Table of Contents

Curve and Surface Design

Parametric curves may be used to describe the motion of objects or that of the synthetic camera. For these curves to describe smooth motions of the camera, they must have a first order derivative that is also continuous. If a user wants the camera to look in the direction of the motion, then a vector tangent to the curve at the position of the camera must be computed in such a way as to determine the gaze direction of the camera. Suppose the the trajectory of the camera is described by the following parametric curve:

$$p(t)=(X(t),Y(t),Z(t))$$

The velocity vector is tangent to this curve and can be used to determine the gaze direction:

$$\vec{v}(t)=\frac{dP(t)}{dt}=\left(\frac{dX(t)}{dt},\frac{dY(t)}{dt},\frac{dZ(t)}{dt}\right)$$

It is then clear that this curve design mechanism requires the parametric curve to be differentiable and that its derivative be also differentiable, such that the gaze vector moves smoothly along the camera path.

Designing Polynomial Bezier Curves

We use control points that are usually specified by a user, and proceed to fit a curve to these points. There are two general methods of accomplishing this: a curve interpolates the points if the curve passes through all the control points, and a curve approximates the points if the curve contains the start and end points only.

Suppose we have three control points P_0, P_1, and P_2. Let's find an approximating curve to those points. First, let's define a parametric form for the first segment from P_0 to P_1 and from P_1 to P_2:

$$\begin{aligned} A(t) &= (1-t)P_0+tP_1 \\ B(t) &= (1-t)P_1+tP_2 \end{aligned}$$

We can define a new segment between $A(t)$ and $B(t)$:

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

This results in $P(t)$ approximating the three control points for $t \in [0,1]$. $P(t)$ has a parametric representation, obtained as:

$$\begin{aligned} P(t) &= (1-t)A(t)+tB(t) \\ &= (1-t)((1-t)P_0+tP_1)+t((1-t)P_1+tP_2) \\ &= (1-t)^2P_0+2(1-t)tP_1+t^2P_2 \end{aligned}$$

This parametric form is a parabola and approximates the control points. If we had four control points, a similar reasoning would lead us to the following expression for their approximation:

$$P(t)=(1-t)^3P_0+3(1-t)^2tP_1+3(1-t)t^2P_2+t^3P_3$$

which is a cubic polynomial in t approximating the four control points. These coefficients to the control points are known as Bernstein's polynomials:

$$\begin{aligned} B_0^3(t) &= (1-t)^3 \\ B_1^3(t) &= 3(1-t)^2t \\ B_2^3(t) &= 3(1-t)t^2 \\ B_3^3(t) &= t^3 \end{aligned}$$

It is important to note that Bernstein's polynomials sum to unity within $t \in [0,1]$. Hence by rewriting the parametric approximating function for four points as,

$$p(t)=B_0^3P_0+B_1^3P_1+B_2^3P_2+B_3^3P_3$$

it becomes clear that it represents a convex affine combination of the four control points. Bernstein's polynomials used in this context are referred to as blending functions, as they blend the control points into a smooth approximation.

When $L+1$ points are present, then the Bezier curve is given by:

$$P(t)=\sum_{k=0}^{L} B_k^L(t)P_k$$

where

$$B_k^L(t)=\left(\frac{L!}{k!(L-k)!}\right)(1-t)^{L-k}t^k$$

for $k<L+1$. The properties of Bezier curves are many:

- The first and last control points of a Bezier curve are always interpolated
- A Bezier curve is affine-invariant
- A Bezier curve is always within the boundaries defined by the convex hull of the control points

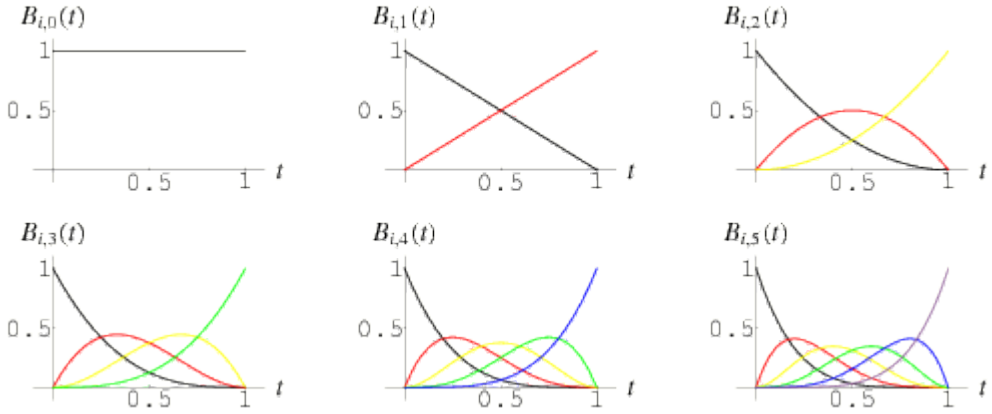

Illustration 1: Bernstein's polynomials of various orders

Derivatives of Bezier curves exist and they are continuous:

$$P'(t) = L\sum_{k=0}^{L-1} B_k^{L-1}(t)\,\Delta P_k$$

where $\Delta P_k = P_{k+1} - P_k$.

While Bezier curves are useful for some applications, they suffer from problems in other types of use. For instance, the exponents involved tend to be large when one has multiple control points. Additionally, local control of the curve is hard to achieve since the entire set of control points influence the curve at any point (except at $t=0$ and $t=1$). It is useful to define a set of blending functions that possess the properties of Bezier curves, but that also provide local control over the curve. That is to say, we would like the influence of control points to be local at any point on the curve.

Piecewise Polynomial Curves and Splines

By defining piecewise polynomial curves as blending functions, we can control their extent over the domain of the parameter t and thus have only a subset of blending functions determining the influence of neighboring control points on the curve in a local way. Consider the following piecewise polynomial:

$$g(t) = \begin{cases} a(t) &= \frac{1}{2}t^2 & t \in [0,1] \\ b(t) &= \frac{3}{4} - \left(t - \frac{3}{2}\right)^2 & t \in [1,2] \\ c(t) &= \frac{1}{2}(3-t)^2 & t \in [2,3] \end{cases}$$

- The support for this piecewise polynomial is $[0,3]$
- The joints are the locations where the polynomials meet
- The knots are the t values at which the polynomials meet

The polynomial $g(t)$ would work well for local control of a curve.

Definition of B-Spline Functions

The definition of a B-Spline is:

$$P(t)=\sum_{k=0}^{L} P_k N_{k,m}(t)$$

where $N_{k,m}(t)$ is the k^{th} B-Spline of order m. There are $L+1$ control points and a knot vector $\vec{T}=(t_0, t_1, \ldots)$ specifies the values of t at the joints. The recursive definition of a B-Spline is thus:

$$N_{k,m}(t)=\left(\frac{t-t_k}{t_{k+m-1}-t_k}\right)N_{k,m-1}(t)+\left(\frac{t_{k+m}-t}{t_{k+m}-t_{k+1}}\right)N_{k+1,m-1}(t)$$

$$N_{k,1}(t)=\begin{cases}1 & if\ t_k<t\le t_{k+1}\\ 0 & otherwise\end{cases}$$

for $k=0,1,\ldots,L$. What is the shape of a quadratic B-Spline $N_{0,3}(t)$, given a knot vector $\vec{T}=(t_0=0, t_1=1, \ldots)$?

$$\begin{aligned}N_{0,3}(t) &= \frac{t}{2}N_{0,2}(t)+\frac{3-t}{2}N_{1,2}(t)\\ &= \frac{t}{2}\left[\left(\frac{t-t_0}{t_1-t_0}\right)N_{0,1}(t)+\left(\frac{t_2-t}{t_2-t_1}\right)N_{1,1}(t)\right]+\left(\frac{3-t}{2}\right)\left[\left(\frac{t-t_1}{t_2-t_1}\right)N_{1,1}(t)+\left(\frac{t_3-t}{t_3-t_2}\right)N_{2,1}(t)\right]\end{aligned}$$

which turns out to be:

$$N_{0,3}(t)=\begin{cases}\frac{1}{2}t^2 & t\in[0,1]\\ \frac{3}{4}-\left(t-\frac{3}{2}\right)^2 & t\in[1,2]\\ \frac{1}{2}(3-t)^2 & t\in[2,3]\end{cases}$$

In general, the cubic B-Spline is the most used. $N_{0,4}(t)$ is symmetrical at $t=2$ and can be compactly written as:

$$N_{0,4}=\begin{cases}u(1-t) & 0\le t\le 1\\ v(2-t) & 1\le t\le 2\\ v(t-2) & 2\le t\le 3\\ u(t-3) & 3\le t\le 4\end{cases}$$

where

$$u(t) = \frac{1}{6}(1-t)^3$$
$$v(t) = \frac{1}{6}(3t^3-6t^2+4)$$

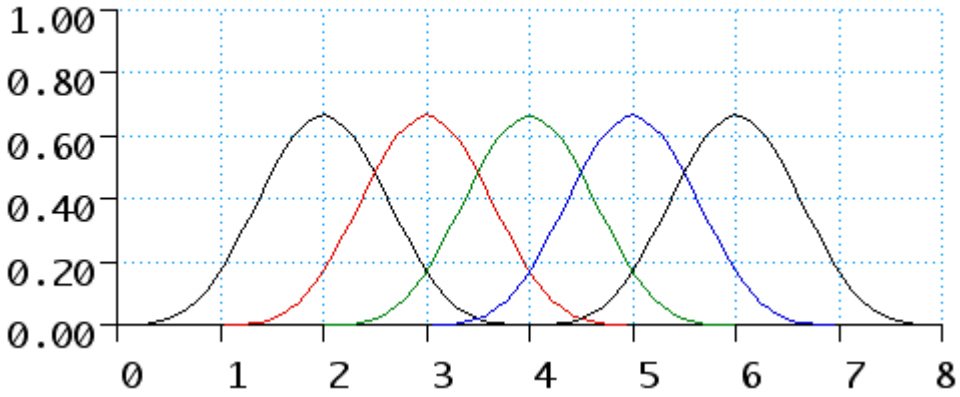

Illustration 2: The shape of the polynomials for a uniform B-Spline

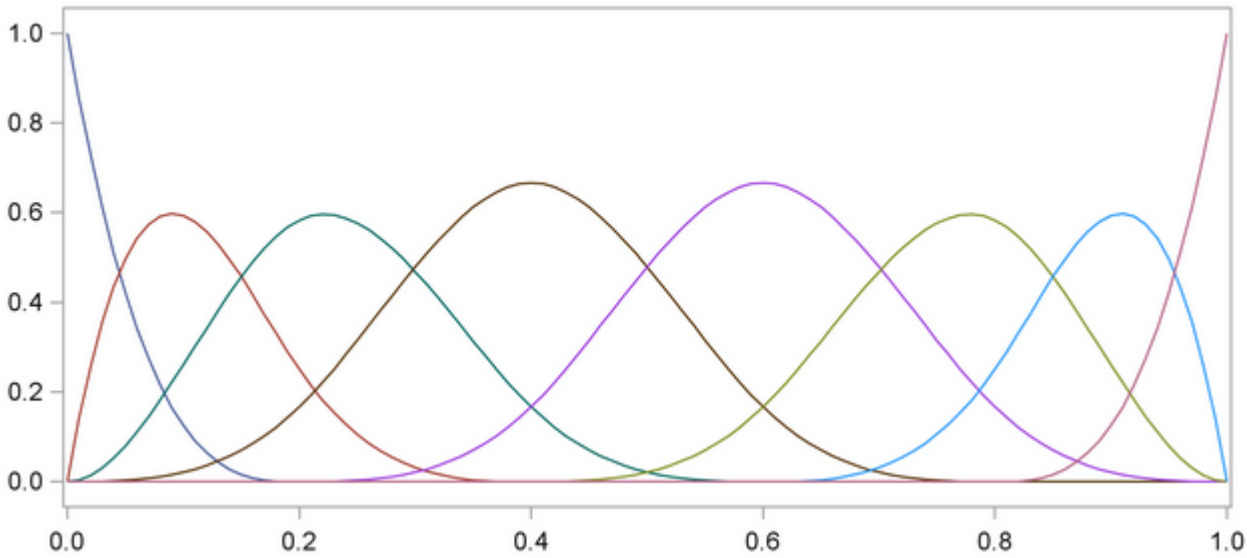

Illustration 3: The shape of the polynomials for an open B-Spline with standard knot vector.

Open B-Splines

Bezier splines are special cases of B-Splines. B-Splines acting as Bezier splines are called open B-Splines. With open B-Splines, the knot vector (known as the standard knot vector in this case) is arranged so that the B-Spline will interpolate the first and the last point. In order

to achieve this, we create a knot vector containing $L+m+1$ knots. The first m knots $t_{0,}\ldots,t_{m-1}$ are set to zero while the last m knots $t_{L+1},\ldots,t_{L+m}$ are set to $L-m+2$. The knots $t_m,\ldots,t_L$ start at value 1, and increase by one up to $L-m+1$. Hence, the order m cannot exceed the number of control points. Also, it is easy to verify that when $m=L+1$, then $N_{k,L+1}(t)=B_k^L(t)$. For example, if we have $L+1=8$ control points and we are using cubic B-Splines ($m=4$), then the standard knot vector is $T=(0,0,0,0,1,2,3,4,5,5,5,5)$. This B-Spline will interpolate the first and last control points.

Properties of B-Splines

- They are piecewise polynomials of order m which are $m-2$ smooth (that is to say, they have $m-2$ continuous successive derivatives.
- $N_{k,m}(t)$ Has support over $[t_k,t_{k+m}]$.
- The last and first control points are interpolated when the standard knot vector is used.
- A B-Spline is always contained within the convex hull formed by the control points.

