

An Introduction to Bézier Curves

Thomas W. Sederberg

January 6, 2003

1 Introduction

Bézier curves are named after their inventor, Dr. Pierre Bézier. Bézier was an engineer with the Renault car company and set out in the early 1960's to develop a curve formulation which would lend itself to shape design.

It is helpful to think of Bézier curves in terms of the center of mass of a set of point masses. For example, consider the four masses $m_0, m_1, m_2,$ and m_3 located at points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$. The center of mass of these

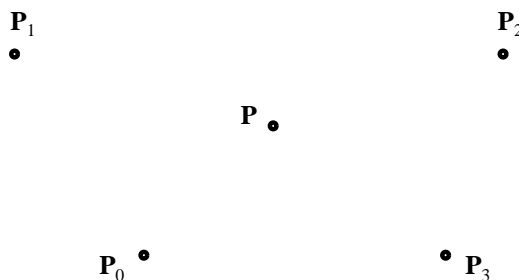


Figure 1: Center of mass of four points

four point masses is given by the equation

$$\mathbf{P} = \frac{m_0\mathbf{P}_0 + m_1\mathbf{P}_1 + m_2\mathbf{P}_2 + m_3\mathbf{P}_3}{m_0 + m_1 + m_2 + m_3}.$$

Next, imagine that instead of being fixed, constant values, each mass varies as a function of some parameter t . In specific, let $m_0 = (1 - t)^3$, $m_1 = 3t(1 - t)^2$, $m_2 = 3t^2(1 - t)$ and $m_3 = t^3$. The values of these masses as a function of t is shown in this graph: Now, for each value of t , the masses assume different weights and their center of mass changes continuously. In fact, as t varies between 0 and 1, a curve is swept out by the center of masses. This curve is a cubic Bézier curve – *cubic* because the mass equations are cubic polynomials in t . Notice that, for any value of t , $m_0 + m_1 + m_2 + m_3 \equiv 1$, and so we can simply write the equation of this Bézier curve as $\mathbf{P} = m_0\mathbf{P}_0 + m_1\mathbf{P}_1 + m_2\mathbf{P}_2 + m_3\mathbf{P}_3$.

Note that when $t = 0$, $m_0 = 1$ and $m_1 = m_2 = m_3 = 0$. This forces the curve to pass through \mathbf{P}_0 . Also, when $t = 1$, $m_3 = 1$ and $m_0 = m_1 = m_2 = 0$, thus the curve also passes through point \mathbf{P}_3 . Furthermore, the curve is tangent to $\mathbf{P}_0 - \mathbf{P}_1$ and $\mathbf{P}_3 - \mathbf{P}_2$. These properties make Bézier curves an intuitively meaningful means for describing free-form shapes. Here are some other examples of cubic Bézier curves which illustrate these properties. These variable masses m_i are normally called *blending functions* and their locations \mathbf{P}_i are known as *control points* or Bézier points. If we draw straight lines between adjacent control points, as in a dot to dot puzzle, the resulting figure is known as a *control polygon*. The blending functions, in the case of Bézier curves, are known as *Bernstein polynomials*. We will later look at other curves forms with different blending functions.

