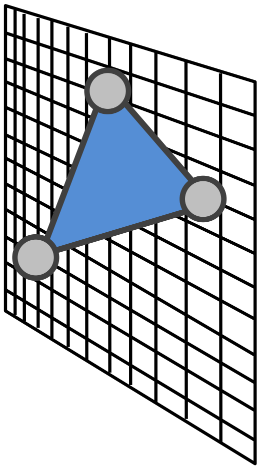
Computer Graphics (COMP0027) 2022/23

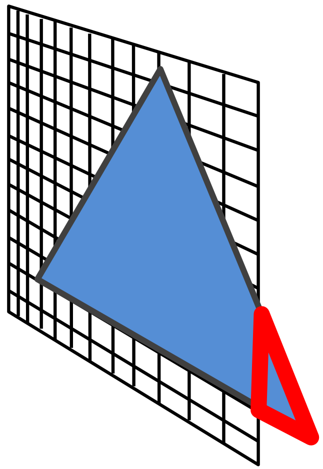
Shadows

Tobias Ritschel

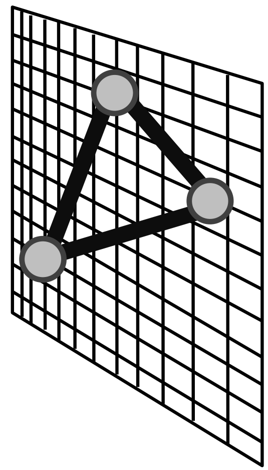
Challenges



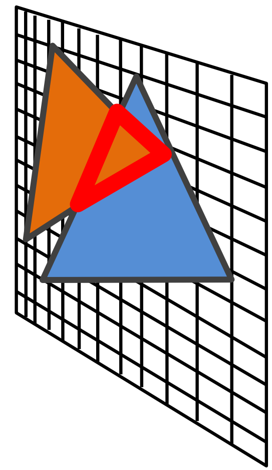
Projection



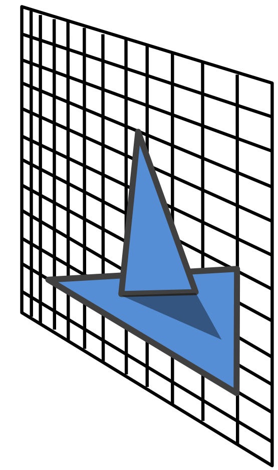
Clipping



Rasterization

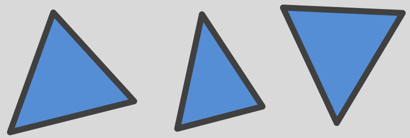


Visibility



Shading

Pipeline



Primitives

Projection

Clipping

Culling

Rasterisation

z test

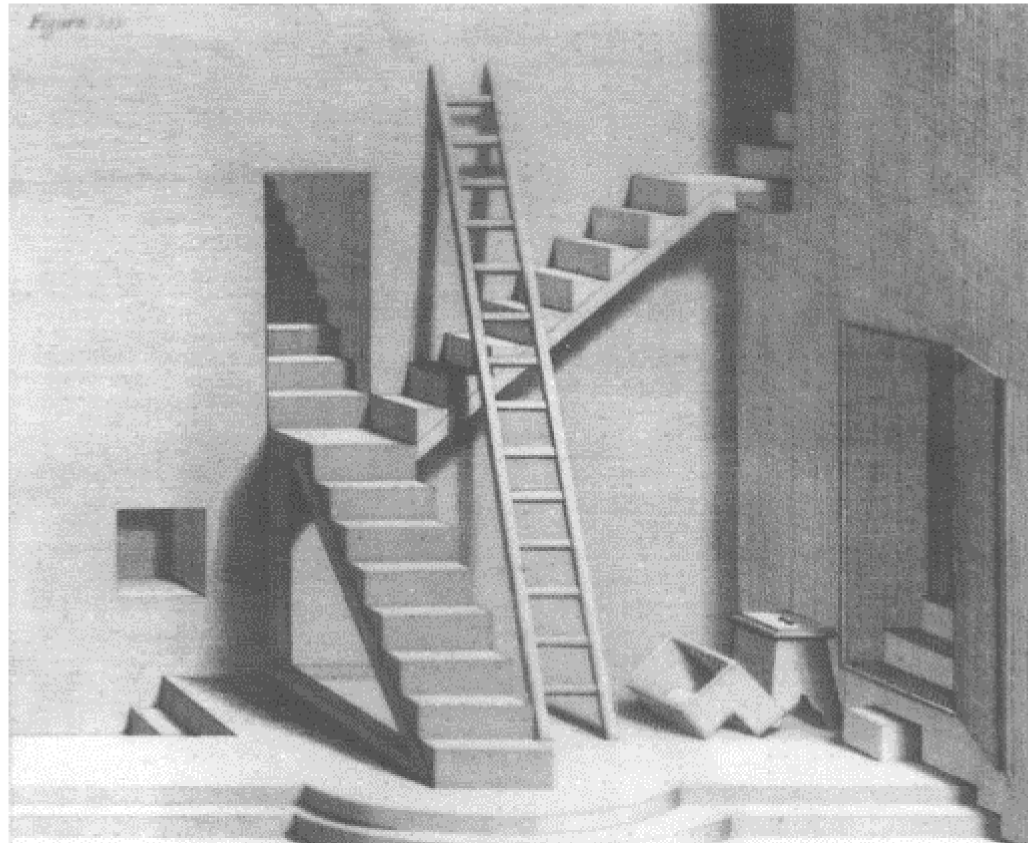
Shading

Outline

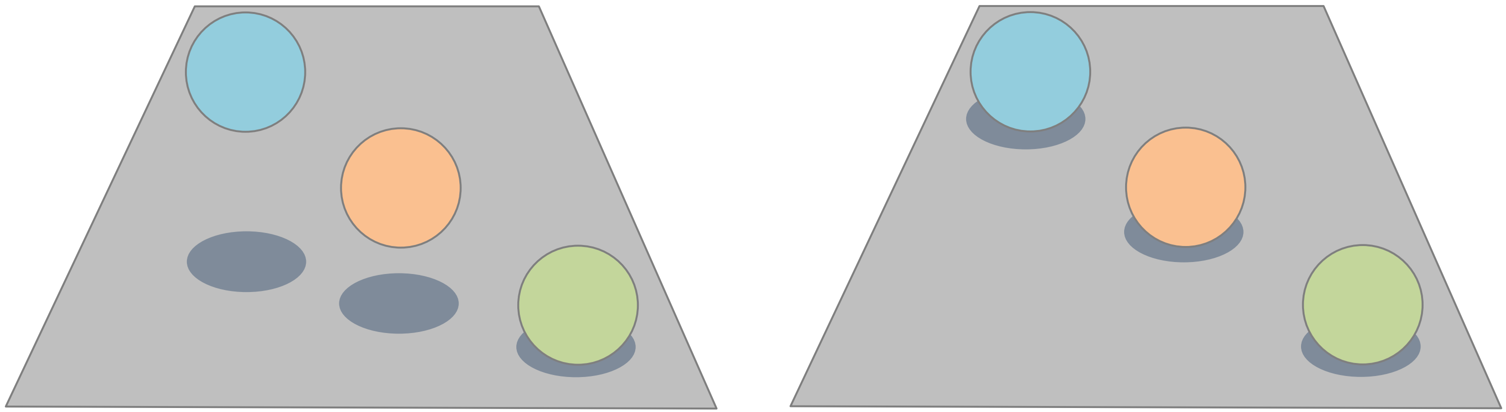
- Introduction
- Sharp shadows
- Soft shadows
- Conclusion

Why are Shadows Important?

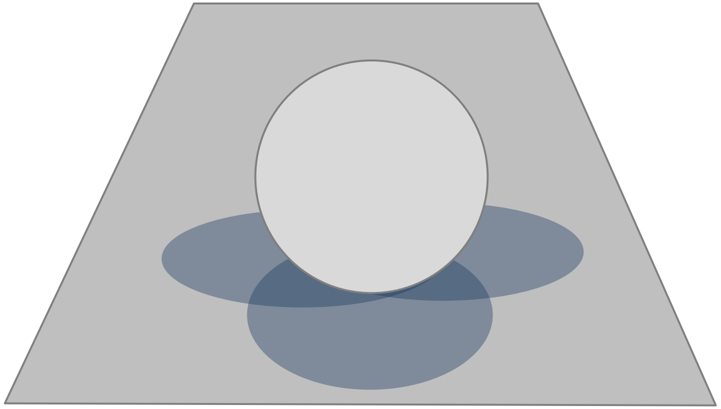
- Depth cue
- Scene Lighting
- Realism
- Contact points



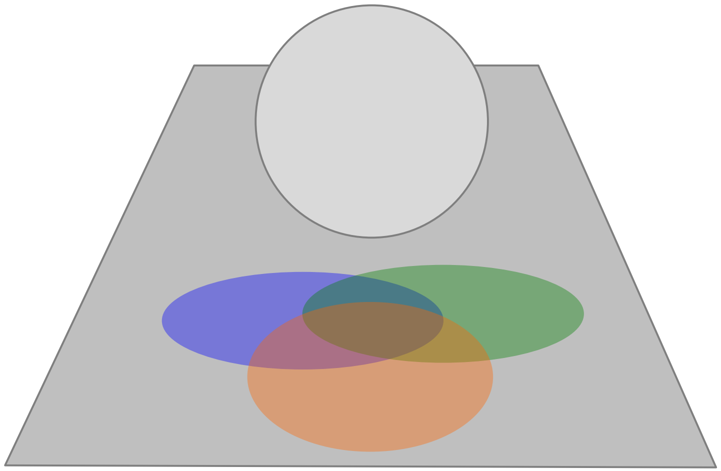
Spatial cue



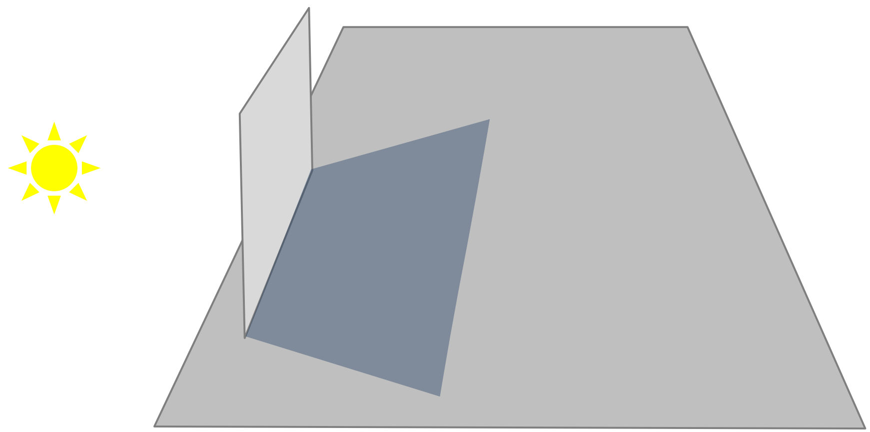
Light position cue (sundial)



