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.

What is OpenGL? & What can it do for me?

- OpenGL is
  - a computer graphics *rendering* API
    - generate high-quality color images by rendering with geometric and image primitives
    - create interactive applications with 3D graphics
  - window system independent
  - operating system independent

OpenGL as a Renderer

- Geometric primitives
  - points, lines and polygons
- Image Primitives
  - images and bitmaps
  - separate pipeline for images & geometry
    - linked through texture mapping
- Rendering depends on state
  - colors, materials, light sources, etc.

OpenGL Libraries

- OpenGL core library
  - OpenGL32 on Windows
  - GL on most unix/linux systems
- OpenGL Utility Library (GLU)
  - part of OpenGL
  - Provides functionality in OpenGL core but avoids having to rewrite code
  - NURBS, tessellators, quadric shapes, etc.

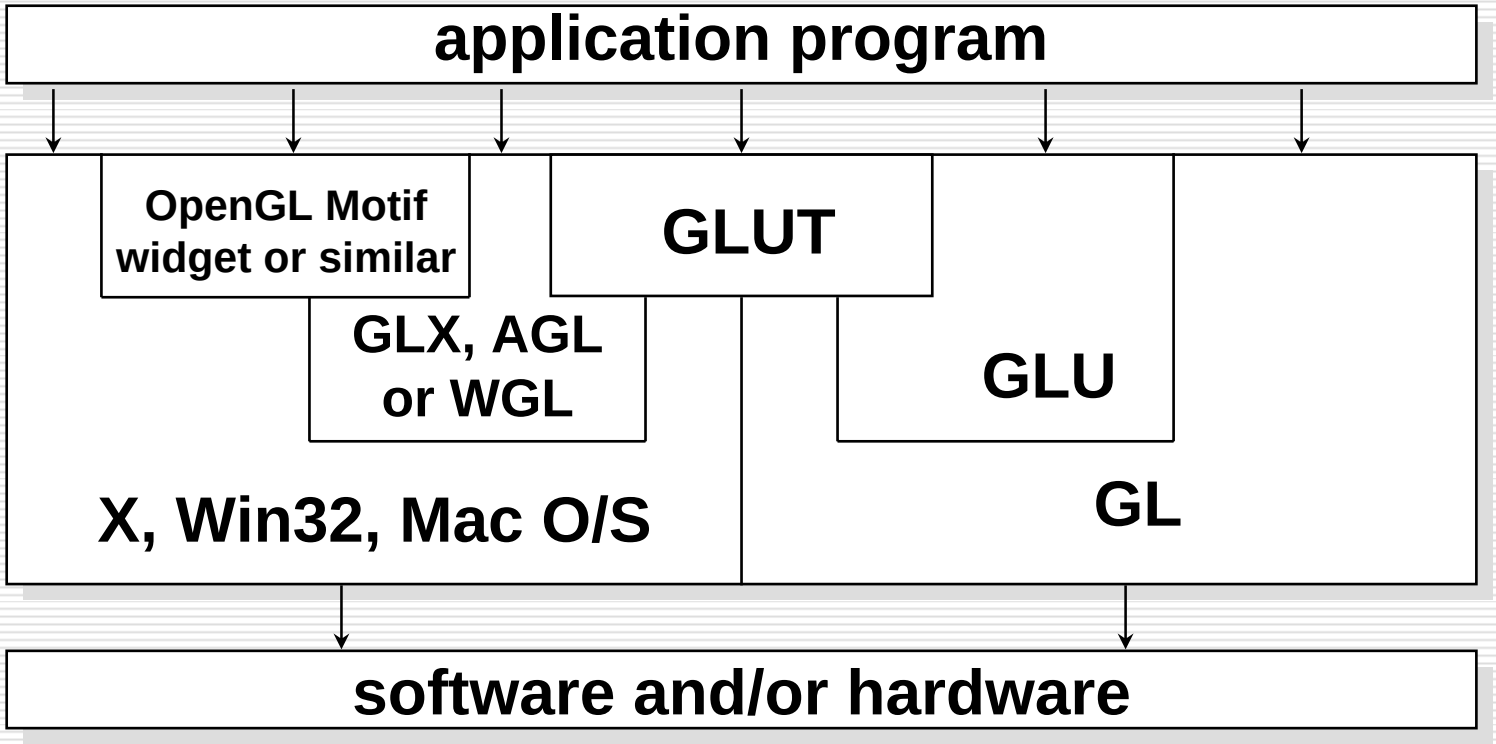
OpenGL Libraries

- Links with window system
  - GLX for X window systems
  - WGL for Widows
  - AGL for Macintosh
  - glue between OpenGL & windowing systems

