LAB ASSIGNMENT

COMPUTER GRAPHICS-UCS505

NAME: ABHILASH JENA

SUBGRP: 3COE3

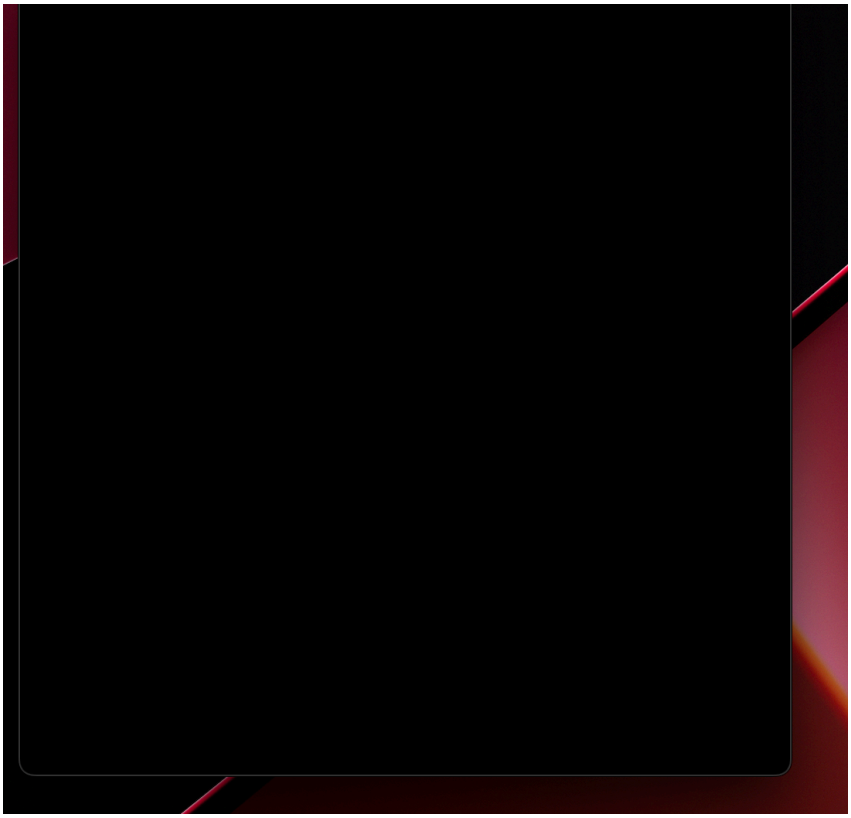
ROLL NO:102053017

Question-1: Create empty window (Black, White and different colors)

```
// C++ program for the above approach
#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#define pi 3.142857

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
```

```
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(0, 0);
```

