**Assignment-2**

**Ques1. Create empty window (Black, White and different Colors)**

**Black Window**

#include<GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // clear buffers to preset values

// glClear - OpenGL 4 Reference Pages (khronos.org)

glColor3f(0.0, 1.0, 0.0); //

// glColor3f function (Gl.h) - Win32 apps | Microsoft Learn

glEnd();

glFlush(); // The glFlush function empties all these buffers

}

void myinit() {

glClearColor(0.0, 0.0, 0.0, 0.0); // The glClearColor function specifies clear values for the color buffers.

glColor3f(1.0, 0.0, 0.0);

glPointSize(5.0); // The glPointSize function specifies the diameter of rasterized points.

glMatrixMode(GL\_PROJECTION); // The glMatrixMode function specifies which matrix is the current matrix.

gluOrtho2D(0.0, 499.0, 0.0, 499.0); // The glOrtho function multiplies the current matrix by an orthographic matrix.

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(10.0, 0);

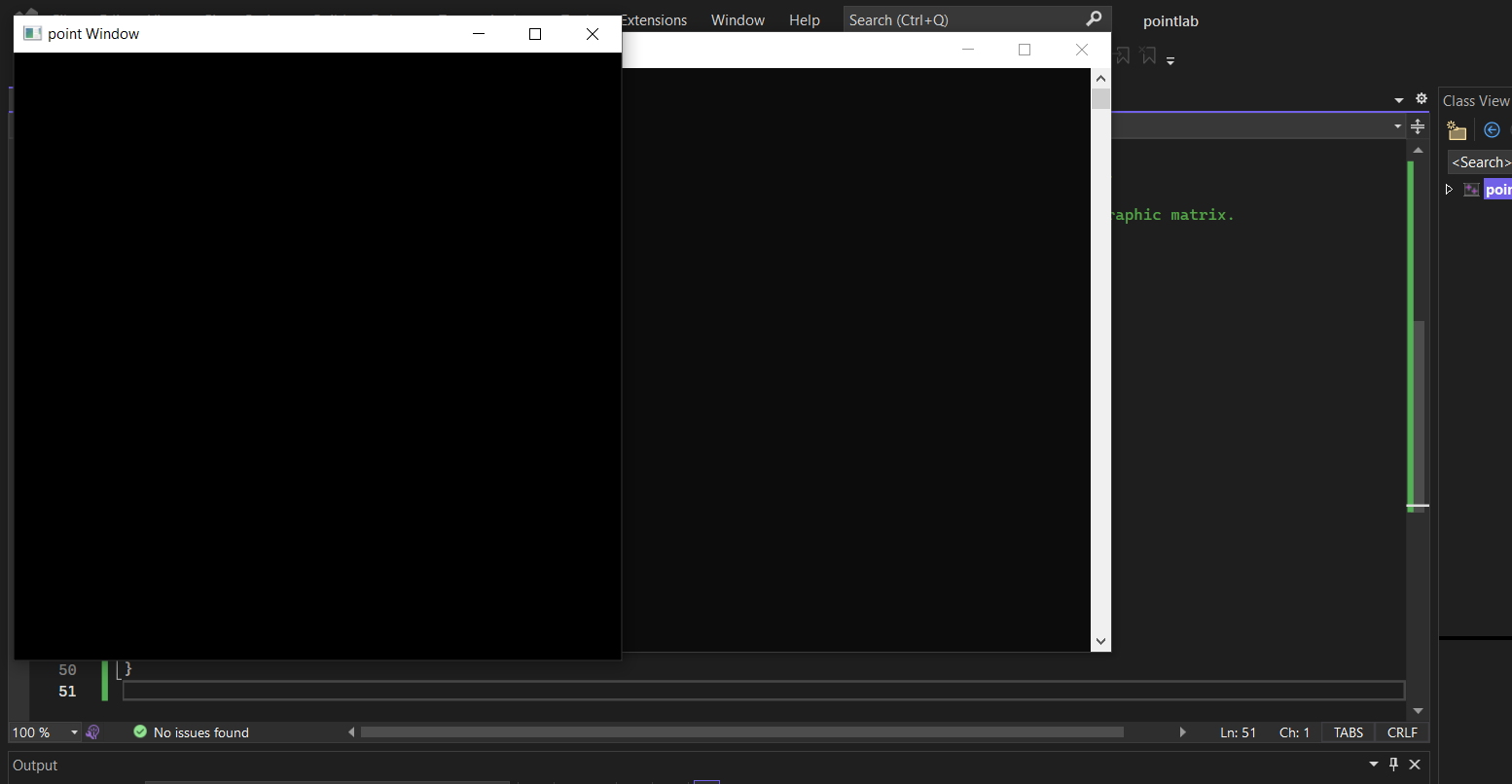
glutCreateWindow("point Window");

glutDisplayFunc(display);

myinit();

glutMainLoop();

