**Lab Report. 07**

**Subject: Computer Graphics Lab**



Submitted to: Dr. Junaid

Submitted by: Nayab Ashraf (RC-299)

**Date of Submission: 30 October, 2024**

National University of Modern Languages, Department of Computer Science, Rawalpindi Campus.

**Task 01:**

**Implement viewport codes by changing the colors of squares.**

#include<windows.h>

#include <GL/gl.h>

#include <gl/glut.h>

void myInit()

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glPointSize(10.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluOrtho2D(0.0, 500.0, 0.0, 500.0);

}

void DDL(int x1, int y1, int x2, int y2)

{

glBegin(GL\_LINES);

glVertex2i(x1, y1);

glVertex2i(x2, y2);

glEnd();

}

void Display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glViewport(0, 0, 320, 240);

glColor3f(0.0, 1.0, 0.0);

glRecti(0, 0, 320, 240);

glColor3f(1.0, 0.0, 1.0);

glRecti(160, 120, 480, 360);

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(0, 0);

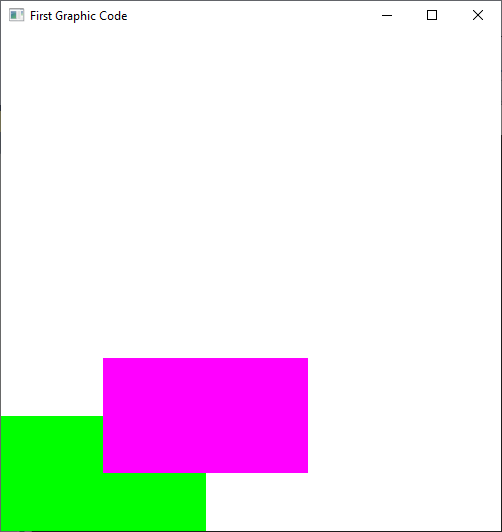
glutCreateWindow("First Graphic Code");

myInit();

glutDisplayFunc(Display);

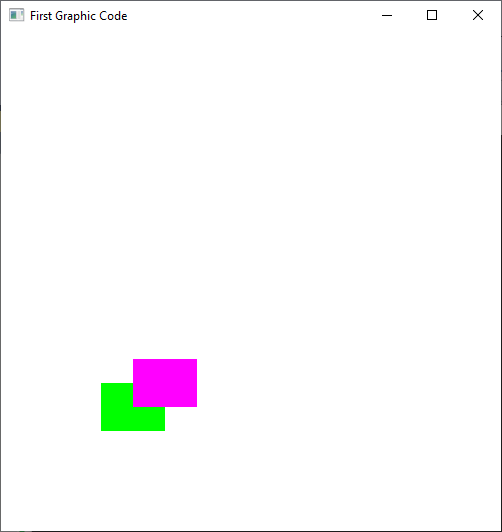
glutMainLoop();

}

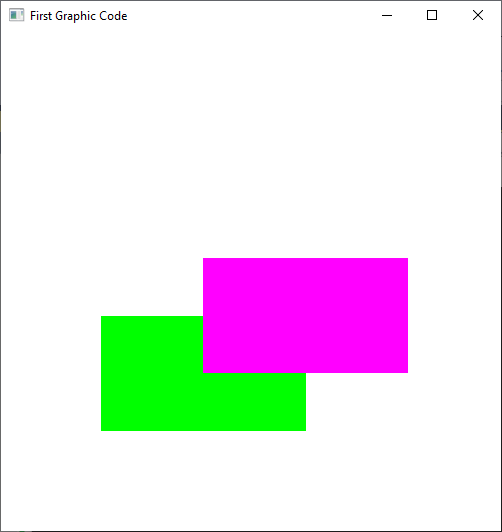
****

**After Changing position (viewport)**

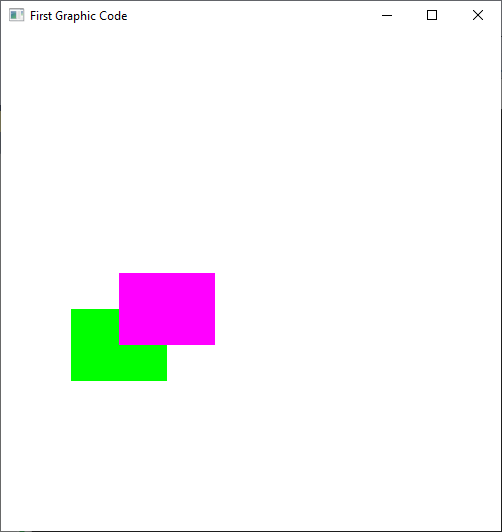
glViewport(100, 100, 100, 100);

****

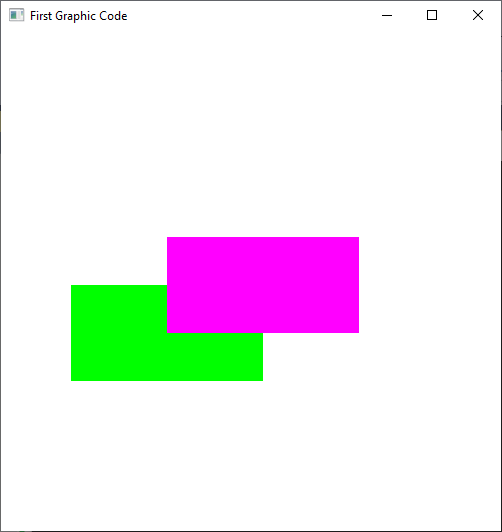
glViewport(100, 100, 320, 240);

