Visibility and Shadows

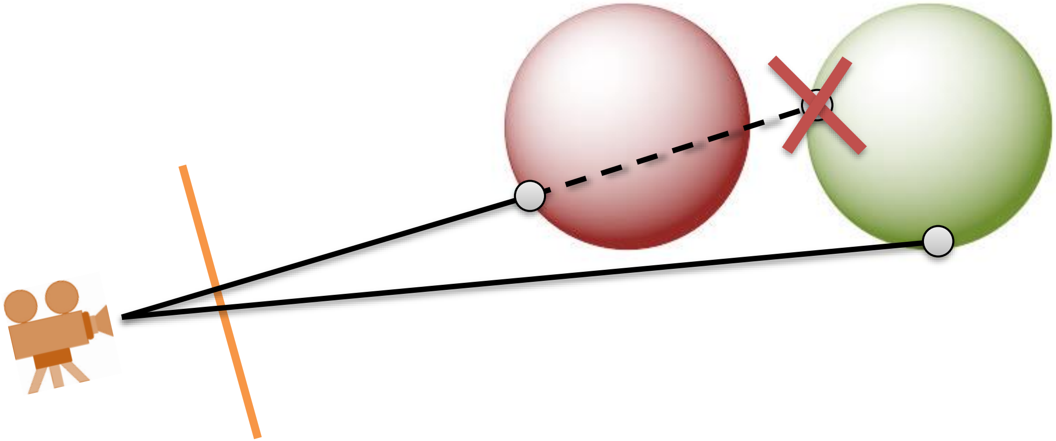
Prof. Dr. Markus Gross

ETH zürich

cgl

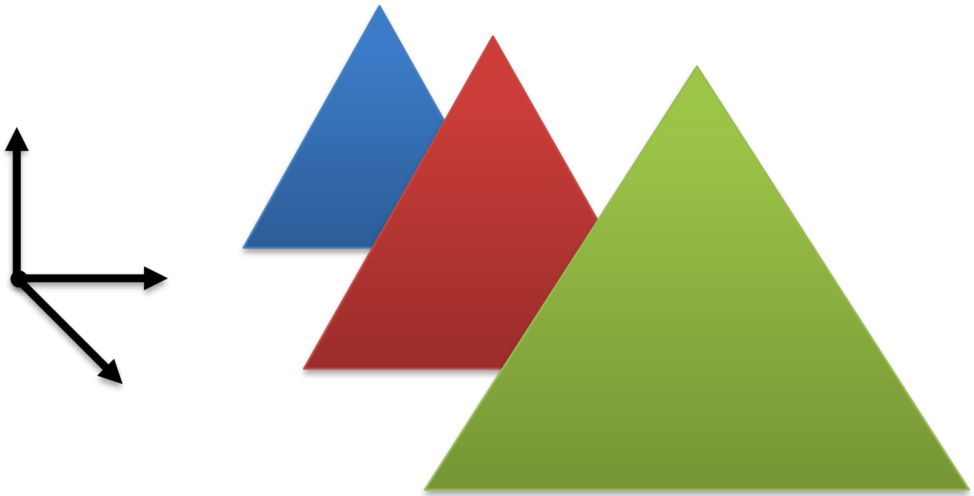
Visibility

- The visibility problem
  - Some parts of some surfaces are occluded



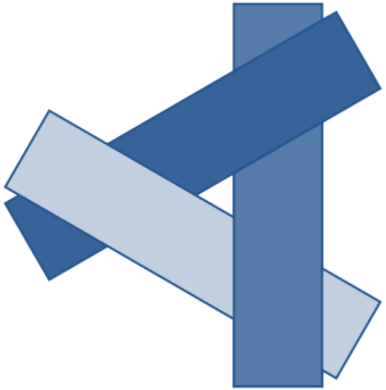
Visibility

- Solution 1: Painter's algorithm
  - Render objects/polygons from furthest to nearest

