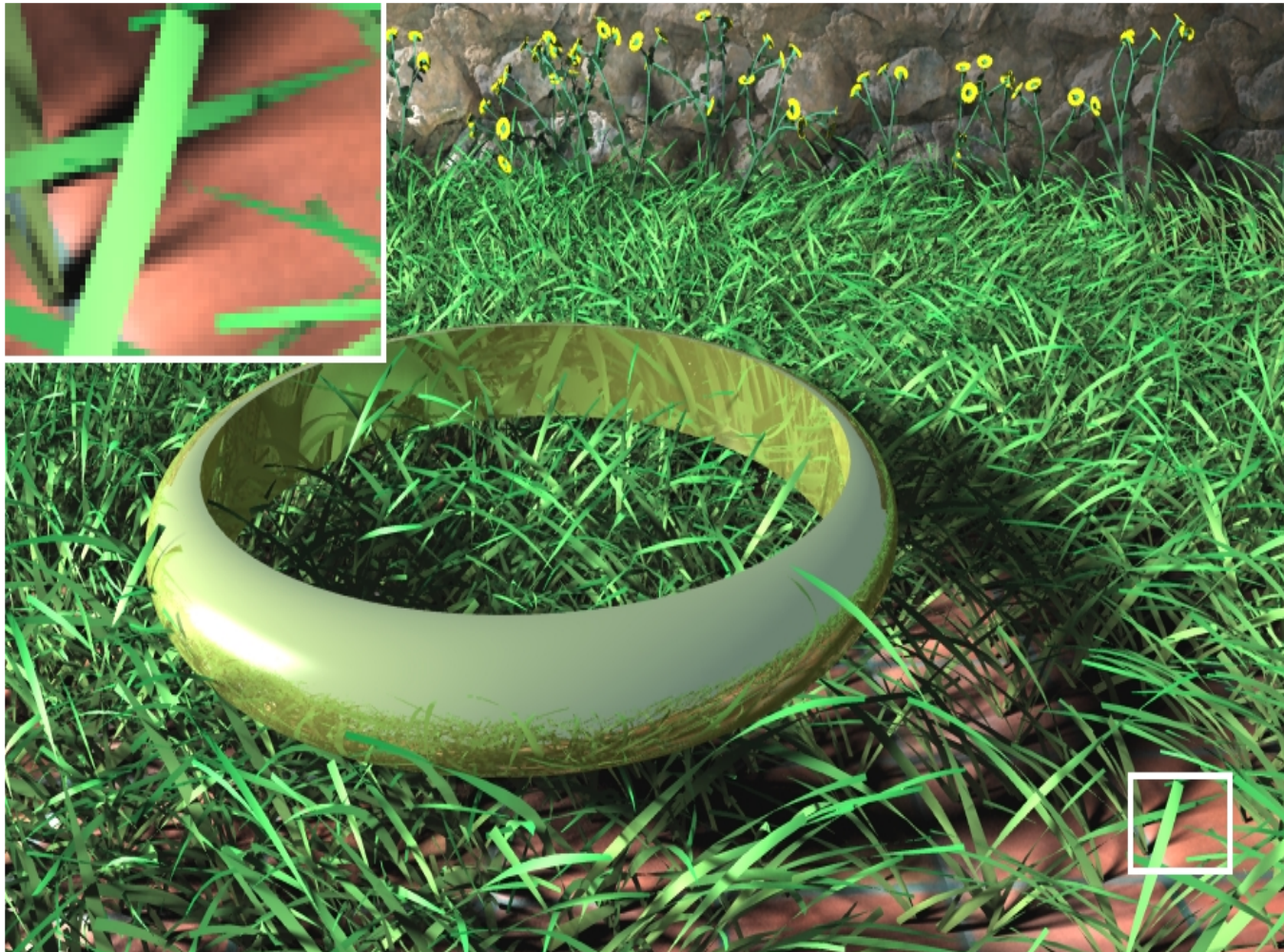




Soft Shadow Volumes for Ray Tracing

Samuli Laine, Timo Aila, Ulf Assarsson, Jaakko Lethinen, Tomas Akenine-Möller
presented by Manuel Lang

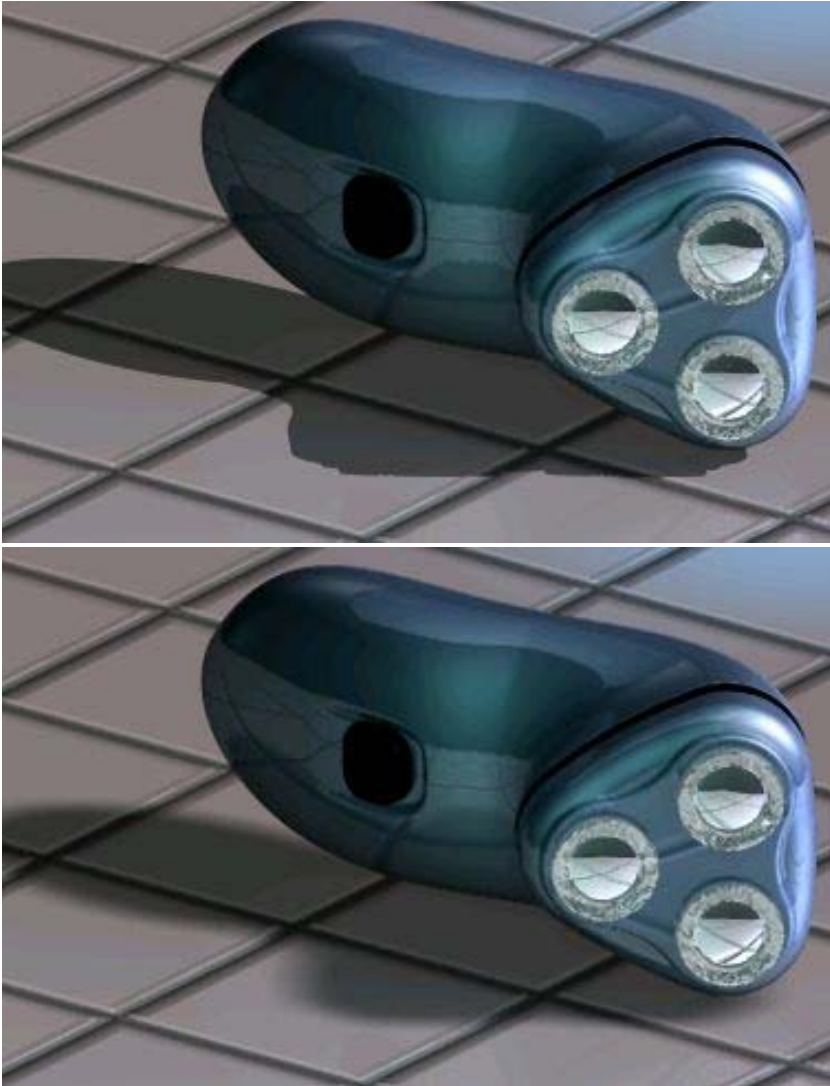




Outline of this presentation

- Introduction to Soft-Shadows
- Soft-Shadows techniques
- Silhouette Edge / Wedges / 3 SS Rules
- Acceleration Structure
- Light Integration
- Results
- Limits and Conclusion

