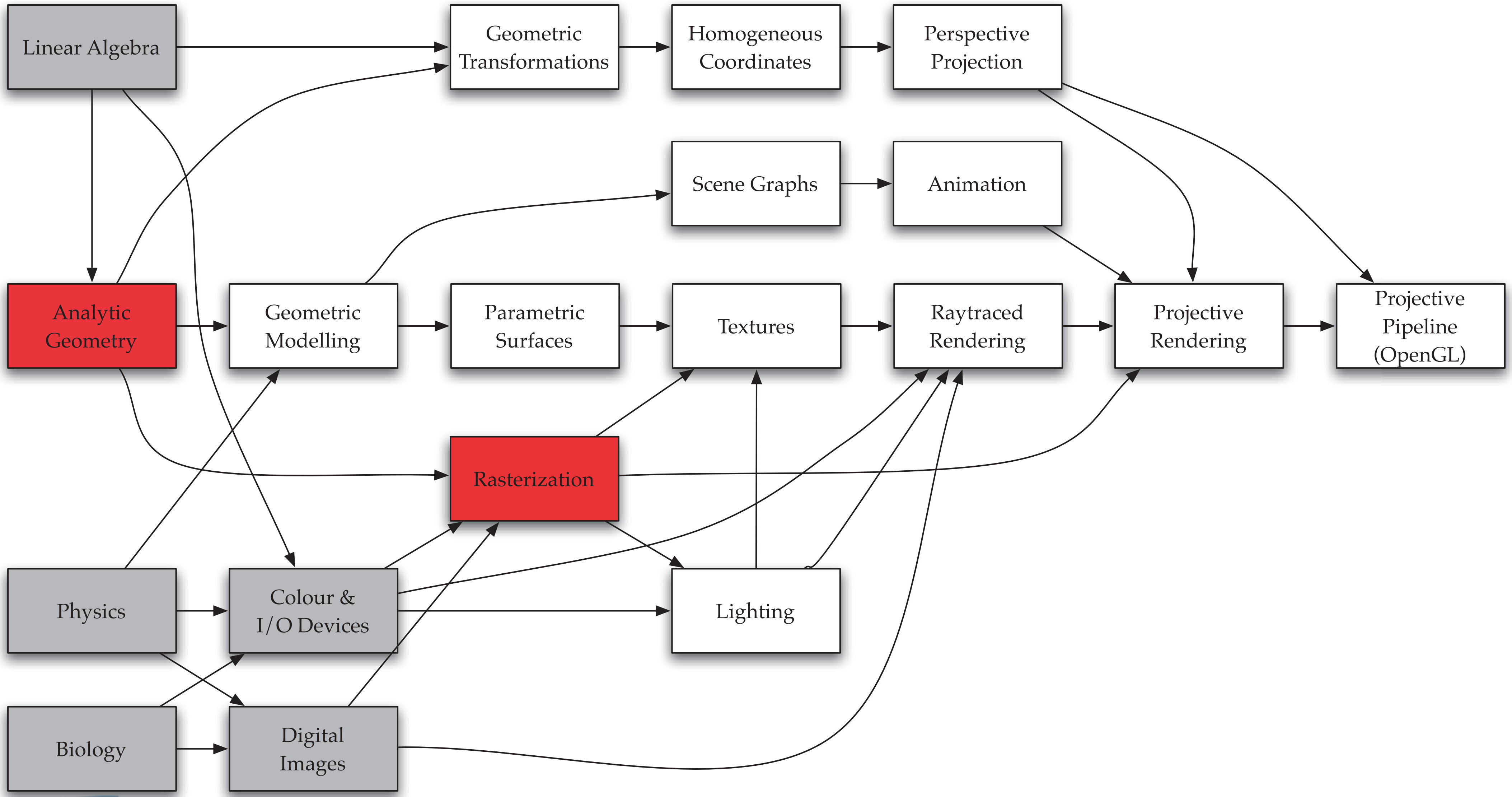
Hermite & Bézier Curves

COMP 30020

Where we Are



Linear Algebra
Geometric Transformations
Homogeneous Coordinates
Perspective Projection
Scene Graphs
Animation
Analytic Geometry
Geometric Modelling
Parametric Surfaces
Textures
Raytraced Rendering
Projective Rendering
Projective Pipeline (OpenGL)
Rasterization
Physics
Colour & I/O Devices
Lighting
Biology
Digital Images

