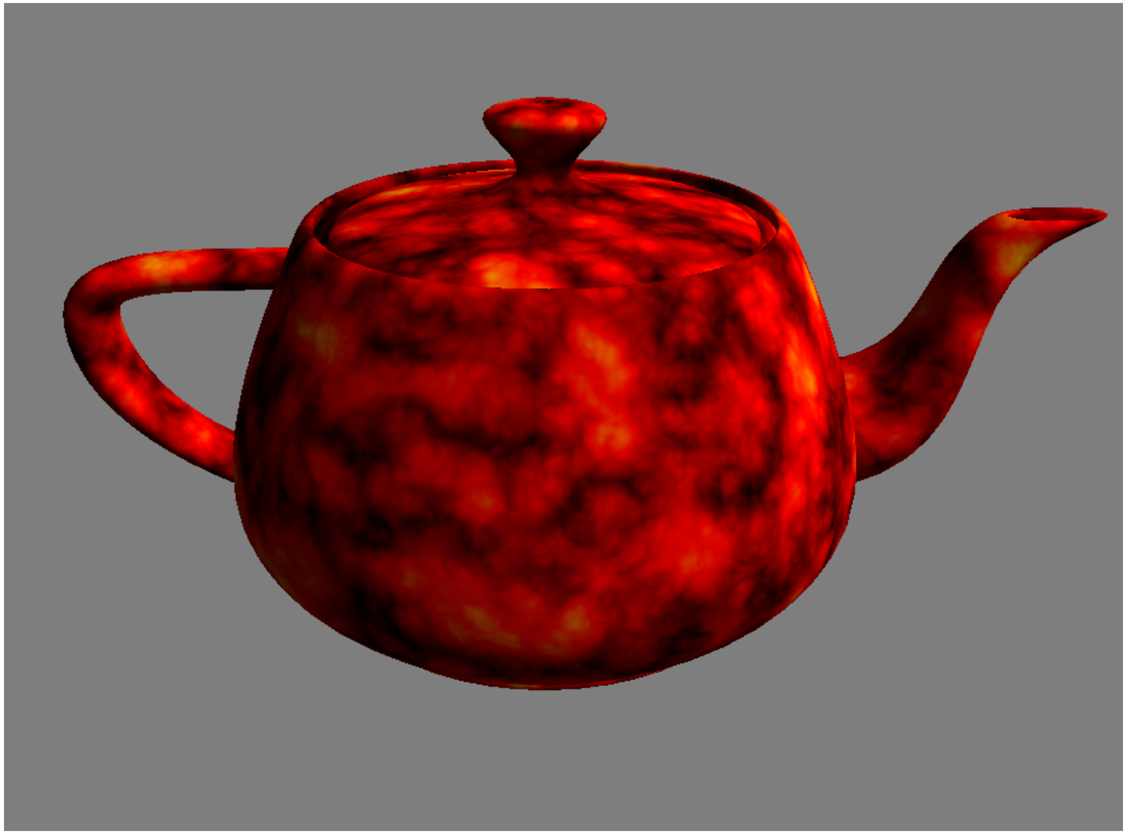
Curves and Surfaces (pp 597-623,643-648,654-660,321-342)



Different forms of curve functions

Explicit: $y = f(x)$, $z=g(x)$

- Cannot get multiple values for single x, infinite slopes

Implicit: $f(x,y,z) = 0$

- Cannot easily compare tangent vectors at joints
- In/Out test, normals form gradient

Parametric: $x=f_x(t)$, $y = f_y(t)$, $z= f_z(t)$

- Overcomes all problems

Describing curves by means of polynomials

Reminder:

Lth degre polynomial

$$p(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_L t^L$$

$a_0, \ldots a_L$ are the coefficients

L : is the degree

$L + 1$ is the order

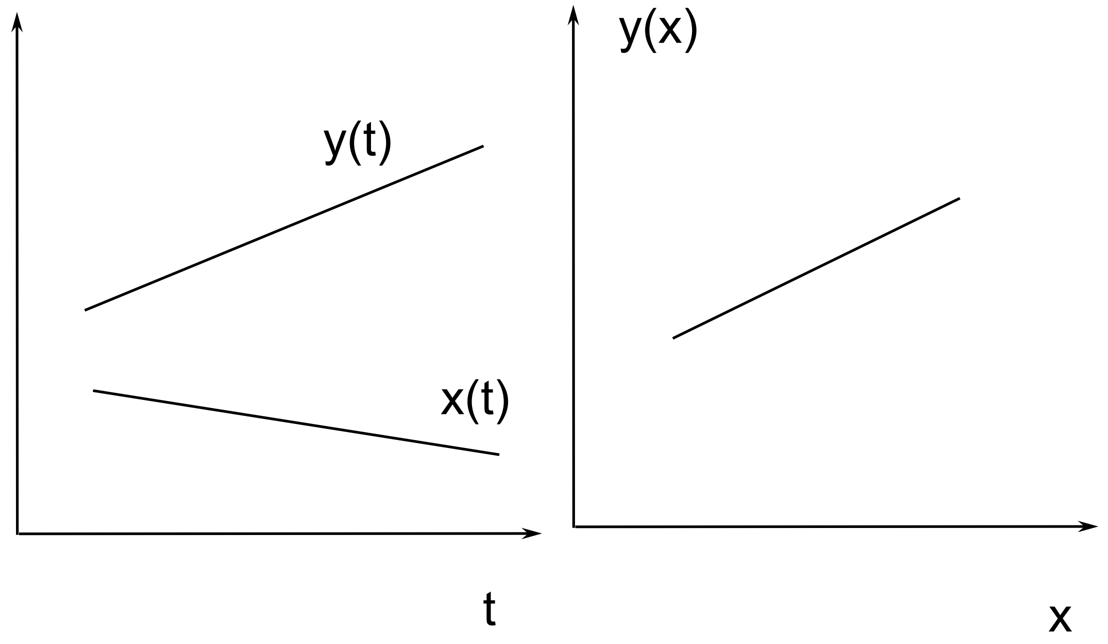
Polynomial curves of Degree 1

Parametric and implicit forms are linear

$x(t) = at + b$

$y(t) = ct + d$

$F(x,y) = kx + ly + m$



Polynomial Curves of Degree 2

Parametric

$x(t) = at^2+2bt+c$

$y(t) = dt^2+2et+f$

For any choice of constants

a,d,c,d,e,f $\rightarrow$parabola

Implicit

$F(x,y) = Ax^2+2Byx+Cy^2+Dx+Ey+F$

Let $d = AC-B^2$

$d > 0 \rightarrow F(x,y) = 0$ is an ellipse

$d = 0 \rightarrow F(x,y) = 0$ is a parabola

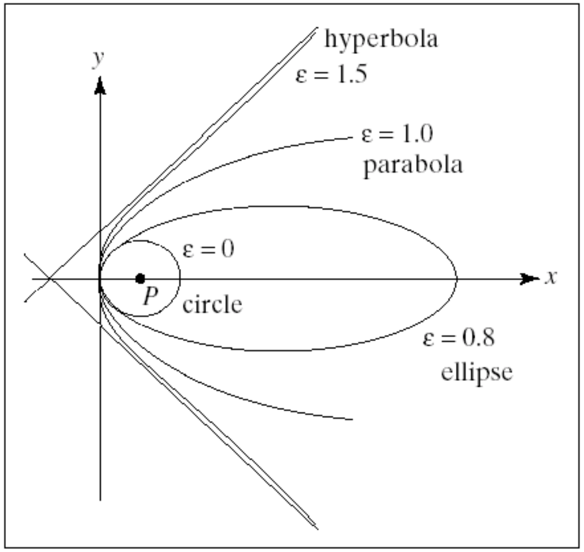
$d < 0 \rightarrow F(x,y) = 0$ is a hyperbola

Polynomial curves of degree 2

Common Vertex form:

$$y^2 = 2px - (1 - \varepsilon^2)x^2$$

FIGURE 11.5 The common-vertex equations of the conics.



from **Computer Graphics Using OpenGL, 2e**, by F. S. Hill
© 2001 by Prentice Hall / Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458

Rational Quadratic Parametric Curves

$$P(t) = \frac{P_0(1-t)^2 + 2wP_1t(1-t) + P_2t^2}{(1-t)^2 + 2wt(1-t) + t^2}$$

$w < 1$ ellipse

$w = 1$ parabola

$w > 1$ hyperbola

So

We will use parametric polynomials and constrain them to create desired types of curves.

How?