}



**White Window:**

#include<GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // clear buffers to preset values

// glClear - OpenGL 4 Reference Pages (khronos.org)

glColor3f(1.0, 1.0, 1.0); //

// glColor3f function (Gl.h) - Win32 apps | Microsoft Learn

glEnd();

glFlush(); // The glFlush function empties all these buffers

}

void myinit() {

glClearColor(1.0, 1.0, 1.0, 1.0); // The glClearColor function specifies clear values for the color buffers.

glColor3f(1.0, 0.0, 0.0);

glPointSize(5.0); // The glPointSize function specifies the diameter of rasterized points.

glMatrixMode(GL\_PROJECTION); // The glMatrixMode function specifies which matrix is the current matrix.

gluOrtho2D(0.0, 499.0, 0.0, 499.0); // The glOrtho function multiplies the current matrix by an orthographic matrix.

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(10.0, 0);

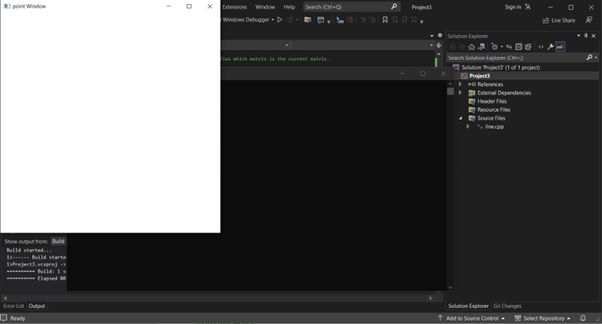
glutCreateWindow("point Window");

glutDisplayFunc(display);

myinit();

glutMainLoop();

}



**Coloured Window:**

#include<GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // clear buffers to preset values

// glClear - OpenGL 4 Reference Pages (khronos.org)

glColor3f(1.0, 1.0, 1.0); //

// glColor3f function (Gl.h) - Win32 apps | Microsoft Learn

glEnd();

glFlush(); // The glFlush function empties all these buffers

}

void myinit() {

glClearColor(2.0, 0.0, 4.0, 1.0); // The glClearColor function specifies clear values for the color buffers.

glColor3f(1.0, 0.0, 0.0);

glPointSize(5.0); // The glPointSize function specifies the diameter of rasterized points.

glMatrixMode(GL\_PROJECTION); // The glMatrixMode function specifies which matrix is the current matrix.

gluOrtho2D(0.0, 499.0, 0.0, 499.0); // The glOrtho function multiplies the current matrix by an orthographic matrix.

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(10.0, 0);

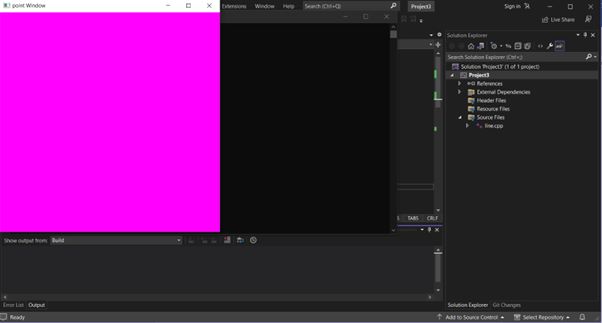
glutCreateWindow("point Window");

glutDisplayFunc(display);

myinit();

glutMainLoop();

}



**Ques2 Draw a point of width 10 pixel**

#include<GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // clear buffers to preset values

// glClear - OpenGL 4 Reference Pages (khronos.org)

glColor3f(1.0, 0.0, 0.0); //

// glColor3f function (Gl.h) - Win32 apps | Microsoft Learn

glBegin(GL\_POINTS);

glVertex2f(150.0, 180.0);

glEnd();

glFlush(); // The glFlush function empties all these buffers

}

void myinit() {

glClearColor(0.0, 0.0, 0.0, 0.0); // The glClearColor function specifies clear values for the color buffers.

glColor3f(1.0, 0.0, 0.0);

glPointSize(10.0); // The glPointSize function specifies the diameter of rasterized points.

glMatrixMode(GL\_PROJECTION); // The glMatrixMode function specifies which matrix is the current matrix.

gluOrtho2D(0.0, 499.0, 0.0, 499.0); // The glOrtho function multiplies the current matrix by an orthographic matrix.

