Shadow Algorithms for Real-time Rendering

Elmar Eisemann
Télécom ParisTech

Ulf Assarsson
Chalmers University

Michael Schwarz
MPI Informatik

Michael Wimmer
TU Vienna

Introduction

Elmar Eisemann

Basic Shadow Techniques

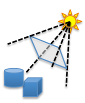
Shadow Maps and Shadow Volumes

Ulf Assarsson

Ways of thinking about shadows

- As volumes of space that are dark
  - Shadow Volumes [Franklin Crow 77]
- As places not seen by a light source looking at the scene
  - Shadow Maps [Lance Williams 78]

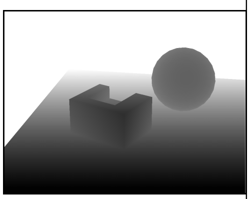
Shadow Maps



Basic Algorithm – the simple explanation:

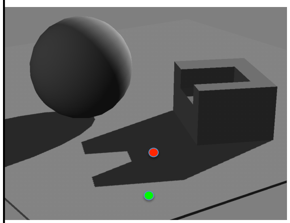
Idea:

- Render image from light source
  - Represents geometry in light
- Render from camera
  - Test if rendered point is visible in the light's view
    - If so -> point in light
    - Else -> point in shadow



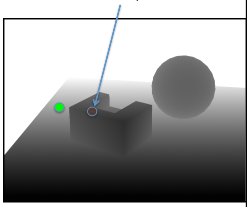
Shadow Map (light's view)

Shadow Maps



Camera's view

Point not represented in shadow map (point is behind box)

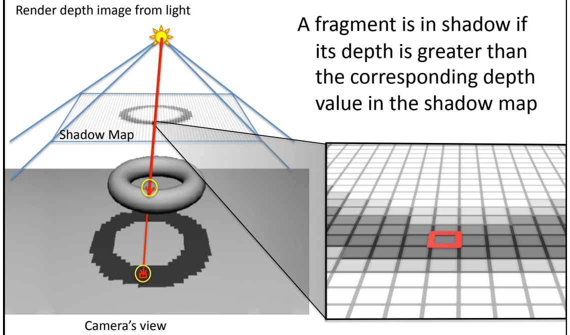


Light's view
(Shadow Map)

Depth Comparison

Render depth image from light

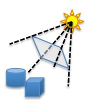
A fragment is in shadow if its depth is greater than the corresponding depth value in the shadow map



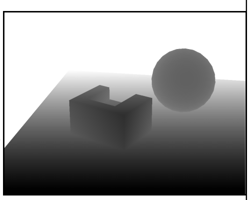
Camera's view

Shadow Maps

Shadow Map Algorithm:



- Render a z-buffer from the light source
  - Represents geometry in light
- Render from camera
  - For every fragment:
    - transform 3D-pos into shadow map (light space)
    - If depth greater-> point in shadow
    - Else -> point in light



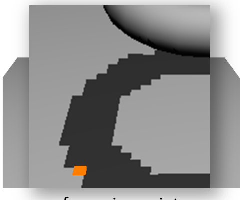
Shadow Map (=depth buffer)

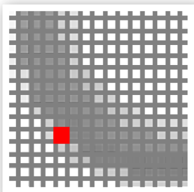
Shadow Maps

- Pros
  - Very efficient: “This is as fast as it gets”
- Cons…

Shadow Maps - Problems

- Low Shadow Map resolution results in jagged shadows


from viewpoint

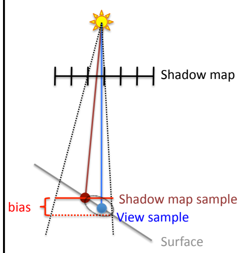

from light

Shadow Maps - Problems

In addition:

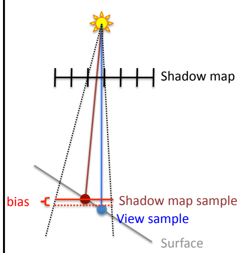
- A tolerance threshold (bias) needs to be tuned for each scene for the depth comparison

Bias

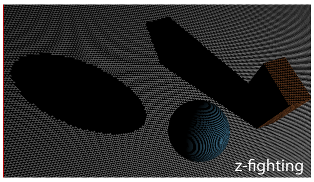


- Need a tolerance threshold (depth bias) when comparing depths to avoid surface self shadowing

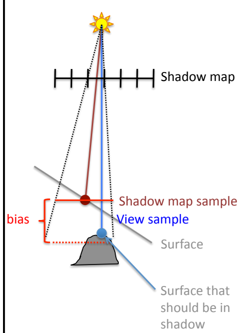
Bias



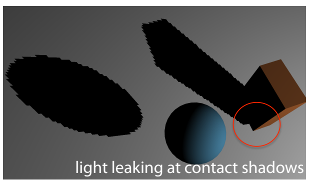
- Need a tolerance threshold (depth bias) when comparing depths to avoid surface self shadowing



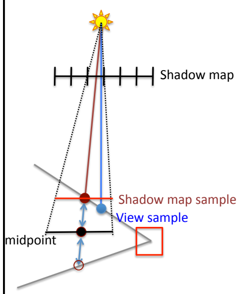
Bias



- Need a tolerance threshold (depth bias) when comparing depths to avoid surface self shadowing



Ameliorating the Bias

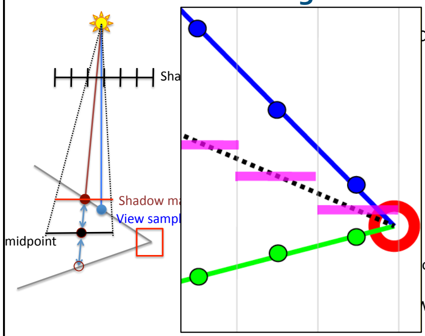


- Midpoint Shadow Maps [Woo 92]

Further methods:
- Second Depth Shadow Mapping [Wang and Molnar94]
- Dual Depth Layer [Weiskopf and Ertl 04]

Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Ameliorating the Bias



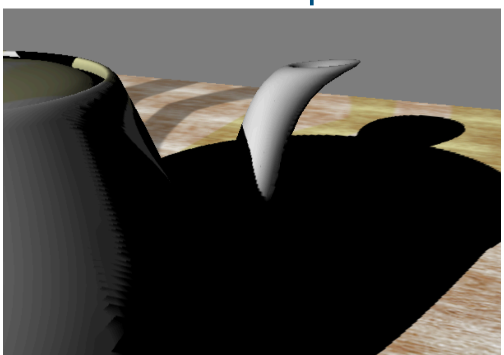
dow Maps

ow Mapping [Wang

Veiskopf and Ertl 04]

Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Shadow Maps

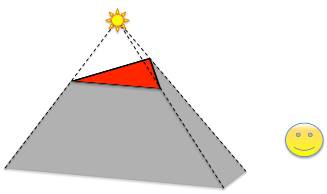


Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Shadow Volumes

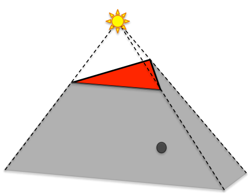
Shadow Volumes

- Concept
  - Create volumes of “space in shadow” from each triangle
    - Each triangle creates 3 quads that extends to infinity



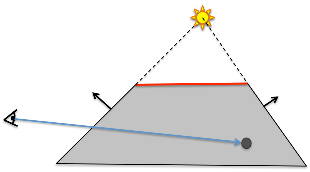
Shadow Volumes

- To test a point, count how many shadow volumes it is located within. More than one means point is in shadow



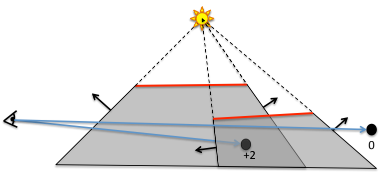
Shadow Volumes

- To test a point, count how many shadow volumes it is located within. More than one means point is in shadow



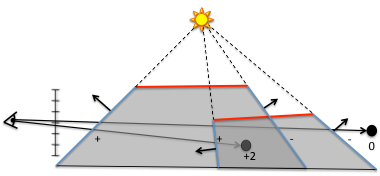
Shadow Volumes

- To test a point, count how many shadow volumes it is located within. More than one means point is in shadow



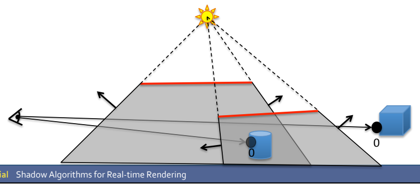
Shadow Volumes - concept

- A counter per pixel
- If we go through more frontfacing than backfacing polygons, then the point is in shadow



Shadow Volumes - concept

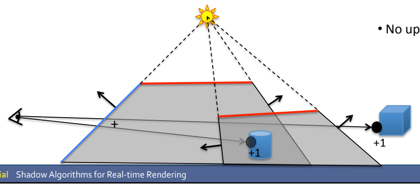
- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume



Shadow Volumes - concept

- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume

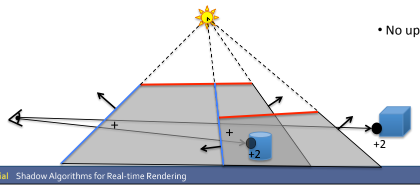
- No updating of z-buffer



Shadow Volumes - concept

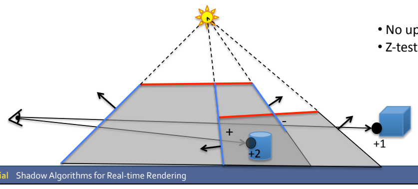
- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume

- No updating of z-buffer



Shadow Volumes - concept

- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume

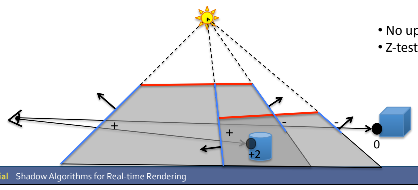


- No updating of z-buffer
- Z-test is enabled as usual

Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Shadow Volumes - concept

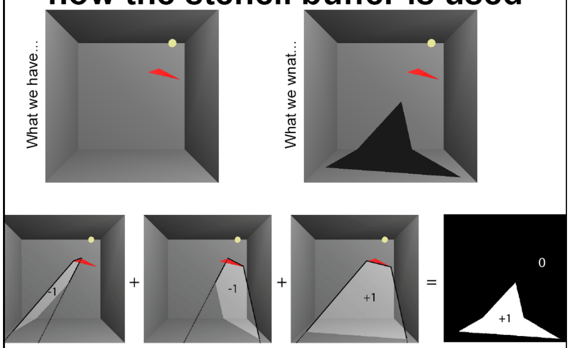
- Perform counting with the stencil buffer
  - Render front facing shadow quads to the stencil buffer
    - Inc stencil value, since those represents entering shadow volume
  - Render back facing shadow quads to the stencil buffer
    - Dec stencil value, since those represents exiting shadow volume



- No updating of z-buffer
- Z-test is enabled as usual

Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Z-pass by example: how the stencil buffer is used

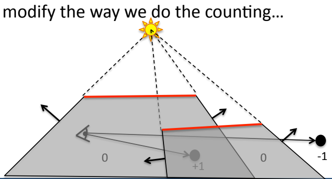


Shadow Volumes with the Stencil Buffer

- A four pass process [Heidmann91]
  - **1st pass:** Render *ambient* lighting
  - Use z-compare and draw to stencil buffer only
    - Turn off updating of z-buffer and writing to color buffer
  - **2nd pass:**
    - Render *frontfacing* shadow volume quads: *incrementing* stencil buffer count
  - **3rd pass:**
    - Render *backfacing* shadow volume quads: *decrementing* stencil buffer count
  - **4th pass:** Render *diffuse and specular* where stencil buffer is 0.

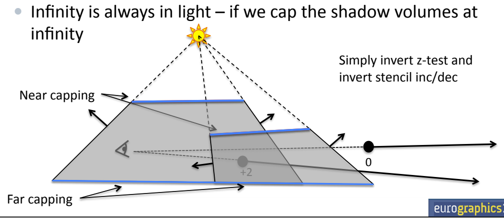
Eye Location Problem

- If the eye is located inside one or more shadow volumes, then the count will be wrong
- Solution:
  - Offset stencil buffer with the #shadow volumes that the eye is located within
  - Or modify the way we do the counting…

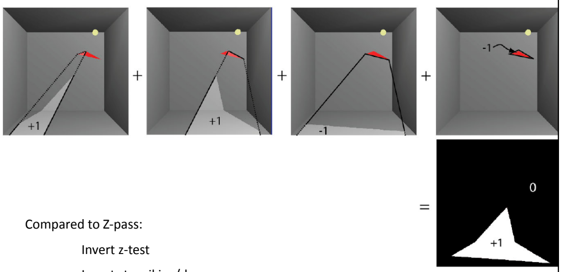


The Z-fail Algorithm

- By [Carmack00] and [Bilodeau and Songy 99]
  - "Carmacks Reverse"
- Count to infinity instead of to the eye
  - We can choose any reference location for the counting
  - A point in light avoids any offset
  - Infinity is always in light – if we cap the shadow volumes at infinity



Z-fail by example



Compared to Z-pass:

Invert z-test

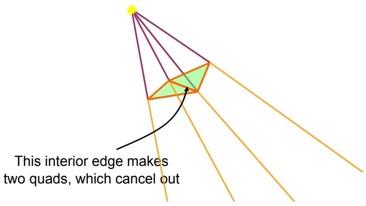
Invert stencil inc/dec

I.e., count to infinity instead of from eye.

Shadow Volumes from Silhouette Edges

Merging shadow volumes:

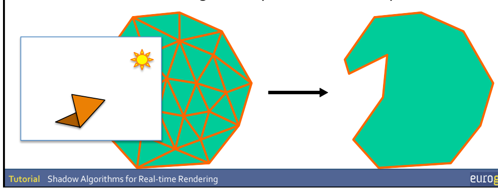
- An edge that is shared by two triangles facing the light creates two shadow quads that cancel each other out:



Shadow Volumes from Silhouette Edges

Merging shadow volumes:

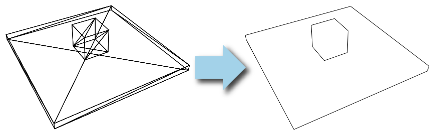
- An edge that is shared by two triangles facing the light creates two shadow quads that cancel each other out
- Thus, popular to create shadow volumes only from silhouette edges as seen from the light source
  - Avoids rendering of many useless shadow quads



Shadow Volumes from Silhouette Edges

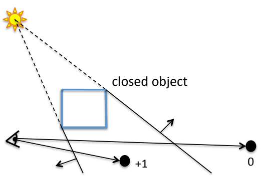
Merging shadow volumes:

- An edge that is shared by two triangles facing the light creates two shadow quads that cancel each other out
- Thus, create shadow volumes only from silhouette edges as seen from the light source
  - Avoids rendering of many useless shadow quads
  - A real example:



Shadow Volumes from Silhouette Edges

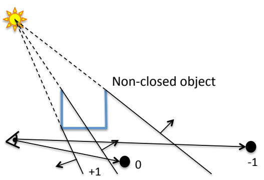
It is a misconception that objects needs to be closed



Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Shadow Volumes from Silhouette Edges

It is a misconception that objects needs to be closed



Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

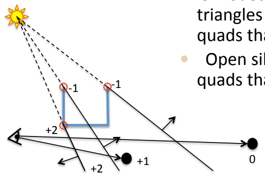
Shadow Volumes from Silhouette Edges

It is a misconception that objects needs to be closed

Fixed by [Bergeron 86]

Observation:

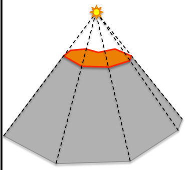
- Silhouette edges with two adjacent triangles should actually create shadow quads that inc/dec count by 2
- Open silhouette edges create shadow quads that inc/dec count by one



Works identically for Z-fail

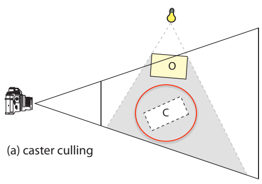
Shadow Volumes - Summary

- Pros:
  - High quality
- Cons:
  - OVERDRAW



Culling and Clamping

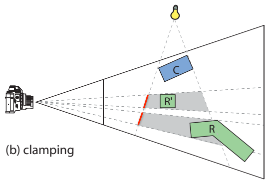
- **Culling of Shadow Volumes** [Lloyd et al. 2004][Stich et al. 2007]
  - **Culling of Shadow Casters** if it is located totally within shadow
    - Tested against a shadow depth map



(a) caster culling

Culling and Clamping

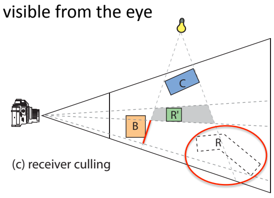
- **Clamping of Shadow Volumes** [Lloyd et al. 2004] [Eisemann and Decoret 2006]
  - Idea: Only render parts of shadow quads that affects a shadow receiver
    - Tested against AABB around shadow receivers



(b) clamping

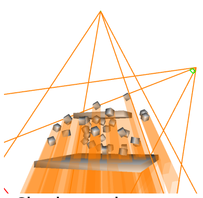
Culling and Clamping

- **Culling of Shadow Volumes** [Lloyd et al. 2004] [Eisemann and Decoret 2006]
  - **Receiver Culling**
    - Idea: Cull part of shadow volumes where shadow receivers are not visible from the eye

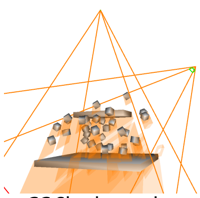


(c) receiver culling

Culling and Clamping



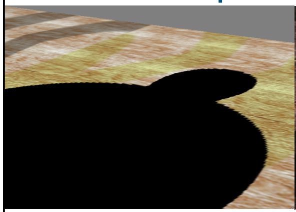
Shadow volumes



CC Shadow volumes

Illustrates reduced depth complexity when using Culling and Clamping

Shadow Maps vs Shadow Volumes



Shadow Maps

- *Good*: Handles any rasterizable geometry, **constant cost** regardless of complexity, map can sometimes be reused.
- *Bad*: Frustum limited. **Jagged shadows** if res too low, **biasing** headaches.

Shadow Volumes

- *Good*: shadows are **sharp**. Handles omni-directional lights.
- *Bad*: **3 or 4 passes**, shadow polygons must be generated and rendered → lots of polygons & **fill**,

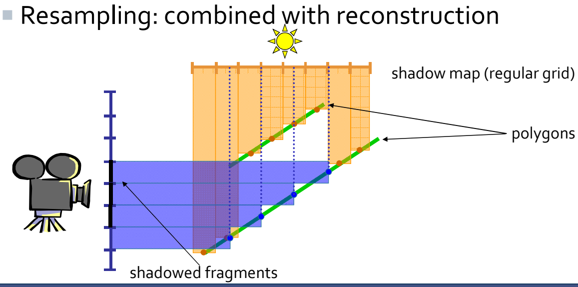
THE END

Hard Shadows

What is Shadow Map Aliasing

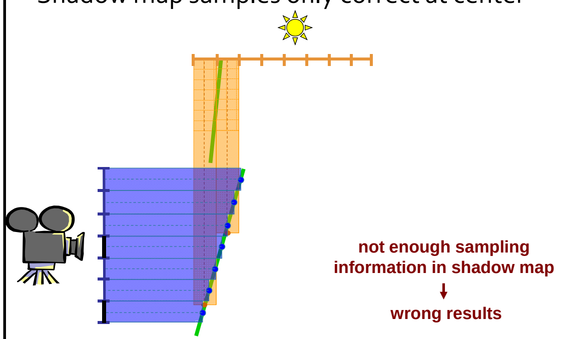
Shadow Map as Signal Reconstruction

- Initial sampling: shadow map rendering
- Reconstruction: nearest neighbor, PCF, …
- Resampling: combined with reconstruction



Shadow Map as Signal Reconstruction

- Shadow map samples only correct at center

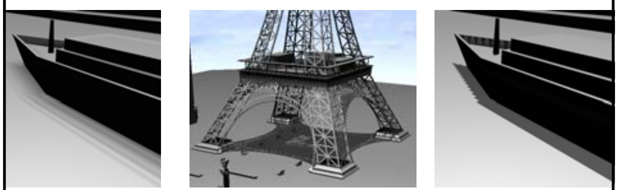


Main Types of Error

Undersampling
- Too low initial sampling frequency

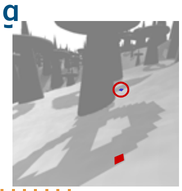
Oversampling
- No bandlimiting

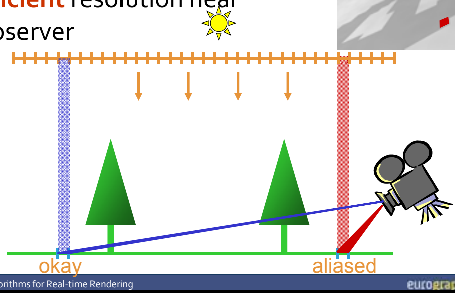
Reconstruction error
- Staircase artifacts



Perspective Aliasing

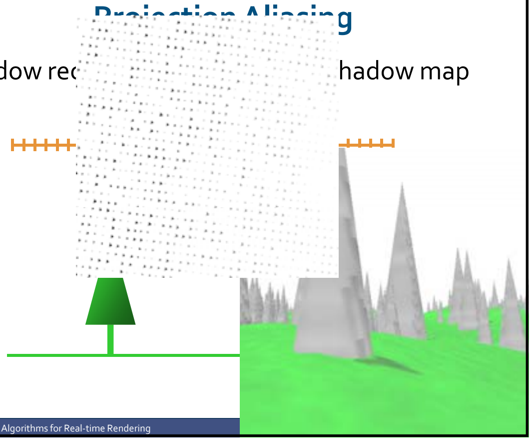
- **Sufficient** resolution far from the observer
- **Insufficient** resolution near the observer





Projection Aliasing

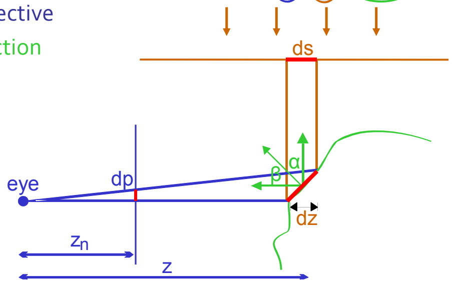
- Shadow rec… hadow map



Initial Sampling Error Analysis

- Aliasing error:
  - Parameterization
  - Perspective
  - Projection

$$\frac{dp}{ds} = \frac{z_n}{z}\frac{dz}{ds}\frac{\cos\alpha}{\cos\beta}$$



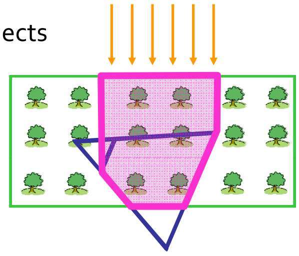
Hard Shadows

Fighting Undersampling - Fitting

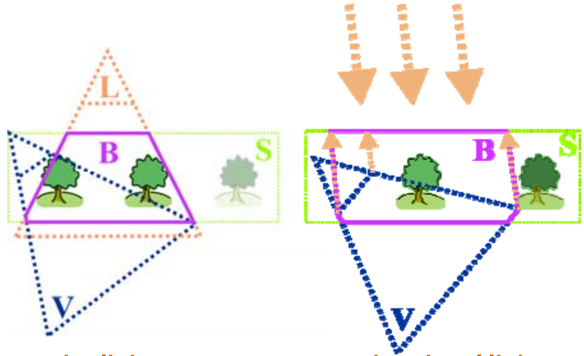
Fitting: Focus the Shadow Map

[Brabec et al. 2002]

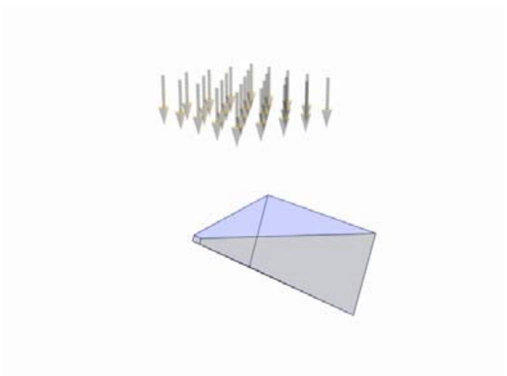
- Only include relevant objects
  - Shadow casters
  - Light source frustum
  - View frustum
- Better use of shadow map resolution



