Shadows Volumes

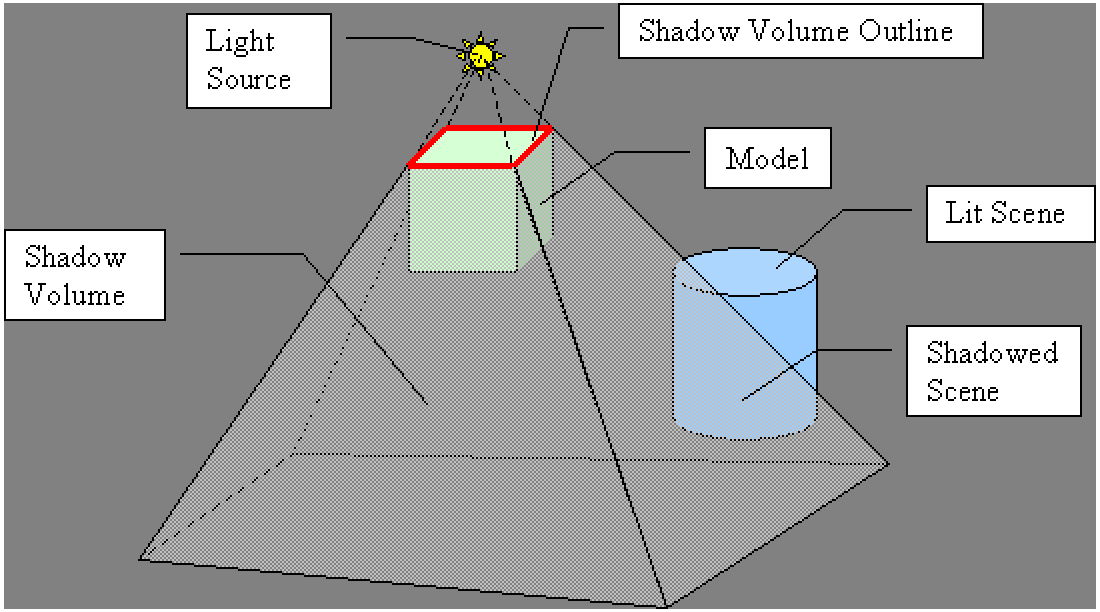
CSCI 4229/5229
Computer Graphics
Fall 2006

The Goal

- Realistic shadows
  - Shadows of objects on the floor and walls
  - Shadows of objects on each other
  - Shadows of each object on itself (if concave)
- Important depth cues
  - Relative positions of objects
  - Relative sizes of objects

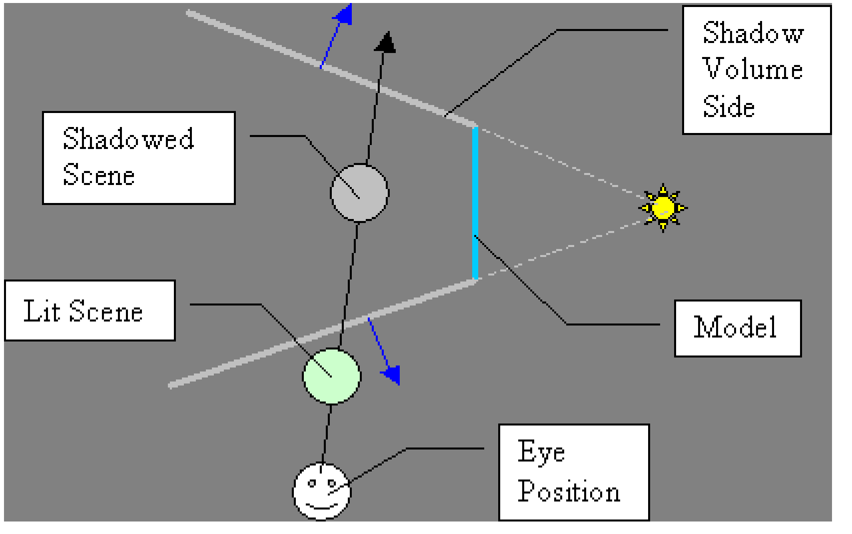
Shadow Volumes

- The volume corresponding to the shadow cast by a facet of each object
  - Potentially multiple shadow volumes per object
  - Shadow of the object is the combination of all shadow volumes for the object



Shadow Volume Algorithm

- Count transitions in and out of shadow volumes
  - Increment of in, decrement for out
  - Similar to polygon winding rule for in/out
- Lit areas has value of zero (initial value)