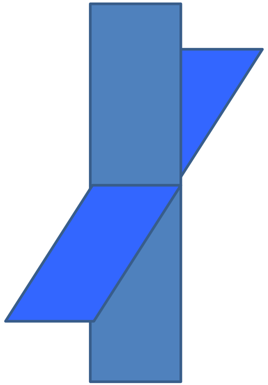


Visibility

- Solution 1: Painter’s algorithm
  - Problems



Cyclic Overlaps



Intersections

Visibility

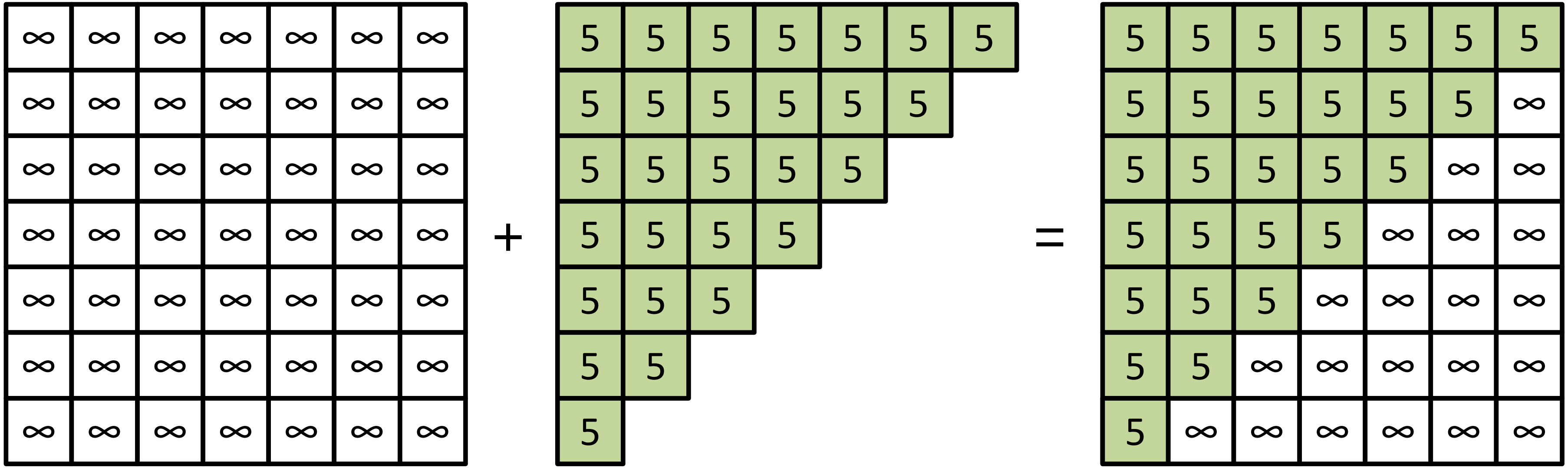
- Solution 2: Z-Buffering
  - Store depth to the nearest object for each pixel

Visibility

- Solution 2: Z-Buffering - algorithm
  1. Initialize all z values to ∞
  2. For each polygon
     - If z value of a pixel for this polygon is smaller than the stored z value, replace the stored z value

Visibility

- Solution 2: Z-Buffering - algorithm



∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞

+

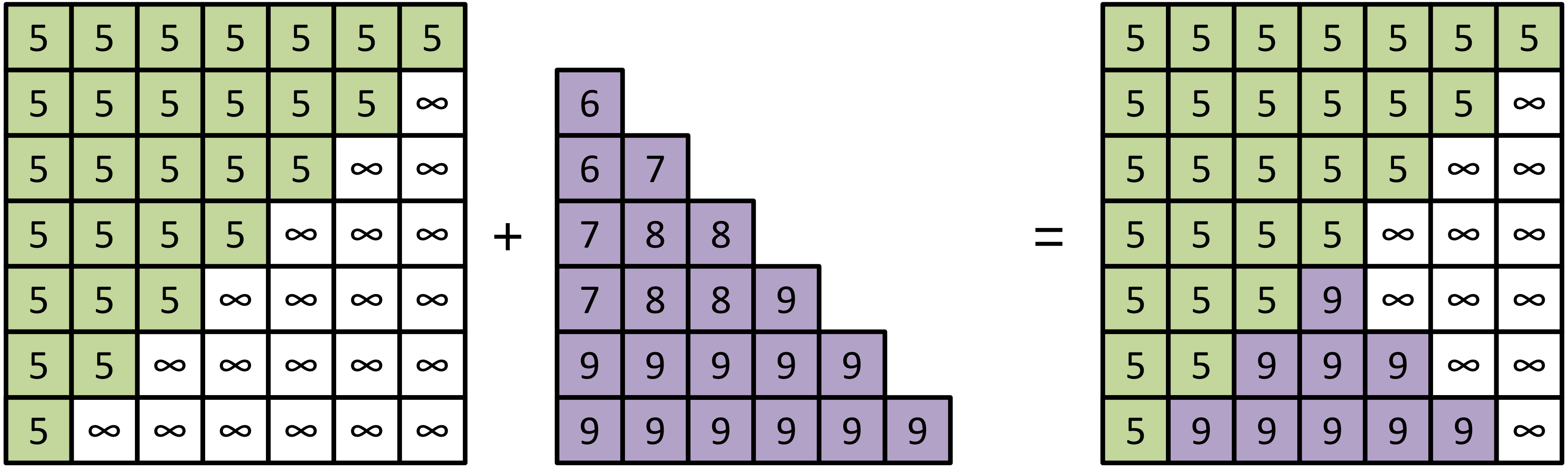
5	5	5	5	5	5	5
5	5	5	5	5	5	
5	5	5	5	5		
5	5	5	5			
5	5	5				
5	5					
5						

