Experiment No. 8

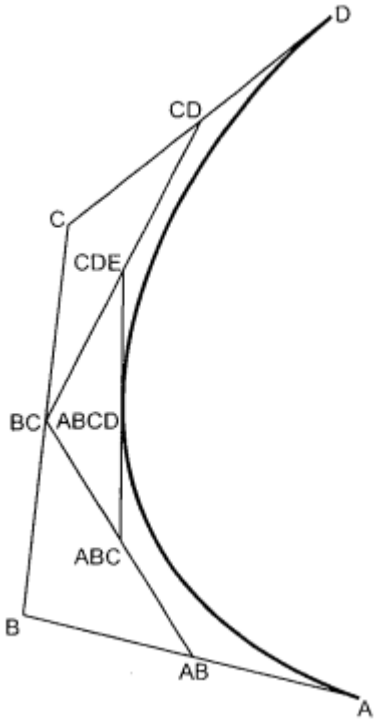
Aim: To implement Bezier curve for n control points. (Midpoint approach)

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.

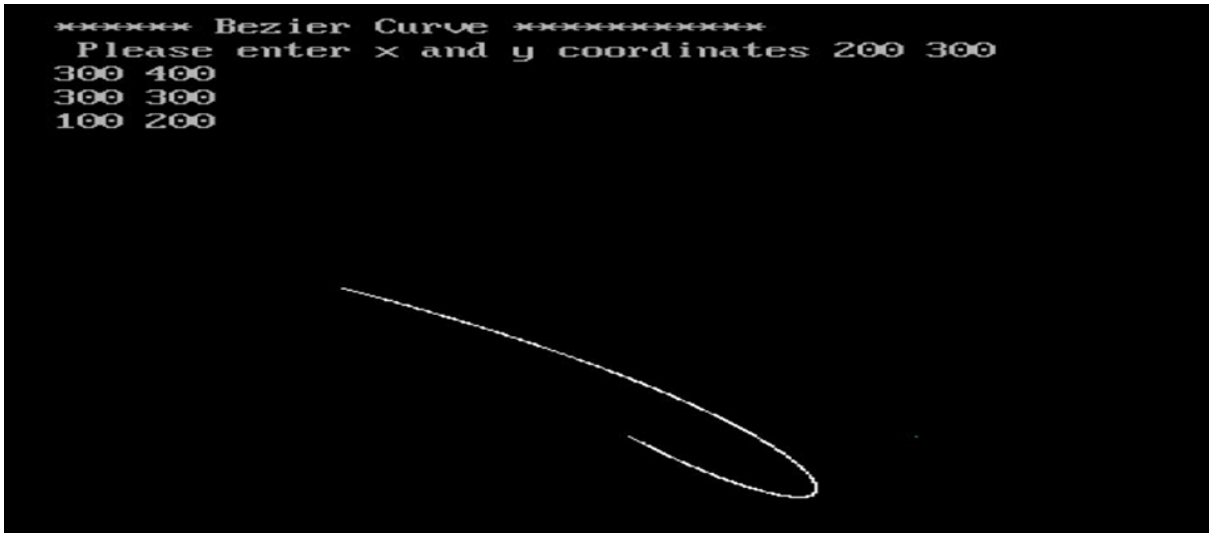
Program:

```
#include<graphics.h>

#include<math.h>
```

```
#include<conio.h>

#include<stdio.h>

void main()

{

int x[4],y[4],i;

double put_x,put_y,t;

int gr=DETECT,gm;

initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");

printf("\n****** Bezier Curve ***********");

printf("\n Please enter x and y coordinates ");

 for(i=0;i<4;i++)

  {

   scanf("%d%d",&x[i],&y[i]);

   putpixel(x[i],y[i],3);

  }


 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]; // Formula to draw curve

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);

}
```

```
getch();

closegraph();

}
```

Output:



Conclusion – Comment on

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

1. Difference from Arc and Line:

- An arc is a segment of a curve, typically part of a circle or ellipse, while a line is a straight path connecting two points.

- Arcs are curved, and lines are straight. Arcs have a radius or radii, and lines have a length.

- Lines have two endpoints, whereas arcs have a center point, radius, and may have start and end angles.

2. Importance of Control Point:

- Control points are essential in defining curves, including Bézier curves. They influence the shape and behavior of the curve.

- In Bézier curves, control points determine the direction and extent of curvature, providing precise control over the curve's path.

- The position of control points affects the shape and characteristics of the curve, making them crucial in design and animation.

3. Applications:

- Lines are fundamental elements in graphics and geometry, used for drawing shapes, paths, and vectors.

- Arcs are used in various applications, such as creating circular or elliptical shapes in design and engineering.

- Bézier curves with control points are extensively used in computer graphics, design software (e.g., Adobe Illustrator), and animations for creating smooth and precise curves in shapes, fonts, and animations.