THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
Department of Computer Science and Engineering

COMP 541 : Advanced Computer Graphics

Fall 2013
Assignment 3: Shadows
Due date: December 2, 2013 at 11:59PM (using CASS)

In the previous assignments, you became familiar with how to use basic OpenGL to process and render a mesh, as well as to shade it using GLSL. In this assignment, your objective is to add shadows to the rendering application.

The starting mesh viewer for the assignment can be downloaded from here:
http://www.cse.ust.hk/~psander/5411/PA3.zip

The included mesh viewer allows you to visualize the mesh in several modes by pressing the keys 1 through 9 or using the popup menu as in previous assignments. The active mode is displayed in the console window. You should finish the implementation of the modes "*Shadow map*" and "*Shadow volume*" in order to have a working version of the two techniques. A lot of the work is already done for you in the sample code. Below are the parts that you should to complete (all of them are specified with a "TODO" comment):

1) **Shadow mapping (50%)**
   You must finish the implementation of the shadow mapping algorithm by completing the following parts of the code:
   a) In the function `SetupShadowMapPOVMatrices()`, compute the transformation matrix that each vertex of the model needs to undergo in order to render the scene from the point of view of the light and generate the shadow map. Note that the transformation should compute the normalized device coordinates (i.e., screen coordinates). Refer to the code comments for specific guidelines.
   b) If the function `SetupShadowMapTextureMatrix()`, compute the transformation matrix that each rendered pixel of the model needs to undergo in order to determine its position in the generated shadow map. Note that the transformation should compute the texture space coordinates. Again, refer to the code comments for more specific guidelines.
   c) In the shader file `render_perlight_shadow_map.glsl`, complete the fragment shader so that it properly performs the depth test and determines whether the pixel is on shadow. The shader code comments describe the specific steps.

   Debugging: The mode "*shadow map visualization*" allows you to view the rendered model from the point of view of the two lights (side-by-side). Also, note that the resulting shadow will have saw-like artifacts. This is not necessarily a bug. Recall from class, that these are common issues with shadow mapping caused by insufficient resolution of the shadow map texture.

2) **Shadow volume (50%)**

You should finish the implementation of the shadow volume algorithm. In the shader file `shadow_volume.glsl`, complete the geometry shader so that it detects silhouette edges. All three edges of each processed triangle need to be processed. The provided skeleton code calls the `DetectAndProcessSilhouette()` function for each of those edges, and your task is to determine if the edge is indeed a silhouette with respect to the light. This can be achieved by checking if the processed triangle is facing the light, and the adjacent triangle is facing away from the light. In case this is true, you should then extrude a quad (i.e., two triangles) to generate the shadow volume. Refer to the shader code comments for additional guidelines.

Debugging: The mode "shadow volume visualization" allows you to visualize the extruded quads overlaid onto the original model. Note that the shadow volume rendering result should be accurate to the pixel, although it runs slower than shadow mapping.

Note: In order to be able to develop in your own machine, you must have a desktop or notebook with a DX10-level or above graphics chip (i.e., Radeon HD2000+ or NVIDIA GeForce 8000+/GTX) and install the latest OpenGL drivers. If you do so and the program doesn't run, email the TA Mr. Ge Chen at gechen@cse.ust.hk and describe the error message that you are getting. If you don't have access to such a machine, you may use the machines in lab 4 (room 4210), which are all equipped with the appropriate hardware and drivers.

Submission: You should only submit the files that you have modified (i.e. main.cpp and the two shader files). A concise report including snapshots and a brief description of your code is appreciated and will help us grade your work.

Figure: Shadow map (left) and shadow volume (right) comparison. Correct implementation of the two should match each other except at shadow boundaries, where the quality of shadow volume is better. Please also refer to the video demo available at the assignment page.