CSC305 Assignment 2 – Real-time Renderer Report

Dora Ou V00835123

In the vertex shader, transform vertices to clip space using the ModelViewProjection matrix. o Seethescene.vertshader.

o After doing this, you should see the silhouettes of the objects.

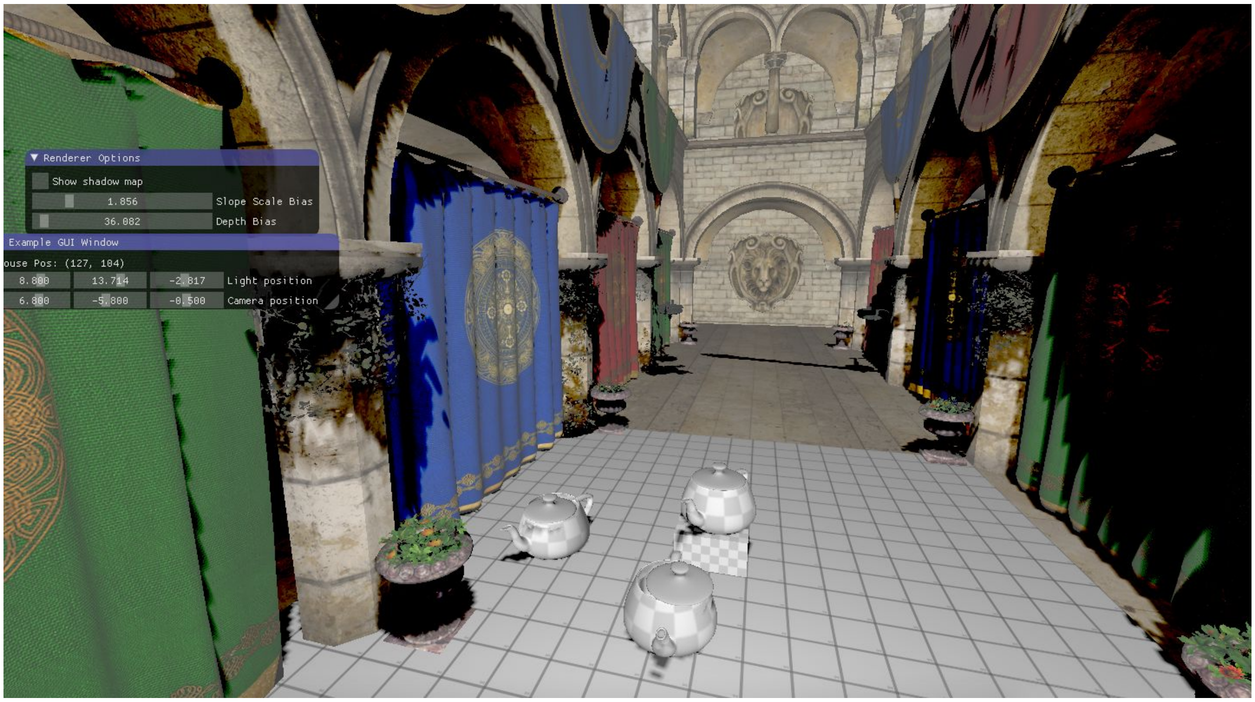
In vertices shader, get the glposition by MVP*position and pass scene texcoord to fragement shader.

In the pixel shader. implement Phong shading for a point light placed at the camera's position.

o Use the object's diffuse map texture for the diffuse color.

o The diffuse map is already hooked up the scene.frag shader.

In the fragment shader, apply the Blinn-Phong shading. Add diffuse map texture, if hasduffsemap not equals 0.