=

5	5	5	5	5	5	5
5	5	5	5	5	5	∞
5	5	5	5	5	∞	∞
5	5	5	5	∞	∞	∞
5	5	5	∞	∞	∞	∞
5	5	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞

Visibility

- Solution 2: Z-Buffering - algorithm



5	5	5	5	5	5	5
5	5	5	5	5	5	∞
5	5	5	5	5	∞	∞
5	5	5	5	∞	∞	∞
5	5	5	∞	∞	∞	∞
5	5	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞

+

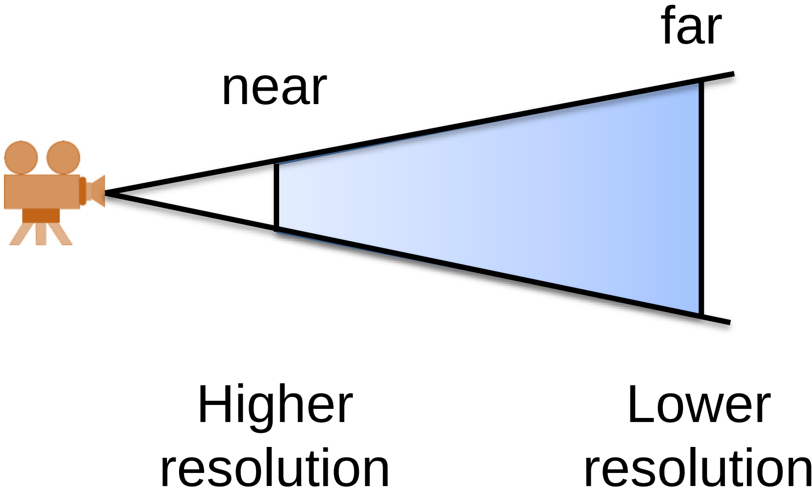
6					
6	7				
7	8	8			
7	8	8	9		
9	9	9	9	9	
9	9	9	9	9	9

=

5	5	5	5	5	5	5
5	5	5	5	5	5	∞
5	5	5	5	5	∞	∞
5	5	5	5	∞	∞	∞
5	5	5	9	∞	∞	∞
5	5	9	9	9	∞	∞
5	9	9	9	9	9	∞

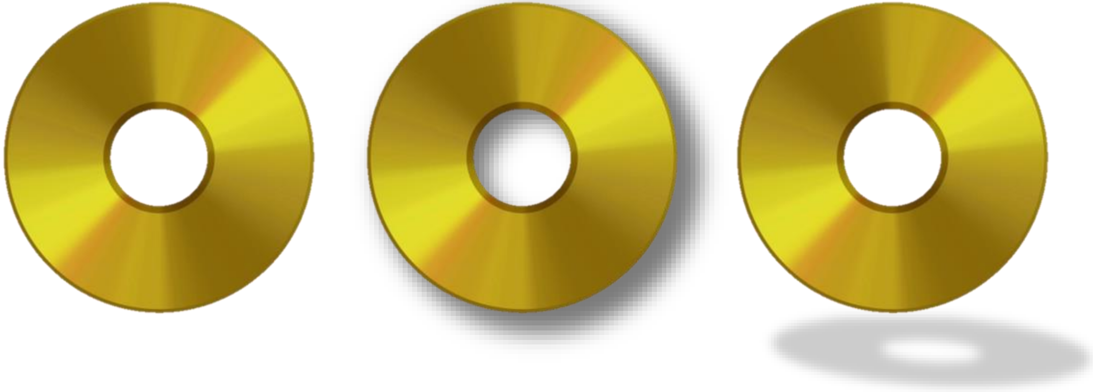
Visibility

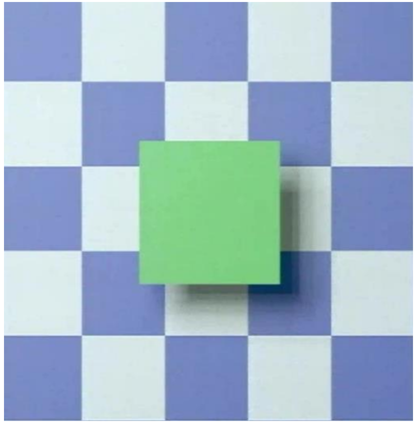
- Solution 2: Z-Buffering
  - Problem: limited resolution
  - Resolution is non-linear
  - Set near plane far from the camera