Interactive curve design

Geometric approach

Any t

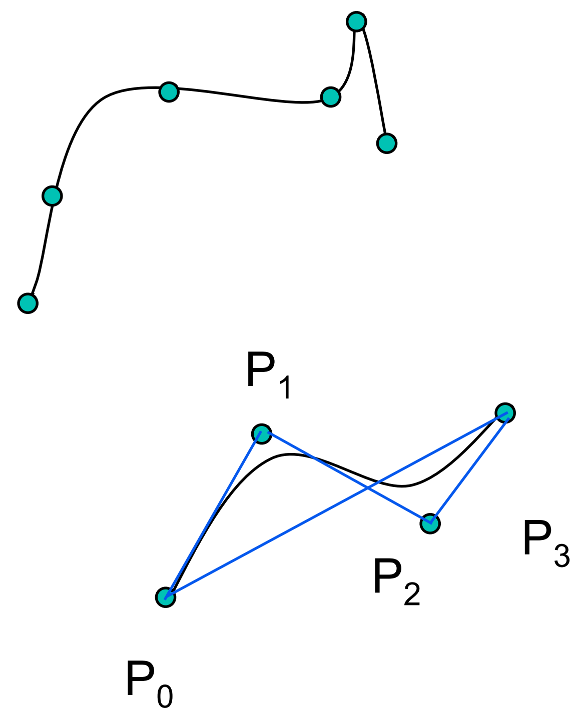
$P_0, \ldots, P_L \rightarrow$ [Curve generation] $\rightarrow P(t)$

Constraints *Polynomial* *Curve*

P_i control points

$P_0 \ldots P_L$ control polygon

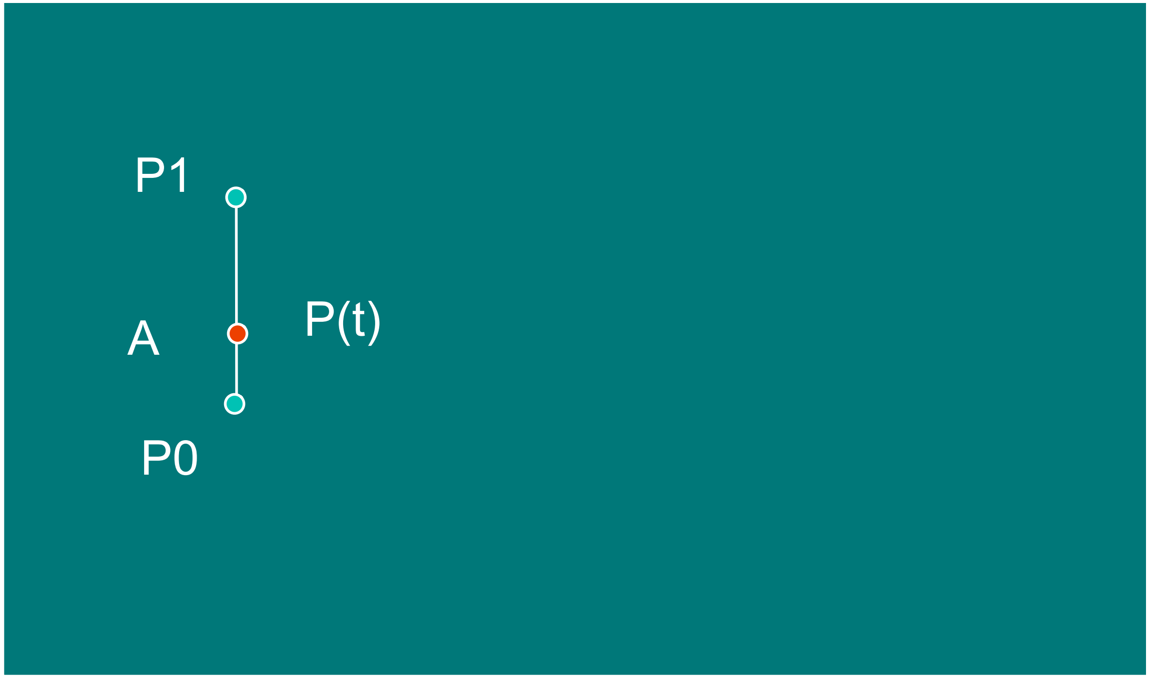
Interpolation vs Appoximation



De Casteljau Algorithm

Tweening

Two points=line

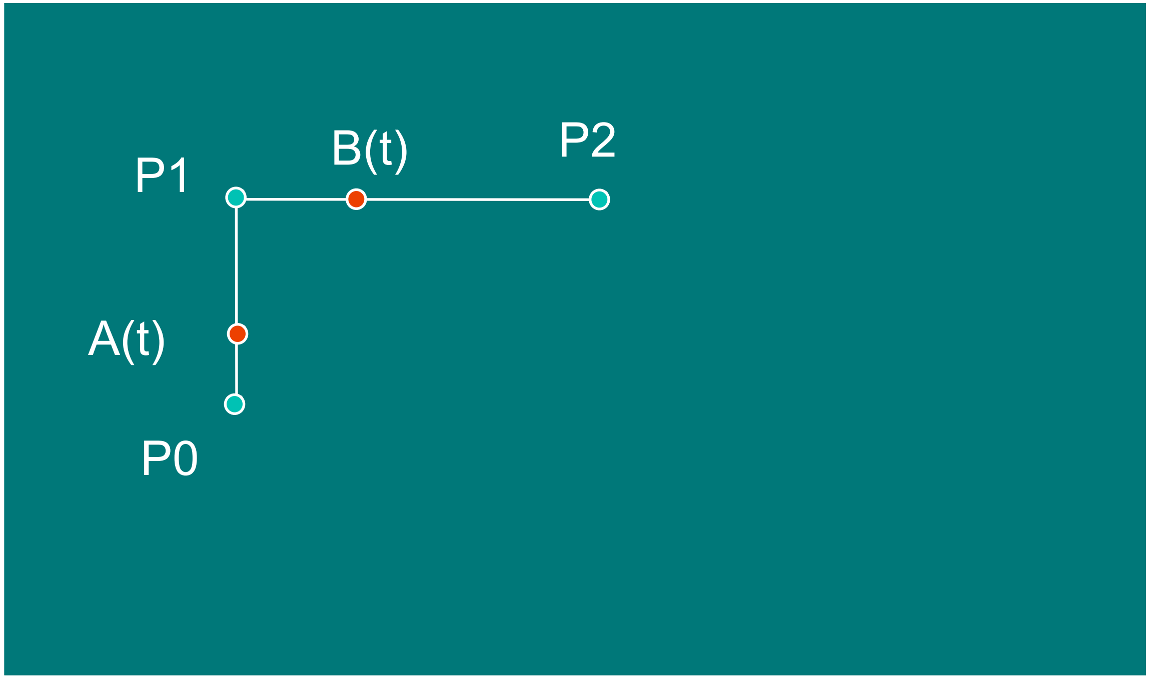


A(t) = (1-t)P0 + tP1

De Casteljau Algorithm

Tweening

Three points



$A(t) = (1-t)P0 + tP1$

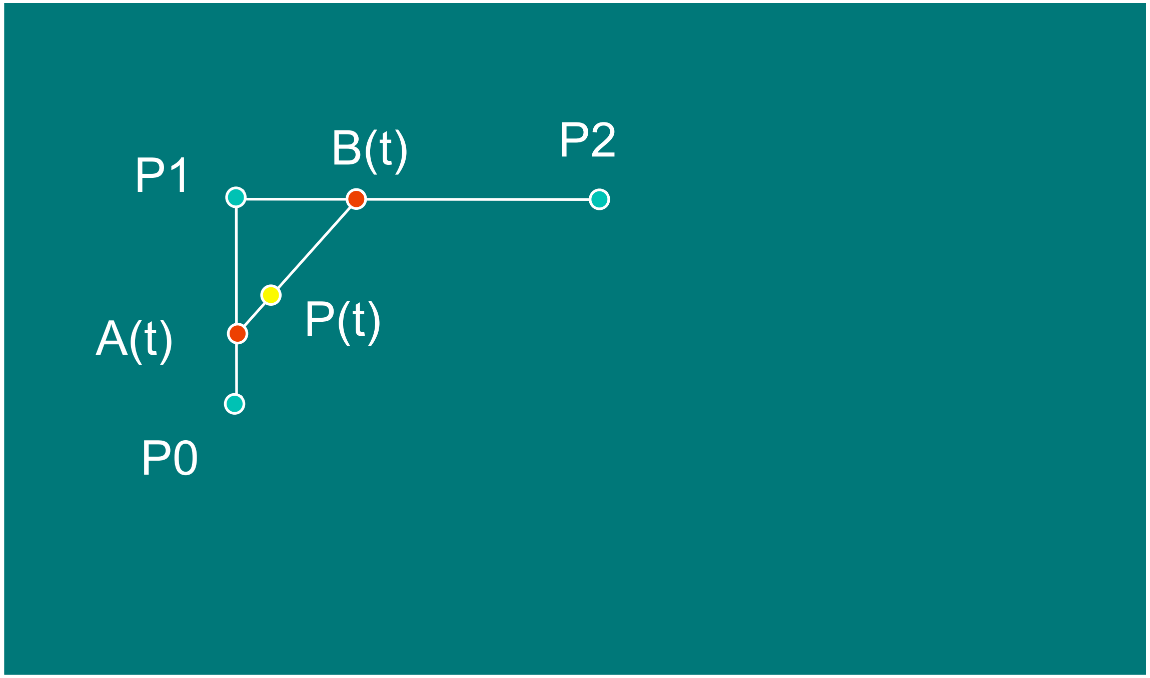
$B(t) = (1-t)P1 + tP2$

De Casteljau Algorithm

Tweening

Three points

(parabola)



