Shadows & occlusion

Georgia Tech College of Computing
Interactive & Intelligent Computing

- Shadow - occlusion duality
- Floor shadows
- Shadow buffer
- Soft shadows

Motivation

- Visibility
  - Most objects in a scene are hidden from any given view point
  - Rendering them wastes GPU cycles
  - Need efficient techniques for rejecting what is obviously hidden
    - Rejection tests can be exact, conservative (reject more), approximate
- Shadows
  - Objects in the shadow do not reflect direct illumination
  - Want to know which light sources illuminate an object
  - Need a quick test for being in the shadow

- Visibility and shadows are defined in terms of occluders
  - Surfaces or objects that may block the passage of light
  - When placed between the object and the viewer/light

Shadows provide depth clues

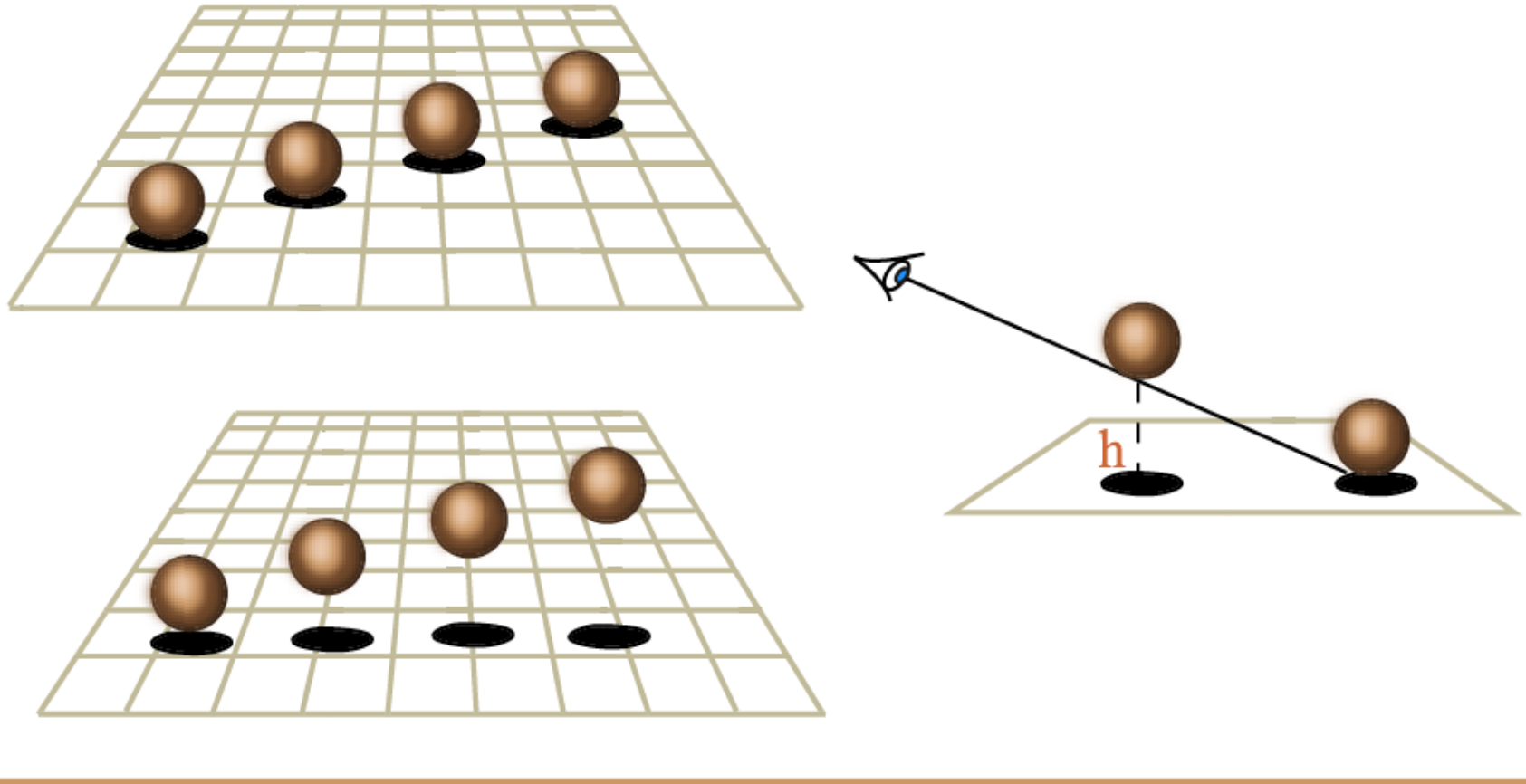


Image adapted from:
Palmer, Stephen E. *Vision Science: Photons to Phenomenology*. MIT Press. May 1999. ISBN: 0-262-16183-4.

