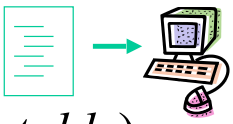
CS770/870 Spring 2017 Introduction:
Computer Graphics with OpenGL

- Setting up your first window with GLFW
- Drawing your first OpenGL primitives
- Changing sizes, color, etc. (OpenGL state)
- An object-oriented approach

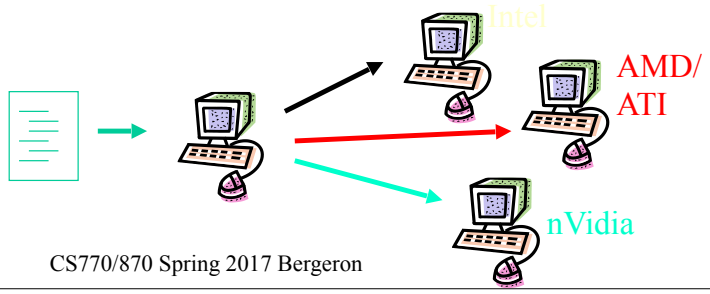
The Problem

- How do you control the contents of a computer screen from your program?
  - What accelerator hardware is present?
  - What format do you send a frame (picture) in?
  - How do you synchronize with display hardware to show a frame exactly when you want?
  - How do you get keyboard and mouse input from the user in an interactive application?

Some Solutions



- Write **for** specific hardware (*not portable*)
- Write for a model of the hardware, create drivers to map the model to the hardware
  - DirectX (*Windows-centric*)
  - Java (*cross-platform*)
  - OpenGL (*cross-platform, supports 3D well*)
  - Others…



OpenGL and GLFW

- OpenGL talks to the graphics hardware
  - Frame buffer to construct image in right format
  - Synchronization of frames for display
  - Provides programming model that allows underlying drivers to exploit hardware acceleration
- GLFW does same for talking to OS
  - Creating/resizing windows
  - Accepting basic user input (mouse, keyboard)

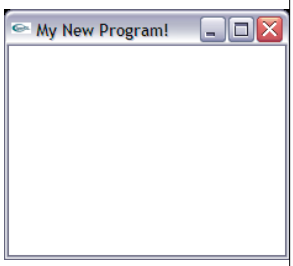
GLEW: GL Execution Wrangler

- OpenGL/GLSL version proliferation is a nightmare for writing portable efficient code
- GLEW is one effort to help applications query the functionality of a given configuration. This is NOT a priority for this course.
- However, GLEW does some other things that facilitate portability. So, all C++ programs will need to include GLEW and invoke *glewInit*.

glew.sourceforge.net

A GLFW/OpenGL C++ Program

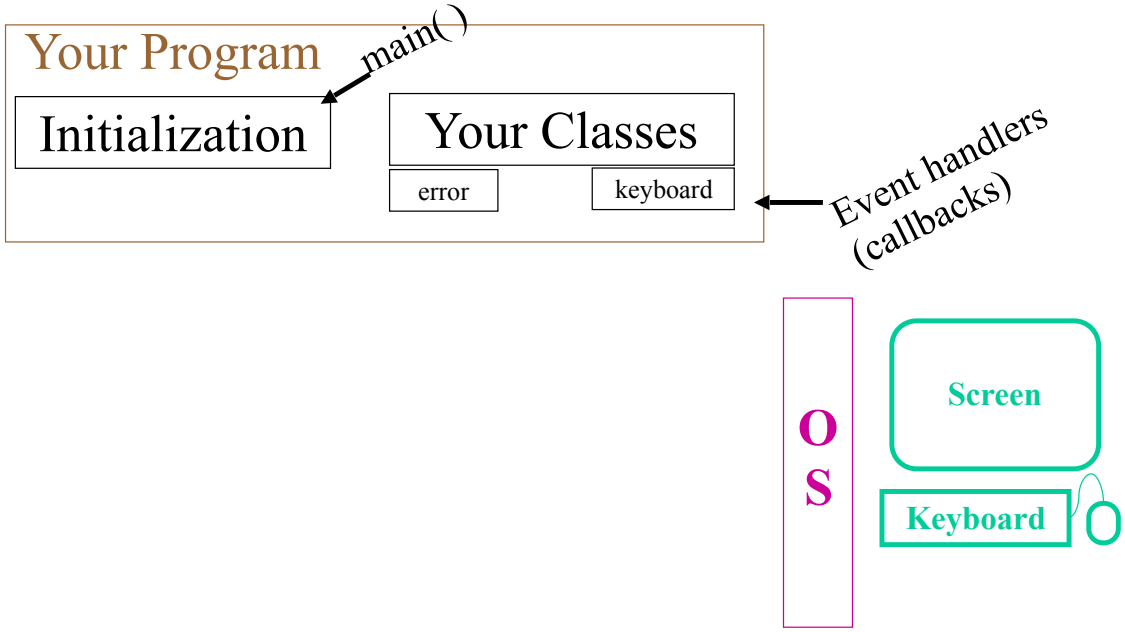
```
int main( int argc, char **argv )
{
   glfwInit();  // initialize toolkit
   glfwWindowHint( . . .); // multiple
   glutInitDisplayMode( GLUT_SINGLE | GLUT_RGB );
   w = glfwCreateWindow( w, h, "title", . . .);
   // register callback functions
   glfwSetKeyCallback( w, callbackFunction );
   glClearColor( 0.0, 0.0, 0.0, 1.0 ); // black
   while ( !finished )
   {
      redraw();  // recreate scene
      glfwSwapBuffers( w );
      glfwPollEvents(); // or glfwWaitEvents

   }
   . . .
}
```



