

Computer Graphics

Bing-Yu Chen
National Taiwan University

Introduction to OpenGL

- General OpenGL Introduction
- An Example OpenGL Program
- Drawing with OpenGL
- Transformations
- Animation and Depth Buffering
- Lighting
- Evaluation and NURBS
- Texture Mapping
- Advanced OpenGL Topics
- Imaging

modified from
Dave Shreiner, Ed Angel, and Vicki Shreiner.
An Interactive Introduction to OpenGL Programming.
ACM SIGGRAPH 2001 Conference Course Notes #54.
& *ACM SIGGRAPH 2004 Conference Course Notes #29.*

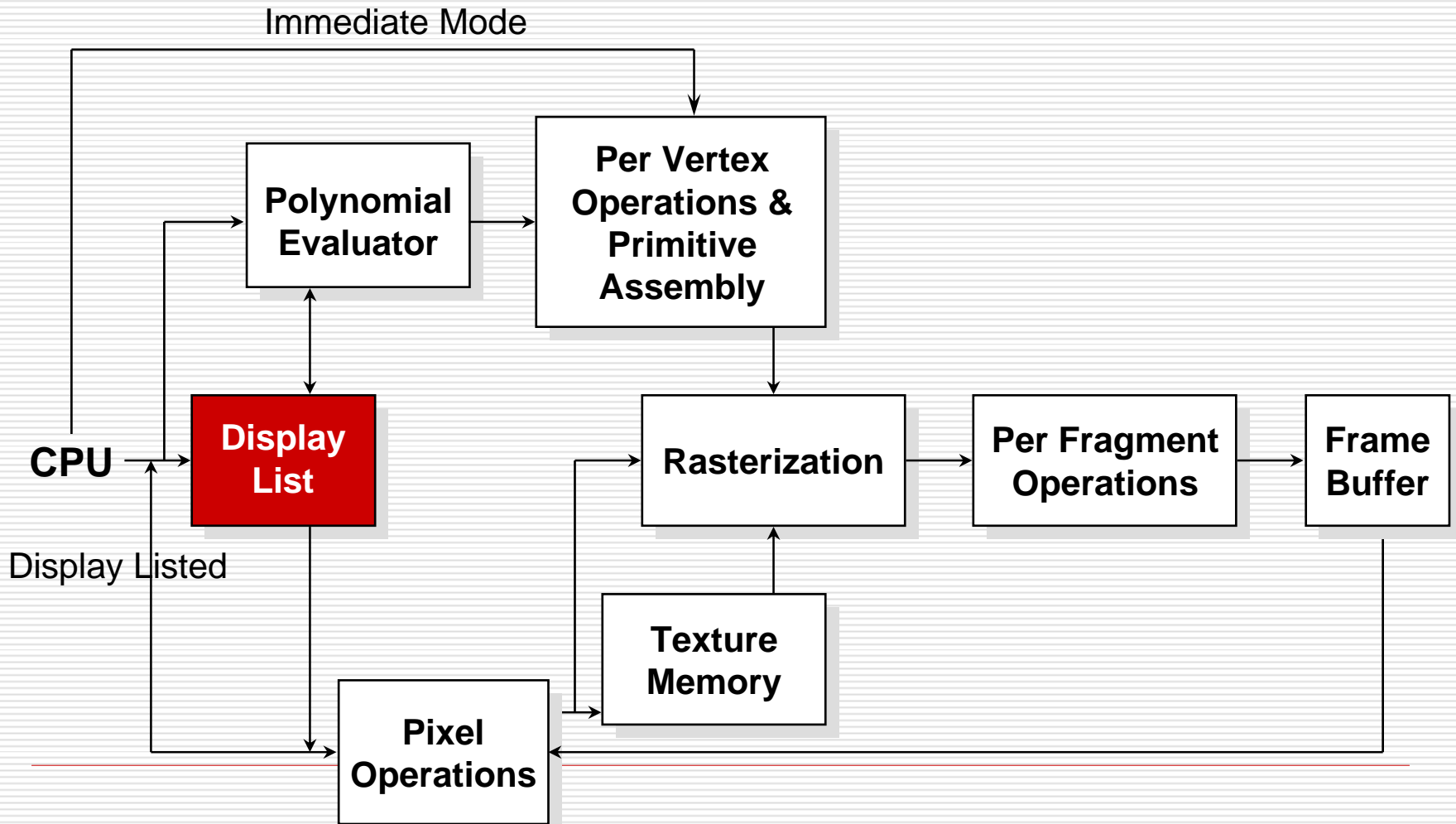
Advanced OpenGL Topics

- Display Lists and Vertex Arrays
 - Alpha Blending and Antialiasing
 - Using the Accumulation Buffer
 - Fog
 - Feedback & Selection
 - Fragment Tests and Operations
 - Using the Stencil Buffer
-

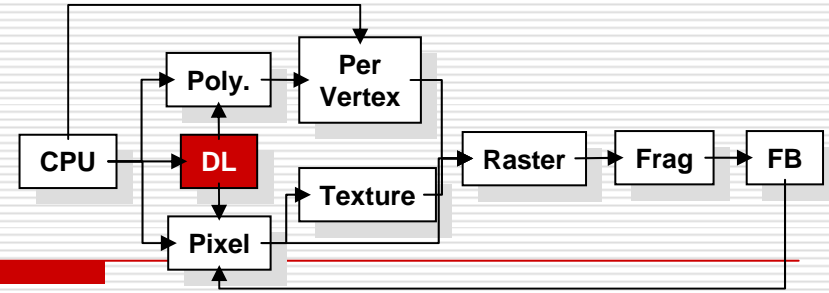
Immediate Mode versus Display Listed Rendering

- Immediate Mode Graphics
 - Primitives are sent to pipeline and display right away
 - No memory of graphical entities
 - Display Listed Graphics
 - Primitives placed in display lists
 - Display lists kept on graphics server
 - Can be redisplayed with different state