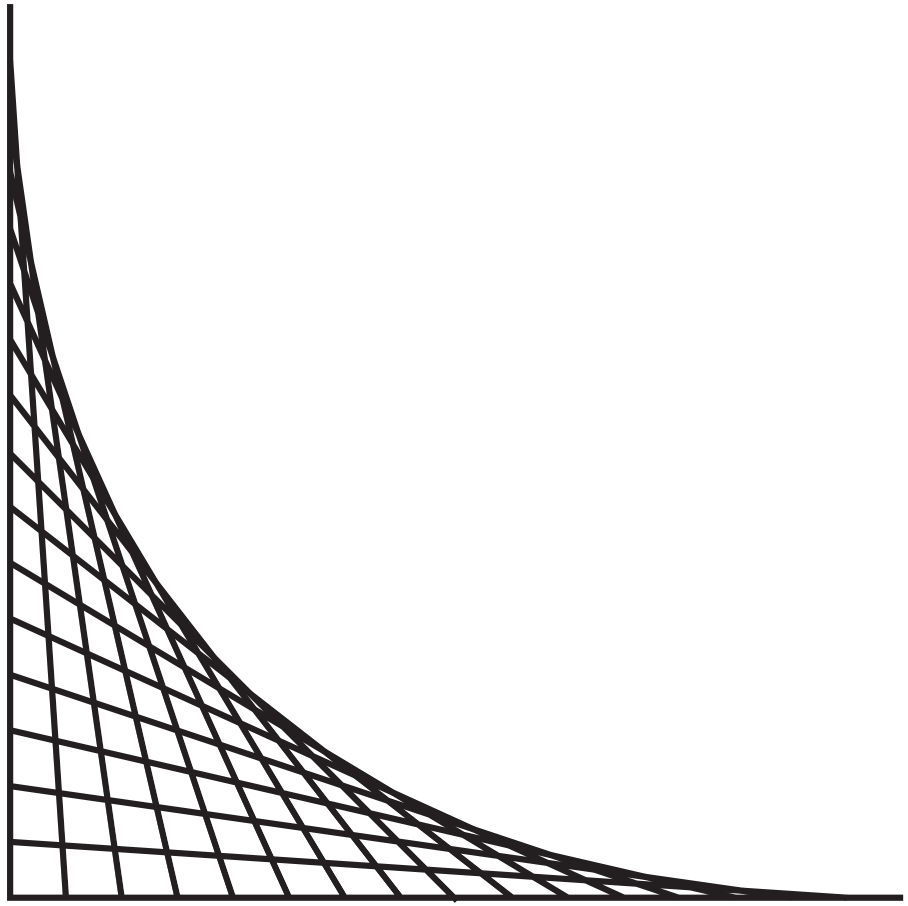
Continuity

- We want *smooth* curves (& surfaces)
- I.e. we need *C^1 continuity*
  - and we want to build them from *lines*
  - repeated linear interpolation

