Today

- General curve and surface representations
- Splines and other polynomial bases
<table>
<thead>
<tr>
<th>Geometry Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Constructive Solid Geometry (CSG)</td>
</tr>
<tr>
<td>• Parametric</td>
</tr>
<tr>
<td>• Polygons</td>
</tr>
<tr>
<td>• Subdivision surfaces</td>
</tr>
<tr>
<td>• Implicit Surfaces</td>
</tr>
<tr>
<td>• Point-based Surface</td>
</tr>
<tr>
<td>• Not always clear distinctions</td>
</tr>
<tr>
<td>• i.e. CSG done with implicits</td>
</tr>
</tbody>
</table>

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

Subdivision surface (different levels of refinement)

Images from Subdivision.org

Tuesday, March 5, 13
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
  \[ t_u(u, v) = \frac{\partial x}{\partial u} \quad t_v(u, v) = \frac{\partial x}{\partial v} \]
- 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: \( \frac{\partial x}{\partial u} = 0 \) or \( t_u \times t_v = 0 \)
**Tangent Space**

- The tangent space at a point on a surface is the vector space spanned by
  \[
  \frac{\partial \mathbf{x}(u)}{\partial u} \quad \frac{\partial \mathbf{x}(u)}{\partial v}
  \]

  - Definition assumes that these directional derivatives are linearly independent.
  - Tangent space of surface may exist even if the parameterization is bad.
  - For surface the space is a plane.
  - Generalized to higher dimension manifolds.

**Non Orthogonal Tangents**

\[
\begin{bmatrix}
\cos(2\pi \theta) \cos(\phi \pi / 2) \\
\sin(2\pi \theta) \cos(\phi \pi / 2) \\
\sin(\phi \pi / 2)
\end{bmatrix} + \begin{bmatrix}
\cos(2\pi \theta) \cos \left( \frac{\pi}{2} \left( 1 - |\phi| \right) \cos(6\pi \theta) \phi + \phi \right) \\
\cos \left( \frac{\pi}{2} \left( 1 - |\phi| \right) \cos(6\pi \theta) \phi + \phi \right) \sin(2\pi \theta) \\
\sin \left( \frac{\pi}{2} \left( 1 - |\phi| \right) \cos(6\pi \theta) \phi + \phi \right)
\end{bmatrix}
\]

\(\theta \in [0..1] \quad \phi \in [-1..1]\)
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

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

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

• 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

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

\[ x(u) = C \cdot \mathbf{p}^d \]

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

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

The elements of \( \mathbf{p}^d \) are linearly independent, i.e., no good approximation

\[ u^k \neq \sum_{i=k} c_i u^i \]

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 \( u_0 = 0 \), \( u_1 = 1 \)
Given desired values (constraints) how do we determine the coefficients for cubic power basis?

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

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

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

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

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

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

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

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

\[ c = \beta_n \cdot p \]

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

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

Hermite basis functions

Tuesday, March 5, 13
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.

Tuesday, March 5, 13
Cubic Bézier

- Similar to Hermite, but specify tangents indirectly

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 2 & 3
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
-3 & 3 & 0 & 0 \\
0 & 0 & -3 & 3
\end{bmatrix}
\begin{bmatrix}
p_0 \\
p_1 \\
p_2 \\
p_3
\end{bmatrix}
\]

\[c = \begin{bmatrix}
1 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 \\
3 & -6 & 3 & 0 \\
-1 & 3 & -3 & 1
\end{bmatrix}
\begin{bmatrix}
p_0 \\
p_1 \\
p_2 \\
p_3
\end{bmatrix}
= \beta_z p
\]

\[x(u) = \begin{bmatrix}
1 - 3u + 3u^2 - u^3 \\
0 + 3u - 6u^2 + 3u^3 \\
0 + 0u + 3u^2 - 3u^3 \\
0 + 0u + 0u^2 + 1u^3
\end{bmatrix}
\begin{bmatrix}
p_0 \\
p_1 \\
p_2 \\
p_3
\end{bmatrix}
\]

Bézier basis functions

\[c = \beta_z p \quad \Rightarrow \quad \begin{bmatrix}
1 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 \\
3 & -6 & 3 & 0 \\
-1 & 3 & -3 & 1
\end{bmatrix}
\begin{bmatrix}
p_0 \\
p_1 \\
p_2 \\
p_3
\end{bmatrix}
\]

\[x(u) = p^3 \cdot c\]
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
- Changing basis \( \mathbb{R}^3 \rightarrow \mathbb{R}^4 \)

\[
\begin{align*}
c &= \beta_Z p_Z \\
c &= \beta_H p_H \\
p_Z &= \beta_Z^{-1} \beta_H p_H
\end{align*}
\]

Useful Properties of a Basis

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

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

- **Invariance under class of transforms**
  - Transforming curve is same as transforming control points
  - Bézier basis invariant for affine transforms
  - Bézier basis NOT invariant for perspective transforms
    - NURBS are though...

\[
x(u) = \sum_i p_i b_i(u) \iff T x(u) = \sum_i (T p_i) b_i(u)
\]

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

- A geometric evaluation scheme for Bézier curves

<table>
<thead>
<tr>
<th>u = 0</th>
<th>u = 0.25</th>
<th>u = 0.5</th>
</tr>
</thead>
</table>

Notice tangent line

- Blue line is always tangent to the curve.

From Wikipedia

Tuesday, March 5, 13
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 a 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...

We’ll extend this idea later on...

Tuesday, March 5, 13
Joining

$c^0 \Rightarrow b = b$
$c^1 \Rightarrow b - a = c - b$
$g^1 \Rightarrow \frac{b - a}{||b - a||} = \frac{c - b}{||c - b||}$

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

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

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

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

Plus symmetries...

Hermite Surface Hump Functions

Plus symmetries...

Tuesday, March 5, 13
Bézier Surface Patch

Beziersurface and 4 x 4 array of control points

Adaptive Tessellation

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

Tuesday, March 5, 13
Adaptive Tessellation

- Avoid cracking

Passes flatness test  Fails flatness test

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”
Adaptive Tessellation

• Triangle Based Method (no cracks)

Tuesday, March 5, 13
Adaptive Tessellation

- Triangle Based Method (no cracks)

\[
\begin{align*}
\frac{(u_1 + u_2)}{2} & \leq (x_1 + x_2) / 2 \\
\left\| B\left(\frac{(u_1 + u_2)}{2}\right) - \frac{(x_1 + x_2)}{2}\right\| & < \tau
\end{align*}
\]
Adaptive Tessellation

- Triangle Based Method (no cracks)

<table>
<thead>
<tr>
<th>Without center test</th>
<th>With center test</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Teapot Image]</td>
<td>![Teapot Image]</td>
</tr>
</tbody>
</table>

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

Adaptive Tessellation

Yiding Jia, CS184 S08

Tuesday, March 5, 13
Adaptive Tessellation

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, CS184 S08
Beziers. Smooth Operators.

Bicubic Bezier Patch

Continuously Moved and Deformed Bezier Curve

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

Split?

<table>
<thead>
<tr>
<th>Split?</th>
<th>a0</th>
<th>a1</th>
<th>a2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>