Shadows and Reflections in Real Time

Tomas Akenine-Möller
Department of Computer Engineering
Chalmers University of Technology

Slides co-developed with Eric Haines

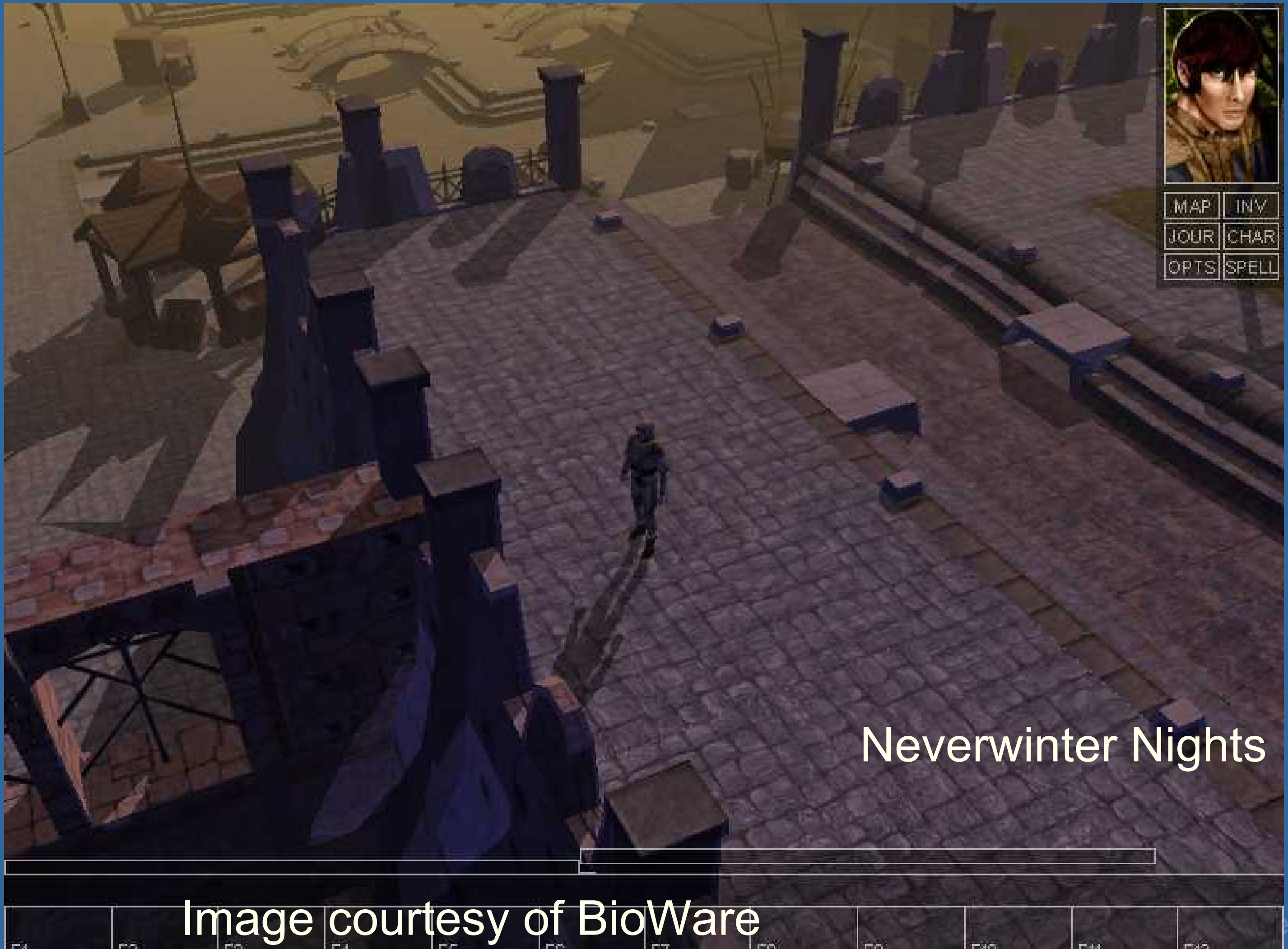
Reading Material

MUST read

- These slides
- OH 298-302 by Magnus Bondesson

Why shadows?