String Art

from www.stringart.eu

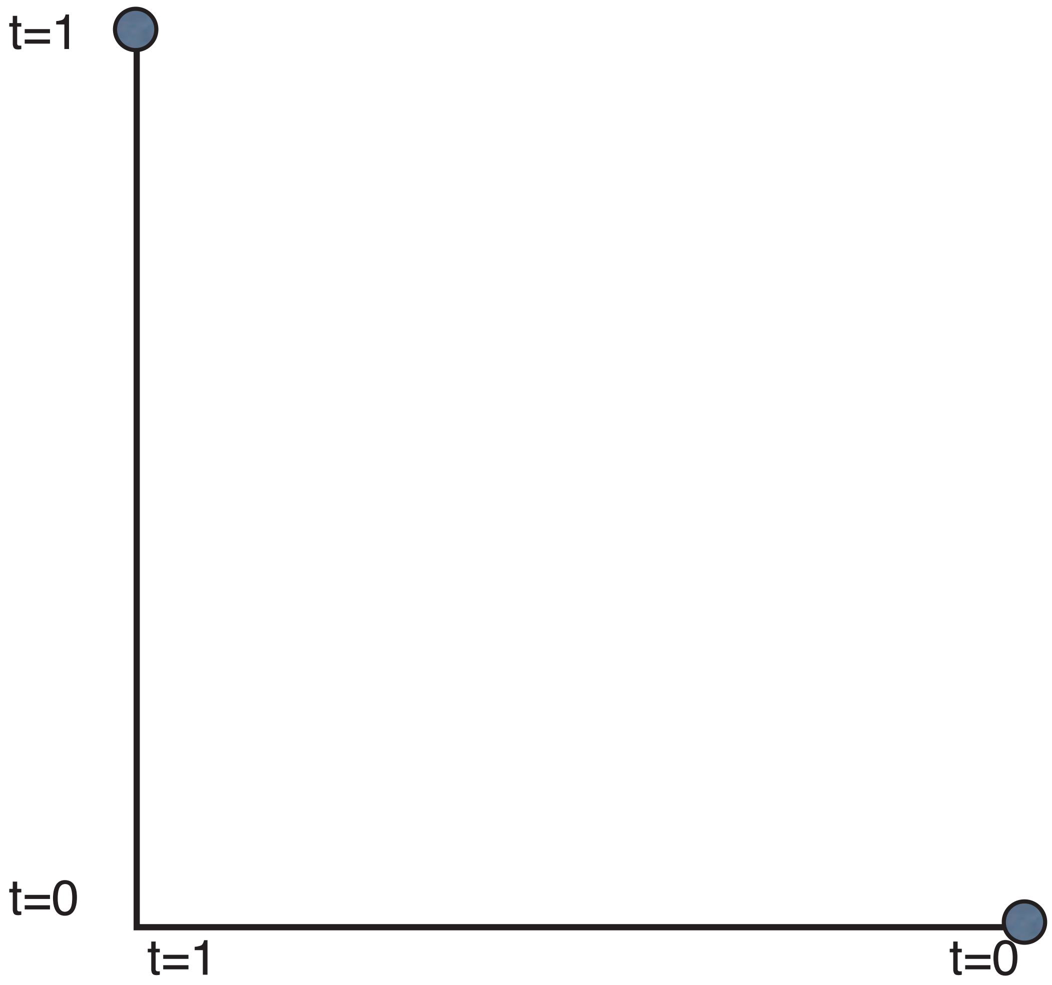
Curves from Lines



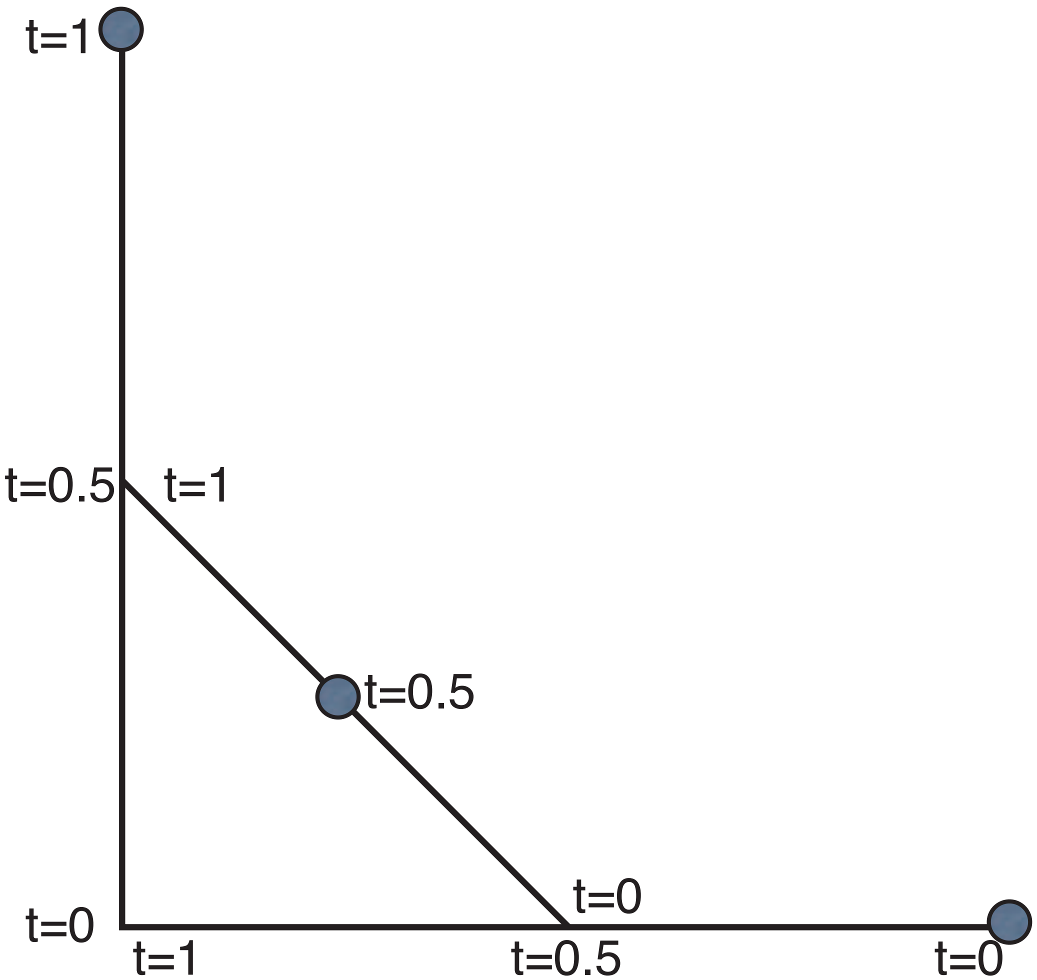
Properties

- All we need is linear interpolation
- Curve is *contained* by original points
- Curve *built* up of small segments
  - in the limit, of individual points
- But lines underneath are superfluous
- And we want to parameterize it in t

