Higher Order Surfaces

Polygons may not be the best modeling primitive

3) Polygonal models have a fixed resolution

5) Parameterization of polygonal models is hard

7) Differential operations on polygons are hard

9) Deformations of polygonal models are hard

Parametric Curves

Parametric Curve

$$Q(u) = \left(X(u), Y(u), Z(u) \right)$$

+ Analytic form
+ Concise
+ Easily deformable
+ Variable resolution

- Not obvious how to render
- Hard to extract from point data (3D scanners…)

Parametric Curves

Parametric Polynomial Curve (order k+1)

$$Q(u) = a_0 + a_1 u^1 + \ldots + a_k u^k$$

Usually use cubics (k = 3)

$$Q(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$$

- Lower orders are not C^2 continuous
- Higher orders may have unwanted oscillations

Specifying the coefficients a_i is hard

Using Basis Functions

Transform to another basis

$$Q(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$$

$$= p_0 b_0(u) + p_1 b_1(u) + p_2 b_2(u) + p_3 b_3(u)$$

$$= \sum_{i=0}^{3} p_i b_i(u)$$

p_i are known as control points

Properties of Basis Functions

Convex Hull Property

$$\sum_{i=0}^{k} b_i(u) = 1, \qquad b_i(u) \geq 0$$

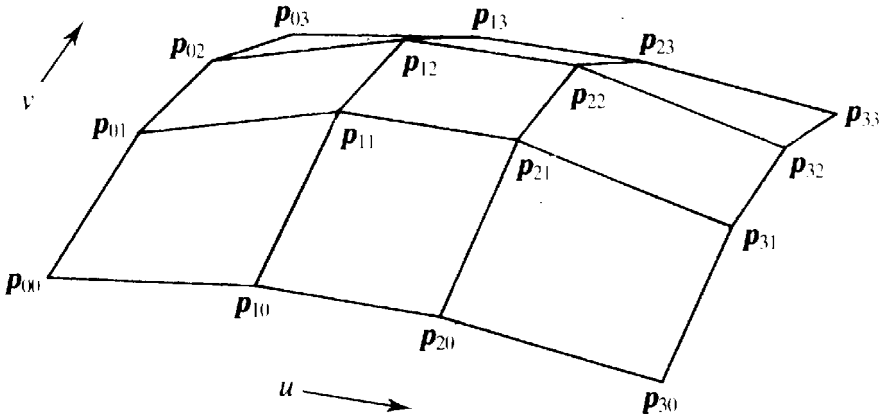
Affine Invariance

$$\Phi\big(Q(u)\big) = \Phi\left(\sum_{i=0}^{k} p_i b_i(u)\right) = \sum_{i=0}^{k} \Phi\big(p_i\big) b_i(u)$$

Extension to Surfaces

Surface is just a tensor product

$$Q(u,v) = \sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij} b_i(u) b_j(v)$$



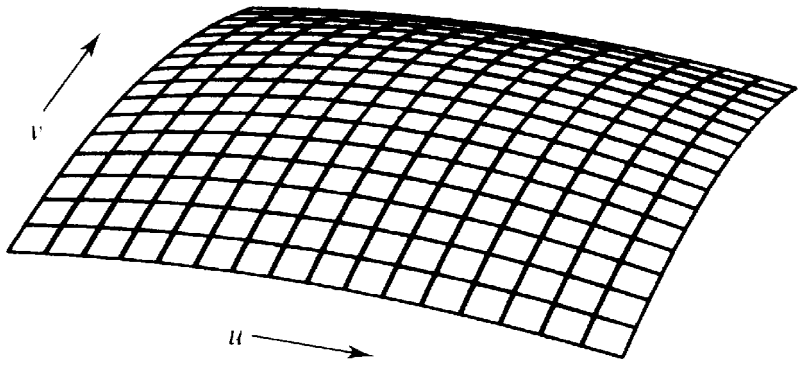


Image courtesy of Watt & Watt, Advanced Animation and Rendering Techniques

Bezier Curves

Use Bernstein polynomials as basis functions

$$Q(u) = \sum_{i=0}^{n} p_i B_{i,n}(u)$$

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

Bezier Curves

Cubic case

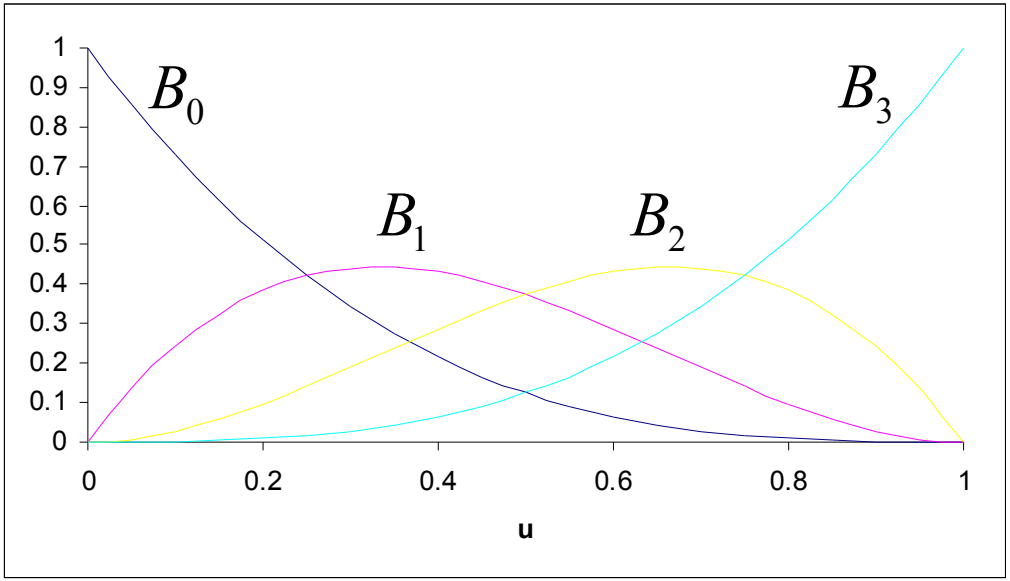
$$Q(u) = \sum_{i=0}^{3} p_i B_{i,3}(u)$$

$$B_{0,3}(u) = (1-u)^3$$

$$B_{1,3}(u) = 3u(1-u)^2$$

$$B_{2,3}(u) = 3u^2(1-u)$$

$$B_{3,3}(u) = u^3$$



Bezier Curves

Matrix Form

$$Q(u) = UMP = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$\begin{bmatrix} B_0(u) & B_1(u) & B_2(u) & B_3(u) \end{bmatrix} = UM$$

Differentiation of Bezier Curves

Derivative with respect to u

$$Q_u(u) = \sum_{i=0}^{n} p_i \left(\frac{d}{du} B_{i,n}(u) \right)$$

Endpoint derivatives

$$Q_u(0) = n(p_1 - p_0)$$

$$Q_u(1) = n(p_n - p_{n-1})$$

Properties of Bezier Curves

Endpoint Interpolation

$$Q(0) = p_0 \qquad Q(1) = p_n$$

Endpoint Derivatives

$$Q_u(0) = n(p_1 - p_0) \quad Q_u(1) = n(p_n - p_{n-1})$$

Convex Hull Property

$$\sum_{i=0}^{n} B_{i,n}(u) = 1$$

Affine Invariance

$$\Phi\big(Q(u)\big) = \sum_{i=0}^{n} \Phi(p_i) B_{i,n}(u)$$

Degree Elevation of Bezier Curves

Create an degree n+1 curve from a degree n curve:

$$p_0, p_1, p_2, \cdots, p_{n-1}, p_n$$

$$p_0, \frac{p_0 + p_1}{n+1}, \frac{2p_{n-1} + (n-1)p_n}{n+1}, \cdots, \frac{np_{n-1} + p_n}{n+1}, p_n$$

Joining of Bezier Curves

Join two Bezier curves with control points:

$$Q_1(u) : p_i \qquad Q_2(u) : r_i$$

C^0 continuity: $p_n = r_0$

G^1 continuity: $(p_n - p_{n-1}) = k(r_1 - r_0)$

C^1 continuity: $(p_n - p_{n-1}) = (r_1 - r_0)$

Rendering Bezier Curves

De Casteljau representation:

$$Q(u) = p_0^n(u)$$

where

$$p_i^r(u) = (1-u)\,p_i^{r-1}(u) + u\,p_{i+1}^{r-1}(u)$$

$$p_i^0(u) = p_i$$

Rendering Bezier Curves

De Casteljau actually subdivides the curve in 2:

$$Q_a(u) = p_0^i(u) \quad i = 0..n$$
$$Q_b(u) = p_i^{n-i}(u) \quad i = 0..n$$

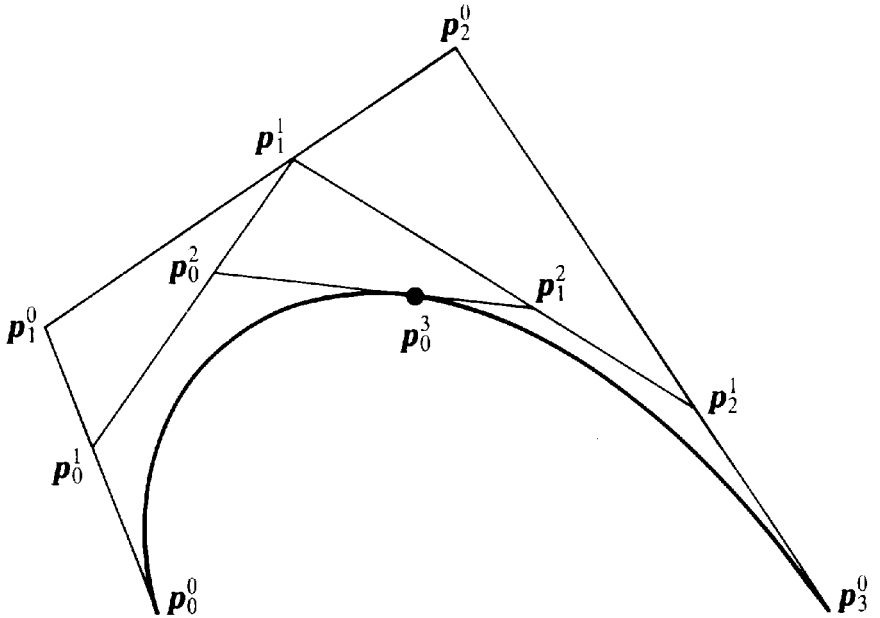


Image courtesy of Watt & Watt, Advanced Animation and Rendering Techniques

Rendering Bezier Curves

Midpoint subdivision (u=0.5) (q,r are the new curves)

$$q_0 = p_0 \qquad r_0 = q_3$$

$$q_1 = \frac{p_0 + p_1}{2} \qquad r_1 = \frac{r_2}{2} + \frac{p_1 + p_2}{4}$$

$$q_2 = \frac{q_1}{2} + \frac{p_1 + p_2}{4} \qquad r_2 = \frac{p_2 + p_3}{2}$$

$$q_3 = \frac{q_2 + r_1}{2} \qquad r_3 = q_3$$

Bezier Patches

Tensor Product of Bezier Curves

$$Q(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{n} p_{ij} B_{i,n}(u) B_{j,n}(v)$$

Matrix Form

$$Q(u,v) = U\ M\ P\ M^T V^T$$

P is now a matrix

Differentiation of Bezier Patches

Endpoint derivatives

$$Q_u(0,0) = n(p_{10} - p_{00})$$

$$Q_v(0,0) = n(p_{01} - p_{00})$$

$$Q_{uv}(0,0) = n^2(p_{00} - p_{01} - p_{10} + p_{11})$$

Other endpoint derivatives have similar forms

Joining Bezier Patches

Join two patches

$$Q_1(u,v): p_{ij} \qquad Q_2(u,v): r_{ij}$$

C^0 continuity: $p_{nj} = r_{0j}$

G^1 continuity: $(p_{ni} - p_{(n-1)i}) = k(r_{1i} - r_{0i})$

C^1 continuity: $(p_{ni} - p_{(n-1)i}) = (r_{1i} - r_{0i})$

Rendering Bezier Patches

Patch Splitting

$$Q(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{n} p_{ij} B_{i,n}(u) B_{j,n}(v)$$

$$= \sum_{j=0}^{n} B_{j,n}(v) \sum_{i=0}^{n} p_{ij} B_{i,n}(u)$$

This means that we can split in each direction separately

Rendering Bezier Patches

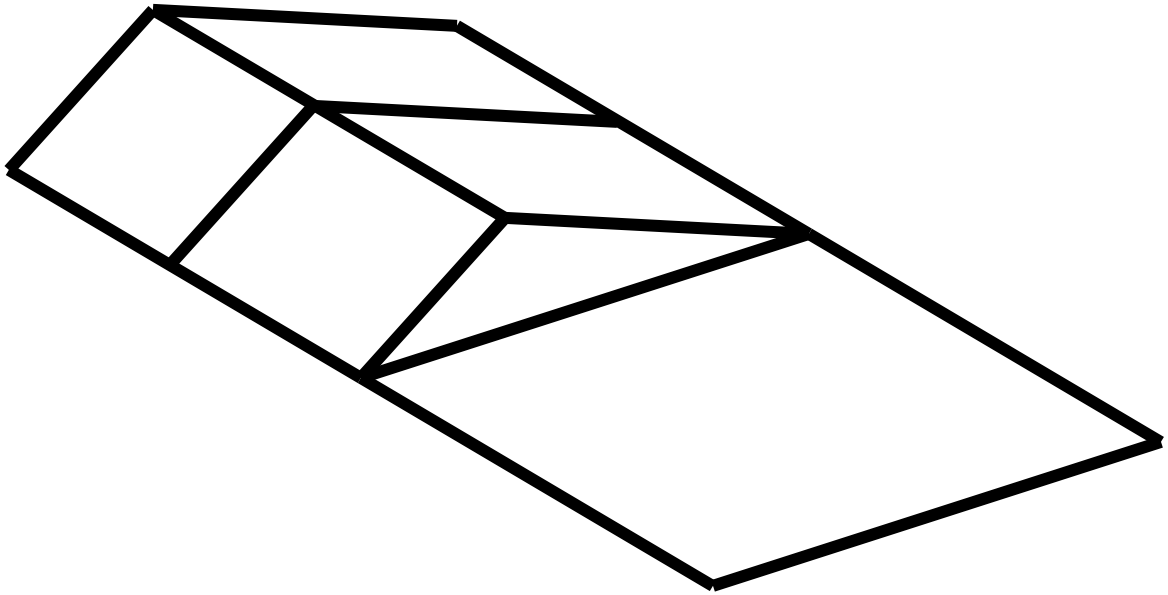
Splitting Termination

Stop when the patch is sufficiently planar
-make a plane out of 3 vertices
and measure how far the others are from the plane

Stop when the patch is sufficiently small

Rendering Bezier Patches

Patch Cracking



When one patch is subdivided independently of another, cracking may occur!!!

