3D Computer Graphics and Animation

From Soft Shadows to Radiosity

How to stop being direct

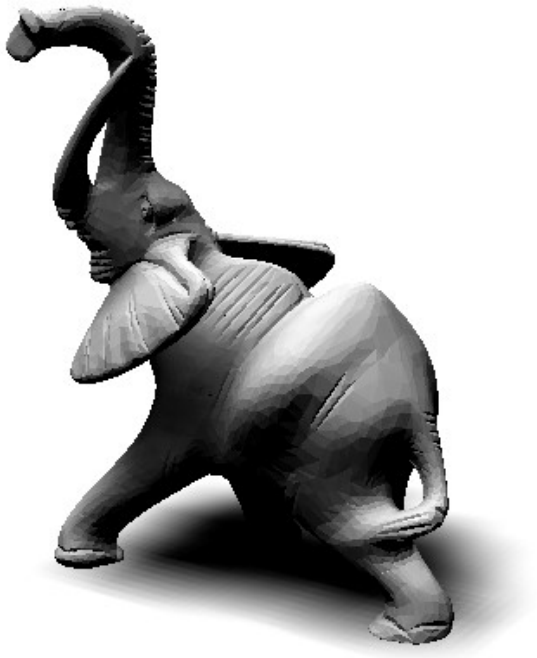
Elmar Eisemann

Delft University of Technology

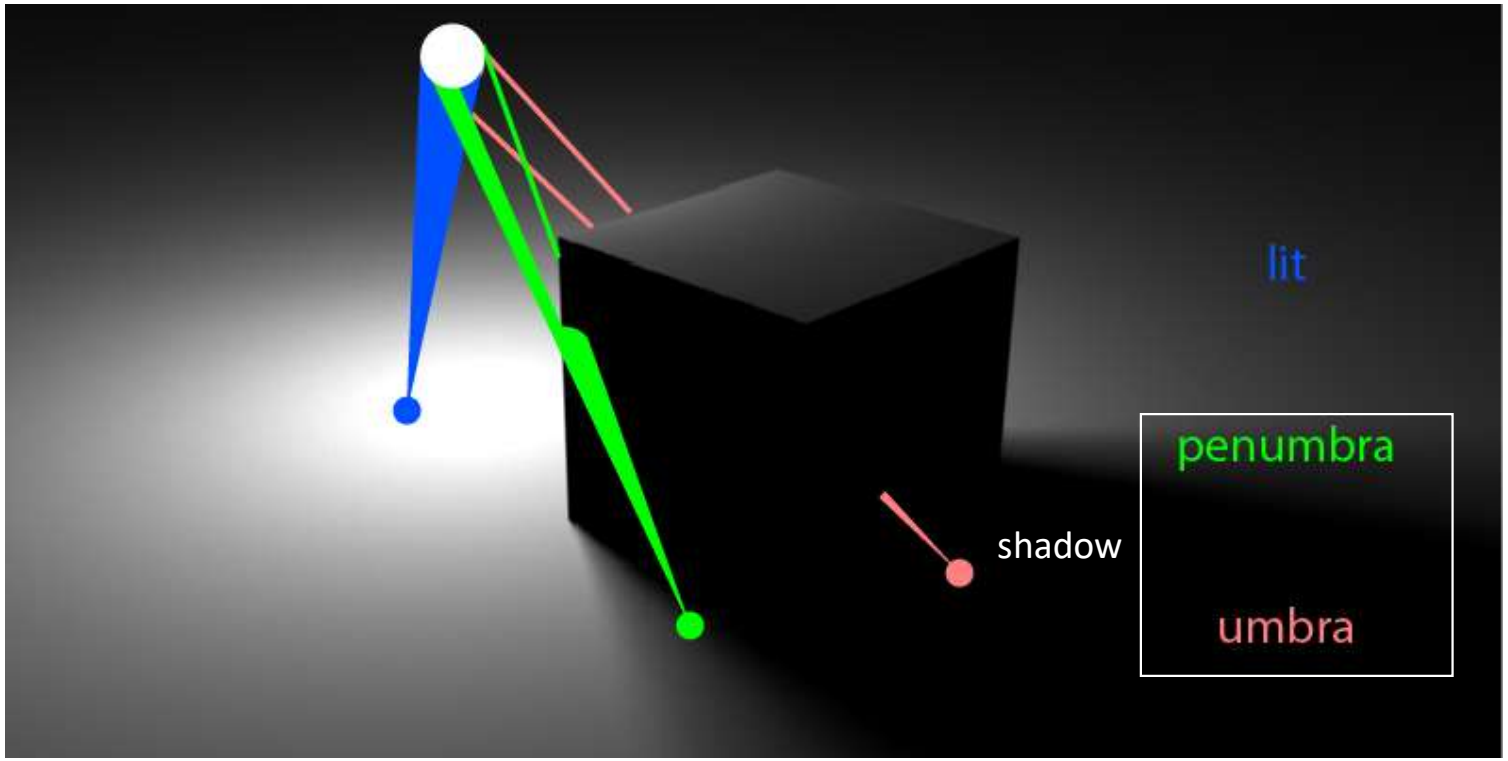
Last time…

- Percentage-Closer Filtering

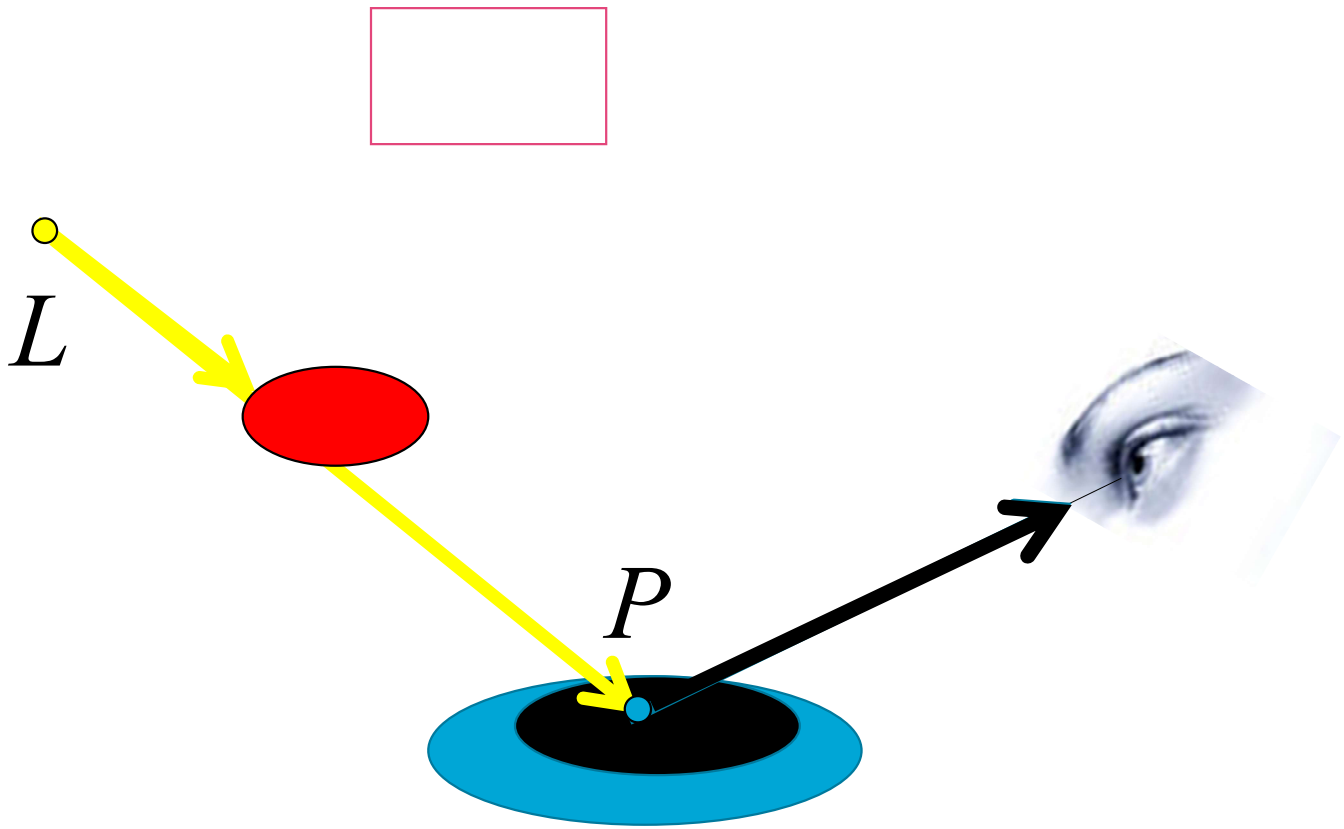
  is NOT the same as computing soft shadows…



What is a Shadow?



How to compute shadows?



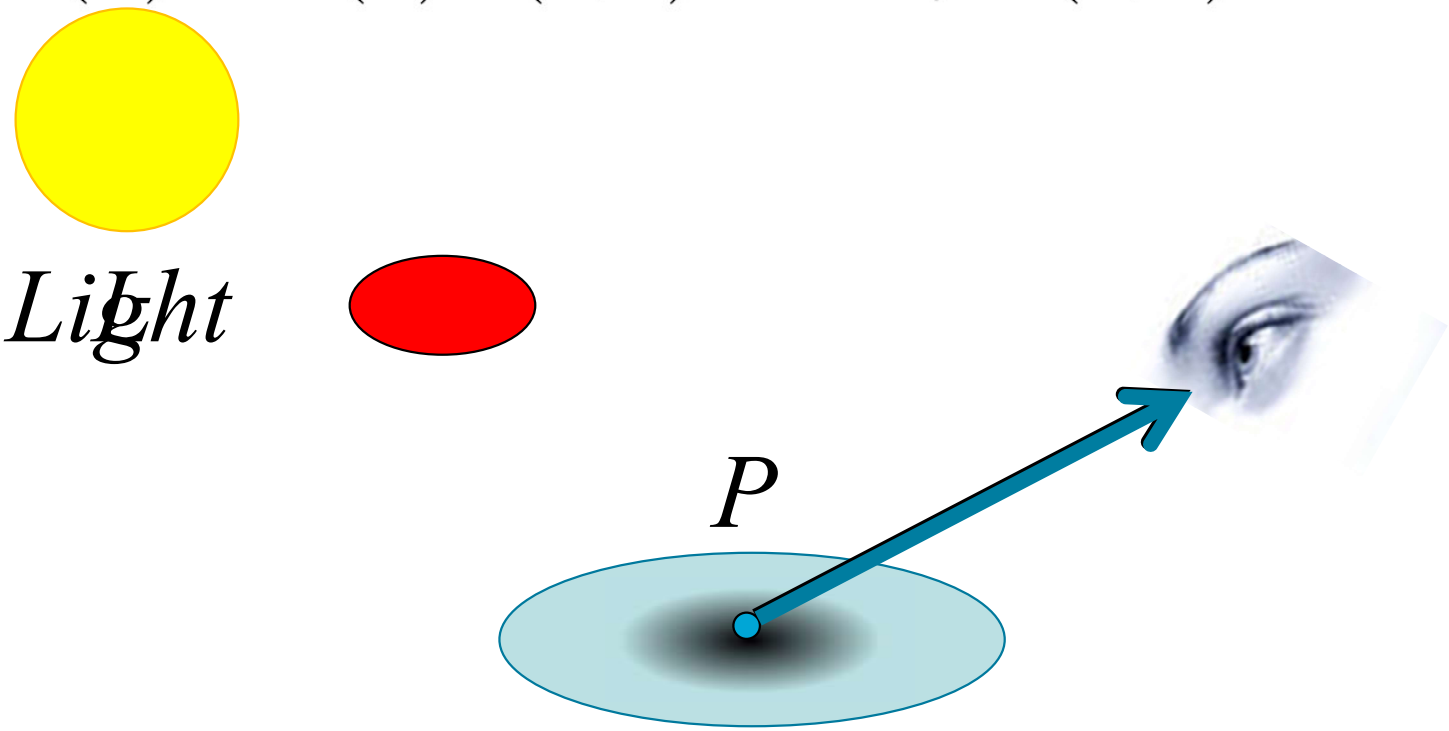
Hard Shadows

Point lights do not create penumbrae

Soft Shadows

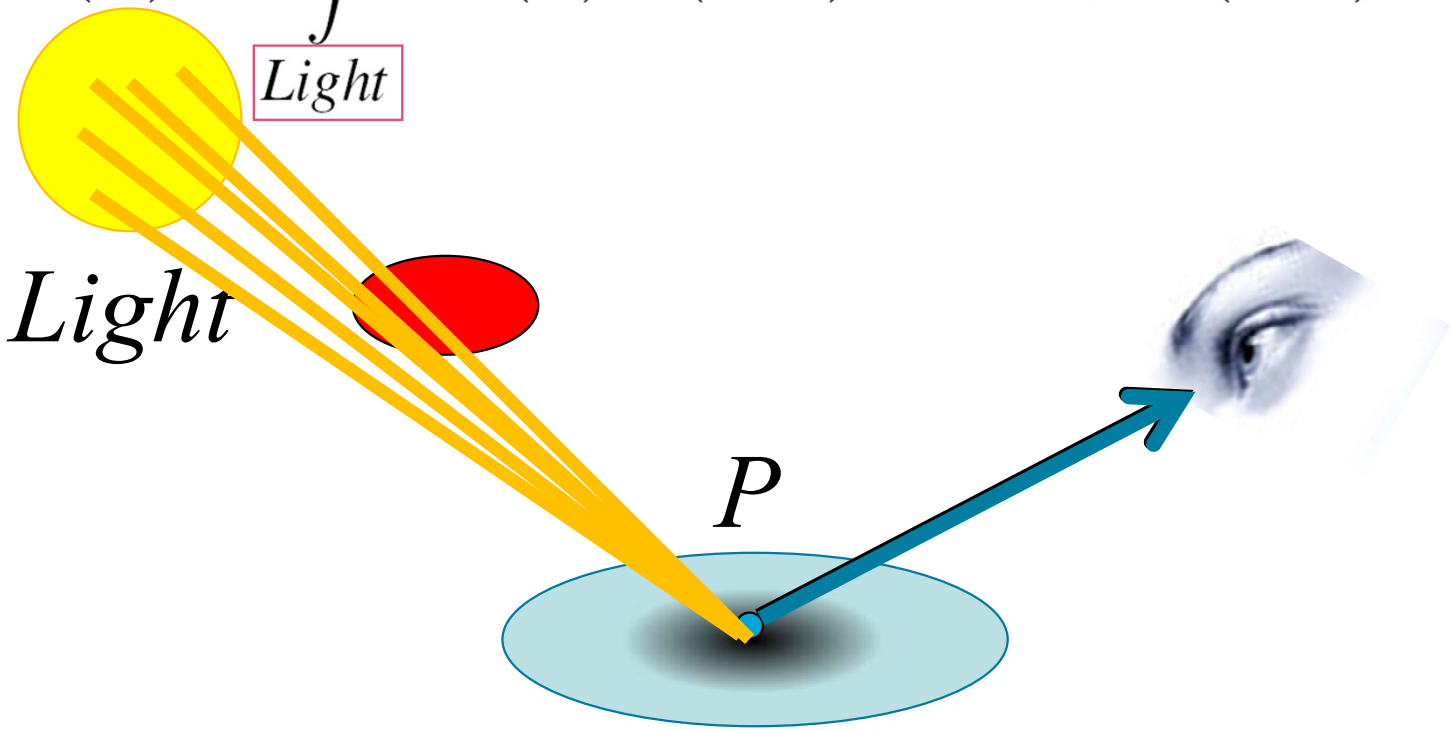
What is a Shadow ? – Part II

$$B(P) = E(L)\ v(P,L)\ Transfer(P,L)$$



What is a Shadow ? – Part II

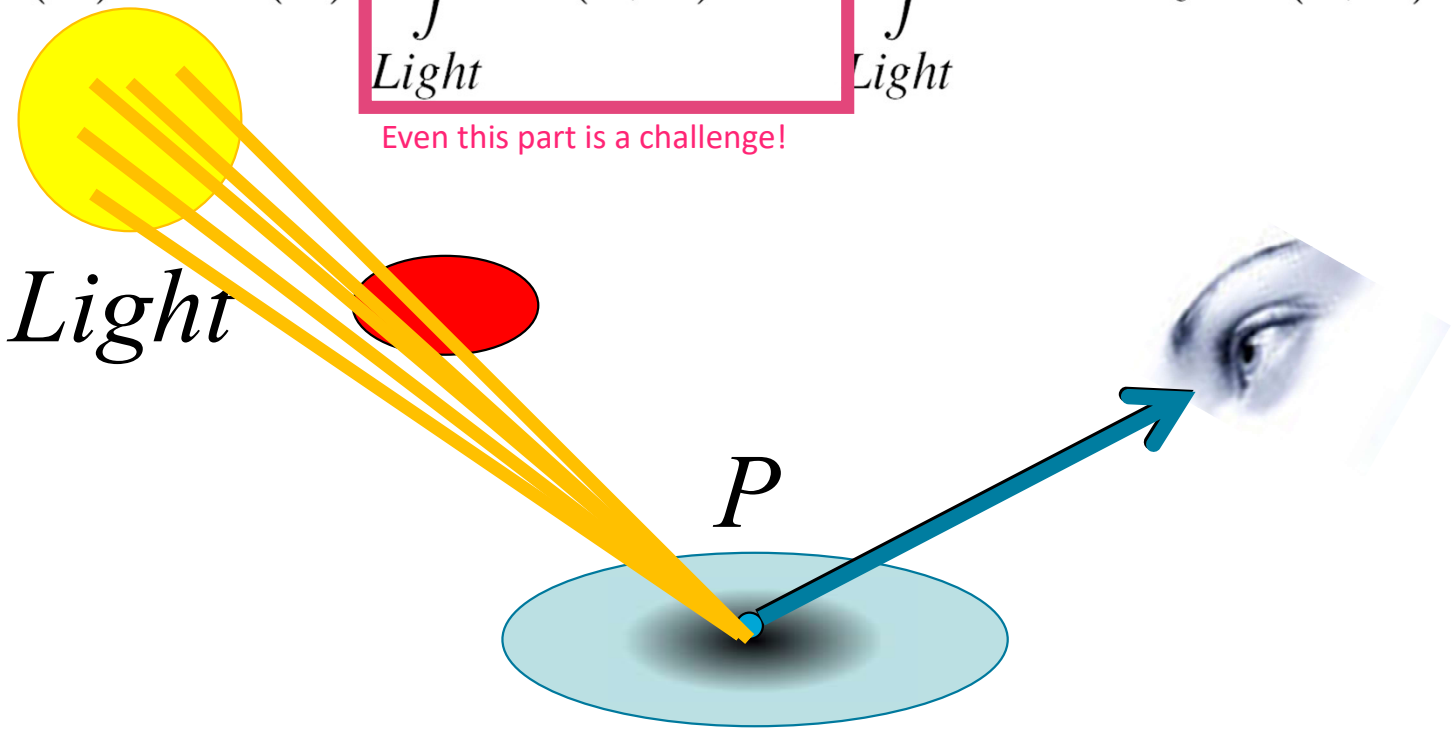
$$B(P) = \int_{Light} E(L)\ v(P,L)\ Transfer(P,L)\ dL$$



What is a Shadow ?

$$B(P) \approx E(L) \int_{Light} v(P,L)\, dL \int_{Light} Transfer(P,L)\, dL$$

Even this part is a challenge!

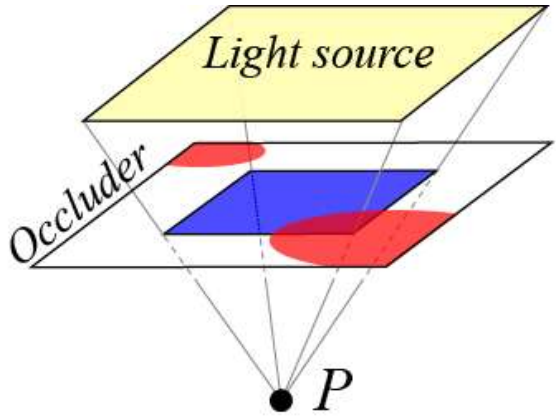


Let’s first focus on a simpler case!

Simpler Case

Simplifying assumption:

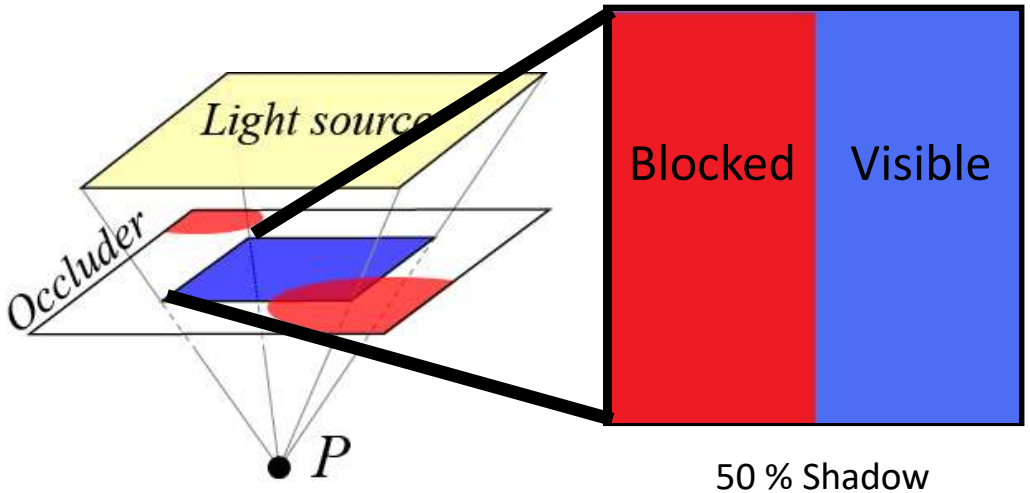
- Rectangular light source
- One planar aligned shadow caster



Occlusion Textures

Simplifying assumption:

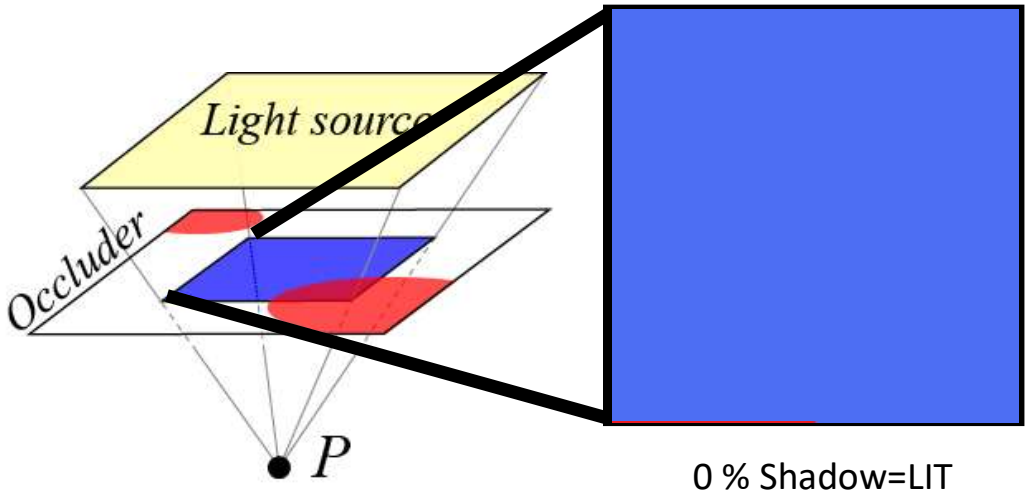
- Rectangular light source
- One planar aligned shadow caster



Occlusion Textures

Simplifying assumption:

- Rectangular light source
- One planar aligned shadow caster

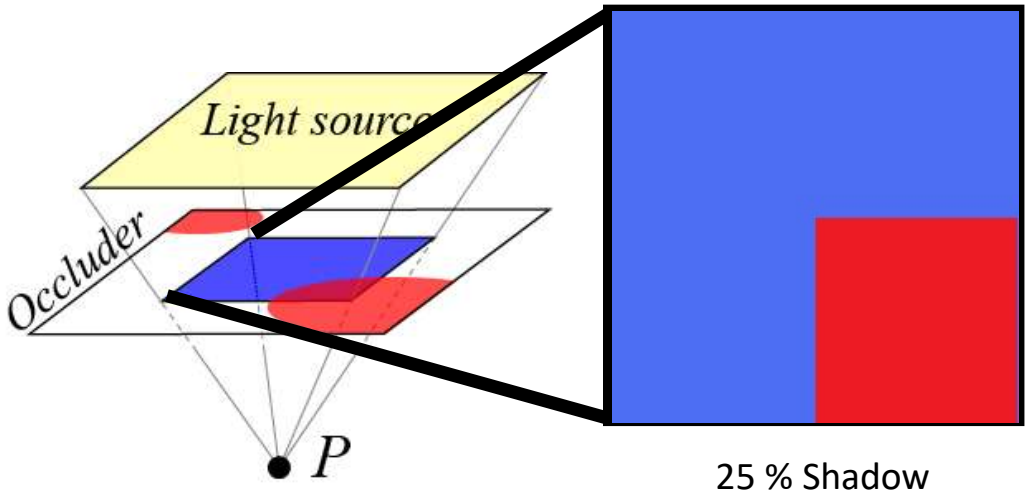


0 % Shadow=LIT

Occlusion Textures

Simplifying assumption:

- Rectangular light source
- One planar aligned shadow caster

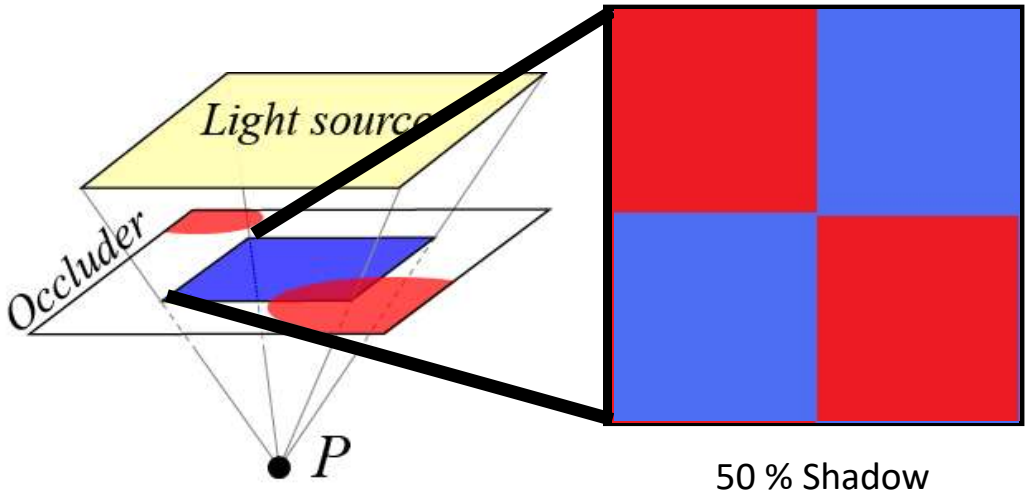


25 % Shadow

Occlusion Textures

Simplifying assumption:

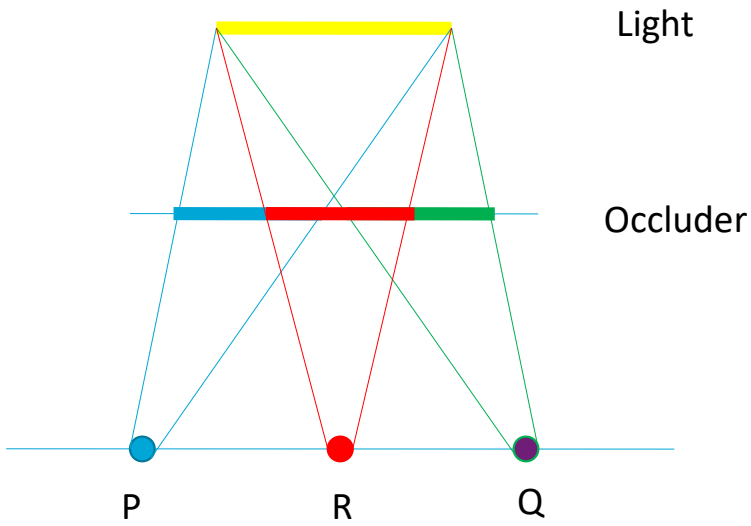
- Rectangular light source
- One planar aligned shadow caster



50 % Shadow

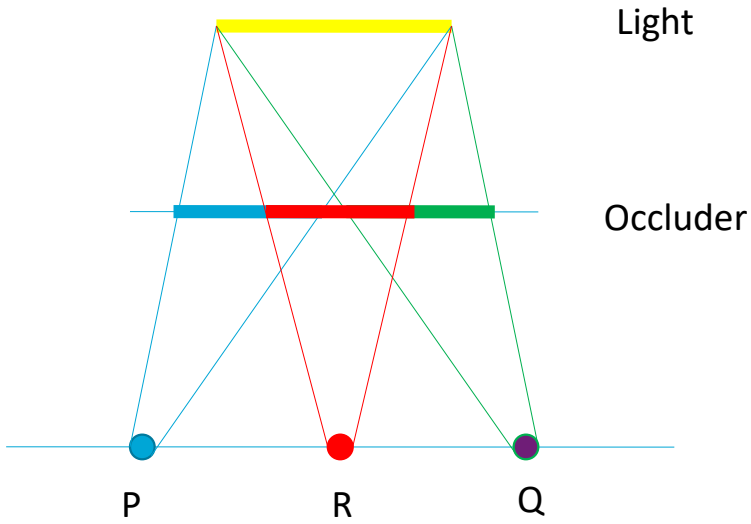
What happens if P moves?

- Movement parallel to light:



What happens if P moves?

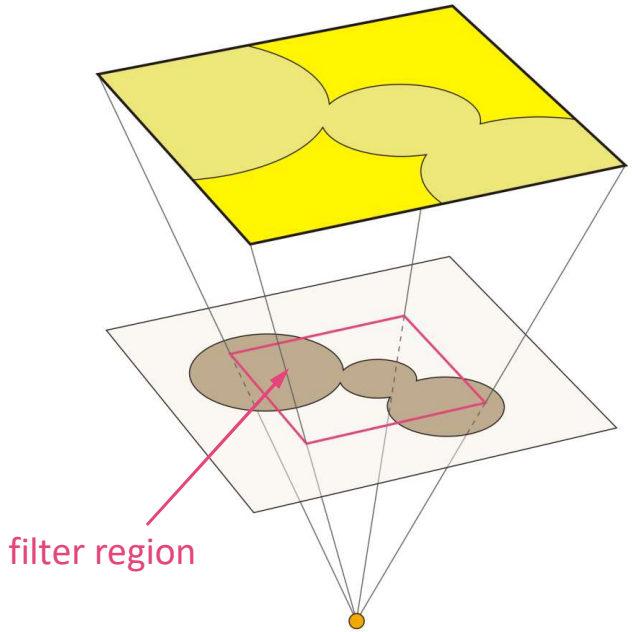
- For parallel movement, constant window size
  and linear position dependency with respect to receiver point



Special configuration = Convolution

- Planar occluder parallel to light
- Visibility is convolution:

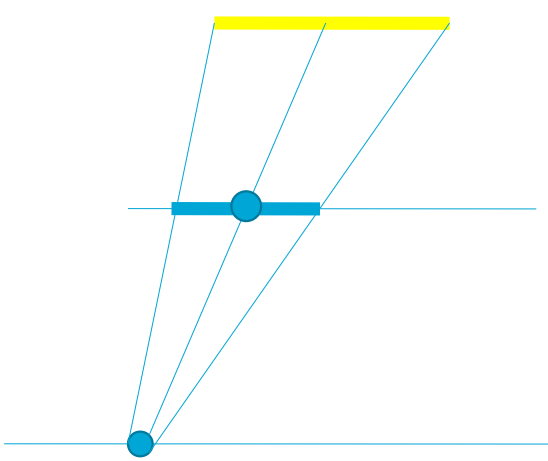
  Filter size = scaled light size



[Soler and Sillion, SIGGRAPH 1998]

What about a general point P?

