Department of Computer Graphics and Interaction
Faculty of Electrical Engineering
Czech Technical University, Prague

Real-time Shadows

XP39VR – Virtuální realita

Miroslav Mikšík

miksimir@fel.cvut.cz

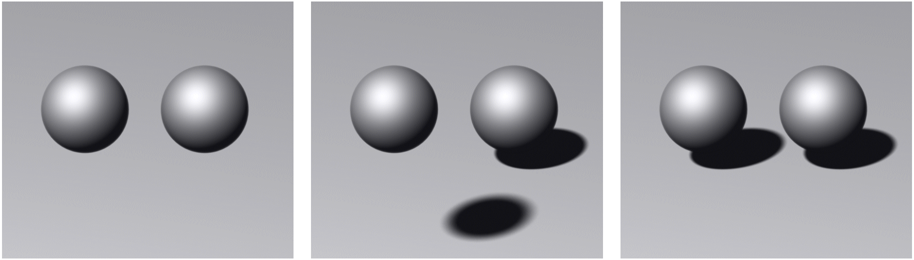
September 1, 2009

Outline

Visual Importance of Shadows

Shadows help to understand:

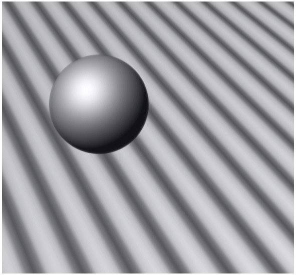
- relative position of objects

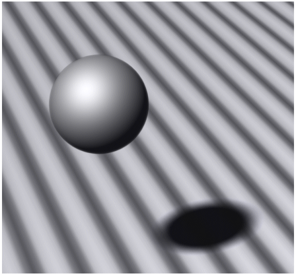


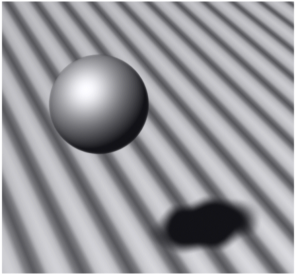
Visual Importance of Shadows

Shadows help to understand:

- relative position of objects
- receiver's geometry







Visual Importance of Shadows

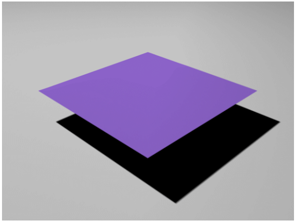
Shadows help to understand:

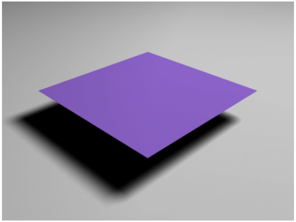
- relative position of objects
- receiver's geometry
- geometry of hidden occluders

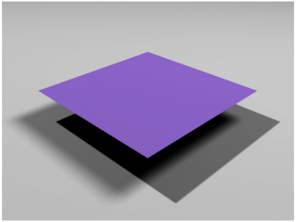
Visual Importance of Shadows

Shadows help to understand:

- relative position of objects
- receiver's geometry
- geometry of hidden occluders
- number and properties of light sources. . .



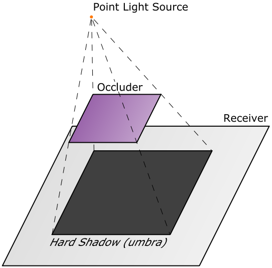


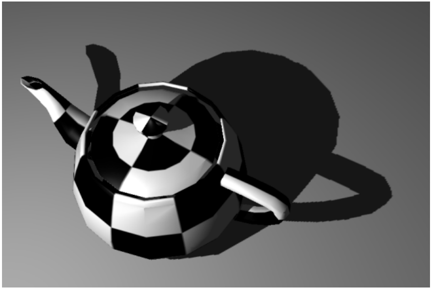


Hard Shadows vs. Soft Shadows

Hard Shadows:

- Produced by point light source.
- Point–Point visibility problem.

