Parametric Curves

Jason Lawrence
Princeton University
COS 426, Spring 2005

Curves in Computer Graphics

- Fonts ABC
- Animation paths
- Shape modeling
- etc…

Animation
(Angel, Plate 1)

Shell
(Douglas Turnbull, CS 426, Fall99)

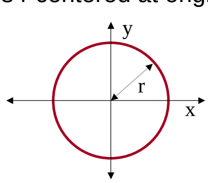
Implicit curves

An implicit curve in the plane is expressed as:

$f(x, y) = 0$

Example: a circle with radius r centered at origin:

$x^2 + y^2 - r^2 = 0$



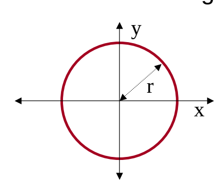
Parametric curves

A parametric curve in the plane is expressed as:

$x = f_x(u)$
$y = f_y(u)$

Example: a circle with radius r centered at origin:

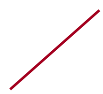
$x = r \cos u$
$y = r \sin u$



Parametric curves

How can we define arbitrary curves?

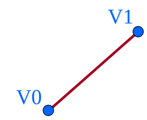
$x = f_x(u)$
$y = f_y(u)$



Parametric curves

How can we define arbitrary curves?

$x = f_x(u)$
$y = f_y(u)$



Use functions that "blend" control points

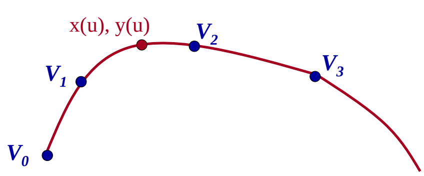
$x = f_x(u) = V0_x*(1 - u) + V1_x*u$
$y = f_y(u) = V0_y*(1 - u) + V1_y*u$

Parametric curves

More generally:

$$x(u) = \sum_{i=0}^{n} B_i(u) * Vi_x$$

$$y(u) = \sum_{i=0}^{n} B_i(u) * Vi_y$$



Parametric curves

What B(u) functions should we use?

$$x(u) = \sum_{i=0}^{n} B_i(u) * Vi_x$$

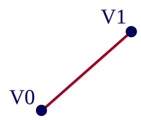
$$y(u) = \sum_{i=0}^{n} B_i(u) * Vi_y$$

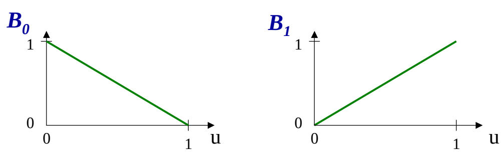
Parametric curves

What B(u) functions should we use?

$$x(u) = \sum_{i=0}^{n} B_i(u) * Vi_x$$

$$y(u) = \sum_{i=0}^{n} B_i(u) * Vi_y$$



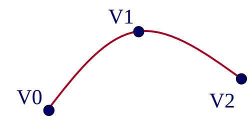


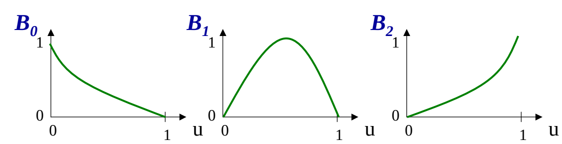
Parametric curves

What B(u) functions should we use?

$$x(u) = \sum_{i=0}^{n} B_i(u) * Vi_x$$

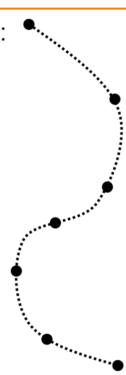
$$y(u) = \sum_{i=0}^{n} B_i(u) * Vi_y$$





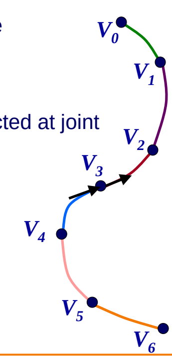
Goals

- Some attributes we might like to have:
  - Interpolation
  - Continuity
  - Predictable control
  - Local control
- We'll satisfy these goals using:
  - Piecewise
  - Parametric
  - Polynomials



Continuity

- Parametric continuity (C^n)
  - How many times differentiable is the curve at a given point
- Continuity at joints:
  - C^0 continuity means curve is connected at joint
  - C^1 continuity means that segments share same first derivative at joint
  - C^n continuity means that segments share same nth derivative at joint



