Basic Shadow and Reflection Techniques in Real-Time

Shadow Maps and Shadow Volumes

Ulf Assarsson

Why shadows?

- More realism and atmosphere

Image courtesy of BioWare

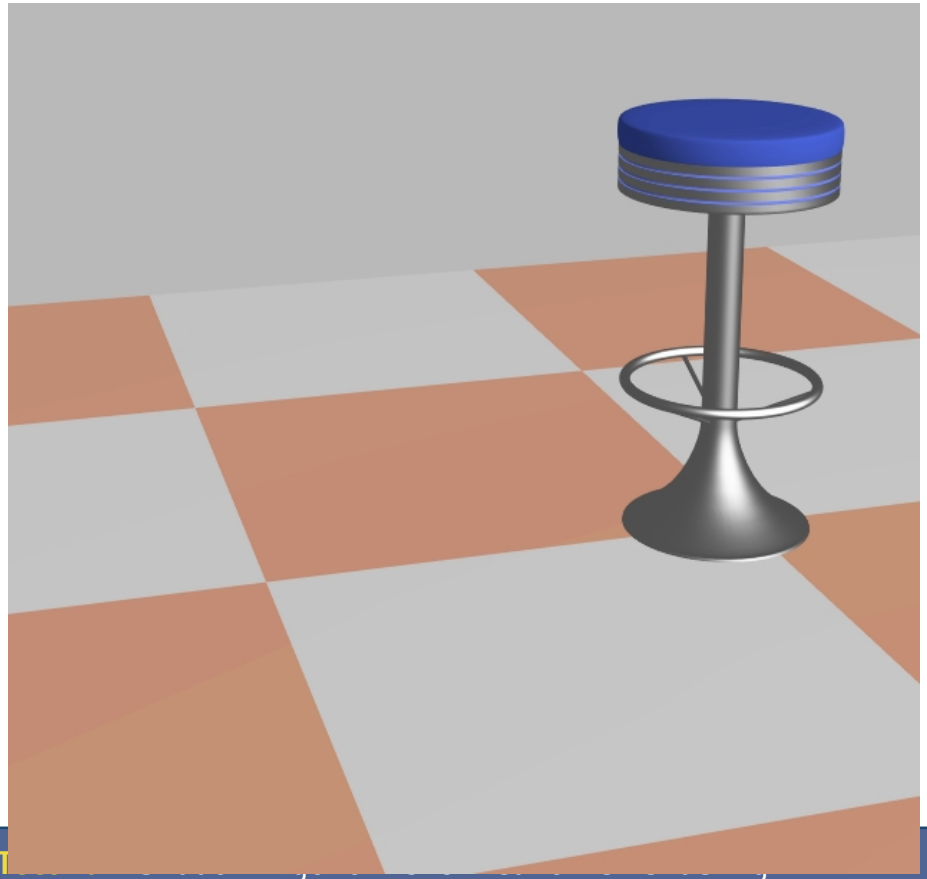
Another example

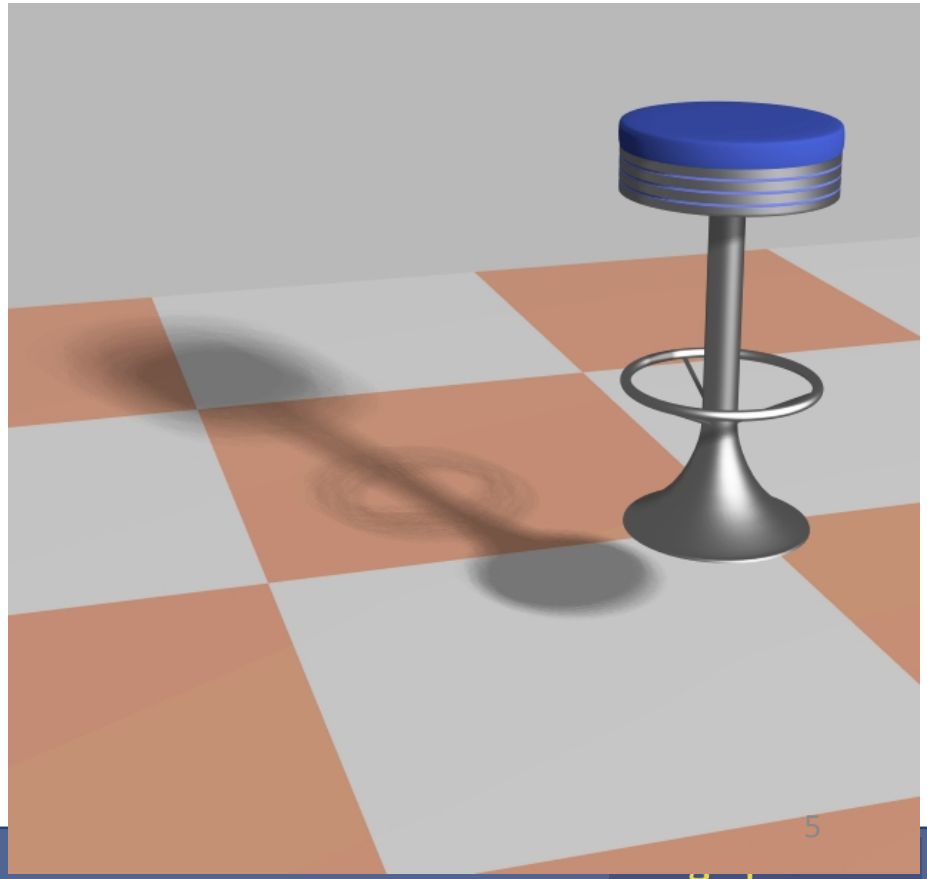
Image courtesy of Codemasters & Rebel Act


pov
is
great!
POV
FLAKES
The Kitchen - Jaime Vives Piqueres - POVCOMP 2004

Why shadows?

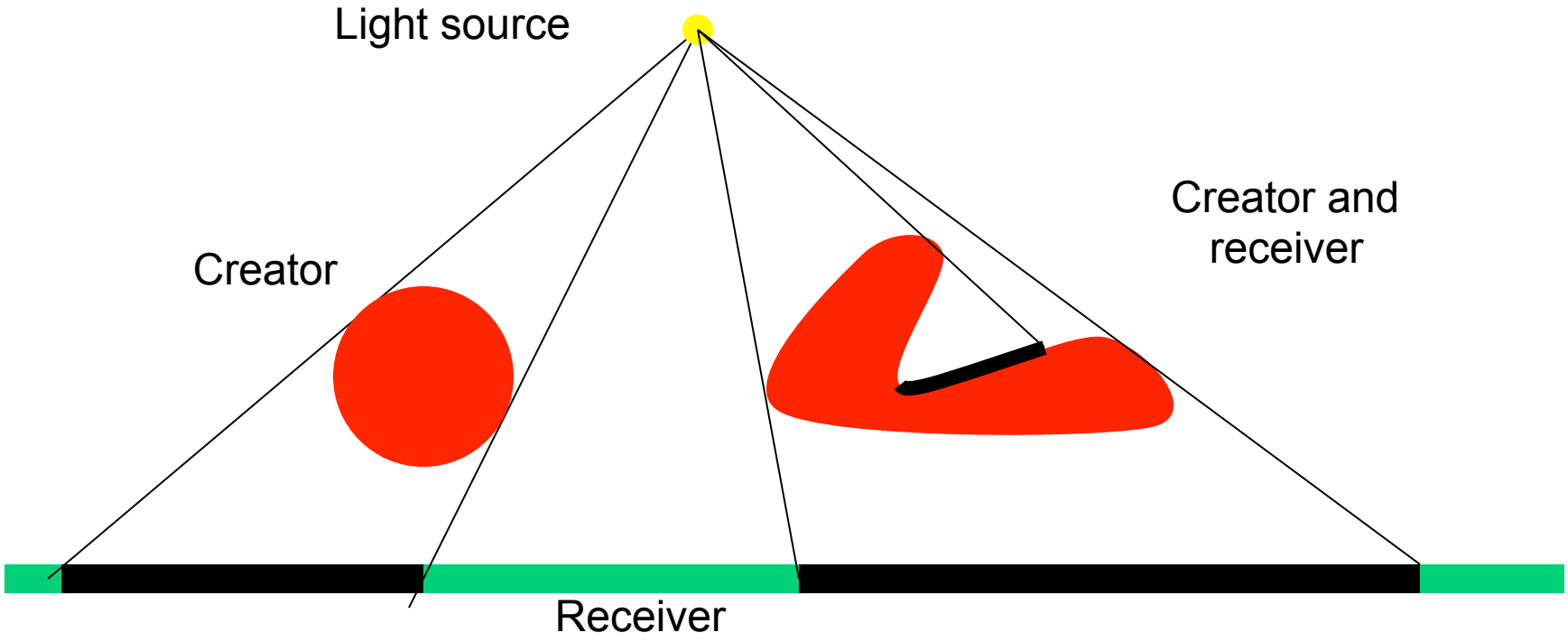
- More clues about spatial relationships
- Orientation & gameplay





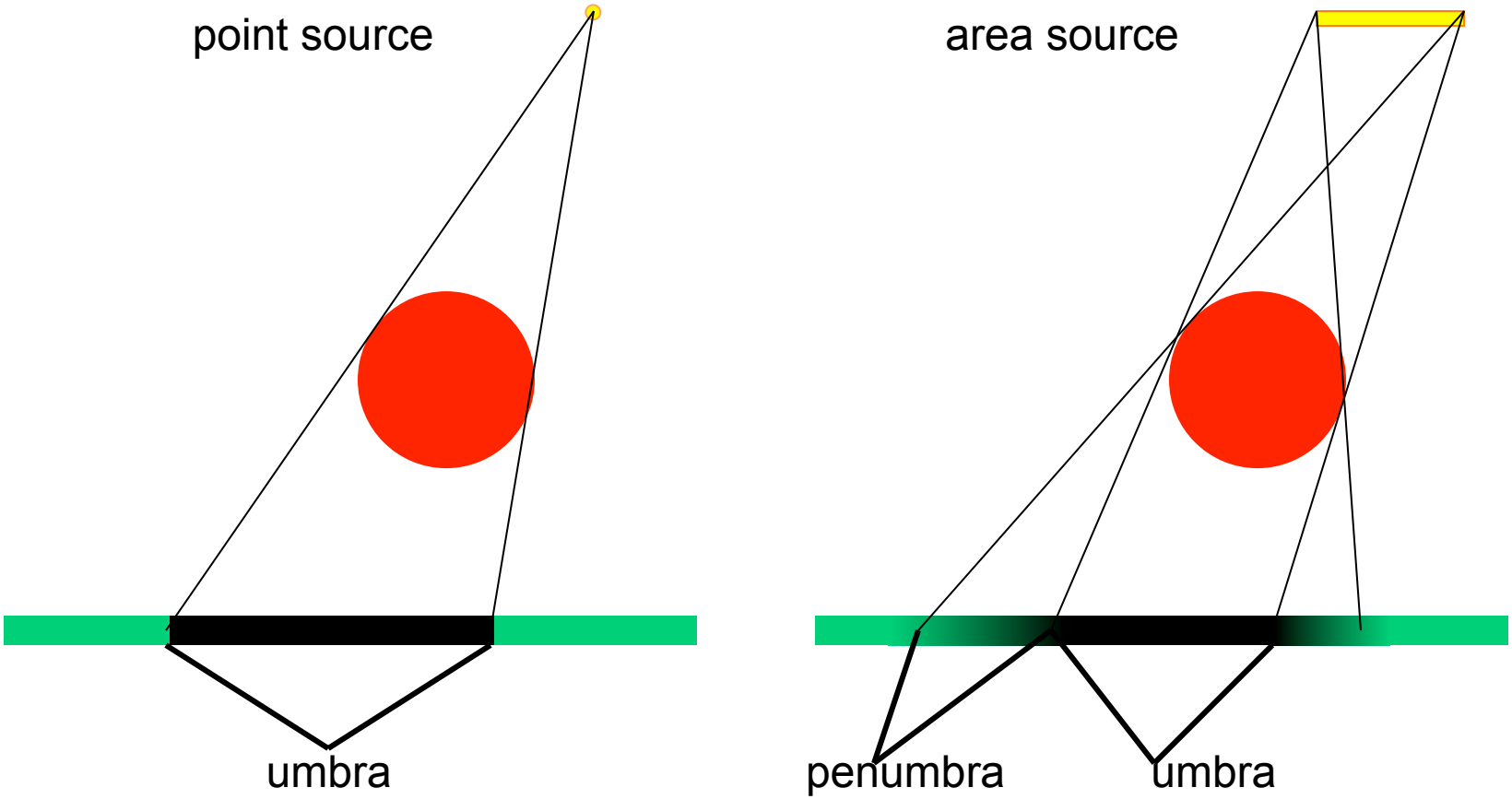
Definitions

- Light sources
- Shadow creators and receivers

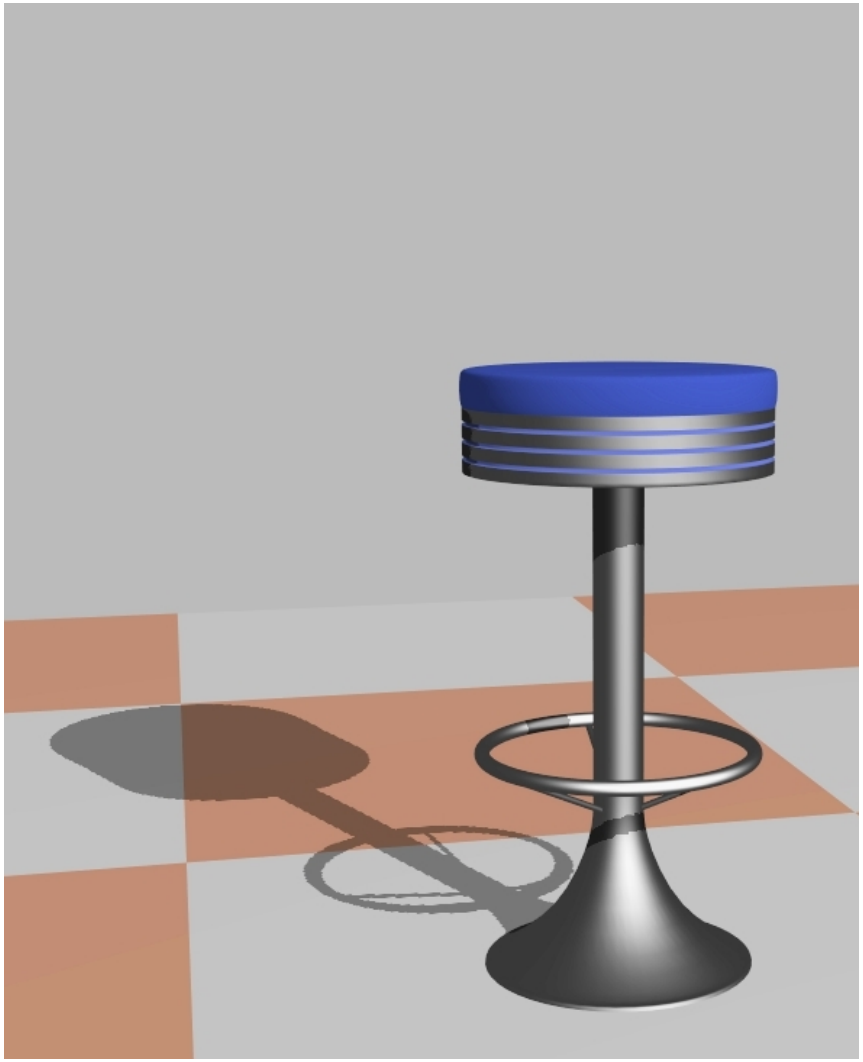


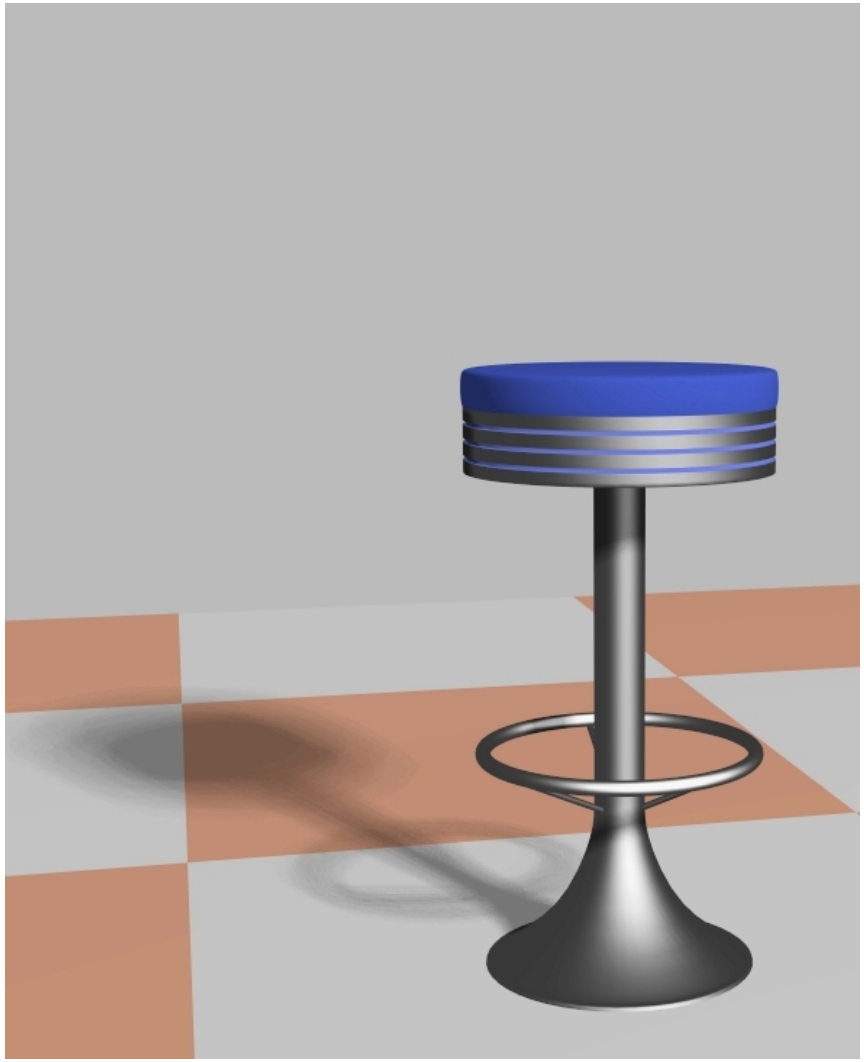
Definitions

- Light source types

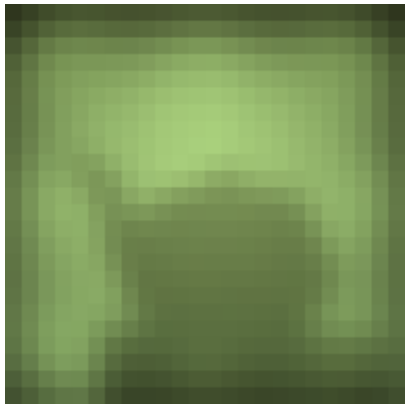


Example: hard vs soft shadows





Store precomputed shadows in textures

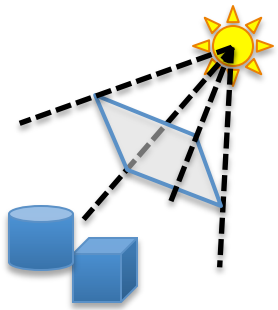


Images courtesy of Kasper Høy Nielsen.