- The average of the blockers in this window gives accurate soft shadows for this setup



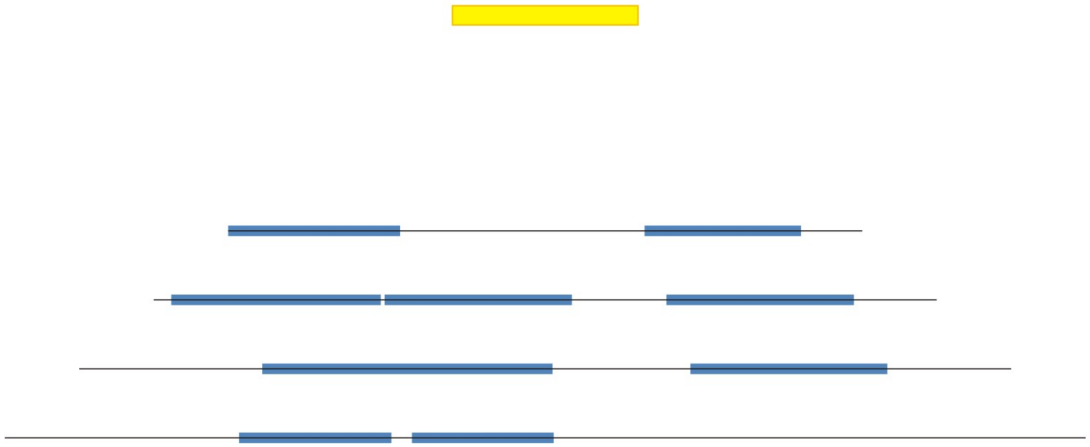
Average of a region in a texture?
We have heard this before…
MipMaps, NBuffer…

Occlusion Textures [Eisemann&Décoret2006]

- Rectangular light source = Box filter

Occlusion Textures [Eisemann&Décoret 2006]

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

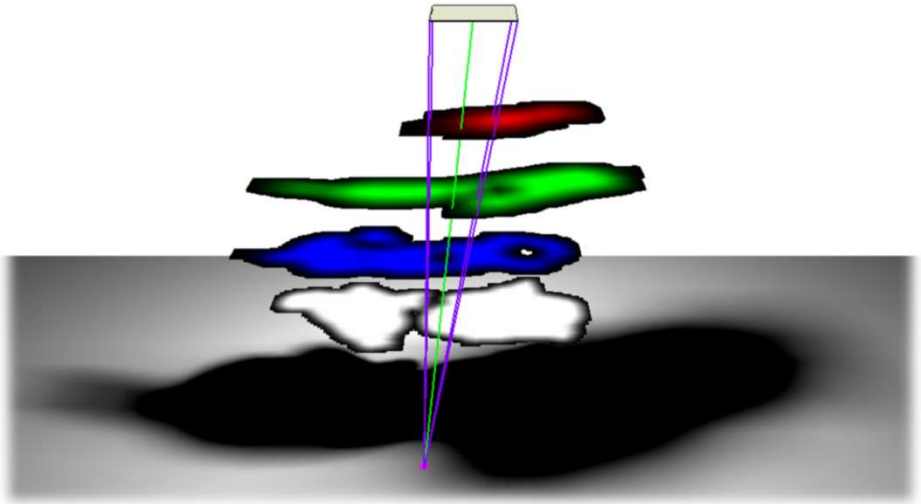


Occlusion Textures

- One efficient option to generate layers:
  Use the color channels as layers to create the representation by blending

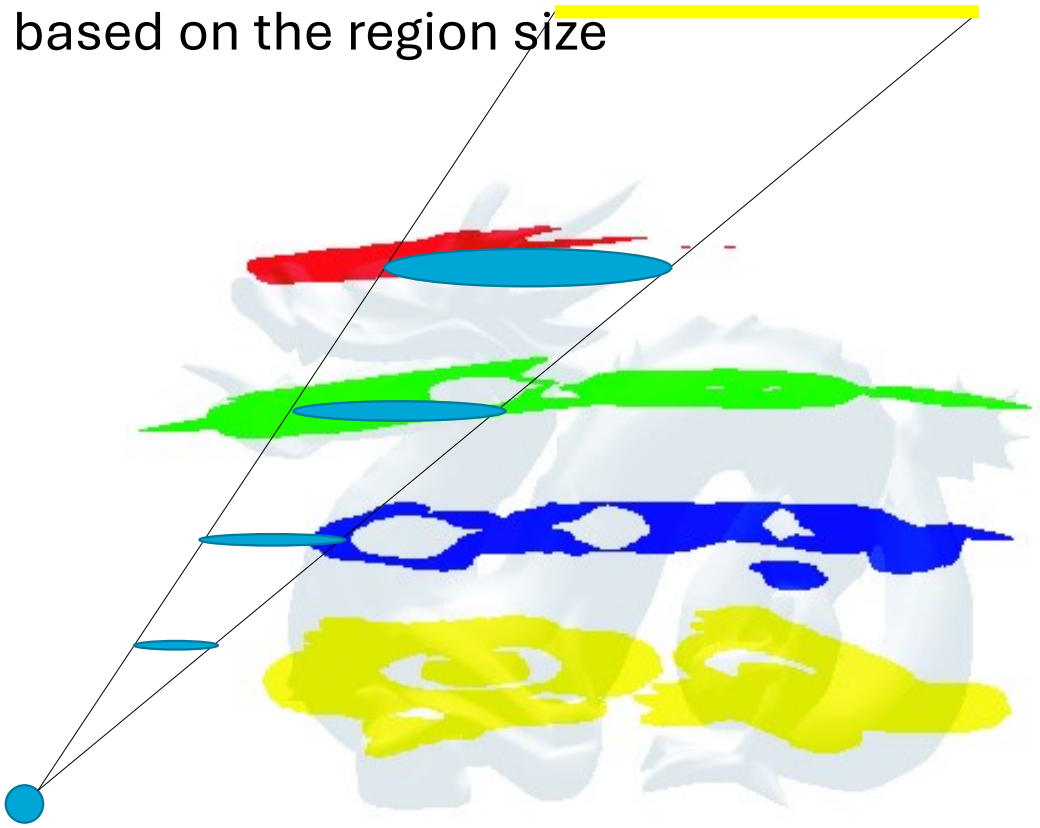
Occlusion Textures

- Prefilter the resulting occlusion textures
  - Mipmapping
  - N-buffers
  - …



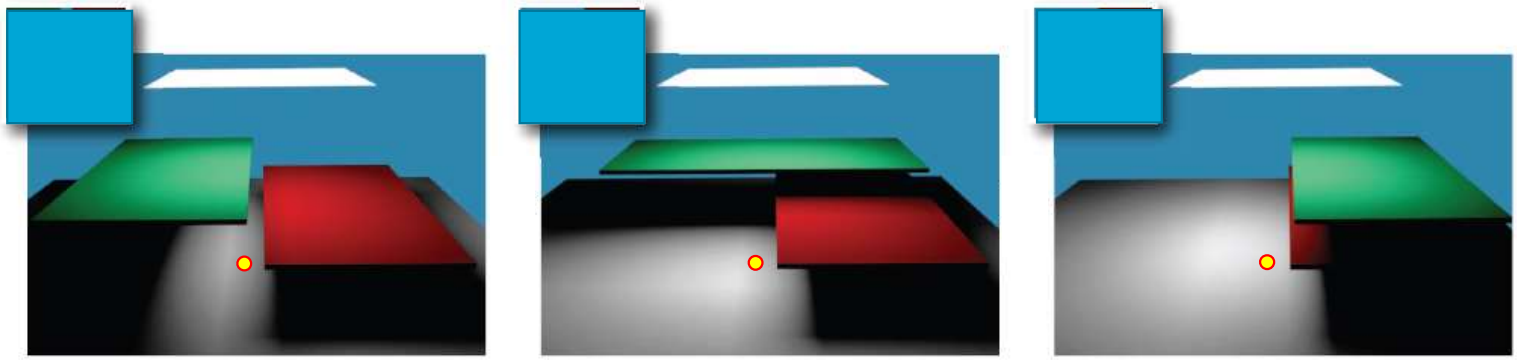
Occlusion Textures

- Adapt lookup based on the region size



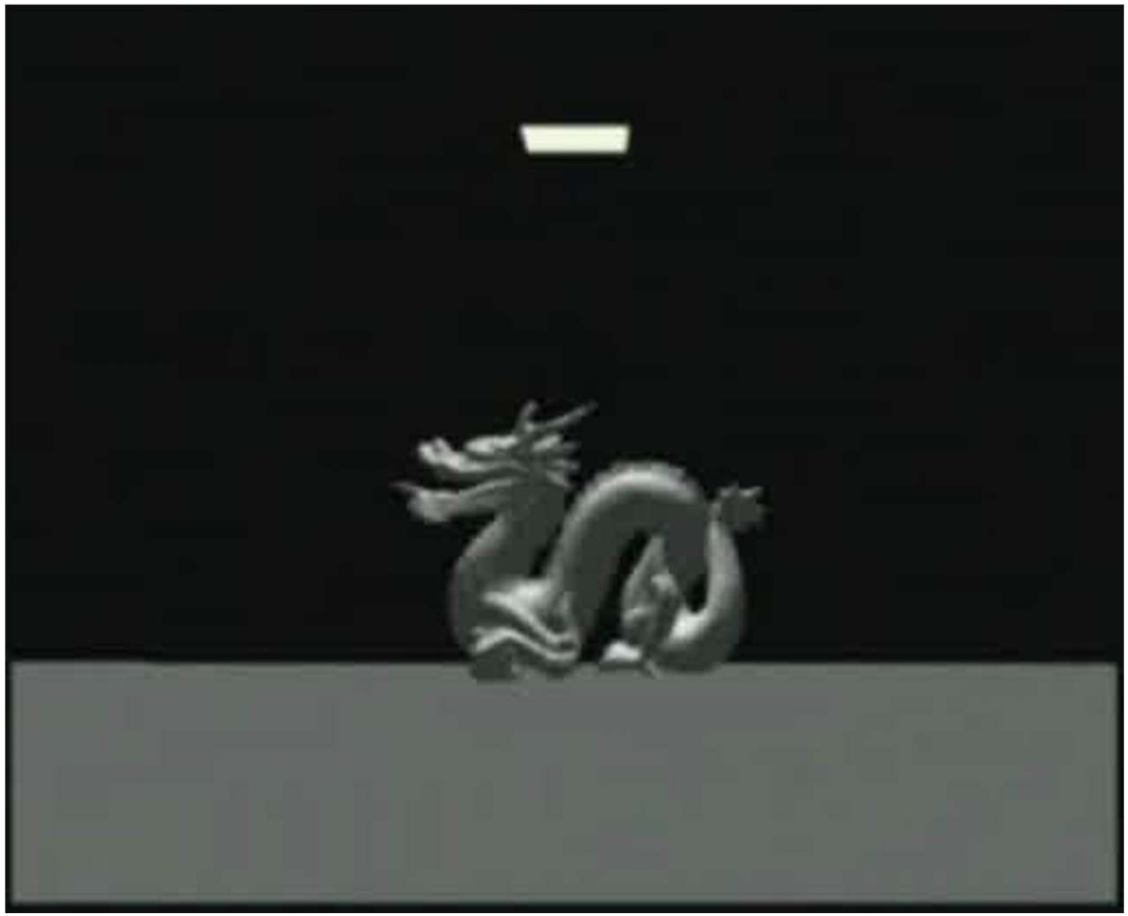
Problem: Occluder Fusion

- Solution is **NOT** accurate!
- We cannot correctly combine layers without more information.

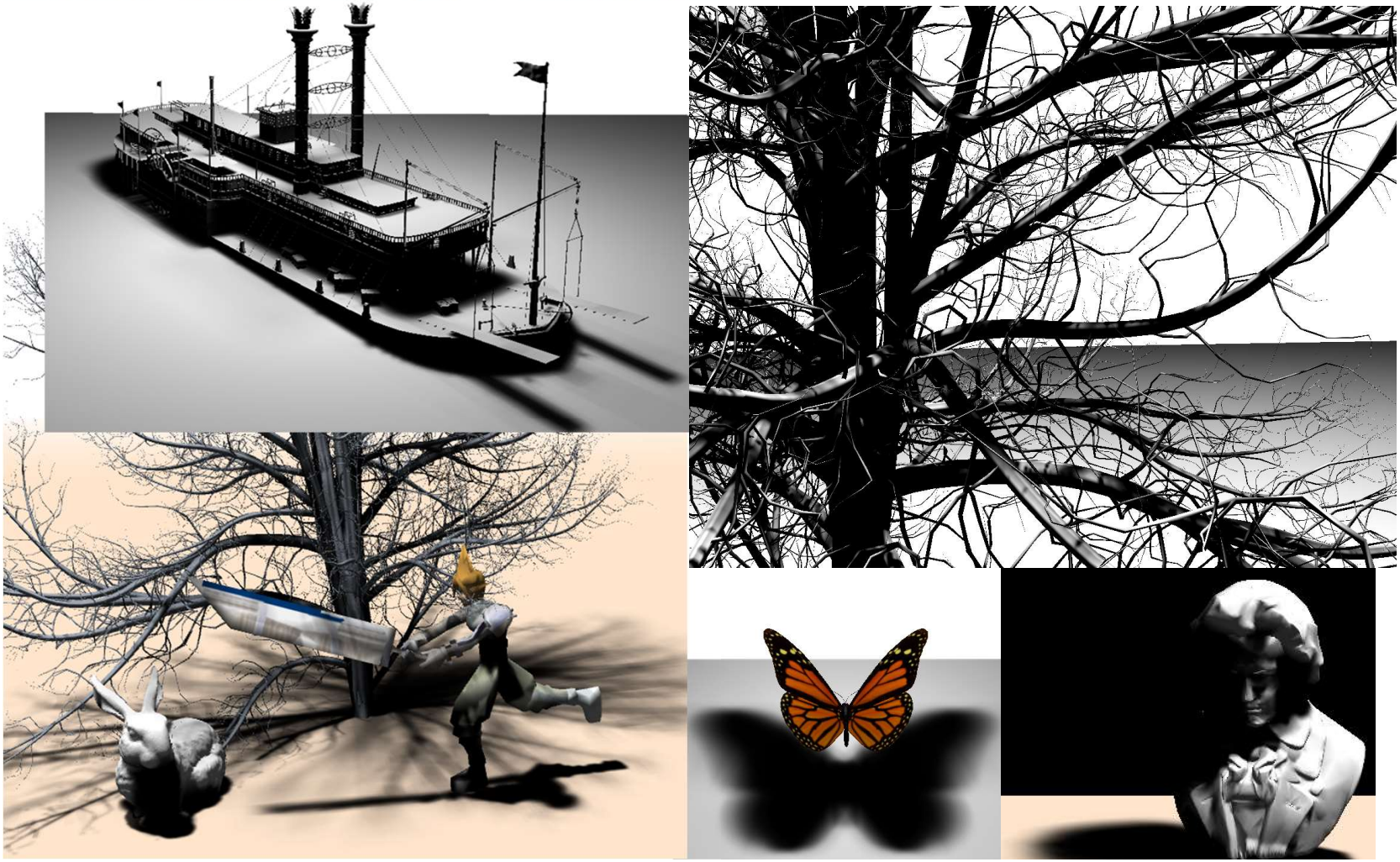


Proposed solution for Occlusion textures: Multiplication of all layer occlusions

Occlusion Textures



Occlusion Textures for Plausible Soft Shadows



Occlusion Textures for Plausible Soft Shadows

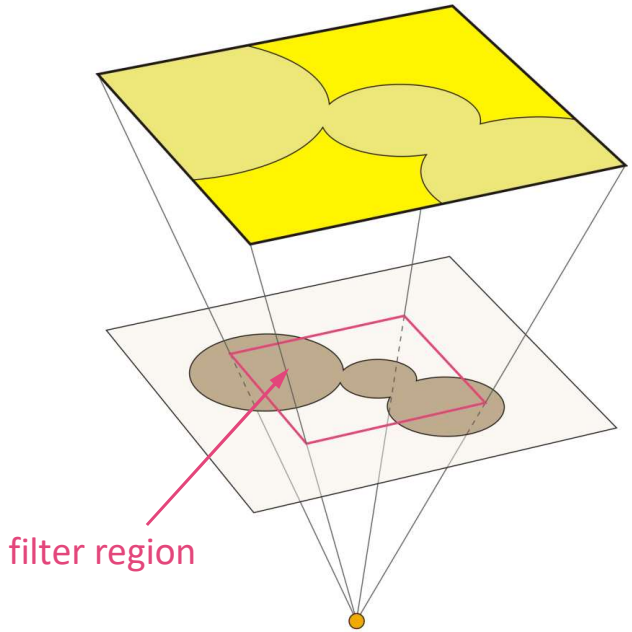
Occlusion Textures

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

- Mainly suited for small indoor scenes
- Heuristic layer handling
- Layer discretization can

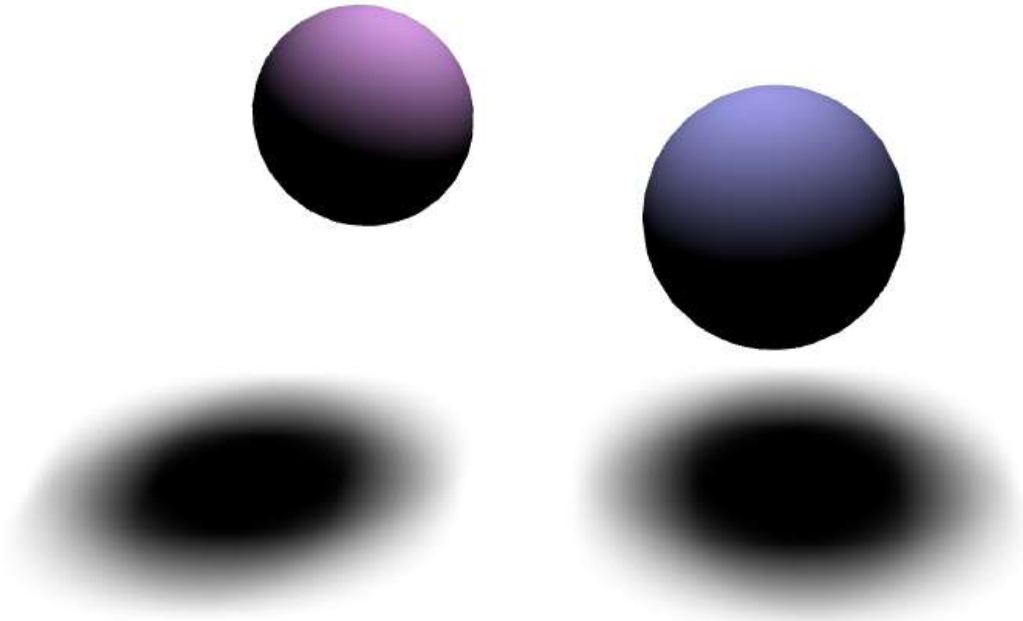
Can we also use Shadow Maps?

- Planar occluder parallel to light???



Last time: Percentage-Closer Filtering (PCF)

- Yields a smooth appearance
- BUT ignores varying penumbra width!

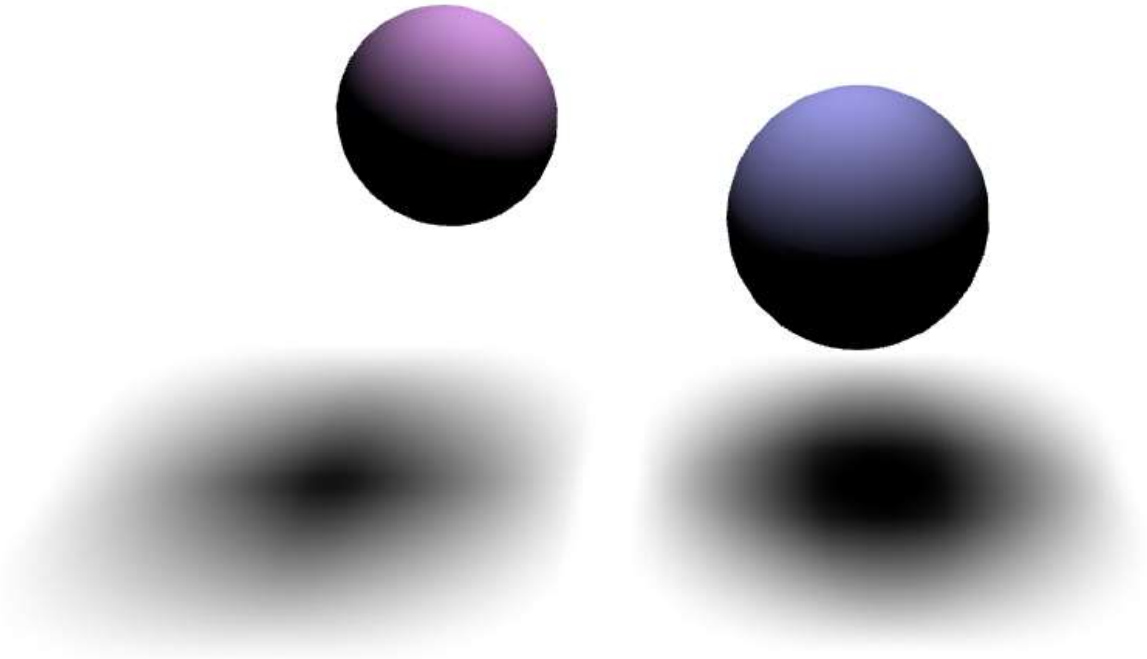


VSM, 512×512, 62×62

Plausible soft shadows with PCF?

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

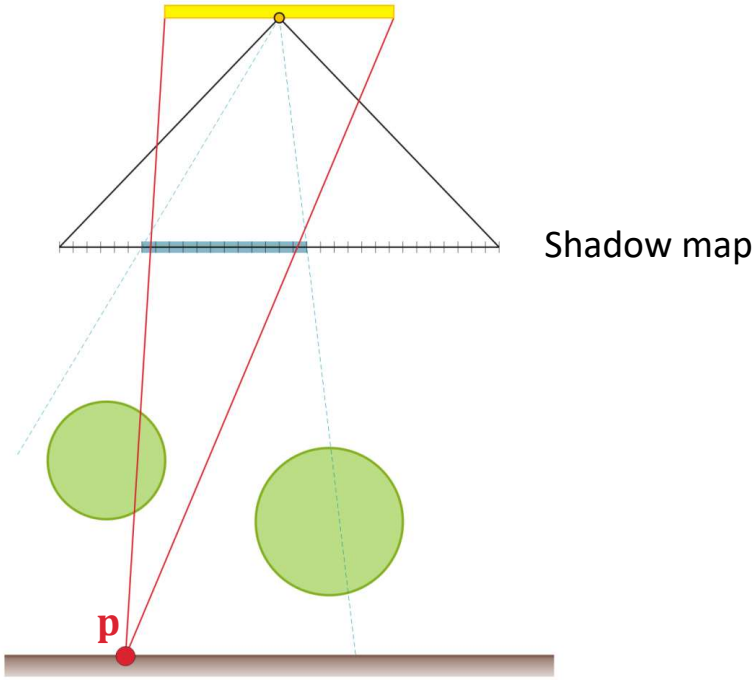
- Percentage-Closer Soft Shadows (**PCSS**)



PCSS

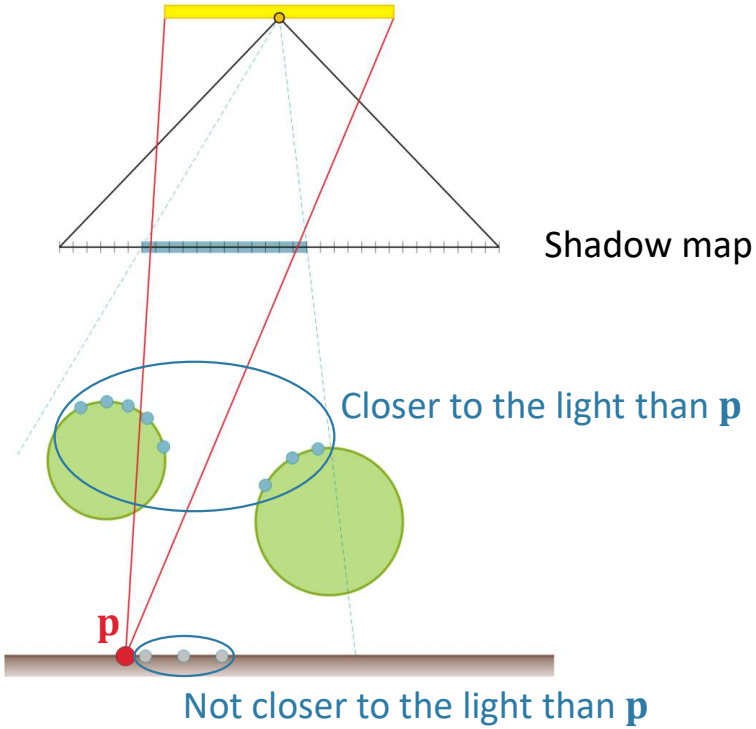
[Fernando, SIGGRAPH 2005 Sketch]

Percentage-Closer Soft Shadows



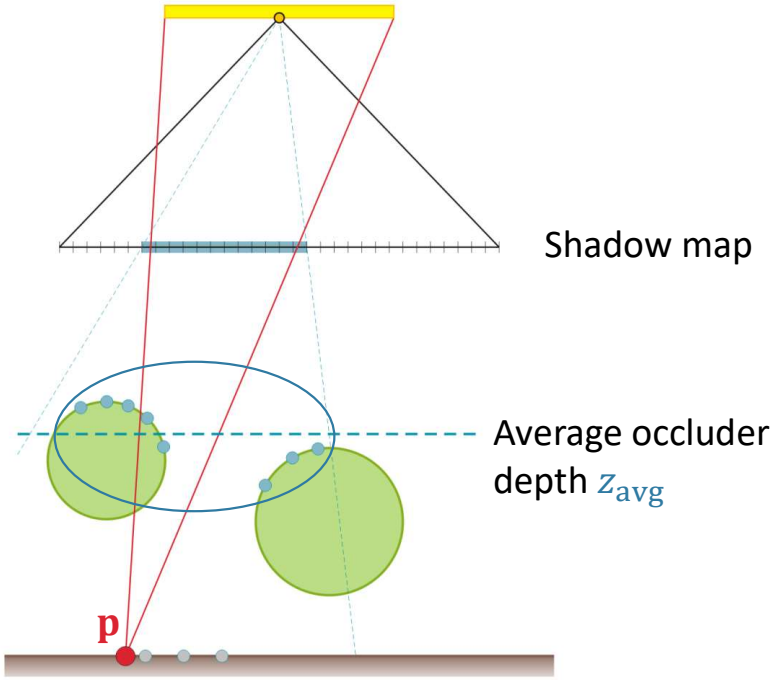
1. Blocker search

Percentage-Closer Soft Shadows



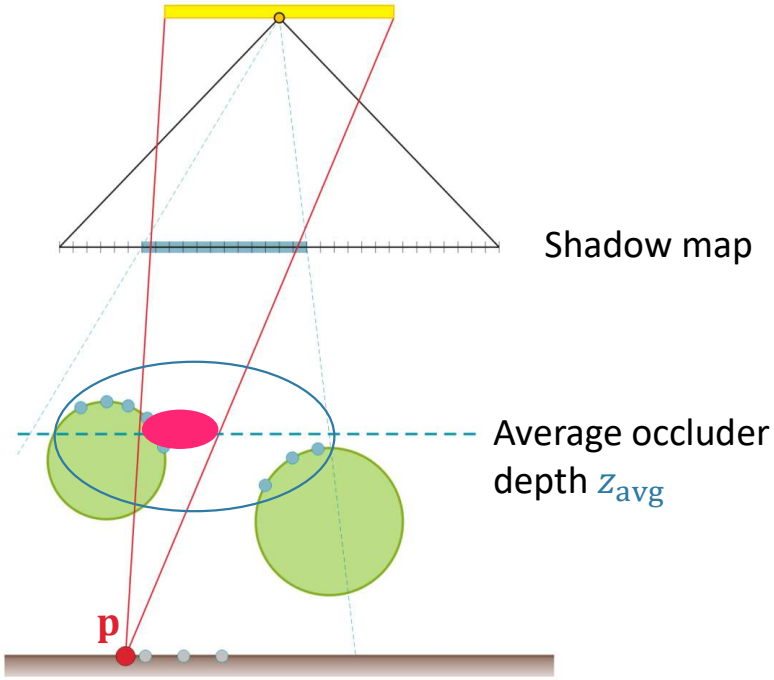
1. Blocker search

Percentage-Closer Soft Shadows



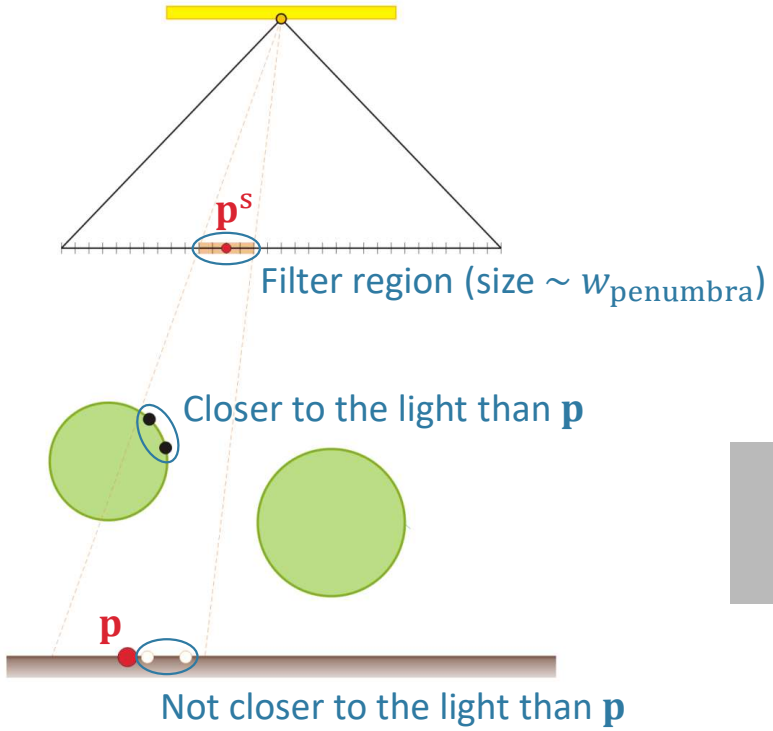
1. Blocker search

Percentage-Closer Soft Shadows



1. Blocker search

Percentage-Closer Soft Shadows



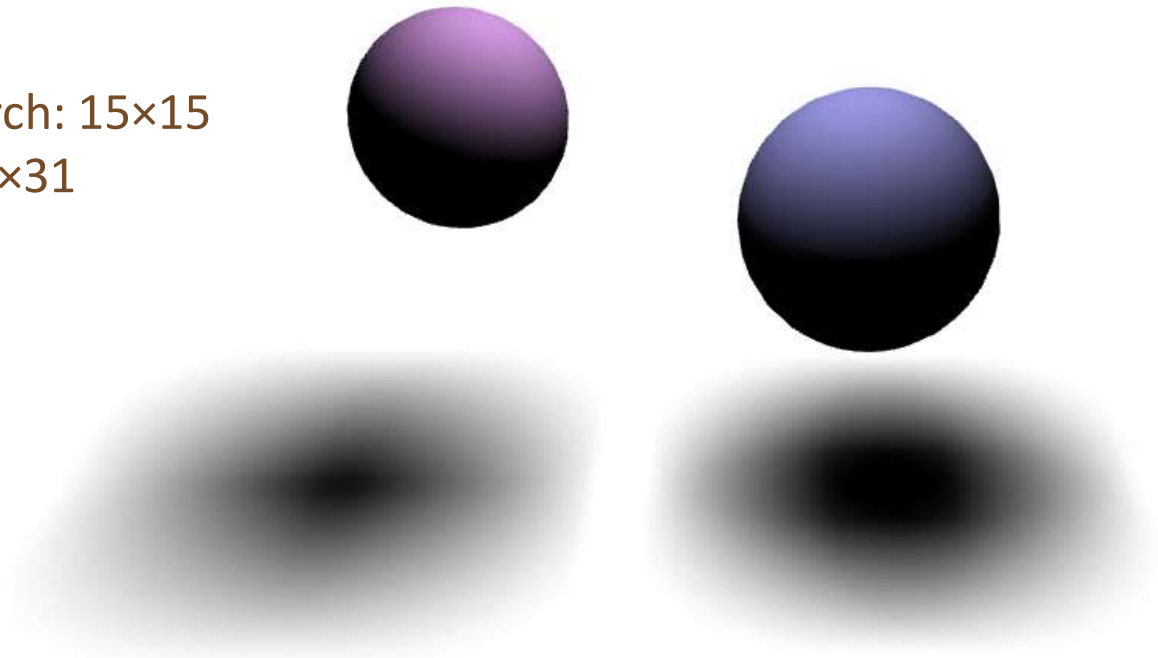
50%

3. Filtering

Percentage-Closer Soft Shadows

- Quality vs. number of shadow map samples

Blocker search: 15×15
Filtering: 31×31

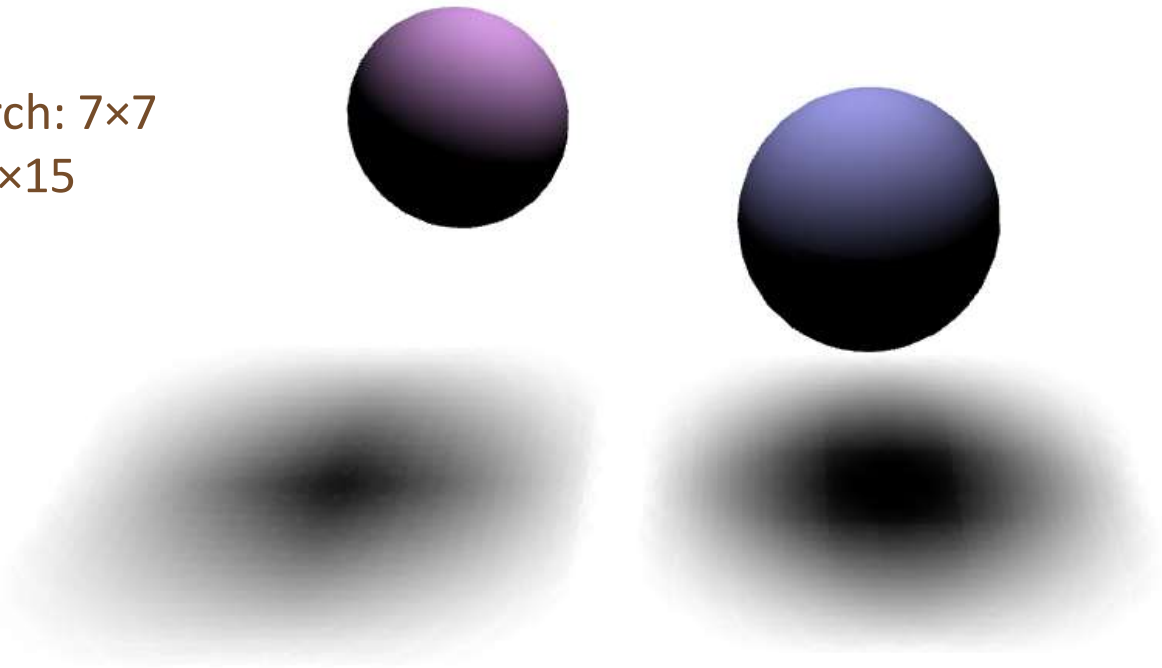


29 fps
(1024×1024, GeForce GTX 285)

Percentage-Closer Soft Shadows

- Quality vs. number of shadow map samples

Blocker search: 7×7
Filtering: 15×15



120 fps
(1024×1024, GeForce GTX 285)

Percentage-Closer Soft Shadows

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

Two of them require many shadow map accesses!

- Accelerations:
  - Subsampling
  - Prefiltering (CSM, ESM…)
    (e.g., [Annen et al. 2008], Convolution Soft Shadows)

Percentage-Closer Soft Shadows

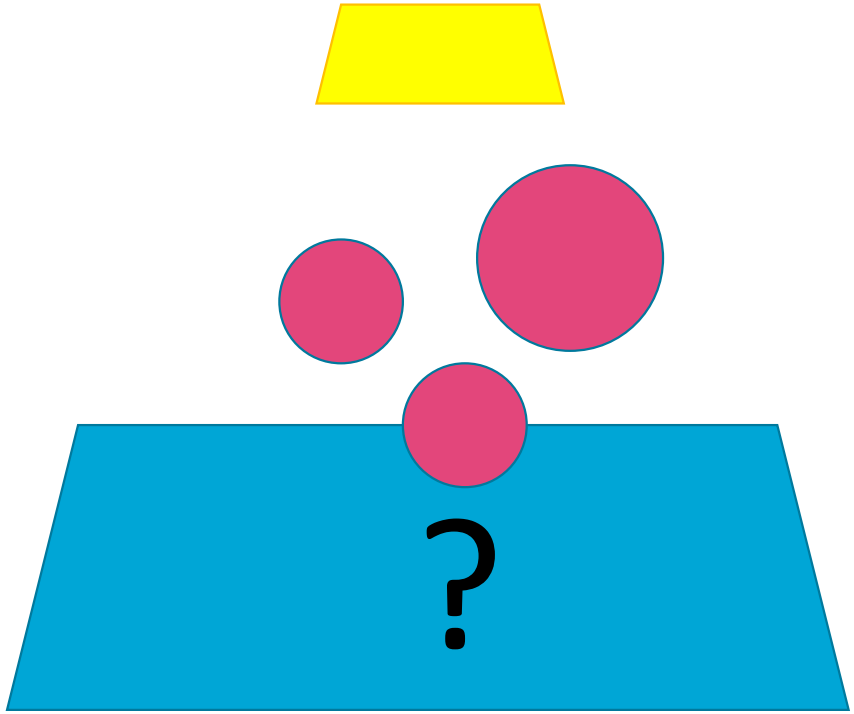
+ Simple and reasonably fast
+ Often visually pleasing results
  (at least for smaller light sources)
+ Applied in practice

– Not really physically plausible
– Only occluders involved that are visible from light center

How to accurately compute soft shadows?

