Program 12

Aim: Write A program to draw an arbitrary curve using Hermite and Bezier curve drawing method.

Theory:

A Hermite curve is a spline where every piece is a third-degree polynomial defined in Hermite form: that is, by its values and initial derivatives at the end points of the equivalent domain interval. Cubic Hermite splines are normally used for interpolation of numeric values defined at certain dispute values x_1, x_2, x_3, … x_n, to achieve a smooth continuous function. The data should have the preferred function value and derivative at each Xk. The Hermite formula is used to every interval (X_k, X_{k+1}) individually. The resulting spline become continuous and will have first derivative.

Cubic polynomial splines are specially used in computer geometric modeling to attain curves that pass via defined points of the plane in 3D space. In these purposes, each coordinate of the plane is individually interpolated by a cubic spline function of a divided parameter ‘t’.

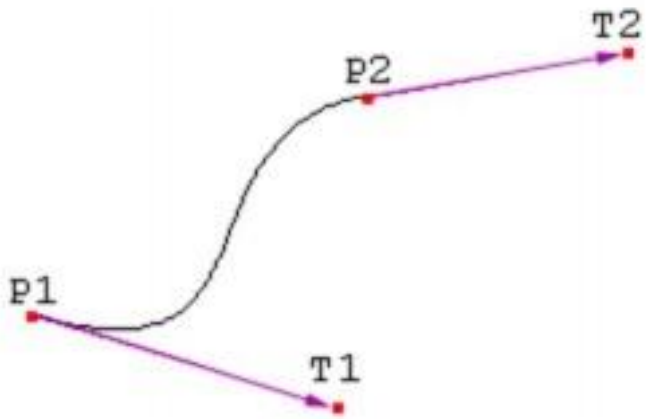


Fig.2.2. Hermite curve

Bezier curve is discovered by the French engineer **Pierre Bézier**. These curves can be generated under the control of other points. Approximate tangents by using control points are used to generate curve. The Bezier curve can be represented mathematically as –

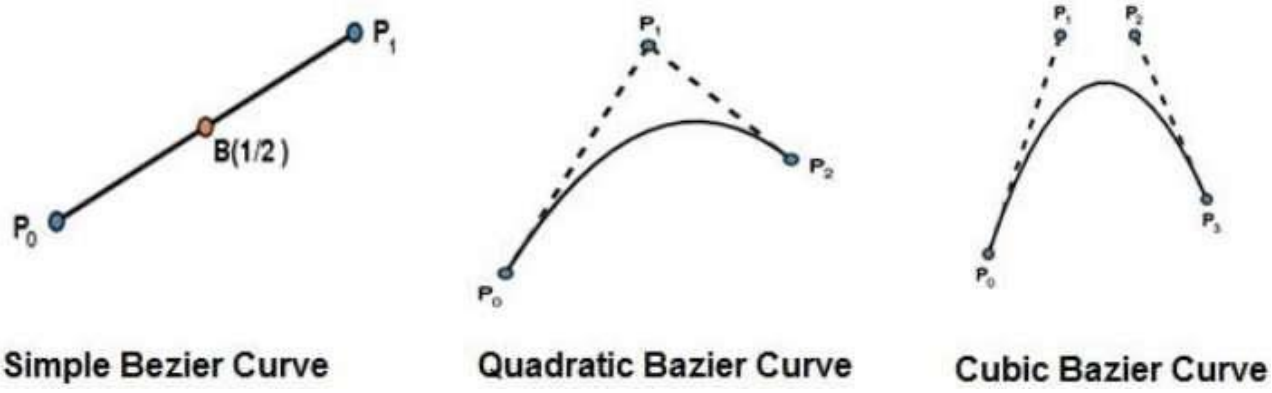
$$\sum_{k=0}^{n} P_i B_i^n(t)$$

Where p_i is the set of points and $B^n_i(t)$ represents the Bernstein polynomials which are given by –

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$$

Where **n** is the polynomial degree, **i** is the index, and **t** is the variable.

The simplest Bézier curve is the straight line from the point P_0 to P_1. A quadratic Bezier curve is determined by three control points. A cubic Bezier curve is determined by four control points.



