Final Project Proposal: Extended Raytracer

Date: Feb 28, 2008

Name: Chi Zhang

Userid: c23zhang

ID: 20126054

Section: 001

Professor: Bill Cowan
Super Candies - Extended Raytracer

1. Purpose

Computer graphics is one of the few courses that I really enjoy during my time in Waterloo. In particular, I enjoy the sense of achievement after creating a visually pleasant image from scratch. The final project of CS488 gives me the opportunity to render an image of my own choice. I am taking the opportunity to demonstrate what I have learned in the course.

2. Topics

- Enhanced ray tracing features

- Use Fine Arts concepts to create visually attractive final scene

- Creation of animation by generating images at each time intervals

3. Statement

The basic elements in my scene will be the “super candies”. The “candies” are modeled with 3D ovals. I plan to render two main images. The reason for rendering multiple images is that the focus of my theme is strong visual attractiveness. The candies will be rendered in a way such that the colours and lighting effects are stunning to the eyes. I am trying to put my basic elements into two settings for visual comparison purposes as well. The first image will have 5 - 6 candies with different colours sitting on top of a mirror (or a surface that acts like a mirror). There will be another such mirror that stands straight behind. This will be an interesting scene because the mirrors will reflect the candies and the viewer will see a lot more candies than there actually are. In the second scene, the candies will be placed above (or on top of) a wooden table surface surrounded by four marble pillars. This scene intends to show the ”soft side” of the candies in settings with real-world objects. That is, the visual effect will not be as sharp as the first scene. For the final (non-animated) scene, I will pick the better one of the two to perform fisheye projection. Also, all the images will have my signature at the bottom right corner. I do this because it is customary for artists to sign their art pieces. For Demo purposes, I will also pre-render a few (either perspective or fisheye projection) images with the camera placing at different angles and make a movie clip.

To implement a Raytracer capable of rendering the above scenes, I will need to implement several extended raytracer features. In summary, I will need reflection, refraction and soft shadows to make the candies look more realistic. Anti-aliasing technique is used to make sure that the edges/surfaces of my objects are smooth. Texture mapping is required for the table top and pillars.
Additional primitive - cylinders - are implemented to effectively model pillars. Also fisheye projection is used to render final scene.

I believe my project will be interesting because the realistically stunning colours of the objects will create a strong visual sensation to the viewer. Also the viewer will be seeing the scene as if viewing through fisheye a lens. The viewer will also feel the difference of the candies when they are being put in two settings.

I hope to learn new Raytracer techniques in photorealistic modeling through the project. For example, I will be learning adaptive aliasing on my own to achieve my objective. I also hope to learn more about how lights and the colour of the objects interact in different settings. My ultimate goal at the end is to create a scene that I can proudly show to other people.

4. Goals

• A Raytracer that renders the scenes described above

• A final scene with fisheye projected view and signature

• For Demo purpose, a clip viewing the scene at different angle

5. Communication

The basic flow of information is as follows:

Input :
A scene description file written in Lua.

Interaction :
The Raytracer will not take in any user input.

Output :
Scene images as described in Lua file and in previous sections of the proposal.

6. Modules

The extended Raytracer is based on the basic Raytracer I did for Assignment 4. Therefore the organization of the files will be the same as A4. I will not be using any assistance from other software. I will be generating the scene descriptions all by myself.

7. Technical Outline
The following sections outline the ten objectives I hope to achieve by the end of the project. My goal is to render a scene full of artistic aspects while at the same time retain technical complexity. The extra feature I implemented in A4 was anti-aliasing with supersampling. Therefore this feature will not be an objective of my project.

7.1 Additional Primitive: Cylinder

The basic Raytracer in Assignment 4 supports spheres and cubes. These are the two primitives I will use to model candies, mirrors and table surface. I will in addition support cylinders, which will be the primitive I use to model pillars. To implement the support, I will need to create new parser for Lua scripts that define cylinders. For example, a gr_cylinder_cmd may be added. Also I will need to implement a function to test for intersections of a ray with a cylinder. The equation of an elliptic cylinder is \( \frac{x^2}{p^2} + \frac{y^2}{q^2} = 1 \). I will use the same way that Prof. Cowan showed us in class for ray-triangle intersection.

7.2 Glossy Reflection

This objective intends to make the candies look more interesting. One property I will add to material is “Glossy”. If a candy has glossy property, its surface will perturb its reflective ray in a cosine distribution.

7.3 Mirror Reflection

For each object with “mirror coefficients > 1”, I will apply (recursively) secondary reflection rays from the point of intersection. In theory, the direction of the secondary ray will be symmetrical to the incident ray that hits the intersection. The mirror coefficients will be taken into account when updating the pixel colours. I will also need to set a threshold for the depth of the recursion.

