obviously open source software is an ideal we strive for from the first commit on, open sourcing proprietary software at the end of it’s life is still a much appreciated gift to the community. Open source projects can now add first class support for connecting to GEE servers.

Of course GEE is ultimately still a legacy product and it’s unclear to me what it’s future looks like. But it is now possible to provide a smooth transition to modern geospatial clients, like Cesium and avoids data loss from vendor lock-in.
Questions?

Come visit our booth!
We’re hiring!

https://cesium.com

matt.amato@gmail.com
@matt_amato