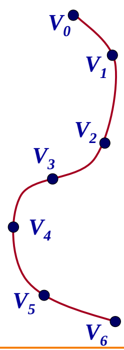
Parametric Polynomial Curves

- Blending functions are polynomials:

$$x(u) = \sum_{i=0}^{n} B_i(u) * Vi_x \qquad B_i(u) = \sum_{j=0}^{m} a_j u^j$$

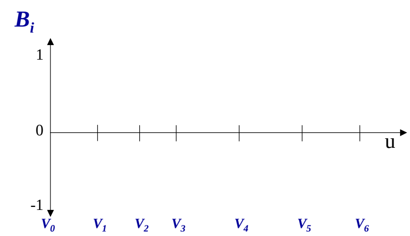
$$y(u) = \sum_{i=0}^{n} B_i(u) * Vi_y$$

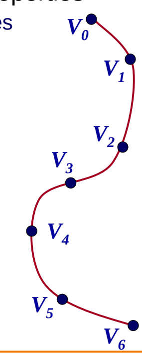
- Advantages of polynomials
  - Easy to compute
  - Infinitely continuous
  - Easy to derive curve properties



Parametric Polynomial Curves

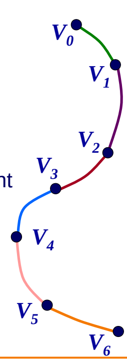
- Derive polynomial $B_i(u)$ to ensure properties
  - Example: interpolation of control vertices
  - What about easy of control?





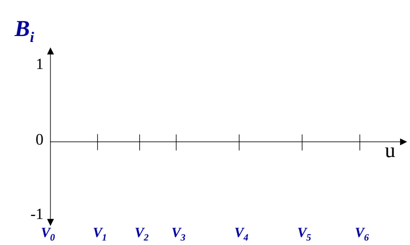
Piecewise Parametric Polynomial Curves

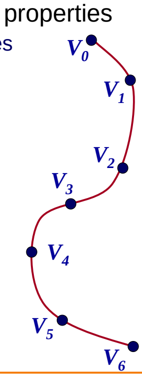
- Splines:
  - Split curve into segments
  - Each segment defined by blending subset of control vertices
- Motivation:
  - Provides control & efficiency
  - Same blending function for every segment
  - Prove properties from blending functions
- Challenges
  - How choose blending functions?
  - How guarantee continuity at joints?



Piecewise Parametric Polynomial Curves

- Compute polynomial $B_i(u)$ to ensure properties
  - Example: interpolation of control vertices and C^2 continuity at joints with cubics





Cubic Piecewise Parametric Polynomial Curves

- From now on, consider cubic blending functions
  - All ideas generalize to higher degrees
- In CAGD, higher-order functions are often used
  - Hard to control wiggles
- In graphics, piecewise cubic curves will do
  - Smallest degree that allows C^2 continuity for arbitrary curves

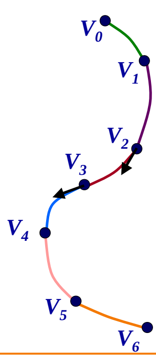
Types of Splines

- Splines covered in this lecture
  - Hermite
  - Bezier
  - Catmull-Rom
  - B-Spline
- There are many others

Each has different blending functions resulting in different properties

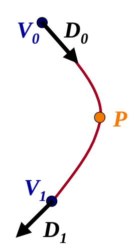
Cubic Hermite Splines

- Definition:
  - Each segment defined by position and derivative at two adjacent control vertices
  - Blending functions are cubic polynomials
- Properties:
  - Interpolates control points
  - C^1 continuity at joints



Cubic Hermite Splines

- Definition:
  - Each segment defined by position and derivative at two adjacent control vertices
  - Blending functions are cubic polynomials
- Properties:
  - Interpolates control points
  - C^1 continuity at joints

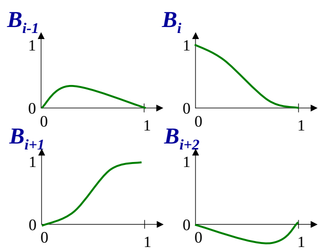


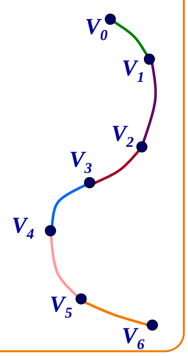
$$P(u) = B_0(u)*D_0 + B_1(u)*V_0 + B_2(u)*V_1 + B_3(u)*D_1$$

