

DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Shadows

Jiří Bittner, Michael Wimmer

Computer Graphics Research – Info Sources

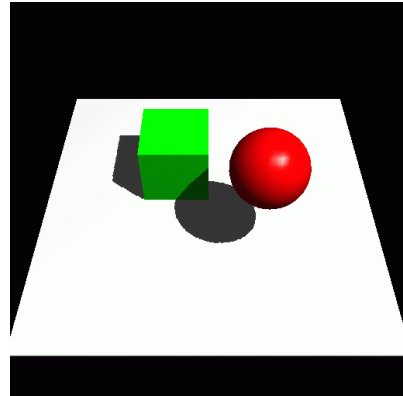
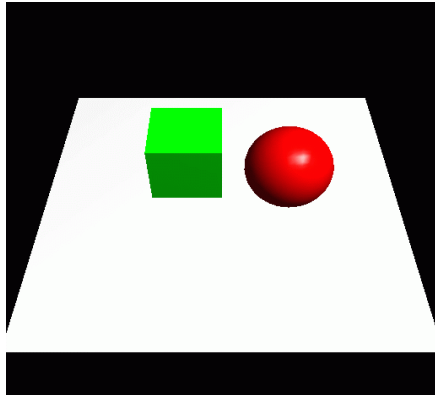
- <http://kesen.realtimerendering.com/>
- SIGGRAPH, SIGGRAPH Asia, Eurographics, EGSR, I3D, ...
- scholar.google.com

Outline

- Motivation & Terminology MPG 12
- Approximate & projection shadows MPG 12.1
- Shadow maps MPG 12.3
- Shadow volumes MPG 12.2
- Summary

What for?

Shadows tell us about the relative locations and motion of objects
And about light positions



What for?

Objects look like they are “floating” → shadows fix that!



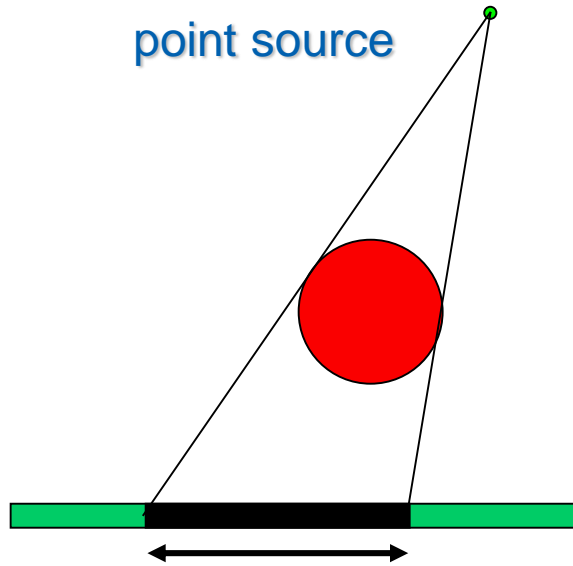
Motivation

- Shadows contribute significantly to realism of rendered images
 - Anchor objects in scene
- **Global** effect → expensive!
- Light source behaves very similar to camera
 - Is a point visible from the light source?
 - shadows are “hidden” regions
 - Shadow is a projection of caster on receiver
 - projection methods

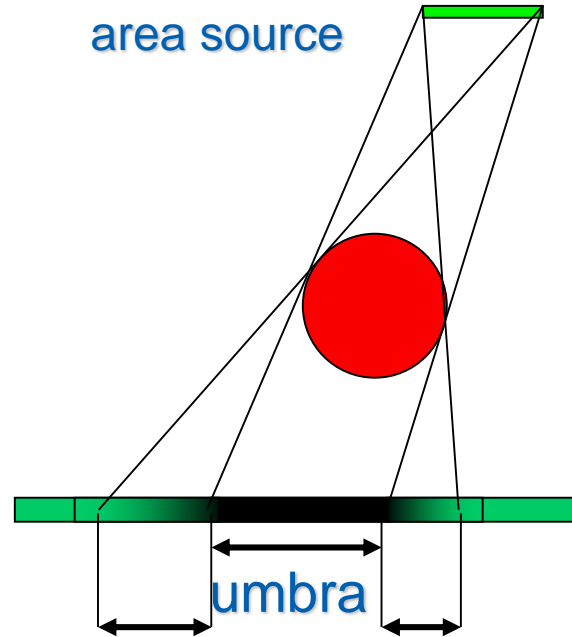
Shadow Algorithms

- Static shadow algorithms (lights + objects)
 - Radiosity, ray tracing → lightmaps
- Approximate shadows
- Projected shadows [Blinn 88]
- Shadow maps [Williams 78]
 - Projective image-space algorithm
- Shadow volumes [Crow 77]
 - Object-space algorithm
- Soft shadow extensions for all above algorithms
 - Still hot research topic (500+ shadow publications)

Hard vs. Soft Shadows



- hard shadow**
- +fast
 - only good for localized lights (sun, projectors)
 - +fake soft shadow through filtering



- umbra**
- penumbra penumbra**
- + very realistic
 - very expensive

Static Shadows

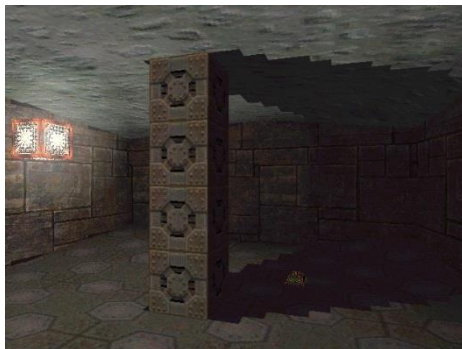
- Glue to surface whatever we want
- Idea: incorporate shadows into light maps
 - For each texel, cast ray to each light source
- “Bake” soft shadows in light maps
 - Not by texture filtering alone, but:
 - Sample area light sources

Static Soft Shadow Example

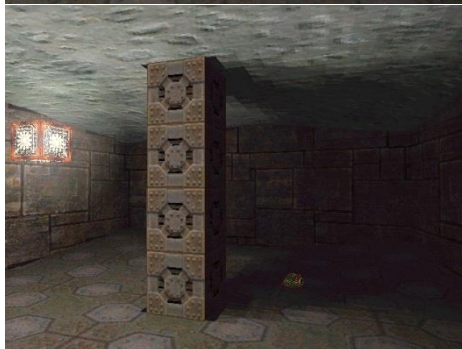
no filtering

filtering

1 sample



n samples



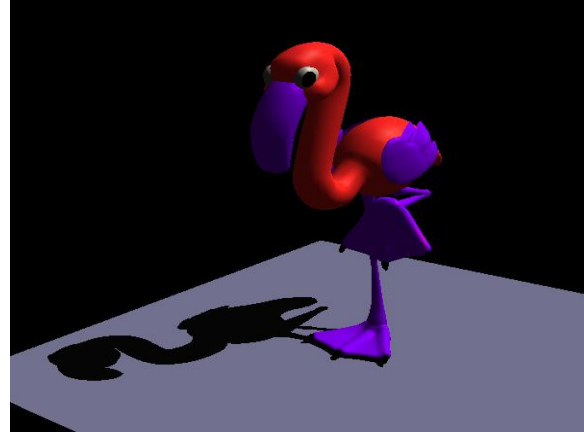
Approximate Shadows

- Handdrawn approximate geometry
 - Perceptual studies suggest: shape not so important
 - Minimal cost



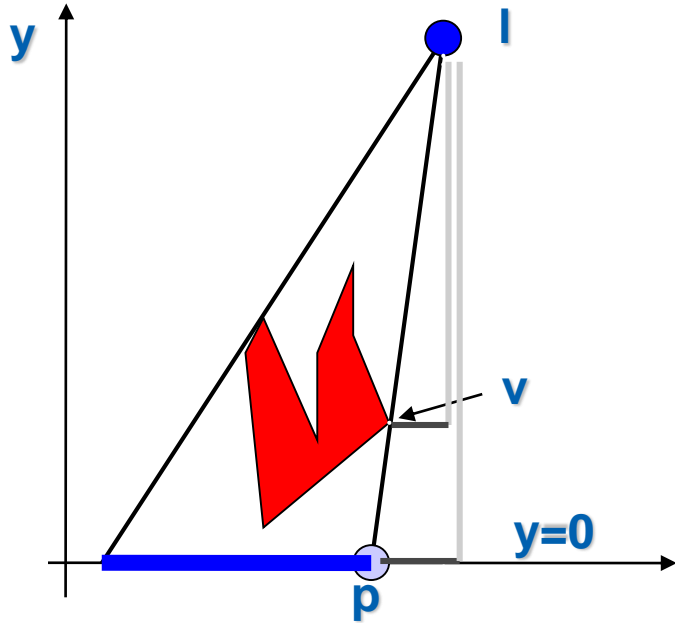
Projection Shadows (Blinn 88)

- Shadows for selected large *planar* receivers
 - Ground plane
 - Walls
- Projective geometry: flatten 3D model onto plane
 - and “darken” using framebuffer blend



Projection for Ground Plane

- Use similar-triangles



$$\frac{p_x - l_x}{v_x - l_x} = \frac{l_y}{l_y - v_y}$$

$$p_x = \frac{l_y v_x - l_x v_y}{l_y - v_y}$$

$$p_z = \frac{l_y v_z - l_z v_y}{l_y - v_y}$$

$$p_y = 0$$

Projection Matrix

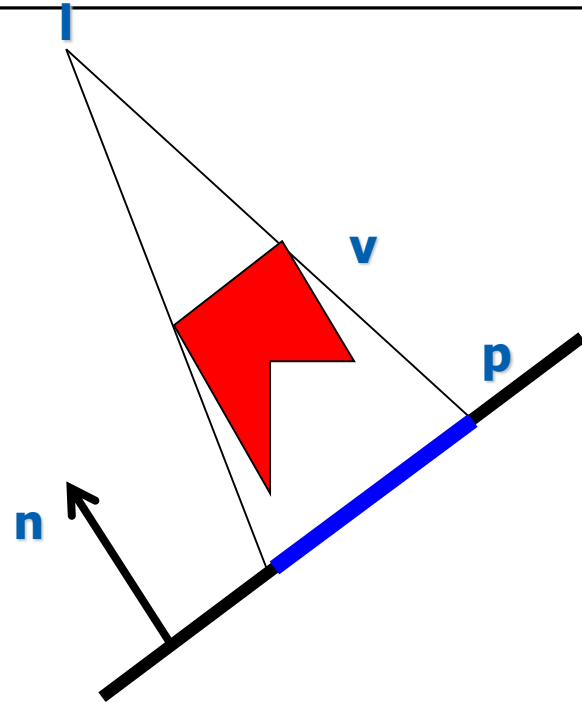
- Projective 4x4 matrix:

$$M = \begin{pmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{pmatrix}$$

- Arbitrary plane:

- Intersect line $\mathbf{p} = \mathbf{l} - \alpha (\mathbf{v} - \mathbf{l})$
- with plane $\mathbf{n} \cdot \mathbf{x} + d = 0$
- Express result as a 4x4 matrix

- Append this matrix to view transform

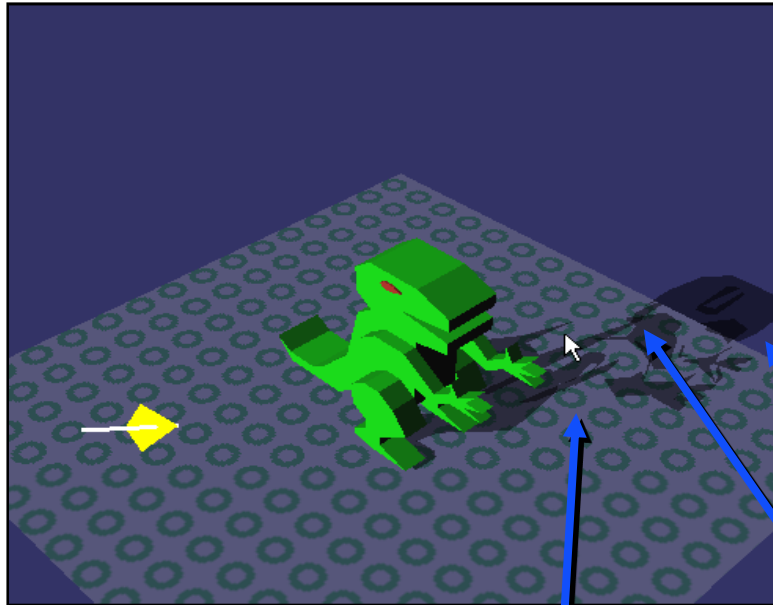


Projection Shadow Algorithm

- Render scene (full lighting)
- For each receiver polygon
 - Compute projection matrix M
 - Append to view matrix
 - Render selected shadow caster
 - With framebuffer blending enabled

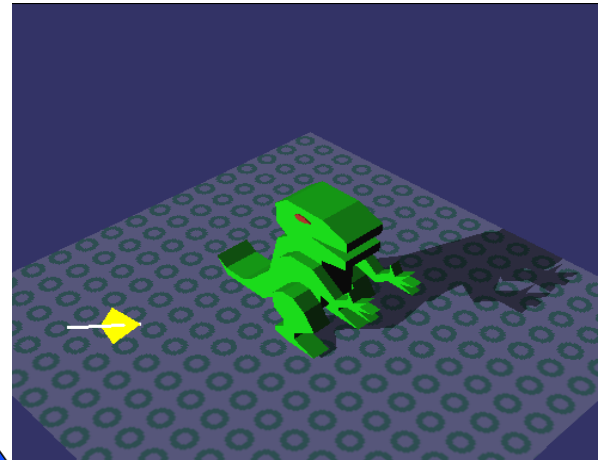
Projection Shadow Artifacts

Bad



Z fighting

Good



extends off
ground region

double blending

Stencil Buffer Projection Shadows

- Stencil can solve all of these problems
 - Separate 8-bit frame buffer for numeric ops
- Stencil buffer algorithm (requires 1 bit):
 - Clear stencil to 0
 - Draw ground polygon last and with
 - `glStencilOp(GL_KEEP, GL_KEEP, GL_ONE);`
 - Draw shadow caster with no depth test but
 - `glStencilFunc(GL_EQUAL, 1, 0xFF); glStencilOp(GL_KEEP, GL_KEEP, GL_ZERO);`
- Every plane pixel is touched at most once

