COSC363-22S1 Assignment 2 Report

Name: Meng Zhang
User Name: mzh103

Project Description

The project creates a set of 3D objects with global illumination using the ray tracing technique. The following objects are included in the scene:

- Basic Objects
    - 1 green transparent hollow sphere
    - 1 mirror-like reflective sphere
    - Cast shadows for all objects, and light shadows for transparent and refractive objects
    - 1 pyramid made from 5 planes
    - 1 room made from 6 planes
    - 1 floor with chequer pattern
- Objects with Extra Features
    - 2 glass-like spheres with different refractive indices
    - 2 light sources, making multiple shadows and specular highlights
    - Anti-aliasing
    - 1 sphere textured using an image of earth
    - Multiple reflections between 2 mirror-like walls
    - 2 silver cylinders, one connecting the earth globe to the ceiling and one connecting the pyramid and the floor

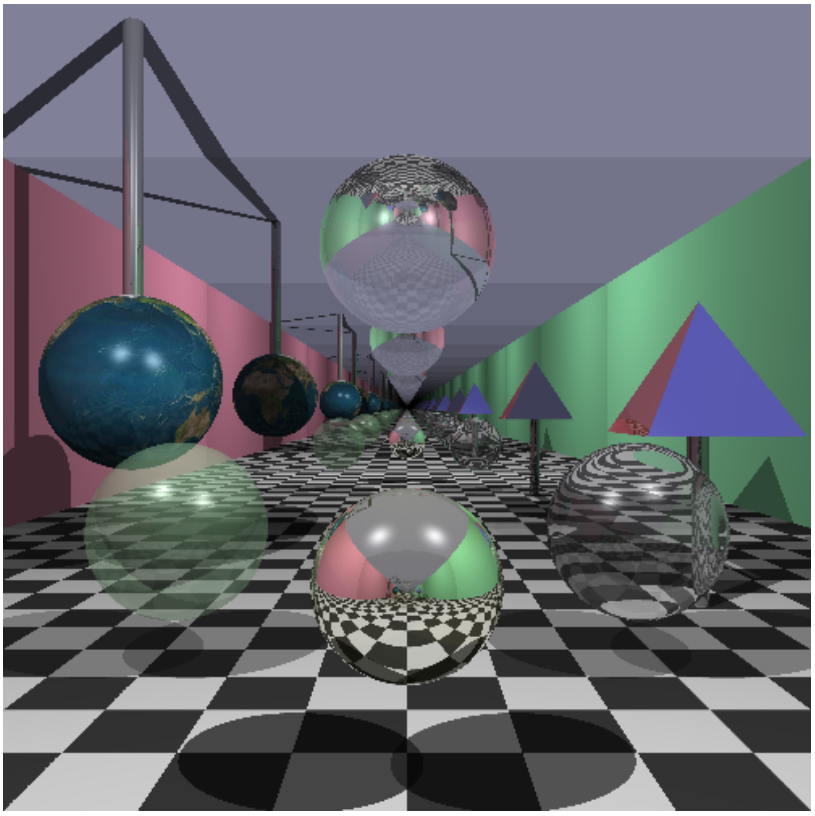


Image-1

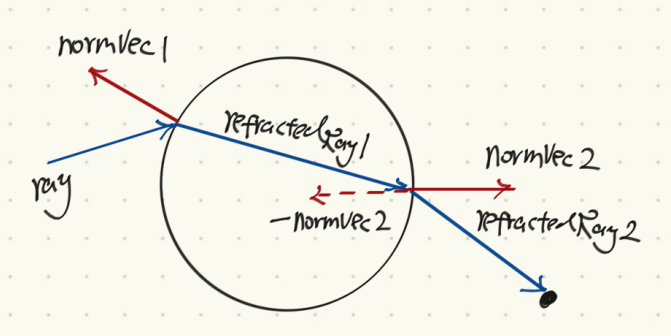
Extra Feature Description

1. **Glass spheres with different refractive indices**

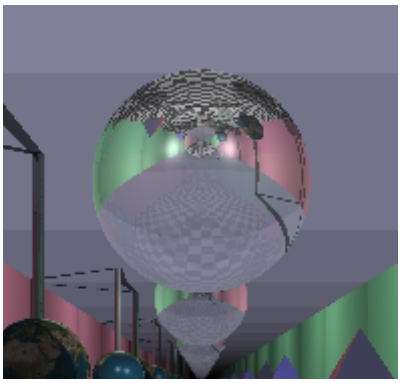
File: RayTracer.cpp
In initialize():

- Create a sphere, and set the sphere's colour to be (0.1, 0.1, 0.1).
- Set the sphere's refractivity attribute to be true, with refractive coefficient as 0.8 and refractive index as needed.
- The reflectivity attribute is set to be true with a low reflective coefficient 0.1.

