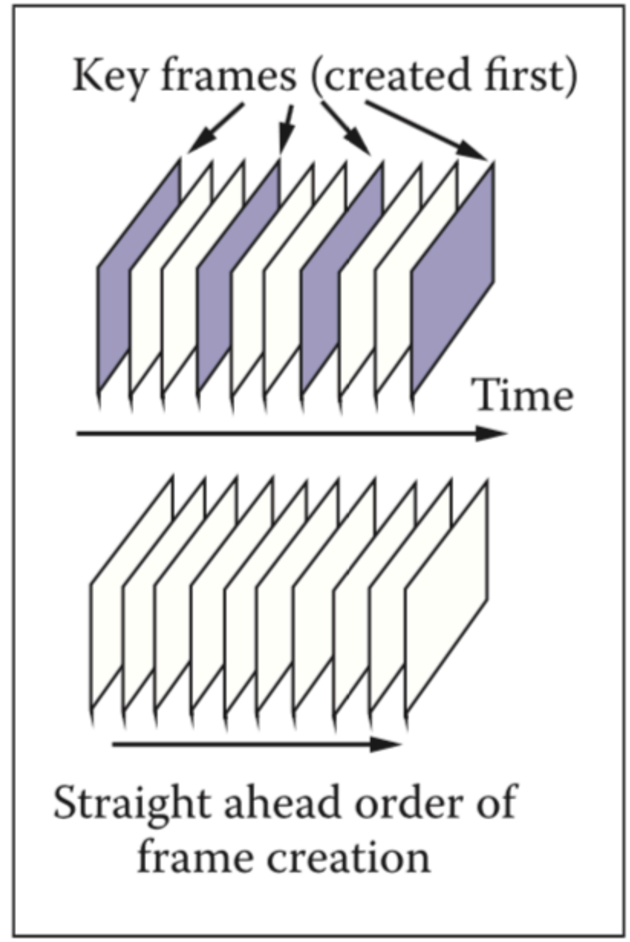
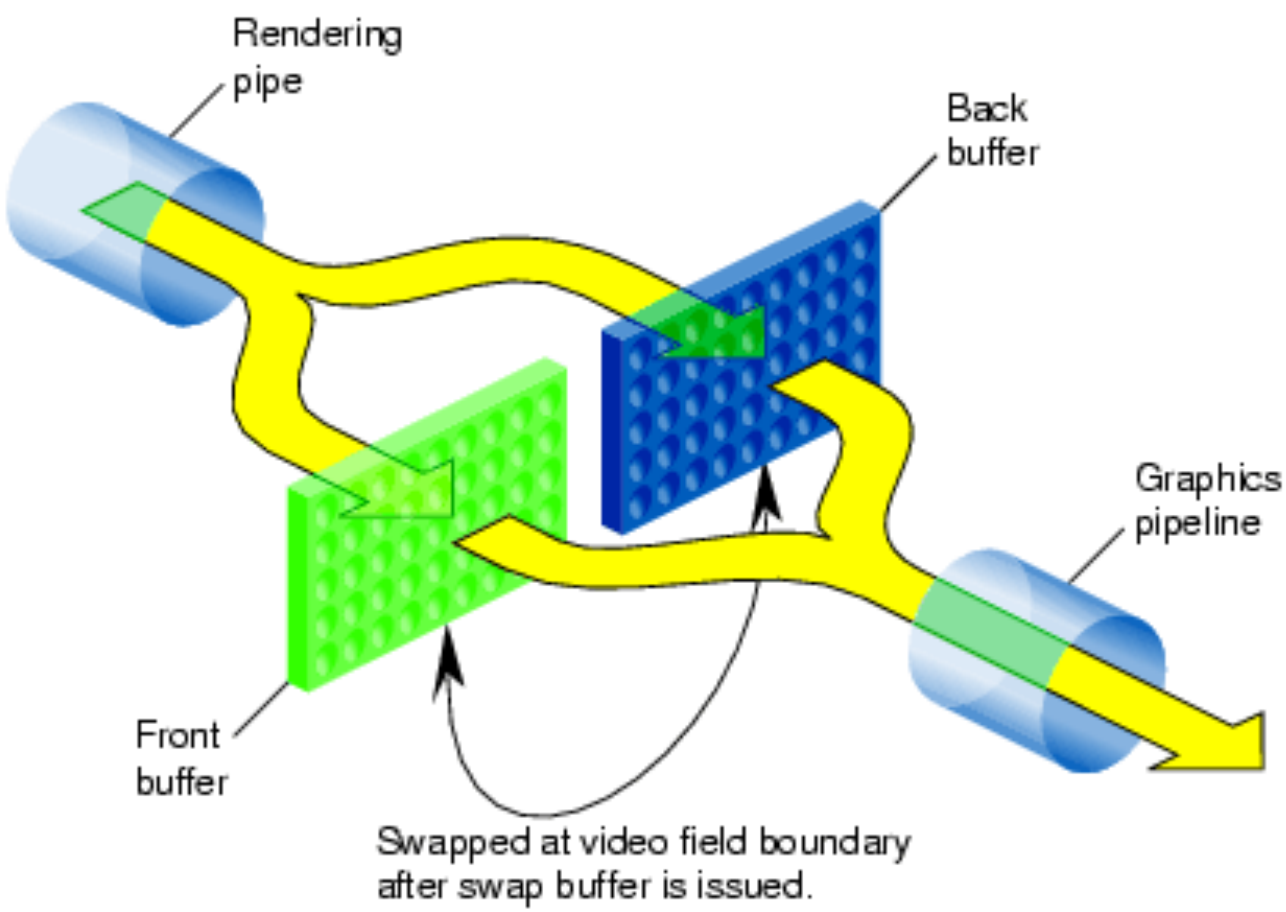
Keyframe Animation

- Idea: Draw a subset of important frames (called **key frames**) and fill in the rest with *in-betweens*
- In hand-drawn animation, the head animator would draw the poses and the assistants would do the rest
- In computer animation, the artist draws the keys and the computer does the in-betweening
  - Interpolation is used to fill in the rest!



