Shadows Volumes

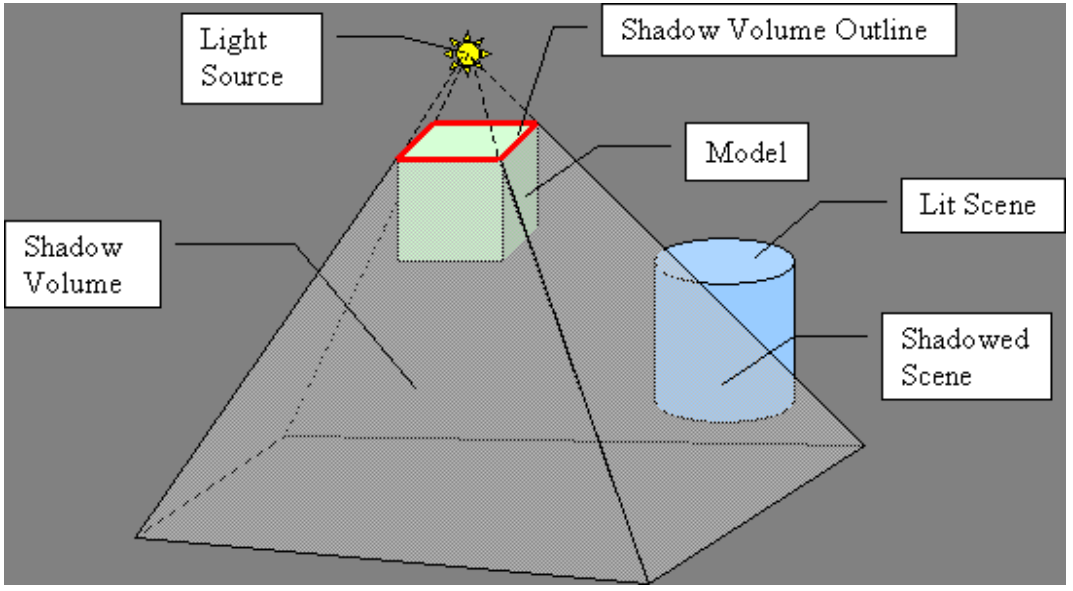
CSCI 4229/5229
Computer Graphics
Fall 2023

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

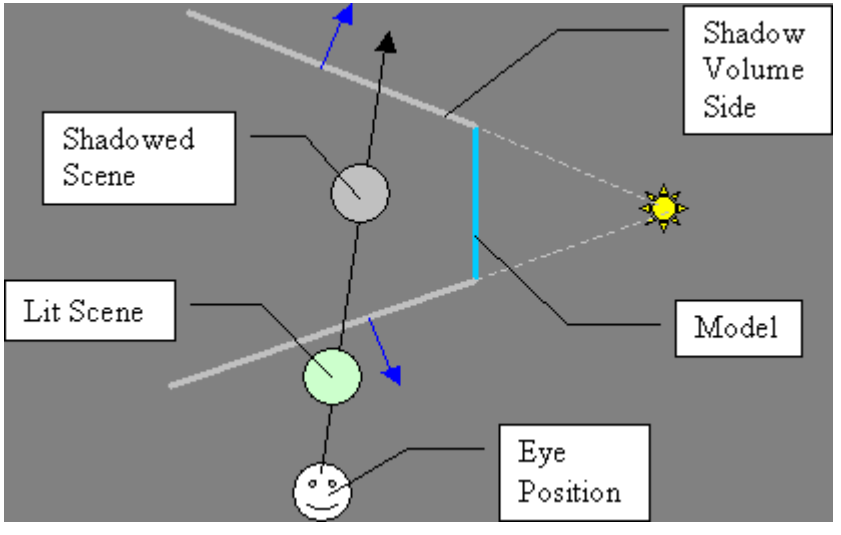
Shadow Volumes

- 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)



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

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

Improved 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