Shadows

Acknowledgement:

Images and many slides from presentations by Mark J. Kilgard and other Nvidia folks, from slides on developer.nvidia.com

Practical & Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering, Cass Everitt & Mark J. Kilgard, GDC 2002
Shadow Mapping with Today's OpenGL Hardware, Mark Kilgard, CEDEC 2001, Tokyo, Japan
Robust Stencil Shadow Volumes, Mark Kilgard, CEDEC 2001, Tokyo, Japan
Shadow Mapping, Cass Everitt
Reflections, Shadows,Transparency, and Fog, Mark Kilgard, GDC 2000 Tutorial
Shadow Mapping with Todays OpenGL Hardware, Mark Kilgard, GDC 2000 Tutorial

Shadows

- Important visual cue
  - All real world scenes have shadows
  - Occlusion from light's point-of-view instead of viewer's
    - Cue for light-object-object relationships

Local Illumination & Shadows

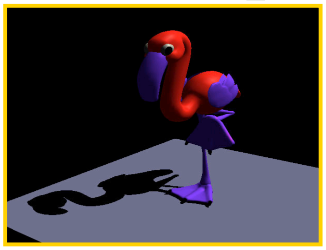
- Typically lack shadow support
  - Ignore occlusion of lights that creates shadows
- Global lighting models account for shadows
  - Too expensive for interactive use

- Interactive shadow algorithms typically operate independent of local lighting models

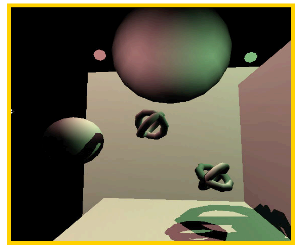
Interactive Shadow Simplifications

- True shadows are expensive, cut corners
  - Simple geometry tricks
    - Projection
    - Volume intersection
  - Pre-compute static shadow results where possible
  - Make simplifying assumptions
    - Treat area light sources as points
  - Exploit hardware features such as stencil

Common Real-time Shadow Techniques

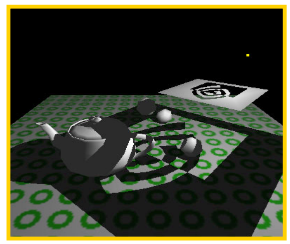


Projected planar shadows



Shadow volumes

Light maps



Hybrid approaches

Problems with Common Techniques

- Mostly tricks with lots of limitations
  - Light maps
    - totally unsuited for dynamic shadows
  - Projected planar shadows
    - well works only on flat surfaces
  - Stenciled shadow volumes
    - determining the shadow volume is hard work
  - Shadow Maps
    - only works with spotlight frustums
    - *shadow acne* artifacts

Pre-computed Shadow Textures (Light maps, 8.5 in book)

- Lightmaps are static shadow textures
  - Compute lightmaps off-line with radiosity solver
    - local lighting models evaluated on tight grid can work too
  - Lightmaps with soft shadows can be built faster with convolution approach, accelerated using the Fast Fourier Transform [Soler & Silion 98]
  - Pre-computed shadow textures work well for building interior with lots of diffuse surfaces

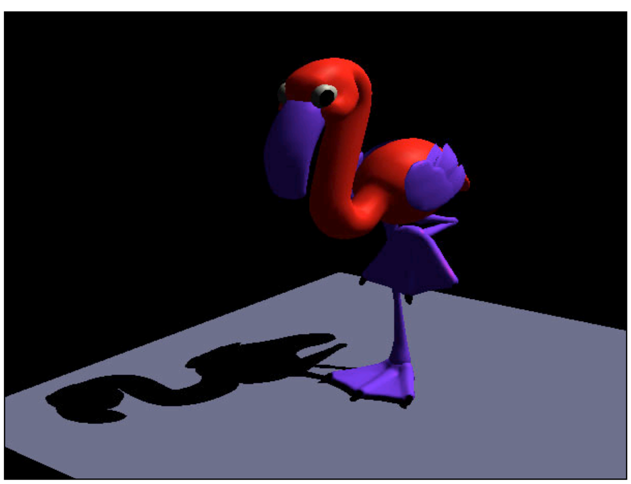
Aside: Stencil Buffers

- An extra test for fine-grain pixel control
  - Standard OpenGL and DirectX 6 feature
  - Per-pixel test similar to depth buffering
    - Tests fragment against pixel's stencil value, rejects fragments that fail
    - Also, can modify pixel's stencil buffer value based on stencil/depth test results
  - Hardware accelerates stencil testing
    - Typically *free* when depth testing too

Stencil Testing

- Similar to Depth Testing but
  - Compares current reference value to pixel's stencil buffer value
  - Same comparison functions as depth test
- Stencil values controlled by stencil ops
  - "stencil" side effects of stencil & depth tests
  - Possible operations
    - Increment, Decrement
    - Keep, Replace, Zero, Invert

Projected Planar Shadows



Projected Planar Shadows

- Classic computer graphics trick
  - Given
    - Ground plane equation, Ax + By + Cz + D = 0
    - Light position (x, y, z, w)
  - Construct projective transform that "squishes" vertices into ground plane based on light position
    - Concatenate this with view transform
  - Rendering 3D object creates shadow-like pile of polygons in ground plane

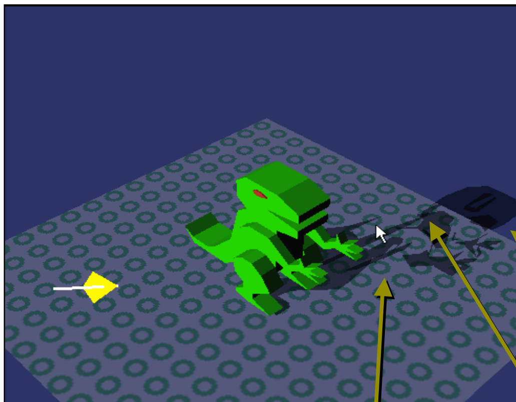
Projected Planar Shadow Issues

- Shadow must be cast on infinite planes
- "Pile of polygons" creates Z-fighting artifacts
  - Polygon offset fixes this, but disturbs Z values
- Difficult to blend shadow with ground texture
  - Just blending creates "double blends"
  - Specular highlights can show in shadow