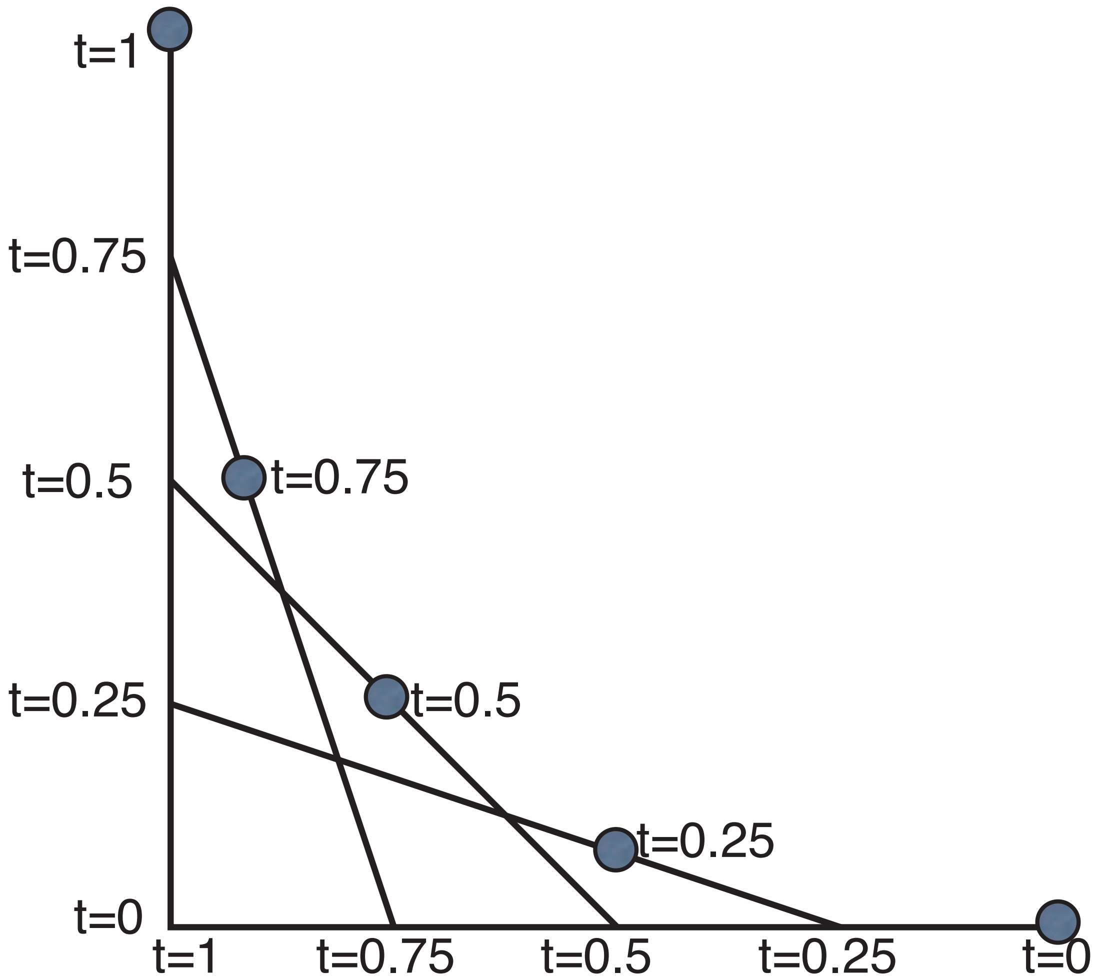
Parametrization



Parametrization



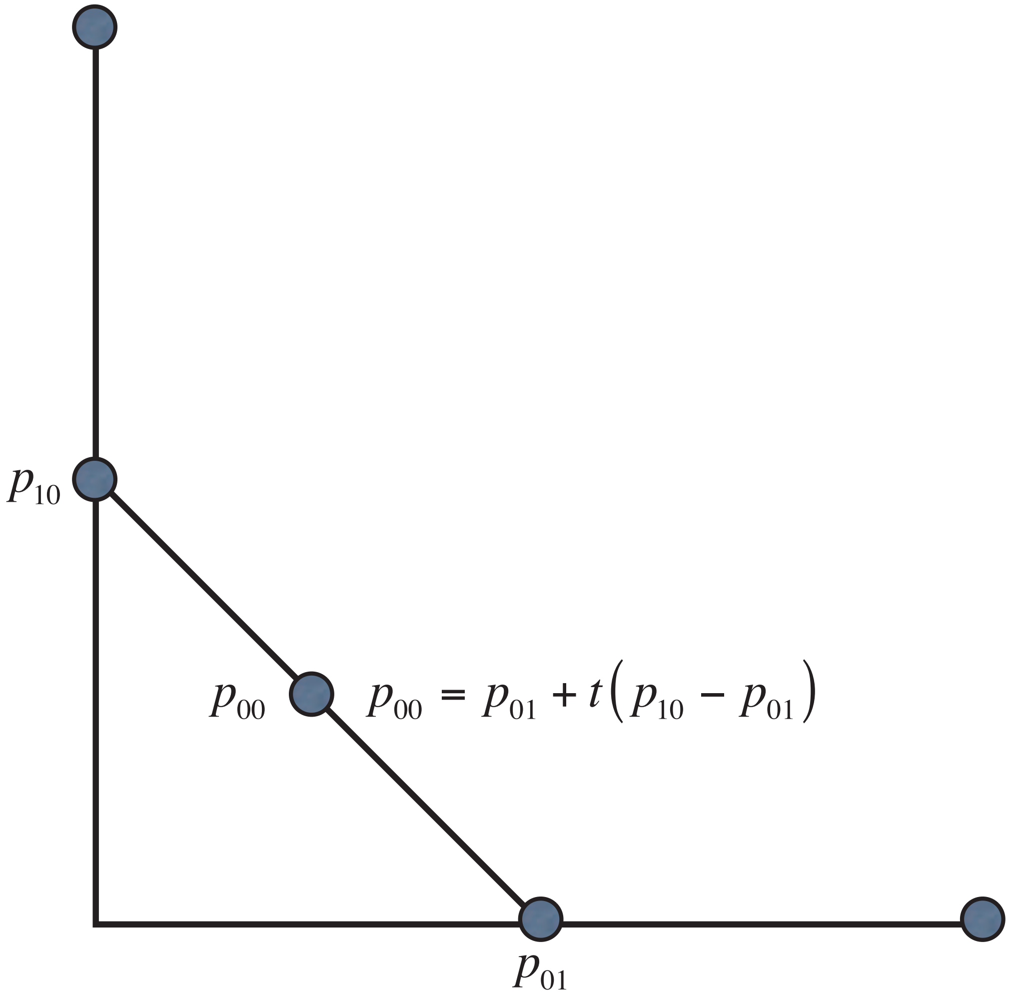
Parametrization



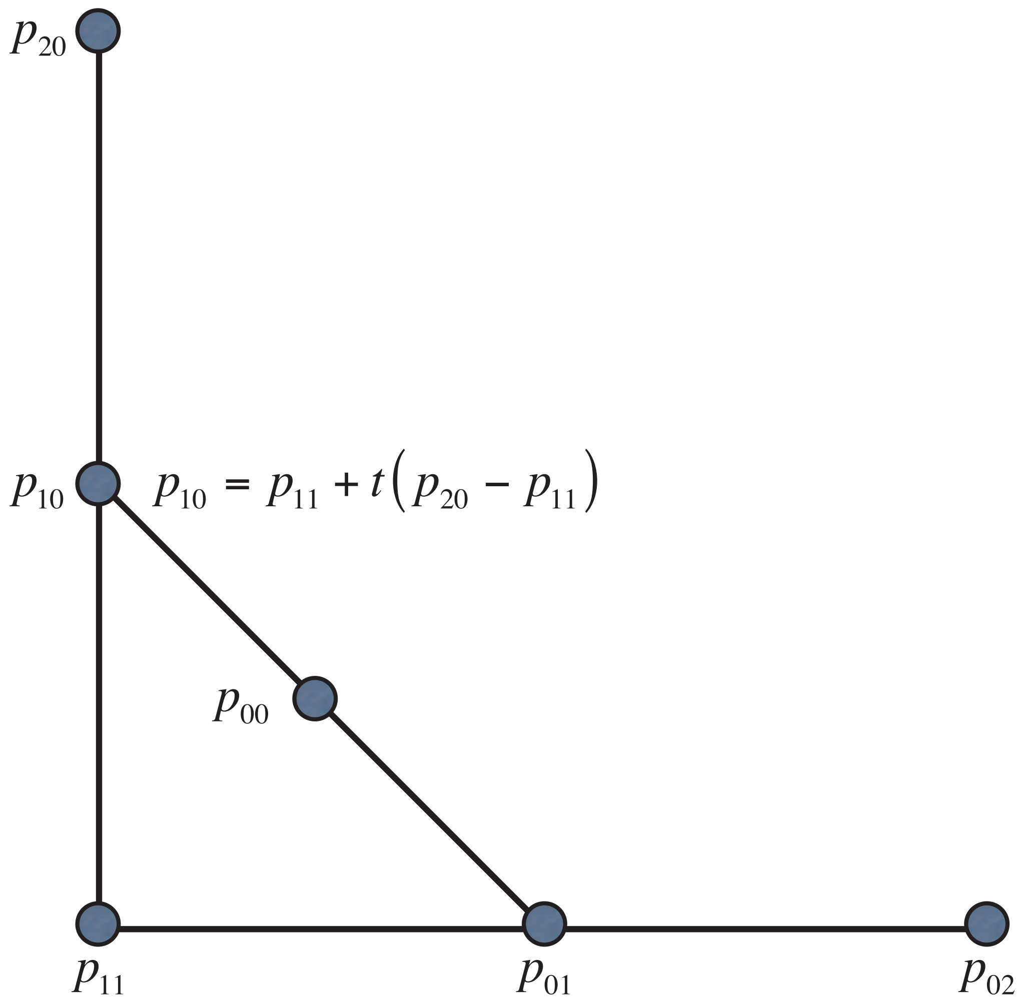
In the limit

- We take *one* point from each line
- For a given t
  - Interpolate along original edges
  - Then along the next edge
  - Repeat until we have a single point

Development



Development



$$p_{10} = p_{11} + t\left(p_{20} - p_{11}\right)$$

$$p_{01} = p_{02} + t\left(p_{11} - p_{02}\right)$$

Algebra

$$p_{10} = p_{11} + t\left(p_{20} - p_{11}\right) = \left(1 - t\right)p_{11} + t\left(p_{20}\right)$$

$$p_{01} = p_{02} + t\left(p_{11} - p_{02}\right) = \left(1 - t\right)p_{02} + t\left(p_{11}\right)$$

$$p_{00} = p_{01} + t\left(p_{10} - p_{01}\right) = \left(1 - t\right)p_{01} + t\left(p_{10}\right)$$

$$= \left(1 - t\right)\left(\left(1 - t\right)p_{02} + t\left(p_{11}\right)\right) + t\left(\left(1 - t\right)p_{11} + t\left(p_{20}\right)\right)$$

$$= p_{02} - 2p_{02}t + p_{02}t^2 + p_{11}t - p_{11}t^2 + p_{11}t - p_{11}t^2 + p_{20}t^2$$

$$= \left(p_{02} - 2p_{11} + p_{20}\right)t^2$$

$$+\left(-2p_{02} + 2p_{11}\right)t$$

$$+p_{02}$$

$$= \begin{bmatrix} p_{02} & p_{11} & p_{20} \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^2 \\ t \\ 1 \end{bmatrix}$$

Table method

