Lecture Notes #9 - Curves

Reading:

Angel: Chapter 9

Foley et al., Sections 11(intro) and 11.2

Overview

Introduction to mathematical splines

Bezier curves

Continuity conditions (C^0, C^1, C^2, G^1, G^2)

Creating continuous splines

C^2 interpolating splines

B-splines

Catmull-Rom splines

Introduction

Mathematical splines are motivated by the "loftsman's spline":

- Long, narrow strip of wood or plastic
- Used to fit curves through specified data points
- Shaped by lead weights called "ducks"
- Gives curves that are "smooth" or "fair"

Such splines have been used for designing:

- Automobiles
- Ship hulls
- Aircraft fuselages and wings

Requirements

Here are some requirements we might like to have in our mathematical splines:

- Predictable control
- Multiple values
- Local control
- Versatility
- Continuity

Mathematical splines

The mathematical splines we'll use are:

- Piecewise
- Parametric
- Polynomials

Let's look at each of these terms......

Parametric curves

In general, a "parametric" curve in the plane is expressed as:

$$x = x(\mathrm{t})$$

$$y = y(t)$$

<u>Example:</u> A circle with radius r centered at the origin is given by:

$$x = r \cos t$$

$$y = r \sin t$$

By contrast, an "implicit" representation of the circle is:

Parametric polynomial curves

A parametric "polynomial" curve is a parametric curve where each function x(t), y(t) is described by a polynomial:

$$x(t) = \sum_{i=0}^{n} a_i t^i$$

$$y(t) = \sum_{i=0}^{n} b_i t^i$$

Polynomial curves have certain advantages:

- Easy to compute
- Infinitely differentiable

Piecewise parametric polynomial curves

A "piecewise" parametric polynomial curve uses different polynomial functions for different parts of the curve.

- **Advantage:** Provides flexibility
- **Problem:** How do you guarantee smoothness at the joints? (Problem known as "continuity.")

In the rest of this lecture, we'll look at:

1. Bezier curves -- general class of polynomial curves
2. Splines -- ways of putting these curves together

Bezier curves

- Developed simultaneously by Bezier (at Renault) and deCasteljau (at Citroen), circa 1960.

- The Bezier curve $Q(u)$ is defined by nested interpolation:

- V_i's are "control points"

- $\{V_0, \ldots , V_n\}$ is the "control polygon"

Bezier curves: Basic properties

Bezier curves enjoy some nice properties:

- Endpoint interpolation:

$$Q(0) = V_0$$

$$Q(1) = V_n$$

- Convex hull: The curve is contained in the convex hull of its control polygon

- Symmetry:

$$Q(u) \text{ defined by } \{V_0, ..., V_n\}$$

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

Bezier curves: Explicit formulation

Let's give V_i a superscript V_i^j to indicate the level of nesting.

An explicit formulation for $Q(u)$ is given by the recurrence:

$$V_i^j = (1 - u)\, V_i^{j-1} + u V_{i+1}^{j-1}$$

Explicit formulation, cont.

For $n = 2$, we have:

$$Q(u) = V_0^2$$

$$= (1 - u)V_0^1 + uV_1^1$$

$$= (1 - u)\,[(1 - u)\,V_0^0 + uV_1^0] + [(1 - u)\,V_1^0 + uV_2^0]$$

$$= (1 - u)^2V_0^0 + 2u(1 - u)V_1^0 + u^2V_2^0$$

In general:

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

$B_i^n(u)$ is the i'th Bernstein polynomial of degree n.

Bezier curves: More properties

Here are some more properties of Bezier curves

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

- Degree: $Q(u)$ is a polynomial of degree n
- Control points: How many conditions must we specify to uniquely determine a Bezier curve of degree n?

More properties, cont.

- Tangents:

$$Q'(0) = n(V_1 - V_0)$$

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

- *k*'th derivatives: In general,
    - $Q^{(k)}(0)$ depends only on $V_0, ..., V_k$
    - $Q^{(k)}(1)$ depends only on $V_n, ..., V_{n-k}$
    - (At intermediate points $u \in (0, 1)$, all control points are involved for every derivative.)

Cubic curves

For the rest of this discussion, we'll restrict ourselves to piecewise cubic curves.

- In CAGD, higher-order curves are often used
  - Gives more freedom in design
  - Can provide higher degree of continuity between pieces
- For Graphics, piecewise cubic let's you do just about anything
  - Lowest degree for specifiying points to interpolate and tangents
  - Lowest degree for specifying curve in space

All the ideas here generalize to higher-order curves

Matrix form of Bezier curves

Bezier curves can also be described in matrix form:

$$Q(u) = \sum_{i=0}^{3} V_i \binom{3}{i} u^i (1-u)^{3-i}$$

$$= (1 - u)^3 V_0 + 3u\,(1 - u)^2 V_1 + 3u^2 (1 - u)\, V_2 + u^3 V_3$$

$$= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

$$= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \; M_{Bezier} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

Display: Recursive subdivision

Q: Suppose you wanted to draw one of these Bezier curves -- how would you do it?

