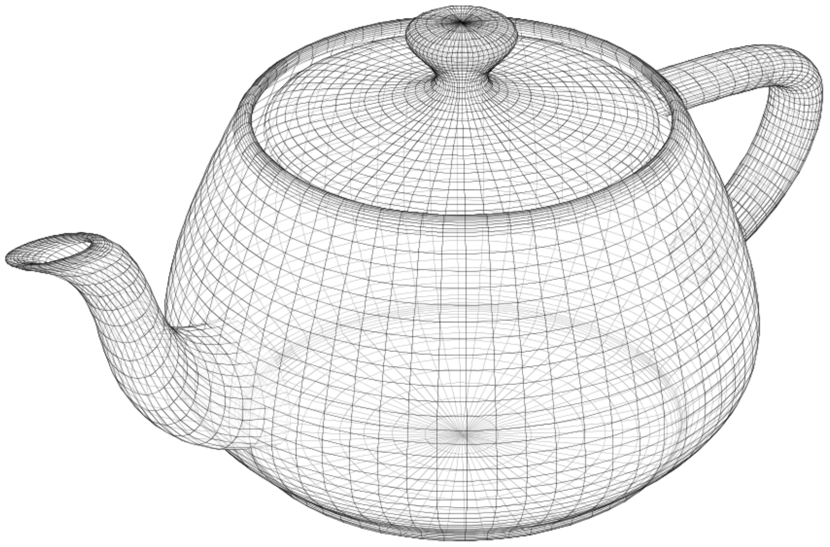
Images and some text courtesy of
The Essentials of CAGD by Farin and Hansford

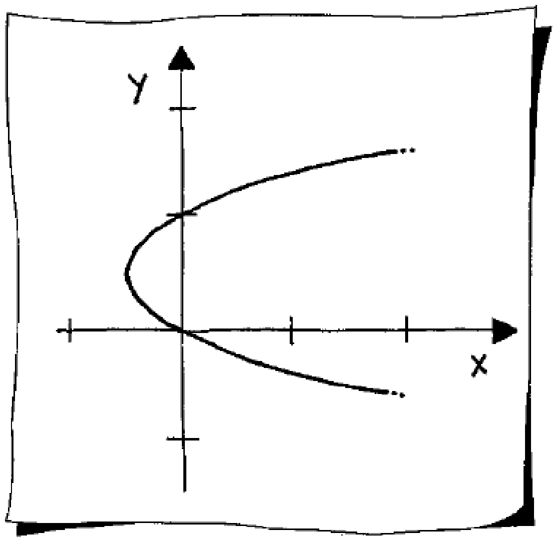


Geometric Design: Bezier Curves

Professor Eric Shaffer

ILLINOIS

Geometric Modeling



We will finish the semester by briefly looking at some math for modeling

Geometric modeling is typically done by engineers and artists
- Assisted by computational tools (e.g. Maya or Blender or AutoCAD)
- The software provides a mathematical models of curves/surfaces

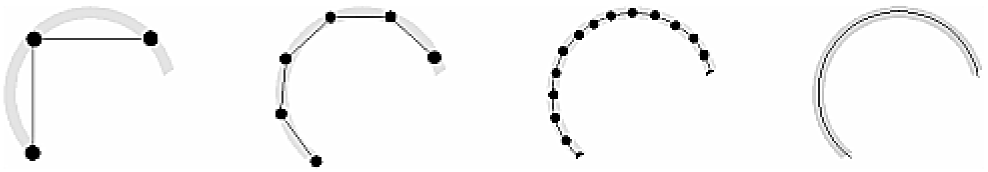
For rendering, ultimately everything will be turned into triangles.

