#include <GL/glut.h>

void drawShapes()

{

glLineWidth(3);

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw line

glColor3f(1.0, 0.0, 0.0); // Red

glBegin(GL\_LINES);

glVertex2f(-0.8, 0.05);

glVertex2f(0.0, -0.4);

glEnd();

// Draw a rectangle

glColor3f(0.0, 1.0, 0.0); // Green

glBegin(GL\_LINE\_LOOP);

glVertex2f(-0.9, 0.1); // Bottom-left vertex

glVertex2f(-0.5, 0.1); // Bottom-right vertex

glVertex2f(-0.5, 0.5); // Top-right vertex

glVertex2f(-0.9, 0.5); // Top-left vertex

glEnd();

// Draw a triangle

glColor3f(0.0, 0.0, 1.0); // Blue

glBegin(GL\_LINE\_LOOP);

glVertex2f(-0.1,0.1);

glVertex2f(-0.4, 0.1);

glVertex2f(0.25, 0.4);

glEnd();

// Draw a hexagon

glColor3f(1.0, 1.0, 0.0); // Yellow

glBegin(GL\_LINE\_LOOP);

glVertex2f(0.1, 0.0); // Bottom-left vertex

glVertex2f(0.35, 0.5); // Top-left vertex

glVertex2f(0.65, 0.5); // Top vertex

glVertex2f(0.9, 0.0); // Top-right vertex

glVertex2f(0.65, -0.5); // Bottom-right vertex

glVertex2f(0.35, -0.5); // Bottom vertex

glEnd();

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(800, 600); // Increase window size

glutInitWindowPosition(100, 100);

glutCreateWindow("Bisan Basnet");

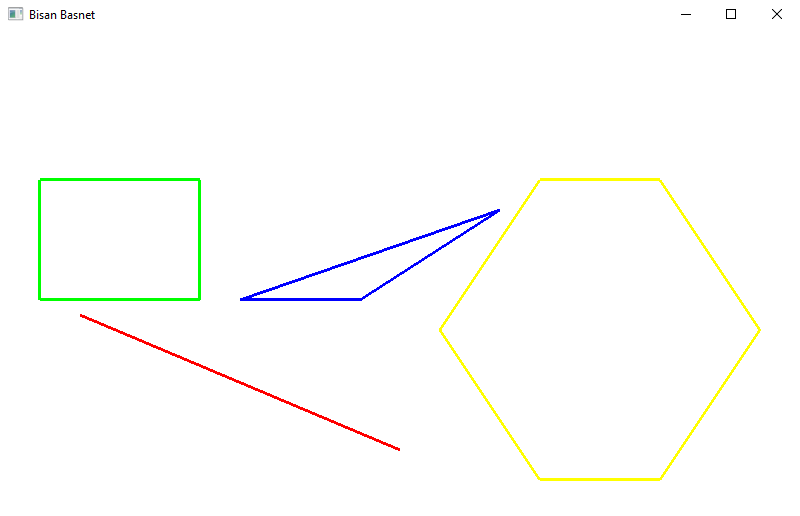
glClearColor(1.0, 1.0, 1.0, 1.0); // Set clear color to black

glutDisplayFunc(drawShapes);

glutMainLoop();

return 0;

}



#include <GL/glut.h>

void drawShapes()

{

glLineWidth(3);

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw a rectangle

glColor3f(1.0, 0.0, 0.0); // red

glBegin(GL\_POLYGON);

glVertex2f(-0.6, -0.4); // Bottom-left vertex

glVertex2f(-0.1, -0.4); // Bottom-right vertex

glVertex2f(-0.1, 0.1); // Top-right vertex

glVertex2f(-0.6, 0.1); // Top-left vertex

glEnd();

// Draw a triangle

glColor3f(0.0, 0.0, 1.0); // Blue

glBegin(GL\_TRIANGLES);

glVertex2f(-0.1f, -0.2f);

glVertex2f(-0.4f, -0.6f);

glVertex2f(0.2f, -0.6f);

glEnd();

glPopMatrix();

// Draw a hexagon

glColor3f(0.0, 1.0, 0.0); // Yellow

glBegin(GL\_POLYGON);

glVertex2f(0.1, -0.9); // Bottom-left vertex

glVertex2f(-0.1, -0.7); // Top-left vertex

glVertex2f(0.1, -0.478); // Top vertex

glVertex2f(0.4, -0.478); // Top-right vertex

glVertex2f(0.6,-0.7); // Bottom-right vertex

glVertex2f(0.4, -0.9); // Bottom vertex

glEnd();

glPopMatrix();