Projection Shadow Summary

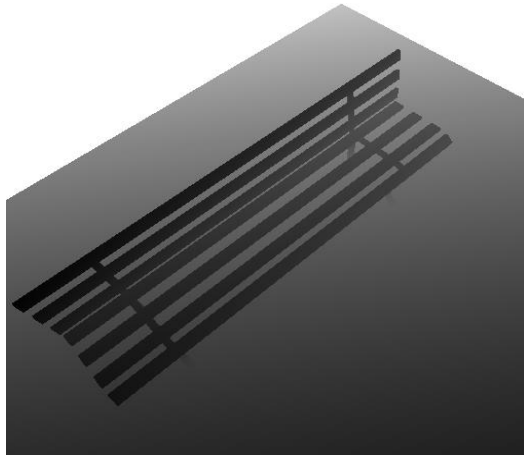
- Easy to implement
 - GLQuake first game to implement it
- Only practical for very few, large receivers
- No self shadowing

Outline

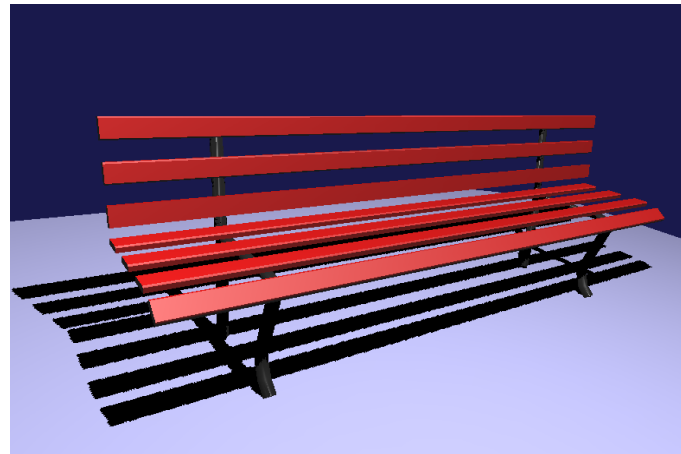
- Motivation & Terminology MPG 12
- Approximate & projection shadows MPG 12.1
- Shadow maps MPG 12.3
- Shadow volumes MPG 12.2
- Summary

Shadow Maps

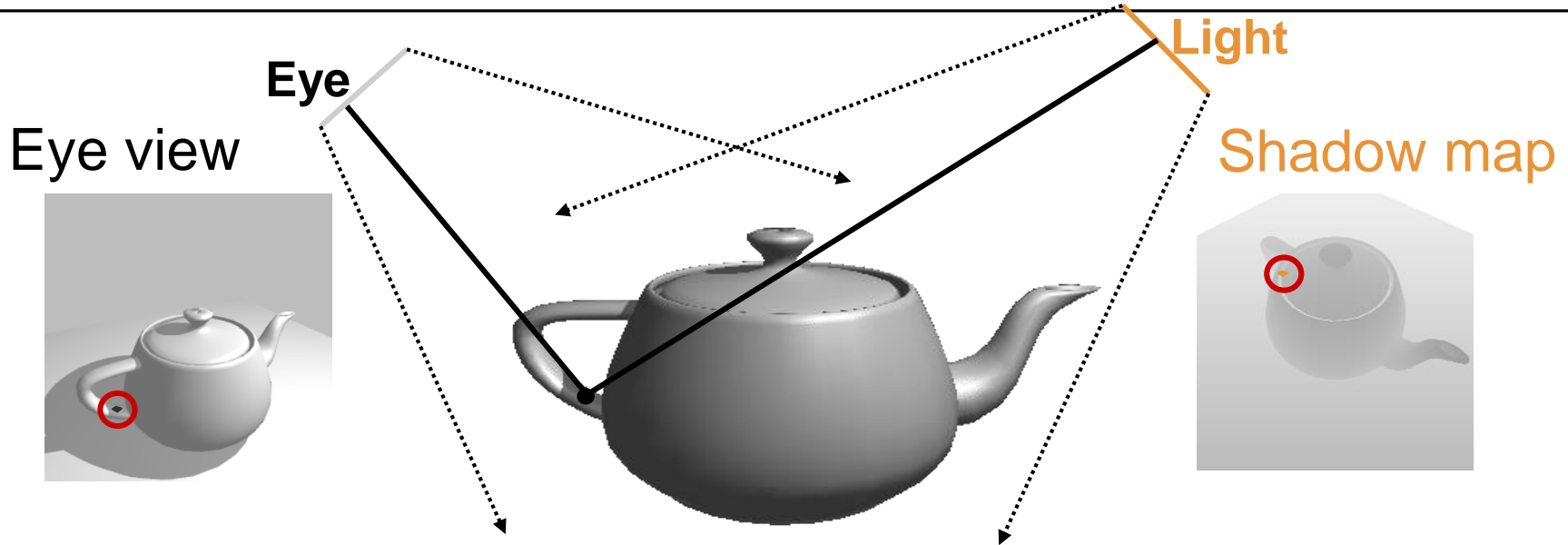
- Casting curved shadows on curved surfaces
 - Image-space algorithm, 2 passes



Shadow map



Shadow Map Algorithm



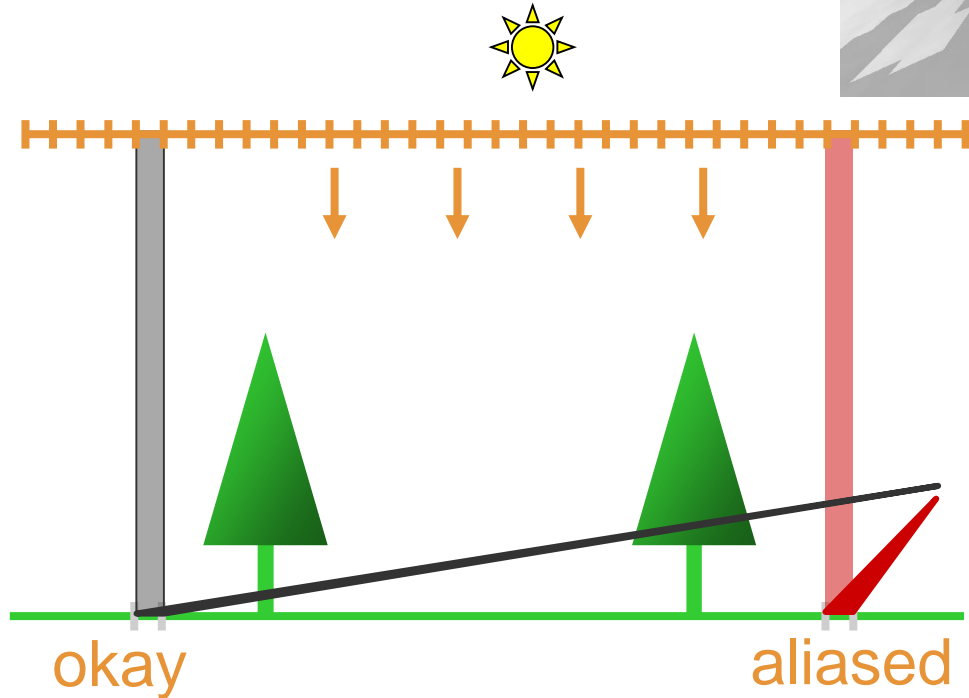
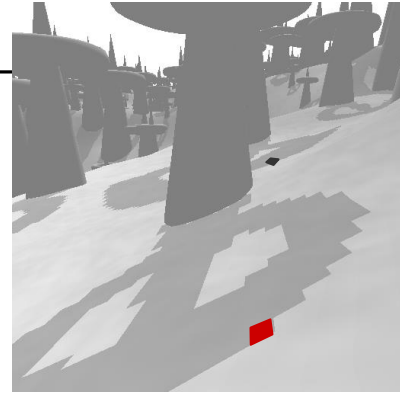
- Render from light; save depth values
- Render from eye
 - Transform all fragments to light space
 - Compare z_{eye} and z_{light} (both in light space!!!)
 - $z_{\text{eye}} > z_{\text{light}}$ \longrightarrow fragment in shadow

Shadow Maps in Hardware

- Render lightspace depth into texture
- In vertex shader:
 - Calculate texture coordinates as in projective texturing
- In fragment shader:
 - Depth compare

Problem: Perspective Aliasing

- Sufficient resolution far from eye
- **Insufficient** resolution near eye



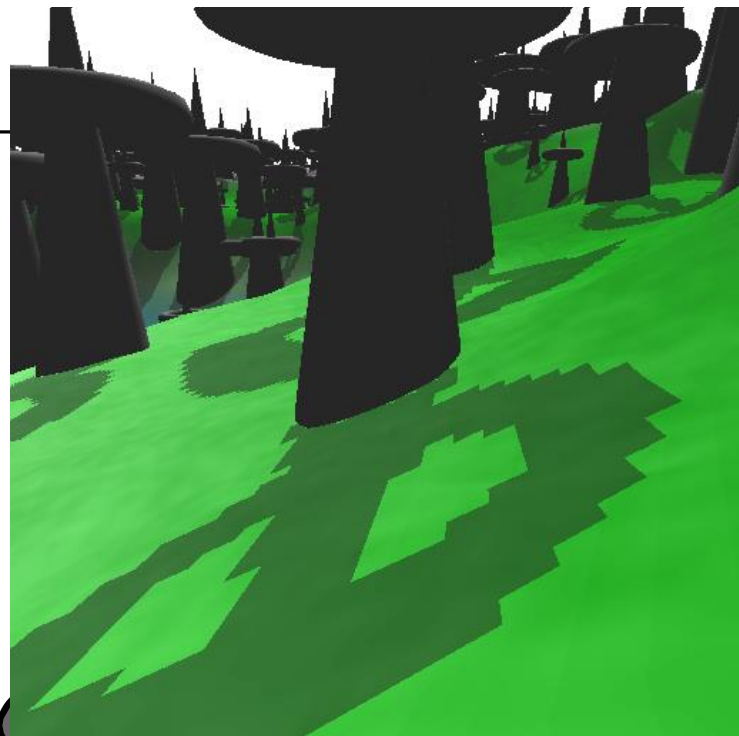
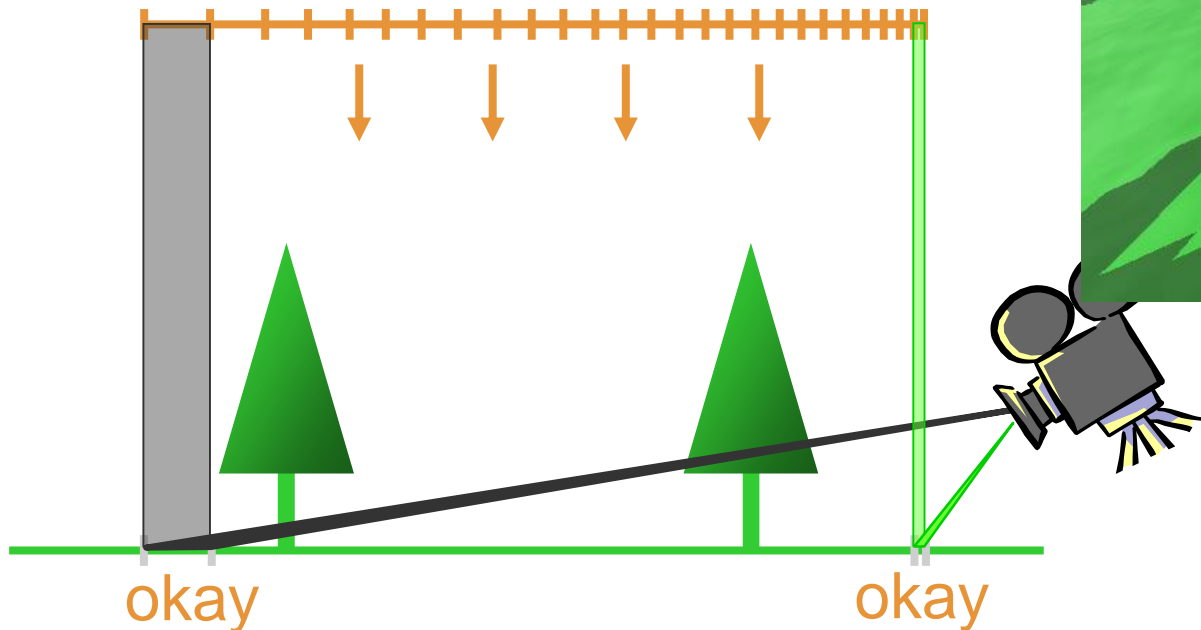
Solution for Perspective Aliasing

