universität
innsbruck

Shadow Mapping

Visual Computing – Final Project

Matthias Ebner, Frederik Hirsch, Paul Prünster

Introduction

Question

Why does this scene look unrealistic?

Introduction

Question

Why does this scene look unrealistic?

- poles cast no shadow

Introduction

Question

Why does this scene look unrealistic?

- poles cast no shadow
- picture is “real”, but taken when the sun is directly above poles

Introduction

Question

Why does this scene look unrealistic?

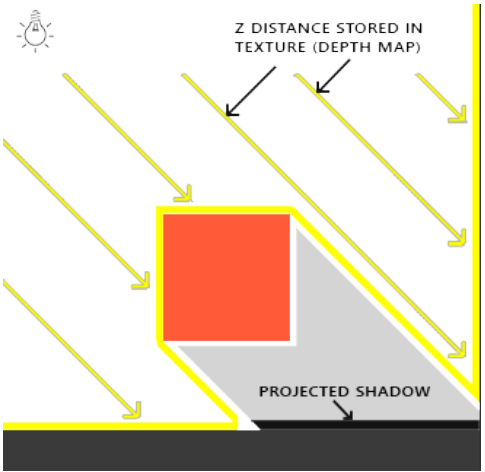
- poles cast no shadow
- picture is “real”, but taken when the sun is directly above poles

⇒ missing shadows can make otherwise realistic scene look unrealistic

Introduction

Shadow Mapping – General Idea

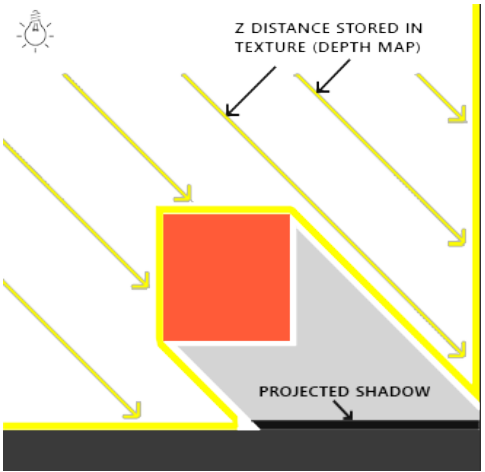
1. render scene from lights perspective



Introduction

Shadow Mapping – General Idea

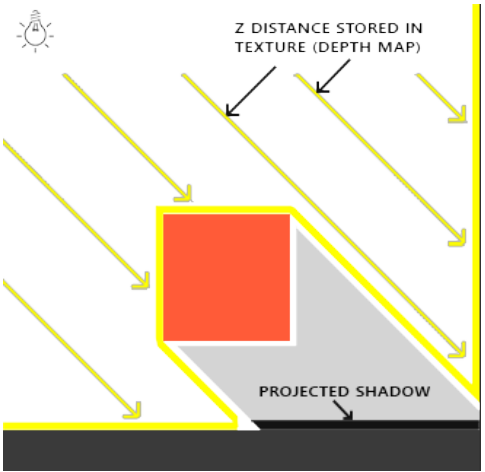
1. render scene from lights perspective
   - everything that is visible is hit by light
   - everything that is not visible is not hit by light and therefore in the shade



Introduction

Shadow Mapping – General Idea

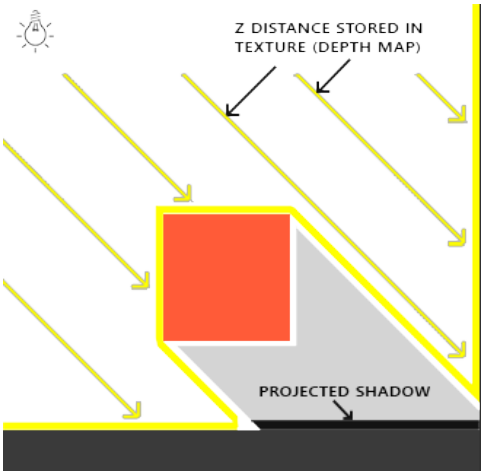
1. render scene from lights perspective
   - everything that is visible is hit by light
   - everything that is not visible is not hit by light and therefore in the shade
2. generate depth map from first rendering



Introduction

Shadow Mapping – General Idea

1. render scene from lights perspective
   - everything that is visible is hit by light
   - everything that is not visible is not hit by light and therefore in the shade
2. generate depth map from first rendering
3. render scene from camera perspective using the depth map to make areas darker that are not visible from light perspective



Overview

Creating the Depth Map

Rendering Shadows with the Depth Map

Shadow Acne

Anti Aliasing

Creating the Depth Map

Transforming the scene to light perspective

same as normal, just with orthographic projection and light direction instead of camera direction:

```
glm::vec3 lightInvDir = glm::vec3(1.0f,2,2);
glm::mat4 depthProjectionMatrix
    = glm::ortho<float>(-10,10,-10,10,-10,20);
glm::mat4 depthViewMatrix
    = glm::lookAt(lightInvDir, glm::vec3(0,0,0), glm::vec3(0,1,0));
glm::mat4 depthModelMatrix
    = glm::mat4(1.0);
glm::mat4 depthMVP
    = depthProjectionMatrix * depthViewMatrix * depthModelMatrix;
```

