Computer Graphics

P. Healy

CS1-08
Computer Science Bldg.
tel: 202727
patrick.healy@ul.ie

Spring 2021–2022

Outline

Tutes and Labs

- Tute02, vector review (see matrix)
- Week02 lab
- Lab Marking
  - 8 labs in total
  - each lab worth $3\frac{6}{8}$% of overall grade
  - marked on attendance and completion of lab
  - completed lab due one week from when “first released”

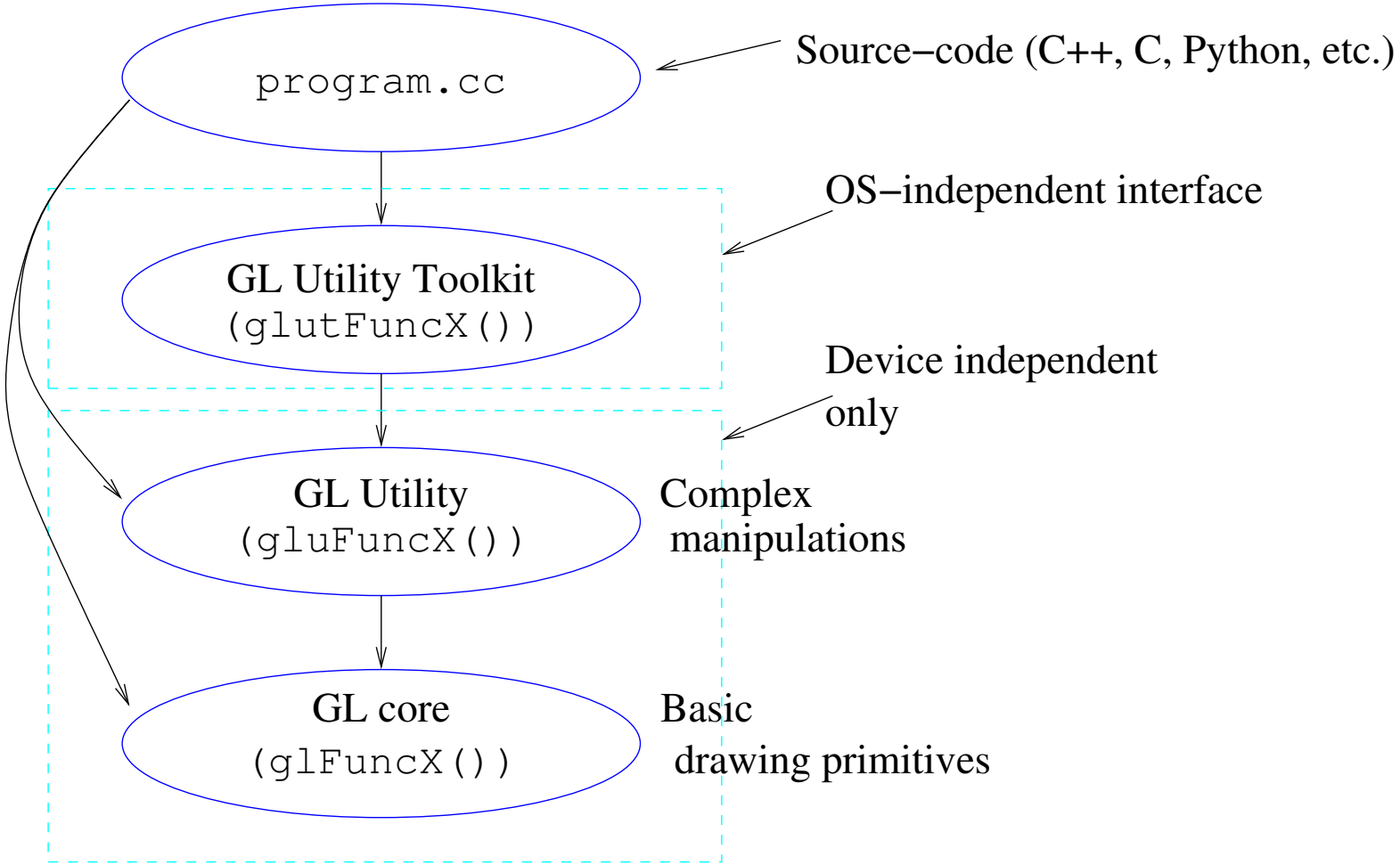
Outline

What is it?

