**NAME:- SURWADE TRISHARAN RAJESH**

**ROLL NO:- 47**

**CLASS -MSC(CS)-IInd Year**

**6. Composite 2-D transformation, (rotation, scaling & reflection)**

**Rotation**

#include <iostream.h>

#include <graphics.h>

#include <conio.h>

#include <math.h>

class Rotation {

private:

int x1, y1, x2, y2, x3, y3; // Original coordinates of the triangle vertices

int xRot1, yRot1, xRot2, yRot2, xRot3, yRot3; // Rotated coordinates of the triangle vertices

float angle; // Rotation angle in degrees

// Function to apply centroid-based rotation

void rotateAroundCentroid() {

float rad = angle \* 3.14 / 180; // Convert angle to radians

// Calculate the centroid

int cx = (x1 + x2 + x3) / 3;

int cy = (y1 + y2 + y3) / 3;

// Rotate each point around the centroid

xRot1 = cx + (x1 - cx) \* cos(rad) - (y1 - cy) \* sin(rad);

yRot1 = cy + (x1 - cx) \* sin(rad) + (y1 - cy) \* cos(rad);

xRot2 = cx + (x2 - cx) \* cos(rad) - (y2 - cy) \* sin(rad);

yRot2 = cy + (x2 - cx) \* sin(rad) + (y2 - cy) \* cos(rad);

xRot3 = cx + (x3 - cx) \* cos(rad) - (y3 - cy) \* sin(rad);

yRot3 = cy + (x3 - cx) \* sin(rad) + (y3 - cy) \* cos(rad);

}

public:

// Constructor to initialize triangle vertices and rotation angle

Rotation(int x1, int y1, int x2, int y2, int x3, int y3, float angle)

: x1(x1), y1(y1), x2(x2), y2(y2), x3(x3), y3(y3), angle(angle) {}

// Function to draw the original triangle

void drawOriginal() {

setcolor(WHITE);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

outtextxy(x1 - 10, y1 - 20, "Before Rotation");

}

// Function to draw the rotated triangle

void drawRotated() {

cleardevice(); // Clear previous drawing to avoid overlap

rotateAroundCentroid();

setcolor(YELLOW);

line(xRot1, yRot1, xRot2, yRot2);

line(xRot2, yRot2, xRot3, yRot3);

line(xRot3, yRot3, xRot1, yRot1);

outtextxy(xRot3 + 10, yRot3 + 5 , "After Rotation");

}

};

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

int x1, y1, x2, y2, x3, y3;

float angle;

cout << "Enter the first coordinate (x y): ";

cin >> x1 >> y1;

cout << "Enter the second coordinate (x y): ";

cin >> x2 >> y2;

cout << "Enter the third coordinate (x y): ";

cin >> x3 >> y3;

cout << "Enter the rotation angle (in degrees): ";

cin >> angle;

// Creating an instance of the Rotation class

Rotation triangle(x1, y1, x2, y2, x3, y3, angle);

// Draw the original triangle

cout << "Original Triangle:\n";

triangle.drawOriginal();

getch();

// Clear the screen and draw the rotated triangle

triangle.drawRotated();

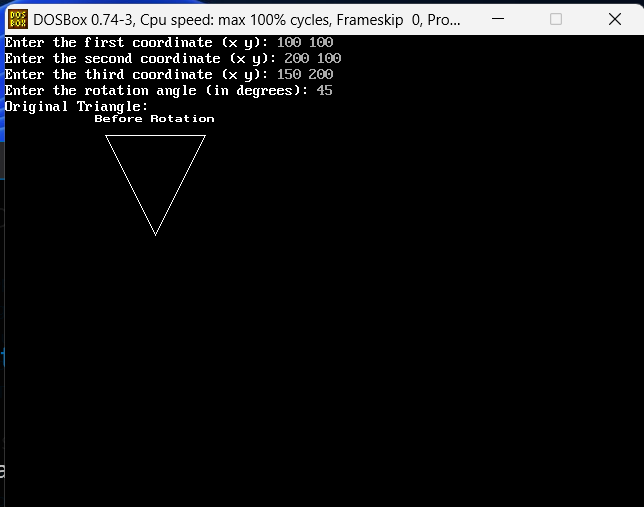
getch();

