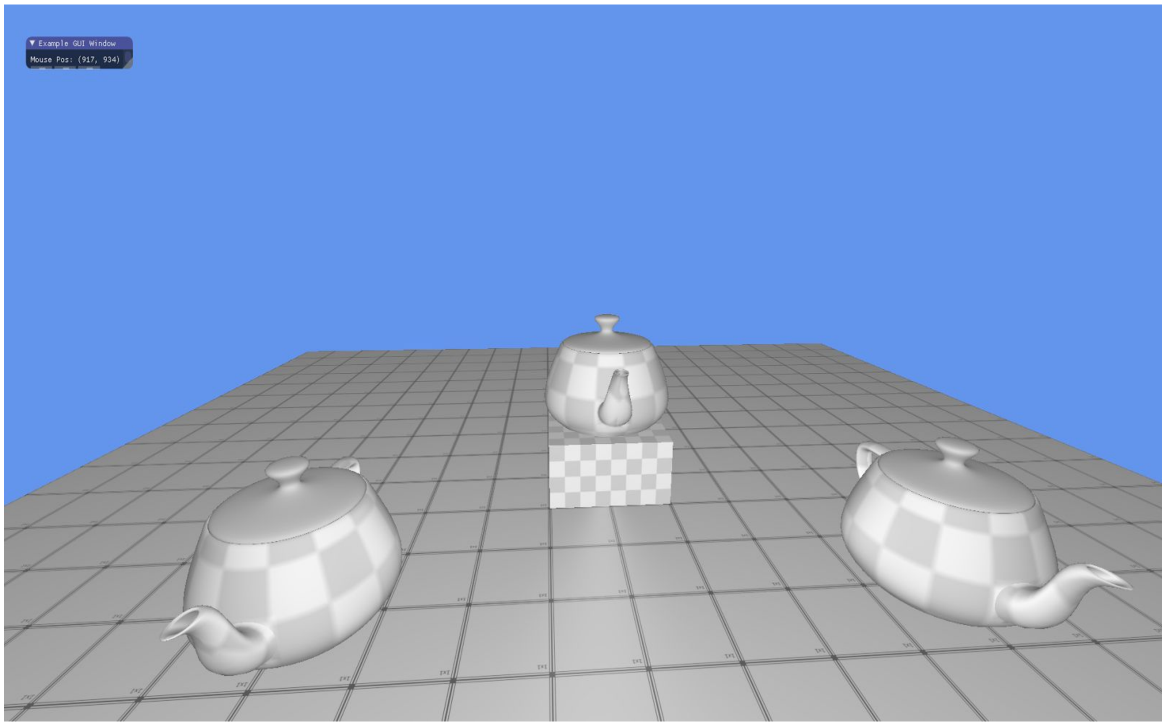
CSC 305 Assignment 2 – Real-time Renderer

