Homework - 12

CS594

Shadow mapping

This assignment introduces shadow mapping in OpenGL. This is an important set of techniques used to create shadows in 3D graphics. It works by first creating shadow maps for each light source, via depth buffering and then using these shadow maps while performing lighting calculations for each vertex. Recall depth buffering from HW-11, wherein, along each line of sight, the distance of the object closest to the camera is noted. This is done in order to determine which vertex is visible and which one is occluded. The same concept may be employed in order to compute shadows. A shadow may be computed by determining which vertices are "visible" from the light-source and which ones are occluded. Hence, if one replaces the camera with the light-source while computing depth buffering, the output is effectively, the shadow map. Click the link below to follow the tutorial and complete the following exercise.

Shadow mapping: This tutorial performs shadow mapping for a parallel light source, i.e., all the rays of light travel parallelly. An example of such a light source is our Sun. Basic shadow mapping is introduced and issues such as shadow acne, aliasing, etc. are discussed and addressed. The code has two sets of shaders. `DepthRTT.vertexshader` and `DepthRTT.fragmentshader` are used to compute shadow maps. `ShadowMapping.vertexshader` and `ShadowMapping.fragmentshader` use the shadow maps computed in the first step to perform lighting calculations.

Exercise: Current code contains one parallel light source with direction $(0.5, 2, 2)$. Change this to $(0.8, 2, 2)$. Add another parallel light source with direction $(-0.8, 2, 2)$. In order to perform the lighting calculations with this updated scene, as the first step, compute shadow maps for the two light

sources. In the second step, combine the two shadow maps to perform the final lighting calculations. A snapshot of the output is shown in Figure 1. Notice how each object now casts two shadows and wherever the two shadows "overlap", the shadow is darker. You may decide the actual value of light intensity (or rather the lack of it) at such points on your own. Submit a snapshot image from the output clearly showing the dual shadows. Also submit all the code files that you modify.

Hints: In the C++ code, you will need two depth-textures for the two shadow maps. Also two depthMVP matrices for the two light sources and corresponding depthBiasMVP matrices. Correspondingly, in the vertex shader, two sets of shadow-coordinates for the vertex, which are passed on to the fragment shader. Also, the two shadow maps are passed on to the fragment shader from the C++ code.

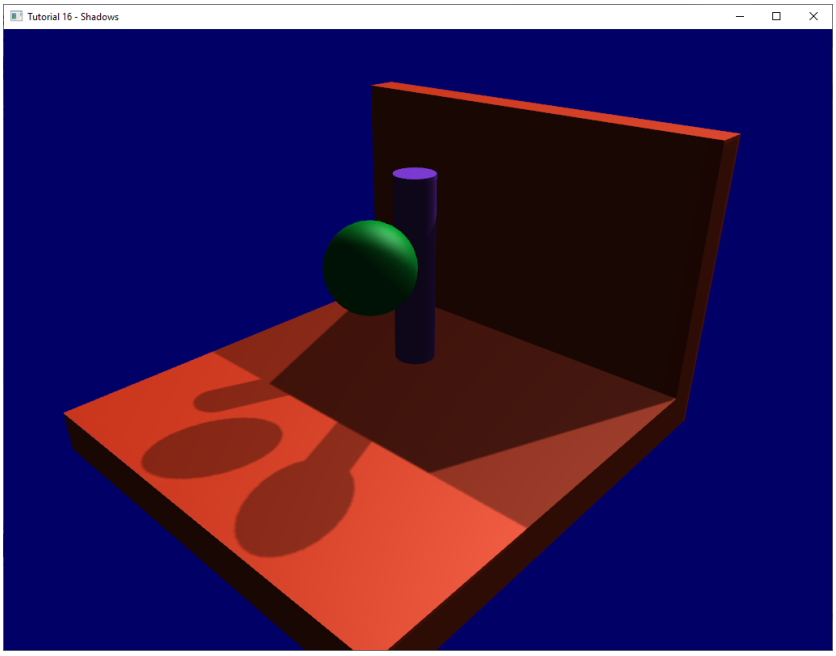

Figure 1: Shadow map with two light sources.