- OpenGL is a completely open graphics library that takes the system-dependency headaches out of graphics programming
- Hardware- and operating system-independent
- Initially developed by SGI (Silicon Graphics)
- Several implementations of OpenGL exist
- Several tutorials, HOWTOs exist
- OpenGL examples site
- Do you know about Khan Academy?

Outline

GL libraries hierarchy



GL libraries hierarchy

Documentation

- OpenGL, `gl`, and OpenGL Utility Library, `glu`, function documentation is here
- OpenGL Utility Toolkit `glut` documentation is here
- We will use the `freeglut` implementation of the OpenGL Utility Toolkit (GLUT) library. home page

Outline

A First Program

For the program that follows (called `try.cc`) the following command compiles:

```
g++ try.cc -l GL -l GLU -l glut -o try
```

A First Program (contd.)

```
int main (int argc, char ** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(50, 100);
  glutInitWindowSize(400, 300);
  glutCreateWindow("A Baby GL example");

  init();  // we will write this

  glutDisplayFunc(lineSegmentGreen);  // and this
  glutMainLoop();
}
```

A First Program (contd.)

```
int main (int argc, char ** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(50, 100);
  glutInitWindowSize(400, 300);
  glutCreateWindow("A Baby GL example");

  init();  // we will write this

  glutDisplayFunc(lineSegmentGreen);  // and this
  glutMainLoop();
}
```

A First Program (contd.)

```
int main (int argc, char ** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(50, 100);
  glutInitWindowSize(400, 300);
  glutCreateWindow("A Baby GL example");

  init();  // we will write this

  glutDisplayFunc(lineSegmentGreen);  // and this
  glutMainLoop();
}
```

Top part of program: `init()`

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

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

  glMatrixMode(GL_PROJECTION);  // ignore for now
  gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}
```

Explanation

- `glClearColor(1.0, 1.0, 1.0, 0.0);` sets [redacted] to white (R=G=B=1.0), with colour *blending* set to 0.0 (transparent)
- `glMatrixMode(GL_PROJECTION);` use the matrix stack concerned with projecting images onto screen
- `gluOrtho2D(0.0, 200.0, 0.0, 150.0);` window's range of coordinates is (0,0) – (200,150) units; but **dimensions** of window are 400 x 300 pixels
- each *x*-, *y*-unit is 2 pixels in this case

Final part of program: `lineSegmentGreen()`

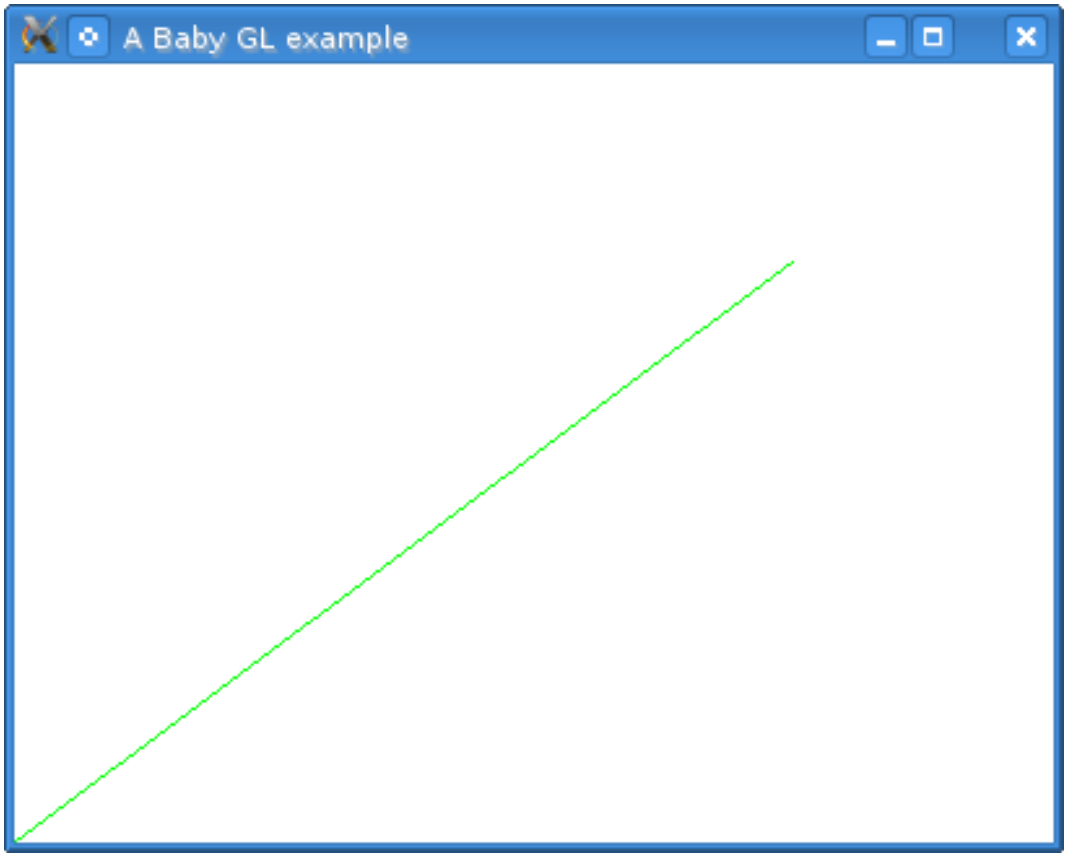
```
void lineSegmentGreen (void)
{
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(0.0, 1.0, 0.0);
  glBegin(GL_LINES);
    glVertex2i(0, 0);      // bottom-left of window
    glVertex2i(150, 112); // (3/4 width, 3/4 height
  glEnd();

  glFlush();
}
```

Result (I)

Resulting picture:



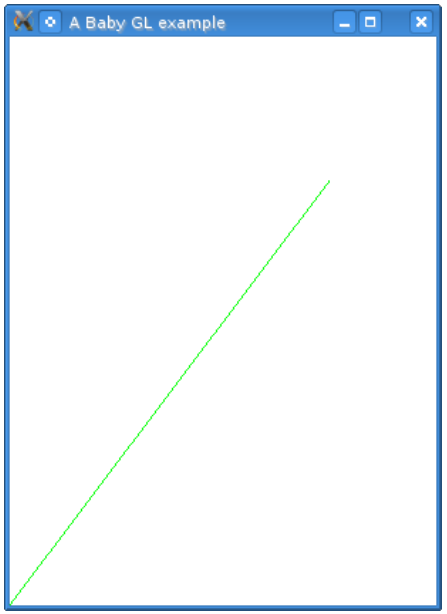
Note the location of $(0, 0)$ and $(150, 112)$

Explanation

- `glutDisplayFunc(lineSegmentGreen);` fn. to call whenever screen needs to be redisplayed
- `glutMainLoop();` loops forever "listening" for interesting events
- `GLUT_SINGLE` is a defined constant (see `glut.h`)
- `glClear(GL_COLOR_BUFFER_BIT);` clears color buffer by coloring all pixels with color of `glClearColor`
- `glColor3f(0.0, 1.0, 0.0);` sets "pen" colour
-
```
glBegin(GL_LINES);
glVertex2i(0, 0);
glVertex2i(150, 112);
glEnd();
```
- `glFlush();` draw the picture **now**

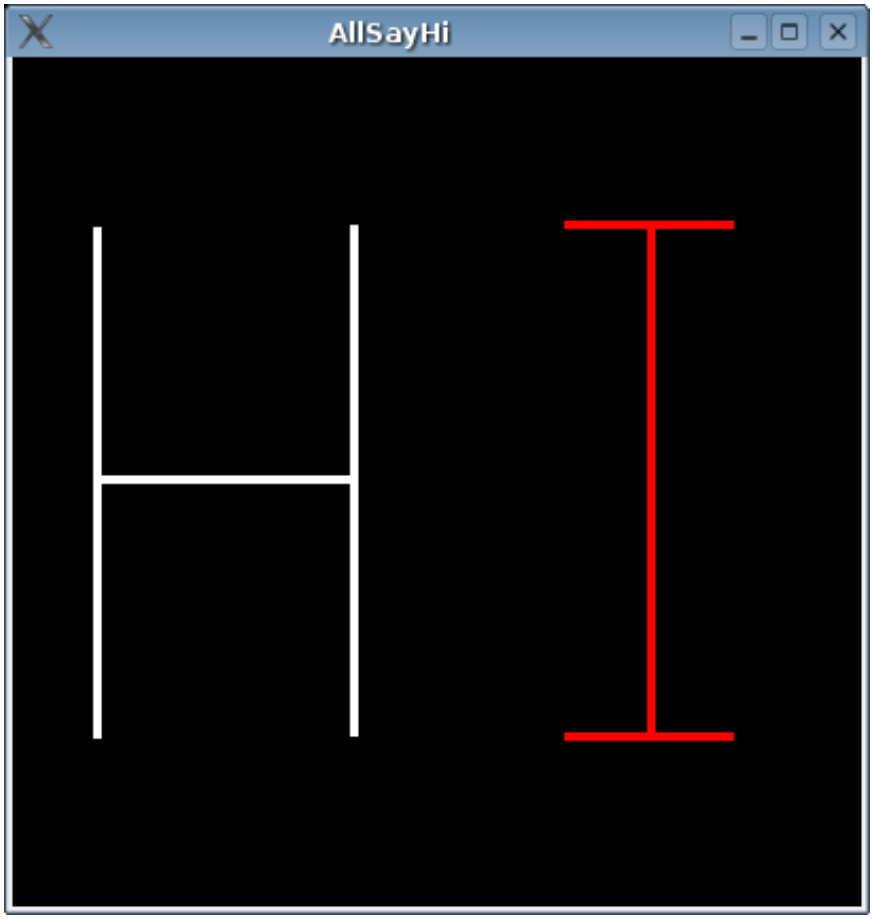
Result (II)

- This is what happens if we change the window's dimensions (**graphic shrunk**)
- That is, we change:
  `glutInitWindowSize(400, 300);`
  to
  `glutInitWindowSize(300, 400);`
  and nothing else
- Note how the re-scaling of x- and y-coordinates to fit the new window dimensions was done automatically
- x-, y-units are no longer the same in pixels: x-unit=300/200; y-unit=



Outline

Program to draw lines of "HI"



- Default coordinates go from -1.0 to 1.0 in both X and Y
- That is, lower left corner is (-1.0, -1.0) and upper right corner is (1.0, 1.0)
- The X axis is horizontal and the Y axis is vertical.
- `gcc hi.c -l GL -l GLU -l glut -o AllSayHi`
- Program will respond to `'q'` from

main()

