EECS 487: Interactive Computer Graphics

Lecture 38:

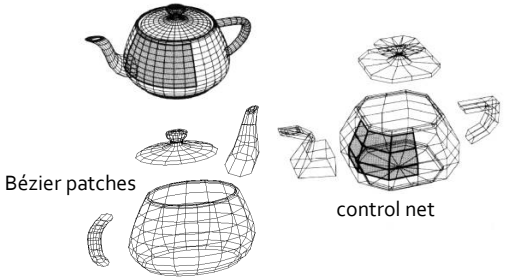
- Tensor-product surfaces: Bézier and B-spline
- Bézier subdivision curves and surfaces

Parametric Surfaces

Our discussions on parametric cubic curves can be generalized to parametric bicubic surfaces:

- parametric curves → parametric surfaces
- splines → parametric patches
- control polygon → control net

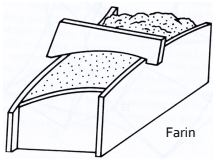
Teapot specified with Bézier patches

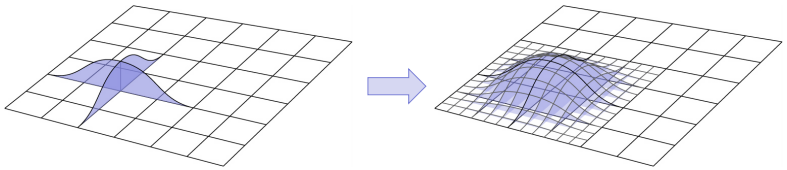


Parametric Surfaces and Patches

A surface is a curve swept through space

Instead of control points along a curve, make each control point itself a curve





Parametric patches: as with long curves, large surfaces are partitioned into piecewise parametric patches

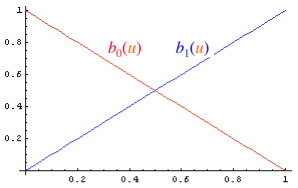
- choice of different splines: type, order, etc.

O'Brien, James

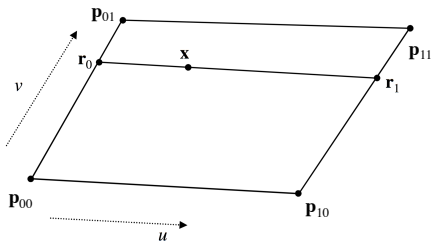
Bilinear Patch

Simplest case: 4 points, cross product of 2 linear segments

- basis function is a 3D tent: interpolates control points
- if all 4 control points are co-planar, the patch is flat



$$\mathbf{q}_0 = \mathbf{p}_{00} + u(\mathbf{p}_{10} - \mathbf{p}_{00})$$
$$= (1-u)\mathbf{p}_{00} + u\mathbf{p}_{10}$$
$$\mathbf{q}_1 = (1-u)\mathbf{p}_{01} + u\mathbf{p}_{11}$$
$$\mathbf{s}(u,v) = \mathbf{x} = (1-v)\mathbf{q}_0 + v\mathbf{q}_1$$

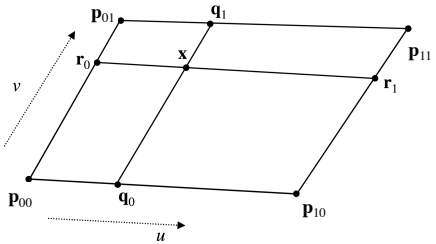


$$\mathbf{r}_0 = (1-v)\mathbf{p}_{00} + v\mathbf{p}_{01}$$
$$\mathbf{r}_1 = (1-v)\mathbf{p}_{10} + v\mathbf{p}_{11}$$
$$\mathbf{s}(u,v) = \mathbf{x} = (1-u)\mathbf{r}_0 + u\mathbf{r}_1$$

James, Schulze

Bilinear Patch

$$\begin{aligned}\mathbf{q}_0 &= \mathbf{p}_{00} + u(\mathbf{p}_{10} - \mathbf{p}_{00}) \\ &= (1-u)\mathbf{p}_{00} + u\mathbf{p}_{10} \\ \mathbf{q}_1 &= (1-u)\mathbf{p}_{01} + u\mathbf{p}_{11} \\ \mathbf{r}_0 &= (1-v)\mathbf{p}_{00} + v\mathbf{p}_{01} \\ \mathbf{r}_1 &= (1-v)\mathbf{p}_{10} + v\mathbf{p}_{11}\end{aligned}$$



$$\mathbf{s}(u,v) = \mathbf{x} = (1-v)\mathbf{q}_0 + v\mathbf{q}_1 = (1-v)\left[(1-u)\mathbf{p}_{00} + u\mathbf{p}_{10}\right] + v\left[(1-u)\mathbf{p}_{01} + u\mathbf{p}_{11}\right]$$

$$\mathbf{s}(u,v) = \mathbf{x} = (1-u)\mathbf{r}_0 + u\mathbf{r}_1 = (1-u)\left[(1-v)\mathbf{p}_{00} + v\mathbf{p}_{01}\right] + u\left[(1-v)\mathbf{p}_{10} + v\mathbf{p}_{11}\right]$$

$$\begin{aligned}\mathbf{s}(u,v) = \mathbf{x} &= \mathbf{p}_{00} + u(\mathbf{p}_{10} - \mathbf{p}_{00}) + v(\mathbf{p}_{01} - \mathbf{p}_{00}) + uv(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) \\ &= (1-u)(1-v)\mathbf{p}_{00} + u(1-v)\mathbf{p}_{10} + (1-u)v\mathbf{p}_{01} + uv\mathbf{p}_{11}\end{aligned}$$

$$= \begin{bmatrix} 1 & u \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ v \end{bmatrix}$$

