实验报告 5

姓名：蒋卓洋
学号：59119125

1、实现

1.1. Render()
(1)位置：Renderer.cpp
(2)实现：

```
1.  void Renderer::Render(const Scene& scene)
2.  {
3.      std::vector<Vector3f> framebuffer(scene.width * scene.height);
4.
5.      float scale = std::tan(deg2rad(scene.fov * 0.5f));
6.      float imageAspectRatio = scene.width / (float)scene.height;
7.
8.      // Use this variable as the eye position to start your rays.
9.      Vector3f eye_pos(0);
10.     int m = 0;
11.     for (int j = 0; j < scene.height; ++j)
12.     {
13.         for (int i = 0; i < scene.width; ++i)
14.         {
15.             // generate primary ray direction
16.             float x;
17.             float y;
18.             // TODO: Find the x and y positions of the current pixel to get the direction
19.             // vector that passes through it.
20.             // Also, don't forget to multiply both of them with the variable *scale*, and
21.             // x (horizontal) variable with the *imageAspectRatio*
22.
23.             /////////////////Solution/////////////////////
24.             ////Name:JiangZhuoyang
25.             ////StudentID:58119125
26.             ////FinishDate:21/11/05
27.
28.             //We must find the relation of coordinate between raster space and world space
29.             //1.Raster space to NDC space
30.             float x_ndc, y_ndc;
```

```
            x_ndc = (float(i)+0.5) / scene.width;//"+0.5"means the ray will trans through the c
        enter
            y_ndc = (float(j)+0.5) / scene.height;
            //2.Move to the screen
            float x_scr, y_scr;
            x_scr = (2*x_ndc - 1) * imageAspectRatio;//screen adaptation.
            y_scr =-(2*y_ndc - 1);//"-"means the defination and application has a little differe
        nce
            //3.Think about the scale (there is a distance between camera and screen)
            x = x_scr * scale;
            y = y_scr * scale;


            ////////////////////////////////////////////

            Vector3f dir = Vector3f(x, y, -1); // Don't forget to normalize this direction!
            framebuffer[m++] = castRay(eye_pos, dir, scene, 0);
        }
        UpdateProgress(j / (float)scene.height);
    }
```

1.2. rayTriangleIntersect()

(1)位置：Triangle.hpp

(2)实现：

```
bool rayTriangleIntersect(const Vector3f& v0, const Vector3f& v1, const Vector3f& v2, co
nst Vector3f& orig,
                          const Vector3f& dir, float& tnear, float& u, float& v)
{
    // TODO: Implement this function that tests whether the triangle
    // that's specified bt v0, v1 and v2 intersects with the ray (whose
    // origin is *orig* and direction is *dir*)
    // Also don't forget to update tnear, u and v.

    /////////////////Solution//////////////////
    ////Name:JiangZhuoyang
    ////StudentID:58119125
    ////FinishDate:21/11/05

    //We need to complete the Moller Trumbore Algorithm
    //1.Define needed vectors of the algorithm
    Vector3f e1 = v1 - v0;
    Vector3f e2 = v2 - v0;
```

```
18.    Vector3f s = orig - v0;
19.    Vector3f s1 = crossProduct(dir,e2);
20.    Vector3f s2 = crossProduct(s,e1);
21.
22.    //2.Construct the Linear Function
23.    tnear = dotProduct(s2,e2) / dotProduct(s1,e1);
24.    u = dotProduct(s1,s) / dotProduct(s1,e1);
25.    v = dotProduct(s2,dir) / dotProduct(s1,e1);
26.
27.    //3.Do the judgement:
28.    if(tnear > 0 && u > 0 && v > 0 && (1.0f - u - v) > 0){
29.        return true;
30.    }
31.    ////////////////////////////////////////////
32.    return false;
33. }
```

2、结果

- 实验结果如下：

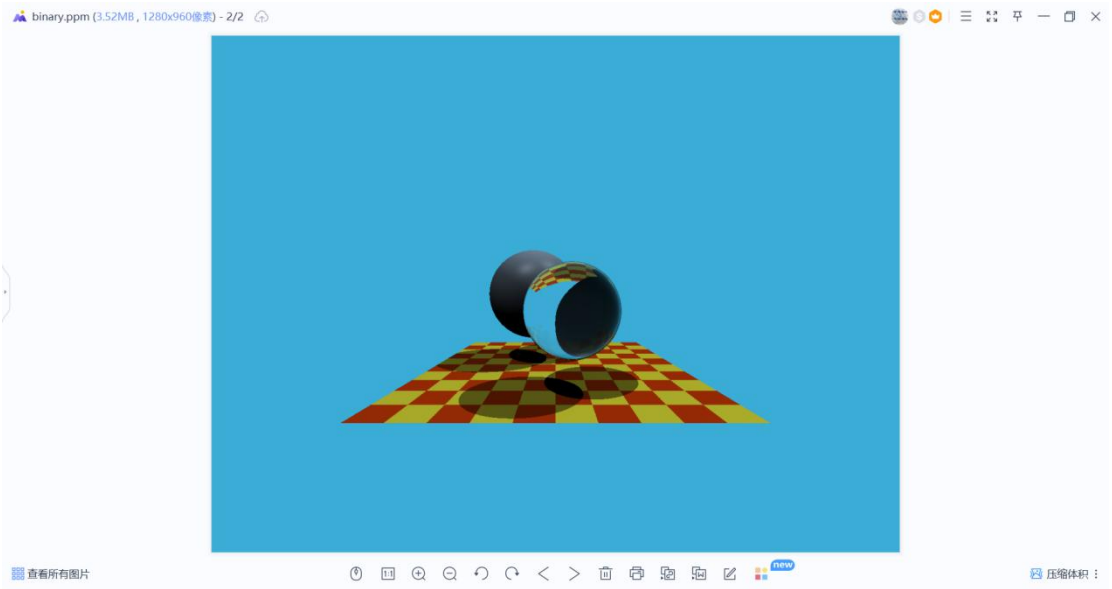

图 1. 实验结果