 - Can be shared among OpenGL graphics contexts
-

Immediate Mode versus Display Lists



Display Lists



- Creating a display list

```
GLuint id;
void init( void )
{
    id = glGenLists( 1 );
    glNewList( id, GL_COMPILE );
    /* other OpenGL routines */
    glEndList();
}
```

- Call a created list

```
void display( void )
{
    glCallList( id );
}
```

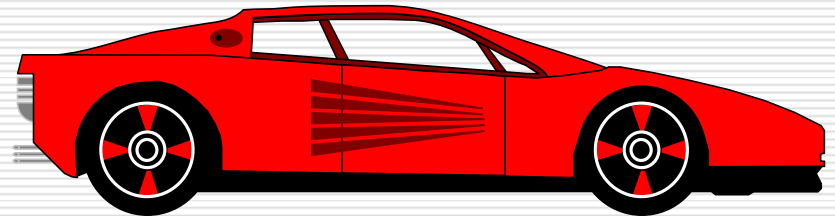
Display Lists

- ❑ Not all OpenGL routines can be stored in display lists
 - ❑ State changes persist, even after a display list is finished
 - ❑ Display lists can call other display lists
 - ❑ Display lists are not editable, but you can fake it
 - make a list (A) which calls other lists (B, C, and D)
 - delete and replace B, C, and D, as needed
-

Display Lists and Hierarchy

- Consider model of a car
 - Create display list for chassis
 - Create display list for wheel

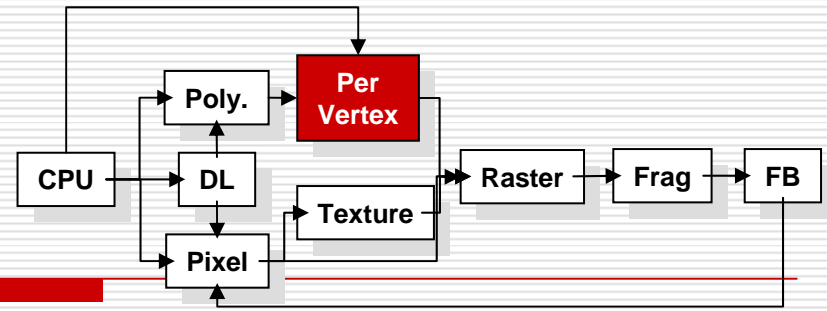
```
glNewList( CAR, GL_COMPILE );  
    glCallList( CHASSIS );  
    glTranslatef( ... );  
    glCallList( WHEEL );  
    glTranslatef( ... );  
    glCallList( WHEEL );  
    ...  
glEndList();
```