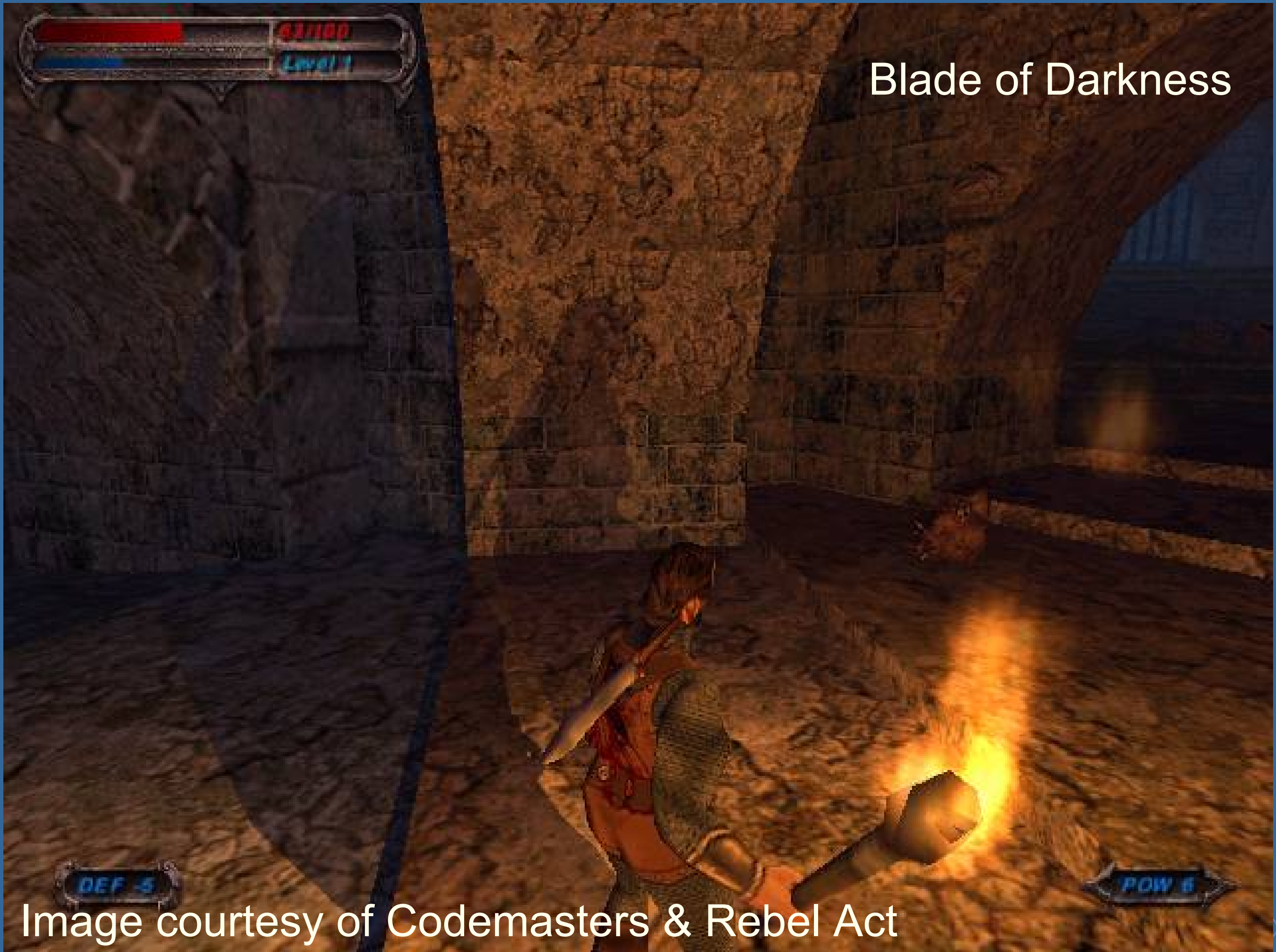
- More realism and atmosphere



Neverwinter Nights

Image courtesy of BioWare

Another example

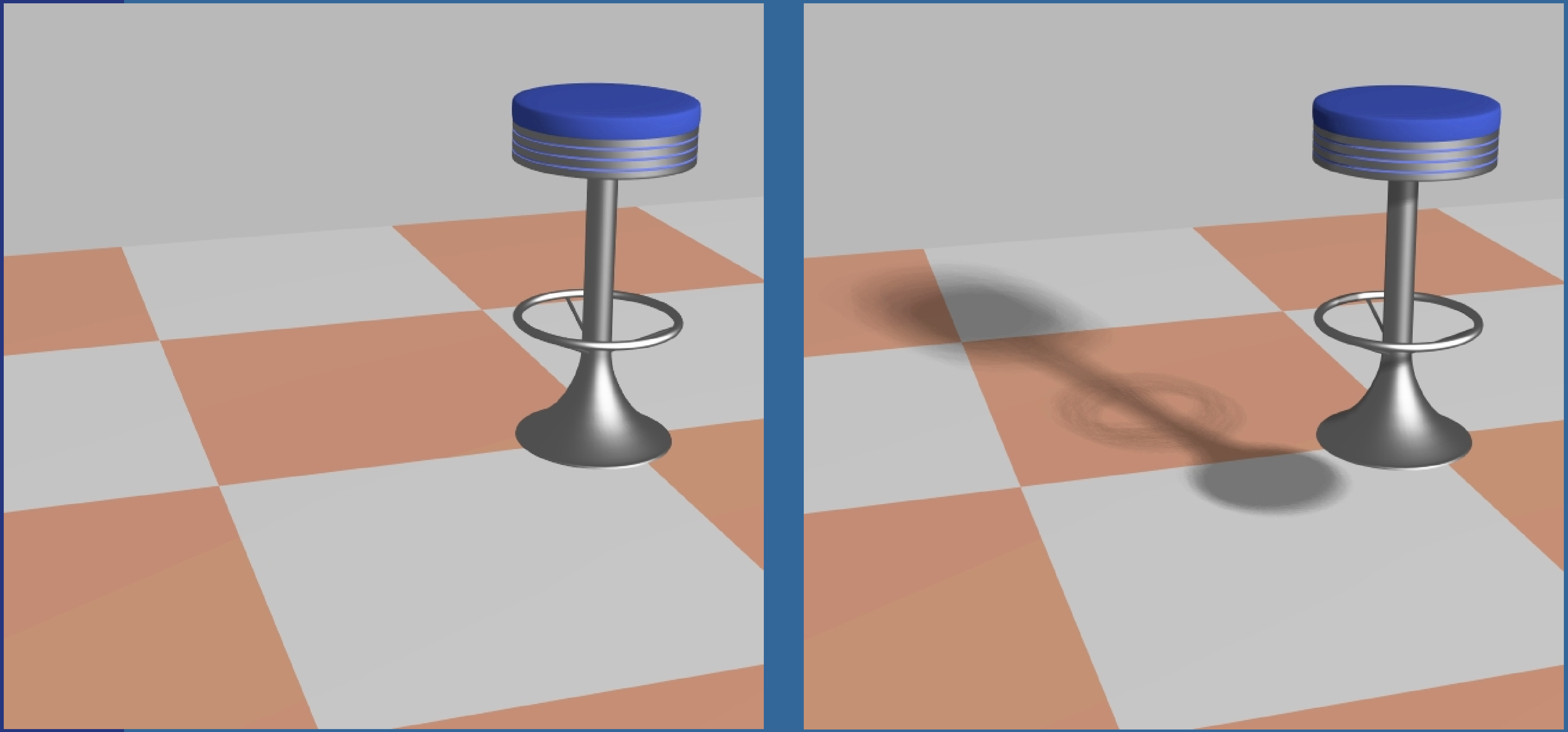


Blade of Darkness

Image courtesy of Codemasters & Rebel Act

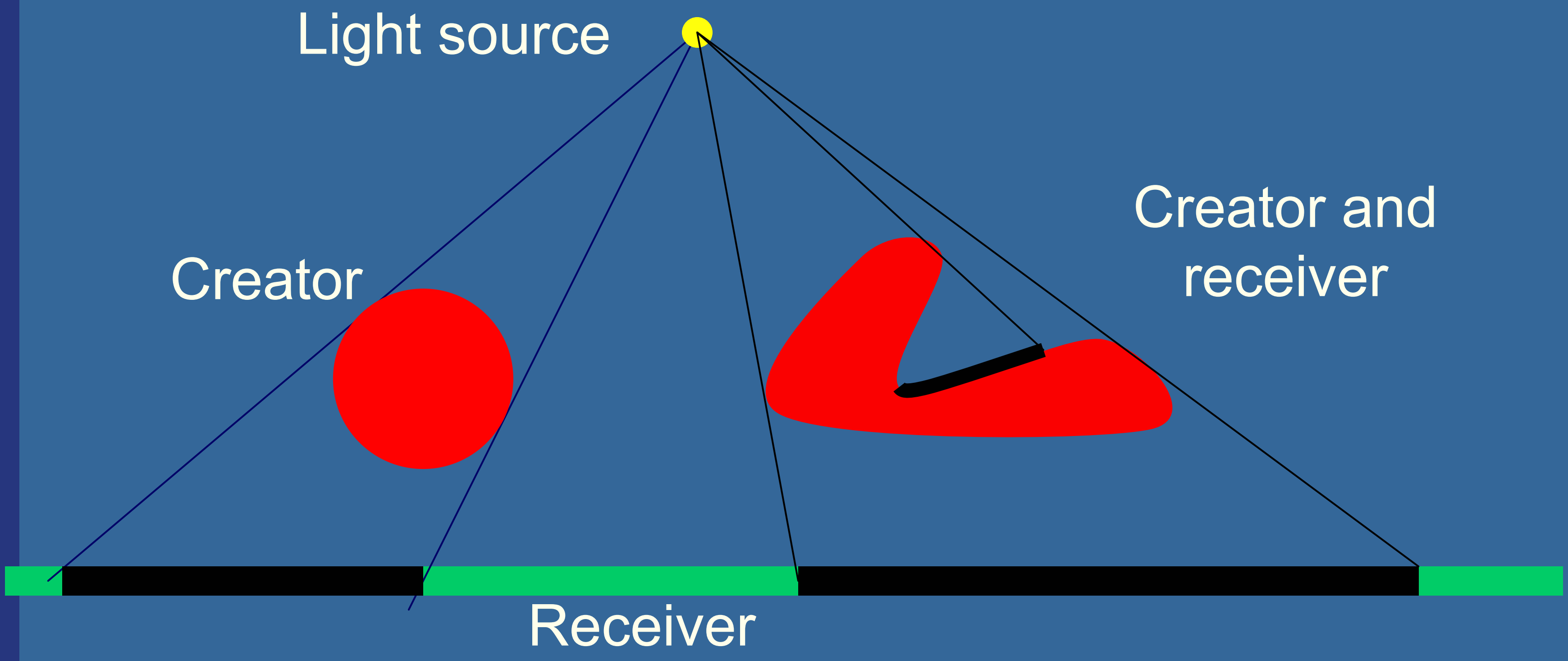
Why shadows?