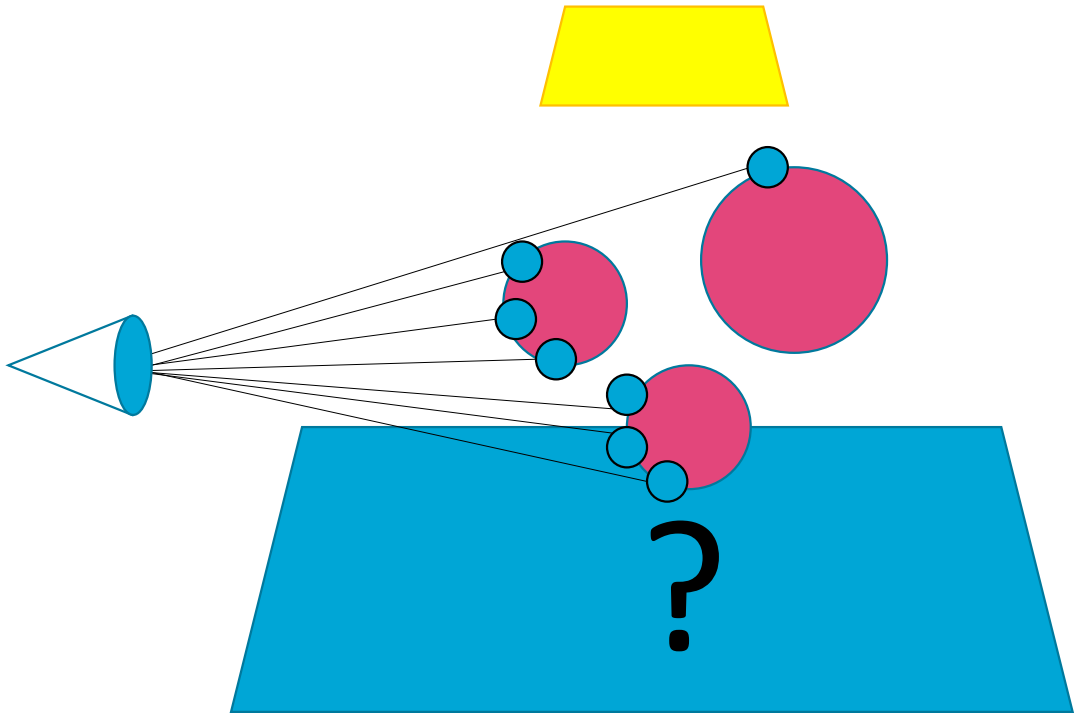
Simpler Problem

- What are the shadows on the plane?



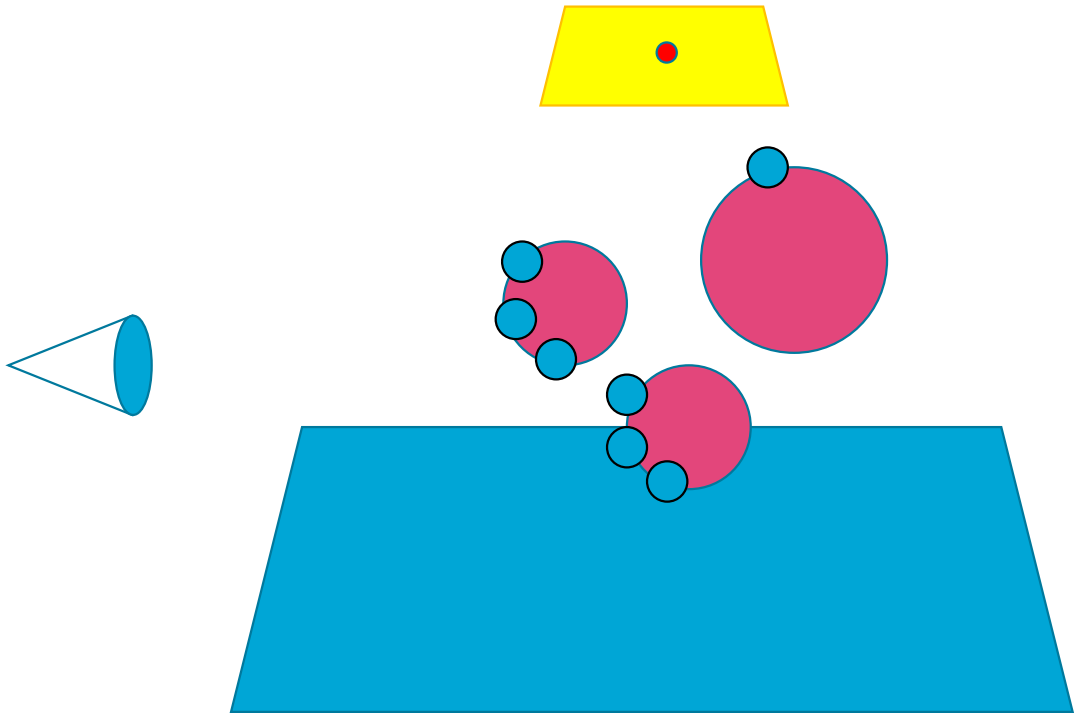
Simpler Problem

- What are the shadows on the plane?



Simpler Problem

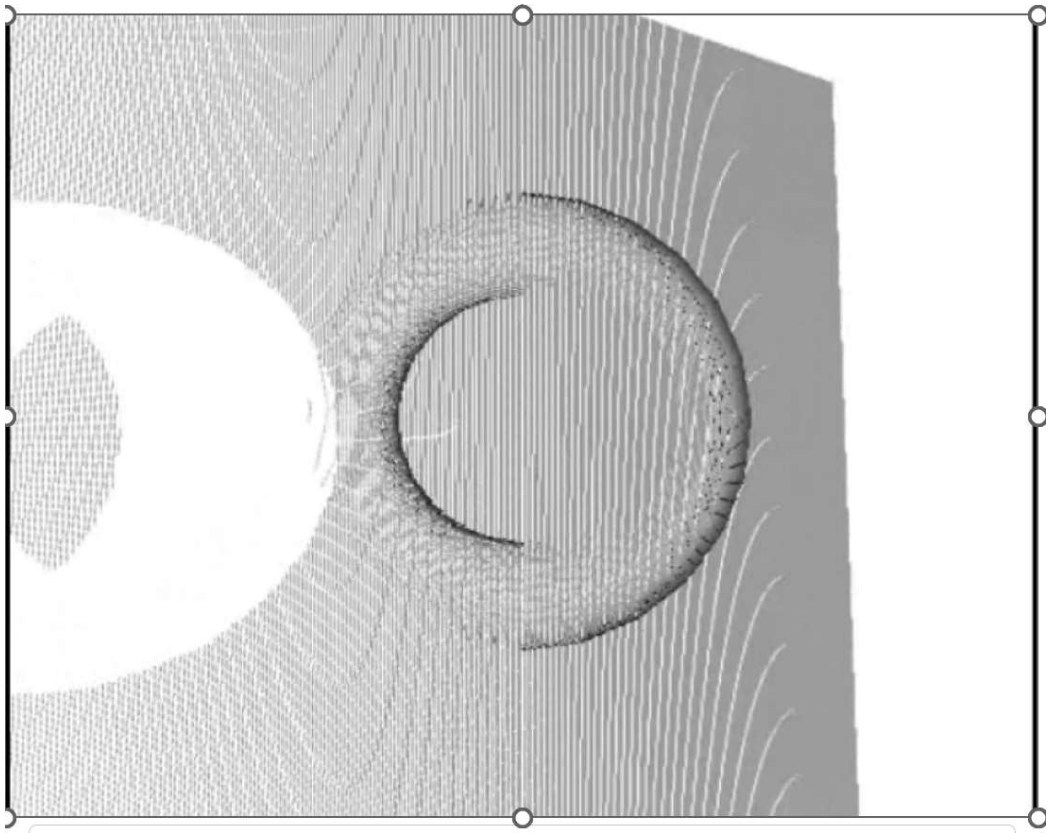
- What are the shadows on the plane?



Excursion: General Scenes

[Sintorn et al. EGSR08]

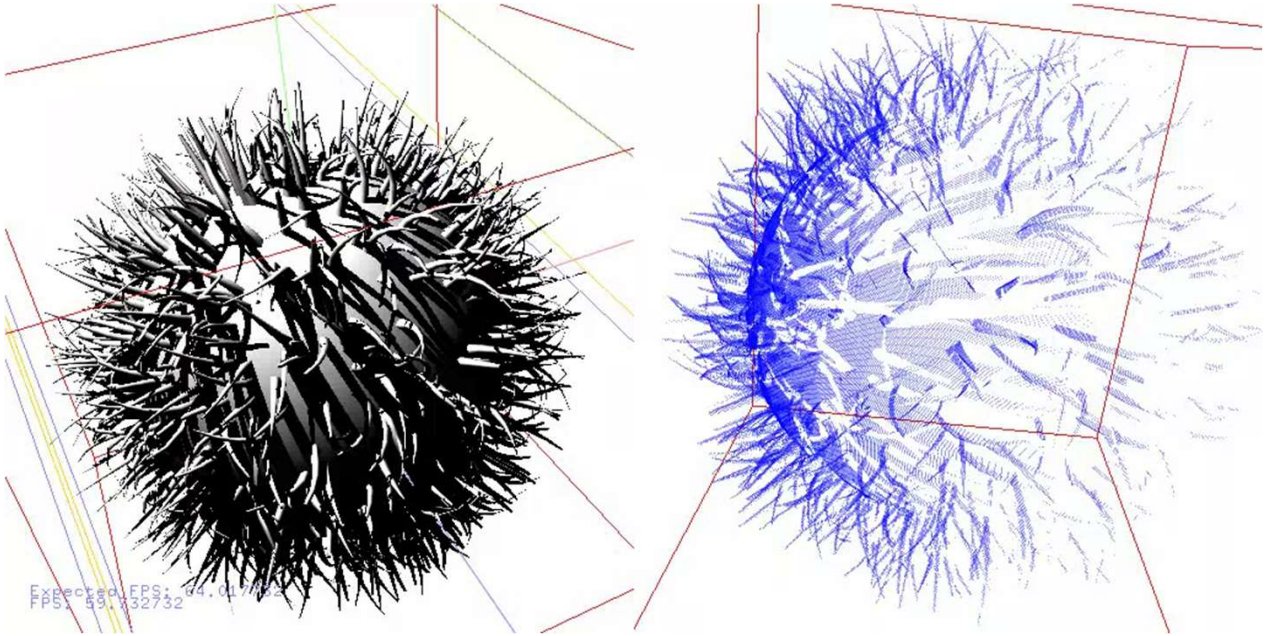
Excursion: General Scenes



- Let’s assume each shadow map pixel only contains one screen pixel
- Then render triangle and mark all covered screen pixels as black

Excursion: General Scenes

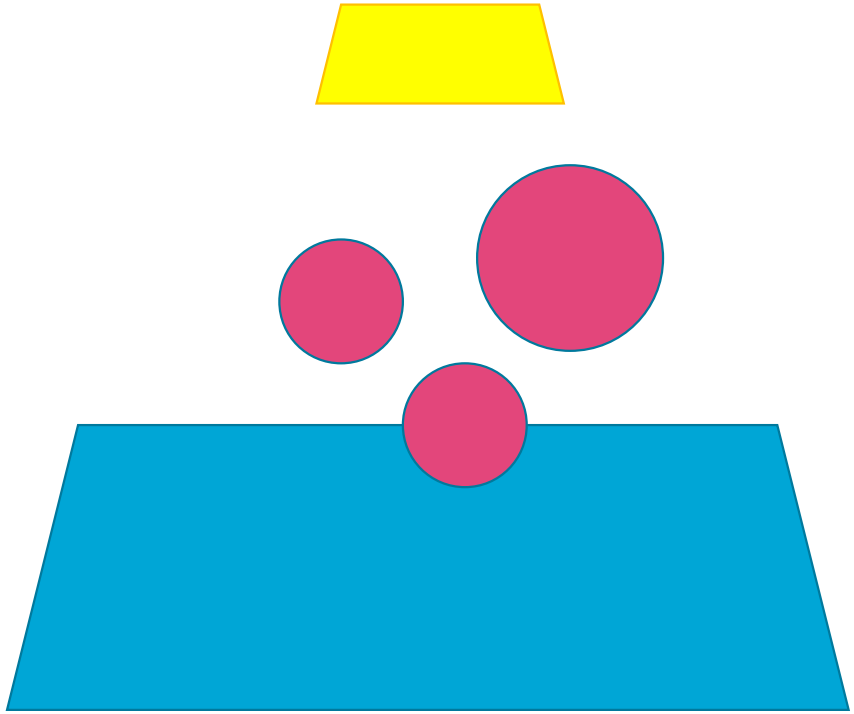
- Transfer the shaded pixels back to the view



[Sintorn et al. EGSR08]

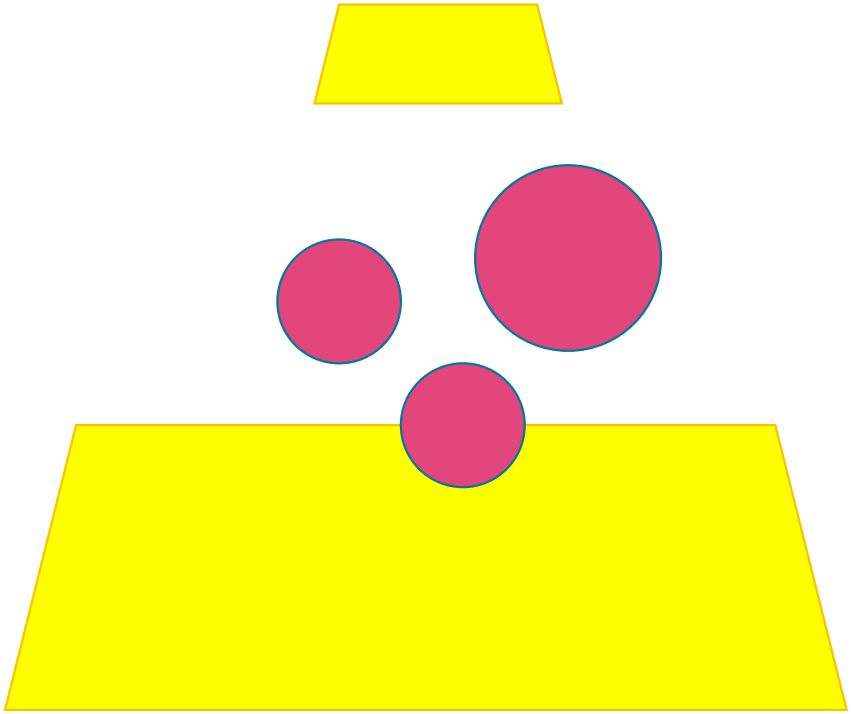
Simpler Problem

- What are the shadows on the plane?



Simpler Problem

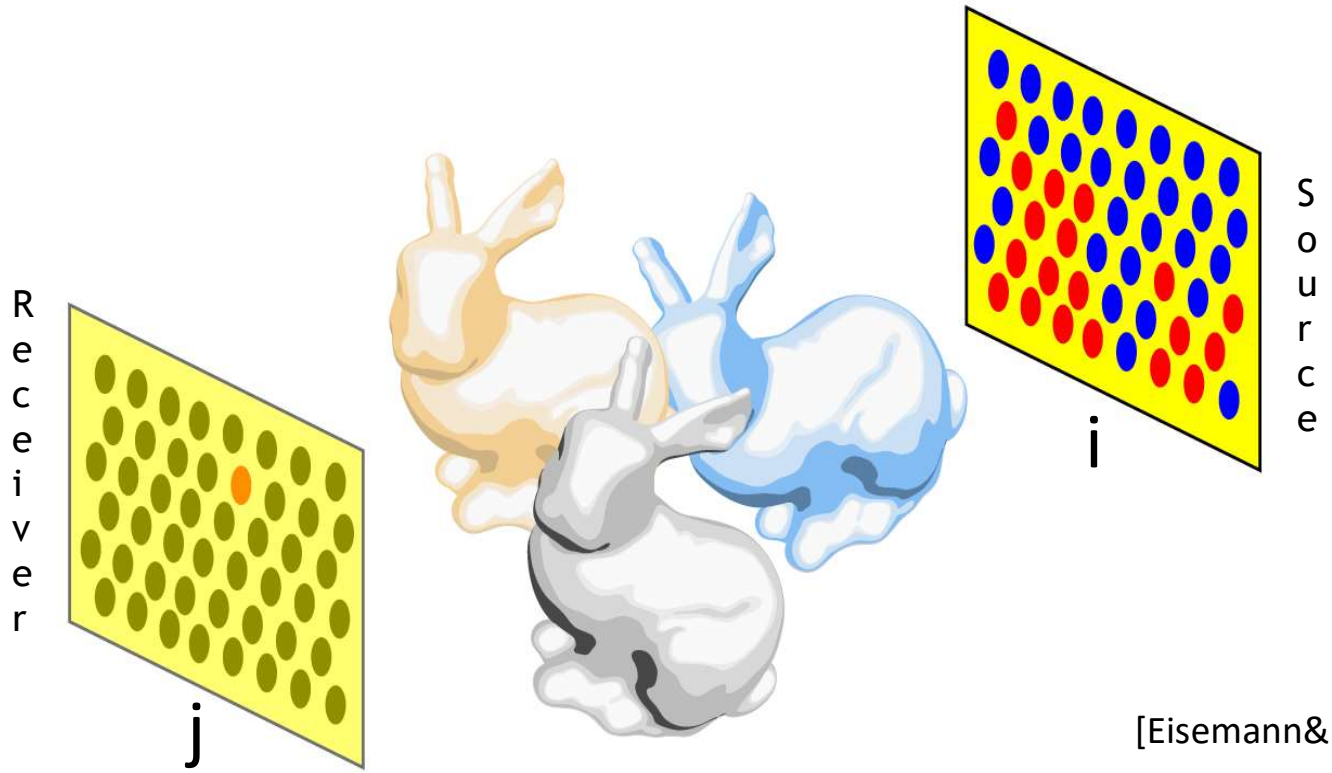
- What is the visibility between the planes?



Visibility Sampling on the GPU

- Compute blocked samples

$$\mathcal{B}_j := \{S_i | \exists k \; [S_i, R_j] \cap T_k \neq \emptyset\}$$



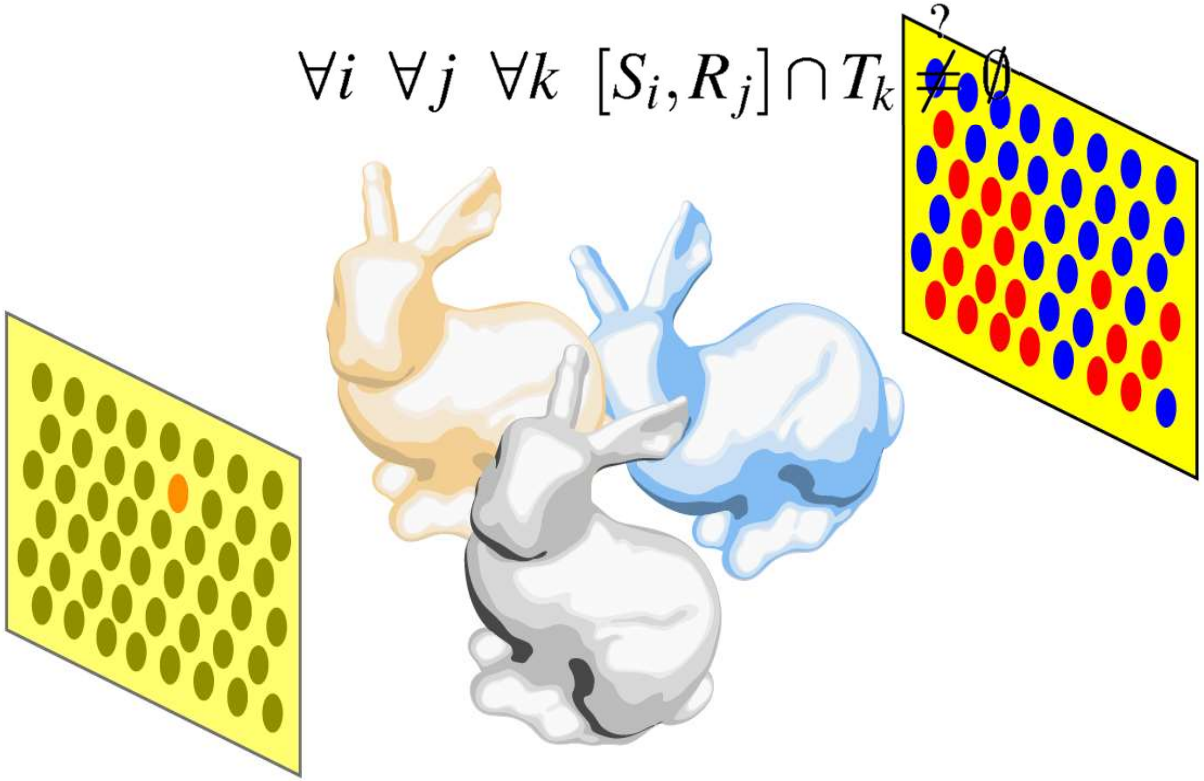
[Eisemann&Décoret07]

Principle

- Compute blocked samples

$$\mathcal{B}_j := \{S_i | \exists k \; [S_i, R_j] \cap T_k \neq \emptyset\}$$

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



Principle

Raytracing

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

$$\forall i \ \forall k \ \forall j \ [S_i, R_j] \cap T_k \neq \emptyset$$

Shadow/Occlusion Mapping

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

See [Laine&Aila05]

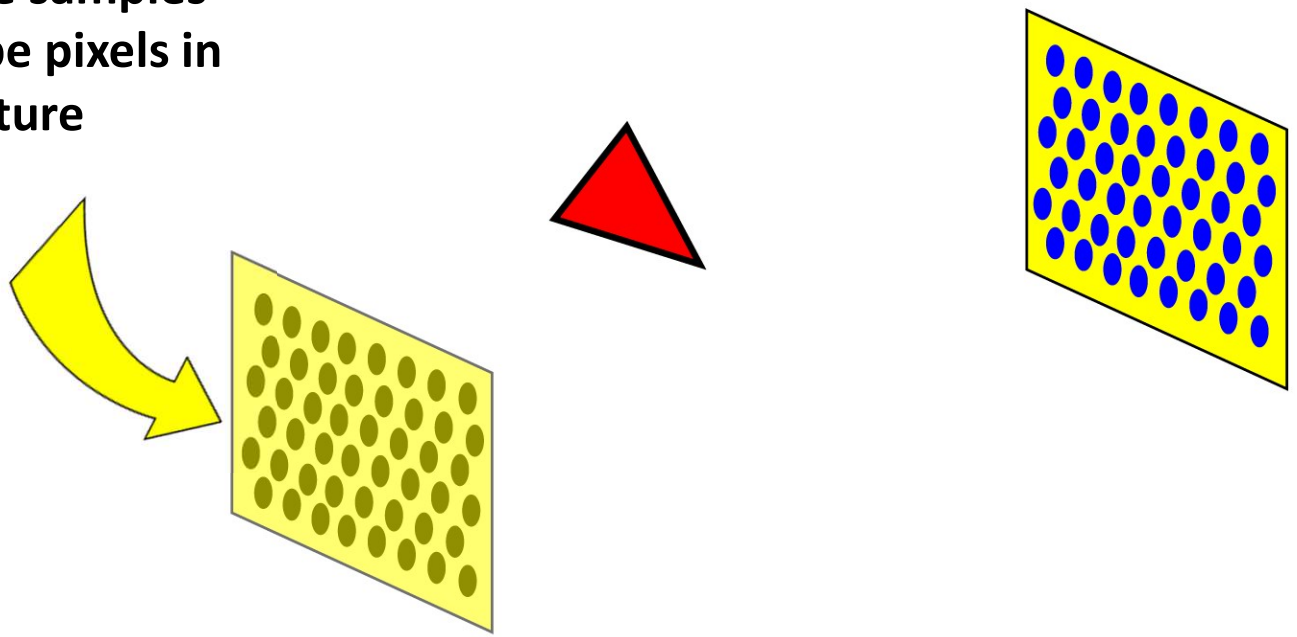
Treat triangles separately

Info available on a single GPU Processor!

Principle

- Basic algorithm

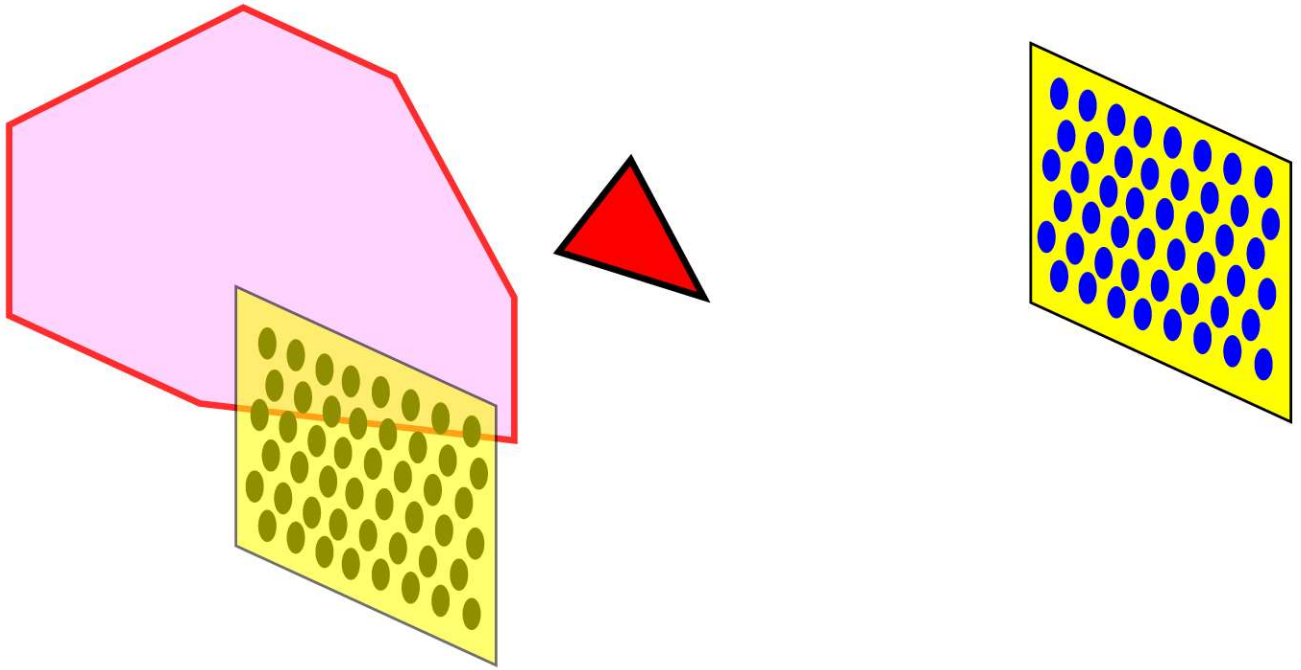
These samples will be pixels in a texture



Two patches with samples and occluding triangle

Principle

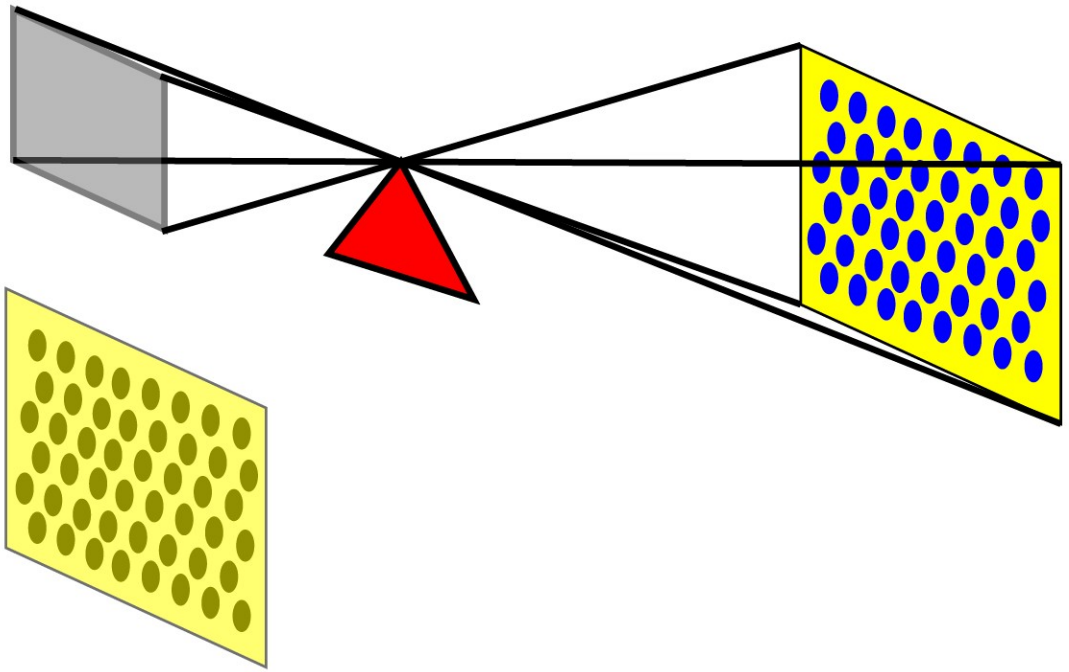
- Basic algorithm



Consider right patch as light source and find penumbra region

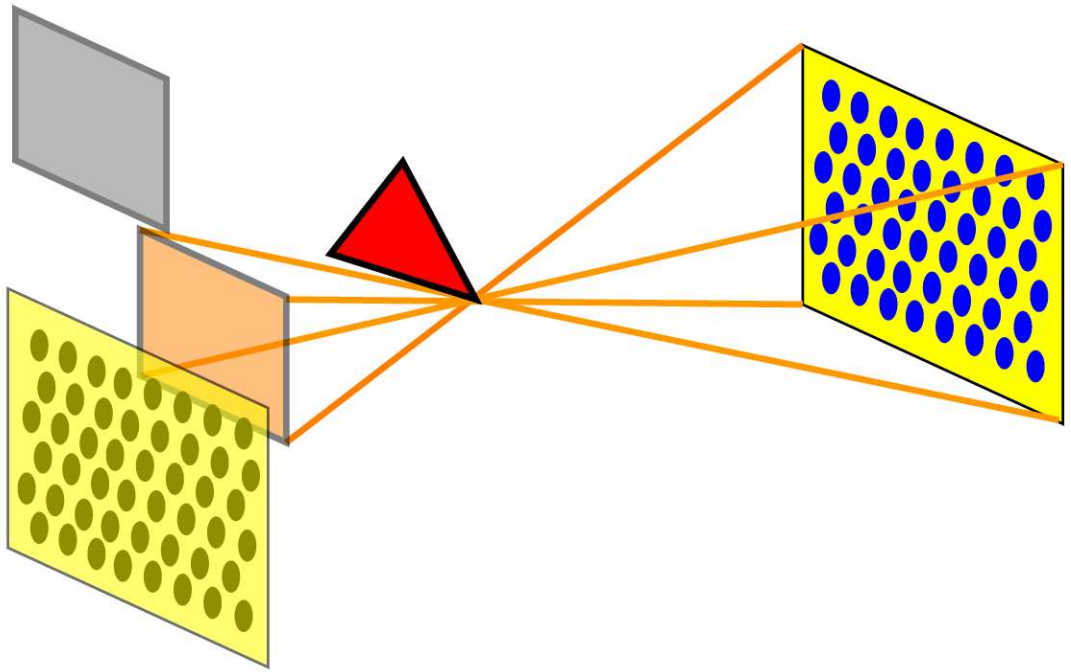
Penumbra Region

- How to determine the penumbra region?



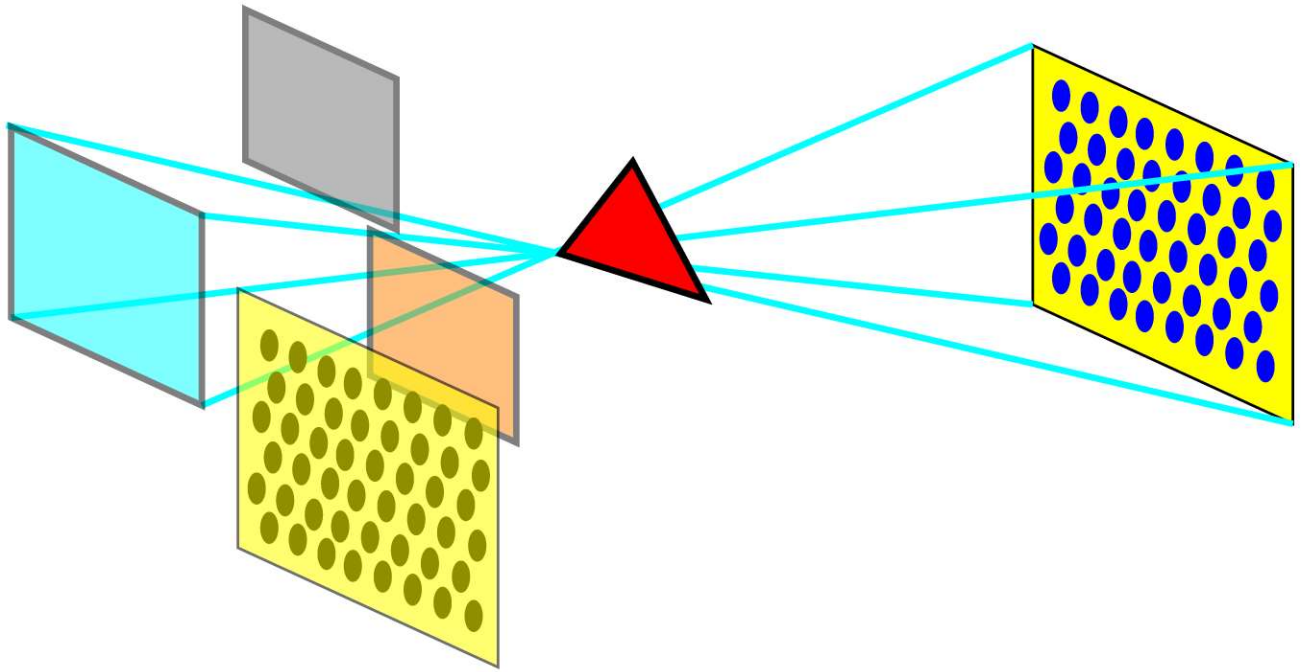
Penumbra Region

- How to determine the penumbra region?



