CG_6_Shader Programming Report

PB21010479 Caoliwen Wang

April 14, 2024

1 Problem Statement

We want to implement local rendering with the Blinn-Phong shading model, and to enhance the realism of the scene, we also want to add shadows using the Shadow Mapping algorithm. By implementing these two algorithms, we aim to achieve a preliminary rendering model.

2 Propose Algorithms

2.1 Blinn-Phong Shading Model

The core expression of the Blinn-Phong shading model, as shown in Equation 2.1, is what we aim to implement first.

$$I = k_a I_a + k_d I_d \cos\alpha + k_s I_s \cos^k \theta. \tag{2.1}$$

Where θ represents the angle between the reflected light and the viewing direction, and α represents the angle between the incident light and the surface normal. The remaining coefficients are defined as follows:

1. k_a represents the ambient coefficient, with $k_a = 0.2$ in this case.
2. k_s represents the specular reflection coefficient, with $k_s = metallic * 0.8$.
3. k_d represents the diffuse reflection coefficient, with $k_d = 1 - k_s$.
4. $k = (1 - roughness) * 0.2$.

For multiple light sources, we can simply accumulate the contributions from each light source.

2.2 Shadow Mapping

If there is occlusion between the shading point and the light source, naturally the shading point should appear as "shadowed", and the intensity of light at this point should only come from ambient light. This can significantly reduce computational complexity and make the scene more realistic.

The core idea of this algorithm is that if the actual distance from the shading point to the light source is greater than the depth obtained by capturing the scene from the light source's

perspective, it indicates that the shading point is occluded and thus becomes shadowed. We only need to use one condition to implement this problem.

2.3 Normal Mapping

In our scene, polygonal objects are abundant, each possibly composed of hundreds or thousands of flat triangles. To enhance realism and conceal the fact that these objects are made up of numerous triangles, we apply textures to the triangles to add extra details. While textures are beneficial, the fact that objects are composed of many triangles becomes apparent upon close inspection. In reality, object surfaces are not perfectly flat but exhibit countless details and irregularities. Normal maps preserve the characteristics of each fragment, allowing for significant enhancements in areas with details.

Since the normal map stores normals in tangent space, it needs to be transformed into world coordinates using a transformation matrix. This transformation matrix is called the tangent-to-world matrix and is composed of the tangent, bitangent, and normal vectors at the point of interest. These three vectors are mutually orthogonal unit vectors. The specific process is as follows:

```
1 normal = mat3 (tangent, bitangent, normal)*(2*normalmap_value-1);
```

3 Programming

3.1 Node programming

In this assignment, we utilized a new framework. For Render Nodes, we adopted the connection method as shown in Figure 1.

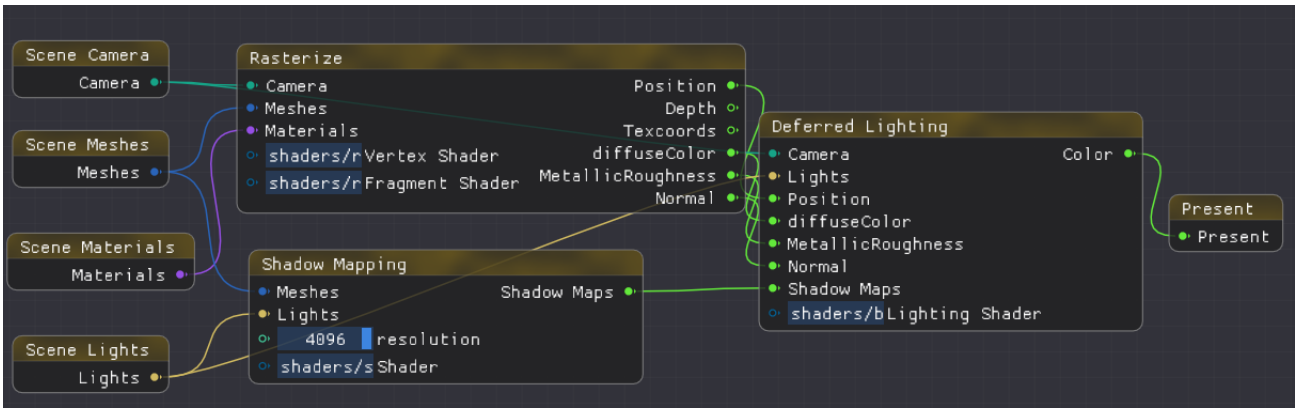

Figure 1: Render Nodes

3.2 Code Optimization

We calculate the diffusion part using the following code:

```
// diffuse color
vec3 norm = normalize(normal);
vec3 lightDir = normalize(lights[i].position - pos);
float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = lights[i].color * (1-metal * 0.8) * diff;
```