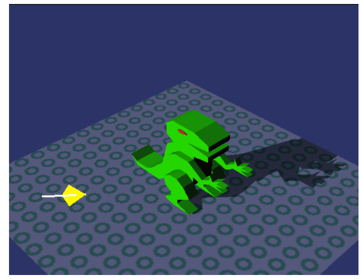
- Stencil testing can fix most of these issues

Projected Planar Shadow Artifacts

Bad



Good

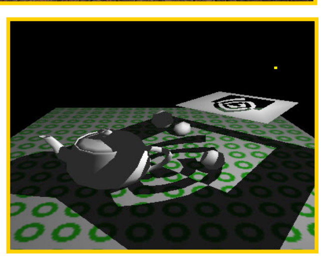


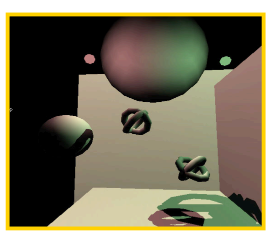
extends off ground region

Z fighting

double blending

Shadow Volumes



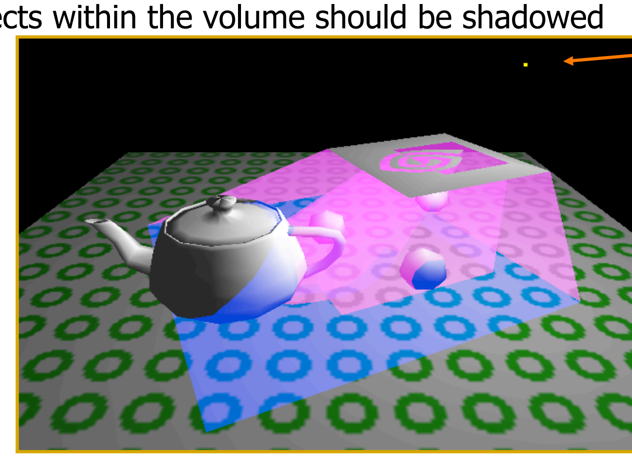


The Shadow Volume Concept

- Volumetric shadows, not just planar
  - A single point light source splits the world in two
    - shadowed regions
    - unshadowed regions
  - A shadow volume is the boundary between these shadowed and unshadowed regions
  - First described by [Crow 77]

Visualizing the Shadow Volume

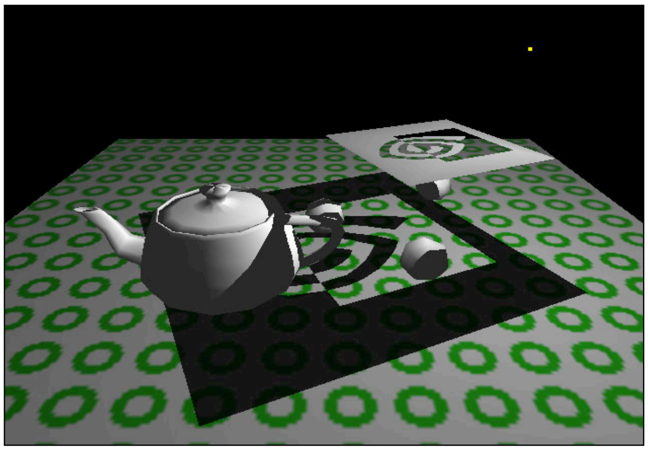
- Occluders and light source cast out a shadow volume
  - Objects within the volume should be shadowed



Light source

Shadow Volume Result

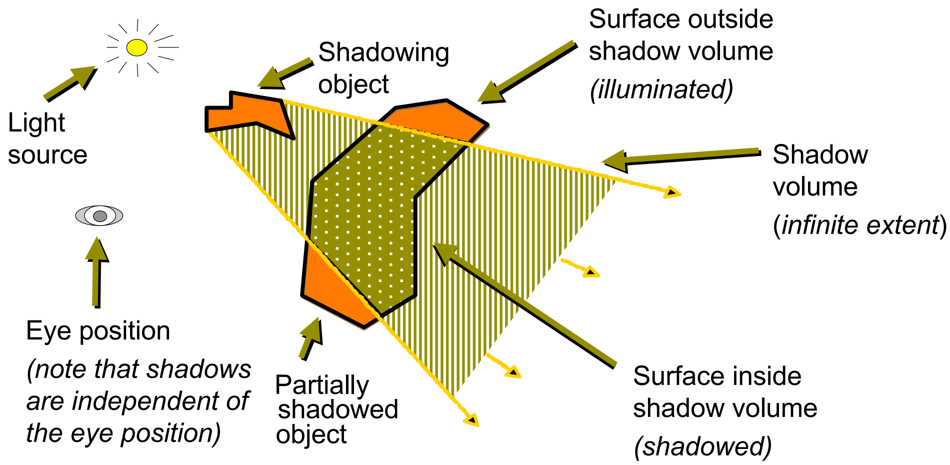
- Objects within the volume are shadowed



Shadow Volume Algorithm

- High-level view of the algorithm
  - Given the scene and a light source position, determine the shadow volume
  - Render the scene in two passes
    - Draw scene with the light *enabled*, updating *only* fragments in *unshadowed* region
    - Draw scene with the light *disabled*, updated *only* fragments in *shadowed* region
  - But how to control update of regions?

