COSC363 Assignment2 Ray Tracer Report

YUAN CUI

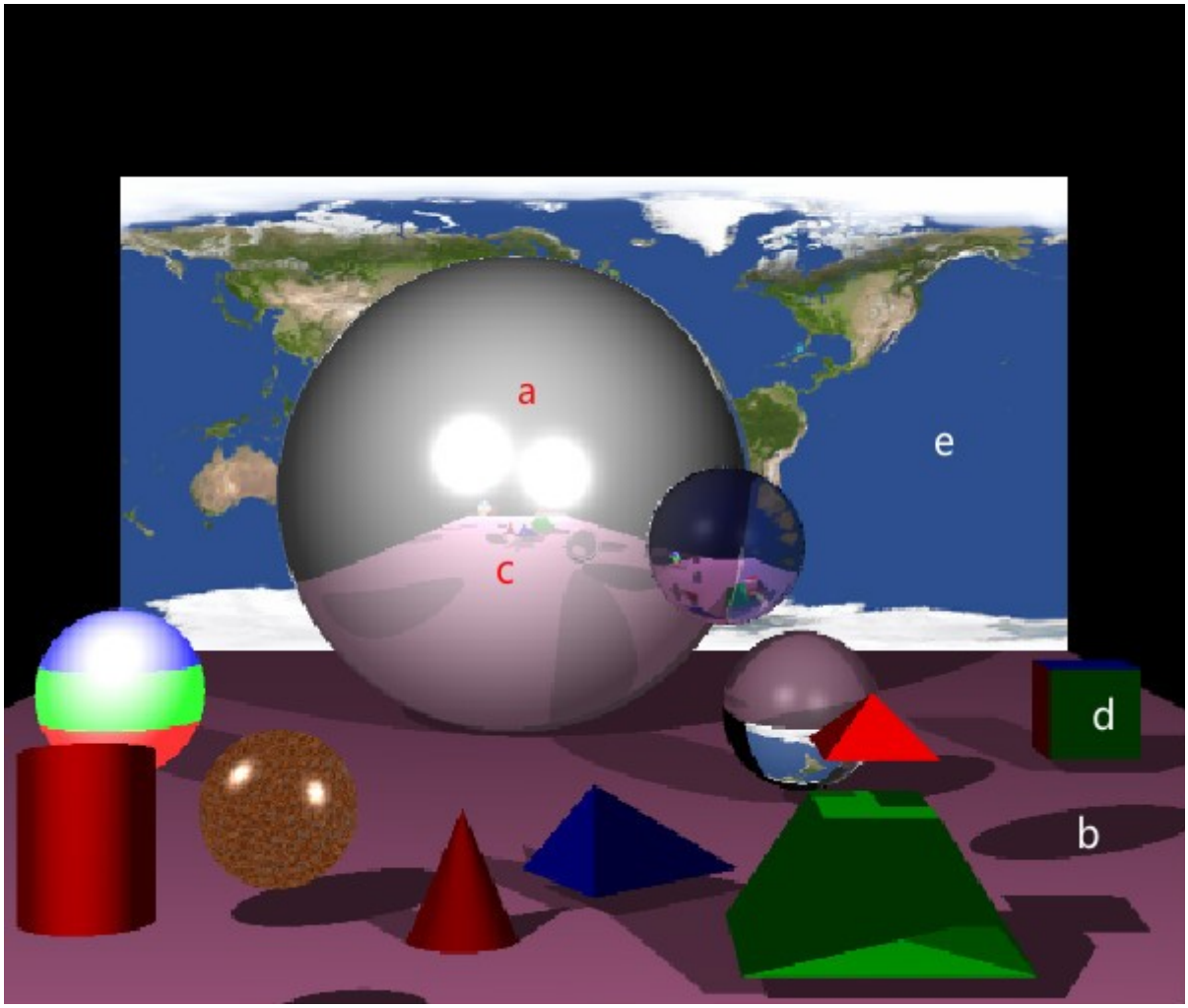
ycu20

63483319

Introduction

My ray tracer program need 10s-15s preparing time before properly shown on the screen of Erskine lab. There are 12 items and 2 light sources being created. It is not an easy work for this assignment, especially calculating the intersection of geometries. But I do finish all basic requirements and more than half extension requirements. I am sure I do learn and recall a lot of geometric knowledge. What an interesting experience with this assignment.