- More clues about spatial relationships
- Orientation & gameplay



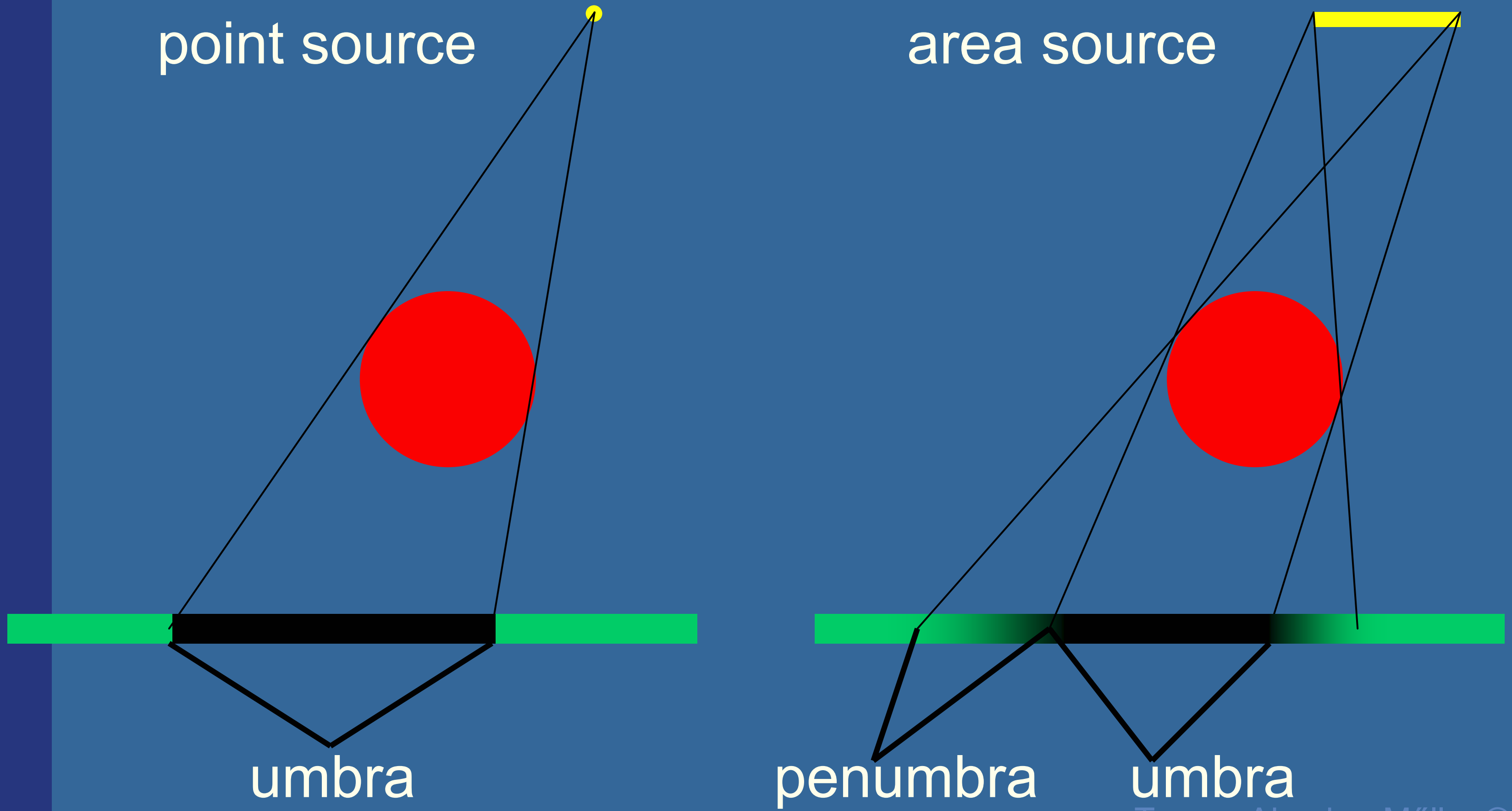
Definitions

- Light sources
- Shadow creators and receivers

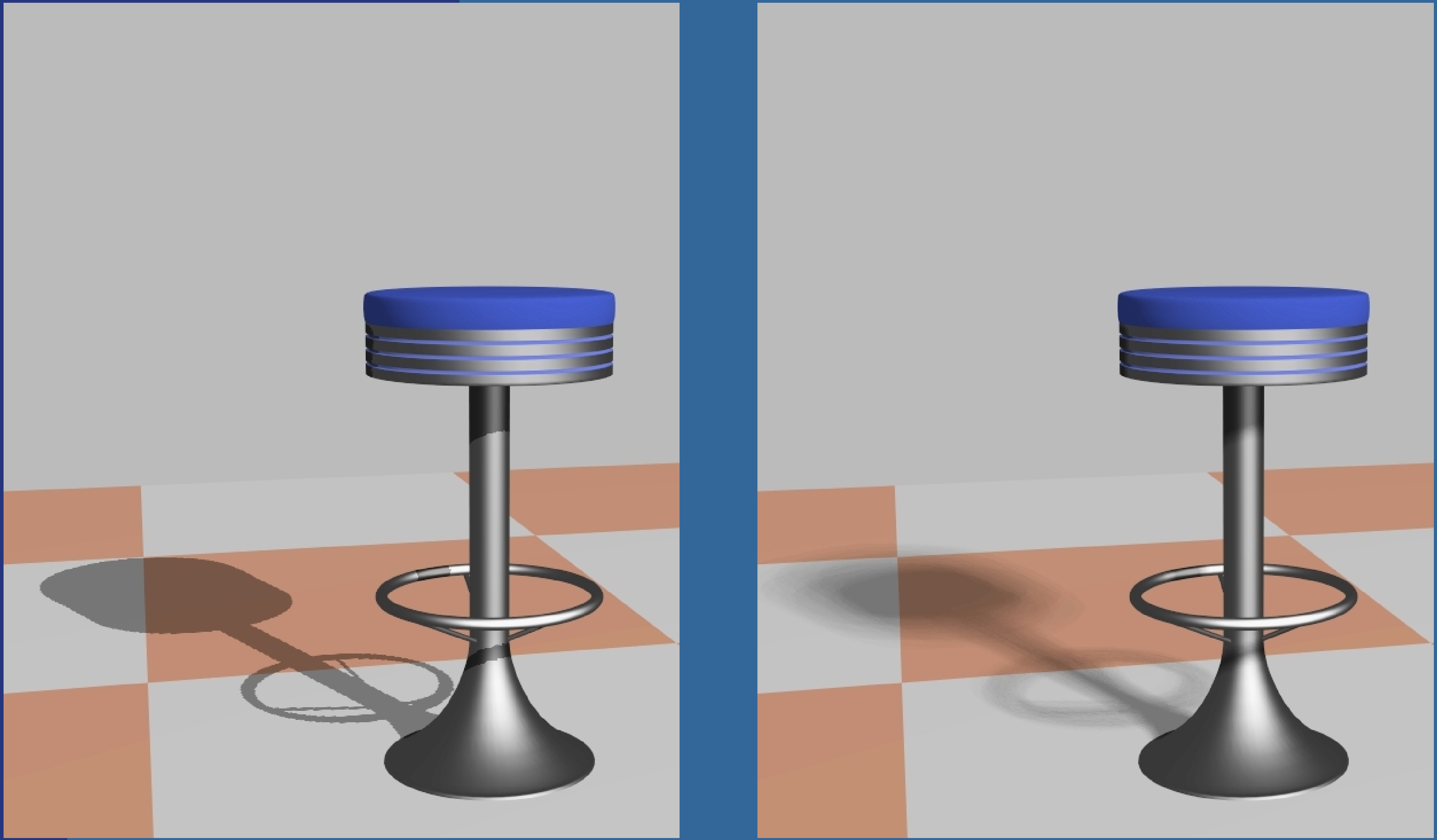


Definitions

- Light source types



Example: hard vs soft shadows

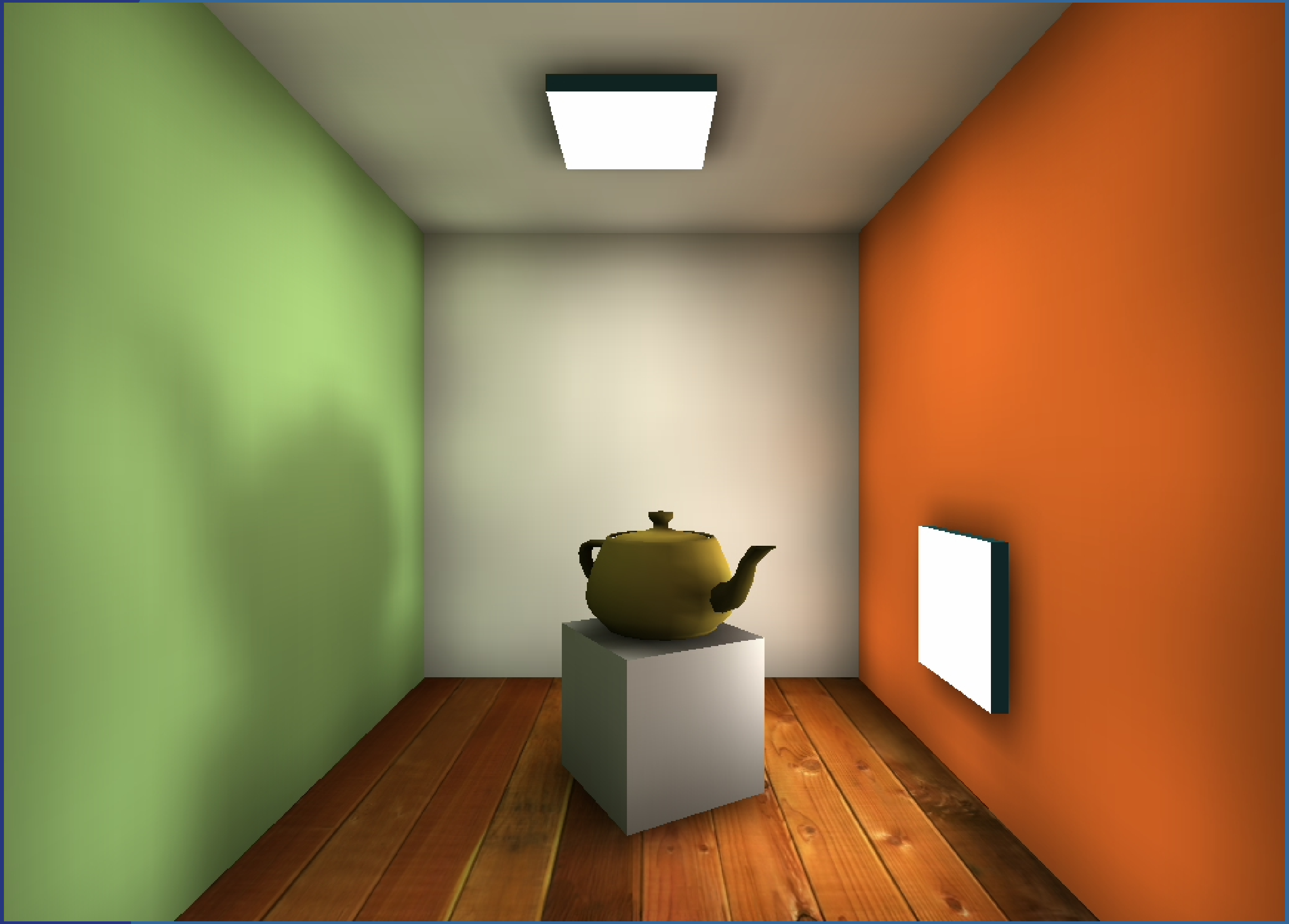


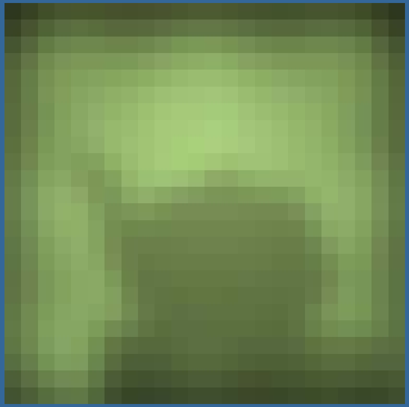
Ways of thinking about shadows

- As separate objects (like Peter Pan's shadow)
- As volumes of space that are dark
- As places not seen from a light source looking at the scene

- Note that we already "have shadows" for objects facing away from light

Store precomputed shadows in textures

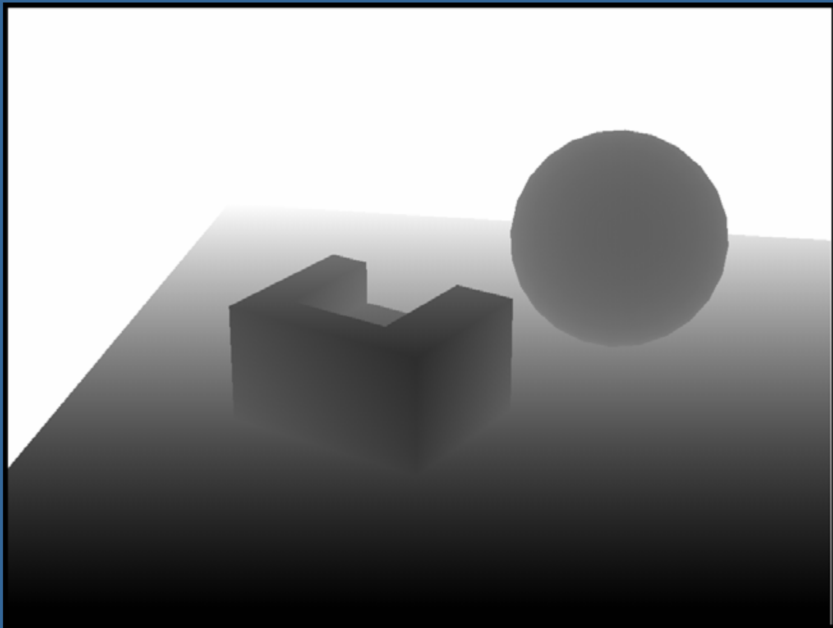




Images courtesy of Kasper Høy Nielsen.

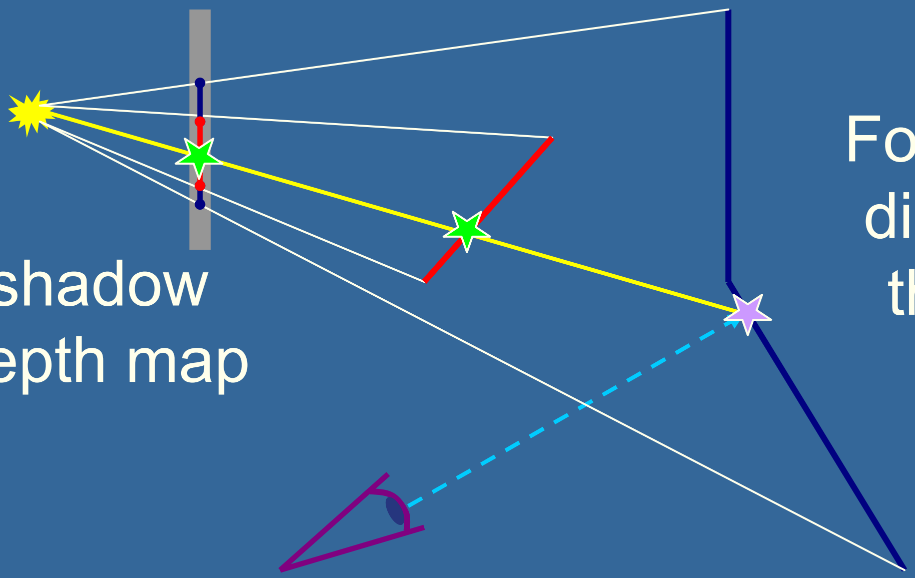
Two major algorithms that can render shadows onto arbitrary geometry

- Shadow mapping and shadow volumes
- Works in real time…
- Shadow mapping is used in Pixar’s rendering software
- Render from light’s view (white is far and black is near)



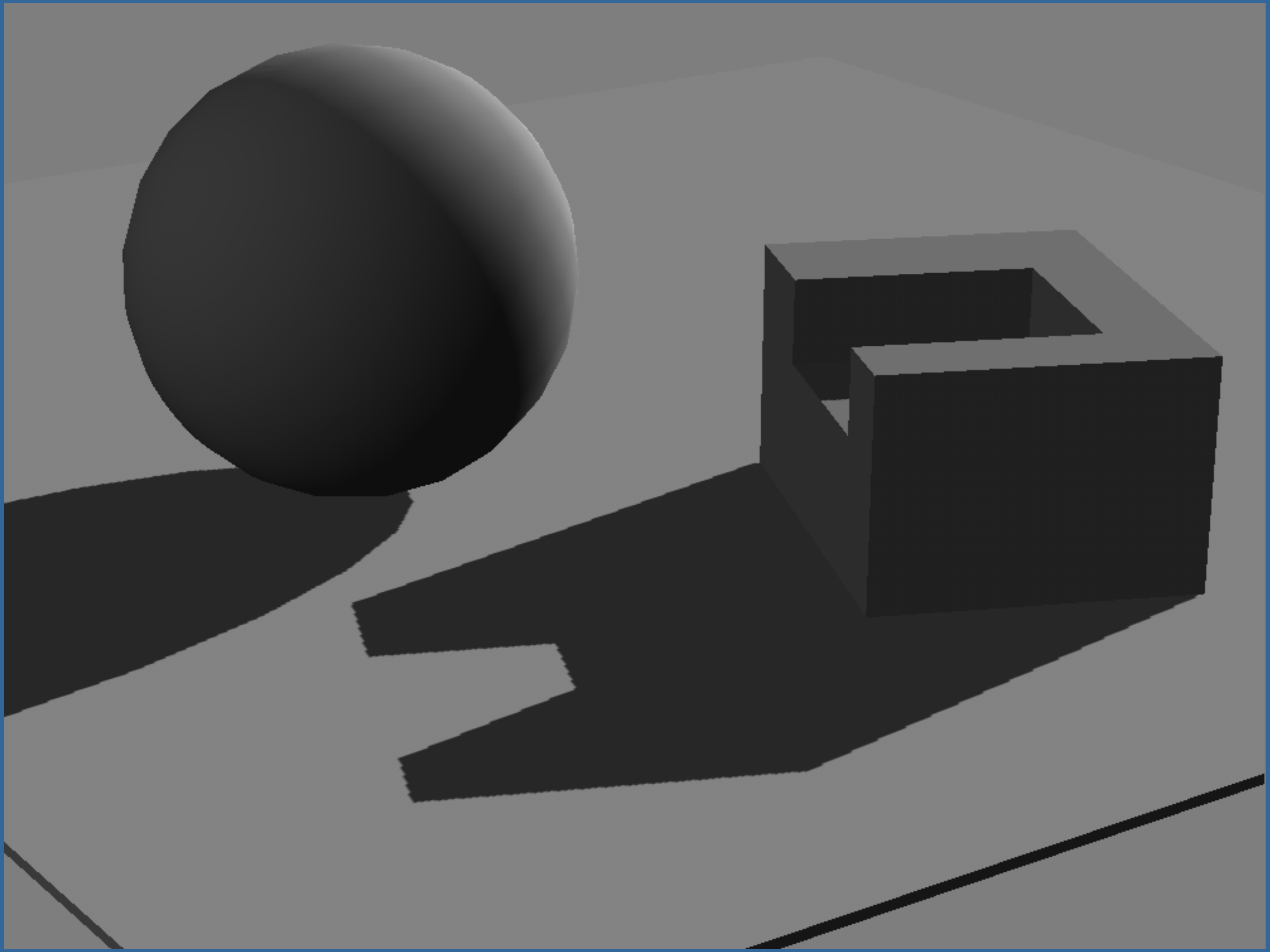
Using the Shadow Map

- When scene is viewed, check viewed location in light's shadow buffer
  - If point's depth is (epsilon) greater than shadow depth, object is in shadow.