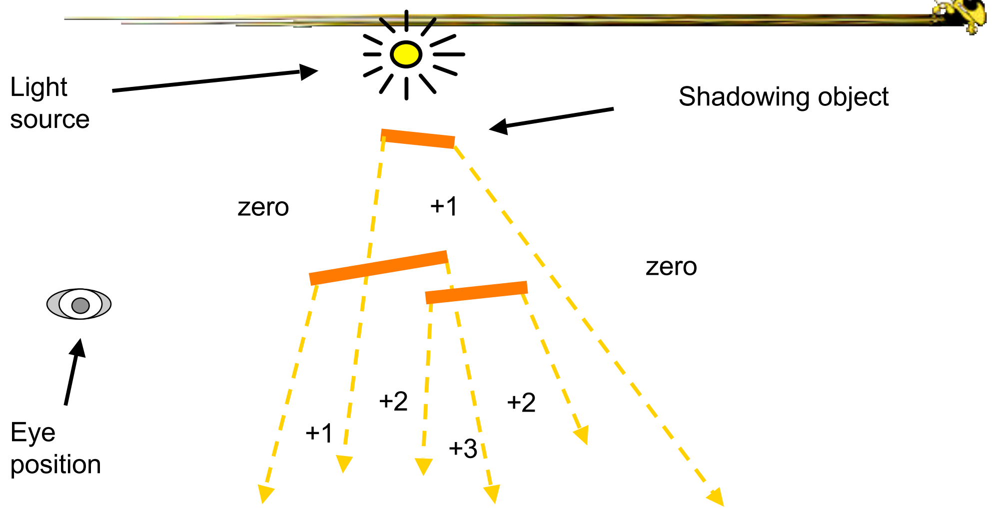
2D Cutaway of a Shadow Volume



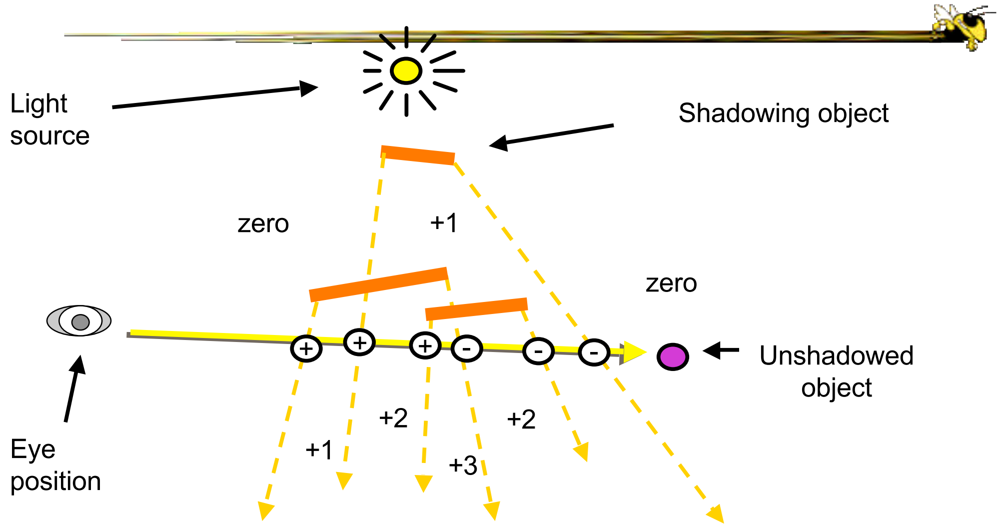
Counting Shadow Volume Enter/Leaves With a Stencil Buffer

- Render scene to initialize depth buffer
  - Depth values indicate the closest visible fragments
- Use a stencil enter/leave counting approach
  - Draw shadow volume twice using face culling
    - 1st pass: render front faces and increment when depth test passes
    - 2nd pass: render back faces and decrement when depth test passes
  - Don't update depth or color
- Afterward, pixel's stencil is non-zero if pixel in shadow, and zero if illuminated

