**Lab Report. 06**

**Subject: Computer Graphics Lab**

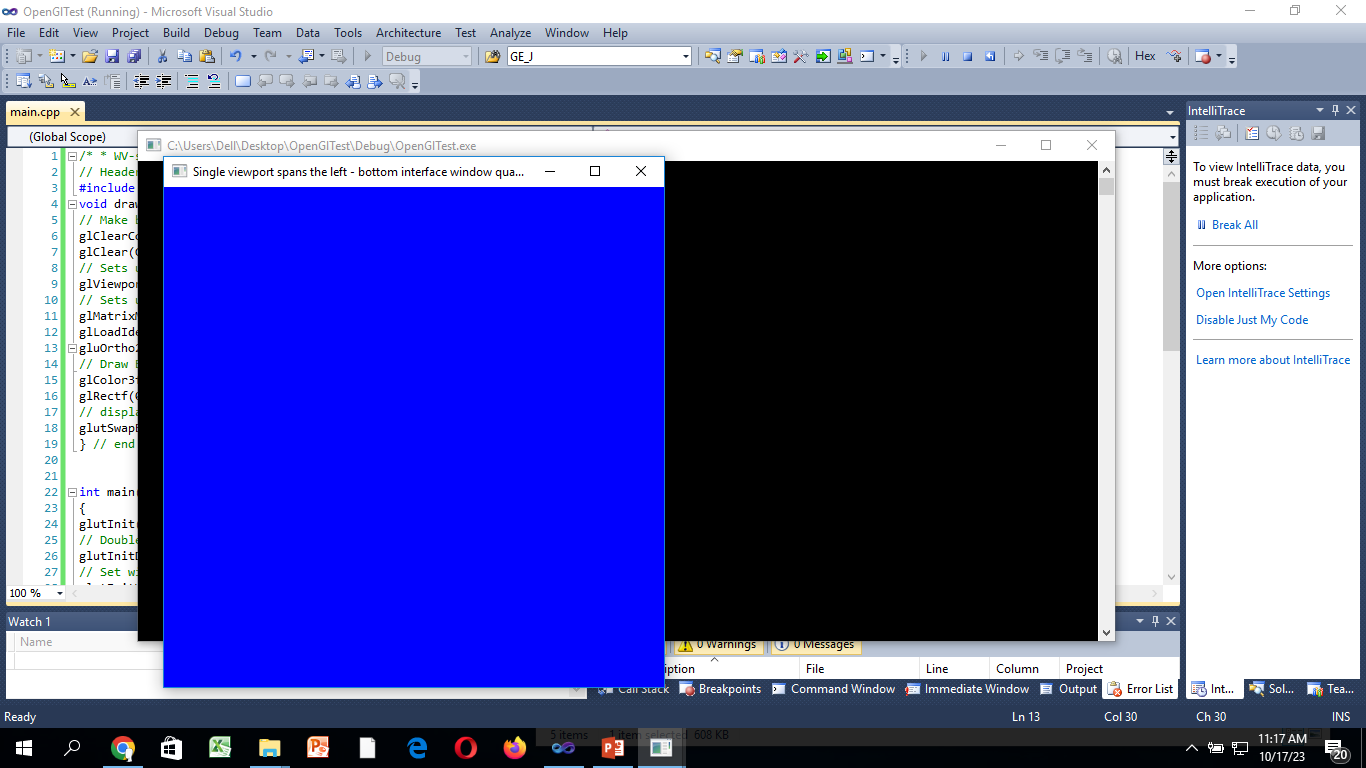
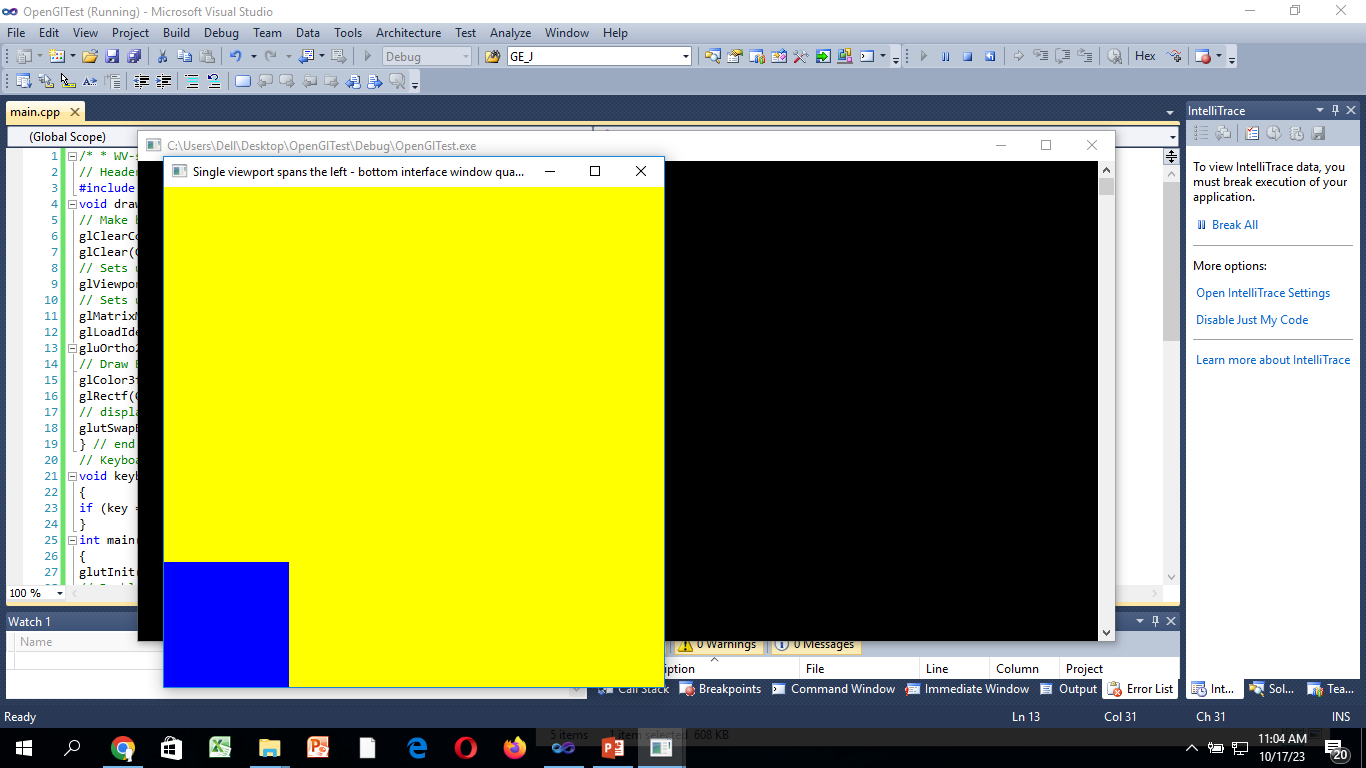