Ways of thinking about shadows

- As separate objects (like Peter Pan's shadow)
- As volumes of space that are dark
  - Shadow Volumes [Franklin Crow 77]
- As places not seen by a light source looking at the scene
  - Shadow Maps [Lance Williams 78]

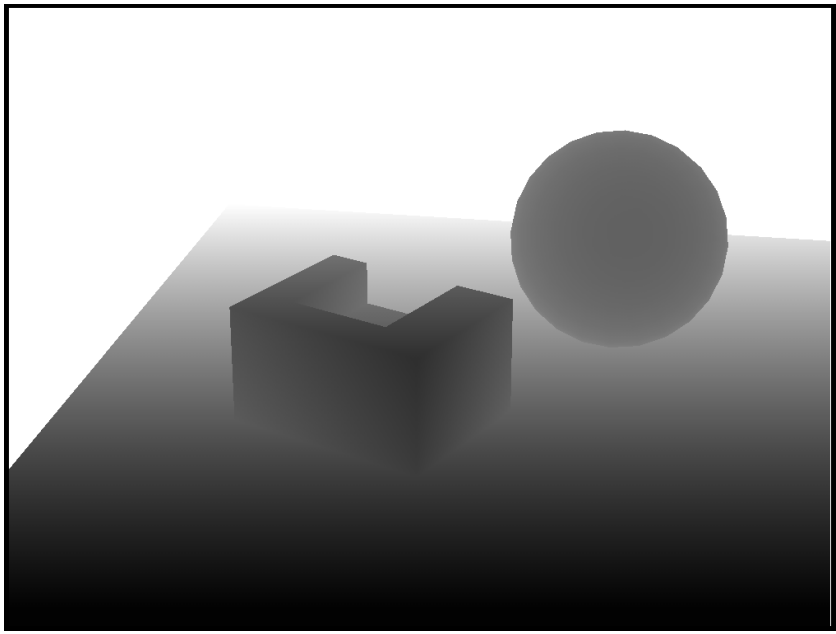
Shadow Maps



Basic Algorithm – the simple explanation:

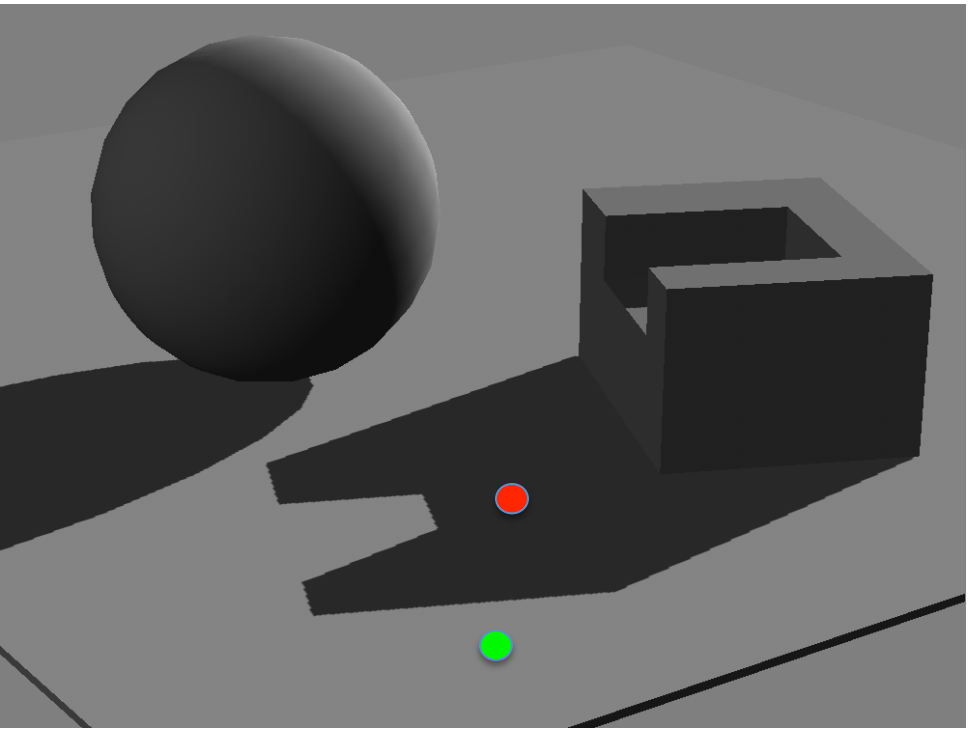
Idea:

- Render image from light source
  - Represents geometry in light
- Render from camera
  - Test if rendered point is visible in the light's view
    - If so -> point in light
    - Else -> point in shadow



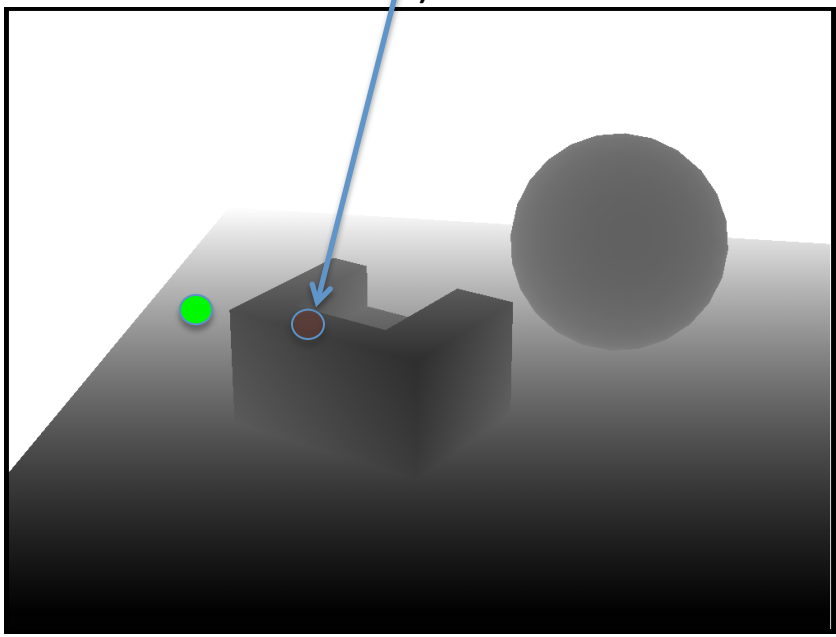
Shadow Map (light's view)

Shadow Maps



Camera's view

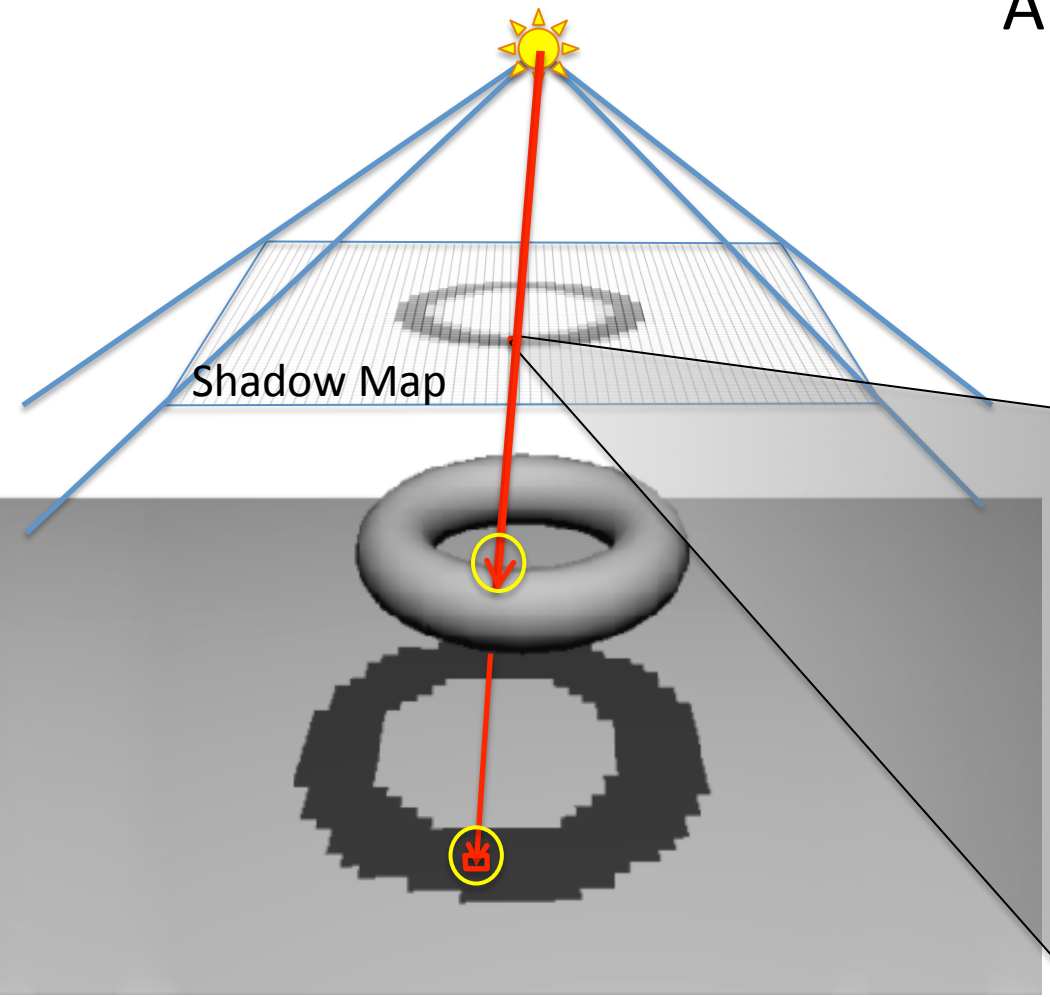
Point not represented in shadow map (point is behind box)



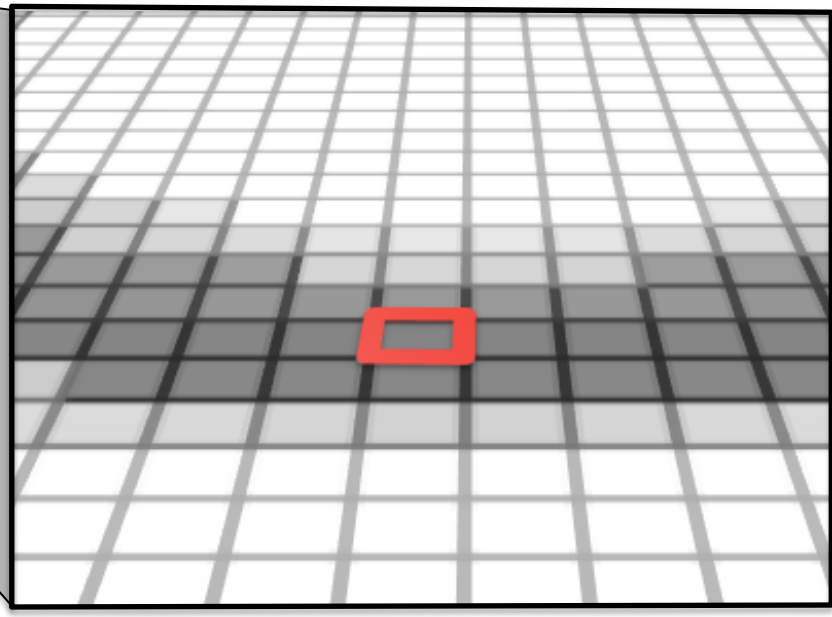
Light's view (Shadow Map)

Depth Comparison

Render depth image from light



A fragment is in shadow if its depth is greater than the corresponding depth value in the shadow map



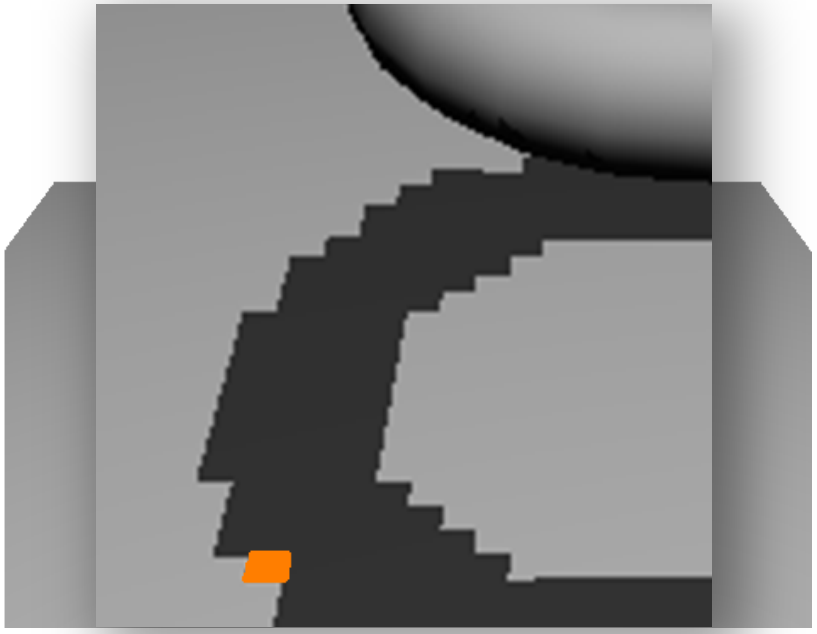
Camera's view

Shadow Maps

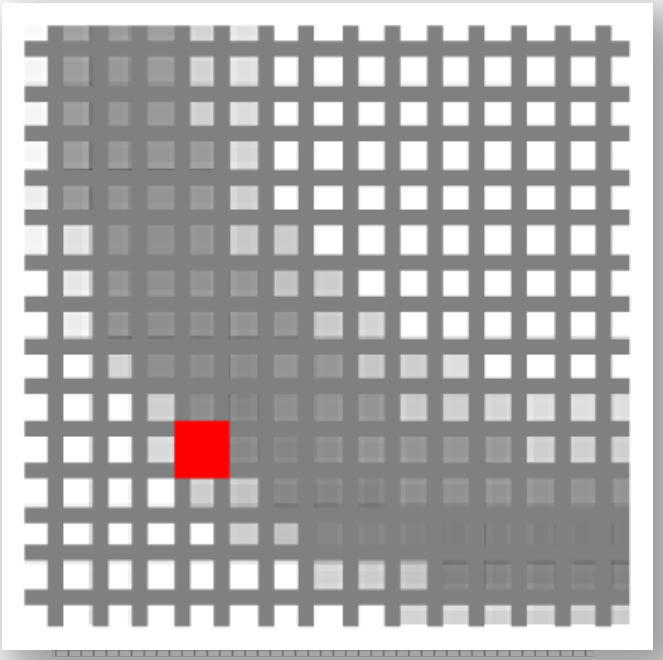
- Pros
  - Very efficient: “This is as fast as it gets”
- Cons…

