Thanks for taking the time to check out this talk! It details our work on advancing not just Cesium, but open standards for massive 3D geospatial visualization on the web.
Cesium is an open-source JavaScript library for creating 3D globes and 2D maps. Since it’s built using web standards, it runs almost anywhere. It’s under the liberal Apache 2.0 license, so it’s usable in both free and commercial applications.

Cesium was born in aerospace, so it was built with precision and accuracy in mind, whether you’re on the ground or in outer space. It also has first-class treatment of time and tuned for dynamic data.

It’s popularity has grown so that it is now used in a diverse array of fields, such as real-estate, city planning, sports, environmental science, and more.
During the talk I showed a few simple Cesium demos, but I encourage you to check out Sandcastle, our live code editor that has dozens of examples:

http://cesiumjs.org/Cesium/Apps/Sandcastle/index.html

We also have a showcase of user-created applications and demos:

http://cesiumjs.org/demos.html
So what does creating something like Cesium take? According to cloc (https://github.com/AlDanial/cloc) we have about 97,000 lines of engine code, that’s of actual JavaScript, no comments or blank lines. (There’s actually about 48,000 lines of comments).

In order to write 100k lines of JavaScript you need to actually write another 100k lines of test code to make sure it all works (and keep it working). We have almost 7,000 unit tests, using Jasmine, with 93% code coverage (most of the uncovered code is actually ancient Cesium code that’s been on the "refactor and add tests" list since the beginning). But tests don’t matter unless people actually run them so we try to keep them fast and can run the whole test suite in under a minute and we do so during development and before every pull request is merged.

Maintaining a JavaScript code base of this size takes a good amount of teamwork, good tools, peer review, and most importantly, developer diligence.
When you are rendering the entire world at scale, people want to fill that world with data, lots of data, terabytes of data. After all, content is king and no matter how good your mapping engine is, it's useless if people can't easily get data onto the map.

From the very beginning Cesium’s goal has been to implement the complete Digital Earth vision originally laid out by Al Gore in 1998. (https://en.wikipedia.org/wiki/Digital_Earth). We want to be more than just traditional imagery and vector data, we want to show terrain, 3D models, point clouds, temporal data, stuff like buildings, trees, mail boxes, cell towers, power lines, fire hydrants, pipelines, taxi cabs, satellites.

We wrote Cesium to be extensible and added numerous plug-in points so that we could implement various geospatial standards to help us achieve this goal, then we ran into a problem...
When we started Cesium 4 years ago, this is how I remember the state of geospatial standards on the web.

Imagery has been around a long time so there were already a ton of standards for it. Even better is that these standards were web-friendly from the start. Imagery has never been a problem for Cesium.

GeoJSON or shapefiles converted to geoJSON were everywhere (and not too much has changed there). KML was popular but mostly only used via Google Earth.

Then you look at Terrain, and 3D models, buildings, point clouds, and all of these things that we wanted to do in Cesium and there wasn’t much there. That’s not to say the data wasn’t there, there’s a ton of data, but there weren’t any good standards or specifications for getting it into a web map.
So our first attempt at publishing an open specification was with terrain. Traditionally people used heightmaps to represent terrain data but that’s a container format, and not a common standard you can implement; everyone does them differently. We introduced a new format called quantized-mesh, which is an open specification and provides many benefits over other approaches, such as non-uniform resolution in tiles allowing higher level of detail where it’s needed and saving space where it’s not. The most important aspect of quantized mesh is that it requires minimal processing before being sent to the GPU for rendering, which is a huge win and avoids the performance problems of JavaScript.

I’m happy to say that we’ve had two other engines adopt quantized-mesh in addition to Cesium. Systems Tool Kit a desktop aerospace engineering platform from Analytical Graphics, and WhirlyGlobe-Maply, which is an open source iOS and Android mobile mapping engine have both added support.