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(1000, 1000);

glutInitWindowPosition(10.0, 0);

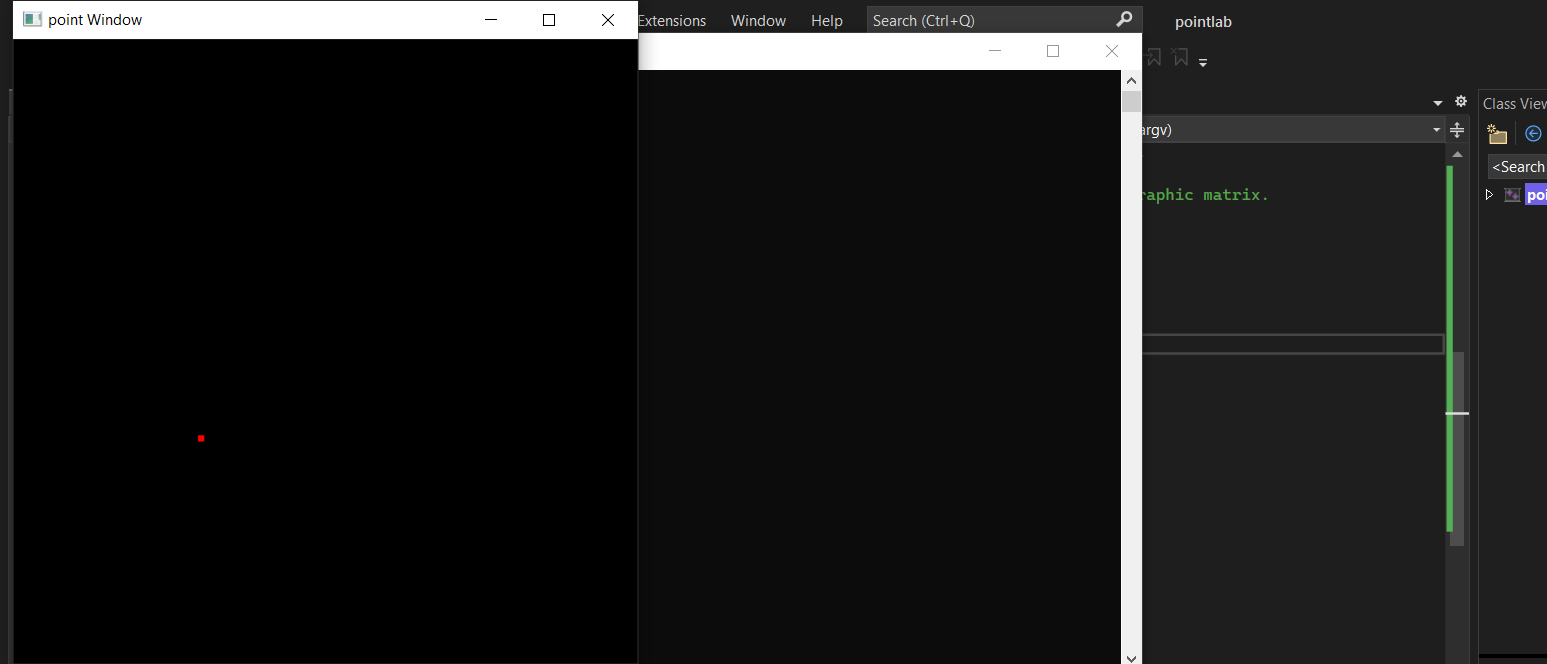
glutCreateWindow("point Window");

glutDisplayFunc(display);

myinit();

glutMainLoop();

}



**Ques3 Draw a green color line from (10,10) to (50,50)**

#include<GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // clear buffers to preset values

// glClear - OpenGL 4 Reference Pages (khronos.org)

glColor3f(0.0, 1.0, 0.0); //

// glColor3f function (Gl.h) - Win32 apps | Microsoft Learn

glBegin(GL\_LINES);

glVertex2f(150.0, 180.0);

glVertex2f(180.0, 190.0);

glEnd();

glFlush(); // The glFlush function empties all these buffers

}

void myinit() {

glClearColor(0.0, 0.0, 0.0, 0.0); // The glClearColor function specifies clear values for the color buffers.

glColor3f(1.0, 0.0, 0.0);

glPointSize(5.0); // The glPointSize function specifies the diameter of rasterized points.

glMatrixMode(GL\_PROJECTION); // The glMatrixMode function specifies which matrix is the current matrix.

gluOrtho2D(0.0, 499.0, 0.0, 499.0); // The glOrtho function multiplies the current matrix by an orthographic matrix.

