

Hard Shadows

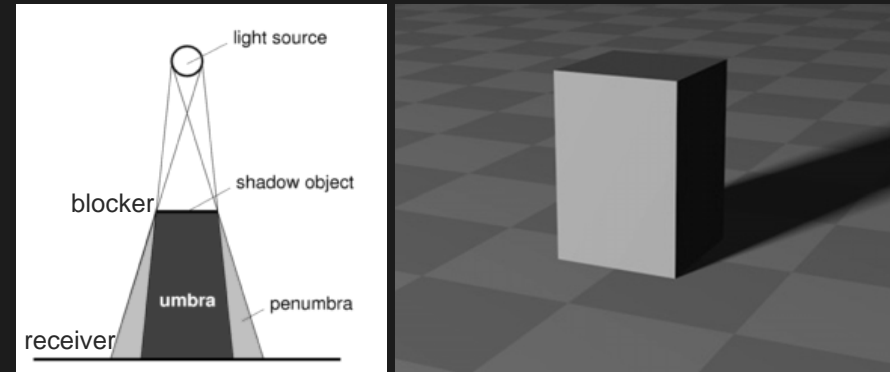
Digital Image Synthesis

1/4/2007

With slides from Eric Chan, Pradeep Sen, Marc Stamminger, Ravi Ramamoorthi, Brandon Lloyd

Why shadows

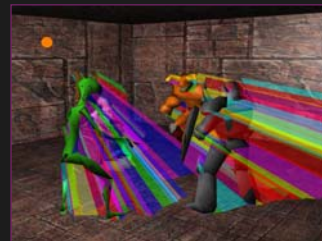
- Crucial for spatial and depth perception



Two Algorithms from the 1970's

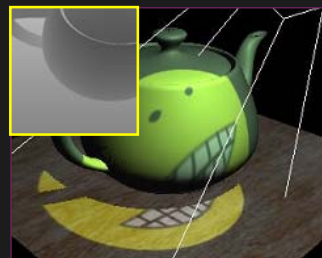
Shadow volumes (Crow 1977)

- Object-space
- Accelerated by hardware stencil buffer
- **Large fillrate consumption**



Shadow maps (Williams 1978)

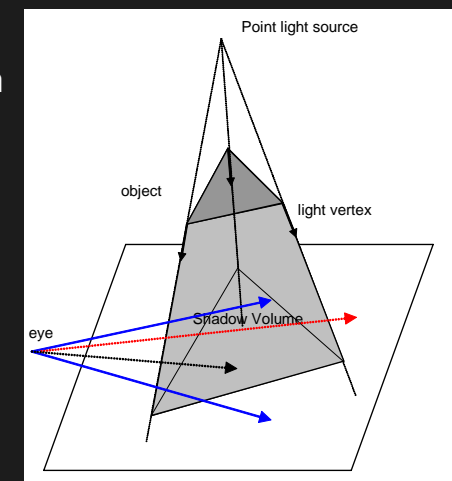
- Image-space
- Fast and simple
- Supported in hardware
- **Undersampling artifacts**



NVIDIA

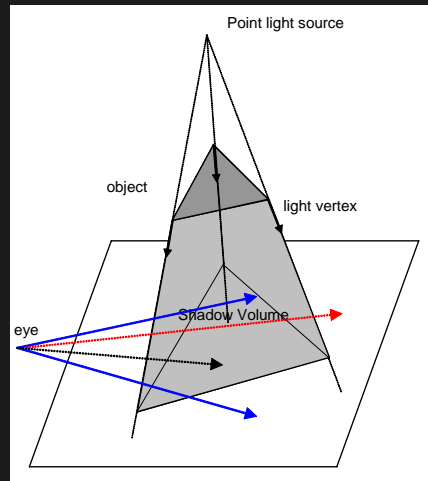
Shadow volumes

- Create volumes of space in shadow from a point light
- Each triangle creates 3 projecting quads



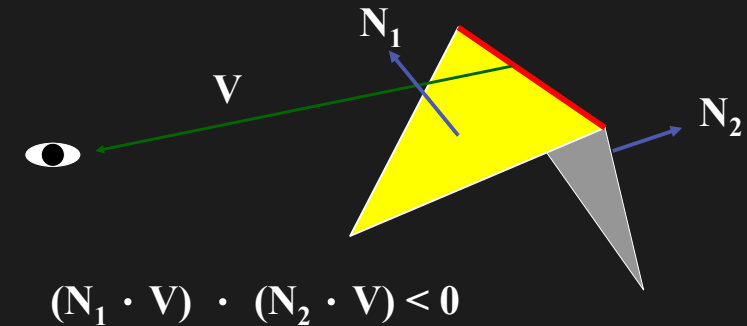
Using the volume

- To test a point, count the number of polygons between it and eye
- If more front-facing than back-facing polygons, then it is in shadow



Creating volumes

- Use the silhouette to speed up
- What is silhouette?

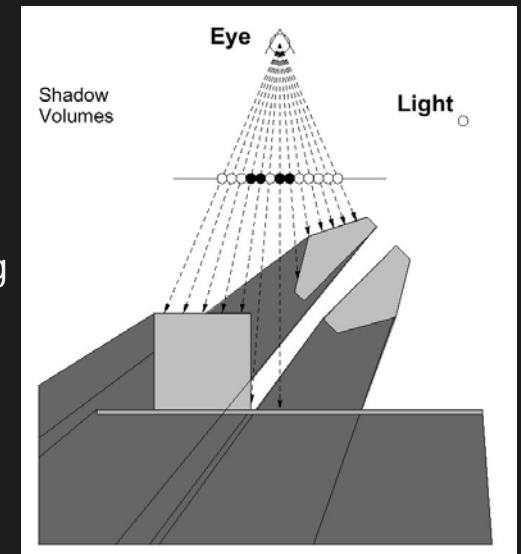


Shadow volume algorithm

- Finding volumes
- Rendering
 - Render scene into z-buffer, freeze z-buffer
 - Draw front-facing quads, increment stencil buffer
 - Draw back-facing quads, decrement stencil buffer
 - If (cnt==0) lit else shadowed

Multiple shadow volumes

- To test a point, count the number of polygons between it and eye
- If more front-facing than back-facing polygons, then it is in shadow



Shadow Mapping

Lance Williams: Brute Force in image space (shadow maps in 1978, but other similar ideas like Z buffer, bump mapping using textures and so on)

Completely image-space algorithm

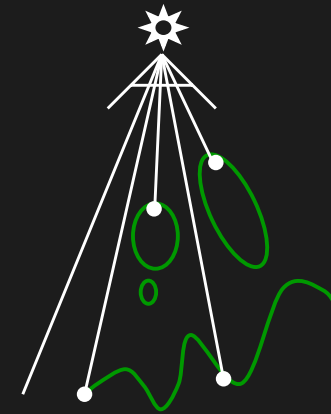
- no knowledge of scene's geometry is required
- must deal with aliasing artifacts

Well known software rendering technique

- Basic shadowing technique for Toy Story, etc.

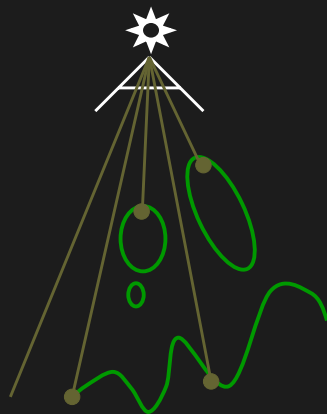
Phase 1: Render from Light

Depth image from light source



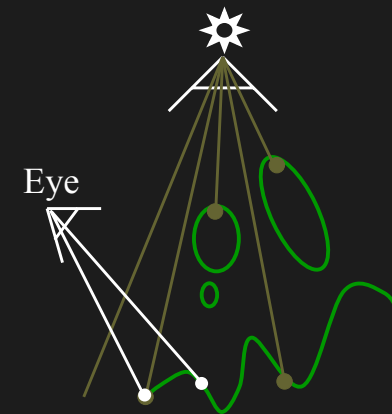
Phase 1: Render from Light

Depth image from light source



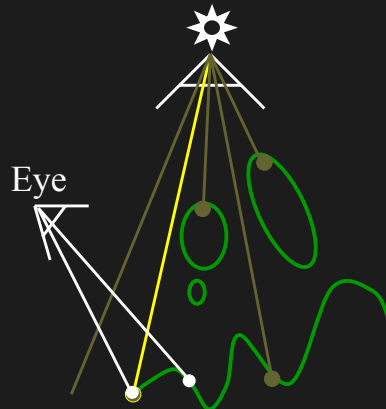
Phase 2: Render from Eye

Standard image (with depth) from eye



Phase 2+: Project to light for shadows

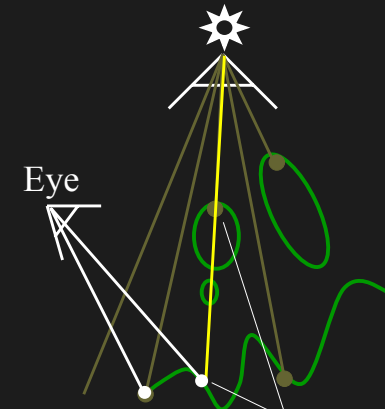
Project visible points in eye view back to light source



(Reprojected) depths match for light and eye. **VISIBLE**

Phase 2+: Project to light for shadows

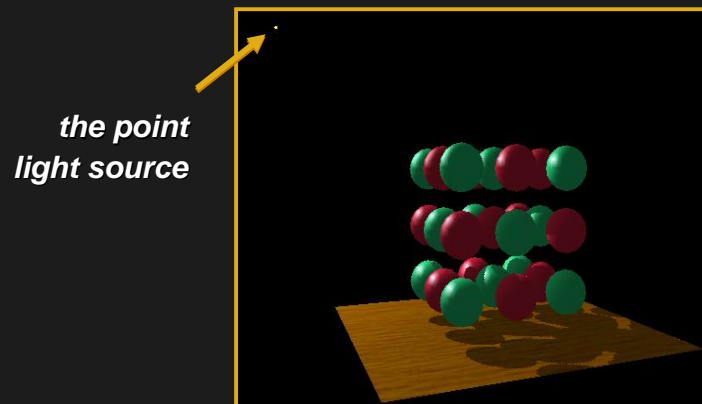
Project visible points in eye view back to light source



(Reprojected) depths from light, eye not the same. **BLOCKED!!**

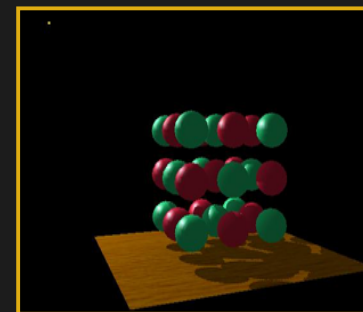
Visualizing Shadow Mapping

A fairly complex scene with shadows

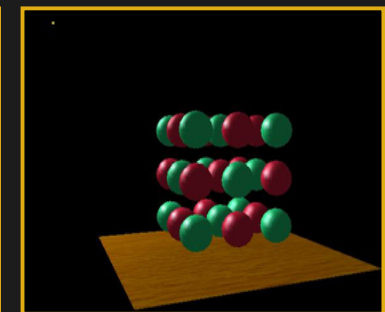


Visualizing Shadow Mapping

Compare with and without shadows



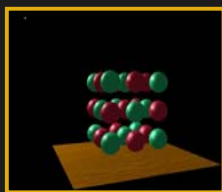
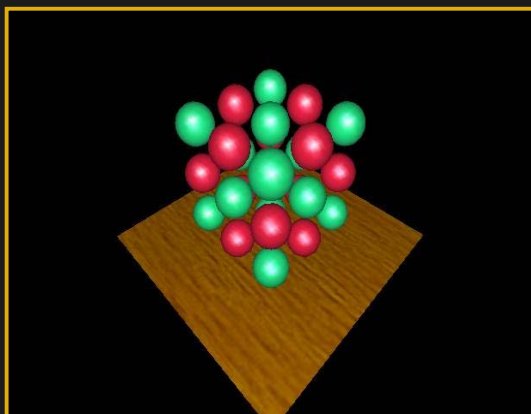
with shadows



without shadows

Visualizing Shadow Mapping

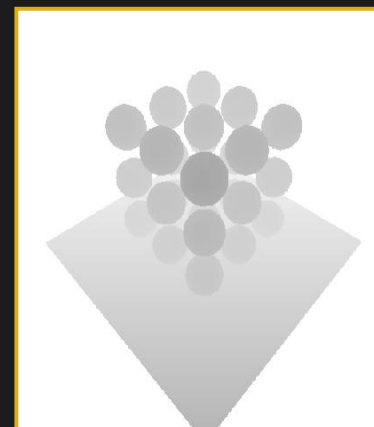
The scene from the light's point-of-view