Changing Bases

Converting from one basis to another

$$Q(u) = UM_1P_1 \qquad Q(u) = UM_2P_2$$

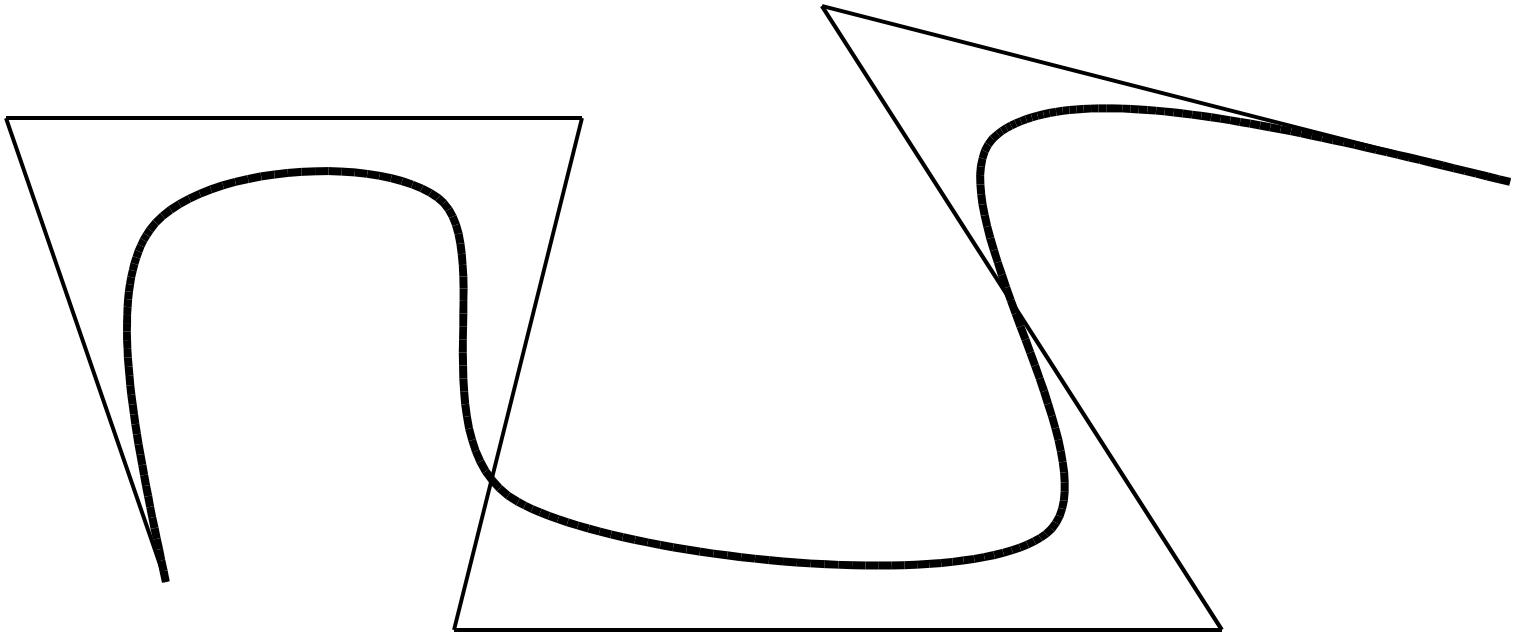
$$UM_1P_1 = UM_2P_2$$

$$M_1P_1 = M_2P_2$$

$$P_1 = M_1^{-1}M_2P_2$$

B Splines

Derive a piecewise cubic curve with C^2 continuity



B Splines

Segment Q_i is defined by the control points:

$$Q_i : p_i, p_{i+1}, p_{i+2}, p_{i+3}$$

Segment Q_{i+1} is defined by the control points:

$$Q_{i+1} : p_{i+1}, p_{i+2}, p_{i+3}, p_{i+4}$$

B-spline segments share control points

B Splines

Segment Q_i defined as:

$$Q_i(u) = \sum_{k=0}^{3} p_{i+k} B_k(u)$$

Segment Q_{i+1} defined as:

$$Q_{i+1}(u) = \sum_{k=0}^{3} p_{i+k+1} B_k(u)$$

B Splines

C^0 continuity implies $Q_i(1) = Q_{i+1}(0)$

$$\sum_{k=0}^{3} p_{i+k} B_k(1) = \sum_{k=0}^{3} p_{i+k+1} B_k(0)$$

Equating coefficients

$$B_0(1) = 0$$

$$B_1(1) = B_0(0)$$

$$B_2(1) = B_1(0)$$

$$B_3(1) = B_2(0)$$

$$0 = B_3(0)$$

B Splines

C^1 continuity

$$B_0'(1) = 1$$

$$B_1'(1) = B_0'(0)$$

$$B_2'(1) = B_1'(0)$$

$$B_3'(1) = B_2'(0)$$

$$0 = B_3'(0)$$

C^2 continuity

$$B_0''(1) = 1$$

$$B_1''(1) = B_0''(0)$$

$$B_2''(1) = B_1''(0)$$

$$B_3''(1) = B_2''(0)$$

$$0 = B_3''(0)$$

B Splines

15 equations and 16 unknowns

Need an additional constraint

$$B_0(0) + B_1(0) + B_2(0) + B_3(0) = 1$$

B Splines

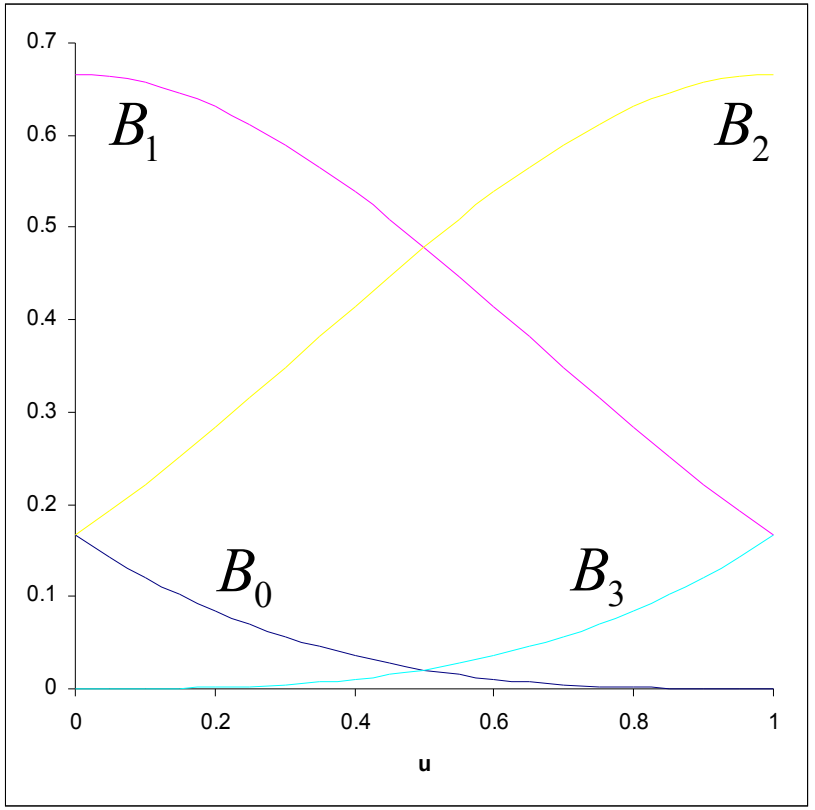
Solving yields

$$B_0(u) = \frac{1}{6}(1-u)^3$$

$$B_1(u) = \frac{1}{6}(3u^3 - 6u^2 + 4)$$

$$B_2(u) = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)$$

$$B_3(u) = \frac{1}{6}u^3$$



B Splines

Matrix form

$$Q_i(u) = UMP$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ p_{i+1} \\ p_{i+2} \\ p_{i+3} \end{bmatrix}$$

Uniform B Splines

Given a local u for the i^{th} curve segment

$$u \in [0,1] \qquad i = 0,\ldots,n-3$$

Find a global parameterization U

$$U = u + i \qquad U \in [0, n-2]$$

Uniform B Splines

Control point p_i is involved in the segments

$$Q_{i-3}, Q_{i-2}, Q_{i-1}, Q_i$$

weighted by the functions

$$B_3, B_2, B_1, B_0$$

Collect these into a single blending function

Uniform B Splines

Blending functions

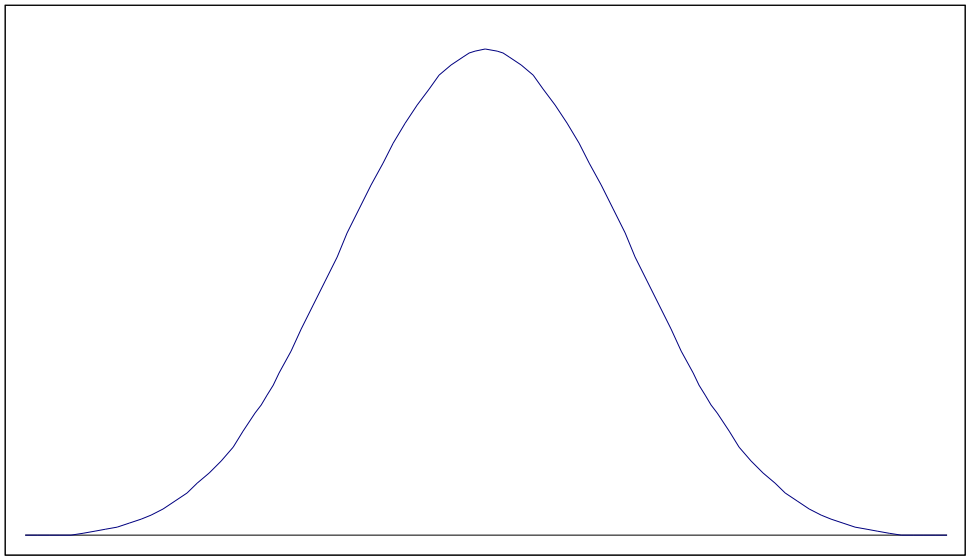
$$Q(U) = \sum_{i=0}^{n} p_i N_i(U)$$

$$N_i(U) = \begin{cases} B_3(U-(i-3)) & i-3 \le U \le i-2 \\ B_2(U-(i-2)) & i-2 \le U \le i-1 \\ B_1(U-(i-1)) & i-1 \le U \le i \\ B_0(U-(i-0)) & i \le U \le i+1 \end{cases}$$

Uniform B Splines

Blending functions

$$N_i(U) = \begin{cases} B_3(U-(i-3)) & i-3 \le U \le i-2 \\ B_2(U-(i-2)) & i-2 \le U \le i-1 \\ B_1(U-(i-1)) & i-1 \le U \le i \\ B_0(U-(i-0)) & i \le U \le i+1 \end{cases}$$



Uniform B Splines

Curve segments end at uniform intervals

$$u = 0,1,2,...,n$$

Call these positions knots,
and the vector of knots, the knot vector

$$knots = 0,1,2,...,n$$

$$knotVector = [0,1,2,...,n]$$

Nonuniform B Splines

Generalize knots to nonuniform intervals

$$knotVector = [t_0, t_1, t_2, ..., t_n]$$

$$t_0 \leq t_1 \leq t_2 \leq ... \leq t_n$$

Nonuniform B Splines

Cox de Boor algorithm for blending functions for a curve of order k

$$N_{i,1}(u) = \begin{cases} 1 & if\ t_i \le u \le t_{i+1} \\ 0 & otherwise \end{cases}$$

$$N_{i,k}(u) = \frac{(u - t_i)N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u)N_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}}$$

Properties of Blending Functions

Convex Hull property

$$\sum_{i=0}^{n} N_{i,k}(u) = 1 \qquad N_{i,k}(u) \geq 0$$

Finite Support

$$N_{i,k}(u) \neq 0 \qquad u \in [t_i, t_{i+k}]$$

Multiple Knots

$$0/0 \equiv 0$$ Allows multiple knots to be defined

Control Points and Knots

$$Q(u) = \sum_{i=0}^{n} p_i N_{i,k}(u)$$

$N_{0,k}(u)$ has support $[t_0, \ldots, t_k]$

$N_{n,k}(u)$ has support $[t_n, \ldots, t_{n+k}]$

Therefore, there are:

$n+1$ control points

$n+k+1$ knots

$$\#knots = \#control\ \ points + order$$

Control Points and Knots

$n+1$ control points

Each curve segment requires k control points

Therefore, there are:

$n-k+2$ curve segments

$n-k+2$ distinct knot intervals

$n-k+3$ distinct knots

$$\#knots - \#distinct\ knots = \#repeated\ knots$$

$$(n+k+1)-(n-k+3) = 2k-2$$

Nonperiodic Knot Vectors

Repeat the first and last knots $k-1$ more times

$$\left[\underbrace{0,\ldots,0}_{k}, t_k, \ldots, t_n, \underbrace{1,\ldots,1}_{k}\right]$$

Knots repeated k times
are said to have multiplicity k

The curve endpoints and the end control points are coincident if end knots have multiplicity k

$$Q(t_{k-1}) = p_0 \qquad Q(t_{n+1}) = p_n$$

Nonperiodic Knot Vectors

Uniform nonperiodic knot vectors

$$t_i = \begin{cases} 0 & i = 0,\ldots,k-1 \\ i-k+1 & i = k,\ldots,n-1 \\ n-k+2 & i = n,\ldots,n+k-1 \end{cases}$$