Shadow Maps - Problems

- Low Shadow Map resolution results in jagged shadows



from viewpoint



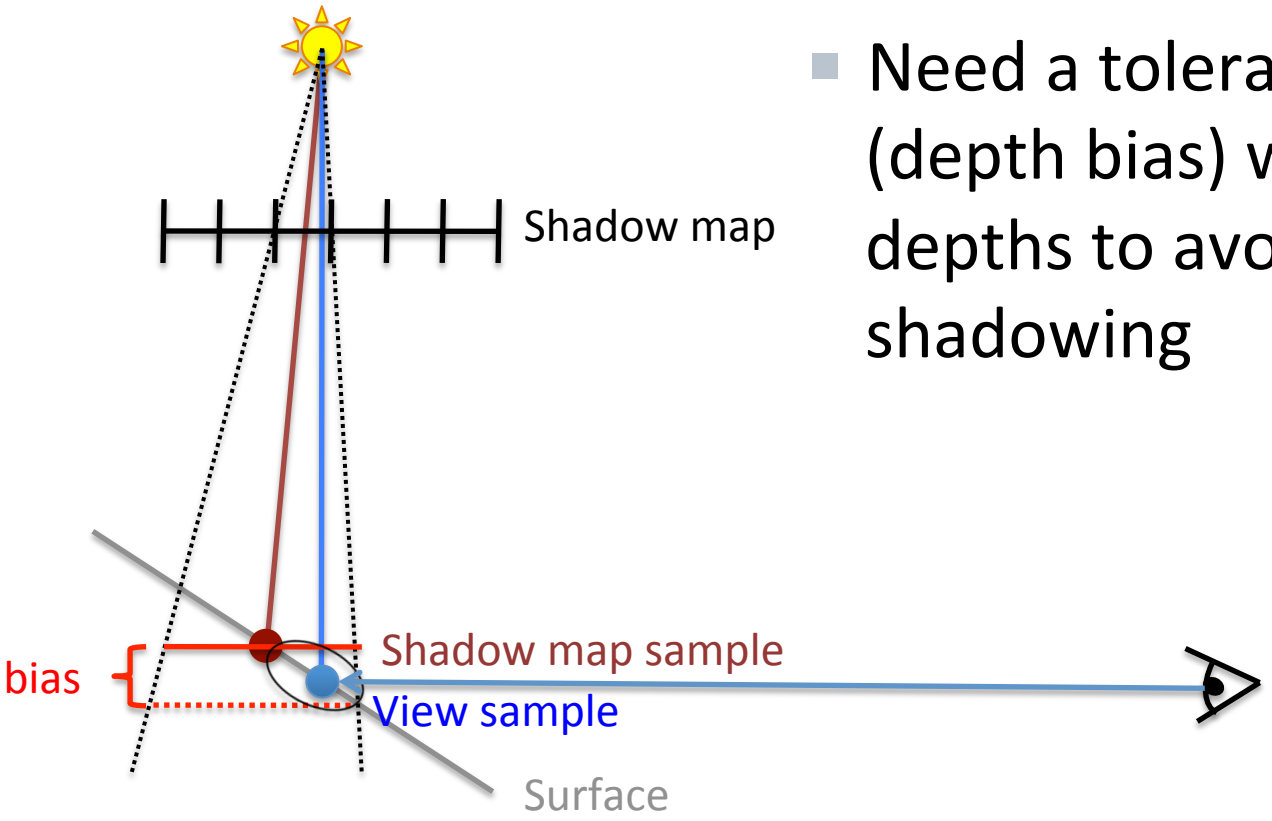
from light

Shadow Maps - Problems

In addition:

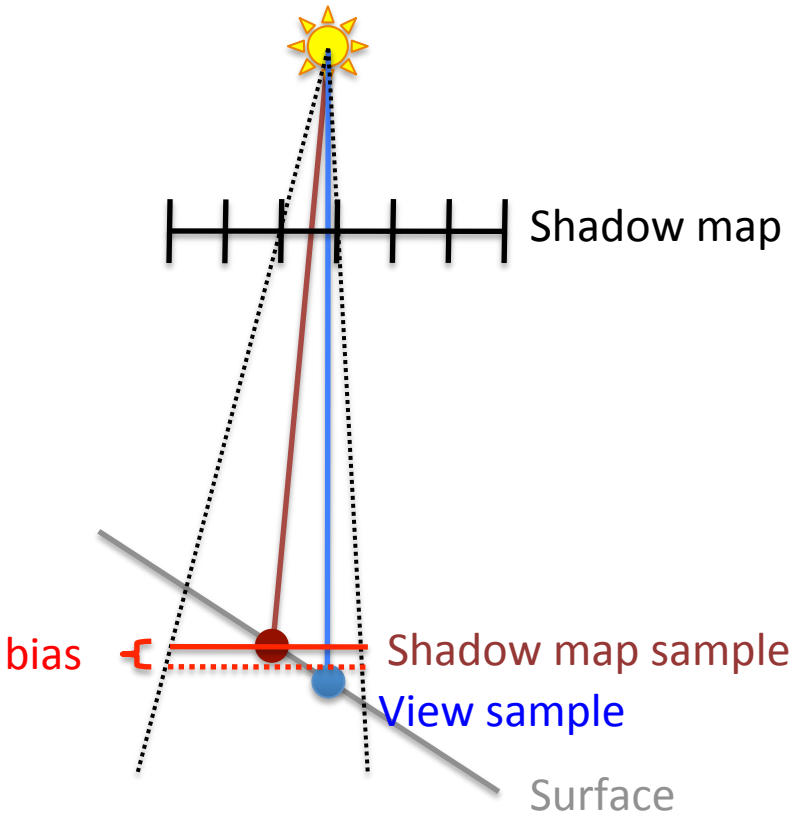
- A tolerance threshold (bias) needs to be tuned for each scene for the depth comparison

Bias

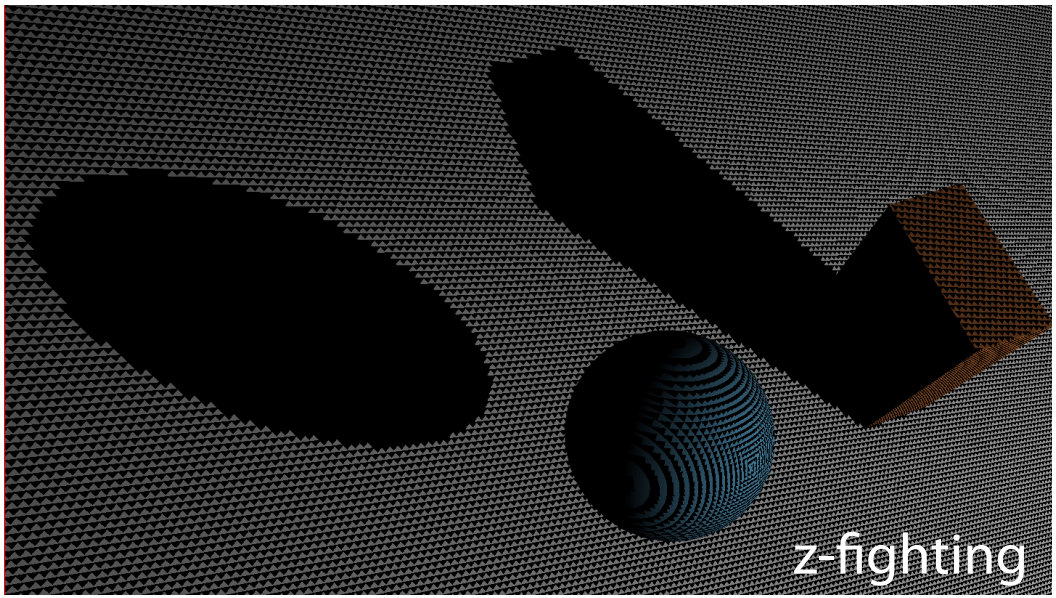


- Need a tolerance threshold (depth bias) when comparing depths to avoid surface self shadowing

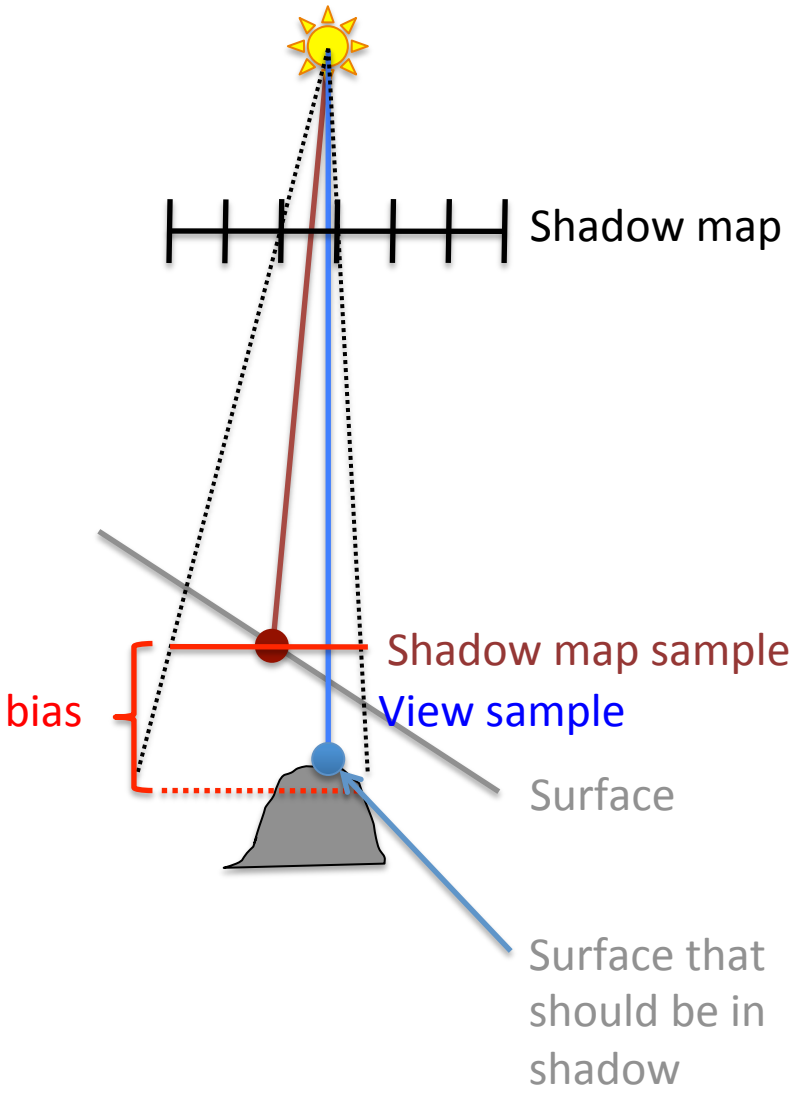
Bias



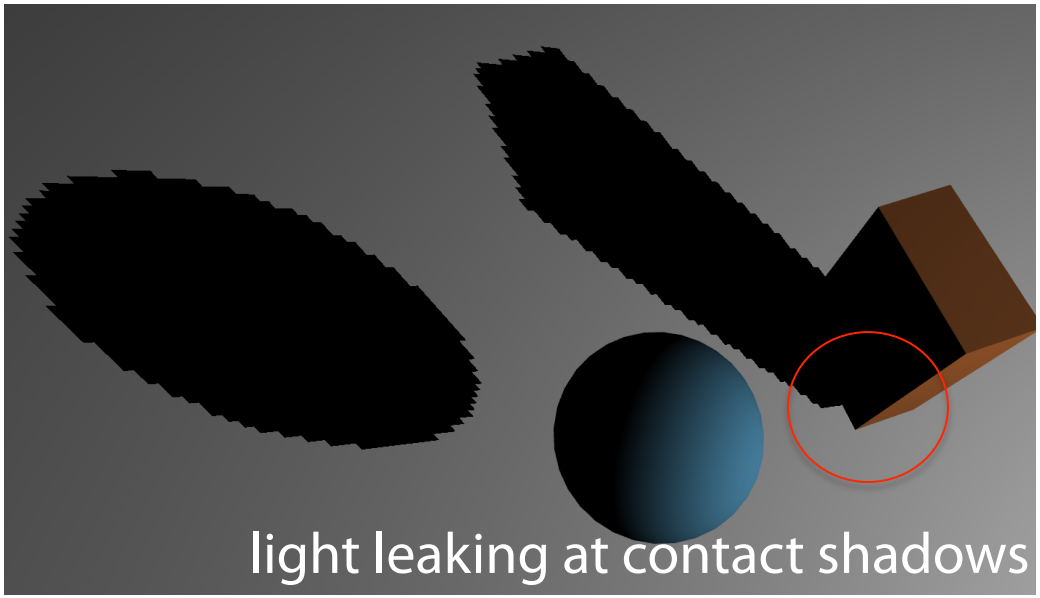
- Need a tolerance threshold (depth bias) when comparing depths to avoid surface self shadowing



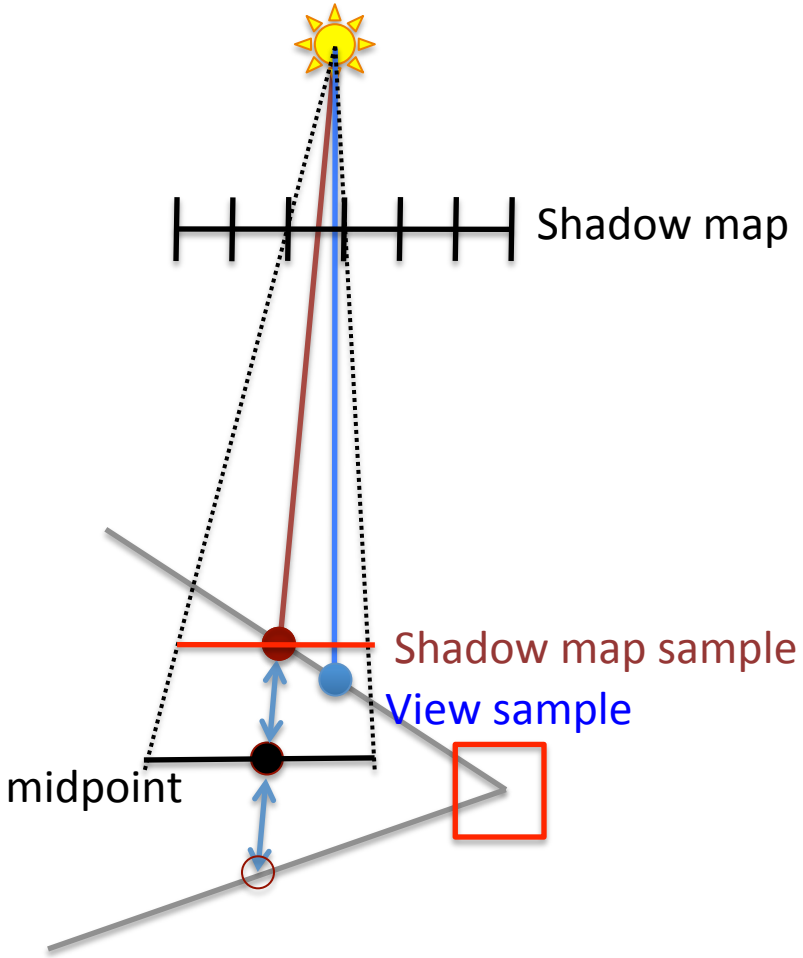
Bias



- Need a tolerance threshold (depth bias) when comparing depths to avoid surface self shadowing



Ameliorating the Bias

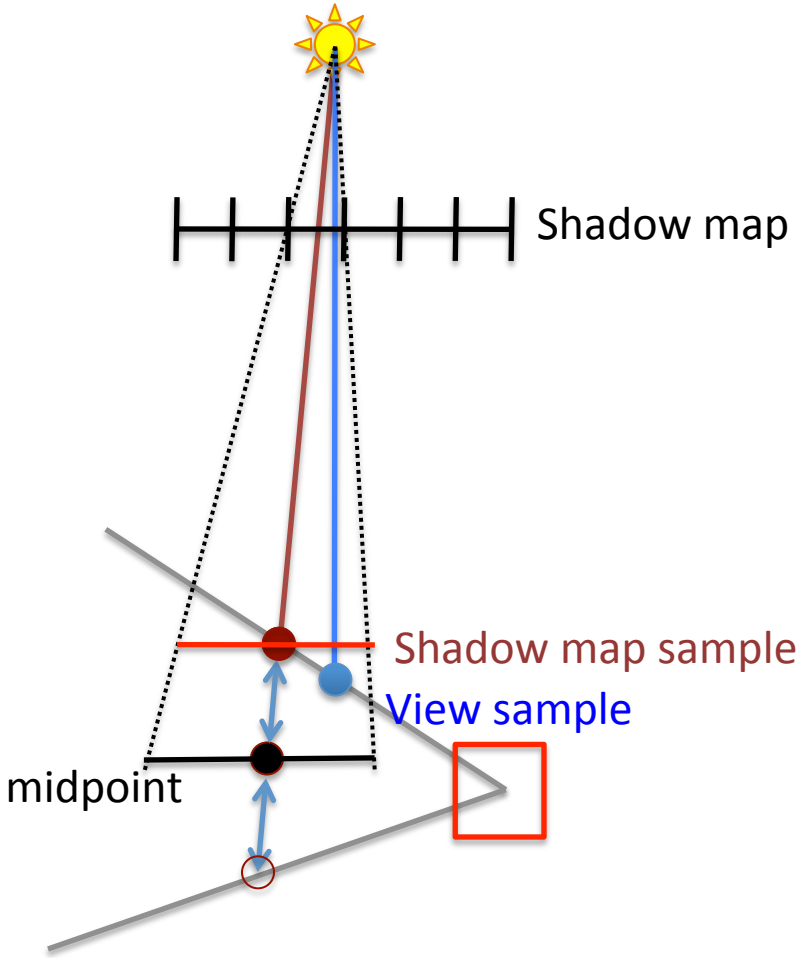


- Midpoint Shadow Maps [Woo 92]

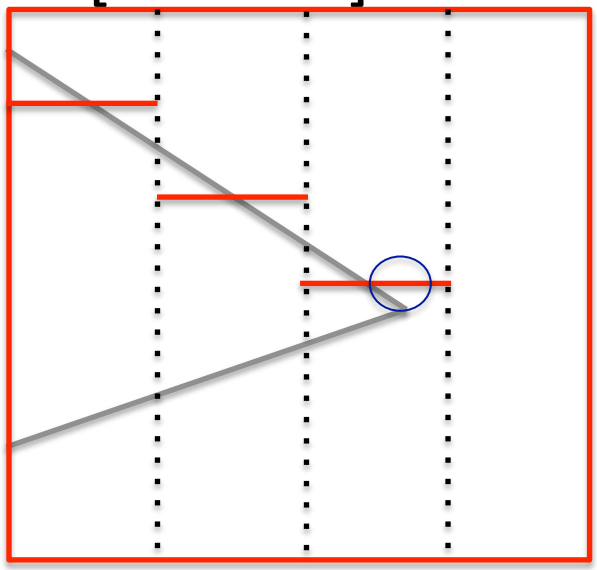
Further methods:

- Second Depth Shadow Mapping [Wang and Molnar94]
- Dual Depth Layer [Weiskopf and Ertl 04]

