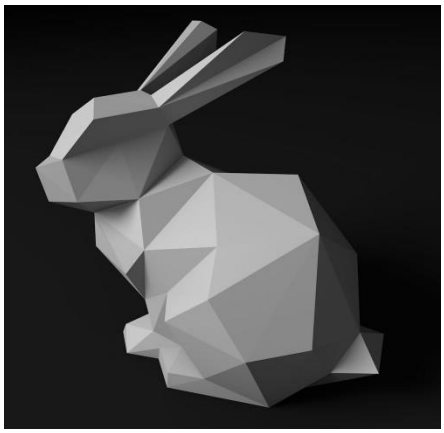
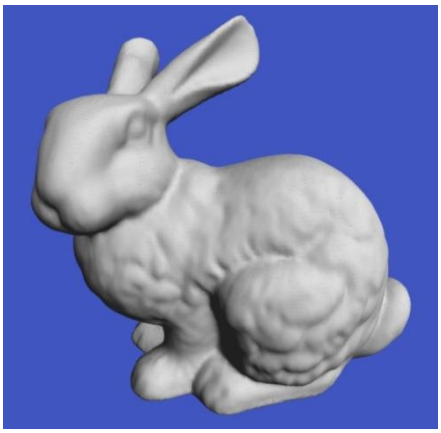
Advanced Graphics 2020/2021 – Assignment 2





Introduction

The goal of this assignment is to extend the framework you built in assignment 1. This can be done in two main directions:

1. **Acceleration structure construction and traversal**. A successful implementation greatly reduces the cost of ray / scene intersection, and potentially enables real-time rendering.
2. **Path tracing.** A successful implementation provides a full evaluation of the rendering equation, which yields photo-realistic images.

Option 1: BVH construction

The recommended acceleration structure for this assignment is the Bounding Volume Hierarchy (see "Freedom" below). An efficient BVH uses a compact node structure (32 bytes), aligned to cache lines to reduce memory traffic during traversal (see "Language Notes" below). Efficient construction is achieved by using a binning builder, as described by Wald [1].

BVH traversal

Traversing the BVH is a straight-forward process. A naïve implementation can be optimized using a custom stack rather than recursion, by using efficient ray/AABB and ray/triangle intersection code, and by traversing the BVH front-to-back (combined with 'early out').

Performance can be enhanced further using basic packet traversal (not discussed in the lectures). This can be implemented efficiently using the algorithms proposed by Overbeck et al. [3]. In C++, these algorithms benefit from (but do not require) SIMD / AVX; this may also be possible in C#. Substantial gains can be achieved without SIMD.

BVHs and Lighthouse

If you are working in Lighthouse, you have several options for handling geometry.

1. Easiest: import a single mesh (.obj file format) and illuminate it with point lights. This will send one mesh to via the RenderSystem to your Core ('SetGeometry'). Build a BVH over this geometry and use this for ray/scene intersection. Assume the geometry is static.
2. Easy: have multiple meshes (multiple .objs, or a gltf file). This will trigger multiple calls to SetGeometry. Assume all geometry arrives before the first call to your Render method. Build the BVH over all received geometry during the first Render call (and *only* during the first render call). Assume the geometry is static.

Hard: build a BVH for each mesh you receive via SetGeometry. Using the matrices received via SetInstance, build a top-level BVH over the per-mesh BVHs. Update the top-level BVH each frame. You may also postpone this to the final assignment, where this is one of the options.

BVHs and the Voxel Template

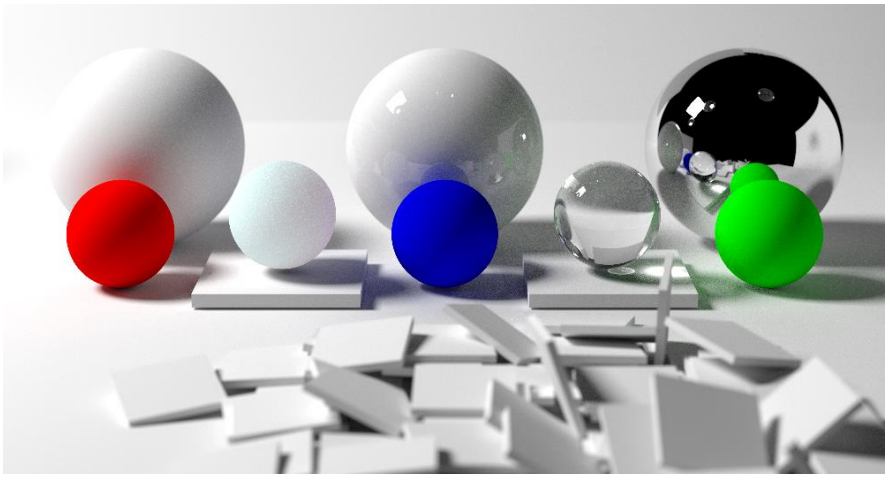
The voxel template is already using an acceleration structure. It may thus seem logical to skip the BVH option when using this framework. There are possibilities though. Consider Minecraft: the game uses voxels for rendering the world, but triangle meshes for everything else. A successful BVH implementation in the voxel world could thus start with a single mesh, represented by a BVH, added to the world. A more advanced implementation could include multiple meshes, instancing, rigid motion, and a top-level BVH.

Path Tracing

A path tracer evaluates the Rendering Equation [5] using Monte Carlo / numerical integration. This algorithm enables exploration of paths that Whitted-style ray tracing ignores, such as paths via diffuse surfaces, which yields indirect light and caustics.

For this assignment, performance is not a requirement. You may work with a simple scene to keep things responsive, but final performance will not play a role in assessment of your code.

If you decide to write a path tracer, you must provide a *correct* minimalistic / brute force renderer (without NEE, MIS, RR etc.) and a mechanism to compare more advanced algorithms to this ground truth. Correctness is key.



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.

Freedom

Most of the acceleration structure concepts discussed in the lectures apply to BVHs and kD-trees. You are free to use either data structure. You may also propose an alternative data structure (a 'brickmap' as in 5Faces perhaps?), if you wish to explore the specifics of that option. Please discuss your plans with me in advance.

You also have a certain level of freedom in the functionality you actually implement. Generally speaking, you can opt for excellence in BVH quality, construction speed or traversal speed to receive a high grade for this assignment.

A bit less freedom is available if you work on a path tracer. The path tracer must first of all be accurate / correct. Beyond that point, some extensions are possible; all of these affect the efficiency of the code. Be careful not to sacrifice accuracy of the renderer.

Practical Details

The deadline for this assignment is **Thursday, December 23, 17:00**. As usual, 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. You are not supposed to share complete ray tracers there, but if everyone uses the same optimal ray/AABB test, that would be considered 'research'.

Tasks & Grading

A passing grade (6) for the **acceleration structure version** of the assignment requires:

- implementing a correct BVH over a large (>10k) set of input triangles or other primitives (2 pts). The resulting BVH should be (at least) reasonably useful: it may use midpoint splits, but it may not store all primitives in the root node.
- implementing ray/scene intersection using this BVH (1 pt), achieving a performance improvement over brute force scene intersections in line with expectations (1 pt);
- correctly build the BVH using the Surface Area Heuristic (1 pt) and binning (1 pt) (see [1,4]).

Once you have implemented the minimum functionality, you can score additional points with the following features:

1. Construct a BVH for dynamic scenery using specialized builders for various types of animation. Add a top-level BVH to combine the resulting sub-BVHs, and adapt your traversal code to handle rigid motion. Provide a demo to prove that your BVH handles animated scenes (2 pts).
2. Construct the BVH for a 5M triangle scene in less than 1 second (still using SAH; 1 pt).
3. Implement packet traversal for primary and secondary rays [3] (2 pts) (warning: only helps Whitted).
4. Construct a 4-way BVH by collapsing a 2-way BVH, and traverse this structure. The resulting traversal speed must be an improvement over 2-way BVH traversal (1 pt) (good for Kajiya, packets are better for Whitted).
5. Render a 1B poly scene in 5 seconds or less for 5 extra points.

A passing grade (6) for the path tracing version of the assignment requires:

- A correct path tracer, which can render scenes with area lights, diffuse materials, pure specular materials and dielectrics, with support for Beer's law, using a full evaluation of the rendering equation (4 pts).
- A mechanism to measure and display overall energy of a frame, to be used for comparisons against alternative algorithms (1 pt).
- A converging scheme ('accumulator') for a stationary camera, which resets when the camera moves (1 pt).

Once you have implemented the minimum functionality, you can score additional points with the following features:

1. Next Event Estimation (1 pt), Russian Roulette (0.5 pts), importance sampling of the BRDF (0.5 pts), importance sampling of lights in case multiple lights are present (1 pt). In all cases, a comparison against ground truth is required. In all cases, correctness is required.
2. Implement depth of field (0.5pts) and / or motion blur (1 pt).
3. Better random sampling using blue noise (1 pt). As usual, compare against ground truth.
4. Correct Multiple Importance Sampling (1.5 pts), with a comparison against ground truth.
5. Implement spectral rendering (warning: hard – bonus: 3 pts if successful).

Additional bonuses may be discussed; contact me if you have plans. Note that smaller features such as an HDR skydome as well as features that were part of assignment 1, such as post processing, do not yield bonus points this time.

Purpose

We will use the results of this assignment in the third assignment. Consider carefully which type of assignment fits your interests best: real-time / animation, or image fidelity. Ideally, this assignment is your final preparation for the final one.

May the Light be with you,

- Jacco.

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] The Rendering Equation, Kajiya, 1986.

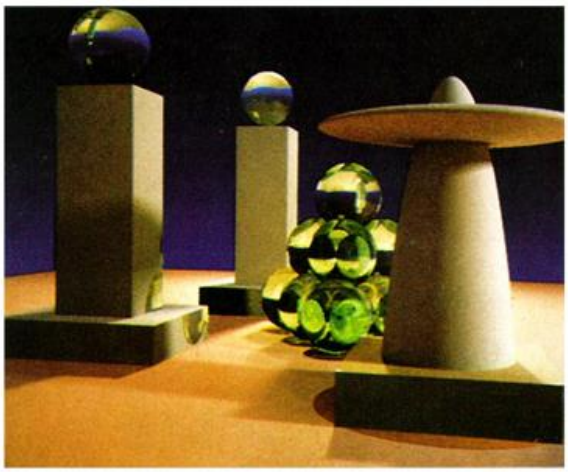


Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.