COSC342:
Computer Graphics

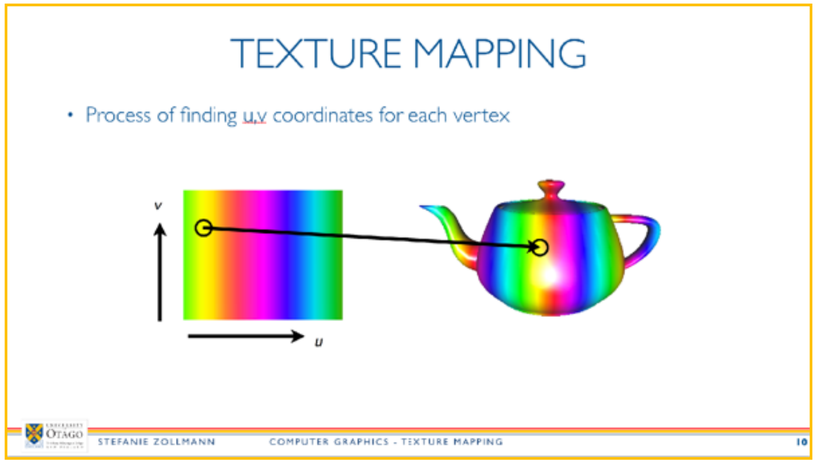
2017

Lecture 19

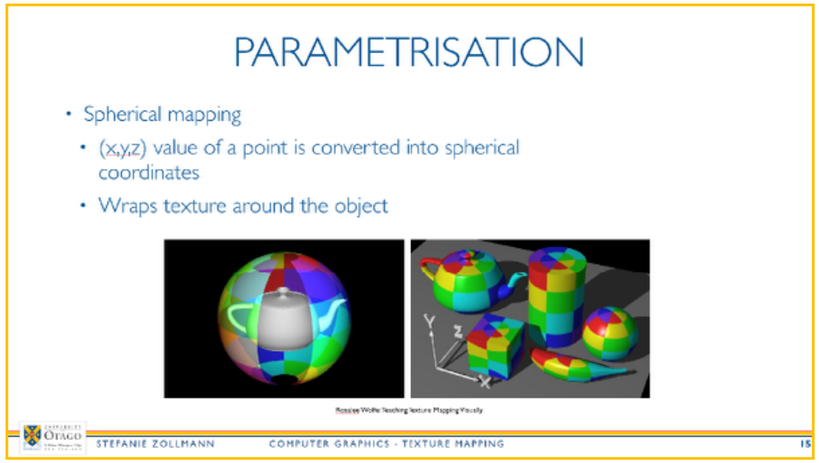
SHADOWS

Stefanie Zollmann

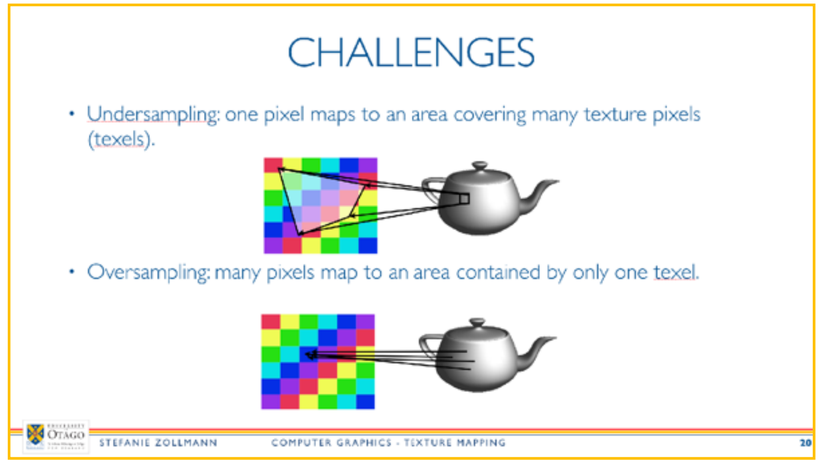
LAST TIME



Texture Mapping

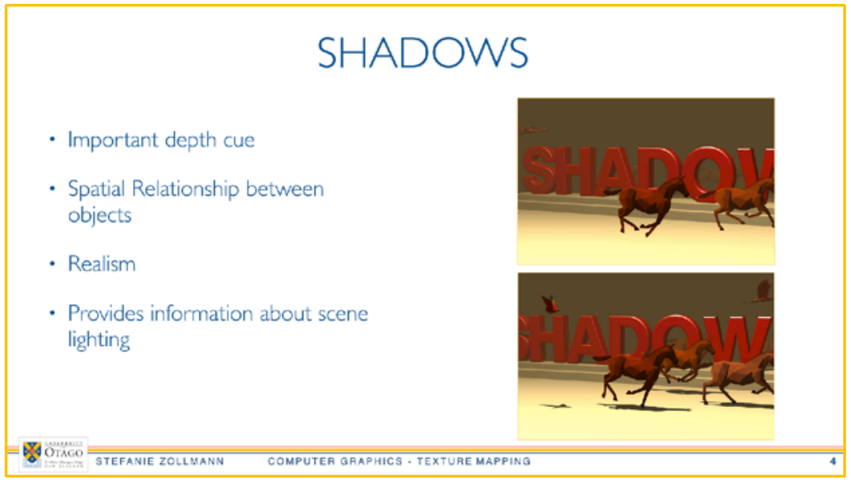


Parametrisation

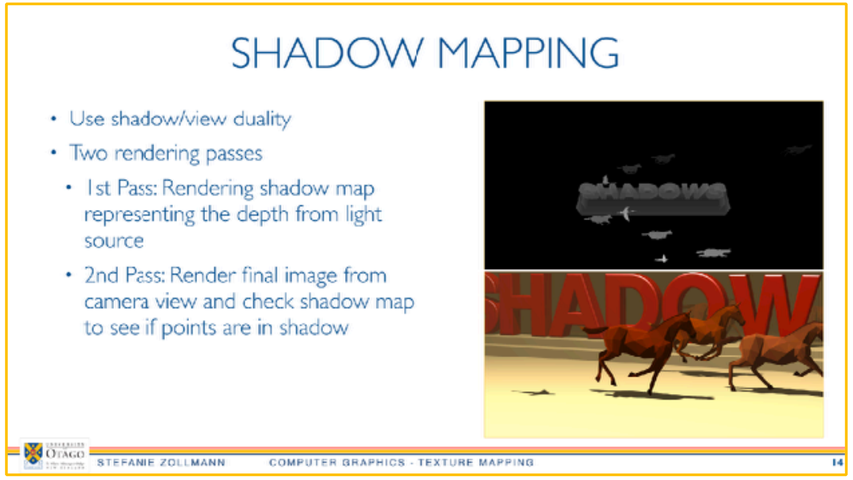


Challenges

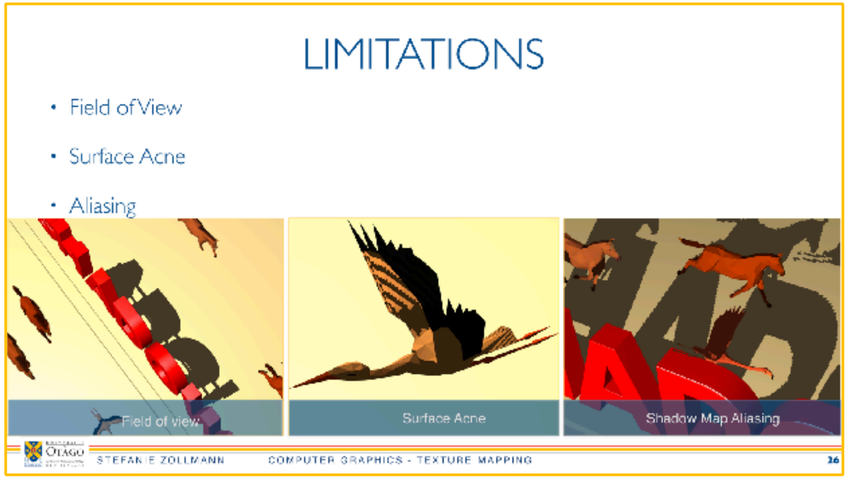
TODAY



Shadows



Methods



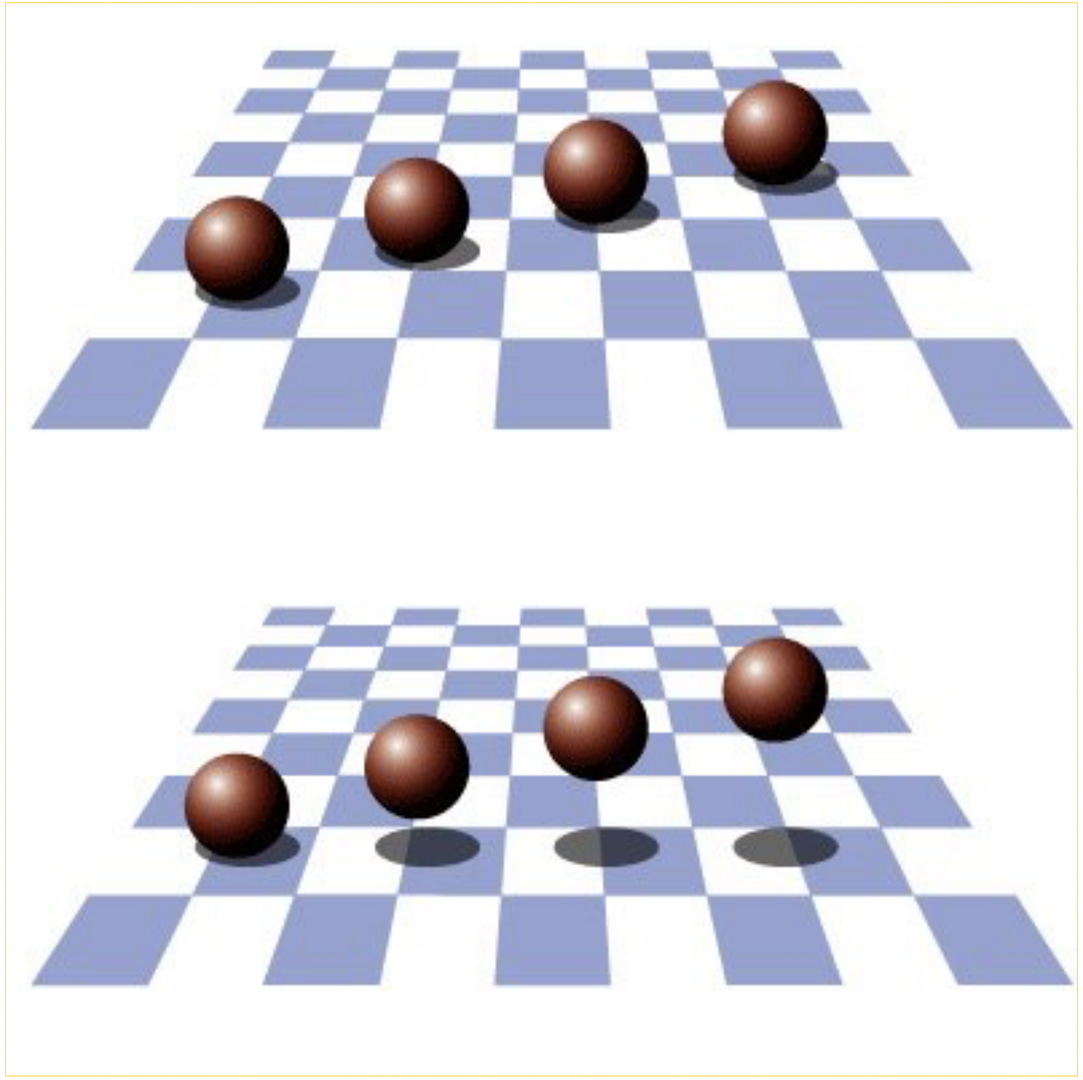
Limitations & Improvements

SHADOWS

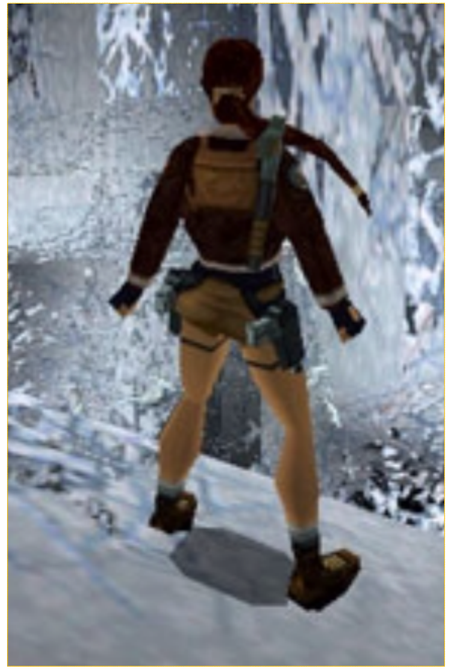
- Important depth cue
- Spatial Relationship between objects
- Realism
- Provides information about scene lighting

SHADOWS AS DEPTH CUE

- Important depth cues
- Provide information about relative locations



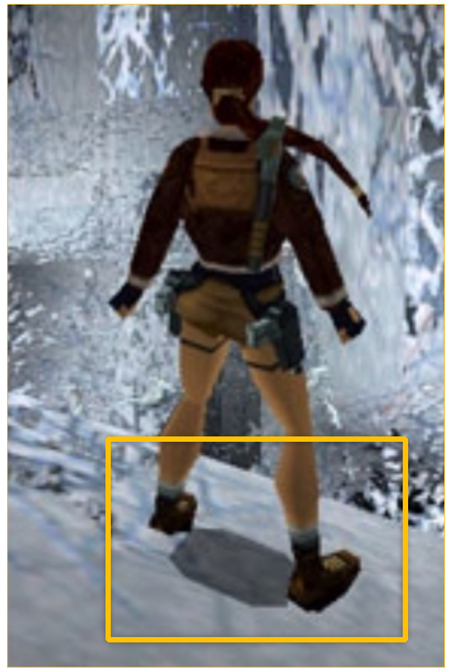
SHADOWS FOR REALISM



Examples:

- Missing Shadows
- Wrong Shadows

SHADOWS FOR REALISM

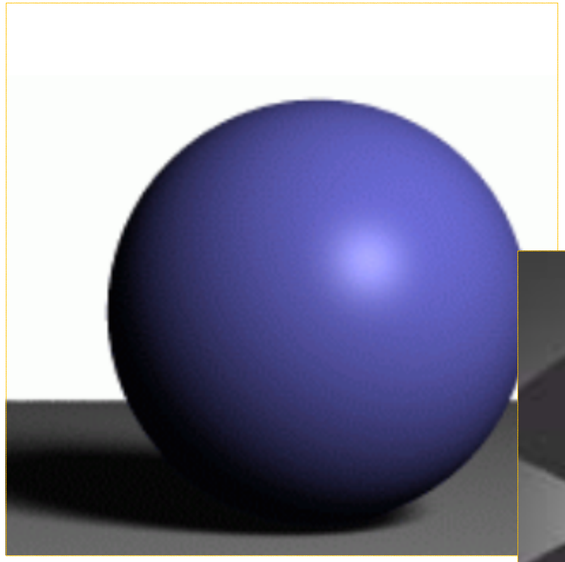


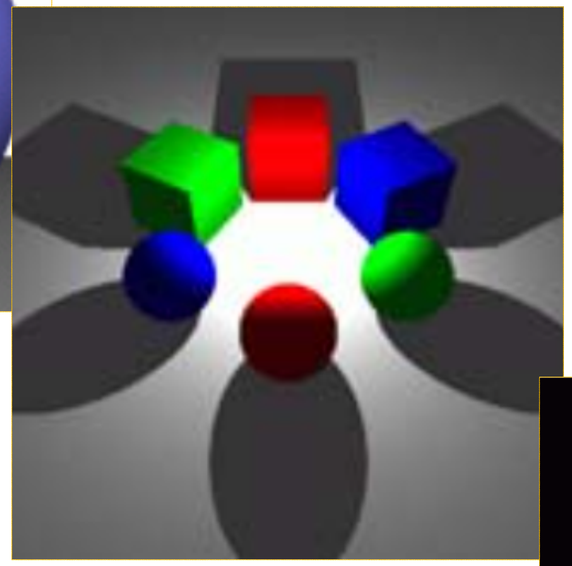
Examples:

- Missing Shadows
- Wrong Shadows

SCENE LIGHTING

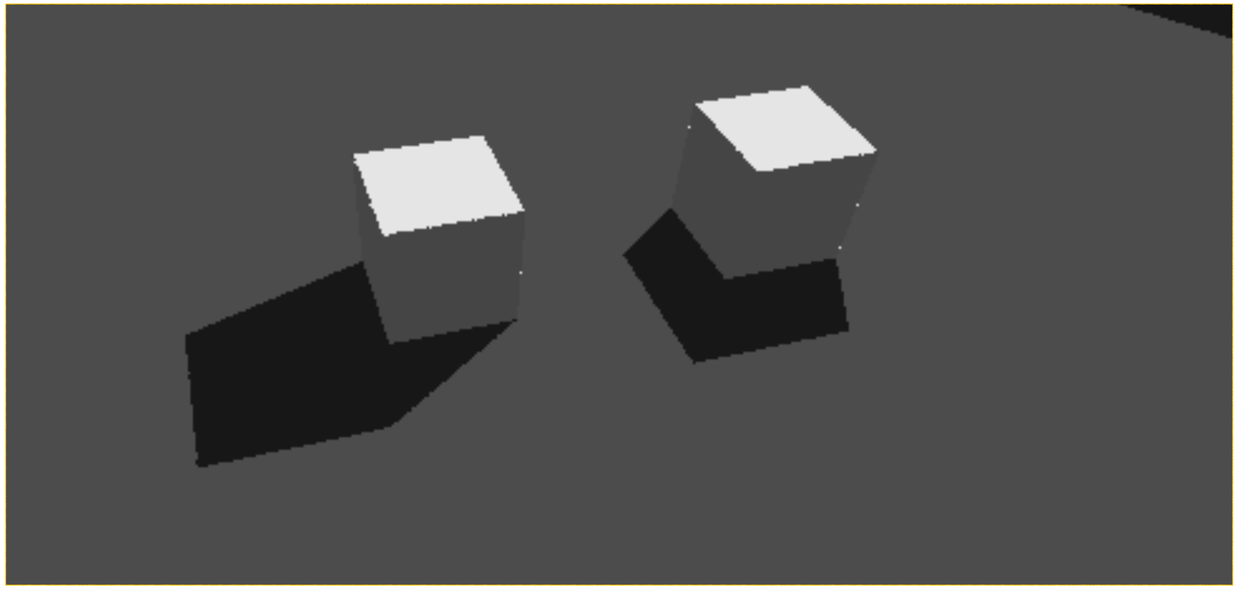
- Position of light sources
- Types of lights:
  - Hard shadows vs. soft shadows
  - Point lights have hard edges
  - Area lights have soft edges
- Coloured light sources



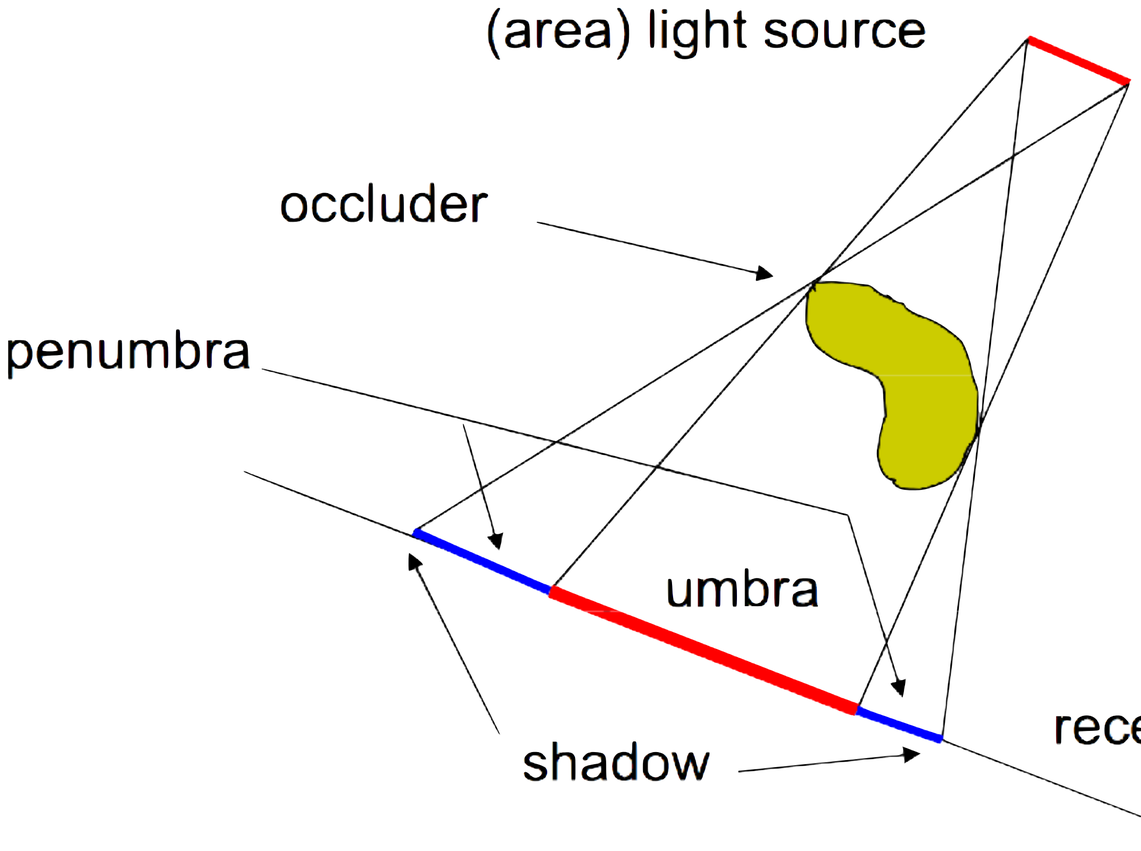


WHAT ARE SHADOWS?

- Shadows are areas hidden from a light source
- Surface is only illuminated if nothing blocks its view of the light



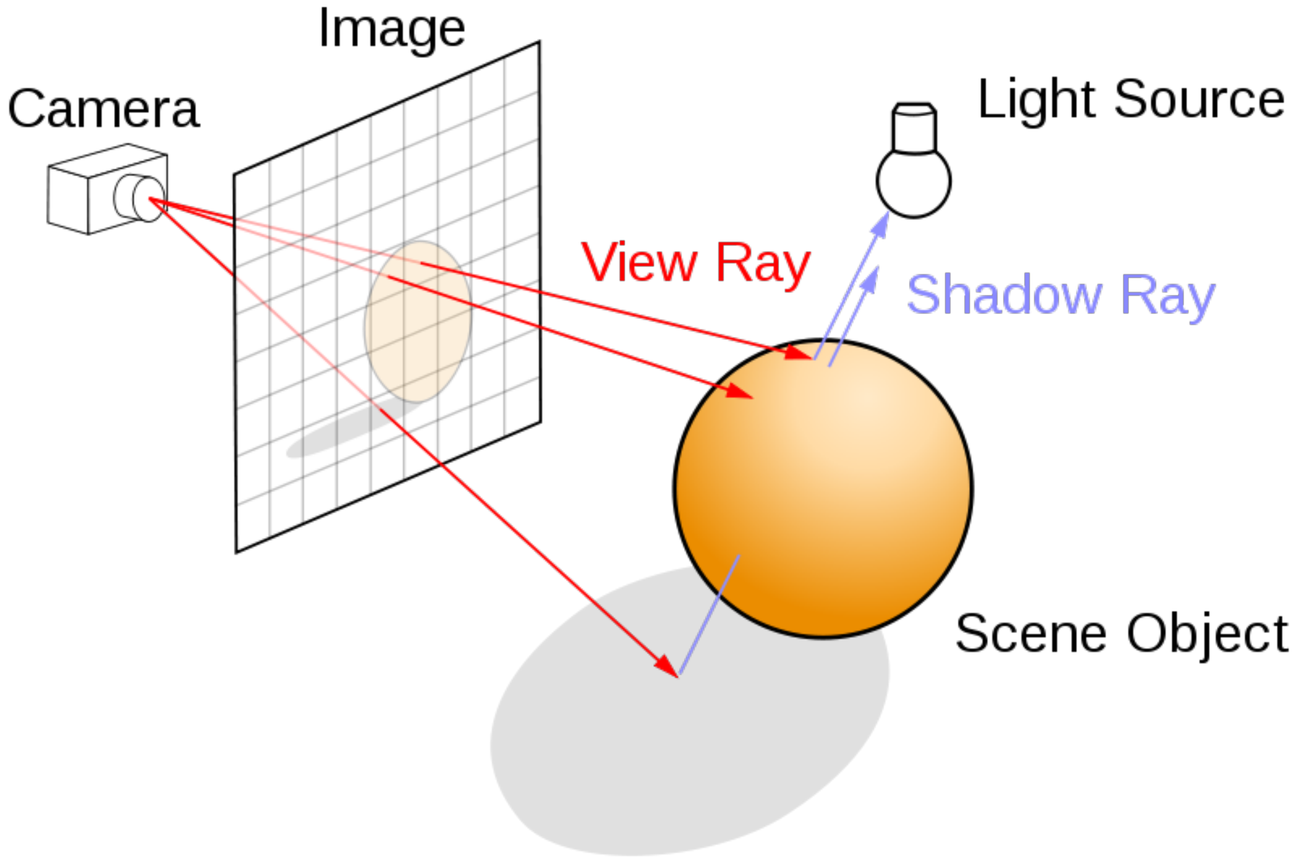
TERMINOLOGY



- **umbra** – fully shadowed region
- **penumbra** – partially shadowed region

RECAP FROM RAYTRACING

- Shoot a secondary ray from visible point to light source
- If the closest hit point is smaller than the distance to the light then the point is in shadow