Creating the Depth Map

Vertex Shader

pass through shader, only convert vertices from model space to camera space (light space):

```
#version 330 core

layout(location = 0) in vec3 vertexPosition_modelspace;
uniform mat4 depthMVP;

void main(){
        gl_Position =  depthMVP * vec4(vertexPosition_modelspace,1);
}
```

Creating the Depth Map

Fragment Shader

write the depth of our fragment to the depth texture at location 0:

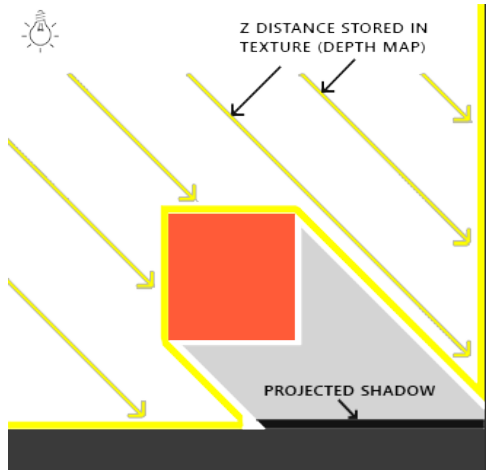
```
#version 330 core

layout(location = 0) out float fragmentdepth;

void main(){
    fragmentdepth = gl_FragCoord.z;
}
```

Creating the Depth Map

Rendering Shadows with the Depth Map

- for each fragment, check if its depth is larger than the one stored in the depth texture
- if yes, then something is in front of it

⇒ make that fragment darker to simulate shadow



Rendering Shadows with the Depth Map

Vertex Shader

calculate the position of that vertex on the depth map, by transforming it to light space, and mapping it from $(-1, 1)$ *to* $(0, 1)$:

```
...
// Output position of the vertex, in camera space : MVP * position
gl_Position =  MVP * vec4(vertexPosition_modelspace,1);

// Same, but with the light's view matrix
ShadowCoord = DepthBiasMVP * vec4(vertexPosition_modelspace,1);
...
```

Rendering Shadows with the Depth Map

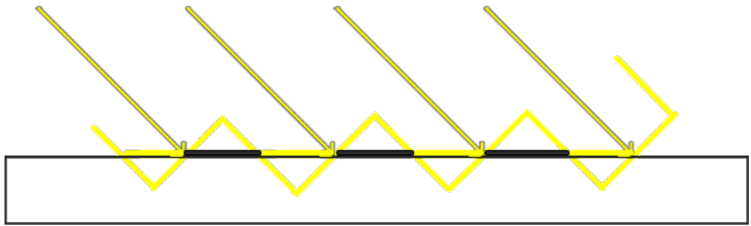
Fragment Shader

retrieve the depth of nearest object to light from depth texture and compare it with the fragment depth:

```
...
float visibility = 1.0;
if ( texture( shadowMap, ShadowCoord.xy ).z  <  ShadowCoord.z){
    visibility = 0.5;
}
FragColor = vec4(visibility * MaterialDiffuseColor * LightColor, 1.0);
...
```

Shadow Acne

- striped shadow lines where everything should be in the light:

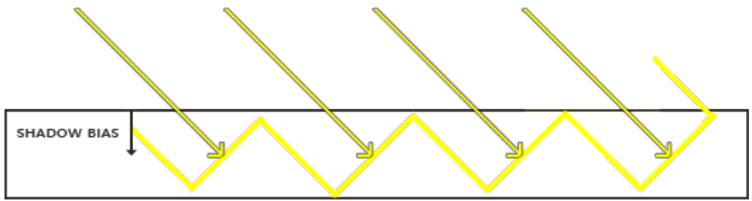


⇒ Solution: subtract small *bias* from depth of the current fragment, so only fragments that are further behind another object are shaded.

Shadow Acne

Solution

```
float bias = 0.005;
float visibility = 1.0;
if ( texture( shadowMap, ShadowCoord.xy ).z  <  ShadowCoord.z - bias){
    visibility = 0.5;
}
```



Aliasing

- depending on the resolution of the shadow map, individual texels of the shadow map are still visible

- higher resolution of depth texture helps
- when that is not possible, we can sample the map multiple times at different points close by, and take the average (poisson sampling)

Aliasing

Poisson Sampling

```
vec2 poissonDisk[4] = vec2[](
  vec2( -0.94201624, -0.39906216 ),
  vec2( 0.94558609, -0.76890725 ),
  vec2( -0.094184101, -0.92938870 ),
  vec2( 0.34495938, 0.29387760 )
);
for (int i=0;i<4;i++){
  if (texture(shadowMap, ShadowCoord.xy + poissonDisk[i]/3000.0).z
      <  ShadowCoord.z-bias){
    visibility-=0.2;
  }
}
```

Thank you for your attention!

Matthias Ebner, Frederik Hirsch, Paul Prünster