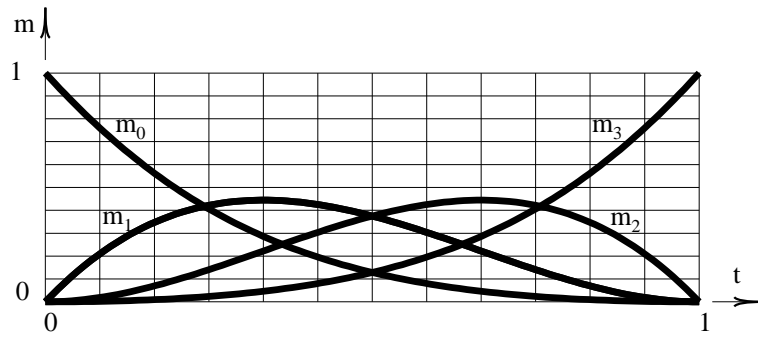


Figure 2: Cubic Bézier blending functions

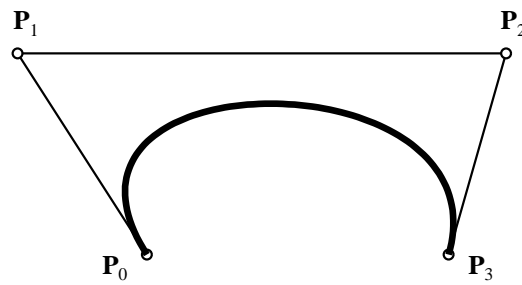


Figure 3: Cubic Bézier curve

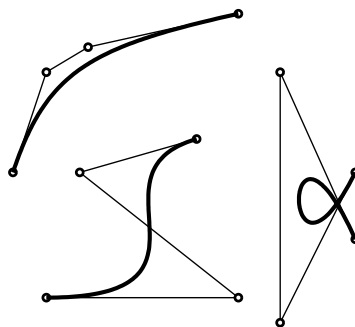


Figure 4: Examples of cubic Bézier curves

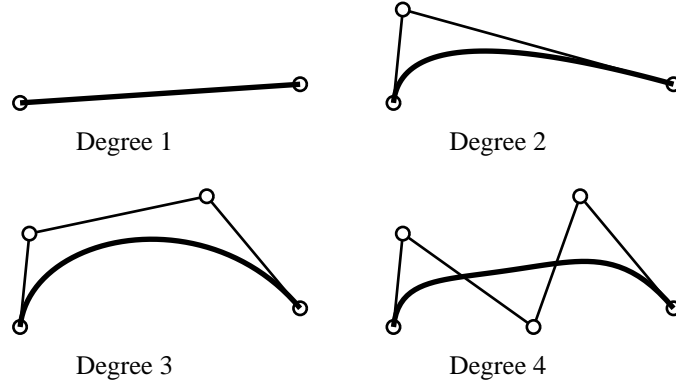


Figure 5: Bézier curves of various degree

Bézier curves of any degree can be defined. Figure 5 shows sample curves of degree one through four. A degree n Bézier curve has $n + 1$ control points whose blending functions are denoted $B_i^n(t)$, where

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i, \quad i = 0, 1, \dots, n.$$

Recall that $\binom{n}{i}$ is called a *binomial coefficient*, sometimes spoken “ n - choose - i ”, and is equal to $\frac{n!}{i!(n-i)!}$. In our introductory example, $n = 3$ and $m_0 = B_0^3 = (1-t)^3$, $m_1 = B_1^3 = 3t(1-t)^2$, $m_2 = B_2^3 = 3t^2(1-t)$ and $m_3 = B_3^3 = t^3$. $B_i^n(t)$ is also referred to as the i th Bernstein polynomial of degree n . The equation of a Bézier curve is thus:

$$\mathbf{P}(t) = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i. \quad (1)$$

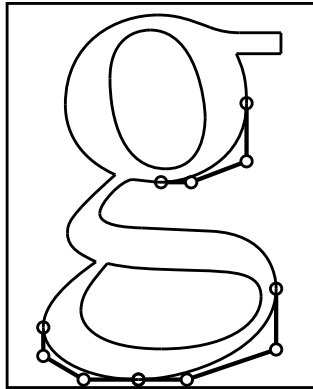


Figure 6: Font definition using Bézier curves

A common use for Bézier curves is in font definition. Figure 6 shows the outline a letter “g” created using Bézier curves. All PostScript font outlines are defined use cubic and linear Bézier curves.