Double Buffering

- If you draw directly to video buffer, the user will see the drawing happen
- Particularly noticeable artifacts when doing animation



http://techpubs.sgi.com/library/dynaweb_docs/linux/SGI_Developer/books/Perf_GetStarted/sgi_html/figures/double.buffering.gif

Controlling geometry conveniently

- Manually place every control point at every keyframe?
  - labor intensive
  - hard to get smooth, consistent motion
- Animate using smaller set of meaningful *degrees of freedom*
  - modeling DOFs are inappropriate for animation

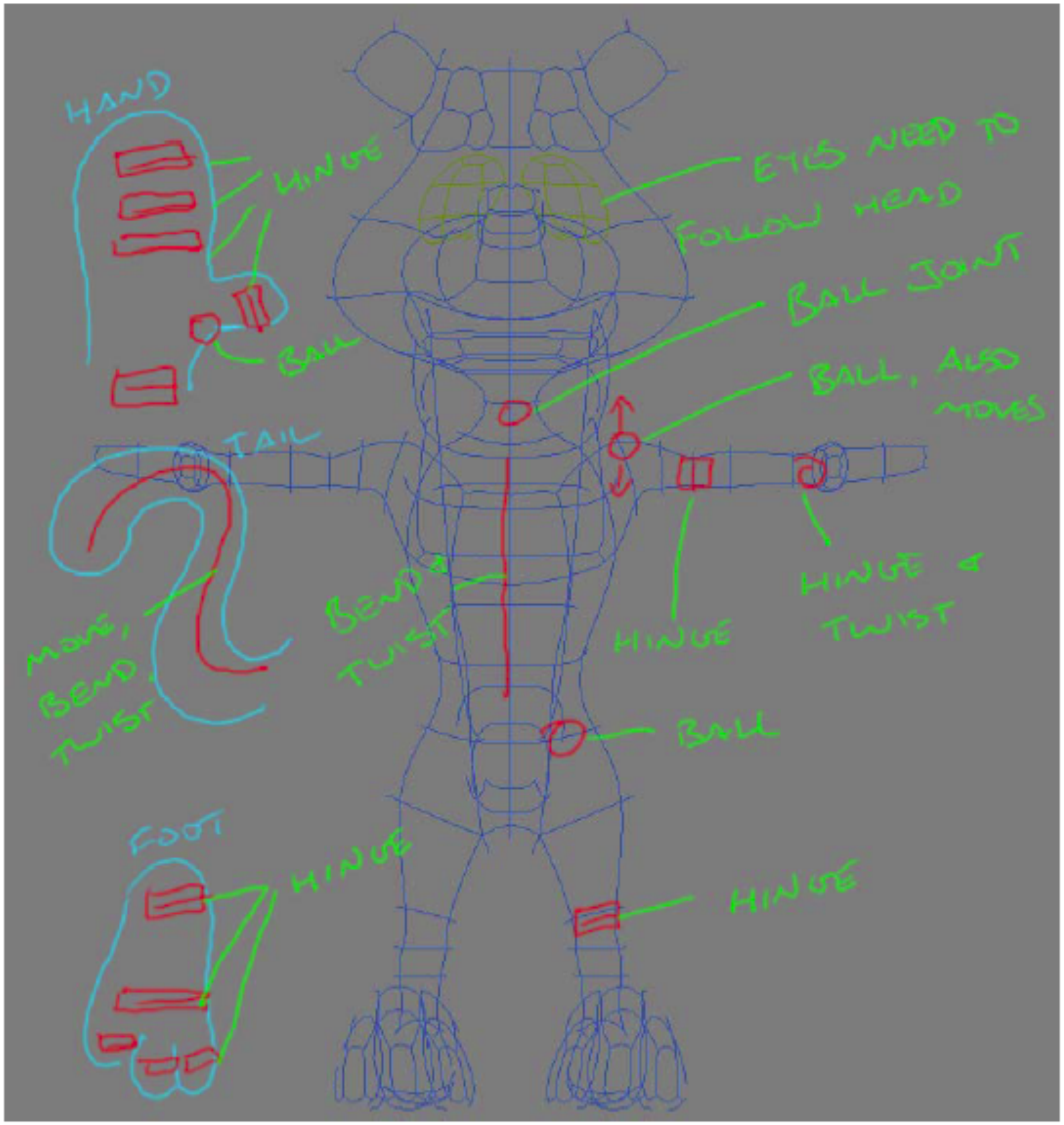
    e.g. *"move one square inch of left forearm"*
  - animation DOFs need to be higher level

