**Develop a program to draw a line using DDA line drawing technique**

#include<stdio.h>

#include<GL/glut.h>

#include<math.h>

//int x1,y1,x2,y2;

int n;

int endpoints[8][4];

void draw\_pixel(int x,int y){

glColor3f(0.0,0.0,1.0);

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

}

void draw\_line(int x1,int y1,int x2,int y2){

int dx = x2 - x1;

int dy = y2 - y1;

int i=0;int steps=0;

// calculate steps required for generating pixels

steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);

// calculate increment in x & y for each steps

float Xinc = dx / (float) steps;

float Yinc = dy / (float) steps;

// Put pixel for each step

float X = x1;

float Y = y1;

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

{

draw\_pixel(round(X),round(Y)); // put pixel at (X,Y)

X += Xinc; // increment in x at each step

Y += Yinc; // increment in y at each step

//delay(100); // for visualization of line-

// generation step by step

}

}

void drawText(int x,int y,char\* s){

int i=0;

glColor3f(1.0,0.0,0.0);

glRasterPos2i(x,y);

for(i=0;s[i]!='\0';i++){

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,s[i]);

}

}

void DDA(){

int i;

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

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

//x1=endpoints[i][0];y1=endpoints[i][1];x2=endpoints[i][2];y2=endpoints[i][3];

draw\_line(endpoints[i][0],endpoints[i][1],endpoints[i][2],endpoints[i][3]);

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINES);

glVertex2i(endpoints[i][0],endpoints[i][1]);

glVertex2i(endpoints[i][2],endpoints[i][3]);

glEnd();

//printf("The points are %d %d %d %d\n",x1,y1,x2,y2);

glFlush();

}

drawText(220,475,"DDA Line drawing algorithm");

glFlush();

}

void myInit(){

//glClearColor(1.0,1.0,1.0,1.0);

//glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

glMatrixMode(GL\_MODELVIEW);

}

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

int i;

printf("Enter number of endpoints:\n");

scanf("%d",&n);

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

printf("Enter the endpoints of line %d\n",i+1);

scanf("%d%d%d%d",&endpoints[i][0],&endpoints[i][1],&endpoints[i][2],&endpoints[i][3]);

}

//printf("Ener the endpoints of line segment");

//scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowPosition(100,100);

glutInitWindowSize(500,500);

glutCreateWindow("DDA line drawing");

glutDisplayFunc(DDA);

myInit();

glutMainLoop();

}

**Develop a program to draw a line using Bresenham’s line drawing technique**

#include<stdio.h>

#include<GL/glut.h>

int x1,y1,x2,y2;

void draw\_pixel(int x, int y)

{ glColor3f(0.0,0.0,1.0);

glPointSize(5);

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

}

void Bresenham()

{ glClear(GL\_COLOR\_BUFFER\_BIT);

glClearColor(1.0,1.0,1.0,1.0);

draw\_line(x1,y1,x2,y2);

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINES);

glVertex2i(x1,y1);

glVertex2i(x2,y2);

glEnd();

glFlush();

}

void myinit()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,100,0,100);

glMatrixMode(GL\_MODELVIEW);

}

void draw\_line(int x1,int y1, int x2, int y2)

{

int dx, dy, i, e;

int incx, incy, inc1, inc2;

int x,y;

dx = x2-x1;

dy = y2-y1;

if (dx < 0) dx = -dx;

if (dy < 0) dy = -dy;

incx = 1;

if (x2 < x1) incx = -1;

incy = 1;

if (y2 < y1) incy = -1;

x = x1; y = y1;

if (dx > dy) //slope lessthan 1

{

draw\_pixel(x, y);

e = 2 \* dy-dx;// initial decision parameter

inc1 = 2\*(dy-dx);//upper pixel

inc2 = 2\*dy; // lower pixel

for (i=0; i<dx; i++)

{

if (e >= 0)

{y += incy;

e += inc1;} //selection of upper pixel

else

e += inc2; //selection of Lower pixel

x += incx;

draw\_pixel(x, y);

}

}

else

{

draw\_pixel(x, y);

e = 2\*dx-dy;

inc1 = 2\*(dx-dy);

inc2 = 2\*dx;

for (i=0; i<dy; i++)

{

if (e >= 0)

{

x+= incx; //upper pixel

e += inc1;

}

else

e += inc2;

y += incy;

draw\_pixel(x, y);

}

}

}

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

{ printf("Enter the endpoints of the line segment");

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(100,100);

glutCreateWindow("Bresenham Line Algorithm");

glutDisplayFunc(Bresenham);

myinit();

glutMainLoop();

}