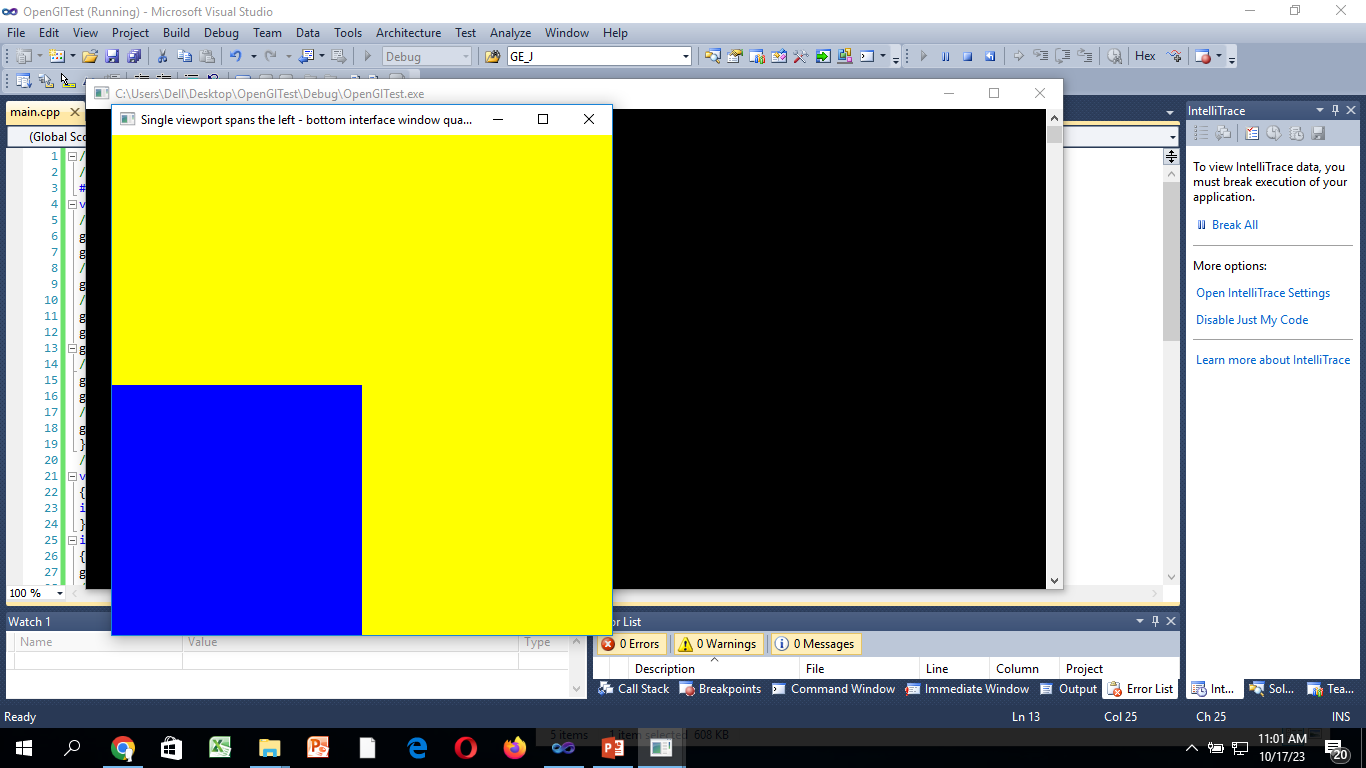
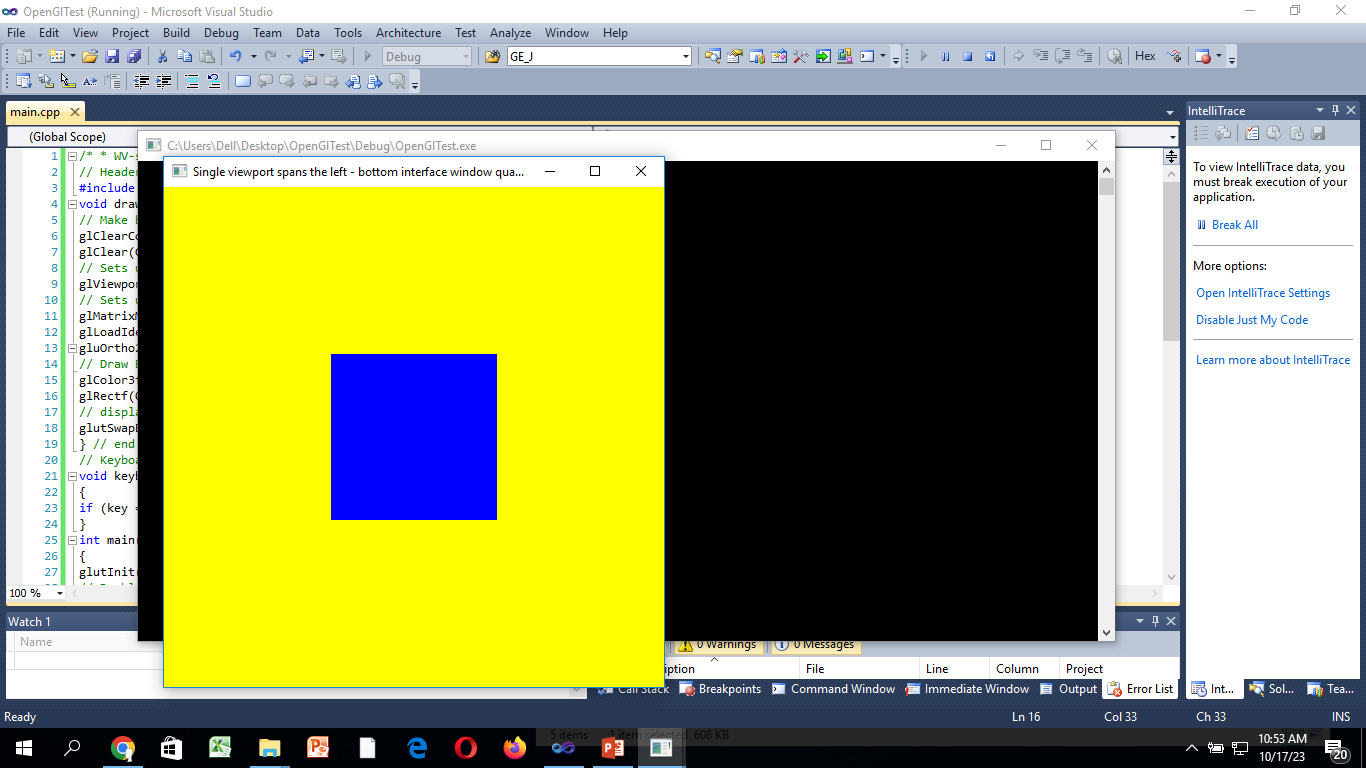