```
int main (int argc, char** argv)
{
  glutInit(&argc, argv);          // initialize glut
  glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
  glutInitWindowSize(400,400); // window width, hei
  glutInitWindowPosition(0,0); // x, y of top left
  glutCreateWindow(argv[0]); // create titled windo
  glutDisplayFunc(display);  // display callback
  glutKeyboardFunc(test4quit); // keyboard callback
  glutMainLoop();              // hand over control

  return 0;
}
```

test4quit()

```
// input keyboard function to quit with q or Q.
// this is called every time kbd key pressed
// must accept three args, even if mouse loc is
//  not used
// See here
void test4quit(unsigned char key, int x, int y)
{
  if (key == 'q' || key == 'Q')
    exit(0);
}
```

Callback function `display()`

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

void display()   // display callback function
{
  glClear(GL_COLOR_BUFFER_BIT); //  now look here

  glLineWidth(4.0);   // set line width, in pixels

  // default view coordinates go from -1.0
  //  to 1.0 in both X and Y.
```

display() (contd.)

```
// Draw H, comprising three lines
glBegin(GL_LINES); // begin a geometric primitive
  glColor3f(1.0, 1.0, 1.0); // set color to white
  glVertex2f(-0.8, -0.6); // left
  glVertex2f(-0.8, 0.6); //   upright
  glVertex2f(-0.2, 0.6); // right
  glVertex2f(-0.2, -0.6);//   upright
  glVertex2f(-0.8, 0.0); // crossbar: start...
  glVertex2f(-0.2, 0.0); // ... and end
glEnd();
```

display() (contd.)

```
// Draw H, comprising three lines
glBegin(GL_LINES); // begin a geometric primitive
  glColor3f(1.0, 1.0, 1.0); // set color to white
  glVertex2f(-0.8, -0.6); // left
  glVertex2f(-0.8, 0.6); //   upright
  glVertex2f(-0.2, 0.6); // right
  glVertex2f(-0.2, -0.6);//   upright
  glVertex2f(-0.8, 0.0); // crossbar: start...
  glVertex2f(-0.2, 0.0); // ... and end
glEnd();
```

display() (contd.)

```
// Draw H, comprising three lines
glBegin(GL_LINES); // begin a geometric primitive
  glColor3f(1.0, 1.0, 1.0); // set color to white
  glVertex2f(-0.8, -0.6); // left
  glVertex2f(-0.8, 0.6); //   upright
  glVertex2f(-0.2, 0.6); // right
  glVertex2f(-0.2, -0.6);//   upright
  glVertex2f(-0.8, 0.0); // crossbar: start...
  glVertex2f(-0.2, 0.0); // ... and end
glEnd();
```

