CS 174C Week 4

Aidan Jan

January 31, 2024

Motion Curves

Examples:

- The position of an object: $x(t), y(t), z(t)$
  - 3 separate splines
- The angle of a simple joint (e.g., elbow)
  - 1 spline
- The angles of a complex joint (e.g., hip)
  - 2 or more splines
- The size of an object
  - 1 spline, or maybe a spline for each of 3 different axes
- The color of an object
  - 3 splines for R, G, B
- Etc.

Using Motion Curves

- Simplest usage:
  - Look at every parameter that changes during the animation
  - Use Hermite interpolation (initialized as a Catmull-Rom spline) in time
  - Allow animator to adjust values, adjust slopes, break continuity, add knots, move knots. ...

Problem

- Re-timing animations is not so simple
  - If you adjust a knot position, it changes the shape of the spline, not just the speed
  - Particularly for Hermite splines, slopes will be off
- Partial solution: Separate the shape of the motion curve fro the timing of the motion along the curve

Time as a Motion Curve

- Rename parameter of motion curves to "u"
- Then make a motion curve for time: $u(t)$
- Could have one global timing curve $u(t)$ or separately adjust timing for each variable, or group of variables.

Parameterization

- Unsatisfactory still: u doesn't really have a good meaning
- For example, to make an object move with constant speed along an arc, $u(t)$ may need to be complicated
- Solution: We will introduce a new map based on arc length s
  - Can easily control the speed of an object
  - Timing curve will now be $s(t)$, where s means distance traveled along the curve

Arc Length of a Curve

- Arc length is just the length of a curve
  - Think of laying a tape measure along the curve $P(u)$
  - Where $P(u)$ is the 3D position of the curve at parameter value u
    - Really three curves: $X(u), Y(u), Z(u)$
- Parametric Definition:

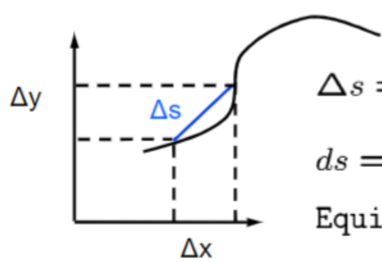
$$s(u) = \int_{u_1}^{u_2} \left| \frac{\mathrm{d}P}{\mathrm{d}u} \right| \mathrm{d}u$$

  - Recall how to measure vector norm:

$$\left| \frac{\mathrm{d}P}{\mathrm{d}u} \right| = \sqrt{\left(\frac{\mathrm{d}X}{\mathrm{d}u}\right)^2 + \left(\frac{\mathrm{d}Y}{\mathrm{d}u}\right)^2 + \left(\frac{\mathrm{d}Z}{\mathrm{d}u}\right)^2}$$

2D Example

- Compute a differential length



$$\Delta s = \sqrt{(\Delta x)^2 + (\Delta y)^2}$$

$$ds = \sqrt{dx^2 + dy^2} \rightarrow ds = \sqrt{(\frac{dx}{dt})^2 + (\frac{dy}{dt})^2} dt$$

Equivalently

$$ds = \sqrt{dx^2 + dy^2} \rightarrow ds = \sqrt{1 + (\frac{dy}{dx})^2} dx$$

- Integrate to compute the total arc length

Parametric Circle

- Parametric equation of a circle:

$$[x(t), y(t)] = [R\cos(t), R\sin(t)], \qquad t \in [0, 2\pi]$$

- Differential arc length:

$$\mathrm{d}s = \sqrt{\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^2}\mathrm{d}t$$

