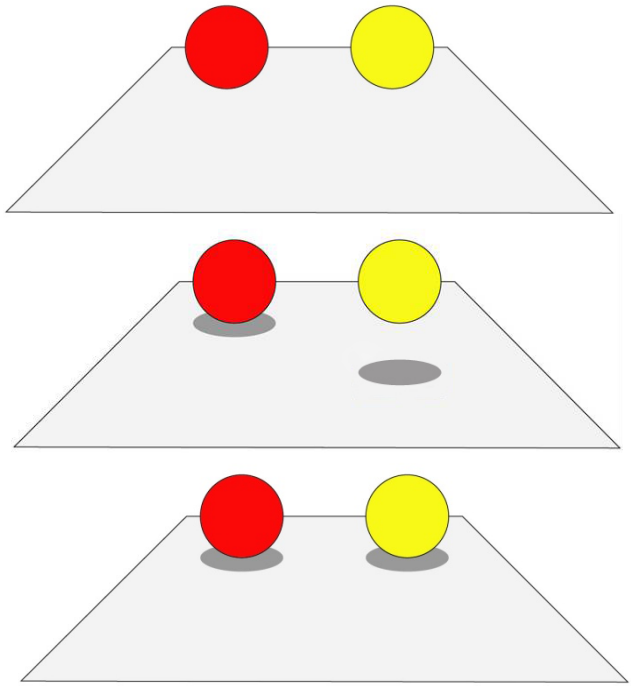
Real-Time Rendering

Intro to Shadows

CSE 781

Prof. Roger Crawfis

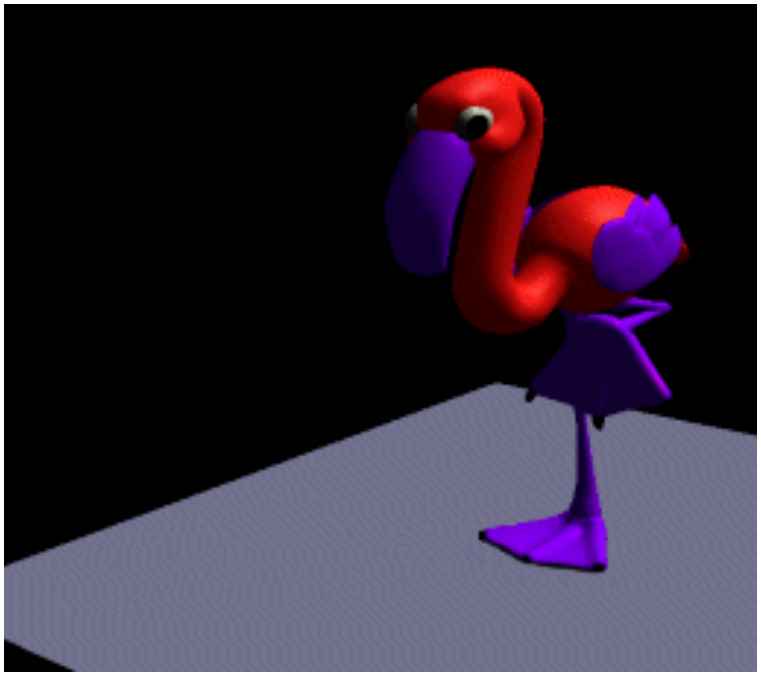
Importance of Shadows

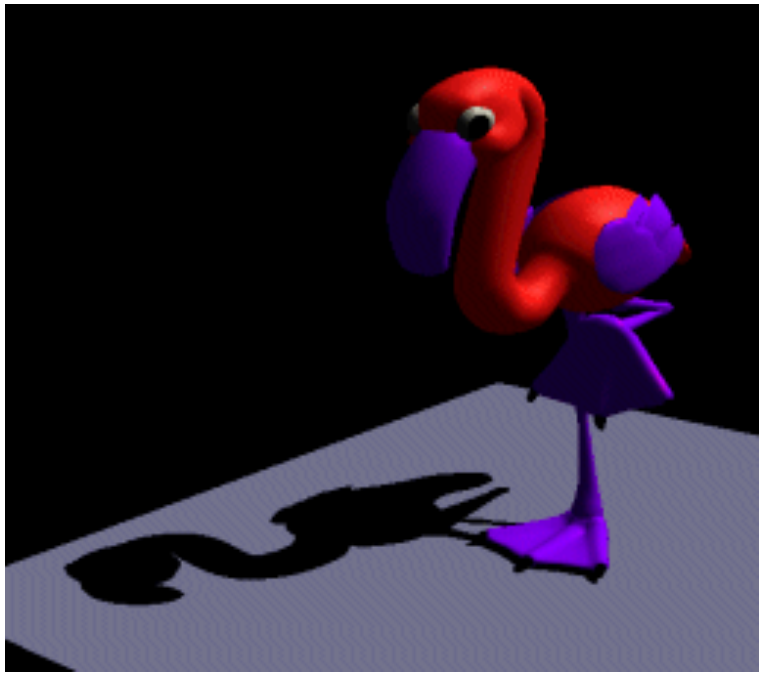


Trapezoid?

