Course title: Computer Graphics Laboratory

Course code: CSE-304

3rd year 1st semester

Date of Submission: 11/06/2023

Submitted to-

Dr. Mohammad Shorif Uddin

Professor

and

Dr. Morium Akter

Associate Professor
Department of Computer Science and Engineering

Jahangirnagar University

Savar, Dhaka-1342

Sl	Class Roll	Exam Roll	Name
01	392	202204	Choyon Sarker

Department of Computer Science and Engineering
Jahangirnagar University
Savar, Dhaka, Bangladesh

Experiment no 1:

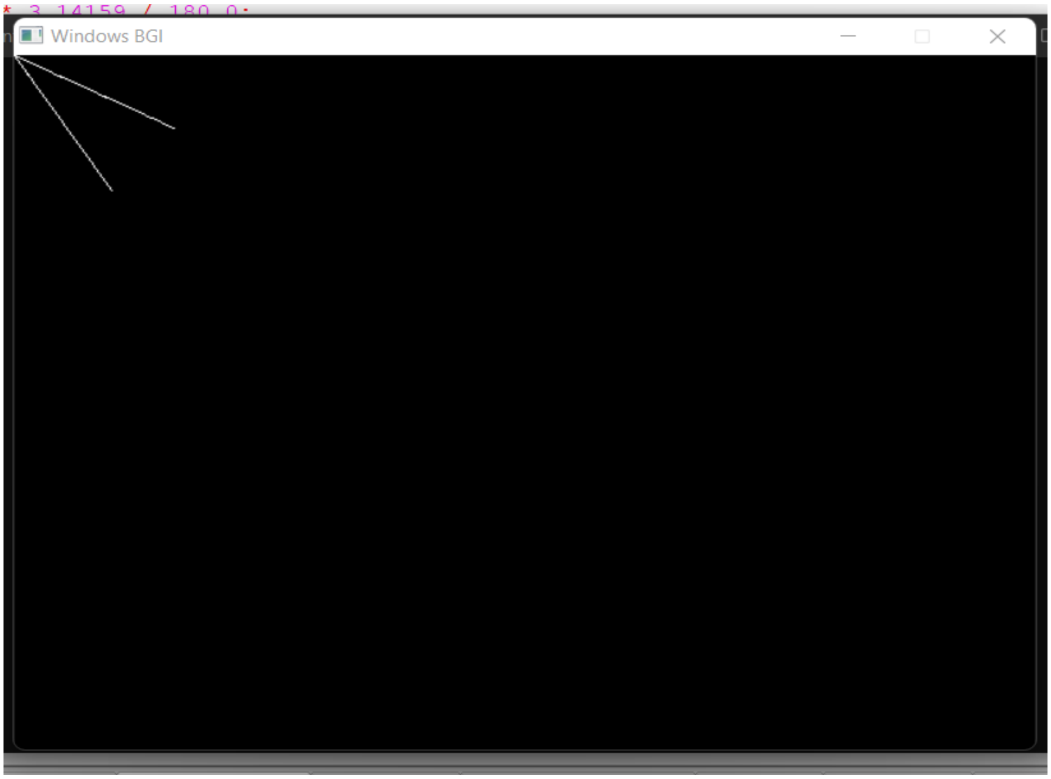
Name of the experiment: Scan converting a line object from (0,0) to (100,50)

Problem 1: Rotating by 30 degree

Code:

```
#include <iostream>
#include <graphics.h>
#include <cmath>

void rotatePoint(int& x, int& y, float angle) {
    float radian = angle * 3.14159 / 180.0;
    float newX = x * cos(radian) - y * sin(radian);
    float newY = x * sin(radian) + y * cos(radian);
    x = (int)newX;
    y = (int)newY;
}

void rotateLine(int x1, int y1, int x2, int y2, float angle) {
    rotatePoint(x1, y1, angle);
    rotatePoint(x2, y2, angle);
    line(x1, y1, x2, y2);
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    int x1 = 0, y1 = 0, x2 = 100, y2 = 50;

    line(x1, y1, x2, y2);
    rotateLine(x1, y1, x2, y2, 30);

    getch();
    closegraph();

    return 0;
}
```