Schulze

Outer/Tensor Product

Tensor product ($\otimes$) of two column vectors, $\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} =$$

$$\begin{bmatrix} a_1 \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} \\ a_2 \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} \\ a_3 \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 & a_1 b_4 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 & a_2 b_4 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 & a_3 b_4 \end{bmatrix}$$

Similarly, we can define a surface as the tensor (a.k.a. Cartesian) product of two curves …

Durand

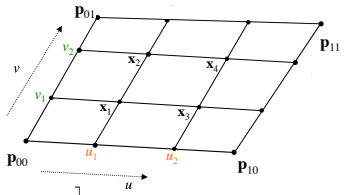
Bilinear Patch as Tensor Product

To compute $\mathbf{s}(u,v)$ for u_1, u_2 and v_1, v_2:

$$= \begin{bmatrix} \begin{bmatrix} 1 & u_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ v_1 \end{bmatrix} \begin{bmatrix} 1 \\ v_2 \end{bmatrix} \\ \begin{bmatrix} 1 & u_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ v_1 \end{bmatrix} \begin{bmatrix} 1 \\ v_2 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 1 & u_1 \end{bmatrix} \\ \begin{bmatrix} 1 & u_2 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & (\mathbf{p}_{01} - \mathbf{p}_{00}) \\ (\mathbf{p}_{10} - \mathbf{p}_{00}) & (\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 \\ v_1 \end{bmatrix} & \begin{bmatrix} 1 \\ v_2 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 1 & u_1 \end{bmatrix} \\ \begin{bmatrix} 1 & u_2 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} + v_1(\mathbf{p}_{01} - \mathbf{p}_{00}) & \mathbf{p}_{00} + v_2(\mathbf{p}_{01} - \mathbf{p}_{00}) \\ (\mathbf{p}_{10} - \mathbf{p}_{00}) + v_1(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) & (\mathbf{p}_{10} - \mathbf{p}_{00}) + v_2(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) \end{bmatrix}$$



Tensor product:

$$= \begin{bmatrix} \begin{bmatrix} 1 & u_1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} + v_1(\mathbf{p}_{01} - \mathbf{p}_{00}) & \mathbf{p}_{00} + v_2(\mathbf{p}_{01} - \mathbf{p}_{00}) \\ (\mathbf{p}_{10} - \mathbf{p}_{00}) + v_1(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) & (\mathbf{p}_{10} - \mathbf{p}_{00}) + v_2(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) \end{bmatrix} \\ \begin{bmatrix} 1 & u_2 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} + v_1(\mathbf{p}_{01} - \mathbf{p}_{00}) & \mathbf{p}_{00} + v_2(\mathbf{p}_{01} - \mathbf{p}_{00}) \\ (\mathbf{p}_{10} - \mathbf{p}_{00}) + v_1(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) & (\mathbf{p}_{10} - \mathbf{p}_{00}) + v_2(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{x}_1 = \mathbf{p}_{00} + u_1(\mathbf{p}_{10} - \mathbf{p}_{00}) + v_1(\mathbf{p}_{01} - \mathbf{p}_{00}) + u_1 v_1(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) & \mathbf{x}_2 = \mathbf{p}_{00} + u_1(\mathbf{p}_{10} - \mathbf{p}_{00}) + v_2(\mathbf{p}_{01} - \mathbf{p}_{00}) + u_1 v_2(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) \\ \mathbf{x}_3 = \mathbf{p}_{00} + u_2(\mathbf{p}_{10} - \mathbf{p}_{00}) + v_1(\mathbf{p}_{01} - \mathbf{p}_{00}) + u_2 v_1(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) & \mathbf{x}_4 = \mathbf{p}_{00} + u_2(\mathbf{p}_{10} - \mathbf{p}_{00}) + v_2(\mathbf{p}_{01} - \mathbf{p}_{00}) + u_2 v_2(\mathbf{p}_{00} - \mathbf{p}_{10} - \mathbf{p}_{01} + \mathbf{p}_{11}) \end{bmatrix}$$

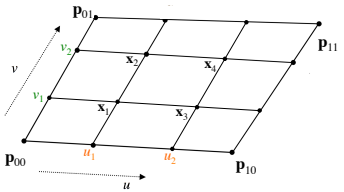
Bilinear Patch as Tensor Product

To compute $\mathbf{s}(u,v)$ for u_1, u_2 and v_1, v_2:

$$= \begin{bmatrix} \begin{bmatrix} 1 & u_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & u_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ v_1 \end{bmatrix} & \begin{bmatrix} 1 \\ v_2 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} (1-u_1) & u_1 \end{bmatrix} \\ \begin{bmatrix} (1-u_2) & u_2 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} (1-v_1) \\ v_1 \end{bmatrix} & \begin{bmatrix} (1-v_2) \\ v_2 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} b_0(u_1) & b_1(u_1) \end{bmatrix} \\ \begin{bmatrix} b_0(u_2) & b_1(u_2) \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} b_0(v_1) \\ b_1(v_1) \end{bmatrix} & \begin{bmatrix} b_0(v_2) \\ b_1(v_2) \end{bmatrix} \end{bmatrix}$$



$$\mathbf{s}(u,v) = \sum_{ij} b_{ij}(u,v)\mathbf{p}_{ij}, \text{ where } b_{ij}(u,v) = b_i(u)b_j(v)$$