$p_{00} = (1-t)p_{01} + t(p_{10}) \xleftarrow{1-t}$ d=0	$p_{01} = (1-t)p_{02} + t(p_{11}) \xleftarrow{1-t}$ d=1	p_{02} d=2
$\uparrow t$ $p_{10} = (1-t)p_{11} + t(p_{20}) \xleftarrow{1-t}$ d=1	$\uparrow t$ p_{11} d=2	
$\uparrow t$ p_{20} d=2		

In general

- Compute diagonals in *descending* order
- And each entry is found by:

$$p_{ij} = (1 - t)\, p_{i,j+1} + t\left(p_{i+1,j}\right) \text{ where } i + j = d$$

- We stop when we reach p_{00}
- And draw it
- Repeat for different values of t

de Casteljau Algorithm

```
int N_PTS = 3;
Point bezPoints[N_PTS][N_PTS];

void DrawBezier()
  { // DrawBezier()
  for (float t = 0.0; t <= 1.0; t += 0.01)
    { // parameter loop
    for (int diag = N_PTS-2; diag >= 0; diag--)
      { // diagonal loop
      for (int i = 0; i <= diag; i++)
         { // i loop
         int j = diag - i;
         bezPoints[i][j] = (1.0-t)*bezPoints[i][j+1] + t*bezPoints[i+1][j];
         } // i loop
      } // diagonal loop
      // set the pixel for this parameter value
      SetPixel(bezPoints[0][0];
    } // parameter loop
  } // DrawBezier()
```

Bézier Curves

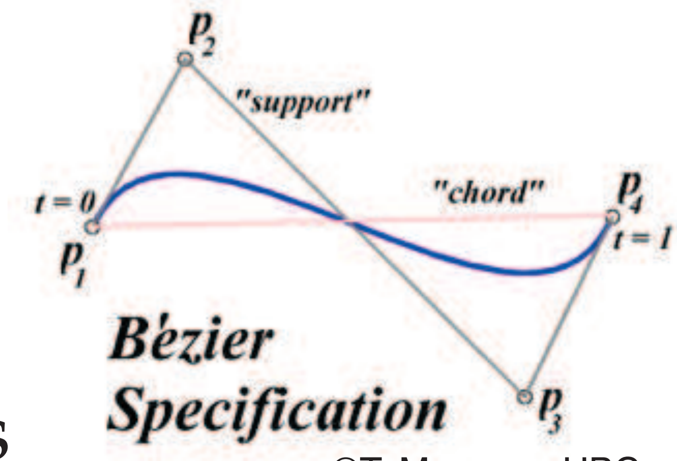
- Oldest form of computed curve
- Invented in automotive industry
- Can be computed for *any* degree
  - 2 points: line
  - 3 points: quadratic
  - 4 points: cubic

