LAB 1: Simple Drawing

In this first lab class, you will learn how to set up and configure environments in order to run OpenGL. After that, you will learn how to draw a simple object and render it on the screen with OpenGL.

When you start to program in **OpenGL**, you might need a decent **Integrated Development** Environment (IDE). Being free, open-source and platform independent, Eclipse (http://www.eclipse.org/) is an excellent choice. You also need a compiler in order to compile OpenGL source. OpenGL is based on C, you can download any C compiler. In this lab, we use MinGW. Go to this website to download MinGW http://www.mingw.org/.

1. Installing Freeglut

Suppose that in your system already comprises Eclipse and MinGW. Now, it is time to install **freeglut** library for OpenGL. For Windows you have to download and extract Martin Payne's binaries for MinGW which are available at http://www.transmissionzero.co.uk/software/freeglut-devel/. In that website, it also has binaries for visual c++ compiler.

- Copy the file **freeglut.dll** to

```
C:\Windows\System32 (for 32bit Operating System)
C:\Windows\SysWOW64
```

- Copy the contents of **include\GL** to the same folder of your MinGW installation, typically

```
C:\MinGW\include\GL
```

- Finally, also copy the contents of **lib** to the same folder of your MinGW installation, typically

```
C:\MinGW\lib
```

- Open or restart Eclipse. Now we need to link the freeglut binaries to our compiler. Right click on you project and select "**Properties**". Go to "**C/C++ Build Settings**" and select the tab "**Tool Settings**". Now select ‘Libraries’ under "**MinGW C++ Linker**". Add the following entries to the libraries:

```
glu32
opengl32
freeglut
```

Important Note: You also need to repeat this step for every project you create.

To generate an image by using computer graphics, it is the same the process as we draw an image on the paper. In order to draw, we need a paper right draw the object. Likewise, in computer graphics, we need a place where the image will be drawn by creating a window. In OpenGL, to create a window to draw the image, it uses the following statement

```
glutInit(&argc, argv);
// set window size 500x 500
glutInitWindowSize (500,500);
// set the position where the created
window should appear on the screen
glutInitWindowPosition (100, 100);
// create window with the given
glutCreateWindow ("Triangle"); name
```

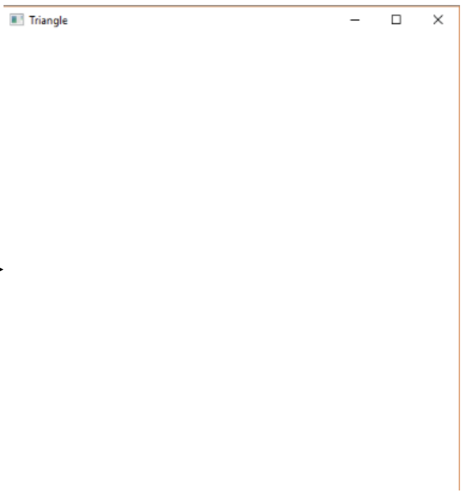

Figure 1: Section 1

In drawing, you can change the color of the paper right? If you want to draw an image on the blue paper, you need to buy blue paper. In OpenGL, you can change the background color of your window as well

```
// Choose color of the background you one
glClearColor (1.0, 1.0, 1.0, 0.0);
```

Figure 2: Section 2

```
// sets the window background color to values previously selected by
glCleareColor
glClear(GL_COLOR_BUFFER_BIT);
```

Figure 3: Section 3

After setting up the window for your drawing, you need to decide which area of your paper you want your object appear. You can choose subset of your entire paper or you can choose the whole paper for your drawing with the following:

```
\\change projection mode
glMatrixMode(GL_PROJECTION);

\\ set current matrix to identiy matrix
glLoadIdentity();
gluOrtho2D(0, 150.0, 0, 150.0);
```

Figure 4: Section 4

To create a model that you want to draw in OpenGL, we use the following statement: In this example, we draw a simple triangle

```
glBegin(GL_POLYGON);
   glVertex2i(75,85);
   glVertex2i(20,40);
   glVertex2i(130,40);
 glEnd();
```

Figure 5: Section 5

Code in section 3 and section 5 are normally wrapped in the same function as follows:

```
void display() {
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POLYGON);
     glVertex2i(75,85);
     glVertex2i(20,40);
     glVertex2i(130,40);
   glEnd();
}
```

Why do we need to do like that? In OpenGL, it follows Event driven programming paradigm. You need the statement that you want to draw in a function and register that function by using function **glutDisplayFunc** in order to make it take effect.

```
glutDisplayFunc(display);
```

Please read more about even-driven programming paradigm below:

Event-Driven Programming

Many modern graphics systems are windows-based and manage the display of multiple overlapping windows. The user can move the windows around the screen using the mouse, and can resize them. Another property of most window-based programs is that they are event driven. This means that the program responds to various events such as a mouse click, the press of a keyboard key, or the resizing of the screen window. The system automatically manages an event queue, which receives messages that certain events have occurred and deals with them on a first-come, first-served basis. The programmer organizes a program as a collection of callback functions that are executed when events occur.

Registering Callback Functions

There must be a way for the programmer to associate each type of event with the desired callback function. This is called registering the callback function. Each even type that is used in the application must be registered with a callback function, which has a name and definition of the programmer’s choosing. For Example:

```
glutDisplayFunc(myDrawObject);

/* the name "myDrawObject" is chosen by the programmer & "glutDisplayFunc" is a name inherent
to GLUT */
```

Code in section 2 and 4 we usually wrap it up into function as well while code in section 1 will be in the main function. To make the statements above work, please include below header file in your program. The full program of the first example is in next page.

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

OpenGL Example

```
        /*****************************************************

         Author: Kor Sokchea
         Faculty of Engineering
        Royal University of Phnom Penh

        *****************************************************/

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

void display(void)
{
   // clear background of current window
   glClear(GL_COLOR_BUFFER_BIT);

   // draw a filled polygon
   glBegin(GL_POLYGON);

       glVertex2i(75,85);
       glVertex2i(20,40);
       glVertex2i(130,40);
    glEnd();

    glFlush ();

}

void init(void)
{
  \\Clear background color to white
  glClearColor (1.0, 1.0, 1.0, 0.0);

  \\change projection mode
  glMatrixMode(GL_PROJECTION);

  \\ set current matrix to identiy matrix
  glLoadIdentity();
  gluOrtho2D(0, 150.0, 0, 150.0);

}

int main(int argc, char** argv)
```

```
{
  glutInit(&argc, argv);

  glutInitWindowSize (500,500);
  glutInitWindowPosition (100, 100);

  glutCreateWindow ("Simple Drawing");

  init();

  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```

Required Tasks:

1. Set the color of triangle to blue
2. Replace the triangle with a circle. Please draw a circle based on the formula that you teach since you learn in high school. Then, fill it color to red