INTRODUCTION TO OPENGL AND GLUT

Yun Jang
jangy@sejong.ac.kr

Early History of APIs

- IFIPS (1973) formed two committees to come up with a standard graphics API
  - Graphical Kernel System (GKS)
    - 2D but contained good workstation model
  - Core
    - Both 2D and 3D
  - GKS adopted as IS0 and later ANSI standard (1980s)
- GKS not easily extended to 3D (GKS-3D)
  - Far behind hardware development

PHIGS and X

- Programmers Hierarchical Graphics System (PHIGS)
  - Arose from CAD community
  - Database model with retained graphics (structures)
- X Window System
  - DEC/MIT effort
  - Client-server architecture with graphics
- PEX combined the two
  - Not easy to use (all the defects of each)

SGI and GL

- Silicon Graphics (SGI) revolutionized the graphics workstation by implementing the pipeline in hardware (1982)
- To access the system, application programmers used a library called GL
- With GL, it was relatively simple to program three dimensional interactive applications

OpenGL

The success of GL lead to OpenGL (1992), a platform-independent API that was

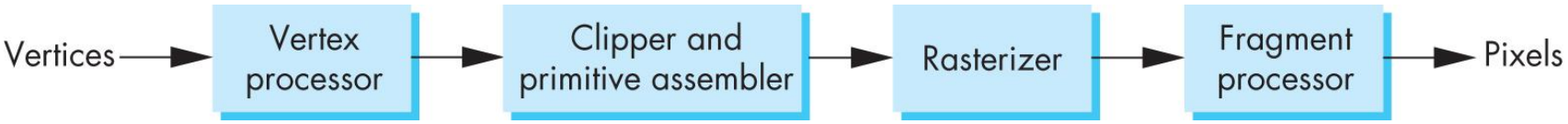
- Easy to use
- Close enough to the hardware to get excellent performance
- Focus on rendering
- Omitted windowing and input to avoid window system dependencies

OpenGL Evolution

- Originally controlled by an Architectural Review Board (ARB)
  - Members included SGI, Microsoft, Nvidia, HP, 3DLabs, IBM,…….
  - Now Kronos Group
  - Was relatively stable (through version 2.5)
    - Backward compatible
    - Evolution reflected new hardware capabilities
      - 3D texture mapping and texture objects
      - Vertex and fragment programs
  - Allows platform specific features through extensions

Modern OpenGL

- Performance is achieved by using GPU rather than CPU
- Control GPU through programs called shaders
- Application's job is to send data to GPU
- GPU does all rendering



OpenGL 3.1

- Totally shader-based
  - No default shaders
  - Each application must provide both a vertex and a fragment shader
- No immediate mode
- Few state variables
- Most 2.5 functions deprecated
- Backward compatibility not required

Other Versions

- OpenGL ES
  - Embedded systems
  - Version 1.0 simplified OpenGL 2.1
  - Version 2.0 simplified OpenGL 3.1
    - Shader based
- WebGL
  - Javascript implementation of ES 2.0
  - Supported on newer browsers
- OpenGL 4.1 and 4.2
  - Add geometry shaders and tessellator

What About Direct X?

- Windows only
- Advantages
  - Better control of resources
  - Access to high level functionality
- Disadvantages
  - New versions not backward compatible
  - Windows only
- Recent advances in shaders are leading to convergence with OpenGL