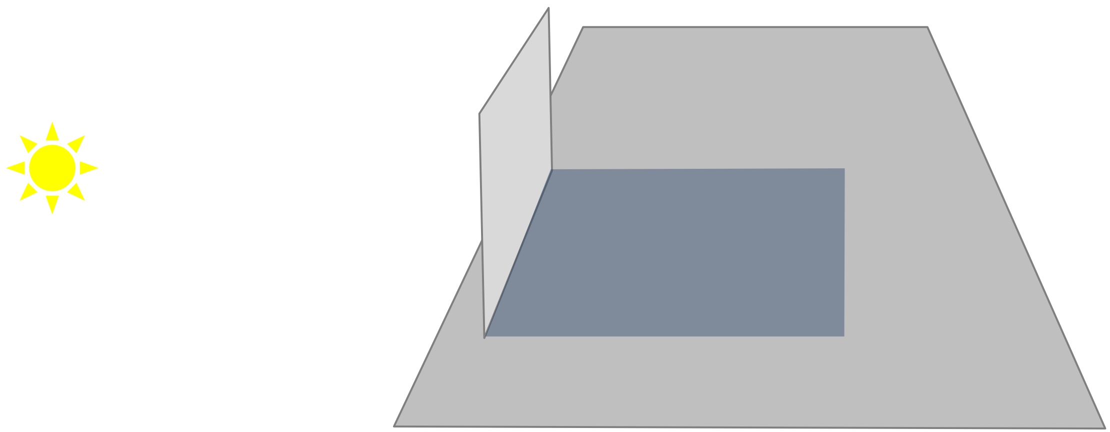
Light color



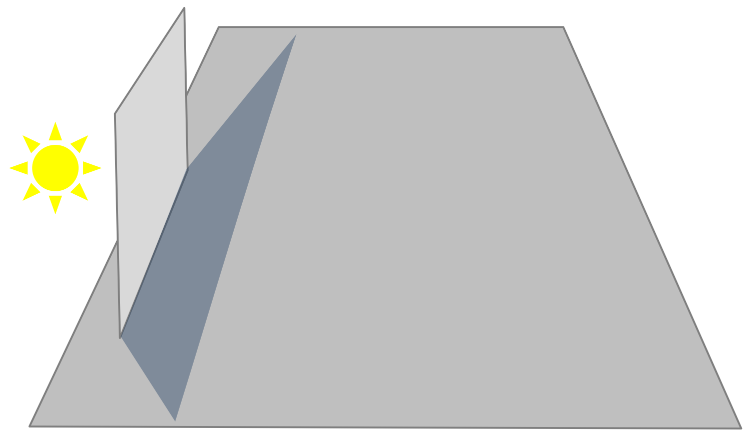
Light distance



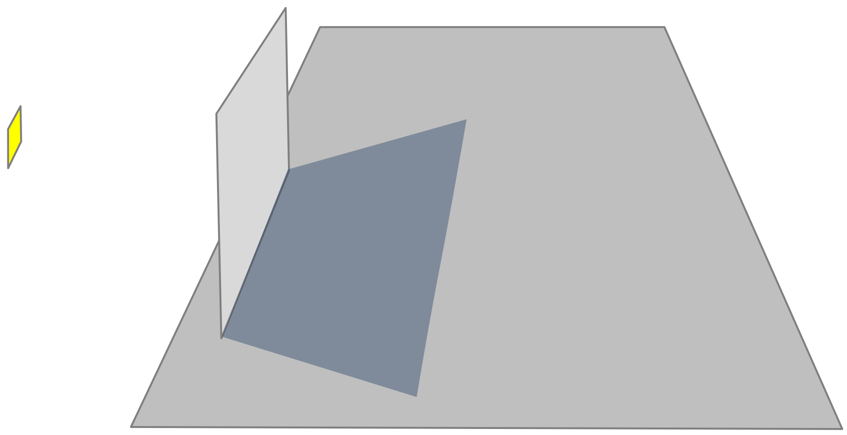
Light distance (far)



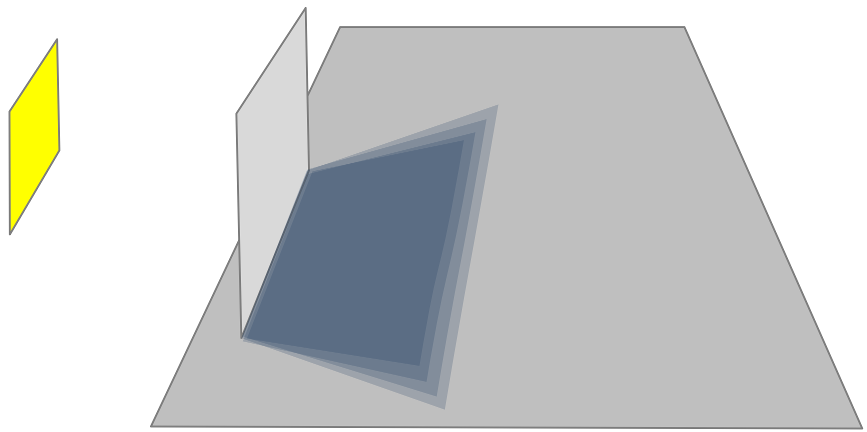
Light distance (near)



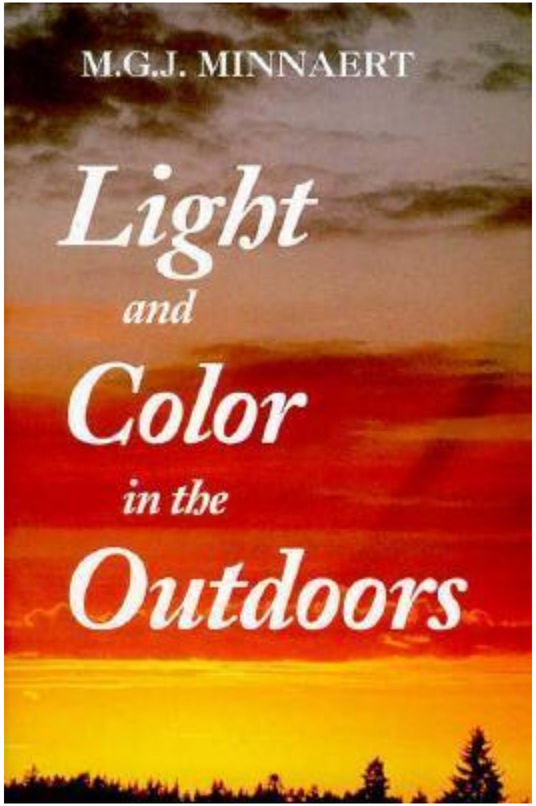
Light size (small)



Light size: large



Good book

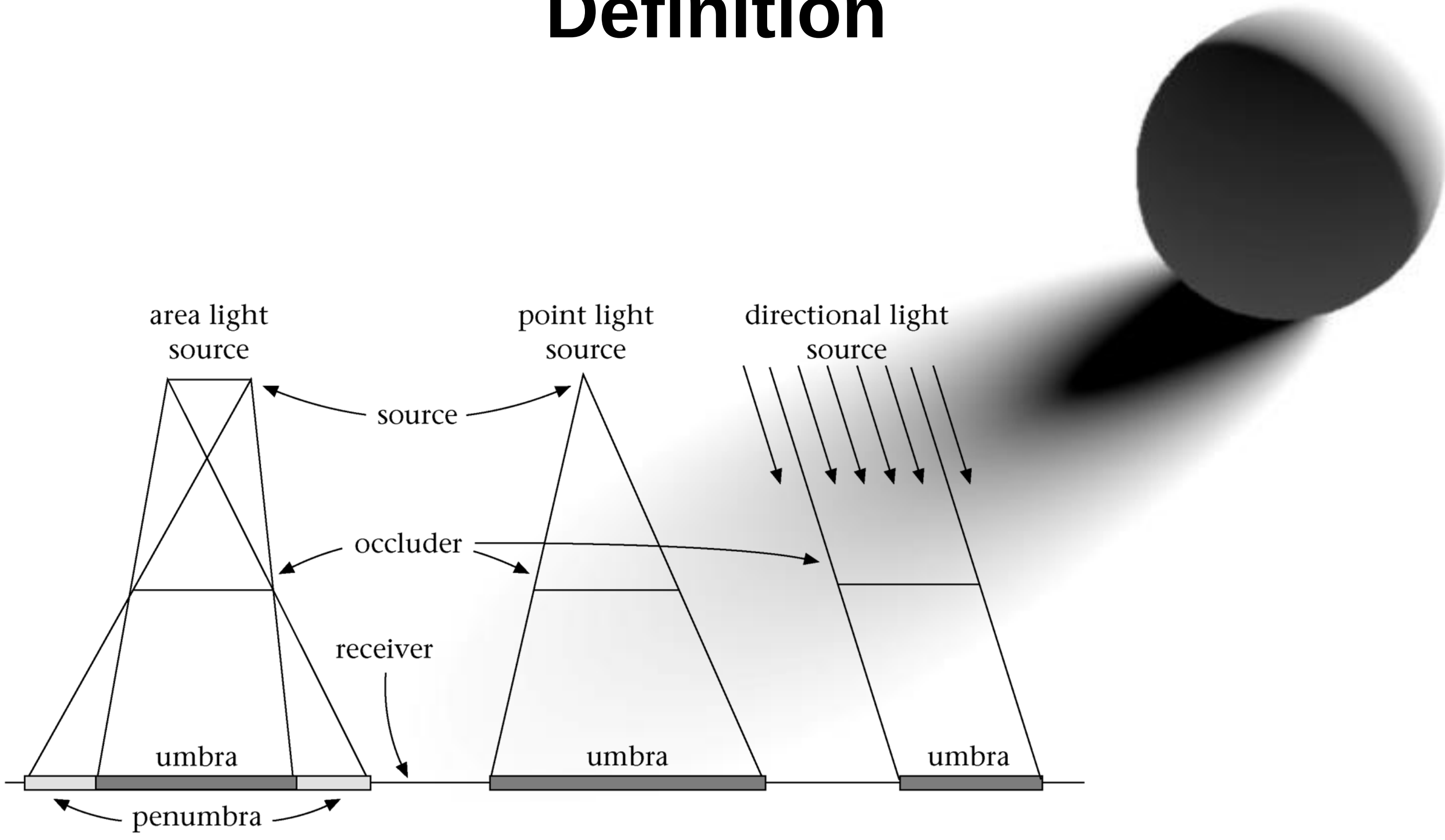


(Will not help you with passing this course)

Shadows are Complex

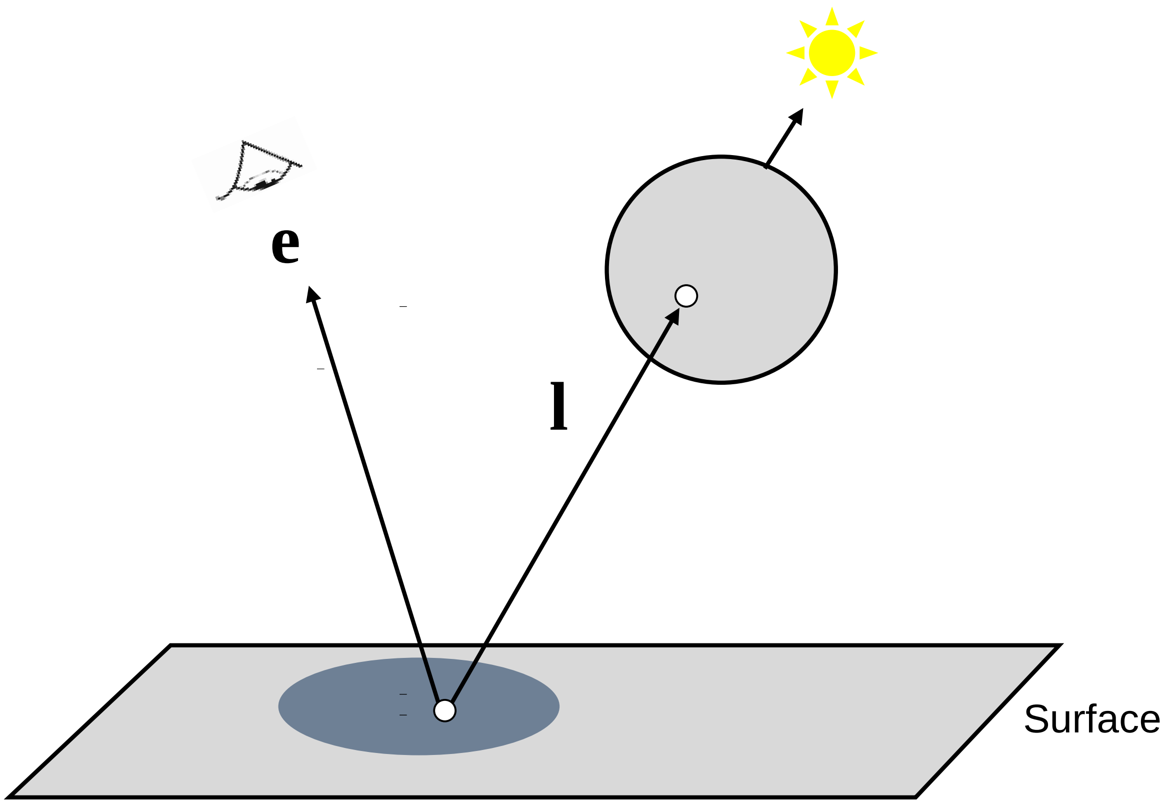
- In the real world sources of light are not points
- The intensity within a shadow is not constant
  - **Umbra**
    part that sees nothing of the source
  - **Penumbra**
    part that receives some light

