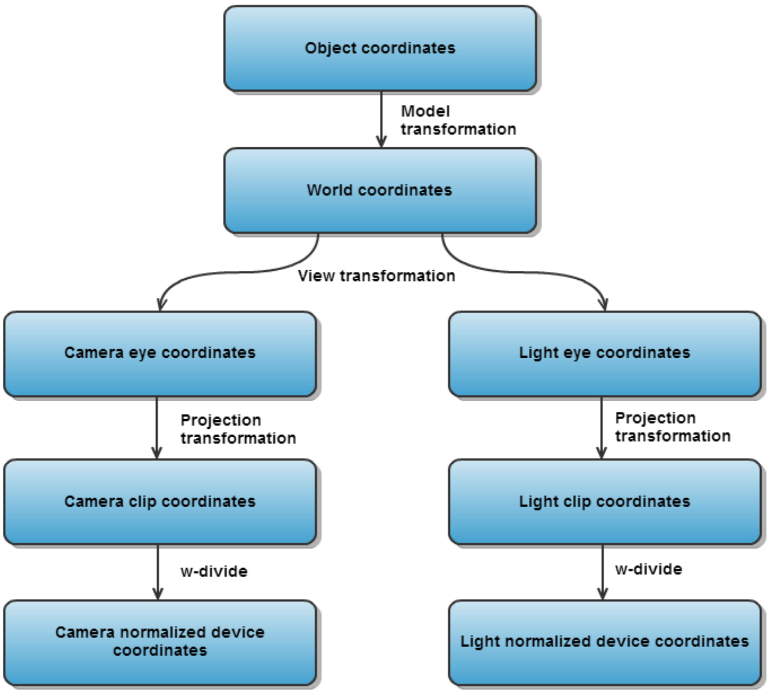
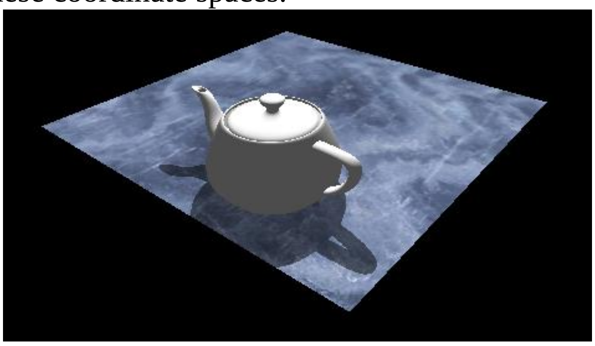
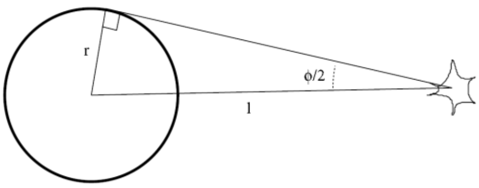
# 02561 COMPUTER GRAPHICS IMM. DTU

## *Exercise 8: Shadow maps*

<table>
<tr><td>Reading</td><td>Angel: Chap. 7.8 – 7.9, 7.1 – 7.4</td></tr>
<tr><td>Introduction</td><td>Shadows are important for our visual perception of an image. The shadow maps we use here is the most simple type of shadow maps, where a shadow texture contains information about if an object is in shadow or not.<br><br>There are two kinds of coordinate spaces used in this assignment: camera relative and light relative. The coordinate spaces and the transformations between these spaces are shown below:<br><br>
</td></tr>
<tr><td>Purpose</td><td>The purpose of this exercise is to understand and implement shadow mapping. This includes a deep understanding of the different coordinate spaces in the pipeline as well as mapping between these coordinate spaces.<br></td></tr>
</table>