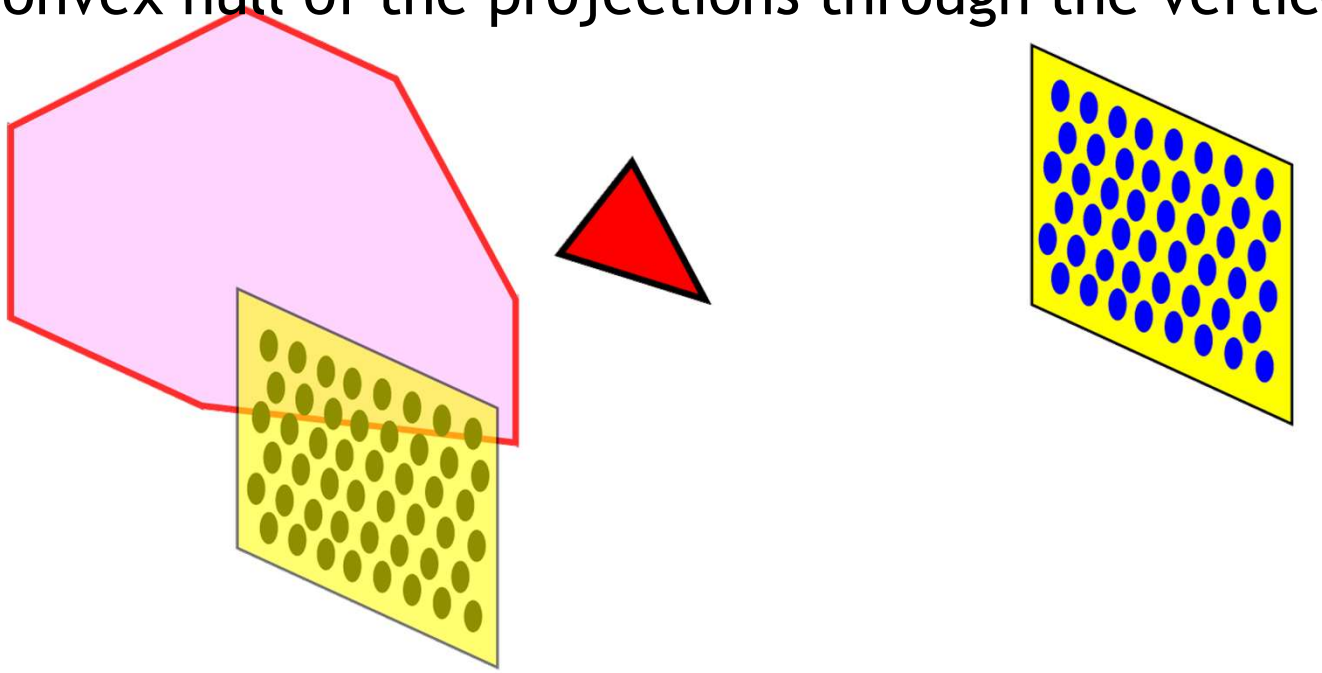
Penumbra Region

- How to determine the penumbra region?



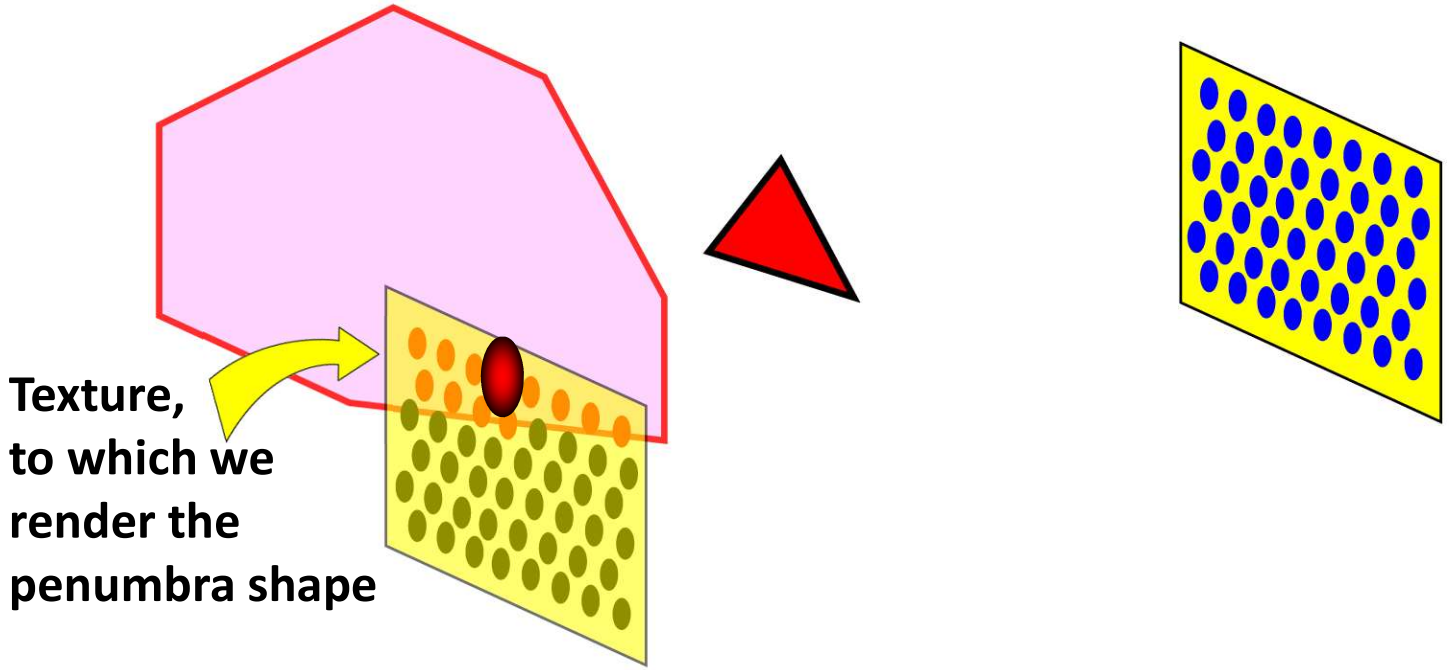
Penumbra Region

- How to determine the penumbra region?
- Convex hull of the projections through the vertices



Principle

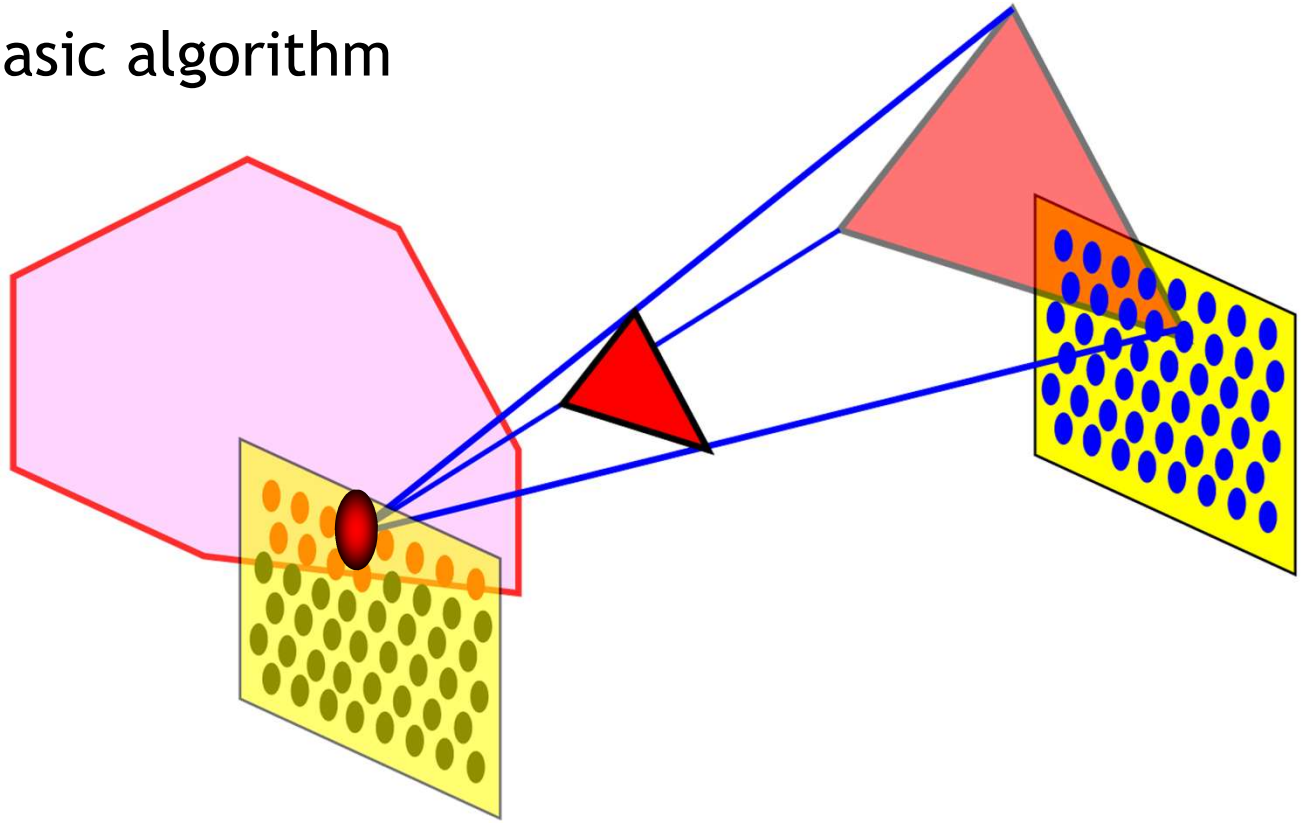
- Basic algorithm



Each sample in penumbra is treated

Principle

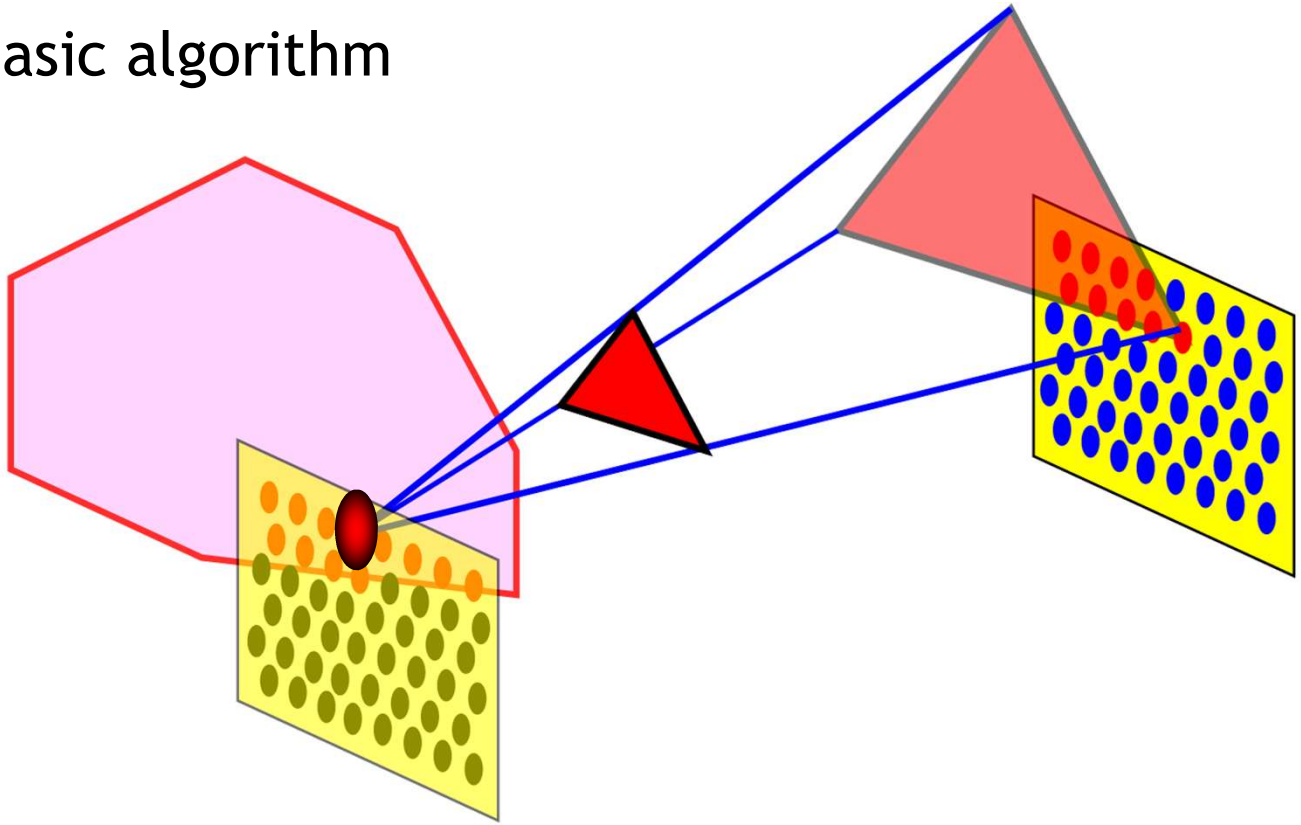
- Basic algorithm



Backproject triangle from sample point

Principle

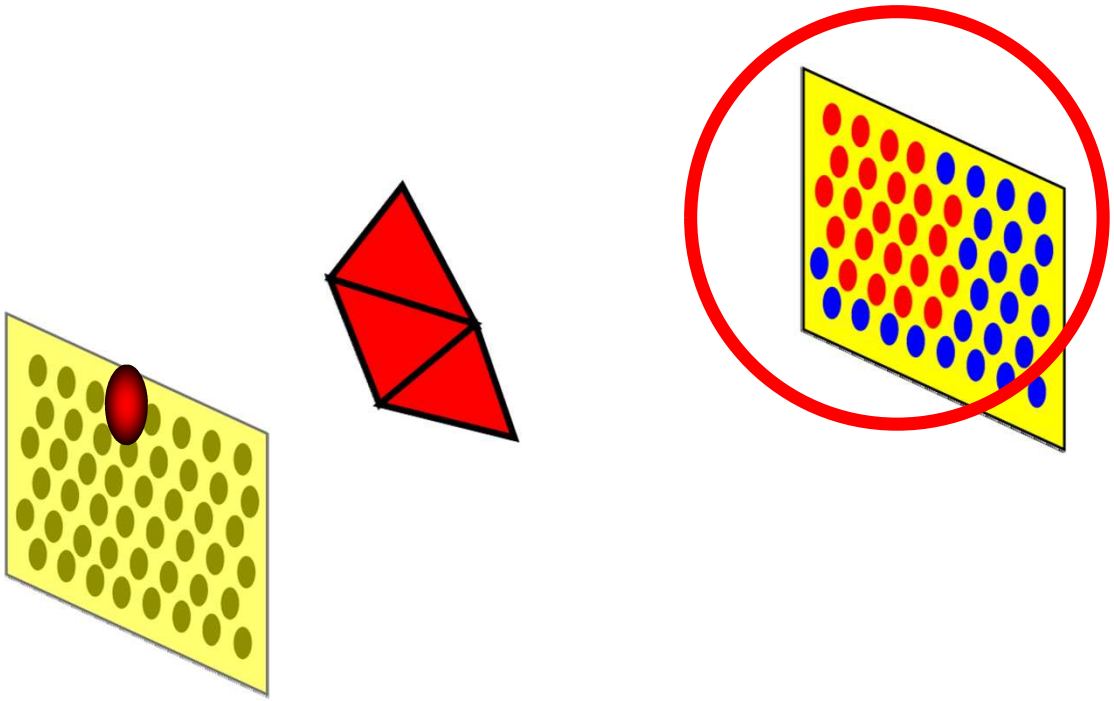
- Basic algorithm



Determine blocked samples

Principle

- Basic algorithm

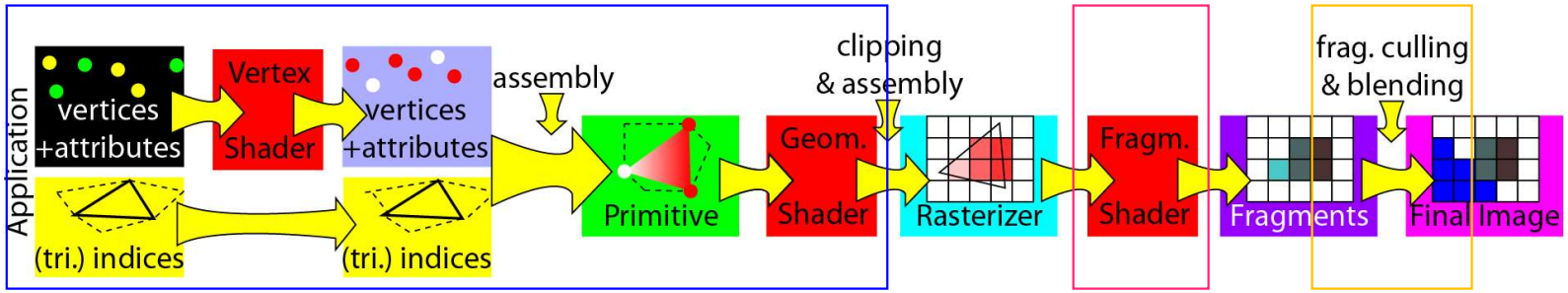


Accumulate over all triangles

View-Sample Mapping

- Overview:

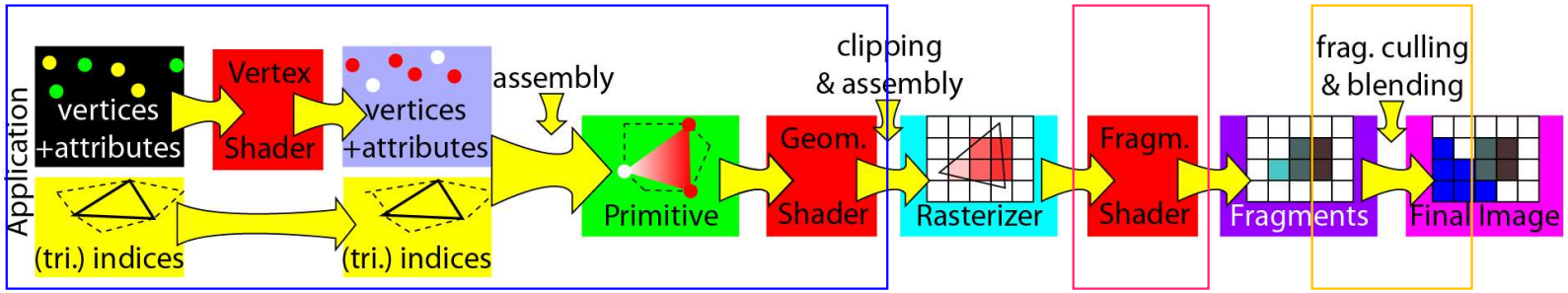
Vertex/Geom Shader	Find penumbra region + construct its geometry
Fragment Shader	Backproject triangle for each fragment
	Determine blocked samples
Blending	Accumulate over all triangles



View-Sample Mapping

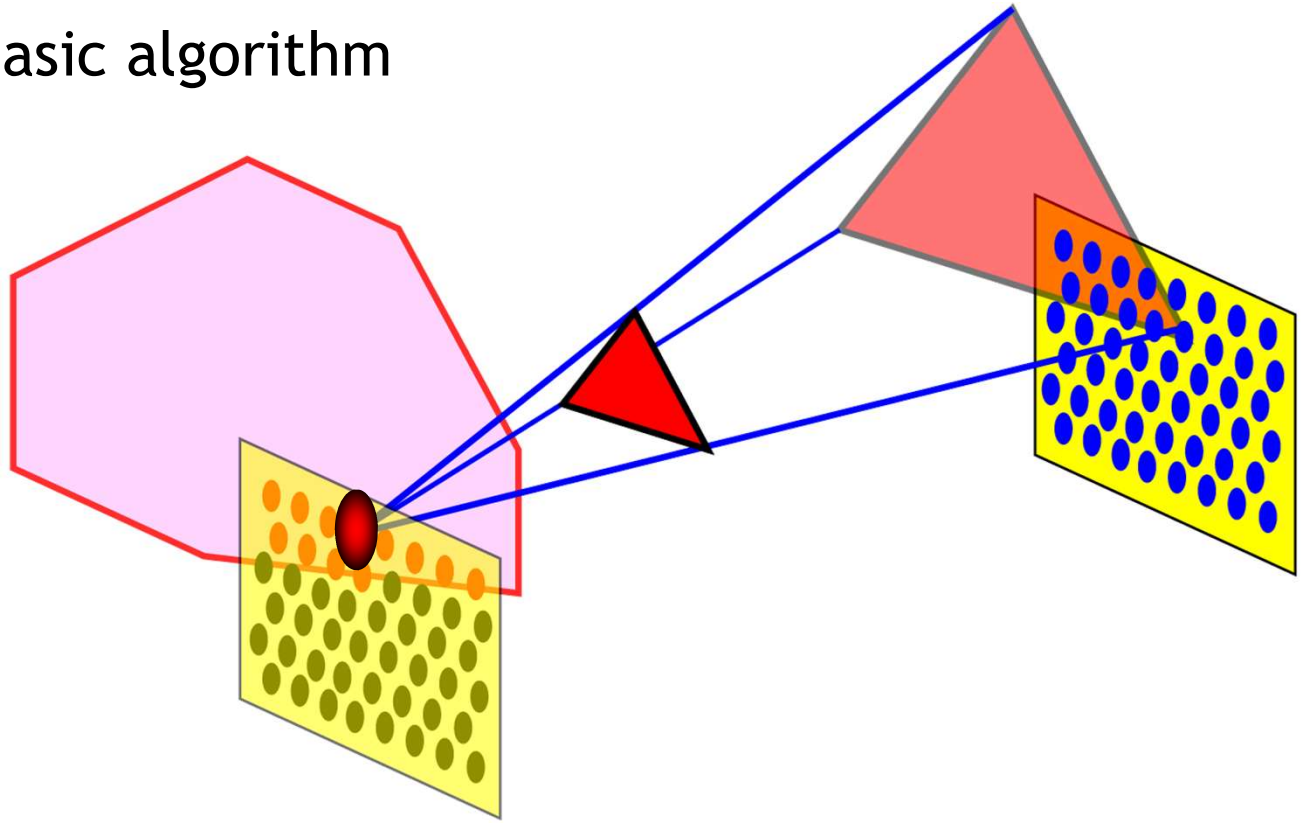
- Overview:

Vertex/Geom Shader	Find penumbra region + construct its geometry
Fragment Shader	Backproject triangle for each fragment
	Determine blocked samples
Blending	Accumulate over all triangles



Where are we?

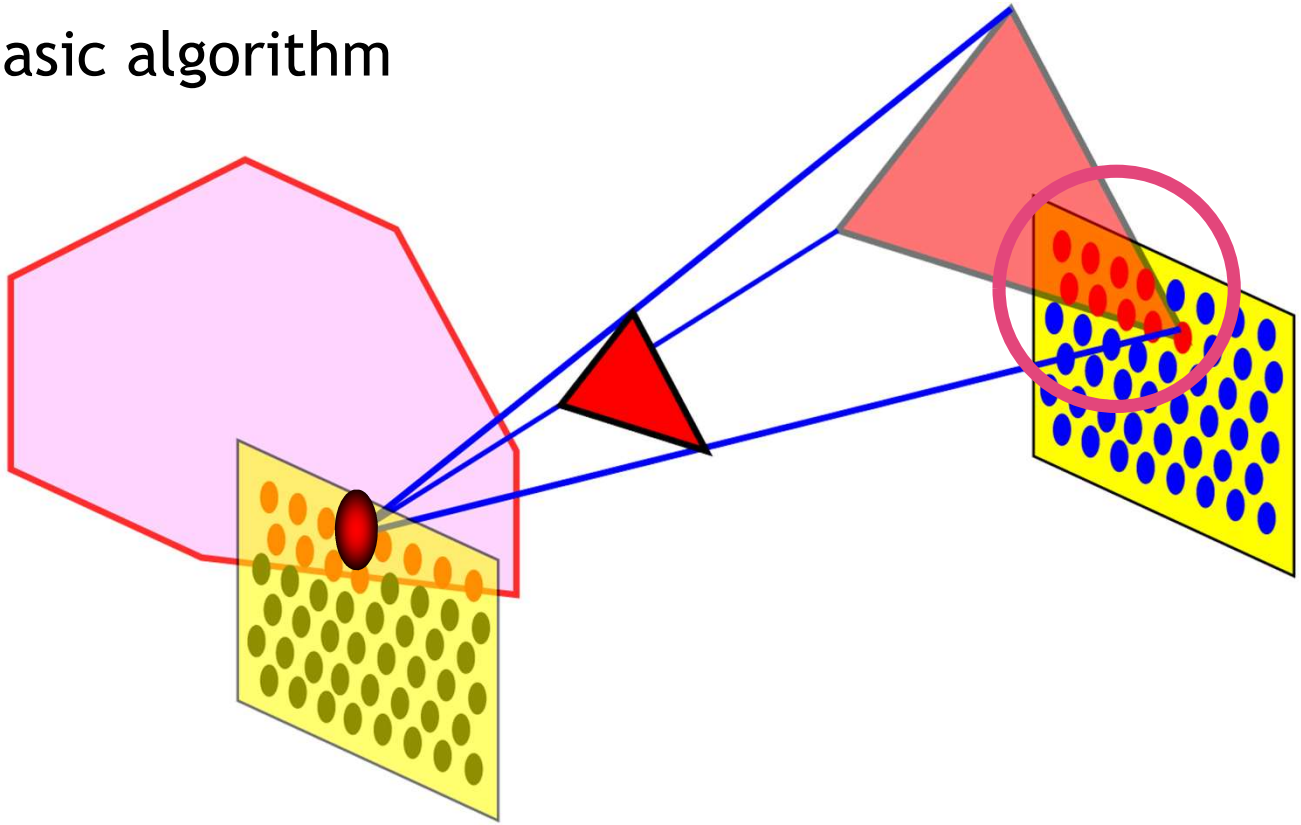
- Basic algorithm



Backproject triangle from sample point

Where are we?

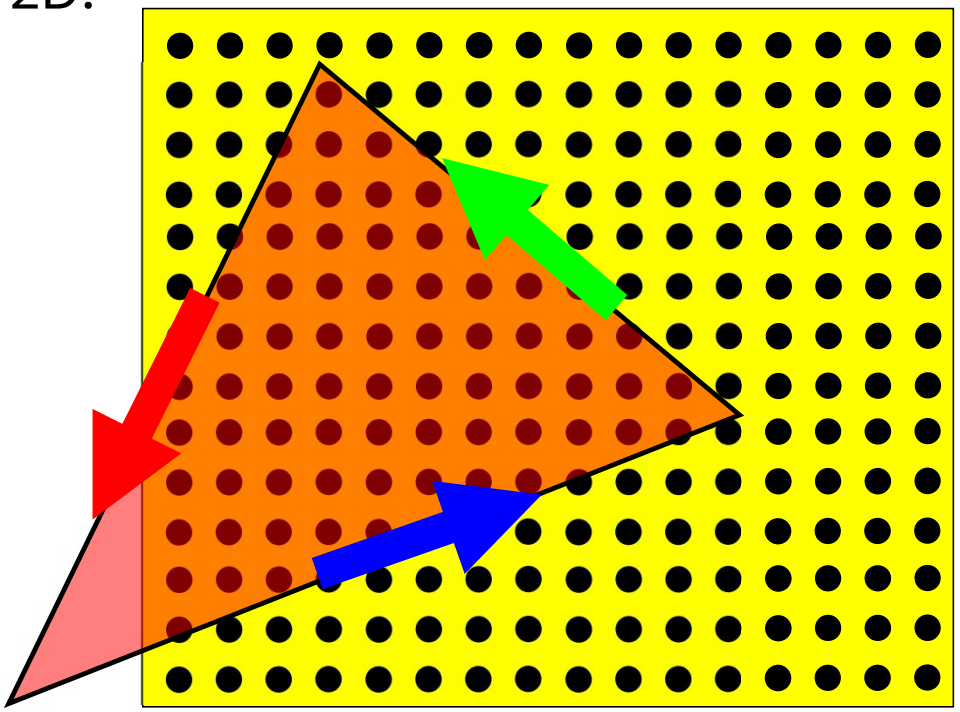
- Basic algorithm



Determine blocked samples

Blocked Samples

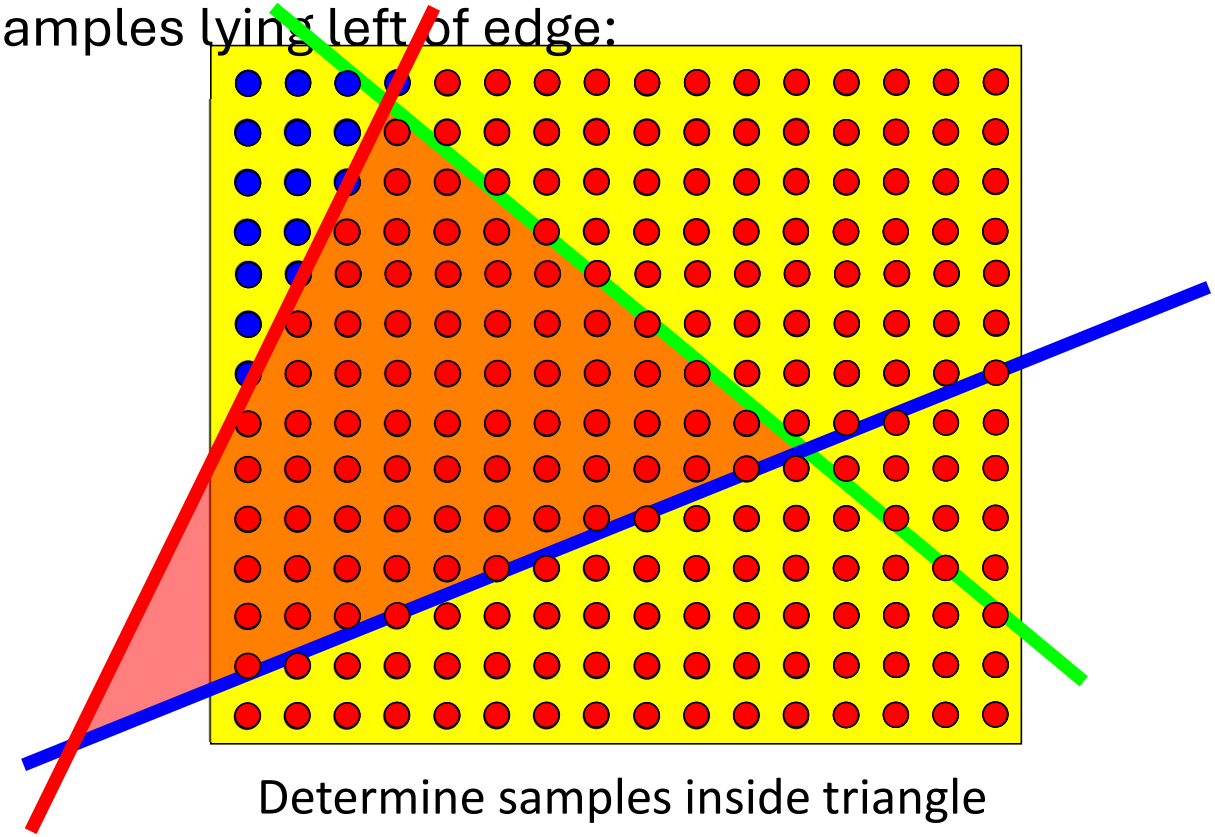
- The problem is 2D:



Determine samples inside triangle

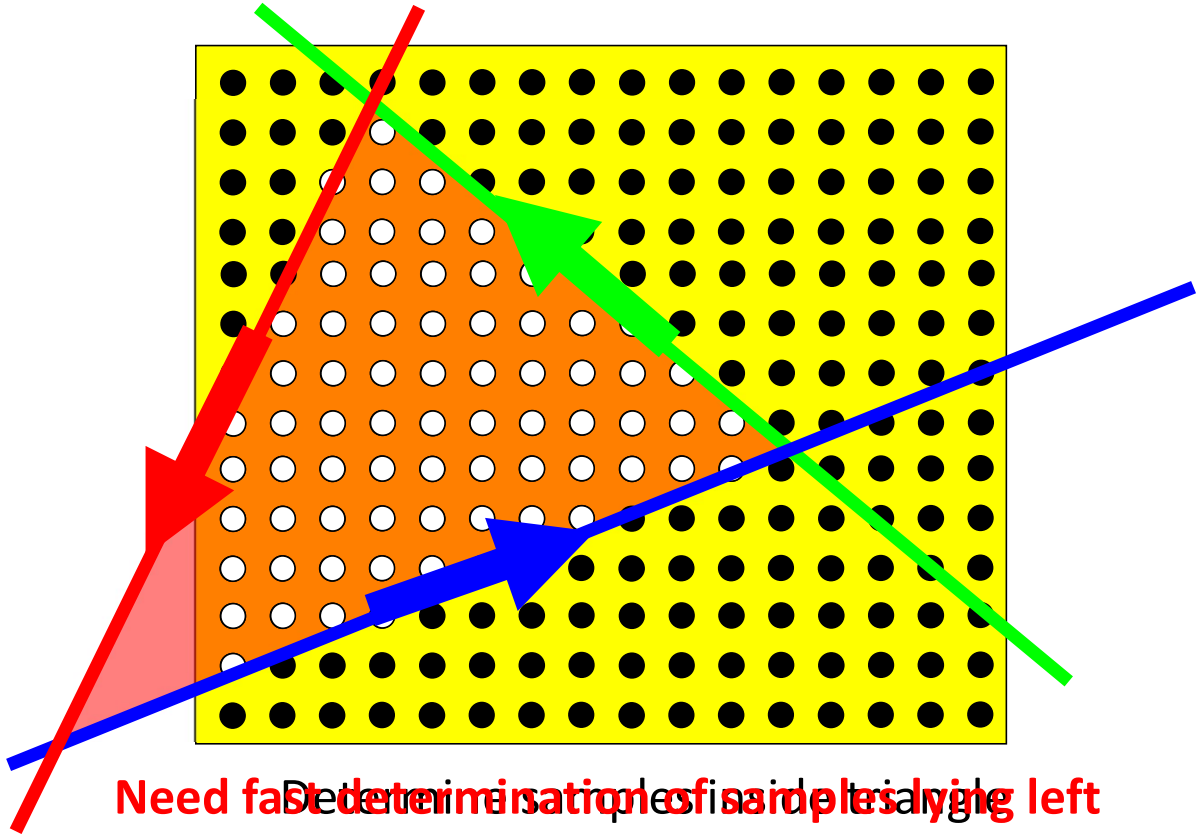
Blocked Samples

- Determine samples lying left of edge:



Determine samples inside triangle

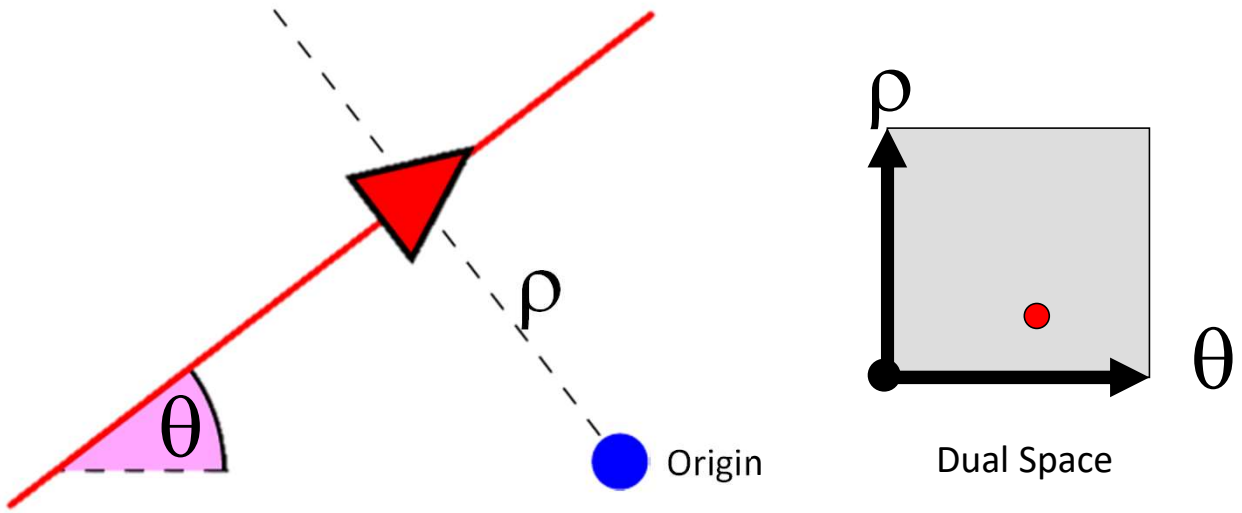
Blocked Samples



Hough Transformation

- Dual line representation:

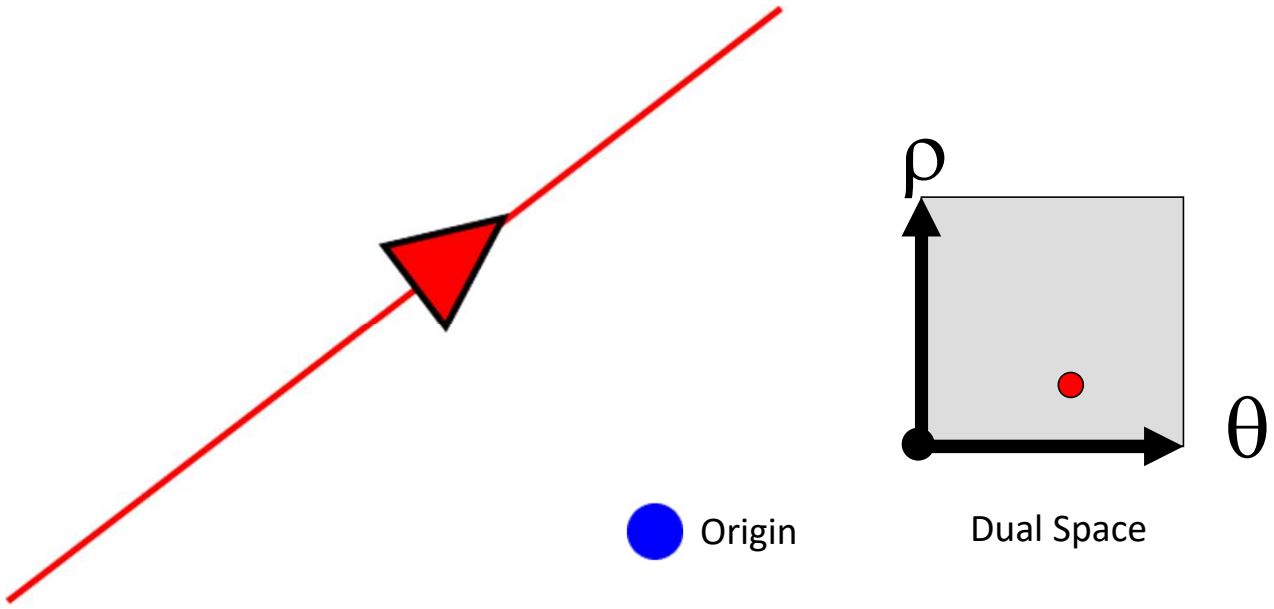
$$L_{\theta\rho}(\alpha) = (\cos\theta, \sin\theta)\,\alpha + \rho\,(-\sin\theta, \cos\theta)$$



Hough Transformation

- Dual line representation:

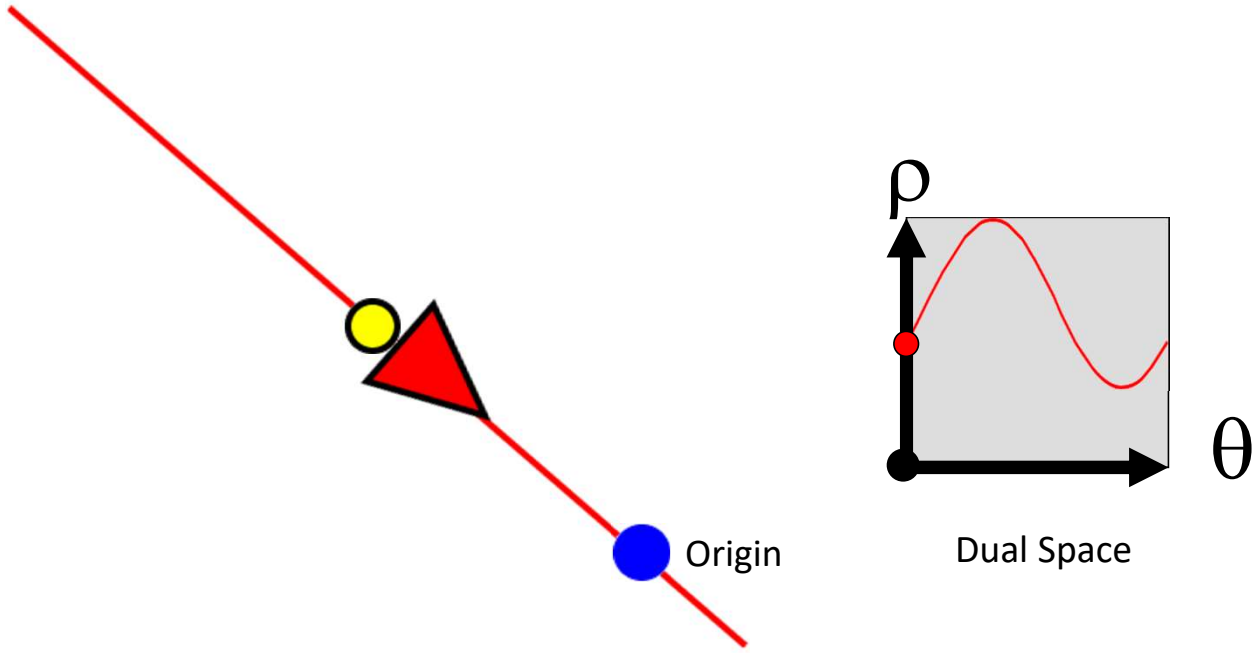
$$L_{\theta\rho}(\alpha) = (\cos\theta, \sin\theta)\,\alpha + \rho\,(-\sin\theta, \cos\theta)$$



Hough Transformation

- Dual line representation:

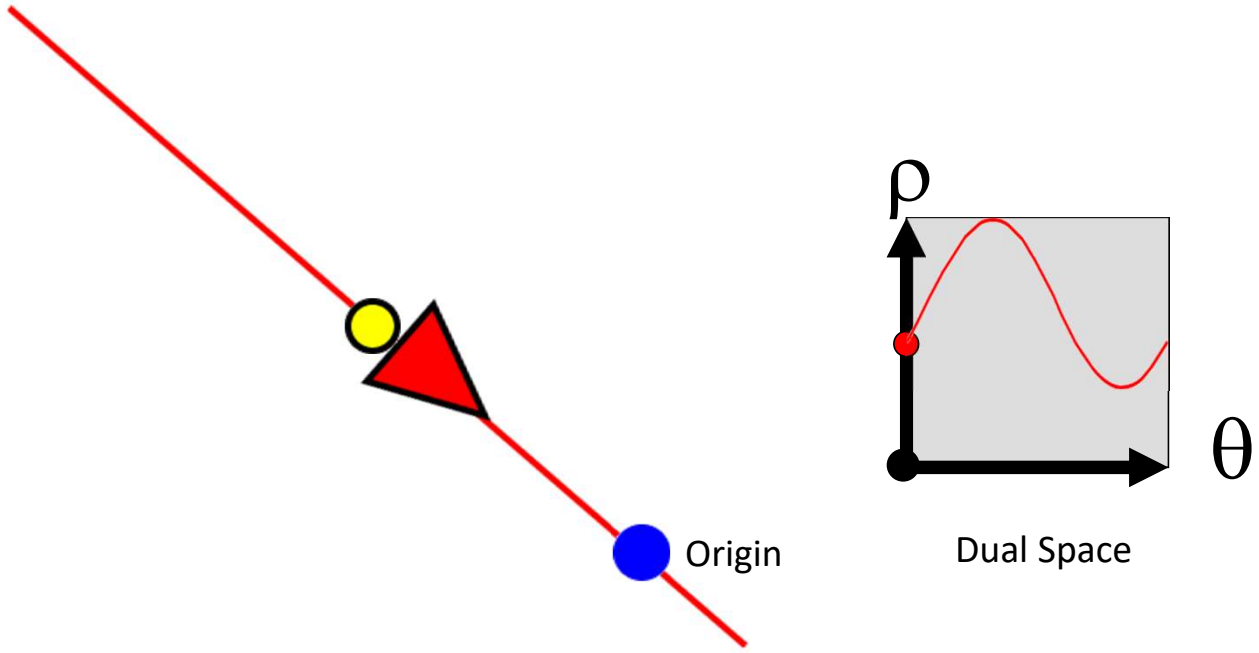
$$L_{\theta\rho}(\alpha) = (\cos\theta, \sin\theta)\,\alpha + \rho\,(-\sin\theta, \cos\theta)$$



Hough Transformation

- Dual line representation:

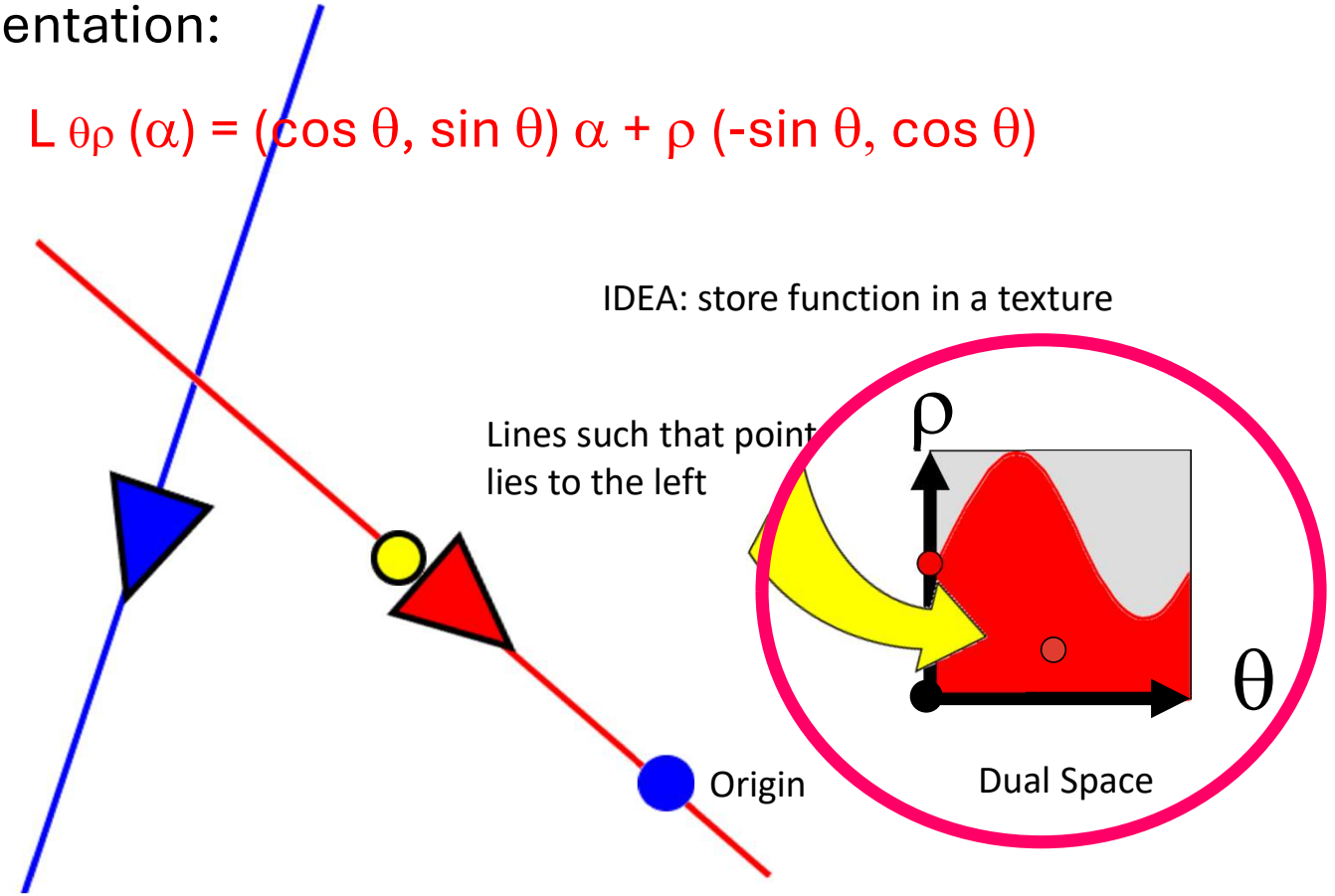
$$L_{\theta\rho}(\alpha) = (\cos\theta, \sin\theta)\,\alpha + \rho\,(-\sin\theta, \cos\theta)$$



Hough Transformation

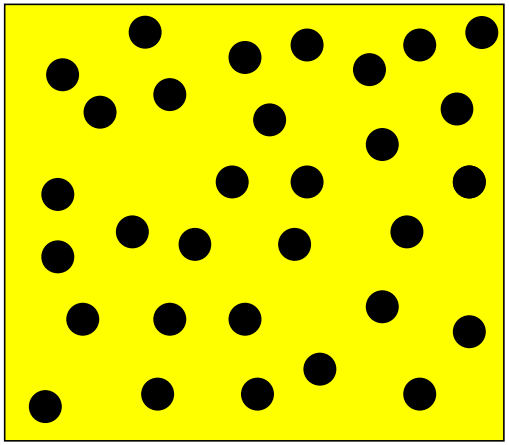
- Dual line representation:

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



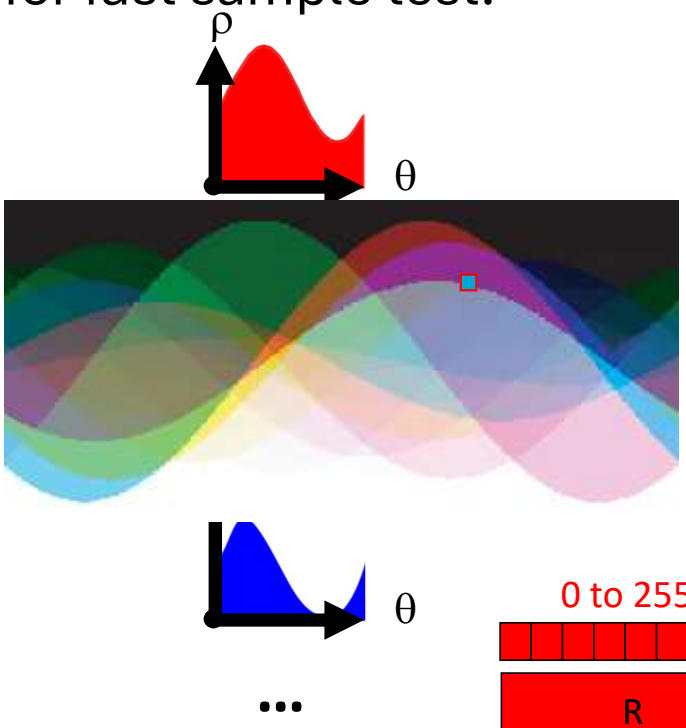
Lookup Texture (Precomputed)

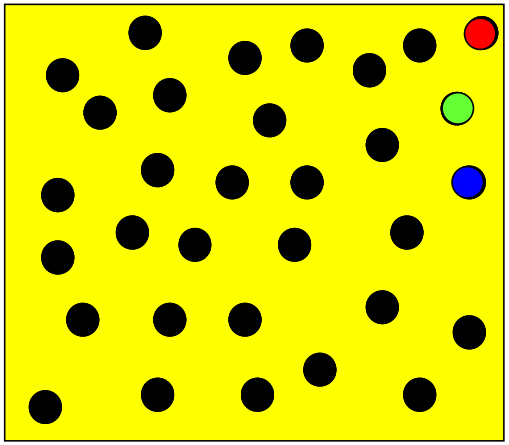
- Select and keep same samples on light source

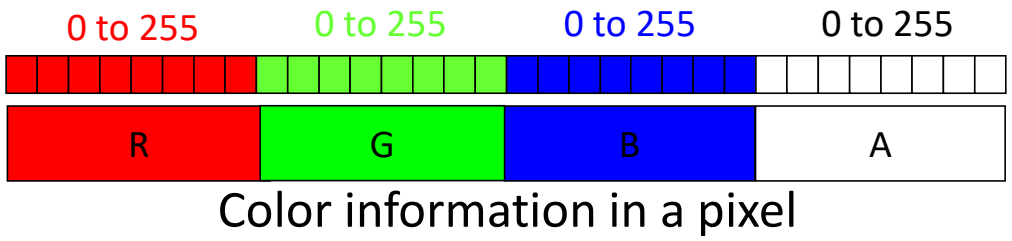


Lookup Texture (Precomputed)

- Texture for fast sample test:



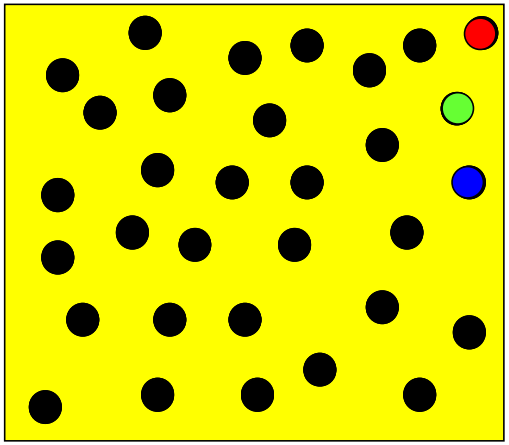


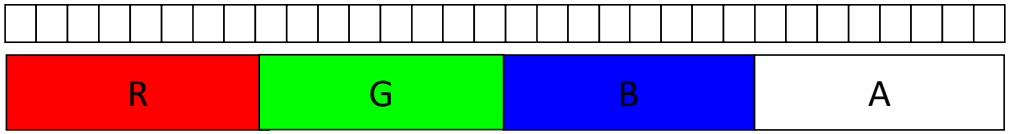


Color information in a pixel

Lookup Texture (Precomputed)

- Texture for fast sample test:



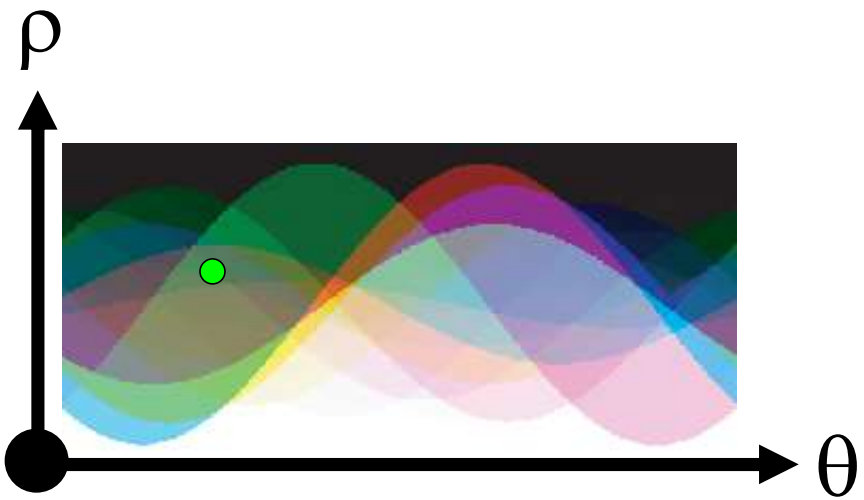


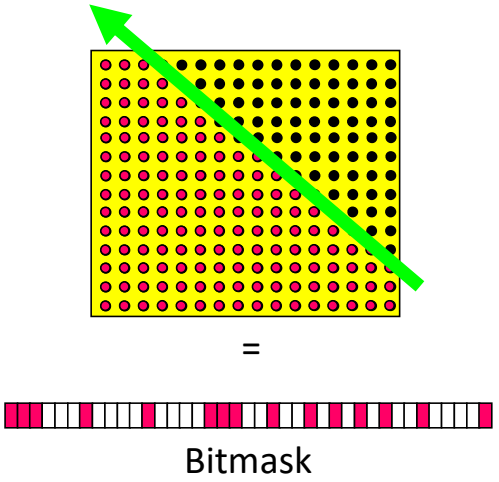
Color information in a pixel

Associate each sample to one bit

Lookup Table

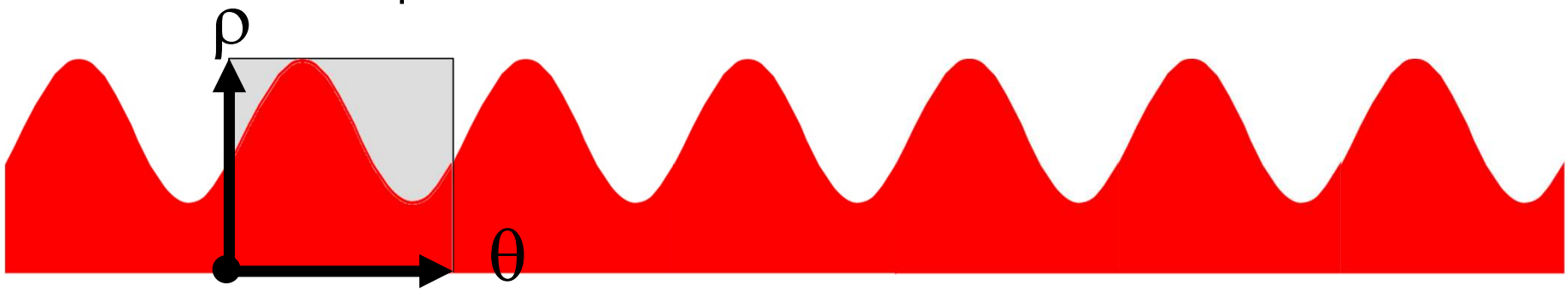
- LU table for fast sample test:





Lookup Table

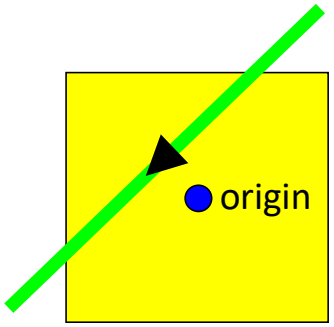
- Store an unlimited space in a texture:



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

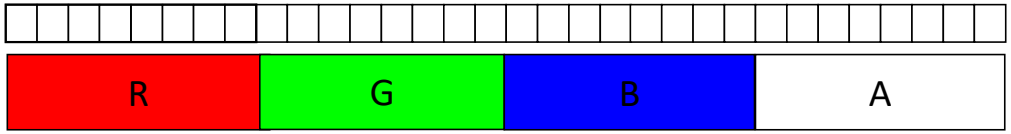
- Binary info:

  (128 bits per texture ~1024=8*(4*32) possible)

„one lookup“ = result for all samples

How to store in bits?

- Imagine channels with 8 bits

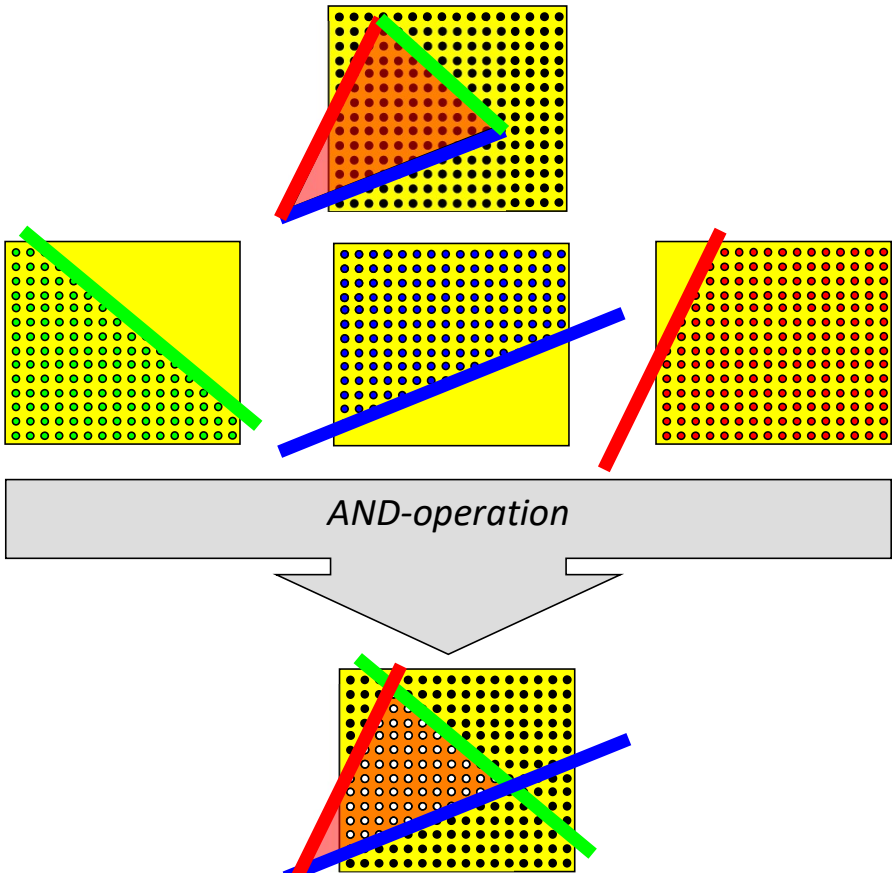


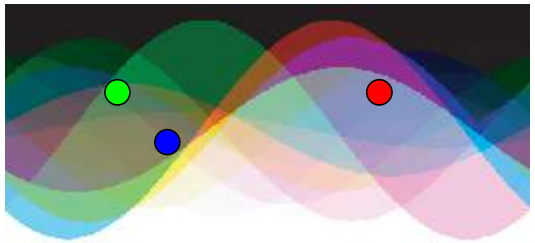
- 1= first bit activated
- 2=second bit activated
- 1+2=first and second bit activated.

Generally: Σ active[i]*2^i

Blocked Samples in Triangle

- Combining edges is a simple *AND*-operation





View-sample Mapping

- Overview:

Find influence region

Backproject triangle

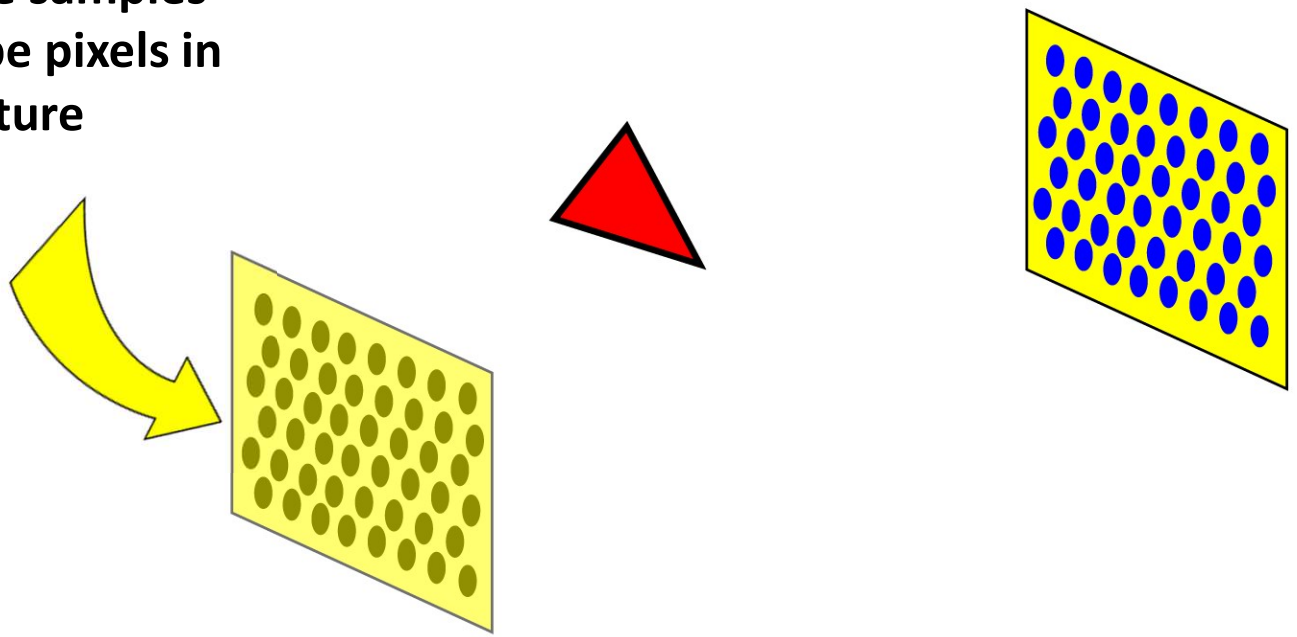
Determine blocked samples

Accumulate over all triangles

Principle

- Basic algorithm

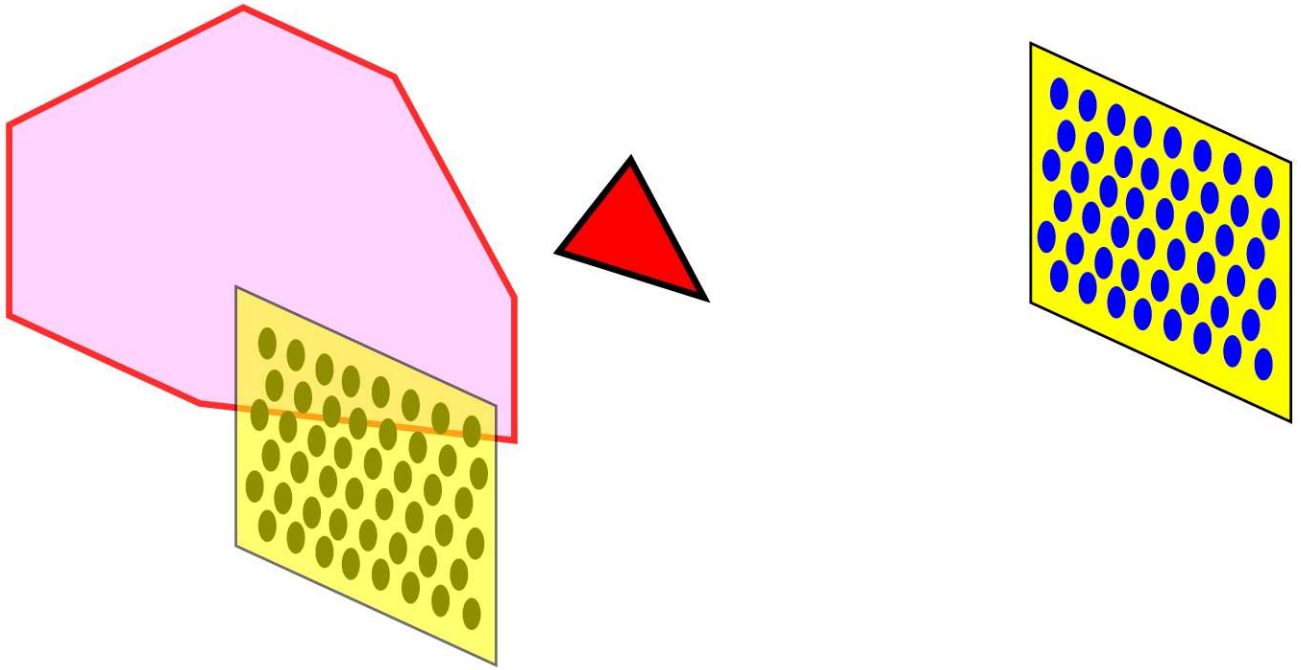
These samples will be pixels in a texture



Two patches with samples and occluding triangle

Principle

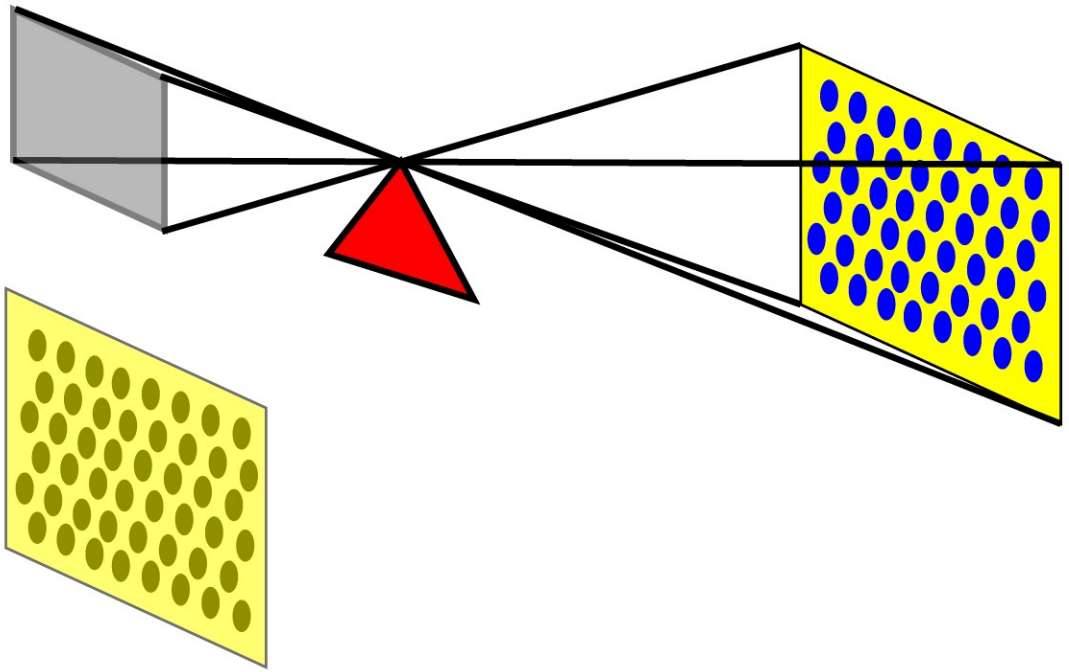
- Basic algorithm



Consider one patch as source and find penumbra region

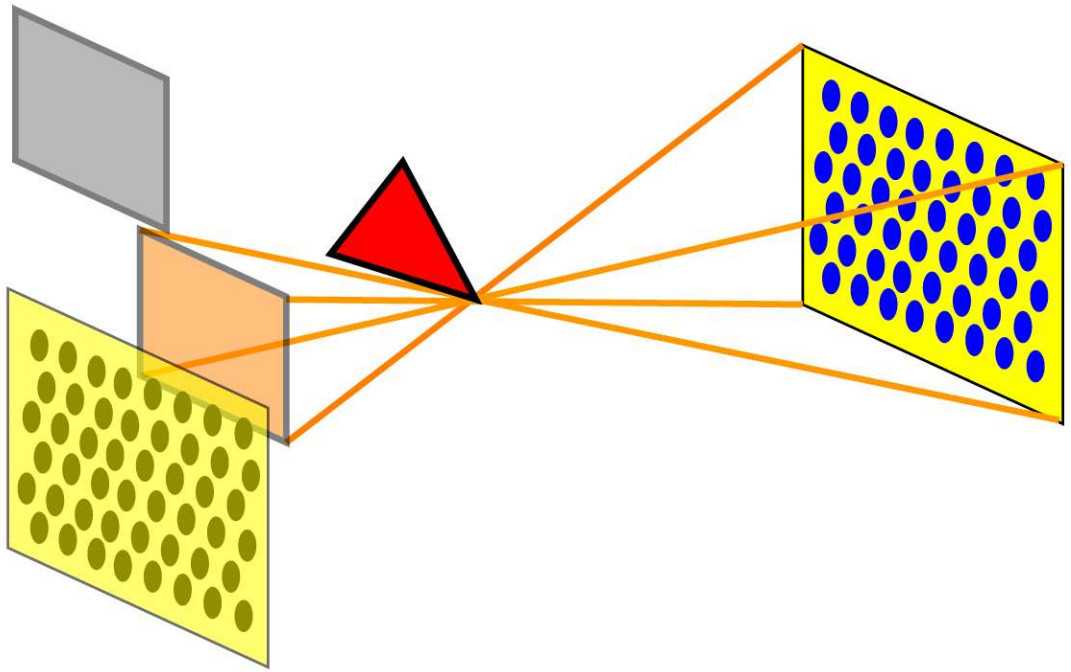
Penumbra Region

- How to determine the penumbra region?



