Quaternions

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

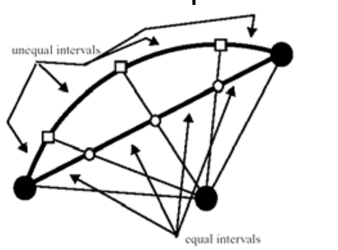
Rotating Vectors Using Quaternions

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

$$Rot_q(Rot_p(v)) = q^{-1} \cdot (p^{-1} \cdot v \cdot p) \cdot q$$
$$= ((pq)^{-1} \cdot v \cdot (pq))$$
$$= Rot_{pq}(v)$$

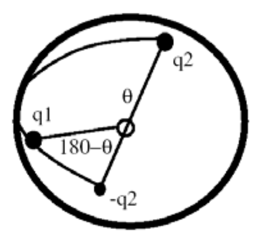
$$Rot^{-1}(Rot(v)) = q \cdot (q^{-1} \cdot v \cdot q) \cdot q^{-1} = v$$

Interpolation of Rotations using Quaternion Representation



Interpolation of Rotations using Quaternion Representation

$$\cos\theta = q1 \bullet q2 = s1 \cdot s2 + v1 \bullet v2$$



$$Slerp(q1, q2, u) = ((\sin((1-u)\cdot\theta))/(\sin\theta))\cdot q1 + (\sin(u\cdot\theta))/(\sin\theta)\cdot q2$$

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?

The problem

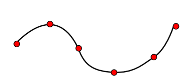
- Imagine an animator wants an object to be at position (-5,0,0) at frame 22 and at position (5,0,0) at frame 67.
  - We want to generate the position values in between frames 22 and 67
  - What is the animator also wants the object to start at 0 velocity at frame 22 and accelerate to reach a maximum speed at frame 34, and finally stop at frame 67.

Interpolation Considerations

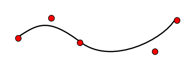
- Interpolation vs. Approximation
- Complexity (i.e. degree of the polynomial)
- Continuity
- Global vs. Local Control

Interpolation vs. Approximation

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



Interpolated



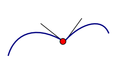
Approximated

Continuity

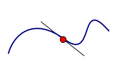
- Parametric equations:

$$x = x(u), \quad y = y(u), \quad z = z(u), \quad u_1 \le u \le u_2$$

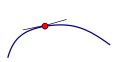
- Parametric continuity: Continuity properties of curve segments.
  - Zero order: Curves intersects at one end-point: C^0



  - First order: C^0 *and* curves has same tangent at intersection: C^1



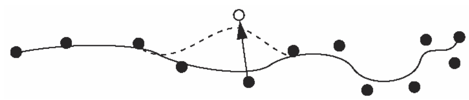
  - Second order: C^0 , C^1 *and* curves has same second order derivative: C^2



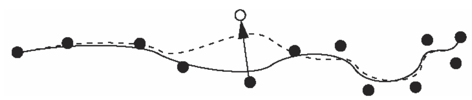
Continuity

- 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:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \qquad 0 \le u \le 1$$
$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} = \mathbf{U} \cdot \mathbf{C}$$

- General form: $x(u) = \mathbf{U} \cdot \mathbf{M_s} \cdot \mathbf{M_g}$
- $\mathbf{M}_s$: spline transformation (blending functions)
  $\mathbf{M}_g$: geometric constraints (control points)

Natural Cubic Splines

- Interpolation of n+1 control points. n curve segments. 4n coefficients to determine
- Second order continuity. 4 equation for each of n-1 common points:

$$x_k(1) = \mathbf{p}_k, \quad x_{k+1}(0) = \mathbf{p}_k, \quad x'_k(1) = x'_{k+1}(0), \quad x''_k(1) = x''_{k+1}(0)$$

  4n equations required, 4n-4 so far.
- Starting point condition, end point condition.

$$x_1(0) = \mathbf{p}_0, \quad x_n(1) = \mathbf{p}_n$$

- Assume second derivative 0 at end-points or add phantom control points p_{-1}, p_{n+1}.

$$x''_1(0) = 0, \quad x''_n(1) = 0$$

- Write 4n equations for 4n unknown coefficients and solve.
- Changes are not local. A control point effects all equations.
- Expensive. Solve 4n system of equations for changes.

Hermite Interpolation

- End point constraints for each segment is given as:

$$\mathbf{P}(0) = \mathbf{p}_k, \quad \mathbf{P}(1) = \mathbf{p}_{k+1}, \quad \mathbf{P}'(0) = \mathbf{Dp}_k, \quad \mathbf{P}'(1) = \mathbf{Dp}_{k+1},$$

- Control point positions and first derivatives are given as constraints for each end-point.

