- I/O
  - In your graphics application, read in the binary file for the terrain mesh before entering your main loop.
  - Use the provided binary file with the following format or export the terrain.fbx file.
    - ➢ uint32_t vert_count; // Number of vertices
    - ➢ std::vector<float3> pos; // Array of vert_count vertex positions
    - ➢ std::vector<float3> norms; // Array of vert_count normals
    - ➢ std::vector<float2> uvs; // Array of vert_count texture coordinates
  - You should keep a copy of the positions available in system memory for further calculations.
  - You will need to be able to render this terrain mesh in wireframe.

Debug Renderer

- Add support for rendering AABBs

AABB Calculation

- Write a function that calculates an AABB based off of a triangle index
- Triangle indices are indices for triples of vertices or vertex indices
- Non-indexed vertex array example
  - Triangle index 0 represents: pos[0], pos[1], pos[2]
  - Triangle index 2 represents: pos[6], pos[7], pos[8]
- Indexed vertex array example
  - Triangle index 0 represents: pos[ind[0]], pos[ind[1]], pos[ind[2]]
  - Triangle index 2 represents: pos[ind[6]], pos[ind[7]], pos[ind[8]]

Data Generation - 20%

- Generate additional data for the triangulated mesh.
- This data must be generated before your main loop.
- Calculate and store the centroid for each triangle (average of three verts)
- You will create an AABB for each triangle.
- Generate a std::vector of triangle indices for the terrain mesh
  - This is simply an std::vector of ints where [0] = 0, [1] = 1, [2] = 2, etc.
- First read the file to get the vertcount (sizeof(uint32_t)). Then read all of the vertex positions (sizeof(float3)*terrain_vert_count).
- You only truly need the vertex positions for this lab.

BVH - 50%

- Create a BVH class
- Build a BVH using the insertion construction method
  - Use the triangle bounds as the objects to insert
  - Keep the BVH ignorant of the specifics of the data
  - Ex: Have leaf nodes store an AABB and an integer id
  - Use the Manhattan distance between AABB centers as your cost function

Testing - 25%

- Add an AABB to your scene that translates around on X,Y,Z
- write a traversal function that takes an AABB and tests it against the nodes in the tree
- of a node bounds is hit, draw the bounds and recursively perform the test on its children
- perform this traversal in your main loop

Update - 5%

- All logic in update.