University of Rajshahi

Assignment on Computer Graphics

Submitted by:

RAKIBUL ISLAM
Id: 1810876111
University of Rajshahi

Submitted to:

ABU RAIHAN SHOYEB AHMED SIDDIQUE
PROFESSOR
University of Rajshahi

Submitted on 12-April-2022

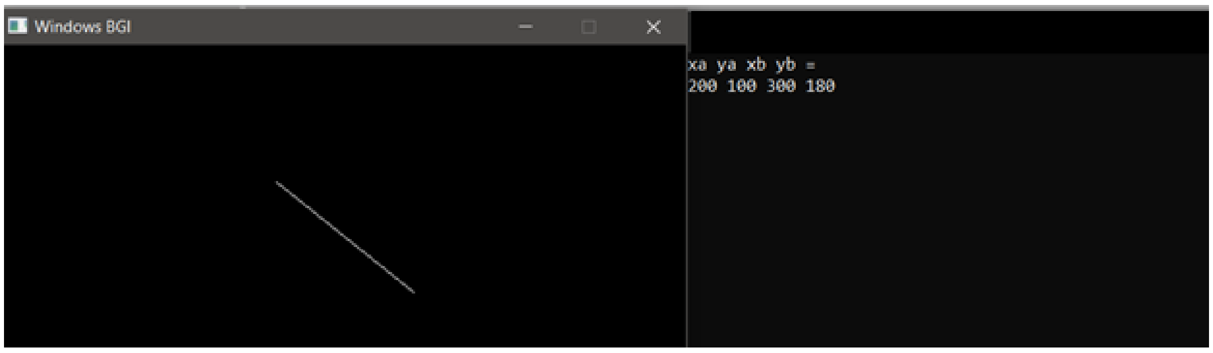
Bresenham's Line Drawing

Source code:

```
include <graphics.h>
#include <stdio.h>
#include <conio.h>
#define color WHITE

void lineBres(int xa, int ya, int xb, int yb)
{
    initwindow(500, 500);
    int dx = abs(xa - xb), dy = abs(ya - yb);
    int p = 2 * dy - dx;
    int twoDy = 2 * dy, twoDyDx = 2 * (dy - dx);
    int x, y, xEnd;

    if(xa > xb)
    {
        x = xb;
        y = yb;
        xEnd = xa;
    }
    else
    {
        x = xa;
        y = ya;
        xEnd = xb;
    }

    putpixel(x, y, color);

    while(x < xEnd)
    {
        x++;
        if(p < 0)
            p += twoDy;
        else
        {
            y++;
            p += twoDyDx;
        }
        putpixel(x, y, color);
    }

}
```

```
int main()
{
    int xa, ya, xb, yb;
    printf("xa ya xb yb = \n");
    scanf("%d %d %d %d", &xa, &ya, &xb, &yb);
    lineBres(xa, ya, xb, yb); // For testing: 200, 100, 300, 180
    getch();
    return 0;
}
```

