Beziers curves are to enable you to draw on the screen or with the laser printer a smooth curve of your chosen shape.

Bézier curves were introduced at different times but independently by the engineers P. Bézier of Rénault and P. de Casteljau of Citroën. The work of de Casteljau preceded the work of Bézier but was never published. So Bézier got the name attached to the curves.

To introduce them we need some notation. Given a sequence of control glueing had the idea of using this fact to obtain the parametric equations of curves and surfaces of desired shape we shall set some notation. Given a sequence of control glueing had the idea of using this fact to obtain the parametric equations of curves and surfaces of desired shape

\[ B_n[t, c_0, c_1, \ldots, c_n] = \sum_{k=0}^{n} c_k \binom{n}{k} t^k (1-t)^{n-k} \quad (1) \]

It develops that if \( f(t) \) is a continuous function in the interval \([0,1]\) and we set \( c_k = f(k/n) \) in (1) then the resulting polynomial

\[ B_n[t, f] = \sum_{k=0}^{n} f(k/n) \binom{n}{k} t^k (1-t)^{n-k} \quad (2) \]

has a graph that approximates that of \( f(x) \). In fact, it can be shown that, as \( n \to \infty \), the sequence \( B_n[x, f] \) will converge to \( f(x) \) uniformly in \([0,1]\). The french engineer and car designer Bézier working for Rénault had the idea of using this fact to obtain the parametric equations of curves and surfaces of desired shape by glueing together pieces of Bernstein polynomials. To translate this idea into a precise algorithm, we need some notation. Given a sequence of control points

\[ P_0 = (a_0, b_0) \ , \ P_1 = (a_1, b_1) \ , \ldots \ , \ P_n = (a_n, b_n) \quad (3) \]

we shall set

\[ x[t] = x[t, P_0, \ldots, P_n] = B_n[t, a_0, a_1, \ldots, a_n] \quad (4) \]

and

\[ y[t] = y[t, P_0, \ldots, P_n] = B_n[t, b_0, b_1, \ldots, b_n] \quad (5) \]

For instance for \( n = 3 \) the binomial coefficients are

\[ \binom{3}{0} = 1 \ , \ \binom{3}{1} = 3 \ , \ \binom{3}{2} = 3 \ , \ \binom{3}{3} = 1 \]

thus if

\[ P_0 = (1, 2) \ , \ P_1 = (2, 6) \ , \ P_2 = (-1, 3) \ , \ P_3 = (5, 1) \]

then

\[ x[t] = B_3[1, 2, -1, 5] = 1 \times 1 \times t^0 (1-t)^3 + 2 \times 3 \times t^1 (1-t)^2 - 1 \times 3 \times t^2 (1-t) + 5 \times 1 \times t^3 (1-t)^0 \]

that is

\[ x[t] = (1-t) + 6t(1-t)^2 - 3t^2(1-t) + 5t^3 = 1 + 3t - 12t^2 + 13t^3 \]

Similarly we get that

\[ y[t] = B_3[2, 6, 3, 1] = 2 \times 1 \times t^0 (1-t)^3 + 6 \times 3 \times t^1 (1-t)^2 + 3 \times 3 \times t^2 (1-t) + 1 \times 1 \times t^3 (1-t)^0 \]

that is

\[ y[t] = 2(1-t)^3 + 18t(1-t)^2 + 9t^2(1-t) + t^3 = 2 + 12t - 21t^2 + 8t^3 \]
It can be seen, by computer experimentation, that even for small values of $n$, the curve with parametric equations $P[t] = (x[t], y[t])$ for $0 \leq t \leq 1$ tries to mimic the shape of the polygon whose vertices are the points $P_i$ given in (3). Bézier’s idea consists of constructing curves by piecing together curves $(x[t], y[t])$ (as given in in (4), (5)) each depending on a small number of control points. For this reason the curve defined by the equations (4), (5) is now referred to as the Bézier curve corresponding to the control points $P_0, P_1, \ldots, P_n$. We give below two Bezier curves together with their control polygons in the case $n = 7$.

