UNIVERSITÉ Concordia UNIVERSITY

Animation for Computer Games
COMP 477/6311

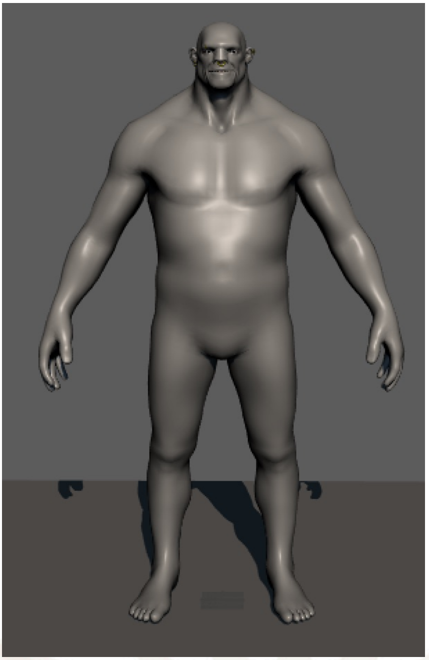
Prof. Tiberiu Popa

Keyframe Animation

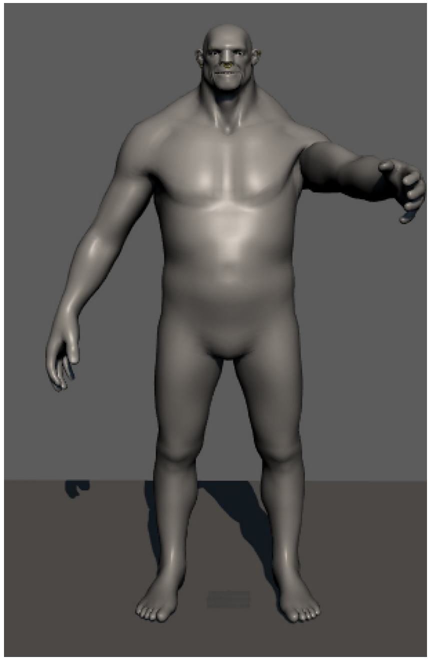
Acknowledgements

- Some of the slides in this lecture used materials from the following sources:
  - MIT Media Lab

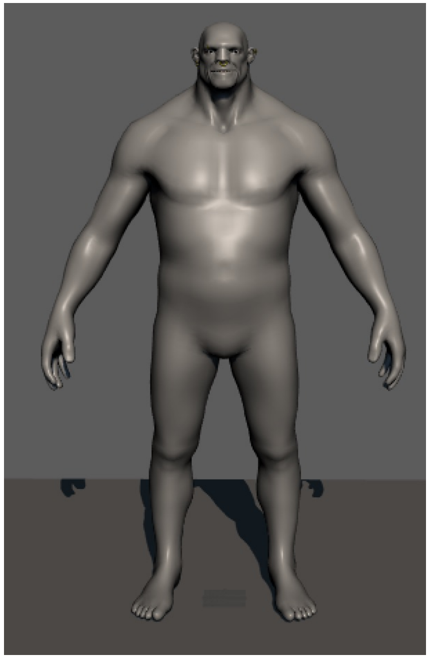
Character Animation using Keyframes



Keyframe 1



Keyframe 2

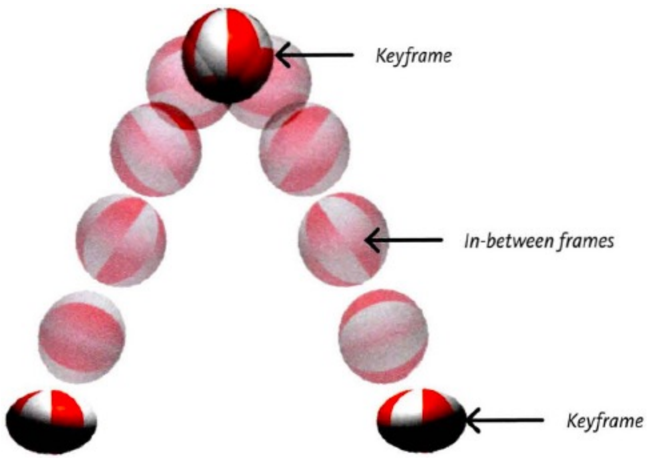


Keyframe 3

https://www.youtube.com/watch?v=c538zkwxgTQ

Character Animation using Keyframe Animation

Character Animation using Keyframe Animation



https://sites.google.com/site/bizzartso/comm-tech---keyframe-vs-cell-animation

Principles of Computer Animation

What types of measure we need to interpolate?