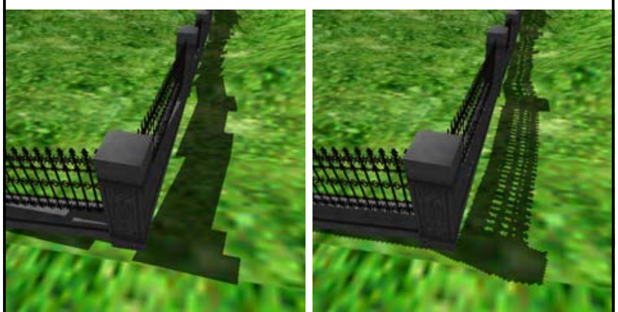
Fitting: Focus the Shadow Map



Fitting: Focus the Shadow Map

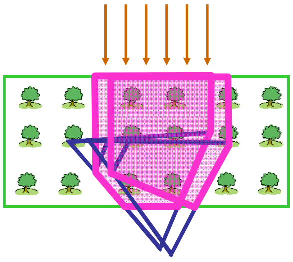


Fitting: Focus the Shadow Map

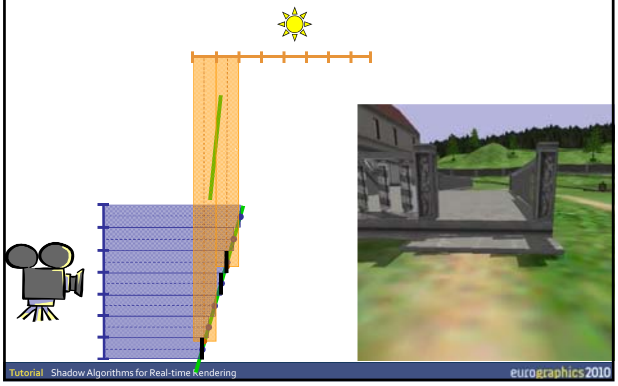


Unfocused Focused

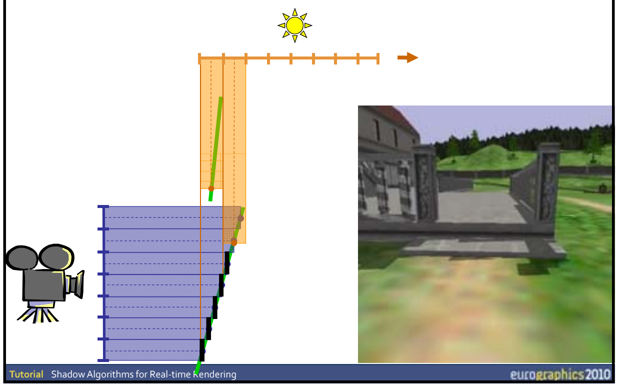
Fitting: Temporal Aliasing



Fitting: Temporal Aliasing



Fitting: Temporal Aliasing



Fitting: Temporal Aliasing

Fitting vs Temporal Aliasing

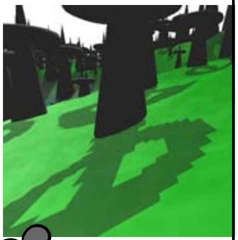
- Commonly used
- Better use of shadow map resolution
- One cause for temporal aliasing
- Temporal aliasing noticeable because of insufficient resolution → warping

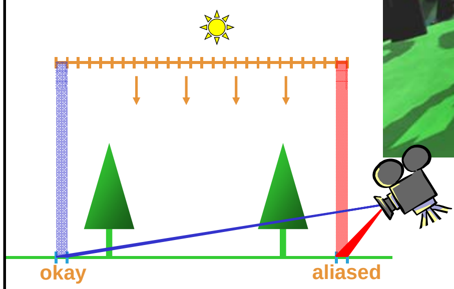
Hard Shadows

Fighting Undersampling - Warping

Solution for Perspective Aliasing

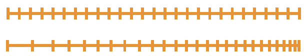
- **Insufficient** resolution near eye





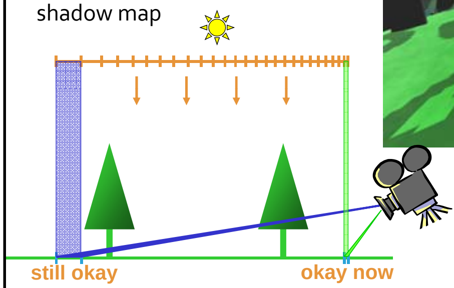
Solution for Perspective Aliasing

- **Insufficient** resolution near eye
- **Redistribute** values in shadow map



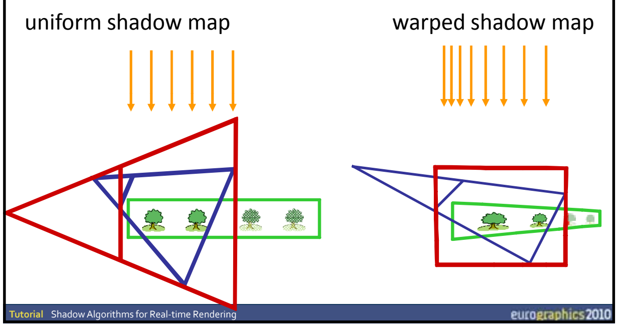
Solution for Perspective Aliasing

- **Sufficient** resolution near eye
- **Redistribute** values in shadow map



Shadow Map Warping

- Use an **additional perspective frustum**

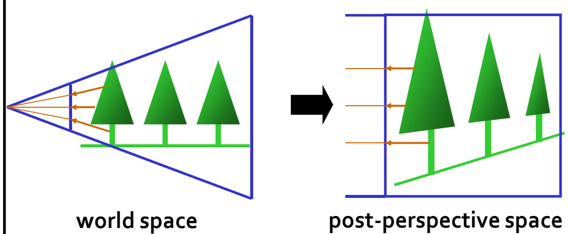


Shadow Map Warping

- Perspective Shadow Maps (PSM)
  [Stamminger & Drettakis 2002]
  - Warp frustum = view frustum
- Light Space Perspective Shadow Maps (LiSPSM)
  [Wimmer et al. 2004]
  - Warp frustum view plane orthogonal to light view plane
  - Optimal parameter choice
- Trapezoidal Shadow Maps (TSM)
  [Martin & Tan 2004]
  - Similar to LiSPSM
  - Parameter choice based on heuristic

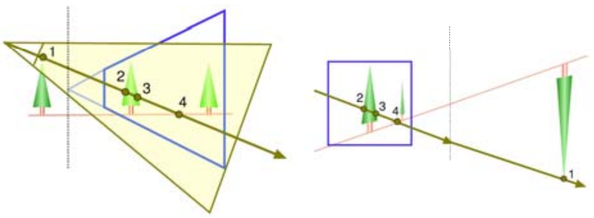
Perspective Shadow Maps
[Stamminger & Drettakis 2002]

- Want to account for perspective warping of eye
- Shadow map in post-perspective eye space
- Reduce perspective aliasing



PSM Problems

- Shadows from behind

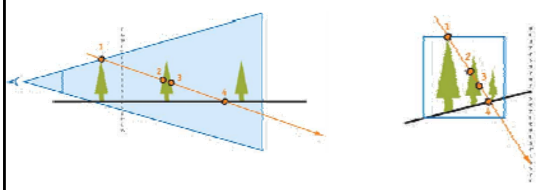


world space post-perspective space

Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

PSM Problems

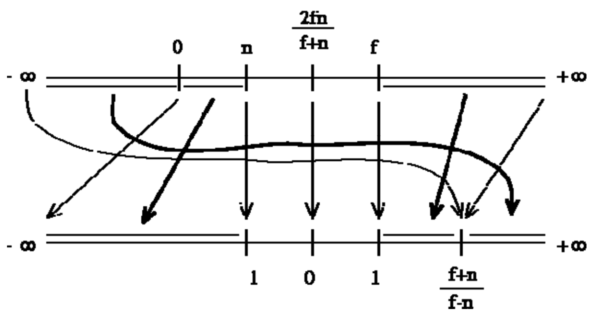
- Shadows from behind
  - Require move back due to singularity
  - Changes resulting quality of shadow



world space post-perspective space

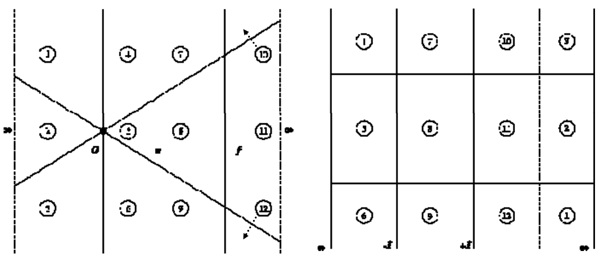
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Post Perspective Space - Depth



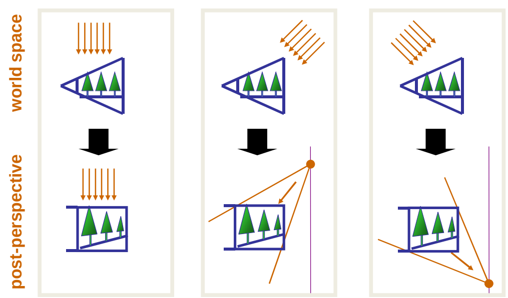
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Post Perspective Space – x or y



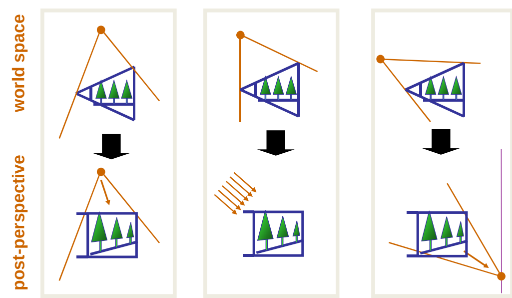
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Parallel Light Transformation



Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Point Light Transformation



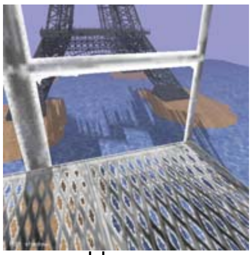
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

PSM Problems

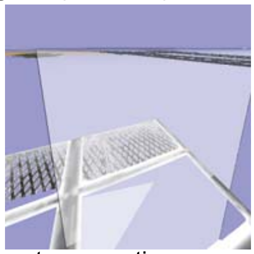
- Shadows from behind
  - Require move back due to singularity
  - Changes resulting quality of shadow
- Lights change their type (point/directional)
  - Many special cases
  - Non-intuitive post-perspective space
- View-dependent shadow quality

PSM Problems

- Most severe: uneven z-distribution
  - Good near viewer, very bad far away!
  - Can be reduced by pushing near plane away



world space



post-perspective space

Observation

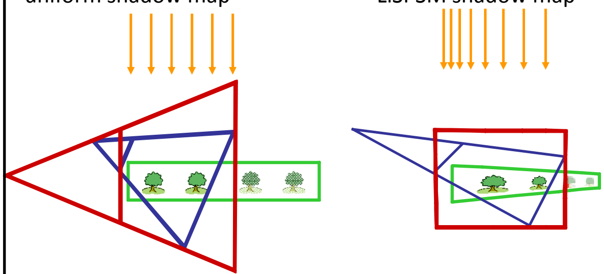
- Want to redistribute samples in shadow map
- Perspective transform is a good option
  - Supported by hardware
- BUT: why choose perspective based on observer transform???

Light Space Perspective Shadow Maps

- Use an **additional perspective frustum** in **light space**

uniform shadow map

LiSPSM shadow map



Light Space

light space

Advantages of LiSPSM

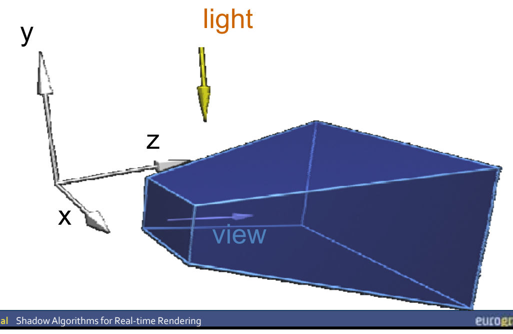
- No singularities in relevant part of the scene
  - ... by construction
- Directional lights remain directional lights
  - ... and direction remains the same
- Points lights are converted to directional lights
  - ... perspective point light transformation already done in light space
- Easy construction, general and robust
- Smooth quality changes
- Most important: **control over shadow quality**
  - ... by choice of free parameter

LiSPSM Overview

- Focus shadow map
- Construct light space
- Create perspective transform
- Apply

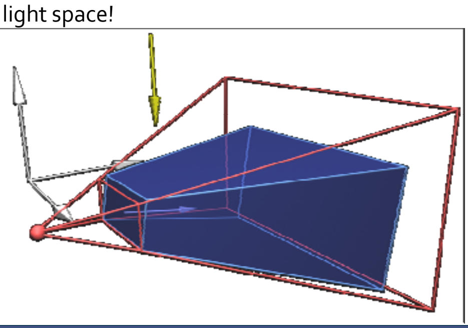
LiSPSM Fitting and Warp Construction

- Light and view vector define yz-plane

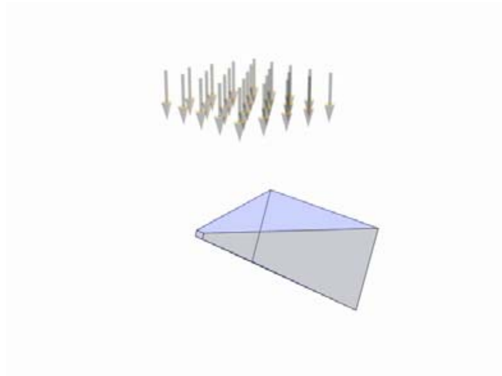


LiSPSM Fitting and Warp Construction

- Find a tight perspective frustum on focused region
  - In light space!

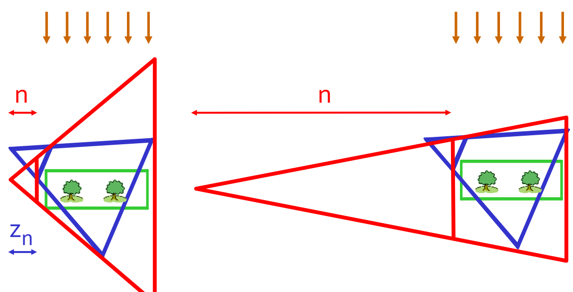


LiSPSM Construction



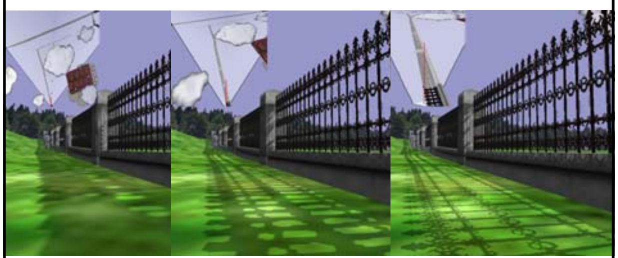
Free Parameter n

- Controls warping effect



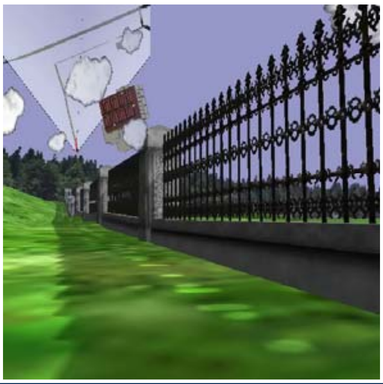
$n = z_n$: PSM   n → infinity : uniform shadow maps

Free Parameter n



very big n   very small n

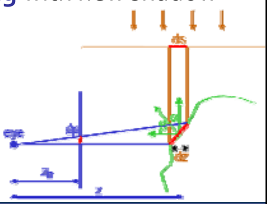
Free Parameter n



How to Choose the Free Parameter n?

- Recall error analysis
- $\frac{dp}{ds} > 1$ → shadow map undersampling
- Projection aliasing cannot be changed
- Counter perspective aliasing with new shadow map parameterization *s*
- Goal: $\frac{dp}{ds} \sim 1$

$$\frac{dp}{ds} = \frac{z_n}{z}\frac{dz}{ds}\frac{\cos\alpha}{\cos\beta}$$



Error Analysis

$$\frac{dp}{ds} = \frac{z_n}{z}\frac{dz}{ds}\frac{\cos\alpha}{\cos\beta}$$

- Perfect: logarithmic re-parameterization

$$s \sim \log z \Rightarrow \frac{ds}{dz} \sim \frac{1}{z} \Rightarrow \frac{dp}{ds} \sim 1$$

  - Hardware support? [Lloyd 2007, 2008]
- Uniform shadow maps

$$s \sim z \Rightarrow \frac{ds}{dz} \sim 1 \Rightarrow \frac{dp}{ds} \sim \frac{1}{z}$$

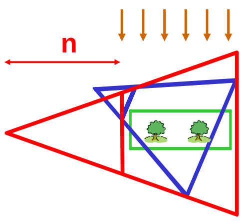
- Perspective shadow maps

$$s \sim \frac{1}{z} \Rightarrow \frac{ds}{dz} \sim \frac{1}{z^2} \Rightarrow \frac{dp}{ds} \sim z$$

  - Linear increase in error!

Error Analysis: LiSPSM Optimal Choice

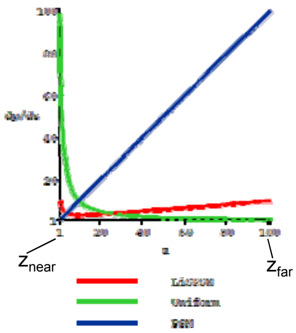
- For LiSPSM, $\frac{dp}{ds}$ depends on n
  - Gives $\frac{dp}{ds}$ between uniform and perspective
- Optimal choice:



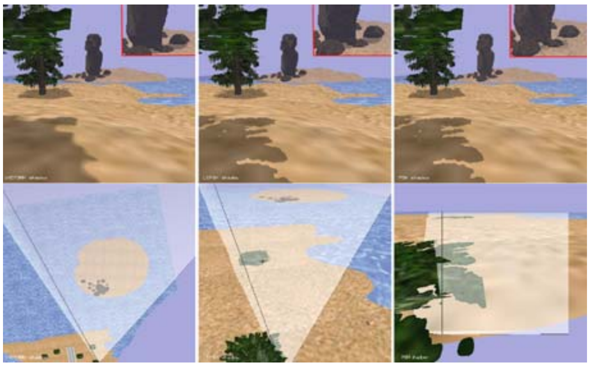
$$n_{opt} = z_n + \sqrt{z_f z_n}$$

Error Comparison

- LiSPSM optimal choice
- Measured along view dir
- LiSPSM vs PSM
  → for same depth range, LiSPSM error much lower
- More advanced analysis was done in [Lloyd 2006]

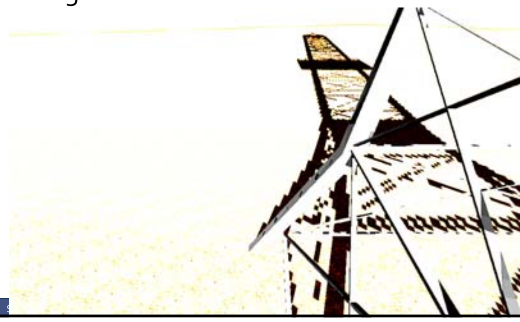


Comparison



Warping: Problems

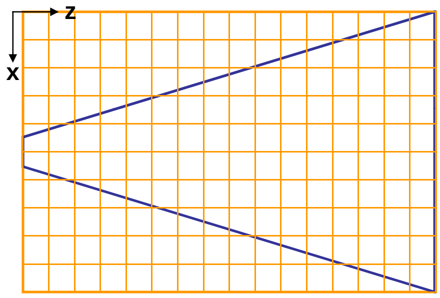
- Only works if large z-range visible from light
- Dueling frusta case



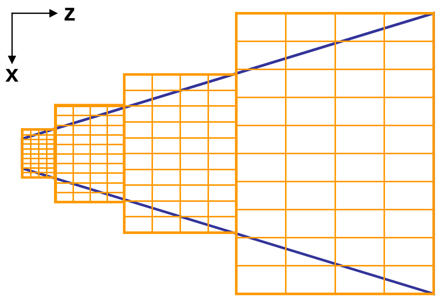
Hard Shadows

Fighting Undersampling - Partitioning

Z-Partitioning: Idea

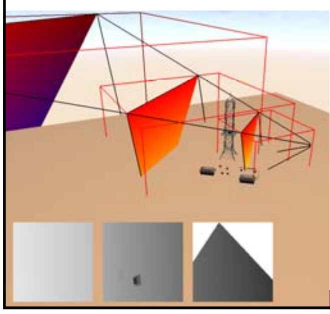


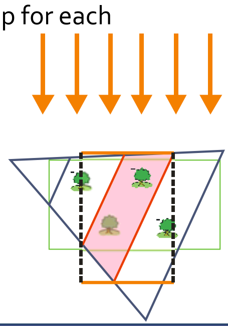
Z-Partitioning: Idea



Z-Partitioning

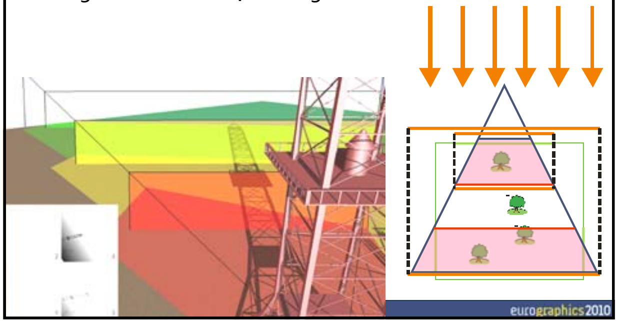
- Partition view frustum into n sub-frusta
- Calculate separate shadow map for each

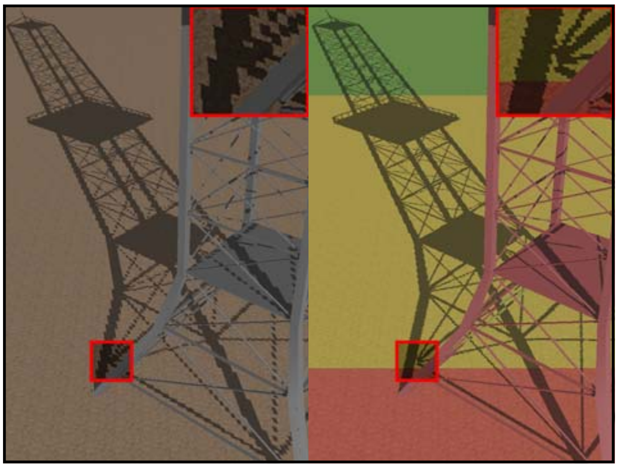




Z-Partitioning

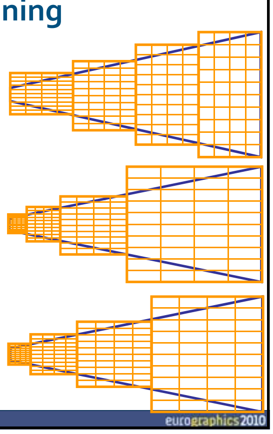
- Works even in cases where warping fails
  - Light from behind, dueling frusta



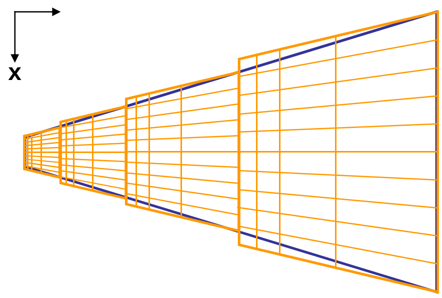


Z-Partitioning

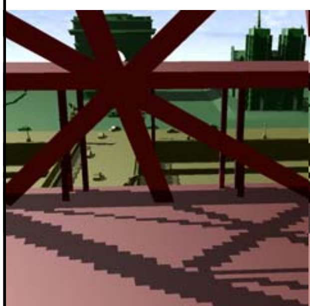
- How to choose partition sizes?
  - Uniform
  - Logarithmic
  - Linear blend between the two



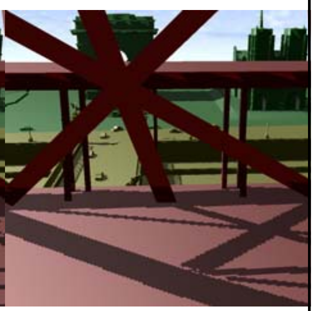
Z-Partitioning and Warping



Z-Partitioning and Warping



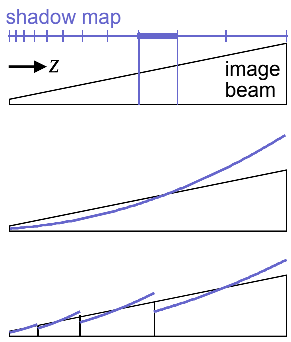
Partitioning



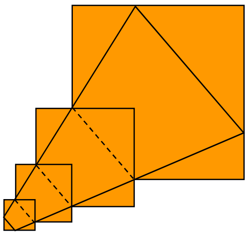
Partitioning + warping

Z-Partitioning

- Optimal shadow map texel spacing
- Spacing from a 4x4 projective transform
- Partitioning gives a better fit