2 Bézier Curves over Arbitrary Parameter Intervals

Equation 1 gives the equation of a Bézier curve which starts at $t = 0$ and ends at $t = 1$. It is useful, especially when fitting together a string of Bézier curves, to allow an arbitrary parameter interval:

$$t_0 \leq t \leq t_1$$

such that $\mathbf{P}(t_0) = \mathbf{P}_0$ and $\mathbf{P}(t_1) = \mathbf{P}_n$. This can be accomplished by modifying equation 1:

$$\mathbf{P}(t) = \frac{\sum_{i=0}^n \binom{n}{i} (t_1 - t)^{n-i} (t - t_0)^i \mathbf{P}_i}{(t_1 - t_0)^n} = \sum_{i=0}^n \binom{n}{i} \left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-i} \left(\frac{t - t_0}{t_1 - t_0}\right)^i \mathbf{P}_i. \quad (2)$$

3 Subdivision of Bézier Curves

The most fundamental algorithm for dealing with Bézier curves is the subdivision algorithm. This was devised in 1959 by Paul de Casteljau (who was working for the Citroen automobile company) and is referred to as the *de Casteljau algorithm*. It is sometimes known as the *geometric construction algorithm*.

Consider a Bézier curve defined over the parameter interval $[0, 1]$. It is possible to subdivide such a curve into two new Bézier curves, one of them over the domain $0 \leq t \leq \tau$ and the other over $\tau \leq t \leq 1$. These two new Bézier curves, considered together, are equivalent to the single original curve from which they were derived.

As illustrated in Figure 7, begin by adding a superscript of 0 to the original control points, then compute

$$\mathbf{P}_i^j = (1 - \tau)\mathbf{P}_i^{j-1} + \tau\mathbf{P}_{i+1}^{j-1}; \quad j = 1, \dots, n; \quad i = 0, \dots, n - j. \quad (3)$$

Then, the curve over the parameter domain $0 \leq t \leq \tau$ is defined using control points $\mathbf{P}_0^0, \mathbf{P}_0^1, \mathbf{P}_0^2, \dots, \mathbf{P}_0^n$ and the curve over the parameter domain $\tau \leq t \leq 1$ is defined using control points $\mathbf{P}_0^n, \mathbf{P}_1^{n-1}, \mathbf{P}_2^{n-2}, \dots, \mathbf{P}_n^0$.

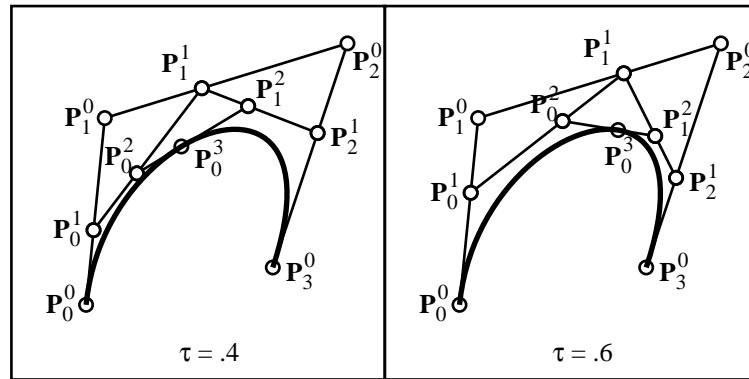


Figure 7: Subdividing a cubic Bézier curve.

Figure 8 shows that when a Bézier curve is repeatedly subdivided, the collection of control polygons converge to the curve.

One immediate value of the de Casteljau algorithm is that it provides a numerically stable means of computing the coordinates of any point along the curve. Its extension to curves of any degree should be obvious.

It can be shown that if a curve is repeatedly subdivided, the resulting collection of control points converges to the curve. Thus, one way of plotting a Bézier curve is to simply subdivide it an appropriate number of times and plot the control polygons.