- Position (x, y, z) → can be interpolated separately → scalar
- Scale (sx, sy, sz) → can be interpolated separately → scalar
- Rotations → generally tricky

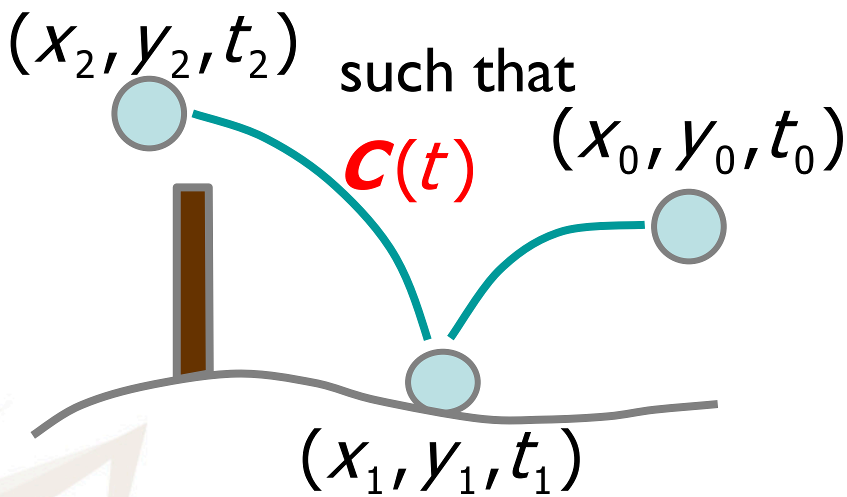
- Will discuss:
  1. Scalar interpolation
  2. Rotation interpolations

Interpolating Positions

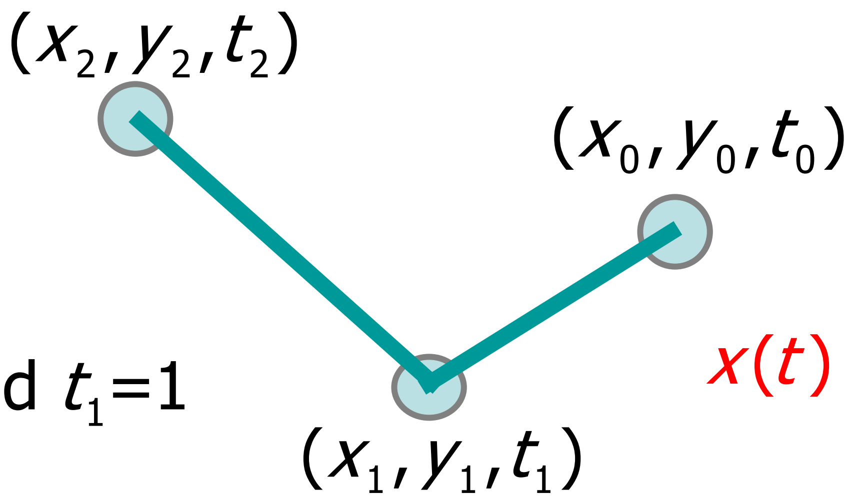
$$(x_i, y_i, t_i),\ i = 0, \ldots, n$$

$$\boldsymbol{c}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \qquad \boldsymbol{c}(t_i) = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

