Alpha Blending
Double Buffering
Picking

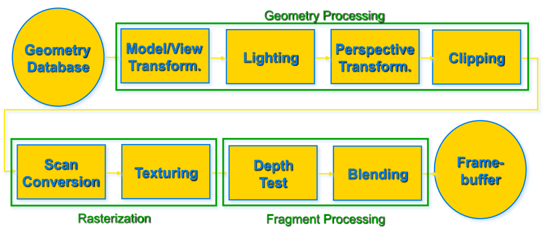
Wolfgang Heidrich

Course News

Assignment 2
- Due Monday!

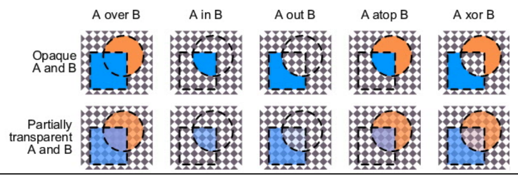
Reading
- No new reading this week

The Rendering Pipeline



Blending

How might you combine multiple elements?
- New color **A**, old color **B**



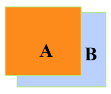
Alpha Blending (OpenGL)

Parameters:
- s = source color
- d = destination color
- b = source blend factor
- c = dest blend factor
- d' = bs + cd

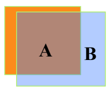
Where
- "Source" means "color/alpha of currently rendered primitive"
- "Destination" means framebuffer value

Over operator

- $d' = \alpha_s\, s + (1-\alpha_s) d$
- Examples: $\alpha_A=1$ $\alpha_B=0.4$



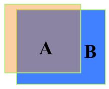
A over B: $d'=1*C_A + (1-1)*C_B$



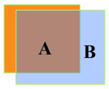
B over A: $d'=0.4*C_B + (0.6)*C_A$

Over operator

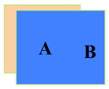
- $d' = \alpha_s s + (1-\alpha_s)d$
- Examples: $\alpha_A=0.4$ $\alpha_B=1.0$



A over B: $d'=0.4{*}C_A +(0.6){*}C_B$



Comparison from previous



B over A: $d'=1{*}C_B+(0){*}C_A$

Wolfgang Heidrich

Over operator

- $d' = \alpha_s s + (1-\alpha_s)d$
- $\alpha' = \alpha_s \alpha_s + (1- \alpha_s) \alpha_d$

Wolfgang Heidrich

OpenGL Blending

In OpenGL:

- Enable blending
  - *glEnable(GL_BLEND)*
- Specify alpha channel for colors
  - *glColor4f(r, g, b, alpha)*
- Specify blending function
  - *E.g: glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPH)*
    - C= alpha_new*Cnew + (1-alpha_new)*Cold

Wolfgang Heidrich

OpenGL Blending

Caveats:

- Note: alpha blending is an order-dependent operation!
  - *It matters which object is drawn first AND*
  - *Which surface is in front*
- For 3D scenes, this makes it necessary to keep track of rendering order explicitly
  - *Possibly also viewpoint-dependent!*
    - E.g. always draw “back” surface first
- Also note: interaction with z-buffer

Wolfgang Heidrich

Double Buffer

Wolfgang Heidrich

Double Buffering

Framebuffer:

- Piece of memory where the final image is written
- Problem:
  - *The display needs to read the contents, cyclically, while the GPU is already working on the next frame*
  - *Could result in display of partially rendered images on screen*
- Solution:
  - *Have TWO buffers*
    - Currently displayed (front buffer)
    - Render target for the next frame (back buffer)

Wolfgang Heidrich

Double Buffering

Front/back buffer:

- Each buffer has both color channels and a depth channel
  - *Important for advanced rendering algorithms*
  - *Doubles memory requirements!*

Switching buffers:

- At end of rendering one frame, simply exchange the pointers to the front and back buffer
- GLUT toolkit: glutSwapBuffers() function
  - *Different functions under windows/X11 if not using GLUT*

Wolfgang Heidrich

Triple Buffering

Used by some game consoles

- Why?

Wolfgang Heidrich

Picking/Object Selection

Wolfgang Heidrich

Interactive Object Selection

Move cursor over object, click

- How to decide what is below?

Ambiguity

- Many 3D world objects map to same 2D point

Common approaches

- Manual ray intersection
- Bounding extents
- Selection region with hit list (OpenGL support)

Wolfgang Heidrich

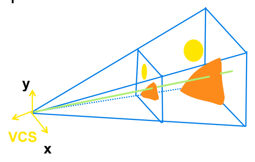
Manual Ray Intersection

Do all computation at application level

- Map selection point to a ray
- Intersect ray with all objects in scene.

Advantages

- No library dependence



Wolfgang Heidrich

Manual Ray Intersection

Do all computation at application level

- Map selection point to a ray
- Intersect ray with all objects in scene.

