Animation Curves and Splines 1
Animation Homework

• Set up a simple avatar
  • it can be a cube/sphere (or square/circle if 2D)
• Specify a number of key frames, positions and orientations
• Associate a time with each key frame
• Create smooth animations for the avatar movement
  • does it pass through or just near the key frames and positions?
• Save the canned animations for later use
• Provide a couple of examples
  • A crawling bug, a bouncing ball, etc...
• Use unity...
Smooth Animation

• Suppose the user pushes down on a button, and holds it down, intending for their avatar to carry out some motion. E.g. move forward, backward, turn, etc.

• Should the avatar be allowed to turn on a dime? Should it be allowed to instantly accelerate to full speed? Should it be allowed to instantly go from full speed to full stop?

• Jerky motion can be rather annoying to the user, and even cause headaches

• Having the right physical feel to the motion helps both immersive-ness and believability (suspension of disbelief)

• E.g. it’s more realistic for a car in a racing game to take a wider turn if going too fast, or even spin-out if the user tries to take too tight of a turn at too high of a speed
Smooth Animation

• On the other hand, “hard core” users may prefer complete control over their avatars
  • smoothing of animations can be perceived as input delays

• Even if the user is given tight instantaneous control over their avatar, it is typically desirable to smooth out the camera motion in order to avoid jerkiness of the entire screen
  • especially if the camera is attached in some way to the player’s unrealistically animated avatar

• It is also desirable to smooth out the motion of all the non-player avatars the player interacts with so that they have a more realistic appearance

• One will also want smooth walk, run, and other motion cycles for the avatar as it carries out its actions
Keyframing

- Keyframing is a method of animating an object by defining starting and ending points of a smooth transition.
- These starting and ending points of transitions are key frames.
- A sequence of key frames defines what movement the viewer will see.
Path Interpolation

• We need enough frames between key frames to give the viewer the illusion of continuous movement.

• The frames between key frames are interpolated from key frames.

\[ p(t) = \sum W_i(t)p_i \]
Path Interpolation

- Interpolation is not foolproof. There may be overshoots, undershoots or other side effects. The animator must be careful choosing interpolation methods.
- The figure shows a reasonable way of using interpolation to obtain a smooth path connecting key frames.
- But this would be an unrealistic path for a bouncing ball.
Temporal Parameterization

- Changing the way of parameterizing the curve as a function of time changes the object’s motion between key frames.
- The figure below shows a plausible path for a bouncing ball, but does not shows a plausible motion.
Temporal Parameterization

• Changing the parameterization of the curve as a function of time makes the motion more realistic

• The ball accelerates as it falls
Both Space and Time...

- Spatial curves determine the path for the motion
- Temporal parameterization along that path determines how fast the object moves
Temporal Parameterization

- First, parameterize the 3D curved path with respect to arc length $s$ to obtain a function $p(s)$.
- Then create a graph relating arc length $s$ versus time $t$ to specify how fast the object moves along the curved path through space.
- Plug $s(t)$ into $p(s)$ to get $p(t)$.
- Adjusting the temporal curve $s(t)$ controls how fast the object moves along the path $p(s)$ without changing the path.
Animation Curves

• Artists use animation curves to parameterize spatial positions as the function of time

• For a 3D motion, each of x, y, z coordinates can have its own animation curve

• Artist workflow:
  • Manipulate key points (blue dots)
  • Set tangent directions and lengths (arrows) at key points
How artists use Animation Curves…
Splines and Curves
Goal: Interpolate Values
Nearest Neighbor Interpolation

Problem: values not continuous
Linear Interpolation

Problem: derivatives not continuous
Smooth Interpolation?
Polynomial Interpolation

# of constraints = 3
polynomial degree = 2
Polynomial Interpolation

# of constraints = 4
polynomial degree = 3
Polynomial Interpolation

# of constraints = 5
polynomial degree = 4
Higher Order Polynomials

# of constraints = 5
polynomial degree = 4

• Curve may oscillate unexpectedly
Overconstrained / Least-Squares

- Curve does not interpolate points
- Instead it approximates the points

# of constraints = 5
polynomial degree = 2
Multiple Lower Order Polynomials

- Two 4-point interpolations
- Two degree 3 polynomials

- Curves don’t agree in region of overlap
Piecewise Polynomial Interpolation

Different curves in each interval

Match values and slopes at endpoints
Cubic Hermite Interpolation

Given: values and derivatives at 2 points
- 4 constraints \( \rightarrow \) need 4 degrees of freedom
- use a degree 3 cubic polynomial
Cubic Hermite Interpolation

