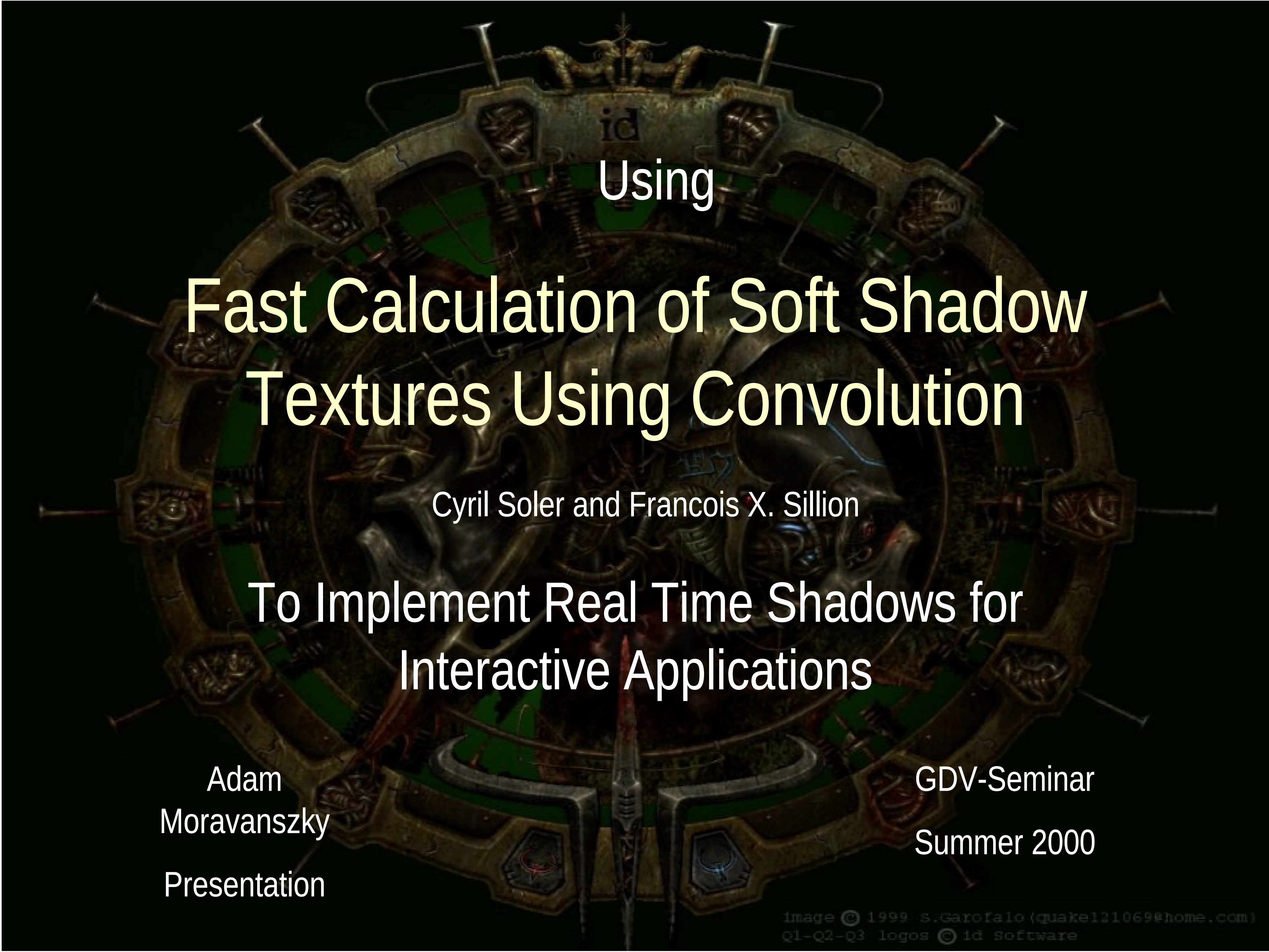


Using

Fast Calculation of Soft Shadow Textures Using Convolution

Cyril Soler and Francois X. Sillion

To Implement Real Time Shadows for Interactive Applications

Adam Moravanszky

Presentation

GDV-Seminar

Summer 2000

Overview

- The Problem and Terminology
- Shadows in Computer Graphics Theory
- Shadows in Practice
- Shadows in this Paper
- Demo of Implementation

What are Shadows?

Definitions

- Light source
- Blocker
- Receiver
- Umbra - completely occluded (all illumination from other sources)
- Penumbra - partially occluded

Shadows in Computer Graphics Theory

•No Shadows

•Raytraced

•Shadow Buffers

•Shadow Volumes

•Radiosity

•Shadow Textures

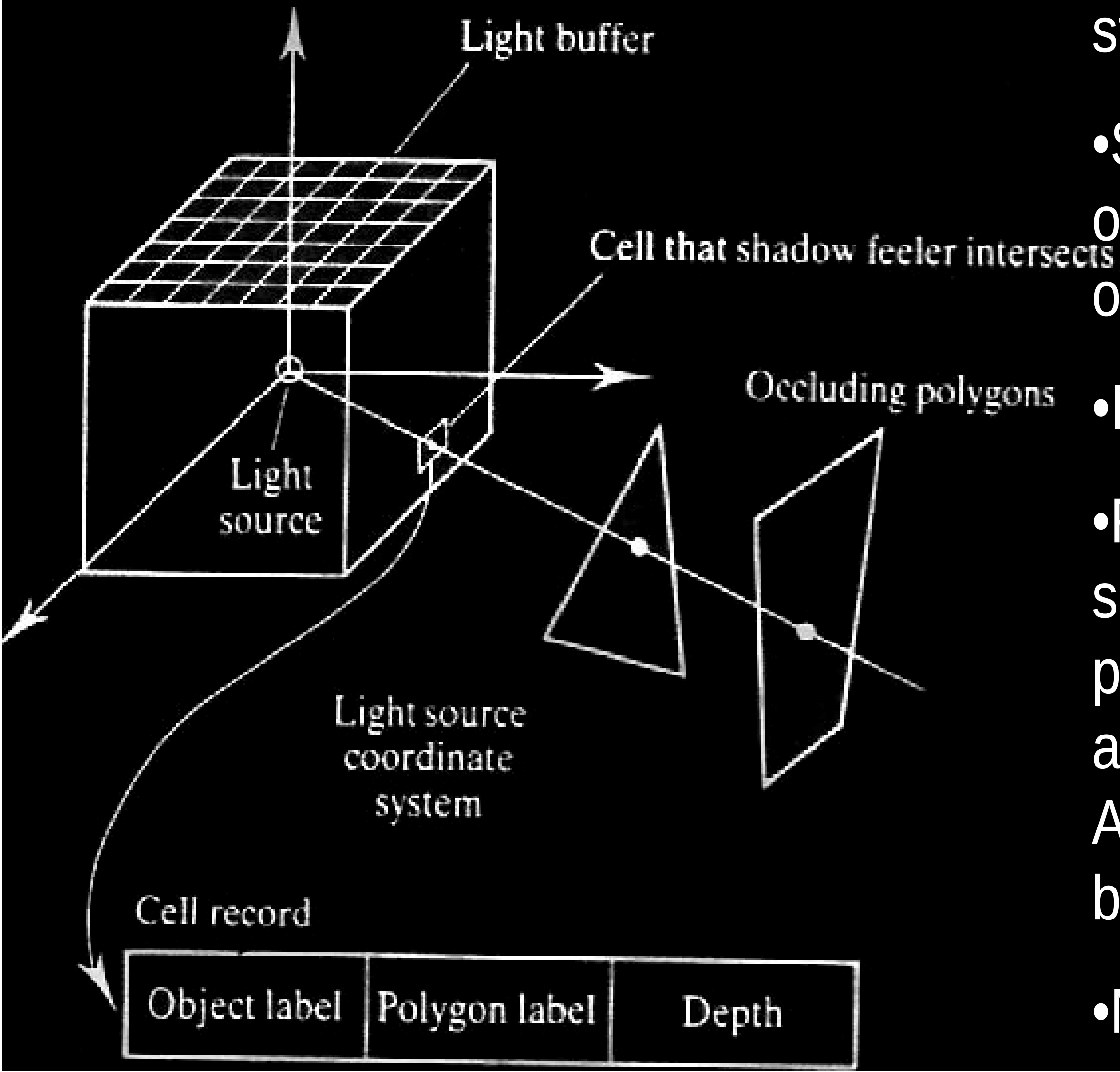
These are all raster/image/texture based algos like that of current paper, but still represent a broad overview. The total number of shadow algos available is much greater!