For each pixel, compare distance to light ★ with the depth ★ stored in the shadow map

The Result



Shadow mapping problems (1)

- Low resolution of shadow map
  - Gives jagged shadow edges
  - Lots of research, and improvements exist

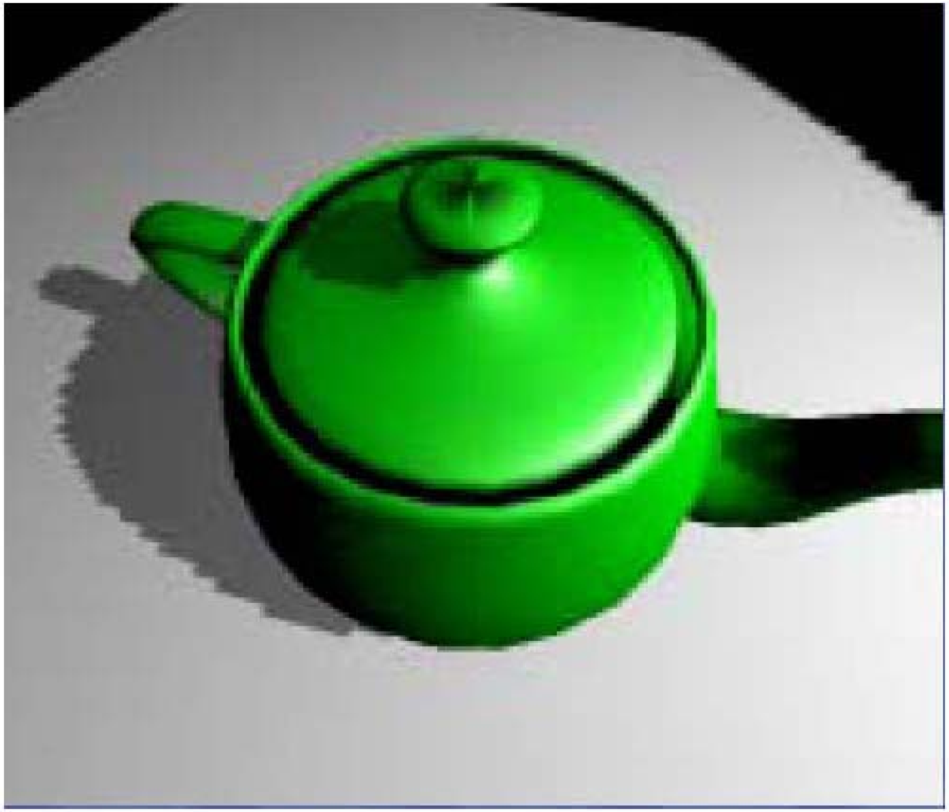
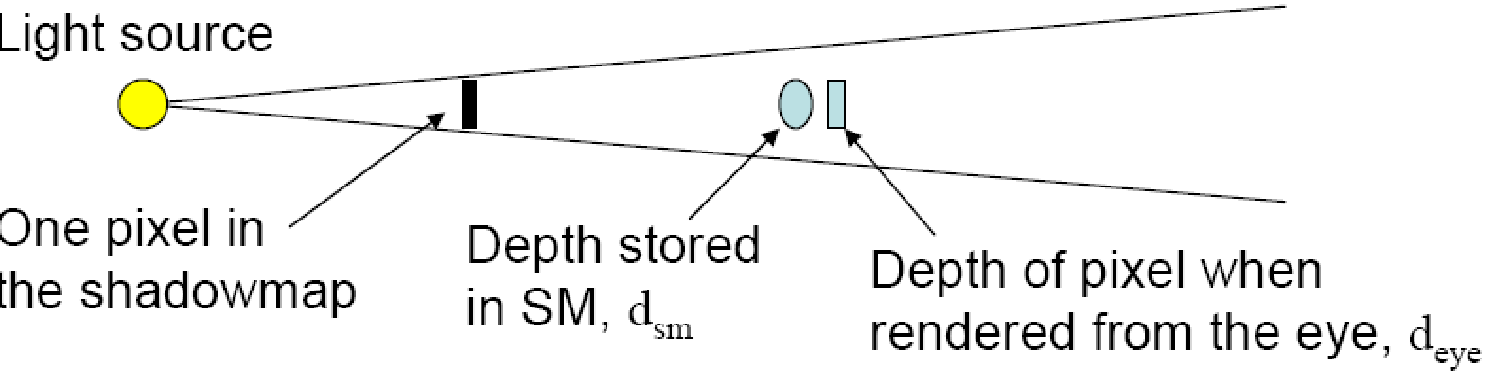


Image courtesy Marc Stamminger

Shadow mapping problems (2)

- Choosing bias (epsilon) is not trivial!



Light source

One pixel in the shadowmap

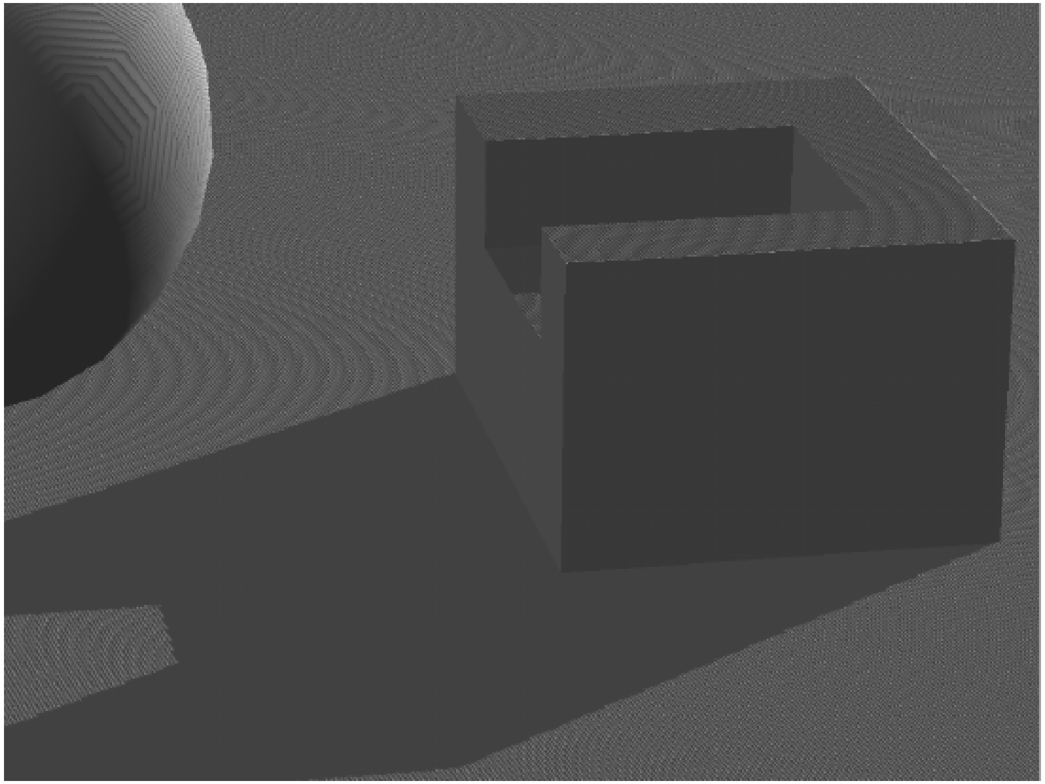
Depth stored in SM, d_{sm}

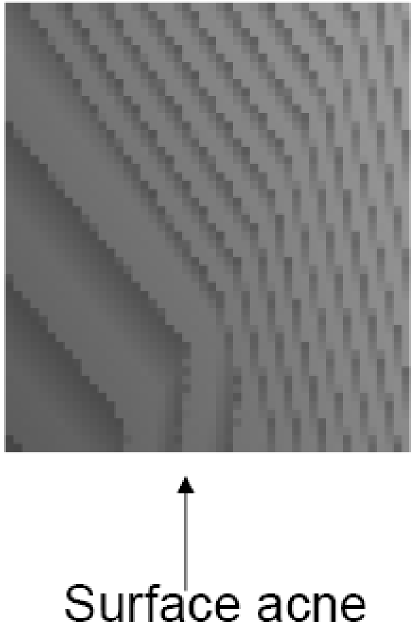
Depth of pixel when rendered from the eye, d_{eye}

- Assume that the ellipse and rectangle is the same surface
  - We do not want incorrect self-shadowing
- Solution: add bias
  - $d_{sm} + \text{bias} < d_{eye}$ → shadow

Too low bias

- You need to make sure the surface seen by the light does not shadow itself





Surface acne

Too high bias

- Too much bias and the shadow “floats”.

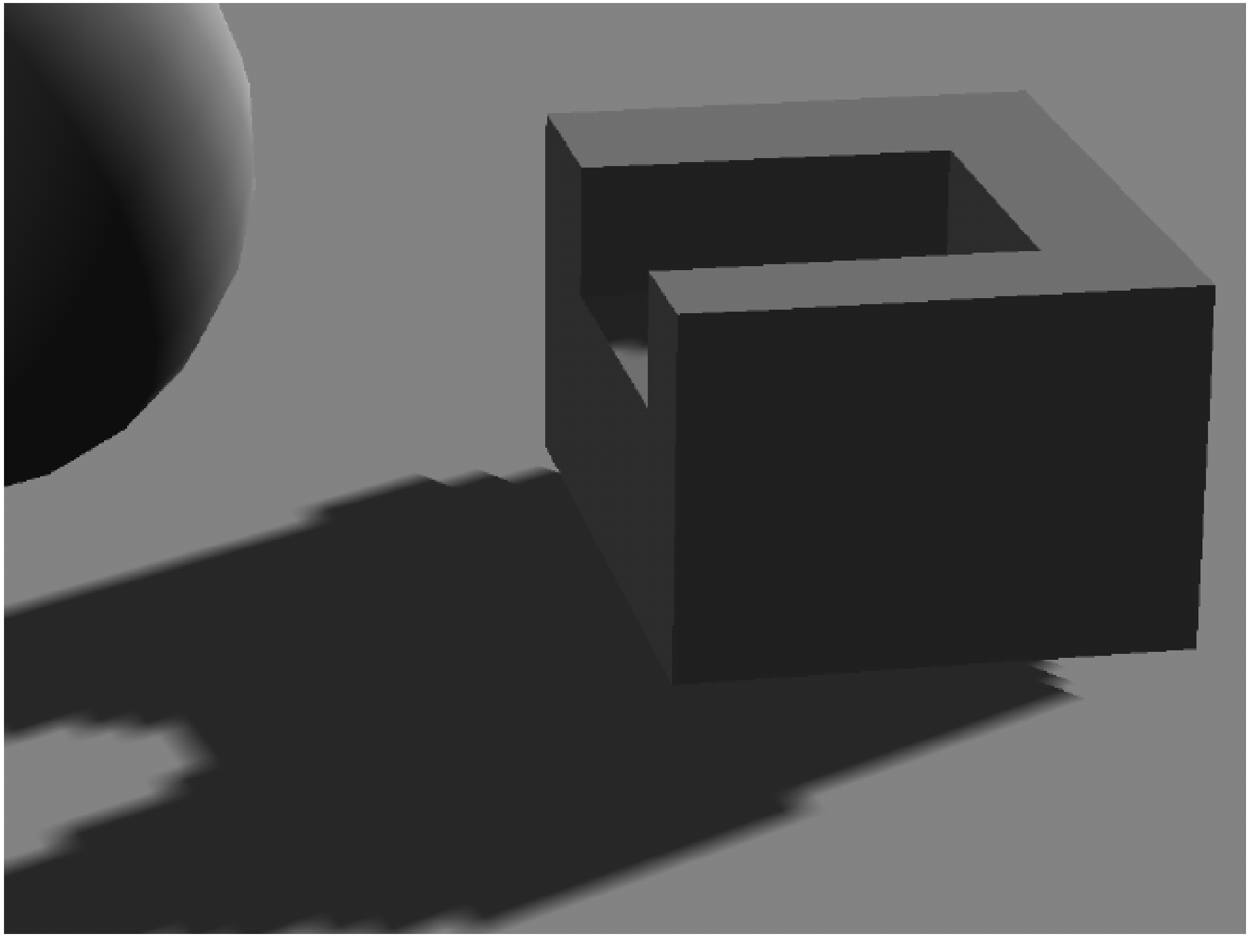
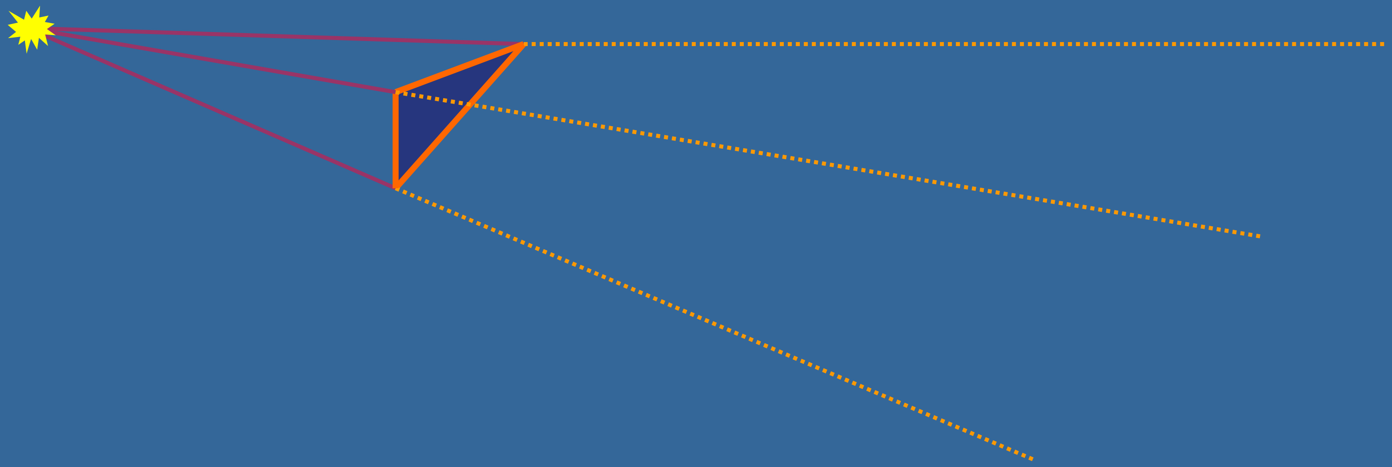