We calculate the specular part using the following code:

```
// specular color
vec3 viewDir = normalize(camPos - pos);
vec3 reflectDir = reflect(-lightDir, norm);
float spec =
pow(max(dot(viewDir, reflectDir), 0.0), (1-roughness)*2);
vec3 specular = lights[i].color * metal * 0.8 * spec;
```

The method for calculating the actual distance during Shadow Mapping is as follows:

```
 // light depth
vec4 temp = lights[i].light_projection *
lights[i].light_view * (vec4(pos, 1.0));
vec3 light_uv = ( temp.xyz / temp.w );

float shadow_map_value = texture(shadow_maps,
vec3(0.5*light_uv.xy+0.5, lights[i].shadow_map_id)).x;
```

According to the definition, the method for the judgement is as follows:

```
if (light_uv.z > shadow_map_value+0.01)
{
    result = ambient;
}
else
```

4 Experimental Results

I think the results of this algorithm are fantastic.

We perform testing using the box_on_plane method. The result is as shown in the Figure 2.

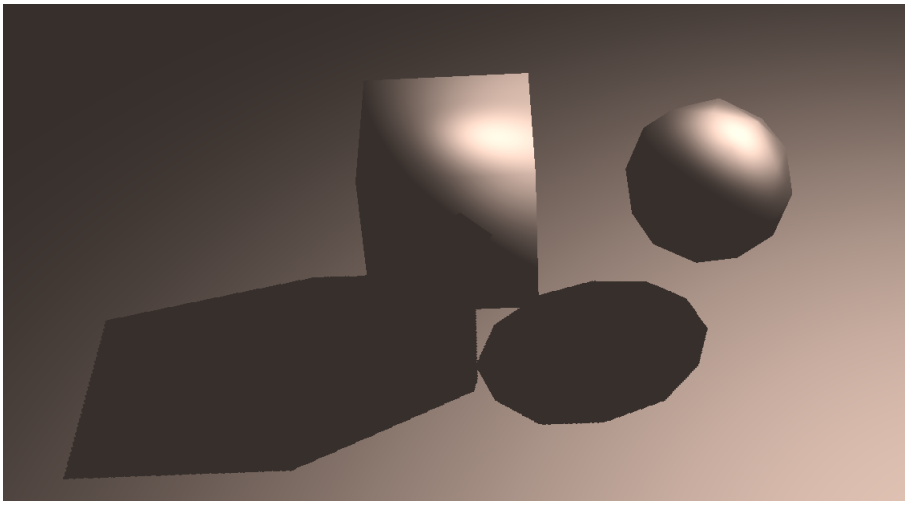

Figure 2: Preliminary Exploration Of the Results

We can observe that both the shadows and highlights appear reasonably accurate.

The testing for multiple light sources is also reasonable. It is shown in Figure 3.

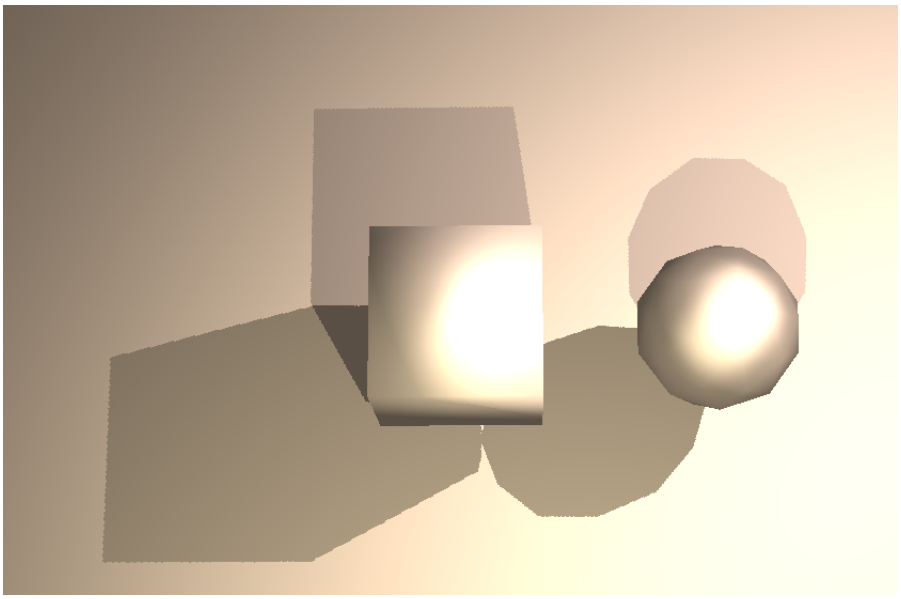

Figure 3: Multiple Light Sources

5 Summary

This experiment has completed the most basic rendering model, including local rendering and the Shadow Mapping algorithm. However, there are still aspects that can be optimized, such as:

1. We can find further approaches to implement soft shadows;

2. In the corners, aliasing artifacts may still occur.