FYI: from the eye's point-of-view again

Visualizing Shadow Mapping

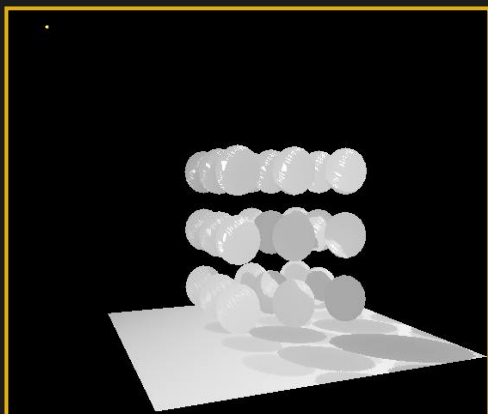
The depth buffer from the light's point-of-view



FYI: from the light's point-of-view again

Visualizing Shadow Mapping

Projecting the depth map onto the eye's view

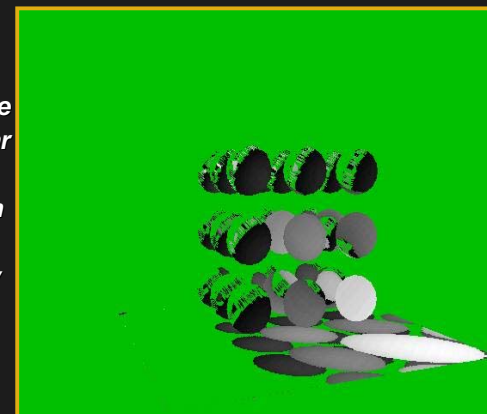


FYI: depth map for light's point-of-view again

Visualizing Shadow Mapping

Comparing light distance to light depth map

Green is where the light planar distance and the light depth map are approximately equal

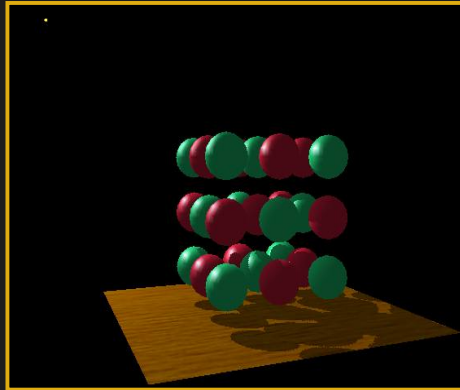


Non-green is where shadows should be

Visualizing Shadow Mapping

Scene with shadows

Notice how specular highlights never appear in shadows

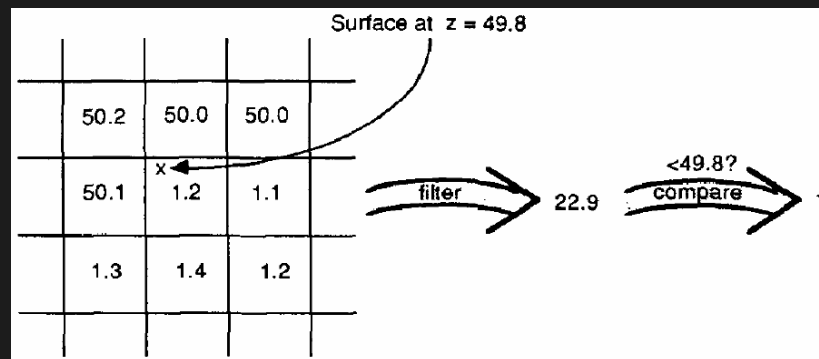


Notice how curved surfaces cast shadows on each other

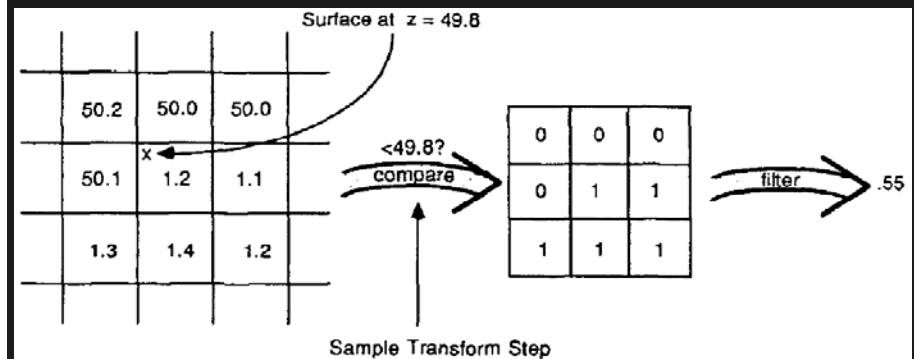
Comparison

- Shadow volumes
 - Pros: accurate, high-quality
 - Cons: fill-rate limited, hard to implement robustly
- Shadow maps
 - Pros: fixed resolution, fast, simple
 - Cons: bias, **aliasing**

Percentage Closer filtering [Reeves 1987]

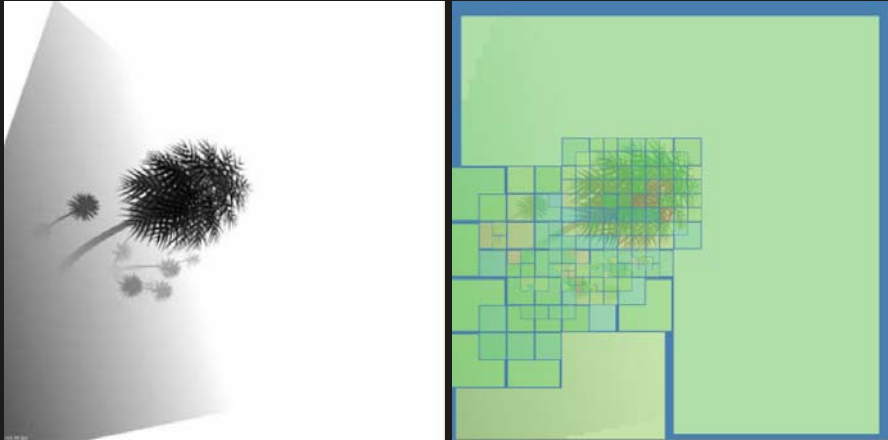


Percentage Closer filtering [Reeves 1987]



Provides anti-aliasing at shadow map edges, not soft shadows in the umbra/penumbra sense

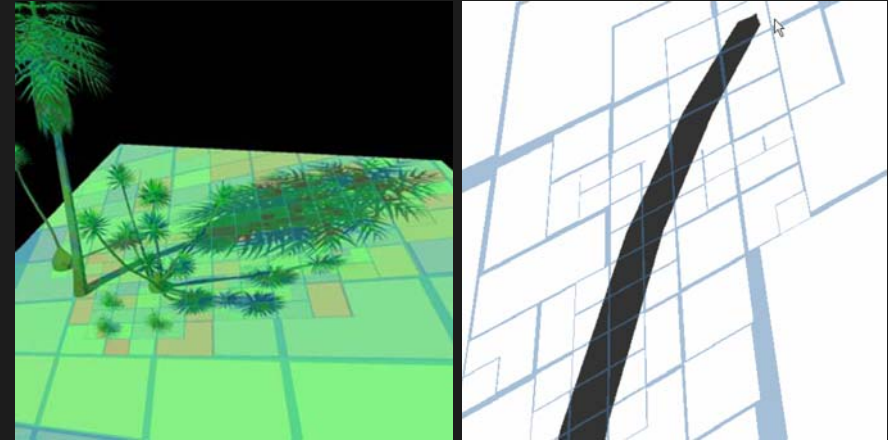
Adaptive shadow maps [Fernando 2001]



high-resolution shadow map

adaptive shadow map

Adaptive shadow maps



rendering

details

Adaptive shadow maps

Pros:

- Progressive and view-dependent
- Confined to a user-specified memory limit, use LRU policy to delete nodes

Cons:

- Not easy to map to GPU; [Lefohn et. al. 2005] has mapped it to GPU recently

Perspective Shadow Maps

Marc Stamminger
George Drettakis

SIGGRAPH 2002

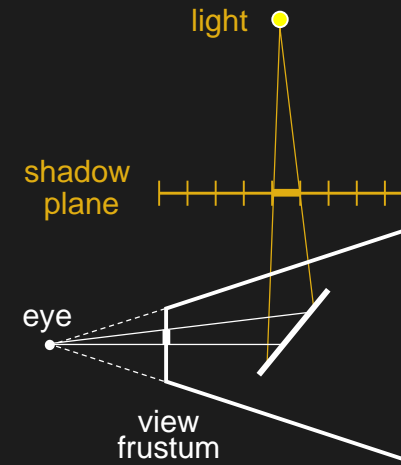
shadow map aliasing

prone to aliasing
when zooming into
shadow boundaries

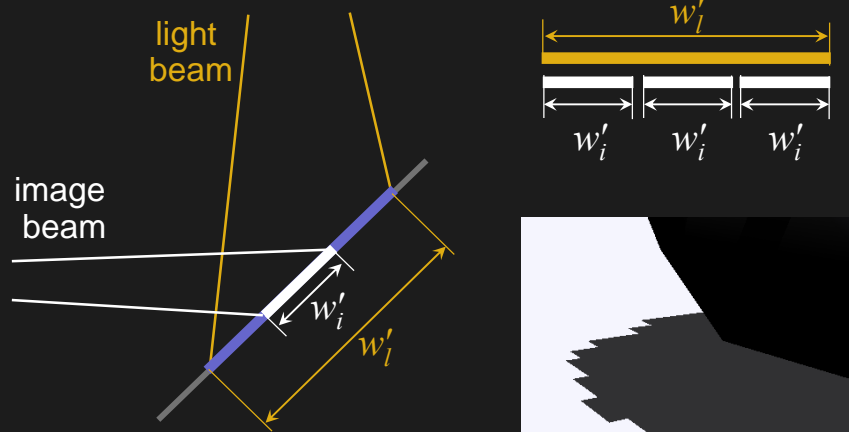


single shadow map pixel

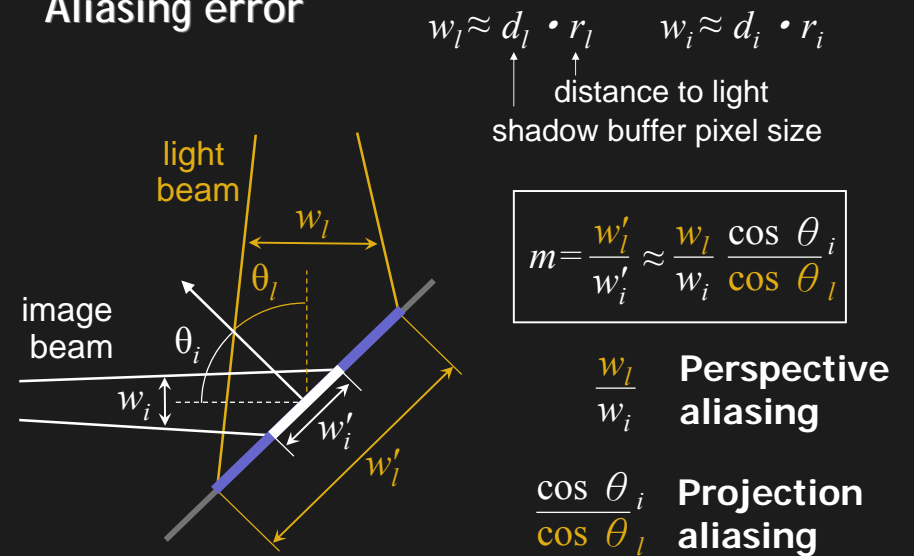
Aliasing error



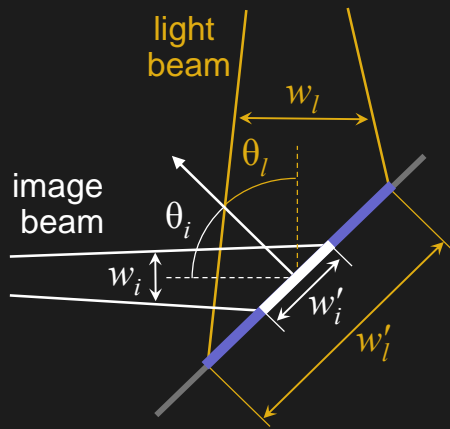
Aliasing error



Aliasing error



Aliasing error



$$m = \frac{w'_l}{w_i} \approx \frac{w_l \cos \theta_l}{w_i \cos \theta_l}$$

$\frac{w_l}{w_i}$ **Perspective aliasing**

~~$\frac{\cos \theta_l}{\cos \theta_l}$ **Projection aliasing**~~

Controlling aliasing error

To eliminate perspective aliasing:

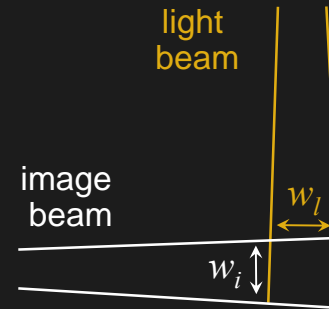
$$w_l \leq w_i$$

Light beam width depends on:

- Shadow map resolution
- Parameterization

We have freedom to choose where and how to set up shadow buffer

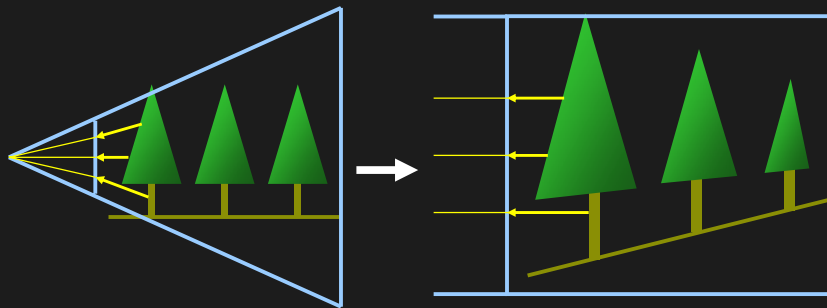
It should be adaptive to view; this means that shadow buffer has to be recomputed for view changes



perspective transformation

normalizing transformation
[Foley] p.258-271

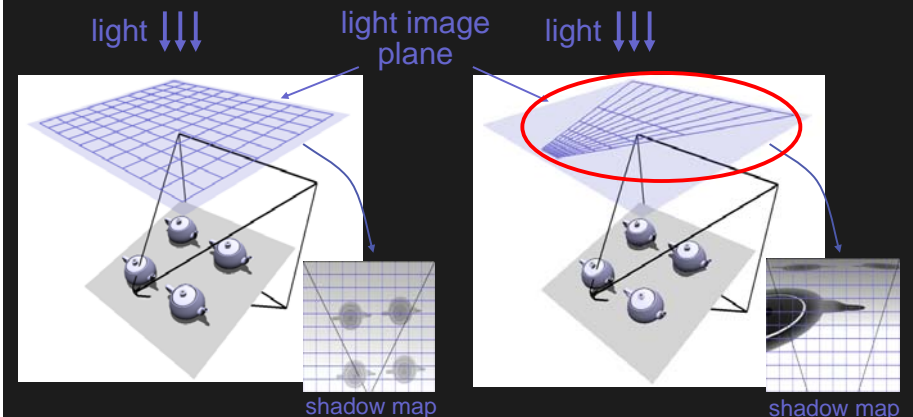
fixed w_i



world space

post-perspective space

Parameterization

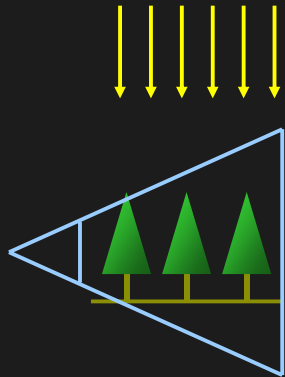


Standard

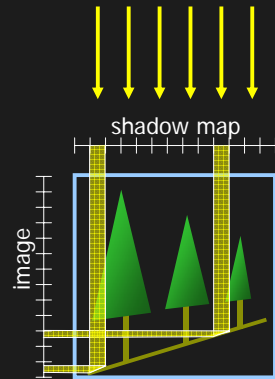
Perspective warped

perspective shadow map

standard shadow map

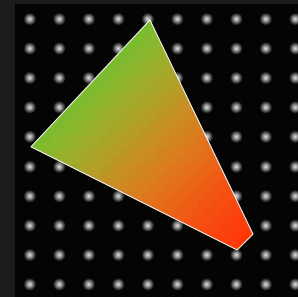


perspective shadow map

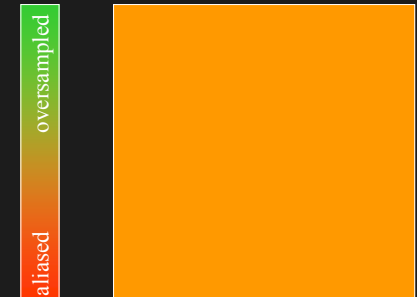


perspective shadow map

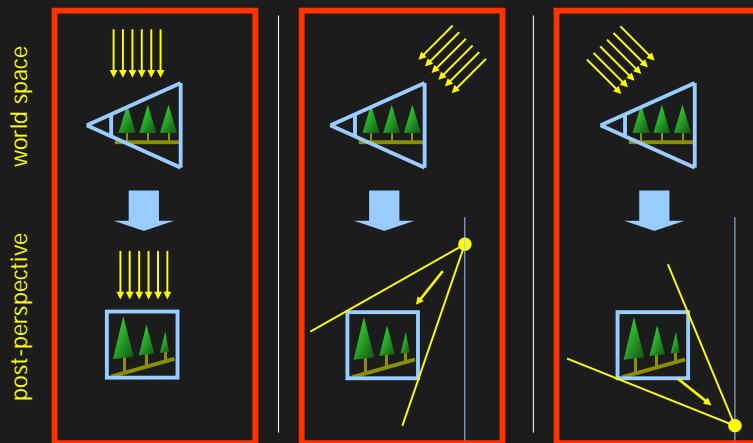
standard shadow map



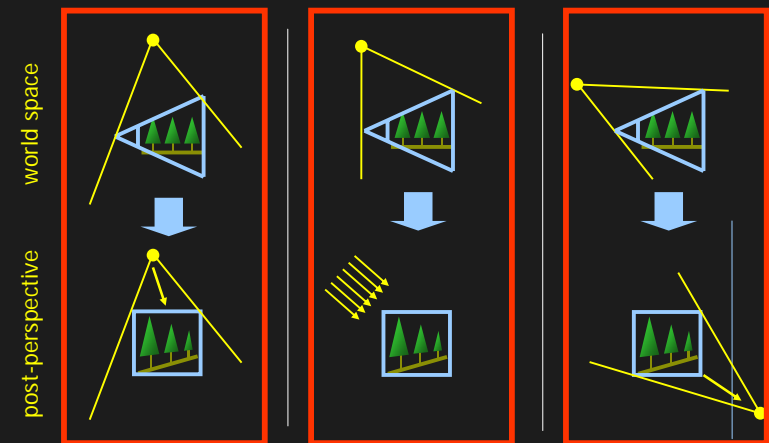
perspective shadow map



parallel light transformation



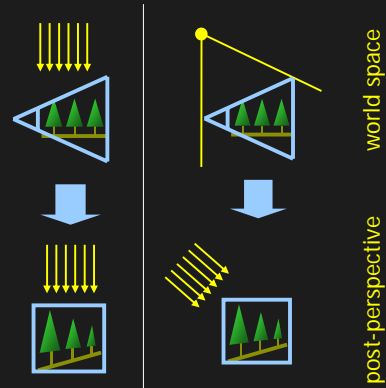
point light transformation



discussion

best case: fixed w_l

- parallel light in post-perspective space
- no new perspective distortion



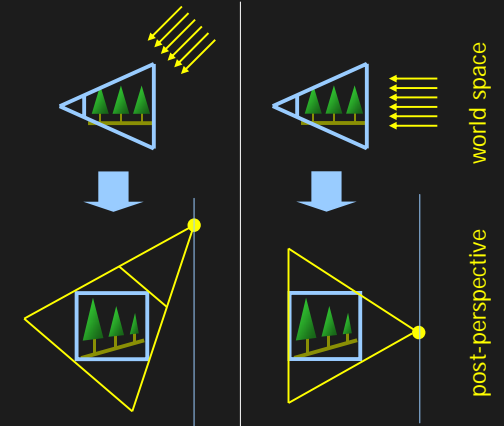
discussion

non-optimal case:

- point light close to frustum

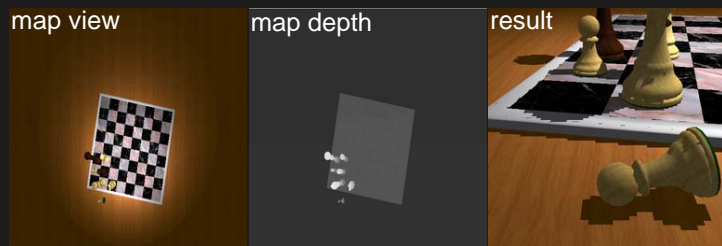
worst case:

- becomes uniform shadow map

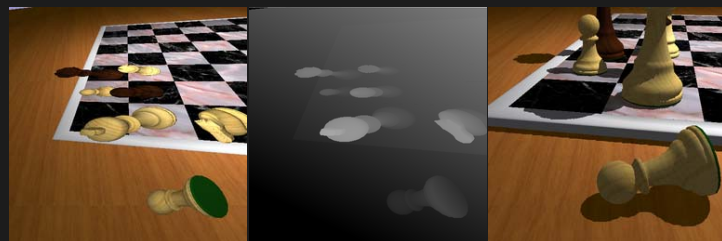


Perspective shadow map

Shadow map



Perspective Shadow map



Shadow maps



Perspective Shadow maps



conclusion

perspective shadow maps

- shadow map in post-perspective space
- just another shadow map matrix
- non-uniform shadow map resolution
- not always possible to find a good solution
- needs recomputation per frame
- minimal overhead for dynamic scenes

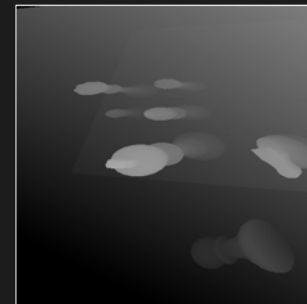
Shadow Silhouette Maps

Pradeep Sen
Mike Cammarano
Pat Hanrahan

SIGGRAPH 2003
(slides from Eric Chan)

Motivation

Why another shadow algorithm?
Why not use perspective shadow maps?



Stamminger and Drettakis, SIGGRAPH 2002

Perspective Shadow Maps

Addresses perspective aliasing

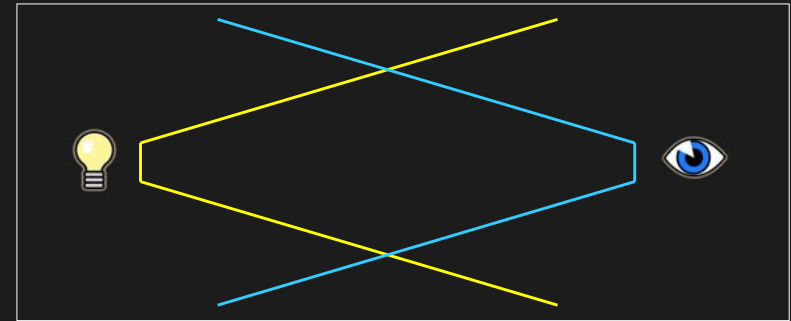
Optimizes distribution of depth samples

Difficulties:

Does not handle projection aliasing

Dueling frusta problem

Dueling Frusta Problem

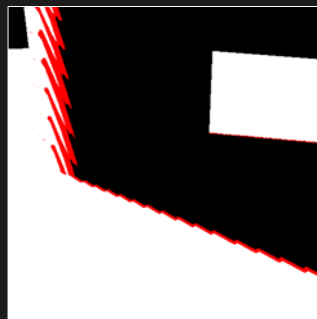
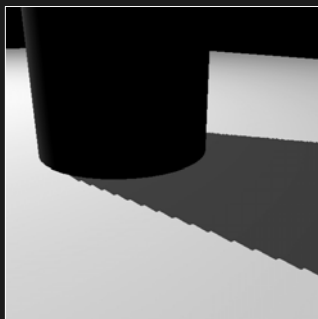


Observation

Shadow maps

undersampling can occur anywhere

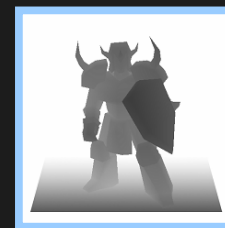
artifacts visible only at shadow edges



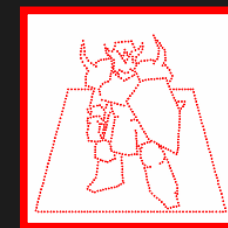
How To Fix Silhouettes?