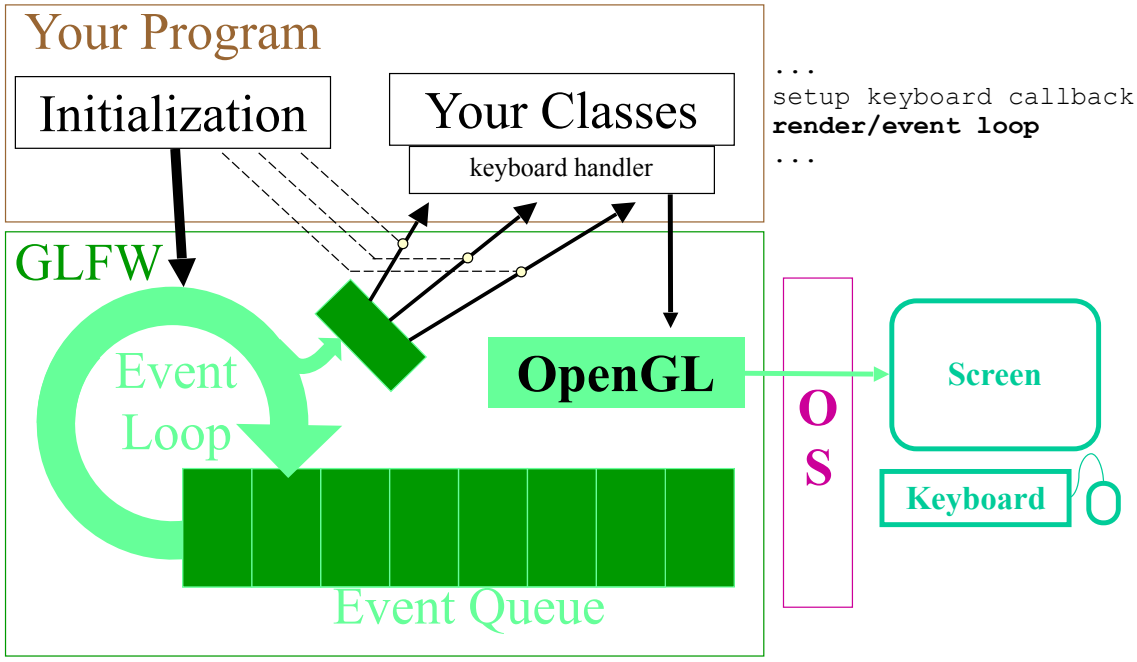
LWJGL for CS770/870

- LWJGL - LightWeight Java Game Library
  - available at www.lwjgl.org
  - not complete, but good enough for us
  - intended to be more efficient
- JOML - Java OpenGL Math Library
- GLFW - Java interface
- CS770/870 Package (on course web site)
  - lwjgl770.jar
  - native interface libraries for Mac, Linux, Windows

