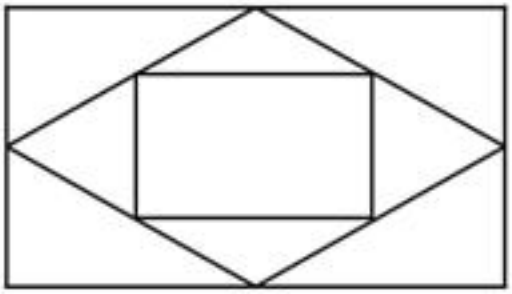
/
*==========
==========
Assignment No: 1
Batch : H10
Roll No : 23257
Date :
Aim : To implement line drawing algorithms

Problem Statement:

Draw the following pattern using any Line drawing algorithms.



==========
==========*/

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

float X1,Y1,X2,Y2;

int Abs(int x) {

	if(x<0)
		return (x*(-1));
	else
		return x;
}

int sign(int x) {

	if(x==0)
		return 0;

	else if(x<0)
		return -1;

	else return 1;
}
```

```
void SWAP(int *x,int *y) {

        int temp=*x;
        *x=*y;
        *y=temp;
}

void DDA(float x1,float y1,float x2,float y2) {

        float dx,dy,steps,x,y,i,Xinc,Yinc;

        dx=x2-x1;
        dy=y2-y1;

        if(Abs(dx)>=Abs(dy))
                steps=Abs(dx);
        else
                steps=Abs(dy);

        Xinc=dx/steps;
        Yinc=dy/steps;

        x=x1;
        y=y1;

        glBegin(GL_POINTS);
                glVertex2i(x,y);
        glEnd();

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

                x=x+Xinc;
                y=y+Yinc;

   glBegin(GL_POINTS);
        glVertex2i(x,y);
   glEnd();
  }
}

void Bresenham(int x1,int y1,int x2,int y2) {

        int dx,dy,steps=1,x,y,i,P,sw=0,s1,s2;

        x=x1;
        y=y1;

        dx=Abs(x2-x1);
        dy=Abs(y2-y1);

        s1=sign(x2-x1);
        s2=sign(y2-y1);

        if(dy>dx) {
```

```
                SWAP(&dx,&dy);
                sw=1;
        }

        P=(2*dy)-dx;

        glBegin(GL_POINTS);
                glVertex2i(x,y);
        glEnd();

        while(steps<=dx) {

                if(P>=0) {

                        x=x+s1;
                        y=y+s2;
                        P=P+2*(dy-dx);
                }

                else {

                        if(sw==1) y=y+s2;
                        else      x=x+s1;

                        P=P+2*dy;
                }
                glBegin(GL_POINTS);
        glVertex2i(x,y);
    glEnd();
    steps++;
  }
}

void GeneratePolygonUsingDDA() {

  int x1,y1,x2,y2,x3,y3,x4,y4,temp1,temp2;

        x1=X1;
  y1=Y1;

        x2=X1;
        y2=Y2;

        x3=X2;
        y3=Y2;

        x4=X2;
        y4=Y1;

        glColor3f(0.0f, 0.5f, 0.5f);