Shadows

- Why are shadows important?
  - Depth cue



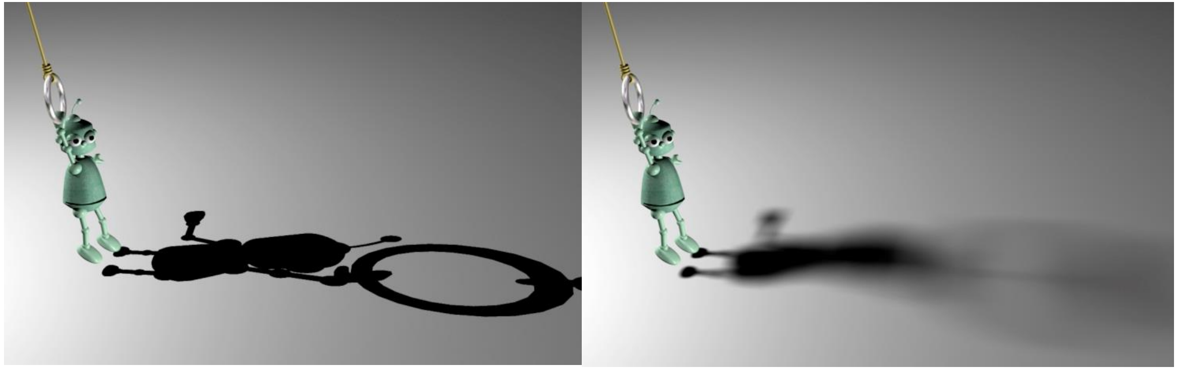


Shadows

- Why are shadows important?
  - Scene lighting

Light Position

Point vs. Area Light

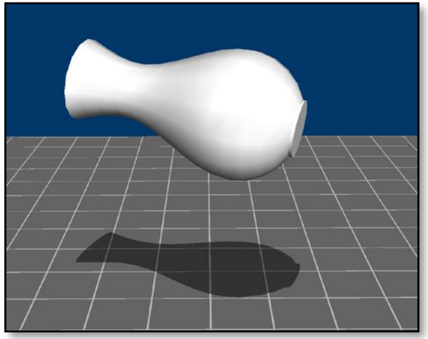


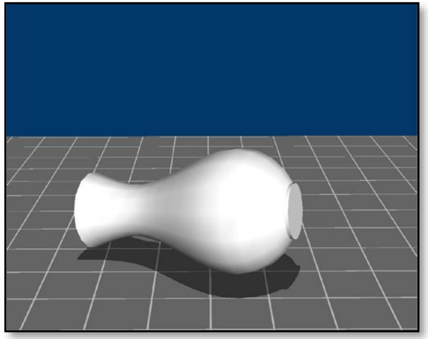
Shadows

- Why are shadows important?
  - – Realism

Basic Shadows

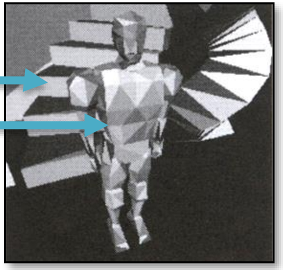
- Planar shadows
  - Draw projection of the object on the ground
  - Limitations
    - Self shadows
    - Shadows on other objects
    - Curved surfaces

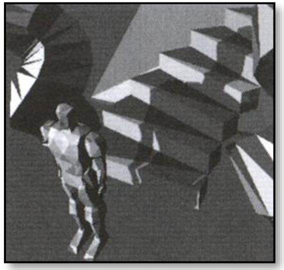




Basic Shadows

- Projective texture shadows
  - Separate obstacle and receiver
  - Compute b/w image of the obstacle from light
  - Use image as projective texture
  - Limitations
    - Need to specify obstacle & receiver
    - No self-shadows