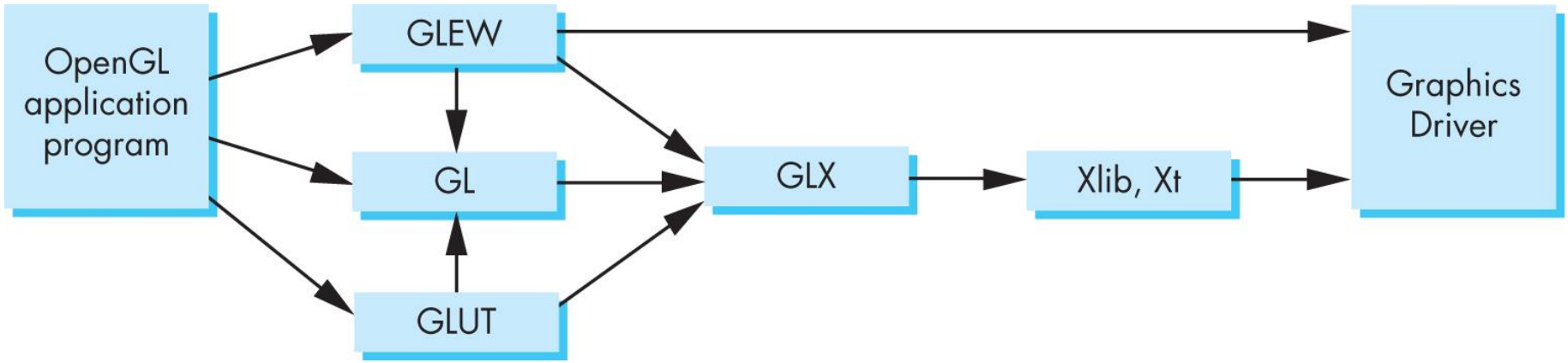
OpenGL Libraries

- OpenGL core library
  - OpenGL32 on Windows
  - GL on most unix/linux systems (libGL.a)
- OpenGL Utility Library (GLU)
  - Provides functionality in OpenGL core but avoids having to rewrite code
  - Will only work with legacy code
- Links with window system
  - GLX for X window systems
  - WGL for Windows
  - AGL for Macintosh

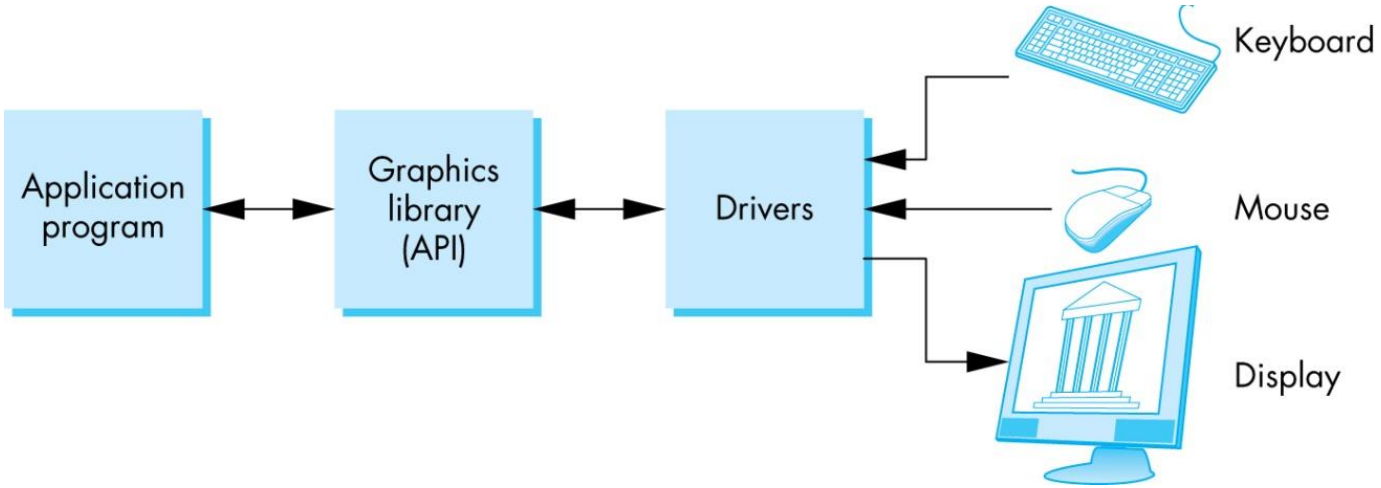
GLEW

- OpenGL Extension Wrangler Library
- Makes it easy to access OpenGL extensions available on a particular system
- Avoids having to have specific entry points in Windows code
- Application needs only to include glew.h and run a glewInit()