But modeling triangle-by-triangle would be too tedious

ILLINOIS

Modeling Curves – Some Questions

Suppose we render curves by approximating them with line segments

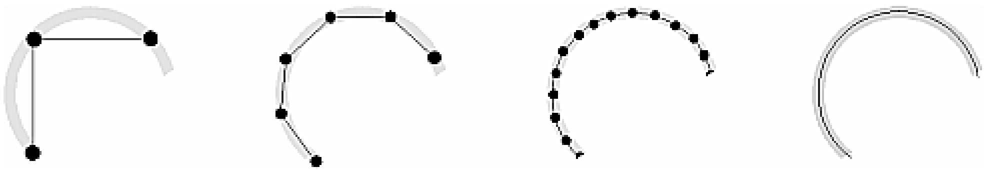


How can we can generate points on a curve…let's try to do it for a parabola

What would be one possible parametric equation for a simple parabola $y = x^2$?

Modeling Curves – Some Questions

Suppose we render curves by approximating them with line segments



How can we can generate points on a curve…let's try to do it for a parabola

What would be one possible parametric equation for a simple parabola $y = x^2$?

$$P(t) = \begin{bmatrix} t \\ t^2 \end{bmatrix}$$

We could generate a bunch of line segments using the parametric equation.
What advantages does storing/representing the curve as the equation have over storing the line segments?

Modeling Curves – Some Questions

How can we can generate points on a curve…let's try to do it for a parabola

What would be one possible parametric equation for a simple parabola $y = x^2$?

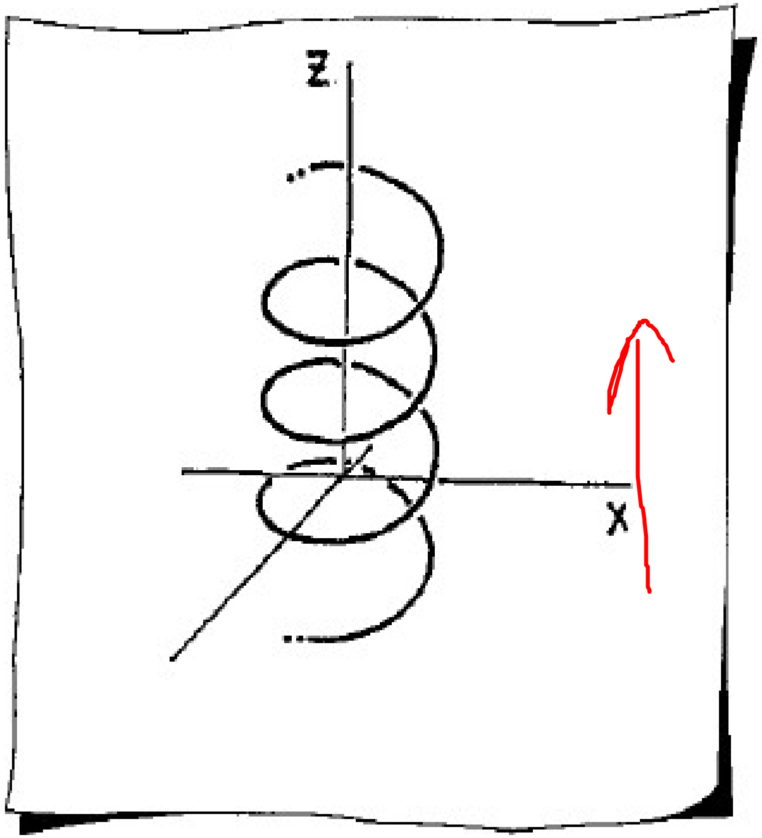
$$P(t) = \begin{bmatrix} t \\ t^2 \end{bmatrix}$$

We could generate a bunch of line segments using the parametric equation.
What advantages does storing/representing the curve as the equation have over storing the line segments?