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(800, 600); // Increase window size

glutInitWindowPosition(100, 100);

glutCreateWindow("Bisan Basnet");

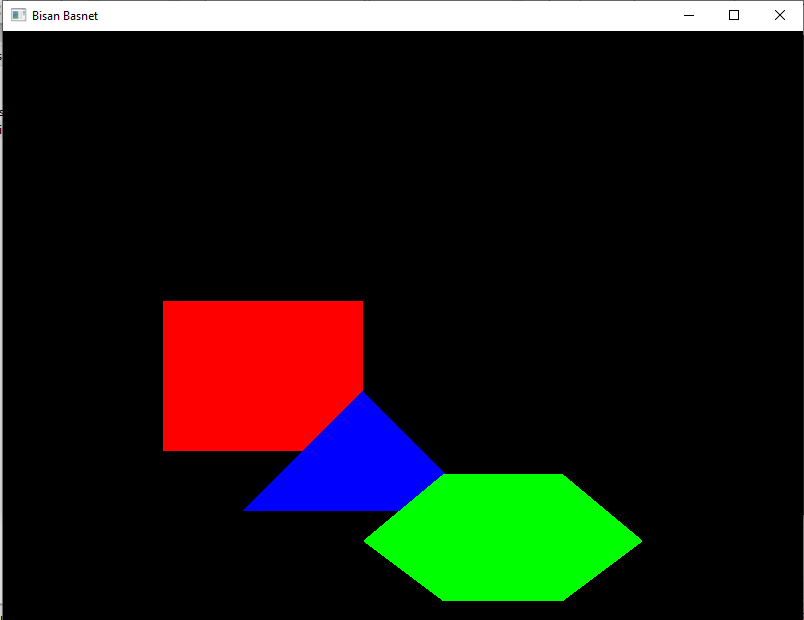
glClearColor(0.0, 0.0, 0.0, 1.0); // Set clear color to black

glutDisplayFunc(drawShapes);

glutMainLoop();

return 0;

}



#include <GL/glut.h>

int x\_max=700,y\_max=600;

int x\_position=100;

int y\_position=100;

int xstepsize=10,ystepsize=10;

int width=100;

int height=100;

void myInit(void)

{

glutInitWindowSize(x\_max,y\_max);

glutInitWindowPosition(100,100);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("Bisan Basnet");

glClearColor(1,1,1,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(0,0,0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,640,0,640);

glMatrixMode(GL\_MODELVIEW);

}

void updatePolygon(int v)

if(x\_position+width>=x\_max || x\_position<0)

xstepsize=-xstepsize;

if(y\_position+height>=y\_max || y\_position<0)

ystepsize=-ystepsize;

x\_position+=xstepsize;

y\_position+=ystepsize;

glutPostRedisplay();

glutTimerFunc(10,updatePolygon,0);

}

void drawRectangle()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int px1=x\_position,py1=y\_position;

int px2=x\_position+width,py2=y\_position;

int px3=x\_position+width,py3=y\_position+height;

int px4=x\_position,py4=y\_position+height;

glColor3d(1,0,0);

glBegin(GL\_POLYGON);

glVertex2f(px1,py1);

glVertex2f(px2,py2);

glVertex2f(px3,py3);

glVertex2f(px4,py4);

glEnd();

glFlush();

}

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

{

glutInit(&argc,argv);

myInit();

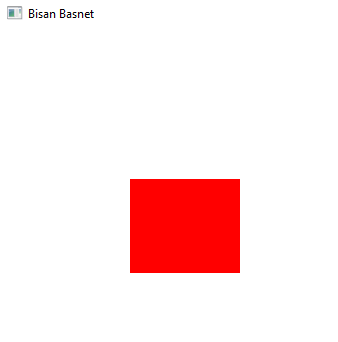
glutDisplayFunc(drawRectangle);

glutTimerFunc(0,updatePolygon,0);

glutMainLoop();

return 0;

}



#include <GL/glut.h>

int windowWidth = 800;

int windowHeight = 600;

int squareSize = 70;

int mouseX = 0;

int mouseY = 0;

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0f, 0.0f, 0.0f); // Red square

glBegin(GL\_POLYGON);

glVertex2f(mouseX, windowHeight - mouseY);

glVertex2f(mouseX + squareSize, windowHeight - mouseY);

glVertex2f(mouseX + squareSize, windowHeight - mouseY + squareSize);

glVertex2f(mouseX, windowHeight - mouseY + squareSize);

glEnd();

glutSwapBuffers();

void mouse(int button, int state, int x, int y)

{

mouseX = x - 2;

mouseY = y - 2;

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutCreateWindow("Bisan Basnet");