Advanced Primitives

- Vertex Arrays
 - Bernstein Polynomial Evaluators
 - basis for GLU NURBS
 - NURBS (Non-Uniform Rational B-Splines)
 - GLU Quadric Objects
 - sphere
 - cylinder (or cone)
 - disk (circle)
-

Vertex Arrays



- Pass arrays of vertices, colors, etc. to OpenGL in a large chunk

```
glVertexPointer( 3, GL_FLOAT, 0, coords )
glColorPointer( 4, GL_FLOAT, 0, colors )
glEnableClientState( GL_VERTEX_ARRAY )
glEnableClientState( GL_COLOR_ARRAY )
glDrawArrays( GL_TRIANGLE_STRIP, 0, numVerts );
```



- All active arrays are used in rendering

Why use Display Lists or Vertex Arrays?

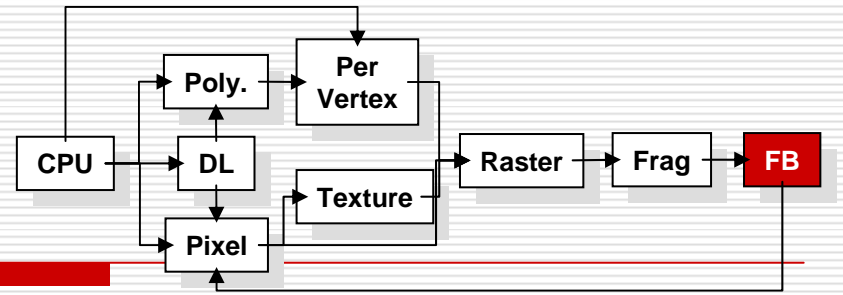
- ❑ May provide better performance than immediate mode rendering
 - ❑ Display lists can be shared between multiple OpenGL context
 - reduce memory usage for multi-context applications
 - ❑ Vertex arrays may format data for better memory access
-

Alpha: the 4th Color Component

- Measure of Opacity
 - simulate translucent objects
 - glass, water, etc.
 - composite images
 - antialiasing
 - ignored if blending is not enabled

