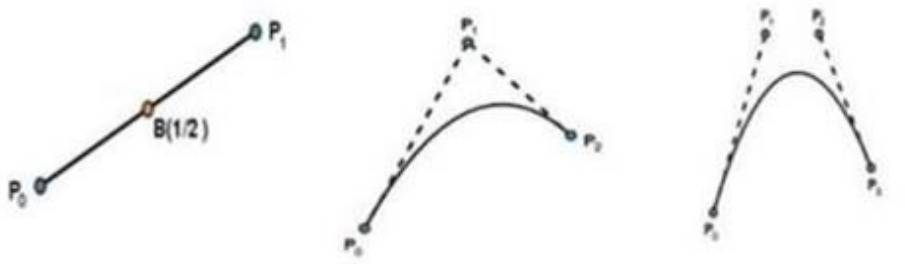
Name:	MAHEK SHAH
Roll No:	
Class/Sem:	SE/III
Experiment No.:	9
Title:	Bezier Curve for n control points
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Experiment No. 9

Aim :Write a program to implement Bezier Curve for n control points in C.

Objective:

A Bezier curve is a parametric curve used in computer graphics and related fields. The curve, which is related to the Bernstein polynomial, is named after Pierre Bezier, who used it in the 1960s for designing curves for the bodywork of Renault cars.



P(u) = Pi Bi,n(u)

0 ≤ u ≤ 1

where, Pi = control points

Bi,n / BEZ i,n = Bezier function or Berstein Polynomials.

The Bernstein polynomial or the Bezier function is very important function will dictate the smoothness of this curve & the weight will be dictated by boundary conditions.

BEZ i,n (u) = nCi. ui (1-u)n-i

$$\frac{n!}{i!(n-i)!}$$

where nCi = [Binomial Coefficient]

P(u) = Pi Bi,n(u)

P(u) = $\sum^{n}$ = P0 B0,3 (u) + P1 B1,3 (u) + P2 B2,3 (u) + P3 B3,3 (u) --------- (1)

$$[u^3 \quad u^2 \quad u \quad 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

where, B 0,3 (u) = 3C0. u0 (1-u)3

= 3! / 0! (3-0)! .1 (1-u)3

= (1-u)3

Similarly,

B1,3 = 3u. (1-u)2

B2,3 = 3u2. (1-u)

B3,3 = u3

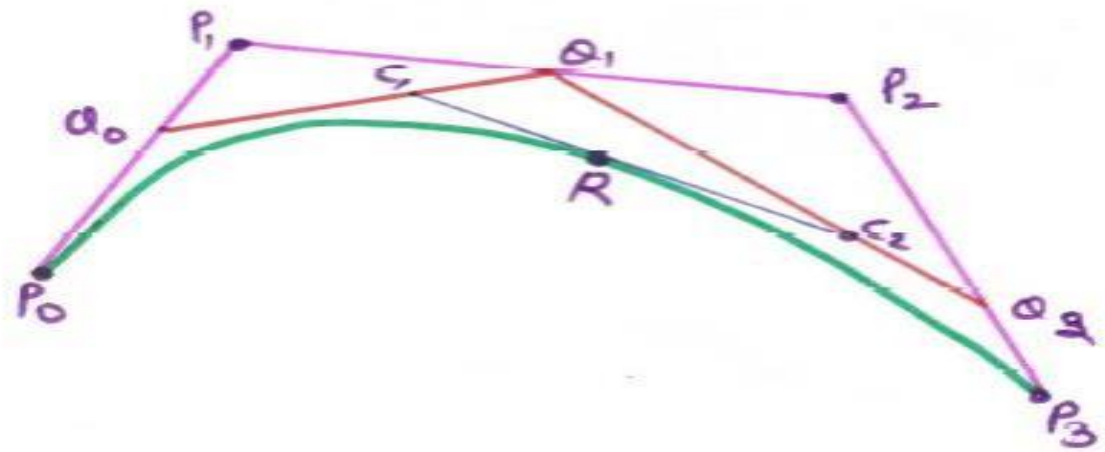
Now substituting these value in equation 1 we get