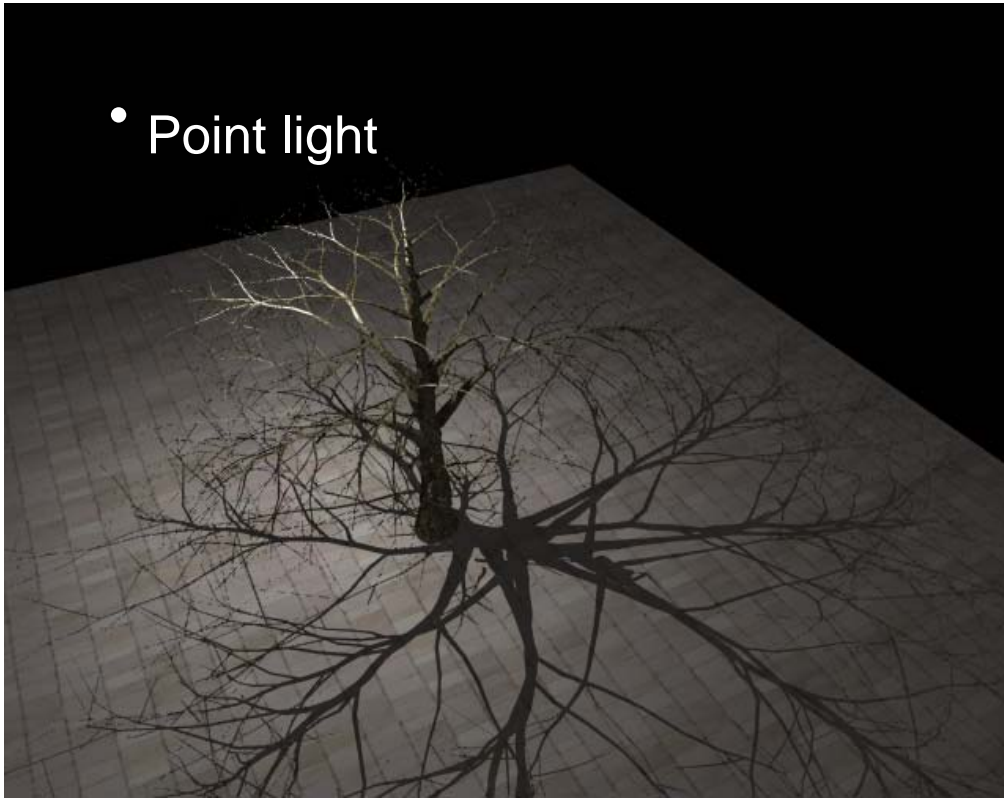
Why Soft Shadow?



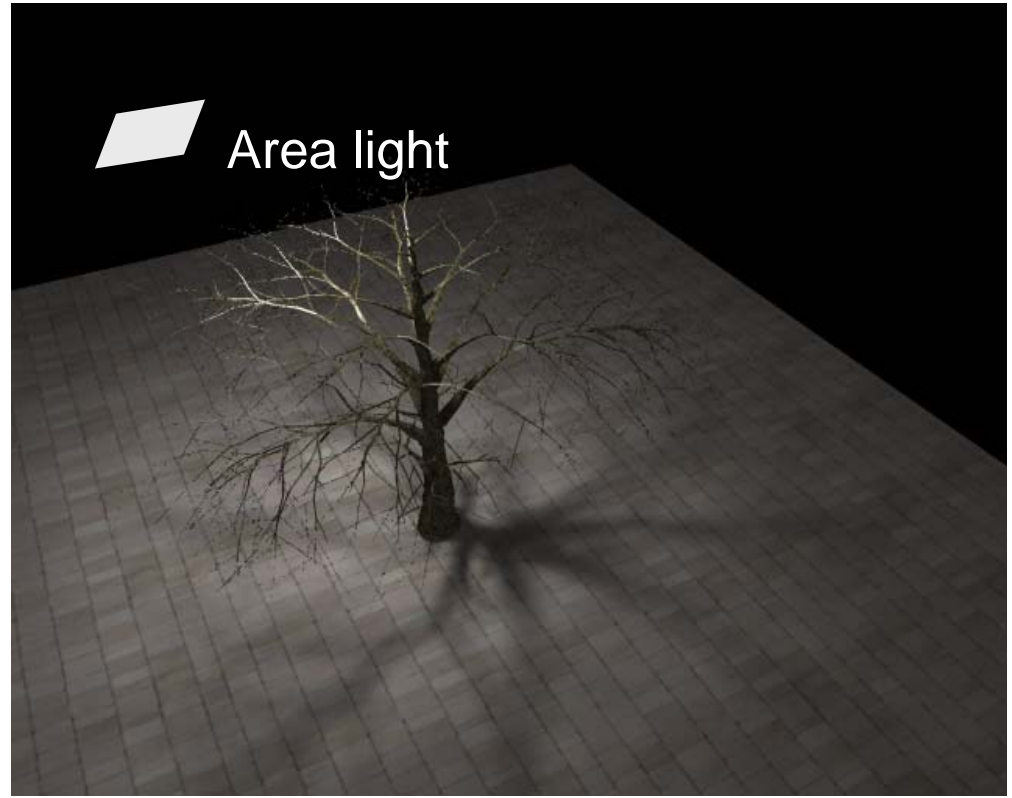
- Physically not possible to build a point light source
- All light sources in real world are **area lights**
- Area light sources produce soft shadow
- We expect soft shadow for “real looking” renderings

