Quaternions

• Quaternion representation both allow for interpolation between arbitrary orientations and for representation of a series of rotations

Rotating Vectors Using Quaternions

\[ v' = Rot(v) = q^{-1} \cdot v \cdot q \]

\[ Rot_q(Rot_p(v)) = q^{-1} \cdot (p^{-1} \cdot v \cdot p) \cdot q \]

\[ = ((pq)^{-1} \cdot v \cdot (pq)) \]

\[ = Rot_{pq}(v) \]

\[ Rot^{-1}(Rot(v)) = q \cdot (q^{-1} \cdot v \cdot q) \cdot q^{-1} = v \]

Interpolation of Rotations using Quaternion Representation

\[ \cos \theta = q_1 \cdot q_2 = s_1 \cdot s_2 + v_1 \cdot v_2 \]

\[ Sleep(q_1, q_2, \theta) = ((\sin((1-n) \cdot \theta))/\sin(\theta)) \cdot q_1 + (\sin(\theta))/\sin(\theta) \cdot q_2 \]

CENG 732
Computer Animation
Spring 2006-2007
Week 3
Interpolation

This week

• Recap on interpolation/approximation splines
  – Natural Cubic Splines
  – Hermite Interpolation
  – Catmull-Rom Splines
  – Bezier Curves

• Timing considerations
  – Curve reparameterization by arclength
  – Speed control

• Path following
The problem

• Imagine an animator wants an object to be at position (-5,0,0) at frame 22 and at position (5,0,0) at frame 67.
  – We want to generate the position values in between frames 22 and 67
  – How?

Interpolation Considerations

• Interpolation vs. Approximation
• Complexity (i.e. degree of the polynomial)
• Continuity
• Global vs. Local Control

Interpolation vs. Approximation

• Interpolated: curve passes through control points
• Approximated guided by control points but not necessarily passes through them.

Continuity

• Parametric equations:
  \[ x = x(u), \quad y = y(u), \quad z = z(u), \quad u_i \leq u \leq u_f \]
• Parametric continuity: Continuity properties of curve segments.
  – Zero order: Curves intersects at one end-point: \( C^0 \)
  – First order: \( C^0 \) and curves has same tangent at intersection: \( C^1 \)
  – Second order: \( C^0, C^1 \) and curves has same second order derivative: \( C^2 \)

Continuity

• Geometric continuity:
  Similar to parametric continuity but only the direction of derivatives are significant. For example derivative \((1,2)\) and \((3,6)\) are considered equal.
• \( G^0, G^1, G^2 \): zero order, first order, and second order geometric continuity.
Global vs. Local Control

Local Control

Global Control

Spline Equations

- Cubic curve equations:
  \[
  x(u) = a_2 u^3 + b_2 u^2 + c_2 u + d_2 \\
  y(u) = a_3 u^3 + b_3 u^2 + c_3 u + d_3 \\
  z(u) = a_4 u^3 + b_4 u^2 + c_4 u + d_4
  \]
  \[0 \leq u \leq 1\]
  \[
  x(u) = [u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix} = U \cdot C
  \]

- General form: \( x(u) = U \cdot M_s \cdot M_g \)
- \( M_s \): spline transformation (blending functions)
- \( M_g \): geometric constraints (control points)

Natural Cubic Splines

- Interpolation of \( n+1 \) control points. \( n \) curve segments. 4\( n \) coefficients to determine.
- Second order continuity. 4 equation for each of \( n-1 \) common points:
  \[
  x_i(t) = p_i, \quad x_{i+1}(t) = p_{i+1}, \quad x'_{i+1}(t) = x'_{i+1}(0), \quad x''_{i+1}(t) = x''_{i+1}(0)
  \]
  4\( n \) equations required, 4\( n \)-4 so far.
- Starting point condition, end point condition.
  \[
  x_i(0) = p_i, \quad x_{i+1}(0) = p_{i+1}
  \]
- Assume second derivative 0 at end-points or add phantom control points \( p_{i+1} \), \( p_{i+1} \).
  \[
  x''(0) = 0, \quad x''(1) = 0
  \]

Hermite Interpolation

- End point constraints for each segment is given as:
  \[
  P(0) = p_0, \quad P'(0) = p_{0+1}, \quad P'(1) = p_{1+1}, \quad P'(1) = p_{1+1}
  \]
- Control point positions and first derivatives are given as constraints for each end-point.

These polynomials are called Hermite blending functions, and tells us how to blend boundary conditions to generate the position of a point \( P(u) \) on the curve.
Hermite blending functions

- Segments are local. First order continuity
- Slopes at control points are required.
- Catmull-Rom splines approximate slopes from neighboring control points.

Catmull-Rom Splines

- The tangent at a point is computed as the one-half of the two neighboring points

\[ P'_i = \frac{1}{2} \cdot (P_{i+1} - P_{i-1}) \]

\[ P_{i+1} - P_{i-1} \]

\[ P'_i \]

Bézier Curves

- A Bézier curve approximates any number of control points for a curve section (degree of the Bézier curve depends on the number of control points and their relative positions)

\[ P(u) = \sum_{i=0}^{n} \text{BEZ}_{i,n}(u), \quad 0 \leq u \leq 1 \]

\[ \text{BEZ}_{i,n}(u) = \binom{n}{k} u^k (1-u)^{n-k}, \quad \binom{n}{k} = \frac{n!}{k!(n-k)!} \]

- The coordinates of the control points are blended using Bézier blending functions \( \text{BEZ}_{i,n}(u) \)
- Polynomial degree of a Bézier curve is one less than the number of control points.
  - 3 points: parabola
  - 4 points: cubic curve
  - 5 points: fourth order curve
Cubic Bézier Curves

- Most graphics packages provide Cubic Béziers.

\[
\begin{aligned}
\text{BEZ}_{0,3} &= (1-u)^3 \\
\text{BEZ}_{1,3} &= 3u(1-u)^2 \\
\text{BEZ}_{2,3} &= 3u^2(1-u) \\
\text{BEZ}_{3,3} &= u^3
\end{aligned}
\]

\[
P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M_{3x4} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}
\]

\[
M_{3x4} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}
\]

Properties of Bézier curves

- Passes through start and end points

\[
P(0) = p_0, \quad P(1) = p_3.
\]

- First derivatives at start and end are:

\[
\begin{aligned}
P'(0) &= -3p_0 + 3p_1 \\
P'(1) &= -3p_{n-1} + 3p_n
\end{aligned}
\]

- Lies in the convex hull

Joining Bézier curves

- Start and end points are same (C^0)
- Choose adjacent points to start and end in the same line (C^1)

\[
P_0 = p_0, \quad P_1 = p_1 + \frac{n}{n'}(p_2 - p_{n-1})
\]

- C^2 continuity is not generally used in cubic Bézier curves. Because the information of the current segment will fix the first three points of the next curve segment

Controlling the speed

- Assume when we increase \(u\) 1 unit, we move along the curve \(x\) units (arclength). When we increase \(u\) 2 units, do we move 2\(x\) units on the curve?

- NO. Because the position is non-linearly dependent on \(u\) in cubic splines.
- For example, if \(u\) is the time parameter, \(m\)
Example

• For example, if \( u \) is the time parameter, the following positions will be generated at unit time intervals for a cubic curve

\[ \begin{align*}
A & : \text{time } = 0 \\
B & : \text{time } = 10 \\
C & : \text{time } = 35 \\
D & : \text{time } = 60
\end{align*} \]

Solution

• Solution to obtain a constant speed
  – We need to reparameterize by the arclength

Time and Position

Computing the arclength

\( s = G(u) \)

• We need to find the length of the curve from its starting position for any given parametric value:
  
  \( s = G(u) \)

• If we can compute \( G^{-1} \), then we can find how much time it takes to move a certain distance.

