|  |  |
| --- | --- |
| **Name:** | Yash Ravindra Kerkar |
| **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** | To implement Bezier Curve which uses control points to draw a curve. |
| **Theory** | 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.    **Simple Bezier Curve Quadratic Bezier Curve Cubic Bezier Curve**   * Let suppose we are given (n+1) control points position then Pi = (xi, yi, zi) from 0 to n. * These coordinate points can be blended to produce the following position vector P(u), which describes the path of an approximation. * So Bezier polynomial function between P0 to Pn is 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  where nCi =  [Binomial Coefficient]  **P(u) = Pi Bi,n(u)** |

P(u) = = P0 B0,3 (u) + P1 B1,3 (u) + P2 B2,3 (u) + P3 B3,3 (u) (1)

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

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

= (1-u)3

Similarly,

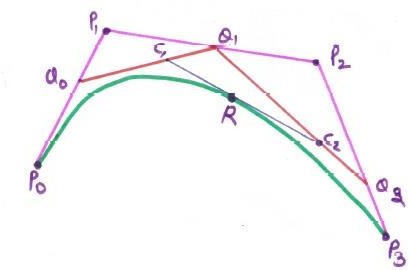
B1,3 = 3u. (1-u)2

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



By using a line parametric equation we can derive Bezier Curve equation for any no. of control points -

Q0 = (1-u) P0 + u P1 Q1 = (1-u) P1 + u P2 Q2 = (1-u) P2 + u P3 C1 = (1-u) Q0 + u Q1 C2 = (1-u) Q1 + u Q2 R = (1-u) C1 + u C2

Substituting the values of C1, C2, Q0, Q1, Q2 in R we get- R = (1-u) [(1-u) Q0 + u Q1] + u [(1-u) Q1 + u Q2]

R = (1-u)2 Q0 + u (1-u) Q1 + u (1-u) Q1 + u2 Q2 R = (1-u)2 Q0 + 2 u (1-u) Q1 + u2 Q2

R = (1-u)2 [(1-u) P0 + u P1] + 2 u (1-u) [(1-u) P1 + u P2] + u2 [(1-u) P2 + u P3] R = (1-u)3 P0 + u (1-u)2 P1 + 2 u (1-u)2 P1 + 2 u2 (1-u) P2 + u2 (1-u) P2 + u3 P3 **R = (1-u)3 P0 + 3u (1-u)2 P1 + 3 u2 (1-u) P2 + u3 P3**

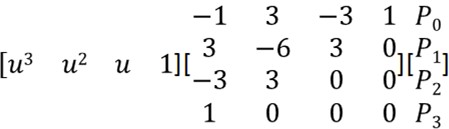
Thus we have received the same equation which we get from Bernstein Polynomial Form

The equation of Bezier Curve is -

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

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

P(u) = P0 (-u3+3u2-3u+1) + P1 (3u3-6u2+3u) +P2 (-3u3+3u2) + P3 u3 P(u) =



P(u) = U. MBEZ. GBEZ

where MBEZ & GBEZ represents Bezier basis matrix and Bezier geometric matrix respectively.



|  |  |
| --- | --- |
| **Properties** | * Always interpolates first and last control points and approximate the remaining two * The slope of the derivative at the beginning is along the line joining the first two points and slope of the derivative at the end is along the line joining the last two points * The degree of the curve is 1 less than the number of control points * Bezier curve always satisfies convex hull property * Bezier curves do not have local control, repositioning one control point changes the entire curve * 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<graphics.h>  #include<math.h>  int x[4],y[4];  void bezier(int x[4],int y[4])  {  int gd=DETECT,gm,i;  double t,xt,yt;  initgraph(&gd,&gm," ");  for(t=0.0;t<1.0;t+=0.0005)  {  xt=pow((1.0-t),3)\*x[0]+3\*t\*pow((1.0-t),2)\*x[1]+3\*pow(t,2)\*(1.0-t)\*x[2]+pow(t,3)\*x[3];  yt=pow((1.0)-t,3)\*y[0]+3\*t\*pow((1.0)-t,2)\*y[1]+3\*pow(t,2)\*(1.0-t)\*y[2]+pow(t,3)\*y[3];  putpixel(xt,yt,4);  delay(5);  }  for(i=0;i<4;i++)  {  putpixel(x[i],y[i],5);  circle(x[i],y[i],2);  delay(2);  }  getch();  closegraph();  }  int main()  {  int i,x[4],y[4];  printf("Enter the four control points : ");  for(i=0;i<4;i++)  {  scanf("%d %d",&x[i],&y[i]);  }  bezier(x,y);  return 0;  } |
| **Output** |  |
|  |  |

**Conclusion:** In this practical, we implemented a program to draw a Bezier curve using the graphics library in C. Bezier curves are essential in computer graphics and design for creating smooth and complex shapes. The program allows the user to input four control points, and it calculates and draws the Bezier curve passing through these points.