$A(t) = (1-t)P0 + tP1$

$B(t) = (1-t)P1 + tP2$

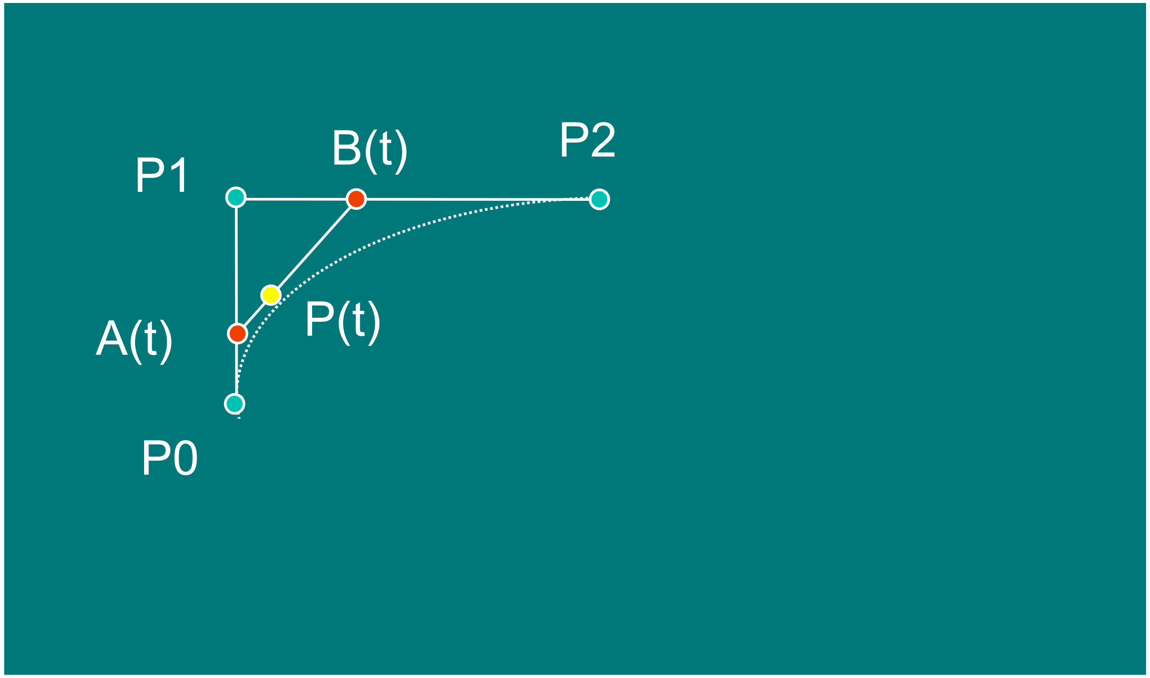
$P(t) = (1-t)\,A + tB = (1-t)^2P0 + 2t(1-t)P1 + t^2P2$

De Casteljau Algorithm

Tweening

Three points

(parabola)



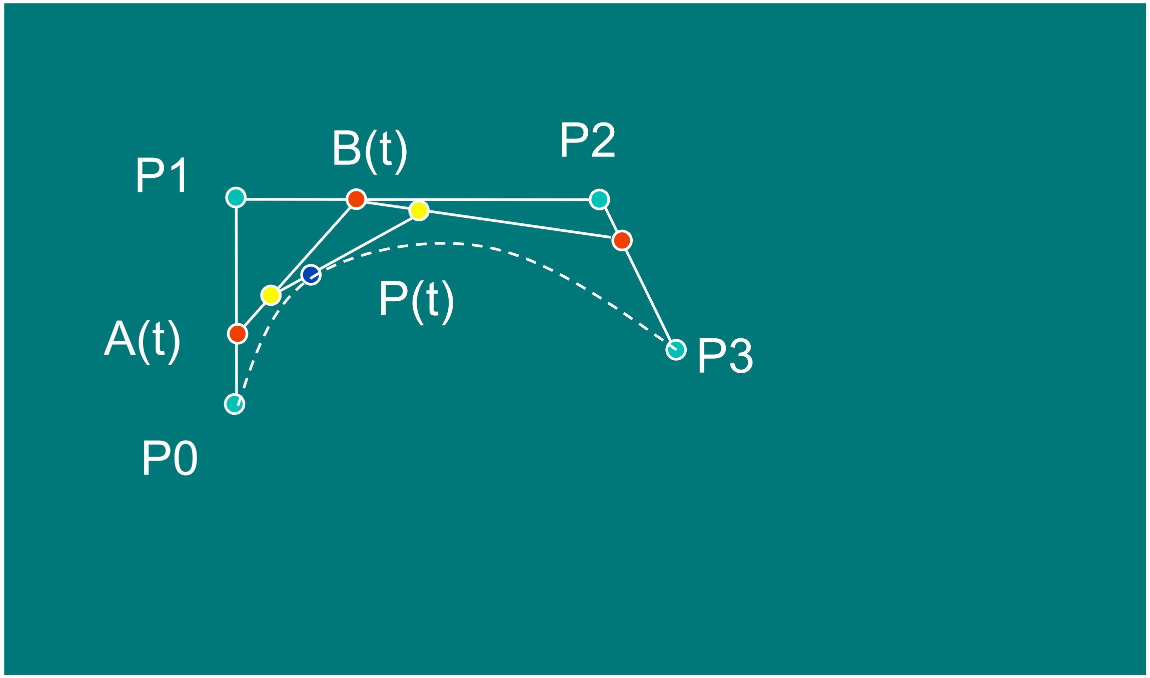
$A(t) = (1-t)P0 + tP1$

$B(t) = (1-t)P1 + tP2$

$P(t) = (1-t)\,A + tB = (1-t)^2 P0 + 2t(1-t)P1 + t^2 P2$

De Calsteljau (cont)

Tweening with four points



$$P(t) = (1-t)^3 P0 + 3(1-t)^2 t P1 + 3(1-t)t^2 P2 + t^3 P3$$

Cubic Berstein polynomials

$P(t) = (1-t)^3 P0 + 3(1-t)^2 t P1 + 3(1-t)t^2 P2 + t^3 P3$

$B^3_0\ (t) = (1-t)^3$

$B^3_1\ (t) = 3(1-t)^2 t$

$B^3_2\ (t) = 3(1-t)t^2$

$B^3_3\ (t) = t^3$

Expansion of $[(1-t) + t]^3 = (1-t)^3 + 3(1-t)^2 t + 3(1-t)t^2 + t^3 \rightarrow$

$\Sigma\ B^3_k\ (t) = 1,\ k = 0,1,2,3$

Affine combination of points

Berstein Polynomials of L degree

L + 1 control points

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

$$B_k^L(t) = \binom{L}{k} (1-t)^{L-k} t^k$$

$$\binom{L}{k} = \frac{L!}{k!(L-k)!}, \quad \text{for } L \geq k$$

$$\sum_{k=0}^{L} B_k^L(t) = 1, \quad \text{for all } t$$

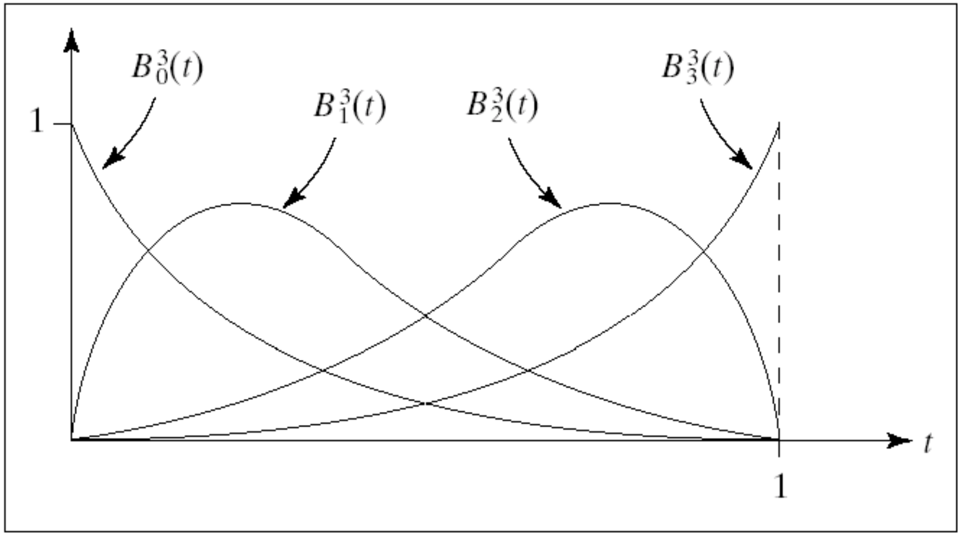
Expansion of $[(1-t) + t]^L$

Berstein Polynomials

Allways positive

Zero only at t =0 or 1

Degree 3



Properties of Bezier curves

- End point interpolation
- Affine Invariance: $$T(P(t)) = \sum_{k=0}^{L} B_k^L(t) T(P)_k$$
- Invariace under affine transformation of the parameter
- Convex Hull property for t in [0,1] $$P = \sum_{k=0}^{L} a_k P_k, \text{ where } \sum_{k=0}^{L} a_k = 1 \text{ and } a_k > 0$$
- Linear precision by collapsing convex hull
- Variation Diminishing property: No straight line cuts the curve more times than it cuts the control polygon

Derivatives of Bezier curves

It can be shown that:

Velocity also a Bezier curve of lower degree

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

Acceleration:

$$P''(t) = L(L-1)\sum_{k=0}^{L-2} B_k^{L-2}(t)\Delta^2 P_k \text{ where } \Delta^2 P_k = \Delta P_{k+1} - \Delta P_k$$

Which degree is best?