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(1000, 1000);

glutInitWindowPosition(10.0, 0);

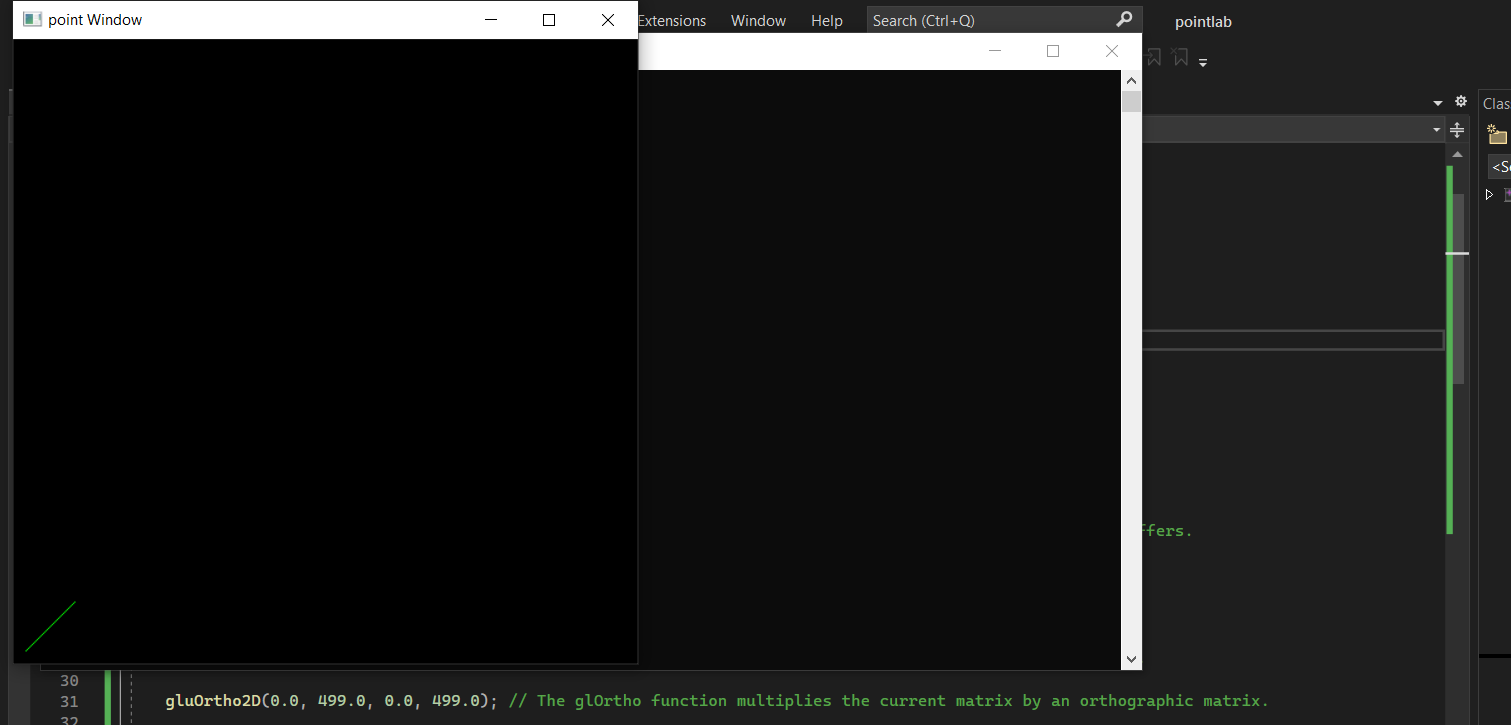
glutCreateWindow("point Window");

glutDisplayFunc(display);

myinit();

glutMainLoop();

}



**Ques4 Draw a triangle on black background**

#include<GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // clear buffers to preset values

// glClear - OpenGL 4 Reference Pages (khronos.org)

glColor3f(1.0, 0.0, 0.0); //

// glColor3f function (Gl.h) - Win32 apps | Microsoft Learn

glBegin(GL\_TRIANGLES);

glVertex2f(100.0, 150.0);

glVertex2f(150.0, 100.0);

glVertex2f(200.0, 300.0);

glEnd();

glFlush(); // The glFlush function empties all these buffers

}

void myinit() {

glClearColor(0.0, 0.0, 0.0, 0.0); // The glClearColor function specifies clear values for the color buffers.

glColor3f(1.0, 0.0, 0.0);

glPointSize(5.0); // The glPointSize function specifies the diameter of rasterized points.

glMatrixMode(GL\_PROJECTION); // The glMatrixMode function specifies which matrix is the current matrix.

gluOrtho2D(0.0, 499.0, 0.0, 499.0); // The glOrtho function multiplies the current matrix by an orthographic matrix.