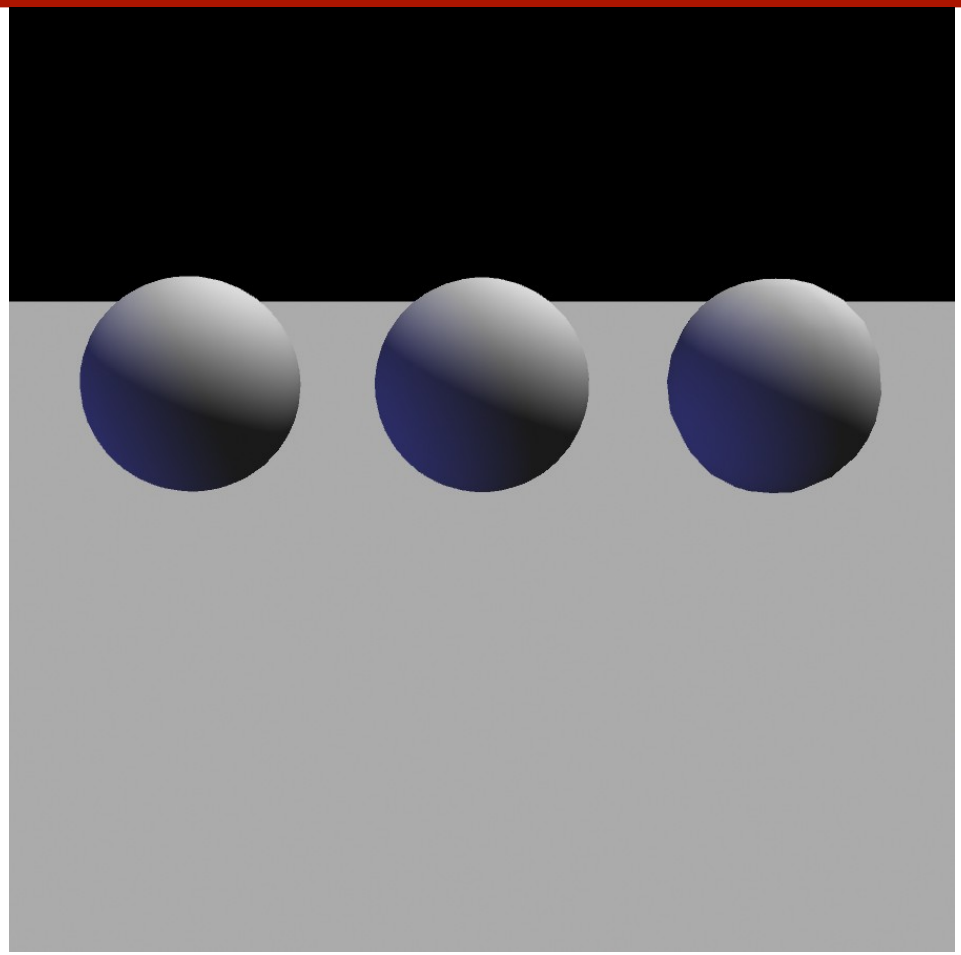
Importance of Shadows

- Cue to object-object relationship.
- Provides additional depth cue.