Image also shows jaggedness of shadow boundary

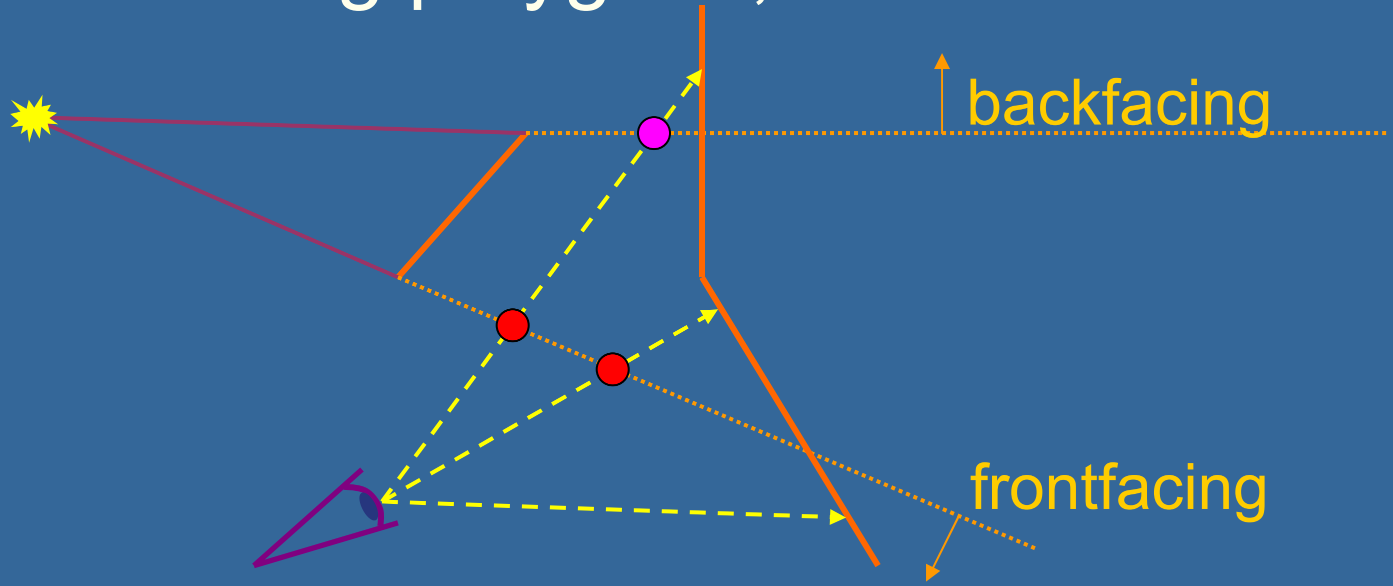
Shadow volumes

- Shadow volume concept
- Create volumes of space in shadow from each polygon in light.
- Each triangle creates 3 projecting quads



Using the Volume

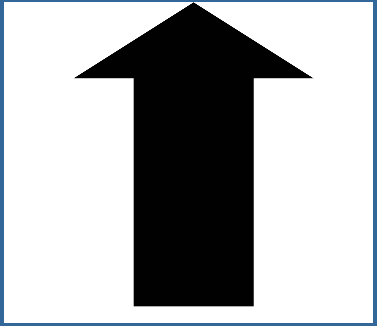
- To test a point, count the number of polygons between it and the eye.
- If we look through more frontfacing than backfacing polygons, then in shadow.



Shadow volume algorithm uses stencil buffer

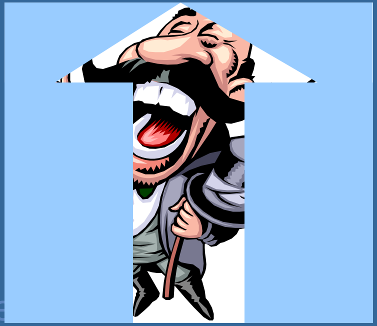
- Stencil what?
- Is just another buffer (often 8 bits per pixel)
- When rendering to it, we can add, subtract, etc
- Then, the resulting image can be used to mask off subsequent rendering

Stencil Buffer Mask



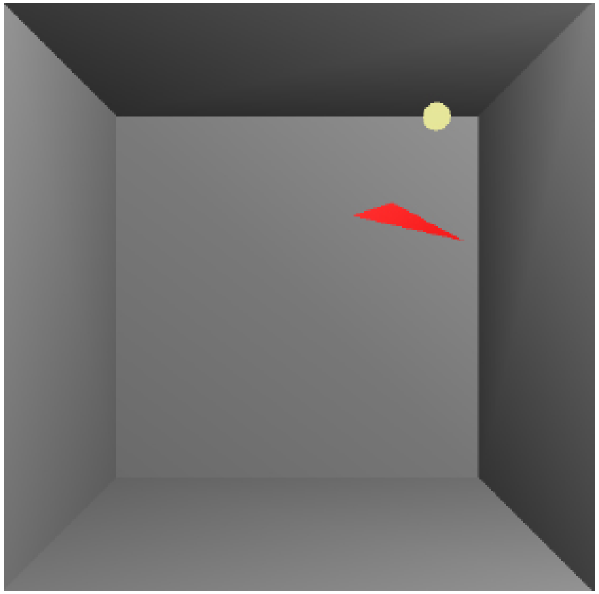
Rendered image

→ result

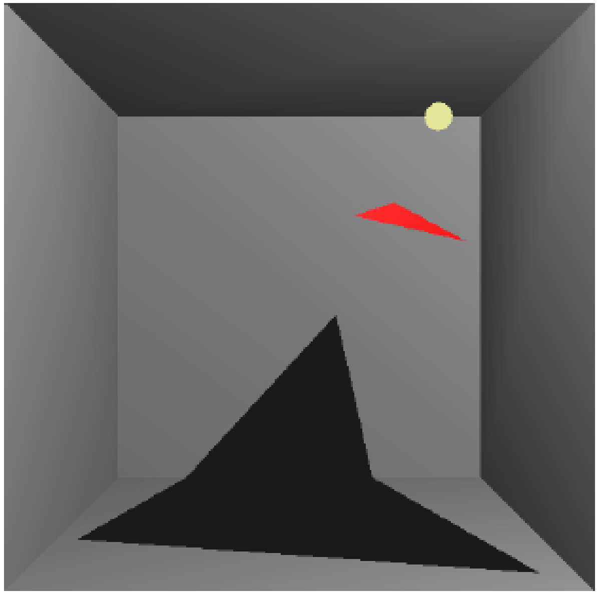


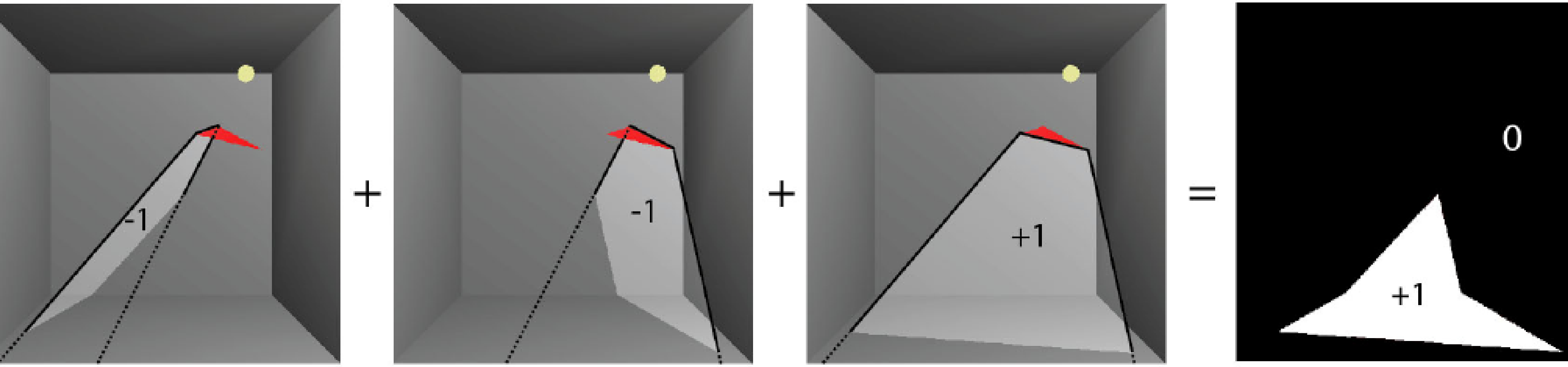
Z-pass by example: how the stencil buffer is used

What we have…



What we wnat…



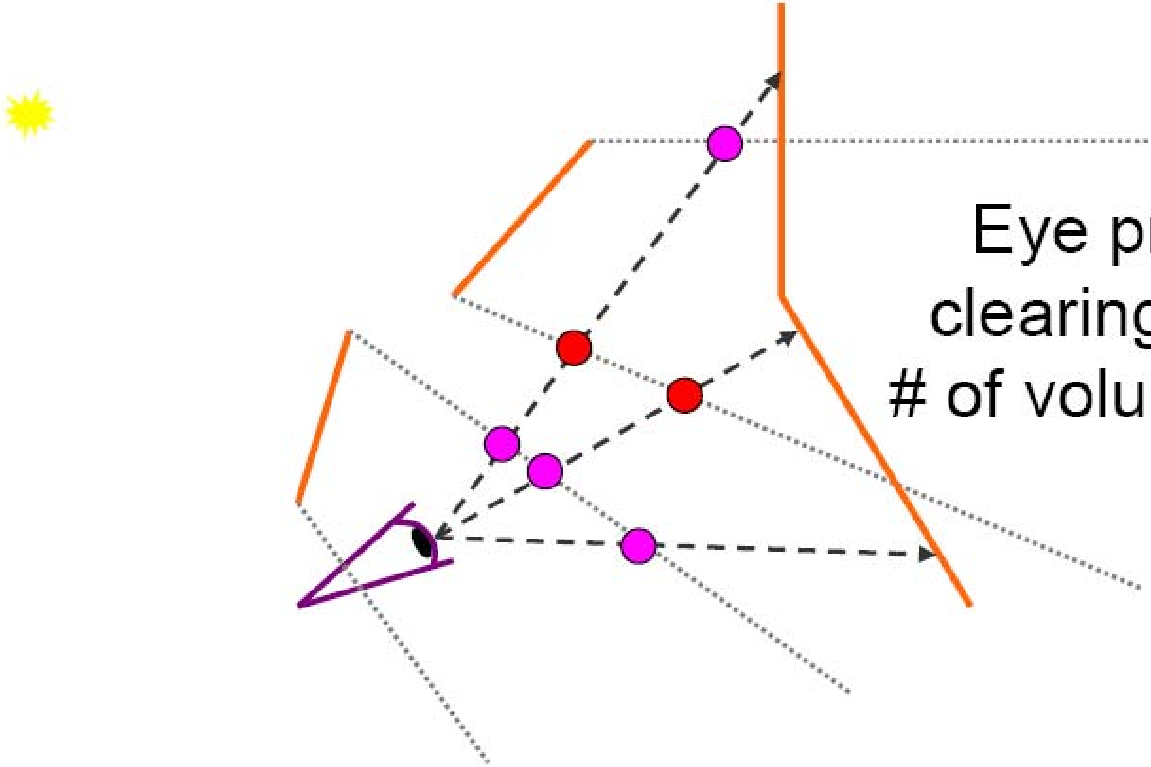


How to implement shadow volumes with stencil buffer (Z-pass)

- A four pass process [Heidmann91]:
  - **1st Pass**: render the scene with just ambient lighting.
  - Turn off updating Z-buffer and writing to color buffer (i.e. Z-compare, draw to stencil only).
  - **2nd pass**: render front facing shadow volume polygons to stencil buffer, incrementing count.
  - **3rd pass**: render backfacing shadow volume polygons to stencil, decrementing.
  - **4th pass**: render diffuse and specular where stencil buffer is 0.