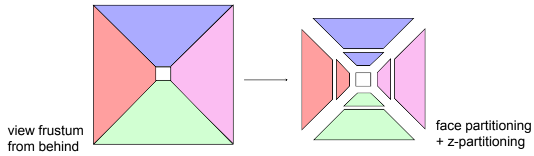
Temporal coherence



- Fix coordinate system in world space
- Shadow maps move at integral multiples of a texel width
- When view changes slowly, enlarged shadow maps permit caching
- Gives up warping

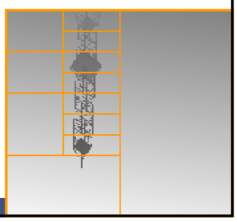
Face partitioning

- Problem: warping not good for all light directions
  - Especially from behind
- Solution: Partition frustum according to faces
- Can be combined with z-partitioning



Adaptive Partitioning

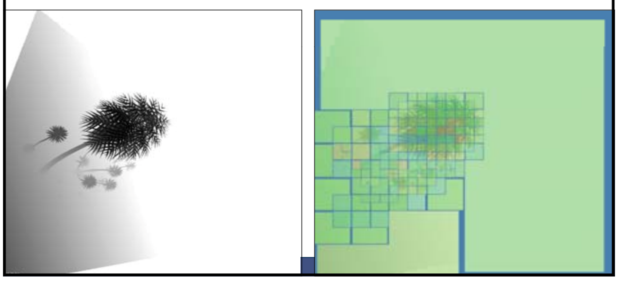
- Warping and z-partitioning are **global** resampling schemes
  - Deal with perspective aliasing
  - Projection aliasing needs local scene adaptive resampling!
- Adaptive partitioning adaptively splits shadow map
  - Usually quad-tree subdivision
- Algorithms mainly differ in termination criteria



Adaptive shadow maps

[Fernando et al. 2001; Lefohn et al. 2006]

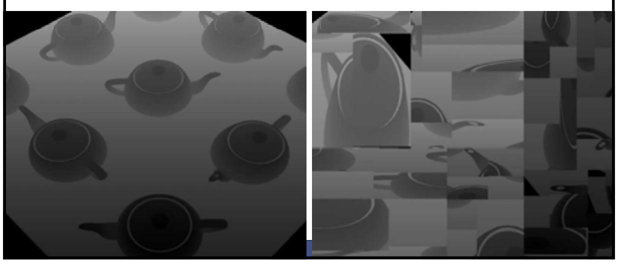
- High resolution only needed at edges
- Search for edge (slow)
  - If edge split



Tiled shadow maps

[Arvo 2004]

- Split according to heuristics
- Based on depth discontinuities, distances, ….
- Allows to trade speed against quality

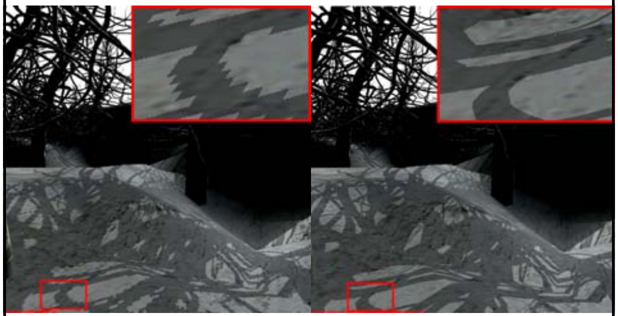


Fitted Virtual Shadow Maps

[Giegl & Wimmer 2007]

- Start with simple approach
- Render 4 shadow maps of a quad-tree node
  - Check if this was necessary (occlusion query)
- Do not calculate all quad-tree levels
- Determine finest levels necessary
- Pre pass, analysed on CPU
- Calculate only finest levels
- Avoids many passes

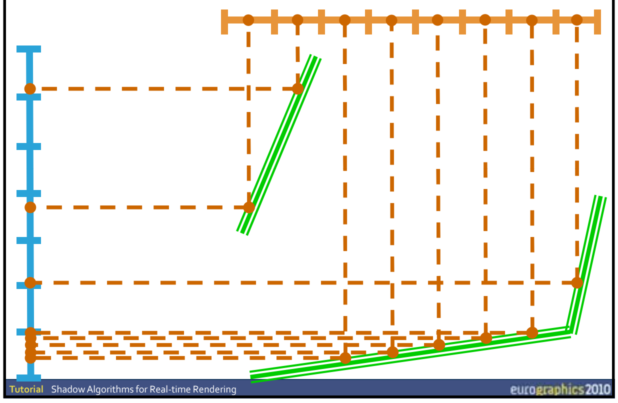
Fitted Virtual Shadow Maps
[Giegl & Wimmer 2007]



Hard Shadows

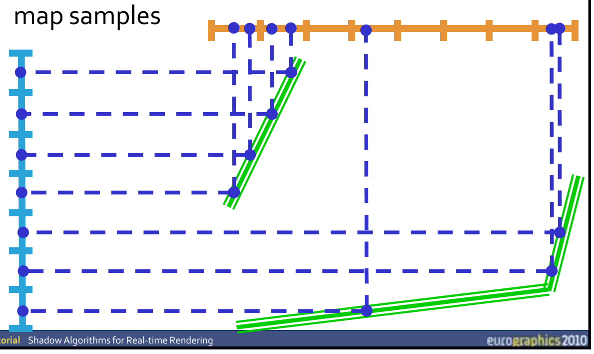
Fighting Undersampling - Irregular Sampling

Shadow Mapping Sampling



What Samples Do We Want?

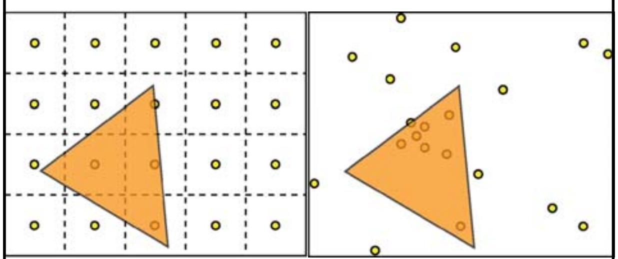
- Idea: use eye space samples to generate shadow map samples



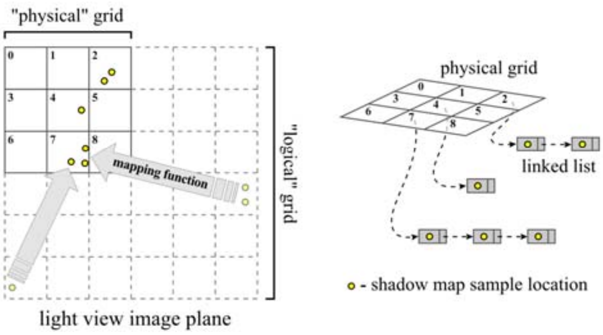
Regular vs Irregular Sampling

Irregular z-Buffer
[Johnson et al. 2004]

- Proposes hardware extension



Irregular z-Buffer



Alias Free Shadow Maps

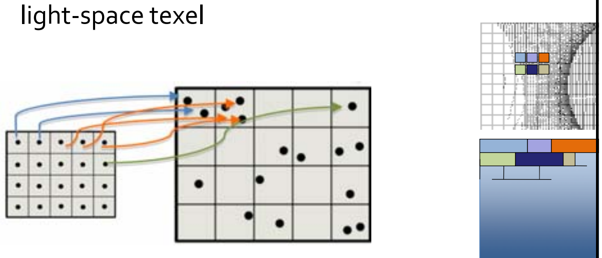
[Aila and Laine 2004]

- Transform and project view-samples into light-space
- Rasterize blocker geometry using them as sampling points
  - Test if sampling point covered
  - Hierarchical processing of sampling points (axis-aligned 2D BSP)
  - In software

Alias Free Hard Shadows

[Sintorn et al. 2008]

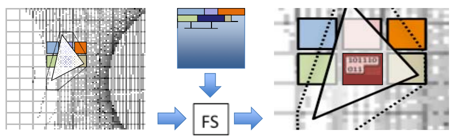
- Transform and project view-samples into light-space
- Store in a compact data structure with a list per light-space texel



Alias Free Hard Shadows

[Sintorn et al. 2008]

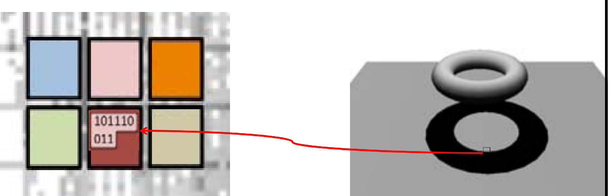
- Render all geometry (conservatively) from lights point of view
- For each generated fragment, test all view-samples in list against triangle
- Set corresponding output bit



Alias Free Hard Shadows

[Sintorn et al. 2008]

- Final screen-space pass
- Use bitmask from previous pass for shadowing



Hard Shadows

Fighting Undersampling - Conclusion

Conclusions

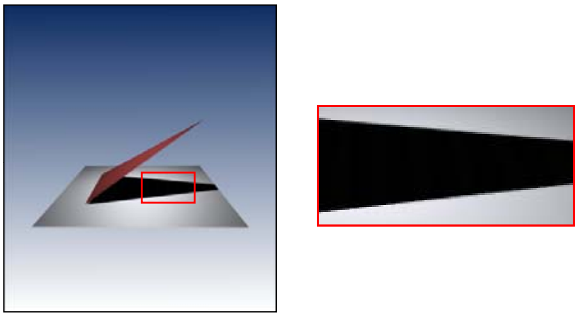
- Fastest speed, single shadow map: **warping**
  - Good for outdoor
- Fast speed, better quality, multiple shadow maps: **z-partitioning**
- High quality, lower speed: **adaptive partitioning**
- Reference quality, even slower: **irregular sampling**

Error Overview

- Undersampling
  - Improve initial sampling
- Oversampling
  - Use bandlimiting filters (Convolution Shadow Maps etc., see later)
- Reconstruction error
  - Use better reconstruction filters (PCF, PCF with Poisson disk sampling)
  - Use different reconstruction algorithm (Silhouette Shadow Maps)

Filtered Hard Shadows

Filtered Hard Shadows

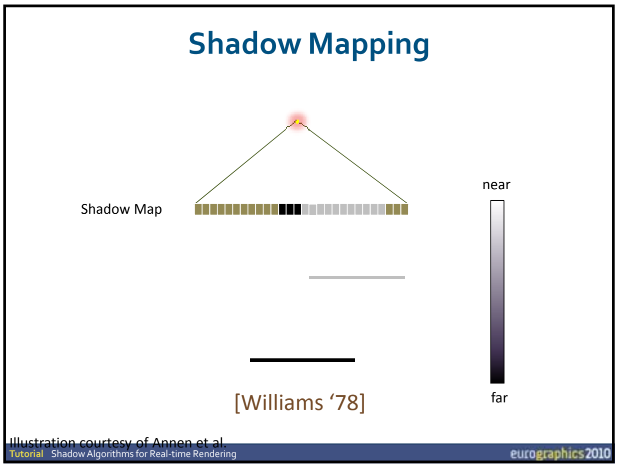


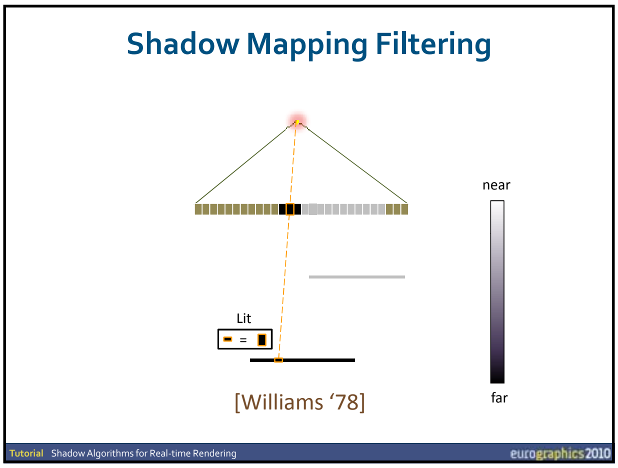
Filtered Shadow Mapping

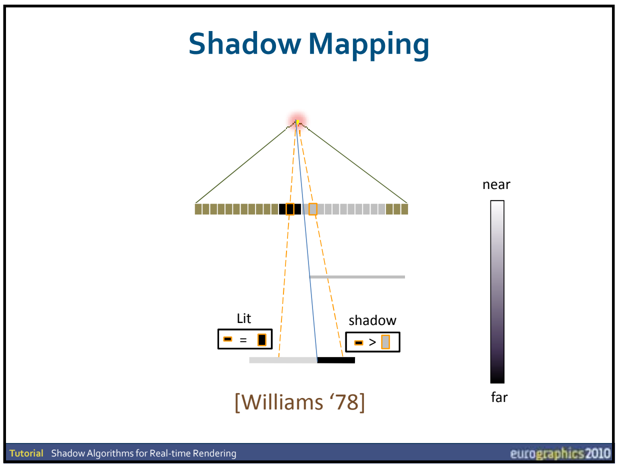
Shadow Mapping

[Williams '78]

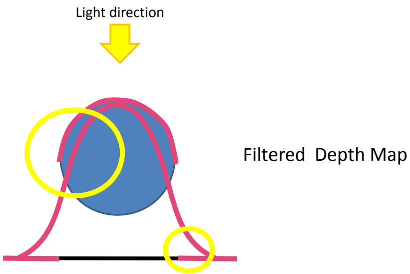
Illustration courtesy of Annen et al.


Shadow Mapping
Shadow Map
near
far
[Williams '78]
Illustration courtesy of Annen et al.
Tutorial Shadow Algorithms for Real-time Rendering
eurographics2010


Shadow Mapping Filtering
near
Lit
far
[Williams '78]
Tutorial Shadow Algorithms for Real-time Rendering
eurographics2010

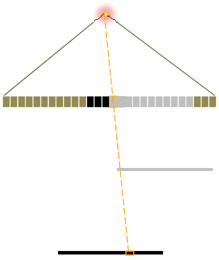

Shadow Mapping
near
Lit
shadow
far
[Williams '78]
Tutorial Shadow Algorithms for Real-time Rendering
eurographics2010

Filter Depth Map?



Percentage-Closer Filtering

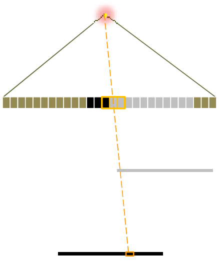

unfiltered



[Reeves et al. '87]

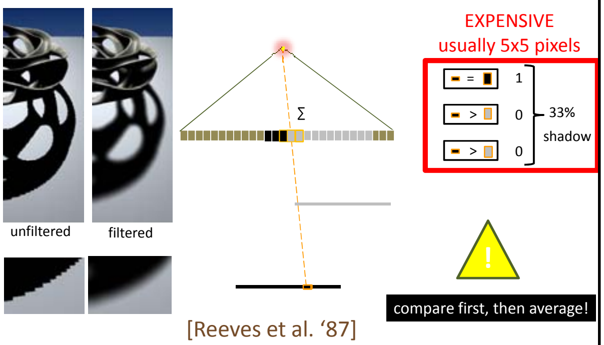
Percentage-Closer Filtering


unfiltered



[Reeves et al. '87]

Percentage-Closer Filtering



[Reeves et al. '87]

Percentage-Closer-like Filtering

+ Standard (simple to implement)
+ Hides discretization

- Costly for large kernels

ATTENTION NOT A SOFT SHADOW!

Problem 1:
How can we accelerate the process?

Problem 2: Perspective Foreshortening

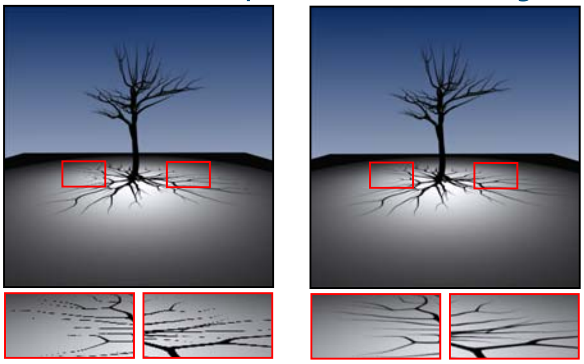


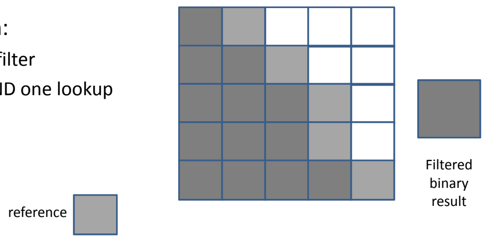
Illustration courtesy of Annen et al.

Improvements

- Variance Shadow maps [Lauritzen&Donnelly06]
- Convolution Shadow Maps [Annen et al. 07]
- Exponential Shadow Maps [Annen et al. 08, Salvi08]
- Layered Variance Shadow Maps [Lauritzen&McCool08]

Variance SMs [Lauritzen & Donnelly 2006]

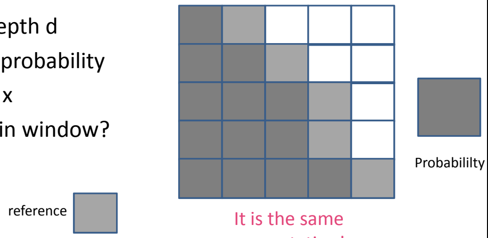
- Avoid many lookups and comparisons
- Solution:
  - FIRST filter
  - SECOND one lookup



Wait a second… didn't we just say:
We cannot first filter then compare???

Variance SMs [Lauritzen & Donnelly 2006]

- Reformulation:
- Given depth d
- What is probability that d < x for all x in window?



It is the same computation!

Idea: Use statistics to guess the outcome

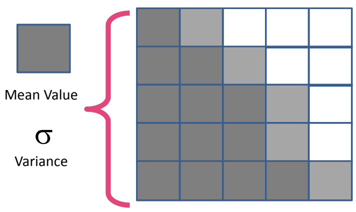
Depth Values become Distribution

- Instead of looking at all individual depth values: Consider a distribution function:

Use distributions mean and variance

Depth Values become Distribution

- Remember:



- Suppose a “Magic Transformation” for the moment

What can we do with Distribution?

- Given Mean μ and Variance σ for a window of depths
- How can we estimate the result?

- Chebyshev Inequality:

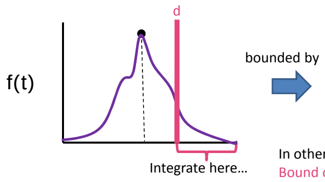
Probality(depth d < D) <= $\frac{\sigma^2}{\sigma^2 + (d-\mu)^2}$,

(for d<=μ, for d>=μ equals 1)

where D is random Variable from an arbitrary depth distribution (mean μ, variance σ)

What can we do with Distribution?

For depth values distribution the test result can be bounded



$$\frac{\sigma^2}{\sigma^2 + (d-\mu)^2}$$

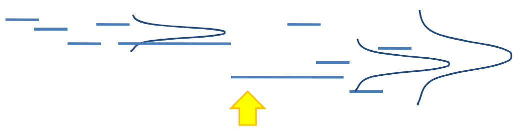
In other words:
Bound on percentage of depth values bigger than d (hence, those not occluding)

If one assumes equality, we have a fast method!

Estimating shadows

- Using: $\frac{\sigma^2}{\sigma^2 + (d-\mu)^2}$

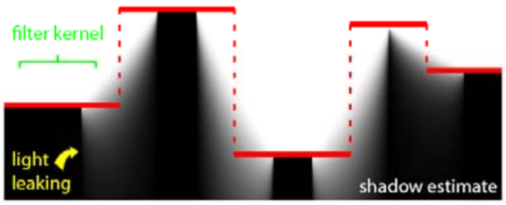
- Small sigma leads to quick fall-off
- Large sigma leads to slow fall-off



- Accurate for planar receiver + planar caster

Estimating shadows

- What is the problem?
- Solution is an upper bound, not accurate in general!



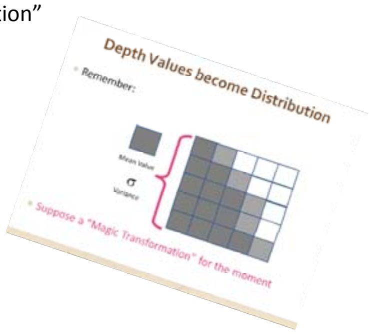
Variance SMs [Lauritzen & Donnelly 2006]

+ Very efficient
+ Independent of Filter size

- Light leaks

But there was still some detail…

- "Magic Transformation"



Transforming Depth Values to Distribution

- Mean Value: $\mu = E(d)$
  an average of depth values
- Variance: $\sigma^2 = E(d^2) - E(d)^2$
  easy to compute given two average values (depth² and depth)

➡ Need to quickly average values in a window

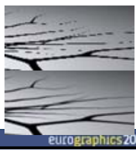
Computing Averages

- Corresponds to filtering:
  Fast Fourier Transformation Methods
- More GPU adapted solutions exist:
  Exploit window's rectangular shape

Sidenote: Computing Averages on the GPU

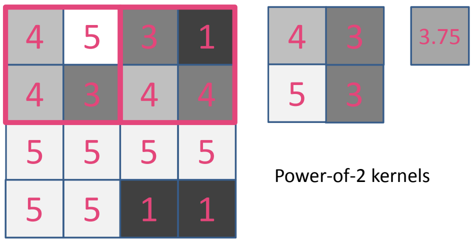
- Separable Convolution
- MipMapping
- N-Buffer
- Summed-Area Tables
- In this particular case:
  Most methods other than MipMap are overkill!
  We only need a small neighborhood of fixed size
  (separable convolution is good enough)

  MipMapping is interesting for perspective filtering



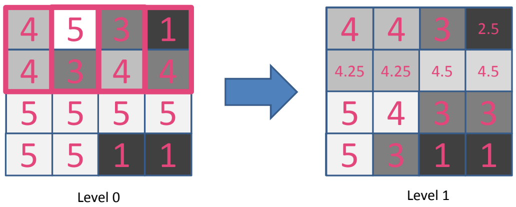
Computing Averages: MipMaps [Williams1983]

- Power of 2 hierarchy
- Average 4 pixels together for next level



Computing Averages: N-Buffers [Décoret2005]

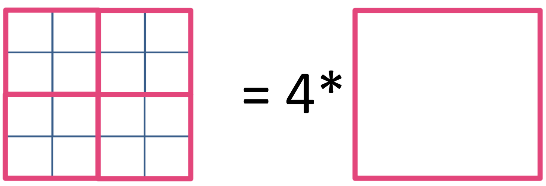
- Multi-Level Representation
- Each Level same resolution
- Level i: Pixel contains average of 2^i window
- Efficient recursive construction



Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Computing Averages: N-Buffers [Décoret2005]

- Efficient Recursive Construction:
  - Use level i for level i+1 (4 LU per texel)



Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Computing Averages: Fast Summed-Area Tables [Crow1984]

1	0	0	1
0	1	1	0
1	0	0	0
0	1	0	1

FSAT

		2	

Computing Averages: Fast Summed-Area Tables [Crow1984]

1	0	0	1
0	1	1	0
1	0	0	0
0	1	0	1

FSAT

		2	3

Computing Averages: Fast Summed-Area Tables [Crow1984]

1	0	0	1
0	1	1	0
1	0	0	0
0	1	0	1

FSAT

		2	3
			4

Computing Averages: Fast Summed-Area Tables [Crow1984]

1	0	0	1
0	1	1	0
1	0	0	0
0	1	0	1

FSAT

0	0	0	0
0	1	1	1
0	1	2	3
0	2	3	4

Computing Averages: Fast Summed-Area Tables [Crow1984]

Get average of any window with just 4 value lookups!

1	0	0	1
0	1	1	0
1	0	0	0
0	1	0	1

FSAT

0	0	0	0
0	1	1	1
0	1	2	3
0	2	3	4

4 -1 -2 +1 = 2

Computing Averages: Fast Summed-Area Tables [Crow1984]

Get average of any window with just 4 value lookups!