□ Show shadow map

1.856 Slope Scale Bias

36.082 Depth Bias

ouse Pos: (127, 104)

8.800 13.714 -2.817 Light position

6.800 -5.800 -0.500 Camera position

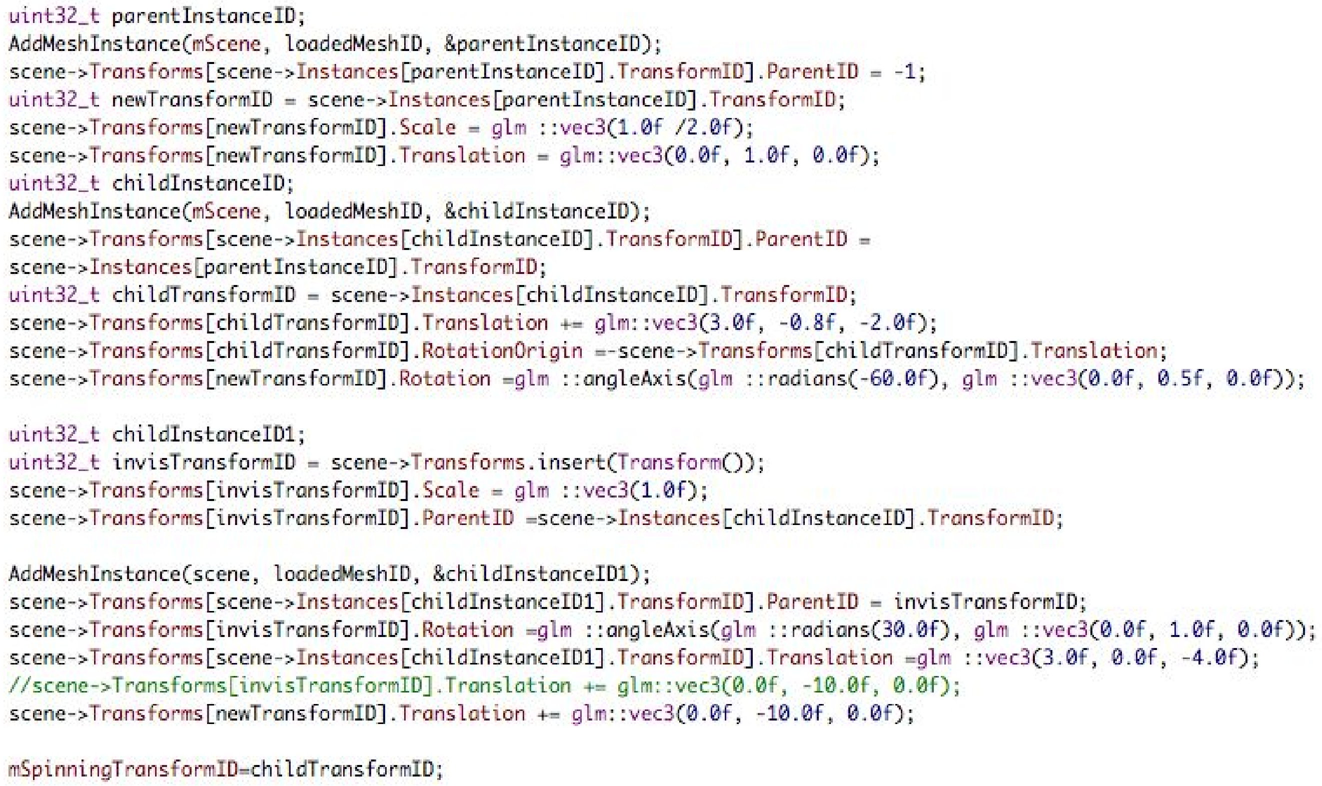
After doing the first 2 requirements, we can get a 3 teapot and a cube with diffusemap and Blinn-Phong shading as the above image shown. My color formula for the above two images is vec3 colorLinear = 0.1*Ambient+lambertian*Diffusemap+spec*Specularmap;

But I feel it is too bright. So I change it to vec3 colorLinear = 0.02*Ambient+lambertian*Diffusemap+spec*Specularmap; for the later images.

Implement a hierarchy of transformations (a “scene graph”) to place objects relative to others.