Tensor product:

$$= \begin{bmatrix} \begin{bmatrix} b_0(u_1) & b_1(u_1) \end{bmatrix} \begin{bmatrix} b_0(v_1)\mathbf{p}_{00} + b_1(v_1)\mathbf{p}_{01} \\ b_0(v_1)\mathbf{p}_{10} + b_1(v_1)\mathbf{p}_{11} \end{bmatrix} \begin{bmatrix} b_0(v_2)\mathbf{p}_{00} + b_1(v_2)\mathbf{p}_{01} \\ b_0(v_2)\mathbf{p}_{10} + b_1(v_2)\mathbf{p}_{11} \end{bmatrix} \\ \begin{bmatrix} b_0(u_2) & b_1(u_2) \end{bmatrix} \begin{bmatrix} b_0(v_1)\mathbf{p}_{00} + b_1(v_1)\mathbf{p}_{01} \\ b_0(v_1)\mathbf{p}_{10} + b_1(v_1)\mathbf{p}_{11} \end{bmatrix} \begin{bmatrix} b_0(v_2)\mathbf{p}_{00} + b_1(v_2)\mathbf{p}_{01} \\ b_0(v_2)\mathbf{p}_{10} + b_1(v_2)\mathbf{p}_{11} \end{bmatrix} \end{bmatrix}$$

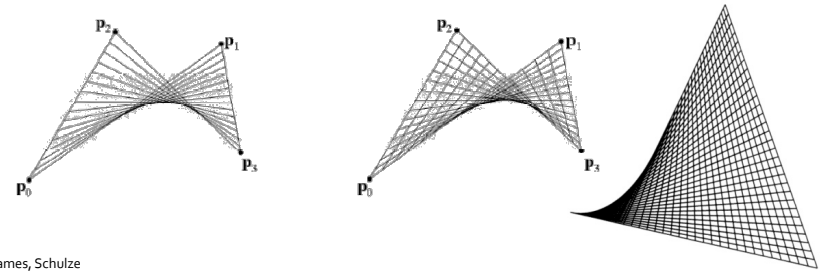
Note the ordering of the i and j indices on the control points to get the correct matching with the basis functions

$$\begin{matrix} \mathbf{x}_1 \\ \mathbf{x}_3 \end{matrix} \begin{bmatrix} b_0(u_1)b_0(v_1)\mathbf{p}_{00} + b_1(u_1)b_0(v_1)\mathbf{p}_{10} + b_0(u_1)b_1(v_1)\mathbf{p}_{01} + b_1(u_1)b_1(v_1)\mathbf{p}_{11} & b_0(u_1)b_0(v_2)\mathbf{p}_{00} + b_1(u_1)b_0(v_2)\mathbf{p}_{10} + b_0(u_1)b_1(v_2)\mathbf{p}_{01} + b_1(u_1)b_1(v_2)\mathbf{p}_{11} \\ b_0(u_2)b_0(v_1)\mathbf{p}_{00} + b_1(u_2)b_0(v_1)\mathbf{p}_{10} + b_0(u_2)b_1(v_1)\mathbf{p}_{01} + b_1(u_2)b_1(v_1)\mathbf{p}_{11} & b_0(u_2)b_0(v_2)\mathbf{p}_{00} + b_1(u_2)b_0(v_2)\mathbf{p}_{10} + b_0(u_2)b_1(v_2)\mathbf{p}_{01} + b_1(u_2)b_1(v_2)\mathbf{p}_{11} \end{bmatrix} \begin{matrix} \mathbf{x}_2 \\ \mathbf{x}_4 \end{matrix}$$

Bilinear Patch

Smooth version of quadrilateral with non-planar vertices

- gives a saddle-shape (hyperbolic paraboloid) curved surface
- the parametric curves are all straight line segments!
  - the surface is doubly ruled: 2 straight lines through every point
  - boundaries are straight line segments: no control of derivatives at the edges
- not a terribly useful modeling primitive for smooth surfaces

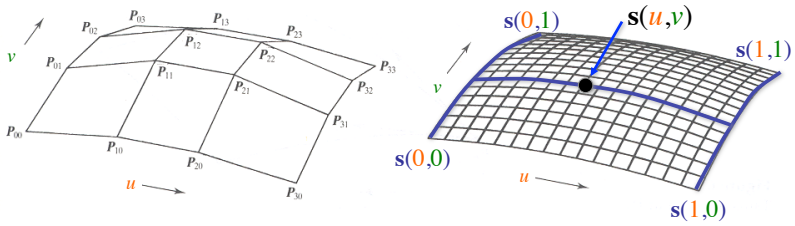


James, Schulze

Bicubic Patch

Bicubic patch $\mathbf{s}(u,v)$ is the tensor product of 2 cubic curves, with 16 unknowns

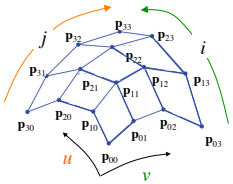
The control points of the 2 cubic curves form a control net with 16 control points, from which we can solve for the 16 unknowns



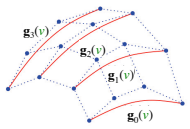
Funkhouser, Watt Fig. 6.21

Bicubic Patch

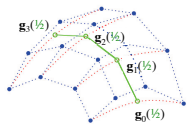
Given parametric curves $\mathbf{f}_j(u) = \mathbf{u}\mathbf{B}\mathbf{p}_{ij}$, $0 \le i \le 3$ and $\mathbf{g}_i(v) = \mathbf{v}\mathbf{B}\mathbf{p}_{ij}$, $0 \le j \le 3$, where $\mathbf{B}$ is the basis matrix (e.g., Bézier, B-splines, etc.)



