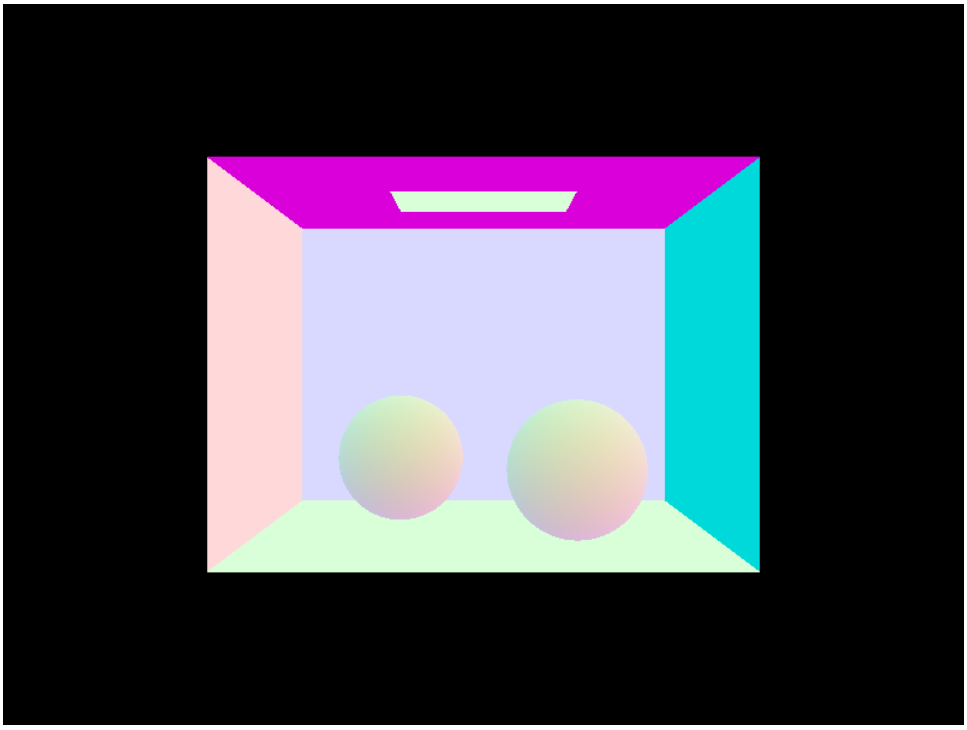
Assignment 3: PathTracer

Khadijah Flowers

In this assignment, we explore the techniques of PathTracing. We start with shooting rays through the scene, sampling color through triangle intersection and barycentric coordinates, and assigning pixel color based on this ray calculation. Next, we move onto BVH construction. Here, to make ray/primitive intersection calculations easier, we divide the scene into a tree where the nodes in the tree hold primitives. We only check a ray's intersection with primitives if it collides with the bounding box containing it. This process saves us a lot of time. After that, we focus on types of shading and illumination for the scene. Here, we'll end up shooting a ray to a primitive and from the primitive to a lighting source. Finally, Adaptive Sampling lets us use a different sampling method for different parts of the scene to get the best rendering result.

Part 1: Ray Generation and Intersection

Ray generation and primitive intersection is a technique used to color a scene based on a viewpoint. From the point of view of the player or viewer's eye, rays are shot into the scene, through pixels, and if the ray collides with a primitive, we color the pixel with the primitive's color at this point. The triangle intersection algorithm is the part of the ray generation and primitive intersection algorithm that decides what the color of the pixel whould be at the intersection point. We shoot a ray and if the ray intersects with a primitive, we find the triangle on the primitive where the ray hit and calculate the barycentric coordinates. We then use them to determine the color of the hit point and this becomes the color of the pixel where the ray was shot.



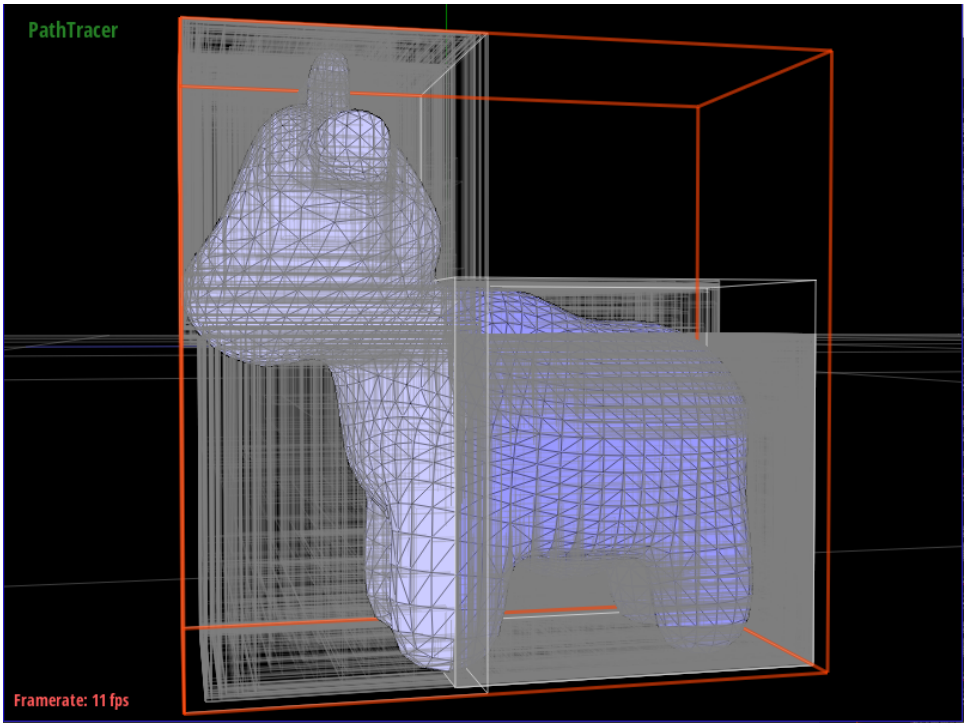
Room with colorful balls.