Output:



```
* 3.14159 / 180.0;
```

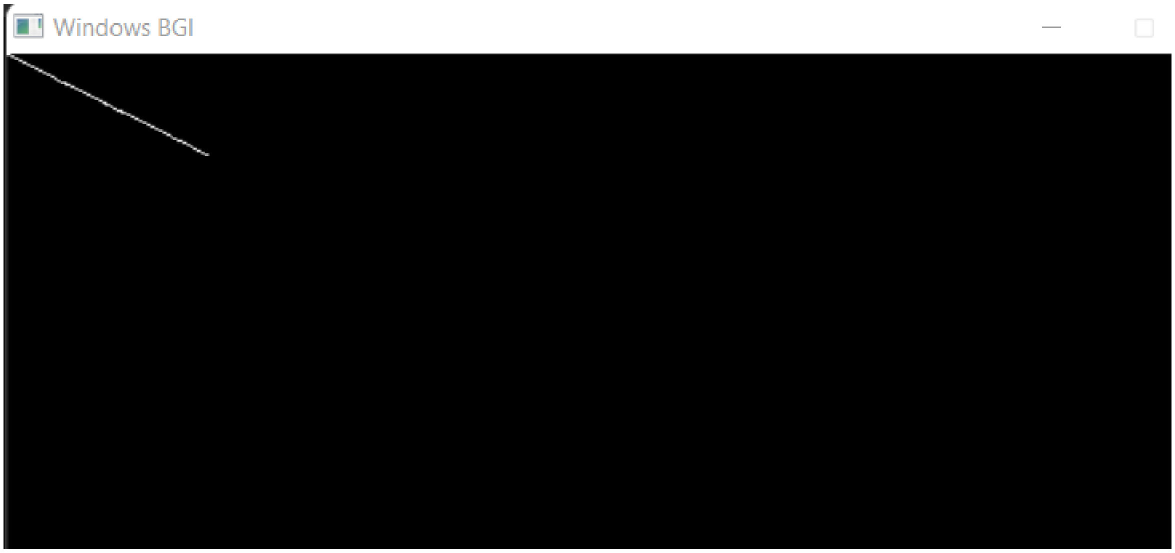
Problem 2: scale it to 50 %

Code:

```
#include <iostream>
#include <graphics.h>

void scaleLine(int x1, int y1, int x2, int y2, float scale) {
   int newX1 = x1 * scale;
   int newY1 = y1 * scale;
   int newX2 = x2 * scale;
   int newY2 = y2 * scale;

   line(newX1, newY1, newX2, newY2);
}

int main() {
   int gd = DETECT, gm;
```

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

    // Original line coordinates
    int x1 = 0, y1 = 0, x2 = 100, y2 = 50;

    // Draw the original line
    line(x1, y1, x2, y2);

    // Scale the line to 50%
    float scale = 0.5;
    scaleLine(x1, y1, x2, y2, scale);

    getch();
    closegraph();

    return 0;
}
```

Output:

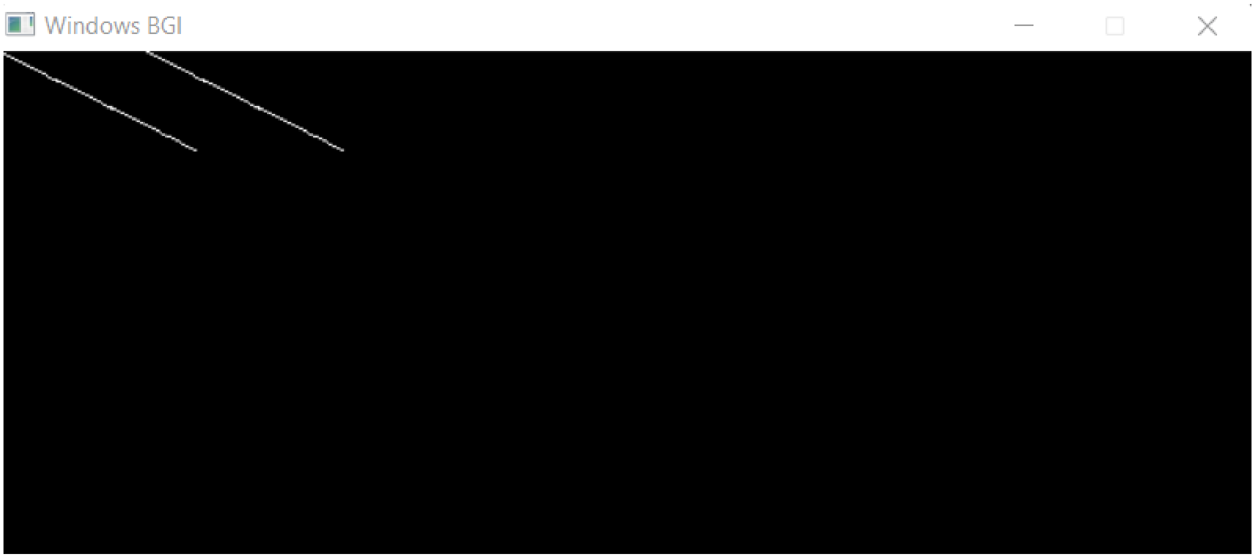


Problem 3: Translate it on 75px

Code:

```
#include <iostream>
#include <graphics.h>

void translateLine(int x1, int y1, int x2, int y2, float translateX) {
   int newX1 = x1 + translateX;
   int newY1 = y1;
   int newX2 = x2 + translateX;
   int newY2 = y2;

   line(newX1, newY1, newX2, newY2);
}

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");

   // Original line coordinates
   int x1 = 0, y1 = 0, x2 = 100, y2 = 50;

   // Draw the original line
   line(x1, y1, x2, y2);