- More compact
- Infinite resolution
- Some tasks are easier
  e.g. finding derivatives or deforming the geometry

ILLINOIS

Parametric Curves



Parametric curves defined in 3D:

$$\mathbf{x}(t) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f(t) \\ g(t) \\ h(t) \end{bmatrix}$$

Simple example: a *helix*

$$\mathbf{x}(t) = \begin{bmatrix} \cos(t) \\ \sin(t) \\ t \end{bmatrix}$$

Bezier Curves

Type of polynomial curve

Curve is defined by a modeler (artist) specifying control points

Can be defined to generate a polynomial of any degree

- Cubics are most common
- Higher degree curve requires more control points

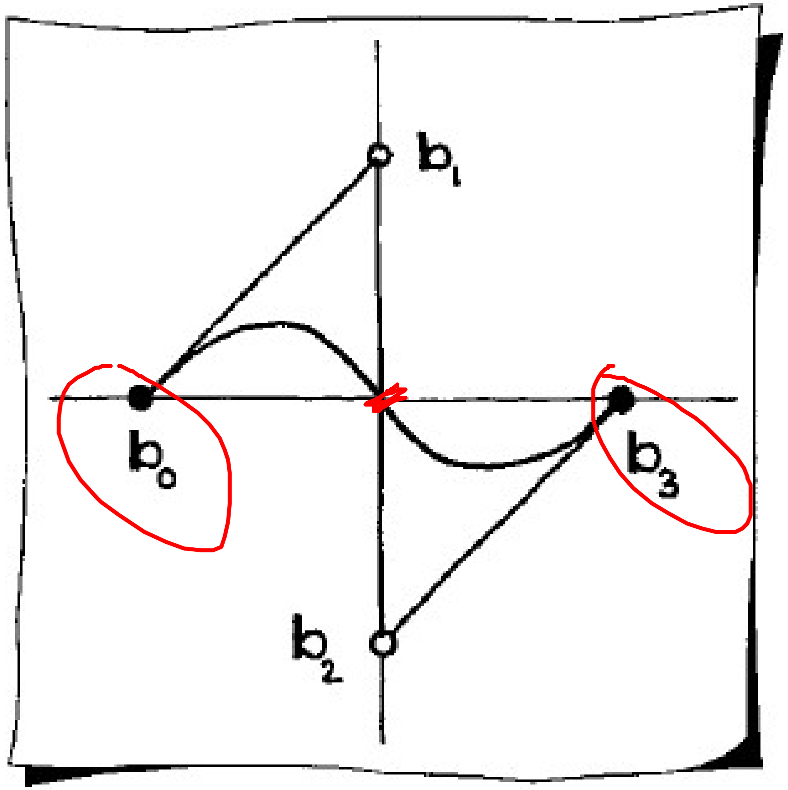
Can be joined together to form piecewise polynomial curves

Can form the basis of Bezier patches which define a surface

Named after Pierre Bezier
French Mechanical Engineer
Worked for Renault
Lived 1910-1999

Cubic Bezier Curves

The b_i are control points that an artist picks
In this example they are (-1,0), (0,1), (0,-1) and (1,0)



$$\mathbf{x}(t) = \begin{bmatrix} -(1-t)^3 + t^3 \\ 3(1-t)^2 t - 3(1-t)t^2 \end{bmatrix}$$

Shape?
Rewrite as a combination of points

$$\mathbf{x}(t) = (1-t)^3 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + 3(1-t)^2 t \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 3(1-t)t^2 \begin{bmatrix} 0 \\ -1 \end{bmatrix} + t^3 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Four points form a polygon
– Resembles curve for $t \in [0,1]$

Cubic Bezier Curves

Define a cubic Bézier curve by

$$\mathbf{x}(t) = (1-t)^3\mathbf{b}_0 + 3(1-t)^2 t\mathbf{b}_1 + 3(1-t)t^2\mathbf{b}_2 + t^3\mathbf{b}_3$$

2D or 3D points $\mathbf{b}_i$ are the Bézier control points
Control points form the Bézier polygon of the curve
Also written as

