Praktikum: Echtzeit Computergrafik

Bonus Assignment – *Shadow Mapping*

10 Points

For the last assignment, we'll add a basic shadowing technique to our game that allows enemy ships to cast a shadow on the ground: shadow mapping.

C.1. Rendering into the shadow map (5P)

For shadow mapping, the depth of all objects as seen from the light source needs to be rendered before the actual final rendering happens; check the slides for an algorithm overview and on how to set up additional render targets.

C.1.1. Create a new Texture2D that will be used to render the shadow map. Also create an according DepthStencilView and a ShaderResourceView for your shadow map. Select the size as you want, we'd recommend something along 2048 x 2048. Important: The format of your resource needs to be R32_TYPELESS, your SRV will be of R32_FLOAT type and your DSV D32_FLOAT.

C.1.2. Create additional rendering passes (or techniques) in your FX file for everything that should cast a shadow – so basically your terrain, ships and ground objects. You can simply copy & paste all your existing vertex-shaders for this. For the pixel shaders, bind NULL for all shadow map passes as we do not need any color output.

C.1.3. Extend your rendering methods: before the actual on-screen rendering, render everything that should cast a shadow into your shadow texture. Only bind your shadow DepthStencilView and no render target for this. Make sure you clear the view each frame before rendering. Remember to also restore your old render targets afterwards.

We'd advise you to perform this in two steps for debugging purposes: first, just keep your viewport and projection matrix (make sure your shadow map resolution matches your screen resolution if you do this!) and check the content of the shadow map (see the next checkpoint). If that's fine, use a fixed shadow map size and adjust your viewport and projection matrix before rendering; remember to save your old viewports and check your shadow map again.

Checkpoint: At this point, we **strongly** advise you to check if everything is rendered to your shadow map correctly! Unfortunately, PIX does not support the display of typeless resources… to view your shadow map, you could render a fullscreen quad and sample from the shadow map texture (remember to bind the SRV, of course!). Depending on your near- and far-plane, you might need so scale the values as most of them will be near 1. For the fullscreen quad, you could also simply abuse your already existing skybox rendering pass.

C.1.4. Adjust your shadow map view and projection matrices such that the scene is rendered in orthogonal projection from the light direction. Check the slides for this!

Checkpoint: Check your shadow map again to see if everything is ok so far. Seriously, you don't want to continue unless you're sure your shadow map is rendered correctly…

C.2. Using your shadow map (5P)

Once your shadow map is created, it's time to modify your existing shaders to check for objects in the shadow.

C.2.1. Bind the shader resource view of your shadow map to a texture resource in your rendering shader. Also pass your light view- and projection-matrix to the shader.

C.2.2. In the vertex shaders of all shadow-receiving objects, calculate the position in light space using the light view- and projection-matrix and pass this to the pixel shader.

C.2.3. In the pixel shaders of all shadow-receiving objects, compare the depth of the so passed position in light space to the position stored in the shadow map with a small bias (about 0.01 or so). If the depth in the shadow map is less than the one of the current fragment, make the fragment look "darker" (e.g. only apply ambient light).
Some traps here: You'll need to divide by w! Also, the position you'll get is in clip coordinates in the range of [-1; 1] – to sample the shadow map texture, you'll need to scale this to [0; 1]. And last but not least, your y-coordinate will be inverted in the texture.

Checkpoint: You should see shadows on all objects.

10.2.4. To make the shadows appear smoother, apply 2x2 percentage closer filtering as supported by hardware. To do this, use `SampleCmpLevelZero` with an appropriate `SamplerComparisonState` and use the result as a "shadow factor". See the slides for additional information.

Checkpoint: The edges of your shadows should now appear smoother.

Note: Completing the Bonus Assignments is voluntary. It won't have any effect on the grading bonus or the exam.

Deadline: 30.07.2020