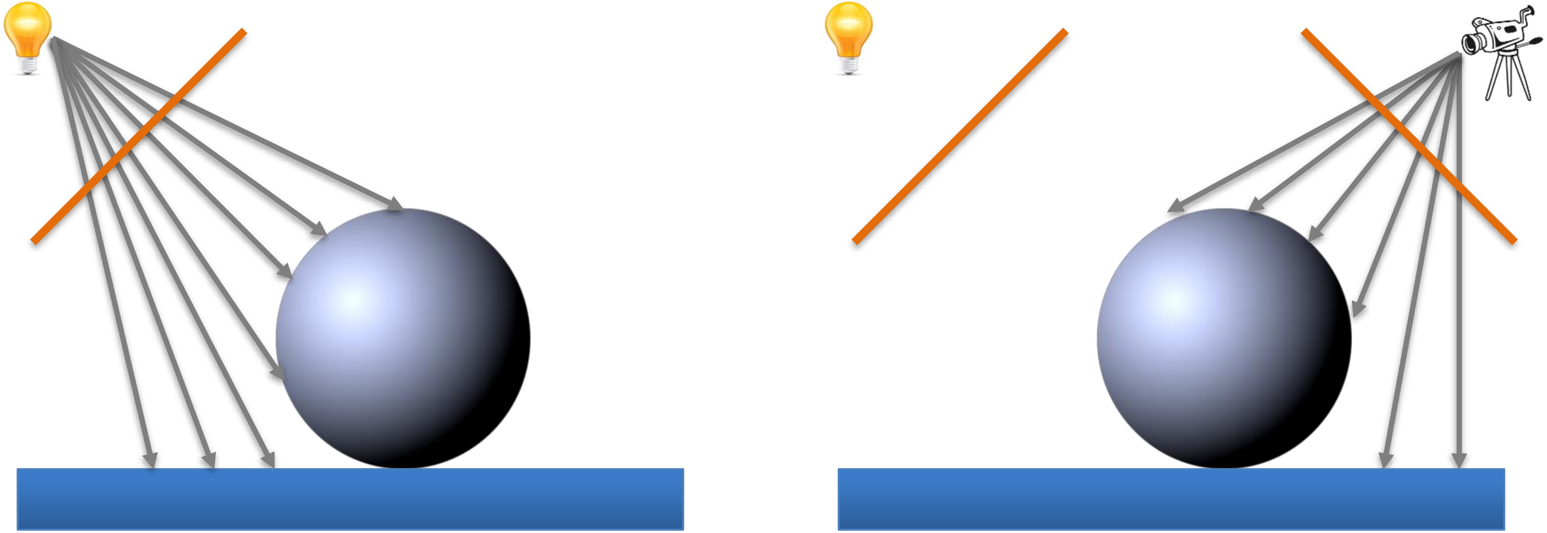


Shadow Maps

- In high-end production software and games

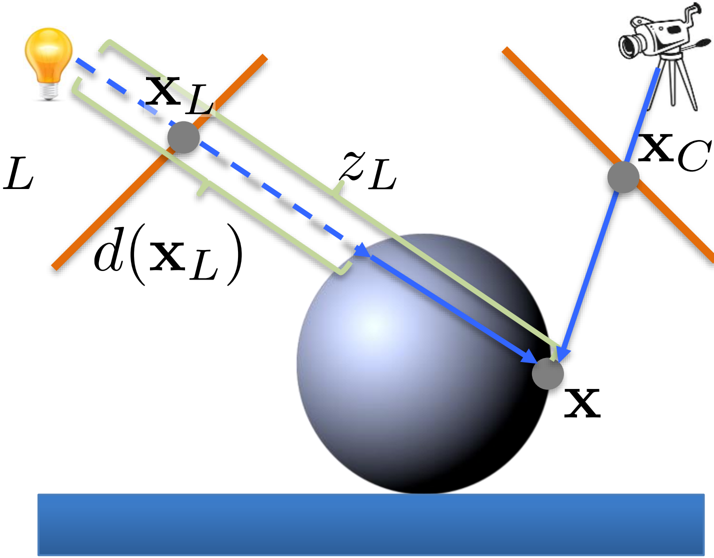
Shadow Maps

- Compute the depths from the light
- Compute the depths from the camera



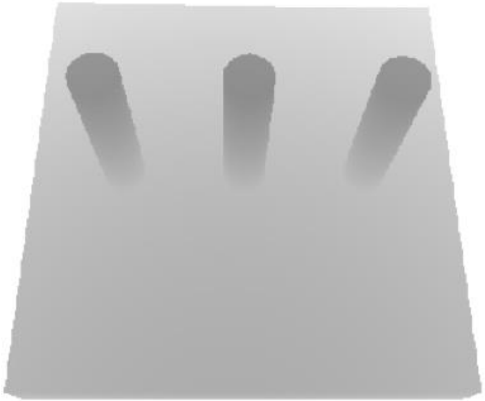
Shadow Maps

- For each pixel on the camera plane
  - – Compute the point in world coordinates
  - – Project point onto the light plane
  - – Compare $d(\mathbf{x}_L)$ (shadow map) and z_L
  - – If $d(\mathbf{x}_L) < z_L$, $\mathbf{x}$ is in shadow