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, windowWidth, 0, windowHeight);

glMatrixMode(GL\_MODELVIEW);

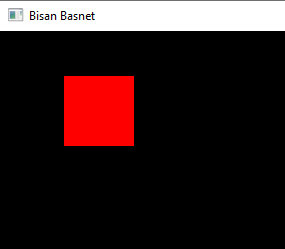
glutDisplayFunc(display);

glutMouseFunc(mouse);

glutMainLoop();

return 0;

}



#include <GL/glut.h>

void drawHut()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw the roof of the hut

glColor3f(0.3490f, 0.5098f, 0.6902f); // Light blue color

glBegin(GL\_TRIANGLES);

glVertex2f(230.0f, 350.0f); // Bottom-left vertex

glVertex2f(570.0f, 350.0f); // Bottom-right vertex

glVertex2f(400.0f, 500.0f); // Top vertex

glEnd();

// Draw the walls of the hut

glColor3f(0.5019f, 0.8314f, 0.1059f); // Red color

glBegin(GL\_QUADS);

glVertex2f(230.0f, 50.0f); // Bottom-left vertex

glVertex2f(230.0f, 350.0f); // Top-left vertex

glVertex2f(570.0f, 350.0f); // Top-right vertex

glVertex2f(570.0f, 50.0f); // Bottom-right vertex

glEnd();

// Draw the door of the hut

glColor3f(0.3490f, 0.5098f, 0.6902f); // Light blue color

glBegin(GL\_QUADS);

glVertex2f(350.0f, 60.0f); // Bottom-left vertex

glVertex2f(350.0f, 185.0f); // Top-left vertex

glVertex2f(450.0f, 185.0f); // Top-right vertex

glVertex2f(450.0f, 60.0f); // Bottom-right vertex

glEnd();

// Draw the first window of the hut

glColor3f(0.9412f, 0.0471f, 0.0471f); // Red color

glBegin(GL\_QUADS);

glVertex2f(430.0f, 200.0f); // Bottom-right vertex

glVertex2f(430.0f, 280.0f); // Top-right vertex

glVertex2f(510.0f, 280.0f); // Top-left vertex

glVertex2f(510.0f, 200.0f); // Bottom-left vertex

glEnd();

// Draw the second window of the hut

glBegin(GL\_QUADS);

glVertex2f(290.0f, 200.0f); // Bottom-right vertex

glVertex2f(290.0f, 280.0f); // Top-right vertex

glVertex2f(370.0f, 280.0f); // Top-left vertex

glVertex2f(370.0f, 200.0f); // Bottom-left vertex

glEnd();

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(800, 600);

glutCreateWindow("Hut-Bisan Basnet");

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 800, 0, 600);

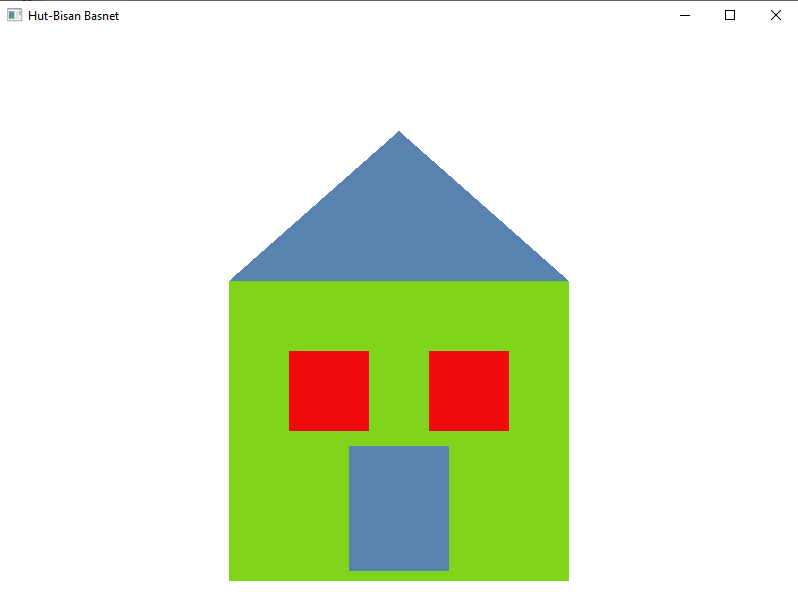
glMatrixMode(GL\_MODELVIEW);

glutDisplayFunc(drawHut);

glutMainLoop();

return 0;

}



#include <GL/glut.h>

int windowWidth = 1000;

int windowHeight = 600;

void plotPoint()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0f, 0.0f, 0.0f);

glPointSize(6.0f);

glBegin(GL\_POINTS);

for (int i = 0; i < 50; i++)