It can be seen that although the curves are not that close to the corresponding polygon, they all start tangent to the segment $P_0P_1$ and end tangent to the segment $P_{n-1}P_n$. This important property permits us to glue them together to form smooth curves of arbitrarily general shapes. In fact, if you experiment a bit on the computer, with $n = 4$, you will quickly learn how to guess the polygon needed to get any desired small piece of your final curve. Basically you will see that you will have to select a polygon which in a sense exaggerates the target shape.

One of my former PhD students David Little constructed under my direction an Applet (available in our website) that you may use to construct Beziér curves of your choice. This Applet is very convenient since it gives also the coordinates of the control points.

There is a recursive geometric algorithm that constructs a Bézier curve from its control polygon, that is due to De Casteljau, which beautifully illustrates the geometric dependence of the curve from its control polygon. We shall illustrate the algorithm in the case of 5 control points $P_0, P_1, P_2, P_3, P_4$.

This given, for a parameter $t$ we set

$$Q_i[t] = (1 - t)P_i + t P_{i+1} \quad (\text{for } i = 0, 1, 2, 3),$$

then

$$R_i[t] = (1 - t)Q_i[t] + t Q_{i+1}[t] \quad (\text{for } i = 0, 1, 2),$$

then

$$S_i[t] = (1 - t)R_i[t] + t R_{i+1}[t] \quad (\text{for } i = 0, 1),$$

and finally

$$T_0[t] = (1 - t)S_0[t] + t S_1[t].$$

If you work it out by hand and or by means of MATHEMATICA, and carry out all these substitutions you will find that the last point is given by the expression

$$T_0[t] = P_0(1 - t)^4 + 4P_1(1 - t)^3 t + 6P_2(1 - t)^2 t^2 + 4P_3(1 - t) t^3 + P_4 t^4 = \sum_{i=0}^{4} \binom{4}{i} (1 - t)^i t^{n-i} P_i.$$
Note that given two points $A$ and $B$ the point 

$$P[t] = (1-t)A + tB$$

(for $0 \leq t \leq 1$)

is the center of gravity of a particle of mass $1-t$ placed at $A$ with a particle of mass $t$ placed at $B$. Thus as $t$ varies from 0 to 1, the point $P[t]$ will describe the segment $AB$, starting at $A$ and ending at $B$. This given it should not be difficult to see that by setting $t = .6$ the collection of polygons 

$$P_0P_1P_2P_3P_4, \quad Q_0Q_1Q_2Q_3, \quad R_0R_1R_2, \quad S_0S_1,$$

and the final point $T_o[t]$ will be as depicted in the figure below

![Bezier Curve Diagram]

So you see that as $t$ varies the red point (which is $T_o[t]$) will describe a curve that starts tangentially to the segment $P_0P_1$ and ends tangentially to the segment $P_3P_4$. We can also see from this picture how the behaviour of the curve described by $T[t]$ will try to follow the control polygon.

Note that the **Mathematica** command that plots a curve defined by the parametric equations

$$x = f(t), \quad y = g(t)$$

for $t$ varying in the interval $[a,b]$ is simply

**ParametricPlot**[ {f(t), g(t)}, {t, a, b}, AspectRatio -> Automatic ]

For instance for

$$f(t) = \sin t, \quad g(t) = \sin 2t, \quad \text{and}, \quad a = 0, \quad b = 2\pi$$

you simply type

**ParametricPlot**[ {sin[t], sin[2 t]}, {t, 0, 2Pi}, AspectRatio -> Automatic ]

You can also get the simultaneous display of several curves by letting the argument of **ParametricPlot** be a sequence of pairs of functions $\{f_i(t), g_i(t)\}$ followed by a range of $t$ values in the form $\{t, a, b\}$. To be precise, your command should be

**ParametricPlot**[
  Evaluate[\n    \{f_1(t), g_1(t)\}, \{f_2(t), g_2(t)\}, \text{ etc }\}\n  , \{t, a, b\}, \text{ AspectRatio } \rightarrow \text{ Automatic}\n]

Of course you must arrange so that the parametric equations of all the desired curves have a common parameter $t$ varying in a common range $[a,b]$. 