Thanks for checking out my slides, there will also be a video available at http://cesiumjs.org/publications.html as soon as FOSS4GNA publishes it.

I’m going to start with a quick review of the basics so everyone is on the same page and then I’m going to dive into some of the specifics of what makes Cesium unique, especially when it comes to time-dynamic data. And then we’re going to close things out with a quick community update and some Q&A.

Since Cesium is such a visual tool I’m going to be running through lots of demos and examples instead of relying on lots of slides. I’ve placed public links for most demos in each set of slide notes.
So, what is Cesium? To put it succinctly, Cesium is an open-source JavaScript library for creating 3D globes and 2D maps. Cesium is used in many fields, such as geospatial, aerospace, sports, environmental science, and more.

It's under the liberal Apache 2.0 license, so it's usable in both free and commercial applications.

It's built using HTML5 and WebGL. For those that may not be very familiar with WebGL, it's the standard for creating hardware-accelerated graphics in a browser, without a plug-in. So Cesium works almost anywhere; desktop and mobile, windows, Linux, mac, android, iOS, and even windows phone.

Finally, one of the main things that sets Cesium apart, is our first class treatment of time. While we can also handle traditional static data sets, Cesium makes is really easy to not only load time-dynamic data, but also customize graphics visualization around it.
Without further ado, I want to jump right into our first demo and show off basic Cesium features.

http://cesiumjs.org/Cesium/Build/Apps/CesiumViewer/index.html

So here’s a blank globe (currently showing Bing imagery). Right off the bat you may notice this animation controls and time slider down the bottom of the page. That allows us to jump around to any point in time and we accurately calculate the sun and moon position to match. This is a default GUI that comes with Cesium; it can be customized or shut off completely.

Enter “Grand Canyon” in the Geocoder (hit the magnifying glass in the upper right). This terrain data is hosted by AGI; using processed open data and is free for use with Cesium. We have up to 10 meter resolution in the US and 90 meters for the rest of the world. We are in the process of updating the new SRTM set to show at least 30 meters everywhere. We’ve also processed data as high as 3 cm resolution.

For non-photorealistic imagery, like OpenStreetMap, we can enable lighting to
show realistic shading of the terrain.

Cesium is not just a globe; we have one API but 3 different views. We can switch to more traditional 2D views. Notice that we have smooth zooming in and out and can easily handle things like rotation. We also have dynamic terrain lighting in 2D, so you can automatically get terrain shading with any base layer.

We also have a mode we call Columbus View. It uses a 2D map projection but still takes height into account; which can be useful for certain data sets. Right now it’s empty but we’ll show some more later.

And finally, we also have water effects so if I zoom to the bay area here you’ll see some nice water and we can watch the sunset over the horizon.
So anyone that's played with a virtual globe like this before knows it's fun to just explore the world, but ultimately we want to populate the map with cool stuff. Perhaps the most obvious use case for Cesium is a traditional Geoportal, and many of our users are building them.

One such example is Australia’s National Map, created by NICTA, an Australian think-tank for information and communication in technology. The National Map is an initiative of the Australian Government's Department of Communications to bring government data sets to the public directly from the agencies responsible for maintaining them.

For example I can overlay broadband availability, which is a WMS layer in this case. We also support WMS feature picking so I can click on any of these features and get back its meta-data.

You can also use Cesium to build non-traditional mapping applications. Anyone who has stopped by our booth this week has probably seen Doarama, which is a sporting app that allows users to upload GPS data and have it automatically create some awesome visualization in Cesium. In this example we have a group of hang gliders flying around the Alps. This is using the same terrain source I mentioned earlier. You can jump around in time and speed up and slow down time. You can also enable different filter effects which are made possible by a prototype post-processing capability in Cesium. For those of you interested in head mounted displays, Cesium has a WebVR plug-in as well which allows you to hook up an Oculus or other device and get a virtual experience.

http://www.doarama.com/view/2171
And just one more demo to show off the versatility of Cesium, this time in the aerospace industry. Here we have about 16,000 space objects being tracked by the US government. This is real data being stream down in real time. We can click around and get meta-data for every one of these objects. Orange dots actually indicate space junk and green are operational functioning satellites. We can turn filter it to just operational or we can filter farther for certain aircraft. For example we can show just the 25 FLOCK cube sats in orbit from Planet Labs.

http://apps.agi.com/SatelliteViewer
So those are just some quick high-level demos that barely scratch the surface of what is possible with Cesium. So let’s take a step back and look at the building blocks and features that make up Cesium. One of our core goals with Cesium is to work with as many open formats as possible. As you can see from this list, we support lots of different standards, particularly for imagery.

