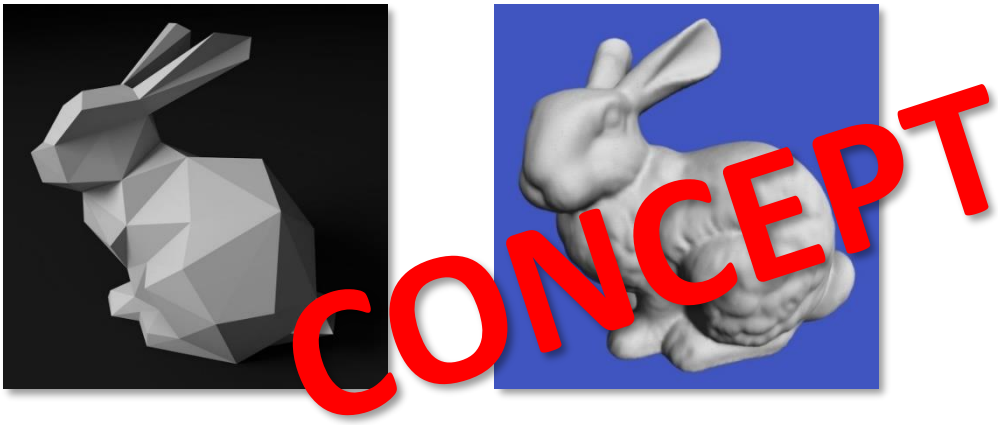
Advanced Graphics 2022/2023 – Assignment 2



Introduction

The goal of this assignment is to extend the framework you built in assignment 1 with a *Bounding Volume Hierarchy (BVH),* to enable rendering complex scenes. A successful introduction of the BVH in your framework greatly reduces the cost of ray / scene intersection, and potentially enables real-time rendering.

Borrowing

Compared to earlier years, there is one major: the ADVGR lecturer just wrote 10 articles on the topic, complete with source code. A quick overview, for your convenience:

1. Basics: data structure, simple construction and ray traversal
2. Faster rays: the Surface Area Heuristic and ordered traversal
3. Faster construction: binned BVH building
4. BVHs for animated models: refitting and rebuilding
5. BVHs for animated scenes: the top-level BVH
6. TLAS & BLAS part 2 – all together now
7. Consolidating, part 1: ray tracing textured and shaded meshes
8. Consolidating, part 2: Whitted-style ray tracing
9. OpenCL: GPU ray tracing, part 1
10. OpenCL: GPU ray tracing, part 2 (series finale)

The accompanying source code, in 10 convenient incremental steps, can be found on Github.

Assignment, part 1: Attach Wheel

Even without the above resource it is obviously possible to use existing work. This is allowed and should in fact be encouraged: this particular wheel already has been invented; it is your task to attach it and – perhaps – to improve on it. For this reason, the basic assignment is as follows:

Add BVH traversal to the ray tracing template you created for assignment 1. Make sure it works with the existing code base and functionality. Concretely:

- Make the BVH accelerate your Whitted-style ray tracer and Kajiya path tracer;
- Make sure the BVH handles spheres, triangles and planes, as well as diffuse and specular materials.

Note that handling spheres is not discussed in the 10 articles. It should however not be very hard to add.

For 50% of the points (5), you need to implement one additional feature that is not discussed in the articles:

- Distinguish ‘occlusion’ rays and ‘nearest hit’ rays. Make sure this is not just an interface matter; a correct implementation exploits the difference during traversal.

Beyond a Five

To get more out of the BVH and score beyond a 5, you may take on one of the following challenges.

- Upgrade the BVH to a QBVH, where each node has not two but four children. A successful implementation is provably faster than the original BVH and scores up to 2 extra points.
- Traverse the BVH with more than a single ray at a time. A successful implementation is substantially faster for certain classes of rays and scores up to 2 extra points.
- It can be beneficial to relax the 'object partition' nature of a BVH for some polygons, cutting them in half for the greater good. Various ways to do this exist. A successful implementation yields a substantial improvement in traversal performance and yields up to 4 extra points.
- A generated tree may benefit from various 'postprocessing' operations. Improving the tree this way and proving that the method is effective may yield up to 2 extra points.
- Experimenting with other heuristics than the standard 'SAH' may yield op to 1 extra point.
- An insightful analysis of the behavior of (the performance of) your implementation may yield up to 1 extra point. Data to look for may include node count, summed node area, traversal steps, intersected primitive count and tree depth.
- Build the BVH on the GPU for 4 points.

Note the deliberate 'up to' formulation: in most or all cases, the requested functionality can be handled in a basic way, which will yield a smaller bonus, or a more advanced way, which is more challenging.

Also note that in all cases references are missing. Feel free to ask around and share information on Teams.

Language Notes

This assignment may be executed in a programming language of choice. Support on the implementation side will be mostly limited to C++ and C# however, and performance is expected to be optimal for C++ code. Choice of programming language will not affect your grade.

Practical Details

The deadline for this assignment is **Friday, December 23, 17:00**. As with the first assignment, an extended deadline is available, at a 1pt penalty. The materials to submit are:

- your project, including sources and build instructions (if these are not obvious);
- a brief report, detailing implemented functionality, division of work, references and other information relevant to grading your submission.

You may work on this assignment alone, or with one other student. Feel free to discuss practical details on Slack/Discord/Teams/ICQ/in person.

May the Light be with you,

Jacco.