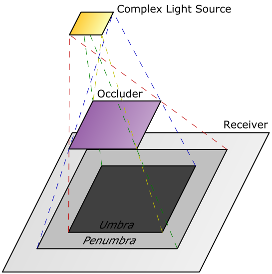




Hard Shadows vs. Soft Shadows

Soft Shadows:

- Produced by surface/volume light source.
- Much realistic.
- Point–Surface/Volume visibility problem:
  - **Analytical solution** – (almost) impossible
  - **Point sampling** – too slow for real-time
  - **Visible percentage estimation** – light must have uniform intensity



Classification of Shadow Algorithms

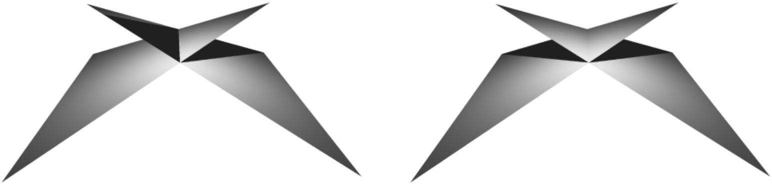
Common properties of a shadow algorithm:

- In which domain does it work?

Classification of Shadow Algorithms

Common properties of a shadow algorithm:

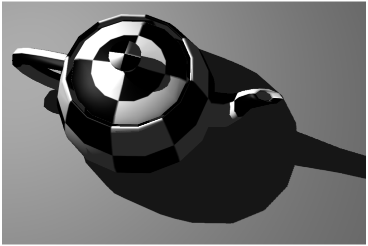
- In which domain does it work?
- Must be objects divided into occluders and receivers?

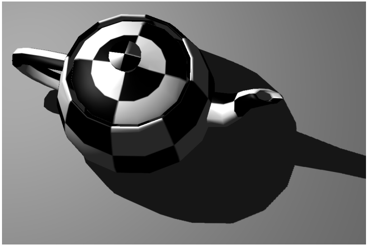


Classification of Shadow Algorithms

Common properties of a shadow algorithm:

- In which domain does it work?
- Must be objects divided into occluders and receivers?
- Does it support self-shadowing?

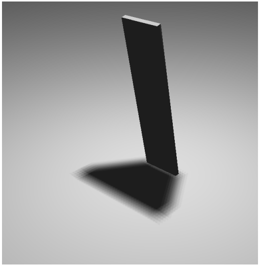




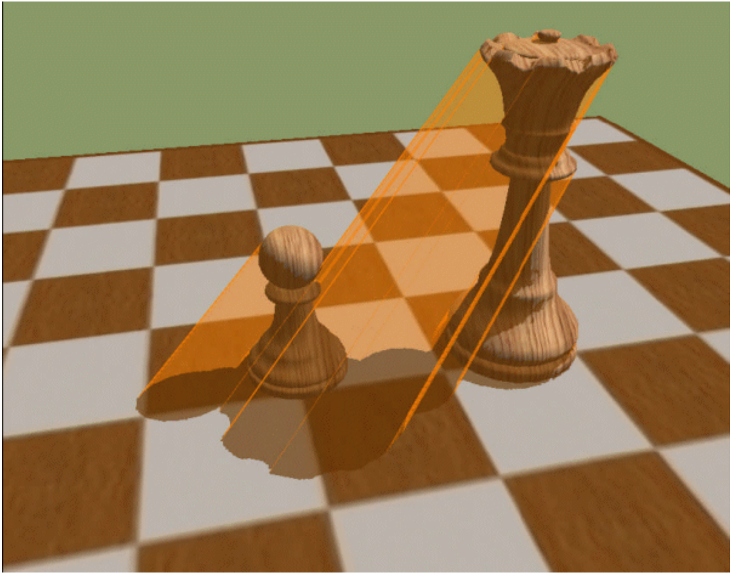
Classification of Shadow Algorithms

Common properties of a shadow algorithm:

- In which domain does it work?
- Must be objects divided into occluders and receivers?
- Does it support self-shadowing?
- Soft shadow fidelity:
    - fake soft shadows
    - realistic soft shadows
    - correct soft shadows

