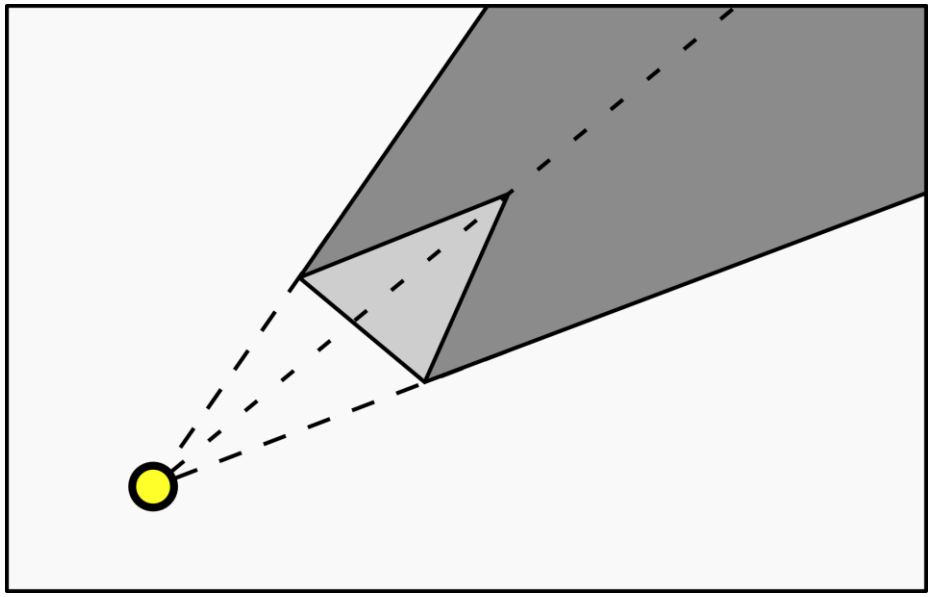
Dynamic Shadow Rendering in 2D

Speaker: Christoph Neuhauser

Technische Universität München (TUM)



Two Rendering Approaches

(1) Shadow Volumes

- Render extruded occluders

(2) Shadow Mapping

- Render scene seen from light

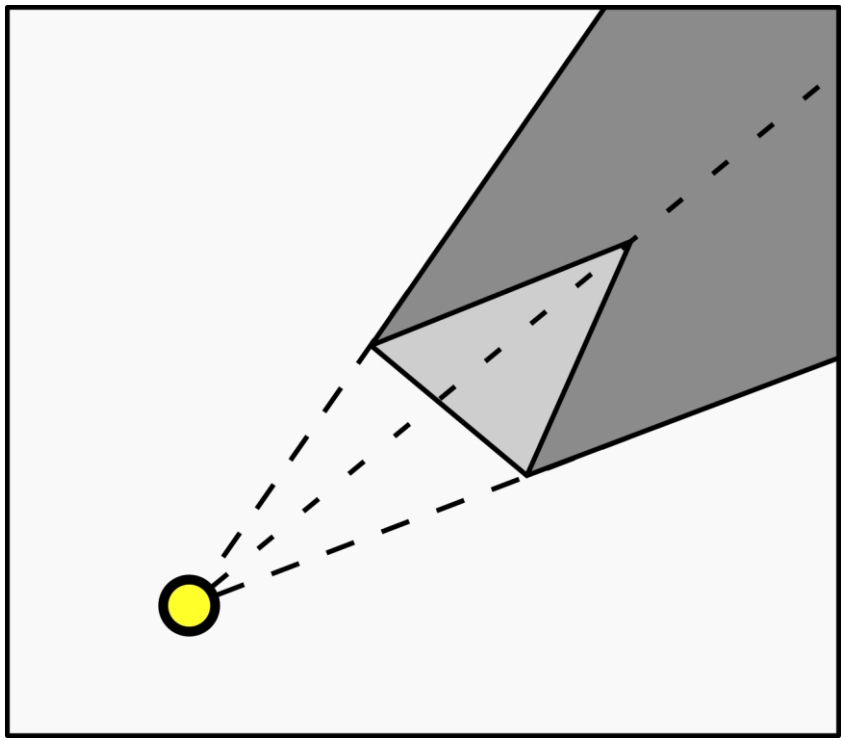
(3) Performance Comparison

1. Shadow Volumes in 2D – The Idea

- **Extrude** vertices of object edges away from light

Geometry shader:

- Extrude object outlines away from light into **infinity**

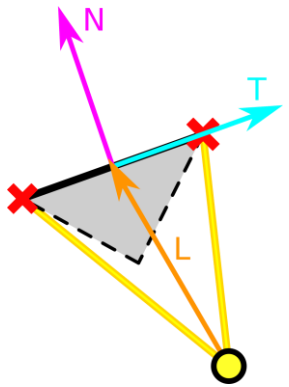


1. Shadow Volumes in 2D - Pros and Cons

Pseudo-Code **Geometry-Shader** in GLSL (Input: edge points pt0, pt1):

```
vec2 tangent = pt1 - pt0;
vec2 normal = normalize(vec2(-tangent.y, tangent.x));
vec2 lightdir = normalize((pt0 + pt1) / 2.0f - lightpos);

// Facing away from light?
if (lightdir ∘ normal < 0):
    // Extrude points to infinity:
    vec4 pt_inf0 = vec4(pt0 - lightpos, 0., 0.);
    vec4 pt_inf1 = vec4(pt1 - lightpos, 0., 0.);
    // ...
```

Homogenous coordinates



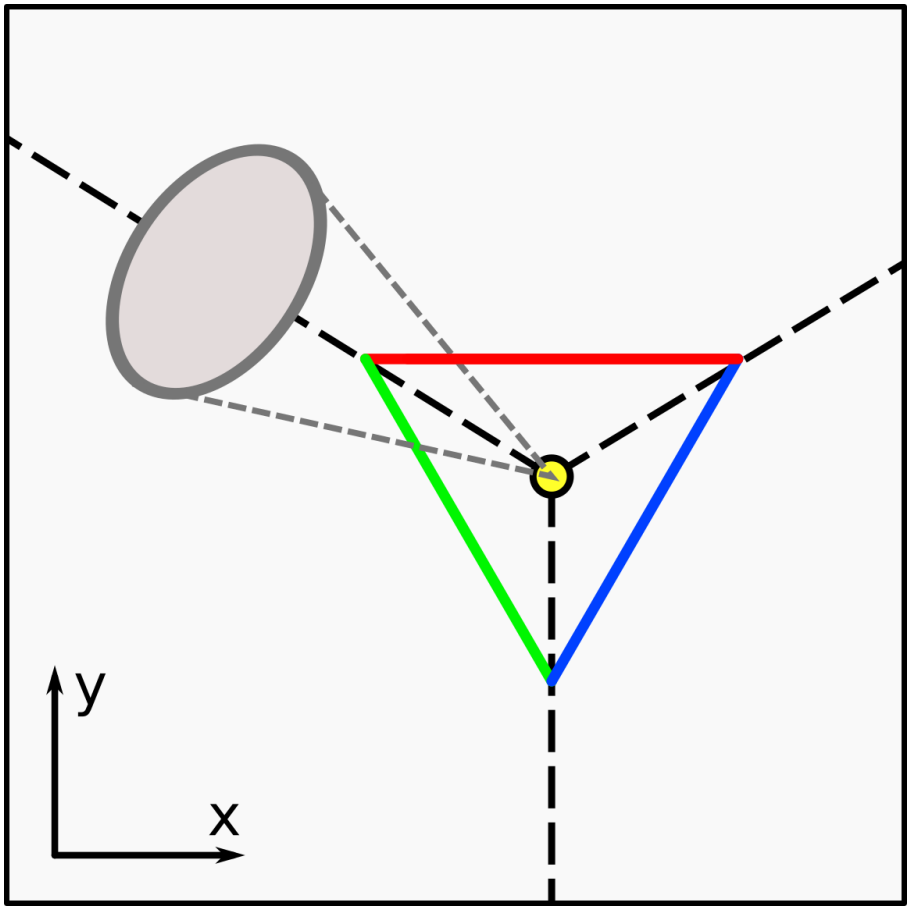
2. Shadow Mapping in 2D

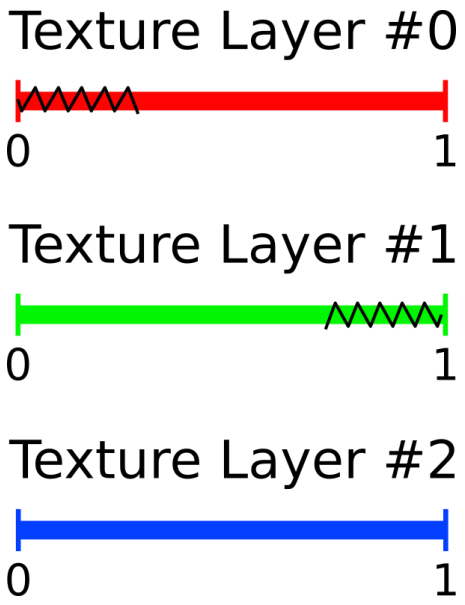
Two steps:

(a) Render scene from light view ➔ **Shadow map**

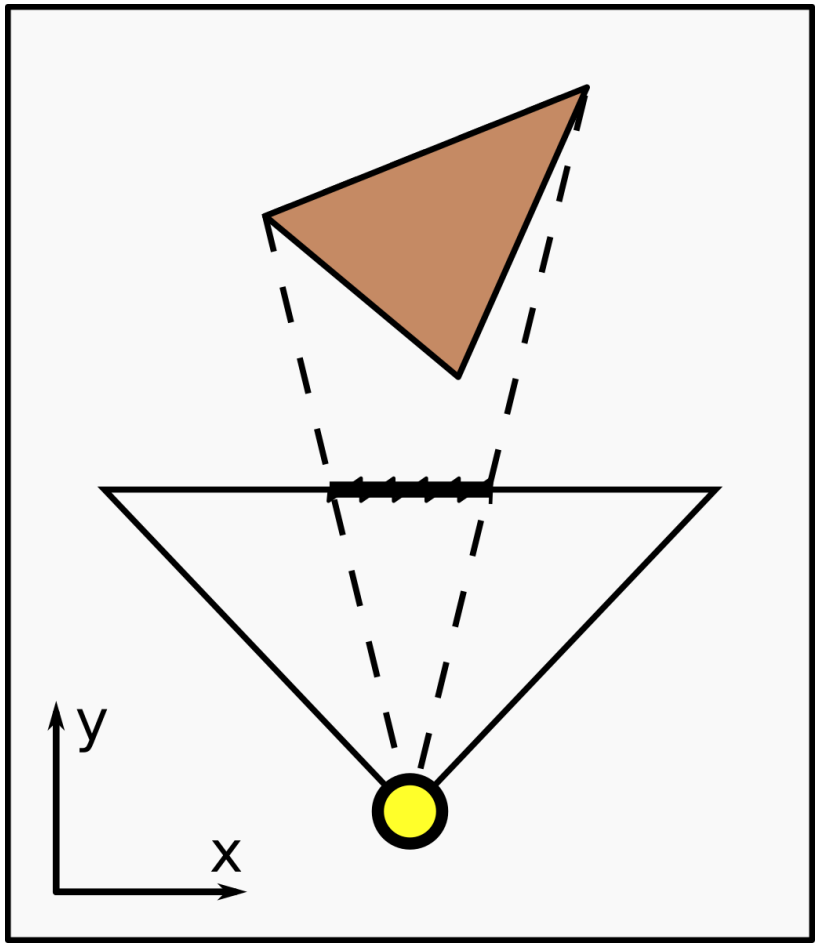
(b) Apply shadow map ➔ **Lit/Shadowed scene**

