**Lab Report** **3**

*Course title: Computer Graphics*

*Course code: CSE-304*

*3rd Year 1st Semester Examination 2022*

**Date of Submission**: 11 June 2023

****

*Department of Computer Science and Engineering*

*Jahangirnagar University*

*Savar, Dhaka-1342*

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl** | Class Roll | Exam Roll | Name |
| 01 | 367 | 202179 | Fatima Binte Aziz |

**Scan convert a line object from (0,0) to (100,50 ):**

**Source code:**

#include <graphics.h>

#include <cmath>

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int x1 = 0, y1 = 0;

int x2 = 100, y2 = 50;

// Calculate the slope and intercept of the line

float dx = x2 - x1;

float dy = y2 - y1;

float slope = dy / dx;

// Determine the number of steps to take based on the greater difference (dx or dy)

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

// Calculate the increment values for x and y

float xIncrement = dx / steps;

float yIncrement = dy / steps;

// Scan convert the line

float x = x1, y = y1;

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

putpixel(round(x), round(y), WHITE);

x += xIncrement;

y += yIncrement;

}

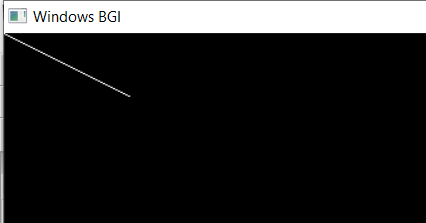
delay(5000);

closegraph();

return 0;

}

**Output:**

****

**Rotate it by 30 degree:**

**Source code:**

#include <graphics.h>

#include <cmath>

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int x1 = 0, y1 = 0;

int x2 = 100, y2 = 50;

// Calculate the rotation angle in radians

float angle = 30.0f \* M\_PI / 180.0f;

// Apply rotation transformation to each point of the line

float rotatedX1 = x1 \* cos(angle) - y1 \* sin(angle);

float rotatedY1 = x1 \* sin(angle) + y1 \* cos(angle);

float rotatedX2 = x2 \* cos(angle) - y2 \* sin(angle);

float rotatedY2 = x2 \* sin(angle) + y2 \* cos(angle);

// Draw the rotated line

line(round(rotatedX1), round(rotatedY1), round(rotatedX2), round(rotatedY2));

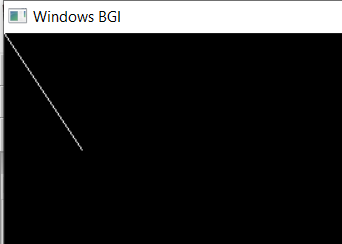
delay(5000);

closegraph();

return 0;

}

**Output:**



**Scale it to 50%:**

**Source code:**

#include <graphics.h>

#include <cmath>

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int x1 = 0, y1 = 0;

int x2 = 100, y2 = 50;

// Calculate the scaling factors

float scaleX = 0.5f;

float scaleY = 0.5f;

// Apply scaling transformation to each point of the line

float scaledX1 = x1 \* scaleX;

float scaledY1 = y1 \* scaleY;

float scaledX2 = x2 \* scaleX;

float scaledY2 = y2 \* scaleY;

// Draw the scaled line

line(round(scaledX1), round(scaledY1), round(scaledX2), round(scaledY2));

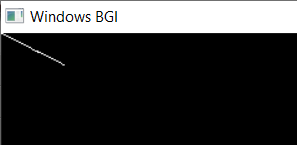
delay(5000);

closegraph();

return 0;

}

**Output:**



**Translate it on x axis by 75 pixels:**

**Source code:**

#include <graphics.h>

#include <cmath>

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int x1 = 0, y1 = 0;

int x2 = 100, y2 = 50;

// Translation offset on the X-axis

int translationX = 75;

// Apply translation transformation to each point of the line

int translatedX1 = x1 + translationX;

int translatedY1 = y1;

int translatedX2 = x2 + translationX;

int translatedY2 = y2;

// Draw the translated line

line(translatedX1, translatedY1, translatedX2, translatedY2);

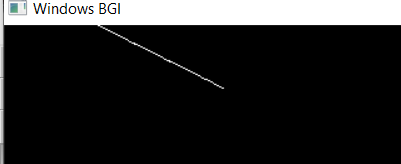
delay(5000);

closegraph();

return 0;

}

**Output:**



**Drawing a kite using Brecenham line algorithm**

**Source code:**

#include <graphics.h>

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

int dx = abs(x2 - x1);

int dy = abs(y2 - y1);

int sx = (x1 < x2) ? 1 : -1;

int sy = (y1 < y2) ? 1 : -1;

int err = dx - dy;

int err2;

while (true) {

putpixel(x1, y1, WHITE);

if (x1 == x2 && y1 == y2)

break;

err2 = 2 \* err;

if (err2 > -dy) {

err -= dy;

x1 += sx;

}

if (err2 < dx) {

err += dx;

y1 += sy;

}

}

}

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int x = getmaxx() / 2; // X-coordinate of the center of the screen

int y = getmaxy() / 2; // Y-coordinate of the center of the screen

int length = 100; // Length of the kite's sides

int halfLength = length / 2;

// Draw the lines of the kite

drawLine(x, y - length, x - halfLength, y);

drawLine(x, y - length, x + halfLength, y);

drawLine(x - halfLength, y, x, y + length);

drawLine(x + halfLength, y, x, y + length);

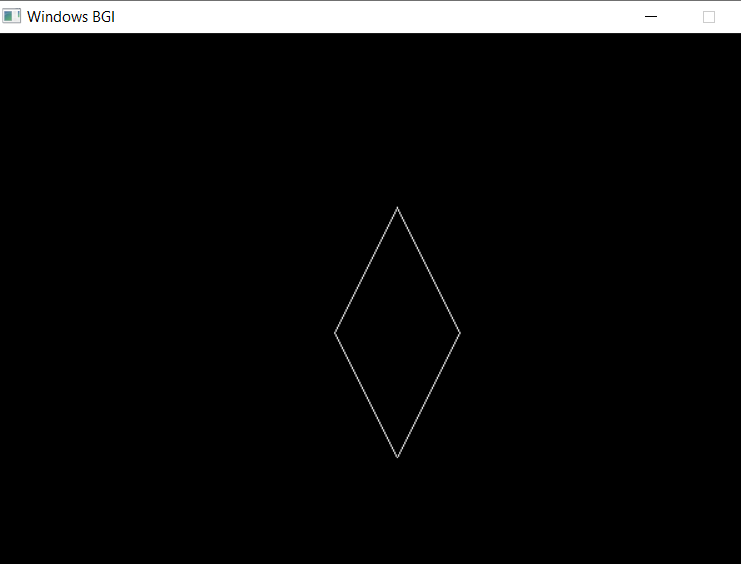
delay(5000);

closegraph();

return 0;

}

**Output:**

****