4x4 matrix formulation (review)

$$S(u,v,w) = \begin{bmatrix} u & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T(u,v,w) = \begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R(U,V,W) = \begin{bmatrix} U_x & V_x & W_x & 0 \\ U_y & V_y & W_y & 0 \\ U_z & V_z & W_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

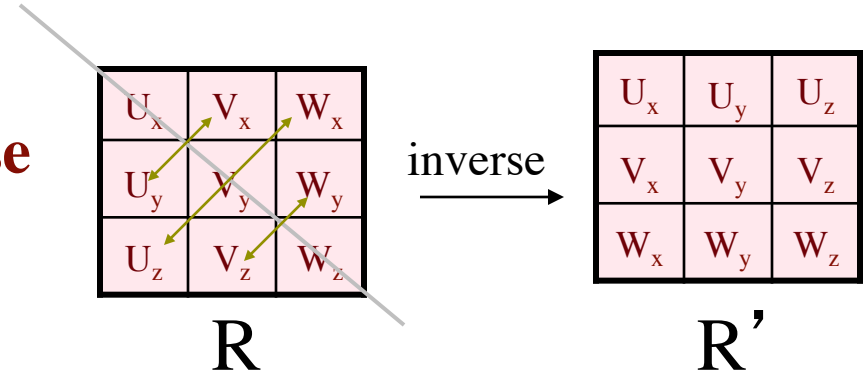
$$R_Z(a) = \begin{bmatrix} C & -S & 0 & 0 \\ S & C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M \bullet P = \begin{bmatrix} U_x & V_x & W_x & u \\ U_y & V_y & W_y & v \\ U_z & V_z & W_z & w \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} U_x x + V_x y + W_x z + u \\ U_y x + V_y y + W_y z + v \\ U_z x + V_z y + W_z z + w \\ 1 \end{bmatrix}$$

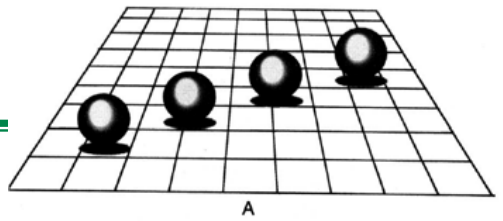
Rotation matrix

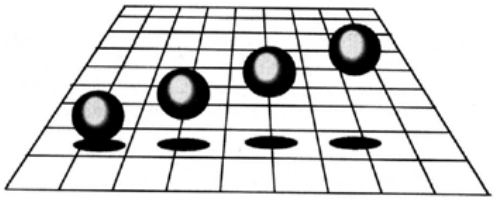
U_x	V_x	W_x	**0**
U_y	V_y	W_y	**0**
U_z	V_z	W_z	**0**
0	**0**	**0**	**1**

- ||U||=||V||=1
- **U•V = 0**
- **W=U×V**
- **R = { U V W }, 3x3 matrix**
- **R' = ◣R, inverse = transpose**
- **Rotation to map Z to W**
  - **U:=Y×W or X×W**
  - **U:=U/||U||**
  - **V:=WxU**
  - **R_{ZtoW}={ U V W }**
- **Rotation to map W to Z axis**
  - **◣R_{ZtoW}**



