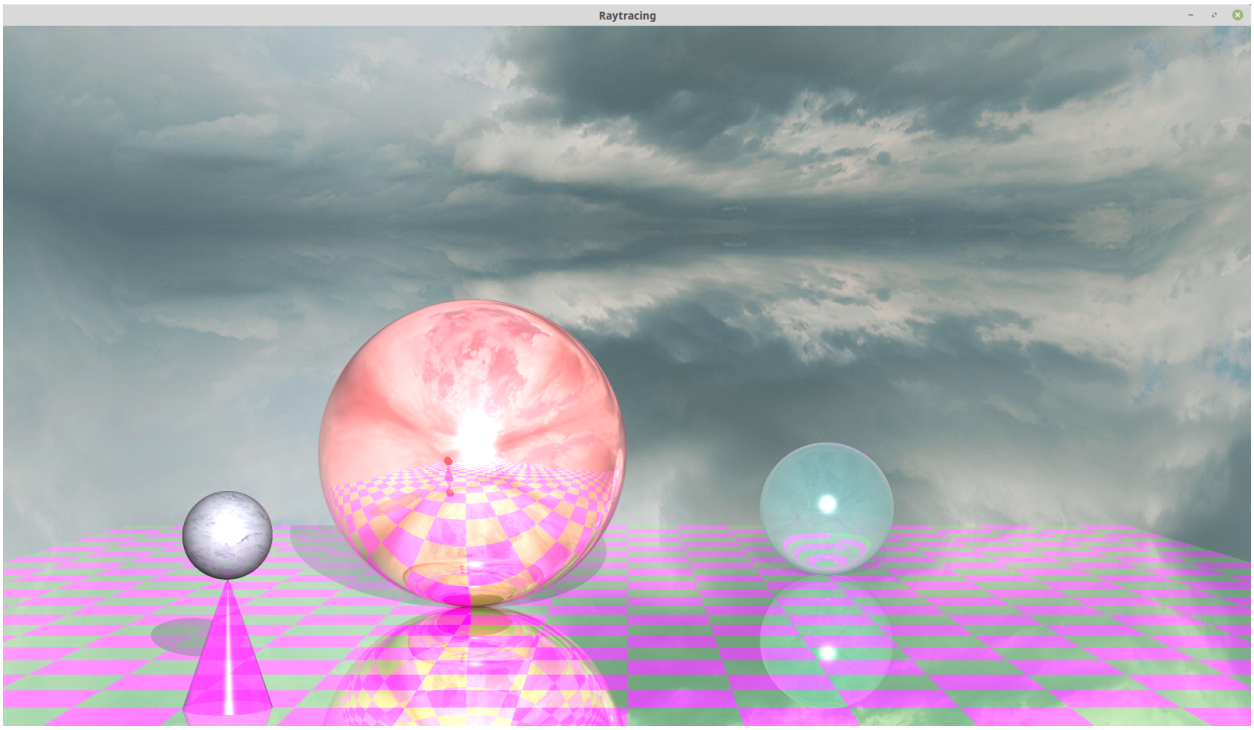
COSC 363 Assignment 2
RayTracer in
OpenGL2



