My Ray Tracer

Name: Yunfei Zhao

Student number: 20614564

Student id: y399zhao

Purpose:

Render picture containing different kinds of objects including primitives I implemented with features such as the Refraction and CSG.

Introduction Statement:

Some of the objectives can be implemented with the same technique. For instance, Anti-aliasing, soft shadow as well as depths of field can be implemented using super-sampling. Bump mapping is implemented based on texture mapping; it's still using the kd value generated by uv interpolation from texture mapping. Moreover, it adds the bumps by changing the normal vector. The rather challenging part of the project id Refraction and CSG. For Refraction you need to take both reflection and refraction into consideration. And CSG need to trace t during the entire process for the 2 interactive objects. Based on these objectives, I made a final scene that combines some of these features.

Technical Outline:

Anti-aliasing:

Improve the appearance of the shapes we have, since I use super sampling to implement anti-aliasing. For each pixel we look at, divide it to 64 subpixels. To use more intensity levels to do anti-aliasing, we take randomize more subpixel across this region. Then, average out the colors generated for all the subpixels in this area so that intensities can be averaged with adjacent pixels to give a rather smother color variation across pixels. [1]

Extra-primitives:

The primitives I implemented are cylinders and cones. To implement the intersection function. We need to order the values of t and calculate the z value at the intersection point with the cylinder. [2] There are 3 cases, one is hit the up cap, another one is hit the bottom cap, the final one is hit the bit. For other cases, we miss the ray. It's quite the similar case with the implementation with come. Except now we have one less cap.

Soft shadows:

Simulate the area light by representing the area light as an infinite number and choose one at random for each viewing ray. To scramble the ray from pixel to light, we can shuffle the samples around to have randomized ray from pixel to light. [3]
For each pixel, generate 2 distinct samples, use the random point in 1 array to avoid the case when pixel for 1 certain direction always generate ray to the same direction of the light source.

Depths of field:

To simulate the camera, fire multiple rays for each pixel, set the focal plane to be the place where you want your object to be in focus. Randomize the position of our eye position, fire same number of rays for every pixel, 2 cases need to be considered. First, if our ray intersected with the focal plane, use this ray to set color of the pixel. If not, then add all color we collected and then divide by the number of rays we cast.[4]

Glossy reflection:

Glossy reflection first use the reflection ray I generated, then obtain some perturbed ray and then average out the color generated.

CSG:

There are 3 cases to consider in CSG. The trivial one the union. Another one is Intersection. [5] First set the brother relationships between the objectives we want to interact with. Test 2 t values for each object. One is the nearest t, another is the furthest t. Compare them with that of the brother object. It's the similar case with difference. One thing to notice is that we need to set the relationship at the start and after we deal with one of the 2 objects interacting, skip the other one.

Refraction:

First determine if an object is transparent. Then get the reflected ray. Then use the surface normal n and the ray direction to form an orthonormal basis for the plane of refraction. Then solve for the transformed ray t. [6] Then approximate the Fresnel equation using Schlick approximation and get R0 which is the reflectance at normal incidence. And after that, get R theta. And using these R value, combine with reflect ray and the refract ray to generate the color.

Texture Mapping:

For each pixel, we find the corresponding di, dj in the raster file. Find the up, vp for that pixel and then make sure that the up and vp are within the raster file. Finally, interpolate the color from the surrounding pixels, using bilinear interpolation. I implemented texture mapping on 3 primitives, cylinder, sphere and cube. Which use different u, v for different primitives.

Bump Mapping:

Bump Mapping is implemented based on Texture Mapping. The kd is the same as generated by texture mapping. However, for the lighting, we need to use the perturbed normal to calculate. For me, I use a website to generate my image with normal already changed to perturb my normal.

Final Scene:

My Final Scene's background uses a scene from my favorite movie Spirited Away. The wooden floor combines both texture mapping and glossy reflection. The Watermelon and the tennis ball use bump mapping and the glass water shows refraction. The watermelon floating in the water combines CSG and texture mapping.

Implementation:

Basketball:
https://www.uihere.com/free-graphics/basketball-texture-free-vector-svg-eps-file-103206

Tennis ball:
https://sp.depositphotos.com/34878821/stock-illustration-illustrated-texture-of-wood-grain.html

World map:
https://pt.wikipedia.org/wiki/Paralelo_43_N

Joker:

https://archive.4plebs.org/tg/thread/54809082/

King:
https://www.pinterest.ca/pin/392587292489048416/

Wooden Wall Panel:
https://www.indiamart.com/proddetail/textured-wooden-wall-panel-16422837233.html

Wooden floor:
https://sp.depositphotos.com/34878821/stock-illustration-illustrated-texture-of-wood-grain.html

Watermelon:
https://www.amazon.com/My-Little-Nest-Placemats-Watermelon/dp/B07BJGSLBQ

Watermelon seed:
https://www.istockphoto.com/ca/vector/watermelon-seeds-background-pattern-juicy-sweet-flesh-watermelon-with-black-grains-gm1133368336-300781594

Sea:
Screen shot from “Spirited away”

Bibliography:

1. Computer graphics with Open GL. 4th, Hearn, Baker, Carithers, 2014, pp. 177-184
2. Cylinder intersection. (n.d.). Retrieved from http://woo4.me/wootracer/cylinder-intersection/
3. Fundamentals of Computer Graphics, 4th, Peter Shirley, Steve Marschner,2015, pp. 331-333

4. Fundamentals of Computer Graphics, 4th, Peter Shirley, Steve Marschner,2015, pp. 332-333
5. http://web.cse.ohio-state.edu/~parent.1/classes/681/Lectures/19.RayTracingCSGHandout.pdf
6. Fundamentals of Computer Graphics, 4th, Peter Shirley, Steve Marschner,2015, pp. 324-327

Compilation:

premake4/make combination

wrintten in Atom

run on gl38.student.cs

Manual:

./A4 Assets/extra-primitives.lua for extra-primitives

./A4 Assets/simple-cows.lua with the antialiasing tag in A4.hpp for anti-aliasing

./A4 Assets/soft_shadow.lua with the SOFTSHADOW tag in A4.hpp for soft shadow

./A4 Assets/simple.lua with the GLOSSY_REFLECTION SOFTSHADOW tag in A4.hpp for glosst reflection

./A4 Assets/my_csg.lua for CSG

./A4 Assets/refraction.lua with the REFRACTION tag in A4.hpp for refraction

./A4 Assets/bump.lua for Bump mapping

./A4 Assets/my_texture.lua for Texture mapping

./A4 Assets/cylinder.lua for final scene

Objectives:

1. Anti-aliasing
2. Extra primitives (cylinder, cone)
3. Soft shadows
4. Depths of field
5. Glossy reflection
6. CSG
7. Refraction
8. Bump mapping
9. Texture mapping
10. Final scene

A4 extra objective: Reflection