- Given positions:
- find curve such that



Linear Interpolation



t_0=0 and t_1=1

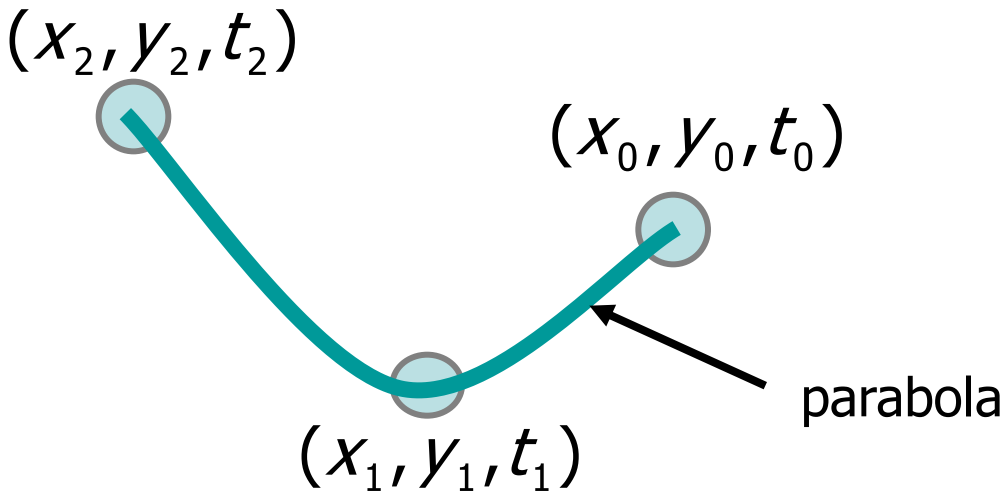
$$x(t) = x_0(1-t) + x_1 t$$

- Simple problem: linear interpolation between first two points assuming :
- The x-coordinate for the complete curve in the figure:

$$x(t) = \begin{cases} \dfrac{t_1 - t}{t_1 - t_0} x_0 + \dfrac{t - t_0}{t_1 - t_0} x_1, & t \in [t_0, t_1) \\ \dfrac{t_2 - t}{t_2 - t_1} x_1 + \dfrac{t - t_1}{t_2 - t_1} x_2, & t \in [t_1, t_2] \end{cases}$$

Derivation?

Polynomial Interpolation

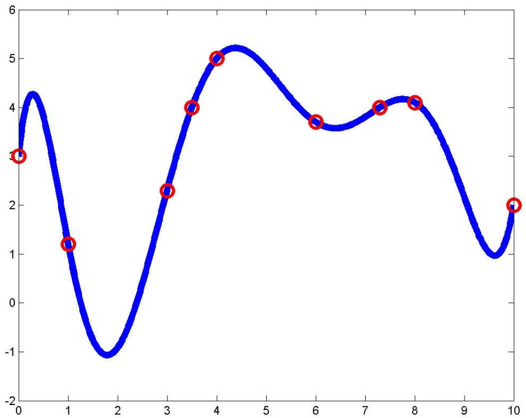


Spline Interpolation

- Lagrange polynomials of small degree are fine but high degree polynomials are too wiggly.