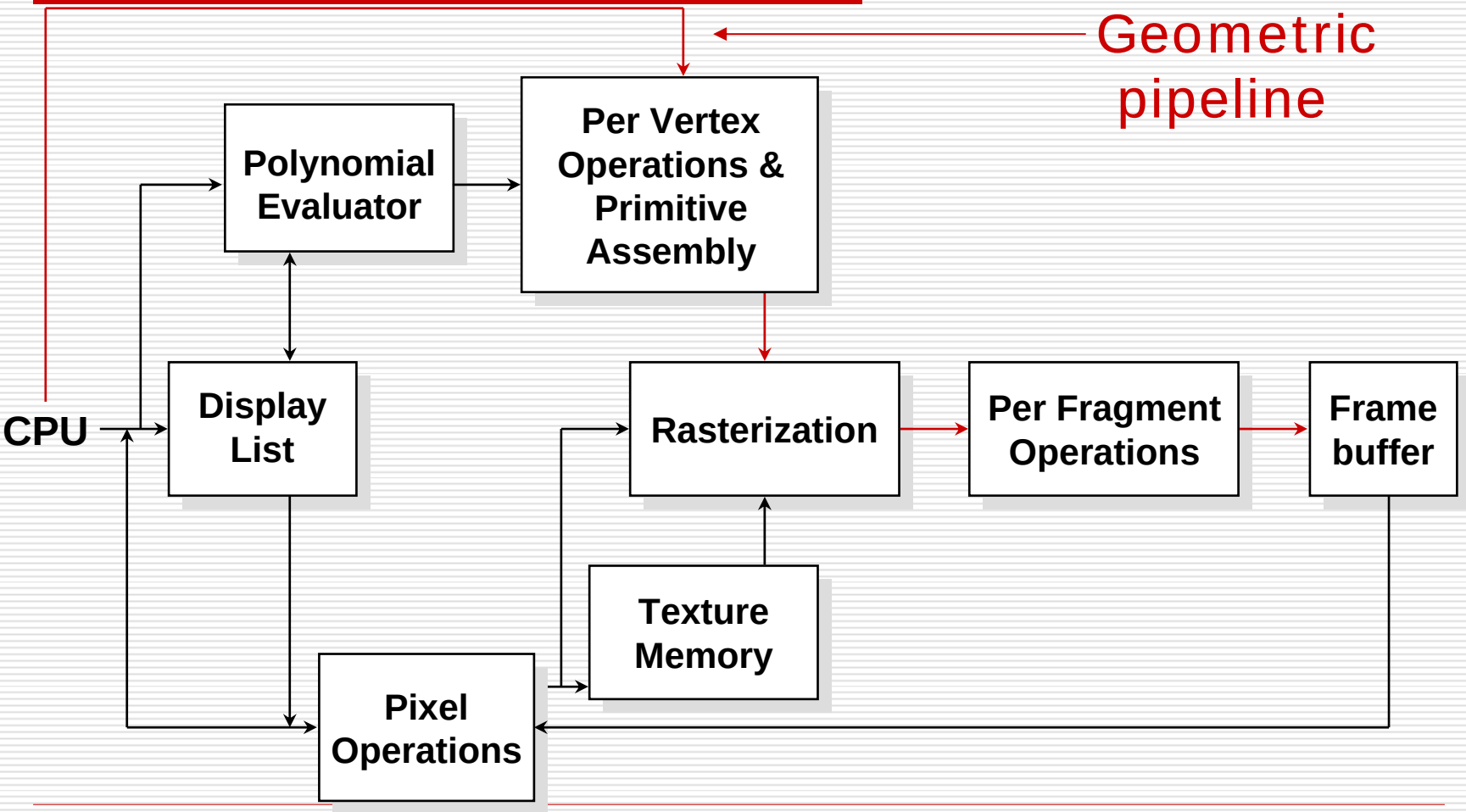
GLUT

- OpenGL Utility Library (GLUT)
  - Provides functionality common to all window systems
    - Open a window
    - Get input from mouse and keyboard
    - Menus
    - Event-driven
  - Code is portable but GLUT lacks the functionality of a good toolkit for a specific platform
    - Slide bars

