CS488 Final Project
Title: Ray Tracer
Name: Tianjun Zhang
Student ID: 20643273
User ID: t269zhan

# Documentation:

**Manual** :

This program is extended from the ray tracer in Assignment 4, with some changes to the way some features are activated. Most objective features are controlled by define's in a4.cpp, refer to the **README** for a detailed guide of how to turn on each feature and render a new scene. For this reason, a recompile might be needed to turn on some of the features.
3 of the features make the rendering time significantly **longer**, as the computation needed are rather costly. These features include :soft shadow, glossy reflection and depth of field.
Every objective was completed except CSG and bump mapping. While bump mapping produces some effect on the image, I did not arrange my time wisely enough to work on CSG.

**Implementation and Technical Outline** :

**!**

Some features can be turned on/off in *A4.cpp* using *#define's*. Refer to the README for a detailed guide and matching scripts to run. Some usages will be included here as well.

**!**

Multi-threading using *future* is implemented and makes the rendering more efficient with more logical cores available. However, this method was not able to run in the graphics Lab, tested on multiple machines. The reason for this is unkown but single-threading works just fine, only slower. (Might have to do with the different processor specs between my intel chips and the amd ones in the lab.)
Accordingly, I made some last minute change so that single/multi threading can be toggled by "define MULTI" in A4.cpp. A recompile is needed.

**Custom Primitives** New primitives created include: Cone, Cylinder and Torus. The mathematical representation of these shapes in 3D spaces had to be studied to be able to determine if a valid intersection has happened. Torus wasn't listed as one of the objectives, but for other reasons it had to be implemented to complete my final scene. The implementation of these classes can be found in *Primitive.hpp/cpp*, loacated after the A4 primitives.

**Reflection and Refraction**

Since part of the physical nature of the light ray was already implemented in Assignment 4, the main task for these two features was further understanding light transmission. I was able to simulate mirror reflection by using a reflectance value and calculating the color value at an intersection point. In this program the recursive depth limit is set to 5, since I was not able to notice much visual difference as I increased the value. Using Snell's law I was able to simulate refraction using vector information from the incident ray and the intersection normal. Implementation are done in *A4.cpp*, using *directLight*, *refract* and *fresneleffect*.

**Fresnel**

I was not familiar with this effect before implementing this program. With some reference of the mathematics equations in online materials and guides I was able to calculate a more proper reflectance value given the index of refraction.

**Glossy Reflection**

Glossy reflection was achieved by perturbing the incident ray. By casting multiple rays and randomly changing the rays' direction, I was able to generate different level of blurriness with the given index. Extra helper functions had to be implemented to cast a ray from a random point in a hemisphere range and eventually find needed amount of valid rays stored in a vector of rays.
A difficulty was that sometimes the program would be stuck in a very lengthy loop trying to find enough valid rays. A get-around was to introduce a hard limit in the ray-casting loop so that the loop would break when the limit is reached. Implementations are in *A4.cpp* To activate this feature, make sure that in A4.cpp:

```
13 #define GLOSSY_RAYCOUNT
```

GLOSS_RAYCOUNT is set to a reasonable value

**Soft shadow**

Soft shadow implementation had something in common with the glossy effect before. To get a softer edge of the shadow, more light rays are casted in random directions towards the intersected object.
A new type of light source was implemented. I chose a sphere shaped light to be able to randomly cast a ray from a hemisphere range. Similar to before. Like glossy reflection, this way of implementing soft shadow also requires increasing the amount of rays casted, which makes the rendering loop a lot more expensive. To activate this feature, make sure that in A4.cpp:

```
13 #define SS_RAYCOUNT
```

