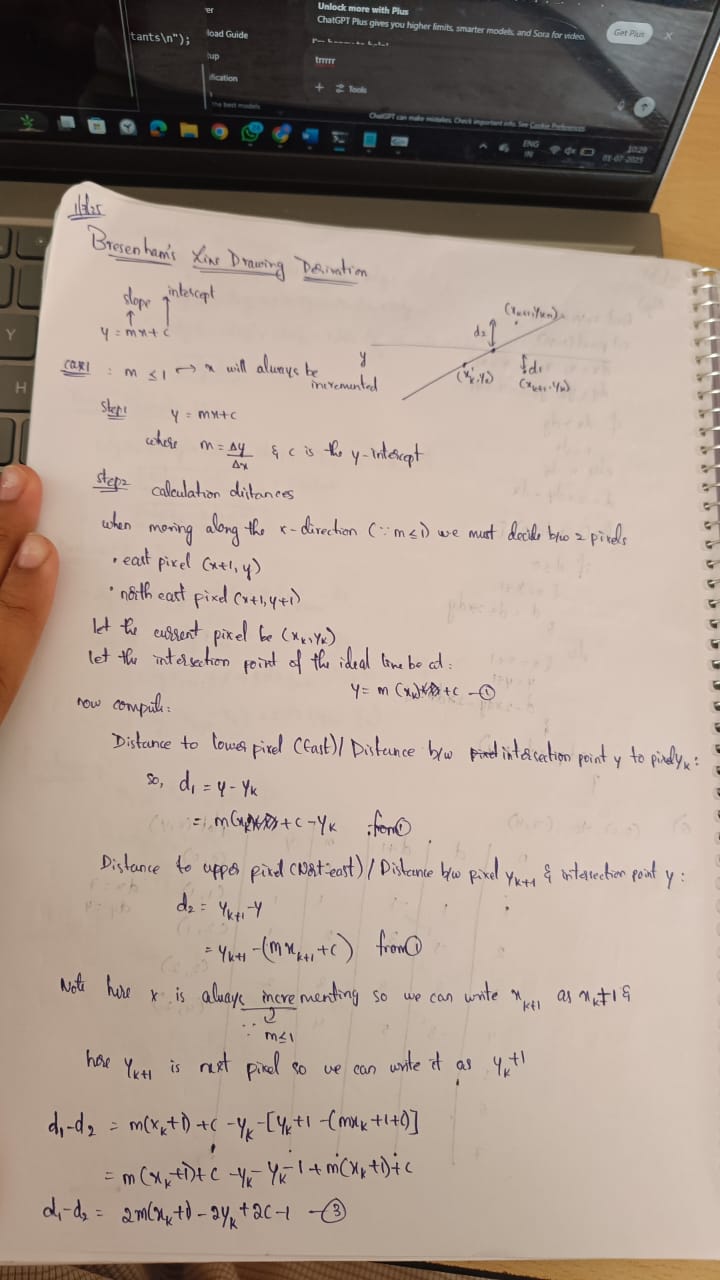
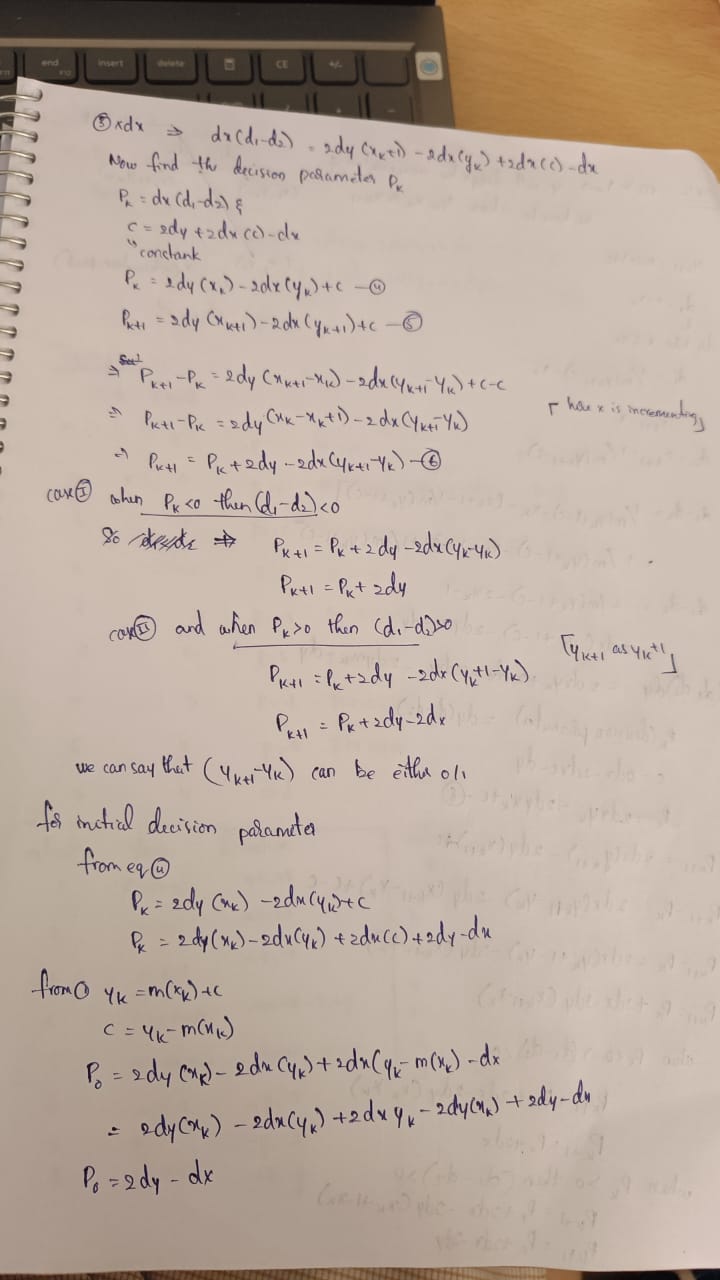
COMPUTER GRAPHICS AND VISUALIZATION LAB2

