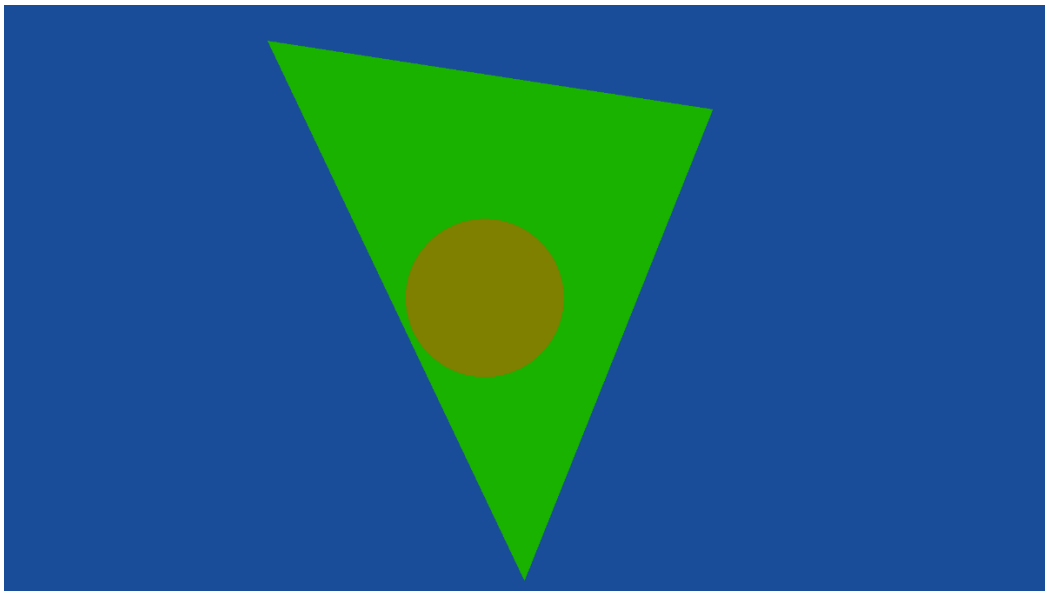
CG Assignment-4

Rahul Oberoi (2021555)

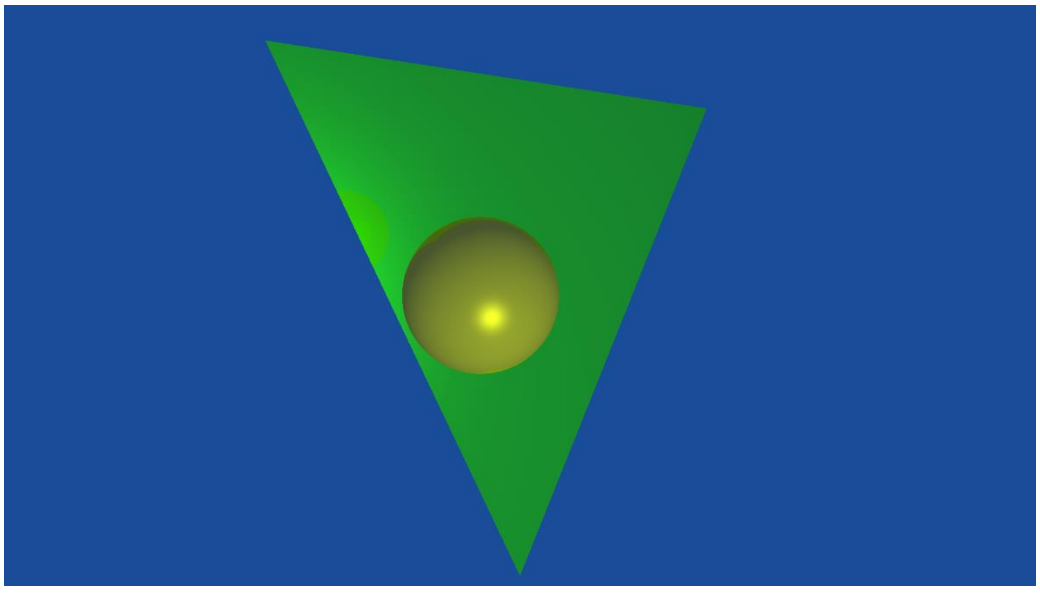
Part 1: Triangle Implementation

The Triangle has been implemented in the code behind the sphere. This Triangle was used as a surface to cast shadows from spheres onto, enhancing the realism of the raytracer. The code maintains a good balance of functionality and code quality with proper documentation.