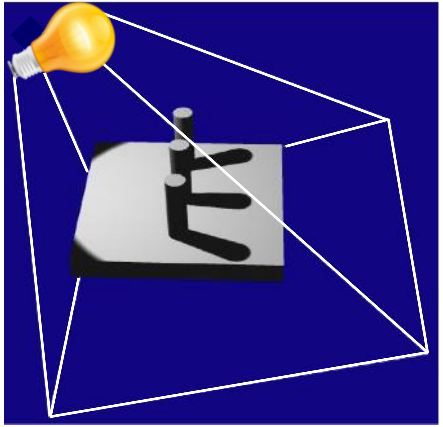


Shadow Maps

Depth map rendered
from the light



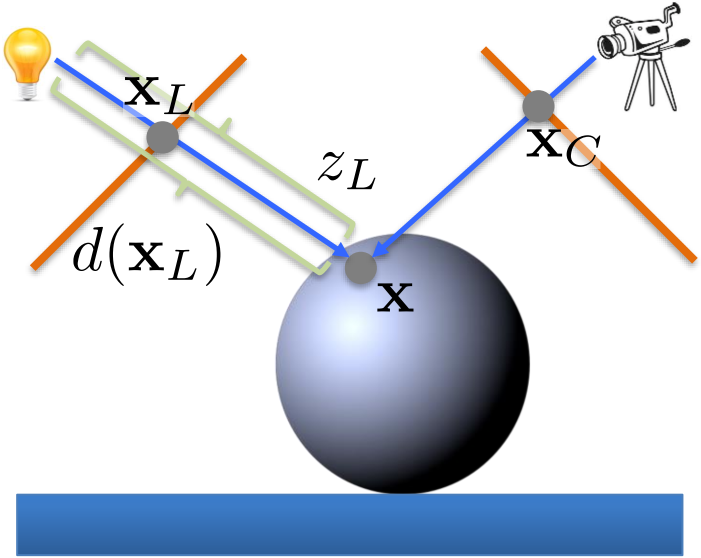
Rendering
from the camera



Shadow Maps

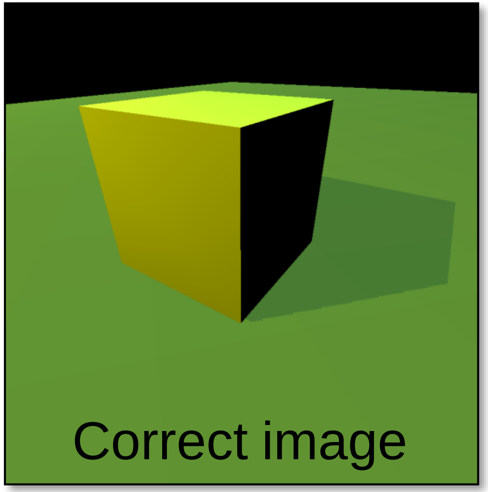
- Limitations – Bias
  - For a visible point $d(\mathbf{x}_L) < z_L$
  - How to avoid self-shadowing?
  - Add bias

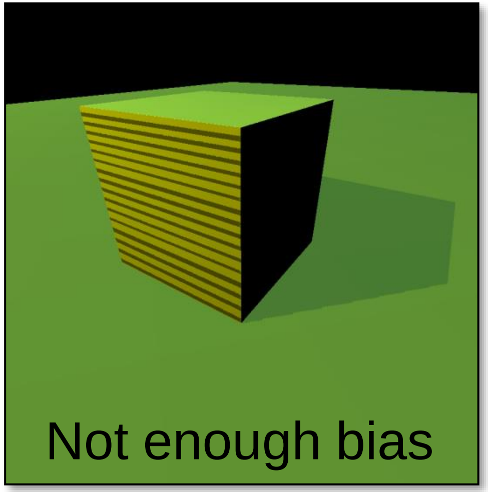
$$d(\mathbf{x}_L) + bias < z_L$$

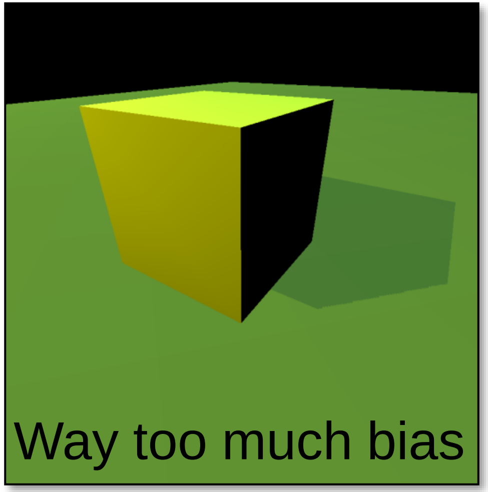


Shadow Maps

- Limitations – Bias $d(\mathbf{x}_L) + bias < z_L$
  - Choosing a good bias can be very tricky

