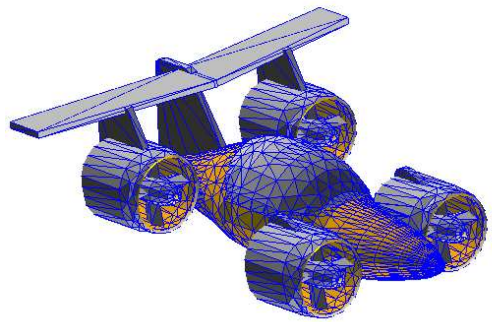
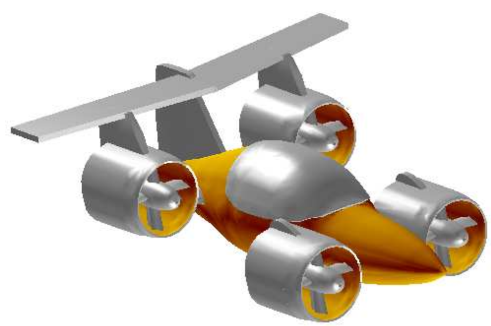
BB-splines, A-Splines and B-Splines





Spline Curves

- Successive linear blend
- Basis polynomials
- Recursive evaluation
- Properties
- Joining segments

Tensor-product-patch Spline Surfaces

- Tensor product patches
- Recursive Evaluation
- Properties
- Joining patches

Triangular-patch Spline Surfaces

- Barycentric Coordinates and Basis

- Recursive Evaluation
- Properties
- Joining patches

Bernstein-Bézier (BB) Polynomials):

- *The control points appear as coefficients of* Bernstein-Bézier polynomials

$$
\begin{aligned}
P_0^0(t) &= P_0 1 \\
P_0^1(t) &= (1-t)P_0 + tP_1 \\
P_0^2(t) &= (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2 \\
P_0^3(t) &= (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3 \\
P_0^n(t) &= \sum_{i=0}^{n} P_i B_i^n(t)
\end{aligned}
$$

where $B_i^n(t) = \frac{n!}{(n-i)!i!}(1-t)^{n-i}t^i = \left(\begin{array}{c} n \\ i \end{array} \right) (1-t)^{n-i}t^i$

- *The BB polynomials of degree* n *form a basis for the space of all degree-*n *polynomials*

Bernstein-Bézier Basis Functions

Bernstein-Bézier Polynomial Properties:

Partition of Unity: $\sum_{i=0}^{n} B_i^n(t) = 1$

Recurrence: $B_0^0(t) = 1$ *and* $B_i^n(t) = (1-t)B_i^{n-1}(t) + B_{i-1}^{n-1}(t)$

Derivatives: $\frac{d}{dt} B_i^n(t) = n \left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t) \right)$

Bernstein -Bézier Splines: Implicit and Parametric

Parametric Bernstein-Bézier Curve Segments and their Properties

Definition:

- *A degree n (order $n+1$)* Bézier curve segment *is*

$$P(t) = \sum_{i=0}^{n} P_i B_i^n(t)$$

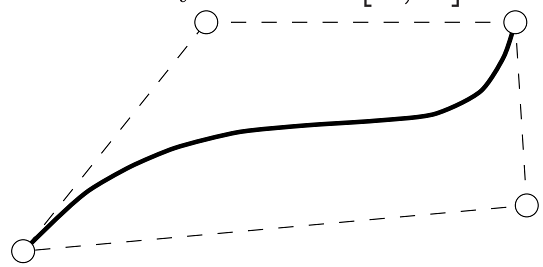
where the P_i are k-dimensional control points.

Convex Hull:

$\sum_{i=0}^{n} B_i^n(t) = 1$, $B_i^n(t) \geq 0$ *for* $t \in [0,1]$

$\Longrightarrow P(t)$ *is a convex combination of the* P_i *for* $t \in [0,1]$

$\Longrightarrow P(t)$ *lies within convex hull of* P_i *for* $t \in [0, 1]$



Affine Invariance:

- *A Bézier curve is an affine combination of its control points*
- *Any affine transformation of a curve is the curve of the transformed control points*

$$T\left(\sum_{i=0}^{n} P_i B_i^n(t)\right) = \sum_{i=0}^{n} T(P_i) B_i^n(t)$$

- This property does not hold for projective transformations!