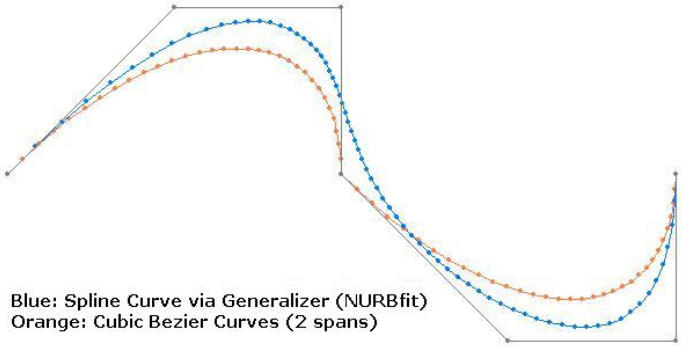

Illustration 4: For the same set of control points the Bezier and B-Spline curves can be quite different.

Putting several control points at the same coordinates attracts the curve more strongly to that point. If we have a Spline of order m , then $m-1$ consecutive multiple points at the same coordinates will make the spline interpolate that point. While knot multiplicity and control point multiplicity are not equivalent, some of the effects are similar.

Rational Splines

We may also use weights to even better control the shape of the curve. For each control point we can associate a weight, known as a shape parameter. If there are $L+1$ control points, then we would have the weights: $\Omega=(\omega_{0,}\omega_{1,}\ldots,\omega_L)$ for control points $P_{0,}P_{1,}\ldots,P_L$, and the rational spline would be written as:

$$P(t)=\sum_{k=0}^{L} R_k(t)$$

where

$$R_k(t) = \frac{\omega_k P_k N_{k,m}(t)}{\sum_{j=0}^{L} \omega_j N_{j,m}(t)}$$

Of course if all the weights are made equal, then this equation reverts to a normal Spline.

B-Spline Surface Patches

The generalization for curves to 2D surface patches is straightforward. A B-Spline patch is given by:

$$P(u,v) = \sum_{i=0}^{M} \sum_{k=0}^{L} P_{i,k} N_{i,m}(u) N_{k,n}(v)$$

and, in the case of a rational B-Spline surface patch (NURB):

$$P(u,v) = \frac{\sum_{i=0}^{M} \sum_{k=0}^{L} \omega_{i,k} P_{i,k} N_{i,m}(u) N_{k,n}(v)}{\sum_{i=0}^{M} \sum_{k=0}^{L} \omega_{i,k} N_{i,m}(u) N_{k,n}(v)}$$

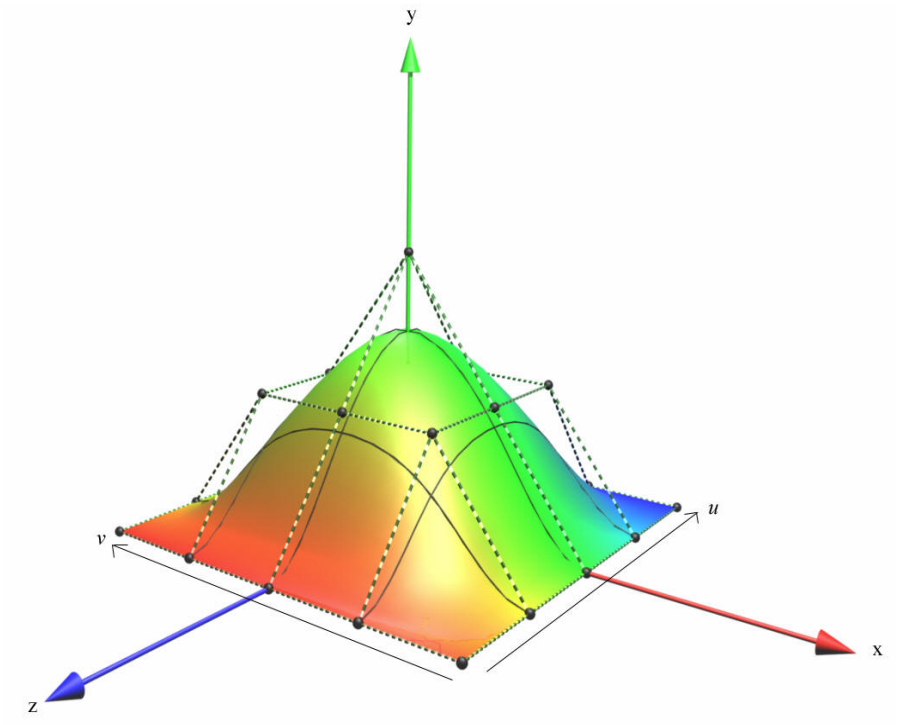

Illustration 5: A NURBS surface defined with control points