Cubic curves

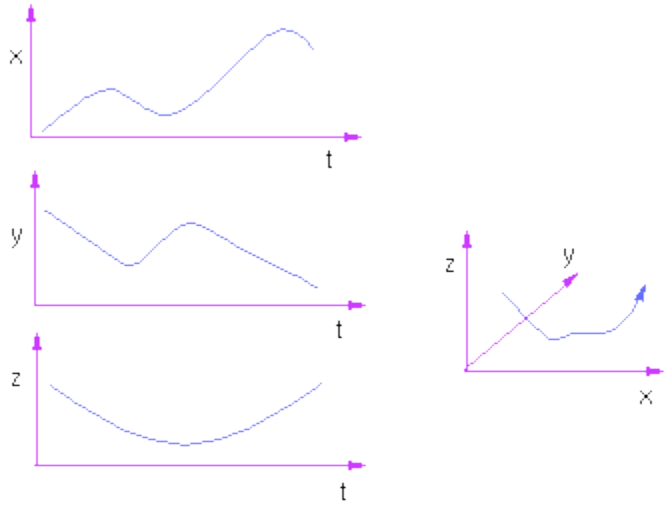
- Lower order not enough flexibility
- Higher order too many wiggles and computationally expensive
- Cubic curves are lowest degree polynomial curves that are not planar in 3D

More complex curves

- Piecewise cubics

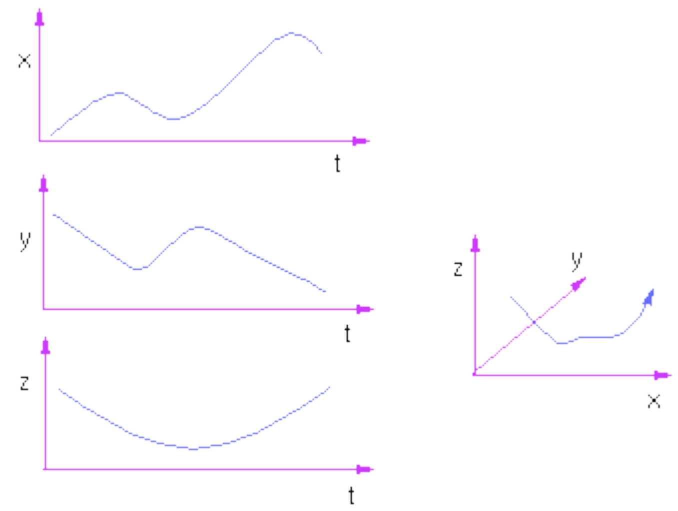
Cubic parametric curves

$$x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$$
$$y(t) = b_3t^3 + b_2t^2 + b_1t + b_0$$
$$z(t) = c_3t^3 + c_2t^2 + c_1t + c_0$$
$$t \in [0, 1]$$



Cubic parametric curves (Matrix Form)

$$x(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$
$$y(t) = b_3 t^3 + b_2 t^2 + b_1 t + b_0$$
$$z(t) = c_3 t^3 + c_2 t^2 + c_1 t + c_0$$
$$t \in [0, 1]$$



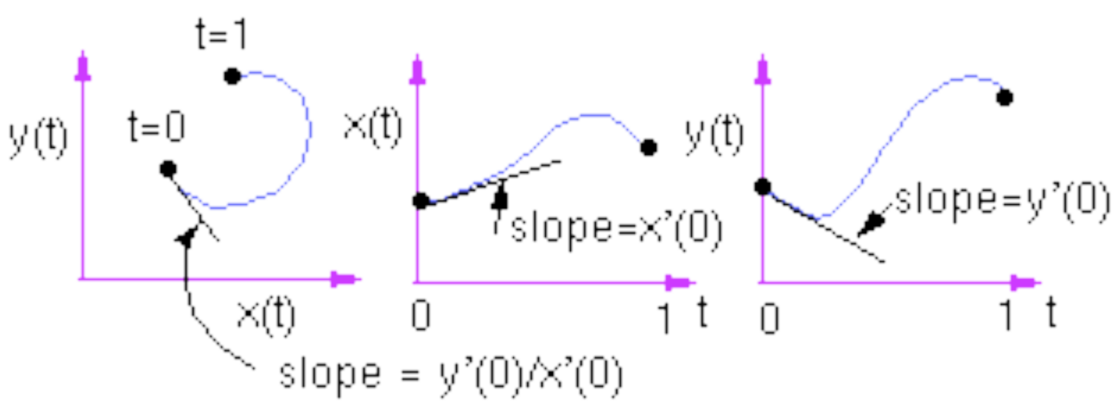
$$x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

$$x(t) = TA$$
$$y(t) = TB$$
$$z(t) = TC$$

Derivative of Cubic Parametric Curves

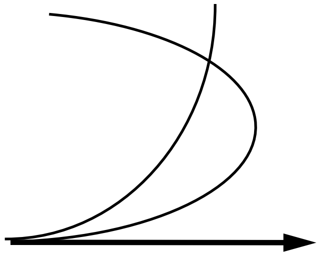
$$x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

$$x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$



How does the magnitude of the tangent affect the curve?

Same lower tangent direction but different magnitude.



The magnitude defines how fast the curve assumes the tangent direction (remember: tangent → velocity in parametric space)

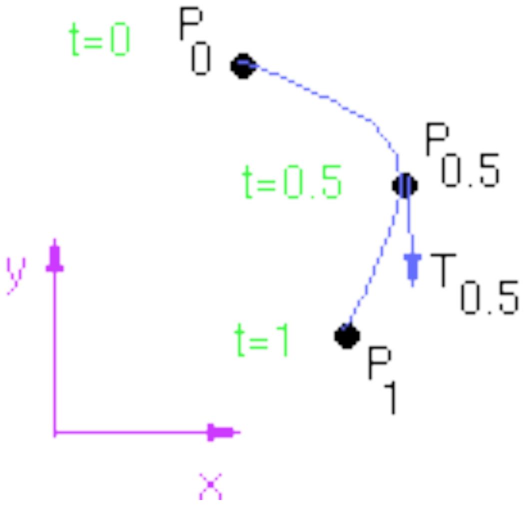
Example

Constraints

Endpoints and a tangent at midpoint

$$x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & t \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

$$x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

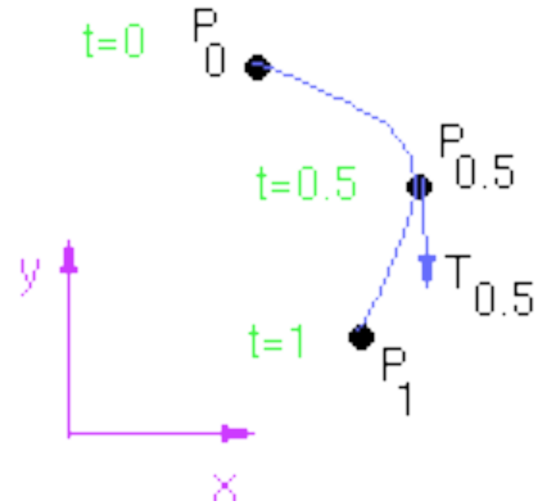


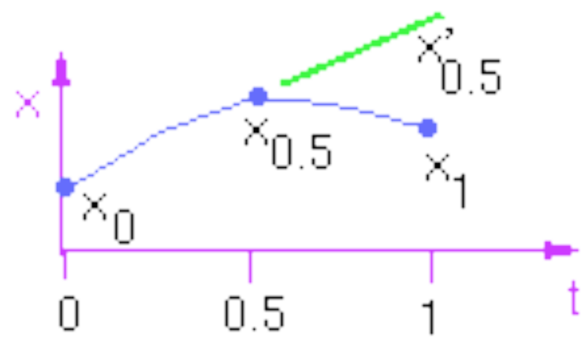
Setting up the curve

Constraints

$$x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} A$$
$$x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} A$$

$$x(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A$$
$$x(0.5) = \begin{bmatrix} 0.5^3 & 0.5^2 & 0.5 & 1 \end{bmatrix} A$$
$$x'(0.5) = \begin{bmatrix} 3(0.5)^2 & 2(0.5) & 1 & 0 \end{bmatrix} A$$
$$x(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A$$
$$G_x = BA$$



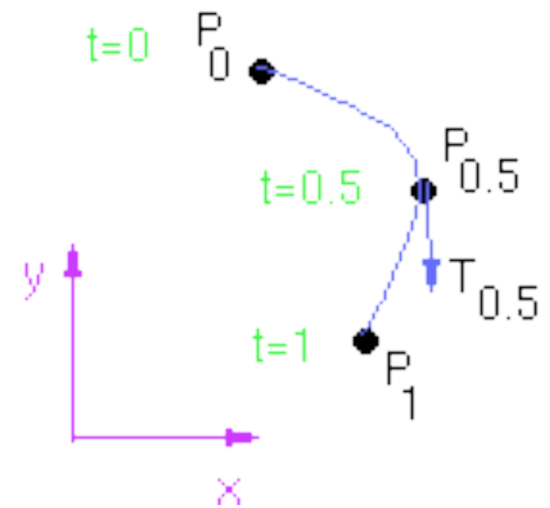


Solving for A

Constraints

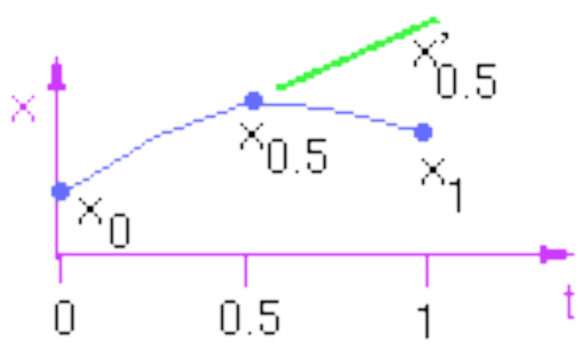
$$x(t) = \begin{bmatrix} t^3 & t^2 & t^1 & 1 \end{bmatrix} A = TA$$

$$x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} A = T'A$$



$$\begin{bmatrix} x_0 \\ x_{0.5} \\ x'_{0.5} \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.5^3 & 0.5^2 & 0.5 & 1 \\ 3(0.5)^2 & 2(0.5) & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} A$$

$$G_x = BA \Rightarrow A = B^{-1}G_x$$



$$x(t) = TA \Rightarrow \boxed{x(t) = TB^{-1}G_x}$$

Final form

Basis matrix

$$x(t) = TB^{-1}G_x$$

$$Set \;\; M = B^{-1}$$

$$x(t) = TMG_x$$

$$y(t) = TMG_y$$

$$z(t) = TMG_z$$

For the example

$$P(t) = TMG$$

$$M = \begin{bmatrix} -4 & 0 & -4 & 4 \\ 8 & -4 & 6 & -4 \\ -5 & 5 & -2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Blending functions