Interpolation of End Control Points:

$B_0^n(0) = 1, B_n^n(1) = 1, \sum i = 0^n B_i^n(t) = 1, B_i^n(t) \geq 0$ *for* $t \in [0, 1]$

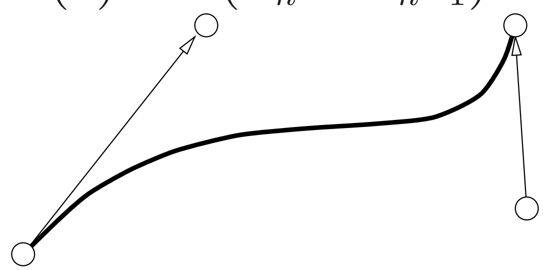
$$\Longrightarrow B_i^n(0) = 0 \textit{ if } i \neq 0, B_i^n(1) = 0 \textit{ if } i \neq n$$

$$\Longrightarrow P(0) = P_0, P(1) = P_n$$

Derivatives:

$$\frac{d}{dt} B_i^n(t) = n\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right)$$

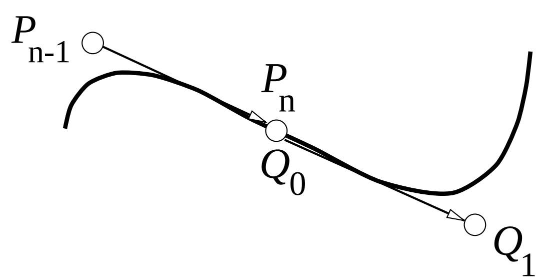
$$\Longrightarrow P'(0) = n(P_1 - P_0), P'(1) = n(P_n - P_{n-1})$$



Smoothly Joined Segments (G^1):

- *Let P_{n-1}, P_n be the last two control points of one segment*
- *Let Q_0, Q_1 be the first two control points of the next segment*

$$\begin{aligned} P_n &= Q_0 \\ (P_n - P_{n-1}) &= \beta(Q_1 - Q_0) \text{ for some } \beta > 0 \end{aligned}$$



Recurrence, Subdivision:

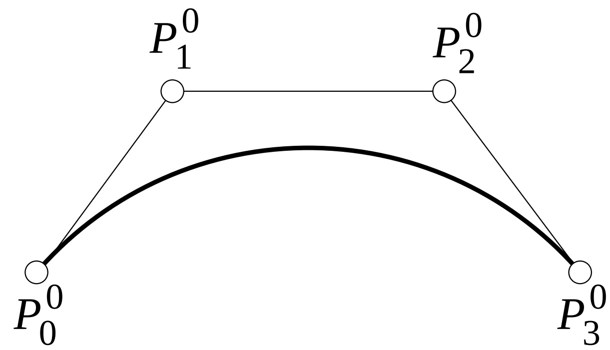
$$B_i^n(t) = (1-t)B_i^{n-1} + tB_{i-1}^{n-1}(t)$$

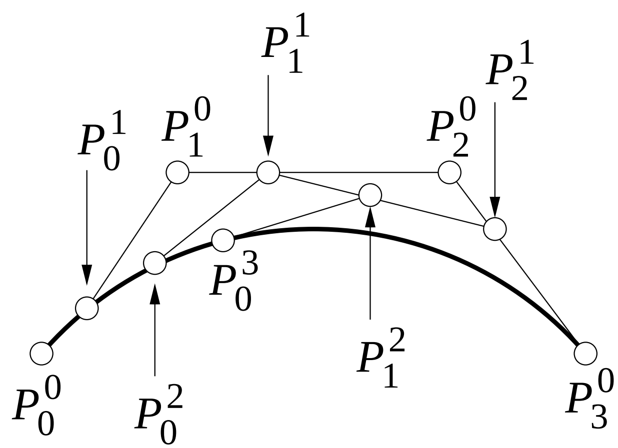
$\Longrightarrow$ *deCasteljau's algorithm:*

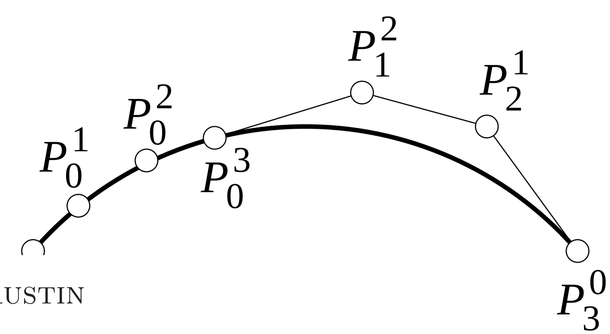
$$\begin{aligned} P(t) &= P_o^n(t) \\ P_i^k(t) &= (1-t)P_i^{k-1}(t) + t)P_{i+1}^{k-1} \\ P_i^0 &= P_i \end{aligned}$$

Use to evaluate point at t, or subdivide into two new curves:

- $P_0^0, P_0^1, \ldots P_0^n$ *are the control points for the left half*
- $P_n^0, P_{n-1}^1, \ldots P_0^n$ *are the control points for the right half*