- **Insufficient** resolution near eye
- **Redistribute** values in shadow map



Solution for Perspective Aliasing

- **Sufficient** resolution near eye
- **Redistribute** values in shadow map

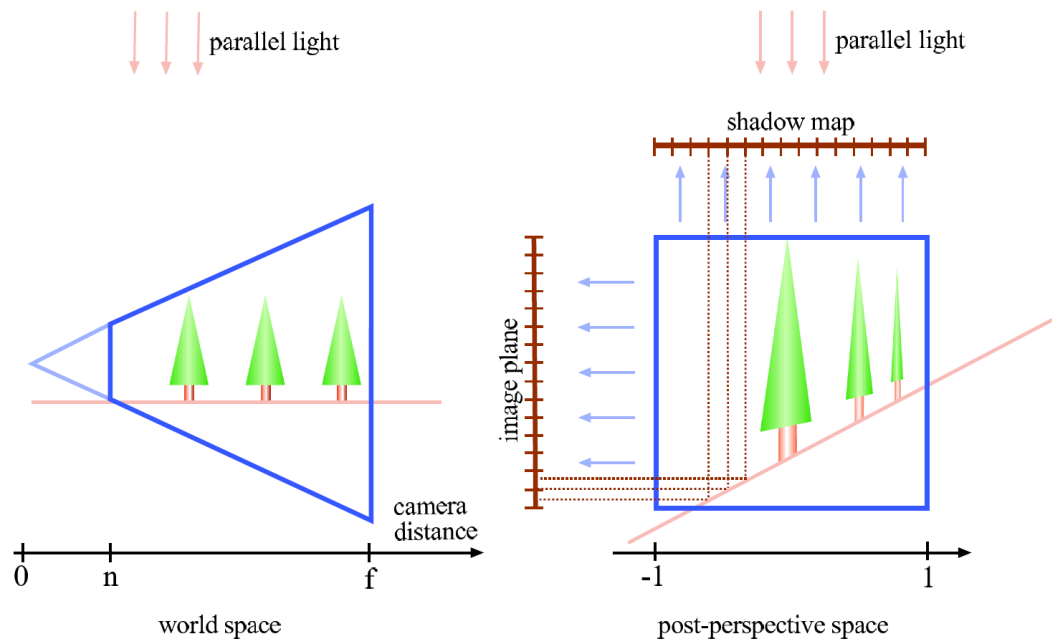


Solution for Perspective Aliasing

- Use warping for light pass (and lookups)



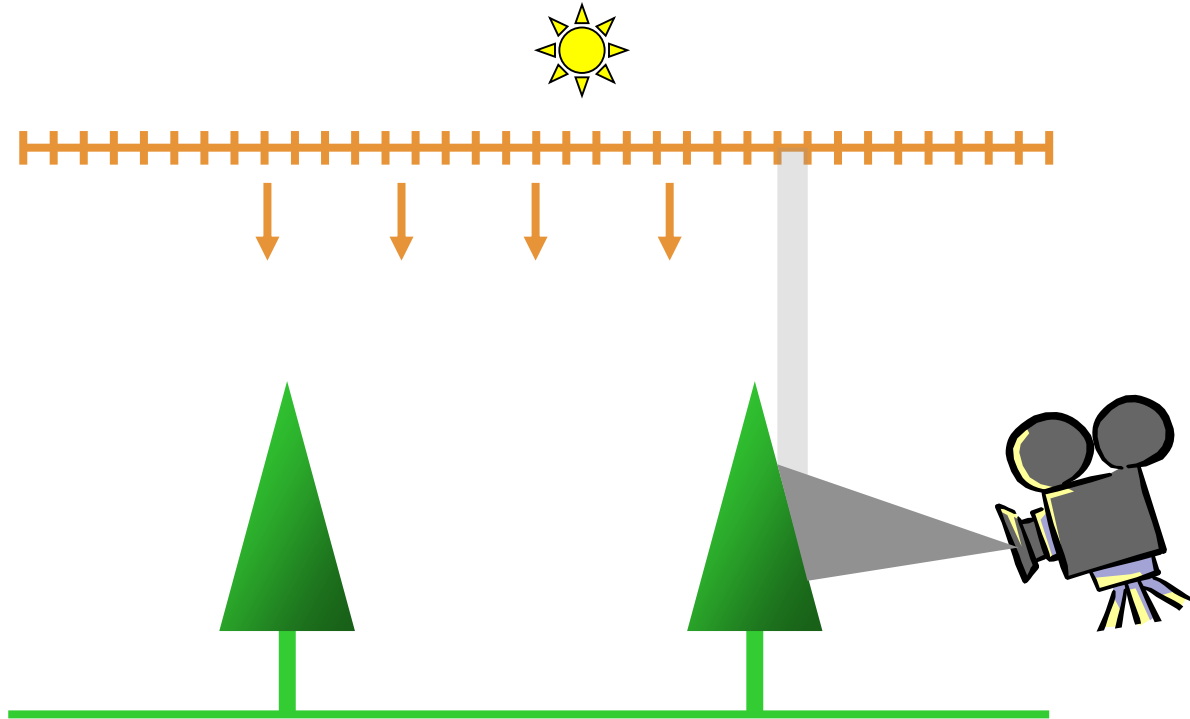
Solution for Perspective Aliasing



- Stamminger, Drettakis - Perspective Shadow Maps
- Wimmer et al. – Light space perspective shadow maps

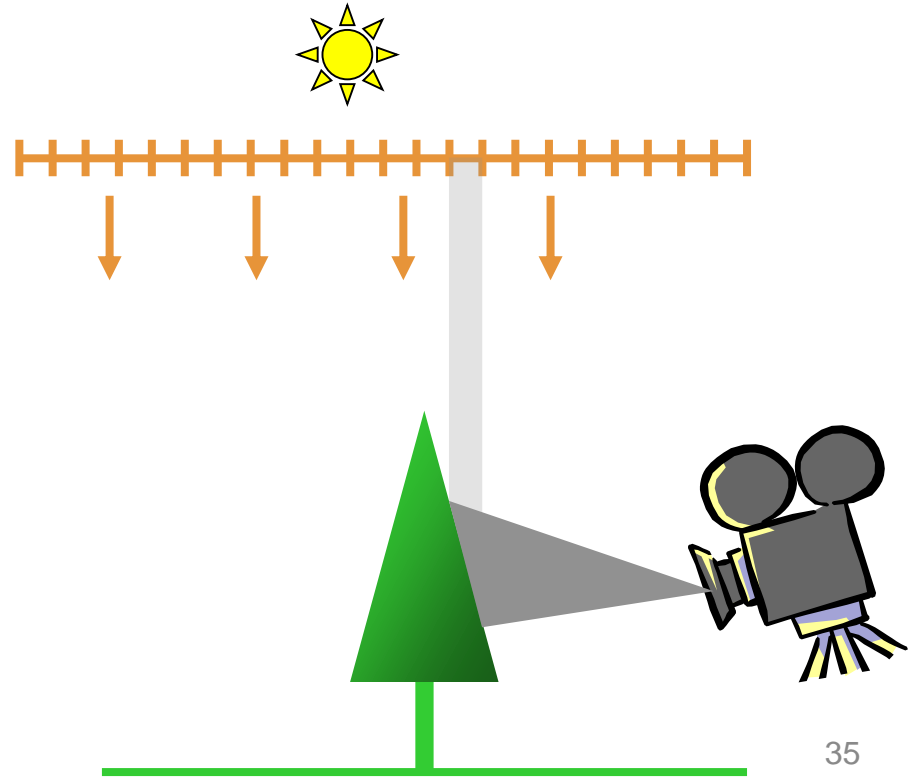
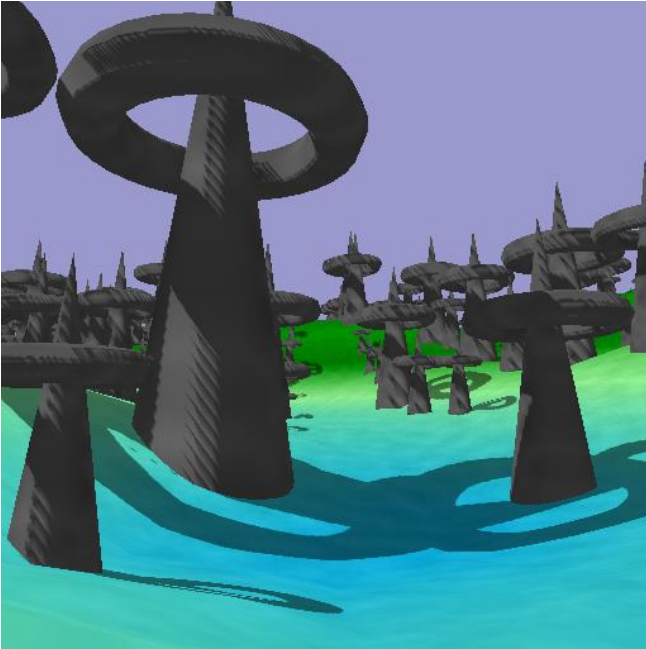
Problem: Projection Aliasing

Shadow receiver ~ **orthogonal** to Shadow Map - viewplane



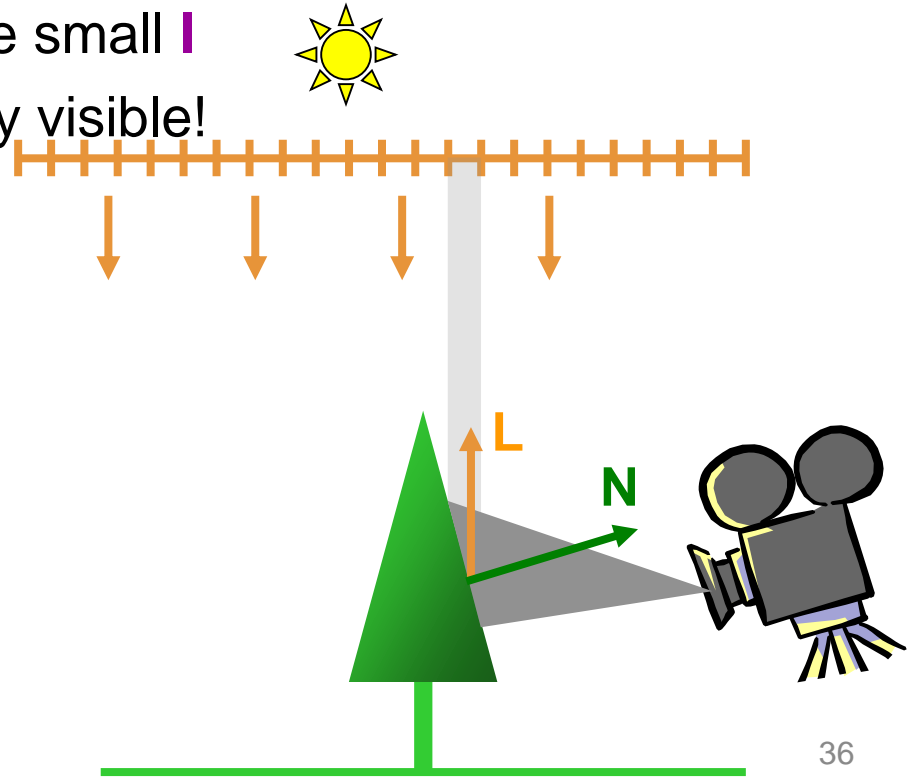
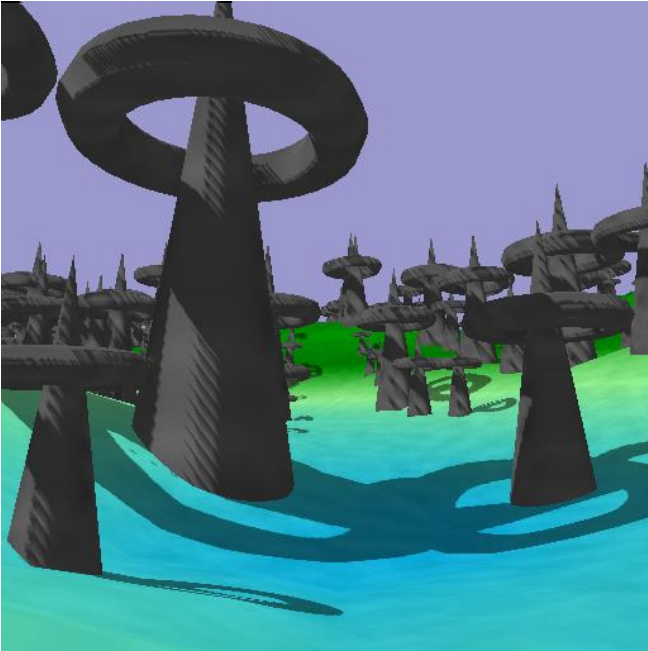
Solution for Projection Aliasing

- Shadow receiver ~ **orthogonal** to Shadow Map plane
- Redistribution does not work
- **But...**



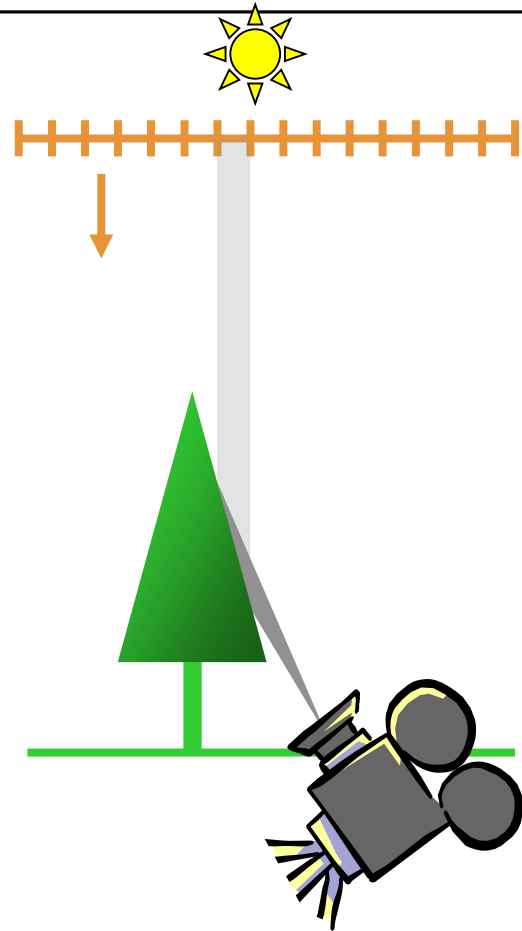
Solution for Projection Aliasing

- Diffuse lighting: $I = I_L \max(\text{dot}(\mathbf{L}, \mathbf{N}), 0)$
- Almost orthogonal receivers have small I
- Dark \longrightarrow artifacts not very visible!