• But in general, there is no analytic solution to the problems above, so numerical techniques are used.

Using a table to calculate \( s = G(u) \)

\[
<table>
<thead>
<tr>
<th>\text{Index}</th>
<th>\text{Parametric Entry}</th>
<th>\text{Arc Length (G)}</th>
<th>\text{Index}</th>
<th>\text{Parametric Entry}</th>
<th>\text{Arc Length (G)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
<td>0.000</td>
<td>11</td>
<td>0.55</td>
<td>0.000</td>
</tr>
<tr>
<td>1</td>
<td>0.05</td>
<td>0.040</td>
<td>12</td>
<td>0.60</td>
<td>0.020</td>
</tr>
<tr>
<td>2</td>
<td>0.10</td>
<td>0.150</td>
<td>13</td>
<td>0.65</td>
<td>0.032</td>
</tr>
<tr>
<td>3</td>
<td>0.15</td>
<td>0.230</td>
<td>14</td>
<td>0.70</td>
<td>0.044</td>
</tr>
<tr>
<td>4</td>
<td>0.20</td>
<td>0.320</td>
<td>15</td>
<td>0.75</td>
<td>0.059</td>
</tr>
<tr>
<td>5</td>
<td>0.25</td>
<td>0.400</td>
<td>16</td>
<td>0.80</td>
<td>0.072</td>
</tr>
<tr>
<td>6</td>
<td>0.30</td>
<td>0.500</td>
<td>17</td>
<td>0.85</td>
<td>0.094</td>
</tr>
<tr>
<td>7</td>
<td>0.35</td>
<td>0.600</td>
<td>18</td>
<td>0.90</td>
<td>0.099</td>
</tr>
<tr>
<td>8</td>
<td>0.40</td>
<td>0.700</td>
<td>19</td>
<td>0.95</td>
<td>0.098</td>
</tr>
<tr>
<td>9</td>
<td>0.45</td>
<td>0.800</td>
<td>20</td>
<td>1.00</td>
<td>1.000</td>
</tr>
</tbody>
</table>
Finding the index closest to a given $u$

$$i = \left( \text{int} \left( \frac{\text{given parametric value}}{\text{distance between entries}} + 0.5 \right) \right)$$

$$= \left( \text{int} \left( \frac{0.73}{0.05} + 0.5 \right) \right) = 15$$

An estimation for $s$ can be $T(15) = 0.959$. A better approach is to use linear interpolation between $T(14)$ and $T(15)$

Finding the index closest to a given $u$

$$i = \left( \text{int} \left( \frac{\text{given parametric value}}{\text{distance between entries}} \right) \right) = \left( \text{int} \left( \frac{0.73}{0.05} \right) \right) = 14$$

$$L = \text{ArcLength}[i] + \left( \frac{\text{Value}[i+1] - \text{Value}[i]}{\text{ArcLength}[i+1] - \text{ArcLength}[i]} \right) \cdot (\text{ArcLength}[i+1] - \text{ArcLength}[i])$$