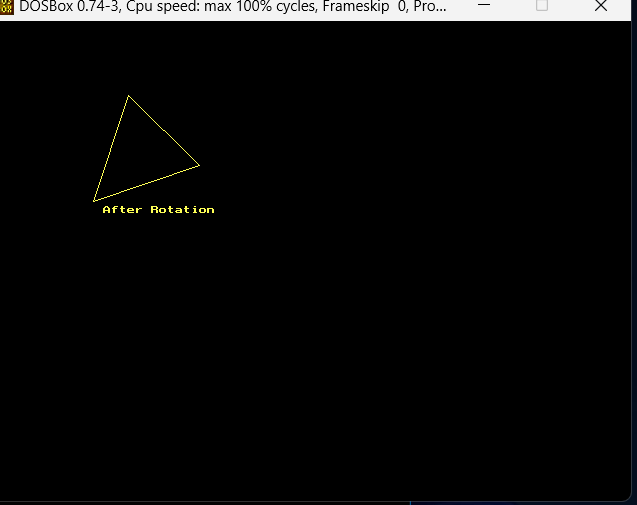
closegraph();

return 0;

}

**OUTPUT:**





**Scaling**

#include <iostream.h>

#include <graphics.h>

#include <conio.h>

#include <math.h>

class ScalingTransformation {

private:

int x, y, x1, y1, x2, y2; // Original coordinates of the triangle vertices

int sx, sy; // Scaling factors

// Function to draw a triangle based on the given vertices

void drawTriangle(int x, int y, int x1, int y1, int x2, int y2, const char\* label) {

line(x, y, x1, y1);

line(x1, y1, x2, y2);

line(x2, y2, x, y);

outtextxy(x - 10, y - 20, label);

}

public:

// Constructor to initialize triangle vertices and scaling factors

ScalingTransformation(int x, int y, int x1, int y1, int x2, int y2, int sx, int sy)

: x(x), y(y), x1(x1), y1(y1), x2(x2), y2(y2), sx(sx), sy(sy) {}

// Function to display the original triangle

void displayOriginal() {

drawTriangle(x, y, x1, y1, x2, y2, "Before Scaling");

}

// Function to scale the triangle and display the scaled triangle

void scaleAndDisplay() {

int newX = x \* sx;

int newY = y \* sy;

int newX1 = x1 \* sx;

int newY1 = y1 \* sy;

int newX2 = x2 \* sx;

int newY2 = y2 \* sy;

drawTriangle(newX, newY, newX1, newY1, newX2, newY2, "After Scaling");

}

};

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

int x, y, x1, y1, x2, y2, sx, sy;

cout << "Enter the first coordinate (x y): ";

cin >> x >> y;

cout << "Enter the second coordinate (x1 y1): ";

cin >> x1 >> y1;

cout << "Enter the third coordinate (x2 y2): ";

cin >> x2 >> y2;

cout << "Enter the scaling factors (sx sy): ";

cin >> sx >> sy;

// Create an object of ScalingTransformation with the input values

ScalingTransformation triangle(x, y, x1, y1, x2, y2, sx, sy);

// Display the original triangle

cout << "Original Triangle:\n";

triangle.displayOriginal();

getch();

// Display the scaled triangle

cleardevice();

cout << "Scaled Triangle:\n";

triangle.scaleAndDisplay();

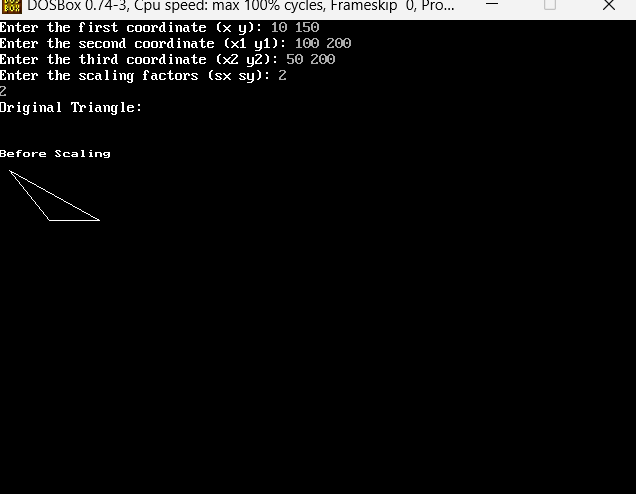
getch();