Why Soft Shadows

- Point light

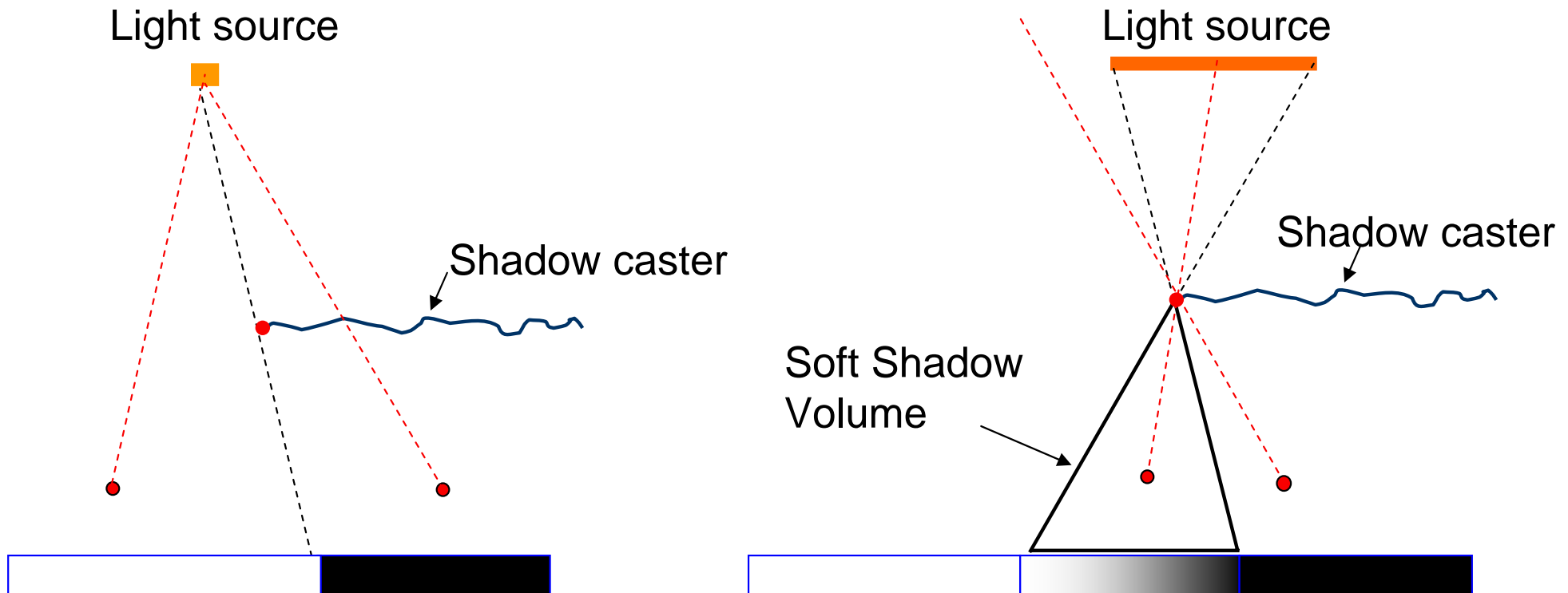


- Area light



- one more example

Area Light and Soft Shadows



- Incoming light intensity is proportional to visible light area

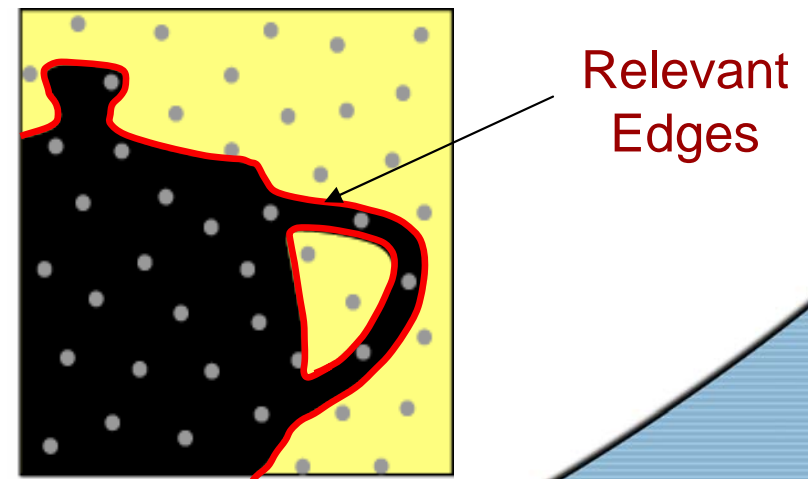
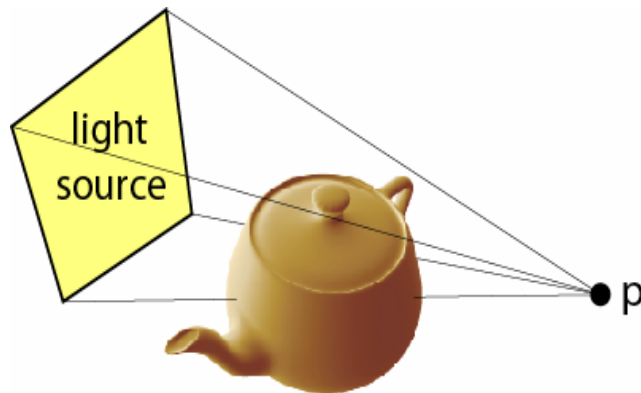
Methods for Soft Shadows

- Stochastic Ray Tracing
 - Use many shadow ray to sample light source
- Radiosity Algorithm
 - Visibility of light source defined on a patch level
- Tracing thick rays
 - Use pyramid beam, intersecting with shadow casters
- Soft Shadow volumes
 - Projection of shadow caster to light source
 - Our method



Soft Shadow Volumes

- Use discrete light samples to integrate
- Project occluders on to the light source
- To speed it up only project relevant edges \rightarrow penumbra wedges





*3 Conditions for **relevant** edges*

- We have to project an edge to the light source to calculate shadow for point p when:
 1. Edge is a silhouette edge from some point on the light source
 2. Edge overlaps light source viewed from point p
 3. Edge is a silhouette edge from p



The Algorithm

for every edge: (pre-compute)

Is **E** a silhouette for LS?

yes

Generate wedge

store wedge footprint to hemicube

for every **P**: (ray tracing)

Project point **P** to hemicube

get wedge list from hemicube

For every wedge(edge)

Is **P** inside the wedge ?

yes

Is **E** a silhouette for **P**?