Software Organization



OpenGL Architecture



OpenGL Functions

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

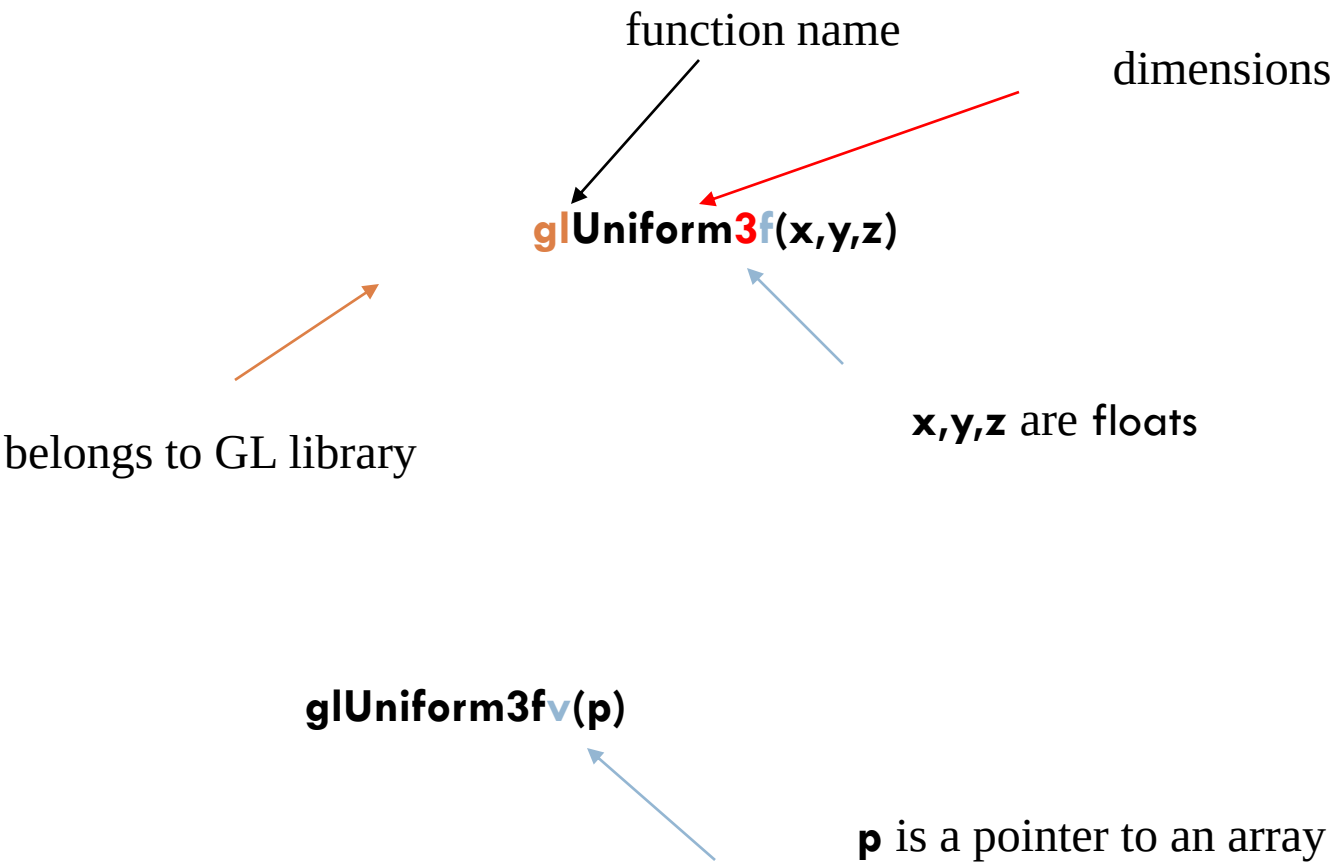
OpenGL State

- OpenGL is a state machine
- OpenGL functions are of two types
  - Primitive generating
    - Can cause output if primitive is visible
    - How vertices are processed and appearance of primitive are controlled by the state
  - State changing
    - Transformation functions
    - Attribute functions
    - Under 3.1 most state variables are defined by the application and sent to the shaders

Lack of Object Orientation

- OpenGL is not object oriented so that there are multiple functions for a given logical function
  - `glUniform3f`
  - `glUniform2i`
  - `glUniform3dv`
- Underlying storage mode is the same
- Easy to create overloaded functions in C++ but issue is efficiency

OpenGL function format



OpenGL #defines

- Most constants are defined in the include files **`gl.h`**, **`glu.h`** and **`glut.h`**
  - Note **`#include <GL/glut.h>`** should automatically include the others
  - Examples
  - **`glEnable(GL_DEPTH_TEST)`**
  - **`glClear(GL_COLOR_BUFFER_BIT)`**
- include files also define OpenGL data types: **`GLfloat`**, **`GLdouble`**,….