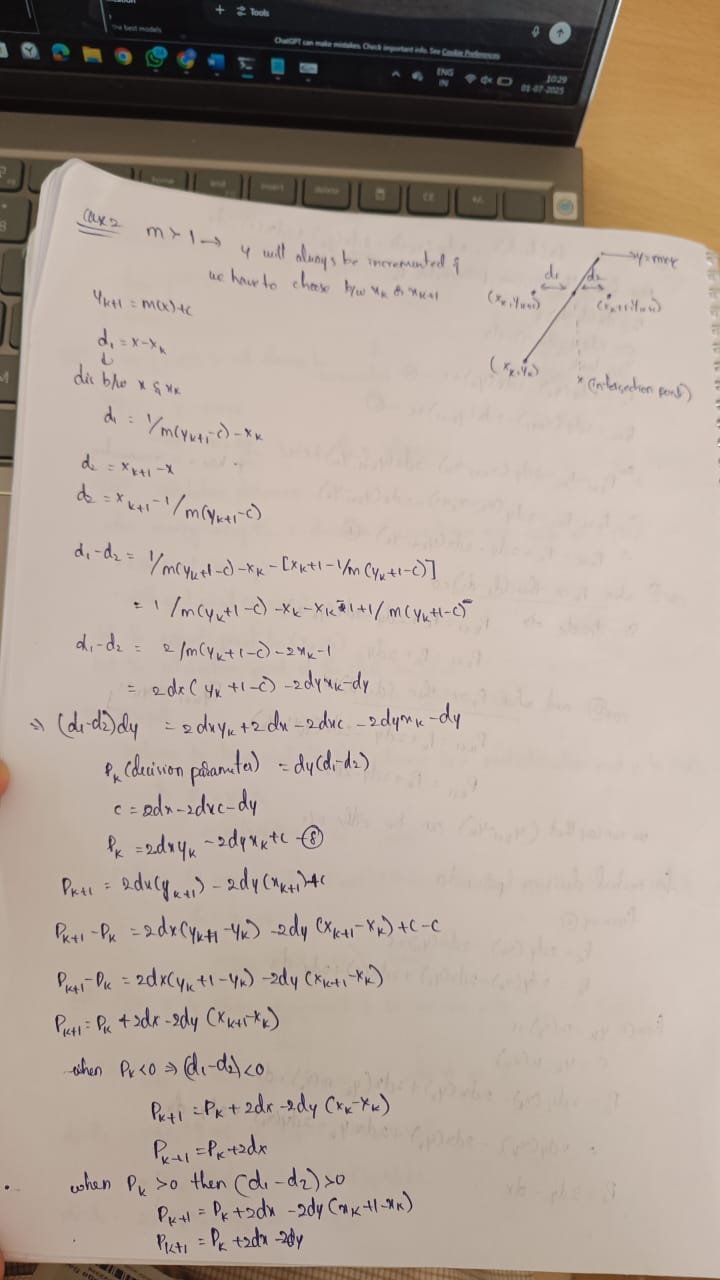
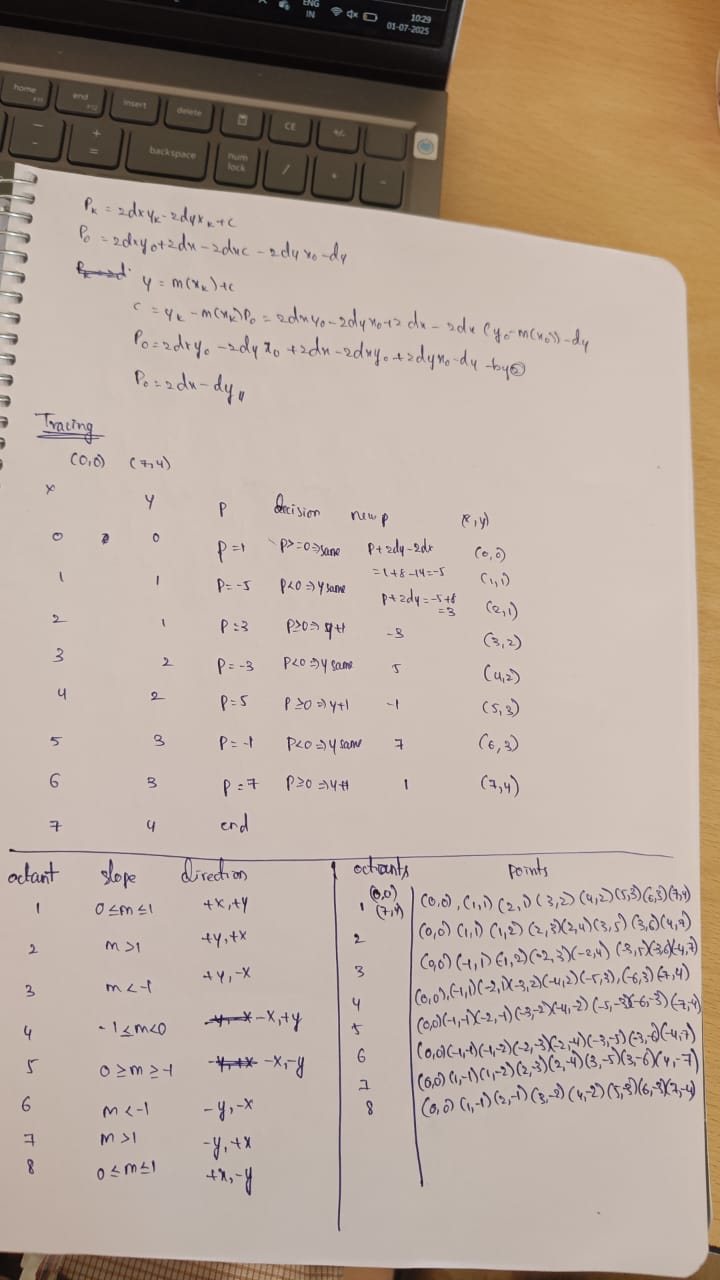
CB.EN.U4CSE22328 K.Bhavya Naga Sai