`glEnable(GL_BLEND)`

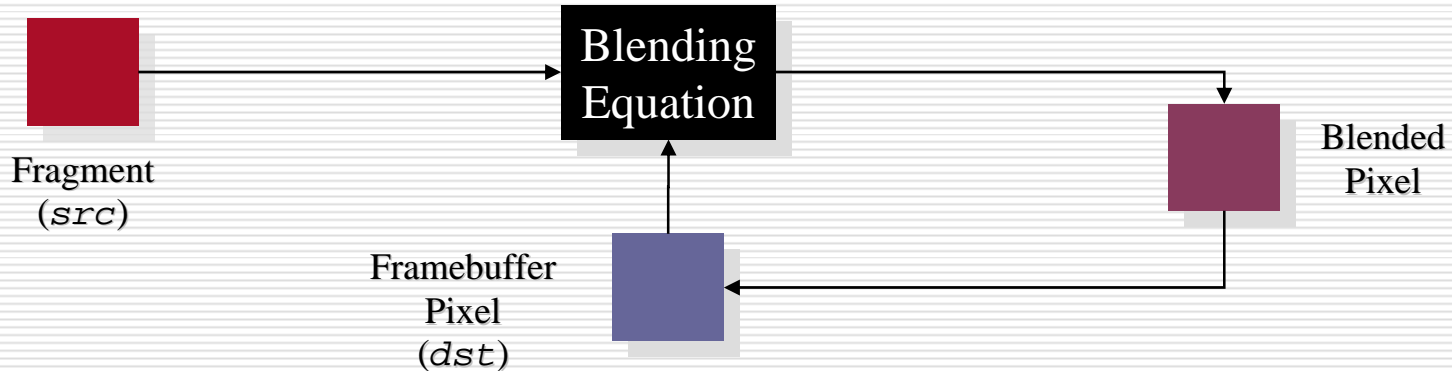
Blending



- Combine pixels with what's already in the framebuffer

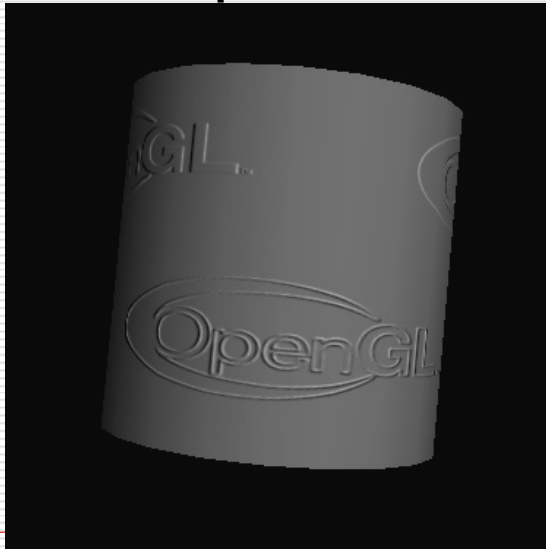
glBlendFunc(*src*, *dst*)

$$\vec{C}_r = src \vec{C}_f + dst \vec{C}_p$$



Multi-pass Rendering

- Blending allows results from multiple drawing passes to be combined together
- enables more complex rendering algorithms



Example of bump-mapping
done with a multi-pass
OpenGL algorithm

Antialiasing

- ❑ Removing the Jaggies

- ❑ `glEnable(mode)`

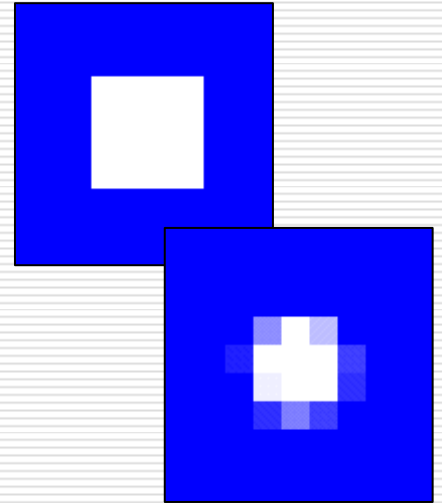
 - ❑ `GL_POINT_SMOOTH`

 - ❑ `GL_LINE_SMOOTH`

 - ❑ `GL_POLYGON_SMOOTH`

- alpha value computed by computing sub-pixel coverage

- available in both RGBA and colormap modes



Accumulation Buffer

- Problems of compositing into color buffers
 - limited color resolution
 - clamping
 - loss of accuracy
 - Accumulation buffer acts as a “floating point” color buffer
 - accumulate into accumulation buffer
 - transfer results to frame buffer
-

Accessing Accumulation Buffer

□ `glAccum(op, value)`

■ operations

□ within the accumulation buffer: *GL_ADD*,
GL_MULT

□ from read buffer: *GL_ACCUM*, *GL_LOAD*

□ transfer back to write buffer: *GL_RETURN*

■ `glAccum(GL_ACCUM, 0.5)` multiplies each value in write buffer by 0.5 and adds to accumulation buffer

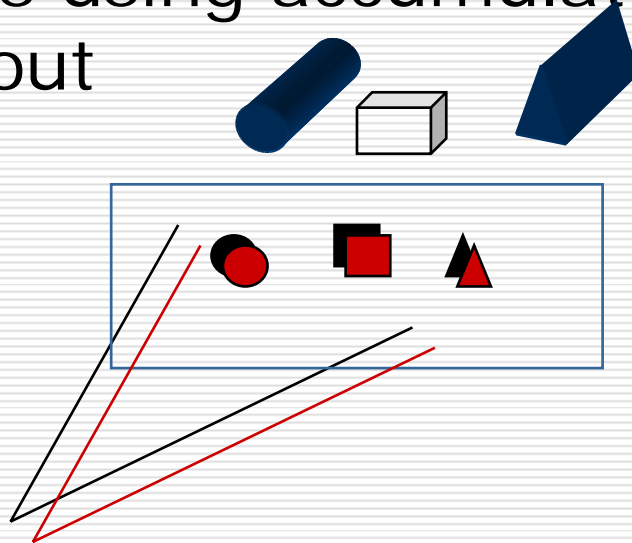
Accumulation Buffer Applications

- Compositing
 - Full Scene Antialiasing
 - Depth of Field
 - Filtering
 - Motion Blur
-

