CS 543 - Computer Graphics: Intro to OpenGL

by

Robert W. Lindeman

gogo@wpi.edu

(with help from Emmanuel Agu ;-)

OpenGL Basics

- Last time:
  - What is *Computer Graphics*?
  - What is a *graphics library*
  - What to expect from class
  - Intro stuff….
- Today:
  - Start learning OpenGL basics
  - OpenGL program structure
  - "Hello, world!" skeleton
- Note: Only basics here!
- Learn a bunch on your own (more fun anyway!)

OpenGL Basics

- Primary goal: rendering
- Rendering?
  - Convert geometric/mathematical object and environment descriptions into images
- OpenGL can render:
  - Geometric primitives (lines, dots, *etc*.)
  - Bitmap images (.bmp, .jpg, *etc*.)
- OpenGL does not do window, mouse, keyboard, or device management

OpenGL Basics

- Low-level graphics rendering Application Programming Interface (API)
- Widely used all over the graphics field
  - Will be used in this class
- Highly portable
  - Display device independent
  - Window system independent (X Windows, Aqua, Windows, *etc*.)
  - OS independent (Unix, OS X, Linux, Windows, *etc.*)
- OpenGL programs behave same on different output devices and OSs
- Event-driven approach

OpenGL: Event-Driven Programming

- Very important programming model!
- Program only responds to events
  - Do nothing until event occurs
  - Sample Events:
    - mouse click, key stroke, window resize
- Programmer defines
  - Events of interest
  - Actions to take upon event occurrence
- System
  - Maintains an event queue
  - Dispatches events to programmer-defined code

OpenGL: Event-Driven vs. Sequential Programming

- Sequential program
  - Read some data
  - Do some processing
  - Print some results
- Event-driven program
  - Initialize some things
  - Enter an infinite loop
    - Wait until defined event occurs
    - Take defined action
- What is the world's most widely used event-driven program?

OpenGL: Event-Driven Programming in OpenGL

- How in OpenGL?
  - Programmer registers callback functions
  - Callback function called when associated event occurs
- Example:
  - Declare a function myMouse to respond to mouse click
  - Register it: Tell OpenGL to call it when mouse clicked
  - Code? glutMouseFunc(myMouse);
    - Notice this is not an OpenGL function!

OpenGL Utility Toolkit (GLUT)

- OpenGL
  - Window system independent
  - Concerned only with drawing
  - No window-management functions (create, resize, *etc*.)
  - Very portable
- GLUT:
  - Minimal window management: fast prototyping
  - Interfaces with different windowing systems
  - Allows easy porting between windowing systems
  - By far the most-widely used library for OpenGL
- GLUI:
  - More-advanced GUI widgets

OpenGL Utility Toolkit (GLUT)

- No bells nor whistles
  - No sliders
  - No dialog boxes
  - No menu bar, *etc.*
- To add bells and whistles, need other API:
  - GLUI
  - Qt
  - X window system
  - Microsoft: WGL, *etc*.

Getting Started with First OpenGL Program

- At top of program, include required headers

```
#include <gl/gl.h>
#include <gl/glu.h>
```

- **gl** directory is sub-directory of your include file location
- Then include GLUT for window management

```
#include <gl/glut.h>
```

Getting Started…

- If you want full-blown, pull-down windows (WGL, AGL, *etc.*)

```
#include <windows.h>  // do this before gl.h, glu.h
```

- Most OpenGL applications use standard C library so

```
#include <stdlib.h>
#include <stdio.h>
```

