Lab03 for CPT205 Computer Graphics

Part 1 - OpenGL Exercise

Aims

- to help you further familiarise with MS Visual Studio 2019 and the OpenGL graphics library
- to show you how to specify the dimensions of the Device Area on-screen and the dimensions of the Window in the virtual world
- to show you how to draw basic graphic primitives on-screen

Task 1 – Type-in a Skeleton OpenGL Program

Type in the following program. Then compile and link the program. Run it and observe the result.

```
// File ID: Lab03a.cpp
// Title:   Working with Graphics Primitives
// Author:

#define FREEGLUT_STATIC
#include <GL/freeglut.h>

void define_to_OpenGL();

////////////////////////////////////
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        // Task 2

        glutCreateWindow("Graphics Primitives");

        glutDisplayFunc(define_to_OpenGL);
        glutMainLoop();
}
////////////////////////////////////
void define_to_OpenGL()
{
        glClearColor(1, 1, 1, 1);
        glClear(GL_COLOR_BUFFER_BIT);

        // The stuff to appear on screen goes here
        // Task 2
        // Task 3
        // Task 4
        // Task 5
        // Tasks 6, 7 and 8

        glFlush();
}
```

Task 2 – Set the Dimensions

Set the dimensions of the device area to be: 600 pixels wide by 400 pixels high, positioned at location (50,50)

```
glutInitWindowSize(w, h);
glutInitWindowPosition(x, y);
```

Set the dimensions of the window on screen to be (instead of normalised space):
L=-100 units, R=+500 units, B=-200 units, T=200 units

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(L,R,B,T);
```

Task 3 – Draw the Axes

Draw some basic graphics primitives in the virtual world as follows: Draw an x-axis running from (0.0, 0.0) to (450.0, 0.0). Now draw a y-axis running from (0.0, -150.0) to (0.0, +150.0).

```
glLineWidth(1.0);
```

```
glColor3f(?,?,?);
glBegin(GL_LINES);
        glVertex2f(?,?);  // start location
        glVertex2f(?,?);  // end location
glEnd();
```

Task 4 – Draw a Dot at the Origin
Draw a dot at the origin of the axes. Set the size of the dot to 10 pixels, and the colour of the dot to yellow.

```
glPointSize(?);
glColor3f(?,?,?);
glBegin(GL_POINTS);
        glVertex2f(?,?);
glEnd();
```

Task 5 – Plot a Sine Wave
A sine-wave is visual representation of the values produces by the mathematical Sine function when calculated for an angle size ranging from 0 degrees to 360 degrees [value = sine(angle)]. Use the following code to show the Sine-wave as a series of dots. The head file `#include "math.h"` will be used at the beginning.

```
// draw a sine wave
int i;
float x, y;

glColor3f(0.0, 0.0, 1.0);
glPointSize(1);
glBegin(GL_POINTS);
for (i = 0;i < 361;i = i + 5)
{
        x = (float)i;
        y = 100.0 * sin(i * (6.284 / 360.0));
        glVertex2f(x, y);
}
glEnd();
```

Now alter the code so that the Sine-wave is shown as a GL_LINE_STRIP.

Task 6 – Draw a Triangle
Draw a triangle. The values of the corner points of the triangle are: (-50, 50), (-50, 0), (0, 0).
A triangle is an example of a polygon. What is a polygon? Is the polygon shown as an outline shape or as a solid shape?

Write the answers here.

```
glBegin(GL_TRIANGLES);
        glVertex2f(?,?);
        glVertex2f(?,?);
        glVertex2f(?,?);
glEnd();
```

Task 7 – Draw A Multi-coloured Triangle
By 'mult-coloured' we mean a triangle whose corners are assigned different colours and where these colours are "smoothed" across the interior of the shape. Assign the colour red to the first corner, the colour green to the second corner and the colour blue to the third corner.

```
glColor3f(?,?,?);
```

What does the colouring on the triangle look like? Record your answer here.

Task 8 – Draw a Single-coloured Triangle
By 'single-coloured' we mean a triangle whose corners and interior is the same colour. Tell OpenGL to turn-off the smoothing capability.

```
glShadeModel(GL_FLAT);
```

What colour is used? How does OpenGL know to use this colour? Write your answers here.

Part 2 – Graphic Primitives

1) DDA Line Algorithm

Read the lecture notes on the DDA algorithm. For each of the following two lines

Line1 from (-2, 3) to (10, 8)
Line 2 from (-2, 3) to (-22, 33)

a) Manually work out the pixel positions using the DDA algorithm.

b) Write a program to decide and output the pixel positions that are to be displayed on the screen.
Hint: This will not involve OpenGL and you can refer to sample code for Lab 02 as a starting point.

2) Circle Algorithms

Read the lecture notes on circle algorithms. Explain how symmetry of a circle can be used to improve the efficiency of computation in generating the circle.

3) Given line AB represented by points A(x_A, y_A) and B(x_B,y_B), and line CD represented by points C(x_C, y_C) and D(x_D,y_D), **write a program that calls OpenGL to**

a) draw AB and CD in the window you have created by altering the sample program in Part I, and using `glBegin(GL_LINES)`

b) calculate and output the lengths of the lines

c) calculate and output the gradients

d) check if the lines are perpendicular or parallel to each other

e) draw points A, B, C and D at the same time when the lines are drawn by calling `glBegin(POINTS)`.

Note: you can define the co-ordinate values of points A, B, C and D, and use `cin>>` and `cout<<` to read in and print the output on screen (e.g. in the control window) as you did in Lab 02. The beginning of your program may look like

```
// File ID: lab03b.cpp
// Title: Interactive program for calculating area of a circle
// Author:

#define FREEGLUT_STATIC
#include <GL/freeglut.h>
#include <math.h>
#include <iostream>

using namespace std;

void define_to_OpenGL();  // Draw geometric elements and output results
void init();              // Initialise coordinates of line end points
float a[8];               // Coordinates of line end points

int main(int argc, char** argv)
{
        ...
}
```