OpenGL and GLSL

- Shader based OpenGL is based less on a state machine model than a data flow model
- Most state variables, attributes and related pre 3.1 OpenGL functions have been deprecated
- Action happens in shaders
- Job is application is to get data to GPU

GLSL

- OpenGL Shading Language
- C-like with
  - Matrix and vector types (2, 3, 4 dimensional)
  - Overloaded operators
  - C++ like constructors
- Similar to Nvidia's Cg and Microsoft HLSL
- Code sent to shaders as source code
- New OpenGL functions to compile, link and get information to shaders

OpenGL

- OpenGL basic library – most commands
- OpenGL Utility (GLU) – some high level commands
  - e.g. generating complex objects, cylinder, sphere
- OpenGL Utility Toolkit (GLUT)
  - windowing commands
  - We use Freeglut instead
    - A completely opensourced alternative
    - Still under maintenance
    - Codes are fully compatible with GLUT

Commands

- Commands are prefixed by gl
  - glBegin, glClear, glCopyPixels
- Constants are Uppercase seperated by “_”
  - GL_POLYGON
- Datatypes
  - GLbyte, GLshort, GLint, …

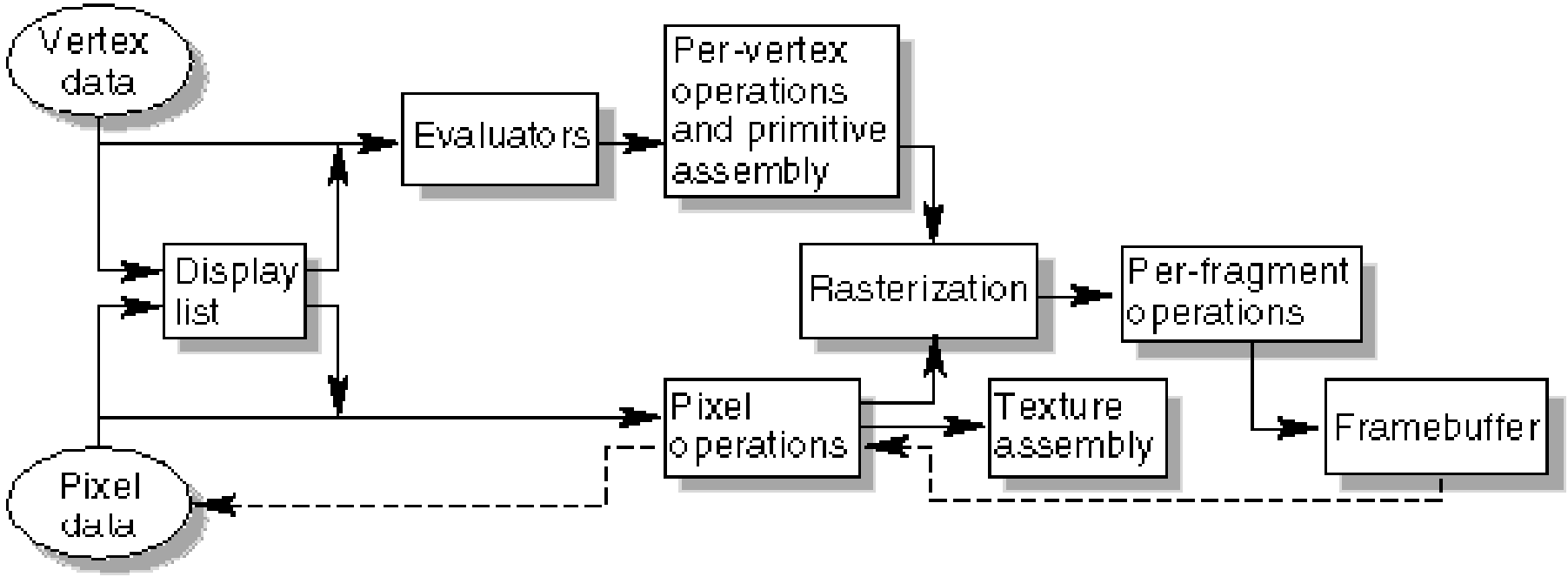
OpenGL as a State Machine

- Maintain pipeline states: current color, viewing and projection transformation, polygon drawing modes, etc.

```
glColor3f(1.0f, 0.0f, 0.0f); // <- alter current state

// Draw a cube. What color is the cube?

…

glColor3f(0.0, 1.0f, 0.0f); // <- alter current state

// Draw a sphere. What color is the sphere?

…
```