The Basic Ray Tracer



Result can be found from the figure above.

(a) Lighting

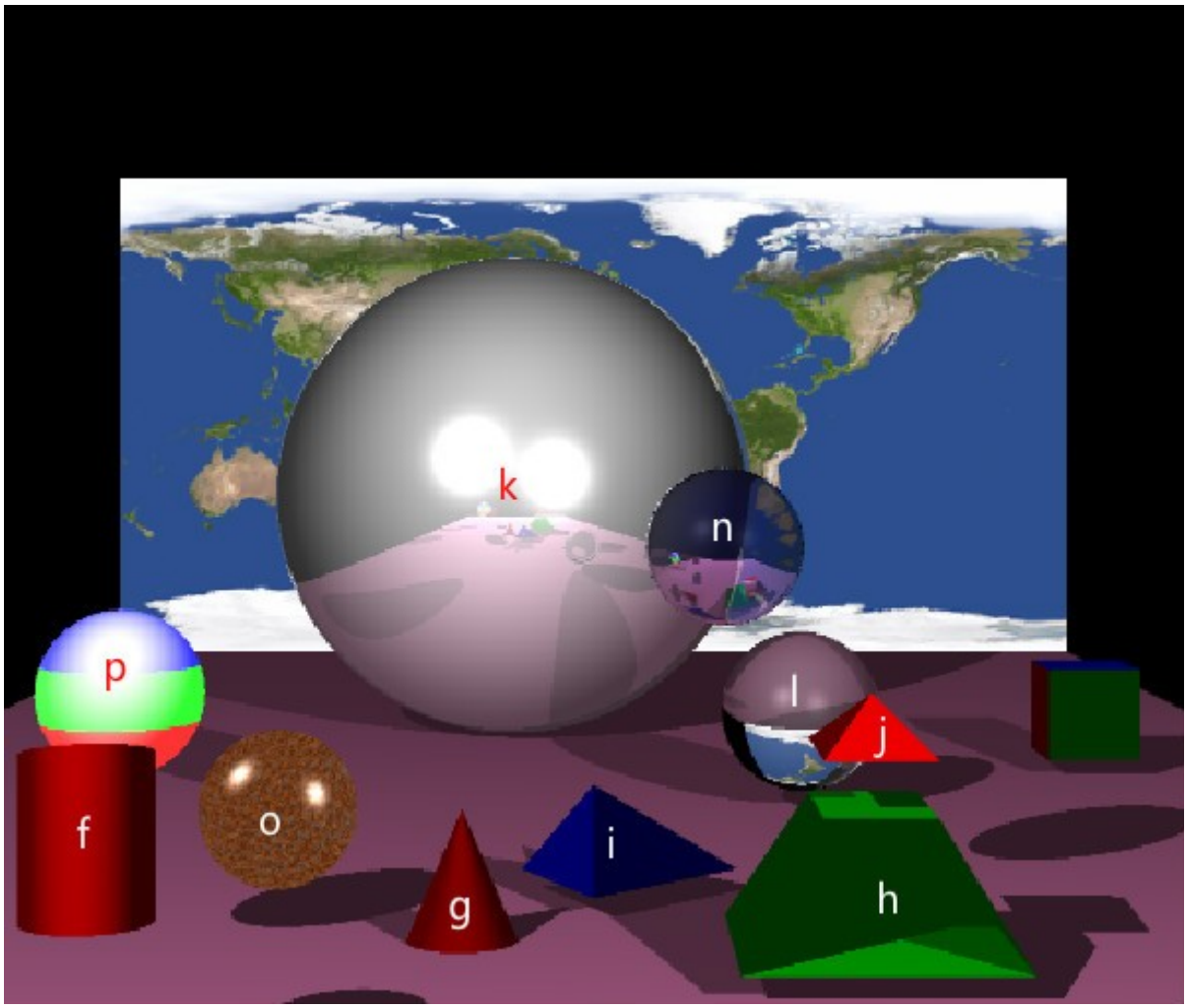
(b) Shadows

(c) Reflections

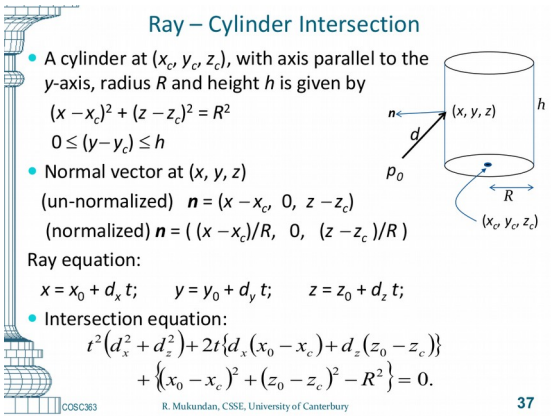
(d) Box

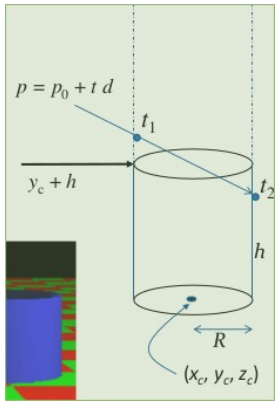
(e) Texture

Extensions