Eye Location Problem

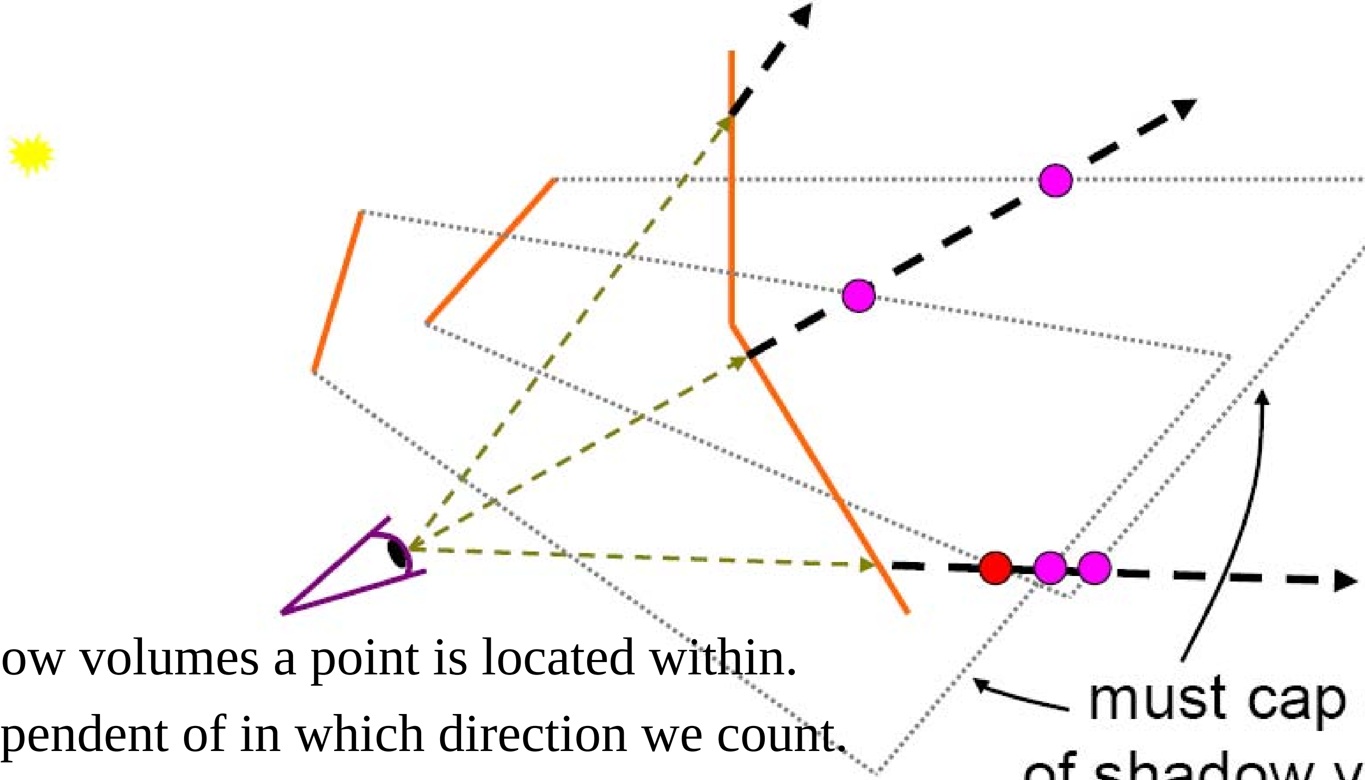
- If the eye location is inside one or more shadow volumes, count is wrong.



Eye problem fixed by clearing stencil buffer to # of volumes eye is inside.

Solution: Count Beyond Surface "Z-fail-algorithm"

- Render to stencil only when shadow volume Z >= stored Z!



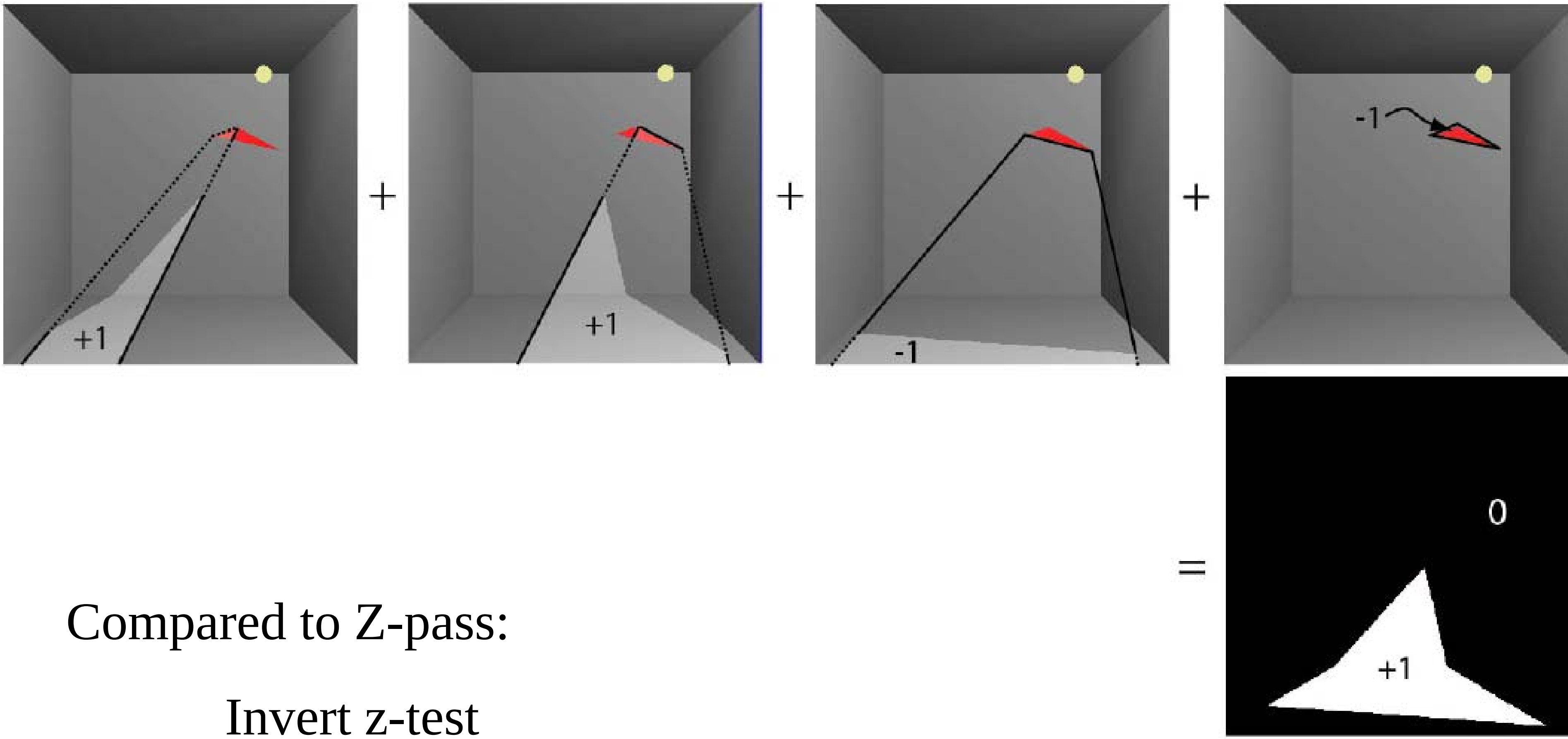
a=#shadow volumes a point is located within.

a is independent of in which direction we count.

Choose point at infinity. That point is always outside shadow, due to the caping of the shadow volumes

must cap ends of shadow volumes (or project to infinity, w=0 for vertices)

Z-fail by example



Compared to Z-pass:

Invert z-test

Invert stencil inc/dec

I.e., count to infinity instead of from eye.

Shadow maps vs shadow volumes

Shadow Volumes

- *Good*: Anything can shadow anything, including self-shadowing, and the shadows are **sharp**.
- *Bad*: **3 or 4 passes**, shadow polygons must be generated and rendered → lots of polygons & **fill**,
- Z-Fail: Near-capping polygons cannot be rendered with tristrips .
- Z-Pass: counting problems, ZP+: more complex.

Shadow Maps

- *Good*: Anything to anything, **constant cost** regardless of complexity, map can sometimes be reused.
- *Bad*: Frustum limited. **Jagged shadows** if res too low, **biasing** headaches.