- Substituting and differentiating:

$$\mathrm{d}s = \sqrt{R^2\sin^2(t) + R^2\cos^2(t)}\mathrm{d}t = R\mathrm{d}t$$

- Integrating:

$$S(t) = \int_0^t R\mathrm{d}t = tR|_0^{2\pi}$$

Cubic Polynomial Splines

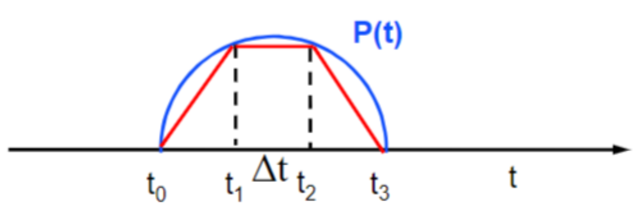
$$P(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \begin{bmatrix} a_x u^3 + b_x u^2 + c_x u + d_x \\ a_y u^3 + b_y u^2 + c_y u + d_y \\ a_z u^3 + b_z u^2 + c_z u + d_z \end{bmatrix} S(u) \qquad = \int_{u_1}^{u_2} \sqrt{\left(\frac{x(u)}{\mathrm{d}u}\right)^2 + \left(\frac{y(u)}{\mathrm{d}u}\right)^2 + \left(\frac{z(u)}{\mathrm{d}u}\right)^2}\mathrm{d}u$$

$$= \int_{u_1}^{u_2} \sqrt{Au^4 + Bu^3 + Cu^2 + Du + E}\mathrm{d}u$$

Numerical Computation

- Analytic approach is hopeless
  - Even analytically solving the integral $s(u)$ is hard, solving for u in terms of s is worse.
- Numerical approach
- Use approximate evaluation of $s(u)$ to get a table of values
  - Cut up curve into small line segments and add up their lengths
- Then interpolate a smooth curve through the values (Catmull-Rom)
  - Use table of s values as knots and associated u values as control points

Forward Differencing



Typically $t_0 = 0$, $t_{i+1} - t_i = \Delta t$

- Red line is a linear approximation of the blue curve.

- To do this, you would find the position at for example, 30 points, and linearly interpolate them. The connected line segments approximate the curve.
  - Once this is done, create a table that maps the parametric entry u to the arc length on the line, s. Then, when animating, interpolate between the arc lengths.
  - Example: [INSERT TABLE]

Calculate s With Rounding

- Given parametric entry find arc length

$$i = (int)\left(\frac{parametric\ entry\ (u)}{distance\ between\ entries} + 0.5\right)$$

- For $u = 0.73$

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

- So, arc length = s[i] = s[15] = 0.959
- Instead of rounding, interpolate:

$$i = (int)\left(\frac{parametric\ entry\ (u)}{distance\ between\ entries}\right) = 14$$

$$i + 1 = 15$$

$$\text{That is } u \in [u[i], u[i+1]]$$

$$\begin{aligned} S(u) &= S[i] + \frac{u - u[i]}{u[i+1] - u[i]}(S[i+1] - S[i]) \\ &= 0.944 + \frac{0.73 - 0.70}{0.75 - 0.70}(0.959 - 0.944) \end{aligned}$$

How Can We Control the Sampling Error?

- Adaptively subdivide
- First compare:
  - length(start, middle) + length(middle, end) to length (start, end)
  - Essentially, sample the points more finely when the line has higher curvature.
- If the difference is too large, subdivide
- Use linked list to store data
- Finally, copy to a fixed-size table

Numerical Computation

- Numerical methods exist to approximate the integral of a curve given sample points and derivatives
- Instead of using the sum of linear segments, use a numerical method to compute the sum of curved segments

Gaussian Quadrature

- Commonly used to integrate a function between -1 and +1
  - Function to be integrated is evaluated at specific points within [-1, +1] and is multiplied by pre-calculated weights
$$\int_{-1}^{1} f(u) = \sum_i w_i f(u_i)$$
  - More sample points lead to better accuracy
- Compute arc length of cubic curve
  - Adaptive Gaussian integration monitors errors to subsample
  - Build a table that maps parameter values to arc length values
  - To solve arc length at u, find u_i, and u_{i+1} that bound u
  - Use Gaussian quadrature to compute intermediate arc length between those at u_i and u_{i+1}

Inverse Map

- However, the question we really want to answer is what value of u gives us a specific arc length s along the curve $P(u)$?
  - i.e., we want to invert the arc length function $s(u)$
  - Let's call this $u(s)$
- Then the timing curve is $s(t)$, which feeds into $u(s)$, which feeds into motion curve $P(u)$:
  - Position at time t is $P(u(s(t)))$
- Question remains: How to calculate $u(s)$?