- Hierarchical state maintenance
  - glPushAttrib(), glPushMatrix(), glPopAttrib(), glPopMatrix(), …

OpenGL Rendering Pipeline



OpenGL Libraries

- OpenGL core library (GL)
  - *#include <GL/gl.h>*
  - OpenGL32 on Windows
  - GL on most UNIX / LINUX systems (libGL.a)
- OpenGL Utility Library (GLU)
  - *#include <GL/glu.h>*
  - Provides functionality in OpenGL core but avoids having to rewrite code
- OpenGL Utility Toolkit (GLUT or freeglut)
  - *#include <GL/glut.h>*
  - Provides functionality common to all window systems
    - Open a window
    - Get input from mouse and keyboard
    - Menus
    - Event-driven
  - Code is portable but GLUT/freeglut lacks the functionality of a good toolkit for a specific platform

GLUT/freeglut: OpenGL Utility Toolkit

- A simplified window system toolkit
  - Create windows
  - Initialize display context
  - Handle basic application events: screen refreshing, timer, resize…
  - Handle user inputs
- A higher level geometric toolkit
  - *glutSolidSphere, glutSolidCube,…*
- Application Structure
  - Configure and open window
  - Initialize OpenGL state
  - Register input callback functions
    - render
    - resize
    - input: keyboard, mouse, etc.
  - Enter event processing loop

Prerequisites for GLUT/freeglut

- *Glut.h* – in your include path
- Freeglut.h, freeglut_std.h, freeglut_ext.h for freeglut (Just include glut.h. You don't need to include other headers in your code unless you want to use some functions not supported by GLUT. Just copy the files in your include path)
- *Glut32.lib/freeglut.lib* (for windows) in your library path
- *Glut32.dll/freeglut.dll* in system path
- For OpenGL calls
  - *Gl.h* and *glu.h* (*included by glut.h*)
  - *Opengl32.lib* and *glu32.lib*
  - *Opengl32.dll* and *glu32.dll*

How to Set up in Visual Studio

- Copy dlls to
  - Project directory or
  - Windows system32
- Copy directory <GL>
  - /directory/to/Visual/Studio/VC/include
  - (C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Include)
- Copy library glut32.lib
  - Project directory or
  - /directory/to/Visual/Studio/VC/lib
  - (C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Lib)

GLUT Window Management

- *glutInit (&argc, argv)*
  - Glut initialization
- *glutCreateWindow ("OpenGL Example")*
  - Create a display window with a title
- *glutDisplayFunc (myDisplay)*
  - Specify what the display window is to contain
- *glutMainLoop ()*
  - Activate the display window and its graphic content
- *glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | …)*
  - Set other initial options for the display window
- *glutInitWindowPosition (int x, int y) & glutInitWindowSize (int width, int height)*
  - An Initial display window location (top-left corner) and size

main() Function


```
int main(int argc, char** argv)
{
  // glut init
        glutInit(&argc, argv);
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  // actual window size
         glutInitWindowSize(500,500);
  // initial window location, top-left corner
         glutInitWindowPosition(0,0);
  // create window with title “simple”
         glutCreateWindow("simple");
  // call mydisplay() function
         glutDisplayFunc(mydisplay);
  // call init() function
         init();
   // main event loop, do not use exit()
         glutMainLoop();
}
```