****

glViewport(70, 150, 150, 150);

****

glViewport(70, 150, 300, 200);

****

**Task 02:**

**Create two viewports.**

#include<Windows.h>

#include<GL\glut.h>

#include<math.h>

void display()

{

// Make background color white

glClearColor(1.0, 1.0, 1.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

// Sets up FIRST viewport spanning the left-bottom quarter of the interface window

**// Draw PINK rectangle**

**glViewport(0, 0, 250, 250);**

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 250.0, 0.0, 250.0); // also sets up world window

**// Draw RED rectangle**

glColor3f(1, 0, 1);

glRectf(0.0, 0.0, 125.0, 125.0);

// continues

**glViewport(150, 150, 150, 150);**

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 250.0, 0.0, 250.0);

// also sets up world window

// Draw RED rectangle

glColor3f(1, 0, 0);

glRectf(125.0, 125.0, 250.0, 250.0);

// display rectangles

glutSwapBuffers();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500, 500);

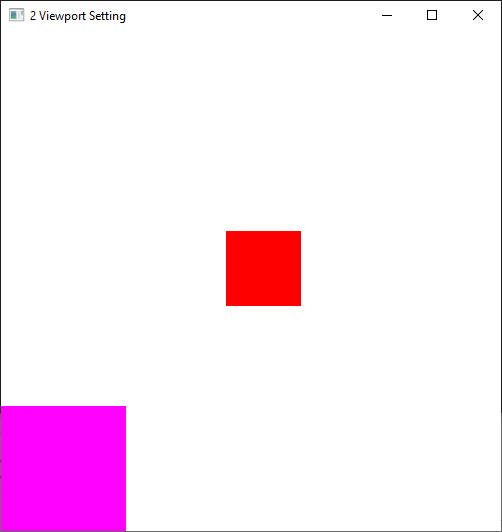
glutInitWindowPosition(150, 150);

glutCreateWindow("2 Viewport Setting");

glutDisplayFunc(display);

glutMainLoop();

}

****

void display()

{

// Make background color white

glClearColor(1.0, 1.0, 1.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

// Sets up FIRST viewport spanning the left-bottom quarter of the interface window

//PINK

glViewport(100, 100, 250, 250);

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 250.0, 0.0, 250.0); // also sets up world window

// Draw RED rectangle

glColor3f(1, 0, 1);

glRectf(0.0, 0.0, 125.0, 125.0);

// continues

glViewport(150, 150, 250, 250);

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 250.0, 0.0, 250.0);

// also sets up world window

// Draw RED rectangle

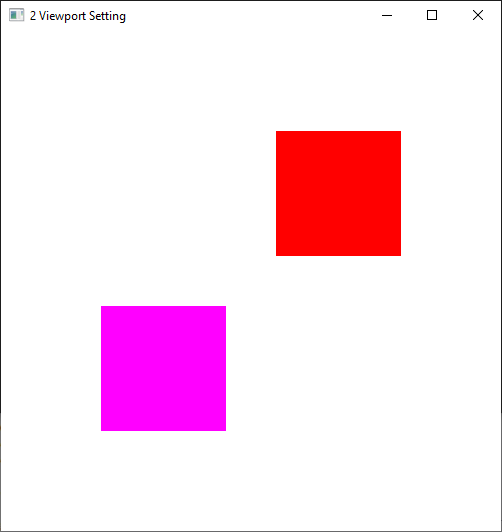
glColor3f(1, 0, 0);

glRectf(125.0, 125.0, 250.0, 250.0);

// display rectangles

glutSwapBuffers();

}

****

**Task 03:**

**Write a program to implement the DDA(Digital Differential Analyzer) line drawing Algorithm**

#include<windows.h>

#include <GL/gl.h>

#include <math.h>

#include <gl/glut.h>

void myInit()

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glPointSize(10.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluOrtho2D(0.0, 500.0, 0.0, 500.0);

}

float sign(float RecDelta)

{

if (RecDelta > 0)

{

return 1.0;

}

else if (RecDelta < 0)

{

return -1.0;

}

else if (RecDelta == 0)

{

return 0.0;

}

}

void DDL(int x1, int y1, int x2, int y2)

{

int length;

if (abs(x2 - x1) >= abs(y2 - y1))

{

length = abs(x2 - x1);

}

else

{

length = abs(y2 - y1);

}

float DeltaX = (float)(x2 - x1) / (float)length;

float DeltaY = (float)(y2 - y1) / (float)length;

float x = (float)x1 + 0.5 \* sign(DeltaX);

float y = (float)y1 + 0.5 \* sign(DeltaY);

int i = 1;

while (i <= length)

{

glBegin(GL\_POINTS);

glVertex2i((int)x, (int)y);

//glVertex2f(x, y);

glEnd();

x = x + DeltaX;

y = y + DeltaY;

i = i + 1;

}

}

void Display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 0.0);

DDL(50, 50, 250, 500);

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(0, 0);

glutCreateWindow("First Graphic Code");

myInit();

glutDisplayFunc(Display);

glutMainLoop();

}

****

DDL(50, 50, 400, 400);

****