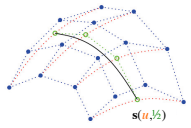
control net



control curves in v



control polygon at v=½
(not $\mathbf{f}_j(u)$, which is smooth)



surface at $\mathbf{s}(u,½)$
(again not $\mathbf{f}_j(u)$,
which is not shown)

To compute the tensor product of surface, transpose each i-th curve $(\mathbf{g}_i(v))^T = (\mathbf{v}\mathbf{B}\mathbf{p}_{ij})^T = \mathbf{p}_{ij}{}^T\mathbf{B}^T\mathbf{v}^T$

Bicubic Patch

The tensor-product surface in matrix form is:

$$\mathbf{s}(u,v) = \mathbf{u}\mathbf{B}\begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{02} & \mathbf{p}_{03} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13} \\ \mathbf{p}_{20} & \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23} \\ \mathbf{p}_{30} & \mathbf{p}_{31} & \mathbf{p}_{32} & \mathbf{p}_{33} \end{bmatrix}\mathbf{B}^T\mathbf{v}^T$$

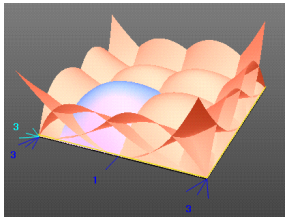
$$= \sum_{ij} b_{ij}(u,v)\mathbf{p}_{ij}, \text{ where } b_{ij}(u,v) = b_i(u)b_j(v),$$

are separable products of the 1D curve basis functions

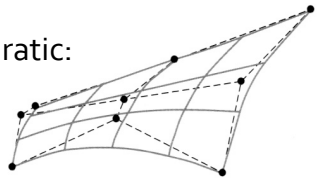
Note again that the ordering of the i and j indices of the control points must match those of the basis functions

Bézier Patches

Tensor product of 2 Bézier segments

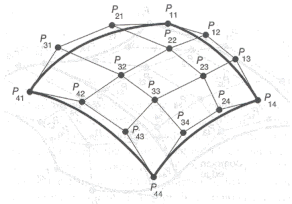


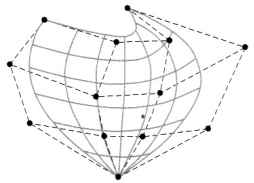
Biquadratic:

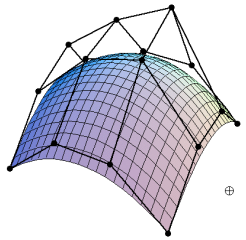


Basis functions of patch: product of 2 Bernstein polynomials:
$b_0(u)b_0(v), \ldots, b_3(u)b_3(v)$

Bicubic:







Durnad,Watt, FvD, Hearn&Baker

Bicubic Bézier Patches

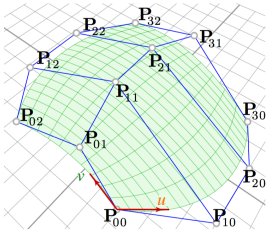
In matrix form:

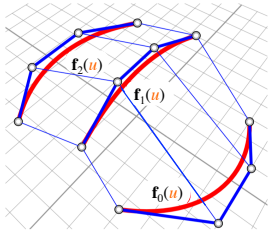
$$\mathbf{s}(u,v) = \mathbf{u}\mathbf{B}_Z \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{02} & \mathbf{p}_{03} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13} \\ \mathbf{p}_{20} & \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23} \\ \mathbf{p}_{30} & \mathbf{p}_{31} & \mathbf{p}_{32} & \mathbf{p}_{33} \end{bmatrix} \mathbf{B}_Z^T \mathbf{v}^T$$

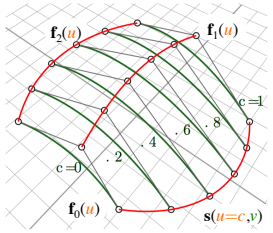
$$= \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{01} & \mathbf{p}_{03} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13} \\ \mathbf{p}_{20} & \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23} \\ \mathbf{p}_{30} & \mathbf{p}_{31} & \mathbf{p}_{32} & \mathbf{p}_{33} \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

Cheng

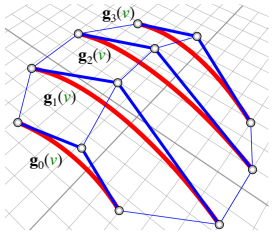
Order 3×4 Bézier Patches

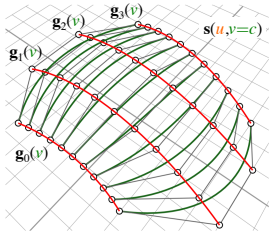






Tensor product of quadratic ($\mathbf{g}_i(v)$) and cubic ($\mathbf{f}_j(u)$) Béziers



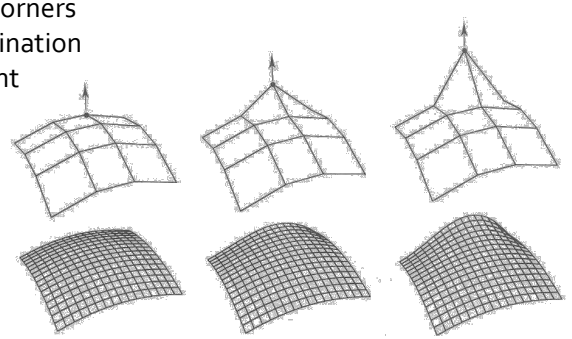


[Sederberg]

Bicubic Bézier Patches

Properties analogous to those of cubic Bézier curves

- interpolate four corner points
- tangency at corners
- convex combination
- affine invariant
- local control
- C^1

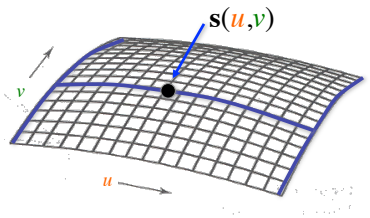


Only variation diminishing doesn't apply!

