# Icosatree Data Partitioning of Massive Geospatial Point Clouds with User-Selectable Entities and Surface Modeling

# Thesis Advisory Panel

- Dr. Reynold Bailey
  - Committee Chair
  - Associate Professor
  - Associate Undergraduate Coordinator
- Dr. Joe Geigel
  - Committee Reader
  - Professor
- Dr. Zack Butler
  - Committee Observer
  - Associate Professor
  - Associate Graduate Coordinator

In Partial Fulfillment of the Requirements for the Degree of

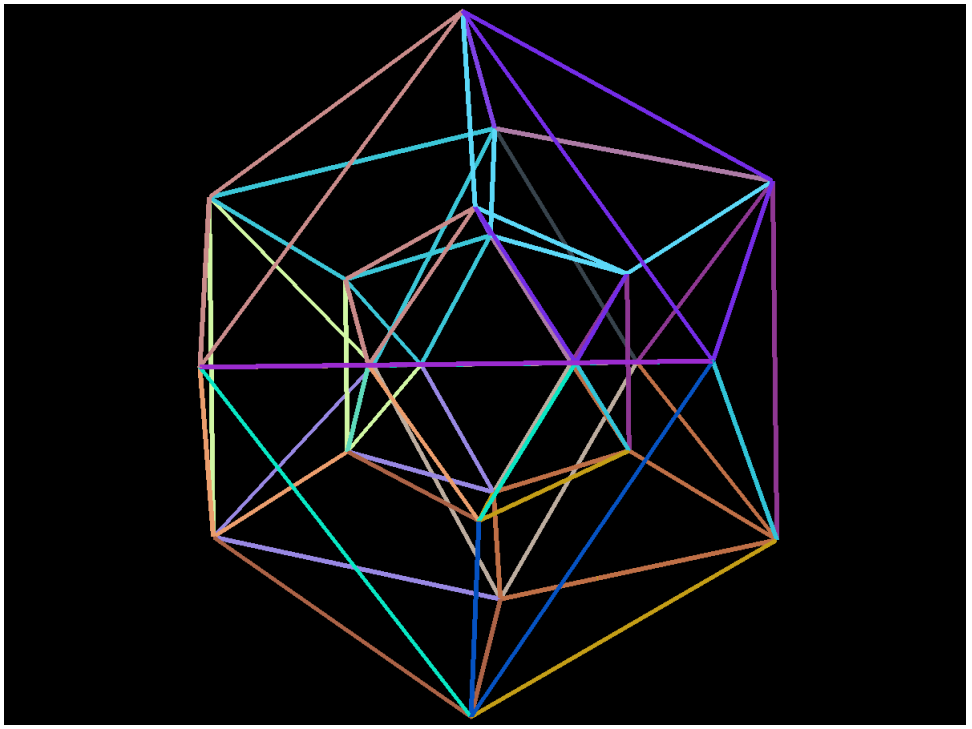
**Master of Science in Computer Science**

By Stephen Ranger

December 12, 2016

# Introduction

Applying Icosatree Data Structure and Adding Selection, Triangulation, and Export Options to Point Clouds



# Related Work

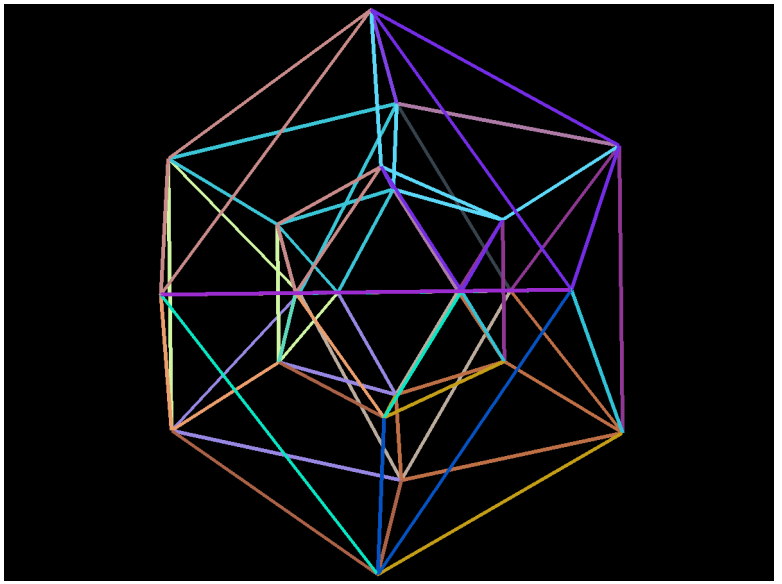
- Lingyun Yu, Konstantinos Efstathiou, Petra Isenberg, and Tobias Isenberg. 2016
  - CAST: Effective and Efficient User Interaction for Context-Aware Selection in 3D Particle Clouds
- Ruggero Pintus, Enrico Gobbetti, and Marco Agus. 2011
  - Real-time Rendering of Massive Unstructured Raw Point Clouds using Screen-space Operators
- K Wenzel, M Rothermel, D Fritsch, and N Haala. 2014
  - An out-of-core octree for massive point cloud processing
- Prashant Goswami, Yanci Zhang, Renato Pajarola, and Enrico Gobbetti. 2010
  - High quality interactive rendering of massive point models using multi-way kd-trees
- Jürgen Richter, RicoDöllner. 2010
  - Out-of-core real-time visualization of massive 3d point clouds

