UCL

Shadow Algorithms

UCL

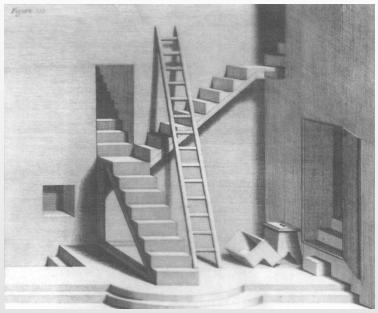
Outline

- Introduction
- Sharp shadows
- Soft shadows
- Conclusion

UCL

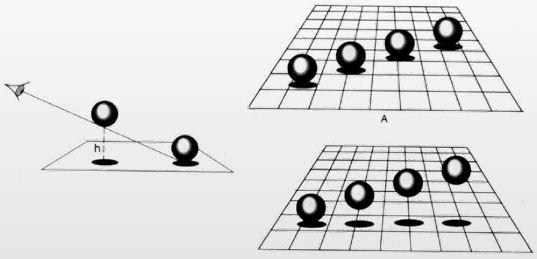
Why are Shadows Important?

- Depth cue
- Scene Lighting
- Realism
- Contact points



UCL

Shadows as a Depth Cue / Spatial Relation



UCL

For Intuition about Scene Lighting

- Position of the light (e.g. sundial)
- Hard shadows vs. soft shadows
- Colored lights
- Directional light vs. point light

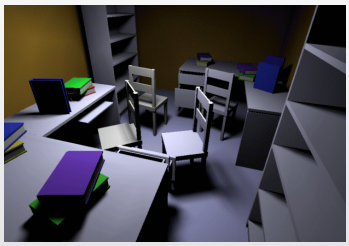
UCL

Shadows are Complex

- In the real world sources of light are not points
- The intensity within a shadow is not constant
  - umbra, the part that sees nothing of the source
  - penumbra, part that receives some light
- In computer graphics we simplify and cheat

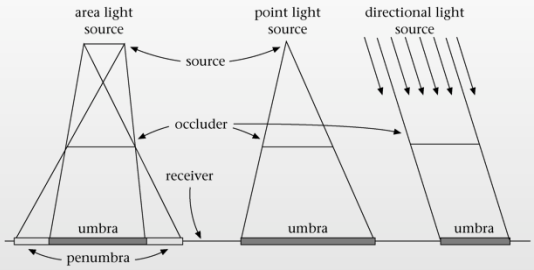
UCL

Definition



UCL

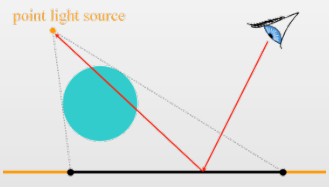
Definition

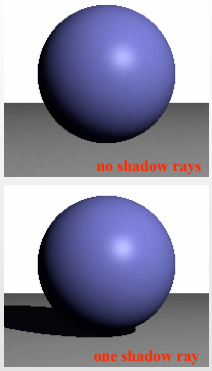


UCL

Shadows

- One shadow ray per intersection per point light source

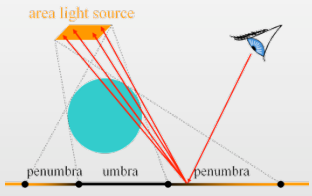


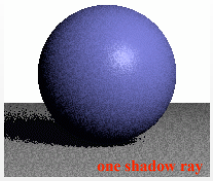


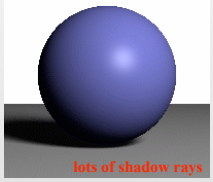
UCL

Soft Shadows

- Multiple shadow rays to sample area light source



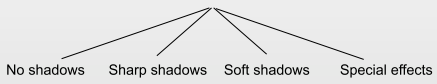




UCL

Current Shadowing Methods

- There exist a very large number of methods
- We are interested in methods suitable for interactive walkthroughs, speed is crucial
- We will classify them by complexity:



UCL

Sharp Shadows

UCL

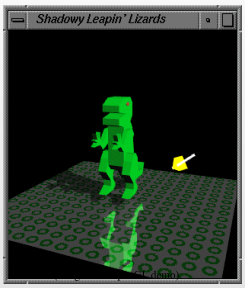
Sharp Shadows

Source is assumed to be a point or direction

Fake	Hardware Assisted	Pre-computed	Ray tracing
Fast but often inadequate • Projection	Use of specialized hardware to generate shadows • Shadow Maps • Shadow Volumes	Shadows are pre-computed and stored for repeated use • Illumination Maps	It's slow for large scenes