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} \qquad \mathbf{P}'(u) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} \qquad \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix}$$

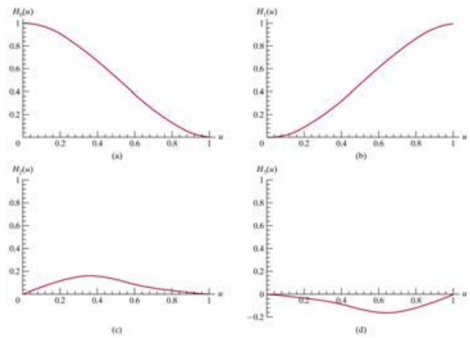
Hermite Interpolation

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix}$$

$$\mathbf{P}(u) = \mathbf{p}_k(2u^3 - 3u^2 + 1) + \mathbf{p}_{k+1}(-2u^3 + 3u^2) + \mathbf{Dp}_k(u^3 - 2u^2 + u) + \mathbf{Dp}_{k+1}(u^3 - u^2)$$

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

Hermite blending functions



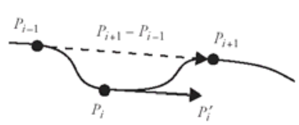
Hermite Interpolation

- 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

$$P_i' = (1/2)\cdot(P_{i+1} - P_{i-1})$$



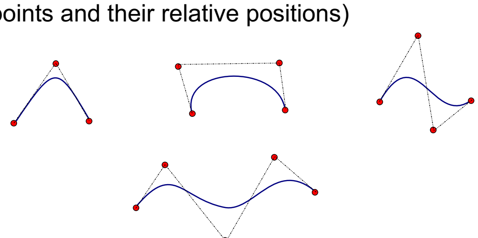
Catmull-Rom Splines

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

$$\mathbf{P}(u) = \mathbf{p}_{k-1}(-0.5u^3 + u^2 - 0.5u) + \mathbf{p}_k(1.5u^3 - 2.5u^2 + 1) + \\ + \mathbf{p}_{k+1}(-1.5u^3 + 2u^2 + 0.5u) + \mathbf{p}_{k+2}(0.5u^3 - 0.5u^2)$$

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)



Bézier Curves

$$\mathbf{P}(u) = \sum_{k=0}^{n} \mathbf{p}_k \mathrm{BEZ}_{k,n}(u), \qquad 0 \le u \le 1$$

$$\mathrm{BEZ}_{k,n}(u) = \binom{n}{k} u^k (1-u)^{n-k}, \qquad \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- The coordinates of the control points are blended using Bézier blending functions $\mathrm{BEZ}_{k,n}(u)$
- 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.

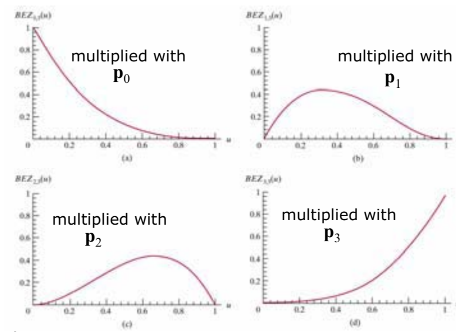
$$\mathrm{BEZ}_{0,3} = (1-u)^3 \qquad \mathrm{BEZ}_{1,3} = 3u(1-u)^2$$

$$\mathrm{BEZ}_{2,3} = 3u^2(1-u) \qquad \mathrm{BEZ}_{3,3} = u^3$$

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \mathbf{M}_{\mathrm{Bez}} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{M}_{\mathrm{Bez}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Cubic Bézier blending functions



The four Bézier blending functions for cubic curves (n=3, i.e. 4 control pts.)

Properties of Bézier curves

- Passes through start and end points

$$\mathbf{P}(0) = \mathbf{p}_0, \qquad \mathbf{P}(1) = \mathbf{p}_n$$

- First derivates at start and end are:

$$\mathbf{P}'(0) = -n\mathbf{p}_0 + n\mathbf{p}_1$$

$$\mathbf{P}'(1) = -n\mathbf{p}_{n-1} + n\mathbf{p}_n$$

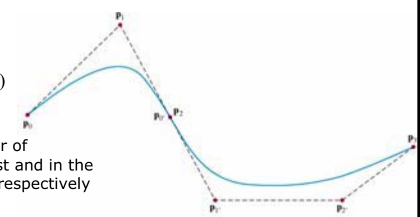
- Lies in the convex hull

Joining Bézier curves

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

$$\mathbf{p}_{0'} = \mathbf{p}_n$$

$$\mathbf{p}_{1'} = \mathbf{p}_n + \frac{n}{n'}(\mathbf{p}_n - \mathbf{p}_{n-1})$$



n and n' are the number of control points in the first and in the second curve segment respectively