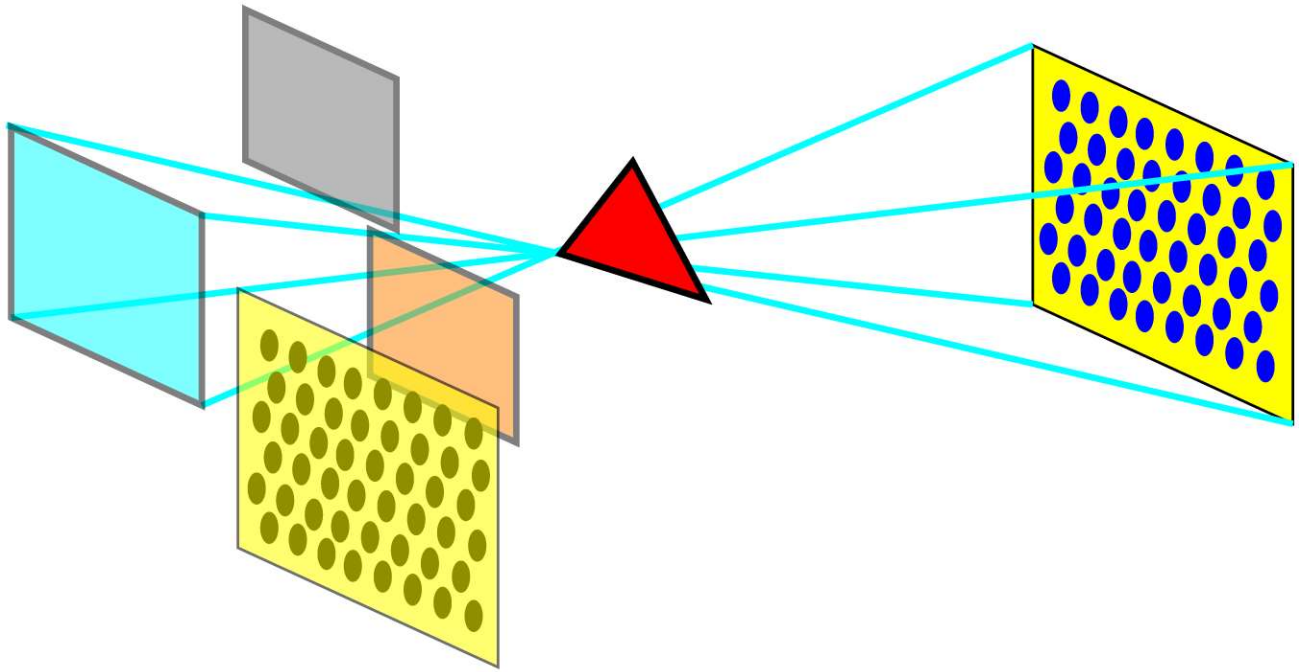
Penumbra Region

- How to determine the penumbra region?



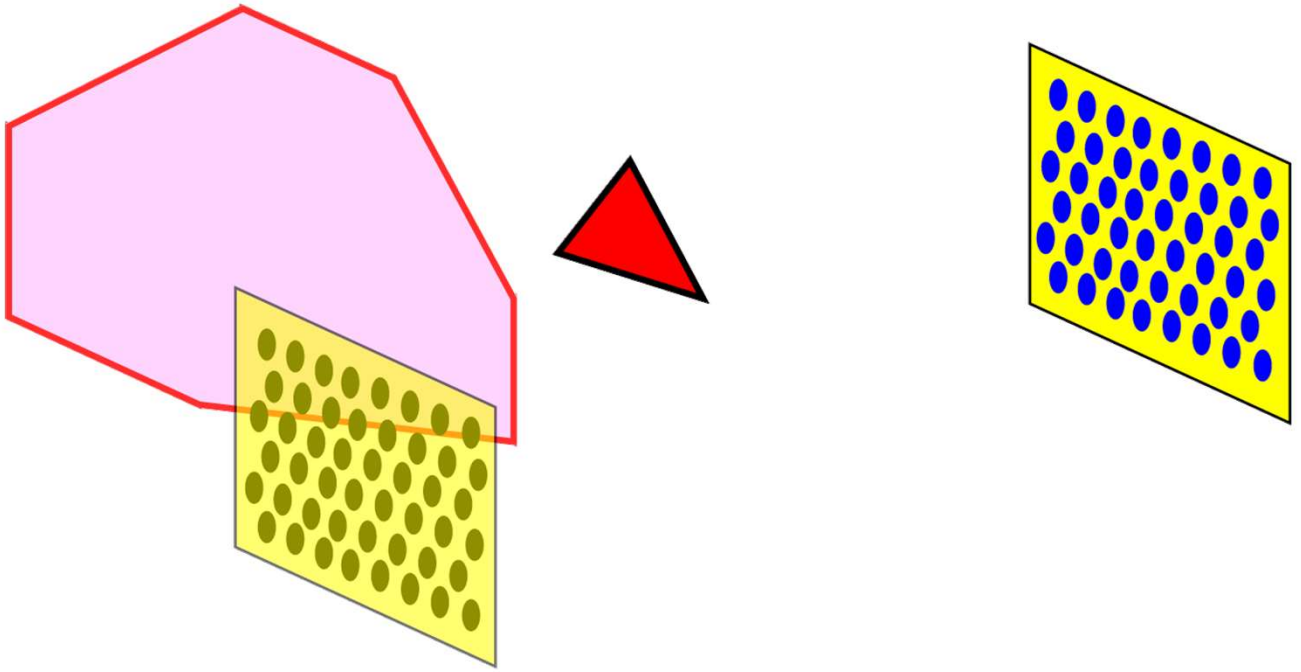
Penumbra Region

- How to determine the penumbra region?



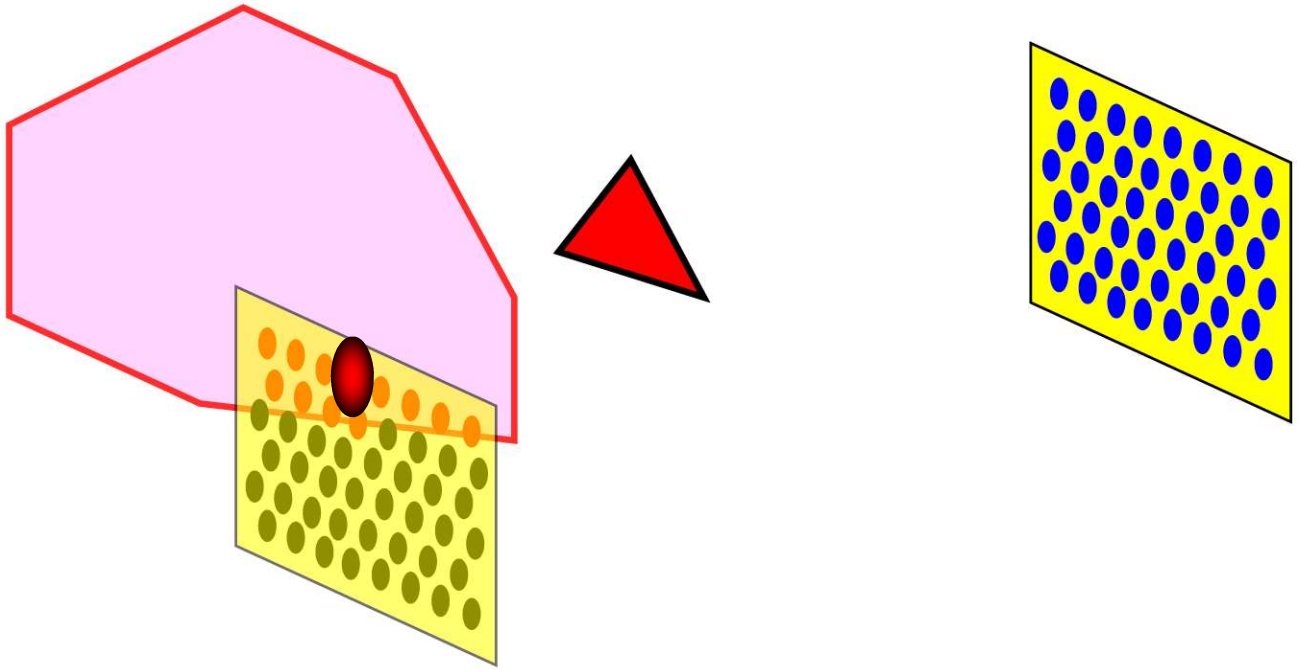
Penumbra Region

- How to determine the penumbra region?



Principle

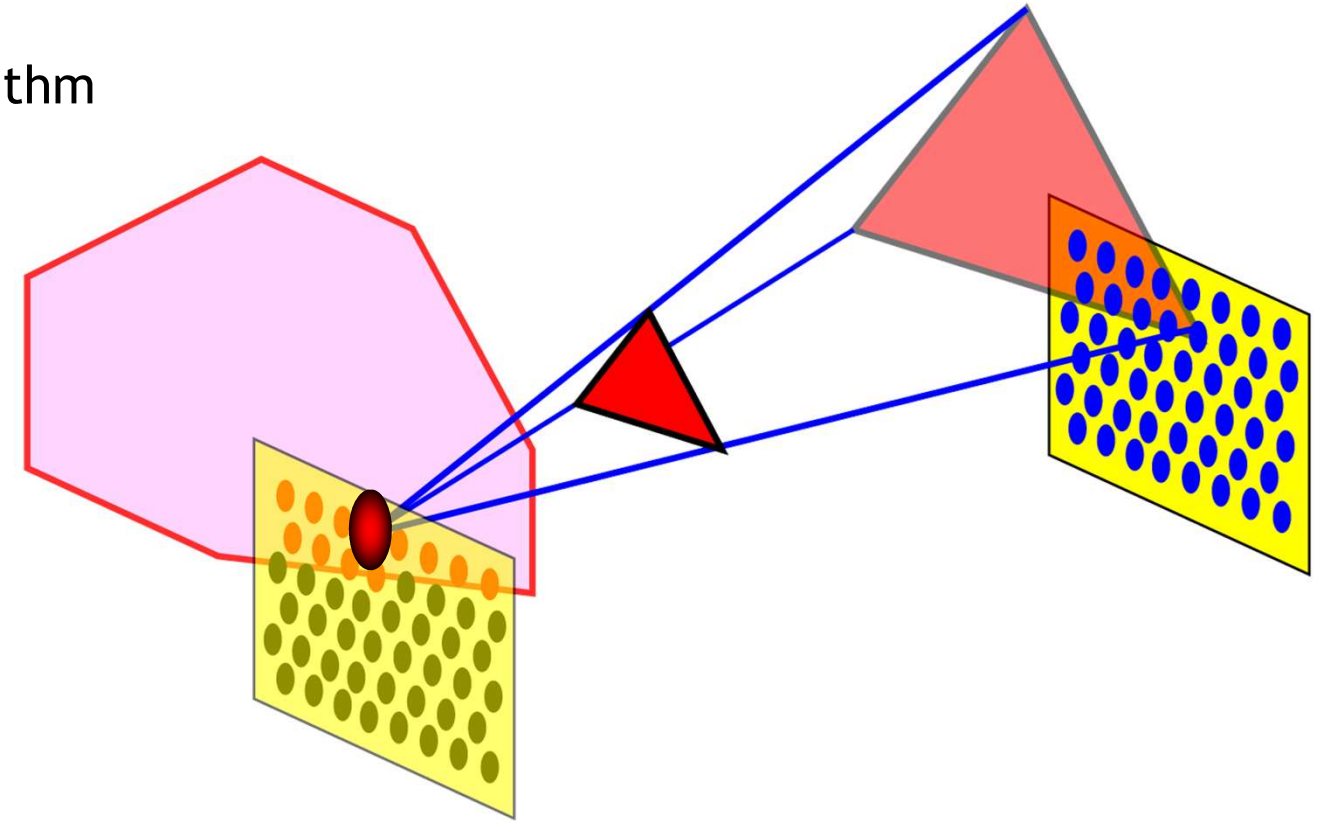
- Basic algorithm



Each sample in penumbra is treated

Principle

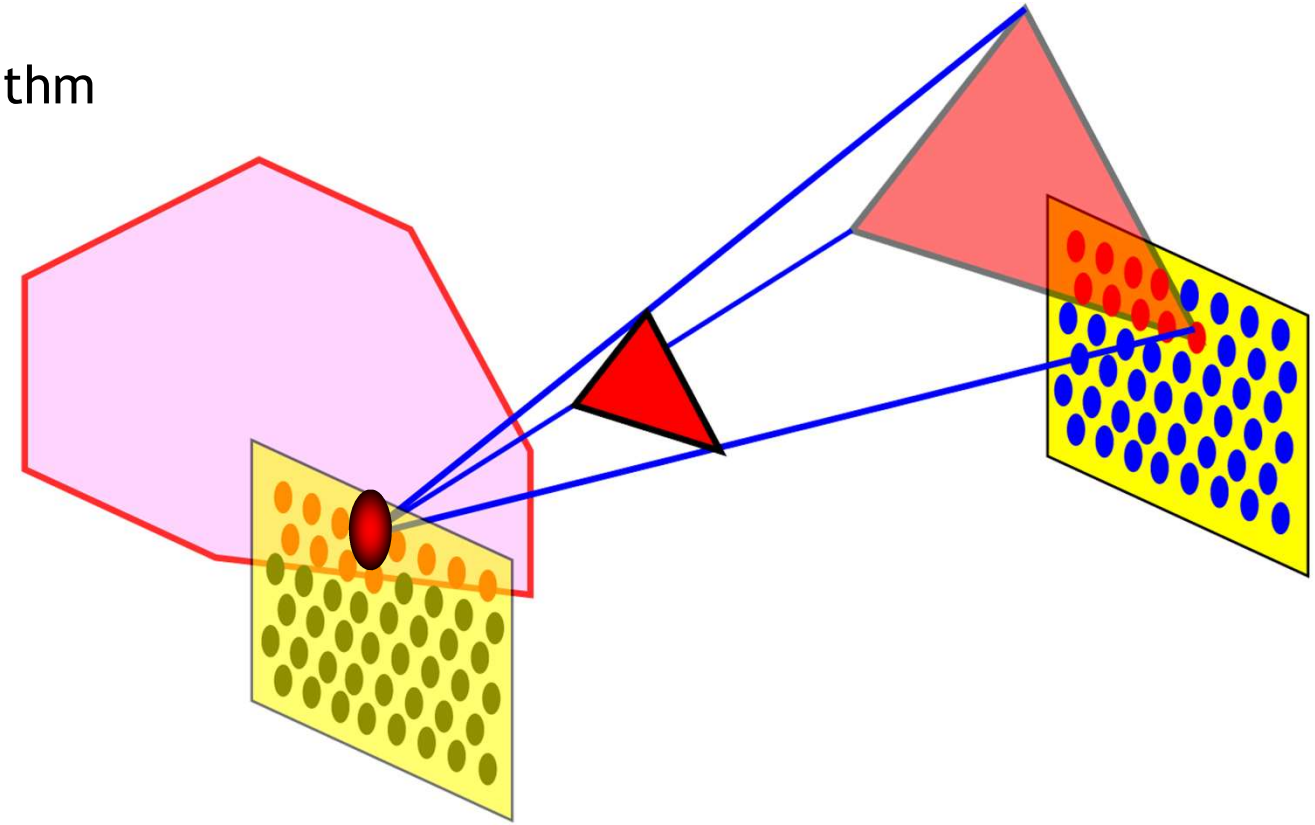
- Basic algorithm



Backproject triangle from sample point

Principle

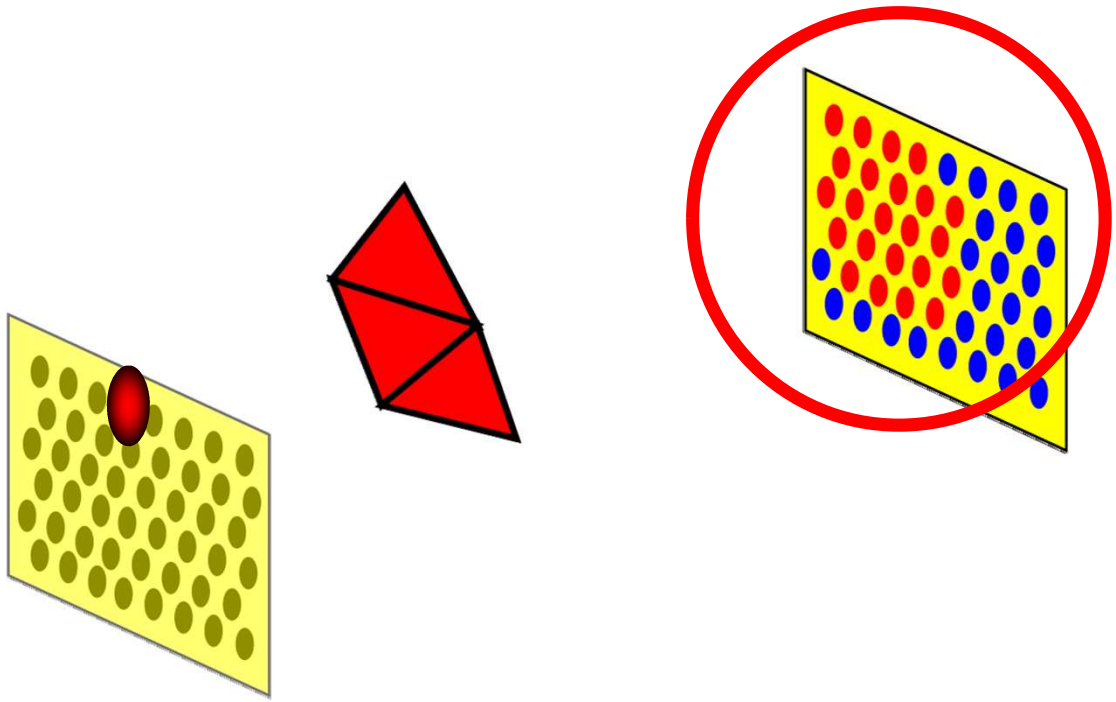
- Basic algorithm



Determine blocked samples

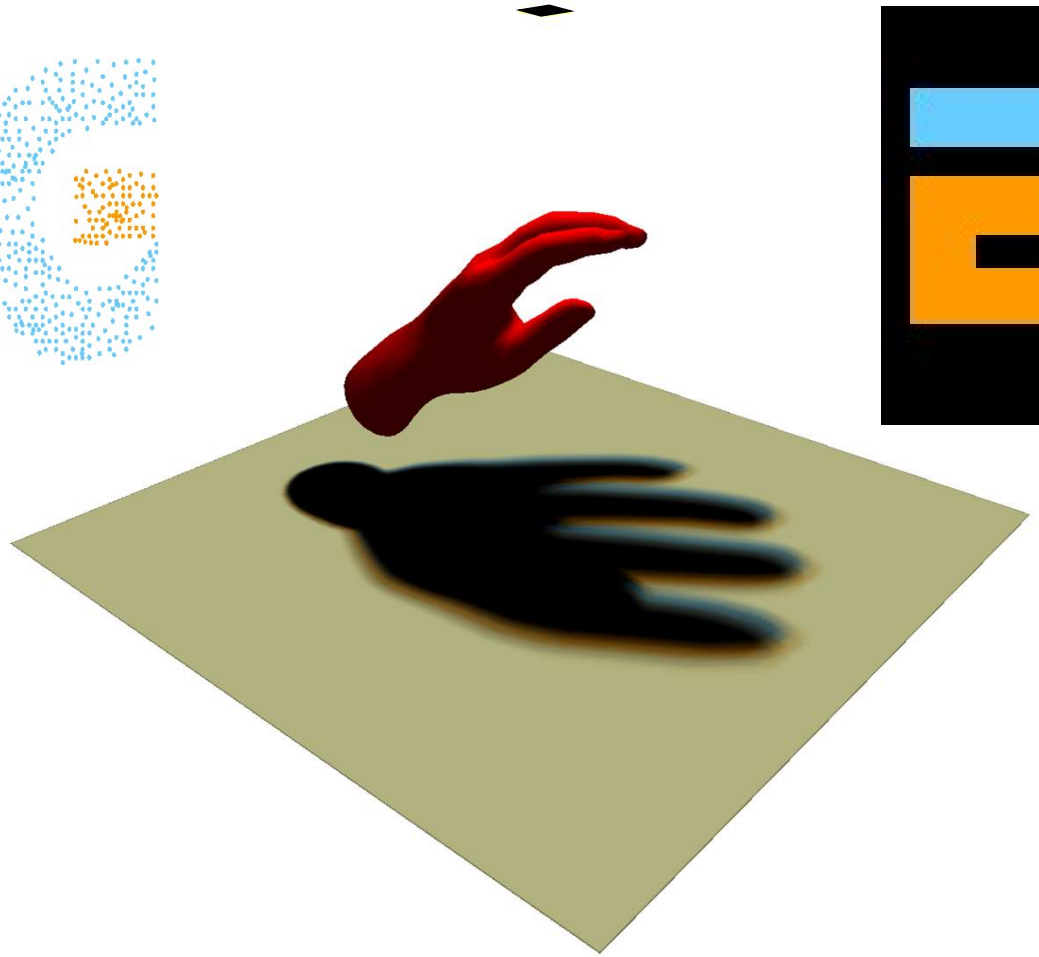
Principle

- Basic algorithm



Accumulate over all triangles

Soft Shadows



Accurate View-Sampling Methods

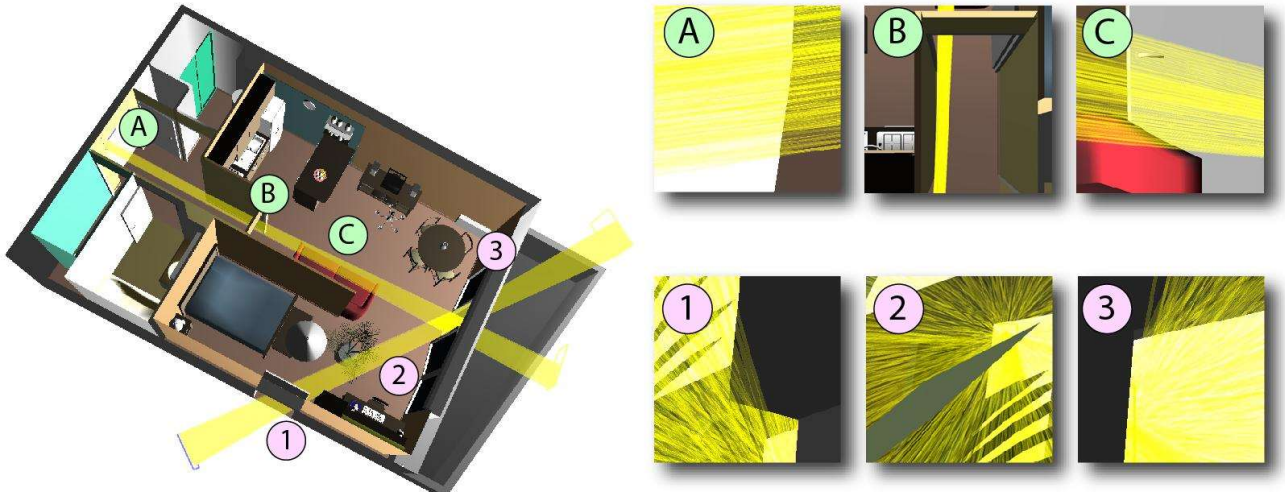
+ Fastest solution for accurate soft shadows

+ Artifact free

+ Physically correct (enough)

- Relatively slow

Many other applications!



Visibility Visualization

Is that all there is? Questions?

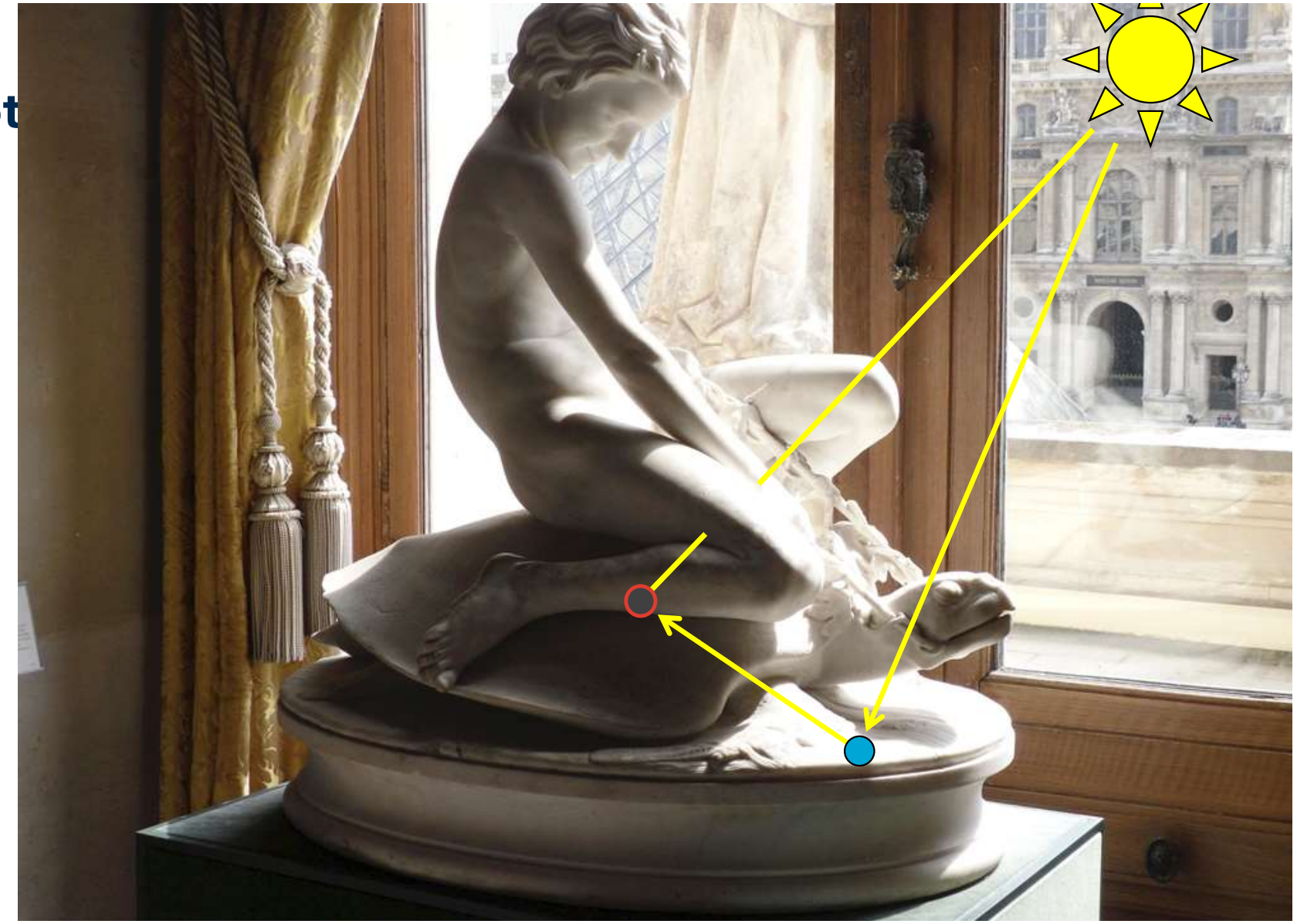
Summary

- Soft Shadow Methods:
  - Approximate soft shadows
    - Very fast
    - Often image-based methods
  - Accurate soft shadows
    - Relatively slow
    - Needs to be geometry-based
    - Many other applications

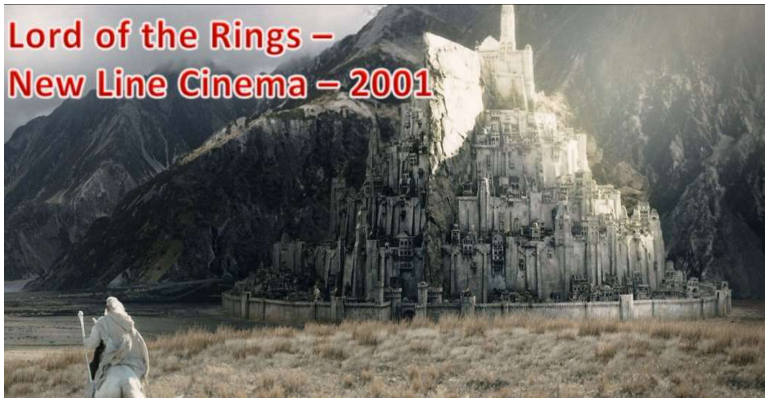
Now we have really realistic images, no?

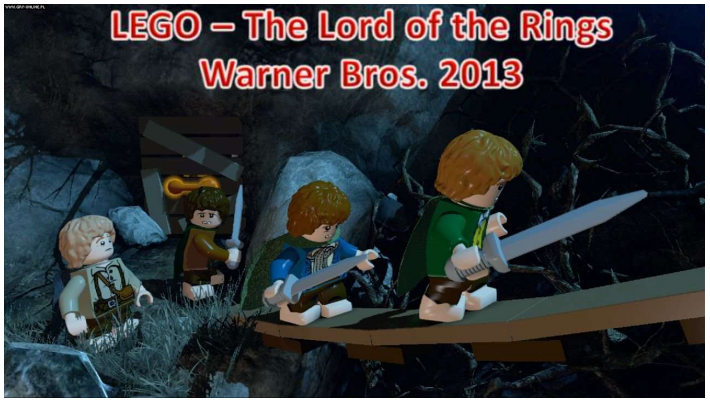
No!