P(u) = P0 (1-u)3 + P1 3u. (1-u)2 + P2 3u2. (1-u) + P3 u3

x(u) = (1-u)3 x0 + 3u(1-u)2 x1 + 3u2(1-u) x2 + u3 x3

y(u) = (1-u)3 y0 + 3u(1-u)2 y1 + 3u2(1-u) y2 + u3 y3

z(u) = (1-u)3 z0 + 3u(1-u)2 z1 + 3u2(1-u) z2 + u3 z3



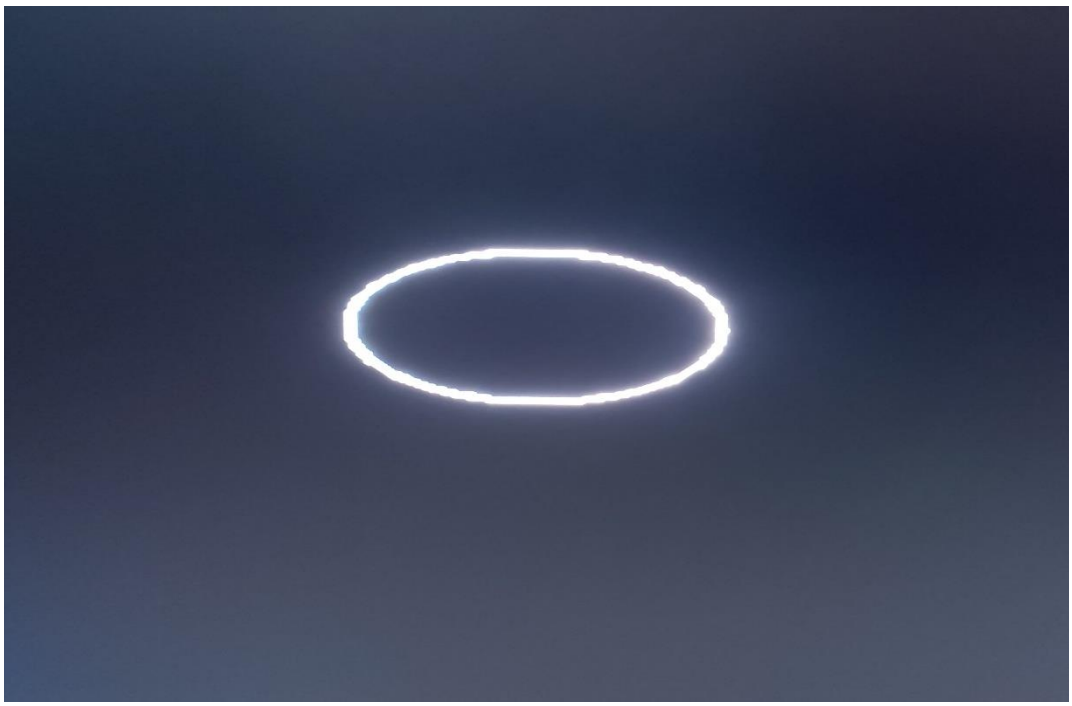
Bezier curve can fit any number of control points Reversing the order of control points yields the same bezier curve The curve begins at P0 and ends at Pn this is the so-called endpoint interpolation property.
The curve is a straight line if and only if all the control points are collinear.

Code :

```
#include <stdio.h>
#include <graphics.h>
int binomialCoefficient(int n, int k) {
    if (k == 0 || k == n) {
        return 1;
    } else {
        return binomialCoefficient(n - 1, k - 1) + binomialCoefficient(n - 1,
k);
    }
}
float bezierPoint(int n, int i, float t, int *ctrlPoints) {
    return binomialCoefficient(n, i) * pow(1 - t, n - i) * pow(t, i) *
ctrlPoints[i];
}
void drawBezierCurve(int n, int *ctrlPoints) {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    float t;
    for (t = 0.0; t <= 1.0; t += 0.01) {
        float x = 0.0, y = 0.0;
        for (int i = 0; i <= n; i++) {
            x += bezierPoint(n, i, t, ctrlPoints);
        }
        t += 0.01;
        for (int i = 0; i <= n; i++) {
            y += bezierPoint(n, i, t, ctrlPoints);
        }
        putpixel(round(x), round(y), WHITE);
    }
    delay(5000);
    closegraph();
}
int main() {
```

```
    int n;
    printf("Enter the number of control points: ");
    scanf("%d", &n);
    int ctrlPoints[n + 1];
    printf("Enter the control points:\n");
    for (int i = 0; i <= n; i++) {
        printf("P%d: ", i);
        scanf("%d", &ctrlPoints[i]);
    }
    drawBezierCurve(n, ctrlPoints);
    return 0;
}
```

Output



Conclusion :

The Bezier curve is a versatile and widely used method for defining smooth curves in computer graphics. This implementation allows users to input an arbitrary number of control points, making it suitable for creating curves of varying complexity.

One key advantage of Bezier curves is their ability to represent a wide range of shapes, from simple curves to more intricate forms. The recursive nature of the binomial coefficient calculation provides an elegant and efficient solution for evaluating points along the curve.

However, it's essential to note that Bezier curves, particularly with a large number of control points, can be computationally expensive. In practice, optimization techniques and more specialized algorithms, such as de Casteljau's algorithm, are often employed to enhance performance.

Despite potential computational complexities, Bezier curves remain a fundamental concept in computer graphics and find applications in areas such as 3D modeling, animation, and font design. Their mathematical elegance and flexibility make them a valuable tool for representing and manipulating curves in digital environments.