Smooth Curves

- For C^1 continuity, choose *slopes* at endpoints
  - two *slopes* + two *points* = 4 *constraints*
  - So we need (4 - 1) = 3 degree polynomials
    - i.e. *cubic* curves
- There are several ways of defining them

Cubic Bézier Curves

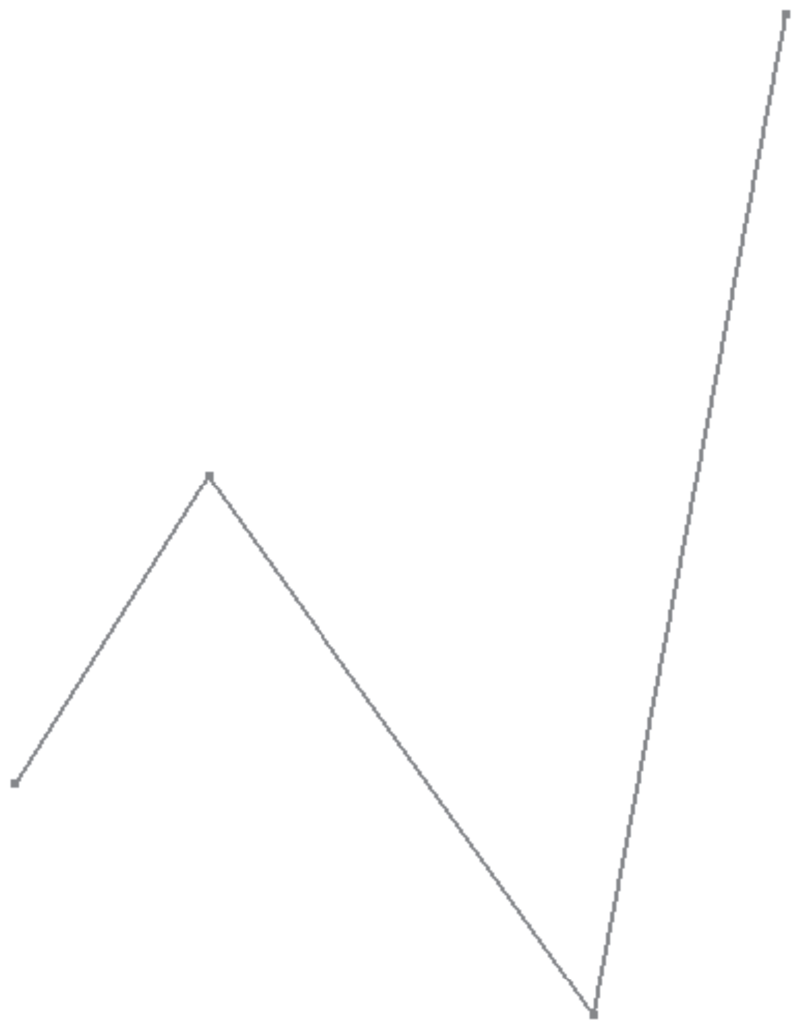
- Curves defined by *4* points
- Curve passes through two points
  - contained in *convex hull* of points