{

glVertex2f(rand() % windowWidth, rand() % windowHeight);

}

glEnd();

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutCreateWindow("Draw 50 Random Points -Bisan Basnet");

glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, windowWidth, 0, windowHeight);

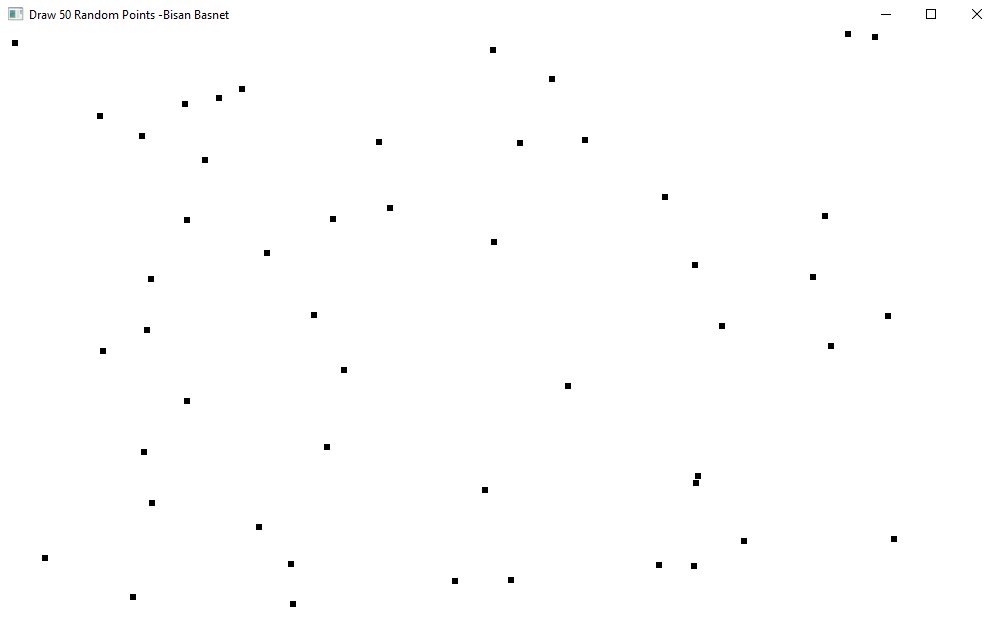
glMatrixMode(GL\_MODELVIEW);

glutDisplayFunc(plotPoint);

glutMainLoop();

return 0;

}



#include <GL/glut.h>

#include <iostream>

#include <cmath>

using namespace std;

void plotPixel()

{

int xi, yi, xf, yf, dx,dy,inc;

float x, y,m,c;

int step,i=0;

cout << "Enter the initial points: ";

cin >> xi >> yi;

cout << "Enter the final points: ";

cin >> xf >> yf;

dx = xf - xi;

dy = yf - yi;

m = dy/dx;

c = yi - m\*xi;

x=xi;

y=yi;

glBegin(GL\_POINTS);

glVertex2f(round(x), round(y));

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

step=abs(dx);

inc=dx/step;

while(i<step){

x+=inc;

y=m\*x+c;

glVertex2f(round(x), round(y));

i++;

}

}

else if(abs(dy)>=abs(dx)){

step=abs(dy);

inc=dy/step;

while(i<step){

y+=inc;

x=(y-c)/m;

glVertex2f(round(x), round(y));

i++;

}

}

glEnd();

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(800, 600);

glutInitWindowPosition(100, 100);

glutCreateWindow("Bisan Basnet"); //give your own name

glClearColor(0.0, 0.0, 0.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

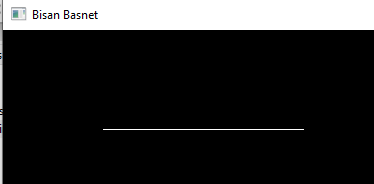
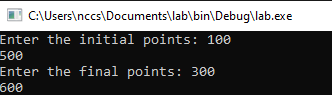
gluOrtho2D(0, 800, 0, 600);

glutDisplayFunc(plotPixel);

glutMainLoop();

return 0;

}



#include <GL/glut.h>

#include <iostream>

#include <cmath>

using namespace std;

void plotPixel()

{

int xi, yi, xf, yf, dx,dy;

float x,y,xinc,yinc;

int step,i=0;

cout << "Enter the initial points: ";

cin >> xi >> yi;

cout << "Enter the final points: ";

cin >> xf >> yf;

dx = xf - xi;

dy = yf - yi;

x=xi;

y=yi;

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

step=abs(dx);

}

else

{

step=abs(dy);

