Point Shadows

University of Applied Sciences and Arts
of Southern Switzerland

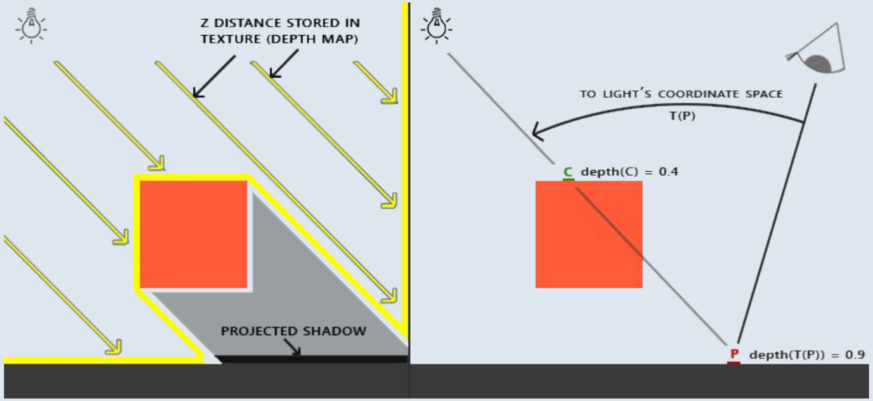
SUPSI

Advanced Computer Graphics

Mattia Dell’Oca and Jaspera Rohner

Point Shadow Basics

Recap: One-dimensional Shadow Mapping



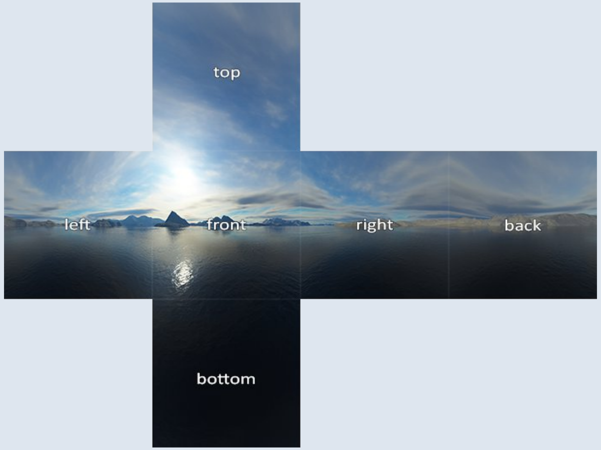
⇒ Works only for spot lights, because it is generated in one direction.

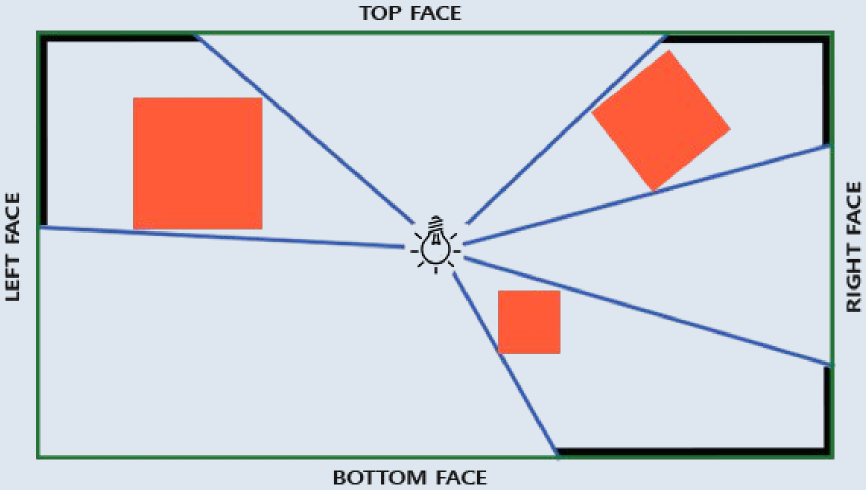
⇒ Doesn't work with point lights!

Point Shadow Basics

Idea: Place the observer into a shadow map skybox

⇒ We have shadows in all directions
⇒ Now we can support point lights





Implementation: Code additions

1) Create a new type of Eng::Texture depth_cube with 6 depth maps

2) We added a new Eng::Pipeline subclass named Eng::PipelineSkybox. This pipeline only renders the skybox.

⇒ The shadow map rendering is still completed using Eng::PipelineShadowMapping!