- **Cubic polynomial**
  \[ f(t) = at^3 + bt^2 + ct + d \]
  \[ f'(t) = 3at^2 + 2bt + c \]

- **Solve for coefficients:**
  \[ f(0) = h_0 = d \]
  \[ f(1) = h_1 = a + b + c + d \]
  \[ f'(0) = h_2 = c \]
  \[ f'(1) = h_3 = 3a + 2b + c \]
Matrix Representation

\[ h_0 = d \]
\[ h_1 = a + b + c + d \]
\[ h_2 = c \]
\[ h_3 = 3a + 2b + c \]

\[
\begin{bmatrix}
  h_0 \\
  h_1 \\
  h_2 \\
  h_3
\end{bmatrix} =
\begin{bmatrix}
  0 & 0 & 0 & 1 \\
  1 & 1 & 1 & 1 \\
  0 & 0 & 1 & 0 \\
  3 & 2 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c \\
  d
\end{bmatrix}
\]
Solve for $a, b, c, d$

$$
\begin{bmatrix}
  a \\
  b \\
  c \\
  d \\
\end{bmatrix}
= 
\begin{bmatrix}
  2 & -2 & 1 & 1 \\
  -3 & 3 & -2 & -1 \\
  0 & 0 & 1 & 0 \\
  1 & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
  h_0 \\
  h_1 \\
  h_2 \\
  h_3 \\
\end{bmatrix}
$$

Inverse Matrix
Hermite Basis Functions?

\[ f(t) = \sum_{i=0}^{3} h_i H_i(t) \]

\[
\begin{bmatrix}
a & b & c & d \\
t^3 & t^2 & t & 1 \\
\end{bmatrix}
\begin{bmatrix}
h_0 \\
h_1 \\
h_2 \\
h_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
H_0(t) \\
H_1(t) \\
H_2(t) \\
H_3(t) \\
\end{bmatrix}
\]

monomial basis  Hermite basis
Insert Identity Matrix

\[
\begin{bmatrix}
a & b & c & d \\
0 & 1 & 0 & 3 \\
0 & 1 & 0 & 2 \\
0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 \\
\end{bmatrix}
\]

Identity

\[
\begin{bmatrix}
2 & -3 & 0 & 1 \\
-2 & 3 & 0 & 0 \\
1 & -2 & 1 & 0 \\
1 & -1 & 0 & 0 \\
\end{bmatrix}
\]

Hermite Basis

\[
\begin{bmatrix}
h_0(t) \\
h_1(t) \\
h_2(t) \\
h_3(t) \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
t^3 \\
t^2 \\
t \\
1 \\
\end{bmatrix}
\]
Hermite Basis Functions

\[
\begin{bmatrix}
H_0(t) \\
H_1(t) \\
H_2(t) \\
H_3(t)
\end{bmatrix} =
\begin{bmatrix}
2 & -3 & 0 & 1 \\
-2 & 3 & 0 & 0 \\
1 & -2 & 1 & 0 \\
1 & -1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
t^3 \\
t^2 \\
t \\
1
\end{bmatrix}
\]

\[
H_0(t) = 2t^3 - 3t^2 + 1
\]

\[
H_1(t) = -2t^3 + 3t^2
\]

\[
H_2(t) = t^3 - 2t^2 + t
\]

\[
H_3(t) = t^3 - t^2
\]

No need for inverse/solve, just add together splines:

\[
\sum_{i=0}^{3} h_i H_i(t)
\]
C^0 Continuous Hermite Splines

- Make the values equal at the shared position of two adjacent Hermite splines
C¹ Continuous Hermite Splines

- Also make the derivatives equal at the shared position of two adjacent Hermite splines
2D/3D Spatial Interpolation

- We can interpolate 2D/3D points as easily as scalars by independently interpolating each component of a vector.
- E.g., consider linear interpolation:
  - **Scalar:** \( f(t) = (1 - t)f(t_k) + tf(t_{k+1}) \)
  - **Point:** \( p(t) = (1 - t)p_k + tp_{k+1} \)
Piecewise Linear Spatial Interpolation