    e.g. *"bend the elbow"*

Controlling shape for animation

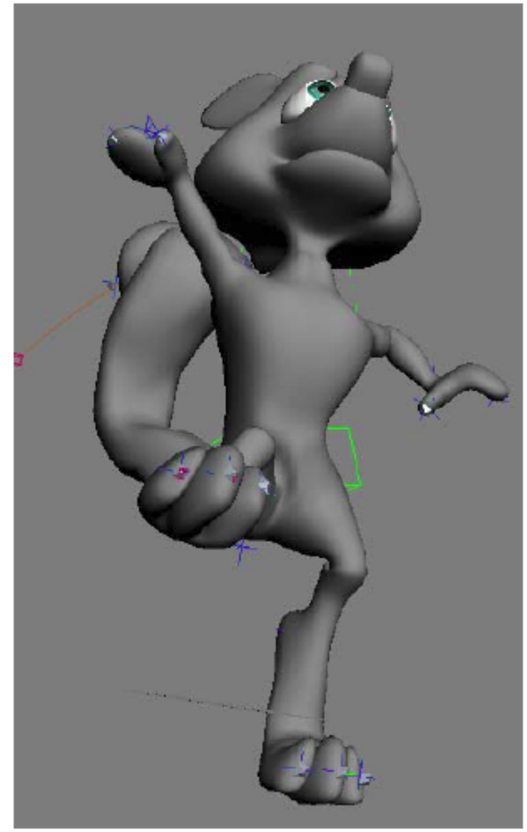
- Start with *modeling DOFs* (control points)
- *Deformations* control those DOFs at a higher level
  - Example: move first joint of second finger on left hand
- *Animation controls* control *those* DOFs at a higher level
  - Example: open/close left hand
- Both cases can be handled by the same kinds of deformers

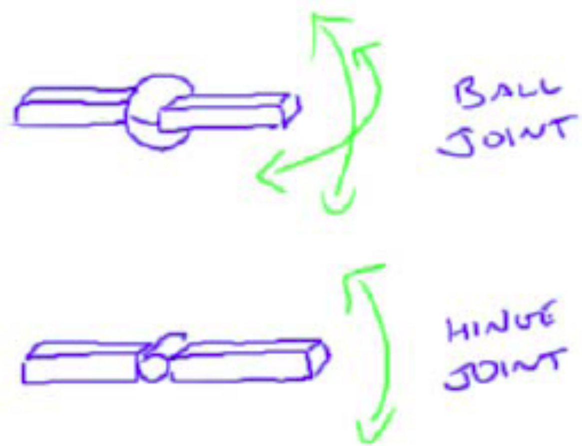
Character with DOFs



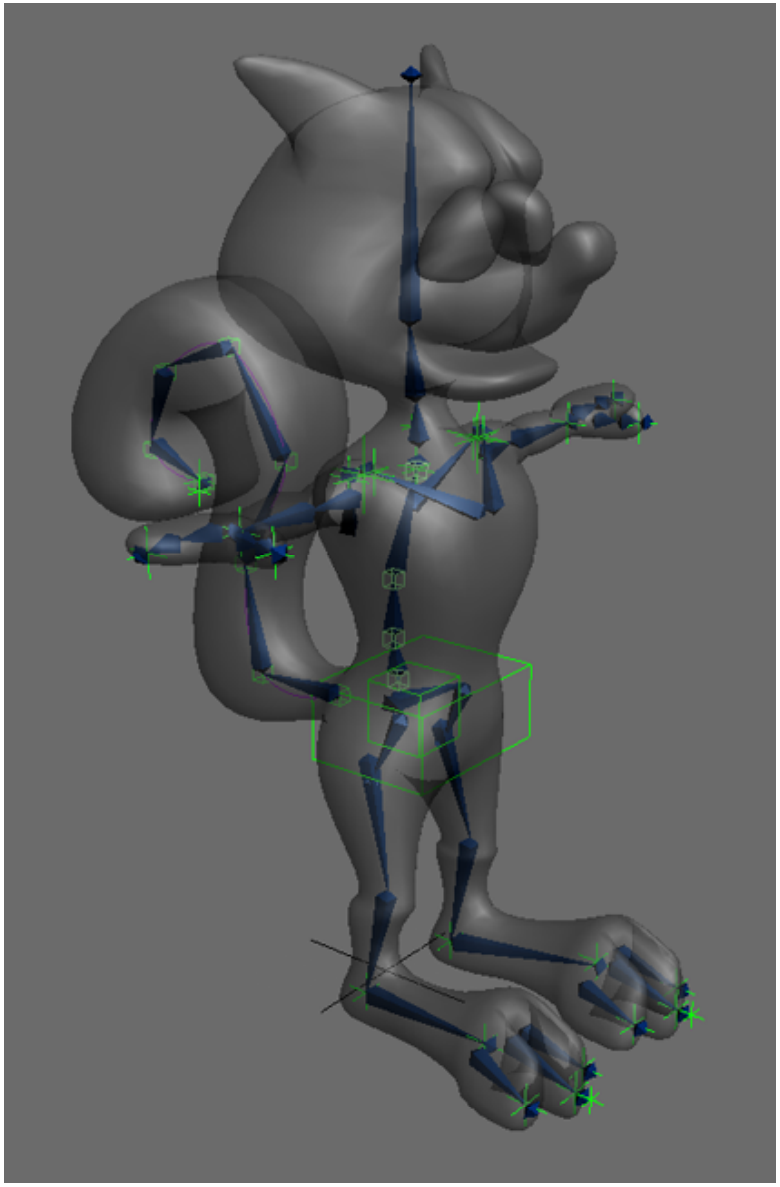
A visual description of the possible movements for the squirrel

[Greenberg/Pellacini | CIS 565]





Rigged character



[CIS 565 staff]

- Surface is deformed by a set of *bones*
- Bones are in turn controlled by a smaller set of *controls*
- The controls are useful, intuitive DOFs for an animator to use

