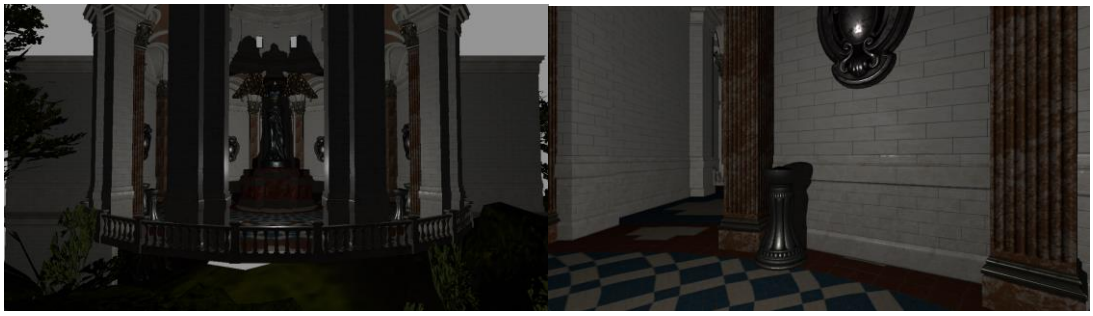


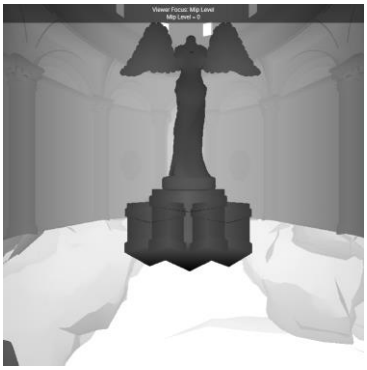
1.1Shadow Mapping

Render Pass Settings:

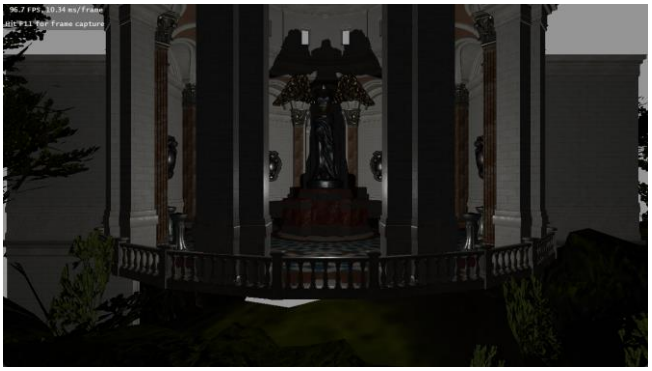
In this work, I employ two render passes to implement shadow mapping within the scope of computer graphics. The first render pass generates the shadow map from the perspective of the light source, while the second render pass, a forward rendering pass, is responsible for scene calculation and lighting computations. Given the two-pass process, two graphics pipelines and corresponding shaders are employed. The shader for the shadow map pipeline is intentionally left empty.

Shadow Map Resolutions:

Initially, I employed a default resolution of 1024x1024 for the shadow map. This provided satisfactory results with no apparent aliasing. To explore the impact of resolution, I experimented with 2048x2048 and 4096x4096, finding smoother shadow maps, but no discernible difference from the 1024x1024 case. However, when the resolution was reduced to 512x512, visible aliasing appeared in the shadow map and the scene. In this context, it's evident that the resolution of a shadow map represents a compromise between performance and rendering quality. Therefore, 1024x1024 was retained as the default resolution.

<table>
<tr>
<td> 
R = 1024 shadow map</td>
<td> 
R = 1024 scene</td>
</tr>
<tr>
<td> 
R = 2048 shadow map</td>
<td> 
R = 2048 scene</td>
</tr>
</table>

R = 4096 shadow map



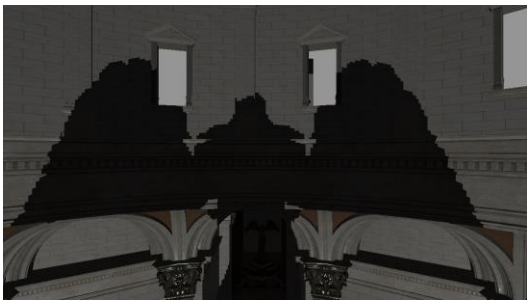
R = 4096 scene

R = 512 shadow map

R = 512 scene

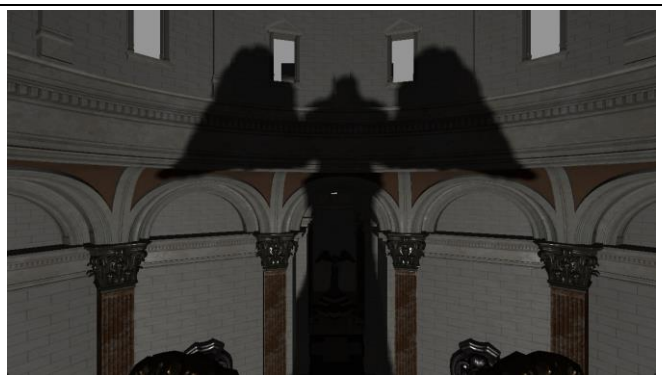
PCF:

PCF is a technique employed to sample an area of pixels from the shadow map, similar to the post-processing observed in CW03 (Bloom). This blurs the shadow, leading to a smoother appearance. Comparing scenarios without PCF, with 2x2 PCF, and 4x4 PCF, I noted that without PCF, shadow edges exhibited significant aliasing. The 2x2 PCF improved this by blurring the shadow edges. However, the 4x4 PCF was my preference as it blurs the edge and provides a sense of depth variation, lightening the color at the shadow edges for a more natural look from a distance.