Output:



```
xa ya xb yb =
200 100 300 180
```

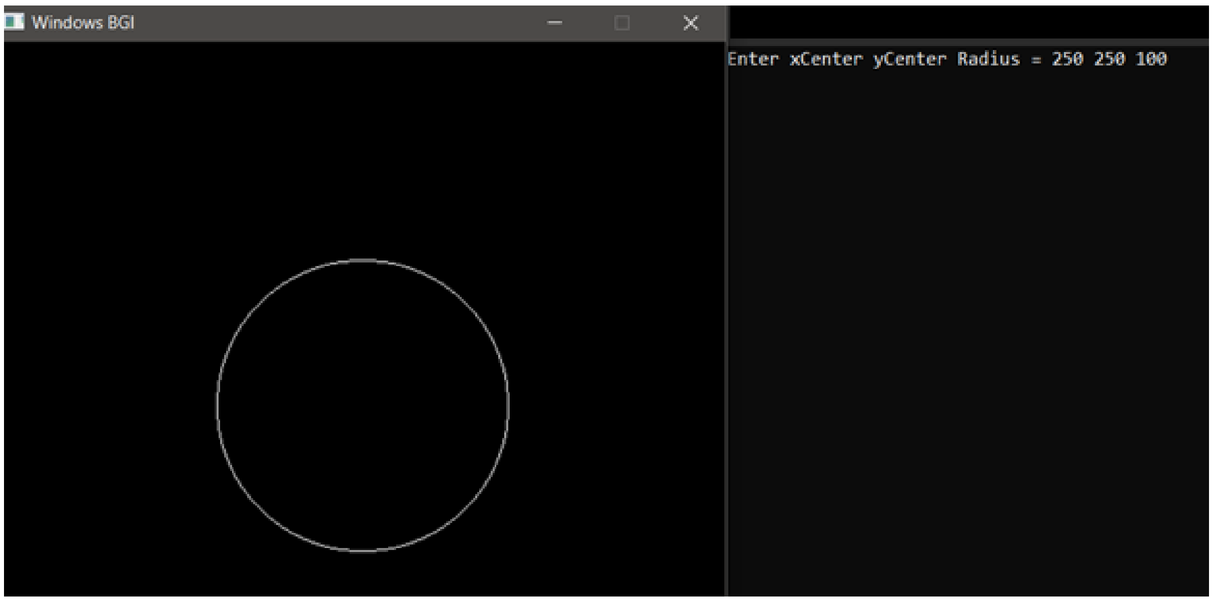
Midpoint circle drawing

Source code:

```
#include <graphics.h>
#include <stdio.h>
#define color WHITE

void circleMidpoint(int xCenter, int yCenter, int radius)
{
    initwindow(500, 500);
    int x = 0;
    int y = radius;
    int p = 1 - radius;
    void circlePlotPoints(int, int, int, int);

    circlePlotPoints(xCenter, yCenter, x, y);

    while(x < y)
    {
        x++;
        if(p < 0)
            p += 2 * x + 1;
        else
        {
            y--;
            p += 2 * (x - y) +1;
        }
        circlePlotPoints(xCenter, yCenter, x, y);
    }
}

void circlePlotPoints(int xCenter, int yCenter, int x, int y)
{
    putpixel(xCenter + x, yCenter + y, color);
    putpixel(xCenter - x, yCenter + y, color);
    putpixel(xCenter + x, yCenter - y, color);
    putpixel(xCenter - x, yCenter - y, color);

    putpixel(xCenter + y, yCenter + x, color);
    putpixel(xCenter - y, yCenter + x, color);
    putpixel(xCenter + y, yCenter - x, color);
    putpixel(xCenter - y, yCenter - x, color);
}
```

```
int main()
{
    int xCenter, yCenter, radius;
    scanf("%d %d %d", &xCenter, &yCenter, &radius);
    circleMidpoint(xCenter, yCenter, radius); // For testing: 250, 250, 100
    getch();
}
```