Why Eye-to-Object Stencil Enter/Leave Counting Approach Works

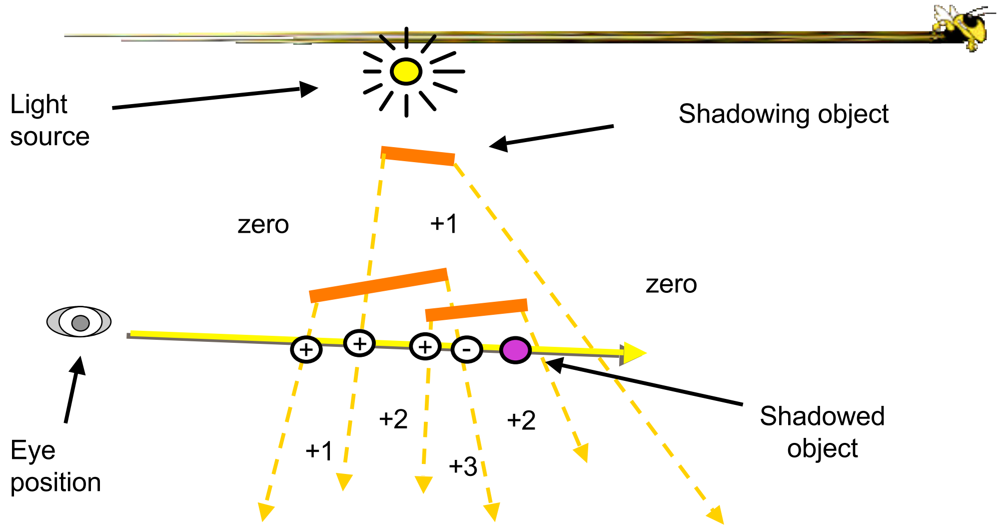


Illuminated, Behind Shadow Volumes



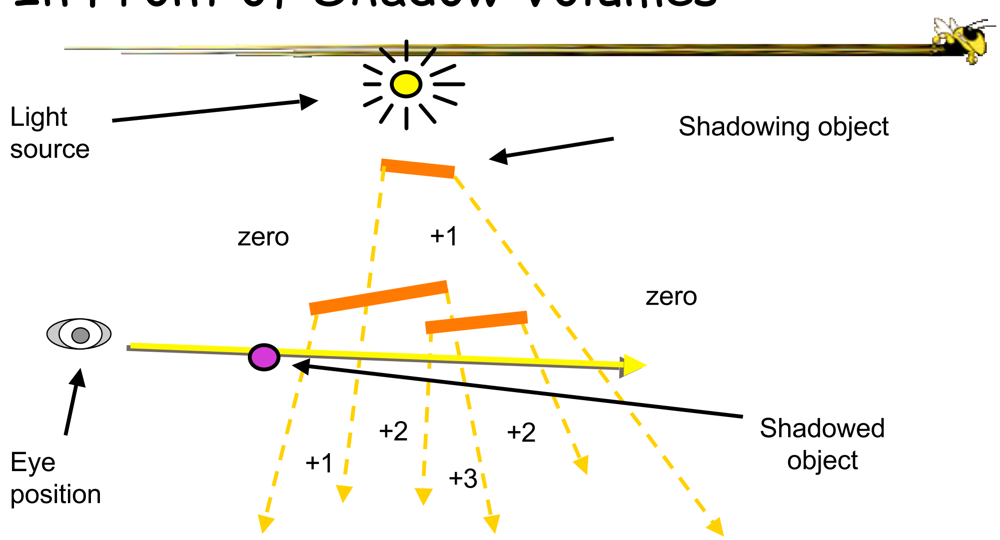
Shadow Volume Count = +1+1+1-1-1-1 = 0

Shadowed,
Nested in Shadow Volumes



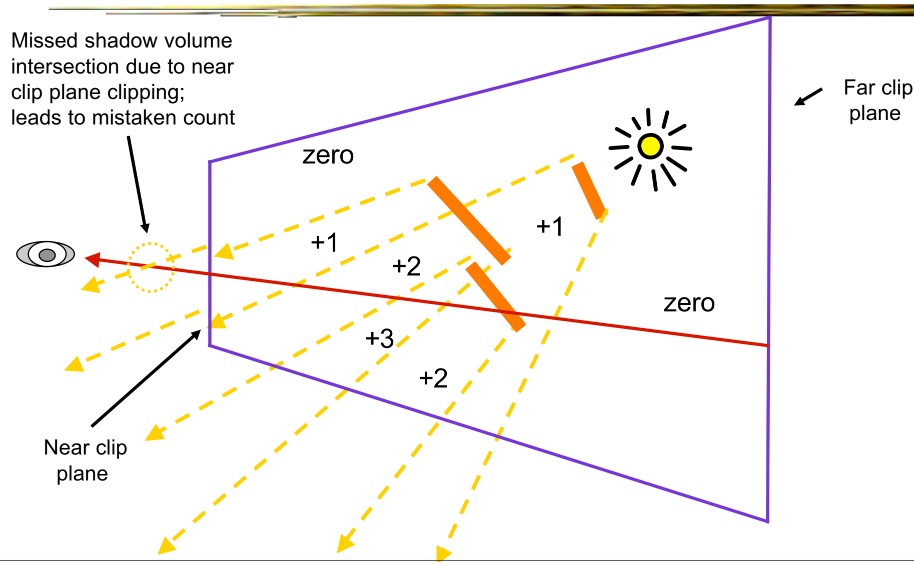
Shadow Volume Count = +1+1+1-1 = 2

Illuminated,
In Front of Shadow Volumes



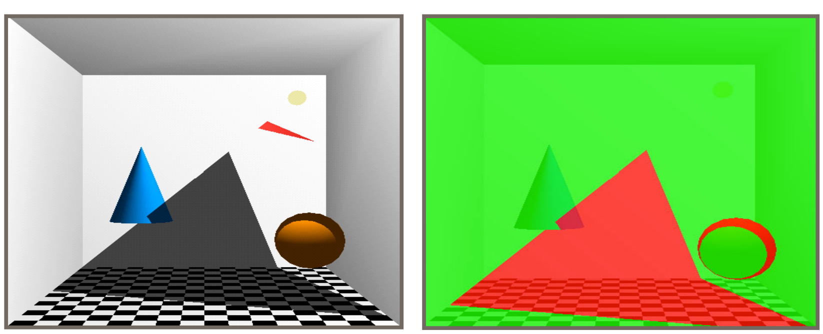
Shadow Volume Count = 0

Problems Created by Near Plane Clipping



Visualizing the Stencil Buffer Counts

Shadowed scene **Stencil buffer contents**



Stencil counts beyond 1 are possible for multiple or complex occluders.

red = stencil value of 1
green = stencil value of 0

GLUT *shadowvol* example credit: Tom McReynolds

Computing Shadow Volumes

- Harder than you might think
  - Easy for a single triangle, just project out three infinite polygons from the triangle, opposite the light position
  - For complex objects, projecting object's 2D silhouette is a good approximation (flat objects are easy)
  - Static shadow volumes can be pre-compiled

Stenciled Shadow Volumes in Practice (1)

Scene with shadows. Yellow light is embedded in the green three-holed object. P_{inf} is used for all the following scenes.

Same scene visualizing the shadow volumes.

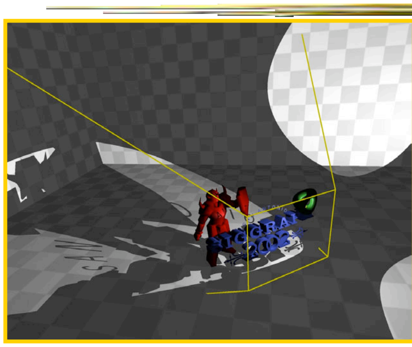
Stenciled Shadow Volumes in Practice (2)

Details worth noting . . .