Bicubic Bézier Patch

$$
\begin{aligned}
P = &\ (1-v)^3 \Big((1-u)^3 P_{00} + 3(1-u)^2 u P_{01} + 3(1-u)u^2 P_{02} + u^3 P_{03} \Big) \\
+ &\ 3(1-v)^2 v \Big((1-u)^3 P_{10} + 3(1-u)^2 u P_{11} + 3(1-u)u^2 P_{12} + u^3 P_{13} \Big) \\
+ &\ 3(1-v)v^2 \Big((1-u)^3 P_{20} + 3(1-u)^2 u P_{21} + 3(1-u)u^2 P_{22} + u^3 P_{23} \Big) \\
+ &\ v^3 \Big((1-u)^3 P_{30} + 3(1-u)^2 u P_{31} + 3(1-u)u^2 P_{32} + u^3 P_{33} \Big) \\
= &\ (1-u)^3 \Big((1-v)^3 P_{00} + 3(1-v)^2 v P_{10} + 3(1-v)v^2 P_{20} + v^3 P_{30} \Big) \\
+ &\ 3(1-u)^2 u \Big((1-v)^3 P_{01} + 3(1-v)^2 v P_{11} + 3(1-v)v^2 P_{21} + v^3 P_{31} \Big) \\
+ &\ 3(1-u)u^2 \Big((1-v)^3 P_{02} + 3(1-v)^2 v P_{12} + 3(1-v)v^2 P_{22} + v^3 P_{32} \Big) \\
+ &\ u^3 \Big((1-v)^3 P_{03} + 3(1-v)^2 v P_{13} + 3(1-v)v^2 P_{23} + v^3 P_{33} \Big)
\end{aligned}
$$

Bicubic Bézier Patch Normal

$$
\begin{aligned}
\frac{\partial P}{\partial u} &= -3(1-u)^2 && \Big((1-v)^3 P_{00} + 3(1-v)^2 v P_{10} + 3(1-v)v^2 P_{20} + v^3 P_{30}\Big) \\
&+ 3(1-3u)(1-u) && \Big((1-v)^3 P_{01} + 3(1-v)^2 v P_{11} + 3(1-v)v^2 P_{21} + v^3 P_{31}\Big) \\
&+ 3u(2-3u) && \Big((1-v)^3 P_{02} + 3(1-v)^2 v P_{12} + 3(1-v)v^2 P_{22} + v^3 P_{32}\Big) \\
&+ 3u^2 && \Big((1-v)^3 P_{03} + 3(1-v)^2 v P_{13} + 3(1-v)v^2 P_{23} + v^3 P_{33}\Big)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial P}{\partial v} &= -3(1-v)^2 && \Big((1-u)^3 P_{00} + 3(1-u)^2 u P_{01} + 3(1-u)u^2 P_{02} + u^3 P_{03}\Big) \\
&+ 3(1-3v)(1-v) && \Big((1-u)^3 P_{10} + 3(1-u)^2 u P_{11} + 3(1-u)u^2 P_{12} + u^3 P_{13}\Big) \\
&+ 3v(2-3v) && \Big((1-u)^3 P_{20} + 3(1-u)^2 u P_{21} + 3(1-u)u^2 P_{22} + u^3 P_{23}\Big) \\
&+ 3v^2 && \Big((1-u)^3 P_{30} + 3(1-u)^2 u P_{31} + 3(1-u)u^2 P_{32} + u^3 P_{33}\Big)
\end{aligned}
$$

$$
N = \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v}
$$

The Stencil Buffer

- Buffer of 1, 4, 8, 16, 24 or 32 bits (often 8)
- One value for each pixel
- Accessed indirectly via operations on color buffer
- Can be used test as a stencil
  - Pixels are only drawn where the stencil buffer allows
- Exercised significantly by the shadow volume algorithms

Stencil Buffer Bits (a bit dated)

OpenGL Implementation	**Stencil Bits**
Most software implementations	8
3Dlabs Permedia II	1
SGI Indigo2 Extreme	4
SGI Octane MXI	8
ATI Rage 128	8
NVIDIA RIVA TNT	8
SGI Onyx2 InfiniteReality	1 or 8

- Sometimes 32 total bits for depth/stencil

Enabling the Stencil Buffer

- Need hardware support
  - glutInitDisplayMode (.... | GLUT_STENCIL);
- Must be enabled explicitly
  - glEnable(GL_STENCIL_TEST);
- Stencil operations only happen if there is both hardware support and it is enabled
  - Stencil tests always pass if not supported or not enabled
  - Test size with *glGetIntegerv(GL_STENCIL_BITS,&k);*

glStencilFunc(func,ref,mask)

- Decides how the stencil buffer effects drawing
  - GL_ALWAYS, GL_NEVER fixed function
  - GL_EQUAL, GL_LESS, GL_GREATER, GL_LEQUAL, GL_GEQUAL, GL_NOTEQUAL compares masked stencil and reference values
- If the test passes (is true) the pixel is drawn
  - GL_LESS => Draw when ref&mask < buf&mask

glStencilOp(fail,Zfail,Zpass)

- Determines what happens to the stencil buffer if
  - fail: the stencil test fails
  - Zfail: the Z-buffer test fails
  - Zpass: the Z-buffer test passes
- Options:
  - GL_KEEP no change
  - GL_ZERO set to zero
  - GL_REPLACE set to reference value
  - GL_INCR, GL_DECR increment or decrement
  - GL_INVERT bitwise inversion
  - GL_INCR_WRAP, GLDEC_WRAP (OpenGL 1.4)