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(1000, 1000);

glutInitWindowPosition(10.0, 0);

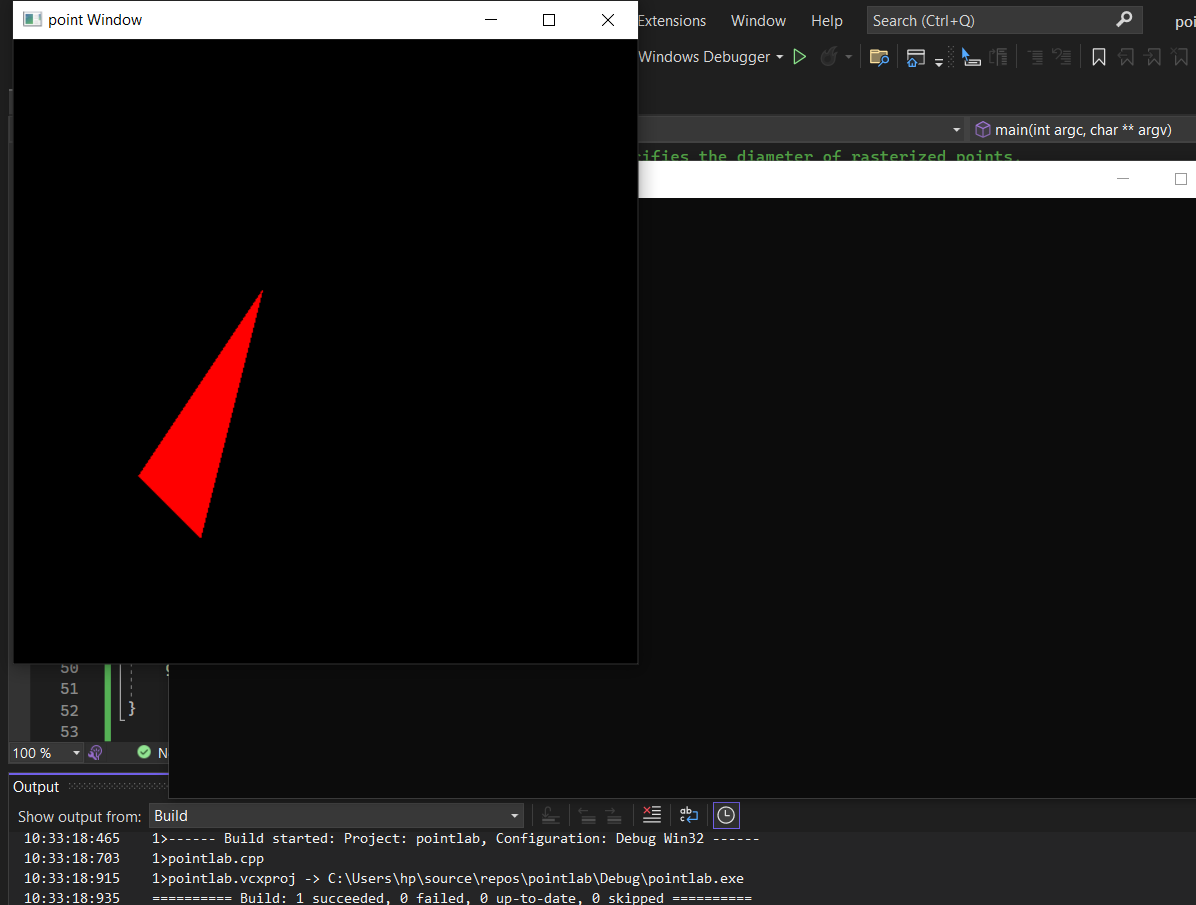
glutCreateWindow("point Window");

glutDisplayFunc(display);

myinit();

glutMainLoop();

}



**Ques5 Draw a rectangle on black background**  
#include<GL/glut.h>

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT); // clear buffers to preset values

