Computer Graphics: Spatial Data Structure

November 30, 2023

1. Authors

1. Hun Rim
2. Georgy Batyrev

2. Description of implementation

1. AABB structure

   In the first implementation of structures, AABB structure is introduced. Before, the program was checking if the ray intersected with the plane of the triangle for every triangle in the objects vector before checking if the ray landed inside the triangle, and this was done for every ray. However, with the AABB structure, this process of checking ray's intersection with the plane is only carried out if the ray first intersects with a large imaginary box containing all the triangles. Hence, it vastly reduces the computation required and provides incredible efficiency. From personal experience, the rendering time of small objects went from 1000+ seconds to approximately 300 seconds.

   Please run **main.cpp** to view the visualization using AABB structure.

2. BVH structure (Bounding Volume Hierarchy)

   It is actually an improvement built based on the previous implementation. Instead of just making one large box, the box is recursively cut in the middle of one of the axises (different axis everytime) and it would create a tree of boxes with multiple layers (each layer with double the number of boxes in the previous layer). Then only the boxes in the lowest layer and intersect with the ray will be accessed. To reduce the complexity, the boxes are accessed by traversing the tree structure of boxes. This method significantly increases the efficiency of visualization in comparison to the previous approach. The objects which took 300 seconds to render

takes less than a second to render with this structure.

Please run **part2Large.cpp** to view the BVH structured rendering of large dataset and **part2.cpp** for smaller objects.

3. Timings

Struct and size	Maximum Time	Minimum Time	Mean Time	Median Time
BVH large	7.9140	7.0577	7.3113	7.4555
BVH small	0.9294	0.7561	0.8094	0.8045
AABB small	29.6033	26.7104	27.8908	27.7556

The real rendering time is approximately 10% less than the values given on the table. It is because the time is measured with a bash script *timer.sh* which also includes the time taken to write the image file. The actual copy of the table can be viewed inside the **result.txt**.

4. Graph

The graph is generated using a python script (*analysis.py*) called through the bash script mentioned before. The bash script runs the code 10 times each for each instance and record their starting and ending time. The time data is passed on to the *analysis.py* script which is manipulated to generate the table above and also the graph below.

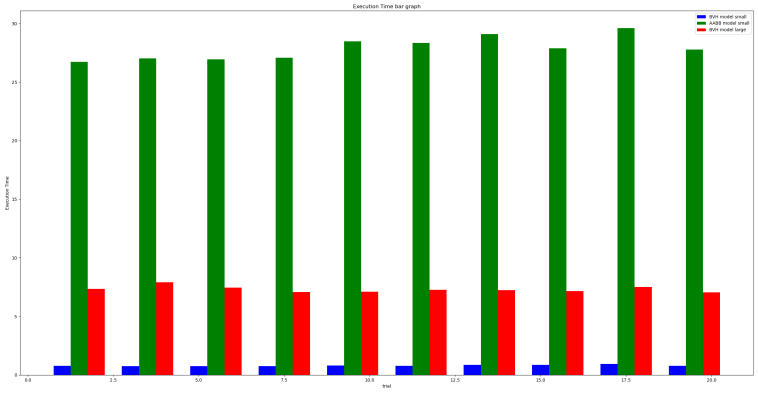

Figure 1: Bar graph of execution time (blue = BVH small), (green = AABB small), (red = BVH large)

5. Problems and solutions

1. The **AABB visualization** of large dataset is astronomically time consuming, especially, in comparison to other instances. It would also lead to generation of an unreadable graph. To avoid this problem, this instance wasn't tested.

2. The time taken to render an image has a quite some marginal error, which is natural. To reduce the effect of this problem. The images were rendered multiple times (preferably the rendering should have been executed more times and some data in the beginning should have been disregard for fair experiment) so their properties like mean, minimum, maximum, and median execution time could provide more precise and stable data.

3. In terms of implementation, BVH implementation could have been improved by using a more advanced algorithm which provides a more even spread of triangles among the boxes. Current algorithm provides even spread of volume and there may be imbalance of triangles among the boxes in certain cases, depending on the positioning of objects. Instead of cutting the boxes into two in the middle, cutting it more efficiently by using the density of triangles as the deciding factor will definitely improve the performance in my opinion.