Full Scene Antialiasing :

Jittering the view

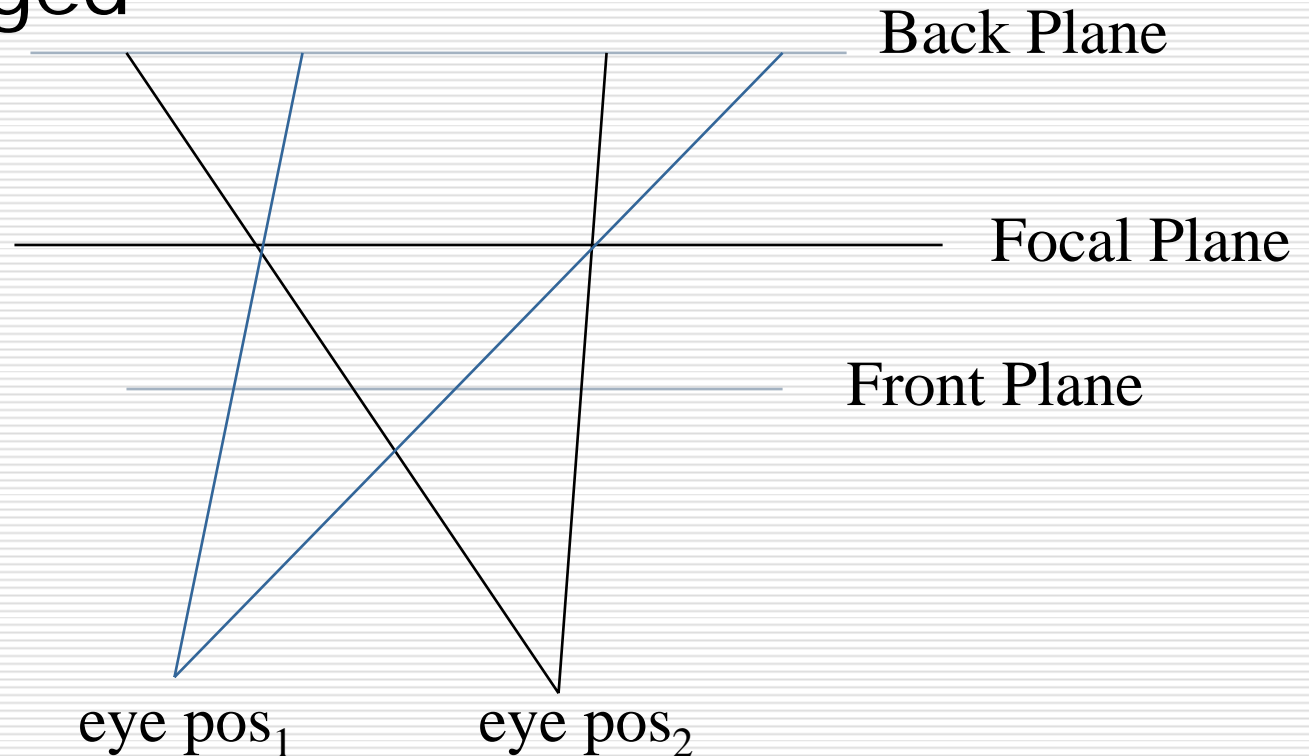
- Each time we move the viewer, the image shifts
 - Different aliasing artifacts in each image
 - Averaging images using accumulation buffer averages out these artifacts



Depth of Focus :

Keeping a Plane in Focus

- Jitter the viewer to keep one plane unchanged



Fog

- `glFog{if}(property, value)`
 - Depth Cueing
 - Specify a range for a linear fog ramp
 - `GL_FOG_LINEAR`
 - Environmental effects
 - Simulate more realistic fog
 - `GL_FOG_EXP`
 - `GL_FOG_EXP2`
-

Fog Tutorial


Fog

Fog equation

$$f = \frac{\text{end} - z}{\text{end} - \text{start}}$$

z is the distance in eye coordinates from origin to fragment being fogged.