Submitted to: Dr. Junaid

Submitted by: Nayab Ashraf (RC-299)

**Date of Submission: 23 October, 2024**

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



**Fig. a Fig. b Fig. c Fig. d**

**Task 01:**

**Use glRecti() function to draw a square/ rectangle (fig. a)**

#include <GL/glut.h> // Header File For The GLut Library

void draw() {

// Make background colour yellow

glClearColor(1.0, 1.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glViewport(0, 0, 500, 500);

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-500.0, 1000.0, -500.0, 1000.0);

// Draw BLUE rectangle

glColor3f(0, 0, 1);

glRecti(0.0, 0.0, 500.0, 500.0);

// display rectangles

glutSwapBuffers();

} // end of draw()

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

{

glutInit(&argc, argv);

// Double Buffered RGB display

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

// Set window size

glutInitWindowSize(500, 500);

glutCreateWindow("Single viewport");

// Declare the display and keyboard functions

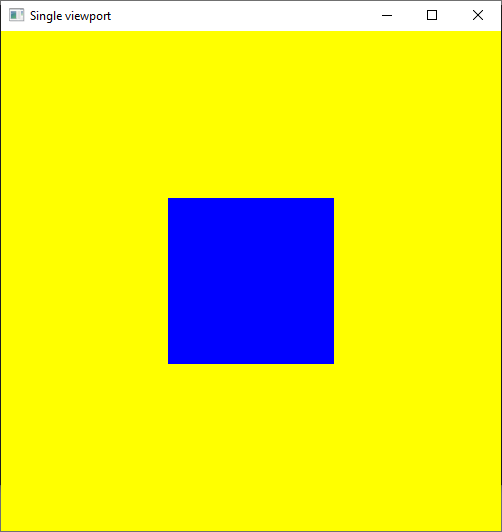
glutDisplayFunc(draw);

