

Drawing Bézier curves

The theory behind creating smooth curves between points

In this issue...

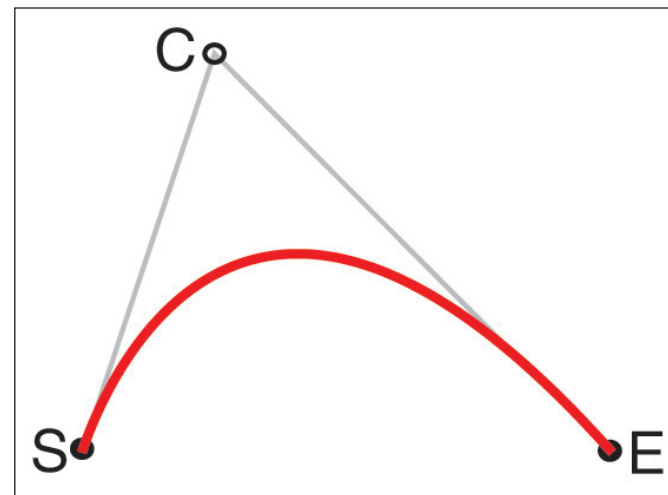
▶ WHAT'S COVERED

One of the standard tasks for any computer graphics application is to give the user the ability to draw a smooth curve between two points, rather than just a straight line. The easiest curves for a computer to draw for this situation are known as Bézier curves or splines. In this article we look at what these curves are and how they're used even in the font that you are reading now.

Back in school, it seemed that we always had to draw some chart or other for Maths homework. We were given an equation and had to plot it on squared paper. Fortunately, I had a marvellous bendy ruler for this task and I would carefully bend this rubber and wire contraption along the points of my graph so that I could draw a smooth line joining them.

These days, kids in school have graphics calculators and computers to do this work, but how does the program draw a smooth curve given just a few discrete points?

Back in 1959, Paul de Casteljau was an engineer and mathematician at Citroën, the French car manufacturer, in charge of their CAD systems (these were very basic affairs in those days, unlike the CAD/CAM graphics applications available to us today). He developed an algorithm for computing a smooth curve, later known as a Bézier curve, based upon a cubic equation. The algorithm, now known as de Casteljau's algorithm, is a recursive algorithm for determining the curve; it's



▲ Figure 1: A quadratic Bézier curve showing the control point.

as if you made little steps along the curve being pulled in a couple of directions at once by what are known as control points.

This very recursiveness makes the algorithm more suited to computer than to human calculation. It turns out that, despite being recursive, it's very stable numerically and produces good curves over a wider variety of start, end and control points than any other polynomial algorithm, such as Horner's algorithm.

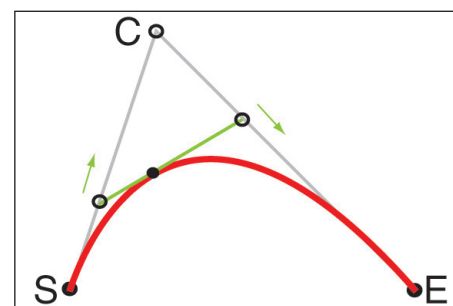
Despite de Casteljau being first off the block with this new algorithm, it fell to another engineer and mathematician, Pierre Bézier, to use them successfully in the design of car bodies. Funnily enough, Bézier worked for Renault,

a major competitor of Citroën, also in the area of CAD systems. It was Bézier's success that led to the curves becoming more widely known and subsequently gathering his name as time passed.

Fast forward to the late '70s and early '80s. Xerox had designed the first laser printer and the researchers at Xerox PARC recognized the need for a page rendering system that could cope with text and images and vector graphics. Adobe Systems was formed by John Warnock and Chuck Geschke and their first product, PostScript, was such a product. The interesting thing about PostScript with regard to our discussion (there were many interesting things about the product) was that text was 'drawn' using vector graphics

rather than rendered as bitmaps. The fonts used by a PostScript printer were defined as a series of Bézier curves, together with 'hints' that defined what happened when the font size changed.

Apple Computers was the first



▲ Figure 2: A quadratic Bézier curve being drawn.

Spotlight on... PostScript

Although most people only think of PostScript as a format for defining pages of text for printing or as a font definition, it is, in fact, a complete programming language. It's a stack-based language like Forth, interpreted, with a garbage collection mechanism, and uses Reverse Polish Notation (RPN) as its calculation engine. It's text-based as well, so it's possible for you to load a PostScript file (usually it has an extension of .ps) into Notepad and read it.

There are several extensions to the PostScript language available, all of which generally include some binary data alongside text. One of them is Extended PostScript (EPS), used for storing vector illustrations, like the figures in this article. Another is even more familiar: the Portable Document Format (PDF), which, although being based on PostScript, has numerous extensions for things like encryption, different character encodings and bitmaps. ■

Nugget

Although Bézier curves allow us to draw some smooth arbitrary curves, amazingly enough they cannot draw simple shapes such as the perfect circle. The closest that can be obtained for the circle (or ellipse) is by drawing it in four pieces, one for each quadrant. Each piece is a separate cubic Bézier curve. If you own a program that does vector drawing, it's a fun five minutes to try and create a circle quadrant. After that of course, you can dupe this shape four times and rotate them into position. ■

company to launch a cheap PostScript laser printer (the Apple LaserWriter) and in the process launched the desktop publishing innovation of the late '80s. Adobe went on to create hardware-independent font definitions using a subset of PostScript and named them Type 1 fonts, thereby sparking their own revolution with fonts.

All this innovation came out of the new generations of printers and high resolution graphics screens, and all was based on curves developed by de Casteljau and Bézier twenty years earlier.

The issue I faced when drawing graphs in mathematics lessons in school was that calculating and plotting lots of points on the graph was, despite the result being nice and smooth, computationally infeasible. With computers, the calculations and plotting are relatively easy and quick, however, it depends on having the right equation to plot first.

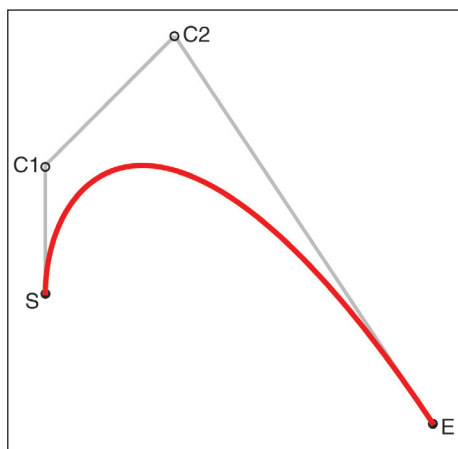
For some applications, we are given a small number of points, and the equation that connects them is unknown. Imagine if we had to draw the sans-serif letter J. We don't want to give a large set of points on this curve, since if the letter is shown very large, the lines we have to interpolate between the points become more and more significant. In other words, at small size we might be able to get away with

straight lines, at large size it will be really obvious. So we'd like to encode the shape as a straight up/down line from the top point to the point where the bowl of the J starts (that's just two points, with the extra information of a straight line), followed by a semicircular shape between the second point and a third point to the left. A J is nice in that the bowl of the letter can be assumed to be a semicircle, but imagine an S. What we'd like to do is to give as little information about points as we can, and then let the drawing algorithm work out the best curve itself. Enter a Bézier curve.

There are three 'low order' Bézier curves, with the third (the cubic) being the most used in drawing applications. Higher-order Bézier curves are generally not used since, for most applications, joining a series of cubic Bézier curves gives almost the same quality curve.

The first Bézier curve is linear. Two points are given: the start and end points, and the linear Bézier curve between them is simply a straight line. The equation of the line is easy to derive using simple mathematics, but with Bézier curves we define the equation in terms of a parameter 't' which varies from the start point to the end point. You can think of 't' as being the point of a plotter pen as it draws the line. 't' is usually thought of as varying between 0 and 1, so when it's 0.5, it's halfway along the line.

Next up is the quadratic Bézier curve. This type of curve is most used by Type 1 and TrueType fonts. The definition of this curve requires three points: again, the start and end points of the curve,



▲ Figure 3: A cubic Bézier curve showing the two control points 'C1' and 'C2'.

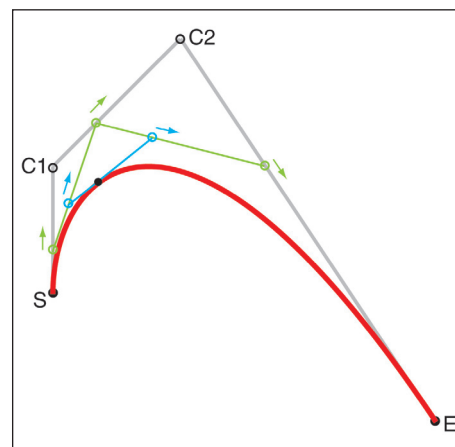
and now a third point, called the control point.

Figure 1 shows a quadratic Bézier curve. The start point is labelled 'S', the end point, 'E', and the control point, 'C'. Again we make use of a parameter 't'. Imagine that there are straight lines attaching 'S' to 'C', and 'C' to 'E', and there is a bead that travels along each

segment. Between the beads is a stretched elastic string. Now vary the bead from 'S' to 'C' linearly with respect to 't' and vary the other bead from 'C' to 'E' in the same fashion. Finally, put the plotter pen on the stretched elastic string and allow it to vary linearly along the length of the string as the other two beads are moving to their destinations. When 't' is 0.33, say, the first bead is one third of the way to 'C' from 'S', the second bead is one third of the way to 'E' from 'C', and the plotter pen is one third of the way along the stretched string. Figure 2 shows this situation.

Notice that the string between the beads will stretch and shrink as they travel, but we don't care: all we track is the ratio of where the pen is on the string (which is 't', of course). Those of you who are mathematicians, or remember just enough to play one on the telly, will recognize that the stretched string is always a tangent to the curve.

The next Bézier curve in the sequence is the cubic one. This is the favoured one in applications like Adobe



▲ Figure 4: A cubic Bézier curve being drawn.

Illustrator, since it enables you to easily join the end point of one curve to the start point of another, without destroying the smoothness of the curve.

The cubic Bézier curve requires four points in its definition: the start and end points of the curve, and, this time, two control points. Figure 3 shows a cubic Bézier curve with the two control points called 'C1' and 'C2'.

This time round it gets a little funky. If we join up the points with straight lines again ('S' to 'C1', 'C1' to 'C2', and 'C2' to 'E') we can place three beads on these lines and vary them with 't' again. This time we join the bead on 'S'-'C1' with that on 'C1'-'C2' with a stretched string, and the same for the beads on 'C1'-'C2' and 'C2'-'E'. Now we imagine two further threaded beads on these strings and between them is the tangent string and the plotter pen moves along this tangent string in the same ratio speed as everything else. Figure 4 shows this situation when 't' is 0.33 again.

You can view the control points as defining directions. The curve starts off aiming first at 'C1', and ends up aiming at 'E' from 'C2'. Hence if you wish to join two cubic curves smoothly you make sure that the first control point of the second curve is pointing in the same direction as 'C2' to 'E' of the first curve. This is how you create paths in programs like Illustrator: a path is just a set of cubic Bézier curves joined up. ■

Julian M Bucknall has worked for companies ranging from TurboPower to Microsoft and is now CTO for Developer Express. feedback@peplus.co.uk

Nugget

If you are at all handy with nails and a hammer, you can create Bézier curves quite easily with string. Take figure 1 as an example. Draw the two lines from 'S' to 'C' and from 'C' to 'E' on a board. Mark off the lines into 10 equidistant segments, say, and hammer a nail at each segment. Now wrap a string around the nails so that you connect the first nail on the 'S'-'C' line to the first nail on the 'C'-'E' line. Ditto for the second, third, etc, nails. You'll end up with the same Bézier curve. ■