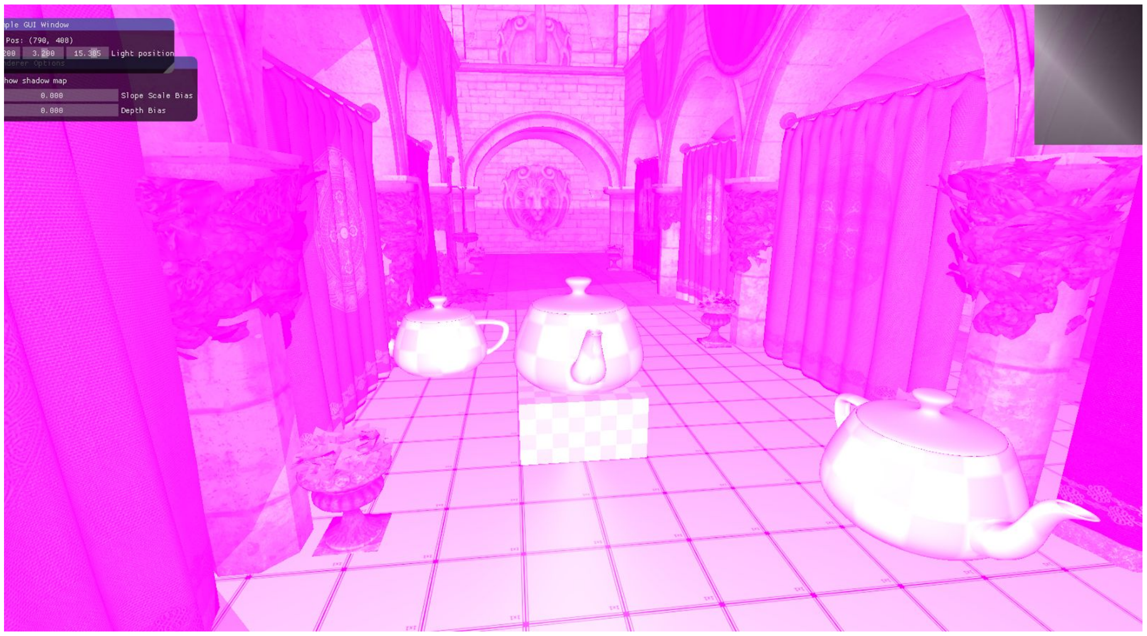
Leanne Feng
V00825004

Vertex Shader, Fragment Shader and Phong Shading: The problem is to passed values from vertex shader to the fragment shader, and implements Blinn-Phong shading with diffuse, shininess and specular light from a point light source. We can first output clip space coordinate by writing to gl_Postion using the ModelViewProjection. It is useful to pass values using in and out declared global variables so we can pass the normals, texture coordinates, positions and colours. In fragment shader, blinn phong shading model is used.

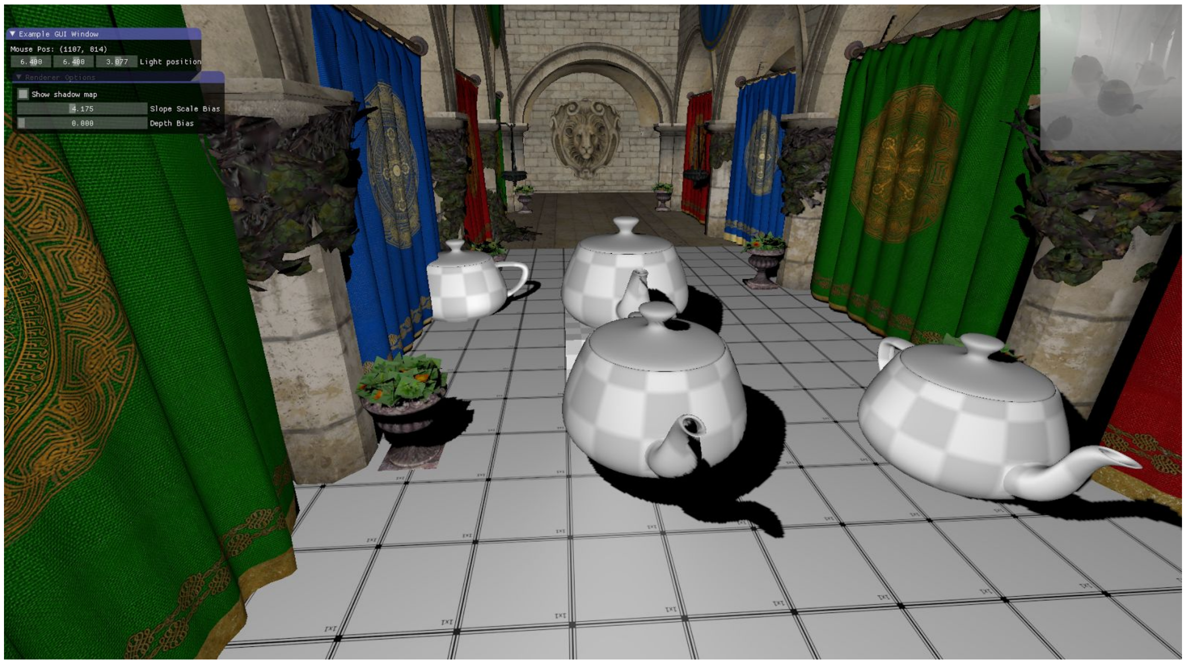
```
  vec3 halfDir = normalize(lightDir + viewDir);
  float specAngle = max(dot(halfDir, normal), 0.0);
  specular = pow(specAngle, Shininess);
```

To output the different colour, diffuse, and specular effect, equation vec3 color = vec3(0.0f) +lambertian * diffuseMap +specular * Specular; is used.





Hierarchy of transformations: The problem is to place objects relative to others and spin around others, the hard part is to make both teapot body and lid child of a transform and rotate parent transform. ParentID==-1 is set to represent the root node. I set global vector unit32_t variable mSpinningTransformIDs, use push_back, and in update use for (uint32_t mSpinningTransformID : mSpinningTransformIDs) to update both teapot body and lid to rotate around parentID. Also, a call of AddMeshInstance() function creates a new teapot.