One solution:

use a better silhouette approximation



depth map



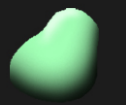
silhouette map



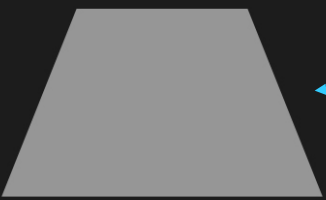
Shadow Map (Review)



light source

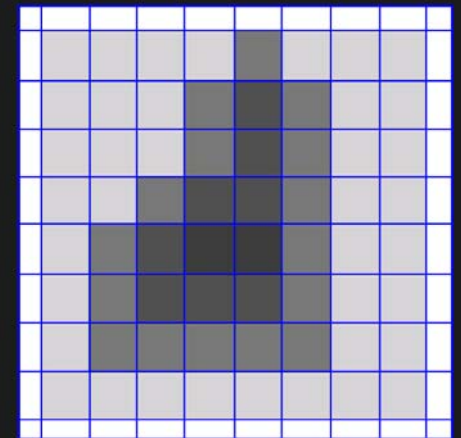
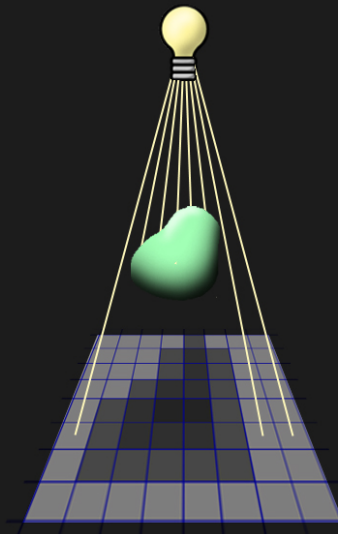


blocker



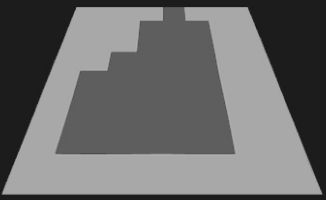
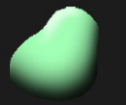
receiver

Shadow Map (Review)



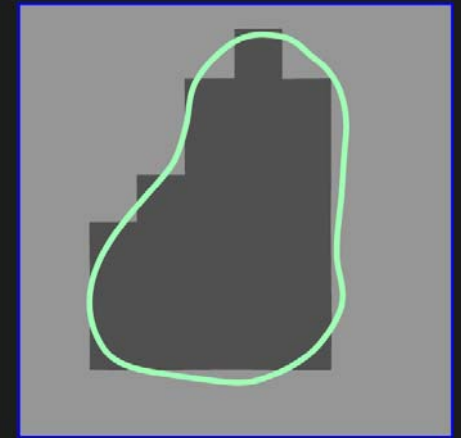
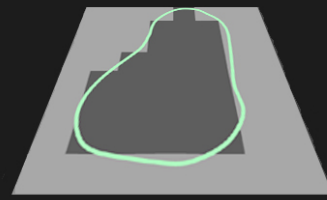
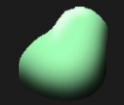
depth map

Shadow Map (Review)



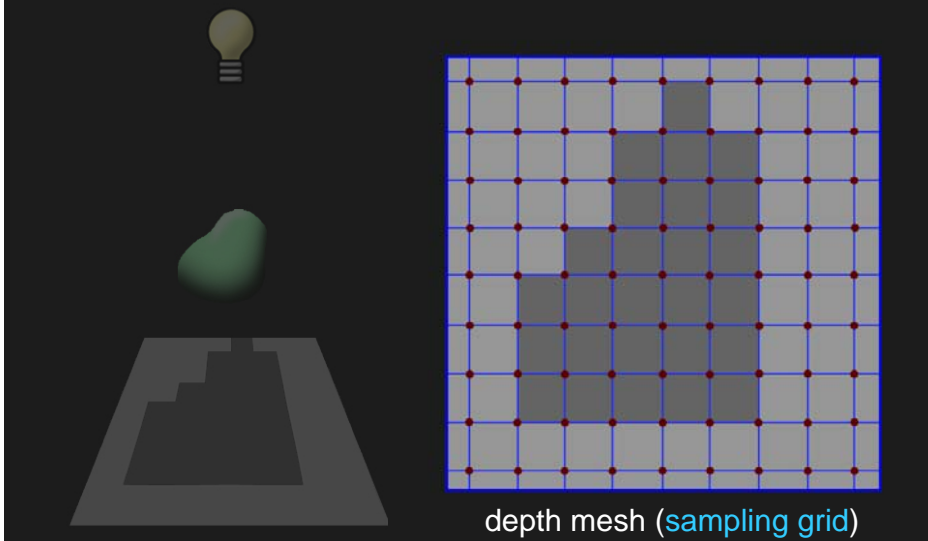
depth map

Shadow Map (Review)

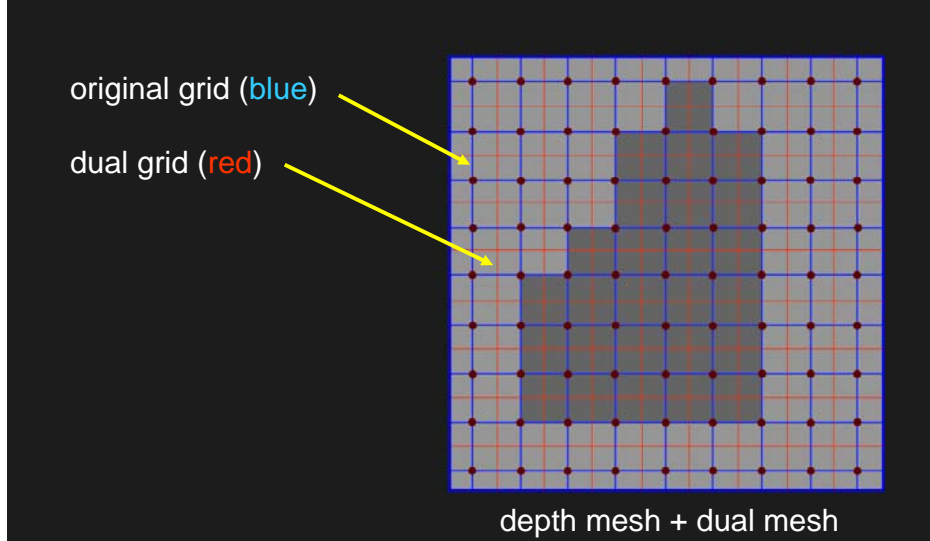


depth map

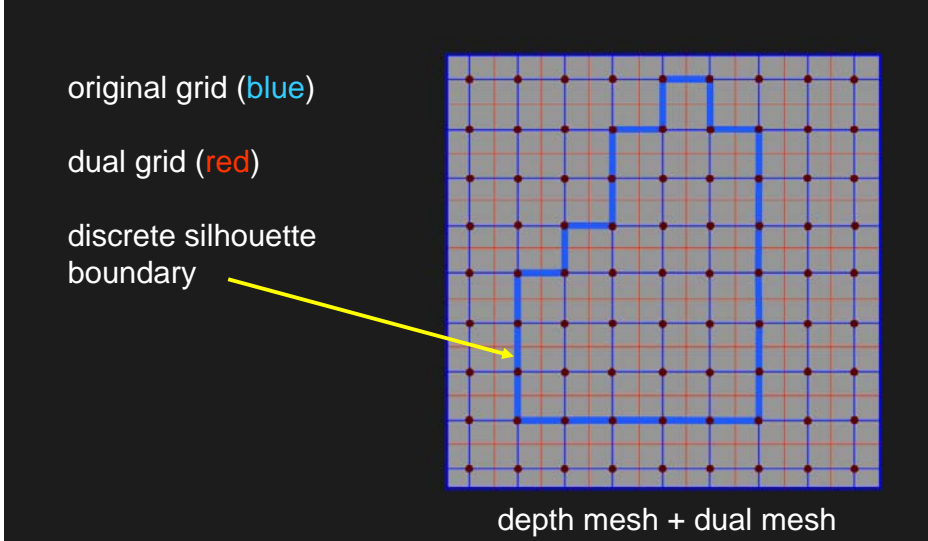
Depth Mesh



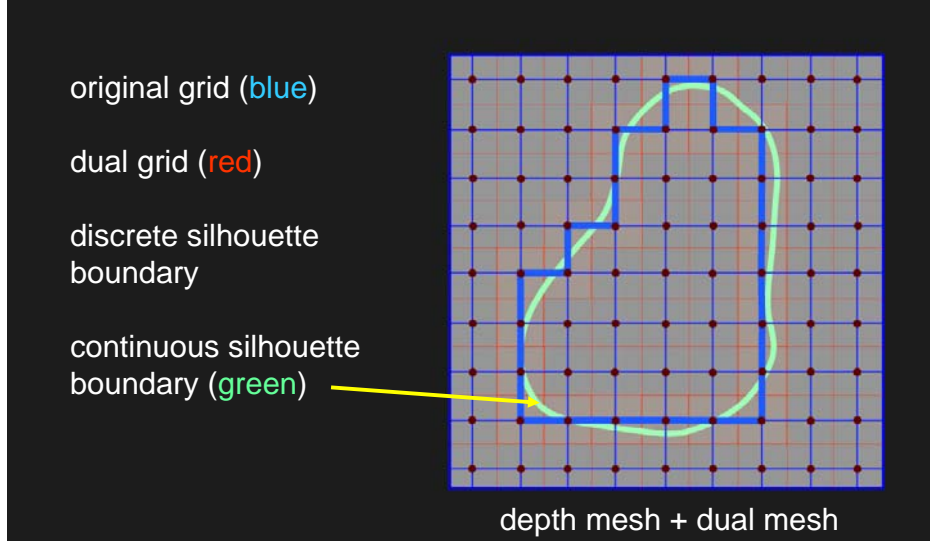
Depth Mesh



Depth Mesh



Depth Mesh



Depth Mesh

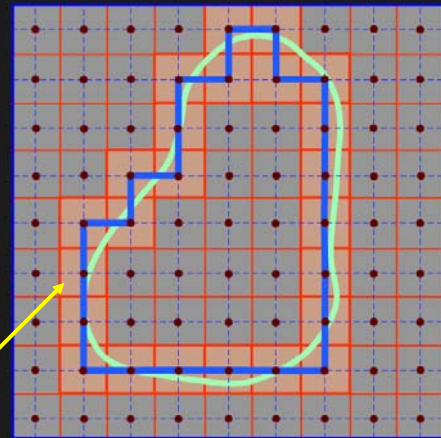
original grid (blue)

dual grid (red)

discrete silhouette boundary

continuous silhouette boundary (green)