   // Translate the line on the x-axis by 75%
   float translateX = 0.75 * x2; // 75% of x2
   translateLine(x1, y1, x2, y2, translateX);

   getch();
   closegraph();

   return 0;
}
```

Output:



Experiment no 2:

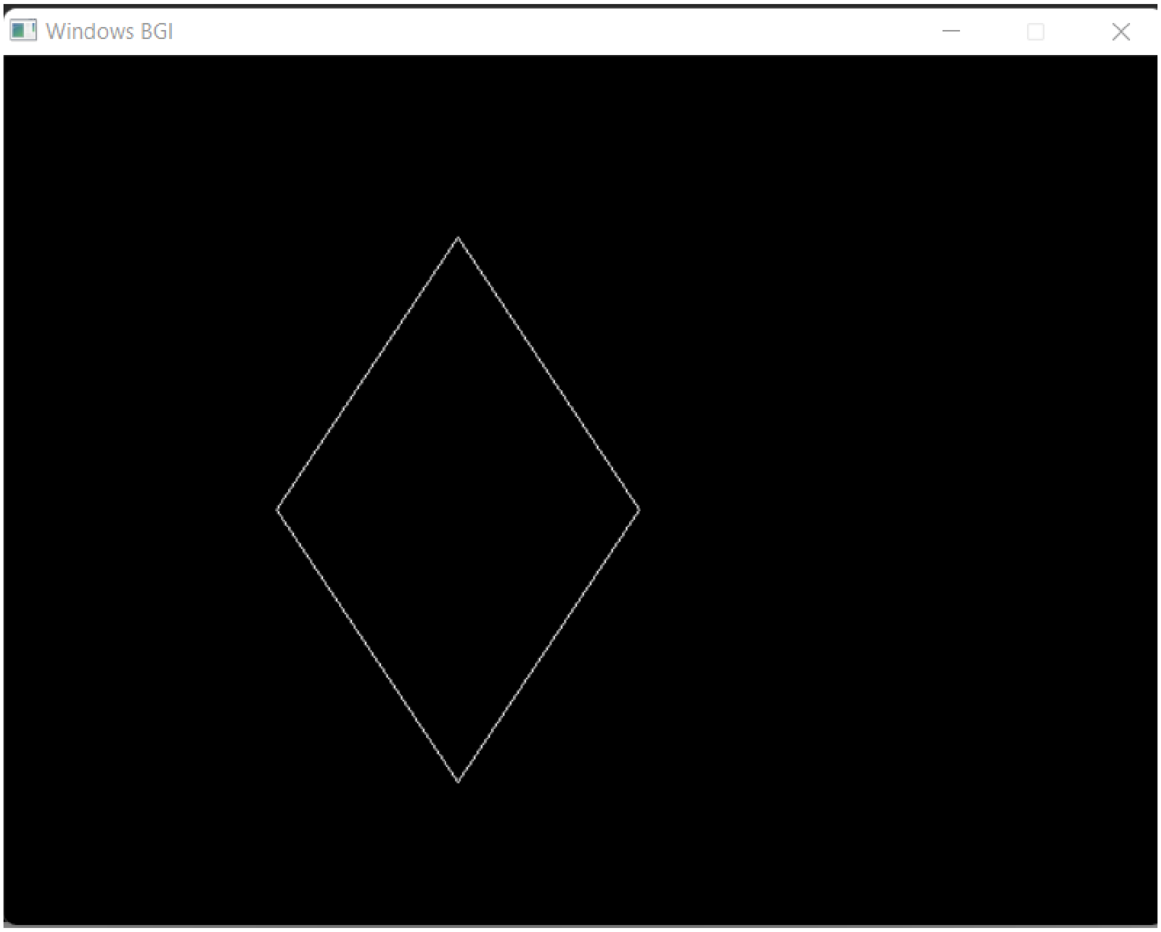
Name of the experiment: Drawing a kite using bresenham's line algorithm

Code:

```
#include <iostream>
#include <graphics.h>

void drawLine(int x1, int y1, int x2, int y2) {
   int dx = abs(x2 - x1);
   int dy = abs(y2 - y1);
   int x = x1, y = y1;
   int xInc = (x2 >= x1) ? 1 : -1;
   int yInc = (y2 >= y1) ? 1 : -1;

   if (dx >= dy) {
      int p = 2 * dy - dx;
      while (x != x2) {
         putpixel(x, y, WHITE);
         if (p >= 0) {
            y += yInc;
            p -= 2 * dx;
         }
```

```
            x += xInc;
            p += 2 * dy;
        }
    } else {
        int p = 2 * dx - dy;
        while (y != y2) {
            putpixel(x, y, WHITE);
            if (p >= 0) {
                x += xInc;
                p -= 2 * dy;
            }
            y += yInc;
            p += 2 * dx;
        }
    }
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    // Kite coordinates
    int topX = 250, topY = 100;
    int leftX = 150, leftY = 250;
    int rightX = 350, rightY = 250;
    int bottomX = 250, bottomY = 400;

    // Draw the kite using Bresenham line algorithm
    drawLine(topX, topY, leftX, leftY);
    drawLine(leftX, leftY, bottomX, bottomY);
    drawLine(bottomX, bottomY, rightX, rightY);
    drawLine(rightX, rightY, topX, topY);

    getch();
    closegraph();

    return 0;
}
```

Output:


Windows BGI