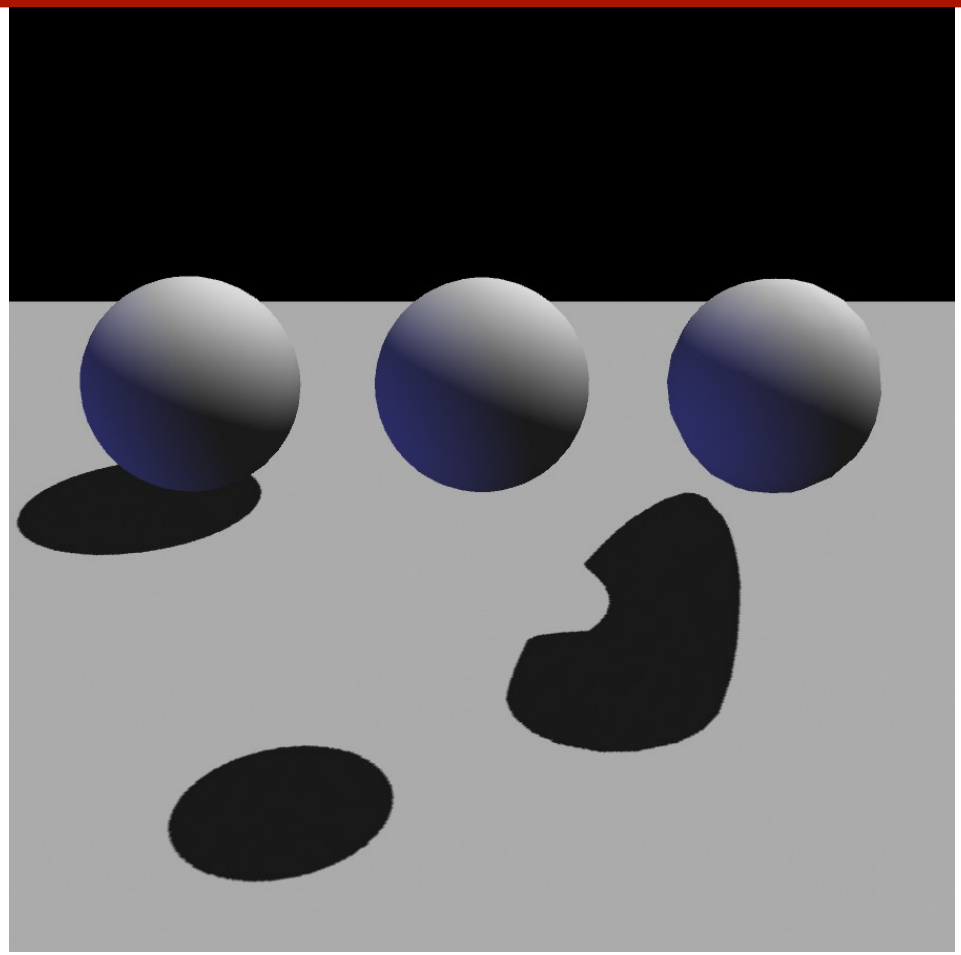




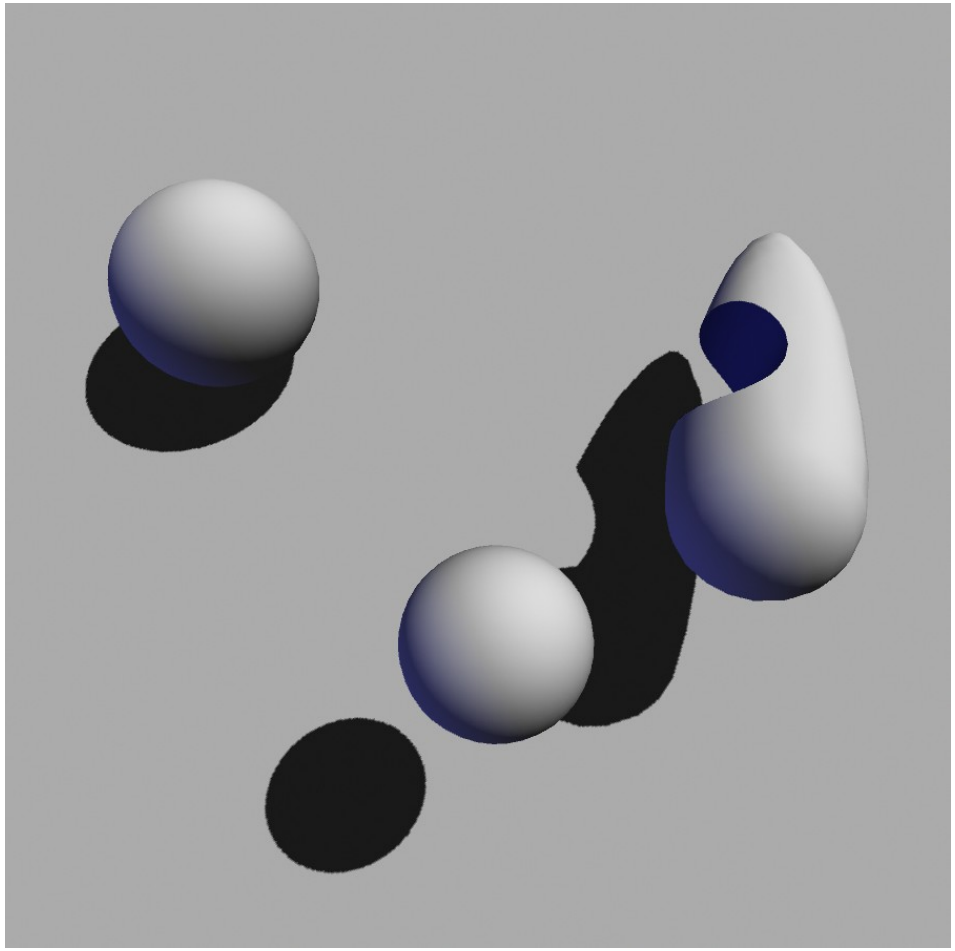
Importance of Shadows



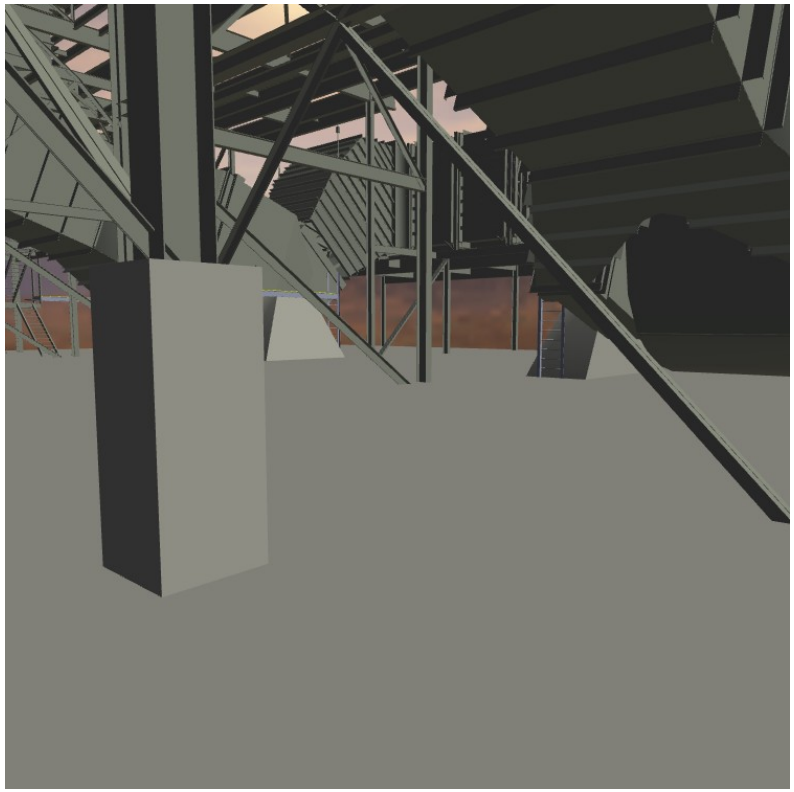
Importance of Shadows



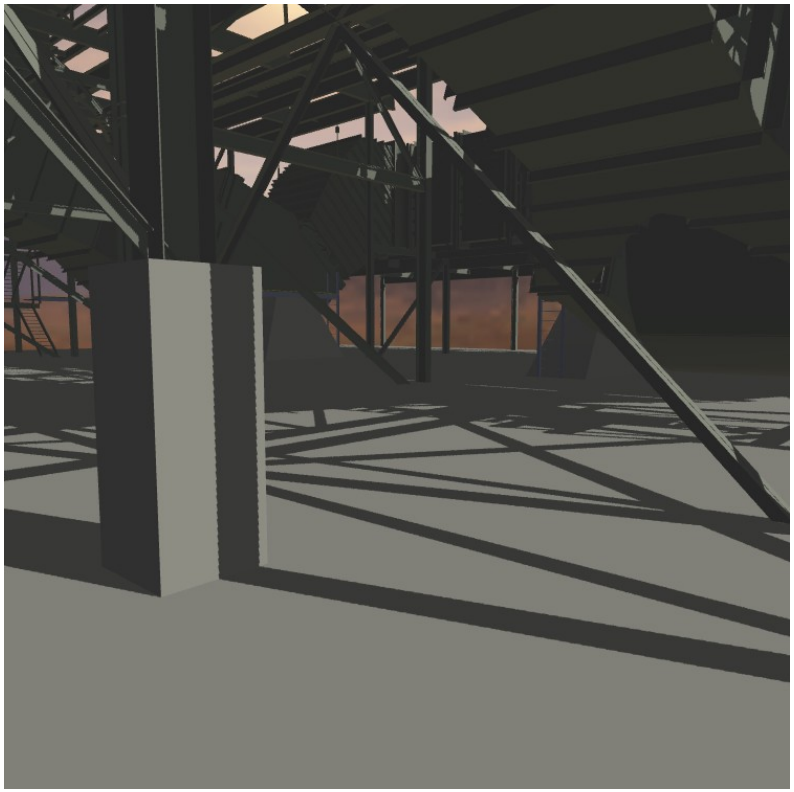
Importance of Shadows



Importance of Shadows



Without shadows