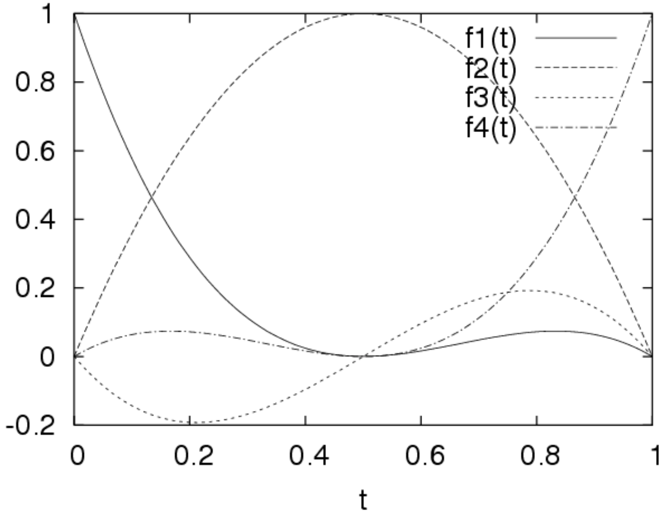
T*M

$$x(t) = TMG_x \Rightarrow$$
$$x(t) = \left[\begin{array}{cccc} f_1(t) & f_2(t) & f_3(t) & f_4(t) \end{array} \right] G_x$$

For the example

$$f_1(t) = -4t^3 + 8t^2 - 5t + 1$$
$$f_2(t) = -4t^2 + 4t$$
$$f_3(t) = -4t^3 + 6t^2 - 2t$$
$$f_4(t) = 4t^3 - 4t^2 + t$$

Each blending function weights the contribution of one of the constraints



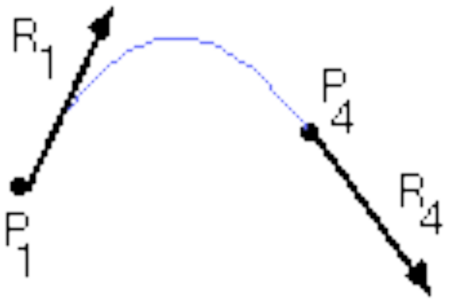
Hermite Curves

Constraints

Two points and two tangents

$$G_h = \begin{bmatrix} P_1 & P_4 & R_1 & R_4 \end{bmatrix}$$
$$x(t) = TA_h = TM_hG_h$$

$$x(0) = P_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A_h$$
$$x(1) = P_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A_h$$
$$x'(0) = R_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} A_h$$
$$x'(1) = R_4 = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} A_h$$
$$G_h = B_hA_h$$
$$A_h = B_h^{-1}G_h$$
$$x(t) = TA_h$$



Hermite Curves

Blending functions

$$M_h = B_h^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$x(t) = TM_hG_h \Rightarrow$$

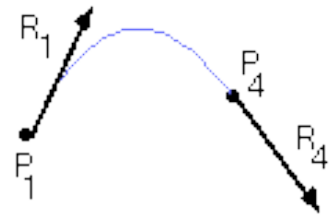
$$x(t) = \begin{bmatrix} f_1(t) & f_2(t) & f_3(t) & f_4(t) \end{bmatrix} G_h$$

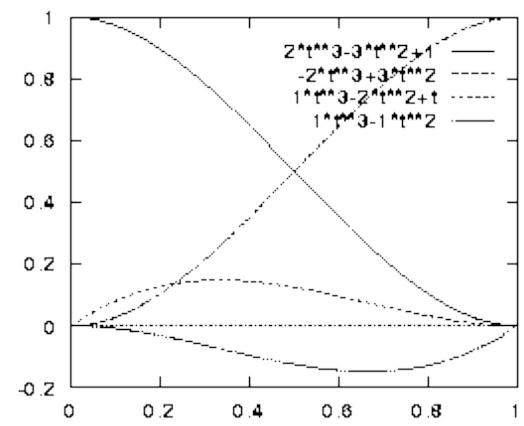
$$f_1(t) = 2t^3 - 3t^2 + 1$$

$$f_2(t) = -2t^3 + 3t^2$$

$$f_3(t) = t^3 - 2t^2 + t$$

$$f_4(t) = t^3 - t^2$$





Bezier Curves

Special case of Hermite curves

$$P_{1,h} = P_1$$
$$P_{4,h} = P_4$$
$$R_{1,h} = 3(P_2 - P_1)$$
$$R_{4,h} = 3(P_4 - P_3)$$

Bezier Curves

Special case of Hermite curves

$$P_{1,h} = P_1$$
$$P_{4,h} = P_4$$
$$R_{1,h} = 3(P_2 - P_1)$$
$$R_{4,h} = 3(P_4 - P_3)$$
$$\begin{bmatrix} P_{1,h} \\ P_{4,h} \\ R_{1,h} \\ R_{4,h} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

Bezier Curves

Special case of Hermite curves

$$\begin{bmatrix} P_{1,h} \\ P_{4,h} \\ R_{1,h} \\ R_{4,h} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

$$G_h = M_{bh} G_b$$

$$P(t) = TM_h G_h \Rightarrow P(t) = TM_h M_{bh} G_b \Rightarrow$$

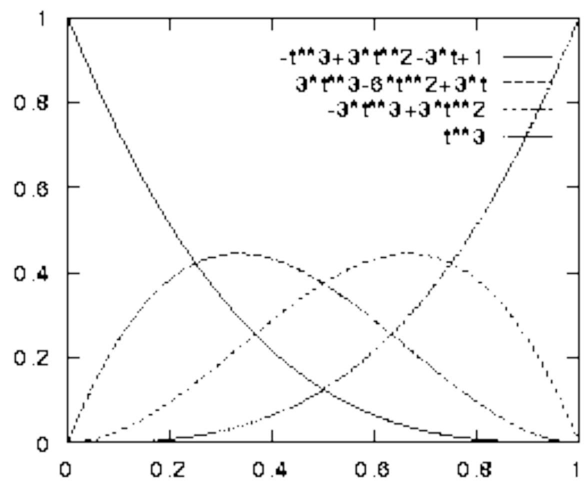
$$P(t) = TM_b G_b$$

Bezier Curves

Special case of Hermite curves

We can verify that TM_b are the bernstein polynomials

$$f_1(t) = (1-t)^3$$
$$f_2(t) = 3t(1-t)^2$$
$$f_3(t) = 3t^2(1-t)$$
$$f_4(t) = t^3$$



Transforming between representations

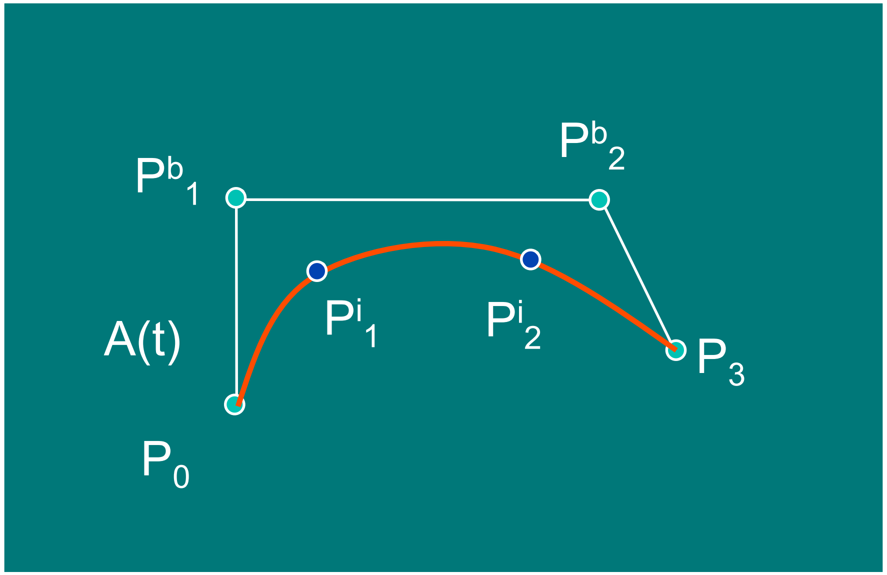
Just like Bezier and Hermites curves can be transformed into each other with a matrix multiplication, other families of curve can do so as well

Bezier to Interpolating curves

Curve interpolates

$P^i_0, P^i_1, P^i_2, P^i_3$

How can we find the P^b points from the P^i?