P_1^0
P_2^0
P_0^0
P_3^0


P_1^1
P_2^1
P_0^1
P_1^0
P_2^0
P_0^3
P_1^2
P_0^0
P_0^2
P_3^0


P_1^2
P_2^1
P_0^2
P_0^1
P_0^3
P_3^0

Tensor Product Patches

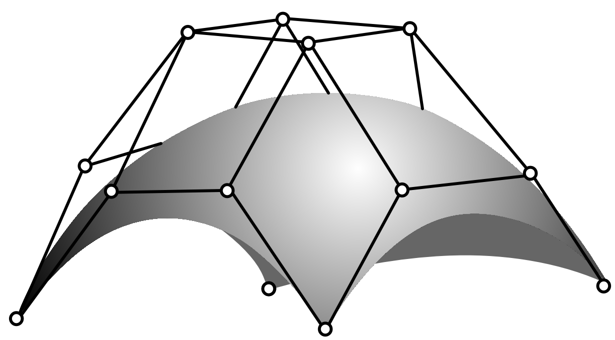
Tensor Product Patches:

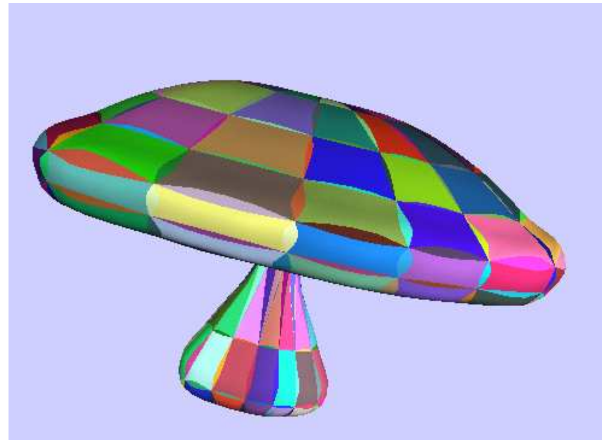
- *The* control polygon *is the polygonal mesh with vertices* $P_{i,j}$
- *The*patch basis functions *are products of curve basis functions*

$$P(s,t) = \sum_{i=0}^{n} \sum_{j=0}^{n} P_{i,j} B^n_{i,j}(s,t)$$

where

$$B^n_{i,j}(s,t) = B^n_i(s) B^n_j(t)$$





Properties:

- *Patch basis functions* sum to one

$$\sum_{i=0}^{n}\sum_{j=0}^{n} B_i^n(s)B_j^n(t) = 1$$

- *Patch basis functions are* nonnegative *on [0, 1] × [0, 1]*

$$B_i^n(s)B_j^n(t) \geq 0 \text{ for } 0 \leq s, t \leq 1$$

$\Longrightarrow$ *Surface patch is in the* convex hull *of the control points*
$\Longrightarrow$ *Surface patch is* affinely invariant
(Transform the patch by transforming the control points)

Subdivision, Recursion, Evaluation:

- *As for curves in each variable separately and independently*
- Tangent plane is not produced!
  - *Normals must be computed from partial derivatives*

Partial Derivatives:

- *Ordinary derivative in each variable separately':*

$$
\begin{aligned}
\frac{\partial}{\partial s} P(s,t) &= \sum_{i=0}^{n} \sum_{j=0}^{n} P_{i,j} \left[\frac{d}{ds} B_i^n(s) \right] B_j^n(t) \\
\frac{\partial}{\partial s} P(s,t) &= \sum_{i=0}^{n} \sum_{j=0}^{n} P_{i,j} B_i^n(s) \left[\frac{d}{dt} B_j^n(t) \right]
\end{aligned}
$$

- *Each of the above is a* tangent vector *in a parametric direction*
- *Surface is* regular *at each* (s,t) *where these two vectors are linearly independent*
- *The (unnormalized)* surface normal *is given at any regular point by*

$$
\pm \left[\frac{\partial}{\partial s} P(s,t) \times \frac{\partial}{\partial t} P(s,t) \right]
$$

(the sign dictates what is the outward pointing normal*)*

- *In particular, the* cross-boundary tangent *is given by (e.g., for the $s = 0$ boundary):*

$$n \sum_{i=0}^{n} \sum_{j=0}^{n} (P_{1,j} - P_{0,j}) B_j^n(t)$$

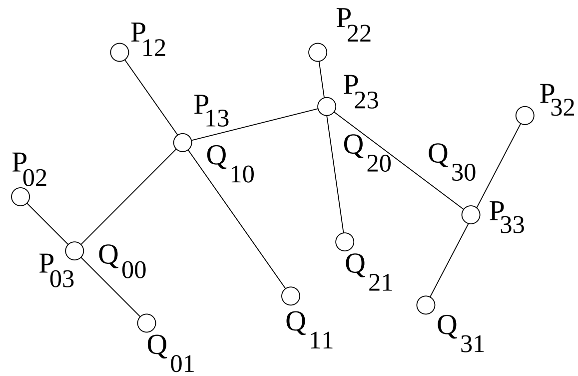
(and similarly for the other boundaries)