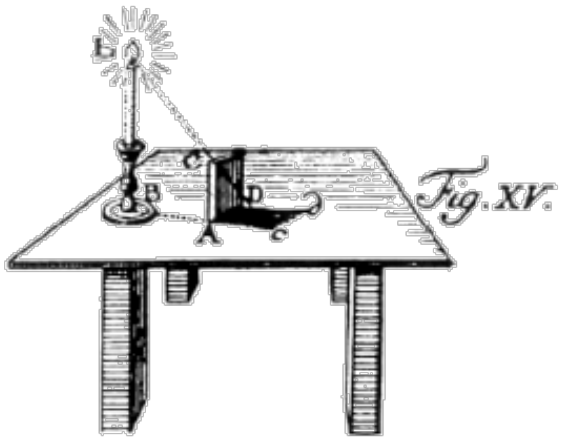


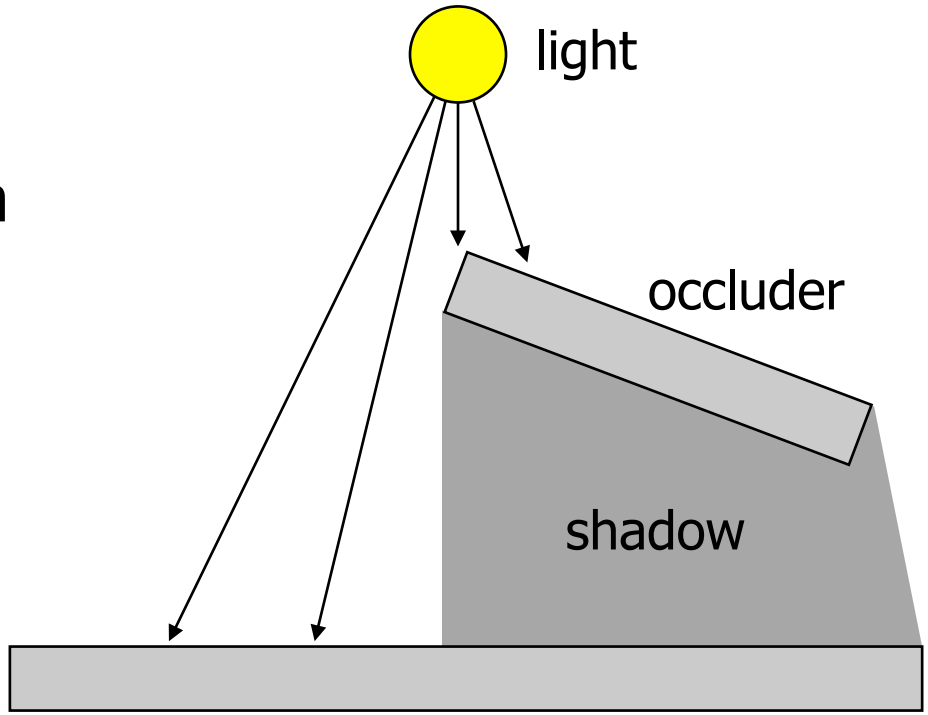
With shadows

Importance of Shadows

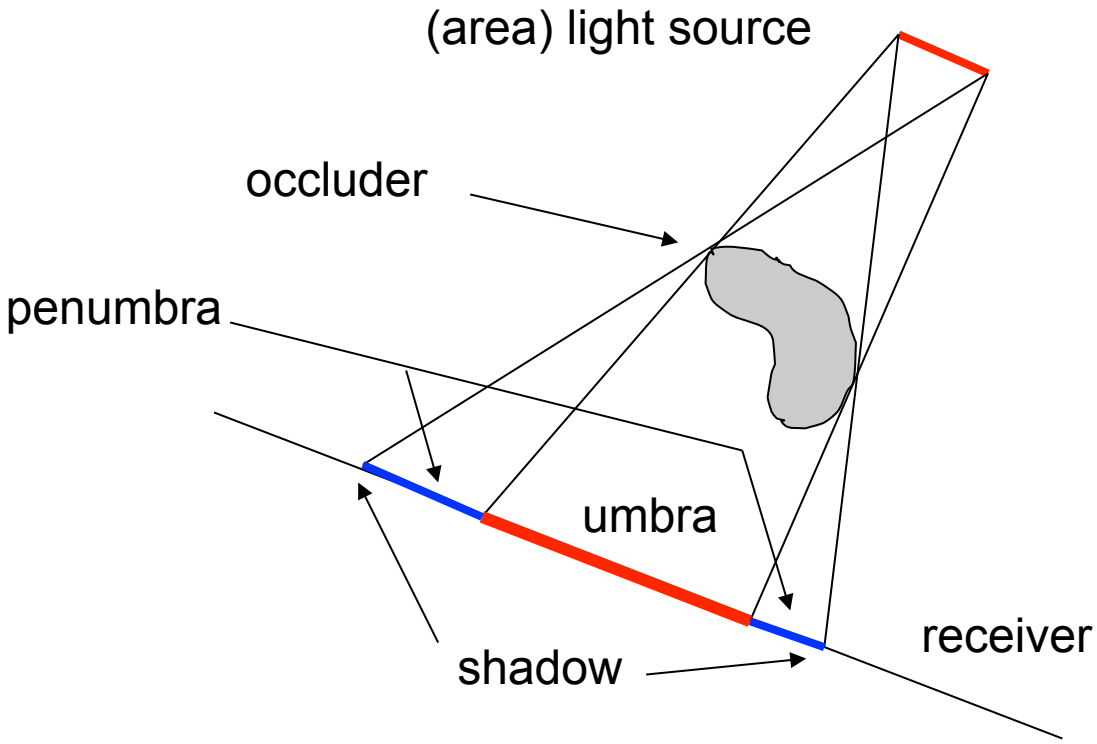
Definition

Shadow: Darkness caused when part or all of the illumination from a light source is blocked by an occluder.