Advantages

- No library dependence

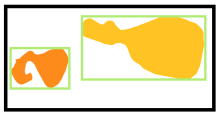
Disadvantages

- Difficult to program
- Slow: work to do depends on total number and complexity of objects in scene

Wolfgang Heidrich

Bounding Extents

Keep track of axis-aligned bounding rectangles



Advantages

- Conceptually simple
- Easy to keep track of boxes in world space

Wolfgang Heidrich

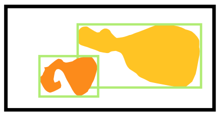
Bounding Extents

Disadvantages

- Low precision
- Must keep track of object-rectangle relationship

Extensions

- Do more sophisticated bound bookkeeping
  - *First level: box check. second level: object check*



Wolfgang Heidrich

OpenGL Picking

"Render" image in picking mode

- Pixels are never written to framebuffer
- Only store IDs of objects that would have been drawn

Procedure

- Set unique ID for each pickable object
- Call the regular sequence of glBegin/glVertex/glEnd commands
  - *If possible, skip glColor, glNormal, glTexCoord etc. for performance*

Wolfgang Heidrich

Select/Hit

OpenGL support

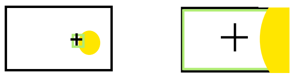
- Use small region around cursor for viewport
- Assign per-object integer keys (names)
- Redraw in special mode
- Store hit list of objects in region
- Examine hit list

Wolfgang Heidrich

Viewport

Small rectangle around cursor

- Change coord sys so fills viewport



Why rectangle instead of point?

- People aren't great at positioning mouse
  - *Fitts's Law: time to acquire a target is function of the distance to and size of the target*
- Allow several pixels of slop

Wolfgang Heidrich

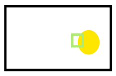
Viewport

Tricky to compute

- Invert viewport matrix, set up new orthogonal projection

Simple utility command

- gluPickMatrix(x,y,w,h,viewport)
  - *x,y: cursor point*
  - *w,h: sensitivity/slop (in pixels)*
- Push old setup first, so can pop it later



Wolfgang Heidrich

Render Modes

glRenderMode(mode)

- GL_RENDER: normal color buffer
  - *default*
- GL_SELECT: selection mode for picking
- (GL_FEEDBACK: report objects drawn)

Wolfgang Heidrich

Name Stack

- "names" are just integers

  glInitNames()
- flat list

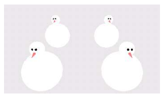
  glLoadName(name)
- or hierarchy supported by stack

  glPushName(name), glPopName
  - *Can have multiple names per object*
  - *Helpful for identifying objects in a hierarchy*

Wolfgang Heidrich

Hierarchical Names Example

```
for(int i = 0; i < 2; i++) {
  glPushName(i);
  for(int j = 0; j < 2; j++) {
    glPushMatrix();
    glPushName(j);
    glTranslatef(i*10.0,0,j * 10.0);
      glPushName(HEAD);
      glCallList(snowManHeadDL);
      glLoadName(BODY);
      glCallList(snowManBodyDL);
      glPopName();
    glPopName();
    glPopMatrix();
  }
  glPopName();
}
```



http://www.lighthouse3d.com/opengl/picking/

Wolfgang Heidrich

Hit List

- glSelectBuffer(int buffersize, GLuint *buffer)
  - *Where to store hit list data*
- If object overlaps with pick region, create hit record
- Hit record
  - *Number of names on stack*
  - *Minimum and minimum depth of object vertices*
    - Depth lies in the z-buffer range [0,1]
    - Multiplied by 2^32 -1 then rounded to nearest int
  - *Contents of name stack (bottom entry first)*

Wolfgang Heidrich

Using OpenGL Picking

Example code:

```
int numHitEntries;
GLuint buffer[1000];
glSelectBuffer( 1000, buffer );
glRenderMode( GL_SELECT );
drawStuff(); // includes name stack calls
numHitEntries= glRenderMode( GL_RENDER );
// now analyze numHitEntries different hit records
// in the selection buffer
...
```

Wolfgang Heidrich

Integrated vs. Separate Pick Function

Integrate: use same function to draw and pick

- Simpler to code
- Name stack commands ignored in render mode

Separate: customize functions for each

- Potentially more efficient
- Can avoid drawing unpickable objects

Wolfgang Heidrich

Select/Hit

Advantages

- Faster
  - *OpenGL support means hardware acceleration*
  - *Only do clipping work, no shading or rasterization*
- Flexible precision
  - *Size of region controllable*
- Flexible architecture
  - *Custom code possible, e.g. guaranteed frame rate*

Disadvantages

- More complex

Wolfgang Heidrich

Coming Up:

Next week

- Texture mapping

Wolfgang Heidrich