

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

i) rotating by 30 degree:

Code:

#include <iostream>

#include <graphics.h>

#include <cmath>

void rotatePoint(int x, int y, int cx, int cy, float angle, int& newX, int& newY) {

float radians = angle \* M\_PI / 180.0;

float cosine = cos(radians);

float sine = sin(radians);

newX = round((x - cx) \* cosine - (y - cy) \* sine) + cx;

newY = round((x - cx) \* sine + (y - cy) \* cosine) + cy;

}

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

int dx = abs(x2 - x1);

int dy = abs(y2 - y1);

int steps = dx > dy ? dx : dy;

float xIncrement = (x2 - x1) / (float)steps;

float yIncrement = (y2 - y1) / (float)steps;

float x = x1;

float y = y1;

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

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

x += xIncrement;

y += yIncrement;

}

}

int main() {

int gd = DETECT, gm;

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

int x1 = 0, y1 = 0;

int x2 = 100, y2 = 50;

int centerX = 0, centerY = 0;

int rotatedX, rotatedY;

float rotationAngle = 30.0;

rotatePoint(x1, y1, centerX, centerY, rotationAngle, rotatedX, rotatedY);

x1 = rotatedX;

y1 = rotatedY;

rotatePoint(x2, y2, centerX, centerY, rotationAngle, rotatedX, rotatedY);

x2 = rotatedX;

y2 = rotatedY;

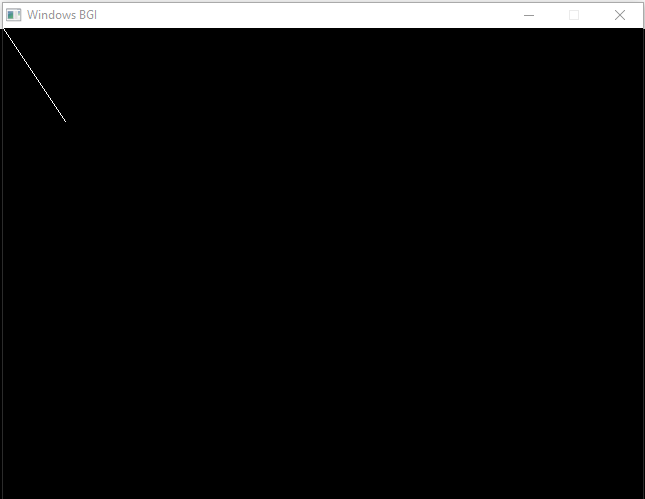
drawLine(x1, y1, x2, y2);

getch();

closegraph();

return 0;

}

Output:  


ii) scale it to 50%:

Code:

#include <iostream>

#include <graphics.h>

#include <cmath>

void rotatePoint(int x, int y, int cx, int cy, float angle, int& newX, int& newY) {

float radians = angle \* M\_PI / 180.0;

float cosine = cos(radians);

float sine = sin(radians);

newX = round((x - cx) \* cosine - (y - cy) \* sine) + cx;

newY = round((x - cx) \* sine + (y - cy) \* cosine) + cy;

}

void scalePoint(int x, int y, float scaleFactor, int& newX, int& newY) {

newX = round(x \* scaleFactor);

newY = round(y \* scaleFactor);

}

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

int dx = abs(x2 - x1);

int dy = abs(y2 - y1);

int steps = dx > dy ? dx : dy;

float xIncrement = (x2 - x1) / (float)steps;

float yIncrement = (y2 - y1) / (float)steps;

float x = x1;

float y = y1;

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

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

x += xIncrement;

y += yIncrement;

}

}

int main() {

int gd = DETECT, gm;

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

int x1 = 0, y1 = 0;

int x2 = 100, y2 = 50;

int centerX = 0, centerY = 0;

int rotatedX, rotatedY;

float rotationAngle = 30.0;

rotatePoint(x1, y1, centerX, centerY, rotationAngle, rotatedX, rotatedY);

x1 = rotatedX;

y1 = rotatedY;

rotatePoint(x2, y2, centerX, centerY, rotationAngle, rotatedX, rotatedY);

x2 = rotatedX;

y2 = rotatedY;

float scaleFactor = 0.5;

scalePoint(x1, y1, scaleFactor, x1, y1);

scalePoint(x2, y2, scaleFactor, x2, y2);

drawLine(x1, y1, x2, y2);

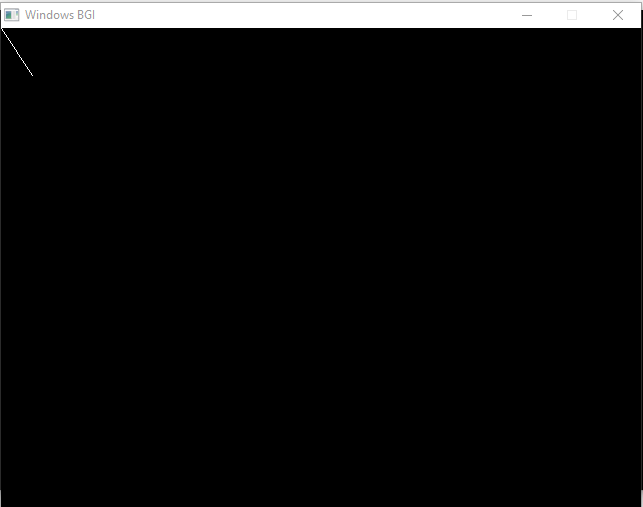
getch();

closegraph();

return 0;