While we strive to use existing standards, in the case where a good open format doesn’t exist, we help the community create one. A perfect example of this is quantized-mesh, which is the primary terrain format for Cesium and what was being used in all of the demos we’ve shown so far. Heightmap terrain tiles, which Cesium also supports, aren’t really a format, they are a container with no real standard. So we created quantized-mesh to be a high-fidelity and web-friendly format for streaming terrain. We publicly document and maintain this spec on our site: http://cesiumjs.org/data-and-assets/terrain/formats/quantized-mesh-1.0.html

We are also big contributors to glTF, which is an emerging spec for 3D models on the web. It’s governed by Khronos, the same group that controls WebGL and Collada.

And as you can see from the long list of items on the screen, we also support

<table>
<thead>
<tr>
<th>3D models</th>
<th>glTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imagery</td>
<td>WMS, WMTS, TMS, OSM, Bing, …</td>
</tr>
<tr>
<td>Terrain</td>
<td>quantized-mesh, heightmaps</td>
</tr>
<tr>
<td>Vector</td>
<td>GeoJSON, TopoJSON, KML, CZML</td>
</tr>
<tr>
<td>3D Buildings</td>
<td>TBA</td>
</tr>
</tbody>
</table>
lots of the standard data formats you probably deal with on a daily basis, and with many more to come.
So let's look at handling some of these vector formats in Cesium. I'm going to do this with our interactive prototyping tool, Sandcastle, which helps our users learn and experiment with Cesium.

You can load an external file into Cesium with one line of code. For example, let's take a quick look at a simple TopoJSON file of the United States. As most of you are probably aware, GeoJSON and TopoJSON don't have a built in styling standard, so most content comes in unstyled, like we see here. We do support Mapbox's simple-style standard for specifying colors and icons, but we also make it trivial to do this in code at load time. So I can easily turn this from yellow transparent to hot pink.

Furthermore, if I click on any of these features, you'll notice that our default behavior is to bring up what we call the InfoBox to show the meta-data for that item. Here's where the power of Cesium really starts to show up. Any data sources we support, GeoJSON, TopoJSON, KML, and our own format CZML,
all get loaded into a standard set of objects we call Entities. This means that no matter where the object come from, we can now customize the visualization through a standard API, which we uncreatively call the Entity API.

So here we have population and date entered into the union as our two pieces of meta data. Since Cesium is 3D, we can use that population information to improve our visualization. In this next sample I’m loading the same exact file as before, but I have a small loop that runs after load which assigns a random color to each state and also sets the extruded height of each polygon to be a ratio of the population. So with very little effort we now have not only cooler looking visualization, but it helps convey more information to the user by making effective use of 3D. And because Cesium supports a single API, you can still view it in both 2D and Columbus Viewer (2.5D).

But this talk is also about Cesium’s time-dynamic capabilities, so I don’t want to stop here, we can use the date contained in the union to specify an interval for display of each state. Using discrete intervals is just one way you can specify temporal data in Cesium. However, as you watch this demo, you’ll notice that each state starts out as flat and then grows over time. This is because I also told each polygon to have an extruded height of 0 at the time it entered the union and then to linearly increase the value of the extruded height overtime until it reaches the population when the data was collected in 2008.

So now we are using both the 3D capabilities and the temporal capabilities of Cesium to convey even more information to the user and create a more interactive and fun visualization. Specifying time tagged data and having Cesium interpolate values at each frame like this is called a SampledProperty, and another way to specify time-dynamic data in Cesium. It’s important to note that all I did was declare what my data looked like up front and Cesium does the rest. I don’t have to worry about updating the data each frame or the best way to store and efficiently render larger data sets. Cesium takes care of it all for me and I can simply jump around in time at will. This form of data driven visualization is extremely powerful and easy to use. (Note the last example with growing states may not be in an official Cesium build on our website yet, but will be in a future release).

Interpolation:

I used the terms interpolation and samples, and I want to move immediately to the next demo to explain that a little more thoroughly what that means. Let’s
say you’re piloting a Tie Fighter around the Grand Canyon and you decide to record a GPS track of it. It might look something like this. As the fighter is flying around, you'll see the motion looks stiff an unnatural. This is because we only actually have data where each of these dots are, and by default we are performing linear interpolation to connect them. The data doesn’t have enough samples to properly convey what the real motion looked like.