Shadow Volume Example

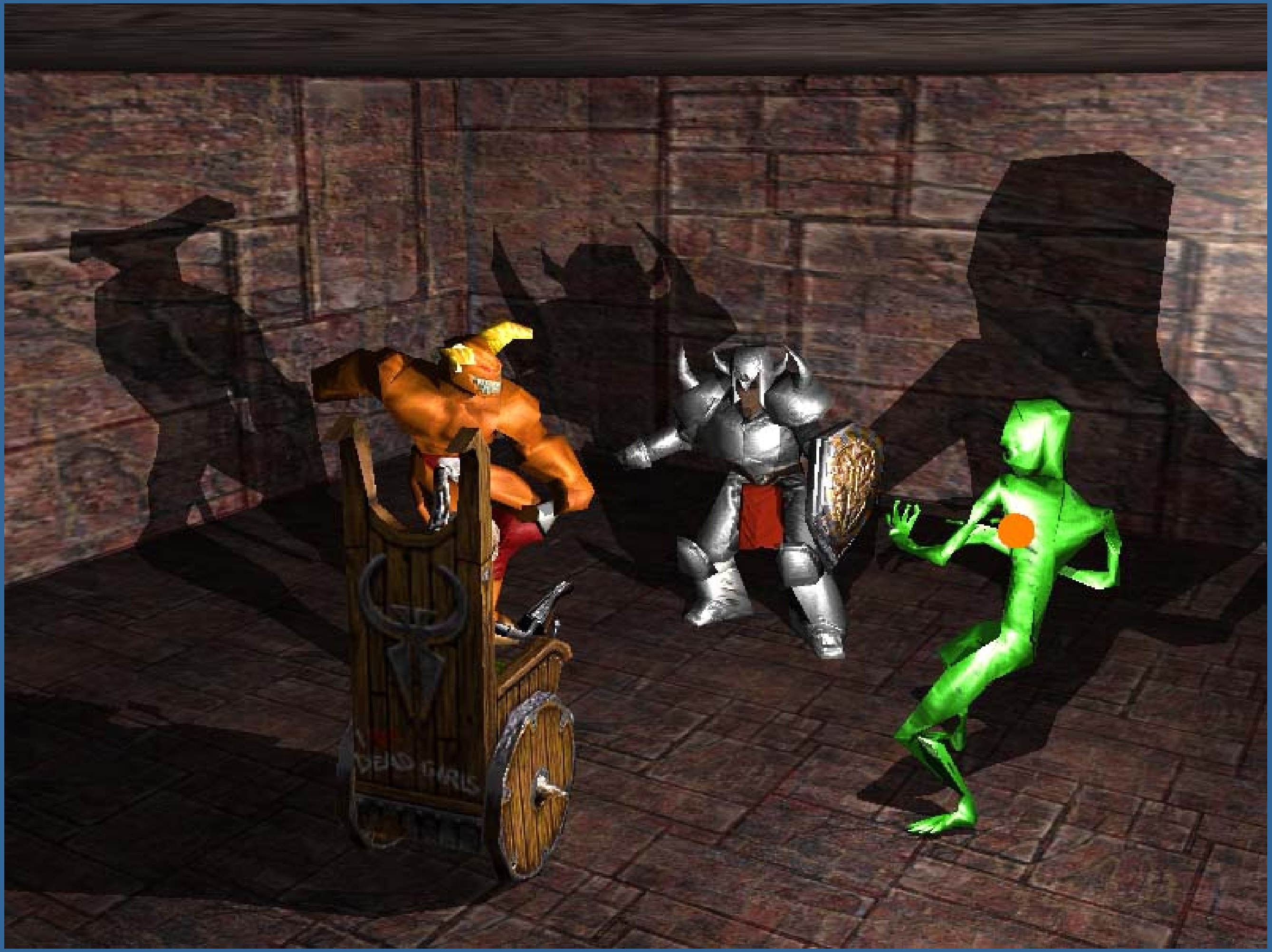
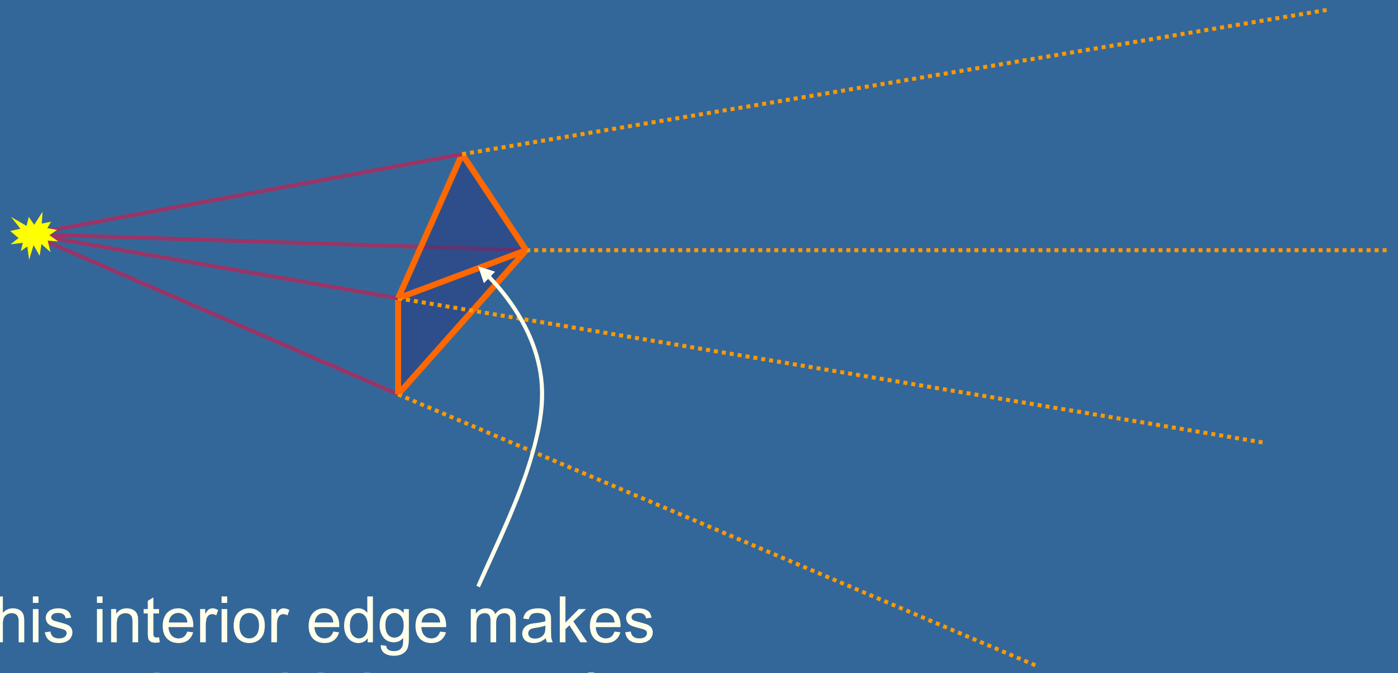


Image courtesy of NVIDIA Inc.

Merging Volumes

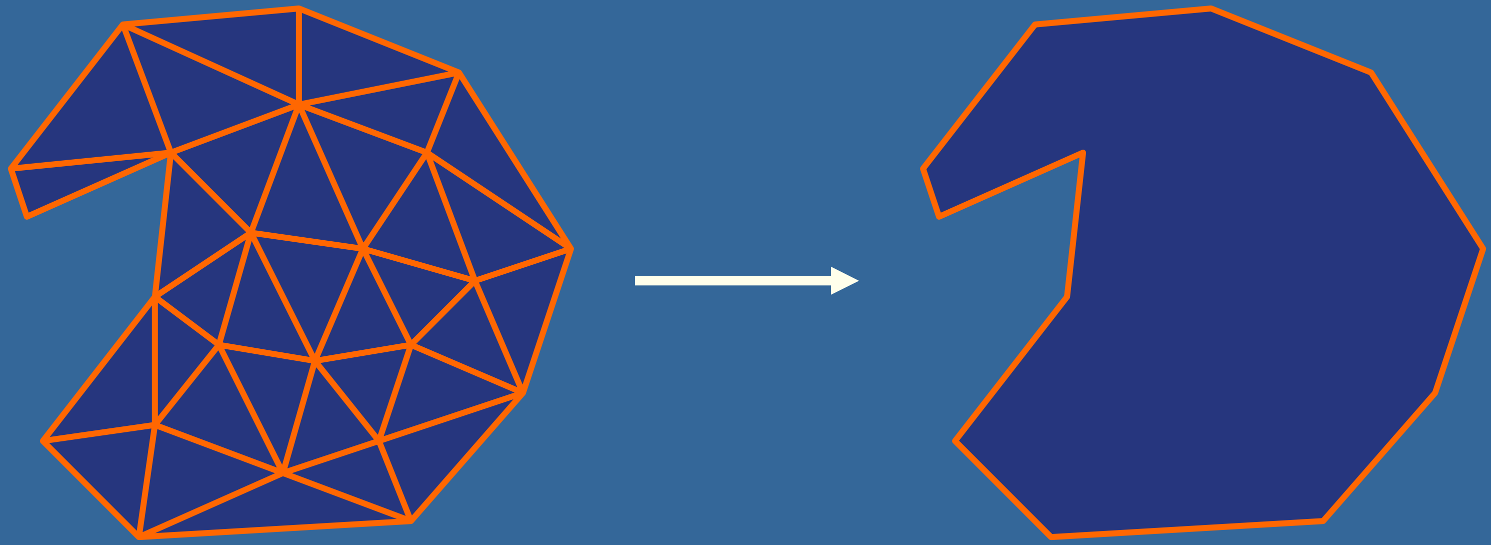
- Edge shared by two polygons facing the light creates front and backfacing quad.



This interior edge makes two quads, which cancel out

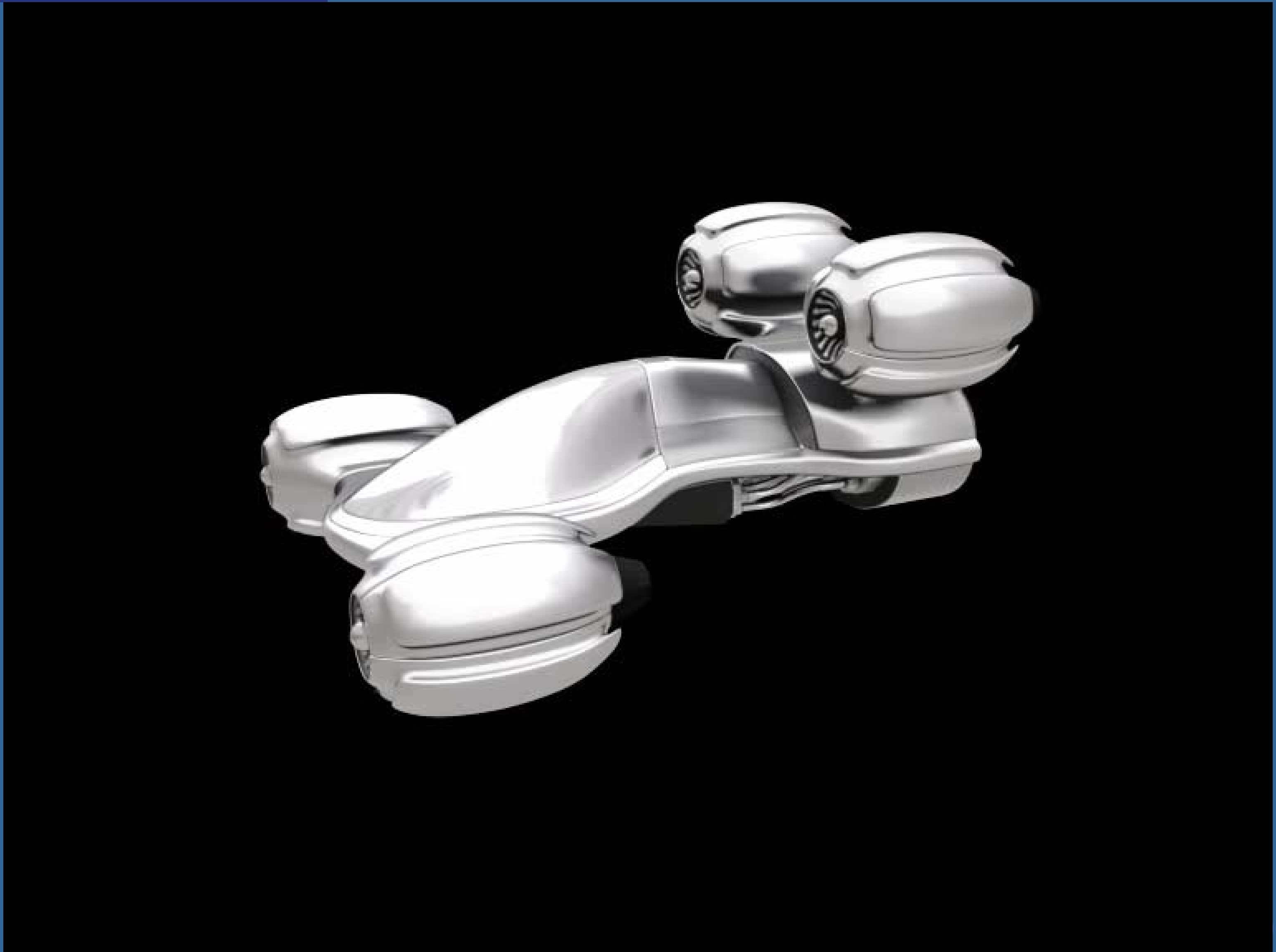
Silhouette Edges

From the light’s view, caster interior edges do not contribute to the shadow volume.



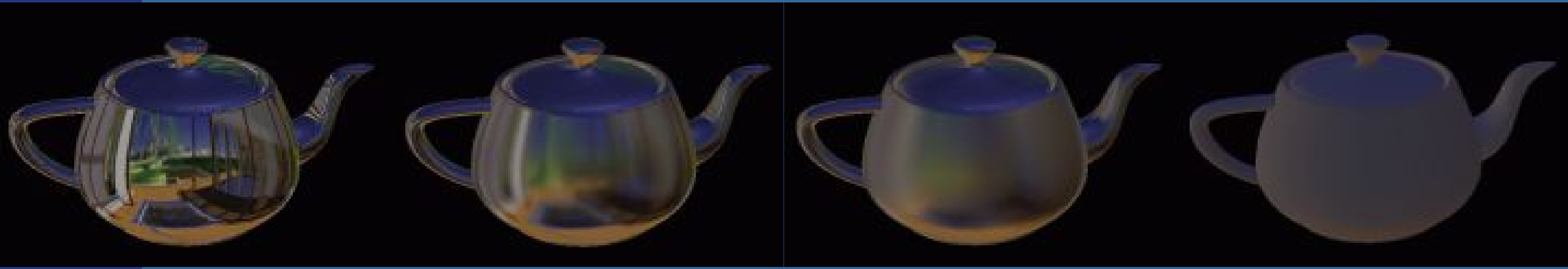
Finding the silhouette edge gets rid of many useless shadow volume polygons.