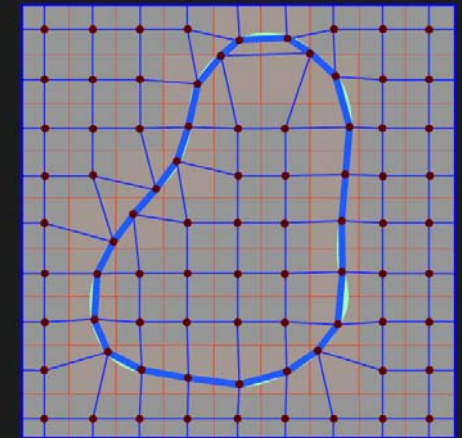
silhouette map pixels



depth mesh + dual mesh

Depth Mesh Deformation

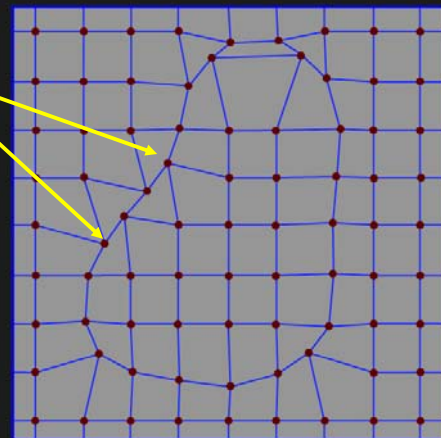
Move depth samples to lie on silhouette curve



deformed depth mesh

Depth Mesh Deformation

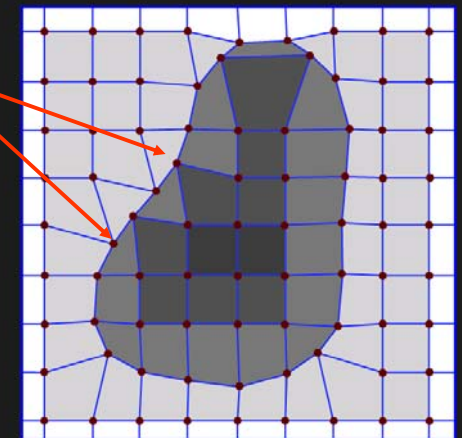
adjusted depth samples



deformed depth mesh

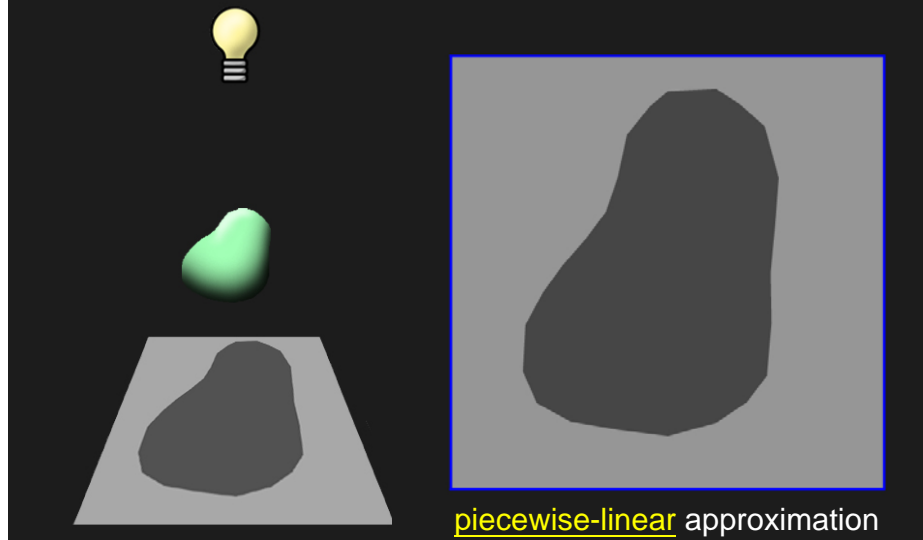
Depth Mesh Deformation

adjusted depth samples

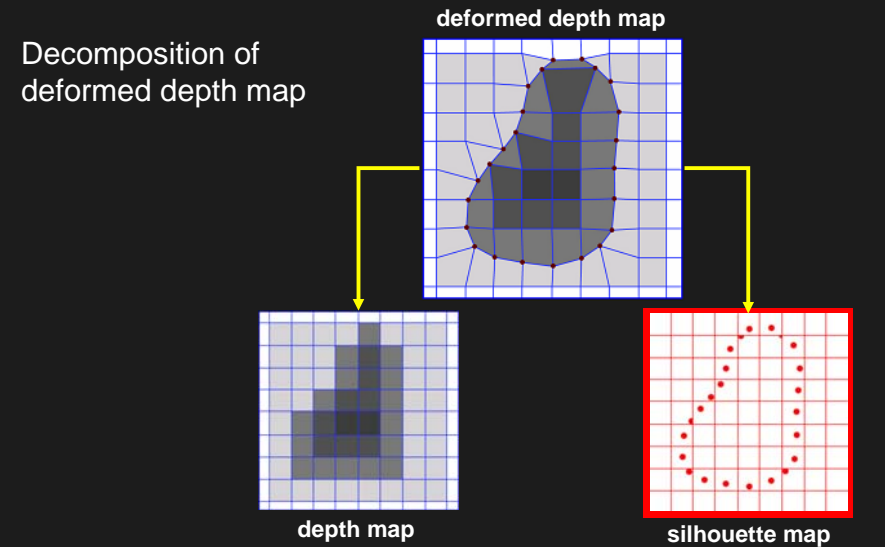


deformed depth mesh

Better Approximation



Silhouette Map



What is a Silhouette Map?

Many ways to think about it:

- Edge representation
- 2D image, same resolution as depth map
- Offset from depth map by $\frac{1}{2}$ pixel in x , y
- Stores xy -coordinates of silhouette points
- Stores only one silhouette point per texel
- Piecewise-linear approximation

Algorithm

Algorithm Properties

Scalable

Treats perspective and projection
aliasing

Supports dynamic scenes

Maps to graphics hardware



Algorithm Overview

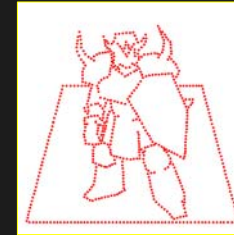
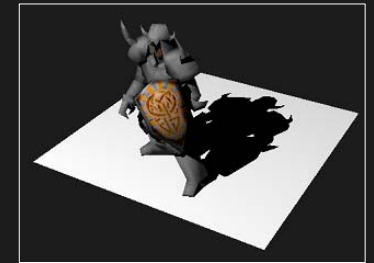


Image-space algorithm



Algorithm Overview

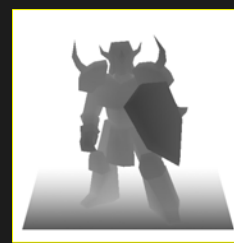


💡 Step 1

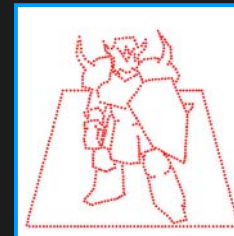


Create depth map

Algorithm Overview

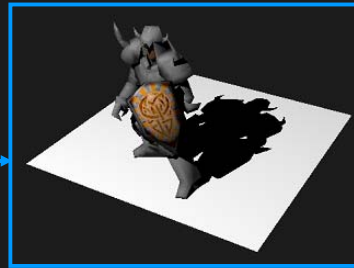
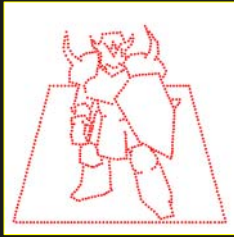


💡 Step 2



Create silhouette map

Algorithm Overview



Step 3

Render scene and shadows

Create Depth Map

Same as in regular shadow maps



Identify Silhouette Edges

Find object-space silhouettes (**light's view**)



Create Silhouette Map

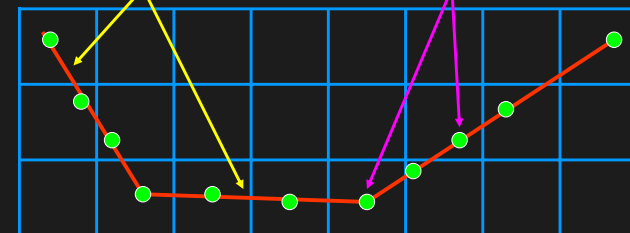
Rasterize silhouette edges (**light's view**)

Find points that lie on silhouette edges

Store one such point per texel

silhouette edges

silhouette points

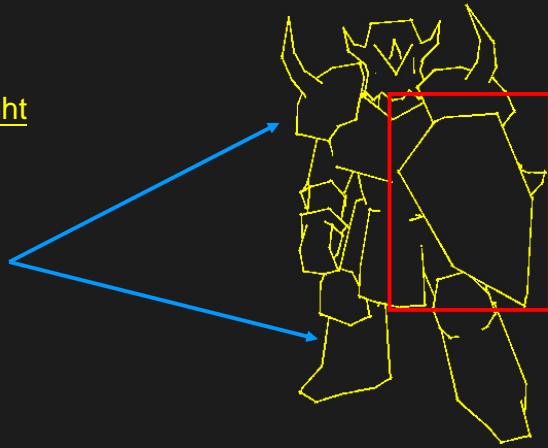


Compute Silhouette Points

Example:

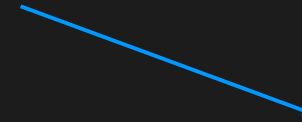
point of view of light

silhouette edges



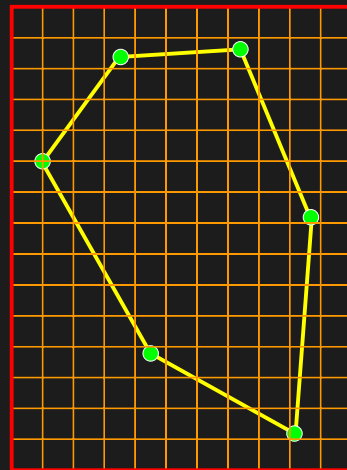
Compute Silhouette Points

silhouette map (dual grid)



Compute Silhouette Points

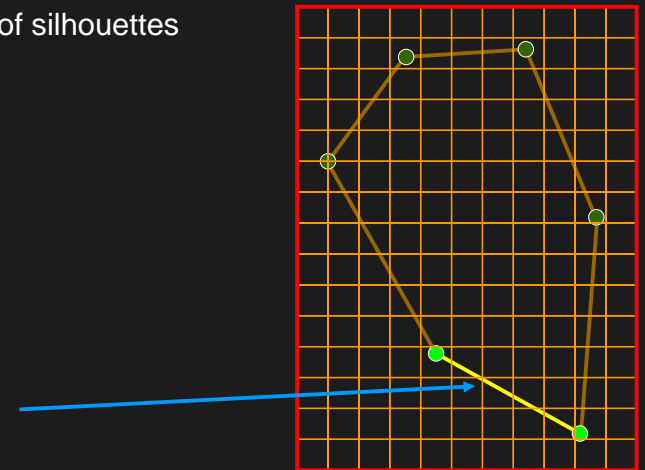
rasterization of silhouettes



Compute Silhouette Points

rasterization of silhouettes

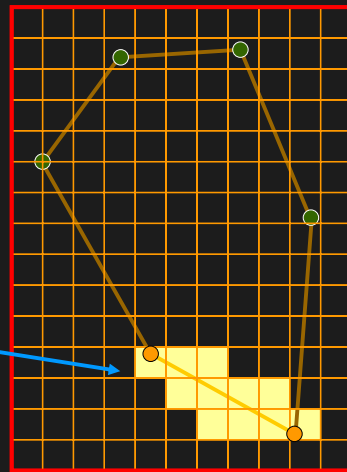
pick an edge



Compute Silhouette Points

rasterization of silhouettes

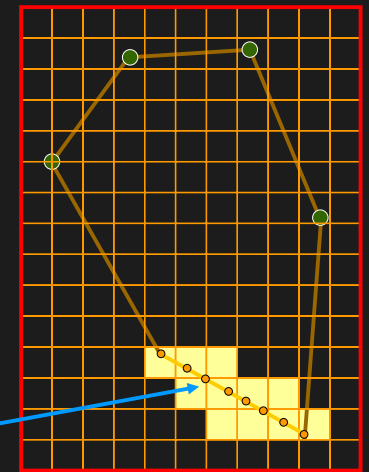
rasterize edge conservatively:
be sure to generate fragments
for silhouette pixels



Compute Silhouette Points

rasterization of silhouettes

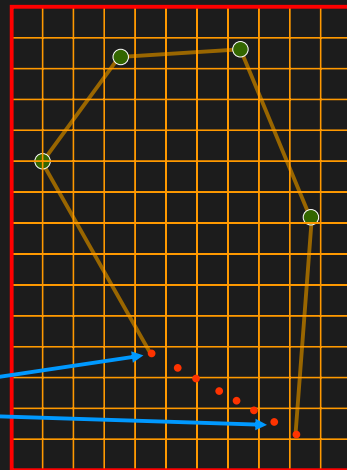
for each fragment:
pick a point on the edge



Compute Silhouette Points

rasterization of silhouettes

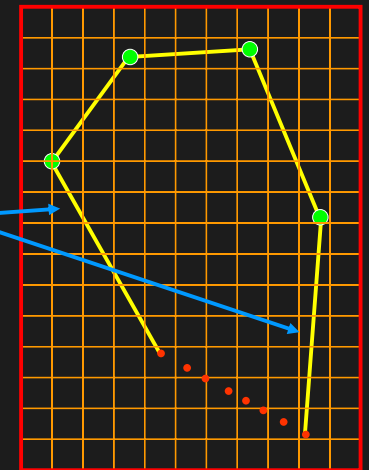
silhouette points



Compute Silhouette Points

rasterization of silhouettes

do the same for other edges



Compute Silhouette Points

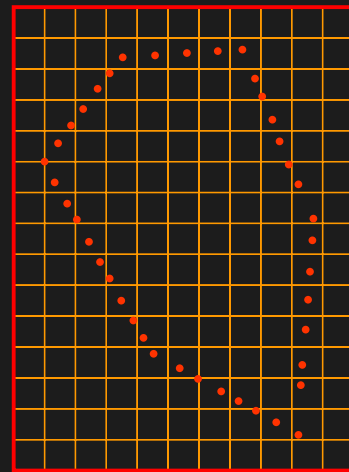
rasterization of silhouettes

completed silhouette map →

subtle issues:

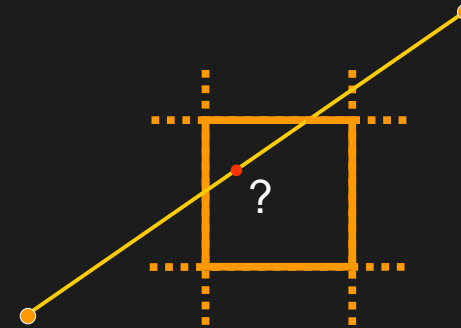
- only one point per texel
- new values overwrite old ones

how to pick silhouette points?



