Cesium is a JavaScript library for creating 3D globes and 2D maps in a web browser without a plugin. It uses WebGL for hardware-accelerated graphics, and is cross-platform, cross-browser, and tuned for dynamic-data visualization. For those who may not be familiar with it; WebGL is a Khronos standard for exposing graphics hardware acceleration through the HTML5 canvas. While WebGL itself is not a W3C specification, it is officially exposed via the HTML5 specification: https://www.khronos.org/webgl/.

Because it’s web-based, it runs almost anywhere; and since we use the Apache 2.0 license, it can be used for both commercial and non-commercial use.

One thing that sets Cesium apart from almost all other map and globe applications out there is it’s treatment of time-dynamic data as a first class citizen; as you’ll see from some of the demos.
Here’s a blank globe (showing Bing imagery default). You can select alternate imagery providers in the upper right. 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 (middle mouse scroll wheel and button). We also have a mode we call Columbus View, 2.5D is another name. 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.

So if we head back to 3D mode and turn on some realistic imagery you can zoom into the grand canyon using the Geocoder (magnifying glass) and show off some of the terrain. Hold down middle mouse or CTRL to rotate. This is actually a global terrain data set hosted by AGI; using processed open data. Africa, South America, Asia, and parts of Canada are 90m. US has 10 meter resolution. We are in the process of updating the world to 30 and have processed data as high as 2.95cm resolution. Data is processed with STK Terrain Server:
http://www.agi.com/products/stk/terrain-server/

While Cesium supports Terrain Providers that serve raw traditional
heightmaps, they are just a standard container and don’t actually specify how the data is stored, all current formats are unable to meet today’s requirements for efficient terrain transmission and rendering. In fact, we found that there is currently no standard terrain runtime format for performant rendering. We are not the first to discover this limitation; most engines tightly couple their terrain format to their runtime engine and some do not publish the format, e.g., Google Earth. Our rule in Cesium is that we want to use open standards if they exist, but create them when they do not. Quantized-mesh, the format used in Cesium, was a good use case for the latter.

You’ll also notice the timeline down in the bottom as well as animation controls. Since we’re on a blank map not much changes, but we do see the sun/moon positions and if I zoom out you can see the stars whizzing around. You can enable lighting if you want; but it’s disabled by default because most people don’t want to look at a dark map.
So we’ve got an awesome empty map; but we want to populate it with cool stuff. What kind of stuff? We already saw terrain and imagery. Point clouds are a hot topic of late and while we don’t have support yet; we are working on it. Cities, buildings, and overlaid information is probably a big interest to a lot of you here. Vector data which runs the gamut from points, markers, lines, polygons, etc… and finally all of the above but with time in the mix.

Terrain: heightfield, triangle mesh (2.5D or full 3D). Often derived from LiDAR point cloud.
Vector data: points (label/billboard), polygons, and polylines.
Vector data use to be rendered as rasters. Now it is often rendered as vectors, e.g., Google Maps, OpenLayers 3, and MapBox GL.
Time-dynamic isn’t just vector data. Users want to study the change of terrain (terrain, imagery, point clouds) over time, e.g., snow coverage in the Sierra mountains.
http://cesiumjs.org/demos/nationalmap.html
D3-Cesium demo started a year ago as a hackathon: http://cesiumjs.org/d3cesium/

Here’s a good example of the power of the web. This app uses Cesium for 3D visualization, but then overlays a d3 visualization of the same dataset and other HTML elements to create a rich and interactive experience.

Explanation of d3 visualization. Original concept courtesy Hans Rosling, then brought into d3 as an example by Mike Bostocks.

200 countries over 200 years, each bubble represents a country through time:
Bottom axis is average income; further to the right the wealthier the nation
Left axis is average life expectancy; further up the chart, the healthier the nation.
Size of the bubble is relative to the population of the country.
Witness catastrophic events and industrial revolutions by the movement/growth of these bubbles over time.

Integration with Cesium takes this visualization one step further, by adding context to each bubble. We can more easily identify where in the world these events are occurring. We can interact, pan around the world, and become more engaged with the dataset, which ultimately promotes a better understanding.
Sandcastle is a live demo/prototyping app that let’s you write and run Cesium code without leaving the browser. Kind of like JSFiddle for those of you that may be familiar with that. There are dozens of demos and examples in this app but I’m just going to highlight a few things that weren’t covered by previous demos.