Shadowing is a form of hidden surface determination problem, so shadow algos have much in common with HSD algos for cameras.

Shadows in Computer Graphics Theory: No Shadows

- Vertex Lighting (OpenGL)

Shadows in Computer Graphics Theory: Raytraced



- Trace ray from eye through pixel, stabbing object.
- See if light is visible from ray-object intersection point. (Other objects may be in the way)
- Improvement: Light Buffers
- For each light's each discretized solid angle, record all the polygons visible in this direction, and sort according to depth. Above visibility computation becomes table lookup.
- NB: Incompatible with grx. hw.

Shadows in Computer Graphics Theory: Shadow Buffers

•Two z buffers: one from viewer POV, other from lightsource.

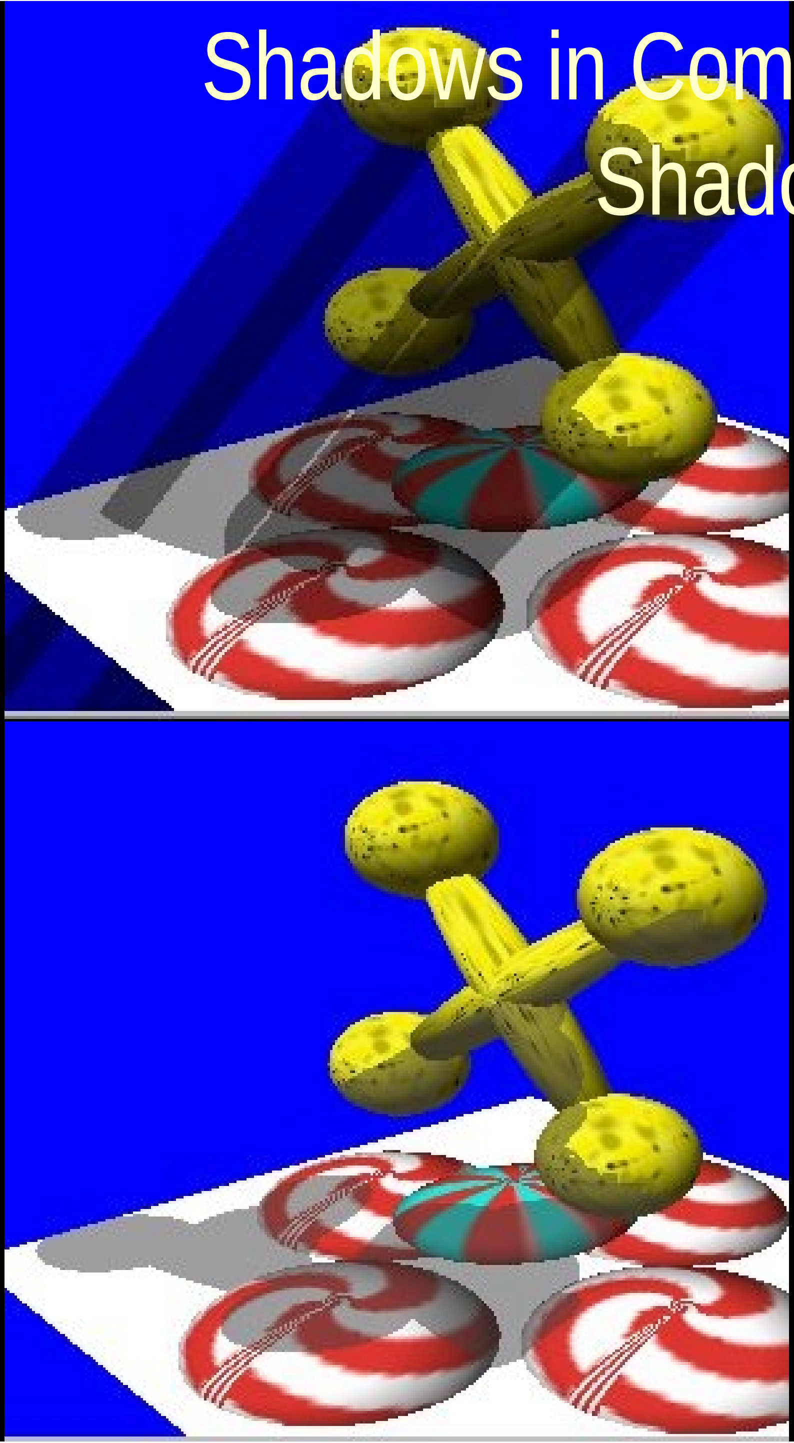
1) Create z buffer from light POV.

2) when a fragment is determined to be visible via viewer z-buffer, it‘s coordinates are transformed into the light‘s coord sys.

3) These coordinates are looked up in the light‘s z buffer. If the value stored there is closer than the actual distance, the fragment is in shadow, else it is illuminated.

•NB: Suitable for eventual hardware implementation, but this level of the graphics pipeline is no longer in the application programmer‘s hands.