Picking Silhouette Points

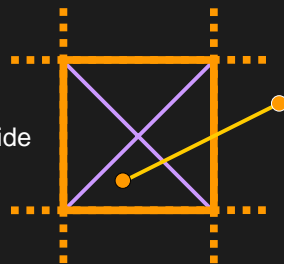
Pick a point on the line that lies inside the texel



Silhouette Point Algorithm

Case 1:

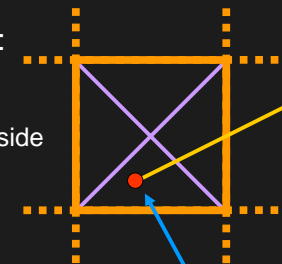
vertex inside



Silhouette Point Algorithm

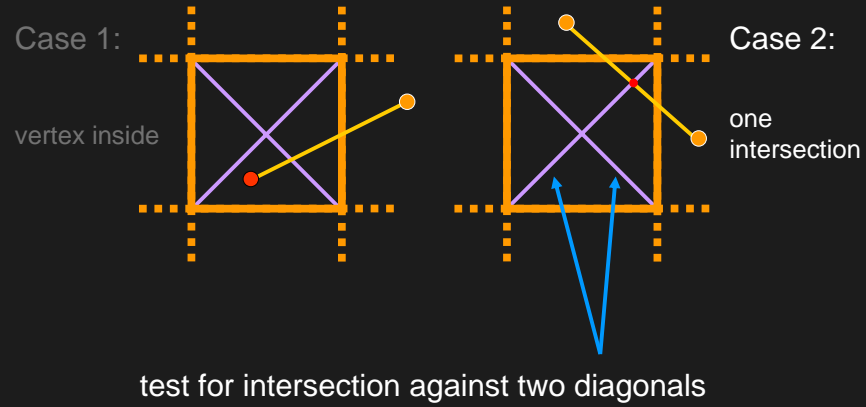
Case 1:

vertex inside

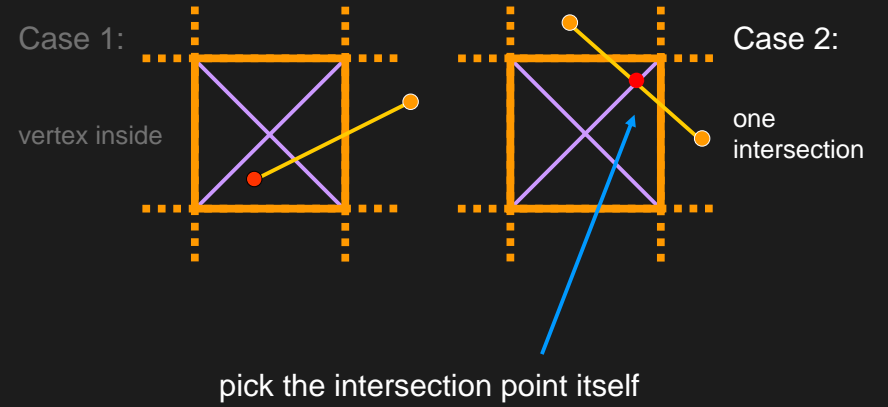


pick the vertex itself

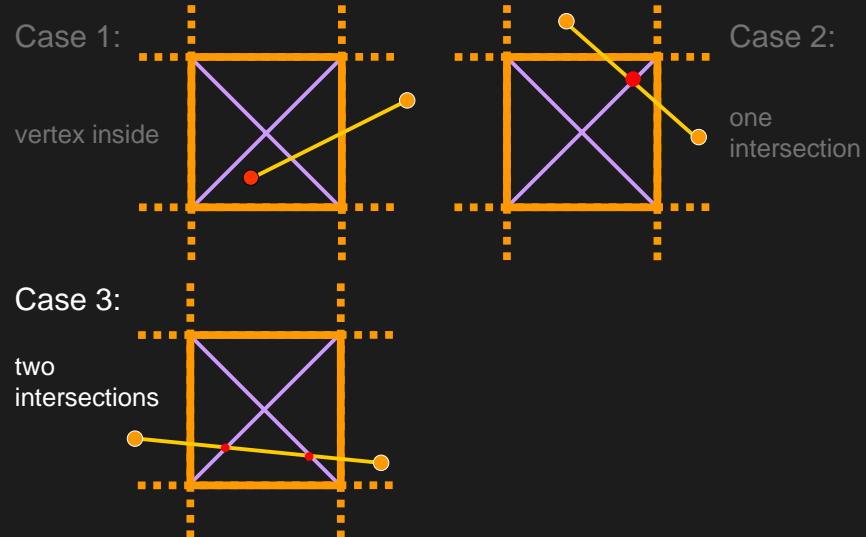
Silhouette Point Algorithm



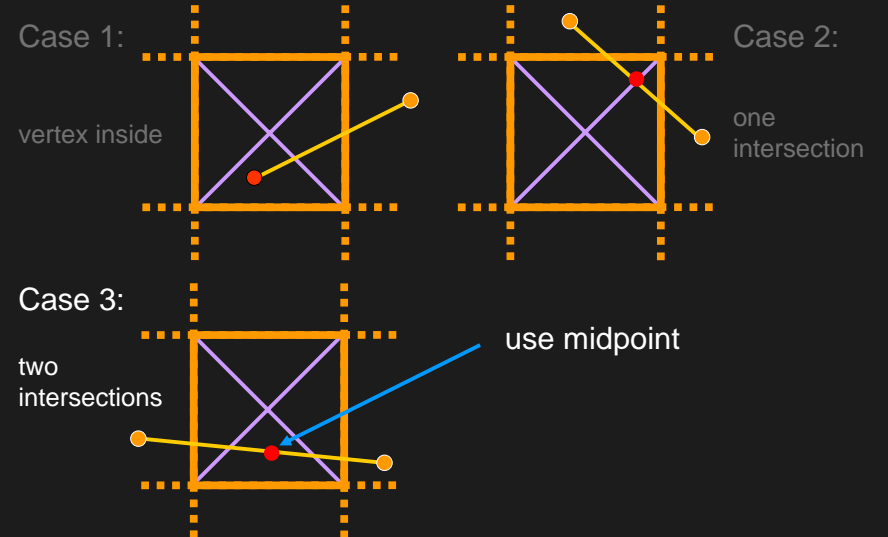
Silhouette Point Algorithm



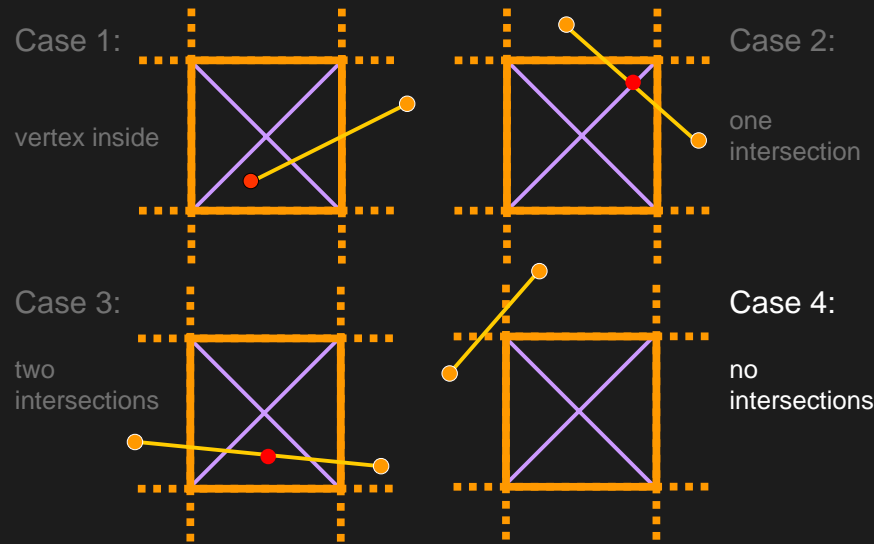
Silhouette Point Algorithm



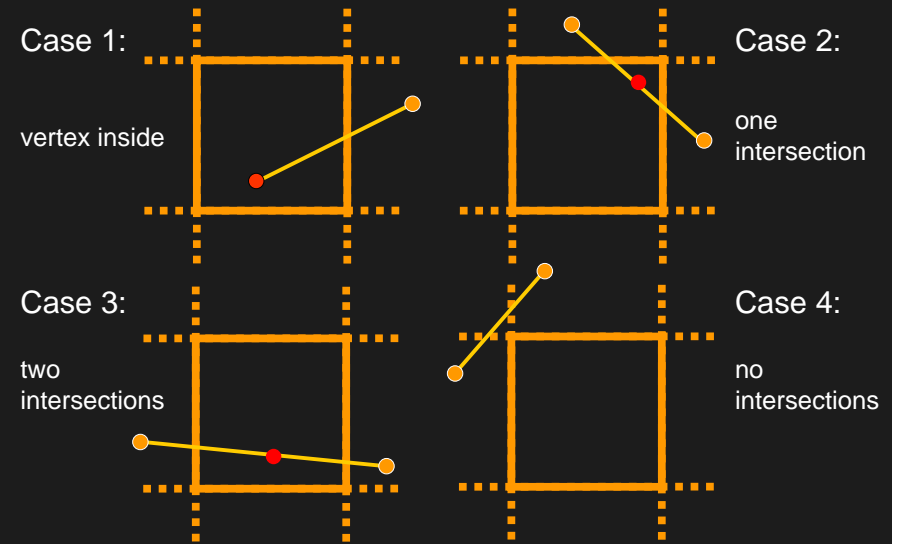
Silhouette Point Algorithm



Silhouette Point Algorithm



Silhouette Point Algorithm



Render scene

How to compute shadows?