Definition

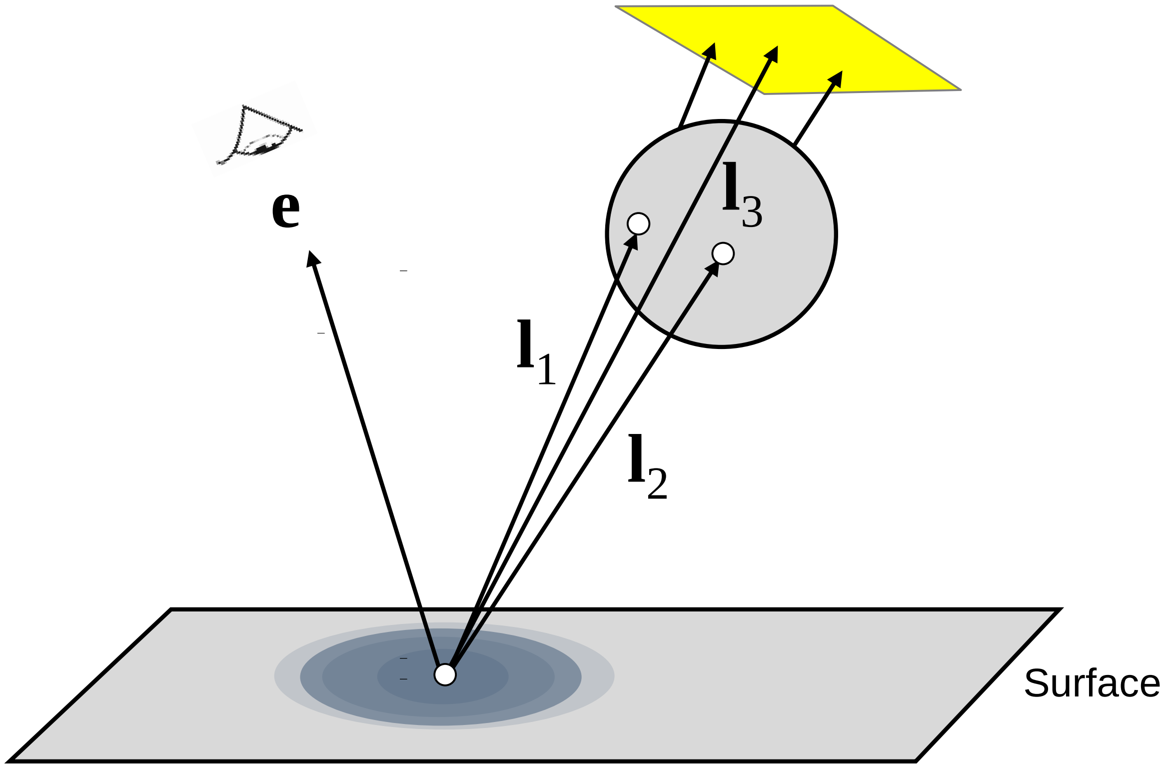


area light
source
point light
source
directional light
source
source
occluder
receiver
umbra
umbra
umbra
penumbra

Shadow in ray-tracing

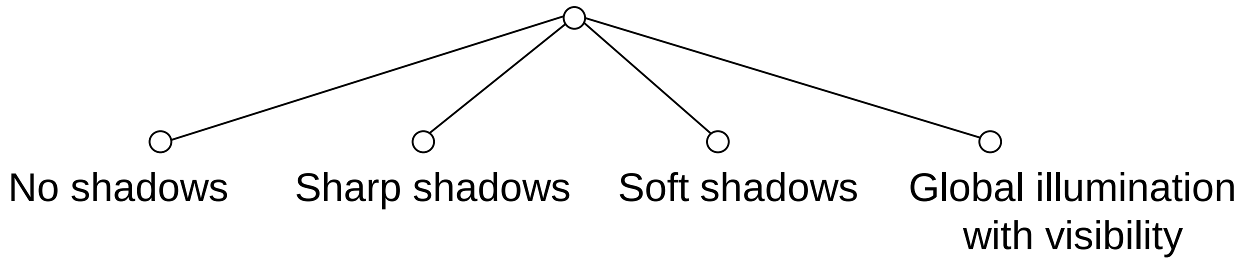


Soft shadows in ray-tracing



Current Shadowing Methods

- There exist a large number of methods
- We are interested in methods suitable for interactive walkthroughs, speed is crucial
- We will classify them by complexity:



Sharp shadows

Sharp Shadows

- Source is assumed to be a point or direction

Dynamic

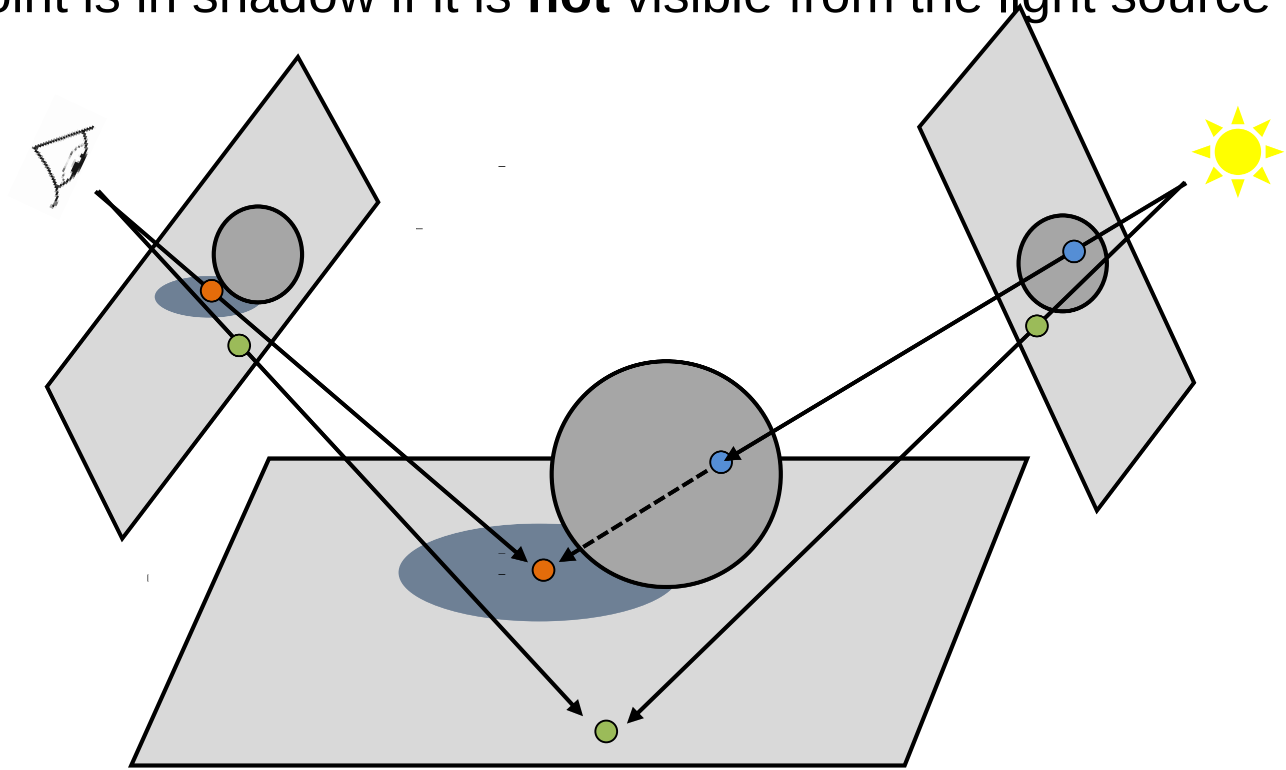
Shadow Maps
Shadow Volumes

Static

Light maps

Shadow Maps

Light / shadow duality:
A point is in shadow if it is **not** visible from the light source