Shadows in Computer Graphics Theory: Shadow Volumes



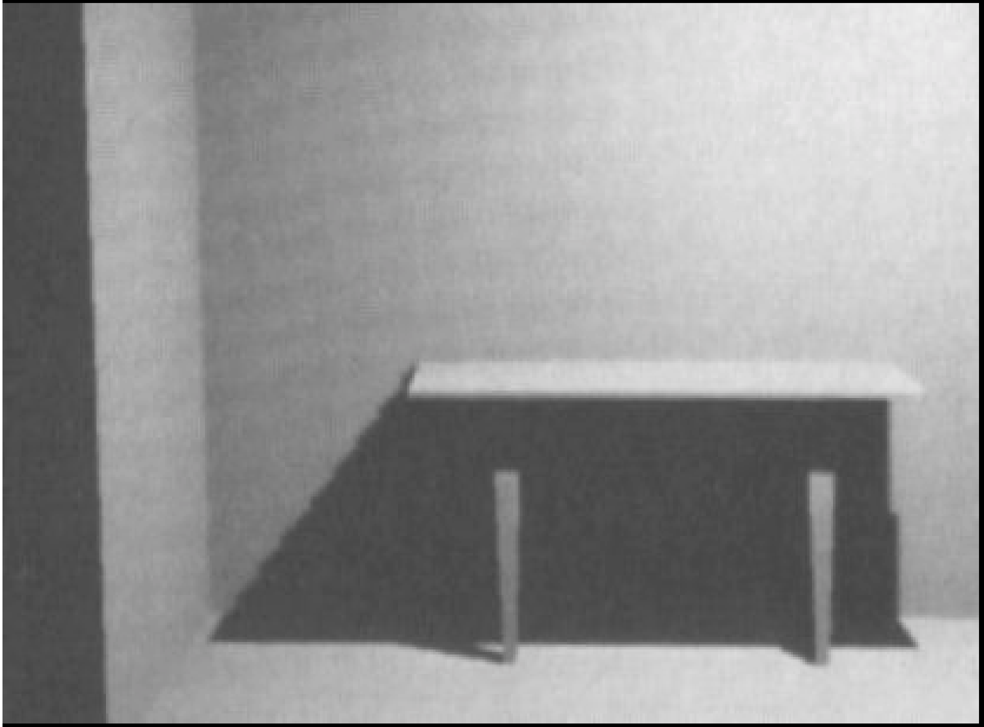
•Create a shadow volume for each blocker+light source.

•Shadow volume is space occluded by silouette of blocker, delimited by ‚shadow polygons‘.

•A point is in shadow if the line segment from the viewer (assumed to be out of shadow) to the point intersects an odd number of shadow polygons. (like 2D in polygon test)

•Potentially practical for real time apps. More later.

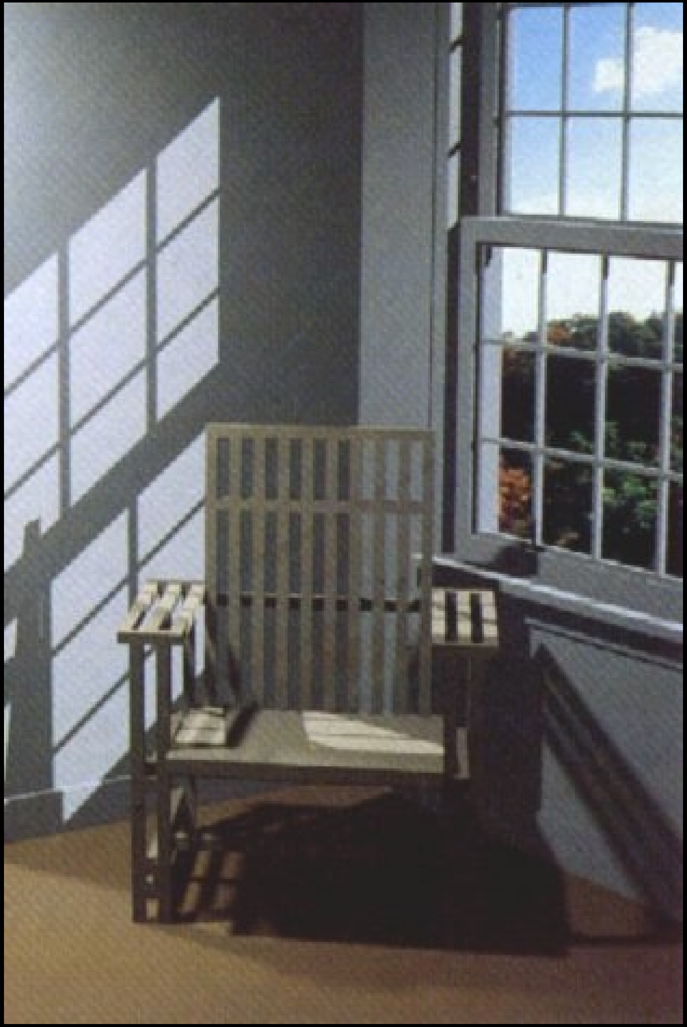
Shadows in Computer Graphics Theory: Radiosity



- •Adaptive subdivision via quadtree produces stair-step effect.

- •View independent global illumination algorithm. (Can be done off-line!)
- •Lighting model evaluated recursively at regular intervals on surfaces in scene.
- •Color determined by light absorbed from all other surfaces.
- •Implicit soft shadows, crispness depends on sampling density.
- •Only diffuse lighting taken into account (specular is view dependent) so only some surface types can be simulated accurately unless combined with other techniques.
- •Interesting for real time simulation, more later.

Shadows in Computer Graphics Theory: Radiosity with Discontinuity Meshing



•Choose radiosity evaluation mesh according to lighting conditions rather than recursive arbitrary orthogonal subdivision scheme as before.