Output:



```
Enter xCenter yCenter Radius = 250 250 100
```

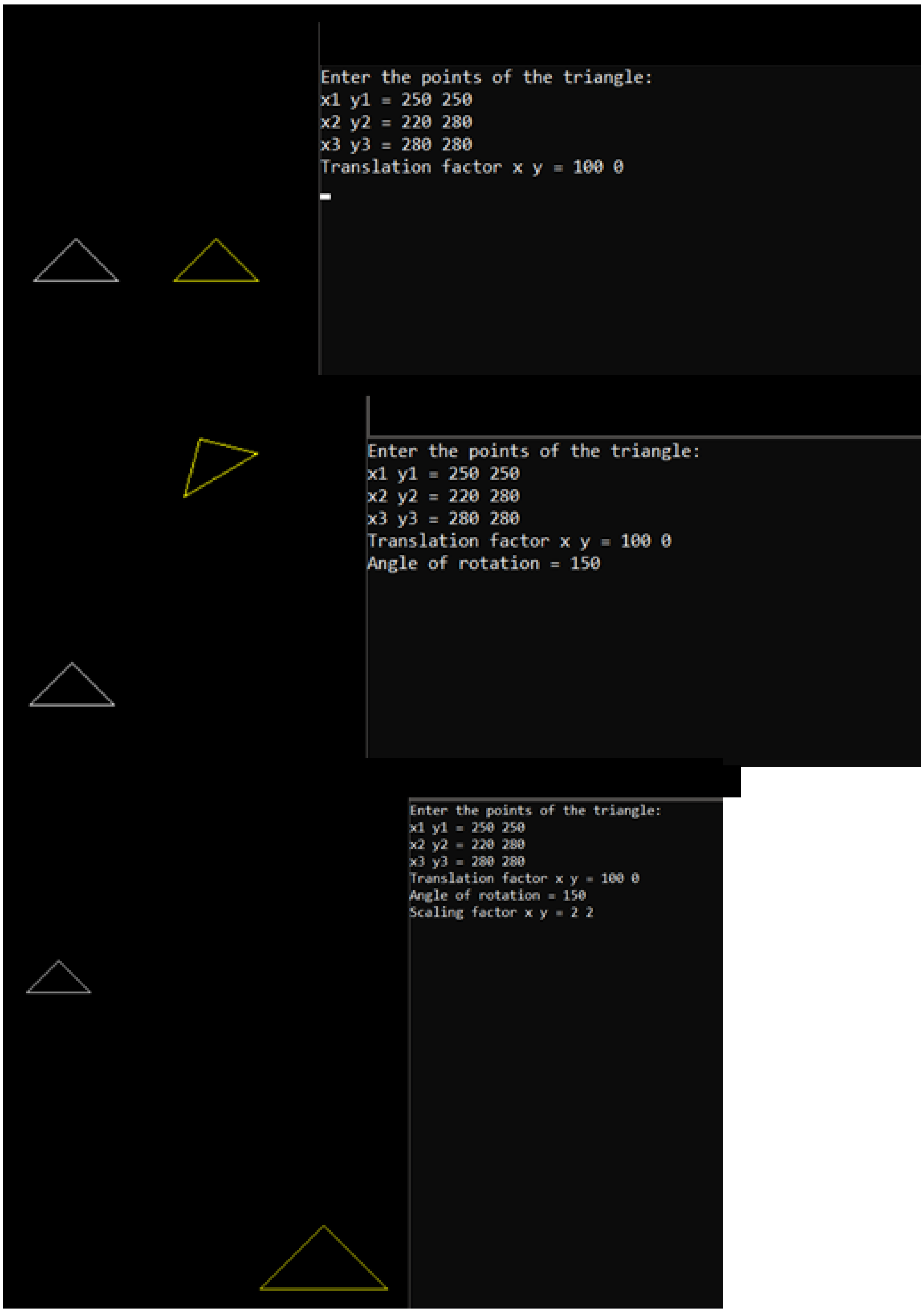
2D Translation, Rotation and Scaling

Source code:

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

int x1,x2,x3,y1,y2,y3,nx1,nx2,nx3,ny1,ny2,ny3;
int sx,sy,xt,yt,r;
float t;

void drawTriangle()
{
   line(x1,y1,x2,y2);
   line(x2,y2,x3,y3);
   line(x3,y3,x1,y1);
}

void translation()
{
   drawTriangle();

   setcolor(YELLOW);

   printf("Translation factor x y = ");
   scanf("%d %d",&xt,&yt);

   nx1=x1+xt;
   ny1=y1+yt;

   nx2=x2+xt;
   ny2=y2+yt;

   nx3=x3+xt;
   ny3=y3+yt;

   line(nx1,ny1,nx2,ny2);
   line(nx2,ny2,nx3,ny3);
   line(nx3,ny3,nx1,ny1);

   setcolor(WHITE);
   getch();
   cleardevice();
}
```

```
void rotation()
{
    drawTriangle();

    setcolor(YELLOW);

    printf("Angle of rotation = ");
    scanf("%d",&r);

    t=3.14*r/180;

    nx1=abs(x1*cos(t)-y1*sin(t));
    ny1=abs(x1*sin(t)+y1*cos(t));

    nx2=abs(x2*cos(t)-y2*sin(t));
    ny2=abs(x2*sin(t)+y2*cos(t));

    nx3=abs(x3*cos(t)-y3*sin(t));
    ny3=abs(x3*sin(t)+y3*cos(t));

    line(nx1,ny1,nx2,ny2);
    line(nx2,ny2,nx3,ny3);
    line(nx3,ny3,nx1,ny1);

    setcolor(WHITE);
    getch();
    cleardevice();
}

void scaling()
{
    drawTriangle();

    setcolor(YELLOW);

    printf("Scaling factor x y = ");
    scanf("%d %d",&sx,&sy);

    nx1=x1*sx;
    ny1=y1*sy;

    nx2=x2*sx;
    ny2=y2*sy;

    nx3=x3*sx;
    ny3=y3*sy;
    line(nx1,ny1,nx2,ny2);
    line(nx2,ny2,nx3,ny3);
    line(nx3,ny3,nx1,ny1);

    setcolor(WHITE);
    getch();
    cleardevice();
}

int main()
{

    printf("Enter the points of the triangle:\n");
    printf("x1 y1 = ");
    scanf("%d %d", &x1,&y1);
    printf("x2 y2 = ");
    scanf("%d %d", &x2,&y2);
    printf("x3 y3 = ");
    scanf("%d %d", &x3,&y3);
    initwindow(1000, 1000);

    translation();
    rotation();
    scaling();

    getch();
    return 0;
}
```

Output:



```
Enter the points of the triangle:
x1 y1 = 250 250
x2 y2 = 220 280
x3 y3 = 280 280
Translation factor x y = 100 0
```

```
Enter the points of the triangle:
x1 y1 = 250 250
x2 y2 = 220 280
x3 y3 = 280 280
Translation factor x y = 100 0
Angle of rotation = 150
```

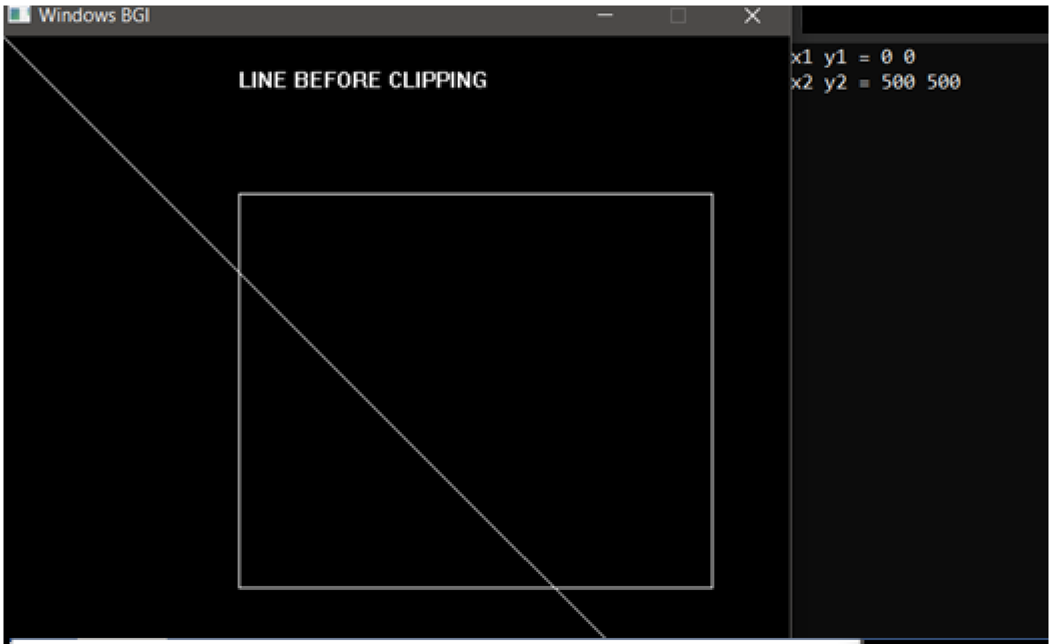
```
Enter the points of the triangle:
x1 y1 = 250 250
x2 y2 = 220 280
x3 y3 = 280 280
Translation factor x y = 100 0
Angle of rotation = 150
Scaling factor x y = 2 2
```

Line Clipping

Source code: [1]

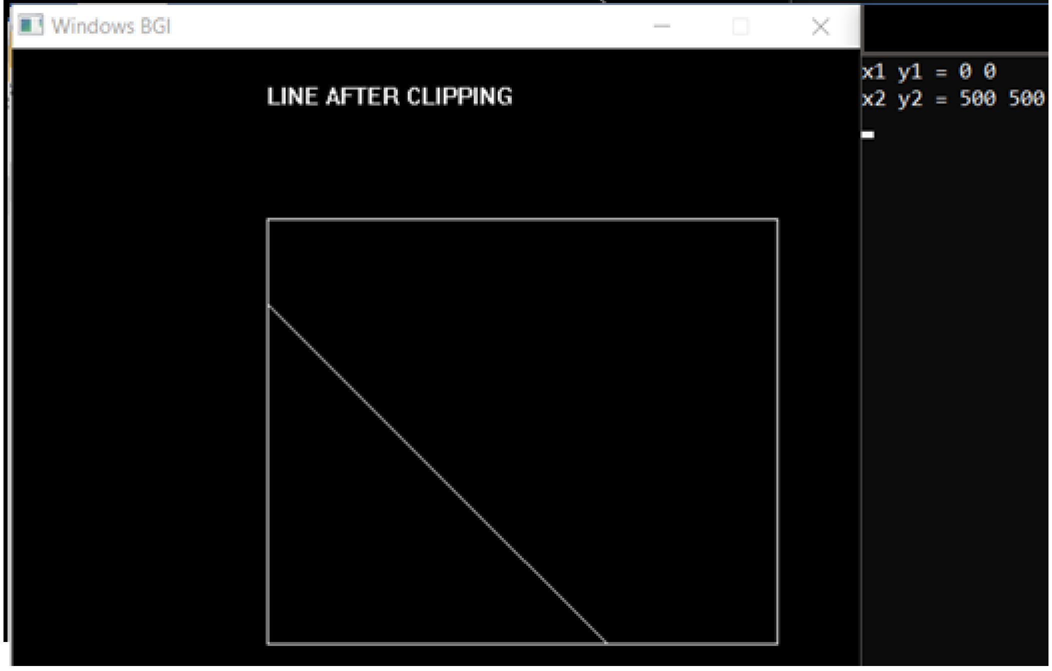
```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>