// glClear - OpenGL 4 Reference Pages (khronos.org)

glColor3f(0.0, 1.0, 0.0); //

// glColor3f function (Gl.h) - Win32 apps | Microsoft Learn

glBegin(GL\_POLYGON);

glVertex2i(50.0, 90.0);

glVertex2i(100.0, 90.0);

glVertex2i(100.0, 150.0);

glVertex2i(50.0, 150.0);

glEnd();

glFlush(); // The glFlush function empties all these buffers

}

void myinit() {

glClearColor(0.0, 0.0, 0.0, 0.0); // The glClearColor function specifies clear values for the color buffers.

glColor3f(1.0, 0.0, 0.0);

glPointSize(5.0); // The glPointSize function specifies the diameter of rasterized points.

glMatrixMode(GL\_PROJECTION); // The glMatrixMode function specifies which matrix is the current matrix.

gluOrtho2D(0.0, 499.0, 0.0, 499.0); // The glOrtho function multiplies the current matrix by an orthographic matrix.

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(1000, 1000);

glutInitWindowPosition(10.0, 0);

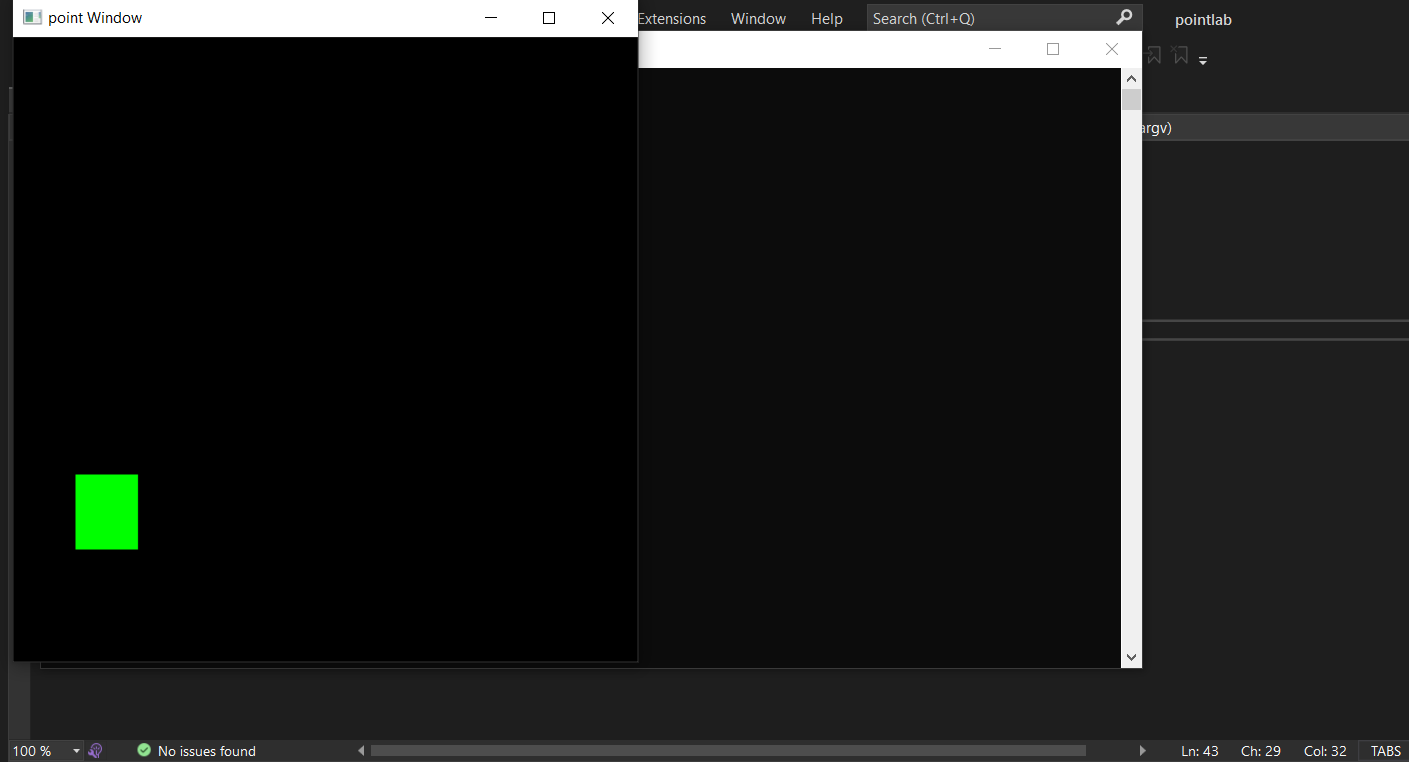
glutCreateWindow("point Window");

glutDisplayFunc(display);

myinit();

glutMainLoop();