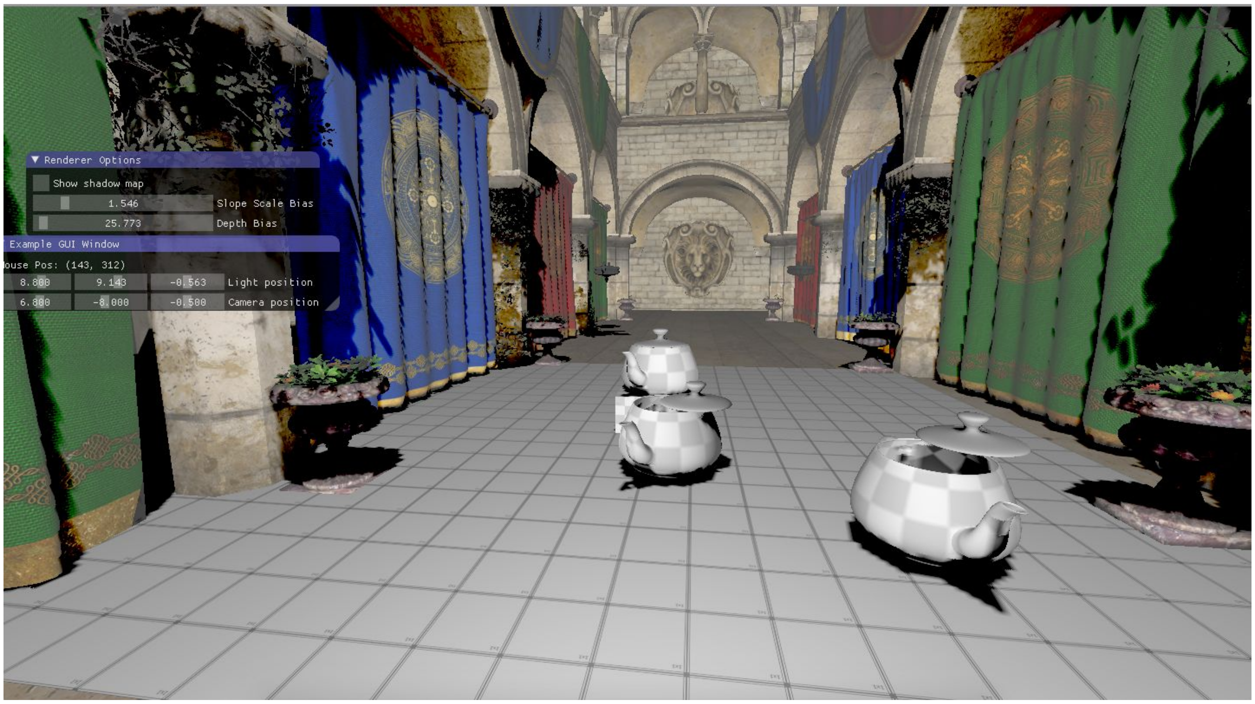
- o Extend “struct Transform” in scene.h to have a “ParentID”
- o From then, each Transform is considered relative to its parent: Transforms[ParentID].
- o ParentID==-1represents a root node.
- o At each frame, compute the absolute transform of each transform (traverse its parents.)
- o Use the absolute transform instead of the relative transform in your rendering.
- o Show that this works by rendering an object that orbits around another.