7.4 Refraction

Refraction involves recursively generating secondary refracted rays for objects with “transparency coefficient” and an index of refraction. As suggested in A4, I will be using Snell’s law to compute the direction of the refracted ray. Snell’s Law:

\[ n_1 \sin \theta_1 = n_2 \sin \theta_2 \]

where \( n_1, n_2 \) are refraction indices and \( \theta_1, \theta_2 \) are refraction angles.

7.5 Texture Mapping

I will be creating a straightforward texture mapping as taught in class and some help from the internet. Texture mapping will only be applied to table surface and pillars.
7.6 Adaptive Anti-aliasing

In the first time, I will send one ray for each pixel. The iterative method will examine each pixel to find out a set of “bad pixels”. A pixel is “bad” if it differs from their surrounding pixels by more than \( \theta \), where I will define \( \theta \) as a threshold. After determining the “bad” set, I will raytrace these pixels again with multiple rays. The advantage of doing adaptive anti-aliasing is that in the second time, I only raytrace the “bad” pixels that usually just happen at the edge of objects. Adaptive anti-aliasing will save a lot of computationals.

7.7 Area Light and Soft Shadow

To implement soft shadow, I will treat light as area light rather than point light source. I will send multiple shadow-rays to various points on the light to compute a percentage of shadow-rays that hit the light. Then I will determine the relative amount of shadow.

7.8 Signature using Embossing (Bump Mapping)

After rendering the image, I will “sign” the image with my initials. To achieve this, I will use bump mapping to create embossing effect. If embossing goes well with the final scene, I may create some additional textures at the corners of the picture.

7.9 Fisheye/Omnimax Projection

For basic raytracers, perspective projection is used. Perspective projection assumes that lights from the objects are landed on a (flat) plane of what we call “imaginary screen”. For Omnimax projection, the “imaginary screen” needs to be bent, having a shape of a hemisphere. The rays from the viewer will then be casted from the viewer, through the hemispherical imaginary screen, to the objects.

7.10 Final Scene with Moving Camera

I will use my artistic skills to create the models for my scenes. Then I will arrange them nicely into the coordinated system. It is hard to imagine which one of the two rendered scenes will turn out to be better, so I will make a choice after they are both rendered. The better one will be picked to have fisheye projection. Since Raytracer tends to take a long time to render, I will pre-render the scenes viewing at different position/angle. I will use scripts to connect the scenes together, play them one by one to create animation effect.

8. Milestones
• Additional Primitive: Cylinder

• Glossy Reflection

• Mirror Reflection

• Refraction

• Adaptive Anti-aliasing

• Texture Mapping

• Area Light and Soft Shadow

• Signature using Bump Mapping

• Fisheye/Omnimax Projection

• Final Scene with Moving Camera

9. Bibliography

I have used and will be using the following sources of help. I will likely to find more sources for help as I proceed to implement my project.


Chris Cooksey, “Antialiasing And Raytracing”, January 1994,

Jacco Bikker, “Raytracing”, June 2005,
URL: http://www.devmaster.net/articles/raytracing_series/part3.php

10. Organisation
The files and organization will be similar to Assignment 4. I will implement extra classes if needed.

11. Documentation

The README file will have brief instructions on how to run the Raytracer. A final REPORT will have details on algorithms and data structures I used in my projects.

12. Sources

Source files will be src/*.cpp and src/*.hpp make will compile the Raytracer.

13. Executable

./rt[cmd]. I may decide to give some freedom to the user to specify certain parameters.

14. Data Files

All the Lua files will be in /data directory.
Objectives:

Full UserID: ______________________
Student ID: ______________________

The extra feature in A4 is anti-aliasing using supersampling

--- **Additional Primitive**: Cylinder is supported and Ray-Cylinder intersection works.
--- **Glossy Reflection**: Objects that have glossy material property exhibits glossy reflection.
--- **Mirror Reflection**: Objects that have mirror-reflective property exhibits mirror reflection.
--- **Refraction**: For translucent objects, Snell’s Law is used to case secondary rays to create refraction effect.
--- **Adaptive Anti-aliasing**: Adaptive Anti-aliasing is carried out in the scene to remove jaggies.
--- **Texture Mapping**: Texture mapping has been implemented.
--- **Area Light and Soft Shadow**: Area lights are supported and additional shadow rays are used to create soft shadows.
--- **Signature using Bump Mapping**: All the scenes have a signature at the bottom right corner of the image.
--- **Fisheye/Omnimax Projection**: Omnimax projection is applied to the final scene.
--- **Final Scene with Moving Camera**: A unique scene is generated to demonstrate the project.

Declaration:

I have read the statements regarding cheating in the CS488/688 course handouts. I affirm with my signature that I have worked out my own solution to this assignment, and the code I am handing in is my own.

Signature: