

Bezier Curves

Curves are trajectories of moving points. We'll specify them as functions assigning a location of that moving point (in 2D or 3D) to a parameter t (think of it as time). In general, all curves we are going to talk about will be defined by a sequence of control points. Curves useful in geometric modeling should have shape which has a clear and intuitive relation to the path of the sequence of control points. One family of curves satisfying this requirement are Bezier Curves.

1 Definition and a few formulas and properties

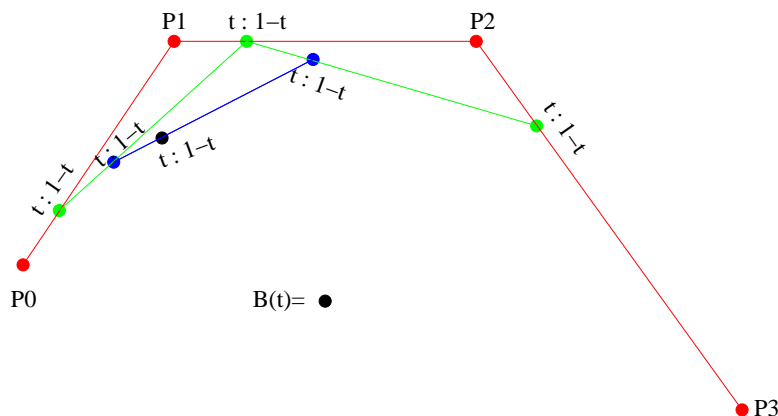


Figure 1: De Casteljau algorithm. The control points P_0 , P_1 , P_2 and P_3 are joined with line segments (shown red; those red intervals form something called 'control polygon', even though they are not really a polygon but rather a polygonal curve). Each of them is then divided in the same ratio $t : 1 - t$, giving rise to the green points. Again, each consecutive two are joined with line segments which are subdivided with blue points and so on, until only one point is left (here: the black one). This is the location of our moving point at time t . The trajectory of that point for times between 0 and 1 is the Bezier curve. Of course, the more control points the more colors I would need to use, i.e. the more the 'levels' of subdivision are needed.

A Bezier curve is defined by a sequence of $N + 1$ control points, P_0 , P_1 , \dots , P_N . We defined the Bezier curve using the (invented by de Casteljau) algorithm, based on recursive splitting of the intervals joining the consecutive control points (Figure 1). It is clear from the picture that $B_{P_0 P_1 \dots P_N}(0) = P_0$ and $B_{P_0 P_1 \dots P_N}(1) = P_N$, i.e. the curve starts at the first and ends at the last control point (think what it means to split in 1:0 or 0:1 ratio...). $P_0 \vec{P}_1$ and $P_{N-1} \vec{P}_N$ are tangent vectors to the curve at its endpoints (Figure 2; also easily visible in the unrelated Figure 3).

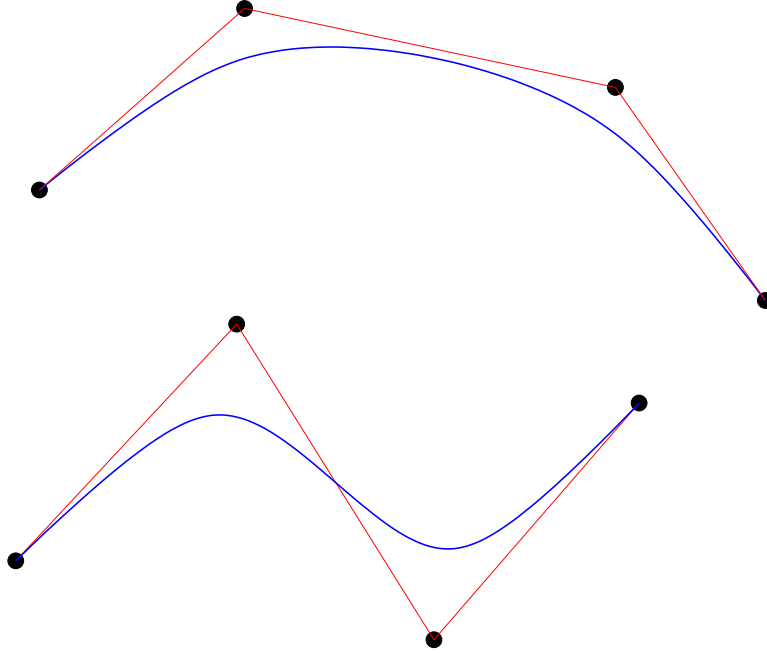


Figure 2: Bezier curves; black dots are the control points and the curve is shown in blue. Both curves have 4 control points and therefore are given by cubic polynomials. Notice that the interval joining the first two control points is tangent to the curve at its starting point and the line joining the last two control points is tangent to the curve at its final point.