Interpolating Rotations

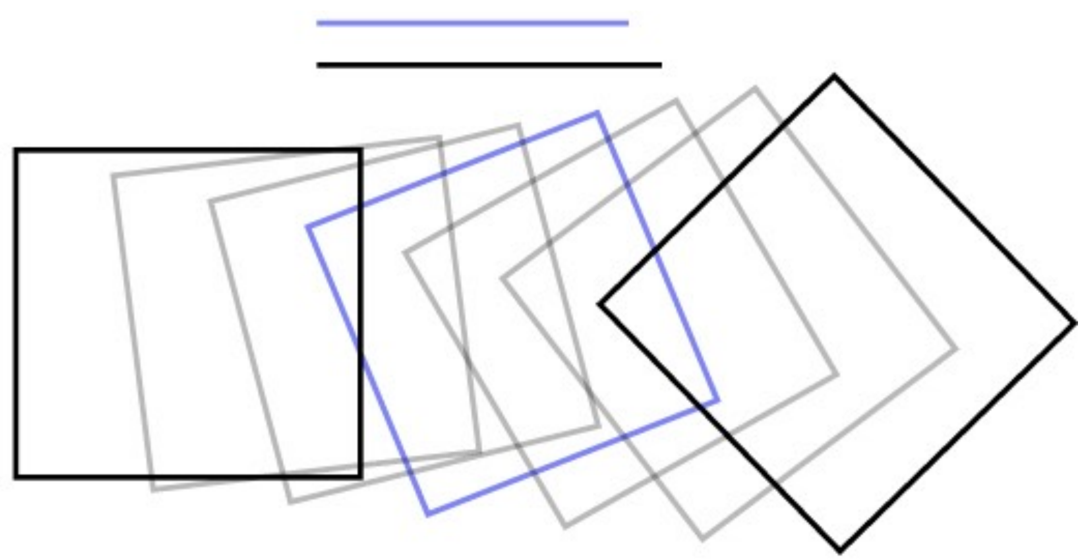
The most basic animation control

- Affine transformations position things in modeling
- Time-varying affine transformations move things around in animation
- A hierarchy of time-varying transformations is the main workhorse of animation
  - and the basic framework within which all the more sophisticated techniques are built

Interpolating transformations

- Move a set of points by applying an affine transformation
- How to animate the transformation over time?
  - interpolate the matrix entries from keyframe to keyframe?

    this is fine for translations but bad for rotations



Interpolating Rotations

$$\frac{1}{2}\underbrace{\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}}_{\text{90° CW}} + \frac{1}{2}\underbrace{\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}}_{\text{90° CCW}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

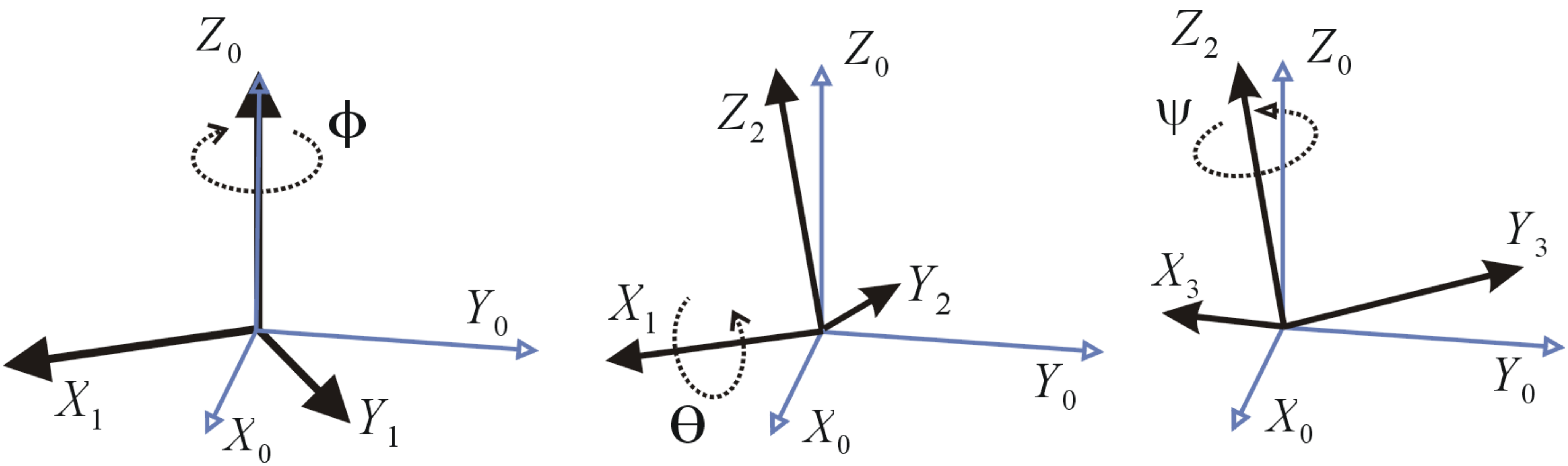
Not a rotation matrix!