yes

Project **E** to light source

Update Light Samples

Cast a ray from **P** to LS

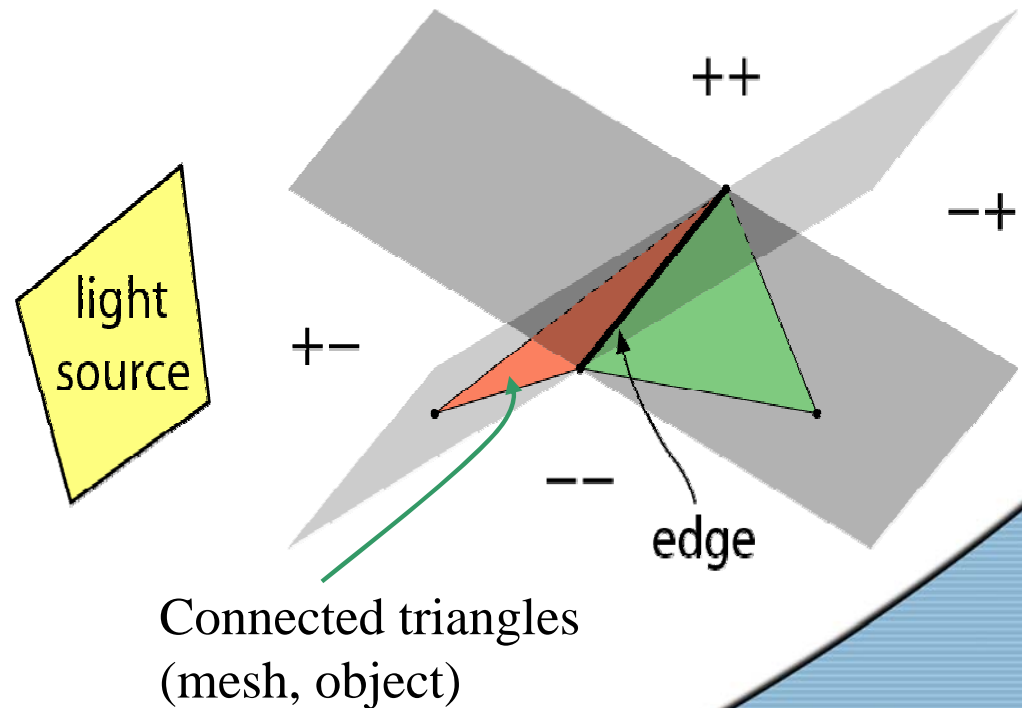
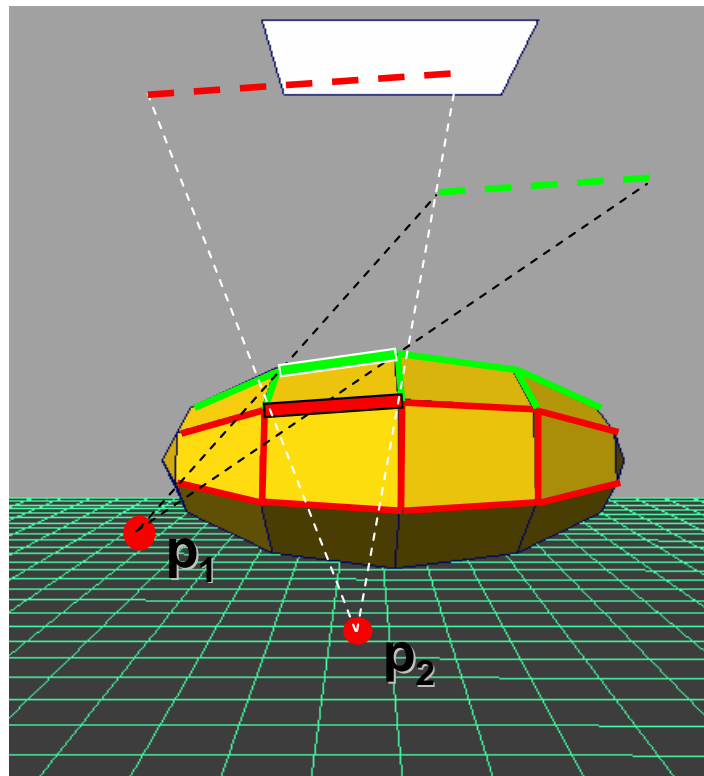
Ray not occluded ?

yes

Return number of visible light samples

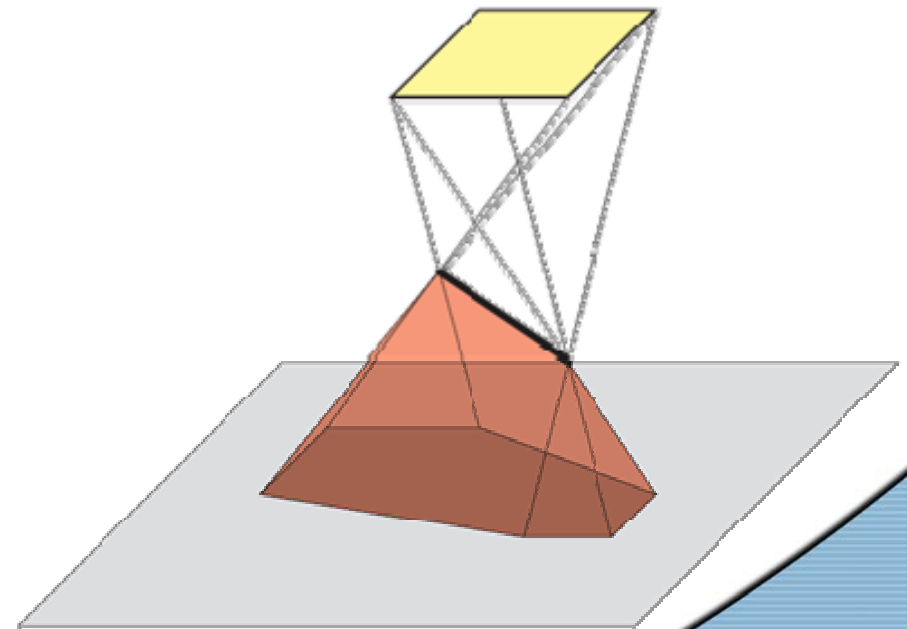
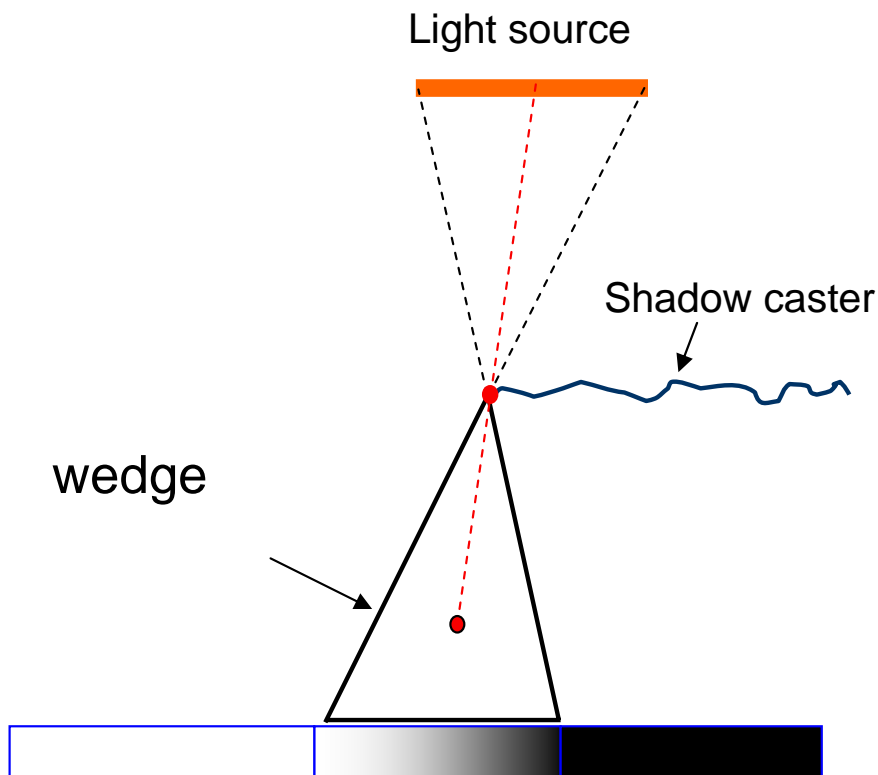
1. *E is silhouette for light source*

- Edge are Silhouette edges from the light only if light source does not lay entirely in subspace -- or ++