Shadows in Computer Graphics Theory: Shadow Textures

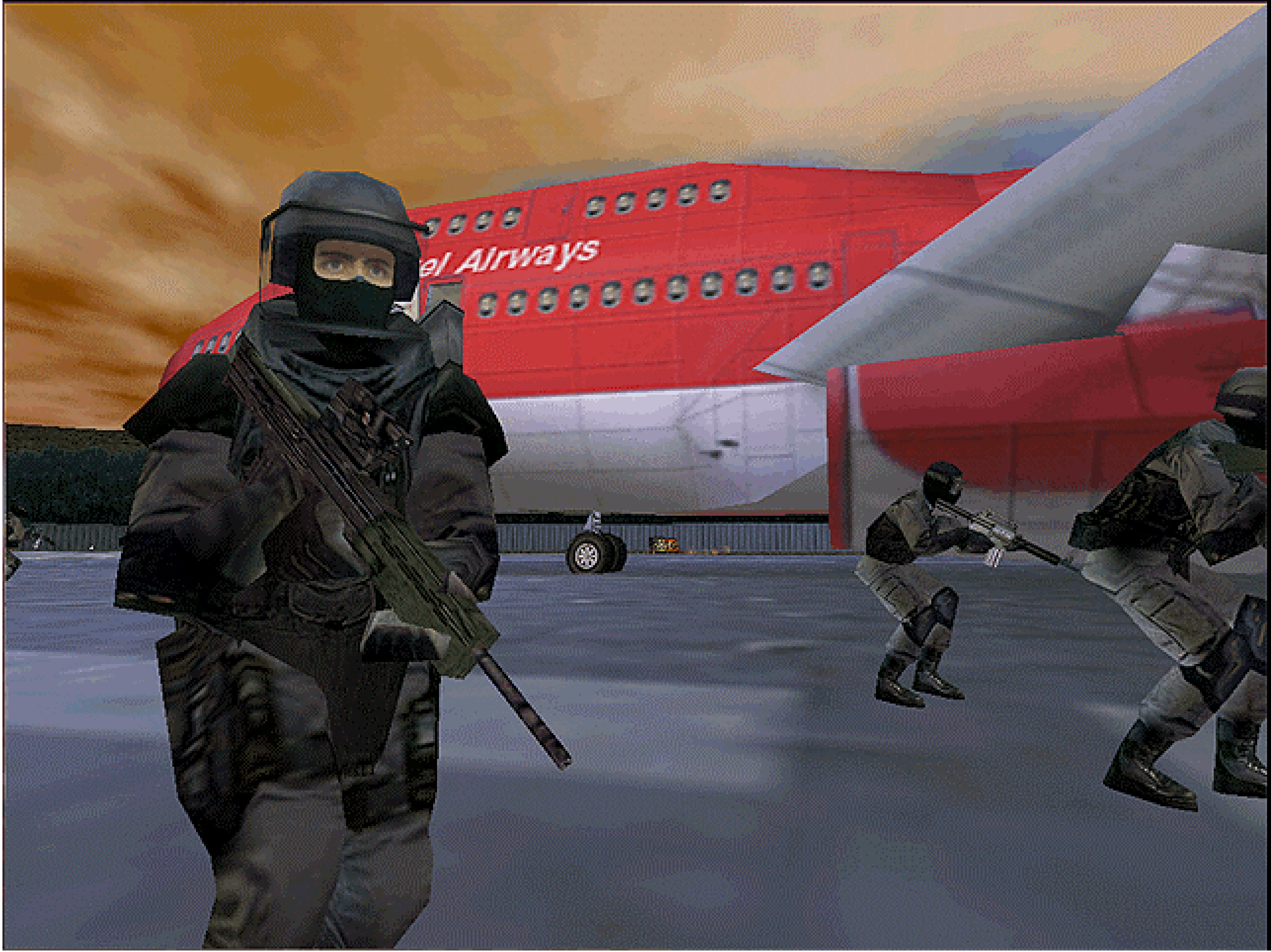
- •First used for real time display of results of Radiosity algorithm / shadows.
- •Texture mapping is often implemented in hardware.
- •Shadow texture maps are,like Radiosity, view independent, and don‘t have to be recomputed if only the POV changes.
- •Soft shadows can be supported by sampling the area light a few times and superimposing the texture images. (Paul Heckbert, 1995)
- •Foundation of the current paper.

Shadows in Practice

- No Shadows
- Circular Splotches
- Projected Geometry
- Stencil Shadows
- Lightmaps

Shadows in Practice

No Shadows



Pros:

•Guaranteed Faster than any other algorithm.

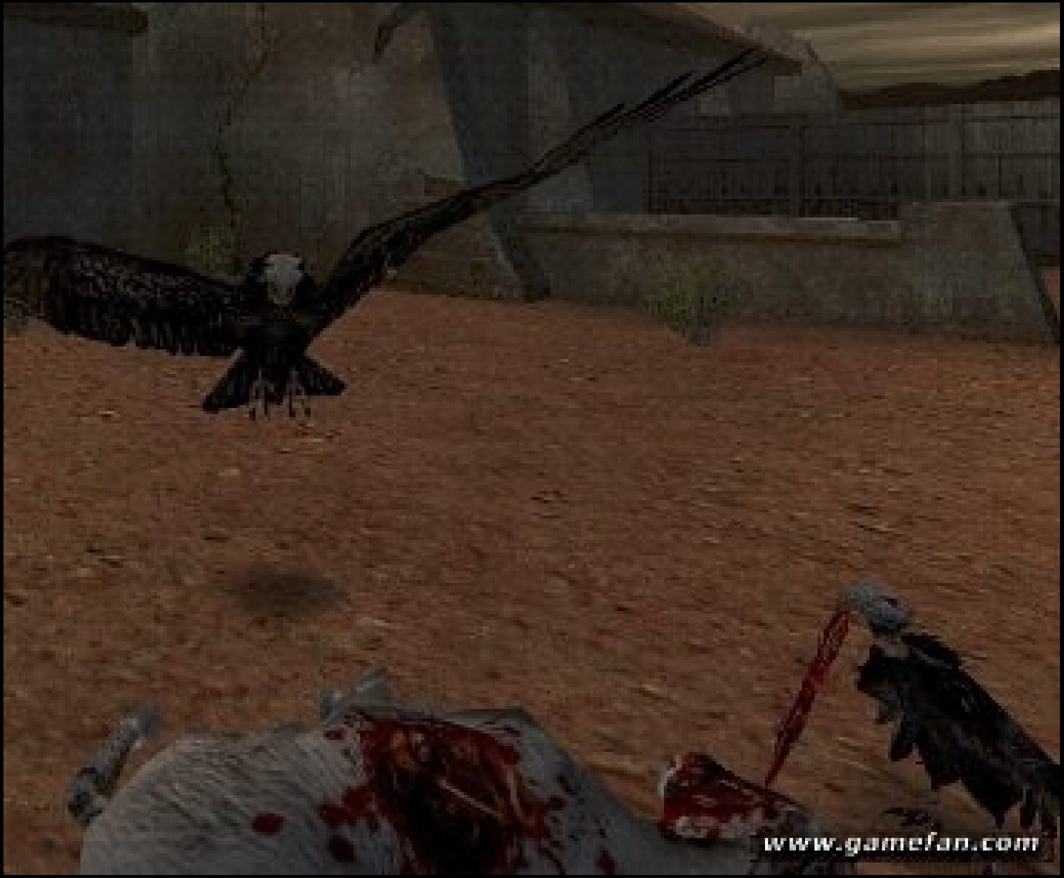
•May look better than hard shadows

Cons:

•Difficult to judge depth of objects. Hovering effect.