4 Rasterizing a Bézier Curve

If the control points of the degree n Bézier curve are $\mathbf{P}_i = (x_i, y_i)$ in *pixel coordinates*, then let

$$d = \max\{x_i - x_{i-1}, y_i - y_{i-1}\} \quad i = 1, \dots, n$$

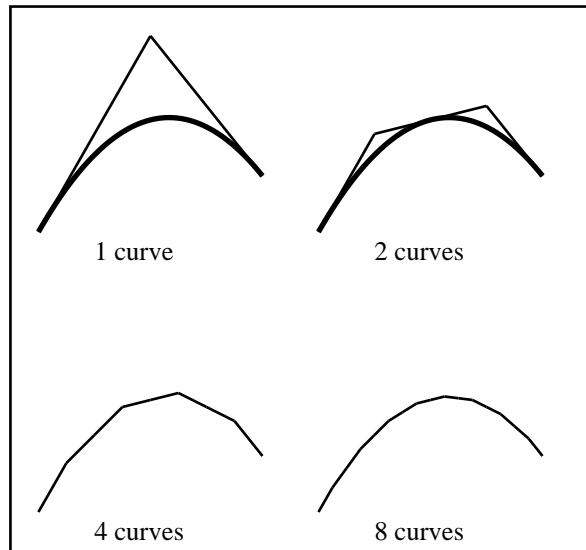


Figure 8: Recursively subdividing a quadratic Bézier curve.

If you now evaluate the curve at $n * d + 1$ evenly spaced values of t and paint each resulting pixel, there will be no gaps in the drawing of the curve. Another way to say this, compute

$$\delta = \frac{1}{n * d}.$$

Then, compute the points $\mathbf{P}(i * \delta)$, $i = 0 \dots, n * d$.

5 Forward Differencing

Horner's algorithm is the fastest method for evaluating a polynomial at a single point. For a degree n polynomial, it requires n multiplies and n adds.

If a polynomial is to be evaluated at several evenly spaced values $t, t + \delta, t + 2\delta, \dots, t + k\delta$, the fastest method is to use *forward differences*.

Consider a degree 1 polynomial

$$f(t) = a_0 + a_1 t.$$

The difference between two adjacent function values is

$$\Delta_1(t) = f(t + \delta) - f(t) = [a_0 + a_1(t + \delta)] - [a_0 + a_1 t] = a_1 \delta.$$

Thus, $f(t)$ can be evaluated at several evenly spaced points using the algorithm:

```

 $\Delta_1 = a_1 \delta$ 
 $t_0 = 0$ 
 $f(0) = a_0$ 
for  $i = 1$  to  $k$  do
 $t_i = t_{i-1} + \delta$ 
 $f(t_i) = f(t_{i-1}) + \Delta_1$ 
endfor

```

Thus, each successive evaluation requires only one add, as opposed to one add and one multiply for Horner's algorithm.

This idea extends to polynomials of any degree. For the quadratic case,

$$f(t) = a_0 + a_1t + a_2t^2.$$

The difference between two adjacent function values is

$$\Delta_1(t) = f(t + \delta) - f(t) = [a_0 + a_1(t + \delta) + a_2(t + \delta)^2] - [a_0 + a_1t + a_2t^2]$$

$$\Delta_1(t) = a_1\delta + a_2\delta^2 + 2a_2t\delta.$$

We can now write

```

t0 = 0
f(0) = a0
for i = 1 to k do
  ti = ti-1 + δ
  Δ1(ti) = a1δ + a2δ2 + 2a2ti-1δ
  f(ti) = f(ti-1) + Δ1(ti-1)
endfor

```

In this case, $\Delta(t)$ is a linear polynomial, so we can evaluate it as above, by defining

$$\Delta_2(t) = \Delta_1(t + \delta) - \Delta_1(t) = 2a_2\delta^2$$

and our algorithm now becomes

```

t0 = 0
f(0) = a0
Δ1 = a1δ + a2δ2
Δ2 = 2a2δ2
for i = 1 to k do
  ti = ti-1 + δ
  f(ti) = f(ti-1) + Δ1
  Δ1 = Δ1 + Δ2
endfor

```

It should be clear that for a degree n polynomial, each successive evaluation requires n adds and no multiplies! For a cubic polynomial

$$f(t) = a_0 + a_1t + a_2t^2 + a_3t^3,$$

the forward difference algorithm becomes

```

t0 = 0
f(0) = a0
Δ1 = a1δ + a2δ2 + a3δ3
Δ2 = 2a2δ2 + 6a3δ3
Δ3 = 6a3δ3

```

```

for  $i = 1$  to  $k$  do
 $t_i = t_{i-1} + \delta$ 
 $f(t_i) = f(t_{i-1}) + \Delta_1$ 
 $\Delta_1 = \Delta_1 + \Delta_2$ 
 $\Delta_2 = \Delta_2 + \Delta_3$ 
endfor

```

Several questions remain. First, what are the initial values for the Δ_i if we want to start at some value other than $t = 0$. Second, what is a general equation for the Δ_i for a general degree n polynomial $f(t)$. Also, what if our polynomial is not in power basis.