A: Recursive subdivision:

Display, cont.

Here's pseudocode for the recursive subdivision display algorithm:

procedure *Display*($\{V_0, ..., V_n\}$):

    if $\{V_0, ..., V_n\}$ flat within ε **then**

        Output line segment V_0V_n

    else

        Subdivide to produce $\{L_0, ..., L_n\}$ and $\{R_0, ..., R_n\}$

        Display($\{L_0, ..., L_n\}$)

        Display($\{R_0, ..., R_n\}$)

    end if

end procedure

Splines

To build up more complex curves, we can piece together different Bezier curves to make "splines."

For example, we can get:

- Positional (C^0) continuity:

- Derivative (C^1) continuity:

Q: How would you build an interactive system to satisfy these constraints?

Advantages of splines

Advantages of splines over higher-order Bezier curves:

- Numerically more stable
- Easier to compute
- Fewer bumps and wiggles

Tangent (G^1) continuity

Q: Suppose the tangents were in opposite directions but <u>not</u> of same magnitude -- how does the curve appear?

This construction gives "tangent (G^1) continuity."

Q: How is G^1 continuity different from C^1?

Curvature (C^2) continuity

Q: Suppose you want even higher degrees of continuity -- e.g., not just slopes but curvatures -- what additional geometric constraints are imposed?

We'll begin by developing some more mathematics.....

Operator calculus

Let's use a tool known as "operator calculus."

Define the operator D by:

$$\mathsf{D} V_i \equiv V_{i+1}$$

Rewriting our explicit formulation in this notation gives:

$$\begin{aligned} Q(u) &= \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} V_i \\ &= \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} \mathsf{D}_i \, V_0 \\ &= \sum_{i=0}^{n} \binom{n}{i} (u\mathsf{D})^i (1-u)^{n-i} V_0 \end{aligned}$$

Applying the binomial theorem gives: $= (u\mathsf{D} + (1 - u))^n \, V_0$

Taking the derivative

One advantage of this form is that now we can take the derivative:

$$Q'(u) = n(u\mathsf{D} + (1 - u))^{n-1} (\mathsf{D} - 1) V_0$$

What's $(\mathsf{D} - 1) V_0$?

Plugging in and expanding:

$$Q'(u) = n \sum_{i=0}^{n-1} \binom{n-1}{i} u^i (1-u)^{n-1-i} \mathsf{D}_i (V_0 - V_1)$$

This gives us a general expression for the derivative $Q'(u)$.

Specializing to n = 3

What's the derivative $Q'(u)$ for a cubic Bezier curve?

Note that:

- When $u = 0$: $Q'(u) = 3(V_1 - V_0)$
- When $u = 1$: $Q'(u) = 3(V_3 - V_2)$

Geometric interpretation:

So for $C1$ continuity, we need to set:

$$3(V_3 - V_2) = 3(W_1 - W_0)$$

Taking the second derivative

Taking the derivative once again yields:

$$Q''(u) = n\,(n - 1)\,(u\mathsf{D} + (1 - u))^{n-2}\,(\mathsf{D} - 1)^2\,V_0$$

What does $(\mathsf{D} - 1)^2$ do?

Second-order continuity

So the conditions for second-order continuity are:

$$(V_3 - V_2) = (W_1 - W_0)$$

$$(V_3 - V_2) - (V_2 - V_1) = (W_2 - W_1) - (W_1 - W_0)$$

Putting these together gives:

Geometric interpretation

C^3 continuity

<u>Summary of continuity conditions</u>

- C^0 straightforward, but generally not enough
- C^3 is too constrained (with cubics)

Creating continuous splines

We'll look at three ways to specify splines with C^1 and C^2 continuity:

1. C^2 interpolating splines

2. B-splines

3. Catmull-Rom splines

C^2 Interpolating splines

The control points specified by the user, called "joints," are <u>interpolated</u> by the spline.

For each of *x* and *y*, we needed to specify ______ conditions for each cubic Bezier segment.

So if there are m segments, we'll need ______ constraints.

Q: How many of these constraints are determined by each joint?

In-depth analysis, cont.

At each <u>interior</u> joint *j*, we have:

1. Last curve ends at *j*
2. Next curve begins at *j*
3. Tangents of two curves at *j* are equal
4. Curvature of two curves at *j* are equal

The *m* segments give:

- ______ interior joints
- ______ conditions

The 2 end joints give 2 further contraints:

1. First curve begins at first joint
2. Last curve ends at last joint

Gives _______ constraints altogether.

End conditions

The analysis shows that specifying $m + 1$ joints for m segments leaves 2 extra degrees of freedom.