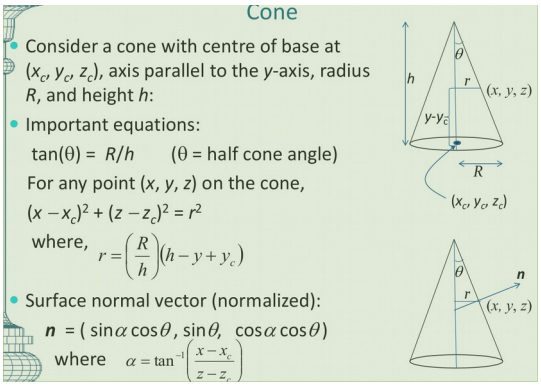
- (f) Cylinder

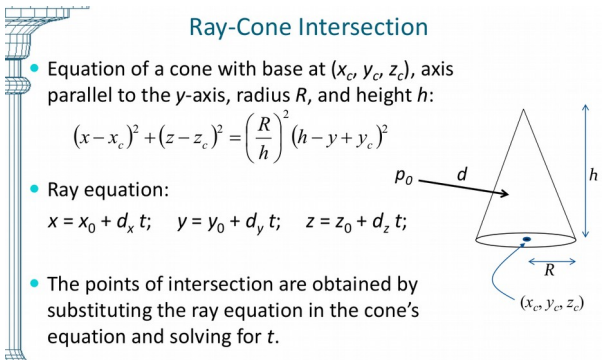




The formulars from slides were used for drawing a cylinder

- (g) Cone





Similarly, the formulars from slides were used for drawing a cone with a changable r value.