If you’re interested in adding terrain to your own map engine, we would love for you to give quantized-mesh a try. The spec is on our website and feel free to drop us a line if you have any questions: https://cesiumjs.org/data-and-assets/terrain/formats/quantized-mesh-1.0.html

Of course having good terrain data adds just as many problems as it solves since
now you have to worry about placing your geospatial data on terrain, but that’s another problem for another talk.

So we mostly solved our terrain problem, but there’s still 3 more big question marks, which leads into the heart of this talk, starting with 3D models.
So what exactly do we mean by 3D models? We mean a 3D representation of a scene or an object in that scene. So here we have Santa and his sleigh from the NORAD Tracking Santa project that we are lucky enough to be a part of each Christmas.

It’s made up of a node hierarchy with associated geometry. A set of materials and textures that get draped over the geometry, as well as animations and skins to show realistic movement. You put that all together and you get Santa riding his sleigh as shown in the lower right.
Obviously 3D models aren’t new and they’ve been around a long time. There are a ton of standards and file formats for storing them, visualizing them, sharing them with others. The problem is that these are interchange formats. They are meant for exchanging data for further editing. They aren’t optimized for real-time visualization, and they definitely were never meant to be used on the web.

So when WebGL took off, lots of smart people realized that we needed a web-friendly runtime format.
This resulted in the creation of glTF. The logo alone is a huge step up from quantized-mesh. 😊
It’s is not technically a Cesium specification. However, Patrick Cozzi (the godfather of Cesium) along with several other Khronos members spearheaded the effort and Cesium has played a large part in its development. For those of you that may not be familiar with Khronos, they are the same standards body that gives us WebGL, OpenGL, and many other graphics standards. We are extremely excited to be part of establishing new industry standard like glTF through such a major governing body.

It’s easy to think of glTF, or GL Transmission Format, as JPEG for 3D:

1. It’s the end of the pipeline, nothing changes about model creation, but when you are ready to publish to the web, you convert it to glTF
2. Smaller file sizes for faster downloads, mesh and animation data can be transmitted as binary
3. Minimal client-side processing is needed so loading can be fast
4. It’s runtime neutral so it can be implemented in all tools, apps, and runtimes
5. It has all of the features people already expect from models, such as materials, animations, skins, cameras, and lights
6. It has a flexible extension system so additional features can be added as needed for specific use cases
Converters currently exist for Collada and FBX, but we expect other formats to get converters as well. We also hope editors and other modeling tools to add direct support for glTF in the future.
While we’ve been using various versions of glTF in Cesium for a couple of years, the 1.0 spec is still a work in progress, but its release is imminent. It’s a full open specification with an open process which you can follow on GitHub. It also has an extension mechanism to allow for additional capability not in 1.0, which Cesium is already taking advantage of. For anyone that needs 3D models in their web application, geospatial or otherwise, glTF should be your first choice.
But don’t just take our word for it. In addition to Cesium, several other high-profile projects have adopted or are in the process of adopting glTF, including three.js, the most popular WebGL library in the world, as well as Microsoft’s Babylon.js and Libre Office.
So we now have an awesome open standard for solving out model needs, but that leaves a couple more question marks. However, since we have 3D models, Cesium can certainly load buildings, and since Cesium can draw points, what’s so hard about point clouds? The answer is scalability.
Desktop products can often get away with brute forcing big data. In traditional use cases, data is stored locally or on a fast local network. Apps can use gigs of RAM or even 10s of gigs on 64 bit systems. For example, some of the large scale Google Earth KML files we’ve encountered would max the CPU and use large amounts of memory, taking several minutes to load everything before being usable. This isn’t how the web is expected to work.

Someone with a hundreds of thousands of buildings, trees, vector data, or other models can’t just make hundreds of thousands of web requests. The data is too big, JavaScript is to slow, and browsers limit per-tab memory. Form factor also plays a big role because the web expected to work on a variety of devices, including low-power mobile ones.