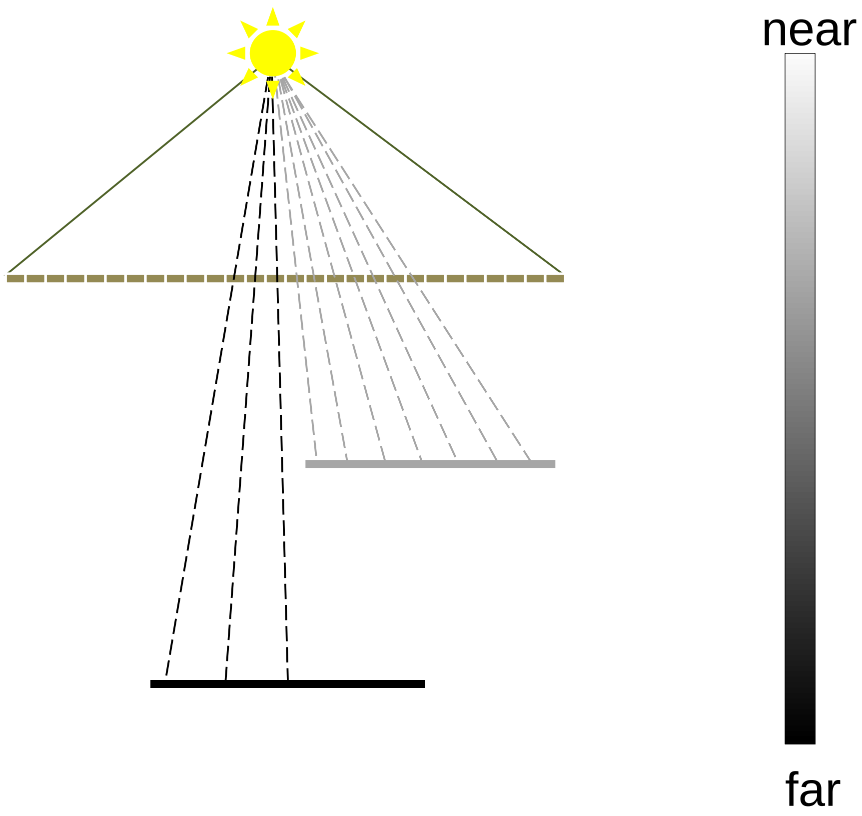


Two passes

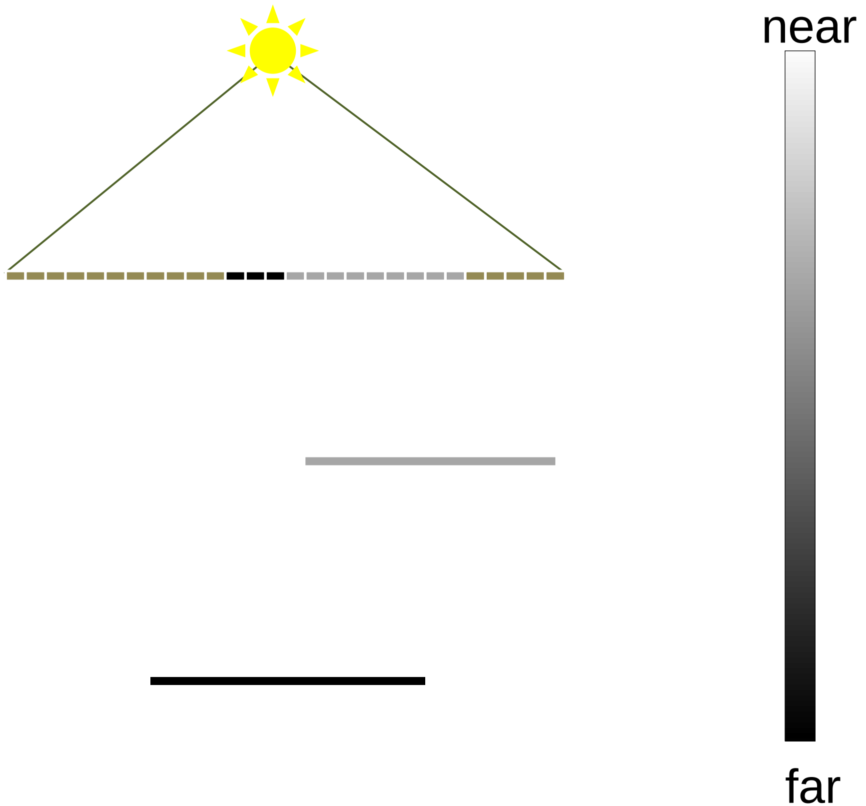
- Shadow map pass
  - Use the light source as a view point (light space)
  - Render scene
  - Store depth information in a shadow z-buffer = **shadow map**
- Shading pass
  - Render scene as usual form the cameras view point
  - Each pixel's position $(x_\mathrm{v}, y_\mathrm{v}, z_\mathrm{v})$ is transformed to light space $(x_\mathrm{s}, y_\mathrm{s}, z_\mathrm{s})$,
  - If
    - the z_s value is less or equal to the shadow map at x_s,y_s it is **lit**,
    - else it is **shadowed**

Shadow Maps

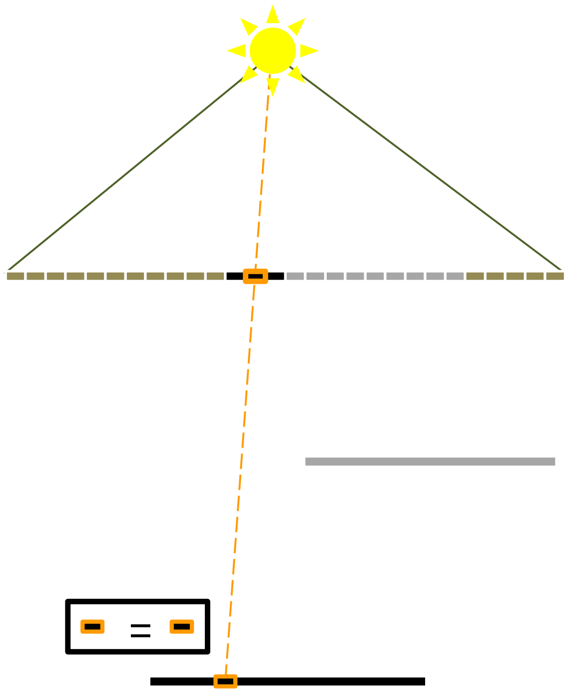
Shadow Maps

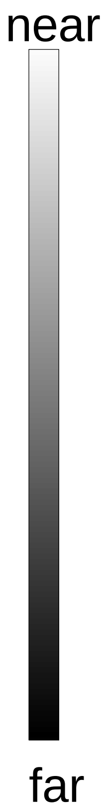


Shadow Maps

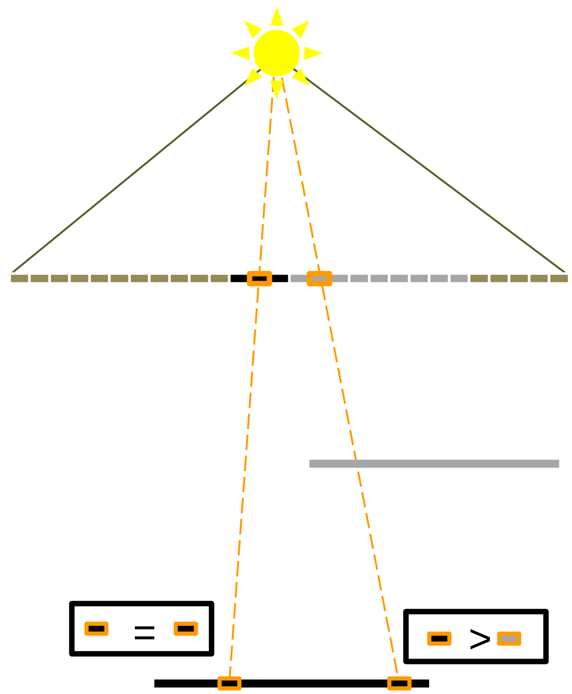


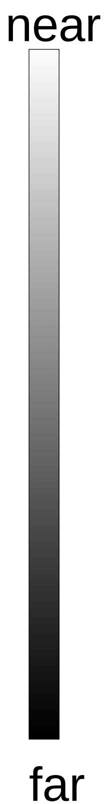
Shadow Maps





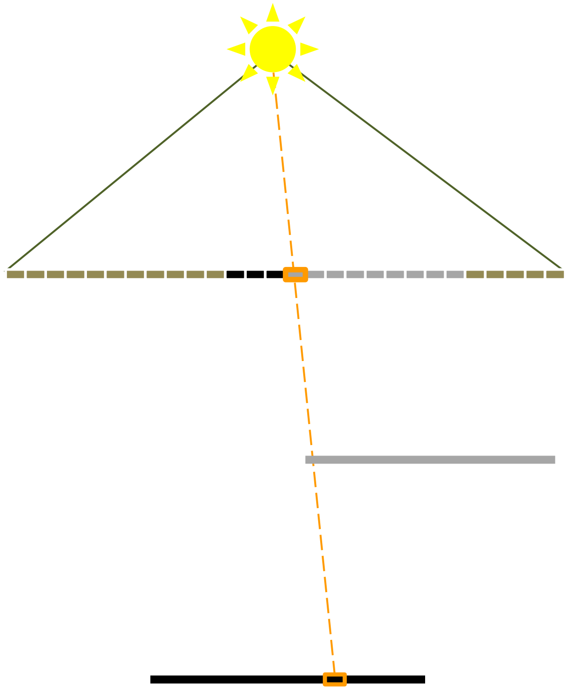
Shadow Maps



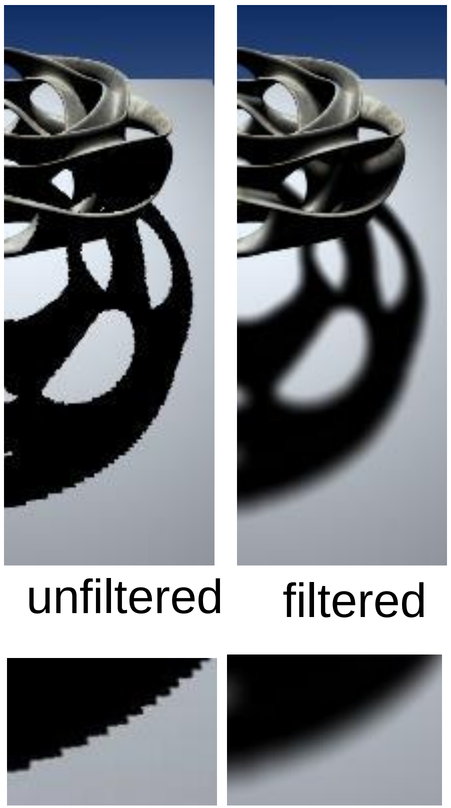


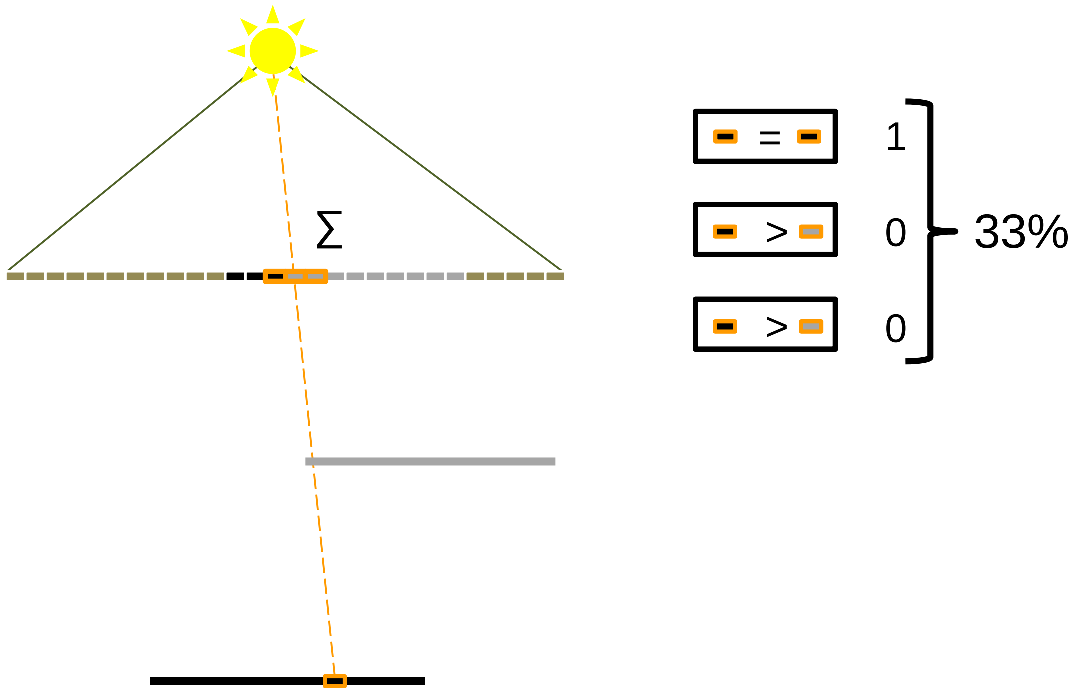
Shadow Map Filtering

unfiltered



Shadow Map Filtering

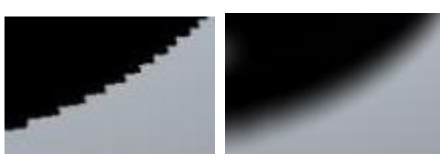


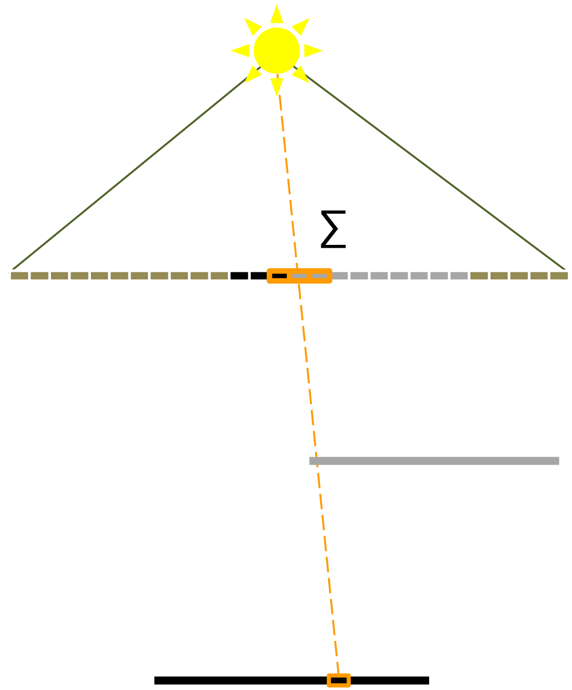


Percentage Closer Filtering
[Reeves et al. ‘87]

