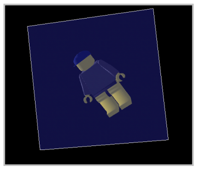
Selection and Picking
Transparency



CS148: Intro to CG
Instructor: Dan Morris
TA: Sean Walker
July 28, 2005

Outline for today

- Selection
- Video break
- Transparency

Selection in OpenGL

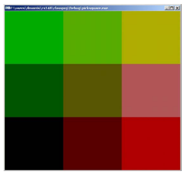
- Usually when we render our scene, the results go to the framebuffer
- We can also tell OpenGL “don’t render anything, just keep track of what *would* have been rendered”
- We can use this to find out what objects live in a certain volume
- We can use this to find out what objects the mouse has clicked on

Basic overview of selection

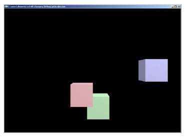
- Use glRenderMode(GL_SELECT) to tell OpenGL we’re doing selection, not rendering
- Use glSelectBuffer() to give OpenGL a place to tell us what objects are selected
- Use gluPickMatrix() to define a viewing volume that’s right around the user’s mouse position (so only those objects *don’t* get clipped)
- Render our scene, Using glPushName() to assign “names” to objects as we go through our scene
- Use glRenderMode(GL_RENDER) to tell OpenGL to go back to rendering, and to ask OpenGL which names were selected (not entirely clipped)

Selection examples

[picksquare.cpp]

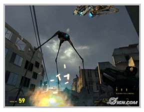


[pickcube.cpp]



Video Break: Half-Life 2 Tech Demo

- Be on the lookout for:
  - Lighting
  - Animation
  - Physics
  - Texture vs. geometry

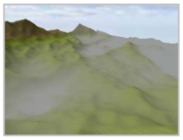


Outline for today

- Selection
- Video break
- Transparency

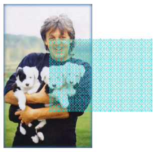
Transparency

- Many real-world objects are partially transparent
- Often light passes through media that affect light without reflecting it (fog, water, etc.)
- Often we want object on the screen that don't totally obstruct farther-away objects



Shading in OpenGL (so far)…

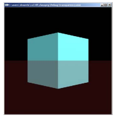
- So far, when we've colored our polygons, we've done something like this:
  - Compute the color of each vertex (lighting)
  - Interpolate those colors to get a color for each fragment (shading)
  - If this fragment passes the depth test, overwrite whatever is in the frame buffer to send this pixel to the screen
- If there's something in the frame buffer *behind* our polygon, it will get thrown away.
- This system doesn't allow us to represent *transparent* objects



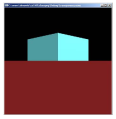
Alpha

- In OpenGL, colors are generally represented as four components: RGBA
- A is "alpha", which controls the transparency of an object.
  - 1.0: completely opaque
  - 0.0: completely transparent
- We can use the glColor4f or glMaterialfv commands to control the transparency of the current object

Quad color: (.8,.2,.2,**.2**)



Quad color: (.8,.2,.2,**1.0**)



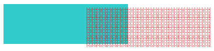
Blending in OpenGL [transparency.cpp]

- To enable *blending* in OpenGL, call glEnable(GL_BLEND)
- We call each new fragment the "source" and we call the current pixel in the framebuffer the "destination"
- OpenGL computes color like this:

  red = s_r*source.R + d_r*dest.R
  …same for all four channels (RGBA)…

  - source.R and dest.R are the source and dest colors
  - s_r and d_r are *blending* factors

What are good blending factors if I want a material that's 20% opaque (A=0.2) to blend in with the framebuffer?



Blending factors in OpenGL

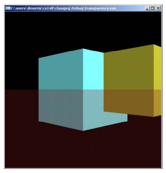
- We can control the way OpenGL compute its blending factors using:

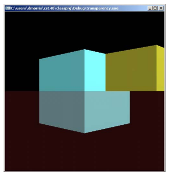
  glBlendFunc(sfactor,dfactor)

- sfactor and dfactor are chosen from:
  - GL_ZERO, GL_ONE, GL_DST_COLOR, GL_ONE_MINUS_DST_COLOR, GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA, GL_DST_ALPHA, GL_ONE_MINUS_DST_ALPHA
- **Which one corresponds to the blending factors we came up with on the previous slide?**

Complications…

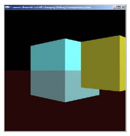
- I want to draw this image… two cubes with a xparent quad in front of them
- **What's wrong with this approach?**
  - Enable blending
  - Draw an opaque cube at z=0
  - Draw a transparent quad at z=2
  - Draw an opaque cube at z=1





A solution…

- Generally turn depth-buffer-writing off when drawing xparent objects:
  - glDepthMask(GL_FALSE);
- **What's *still* wrong with this approach?**



A solution…

- Generally try to draw transparent objects *after* you draw all of your opaque objects
- Sometimes this is too much of a pain, and you settle for less-than-perfect results
  - E.g. when an object has some transparent parts and you'd have to rip apart your code to draw them separately

Transparent Textures

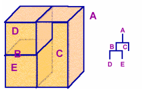
- Textures can have transparency too… very useful if you have multiple billboards in front of each other
- The .tga loader we gave you can load 32-bit (RGBA) .tga files
  - Not all images have alpha channels, but if your image does, you can load it…

Opaque texture

Transparent texture

Next Time

- Hidden-surface elimination
- Terrain
- Raytracing


D
B
E
C
A
A
B
C
D
E