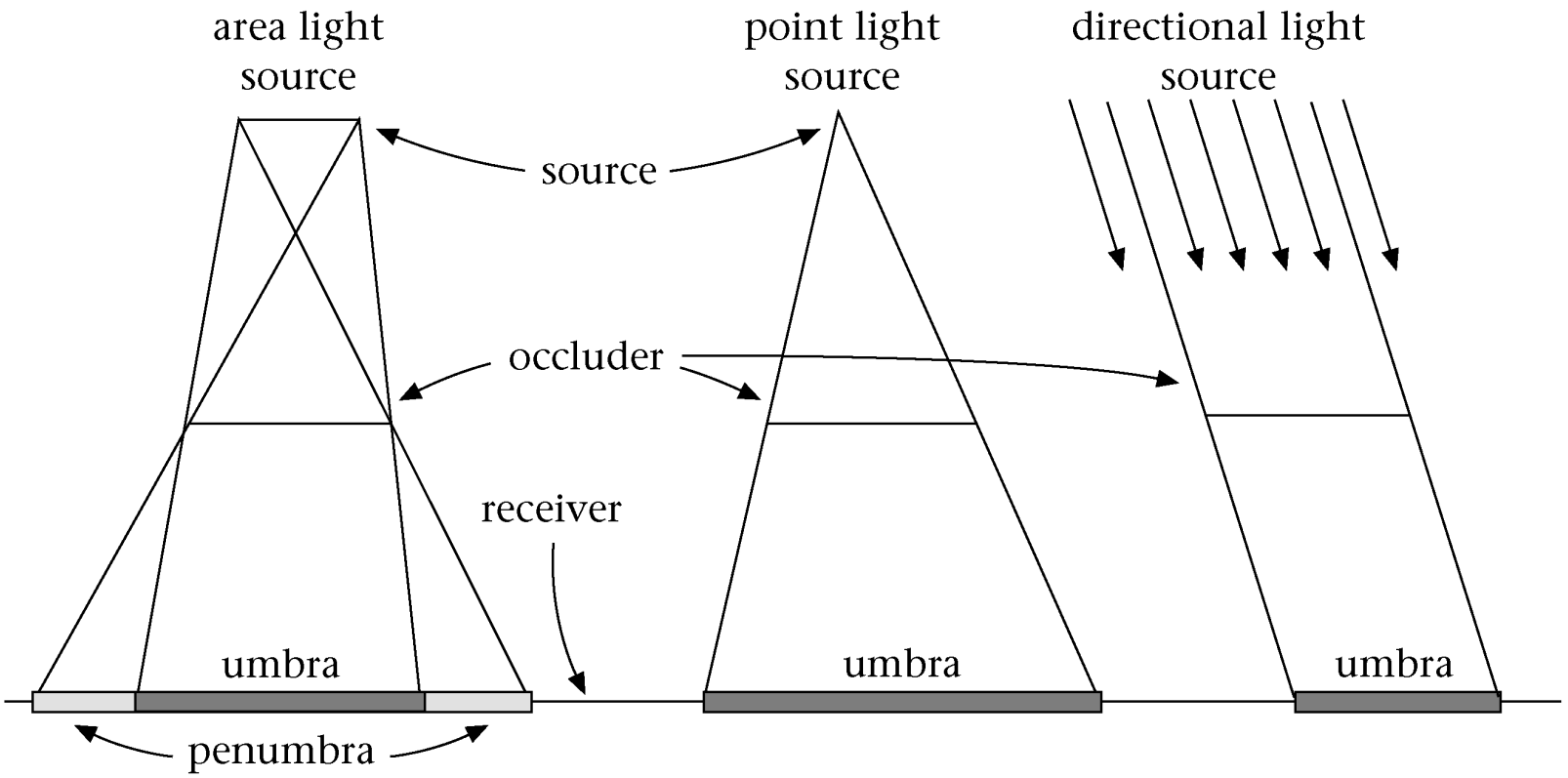


Terminology



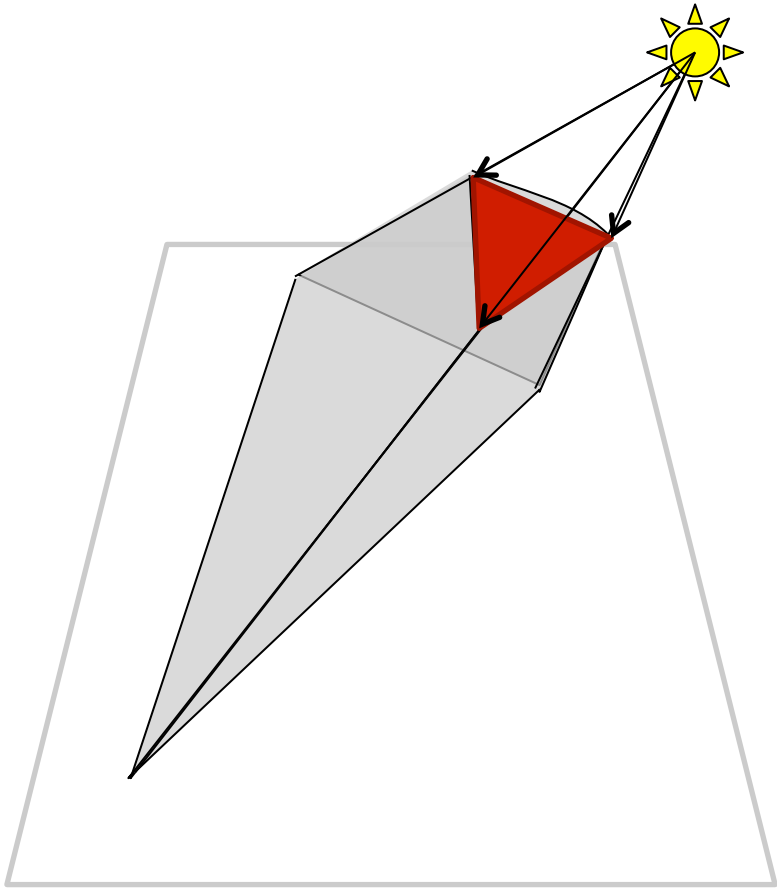
- **umbra** – fully shadowed region
- **penumbra** – partially shadowed region