Ameliorating the Bias



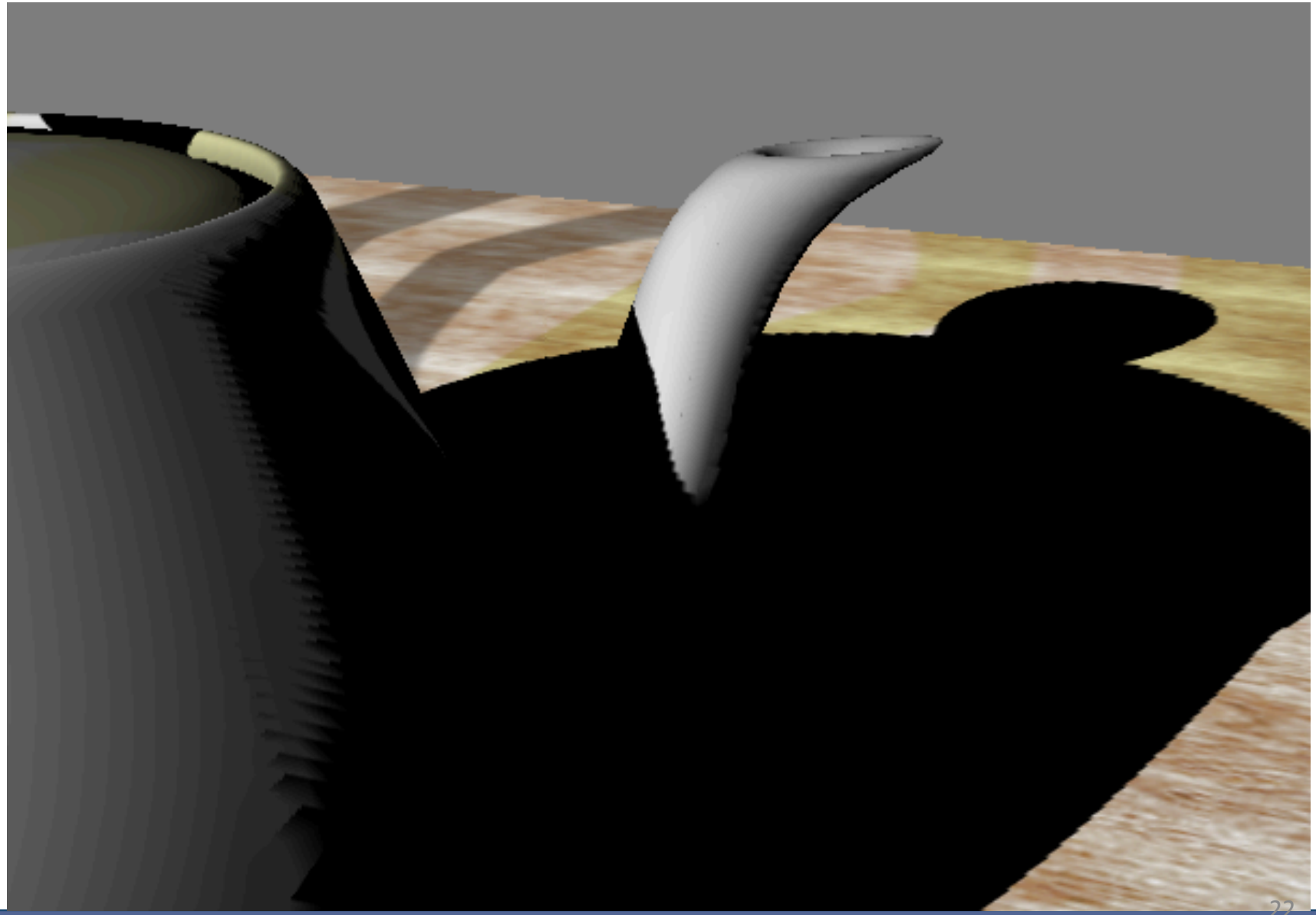
- Midpoint Shadow Maps [Woo 92]



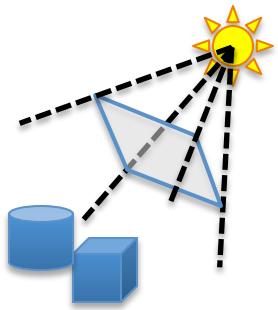
Further methods:

- Second Depth Shadow Mapping [Wang and Molnar94]
- Dual Depth Layer [Weiskopf and Ertl 04]

Shadow Maps

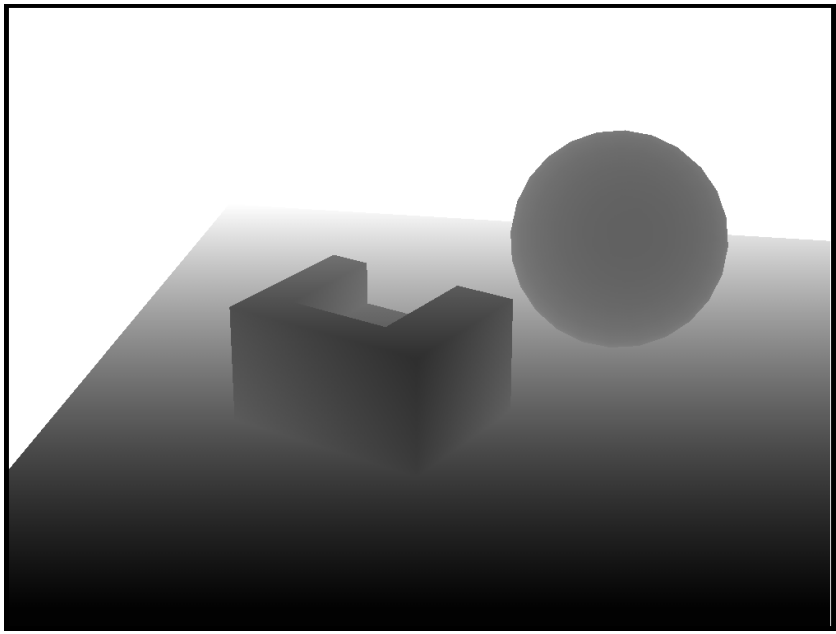


Shadow Maps - Summary



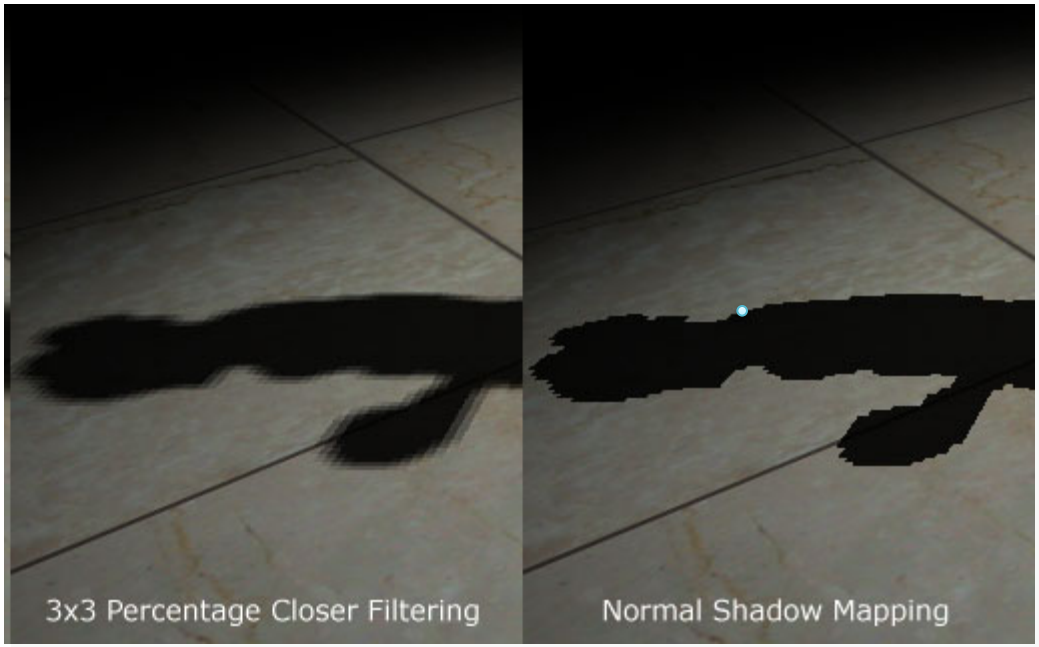
Shadow Map Algorithm:

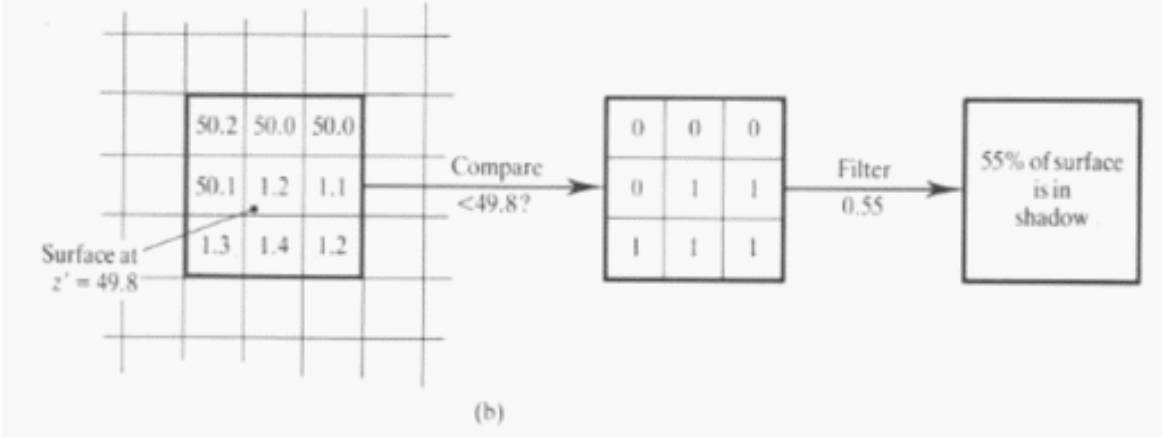
- Render a z-buffer from the light source
  - Represents geometry in light
- Render from camera
  - For every fragment:
    - transform its 3D-pos into shadow map (light space)
    - If depth greater-> point in shadow
    - Else -> point in light
    - Use a bias at the comparison



Shadow Map (=depth buffer)

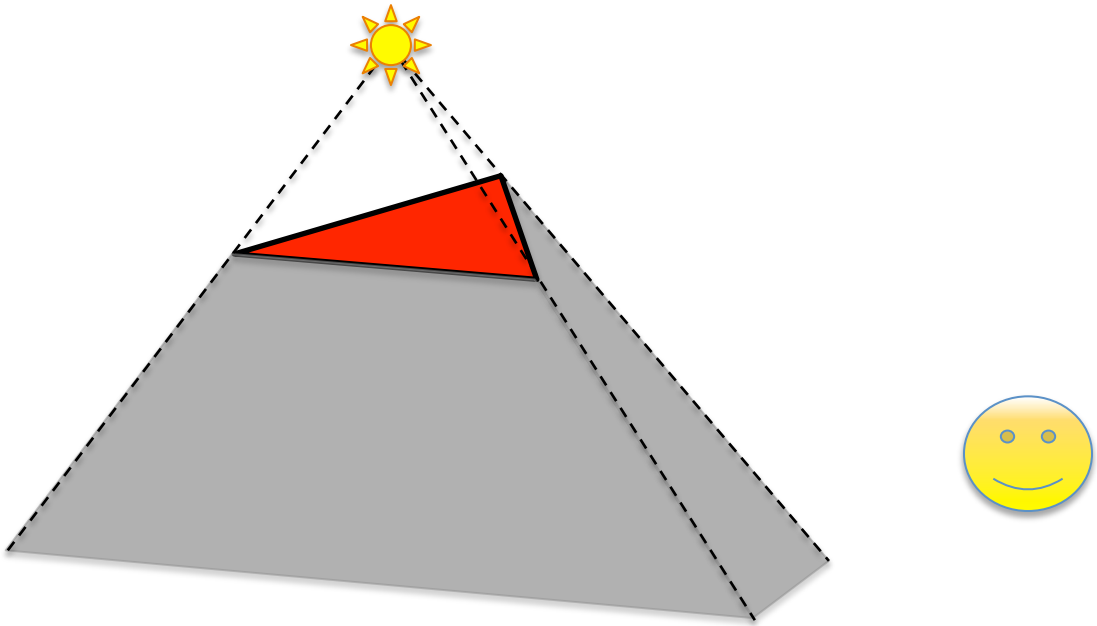
Percentage Closer Filtering





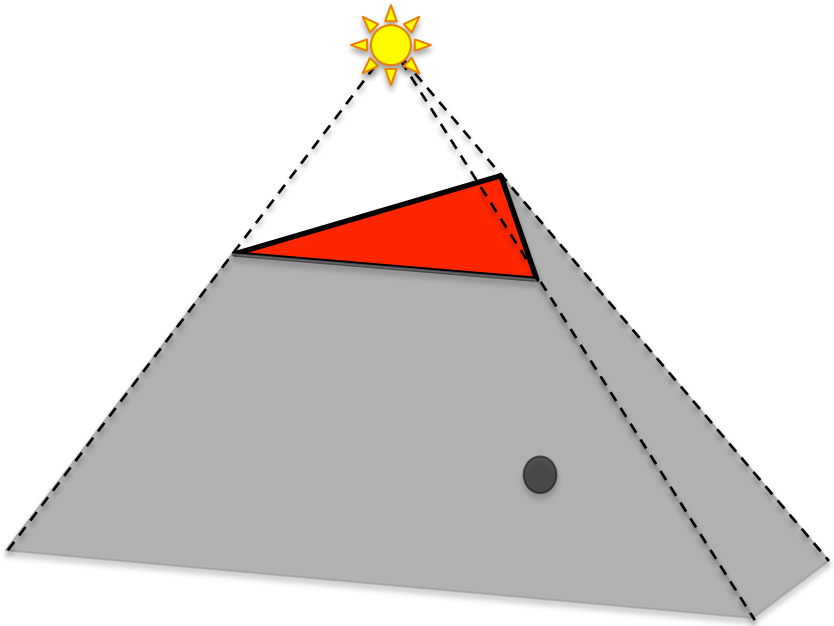
Shadow Volumes

- Concept
  - Create volumes of “space in shadow” from each triangle
    - Each triangle creates 3 quads that extends to infinity



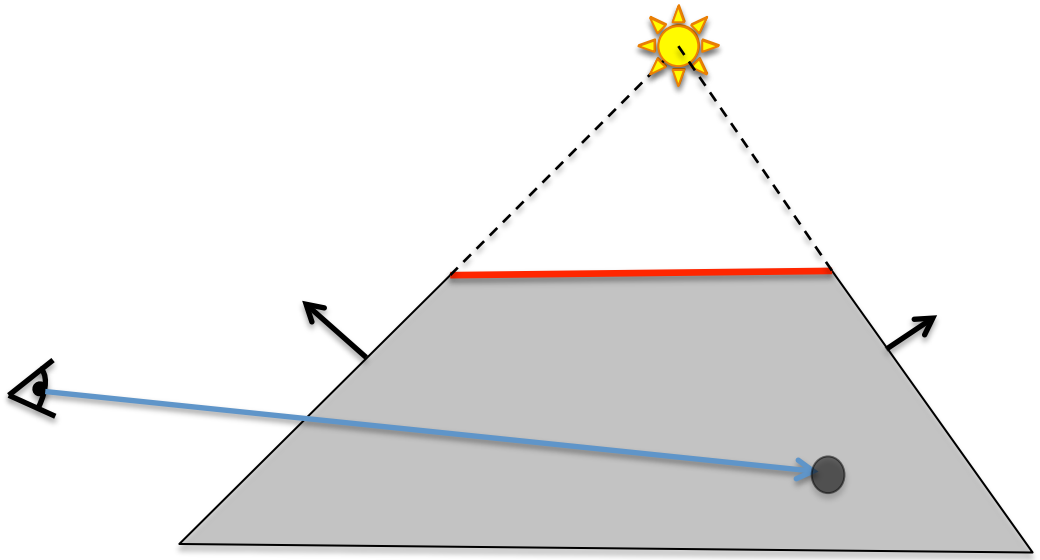
Shadow Volumes

- To test a point, count how many shadow volumes it is located within. One or more means the point is in shadow



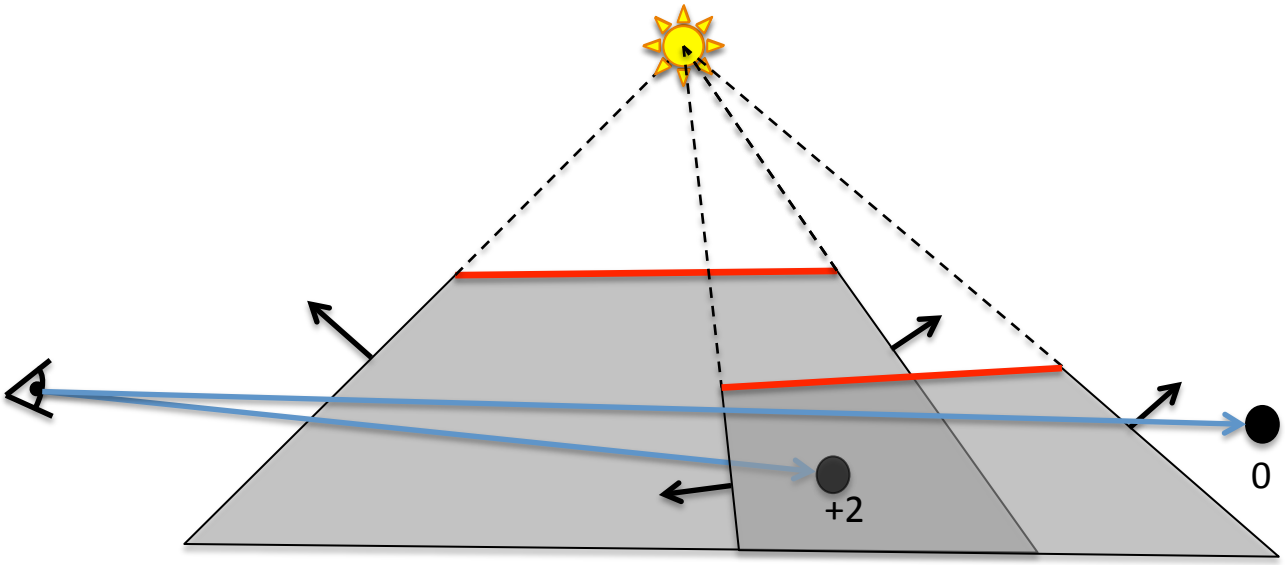
Shadow Volumes

- To test a point, count how many shadow volumes it is located within. One or more means the point is in shadow