1	0	0	1
0	1	1	0
1	0	0	0
0	1	0	1

FSAT

0	0	0	0
0	1	1	1
0	1	2	3
0	2	3	4

4 -1 -2 +1 = 2

Fast Summed-Area Tables

- Efficient GPU Construction [Hensley et al.2005]

... but out of the scope of this presentation

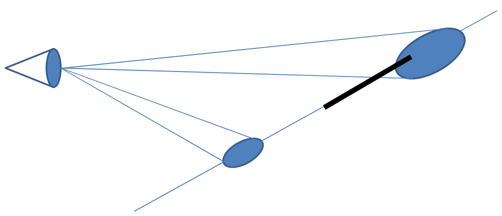
Filtering Quality

- MipMaps : fastest, but lowest quality
- N-Buffer : average construction time, medium quality
  (higher memory consumption)
- FSAT: slower construction time, highest quality
  (slightly more costly lookups)

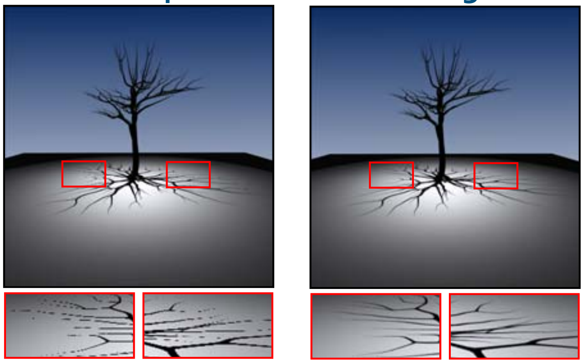
- These methods are especially useful for adaptive filtering (e.g., varying filter size)

Perspective Foreshortening

- Choose kernel size according to viewing distance



Perspective foreshortening



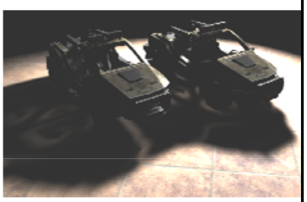
Applied to Variance Shadow Maps

- All techniques compatible with Variance Shadow Maps:

1. Create Shadow Map and Squared Shadow Map (d, d^2)
2. Create filtered maps $(E(d), E(d^2))$
3. Use maps for shadow computation $(\mu = E(d), \sigma^2 = E(d^2) - E(d)^2$

Distribution Assumption

- Bound is
  - Very approximate
  - leads to light leaks



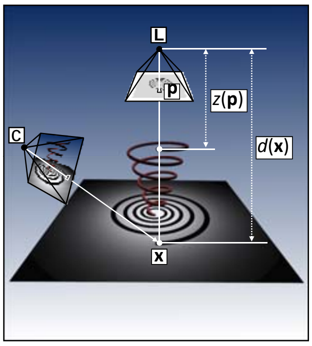
partially addressed in [Lauritzen&McCool 2008] via layered-variance SM, but still similar approach

- Are there other ways?

Approximate the Shadow Function

- Convolution Shadow Maps [Annen et al. 2007]
- Exponential Shadow Maps [Annen et al. 2008, Salvi 2008]

Shadow Mapping [Williams 1978]



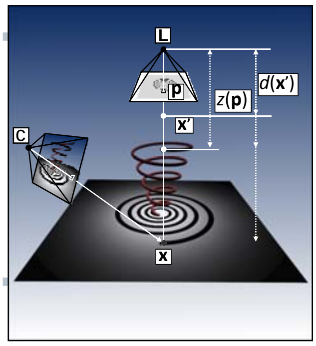
- $\mathbf{x} \in \mathbb{R}^3$
- $\mathbf{p} \in \mathbb{R}^2$

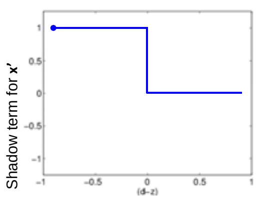
Shadow Test:
$f(d(\mathbf{x}),z(\mathbf{p})) := s(d(\mathbf{x})-z(\mathbf{p}))$,
where

$$s(\mathbf{x}) := \begin{cases} 1 \text{ if } \mathbf{x} <= \mathbf{0} \\ 0 \text{ else} \end{cases}$$

Animation courtesy of Annen et al.

Shadow Test: $s(\mathbf{x})$

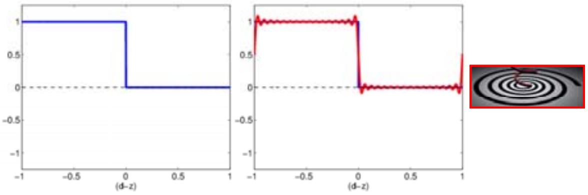


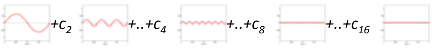


Animation courtesy of Annen et al.

Fourier Analysis

- Approximate shadow test with Fourier series



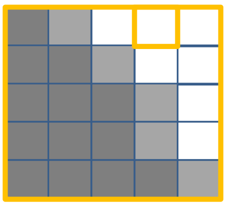


$s(t) \approx c_1 \ldots + c_2 \ldots + .. + c_4 \ldots + .. + c_8 \ldots + .. + c_{16} \ldots$

Fourier Analysis

- Why is this useful???

$$\sum_{s=1}^{N} f(d,z_s) = \sum_{s=1}^{N} \sum_{i=1}^{\infty} a_i(d)\, B_i(z_s)$$

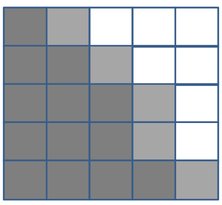


Fourier Analysis

- Why is this useful???

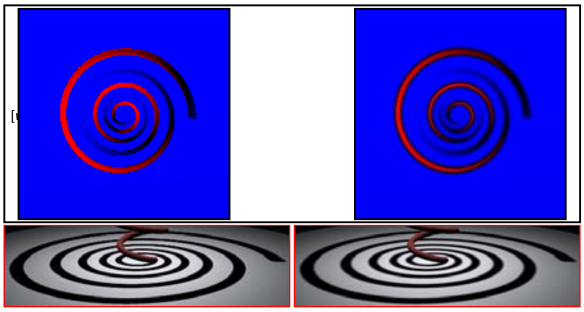
Only depends on depths in SM

$$\sum_{s=1}^{N} f(d,z_s) = \sum_{i=1}^{\infty} a_i(d) \sum_{s=1}^{N} B_i(z_s)$$



Filtering Example

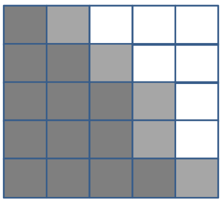
Original $B_i(z)$ | After filtering $B_i(z)$



Reality strikes back…

- Why is this useful???

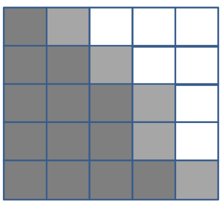
$$\sum_{s=1}^{N} f(d,z_s) = \sum_{i=1}^{\infty} a_i(d) \sum_{s=1}^{N} B_i(z_s)$$



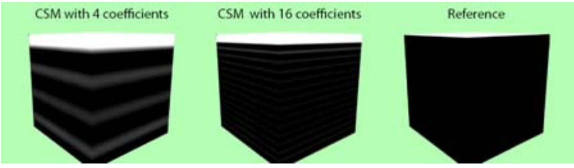
Reality strikes back…

- Why is this useful???

$$\sum_{s=1}^{N} f(d,z_s) \approx \sum_{i=1}^{M} a_i(d) \sum_{s=1}^{N} B_i(z_s)$$

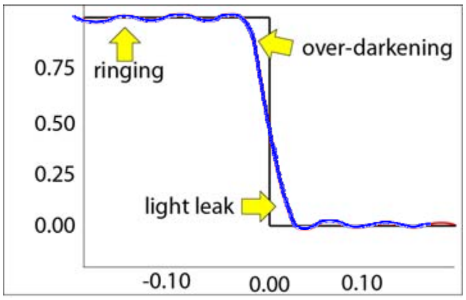


Truncation can lead to ringing



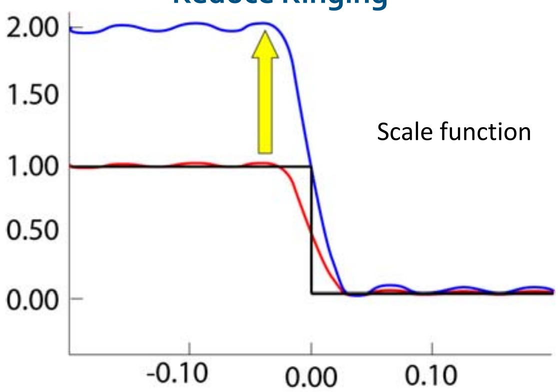
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Reduce Ringing



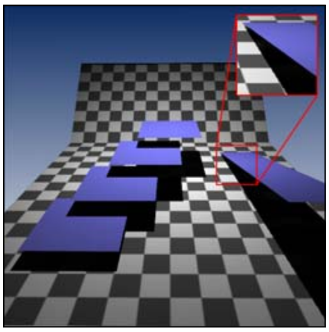
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Reduce Ringing



Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

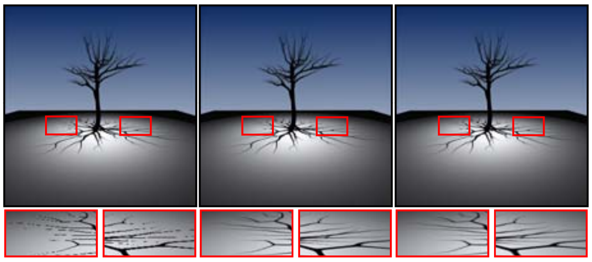
Influence of reconstruction order M



M = 816

- Memory consumption increases as M grows
- Performance (filtering) decreases as M grows

Results



PCF (NVIDIA) CSM CSM – 7x7 Gauss

CSM [Annen et al. 2007]

+ Beautiful theoretical result
+ high-quality smoothing (includes anisotropic blur)

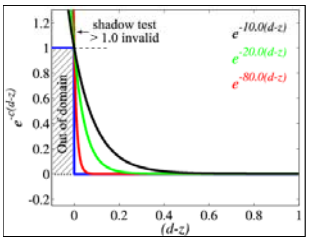
~ relatively fast (M=8, ~50fps on GTX8800)

- Relatively high memory consumption

Exponential SM [Annen et al. 2008, Salvi2008]

- Single SM via different decomposition:

$$s(\mathbf{x}) = f(d(\mathbf{x}), z(\mathbf{p})) = e^{-c(d(\mathbf{x}) - z(\mathbf{p}))}$$

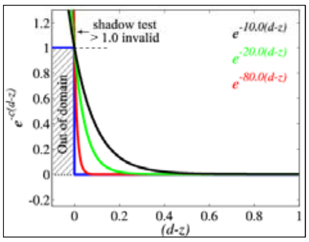


Exponential SM [Annen et al. 2008, Salvi2008]

- Single SM via different decomposition:

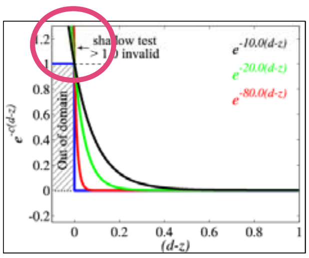
decomposition

$$s(\mathbf{x}) = f(d(\mathbf{x}), z(\mathbf{p})) = e^{-c(d(\mathbf{x}) - z(\mathbf{p}))} = e^{-c(d(\mathbf{x}))}e^{cz(\mathbf{p})}$$



Problem: Overshooting

- d needs to be >= z !



Detection of Erroneous Pixels

- Detection necessary and
  those pixels need PCF computation

ESM [Annen et al. 2008, Salvi2008]

+ Faster than CSM some scenes (depends on correction)
+ Same quality as CSM
+ Memory consumption lower than CSM
+ Creation of ESM is faster

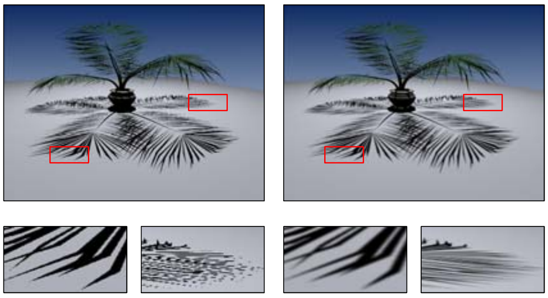
- Correction cost unpredictable
- Should be used with smaller kernels

Extensions

- A modification was suggested in
  [Lauritzen and McCool 2008]

  Basic idea:
  two exponentials
  select the more appropriate (smaller) solution

Results



PCF with 2x2 kernel

Blurred shadows

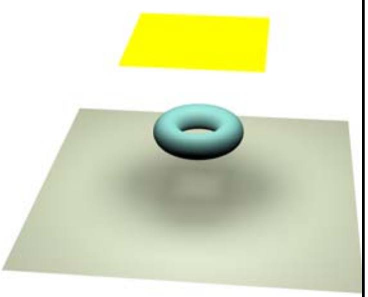
Overview

- Really fast:
  Variance SM
- Higher Quality:
  Exponential Shadow Maps
- CSM is involved in a soft shadow algorithm

BUT all these methods filter hard shadows.
There is no real penumbra computation…

Outlook

- We are not here yet:



… but we will get there… after the break!

Shadow Algorithms for Real-time Rendering

Elmar Eisemann
Télécom ParisTech

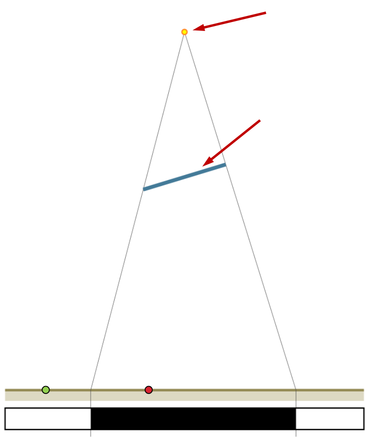
Ulf Assarsson
Chalmers University

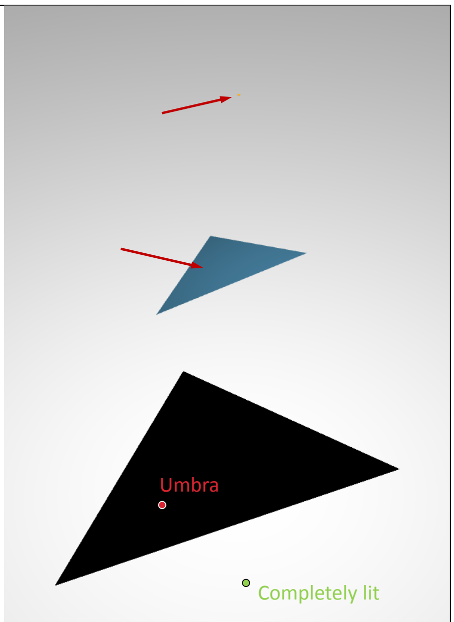
Michael Schwarz
MPI Informatik

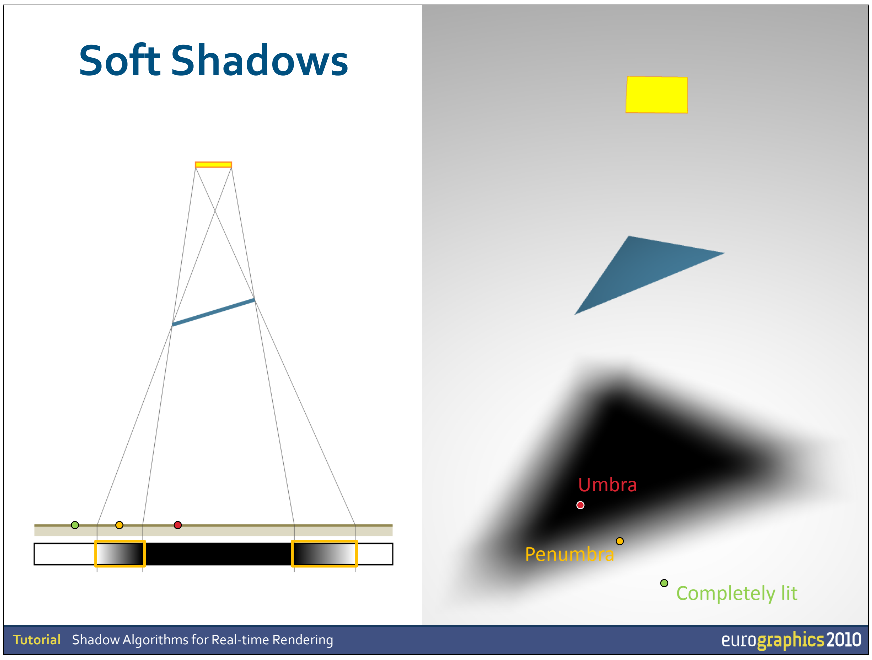
Michael Wimmer
TU Vienna

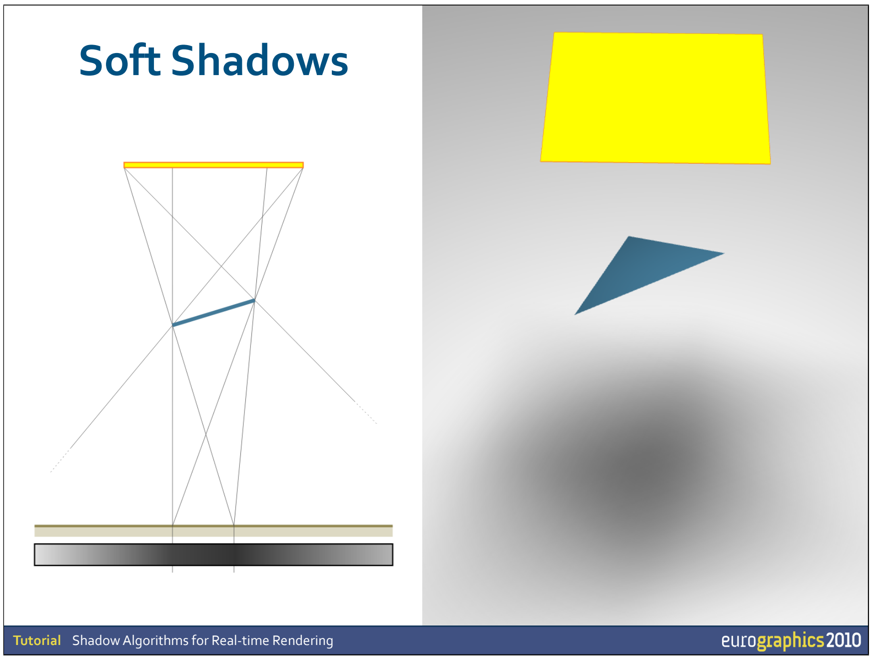
Soft Shadows

Hard Shadows





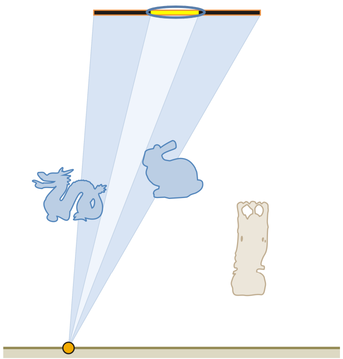

Soft Shadows
Umbra
Penumbra
Completely lit
Tutorial Shadow Algorithms for Real-time Rendering
eurographics2010


Soft Shadows
Tutorial Shadow Algorithms for Real-time Rendering
eurographics2010

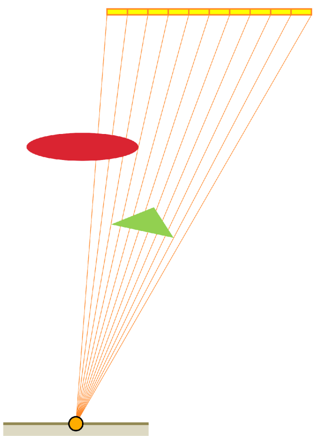

Shadow Hardening on Contact
Tutorial Shadow Algorithms for Real-time Rendering
eurographics2010

Shadowing = Point–Region Visibility

Task: Determine visible fraction of light source (region)
for each receiver sample (point)



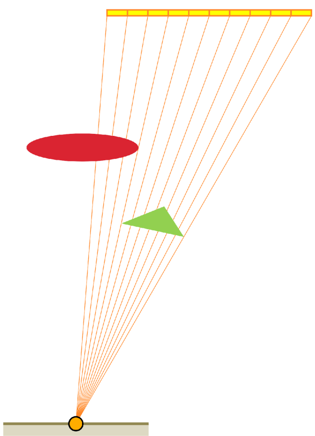
Occluder Fusion



- Occlusion O_1: 30%
- Occlusion O_2: 70%
- Total occlusion: 90%

- $\sum_i O_i$: 100%
- $\max_i O_i$: 70%
- $1 - \prod_i (1 - O_i)$: 79%

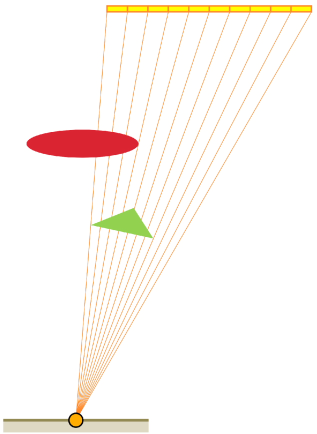
Occluder Fusion



- Occlusion O_1: 30%
- Occlusion O_2: 70%
- Total occlusion: 100%

- $\sum_i O_i$: 100%
- $\max_i O_i$: 70%
- $1 - \prod_i (1 - O_i)$: 79%

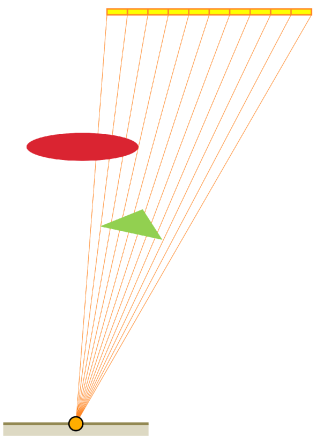
Occluder Fusion



- Occlusion O_1: 30%
- Occlusion O_2: 70%
- Total occlusion: 70%

- $\sum_i O_i$: 100%
- $\max_i O_i$: 70%
- $1 - \prod_i (1 - O_i)$: 79%

Occluder Fusion



- Occlusion O_1: 30%
- Occlusion O_2: 70%
- Total occlusion: 79%

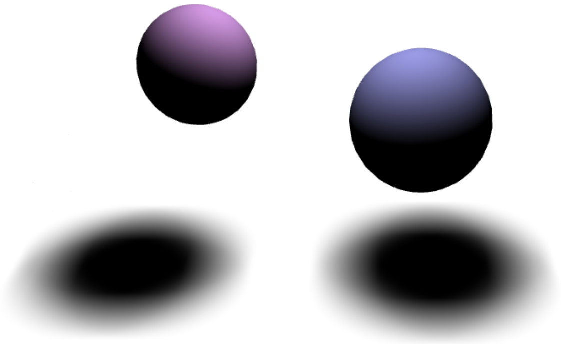
- $\sum_i O_i$: 100%
- $\max_i O_i$: 70%
- $1 - \prod_i (1 - O_i)$: 79%

Soft Shadows

Image-based Approaches

Blurring of Hard Shadow Test Results

- Use any of the approaches presented before
- Yields a soft-shadow-like appearance

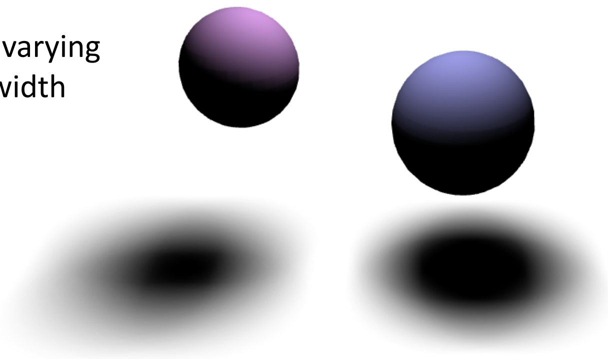


VSM, 512×512, 62×62

Blurring of Hard Shadow Test Results

- Use any of the approaches presented before
- Yields a soft-shadow-like appearance

- But ignores varying penumbra width

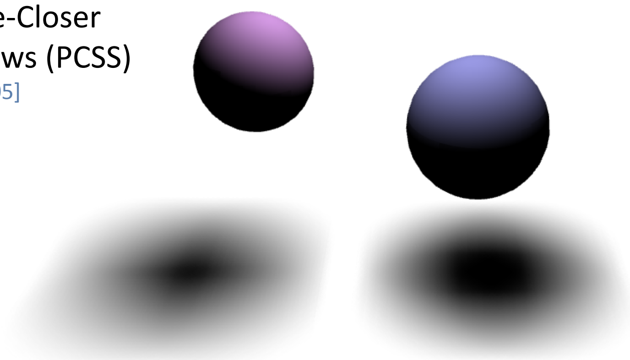


Reference

Blurring of Hard Shadow Test Results

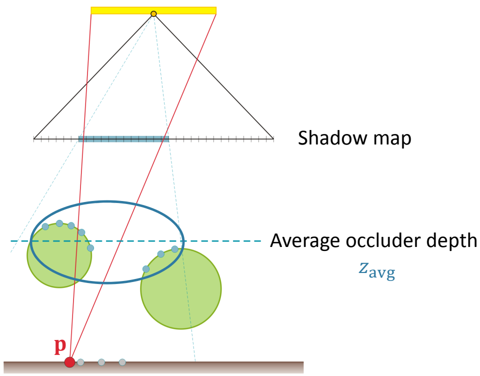
- Idea: Choose blur kernel size adaptively
  - But how?