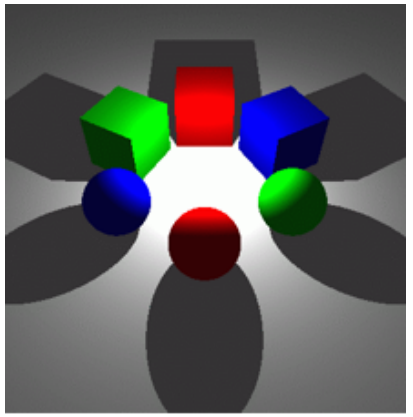
Floor shadows

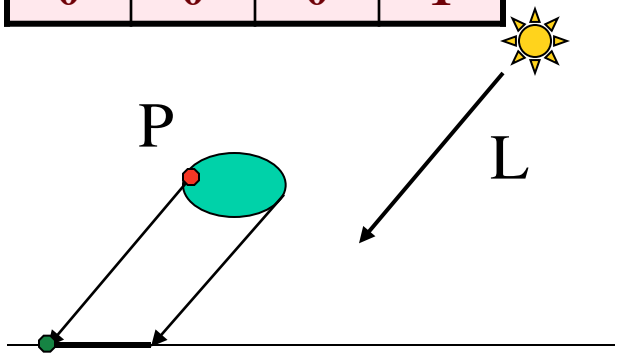




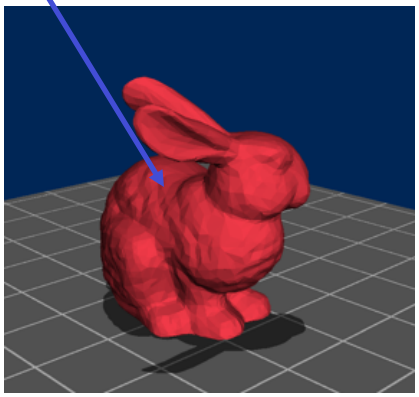
- Depth cue, realism, light position
- Draw object twice
  - Second time: projected on the ground
- **Does not support self-shadows**

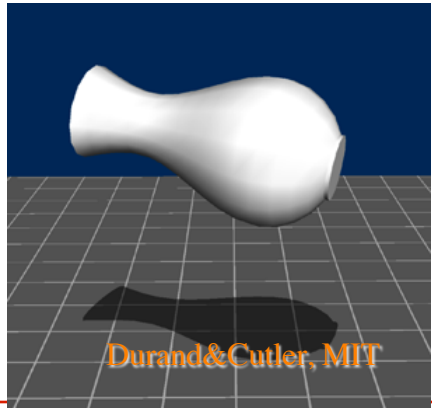
1	0	L_x/L_z	0
0	1	L_y/L_z	0
0	0	0	0
0	0	0	1





$$\mathbf{S} = \mathbf{P} + (P_z/L_z)\,\mathbf{L}$$

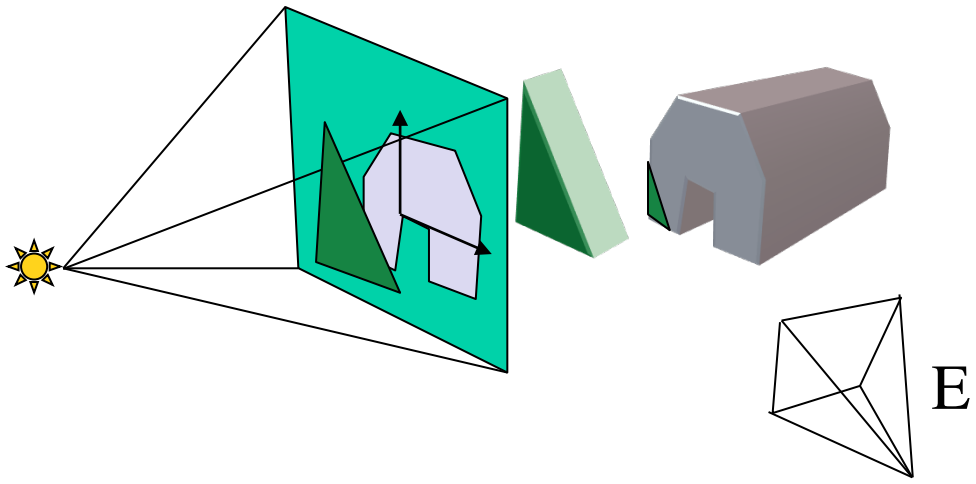


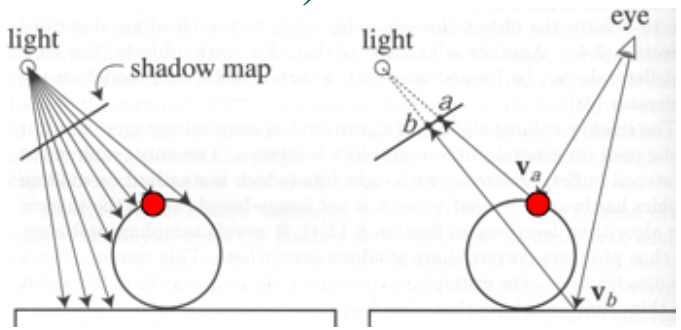


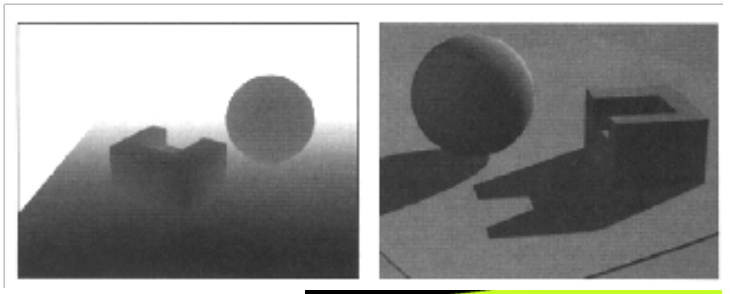
Shadow map (supports self-shadows)