# Methods: Overview

- Icosatree designed to apply Cartesian values to projected surface
  - Tree cells tangential to WGS84 projection
  - Higher fill ratio compared to Octree
  - Uniform cell shape in visualization
- Selection algorithm adds analysis utility to visualization
  - 2D lasso selection for intuitive interaction
  - Configurable steps allow tailoring to use-case
  - Export functionality to support external processing

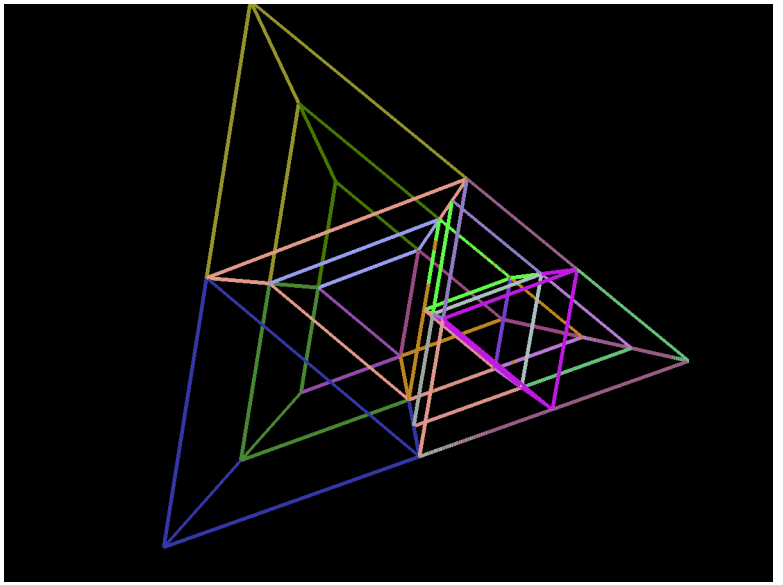
# Methods: Icosatree

- Icosatree: Icosahedron Data Structure
  - Based on Octree algorithm
  - Root cell splits into twenty triangular prism cells
    - Labeled A-T
  - Triangular prism cells split into eight child cells
    - Labeled 0-7
  - Traversed in same manner as Octree



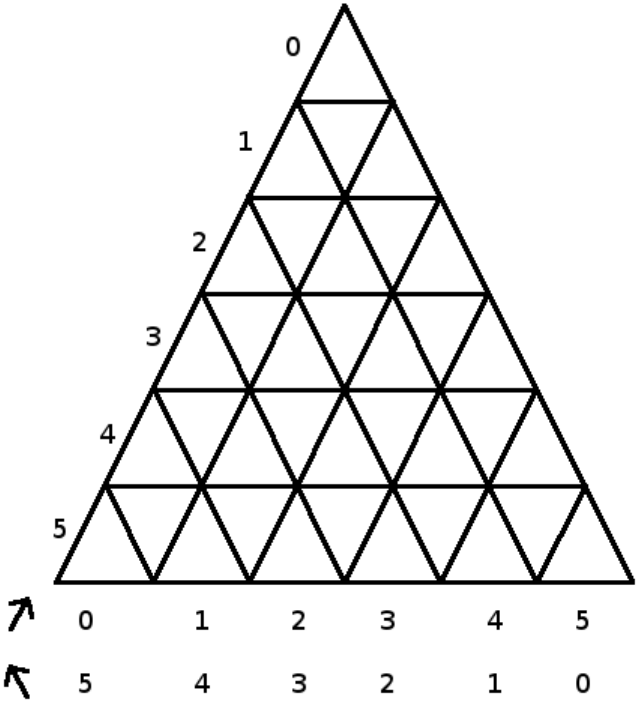
# Methods: Icosatree

- Root cell defined as axis-aligned bounding box
  - Does not store point data
- Twenty triangular prism cells encompass WGS84 surface
  - Average Effective altitude range: [-1434 km, 1465 km]
    - Corners of bottom face to center of top face
  - Max radius based on Octree radius ($2^{23}$ meters)
  - Min radius = Max radius / 2.0