typedef struct coordinate
{
    int x,y;
    char code[4];
}PT;

void drawwindow();
void drawline(PT p1,PT p2);
PT setcode(PT p);
int visibility(PT p1,PT p2);
PT resetendpt(PT p1,PT p2);

int main()
{
    int v;
    PT p1,p2,p3,p4,ptemp;

    printf("x1 y1 = ");
    scanf("%d %d",&p1.x,&p1.y);

    printf("x2 y2 = ");
    scanf("%d %d",&p2.x,&p2.y);

    initwindow(500, 500);
    drawwindow();
    delay(500);

    outtextxy(150, 20, "LINE BEFORE CLIPPING");
    drawline(p1,p2);
    getch();

    cleardevice();
    delay(500);

    outtextxy(150, 20, "LINE AFTER CLIPPING");
    p1=setcode(p1);
    p2=setcode(p2);
    v=visibility(p1,p2);
    delay(500);

    switch(v)
    {
        case 0:
            drawwindow();
            delay(500);
            drawline(p1,p2);
            break;

        case 1:
            drawwindow();
            delay(500);
            break;
        case 2:
            p3=resetendpt(p1,p2);
            p4=resetendpt(p2,p1);
            drawwindow();
            delay(500);
            drawline(p3,p4);
            break;
    }

    getch();
    closegraph();
    return 0;
}

void drawwindow()
{
    line(150,100,450,100);
    line(450,100,450,350);
    line(450,350,150,350);
    line(150,350,150,100);
}

void drawline(PT p1,PT p2)
{
    line(p1.x,p1.y,p2.x,p2.y);
}

PT setcode(PT p)
{
    PT ptemp;
    if(p.y<100)
        ptemp.code[0]='1';
    else
        ptemp.code[0]='0';

    if(p.y>350)
        ptemp.code[1]='1';
    else
        ptemp.code[1]='0';

    if(p.x>450)
        ptemp.code[2]='1';
    else
        ptemp.code[2]='0';

    if(p.x<150)
        ptemp.code[3]='1';
    else
        ptemp.code[3]='0';

    ptemp.x=p.x;
    ptemp.y=p.y;

    return(ptemp);
}

int visibility(PT p1,PT p2)
{
    int i,flag=0;

    for(i=0;i<4;i++)
    {
        if((p1.code[i]!='0') || (p2.code[i]!='0'))
        flag=1;
    }

    if(flag==0)
        return(0);

    for(i=0;i<4;i++)
    {
        if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))
            flag='0';
    }

    if(flag==0)
        return(1);
    return(2);
}

PT resetendpt(PT p1,PT p2)
{
    PT temp;
    int x,y,i;
    float m,k;

    if(p1.code[3]=='1')
        x=150;

    if(p1.code[2]=='1')
        x=450;

    if((p1.code[3]=='1') || (p1.code[2]=='1'))
    {
        m=(float)(p2.y-p1.y)/(p2.x-p1.x);
        k=(p1.y+(m*(x-p1.x)));

        temp.y=k;
        temp.x=x;

        for(i=0;i<4;i++)
            temp.code[i]=p1.code[i];

        if(temp.y<=350 && temp.y>=100)
            return (temp);
    }

    if(p1.code[0]=='1')
        y=100;

    if(p1.code[1]=='1')
        y=350;

    if((p1.code[0]=='1') || (p1.code[1]=='1'))
    {
        m=(float)(p2.y-p1.y)/(p2.x-p1.x);
        k=(float)p1.x+(float)(y-p1.y)/m;
        temp.x=k;
        temp.y=y;
        for(i=0;i<4;i++)
        temp.code[i]=p1.code[i];
        return(temp);
    }
    else
    return(p1);
}
```

Output:



LINE BEFORE CLIPPING