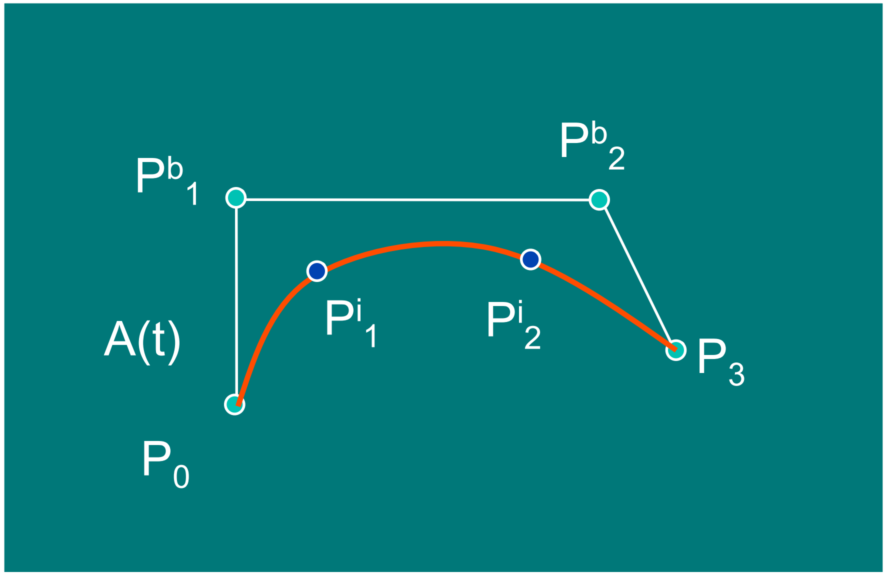


Bezier to Interpolating curves

For the next three slides points are row vectors!!

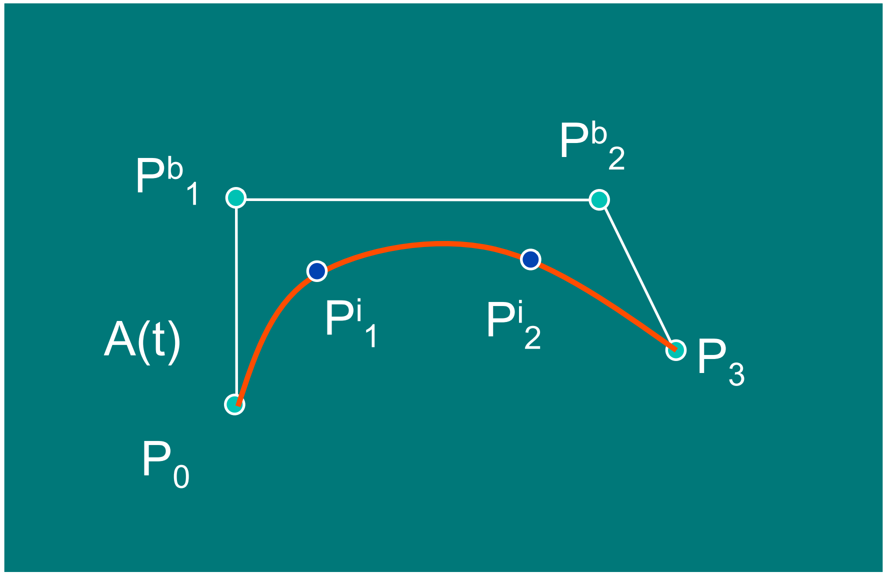
$$P_j^i = [\ P_j^i, x \quad P_j^i, y \quad P_j^i, z\ ]$$

$$G^b = \begin{pmatrix} P_0^b \\ P_1^b \\ P_2^b \\ P_3^b \end{pmatrix}$$



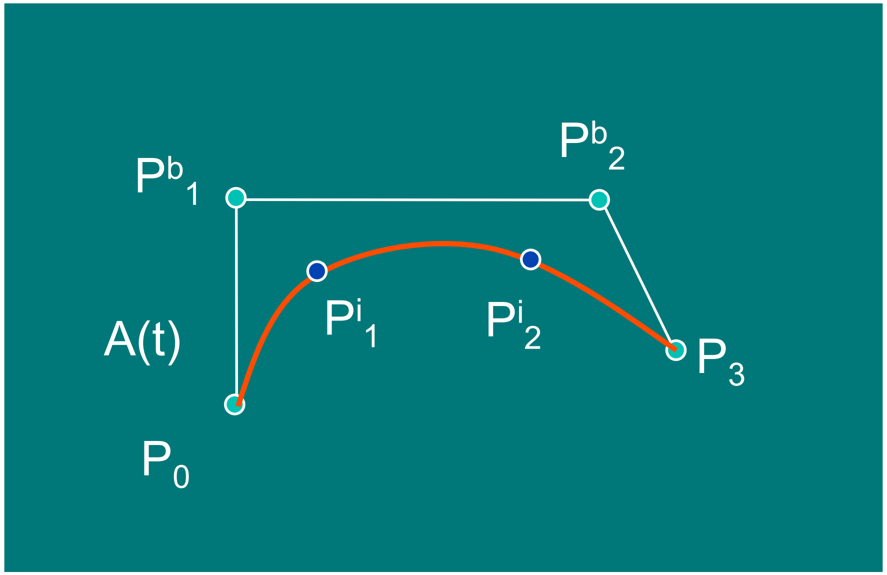
$$\begin{aligned} P_0^i &= T(0)M_bG^b \\ P_1^i &= T(\tfrac{1}{3})M_bG^b \\ P_2^i &= T(\tfrac{2}{3})M_bG^b \\ P_3^i &= T(1)M_bGx^b \end{aligned} \rightarrow \begin{pmatrix} P_0^i \\ P_1^i \\ P_2^i \\ P_3^i \end{pmatrix} = \begin{pmatrix} T(0) \\ T(\frac{1}{3}) \\ T(\frac{2}{3}) \\ T(1) \end{pmatrix} M_b \begin{pmatrix} P_0^b \\ P_1^b \\ P_2^b \\ P_3^b \end{pmatrix}$$

Bezier to Interpolating curves



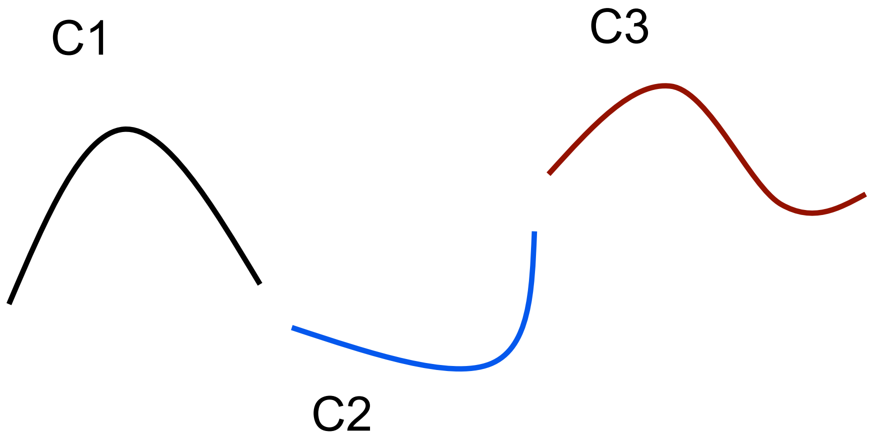
$$\begin{pmatrix} P_0^i \\ P_1^i \\ P_2^i \\ P_3^i \end{pmatrix} = \begin{pmatrix} T(0) \\ T(\frac{1}{3}) \\ T(\frac{2}{3}) \\ T(1) \end{pmatrix} M_b \begin{pmatrix} P_0^b \\ P_1^b \\ P_2^b \\ P_3^b \end{pmatrix} \Rightarrow \mathbf{P}^i = T M_b \mathbf{P}^b$$

Bezier to Interpolating curves



$$\mathbf{P}^i = TM_b\mathbf{P}^b \Leftrightarrow \mathbf{P}^b = (TM_b)^{-1}\mathbf{P}^i$$

Piecewise cubic curves



Connection?

Continuity

Geometric G^k-continuity

$P^{(i)}(t-) = c_i P^{(i)}(t+)$ $\forall t$ in [a,b]

for i = 0,…,k and

for some c_i constants

Parametric C^k-continuity

$P^{(i)}$ exists and is continuous $\forall t$ in [a,b], for i = 0,…,k

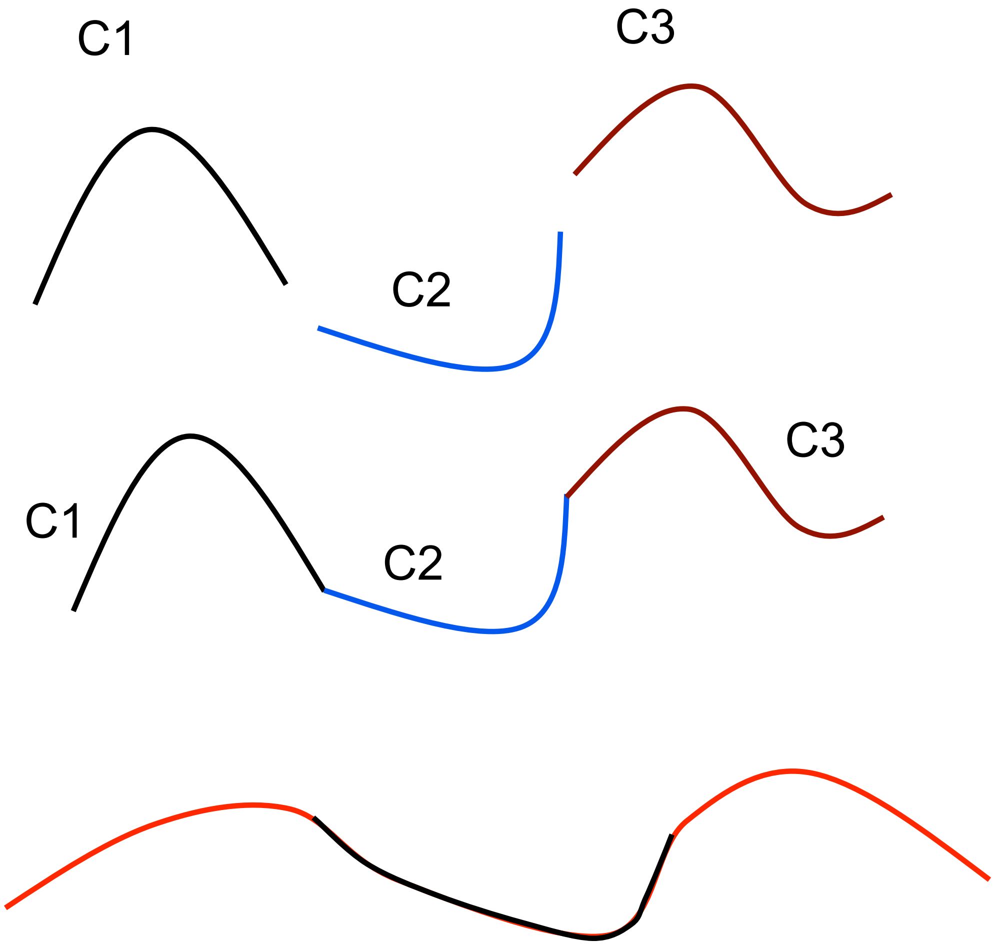
Terminology:

P is k-smooth

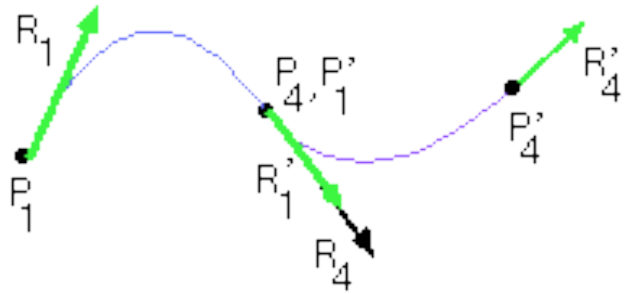
P has kth-order continuity

Is a C^k-continuous function G^K continuous as well?

Examples

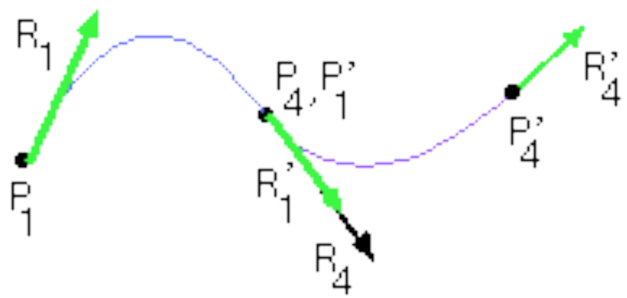


Piecewise Cubic Hermite Curves



What are the conditions for G1 continuity?

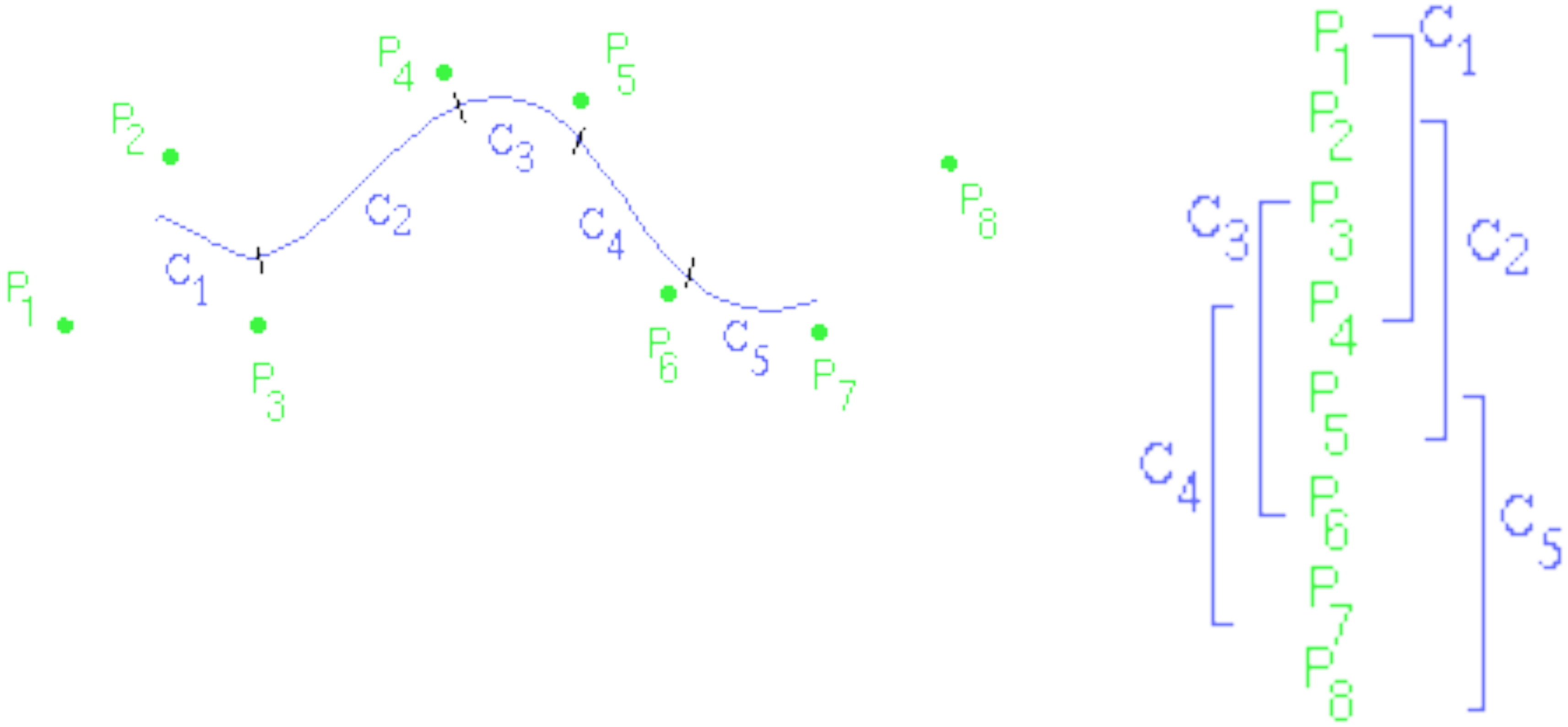
Piecewise Cubic Hermite Curves



$R'1 = kR4$

$P1' = P4$

BSplines



Matrix form

For a bspline curve with:

- m+1 control points $P_0, \ldots, P_m$
- m-2 segments $Q_3, \ldots, Q_m$
- t in [3,…,m]

$$Q_i(t) = \left[\begin{array}{cccc} (t-t_i)^3 & (t-t_i)^2 & (t-t_i) & 1 \end{array} \right] \mathbf{M}_{bspline} \left[\begin{array}{c} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{array} \right]$$

Properties

C2 continuous

Convex hull property

NO invariace under perspective projection!

NURBS: Nonuniform Rational B-splines

$$X(t) = X(t) / W(t)$$

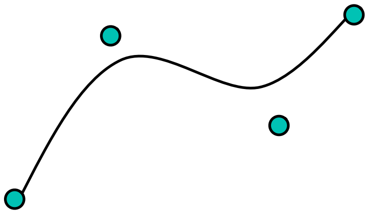
$$Y(t) = Y(t) / W(t)$$

$$Z(t) = Z(t) / W(t)$$

- Exact conic sections
- Invariance under perspective projection

Summary: General problem

$P_0, \ldots, P_L \rightarrow$ Curve generation $\rightarrow P(t)$



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

where

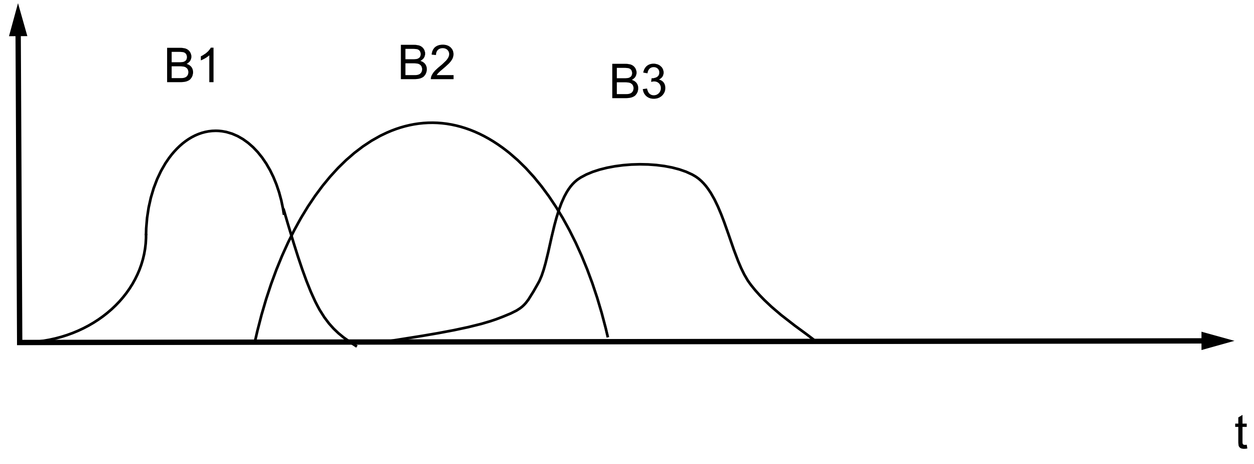
$B_k(t)$: Blending functions

$P_k, k = 1, \ldots L$ Control Points

$t \in [a, b]$

Blending functions

Weight the influence of each constraint (e.g. control point) on the curve created.