• Draw a line between each pair of points
• Each interval is a linear function (linear interpolation) constrained by its endpoints
• Vector formulation: Consider linear interpolation between point $p_0 = (x_0, y_0)$ and $p_1 = (x_1, y_1)$

$$
x(t) = (1 - t)x_0 + tx_1 \\
y(t) = (1 - t)y_0 + ty_1 \\
p(t) = (1 - t)p_0 + tp_1$$
Basis Functions to Define Curves

- Consider general spatial interpolation among spatial points $p_i, \ i = 0, 1, ..., n$, where $p_i = (x_i, y_i)$. We interpolate each component of $p(t)$:

  $$x(t) = \sum_{i=0}^{n} x_i W_i(t) \quad y(t) = \sum_{i=0}^{n} y_i W_i(t)$$

- Since the basis functions are the same for every component, we can write:

  $$p(t) = \sum_{i=0}^{n} p_i W_i(t)$$

- Coefficients of the basis functions can be points/vectors, not just scalars
Spatial Cubic Hermite Curves

- Given two points \( p_0 = (x(0), y(0)) \) and \( p_1 = (x(1), y(1)) \)
- To apply cubic Hermite interpolation to each component, we also need derivative constraints: \( x'(0), y'(0), x'(1), y'(1) \)
- \( (x'(0), y'(0)) \) is the (un-normalized) direction of the tangent at \( p_0 \)
- **Different magnitudes of \( (x', y') \) at end points lead to different Hermite splines!**

\[
\begin{align*}
v_0 &= (x'(0), y'(0)) \\
\end{align*}
\]
Spatial Cubic Hermite Curves

• Cubic Hermite interpolation for each component

\[ x(t) = H_0(t)x(0) + H_1(t)x(1) + H_2(t)x'(0) + H_3(t)x'(1) \]
\[ y(t) = H_0(t)y(0) + H_1(t)y(1) + H_2(t)y'(0) + H_3(t)y'(1) \]

• Assemble the equations of each component to get the spatial interpolation equation

\[ \mathbf{p}(t) = H_0(t)\mathbf{p}_0 + H_1(t)\mathbf{p}_1 + H_2(t)\mathbf{v}_0 + H_3(t)\mathbf{v}_1 \]
\( C^0 \) Continuity between 2 Hermite Curves

- 2\textsuperscript{nd} point of the 1\textsuperscript{st} curve is the same as the 1\textsuperscript{st} point of the 2\textsuperscript{nd} curve
C¹ Continuity between 2 Hermite Curves

- 2nd point of the 1st curve is the same as the 1st point of the 2nd curve
- Tangent of the 1st curve is equal to the tangent of the 2nd curve at the shared point
Catmull–Rom Interpolation

\[
\frac{f(t_3) - f(t_1)}{t_3 - t_1}
\]

\[
\frac{f(t_2) - f(t_0)}{t_2 - t_0}
\]

Automatically define derivatives as central difference
Catmull–Rom Interpolation

Then use Hermite Interpolation

\[ f(t_2) - f(t_0) \]
\[ \frac{t_2 - t_0}{t_2 - t_0} \]

\[ f(t_3) - f(t_1) \]
\[ \frac{t_3 - t_1}{t_3 - t_1} \]
Catmull–Rom Spatial Interpolation

- We can define the derivatives with respect to a parameter $t$ for each component using central difference.
- But how do we choose $t$?

\[ \mathbf{v}_0 = \frac{\mathbf{p}_2 - \mathbf{p}_0}{t_2 - t_0} \]
\[ \mathbf{v}_1 = \frac{\mathbf{p}_3 - \mathbf{p}_1}{t_3 - t_1} \]
Catmull–Rom Spatial Interpolation

\[ \mathbf{v}_0 = \frac{1}{2} (\mathbf{p}_2 - \mathbf{p}_0) \]

\[ \mathbf{v}_1 = \frac{1}{2} (\mathbf{p}_3 - \mathbf{p}_1) \]

- Common to just set the spacing between points to be 1, then \( t_{k+1} - t_{k-1} = 2 \)