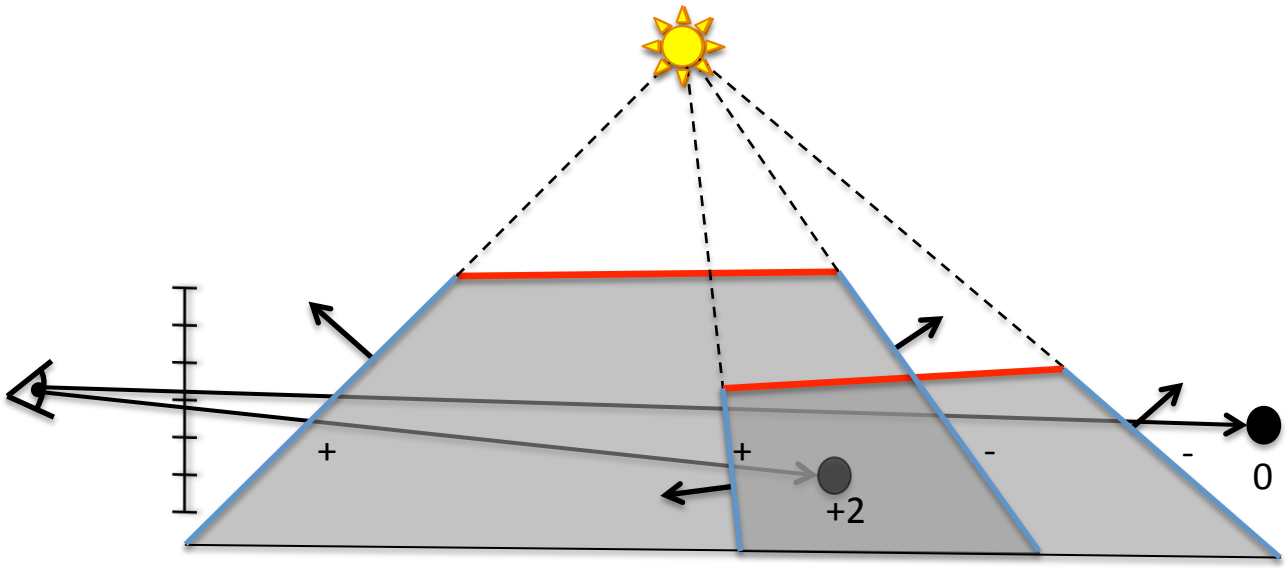
Shadow Volumes

- To test a point, count how many shadow volumes it is located within. One or more means the point is in shadow



Shadow Volumes - concept

- A counter per pixel
- If we go through more frontfacing than backfacing polygons, then the point is in shadow

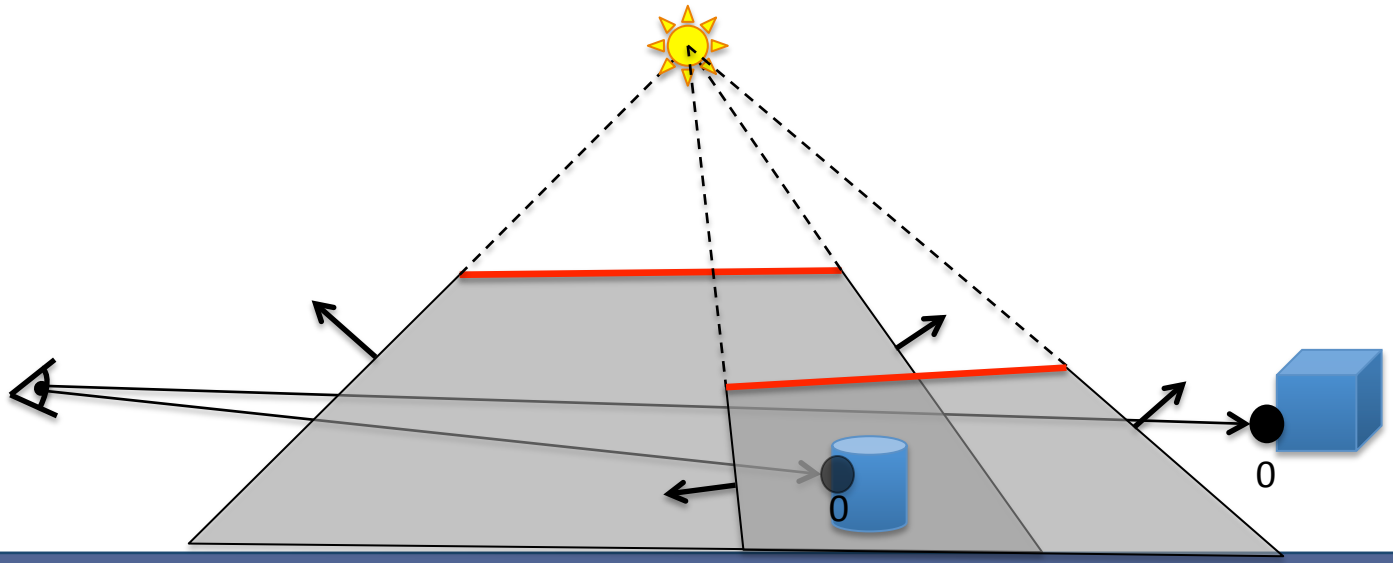


Shadow volume algorithm uses stencil buffer

- 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

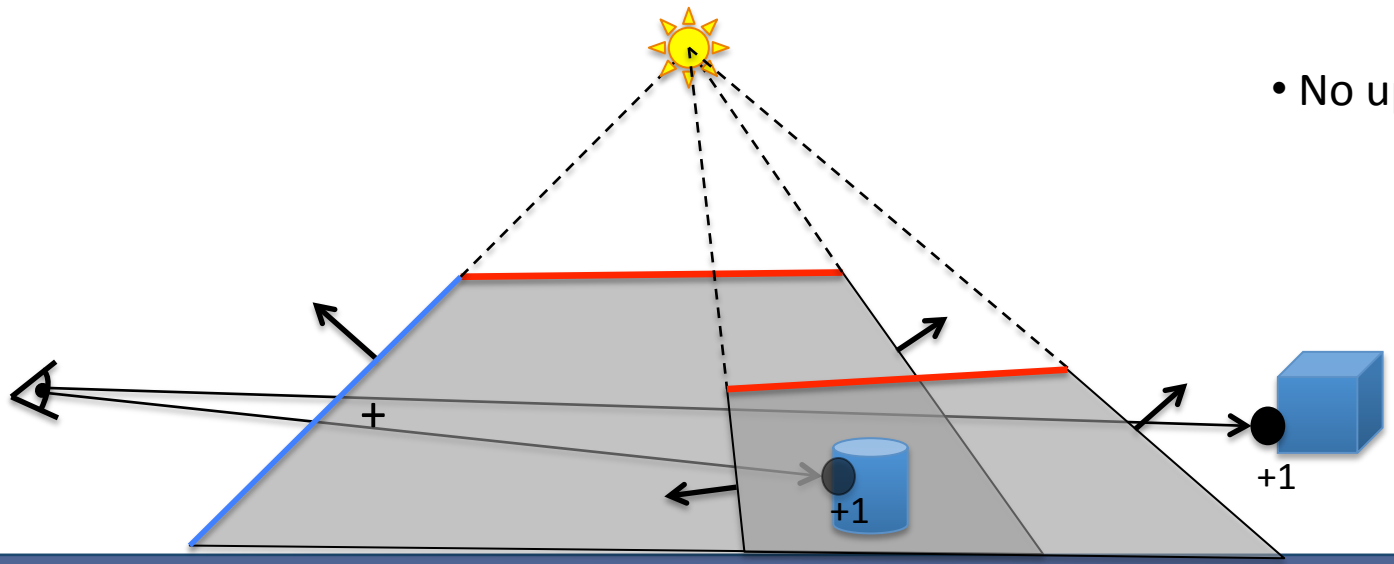
Shadow Volumes - concept

- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume



Shadow Volumes - concept

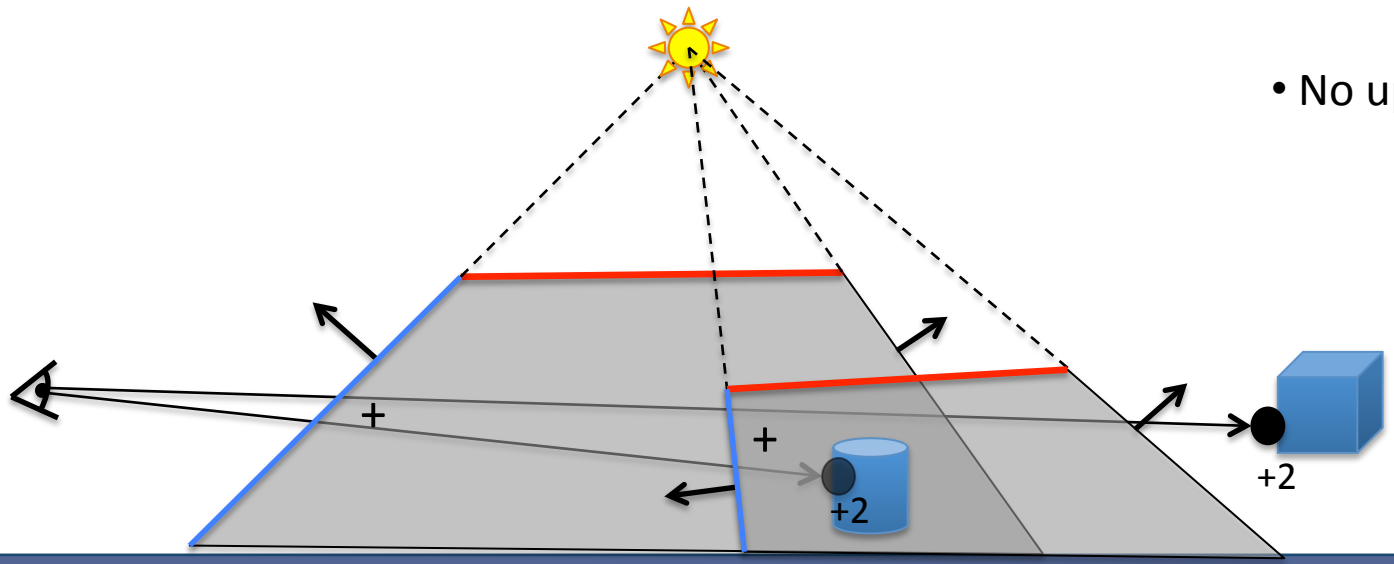
- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume



• No updating of z-buffer

Shadow Volumes - concept

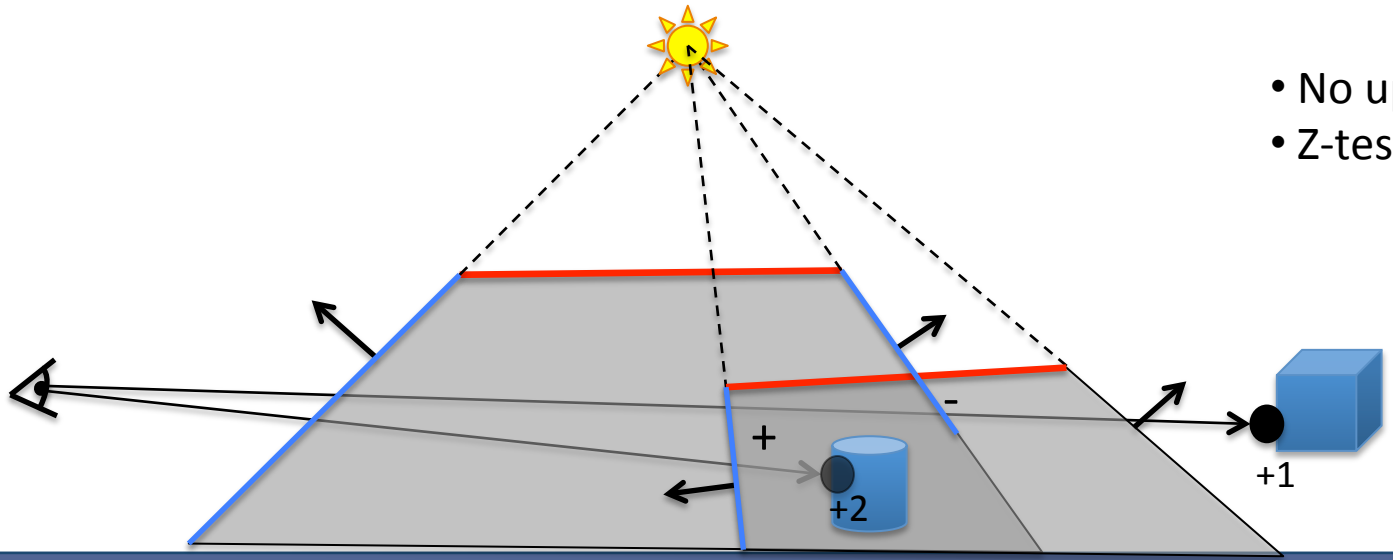
- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume



• No updating of z-buffer

Shadow Volumes - concept

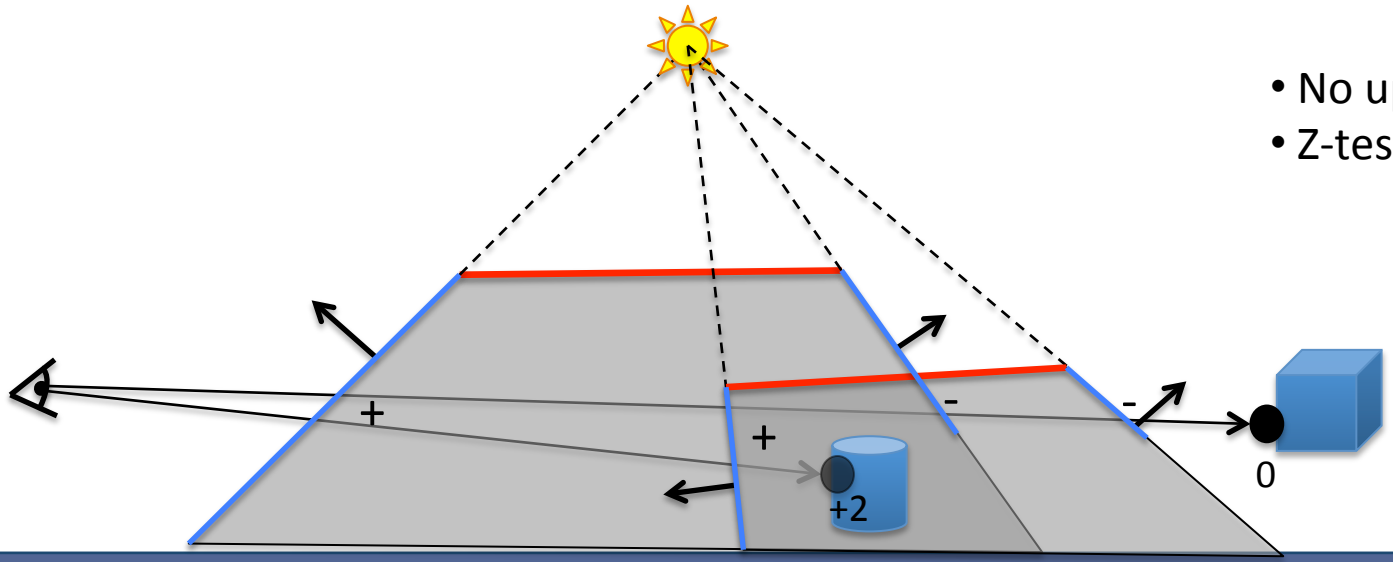
- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume



- No updating of z-buffer
- Z-test is enabled as usual

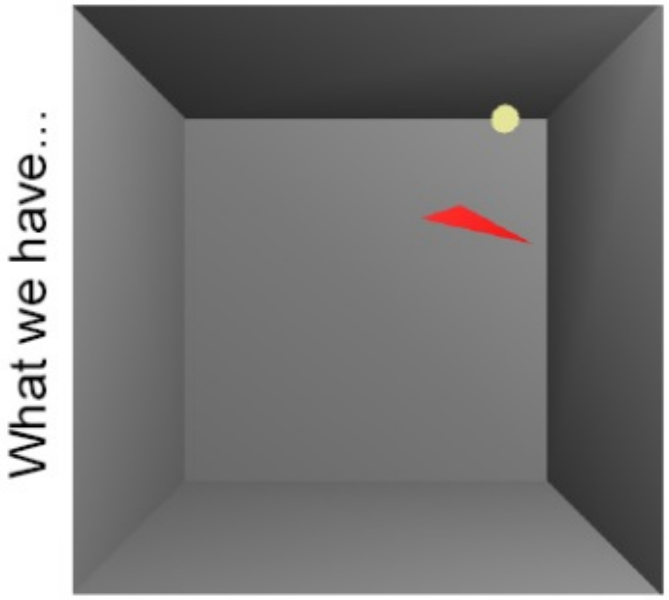
Shadow Volumes - concept

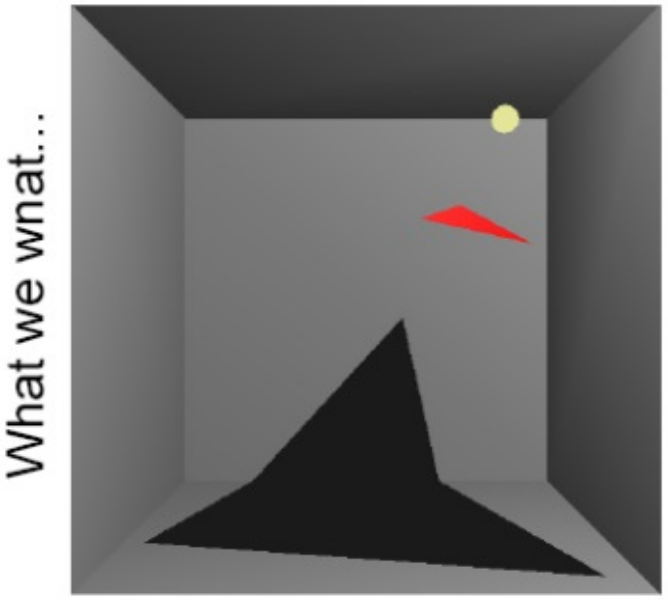
- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume

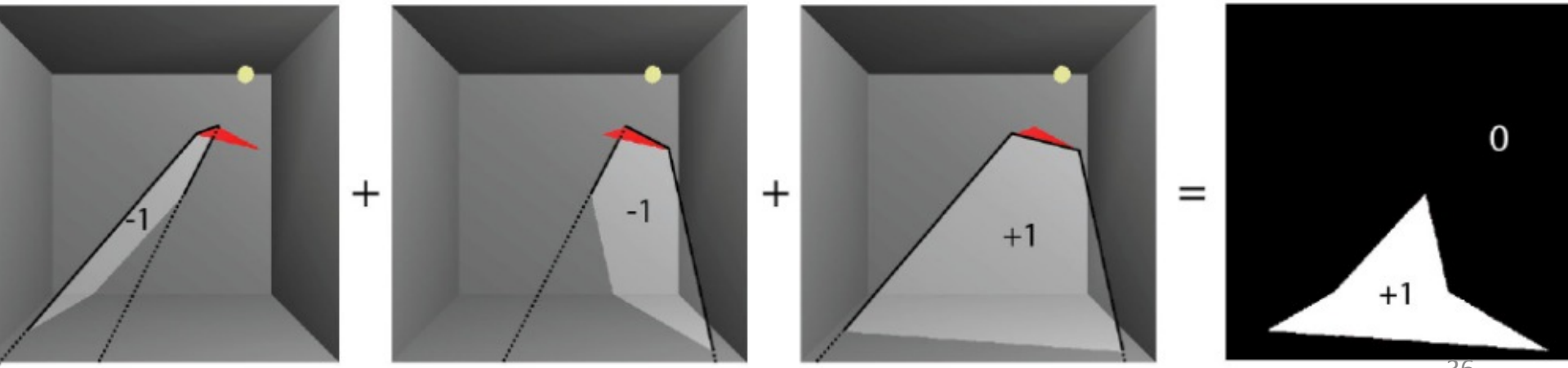


- No updating of z-buffer
- Z-test is enabled as usual

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







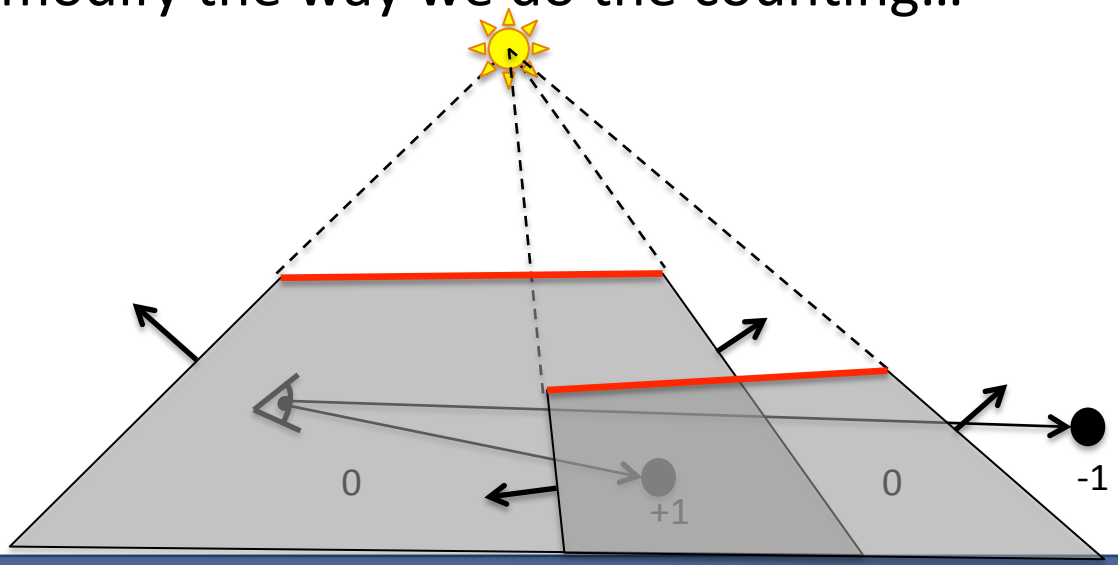
Shadow Volumes with the Stencil Buffer

- A three pass process:
  - **1st pass:** Render *ambient* lighting
  - **2nd pass:**
    - Draw to stencil buffer only
      - Turn off updating of z-buffer and writing to color buffer but still use standard depth test
      - Set stencil operation to
        - » *incrementing* stencil buffer count for *frontfacing* shadow volume quads, and
        - » *decrementing* stencil buffer count for *backfacing* shadow volume quads

        use **glStencilOpSeparate(…)**
  - **3rd pass:** Render *diffuse and specular* where stencil buffer is 0.

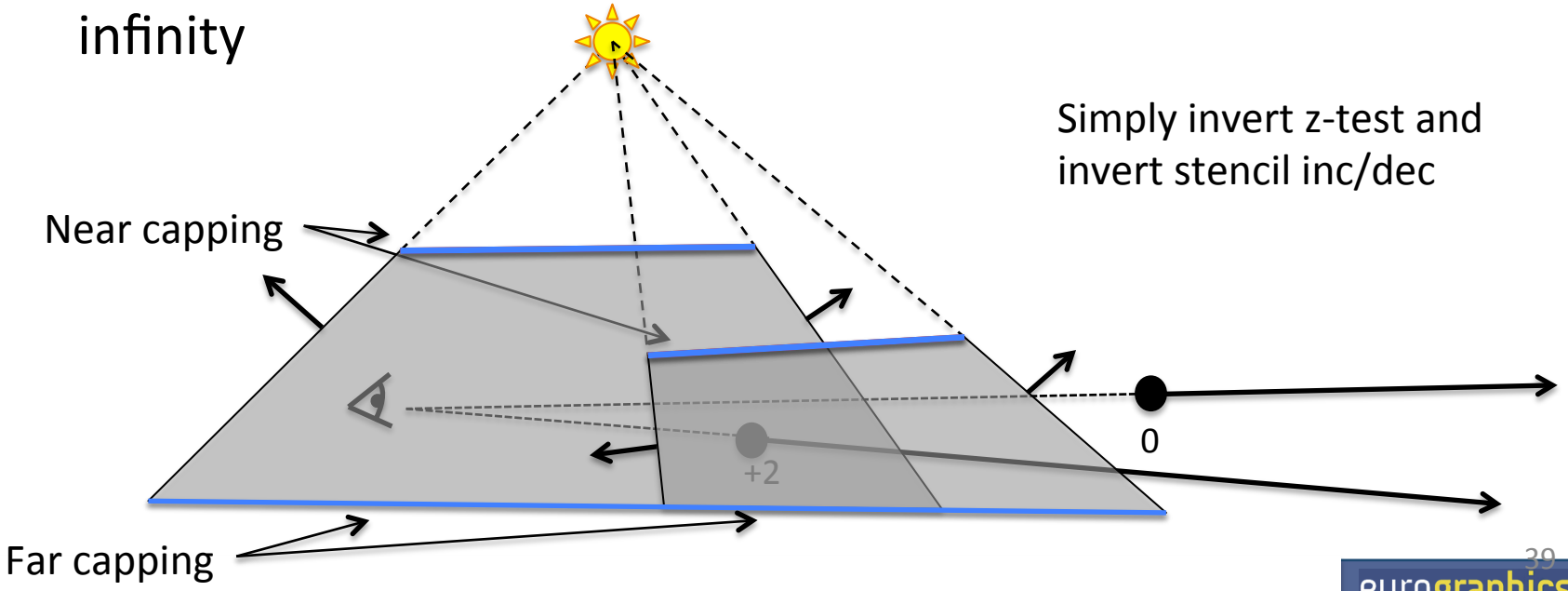
Eye Location Problem

- If the eye is located inside one or more shadow volumes, then the count will be wrong
- Solution:
  - Offset stencil buffer with the #shadow volumes that the eye is located within
  - Or modify the way we do the counting…

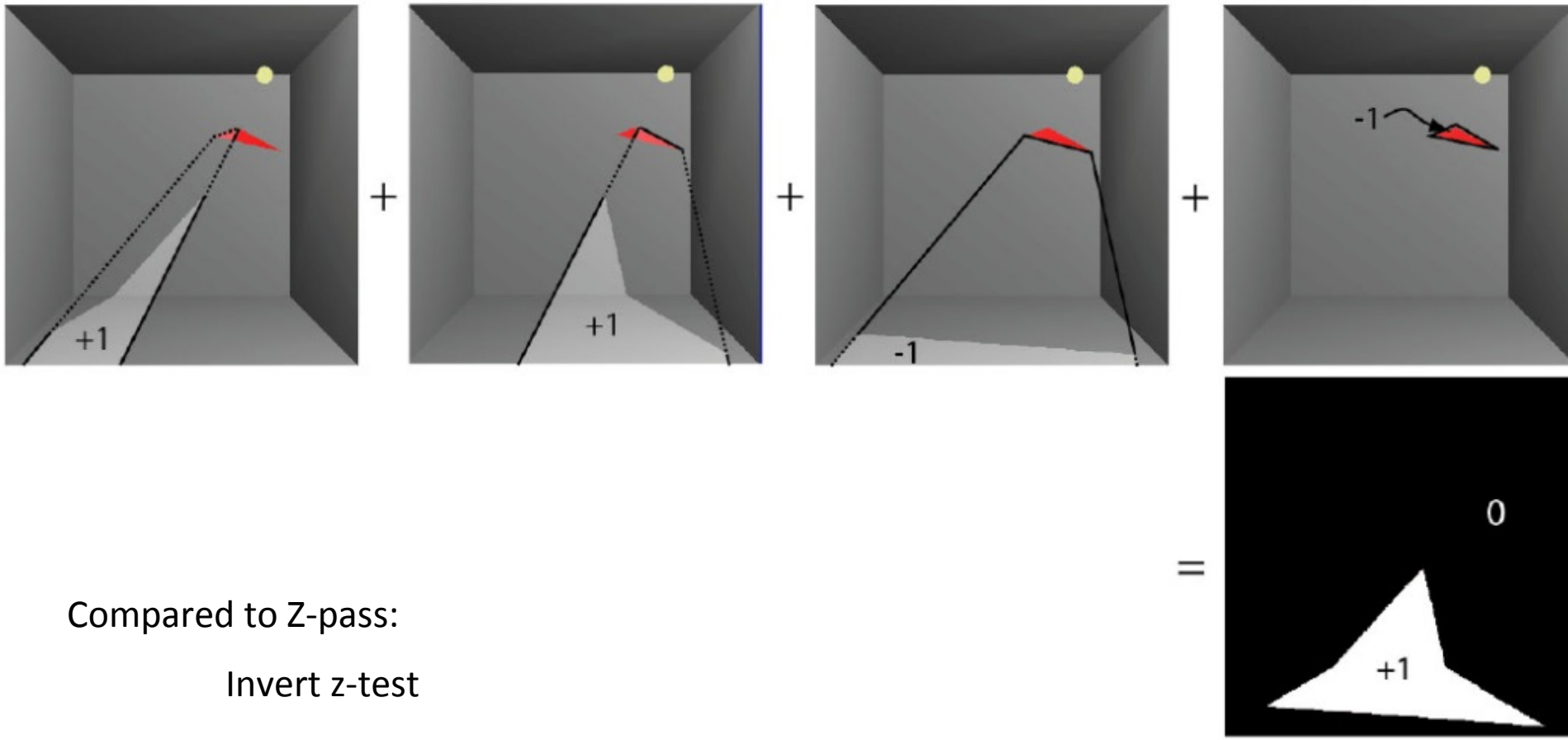


The Z-fail Algorithm

- By [Carmack00] and [Bilodeau and Songy 99]
  - "Carmacks Reverse"
- Count to infinity instead of to the eye
  - We can choose any reference location for the counting
  - A point in light avoids any offset
  - Infinity is always in light – if we cap the shadow volumes at infinity



Z-fail by example



Compared to Z-pass:

Invert z-test

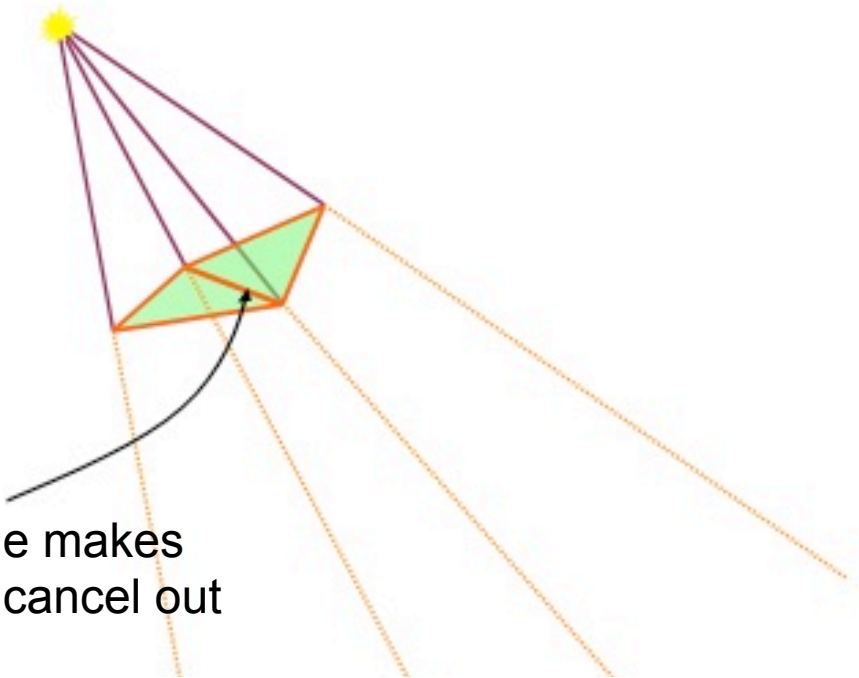
Invert stencil inc/dec

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

Shadow Volumes from Silhouette Edges

Merging shadow volumes:

- An interior edge (non-silhouette edge as seen from the light position) creates two shadow quads that cancel each other out:

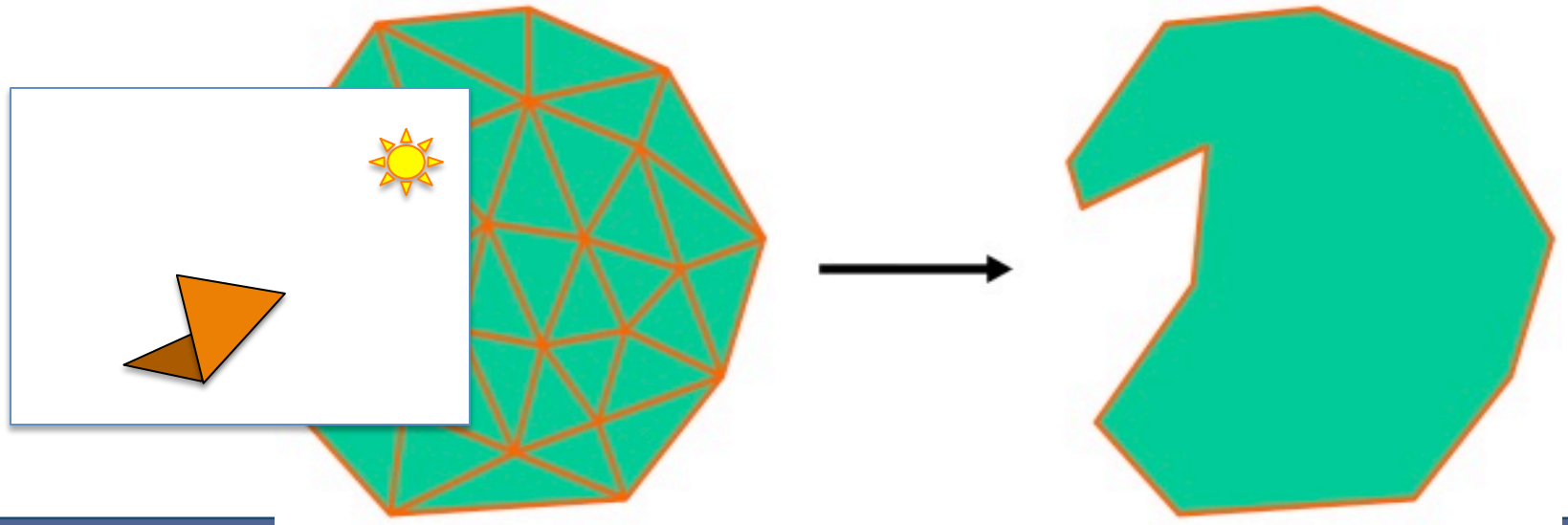


This interior edge makes two quads, which cancel out

Shadow Volumes from Silhouette Edges

Merging shadow volumes:

- An interior edge (non-silhouette edge as seen from the light position) creates two shadow quads that cancel each other out:
- Thus, popular to create shadow volumes only from silhouette edges as seen from the light source
  - Avoids rendering of many useless shadow quads