1. Derive the equations for bresenhams line algorithm

c)Trace the algorithm to scan convert the lines drawn





**(b) Modify the code to draw line in all the octants**

CODE:

#include <windows.h>

#include <GL/glut.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

void init(void) {

glClearColor(1.0, 1.0, 1.0, 0.0); // White background

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 400.0, 0.0, 400.0);

}

void setPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

// Bresenham's Line Drawing Algorithm (generalized for all octants)

void drawBresenhamLine(int x0, int y0, int x1, int y1) {

int dx = abs(x1 - x0);

int dy = abs(y1 - y0);

int x = x0, y = y0;

int sx = (x0 < x1) ? 1 : -1;

int sy = (y0 < y1) ? 1 : -1;

int err = dx - dy;

while (1) {

setPixel(x, y);

if (x == x1 && y == y1) break;

int e2 = 2 \* err;

if (e2 > -dy) {

err -= dy;

x += sx;

}

if (e2 < dx) {

err += dx;

y += sy;

}

}

}

void display(void) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(2.0);

int cx = 200, cy = 200;

// Octant 1

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

drawBresenhamLine(cx, cy, cx + 80, cy + 40);

// Octant 2

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

drawBresenhamLine(cx, cy, cx + 40, cy + 80);

// Octant 3

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

drawBresenhamLine(cx, cy, cx + 40, cy - 80);

// Octant 4

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

drawBresenhamLine(cx, cy, cx + 80, cy - 40);

// Octant 5

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

drawBresenhamLine(cx, cy, cx - 80, cy - 40);

// Octant 6

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

drawBresenhamLine(cx, cy, cx - 40, cy - 80);

// Octant 7

glColor3f(0.5, 0.5, 0.5); // Gray

drawBresenhamLine(cx, cy, cx - 40, cy + 80);

// Octant 8

glColor3f(1.0, 0.5, 0.0); // Orange

drawBresenhamLine(cx, cy, cx - 80, cy + 40);

