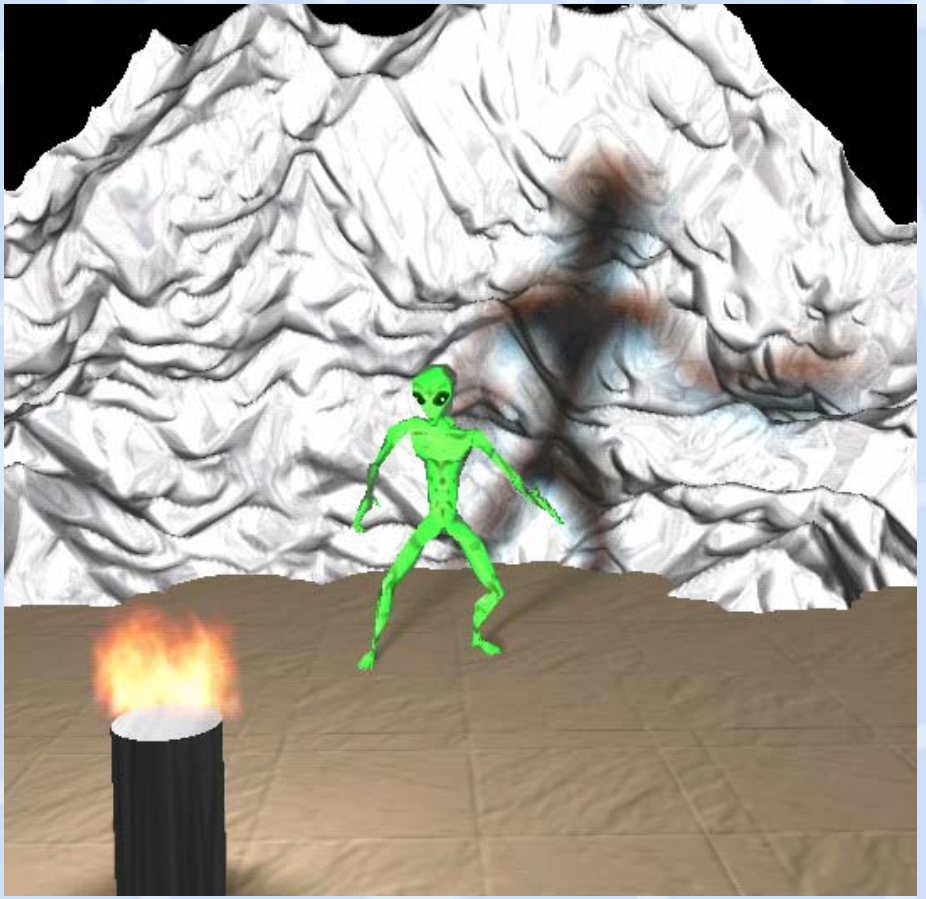
A Geometry-Based Soft Shadow Volume Algorithm Using Graphics Hardware

Ulf Assarsson
and
Tomas Akenine-Möller

SIGGRAPH 2003



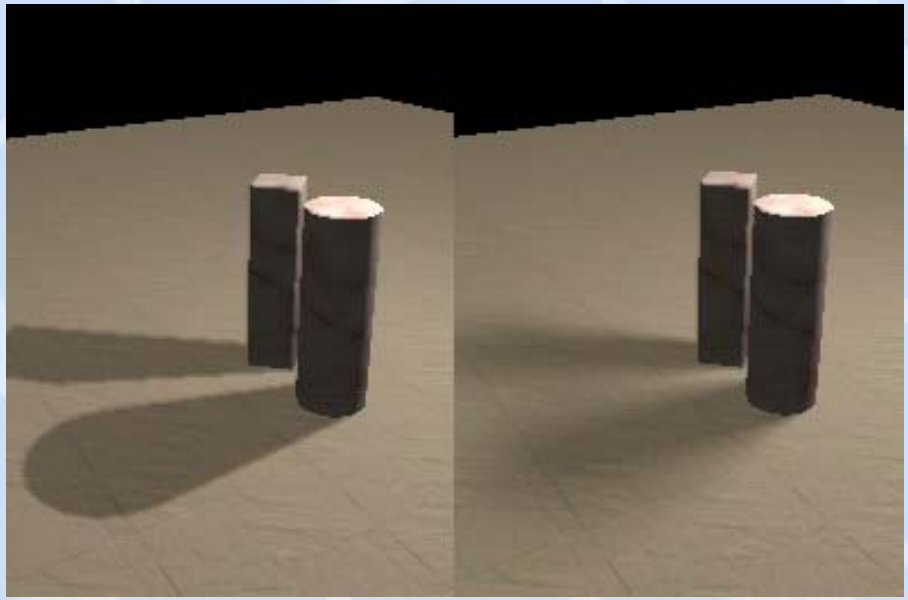
We present:

A Soft Shadow Volume Algorithm:

- Area light sources
- Simple volumetric light sources
- Textures and short video textures as lights
- Real-time performance with programmable graphics hardware
- Approximate soft shadows
  - Trade speed vs accuracy

Intro demo

A short recap…

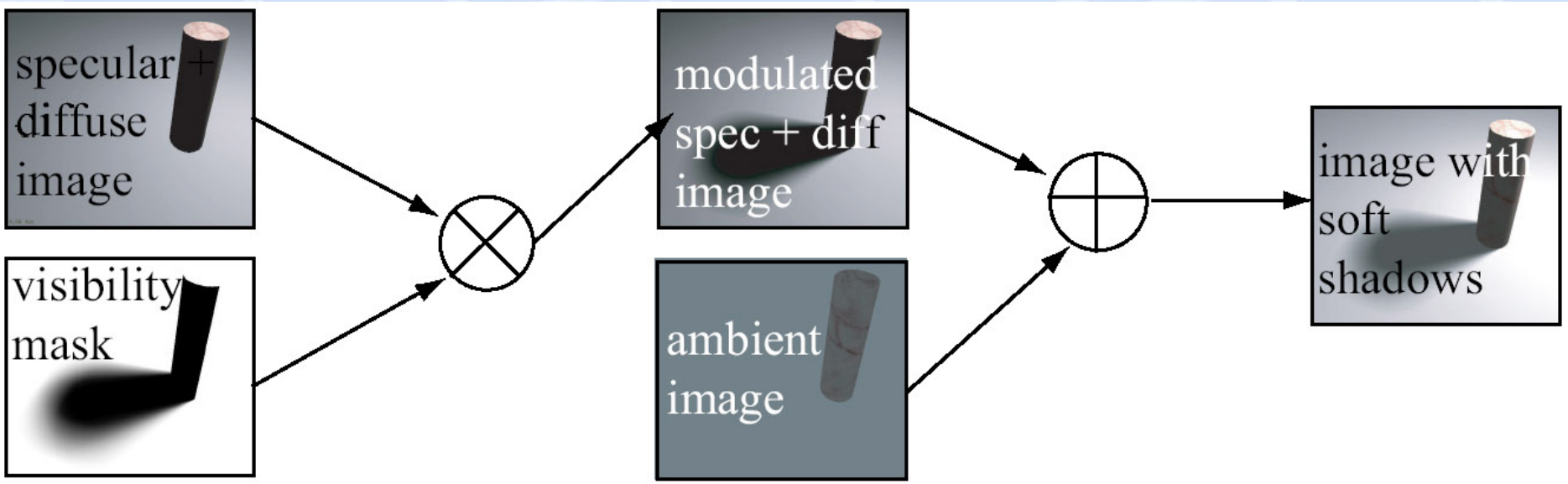


HARD SOFT

Area/volumetric lights give soft shadows

- Real lights have area or volume
- Thus, soft shadows more realistic

Overview

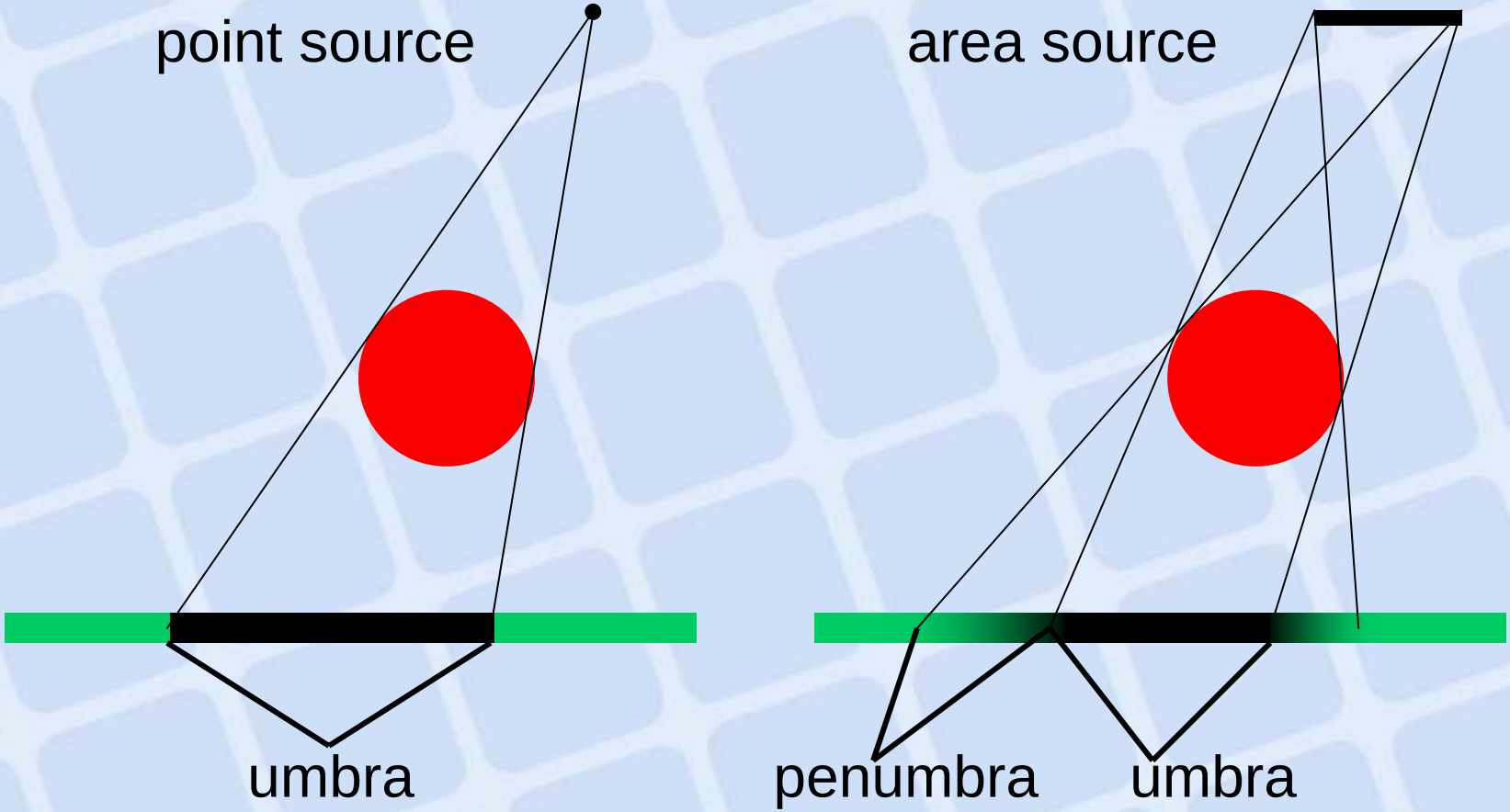


Computation of visibility mask:

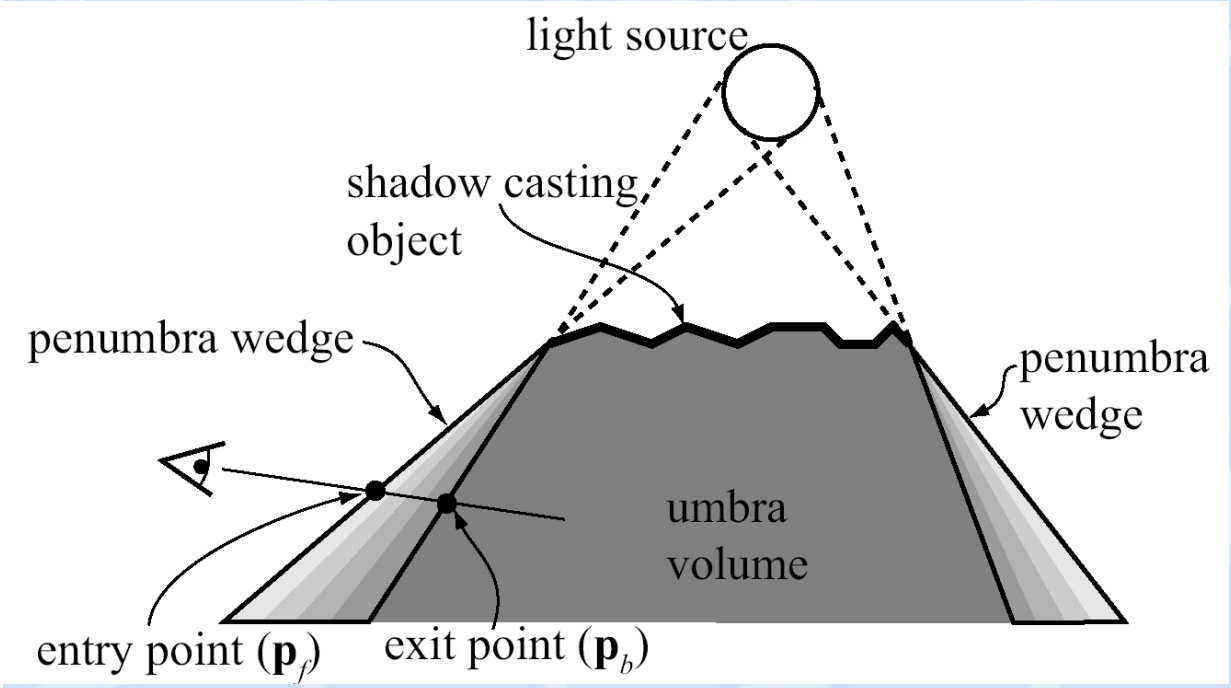
- 1st pass: Render hard shadow
- 2nd pass: compensate for overstated umbra

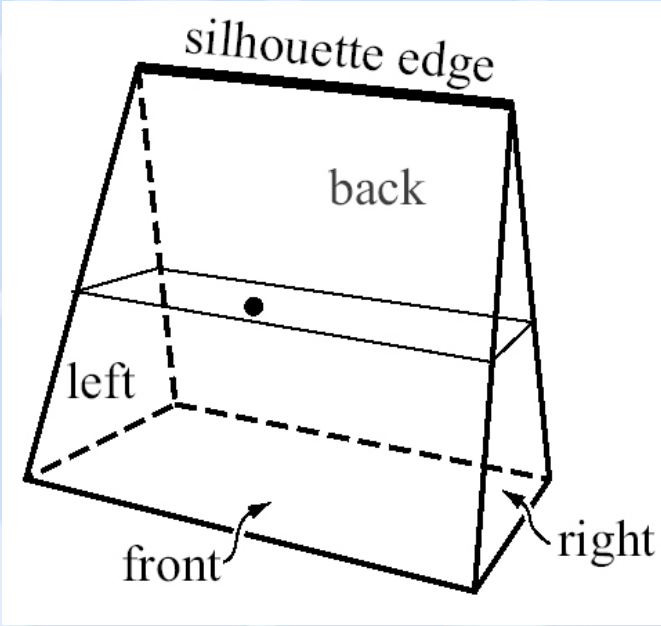
Hard vs. soft shadows

Two different light source types:



A Real-Time Soft Shadow Volume Algorithm

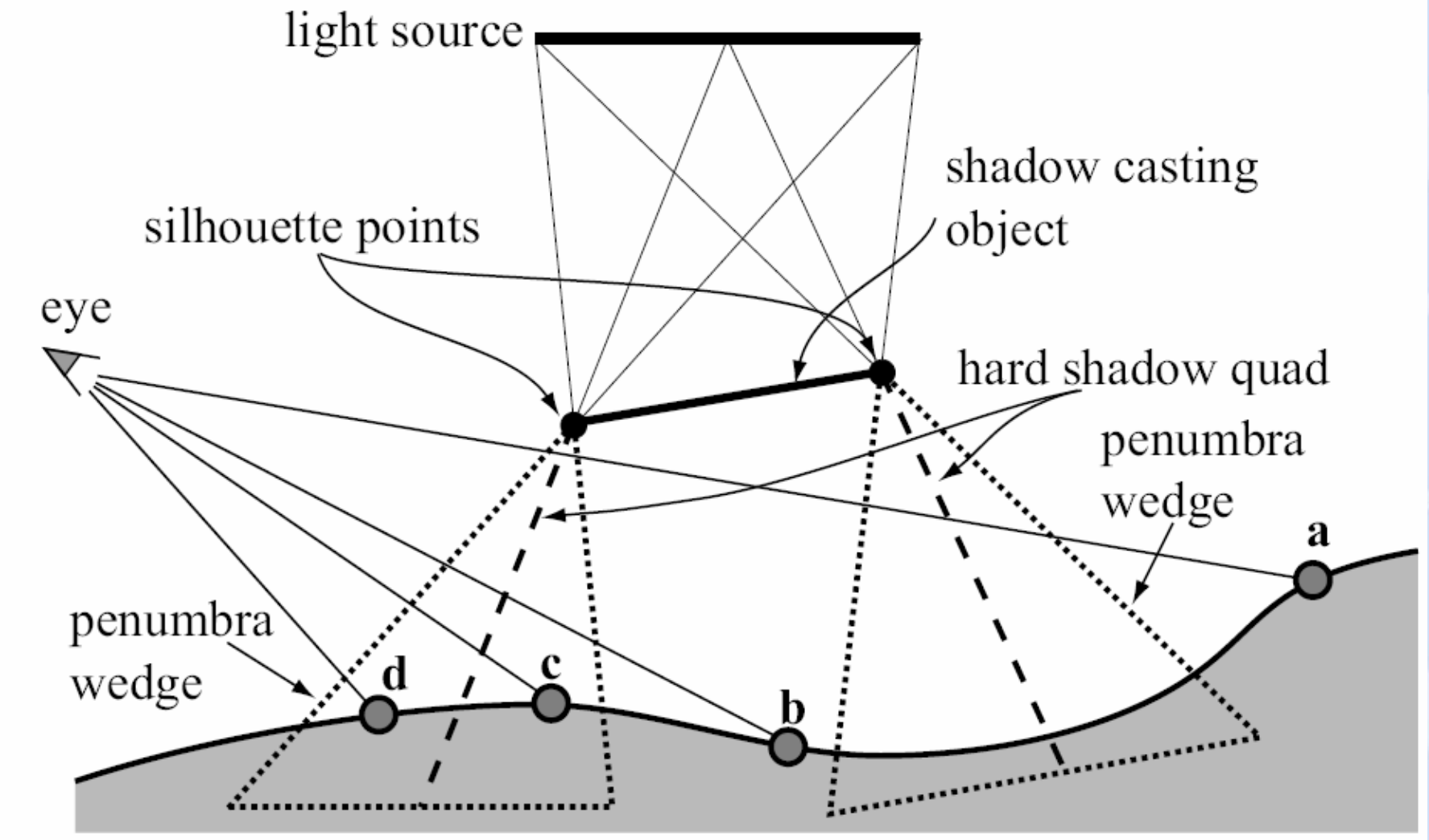




Wedges

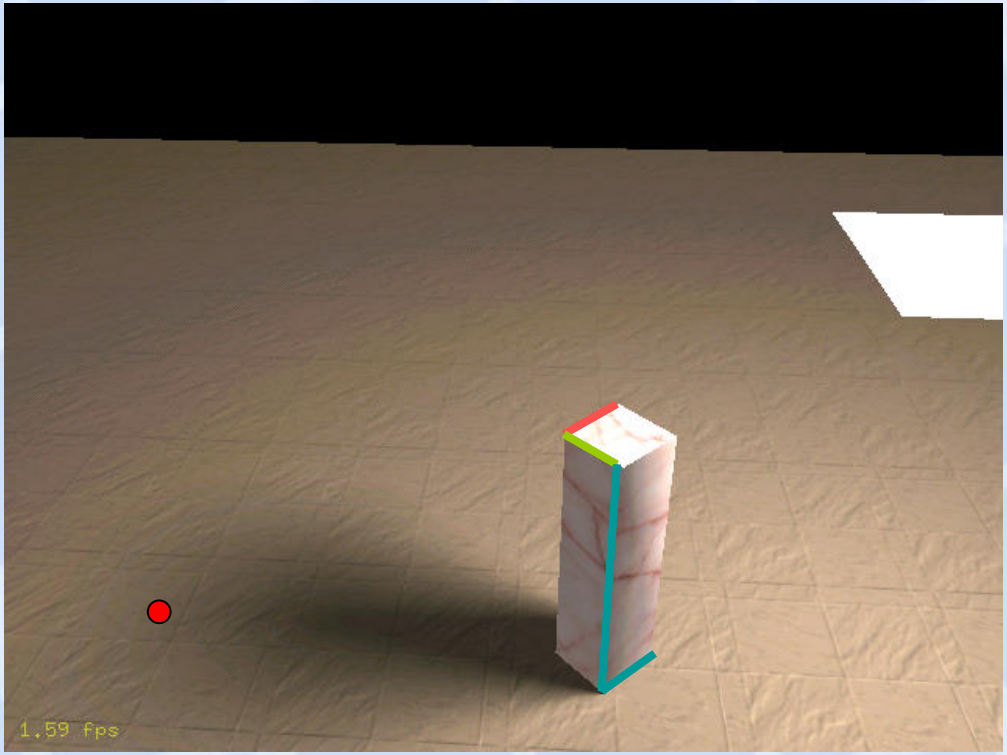
- Each silhouette edge has a corresponding wedge
  - Provides a piece of penumbra contribution
  - Rasterized by pixelshader