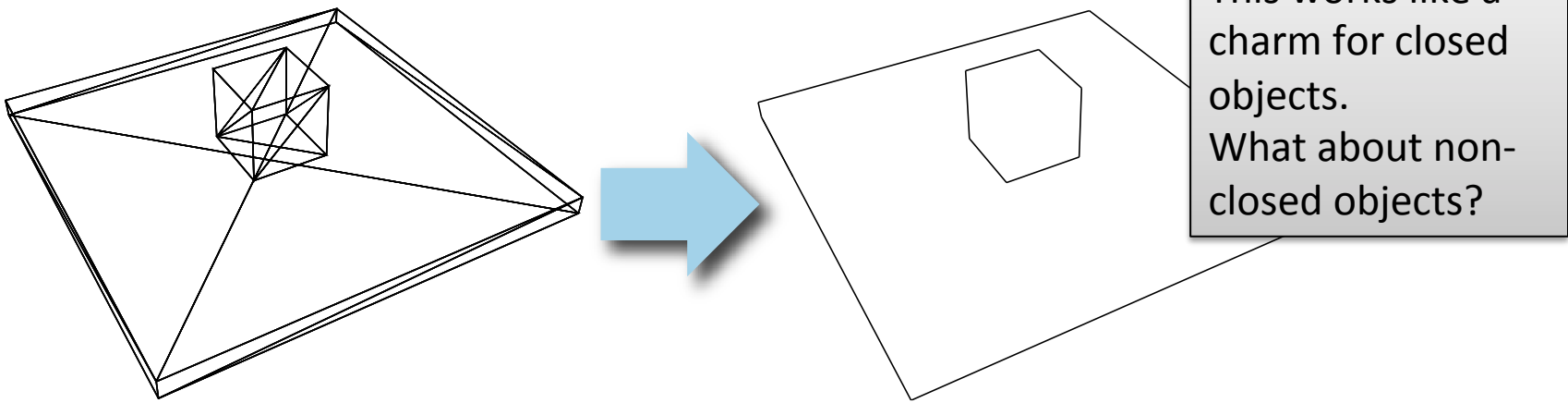


Example of silhouettes from light position

Shadow Volumes from Silhouette Edges

Merging shadow volumes:

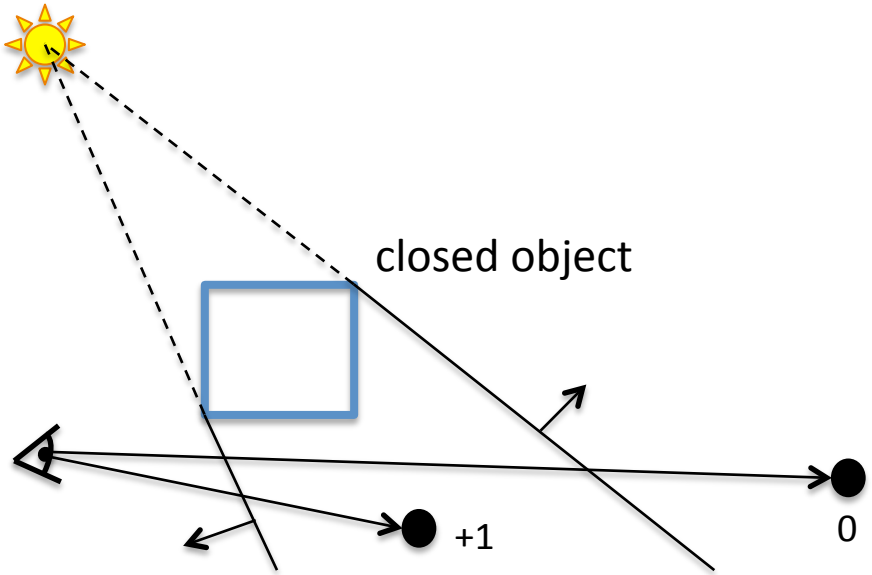
- An edge that is shared by two triangles facing the light creates two shadow quads that cancel each other out
- Thus, create shadow volumes only from silhouette edges as seen from the light source
  - Avoids rendering of many useless shadow quads
  - A real example:



This works like a charm for closed objects.
What about non-closed objects?

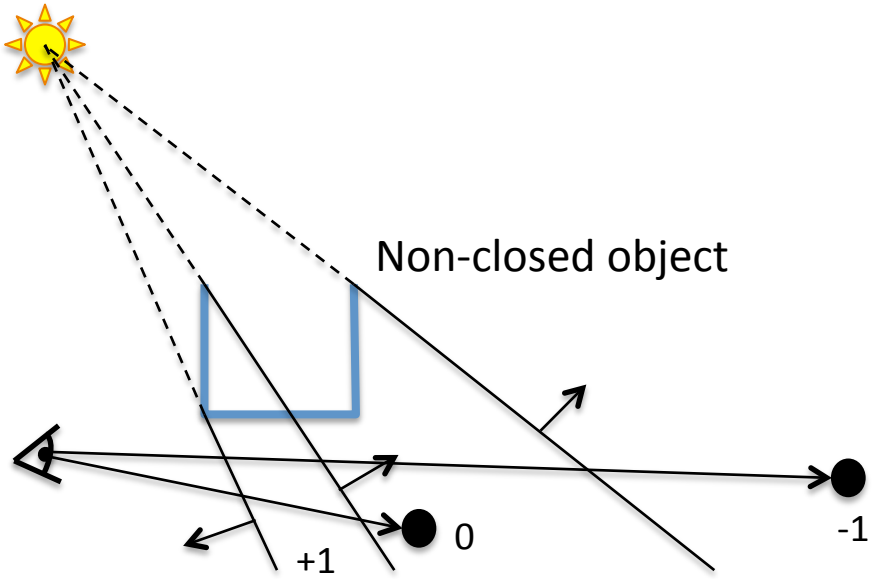
Shadow Volumes from Silhouette Edges

It is a misconception that objects **needs** to be closed to remove non-silhouette edges.



Shadow Volumes from Silhouette Edges

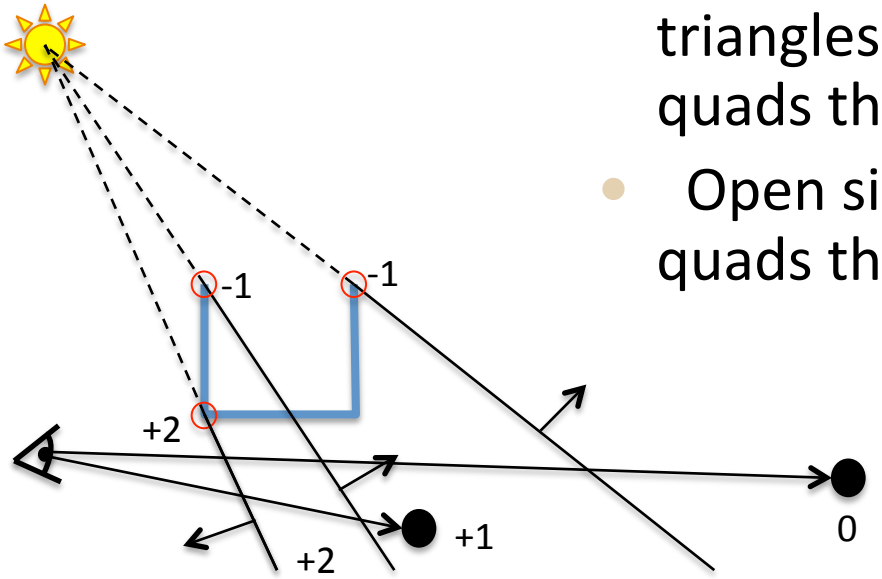
It is a misconception that objects needs to be closed



Shadow Volumes from Silhouette Edges

It is a misconception that objects needs to be closed

Fixed by [Bergeron 86]



Observation:

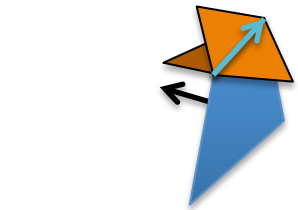
- Silhouette edges with two adjacent triangles should actually create shadow quads that inc/dec count by 2
- Open silhouette edges create shadow quads that inc/dec count by one

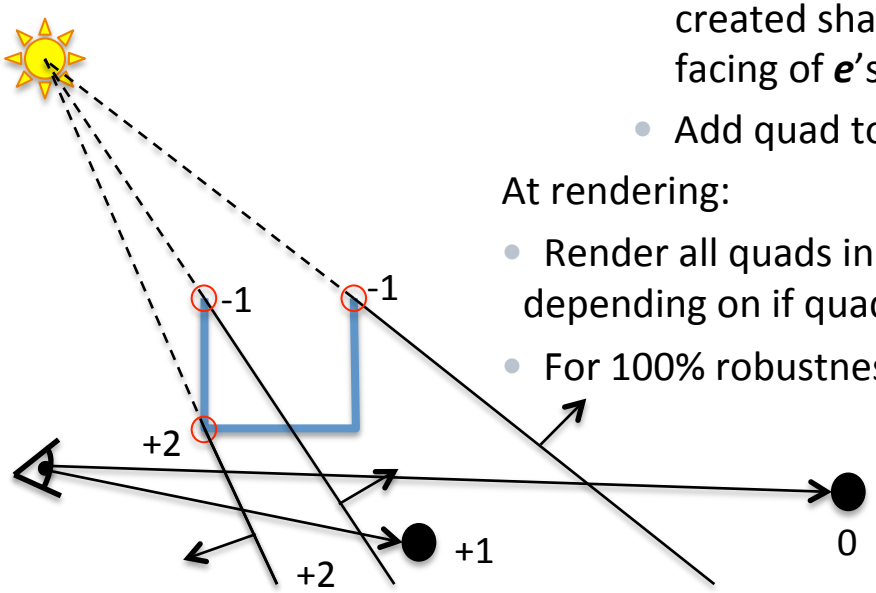
Stencil value >0 means shadow

Works identically for Z-fail

Shadow Volumes from Silhouette Edges

For general objects with edges that are shared by many triangles:





Preprocess (or in geometry shader):

- For each triangle edge ***e*** in scene:
  - Choose edge ***e***'s direction
    - Create ***e***'s shadow volume quad
    - For each adjacent triangle
    - Inc/dec per-edge counter c_e depending on if triangle's created shadow volume quad would have same/opposite facing of ***e***'s quad.
    - Add quad to list L, if c_e != 0.

At rendering:

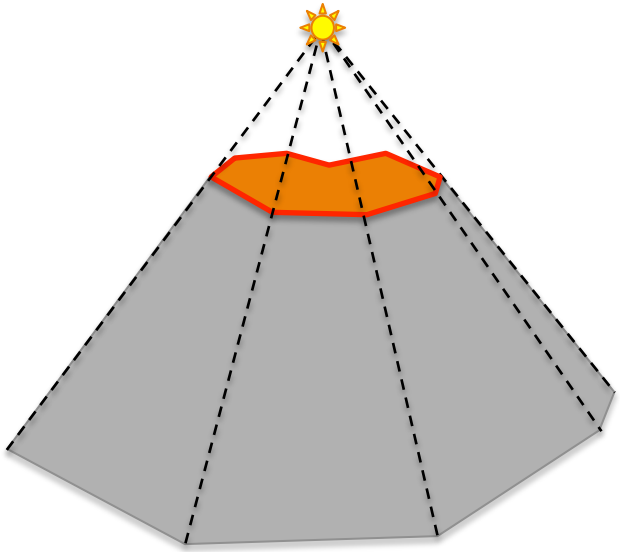
- Render all quads in L, and inc/dec stencil by the quad's c_e depending on if quad is front/back-facing eye.
- For 100% robustness, see our *book Real-Time Shadows*

Stencil value >0 means shadow

Works identically for Z-fail

Shadow Volumes - Summary

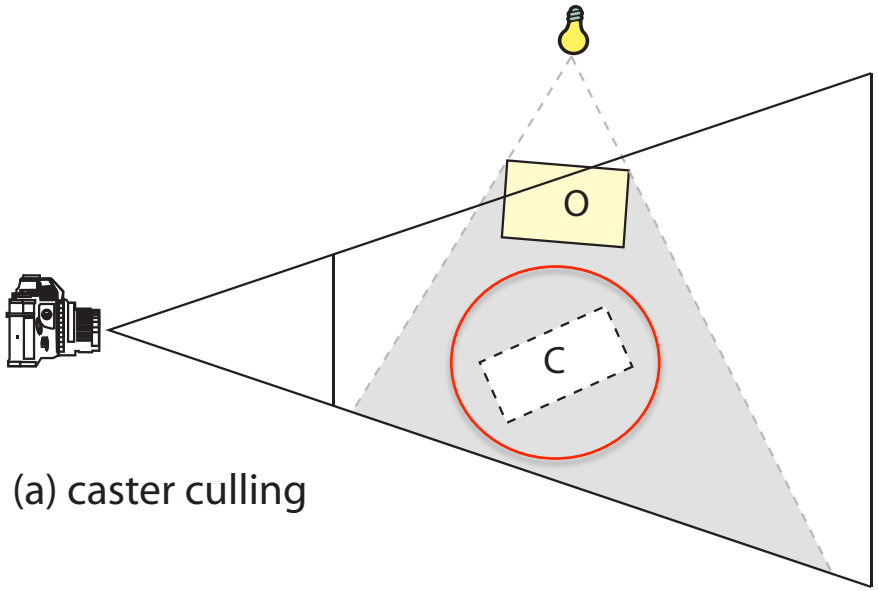
- Pros:
  - High quality
- Cons:
  - OVERDRAW



Bonus

Culling and Clamping

- **Culling of Shadow Volumes** [Lloyd et al. 2004][Stich et al. 2007]
  - **Culling of Shadow Casters** if it is located totally within shadow
    - Tested against a shadow depth map

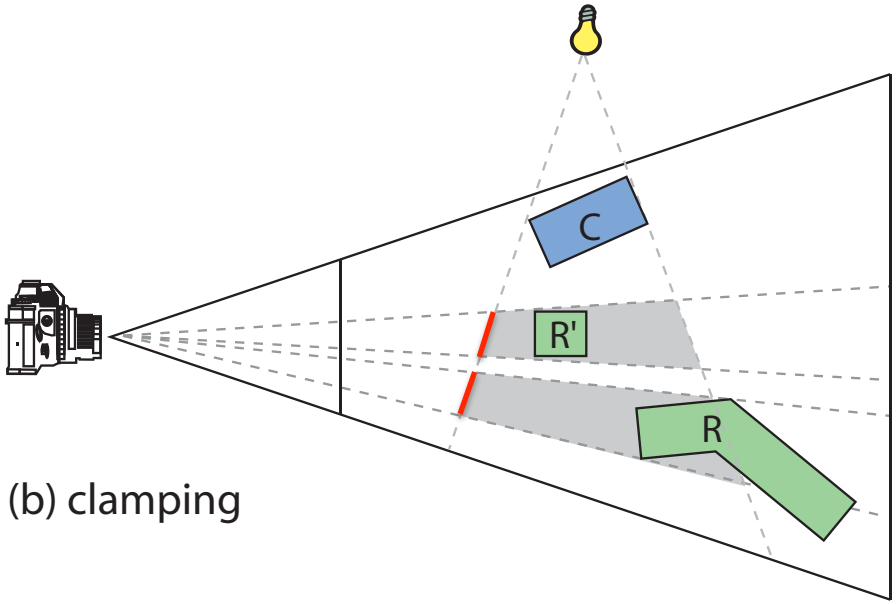


(a) caster culling

Bonus

Culling and Clamping

- **Clamping of Shadow Volumes** [Lloyd et al. 2004] [Eisemann and Decoret 2006]
  - Idea: Only render parts of shadow quads that affects a shadow receiver
    - Tested against AABB around shadow receivers