OpenGL and Related APIs



OpenGL Architecture



OpenGL Functions

- Primitives
  - Points
  - Line Segments
  - Polygons
- Attributes
- Transformations
  - Viewing
  - Modeling
- Control
- Input (GLUT)

What is Required for your Programs

- Headers Files

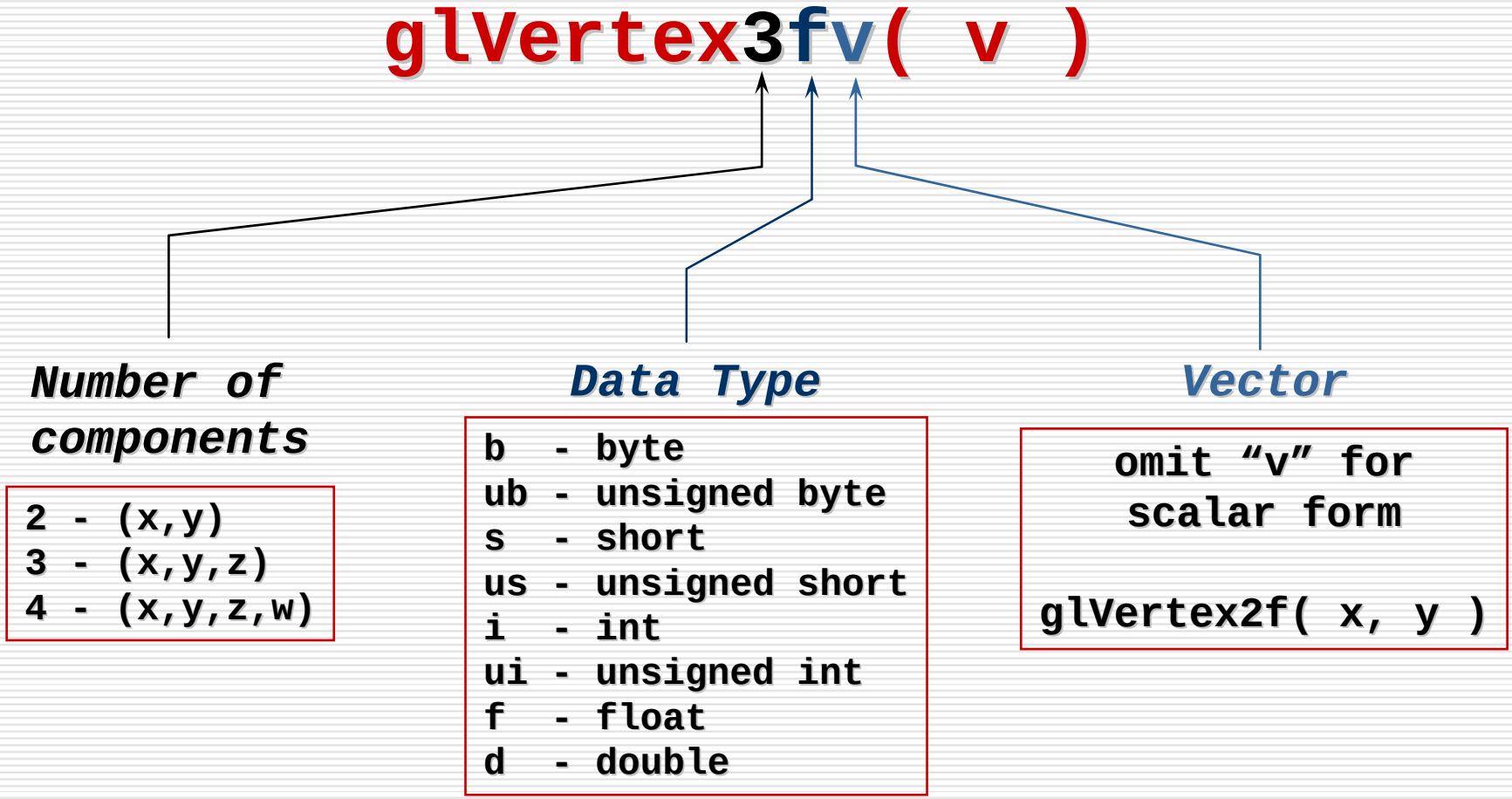
```
#include <GL/gl.h>
#include <GL/glext.h>
#include <GL/glu.h>
#include <GL/glut.h>
```