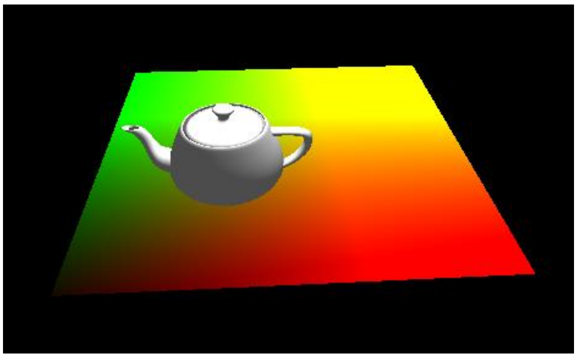
## *Exercise 8: Shadow maps*

| | |
|---|---|
| Part 1<br>Compute shadow projection and view | We need to be able to render the scene from light's point of view. Since the light-source is a point-light we use a Perspective projection. The radius of the teapot is found in the variable<br><br><br>`teapotBoundingRadius`.<br>• Implement the `getLightProjection()` with the correct radius. The near and far plane should not clip the teapot, so the far clip plane should be large (such as 400.0). Hint: Be careful where to use angles in radians vs. degrees<br>• Implement `getLightView()` which should return the matrix that changes the view to the light looking at the teapot. |

## *Exercise 8: Shadow maps*

| | |
|---|---|
| Part 2<br>Render shadow map | We now need to render the teapot to the shadow map. The shadow map we use will have a value of 1 in lit areas and 0 in unlit areas.<br><br>Implement the function `updateProjShadowTexture()`.<br>• Bind framebuffer, set viewport to the size of the shadowmap and clear to white (clear both color buffer and depth buffer).<br>• Change the teapot's color to black (remember to change it back afterwards). Render the teapot using the function `drawMeshObject()`.<br>• Release the framebuffer and set the viewport size to the windows dimensions.<br>• Pressing 'd' will render the content of the shadow map instead of the plane texture. Make sure that your teapot outline is visible like the screenshot below.<br><br> |

## *Exercise 8: Shadow maps*

| | |
|---|---|
| Part 3<br>Compute the shadow map texture coordinates | To render the shadow map at the correct location we need to know the correct UV coordinates from the shadow map when rendering the plane from the cameras point of view.<br><br>• plane.frag contains a mat4 uniform called `lightViewProjection`. Modify the C++ program to transfer the value from the C++ program (the matrix is computed using `getLightViewProjection()` ). You need to modify the `Shader` struct, the `loadShader` function and the `drawMeshObject` function.<br>• Compute the world space coordinates in the plane vertex shader and send the result to the plane fragment shader.<br>• Compute the shadow map normalized device coordinates (`shadowUV`) by transforming the world coordinates into clip coordinates (by multiplying the `lightViewProjection` matrix with the world coordinates) and do the perspective division (divide the result with w).<br>• The normalized device coordinates (ranging from (-1,-1) to (1,1)) should be transformed into texture coordinates (ranging from (0,0) to (1,1)). Use the variable `shadowUV` for the result.<br>• Check the result by visualizing the `shadowUV` instead of the texture lookup in plane.frag (`fragColor=vec4(shadowUV,0.0,1.0);`). The result should look like the image below:<br><br> |

## *Exercise 8: Shadow maps*

| | |
|---|---|
| Part 4<br>Combine shadow map with texture | In this final part we will use the shadow texture coordinate and lookup in the shadow map texture to find out if the fragment is in shadow.<br>• Add the shadow map as a uniform to the plane.frag and bind the shadowmapTextureId in the C++ program.<br>• Use the shadow map texture coordinates in the plane.frag to find the shadow color.<br>• Change the shadow color to be a value of either 0.5 or 1.0 (it is currently 0.0 or 1.0)<br>• Set fragColor to the shadow color multiplied with the color from the texture lookup.<br>You should now see shadows projected onto the plane under the teapot. |
| Part 5<br>Smooth shadows<br>Optional | Currently we see hard shadows, since the shadow map contains a boolean value. Try to implement smooth shadows by taking the average of values close the shadow map texture coordinates instead of the single lookup in the shadow map. This technique is called Percentage Closer Filtering. |