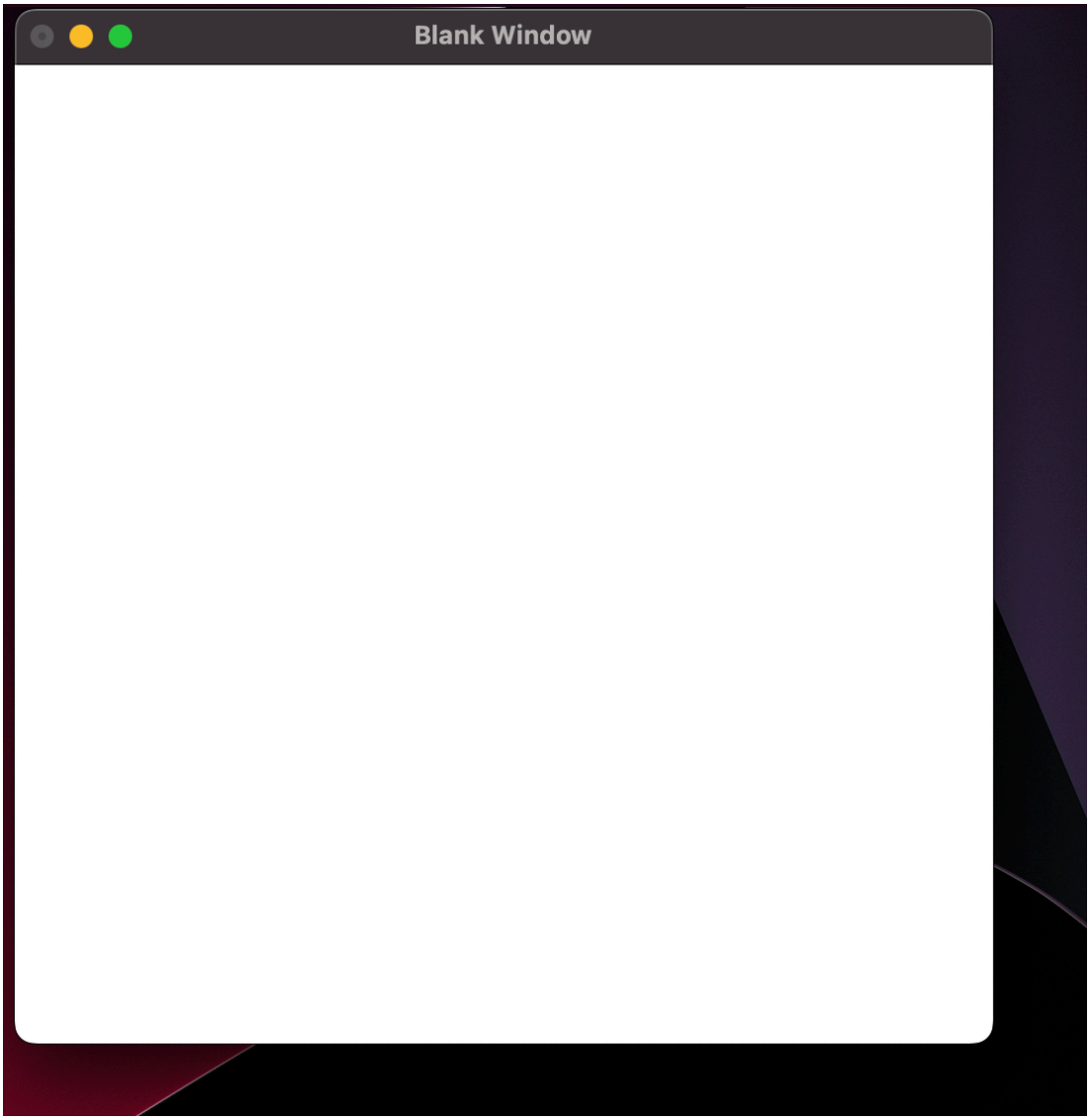
```
    glutCreateWindow("Blank Window");

    glutDisplayFunc(display);

    myinit();

    glutMainLoop();
}
```



2)White Background

```
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#define pi 3.142857

void display() {

glClear(GL_COLOR_BUFFER_BIT); // clear buffers to preset values

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

glColor3f(0.0, 0.0, 0.0);  //

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

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);
```