Interpolating transformations

- Linear interpolation of matrices is not effective
  - leads to shrinkage when interpolating rotations
- One approach: always keep transformations in a canonical form (e.g. translate-rotate-scale)
  - then the pieces can be interpolated separately
  - rotations stay rotations, scales stay scales, all is good

Issues occurs when the source and target angles are not close to each other

Could Instead Decompose Rotation by Euler Angles

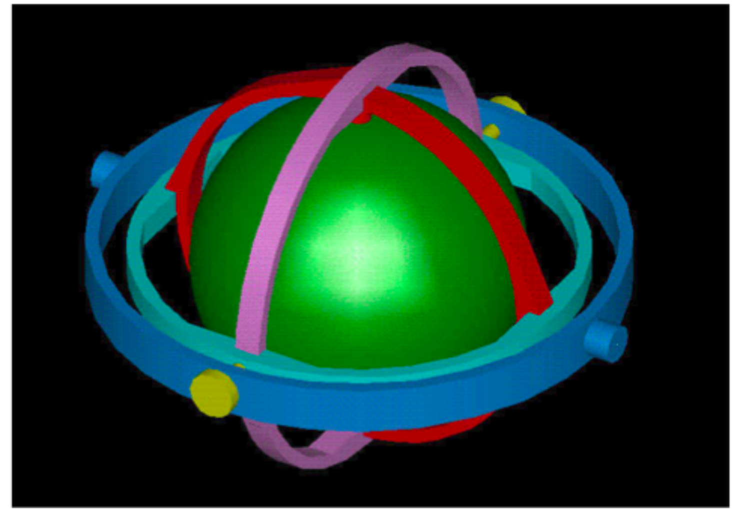


http://upload.wikimedia.org/wikipedia/commons/7/73/EulerG.png

Parameterizing rotations

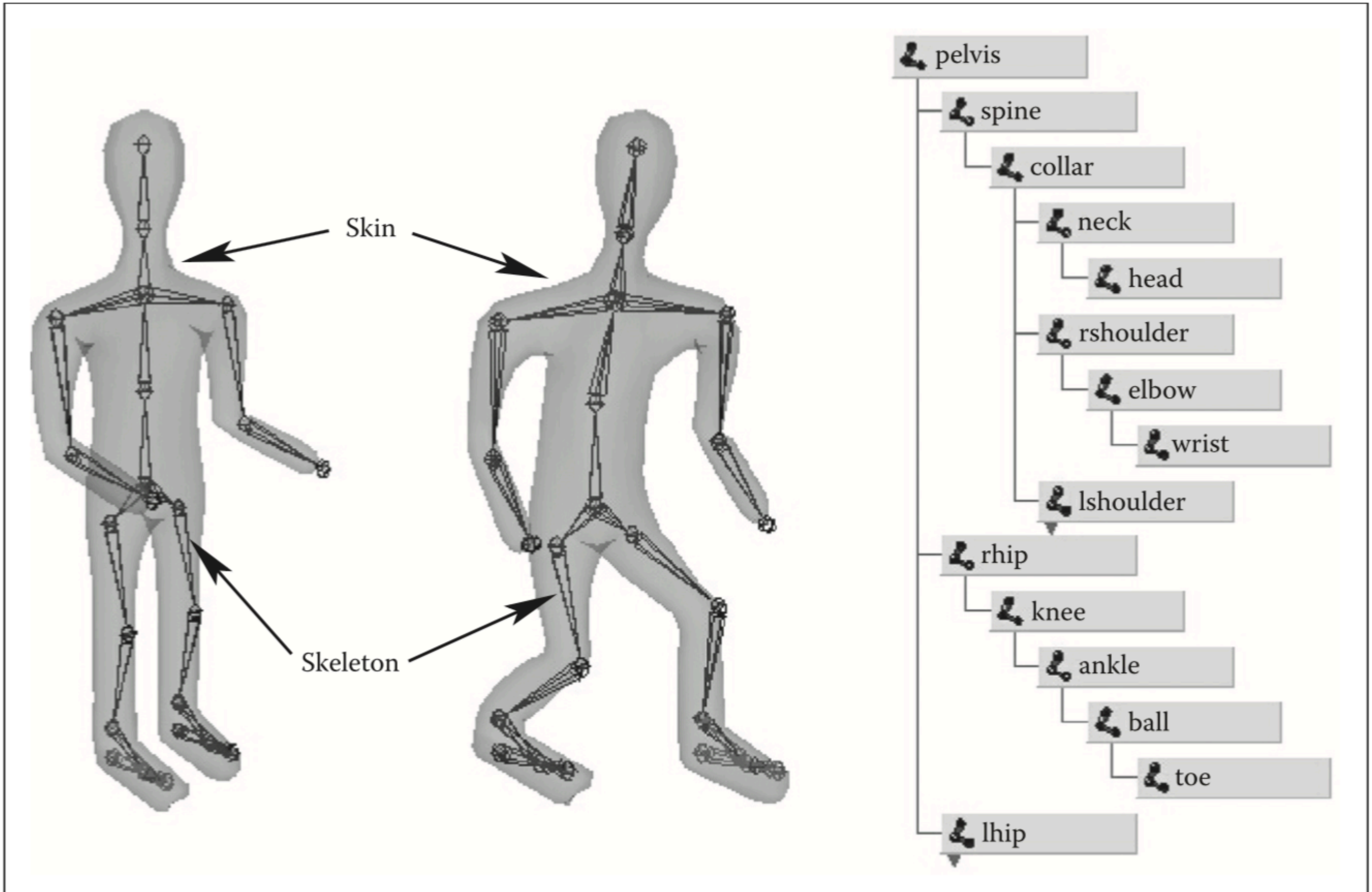
- Euler angles
  - rotate around x, then y, then z
  - nice and simple

$$R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z) R_y(\theta_y) R_x(\theta_x)$$