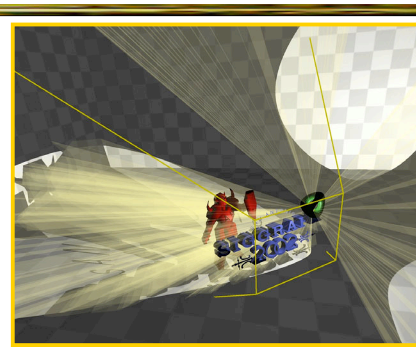
Fine details: Shadows of the A, N, and T letters on the knight's armor and shield.

Hard case: The shadow volume from the front-facing hole would definitely intersect the near clip plane.

Stenciled Shadow Volumes in Practice (3)



Alternate view of same scene with shadows. Yellow lines indicate previous view's view frustum boundary. Recall shadows are view-independent.



Shadow volumes from the alternate view.

Shadow Volume Issues

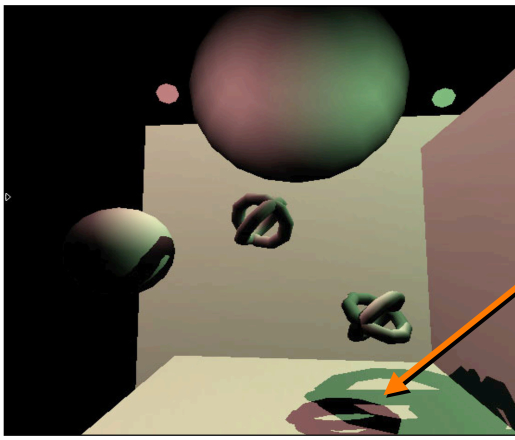
- Practical considerations [Bergeron 86]
  - If eye is in shadow volume, need to determine this and flip enabled & disabled lighting passes
  - Shadow volume only as good as its tessellation
    - Shadows tend to magnify limited tessellation of curved surfaces
  - Must cap the shadow volume's intersection with the near clipping plane
  - Open models and non-planar polygons

Reconstructing Shadow Volume From a Depth Buffer

- Very clever idea [McCool 98]
  - Render scene from light source with depth testing
  - Read back the depth buffer
  - Use computer vision techniques to reconstruct the shadow volume geometry from the depth buffer image
  - Very reasonable results for complex scenes

Multiple Lights and Shadow Volumes

- Requires still more rendering passes, but can work!



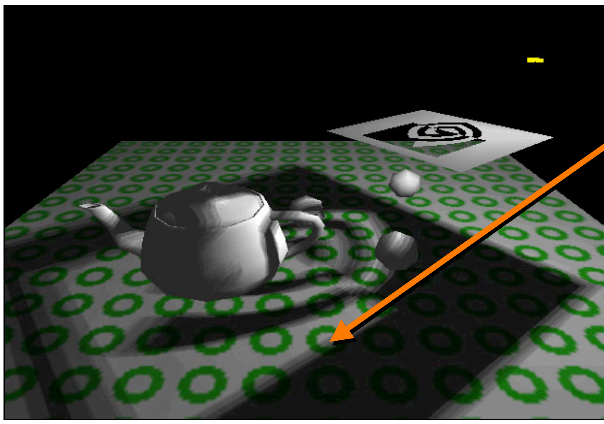
Shadows from different light sources overlap correctly

Shadow Volumes from Area Light Sources

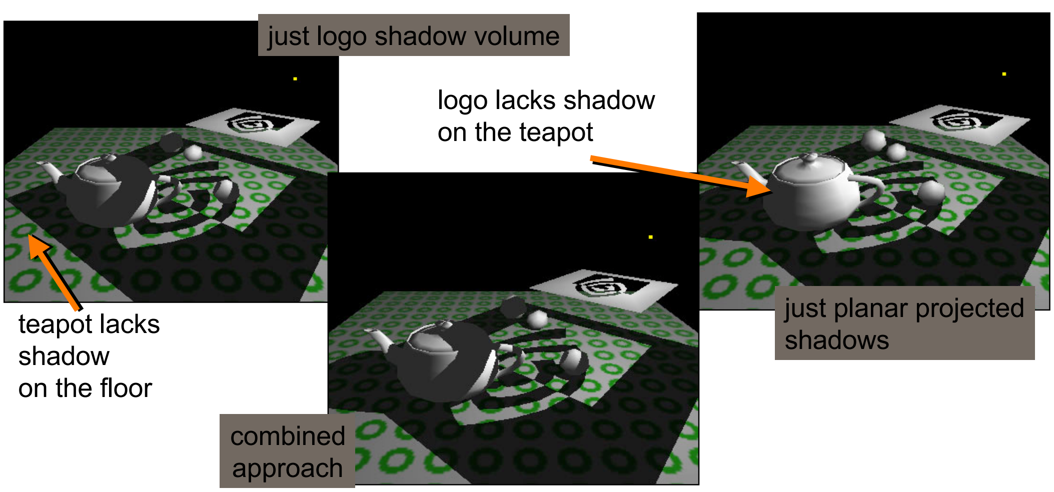
- Make soft shadows
  - Shadow volumes work for point light sources
  - Area light sources are common and make soft shadows
  - Model an area light source as a collection of point light sources [Brotman & Badler 84]
  - Use accumulation buffer or additive blending to accumulate soft shadow
  - Linear cost per shadow volume sample

Soft Shadow Example

- Eight samples (more would be better)