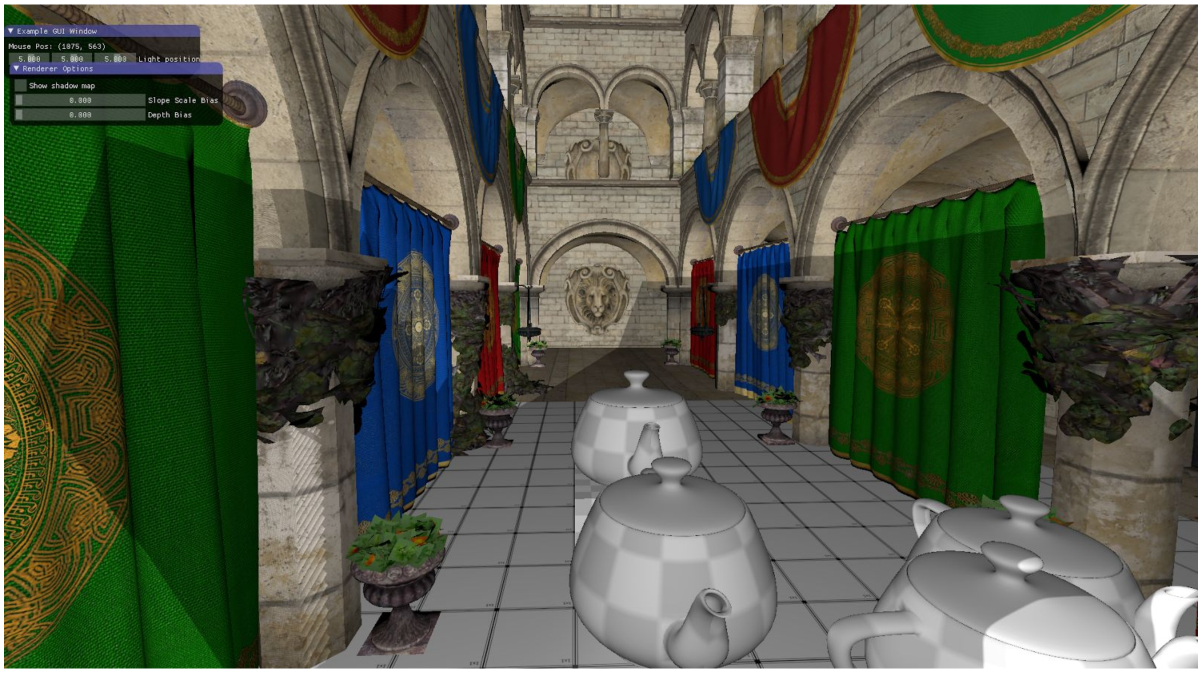
```
Mouse Pos: (1107, 814)
6.400  6.400  3.077  Light position
Show shadow map
4.175  Slope Scale Bias
0.000  Depth Bias
```

Directional shadow map:

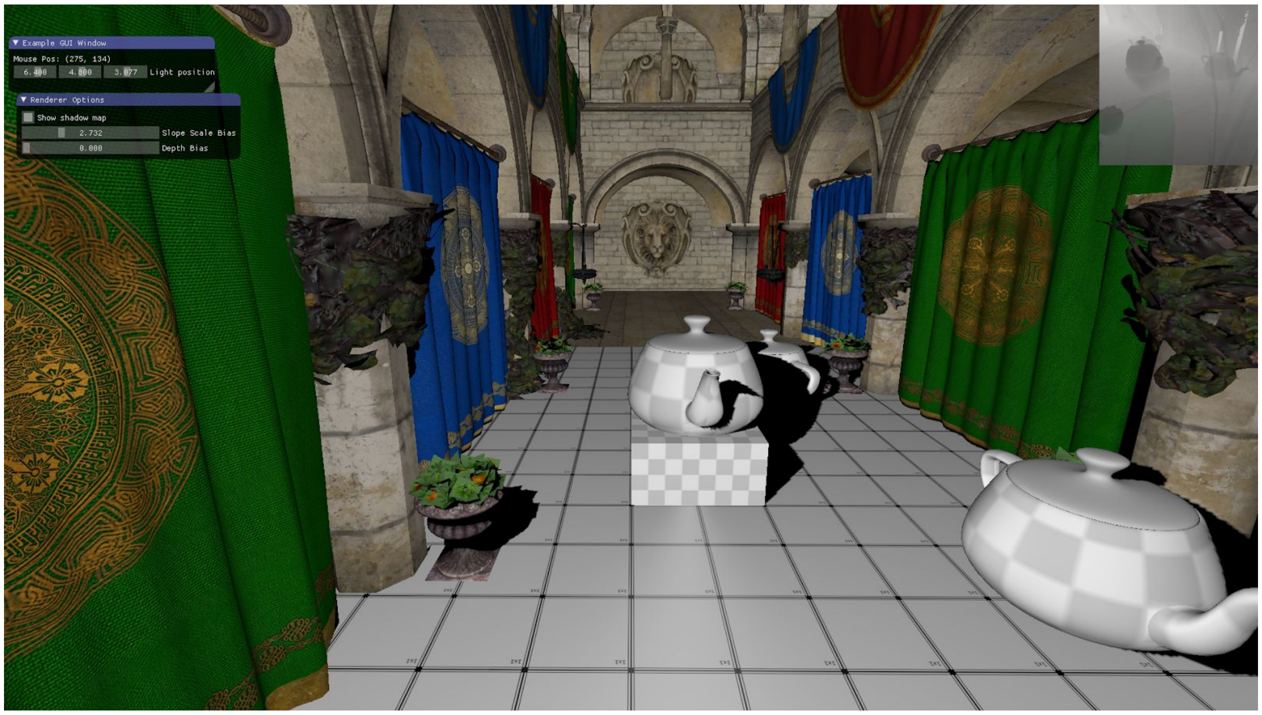
First of all, I initial a shadow map depth texture by the created global variables mShadowFBO and mShadowDepthTO and shadow shader. Also create new light's wolrdViewProjection and light matrix for light source.Then, first pass the shadow map rendering and using the lightviewprojection matrix. In shadow rendering, shadow map is binded as a texture to scene fragment shader. Second pass the scene rendering, pass the texture to shader and pass the clip space coordinate using the light matrix by "in" and "out". There is also a similar renderer mDepthVisSP using for shadow map Debug visualization.