  DDA(x1,y1,x2,y2);
        DDA(x2,y2,x3,y3);
```

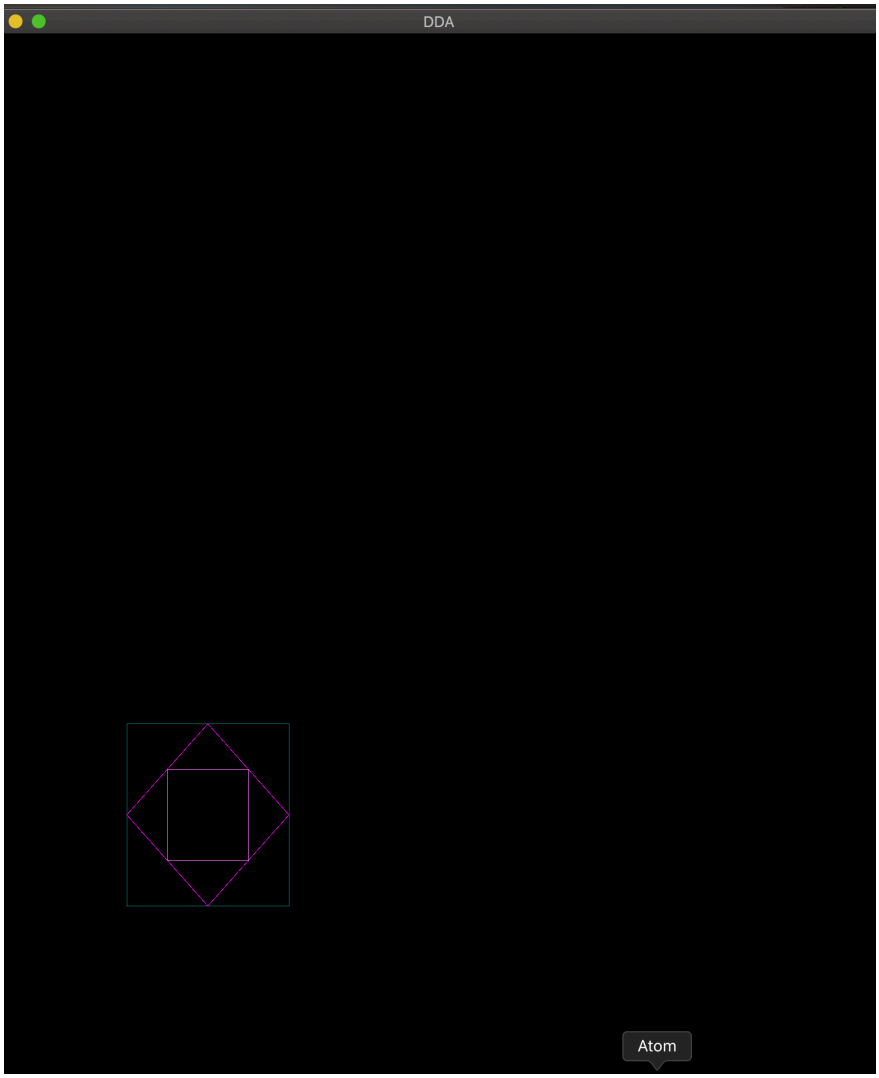
```
        DDA(x3,y3,x4,y4);
        DDA(x4,y4,x1,y1);

        temp1=x1;
        temp2=y1;

        x1=(x1+x2)/2;
        y1=(y1+y2)/2;

        x2=(x2+x3)/2;
        y2=(y2+y3)/2;

        x3=(x3+x4)/2;
        y3=(y3+y4)/2;

        x4=(temp1+x4)/2;
        y4=(temp2+y4)/2;

        glColor3f(1.0f, 0.0f, 1.0f);

  DDA(x1,y1,x2,y2);
        DDA(x2,y2,x3,y3);
        DDA(x3,y3,x4,y4);
        DDA(x4,y4,x1,y1);

        temp1=x1;
        temp2=y1;

        x1=(x1+x2)/2;
        y1=(y1+y2)/2;

        x2=(x2+x3)/2;
        y2=(y2+y3)/2;

        x3=(x3+x4)/2;
        y3=(y3+y4)/2;

        x4=(temp1+x4)/2;
        y4=(temp2+y4)/2;

        glColor3f(2.0f, 0.5f, 1.0f);

  DDA(x1,y1,x2,y2);
        DDA(x2,y2,x3,y3);
        DDA(x3,y3,x4,y4);
        DDA(x4,y4,x1,y1);

  glFlush();
}

void GeneratePolygonUsingBresenham() {

  int x1,y1,x2,y2,x3,y3,x4,y4,temp1,temp2;
```

```
        x1=(int)X1;
y1=(int)Y1;

        x2=(int)X1;
        y2=(int)Y2;

        x3=(int)X2;
        y3=(int)Y2;

        x4=(int)X2;
        y4=(int)Y1;

        glColor3f(0.0f, 0.5f, 0.5f);