Combined Shadow Algorithm Example: Shadow Volume + Planar Projection



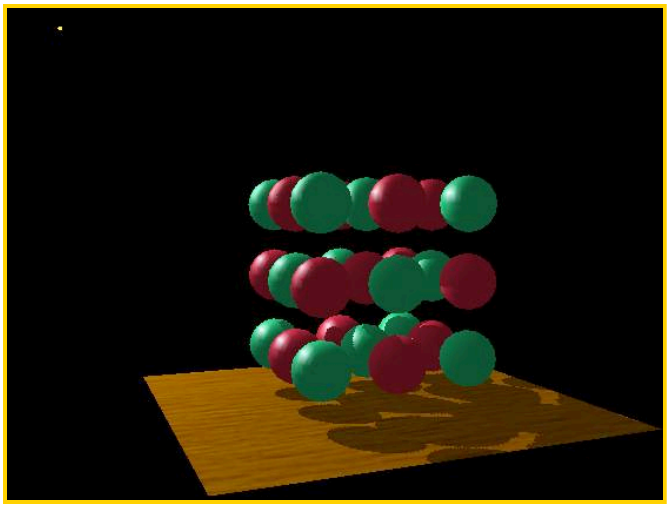
Shadow Volume Advantages

- Omni-directional approach
  - Not just spotlight frustums as with shadow maps
- Automatic self-shadowing
  - Everything can shadow everything, including self
  - Without *shadow acne* artifacts as with shadow maps
- Window-space shadow determination
  - Shadows accurate to a pixel
  - Or sub-pixel if multisampling is available
- Required stencil buffer broadly supported today
  - OpenGL support since version 1.0 (1991)
  - Direct3D support since DX6 (1998)

Shadow Volume Disadvantages

- Ideal light sources only
  - Limited to local point and directional lights
  - No area light sources for soft shadows
- Requires polygonal models with connectivity
  - Models must be closed (2-manifold)
  - Models must be free of non-planar polygons
- Silhouette computations are required
  - Can burden CPU
  - Particularly for dynamic scenes
- Inherently multi-pass algorithm
- Consumes lots of GPU fill rate

Shadow Mapping



Shadow Mapping

- Image-space shadow determination
  - Lance Williams published the basic idea in 1978
  - Completely image-space algorithm
    - no knowledge of scene's geometry is required
    - must deal with aliasing artifacts
  - Well known software rendering technique
    - Pixar's RenderMan uses the algorithm
    - Basic shadowing technique for Toy Story, etc.

The Shadow Mapping Concept (1)

- Depth testing from the light's point-of-view
  - Two pass algorithm
  - First, render depth buffer from the light's point-of-view
    - the result is a "depth map" or "shadow map"
    - essentially a 2D function indicating the depth of the closest pixels to the light
  - This depth map is used in the second pass

The Shadow Mapping Concept (2)

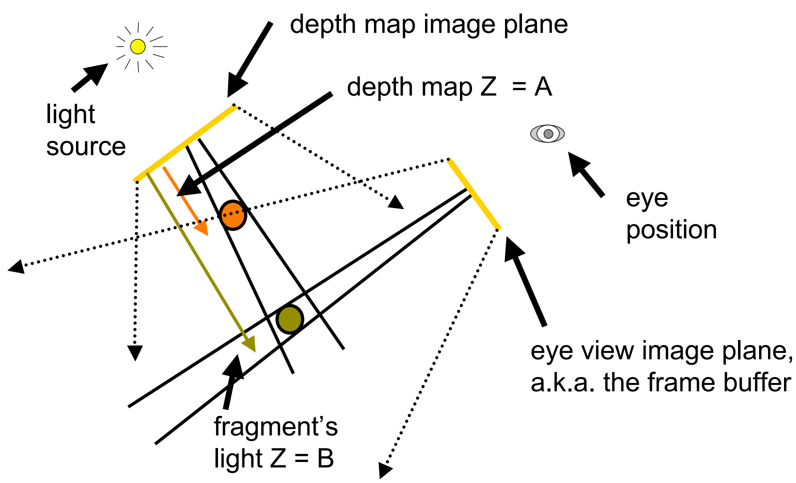
- Shadow determination with the depth map
  - Second, render scene from the eye's point-of-view
  - For each rasterized fragment
    - determine fragment's XYZ position relative to the light
    - compare the depth value at light position XY in the depth map to fragment's light position Z

The Shadow Mapping Concept (3)

- The Shadow Map Comparison
  - Two values
    - A = Z value from depth map at fragment's light XY position
    - B = Z value of fragment's XYZ light position
  - If B is greater than A, then there must be something closer to the light than the fragment
    - then the fragment is shadowed
  - If A and B are approximately equal, the fragment is lit

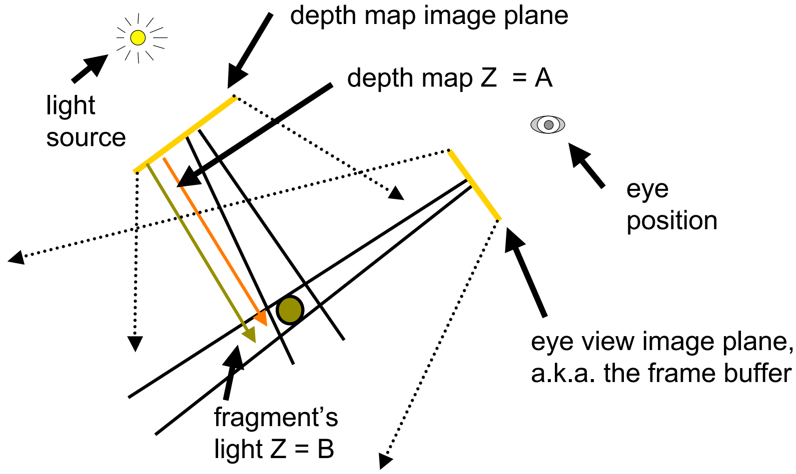
Shadow Mapping in 2D (1)

The A < B shadowed fragment case



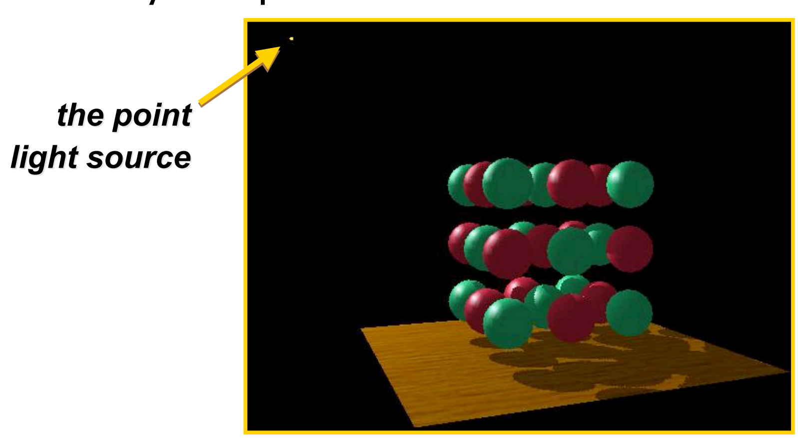
Shadow Mapping in 2D (2)