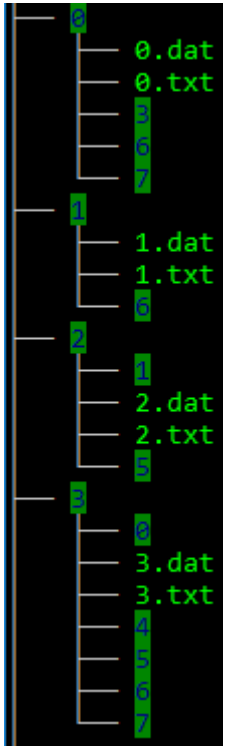
# Methods: Icosatree

- Triangular Prism Point Allocation
  - Split into sub-cells
  - Triangular Coordinates used for indexing
- Tree insert
  - Find child of root containing point
  - Insert into child
  - Compute triangular coordinate index
  - If sub-cell empty: insert point
  - Else:
    - If new point closer to sub-cell center: swap point
    - Insert remaining point into child cell



# Methods: Tree Data Structure

- Data structure built separately from visualization
- Application stores data structure in file structure defined by cell path
- Root directory contains point attribute CSV file and cell point statistics
- Each directory contains
  - Directories for child cells
  - Data file for stored points
  - Text file listing available child cells
- Accessible from local file system or over HTTP connection



# Methods: Visualization Application

- OpenGL / Java based (JOGL)
- Geospatial Navigation
- WGS84 based Earth
- Digital Elevation Model support
- Slippy map tiles imagery support (ie. OpenStreetMap, Stamen)

# Methods: Tree Structure Rendering Element

- Handles Octree and Icosatree implementations
- Initializes tree with metadata (CSV attributes and cell statistics)
- Stores cell point data in Dynamic Vertex Buffer Pool
- Rendering Operations
  - Compute cells to render (frustum culling)
  - Upload any pending cells to vertex buffer pool
  - Build bounding volume for clipping planes
  - Render points
    - For each VBO pool, render all cells mapped to each VBO segment
    - Limit enable/disable calls

# Methods: Tree Structure Rendering Element

- Determining cells to render
  - Starting with root cell
  - Frustum intersection test
    - IN: Skip frustum tests for remaining cells below this cell
    - INTERSECTS: Continue as-is
    - OUT: Return, skip children
  - Check level of detail
    - On-screen area vs. user-defined area threshold
  - If level of detail within threshold, mark to render
    - If no point data, skip children, don't render, and queue point request
  - If cell has children, continue at start for each child

# Methods: Selection and Triangulation

- 2D lasso defined by user using mouse
- Closed polygon from lasso created
- Projected into scene as selection frustum
- Points within frustum are stored and highlighted

# Methods: Selection and Triangulation

- User defined pruning stage order

Orthonormal

Obstruction

Altitude threshold

# Methods: Selection and Triangulation – Orthonormal Pruning

- Orthonormal Pruning
  - Computes global eigenvector as surface normal
  - For each point, compute local eigenvector
    - Remove point if furthest neighbor is outside of grid size * 2 (outlier removal)
  - If dot product between global and local normal above threshold; remove point
    - Normal threshold of 1.0 means normals are parallel, 0.0 means perpendicular
- Useful for removing flat surfaces
- Unfortunately removes flat building roofs
  - However, triangulation does successfully connect building edges
  - Doesn’t work if no points for sides of buildings (aerial scans)

# Methods: Selection and Triangulation – Obstruction Pruning

- Obstruction Pruning
  - For each point:
    - Compute angle between current point and all other points relative to camera
    - If computed angle less than threshold value AND behind test point; remove current point
- Useful for removing points hidden by others
- Not useful for natural surfaces; removes detail
- Helps triangulation build smoother surfaces

# Methods: Selection and Triangulation – Altitude Threshold Pruning

- Altitude Threshold Pruning
  - Compute min/max altitude of all points
  - Remove any points within threshold of minimum
- Useful for cropping noise or ground level unevenness

# Methods: Selection and Triangulation

- Once all points are collection and (optionally) pruned
  - Convert XYZ coordinates into XY-depth in screen space
  - Optionally remove occluded screen space points
  - Use Delaunay triangulation algorithm to compute triangle mesh
  - Project XY-depth mesh back to XYZ coordinates
  - Send list of triangles to rendering component