In my simulation.cpp, I changed the teapot loading to



```
uint32_t parentInstanceID;
AddMeshInstance(mScene, loadedMeshID, &parentInstanceID);
scene->Transforms[scene->Instances[parentInstanceID].TransformID].ParentID = -1;
uint32_t newTransformID = scene->Instances[parentInstanceID].TransformID;
scene->Transforms[newTransformID].Scale = glm ::vec3(1.0f /2.0f);
scene->Transforms[newTransformID].Translation = glm::vec3(0.0f, 1.0f, 0.0f);
uint32_t childInstanceID;
AddMeshInstance(mScene, loadedMeshID, &childInstanceID);
scene->Transforms[scene->Instances[childInstanceID].TransformID].ParentID =
scene->Instances[parentInstanceID].TransformID;
uint32_t childTransformID = scene->Instances[childInstanceID].TransformID;
scene->Transforms[childTransformID].Translation += glm::vec3(3.0f, -0.8f, -2.0f);
scene->Transforms[childTransformID].RotationOrigin =-scene->Transforms[childTransformID].Translation;
scene->Transforms[newTransformID].Rotation =glm ::angleAxis(glm ::radians(-60.0f), glm ::vec3(0.0f, 0.5f, 0.0f));

uint32_t childInstanceID1;
uint32_t invisTransformID = scene->Transforms.insert(Transform());
scene->Transforms[invisTransformID].Scale = glm ::vec3(1.0f);
scene->Transforms[invisTransformID].ParentID =scene->Instances[childInstanceID].TransformID;

AddMeshInstance(scene, loadedMeshID, &childInstanceID1);
scene->Transforms[scene->Instances[childInstanceID1].TransformID].ParentID = invisTransformID;
scene->Transforms[invisTransformID].Rotation =glm ::angleAxis(glm ::radians(30.0f), glm ::vec3(0.0f, 1.0f, 0.0f));
scene->Transforms[scene->Instances[childInstanceID1].TransformID].Translation =glm ::vec3(3.0f, 0.0f, -4.0f);
//scene->Transforms[invisTransformID].Translation += glm::vec3(0.0f, -10.0f, 0.0f);
scene->Transforms[newTransformID].Translation += glm::vec3(0.0f, -10.0f, 0.0f);

mSpinningTransformID=childTransformID;
```

In my code, I let parentInstanceID be a parent of childInstanceID,
childInstanceID to be a parent of invisTransformID,
and invisTransformID to be a parent of childInstanceID1.
Hence, it is parentInstanceID->childInstanceID->invisTransformID->childInstanceID1
So when i Spinning Trans childTransformID, the childInstanceID, invisTransformID and
childInstanceID1 will rotate around parentInstanceID as follow.



▼ Renderer Options

□ Show shadow map

1.546 Slope Scale Bias

25.773 Depth Bias

Example GUI Window

ouse Pos: (143, 312)

