CS-184: Computer Graphics

Lecture #12: Curves and Surfaces

Prof. James O’Brien
University of California, Berkeley
Today

- General curve and surface representations
- Splines and other polynomial bases
Geometry Representations

- Constructive Solid Geometry (CSG)
- Parametric
  - Polygons
  - Subdivision surfaces
- Implicit Surfaces
- Point-based Surface

- Not always clear distinctions
  - i.e. CSG done with implicits
Geometry Representations

Object made by CSG
Converted to polygons
Geometry Representations

Object made by CSG
Converted to polygons
Converted to implicit surface
Geometry Representations

CSG on implicit surfaces
Geometry Representations

Point-based surface descriptions

Ohtake, et al., SIGGRAPH 2003
Geometry Representations

Subdivision surface (different levels of refinement)

Images from Subdivision.org
Geometry Representations

- Various strengths and weaknesses
  - Ease of use for design
  - Ease/speed for rendering
  - Simplicity
  - Smoothness
  - Collision detection
  - Flexibility (in more than one sense)
  - Suitability for simulation
  - many others...
Parametric Representations

Curves: \( \mathbf{x} = \mathbf{x}(u) \quad \mathbf{x} \in \mathbb{R}^n \quad u \in \mathbb{R} \)

Surfaces: \( \mathbf{x} = \mathbf{x}(u, v) \quad \mathbf{x} \in \mathbb{R}^n \quad u, v \in \mathbb{R} \)
\( \mathbf{x} = \mathbf{x}(u) \quad u \in \mathbb{R}^2 \)

Volumes: \( \mathbf{x} = \mathbf{x}(u, v, w) \quad \mathbf{x} \in \mathbb{R}^n \quad u, v, w \in \mathbb{R} \)
\( \mathbf{x} = \mathbf{x}(u) \quad u \in \mathbb{R}^3 \)

and so on...

Note: a vector function is really \( n \) scalar functions
Parametric Rep. Non-unique

- Same curve/surface may have multiple formulae

\( \mathbf{x}(u) = [u, u] \)

\( \mathbf{x}(u) = [u^3, u^3] \)
Simple Differential Geometry

- **Tangent to curve**
  \[ t(u) = \left. \frac{\partial x}{\partial u} \right|_u \]

- **Tangents to surface**
  \[ t_u(u, v) = \left. \frac{\partial x}{\partial u} \right|_{u,v} \quad t_v(u, v) = \left. \frac{\partial x}{\partial v} \right|_{u,v} \]

- **Normal of surface**
  \[ \hat{n} = \frac{t_u \times t_v}{\|t_u \times t_v\|} \]

- Also: curvature, curve normals, curve bi-normal, others...
- **Degeneracies:** \( \partial x/\partial u = 0 \) or \( t_u \times t_v = 0 \)
Discretization

- Arbitrary curves have an uncountable number of parameters

i.e. specify function value at all points on real number line
Discretization

- Arbitrary curves have an uncountable number of parameters
- Pick complete set of basis functions
  - Polynomials, Fourier series, etc.
- Truncate set at some reasonable point

\[ x(u) = \sum_{i=0}^{\infty} c_i \phi_i(u) \]

- Function represented by the vector (list) of \( c_i \)
- The \( c_i \) may themselves be vectors

\[ \mathbf{x}(u) = \sum_{i=0}^{3} c_i \phi_i(u) \]
Polynomial Basis

• Power Basis

\[ x(u) = \sum_{i=0}^{d} c_i u^i \]

\[ x(u) = C \cdot P^d \]

\[ C = [c_0, c_1, c_2, \ldots, c_d] \]

\[ P^d = [1, u, u^2, \ldots, u^d] \]

The elements of \( P^d \) are linearly independant

i.e. no good approximation

\[ u^k \not\approx \sum_{i \neq k} c_i u^i \]

Skipping something would lead to bad results... odd stiffness
Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

For now, assume $u_0 = 0$ $u_1 = 1$
Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

\[ x(0) = c_0 = x_0 \]
\[ x(1) = \sum c_i = x_1 \]
\[ x'(0) = c_1 = x'_0 \]
\[ x'(1) = \sum i c_i = x'_1 \]
Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

\[
\begin{bmatrix}
  x_0 \\
  x_1 \\
  x'_0 \\
  x'_1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  1 & 1 & 1 & 1 \\
  0 & 1 & 0 & 0 \\
  0 & 1 & 2 & 3
\end{bmatrix} \cdot
\begin{bmatrix}
  c_0 \\
  c_1 \\
  c_2 \\
  c_3
\end{bmatrix}
\]

\[
p = B \cdot c
\]
Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

\[ c = \beta_H \cdot p \]

\[ \beta_n = B^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & 1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \]
Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

\[ c = \beta_H \cdot p \]

\[ x(u) = P^3 \cdot c = P^3 \beta_H p \]

\[
\begin{pmatrix}
1 + 0u - 3u^2 + 2u^3 \\
0 + 0u + 3u^2 - 2u^3 \\
0 + 1u - 2u^2 + 1u^3 \\
0 + 0u - 1u^2 + 1u^3
\end{pmatrix} p
\]

\[
\beta_H = B^{-1} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
-3 & 3 & -2 & 1 \\
2 & -2 & 1 & 1
\end{pmatrix}
\]
Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

\[ c = \beta_H \cdot p \]

\[ x(u) = \begin{bmatrix} 1 + 0u - 3u^2 + 2u^3 \\ 0 + 0u + 3u^2 - 2u^3 \\ 0 + 1u - 2u^2 + 1u^3 \\ 0 + 0u - 1u^2 + 1u^3 \end{bmatrix} p \]

Hermite basis functions

\[ x(u) = \sum_{i=0}^{3} p_i b_i(u) \]
Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

\[ x(u) = \sum_{i=0}^{3} p_i b_i(u) \]

Hermite basis functions
Hermite Basis

- Specify curve by
  - Endpoint values
  - Endpoint tangents (derivatives)
- Parameter interval is arbitrary (most times)
  - Don’t need to recompute basis functions
- These are cubic Hermite
  - Could do construction for any odd degree
  - \((d - 1)/2\) derivatives at end points
Cubic Bézier

- Similar to Hermite, but specify tangents indirectly

\[
\begin{align*}
x_0 &= p_0 \\
x_1 &= p_3 \\
x'_0 &= 3(p_1 - p_0) \\
x'_1 &= 3(p_3 - p_2)
\end{align*}
\]

Note: all the control points are points in space, no tangents.
Cubic Bézier

- Similar to Hermite, but specify tangents indirectly

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 2 & 3
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
c_3
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
-3 & 3 & 0 & 0 \\
0 & 0 & -3 & 3
\end{bmatrix}
\begin{bmatrix}
p_0 \\
p_1 \\
p_2 \\
p_3
\end{bmatrix}
\]

\[
x_0 = p_0 \\
x_1 = p_3 \\
x_0' = 3(p_1 - p_0) \\
x_1' = 3(p_3 - p_2)
\]
Cubic Bézier

- Plot of Bézier basis functions
Changing Bases

- **Power basis, Hermite, and Bézier all are still just cubic polynomials**
  - The three basis sets all span the same space
  - Like different axes in $\mathbb{R}^3 \times \mathbb{R}^4$

- **Changing basis**
  
  $c = \beta_Z p_Z$
  
  $c = \beta_H p_H$

  $p_Z = \beta_Z^{-1} \beta_H p_H$
Useful Properties of a Basis

- **Convex Hull**
  - All points on curve inside convex hull of control points
    - \[ \sum b_i(u) = 1 \quad b_i(u) \geq 0 \quad \forall u \in \Omega \]
  - Bézier basis has convex hull property
Useful Properties of a Basis

- Invariance under class of transforms
  - Transforming curve is same as transforming control points
    - \( x(u) = \sum_i p_i b_i(u) \iff T x(u) = \sum_i (T p_i) b_i(u) \)
  - Bézier basis invariant for affine transforms
  - Bézier basis NOT invariant for perspective transforms
    - NURBS are though...
Useful Properties of a Basis

- **Local support**
  - Changing one control point has limited impact on entire curve
- **Nice subdivision rules**
- **Orthogonality** \( \int_{\Omega} b_i(u)b_j(u)\, du = \delta_{ij} \)
- **Fast evaluation scheme**
- **Interpolation -vs- approximation**
DeCasteljau Evaluation

- A geometric evaluation scheme for Bézier
Adaptive Tessellation

- Midpoint test subdivision

- Possible problem
  - Simple solution if curve basis has *convex hull* property

If curve inside convex hull and the convex hull is nearly flat: curve is nearly flat and can be drawn as straight line

Better: draw convex hull

Works for Bézier because the ends are interpolated
Bézier Subdivision

- Form control polygon for half of curve by evaluating at $u=0.5$

Repeated subdivision makes smaller/flatter segments

Also works for surfaces...
Joining

\[ C^0 \iff b = b \]
\[ C^1 \iff b - a = c - b \]
\[ G^1 \iff \frac{b - a}{\|b - a\|} = \frac{c - b}{\|c - b\|} \]

If you change \( a, b, \) or \( c \) you must change the others.

But if you change \( a, b, \) or \( c \) you do not have to change beyond those three. *Local Support*
“Hump” Functions

- Constraints at joining can be built in to make new basis
Tensor-Product Surfaces

- Surface is a curve swept through space
- Replace control points of curve with other curves

\[ x(u, v) = \sum_i p_i b_i(u) \]
\[ \sum_i q_i(v) b_i(u) \]
\[ q_i(v) = \sum_j p_{ji} b_j(v) \]

\[ x(u, v) = \sum_{ij} p_{ij} b_i(u)b_j(v) \]
\[ b_{ij}(u, v) = b_i(u)b_j(v) \]

\[ x(u, v) = \sum_{ij} p_{ij} b_{ij}(u, v) \]
Hermite Surface Bases

Plus symmetries...
Hermite Surface Hump Functions

Plus symmetries...
Adaptive Tessellation

- Given surface patch
  - If close to flat: draw it
  - Else subdivide 4 ways
Adaptive Tessellation

- Avoid cracking

Passes flatness test

Fails flatness test
Adaptive Tessellation

- Avoid cracking

Cracks may be okay in some contexts...
Adaptive Tessellation

- Avoid cracking
Adaptive Tessellation

- Avoid cracking

Test interior and boundary of patch
Split boundary based on boundary test
Table of polygon patterns
May wish to avoid “slivers”