# Results: Point Cloud Dataset

- Morro Bay, San Simeon, CA
  - Geodesic Bounds: (-120.88, 35.36), (-120.84, 35.40)
  - 163,251,931 total points

# Results: Dataset Analysis

- Dataset construction

Table 5.1: Octree Building Statistics

| Sub-Cell Size | Tree Creation Time | Cells Created | Filesystem Export Time | Max Points Per Cell | Average Points Per Cell | Max Tree Depth | Directory Count | File Count | Total File Size |
|---|---|---|---|---|---|---|---|---|---|
| 50x50x50 | 5h 8m | 1,093,494 | 20m 48s | 15548 | 150 | 32 | 1,093,494 | 2,186,989 | 11 GiB, 124 MiB |
| 100x100x100 | 4h 42m | 389,771 | 15m 55s | 44223 | 419 | 31 | 389,771 | 779,543 | 11 GiB, 109 MiB |

Table 5.2: Icosatree Building Statistics

| Sub-Cell Size | Tree Creation Time | Cells Created | Filesystem Export Time | Max Points Per Cell | Average Points Per Cell | Max Tree Depth | Directory Count | File Count | Total File Size |
|---|---|---|---|---|---|---|---|---|---|
| 100x100x100 | 5h 25m | 725,672 | 26m 41s | 17123 | 225 | 32 | 725,672 | 1,451,345 | 11 GiB, 115 MiB |
| 100x20 | 7h 18m | 653,475 | 32m 38s | 10233 | 250 | 31 | 653,475 | 1,306,951 | 11 GIB, 114 MiB |
| 100x100 | 6h 27m | 583,163 | 25m 43s | 14569 | 280 | 31 | 583,163 | 1,166,327 | 11 GiB, 112 MiB |
| 150x100 | 6h 14m | 340,758 | 23m 54s | 22618 | 480 | 30 | 340,758 | 681,517 | 11 GiB, 107 MiB |
| 200x100 | 5h 49m | 235,156 | 20m 4s | 38081 | 695 | 29 | 235,156 | 470,313 | 11 GiB, 105 MiB |

# Results: Dataset Analysis

- Data Structure Statistics

Table 5.3: Octree Cell Statistics

| Tree Depth | Type | X-Span (m) | Y-Span (m) | Z-Span (m) | Side Area ($m^2$) | Volume ($m^3$) |
|---|---|---|---|---|---|---|
| 18 | AABB | 64 | 64 | 64 | 4096 | 262144 |
| 19 | AABB | 32 | 32 | 32 | 1024 | 32768 |
| 20 | AABB | 16 | 16 | 16 | 256 | 4096 |
| 21 | AABB | 8 | 8 | 8 | 64 | 512 |
| 22 | AABB | 4 | 4 | 4 | 16 | 64 |

Table 5.4: Icosatree Cell Statistics

| Tree Depth | Type | X-Span (m) | Y-Span (m) | Z-Span (m) | Depth (m) | Top Area ($m^2$) | Bottom Area ($m^2$) | Volume ($m^3$) |
|---|---|---|---|---|---|---|---|---|
| 18 | Triangle Prism | 83.7769 | 95.9998 | 64.0 | 29.8935 | 2709.8502 | 2709.8296 | 81006.642 |
| 19 | Triangle Prism | 41.8885 | 48.0 | 32.0 | 14.9468 | 677.4626 | 677.46 | 10125.8496 |
| 20 | Triangle Prism | 20.9443 | 24.0 | 16.0 | 7.4734 | 169.3656 | 169.3653 | 1265.7324 |
| 21 | Triangle Prism | 10.4721 | 12.0 | 8.0 | 3.7367 | 42.3414 | 42.3414 | 158.2166 |
| 22 | Triangle Prism | 5.2361 | 6.0 | 4.0 | 1.8683 | 10.5854 | 10.5853 | 19.7771 |

# Results: Rendering Analysis

Table 5.6: Rendering Statistics (depth = 3.0)

| Cell Type | Point Count | Cell Count | Culling | Rendering | Building Bounds |
|---|---|---|---|---|---|
| Icosatree (100x100) | 1097116 | 200 | 67 ms | 636 $\mu s$ | 53 $\mu s$ |
| Icosatree (150x100) | 2280826 | 201 | 47 ms | 408 $\mu s$ | 60 $\mu s$ |
| Icosatree (200x100) | 3825534 | 201 | 96 ms | 332 $\mu s$ | 52 $\mu s$ |
| Octree (AABB) | 880582 | 99 | 39 ms | 379 $\mu s$ | 35 $\mu s$ |