Split problem into two parts:

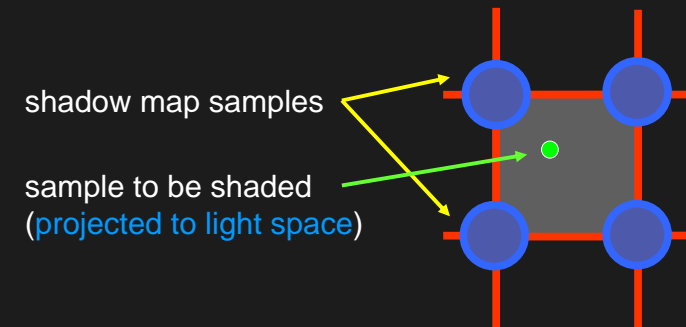
non-silhouette pixels: use [shadow map](#)

silhouette pixels: use [silhouette map](#)

Find Silhouette Pixels

Project sample into light space

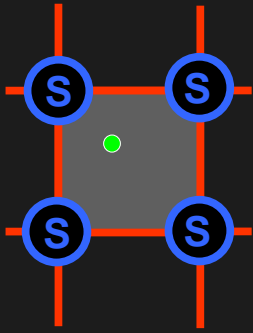
Compare depth against 4 nearest samples in shadow map



Find Silhouette Pixels

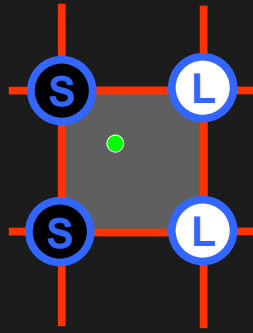
Case #1

results agree:
non-silhouette pixel



Case #2

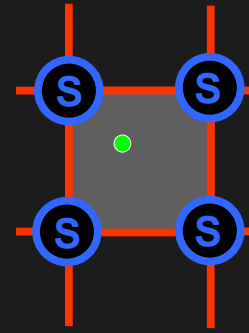
results disagree:
silhouette pixel



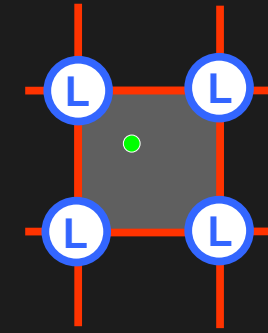
Treat Non-Silhouette Pixels

Easy: use depth comparison result

in shadow

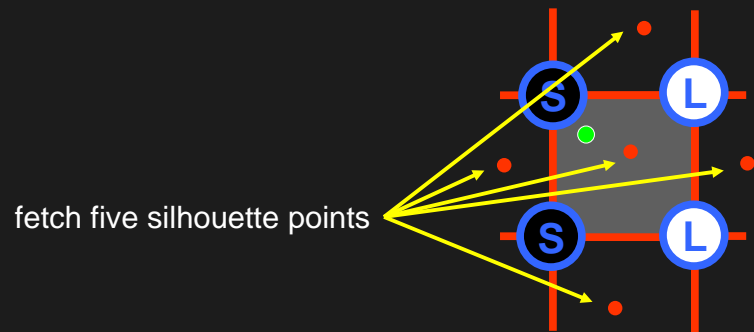


illuminated



Treat Silhouette Pixels

Reconstruct edge using silhouette map



Treat Silhouette Pixels

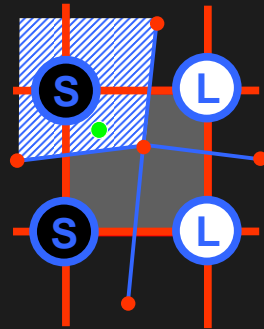
Reconstruct edge using silhouette map



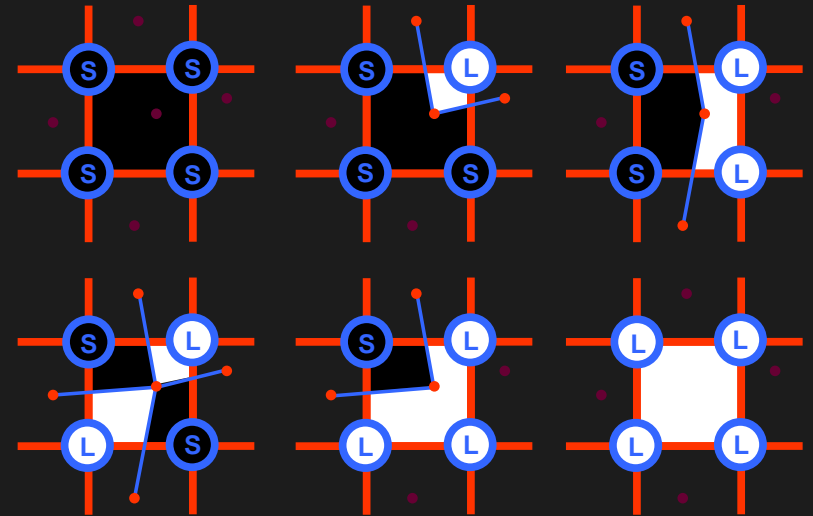
Treat Silhouette Pixels

Shade sample according to quadrant

example: sample in shadow



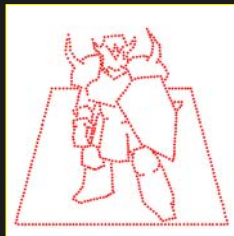
Six Combinations



Algorithm Recap



Image-space algorithm



Algorithm Recap (1 of 3)

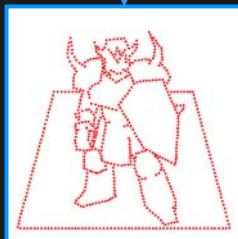


Create depth map



Easy: just like regular shadow map

Algorithm Recap (2 of 3)

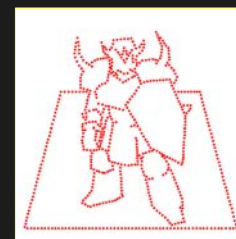


Create silhouette map



Rasterize silhouette edges
Pick silhouette points, 1 per texel

Algorithm Recap (3 of 3)

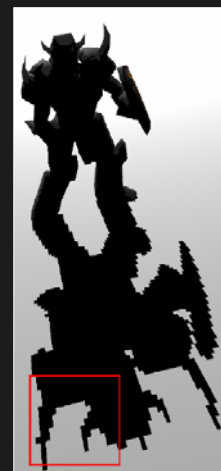


Render scene and shadows

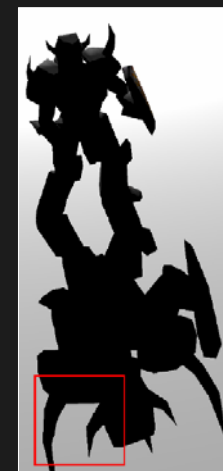
Fetch local silhouette points
Reconstruct shadow edge

Examples and Analysis

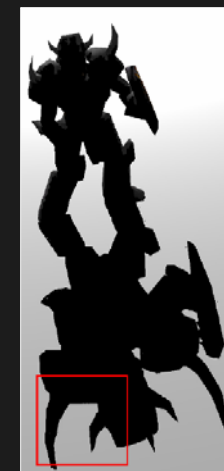
Example 1



shadow maps



shadow volumes



silhouette maps

Example 1 (closeup)

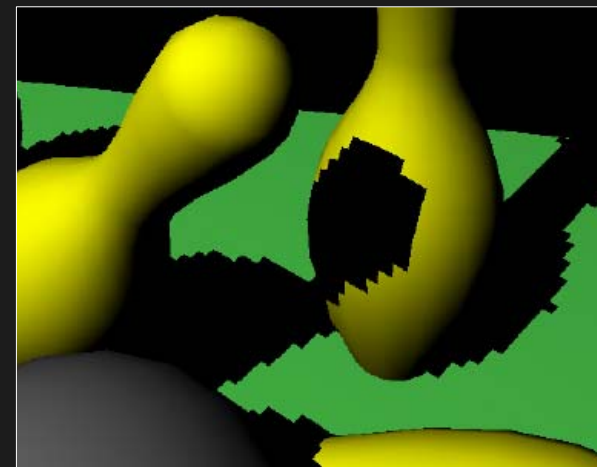


shadow maps

shadow volumes

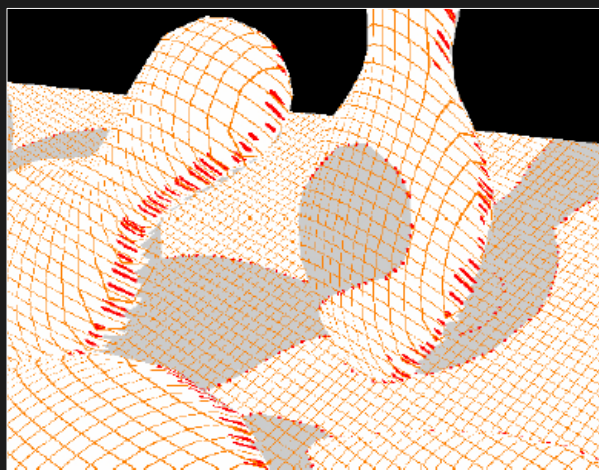
silhouette maps

Example 2



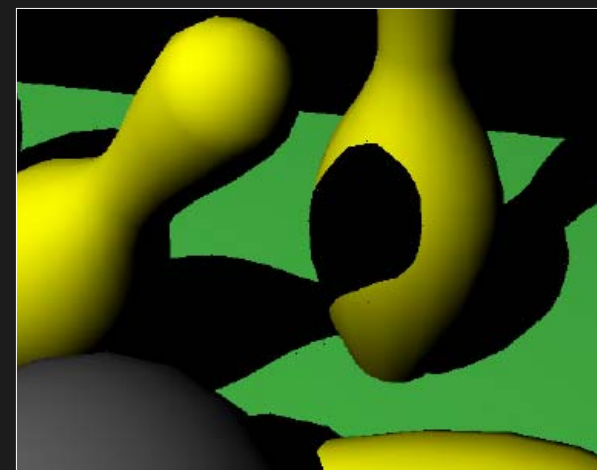
shadow maps

Example 2



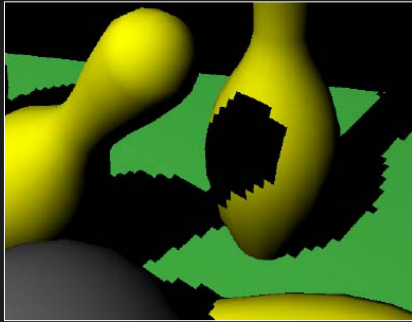
projected silhouette map

Example 2

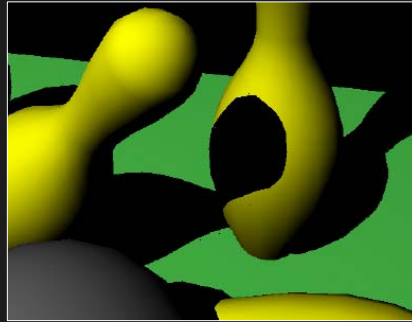


shadows using silhouette map

Quality Comparison



shadow map

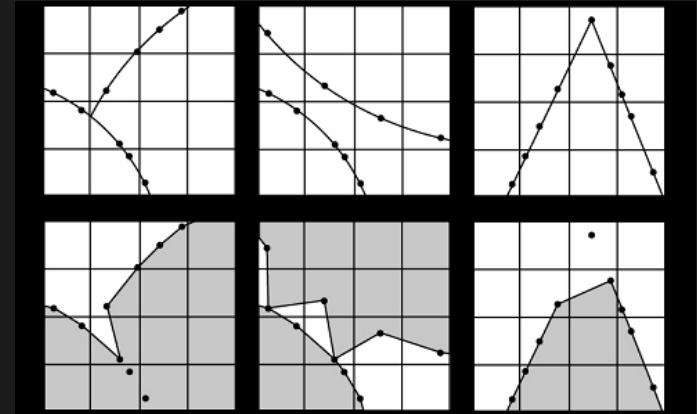


silhouette map

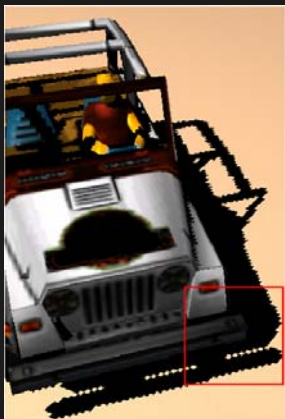
Artifacts

Silhouette map: one point per texel

Multiple edges inside a texel



Artifacts



shadow maps



shadow volumes

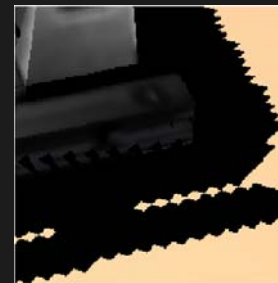


silhouette maps

Artifacts (closeup)

Artifacts due to multiple edges

More noticeable when animated



shadow maps



shadow volumes



silhouette maps

Algorithm Comparison

Perspective Shadow Maps:

- same generality as shadow maps
- minimal overhead (2 passes)
- doesn't address aliasing in all cases

Shadow Silhouette Maps:

- addresses aliasing more generally
- more overhead (3 passes + big shaders)
- less general than shadow maps

Combination of Algorithms

Why not combine techniques?