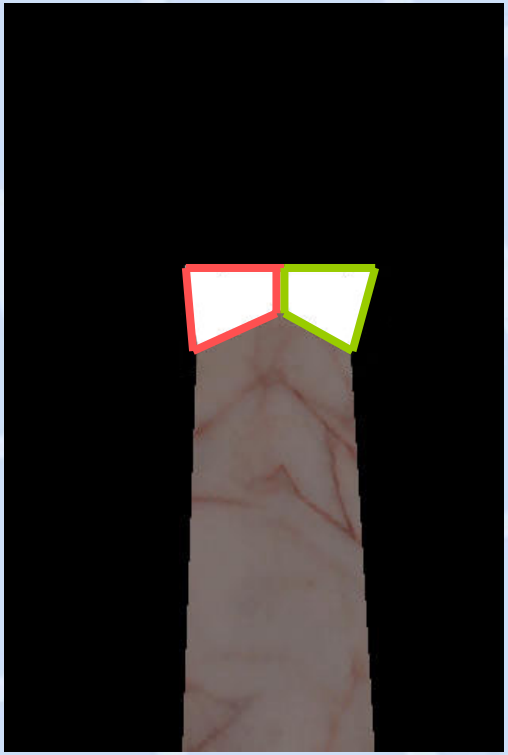
Two-pass algorithm



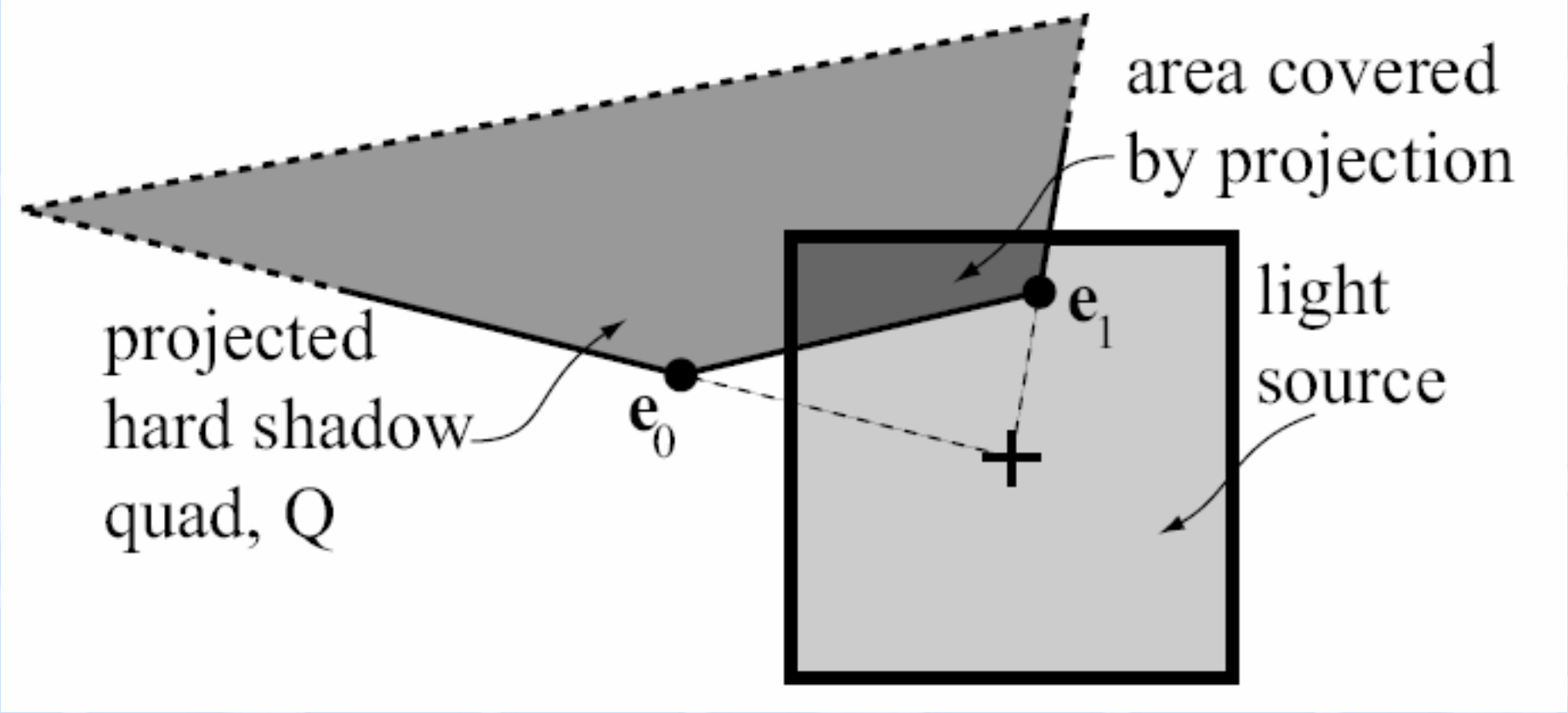
Visibility computation

- Really want to compute how much we can see of the light source

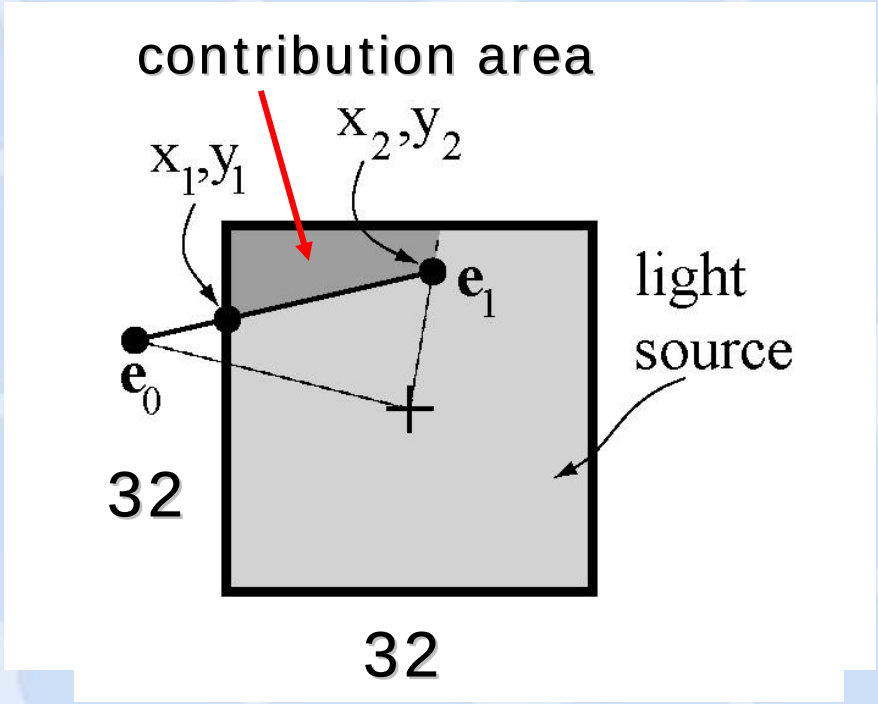


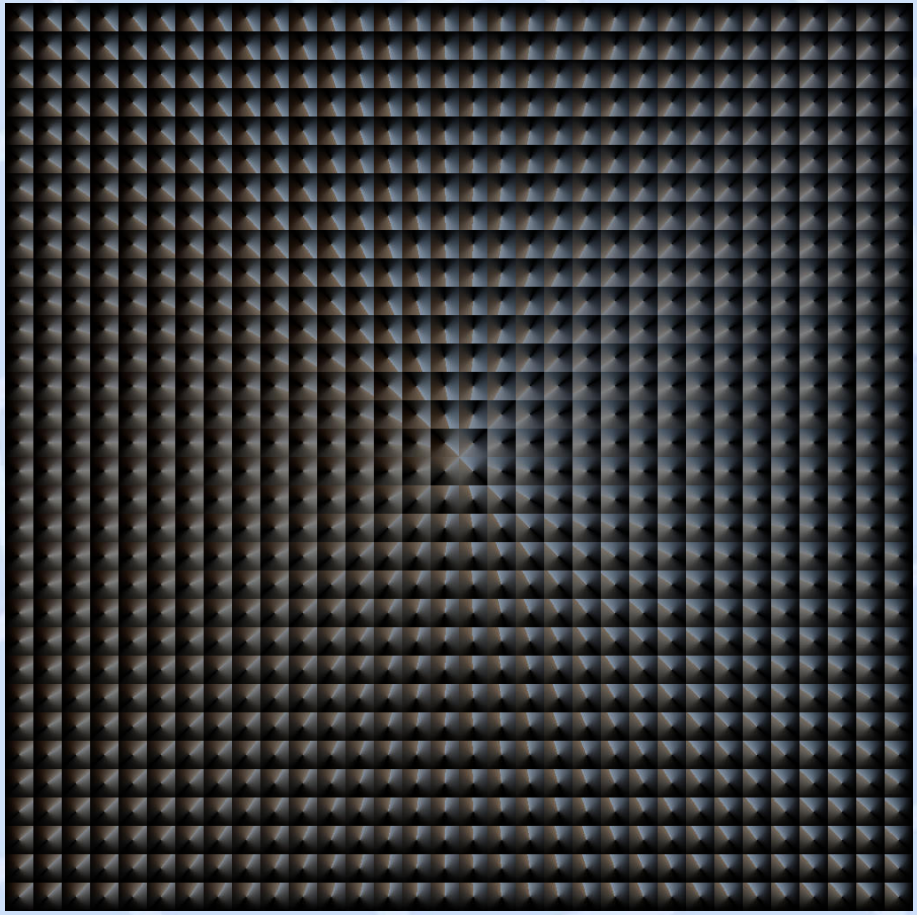


Visibility computation

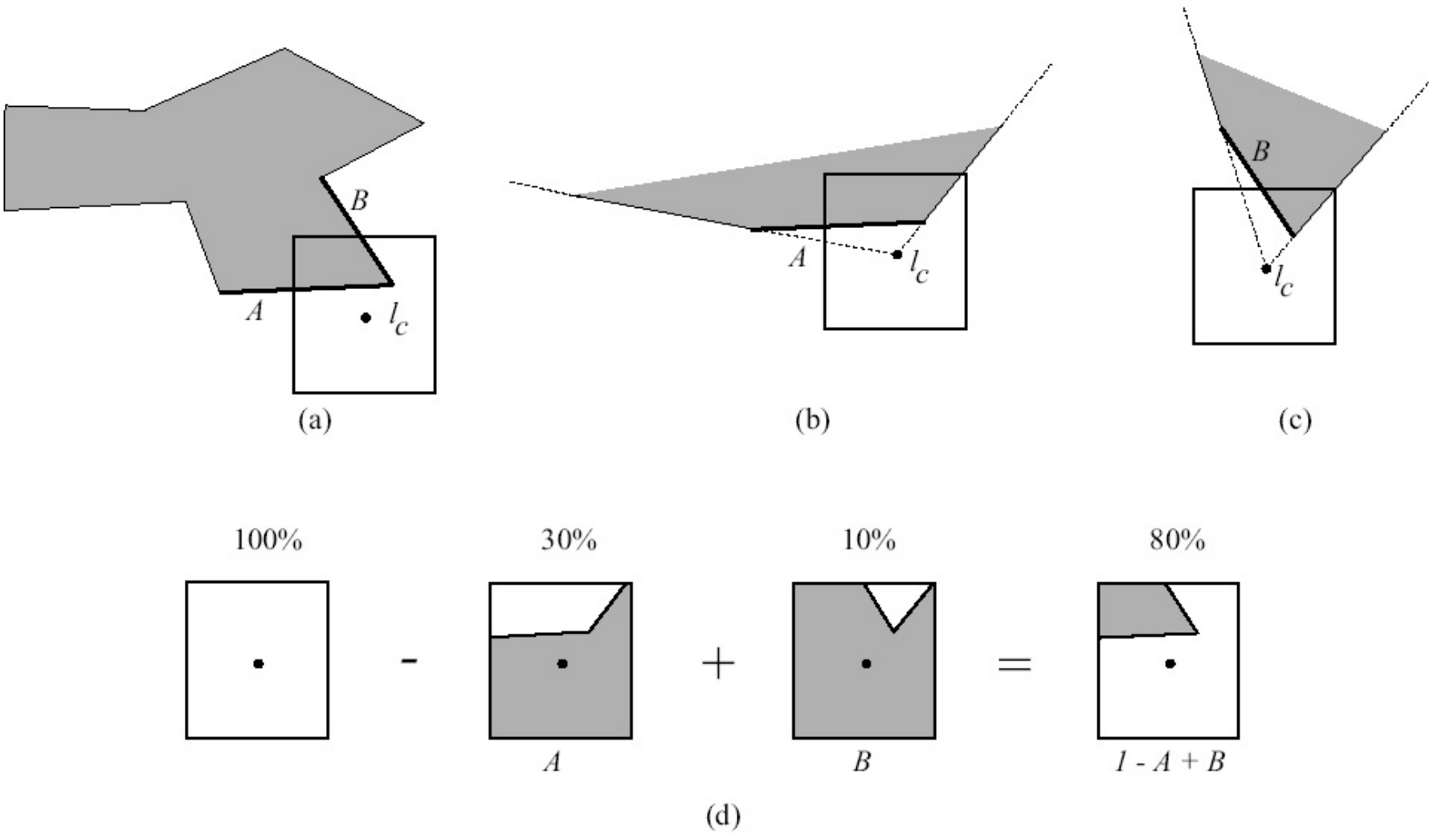


Precomputed contribution in 4D textures





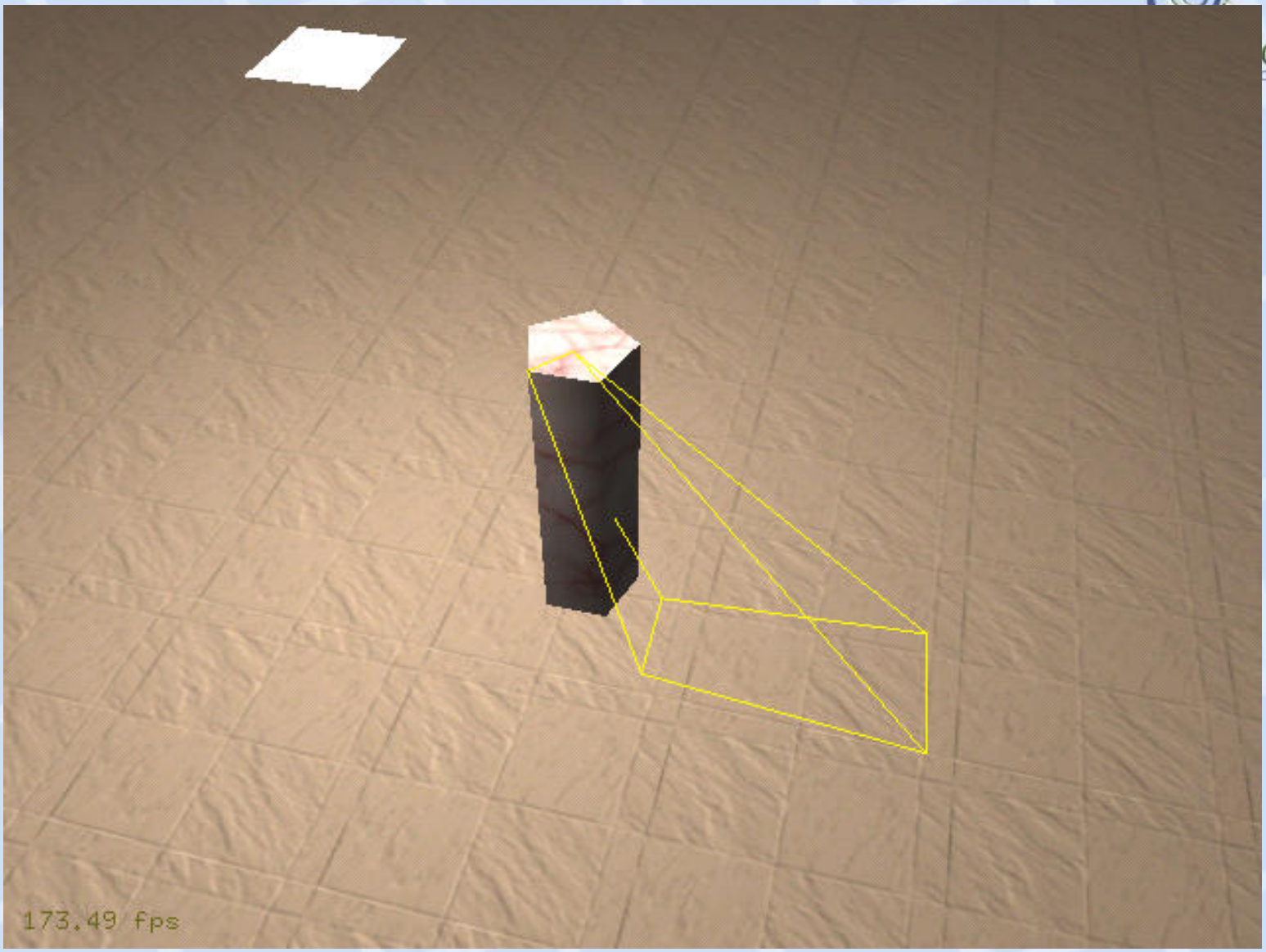
How the visibility computation works:



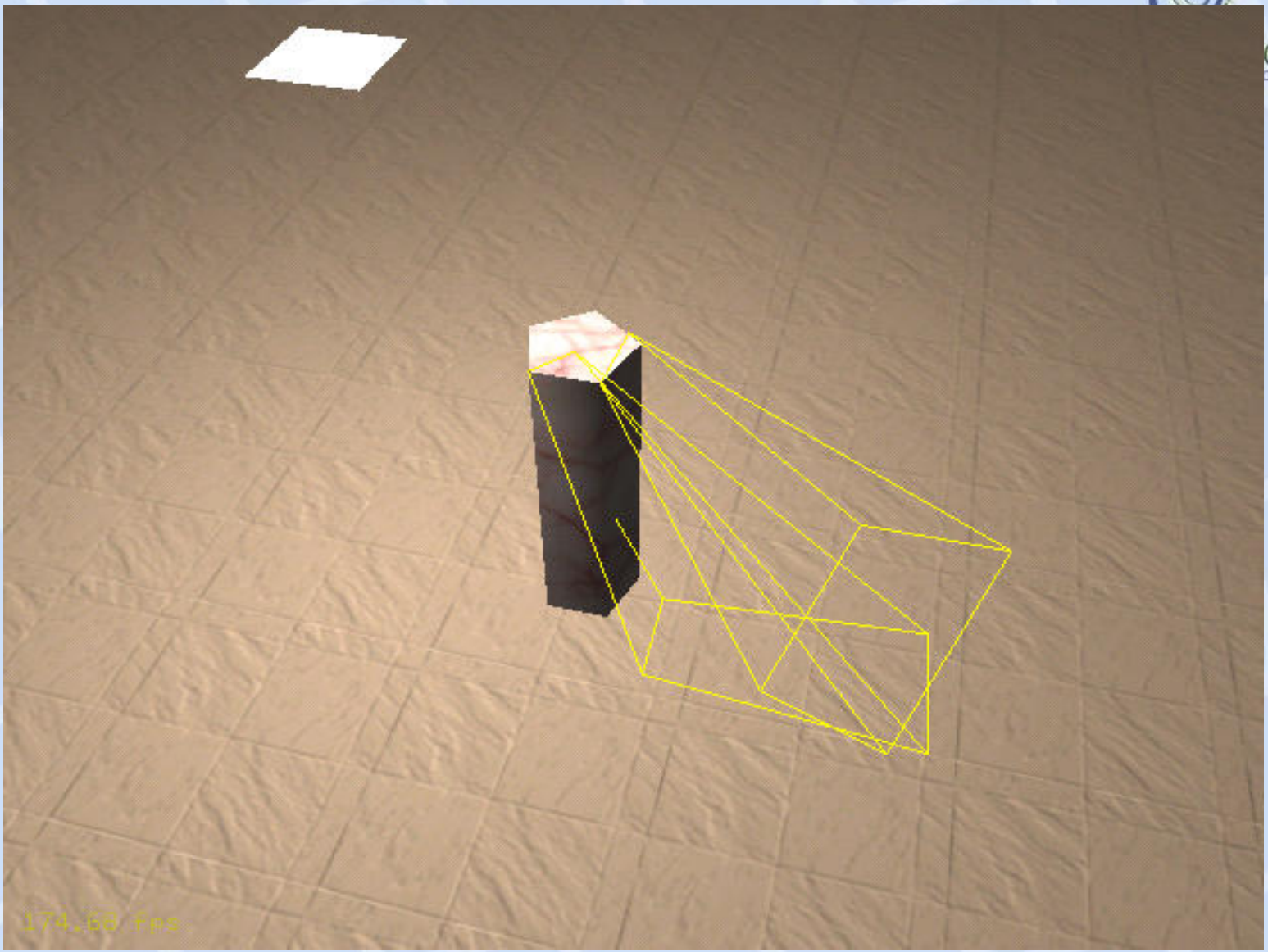
Rasterize a wedge

rasterizeWedge(wedge W, hard shadow quad Q, light L)
for each pixel (x, y) covered by front facing triangles of wedge
  $\mathbf{p} = \text{point}(x, y, z)$; // z is depth buffer value
  if $\mathbf{p}$ is inside the wedge
    $v_{\mathbf{p}} = \text{projectQuadAndComputeCoverage}(W, \mathbf{p}, Q)$;
    if $\mathbf{p}$ is in positive half space of Q
      $\overline{v}(x, y) = \overline{v}(x, y) - v_{\mathbf{p}}$; // update V–buffer
    else
      $\overline{v}(x, y) = \overline{v}(x, y) + v_{\mathbf{p}}$; // update V–buffer
    end;
  end;
end;