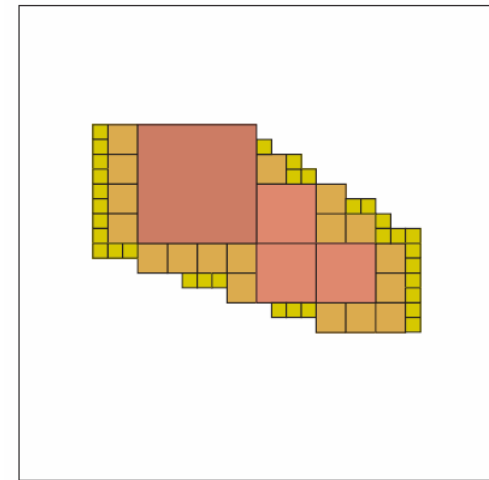
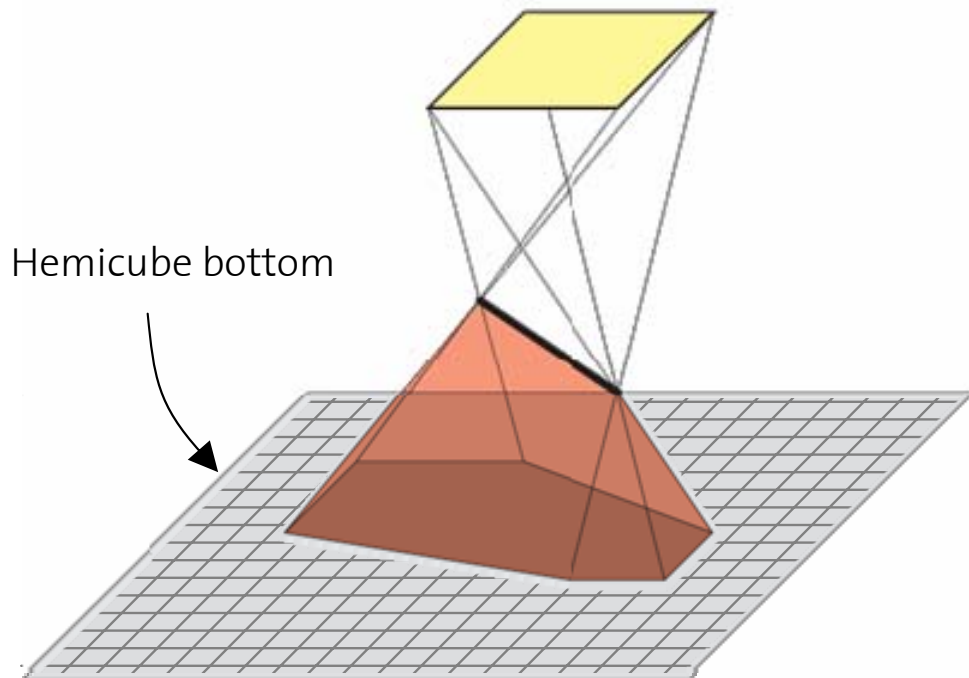
2. *Edge overlaps light source*

- Edge overlaps light source only for points inside shadow wedge



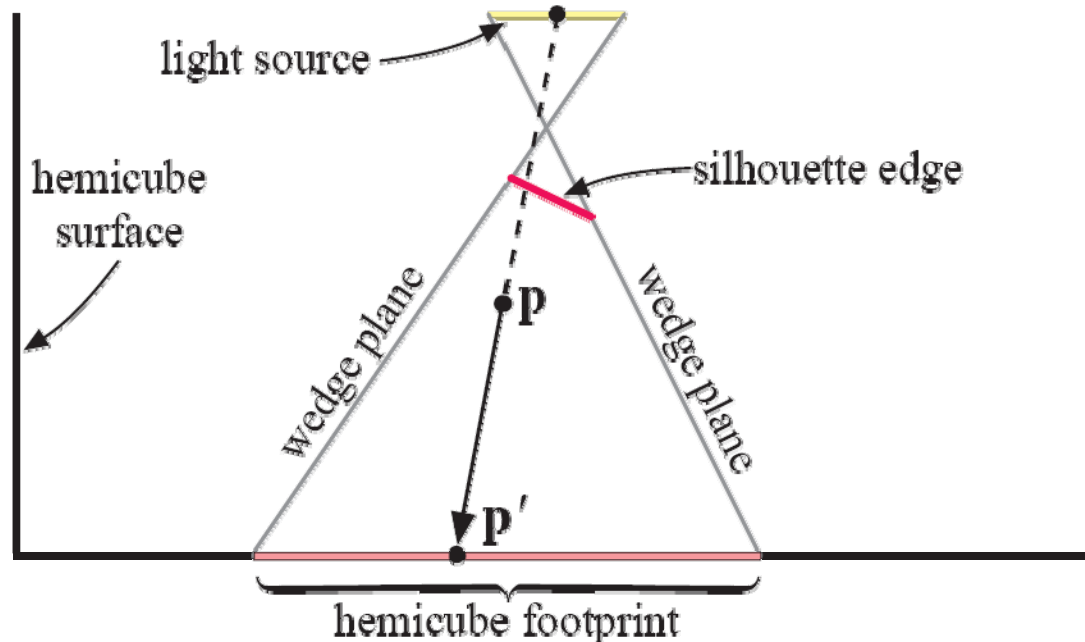
Acceleration Structure for Condition 2

- Pre compute foot prints of wedges (from edges passed Test1) before rendering of frame
- Store in a **hemicube grid** a list of wedges (conservative)



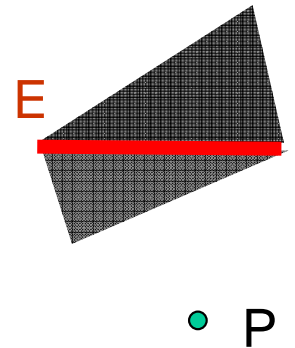
Test for Condition 2

- Find all possible wedges by projecting point p from mid of light source to the hemicube
- This list of wedges and corresponding edges is conservative
- Test if p is inside the wedge



Test for Condition 3

- Next we test if edges returned from the hemicube data structure are silhouette edges from point P
- This is true if one (of two) triangle connected to the edge is front facing when viewed from point p
- If there is only one triangle -> edge is always silhouette





The Algorithm

for every edge **E**: (precompute)

Is **E** a silhouette for LS?

yes

Generate wedge

store wedge footprint to hemicube

for every point **P**: (ray tracing)

Project point **P** to hemicube

get wedge list from hemicube

For every wedge(edge)

Is **P** inside wedge ?

yes

Is **E** a silhouette for **P**?

yes

Project **E** to light source

Update Light Samples

Cast a ray from **P** to LS

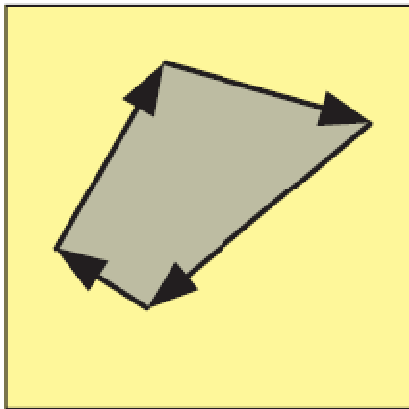
Ray not occluded ?