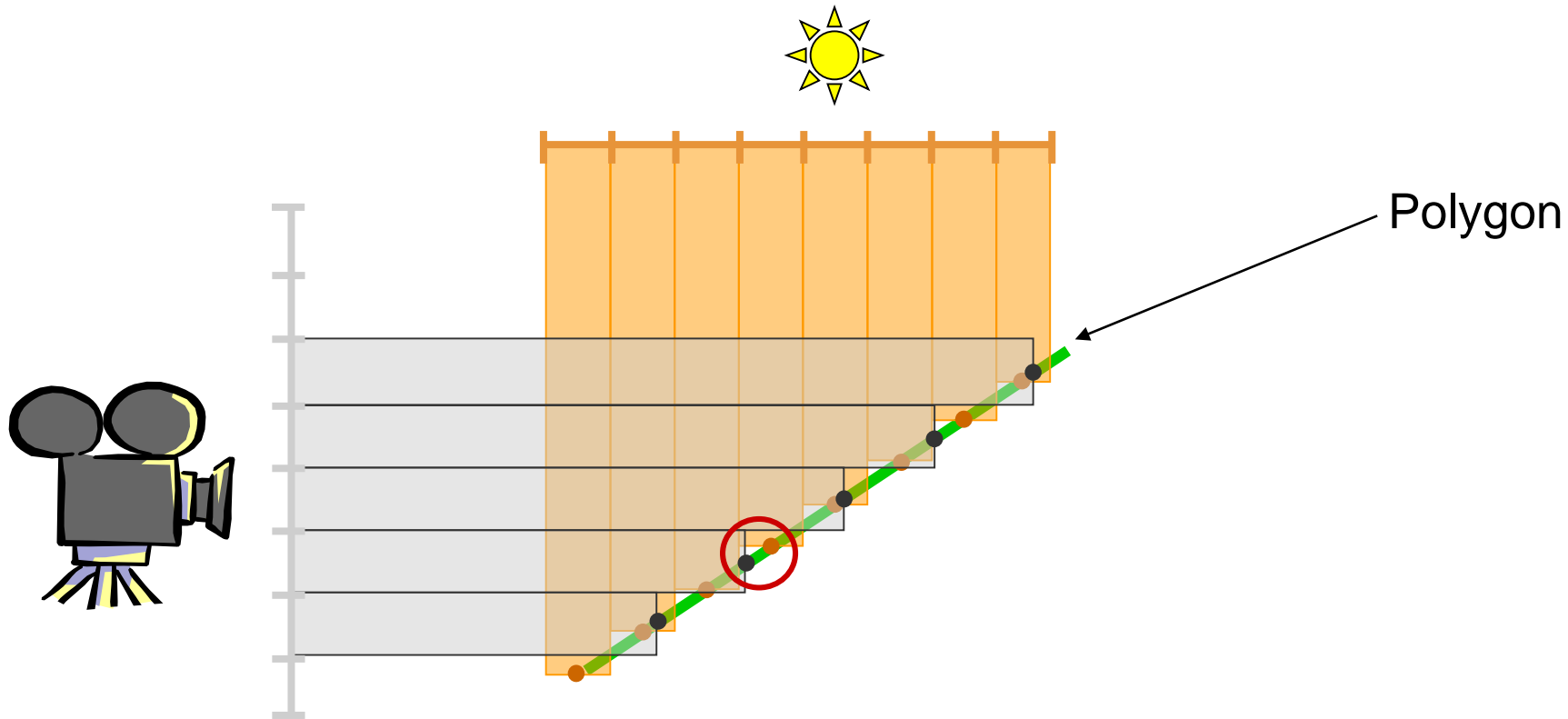


Solution for Projection Aliasing

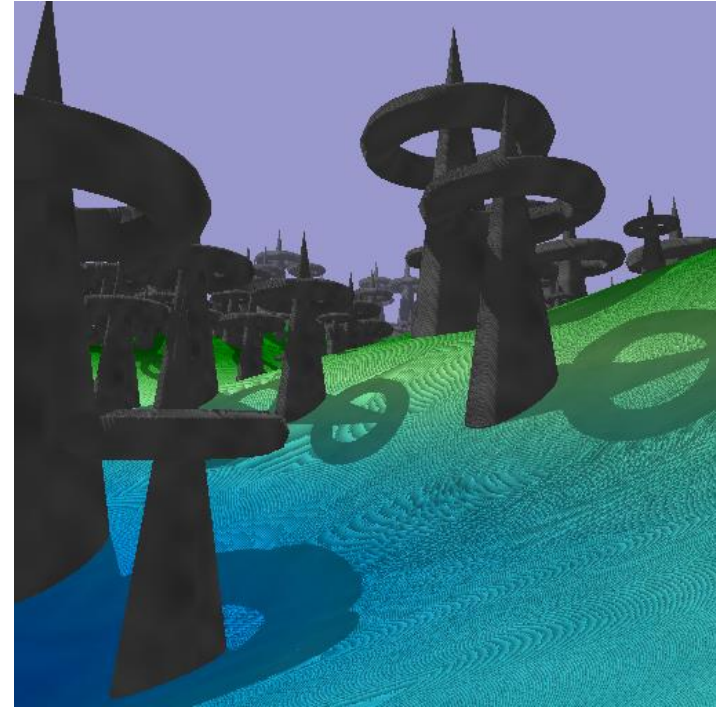
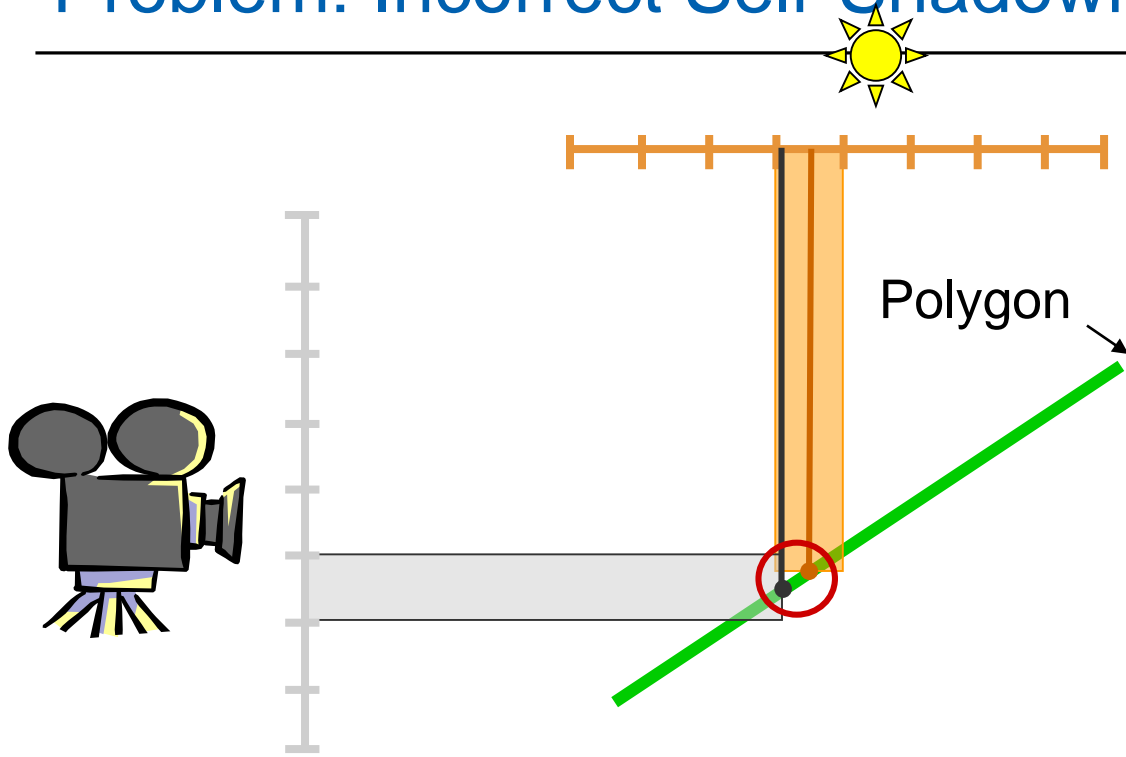
- Recommendations
 - Small **ambient** term
 - **Diffuse term** hides artifacts
 - **Specular term** not problematic
 - Light and view direction almost identical
 - Shadow Map resolution sufficient



Problem: Incorrect Self-Shadowing



Problem: Incorrect Self-Shadowing

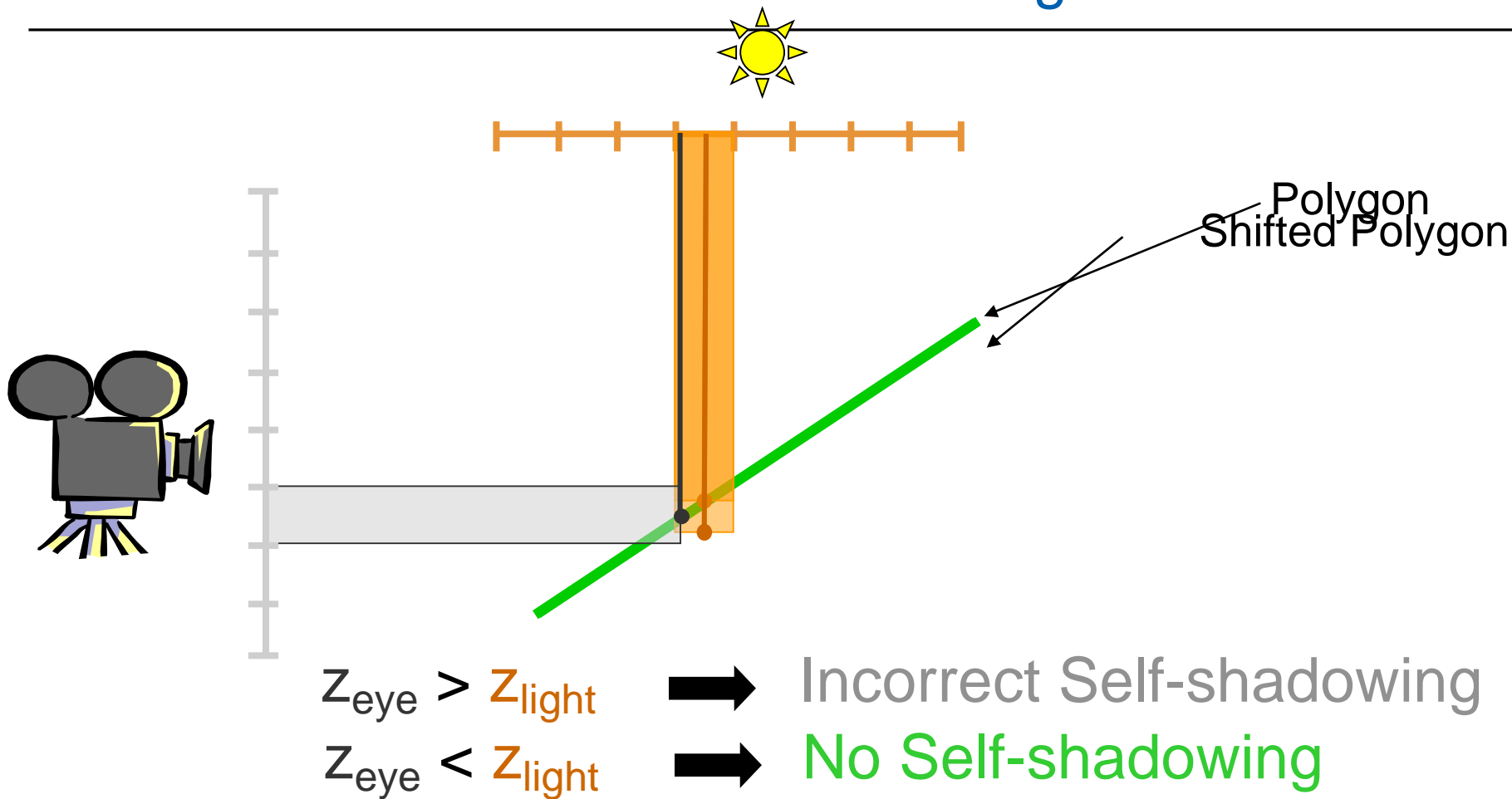


$Z_{\text{eye}} > Z_{\text{light}}$



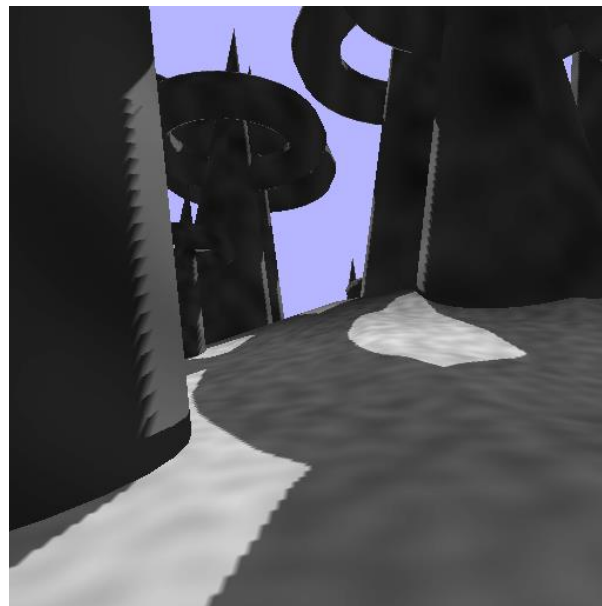
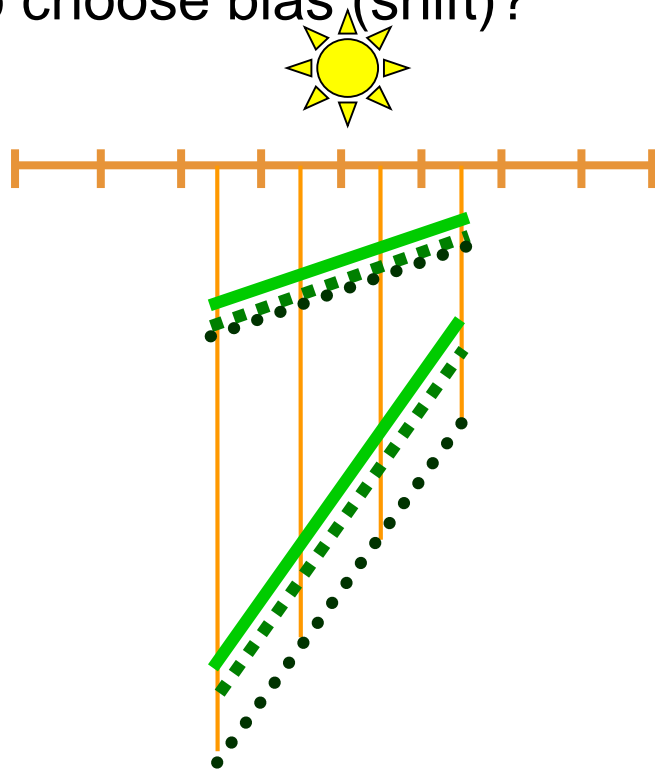
Incorrect Self-shadowing

Solution for Incorrect Self-Shadowing



Solution for Incorrect Self-Shadowing

- How to choose bias (shift)?