Terminology

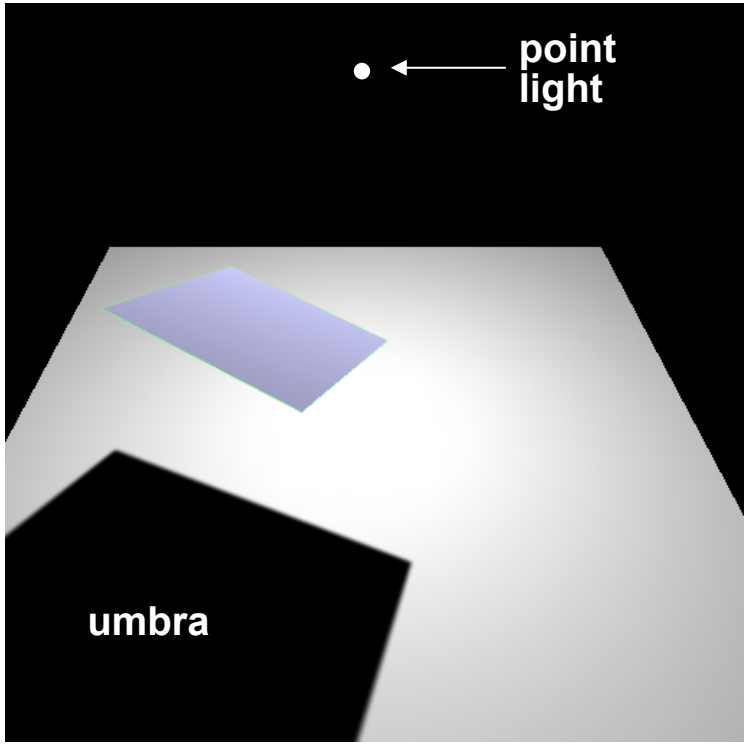


Definition: Shadow Volume

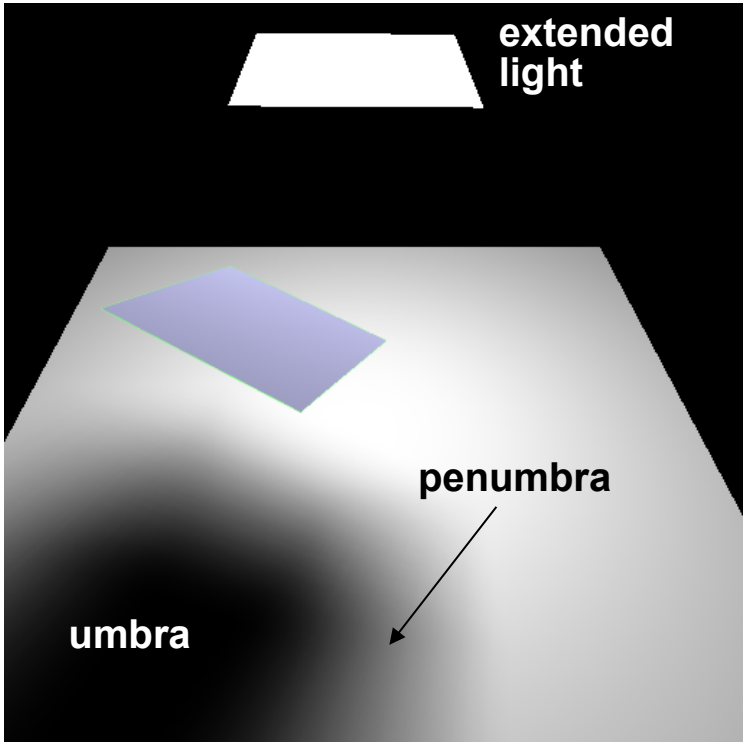
- Volume formed by extruding the occluder from the light source.
- Open and infinite
- Space inside the volume is in shadow.
- Space outside the volume is not.



Hard and soft shadows



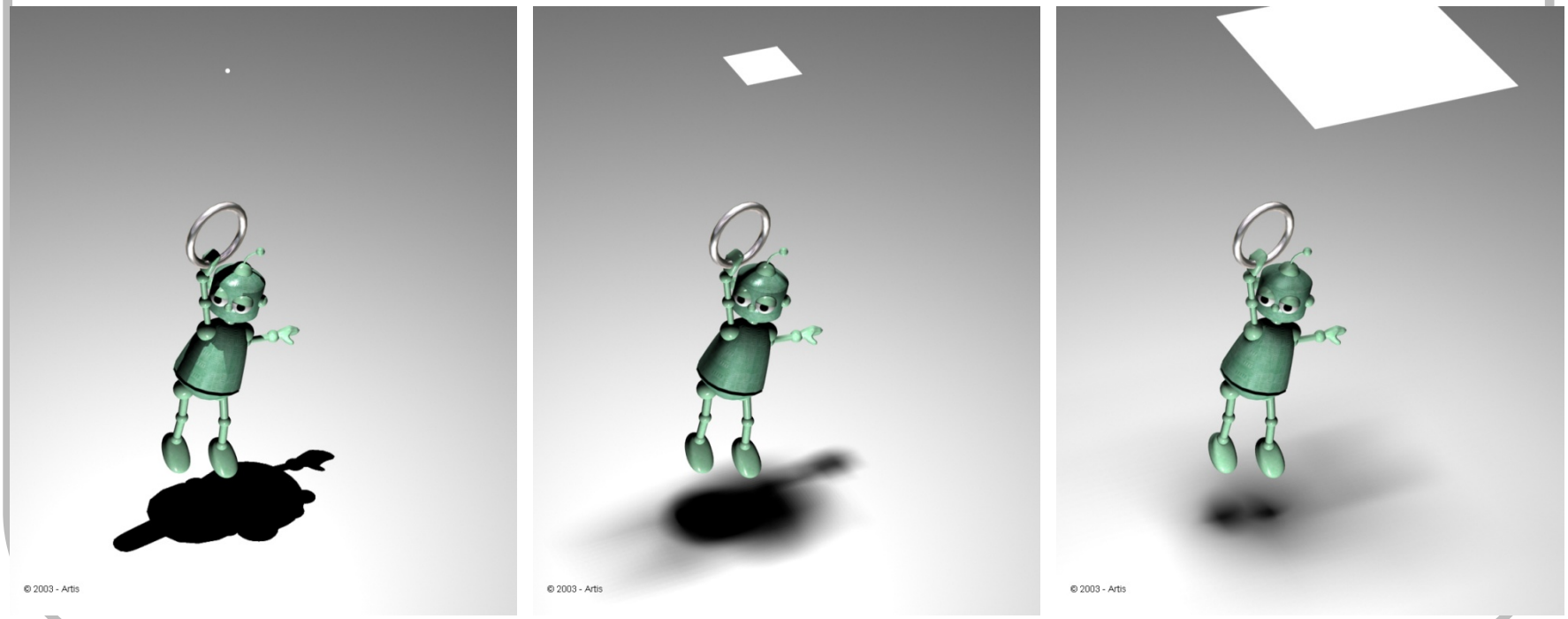
Hard shadow



Soft shadow

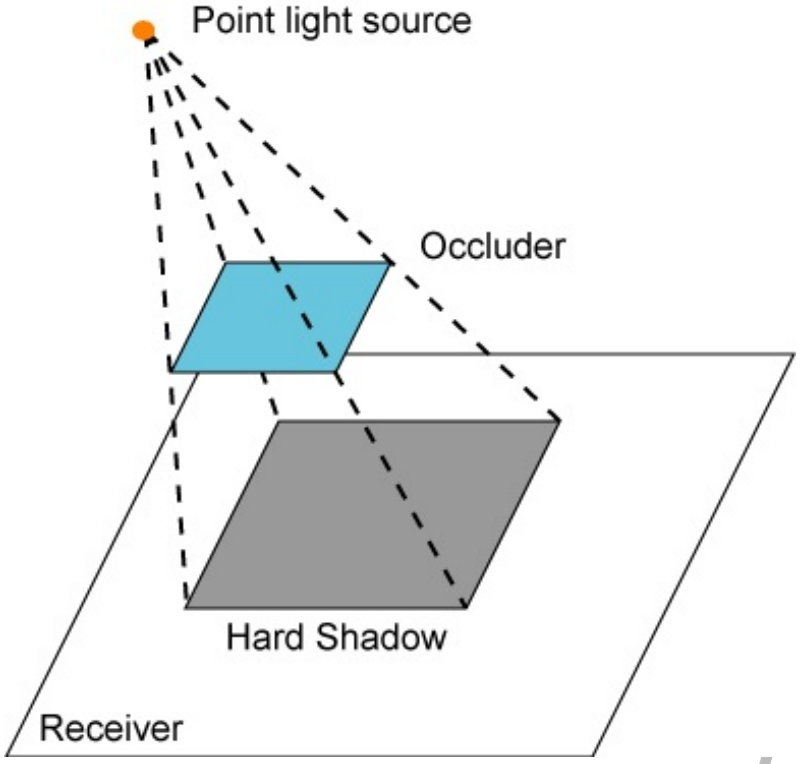
Soft Shadows

- Shadow is a function of the area of the light source and the distance.