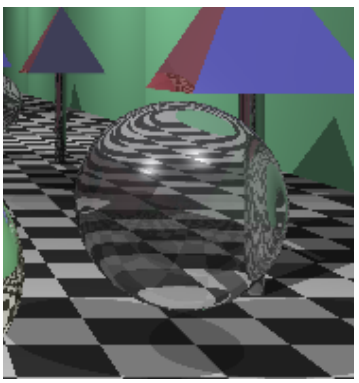
In trace():



- Create refractivity effect if an object is set refractive and step number is less than MAX_STEPS.
- Get the object's refractCoeff, refracIndex and set the eta, the ratio of indices of the refraction, to be 1 / refracIndex.
- Get the normal vector normalVec1 at the closest point of intersection ray.hit with the object's normal() method.
- Get the refracted direction refractedDir1 at ray.hit by glm::refract() with arguments ray.dir, normalVec1 and eta.
- Create the first refracted ray refractedRay1 with arguments ray.hit and refractedDir1.
- Get the closest intersection point of refractedRay1 with Ray.closestPt() method, and get the normal vector normalVec2 and the refracted direction refractedDir2 with normal vector -normalVec2 and 1/eta at this point, then create the second refracted ray refractedRay2 and get its closest intersection point.
- Get the refracted colour refractedColor at the closest intersection point of refractedRay2 by recursively using trace().
- Finally, the `color = color + (refracCoeff * refractedColor)`



eta = 1/1.5



eta = 1/1.01

2. Two light sources

File: SceneObject.cpp
In lighting():

- Reduce the intensity of light by setting lightColor to (0.45, 0.45, 0.45).
- Include lightColor value into the calculation of colorSum, so that
  colorSum = ambientTerm * color_ + lDotn * color_ * lightColor +
  specularTerm * glm::vec3(1) * lightColor

File: RayTracer.cpp
In trace():

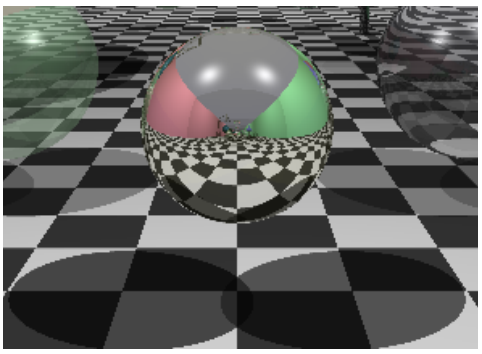
- Create two light positions: lightPos1(20, 30, -20) and lightPos2(-20, 30, -20).
- Create a variable as object's ambient term color :
  `glm::vec3 ambienTerm = glm::vec3 (0.2, 0.2, 0.2) * obj->getColor();`
- Apply the lighting effect of the two light sources to the object at the same time and subtract one ambienTerm.

```
color = obj->lighting(lightPos1, -ray.dir, ray.hit) +
        obj->lighting(lightPos2, -ray.dir, ray.hit) -
        ambienTerm;
```

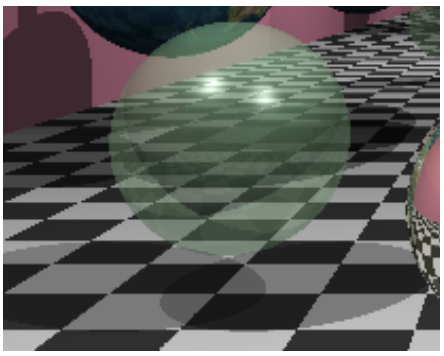
- In shadow settings, calculate the two light vectors lightVec1 and lightVec2 with lightPos1 and lightPos2, and the closest intersection point ray.hit. Create two shadow rays shadowRay1 and shadowRa2 with their light vectors and ray.hit. Get the close intersection points of shadowRay1 and shadowRay2 with Ray.closestPt().
- For both of the shadowRays, add shadow to color by multiplying the object's color with 0.4f if and only if shadowRay hits an object and the distance between the shadowRay's source and its closest interaction point is less than the distance between the shadowRay's source and the light source.
- If a point is in the shadow created by both light sources, the color at the point is set as ambientTerm colour.
- Check if the shadow is created by a transparent or refractive object by:

```
bool isTranspObj = sceneObjects[shadowRay.index]->isTransparent();
bool isRefractObj = sceneObjects[shadowRay.index]->isRefractive();
```

  If so, lighten the shadow colour.



