Massive Terrain Rendering
Cesium Platform

- JavaScript, WebGL, Web Workers, Typed Arrays
- Chrome, Firefox, Internet Explorer 11 (sort of)
- Runs on Windows, Linux, Mac OS X, and Android
Powder Tracks demo
Terrain Rendering

• Geometry or shape
  – Heightmap
  – Triangle mesh

• Shading or color
  – Imagery (aerial photography, color map)
  – Various effects
Earth data

• Terrain and imagery collected by satellites and aerial instruments
  – Shuttle Radar Topography Mission (SRTM)
  – National Elevation Dataset (NED)
  – Landsat satellites
  – Digital Globe, SPOT, etc.

• Terrain usually provided as a heightmap
Heightmap
Unfortunately…

- The best data is hidden:
  - Within private networks
  - Within classified networks
  - Behind pay walls
  - By strict terms of use restrictions
- Data is huge – up to hundreds of terabytes
- We can’t access it ahead of time to transform it to a different form.
Terrain/imagery sources

- Web Map Service (WMS)
- Web Map Tile Service (WMTS)
- Tile Map Service (TMS)
- ArcGIS Map and Image servers
- OpenStreetMap
- Bing Maps
- Google Maps
Layers are different

• Different:
  – Map projections
  – Extents
  – Tiling schemes

• May need on-the-fly adjustments:
  – Hue, saturation, gamma
Agenda

• LOD algorithm – Chunked LOD
• Processing terrain for rendering
• Mapping imagery to terrain
• Map reprojection on the GPU
• Horizon culling
Agenda

• LOD algorithm – Chunked LOD
• Processing terrain for rendering
• Mapping imagery to terrain
• Map reprojection on the GPU
• Horizon culling
Chunked LOD

0 1

2

http://tulrich.com/geekstuff/chunklod.html

[Ulrich02]
Tiles
If current tile meets screen-space error requirement
    Add to render list
Else if all children are loaded
    Recurse on visible children
Else
    Queue not-yet-loaded children to load
    Add to render list
Screen-space error (SSE)

\[ SSE = \frac{\varepsilon x}{2d \tan \frac{\theta}{2}} \]

- \( \varepsilon \) = geometric error of tile or level, in meters
- \( x \) = width of the screen, in pixels
- \( d \) = distance to tile, in meters
- \( \theta \) = camera field of view, in radians
Geometric Error, $\varepsilon$

Geometric error is constant per level in the quadtree

[Klein96]
Distance to tile, $d$

- **Accurate**: distance to closest triangle
- **Simple**: distance to bounding sphere
- **Balanced**: distance to
  - Maximum height surface
  - East and West boundary planes
  - North and South approximate boundary planes
LOD Demo
Agenda

• LOD algorithm – Chunked LOD
• Processing terrain for rendering
• Mapping imagery to terrain
• Map reprojection on the GPU
• Horizon culling
Efficient rendering requires:

- Quadtree of tiles
- Tiles with a *reasonable* number of vertices
- A known geometric error for tiles in each level
- Just one set of tiles
  - Stitching them together at runtime is expensive
Terrain processing server

- Make it easy to import terrain data
- Offer an optimal (performance, visual quality) experience in Cesium
- Currently geometry only – no imagery
Terrain tileset

- Quadtree of tiles
- Built from multiple input heightmaps
  - Ex: GTOPO30 1km for the whole world, SRTM 90m between -60 and 60 degrees latitude, and NED for the United States
- Different max level in different areas
- About 150 kilometers error at root level
If tile does not overlap source extent
  return

If tile overlaps less than 64x64 source samples
  Add source samples to tile
Else
  Recurse on four child tiles
  Aggregate four child meshes
  Simplify combined mesh to geometric error target
Leaf tile
Non-leaf tile
Non-leaf tile

[Garland95]
Non-leaf tile

[Garland95]
Tile demo
Agenda

• LOD algorithm – Chunked LOD
• Processing terrain for rendering
• Mapping imagery to terrain
• Map reprojection on the GPU
• Horizon culling
Load Pipeline

**Terrain Tile**

- **Request**
- **Transform**
- **Create Resources**

- **Associate Imagery**

**Imagery Tile**

- **Request**
- **Create Texture**
- **Reproject**

- **Upsample**
- **Transform**
- **Create Resources**
Mapping imagery to terrain
**Fragment Shader**

```csharptext
vec2 textureCoordinates = terrainTileTextureCoordinates * scale + translation;
vec4 sample = texture2D(texture, textureCoordinates);

vec2 alphaMultiplier = step(extentMin, terrainTileTextureCoordinates);
float textureAlpha = textureAlpha * alphaMultiplier.x * alphaMultiplier.y;

alphaMultiplier = step(vec2(0.0), extentMax - terrainTileTextureCoordinates);
textureAlpha = textureAlpha * alphaMultiplier.x * alphaMultiplier.y;

float alpha = textureAlpha * sample.a;
```
Texture Scale

\[
scaleX = \frac{\text{terrainWidth}}{\text{imageryWidth}}
\]