Durnad,Watt, FvD Fig. 11.43

Surface Normal

Tangents to the surface at any point can be computed from infinitesimally nearby points



Holding one parameter constant, we compute the partial derivative (tangent vector) in each direction

$$\mathbf{t}_u = \frac{\partial \mathbf{s}(u,v)}{\partial u} = \sum_{i=0}^{n}\sum_{j=0}^{n}\left[\frac{d}{du}b_i(u)\right]b_j(v)\mathbf{p}_{ij}$$

$$\mathbf{t}_v = \frac{\partial \mathbf{s}(u,v)}{\partial v} = \sum_{i=0}^{n}\sum_{j=0}^{n}b_i(u)\left[\frac{d}{dv}b_j(v)\right]\mathbf{p}_{ij}$$

Unit normal:

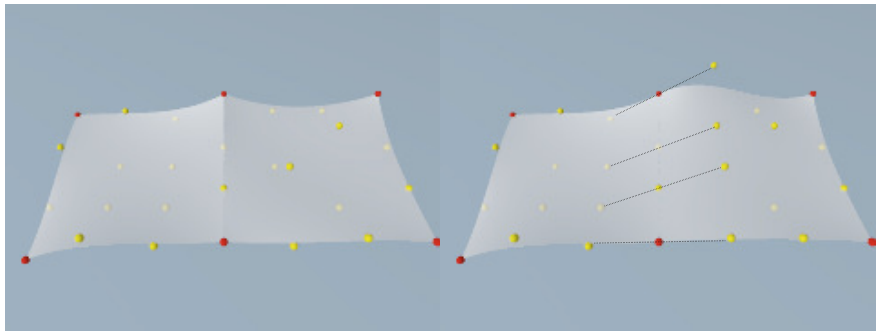
$$\mathbf{n}(u,v) = \frac{\mathbf{t}_u \times \mathbf{t}_v}{\|\mathbf{t}_u \times \mathbf{t}_v\|}$$

Yu,Curless,Chenney

Bicubic Bézier Patch Join

C^0: positional continuity
requires aligning boundary curves

C^1: tangential continuity
requires aligning boundary curves and derivatives



(and similarly for the other boundaries, in the v and u directions)

Watt oo

Bézier Curve/Surface Problems

To make a long continuous curve with Bézier segments requires using many segments

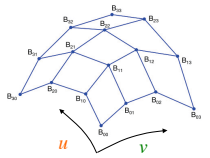
Maintaining continuity requires constraints on the control point positions

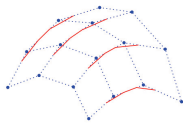
- the user cannot arbitrarily move control vertices and automatically maintain continuity
- the constraints must be explicitly maintained
- it is not intuitive to have control points that are not free

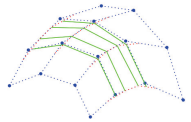
Consider: B-spline

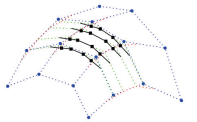
B-spline Patches

Generated as tensor product of B-spline curves



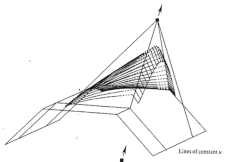


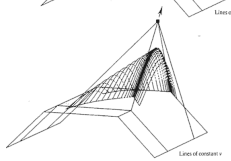




All properties of B-spline curves apply, except for the VD property

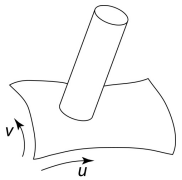
- C^{d-1} "for free" for B-splines of degree d
  - linear B-splines have C^0 continuity, cubic have C^2, etc.





[Culess,Watt00]

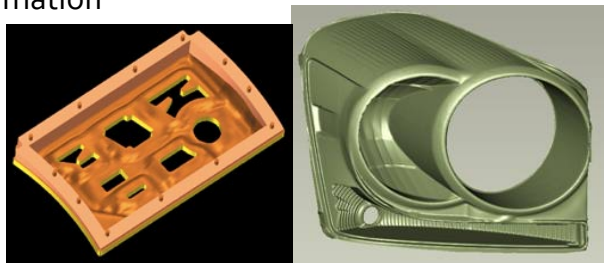
Trimmed NURBS Surfaces



Sometimes we want "holes" in the surface

We can do this by trimming the u-v domain
- define a closed curve (a trim curve) on the NURBS surface
- draw the surface everywhere except inside this curve

Continuity in these regions hard to maintain, especially in animation



Curless, Merrell

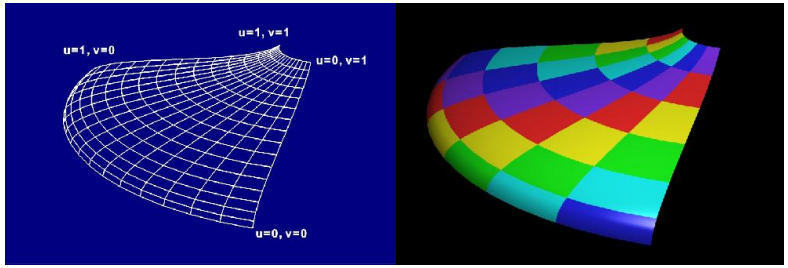
Parametric Surface: Advantages

Parametric curves and surfaces are generalization of piecewise linear/first degree polylines and polygons
- higher degree polynomials are piecewise smooth