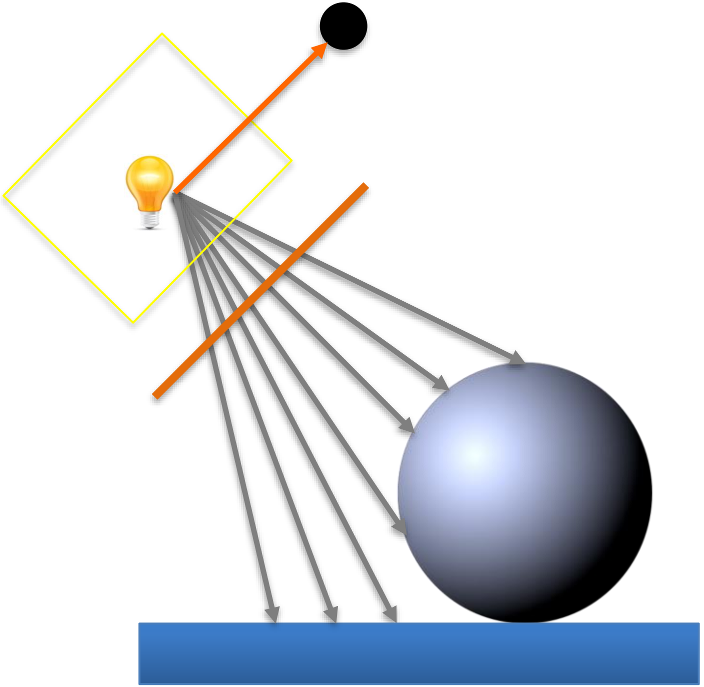

Correct image


Not enough bias


Way too much bias

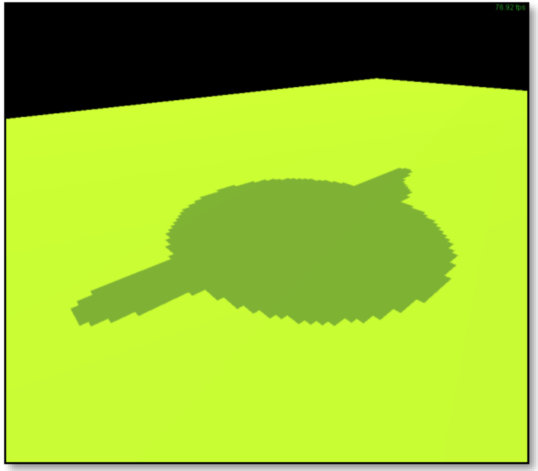
Shadow Maps

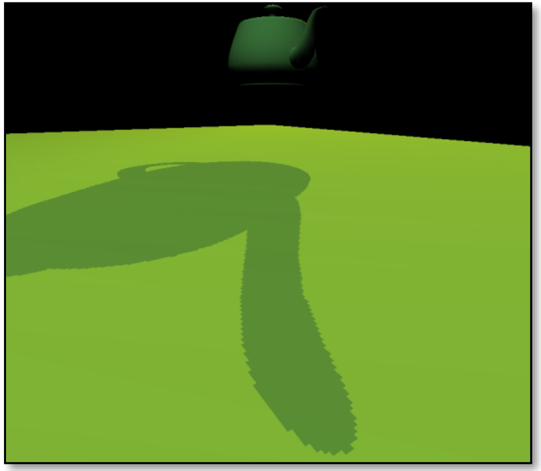
- Limitations – Field of view
  - A point to shadow can be outside the field of view of shadow map
  - Use cubical shadow map or spot lights



Shadow Maps

- Limitations – Aliasing
  – Undersampling of the shadow map





Shadow Maps

- Filtering
  - Should we filter depth? No.
  - Instead, filter the result of the test

$$d(\mathbf{x}_L) + bias < z_L$$

  - Take a weighted average of comparisons

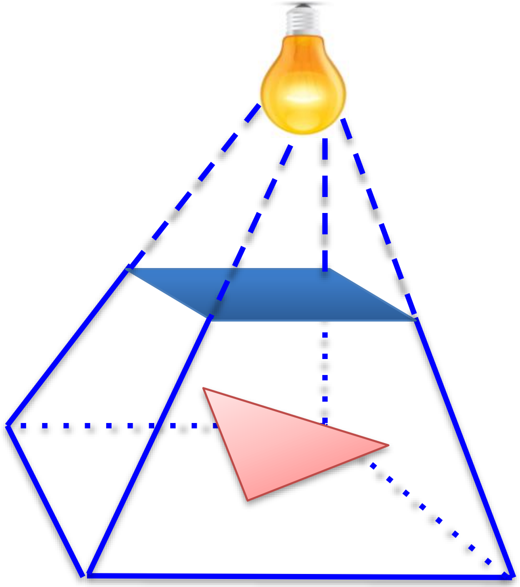
Shadow Maps

- Filtering
  - – Take a weighted average of comparisons
  - – Bigger filter produces fake soft shadows
  - – Setting bias is tricky

