Assignment 5: Ray Tracing

In this assignment, you will implement a ray tracer. Once again, following the steps laid out here is designed to make the assignment easier. I recommend that you try to do the assignment in the order presented here, confirming along the way that each step works as expected before going to the next part.

1. The first step is to get a basic ray tracer going on a simple object. I recommend that your first ray tracer have the following properties:

   (a) Define a single simple geometric object. I recommend a cube. Give each triangle in the cube a different color. This will help you debug.

   (b) You do not need to rotate the object or move it. You only have to render it once. You can rotate it if you want to, however. This may help you debug things.

   (c) To start with, you can use the most basic “lighting” scheme, which is simply to color a face with the basic triangle color. Later, you will need to add more complicated lighting models that involve the computation of normal vectors and the incorporation of light sources.

   (d) To start, use just a single ray for each pixel, and a single point for the light source.

   (e) To start, do not implement shadows.

To implement the basic ray tracer, you should use the functions I described in class,

\[
\text{void ray\_trace()}
\]

\[
\text{bool ray\_hits\_any\_triangle(vector3 e, vector3 d, ...)}
\]

\[
\text{bool ray\_triangle\_intersect(vector3 e, vector3 d, triangle T, ...)}
\]

I will make your life a lot easier by giving you the function Alex wrote for computing ray-triangle intersections. This function, along with a struct definition that he uses to return some values, can be found on the course web page just after the link to this assignment document.

2. After you get the basic ray tracer working, the next step should be to add Gouraud shading. That is, you should add code to made the brightness of a face depend upon the lighting source and the normal to the surface at the current point. This should be quite easy if you have done assignment 4.

3. Next, I would like you to add a new kind of object to your renderings. In particular, I want you to add a sphere. For a sphere, you will need a new kind of intersection routine. I went over this in class, but if you don’t remember how to do it, it is in the book. You will also need a routine that computes the normal to a sphere at a given surface point. I also went over that in class. Note that you will need to slightly replace your routine \( \text{ray\_hits\_any\_triangle} \) with \( \text{ray\_hits\_any\_shape} \). This new routine should check all of the triangles and all of the spheres in your model. A good way to implement this in C++ is to have a primitive shape structure called \text{shape} and to derive two classes from this called \text{triangle} and \text{sphere}. Then each of these derived classes can have a separate ray intersection routine. You do not have to implement it in this way; it’s just a suggestion. If you have not done inheritance in C++, it will probably take you a little longer to figure this out using derived classes.

4. After you get the sphere working, add shadows to your code. Remember that your ray intersection routine should be careful not to detect an intersection of a shadow ray with the surface it is STARTING from. In other words, look for intersections with \( t >= \epsilon \) in the ray equation, not \( t >= 0 \), as I mentioned in class. This is also discussed in the book if you don’t understand. You might want to add a few cubes in different places or a few spheres, to make sure they cast shadows on each other that you can see.
5. For the last piece of the assignment, you have a choice among many different options. You can choose any ONE of these options you want. They are

(a) Add a surface with a totally reflective surface to your model. This could be a chrome sphere, or a mirror made of one or more triangles. The rendering should make it obvious that the surface is mirrored. In other words, you should be able to see some of the other non-reflective surfaces in the mirror when the object is rendered.

(b) Add a very large plane under all of your objects. This can be done with just one or two giant triangles. Use texture mapping (whichever type you like) to put a pattern on the plane, such as a checkerboard pattern or a brick pattern.

(c) Embed your rendering in an environment using an environment map, like a cube map. You can get your cube map from anywhere you like. There are many available on the web, or you can try to make your own with a personal camera.

(d) Add an object made of glass to your rendering. I did NOT discuss how to do this in class, but it is covered in the book.

(e) Add multiple rays per pixel to your rendering. Show a before and after example of your rendering to show that it makes a different in the quality of the rendering. You may also want to add multiple points per light source, but this is not required.

This assignment will be worth 1.5 times the normal value. Each part will be worth 20% of the grade on the assignment.

**EXTRA CREDIT**

If you do MORE THAN ONE piece of part 5, you will get extra credit. Here is how the extra credit will break down:

1 extra piece (2 total): 10 percentage points.
2 extra pieces (3 total): 15 percentage points.
3 extra pieces (4 total): 18 percentage points.
4 extra pieces (5 total): 20 percentage points.

If you implement a reflective sphere and an environment map, you will get supercool results!