CS350
Bounding Volume Hierarchies

Description

The objective of this assignment is to develop a basic Bounding Volume Hierarchy for Static Objects. In order to get full grade you must:

- Implement BVH - TopDown properly
  ```
  void build_top_down(aabb const* bvs, unsigned bv_count, unsigned max_depth);
  ```
- Implement BVH - BottomUp properly
  ```
  void build_bottom_up(aabb const* bvs, unsigned bv_count);
  ```
- Implement BVH - Incremental properly
  ```
  void insert(aabb const& bv, unsigned id, unsigned max_depth);
  ```
- **Have a functional demo that showcase the three building methods**. This assignment is more focused on the demo than the tests. Make sure that in the demo the tree is easy to navigate

A public interface has been provided, you will notice that the class is not templatized. It will only hold AABBs, in order to refer to the original objects only their indices will be used. You may add private members but **do not change the public interface.**

Top down

- Must be fast to compute
- Partitioning axis: Choose the longest cardinal axis from the current data (for axes that have same length, prefer them in order XYZ).
- Partitioning point: Choose the average of the centroids
- Do not split objects, classify them by their center
- Base cases:
  - Any of the splitting child ends empty.
  - Both children bounding volumes are as large as the parent bounding volume.
  - A single object in the node.
  - Maximum depth reached.

Bottom up

- The slowest of the 3 methods
- Use a naive approach, consider every possible pair
- Merge the pair that minimizes the Surface Area.
    - If two pairs have the same surface area with an epsilon (e.g. 0.001), choose the one with minimum volume
- Continue until a single node is left.

Incremental

- When traversing the tree for insertion, choose the node in which adding the object would have less impact in the Surface Area
- Have a maximum depth end condition
- This method makes more sense for moving objects. Your obtained result may not make much visual sense. Even more, if the input is randomized, each build may be completely different.
- **Have a randomize option in the demo, that will shuffle objects prior to insertion**

Others

- Implement a level order traversal function that will receive a callback function, this function will be used to debug information. This function predicate is expected is of the form `void(bvh_indexed::node const*)`. Go through each node in level order, invoking the callback with each node.
- Implement a function that will print debug information into a stream, for example (but not necessarily equal)

```
...
NODE [0x56317f8594b0]
                BV: -2.128090 -0.004774 -1.380900 2.128090 3.292660 1.380900
            Volume: 38.760403
      Surface area: 69.792084
          Children:
                   NODE [0x56317f860f10]
                       Depth: 9
                        Size: 273
                   NODE [0x56317f876bf0]
                       Depth: 9
                        Size: 263
...
```

- You will notice that this assignment requires **shapes** (Such as `aabb, triangle,` etc.), create them in the file `shapes.cpp/hpp`. These structures will hold data that represents geometrical shapes, but they must not contain graphics data
- A function `load_triangles_cs350` will need to be implemented. Its implementation should be trivial, as the format is just:

- An integer N: Amount of triangles
- N triangles

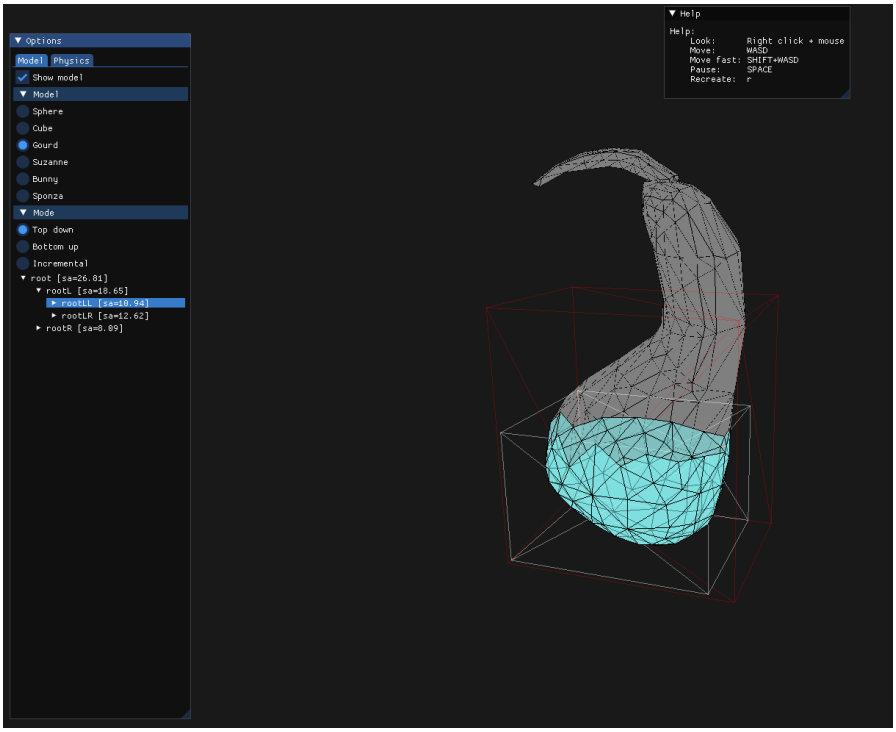
```
682
-0.130033 1.48208 -0.75046 -0.361103 1.58941 -0.812066 -0.272128 1.67716 -0.925396
-0.109241 1.65029 -0.943897 -0.130033 1.48208 -0.75046 -0.272128 1.67716 -0.925396
-0.174663 1.86957 -1.10977 -0.109241 1.65029 -0.943897 -0.272128 1.67716 -0.925396
-0.383406 1.85176 -1.02843 -0.174663 1.86957 -1.10977 -0.272128 1.67716 -0.925396
-0.567643 1.81243 -0.887877 -0.383406 1.85176 -1.02843 -0.272128 1.67716 -0.925396
-0.361103 1.58941 -0.812066 -0.567643 1.81243 -0.887877 -0.272128 1.67716 -0.925396
...
```

- A function `triangles_to_aabbs` will need to be implemented: Given a set of triangles, return their corresponding bounding volumes

Demo

The user must be able to:

- Switch the mesh
- Switch the building method (TopDown, BottomUp, Incremental)
- Display the BVH tree. User should be able to view the nodes of the BVH along with this information (Recommended `ImGui::TreeNodeEx`):
  - Debug draw of the triangles contained from this node down to its leaves
  - Debug draw the bounding volume of the node (wireframe strongly preferred)
  - View the property that is relevant for node selection (i.e. heuristic, surface area, volume, ...)



Notes

- You may include extra files if you see fit. You may not include other libraries.
- Comment your code: Files + Functions headers too!
- **Use GIT as a control version to track your progress**

Delivery

- Files to submit:
  - All source code!
  - CMakeLists.txt (as you may have added new files)
  - `.git` folder
- DO NOT SUBMIT BINARY FILES NOR INTERMEDIATE FILES