```
x1 y1 = 0 0
x2 y2 = 500 500
```



LINE AFTER CLIPPING

```
x1 y1 = 0 0
x2 y2 = 500 500
```

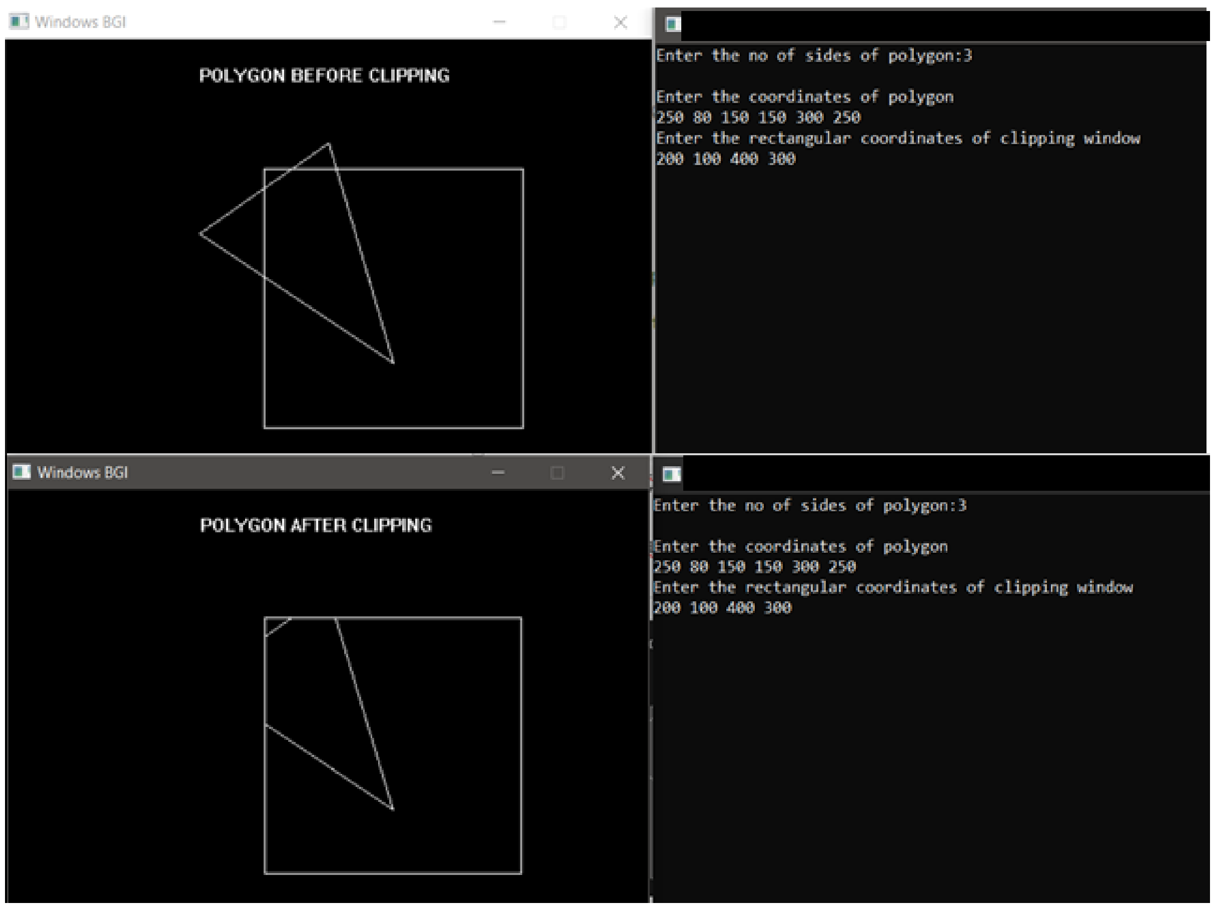
Polygon Clipping

Source code: [2]

```
Source code:
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <process.h>

#define TRUE 1
#define FALSE 0

typedef unsigned int outcode;
outcode CompOutCode(float x, float y);
enum {
   TOP = 0x1,
      BOTTOM = 0x2,
      RIGHT = 0x4,
      LEFT = 0x8
};
float xwmin, xwmax, ywmin, ywmax;
void clip(float x0, float y0, float x1, float y1) {
   outcode outcode0, outcode1, outcodeOut;
   int accept = FALSE, done = FALSE;
   outcode0 = CompOutCode(x0, y0);
   outcode1 = CompOutCode(x1, y1);
   do

   {
      if (!(outcode0 | outcode1)) {
         accept = TRUE;
         done = TRUE;
      } else
      if (outcode0 & outcode1)
         done = TRUE;
      else {
         float x, y;
         outcodeOut = outcode0 ? outcode0 : outcode1;
         if (outcodeOut & TOP) {
            x = x0 + (x1 - x0) * (ywmax - y0) / (y1 - y0);
            y = ywmax;
         } else
         if (outcodeOut & BOTTOM) {
            x = x0 + (x1 - x0) * (ywmin - y0) / (y1 - y0);
```