Shadow Volumes

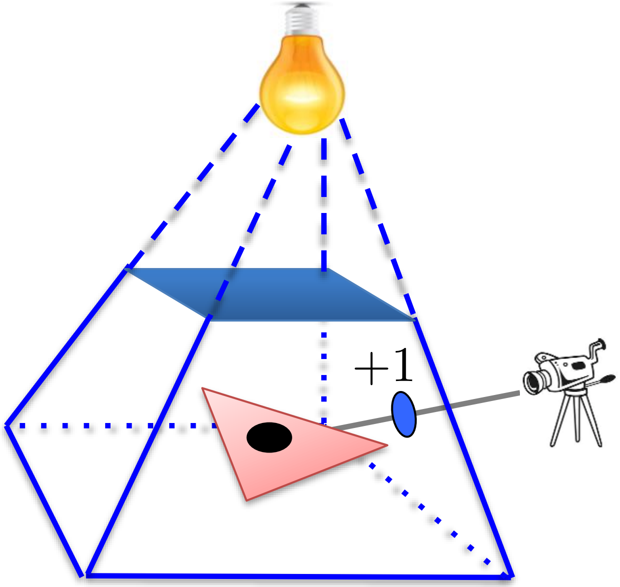
- Explicitly represent the volume of space in shadow
- If a polygon is inside the volume, it is in shadow
- Similar to clipping
- Naïve implementation:

$$O(\#polygons \times \#lights)$$



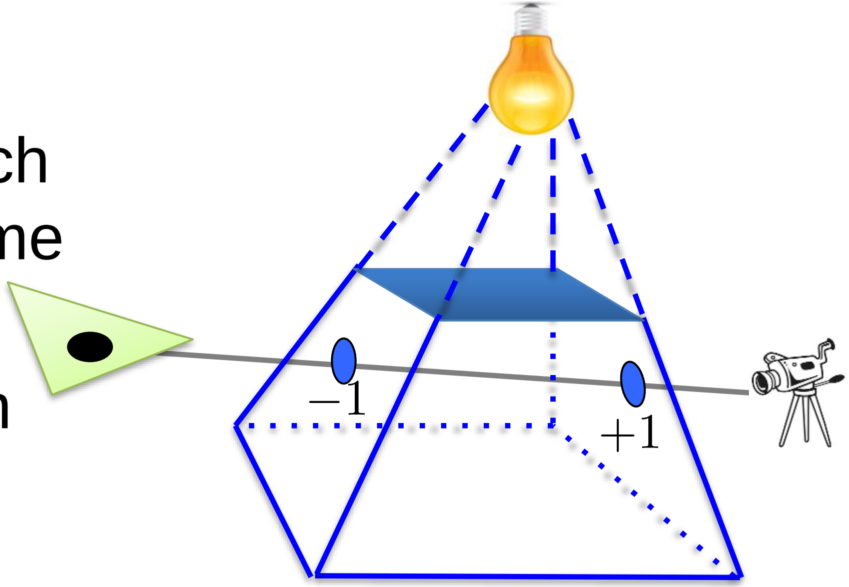
Shadow Volumes

- Algorithm
  - Shoot a ray from the eye
  - Incre-/decrement a counter each time boundary of shadow volume is intersected
  - If counter > 0, primitive is in shadow



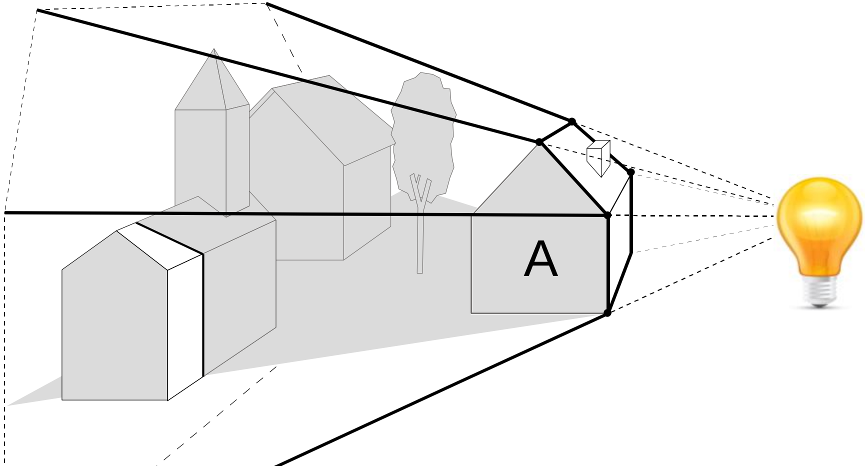
Shadow Volumes

- Algorithm
  - Shoot a ray from the eye
  - Incre-/decrement a counter each time boundary of shadow volume is intersected
  - If counter = 0, primitive is not in shadow



Shadow Volumes

- Optimization:
  - Use silhouette edges only
    (where a back-facing & front-facing polygon meet)



Shadow Volumes

- Limitations
  - Introduces a lot of new geometry
  - Expensive to rasterize long skinny triangles
  - Objects must be watertight to use the silhouette optimization
  - Rasterization of polygons sharing an edge must not overlap & not have gap

Comparisons

Features/Limitations	Planar Fake Shadows	Projective Texture Shadows	Shadow Maps	Shadow Volumes
Allows objects to cast shadows on themselves (self-shadowing)				
Permits shadows on arbitrary surfaces (i.e. curved)				
Generates extra geometric primitives				
Limited resolution of intermediate representation can result in jaggy shadow artifacts				