UCL

Fake Shadows: Projection on Ground



- Objects are compressed using a matrix transformation and pasted to the ground
- No inter-object shadows
- Very fast

UCL

Shadow Maps

- Compute a Z-buffer from the source
  - use the light source as a view point and render the objects to get the depth information (shadow Z-buffer)
- Run a normal Z-buffer with shadow calculation
  - from the view point,
    - each pixel P (x_v,y_v,z_v) in this buffer is mapped to the shadow buffer (x_s,y_s,z_s) ,
    - if the z_s value is less or equal to that stored there then the point is lit,
    - otherwise is in shadow

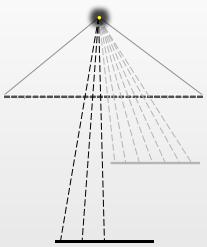
UCL

Shadow Maps

[Williams '78]

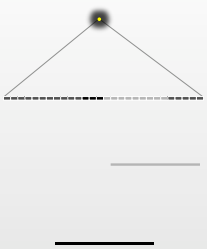
UCL

Shadow Maps



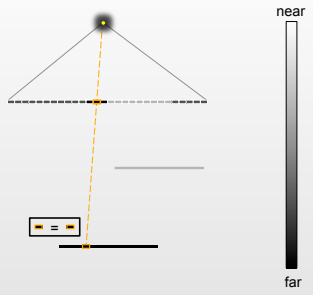
UCL

Shadow Maps



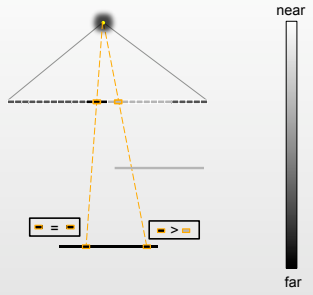
UCL

Shadow Maps



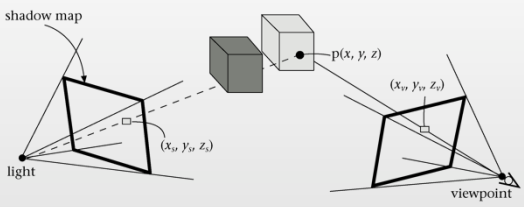
UCL

Shadow Maps



UCL

Shadow Maps



UCL

Shadow Maps with OpenGL

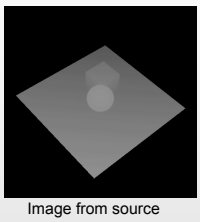
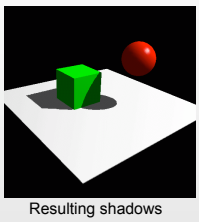


Image from source



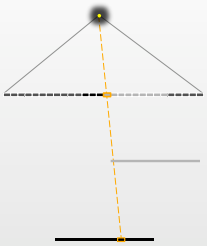
Resulting shadows

- This technique can be accelerated by using texture mapping hardware

UCL

Shadow Map Filtering

unfiltered

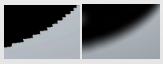


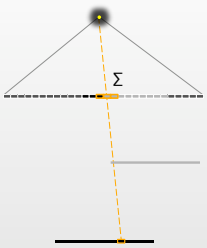
UCL

Shadow Map Filtering

unfiltered

filtered

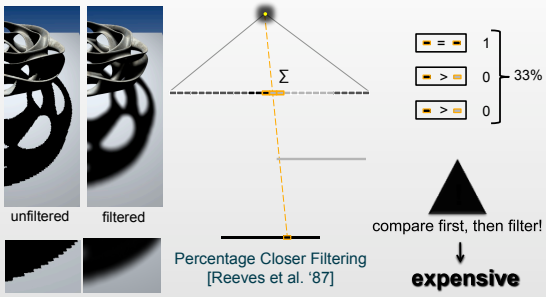




Percentage Closer Filtering
[Reeves et al. '87]

UCL

Shadow Map Filtering



Percentage Closer Filtering
[Reeves et al. '87]

UCL

Shadow Maps

- "Less than or equal" test is imprecise
  - Gives rise to "shadow acne"
- Often found in hardware now
  - Otherwise high cost operation
- Imprecise since it is only accurate in the image space of the light
  - Imagine a shadow throw over complex objects or long distances
- Quality depends on resolution (jagged edges)
  - Percentage-closer filtering helps
- FOV of shadow map?