```
                y = ywmin;
            } else

            if (outcodeOut & RIGHT) {
                y = y0 + (y1 - y0) * (xwmax - x0) / (x1 - x0);
                x = xwmax;
            } else {
                y = y0 + (y1 - y0) * (xwmin - x0) / (x1 - x0);
                x = xwmin;
            }
            if (outcodeOut == outcode0) {
                x0 = x;
                y0 = y;
                outcode0 = CompOutCode(x0, y0);
            } else {
                x1 = x;
                y1 = y;
                outcode1 = CompOutCode(x1, y1);
            }
        }
    }
    while (done == FALSE);

    if (accept)
        line(x0, y0, x1, y1);
    outtextxy(150, 20, "POLYGON AFTER CLIPPING");
    rectangle(xwmin, ywmin, xwmax, ywmax);
}
outcode CompOutCode(float x, float y) {
    outcode code = 0;
    if (y > ywmax)
        code |= TOP;
    else
    if (y < ywmin)
        code |= BOTTOM;
    if (x > xwmax)
        code |= RIGHT;
    else
    if (x < xwmin)
        code |= LEFT;
    return code;
}
int main() {
    // For testing, Window 200,100; 400,300 and polygon 250, 80; 150, 150; 300, 250;
    initwindow(500, 500);
    float x1, y1, x2, y2;
    /* request auto detection */
    int n, poly[14], i;

    cleardevice();
    printf("Enter the no of sides of polygon:");
    scanf("%d", & n);
    printf("\nEnter the coordinates of polygon\n");
    for (i = 0; i < 2 * n; i++) {
        scanf("%d", & poly[i]);
    }
    poly[2 * n] = poly[0];
    poly[2 * n + 1] = poly[1];
    printf("Enter the rectangular coordinates of clipping window\n");
    scanf("%f%f%f%f", & xwmin, & ywmin, & xwmax, & ywmax);

    outtextxy(150, 20, "POLYGON BEFORE CLIPPING");
    drawpoly(n + 1, poly);
    rectangle(xwmin, ywmin, xwmax, ywmax);
    getch();
    cleardevice();
    for (i = 0; i < n; i++)
        clip(poly[2 * i], poly[(2 * i) + 1], poly[(2 * i) + 2], poly[(2 * i) + 3]);
    getch();
    restorecrtmode();
    return 0;
}
```

Output:



```
Enter the no of sides of polygon:3

Enter the coordinates of polygon
250 80 150 150 300 250
Enter the rectangular coordinates of clipping window
200 100 400 300
```

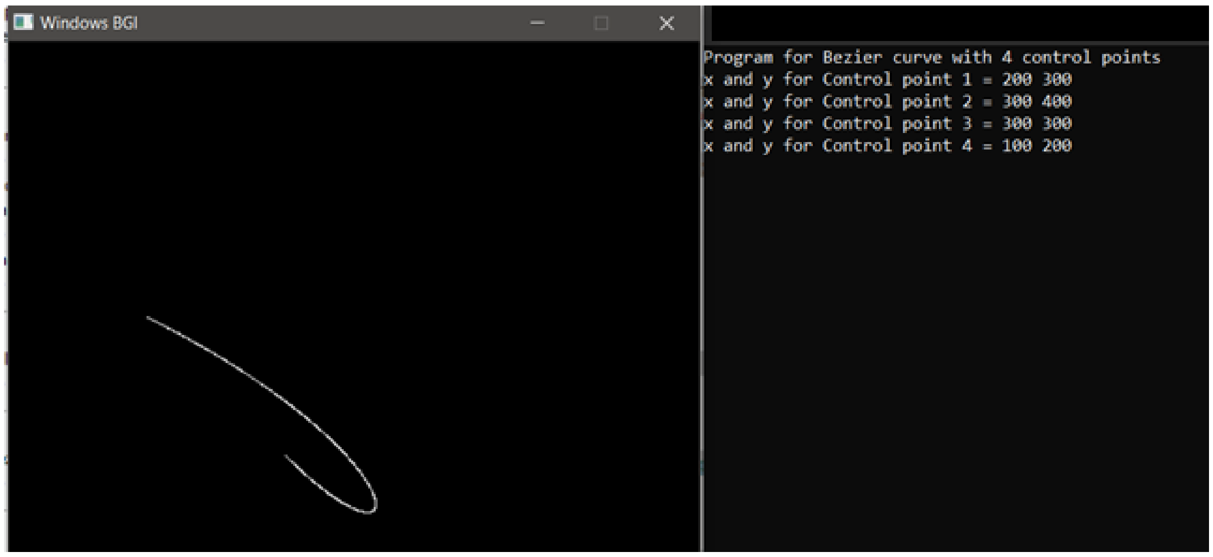
```
Enter the no of sides of polygon:3