These questions can be answered almost trivially by observing the following. Since $t_{i+1} = t_i + \delta$, we have

$$\begin{aligned} \Delta_1(t_i) &= f(t_{i+1}) - f(t_i); \\ \Delta_j(t_i) &= \Delta_{j-1}(t_{i+1}) - \Delta_{j-1}(t_i), \quad j = 2, \dots, n; \\ \Delta_n(t_i) &= \Delta_n(t_{i+1}) = \Delta_n(t_{i+k}) = \text{a constant} \\ \Delta_{n+1} &= 0 \end{aligned}$$

Thus, our initial values for $\Delta_j(t_i)$ can be found by simply computing $f(t_i), f(t_{i+1}), \dots, f(t_{i+n})$ and from them computing the initial differences. This lends itself nicely to a table. Here is the table for a degree four case:

$f(t_i)$	$f(t_{i+1})$	$f(t_{i+2})$	$f(t_{i+3})$	$f(t_{i+4})$
$\Delta_1(t_i)$	$\Delta_1(t_{i+1})$	$\Delta_1(t_{i+2})$	$\Delta_1(t_{i+3})$	
$\Delta_2(t_i)$	$\Delta_2(t_{i+1})$	$\Delta_2(t_{i+2})$		
$\Delta_3(t_i)$	$\Delta_3(t_{i+1})$			
$\Delta_4(t_i)$				
0	0	0	0	0

To compute $f(t_{i+5})$, we simply note that every number R in this table, along with its right hand neighbor R_{right} and the number directly beneath it R_{down} obey the rule

$$R_{right} = R + R_{down}.$$

Thus, we can simply fill in the values

$$\begin{aligned} \Delta_4(t_{i+1}) &= \Delta_4(t_i) + 0 \\ \Delta_3(t_{i+2}) &= \Delta_3(t_{i+1}) + \Delta_4(t_{i+1}) \\ \Delta_2(t_{i+3}) &= \Delta_2(t_{i+2}) + \Delta_3(t_{i+2}) \\ \Delta_1(t_{i+4}) &= \Delta_1(t_{i+3}) + \Delta_2(t_{i+3}) \\ f(t_{i+5}) &= f(t_{i+4}) + \Delta_1(t_{i+4}) \end{aligned}$$

Note that this technique is independent of the basis in which $f(t)$ is defined. Thus, even if it is defined in Bernstein basis, all we need to do is to evaluate it $n + 1$ times to initiate the forward differencing.

For example, consider the degree 4 polynomial for which $f(t_i) = 1, f(t_{i+1}) = 3, f(t_{i+2}) = 2, f(t_{i+3}) = 5, f(t_{i+4}) = 4$. We can compute $f(t_{i+5}) = -24, f(t_{i+6}) = -117$, and $f(t_{i+7}) = -328$ from the following difference table:

$t :$	t_i	t_{i+1}	t_{i+2}	t_{i+3}	t_{i+4}	t_{i+5}	t_{i+6}	t_{i+7}
$f(t) :$	1	3	2	5	4	-24	-117	-328
$\Delta_1(t) :$	2	-1	3	-1	-28	-93	-211	
$\Delta_2(t) :$	-3	4	-4	-27	-65	-118		
$\Delta_3(t) :$	7	-8	-23	-38	-53			
$\Delta_4(t) :$	-15	-15	-15	-15	-15			
$\Delta_5(t) :$	0	0	0	0	0	0	0	0