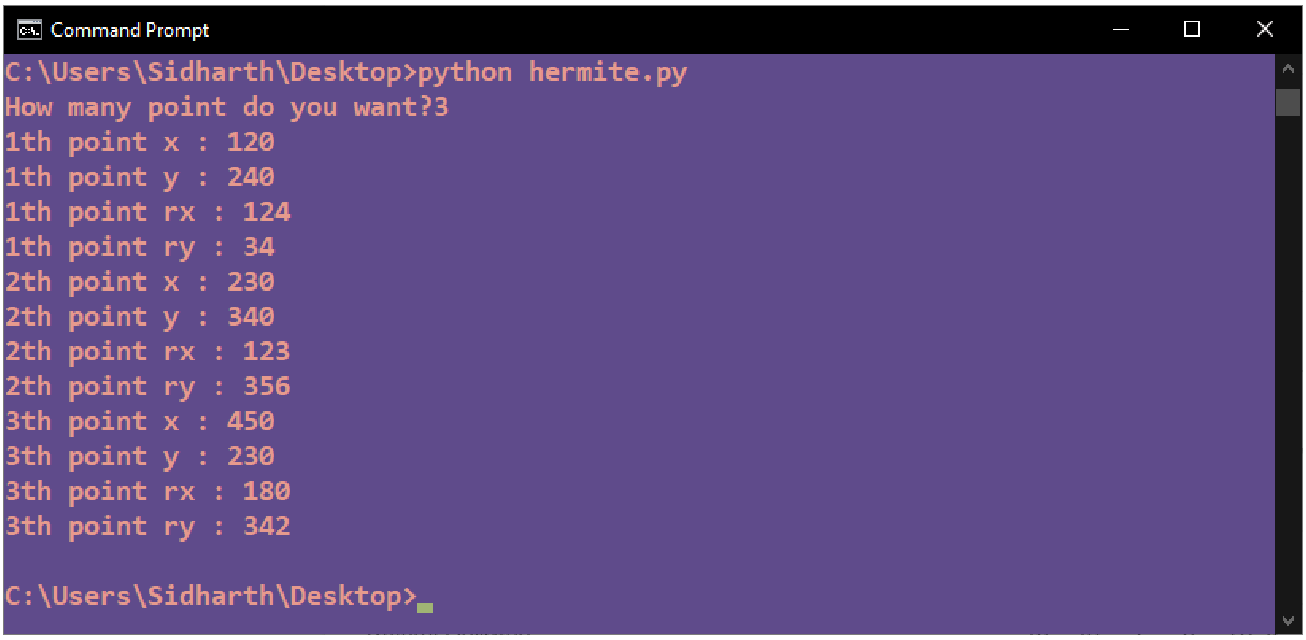
Simple Bezier Curve Quadratic Bazier Curve Cubic Bazier Curve

