Experiment No. 8
Topic : To implement Bezier curve for n control points.
Name: Chirag raut
Roll Number: 50
Date of Performance:
Date of Submission:

Experiment No. 8

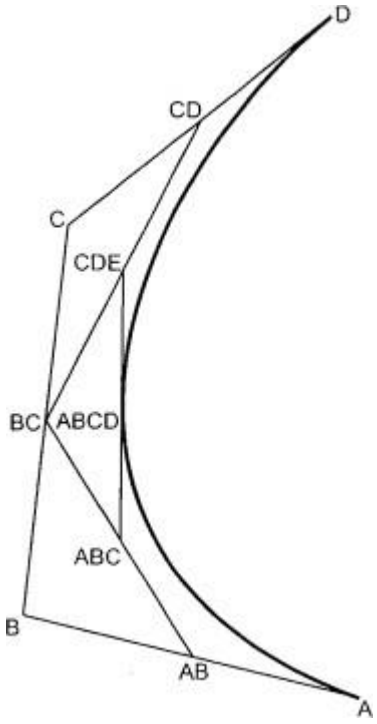
Aim: To implement Bezier curve for n control points. (Midpoint approach)

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.

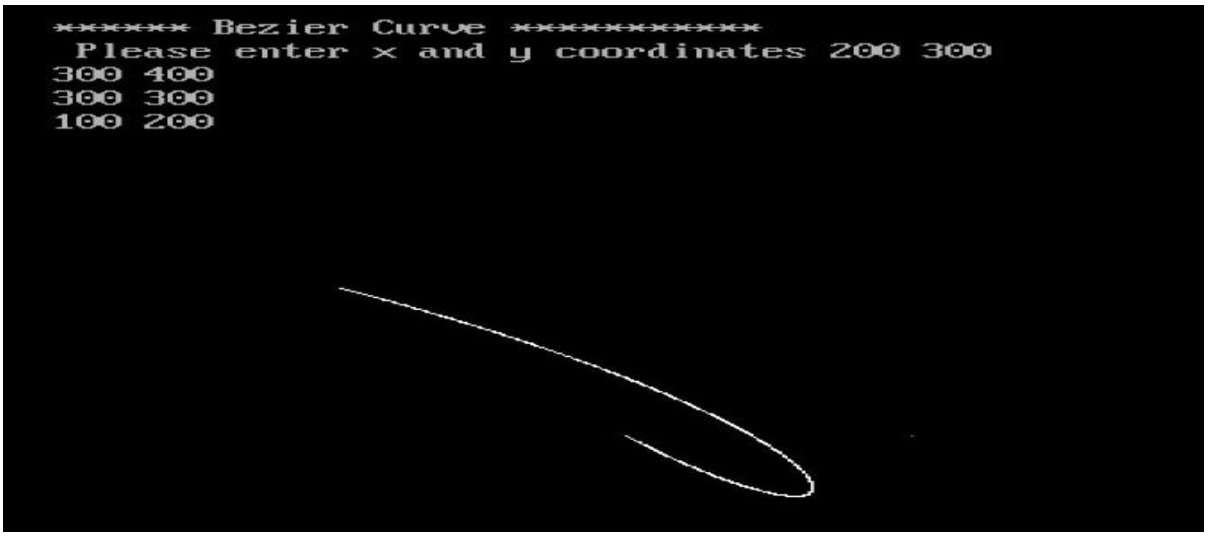
6) Stop.

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

Difference from arc and line:

Bezier curves differ from arcs and lines in their mathematical representation and the way they interpolate points. Unlike arcs that are defined by a radius and angle, and lines that have a constant slope, Bezier curves are defined by a set of control points that influence the shape of the curve. Bezier curves can produce more complex and flexible shapes than arcs and lines.

Importance of control points:

Control points are crucial in determining the shape and characteristics of a Bezier curve. They define the direction and extent of the curve segments, allowing for smooth and precise control over the curve's shape. By adjusting the positions and weights of control points, users can manipulate the curve's behavior and produce a wide variety of shapes, from simple curves to more complex and intricate designs.

Applications:

Bezier curves find extensive use in various fields, including computer graphics, animation, and industrial design. Some common applications include:

Computer Aided Design (CAD) software for creating and editing curves and surfaces.

Graphic design software for creating smooth and intricate shapes such as fonts, logos, and illustrations.