Screen-space view



Command manipulation window

```
GLfloat color[4] = { 0.70 , 0.70 , 0.70 , 1.00 };  
glFogfv(GL_FOG_COLOR, color);  
glFogf(GL_FOG_START, 0.50 );  
glFogf(GL_FOG_END, 2.00 );  
glFogi(GL_FOG_MODE, GL_LINEAR);
```

Click on the arguments and move the mouse to modify values.

Feedback Mode

- Transformed vertex data is returned to the application, not rendered
 - useful to determine which primitives will make it to the screen
 - Need to specify a feedback buffer
`glFeedbackBuffer(size, type, buffer)`
 - Select feedback mode for rendering
`glRenderMode(GL_FEEDBACK)`
-

Selection Mode

- ❑ Method to determine which primitives are inside the viewing volume
- ❑ Need to set up a buffer to have results returned to you

`glSelectBuffer(size, buffer)`

- ❑ Select selection mode for rendering

`glRenderMode(GL_SELECT)`

Selection Mode (cont.)

- To identify a primitive, give it a name
 - “names” are just integer values, not strings
- Names are stack based
 - allows for hierarchies of primitives
- Selection Name Routines

`glLoadName(name)`

`glPushName(name)`

`glInitNames()`

Picking

- Picking is a special case of selection
 - Programming steps
 - restrict “drawing” to small region near pointer
 - use `gluPickMatrix()` on projection matrix
 - enter selection mode; re-render scene
 - primitives drawn near cursor cause hits
 - exit selection; analyze hit records
-

Picking Template

`glutMouseFunc(pickMe);`

```
void pickMe( int button, int state, int x, int y )
{
    GLuint nameBuffer[256];
    GLint hits;
    GLint myViewport[4];
    if (button != GLUT_LEFT_BUTTON ||
        state != GLUT_DOWN) return;
    glGetIntegerv( GL_VIEWPORT, myViewport );
    glSelectBuffer( 256, nameBuffer );
    (void) glRenderMode( GL_SELECT );
    glInitNames();
```