This knot vector is equivalent to the uniform B spline

NonUniform Rational B Splines

Extend control points to homogeneous space

$$Q(u) = \frac{\sum_{i=0}^{n} p_i w_i N_{i,k}(u)}{\sum_{i=0}^{n} w_i N_{i,k}(u)} = \sum_{i=0}^{n} p_i w_i R_{i,k}(u)$$

$$R_{i,k}(u) = \frac{N_{i,k}(u)}{\sum_{j=0}^{n} w_j N_{j,k}(u)}$$

w_i are called weights

Knot Insertion

Add a new knot without altering the original curve

$$Q(u) = \sum_{i=0}^{n} p_i N_{i,k}(u) \qquad t = [t_0, \ldots, t_{n+k}]$$

Adding a new knot t'

$$t_j < t' \leq t_{j+1}$$

Yields the new control points:

$$p_0^{'} = p_0$$

$$p_{n+1}^{'} = p_n$$

$$p_i^{'} = (1 - a_i) p_i + a_i p_i$$

$$a_i = \begin{cases} 1 & i = 1, \ldots, j-k \\ \dfrac{t' - t_i}{t_{i+k} - t_i} & i = j-k+1, \ldots, j \\ 0 & i = j+1, \ldots, n \end{cases}$$

Arc Length Parameterization

Generally, equally spaced samples in parameter space are not equally spaced along the curve

$$u_2 - u_1 \neq s(u_2) - s(u_1)$$

where $s(u)$ is the arc length of the curve

We can create a parameterization where parametric distance is equal to arc length

$$u_2 - u_1 = s(u_2) - s(u_1)$$

Interpolation

Given a set of data points $X_i \qquad i = 0,\ldots,n-2$

find the control points $p_i \qquad i = 0,\ldots,n$

such that the curve interpolates the data points

Set the curve breakpoints equal to the data points

$$Q(t_{i+3}) = X_i \qquad i = 0,\ldots,n-2$$

The system needs 2 additional constraints

$$p_0 = p_1 \qquad p_n = p_{n-1}$$

Interpolation

We need to evaluate the blending functions at the knot values

$$N_{i,4}(t_{i+1}) = \frac{(t_{i+1} - t_i)^2}{(t_{i+2} - t_i)(t_{i+3} - t_i)}$$

$$N_{i,4}(t_{i+2}) = \frac{(t_{i+2} - t_i)(t_{i+3} - t_{i+2})}{(t_{i+3} - t_{i+1})(t_{i+3} - t_i)} + \frac{(t_{i+4} - t_{i+2})(t_{i+2} - t_{i+1})}{(t_{i+4} - t_{i+1})(t_{i+3} - t_{i+1})}$$

$$N_{i,4}(t_{i+3}) = \frac{(t_{i+4} - t_{i+3})^2}{(t_{i+4} - t_{i+1})(t_{i+4} - t_{i+2})}$$

Interpolation

We can now write

$$Q(t_i) = N_{i-3,4}(t_i)p_{i-3} + N_{i-2,4}(t_i)p_{i-2} + N_{i-1,4}(t_i)p_{i-1}$$

Equivalently

$$Q(t_{i+3}) = N_{i,4}(t_{i+3})p_i + N_{i+1,4}(t_{i+3})p_{i+1} + N_{i+2,4}(t_{i+3})p_{i+2}$$

Which gives

$$X_i = \alpha_i p_i + \beta_i p_{i+1} + \Gamma_i p_{i+2}$$

$$\alpha_i = N_{i,4}(t_{i+3}) \qquad \beta_i = N_{i+1,4}(t_{i+3}) \qquad \Gamma_i = N_{i+2,4}(t_{i+3})$$

Interpolation

The complete set of equations is

$$\begin{bmatrix} 1 & & & & & & \\ \alpha_1 & \beta_1 & \Gamma_1 & & & & \\ & & \ddots & & & & \\ & & & \alpha_{n-3} & \beta_{n-3} & \Gamma_{n-3} & \\ & & & & & 1 & \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n-2} \\ p_{n-1} \end{bmatrix} = \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{n-3} \\ X_{n-2} \end{bmatrix}$$

and $p_0 = p_1 \qquad p_n = p_{n-1}$

Chord Length Parameterization

Heuristic approximating arc length parameteriztion

$$\frac{t_{i+4} - t_{i+3}}{t_{i+5} - t_{i+4}} = \frac{\left| X_{i+1} - X_i \right|}{\left| X_{i+2} - X_{i+1} \right|}$$

Knot spacing is proportional to the distance between data points