

# Curves and Surfaces

COMP 3003 Autumn 2005

# Why Triangles?

- Flat
- Simple
- Cheap to process
- Interpolation is easy

# Why Not Triangles?

- Flat
- Simple
- Visible artifacts
- Above all, not *smooth*
  - More important for movies

# Closed Surfaces

- We use *surfaces* to model *objects*
  - Usually *closed* surfaces
    - i.e. *watertight*
    - prevents visual problems
- Closed surfaces are *continuous*
  - i.e. no abrupt jumps (holes)

# Surface Examples



Open  
(Leaky)



Closed  
(Watertight)



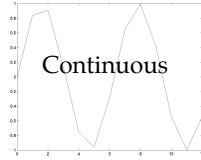
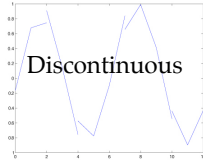
Smooth

# Smoothness

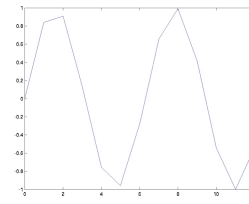
- What is *smoothness*?
- Not the same as *closed*
- Smooth surfaces are always closed
  - but not vice versa
- As usual, go back to mathematics

# Continuity

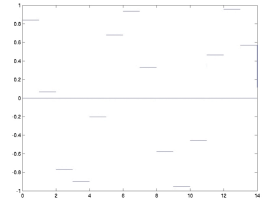
- A *continuous function*  $f(x)$  satisfies:
 
$$\lim_{x \rightarrow a^-} f(x) = f(a) = \lim_{x \rightarrow a^+} f(x)$$
- Also called  $C^0$  continuous



# Continuous $\neq$ Smooth



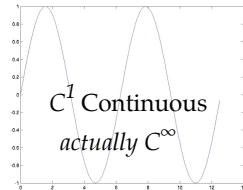
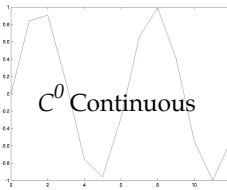
Not *smooth*  
Why not?



Slope (derivative)  
*slope is discontinuous*

# $C^n$ Continuity

- A function  $f(x)$  is  $C^n$  continuous if:
 
$$\lim_{x \rightarrow a^-} f^{(n)}(x) = f^{(n)}(a) = \lim_{x \rightarrow a^+} f^{(n)}(x)$$



# Surface Continuity

- For parameterized curves and surfaces
  - *smooth* means  $G^1$  continuity
  - curve segments meet
  - with identical tangent directions
  - $C^1$  continuous implies  $G^1$  continuous
  - *except* in one special case

# Surface Examples



Discontinuous



Continuous



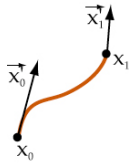
Smooth

# 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

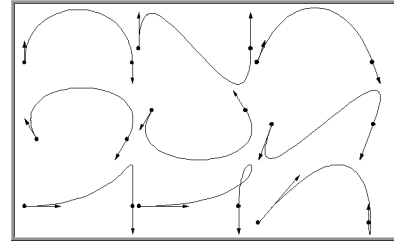
# Hermite Curves

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



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

# Hermite Examples



©T. Munzner, UBC

# Blending Functions

- *Blending functions* set weights for each constraint
- Like this:

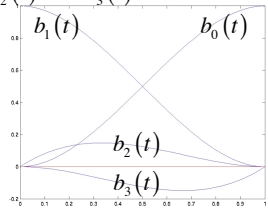
$$x(t) = b_0(t)x_1 + b_1(t)x_0 + b_2(t)\bar{x}'_1 + b_3(t)\bar{x}'_0$$

$$b_0(t) = (-2t^3 + 3t^2)$$

$$b_1(t) = (2t^3 - 3t^2 + 1)$$

$$b_2(t) = (t^3 - 2t^2 + t)$$

$$b_3(t) = (t^3 - t^2)$$

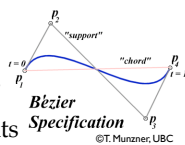


# Hermite Curves

- Used in *drawing* software
- Adobe Illustrator, &c.
- Vectors shown as *handles*
- Not always easy to get desired result
- Many people want *four* control points

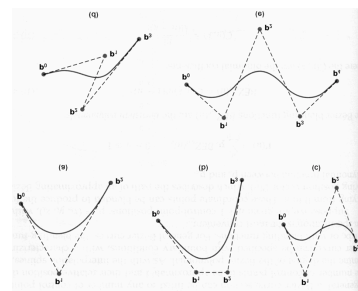
# Bézier Curves

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



$$x(t) = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix}$$

# Some Examples



©T. Munzner, UBC

## Blending Functions

- For Bézier curves, called *Bernstein polynomials*

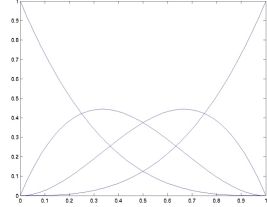
$$x(t) = b_0(t)x_0 + b_1(t)x_1 + b_2(t)x_2 + b_3(t)x_3$$

$$b_0(t) = -t^3 + 3t^2 - 3t + 1$$

$$b_1(t) = 3t^3 - 6t^2 + 3t$$

$$b_2(t) = -3t^3 + 3t^2$$

$$b_3(t) = t^3$$



## de Casteljau Algorithm

- Given parameter  $t$ :
  - Interpolate linearly between pairs of points
  - From 4 points we get 3
  - Repeat until we only have 1
- This is *exactly* the same Bézier curve!

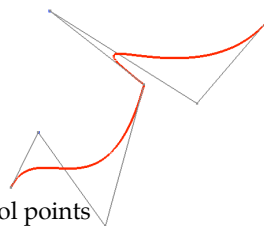
## de Casteljau Example

## Bézier Curves

- Oldest form of computed cubic curve
- Invented in automotive industry
  - Used for designing smooth curves
- Can be converted to or from Hermites
- Can be lined up *piecewise*
  - in the parameter space

## Piecewise Béziars

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



## B-splines

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

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

## 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

## B-Splines

- Knots need not be evenly spaced
  - *Non-Uniform Rational B-Splines (NURBS)*
- Splines over two parameters give surfaces
- OpenGL can compute splines for you
  - using *evaluators*
  - see the Red Book for details

## 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  $\{p \in \mathbb{R}^d : f(p) = h\}$* 
    - *contour lines, isosurfaces, soft (blobby) surfaces*

## Non-Surface Rendering

- Not all *objects* are surfaces
  - Fire, smoke, liquid, jelly, clouds
- Need other rendering techniques
  - *volume rendering - pseudo-X-rays*
  - *particle systems*
  - *physical simulation of propagated light*