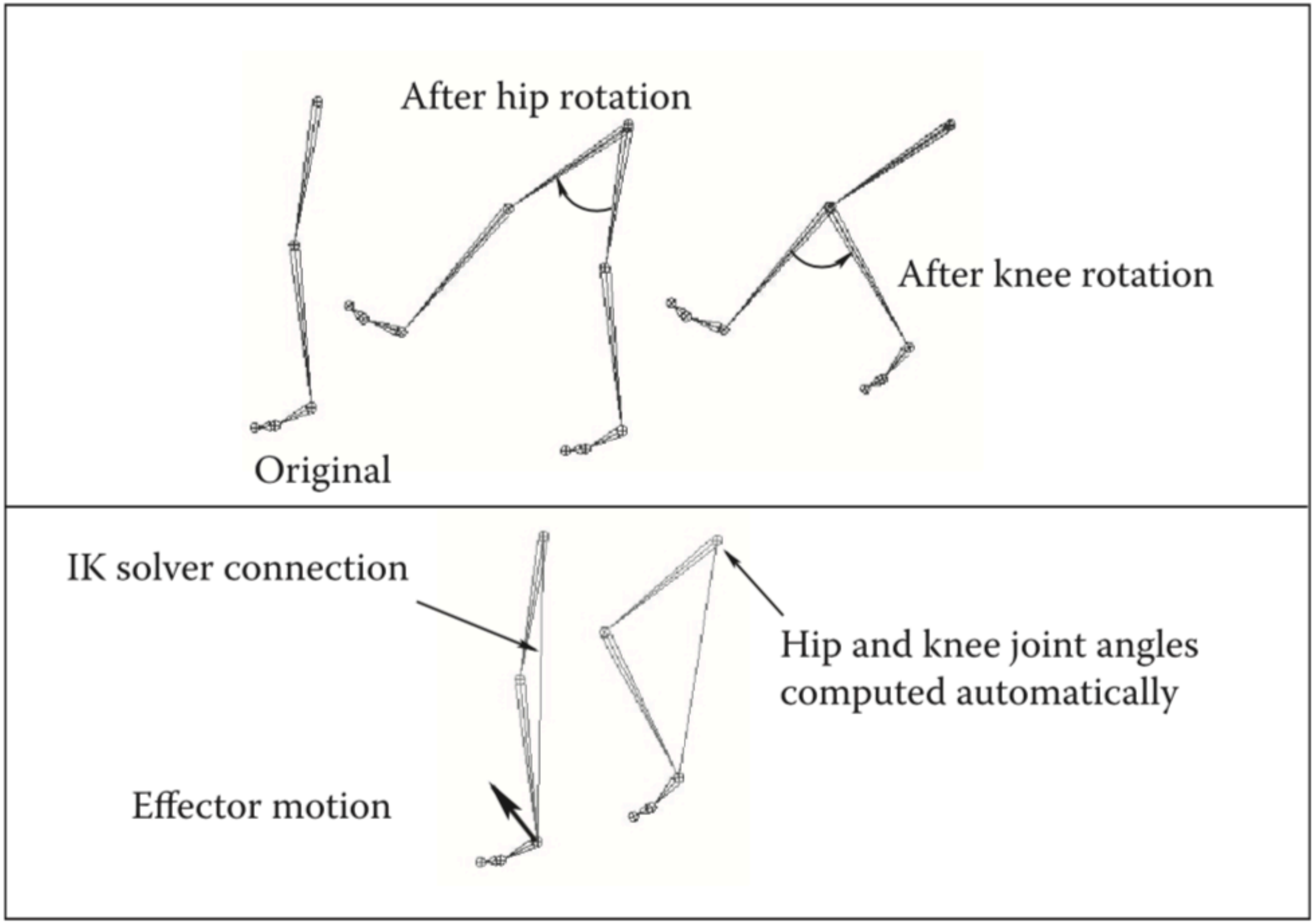


Character Animation

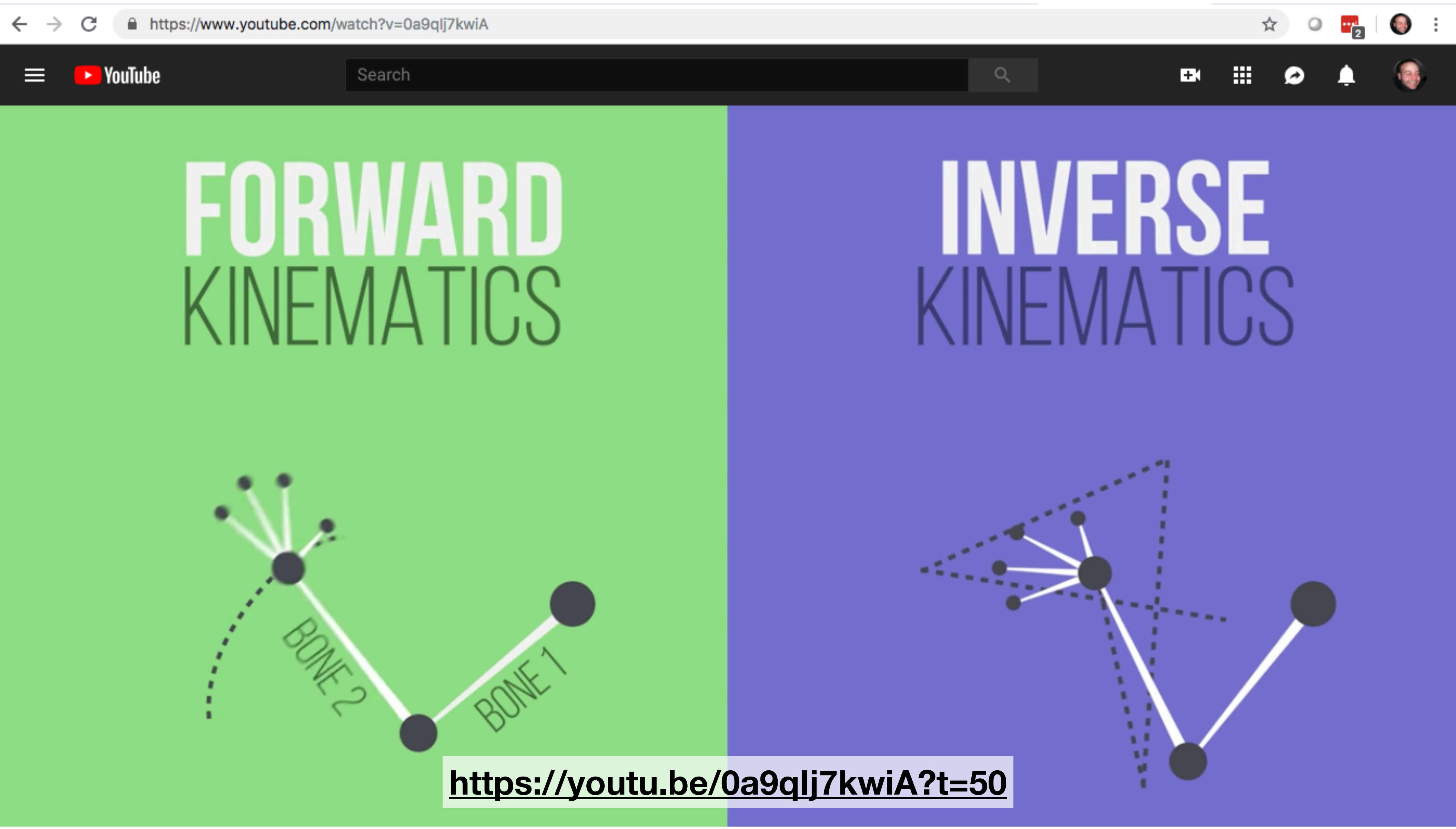
Animating w/ Skeletal Hierarchies



Forward vs. Inverse Kinematics



Inverse Kinematics Solves for all Intermediate Constraints



https://youtu.be/0a9qlj7kwiA?t=50

Physics-Based Animation

Animation vs. Simulation

- Animation methods use scripted actions to make objects change
- Simulation: simulate physical laws by associating physical properties to objects
- Solve for physics to achieve (predict) realistic effects

Using Particle Systems

- Idea: Represent the physics on the simplest possible entity: particles
- Used for effects like smoke, fire, water, sparks, and more
- Plenty of other approaches, this is just one family

Integration Algorithm 1

Calculating Particle State from Forces: First attempt

- Use forces to update velocity: $\vec{v}(t+h) = \vec{v}(t) + \frac{h}{m}\vec{f}(t)$
- Use old velocity to update position: $\vec{x}(t+h) = \vec{x}(t) + h\vec{v}(t)$

Physically-based Motion

Acceleration based on Newton's laws

- $\vec{f}(t) = m\vec{a}(t)$ …or, equivalently… $\vec{a}(t) = \vec{f}(t)/m$
- i.e., force is mass times acceleration

Forces are known beforehand

- e.g., gravity, springs, others….
- Multiple forces sum together
- These often depend on the position, i.e., $\vec{f}(t) \equiv \vec{f}(\vec{x}(t))$
- Sometimes velocity, too

If we know the values of the forces, we can solve for particle's state

Secure | https://processing.org/examples/simpleparticlesystem.html

Processing | p5.js | Processing.py | Processing for Android | Processing Foundation