- Percentage-Closer Soft Shadows (PCSS) [Fernando, 2005]



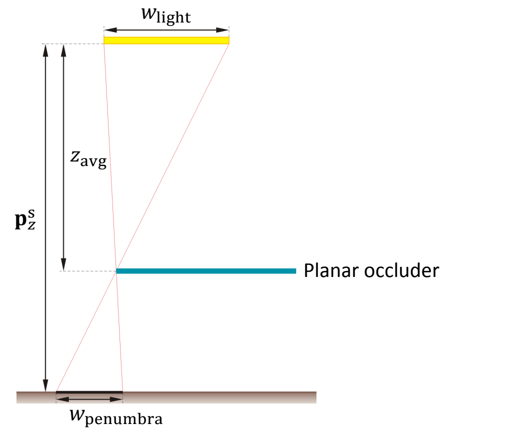
PCSS

Percentage-Closer Soft Shadows



1. Blocker search

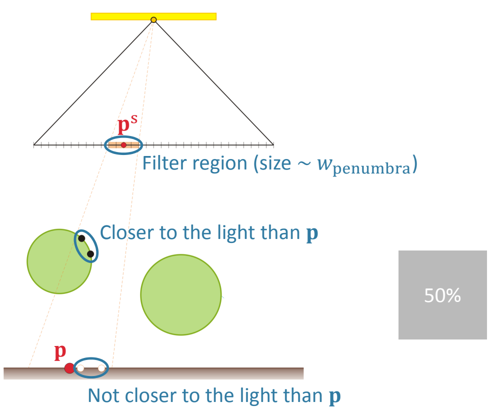
Percentage-Closer Soft Shadows



2. Penumbra width estimation

$$w_{\text{penumbra}} = \frac{\mathbf{p}_z^s - z_{\text{avg}}}{z_{\text{avg}}} w_{\text{light}}$$

Percentage-Closer Soft Shadows



3. Filtering

Percentage-Closer Soft Shadows

- Three steps
  - Blocker search
  - Penumbra width estimation
  - Filtering

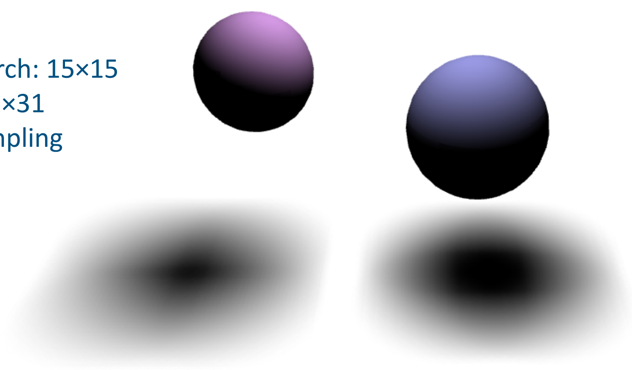
Two of them require many
shadow map accesses!

- Acceleration approaches
  - Subsampling
  - Prefiltering

Percentage-Closer Soft Shadows

- Quality vs. number of shadow map samples

Blocker search: 15×15
Filtering: 31×31
Regular sampling

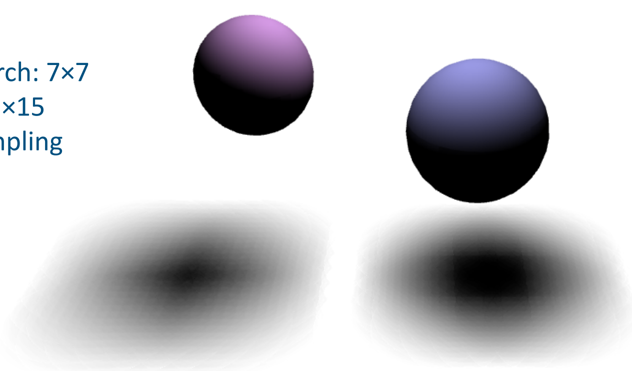


29 fps
(1024×1024, GeForce GTX 285)

Percentage-Closer Soft Shadows

- Quality vs. number of shadow map samples

Blocker search: 7×7
Filtering: 15×15
Regular sampling

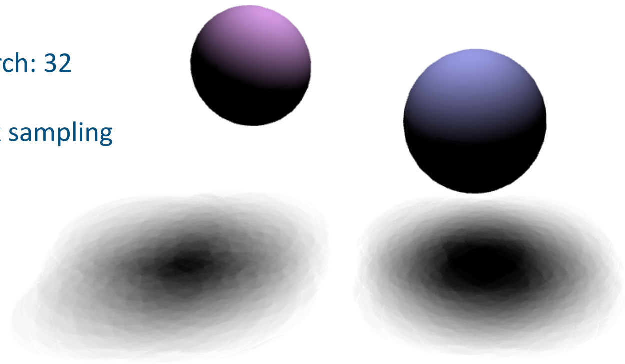


120 fps
(1024×1024, GeForce GTX 285)

Percentage-Closer Soft Shadows

- Quality vs. number of shadow map samples

Blocker search: 32
Filtering: 64
Poisson disk sampling

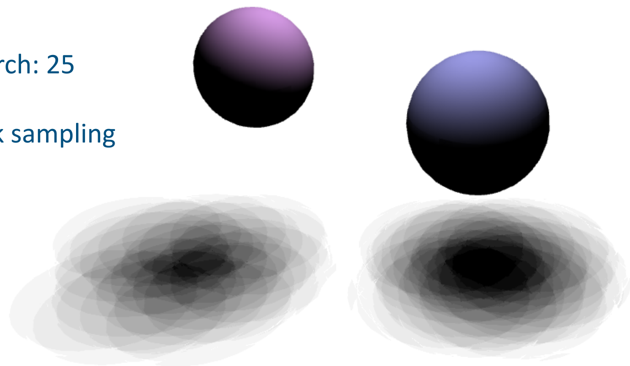


321 fps
(1024×1024, GeForce GTX 285)

Percentage-Closer Soft Shadows

- Quality vs. number of shadow map samples

Blocker search: 25
Filtering: 25
Poisson disk sampling



519 fps
(1024×1024, GeForce GTX 285)

Prefiltering

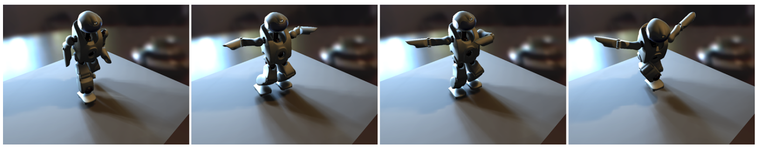
- Filtering step does just percentage-closer filtering
  - Using alternative shadow map representations (like VSM or CSM) allows prefiltering
  - Then, blurring reduces to a single texture fetch

But how to support adaptive filter region sizes?

- Mipmapping (or alternatively N-buffers)
  - Store results for several discrete filter sizes and interpolate
- Summed-area table
  - Supports arbitrary rectangular box filter kernels

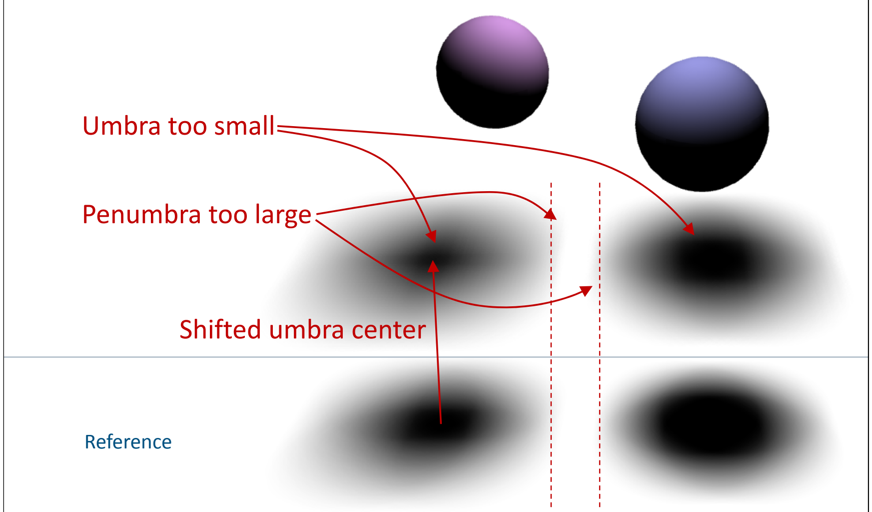
Prefiltering

- Blocker search can also be sped up with prefiltering
- Convolution Soft Shadows
  [Annen et al., 2008: "Real-Time, All-Frequency Shadows in Dynamic Scenes"]
  - Observation: Averaging the depth of shadow map samples closer to the light can be expressed as a convolution
  - Hence approach analogous to CSM is possible



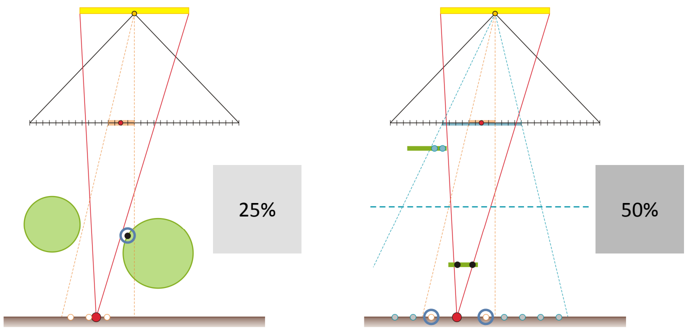
Annen et al., 2008

Percentage-Closer Soft Shadows



Percentage-Closer Soft Shadows

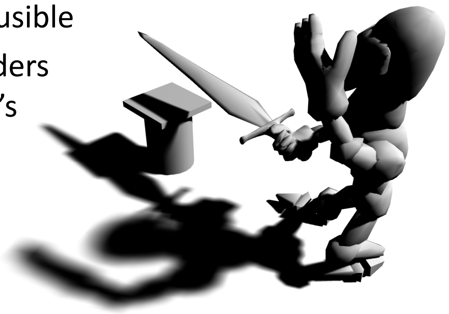
- Main sources of physical incorrectness
  - Single planar occluder assumption
  - Classification as light blocking solely based on depth test



Percentage-Closer Soft Shadows

+ Simple and reasonably fast
+ Often visually pleasing results
  (at least for smaller light sources)

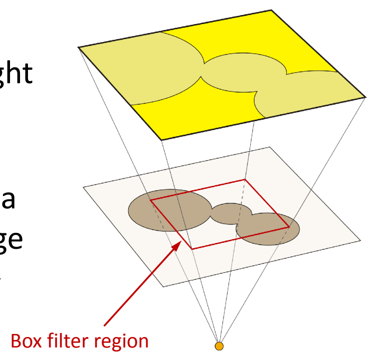
– Not really physically plausible
– Only accounts for occluders visible from light source's center



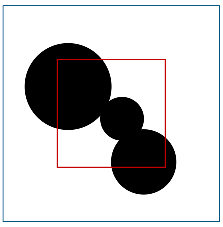
Visibility Computation = Filtering

[Soler and Sillion, 1998]

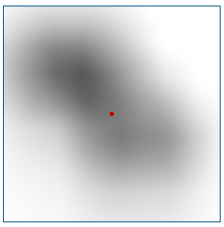
- Rectangular light source
- Planar occluder parallel to light
  - Represented by blocker image
- Visibility factor is obtained via box filtering the blocker image
  - Filter size equals appropriately scaled light size



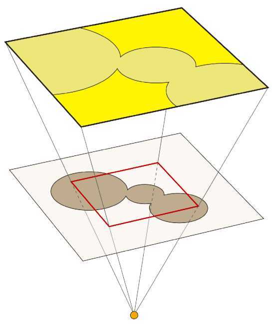
Visibility Computation = Filtering



Blocker image



Box-filtered blocker image

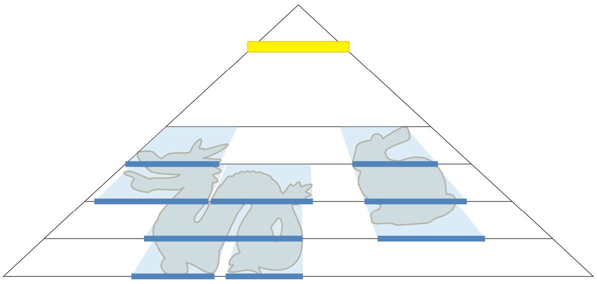


- Advantages
  - Enables prefiltering (e.g. summed-area tables)
  - Query becomes constant-time, i.e. independent of light size

Occlusion Textures

[Eisemann and Décoret, 2006]

- Decompose scene into multiple planar layers
  - Slice scene parallel to light source
  - Project geometry within slice onto slice's bottom plane

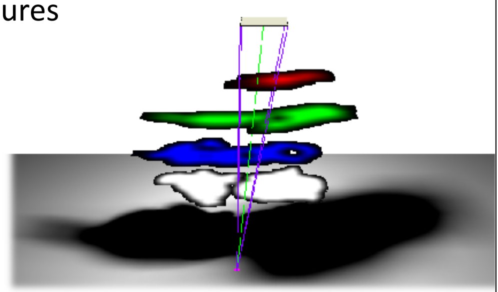


Occlusion Textures

- The covered parts of each slice are encoded in a binary occlusion texture (= blocker image)

Occlusion Textures

- Prefilter occlusion textures
  - Mipmapping
  - N-buffers
  - Summed-area table



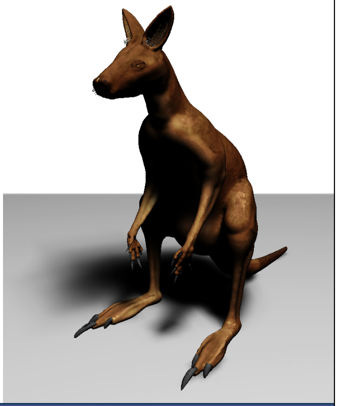
- For each blocking slice, lookup appropriately filtered response in prefiltered occlusion texture
  - Accumulate shadow contributions (multiplicatively)

Occlusion Textures

+ Plausible soft shadows at high frame rates
+ Performance independent of light size

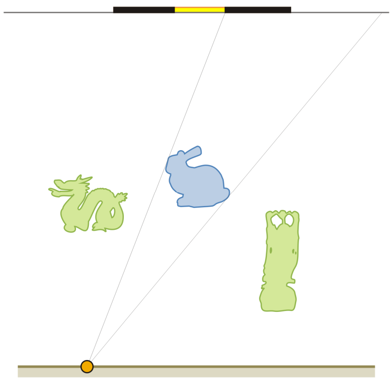
- Mainly suited for compact indoor environments

– Only heuristic handling of occluder fusion
– Discretizes occluders into small number of perforated planes



Occluder Backprojection

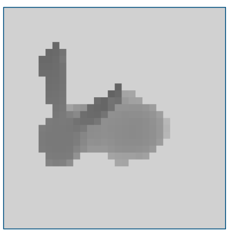
- For each (relevant) occluder
  - Project it onto light source
  - Determine covered light area
  - Aggregate this occlusion information
- Gathering
  For all receiver points
  For all potential occluders
- Scattering
  For all occluders
  For all affected receiver points

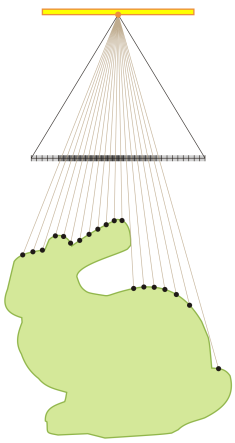


Soft Shadow Mapping

[Atty et al., 2006; Guennebaud et al., 2006]

- Approximate (subset of) occluder geometry
- Generate shadow map (from light's center)

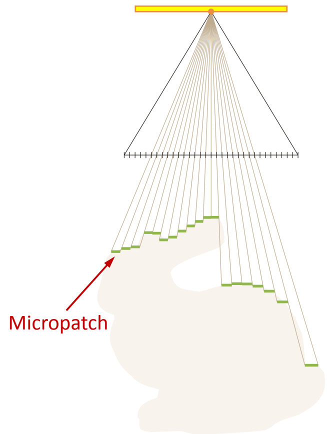




Soft Shadow Mapping

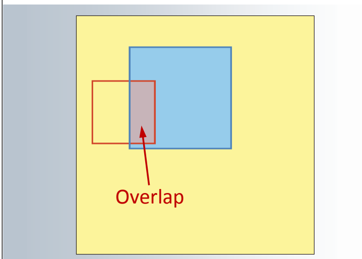
[Atty et al., 2006; Guennebaud et al., 2006]

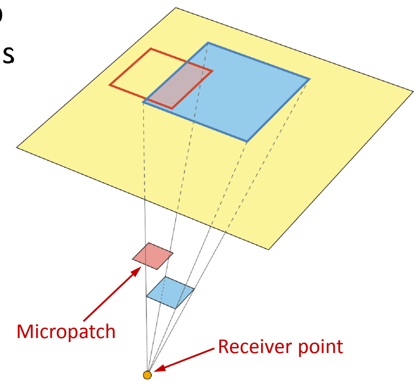
- Approximate (subset of) occluder geometry
- Generate shadow map (from light's center)
- Derive occluder approximation by unprojecting texels into world space
  - ➡ Micropatches



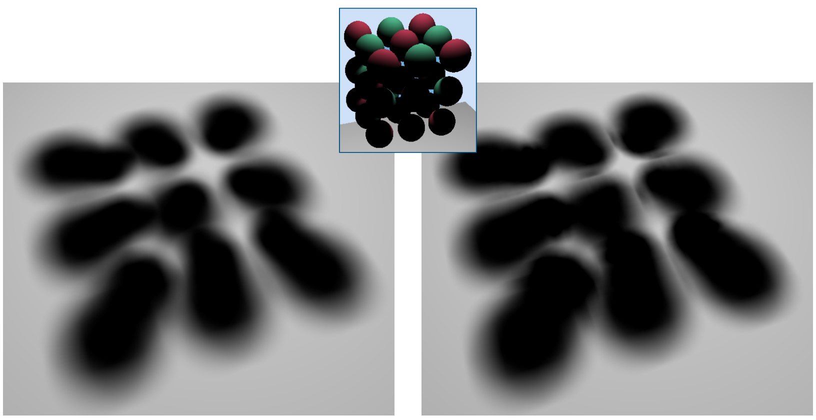
Soft Shadow Mapping

- Backproject micropatches onto light source to determine visibility
- Simple approach: Sum up projections' covered areas
  - Ignores overlaps





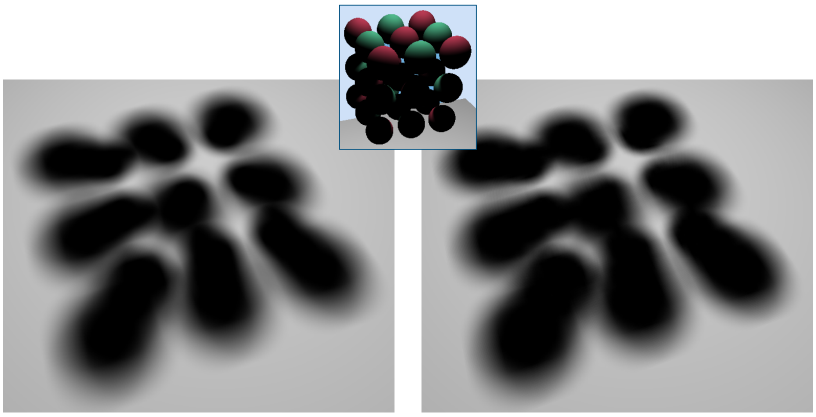
Overlapping Artifacts



Reference

Area accumulation
(without accounting for overlaps)

Overlapping Artifacts



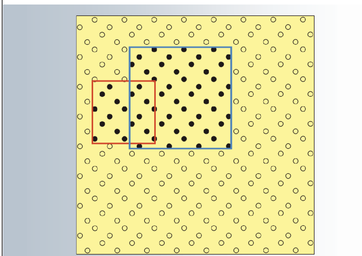
Reference

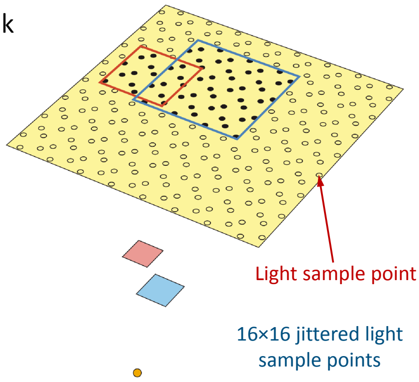
Occlusion bitmasks
(with correct overlap handling)

Occlusion Bitmasks

[Schwarz & Stamminger, 2007: "Bitmask Soft Shadows"]

- Sample visibility instead of accumulating areas
  - Set of binary point-to-point visibility relations
  - Bit field is employed to track visibilities of sample points on light source





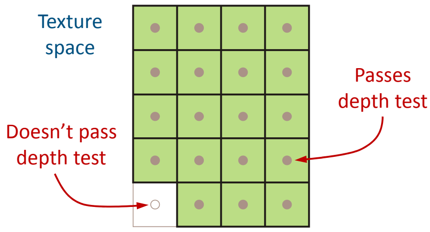
Soft Shadow Mapping

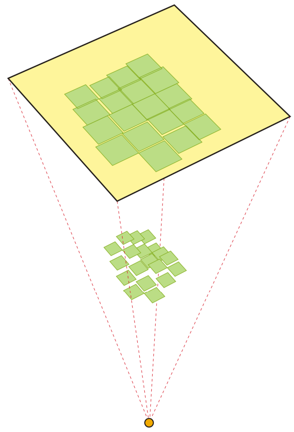
- Generate shadow map
- For each receiver point (pixel shader)
  - Loop over (relevant) occluder approximations (derived on-the-fly from shadow map)
    - Micropatches
    - Microquads (and microtris)
    - Occluder contours
  - Backproject onto light source
  - Determine and accumulate occlusion
    - Area accumulation (heuristic)
    - Occlusion bitmasks (sample-based)

Micropatches

[Atty et al., 2006; Guennebaud et al., 2006]

- Texel-sized rectangles constructed around each unprojected sample that passes depth test

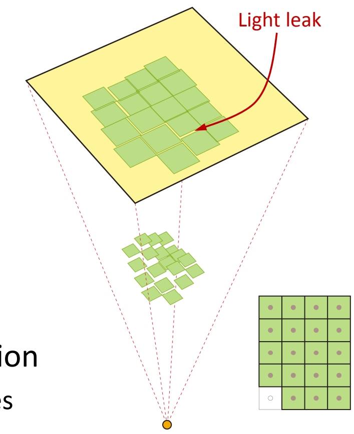




Micropatches

[Atty et al., 2006; Guennebaud et al., 2006]

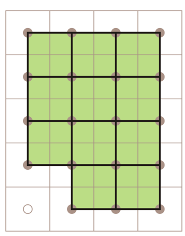
- Texel-sized rectangles constructed around each unprojected sample that passes depth test
- Suffer from light leaks
  - Requires gap filling, e.g. by extending micropatches
- Frequent occluder overestimation
  - But helps capturing fine structures

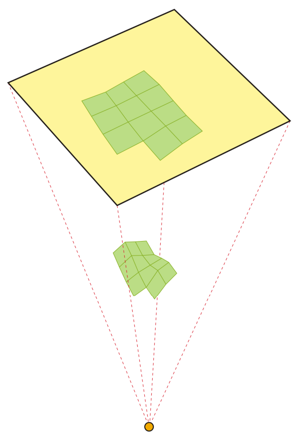


Microquads & Microtris

[Schwarz & Stamminger, 2007, 2008]

- Quads constructed from 2×2 adjacent unprojected samples that pass depth test

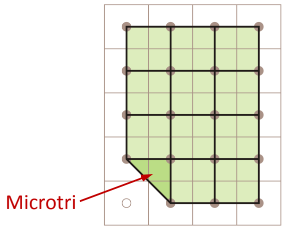


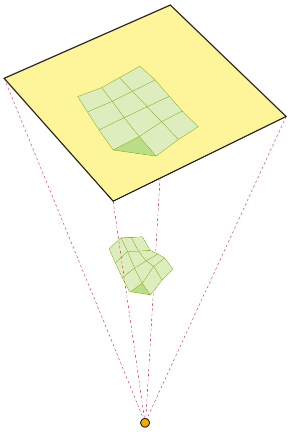


Microquads & Microtris

[Schwarz & Stamminger, 2007, 2008]

- Quads constructed from 2×2 adjacent unprojected samples that pass depth test

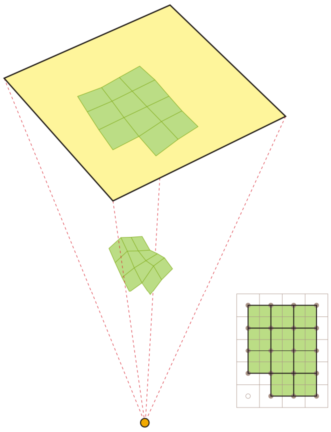




Microquads & Microtris

[Schwarz & Stamminger, 2007, 2008]