$$= 0.944 + \frac{0.73 - 0.70}{0.75 - 0.70} \cdot (0.959 - 0.944)$$

$$= 0.953$$

Solving the other problems using the table

- Finding $u = G^{-1}(s)$
- Finding $u_2$ given $u_1$ and $s$

<table>
<thead>
<tr>
<th>Index</th>
<th>Parametric Entry</th>
<th>Arc Length (G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
<td>0.000</td>
</tr>
<tr>
<td>1</td>
<td>0.05</td>
<td>0.000</td>
</tr>
<tr>
<td>2</td>
<td>0.10</td>
<td>0.150</td>
</tr>
<tr>
<td>3</td>
<td>0.15</td>
<td>0.320</td>
</tr>
<tr>
<td>4</td>
<td>0.20</td>
<td>0.520</td>
</tr>
<tr>
<td>5</td>
<td>0.25</td>
<td>0.660</td>
</tr>
<tr>
<td>6</td>
<td>0.30</td>
<td>0.800</td>
</tr>
<tr>
<td>7</td>
<td>0.35</td>
<td>0.900</td>
</tr>
<tr>
<td>8</td>
<td>0.40</td>
<td>0.970</td>
</tr>
<tr>
<td>9</td>
<td>0.45</td>
<td>1.000</td>
</tr>
<tr>
<td>10</td>
<td>0.50</td>
<td>1.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Parametric Entry</th>
<th>Arc Length (G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0.55</td>
<td>0.900</td>
</tr>
<tr>
<td>12</td>
<td>0.60</td>
<td>0.920</td>
</tr>
<tr>
<td>13</td>
<td>0.65</td>
<td>0.932</td>
</tr>
<tr>
<td>14</td>
<td>0.70</td>
<td>0.944</td>
</tr>
<tr>
<td>15</td>
<td>0.75</td>
<td>0.959</td>
</tr>
<tr>
<td>16</td>
<td>0.80</td>
<td>0.972</td>
</tr>
<tr>
<td>17</td>
<td>0.85</td>
<td>0.984</td>
</tr>
<tr>
<td>18</td>
<td>0.90</td>
<td>0.994</td>
</tr>
<tr>
<td>19</td>
<td>0.95</td>
<td>0.998</td>
</tr>
<tr>
<td>20</td>
<td>1.00</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Adaptive subdivision

Length $a - b$: Control curve tolerance

Length $c - d$: Common curve tolerance

Speed Control

- Specifying the speed along the curve

$$d(t) = (2-4)t^3$$

Generating ease-in/ease-out by sine curves

Sine curve segment to use as ease-in/ease-out control

$$s(t) = \sin \left( t \cdot \pi - \frac{\pi}{2} \right) + 1$$

Sine curve segment mapped to useful values

$$i(t) = e_{min} \cdot f \left( t \cdot \pi - \frac{\pi}{2} \right)$$

$$i(t) = \frac{\sin \left( t \cdot \pi - \frac{\pi}{2} \right) + 1}{2}$$
Ease-in/Ease-out alternative way

- Use high-school physics of motion

Curve fitting to position-time pairs

- Using an interpolating piecewise spline determine the piecewise $P(u)$ equations between control points
- Determine the arc-length of the segments by sampling $u$
- Compute the average velocity of the object between intervals by arc-length/time
- Move at constant speeds (average velocity) between intervals.

Path following

- Apart from the position of the object, the orientation of the object also has to be considered.

Frenet Frame

- If an object is moving along a path, the orientation can be made directly dependent on the properties of the curve (i.e., tangent and curvature).

\[ w = P'(\phi) \]
\[ u = P'(\phi) \times P''(\phi) \]
\[ v = w \times u \]
Looking towards a Center of Interest

\[ w = COI - POS \]
\[ u = w \times y\text{-axis} \]
\[ v = u \times w \]

Key-Frame Systems

• Shape-interpolation

\[ P(w) \]
\[ Q(v) \]

Specification of point correspondences and interpolation constraints

Articulation Variables

• AKA avar, track, or channel
• Associating the value of a variable with a function (e.g., time)

Animation Languages

• Abilities:
  – I/O operations for graphical objects
  – Support hierarchical composition of objects
  – A time variable
  – Interpolation functions
  – Transformations
  – Rendering-parameters
  – Camera attributes
  – Producing, viewing, and storing of one or more frames of animation
• A program written in an animation language is referred to as a \textit{script}.

Animation Languages

• Example:
  – Alias/Wavefront’s MEL

```plaintext
# Global proc emitAway()
{
  emitter pos 0 0 0 -type direction -sp 0.3 -name emit -r 50 -spd 1
  particle -name spray;
  connectDynamic -em emit.spray
  connectAttr emit.tx emitShape.dx;
  connectAttr emit.ty emitShape.dy;
  connectAttr emit.tz emitShape.dz;
  rename emit "emitAway";
  rename spray "sprayAway";
}
```
Another example: Houdini