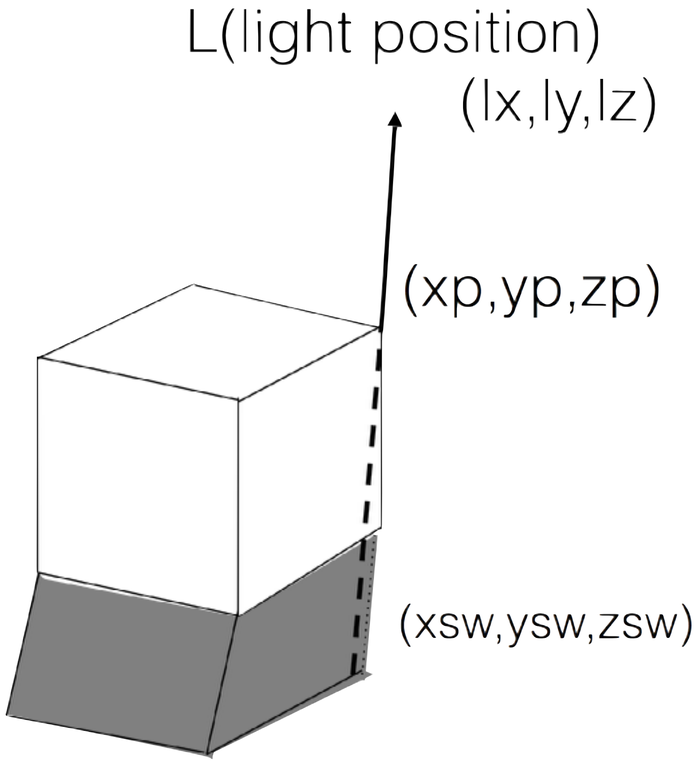


http://upload.wikimedia.org/wikipedia/commons/8/83/Ray_trace_diagram.svg

METHODS

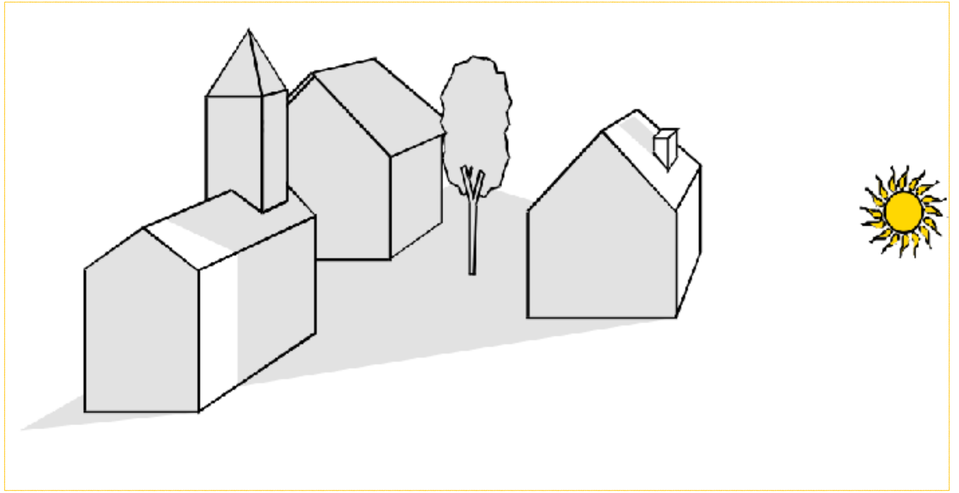
PROJECTIVE SHADOWS

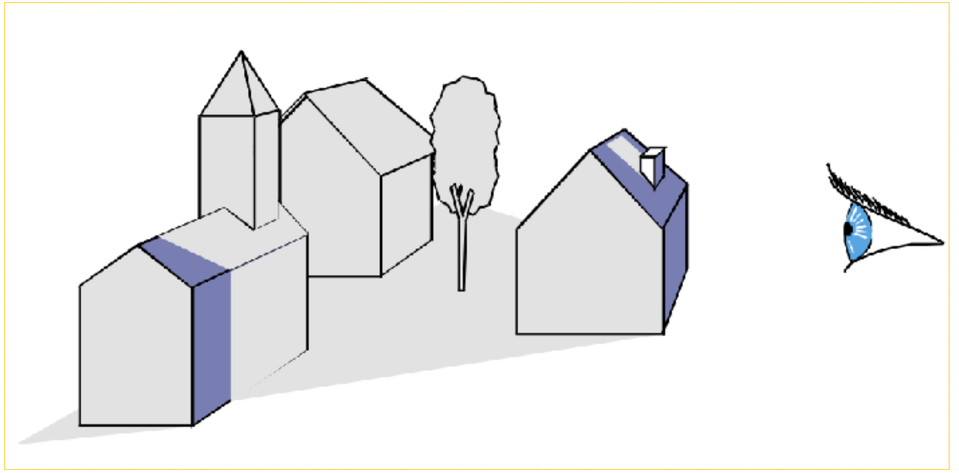
- Easy to compute
- Simply project geometry on ground plane
- Only works on flat shadow receivers, e.g. car on street, player on ground
- No self-shadows
- Expensive: needs to be rendered every frame even for static scenes



SHADOW/VIEW DUALITY

- A point is in shadow if it not visible from the perspective of the light source
- Use the view from the light source to compute shadows





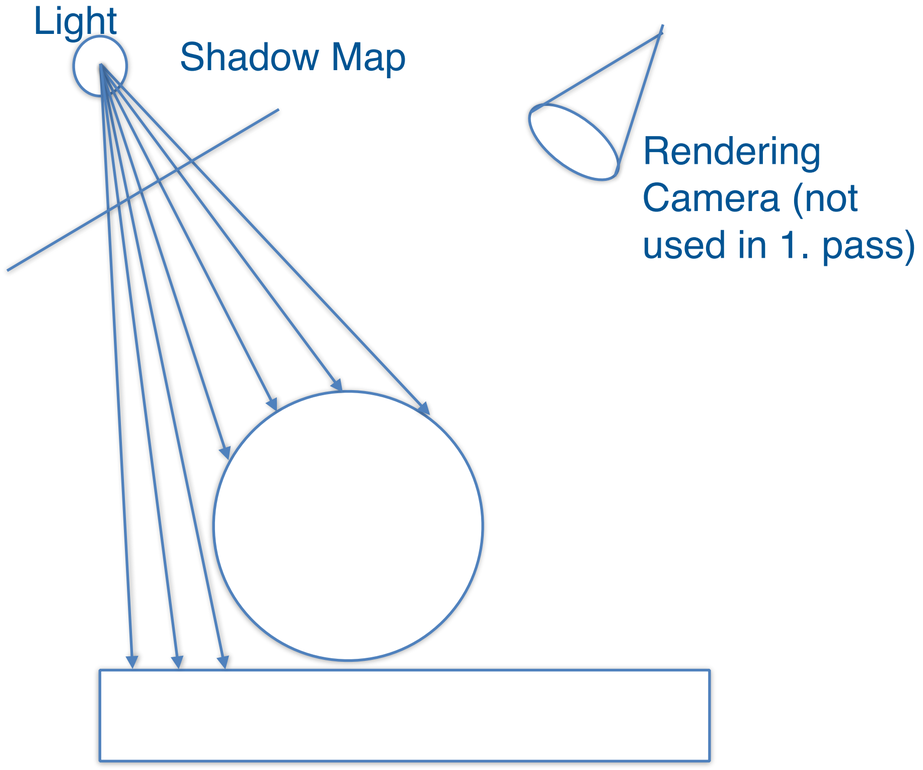
SHADOW MAPPING

- Use shadow/view duality
- Two rendering passes
  - 1st Pass: Rendering shadow map representing the depth from light source
  - 2nd Pass: Render final image from camera view and check shadow map to see if points are in shadow

FIRST PASS: RENDER SHADOW MAP

Create a map of depth values as seen from the light's point of view

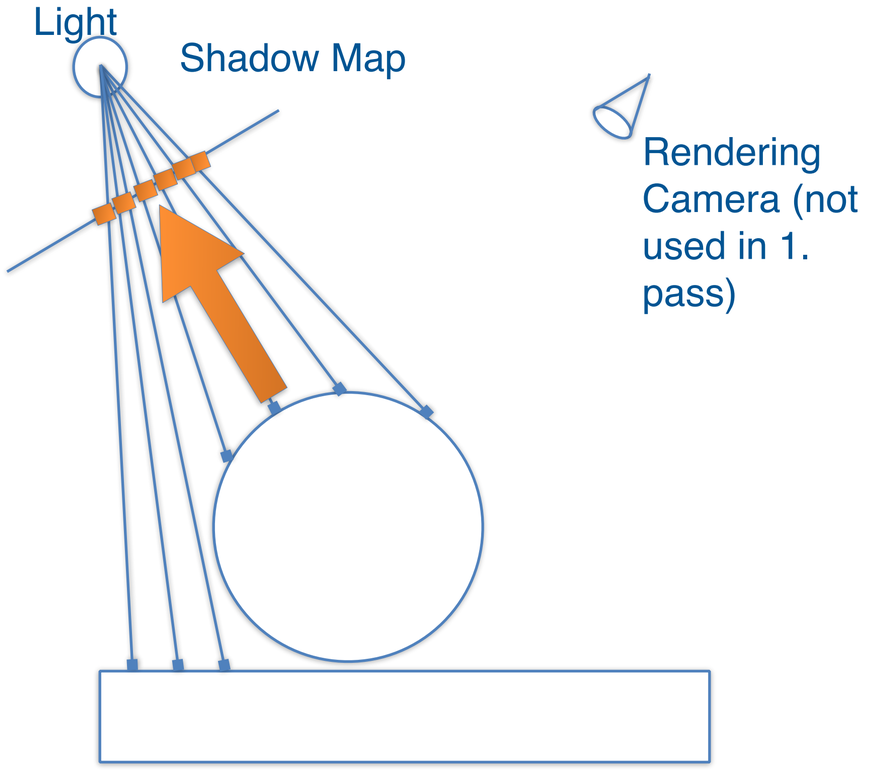
- Geometry is rendered into a depth buffer from the point of view of the light
- Transform the geometry into light-view space



FIRST PASS: RENDER SHADOW MAP

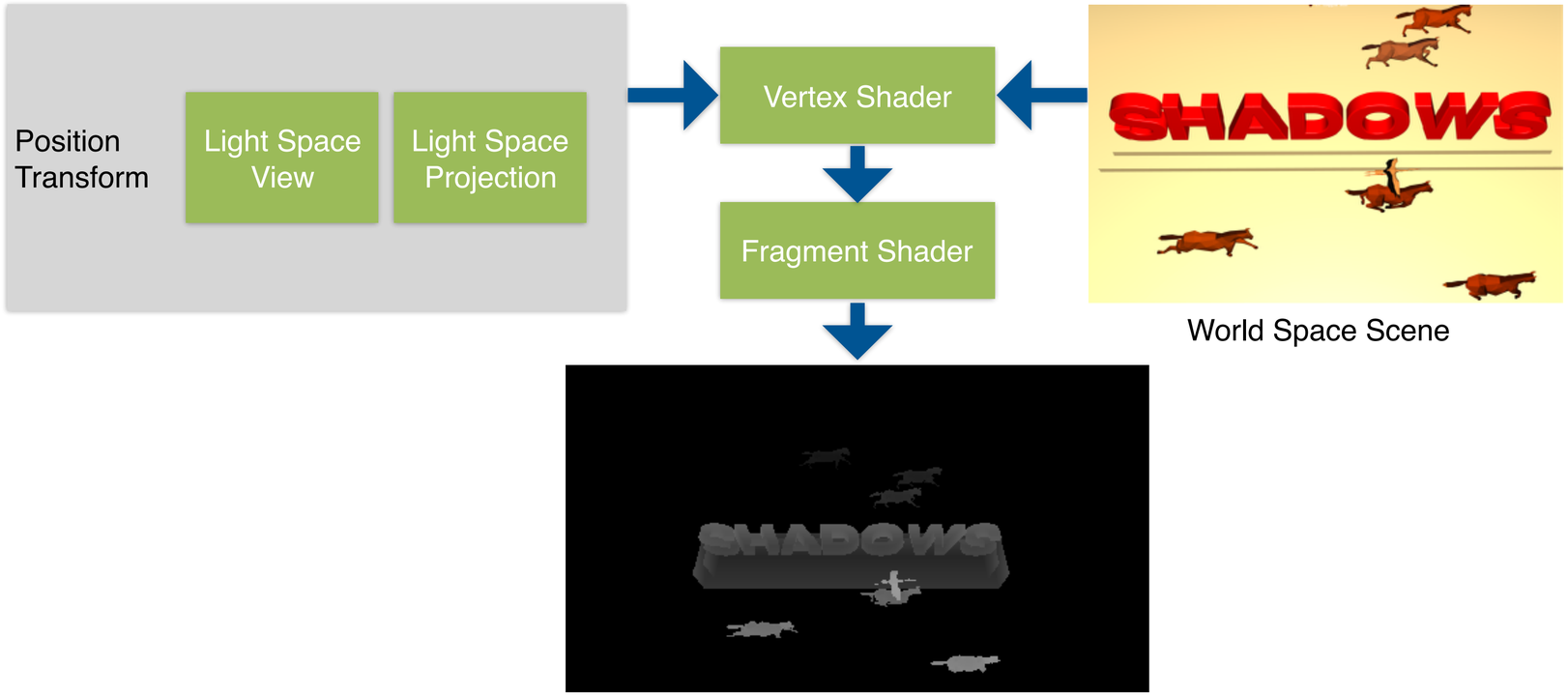
Create a map of depth values as seen from the light's point of view

- Geometry is rendered into a depth buffer from the point of view of the light
- Transform the geometry into light-view space