\[
scaleY = \frac{\text{terrainHeight}}{\text{imageryHeight}}
\]
Texture Translation

\[
\text{translationX} = \text{scaleX} \times \text{offsetX} / \text{terrainWidth} \\
\text{translationY} = \text{scaleY} \times \text{offsetY} / \text{terrainHeight}
\]
Texture Extent

offsetX

offsetY

extentMinX = offsetX / terrainWidth
extentMinY = offsetY / terrainHeight
extentMaxX = (offsetX + imageryWidth) / terrainWidth
extentMaxY = (offsetY + imageryHeight) / terrainHeight
vec2 textureCoordinates = terrainTileTextureCoordinates * scale + translation;
vec4 sample = texture2D(texture, textureCoordinates);

vec2 alphaMultipler = step(extentMin, terrainTileTextureCoordinates);
float textureAlpha = textureAlpha * alphaMultipler.x * alphaMultipler.y;

alphaMultipler = step(vec2(0.0), extentMax - terrainTileTextureCoordinates);
textureAlpha = textureAlpha * alphaMultipler.x * alphaMultipler.y;

float alpha = textureAlpha * sample.a;
Agenda

• LOD algorithm – Chunked LOD
• Processing terrain for rendering
• Mapping imagery to terrain
• **Map reprojection on the GPU**
• Horizon culling
Map Reprojection

Geographic / Plate Carrée (EPSG:4326)  Web Mercator (EPSG:3857)
Web Mercator Reprojection

mercatorY = 0.5 * log((1.0 + sinLatitude) / (1.0 - sinLatitude))
mercatorOffset = mercatorY – southMercatorY
mercatorT = mercatorOffset * oneOverMercatorHeight
GPU Reprojection Gotchas

Problem: Limited floating-point precision

Solution: Texel spacing cutoff
Texel Spacing Cutoff

![Graph showing the relationship between level of detail and max. error (texture⁻¹). The x-axis represents the level of detail, ranging from 0 to 20. The y-axis represents max. error (texture⁻¹), ranging from 1 to 0.000001. The graph includes lines for 1 texel, 1/10 texel, and 0.0001, showing a decreasing trend as level of detail increases.](image-url)
Problem: Many mobile GPUs have limited fragment shader precision.

Solution: Reproject in the vertex shader instead of the fragment shader.
Doarama demo
Agenda

- LOD algorithm – Chunked LOD
- Processing terrain for rendering
- Mapping imagery to terrain
- Map reprojection on the GPU
- Horizon culling
Horizon Culling
Ellipsoid-scaled space

Ellipsoid: \( \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \)

- \( x, y, z \) is any position on the surface of the ellipsoid
- \( a, b, c \) are ellipsoid semi-axis lengths

Ellipsoid-scaled space:
- Is centered on the ellipsoid
- Axes are aligned with ellipsoid axes
- Ellipsoid is a unit sphere.

WGS84 Length (meters)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>6,378,137.0</td>
</tr>
<tr>
<td>b</td>
<td>6,378,137.0</td>
</tr>
<tr>
<td>c</td>
<td>6,356,752.314245</td>
</tr>
</tbody>
</table>

Scale to ellipsoid-scaled space:

\[
D = \begin{bmatrix}
\frac{1}{a} & 0 & 0 \\
0 & \frac{1}{b} & 0 \\
0 & 0 & \frac{1}{c}
\end{bmatrix}
\]

[Cozzi10]
Horizon occlusion point
Horizon occlusion point

\[ \|\overrightarrow{OP}\| = \frac{1}{\cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)} \]

\[ \cos(\alpha) = \hat{O}V \cdot \hat{O}P \]

\[ \sin(\alpha) = \|\hat{O}V \times \hat{O}P\| \]

\[ \cos(\beta) = \frac{1}{\|\overrightarrow{OV}\|} \]

\[ \sin(\beta) = \frac{\sqrt{\|\overrightarrow{OV}\|^2 - 1}}{\|\overrightarrow{OV}\|} \]

[Ring13b]
Testing the occlusion point

Plane Test:
\[ \overrightarrow{VT} \cdot \overrightarrow{VC} > \|\overrightarrow{VC}\|^2 - 1 \]

Cone Test:
\[ \frac{(\overrightarrow{VT} \cdot \overrightarrow{VC})^2}{\|\overrightarrow{VT}\|^2} > \|\overrightarrow{VC}\|^2 - 1 \]

V - viewer position
C - center of unit sphere
H - horizon point
T - target point to test against horizon plane
P - projection of VH onto VC
Q - projection of VT onto VC

[Ring13a]
Horizon culling demo
Cheating
Acknowledgements

- Frank Stoner
- Scott Hunter
Open problems

- Placing lines, polygons, etc. on terrain
- Shadows casted by and casted onto terrain
- Faster faster faster!
- Lighting imagery – it’s usually already lit!
Final Thoughts

Cesium is open source, join us!
http://cesiumjs.org

Thanks for listening!
References