Render the Sponza scene:

In simulation, initial the sponza scene, loaded from the file where you put the sponza information and add its instance meshes to output sponza scene effect.



```
▼ Example GUI Window
Mouse Pos: (1075, 563)
5.000  5.000  5.000  Light position
▼ Renderer Options
Show shadow map
0.000  Slope Scale Bias
0.000  Depth Bias
```



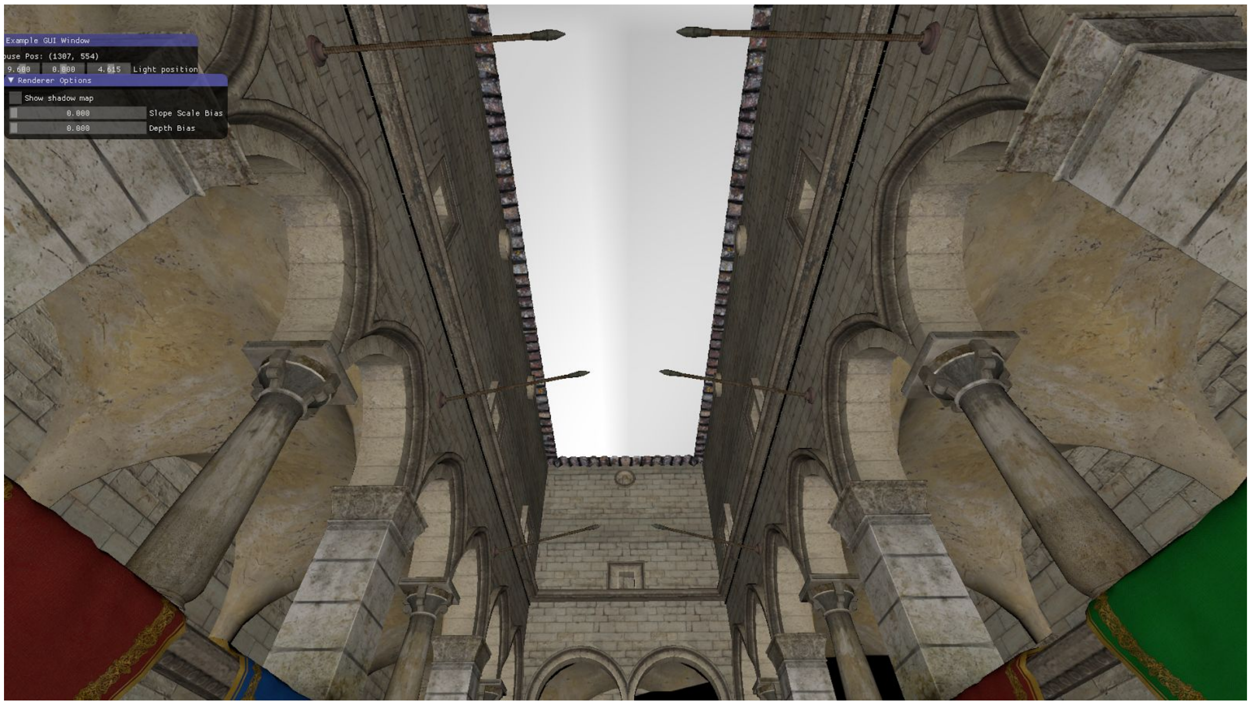
```
▼ Example GUI Window
Mouse Pos: (275, 134)
6.400  4.800  3.077  Light position
▼ Renderer Options
Show shadow map
2.732  Slope Scale Bias
0.000  Depth Bias
```