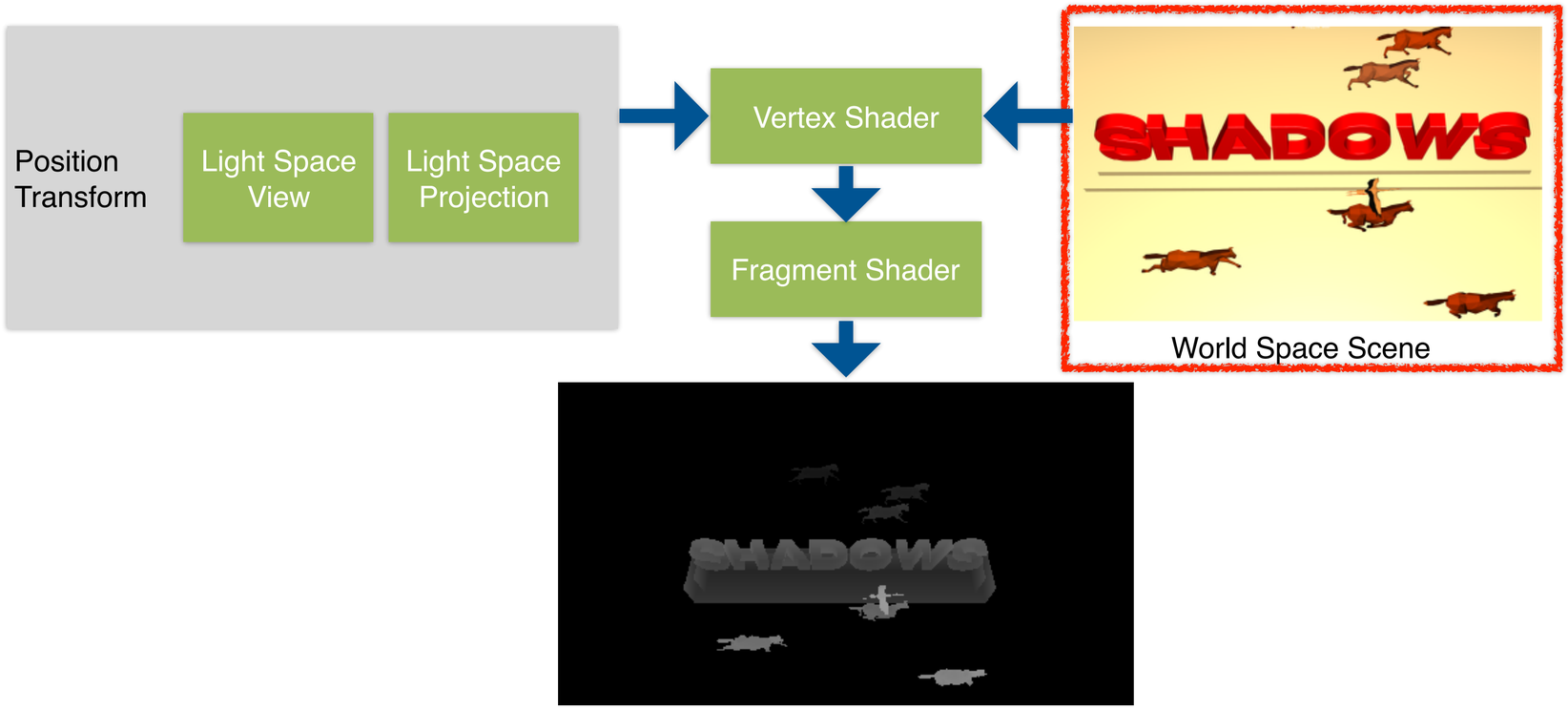
FIRST PASS: RENDER SHADOW MAP

In practice: Vertex shader transforms the geometry into light-view space



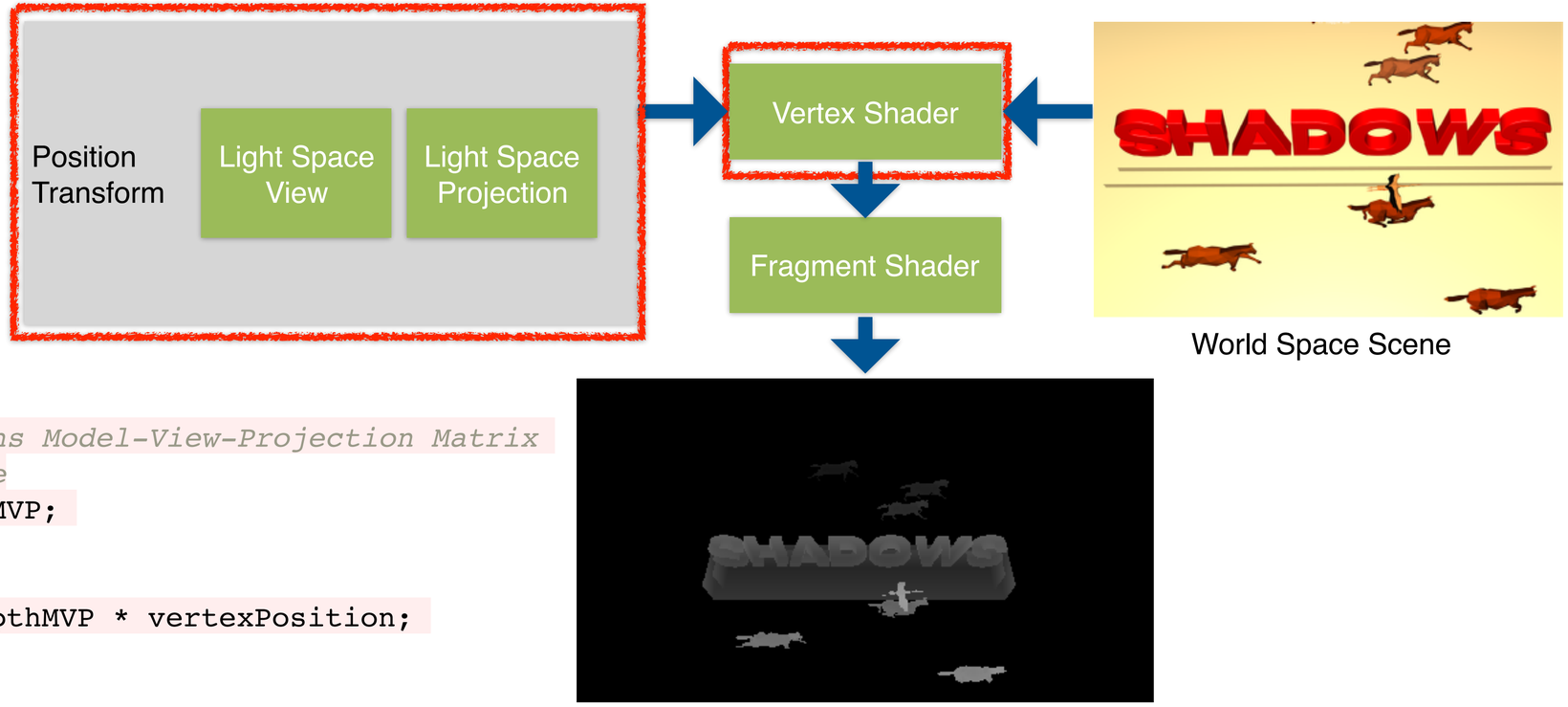
FIRST PASS: RENDER SHADOW MAP

In practice: Vertex shader transforms the geometry into light-view space



FIRST PASS: RENDER SHADOW MAP

In practice: Vertex shader transforms the geometry into light-view space

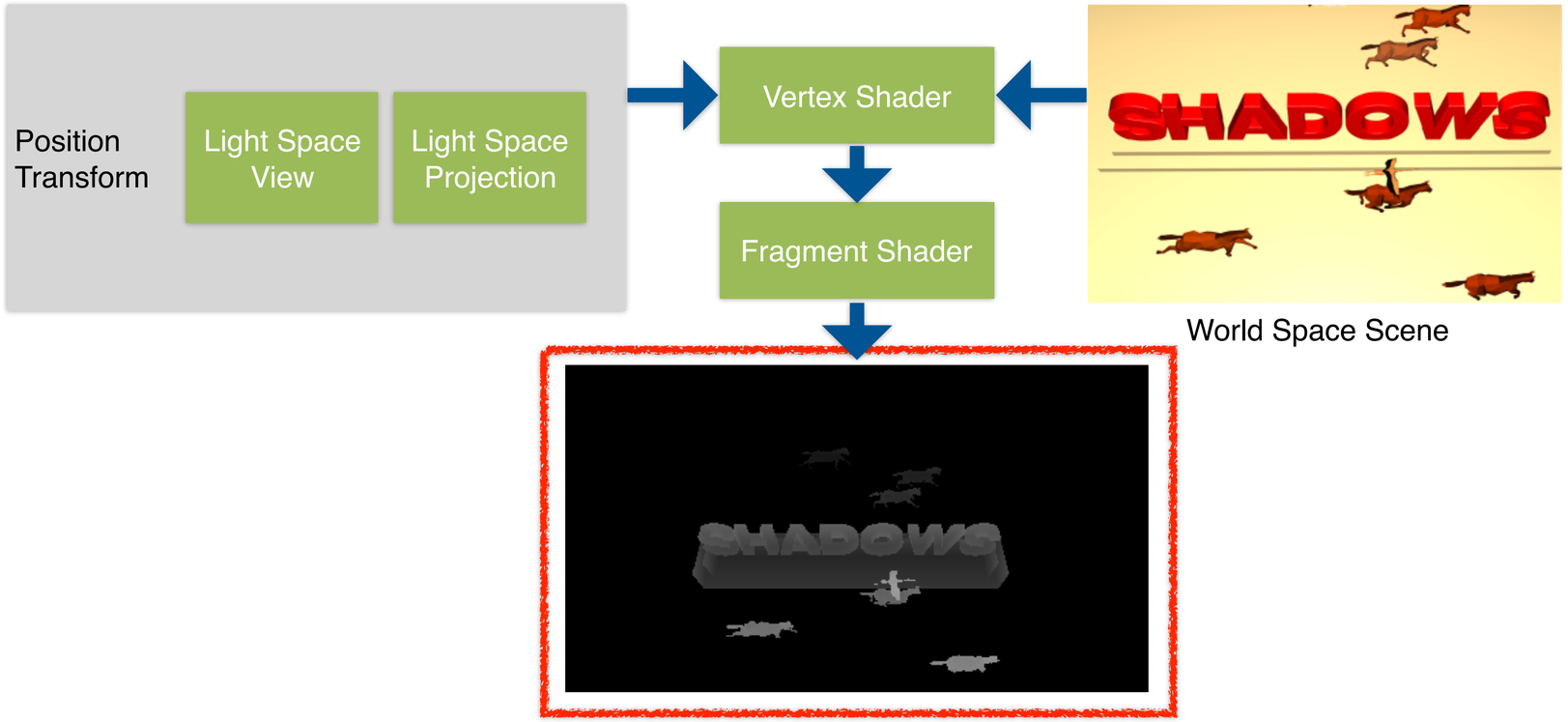



```
// depthMVP contains Model-View-Projection Matrix
// for light source
uniform mat4 depthMVP;

void main(){
 gl_Position =  depthMVP * vertexPosition;
}
```

FIRST PASS: RENDER SHADOW MAP

In practice: Vertex shader transforms the geometry into light-view space

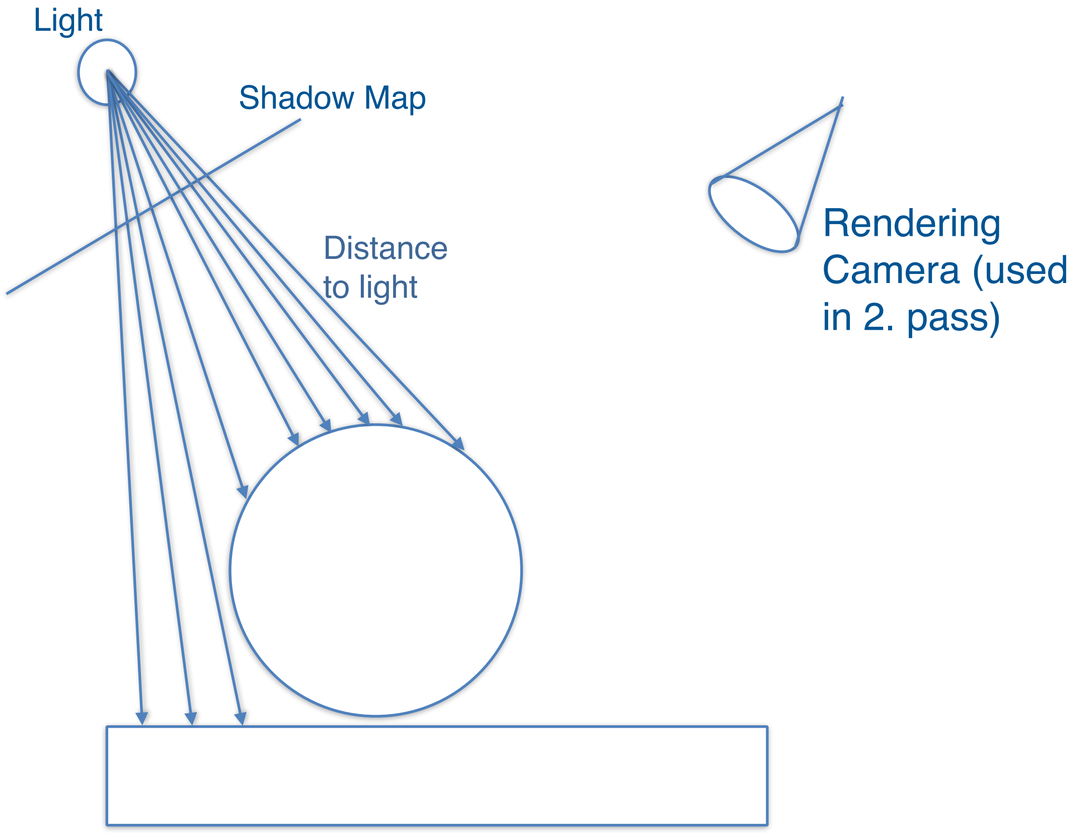


FIRST PASS: RENDER SHADOW MAP

- Render to texture: shadowMap
- Use shadowMap as input for second rendering pass

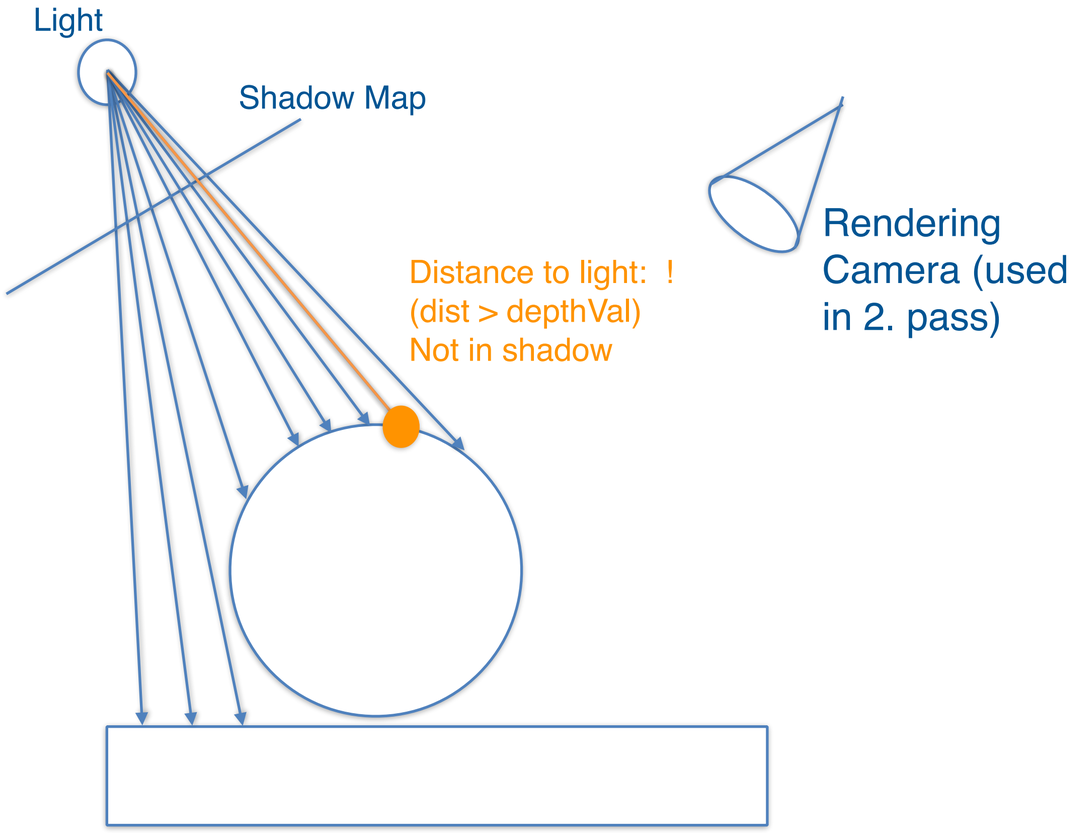
SECOND PASS: RENDER SCENE

- Render final output from camera view and check shadow map to see if points are in shadow
- Compare distance between point and light source and value in shadow map
- If distance larger than value in shadow map point is in shadow