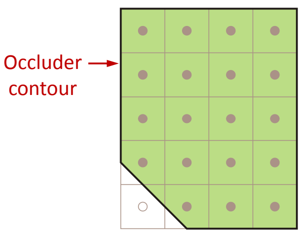
- Quads constructed from 2×2 adjacent unprojected samples that pass depth test
- No light leaks
- Less prone to surface acne
- Tendency to underestimate occluders' extents
  - May miss fine structures

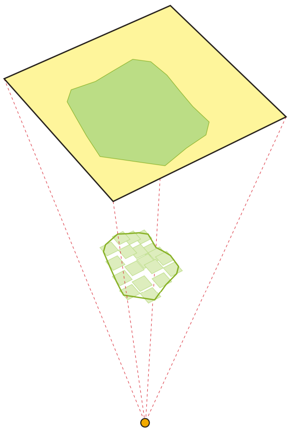


Occluder Contours

[Guennebaud et al., 2007]

- Contours constructed for each connected region of samples passing depth test

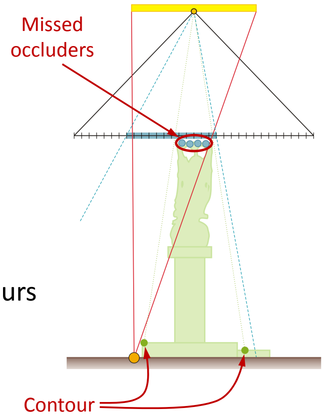




Occluder Contours

[Guennebaud et al., 2007]

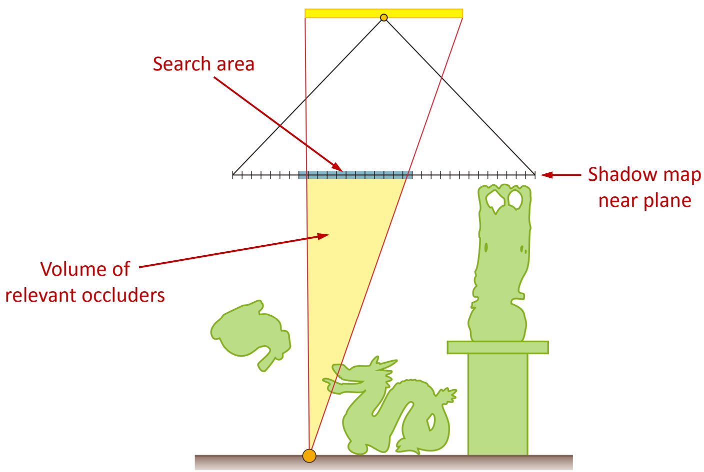
- Almost no light leaks
- Contour ≠ Silhouette
  - Occluders recorded in shadow map may be missed
- But also causes coherence
  - Neighboring receiver points often exclusively process identical contours
  - Allows for packet-based approach [Yang et al., 2009]



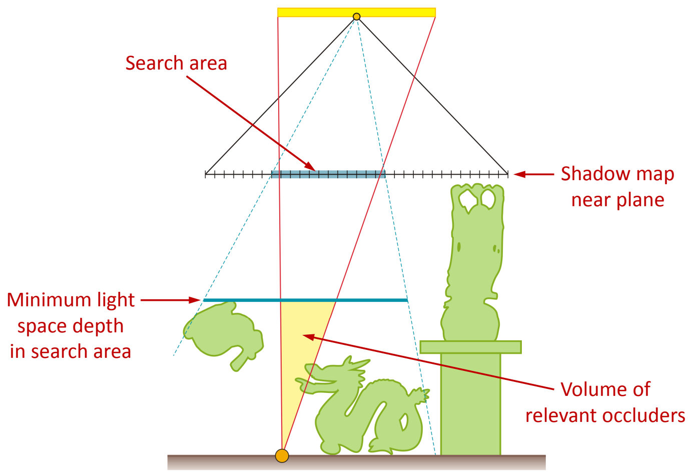
Acceleration

- Avoid useless computations
  - Multi-scale representations
  - Search area pruning
  - Hierarchical occluder construction
- Adapt accuracy
  - Micro-occluder subsampling
  - Coarser occluder approximations
  - Subsampling in screen space

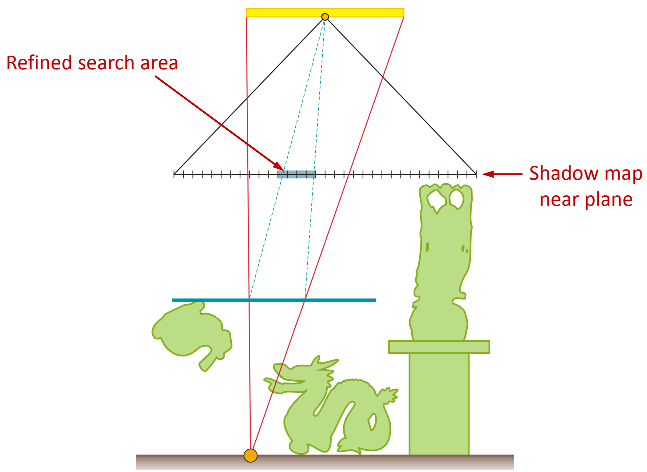
Search Area Determination



Search Area Determination

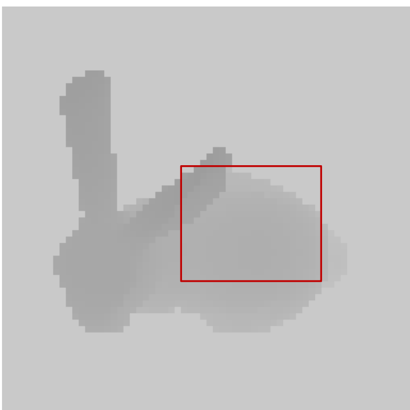


Search Area Determination



Acceleration Structures

How to determine depth range of a shadow map region?



Hierarchical Shadow Map

[Guennebaud et al., 2006]

- Min/max pyramid of shadow map (hierarchical z-buffer)
- Stored in mipmap chain of shadow map

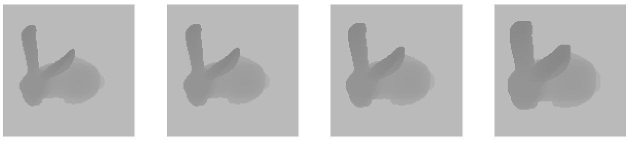
...

- Cheap, but often yields loose search areas

Multi-scale Shadow Map

[Schwarz & Stamminger, 2007]

- Stack of depth ranges for all power-of-two-sized neighborhoods (similar to N-buffers)
- Stored in array texture



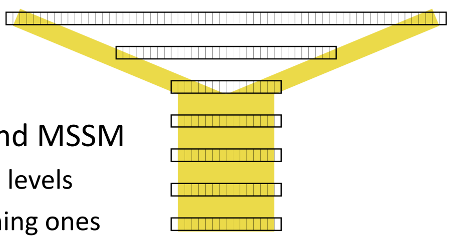
...

- Tight search areas, but can be costly

Hybrid Y Shadow Map

[Schwarz & Stamminger, 2008]

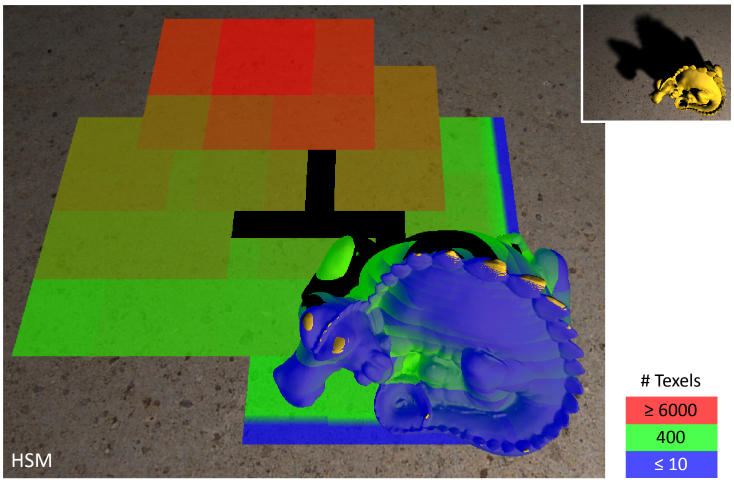
- Goal: Get the best of both
  - Low cost from HSM
  - Tight search areas from MSSM



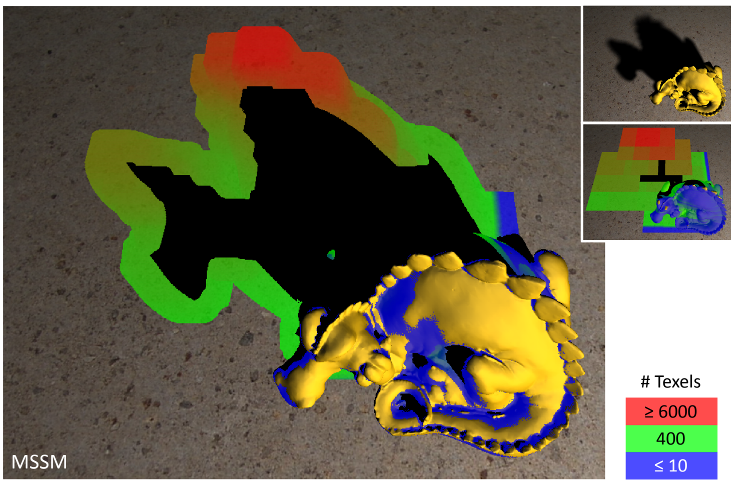
- Hybrid between HSM and MSSM
  - Pyramid (HSM) for finest levels
  - Stack (MSSM) for remaining ones

Acceleration Structures

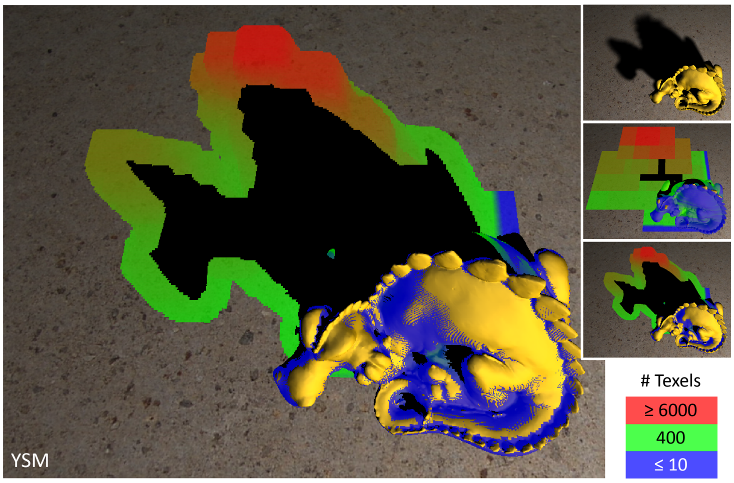
Acceleration Structures



Acceleration Structures



Acceleration Structures



Hierarchical Occluder Construction

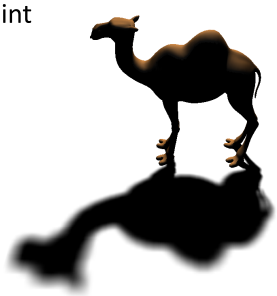
- HSM is a quadtree constructed over shadow map
  - Traverse this tree to identify and process relevant micropatches
    [Dmitriev, 2007: "Soft Shadows", NVIDIA WP-03016-001_v01]
- MSSM is essentially a forest of quadtrees
  - Hierarchically extract occluder contours
    [Yang et al., 2009: "Packet-based Hierarchal Soft Shadow Mapping"]

Acceleration

- Avoid useless computations
  - Multi-scale representations
  - Search area pruning
  - Hierarchical occluder construction
- Adapt accuracy
  - Micro-occluder subsampling
  - Coarser occluder approximations
  - Subsampling in screen space

Coarser Occluder Approximations

- Goal: Limit number of processed shadow map texels
- Approach: Use coarser-resolution shadow map
  - Can take appropriate HSM level (min channel)
  - Level selection is done per receiver point
  - Correct depth bias?
- May lower visual quality
  - Decreased smoothness and detail in the shadow shape
  - Changes in shadow region size



Coarser Occluder Approximations

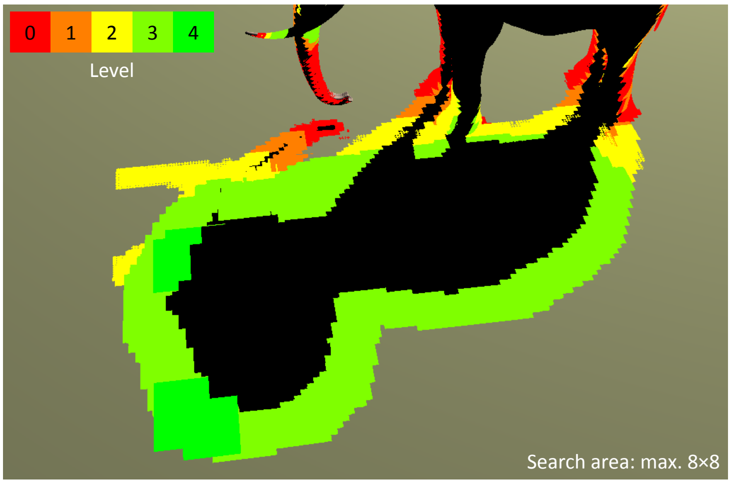
Coarser Occluder Approximations

Coarser Occluder Approximations

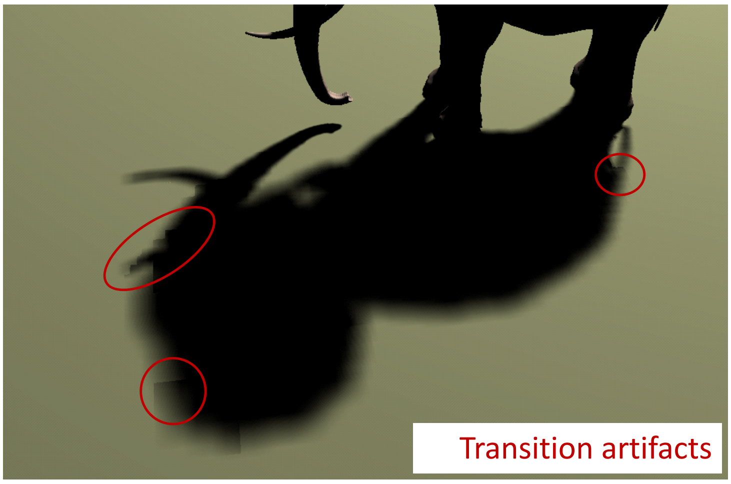
Coarser Occluder Approximations

Coarser Occluder Approximations

Coarser Occluder Approximations



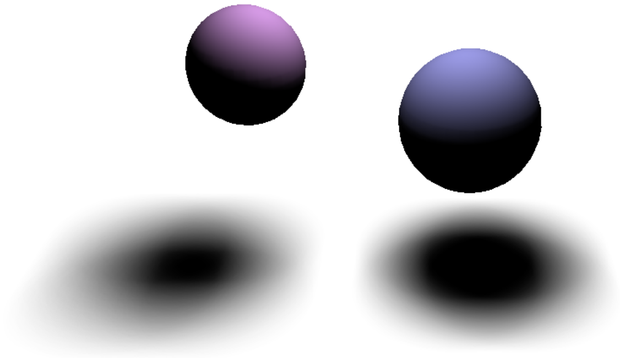
Coarser Occluder Approximations



Coarser Occluder Approximations

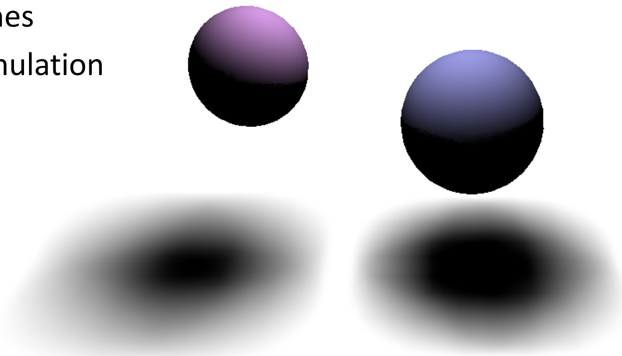
Example

- Reference



Example

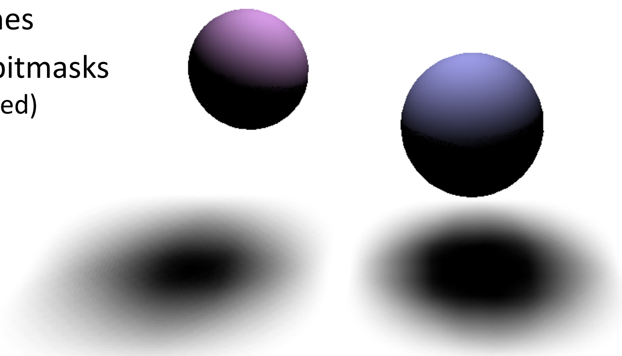
- Search area: max. 20×20
- Micropatches
- Area accumulation
- YSM



105 fps
(1024×1024, GeForce GTX 285)

Example

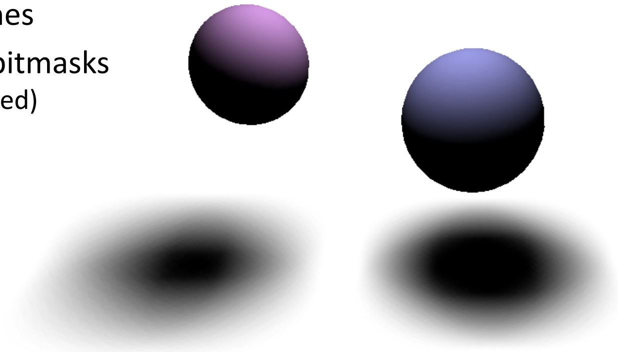
- Search area: max. 20×20
- Micropatches
- Occlusion bitmasks (16×16, jittered)
- YSM



51 fps
(1024×1024, GeForce GTX 285)

Example

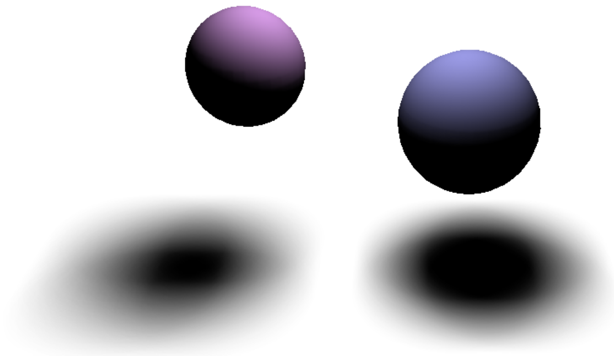
- No imposed search area bound
- Micropatches
- Occlusion bitmasks
  (16×16, jittered)
- YSM



< 1 fps
(1024×1024, GeForce GTX 285)

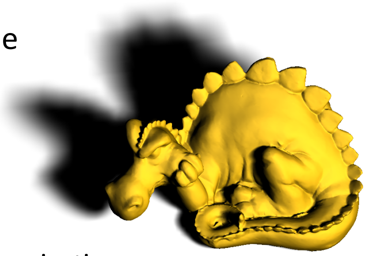
Example

- Reference again



Soft Shadow Mapping

+ Physically plausible
+ Rather high quality at real-time frame rates possible



- Performance strongly dependent on
  - Search area size
  - Number of pixels requiring backprojection

– Uses only approximation of subset of occluders
  - Typically those visible from the light source’s center

– Usually all gaps are closed invariably

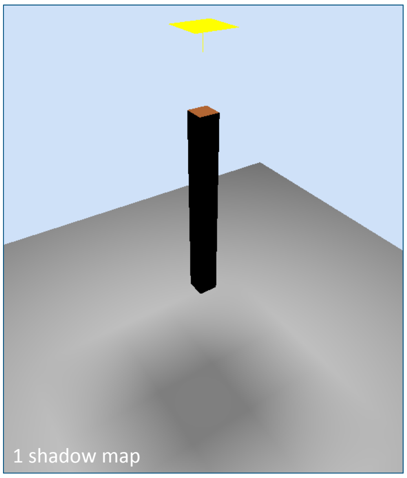
Image-based Approaches

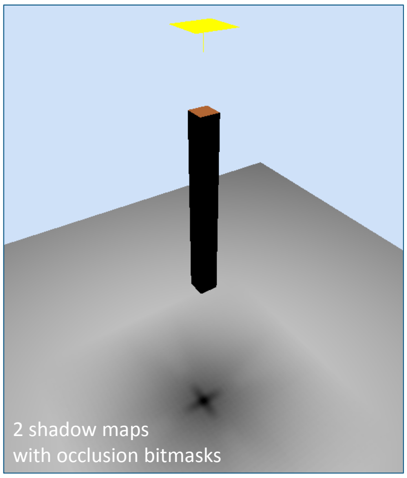
- Percentage-closer soft shadows
- Occlusion textures
- Soft shadow mapping

- Use sampled representation of (subset of) occluders
  - \+ Easily supports versatile geometry
  - – Limits accuracy

- Alternative: Geometry-based representation
  - \+ Can be more accurate and avoids aliasing problems
  - – Approaches typically slower

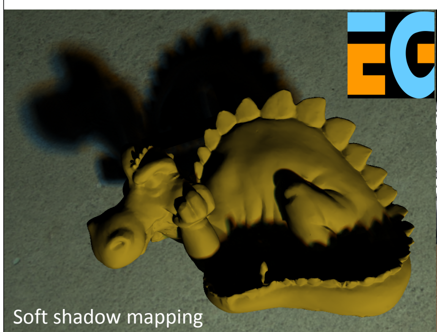
Some vs. All Occluders

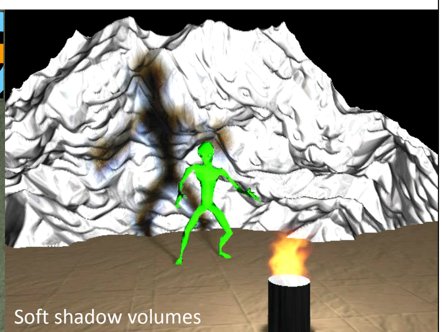

1 shadow map


2 shadow maps
with occlusion bitmasks

Occluder Backprojection

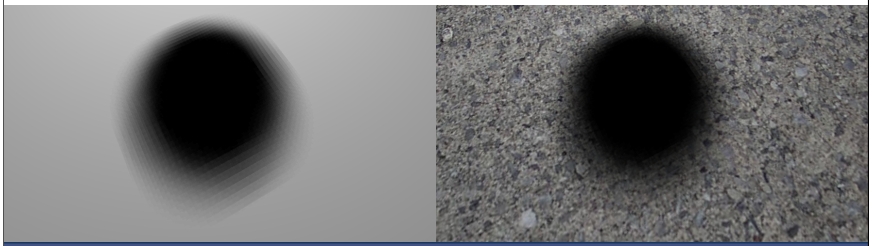
- Identifies part of light source blocked by occluder
  - And thus whether occluder blocks light at all!
- Enables support for textured light sources


Soft shadow mapping


Soft shadow volumes

Point Sampling of Light Area

- Enables correct occluder fusion
- Enables evaluating direct area lighting equation via Monte-Carlo integration
- Can lead to discretization artifacts
  - Use a sufficiently high number of well-distributed samples



Soft Shadows – Geometry Based Approaches

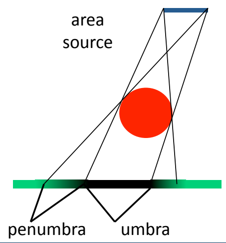
By Ulf Assarsson

Geometry Based Approaches

Related algorithms:

- *Soft Planar Shadows Using Plateaus* [Haines 2001]
- *Penumbra Maps* [Wyman and Hansen 03]
- *Smoothies* [Chan and Durand 03]
- *Penumbra Wedges* [Akenine-Möller and Assarsson 02]
- *Soft Shadow Volumes* [Assarsson and Akenine-Möller 03]

Idea:

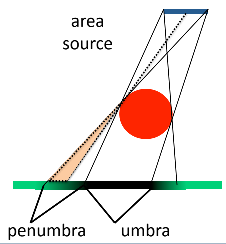


Geometry Based Approaches

Related algorithms:

- ***Soft Planar Shadows Using Plateaus*** [Haines 2001]
- ***Penumbra Maps* [Wyman and Hansen 03]**
- ***Smoothies* [Chan and Durand 03]**
- *Penumbra Wedges* [Akenine-Möller and Assarsson 02]
- *Soft Shadow Volumes* [Assarsson and Akenine-Möller 03]

Idea:

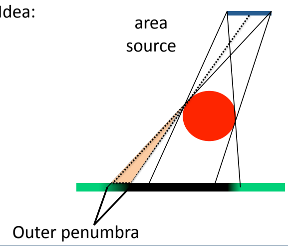


Geometry Based Approaches

Related algorithms:

- ***Soft Planar Shadows Using Plateaus*** [Haines 2001]
- ***Penumbra Maps* [Wyman and Hansen 03]**
- ***Smoothies* [Chan and Durand 03]**
- *Penumbra Wedges* [Akenine-Möller and Assarsson 02]
- *Soft Shadow Volumes* [Assarsson and Akenine-Möller 03]

Idea:

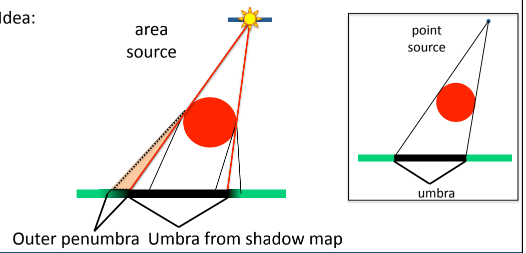


Geometry Based Approaches

Related algorithms:

- *Soft Planar Shadows Using Plateaus* [Haines 2001]
- ***Penumbra Maps* [Wyman and Hansen 03]**
- ***Smoothies* [Chan and Durand 03]**
- *Penumbra Wedges* [Akenine-Möller and Assarsson 02]
- *Soft Shadow Volumes* [Assarsson and Akenine-Möller 03]

Idea:

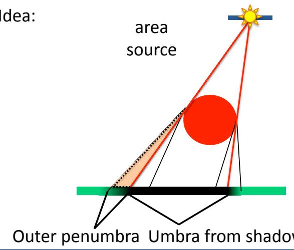


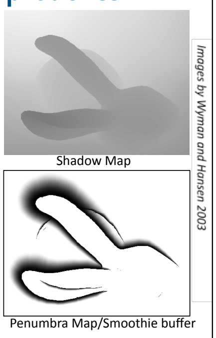
Geometry Based Approaches

Related algorithms:

- *Soft Planar Shadows Using Plateaus* [Haines 2001]
- ***Penumbra Maps* [Wyman and Hansen 03]**
- ***Smoothies* [Chan and Durand 03]**
- *Penumbra Wedges* [Akenine-Möller and Assarsson 02]
- *Soft Shadow Volumes* [Assarsson and Akenine-Möller 03]

Idea:

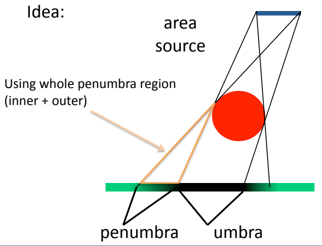




Geometry Based Approaches

Related algorithms:
- *Soft Planar Shadows Using Plateaus* [Haines 2001]
- *Penumbra Maps* [Wyman and Hansen 03]
- *Smoothies* [Chan and Durand 03]
- ***Penumbra Wedges*** [Akenine-Möller and Assarsson 02]
- ***Soft Shadow Volumes*** [Assarsson and Akenine-Möller 03]

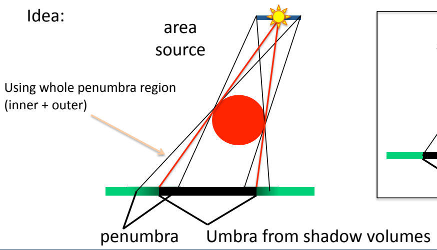
Idea:

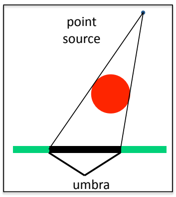


Geometry Based Approaches

Related algorithms:
- *Soft Planar Shadows Using Plateaus* [Haines 2001]
- *Penumbra Maps* [Wyman and Hansen 03]
- *Smoothies* [Chan and Durand 03]
- ***Penumbra Wedges*** [Akenine-Möller and Assarsson 02]
- ***Soft Shadow Volumes*** [Assarsson and Akenine-Möller 03]

Idea:

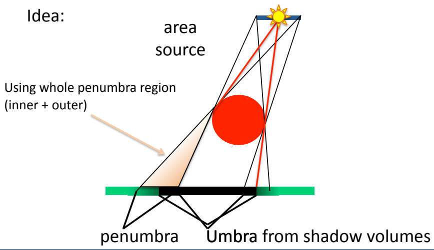




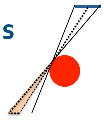
Geometry Based Approaches

Related algorithms:
- *Soft Planar Shadows Using Plateaus* [Haines 2001]
- *Penumbra Maps* [Wyman and Hansen 03]
- *Smoothies* [Chan and Durand 03]
- ***Penumbra Wedges*** [Akenine-Möller and Assarsson 02]
- ***Soft Shadow Volumes*** [Assarsson and Akenine-Möller 03]

Idea:

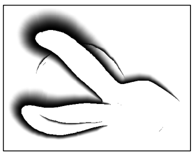


Outer Penumbra methods



- Penumbra Maps[Wyman and Hansen 03]
- Smoothies [Chan and Durand 03]
- Idea:
  - Add soft shadows by extending shadow map outwards

Shadow Map

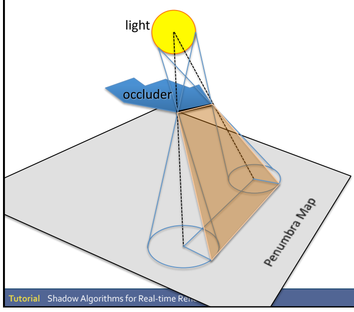


Penumbra Map/Smoothie buffer

Images by Wyman and Hansen 2003

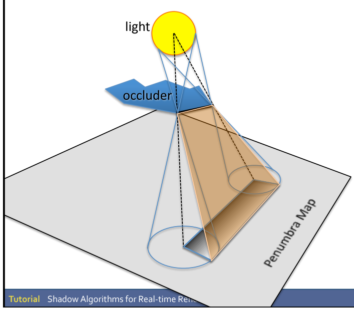
Outer Penumbra methods

- Penumbra Maps:



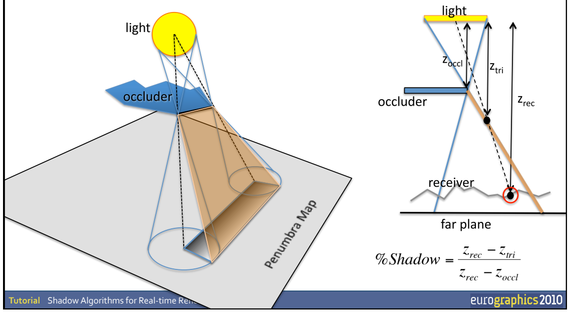
Outer Penumbra methods

- Penumbra Maps:



Outer Penumbra methods

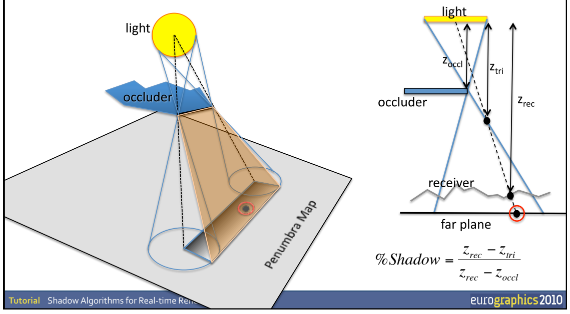
- Penumbra Maps:



Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Outer Penumbra methods

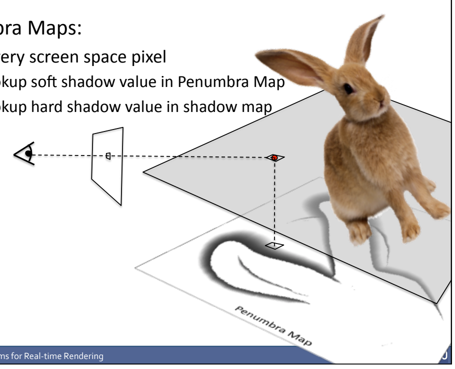
- Penumbra Maps:



Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Outer Penumbra methods

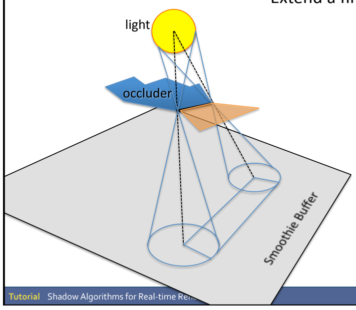
- Penumbra Maps:
  - For every screen space pixel
    - Lookup soft shadow value in Penumbra Map
    - Lookup hard shadow value in shadow map



Tutorial Shadow Algorithms for Real-time Rendering

Outer Penumbra methods

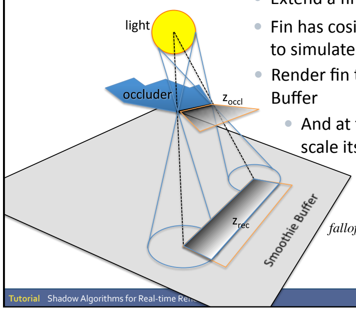
- Smoothies:
  - For each silhouette edge
    - Extend a fin (always same extent)



Tutorial Shadow Algorithms for Real-time Re... eurographics2010

Outer Penumbra methods

- Smoothies:
  - For each silhouette edge
    - Extend a fin (always same extent)
    - Fin has cosine-falloff shading to simulate soft shadow
    - Render fin to Smoothie Buffer
      - And at the same time: scale its falloff with

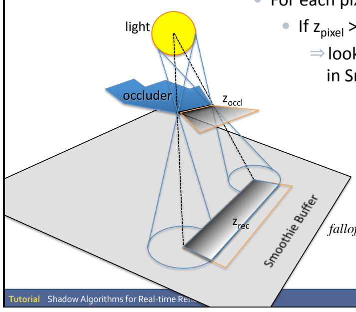


$$falloff = falloff * \left(\frac{z_{occl}}{z_{rec}} > 1\right) ? \ 1 \ : \ \frac{1}{1 - \frac{z_{occl}}{z_{rec}}}$$

Tutorial Shadow Algorithms for Real-time Re... eurographics2010

Outer Penumbra methods

- Smoothies:
  - Finally, render scene from camera
    - For each pixel
      - If $z_{pixel} > z_{occl}$
        $\Rightarrow$ lookup soft shadow value in Smoothie buffer

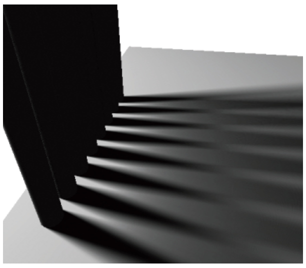


$$falloff = falloff * \left(\frac{z_{occl}}{z_{rec}} > 1\right) ? \ 1 \ : \ \frac{1}{1 - \frac{z_{occl}}{z_{rec}}}$$

Tutorial Shadow Algorithms for Real-time Re... eurographics2010

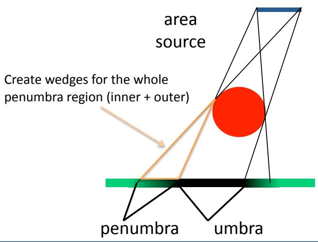
Problems

Outer Penumbra methods can never produce diminishing umbra

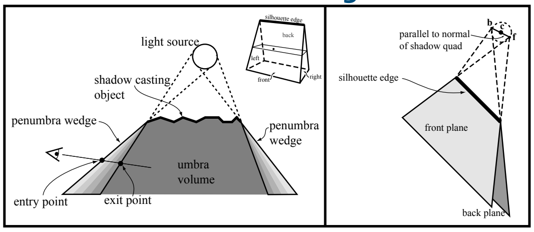


Penumbra Wedges

- Penumbra Wedges was introduced 2002 [Akenine-Möller and Assarsson 02]



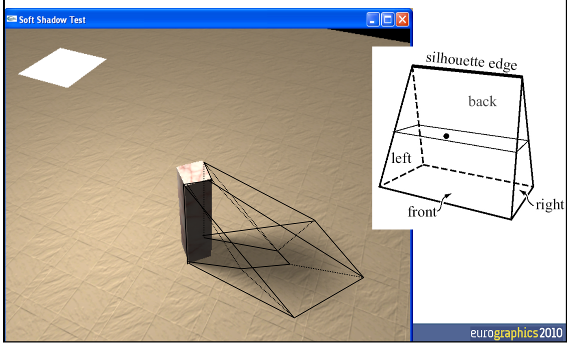
Penumbra Wedges



- Apply soft shadow value by linear interpolation inside wedge
  - Wedge rasterization done in software

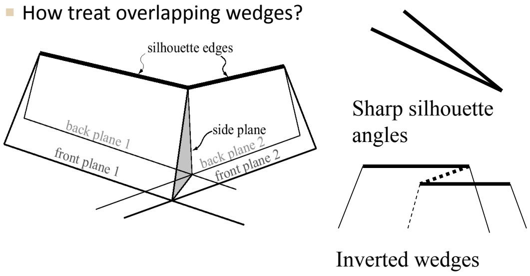
Penumbra Wedges

- Simple example:



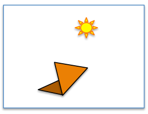
Penumbra Wedges

- Caveat:
  - Separating side plane
- How treat overlapping wedges?



Penumbra Wedges

- Silhouettes are complex

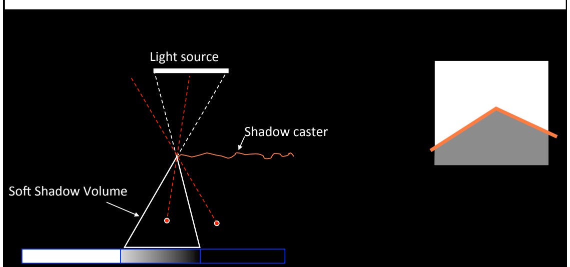


An edge with a front-facing and a back-facing adjacent triangle is considered a silhouette edge

Soft Shadow Volumes

- Soft Shadow Volumes was introduced 2003
  - [Assarsson and Akenine-Möller 03]
- Idea:
  - Every edge has an influence region enclosed by its wedge
  - Wedges are allowed to overlap
  - Somehow make a wedge's shadow contribution dependent only on its edge and not its edge's neighbors
    - Solution: Compute occluded light source's area by integrating area from the silhouette edges (Green's Theorem)

What's a Soft Shadow Volume?

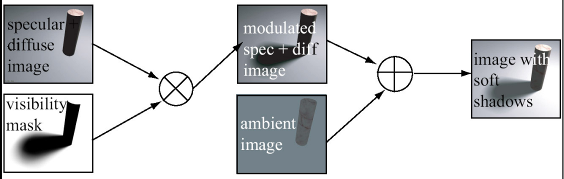


Soft Shadow Volume =

A. Volume from which an edge projects onto the light source

B. Region of penumbra caused by an edge

Overview



Computation of visibility mask:
- 1st pass: Render hard shadow using shadow volumes
  - This computes the integration constant in Green's Theorem, per pixel
- 2nd pass: Compute soft shadow value (using Green's Theorem)
  - By rasterizing the wedges

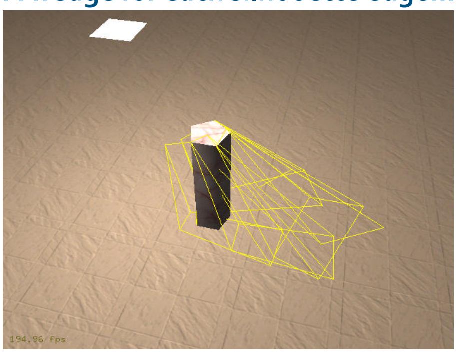
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Wedges

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

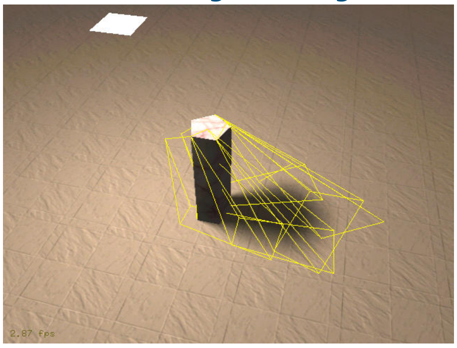
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

A wedge for each silhouette edge…

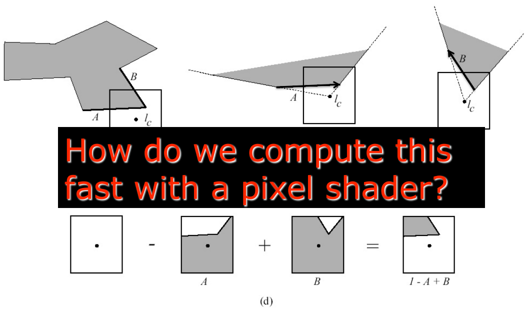


Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

Rasterizing the wedges

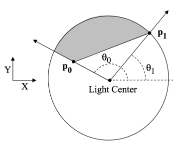


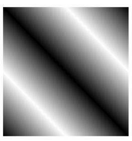
How the visibility computation works:



Computing light source occlusion

- Spherical light source[Assarsson et al 03]:
  - Project edge onto light source and clip
  - Analytically compute the coverage area with the help of an atan()-lookup table





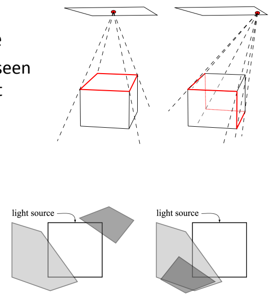
Lookup table: atan(y/x)

- [Johnson et al 09] avoids the lookup-table by using an inverse cosine function call
  - Probably faster on today's bandwidth limited GPUs.

Artifacts and a Solution

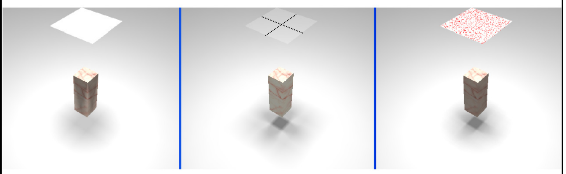
Two major approximation:

- Assume constant silhouette
  - I.e., one single silhouette as seen from all locations on the light source
  - Ignore overlap between occluders



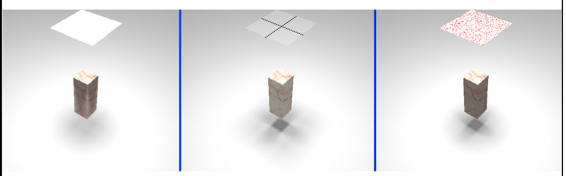
Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

For better quality, use 2x2 area lights



One light source | 2x2 light sources | 1024 point lights

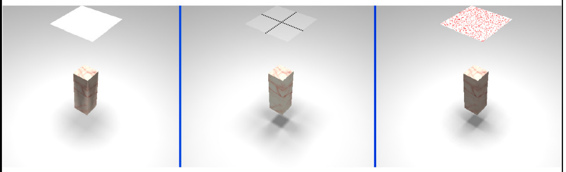
For better quality, use 2x2 area lights



One light source | 2x2 light sources | 1024 point lights


One light source | 1024 point lights

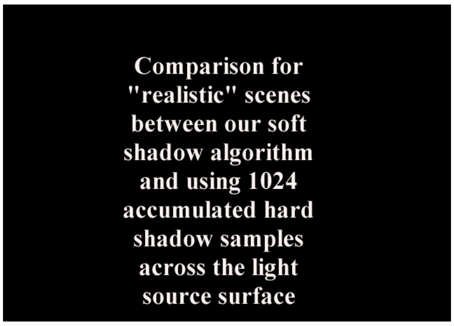
For better quality, use 2x2 area lights



One light source | 2x2 light sources | 1024 point lights

One light source | 2x2 light sources | 3x3 light sources | 1024 point lights

Quality (without splitting)



Soft Shadow Volumes | Ground Truth

Improving Occluder Fusion

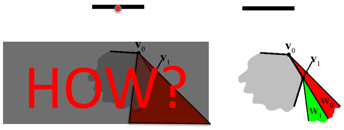
- Forest et al. [2006]
  - Track occlusion separately for 4 light source quadrant
    - Compute approx occlusion per silhouette edge (bounding boxes)
    - Using this info to estimate overlapping occlusion compensation
  - Roughly twice as expensive
  - Significantly higher quality than no split
  - Single silhouette

Correct Depth Complexity Sampling

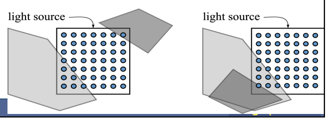
Soft Shadow Volumes with correct depth complexity sampling was introduced for Ray Tracing in 2005 [Laine et al. 05] and improved 2006 [Lehtinen et al. 06]

It features:

- Exact silhouettes

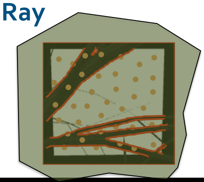


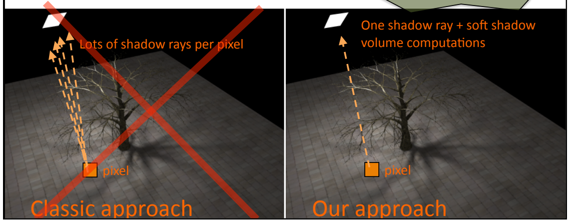
- Depth counters



Soft Shadow Volumes for Ray Tracing - overview

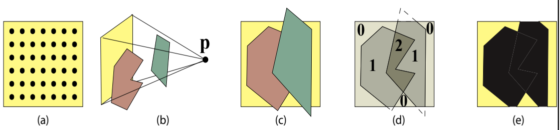
- Replace the shadow rays
  - With soft shadow volume computations
  - Plus one reference shadow ray





Correct Depth Complexity Sampling

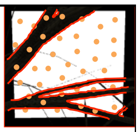
Overview:



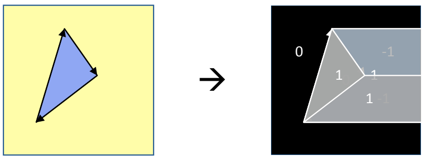
(a) (b) (c) (d) (e)

- a) A depth counter per light sample
- b) Two occluders located between point **p** and the light
- c) The occluders projected onto the light source from **p**
- d) The depth complexity function (#surface overlaps)
- e) The visibility function

Integration: Example



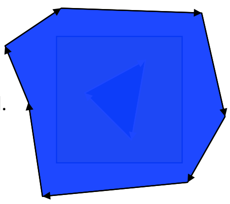
- Left-to-right integration of a triangular silhouette



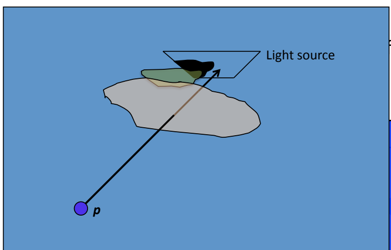
Light source as seen from ***p***   Depth complexity function

From Relative Depth Complexity to Visibility

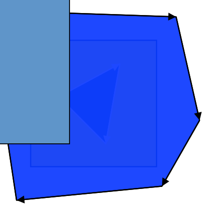
- We are not done yet, since the constant of integration is not known → cannot solve visibility
- Solution: cast a shadow ray ray to one s_i with lowest relative depth complexity
- If blocked, all s_i are blocked
- Otherwise, all s_i with lowest rel. depth complexity are visible



From Relative Depth Complexity to Visibility



f integration is

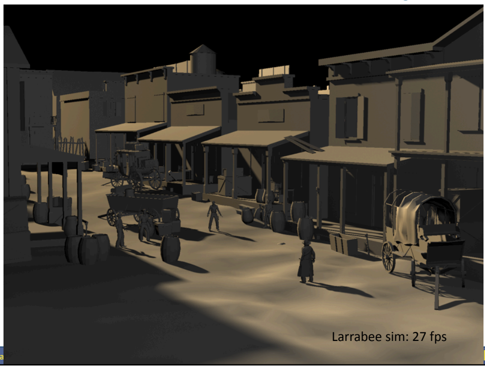


Real-time version

- Forest et al [08]
  - Shadow volumes to measure integration constant
    - Instead of a shadow ray to the sample with lowest depth complexity, they measure the depth complexity to the center sample using Shadow Volumes (as the original SSV-algorithm [Assarsson and Akenine-Möller 03])
  - Fragment shaders handles depth complexity sampling
    - 16-64 samples
    - (The CPU-versions by [Laine05,Lehtinen06] used 256-1024 samples)
    - Single silhouette
  - Doom3-scene, 18k triangles, ~3 fps, (small penumbra), 16 samples

Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

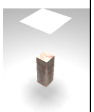
Soft irregular shadow mapping: Fast, high-quality, and robust soft shadows [Johnson et al. 2009]

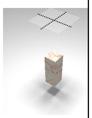


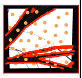
Pros & Cons

Options:

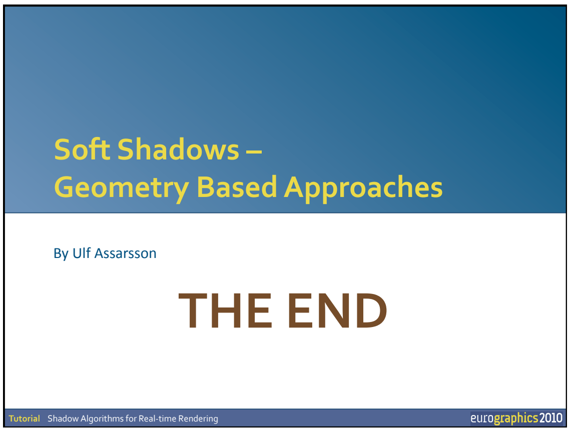
- **One light source with continuous (smooth) coverage value [0-1]**
  - Advantage:
    - cheapest fragment shader (16 instr [Forest et al. 06]),
    - smooth penumbra without undesirable discontinuities.
  - Disadvantage:
    - Overlapping artifact (pure additive occluder)
      - » This can easily result in too dark penumbra regions.
    - Single silhouette artifact
      - » Can lead to too bright and wide penumbra
- **Splitting light source area into four or more parts, with continuous (smooth) coverage value [0-1]** :
  - Advantage:
    - Drastically improved quality
    - Often indistinguishable from ground truth even for complicated cases
  - Disadvantage:
    - More expensive (~sqrt(#lights))
- **16-64 light samples with binary coverage values**
  - Advantage:
    - Physically accurate sample based soft shadows
  - Disadvantage:
    - An order of magnitude slower than just using one light with smooth coverage value
    - Discretization artifacts unless a huge number of samples is used
    - Higher GPU-memory cost for storing all samples during rendering







Tutorial Shadow Algorithms for Real-time Rendering eurographics2010

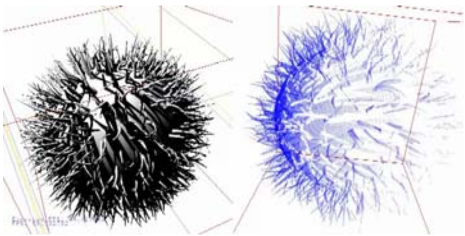

Soft Shadows –
Geometry Based Approaches
By Ulf Assarsson
THE END
Tutorial Shadow Algorithms for Real-time Rendering
eurographics2010

View-Sample Mapping

Background

- View-Sample Mapping for hard shadows

Camera View Light View

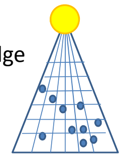


Why compute Soft Shadows with this?