Skybox:

Load the skybox meshes in simulation init and scale down. However, there are colours when open skybox.obj, but showing white after running the program.

```
  loadedMeshIDs.clear();
```

```
LoadMeshesFromFile(mScene, "assets/skybox/skybox.obj", &loadedMeshIDs);
for (uint32_t loadedMeshID : loadedMeshIDs)
{
	uint32_t newInstanceID;
	AddMeshInstance(mScene, loadedMeshID, &newInstanceID);
	uint32_t newTransformID = scene->Instances[newInstanceID].TransformID;
	scene->Transforms[newTransformID].Scale = glm::vec3(1.0f /300.0f );
```


Example GUI Window

ouse Pos: (1307, 554)

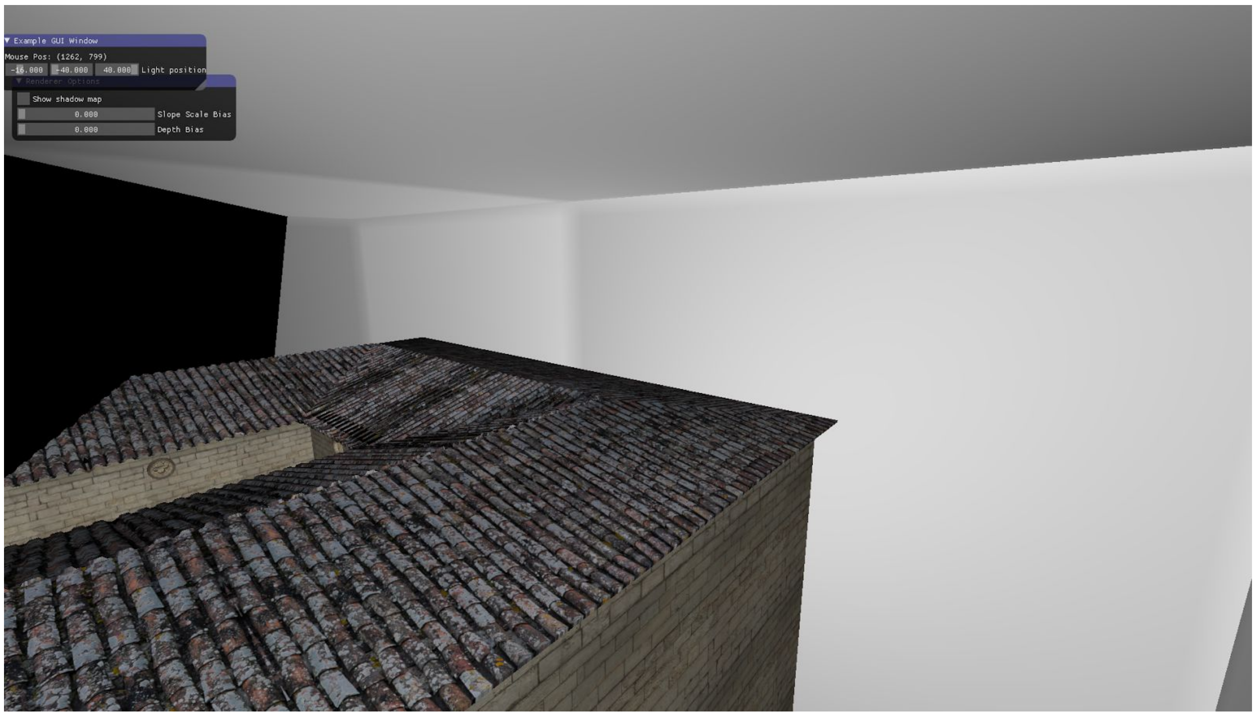
9.600 0.000 4.615 Light position

▼ Renderer Options

Show shadow map

0.000 Slope Scale Bias

0.000 Depth Bias


▼ Example GUI Window

Mouse Pos: (1262, 799)

-16.000 -40.000 40.000 Light position

▼ Renderer Options

Show shadow map

0.000 Slope Scale Bias

0.000 Depth Bias