Smoothly Joined Patches:

- *Can be achieved by ensuring that*

$$(P_{i,n} - P_{i,n-1}) = \beta(Q_{i,1} - Qi, 0) \text{ for } \beta > 0$$

(and correspondingly for other boundaries)


P12
P22
P23
P13
P32
Q10
Q20
Q30
P02
P33
Q00
P03
Q21
Q11
Q31
Q01

Rendering:

- *Divide up into polygons:*
  1. *By stepping*

$$\begin{aligned} s &= 0, \delta, 2\delta, \ldots, 1 \\ t &= 1, \gamma, 2\gamma, \ldots, 1 \end{aligned}$$

     and joining up sides and diagonals to produce a triangular mesh
  2. *By subdividing and rendering the control polygon*

Triangular Patches

deCasteljau Revisited Barycentrically:

- *Linear blend expressed in barycentric coordinates*

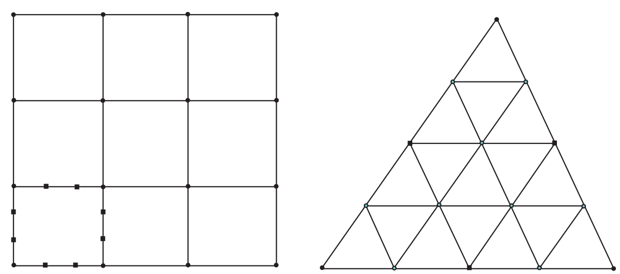
$$(1-t)P_0 + tP_1 = rP_0 + tP_1 \text{ where } r + t = 1$$

- *Higher powers and a symmetric form of the Bernstein polynomials:*

$$P(t) = \sum_{i=0}^{n} P_i \left(\frac{n!}{i!(n-i)!}\right)(1-t)^{n-i}t^i$$

$$= \sum_{\substack{i+j=n \\ i\geq 0, j\geq 0}} P_i \left(\frac{n!}{i!j!}\right) t^i r^j \text{ where } r+t=1$$

$$\Longrightarrow \sum_{\substack{i+j=n \\ i\geq 0, j\geq 0}} P_{ij} B^n_{ij}(r,t)$$



- *Examples*

$$
\begin{aligned}
\{B^0_{00}(r,t)\} &= \{1\} \\
\{B^1_{01}(r,t), B^1_{10}(r,t)\} &= \{r, t\} \\
\{B^2_{02}(r,t), B^2_{11}(r,t), B^2_{20}(r,t)\} &= \{r^2, 2rt, t^2\} \\
\{B^3_{03}(r,t), B^3_{12}(r,t), B^3_{21}(r,t), B^3_{30}(r,t)\} &= \{r^3, 3r^2t, 3rt^2, t^3\}
\end{aligned}
$$

Surfaces – Barycentric Blends on Triangles:

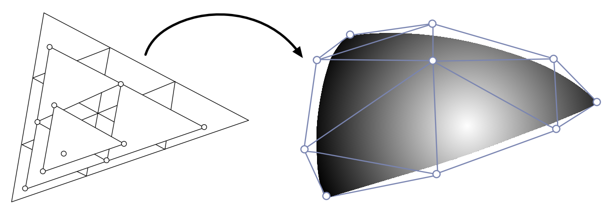
- *Formulas*

$$P(r,s,t) = \sum_{\substack{i+j+k=n \\ i \geq 0, j \geq 0, k \geq 0}} P_{ijk} B^n_{ijk}(r,s,t)$$

$$B^n_{ijk}(r,s,t) = \frac{n!}{i!j!k!} r^i s^j t^k$$

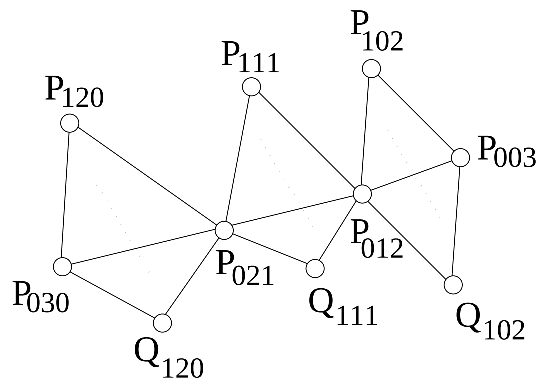
Triangular deCasteljau:

- *Join adjacently indexed P_{ijk} by triangles*
- *Find $r : s : t$ barycentric point in each triangle*
- *Join adjacent points by triangles*
- *Repeat*
  - *Final point is the surface point $P(r, s, t)$*
  - *final triangle is tangent to the surface at $P(r, s, t)$*
- *Triangle up/down schemes become tretrahedral up/down schemes*



Properties:

- *Each boundary curve is a Bézier curve*
- *Patches will be joined smoothly if pairs of boundary triangles are planar as shown*



Splines

If give up on small support, get natural splines; every control point influences the whole curve.