Phot



Offline vs. Real time



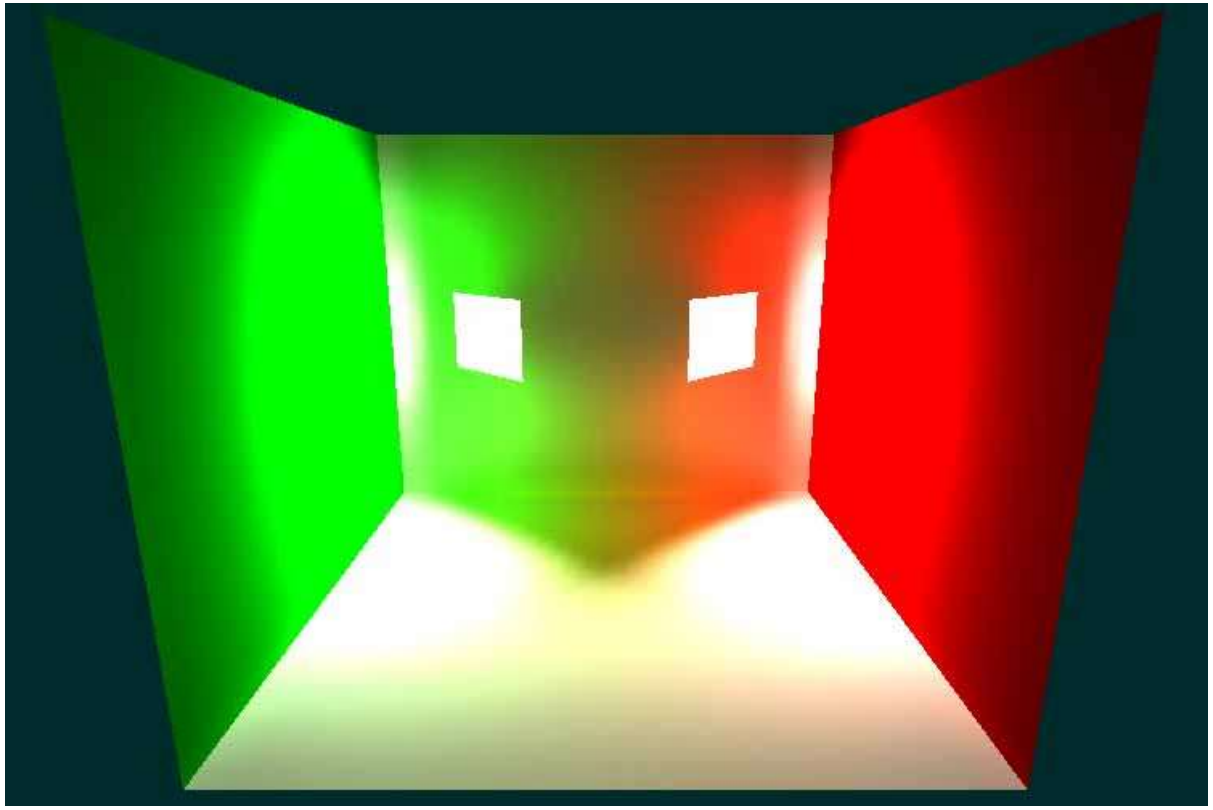


I personally never saw light bounce!

One Trillion Frames Camera

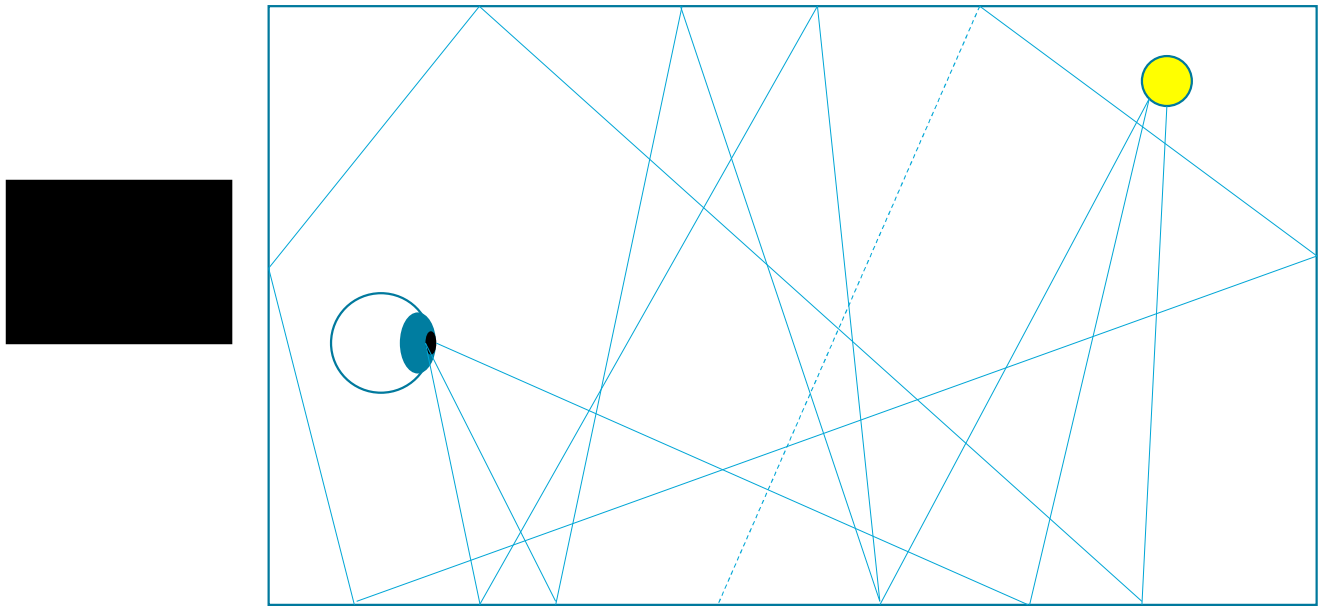
- R. Raskar
  MIT (https://www.youtube.com/watch?v=Y_9vd4HWlVA)

Global Illumination



Theoretical Solution

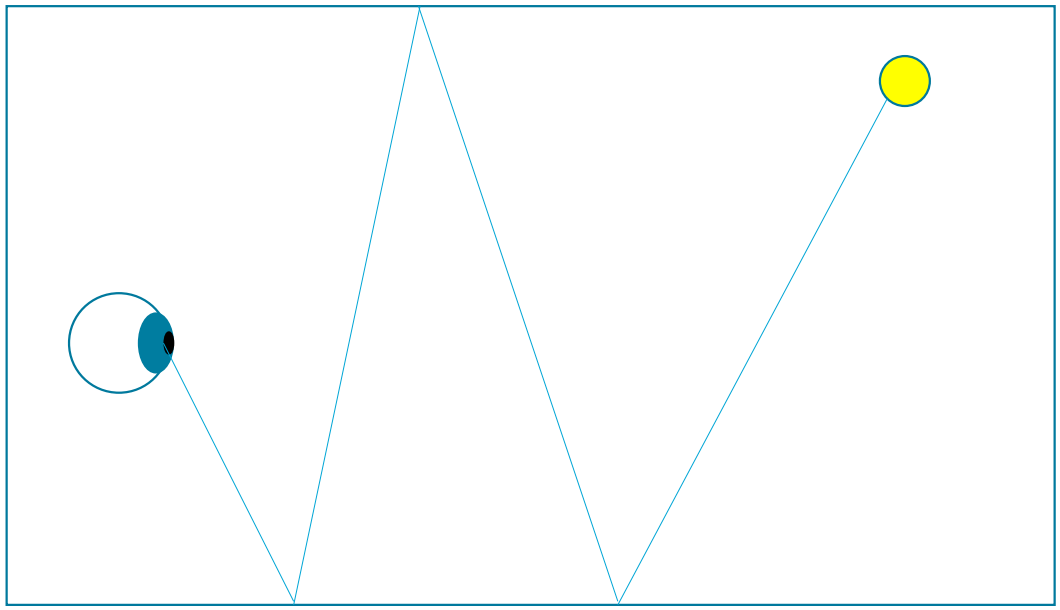
- Shoot random paths from the light and wait until they hit the camera…



- Not possible: likelihood to reach camera is ~0!
- Light is reduced at each reflection…

Path Tracing

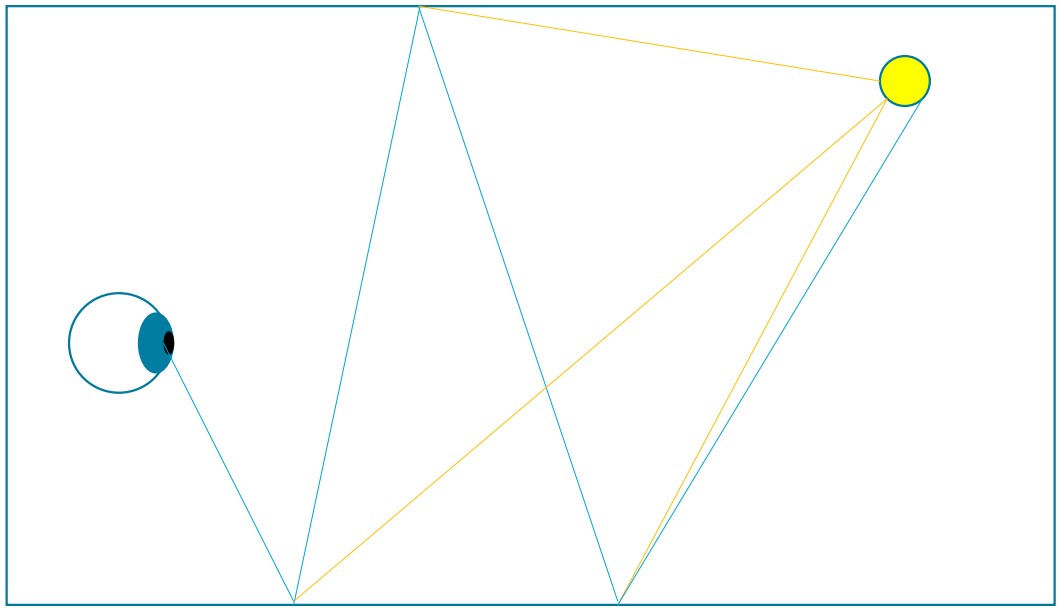
- Idea: create paths from camera



- Still hard to hit light but it is typically at least not a point!

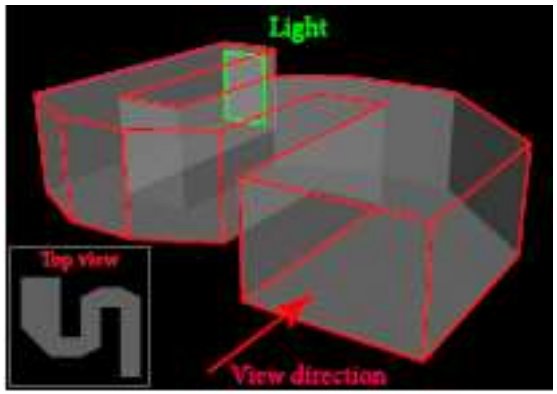
Path Tracing + Next Event Estimation

- Idea: create paths from camera

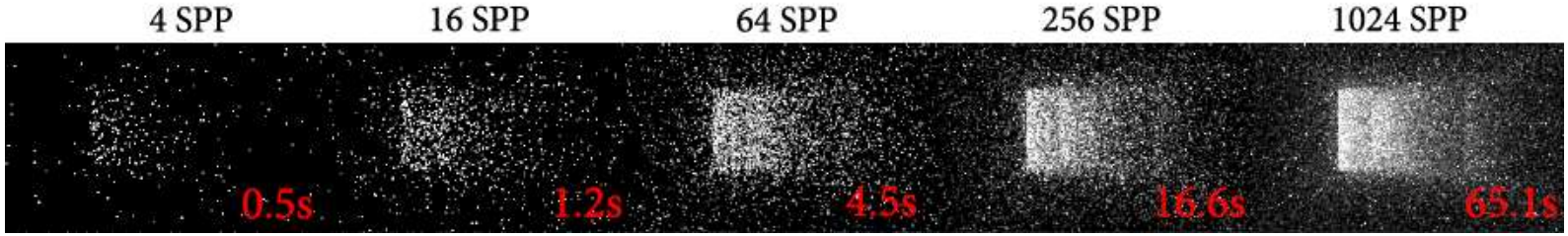


Path Tracing + Next Event Estimation

- Many paths/samples per pixel (spp) needed…



102.400 SSP



Rendering Cost is high!

- 76K hours

© Disney

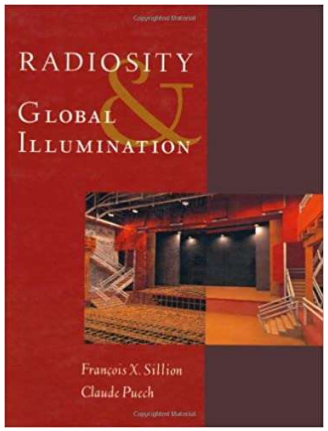
Outlook:

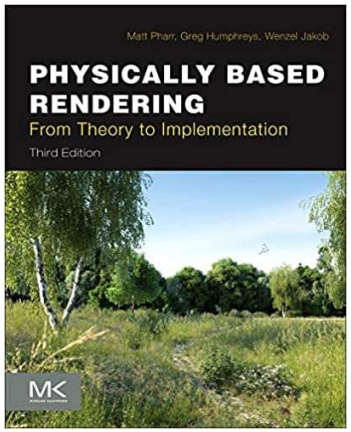
- Today: Global Illumination using the Radiosity method
- Next week: Stochastic Methods

TU Delft

Disclaimer

- Global Illumination can fill an entire course
- A simplified description follows…
- If you are interested in a thorough introduction to physics and basic algorithms
- Please refer to:





Rendering Equation [Kajiya 86]

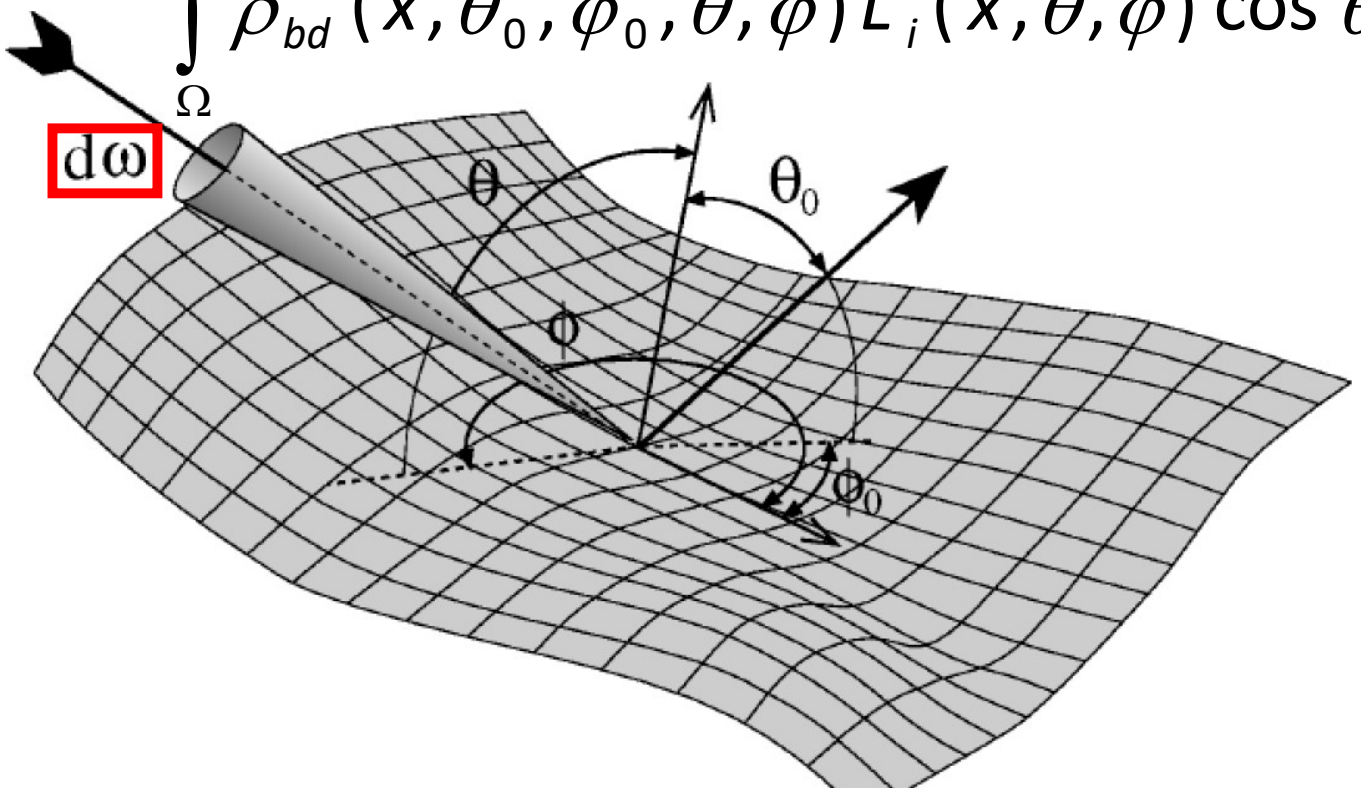
- Energy balance:

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$

- Total radiance= emitted radiance + reflected radiance

Rendering Equation [Kajiya 86]

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_{\Omega} \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta \, d\omega$$

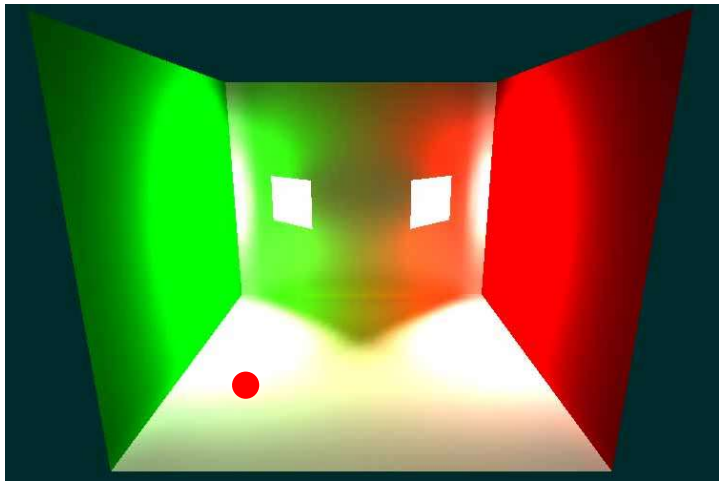


Rendering Equation

- Position

Spatial domain

$$L(x, \theta_0, \varphi_0) = L_e(x, \theta_0, \varphi_0) + \int_\Omega \rho_{bd}(x, \theta_0, \varphi_0, \theta, \varphi) L_i(x, \theta, \varphi) \cos\theta d\omega$$

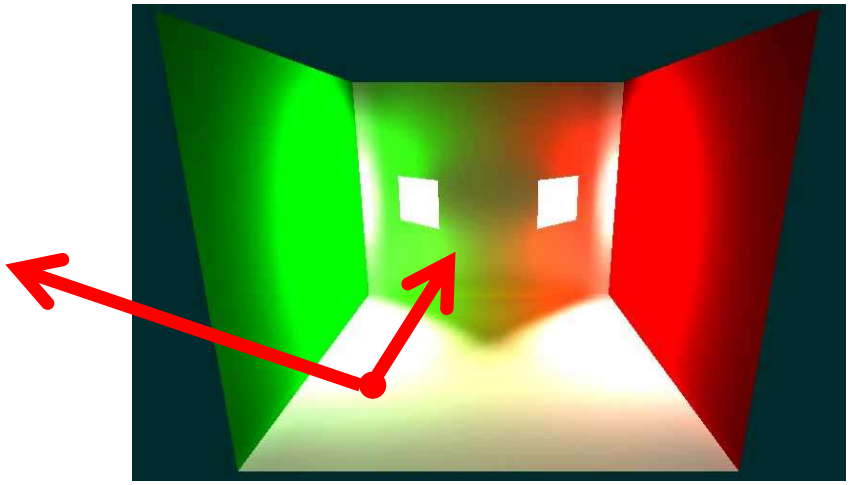


Rendering Equation

- Orientation

Angular domain

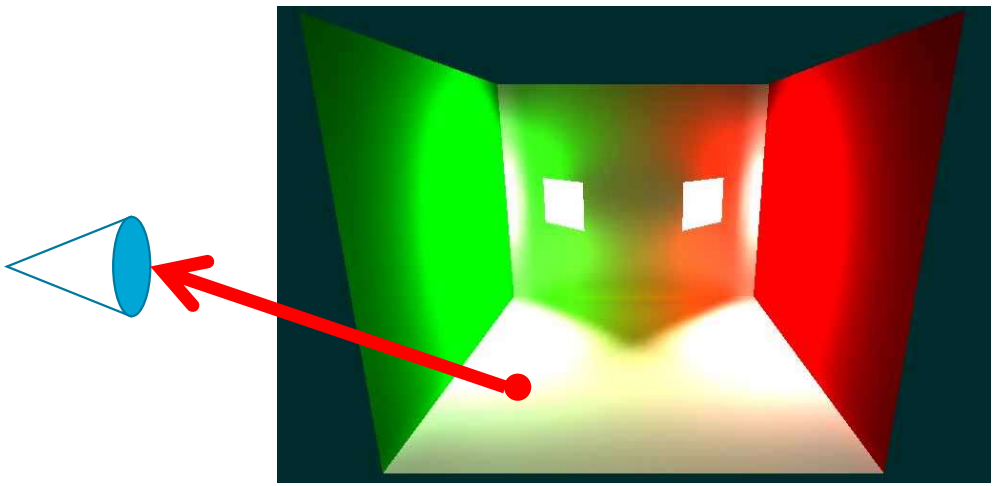
$$L(x, \theta_0, \varphi_0) = L_e(x, \theta_0, \varphi_0) + \int_\Omega \rho_{bd}(x, \theta_0, \varphi_0, \theta, \varphi) L_i(x, \theta, \varphi) \cos\theta d\omega$$



Rendering Equation

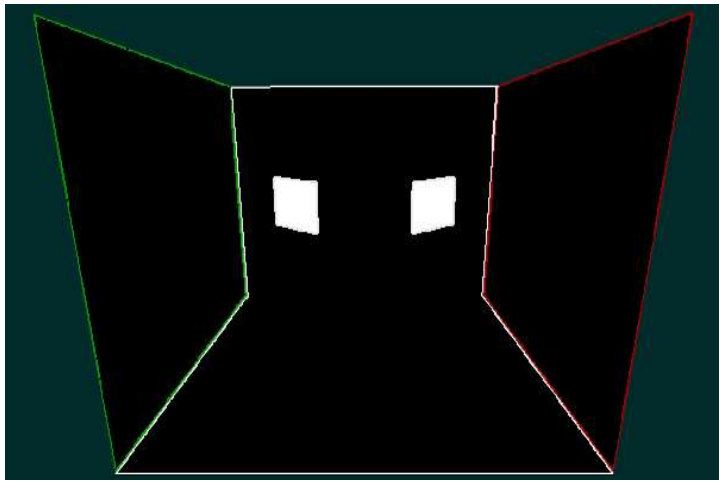
- Radiance

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$



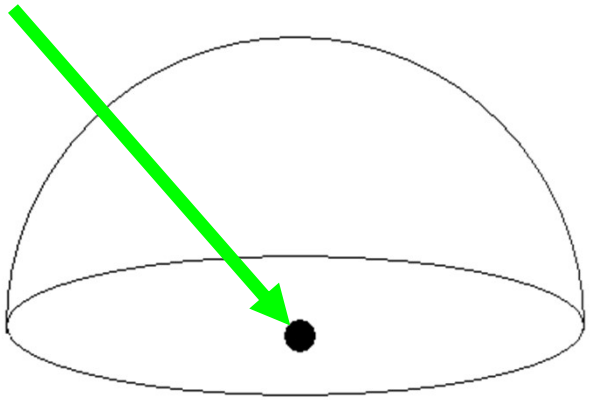
Rendering Equation

$$L(x,\theta_0,\varphi_0) = \boxed{L_e(x,\theta_0,\varphi_0)} + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$



Rendering Equation

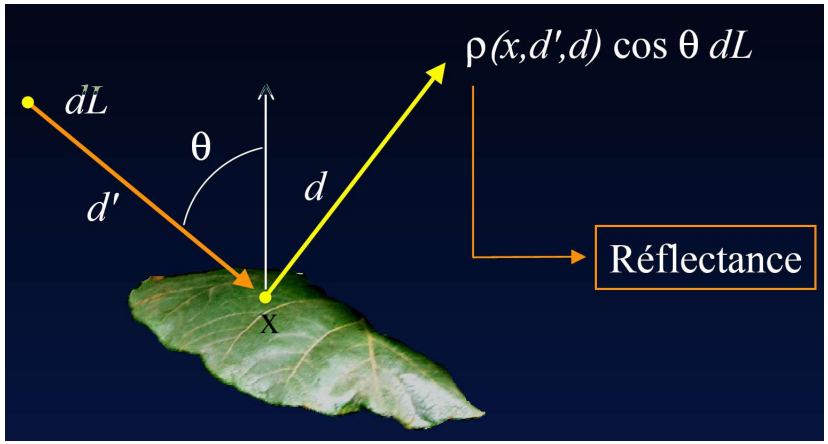
$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta \, d\omega$$



Rendering Equation

- Bidirectional Reflectance Distribution Function (BRDF)

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$

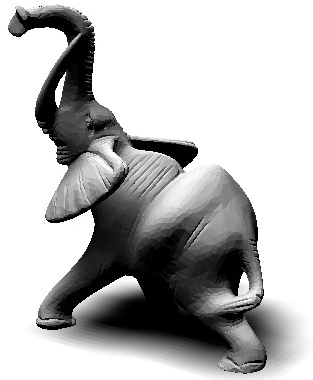


Example: Ideal Diffuse/Lambertian Surface

- Reflected light is identical in all directions
- e.g., “Phong Model without specular”

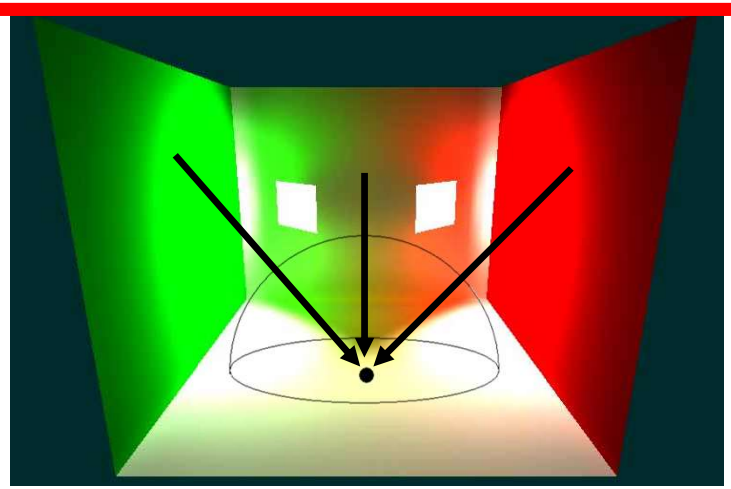
$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$

$$\rho_{bd}(\theta_0,\varphi_0,\theta,\varphi) \equiv \rho$$



Rendering Equation

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$



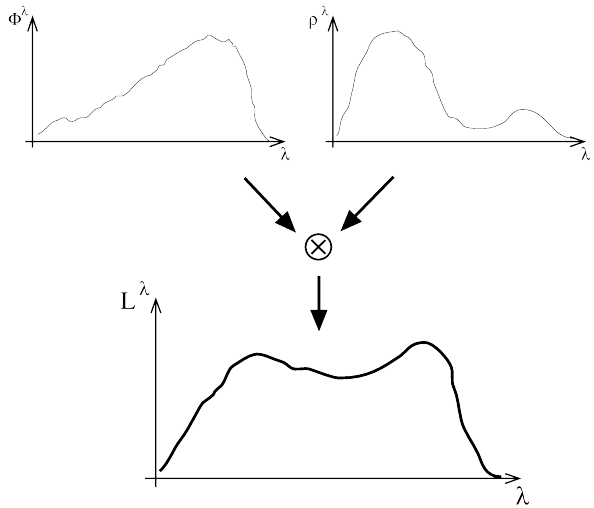
Everything understood?

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$

The awful truth…

- Everything depends on the wavelength!

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$



- You have heard this before…

  Let's just forget about it…☺

(i.e. think of red, green, blue)

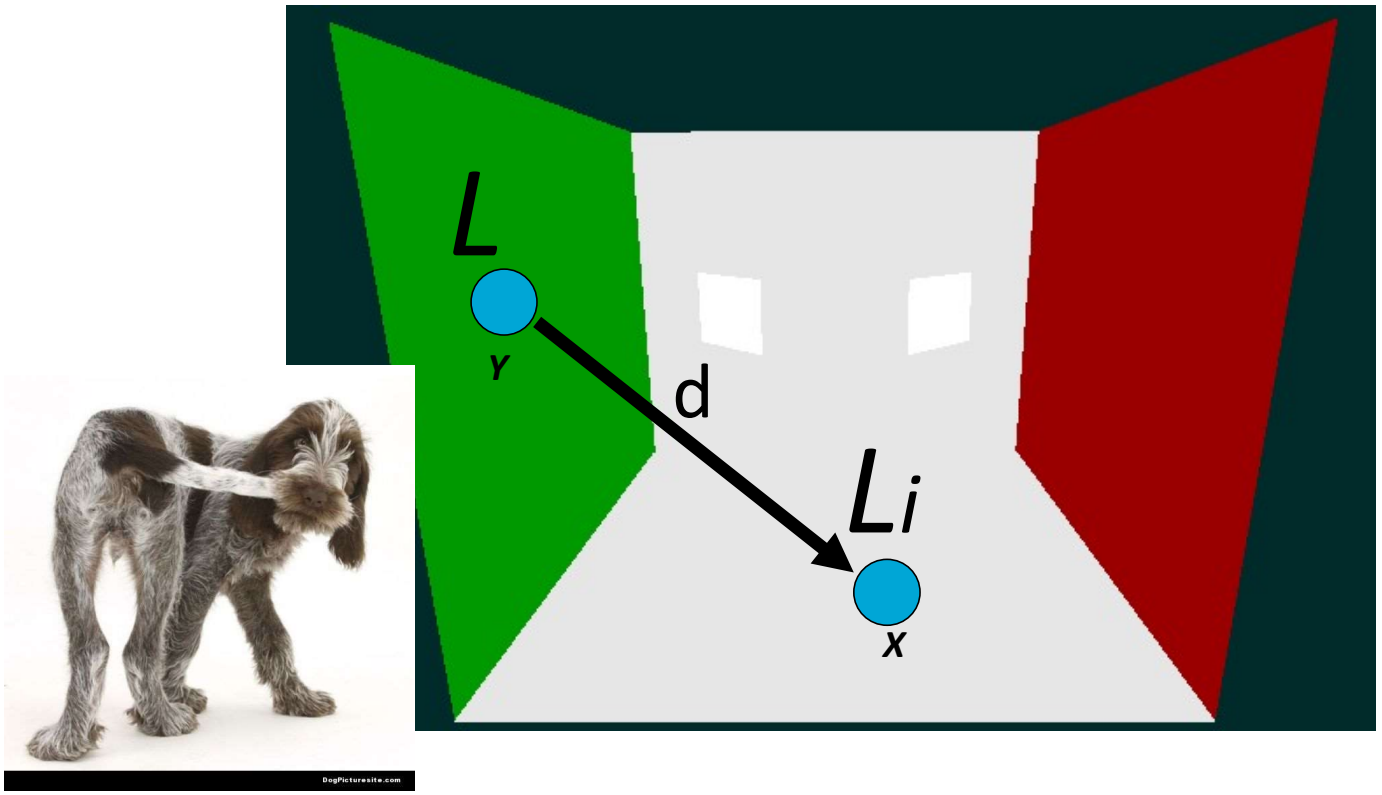
Rendering Equation - hard to solve

- Energy balance:

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$

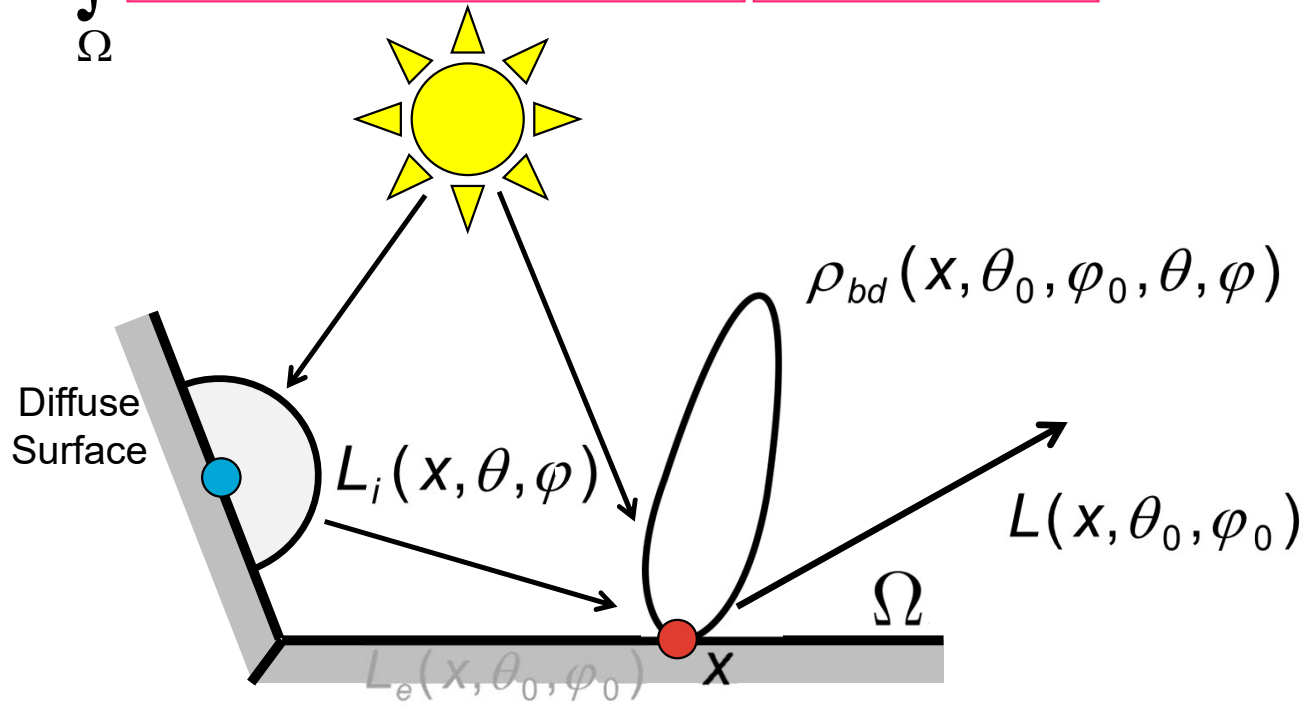
Rendering Equation - hard to solve

$$L_i(X,Y \rightarrow X) = L(Y,\ Y \rightarrow X)$$



Rendering Equation – Recap

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi)\, L_i(x,\theta,\varphi)\cos\theta d\omega$$



Questions?


..WHY DID YOU LEAVE YOUR LAST JOB?
...THE COMPANY RELOCATED AND DIDN'T TELL ME WHERE.