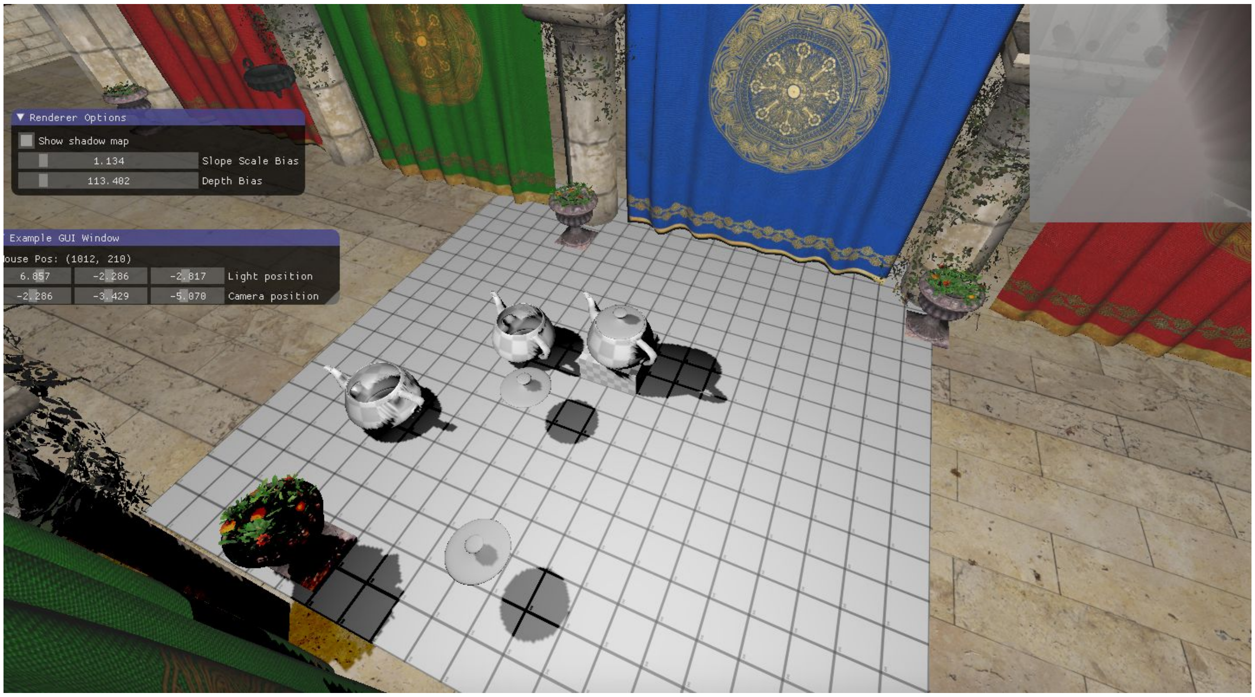
8.800 9.143 -0.563 Light position

6.800 -8.000 -0.500 Camera position

Implement a directional shadow map (from the sun.)

- o Create a separate FBO and depth texture to render a shadow map.
- o Make sure the depth comparison mode is set on the texture. (see glTexParameter docs)
- o Render the scene into the shadow map in a depth-only pass.
- o Bind the shadow map as a texture to your scene fragment shader.
- o Sample the shadow map with sampler2DShadow using the output of the light matrix.
- o Incorporate the shadowing into your lighting computation
- o Therearegoodshadowmaptutorialsontheweb.Checkthemout.

I think it is the hardest part. At first, I didn't set kShadowMapResolution, then it always give a error in shadowdepth. Then I set kShadowMapResolution as 512. In shadow rendering, I pass the ModelViewProjction like scene rendering did, but the shadow one is dependes on light position rather than camera position. In scene rendering, I pass the shadow depth and light matrix to shader and draw in FragColor = vec4(colorLinear - (1.0 - visibility) * Ambient*0.5, 1.0); The slide recommand FragColor = vec4(colorLinear*visibility), but I feel my formula looks better. The result shows below.



▼ Renderer Options

□ Show shadow map

1.134 Slope Scale Bias

113.402 Depth Bias

Example GUI Window

ouse Pos: (1012, 210)

