Satwiko Wirawan Indrawanto - 6201539
Basar Oguz - 6084990

Assignment 2: Interactive Ray Tracer Documentation

Introduction

This project has been developed with the purpose of achieving interactive frame rates using acceleration structures on the Whitted-style ray tracer we have implemented on Assignment 1. The Bounding Volume Hierarchy (BVH) data structure has been utilized as the acceleration structure to achieve such a speedup in rendering.

Bounding Volume Hierarchy Architecture

The nodes of the BVH are compactly structured, consisting of only 32-bytes. This compact representation of a BVH node consists of a bounding box and indices of left-first child and child count. These compact BVH nodes are then aligned to memory cache lines during BVH construction for increased efficiency during traversal. We have used spatial splits during construction using Surface Area Heuristics and Binned construction principles combined (see Additional Points #2).

The traversal of the BVH is implemented with three different runtime options, the standard one being the standard traversal, where the nodes of the BVH are being traversed in no particular order. Efficient ray-box intersection [5] and ray-triangle intersection (Möller-Trumbore) algorithms have been deployed to maximize the naive BVH traversal mode. Further improvements in BVH traversal have been implemented (see Additional Points #1).

Additional Points

1. We have implemented two different ordered BVH traversal options for primary and secondary rays, excluding the basic standard traversal. These different traversal methods can be switched into at runtime with [T]. The performance of these different traversal methods can be kept track of using the FPS counter. Depth rendering can be toggled with [B] to visualize how deep into the BVH tree these different traversal methods go into.

Implemented traversal methods are:

I) Distance Calculation: Calculates the distance to both child nodes, traverses the nearest child node first. The renderer runs at the highest frame rate using this traversal method. The depth that this mode traverses into is deeper than the Ray Direction Sign method.

II) Ray Direction Sign: Determines the axis for which the child node centroids are furthest apart. Uses the ray direction sign for that axis to determine near and far child nodes. This traversal method has increased frame rate compared to basic traversal, and has the least general depth when traversing into the BVH.

We have implemented a Surface Area Heuristic (SAH) combined with Binning the scene in regular intervals to determine optimal splits and ensure a quality BVH construction. The SAH assumes that the cost of a split is $Aleft * Nleft + Aright * Nright$ (where A is the area and N is the number of primitives).

SAH improves the performance significantly where small polygons are densely populated, however, fails to provide speedups for scenes with large polygons that take up much of the screen. We have combined Binning with SAH, to make sure we split even if there is a large polygon taking up a large part of the screen. We have used 7 equally distributed bins across 3 dimensions, combined with SAH cost calculations to achieve a speedup in every scene configuration.

3.. A top level BVH framework, consisting of BVHs attached to separate rigid objects has also been implemented, to enable translational animation of rigid bodies without the need to rebuild entire BVHs. A simple median split was used during the construction of the top level BVH. Implementation is still experimental and only renders the first object loaded. Can be enabled with **ENABLETOPBVH** in game.h.

Results & Performance

An alternate rendering method has been implemented to work with loaded .OBJ file materials, can be disabled with **ALTERNATERENDERMODE** in game.h.

We tested the performance with a 3d object loaded with 34 thousand triangles and several point light sources, using the same camera setup on every test. BVH can be disabled with **ENABLEBVH** in game.h. (Scene 3)

BVH disabled: Takes forever to render the first frame
BVH enabled: 17 frames per second

So instead, testing a small 3d cube. (Scene 0)

BVH disabled: 40 frames per second
BVH enabled: 120 frames per second

Performance increases at least 3 times without using BVH.

We also measured SAH BVH construction time with and without Binning enabled using a complex 3d model with 54 thousand triangles. Disable binning with **ENABLEBINNING** in game.h.. (Scene 4)

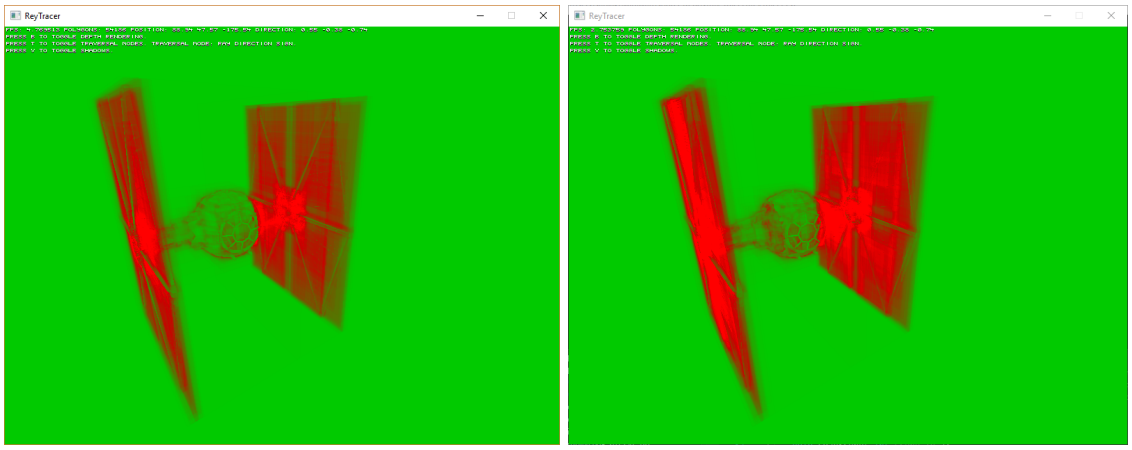
Binning disabled: 249.792 seconds (roughly 4 minutes), rendering at 4.7 fps
Binning enabled: 0.328 seconds, rendering at 2.7 fps

Although, by disabling binning, we have better quality BVHs and thus improving render time.

With 3 traversal modes and dept rendering, we can also check how deep the traverse method must go to be able to hit a primitive. A 3d model with 8 thousand triangles was tested, by enabling depth mode rendering (Press B), and toggling between traversal mode (Press T). (Scene 2)

Standard traversal: 22 frames per second, most depth
Distance Calculation: 33 frames per second, currently the most optimal setting
Ray Direction Sign: 28 frames per second, least depth

Scene 4 with binning disabled and enabled.



References

[1] On fast Construction of SAH-based Bounding Volume Hierarchies, Wald, 2007

[2] Spatial Splits in Bounding Volume Hierarchies, Stich et al., 2009

[3] Large Ray Packets for Real-time Whitted Ray Tracing, Overbeck et al., 2008

[4] Heuristics for Ray Tracing using Space Subdivision, MacDonald & Booth, 1990

[5] An Efficient and Robust Ray–Box Intersection Algorithm. Amy Williams et al. 2004.