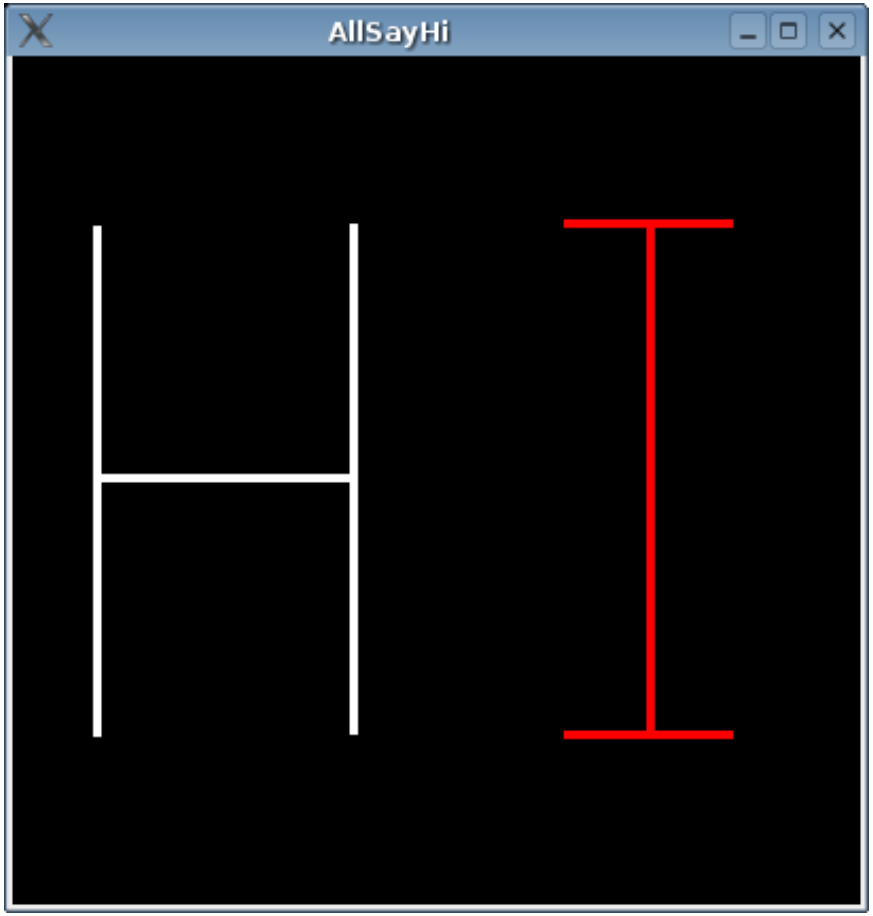
display() (contd.)

```
// Draw H, comprising three lines
glBegin(GL_LINES); // begin a geometric primitive
  glColor3f(1.0, 1.0, 1.0); // set color to white
  glVertex2f(-0.8, -0.6); // left
  glVertex2f(-0.8, 0.6); //   upright
  glVertex2f(-0.2, 0.6); // right
  glVertex2f(-0.2, -0.6);//   upright
  glVertex2f(-0.8, 0.0); // crossbar: start...
  glVertex2f(-0.2, 0.0); // ... and end
glEnd();
```

display() (contd.)

```
// Draw H, comprising three lines
glBegin(GL_LINES); // begin a geometric primitive
  glColor3f(1.0, 1.0, 1.0); // set color to white
  glVertex2f(-0.8, -0.6); // left
  glVertex2f(-0.8, 0.6); //   upright
  glVertex2f(-0.2, 0.6); // right
  glVertex2f(-0.2, -0.6);//   upright
  glVertex2f(-0.8, 0.0); // crossbar: start...
  glVertex2f(-0.2, 0.0); // ... and end
glEnd();
```

`display()` (contd.)



display() (contd.)

```
  // Draw I, comprising three different lines
  glBegin(GL_LINES);
    glColor3f(1.0, 0.0, 0.0); // set color to red
    glVertex2f(0.5, -0.6);
    glVertex2f(0.5, 0.6);
    glVertex2f(0.3, 0.6);
    glVertex2f(0.7, 0.6);
    glVertex2f(0.3, -0.6);
    glVertex2f(0.7, -0.6);
  glEnd();

  glFlush();
}
```