- 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

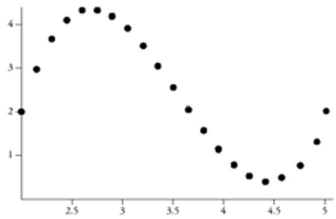
- Assume when we increase u 1 unit, we move along the curve x units (arclength). When we increase u 2 units, do we move $2x$ units on the curve?

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 $2x$ 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

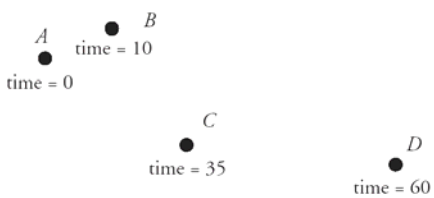
- For example, if u is the time parameter, the following positions will be generated at unit time intervals for a cubic curve



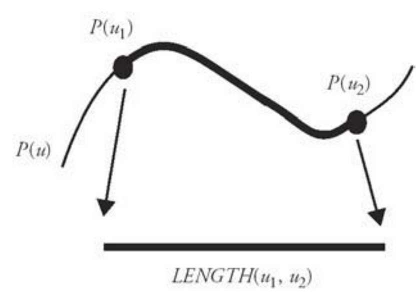
Solution

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

Time and Position



Computing the arch length



u versus arc length

- We need to find the length of the curve from its starting position for any given parametric vale:

  $s = G(u)$

- 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 $s = G(u)$

Index	Parametric Entry	Arc Length (G)
0	0.00	0.000
1	0.05	0.080
2	0.10	0.150
3	0.15	0.230
4	0.20	0.320
5	0.25	0.400
6	0.30	0.500
7	0.35	0.600
8	0.40	0.720
9	0.45	0.800
10	0.50	0.860
11	0.55	0.900
12	0.60	0.920
13	0.65	0.932
14	0.70	0.944
15	0.75	0.959
16	0.80	0.972
17	0.85	0.984
18	0.90	0.994
19	0.95	0.998
20	1.00	1.000

Finding the index closest to a given u

$$i = (int)\left(\frac{given\ parametric\ value}{distance\ between\ entries} + 0.5\right)$$
$$= (int)\left(\frac{0.73}{0.05} + 0.5\right) = 15$$

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

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

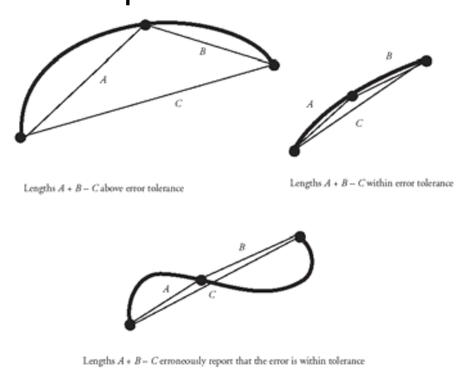
$$L = ArcLength[i] + \frac{(GivenValue - Value[i])}{(Value[i+1] - Value[i])} \cdot (ArcLength[i+1] - ArcLength[i])$$
$$= 0.944 + \frac{0.73 - 0.70}{0.75 - 0.70} \cdot (0.959 - 0.944)$$
$$= 0.953$$

Solving the other problems using the table

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

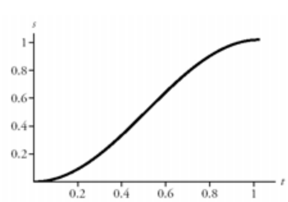
Index	Parametric Entry	Arc Length (G)
0	0.00	0.000
1	0.05	0.080
2	0.10	0.150
3	0.15	0.230
4	0.20	0.320
5	0.25	0.400
6	0.30	0.500
7	0.35	0.600
8	0.40	0.720
9	0.45	0.800
10	0.50	0.860
11	0.55	0.900
12	0.60	0.920
13	0.65	0.932
14	0.70	0.944
15	0.75	0.959
16	0.80	0.972
17	0.85	0.984
18	0.90	0.994
19	0.95	0.998
20	1.00	1.000