- Pre-render from light : store z-buffer as shadow map (texture)
- While rendering, check whether fragments are in shadow
  - Use GPU to perform the check (compare z to texture)

Shadow-castikng fragments must be in light frustum

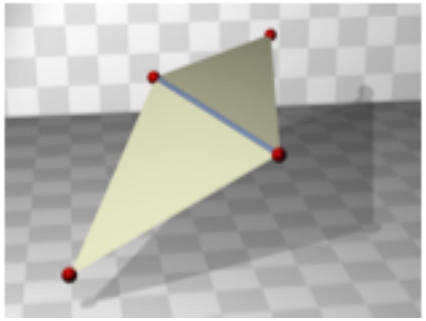


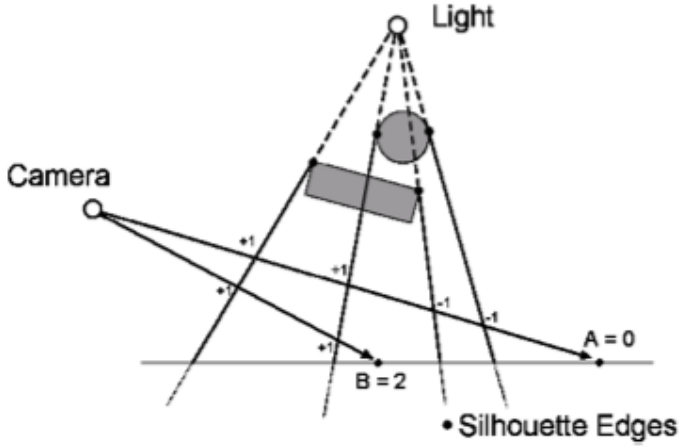


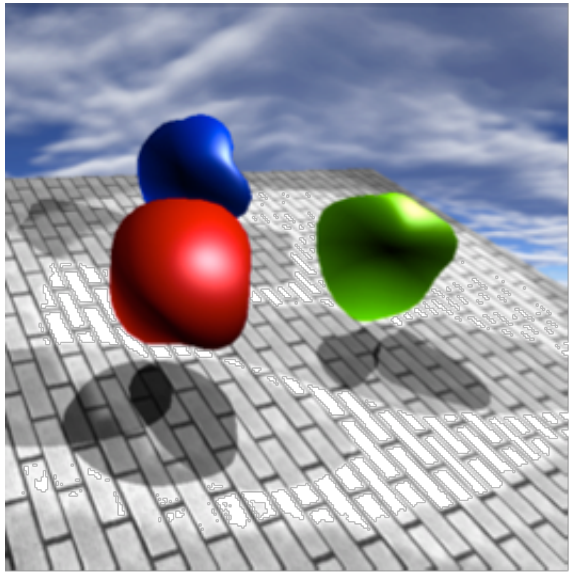


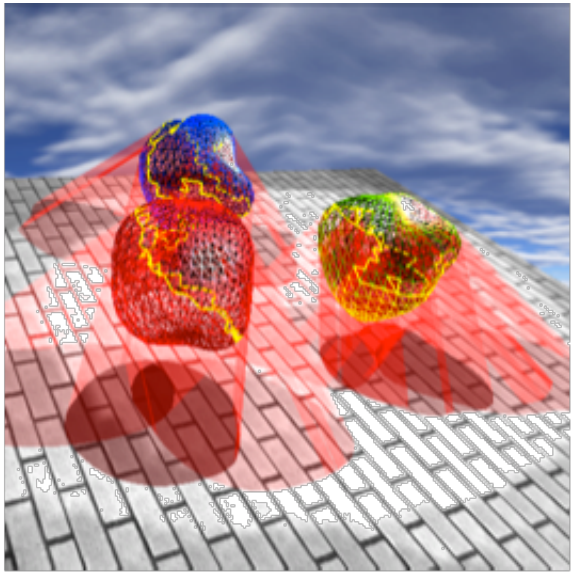
Shadow volumes on GPU

- Heidmann (IRIS Universe 1991), Everitt (nVidia 2002)
  - Shadow volume = triangles facing light + silhouette extrusions
- Brabec-Seidel (EUROGRAPHICS 2003)
  - Add silhouette identification in hardware