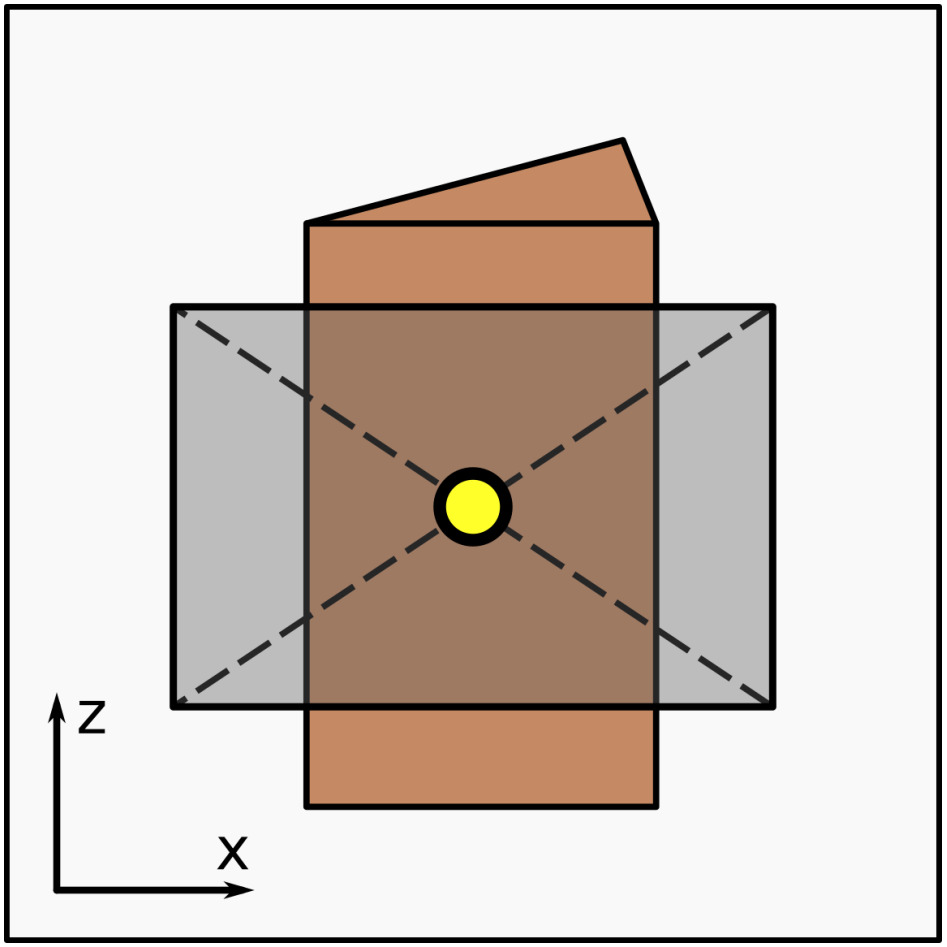
(a) Render scene from light view





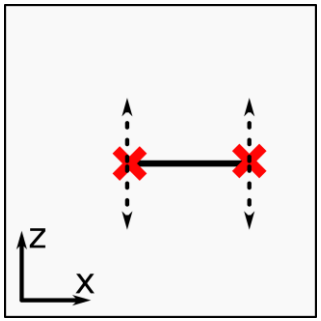
(a) Render scene from light view





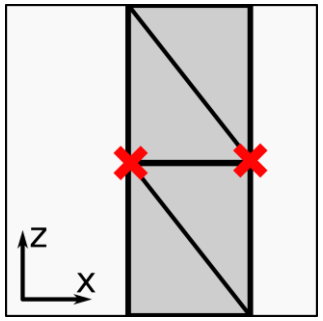
(a) Render scene from light view

Object edges
(lines, two vert.)



Geometry Shader

Shadow volumes
(quads/2 triangles)



Pseudo-Code **Geometry-Shader**:

```
foreach layer in {0,1,2}:
    gl_Layer = layer;
    // extrude edges in z (facing away from light):
    vec4 posz_inf = vec4(0, 0, +1, 0);
    vec4 negz_inf = vec4(0, 0, -1, 0);
    // ... create triangles with pt0, pt1 ...
```

(a) Render scene from light view

Pseudo-Code **Fragment-Shader**:

```
float lightDist = length(fragpos - lightpos);
gl_FragDepth = lightDist / farPlaneDist;
```

→ Map light distance to [0;1]

(b) Shadow Map → Lit Scene

Pseudo-Code **Fragment-Shader**:

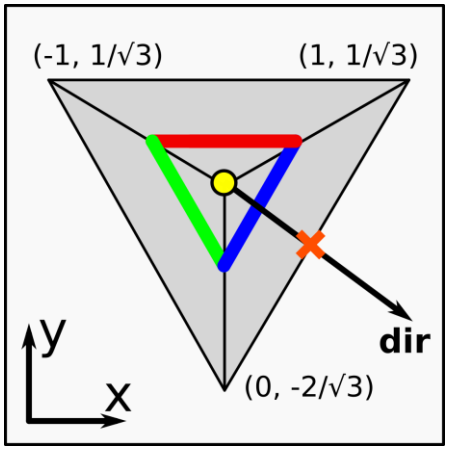
```
vec2 dir = fragpos - lightpos;
float lightDist = length(dir);
float occlusionDepth = getDepthMapVal(dir);

if (lightDist > occlusionDepth - BIAS)
    → BLACK
else
    → COLOR
```

(b) Shadow Map → Lit Scene

How does `float getDepthMapVal(vec2 dir)` work?