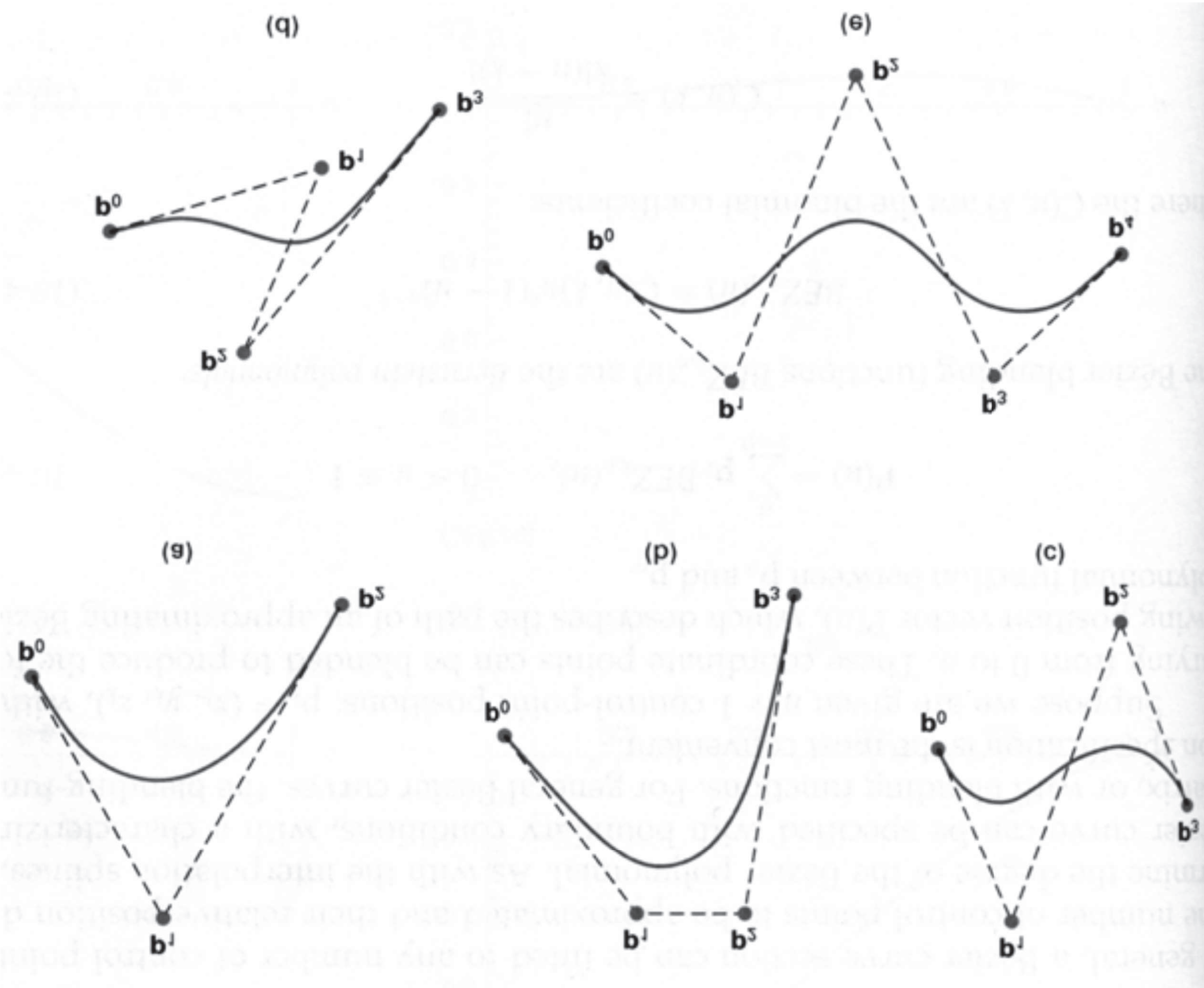
©T. Munzner, UBC

$$p_{00}(t) = \begin{bmatrix} p_{03} & p_{12} & p_{21} & p_{30} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix}$$

de Casteljau Example

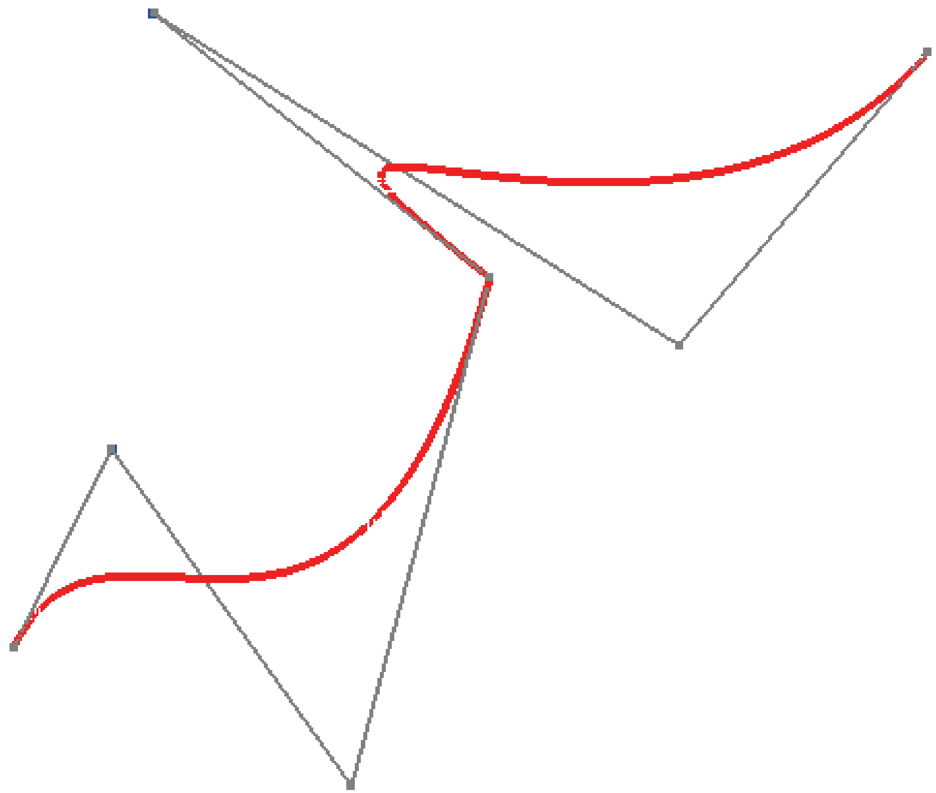


Some Examples



Piecewise Béziers

- Convenient, but
  - *not* C^1 continuous
  - *not* G^1 continuous
  - need 4 points / piece
  - we want *slopes* to match
    - rather like line strips

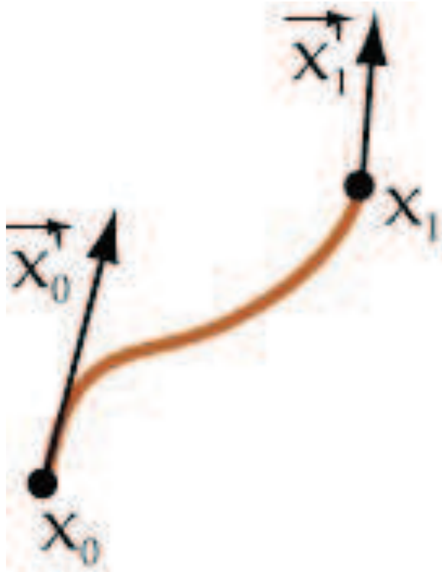


Hermite Curves

- Used in *drawing* software
  - Adobe Illustrator, &c.
  - Vectors shown as *handles*
- Not always easy to get desired result