- Libraries
- Enumerated Types
  - OpenGL defines numerous types for compatibility
    - **GLfloat**, **GLint**, **GLenum**, etc.

Lack of Object Orientation

- OpenGL is not object oriented so that there are multiple functions for a given logical function,
  - e.g. **`glVertex3f`**, **`glVertex2i`**, **`glVertex3dv`**,.....
- Underlying storage mode is the same
- Easy to create overloaded functions in C++ but issue is efficiency

OpenGL Command Formats



OpenGL State

- OpenGL is a state machine
- OpenGL functions are of two types
  - Primitive generating
    - Can cause output if primitive is visible
    - How vertices are processes and appearance of primitive are controlled by the state
  - State changing
    - Transformation functions
    - Attribute functions

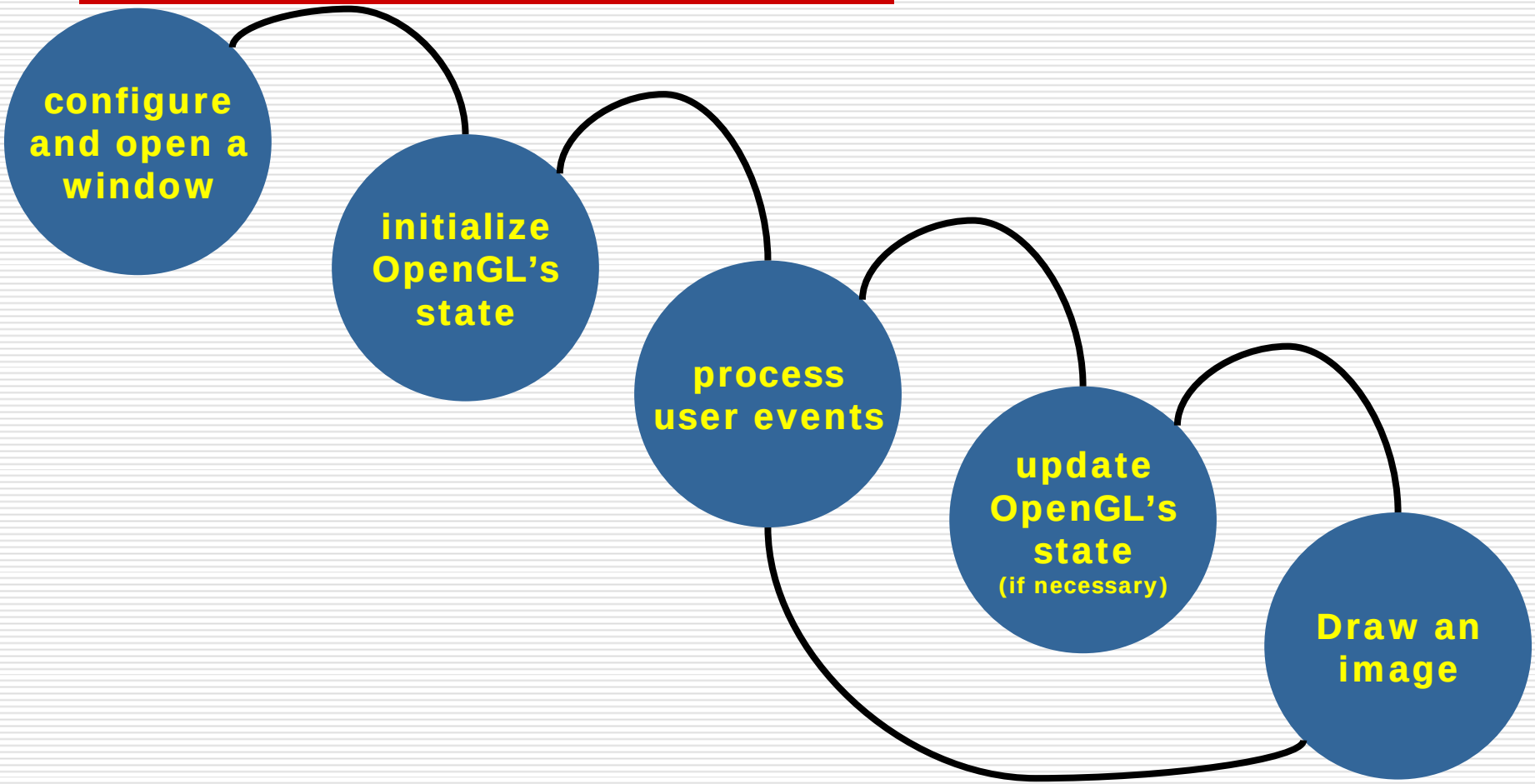
The OpenGL Pipeline

- Processing is controlled by setting OpenGL's *state*
  - colors, lights and object materials, texture maps
  - drawing styles, depth testing