Rogue Spear

Shadows in Practice
Circular Splotches



Pros:

•Fast

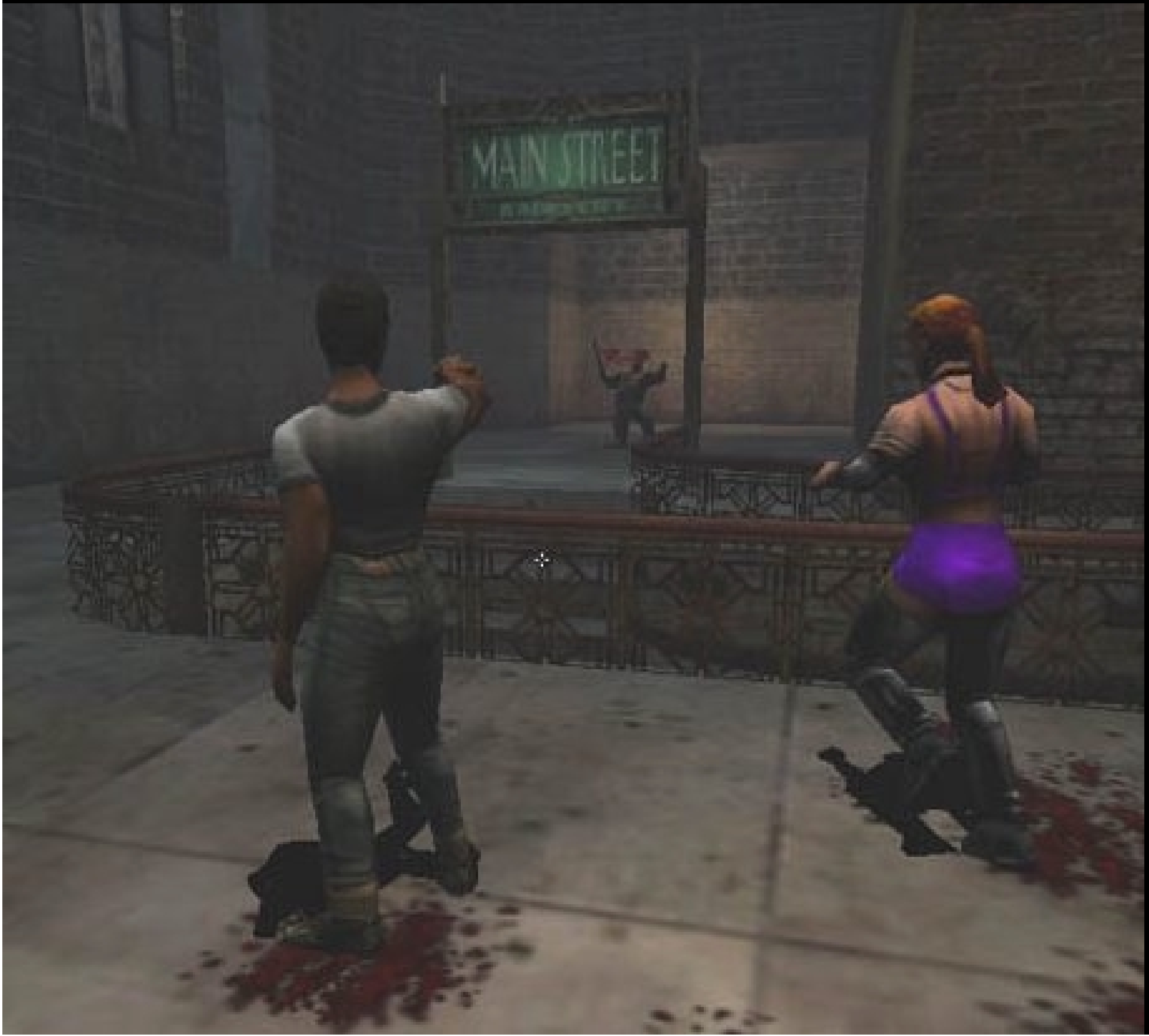
•Gives a minimal sense of depth.

Cons:

•Not very realistic at all.

Duke Nukem Forever

Shadows in Practice
Projected Geometry



Most popular technique today.

Pros:

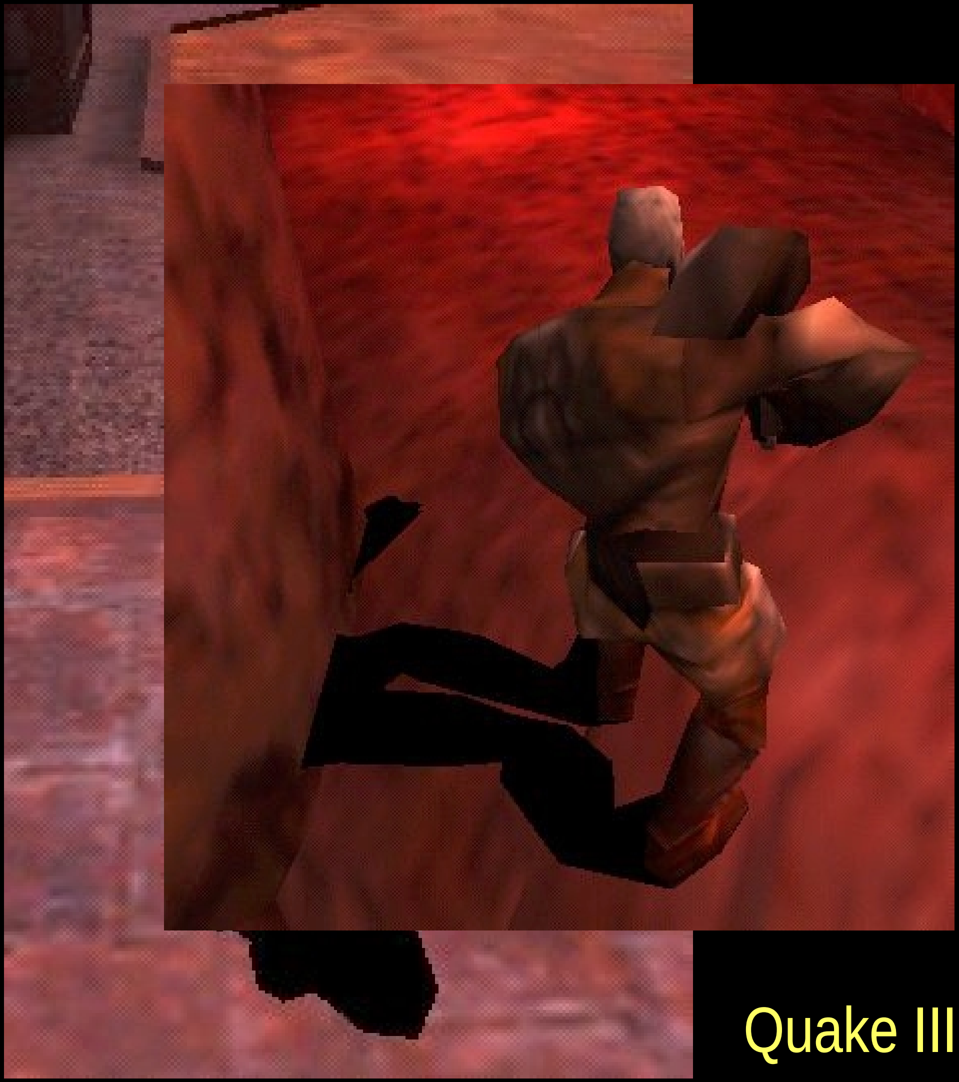
•Shadow of approximately correct shape

•Fast

Kingpin

Shadows in Practice

Projected Geometry [BLIN88]



Quake III Arena

Usually paralell projection so shadow doesn‘t grow to minimize ugly effects.