- (h) frustum
- (i) tetrahedron
- (j) squrepyyramid

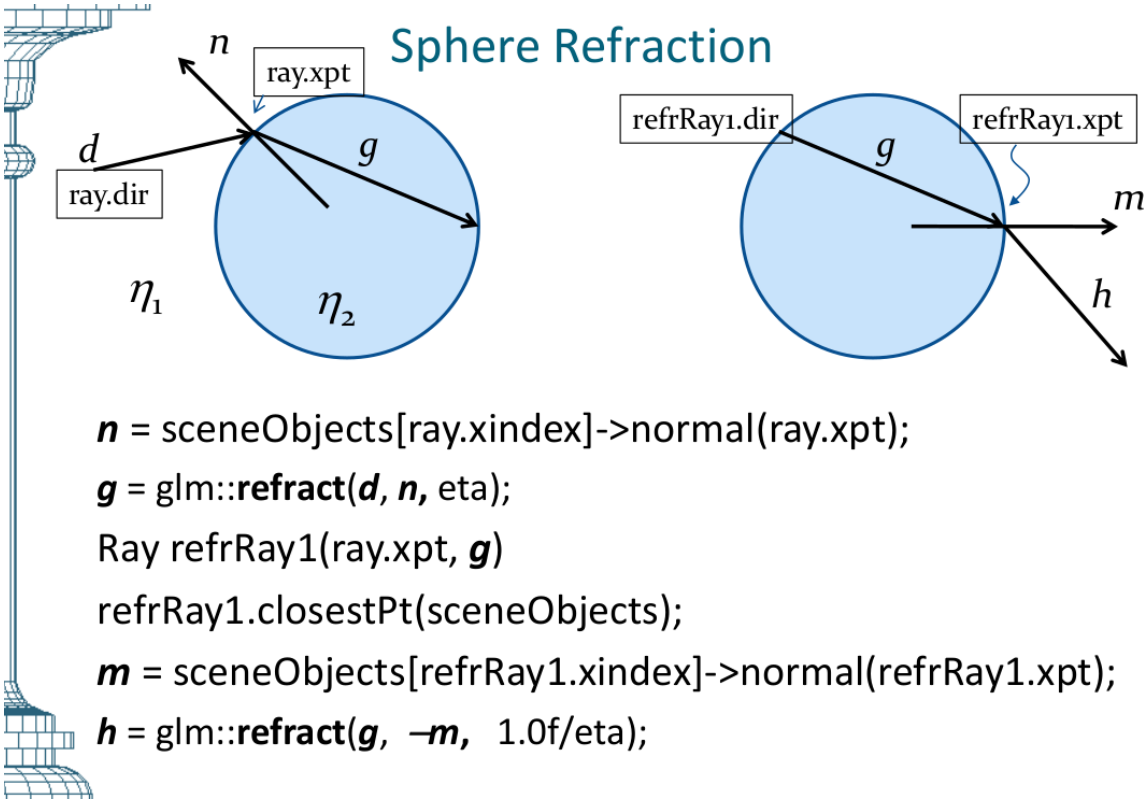
These three objects were drawn by different shaple of boards, quite simple way.

- (k) multiple lights

```
glm::vec3 materialCol = sceneObjects[ray.xindex]->getColor(); //else return object's colour
glm::vec3 normalVector = sceneObjects[ray.xindex]->normal(ray.xpt);
glm::vec3 lightVector = light - ray.xpt;
glm::vec3 lightVector1 = light1 - ray.xpt;
glm::vec3 lightVector_n = glm::normalize(lightVector);
glm::vec3 lightVector1_n = glm::normalize(lightVector1);
glm::vec3 reflVector = glm::reflect(-lightVector_n, normalVector);
glm::vec3 reflVector1 = glm::reflect(-lightVector1_n, normalVector);
Ray shadow(ray.xpt, lightVector_n);
Ray shadow1(ray.xpt, lightVector1_n);
shadow.closestPt(sceneObjects);
shadow1.closestPt(sceneObjects);

float lDotn = glm::dot(lightVector_n, normalVector);
float lDotn1 = glm::dot(lightVector1_n, normalVector);
float rDotv = glm::dot(reflVector, -ray.dir);
float rDotv1 = glm::dot(reflVector1, -ray.dir);
// no specular color from floor
specCol = rDotv >= 0 && ray.xindex !=3 && ray.xindex !=8? glm::vec3(1, 1, 1) * pow(rDotv, f_value) : glm::v
specCol1 = rDotv1 >= 0  && ray.xindex !=3 && ray.xindex !=8? glm::vec3(1, 1, 1) * pow(rDotv1, f_value) : gl
sumCol = (lDotn <= 0 || shadow.xindex > -1 || shadow.xdist >= glm::length(lightVector)) ? ambientCol * mate
sumCol = (lDotn1 <= 0 || shadow1.xindex > -1 || shadow1.xdist >= glm::length(lightVector1)) ? ambientCol *
```

Set two lightVectors, two reflVector, two shadowRays ...

- (l) refraction

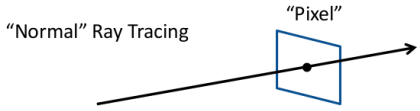


Use formulas from booklets, calculate ray in and out.

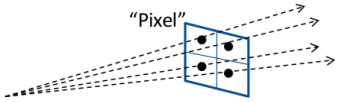
- (m) anti-aliasing

Anti-Aliasing

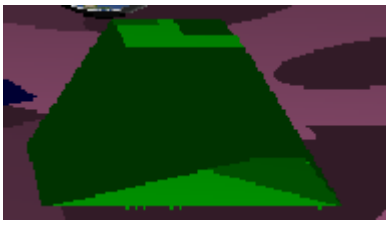
The ray tracing algorithm samples the light field using a finite set of rays generated through a discretized image space. This results in distortion artefacts such as jaggedness along edges of polygons and shadows.