// Start the Main Loop

glutMainLoop();

return 0;

}

****

**Task 02:**

**Without changing the vertices of the square, move it towards the lower bottom of the screen. Hint: focus on gluOrtho2D() function (fig b).**

#include <GL/glut.h> // Header File For The GLut Library

void draw() {

// Make background colour yellow

glClearColor(1.0, 1.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glViewport(0, 0, 500, 500);

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 1000.0, 0.0, 1000.0);

// Draw BLUE rectangle

glColor3f(0, 0, 1);

glRecti(0.0, 0.0, 500.0, 500.0);

// display rectangles

glutSwapBuffers();

} // end of draw()

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

{

glutInit(&argc, argv);

// Double Buffered RGB display

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

// Set window size

glutInitWindowSize(500, 500);

glutCreateWindow("Single viewport");

// Declare the display and keyboard functions

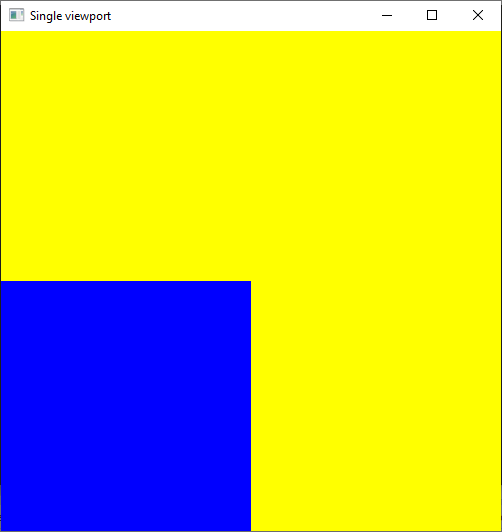
glutDisplayFunc(draw);

// Start the Main Loop

glutMainLoop();

return 0;

}

****

**Task 03:**

**Without changing the vertices of the square, reduce its size. Hint: focus on gluOrtho2D() function(fig c)**

#include <GL/glut.h> // Header File For The GLut Library

void draw() {

// Make background colour yellow

glClearColor(1.0, 1.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glViewport(0, 0, 500, 500);

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 2000.0, 0.0, 2000.0);

// Draw BLUE rectangle

glColor3f(0, 0, 1);

glRecti(0.0, 0.0, 500.0, 500.0);

// display rectangles

glutSwapBuffers();

} // end of draw()

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

{

glutInit(&argc, argv);

// Double Buffered RGB display

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

// Set window size

glutInitWindowSize(500, 500);

glutCreateWindow("Single viewport");

// Declare the display and keyboard functions

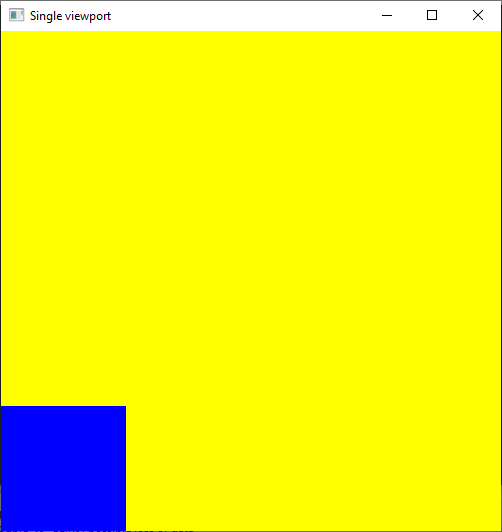
glutDisplayFunc(draw);

// Start the Main Loop

glutMainLoop();

return 0;

}

****

**Task 04:**

**Show a big square full screen size (fig d) without changing its vertices.**

#include <GL/glut.h> // Header File For The GLut Library

void draw() {

// Make background colour yellow

glClearColor(1.0, 1.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glViewport(0, 0, 500, 500);

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 500.0, 0.0, 500.0);

