Teddy: A Sketching Interface for 3D Freeform Design
Takeo Igarashi
Hidehiyo Tanaka
Satoshi Matsumo

iWIREs
An Analyze-and-Edit Approach to Shape Manipulation

Ran Gal
Tel-Aviv University

Olga Sorkine
New York University

Niloy Mitra
Indian Institute of Technology

Daniel Cohen-Or
Tel-Aviv University

CS-184: Computer Graphics
Lecture 12: Curves and Surfaces
Maneesh Agrawala
University of California, Berkeley
Announcements

Assignment 5: due Fri Nov 5 by 11pm

Announcements

Assignment 3 graded
  * glookup -s as3, glookup -s as3ec

Announcements

Midterm graded
  * glookup -s midterm, glookup -s midtermec
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

Curve Drawing

Drawing Curves

\[ y = f(x) \]

Only one value of \( y \) for each value of \( x \)...
Drawing Curves

Parametric curves
- Both $x$ and $y$ are a function of some third parameter

\[
\begin{align*}
x &= f(u) \\
y &= f(u)
\end{align*}
\]

$u \in [u_0 \ldots u_1]$
Drawing Curves

Draw curves by drawing line segments

- Must take care in computing end points for lines
- How long should each line segment be?
- Variable spaced points

\[ x = f(u) \quad u \in [u_0 \ldots u_1] \]

Drawing Curves

Midpoint-test subdivision

\[ |f(u_{mid}) - l(0.5)| \]

Drawing Curves

Midpoint-test subdivision

\[ |f(u_{mid}) - l(0.5)| \]
Drawing Curves

Midpoint-test subdivision

- Not perfect
- We need more information for a guarantee...

Curve Representations
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:
\[ x = x(u) \quad x \in \mathbb{R}^n \quad u \in \mathbb{R} \]

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

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

and so on...

Note: a vector function is really \( n \) scalar functions

Parametric Rep. Non-unique

Same curve/surface may have multiple formulae

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

Tangent to curve
\[ t_u = \frac{\partial x}{\partial u} \]

Tangents to surface
\[ t_u(u, v) = \frac{\partial x}{\partial u}, t_v(u, v) = \frac{\partial x}{\partial v} \]

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:
\[ \frac{\partial x}{\partial u} = 0 \quad \text{or} \quad t_u \times t_v = 0 \]

Non Orthogonal Tangents

\[
\begin{bmatrix}
\cos(\theta) \\
\sin(\theta)
\end{bmatrix}
\begin{bmatrix}
\cos(\phi) \\
\sin(\phi)
\end{bmatrix}
\]

\[ \theta \in [0, 1] \quad \phi \in [-1, 1] \]

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

$$x(u) = \sum_{i=0}^{3} c_i \phi_i(u)$$

Polynomial Basis

Power Basis

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

$$x(u) = C \cdot \mathbf{P}^d$$

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

$$\mathbf{P}^d = [1, u, u^2, \ldots, u^d]$$

The elements of $\mathbf{P}^d$ are linearly independent

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

$u_0 = 0, \quad u_1 = 1 \quad x(u) = \sum_{i=0}^{d} c_i u^i$
Specifying a Curve

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

For now, assume

\[ u_0 = 0 \quad u_1 = 1 \quad x(u) = \sum_{i=0}^{d} c_i u^i \]

\[
\begin{align*}
x(0) &= c_0 = x_0 \\
x(1) &= c_1 = x_1 \\
x'(0) &= c_1 = x'_0 \\
x'(1) &= \sum c_i = x'_1
\end{align*}
\]

\[
\beta = B^{-1} \cdot p
\]

\[
p = B \cdot c
\]

\[
c = B^{-1} \cdot p
\]
Specifying a Curve

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

\[ \mathbf{c} = \beta_n \cdot \mathbf{p} \]

\[ x(u) = \mathcal{P}^3 \cdot \mathbf{c} = \mathcal{P}^3 \beta_n \mathbf{p} \]

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

Hermite basis functions

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

Cubic Bézier

Similar to Hermite, but specify tangents indirectly

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

Note: All 4 control points are points in space, no tangents
Edges of control polygon tangent to curve at endpts

Cubic Bézier

Similar to Hermite, but specify tangents indirectly

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 2 & 3
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
-3 & 3 & 0 & 0 \\
0 & 0 & -3 & 3
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_0' \\
x_1'
\end{bmatrix}
= \begin{bmatrix}
p_0 \\
p_3 \\
3(p_1 - p_0) \\
3(p_3 - p_2)
\end{bmatrix}
\]
Cubic Bézier

Bézier basis functions

\[ c = \beta_z p \]
\[ c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} p \]

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

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 \( \mathbb{R}^3 \) \( \mathbb{R}^4 \)

Changing basis

\[ c = \beta_z p \]
\[ c = \beta_H p_H \]

\[ p_z = \beta_z^{-1} \beta_H p_H \]

Useful Properties of Bézier Basis

Convex Hull

- All points on curve inside convex hull of control points
- Bézier basis has convex hull property

\[ \sum_i b_i(u) = 1 \quad b_i(u) \geq 0 \quad \forall u \in \Omega \]
Useful Properties of a Basis

Invariance under class of transforms

• Transforming curve is same as transforming control points
  • Bézier basis invariant for affine transforms

\[ \mathbf{x}(u) = \sum_i \mathbf{p}_i b_i(u) \iff T \mathbf{x}(u) = \sum_i (Tp_i)b_i(u) \]

• Bézier basis NOT invariant for perspective transforms
  • NURBS are though...

Why is such invariance useful?

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
DeCasteljau Evaluation
A geometric evaluation scheme for Bézier
DeCasteljau Evaluation
A geometric evaluation scheme for Bézier

Notice: Blue line always tangent to curve
Curve Drawing: Adaptive Subdiv.

Recall problem: Midpoint test subdivision
  - 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

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

Demo in Illustrator

Extension to Surfaces
Tensor-Product Surfaces

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

\[ x(u, v) = \sum_i p_i b_i(u) \]
\[ \sum_i q_i(v) b_i(u) \]

\[ q_i(v) = \sum_j p_{ij} b_j(v) \]

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

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

Domain [-1,1] for u and v

Plus symmetries...

Bézier Surface Patch

Bezier surface and 4 x 4 array of control points

Bezier Surfaces. Smooth Operators.

Bicubic Bézier Patch

Continuously Refined and Deformed Curve

More control points along Bézier curves; leading to a total of 16 control points for the cubic case.

Split?

0 0 0 output as is
0 0 1
0 1 0
1 0 0
0 1 1
1 1 0
1 0 1
1 1 1

Below is a sample of the text content extracted from the image:

**Hermite Surface Hump Functions**

Domain [-1,1] for u and v

Plus symmetries...

**Bézier Surface Patch**

Bezier surface and 4 x 4 array of control points

**Bezier Surfaces. Smooth Operators.**

Bicubic Bézier Patch

Continuously Refined and Deformed Curve

More control points along Bézier curves; leading to a total of 16 control points for the cubic case.

Split?

0 0 0 output as is
0 0 1
0 1 0
1 0 0
0 1 1
1 1 0
1 0 1
1 1 1

### Bézier Surface Patch

-Bezier surface and 4 x 4 array of control points

### Bezier Surfaces. Smooth Operators.

**Bicubic Bézier Patch**

Continuously Refined and Deformed Curve

More control points along Bézier curves; leading to a total of 16 control points for the cubic case.

Split?

0 0 0 output as is
0 0 1
0 1 0
1 0 0
0 1 1
1 1 0
1 0 1
1 1 1
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

Crack in the surface

Cracks may be okay in some contexts...
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"

Triangle Based Method (no cracks)
Adaptive Tessellation

Triangle Based Method (no cracks)

\[ \frac{(u_1 + u_2)}{2} \]

\[ B\left(\frac{(u_1 + u_2)}{2}\right) \]

\[ \frac{(x_1 + x_2)}{2} \]

\[ \frac{\|B\left(\frac{(u_1 + u_2)}{2}\right) - \frac{(x_1 + x_2)}{2}\|}{\tau} < ? \]
Adaptive Tessellation

Triangle Based Method (no cracks)

Center test tends to generate slivers. Often better to leave it out.

Adaptive Tessellation

Without center test  With center test

Second row shows typical error of swapping tests.

Yiding Jia, CS184 S08 -- I broke his code to make this example.
Adaptive Tessellation

Visible artifacts from cracks.

Apollo Ellis, CS 184 S08