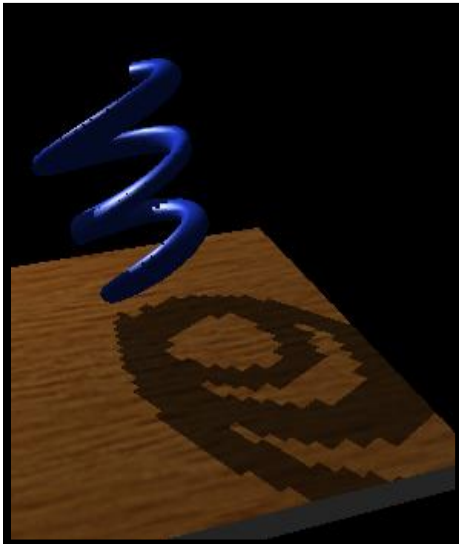


- No Bias**
- - - Constant Bias**
- Slope-Scale Bias**

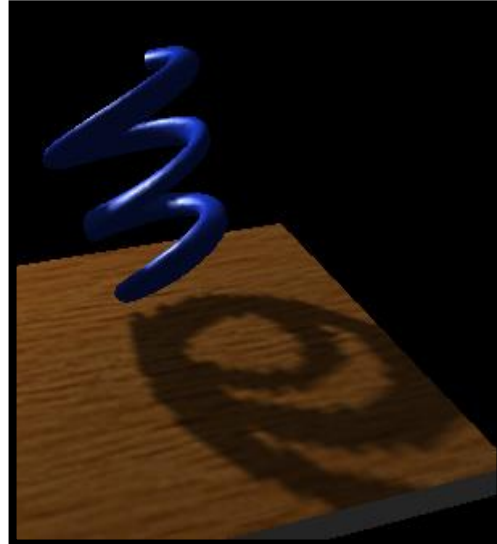
Percentage closer filtering

- Filter lookup result, not depth map values!
- Normal color filtering cannot be used
- 2x2 PCF in hardware for NVIDIA

NEAREST



LINEAR PCF



Shadow Map Summary

- Advantages

- Fast – only one additional pass
- Independent of scene complexity (no additional shadow polygons!)
- Self shadowing
- Can sometimes reuse depth map

- Disadvantages

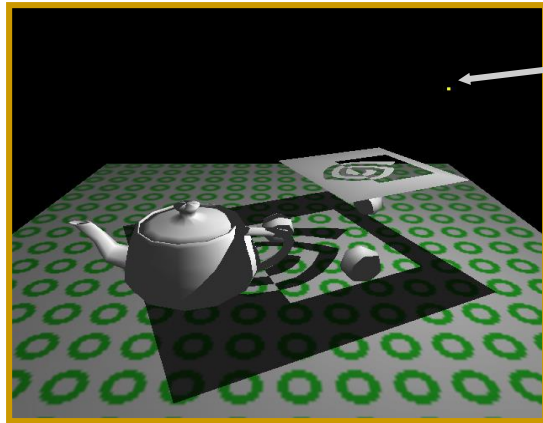
- Problematic for omnidirectional lights
- Biasing tweaks (light leaks, surface acne)
- Jagged edges (aliasing)

Outline

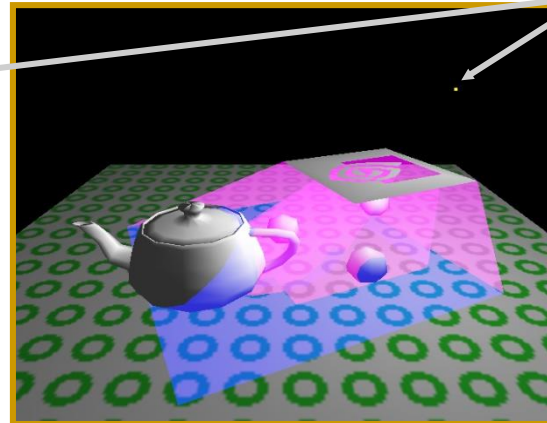
- Motivation & Terminology MPG 12
- Approximate & projection shadows MPG 12.1
- Shadow maps MPG 12.3
- Shadow volumes MPG 12.2
- Summary

Shadow Volumes (Crow 1977)

- Occluders and light source cast out a 3D shadow volume
 - Shadow through new geometry
 - Results in pixel correct shadows



Shadowed scene



Visualization of shadow volume

Light
source

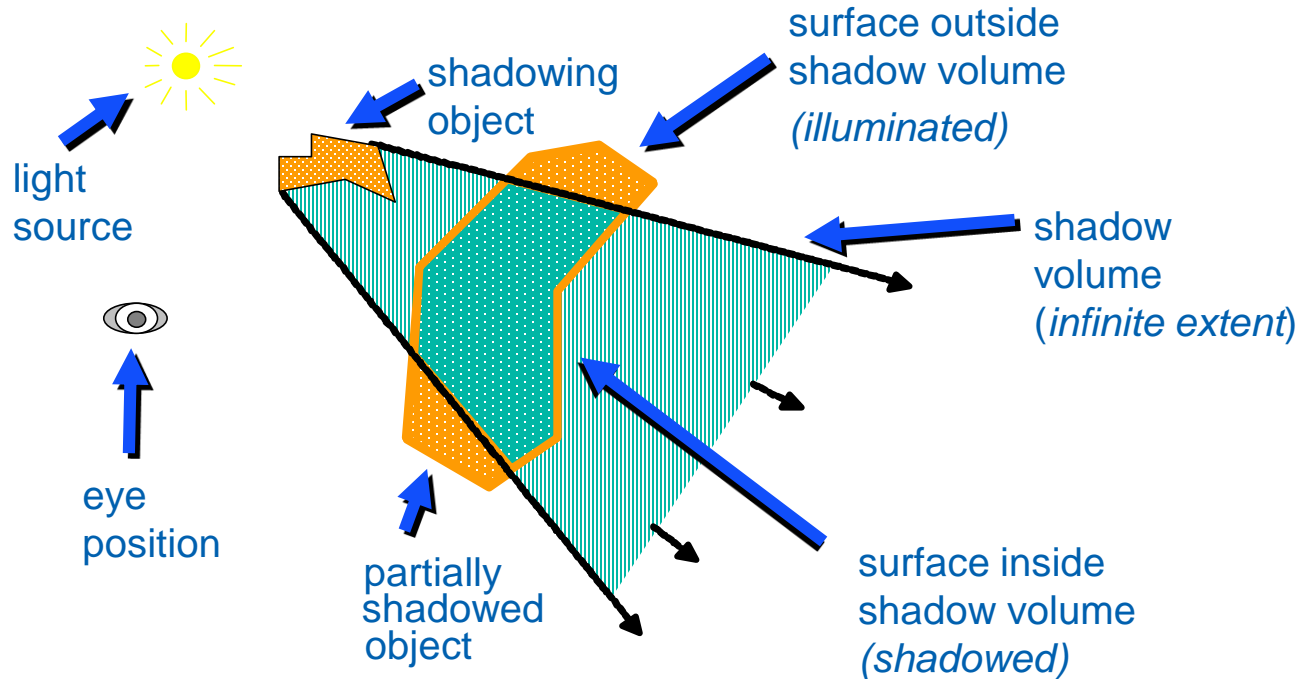
Shadow Volumes (Crow 1977)

- Heavily used in Doom3



2D Cutaway of Shadow Volume

- Occluder polygons extruded to semi-infinite volumes

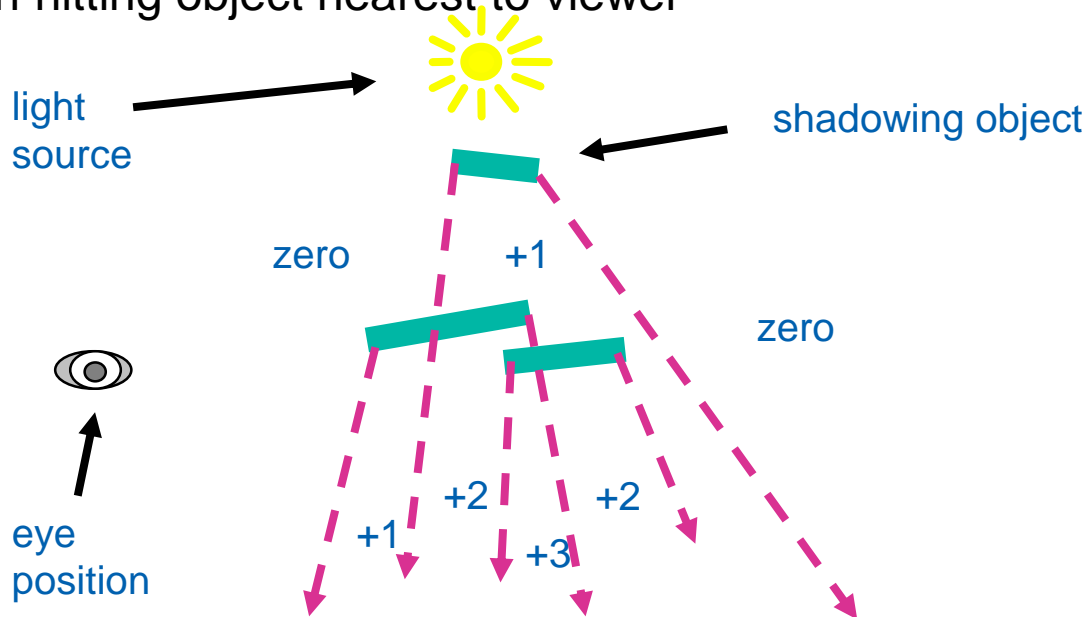


Shadow Volume Algorithm

- 3D point-in-polyhedron inside-outside test
- Principle similar to 2D point-in-polygon test
 - Choose a point known to be outside the volume
 - Count ray intersections from test point to known point with polyhedron faces
 - Front face +1
 - Back face -1

Enter/Leave Approach

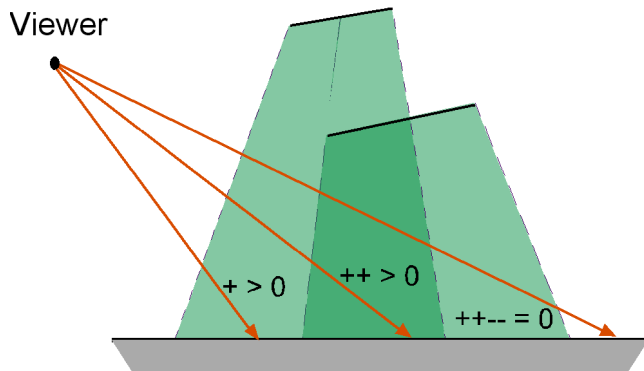
- Increment on enter, decrement on leave
- Simultaneously test all visible pixels
→ Stop when hitting object nearest to viewer



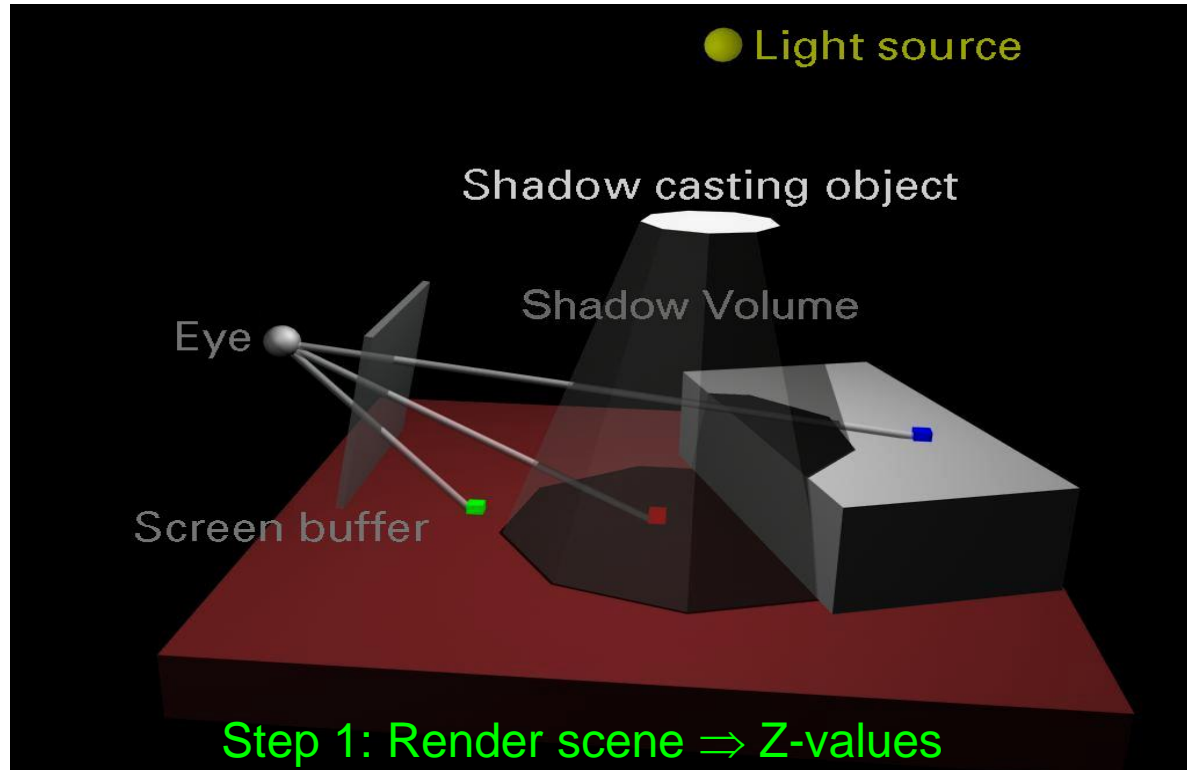
Shadow Volume Algorithm

- Shadow volumes in object precision
 - Calculated by CPU/Vertex Shaders
- Shadow test in image precision
 - Using stencil buffer as counter!