Init() Function

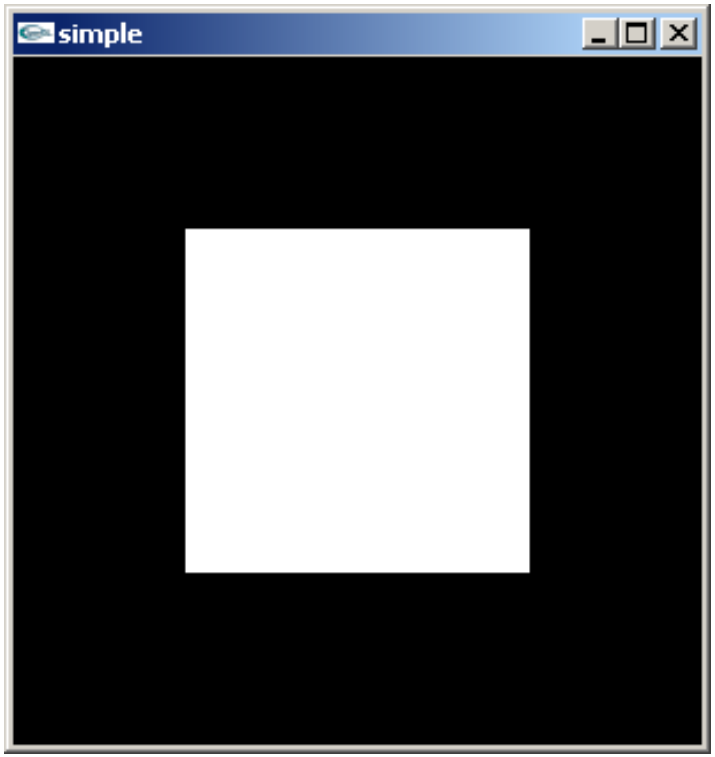

```
void init()
{
    glClearColor (0.0, 0.0, 0.0, 1.0);          // black clear color, opaque window

    glColor3f(1.0, 1.0, 1.0);                   // white

    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();

    glOrtho2D(-1.0, 1.0, -1.0, 1.0);            // screen size (-1.0,-1.0) to (1.0,1.0)
}
```

display() Function


```
void mydisplay()
{
    glClear(GL_COLOR_BUFFER_BIT);          // clear the window
    glBegin(GL_POLYGON);                   // fill connected polygon
        glVertex2f(-0.5, -0.5);            // vertices of the square
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
    glEnd();
}
```

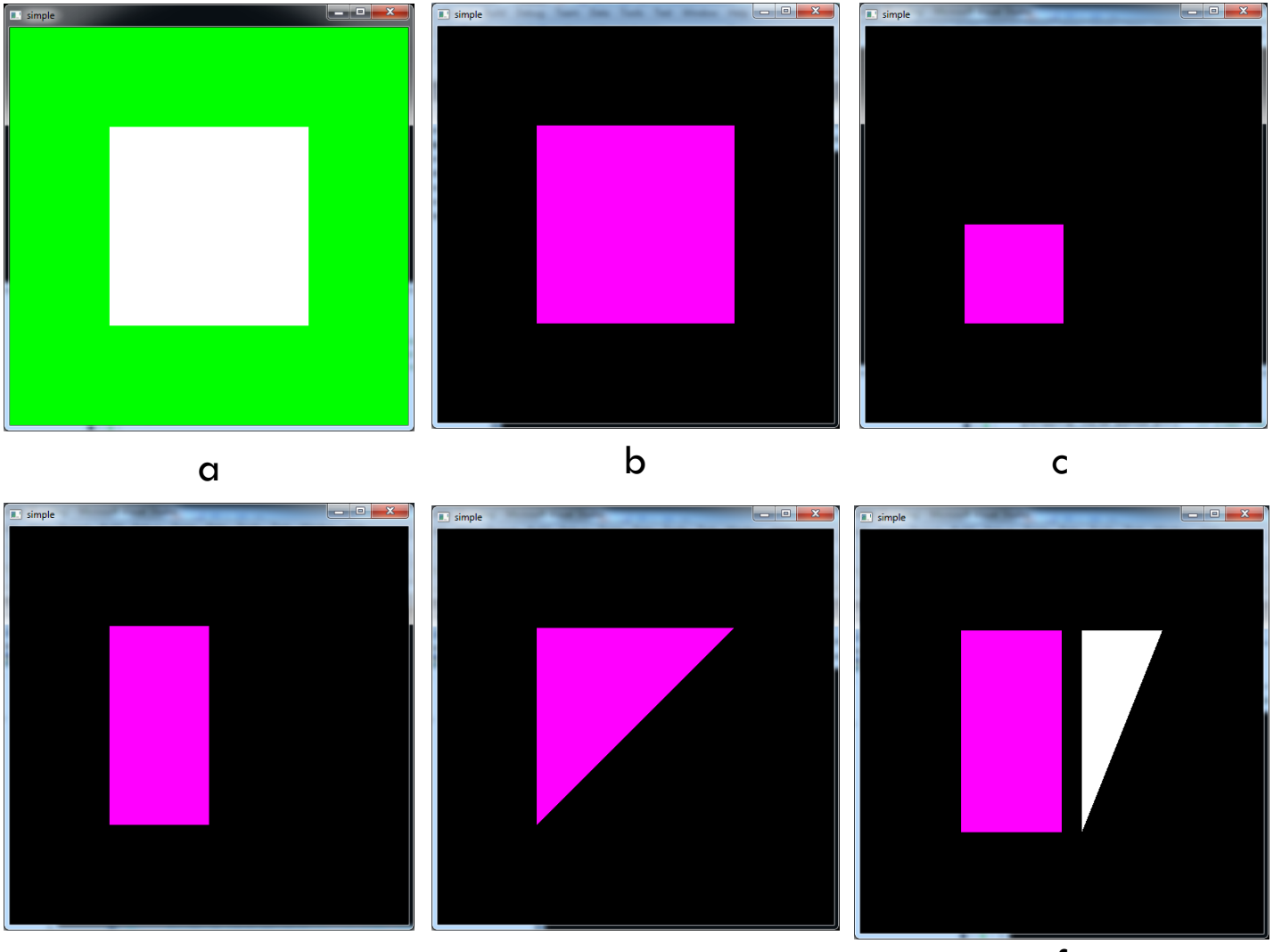
A Simple Program (?)