Z-Pass Algorithm

- Render scene with lights off
  - All shadows and sets Z-buffer
- Make Z-buffer and color buffer read-only
- Render facets facing eye and pass depth test
  - Increment stencil buffer, depth and color unchanged
- Render facets opposite eye and pass depth test
  - Decrement stencil buffer, depth and color unchanged
- Make Z-buffer and color buffer read-write
- Render scene with lighting on and stencil=0

Z-pass Pros and Cons

- Works for objects of arbitrary shape
  - Cast shadows on walls, other objects and itself
- Fast and has hardware support
  - Does require 4 passes through scene
  - Face culling cuts effort in half on shadow passes
- Does not always work
  - Fails when eye is in the shadow
  - Fails when shadow volume clipped by front plane
  - Hollow objects (like spout of teapot)

Fixing Z-Pass

- Start at the back instead of the front
- Officially known as the Z-Fail algorithm
- Sometimes called Carmacks' Reverse
- Fixes the problem when the eye is in the shadow, but really just moves the problem to the back
- Still fails if shadow volumes are clipped by the back plane
  - Finite Z buffer size can be a problem
  - Fix by adjusting *infinity* adaptively

Z-Fail Algorithm

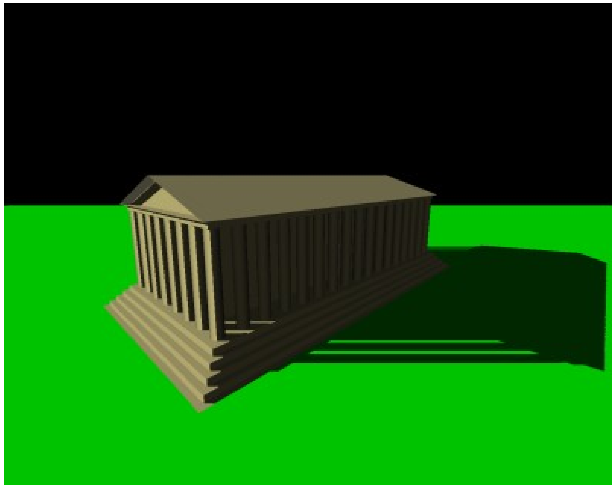
- Render scene with lights off
  - All shadows and sets Z-buffer
- Make Z-buffer and color buffer read-only
- Render facets **opposite** eye and **fail** depth test
  - Increment stencil buffer, depth and color unchanged
- Render facets **facing** eye and **fail** depth test
  - Decrement stencil buffer, depth and color unchanged
- Make Z-buffer and color buffer read-write
- Render scene with lighting on and stencil=0

Other methods

- Z-pass generally several times faster than Z-fail
  - The front object can hide lots objects behind
- ZP+ corrects Z-pass failures
  - Adds *front cap* to correct light/shadow count
- Shadow Mapping
  - Requires hardware support to do efficiently
    - SGI Reality Engine
    - NVIDIA GeFORCE
  - Supported by vendor OpenGL extensions

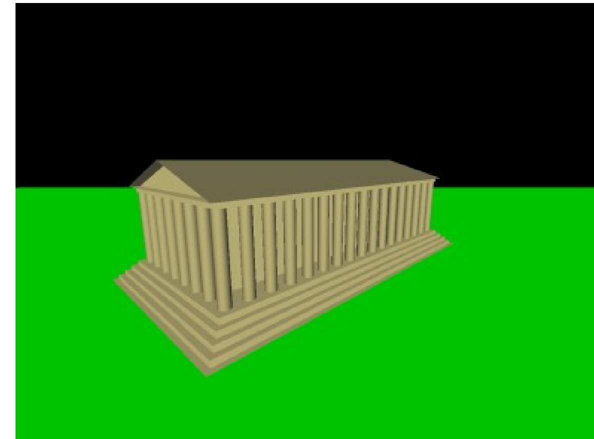
Shadow Mapping

- Project with light as viewpoint
- Depth buffer from light
- Light/shadow determined just like visibility
  - Objects in light foremost in depth buffer
  - Objects in shadow depth obscured
- Requires second depth buffer
- Not in vanilla OpenGL
- Used in *Toy Story* etc.

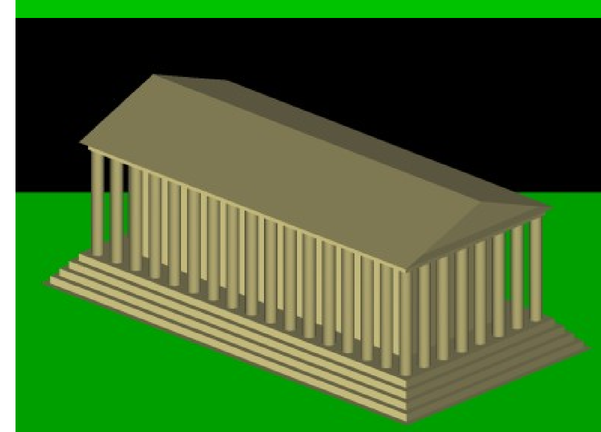


Shadow Map Example

No Shadows



Light View



Light View Depth