A GLFW/OpenGL Program

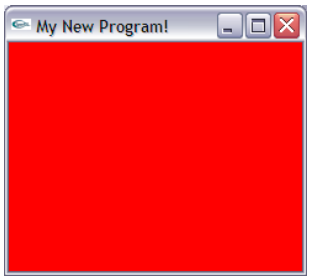


A GLUT/OpenGL Program

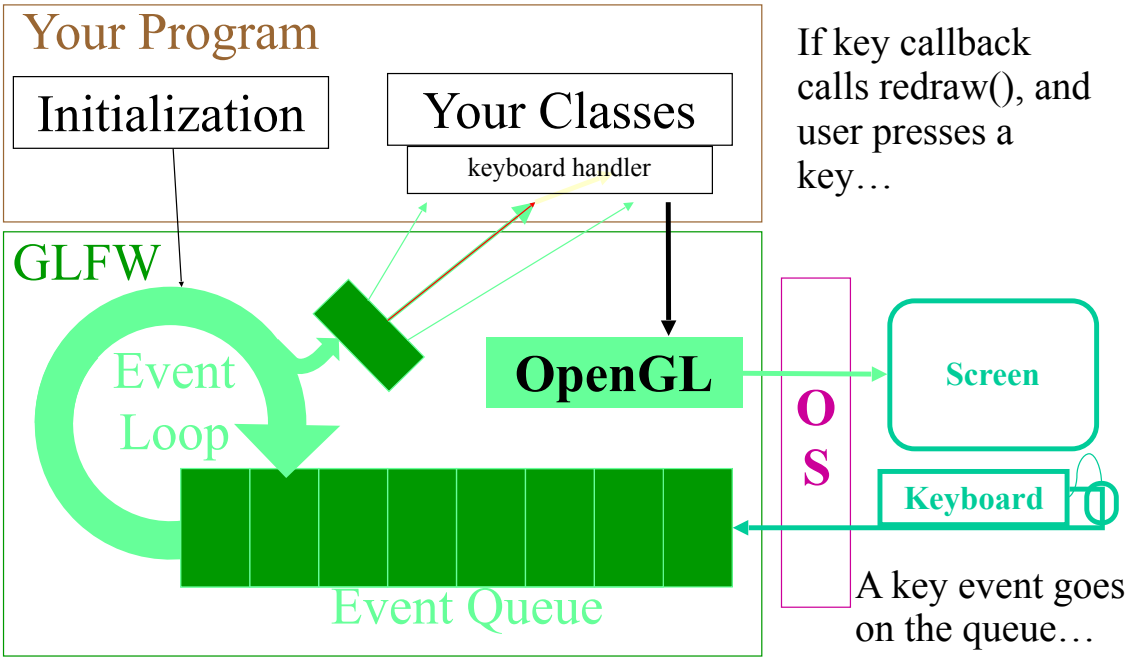


A GLFW/OpenGL Program

```
void redraw( void )
{
  glClear( GL_COLOR_BUFFER_BIT ); // clear screen
  glFlush();          // send all output to display
}

void key_callback( . . . )
{
  // set the bg color to a vibrant red
  glClearColor( 1.0, 0.0, 0.0, 1.0 );
  redraw();
}
```



A GLFW/OpenGL Program



OpenGL Coordinate System

- Default:
  - Center at (0.0, 0.0)
  - Lower-left: (-1.0, -1.0); upper-right: (1.0, 1.0)
- Some initialization magic *(explained later…)*
  - Lower-left at (0, 0)
  - Upper-right at (windowWidth, windowHeight)

Learning OpenGL and GLFW

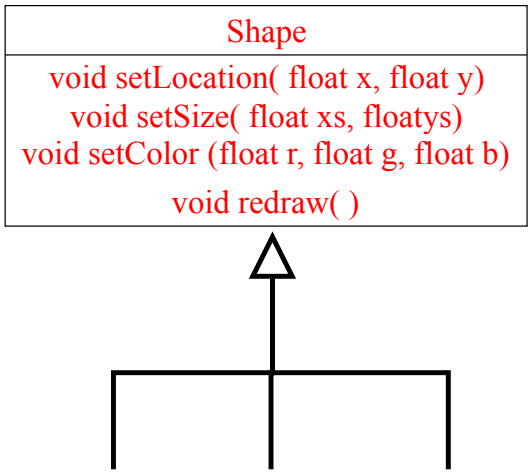
- Great resources: online manuals
  - OpenGL
    http://www.opengl.org/sdk/docs
  - GLFW
    http://www.glfw.org/documentation.html
- What they give you:
  - More argument constants, options, explanations
  - Additional commands

An Object-Oriented Approach

- Want to be able to have objects draw themselves: houses, teapots, alien spacecraft
- OpenGL only provides primitive operations
  - Draw point
  - Draw line
  - Draw polygon
  - Others we'll see later
- What's a good design?

What's a Good O-O Design?

- A class of objects that
  - Keeps own state: position, size, color, …
  - Can draw itself
  - and more …
- A mechanism for drawing all these shapes from the redraw callback in `main( )`



What does it look like in C++?

- Base class declaration (in header file)

See basicDemo Source for full implementation

```
class Shape
{
  public:
    Shape();
    virtual ~Shape();
    void setLocation( float x, float y );
    void setSize( float xs, float ys );
    void setColor( float r, float g, float b );
    virtual void redraw() = 0;
  protected:
    float xLoc, yLoc; // location of the object
    float xSize, ySize; // size of the object
    float red, green, blue; // color
};
```

What does it Look Like in C++?

- One way to create a list of Shapes

```
#include <vector>
...
std::vector<Shape> shapes;
...
std::vector<Shape*>::iterator it;
for (it = shapes.begin(); it < shapes.end(); it++)
  (*it).redraw();
```

See *basicDemo* Source for full implementation

Quick *make* Intro

- *make* is an old, simple, idiosyncratic tool for building software applications
- Variables

```
GL_INC = -I/usr/X11/include/GL
CPPFLAGS = -Wall $(X_INC) $(GL_INC)
```

- Dependencies

```
compile:  $(OBJS)
%.o: %.cpp %.h
```

- Actions to do when dependency satisfied

```
$(CCC) -c $(CPPFLAGS $*.cpp
```

- Should not need to make any changes to supplied *Makefile*s

Review

- OpenGL one solution to drawing problem
- Created first simple OpenGL/GLUT app
- OpenGL state
- OpenGL coordinates
- An object-oriented approach

Next

- GLSL - OpenGL Shader Language