If give up on interpolation, get cubic B-splines.

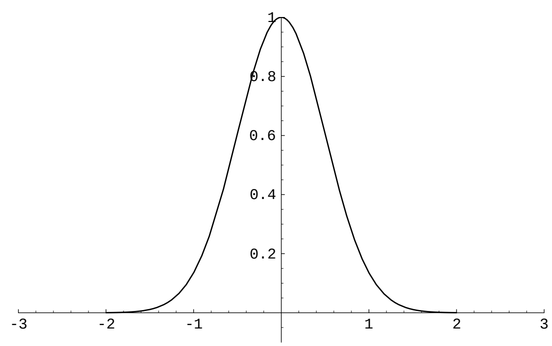


Figure 1: Cubic B-spline basis function

B-Splines

Need only one basis function, all $B_i(t)$ are obtained by shifts: $B_i(t) = B(t - i)$.
The basis function is piecewise polynomial:

$$B(t) = \begin{cases} 0 & t \leq -2 \\ \frac{1}{6}t^3 + 2t + \frac{4}{3} + t^2 & t \leq -1 \\ \frac{2}{3} - t^2 - \frac{1}{2}t^3 & t \leq 0 \\ \frac{2}{3} - t^2 + \frac{1}{2}t^3 & t \leq 1 \\ \frac{4}{3} - 2t + t^2 - \frac{1}{6}t^3 & t \leq 2 \\ 0 & 2 \leq t \end{cases}$$

B-Splines

The curve with control points $p_0, p_1, p_2, \ldots p_n$ is computed using

$$p(t) = \sum_{i=0}^{n} p_i \; B(t-i)$$

The allowed range of t is from 1 to $n-1$; outside this interval our functions do not sum up to 1, which means in particular that if we move control points together in the same way, the curve outside the interval will not move rigidly.

B-Splines

The minimal number of points required is 4;
this corresponds to the interval for t of length 1.

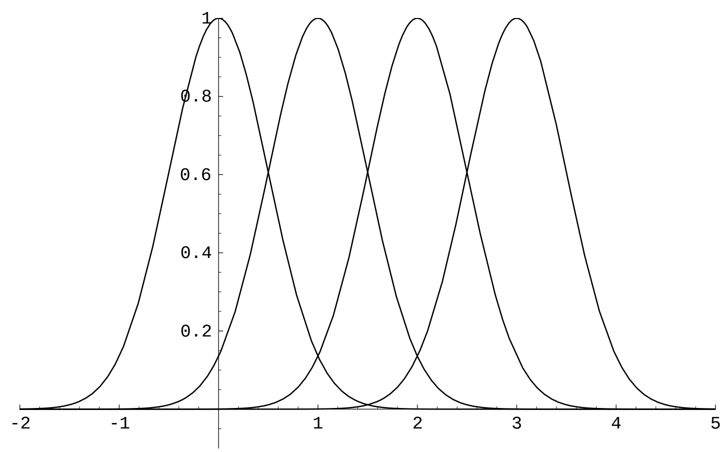
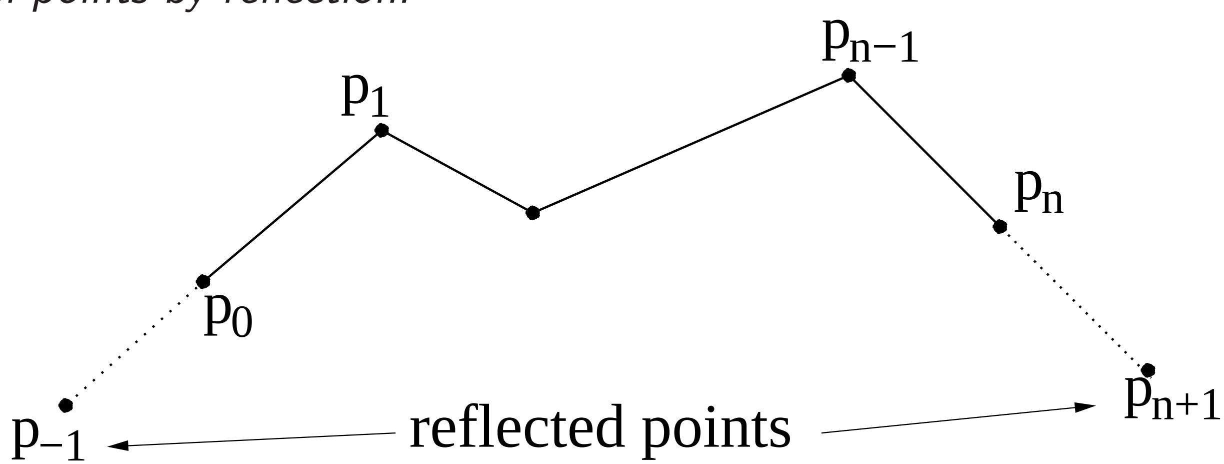


