Project Proposal

Title: Ray Tracer

Name: William McDonald

Student ID: 20418145

User ID: wmcdonal

Final Project: Ray Tracer

Purpose :

Apply advanced raytracing effects to efficiently render a more photorealistic scene.

Statement :

The main goal of this project is to implement advanced and interesting ray tracing effects to *efficiently* create a realistic final scene.

The final scene will center around a magnifying glass looking at a small complex object and a mirror. There will be additional novel objects around the magnifying glass to provide character to the scene. The scene will take place on a table in a well lit room.

In order to render this scene, reflection and refraction are the two most important features. Texture mapping will make the table look like a table. Phong shading for meshes will help the objects look smooth and realistic with fewer faces.

Additional primitives including a torus will be added for the surrounding objects, and soft shadows will add to the realism of the scene.

This is interesting and challenging because many of the objectives are complex algorithms that included nontrivial changes to the workings of the ray tracer. I am especially excited to be able to play with refractions and reflections and their interactions.

Technical Outline :

Additional basic primitives will be implemented to get smoother objects for rendering: the torus and cylinder. Torus-ray intersection can be computed using the given quartic roots solver as described by Cychosz in [1] or by Skala in [2]. A torus can be specified by giving a tube radius - radius of the torus is assumed to be 1 (it can later be scaled).

Cylinder-ray intersections are far simpler and are described in the course notes in section 18.1 on page 122.

Specular reflection has been described in class and can be implemented by recursively casting a reflected ray from the intersection point with a specular object up to a pre-determined depth. Each object can have a "mirror" coefficient added to determine how reflective it is.

Refraction and transparency are also implemented by recursively casting a ray from the intersection point with a transparent object. Each material will have 3 alpha values added: one for each of the RGB colours. Refraction is done by modifying the direction of the newly casted ray, based on a coefficient defined for each material.

Texture mapping onto spheres and planes will be implemented according to the details described in class. Both using a bitmap and using a basic function (e.g. a grid) will be supported, as the implementation is nearly identical.

Adaptive supersampling will be implemented by casting rays to the corners of each pixel. If the corners vary too much in colour then the pixel will be divided into four sub-pixels, and this will continue recursively. The final colours of all sub-pixels will be averaged (weighted according to the size of the sub-pixel) to produce the final pixel colour.

Uniform grid acceleration will be added by dividing the world into a grid of cubes of uniform size. Each cube will have a set of objects that intersect it. When casting a ray, only objects that intersect cubes that the ray passes through will be checked for ray-object intersections. This should result in a large performance gain for scenes involving many small objects.

In order to implement soft shadows, we must first implement area lights. Square area lights will be used in this implementation, so each light will have a direction (normal) and size. Additional interesting light shapes are possible (spheres, cubes, circles, etc) but will be left out to focus on soft shadows.

Instead of casting a single shadow ray, multiple shadow rays are cast, each going to a random point on the light source. Shadows from these rays are then averaged to determine the true shadow.

Proper Phong shading for meshes will be added by computing a normal for each vertex in a mesh and then interpolating that normal across the face as described both in class and by Phong in [3].

Glossy reflection can be implemented by casting multiple rays in random directions off of a reflective surface and averaging the results[4]. The rays can be cast in directions chosen according to a particular distribution, such as a simple uniform distribution or a cosine weighted distribution. Some distributions may result in significantly better images than others. The cosine weighted distribution will likely be chosen, as it is recommended in [4].

To avoid an exponential increase in runtime, glossy reflections may be limited to the initial reflective ray only, and not be used at further depths. At the very least, the number of reflective rays used will decrease significantly with each level of recursion.

Objectives that I want to implement but did not include for fear of overreaching are photon mapping (for caustics), depth of view (aperture at eye), CSG modelling and octree acceleration.

Bibliography :

1. Cychosz, J. M.: "Intersecting a Ray With an Elliptical Torus." Graphic Gems II, 1991, p. 251-265.
2. Skala, V.: "Line-Torus Intersection for Ray Tracing: Alternative Formulation". WSEAS Transactions on Computers, Issue 7, Volume 12, 2013, p. 288-297.
3. Phong, B. T.: "Illumination for Computer Generated Pictures". Communications of the ACM Number 6, Volume 18, 1975, p. 311-317.
4. Stuffern, K.: "Ray Tracing from the Ground Up". Chapter 25, p. 529-542.

Objectives:

Full UserID: wmcdonal **Student ID: 20418145**

___ 1: Additional primitives have been implemented: Cylinder and torus.

___ 2: Specular reflection has been added.

___ 3: Refraction and transparency are implemented.

___ 4: Texture mapping for planes and spheres: functions and bitmaps.

___ 5: Adaptive supersampling is implemented.

___ 6: Uniform grid acceleration is implemented and speedup data gathered.

___ 7: Soft shadows (and area lights) have been implemented.

___ 8: Phong shading with normal interpolation.

___ 9: Glossy reflection.

___ 10: Unique final scene.

A4 extra objective: Supersampling (not adaptive).