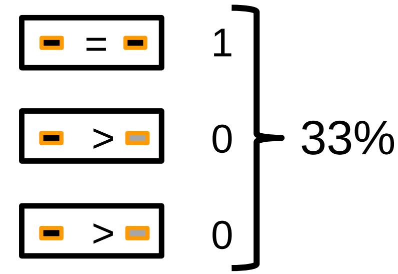
Shadow Map Filtering

unfiltered filtered





Percentage Closer Filtering
[Reeves et al. ‘87]

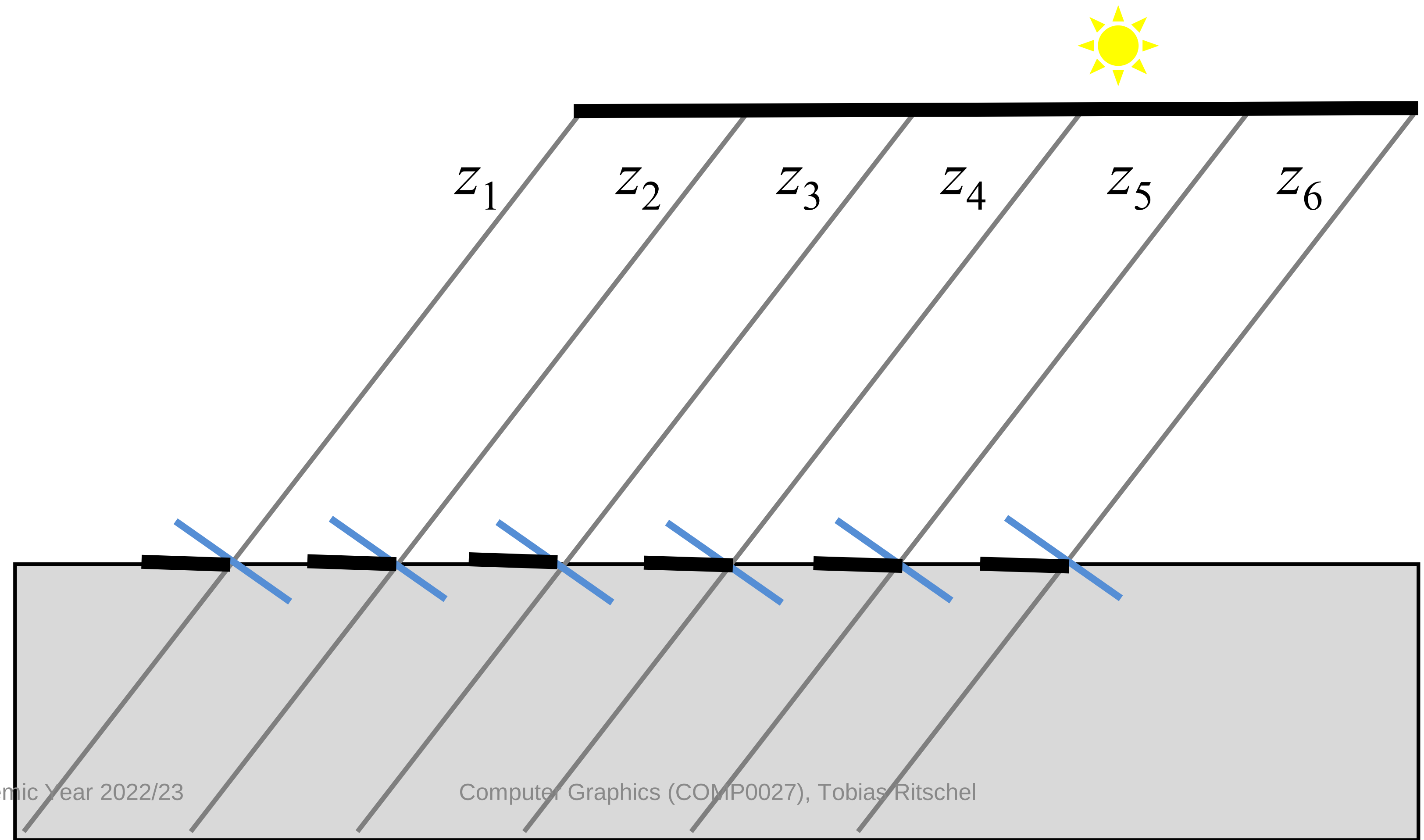


compare first, then filter!