Table 5.7: Rendering Statistics (depth = 6.0)

| Cell Type | Point Count | Cell Count | Culling | Rendering | Building Bounds |
|---|---|---|---|---|---|
| Icosatree (100x100) | 2391155 | 494 | 138 ms | 910 $\mu s$ | 310 $\mu s$ |
| Icosatree (150x100) | 4368414 | 494 | 92 ms | 630 $\mu s$ | 140 $\mu s$ |
| Icosatree (200x100) | 6239669 | 493 | 110 ms | 502 $\mu s$ | 87 $\mu s$ |
| Octree (AABB) | 1915896 | 223 | 126 ms | 312 $\mu s$ | 46 $\mu s$ |

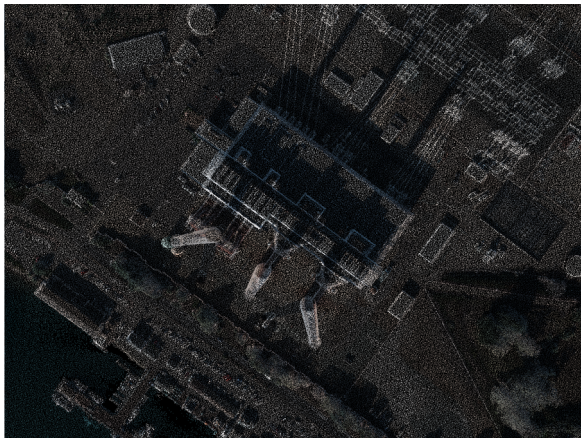
Table 5.5: Rendering Statistics (depth = 10.0)

| Cell Type | Point Count | Cell Count | Culling | Rendering | Building Bounds |
|---|---|---|---|---|---|
| Icosatree (100x100) | 3355393 | 1276 | 252 ms | 1 ms 172 $\mu s$ | 265 $\mu s$ |
| Icosatree (150x100) | 6065591 | 1533 | 236 ms | 1 ms 402 $\mu s$ | 220 $\mu s$ |
| Icosatree (200x100) | 7325988 | 1522 | 187 ms | 1 ms 765 $\mu s$ | 325 $\mu s$ |
| Octree (AABB) | 3692901 | 630 | 103 ms | 755 $\mu s$ | 91 $\mu s$ |