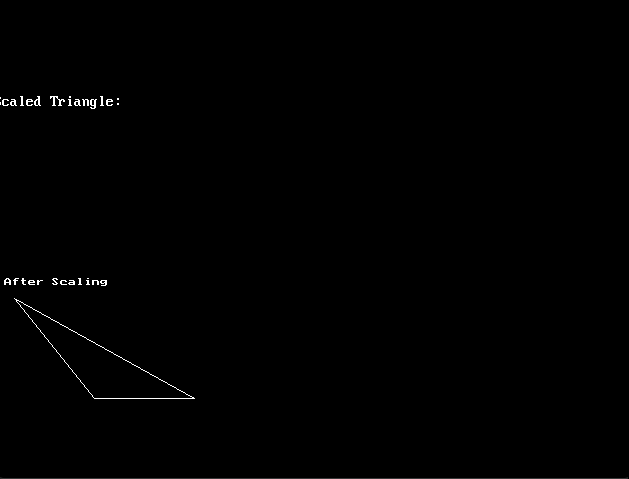
closegraph();

return 0;

}

**OUTPUT:**





**reflection**

#include <iostream.h>

#include <graphics.h>

#include <conio.h>

#include <math.h>

class Reflection {

private:

int x1, y1, x2, y2, x3, y3; // Coordinates of the triangle vertices

int xCenter; // Center x-axis (middle of the screen)

int yCenter; // Center y-axis (middle of the screen)

// Draws coordinate axes

void drawAxes() {

line(xCenter, 0, xCenter, 480); // Vertical axis

line(0, yCenter, 640, yCenter); // Horizontal axis

}

public:

// Constructor to initialize the triangle vertices and center coordinates

Reflection(int x1, int y1, int x2, int y2, int x3, int y3)

: x1(x1), y1(y1), x2(x2), y2(y2), x3(x3), y3(y3), xCenter(320), yCenter(240) {}

// Function to draw the original triangle

void drawOriginal() {

drawAxes();

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

}

// Reflects the triangle along the x-axis

void reflectX() {

drawAxes();

int newX1 = (xCenter - x1) + xCenter;

int newX2 = (xCenter - x2) + xCenter;

int newX3 = (xCenter - x3) + xCenter;

line(newX1, y1, newX2, y2);

line(newX2, y2, newX3, y3);

line(newX3, y3, newX1, y1);

}

// Reflects the triangle along the y-axis

void reflectY() {

drawAxes();

int newY1 = (yCenter - y1) + yCenter;

int newY2 = (yCenter - y2) + yCenter;

int newY3 = (yCenter - y3) + yCenter;

line(x1, newY1, x2, newY2);

line(x2, newY2, x3, newY3);

line(x3, newY3, x1, newY1);

}

};

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

int x1, y1, x2, y2, x3, y3;

cout << "Enter the first coordinate (x y): ";

cin >> x1 >> y1;

cout << "Enter the second coordinate (x y): ";

cin >> x2 >> y2;

cout << "Enter the third coordinate (x y): ";

cin >> x3 >> y3;

Reflection triangle(x1, y1, x2, y2, x3, y3);

// Draw the original triangle

cout << "Original Triangle:\n";

triangle.drawOriginal();

getch();

// Reflect triangle along the x-axis

cleardevice();

cout << "Reflection along X-axis:\n";

triangle.reflectX();

getch();

// Reflect triangle along the y-axis

cleardevice();

cout << "Reflection along Y-axis:\n";

triangle.reflectY();

getch();

closegraph();

return 0;}

**OUTPUT:**