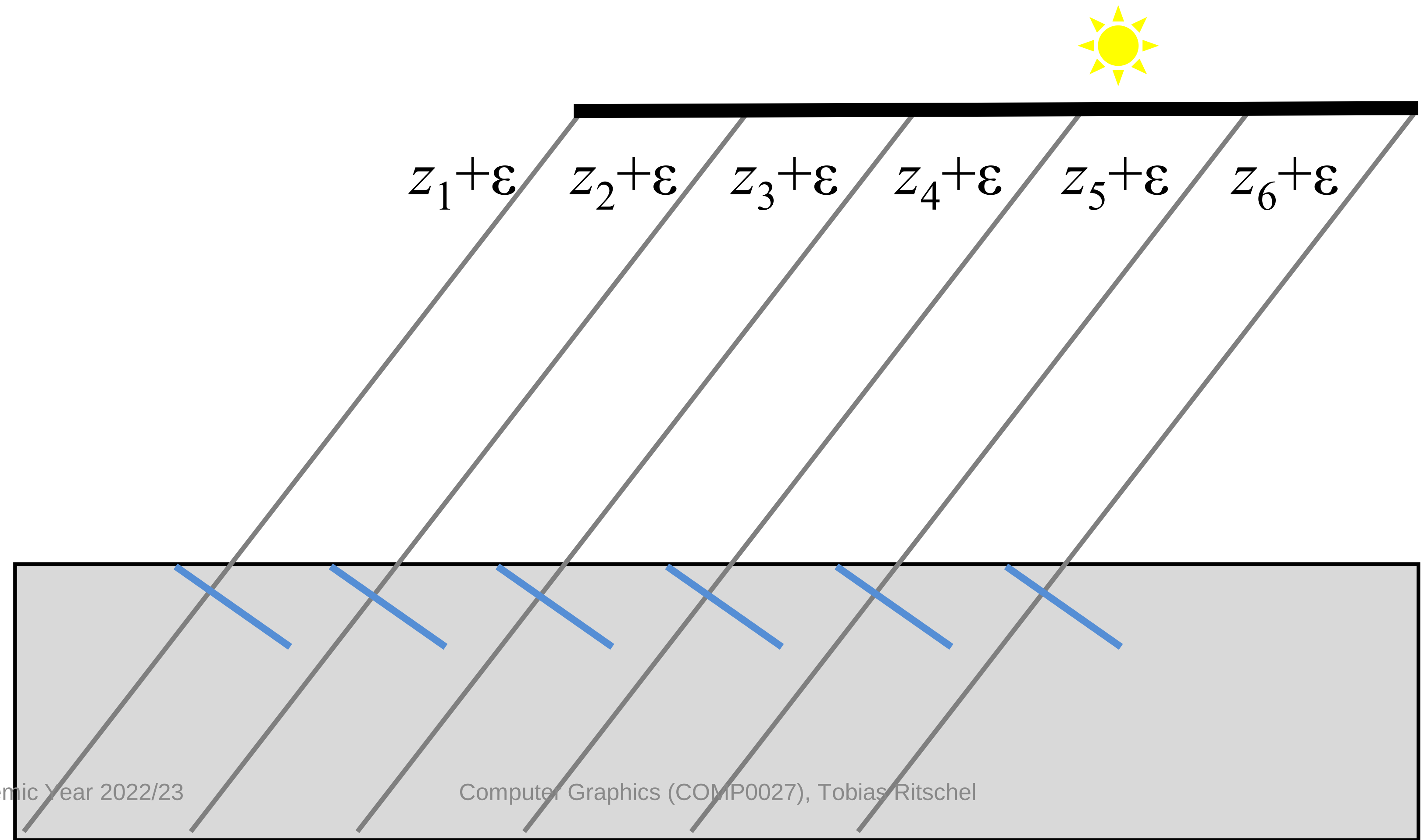
↓

Expensive

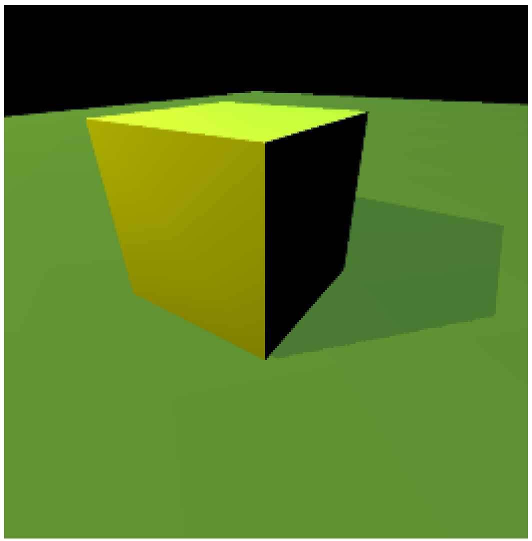
Self shadowing: Disaster



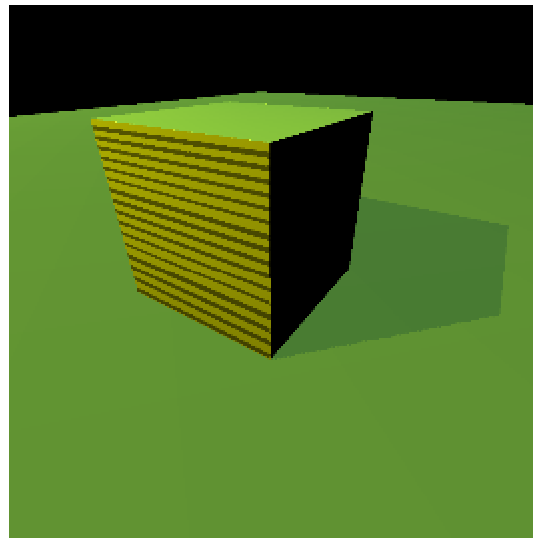
Self shadowing fix



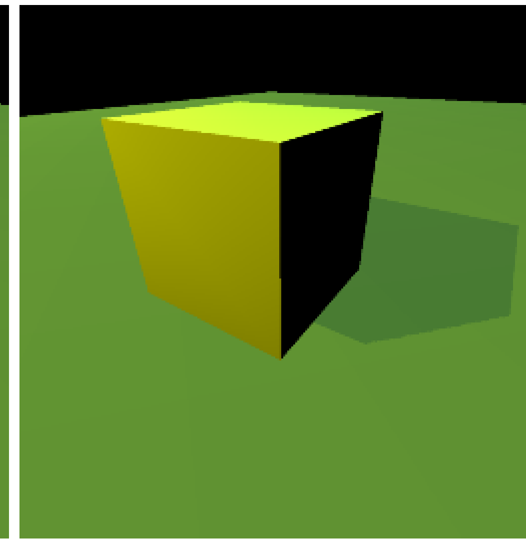
Bias (Epsilon) for Shadow Maps



Correct result



Not enough bias



Too much bias

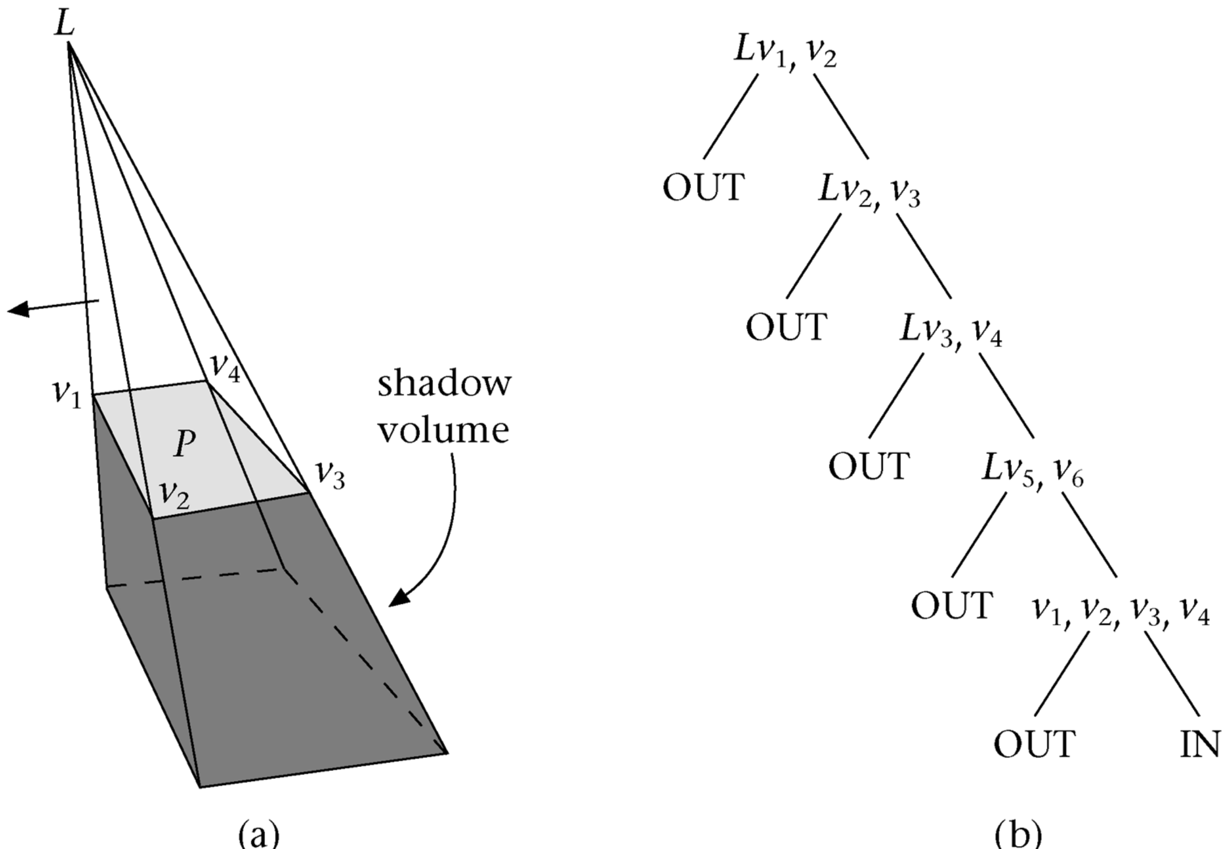
Shadow Maps

- “Less than or equal” test is imprecise
  - Gives rise to “shadow acne”
- Often found in hardware now
  - Otherwise high-cost operation
- Imprecise since it is only accurate in the image space of the light
  - Imagine a shadow throw over complex objects or long distances
- Quality depends on resolution (jagged edges)
  - Percentage-closer filtering helps
- FOV of shadow map?

Shadow Volume Method

- Shadow volume (SV) is the volume of space below a polygon that cannot see the source (a culled pyramid)
- During rendering of image, the line from a point visible through a pixel to the eye is intersected with all object SVs
- The number of intersections indicates if the point is in shadow or not

Shadow Volumes



Shadow Volumes

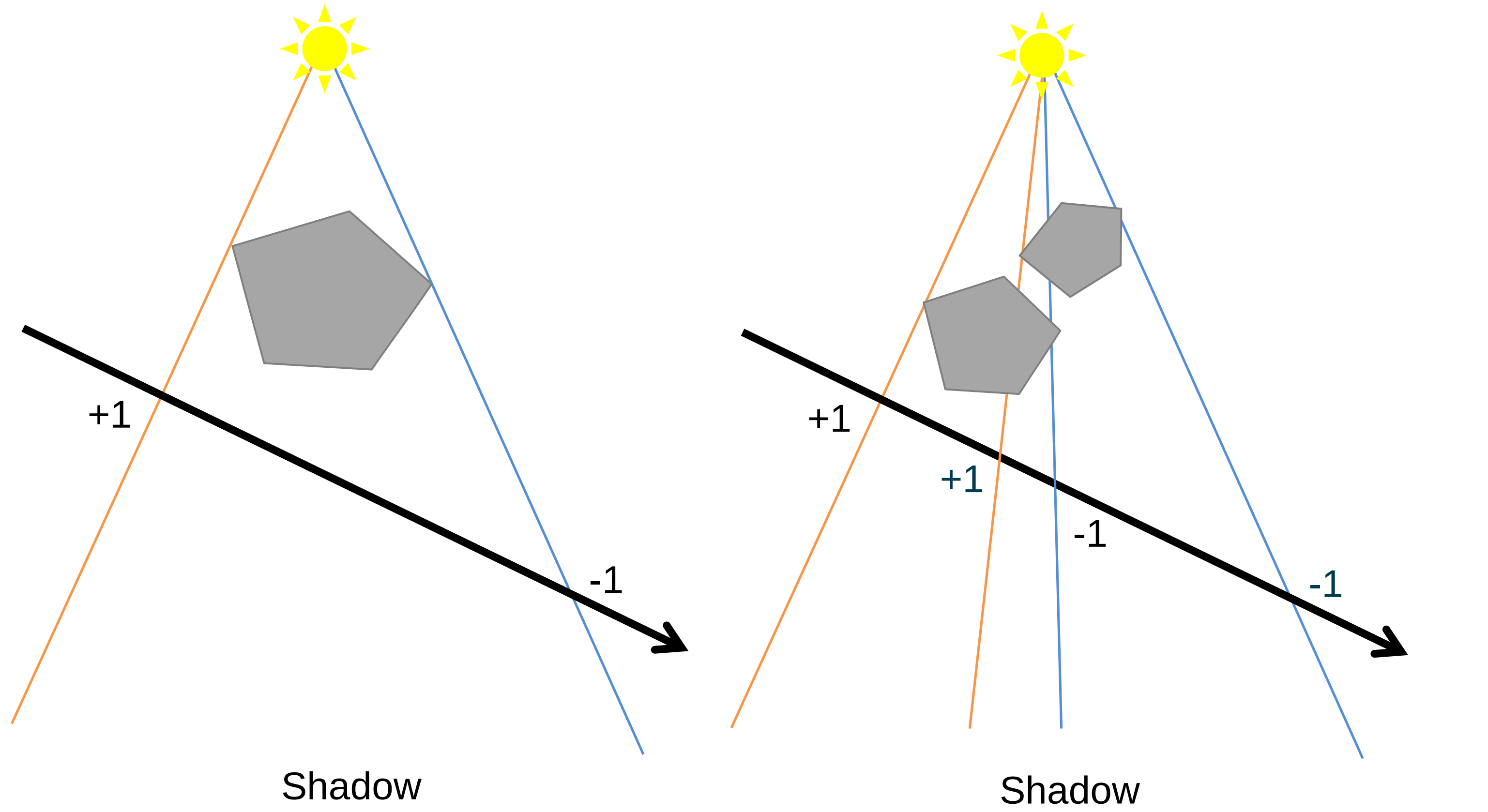
- Just like a polygon - you are inside a volume if you need to cross a surface to exit it
- General idea of shadow volumes is count the number of shadow planes you cross
  - +1 for front facing
  - -1 for back facing
- If total is >0 you are in shadow
- Special case if the eye itself is in shadow

Shadow Volumes

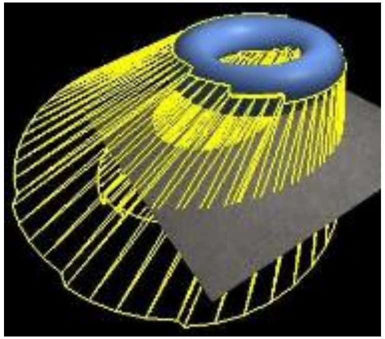
Two stages:

1) Volume **construction**
   – Find all planes of the shadow volume and their plane equations
2) Volume **test**
   – Determine shadow plane count per pixel
   – Use a scan-line method OR stencil test

Shadow Volume Example



Shadow Volumes with OpenGL



- Shadow volumes are rendered at each frame
- The stencil buffer is used for counting how many SV are crossed
- Sometimes not all objects are used for casting shadows