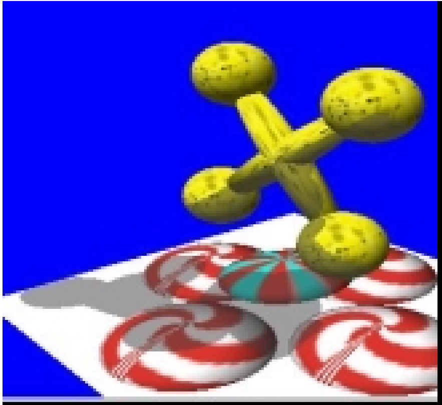
Cons:

•Hard shadow

•Receiver is assumed to be a single plane

•Usually not clipped at ledges

•Performance is function of blocker mesh complexity - usually low LOD mesh used.



Shadows in Practice
Stencil Buffers

Much talk about it, but no succesful commercial implementation found.

Pros:

•Perfect shape even on arbitrary geometry.

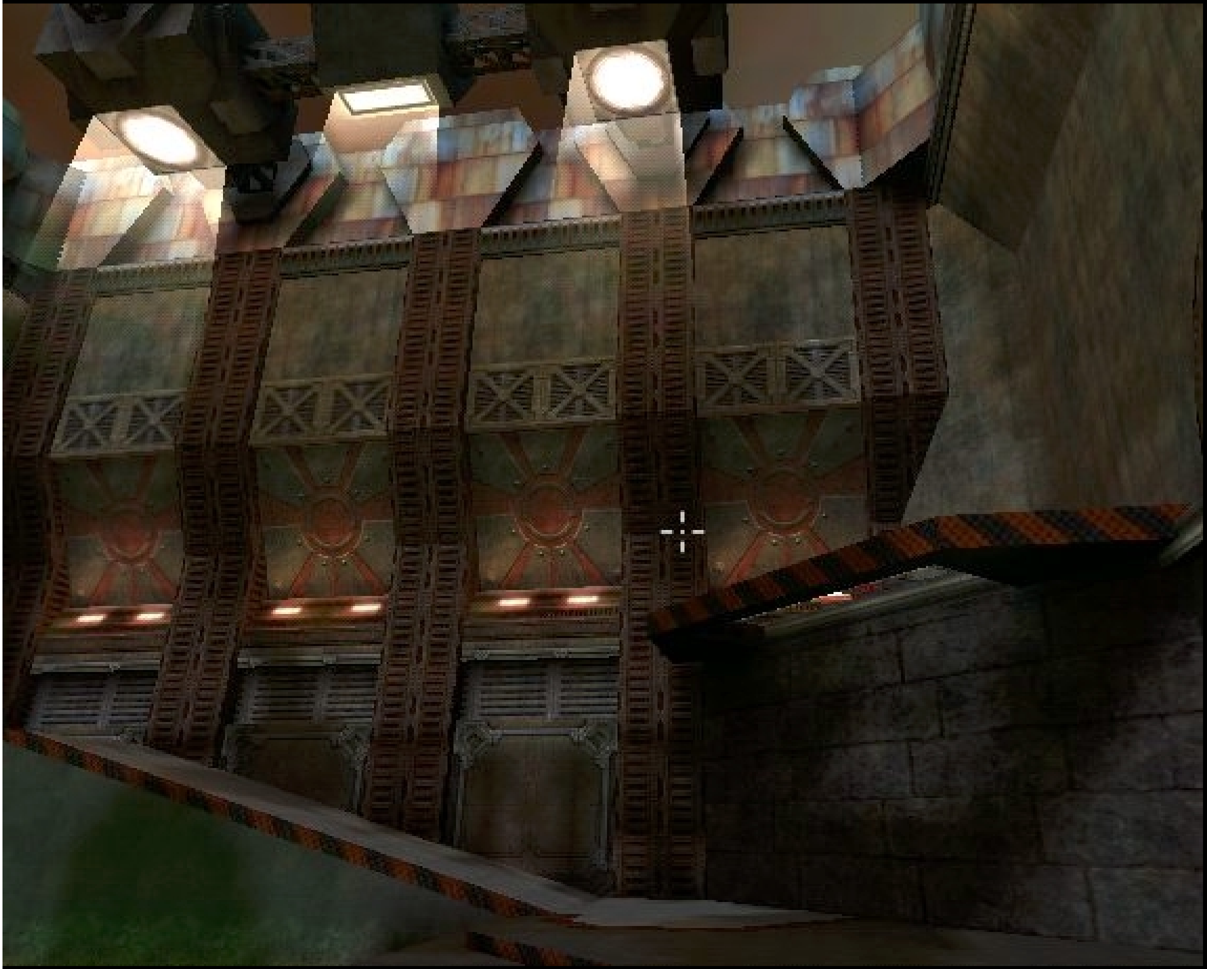
Cons:

•Hard shadow

•Lacking stencil buffer support

•Needs second pass -> Image buffer res. dependant

Shadows in Practice
Lightmap textures



Pros:

•Computed off-line - w. radiosity (Free!)

•Soft shadow

Cons:

•Storage intensive

•Usu. low resolution

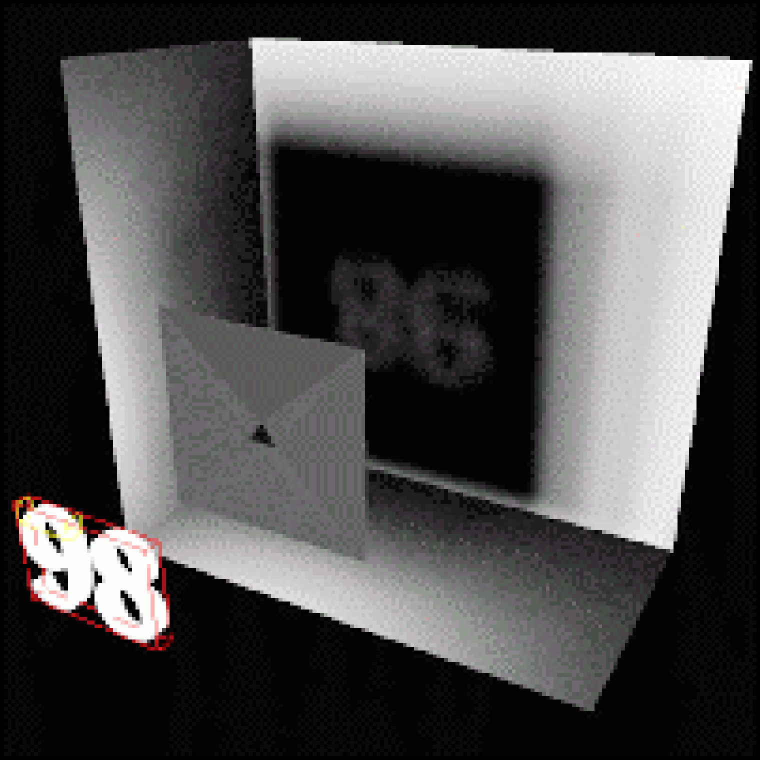
•Only for static geometry

Quake III Arena
(uniform sampling mesh)