6.857 -2.286 -2.817 Light position

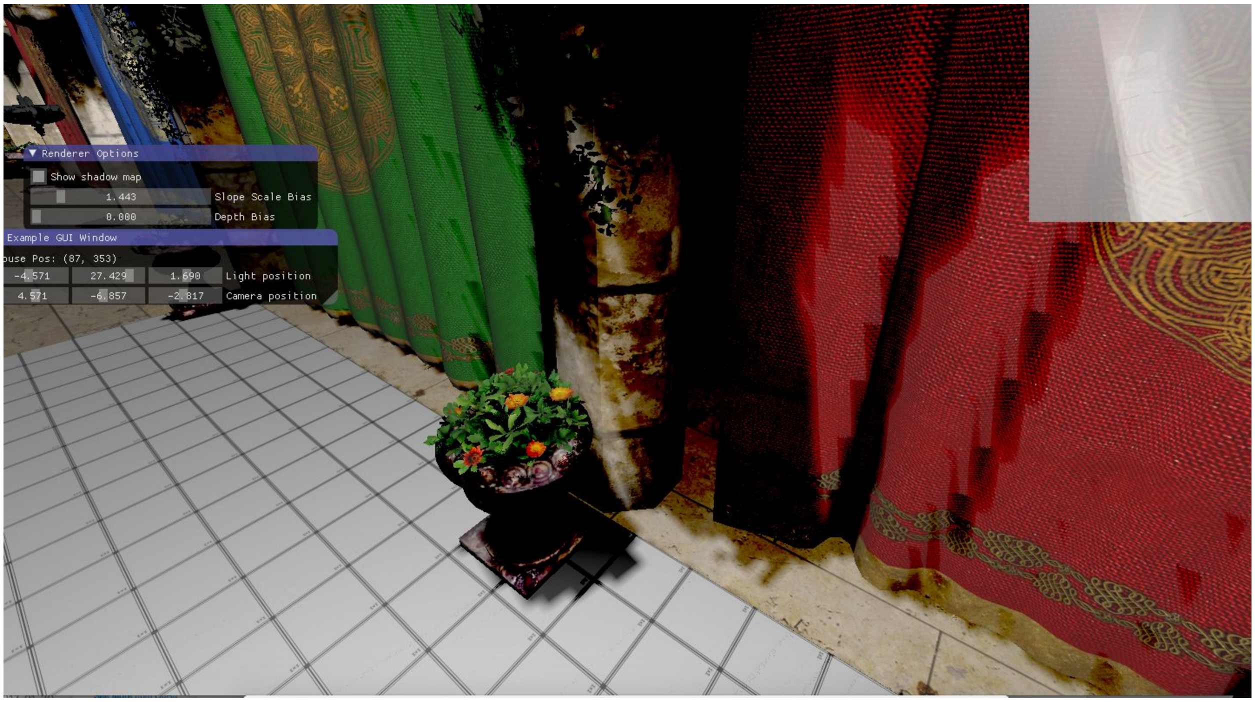
-2.286 -3.429 -5.070 Camera position

Render the Sponza scene (see http://www.crytek.com/cryengine/cryengine3/downloads)

o Implement its diffuse maps, specular maps, bump maps, and alpha masks.

o You must extend the scene loading code in scene.h/.cpp to handle these materials.

I pass its diffuse maps, specular maps, bump maps,and alpha masks successfully, and apply diffuse maps, specular maps and alpha masks. For diffuse maps,and specular maps, I use the formula “vec3 colorLinear = 0.02*Ambient+lambertian*Diffusemap+spec*Specularmap;”. For alpha mask, I use “ if (texture(AlphaMask, fragment_texcoord).r < 0.9){ discard;}”. The screenshot images below shows the alpha masks in plants.



□ Show shadow map

1.443 Slope Scale Bias

0.000 Depth Bias

ouse Pos: (87, 353)

-4.571 27.429 1.690 Light position

4.571 -6.857 -2.817 Camera position

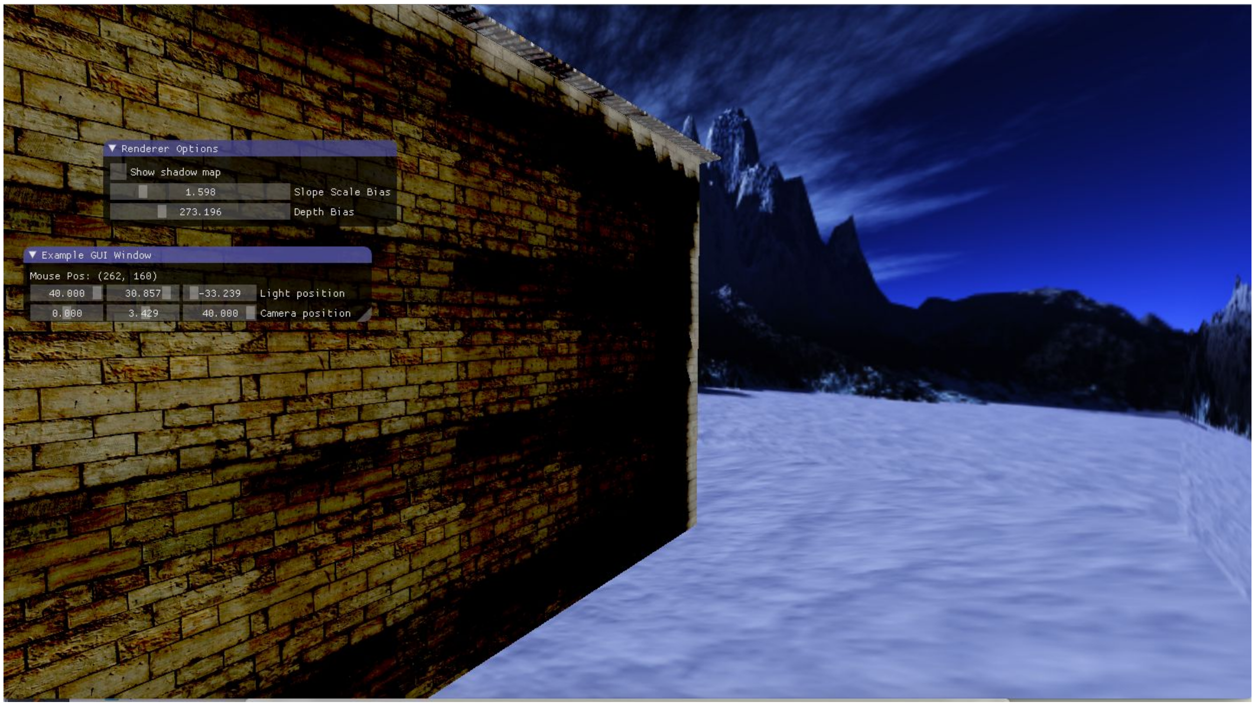
Implement a skybox.

