NAME : VAIBHAV DNYANESHWAR MOHITE

ROLLNO: 25045

CLASS: SE IT A

# ASSIGNMENT NO - 2

CODE:

#include<iostream>

#include<GL/glut.h>

#include <math.h> #define ROUND(X)((int)X+0.5) using namespace std; static int menu\_id; static int submenu\_id1; static int submenu\_id2; int w = 640; int h = 480; void DDA\_simple(float,float,float,float); void DDA\_dashed(float, float, float, float); void DDA\_dotted(float, float, float, float); void DDA\_solid(float, float, float, float); void BH\_dotted(float, float, float, float); void BH\_dashed(float, float, float, float); void BH\_simple(float, float, float, float); void BH\_solid(float, float, float, float); void init(void) {

glClearColor(1.0, 1.0, 1.0, 0); //clear buffer glColor3f(0.0, 0.0, 0.0); //set background colour glPointSize(4.0);

gluOrtho2D(-w / 2, w / 2, -h / 2, h / 2); //dividing screen into 4 quadrants

} void draw() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINES); glVertex2d(-w / 2, 0); glVertex2d(w / 2, 0); glVertex2d(0, -h / 2); glVertex2d(0, h / 2); glEnd(); glFlush();

}

void DDA\_simple(float x1,float y1,float x2,float y2)

{

float dx, dy, X, Y,steps, xinc, yinc;

int i = 0; dx = x2 - x1; dy = y2 - y1; if (abs(dx) >= abs(dy))

{

steps = abs(dx);

} else steps = abs(dy); } xinc = dx / steps; yinc = dy / steps;

1. = x1; //assign initial values(starting coordinates)
2. = y1;

glPointSize(4.0); glBegin(GL\_POINTS);

glVertex2f(ROUND(X), ROUND(Y)); glEnd(); i = 1; while (i <= steps)

{ glPointSize(4.0); glBegin(GL\_POINTS);

glVertex2f(ROUND(X), ROUND(Y));

glEnd();

1. = X + xinc; //changing value of xnew and ynew
2. = Y + yinc; glFlush(); i++; }

} void DDA\_dashed(float x1, float y1, float x2, float y2)

{

float dx, dy, X, Y,steps;

int i = 0; int count = 0; dx = x2 - x1; dy = y2 - y1; if (abs(dx) >= abs(dy))

{ steps = abs(dx); } else { steps = abs(dy); } dx= dx / steps; dy = dy / steps; X = x1; Y = y1; glBegin(GL\_POINTS);

glVertex2f(ROUND(X), ROUND(Y)); glEnd(); glFlush(); i = 1; count = 1; while(i<=steps) {

X = X + dx; Y = Y + dy;

if (count<=10) //draw for 10 steps

{

glBegin(GL\_POINTS);

glVertex2f(ROUND(X), ROUND(Y));

glEnd();

glFlush(); count++;

}

if (count >= 10) //skip for 10 steps; count=11

{

count++; //count is incremented 12,13,etc

if (count == 20) //reset count if 20 steps skipped

{ count = 1;

} } i++; }

}

void DDA\_dotted(float x1, float y1, float x2, float y2)

{

float dx, dy, X, Y, steps; dx = x2 - x1; dy = y2 - y1; if (abs(dx) >= abs(dy))

{

steps = abs(dx);

} else {

steps = abs(dy);

} dx = dx / steps; dy = dy / steps;

X = x1; Y = y1; glBegin(GL\_POINTS);

glVertex2f(ROUND(X), ROUND(Y)); glEnd(); int count = 1; int i = 1; while (i <= steps) {

X = X + dx; Y = Y + dy; if (count % 10 == 0) //count=10(dot),20(dot) etc...1%10==0

{

glBegin(GL\_POINTS);

glVertex2f(ROUND(X), ROUND(Y)); glEnd(); glFlush();

}

count++; i++; }

}

void DDA\_solid(float x1, float y1, float x2, float y2)

{

float dx, dy, steps, X, Y; dx = x2 - x1; dy = y2 - y1; if (abs(dx) >= abs(dy)) steps = abs(dx);

} else {

steps = abs(dy);

} dx = dx / steps; dy = dy / steps;

X = x1; Y = y1;

int i = 0; while (i <= steps)

{

glPointSize(8.0); //same as simple, only point size has been increased

glBegin(GL\_POINTS); glVertex2f(ROUND(X), ROUND(Y)); glEnd(); X = X + dx; Y = Y + dy; glFlush(); i++; }

}

void BH\_simple(float x1, float y1, float x2, float y2)

{

float dx, dy, di, ds, dt; int count = 0; dx = x2 - x1; dy = y2 - y1;

di = (2 \* (dy)) - (dx); //decision parameter ds = 2 \* (dy); //if value of decision parameter<0 dt = 2 \* ((dy)-(dx)); //if value of decision parameter>0 glPointSize(4.0);

glBegin(GL\_POINTS); glVertex2f(x1, y1); glEnd(); while (x1 < x2)

{ x1++;

if (di < 0) //di=di+2dy=>di=di+ds; ynew=yold; xnew=xold+1

{ di += ds;

}

else //di=di+2(dy-dx)=>di=di+dt; ynew=yold+1; xnew+xold+1

{ y1++; di += dt; } glPointSize(4.0);

glBegin(GL\_POINTS); glVertex2f(x1, y1); glEnd();