Cubic Hermite Splines

Blending functions:

$$B_i(u) = \sum_{j=0}^{m} a_j u^j$$





Types of Splines

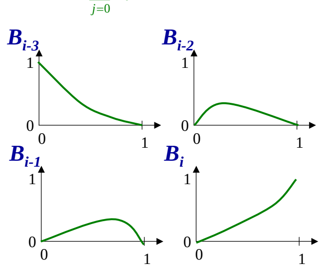
- Splines covered in this lecture
  - Hermite
  - Bezier
  - Catmull-Rom
  - B-Spline
- There are many others

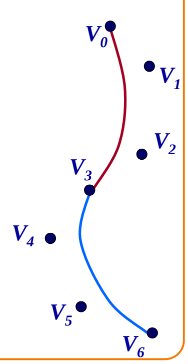
Each has different blending functions resulting in different properties

Bezier curves

Blending functions:

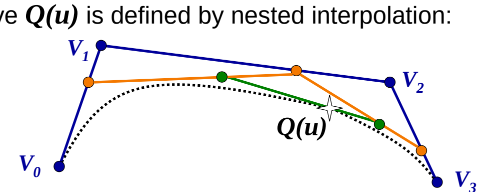
$$B_i(u) = \sum_{j=0}^{m} a_j u^j$$





Bézier curves

- Developed simultaneously in 1960 by
  - Bézier (at Renault)
  - deCasteljau (at Citroen)
- Curve ***Q(u)*** is defined by nested interpolation:



V_i's are *control points*
$\{V_0, V_1, \ldots, V_n\}$ is *control polygon*

Basic properties of Bézier curves

- Endpoint interpolation:

$$Q(0) = V_0$$
$$Q(1) = V_n$$

- Convex hull:
  - Curve is contained within convex hull of control polygon
- Symmetry

$$Q(u) \text{ defined by } \{V_0,\ldots,V_n\} \equiv Q(1-u) \text{ defined by } \{V_n,\ldots,V_0\}$$

Explicit formulation

- Let's indicate level of nesting with superscript j:
- An explicit formulation of ***Q(u)*** is given by:

$$V_i^j = (1-u)V_i^{j-1} + uV_{i+1}^{j-1}$$

- Case n=3 (expand recurrence):

$$\begin{aligned} Q(u) &= V_0^3 \\ &= (1-u)V_0^2 + uV_1^2 \\ &= (1-u)[(1-u)V_0^1 + uV_1^1] + u[(1-u)V_1^1 + uV_2^1] \\ &= (1-u)[(1-u)[(1-u)V_0^0 + uV_1^0] + u[(1-u)V_1^0 + uV_2^0] \ldots\ldots \end{aligned}$$

More properties

- General case: Bernstein polynomials

$$Q(u) = \sum_{i=0}^{n} V_i \binom{n}{i} u^i (1-u)^{n-i}$$

- Degree: is a polynomial of degree n
- Tangents:

$$Q'(0) = n(V_1 - V_0)$$
$$Q'(1) = n(V_n - V_{n-1})$$

Matrix form

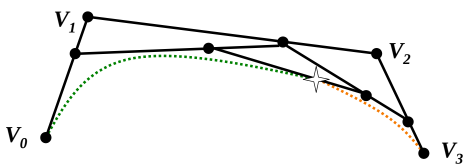
Bézier curves may be described in matrix form:

$$\begin{aligned} Q(u) &= \sum_{i=0}^{n} V_i \binom{n}{i} u^i (1-u)^{n-i} \\ &= (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u) V_2 + u^3 V_3 \\ &= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \underbrace{\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}}_{\mathbf{M}_{\text{Bezier}}} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} \end{aligned}$$

Display

Q: How would you draw it using line segments?

A: Recursive subdivision!



Display

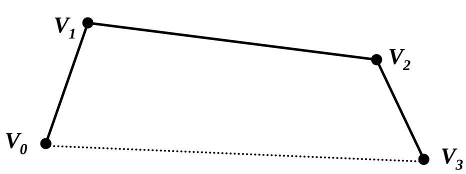
Pseudocode for displaying Bézier curves:

```
procedure Display({Vi}):
    if {Vi} flat within ε
    then
        output line segment V0Vn
    else
        subdivide to produce {Li} and {Ri}
        Display({Li})
        Display({Ri})
    end if
end procedure
```

Flatness

Q: How do you test for flatness?