// Center point

glColor3f(0.0, 0.0, 0.0); // Black

glPointSize(5.0);

setPixel(cx, cy);

glFlush();

}

void keyboard(unsigned char key, int x, int y) {

switch (key) {

case 27: // ESC

exit(0);

break;

case 'r':

case 'R':

glutPostRedisplay();

break;

}

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 400);

glutInitWindowPosition(100, 100);

glutCreateWindow("Bresenham Line - All 8 Octants");

init();

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

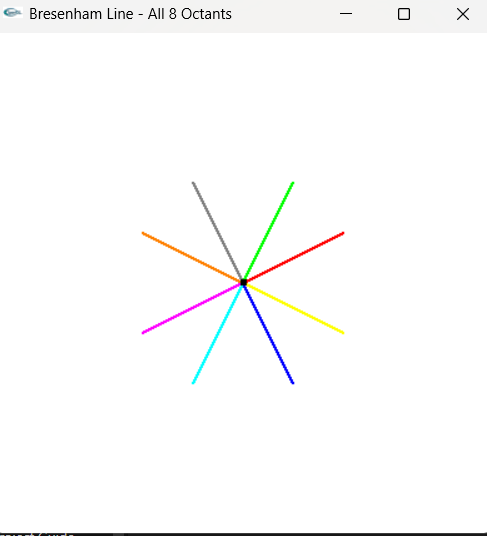
printf("Bresenham's Line Algorithm - All Octants\n");

printf("ESC to exit, R to redraw.\n");

glutMainLoop();

return 0;

}



CODE with user input:  
#include <windows.h>

#include <GL/glut.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

// User-defined line endpoints

int x1\_input, y1\_input, x2\_input, y2\_input;

void init(void) {

glClearColor(1.0, 1.0, 1.0, 0.0); // White background

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 400.0, 0.0, 400.0);

}

void setPixel(int x, int y) {

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

// Bresenham's Line Algorithm (all octants)

void drawBresenhamLine(int x0, int y0, int x1, int y1) {

int dx = abs(x1 - x0);

int dy = abs(y1 - y0);

int x = x0, y = y0;

int sx = (x0 < x1) ? 1 : -1;

int sy = (y0 < y1) ? 1 : -1;

int err = dx - dy;

while (1) {

setPixel(x, y);

if (x == x1 && y == y1) break;

int e2 = 2 \* err;

if (e2 > -dy) { err -= dy; x += sx; }

if (e2 < dx) { err += dx; y += sy; }

}

}

void display(void) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(2.0);

// Line 1: original

glColor3f(1, 0, 0); // Red

drawBresenhamLine(x1\_input, y1\_input, x2\_input, y2\_input);

// Line 2: reflection over x-axis

glColor3f(0, 1, 0); // Green

drawBresenhamLine(x1\_input, 400 - y1\_input, x2\_input, 400 - y2\_input);

// Line 3: reflection over y-axis

glColor3f(0, 0, 1); // Blue

drawBresenhamLine(400 - x1\_input, y1\_input, 400 - x2\_input, y2\_input);

// Line 4: reflection over origin

glColor3f(1, 1, 0); // Yellow

drawBresenhamLine(400 - x1\_input, 400 - y1\_input, 400 - x2\_input, 400 - y2\_input);

// Line 5: reflection across y = x

glColor3f(1, 0, 1); // Magenta

drawBresenhamLine(y1\_input, x1\_input, y2\_input, x2\_input);

// Line 6: reflection across y = -x

glColor3f(0, 1, 1); // Cyan

drawBresenhamLine(400 - y1\_input, 400 - x1\_input, 400 - y2\_input, 400 - x2\_input);

// Line 7: diagonal flip (x,y) to (y,400 - x)

glColor3f(0.5, 0.5, 0.5); // Gray

drawBresenhamLine(y1\_input, 400 - x1\_input, y2\_input, 400 - x2\_input);

// Line 8: diagonal flip (x,y) to (400 - y,x)

glColor3f(1, 0.5, 0); // Orange

drawBresenhamLine(400 - y1\_input, x1\_input, 400 - y2\_input, x2\_input);

glFlush();

}

void keyboard(unsigned char key, int x, int y) {

switch (key) {

case 27: // ESC

exit(0);

break;

case 'r':

case 'R':

glutPostRedisplay();

break;

}

}

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

printf("Enter starting point (x1, y1): ");

scanf("%d %d", &x1\_input, &y1\_input);

printf("Enter ending point (x2, y2): ");

scanf("%d %d", &x2\_input, &y2\_input);

printf("\nGenerating all 8 octant reflections for the line...\n");

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(400, 400);

glutInitWindowPosition(100, 100);

glutCreateWindow("Bresenham Line - 8 Octant Reflections");

init();

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}