t

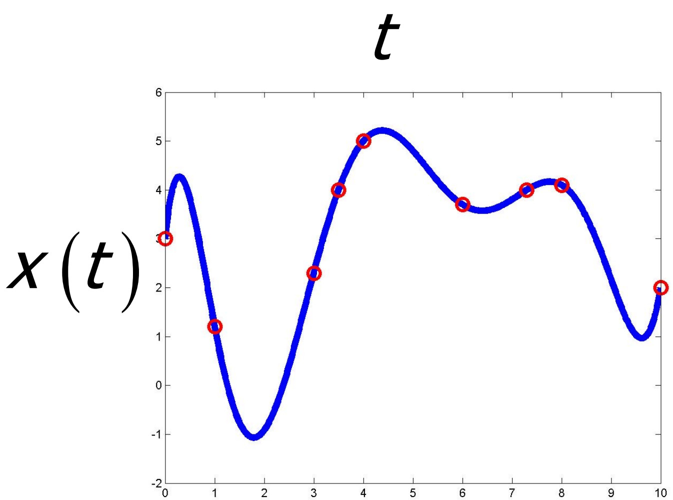
$x(t)$



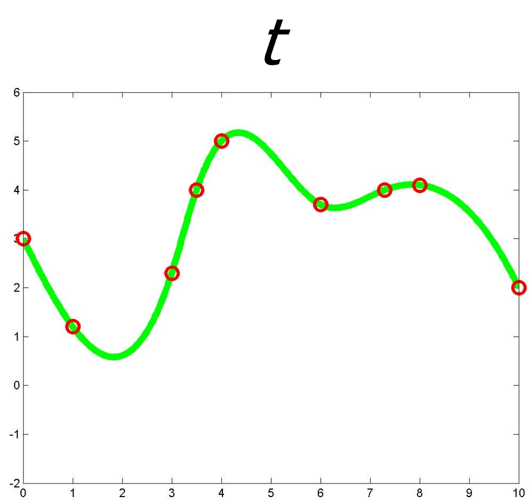
8-degree polynomial

Spline Interpolation

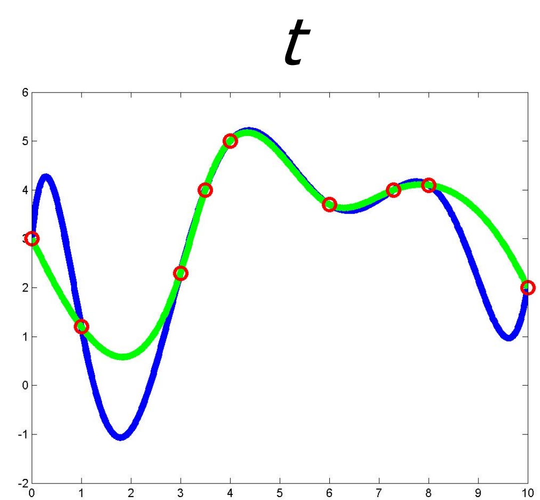
- Lagrange polynomials of small degree are fine but high degree polynomials are too wiggly.
- **Spline** (piecewise cubic polynomial) interpolation produces nicer interpolation.



8-degree polynomial



spline



spline vs. polynomial

Spline Interpolation

$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

- A cubic polynomial between each pair of points:
- Four parameters (degrees of freedom) for each spline segment.
- Why cubic?
  - Allows sufficient degrees of freedom for smoothness (i.e. position, velocity, acceleration)
  - Not too high degree (i.e. avoiding unnecessary oscillations)
- Number of parameters:
- n+1 keyframes → n cubic polynomials → 4n degrees of freedom

Spline Interpolation

$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

- Number of variables: 4n
- We have:
  - n+1 points → 2 + 2 (n-1) = 2n interpolation constraints
  - First derivative (velocity) continuity between segments = n-1 constraints
  - Second derivative (acceleration) continuity between segments = n-1 constraints
  - Total 4n-2
  - Two constraints left to the user:
    - Often specify 1st derivative at end-points
    - Hermite Splines
    - Intuitive as 1st derivative ~ velociy

Linear Interpolation

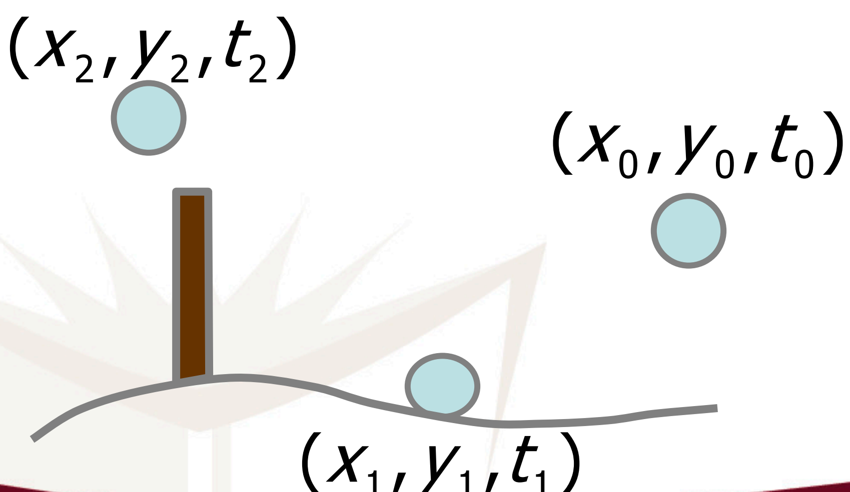
Find (8 unknowns)