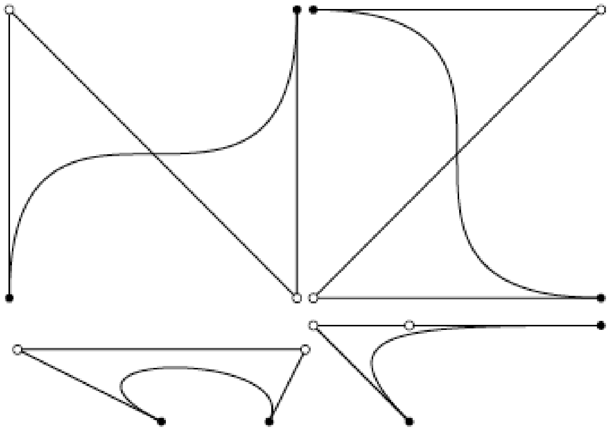
$$\mathbf{x}(t) = B_0^3(t)\mathbf{b}_0 + B_1^3(t)\mathbf{b}_1 + B_2^3(t)\mathbf{b}_2 + B_3^3(t)\mathbf{b}_3$$

B_i^3 are called the cubic Bernstein polynomials
The $\mathbf{b}_i$ are called the coefficients of the polynomial $\mathbf{x}(t)$

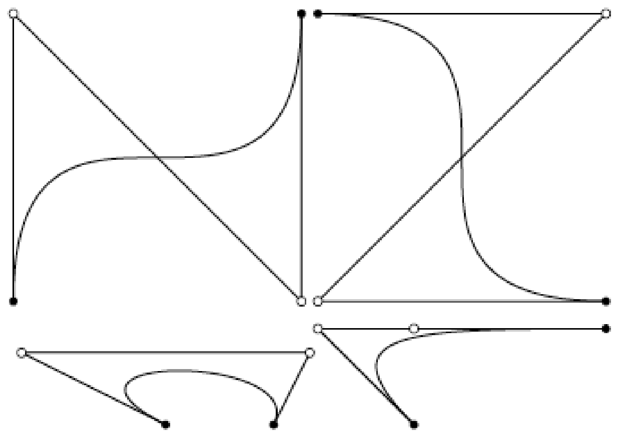
Bezier Curves

Important Properties of Bezier Curves

- Endpoint Interpolation
- Symmetry
- Invariance under affine transformations
- Convex hull property
- Linear precision



Properties of Bezier Curves



Endpoint Interpolation

The curve will pass through the first and last control points:

$x(0.0) = b_0$

$x(1.0) = b_3$

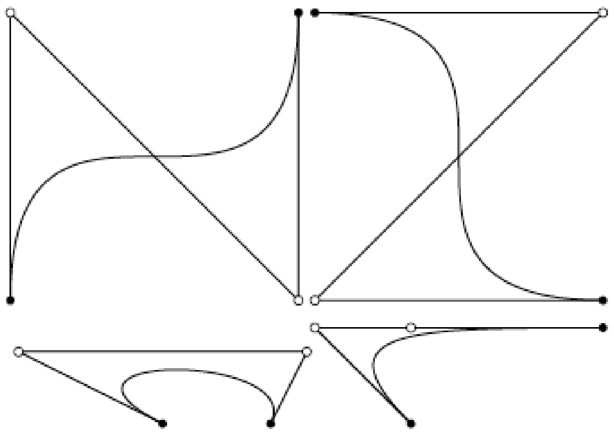
Symmetry

Specifying contol points in order b_0,b_1,b_2,b_3 generates same curve as the order b_3,b_2,b_1,b_0