First Approach

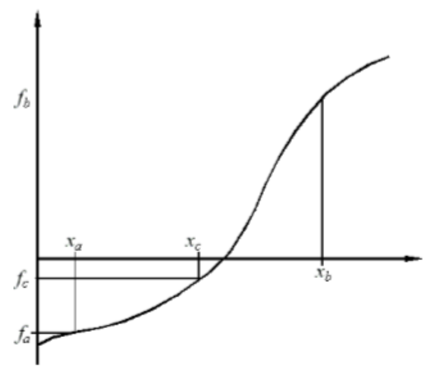
- Use the previous table of samples
- Problem?
- Arc Length s is not a linear function of u
  - Given s, we must search to find the value of t (index) in the table

Computs u Given s

- Compute u such that $f(u) = s - \text{length}(u_0, u) = 0$.
- This is a classic problem of root finding
- Methods?
  - Bisection
  - Newton-Raphson
  - Secant

Bisection Method

- Need two initial guesses that bracket the root
  - let $x_c = \frac{1}{2}(x_a + x_b)$
  - if $f_c = f(x_c) = 0$ then $x = x_c$ is an exact solution: return x_c
  - else, if $f_a \cdot f_c < 0$ then the root lies in the interval (x_a, x_c),
  - else the root lies in the interval (x_c, x_b).
  - replace the interval (x_a, x_b) with either (x_a, x_c) or (x_c, x_b), whichever brackets the root
  - repeat until $f(x_c) <$ threshold
  - return x_c



Newton-Raphson Method

- Based on Taylor expansion:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2} f''(x_0) + \cdots + \frac{(x - x_0)^n}{n!} f^{(n)}(x_0)$$

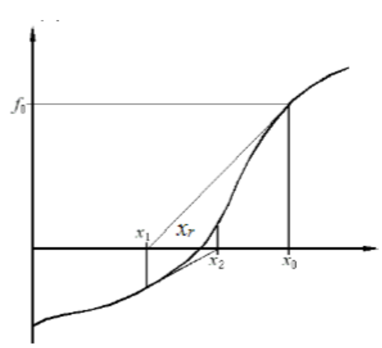
$f(x) \approx f(x_0) + (x - x_0)f'(x_0) + O(|x - x_0|^3)$

- Set $f(x) = 0$ and solve for $x = x_r$

$$\begin{aligned} 0 = f(x_r) &\approx f(x_0) + (x_r - x_0)f'(x_0) \\ \rightarrow 0 &\approx f(x_0) + (x_r - x_0)f'(x_0) \\ \rightarrow x_r &\approx x_0 - f(x_0)/f'(x_0) \end{aligned}$$

- Iteratively converge to the root

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$



Secant (Chord)

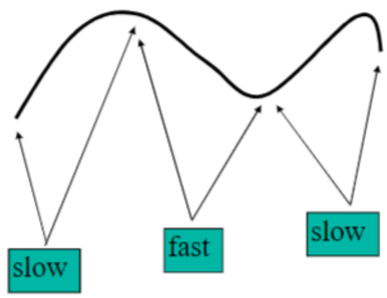
- Same as Newton-Raphson except that the derivative is approximated assets

$$f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

  - Substituted in Newton-Raphson, this gives

$$x_{n+1} = x_n - \frac{(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} f(x_n)$$

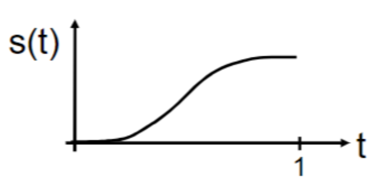
Controlling the Speed Along a Curve



- **Arc length is not proportional to time.**
- Equal Arc lengths
  - Constant speed
- Smooth motion
  - Easy in - Easy out
  - Acceleration - Deceleration
- Relate time to arc length: easy(t)