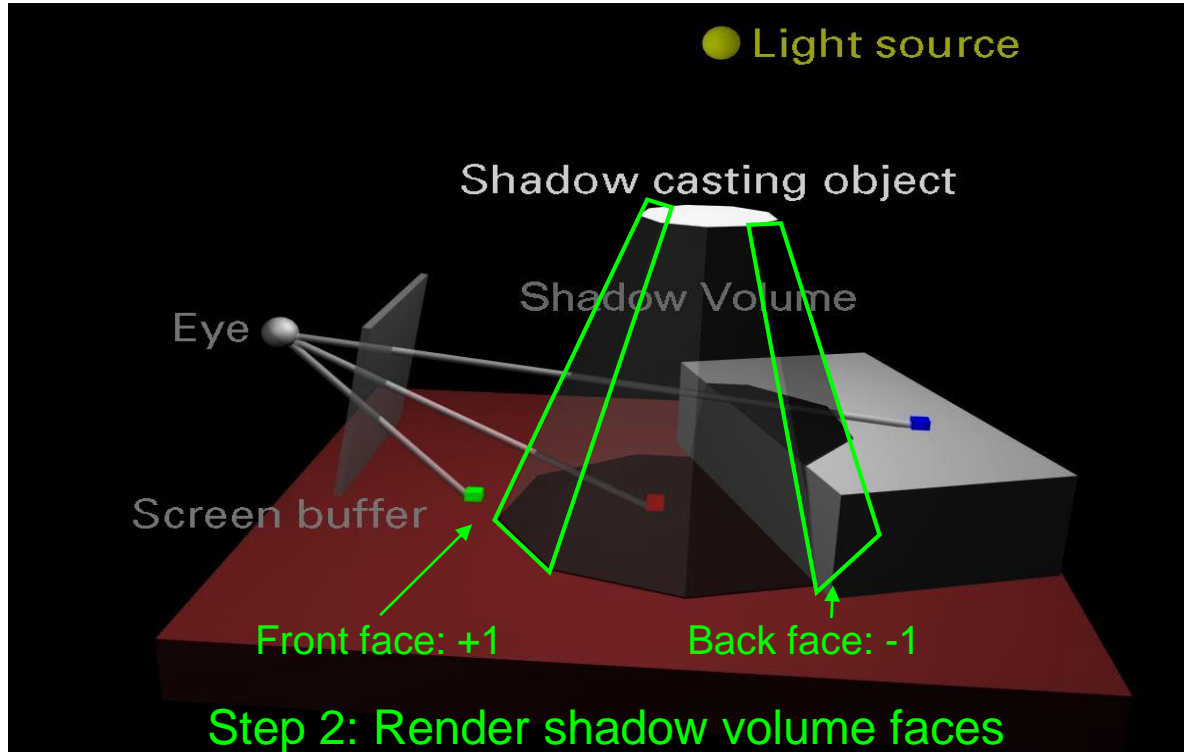
• Light Source



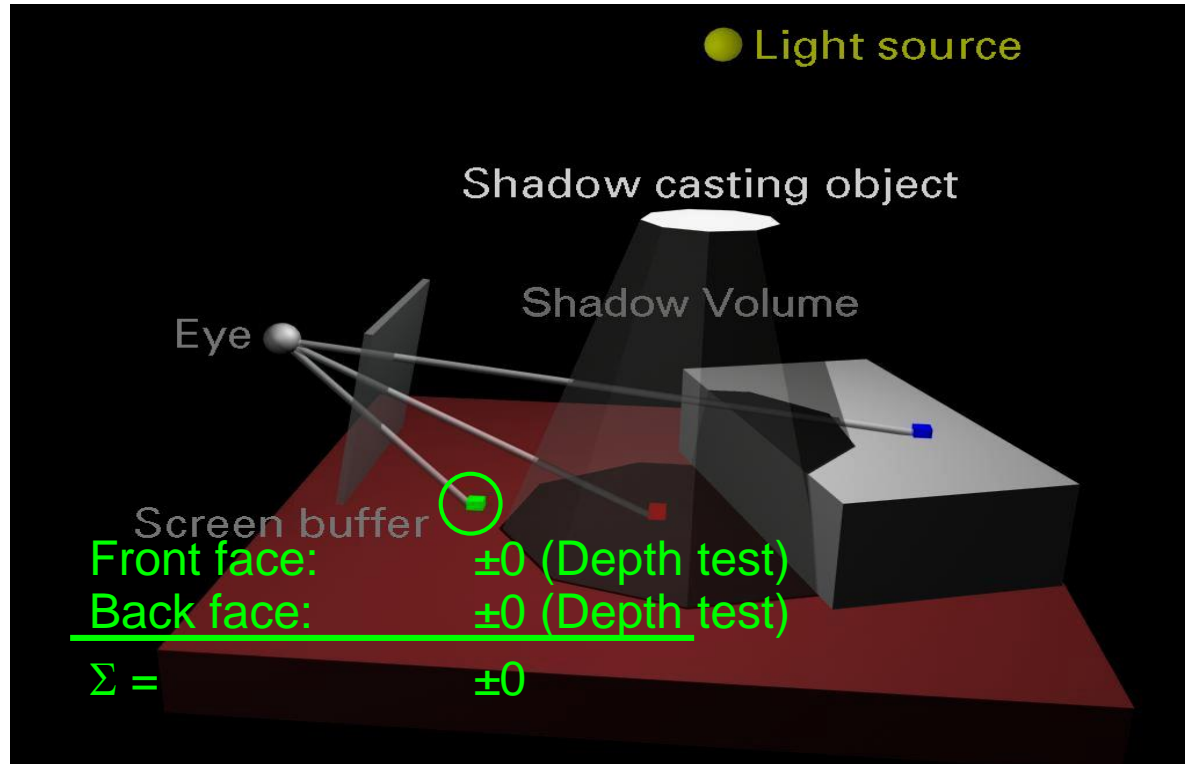
Shadow Volume Algorithm



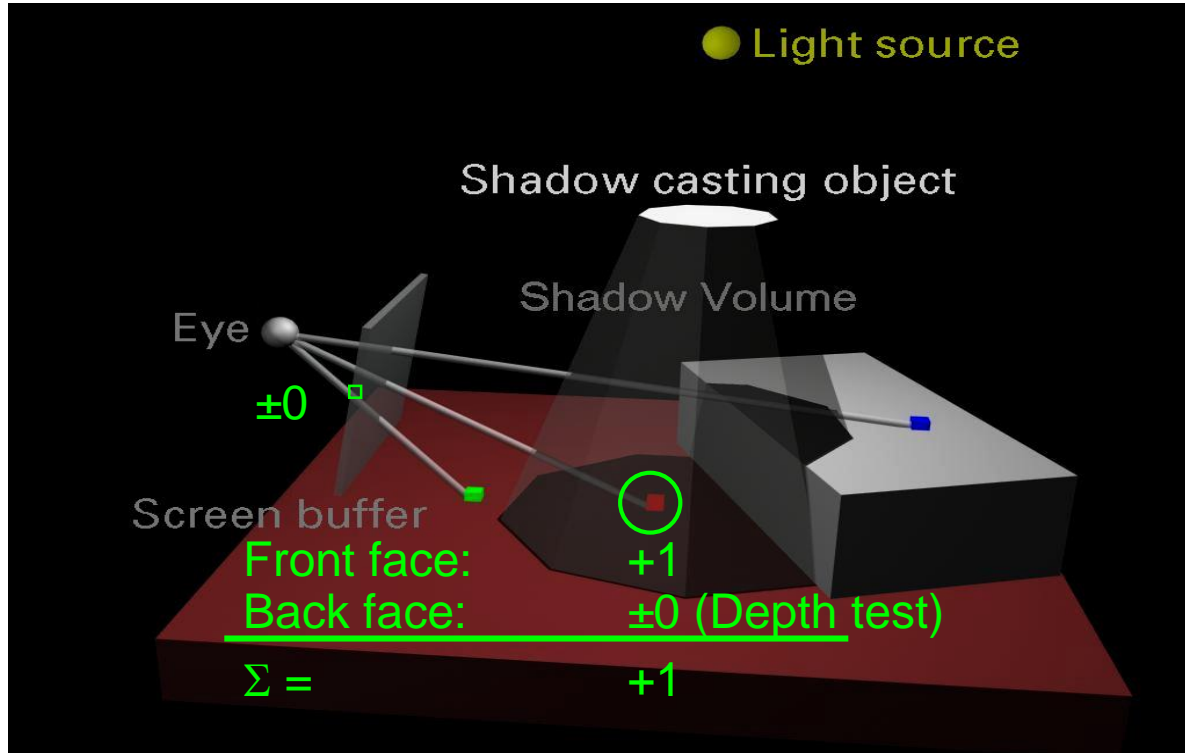
Shadow Volume Algorithm



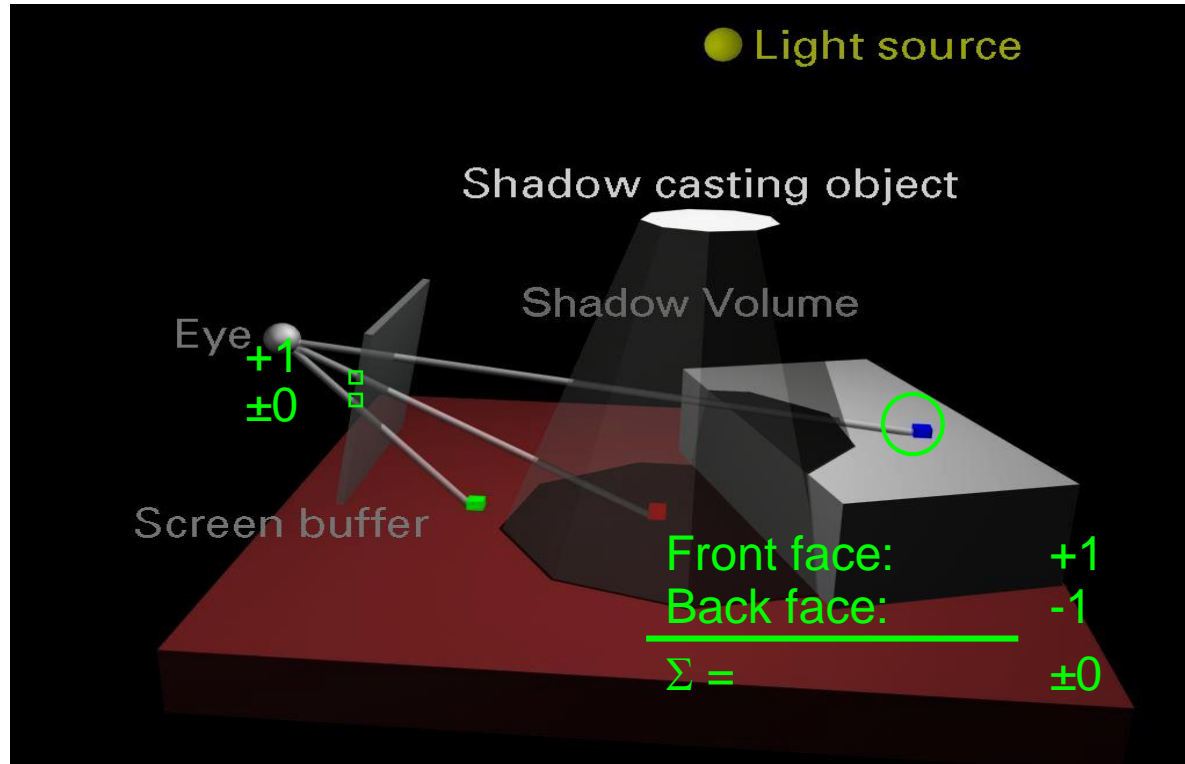
Shadow Volume Algorithm



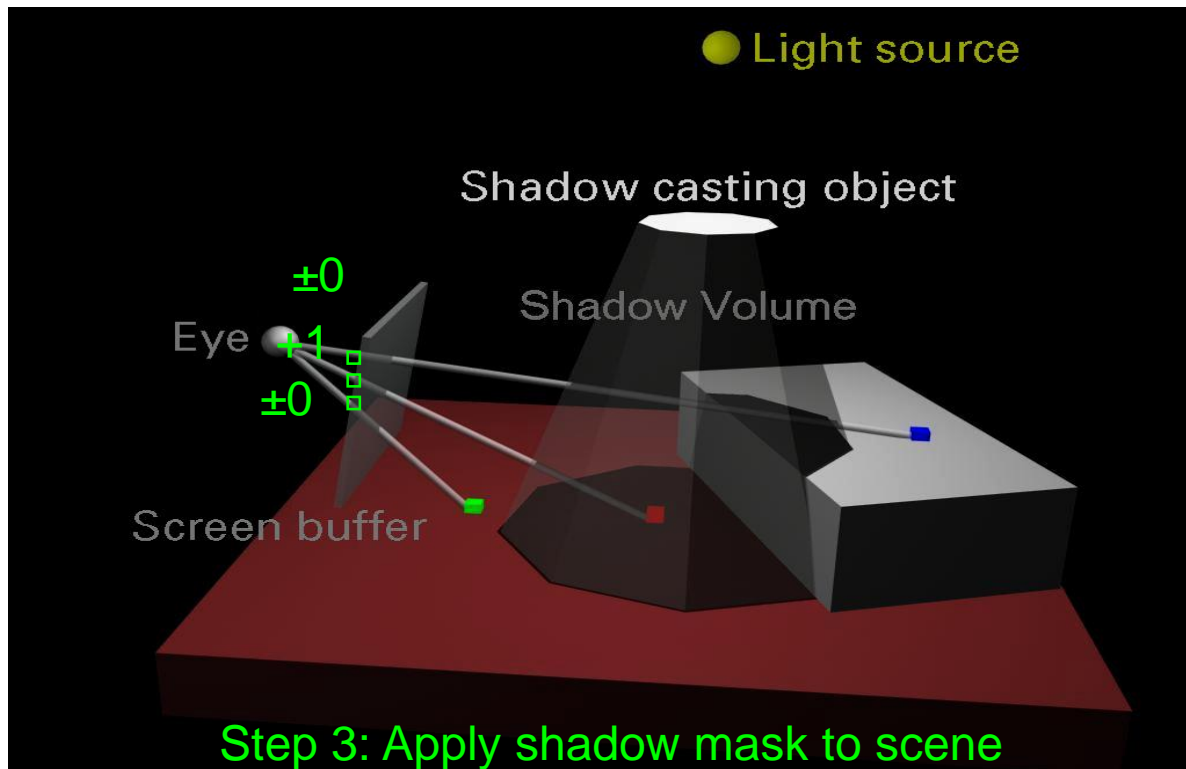
Shadow Volume Algorithm



Shadow Volume Algorithm



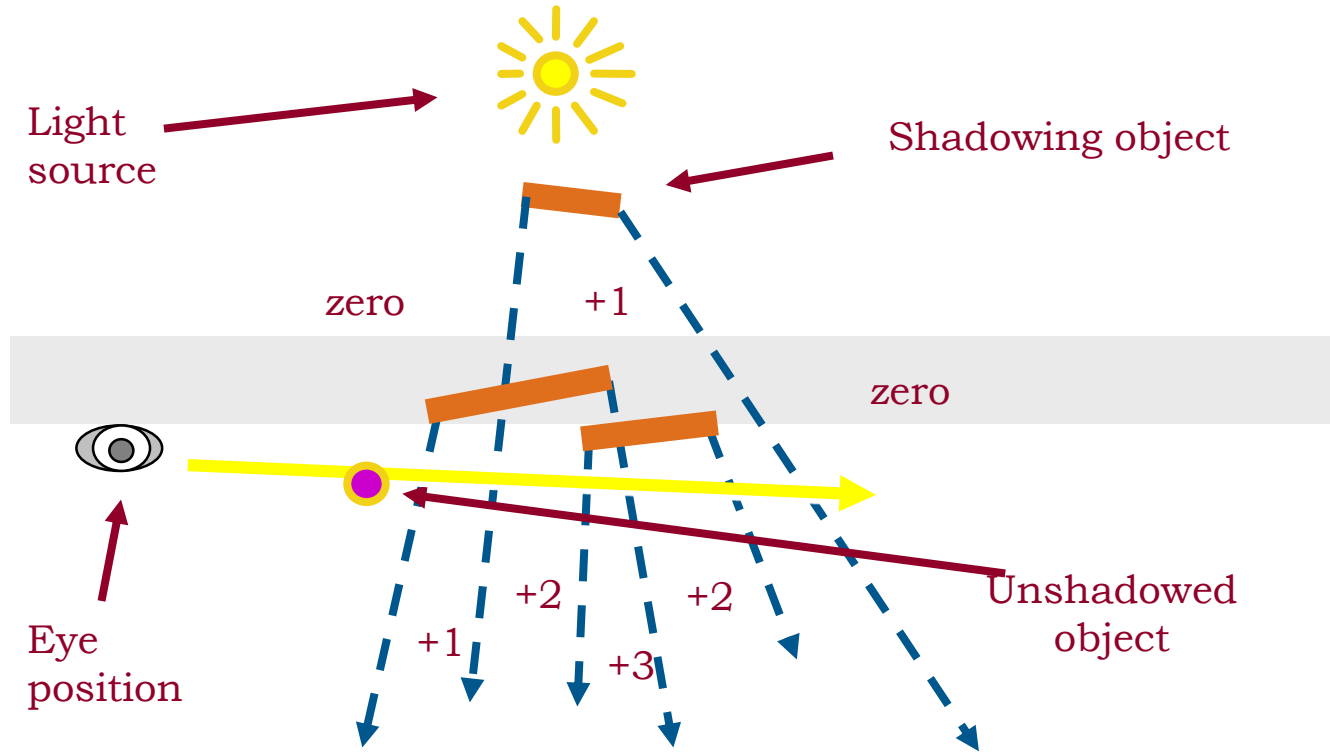
Shadow Volume Algorithm



Shadow Volume Algorithm (Zpass)

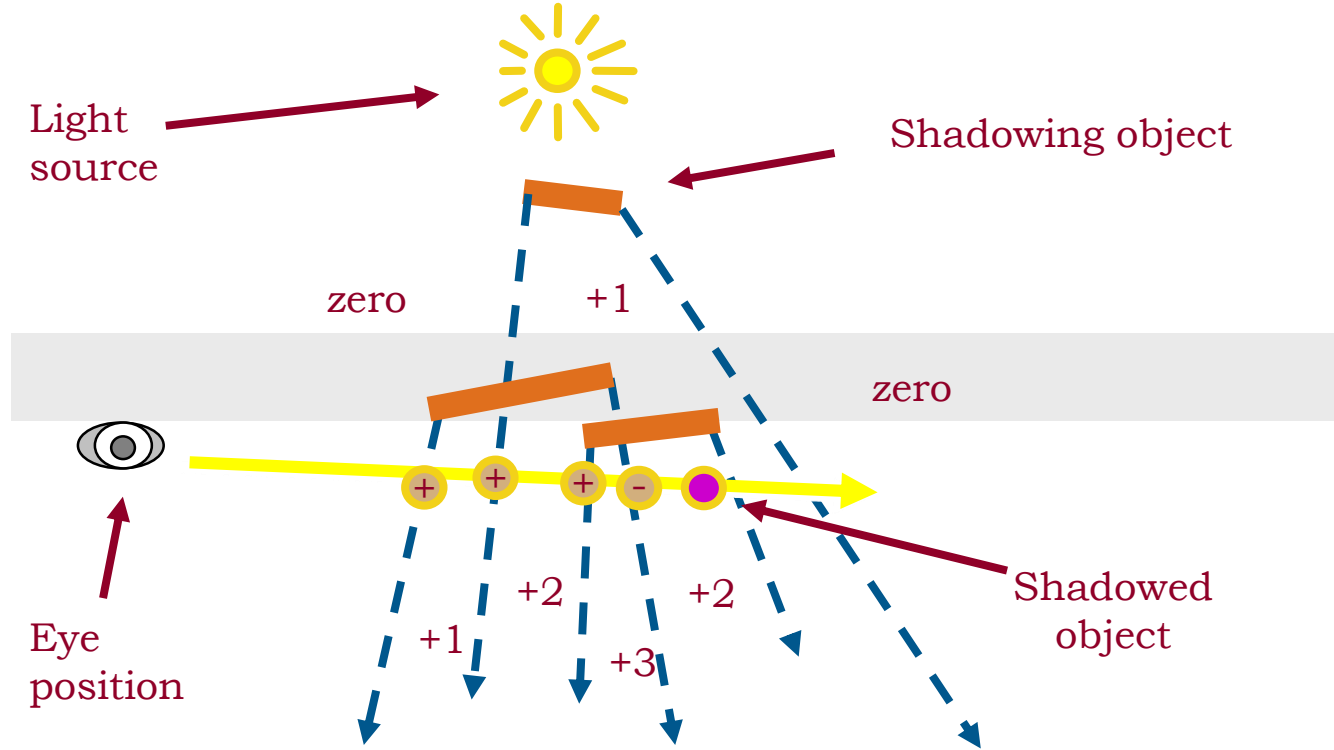
- Render scene to establish z-buffer