These functions still only approximate the desired shapes, but advantages:
- easy to construct and manipulate from control points
- easy to enumerate points on surface
  - useful for texture mapping!
- much more compact than polygonal mesh (less storage)
- scalable geometric primitives

Texture Mapping

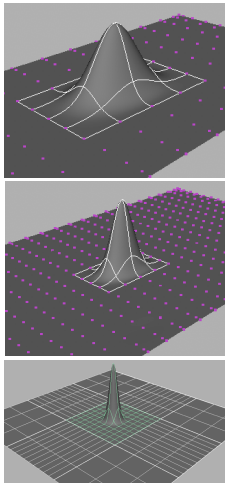
Mapping for parametric surfaces is easy: map surface parameters directly to texture coordinates: $u \rightarrow s$, $v \rightarrow t$



Wolfe97

Parametric Surface: Disadvantages

Disadvantages of approximate surfaces:
- control net must have specific topology
  - tensor product surfaces require control points to form a quad
  - for B-spline surfaces, control points must be distributed relatively "uniformly"
- must split surface into discrete patches
- works well mostly for smooth surfaces



Other disadvantages:
- intersection test? inside/outside test?
- higher rendering times than polygonal mesh:
  - patch must be tessellated into triangles
  - may need adaptive subdivision

[Funkhouser, Hodgins, Durand, James]

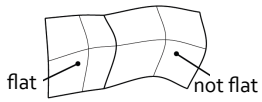
Adaptive Subdivision/Tessellation

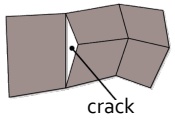
When rendered as polygonal mesh:

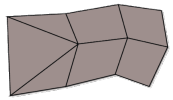
- test patch for flatness
- if not flat enough, subdivide patch along each of the two dimensions into four parts
- recurse
  - as with curves, use convex hull property for termination testing (see 3 slides forward)

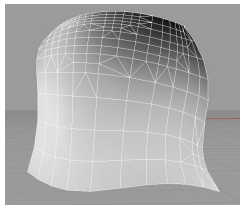
Cracks (discontinuity) may form between patches of different degrees of tessellation

- solution: split shared edge at both faces, no T-junction









[Funkhouser, Hodgins, Durand, James]

Smooth Surfaces

How do you get smooth surfaces?

Tensor product surfaces: piecewise parametric patches

- bicubic Bézier patches: manual C^1 continuity
- bicubic B-splines, NURBS: built-in C^2 continuity

General, mathematically elegant, but problematic

- control mesh dictates surface feature size
- control mesh must be rectangular
  - refinement constant across u and v directions
  - same level of detail throughout surface: non-adaptive
  - hard to make creases and sharp edges
- surface must be tessellated to be rendered: cracking issue

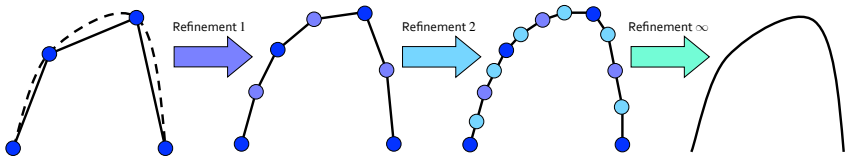
Gleicher

Subdivision

How do you render a smooth curve?
Approximate the curve as a series of line segments

Subdivision scheme: a process of recursively subdividing the polylines into smaller pieces (finer resolution)

- the resulting limit curve will be piecewise smooth



Hart/Carr

Bézier Curve Subdivision

de Casteljau algorithm easily and adaptively subdivide a Bézier curve into smaller segments

- control polygon has sharp corners (non-smooth)
- smooth out this curve by "cutting off" the sharp corners
- using the new control point(s), split the curve into two
- continue to smooth out and subdivide until the limit curve

Fussell, Shirley, TP3

Bézier Curve Subdivision

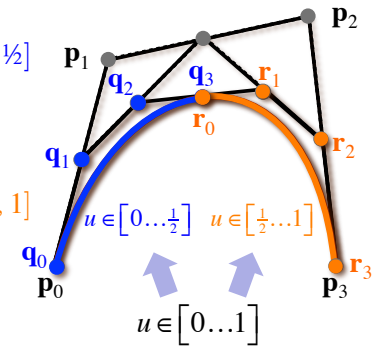
Given a Bézier curve with $n+1$ control points: $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ on $u \in [0, 1]$, use de Casteljau's algorithm to compute two sets of $n+1$ control points $\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$ and $\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n$ such that:

- $\mathbf{q}_i$'s form the control polygon of the original Bézier curve on $u \in [0, ½]$

$$\mathbf{P}_1^T = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \end{bmatrix}$$

- $\mathbf{r}_i$'s form the control polygon of the original Bézier curve on $u \in [½, 1]$

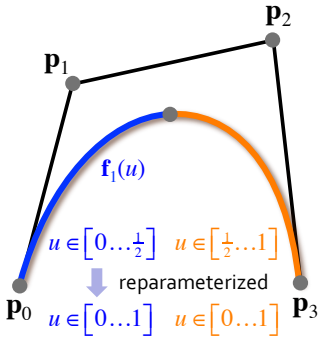
$$\mathbf{P}_2^T = \begin{bmatrix} \mathbf{r}_0 & \mathbf{r}_1 & \mathbf{r}_2 & \cdots & \mathbf{r}_n \end{bmatrix}$$



Shirley, Manocha

Reparameterized

For $u \in [0, 1]$, Bézier curve's canonical form is:



$$\mathbf{f}(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{P}$$

For $u \in [0, ½]$ and $u \in [½, 1]$, want to retain canonical form:

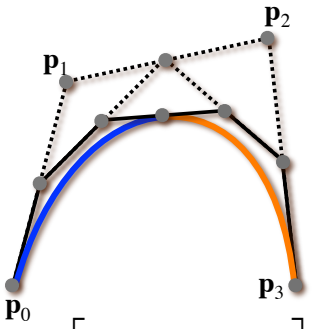
$$\mathbf{f}_1(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{P}$$

with u reparameterized to $\in [0, 1]$

Both $\mathbf{u}$ and $\mathbf{B}$ can't be changed, so much change $\mathbf{P}$, to what?