SS_RAYCOUNT is set to a reasonable value and an area light source is used

**Depth of Field**

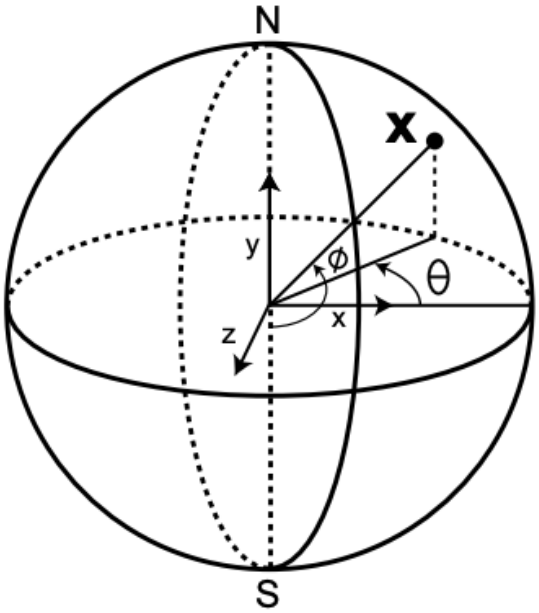
This feature again requires casting multiple rays to simulate the effect. What I did was that I traced the eye vector to the intersection point. At a given focus distance along the eye vector, I trace back to a randomly selected sample point in front of the eye position(Points chosen by uniform distribution in a small block). The randomizing makes blurs the scene. The quality goes up when I increase the amount of rays casted, but so does the cost and rendering time. To activate this feature, make sure that in A4.cpp:

```
10 #define ANTIALIASINg
11 #define AA_Mode 2
```

ANTIALIASING is turned ON and AA_Mode set to 2

**Texture Maps**

I implemented texture mapping for planes and spheres. The difficult part was mainly to find the representation of any given point on the object and map that to a pixel of the given picture (uv-mapping). Planes are trivial Parametric coordinates system was used for spheres.




By computing $\phi$ and $\theta$ we get a complete map from pixel map to sphere to texture map.
A new class *Texture* (and its subclass *ImageTexture*) is created to store and pass the map information.

**Bump Maps**

Well this feature really "bump"ed me. In my attempts to finishing it, I studied the geometry and some similar features like normal mapping, using a TBN matrix. In the end this feature failed to produce ideal effects as it can capture the color patterns in the bump map but can't produce rough surfaces. Must be something wrong with the normal manipulation.

**CSG**

Underestimated the difficulty to add this into my program. No implementation. The rim in my final scene was planned to be made with 2 cylinders subtracting each one another, a replacement appraoch was to use a turos primitive.

New *input language commands* will include:

gr.cylinder
gr.cone
gr.light
gr.turos
gr.spherical$_l ight$
$set_m aterial$
$set_t exture$
$set_g lossy_i nd$

**Final Scene:** The final scene was as planned an image of a indoor ballcourt. Due to limitations of machine performance and code optimization, I couldn't generate a scene with every feature enabled in time. Instead, I have a few of them each having different features. How to manage the cost of distributed ray tracing features like glossy reflection and depth of field is an interesting problem.

**Bibliography** :


Cook, R., Porter, T., Carpenter, L. (1984). **Distributed ray tracing**. ACM SIGGRAPH Computer Graphics, 18(3), 137-145. doi: 10.1145/964965.808590

Eric A. Bier, Kenneth R. Sloan, Jr., **Two-Part Texture Mappings**, IEEE Computer Graphics and Applications, Vol. 6, No. 9, Sept. 1986, pp. 40-53.

James F. Blinn, Martin E. Newell, **Texture and Reflection in Computer Generated Images**, CACM, Vol. 19, No. 10, Oct. 1976, pp. 542-547.

Policarpo, F., Oliveira, M., Comba, J. (2005). **Real-time relief mapping on arbitrary polygonal surfaces**. ACM Transactions On Graphics, 24(3), 935. doi: 10.1145/1073204.1073292

Reeves, T.(1983). **Particle Systemsm A Technique for Modeling a Class of Fuzzy Objects**. ACM SIGGRAPH Computer Graphics, Vol. 17, pp. 359-375

Suffern, K.(2007). **Ray Tracing From The Ground Up**, Chap. 25 Glossy Reflection, pp. 529-542

Shirley, P., Ashikhmin, M. (2005). **Fundamentals of computer graphics**, third edition. Wellesley, Mass.: A K Peters, pp. 310-317

Introduction to Shading (Reflection, Refraction and Fresnel). (2019). Retrieved 3 December 2019, from https://www.scratchapixel.
basic-rendering/introduction-to-shading/reflection-refraction-fresnel
https://people.cs.clemson.edu/ dhouse/courses/405/notes/texture-maps.pdf

## Objectives:

**Full UserID:t269zhan** **Student ID:20643273**

___ 1: Add primitives: cylinder, cone

___ 2: Adding reflection Adding refraction

___ 3: Adding refraction

___ 4: glossy reflection

___ 5: soft shadow

___ 6: texture mapping

___ 7: depth of field

___ 8: bump mapping

___ 9: CSG

___10: Complete the final scene

A4 extra objective: Anti-Aliasing