Generate a square on a solid background



Is This Working??

Exercise



Callbacks

- Callback functions refresh
  - *void (*func)(void)* – what is this thing?
- Virtually all interactive graphics programs are event driven
- Glut/freeglut uses callbacks to handle events
  - Windows system invokes a particular procedure when an event of particular type occurs.
  - MOST IMPORTANT: display event
    - Signaled when window first displays and whenever portions of the window reveals from blocking window
    - *glutDisplayFunc(void (*func)(void))* registers the display callback function

GLUT/freeglut Callbacks Overview

- *glutDisaplyFunc(void (*func)(void))*

  whenever GLUT/freeglut decides to redisplay the window, the registerd callback is executed.

- *glutReshapeFunc(void (*func)(int w, int h))*

  indicates what action should be taken when the window is resized.

- *glutKeyboardFunc(void (*func)(unsigned char key, int x, int y))*
  *glutMouseFunc(void (*func)(int button, int state, int x, int y))*
  *glutSpecialFunc(void (*func)(unsigned char key, int x, int y))*

  allow you to link a keyboard key or a mouse button with a routine that's invoked when the key or mouse button is pressed or released.

- *glutMotionFunc(void (*func)(int x, int y))*

  registers a routine to call back when the mouse is moved while a mouse button is also pressed.

- *glutIdleFunc(void (*func)(void))*

  registers a function that's to be executed if no other events are pending – use for animation or continuous update

Display Callback Code Example

glutDisplayFunc(display);

```
void display( void )
{
  ... ... ...
  glBegin( GL_LINE_LOOP );
    glVertex3fv( v[0] );
    glVertex3fv( v[1] );
    glVertex3fv( v[2] );
    glVertex3fv( v[3] );
  glEnd();
  glutSwapBuffers();  // if double buffers used
}
```

Keyboard Callback Code Example


```
glutKeyboardFunc( keyboard );
void keyboard(unsigned char key, int x, int y )
{
  switch( key ) {
    case ‘q’ : case ‘Q’ :
      exit(0);                              // exit the program
      break;


    case ‘r’ : case ‘R’ :
      rotate += delta;
            glutPostRedisplay();            // update the display
      break;
  }
}
```

Special Key Callback Code Example


```
glutSpecialFunc( specialKeys );
void specialKeys(unsigned char key, int x, int y )
{
  switch( key ) {

    case GLUT_KEY_F1:                    // F1 function key
      red   = 1.0;
      green = 0.0;
      blue  = 0.0;
      break;


    case GLUT_KEY_UP:                    // up function key
      movement = GL_TRUE;
      break;
  }
 glutPostRedisplay();
}
```