}



**Assignment-3**

**Ques: DDA algorithm**

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

#define ROUND(x) ((int)(x+0.5))

int xa, xb, ya, yb;

void display(void) {

int dx = xb

- xa, dy = yb

- ya, steps, k;

float xIncrement, yIncrement, x = xa, y = ya;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 0.0);

if (abs(dx) > abs(dy))

steps = abs(dx);

else

steps = abs(dy);

xIncrement = dx / (float)steps;

yIncrement = dy / (float)steps;

glBegin(GL\_POINTS);

glVertex2s(ROUND(x), ROUND(y));

for (k = 0; k < steps; k++) {

x += xIncrement;

y += yIncrement;

glVertex2s(ROUND(x), ROUND(y));

printf("%lf %lf\n", x, y);

}

glColor3f(1.0, 1.0, 1.0);

for (int i = -100; i <= 100; i++) {

glVertex2s(i, 0);

glVertex2s(0, i);

}

glEnd();

glFlush();

}

void init(void) {

glClearColor(0.0, 0.0, 0.0, 0.0);

glOrtho(-100.0, 100.0, -100.0, 100.0, -1.0, 1.0);

}

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

printf("Enter coordinates of two points :\n");

scanf\_s("%d %d %d %d", &xa, &ya, &xb, &yb);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

glutCreateWindow("Simple DDA ");

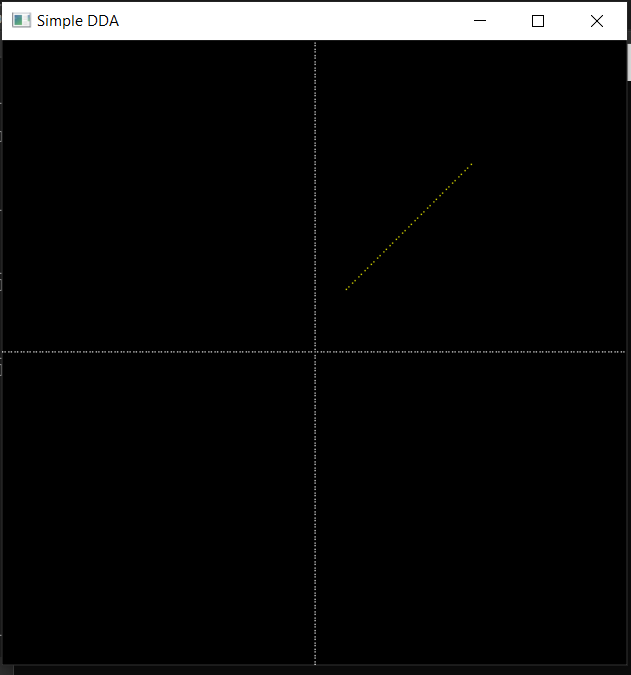
init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}



**Ques: Bresenham Algorithm**

#include <GL/glut.h>

#include<math.h>

void bresenham(int x0, int y0, int x1, int y1) {

int dx = abs(x1 - x0);

int dy = abs(y1 - y0);

int sx = (x0 < x1) ? 1 : -1;

int sy = (y0 < y1) ? 1 : -1;

int err = dx - dy;

glBegin(GL\_POINTS);

while (true) {

glVertex2i(x0, y0);

if (x0 == x1 && y0 == y1) {

break;

}

int e2 = 2 \* err;

if (e2 > -dy) {

err -= dy;

x0 += sx;

}

if (e2 < dx) {

err += dx;

y0 += sy;

}

}

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1, 1, 1);

bresenham(0, 0, 300, 300);

glFlush();

}

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

glutInit(&argc, argv);

glutInitWindowSize(400, 400);

glutCreateWindow("Bresenham Algorithm");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

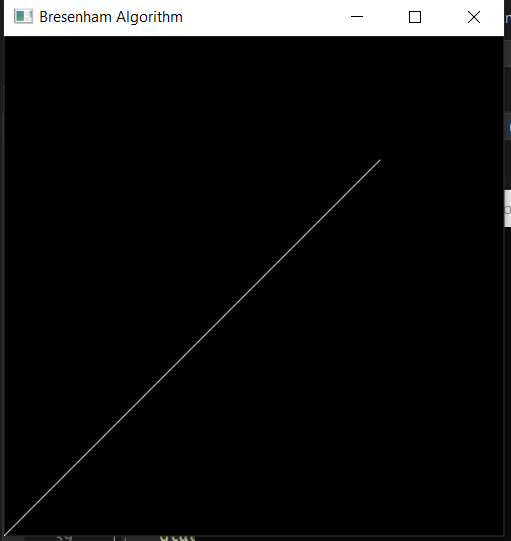
gluOrtho2D(0, 400, 0, 400);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}