- Reduce overdraw for Soft Shadow Volumes [Johnson et al. 2009]
- Compute truly accurate soft shadows [Eisemann&Décoret2007, Sintorn et al. 2008]

Silhouette-based Solution [Johnson et al. 2009]

- Compute Hard Shadows [Johnson et al. 2005]
- Detect silhouettes from light's center
- Test view samples against silhouette wedge



- Additive silhouette contributions [Assarsson and Akenine-Möller 2003]

Silhouette-based Solution [Johnson et al. 2009]

- Soft-shadow volume like shadow calculation
  - + No sampling artifacts
  - - Additive Occluder fusion
  - - Light center assumption

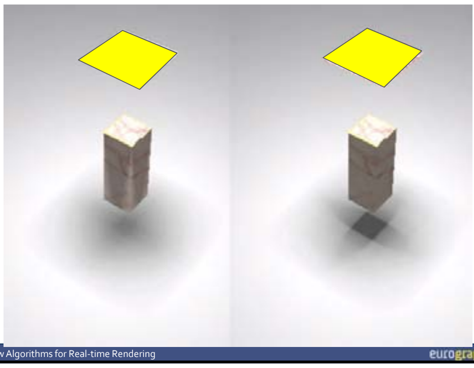
Reference

Soft Shadow Volumes

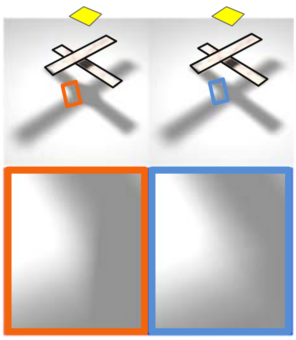
- Intelligent Mapping on Larrabee architecture

Truly Accurate Real-time Soft Shadows

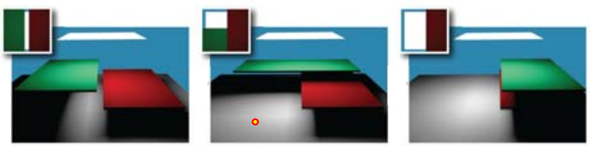
Single Silhouette Problem



Occluder Fusion Problem



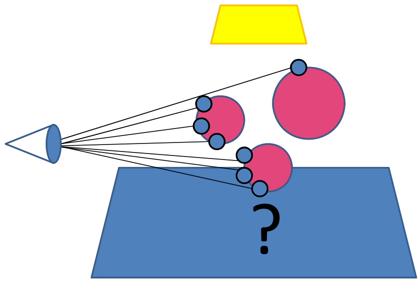
Occluder Fusion Problem



- What does a point see?

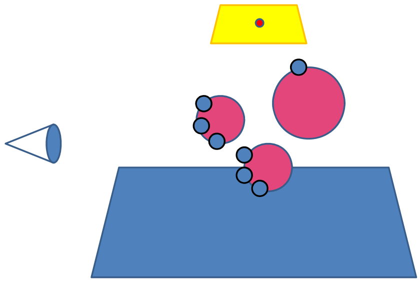
Simpler Problem

- What are the shadows on the plane?



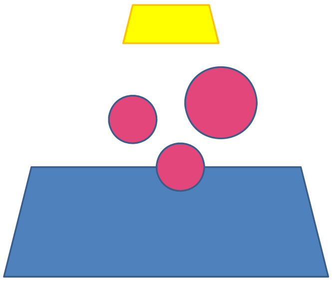
Simpler Problem

- What are the shadows on the plane?



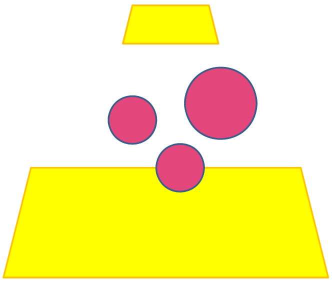
Simpler Problem

- What are the shadows on the plane?



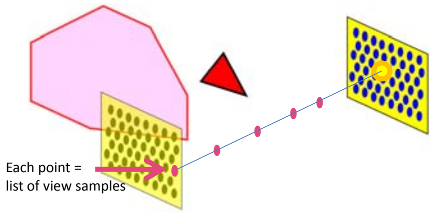
Simpler Problem

- What is the visibility between the planes?



General Scene

- Store list of view samples
- Far-Light plane = conservative distance for penumbra



- Assume a single receiver point per texel

Introduction

- Our Approach:
  - Reformulate Visibility Sampling
  - No special structures
  - Decorrelate sampling
  - Output: complete rays

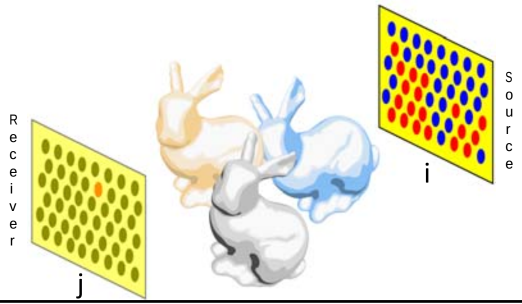
GPU adapted

- Applications:
  - Soft Shadows
  - Visibility visualization for level design

Visibility Sampling on the GPU [Eisemann&Décoret07]

- Compute blocked samples

$$B_j := \{S_i | \exists k\ [S_i, R_j] \cap T_k \neq \emptyset\}$$

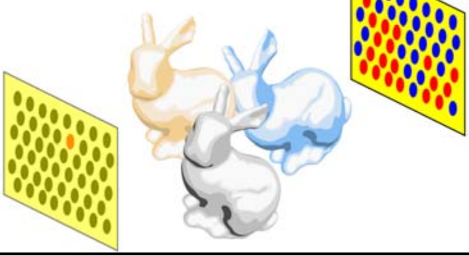


Principle

- Compute blocked samples

$$B_j := \{S_i | \exists k\ [S_i, R_j] \cap T_k \neq \emptyset\}$$

$$\forall i\ \forall j\ \forall k\ [S_i, R_j] \cap T_k \overset{?}{\neq} \emptyset$$



Principle

Raytracing

$$\forall i\ \forall j\ \forall k\ [S_i, R_j] \cap T_k \overset{?}{\neq} \emptyset$$

$$\forall i\ \forall k\ \forall j\ [S_i, R_j] \cap T_k \overset{?}{\neq} \emptyset$$

Shadow/Occlusion Mapping

$$\forall k\ \forall i\ \forall j\ [S_i, R_j] \cap T_k \overset{?}{\neq} \emptyset$$

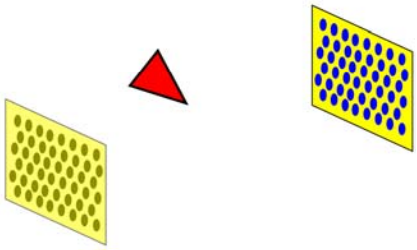
See [LA05]

Treat triangles separately

Info available in a vertex/geometry/fragment

Principle

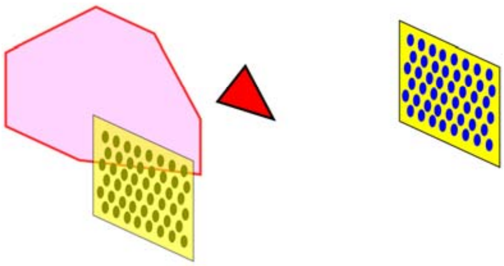
- Basic algorithm



Two patches with samples and occluding triangle

Principle

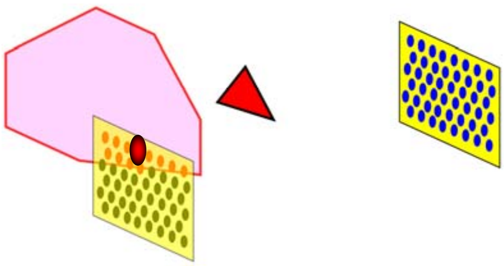
- Basic algorithm



Consider one patch as source and find penumbra region

Principle

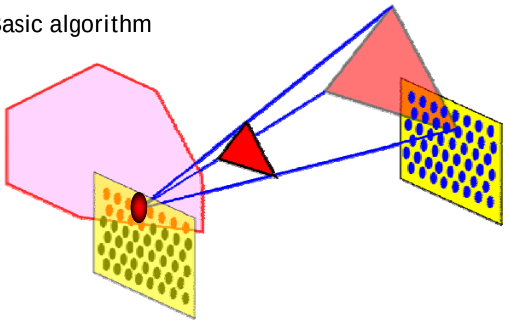
- Basic algorithm



Each sample in penumbra is treated

Principle

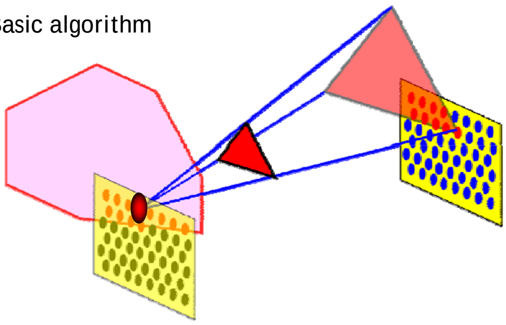
- Basic algorithm



Backproject triangle from sample point

Principle

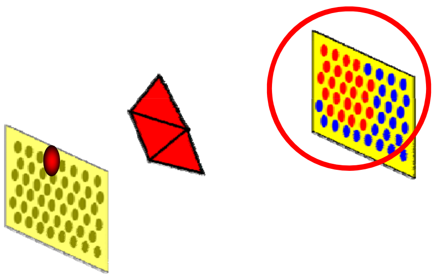
- Basic algorithm



Determine blocked samples

Principle

- Basic algorithm



Accumulate over all triangles

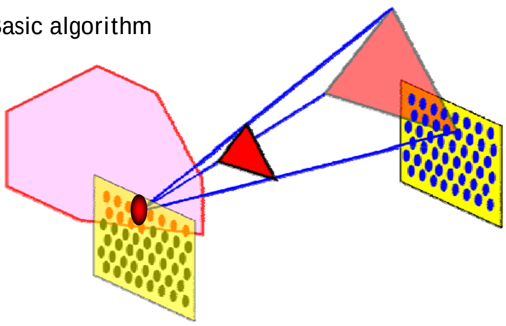
Our approach

- Overview:

Vertex/Geom Shader	Find influence region
	Backproject triangle
Fragment Shader	Determine blocked samples
Blending	Accumulate over all triangles

Where are we?

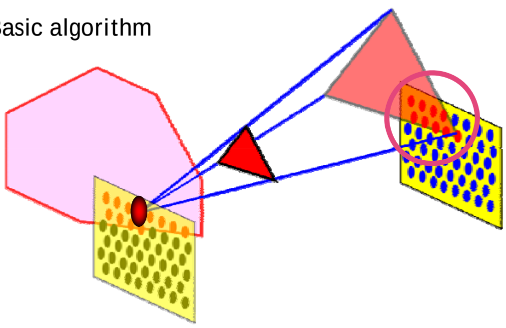
- Basic algorithm



Backproject triangle from sample point

Where are we?

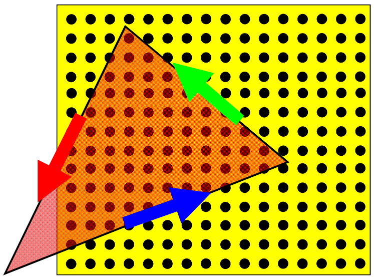
- Basic algorithm



Determine blocked samples

Blocked Samples

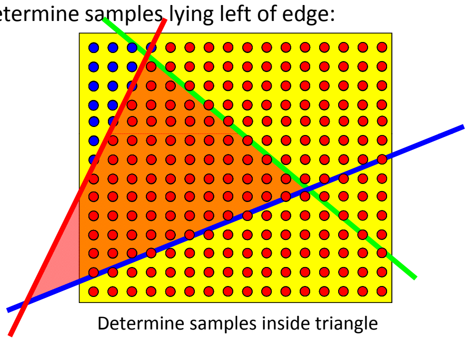
- The problem is 2D:



Determine samples inside triangle

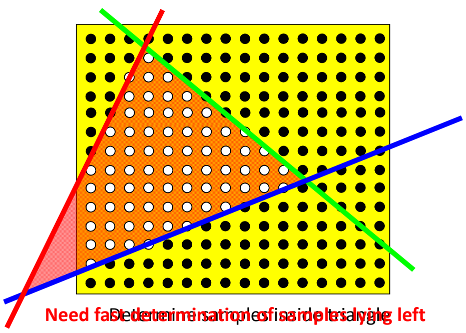
Blocked Samples

- Determine samples lying left of edge:



Determine samples inside triangle

Blocked Samples



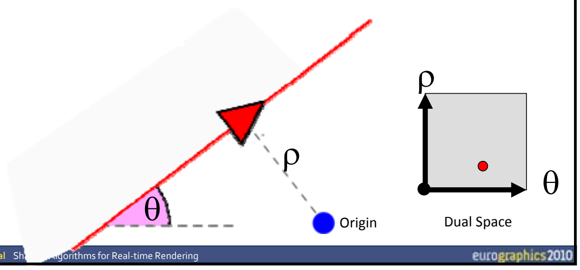
Need fast determination of samples lying left

Determine samples inside triangle

Hough Transformation

- Dual line representation:

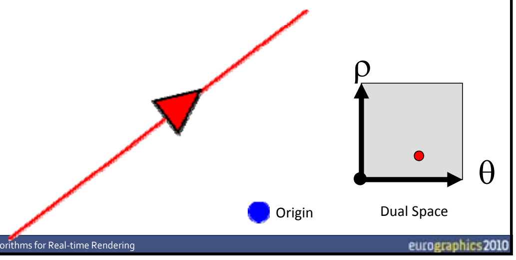
$$L_{\theta\rho}(\alpha) = (\cos\theta, \sin\theta)\,\alpha + \rho\,(-\sin\theta, \cos\theta)$$



Hough Transformation

- Dual line representation:

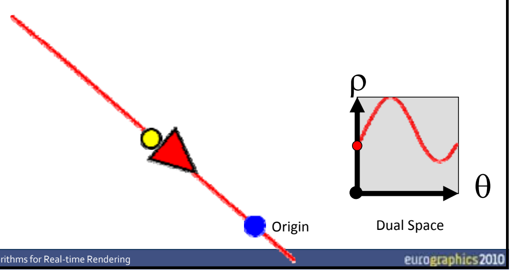
$$L_{\theta\rho}(\alpha) = (\cos\theta, \sin\theta)\,\alpha + \rho\,(-\sin\theta, \cos\theta)$$



Hough Transformation

- Dual line representation:

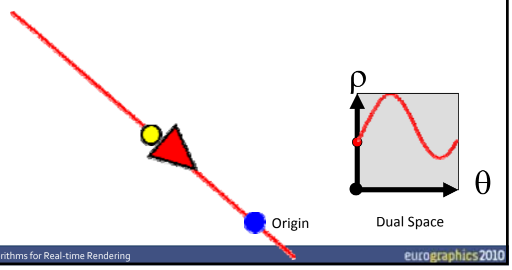
$$L_{\theta\rho}(\alpha) = (\cos\theta, \sin\theta)\,\alpha + \rho\,(-\sin\theta, \cos\theta)$$



Hough Transformation

- Dual line representation:

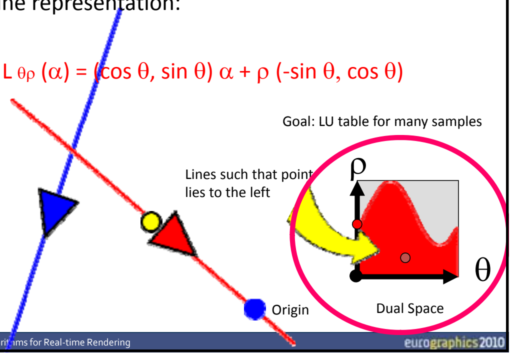
L θρ (α) = (cos θ, sin θ) α + ρ (-sin θ, cos θ)



Hough Transformation

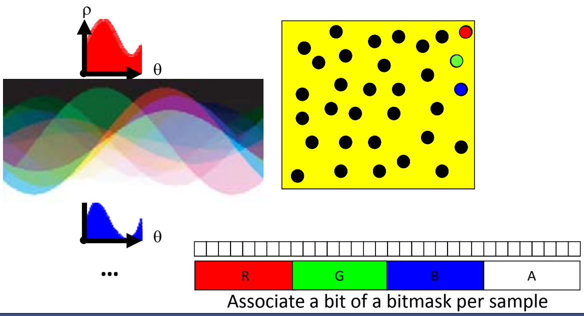
- Dual line representation:

L θρ (α) = (cos θ, sin θ) α + ρ (-sin θ, cos θ)



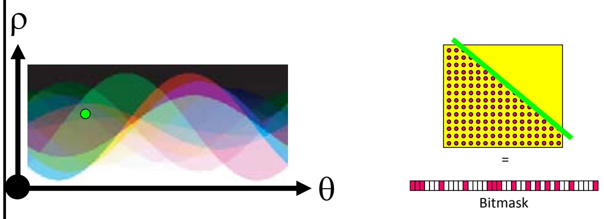
Lookup Table

- LU table for fast sample test:



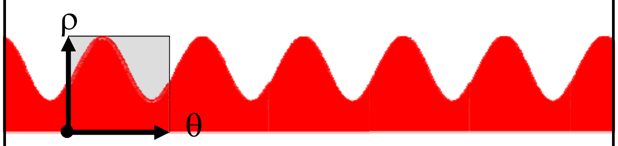
Lookup Table

- LU table for fast sample test:



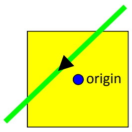
Lookup Table

- Store an unlimited space in a texture:



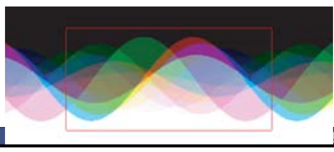
Dual Space

- Periodic in angle
- Clamp if *distance* > *patch size* (patch center = origin)



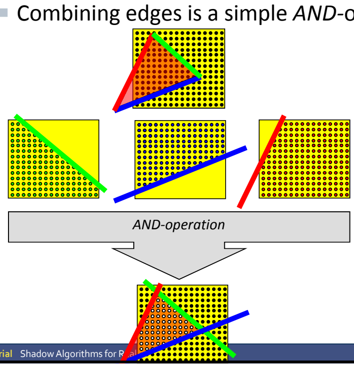
Lookup Table stored as Texture

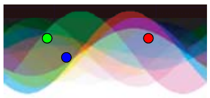
- Clamp mode for distance (virtually scale patch to uniform size)
- Repeat mode for angle
- Binary info: (128 bits per texture ~1024=8*128 possible) „one lookup“ = result for all samples



Blocked Samples in Triangle

- Combining edges is a simple *AND*-operation





Accurate Real-Time Shadows

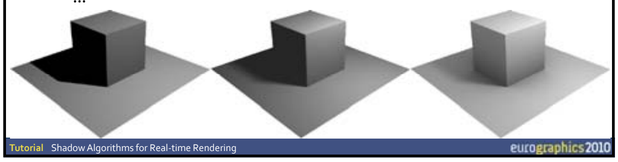
Accurate View-Sampling Methods

\+ Fastest solution for accurate soft shadows
\+ Artifact free
\+ Physically correct (enough)

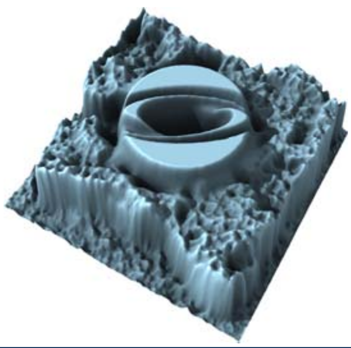
\- Relatively slow

What will happen in the future?

- Soft shadows will "extend"
- More general sources:
  - Entire environments
  - Many surfaces
  - Many point lights
  - …



Shadows will still be needed!



Environmental Lighting

- Environment lighting received much attention

Two examples:

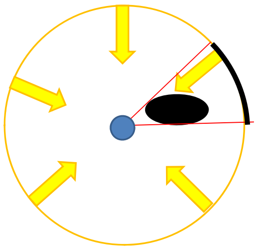
- All-frequency Soft Shadows [Annen et al. 2008]
- Ambient Occlusion [Miller 1994]

All-frequency Soft Shadows [Annen et al. 2008]

- General option:
  - Approximate environment map with several area lights

Ambient Occlusion [Miller 1994]

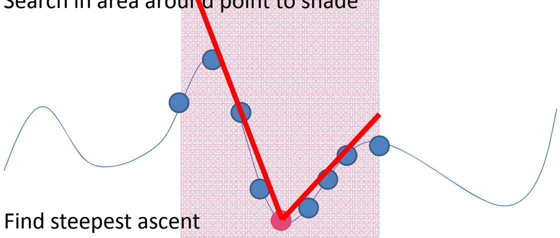
- Simpler Situation: Uniform lighting from environment



- Costly if solved via ray-tracing, alternatives exist
  see survey: [Méndez-Feliu and Sbert, 2009]

Screen-Space Ambient Occlusion
[Crytek2007, Bavoil&Sainz2008]

- Work directly on the depth buffer from viewpoint
- Search in area around point to shade



- Find steepest ascent
  -> defines cone that in turn defines light accessibility

Screen-Space Ambient Occlusion

+ Surprisingly good results
+ Relatively fast

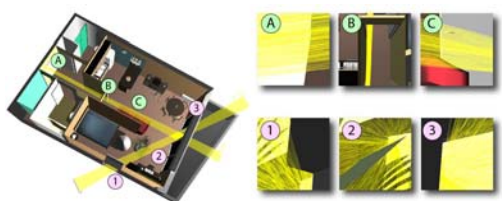
- Inconsistencies

Environmental Lighting: Summary

- Uniform illumination: simple solutions work well
  - It is a good complement to soft shadows
- Complex lights: Next big challenge
- New applications for visibility algorithms?

E.g., Level Design based on Visibility

- [Eisemann and Décoret EG 2007]



Now, it is up to you…

Thank you very much for your attention!

Many thanks go to:

- **T. Annen**, **P. Beckaert**, **Z. Dong**, **E.Flerakers**, **J. Kautz**, **T. Mertens**, **H.-P. Seidel** for providing figures and animations that helped illustrate PCF, CSM, and ESM
- **T. Akenine-Möller** for illustrations on SMs.
- **L. Bavoil** and **M. Sainz** for sharing the SSAO code and particularly Louis for many discussions
- **B. Lloyd** for discussions and some figures for the text
- **L. Baboud** for the height-field illustration
- **Betuca** for the dessert shadow photos
- **H. Bezerra** for discussions and support

Thank you very much for your attention!