We want to solve this problem.
Enter Cesium 3D Tiles, built for streaming.
3D Tiles are the missing link between massive heterogeneous 3D geospatial datasets and 3D mapping engines. Bringing techniques from graphics research, the movie industry, and the game industry to geospatial, 3D Tiles define a spatial data structure and a set of tile formats designed for 3D and optimized for streaming and rendering.
3D tiles is an open specification and you can follow along its development via our GitHub repository:

https://github.com/AnalyticalGraphicsInc/3d-tiles

Rather than starting from scratch, we’re building on existing standards, like glTF for models, in order to make them scalable to meet the needs of a digital earth.

And it’s not just about pretty pictures, 3D tiles are interactive and styleable, to keep with the goal of creating useful geospatial applications.

We hope to revolutionize massive geospatial datasets on the web.
The open data movement continues to gain traction and we wanted to leverage that to showcase what 3D Tiles can do. So we took the New York City OpenStreetMap extract from Mapzen and turned it into 3D Tiles. We started with a traditional approach of creating a Collada model for each building in the OSM extract, which ended up being 1,140,378 models, taking up 10.3 GB on disc. Impossible for the web. Once we processed that into 3D tiles, we were down to 4,199 files and 345 MB on disk. That’s smaller than most imagery sets and 271 times fewer files and 30 times smaller on disc than the original models with no loss of information.

And it’s keeping all the information that’s key. Normally when you batch files like this you end up with a pretty picture but greatly reduced interactivity, however with 3D tiles every one of these 1.1 million buildings is still selectable and has its own set of meta-data embedded with it, in this case we embedded the length, width, and height of the building for easy zooming as well as the name so you can see it instantly on mouseover as well as the OSM identifier so when you click on one of them we can pull up actual meta-data from OSM.

Here’s the World Trade Center and World Financial center, and if we look in Times Square we even see the small TKTS booth that operates in the middle of it. Of
course exploring this data would be hard if we didn't have good mouse control. With the 3D tiles effort we've added more context sensitive controls for cesium. For example, we can grab onto a building when looking up from ground level and easily walk up the side to get to a better height. We can right-click on a building to make it the center of rotation and even grab the sky and look around. This is a true 3D map with completely freedom of movement, not an isometric view that we see with many other engines and demos.

Certainly building data is the furthest along of our current 3D tiles effort, but as I mentioned, it’s heterogeneous. So let's look at what else we’re working on.

Source: 1,140,378 models, 10.3 GB on disk
3D Tiles 4,199 files, 345 MB on disk
Point cloud support in 3D Tiles is still in the prototype phase, and while we have an early demo, it's still a long way from the high-performance, low-bandwidth promise of 3D tiles. The main takeaway here is that 3D Tiles are heterogeneous. The same code on the client that is used for 3D buildings is also being used for point clouds. The difference is the payload of individual tiles. While 3D buildings have a non-uniform, overlapping, quadtree-like tiling scheme, the point clouds have a similar octree-like one, but both can be represented as a tree in the 3D Tiles format and Cesium can handle either at runtime without caring. This is very different than the rigidity of traditional 2D tiling schemes, like TMS.

At the same time, we’re also working on Model instancing, which allows us to save a ton of memory when you need to load many versions of the same model. This will enable things like visualizing 100,000s of trees which only differ in location and scale. Don’t be surprised if you see a tweet from us in a couple of weeks showcasing OSM tree data as part of our New York demo (or it may already be there by the time you read this!).
Like Cesium enough to want to work on it and with it? We’re hiring.

• Software Developers
• 3D Software Developers
• Co-ops and Internships
• Contact Me: mamato@agi.com

cesiumjs.org/jobs
Presenting at (and attending) JS.Geo was a blast and I want to thank everyone that came out. I also want to thank you for reading through these slides even if you weren't there!

You can always get the latest news on Cesium on our website and twitter.

Feel free to drop me a line or follow me on Twitter as well.

Thanks!

**Website:** http://cesiumjs.org

**3D Tiles repository:** https://github.com/AnalyticalGraphicsInc/3d-tiles

**3D Tiles blog announcement and overview:**
http://cesiumjs.org/2015/08/10/Introducing-3D-Tiles/