GeoJSON SimpleStyle:

GeoJSON:

One of the strengths of Cesium is that we’ve abstracted away a lot of the details into a high-level API; and more importantly, an API for loading already existing standard data sources. For example, let’s look at a couple of GeoJSON demos. First, we’re loading a dirt simple file to illustrate our support for Mapbox’s simplestyle specification. For those that you that don’t know, GeoJSON data has no style information in it’s standard; it’s just vector data. Simplestyle extends GeoJSON with basic graphic settings and icons. This demo is only 5 lines of code and in reality it could be 0 because you can load an external data file through a URL query parameter without any code at all.
Now that’s great and all, but hardly anything other mapping library can’t do. So let’s head over to a second GeoJSON example that shows some Cesium specific capabilities. Here we load in polygons for all 50 states. This is actually a TopoJSON file, but in practice Cesium treats them the same as GeoJSON. This one is just vanilla and unstyled; but it does have some meta-data associated with it. In particular it has population from the last census. With a few additional lines of code we can use that information to turn this data into a more impressive visualization. In this demo we also give each state a random color, just to jazz things up a little.

So now we’ve shown one simple way we can manipulate existing GeoJSON or TopoJSON data to enhance it for 3D. But there’s a problem. We’re back to having to write a custom app and custom code to do what we want.

Wouldn’t it be great if there was a data format that could support the features of Cesium so that I could take advantage of it’s capabilities without having to write a custom app each time. That’s where CZML comes in.
• Short for Cesium Language
• A streamable JSON scene description for data-driven visualization.
• Developed alongside Cesium but can be implemented in other projects.
• CZML Guide: [http://git.io/czml](http://git.io/czml)
http://www.geo-animate.com/category/commonwealth-war-graves/

Being the 100th anniversary of the start of WWI, geo-animate.com wanted to create animations that put the events on WWI's Western Front in context. The Commonwealth War Graves Commission maintains records of more than 1,000,000 Commonwealth soldiers (Australia, Britain, Canada, India, etc) that died during World War I. At the time most soldiers were buried in very proximity to where they fell. Because of this, we can combine grave locations with the date each soldier died to create an animation that illustrates the location and intensity of fighting through the four grueling years of World War I.
Senior Graphics Architect -
Developer Relations -
Questions?

Web: http://www.cesiumjs.org
Email: matt.amato@gmail.com
Twitter: @matt_amato
Backup slides
http://comspoc.com/spacebook/

Developed by AGI

Server-sent events with CZML. Web sockets were available when we needed it and we are only one way.
Community

- Code on GitHub since April 2012
- Contributors
  - 20 contributors from AGI
  - 20 contributors from NICTA, EU Edge, Raytheon ISS, Evax Software, Aviture, Google Summer of Code, ESA SOCIS, and individuals
- 422 forum members
Last July 2013, it was
• 54K lines of engine code
• 55K lines of test code
• 4,138 tests
• 93% coverage
These datasets are used to form the terrain dataset we use in Cesium: https://cesiumjs.org/data-and-assets/terrain/stk-world-terrain.html

There are many other datasets to add like

Norway: http://data.kartverket.no/download/content/digital-terrengmodell-10-m-utm-33
Data Standards

- CZML
- GeoJSON
- TopoJSON
- simplestyle
- KML (in development)
- quantized-mesh
- glTF
- Lots of streaming imagery standards
Imagery in Cesium

- **Web Map Service** (WMS) - An OGC standard for requesting map tiles for a geographic region from distributed geospatial databases. In Cesium, see WebMapServiceImageryProvider.

- **Tile Map Service** (TMS) - A REST interface for accessing map tiles. Tiles can be generated with MapTiler or GDAL2Tiles. In Cesium, see TileMapServiceImageryProvider.

- **OpenGIS Web Map Tile Service** (WMTS) - An OGC standard for serving pre-rendered georeferenced map tiles over the Internet. In Cesium, see WebMapServiceImageryProvider.

- **OpenStreetMap** - Access to OpenStreetMap tiles or any Slippy map tiles. There are several ways to host these tiles. In Cesium, see OpenStreetMapImageryProvider.

- **Bing Maps** - Uses Bing Maps REST Services to access tiles. A Bing Maps key can be created at https://www.bingmapsportal.com/. In Cesium, see BingMapsImageryProvider.

- **Esri ArcGIS MapServer** - Uses the ArcGIS Server REST API to access tiles hosted by an ArcGIS MapServer. In Cesium, see ArcGisMapServerImageryProvider.

- **Google Earth Enterprise** - Provides access to the imagery stored in your organization's Google Earth Enterprise server. In Cesium, see GoogleEarthEnterpriseImageryProvider.
• Standard image files - Create a tile from a single image. In Cesium, see `SingleTile ImageryProvider`. 