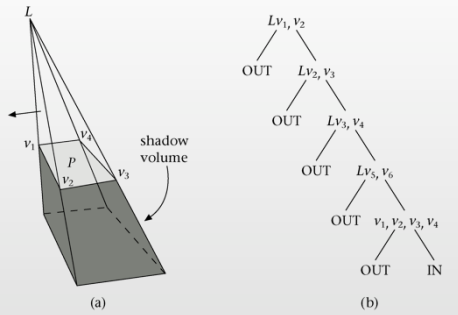
UCL

Shadow Volume Method

- Shadow volume (SV) is the volume of space below a polygon that cannot see the source (a culled pyramid)
- During rendering of image, the line from a point visible through a pixel to the eye is intersected with all object SVs
- The number of intersections indicates if the point is in shadow or not

UCL

Shadow Volumes



UCL

Shadow Volumes

- Just like a polygon - you are inside a volume if you need to cross a surface to exit it
- General idea of shadow volumes is count the number of shadow planes you cross
  - +1 for front facing
  - -1 for back facing
- If total is >0 you are in shadow

- Special case if the eye itself is in shadow

UCL

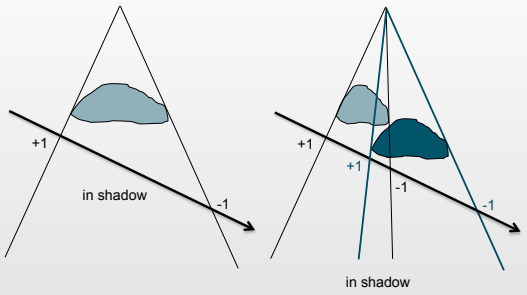
Shadow Volumes

Two stages:

1) Preprocessing
   - Find all planes of the shadow volume and their plane equations
2) At run-time
   - Determine shadow plane count per pixel
   - Use a scan-line method OR stencil test

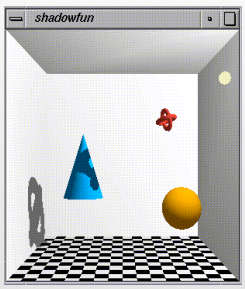
UCL

Shadow Volume Example



UCL

Shadow Volumes with openGL



(image from an SGI demo)

- Shadow volumes are rendered at each frame
- The stencil buffer is used for counting how many SV are crossed
- Sometimes not all objects are used for casting shadows

UCL

Shadow Volumes with Stencil Test

- A stencil buffer is screen sized buffer (1-8bit) that stores a flag about a rendering operation
  - E.G. stencil[x,y] is negated if zbuffer[x,y] is less than current z value (i.e. stencil is set if and only if z buffer test passes)
- Many uses in graphics

UCL

Shadow Volumes with Stencil Test

- Render the scene into the RGB and z-buffer
- Turn z-buffer *writing* off, then render all shadow polygons with the stencil buffer
  - Increment stencil count for front-facing
  - Decrement for back facing
- Re-render scene with lighting OFF and only render pixels where stencil is non-zero

UCL

Summary for sharp shadows

- Four shadow umbra techniques
- Image space
  - Shadow maps
  - Shadow volumes
- Object space
  - Fake shadows

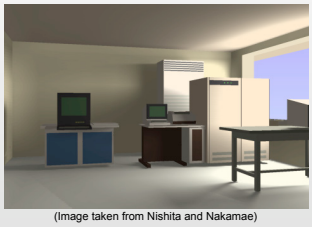
UCL

Soft Shadows

UCL

Soft Shadows

- Source has a finite extend
- Images look a lot more realistic



(Image taken from Nishita and Nakamae)

UCL

Soft Shadows

Hardware Assisted

Mainly treat the light source as a collection of points
- Accumulation buffer
- Shadow volumes
- Shadow textures

Pre-computed

Mainly analytical computation on the geometry of the source
- Discontinuity Meshing
- Illumination Maps

Radiosity

This is also pre-computed
- Hemi-cube
- Ray casing

Ray-based

- Distributed ray tracing
- Cone Tracing

UCL

Analytical v. Sampling

- Analytical
  - Find all boundaries within the penumbra. Done almost exclusively for polygonal light sources
- Sampling
  - Approximate solution that treat the light source as a set of points. Any shape source is possible.