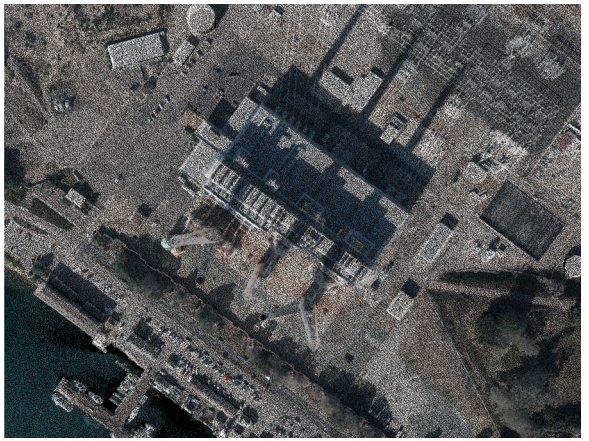
# Results: Rendering Analysis

- Rendering with level of detail

Depth = 3.0

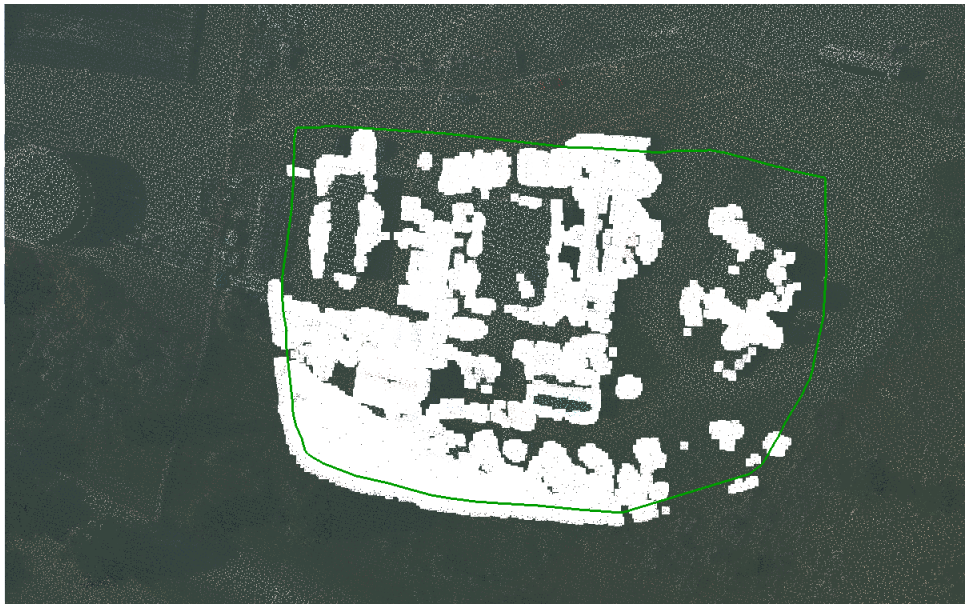


Depth = 6.0

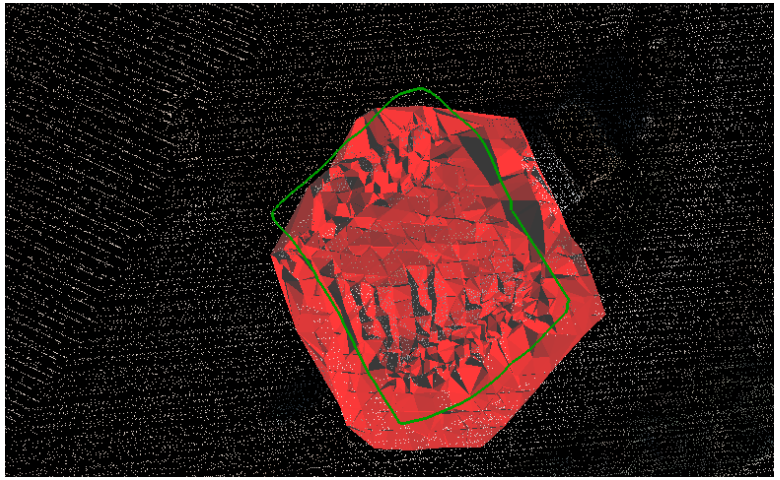


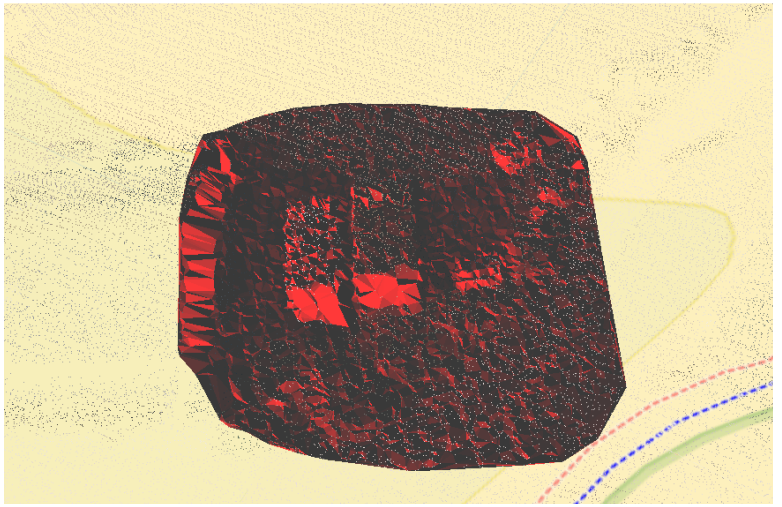
Depth = 10.0

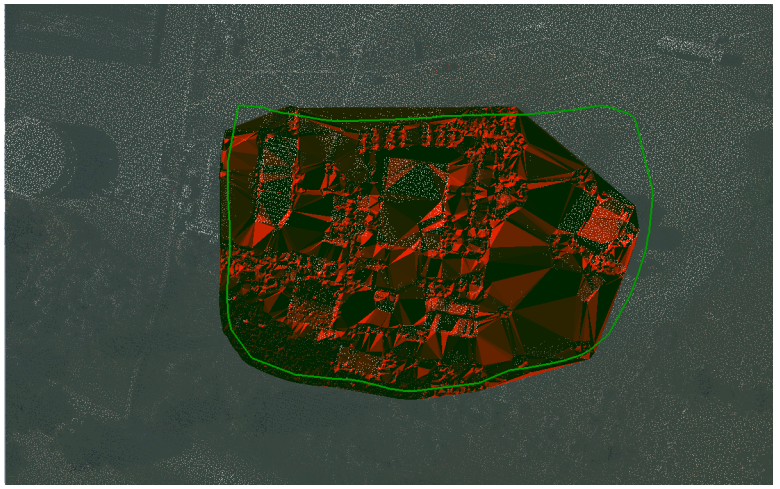
# Results: Point Selection Output

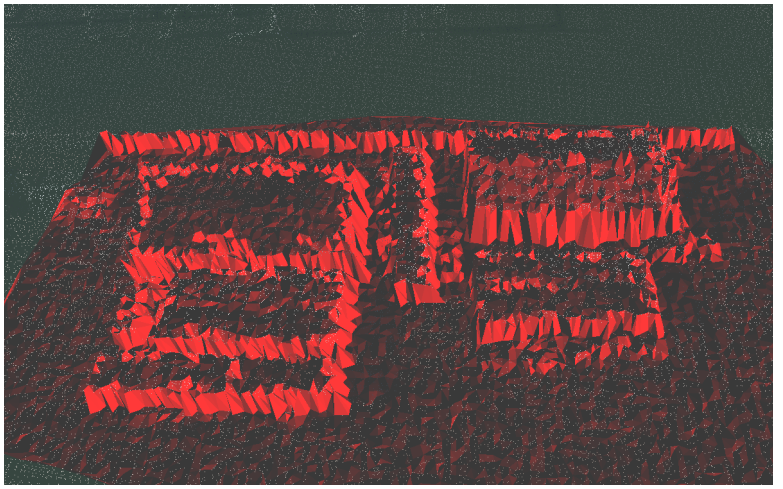



# Results: Triangulation Output



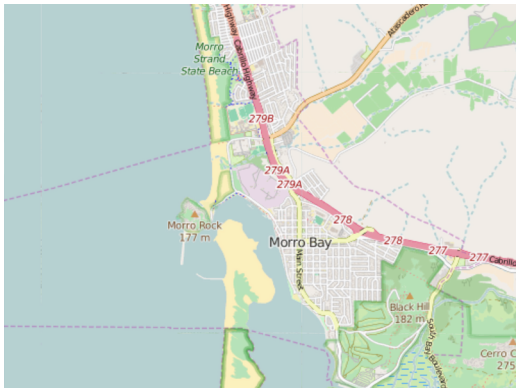






# Discussion: Current Status

- Octree/Icosatree dataset creation and rendering
  - X, Y, Z, Red, Green, Blue, Altitude, Intensity attributes
- Geospatial camera navigation
  - Geodesic anchor
  - Spherical offset
- Earth scene element
  - WGS84 ellipsoid
  - Digital Elevation Model for altitude
  - Slippy Map Tiles for imagery
  - Tessellated level of detail



# Discussion: Current Status

- Point Selection with lasso
- - Point pruning
  - Orthonormal surface
  - Obstruction
  - Altitude
- Triangulation using Delaunay algorithm
- Export using PLY format

# Conclusion

- Performance overall promising
  - Icosatree cells visually pleasing
  - Cell point allocation more efficient
  - Difficult to determine optimal sub-cell sizes
    - Cells can end up too large for GPU
    - Requires more cells to be rendered
  - Triangular Prism Bounds computationally intensive compared to axis-aligned bounds
  - Point Selection fairly quick
  - Pruning and Tessellation are slow

# Future Work

- Custom point attribute mappings
- Import LAS/LAZ files directly
- Icosatree cell and sub-cell size optimization
- Updated dataset creation tool (threading, in-place file caching)
- Pruning and triangulation optimizations
- OpenCL or CUDA support
- Various rendering engine fixes

