CS174 C - Computer Animation

Discussion 1
Topics

- Rotations
- Splines
- Motion Control
Rotations

General Form

\[ M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

- Rotation matrix
- Fixed angle
- Axis-angle
- Euler angle
- Quaternion
- Exponential Map
3 subsequent rotations

\[ R = R_z(\theta_3)R_y(\theta_2)R_x(\theta_1) \]

Issues:

- Gimbal Lock
- Weird paths

Why does Gimbal Lock even appear?
- we are trying to map \( R^3 \) to \( SO(3) \) which cannot happen without singularities

Weird Paths
- More: Euler Explained
Quaternions span the $S(3)$ space.

$$q = a + bi + cj + dk, \quad a, b, c, d \in \mathbb{R}$$

$$q = [s, x, y, z] = [s, \mathbf{v}]$$

$$[s_1, \mathbf{v}_1] + [s_1, \mathbf{v}_1] = [s_1 + s_2, \mathbf{v}_1 + \mathbf{v}_2]$$

$$[s_1, \mathbf{v}_1][s_2, \mathbf{v}_2] = [s_1s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2]$$

$$(q_1q_2)q_3 = q_1(q_2q_3)$$

$$q_1q_2 \neq q_2q_1$$

$$|q| = \sqrt{s^2 + x^2 + y^2 + z^2}$$

$$q = (\cos(\theta), \sin(\theta)\mathbf{v}), \quad \mathbf{v} \text{ in } \mathbb{R}^3, \quad |\mathbf{v}| = 1$$
Idea: try to create a parameterization based from $R^3$ to $SO(3)$ (impossible)

$\Rightarrow$ Matrix Exponential

Summing an infinite series of skew-symmetric matrices

Second Idea: try to create a parameterization based from $R^3$ to $S^3$ and then map to $SO(3)$

$$e^v = \sum_{n=0}^{\infty} \left( \frac{1}{2} \hat{v} \right)^m$$

$\|v\|$  angle

$\frac{v}{\|v\|}$  axis of rotation
All of them:

OpenGL

- Interface based on Euler angles
- Internal representation using quaternions
- Drawing using matrices
All of them:

OpenGL

- Interface based on Euler angles
- Internal representation using quaternions
- Drawing using matrices
Very important tool for graphics

http://www.youtube.com/watch?feature=player_embedded&v=MJxCmpOtZlg#t=5s

- Curves are aesthetically pleasing
- The behavior of many objects exhibits a “smooth” quality that can be characterized functionally as curves
- Splines are used to model curved entities **effectively**
Piecewise-polynomial function

- Split a the real line into intervals
- Over each interval, pick a different polynomial
- Enforce continuity constraints

\[ P(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} \]
Use cubic polynomials (Cubic Hermite splines)

Constraints are given by:
- Positions at end points
- Tangents at end points
Generic Cubic Polynomial in interval \([t_i, t_{i+1}]\)

\[
q_i(t) = a_i(t - t_i)^3 + b_i(t - t_i)^2 + c_i(t - t_i) + d_i
\]
A spline can be used to drive a parameter over time

- Fade out Effect

\[ \text{Alpha} = a(t) \]
Just specifying position will lead to inconsistent velocity

Solution
– Make the motion consistent first by making the velocity constant throughout
– Use an additional function to retime the velocity for the final animation

http://www.youtube.com/watch?v=Zh0PdCQiEjo
Enforcing constant velocity is tricky

- The object should traverse a fixed length on the curve for every time step
- We need to parameterize the curve in terms of arclength

Unfortunately:

- There is no closed form solution
- Use piecewise approximation
- Construct a table

http://www.youtube.com/watch?v=nsdBFHAMUSM

<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>0.080</td>
</tr>
<tr>
<td>2</td>
<td>0.10</td>
<td>0.125</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>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>