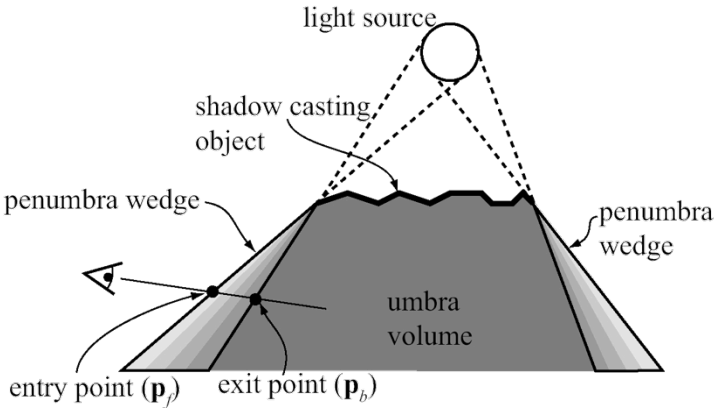


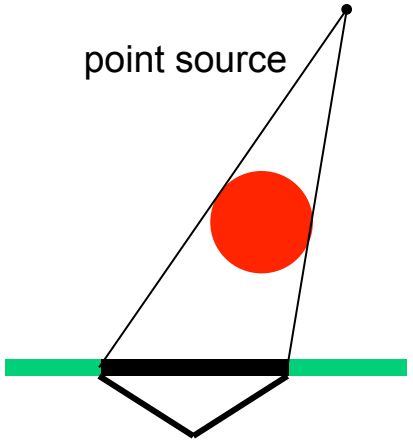


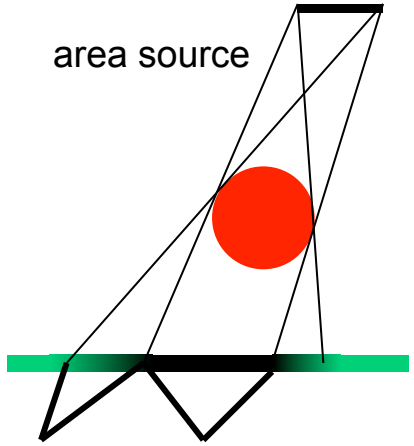


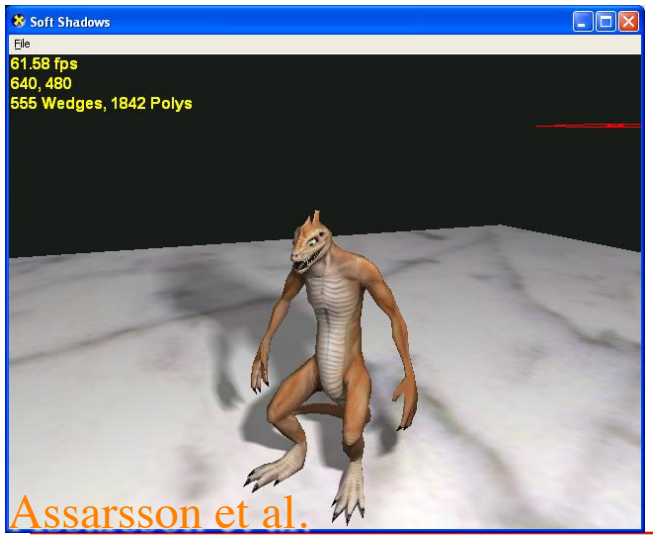
Soft shadows (area light sources)