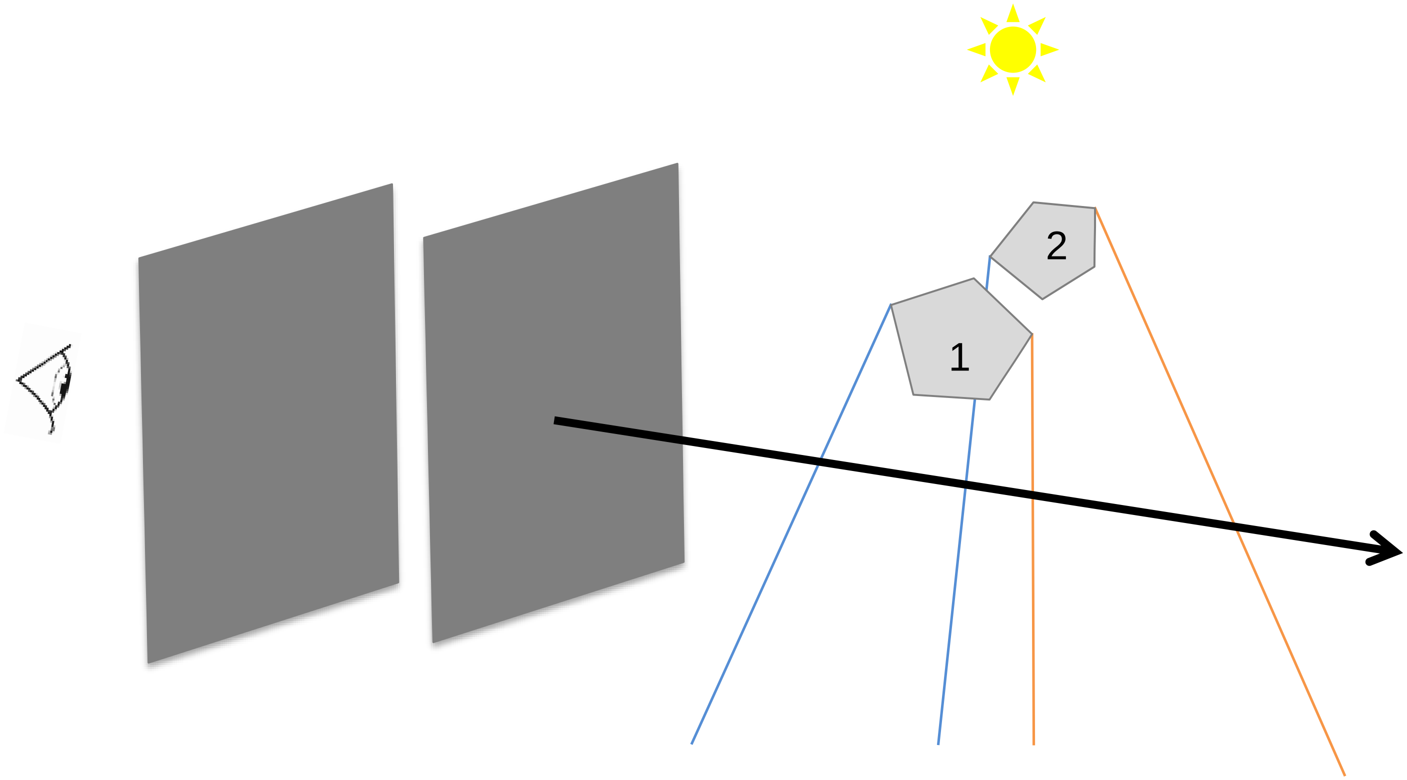
Shadow Volumes & Stencil Test

- A stencil buffer is screen sized buffer (1-8 bit) that stores a flag about a rendering operation
  - E.g., `stencil[x, y]` is negated if `zbuffer[x, y]` is less than current z value (i.e. stencil is set if and only if z buffer-test passes)
- Many uses in graphics

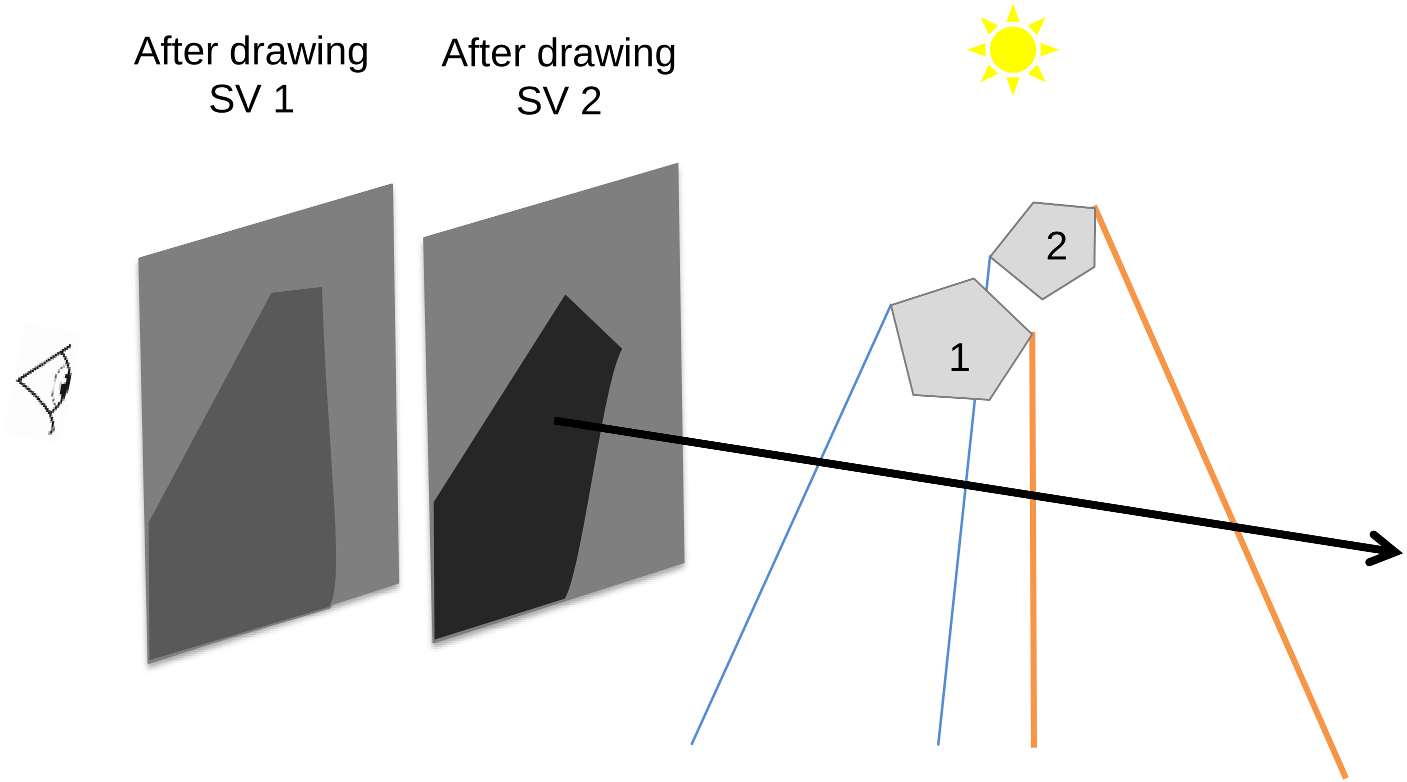
Shadow Volumes & Stencil Test

- Render the scene into the RGB and z-buffer
- Turn z-buffer **writing** off
- Render all shadow volume polygons with the stencil buffer
  - Increment stencil count for front-facing
  - Decrement for back facing
- Re-render scene with lighting **off** and only render pixels where stencil is non-zero