Part 2: Bounding Volume Hierarchy

For my BVH construction algorithm, I split the primitives into two sets, left and right, and choose the largest axis in the current box's extent. Based on whether or not the primitive's centroid's corresponding axis is less than or greater than half of this point, I puth them in the left and right lists. I then recursively call the function and repeat this process until there are few enough primitives to put in a leaf node. I sometimes ran into an issue where one list had no elements, so to fix this, I'd change the extent we're looking at. And if that didn't work, I'd just iterate through all of the primitives and assign the even numbered ones to one list and odd numbered ones to the other. For my intersection algorithm, check to see if the curent node is a leaf. If not, I skip it. Otherwise, I check to see if the ray intersects with the primitive. If it does, I take the primitive's dimensions and check to see if the ray's current min and max values are less than the ray's current min and max values, then I reassign the ray's min and max values to reflect this as being the closest point.

Teapot Rendering.

Cow Rendering.



Cow BVH.

With rendering scene accelearation, images with thousands of primitives like the cow are quickly generated. Going from checking one ray against thousands of primitives to one with just a few makes a significant difference. Now almost regardless of the number

Part 3: Direct Illumination

The two forms of direct lighting that I implemented are direct lighting with uniform sampling and direct lighting with importance sampling. Direct lighting with uniform sampling was done by shooting rays in random directions and sampling the light, no matter how intense, summing them, and averaging them out into the final light. Direct lighting with importance sampling was implemented by shooting rays, and giving more intense lights a higher weight when summing, and then taking the average, returning this as the final light.

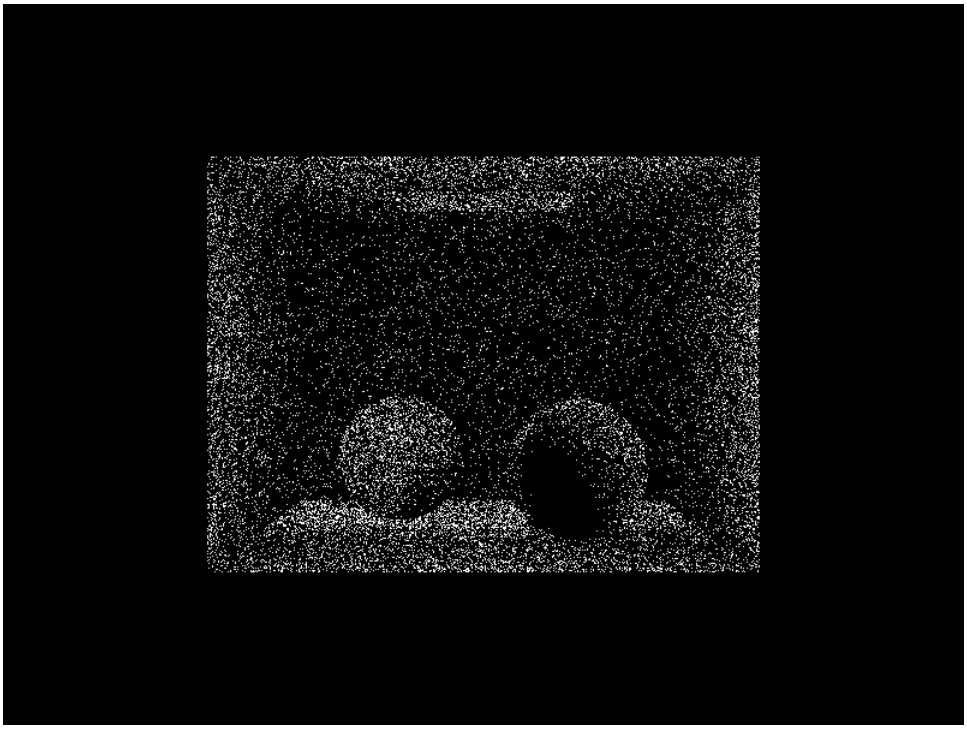


Image with hemisphere uniform sampling, 64-pixel sampling.