yes

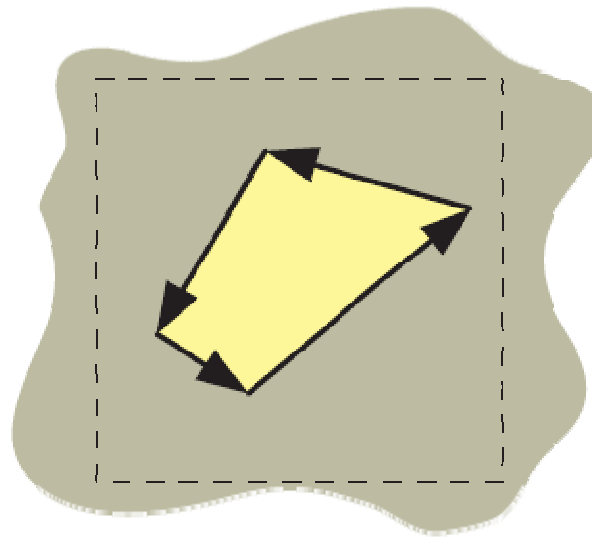
Return number of visible light samples

Integration: Projection

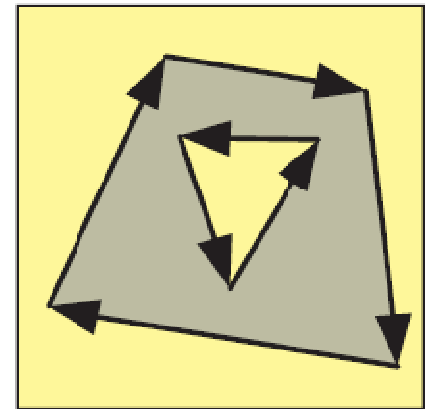
- We add an orientation to each projected edge so that the right side is the side where the occluder is located



(a)



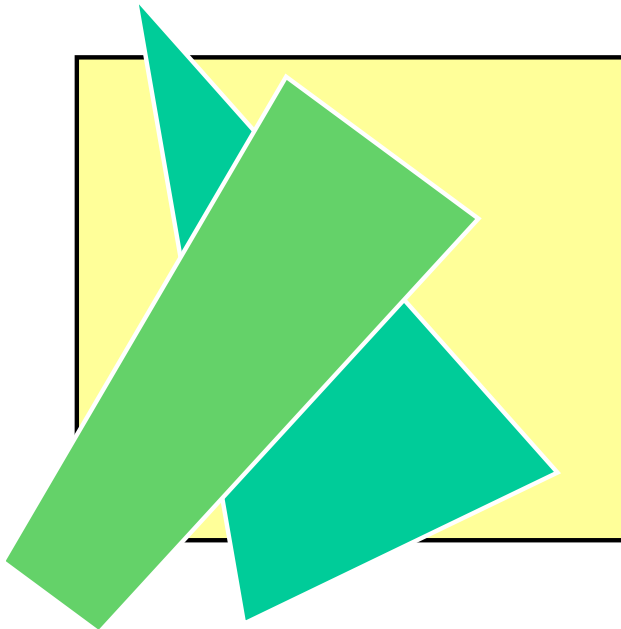
(b)



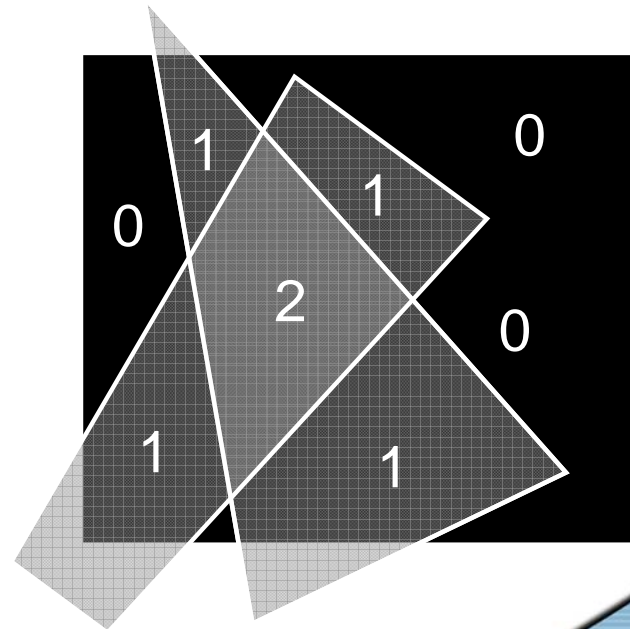
(c)

Integration: Depth Complexity Function

- Depth complexity Function returns the number of objects in front of the light source
- The projected edges are “changing events” of depth complexity function