1. Compute index of right depth map

$$index(dir) = \left(\left\lfloor \frac{atan2(dir.y,\ dir.x) + \frac{11}{6}\pi}{\frac{2}{3}\pi} \right\rfloor \right) mod\ 3$$

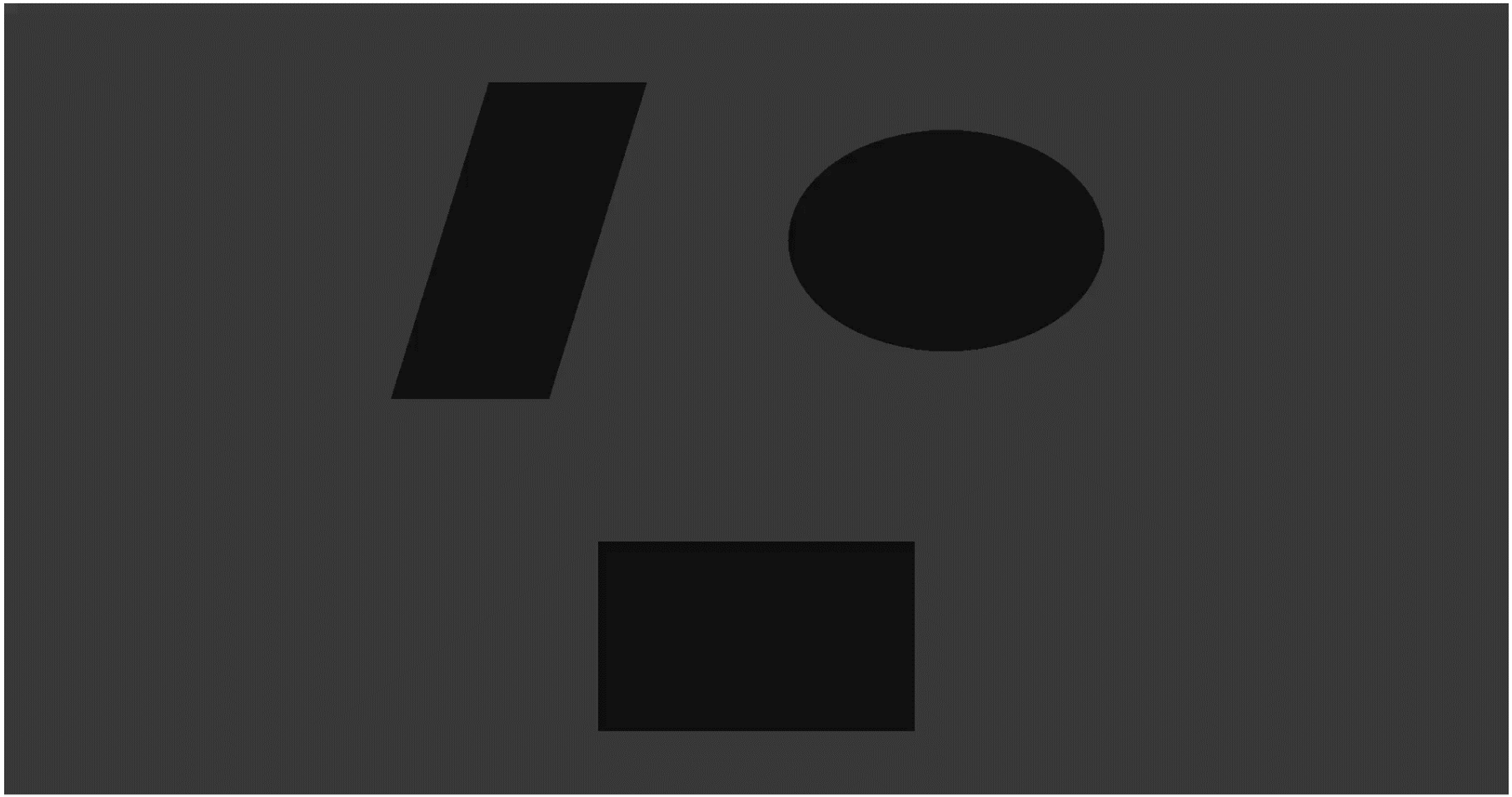