}

Output:



iii)translate it on x axis by 75 pixels:

Code:

#include <iostream>

#include <graphics.h>

#include <cmath>

void rotatePoint(int x, int y, int cx, int cy, float angle, int& newX, int& newY) {

float radians = angle \* M\_PI / 180.0;

float cosine = cos(radians);

float sine = sin(radians);

newX = round((x - cx) \* cosine - (y - cy) \* sine) + cx;

newY = round((x - cx) \* sine + (y - cy) \* cosine) + cy;

}

void scalePoint(int x, int y, float scaleFactor, int& newX, int& newY) {

newX = round(x \* scaleFactor);

newY = round(y \* scaleFactor);

}

void translatePoint(int x, int y, int translateX, int translateY, int& newX, int& newY) {

newX = x + translateX;

newY = y + translateY;

}

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

int dx = abs(x2 - x1);

int dy = abs(y2 - y1);

int steps = dx > dy ? dx : dy;

float xIncrement = (x2 - x1) / (float)steps;

float yIncrement = (y2 - y1) / (float)steps;

float x = x1;

float y = y1;

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

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

x += xIncrement;

y += yIncrement;

}

}

int main() {

int gd = DETECT, gm;

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

int x1 = 0, y1 = 0;

int x2 = 100, y2 = 50;

int centerX = 0, centerY = 0;

int rotatedX, rotatedY;

float rotationAngle = 30.0;

rotatePoint(x1, y1, centerX, centerY, rotationAngle, rotatedX, rotatedY);

x1 = rotatedX;

y1 = rotatedY;

rotatePoint(x2, y2, centerX, centerY, rotationAngle, rotatedX, rotatedY);

x2 = rotatedX;

y2 = rotatedY;

float scaleFactor = 0.5;

scalePoint(x1, y1, scaleFactor, x1, y1);

scalePoint(x2, y2, scaleFactor, x2, y2);

int translateX = 75;

int translateY = 0;

translatePoint(x1, y1, translateX, translateY, x1, y1);

translatePoint(x2, y2, translateX, translateY, x2, y2);

drawLine(x1, y1, x2, y2);

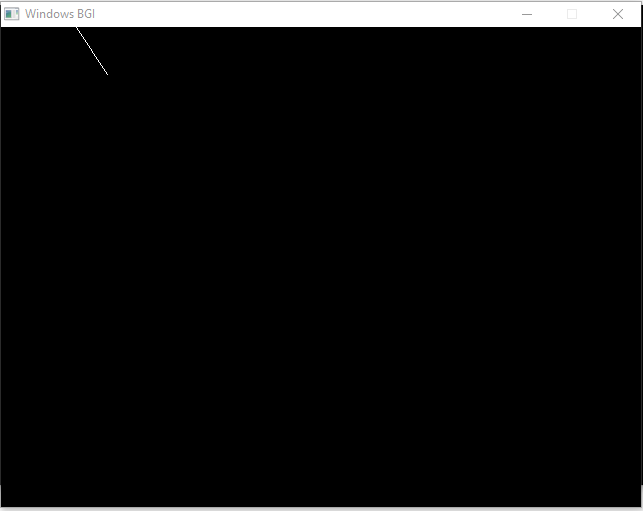
getch();

closegraph();

return 0;

}

Output:



**kite using brasenham algorithm:**

Code:

#include <iostream>

#include <graphics.h>

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

int dx = abs(x2 - x1);

int dy = abs(y2 - y1);

int x = x1;

int y = y1;

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

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

int error = dx - dy;

while (true) {

putpixel(x, y, WHITE);

if (x == x2 && y == y2)

break;

int e2 = 2 \* error;

if (e2 > -dy) {

error -= dy;

x += sx;

}

if (e2 < dx) {

error += dx;

y += sy;

}

}

}

int main() {

int gd = DETECT, gm;

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

// Coordinates of the kite shape

int x1 = 100, y1 = 100;

int x2 = 200, y2 = 200;

int x3 = 300, y3 = 100;

int x4 = 200, y4 = 0;

// Draw lines to form the kite shape

drawLineBresenham(x1, y1, x2, y2);

drawLineBresenham(x2, y2, x3, y3);

drawLineBresenham(x3, y3, x4, y4);

drawLineBresenham(x4, y4, x1, y1);

getch();

closegraph();

return 0;

}

Output:

