Motivation:
- Necessary to model organic/machined surfaces
- Needed to “blend” differently-oriented primitives
- Designers tire of linear (polyhedral) primitives
- Model plane and space curves first, then surfaces
- Applications beyond shape modeling:
  - Smooth variation of position, pose, camera parameters, etc.

Curved Primitives

Plane and space curves

Want a curve primitive that has:
- Explicit representation (for rendering)
- Controllable start/end points
- Controllable start/end derivatives

Try: implicit curves $Ax^2 + By^2 + ... F = 0$
- How to model “less” than the whole curve?
- How to fit to given position, derivative?

Try: functions $y = f(x)$
- How to model double-valued or vertical curves?
- How to rotate the curve?

Parametric Curves

Express (simple) lerp in (complicated) new way

$$Q(t) = \begin{pmatrix} Q_x(t) \\ Q_y(t) \\ Q_z(t) \end{pmatrix} = \begin{pmatrix} (P_0) & (P_1) \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} t \\ 1 \end{pmatrix}$$

What is the derivative $Q'(t)$ w.r.t. $t$?

$$Q'(t) = \frac{d}{dt} Q(t) = \begin{pmatrix} x'(t) \\ y'(t) \\ z'(t) \end{pmatrix} = \begin{pmatrix} (P_0) & (P_1) \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Note that $Q(t), Q'(t)$ can be written as:

$Q(t) = \text{Geometry} \cdot \text{Spline Basis} \cdot \text{Power Basis} \cdot T(t)$

$$Q(t) = \textbf{GBT}(t) \quad Q'(t) = \textbf{GBT}'(t)$$
Hermite Curves

Want to specify position, derivative at start, endpoint

What do we know?
Linear, quadratic aren’t sufficient. Why?

Given:
\[ Q(t) = GBT(t), \quad Q'(t) = GBT'(t); \]

Solve for B:
\[ GBT(0) = P_1 \]
\[ GBT(1) = P_4 \]
\[ GBT'(0) = R_1 \]
\[ GBT'(1) = R_4 \]

MIT 6.837 Computer Graphics
Tuesday, 19 November 2002 (L21)
Hermite Curves

\[ Q(t) = GBT = \begin{pmatrix} Q_1(t) \\ Q_2(t) \\ Q_3(t) \end{pmatrix}; \quad B = \begin{pmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix} \]

Write out \( Q(t), Q'(t) \):

\[ Q(t) = (2t^3 - 3t^2 + 1)P_1 + (2t^3 - 3t^2 + t)P_4 + (t^3 - 2t^2 + 1)R_1 + (t^3 - t^2)R_4 \]
\[ Q'(t) = (6t^2 - 6t + 0)P_1 + (6t^2 - 6t)P_4 + (3t^3 - 4t^2 + 1)R_1 + (3t^3 - 2t)R_4 \]

Check \( Q(0), Q(1), Q'(0), Q'(1) \)

Call polynomials in \( t \) the Hermite basis

Graph them with respect to \( t \):

Note how control point “weights” vary over interval

Bézier Curve

\[ Q(t) = (1 - t)^3P_1 + 3t(1-t)^2P_2 + 3t^2(1-t)P_3 + t^3P_4 \]
\[ Q'(t) = -3(1-t)^3P_1 + (9t^2 - 12t + 3)P_2 + (-9t^2 + 6t)P_3 + 3t^2P_4 \]

\[ Q(0) = P_1 \]
\[ Q(1) = P_4 \]
\[ Q'(0) = 3(P_2 - P_3) \]
\[ Q'(1) = 3(P_4 - P_3) \]

Properties:

- Endpoint interpolation
- Endpoint tangents

Bézier Curve Properties

Convex hull property:

Is the curve planar? Why or why not?
**Bézier Basis Functions**

Bézier “basis functions” weight control point contributions

\[ B_1(t) = (1 - t)^3; \quad B_2(t) = 3t(1 - t)^2; \quad B_3(t) = 3t^2(1 - t); \quad B_4(t) = t^3 \]

**Tangent Matching**

Can use control points to match tangents:

What are the design degrees of freedom?

**Variation Diminishing**

Curve crosses line (plane) at most the number of times control polygon crosses line (plane)

**Defined for any degree**

Use “Bernstein polynomials,” one per control point

\[ B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1 - t)^{n-i}, \quad 0 \leq i \leq n \]

Linear, Cubic, we’ve seen

What about quadratic?

**Affine Invariance**

Affine transformation, Bézier generation commute!

Note: does not commute with perspective!

For that, we must use *rational* curves
Stability of Evaluation

High degree power basis is numerically **unstable**
But… beautiful property of Bézier curves:
Evaluation via nested interpolation!

![Bézier Curve Diagram](image)

Works for any degree! (de Casteljau evaluation)
How long does it take?

Quality of Evaluation

Still, *discretization artifacts* can arise

![Discretization Artifacts](image)

How can this problem be avoided?

Subdivision:

Split curve segment $Q$ into two segments $L$, $R$

![Subdivision Diagram](image)

Why? How?

B-Splines (F-CDFHP §9.2.4)

(Sometimes) Undesirable properties of Hermite, Bézier
- Asymmetry; non-local effect of point manipulation

Using B-splines addresses both issues

Given four control points $P_1$, $P_4$

![B-Spline Diagram](image)

Define $Q(t)$ with different basis functions:

$$Q(t) = \frac{(1-t)^3}{6} P_{-3} + \frac{3t^3 - 6t^2 + 4}{6} P_{-2} + \frac{-3t^3 + 3t}{6} P_{-1} + \frac{t^3}{6} P$$
B-Splines

What are \( Q(0) \), \( Q(1) \)? Why would you want to do this?

\[
Q(t) = \frac{(1-t)^3}{6} P_{i-3} + \frac{3(1-t)^2 t}{6} P_{i-2} + \frac{(1-t) t^2}{6} P_{i-1} + \frac{t^3}{6} P_i,
\]

(in figure, \( i = 4 \))

B-Splines

What do we gain? (Contrast to Bézier)

- Symmetry: every point plays the same role!
- Smoothness: curve is \( C^2 \) everywhere
- Local control: Every point has limited effect on curve

What do we lose?

B-Splines

Single segment unremarkable. But \emph{join} segments:

Now notation makes sense:

Each interval \( 0 \leq t \leq 1 \) indexed by \( i \)

Spline Bases

Recall compact spline representation

\( Q(t) = \text{Geometry} \ G \cdot \text{Spline Basis} \ B \cdot \text{Power Basis} \ T(t) \)

But \( Q(t) \) expressible as Hermite, Bézier, ...

\( Q(t) = G_H \cdot B_H \cdot T(t) = G_B \cdot B_B \cdot T(t) \)

What is the relationship between \( G_H \) and \( G_B \)?

We can convert among several representations!
Basis Unification

Spline bases:

\[ B_{\text{Hermite}} = \begin{pmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix} \]

\[ B_{\text{Bezier}} = \begin{pmatrix} 1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \]

\[ B_{\text{B-Spline}} = \begin{pmatrix} 1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \]

\[ Q(t) = GBT = \begin{pmatrix} Q_x(t) \\ Q_y(t) \\ Q_z(t) \end{pmatrix} \]

Parametric Surfaces

What do we “want” from a surface primitive?
- Interpolation of corner points
- Tangent control
- Local control

First attempt: Bilinear interpolation
(analogous to linear interpolation)
Bi-lerp a (typically non-planar) quadrilateral

Notation: \( L(P_1, P_2, \alpha) \equiv (1 - \alpha)P_1 + \alpha P_2 \)

\[ Q(s, t) = L(L(P_1, P_2, t), L(P_3, P_4, t), s) \]

Bilinear Interpolation

How does this surface primitive work?
- Interpolates endpoints
- Tangent control?

What is implicit degree of bilinear patch
(i.e., degree of equivalent polynomial in \( x, y, z \))? 
- Desire interpolation of control points
- (at least at corners)
- And independent control of tangents
- (again, at least at corners)
- By inspection, linear, bilinear don’t work
- Why don’t quadratic or biquadratic patches work?

Bicubic Bézier Patch

Define “Tensor-product” Bézier surface
Notation: \( CB(P_1, P_2, P_3, P_4, \alpha) \) is Bézier curve
with control points \( P_i \) evaluated at \( \alpha \)

Then define Bézier patch \( Q(s, t) \) as

\[ Q(s, t) = CB( \begin{pmatrix} CB(P_{10}, P_{11}, P_{12}, P_{13}, t), \\ CB(P_{20}, P_{21}, P_{22}, P_{23}, t), \\ CB(P_{30}, P_{31}, P_{32}, P_{33}, t), \\ CB(P_{40}, P_{41}, P_{42}, P_{43}, t), s \end{pmatrix} ) \]
Bézier Patch

Can rewrite Bézier patch equation as:

\[ Q(s, t) = \sum B_{ij}(s, t) P_{ij} \]

Look at function multiplying one control point:

Bézier Patch Properties

Corner interpolation

What is the normal to the patch at \((s, t)\)?

Hint: consider curves of constant \(s\) or \(t\)

What is the implicit degree of the patch?
(i.e., degree of equivalent polynomial in \(x, y, z\)?)

Bézier Basis Functions

What do we know about the basis functions \(B_{ij}\)?

What properties does this imply? (Locality?)

Problem with Splines

Spline evaluation seems to involve:

- Several additional degrees of freedom (parametrization)
- Significant computational machinery

Can we generate smooth curve/surface directly from control net?
Chaikin’s Algorithm

Generates piecewise-linear approximation of smooth curve
Idea: “clip” vertices of control polygon
Introduce new vertices at $\frac{1}{4}, \frac{3}{4}$ points along each edge

Chaikin’s Algorithm

Should we subdivide all parts of control net to the same depth?
Can we easily generalize this to 2D control nets / 3D polyhedra?

Doo-Sabin Algorithm

Idea: introduce a new vertex for each face
At the midpoint of old vertex, face centroid

Doo-Sabin Example

Then connect new vertices in fixed topology
Catmull-Clark Subdivision

Similar idea, but different weighting factors
New vertices from face centroids \( \frac{1}{4} \), edge midpoints \( \frac{1}{2} \), old vertex \( \frac{1}{4} \)

Many other subdivision schemes have been proposed

Summary

Curve/surface primitives have conflicting requirements
Designers want:
- Intuitive “handles” on position, derivative
- Multi-resolution editing capability
- Smooth (multiply differentiable) everywhere
- Easily determinable limit surface
Implementers want:
- Stable, efficient evaluation method
- Local control (for caching, re-evaluation)
Curve/surface generation methods still an active research area