}

}

void BH\_dashed(float x1, float y1, float x2, float y2) {

float dx, dy, di, ds, dt; int count = 0; dx = x2 - x1; dy = y2 - y1; di = (2 \* dy) - dx; ds = 2 \* dy; dt = 2 \* (dy - dx); glPointSize(4.0); glBegin(GL\_POINTS); glVertex2f(x1, y1); glEnd(); count = 1; while (x1 < x2) { x1++;

if (di < 0) { di += ds; } else { y1++; di += dt; }

if (count < 6) //draw for 6 steps

{ glPointSize(4.0); glBegin(GL\_POINTS); glVertex2f(x1, y1); glEnd(); count++;

} if (count >= 6)

{

count++; //increment count

if (count == 12)

{ count = 0;

}

}

}

}

void BH\_dotted(float x1, float y1, float x2, float y2)

{

float dx, dy, di, ds, dt; int count = 0; dx = x2 - x1; dy = y2 - y1; di = (2 \* dy) - dx; ds = 2 \* dy; dt = 2 \* (dy - dx); glPointSize(4.0); glBegin(GL\_POINTS); glVertex2f(x1, y1); glEnd(); count = 1; while (x1 < x2)

x1++;

if (di < 0) { di += ds; } else { y1++; di += dt; }

if (count % 10 == 0)

{ glPointSize(4.0); glBegin(GL\_POINTS); glVertex2f(x1, y1); glEnd();

}

count++;

}

}

void BH\_solid(float x1, float y1, float x2, float y2)

{

float dx, dy, di, ds, dt; int count = 0; dx = x2 - x1; dy = y2 - y1; di = (2 \* dy) - dx; ds = 2 \* dy; dt = 2 \* (dy - dx); glPointSize(7.0); glBegin(GL\_POINTS); glVertex2f(x1, y1); glEnd(); while (x1 < x2) { x1++;

if (di < 0) { di += ds; } else { y1++; di += dt; } glPointSize(8.0); glBegin(GL\_POINTS); glVertex2f(x1, y1); glEnd();

}

}

void menu\_dda(int a) {

switch (a) { case 1:

DDA\_simple(10, 50, 150, 200);

break; case 2:

DDA\_dashed(10, 50, 150, 200);

break; case 3:

DDA\_dotted(10, 50, 150, 200);

break; case 4:

DDA\_solid(10, 50, 150, 200);

break; }

}

void menu\_bh(int a)

{

switch (a) { case 1:

BH\_simple(10, 50, 150, 200);

break; case 2:

BH\_dashed(10, 50, 150, 200);

break; case 3:

BH\_dotted(10, 50, 150, 200);

break; case 4:

BH\_solid(10, 50, 150, 200);

break; }

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(800, 600); glutInitWindowPosition(100, 100); glutCreateWindow("DDA and Bresenham"); glutDisplayFunc(draw); submenu\_id1=glutCreateMenu(menu\_dda); glutAddMenuEntry("1.Simple Line", 1); glutAddMenuEntry("2.Dashed Line", 2); glutAddMenuEntry("3.Dotted Line", 3); glutAddMenuEntry("4.Solid Line", 4); submenu\_id2 = glutCreateMenu(menu\_bh); glutAddMenuEntry("1.Simple Line", 1); glutAddMenuEntry("2.Dashed Line", 2); glutAddMenuEntry("3.Dotted Line", 3); glutAddMenuEntry("4.Solid Line", 4); menu\_id = glutCreateMenu(menu\_dda); glutAddSubMenu("DDA Algorithm", submenu\_id1); glutAddSubMenu("Bresenham Algorithm", submenu\_id2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

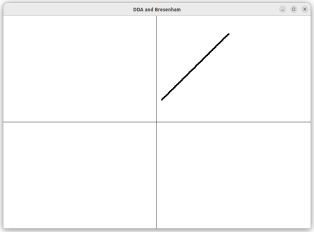
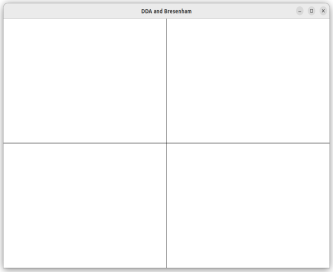
init();

glutMainLoop();

return(0);

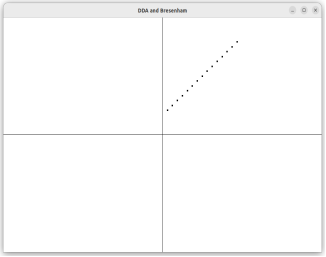
}

OUTPUT:



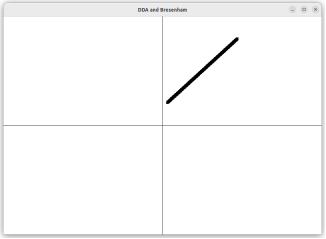
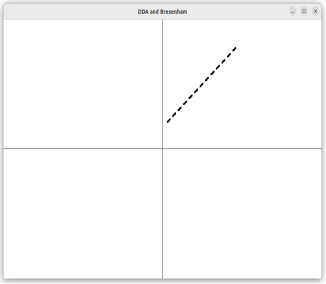
FIRST PAGE

SIMPLE LINE



DOTTED

LINE



DASHED

LINE

SOLID LINE