CSE 167 (FA 2022) Shadow Mapping. Final project expectations, guide and hints

In this final project topic, we implement shadows in OpenGL using the aid of texture. In essential idea is presented in the slides of 10/28. Bonus credits will be given if you also implement additional techniques for better shadows, such as the perspective warping of the light space to rearrange the light space resolution to reduce pixelation of shadows, or techniques for soft shadowing.

There are some good resources you may find helpful.

- https://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/
- https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping

Expectation

Present a scene (like HW3) with shadows from at least one light source. The light source can be a distant light (light at infinity) or a point light.

While bonus credits will be given as long as you have excellent exposition and demonstration, one possible direction to get bonus is to implement one of the methods to reduce the pixelated artifacts on the shadow (*e.g. Perspective Shadow Maps* (PSM)[1], *Light Space Perspective Shadow Maps* (LiPSM)[2][3], *Trapezoidal Shadow Maps* (TSM)[4], or your own heuristics.) All of these methods take advantage of a perspective transformation to rearrange the density of the light rays so that we have less pixelated effect on the edge of the shadow.

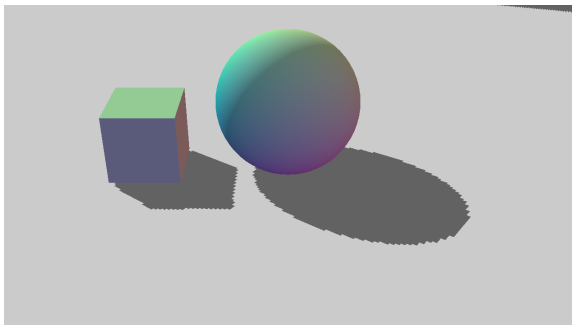

Figure 1 An example of demonstrating having shadows in the scene.

Basics

The shadow mapping technique requires two rendering passes. That is, in the display callback function, we will call draw scene twice. However, the two rendering are set with two different cameras and different shaders.

- In the first rendering, the camera is at the light responsible for casting shadows.
- In the second rendering, the camera is placed at the actual camera, forming the final image.

[1] https://www-sop.inria.fr/reves/Basilic/2002/SD02/PerspectiveShadowMaps.pdf

[2] Original paper https://www.cg.tuwien.ac.at/research/vr/lispsm/shadows_egsr2004_revised.pdf

[3] Sec 3.2.2 of the course note http://research.michael-schwarz.com/publ/files/shadowcourse-eg10.pdf

[4] https://www.comp.nus.edu.sg/~tants/tsm/tsm.pdf

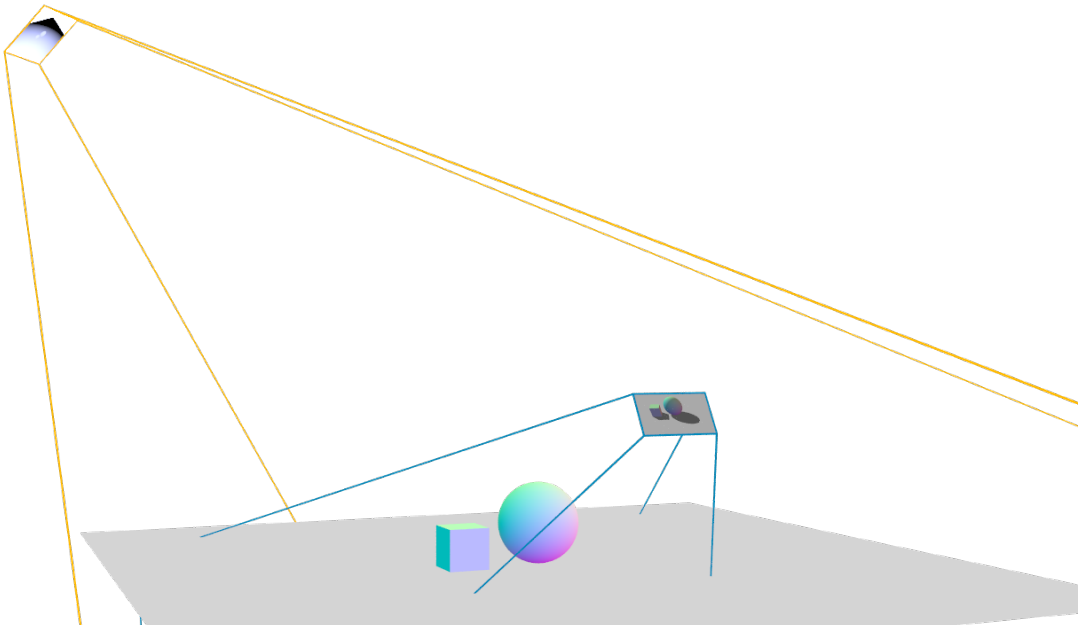

Figure 2 Two pass rendering. The first rendering from the light records the distance from the geometry to the light. The second rendering is the final image whose color is evaluated using information sampled from the first image.

