Tommy Walton

CS 500

Herron

3/12/2015

Project 1: Ray-casting

### Project Description

This project is an implementation of a basic rendering loop using ray-casting. A scene is comprised of spheres, cylinders, slabs, and meshes formed out of triangles. For each pixel in the generated image, the ray-caster calculates a ray from the position of the camera through the center of the pixel. The intersection algorithm for every geometric object in the scene is then used so see if the ray intersects with any objects. If the ray intersects more than one object, the intersection that is closest to the camera is stored, and all other objects behind the closest one are occluded.

For each intersection, the surface properties of the object are stored, including the normal, diffuse color, specular color, and roughness. The direction to each light in the scene (only one light was used in the test scene) is then calculated and used along with the surface properties in a basic lighting model. Each light’s contribution is summed into the final color of the pixel. No consideration is taken to whether or not it is blocked by another object.

### Screenshots



### Implementation

The majority of the relevant code is in Shape.cpp. Each shape has an Intersect function that returns true if there is an intersection, false otherwise. It takes a ray as input, as well as an intersection struct that, in the event of an intersection, stores the position, normal, and t-value of the intersection as well as a pointer to the object that can be used to get the material. The shapes that are implemented are sphere, slab, box, triangle, and cylinder.

Camera.h implements a simple camera which stores the position and orientation of the camera, and also calculates the ray going through each pixel. Raytrace.cpp has the main ray-casting algorithm and the code for reading in the scene, which is largely unchanged from the original framework.

### Build and Run Instructions

The project was built using Visual Studio 2013. I modified the format of the scenetest.scn slightly. The line for defining a light takes the same arguments as a brdf. This was done to enable re-using the same code for creating a regular object and a light. Otherwise, the project runs just like the original framework.