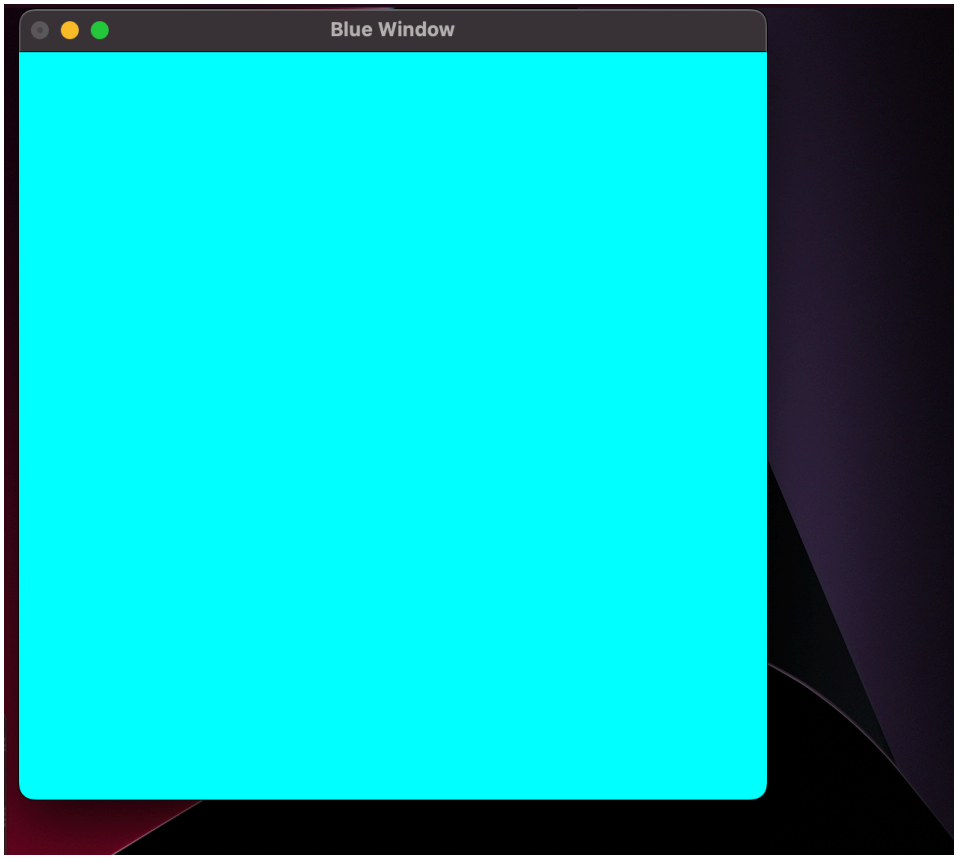
```
glPointSize(5.0); // The glPointSize function specifies the diameter of rasterized points.

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(0, 0);

    glutCreateWindow("Blank Window");

    glutDisplayFunc(display);

    myinit();

    glutMainLoop();
}
```


Blank Window

Blue Background

```
// C++ program for the above approach
#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#define pi 3.142857

void display() {

glClear(GL_COLOR_BUFFER_BIT); // clear buffers to preset values

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

glColor3f(0.0, 0.0, 0.0);  //

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

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

}

void myinit() {

glClearColor(0.0, 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(0, 0);

    glutCreateWindow("Blue Window");

    glutDisplayFunc(display);

    myinit();

    glutMainLoop();
}
```

