Albert Technical Memo

#5: How to use Knot Vectors

Written by: John Peterson June 1990

This gives a basic explanation of how to use the knot vector of a Nurb curve (or surface) to manipulate its shape

Introduction

A Non-Uniform Rational B-Spline curve is defined by three things:

- Control points
- The curve's order.
- A knot vector

Control points are familiar to people who have used Bézier curves in applications such as Illustrator and MacDraw. The knot vector is an additional feature of the Nurb representation. It is easy to use the curves without ever accessing knot vectors, but learning the information in this note extends what you can do with them.

Curves like Nurbs and Béziers are parametric curves. The curve is a function $\mathbf{P}(u)$ that returns a point $\mathbf{P}$ on the curve for a particular value of the parameter u. As u varies from initial value u_{min} - towards u_{max} the curve is drawn. Knots are defined in the parameter space of a curve. We'll use the notation u_i to refer to a particular knot. The collection of knots for a particular curve is called the *knot vector.*

A curve is defined with a series of polynomials.[1] The number of polynomials needed to define a curve depends on the number of control points and the order of the curve. The knot vector determines where in the parameter range these polynomials start and stop as the curve is drawn. (They're called "knots" because the values "tie together" the polynomials.) Fortunately, you don't need to understand the details of polynomial math in order to use knot vectors.

[1]Remember plotting $x(u) = u,\ y(u) = u^2$ in algebra class?

This memo gives a brief overview of the rules for producing correct curves, several examples of how knot vectors are used, and some suggested reading for those interested in the theory.

The Rules

There are some basic rules governing how Nurbs are defined. First, the number of control points defining the curve must always be equal to or greater than the order of the curve. E.g., a quadratic (order = 3) must have at least three points, a cubic at least four, etc.

Second, the number of knots in the knot vector is always equal to the number of control points plus the order of the curve. E.g., a cubic (order=4) with four control points has eight items in the knot vector.

Third, the order of a curve must be at least two.

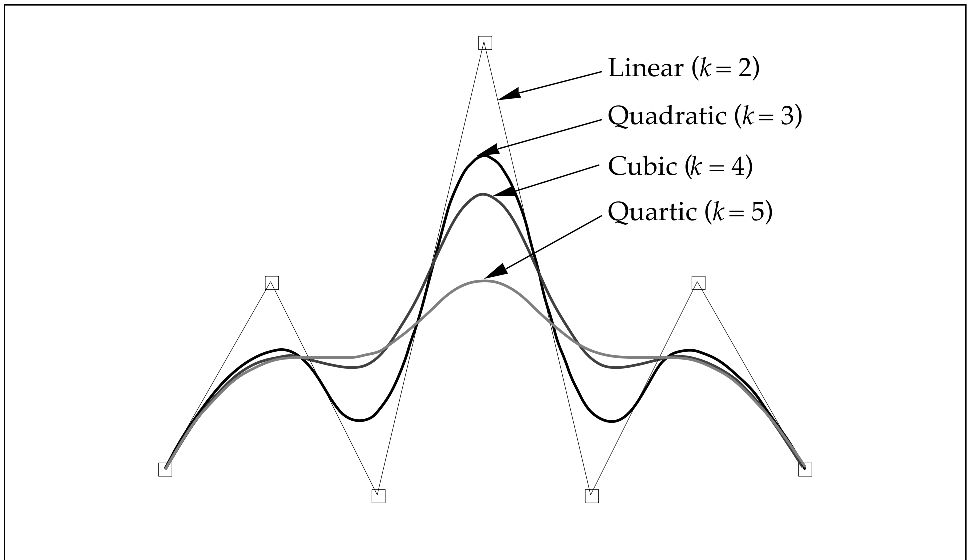
Fourth, the values in the knot vector must always be in ascending order. E.g., [0 0 0 1 2 3 3 3] is a valid knot vector, but [0 0 0 2 1 3 3 3] is not.

Finally, the valid parameter range for a curve starts at $u_{\text{min}}=u_{\text{order-1}}$ and goes up to (but does not include!) $u_{\text{max}} = u_m$, where m is the number of control points. Values less than the minimum parameter, or equal or above the maximum parameter aren't defined. (In other words, u must be $u_{\text{min}} \leq u < u_{\text{max}}$).

It's also worth noting that the magnitude of the knots doesn't make any difference. All that counts is just the ratios of the values to each other. So the knot vectors [0 0 0 1 2 2 2], [100 100 100 200 300 300 300] and [-0.5 -0.5 -0.5 0 0.5 0.5 0.5] all produce the same curve. In general, integer sequences starting at zero are used to keep things simple.

Curve Orders

The most common orders for curves are 2 (linear), 3 (quadratic) and 4 (cubic). A curve with order 2 is simply a polyline. Quadratic curves are commonly used to represent arcs and sections of an ellipse, and for simple applications like "smoothing" the corners of polygons (a la MacDraw). TrueType™ also uses quadratic curves for the outline description of characters. Cubic curves are commonly used as free-from design tools in programs like Illustrator and Freehand.



Curve Orders

The effect of moving a particular control point depends on the order of the curve. If the order is high, moving a single control point affects more of the curve than if the order is low.

As the order increases, it takes longer to render the curve. Orders greater than cubic are rarely used directly by users. High orders usually arise as a result of computations like deformation or interpolation (beyond the scope of this memo).

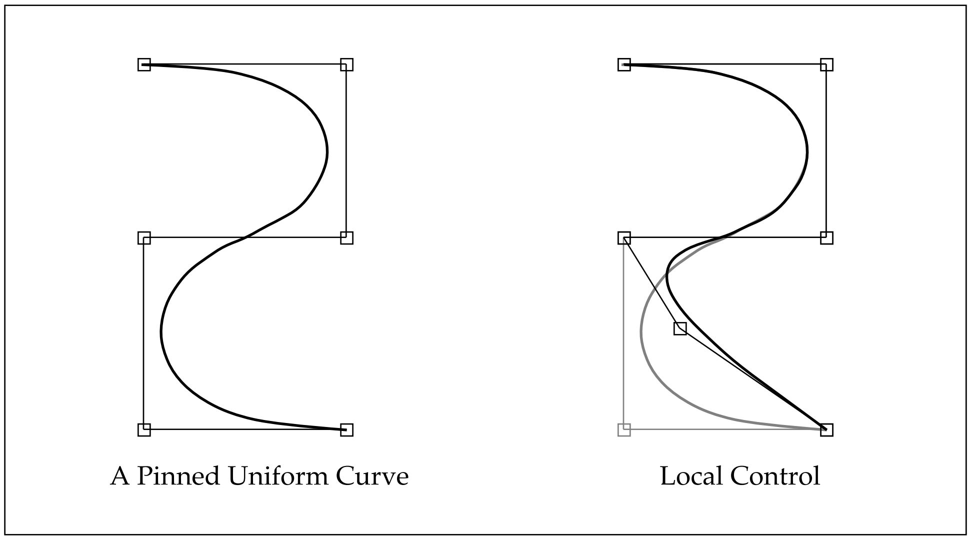
Basic knot vectors

Pinned Uniform

The most common knot vector is the *pinned uniform.* The curve passes through the first control point, and ends up at the last control point (i.e., the endpoints are "pinned"). In between, the curve passes near the control points but doesn't go through them.

If the number of control points is equal to the order, then moving any control point changes the shape of the entire curve[2]. If we have more control points, then moving a particular point only affects the curve near that point (this property is called *local control*).

[2]A pinned uniform curve with the number control points equal to the order is by definition a Bézier curve.

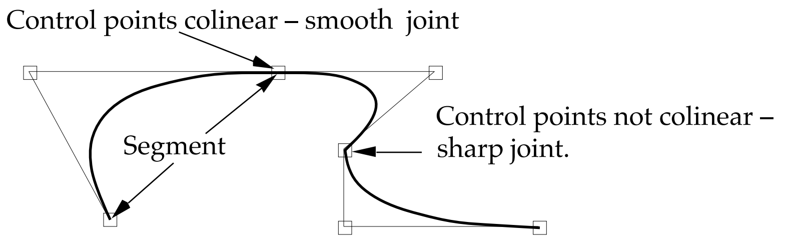


Just so you know what it looks like, here's the format of the pinned uniform knot vector. If *k* is the order, then the first *k* knots have the same value. Then the knots increase in uniform steps (usually by one) until we get to the last *k* knots which are also the same. So for the cubic ($k = 4$) curve above, with six control points, the knot vector would look like: [0 0 0 0 1 2 3 3 3 3].

Piecewise Bézier

Another common knot vector format is for piecewise Bézier curves. These are usually used when a bunch of simple primitives, such as arcs, lines or simple curve segments are strung together into a single large curve. Again, if *k* is the order, then the curve passes through each *k*'th point on the curve, and passes near all of the others. In other words, the curve is broken into segments of *k* points each. Moving any control point within a particular segment affects only that segment, and moving a point where two segments join affects both. If the joint and the control points on either side of it are in a straight line, then the two segments form a smooth continuous curve.[3] Otherwise a sharp "kink" or discontinuity occurs. (This is discussed in more detail in the section on continuity).

[3]This is what the "rocker arm" interface in Adobe Illustrator is doing. Each arm is really three control points - the joint between two segments and a point on either side. This user interface forces the three points to be in a straight line, avoiding kinks in the curve.


Control points colinear – smooth joint
Segment
Control points not colinear – sharp joint.

Piecewise Bézier curve

A quadratic piecewise Bézier knot vector with seven control points will look like this [0 0 0 1 1 2 2 3 3 3]. In general, in a piecewise Bézier knot vector the first *k* knots are the same, then each subsequent group of *k*-1 knots is the same, until you get to the end. Note that a piecewise Bézier curve must have *nk*–1 control points, where *n* is the number of segments (i.e., the number of control points is one less than an even multiple of the order).

Kinks and breaks – continuity.

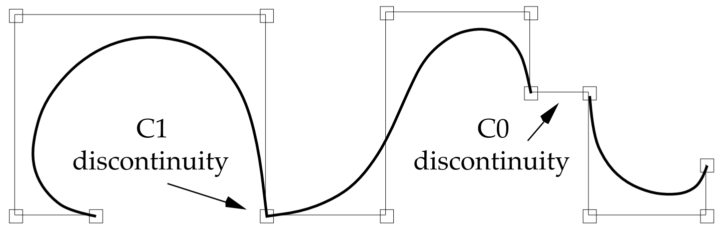
"Continuity" is a math buzzword for describing the smoothness of a curve. If a kink, break or speed change occurs in the curve this is called a *discontinuity*. The most useful kinds of discontinuity[4] are:

- C_0 – Change in position - in this case, the curve actually has a break in it. [paths w/multiple pieces, etc]

- C_1 – Change in tangent - a kink in the curve (e.g., the corner of a rectangle).

- C_2 – Change in acceleration - e.g., a change in the speed of a camera as it moves along a path.

[4]Technically, a C*n* discontinuity is a jump in the *n*th derivative.



Curve discontinuities

You get a C_n discontinuity in a curve if the knot vector contains the same value *k–n* times in a row.[5] For example, in the cubic curve above the knot vector is [0 0 0 0 1 2 2 2 3 4 4 4 4 5 5 5 5]. The C_1 discontinuity occurs at 2 ($k - 1 = 3$ two's in a row) and the C_0 discontinuity occurs at 4 ($k - 0 = 4$ four's in a row). You can use the knot insertion techniques discussed below to add discontinuities to curves.

Note: Some textbooks suggest placing two or three control points in the same place to get discontinuities. This approach isn't recommended, because it forces a portion of the curve on either side of the multiple control point to be a straight line.

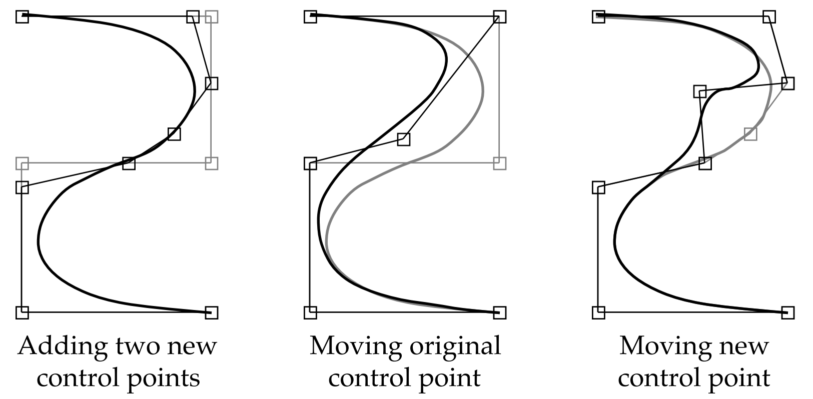
Adding Control points

It is possible to add control points to a curve without changing it's shape. This process is called *refinement* or *knot insertion.* Knot insertion is a way of adding more flexibility to a curve, and also restricting the effects of moving a particular control point. If a particular curve has more control points, the effect of moving any one of them is smaller.

Remember in the rules section that the total number of knots is equal to the order plus the number of control points. So each time you add ("insert") a knot, another control point is added to the curve. The placement of the new point depends on the value of the new knot. Curve refinement moves the other control points near the new one to preserve the shape of the curve. As you add more knots, the control points get closer and closer to the actual curve.

[5]Take a quick look back at the piecewise Bézier knot vector – you'll see *k* knots in a row on either end, and the curve has C_0 discontinuities there. In between the ends the knots are in groups of *k*–1 groups with the same value, and we have C_1 discontinuities (if the control points aren't in a straight line).

In the example below, the original knot vector was [0 0 0 0 1 2 3 3 3 3]. In order to add some flexibility, two control points are added at 1.5 and 2.5, making the knot vector [0 0 0 0 1 1.5 2 2.5 3 3 3 3]. Note how moving a control point in the third picture effects a much smaller portion of the curve than in the second picture.



Adding control points

If you insert multiple knots with the same values, you'll introduce the discontinuities described in the previous section (of course, the kink or a break won't actually appear until you move the control points).

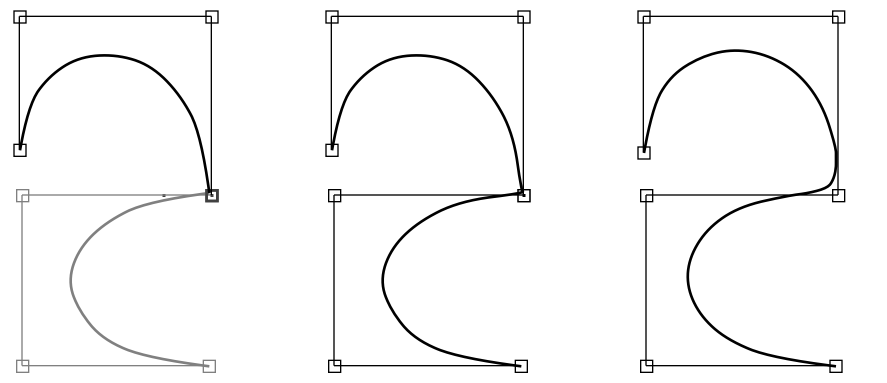
Joining Curves

As a final example of using knot vectors, it's sometimes desirable to join two adjoining curves and then smooth out the join between them. The knot vectors provide a straightforward mechanism for doing this. Suppose we have two pinned uniform cubic curves, with knot vectors starting at zero, that meet at a common point. Here's how to hook them up:

1. Add the last value in the first curve's knot vector to all the knots in the second curve.

2. Remove the first control point and the *k* first knots from the second curve (*k* = order). Remove the last knot from the first curve. Concatenate the second curve's control points and knot vector.

3. To smooth the join, space apart the knots where the two knot vectors were joined. This removes the discontinuity. As they're moved further apart, the join becomes less sharp.



Three steps in joining two curves.

Further Reading

The sections above gave several examples of how the knot vectors are commonly used. Along with the methods provided in Albert, this should be enough to get you going.

If you want to understand more of the mathematics behind curves and knot vectors, the book by Bartels, Beatty and Barsky, *An Introduction to Splines for use in Computer Graphics and Geometric Modeling* (Morgan Kaufmann, 1987) is an excellent place to start. The first four chapters give an introduction to the math and the notation. Chapter eight gives a good overview of the various properties of splines. Chapters 5-7 provide a detailed description of the mathematics. The book also has good descriptions of refinement algorithms, rendering techniques and applications.

Gerald Farin's book, "Curves and Surfaces for Computer Aided Geometric Design" (Academic Press, 1988) also discusses curve formulations, using a somewhat different approach. Farin's book has a nice introduction to rational curves, used to represent exact arcs and ellipses (chapter 15). Both books have excellent bibliographies.