[O'Brien]

Subdivision Reparameterized



For $u \in [0 \ldots ½]$:

$$\mathbf{f}_1(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{P}$$

reparameterized to $u \in [0 \ldots 1]$:

$$\mathbf{f}_1(u) = \begin{bmatrix} 1 & \frac{1}{2}u & \frac{1}{4}u^2 & \frac{1}{8}u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{P}$$

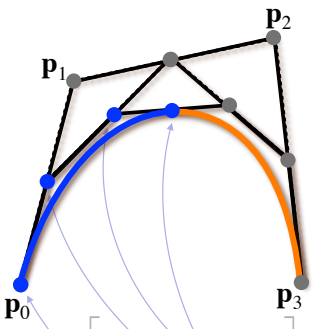
$$\mathbf{f}_1(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{S}_1 \mathbf{B}_Z \mathbf{P}$$

$$\mathbf{S}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix}$$

reparameterization matrix

[O'Brien]

New Control Points



$$\mathbf{B}_Z^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{3} & 0 & 0 \\ 1 & \frac{2}{3} & \frac{1}{3} & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{f}_1(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{S}_1 \mathbf{B}_Z \mathbf{P}$$

$$\mathbf{f}_1(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{B}_Z^{-1} \mathbf{S}_1 \mathbf{B}_Z \mathbf{P}$$

$$\mathbf{f}_1(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{H}_{Z1} \mathbf{P}$$

$$\mathbf{H}_{Z1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{bmatrix}$$

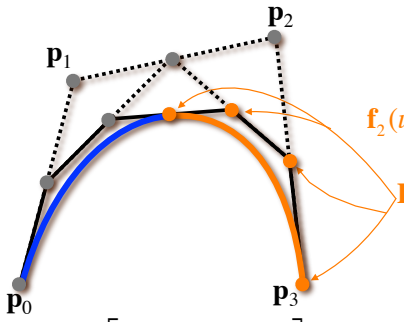
a.k.a. the splitting matrix

$$\mathbf{f}_1(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{P}_1$$

$$\mathbf{P}_1 = \mathbf{H}_{Z1} \mathbf{P} = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \end{bmatrix}^T$$

[O'Brien]

Subdivision → New Control Points



$$\mathbf{f}_2(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{P}_2$$

$$\mathbf{P}_2 = \mathbf{H}_{Z2}\mathbf{P} = \begin{bmatrix} \mathbf{r}_0 & \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix}^T$$

$$\mathbf{S}_2 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} \\ 0 & \frac{1}{2} & \frac{1}{2} & \frac{3}{8} \\ 0 & 0 & \frac{1}{4} & \frac{3}{8} \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix} \qquad \mathbf{H}_{Z2} = \begin{bmatrix} \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[O'Brien]

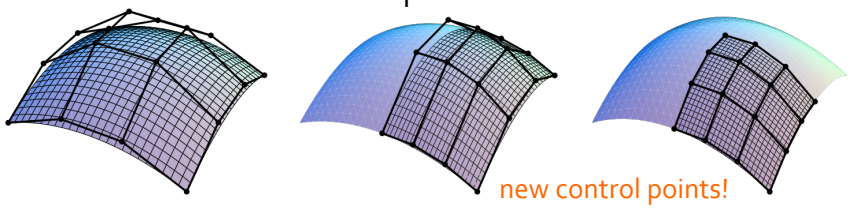
Bézier Patch Subdivision

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{02} & \mathbf{p}_{03} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13} \\ \mathbf{p}_{20} & \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23} \\ \mathbf{p}_{30} & \mathbf{p}_{31} & \mathbf{p}_{23} & \mathbf{p}_{33} \end{bmatrix}$$

Bézier patch:

$$\mathbf{s}(u,v) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \mathbf{B}_Z \mathbf{P} \mathbf{B}_Z^T \begin{bmatrix} 1 & v & v^2 & v^3 \end{bmatrix}^T$$

Bézier patch subdivisions:



new control points!

$$\mathbf{P}_{2\cdot} = \mathbf{H}_{Z2}\mathbf{P} \qquad \mathbf{P}_{21} = \mathbf{H}_{Z2}\mathbf{P}\mathbf{H}_{Z1}^T$$

[O'Brien]

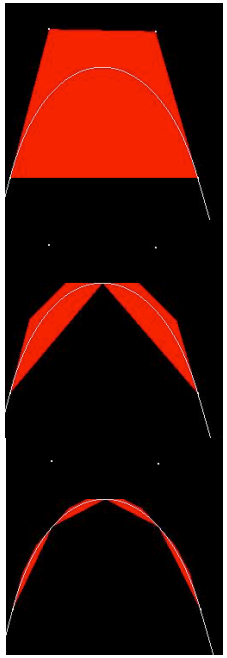
Subdividing Bézier Curves

Subdivision doesn't change the shape of a Bézier curve

- asymptotically, the control polygons of the subdivided curve converge to the actual curve (at a quadratic rate)

Other uses of subdivision:

- collision/intersection detection
  - the union of convex hulls of the subdivided curve is a subset of the convex hull of the original curve
- recursive search
- good for curve editing and approximation:
  - local refinement: change the control point(s) of one of the subdivided curve



Manocha, Durand

Rendering Bézier Curves

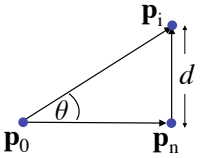
How do you rasterize a Bézier curve?