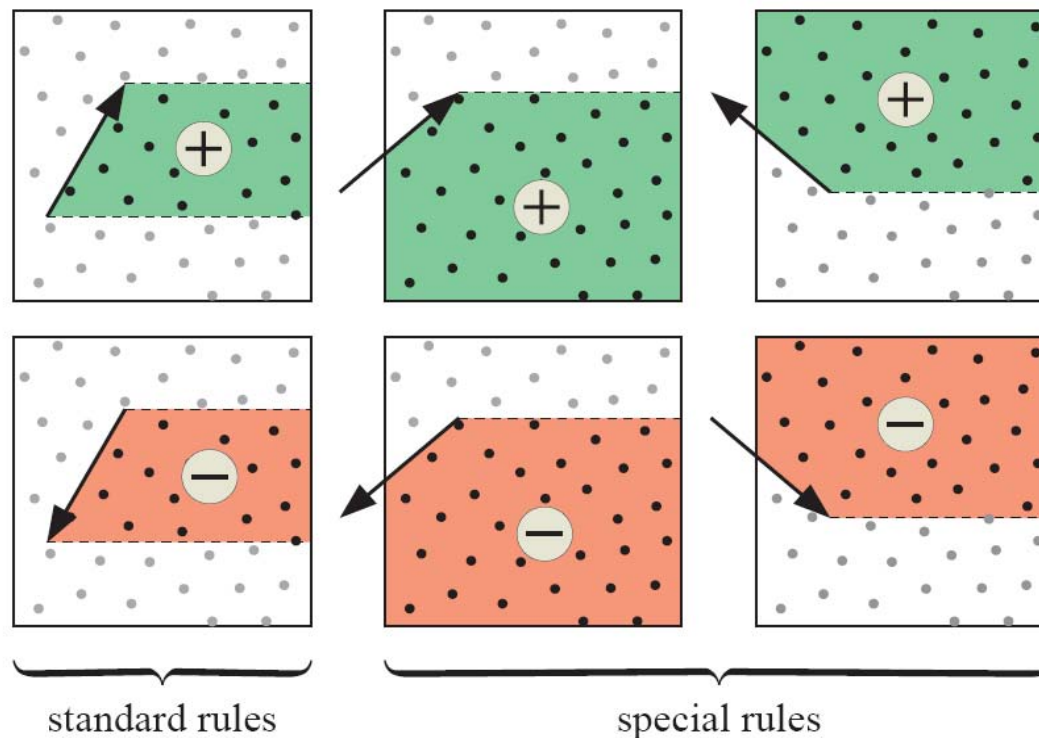
Light source as seen from p



Depth complexity function

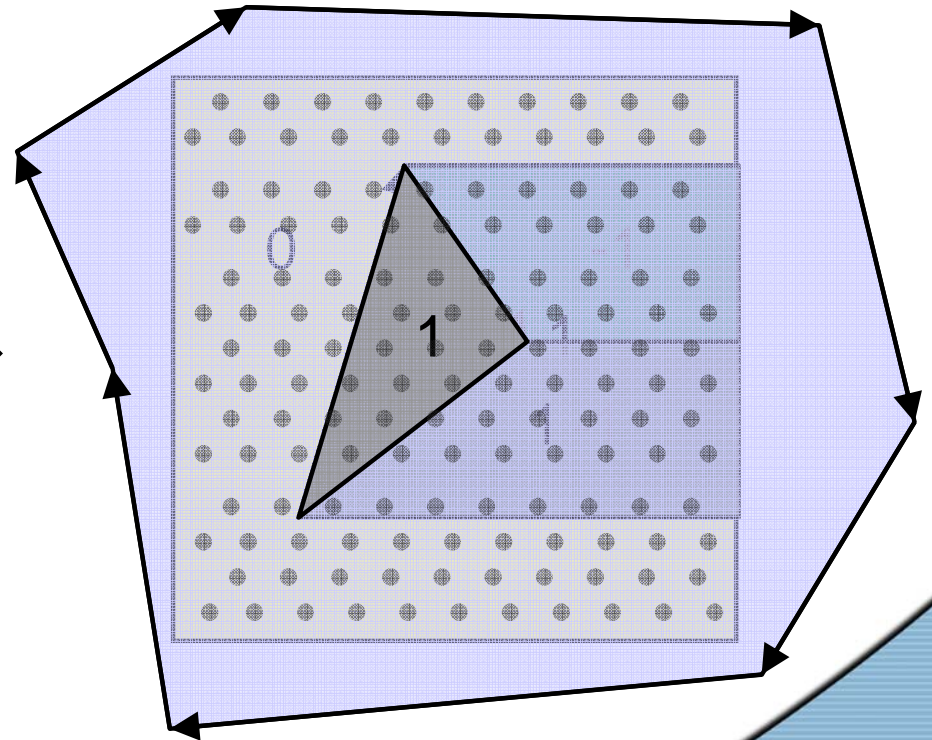
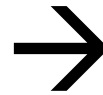
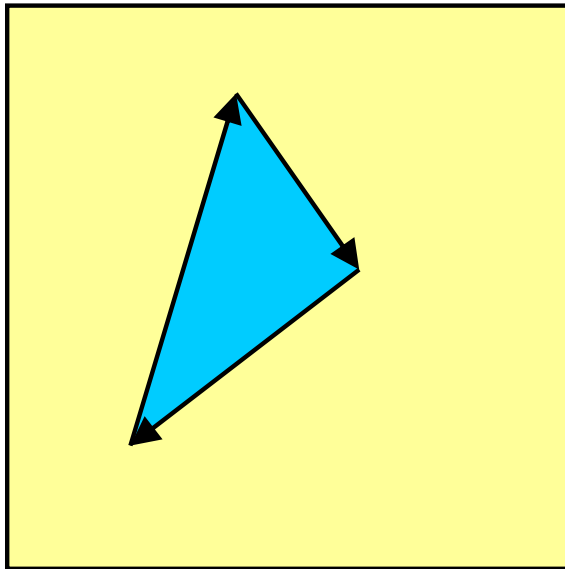
Integration Rules

- Build “relative” depth complexity function by using a counter at each light sample and the following rules:



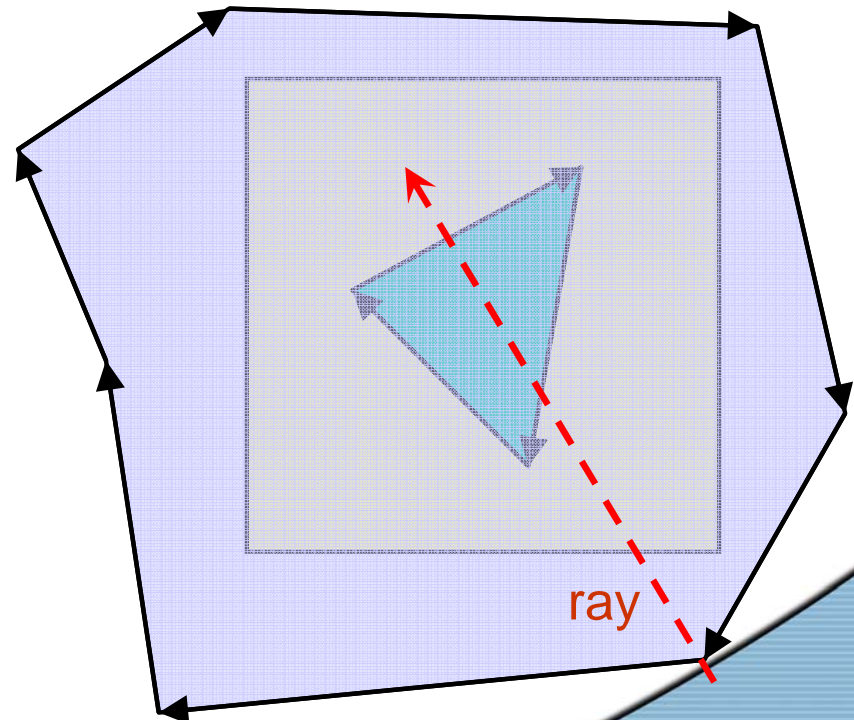
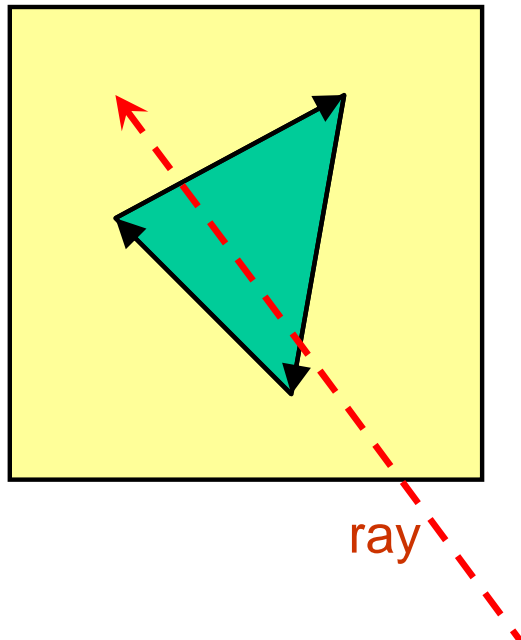
Integration Step by Step

- Each edge can be processed separately



We use only one shadow ray

- Cast a shadow ray to a point with smallest relative depth
- To check if light area is visible
- “finding integration constant”





The Algorithm

for every edge: (precompute)

Is **E** a silhouette for LS?

yes

Generate wedge

store wedge footprint to hemicube

for every **P**: (ray tracing)

Project point **P** to hemicube

get wedge list from hemicube

For every wedge(edge)

Is **P** inside wedge ?

yes

Is **E** a silhouette for **P**?