3) We add a geometry shader to Eng::PipelineShadowMapping because we need to map shadows to 6 different faces

Implementation: Geometry Shader

⇒ The Geometry Shader is in between the vertex and fragment shaders

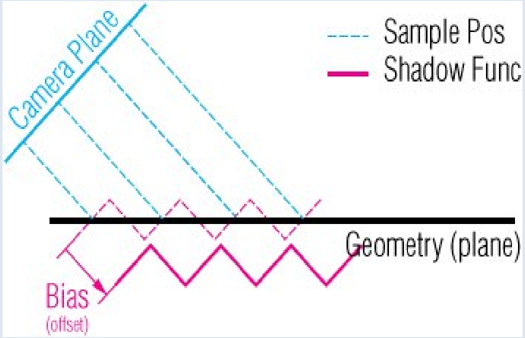
Code for geometry shader:

```
for(int face = 0; face < 6; ++face)
{
    gl_Layer = face;

    for(int i = 0; i < 3; ++i)
    {
        FragPos = gl_in[i].gl_Position;
        gl_Position = shadowMatrices[face] * FragPos;
        EmitVertex();
    }
    EndPrimitive();
}
```

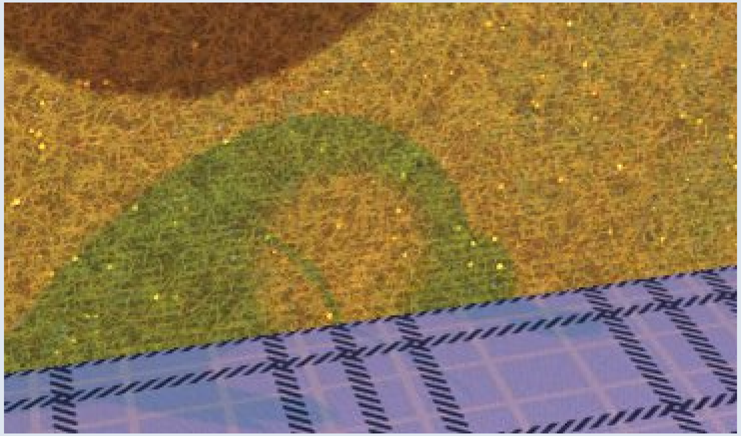
Artifacts: Acne and Peter panning

Acne: Caused by the fact that shadow maps are discrete and surfaces and not

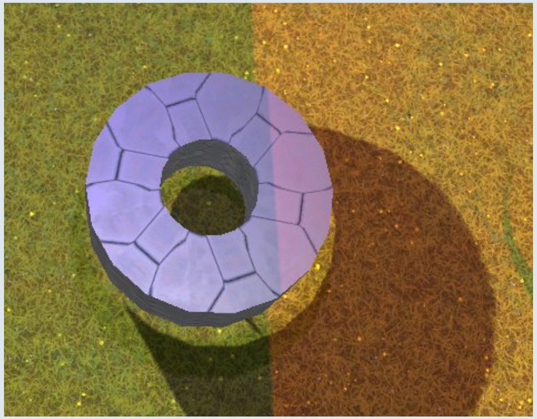


Downfalls: Offsetting the shadow function too far disassociates the shadow from the object!
⇒ Peter panning

Acne:

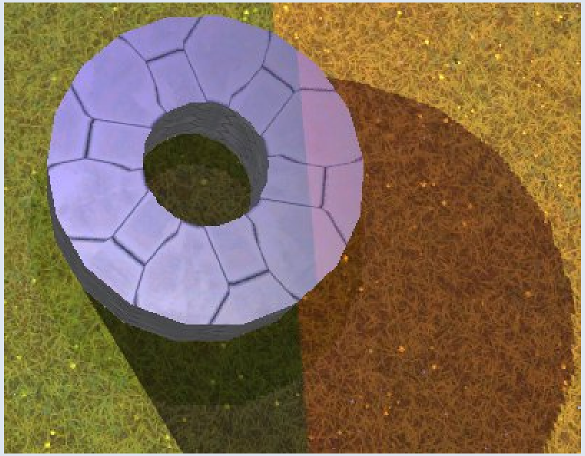


Peter panning:



Artifacts: Aliasing

Another victim of the discreteness of shadowmaps: **Aliasing**
Aliasing is when you can see the discrete pixels of the shadow map



We know to soften shadows for regular z-coordinate shadow maps. How do we solve aliasing for skybox shadow maps?

Percentage-closer Filtering

⇒ Jagged edge shadows don’t look good
⇒ Either we can increase the cubemap resolution, or implement PCF

Basic Algorithm:

```
For every shadow map value d:
        Loop over $n^3$ adjacent depth values $d_{neighbor}$:
                $d \mathrel{+}= d - bias > d_{neighbor}$
        $d \mathrel{/}= n^3$
```

⇒ This algorithms loops over n^3 values! Most of these values are very similar. Better Idea: Loop over only one pixel per direction

Demo

Image Sources:

All images in Point Shadow Basics Chapter:

Joey de Vries of LearnOpenGL.com

Acne Description Picture:

User joojaa of Stackoverflow: https://computergraphics.stackexchange.com/questions/2192/cause-of-shadow-acne

All other images are attributed to Mattia and Jaspera