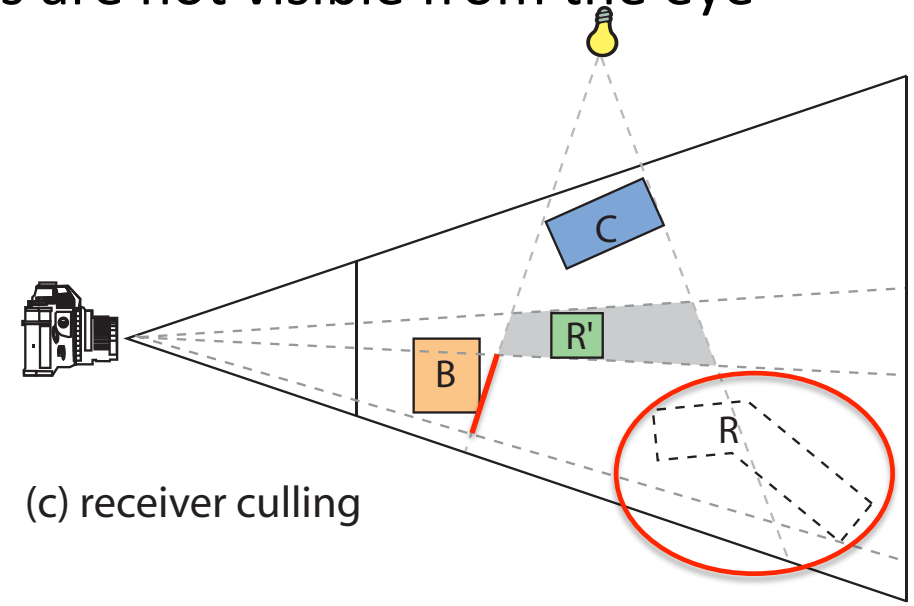


(b) clamping

Bonus

Culling and Clamping

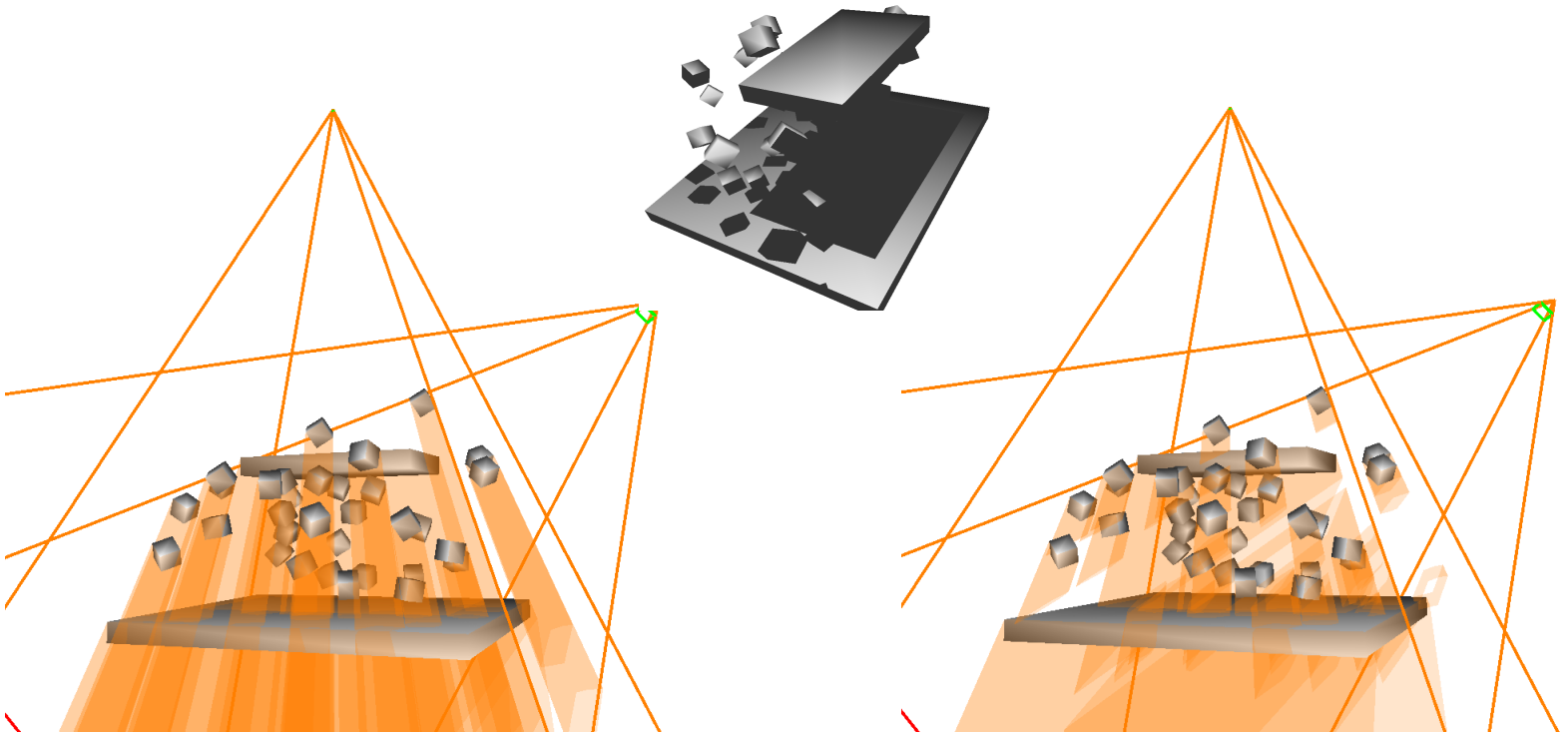
- **Culling of Shadow Volumes** [Lloyd et al. 2004] [Eisemann and Decoret 2006]
  - **Receiver Culling**
    - Idea: Cull part of shadow volumes where shadow receivers are not visible from the eye



(c) receiver culling

Bonus

Culling and Clamping

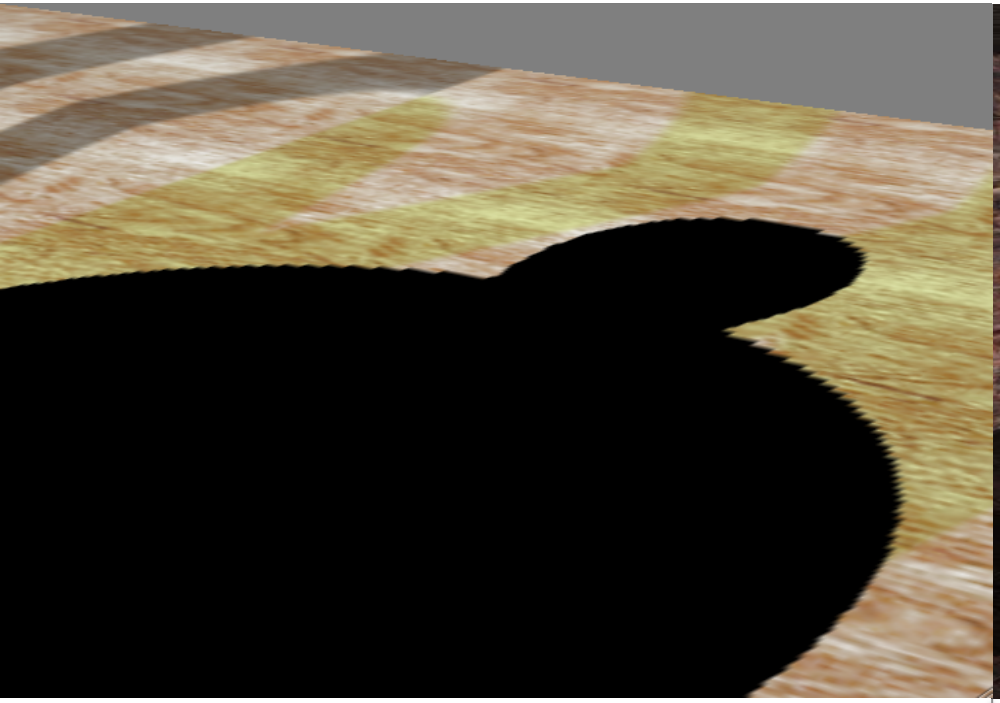


Shadow volumes

CC Shadow volumes

Illustrates reduced depth complexity when using Culling and Clamping

Shadow Maps vs Shadow Volumes



Shadow Maps

- *Good*: Handles any rasterizable geometry, **constant cost** regardless of complexity, map can sometimes be reused. **Very fast**.
- *Bad*: Frustum limited. **Jagged shadows** if res too low, **biasing** headaches.
  - Solution:
  - 6 SM (cube map), high res., use filtering (huge topic)

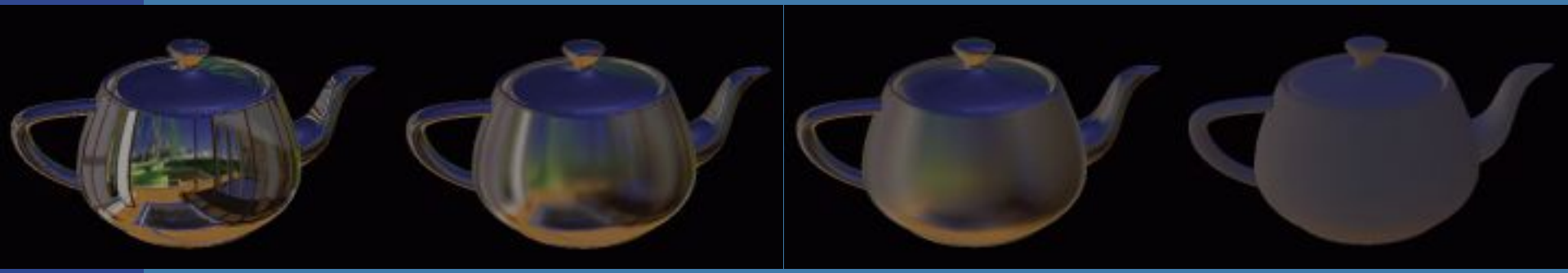
Shadow Volumes

- *Good*: shadows are **sharp**. Handles omni-directional lights.
- *Bad*: **3 passes**, shadow polygons must be generated and rendered → lots of polygons & **fill**
  - Solution: culling & clamping (or per-triangle SV using hierarchical shadow buffer)

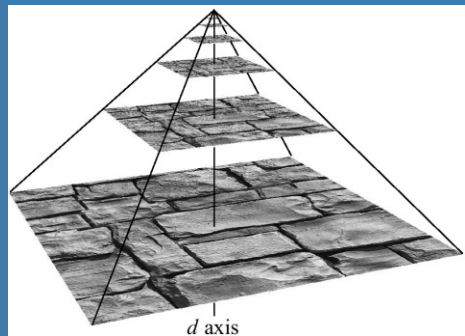
Reflections

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);



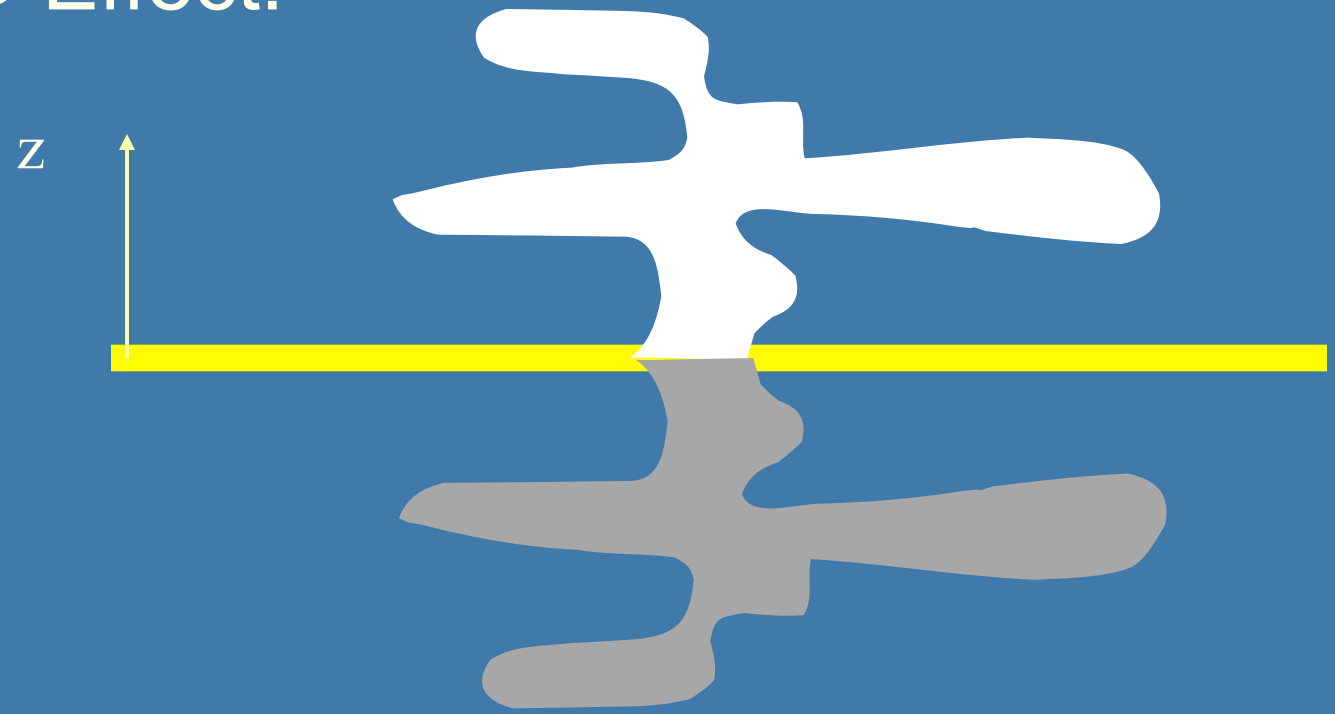
Planar reflections

- 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

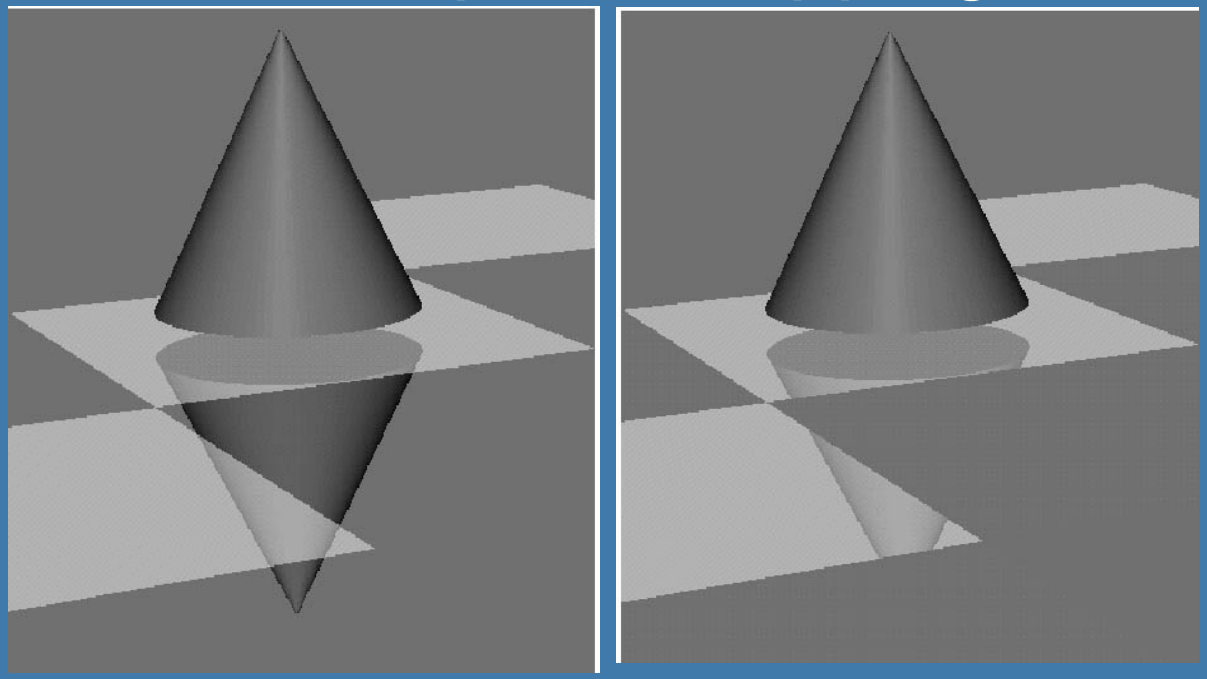
Planar reflections

- Assume plane is z=0
- Then apply a scaling matrix **S**(1,1,-1);
- Effect:

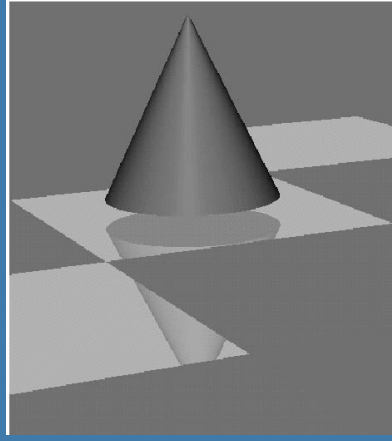


Planar reflections

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

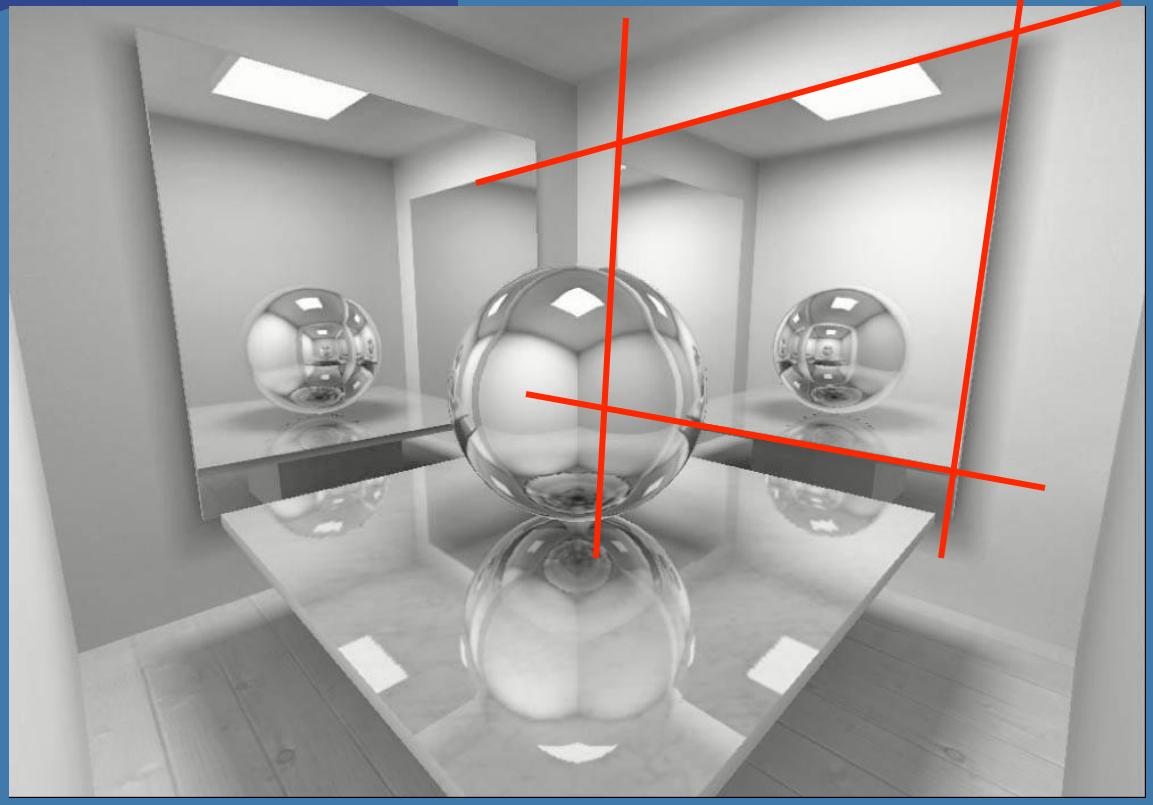


Planar reflections



- How should you render?
- 1) the reflective ground plane polygons 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



1. Render mirror to stencil buffer
2. Reflect camera (including cam axes)
3. Set user clip plane in mirror plane to cull anything between mirror and reflected camera
4. Render scene to screen.

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

Image courtesy of Kasper Hoey Nielsen

Study Questions

- What is “Planar shadows”
  - Answer: you project the objects’ triangles onto the plane and draw them with dark color.
- Explain shadow maps
- Explain shadow volumes
  - Both z-pass and z-fail
- What are the pros and cons of shadow maps vs. shadow volumes?
- How can you render planar reflections?