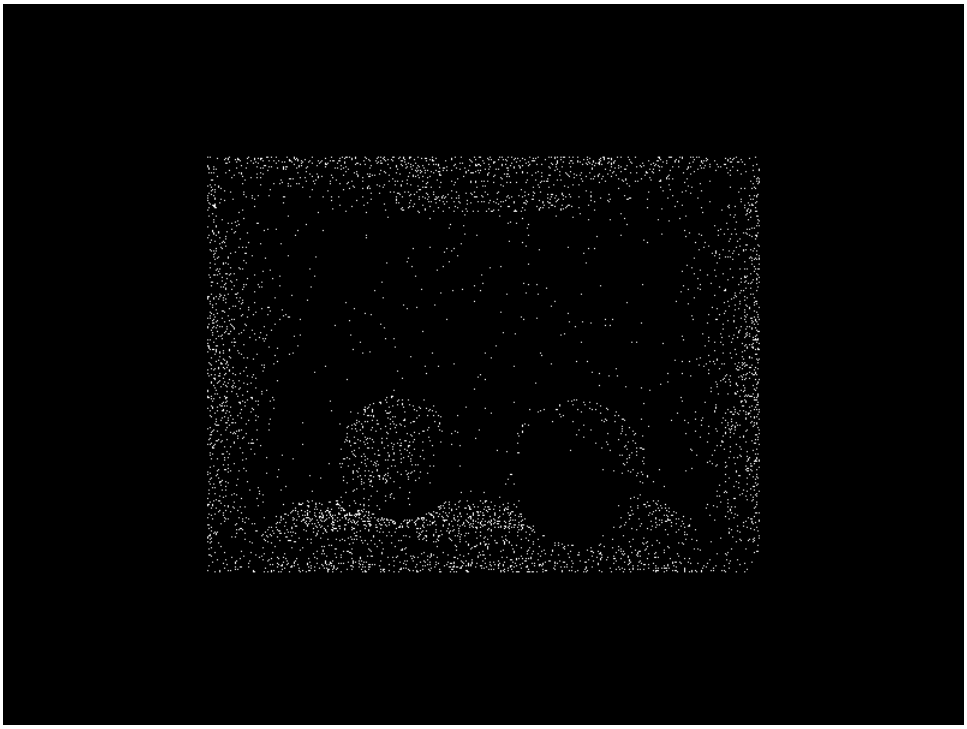


Image with hemisphere uniform sampling, 16-pixel sampling.

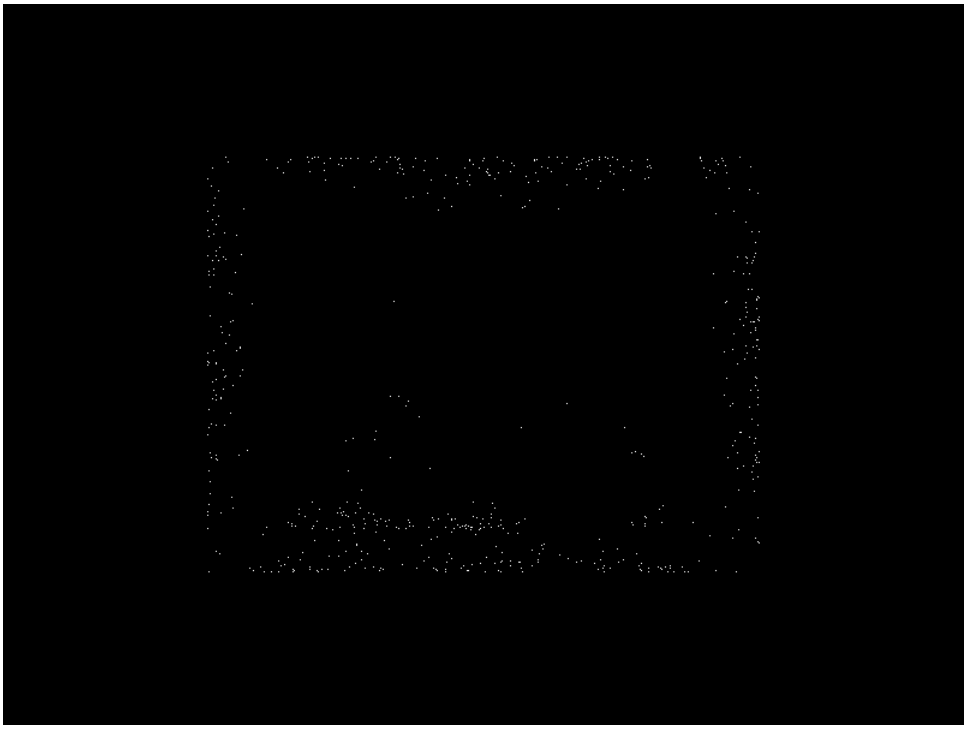


Image with hemisphere uniform sampling, 4-pixel sampling.

Image with hemisphere uniform sampling, 1-pixel sampling.

The image rendered with uniform lighting leaves much more noise in the image than rendering with importance sampling. Importance sampling weighs every light source equally. The image gets better in importance sampling because the intensity from brighter, stronger light sources is weighed heavier with a pdf based on its intensity.

PART 4 AND 5 NOT COMPLETED.