- Polygonal area light source



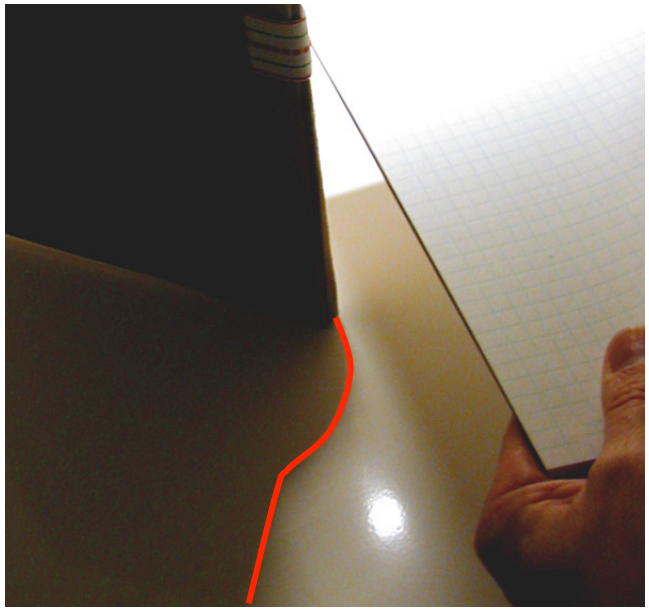






Assarsson et al.

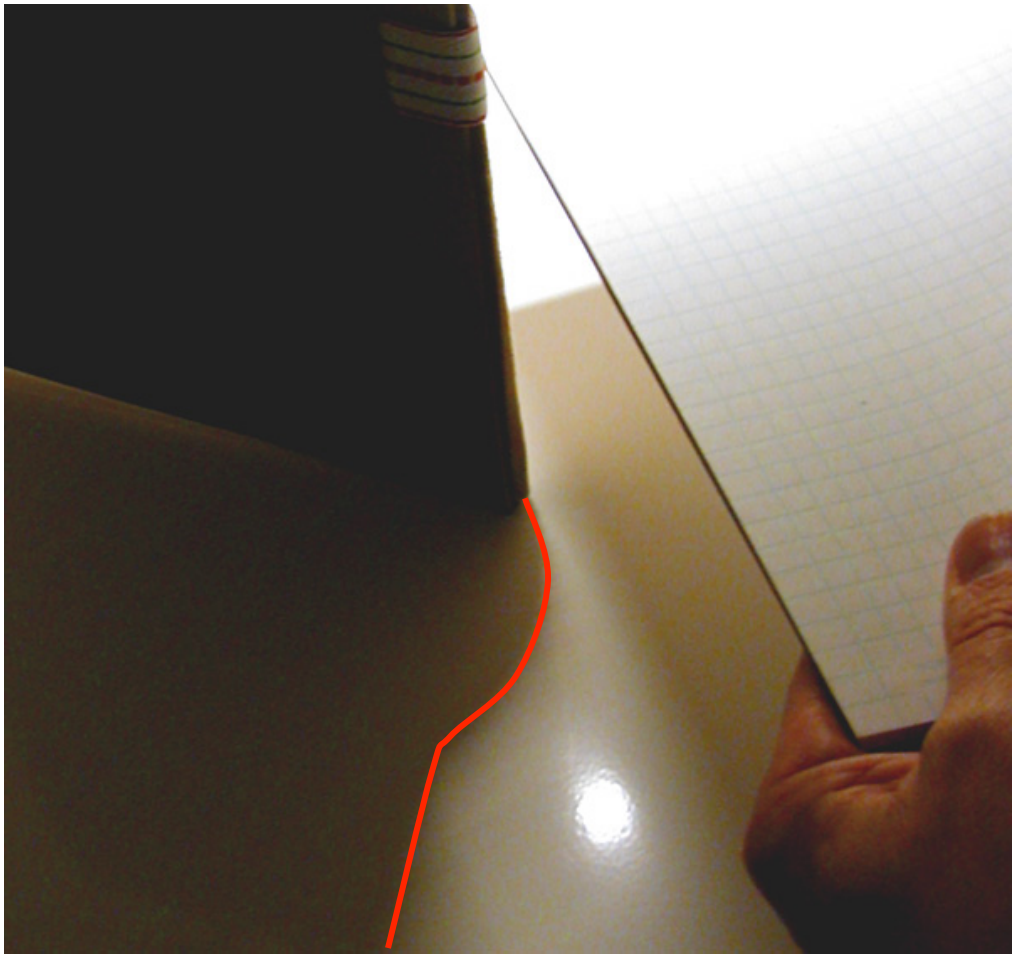
Polygons cast curved shadows



How can we compute (pen)umbras?

- Difficult because they involve cells of 3D space partition by planes and curved surfaces (even if the shape and light source is polygonal)

Polygons cast curved shadows!



