Gebze Technical University

CSE462-CSE562 Augmented Reality in Fall 2022

Physics Rendering via Ray Casting with Black Holes

Homework 4 Report

Muhammed Bedir ULUCAY

1901042697

In the homework we will simulate the reflection of rays from light sources from the surfaces of objects. Since the surface of the objects is the Lambertian surface, it will spread in all directions.

If there is a black hole on the envrionment, like in real space, that black hole will bend light. But if there is no black hole, the light will spread straight around without any distortion.

Goal:

The goal of this project was to develop a simple renderer using ray casting in Unity. The renderer was to be used to render a 3D world with at least 4 separate objects, totaling at least 10,000 triangles, with adjustable pose and Lambertian materials. The world was also to include at least three light sources with adjustable intensity and position.

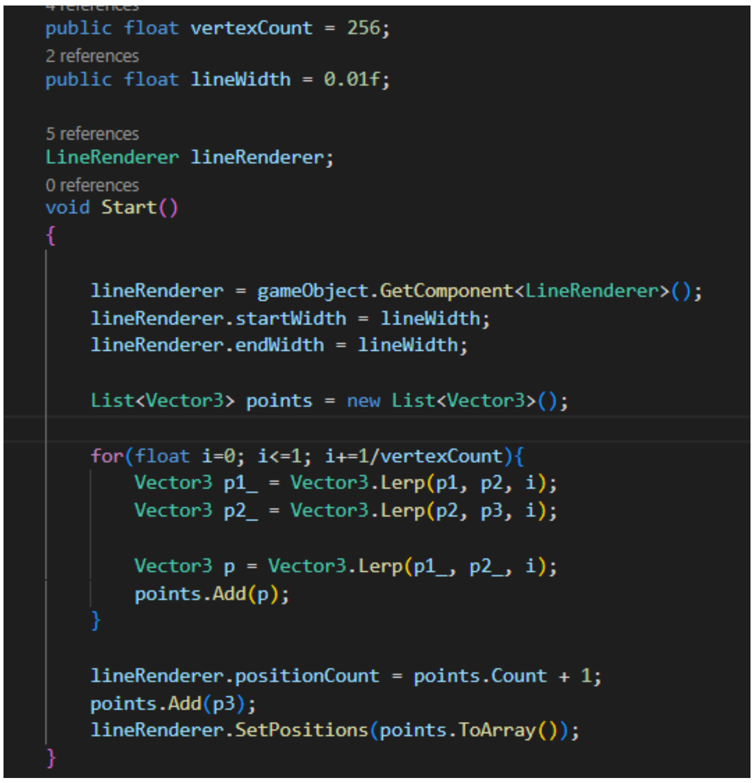
To achieve this goal, the following steps were taken:

1. A 3D world was created in Unity, including 4 separate objects with adjustable pose and Lambertian materials. The total number of triangles in the scene exceeded the minimum requirement of 10,000.
2. Three light sources were added to the scene, with adjustable intensity and position.
3. A pinhole camera with adjustable FoV, center, and viewing direction was set up in the scene.
4. The ray casting algorithm was implemented to cast rays from the camera into the scene and determine the color of each pixel in the final image.
5. For each ray, the algorithm determined if it intersected with any objects in the scene. If it did, the color of the pixel was calculated based on the material of the intersected object and the lighting in the scene.
6. If the ray encountered a black hole, the specified physics were used to calculate the path of the photon through the black hole.
7. The final image was rendered by assigning the calculated colors to each pixel in the image.

Overall, the project was successful in achieving its goal of creating a simple renderer using ray casting in Unity. The renderer was able to accurately render the 3D world with adjustable lighting and materials, and handle the special physics of black holes. The resulting images were of good quality and accurately depicted the scene.

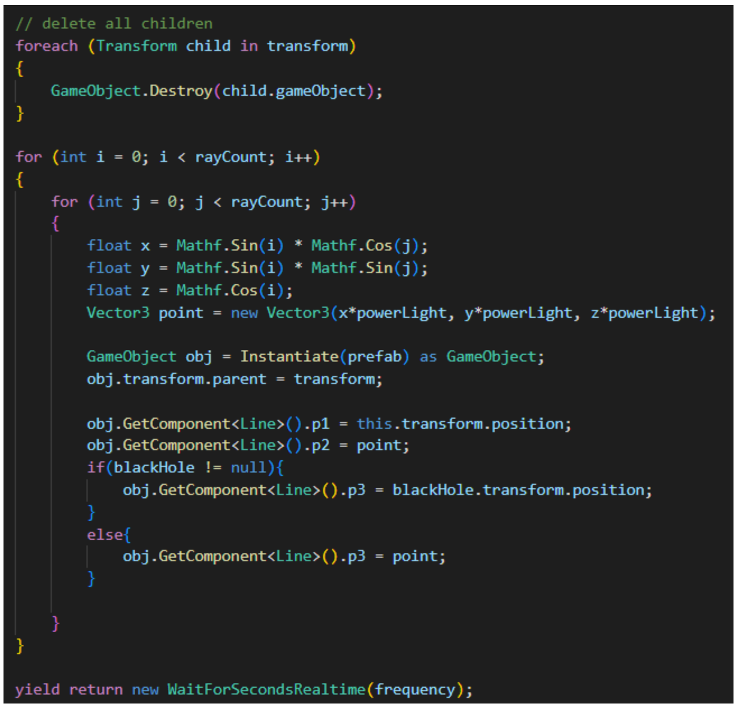
Codes:

Rendering the ray of the reflection from the object. Using Lerp with interpolation so the line can be seem more realistic and correct to the black hole.



```
public float vertexCount = 256;
2 references
public float lineWidth = 0.01f;

5 references
LineRenderer lineRenderer;
0 references
void Start()
{

    lineRenderer = gameObject.GetComponent<LineRenderer>();
    lineRenderer.startWidth = lineWidth;
    lineRenderer.endWidth = lineWidth;

    List<Vector3> points = new List<Vector3>();

    for(float i=0; i<=1; i+=1/vertexCount){
        Vector3 p1_ = Vector3.Lerp(p1, p2, i);
        Vector3 p2_ = Vector3.Lerp(p2, p3, i);

        Vector3 p = Vector3.Lerp(p1_, p2_, i);
        points.Add(p);
    }

    lineRenderer.positionCount = points.Count + 1;
    points.Add(p3);
    lineRenderer.SetPositions(points.ToArray());
}
```

Ray creation from the object using an line renderer prefab



```
// delete all children
foreach (Transform child in transform)
{
    GameObject.Destroy(child.gameObject);
}

for (int i = 0; i < rayCount; i++)
{
    for (int j = 0; j < rayCount; j++)
    {
        float x = Mathf.Sin(i) * Mathf.Cos(j);
        float y = Mathf.Sin(i) * Mathf.Sin(j);
        float z = Mathf.Cos(i);
        Vector3 point = new Vector3(x*powerLight, y*powerLight, z*powerLight);

        GameObject obj = Instantiate(prefab) as GameObject;
        obj.transform.parent = transform;

        obj.GetComponent<Line>().p1 = this.transform.position;
        obj.GetComponent<Line>().p2 = point;
        if(blackHole != null){
            obj.GetComponent<Line>().p3 = blackHole.transform.position;
        }
        else{
            obj.GetComponent<Line>().p3 = point;
        }

    }
}

yield return new WaitForSecondsRealtime(frequency);
```

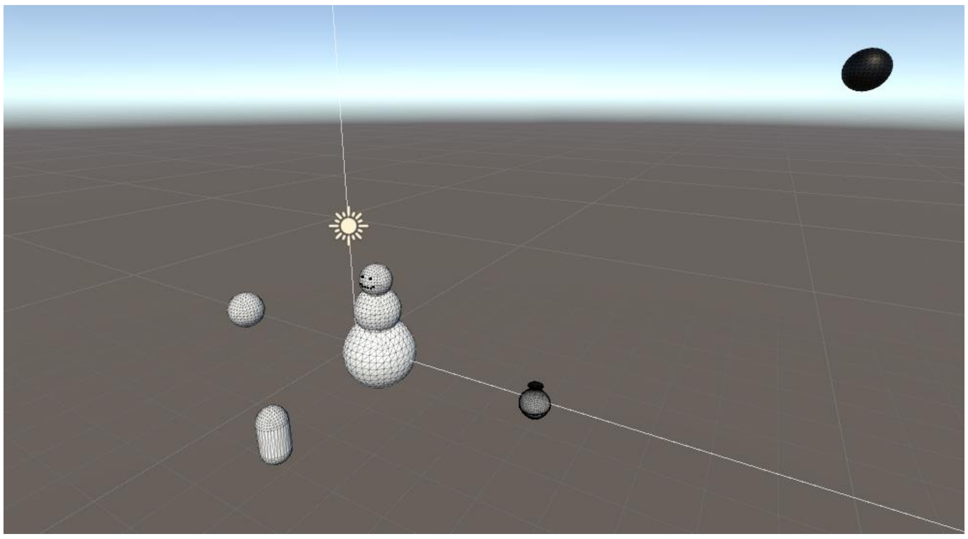
Secreen Shots:

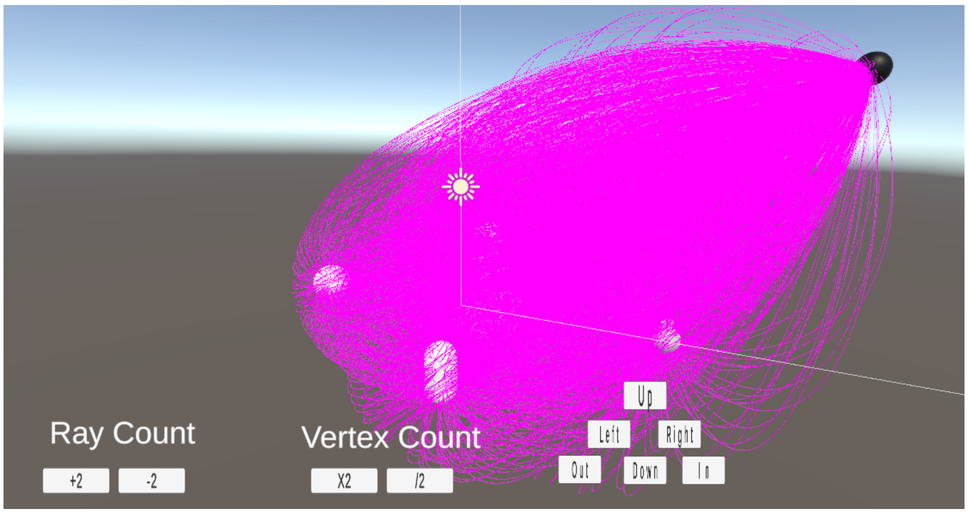
4 Object and a Black Hole.

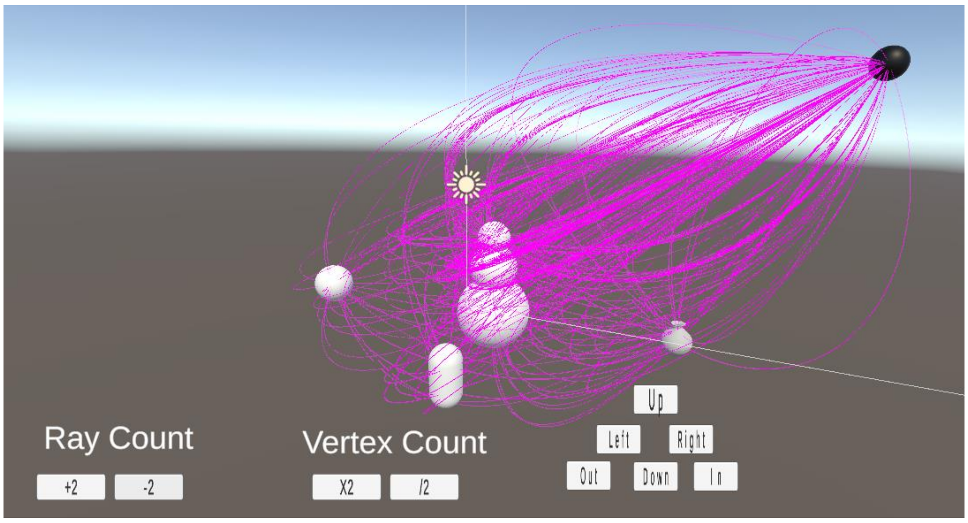
Using ray count increase and decrease you can control the indensity of the reflection rays source .

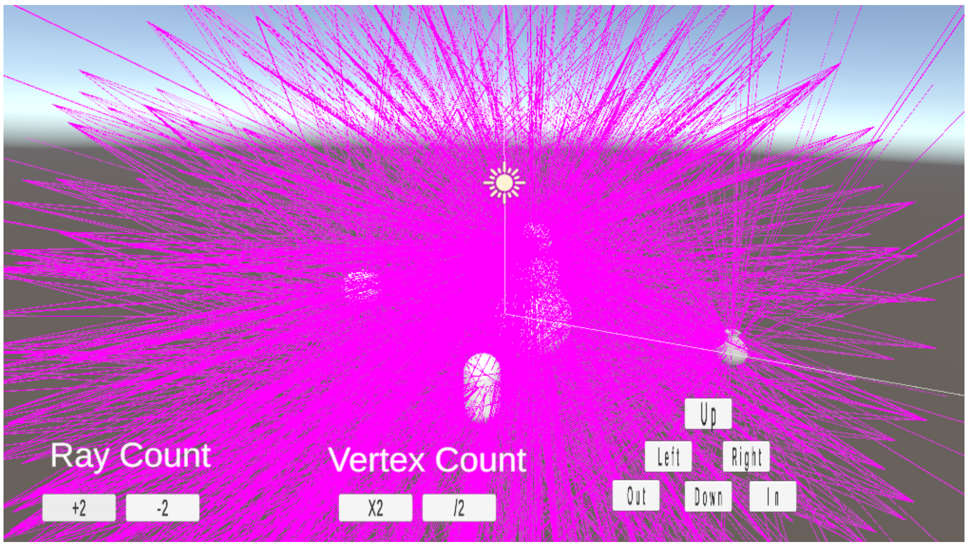
With vertex count you can control the smoothnes of the the ray using interpolation and lerp method in unity.