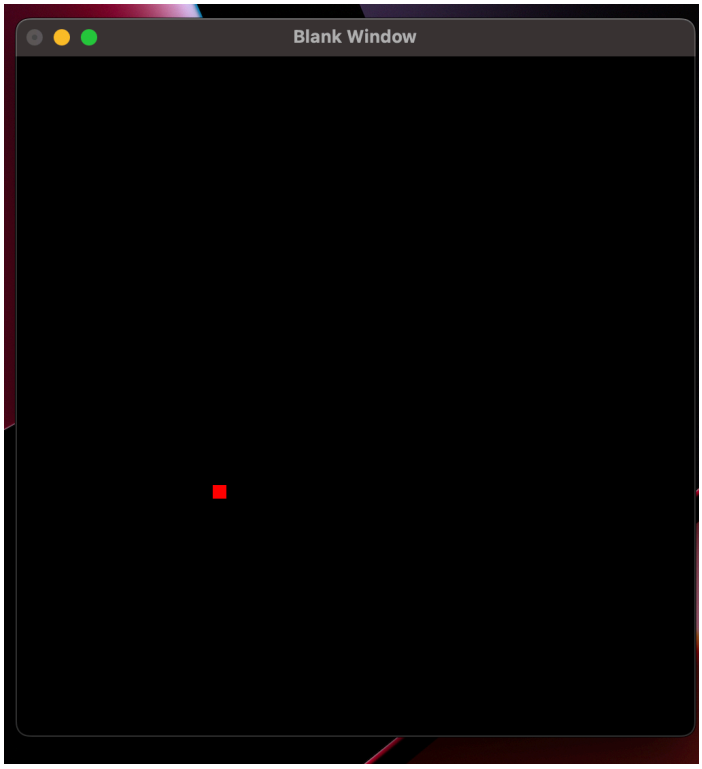

Blue Window

Question-2:Draw a point of width 10 pixel

```
// C++ program for the above approach

#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#define pi 3.142857
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); // 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(500, 500);

glutInitWindowPosition(10.0, 0);

glutCreateWindow("Blank Window");

glutDisplayFunc(display);
```

```
myinit();

glutMainLoop();

}
```

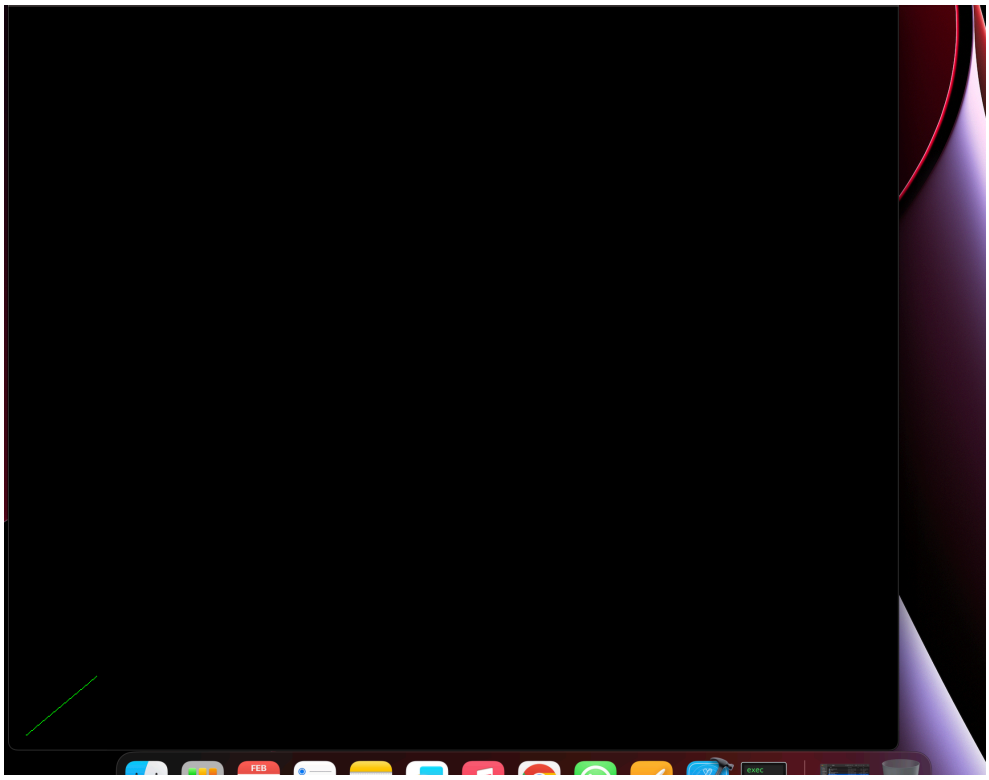


Question-3:Draw a green color line from (10,10) to (50,50)

```
// C++ program for the above approach
#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#define pi 3.142857
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(10.0, 10.0);
   glVertex2f(50.0, 50.0);
   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();

}
```

