Name: Gaurav Kishor Patil

Div: 2 Batch: C

Roll No: 54

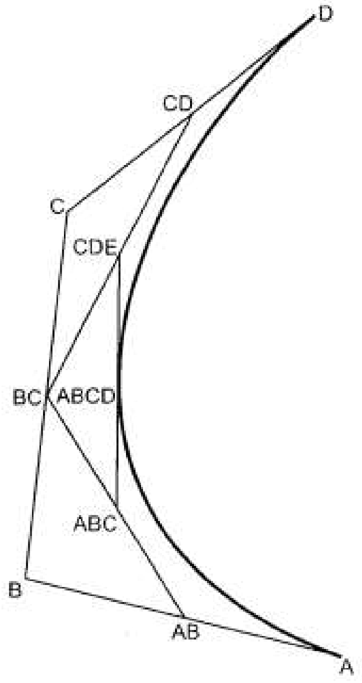
Aim: To implement Bezier curve for n control points. (Midpoint approach)

Objective:

Draw a Bezier curves and surfaces written in Bernstein basis form. The goal of interpolation is to create a smooth curve that passes through an ordered group of points. When used in this fashion, these points are called the control points.

Theory:

In midpoint approach Bezier curve can be constructed simply by taking the midpoints. In this approach midpoints of the line connecting four control points (A, B, C, D) are determined (AB, BC, CD, DA). These midpoints are connected by line segment and their midpoints are ABC and BCD are determined. Finally, these midpoints are connected by line segments and its midpoint ABCD is determined as shown in the figure –



The point ABCD on the Bezier curve divides the original curve in two sections. The original curve gets divided in four different curves. This process can be repeated to split the curve into smaller sections until we have sections so short that they can be replaced by straight lines.

Algorithm:

1) Get four control points say A(xa, ya), B(xb, yb), C(xc, yc), D(xd, yd).

2) Divide the curve represented by points A, B, C, and D in two sections.

```
    xab = (xa + xb) / 2

    yab = (ya + yb) / 2

    xbc = (xb + xc) / 2

    ybc = (yb + yc) / 2

    xcd = (xc + xd) / 2

    ycd = (yc + yd) / 2

    xabc = (xab + xbc) / 2

    yabc = (yab + ybc) / 2

    xbcd = ( xbc + xcd) / 2
    ybcd = (ybc + ycd) / 2
    xabcd = (xabc + xbcd) / 2
    yabcd = (yabc + ybcd) / 2
3) Repeat the step 2 for section A, AB, ABC, ABCD and section ABCD, BCD, CD, D.
4) Repeat step 3 until we have sections so that they can be replaced by straight lines.
5) Repeat small sections by straight lines.
6) Stop.
```

Program:

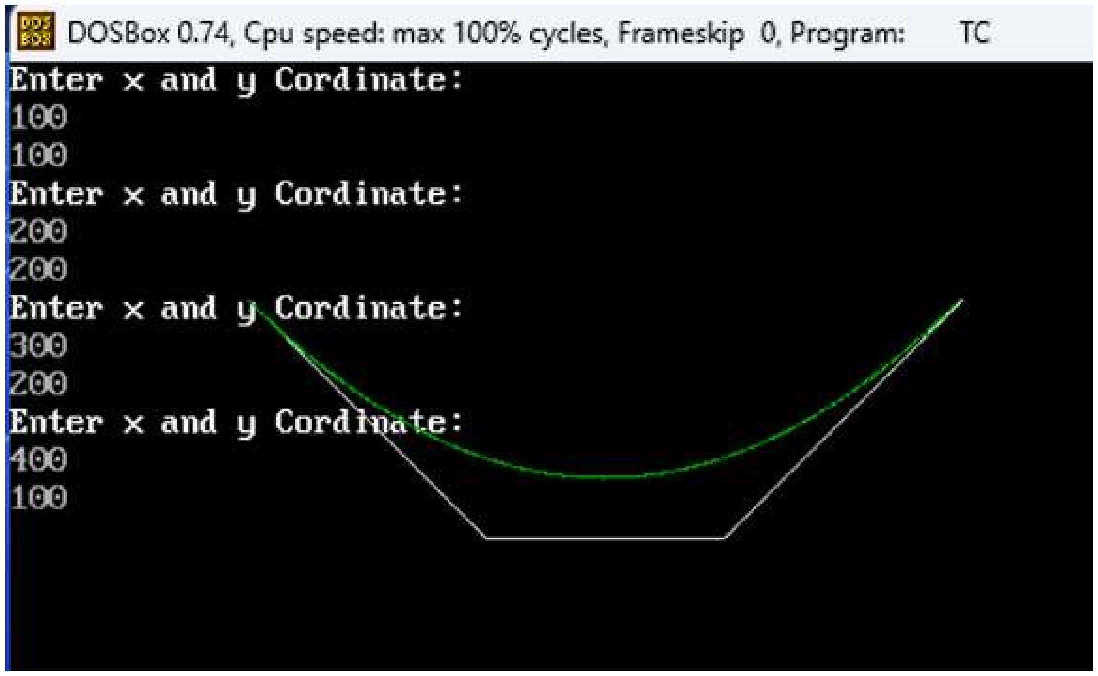
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>

void main()
{
int x[4],y[4],i;
float px,py,u;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
for(i=0;i<4;i++){
printf("Enter x and y Cordinate:\n");
scanf("%d %d",&x[i],&y[i]);
putpixel(x[i],y[i],RED);
}
line(x[0],y[0],x[1],y[1]);
line(x[1],y[1],x[2],y[2]);
line(x[2],y[2],x[3],y[3]);

for(u=0;u<=1.0;u=u+0.001){
px=x[0]*pow(1-u,3)+x[1]*pow(1-u,2)*3*u+x[2]*3*pow(u,2)*(1-u)+x[3]*pow(u,3);
py=y[0]*pow(1-u,3)+y[1]*pow(1-u,2)*3*u+y[2]*3*pow(u,2)*(1-u)+y[3]*pow(u,3);
putpixel(px,py,18);
}
getch();
getch();
closegraph();

}
```

Output:



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter x and y Cordinate:
100
100
Enter x and y Cordinate:
200
200
Enter x and y Cordinate:
300
200
Enter x and y Cordinate:
400
100
```

Conclusion – Comment on

1. Difference from arc and line
2. Importance of control point
3. Applications