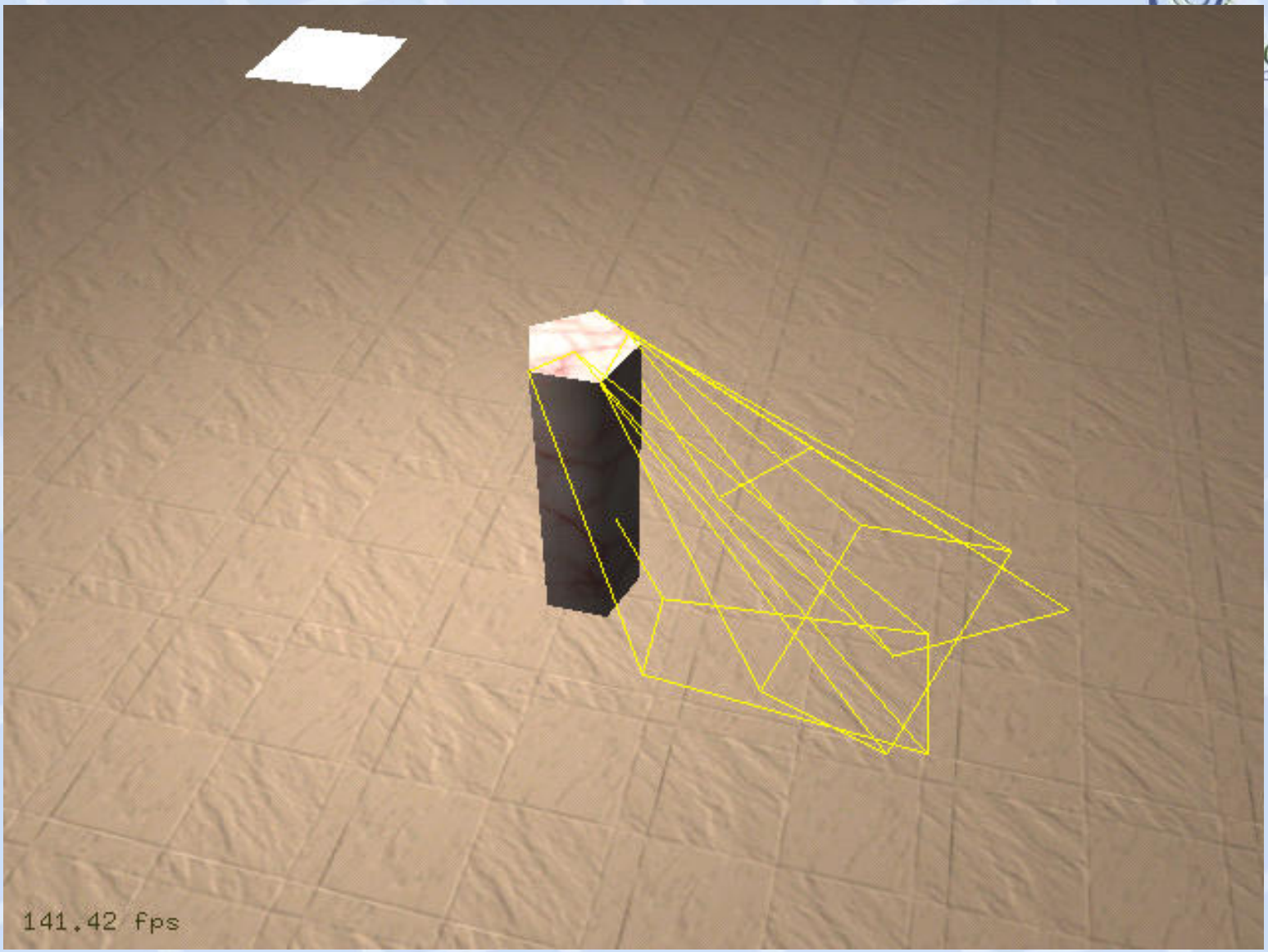
A wedge for each silhouette edge…



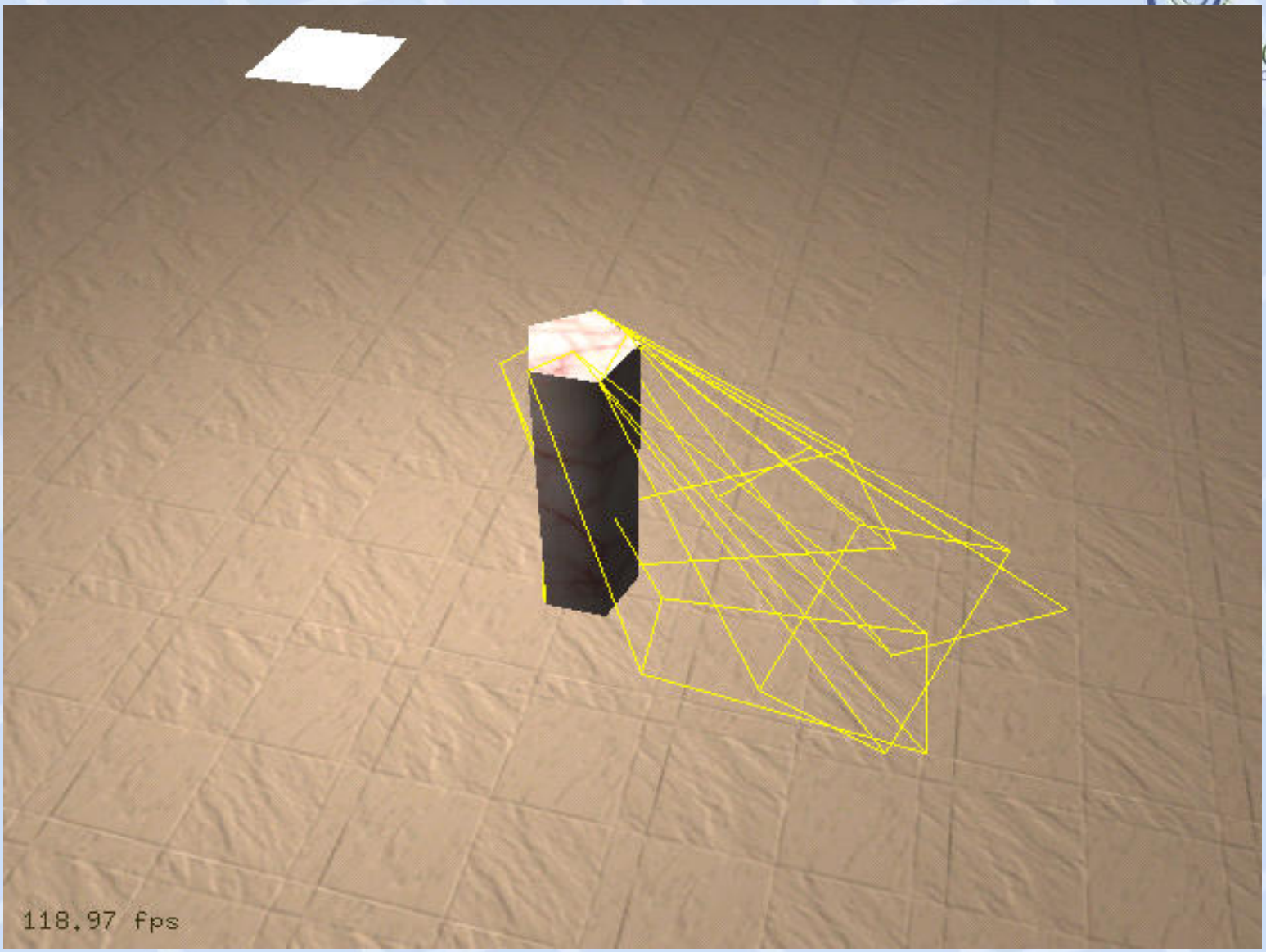
A wedge for each silhouette edge…



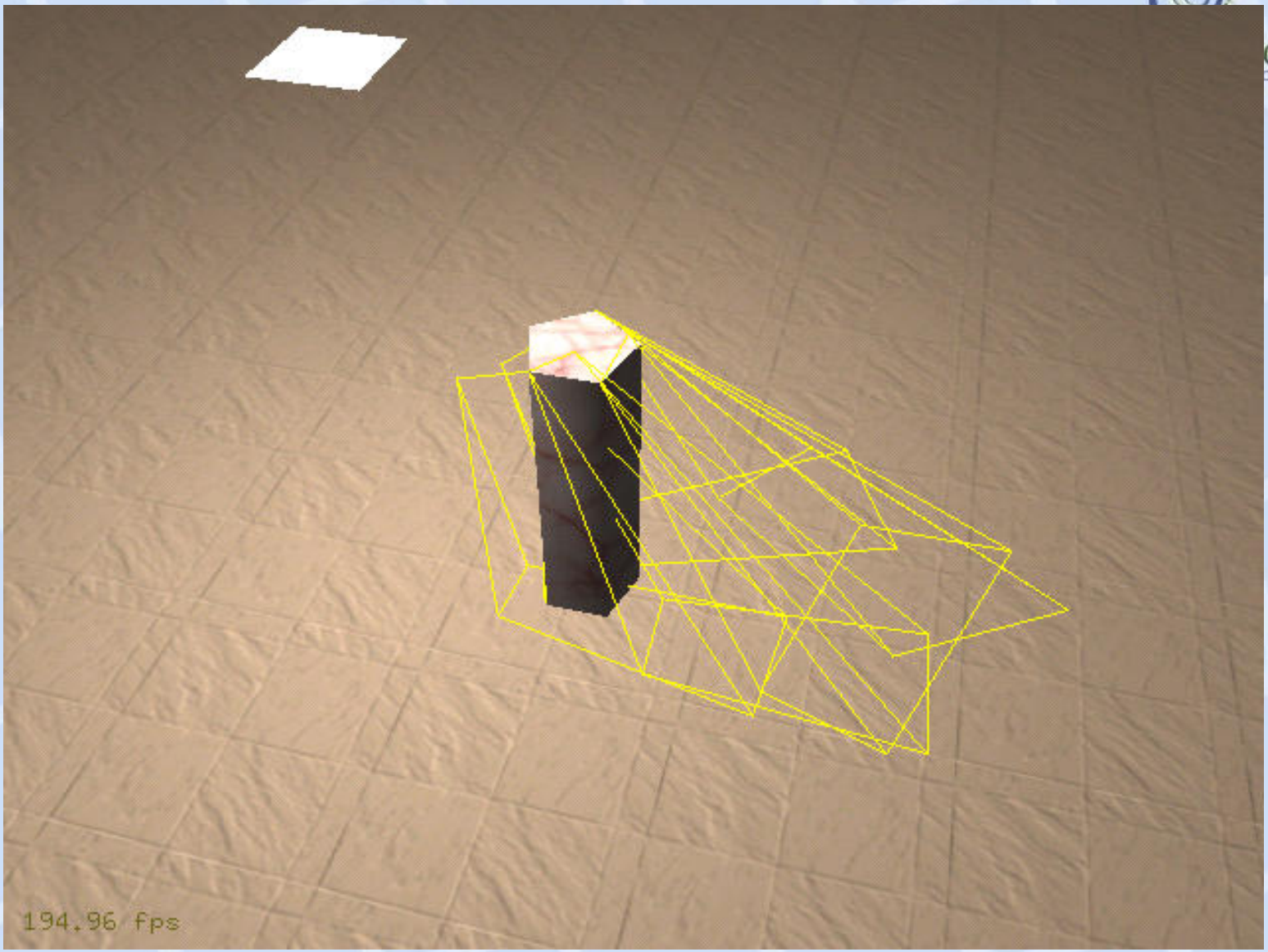
A wedge for each silhouette edge…



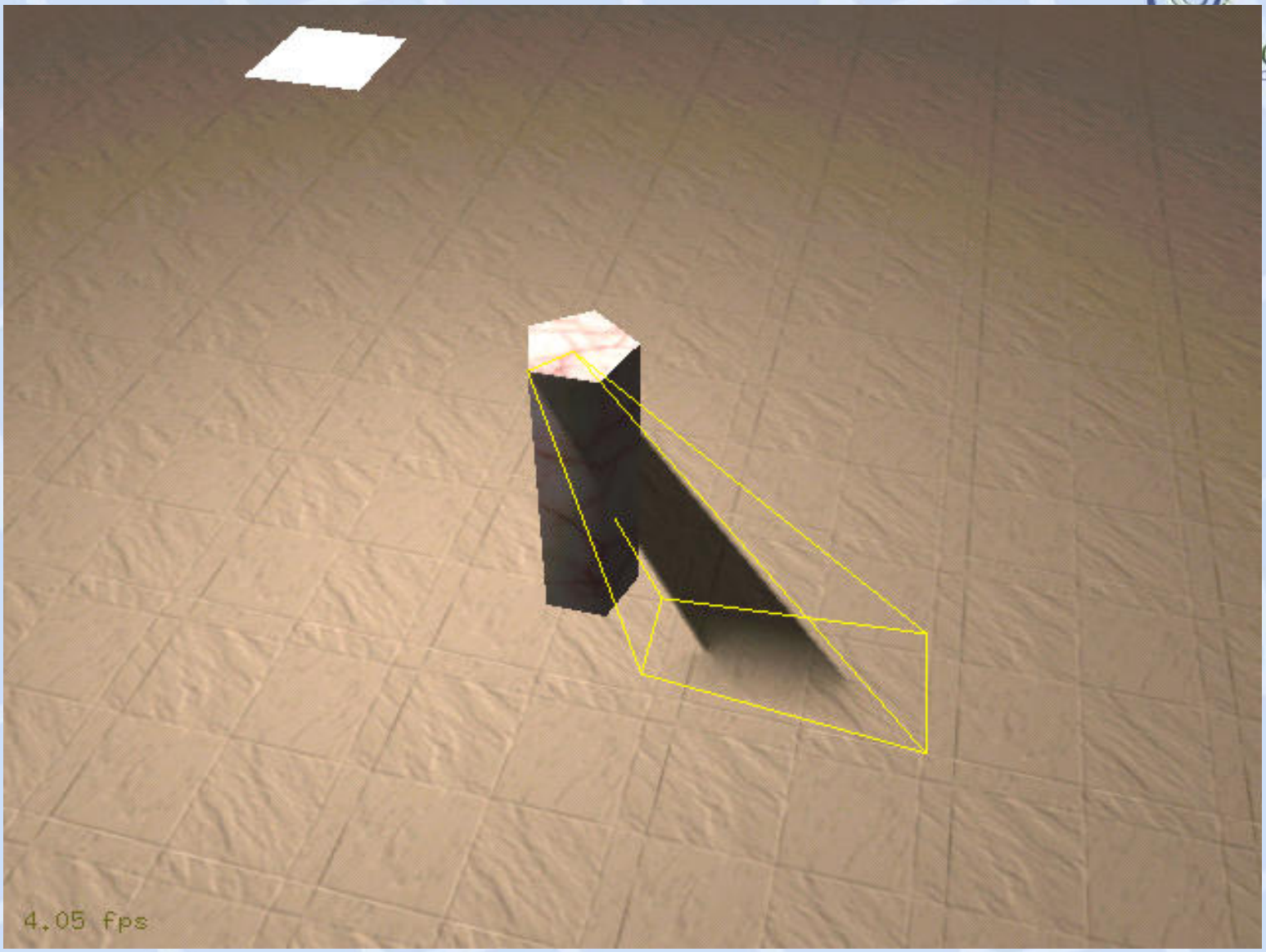