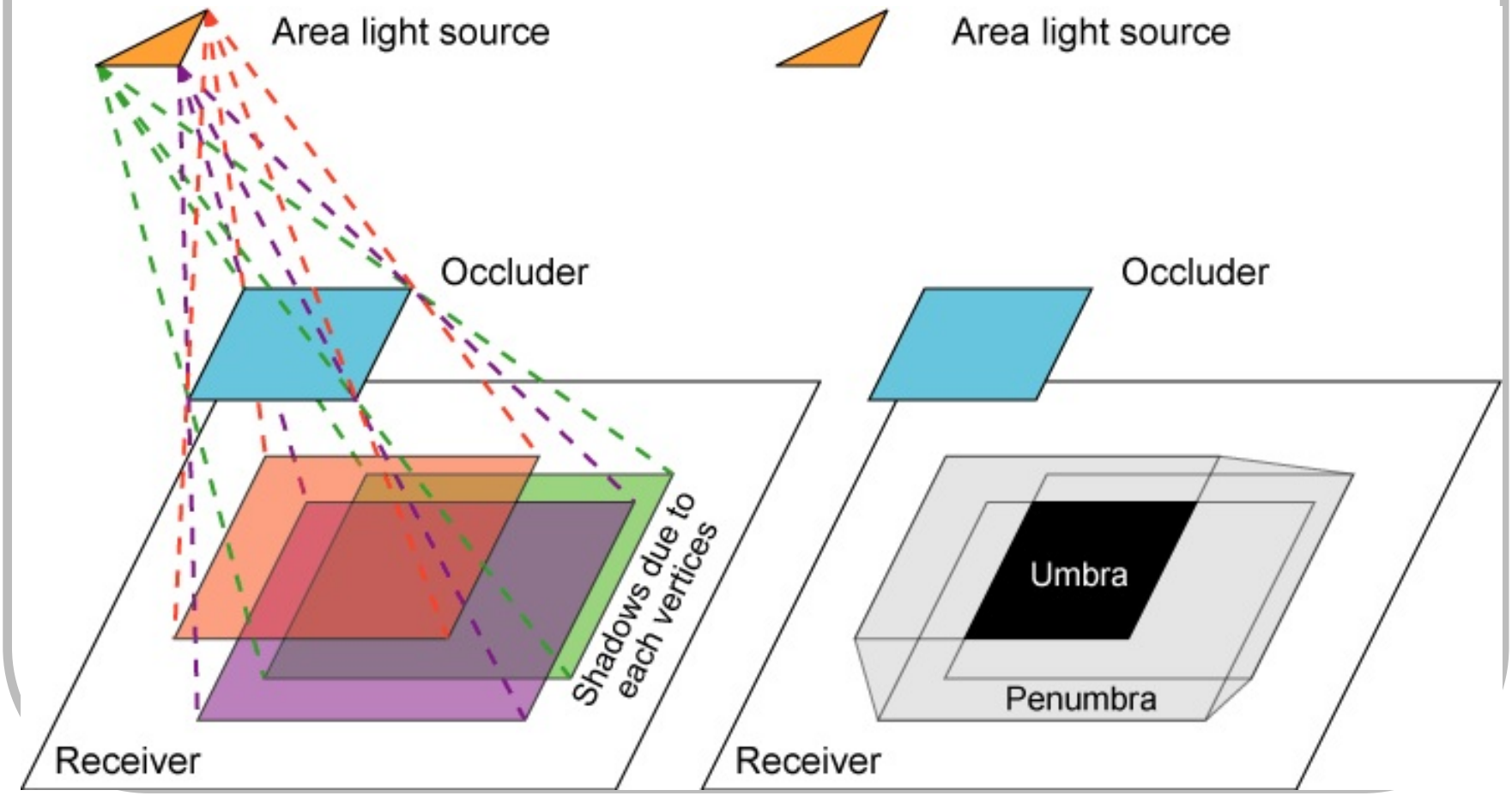


Hard shadow creation

- For every pixel, light source is either visible or occluded



Soft shadow creation

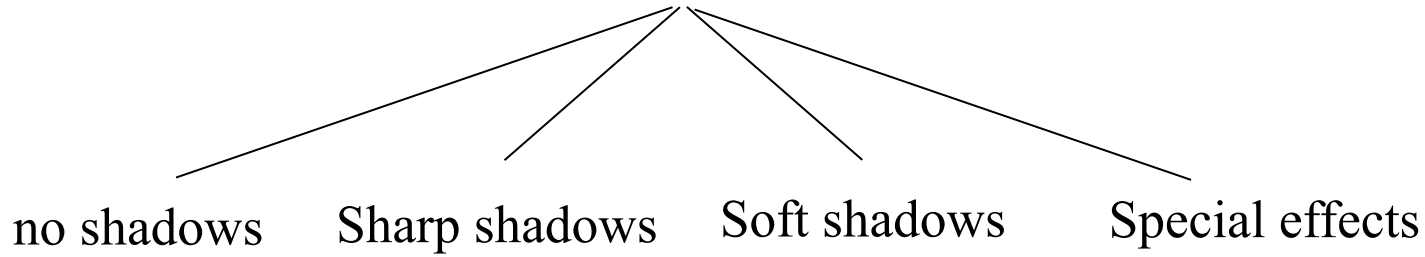


Issues Affecting Shadows

- Scene Complexity
  - Number of light sources
  - Types of light sources
  - Number of occluders
  - Number of receivers
  - Position, size and strength of lights
- Static vs. dynamic
  - Objects
  - Lighting
- Self-shadowing
- Opaque vs. transparent objects
- Precision or realism of shadows