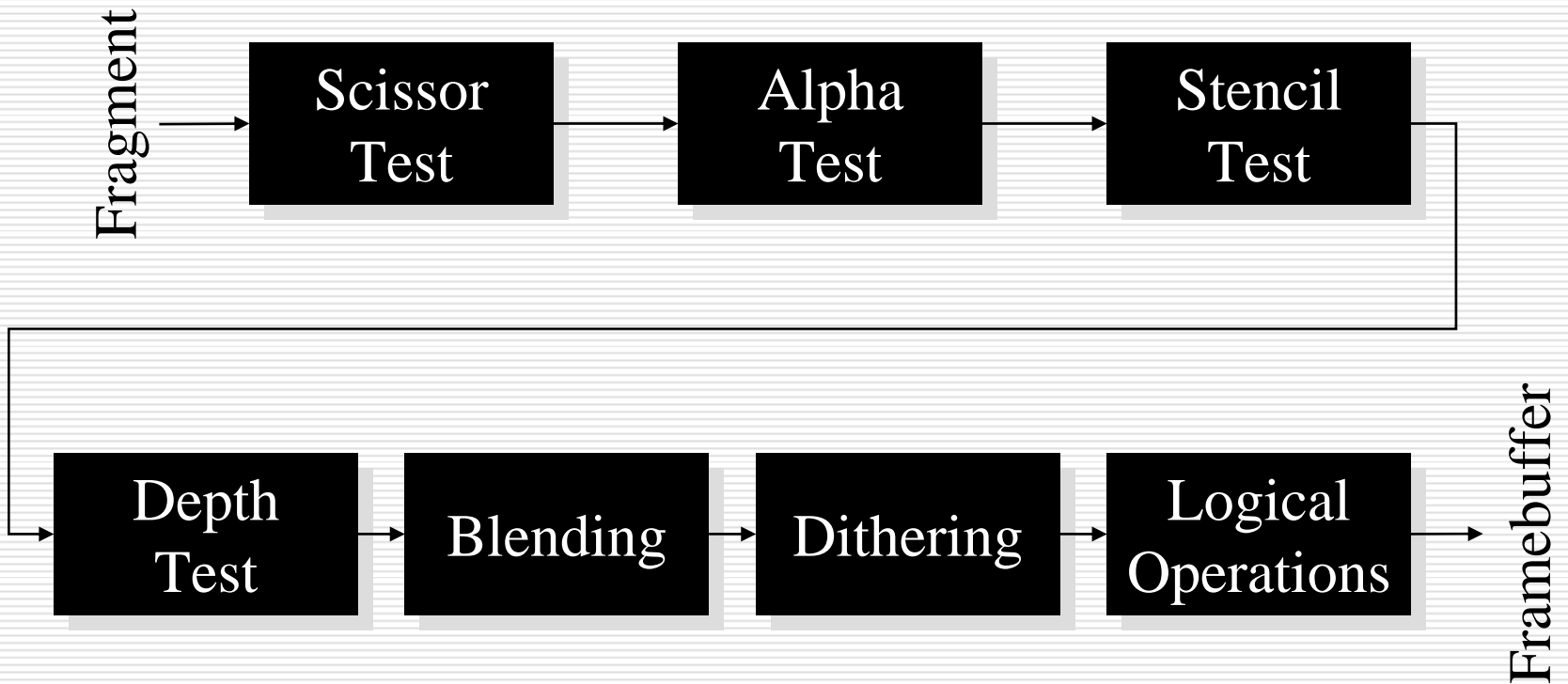
Picking Template (cont.)

```
glMatrixMode( GL_PROJECTION );
glPushMatrix();
glLoadIdentity();
gluPickMatrix( (GLdouble) x, (GLdouble)
               (myViewport[3]-y), 5.0, 5.0, myViewport );
/*   gluPerspective or glOrtho or other projection   */
glPushName( 1 );
/*   draw something   */
glLoadName( 2 );
/*   draw something else   */
glMatrixMode( GL_PROJECTION );
glPopMatrix();
hits = glRenderMode( GL_RENDER );
/*   process nameBuffer   */
}
```

Picking Ideas

- For OpenGL Picking Mechanism
 - only render what is pickable (e.g., don't clear screen!)
 - use an “invisible” filled rectangle, instead of text
 - if several primitives drawn in picking region, hard to use z values to distinguish which primitive is “on top”
 - Alternatives to Standard Mechanism
 - color or stencil tricks (for example, use `glReadPixels()` to obtain pixel value from back buffer)
-