Properties of Bezier Curves

Invariance under affine transformations

Transforming the control polygon similarly transforms the curve



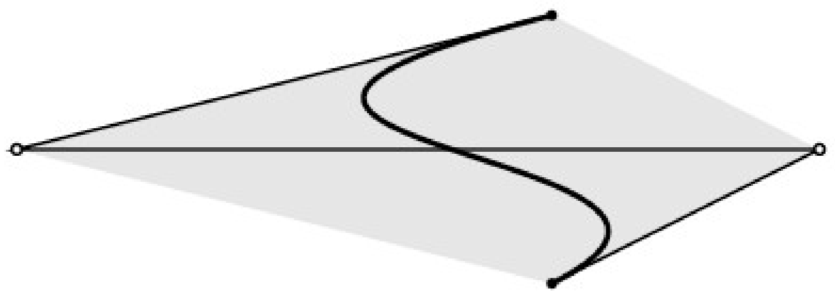
Linear Precision

If b_1 and b_2 are evenly spaced on a straight line,

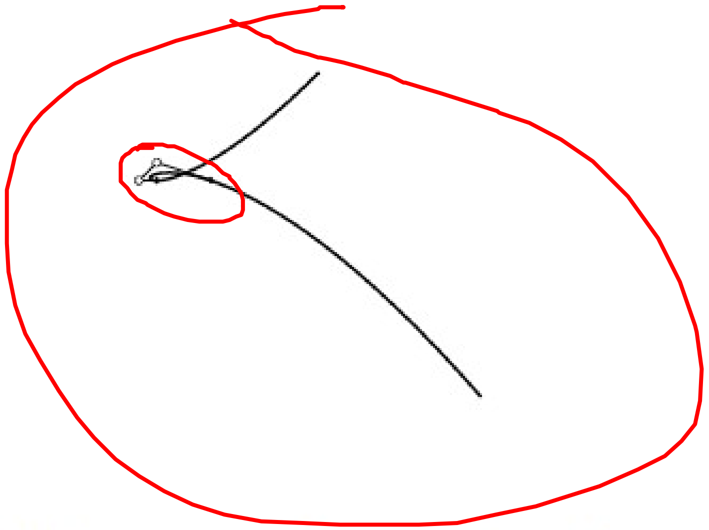
the cubic Bezier curve will be the linear interpolant between b_0 and b_3

ILLINOIS

Properties of Bezier Curves



The convex hull property



A Bézier curve for $t \in [-1, 2]$

Extrapolation: t outside $[0, 1]$
– Curve not within convex hull (in general)
– Unpredictable behavior

Derivatives

Differentiate each component with respect $t \Rightarrow$ the tangent vector

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = -3(1-t)^2\mathbf{b}_0 + [3(1-t)^2 - 6(1-t)t]\mathbf{b}_1 + [6(1-t)t - 3t^2]\mathbf{b}_2 + 3t^2\mathbf{b}_3$$

Group like terms

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = 3[\mathbf{b}_1 - \mathbf{b}_0](1-t)^2 + 6[\mathbf{b}_2 - \mathbf{b}_1](1-t)t + 3[\mathbf{b}_3 - \mathbf{b}_2]t^2$$

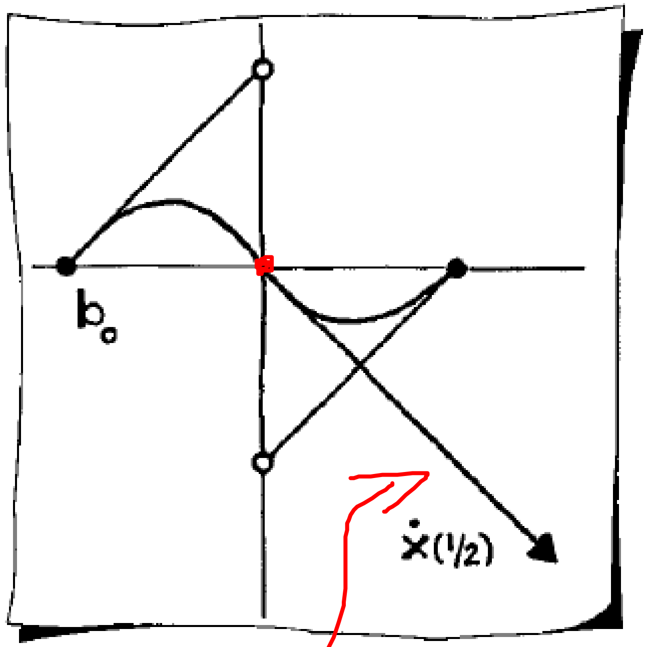
Abbreviated as

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = 3\Delta\mathbf{b}_0(1-t)^2 + 6\Delta\mathbf{b}_1(1-t)t + 3\Delta\mathbf{b}_2 t^2$$

where $\Delta\mathbf{b}_i$ is known as the forward difference

Shorten notation: $\dot{\mathbf{x}}(t) \equiv \mathrm{d}\mathbf{x}(t)/\mathrm{d}t$

Derivatives



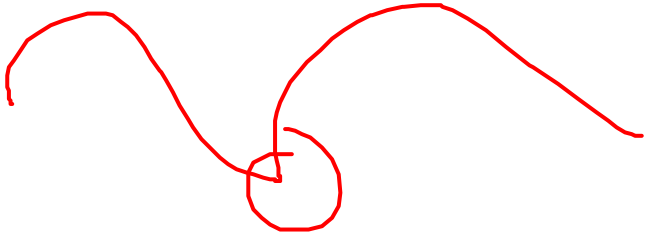
Example

$$\mathbf{x}(t) = (1-t)^3 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + 3(1-t)^2 t \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 3(1-t)t^2 \begin{bmatrix} 0 \\ -1 \end{bmatrix} + t^3 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

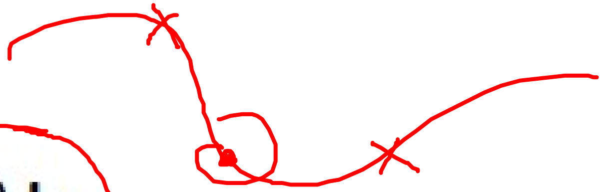
$$\dot{\mathbf{x}}(t) = 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} (1-t)^2 + 6 \begin{bmatrix} 0 \\ -2 \end{bmatrix} (1-t)t + 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} t^2$$

$$\dot{\mathbf{x}}(0.5) = \begin{bmatrix} 1.5 \\ -1.5 \end{bmatrix}$$

Piecing Curves Together



Tangent vectors at the curve's endpoints:

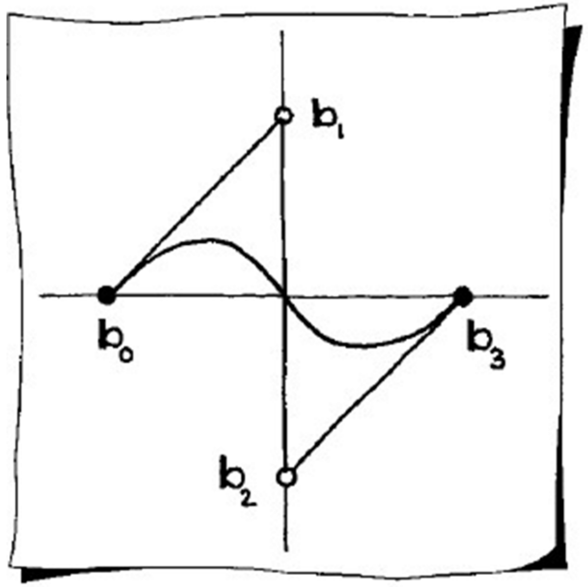


$$\dot{\mathbf{x}}(0) = 3\Delta\mathbf{b}_0 \qquad \dot{\mathbf{x}}(1) = 3\Delta\mathbf{b}_2$$