// Draw BLUE rectangle

glColor3f(0, 0, 1);

glRecti(0.0, 0.0, 500.0, 500.0);

// display rectangles

glutSwapBuffers();

} // end of draw()

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

{

glutInit(&argc, argv);

// Double Buffered RGB display

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

// Set window size

glutInitWindowSize(500, 500);

glutCreateWindow("Single viewport");

// Declare the display and keyboard functions

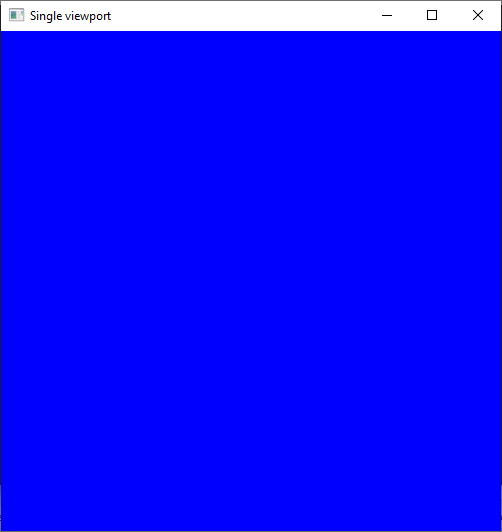
glutDisplayFunc(draw);

// Start the Main Loop

glutMainLoop();

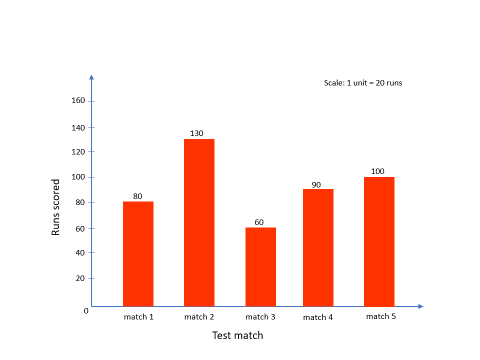
return 0;

}

****

**Task 05:**

**Make a graph that shows score in an over (cricket match).**

****

#include <GL/glut.h> // Header File For The GLut Library

void draw() {

// Make background colour yellow

glClearColor(1.0, 1.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

glViewport(0, 0, 500, 500);

// Sets up the PROJECTION matrix

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 500.0, 0.0, 500.0);

// Draw BLUE rectangle

glColor3f(0, 0, 1);

glRecti(20.0, 0.0, 40.0, 80.0);

glRecti(60.0, 0.0, 80.0, 130.0);

glRecti(100.0, 0.0, 120.0, 60.0);

glRecti(140.0, 0.0, 160.0, 90.0);

glRecti(180.0, 0.0, 200.0, 100.0);

// display rectangles

glutSwapBuffers();

} // end of draw()

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

{

glutInit(&argc, argv);

// Double Buffered RGB display

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

// Set window size

glutInitWindowSize(500, 500);

glutCreateWindow("Single viewport");

// Declare the display and keyboard functions

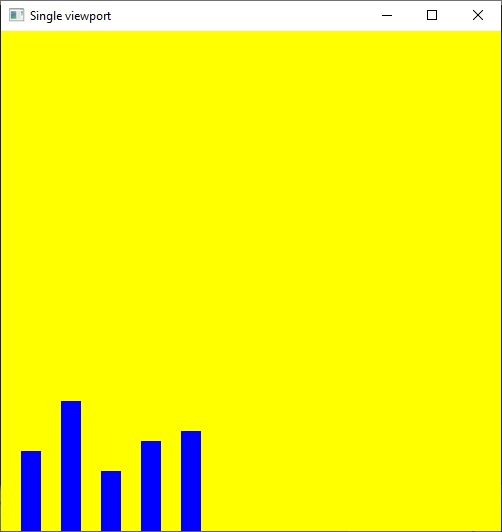
glutDisplayFunc(draw);

// Start the Main Loop

glutMainLoop();

return 0;

}

****