 - Can also do ambient illumination
- For each light
 - Clear stencil
 - Draw shadow volume twice using culling
 - Render front faces and increment stencil
 - Render back faces and decrement stencil
 - Illuminate all pixels not in shadow volume
 - Render testing stencil = 0
 - Use additive blend

Zpass Technique (Before Shadow)



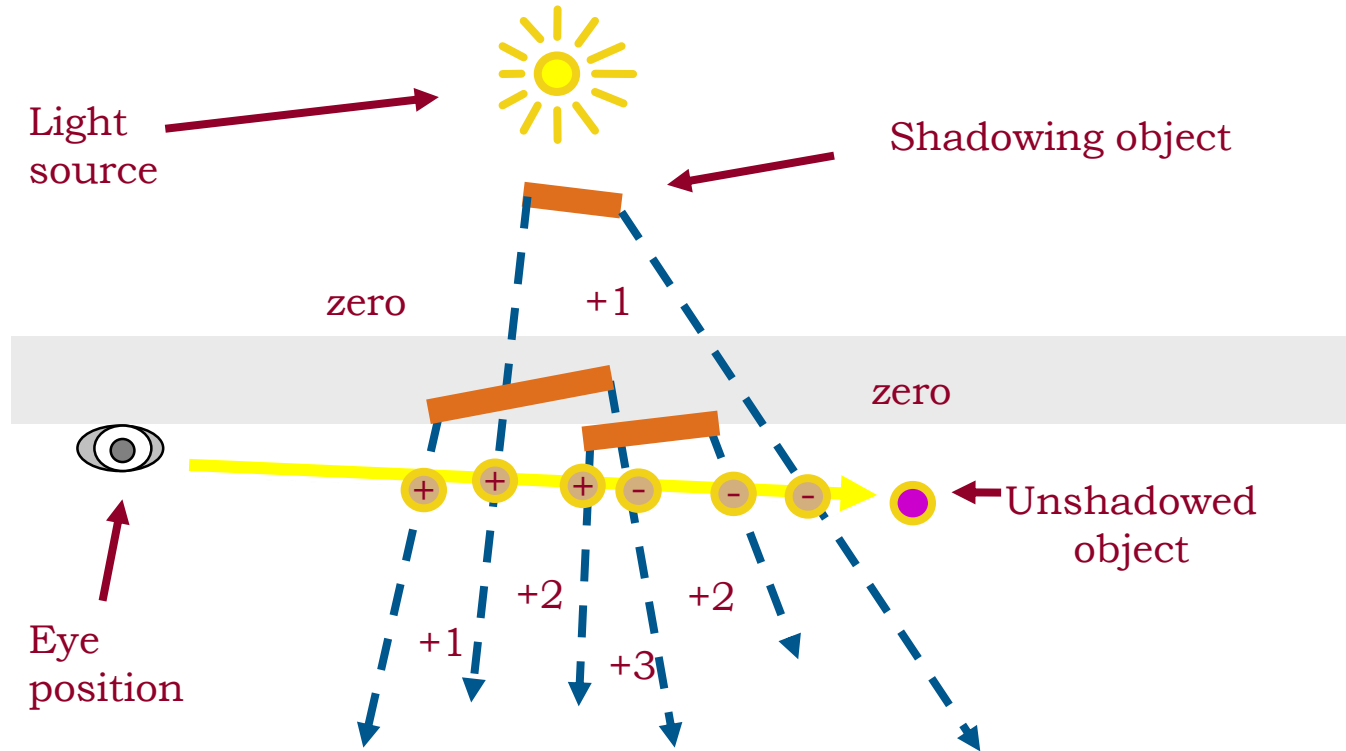
Shadow Volume Count = 0 (no depth tests passes)

Zpass Technique (In Shadow)



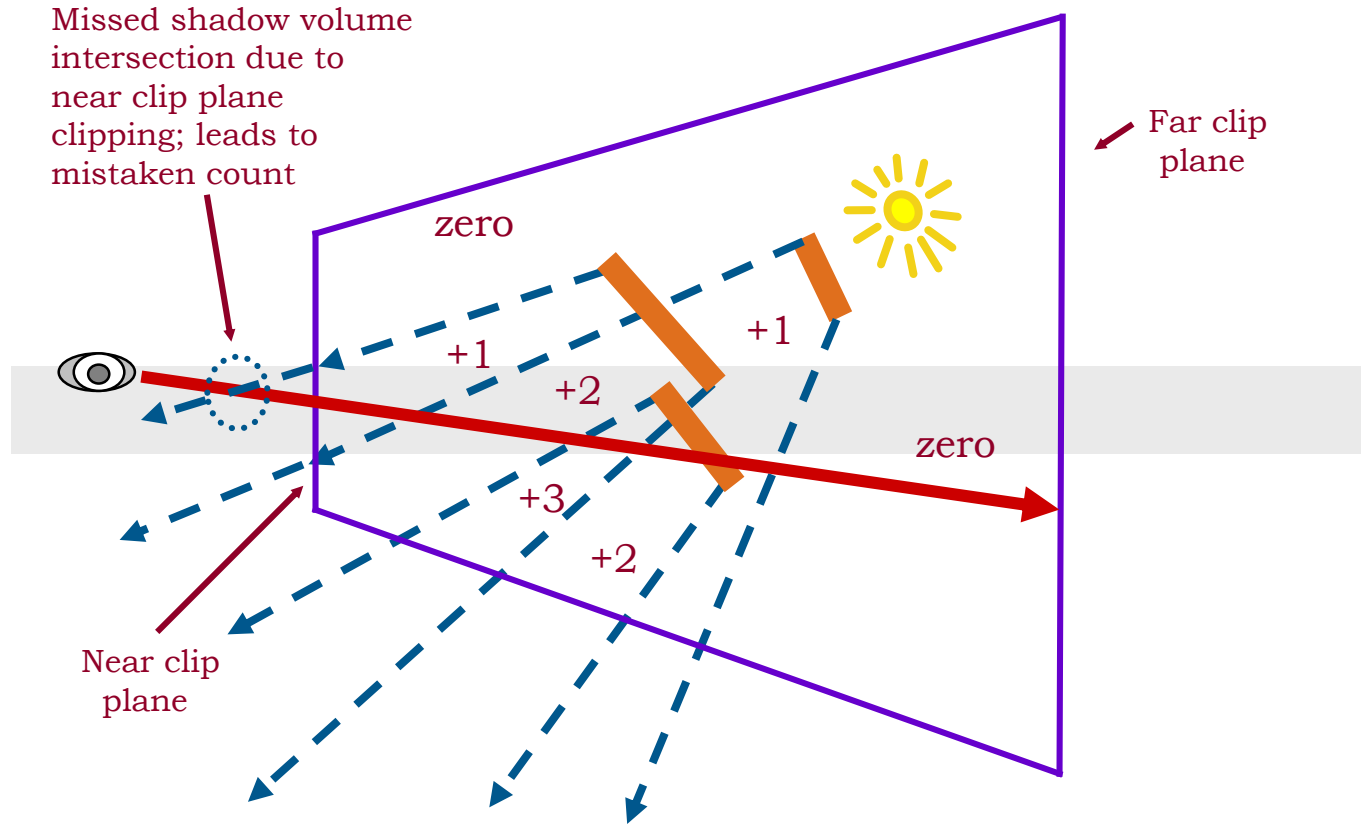
Shadow Volume Count = $+1+1+1-1 = 2$

Zpass Technique (Behind Shadow)