$\Rightarrow$ control polygon is tangent to the curve at the endpoints

– property helps with piecing together several Bézier curves

The de Casteljau Algorithm



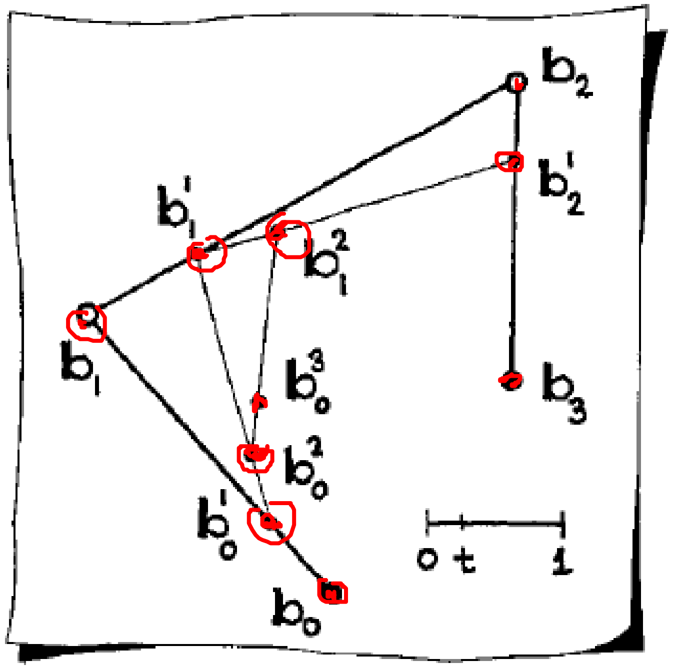
How do you generate points on a Bezier Curve?

You could just plug values for ***t*** into the formula we have already seen and evaluate ***x(t)***

The de Casteljau algorithm is an alternative way to generate points

- More computationally efficient
- Uses repeated linear interpolation
- Can be implemented recursively or interatively
- Invented by Paul de Faget de Casteljau in 1959

The de Casteljau Algorithm



Given: $\mathbf{b}_0, \ldots, \mathbf{b}_3$

and a parameter value t

Find: $\mathbf{x}(t)$

Compute:

$$\mathbf{b}_0^1 = (1-t)\mathbf{b}_0 + t\mathbf{b}_1$$
$$\mathbf{b}_1^1 = (1-t)\mathbf{b}_1 + t\mathbf{b}_2$$
$$\mathbf{b}_2^1 = (1-t)\mathbf{b}_2 + t\mathbf{b}_3$$

$$\mathbf{b}_0^2 = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1$$
$$\mathbf{b}_1^2 = (1-t)\mathbf{b}_1^1 + t\mathbf{b}_2^1$$

$$\mathbf{x}(t) = \mathbf{b}_0^3 = (1-t)\mathbf{b}_0^2 + t\mathbf{b}_1^2$$

Simply repeated linear interpolation!

The de Casteljau Algorithm

A convenient schematic tool for describing the algorithm
– Arrange the involved points in a triangular diagram

$$
\begin{array}{llll}
\mathbf{b}_0 & & & \\
\mathbf{b}_1 & \mathbf{b}_0^1 & & \\
\mathbf{b}_2 & \mathbf{b}_1^1 & \mathbf{b}_0^2 & \\
\mathbf{b}_3 & \mathbf{b}_2^1 & \mathbf{b}_1^2 & \mathbf{b}_0^3
\end{array}
$$

In the implementation of the de Casteljau algorithm:
– Not necessary to use a 2D array to simulate the triangular diagram
– A 1D array of control points is sufficient
For example $\mathbf{b}_0^1$ is calculated and loaded into $\mathbf{b}_0$
(Must save original control polygon)

The de Casteljau Algorithm

Example

$$\mathbf{x}(t) = (1-t)^3 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + 3(1-t)^2 t \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 3(1-t)t^2 \begin{bmatrix} 0 \\ -1 \end{bmatrix} + t^3 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Evaluate at $t = 0.5$

$$\begin{array}{llll}
\begin{bmatrix} -1.0 \\ 0.0 \end{bmatrix} & & & \\
\begin{bmatrix} 0.0 \\ 1.0 \end{bmatrix} & \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} & & \\
\begin{bmatrix} 0.0 \\ -1.0 \end{bmatrix} & \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} & \begin{bmatrix} -0.25 \\ 0.25 \end{bmatrix} & \\
\begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} & \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} & \begin{bmatrix} 0.25 \\ -0.25 \end{bmatrix} & \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} = \mathbf{x}(0.5)
\end{array}$$

Modeling with Cubic Bezier Curves

Lots of nice properties…
- Curvy…artistically expressive
- Only 4 control points…control polygon easy for artist to visualize and work with…
- Can be joined piecewise with matching tangents at endpoints

But…can we express any cubic as a Bezier curve? Not immediately obvious….