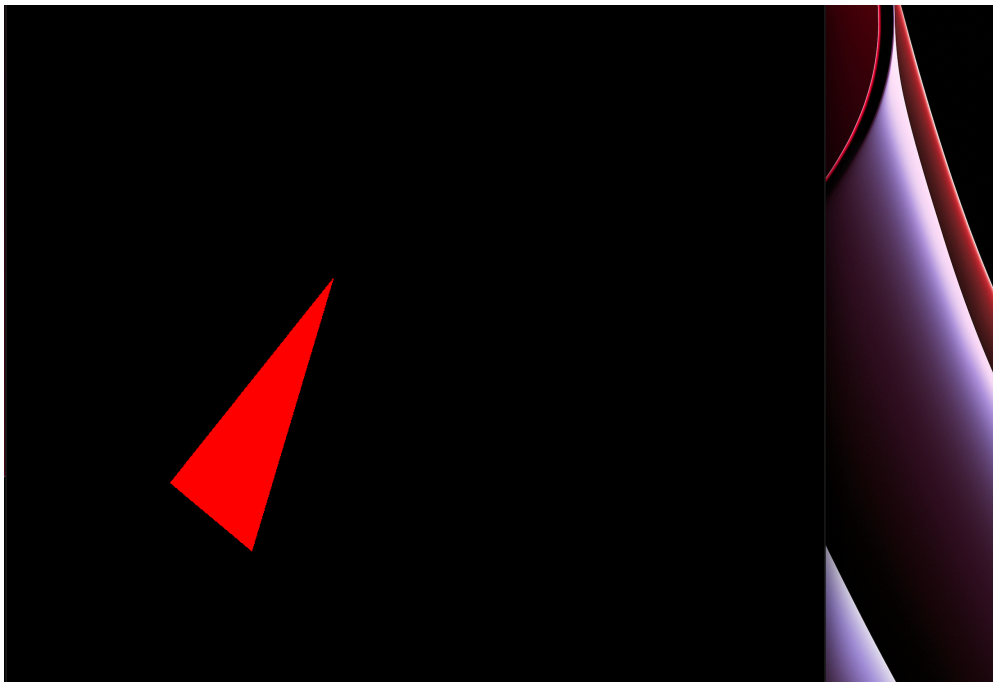


Question-4:Draw a triangle on black background

```
// C++ program for the above approach
#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#define pi 3.142857
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();

}
```



Question-5:Draw a rectangle on black background

```
// C++ program for the above approach

#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#define pi 3.142857

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, 200.0);
   glVertex2i(100.0, 200.0);
   glVertex2i(100.0, 150.0);
   glVertex2i(50.0, 150.0);
   glEnd();

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

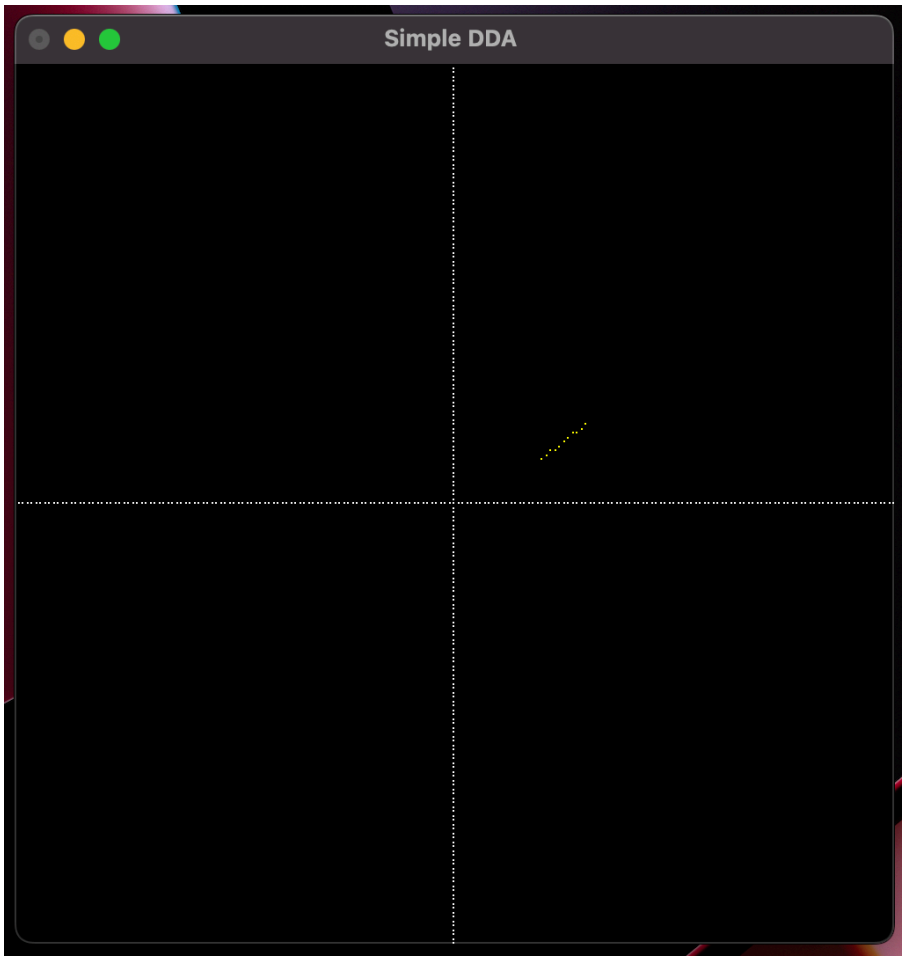
```
}