The A ≅ B unshadowed fragment case



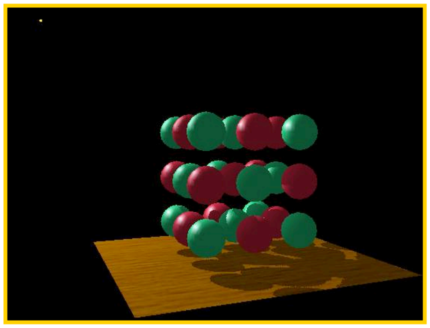
Visualizing the Shadow Mapping Technique (1)

- A fairly complex scene with shadows

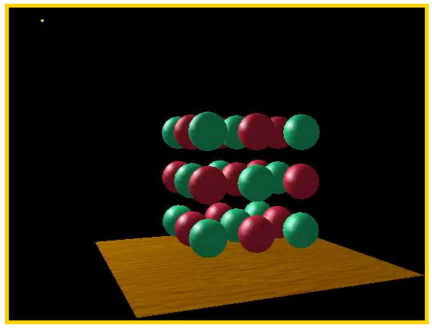


Visualizing the Shadow Mapping Technique (2)

- Compare with and without shadows



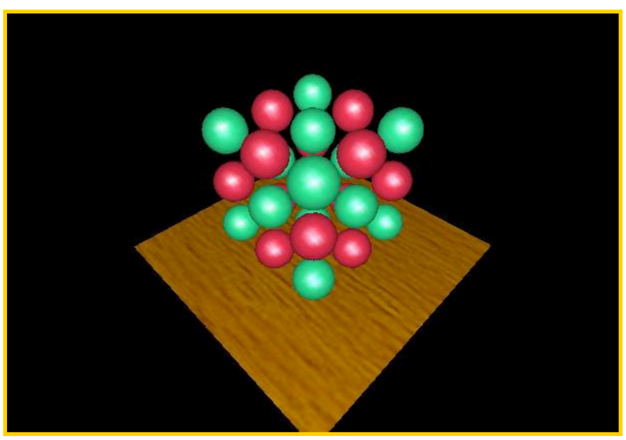
with shadows

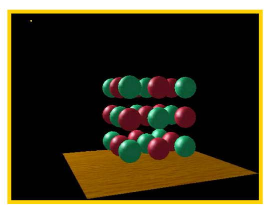


without shadows

Visualizing the Shadow Mapping Technique (3)

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

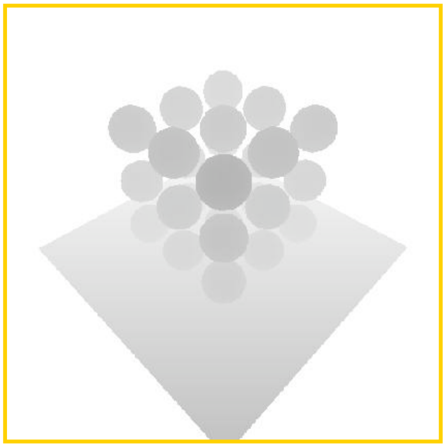


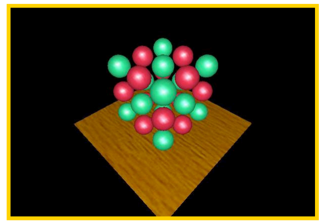


FYI: from the eye's point-of-view again

Visualizing the Shadow Mapping Technique (4)

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

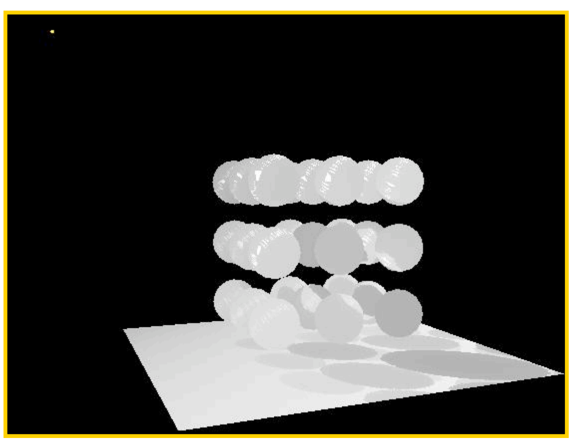


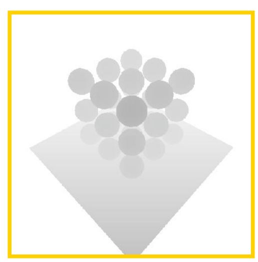


FYI: from the light's point-of-view again

Visualizing the Shadow Mapping Technique (5)

- Projecting the depth map onto the eye's view

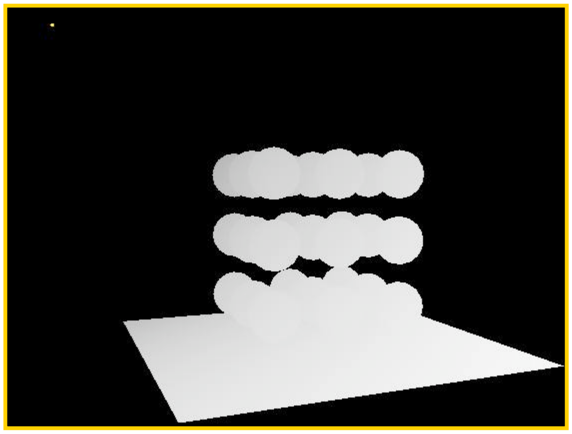


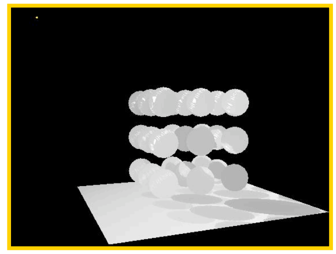


FYI: depth map for light's point-of-view again

Visualizing the Shadow Mapping Technique (6)

