02561 COMPUTER GRAPHICS DTU COMPUTE

Worksheet 8: Projection shadows and render pipeline

Reading	Angel: 5.10, 7.5.6, 7.10-7.10.3, 8.11
Purpose	The purpose of this set of exercises is to produce simple shadows using projection matrices. As a side product, the aim is to get a better understanding of the rasterization pipeline. We are only concerned with generating the shadows – this means that using Phong lighting is an optional extension.
Part 1 Scene	The scene to be rendered consists of three quadrilaterals (quads). One is a large texture mapped quad in the plane $y = -1$ ($x \in [-2, 2], z \in [-1, -5]$), the others are smaller quads colored red. Let us refer to the large quad as the ground. One of the two smaller quads should be parallel to $y = -1$, but placed above the ground ($y = -0.5, x \in [0.25, 0.75], z \in [-1.25, -1.75]$). The other should be perpendicular to $y = -1$ with two vertices intersecting the ground ($x = -1, y \in [-1, 0], z \in [-2.5, -3]$). Create a WebGL program that draws this scene. Here are some steps: • Start from Part 1 of Worksheet 6. Use the coordinates given above to set the vertex coordinates of the ground. Adjust the texture coordinates of the ground so that the texture fills out the square without being repeated. • Replace the checkerboard texture by the texture image in `xamp23.png` (available on DTU Learn). • In initialization, switch to `gl.TEXTURE1` using `gl.activeTexture` and create a new texture of 1×1 resolution, where you store just a single red color: `Uint8Array([255, 0, 0])`. [Angel 7.5.6] • Add the two smaller quads to your vertex and texture coordinate buffers. Draw the ground quad with texture `0` and the smaller red quads with texture `1`. [Angel 7.5.6]

Worksheet 8: Projection shadows and render pipeline

Part 2 Projection shadows	A light source position is needed to cast shadows. Introduce an animated point light that circles the scene with circle center $(0, 2, -2)$ and radius 2. Implement projection shadows using the following steps. • Create a projection matrix $\boldsymbol{M}_p$ that projects geometry onto the ground plane $y = -1$. Projection to a plane different from $y = 0$ is done by subtracting the y-coordinate of the plane from the y-coordinate of the light source in $\boldsymbol{M}_p$. [Angel 5.10][1] • Construct a shadow model matrix $\boldsymbol{M}_s$ by concatenating $\boldsymbol{M}_p$ with model and translation matrices so that shadow polygons are projected from the current position of the point light onto the ground plane. [Angel 5.10] • Use the shadow model matrix to draw the smaller quads again but as shadow polygons. Note that drawing order is important. Ensure that the shadow polygons are in front of the ground polygon, but behind the smaller quads. [Angel 8.11.7]
Part 3 Shadow polygon culling using the z-buffer	One problem with shadow polygons is that they are drawn even if there is no ground polygon. Use the depth buffer with a depth test function that accepts fragments with greater depth values to draw shadow polygons only if there is also a ground polygon. Handle z-fighting using an offset in the projection matrix. [Angel 8.11.5] Introduce a uniform visibility variable in your fragment shader. Use this variable as a multiplication factor to draw the shadow polygons in black.
Part 4 Ambient light in shadows using transparency	The black shadows seem too dark. We would like to see a darker version of the ground texture in the shadows. Semi-transparent shadow polygons can achieve this effect. Enable blending and set an appropriate blending function to render a darker version of the ground texture in the shadows. [Angel 7.10-7.10.3] Blending in WebGL is influenced by browser compositing. Use `var gl = WebGLUtils.setupWebGL(canvas, { alpha: false });` to switch off this effect.
Part 5 **Optional**	Implement Phong lighting for the geometry in the scene.

[1] Please note that the indexing is faulty in the example code of the textbook that creates a projection matrix.