Wish list for blending functions

- Easy to compute and stable
- Sum to unity for every t in [a,b]
- Support over portion of [a,b]
- Interpolate certain control points
- Sufficient smoothness

Example: Bezier curves

- Sum up to unity
- Smooth
- Interpolate first and last
- Expensive to compute for large L
- No local control

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

$$B_k^L(t) = \binom{L}{k} (1-t)^{L-k} t^k$$

$$\binom{L}{k} = \frac{L!}{k!(L-k)!}, \quad \text{for } L \geq k$$

$$\sum_{k=0}^{L} B_k^L(t) = 1, \quad \text{for all } t$$

Rendering parametric curves

Transform into primitives we know how to handle

Curves

- Line segments

Converting to Lines

Straightforward

Uniform subdivision

Evaluation of C(t) at t: 0, dt, 2dt,…,1.

Draw as lines.

Curves in OpenGL

```
GLfloat ctrlpoints[4][3] = { { -4.0, -4.0, 0.0},
                             { -2.0, 4.0, 0.0},
                             {2.0, -4.0, 0.0},
                             {4.0, 4.0, 0.0}
};

void myinit(void) {
      glClearColor(0.0, 0.0, 0.0,1.0);
                                    // t1,t2,stride,order
      glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &ctrlpoints[0][0]);
      glEnable(GL_MAP1_VERTEX_3);
}
```

Stride

OpenGL allows interleaved information

```
GLfloat ctrlpts[1000] = {
    x1, y1,z1, nx1, ny1,nz1, tx1,ty1,
    x2,y2,z2, nx2,ny2,nz2,tx2,ty2,
    …..
}
```

Stride here is 8

Evaluating and displaying

```
void display(void) {
    int i;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);

    glBegin(GL_LINE_STRIP);
    for (i = 0; i <= 30; i++)
            glEvalCoord1f((GLfloat) i/30.0);
    glEnd();

    /* The following code displays the control points as dots. */
    glPointSize(5.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_POINTS);
    for (i = 0; i < 4; i++)
            glVertex3fv(&ctrlpoints[i][0]);
    glEnd();
    glFlush();
}
```

Uniform subdivision

void glMapGrid1{fd}(GLint n, TYPE u1, TYPE u2);

Defines a grid that goes from u1 to u2 in n steps, which are evenly spaced.

Evaluation for n in [n1,n2] using:

void glEvalMesh1(GLenum mode, GLint t1, GLint t2);

Uniform subdivision

```
void display(void) {
    int i;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);

    glMapGrid1(30,0.0,1.0) ;
    glEvalMesh1(GL_LINE,0,30) ;

    /* The following code displays the control points as dots. */
    glPointSize(5.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_POINTS);
    for (i = 0; i < 4; i++)
                glVertex3fv(&ctrlpoints[i][0]);
    glEnd();
    glFlush();
}
```

Evaluators can do more than position

Color

Normal

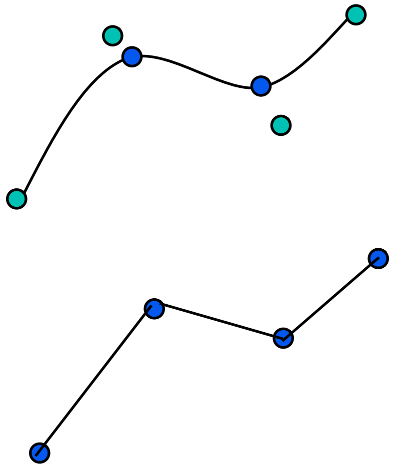
Texture coordinates

How many evaluation points are enough for Bezier curves?

Not too few

Not too many

Ok, how many?

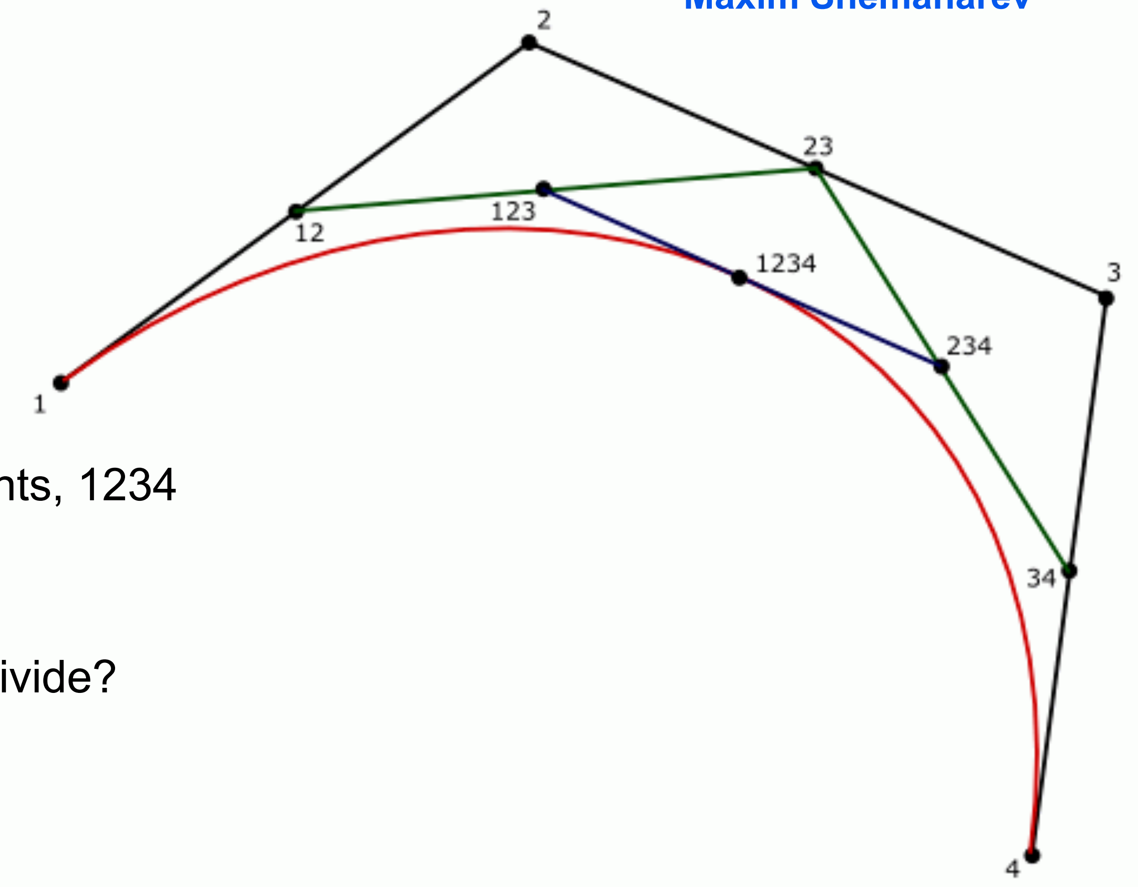


Adaptive Subdivision of Bezier Curves

Images courtesy of
Maxim Shemanarev

de Casteljau subdivision

One Bezier curve becomes 2 flatter curves



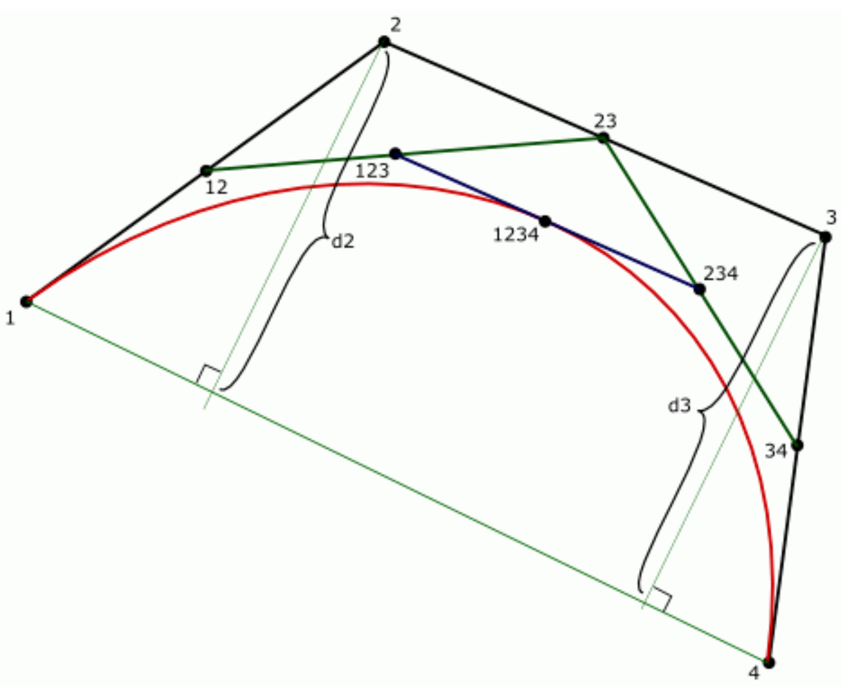
Original points 1,2,3,4 →
Midpoints 12, 23, 34
Midpoints of midpoints: 123, 234
Midpoints of midpoints of midpoints, 1234
Remember: tweening for t = 0.5
Can chose any t we want
Ok, how many times do we subdivide?

Error metrics

Images courtesy of
Maxim Shemanarev

Examples:

Point distance



Tangent distance