cs350 - assignment#3
bounding volume hierarchies

Due date: February 20th, 23:55

1 Requirements

Implement the following bounding volume hierarchy construction techniques:

– Top-Down:

1. Create a bounding volume for all objects
2. Decide in which axis should you split the bounding volume. (longest axis recommended)
3. Given that axis, decide in which point will the bounding volume will be split: (center mean recommended)
4. Terminating conditions: Choose the method you prefer, whether all objects are to the same side or there are no more objects, ...etc.

– Bottom-Up

1. Merge a pair that minimizes some cost function (minimum surface area or volume recommended).
2. Consider this merged pair a new node to be merged with.
3. Repeat until only one node left

– Insertion

1. Build the BVH incrementally, adding one object at a time.
2. Every time an object is inserted, traverse through the branch that minimizes some cost function. (minimum surface area delta or volume delta recommended).

1.1 Other requirements

– Interactive demo in which the Bounding Volumes are rendered and the user can transform the objects. User should be able to select the BVH mode and render the levels. It is important that the user can interactively see which objects are in which level/node.

– Debug draw the BVH nodes bounding volume

– Preview a certain node of the BVH (with or without all it's children)

– Create random scenes

– Unit tests, they should ensure (at least) correctness of the BVH: (ie. all children objects are contained)

1.2 Starting point

This is a possible starting code.

```
template<typename T>
class bounding_volume_hierarchy {
public:
  struct node {
    std::vector<T>        objects{};
    aabb                  bounding_volume{};
    std::array<node*, 2>  children{};

    ~node();
    void destroy();
    void increase_bounding_volume(const aabb& bv);
  };
private:
  node* m_root;
public:
  bounding_volume_hierarchy();
  ~bounding_volume_hierarchy();
  node* build_top_down(const std::vector<T>& objects);
  void build_bottom_up(const std::vector<T>& objects);
  void add_object(const T& object);
  bool remove_object(const T& object, node** current = nullptr);
  void destroy();
  const node* root() const {return m_root;}
};
```

2 What to Submit

A zip file containing *.cpp, *.h. The CMakeLists.txt provided by the instructor will be used for compiling the project, along with the test files. You can use the IDE of your choice.

- Filename: ***cs350_studentlogin_assignmentnumber.zip***. For example for user *eder.b1* and assignment *#2*: cs350_eder.b1_2.zip.
- Files: *.cpp, *.h
- Assets: Inside **resources** folder
- (optional) readme.txt: Explain how to use the demo program. Alternatively, it can be shown in the application itself.