Shadows in Practice

PowerPoint

Soft Shadow Textures Using Convolution

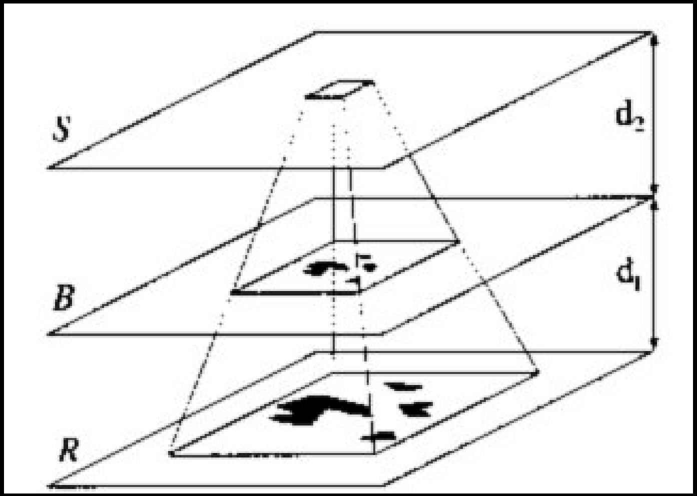


Features:

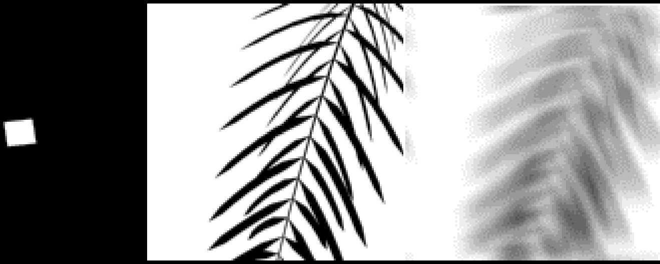
- Compatible with any graphics pipeline which supports textures.
- Takes volume lights into account.
- Soft shadows.
- Object based.
- Produces approximately correct results.
- Based on projected textures tech.

Simple Paralell Configuration

- Basic problem: How to generate blurry shadow image which takes shape and distance of light source into account?
- First, a simple assumption of the configuration of the scene:



Simple paralell configuration



Math Foundations for Paralell Scheme

- Our math. foundation for correct shading is our definition of Irradiance H(x)- the total radiation arriving at a point y on a surface.

$H(y) = \int L_i(y, \theta,\phi)\, d\theta d\phi$

- integral of radiance coming in from all directions.

Separating Formula Into Two Pieces

For one light source with directionally invariant exitance E this becomes:

$H(y) = (E/\pi)\int \cos\theta \cos\theta' \, d^{-2} \, v(x,y) \, dx$

Approximation / Fudging as usual:

$H(y) \approx E\int \cos\theta \cos\theta' \, \pi^{-1} \, d^{-2} \, dx \qquad E \qquad \int v(x,y) \, dx$

$F(y) \qquad E \qquad V(y)$

Form factor Visible area of the source

$H(y) \approx F(y) \, E \, V(y)$

x - point on source, d - distance(x,y) θ θ' - ray angles on s&r.

Computing E, F and V

- E*F is unoccluded illumination term, can be computed with Radiosity, or direct illumination formulas (OpenGL).
- V can be computed as integral of projection of blocker on source as y moves on the receiver. It can be shown that this is a convolution operation on the source and blocker image. (next slide!)
- E can be taken inside the integral. Practically this means that we may modulate the source image by it‘s exitance function (textured light!) before doing the convolution.
- The product of F and (EV) can be executed as a (hardware) texture blending operation.

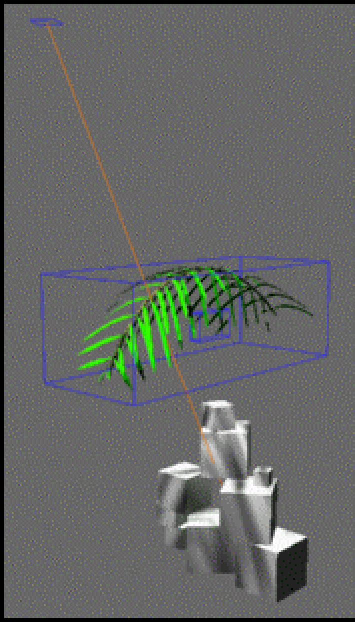
Proof that Convolution is Appropriate

- Intuitively:
  - A spherical light source will act as a regular lowpass filter and blur the blocker image uniformly - and produce even penumbras.
  - An cylindric flour. light‘s image will blur primarily along its length‘s direction, which is the type of shadow it produces.
- Math: see paper for derivation. Resulting formula:
  - $a := d_1/d_2$ ---> scale factor for textures.
  - binary f-s (textures): $S(x) = 1$ if x on source. $P(x) = 0$ if x on blocker
  - $V(y) = a^{-2} (S(-a^{-1}x) * P((1+a)^{-1} x)) (y)$
- Btw: (discrete) 2D convolution * is:

$$V[u][v] = \Sigma_j \Sigma_i S [u + i][v + j] \; P[i][j]$$

General Configuration

- Before we assumed that all geometry was paralell planar.
- Now we transform the actual arbitrary geometry to such an ideal configuration:
  - choose a projection direction D.
  - choose the three planes for source, blocker and receiver orthogonal to D.



Transforming into Paralell Configuration

- The Projection direction D is chosen to be a good representative of all rays from the source to the receiver --> come up with an approximate frustum using the bounding volume of the receiver, then take the direction of the frustum for D.
- The altitudes of the orthogonal planes on D are effectively the assumed distances between the ‚objects‘.
  - Choose them so the centers of the altitude ranges of the objects.
  - Can choose blocker plane altitude so that we end up with in avg. correctly sized penumbras based on penumbra error metric (later).

Rendering Images

- When two images have been rendered, convolve them either directly, or better, with FFTs by using the property:

  $f * g\,(y) = FT^{-1}(\,FT(f)\,FT(g)\,)$

- When using FFTs, we have to enlarge the frustra a little to protect from wrap around which comes from FT assuming that the images are periodic.

Problems and Sources of Error

Assuming that geometry can easily be separated into blocker and receiver may be a serious restriction.