A wedge for each silhouette edge…



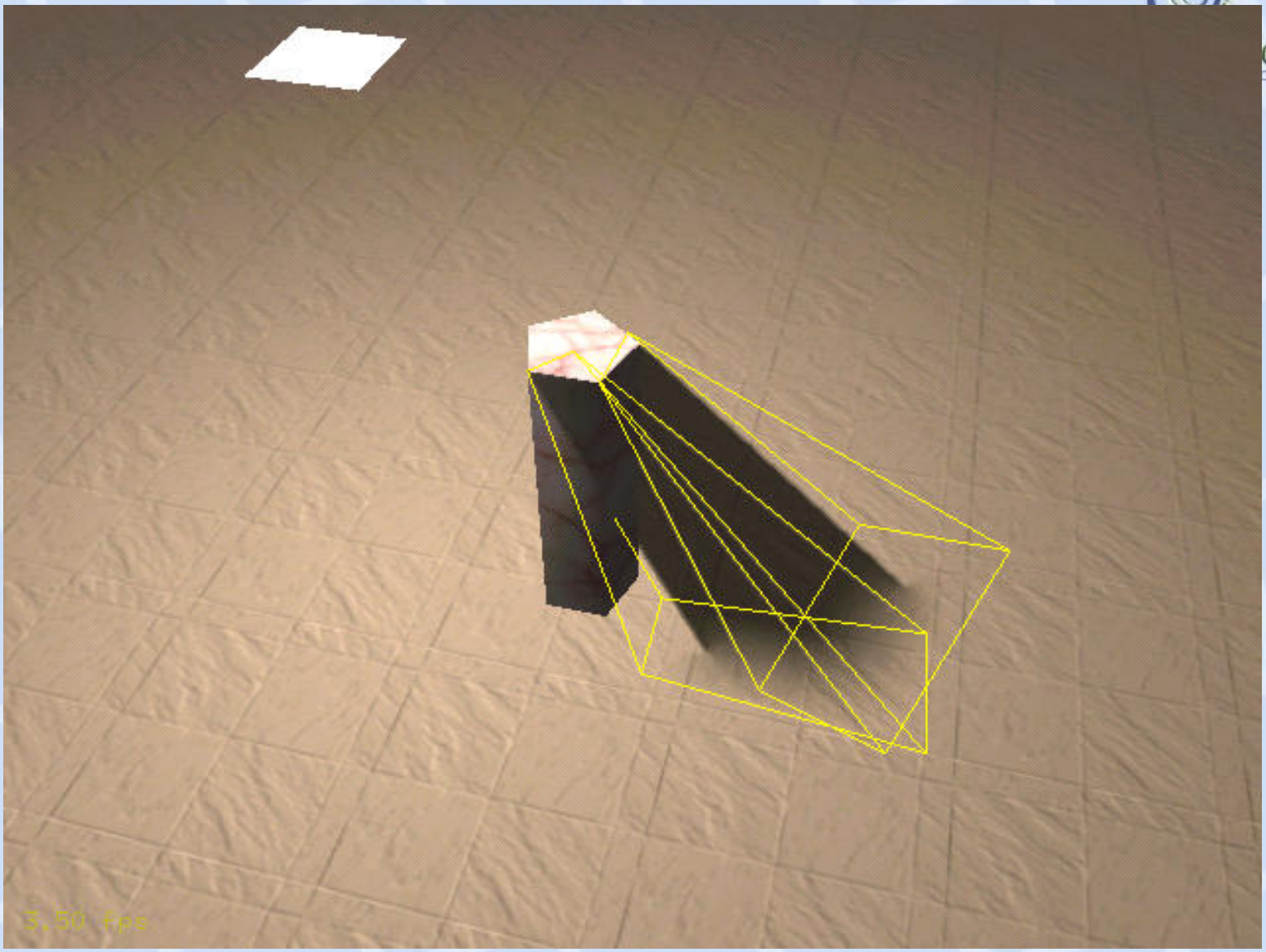
A wedge for each silhouette edge…



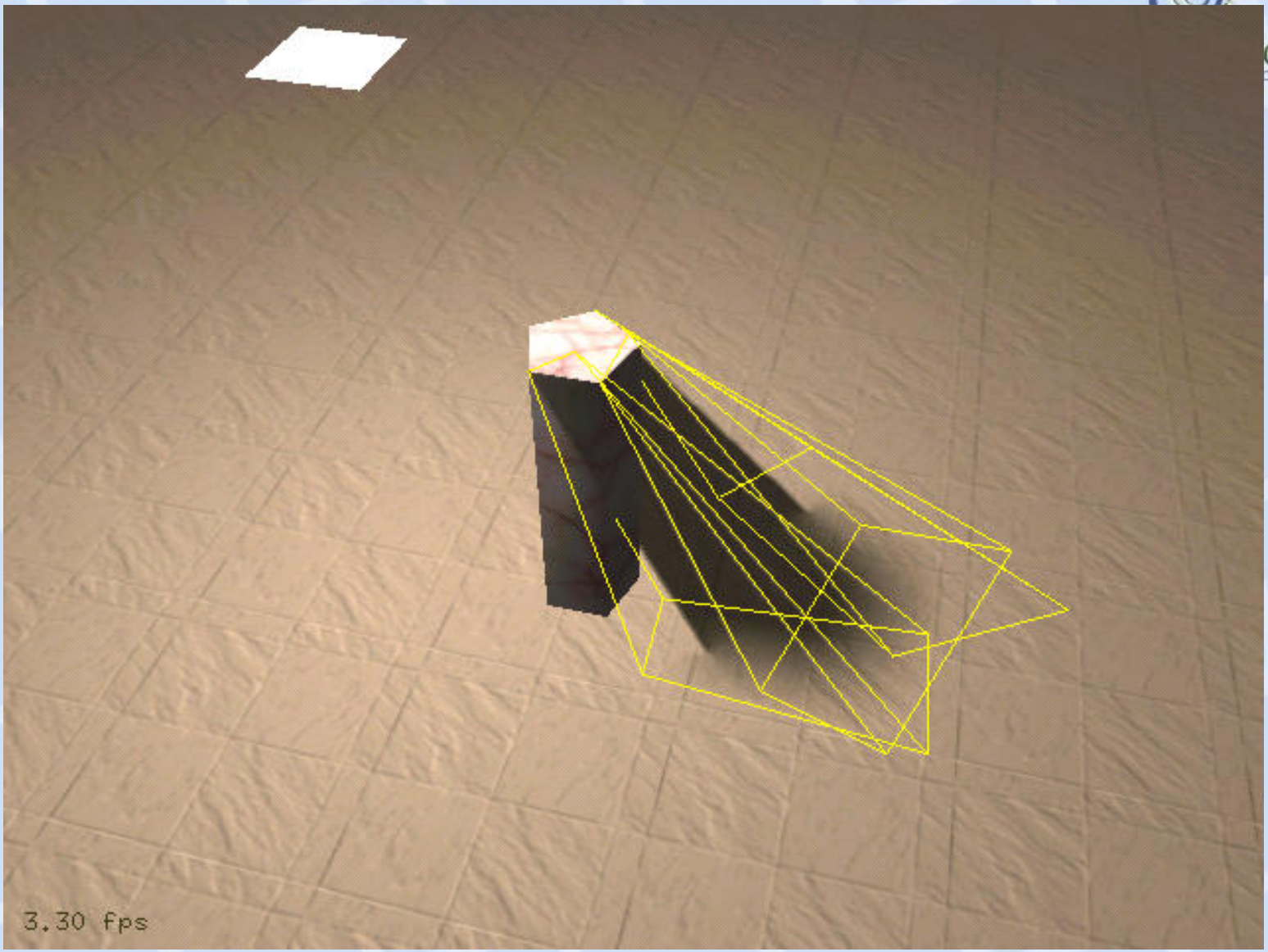
Rasterizing the wedges



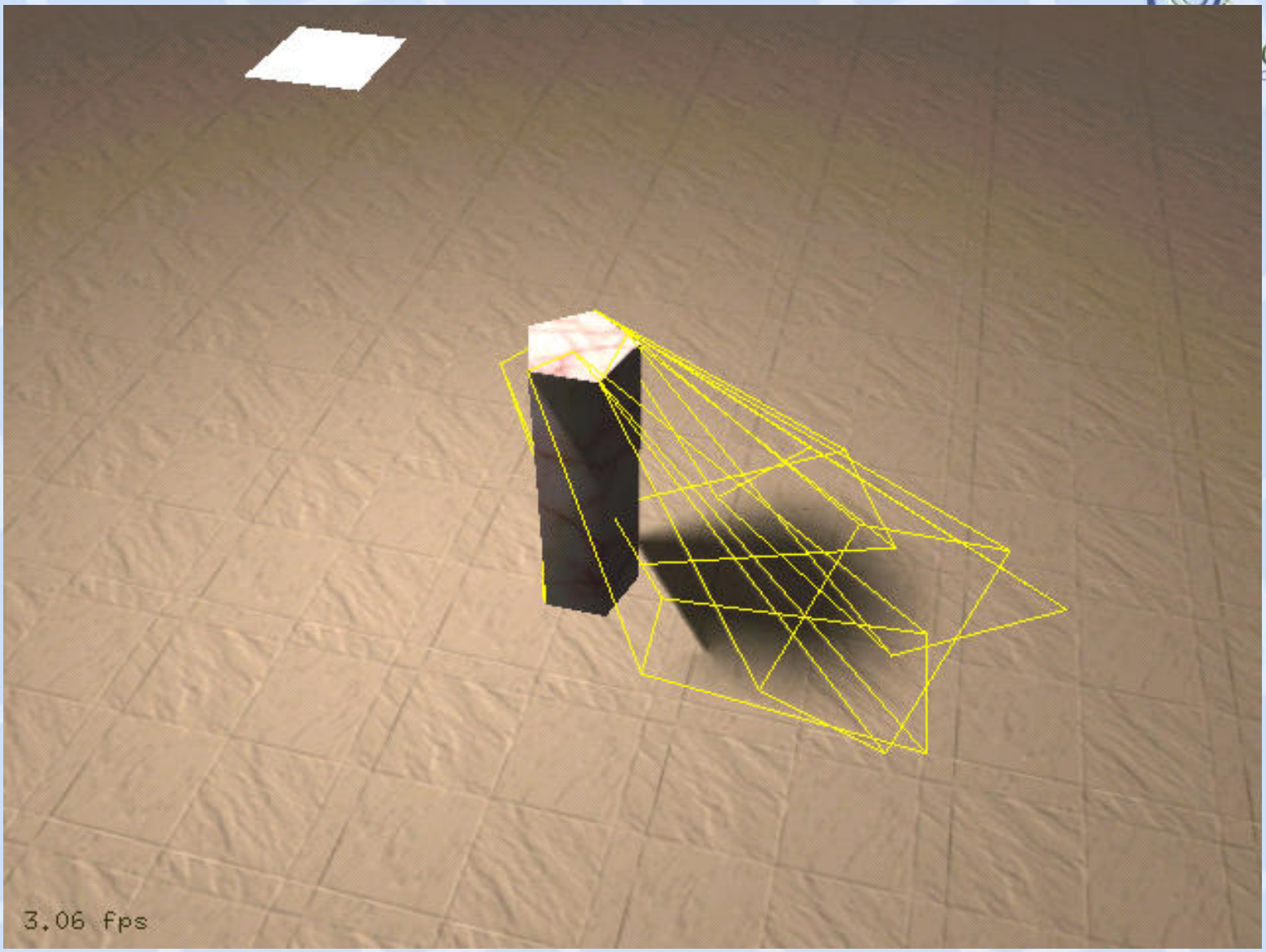
Rasterizing the wedges



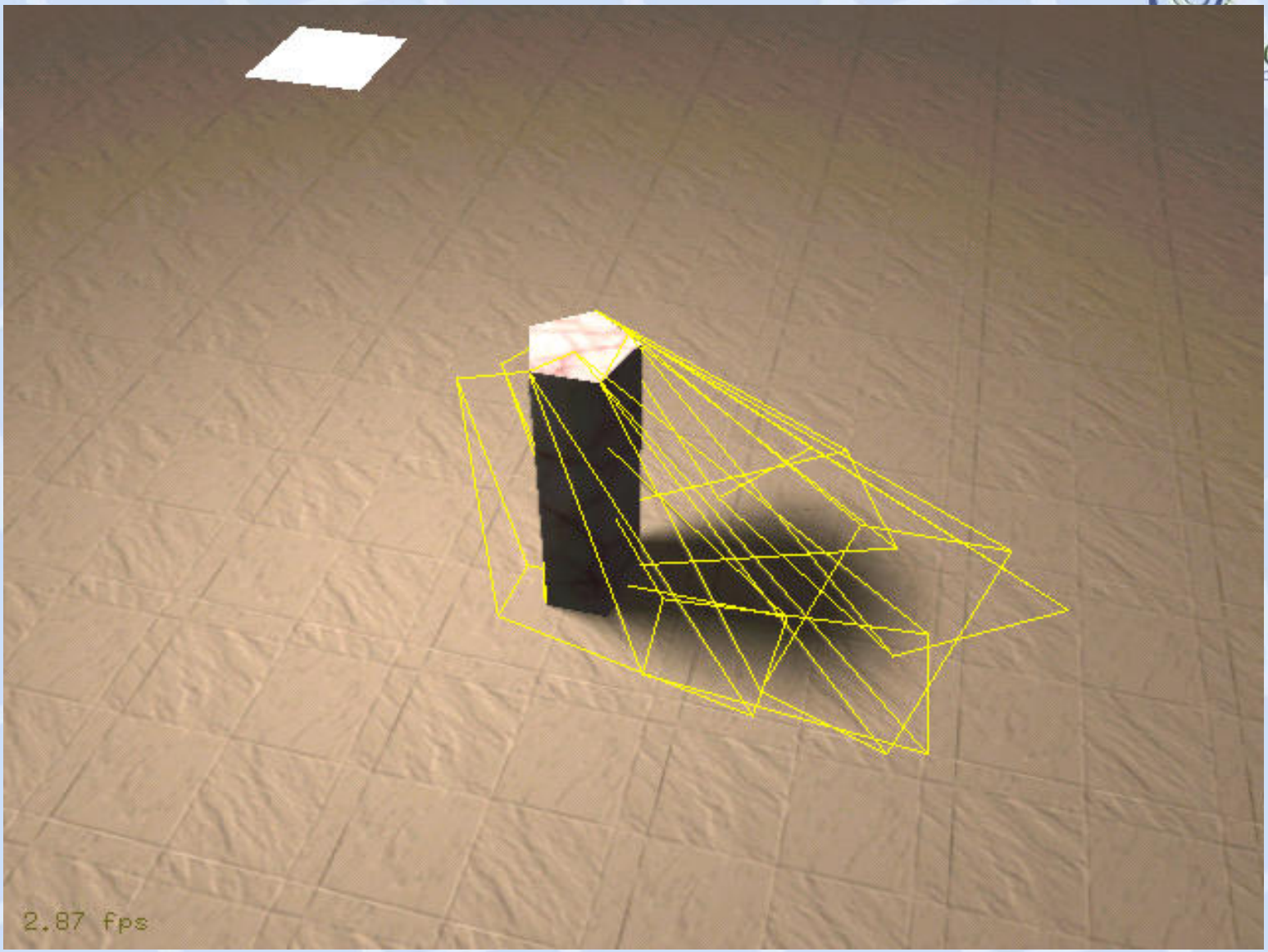
Rasterizing the wedges