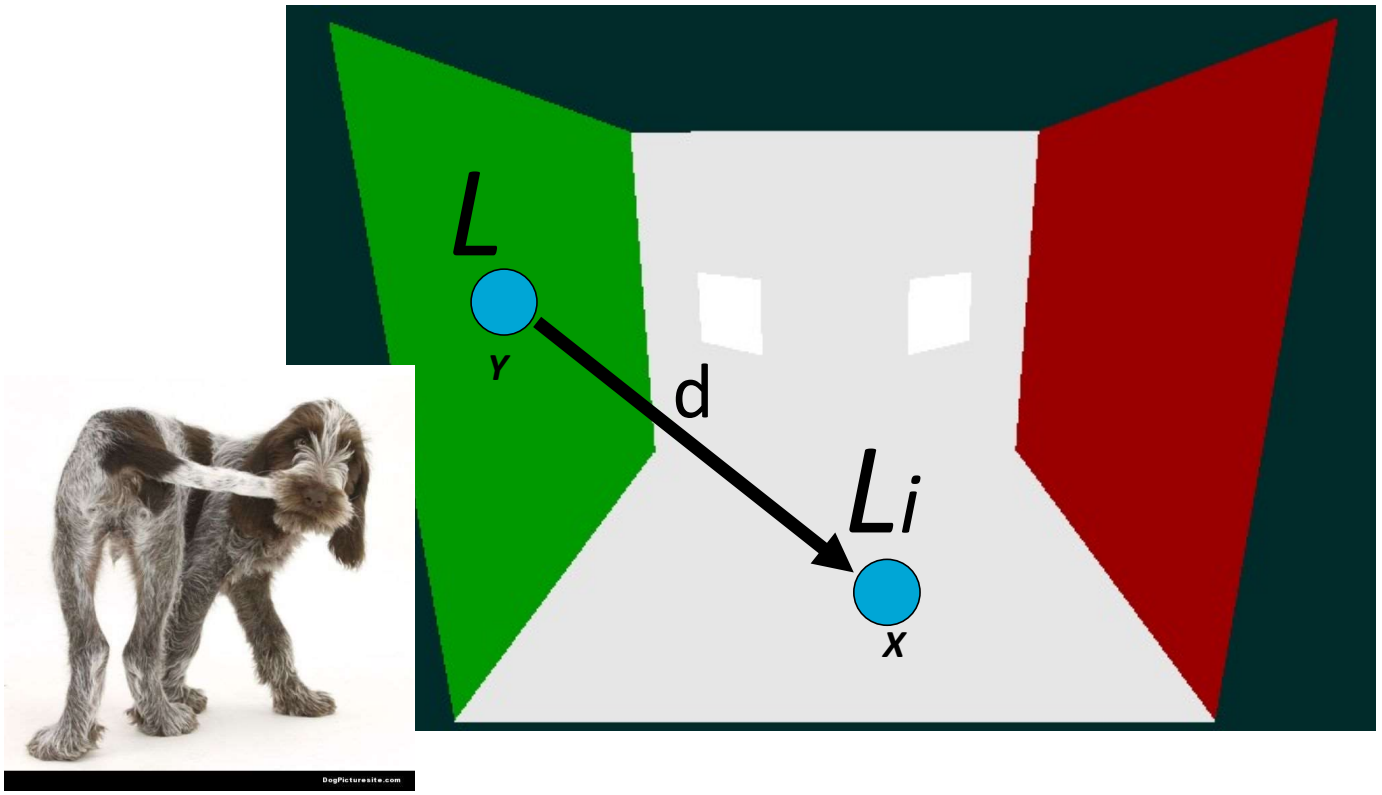
Formal Solution of Rendering Equation

- *Reflection* operator R
  - Use radiance distribution L to define L_i and integrate contribution in each point

$$(RL)(x,\theta_0,\varphi_0) := \int_{\Omega} \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$

Rendering Equation - Recursion

$$L_i(X,Y \rightarrow X) = L(Y,\ Y \rightarrow X)$$



Formal Solution of Rendering Equation

$$(RL)(x,\theta_0,\varphi_0) := \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi)\cos\theta d\omega$$

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi)\cos\theta d\omega$$

$$= L_e(x,\theta_0,\varphi_0) + (RL)(x,\theta_0,\varphi_0)$$

Formal Solution of Rendering Equation

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + (RL)(x,\theta_0,\varphi_0)$$

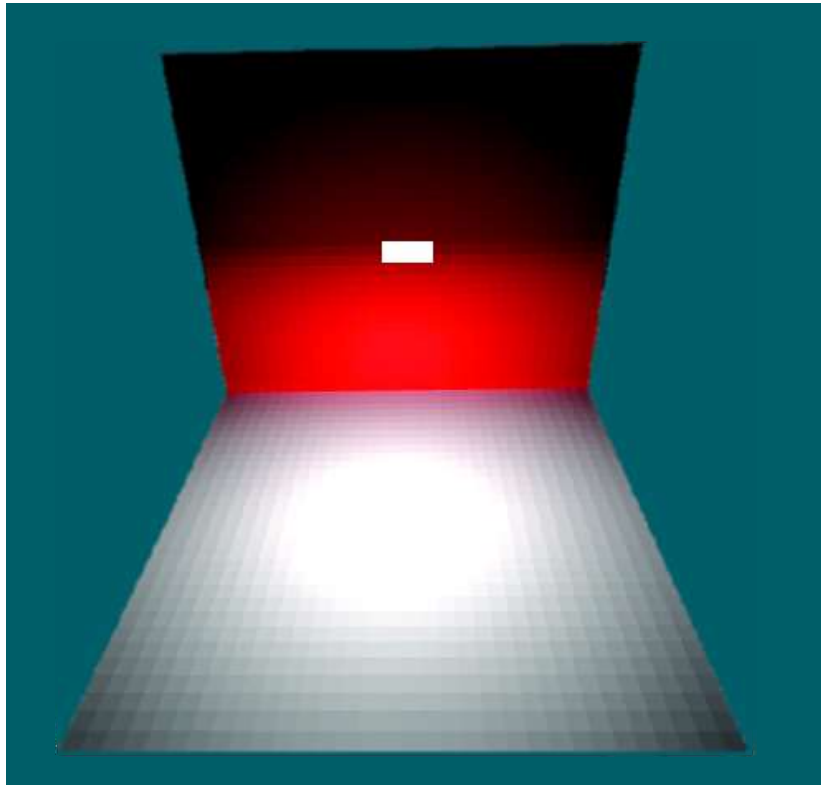
- The equation becomes: $L = L_e + RL$
- Therefore: $L = [I - R]^{-1} L_e$
- Using Neumann Series: $L = \sum_{n=0}^{\infty} (R^n) L_e$ What does this equation represent?

Physical Interpretation

- Emitted radiance (L_e)…
- Plus reflected radiance once (RL_e)…
- Plus twice reflected radiance (R^2L_e)…
- Plus three times reflected radiance (R^3L_e)…

$$L = \sum_{n=0}^{\infty} (R^n) L_e$$

Physical Interpretation



$$L_e + (RL_e) + (R^2L_e) + (R^3L_e) + \dots$$

In practice, R...

- ...has no analytical solution

- BUT:
  If we can approximate R, then we can approximate one bounce,
  and, thus, several bounces by repeatedly applying R

Questions?

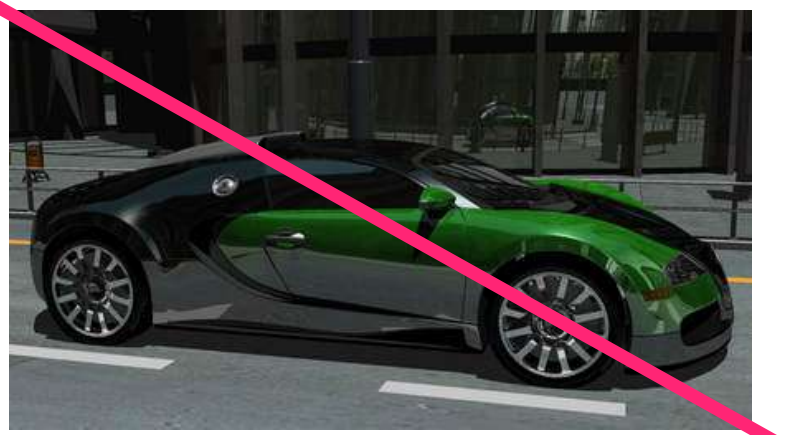

"Remember class, there is no such thing as a, STUPID QUESTION"
*Teacher
CHALLENGE ACCEPTED

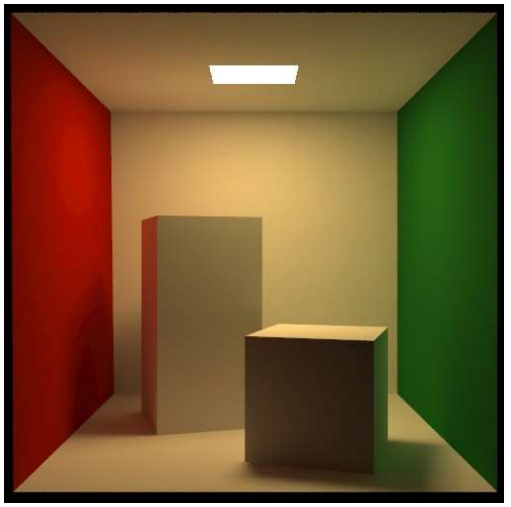
How can we make our life simpler?

- Radiosity solutions work with 2 simplifications:
  - Only diffuse objects are allowed
  - We discretize the scene in patches and assume constant illumination on each patch

Radiosity

- First simplification hypothesis:
  - All surfaces are diffuse





Radiosity

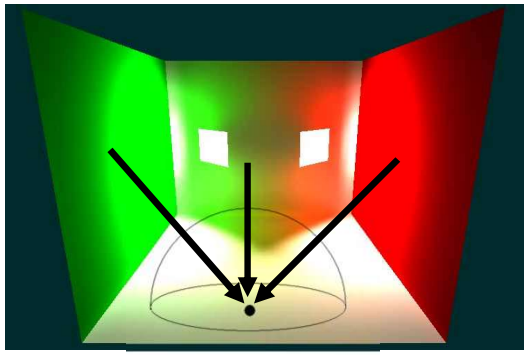
- First simplification hypothesis:
  - All surfaces are diffuse

$$L(x) = L_e(x) + \int_{\Omega} \rho_{bd}(x) L_i(x, \theta) \cos\theta d\omega$$

Radiosity

- First simplification hypothesis:
  - All surfaces are diffuse
  - Radiosity is traditionally denoted with B and E

$$B(x) = E(x) + \rho_{bd}(x) \int_{\Omega} B_i(x,\theta) \cos\theta \, d\omega$$

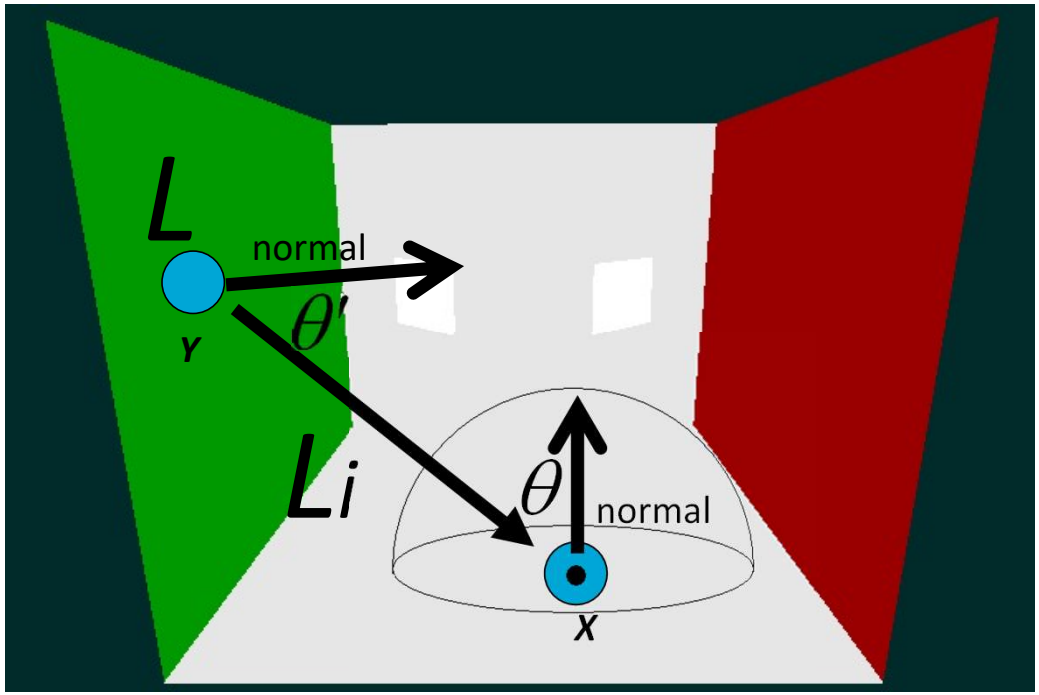


Change of variables:
Instead of directions,
integrate over surfaces

$$\int_{y \in S} B(y) V(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} dy$$

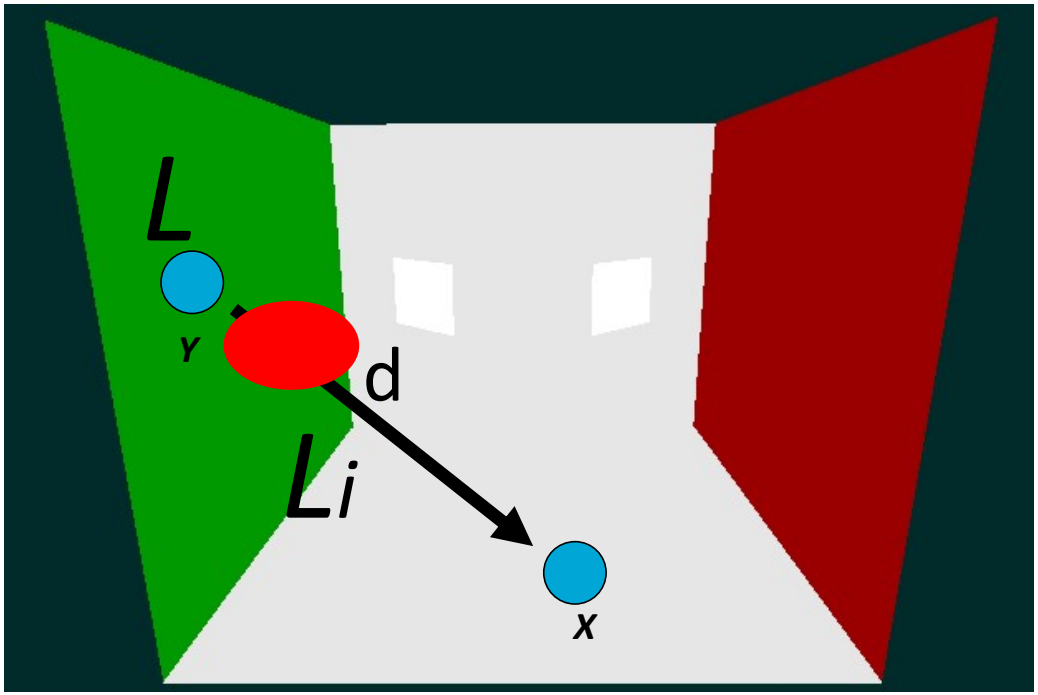
What are those angles?

$$B(x) = E(x) + \rho_d(x) \int_{y \in S} B(y) V(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} dy$$



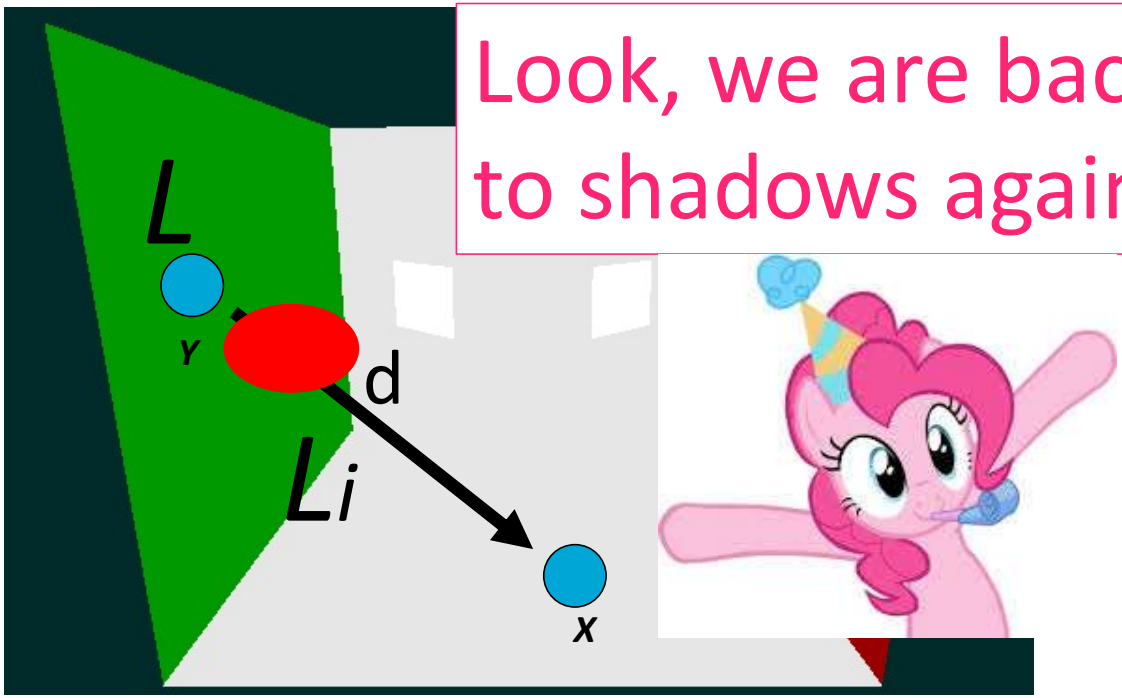
Reparametrization

$$B(x) = E(x) + \rho_d(x) \int_{y \in S} B(y) V(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} dy$$



Reparametrization

$$B(x) = E(x) + \rho_d(x) \int_{y \in S} B(y) V(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} dy$$



Look, we are back to shadows again!

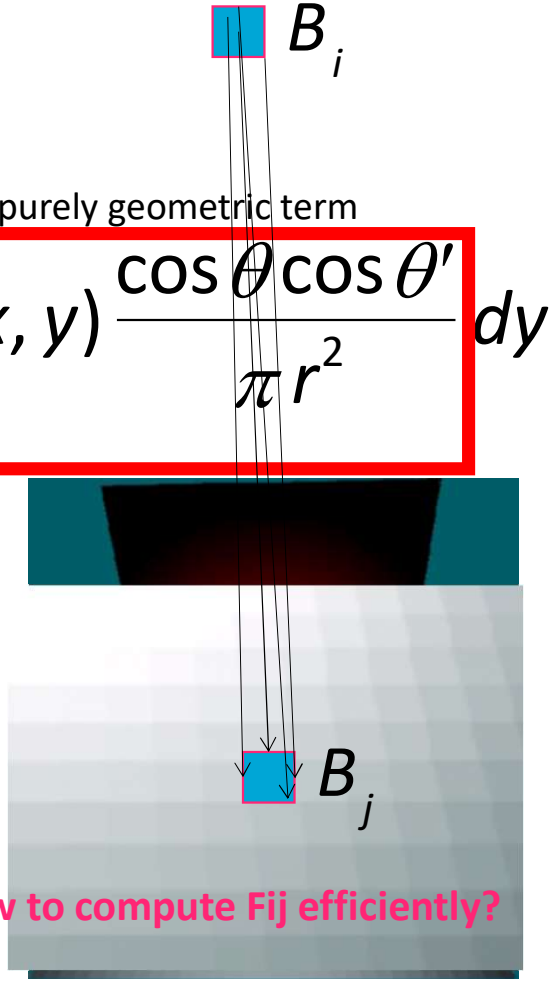
Radiosity

- 2nd hypothesis:

  piecewise constant patches

$$B(x) = E(x) + \rho_d(x) \int_{y \in S} B(y) V(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} dy$$

purely geometric term



F_{ij} describes ratio of energy reaching patch i from patch j

Discretizing this equation:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

Radiosity – "average" energy on patch i

How to compute Fij efficiently?

Form Factor

- F_{ij} : form factor
  - Purely geometric
  - Relation of transmitted energy between two patches

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

Form Factor

- F_{ij} : form factor
  - Purely geometric
  - Relation of transmitted energy between two patches

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

$$\frac{1}{\text{area(PatchX)}} \int_{x \in PatchX} \int_{y \in PatchY} V(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} dy$$

- Computation methods:
  - Analytical solution (no visibility): Schröder&Hanrahan92

On the Form Factor Between Two Polygons

Lambert 1760: "... the complex calculation would bring even the most patient to be frustrated and abandon the task..."

*P Schröder, **P Hanrahan** "On the form factor between two polygons" Proc. Computer graphics and interactive techniques, 1993*

But not Peter Schröder en Pat Hanrahan…

$$\int_0^{c_2}\int_0^{c_0} \ln f(s,t)\,ds\,dt$$

$$= \left\{\left[(s+\frac{c_3}{2})\mathrm{G}(f(.,t))(s)+\frac{c_1}{2}\mathrm{H}(f(.,t))(s)\right]\Big|_{s=0}^{s=c_0}\right\}\Big|_{t=0}^{t=c_2}$$

$$-2c_0c_2 + c_{14}c_{15}\left\{\left[\pi(2k(s)+1)\mathrm{M}(t)\right.\right.$$

$$-i\left(\mathrm{L}(-c_{17}(s))(t)+\mathrm{L}(-c_{18}(s))(t)\right.$$

$$\left.\left.-\mathrm{L}(c_{17}(s))(t)-\mathrm{L}(c_{18}(s))(t)\right)\right]\Big|_{s=0}^{s=c_0}\right\}\Big|_{t=\sqrt{\frac{c_{13}}{c_{13}+c_2}}}^{t=\sqrt{\frac{c_{13}+c_2}{c_{13}}}}$$

where $k(s) \in \{-1,0,1\}$ according to the particular branchcut of the complex logarithm choosen in L. The auxiliary functions G, H, L, and M are given in Table 1.

This does not consider visibility!

$$c_0 = \|E_j\|$$
$$2\vec{d}_i \cdot \vec{d}_j$$
$$\|E_i\|$$
$$2\vec{d}_j \cdot (\vec{p}_i - \vec{p}_j)$$
$$\vec{d}_i \cdot (\vec{p}_i - \vec{p}_j)$$
$$\|\vec{p}_i - \vec{p}_j\|^2$$
$$-c_1^2$$
$$\ldots - 2c_1c_3$$
$$\ldots - c_3^2$$

$$c_{13} = \frac{c_{11}-\sqrt{c_{11}^2-4c_{10}c_{12}}}{2c_{10}}$$
$$c_{14} = \frac{\sqrt{c_{11}^2-4c_{10}c_{12}}}{c_{10}}$$
$$c_{15} = \sqrt{c_{10}}c_{14}$$
$$c_{16}(s) = c_1c_{13} - c_3 - 2s$$
$$c_{17}(s) = \frac{-c_{15}+\sqrt{c_{15}^2-4|c_{16}(s)|^2}}{2\sqrt{c_{10}}c_{16}(s)}$$
$$c_{18}(s) = \frac{-c_{15}-\sqrt{c_{15}^2-4|c_{16}(s)|^2}}{2\sqrt{c_{10}}c_{16}(s)}$$

expressions for two edges E_{ij} with parameterization $\vec{d}_i$ and $\vec{x}_j(s) = \vec{p}_i + s\vec{d}_i$ ($\|\vec{d}_{i,j}\| = 1$).

$$\mathrm{L}(b)(y) := \int^y t^2(1-t^2)^{-3}\ln(b+t)\,dt = \frac{1}{16}\left[\frac{-b\ln(y-1)}{(b+1)^2} - \frac{b\ln(1+y)}{(b-1)^2} + \left(\frac{2(b+y)((1+by)((b-y)^2+(by-1)^2]}{(b^2-1)^2(y^2-1)^2} + \ln\frac{(1-y)(1-b)}{(1+y)(1+b)}\right)\ln(b+y)\right.$$
$$\left. + \frac{2(b-y)}{(b^2-1)(y^2-1)} + \mathrm{Li}_2\left(\frac{1-y}{1+b}\right) - \mathrm{Li}_2\left(\frac{1+y}{1-b}\right)\right]$$

$$\mathrm{M}(y) := \int^y t^2(1-t^2)^{-3}\,dt = \frac{1}{16}\left[4y(y^2-1)^{-2} + 2y(y^2-1)^{-1} + \ln\frac{y-1}{y+1}\right]$$

$$\mathrm{G}(q)(y) := \int^y \ln q(t)\,dt = \frac{q'(y)}{2a}\ln q(y) - 2y + \frac{d}{a}\tan^{-1}\frac{q'(y)}{d}$$

$$\mathrm{H}(q)(y) := \int^y t\ln q(t)\,dt = \left(\frac{y^2}{2} + \frac{c}{2a} - \frac{b^2}{4a^2}\right)\ln q(y) - \frac{y(ay-b)}{2a} - \frac{bd}{2a^2}\tan^{-1}\frac{q'(y)}{d}$$

Table 1: Four auxiliary integrals needed in the solution. Notice that $\mathrm{L}(b)(y)$ uses the dilogarithm [1], $\mathrm{Li}_2(z) = \sum_1^\infty \frac{z^k}{k^2}$, $\frac{d}{dz}\mathrm{Li}_2(z) = -\frac{\ln(1-z)}{z}$. In G and H the argument q is an arbitrary quadratic polynomial $q(t) = at^2 + bt + c$ and $d = \sqrt{4ac - b^2}$.

Approximate the Form Factor…

- F_{ij} : form factor
  - Purely geometric
  - Relation of transmitted energy between two patches

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

$$F_{ij} = \frac{1}{\text{area(Patch i)}} \int_{x \in \text{Patch } i} \int_{y \in \text{Patch } j} V(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} dy$$

OMG, we are back to visibility between two patches!

Rewriting the Equation

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

- We regroup the elements of the equation:

$$\begin{bmatrix} B_0 \\ \\ B_n \end{bmatrix} = \begin{bmatrix} E_0 \\ \\ E_n \end{bmatrix} + \begin{bmatrix} \rho_i F_{ij} \end{bmatrix} \begin{bmatrix} B_0 \\ \\ B_n \end{bmatrix}$$

- Or using vector notation: $\mathbf{B} = \mathbf{E} + \mathbf{MB}$

We have come a LONG way:

- From here:

$$L(x,\theta_0,\varphi_0) = L_e(x,\theta_0,\varphi_0) + \int_\Omega \rho_{bd}(x,\theta_0,\varphi_0,\theta,\varphi) L_i(x,\theta,\varphi) \cos\theta d\omega$$

- We got here: $\mathbf{B = E + MB}$

One question remains: How to find B?

It's linear algebra! Thus, lecture 2!

Solving the System?

Matrix equation: $\mathbf{B = E + MB}$

Possible solution:

Use Linear Algebra…

$B=E+MB$

$\Leftrightarrow$ $B-MB=E$

$\Leftrightarrow$ $(I-M)B=E$

$\Leftrightarrow$ $\mathbf{B = (I-M)^{-1}E}$

Unfortunately, M can be really large;

e.g., 10^6 patches then M has 10^{12} entries

Solving the System?

Matrix equation: $\mathbf{B = E + MB}$

1. solution

Use Linear Algebra…

$$\mathbf{B = (I-M)^{-1}E}$$

2. solution: Don't compute any inverse…

$\mathbf{B = E + MB}$ ⟹ $B=E+M(E+MB)$

The function R(X):=E+MX has B as a fix point!

E… ME+E … $M^2E+ME+E$… $M^3E+M^2E+ME+E$

Radiosity

Radiosity

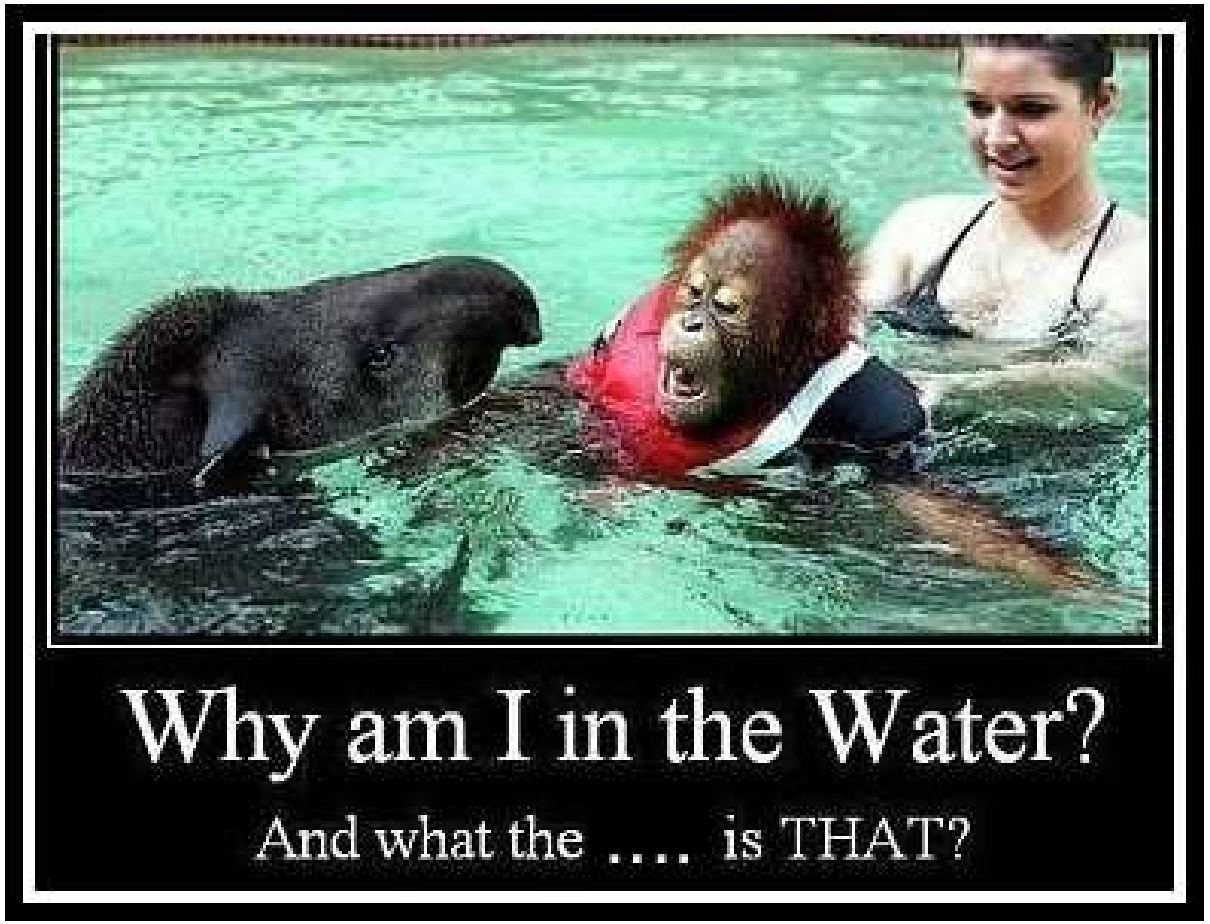
Summary

- Radiosity means solving a large linear system
  - Long computational time to set up the system
  - and long computation time to solve it

- STATIC SCENE!

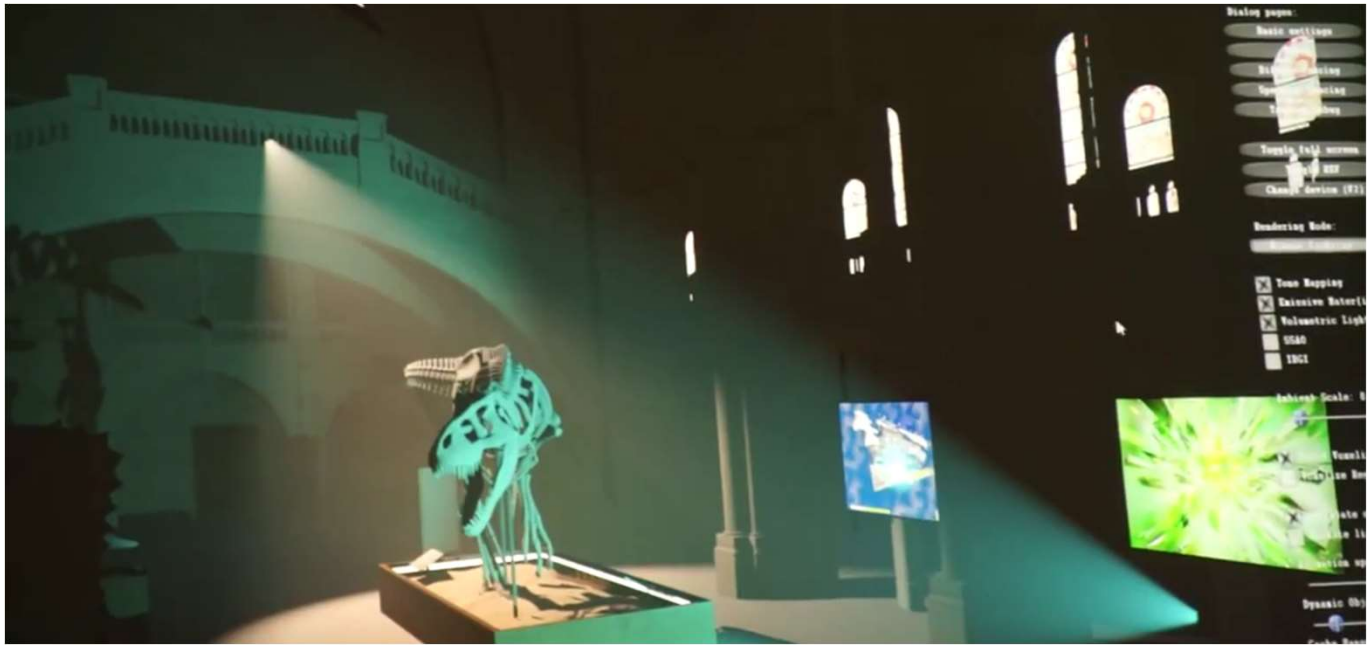
- Possible accelerations
  - Scene Structures
  - Hierarchical radiosity
  - Clustering
  - Instantiation

Questions?



So how do you do real-time GI?

- GI Works – © NVIDIA
- Based on our voxel cone tracing
  [Crassin, Neyret, Sainz, Green, Eisemann - Pacific Graphics 2011]



Thank you very much!