- Projecting light's planar distance onto eye's view

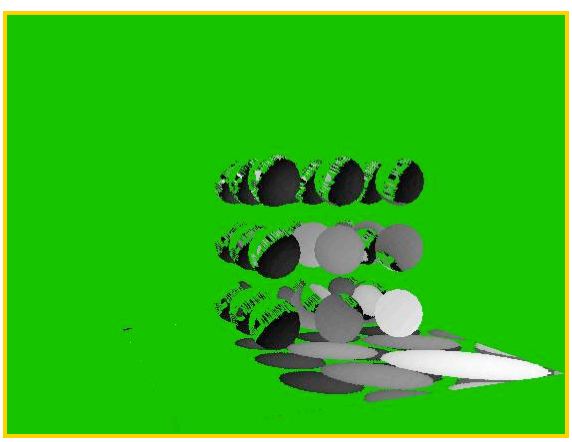




Visualizing the Shadow Mapping Technique (6)

- Comparing light distance to light depth map

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

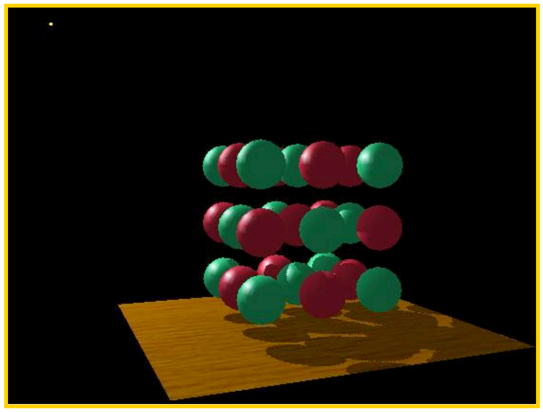


Non-green is where shadows should be

Visualizing the Shadow Mapping Technique (7)

- Scene with shadows

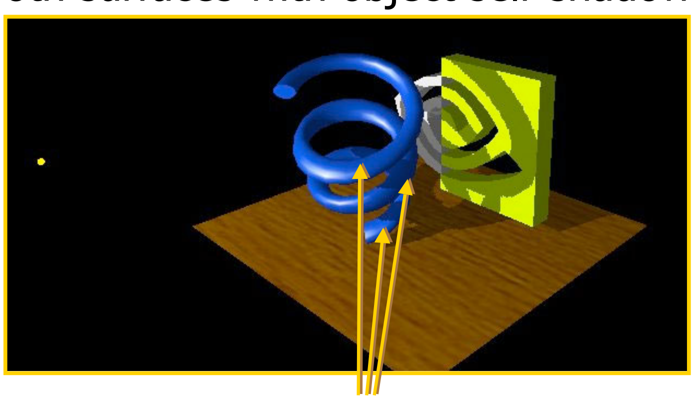
Notice how specular highlights never appear in shadows



Notice how curved surfaces cast shadows on each other

More Examples

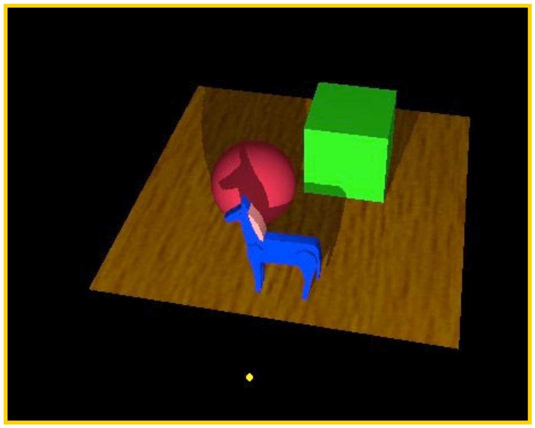
- Smooth surfaces with object self-shadowing



Note object self-shadowing

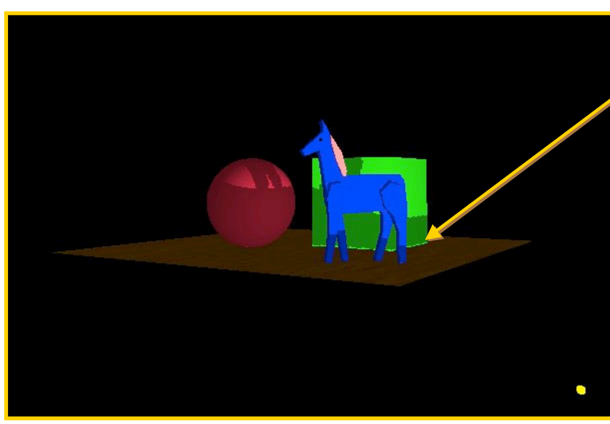
More Examples

- Complex objects all shadow



More Examples

- Even the floor casts shadow



Note shadow leakage due to infinitely thin floor

Could be fixed by giving floor thickness

Luxo Jr. in Real-time using Shadow Mapping

- Steve Jobs at MacWorld 2001 Japan shows this on a Mac with OpenGL using hardware shadow mapping

Luxo Jr. Demo Details

- Luxo Jr. has two animated lights and one overhead light
  - Three shadow maps dynamically generated per frame
- Complex geometry (cords and lamp arms) all correctly shadowed
- User controls the view, shadowing just works
- Real-time Luxo Jr. is technical triumph for OpenGL
- Only available in OpenGL.

(Images are from web cast video of Apple's MacWorld Japan announcement.)