}

xinc=dx/step;

yinc=dy/step;

glBegin(GL\_POINTS);

glVertex2f(round(x), round(y));

while(i<step)

{

x+=xinc;

y+=yinc;

glVertex2f(round(x), round(y));

i++;

}

glEnd();

glFlush();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(800, 600);

glutInitWindowPosition(100, 100);

glutCreateWindow("Bisan Basnet");

glClearColor(0.0, 0.0, 0.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

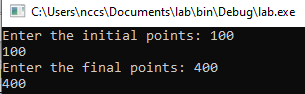
gluOrtho2D(0, 800, 0, 600);

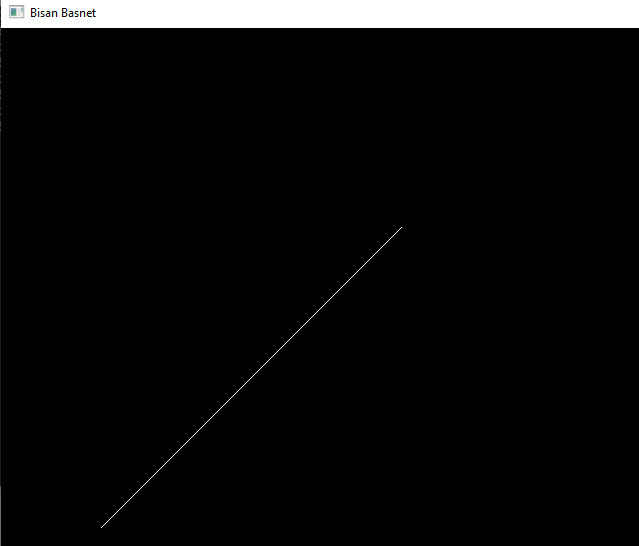
glutDisplayFunc(plotPixel);

glutMainLoop();

return 0;

}





#include <GL/glut.h>

#include <iostream>

#include <ctime>

#include <cmath>

using namespace std;

float xi, yi, xf, yf;

void bresenham()

{

float x, y, dx, dy, p, xend;

dx = xf - xi;

dy = yf - yi;

if (dx < 0)

{

dx = -dx;

}

if (dy < 0)

{

dy = -dy;

}

if (xi > xf)

{

x = xf;

y = yf;

xend = xi;

}

else

{

x = xi;

y = yi;

xend = xf;

}

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_POINTS);

glVertex2f(x, y);

if (dx < dy)

{

p = 2 \* dx - dy;

while (x <= xend)

{

if (p < 0)

{

p += 2 \* dx;

}

else

{

p += 2 \* (dx - dy);

y += (yf > yi) ? 1 : -1;

}

x += 1;

glVertex2f(x, y);

}

}

else

{

p = 2 \* dy - dx;

while (x <= xend)

{

if (p < 0)

{

p += 2 \* dy;

}

else

{

p += 2 \* (dy - dx);

x += (xf > xi) ? 1 : -1;

}

y += 1;

glVertex2f(x, y);

}

}

glEnd();

glFlush();

}

void dda()

{

int steps;

int dx = xf - xi;

int dy = yf - yi;

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

{

steps = abs(dx);

}

else

{

steps = abs(dy);

}

float xIncrement = (float)dx / (float)steps;

float yIncrement = (float)dy / (float)steps;

float x = xi;

float y = yi;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_POINTS);

glVertex2f(round(x), round(y));

for (int i = 0; i <= steps; i++)

{

x += xIncrement;

y += yIncrement;

glVertex2f(round(x), round(y));

}

glEnd();

glFlush();

}

void plotPixel()

{

clock\_t dda\_start, dda\_end, bresenham\_start, bresenham\_end, direct\_start, direct\_end;

dda\_start = clock();

dda();

dda\_end = clock();

bresenham\_start = clock();

bresenham();

bresenham\_end = clock();

cout << endl;

cout << "The time taken by DDA is " << (float)(dda\_end - dda\_start) / CLOCKS\_PER\_SEC << " seconds" << endl;

cout << "The time taken by Bresenham is " << (float)(bresenham\_end - bresenham\_start) / CLOCKS\_PER\_SEC << " seconds" << endl;

}

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

{

cout << "Enter the initial points: ";

cin >> xi >> yi;

cout << "Enter the final points: ";

cin >> xf >> yf;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(800, 600);

glutInitWindowPosition(100, 100);

glutCreateWindow("Bresenham Algorithm - Bisan Basnet");

glClearColor(1.0, 1.0, 1.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 800, 0, 600);

glutDisplayFunc(plotPixel);

glutMainLoop();

return 0;

}