Enter the coordinates of polygon
250 80 150 150 300 250
Enter the rectangular coordinates of clipping window
200 100 400 300
```

Bézier curve

Source code:

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

int x[4],y[4];

void curveBezier()
{


    double put_x,put_y,t;
    initwindow(500, 500);



    for(t=0.0;t<=1.0;t=t+0.001)
    {
        put_x = pow(1-t,3)*x[0] + 3*t*pow(1-t,2)*x[1] + 3*t*t*(1-t)*x[2] + pow(t,3)*x[3];
        put_y =  pow(1-t,3)*y[0] + 3*t*pow(1-t,2)*y[1] + 3*t*t*(1-t)*y[2] + pow(t,3)*y[3];

        putpixel(put_x,put_y, WHITE);
    }


}
int main()
{

    printf("Program for Bezier curve with 4 control points \n");

    for(int i=0; i<4; i++)
    {
        printf("x and y for Control point %d = ", i+1);
        scanf("%d %d",&x[i],&y[i]);
    }

    curveBezier(); // For testing: 200, 300; 300, 400; 300, 300; 100, 200
    getch();
    return 0;
}
```

Output:



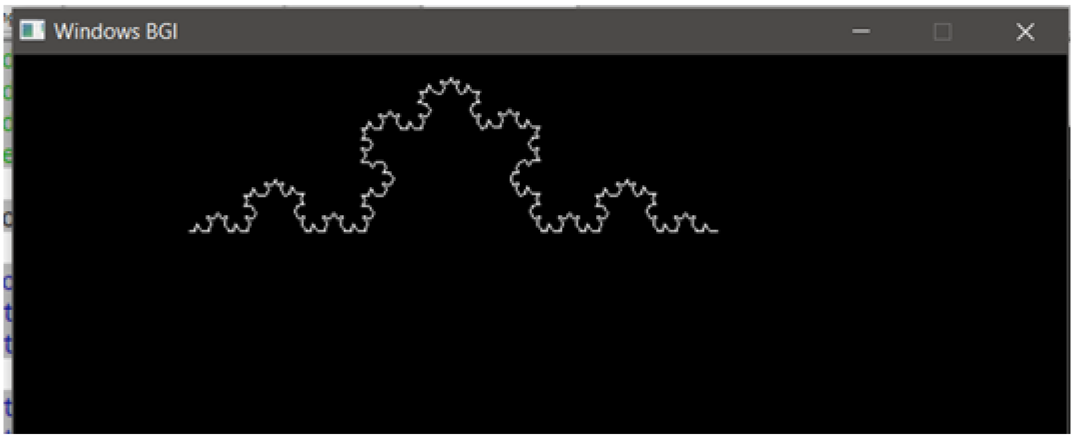
```
Program for Bezier curve with 4 control points
x and y for Control point 1 = 200 300
x and y for Control point 2 = 300 400
x and y for Control point 3 = 300 300
x and y for Control point 4 = 100 200
```

Koch Curve

Source code:

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
#define M_PI 3.14159265358979323846

void koch(int x1, int y1, int x2, int y2, int it)
{
    float angle = 60*M_PI/180;
    int x3 = (2*x1+x2)/3;
    int y3 = (2*y1+y2)/3;

    int x4 = (x1+2*x2)/3;
    int y4 = (y1+2*y2)/3;

    int x = x3 + (x4-x3)*cos(angle)+(y4-y3)*sin(angle);
    int y = y3 - (x4-x3)*sin(angle)+(y4-y3)*cos(angle);

    if(it > 0)
    {
        koch(x1, y1, x3, y3, it-1);
        koch(x3, y3, x, y, it-1);
        koch(x, y, x4, y4, it-1);
        koch(x4, y4, x2, y2, it-1);
    }
    else
    {
        line(x1, y1, x3, y3);
        line(x3, y3, x, y);
        line(x, y, x4, y4);
        line(x4, y4, x2, y2);
    }
}

int main()
{
    initwindow(600, 600);
    int x1 = 100, y1 = 100, x2 = 400, y2 = 100;
    koch(x1, y1, x2, y2, 3);
    getch();
    return 0;
}
```

Output:



Inspired from:

1. For line clipping algorithm - https://www.thecrazyprogrammer.com/2017/02/cohen-sutherland-line-clipping-algorithm.html
2. For polygon clipping algorithm - https://codewithfriend.blogspot.com/2018/10/polygon-clipping-program-in-c.html