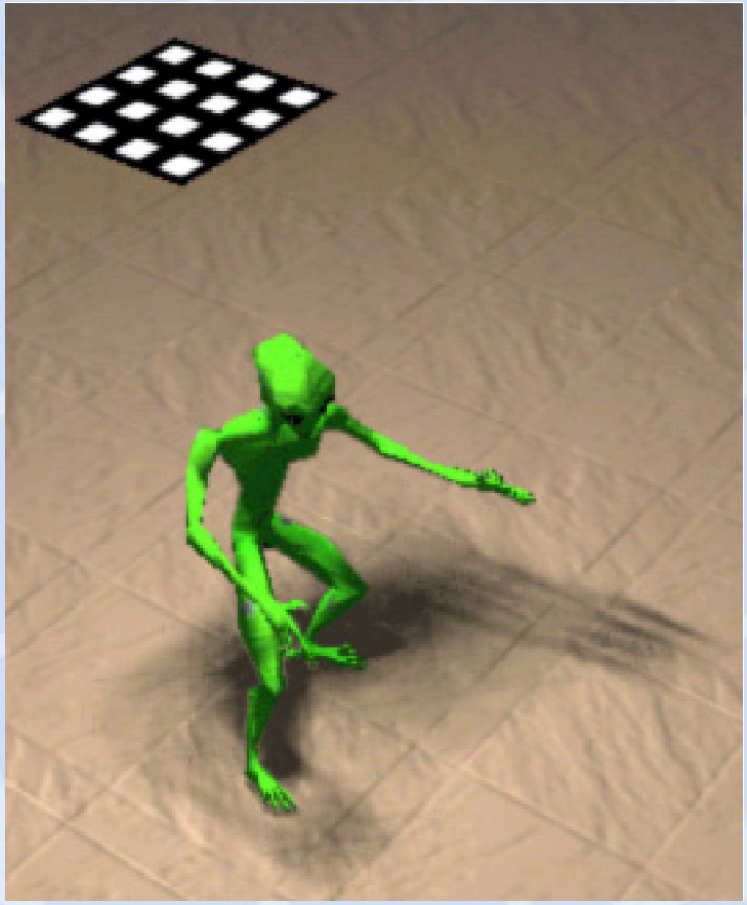
Rasterizing the wedges



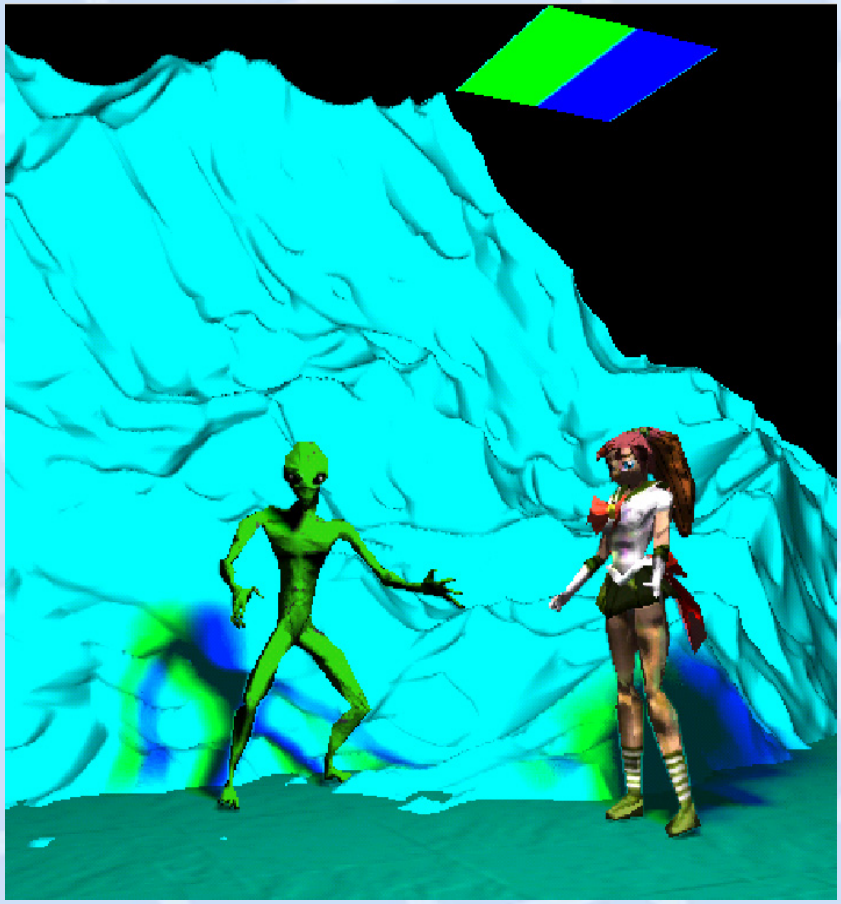
Rasterizing the wedges



Examples using textured lights



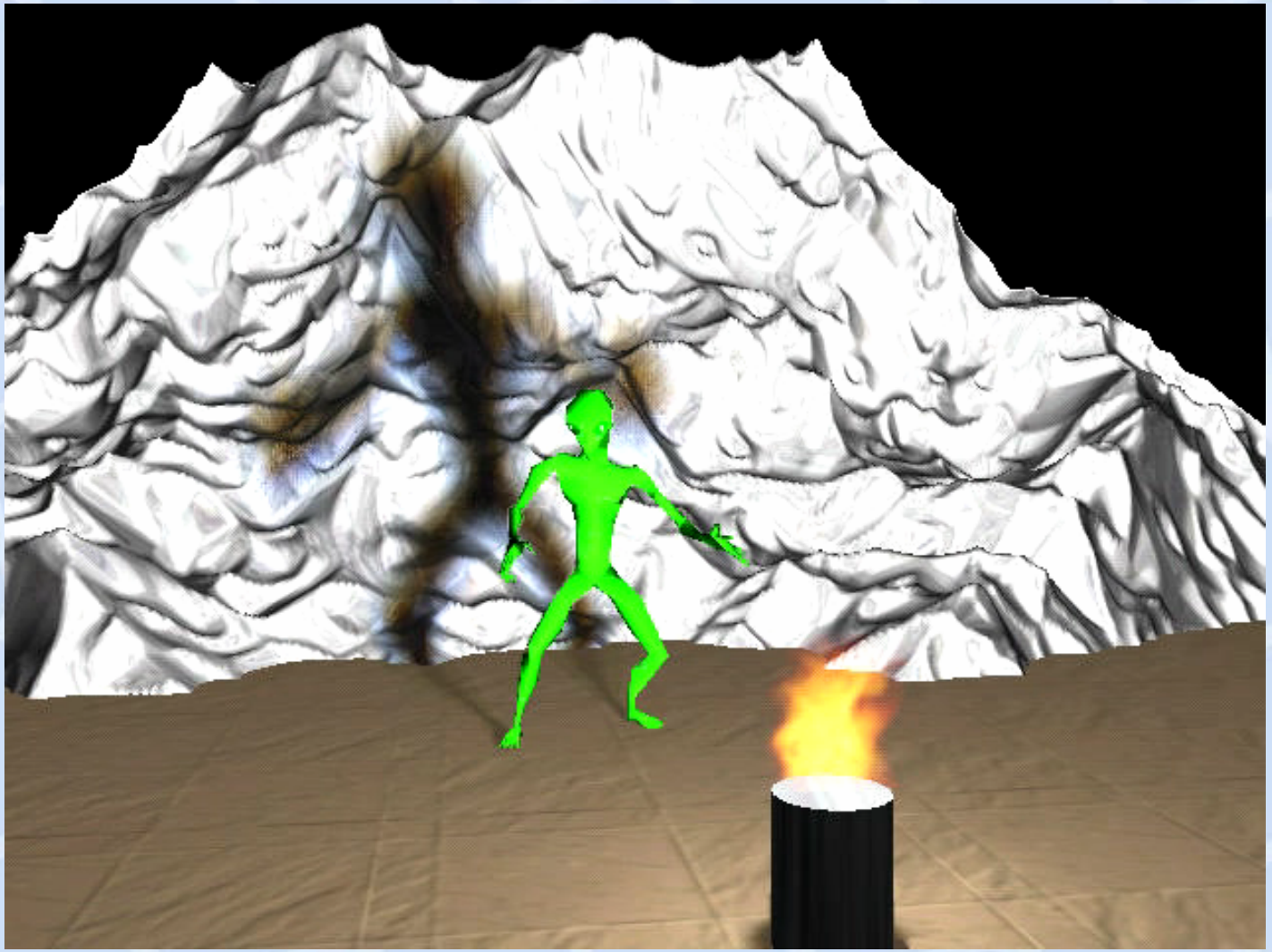
Texture of 16 area lights



Texture of two colors

Fire Demo

Fire Demo