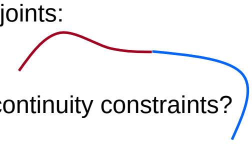
A: Compare the length of the control polygon to the length of the segment between endpoints



$$\frac{|V_1 - V_0| + |V_2 - V_1| + |V_3 - V_2|}{|V_3 - V_0|} < 1 + \varepsilon$$

Splines

- For more complex curves, piece together Béziers
- We want continuity across joints:
  - Positional (C^0) continuity
  - Derivative (C^1) continuity



- Q: How would you satisfy continuity constraints?
- Q: Why not just use higher-order Bézier curves?
- A: Splines have several of advantages:
  - Numerically more stable
  - Easier to compute
  - Fewer bumps and wiggles

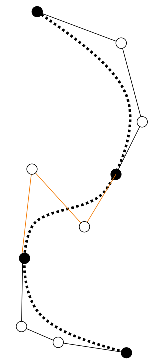
Types of Splines

- Splines covered in this lecture
  - Hermite
  - Bezier
  - ➢ Catmull-Rom
  - B-Spline
- There are many others

Each has different blending functions resulting in different properties

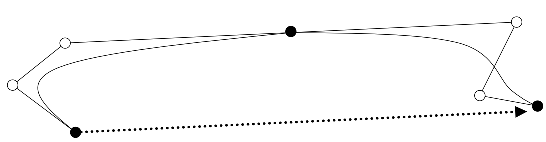
Catmull-Rom splines

- Properties
  - Interpolate control points
  - Have C^0 and C^1 continuity
- Derivation
  - Start with joints to interpolate
  - Build cubic Bézier between each joint
  - Endpoints of Bézier curves are obvious
- What should we do for the other Bézier control points?



Catmull-Rom Splines

- Catmull & Rom use:
  - half the magnitude of the vector between adjacent CP's



- Many other formulations work, for example:
  - Use an arbitrary constant τ times this vector
  - Gives a "tension" control
  - Could be adjusted for each joint

Properties

- Catmull-Rom splines have these attributes:
  - C1 continuity
  - Interpolation
  - Locality of control
  - No convex hull property

  (Proof left as an exercise.)

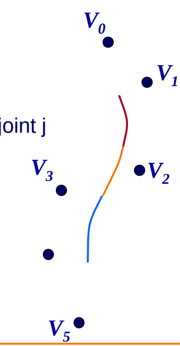
Types of Splines

- Splines covered in this lecture
  - Hermite
  - Bezier
  - Catmull-Rom
  - B-Spline
- There are many others

Each has different blending functions resulting in different properties

B-Splines

- Properties:
  - Local control
  - C^2 continuity
  - Cubic polynomials
- Constraints:
  - Three continuity conditions at each joint j
    - Position of two curves same
    - Derivative of two curves same
    - Second derivatives same
  - Local control
    - Each joint affected by 4 control vertices
- Give up interpolation :)



Matrix formulation for B-splines

- List mathematical constraints:

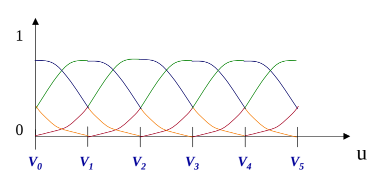
$$Q_i(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} M_{BSPLINE} \begin{pmatrix} V_{i-3} \\ V_{i-2} \\ V_{i-1} \\ V_i \end{pmatrix} \qquad \begin{aligned} Q_i(1) &= Q_{i+1}(0) \\ Q_i'(1) &= Q_{i+1}'(0) \\ Q_i''(1) &= Q_{i+1}''(0) \end{aligned}$$

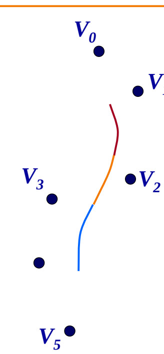
- Grind through some messy math to get:

$$M_{BSPLINE} = \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}$$

B-Splines

- Blending functions:
  - Local control: how can we tell?
  - Interpolates control points?



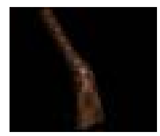


Summary

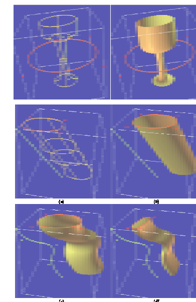
- Splines: mathematical way to express curves
- Motivated by “loftsman’s spline”
  - Long, narrow strip of wood/plastic
  - Used to fit curves through specified data points
  - Shaped by lead weights called “ducks”
  - Gives curves that are “smooth” or “fair”
- Have been used to design:
  - Automobiles
  - Ship hulls
  - Aircraft fuselage/wing

What’s next?

- Use curves to create parameterized surfaces
- Surface of revolution
- Swept surfaces
- Surface patches



Przemyslaw Prusinkiewicz



Demetri Terzopoulos