GLUT Basics

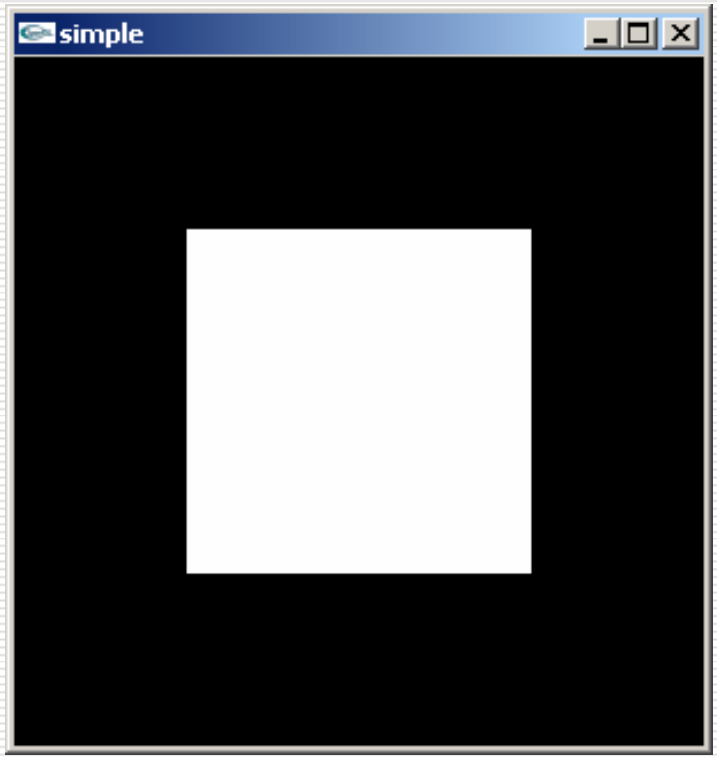
- Application Structure
  - Configure and open window
  - Initialize OpenGL state
  - Register input callback functions
    - render
    - resize
    - input: keyboard, mouse, etc.
  - Enter event processing loop

Sequence of Most OpenGL Programs



A Simple Program

- Generate a square on a solid background



simple.c

```
#include <GL/glut.h>
void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();
    glFlush();
}
int main(int argc, char** argv){
    glutCreateWindow("simple");
    glutDisplayFunc(mydisplay);
    glutMainLoop();
}
```

Event Loop

- Note that the program defines a *display callback* function named **`mydisplay`**
  - Every glut program must have a display callback
  - The display callback is executed whenever OpenGL decides the display must be refreshed, for example when the window is opened
  - The **`main`** function ends with the program entering an event loop