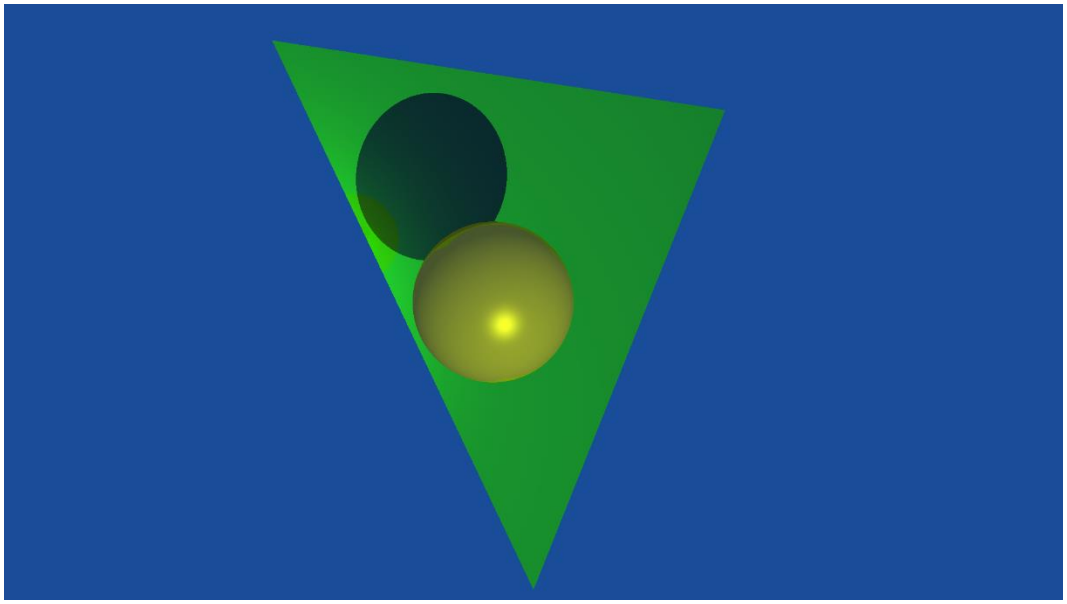
Part 2: Blinn-Phong Shaders

The code calculates the shading of a material based on ambient (a), specular (s), and diffuse (d) components. These components are computed by considering the interaction between light, surface, and the viewing direction.



Part 3: Shadow Implementation

A shadow ray is created to check for intersections with objects in the world. If the shadow ray intersects with an object, only the ambient and specular components are returned, simulating shadows.



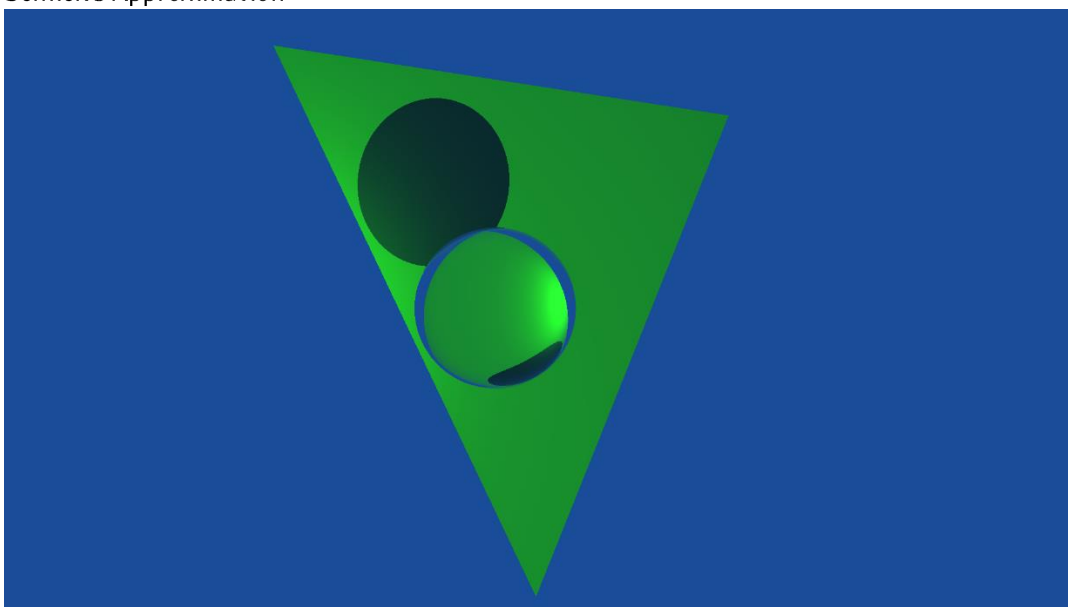
Part 4: Schlick's Approximation and Beer's Law (Dielectric materials)