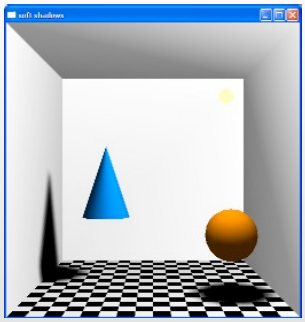
UCL

Soft Shadows using Point Light Source

- Place many point lights on an area light
  - Random positions work just fine
- Render hard shadows from each point light
  - E.g., using shadow volumes or shadow maps
- Sum up all contributions
  - Can be done on the GPU (in the frame-buffer)
- Similar to what ray-tracing does to get soft shadows

UCL

Example



UCL

Illumination Maps (Shadow Textures)

- Shadows are pre-computed and stored as textures on the receiving polygons
- Displayed using graphics hardware in real-time
- Disadvantage: lighting cannot change

UCL

Analytical methods

- Find all boundaries within the penumbra. Done almost exclusively for polygonal light sources.

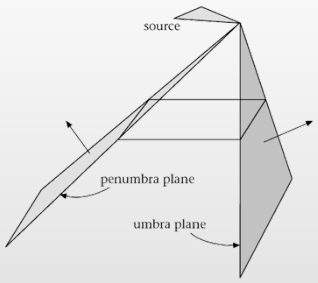
UCL

Extremal Shadow Boundaries

- What is the potential area of the penumbra and umbra?
- For penumbra:
  - Bounded by planes define by a pair of source vertex and occluder edge where the source is in the front space and the occluder on the back
- For umbra:
  - Similarly defined planes, but where source and occluder are in the back space

UCL

Extremal Shadow Boundaries



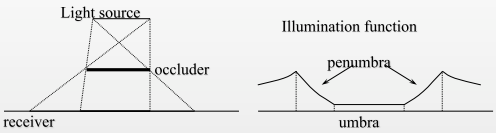
UCL

Shading Using Extremal Planes

- If you write these planes into object space
- We can use a scan-conversion as we have before
  - At each pixel we must estimate the proportion of the light source that can be seen
  - [Usually done with SVBSP tree(s)]

UCL

Discontinuity Meshing



- Subdivide at discontinuity points
- Compute illumination intensity at discontinuity points
- Quadratic approximation on segments between discontinuity points

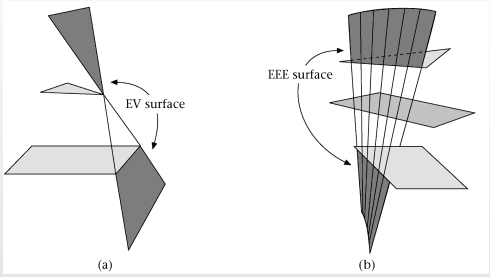
UCL

Discontinuity Meshing

- Borrowing aspect graphs from computer vision
- Define critical surfaces where visual events occur
  - EV surfaces: planes defined by edge and vertex
  - EEE surfaces: quadratic surfaces defined by three non-adjacent edges.
- Penumbra volumes so far have used EV only

UCL

EV and EEE Surfaces



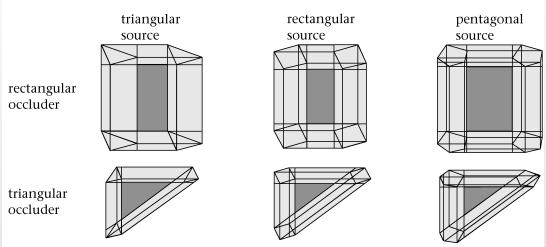
UCL

Discontinuity Meshing

- Discontinuities of the illumination of a polygon occur at the places where EV and EEE surfaces intersect the polygon
- Discontinuities occur at different degrees
- Discontinuities are written into the geometry of the scene as before

UCL

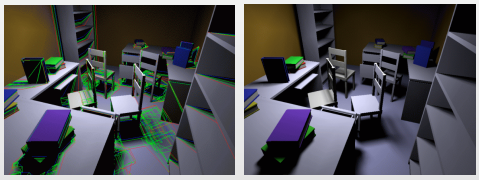
Examples of meshes



UCL

Discontinuity Meshing

- Very high quality shadows
- Slow and prone to floating point errors



UCL

Radiosity

- Scene polygons are subdivided into a mesh, or the illumination is stored as a texture
- Very realistic results
- Good for static scenes but not for moving objects

UCL

Conclusion

- A very large number of shadow algorithms exist
- Many of them are unsuitable for walkthroughs of very complex scenes:
  - with pre-computation methods scene cannot be modified
  - or are to slow (ray-tracing, soft shadows)
- Hard shadows
  - on-the-fly methods (SM and SV) are fast enough