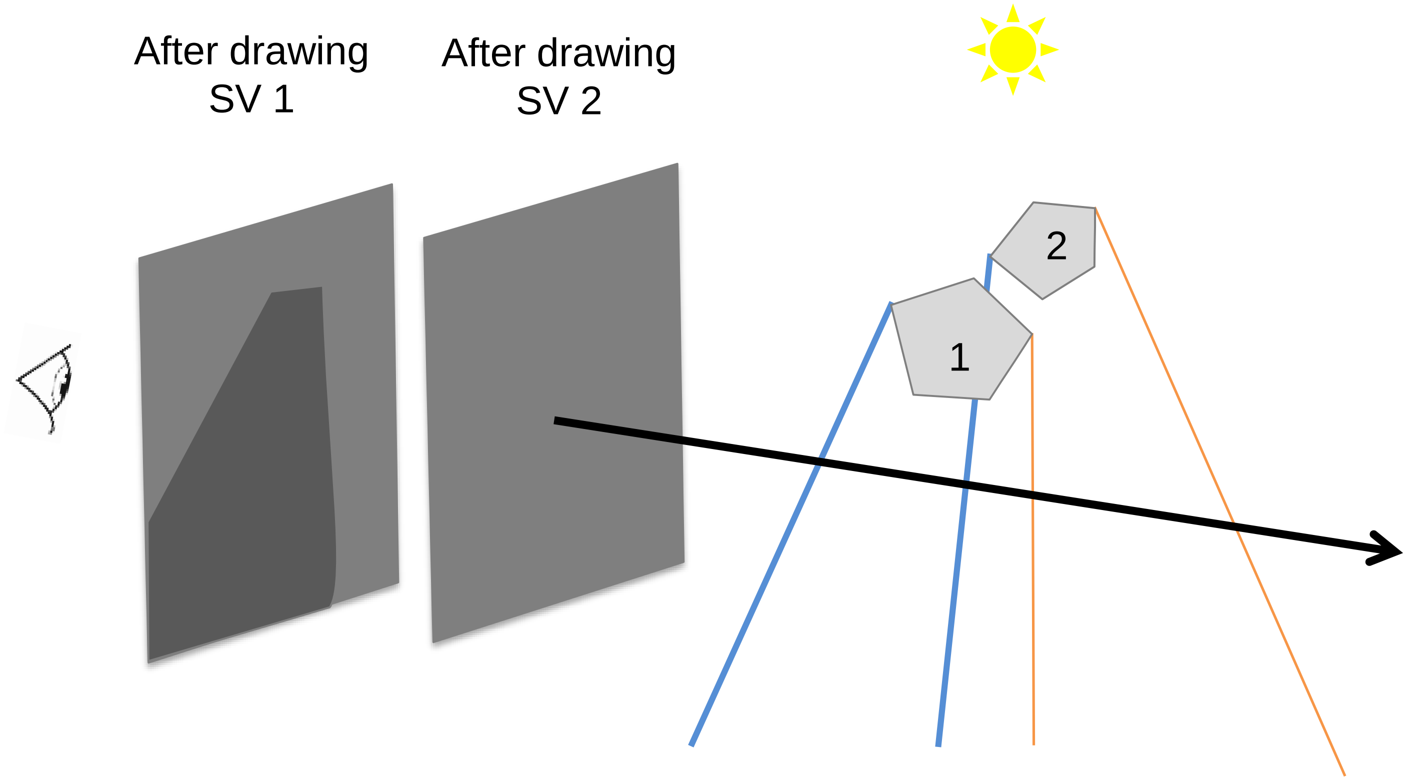
No-shadow example



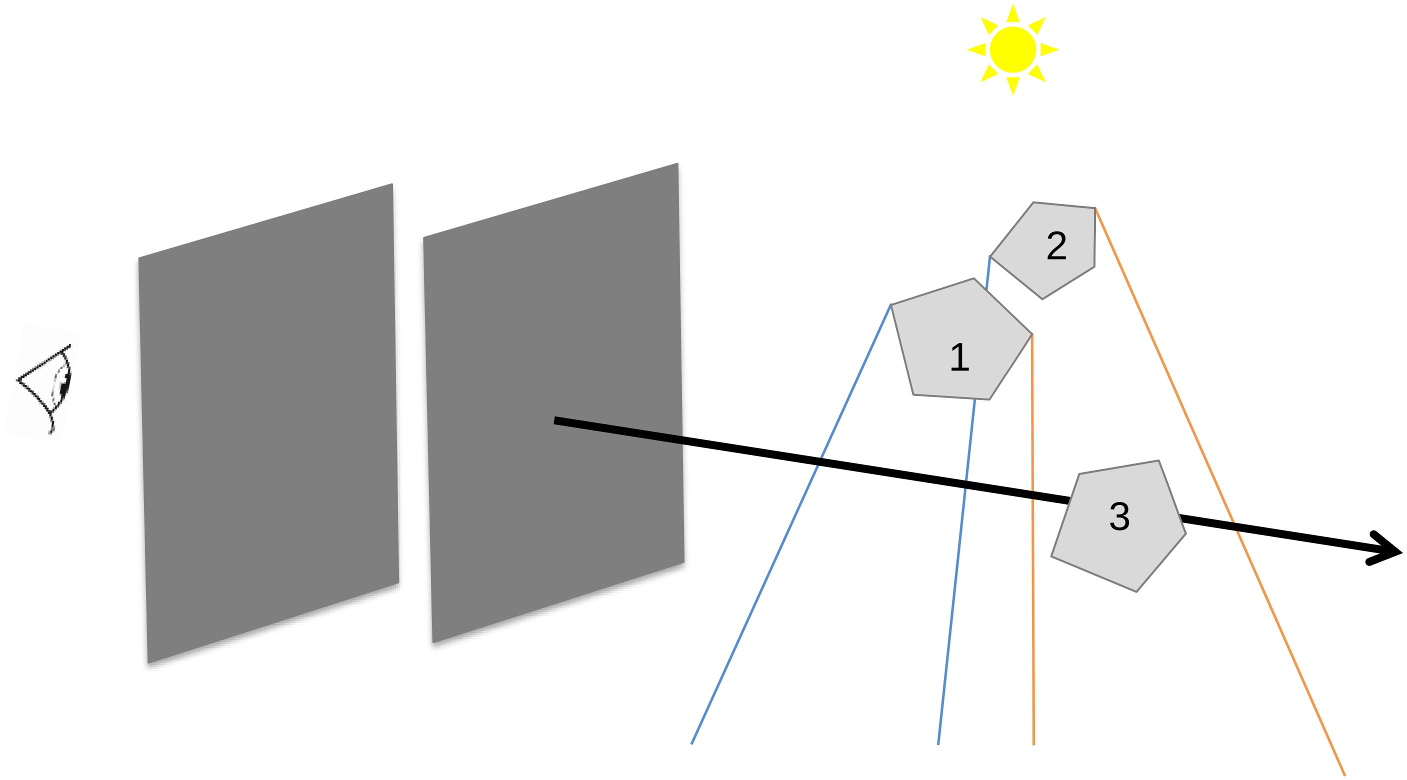
No-shadow example



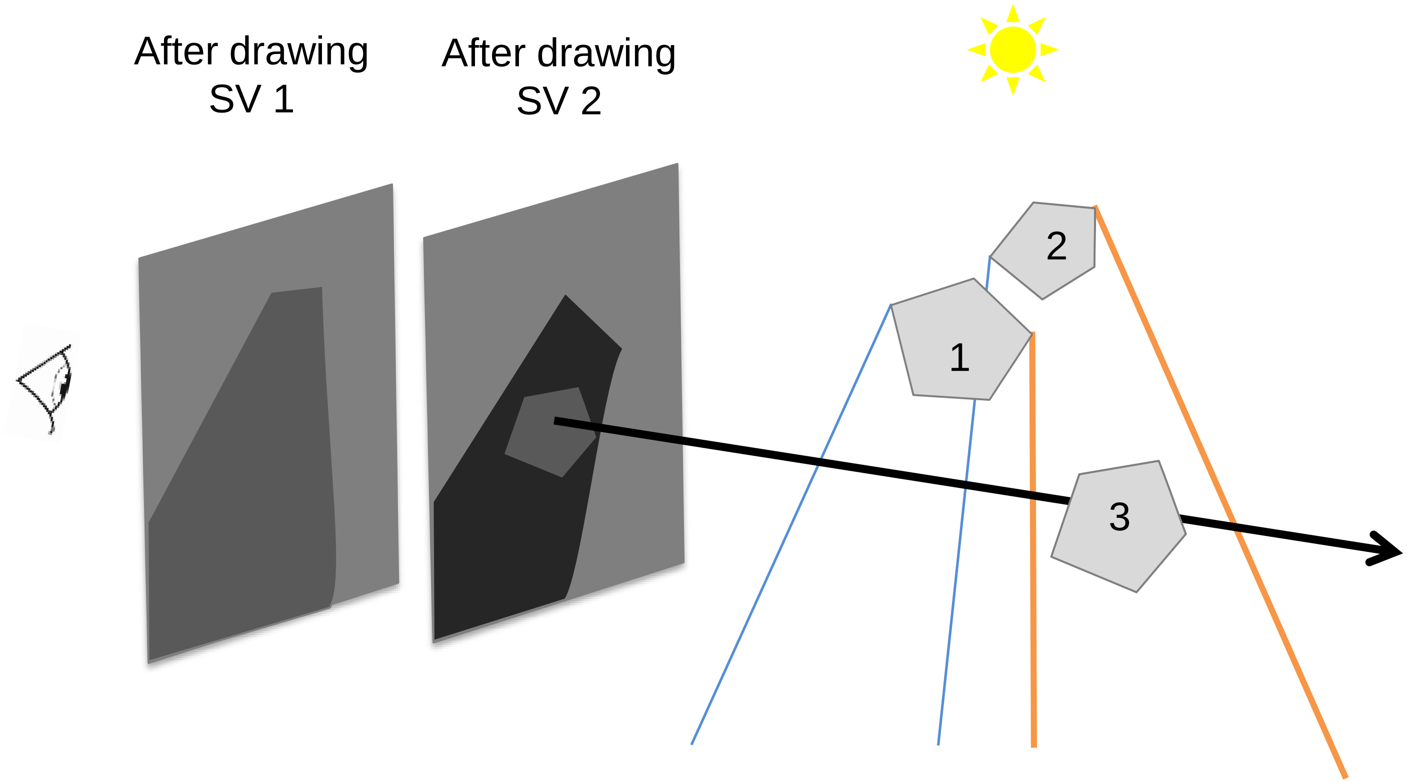
No-shadow example



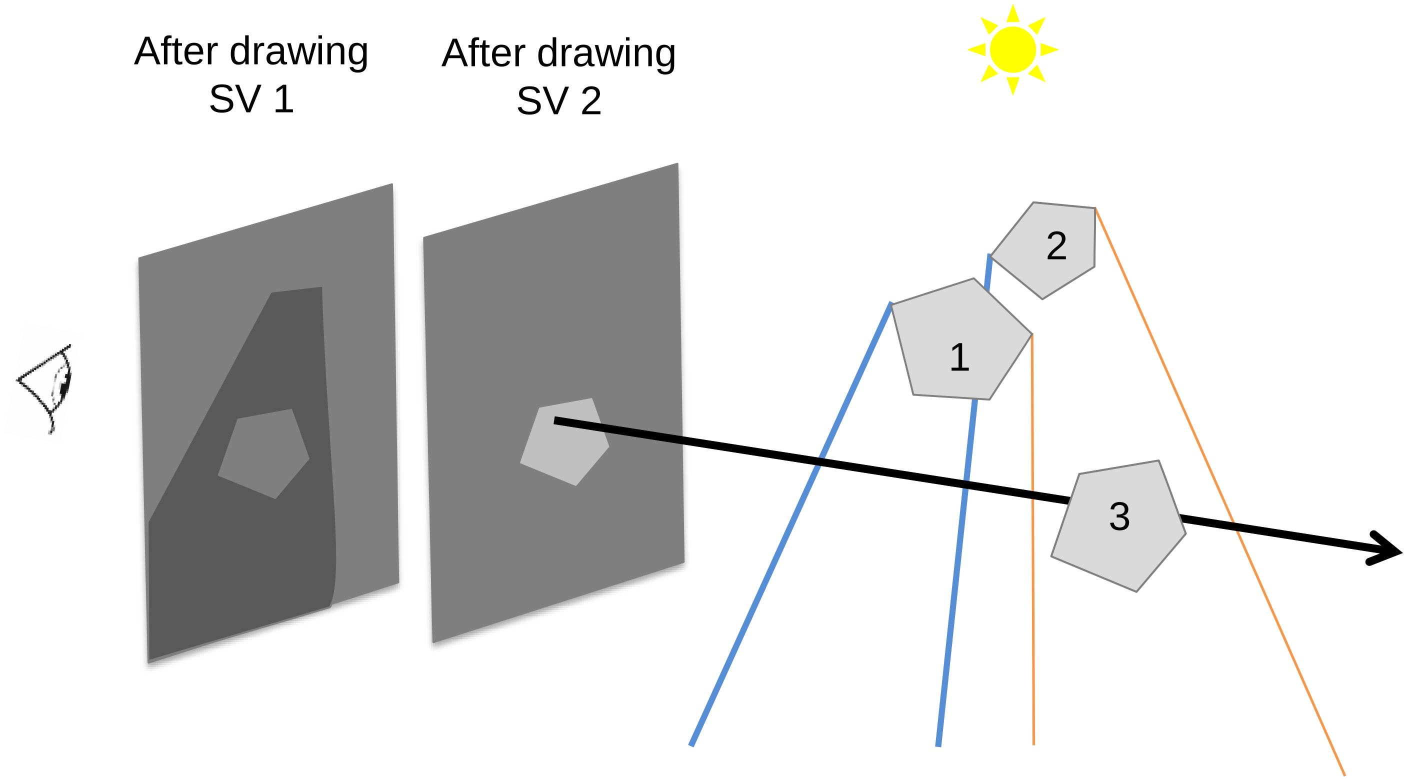
Has-shadow example



Has-shadow example



Has-shadow example



Summary for Sharp Shadows

- Four shadow umbra techniques
- Image space
  - Shadow maps
  - Shadow volumes

Soft Shadows

Soft Shadows

- Source has a finite extend
- Images look a lot more realistic

Soft Shadows

Hardware-assisted
- Accumulation buffer
- Shadow volumes
- Shadow textures

Pre-computed
- Illumination Maps

Radiosity
- Hemi-cube
- Ray-casing

Ray-based
- Distribution ray tracing
- Cone Tracing

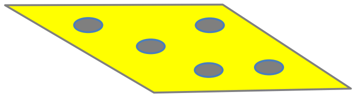
Analytical v. Sampling

- Analytical
  - Find all boundaries within the penumbra. Done almost exclusively for polygonal light sources

- Sampling
  - Approximate solution that treat the light source as a set of points. Any shape source is possible.

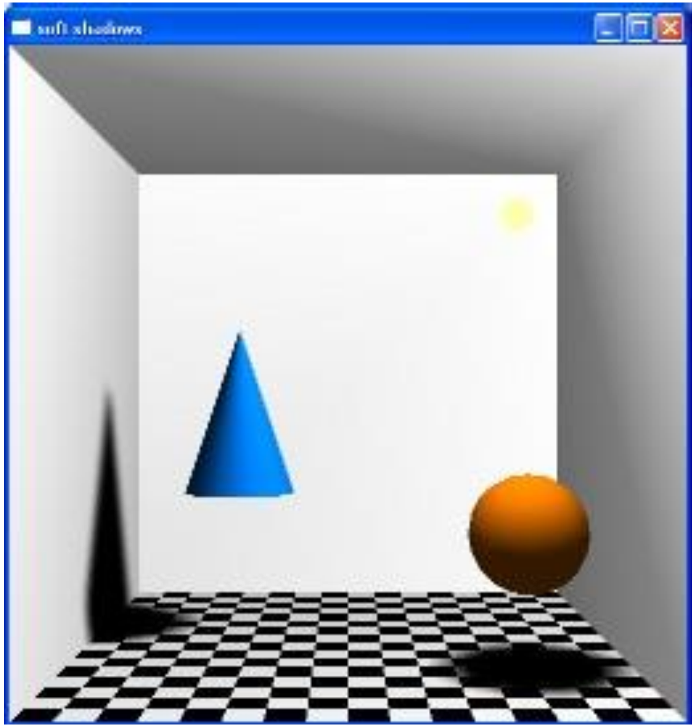
Soft Shadows using Point Sampling

- Place many point lights on an area light
  - Random positions work just fine
- Render hard shadows from each point light
  - E.g., using shadow volumes or shadow maps
- Sum up all contributions
  - Can be done on the GPU (in the frame-buffer)



- Similar to what ray-tracing does to get soft shadows

Example



Illumination Maps (Shadow Textures)

- Shadows are pre-computed and stored as textures on the receiving polygons
- Displayed using graphics hardware in real-time
- Often use: Radiosity / Path tracing / Photon mapping
- Sometimes called “baked” lighting, very common in game engines
- Disadvantage: lighting cannot change

Recap

- In order to regain shadows in real-time engines, we have to do a lot of work
- A very large number of shadow algorithms exist
- Many of them are unsuitable for walkthroughs of very complex scenes:
  - with pre-computation methods scene cannot be modified
  - or are to slow (ray-tracing, soft shadows)
- Hard shadows
  - on-the-fly methods (SM and SV) are fast enough