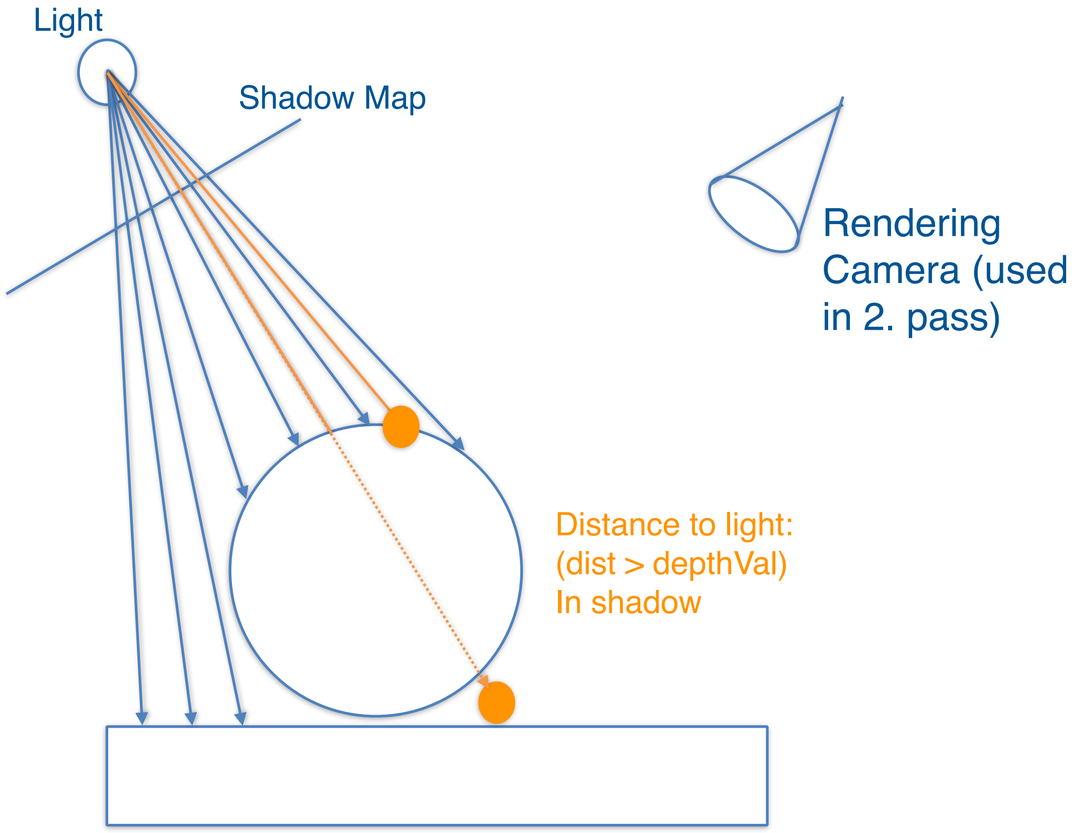
SECOND PASS: RENDER SCENE

- Render final output from camera view and check shadow map to see if points are in shadow
- Compare distance between point and light source and value in shadow map
- If distance larger than value in shadow map point is in shadow



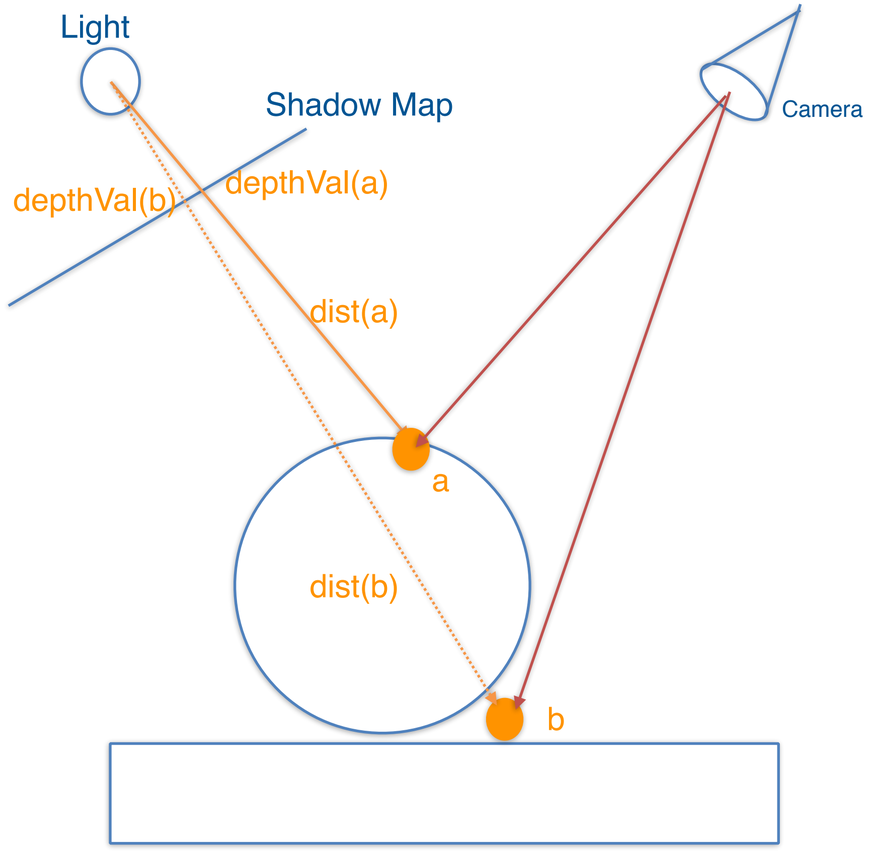
SECOND PASS: RENDER SCENE

- Render final output from camera view and check shadow map to see if points are in shadow
- Compare distance between point and light source and value in shadow map
- If distance larger than value in shadow map point is in shadow



SECOND PASS: RENDER SCENE

- Transform points in light space
- Transform points in shadow map
- Look up value from shadow map (depthVal(p))
- Compute visibility based on:
  - dist(p) > depthVal(p)
- **In practice**: use vertex shader and fragment shader



SECOND PASS: RENDER SCENE

- Transform points in light space
- Transform points in shadow map
- Look up value from shadow map (depthVal(p))
- Compute visibility based on:
  - dist(p) > depthVal(p)
- **In practice**: use vertex shader and fragment shader

Vertexshader:

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

// Same, but with the light's view matrix
shadowCoord = textureSpace* DepthMVP * vec4(vertPos_modelspace,1);
```

Fragmentshader:

```
float visibility = 1.0;
float depthVal = texture( shadowMap, shadowCoord.xy ).z;
float dist = shadowCoord.z;
if ( dist > depthVal){
    visibility = 0.0;
}
```

SECOND PASS: RENDER SCENE

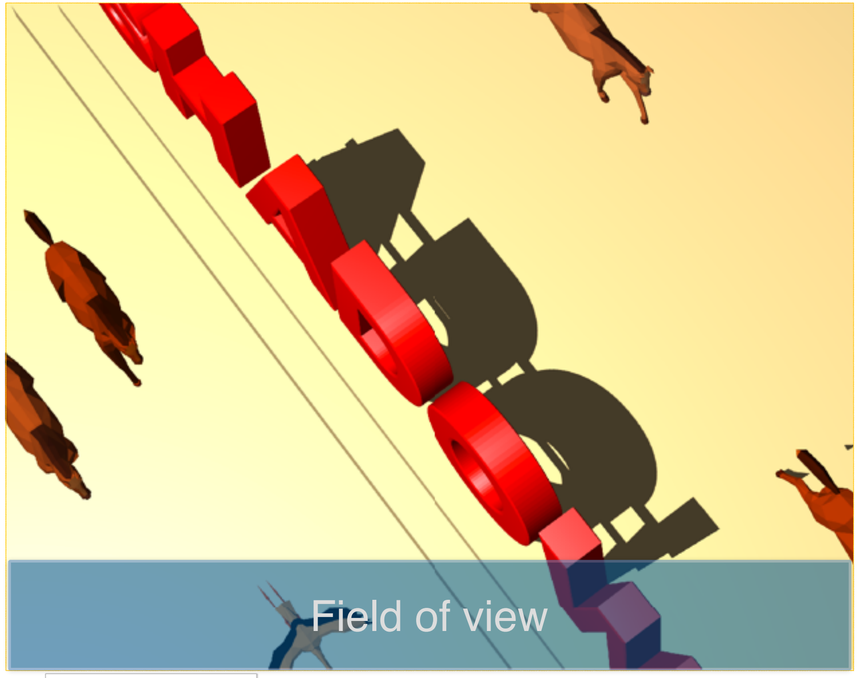
Final Shading Step

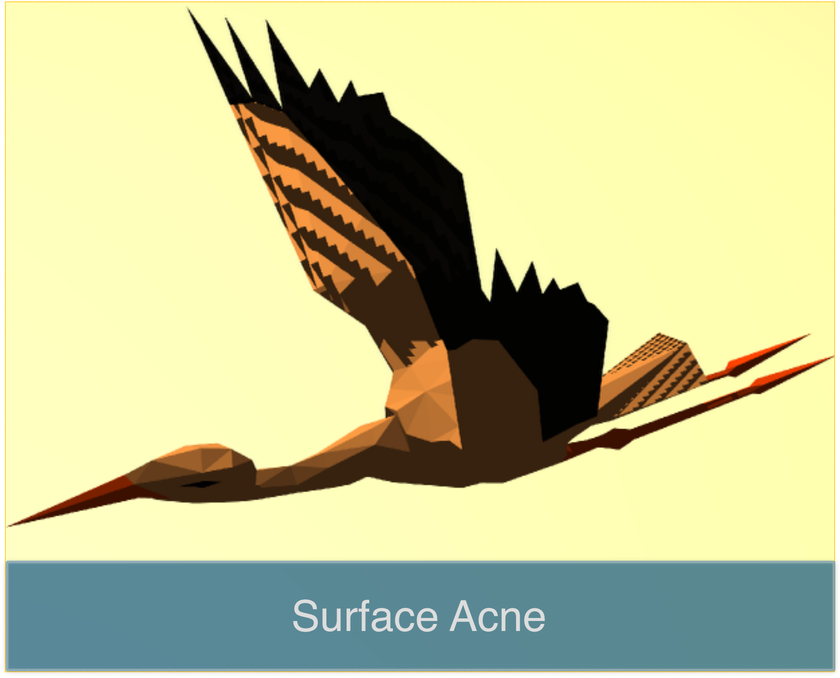
- Use calculated visibility to compute fragment's output colour
- **In practice:** use visibility in fragment shader:

```
outputColor =
 // Ambient : simulates indirect lighting
 ambientColor +
 // Diffuse : "color" of the object
 visibility * diffuseColor
 // Specular : reflective highlight, like a mirror
 visibility * specularColor;
```

LIMITATIONS

- Field of View
- Surface Acne
- Aliasing

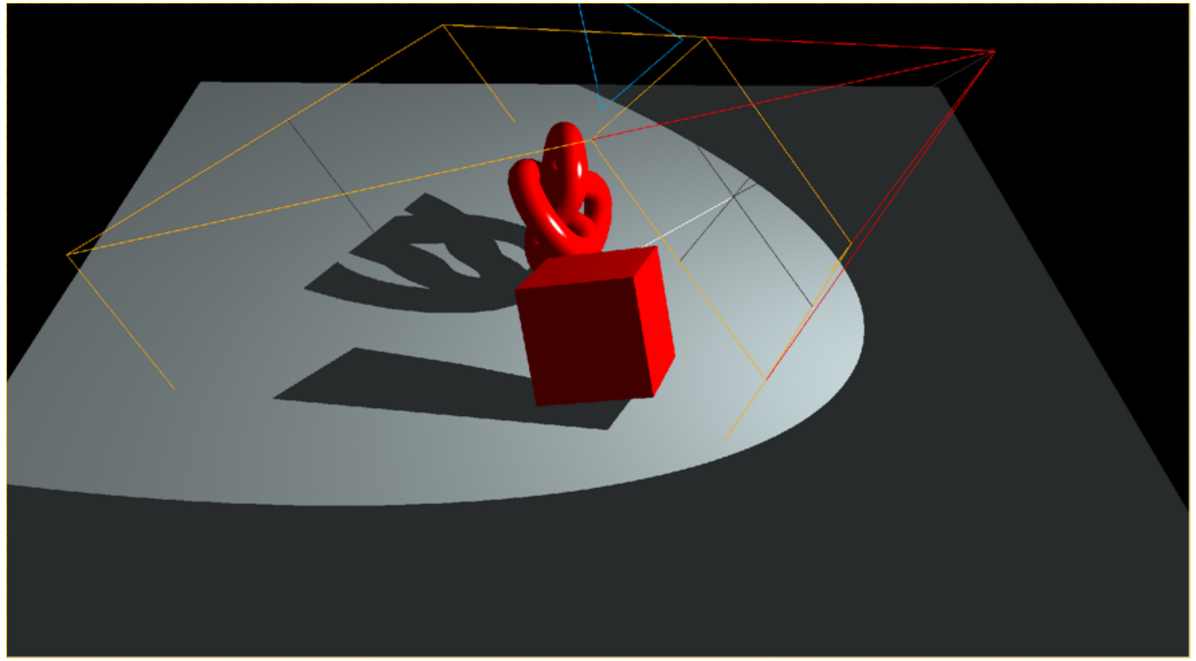

Field of view


Surface Acne


Shadow Map Aliasing

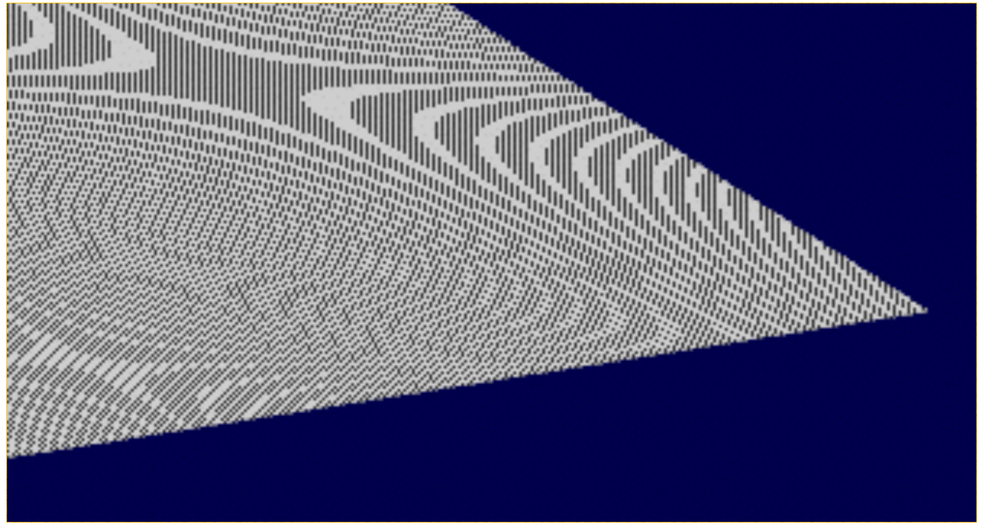
FIELD OF VIEW PROBLEM

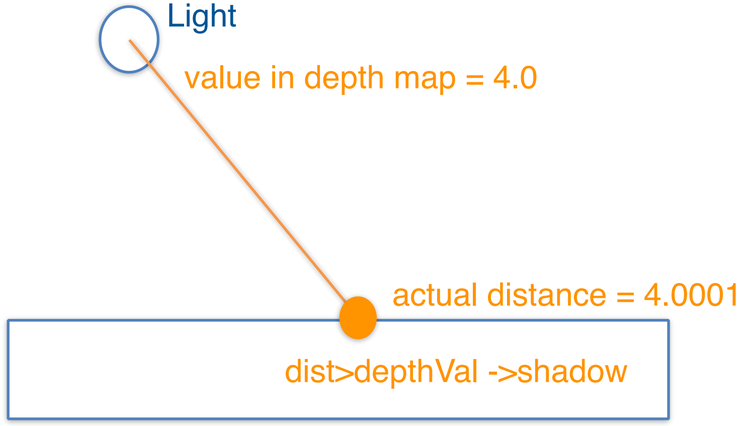
- What if object is outside field of view of shadow map?
- No shadows or partial shadows
- For spot lights, this can be changed by tweaking its range
- Problem in particular for larger scenes



SURFACE ACNE

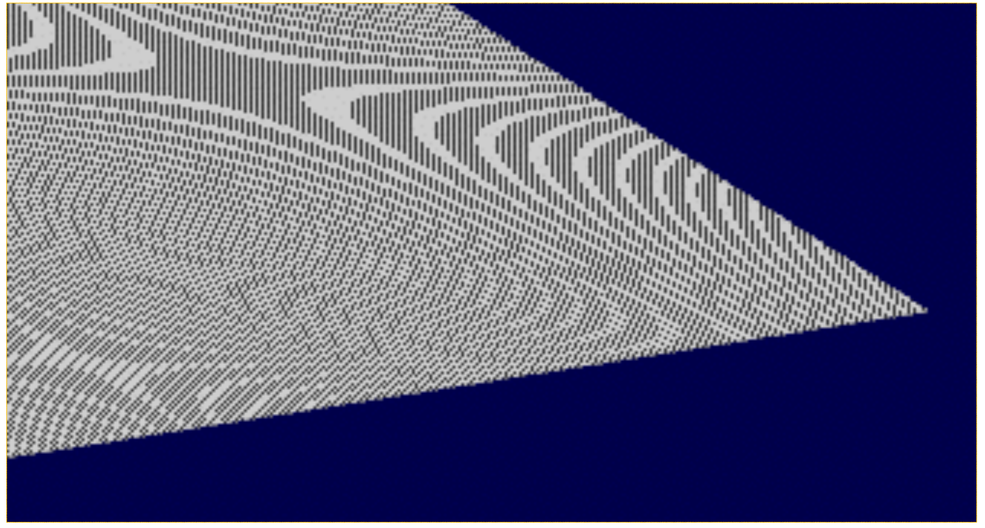
- Self-shadowing problem due to precision and depth map resolution
  - Depth value in map can differs from actual distance between object and light source
  - Sampling problem: neighbouring vertices map to the same depth map pixel

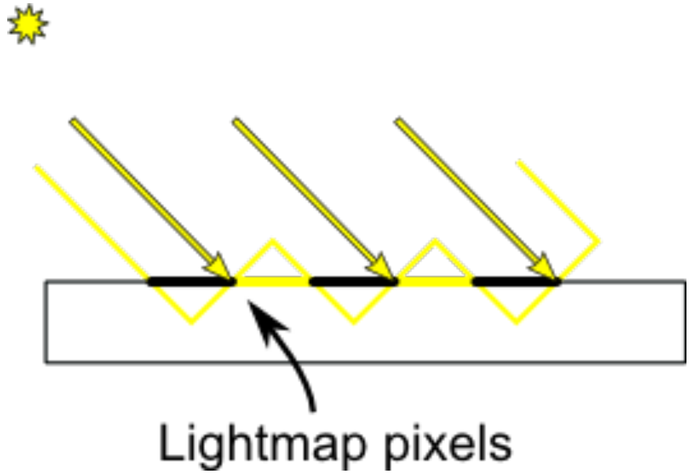




SURFACE ACNE

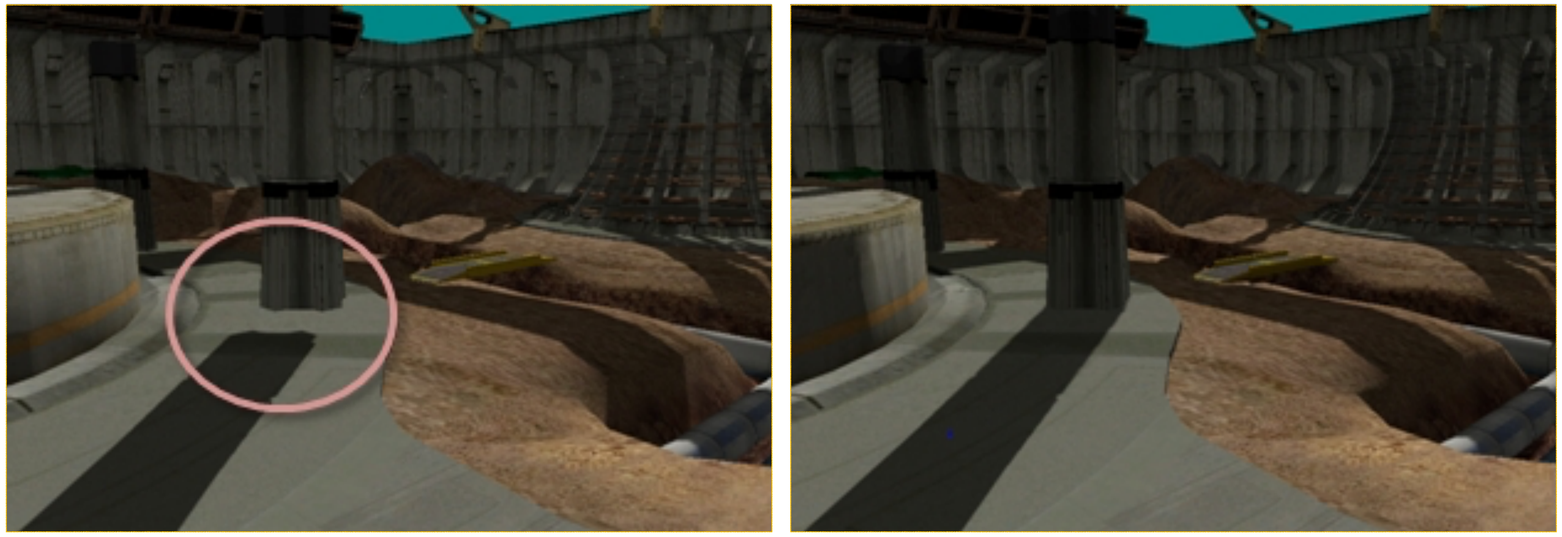
- Self-shadowing problem due to precision and depth map resolution
  - Depth value in map can differs from actual distance between object and light source
  - Sampling problem: neighbouring vertices map to the same depth map pixel





SHADOW MAP BIAS

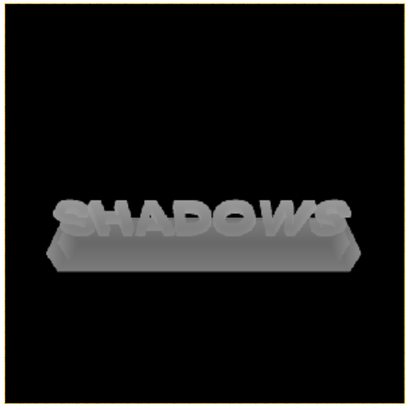
- Solution for surface acne:
  - Shadow map bias for shadow test: ShadowMap(x,y) + bias < dist
  - Choosing a good bias value can be tricky - otherwise Peter Panning



SHADOW MAP ALIASING

- Quality depends on shadow map resolution
- Higher resolution:
  - Higher quality
  - More memory required

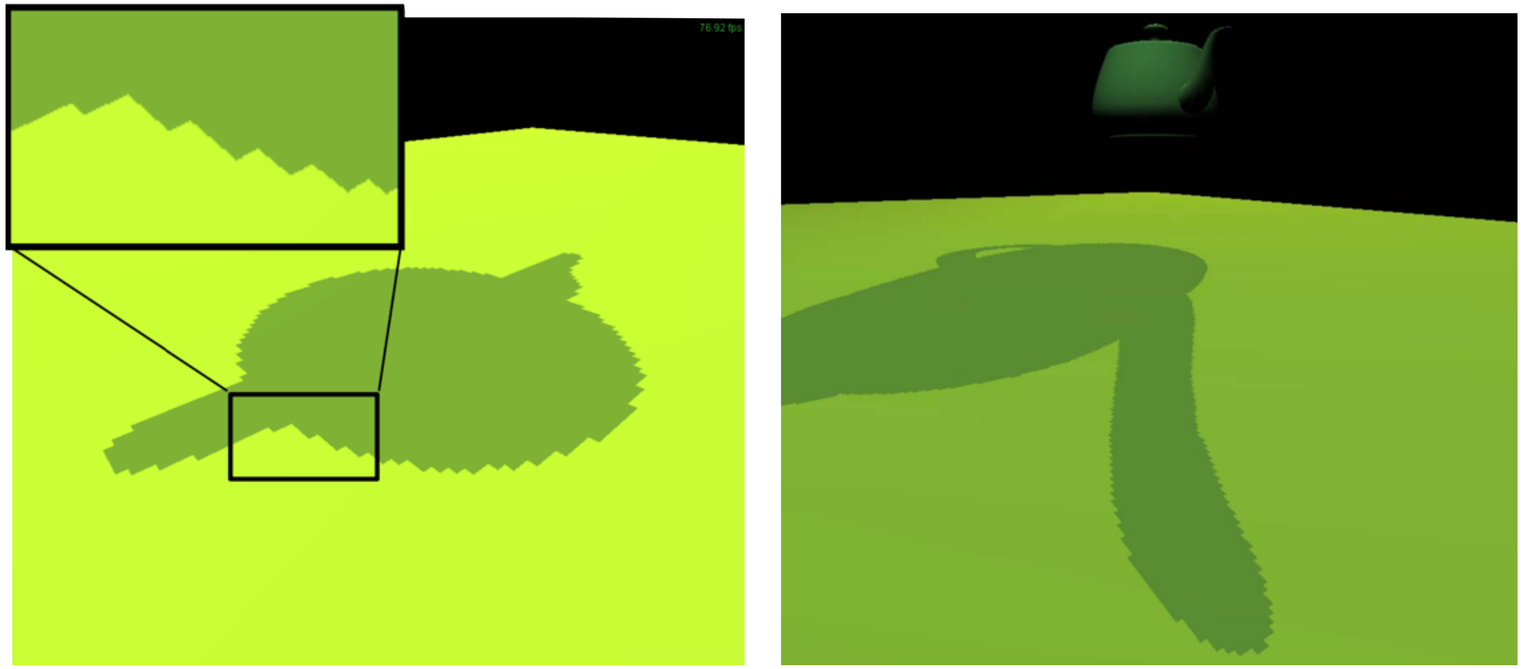
1024x1024



2048x2048

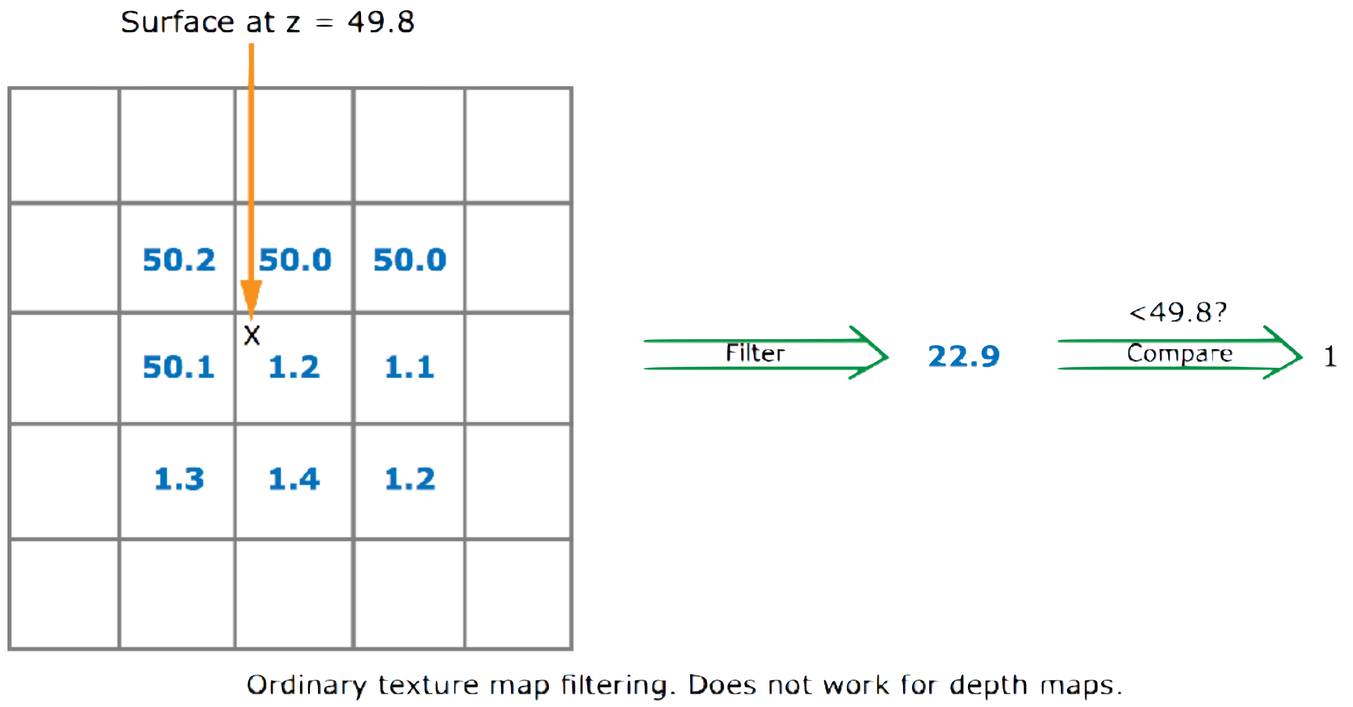
SHADOW MAP ALIASING

- Under-sampling of the shadow map – Jagged shadow edges
- Solution: Percentage closer filtering (PCF)



PERCENTAGE CLOSER FILTERING (PCF)

- Creating softer shadows
- Unlike normal textures, shadow map textures cannot be pre-filtered to remove aliasing
- Weighted average of neighbouring depth values is not meaningful



Ordinary texture map filtering. Does not work for depth maps.

Image by MIT OpenCourseWare.

PERCENTAGE CLOSER FILTERING (PCF)

- Creating softer shadows
- Unlike normal textures, shadow map textures cannot be pre-filtered to remove aliasing
- Weighted average of neighbouring depth values is not meaningful

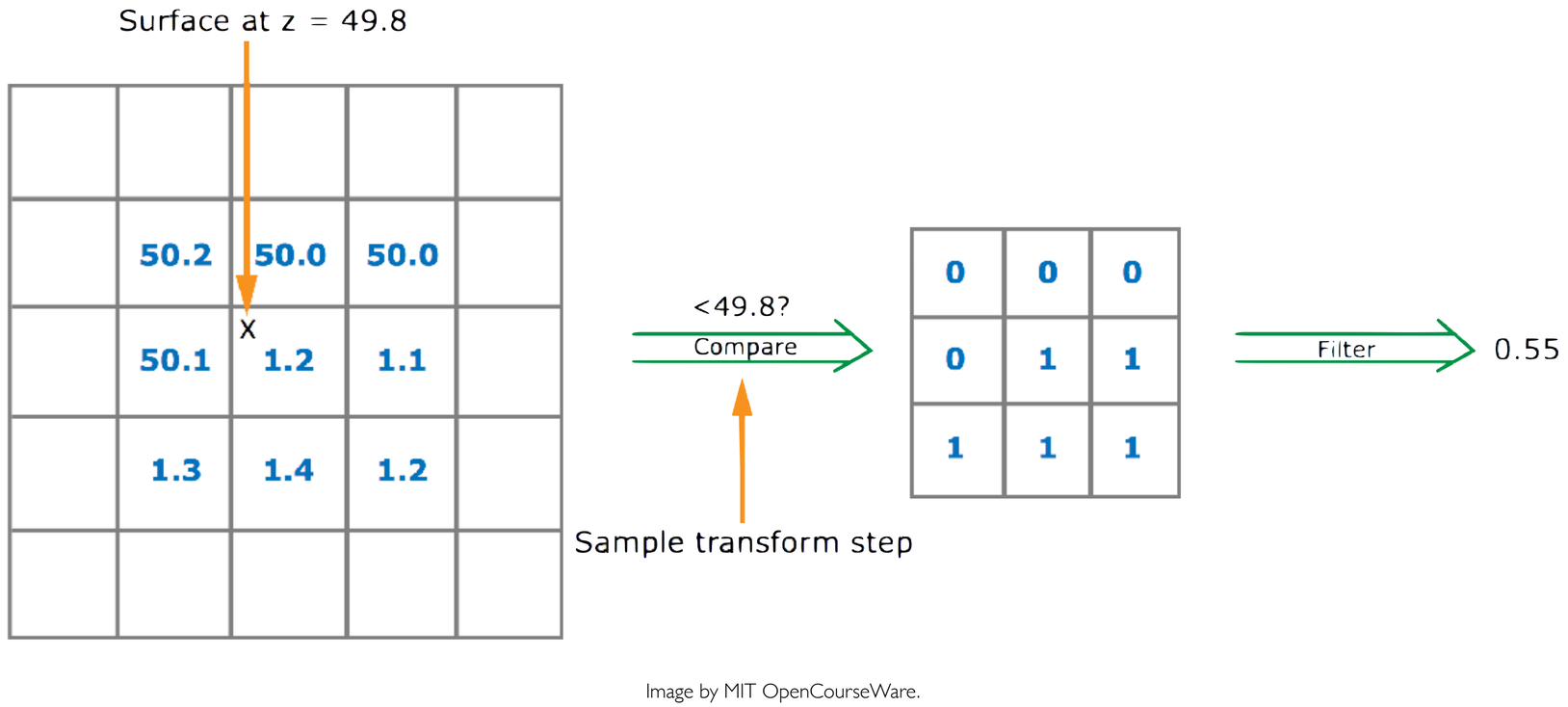


Image by MIT OpenCourseWare.

PERCENTAGE CLOSER FILTERING (PCF)

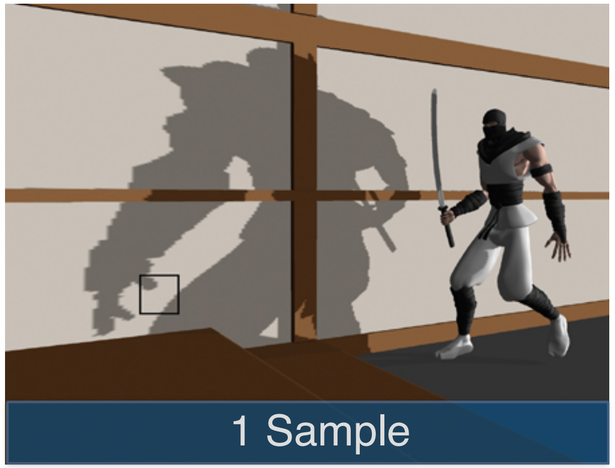
- Quality increases with Percentage closer filtering
- Softer appearance of shadows

1024x1024

1024x1024

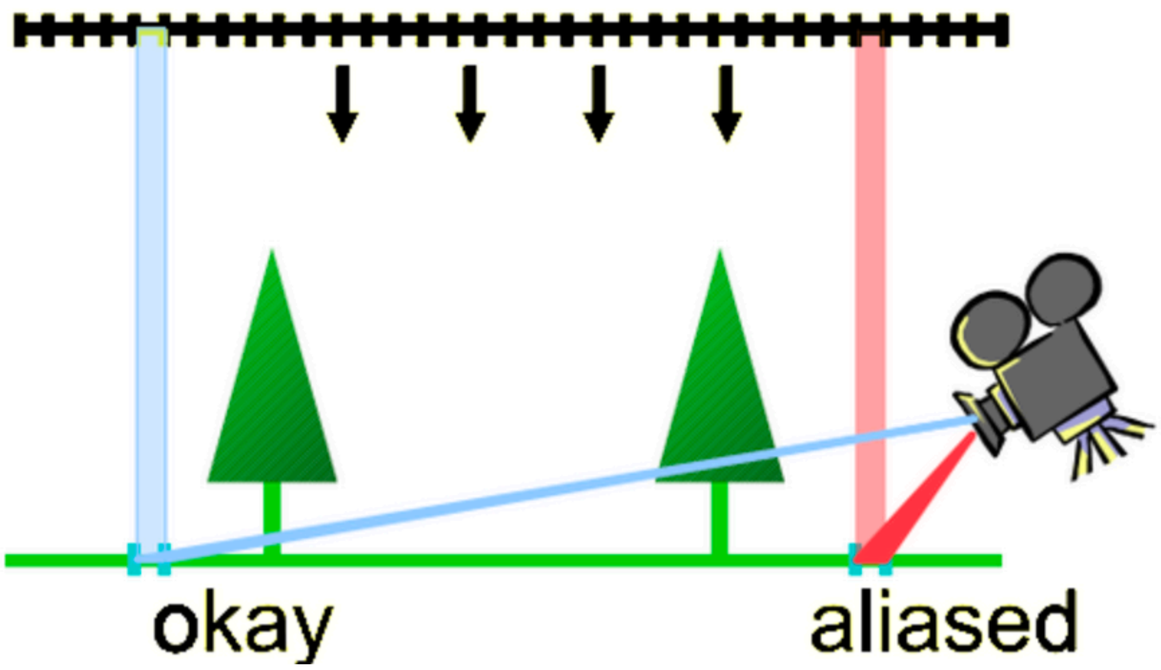
PERCENTAGE CLOSER FILTERING (PCF)