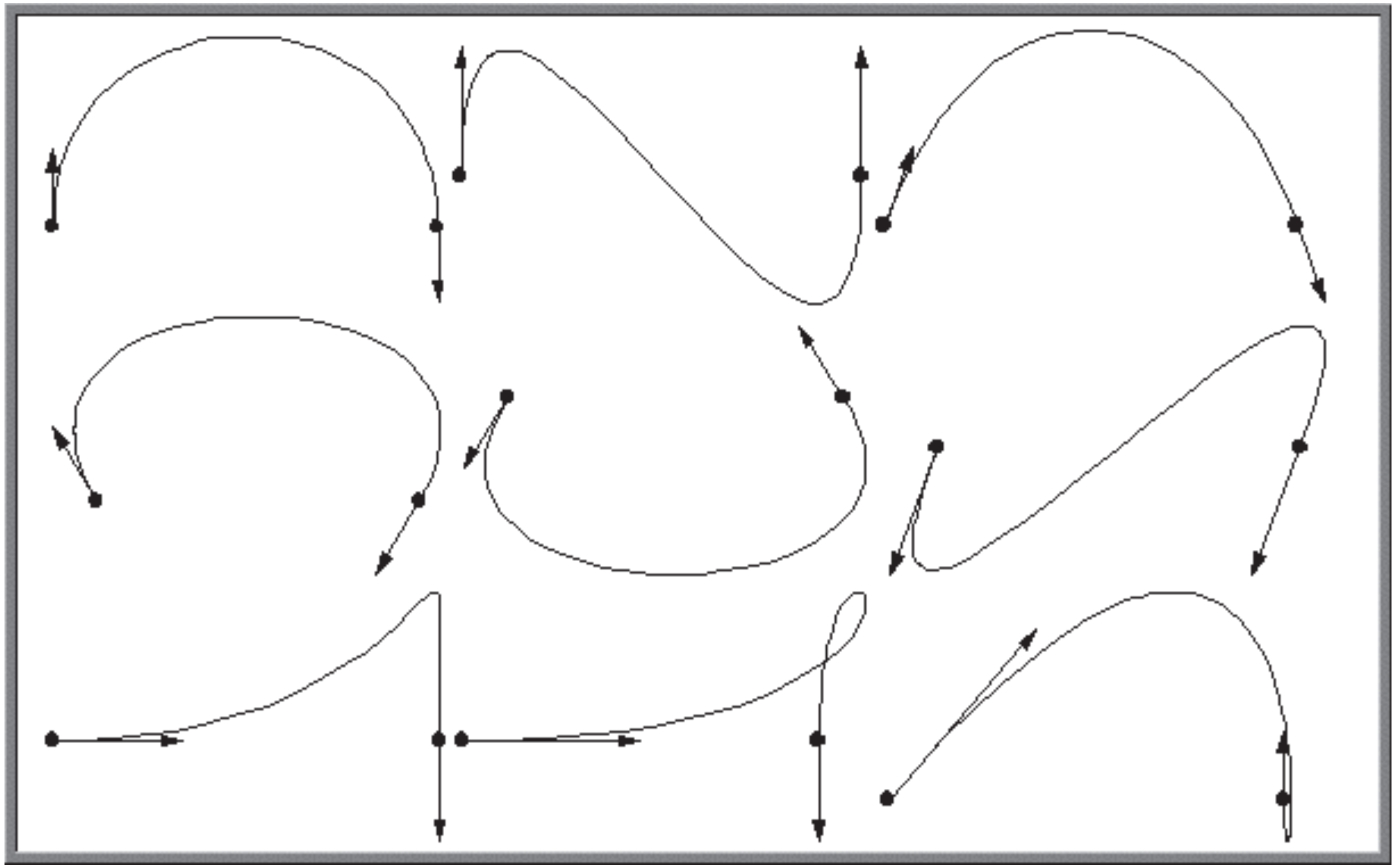
Hermite Curves

- A *Hermite* curve is given by:
  - 2 *endpoints* x_1, x_0
  - 2 *slopes* $\vec{x}_1{}', \vec{x}_0{}'$
  - Given by this equation:



$$x(t) = \begin{bmatrix} x_1 & x_0 & \vec{x}_1{}' & \vec{x}_0{}' \end{bmatrix} \begin{bmatrix} -2 & 3 & 0 & 0 \\ 2 & -3 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix}$$

Hermite Examples



©T. Munzner, UBC

Conversion

- Hermites can be converted to Béziers

$$\begin{bmatrix} p_{03} & p_{12} & p_{21} & p_{30} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_0 & \vec{x}_1' & \vec{x}_0' \end{bmatrix} \begin{bmatrix} -2 & 3 & 0 & 0 \\ 2 & -3 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} p_{03} & p_{12} & p_{21} & p_{30} \end{bmatrix} = \begin{bmatrix} x_1 & x_0 & \vec{x}_1' & \vec{x}_0' \end{bmatrix} \begin{bmatrix} -2 & 3 & 0 & 0 \\ 2 & -3 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}^{-1}$$

$$\begin{bmatrix} p_{03} & p_{12} & p_{21} & p_{30} \end{bmatrix} = \begin{bmatrix} x_1 & x_0 & \vec{x}_1' & \vec{x}_0' \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & -3 & 0 \end{bmatrix}$$

B-splines

- A *spline* is any piecewise-cubic curve
- B-splines use a different *matrix*:
  - identical to Béziers except last row

$$x(t) = \begin{bmatrix} x_{i-2} & x_{i-1} & x_i & x_{i+1} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} (t-i)^3 \\ (t-i)^2 \\ (t-i)^1 \\ 1 \end{bmatrix}$$

B-splines

- Each control point is called a *knot*
- Only need $m+3$ points for m pieces
- The pieces of the function are *uniform*
    - i.e. each piece is length 1 $(i \,..\, i+1)$
- And they are G^1 continuous

Other Curves / Surfaces

- Other types of curves / surfaces include:
  - *higher–order: quadrics*, *multi–linear*, *Gaussian*
  - *limit surfaces:* defined by iterative refinement
    - *fractals*, *subdivision surfaces*
  - *geometric* surfaces: *spheres*, *hyperbolic surfaces*
  - *contours:* defined by $\left\{ p \in \mathbb{R}^d : f\left(p\right) = h \right\}$
    - *contour lines*, *isosurfaces*, *soft (blobby) surfaces*