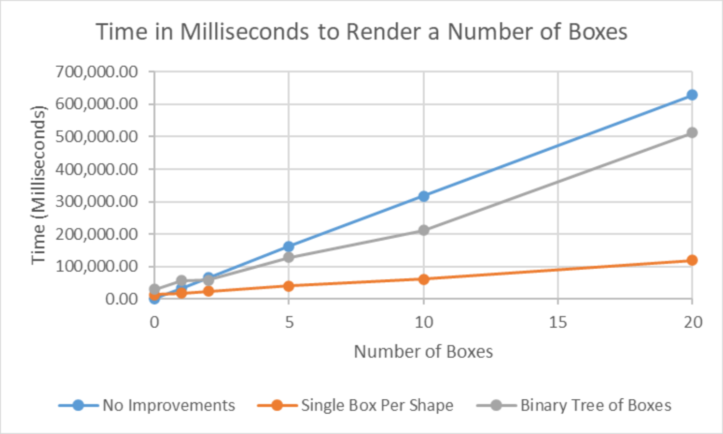
Kyle Rathman
CSCI 441
Lab 9 Writeup



No Improvements

This intersector was the default BruteForceIntersector that was already implemented in the lab. It works by checking every ray (one per pixel) against every surface in the World. Naturally, this gets extremely expensive as more surfaces are introduced. Each box added brings in an additional 12 surfaces to check, which increases the runtime in a linear fashion, to the point where 20 boxes (along with 4 spheres) take around 10.5 minutes.

Single Box Per Shape

This intersector (declared as MySlickIntersector) took in a vector of BoundingBoxes instead of a World. Each bounding box had a World variable, each with a single Shape. The intersector first tests if the ray hits the bounding box, which is a rectangular prism that encloses the shape inside. Since the hit detection for a bounding box is faster than for a set of 12 Triangles, this is faster than checking each surface individually. Each ray tests every bounding box, but only check the Shapes inside if it hits its enclosing bounding box. This is demonstrated in the dramatically lowered rendering time, rendering 20 boxes (and 4 spheres) in just less than 2 minutes.

Binary Tree of Boxes

This intersector (declared as MySlickIntersector2) also took in a vector of BoundingBoxes. This vector was processed to try to improve performance. Starting with a vector similar as was passed to MySlickIntersector, each bounding box was grouped with the closest bounding box inside a larger, encompassing bounding box, recursively until a single all-encompassing bounding box was created that held all shapes in a tree of bounding boxes. Next, if a bounding box is hit, its child boxes are checked next. This happens recursively until all hit boxes have been checked for objects. A single Hit object is passed through the process to keep the closest t-value accurate.

The original intent was to lower the number of boxes that would need to be checked, and while the process takes less time than no improvements on numbers of boxes above two, it still takes dramatically more time than a single box per shape, taking about 8.5 minutes to render 20 boxes (and 4 spheres). This may be due to the packaging process, could have been built to group shapes more intelligently. A binary tree may be too unnecessarily deep, leading to too many bounding boxes that need to be checked before the shape is reached.