Reflections



Misc

- Michael Ashikhmin and Abhijeet Ghosh. **Simple blurry reflections with environment maps**. Journal of graphics tools, 7(4):3-8, 2002



```
glTexParameterf(GL_TEXTURE_CUBE_MAP_ARB,
        GL_TEXTURE_MIN_LOD, lambda);
```

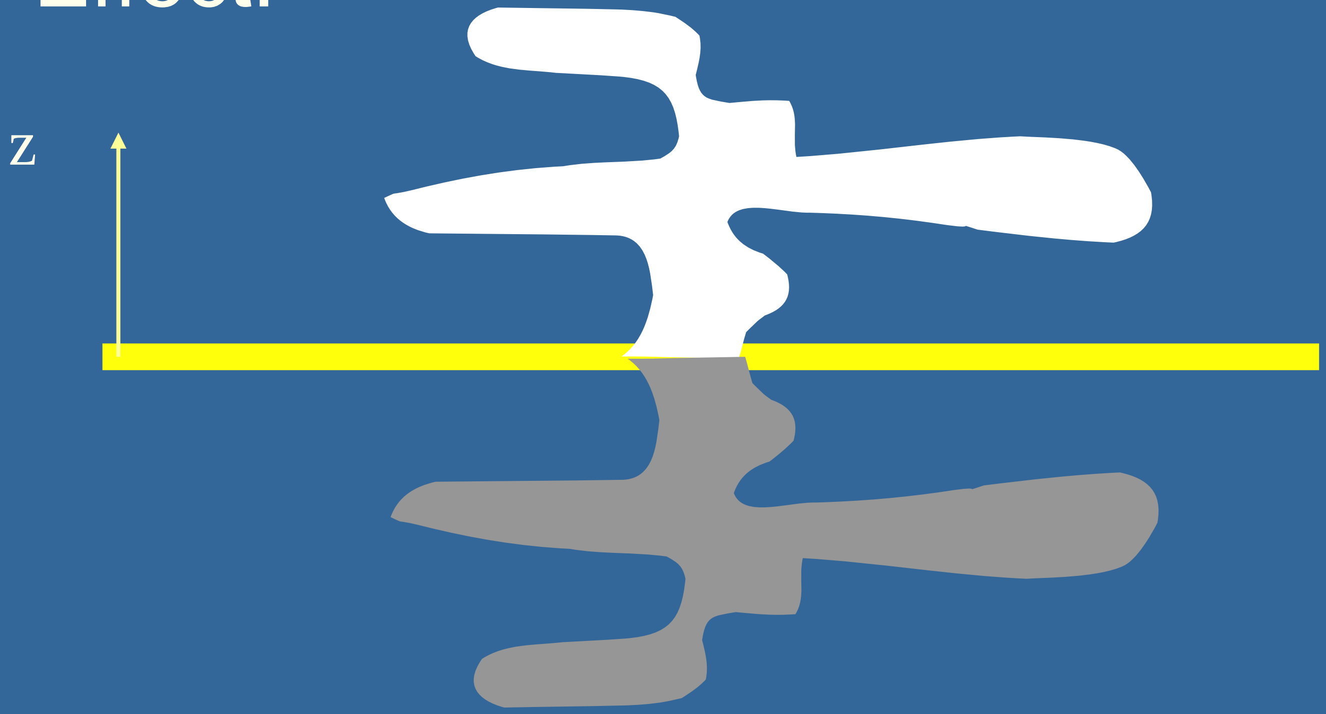
Planar reflections

- We’ve already done reflections in curved surfaces with environment mapping
- Does not work for planar surfaces
- Planar reflections are important, because they too give clues about spatial relationships and increases realism

- Based on law of reflection:
  - Incoming angle is equal to outgoing angle

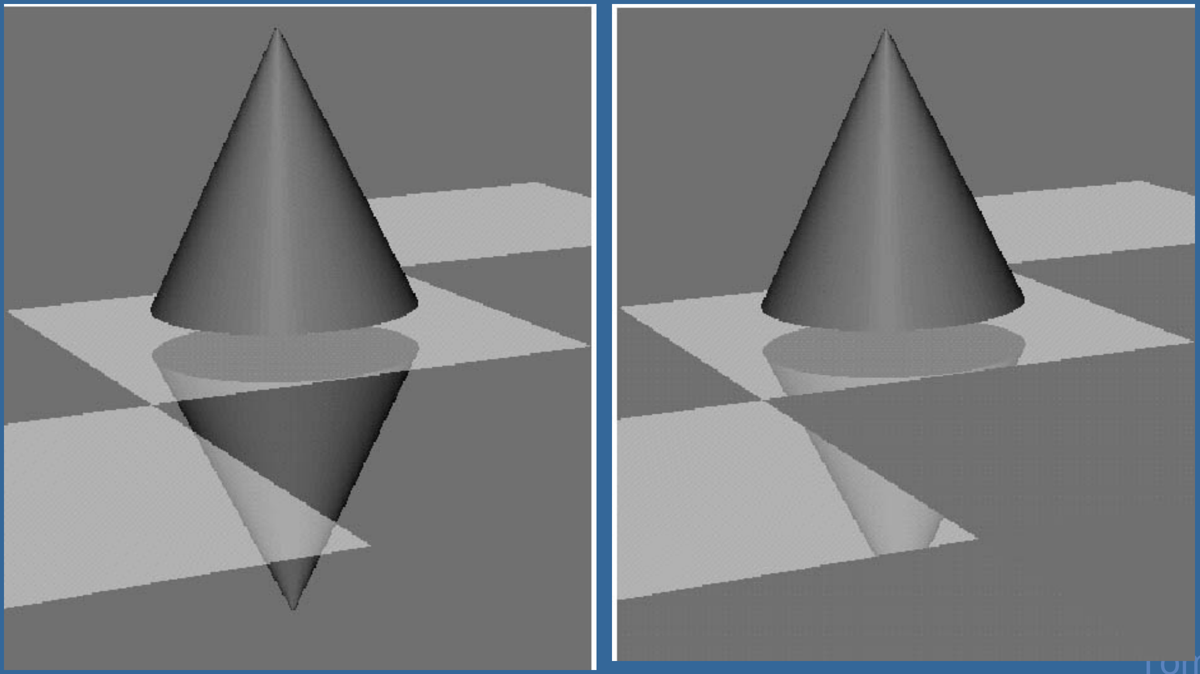
Planar reflections

- Assume plane is z=0
- Then apply **`glScalef(1,1,-1);`**
- Effect:



Planar reflections

- Backfacing becomes front facing!
- Lights should be reflected as well
- Need to clip (using stencil buffer)
- See example on clipping:



Planar reflections

- How should you render?
- 1) the ground plan polygon into the stencil buffer
- 2) the scaled (1,1,-1) model, but mask with stencil buffer
  - Reflect light pos as well
  - Use front face culling
- 3) the ground plane (semi-transparent)
- 4) the unscaled model

Final slide
Another example

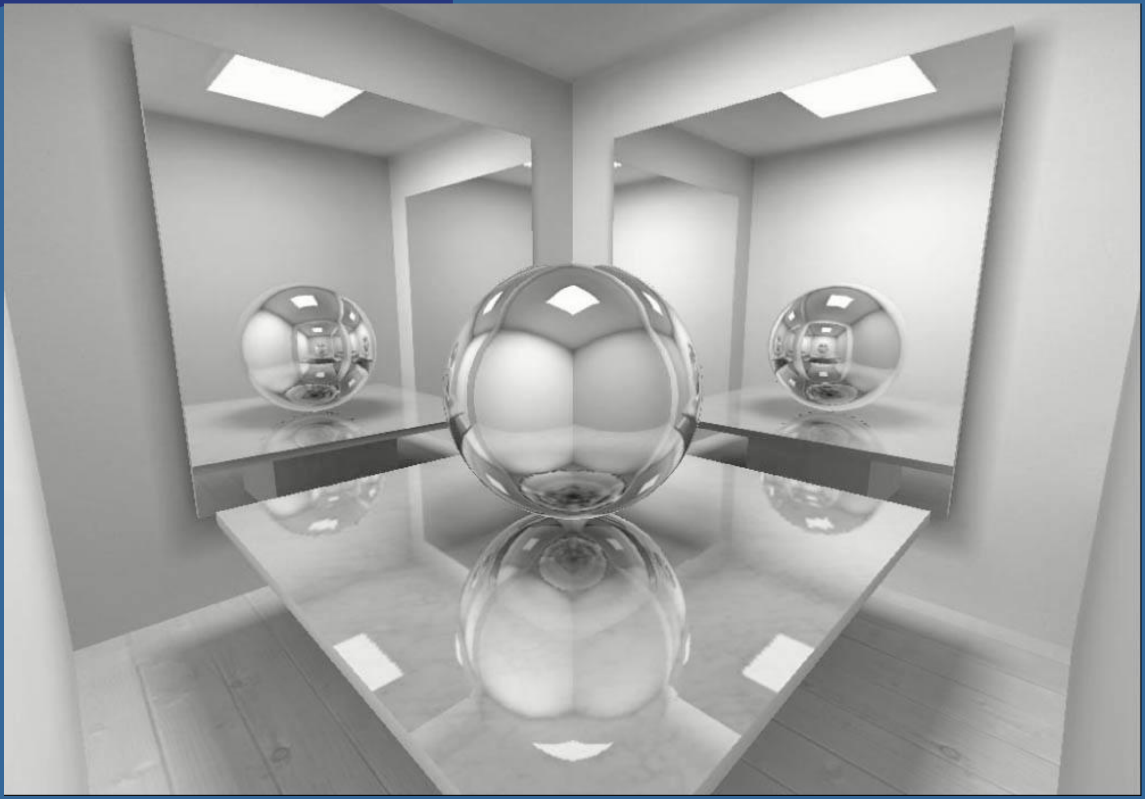
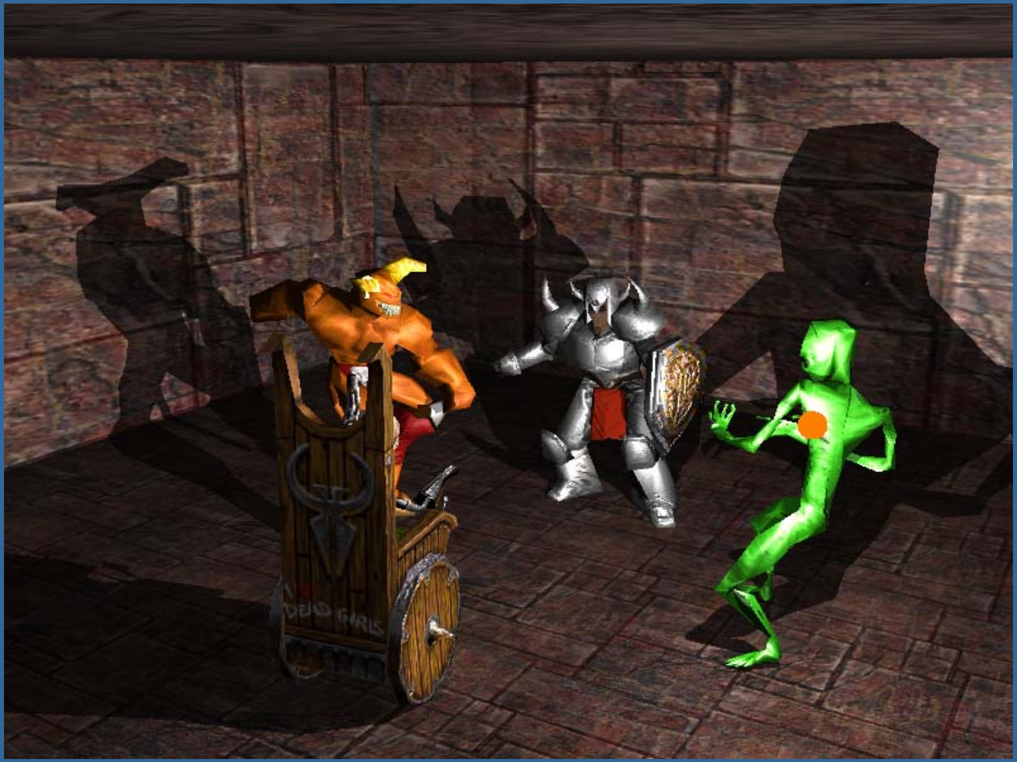


Image courtesy of Kasper Hoey Nielsen

- Instead of the scale-trick, you can reflect the camera position and direction in the plane
- Then render reflection image from there

If we got time

- Stencil shadow demo



"C:\Kurs 2004\teaching\adv_cg 2004\10.2 shadows_reflections\md2shader\md2shader.exe"