These 2 extra constraints can be specified in a variety of ways:

- An interactive system
  - Constraints specified as _________
- "Natural" cubic splines
  - Second derivatives at endpoints defined to be 0
- Maximal continuity
  - Require C^3 continuity between first and last pairs of curves

C^2 Interpolating splines

Problem: Describe an interactive system for specifiying C2 interpolating splines.

Solution:

1. Let user specify first four Bezier control points.
2. This constrains next _____ control points -- draw these in.
3. User then picks _____ more
4. Repeat steps 2-3.

Global vs. local control

These C^2 interpolating splines yield only "global control" -- moving any one joint (or control point) changes the entire curve!

Global control is problematic:

- Makes splines difficult to design
- Makes incremental display inefficient

There's a fix, but nothing comes for free. Two choices:

- B-splines
    - Keep C^2 continuity
    - Give up interpolation
- Catmull-Rom splines
    - Keep interpolation
    - Give up C^2 continuity -- provides C^1 only

B-splines

Previous construction (C^2 interpolating splines):

- Choose joints, constrained by the "A-frames."

New construction (B-splines):

- Choose points on A-frames
- Let these determine the rest of Bezier control points and joints

The B-splines I'll describe are known more precisely as "uniform B-splines."

B-spline construction

The points specified by the user in this construction are called "de Boor points."

B-spline properties

Here are some properties of B-splines:

- <u>C^2 continuity</u>
- <u>Approximating</u>
  - Does not interpolate deBoor points
- <u>Locality</u>
  - Each segment determined by 4 deBoor points
  - Each deBoor point determines 4 segments
- <u>Convex hull</u>
  - Curve lies inside convex hull of deBoor points

Algebraic construction of B-splines

$$V_1 = \underline{\hspace{2cm}}\, B_1 + \underline{\hspace{2cm}}\, B_2$$

$$V_2 = \underline{\hspace{2cm}}\, B_1 + \underline{\hspace{2cm}}\, B_2$$

$$V_0 = \underline{\hspace{2cm}}\, [\underline{\hspace{2cm}}\, B_0 + \underline{\hspace{2cm}}\, B_1] + \underline{\hspace{2cm}}\, [\underline{\hspace{2cm}}\, B_1 + \underline{\hspace{2cm}}\, B_2]$$

$$= \underline{\hspace{2cm}}\, B_0 + \underline{\hspace{2cm}}\, B_1 + \underline{\hspace{2cm}}\, B_2$$

$$V_3 = \underline{\hspace{2cm}}\, B_1 + \underline{\hspace{2cm}}\, B_2 + \underline{\hspace{2cm}}\, B_3$$

Algebraic construction of B-splines, cont.

Once again, this construction can be expressed in terms of a matrix:

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix}$$

Drawing B-splines

Drawing B-splines is therefore quite simple:

> **procedure** *Draw-B-Spline* ($\{B_0, ..., B_n\}$):
>
>   **for** i = 0 to n - 3 do
>
>   Convert $B_i, ..., B_{i+3}$ into a Bezier control polygon $V_0, ..., V_3$
>
>   *Display* ($\{V_0, ..., V_3\}$)
>
>   **end for**
>
> **end procedure**

Multiple vertices

Q: What happens if you put more than one control point in the same place?

Some possibilities:

- Triple vertex
- Double vertex
- Collinear vertices

End conditions

You can also use multiple vertices at the endpoints:

- Double endpoint
  - Curve tangent to line between first distinct points
- Triple endpoint
  - Curve interpolates endpoint
  - Starts out with a line segment
- Phantom vertices
  - Gives interpolation without line segment at ends

Catmull-Rom splines

The Catmull-Rom splines

- Give up C^2 continuity
- Keep interpolation

For the derivation, let's go back to the interpolation algorithm. We had 4 conditions at each joint *j*:

1. Last curve ends at *j*
2. Next curve begins at *j*
3. Tangents of two curves at *j* are equal
4. Curvature of two curves at *j* are equal

If we ...

- Eliminate condition 4
- Make condition 3 depend only on local control points

... then we can have local control!

Derivation of Catmull-Rom splines

Idea: (Same as B-splines)

- Start with joints to interpolate
- Build a cubic Bezier curve between successive points

The endpoints of the cubic Bezier are obvious:

$$V_0 = B_1$$
$$V_3 = B_2$$

Q: What should we do for the other two points?

Derivation of Catmull-Rom, cont.

A: Catmull & Rom use *half the magnitude of the vector between adjacent control points*:

Many other choices work -- for example, using an arbitrary constant τ times this vector gives a "tension" control.

Matrix formulation

The Catmull-Rom splines also admit a matrix formulation:

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 0 & 6 & 0 & 0 \\ -1 & 6 & 1 & 0 \\ 0 & 1 & 6 & -1 \\ 0 & 0 & 6 & 0 \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix}$$

<u>Exercise:</u> Derive this matrix.

Properties

Here are some properties of Catmull-Rom splines:

- C^1 Continuity
- Interpolating
- Locality
- **No** convex hull property
    - (Proof left as an exercise.)