Speed Control

- Given arc-length parameterized curve
- Let $s(t)$ be the distance along the curve at time t
  - Normalize total distance to 1.0
  - Monotonically increasing and continuous
- To ease in/out, make $s(t) = easy(t)$ that looks like this:



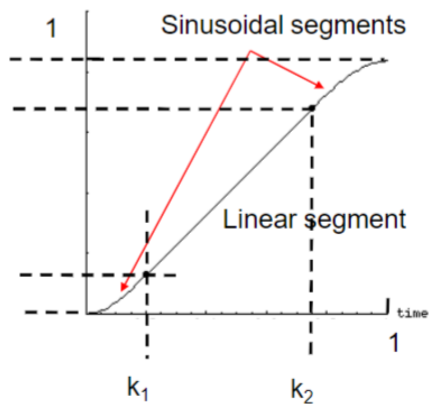
Approach 1 for Easy-In/Easy-Out

- Let's use the sine function
- Scale so s and t in $[0, 1]$

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

Approach 2: Slow in, Constant Middle, Slow Out

- Let's use 2 sinusoidal segments and a line
  - First part: Sine from $-\pi/2$ to 0
  - Second part: line (45 deg. slope)
  - Third part: Sine from 0 to $\pi/2$
- We can actually use the previous easy-in/easy-out function between $[0, 0.5]$ and $[0.5, 1]$



- We need to scale the timing and the values to ensure continuous first derivatives!

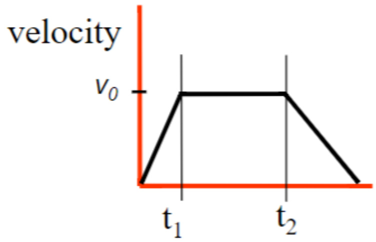
Ensuring Continuous Derivatives

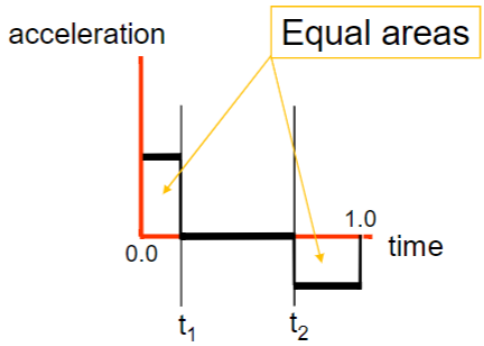
- Constraints

$$S(k_1) = k_1$$
$$S(k_2) = k_2$$
$$S'(k_1+) = S'(k_1-)$$
$$S'(k_2+) = S'(k_2-)$$

Approach 3: Parabolic Ease-in/Ease-out

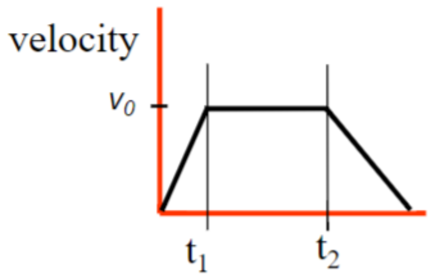
- Sine functions are expensive, so...
  - Constant Acceleration
  - Constant intermediate velocity
  - Constant deceleration





Velocity and Distance

- Velocity:



  - $v = v_0 \frac{t}{t_1}$ $0.0 < t < t_1$
  - $v = v_0$ $t_1 < t < t_2$
  - $v = v_0(1.0 - \frac{t-t_2}{1-t_2})$ $t_2 < t < 1$