Double specular highlights
Double shadows of solid sphere



Double specular highlights
Lighter shadows of transparent sphere

3. Anti-aliasing

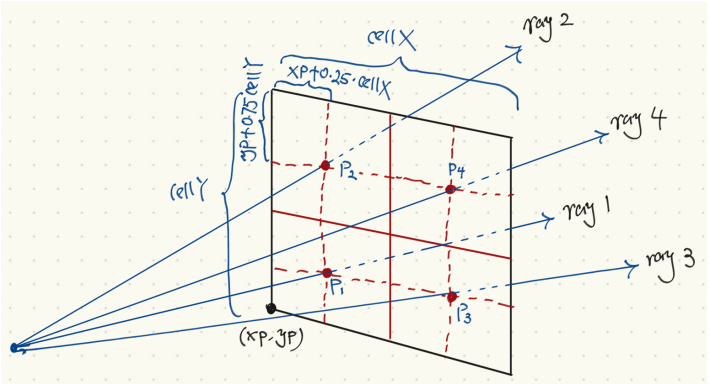
Anti-aliasing is implemented in this project by supersampling, in order to reduce the jaggedness along edges of objects and shadows.

File: RayTracer.cpp

- Create a boolean antiAliasing variable to switch on and off the antiAliasing effect.

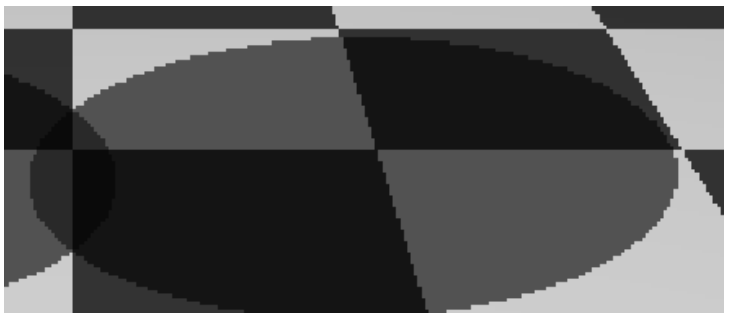
In display():

- If anti-aliasing is set true, divide each cell squad into four sub-cells, and primary rays are generated through the centres of each sub-cell.

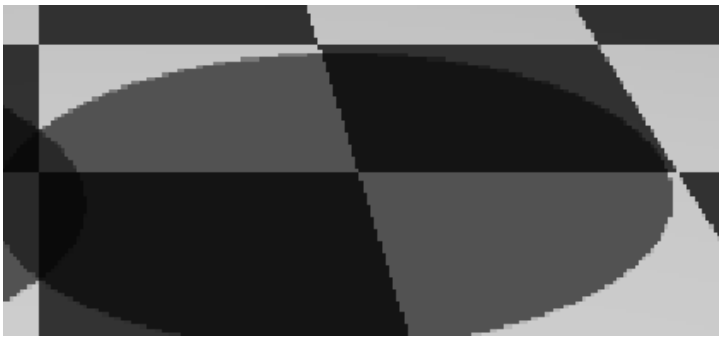


For example, to create ray2 through p2, we calculate the direction vector dir2 by
`glm::vec3 dir2(xp + 0.25 * cellX, yp + 0.75 * cellY, -EDIST);`
and then create: `Ray ray2 = Ray(eye, dir2);`

- Get the colour for each sub-cell with `trace(ray,1)`, and calculate the average value of the four colours as the colour of the whole cell.
- With NUMDIV = 600, rendering time with anti-aliasing off is 22s, and 90s with anti-aliasing on.



Anti-aliasing off



Anti-aliasing on

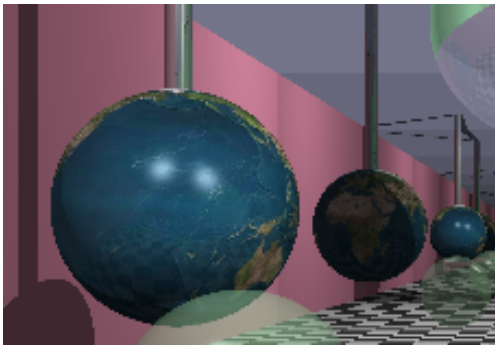
4. **Sphere textured using an image of earth**

File: RayTracer.cpp
In initialize():

- Create a sphere with object index 6, and set the sphere's reflectivity to be true with coefficient 0.05.