- generally no low-level support for drawing curves
- can only draw line segments or individual pixels

de Casteljau algorithm easily and adaptively subdivide a Bézier curve into smaller segments

- continue to subdivide until the new control points are close to being collinear, then approximate with a straight line
- collinearity test: distance of control point $\mathbf{p}_i$ from the line through $\mathbf{p}_0$ and $\mathbf{p}_n$:

$$d = \frac{\left|(\mathbf{p}_n - \mathbf{p}_0) \times (\mathbf{p}_i - \mathbf{p}_0)\right|}{\left|\mathbf{p}_n - \mathbf{p}_0\right|} = \left|\mathbf{p}_i - \mathbf{p}_0\right| \sin\theta$$



Fussell, Shirley, TP3

Converting Spline Representations

All the cubic splines are equivalent (span the same space):

$$\mathbf{f}(u) = \mathbf{ua} = \mathbf{uBp}$$

where:

- $\mathbf{u}$ is the canonical basis set,
- $\mathbf{B}$ the basis matrix, and
- $\mathbf{p}$ the control points

Other curve representations can be converted to Bézier for rendering!

To use a different spline is to change basis matrix:

$$\begin{aligned}\mathbf{f}(u) = \mathbf{ua} = \mathbf{uBp} &= \mathbf{u}\,(\mathbf{B}_{alt}\,\mathbf{B}_{alt}^{-1})\,\mathbf{B}\,\mathbf{p} \\ &= \mathbf{u}\,\mathbf{B}_{alt}\,(\mathbf{B}_{alt}^{-1}\,\mathbf{B}\,\mathbf{p}) = \mathbf{u}\,\mathbf{B}_{alt}\,\mathbf{p}_{alt}\end{aligned}$$

we transform the control points from one type to the other: $\mathbf{p}_{alt} = \mathbf{B}_{alt}^{-1}\,\mathbf{B}\,\mathbf{p}$

Marschner

Rendering Curves

Other curve representations can be converted to Bézier for rendering

Catmull-Rom to Bézier:

$$\begin{bmatrix}\mathbf{p}_{0,Z}\\ \mathbf{p}_{1,Z}\\ \mathbf{p}_{2,Z}\\ \mathbf{p}_{3,Z}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ -\frac{1}{6} & 1 & \frac{1}{6} & 0\\ 0 & \frac{1}{6} & 1 & -\frac{1}{6}\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\mathbf{p}_{0,CR}\\ \mathbf{p}_{1,CR}\\ \mathbf{p}_{2,CR}\\ \mathbf{p}_{3,CR}\end{bmatrix}$$

Catmull-Rom to Hermite:

$$\begin{aligned}\mathbf{p}_{0,H} &= \mathbf{p}_{1,CR}\\ \mathbf{p}_{1,H} &= \tfrac{1}{2}\,(\mathbf{p}_{2,CR} - \mathbf{p}_{0,CR})\\ \mathbf{p}_{2,H} &= \mathbf{p}_{2,CR}\\ \mathbf{p}_{3,H} &= \tfrac{1}{2}(\mathbf{p}_{3,CR} - \mathbf{p}_{1,CR})\end{aligned}$$

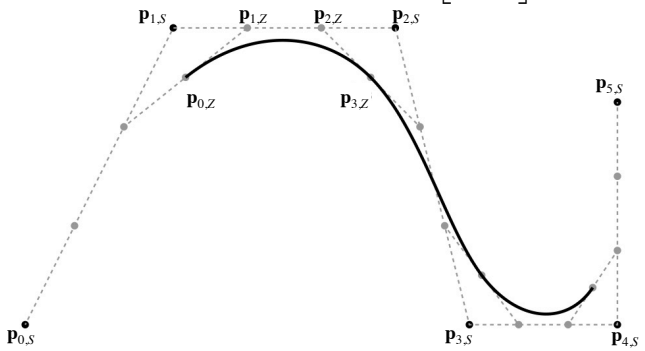
Bézier to Hermite:

$$\begin{aligned}\mathbf{p}_{0,H} &= \mathbf{p}_{0,Z}\\ \mathbf{p}_{1,H} &= 3(\mathbf{p}_{1,Z} - \mathbf{p}_{0,Z})\\ \mathbf{p}_{2,H} &= \mathbf{p}_{3,Z}\\ \mathbf{p}_{3,H} &= 3(\mathbf{p}_{3,Z} - \mathbf{p}_{2,Z})\end{aligned}$$

Hanrahan

Uniform B-splines to Bézier

$$\begin{bmatrix}\mathbf{p}_{0,Z}\\ \mathbf{p}_{1,Z}\\ \mathbf{p}_{2,Z}\\ \mathbf{p}_{3,Z}\end{bmatrix} = \begin{bmatrix}\frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0\\ 0 & \frac{2}{3} & \frac{1}{3} & 0\\ 0 & \frac{1}{3} & \frac{2}{3} & 0\\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6}\end{bmatrix}\begin{bmatrix}\mathbf{p}_{0,S}\\ \mathbf{p}_{1,S}\\ \mathbf{p}_{2,S}\\ \mathbf{p}_{3,S}\end{bmatrix}$$



Curless