Without PCF

With 2*2 PCF



With 4*4 PCF

Upon closer inspection, the 4x4 PCF revealed significant blurring, a phenomenon also presents with the 2x2 PCF but to a lesser extent. Upon increasing the resolution from 1024 to 2048 and then to 4096, this issue was mitigated and finally eliminated. Thus, I concluded that the PCF sampling area is intrinsically tied to the resolution of the shadow map: the higher the resolution, the more precise the PCF sampling.

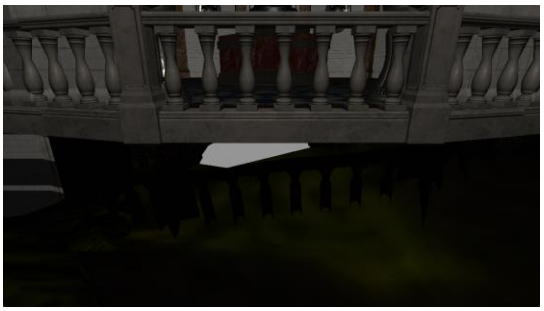
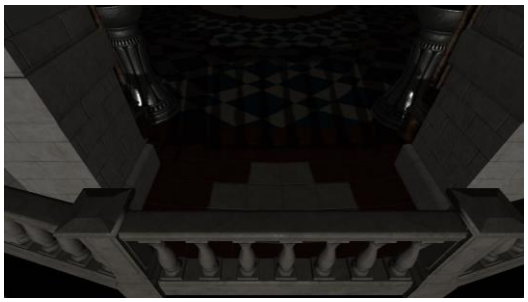
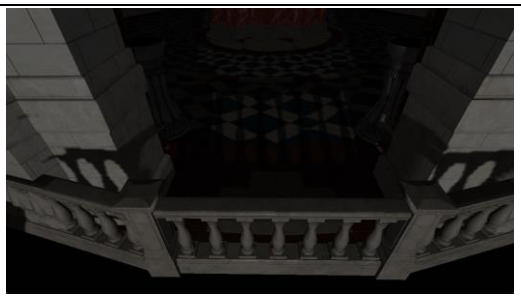
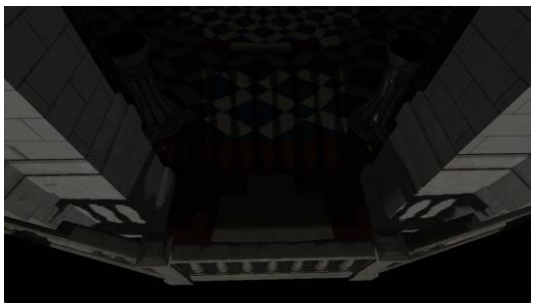
4*4 PCF with resolution 1024 * 1024

4*4 PCF with resolution 2048 * 2048

4*4 PCF with resolution 4096 * 4096

Bias:

Interestingly, I observed that shadow acne disappeared in my implementation when bias was set to zero. However, Peter Panning occurred when the bias was overly high. For example, with bias set to 0.05, the shadow position was inaccurately represented, while a bias of 0.005 led to a large displacement between the shadow and the object. A bias of 0.0005 yielded better results, with 0.00005 producing similar results to the 0.0005 setting.

<table>
<tr><td> 
Bias = 0.05</td><td> 
Bias = 0.005</td></tr>
<tr><td> 
Bias = 0.0005</td><td> 
Bias = 0.00005</td></tr>
</table>

Frustum/ Projection Settings:

In my implementation, a directional light source with a perspective frustum was utilized. The light's field of view was set at 90, the aspect ratio at 1, with near and far planes matching the camera settings at 0.1 and 1000, respectively. The intention was to ensure that the light's field of view was sufficiently broad (hence, 90), and that the shadow map would not be a rectangle, but rather a square (hence, aspect ratio 1). Moreover, the far plane needed to encompass the entire scene, which is why a sufficiently large number (1000) was chosen.

My light view matrix employs a LookAt Matrix, thanks to the GLM library. Here, the light's position can move, always looking at the negative z-axis, represented in light position + vec(0, 0, -1). To construct the matrix, I set the up vector. Given that Vulkan's Z-axis is positive inside, the up vector should negate the world up vector, which should always be (0, -1, 0).

```
const float aspect = 1.f;

glm::mat4 lightProjection = glm::perspective(glm::radians(90.0f), aspect, cfg::kCameraNear, cfg::kCameraFar );

glm::mat4 lightView = glm::lookAt(state.lightPos, state.lightPos + glm::vec3{0,0,-1}, glm::vec3(0, -1, 0));

glm::mat4 lightSpaceMatrix = lightProjection * lightView;
```

1.2Extended Shadow maps

Face Culling:

I recommend front face culling for the shadow map and back face culling for scene rendering. Front face culling reduces the number of triangle draws. If a back face obscures the light, it must be unseen by the light source, thereby casting a shadow during rendering.

Multiple Shadow Mapping:

When multiple light sources are present, the computational cost increases significantly. In such scenarios, deferred shading is the best practice for handling multiple shadow maps and opacity scenes. It reduces the light computation from n to 1, performs shadow calculations for the shadow map, and finally renders the result.

Limitations:

1. **Opacity**: The scene must be opaque for deferred shading.
2. **Overlapping Shadows**: With multiple light sources casting shadows, there can be issues with overlapping shadows. Accurate representation of overlapping shadows can be computationally complex and can lead to visual artifacts if not properly managed.
3. **Complex Light Interactions**: Multiple light sources can lead to more complex light interactions, including specular highlights, diffuse illumination, and reflections. These interactions can be challenging to accurately calculate and render, particularly in real-time applications.