Each of the two cameras has a camera matrix $\mathbf{C}$, view matrix $\mathbf{V}$ (inverse of the camera matrix) and a projection matrix $\mathbf{P}$. Let us call them $\mathbf{C}_{\text{light}}, \mathbf{V}_{\text{light}}, \mathbf{P}_{\text{light}}$ and $\mathbf{C}_{\text{cam}}, \mathbf{V}_{\text{cam}}, \mathbf{P}_{\text{cam}}$ respectively.

In the first pass of rendering (using $\mathbf{V}_{\text{light}}$, $\mathbf{P}_{\text{light}}$), we product an image whose pixel value is the *depth* of the scene. This depth is the *Z value after the multiplication by* $\mathbf{P}_{\text{light}}$ *and dehomogenized (divided by the w coordinate)*. That is, the Z value in light's normalized device coordinate.

Now, store this first pass rendering result in a texture.[5] Of course, as you develop the program, you may want to visualize this first pass rendering. In that case you can show it on the screen for the visualization purpose.

In the second pass of rendering (using $\mathbf{V}_{\text{cam}}$, $\mathbf{P}_{\text{cam}}$), render it as usual (like in HW3). The only difference from HW3 is in the fragment shader where we shade the color.

- If $\mathbf{n} \cdot \mathbf{l} < 0$, the surface is already facing away from the light, so we don't need to invoke the shadow technique.
- If $\mathbf{n} \cdot \mathbf{l} > 0$, then we check whether we are in the shadow or not:
  - Transform the fragment position to the light's normalized device coordinate (multiply by $\mathbf{P}_{\text{light}}\mathbf{V}_{\text{light}}$ from its world coordinate, dehomogenize). This is the coordinate of the fragment in the 1st pass image. This is the fragment's *light space coordinate*.
  - Note that the texture coordinates range in $[0,1]\times[0,1]$, whereas the XY coordinate in the normalized device coordinate has the range of $[-1,1] \times [-1,1]$. A simple rescaling is needed.
  - Sample the depth that was recorded in the texture.
  - Compare the previous depth with the light-space depth of the current fragment.
  - If the sampled depth from the texture is (much) shorter than the new depth value,

[5] See http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-14-render-to-texture/ for direct rendering into a texture buffer.

then we have a shadow.

- Add contribution of the light only when it is not under the shadow. An exception is the ambient component of the lighting. Without a bit of ambient shading, the shadow would look pitch black.

Advanced

As you can see in Figure 1, the edges of the cast shadows have lego/pixelated boundary. This is due to the finite resolution of the image of the first pass rendering. You are encouraged to work on a solution to this problem using a projective transformation. (The principle of shadows is invariant under collinear transformation. As long we map straight lines to straight lines like in projective transformation, the resulting light-visibility will be the same, but with different distribution of resolution!)

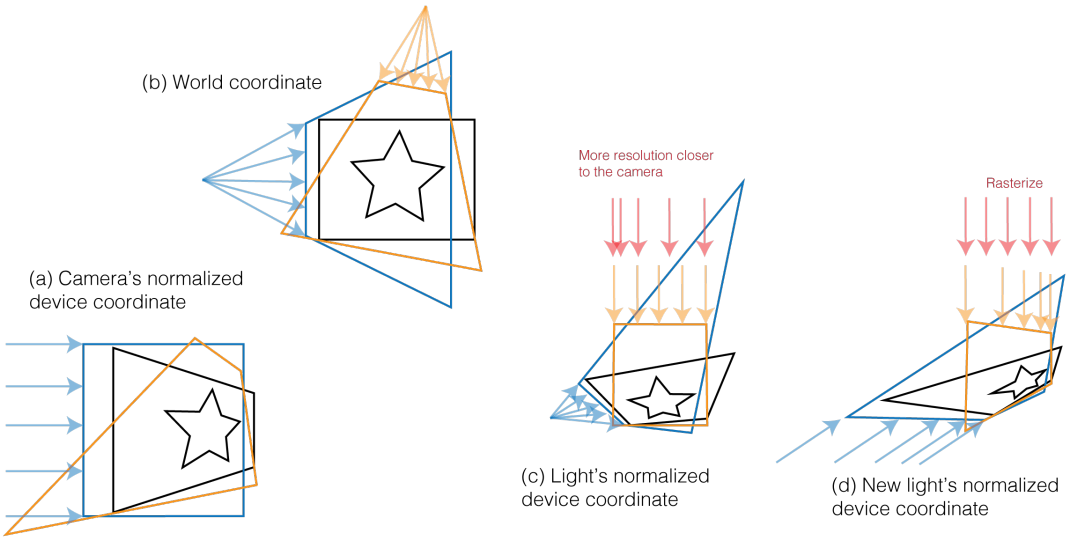

Figure 3 In Light Space Perspective Shadow Map (LiSPSM), one performs an additional perspective transformation on (c) light's normalized device coordinate (NDR) to get (d) a new light's NDR. In the new light's NDR, the Z axis remains the same so that the rasterization visibilities are equivalent. However, the density of pixel is rearranged (red vs orange) so that we have more light-space pixel resolution closer to the camera.

In general, the optimal transformation for the light satisfies the conditions:

- The light rays are parallel to the Z axis in the final light space coordinate, so that the rasterization pipeline still makes sense.
- The camera rays are parallel, so that the correspondence between the resolution of the pixels from the light and the camera is affine and hence uniformity in size.