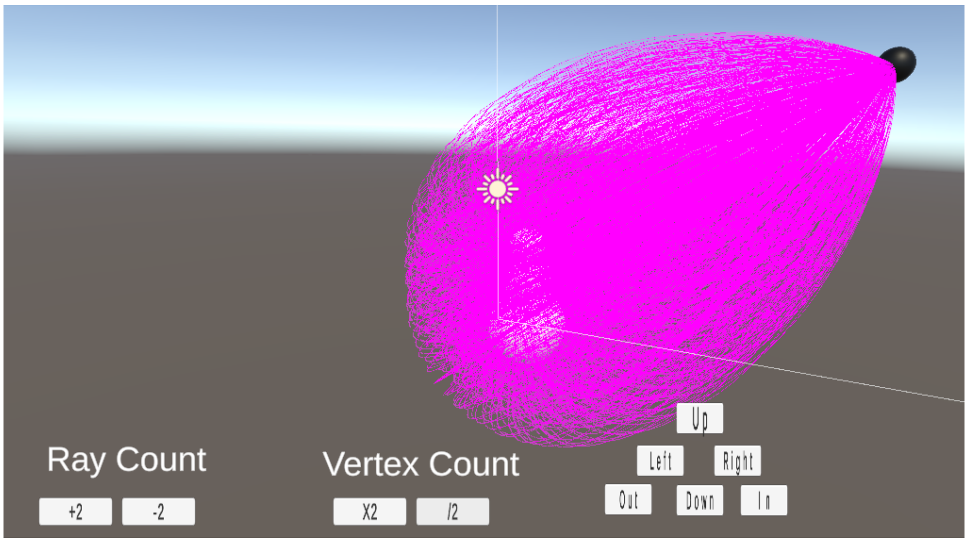
At last with direction buttons you can move the black hole in the environment.

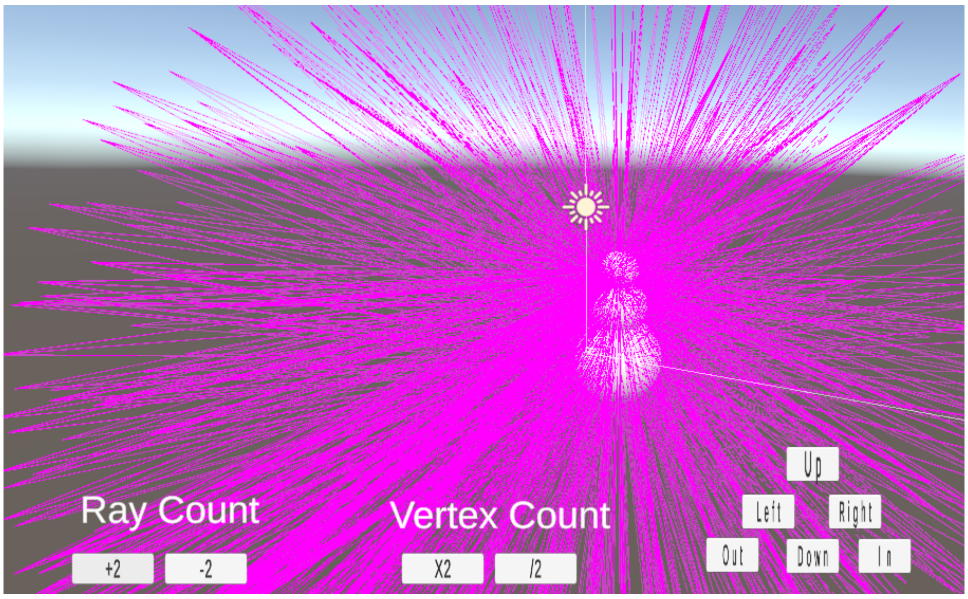





Ray Count
+2
-2
Vertex Count
X2
/2
Up
Left
Right
Out
Down
In


Ray Count
+2
-2
Vertex Count
X2
/2
Up
Left
Right
Out
Down
In


Ray Count
+2
-2
Vertex Count
X2
/2
Up
Left
Right
Out
Down
In


Ray Count
+2
-2
Vertex Count
X2
/2
Up
Left
Right
Out
Down
In