Figure 2: Cubic B-spline basis function

This is inconvenient - but we can always add control points by reflection.

B-Splines

Adding control points by reflection:



$$p_{-1} = 2p_0 - p_1; \qquad p_{n+1} = 2p_n - p_{n-1}$$

Drawing B-Splines

Hardware typically can draw only line segments. Need to approximate B-spline with piecewise linear curve. Simplest approach:

Choose small Δt.
Compute points $p(0), p(\Delta t), p(2\Delta t), \cdots$.
Draw line segments connecting the points.

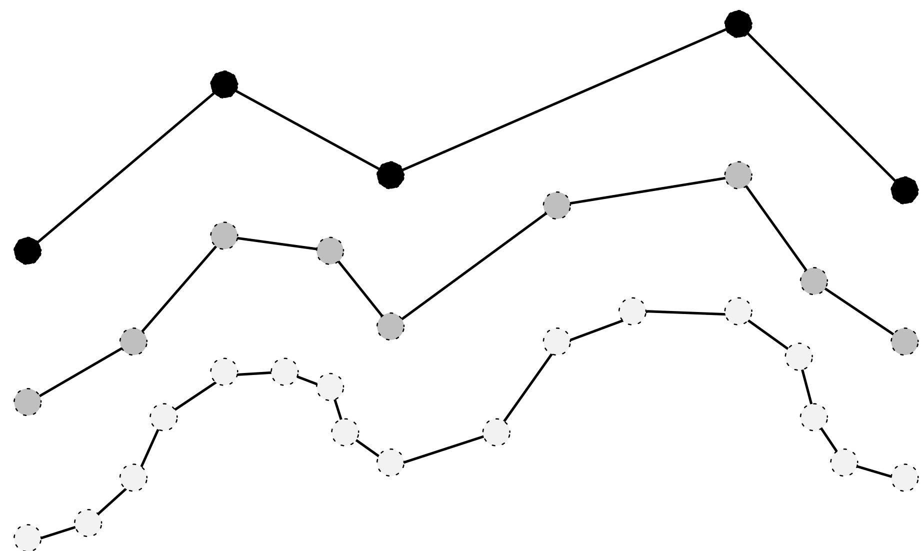
Not very efficient — have to evaluate a cubic polynomial (or several) at each point.

Can do better using a magic algorithm (subdivision).

B-Splines via subdivision

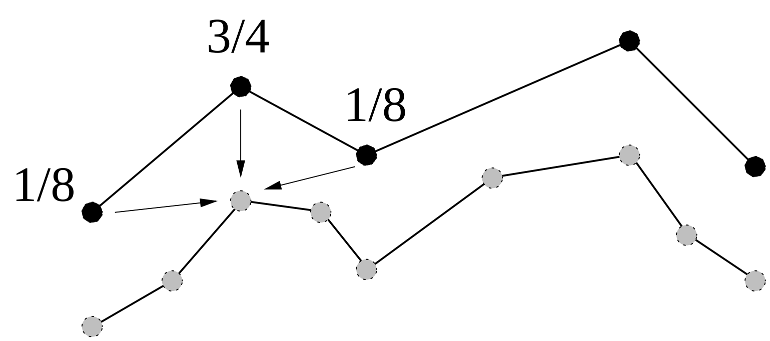
It turns out that the smooth curve be obtained by subdivision of the original polyline:

Subdivision adds new control points between the original control points and updates positions of original control points.