Shadow Volume Count = +1+1+1-1-1-1 = 0

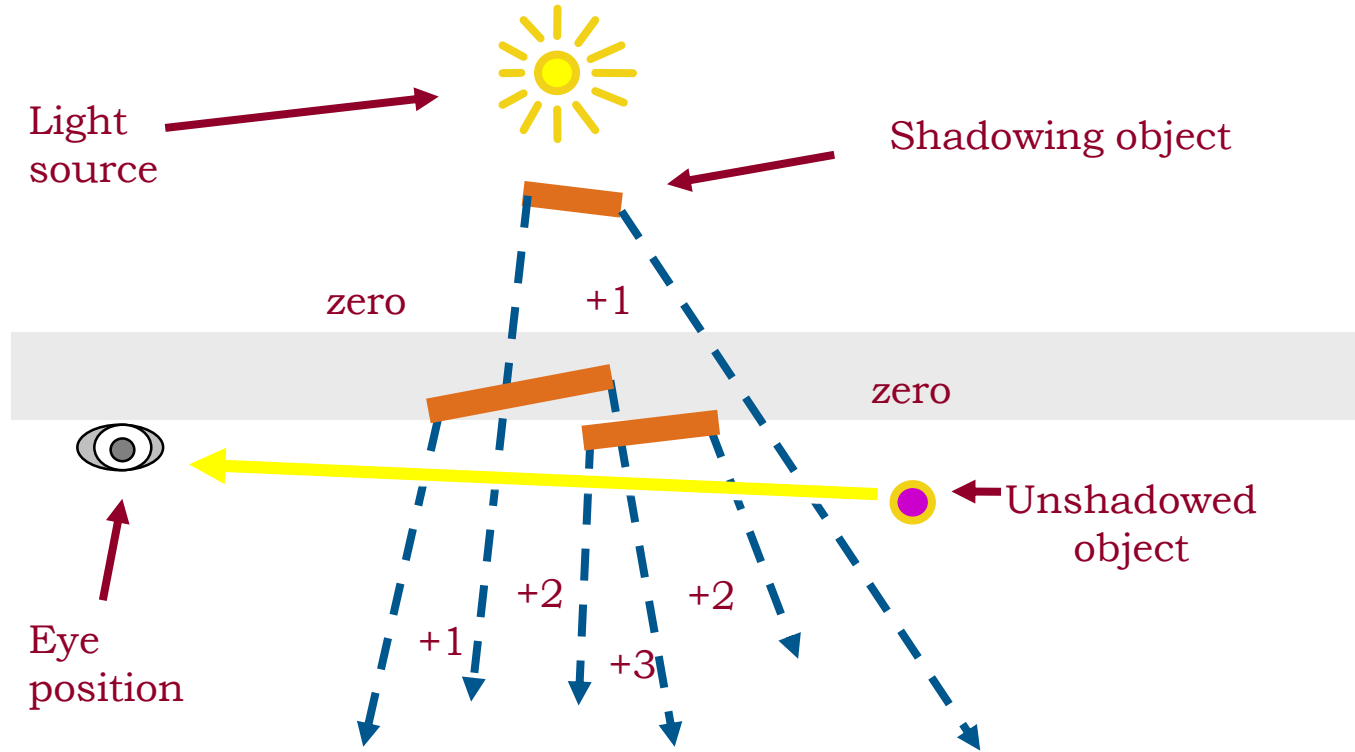
Zpass Near Plane Problem



Alternative: Zfail Technique

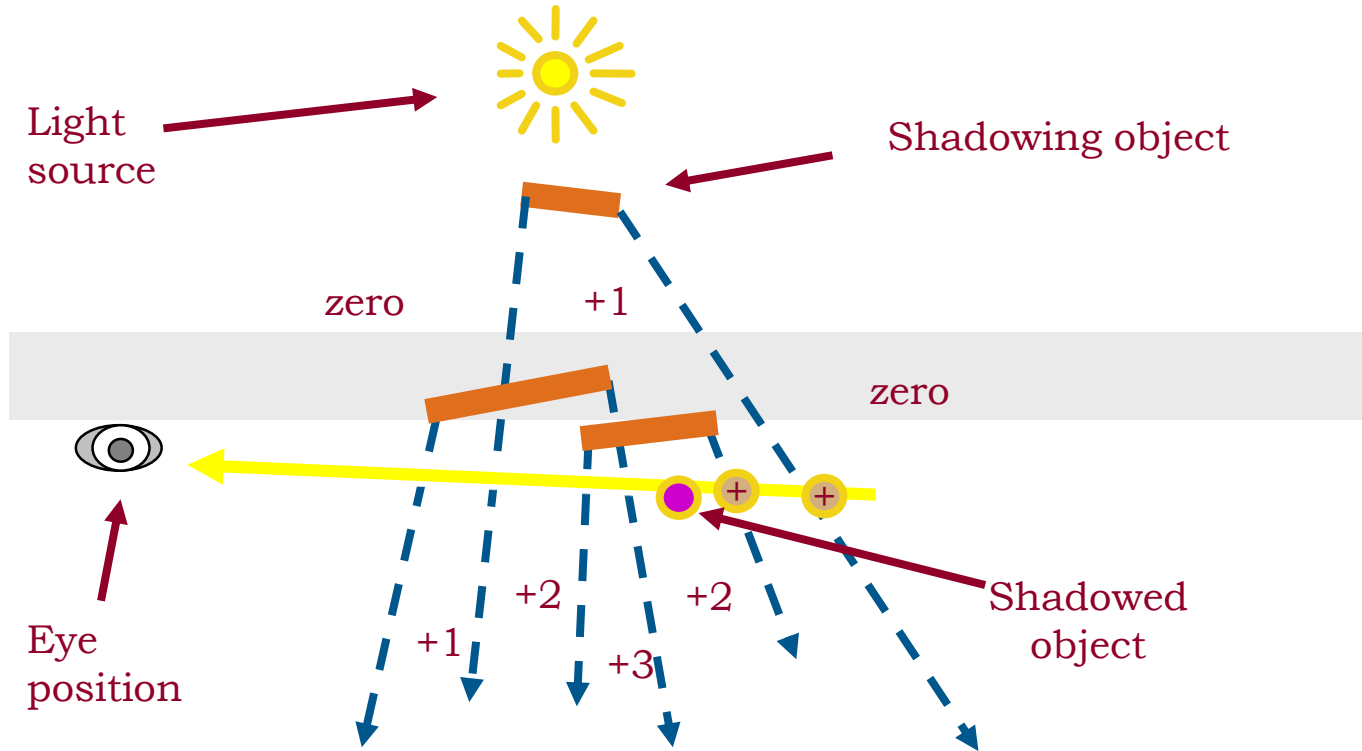
- Zpass near plane problem difficult to solve
 - Have to “cap” shadow volume at near plane
 - Expensive and not robust, many special cases (camera in shadow)
- **Reversing test order** → Zfail technique
(also known as Carmack’s reverse)
 - Start from infinity and stop at nearest intersection
 - Render shadow volume fragments only when depth test fails
 - Render back faces first and increment
 - Then front faces and decrement
 - Need to cap shadow volume at infinity or light extent
 - Special projection matrix that moves zfar to infinity! [Kilgard2004]

Zfail, Behind Shadow



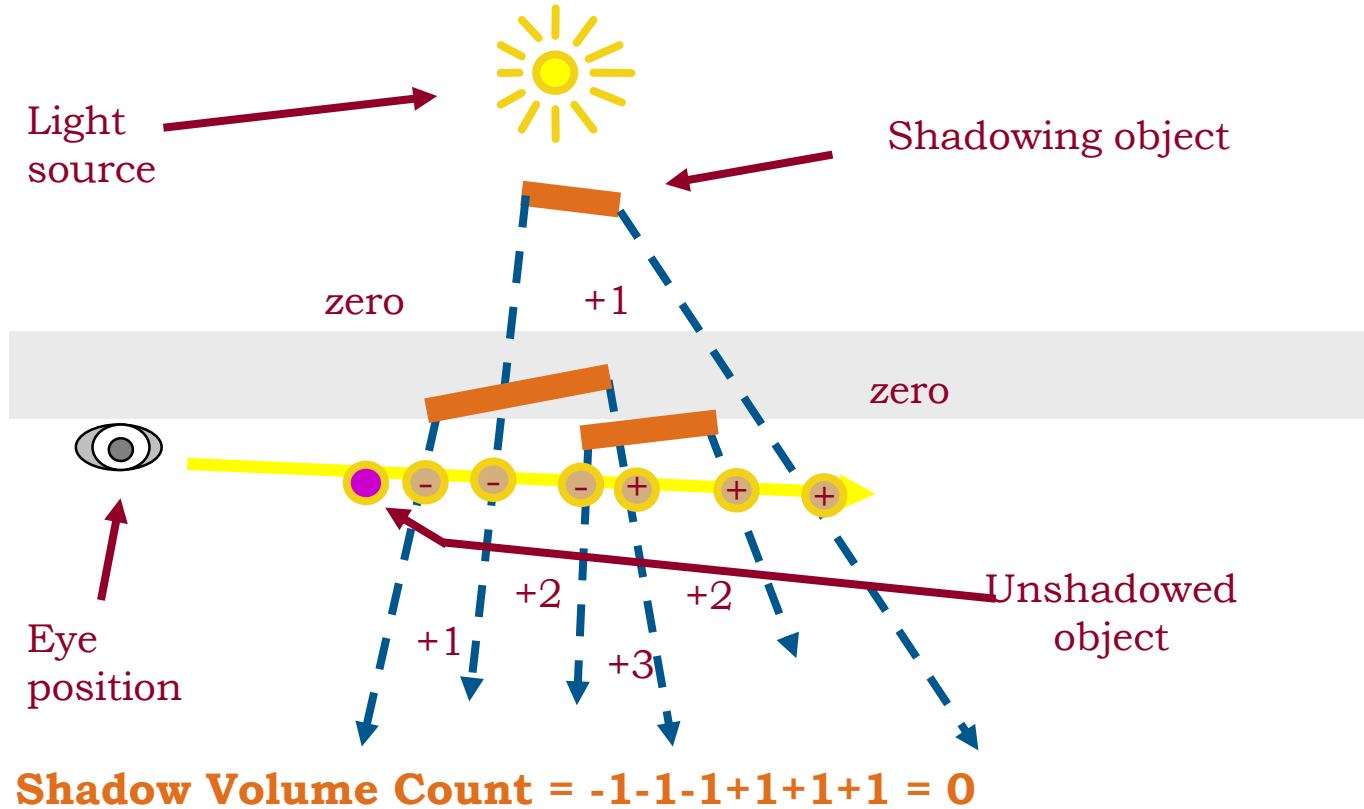
Shadow Volume Count = 0 (zero depth tests fail)

Zfail, in Shadow



Shadow Volume Count = +1+1 = 2

Zfail, before Shadow



Shadow Volumes

- Shadow volume = closed polyhedron
- Actually 3 sets of polygons!
 1. Object polygons facing the light (“light cap”)
 2. Object polygons facing away from the light and projected to infinity (with $w = 0$) (“dark cap”)
 3. Actual shadow volume polygons (extruded object edges) (“sides”)
→ but which edges?

Computing Actual SV Polygons

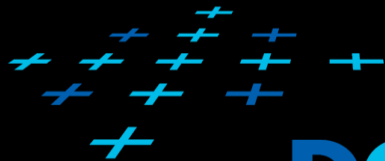
- Trivial but bad: one volume per triangle
 - 3 shadow volume polygons per triangle
- Better: find exact silhouette
 - Expensive on CPU
- Even better: **possible silhouette edges**
 - Edge shared by a back-facing and front-facing polygon (with respect to light source!), extended to infinity
 - Actual extrusion can be done by vertex shader

Shadow Volumes Summary

- Advantages
 - Arbitrary receivers
 - Fully dynamic
 - Omnidirectional lights (unlike shadow maps!)
 - Exact shadow boundaries (pixel-accurate)
 - Automatic self shadowing
 - Broad hardware support (stencil)
- Disadvantages
 - Fill-rate intensive
 - Difficult to get right (Zfail vs. Zpass)
 - Silhouette computation required
 - Doesn't work for arbitrary casters (smoke, fog...)

Outline

- Motivation & Terminology MPG 12
- Approximate & projection shadows MPG 12.1
- Shadow maps MPG 12.3
- Shadow volumes MPG 12.2
- Summary



DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Questions?