o Create a cube map texture where the 6 faces correspond to the skybox faces.
You can grab a cubemap from the web.

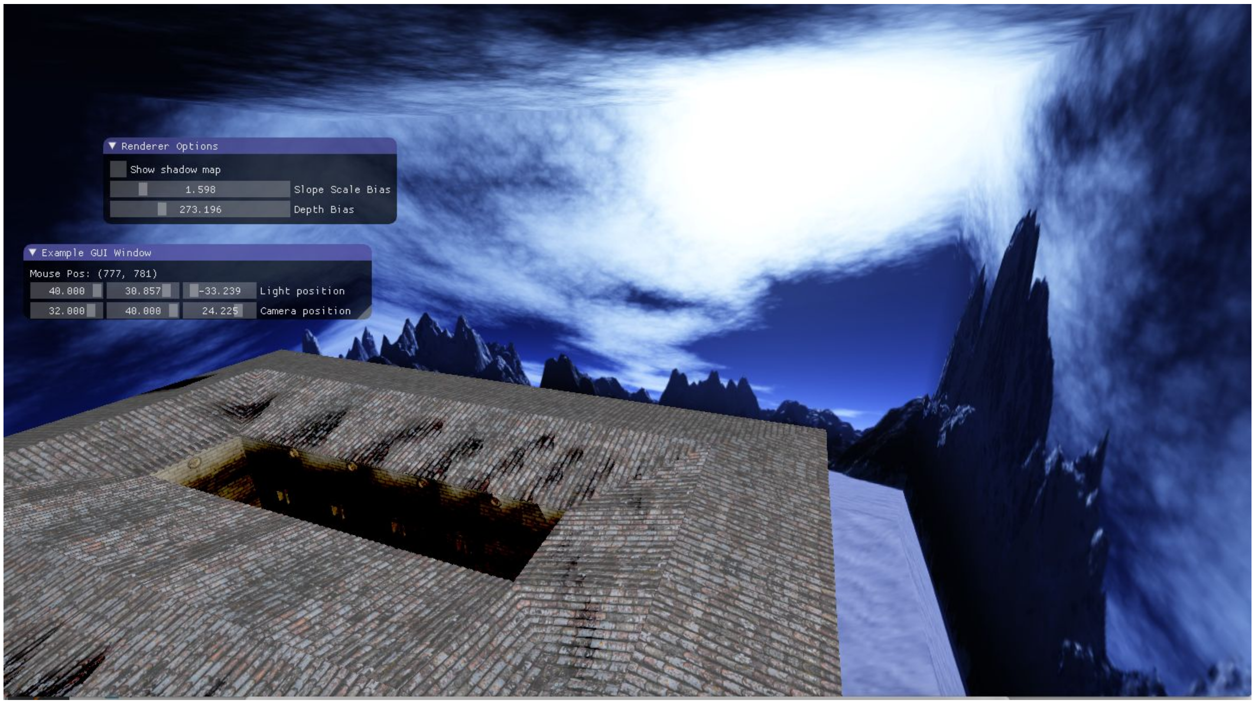
o Add a skybox rendering pass after rendering the scene.

o Make sure to use depth testing to avoid rendering the skybox where it isn't necessary.