Soft Shadow volumes

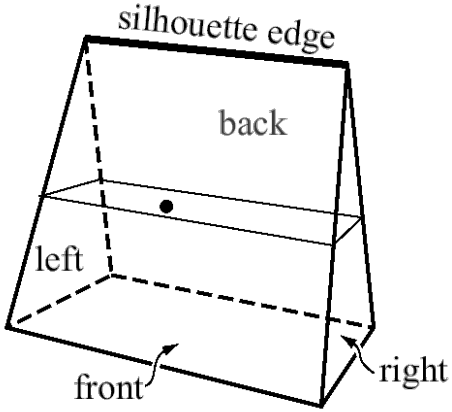
- Using graphics hardware

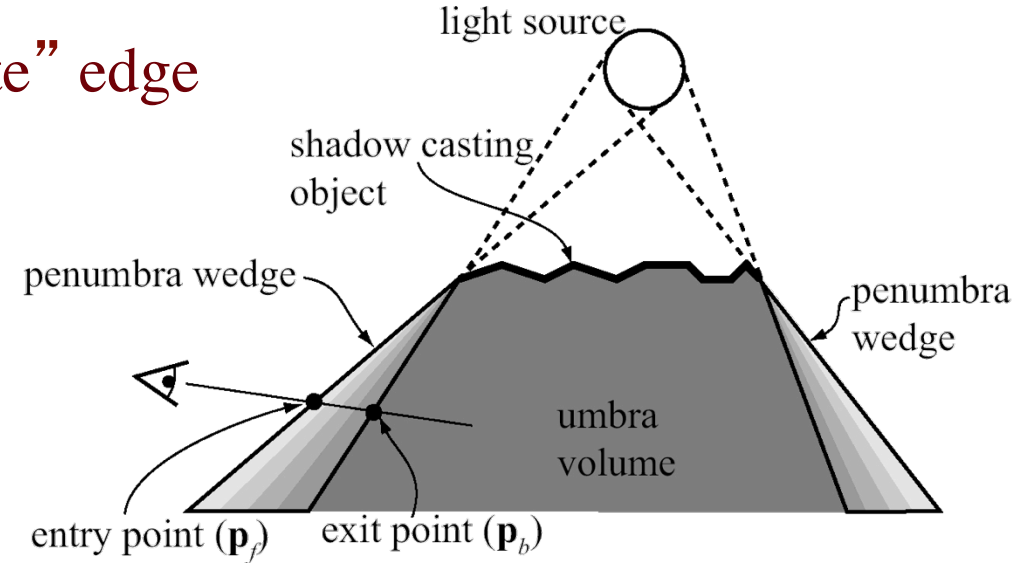
Ulf Assarsson[1],
Michael Dougherty[2],
Michael Mounier[2],
and Tomas Akenine-Möller[1]

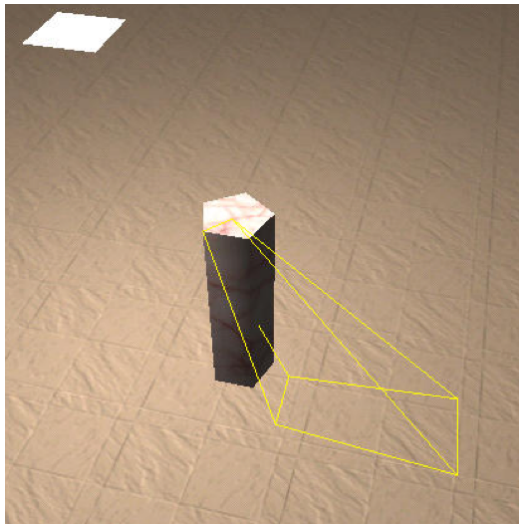
[1]Department of Computer Engineering
Chalmers University of Technology
[2]Xbox Advanced Technology Group, Microsoft