In trace():

- If the closest intersection point is on the sphere of index 6, apply texturing.
- Get the sphere centre and intersection point `p = ray.hit`.
- Calculate the unit vector from p to centre with `unitVec = glm::normalize(centre - po)`;
- Compute the texture coordinates with the following uv mapping formula:
  u = 0.5 + (arctan2(unitVec.x, unitVec)) / 2*PI
  v = 0.5 + arcsin(unitVec.y) / PI
  In the program we use <cmath>'s atan2 and asin functions:
  ```
  float texcoords = 0.5f + (atan2(unitVec[0], unitVec[2]) / (2.0 * PI));
  float texcoordt = 0.5f + (asin(unitVec[1]) / PI);
  ```
- Apply the texture coordinates by:
  ```
  color = texture.getColorAt(texcoords, texcoordt);
  obj->setColor(color);
  ```



Texturing Sphere

5. **Multiple reflections between 2 mirror-like walls**

File: RayTracer.cpp
In initialize():

- Create a room space with 6 planes, of which the ceiling position is set above all the light positions and the ceiling plane's lighting effect is disabled in trace(), so that its colour is not affected by lighting.
- Set the z value of the 4 vertices of the front wall plane to be 1 so that the eye with coordinates(0, 0, 0) is between the front wall and back wall planes.
- Set the colour of front wall and back wall planes to be (0, 0, 0) and their reflectivity to be true with coefficient 0.9, and disable their specularity.
- Set the MAX_STEPS to 50 to have more reflection levels. Refer to Image-1 on page 1 to see the effect.

6. Cylinders

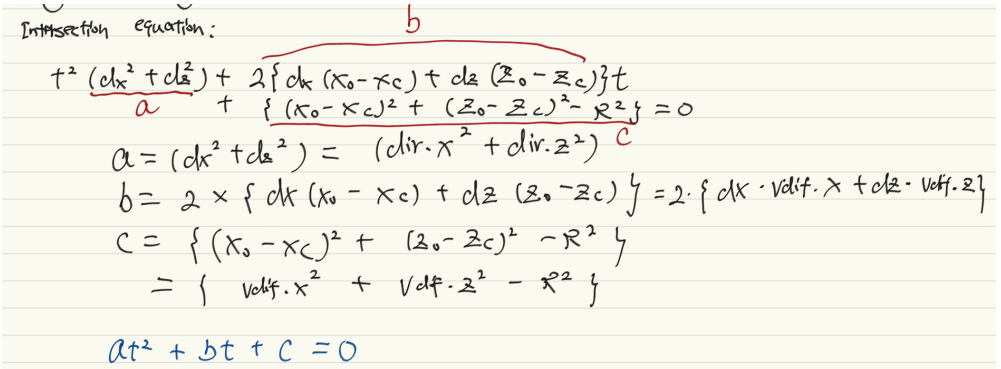
A Cylinder class inherited from SceneObject is created with Cylinder.h and Cylinder.cpp. Three private attributes center, radius and height are declared.

To implement normal():

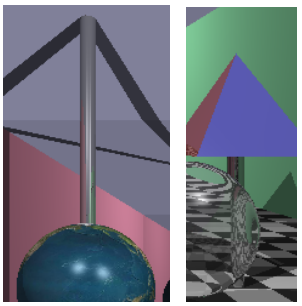
- Given a point p on the cylinder, get the the vector n from the center to p by:
  `glm::vec n = glm::vec3 (p[0]-center[0], 0, p[2]-center[2]);`
- Normalize n with the glm::normalize() method and return n.

To implement intersect():

- Rearrange the intersection equation (p40 Ray Tracing Slide) to be a general quadratic equation, and compute a, b, c.



- Using the discriminant $b^2 - 4ac$ to decide if variable t exists or not, and if so get the solutions t1, t2 with formula (-b±√(b²-4ac))/(2a), if not return -1.
- If t1 and t2 both are less than 0, return -1.
- To implement a cylinder with height:
  - Get the closest intersection point p1 and second closest point p2 by:
    glm::vec3 p1 = p0 + t1 * dir; glm::vec3 p2 = p0 + t2 * dir;
    and the max y value of points in cylinder: `float heightY = center.y + height`;
  - If p1.y is above center.y and lower than heightY, return t1. If not,
  - Check if p2.y above center.y and lower than heightY, if so return t2, if not,
  - Return -1.



2 Cylinders

Rendering Time

Settings: NUMDIV = 600, MAX_STEPS = 50
Time with anti-aliasing off: 22s
Time with anti-aliasing on: 90s

Building Commands

In a Linux system, unzip the mzh103_cosc363_assign2.zip file.
Step 1: Go to src folder and start a terminal at the location
Step 2: Run `cmake CMakeLists.txt`
Step 3: Run `make`
Step 4: Start the program with command `./RayTracer.out`

References:

1. Wikipedia contributors. (n.d.). UV mapping. Wikipedia. Retrieved June 4, 2022, from https://en.wikipedia.org/wiki/UV_mapping