- Distance:
  - The distance traveled is the area under the velocity curve

$$1. = \frac{1}{2}v_0 t_1 + v_0(t_2 - t_1) + \frac{1}{2}v_0(1 - t_2)$$

  - Integrate the velocity to find the distance at time t

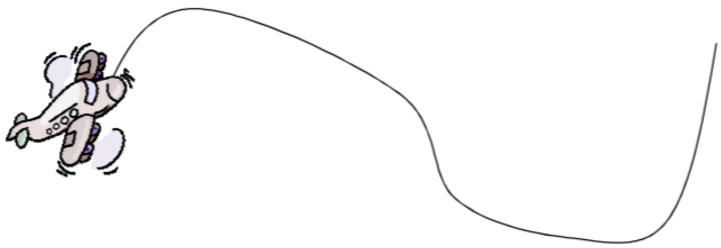
$$\begin{aligned} d(t) &= v_0 \frac{t^2}{2t_1}, && 0 < t < t_1 \\ d(t) &= v_0 \frac{t_1}{2} + v_0(t - t_1), && t_1 < t < t_2 \\ d(t) &= v_0 \frac{t_1}{2} + \left(v_0 - \frac{v_0 \frac{t-t_2}{1-t_2}}{2} \right)(t - t_2), && t_2 < t < 1 \end{aligned}$$

General Velocity Curves

- Velocity curves can have arbitrary shapes; for example, Splines
- If the animator works with velocity Curves
  - The curve must maintain the average velocity otherwise the distance changes

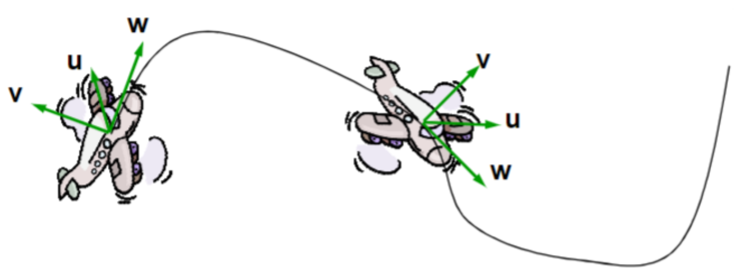
Path Following

- What's wrong with this?



Frenet Frame

- Partial solution: Moving right-handed system along path: (u, v, w)



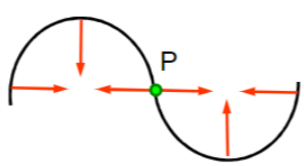
How Do WE Compute u, v, w?

$w = P'(t)$	tangent along the derivative
$u = P'(t) \times P''(t)$	perpendicular to both tangent $P'(t)$ and curvature $P''(t)$
$v = w \times u$	perpendicular to both w and u

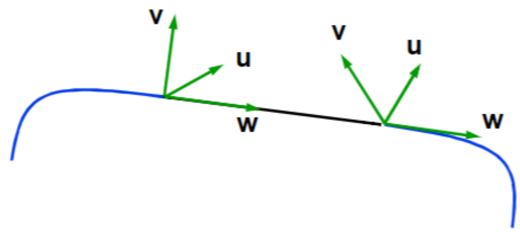
- Normalize the vectors!

Discontinuous Curvature

- Example: Two semi-circles
  - Curvature discontinuous at P



Straight Segments

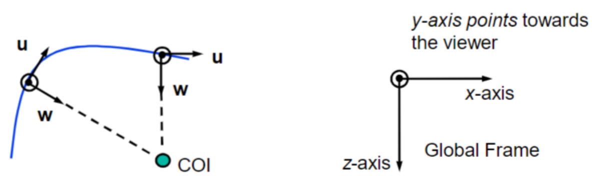


- $P''(t) = 0$
- Solution: Interpolate frames before and after straight segments
  - They only differ by a rotation around w

Other Limitations

- Camera motion
  - Bumpy ride makes the camera look up / down
  - Often people look ahead and not along the tangent; e.g., driving along a curve

Adding a Center of Interest



$w = COI - P(t)$
$u = y - axis \times w$
$v = w \times u$(parallel to y-axis)

- (The textbook has the formulas for negative w)