- Reflection Calculation:
  - The code handles reflective materials by calculating the reflection direction (r) when a ray hits an object.
  - It creates a new reflected ray and recursively traces it to calculate the reflection.

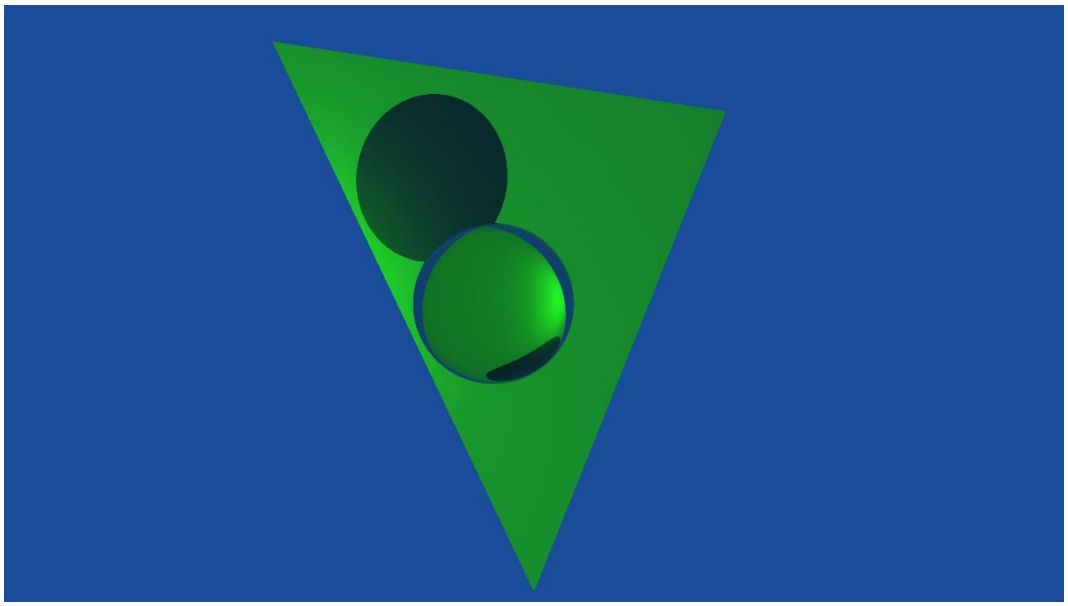
- Recursion Limit:
  - The recursion is limited to 10 levels (controlled by the 'count' parameter) to prevent infinite loops and excessive computation.

- Refraction Calculation:
  - For dielectric materials, the code calculates refraction and reflection.
  - It uses Snell's law to calculate the refracted direction (t) and checks for Total Internal Reflection (TIR) with the 'tir_check' function.

Schlick's Approximation



Beer's Law



Bonus Component:

```
glfwInit();
glfwWindowHint(GLFW_SAMPLES, 4);
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GLFW_TRUE);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
```

References:

- Lecture Notes
- https://learnopengl.com/Introduction
- https://github.com/glfw/glfw
- https://stackoverflow.com/questions/3667218/how-to-do-ray-tracing-in-modern-opengl
- https://github.com/engilas/raytracing-opengl