Program Structure

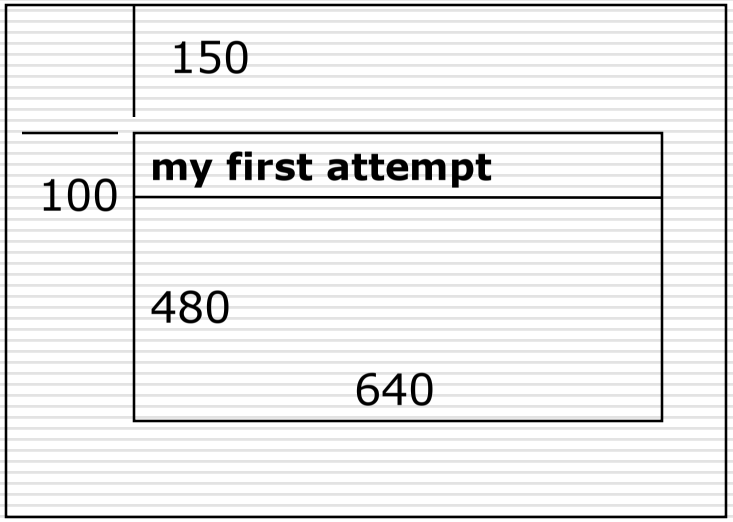
- Configure and open window (GLUT)
- Initialize OpenGL state
- Register input callback functions (GLUT)
  - Render
  - Resize
  - Input: keyboard, mouse, *etc*.
- My initialization
  - Set background color, clear color, drawing color, point size, establish coordinate system, *etc*.
- **`glutMainLoop( );`**
  - Draws image and waits infinitely until action occurs

GLUT: Opening a window

```
glutInit( &argc, argv );
     // initializes

glutInitDisplayMode( GLUT_SINGLE | GLUT_RGB );
     // sets display mode (e.g., single buffer
       with RGB colorspace)

glutInitWindowSize( 640, 480 );
     // sets window size (WxH)

glutInitPosition( 100, 150 );
     // sets upper left corner of window

glutCreateWindow( "my first attempt" );
     // open window with title "my first
       attempt"
```

OpenGL Skeleton

```
int main( int argc, char *argv[] )  {
  // First initialize toolkit, set display mode and create window
  glutInit( &argc, argv );
  glutInitDisplayMode( GLUT_SINGLE | GLUT_RGB );
  glutInitWindowSize( 640, 480 );
  glutInitWindowPosition( 100, 150 );
  glutCreateWindow( "my first attempt" );

  // ...then register callback functions
  // ...do my initialization
  // ...wait in glutMainLoop for events
}
```



GLUT Callback Functions

- Register all events your program will react to
- Event occurs => system generates callback
- Callback: routine system calls when event occurs
- No registered callback = no reaction!
- Example: if you do not define a keyboard callback function, you can bang on keyboard all you want, NO RESPONSE!!
  - Except from your neighbor.

GLUT Callback Functions in Skeleton


```
glutDisplayFunc( myDisplay );
   // called when window contents need to be redrawn

glutReshapeFunc( myReshape );
   // called when window is reshaped

glutMouseFunc( myMouse );
   // called when mouse button is pressed

glutKeyboardFunc( myKeyboard );
   // called when keyboard is pressed or released

glutMainLoop( );
   // Now draw the initial picture and enter infinite
     loop until a registered event occurs
```

Example: Rendering Callback

- Do all your drawing in the display function
  - Called initially & when picture changes (*e.g.*, resize)
- First, register callback in main( ) function

```
glutDisplayFunc( myDisplay );
```

- Then, implement display function

```
void myDisplay( void )  {
  // put drawing stuff here
  ...
  glBegin( GL_LINES );
    glVertex3fv( v[0] );
    glVertex3fv( v[1] );
    ...
  glEnd( );
}
```

OpenGL Skeleton

```
int main( int argc, char *argv[] )  {
  // First initialize toolkit, set display mode and create window
  glutInit( &argc, argv );
  glutInitDisplayMode( GLUT_SINGLE | GLUT_RGB );
  glutInitWindowSize( 640, 480 );
  glutInitWindowPosition( 100, 150 );
  glutCreateWindow( "my first attempt" );

  // Now register callback functions
  glutDisplayFunc( myDisplay );
  glutReshapeFunc( myReshape );
  glutMouseFunc( myMouse );
  glutKeyboardFunc( myKeyboard );

  myInit( );
  glutMainLoop( );
}
```

Final Thoughts

- Need for global variables
  - Callback API is predefined
  - No way to add parameters

References

- Hill, chapter 2