$$x^1(t) = c_0^1 + c_1^1 t + c_2^1 t^2 + c_3^1 t^3$$

$$x^2(t) = c_0^2 + c_1^2 t + c_2^2 t^2 + c_3^2 t^3$$

How?

Look at constraints



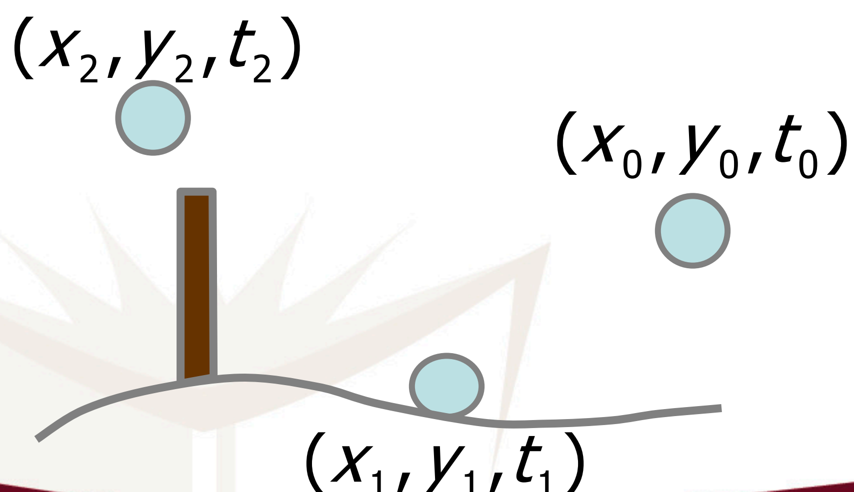
Linear Interpolation

Positional constraints (4)

$$x^1(t_0) = x_0 = c_0^1 + c_1^1 t_0 + c_2^1 t_0^2 + c_3^1 t_0^3$$
$$x^1(t_1) = x_1 = c_0^1 + c_1^1 t_1 + c_2^1 t_1^2 + c_3^1 t_1^3$$
$$x^2(t_1) = x_1 = c_0^2 + c_1^2 t_1 + c_2^2 t_1^2 + c_3^2 t_1^3$$
$$x^2(t_2) = x_2 = c_0^2 + c_1^2 t_2 + c_2^2 t_2^2 + c_3^2 t_2^3$$

How?

Look at constraints



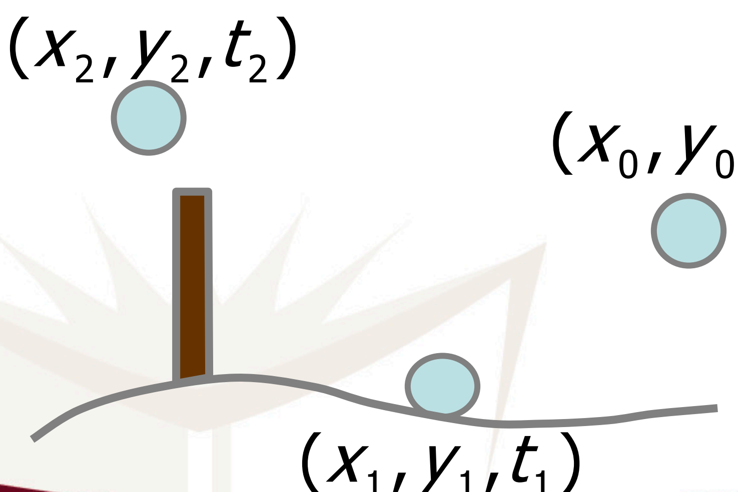
Linear Interpolation

Smooth first derivative constraints (1)

$$c_1^1 + 2c_2^1 t_1 + 3c_3^1 t_1^2 = c_1^2 + 2c_2^2 t_1 + 3c_3^2 t_1^2$$

Smooth second derivative constraints (1)