Firstly, I the box position like x=(-50,50), y=(-10,50), z=(-50,50). Then I load the image by stbi_load and load each face to GL_TEXTURE_CUBE_MAP by glTexImage2D() function. To display the skybox, I create skybox shaders. In skybox rendering, I pass viewprojection and worldview to determine the g1position in vertices shader and pass the cube texture to render image in the cube. The screenshot image for skybox shows below.

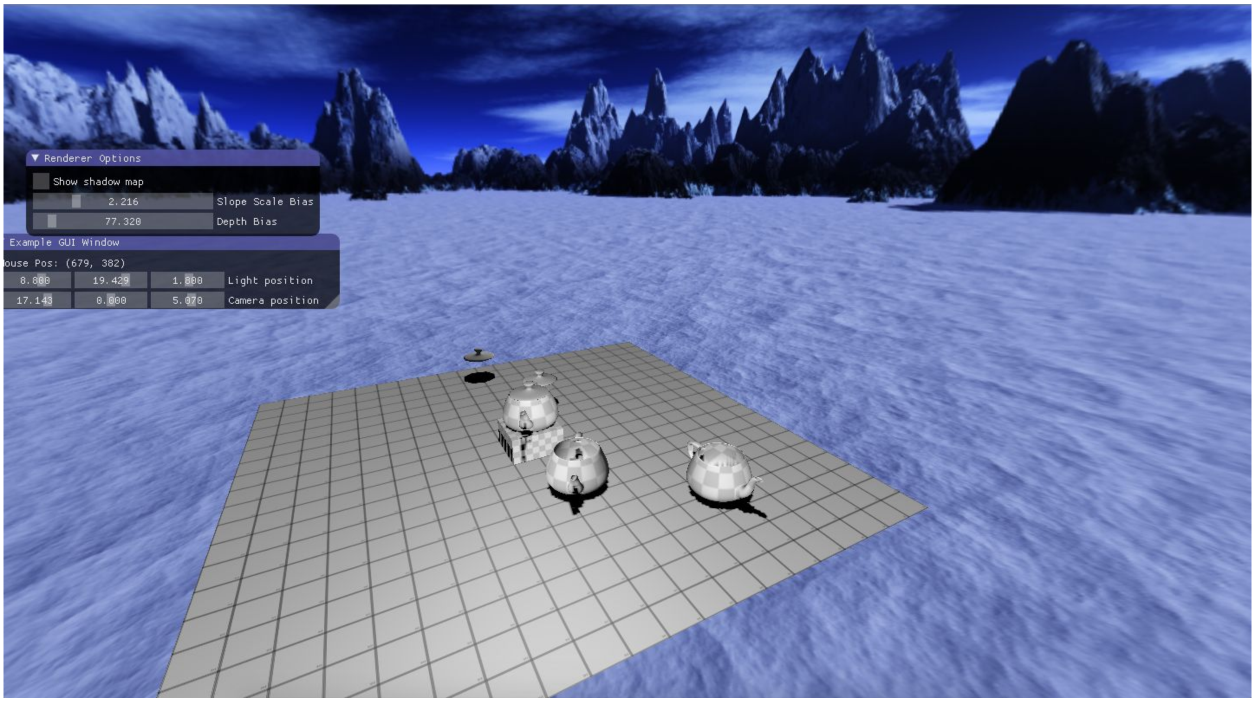


```
▼ Renderer Options
□ Show shadow map
      1.598              Slope Scale Bias
      273.196            Depth Bias

▼ Example GUI Window
Mouse Pos: (262, 160)
  40.000    30.857    -33.239   Light position
  0.000     3.429     40.000    Camera position
```



```
▼ Renderer Options
□ Show shadow map
      1.598              Slope Scale Bias
      273.196            Depth Bias

▼ Example GUI Window
Mouse Pos: (777, 781)
  40.000    30.857    -33.239   Light position
  32.000    40.000    24.225    Camera position
```

The connex submission omit the sponza scene, which is



▼ Renderer Options

□ Show shadow map

2.216 Slope Scale Bias

77.320 Depth Bias

Example GUI Window

ouse Pos: (679, 382)

8.800 19.429 1.800 Light position

17.143 0.000 5.070 Camera position