Adaptive subdivision

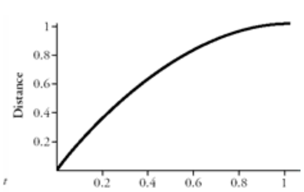


Speed Control

- Specifying the speed along the curve

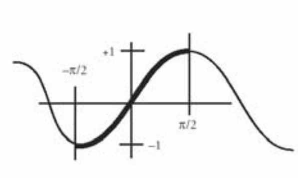


Ease-in / Ease-out

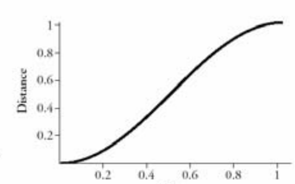


d(t)=(2-t)*t

Generating ease-in/ease-out by sine curves



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



Sine curve segment mapped to useful values

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

Ease-in/Ease-out alternative way

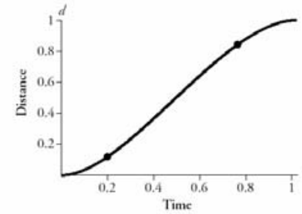
- Use high-school physics of motion

$$v = v_0 \cdot \frac{t}{t_1} \qquad 0.0 < t < t_1$$

$$v = v_0 \qquad t_1 < t < t_2$$

$$v = v_0 \cdot \left(1.0 - \frac{t - t_2}{1.0 - t_2}\right) \qquad t_2 < t < 1.0$$

Ease-in/Ease-out alternative way

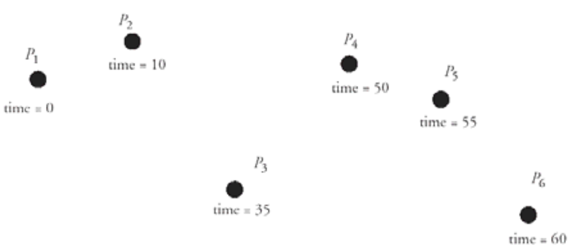


$$d = v_0 \cdot \frac{t^2}{2 \cdot t_1} \qquad 0.0 < t < t_1$$

$$d = v_0 \cdot \frac{t_1}{2} + v_0 \cdot (t - t_1) \qquad t_1 < t < t_2$$

$$d = v_0 \cdot \frac{t_1}{2} + v_0 \cdot (t_2 - t_1) + \left(v_0 - \frac{\left(v_0 \cdot \frac{t - t_2}{1 - t_2}\right)}{2}\right) \cdot (t - t_2) \qquad t_2 < t < 1.0$$

Curve fitting to position-time pairs

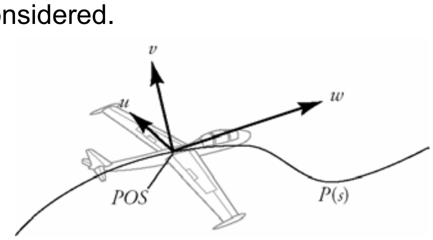


Possible solution

- 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.

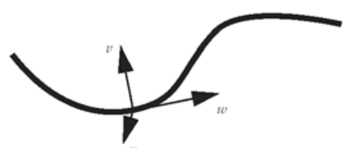
Path following

- 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).



$$w = P'(s)$$

$$u = P'(s) \times P''(s)$$

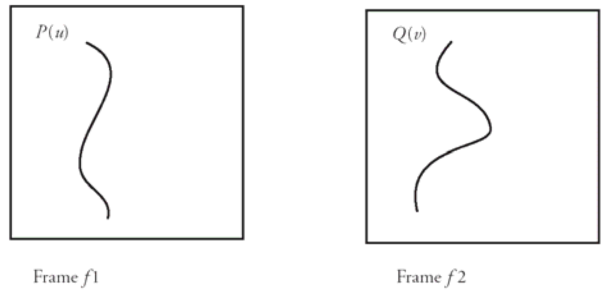
$$v = w \times u$$

Looking towards a Center of Interest

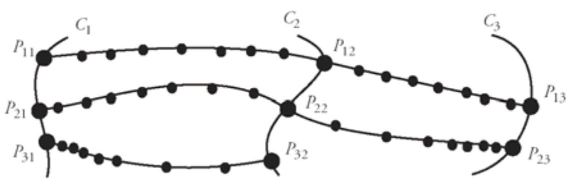
$$
\begin{aligned}
w &= COI - POS \\
u &= w \times y\text{-axis} \\
v &= u \times w
\end{aligned}
$$

Key-Frame Systems

- Shape-interpolation



Specification of point correspondences and interpolation constraints



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 of more frames of animation
- A program written in an animation language is referred to as a *script*.

Articulation Variables

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

Animation Languages

- Example:
  - Alias/Wavefront's MEL

```
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