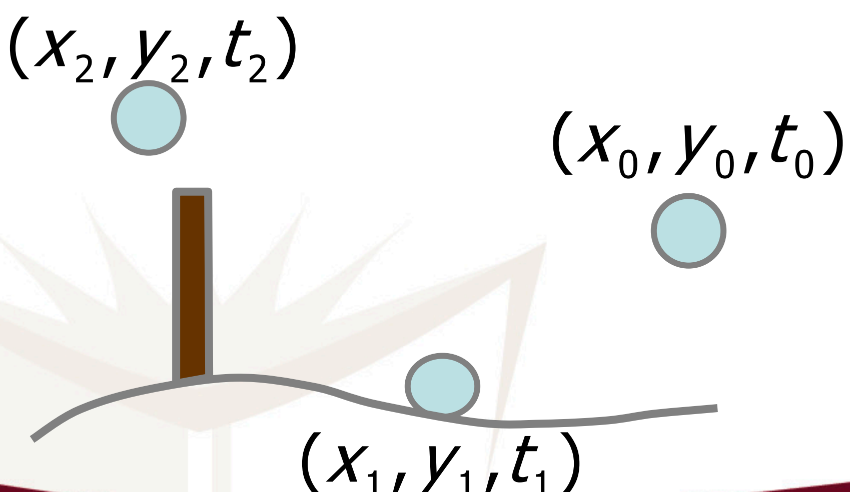
$$2c_2^1 + 6c_3^1 t_1 = 2c_2^2 + 6c_3^2 t_1$$



Linear Interpolation

User first derivative constraints at end-points (2)

$$c_1^1 + 2c_2^1 t_0 + 3c_3^1 t_0^2 = user_0$$
$$c_1^2 + 2c_2^2 t_2 + 3c_3^2 t_2^2 = user_2$$



Linear Interpolation

Putting it together:

$$x_0 = c_0^1 + c_1^1 t_0 + c_2^1 t_0^2 + c_3^1 t_0^3$$

$$x_1 = c_0^1 + c_1^1 t_1 + c_2^1 t_1^2 + c_3^1 t_1^3$$

$$x_1 = c_0^2 + c_1^2 t_1 + c_2^2 t_1^2 + c_3^2 t_1^3$$

$$x_2 = c_0^2 + c_1^2 t + c_2^2 t^2 + c_3^2 t^3$$

$$c_1^1 + 2c_2^1 t_1 + 3c_3^1 t_1^2 = c_1^2 + 2c_2^2 t_1 + 3c_3^2 t_1^2$$

$$2c_2^1 + 6c_3^1 t_1 = 2c_2^2 + 6c_3^2 t_1$$

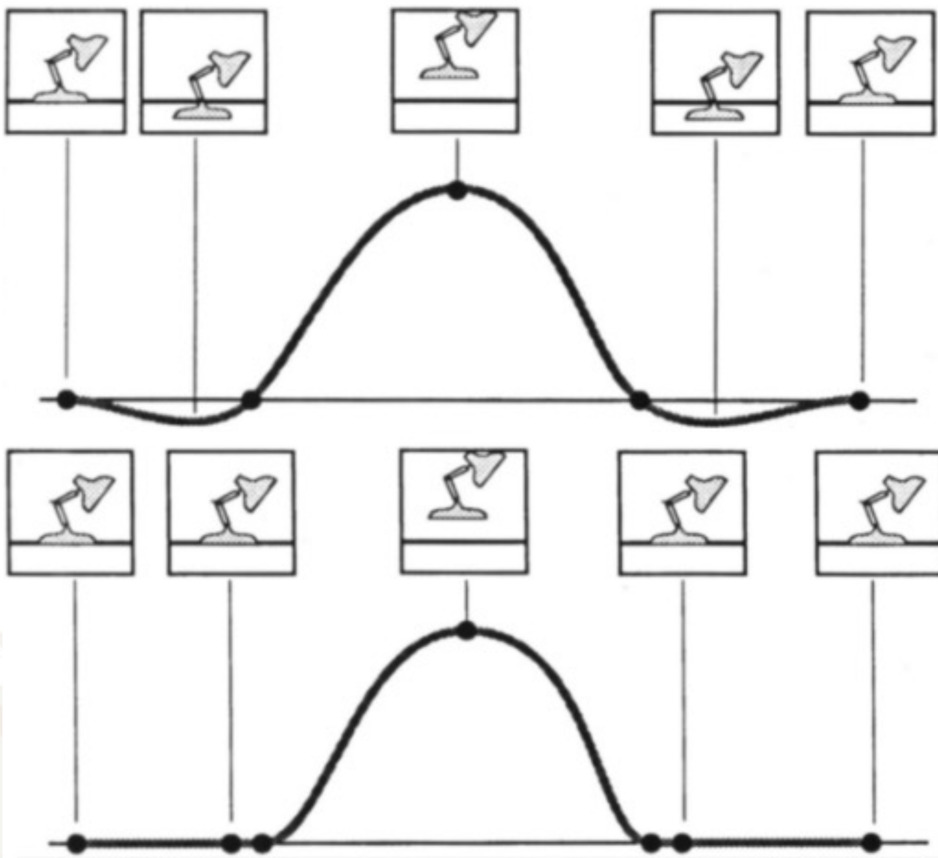
$$c_1^1 + 2c_2^1 t_0 + 3c_3^1 t_0^2 = user_0$$

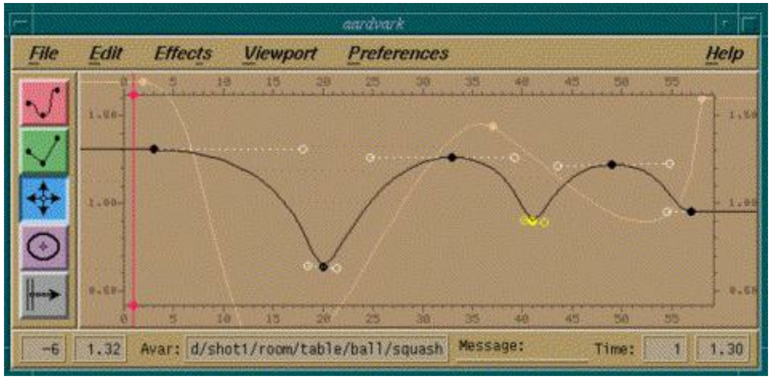
$$c_1^2 + 2c_2^2 t_1 + 3c_3^2 t_1^2 = user_2$$

Straight forward linear system!!!

Interpolating Key Frames

Interpolation is not fool proof. The splines may undershoot and cause interpenetration. The animator must also keep an eye out for these types of side-effects.





UNIVERSITÉ Concordia UNIVERSITY

Squash and stretch

- **Squash**: flatten an object or character by pressure or by its own power
- **Stretch**: used to increase the sense of speed and emphasize the squash by contrast

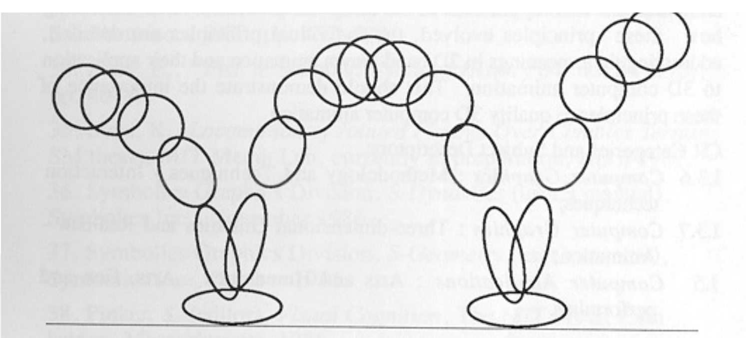


FIGURE 2. Squash & stretch in bouncing ball.