Shai Levin
59368709

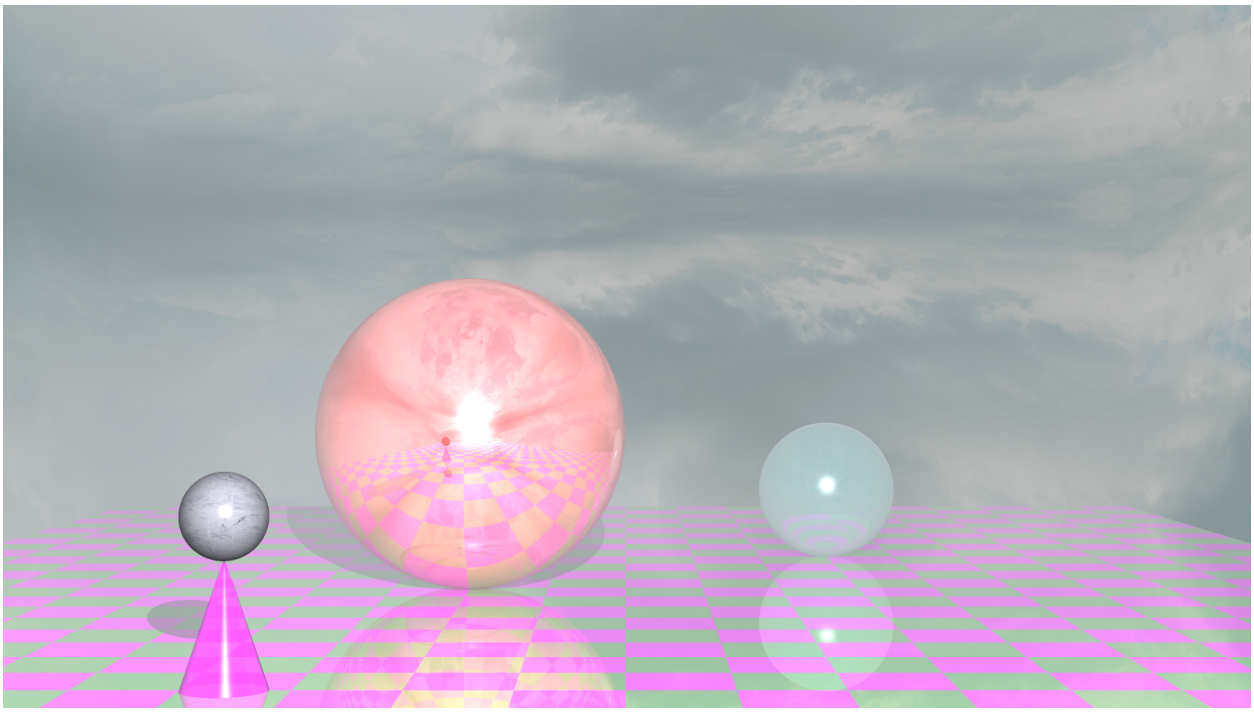
Successes and Failures

My personal goal of my RayTracer scene was to generate a realistic and visually appealing image free of jagged edges, artifacts, in high resolution. The drawbacks of this is a high level of attention to detail was required to assemble the scene, and the generation of a high resolution image would be computationally expensive. I think I succeeded and the image is very visually appealing and demonstrates all the important features of raytracing, with no glitches or artifacts.

A couple failures. The cone object caused many issues and I struggled to derive a formulation from the notes alone, after some trial and error. Oddly enough, it would generate a paraboloid instead of a cone. I eventually looked online and used the equations in (4). These worked well in the end.

I also encountered a strange bug with Textures. When I mapped my textures to planes, they would sometimes wrap around slightly along the s parameter. The easy solution was not ideal, but I simply adjusted the mapping to account for this.

Lastly, there were some issues with black artifacts being caused when transparent objects would intersect the same point as soon as it reflected. To fix these artifacts I simply increased the value at which objects would be ignored if they intersect with a ray near-asymptotically.