2. Compute texture coordinate *s* in x direction

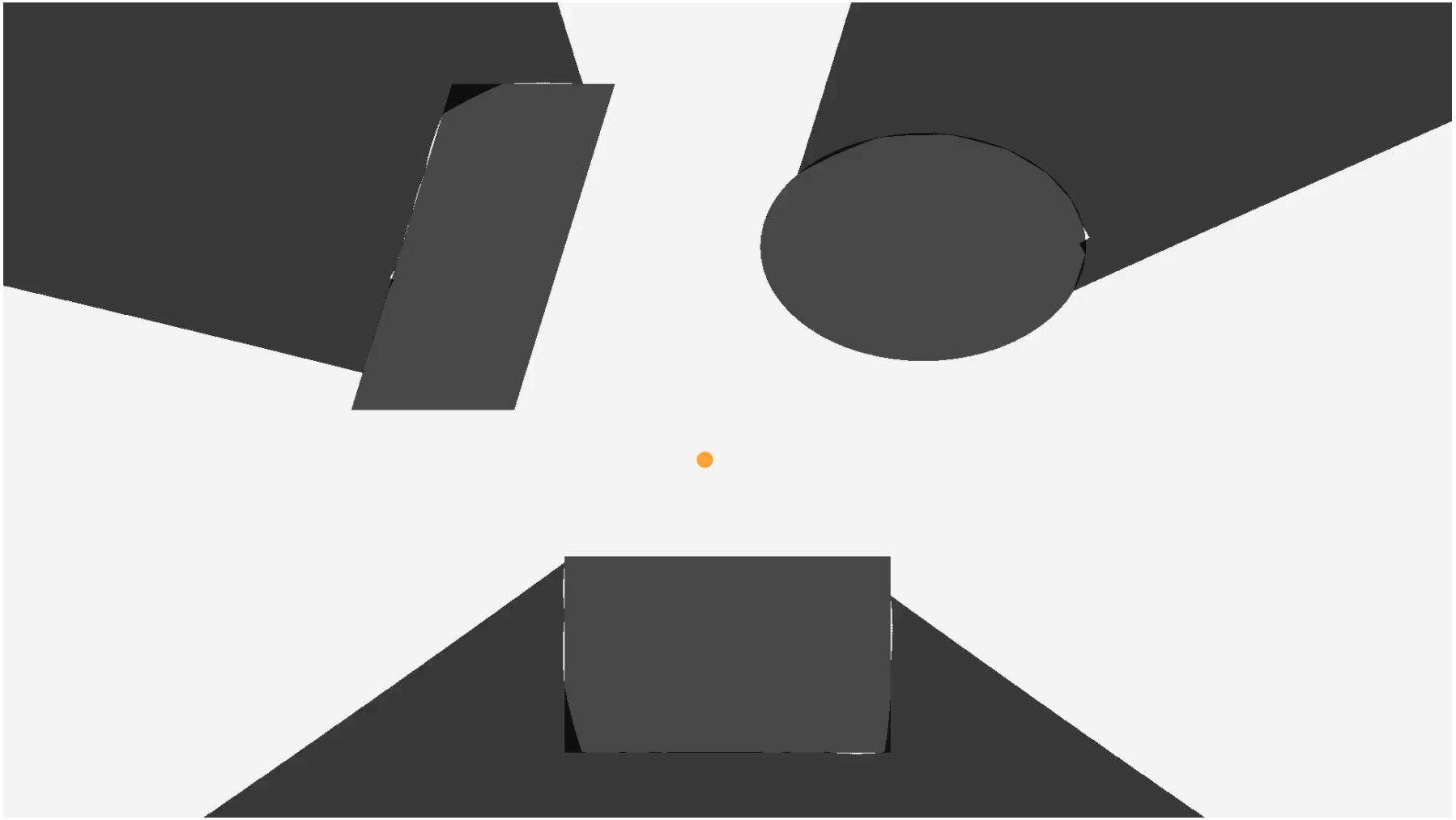
$$\texttt{float s} = \left(\frac{dir.x}{\sqrt{3} \cdot dir.y} + 1 \right) / 2$$

(For index==1/2: Rotate *dir* by 240°/120° beforehand)