Perspective shadow map:

Optimizes depth sample distribution

More samples closer to viewer

Shadow silhouette map:

Optimizes depth sample information

Exact silhouette edge locations

Summary

Image-space algorithm

Silhouette map: deformed depth map

Piecewise-linear approximation

Scalable

Compared to (perspective) shadow maps:

Removes aliasing in more cases

Additional overhead and requirements

An Efficient Hybrid Shadow Rendering Algorithm

Eric Chan

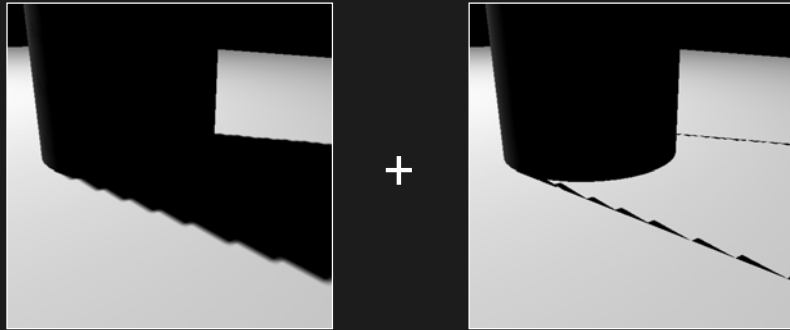
Fedro Durand

EGSR 2004

Not Another Talk on Shadows?!

Main ideas:

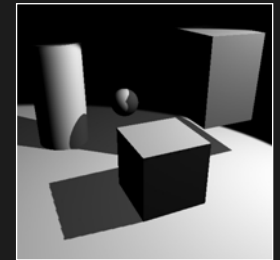
- combination of shadow maps + shadow volumes
- computation masks



Fillrate Problem

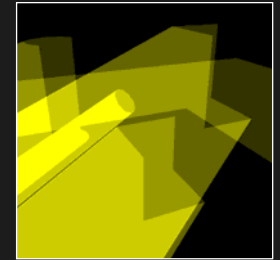
Lots and lots of fillrate!

- rasterization
- stencil updates



Why?

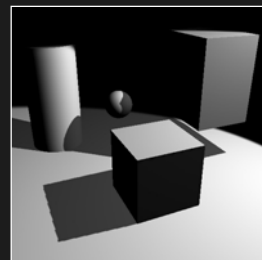
- polygons have large screen area
- polygons overlap



Fillrate Problem

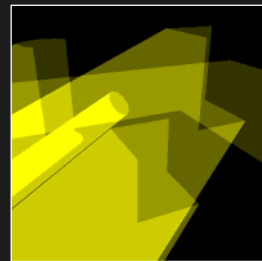
Lots and lots of fillrate!

- rasterization
- stencil updates



Why?

- polygons have large screen area
- polygons overlap



But is this really a problem?

But Is This *Really* A Problem?

Case study: Doom 3 engine (id Software)

- bump mapping
 - per-pixel surface shading
 - dynamic and projected lights
 - atmospheric effects
 - particle effects
 - shadow volumes
- } 50%
- } 50%

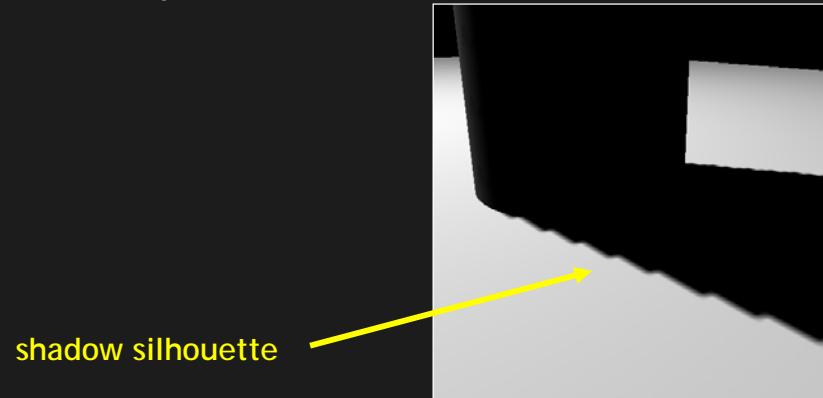
"Shadowing accounts for about half of the game's rendering time."

— John Carmack

Two Observations (**shadow maps**)

Shadow-map aliasing is ugly

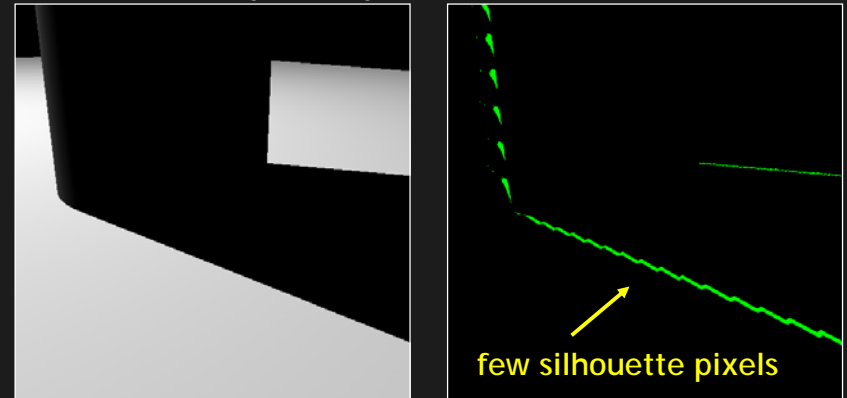
But — only noticeable at shadow silhouettes



Two Observations (**shadow volumes**)

Shadow volumes are accurate everywhere

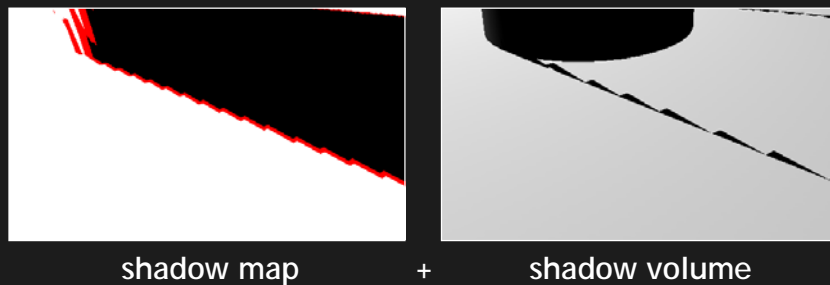
But — accuracy is only needed at silhouettes



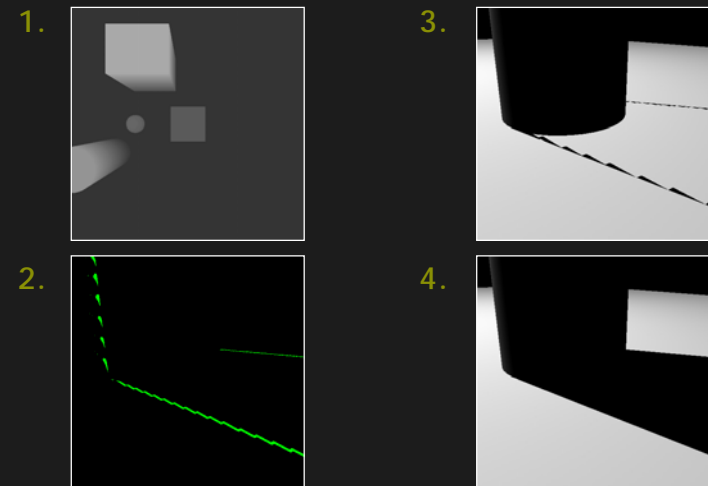
Hybrid Approach

Decompose the problem:

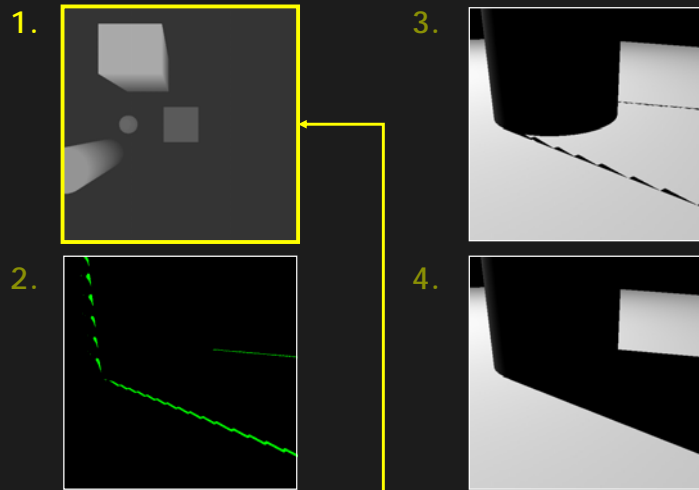
- use shadow volumes at silhouettes
- use shadow maps everywhere else



Algorithm

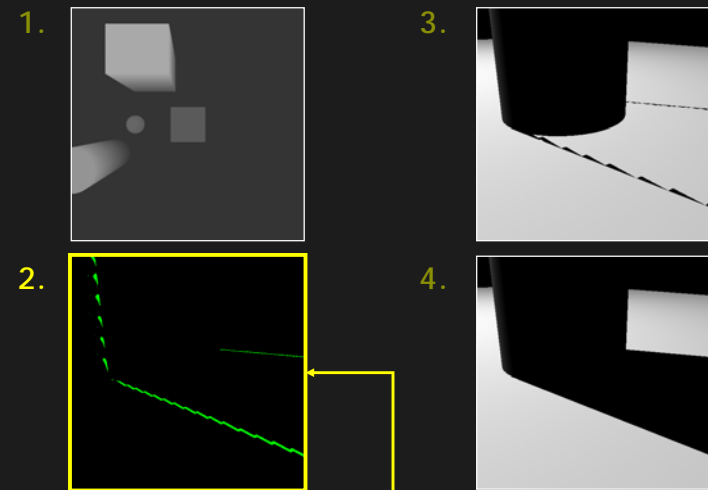


Algorithm



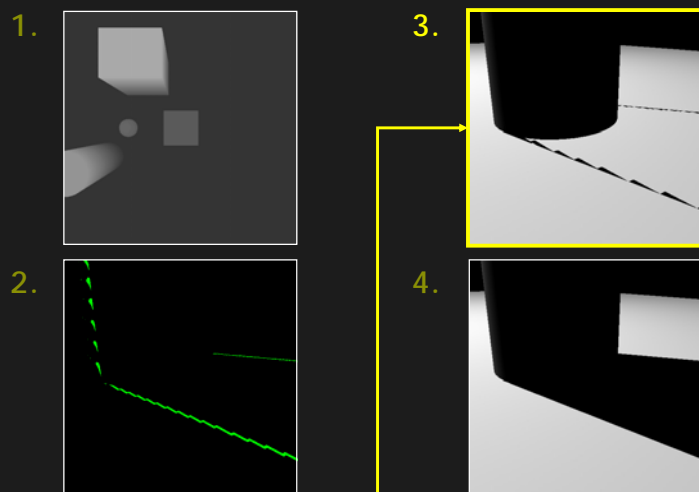
create a shadow map

Algorithm



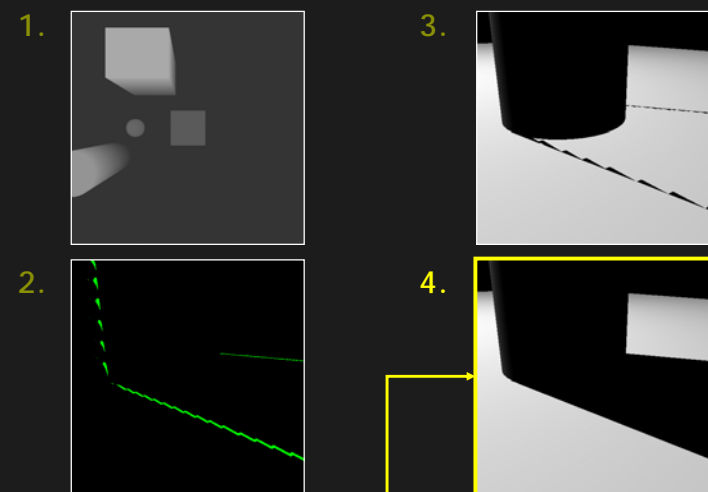
find silhouette pixels

Algorithm



apply shadow volumes only at silhouette pixels

Algorithm



apply shadow maps everywhere else

Test Scenes

