Ray Tracer Report

DPE58 89570357

Build Commands

1. **Navigate to the directory where the files are stored**
2. **Run the following commands one line at a time**

   g++ RayTracer.cpp Sphere.cpp SceneObject.cpp Ray.cpp Plane.cpp TextureBMP.cpp -o raytracer -lGL -lglut -lGLU

   ./raytracer

Estimated generation time: 1-2 seconds

Successes and failures

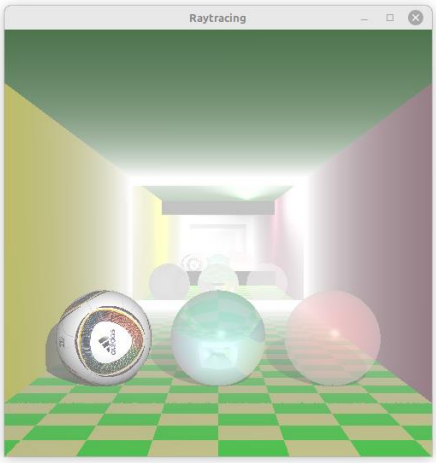

Figure 1. The Ray Tracer scene

The ray tracer successfully meets the basic ray tracer requirements. The scene includes a box defined by 5 axis-aligned planes with each plane a different colour along with the floor plane featuring a chequered pattern. The chequered pattern was generated by extending the lab7 stripe pattern. One challenge and initial failure while working on the project was encountered through its generation. My attempts to generate the pattern had a minor error where the tiles that met at x = 0 were the same sequence as shown in Figure 2. As a solution, I added a simple if statement that switched the colour order on the other side of the x axis.

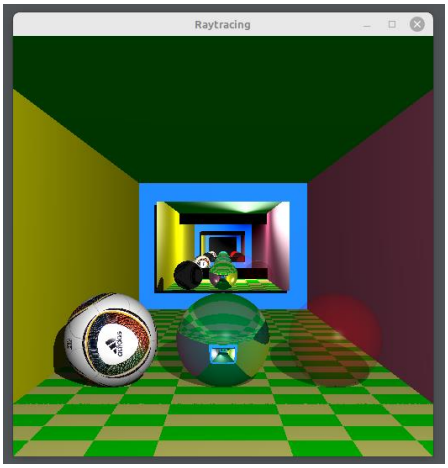

Figure 2. The Ray Tracer scene with incorrect floor pattern

The program also includes a transparent object that was generated through recursively tracing rays where the resulting colour is a combination of the refracted colour and the red colour of the sphere. Both this transparent sphere and the refracted sphere have lighter shadows compared to the solid textured sphere as shown in Figure 2. The scene also contains an example of reflection through the mirror-like at the back of the box that reflects the surrounding objects. This was accomplished through the same method as the reflective sphere in Lab07, with the difference being the material colour of (0, 0, 0) and coefficient of reflection of 1.

The ray tracer fails in its lack of more complex examples of ray tracing such as depth of field, soft shadows, etc. Additionally, the ray tracer program features no anti-aliasing and hence the scene contains more pixelated objects and shadows than possible. The implementation of these features were attempted over the period of the assignment but not successfully implemented.

Extra Features

Refraction of light through sphere

The Ray Tracer implements refraction of light through an object as seen in the middle sphere. This was accomplished through ray tracing algorithm from lectures that traces rays recursively to account for several internal reflections. The refraction has an eta value of 1.5 to emphasize the effect of refraction.

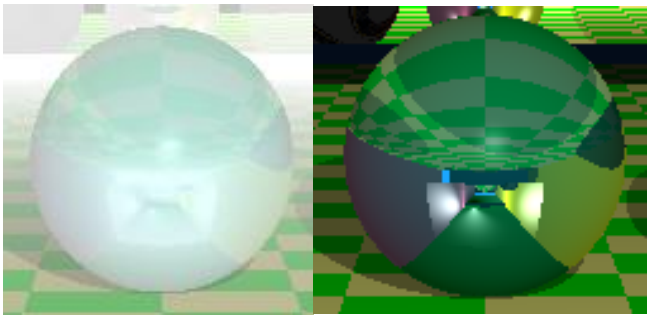

Figure 3. The refraction of light through a sphere with and without the fog

```
if(obj->isRefractive() && step < MAX_STEPS)
{
    float eta = 1/obj->getRefractiveIndex();

     glm::vec3 normalVec = obj->normal(ray.hit);
     glm::vec3 refractedDir = glm::refract(ray.dir, normalVec, eta);

    Ray refractedRay(ray.hit, refractedDir);
    refractedRay.closestPt(sceneObjects);

    glm::vec3 m = obj->normal(refractedRay.hit);
    glm::vec3 h = glm::refract(refractedDir, -m, 1.0f/eta);

    Ray secondaryRay(refractedRay.hit, h);
    glm::vec3 refractedColor = trace(secondaryRay, step + 1); //recursion
    color = refractedColor * obj->getRefractionCoeff()+(1-obj->getRefractionCoeff())* color;
}
```

Figure 4. The refraction code using the ray tracing algorithm

Multiple reflections generated using parallel mirror-like surfaces

The multiple reflections are generated through the use of an additional mirror plane that is facing the first creating an infinity like effect. Both mirrors were implemented the same way as described earlier.

Figure 5. The multiple reflections with and without the fog

Textured sphere

The left sphere was textured using the implementation outlined in the lecture notes that maps the point of intersection to coordinates of the Jabulani texture.

Figure 6. The textured sphere

Fog

The fog has been generated using the method outlined in Notes07 and with the vanishing point of -195 on the z axis. The difference between the scene with and without the inclusion of fog feature is demonstrated in Figure 7.

Figure 7. They ray tracer scene with and without fog

```
float t = ray.hit.z / -195;
color = (1-t) * color + t * glm::vec3(1, 1, 1);
```

Figure 8. The code used to add fog to the scene

References

Jabulani Texture

https://www.flickriver.com/photos/dmswart/4690051366/

Declaration

I declare that this assignment submission represents my own work (except for allowed material provided in the course), and that ideas or extracts from other sources are properly acknowledged in the report. I have not allowed anyone to copy my work with the intention of passing it off as their own work.

Daniel Peploe

89570357

2/06/2023