We can express any cubic as a sum of the monomials t^0, t^1, t^2, t^3

$$P(t) = at^3 + bt^2 + ct^1 + dt^0$$

So…let's see if we can convert between the monomial basis and the Berntsein basis

The Matrix Form and Monomials

A cubic Bézier curve:

$$\mathbf{b}(t) = B_0^3(t)\mathbf{b}_0 + B_1^3(t)\mathbf{b}_1 + B_2^3(t)\mathbf{b}_2 + B_3^3(t)\mathbf{b}_3$$

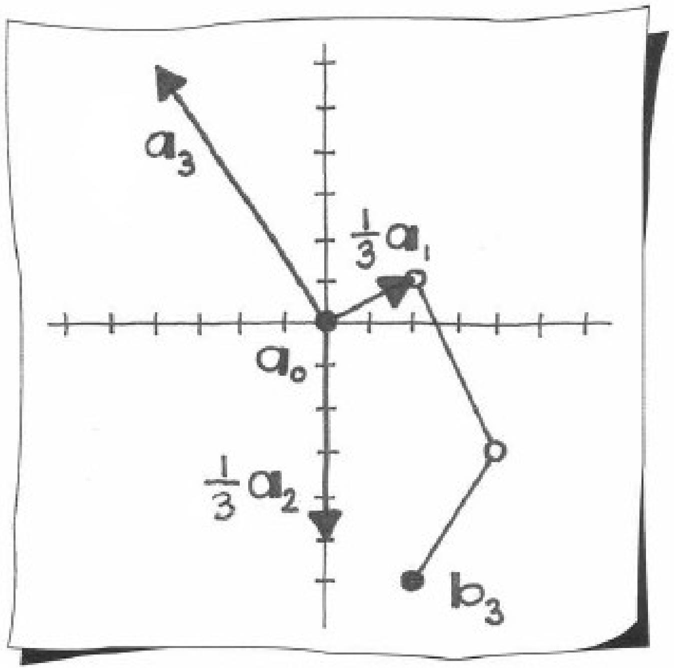
Rewritten in matrix form:

$$\mathbf{b}(t) = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \begin{bmatrix} B_0^3(t) \\ B_1^3(t) \\ B_2^3(t) \\ B_3^3(t) \end{bmatrix}$$

A more concise formulation using matrices:

$$\mathbf{b}(t) = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

The Matrix Form and Monomials



Monomial polynomials are the most familiar type

– Cubic case: $1, t, t^2, t^3$

Can reformulate a Bézier curve

$$\boxed{\begin{aligned}\mathbf{b}(t) &= \mathbf{b}_0 + 3t(\mathbf{b}_1 - \mathbf{b}_0) \\ &+ 3t^2(\mathbf{b}_2 - 2\mathbf{b}_1 + \mathbf{b}_0) \\ &+ t^3(\mathbf{b}_3 - 3\mathbf{b}_2 + 3\mathbf{b}_1 - \mathbf{b}_0)\end{aligned}}$$

$$= \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \mathbf{a}_3 t^3$$

Geometric interpretation of $\mathbf{a}_i$ and $\mathbf{b}_i$ different

The Matrix Form and Monomials

The monomial coefficients $\mathbf{a}_i$ are defined as

$$\begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Inverse process:

$$\begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

The square matrix in this equation is nonsingular

$\Rightarrow$ Any cubic curve can be written in Bézier or monomial form