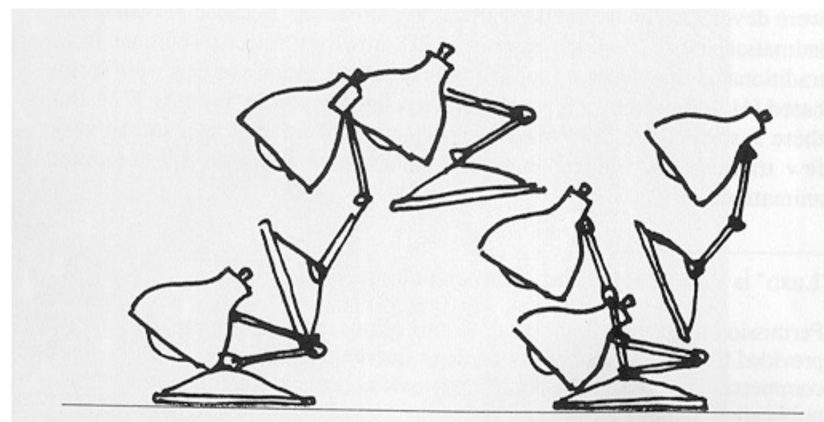


FIGURE 3. Squash & stretch in Luxo Jr.'s hop.

Timing

- Timing affects weight:
  - Light object move quickly
  - Heavier objects move slower
- Timing completely changes the interpretation of the motion. Because the timing is critical, the animators used the draw a time scale next to the keyframe to indicate how to generate the in-between frames.

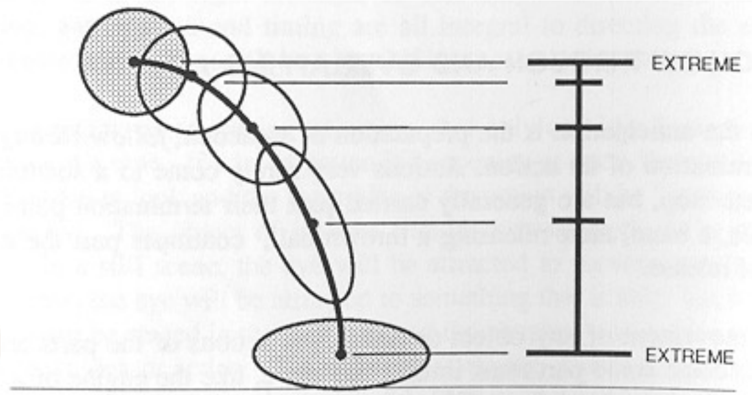


FIGURE 9. Timing chart for ball bounce.

Anticipation

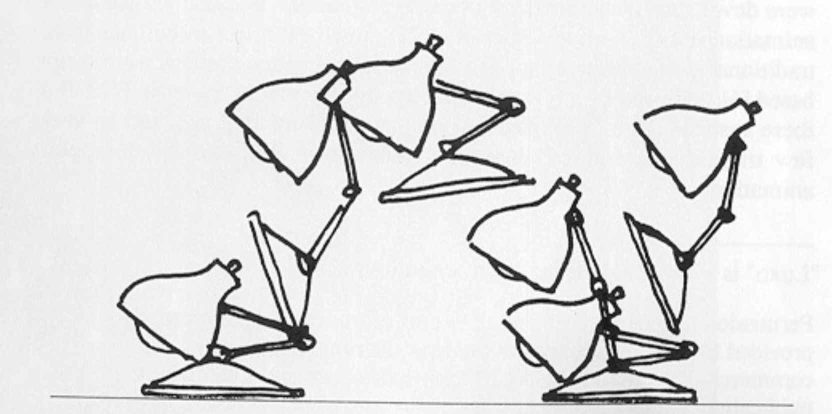


FIGURE 3. Squash & stretch in Luxo Jr.'s hop.

- An action breaks down into:
  - Anticipation
  - Action
  - Reaction
- Anatomical motivation: a muscle must extend before it can contract. Prepares audience for action so they know what to expect. Directs audience's attention. Amount of anticipation can affect perception of speed and weight.