Sean Higgins

March 18, 2017

CS 500 – Ray Tracing

Prof. Herron

CS 500 Ray Tracing – Proj. 2 Report

Overview:

Compared to the previous project and its massive share of setbacks, this project to implement path tracing was comparatively straightforward and simple—and, almost more importantly, simple to extend further for the future projects.

Overall, the additions to the project are focused primarily on implementing proper path tracing and accumulating various color outputs over a number of passes to produce the final image. Tracing out the paths naively is relatively simple, but it is excessively, exorbitantly slow process. Thousands, tens of thousands of passes must be run through in order to get a proper output with just random guessing, which is far from ideal. This slowdown is why we use MIS weights and an additional explicit light test alongside the randomly chosen ray—both techniques help immensely in increasing the rate of convergence for the image.

The initial version of the algorithm only goes through and bounces rays randomly off of objects in an attempt to hit a light in the scene. The chance to continue looping until a light is hit, as well as the direction in which the ray bounces, is chosen based on various probabilities. Through the Monte Carlo integration technique, we can theoretically choose any way of choosing the direction of the new ray, so long as it allows for any direction and stays consistent throughout passes. Ultimately, these passes are aggregated and averaged by the number of passes in order to write out the output image. However, the number of passes required to achieve a proper image is exorbitantly long without any further improvements, as it is extremely likely that rays chosen at random will not hit the light. In order to expedite this process, we add in two additional steps: an explicit light connection test, and multiple importance sampling (MIS) weights. In the explicit light connection test, whenever a new intersection point is reached, we first test to see if the ray intersects a random point on a random light—if there’s an open path to a particular light—then use the color of that light and intersection point, if it does intersect a light, speeding up . Similarly, the MIS weights are calculated each pass, taking into account the ray directions that aren’t chosen, then applied and multiplied to the cumulative color at each pass, pushing it towards the average we expect.

The final output image after 5005 passes through the loop, with all major lighting features accounted for—save reflection and refraction.