- As entire blocker is projected into a single plane, no chance for gradual loss of shadow focus for long objects. All penumbra regions of an object will have same sharpness.
- Texture resolution is a variable.

Measuring the Error

We have math to back up virtual paralell scheme so that is OK.

- What is error of projecting geometry into the planes?
- Paper presents a way to estimate human perceived rather than actual error (compare lum. functions), looking at only rendered vs. correct penumbra sizes as this is most obvious error.
- Computation compares penumbra size computed using virtual plane altitudes vs. hypothetical vertex altitudes = error variable.

Soft Shadow Textures Using Convolution

Reducing the error:

- Source Subdivision
- Blocker Subdivision
  - combining shadow textures from different blocker clusters is not trivial.
- Receiver Subdivision
  - separate shadow textures computed for each piece.
  - boundary artifacts where textures meet.
- Shadow texture resolution
- Hierarchical refinement.

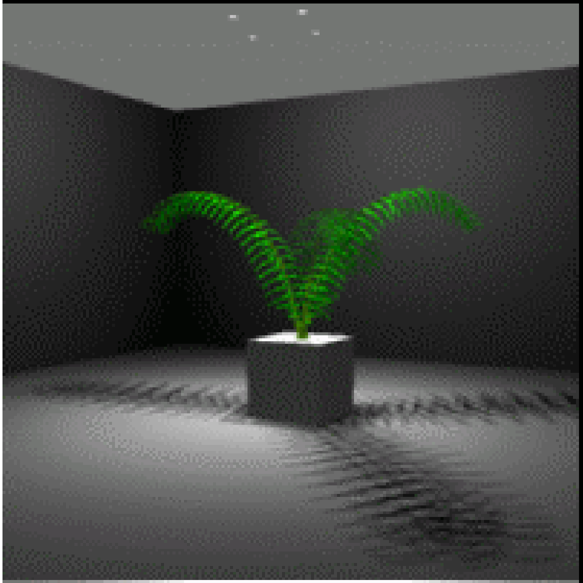
Soft Shadow Textures Using Convolution

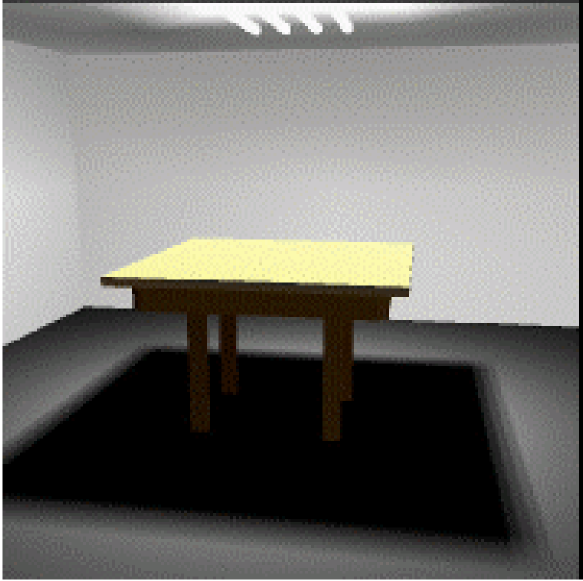
My opinion:

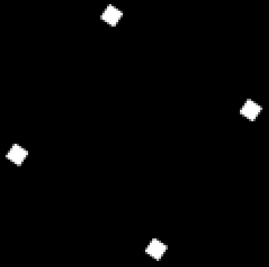
- Can be done very well in real time. FFT is surprisingly fast.
- Best for indoor character shadows where object based system is not a problem, and we expect nice penumbras.
- Small inaccuracies in penumbra size not noticed by casual observer. Best looking real time shadows yet.
- Good use of HW multitexture capability.
- Readback from rendertarget is not optimized. Rendering the textures eventually a bottle neck. This may change soon.
- Not too difficult to implement.

Bottom line: Best algo for this purpose at this time!

Soft Shadow Textures Using Convolution



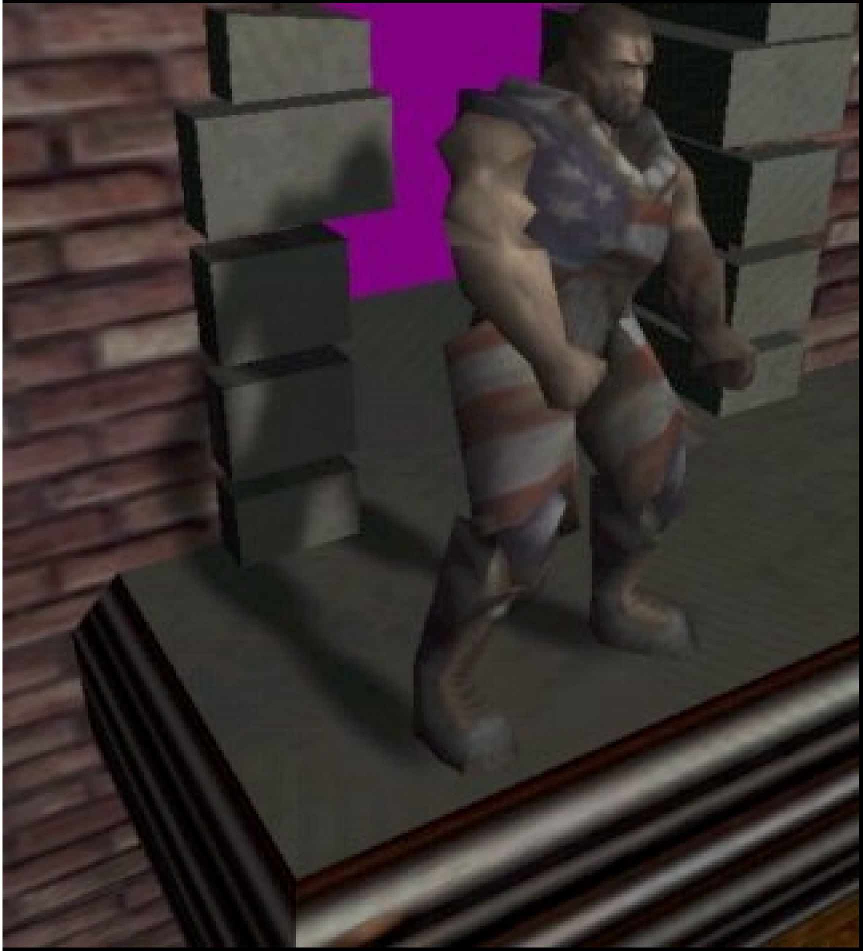




Their implementation:

- Part of off-line Radiosity Renderer.

Soft Shadow Textures Using Convolution



My implementation:

- Attempt at using variations of this technique for real time shadows in a game-like environment.
- Visualization for technique.
- OpenGL vertex lighting on original geometry (no subdivision for lighting).
- Source images pre-rendered.
- Not done yet. Not at all optimzied.
- Works at interactive framerates on newer hardware accelerated PCs.