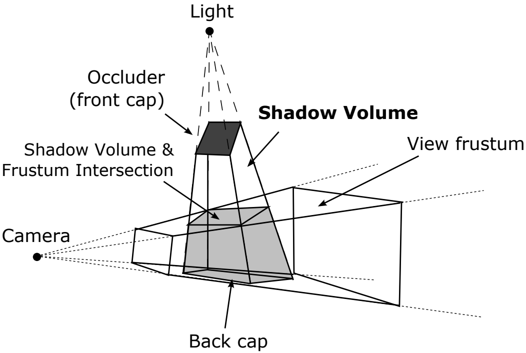


Shadow Volumes



Shadow Volumes [Cro77]

Shadow volume is the boundary between litted and shadowed space in scene:



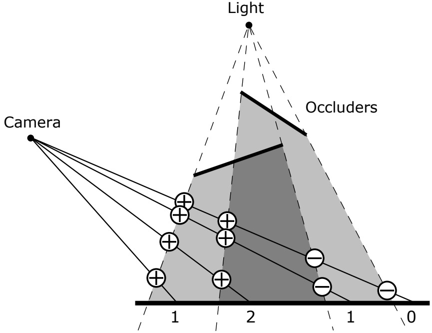
Idea:

- Determine shadow volumes.
- Render scene twice:
  - Fully lit for those fragments that are outside shadow volume
  - Fully dimmed for those fragments that are inside shadow volume

Z-pass Algorithm

Implemented using stencil buffer:

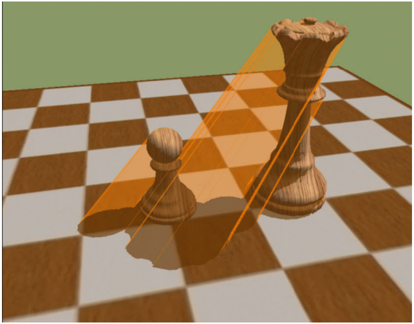
- Clear stencil & Z-buffer.
- Render scene to set up Z-buffer.
- Render shadow volume:
  - For front-facing faces increment stencil buffer if Z-test passes
  - For back-facing faces decrement stencil buffer if Z-test passes



Z-pass Algorithm

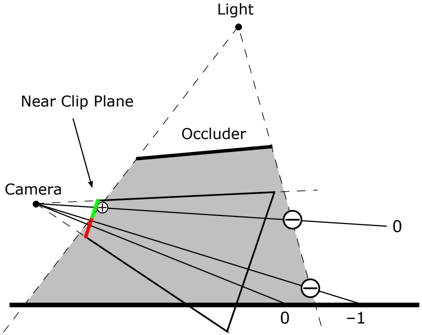
Shadow volume determination:

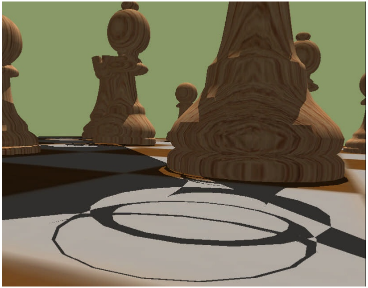
- Find countour of the object when looking from the light source.
- Extrude countour edges to infinity to create shadow volume faces.



Z-pass Algorithm

Algorithm fails when camera is inside shadow volume.

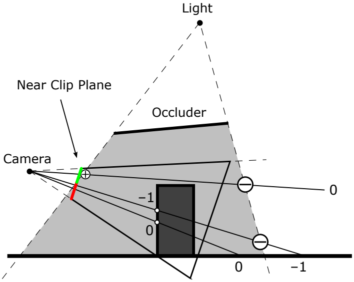




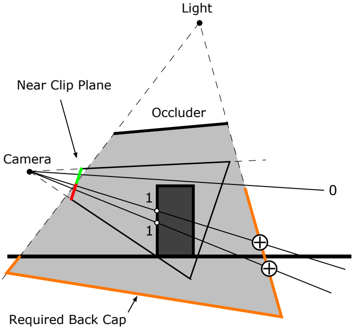
Z-fail Algorithm

Solution – Carmack's reverse (Z-fail algorithm):

- for back-facing volume's polygons – increment stencil buffer when depth test failed
- for front-facing volume's polygons – decrement stencil buffer when depth test failed