Notes on Compilation

- Unix/linux
  - Include files usually in …/include/GL
  - Compile with –lglut –lglu –lgl loader flags
  - May have to add –L flag for X libraries
  - Mesa implementation included with most linux distributions
  - Check web for latest versions of Mesa and glut

Compilation on Windows

- Visual C++ / Borland C
  - Get glut.h, glut32.lib and glut32.dll
  - Create a console application
  - Add opengl32.lib, glu32.lib, glut32.lib to project settings (under link tab)
- Cygwin (linux under Windows)
  - Can use gcc and similar makefile to linux
  - Use –lopengl32 –lglu32 –lglut32 flags

Another Sample Program

```
void main( int argc, char** argv )
{
  int mode = GLUT_RGB|GLUT_DOUBLE;
  glutInitDisplayMode( mode );
  glutCreateWindow( argv[0] );
  init();
  glutDisplayFunc( display );
  glutReshapeFunc( resize );
  glutKeyboardFunc( key );
  glutIdleFunc( idle );
  glutMainLoop();
}
```

OpenGL Initialization

- Set up whatever state you're going to use

```
void init( void )
{
  glClearColor( 0.0, 0.0, 0.0, 1.0 );
  glClearDepth( 1.0 );

  glEnable( GL_LIGHT0 );
  glEnable( GL_LIGHTING );
  glEnable( GL_DEPTH_TEST );
}
```

GLUT Callback Functions

- Routine to call when something happens
  - window resize or redraw
  - user input
  - animation
- “Register” callbacks with GLUT

```
glutDisplayFunc( display );
glutIdleFunc( idle );
glutKeyboardFunc( keyboard );
```

Rendering Callback

- Do all of your drawing here

```
glutDisplayFunc( display );
void display( void )
{
  glClear( GL_COLOR_BUFFER_BIT );
  glBegin( GL_TRIANGLE_STRIP );
    glVertex3fv( v[0] );
    glVertex3fv( v[1] );
    glVertex3fv( v[2] );
    glVertex3fv( v[3] );
  glEnd();
  glutSwapBuffers();
}
```

Idle Callbacks

- Use for animation & continuous update

```
glutIdleFunc( idle );

void idle( void )
{
  t += dt;
  glutPostRedisplay();
}
```

User Input Callbacks

- Process user input

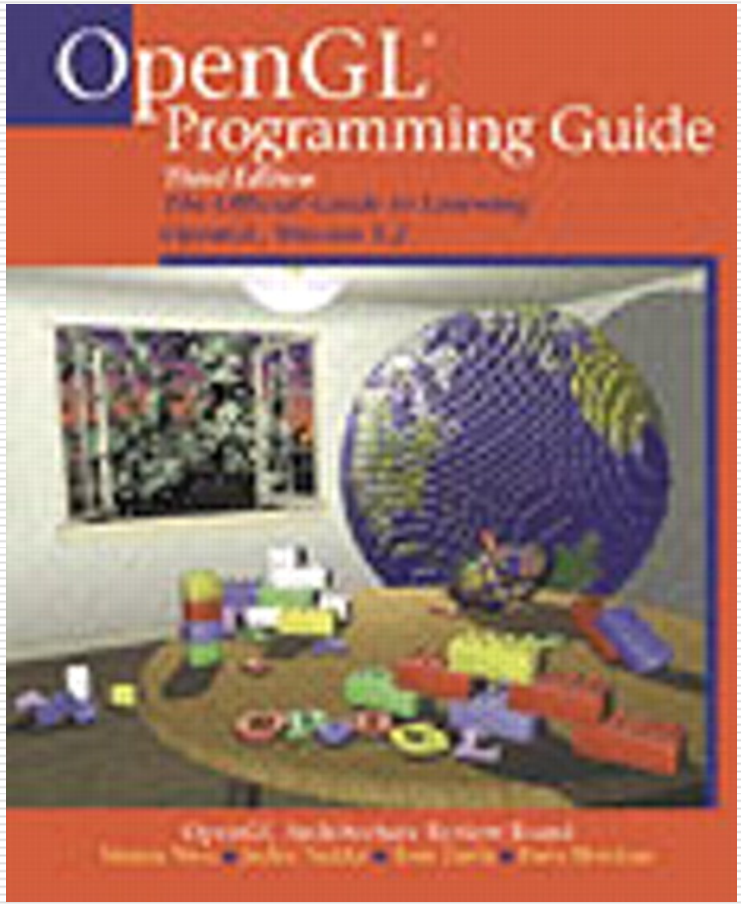
```
glutKeyboardFunc( keyboard );
void keyboard( unsigned char key, int x, int y )
{
  switch( key ) {
    case ‘q’ : case ‘Q’ :
      exit( EXIT_SUCCESS );
      break;
    case ‘r’ : case ‘R’ :
      rotate = GL_TRUE;
      glutPostRedisplay();
      break;
  }
}
```