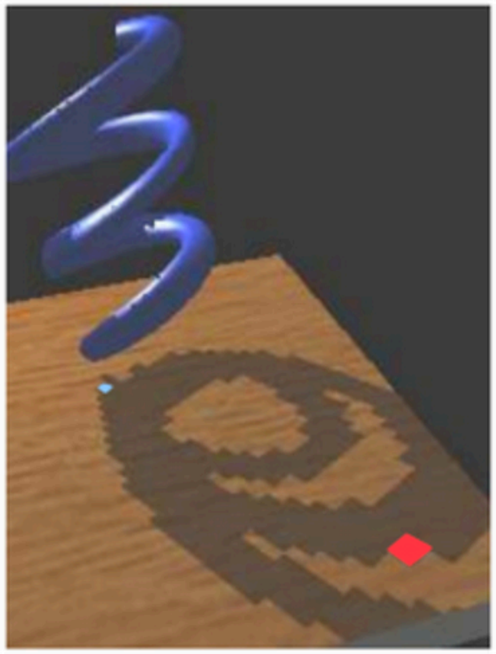
- Results for different filter size



PERSPECTIVE ALIASING

- View space resolution vs. shadow map resolution



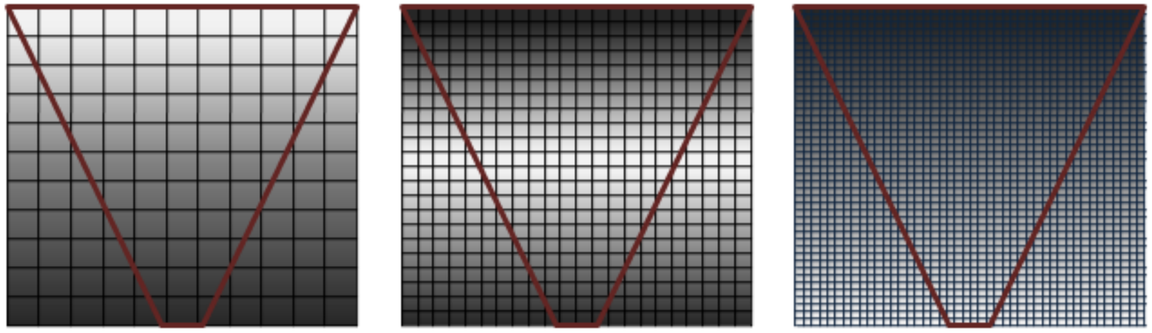


http://www.cg.tuwien.ac.at/~scherzer/files/papers/LispSM_survey.pdf

CASCADE SHADOW MAPS

- Providing higher resolution of the depth texture near the viewer and lower resolution far away

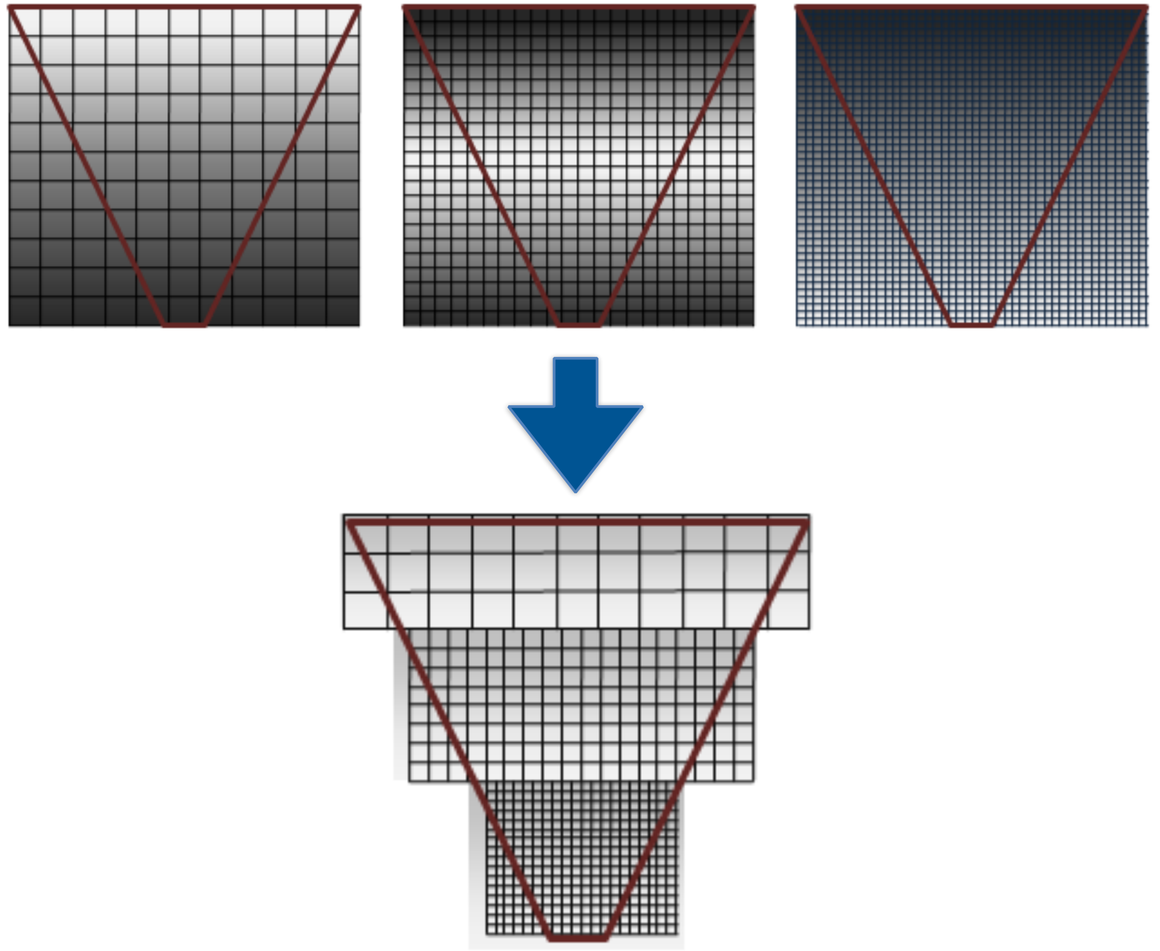
http://msdn.microsoft.com/en-us/library/windows/desktop/ee416307(v=vs.85).aspx



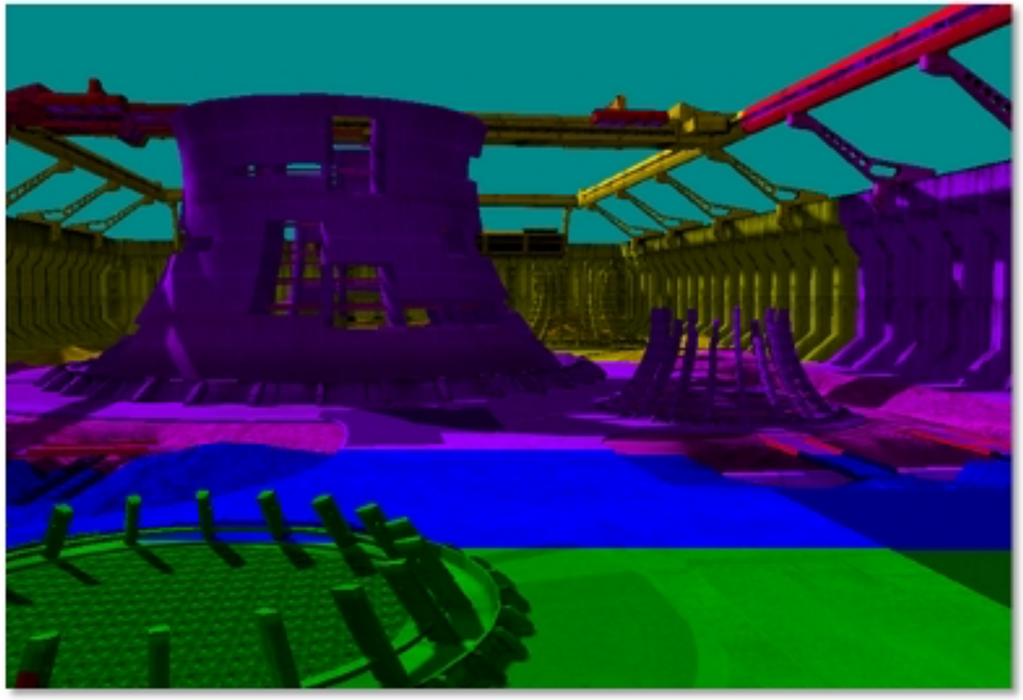
CASCADE SHADOW MAPS

- Providing higher resolution of the depth texture near the viewer and lower resolution far away
- Splitting the camera view frustum and creating a separate depth-map for each partition

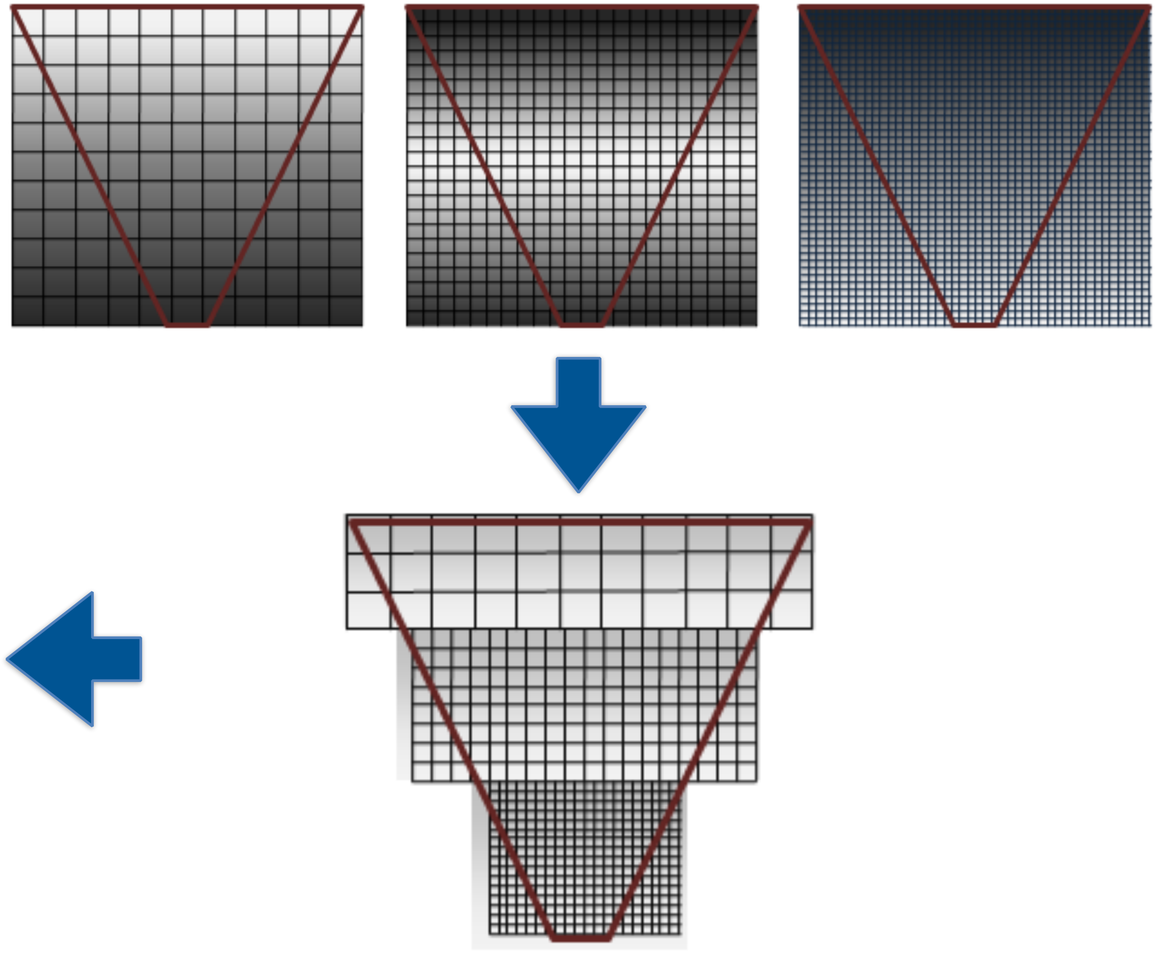
http://msdn.microsoft.com/en-us/library/windows/desktop/ee416307(v=vs.85).aspx



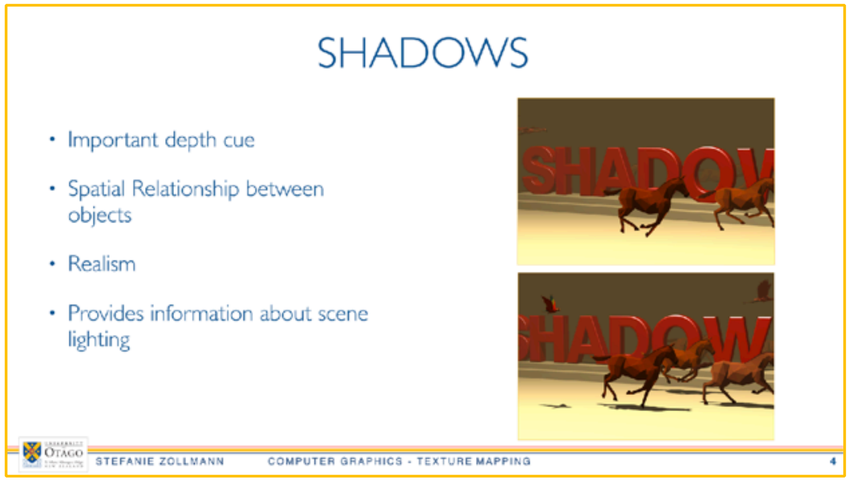
CASCADE SHADOW MAPS

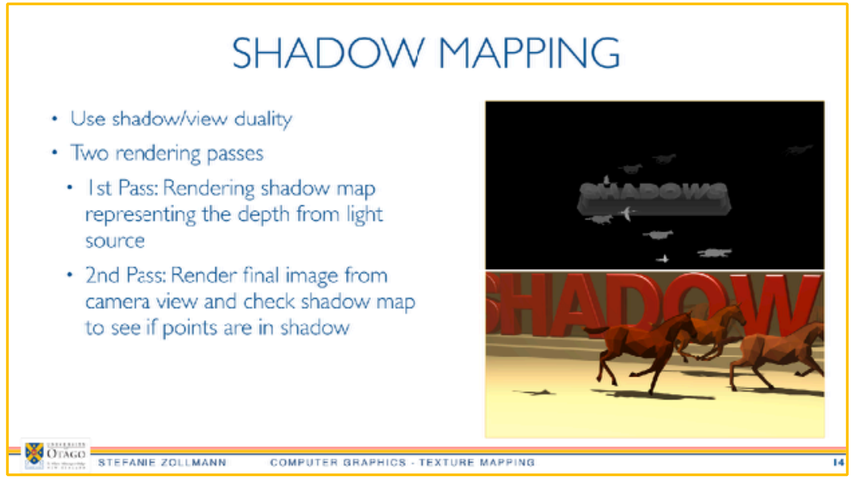


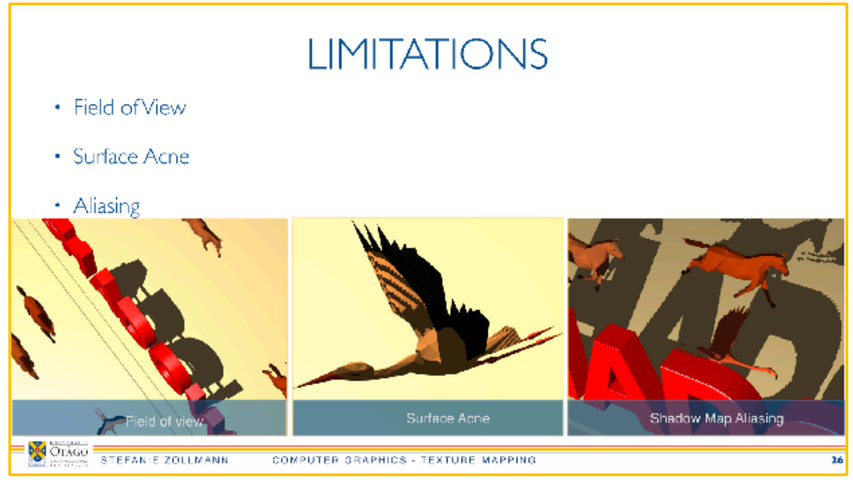
http://msdn.microsoft.com/en-us/library/windows/desktop/ee416307(v=vs.85).aspx



SUMMARY





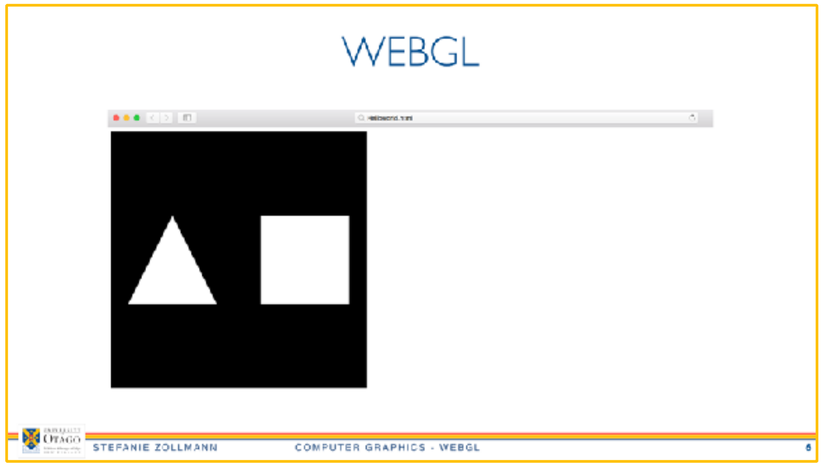


Shadows

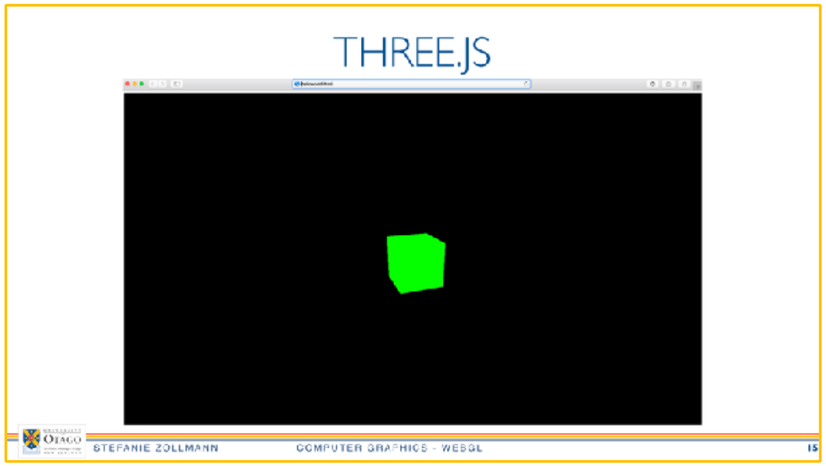
Methods

Limitations & Improvements

WHAT'S NEXT



WEBGL



THREE.JS

INTERACTIVE DEMOS

Thank You!

For more material visit

http://www.cs.otago.ac.nz/cosc342/