Special Keys

GLUT_KEY_F1	F1 function key
… … …	
GLUT_KEY_F12	F12 function key
GLUT_KEY_LEFT	Left Arrow Key
GLUT_KEY_RIGHT	Right Arrow Key
GLUT_KEY_UP	Up arrow key
GLUT_KEY_DOWN	Down arrow key
GLUT_KEY_PAGE_UP	PgUp key
GLUT_KEY_PAGE_DOWN	PgDn key
GLUT_KEY_HOME	Home key
GLUT_KEY_END	End key
GLUT_KEY_INSERT	Insert key

Other Special Keys

- int glutGetModifiers(void)

  to detect if any modifier key is pressed

  glutKeyboardFunc(*modifierKeys*);

```
void modifierKeys(unsigned char key, int x, int y )
{
  if (key == 'r')  {
      int mod = glutGetModifiers();

      if(mod == GLUT_ACTIVE_ALT) // if ALT key
         red = 0.0;
      else
         red = 1.0;
   }

  … …
}
```

  - **GLUT_ACTIVE_SHIFT** SHIFT key
  - **GLUT_ACTIVE_CTRL** CTRL key
  - **GLUT_ACTIVE_ALT** ALT key

Mouse Function Callbacks

glutMouseFunc(*mouseMovement*);

```
void mouseMovement(int button,int state,int x,int y)
{
// button: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON,
          GLUT_RIGHT_BUTTON
// state:  GLUT_DOWN, GLUT_UP

  if (button == GLUT_LEFT_BUTTON &&
      state == GLUT_DOWN)
  {
     startMovement = GL_TRUE;
     // do something
  }

  mouseCurPositionX = x;   // record mouse position
  mouseCurPositionY = y;
  mouseCurButton    = button;
}
```

Mouse Motion Callback

glutMotionFunc(mouseMotion);

```
void mouseMotion(int x, int y)
{
  if(mouseCurButton == GLUT_LEFT_BUTTON) {
    x_angle += 360.0*(x-mouseCurPositionX)/width;
    y_angle += 360.0*(y-mouseCurPositionY)/height;
   }
   if(mouseCurButton == GLUT_RIGHT_BUTTON)
     scale += (y-mouseCurPositionY)/100.0;
   mouseCurPositionX = x;
   mouseCurPositionY = y;

   glutPostReDisplay();
}
```

In display( ) we have …. ….. ….

```
      glScalef (scale, scale, scale);
      glRotatef (x_angle, 1.0f, 0.0f, 0.0f);
      glRotatef (y_angle, 0.0f, 1.0f, 0.0f);   … …. ….
```

Mouse Passive Motion Callback

- g*lutPassiveMotionFunc(mousePMotion);*
- Almost as same function as the *glutMotionFunc();*
- The (active) motion callback is called when the mouse moves within the window while one or more mouse buttons are pressed.
- The (active) motion callback is called when the mouse moves within the window while one or more mouse buttons are pressed.

Double Buffering

- Double buffering is necessary for almost all OpenGL applications:
  - Render into back buffer
  - Swap buffers when finished rendering a frame: The old back buffer becomes the front buffer that is displayed. The old front buffer becomes the back buffer that is rendered into.
- What happens when you do not use double buffering?
  - flickering artifacts, tearing artifacts
- Some commands:
  - glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  - glutSwapBuffers();

Readings

- OpenGL command syntax
  - Read Red Book(pdf): ch1
- Understand OpenGL's state machine model
  - Red Book(pdf):ch1 p 8
- Understand OpenGL's client server model
  - An example, glFlush, Red Book(pdf): ch2 p26-27
  - Think about the differences between *glFlush, glFinish, glutSwapBuffers*. To better understand *glFinish*, think if you want to measure the exact rendering time of one frame on GPU.
- Animation & Double buffering
  - Red Book(pdf): ch1 p. 18, explains double buffering and glutSwapBuffers
- GLUT/freeglut (important!!!)(Everything in the book is compatible for freeglut)
  - Red Book(pdf): ch1 p11 - 19
  - Red Book Appendix D (p433 – 436)
  - Example 1-2, Example 1-3 (Animation)
  - *Glut.h* will show you the technical details

Questions?

- Ask now or e-mail later
- Acknowledgements
  - Previous instructors at Purdue
    - David Ebert, ECE
    - Niklas Elmqvist, ECE
  - Previous instructors at Arizona state university
    - Ross Maciejewski
  - Textbook (Ed Angel)
  - Google Image Search
    - Copyright respective owners