yes

Project **E** to light source

Update Light Samples

Cast a ray from **P** to LS

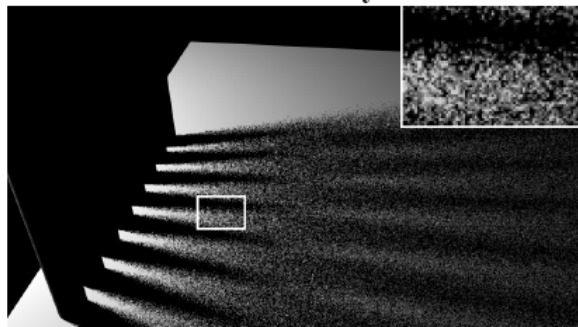
Ray not occluded ?

yes

Return number of visible light samples

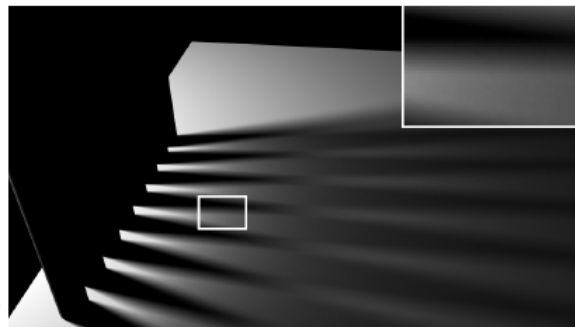
Results

Shadow Rays



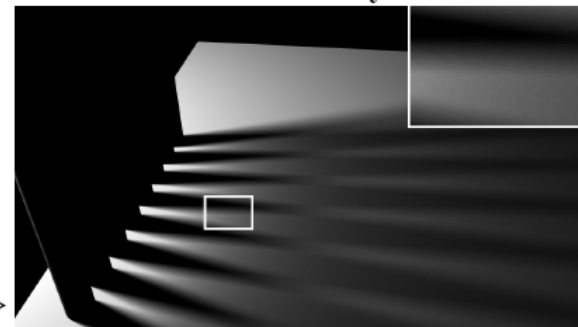
⇕ Comparable time

Soft Shadow Volumes



⇕ Equal quality

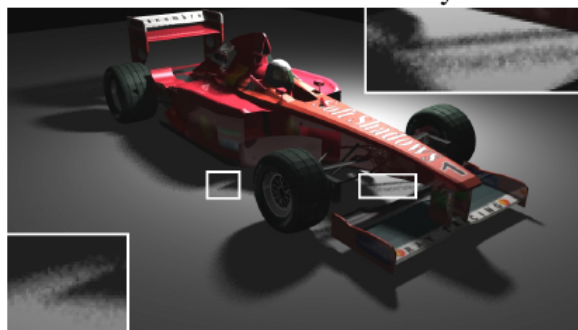
Shadow Rays



7 seconds / 2 shadow rays

4 seconds

7 min 34 seconds / 256 shadow rays



⇕ Comparable time



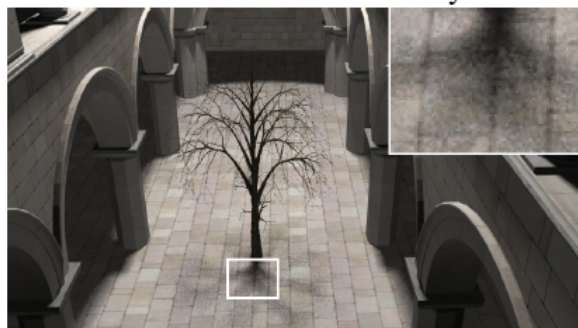
⇕ Equal quality



11 seconds / 8 shadow rays

10 seconds

4 min 55 seconds / 200 shadow rays



⇕ Comparable time



⇕ Equal quality



37 seconds / 12 shadow rays

34 seconds

6 min 2 seconds / 150 shadow rays



Limits

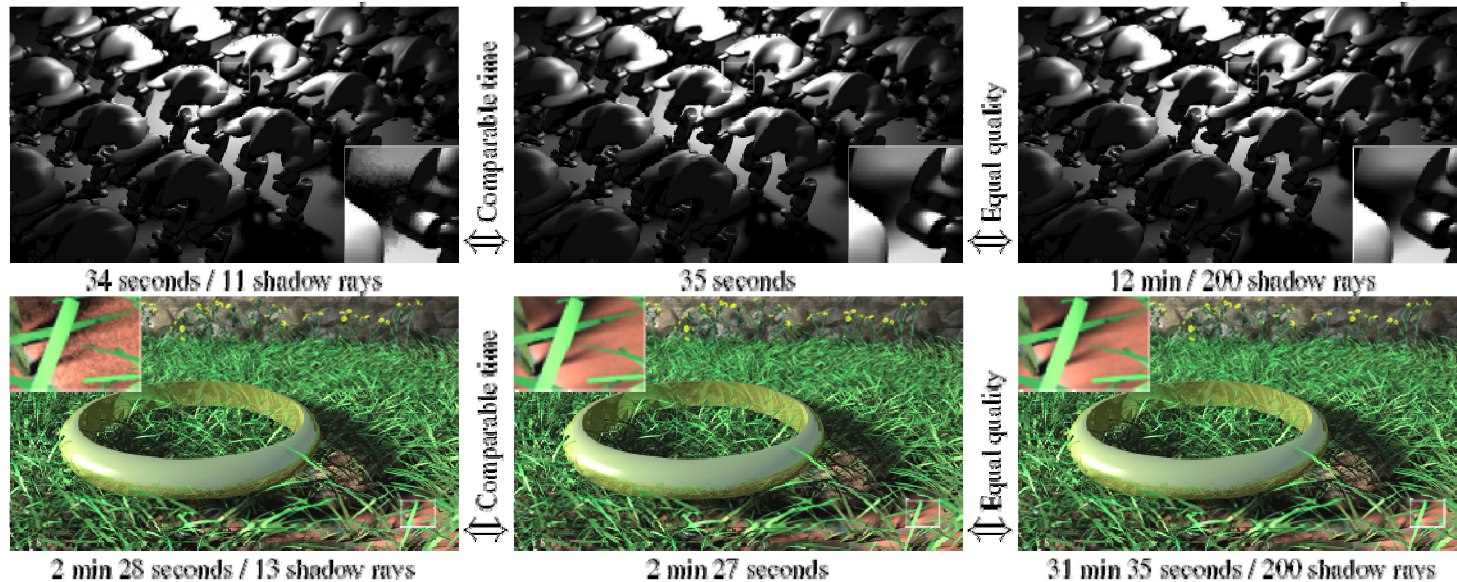
- Objects have to be triangle meshes
=> NURBS not directly supported
- Only planar light sources
- Inefficient for many unconnected triangles
- Speed depends on “light source size”



Future work

- Maybe use graphics hardware (GPU, RPU)
- Maybe possible to speed it further up for series of nearly identical frames (movies)
- BRDF of light source

Discussion



- Pros
 - easy to understand and implement
 - Significant speedup
- Cons
 - Still slow (not real-time, games ☺)
 - Only shown for rectangular light sources
 - Fuzzy explanations how to build wedge footprints