Final Result of Project with Fog and AA Enabled

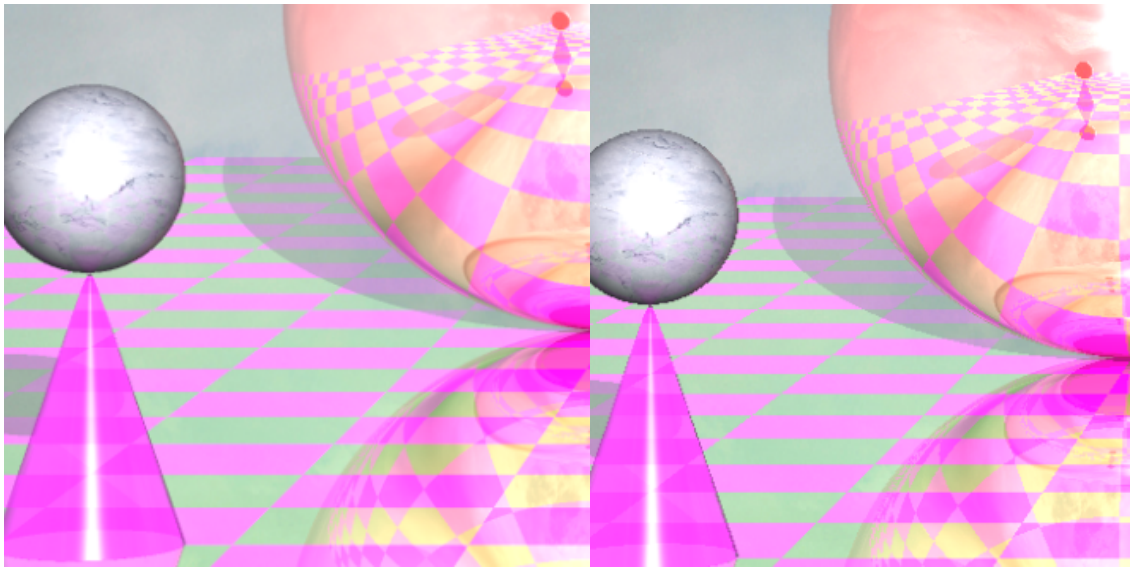
Required Features

All required features implemented. It should be noted the object constructed from a set of planes is the skybox, and the transparent object is the cone.

Bonus Features

For the equations I used in some of my additional features - since I would essentially be copy-pasting from my references, and they can be lengthy, I have omitted them - please check my references if they are required.

Anti-aliasing (Supersampling)



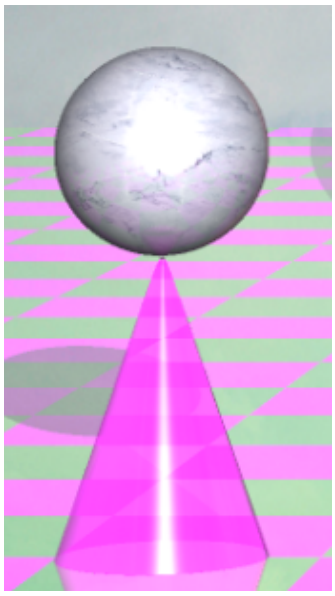
The AA implements 4x supersampling as described in the lecture notes. In each pixel, a ray is traced from the center of each quartile of the pixel, and the resulting color value is the average of the 4 colours. See above - left is with AA, right is without AA.

Refractive Objects

The refraction is implemented exactly as in the lecture notes. Using glm::refract(). See the notes for the equations.

Cone

The cone intersection method is based on the equations found in (4). With one alteration - the check whether or not a point of intersection lies within the finite cone boundaries. I simply checked if the y value of the point of intersection was between the base and tip of the cone.



Textured Sphere

A seamless bmp texture was mapped to the sphere by UV mapping where u is in [0,1] and v is in [0,1]:

$$u = 0.5 + \frac{\arctan2(d_x, d_z)}{2\pi},$$
$$v = 0.5 - \frac{\arcsin(d_y)}{\pi}.$$

Fog

Fog is implemented just as described as in lecture notes. It can be disabled in the constant variables at the top of the program.

Skybox

The skybox is constructed by mapping front, left, right, back and top planes to a cubemap. This is similar to the Assignment 1.

Render Time

As my goal was to produce a clean, high resolution 16:9 output, render times definitely take a hit. At 1600x900 pixels, with AA, render time is approximately 40 seconds. Without AA closer to 10 seconds. I expect this might take a while on a slower machine, so I recommend disabling AA, and changing the NUMDIVX, NUMDIVY values to a smaller 16:9 ratio if render time is an issue.

References

1. Skybox textures retrieved from: https://opengameart.org/node/11726
2. Seamless cubemap texture for marble sphere retrieved from https://3dtextures.me/2019/10/07/marble-tiles-001/
3. (U,V) mapping to sphere equations retrieved from https://en.wikipedia.org/wiki/UV_mapping
4. Cone intersection method calculated using equations from http://www.illusioncatalyst.com/notes_files/mathematics/line_cone_intersection.php
5. Raytracer, Ray, SceneObject, and other classes are based on the code from Lab7/8.