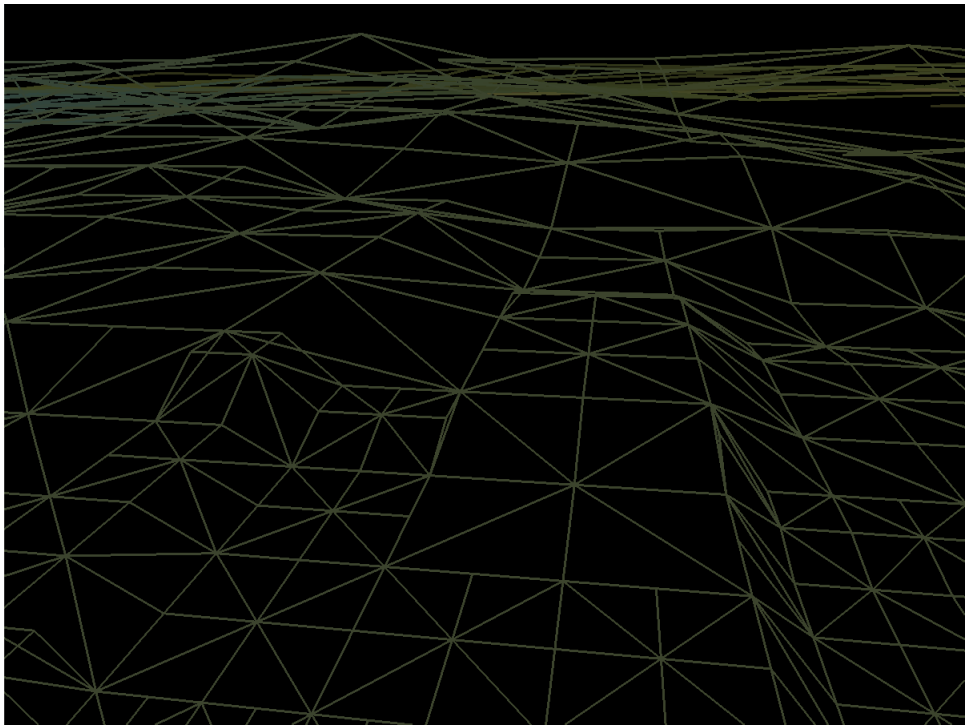
End

# Geospatial Navigation

- WGS84 Surface navigation
  - Geospatial anchor
  - Spherical coordinate offset
  - Left mouse intersects navigation surface (ellipsoid) to set anchor
  - Right mouse modifies elevation and azimuth offset
  - Middle mouse scroll modifies range

# WGS84 Earth

- Tessellated ellipsoid
- Level of detail based on distance of camera from geometry
- Digital Elevation Model data used for ellipsoid elevation
- Slippy map tiles used for imagery

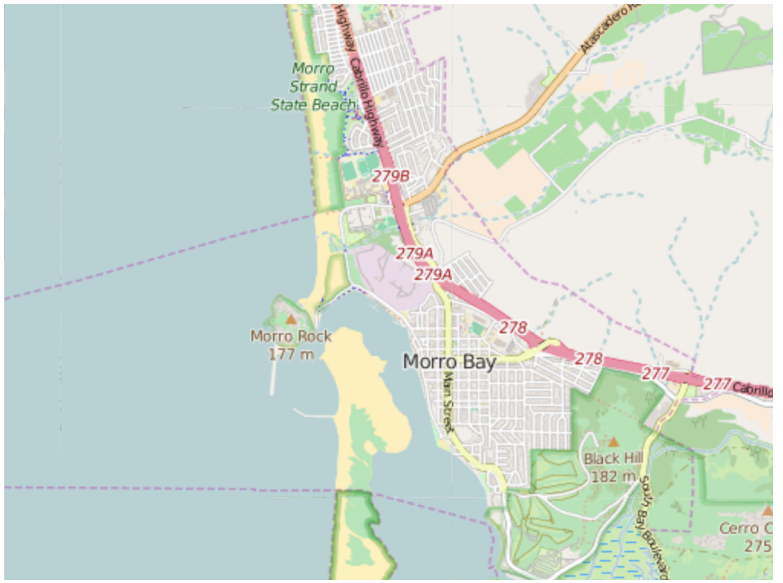


# Digital Elevation Model

- Digital Elevation Model
  - Creates index of all available DEM files
  - Location set with JVM flag -Ddem3.directory
  - Recursively searches directory for .hgt files
  - Stores lookup using lon/lat integer index
  - File random access for runtime value lookup

# Slippy Map Tiles

- Slippy map tiles
  - Support for OpenStreetMap and Stamen
    - -Dosm.server=<base_url>
    - -Dstamen.server=<base_url>
    - -Dimage.cache=<filesystem_path>
  - URL format for geospatial tile imagery
  - Quadtree based tiling system
  - <base_url>/zoom/x/y.png (or .jpg for Stamen)
  - 256px images




# Dynamic Vertex Buffer Pool

- Dynamic Vertex Buffer Object Pool
  - Actually pool of pools
  - Defines max segment size and number of segments per VBO
  - Pools created for max segment size and below
    - Loops until segment pool is less than 100 points
    - ie. max = 1000, pools = {1000, 500, 250, 125, 62}
  - Pool starts with one VBO and adds/removes additional as necessary
  - Each pool stores global index so number of VBO's seamless
  - Tree Cells store pool index and segment index for rendering
  - Allows tree cells to be at least 50% space efficient