

# CS350 Project 3

## Synopsis

Continuing from the previous project, process the list of triangles into a bounding volume hierarchy, and modify the collision detection and triangle selection operations to use that hierarchy. Compare the efficiency of this Project 3 hierarchical representation and the Project 2 linear list.

## Instructions

### Triangle list:

For project 2, you produced a list triangles. Modify that to be a list of tree leaf nodes, one per triangle. A plausible class structure would be:

```
class TreeNode:  
    Box* aabb; // An AABB representing this node's extent.  
    Triangle* tri; // A triangle (for leaf nodes)  
    TreeNode *lChild, *rChild; // Two child treenodes (for internal nodes)
```

(**Or do better:** Eliminate the wasted space of **null pointers** in **every** node.)

### AABBS:

This project will require manipulation of AABBs. Be prepared to write some utility functions for this. In particular you'll need the ability to create a new AABB surrounding several existing AABBs.

### Hierarchical tree construction:

From the list of leaf nodes (triangles and their AABBs), create a Bounding Volume Hierarchy using a **top-down method**:

- **Termination criteria:** The recursion terminates when the input list is a single node, and that single node becomes a leaf node.
- **Split criteria:** Choose the axes of largest extent, and split the list of input AABBs using the “Median of BV centers” on that axis (or experiment with other split methods).

### Hierarchical tree traversal:

Replace the **Collision Detection** and **Triangle Selection** operations from project 2 (which traversed a linear list of triangles) to traverse the hierachal tree recursively.

**Include both optimizations mentioned in class:** (1) tracking the portion of the ray still in play, and (2) ordering (and possibly eliminating) the two recursive calls.

## **Documenting the tree:**

For this portion, keep the number of triangles relatively small - say under 10,000.

You must document the tree you create in one of several ways:

- Expand the on-screen documentation as directed in Project 2, or
- Produce a document named **report.pdf**, and include it in the submitted ZIP file.

The documentation must include:

- The number of leaf nodes in the tree (= #triangles)
- The minimum and maximum depth of the leaf nodes. (Expect some spread of depths around  $\log_2(\# \text{triangles})$ . Too much spread indicates an unbalanced tree. Avoid this.)
- A running average of ray/box intersection calculations per frame. (Expect far, far less the project 2's count of  $2 * \# \text{triangles}$ .)
- A method to view **all** the bounding boxes in the hierarchy. Either display all the bounding boxes at once or display them in batches (say upper 1/3 of levels, middle 1/3, and lowest 1/3).
- Use different colors for each level.

## **Stress test:**

In project 2, you calculated roughly how many triangles you could handle without dropping the frame rate. Do that again (**cautiously**) for this bounding volume hierarchy.

Increase the number of triangles (using **GOAL** in **student\_code.h**) in **careful** increments:

- Until the frame rate starts to drop,
- or until the time it takes to initialize the large number of triangles surpasses your patience (20-30 seconds),
- or, until the application crashes with out-of-memory errors.
- Stop increasing the number of triangles before Windows hangs or crashes. (This is possible - be cautious.)

Report values similar to the previous section:

- The number of leaf nodes in the tree (= #triangles), and the reason you didn't go higher.
- The minimum and maximum depth of the leaf nodes.
- A running average of ray/box intersection calculations per frame.
- **DO NOT** produce and debug drawings of the boxes!

## **What to submit**

Submit to Moodle a single ZIP file containing **SOURCE CODE:** (\*.cpp and \*.h) and the **report.pdf** if you decided to report your results that way.

**Do not include any of the following:**

- Build artifacts (\*.o, \*.pdb, \*.sdf, etc...) ,
- Executable files (\*.exe),
- The lib, Debug, Release, or .vs folders.