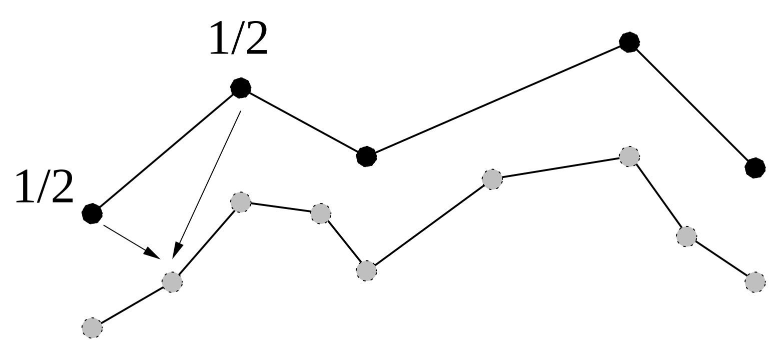


Subdivision

Subdivision rules for updating old points:



Subdivion rules for inserting new points:



Subdivision rules

Even rule (in the new sequence of points the points with even numbers are the old points with updated positions).

$$p_{2i}^{j+1} = \frac{1}{8}(p_{i-1}^{j} + 6p_{i}^{j} + p_{i+1}^{j})$$

Odd rule (in the new sequence the points with odd numbers are newly inserted points).

$$p_{2i+1}^{j+1} = \frac{1}{8}(4p_{i}^{j} + 4p_{i+1}^{j})$$

Subdivision

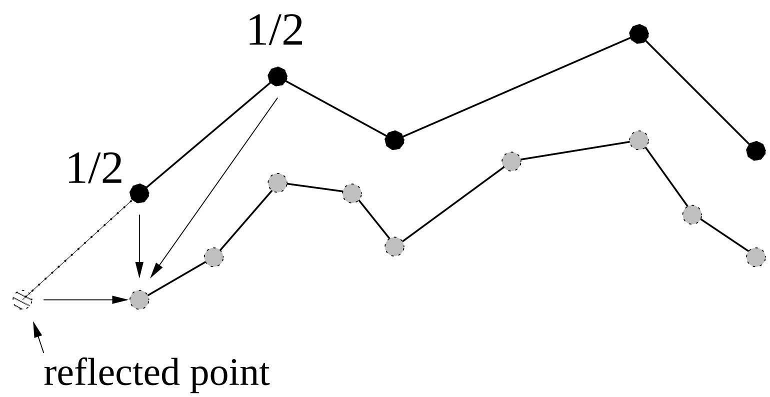
Of course, in a finite number of steps subdivision generates only polylines. But they get arbitrarily close to a limit C^2 and this curve is exactly a cubic B-spline.

Algorithm:

Start with an array of control points of length $n + 1$.
Compute from the original points new array of length $2n + 1$ using subdivision rules for even and odd points.
Then from the new array compute an array of length $4n + 1$ etc., (typically 4-5 steps is enough).
Then draw the line segments connecting sequential control points.

Endpoints

What do we do when a point is missing?



In this case, apply reflection (recall adding missing control points). This results in the following trivial rule:

$$p_0^{j+1} = \frac{1}{8}(p_1^j + 6p_0^j + (2p_0^j - p_1^j)) = p_0^j$$

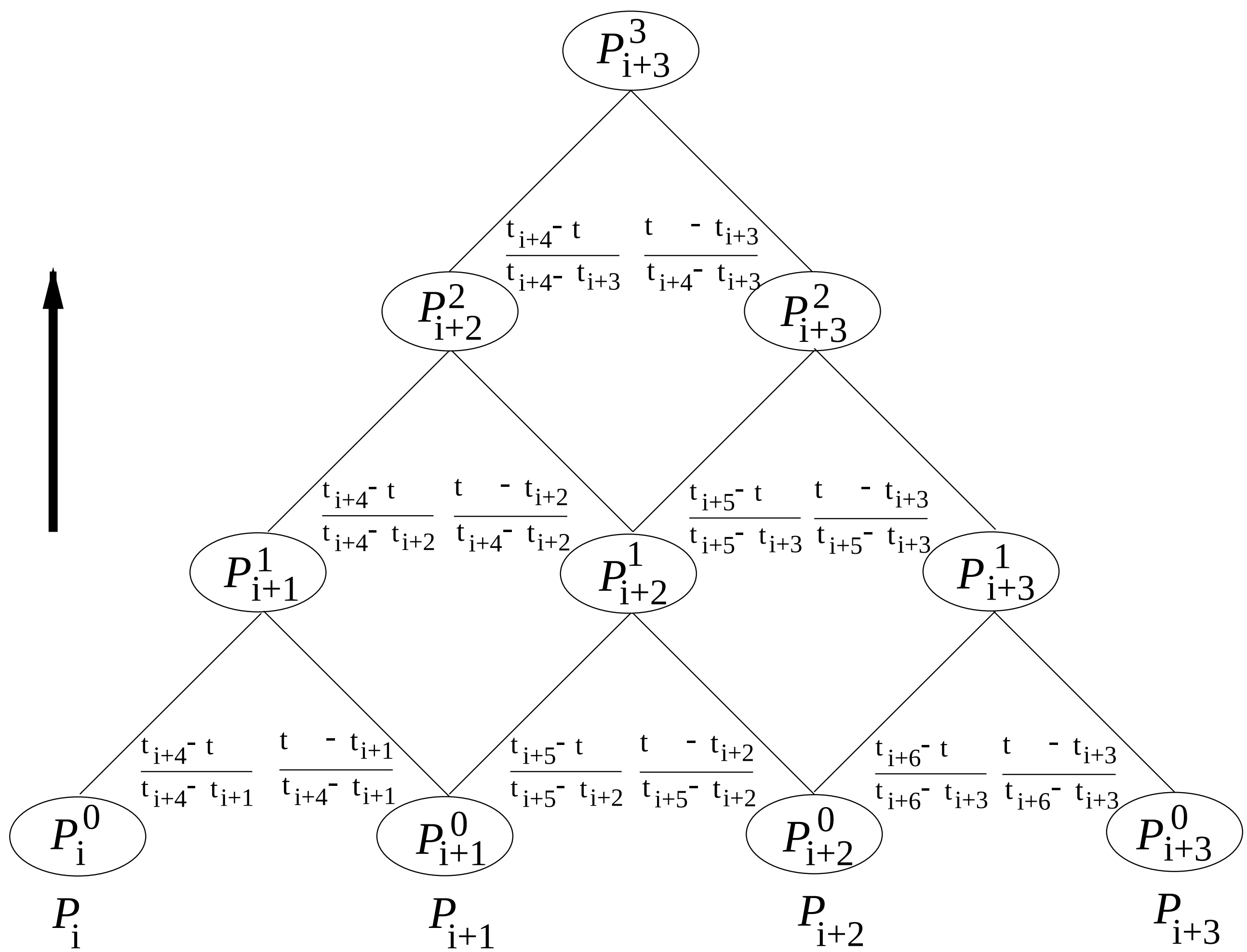
that is, just keep the old value.

