Tribhuvan University

Faculty of Humanities and Social Sciences

Lab Report of Computer Graphics

Submitted to

Department of Computer Application

Shahid Smarak College

Submitted by**:**

Name: Amir Maharjan

Tu Registration Number: 6-2-262-3-2020

Under the Supervision of

Shaya Adhikari

Table of Contents

1. Program to implement DDA Line Drawing Algorithm:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
intgd = DETECT ,gm, i;
float x, y,dx,dy,steps;
int x0, x1, y0, y1;
initgraph(&gd, &gm, "C:\\TC\\BGI");
setbkcolor(WHITE);
x0 = 100 , y0 = 200, x1 = 500, y1 = 300;
dx = (float)(x1 - x0);
dy = (float)(y1 - y0);
if(dx>=dy)
{
steps = dx;
}
else
{
steps = dy;
}
dx = dx/steps;
dy = dy/steps;
x = x0;
y = y0;
i = 1;
while(i<= steps)
```

```
{
putpixel(x, y, RED);
x += dx;
y += dy;
i=i+1;
}
getch();
closegraph();
}
```

2. Program to draw a circle using Midpoint Algorithm:

```
#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <stdio.h>
#include <graphics.h>

void drawCircle(int xc, int yc, int radius) {
    int x = radius;
    int y = 0;
    int err = 0;

    // Open a graphics window
    initwindow(400, 400, "Circle Drawing");

    // Plot the initial point in each quadrant
    putpixel(xc + x, yc - y, WHITE);
    putpixel(xc - x, yc - y, WHITE);
    putpixel(xc + y, yc + x, WHITE);
    putpixel(xc - y, yc + x, WHITE);

    while (x >= y) {
        y++;

        // Mid-point is inside or on the perimeter
```

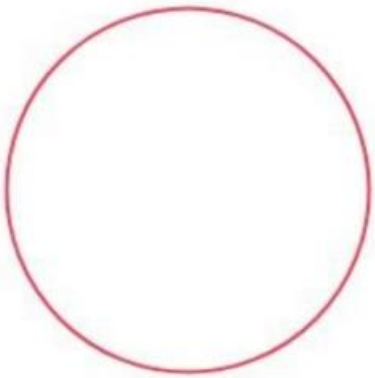
```
if (err <= 0) {
    err += 2*y + 1;
}
// Mid-point is outside the perimeter
else {
    x--;
    err += 2 * (y - x) + 1;
}

// All the perimeter points have already been printed
if (x < y) {
    break;
}

// If the generated point is on the line x = y, then
// the perimeter points have already been printed
if (x != y) {
    putpixel(xc + x, yc - y, WHITE);
    putpixel(xc - x, yc - y, WHITE);
    putpixel(xc + y, yc + x, WHITE);
    putpixel(xc - y, yc + x, WHITE);
}

// If the generated point is on the line x = y, then
// the perimeter points have already been printed
if (x != y) {
    putpixel(xc + y, yc - x, WHITE);
    putpixel(xc - y, yc - x, WHITE);
```

```
            putpixel(xc + x, yc + y, WHITE);
            putpixel(xc - x, yc + y, WHITE);
        }
    }

    delay(5000);  // Display the window for 5 seconds before closing
    closegraph(); // Close the graphics window
}

int main() {
    int xc, yc, radius;

    // Get center and radius input from the user
    printf("Enter the center (x y): ");
    scanf("%d %d", &xc, &yc);

    printf("Enter the radius: ");
    scanf("%d", &radius);

    // Call the function to draw the circle
    drawCircle(xc, yc, radius);

    return 0;
}
```

ENTER CENTER AND RADIUS

ENTER (a, b) 319, 239

ENTER r 100



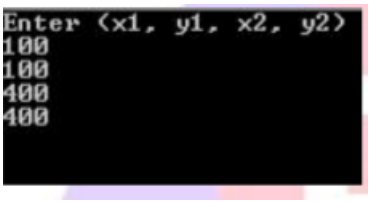
3.WAP to implement bresenham’s line drawing algorithm.

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

void drawLine(int x1, int y1, int x2, int y2) {
    // Open a graphics window
    initwindow(400, 400, "Bresenham's Line Drawing");

    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int twoDy = 2 * dy;
    int twoDx = 2 * dx;
    int twoDyMinusDx = 2 * (dy - dx);
    int twoDxMinusDy = 2 * (dx - dy);
    int decisionParameter;

    int x, y, xEnd;

    // Determine the initial decision parameter and starting point
    if (dx > dy) {
        if (x1 > x2) {
            x = x2;
            y = y2;
            xEnd = x1;
        } else {
            x = x1;
            y = y1;
            xEnd = x2;
```

```
    }
    putpixel(x, y, WHITE);
    decisionParameter = twoDy - dx;
} else {
    if (y1 > y2) {
        x = x2;
        y = y2;
        xEnd = x1;
    } else {
        x = x1;
        y = y1;
        xEnd = x2;
    }
    putpixel(x, y, WHITE);
    decisionParameter = twoDx - dy;
}

// Plot the line
while (x < xEnd) {
    x++;
    if (decisionParameter < 0) {
        decisionParameter += twoDy;
    } else {
        if (dx > dy) {
            if ((x2 - x1) > 0) {
                y++;
            } else {
                y--;
```

```
            }
        } else {
            if ((y2 - y1) > 0) {
                x++;
            } else {
                x--;
            }
        }
        decisionParameter += twoDyMinusDx;
    }
    putpixel(x, y, WHITE);
  }

  delay(5000);  // Display the window for 5 seconds before closing
  closegraph(); // Close the graphics window
}

int main() {
  int x1, y1, x2, y2;

  // Get the coordinates of the two endpoints from the user
  printf("Enter the coordinates of the first point (x1 y1): ");
  scanf("%d %d", &x1, &y1);

  printf("Enter the coordinates of the second point (x2 y2): ");
  scanf("%d %d", &x2, &y2);

  // Call the function to draw the line
```

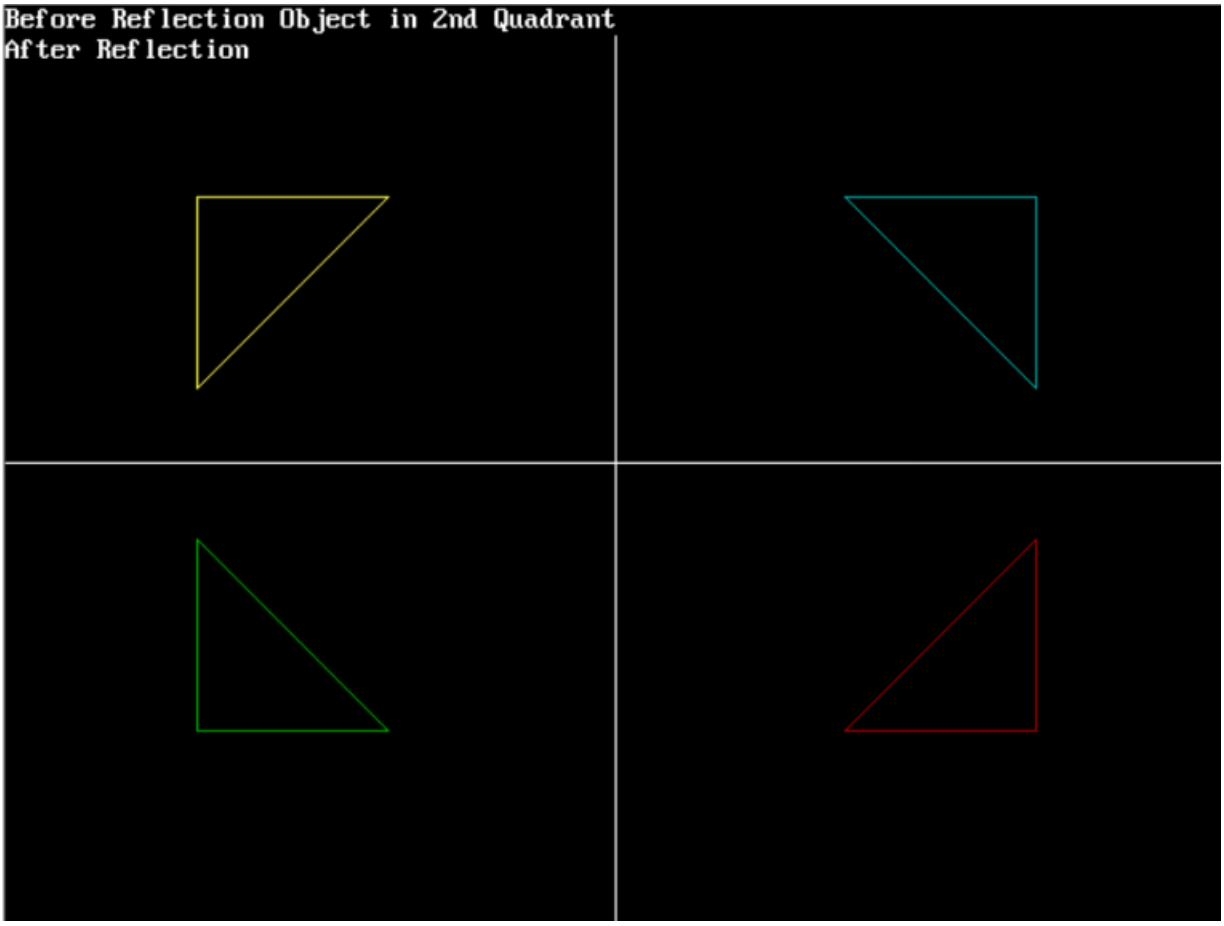
```
    drawLine(x1, y1, x2, y2);

    return 0;
}
```



4.Program to implement 2D reflection code

```
#include <conio.h>

#include <graphics.h>

#include <stdio.h>

{

// Initialize the drivers int gm, gd = DETECT, ax, x1 = 100; int x2 = 100, x3 = 200, y1 = 100; int
y2 = 200, y3 = 100;

// Add in your BGI folder path

// like below initgraph(&gd, &gm,

// "C:\\TURBOC3\\BGI");

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

cleardevice();

// Draw the graph

 line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());

line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);

// Object initially at 2nd quadrant

 printf("Before Reflection Object" in 2nd Quadrant");

// Set the color

setcolor(14);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

 line(x3, y3, x1, y1);

 getch();

}

// After reflection

printf("\nAfter Reflection");

// Reflection along origin i.e.,

// in 4th quadrant setcolor(4);
```

```
line(getmaxx() - x1, getmaxy() - y1, getmaxx() - x2, getmaxy() - y2);
line(getmaxx() - x2, getmaxy() - y2, getmaxx() - x3, getmaxy() - y3);
line(getmaxx() - x3, getmaxy() - y3, getmaxx() - x1, getmaxy() - y1);
// Reflection along x-axis i.e.,
// in 1st quadrant
setcolor(3);
    line(getmaxx() - x1, y1,getmaxx() - x2, y2);
line(getmaxx() - x2, y2,getmaxx() - x3, y3);
line(getmaxx() - x3, y3, getmaxx() - x1, y1);
// Reflection along y-axis i.e.,
// in 3rd quadrant
setcolor(2);
line(x1, getmaxy() - y1, x2, getmaxy() - y2);
 line(x2, getmaxy() - y2, x3, getmaxy() - y3);
line(x3, getmaxy() - y3, x1, getmaxy() - y1);
getch();
// Close the graphics
closegraph();
```


Before Reflection Object in 2nd Quadrant
After Reflection