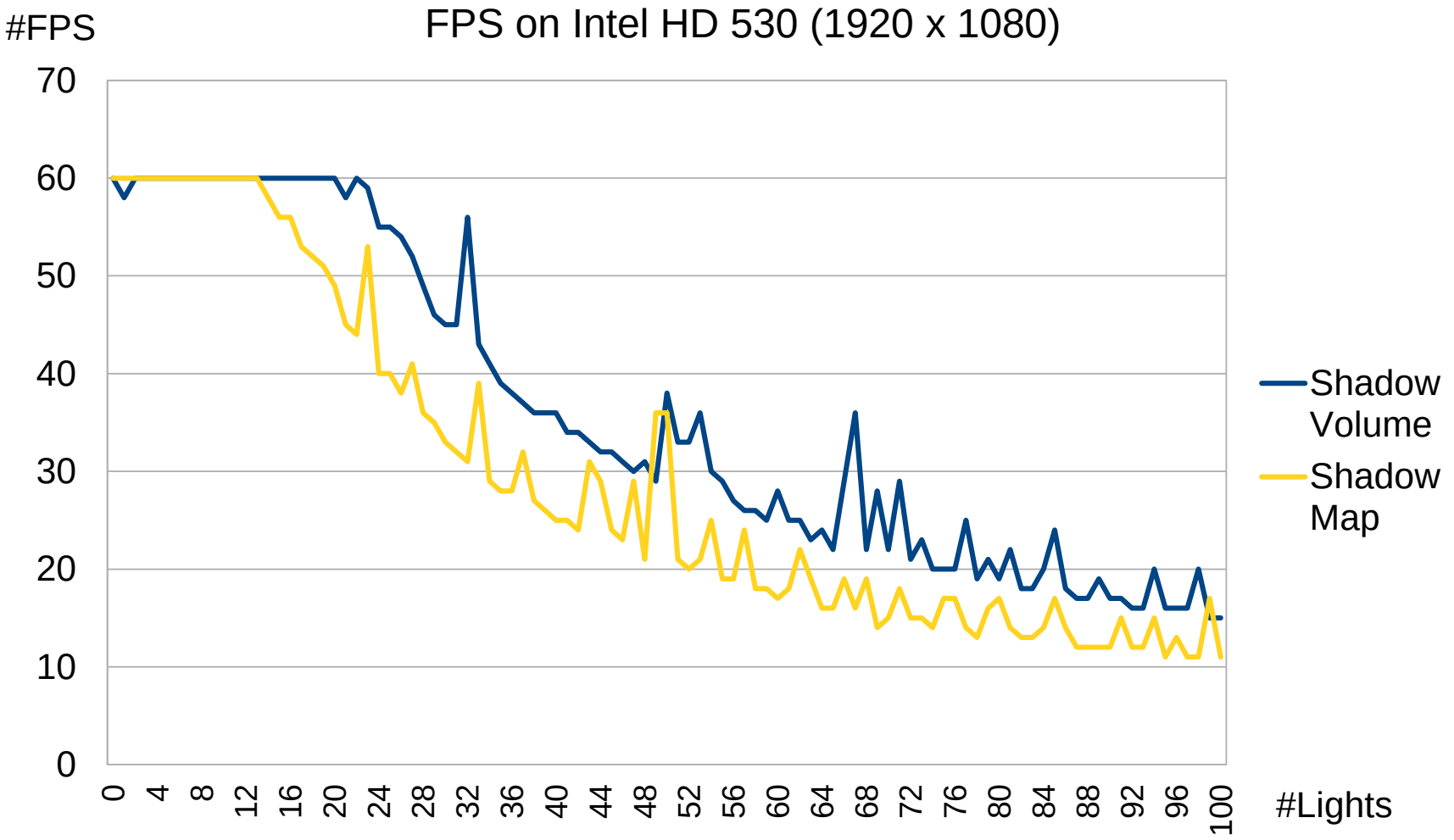
3. Video (both algorithms visually equal)



3. Video (low-res shadow mapping, 3x16px)



3. Comparison



4. Sources

- Slides from the lecture *Image Synthesis* at TUM, Rüdiger Westermann, 2017
  http://wwwcg.in.tum.de/teaching/teaching/winter-term-1718/image-synthesis/material.html
- https://csantosbh.wordpress.com/2014/03/29/2d-omnidirectional-shadow-mapping-with-three-js/

Any questions?