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();

                                                                    }
```

Ques 6- Simple DDA

```
#include<GLUT/GLUT.h>
#include<OpenGL/OpenGL.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.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("%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;
}
```

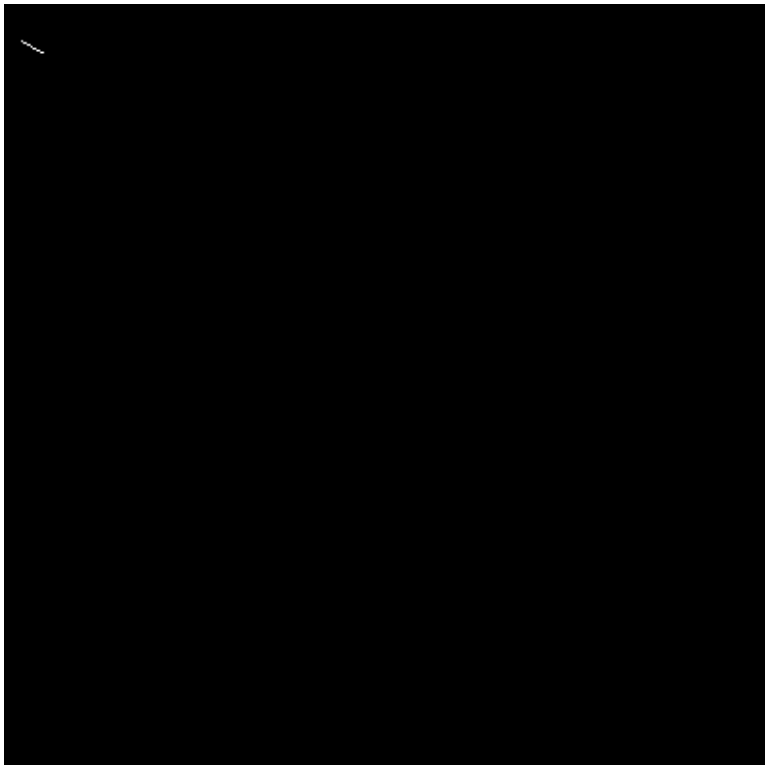

Simple DDA