On-Line Resources

- http://www.opengl.org/
  - start here; up to date specification and lots of sample code
- news:comp.graphics.api.opengl
- http://www.sgi.com/software/opengl/
- http://www.mesa3d.org/
  - Brian Paul's Mesa 3D
- http://www.cs.utah.edu/~narobins/opengl.html
  - very special thanks to Nate Robins for the OpenGL Tutors
  - source code for tutors available here!

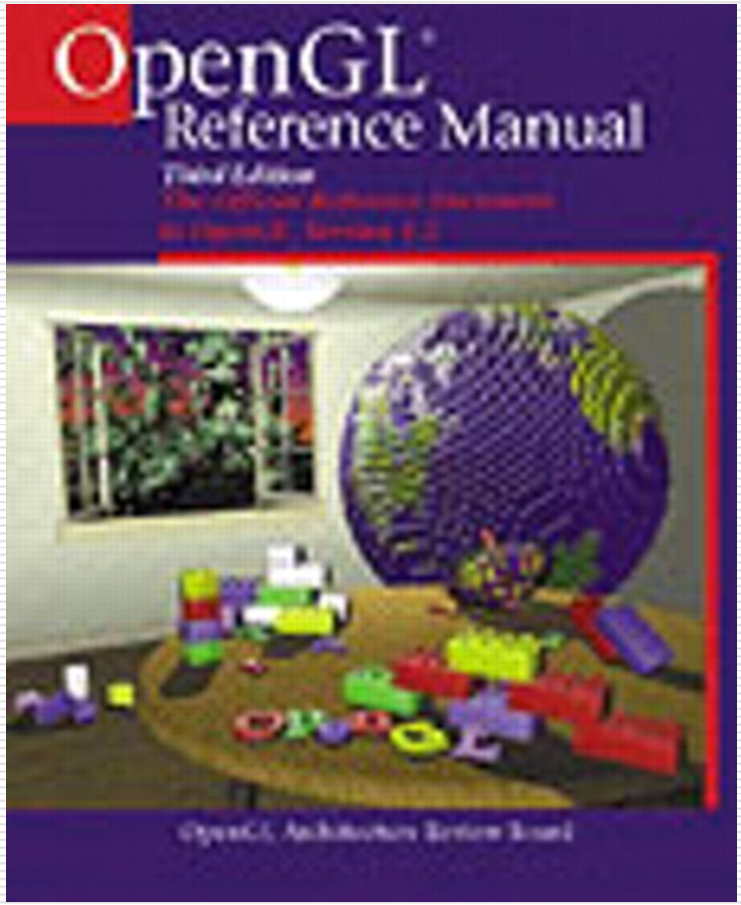
Books

- M. Woo,
  J. Neider,
  T. Davis,
  D. Shreiner.
  OpenGL® Programming Guide: The Official Guide to Learning OpenGL, ver. 1.2, 3rd. ed.,
  Addison-Wesley, 1999



Books

- D. Shreiner. *OpenGL® Reference Manual: The Official Reference Document to OpenGL, ver. 1.2, 3rd. ed.*, Addison-Wesley, 1999



Books

- E. Angel.
  Interactive Computer Graphics: A Top-Down Approach with OpenGL™, 3rd. ed.,
  Addison-Wesley, 2002