Cesium addresses this problem by allowing you to specify different forms of curve fitting, specifically Lagrange and Hermite polynomial approximation. If I turn on Lagrange for this track you’ll see that while the actual sample points haven’t changed, the way they are connected has and the path is now restored to better represent what was actually happening at the time the data was collected. Lagrange is usually used for well-behaved objects that don’t have rapid change in velocities, such as a satellite orbit that we saw earlier. For aircraft (and spaceships) that maneuver and bank, Hermite is better suited for this task. Additionally, Cesium lets you specify velocity samples and other derivatives to go along with your position data, and this data is taking into account by the Hermite algorithm.

The ability to interpolate data this way has two major advantages. First, as you can visually see from this example, it’s going to better approximate realistic motion. But equally important for a web application, it means that we don’t have to send a ton of data to the client in order to represent a track. We can send just the right amount of samples needed to fit the curve. This means that you’re going to save bandwidth or be able to handle a lot of objects at once, such as the Satellite example we looked at earlier, which was using Lagrange interpolation for satellite samples that were 5 minutes apart.

KML:
http://cesiumjs.org/Cesium/Apps/Sandcastle/index.html?src=KML.html&label=DataSources

Before I jump back to the slides, I might as well show you some of the new KML functionality we added as well. We can load both KML and KMZ files, show balloon descriptions and even support some of Google’s extensions, such as gx Track for time-dynamic data.
In addition to GeoJSON and KML, we also support a new format which we wrote specifically for Cesium called CZML, short for Cesium Language. It’s a declarative JSON schema which makes it possible to define objects and their graphics in Cesium. Like Cesium, it has first class treatment of time-dynamic data and is usually kept in step with Cesium capabilities. It’s also easily packetized and streamable, making it ideal for real-time data, such as the kind we saw in the Satellite Viewer demo.

CZML GPS:

Sandcastle CZML demo:
http://cesiumjs.org/Cesium/Apps/Sandcastle/index.html?src=CZML.html&label=Showcases
As I’m sure many of you are aware, Google has deprecated the Google Earth API and plugin, and it will be going away later this year. We’ve received a huge influx of users who are looking to migrate from Google to Cesium. In order to make that easy, we’ve ported 30 examples from the Earth API into, which are available on our website. The fact that Cesium is tuned for time-dynamic data also makes it good for interactive data. You can see this in the ported Monster Milk Truck example which lets you drive all over the globe with terrain.

http://analyticalgraphicsinc.github.io/cesium-google-earth-examples/ (Scroll to bottom for Milk Truck).
We’re actively working on a building streaming capabilities into Cesium, much in the same way we support terrain. This particular data set is from Cyber City 3D. This is still in the early stages but we plan on making a lot of progress throughout this year. Unfortunately we can’t share a public link yet, so the screen shot will have to do.
We are also working on textured buildings, such as this smaller data set in Santa Monica. Unfortunately we can’t share a public link at this time.
Cesium has a ton of capabilities, and there’s actually a lot more that we didn’t have time to show today. In case you’re curious as to how much code goes into creating something like this, as of our last release on March 6\textsuperscript{th}, Cesium was around 90,000 lines of JavaScript code, with another 87,000 lines of unit tests to make sure it works. I don’t think it’s an exaggeration to call Cesium one of the most ambitious JavaScript applications in existence. That being said, our goal is to keep Cesium as lean and mean as possible, and we also make effective use of modules to allow users to only include the pieces they need.
Of course Cesium would be nothing without the community, and its growing fast. We went from 280 members in October to over 550 as of last night (and hopefully we’ll have a few more after this presentation). I want to encourage anyone that is using Cesium to take part in the conversations that go on in the forum and don’t hesitate to provide us with feedback, good or bad.
I didn’t want to miss the chance to mention that AGI is hiring Cesium developers, dev relations/community manage, and business leaders. If you’re interested, or you can contact me directly via e-mail.

Web Application Developer -  

Senior Graphics Architect -  

Developer Relations -  

Business Lead -  
https://search8.smartsearchonline.com/agi/jobs/jobdetails.asp?job_number=190
We were also asked to remind everyone to please help evaluate the various sessions on conference website, so please go to the website and not only rate us, but rate any of the other talks that you've been too and help improve the next FOSS4G conference.
Any more questions just go to cesiumjs.org and you can get to everything else from there.