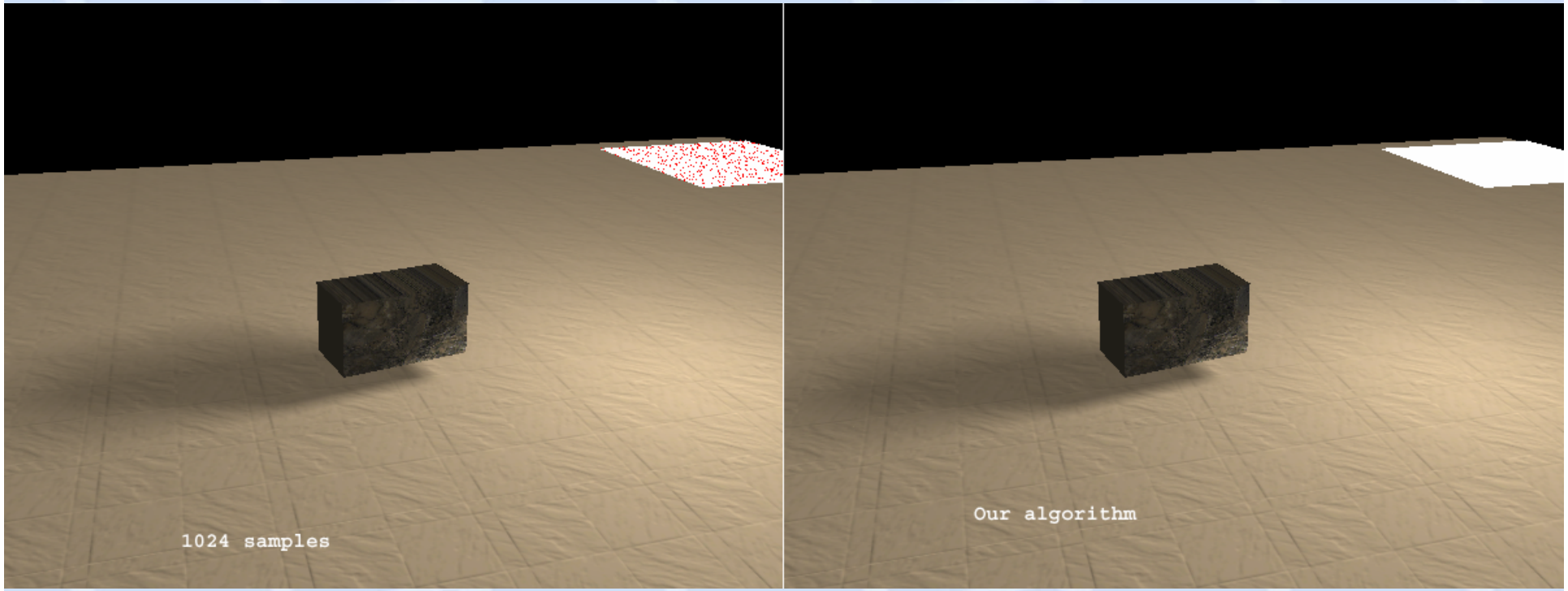


Comparisons

Reference image

Our algorithm

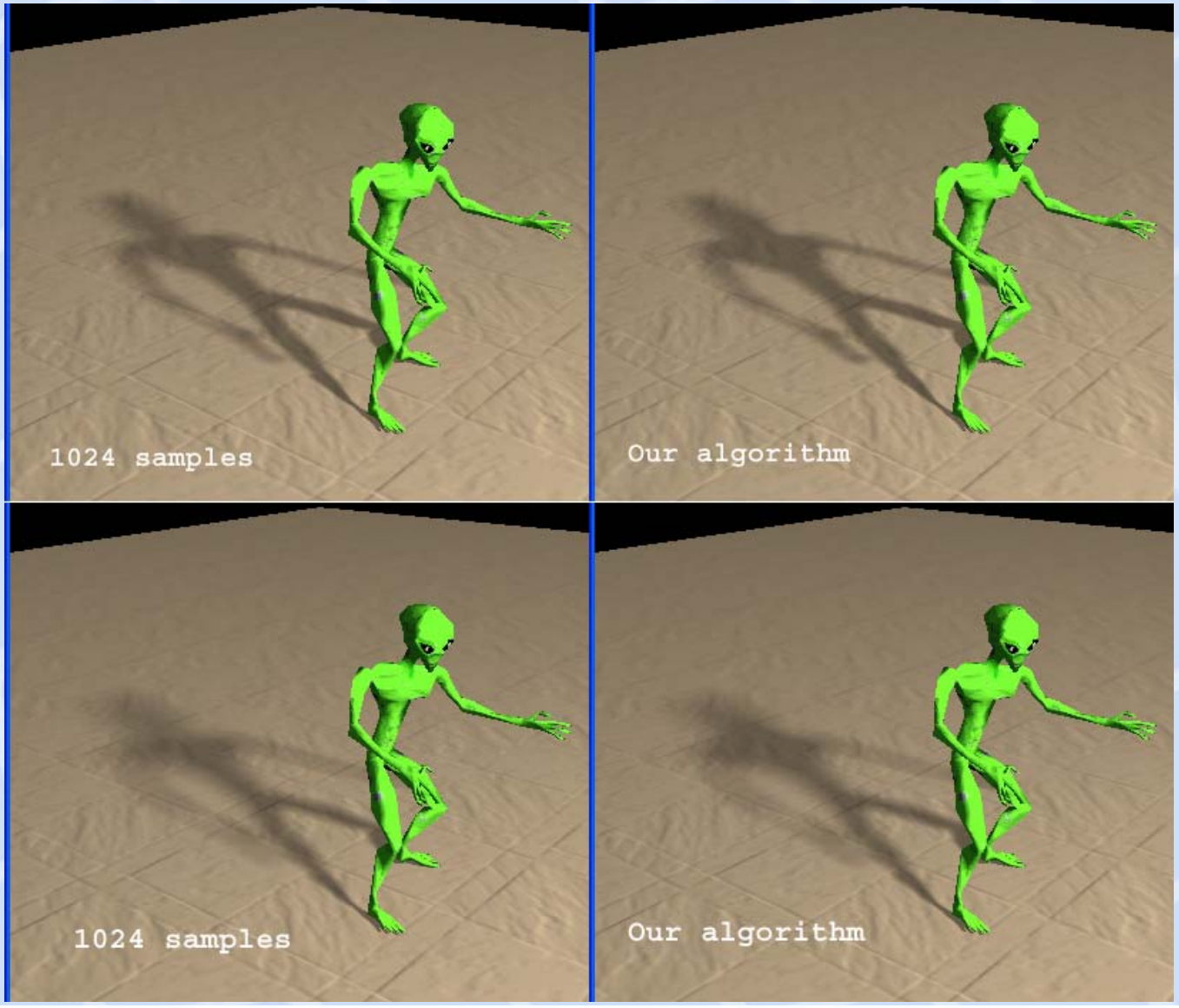
Comparisons



Reference image

Our algorithm

Comparisons



Small light source

Large light source

Comparisons

512 point lights

Our algorithm

The Last Demo