Bresenham(x1,y1,x2,y2);
        Bresenham(x2,y2,x3,y3);
        Bresenham(x3,y3,x4,y4);
        Bresenham(x4,y4,x1,y1);

        temp1=x1;
        temp2=y1;

        x1=(x1+x2)/2;
        y1=(y1+y2)/2;

        x2=(x2+x3)/2;
        y2=(y2+y3)/2;

        x3=(x3+x4)/2;
        y3=(y3+y4)/2;

        x4=(temp1+x4)/2;
        y4=(temp2+y4)/2;

        glColor3f(1.0f, 0.0f, 1.0f);

Bresenham(x1,y1,x2,y2);
        Bresenham(x2,y2,x3,y3);
        Bresenham(x3,y3,x4,y4);
        Bresenham(x4,y4,x1,y1);

        temp1=x1;
        temp2=y1;

        x1=(x1+x2)/2;
        y1=(y1+y2)/2;

        x2=(x2+x3)/2;
        y2=(y2+y3)/2;

        x3=(x3+x4)/2;
        y3=(y3+y4)/2;

        x4=(temp1+x4)/2;
```

```
        y4=(temp2+y4)/2;

        glColor3f(2.0f, 0.5f, 1.0f);

  Bresenham(x1,y1,x2,y2);
        Bresenham(x2,y2,x3,y3);
        Bresenham(x3,y3,x4,y4);
        Bresenham(x4,y4,x1,y1);

  glFlush();
}

void init() {

        glClearColor(0.0, 0.0, 0.0, 0.0);
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0, 1.0, 1.0);

  glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
  gluOrtho2D(-1000,1000,-1000,1000);
}

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

        int ch,flag=0;

        do {

          cout<<"==================================\n";
          cout<<"Select Correct Option\n";
          cout<<"==================================\n";
          cout<<"1. Enter Coordinates\n";
          cout<<"2. Generate Polygon by DDA\n";
          cout<<"3. Generate Polygon by Bresenham\n";
          cout<<"4. Exit\n";
          cout<<"Your Chice: ";
          cin>>ch;
          cout<<"==================================\n";

    if(ch==1 || flag==1 || ch==4) {

            switch(ch) {

              case 1: {

                cout<<"Starting Coordinates:\n";
                cout<<"X1: ";
                cin>>X1;
                cout<<"Y1: ";
                cin>>Y1;
                cout<<"End Coordinates:\n";
                cout<<"X2: ";
                cin>>X2;
```

```
                cout<<"Y2: ";
                cin>>Y2;
                flag=1;
                break;
               }

               case 2: {

                glutInit(&argc, argv);
                glutInitDisplayMode(GLUT_SINGLE);
                glutInitWindowSize(800,900);
                glutInitWindowPosition(320,240);
                glutCreateWindow("DDA");
                init();
                glutDisplayFunc(GeneratePolygonUsingDDA);
                glutMainLoop();
                break;
               }

               case 3: {

                glutInit(&argc, argv);
                glutInitDisplayMode(GLUT_SINGLE);
                glutInitWindowSize(800,900);
                glutInitWindowPosition(320,240);
                glutCreateWindow("Bresenham");
                init();
                glutDisplayFunc(GeneratePolygonUsingBresenham);
                glutMainLoop();
                break;
               }

               case 4: cout<<"End\n";
                     break;

              default: cout<<"Select correct Option\n\n";
             }
   }
          else cout<<"Please Accept Coordinates First\n\n";
         }
         while(ch!=4);
  return 0;
}

/*

Output:

Select Correct Option
================================
1. Enter Coordinates
2. Generate Polygon by DDA
3. Generate Polygon by Bresenham
4. Exit
```

```
Your Chice: 1
==================================
Starting Coordinates:
X1: -350
Y1: -350
End Coordinates:
X2: 350
Y2: 350
==================================
Select Correct Option
==================================
1. Enter Coordinates
2. Generate Polygon by DDA
3. Generate Polygon by Bresenham
4. Exit
Your Chice: 2
==================================
```




```
*/
```