```
#include <GLUT/GLUT.h>
#include <OpenGL/OpenGL.h>
#include <iostream>
#include <math.h>
using namespace std;

int X1, Y1, X2, Y2;

void display()
{
    int dx = X2 - X1;
    int dy = Y2 - Y1;

    if (dx < 0)
        dx = -dx;
    if (dy < 0)
        dy = -dy;

    int inc_x = X1 < X2 ? 1 : -1;
    int inc_y = Y1 < Y2 ? 1 : -1;

    int x = X1;
    int y = Y1;

    cout << x << " " << y << endl;

    if (dx > dy)
    {
```

```
    int d = 2 * dy - dx;
    int d1 = 2 * dy;
    int d2 = 2 * (dy - dx);

    glBegin(GL_POINTS);
    while (x != X2)
    {
        cout << x << ", " << y << endl;
        glVertex2i(x, y);
        x += inc_x;
        if (d > 0)
        {
            d += d2;
        }
        else
        {
            d += d1;
            y += inc_y;
        }
    }
    glEnd();
}
else
{
    int d = 2 * dx - dy;
    int d1 = 2 * dx;
    int d2 = 2 * (dx - dy);

    glBegin(GL_POINTS);
```

```
        while (y != Y2)
        {
            cout << x << ", " << y << endl;
            glVertex2i(x, y);
            y += inc_y;
            if (d > 0)
            {
                d += d2;
            }
            else
            {
                d += d1;
                x += inc_x;
            }
        }
        glEnd();
    }

    glFlush();
}

void init()
{
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(100, 200);
    glutCreateWindow("bresenhems Line Algorithm");
    gluOrtho2D(0, 400, 400, 0);
}
```

```
int main(int argc, char** argv)
{
    printf("Enter coordinates of first point: ");
    scanf("%d%d", &X1, &Y1);
    printf("\nEnter coordinates of second point: ");
    scanf("%d%d", &X2, &Y2);
    glutInit(&argc, argv);
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