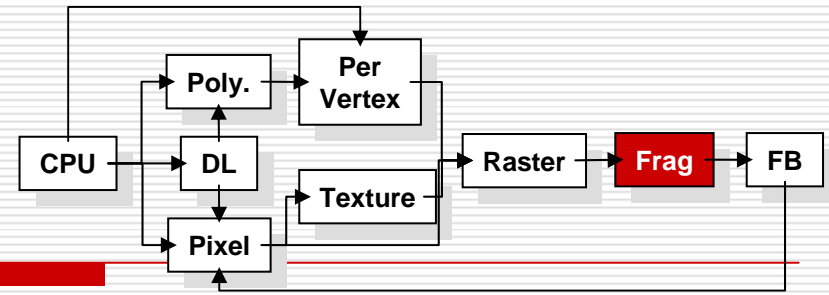
Getting to the Framebuffer



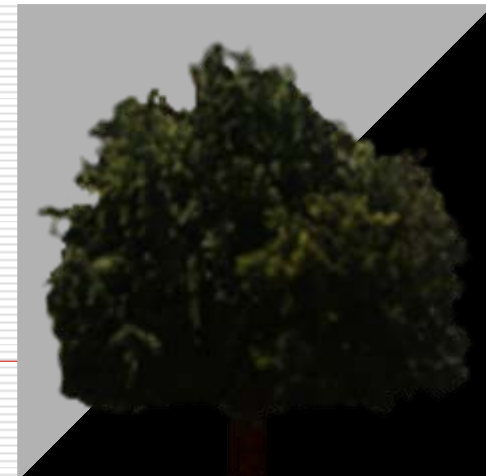
Scissor Box

- Additional Clipping Test
 - `glScissor(x, y, w, h)`
 - any fragments outside of box are clipped
 - useful for updating a small section of a viewport
 - affects `glClear()` operations
-

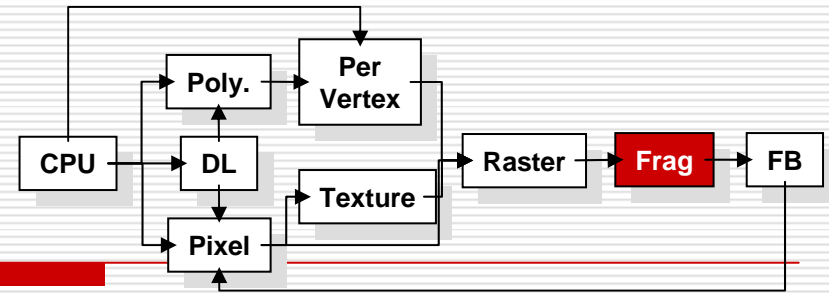
Alpha Test



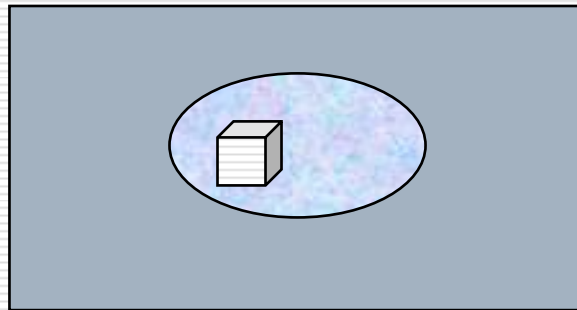
- Reject pixels based on their alpha value
- `glAlphaFunc(func, value)`
- `glEnable(GL_ALPHA_TEST)`
 - use alpha as a mask in textures



Stencil Buffer



- Used to control drawing based on values in the stencil buffer
 - Fragments that fail the stencil test are not drawn
 - Example: create a mask in stencil buffer and draw only objects not in mask area



Controlling Stencil Buffer

□ `glStencilFunc(func, ref, mask)`

- compare value in buffer with `ref` using `func`
- only applied for bits in `mask` which are 1
- `func` is one of standard comparison functions

□ `glStencilOp(fail, zfail, zpass)`

- Allows changes in stencil buffer based on passing or failing stencil and depth tests:
`GL_KEEP`, `GL_INCR`
-

Creating a Mask

- `glInitDisplayMode(...|GLUT_STENCIL|...);`
 - `glEnable(GL_STENCIL_TEST);`
 - `glClearStencil(0x0);`

 - `glStencilFunc(GL_ALWAYS, 0x1, 0x1);`
 - `glStencilOp(GL_REPLACE, GL_REPLACE,
GL_REPLACE);`
 - draw mask
-

Using Stencil Mask

- Draw objects where stencil = 1

```
glStencilFunc( GL_EQUAL, 0x1, 0x1 )
```

- Draw objects where stencil != 1

```
glStencilFunc( GL_NOTEQUAL, 0x1, 0x1 );
```

```
glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );
```

Dithering

- ❑ `glEnable(GL_DITHER)`
 - ❑ Dither colors for better looking results
 - Used to simulate more available colors
-

Logical Operations on Pixels

- Combine pixels using bitwise logical operations
 - `glLogicOp(mode)`
 - Common modes
 - `GL_XOR`
 - `GL_AND`
-

Advanced Imaging

Imaging Subset

- Only available if `GL_ARB_imaging` defined
 - Color matrix
 - Convolutions
 - Color tables
 - Histogram
 - MinMax
 - Advanced Blending
-