**Assignment-4**

**Ques: Draw a circle using Midpoint circle algorithm**

#include <GL/glut.h>

void midpoint\_circle(int xc, int yc, int r) {

int x = 0;

int y = r;

int p = 1 - r;

glBegin(GL\_POINTS);

while (x <= y) {

glVertex2i(xc + x, yc + y);

glVertex2i(xc - x, yc + y);

glVertex2i(xc + x, yc - y);

glVertex2i(xc - x, yc - y);

glVertex2i(xc + y, yc + x);

glVertex2i(xc - y, yc + x);

glVertex2i(xc + y, yc - x);

glVertex2i(xc - y, yc - x);

if (p < 0) {

p += 2 \* x + 3;

}

else {

p += 2 \* (x - y) + 5;

y--;

}

x++;

}

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1, 1, 1);

midpoint\_circle(200, 200, 100);

glFlush();

}

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

glutInit(&argc, argv);

glutInitWindowSize(400, 400);

glutCreateWindow("Midpoint Circle Algorithm");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

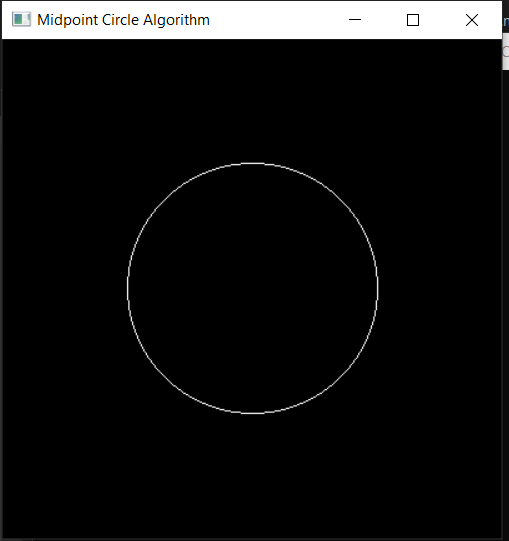
gluOrtho2D(0, 400, 0, 400);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}



**Ques: Draw an ellipse using Midpoint ellipse algorithm**

#include <GL/glut.h>

#include <stdio.h>

#include <math.h>

int round(float x) {

return (int)(x + 0.5);

}

void midpoint\_ellipse(int a, int b, int xc, int yc) {

float dx, dy, d1, d2, x, y;

x = 0;

y = b;

d1 = (b \* b) - (a \* a \* b) + (0.25 \* a \* a);

dx = 2 \* b \* b \* x;

dy = 2 \* a \* a \* y;

glBegin(GL\_POINTS);

while (dx < dy) {

glVertex2i(round(x + xc), round(y + yc));

glVertex2i(round(-x + xc), round(y + yc));

glVertex2i(round(x + xc), round(-y + yc));

glVertex2i(round(-x + xc), round(-y + yc));

if (d1 < 0) {

x++;

dx = dx + (2 \* b \* b);

d1 = d1 + dx + (b \* b);

}

else {

x++;

y--;

dx = dx + (2 \* b \* b);

dy = dy - (2 \* a \* a);

d1 = d1 + dx - dy + (b \* b);

}

}

d2 = ((b \* b) \* ((x + 0.5) \* (x + 0.5))) +

((a \* a) \* ((y - 1) \* (y - 1))) - (a \* a \* b \* b);

while (y >= 0) {

glVertex2i(round(x + xc), round(y + yc));

glVertex2i(round(-x + xc), round(y + yc));

glVertex2i(round(x + xc), round(-y + yc));

glVertex2i(round(-x + xc), round(-y + yc));

if (d2 > 0) {

y--;

dy = dy - (2 \* a \* a);

d2 = d2 + (a \* a) - dy;

}

else {

y--;

x++;

dx = dx + (2 \* b \* b);

dy = dy - (2 \* a \* a);

d2 = d2 + dx - dy + (a \* a);

}

}

glEnd();

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1, 1, 1);

midpoint\_ellipse(100, 50, 200, 200);

glFlush();

}

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

glutInit(&argc, argv);

glutInitWindowSize(400, 400);

glutCreateWindow("Midpoint Ellipse Algorithm");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 400, 0, 400);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