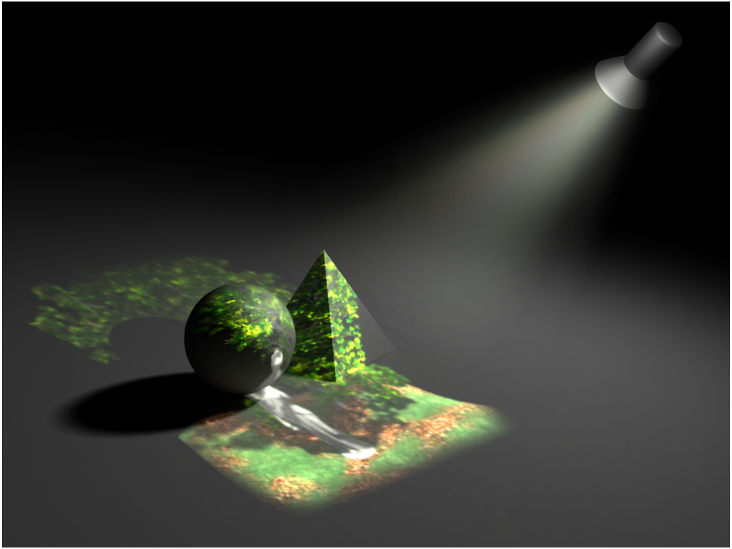


Z-pass



Z-fail

Projective Texture Mapping



Projective Texture Mapping [Eve01]

Create a camera for the light:

- same position, orientation, field-of-view, ...

For each point in the scene:

1. Transform its world coordinates to light's camera clip coordinates.
2. Adjust the coordinates from camera's clip coordinates range $\langle -1, 1 \rangle$ to texture coordinates range $\langle 0, 1 \rangle$:

$$\begin{bmatrix} s \\ t \\ r \\ q \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{P}_L \mathbf{V}_L \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

3. $(s/q, t/q)$ are perspective correct texture coordinates.

Projective Texture Mapping (cont'd)

Regular (perspective) mapping:

- Linear interpolation of (s, t) coordinates over primitive $=$
  $=$ linear interpolation of $(s/w, t/w)$ over pixels.

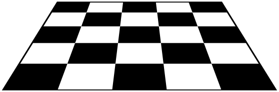
Projective mapping:

- Interpolate $(s, t, 1)$ over primitive; i.e. $(s/w, t/w, 1/w)$ over pixels.
- In fragment program use as $\left(\frac{s/w}{1/w}, \frac{t/w}{1/w}\right)$.
- Similar to homogenous clip coordinates:

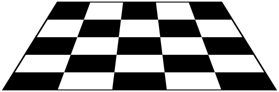
$$(x, y, z, w) \rightarrow (x/w, y/w, z/w)$$

- We use *homogenous texturing coordinates*:

$$(s, t, p, q) \rightarrow (s/q, t/q, p/q)$$



Perspective Mapping



Projective Mapping

Projective Shadows

Simple shadow algorithm:

1. Separate objects into receivers and occluders.
2. Draw occluders from light's view with black color on white background.
3. Draw receivers from camera's view. For each visible fragment find its projective texture coordinates and lookup in texture (black = in shadow, white = lit).
4. Draw occluders fully lit.

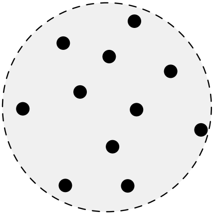
Shadow Texture

Result

Poisson Disk Filtering [Mit04]

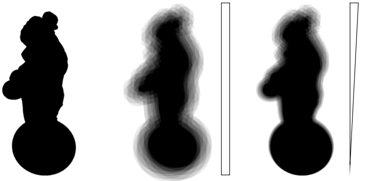
The texture can be filtered using e.g. Poisson Disk filter:

- Kernel with random samples but well distributed



Poisson Disk Kernel

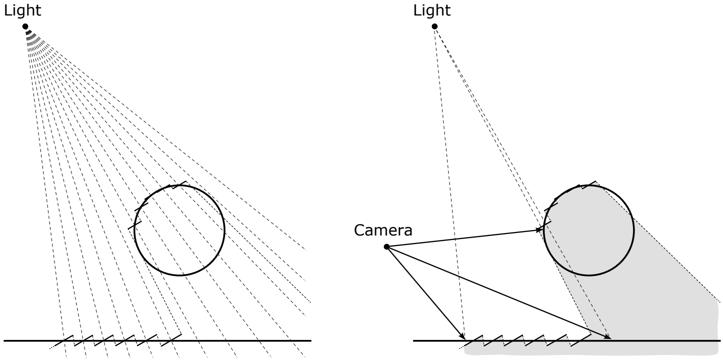
Radius modulation:



No Filtering Fixed Kernel Size Variable Kernel Size

Shadow Maps [Wil78]

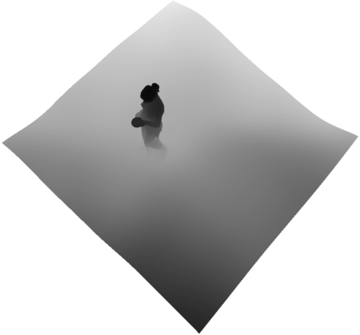
Component p/q the homogenous texture coordinates represents normalized depth in the light's frustum.



Shadow Maps

Shadow map algorithm:

1. Draw scene from light's view and save Z-buffer to the texture (shadow map).
2. Draw scene from camera's view. For each fragment compare its depth p/q with depth in the texture at coordinates $(s/q, t/q)$.



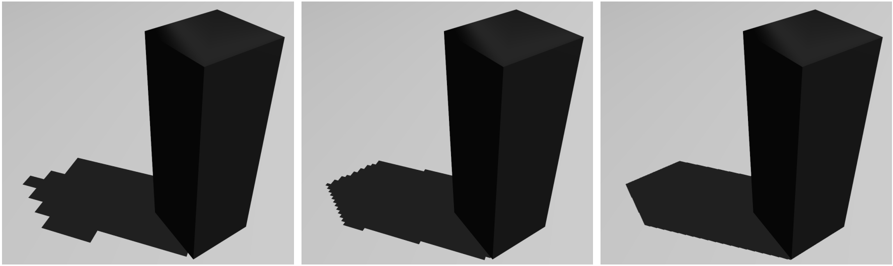
Shadow Map

Result

Shadow Maps Problems

Shadow maps problems:

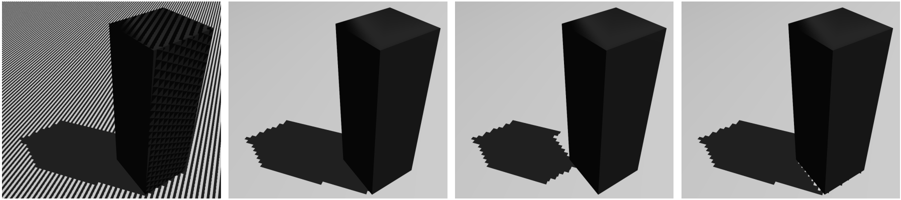
- texture resolution



Shadow Maps Problems

Shadow maps problems:

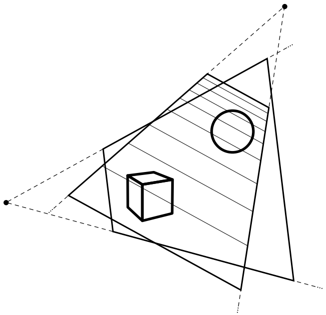
- texture resolution
- self-shadow alias



Shadow Maps Problems

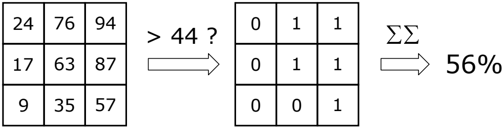
Shadow maps problems:

- texture resolution
- self-shadow alias
- non-uniform depth distribution

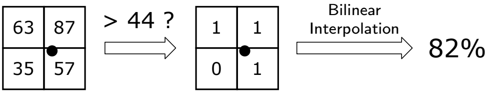


Percentage Closer Filtering [RSC87]

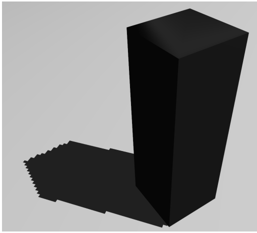
Shadow maps cannot be directly filtered.
Filter depth comparision instead.
Percentage Closer Filtering:

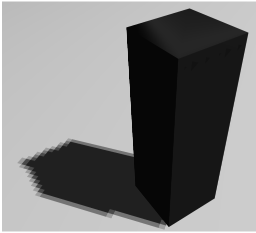


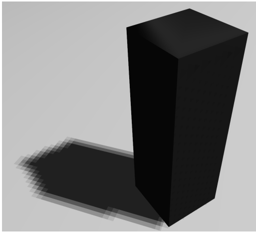
Good hardware support (PCF bilinear interpolation):

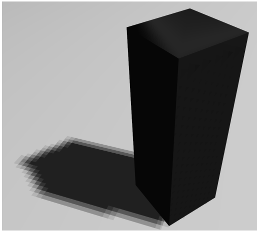


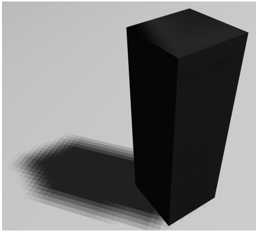
Percentage Closer Filtering (cont'd)

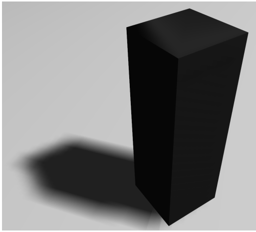

No PCF


2×2 PCF


3×3 PCF


4×4 PCF


5×5 PCF


5×5 PCF with GPU interpolation

Variance Shadow Maps [DL06]

Shadow map values are treated as a random variable:

- light occlusion is represented as a probability $P[d_{fragment} > d_{map}]$
- use Chebyshev's inequality for occluded fragments to estimate the probability of light visibility:

$$P[d_{fragment} \leq d_{map}] \quad \leq \quad p_{\max}(d_{fragment}) = \frac{\sigma^2}{\sigma^2 + (d_{fragment} - \mu)^2}$$

- where: $\mu = E(d_{map}) = \mathbf{M_1}$ – mean value
  $\sigma^2 = E({d_{map}}^2) - E(d_{map})^2 = \mathbf{M_2} - \mathbf{M_1^2}$ – variance

Use two shadow maps:

- for M_1 – regular shadow map
- for M_2 – shadow map with depth squared

Now we can blur them!

Variance Shadow Maps (cont'd)

Shadow maps can be filtered to:

- produce soft shadows
- naturally suppress self-shadow alias

Produces "light bleeding" artifacts – in areas with high variance (probability approximation).

Variance Shadow Map
with Poisson Disk Filtering

Classical Shadow Map
with the Same Parameters

Light Bleeding Problem

Conclusion

Shadow Volumes	Shadow Maps
Advantages	
pixel-precise shadows	good scalability
accelerated by GPU	fully accelerated on GPU
view-independent	fast & simple
robust	hard & soft shadows
omnidirectional lights	
Disadvantages	
contour detection	artifacts, aliasing
need of connectivity information	memory expensive
only hard shadows	view-dependent
bad scalability	not robust
fill-rate intensive	directional & spot lights

Comparision

Shadow Volumes

Projectively Mapped Texture

Variance Shadow Maps

References

[Cro77] Franklin C. Crow.
Shadow algorithms for computer graphics.
SIGGRAPH Comput. Graph., 11(2):242–248, 1977.

[DL06] William Donnelly and Andrew Lauritzen.
Variance shadow maps.
In *Proceedings of the Symposium On Interactive 3D Graphics and Games 2006*, pages 161–166. ACM Press, 2006.

[Eve01] Cass Everitt.
Projective Texture Mapping.
Technical report, NVIDIA Corp., 2001.

[HLHS03] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion.
A Survey of Real-Time Soft Shadows Algorithms.
In *Eurographics – State of The Art Report*. Eurographics Association, 2003.

[Mit04] Jason L. Mitchell.
Poisson Shadow Blur.
In *ShaderX*^{3}, pages 403–409. Charles River Media, 2004.

[RSC87] William T. Reeves, David H. Salesin, and Robert L. Cook.
Rendering Antialiased Shadows with Depth Maps.
In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 283–291. ACM Press, 1987.

[Wil78] Lance Williams.
Casting Curved Shadows on Curved Surfaces.
In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274. ACM Press, 1978.

The End

. . . any questions?