Notice that the endpoints of the interval which is split as the last one (the blue one in Figure 1) are points of Bezier curves defined by sequences of control points $P_0P_1 \dots P_{N-1}$ and $P_1P_2 \dots P_N$. This leads to the formula

$$B_{P_0P_1 \dots P_N}(t) = (1-t) * B_{P_0P_1 \dots P_{N-1}}(t) + t * B_{P_1P_2 \dots P_N}(t).$$

Remembering that for just one control point we have constant Bezier curve i.e. $B_{P_0}(t) = P_0$, we can use the above equation to derive an explicit formula for a B-spline curve of any degree. Here's how:

$$B_{P_0P_1} = (1-t) * B_{P_0} + t * B_{P_1} = (1-t) * P_0 + t * P_1$$

having this:

$$\begin{aligned} B_{P_0P_1P_2} &= (1-t) * B_{P_0P_1} + t * B_{P_1P_2} = \\ &= (1-t) * ((1-t) * P_0 + t * P_1) + t * ((1-t) * P_1 + t * P_2) = \\ &= (1-t)^2 * P_0 + 2t(1-t) * P_1 + t^2 * P_2. \end{aligned}$$

It's not hard to guess that, in general, to evaluate $B_{P_0P_1 \dots P_N}$ we need to weight the consecutive control points with the terms of the expansion of $((1-t)+t)^{N-1}$.

That is, the weight sequences are (see above) 1 (for $N = 0$), $(1 - t), t$ ($N = 1$) and $(1 - t)^2, 2t(1 - t), t^2$ ($N = 2$). If we have 5 control points, then the weights are $(1 - t)^4, 4t(1 - t)^3, 6t^2(1 - t)^2, 4t^3(1 - t), t^4$. Thus, the formula for the B-spline curve (which is a degree-4 polynomial in this case) is

$$B_{P_0 P_1 P_2 P_3 P_4} = (1 - t)^4 * P_0 + 4t(1 - t)^3 * P_1 + 6t^2(1 - t)^2 * P_2 + 4t^3(1 - t) * P_3 + t^4 * P_4.$$

Notice that this means that each point on a Bezier curve is a weighted average of the input points. Therefore, it lies inside the *convex hull* of the control points, i.e. the smallest convex polygon enclosing them (see Figure 3).

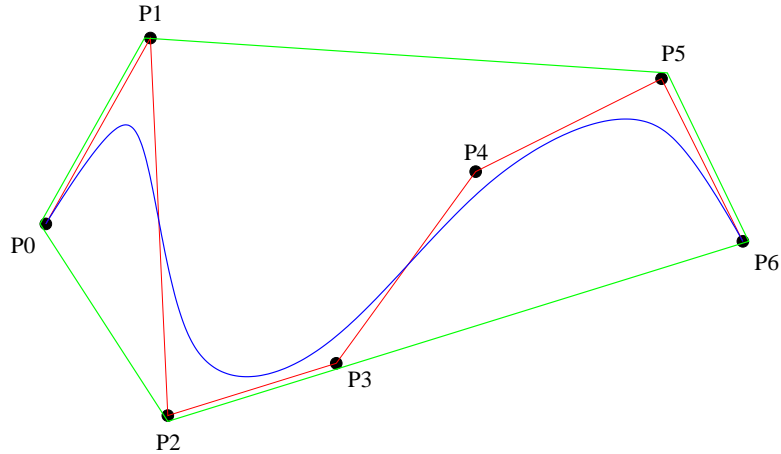


Figure 3: The control polygon is shown red. The convex hull of the control points is the green polygon (smallest convex having all the control points inside). The Bezier curve is shown in blue. The convex hull property states that blue has to be contained inside the green polygon.

2 Joining Bezier segments

Based on the previous section, it is not hard to figure out how to join two or more Bezier curves so that they all form a smooth curve. Figure 4 explains it all.

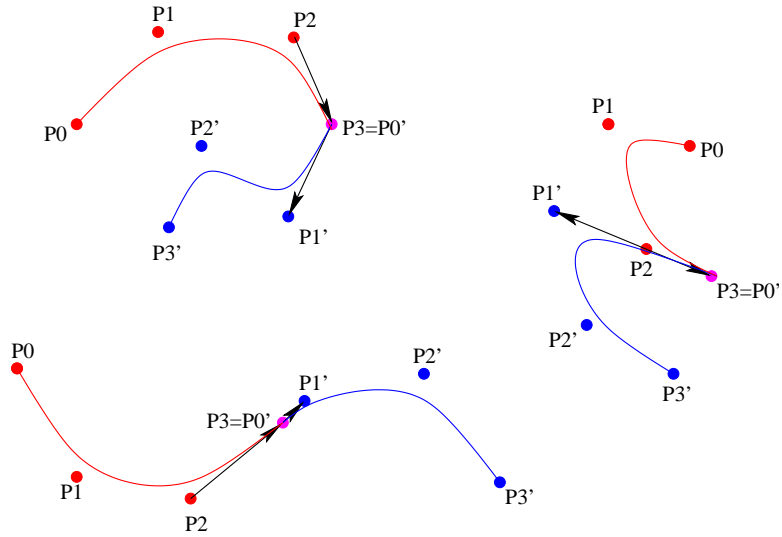


Figure 4: Joining two Bezier curves. Control points for one are red and for the other (primed) – blue. We want them to *join*, so the second one should start where the first ends. Thus, the last control point of the first curve has to overlap with the first of the second curve. This point is shown in magenta (=red+blue). Top left: the vectors $P_2\vec{P}_3$ and $P_0\vec{P}'_1$ are not colinear. Therefore, the red and blue curves have different tangent lines at the point they meet at (magenta dot) and so they don't join smoothly there. Right: the vectors are colinear now, but they point in different directions. So, the red curve arrives from one direction, at the magenta point, but the blue one leaves in the opposite direction. This causes a very ('infinitely?') sharp cusp in the curve. Bottom left: The vectors are parallel and they point in the same direction; the two curves join smoothly.