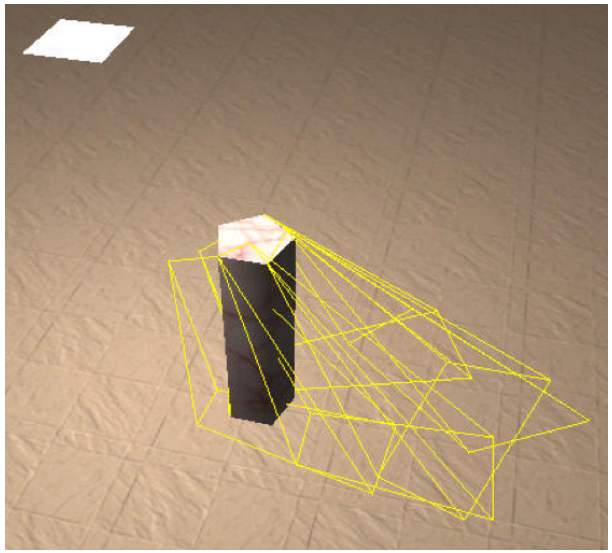
Shadow Volume

- A wedge for each "silhouette" edge

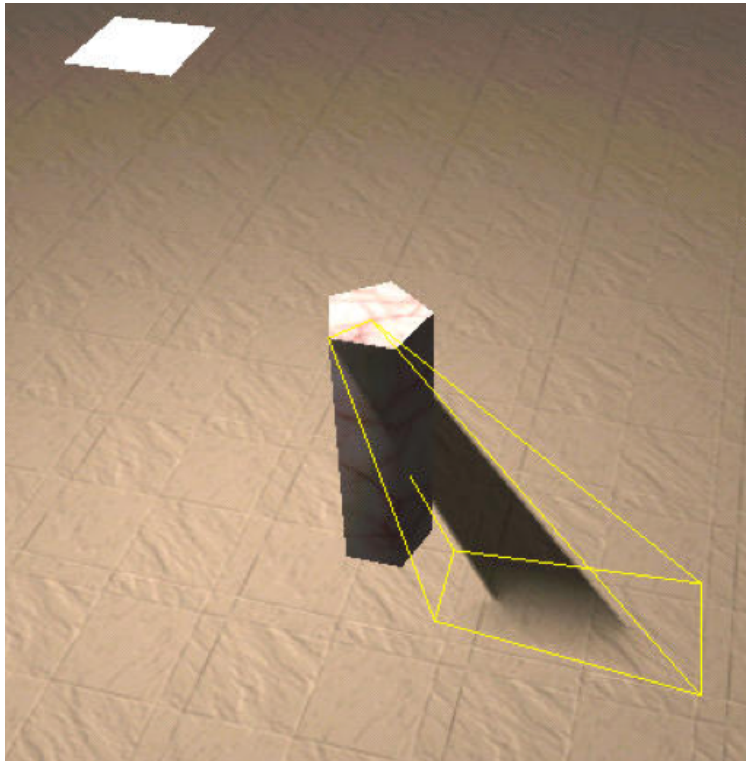


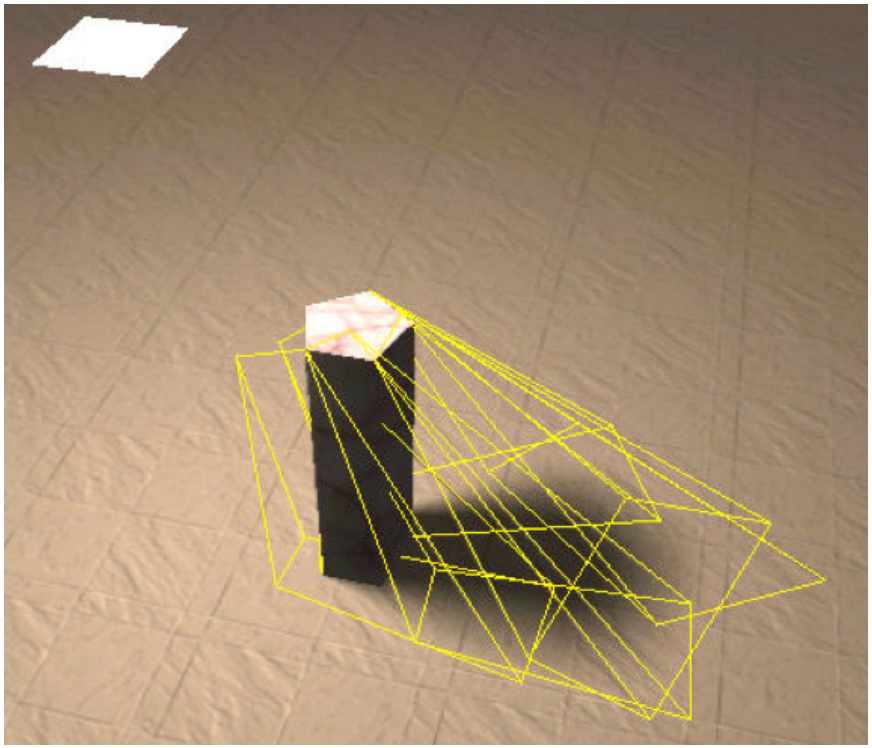






Rasterize the edges





Results

- www.ce.chalmers.se/staff/tomasm/soft/