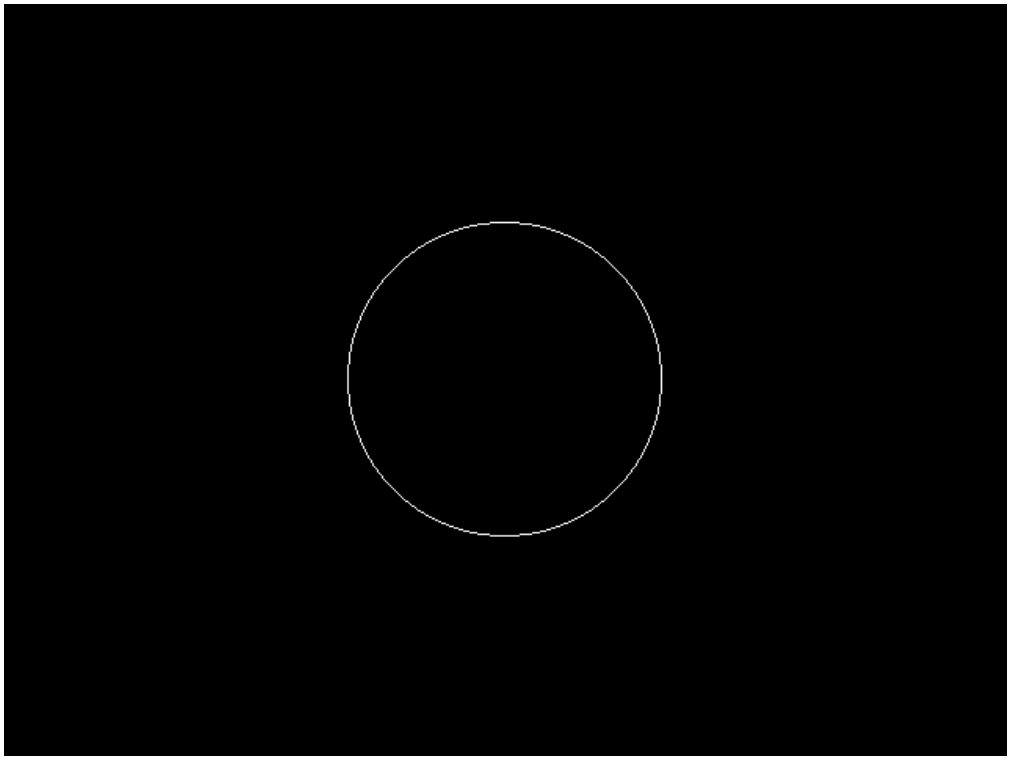


Ques 7-Circle using midpoint algorithm

```
#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0, 640, 0, 480);
}

// Draw a point
void drawPixel(int x, int y)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}

void midpointCircle(int xc, int yc, int r)
{
    int x = 0;
    int y = r;
    int p = 1 - r;
    drawPixel(xc + x, yc + y);
    drawPixel(xc - x, yc + y);
```

```
    drawPixel(xc + x, yc - y);
    drawPixel(xc - x, yc - y);
    drawPixel(xc + y, yc + x);
    drawPixel(xc - y, yc + x);
    drawPixel(xc + y, yc - x);
    drawPixel(xc - y, yc - x);

    while (x < y)
    {
        x++;
        if (p < 0)
        {
            p += 2 * x + 1;
        }
        else
        {
            y--;
            p += 2 * (x - y) + 1;
        }
        drawPixel(xc + x, yc + y);
        drawPixel(xc - x, yc + y);
        drawPixel(xc + x, yc - y);
        drawPixel(xc - x, yc - y);
        drawPixel(xc + y, yc + x);
        drawPixel(xc - y, yc + x);
        drawPixel(xc + y, yc - x);
        drawPixel(xc - y, yc - x);
    }
}
```

```
// Display Function
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 1.0, 1.0);
    midpointCircle(320, 240, 100);

    glFlush();
}

// Main Function
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Midpoint Circle Algorithm using OpenGL");
    init();
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```



Ques 8- Ellipse using Midpoint Algorithm

```
#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#include<stdio.h>
#include<math.h>
#include <iostream>
#include <GLUT/glut.h>
#include <cmath>
using namespace std;
int xc, yc, rx, ry;

void plot(int x, int y) {
    glBegin(GL_POINTS);
    glVertex2i(xc + x, yc + y);
    glVertex2i(xc + x, yc - y);
    glVertex2i(xc - x, yc + y);
    glVertex2i(xc - x, yc - y);
    glEnd();
}

void midpointEllipse() {
    int x = 0, y = ry;
    double d1 = ry * ry - rx * rx * ry + rx * rx / 4.0;
    plot(x, y);

    while (rx * rx * (y - 0.5) > ry * ry * (x + 1)) {
        if (d1 < 0) {
            d1 += ry * ry * (2 * x + 3);
        } else {
            d1 += ry * ry * (2 * x + 3) + rx * rx * (-2 * y + 2);
            y--;
        }
        x++;
        plot(x, y);
    }

    double d2 = ry * ry * (x + 0.5) * (x + 0.5) + rx * rx * (y - 1) * (y - 1) - rx * rx * ry * ry;
    while (y > 0) {
        if (d2 < 0) {
            d2 += ry * ry * (2 * x + 2) + rx * rx * (-2 * y + 3);
            x++;
        } else {
            d2 += rx * rx * (-2 * y + 3);
        }
        y--;
        plot(x, y);
    }
}
```

```
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    midpointEllipse();
    glFlush();
}

void init() {
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glColor3f(1.0, 1.0, 1.0);
    glPointSize(2.0);
    gluOrtho2D(0, 640, 0, 480);
}

int main(int argc, char **argv) {
    cout << "Enter the coordinates of the center of the ellipse: ";
    cin >> xc >> yc;
    cout << "Enter the major and minor axes of the ellipse: ";
    cin >> rx >> ry;

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Midpoint Ellipse Algorithm");
    init();
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```