General B-Splines:

- *Nonuniform B-splines (NUBS) generalize this construction*
- *A B-spline,* $B_i^d(t)$, *is a piecewise polynomial:*
  - *each of its segments is of degree* $\leq d$
  - *it is defined for all* t
  - *its segmentation is given by* $knots$ $t = t_0 \leq t_1 \leq \cdots \leq t_N$
  - *it is zero for* $T < T_i$ *and* $T > T_{i+d+1}$
  - *it may have a discontinuity in its* $d - k + 1$ *derivative at* $t_j \in \{t_i, \ldots, t_{i+d+1}\}$, *if* t_j *has multiplicity* k
  - *it is nonnegative for* $t_i < t < t_{i+d+1}$
  - $B_i^d(t) + \cdots + B_{i+d}(t) = 1$ *for* $t_{i+d} \leq t < t_{i+d+1}$, *and all other* $B_j^d(t)$ *are zero on this interval*
  - *Bézier blending functions are the special case where all knots have multiplicity* $d + 1$

Evaluation:

- *There is an efficient, recursive evaluation scheme for any curve point*
- *It generalizes the triangle scheme (deCasteljau) for Bézier curves*
- *Example (for cubics and* $t_{i+3} \leq t < t_{i+4}$*):*


P^3_{i+3}
$\frac{t_{i+4}-t}{t_{i+4}-t_{i+3}}$
$\frac{t-t_{i+3}}{t_{i+4}-t_{i+3}}$
P^2_{i+2}
P^2_{i+3}
$\frac{t_{i+4}-t}{t_{i+4}-t_{i+2}}$
$\frac{t-t_{i+2}}{t_{i+4}-t_{i+2}}$
$\frac{t_{i+5}-t}{t_{i+5}-t_{i+3}}$
$\frac{t-t_{i+3}}{t_{i+5}-t_{i+3}}$
P^1_{i+1}
P^1_{i+2}
P^1_{i+3}
$\frac{t_{i+4}-t}{t_{i+4}-t_{i+1}}$
$\frac{t-t_{i+1}}{t_{i+4}-t_{i+1}}$
$\frac{t_{i+5}-t}{t_{i+5}-t_{i+2}}$
$\frac{t-t_{i+2}}{t_{i+5}-t_{i+2}}$
$\frac{t_{i+6}-t}{t_{i+6}-t_{i+3}}$
$\frac{t-t_{i+3}}{t_{i+6}-t_{i+3}}$
P^0_i
P^0_{i+1}
P^0_{i+2}
P^0_{i+3}
P_i
P_{i+1}
P_{i+2}
P_{i+3}

Reading Assignment and News

Chapter 11 pages 583 - 600, of Recommended Text.

Please also track the News section of the Course Web Pages for the most recent Announcements related to this course.

(http://www.cs.utexas.edu/users/bajaj/graphics25/cs354/)