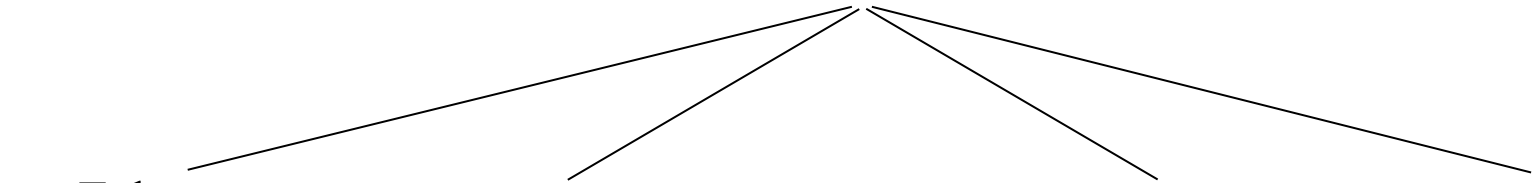
Current Shadow Methods

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



Sharp Shadows

- Point or directional light sources



Fake

Fast but often inadequate
- Billboard
- Projected object

Hardware Assisted

Use of specialized hardware to generate shadows
- Shadow texture
- Shadow volumes

Pre-computed

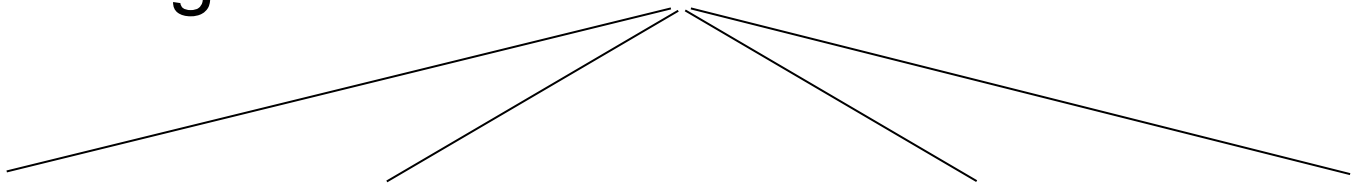
Shadows are pre-computed and stored for repeated use
- Detail polygons

Ray tracing

Rather slow for large scenes

Soft Shadows

- Area light sources



Hardware Assisted

Mainly treat the light source as a collection of points

- Accumulation buffer
- Shadow volumes
- Shadow textures

Pre-computed

Mainly analytical computation on the geometry of the source

- Light maps
- Discontinuity Meshing

Radiosity

This is also pre-computed

- Hemi-cube
- Ray casing

Ray-based

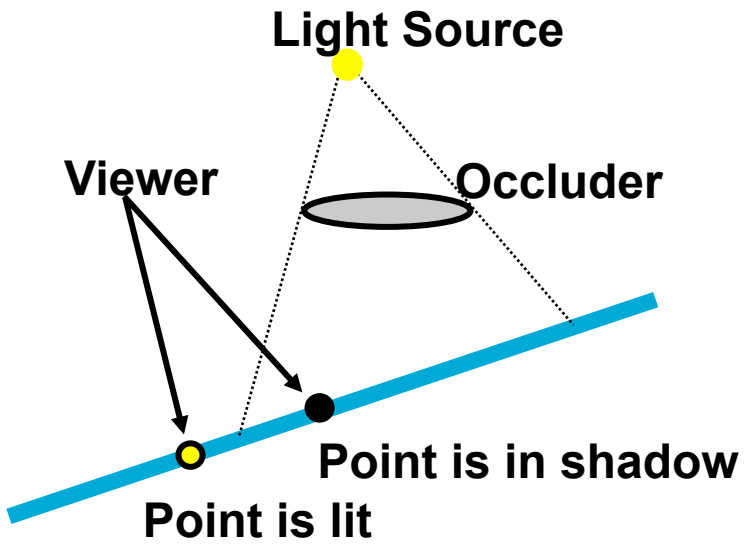
- Distributed ray tracing
- Cone Tracing

Approaches

- Ad-hoc / Custom
  - Artist directed shadows
  - Very simple and constrained conditions.
- Analytical
  - Find all boundaries within the umbra / penumbra. Precise.
- Sampling
  - Probabilistic sampling of whether a particular fragment is within the umbra / penumbra. With enough samples can be made precise.

Shadows and Illumination

- Given a point is in shadow, how do we change the illumination?



Shadows and Illumination

- Illumination model with included visibility term:

$$I = k_a I_a + \sum_{i=1}^{N} V_i I_{l_i} (k_d \max(0, \mathbf{n} \cdot \mathbf{l_i}) + k_s \max(0, \mathbf{v} \cdot \mathbf{r_i})^n)$$

For all lights

Visibility Term: 0 or 1 for point/directional lights

- Visibility term determines if a light can "see" the point
- If point is in shadow, only ambient term applies

Shadows and OpenGL

- In OpenGL we send the geometry for a model through the pipeline.
- The Visibility function, ***V***, is not a constant in our illumination model.
  - Per vertex information?
  - Per fragment using a texture map?
  - Some per-pixel masking function?
- Recall that we need a ***V*** for each light.

Shadows and Illumination

- Note, we can re-write our illumination equation:

$$I = k_a I_a + \sum_{i=1}^{N} V_i I_{l_i} \overbrace{(k_d \max(0, \mathbf{n} \cdot \mathbf{l_i}) + k_s \max(0, \mathbf{v} \cdot \mathbf{r_i})^n)}^{c_i}$$

$$= k_a I_a + \sum_{i=1}^{N} V_i I_{l_i} c_i$$

$$= k_a I_a + \sum_{i=1}^{N} \left[I_{l_i} c_i - (1 - V_i) I_{l_i} c_i \right]$$

- Negative light.

Masking in OpenGL

- OpenGL provides several ways of *masking* pixels
  - Stencil buffer with stencil test
  - Alpha test with fragment's alpha values
  - Blending with fragment's and framebuffer's alpha values.
  - Texture sampling and shaders.

Negative Light

- Algorithm (single light)
  - Render receivers with full illumination
  - For each occluder
    - Project occluder from the light to the receivers
    - Darken (set to black or ambient) the illumination.

Negative Light

- With multiple light sources this technique does not work. If the algorithm simply sets the pixels to black (or ambient), then it will erases the contributions from all light sources.

Positive Light

- Algorithm
  - Render scene with ambient illumination only
  - For each light source
    - Render scene with illumination from this light only
    - Scale illumination by shadow mask
    - Add contribution to frame buffer