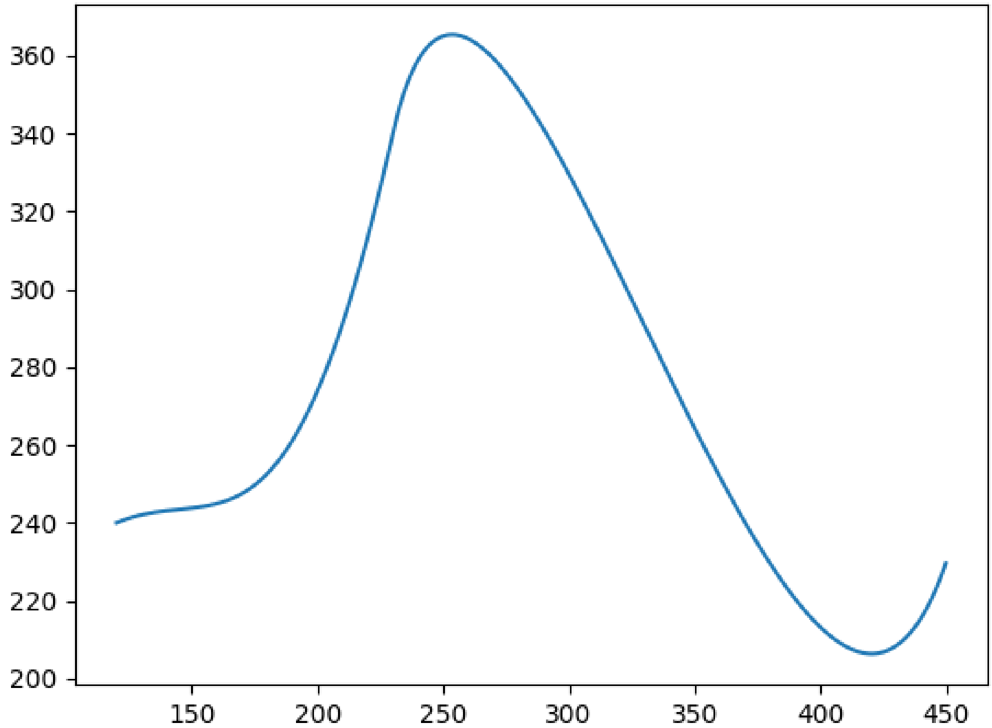
Code:

```
import math
import matplotlib.pyplot as plt

x = []
y = []
u = 0.001

def hermite(px , py , qx , qy , prx , pry , qrx , qry , u):
    tt = u
    while(tt < 1):
        resx = (1-3*math.pow(tt,2) + 2*math.pow(tt,3))*px +
(3*math.pow(tt,2) - 2*math.pow(tt,3))*qx + (tt-2*math.pow(tt,2) +
math.pow(tt,3))*prx + (-math.pow(tt,2)+math.pow(tt,3))*qrx
        resy = (1-3*math.pow(tt,2) + 2*math.pow(tt,3))*py +
(3*math.pow(tt,2) - 2*math.pow(tt,3))*qy+(tt-2*math.pow(tt,2) +
math.pow(tt,3))*pry + (-math.pow(tt,2)+math.pow(tt,3))*qry

        x.append(resx)
        y.append(resy)
        tt = tt+u

def main():
    n = int(input("How many point do you want?"))
    px , py , qx , qy , prx , pry , qrx , qry = [] , [] , [] , []
 , [] , [] , [] , []
    for i in range(n):
        px.append(int(input(str(i + 1) +"th point x : ")))
        py.append(int(input(str(i + 1) +"th point y : ")))
        prx.append(int(input(str(i + 1) +"th point rx : ")))
        pry.append(int(input(str(i + 1) +"th point ry : ")))
        if(i > 0):
            hermite(px[i - 1] , py[i - 1] , px[i] , py[i] , prx[i
 - 1] , pry[i - 1] , prx[i] , pry[i] , u)

    plt.plot(x , y)
    plt.show()

if __name__ == '__main__':
    main()
```

```
import math
import matplotlib.pyplot as plt


u = 0.001

def binomial(i, n):
    return math.factorial(n) / float(
        math.factorial(i) * math.factorial(n - i))


def bezier(px , py , u , x , y):
    n = len(px)
    for i in range(n):
        bio = binomial(i , n - 1)
        t = 0
        while t < 1:
            resX = math.pow(t , i) * math.pow( (1 - t) , n - i - 1) *
px[i]
            resY = math.pow(t , i) * math.pow( (1 - t) , n - i - 1) *
py[i]
            x.append(bio * resX)
            y.append(bio * resY)
            t = t + u
    for i in range(n - 1):
        for j in range(1000):
            x[j] = x[j] + x[(1000*(i+1)) + j]
            y[j] = y[j] + y[(1000*(i+1)) + j]

def main():
    x = []
    y = []
    n = int(input("How many point do you want?"))
    px , py = [] , []
    for i in range(n):
        px.append(int(input(str(i + 1) +"th point x : ")))
        py.append(int(input(str(i + 1) +"th point y : ")))

    bezier(px , py , u , x , y)
    x = x[:1000]
    y = y[:1000]

    plt.plot(x , y)
    plt.show()


if __name__ == '__main__':
    main()
```

Output:


Command Prompt
C:\Users\Sidharth\Desktop>python hermite.py
How many point do you want?3
1th point x : 120
1th point y : 240
1th point rx : 124
1th point ry : 34
2th point x : 230
2th point y : 340
2th point rx : 123
2th point ry : 356
3th point x : 450
3th point y : 230
3th point rx : 180
3th point ry : 342
C:\Users\Sidharth\Desktop>


360
340
320
300
280
260
240
220
200
150
200
250
300
350
400
450

```
Command Prompt
C:\Users\Sidharth\Desktop>python  bezier.py
How many point do you want?3
1th point x : 120
1th point y : 240
2th point x : 250
2th point y : 360
3th point x : 400
3th point y : 100

C:\Users\Sidharth\Desktop>
```