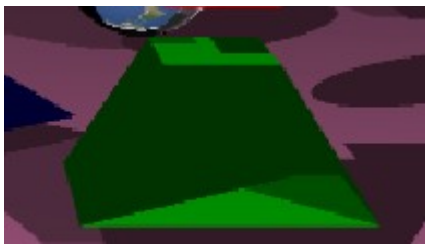
- Supersampling: Generate several rays through each square pixel (eg. divide the pixel into four equal segments) and compute the average of the colour values.



before:



after:



Calculate 4 surrounding pixels and use the averge color as the color value

- (n) transparent
  Combine with refraction and reflection , set eta = 1.005
- (o) texture on non-planar object

```
//texture on sphere
if(ray.xindex == 1)
{
    float a = asin(normalVector.x) / M_PI + 0.5;
    float b = asin(normalVector.y) / M_PI + 0.5;
    sumCol = texture1.getColorAt(a, b)+ specCol + specCol1; // add two specular spots
}
```

return color at texture coord (a,b), where a, b are both in [0,1]

- (p) A non-planar object textured using a procedural pattern

```
if(ray.xindex == 8)
{
    if (ray.xpt.y >= -13 and ray.xpt.y < -11) sumCol += red * 0.8f;
    if (ray.xpt.y >= -11 and ray.xpt.y < -9) sumCol += green * 0.8f;
    if (ray.xpt.y >= -9 and ray.xpt.y <= -7) sumCol += blue * 0.8f;
}
```

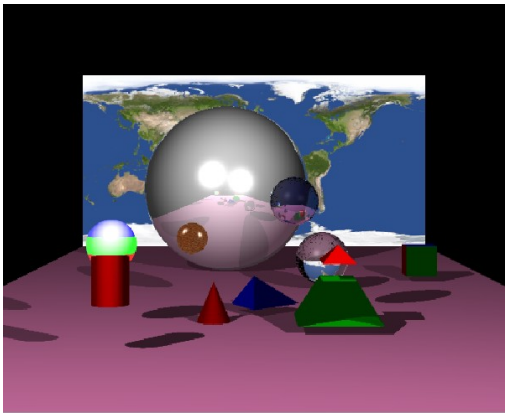
Give different color by different y values.

- (q) camera motion

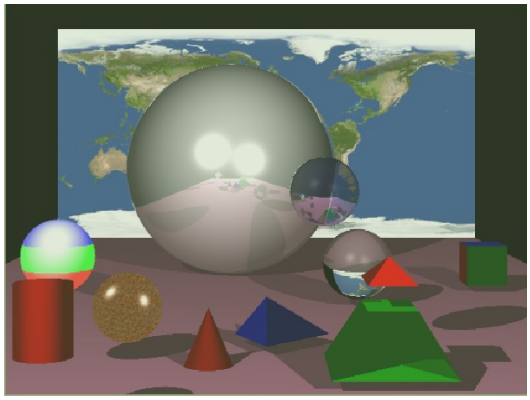
```
void special(int key, int x, int y)
{
   if(key == GLUT_KEY_LEFT) lookAngle -= 0.1;  //Change direction
   else if(key == GLUT_KEY_RIGHT) lookAngle += 0.1;
   else if(key == GLUT_KEY_DOWN)
   { //Move backward
       eye_x -= 5*sin(lookAngle);
       eye_z += 5*cos(lookAngle);
   }
   else if(key == GLUT_KEY_UP)
   { //Move forward
       eye_x += 5*sin(lookAngle);
       eye_z -= 5*cos(lookAngle);
   }
   //shift view


   glutPostRedisplay();
```

Similar as assignment 1, change eye location.

Backward:



Original:



Overall

- Build Command: g++ -Wall -o "%e" "%f" Sphere.cpp SceneObject.cpp Ray.cpp Plane.cpp Cylinder.cpp Cone.cpp TextureBMP.cpp -lGL -lGLU -lglut

I really like this course, which gives me such lots fo fun. I think I can finish more features if I can get more time. Thanks for your time.

😆