Processing

Cover

Download
Donate

Exhibition

Reference
Libraries
Tools
Environment

Tutorials
Examples
Books
Handbook

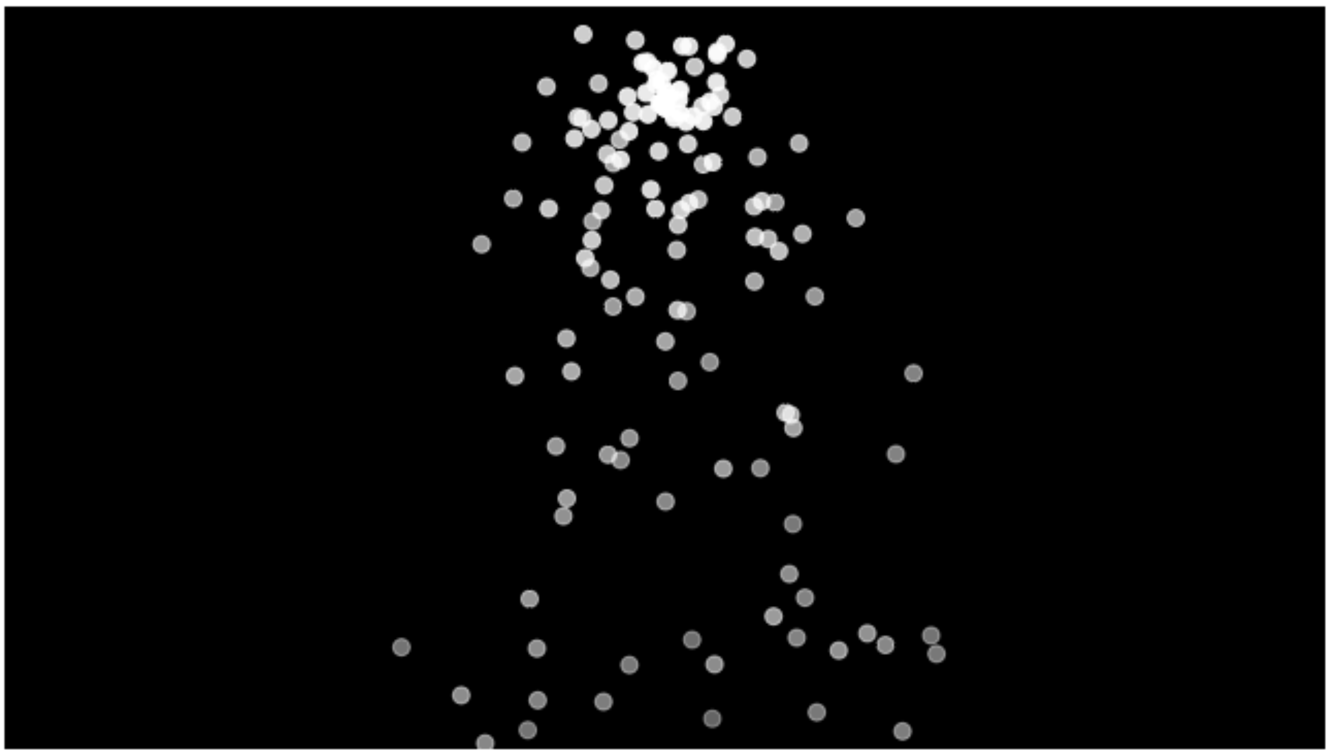
Overview
People

Shop

» Forum
» GitHub
» Issues
» Wiki
» FAQ
» Twitter
» Facebook
» Medium

← Back To List

This example is for Processing 3+. If you have a previous version, use the examples included with your software. If you see any errors or have suggestions, please let us know.

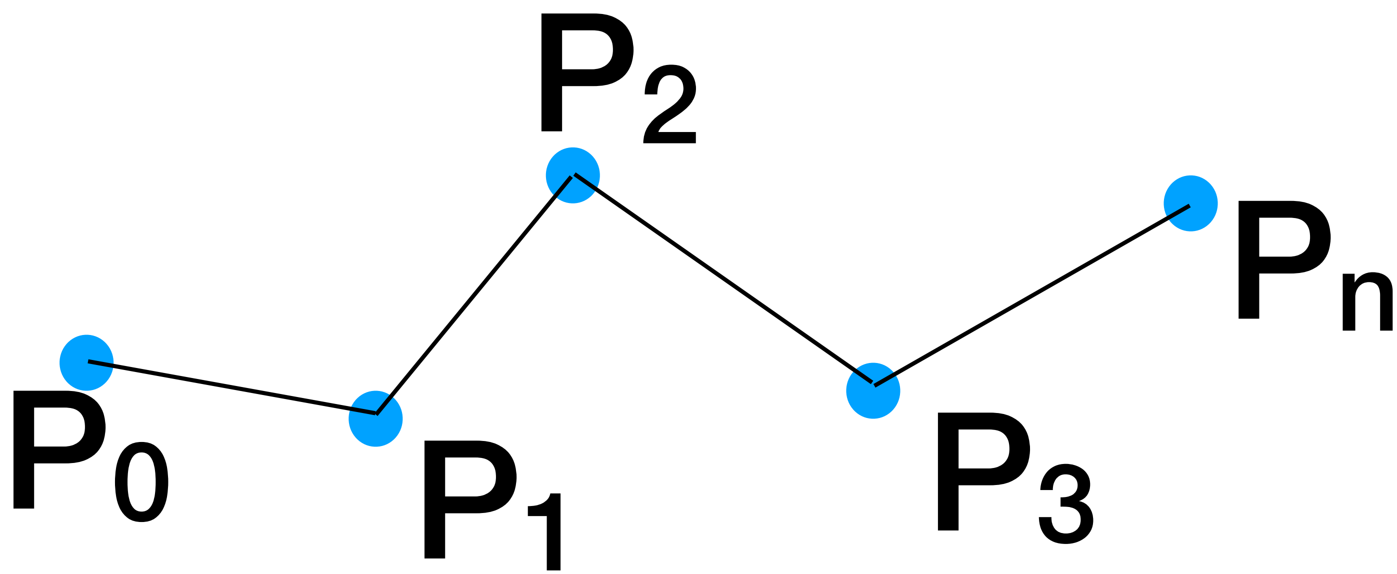


Simple Particle System by Daniel Shiffman.

https://processing.org/examples/simpleparticlesystem.html

Particles are generated each cycle through draw(), fall with gravity and fade out over time A ParticleSystem

How to create a curves that represents the points P_0 , P_1 P_2 ...P_n



Option 1: Linear interpolation. Not great for animation
Need something more smooth.
Solutions - cubic splines or B-splines

Hermite Cubic Basis

Lets look at cubic polynomial (max degree =3)

$h(t) = a\,t^3 + b\,t^2 + c\,t + d$

Let prepare actually 4 such polynomials

Curve	$h(0)$	$h(1)$	$h'(0)$	$h'(1)$
$h_{00}(t)$	1	0	0	0
$h_{01}(t)$	0	1	0	0
$h_{10}(t)$	0	0	1	0
$h_{11}(t)$	0	0	0	1

$$h_{00}(0) = 1 = d,$$
$$h_{00}(1) = 0 = a + b + c + d,$$
$$h_{00}'(0) = 0 = c,$$
$$h_{00}'(1) = 0 = 3a + 2b + c.$$

Hermite Cubic Basis

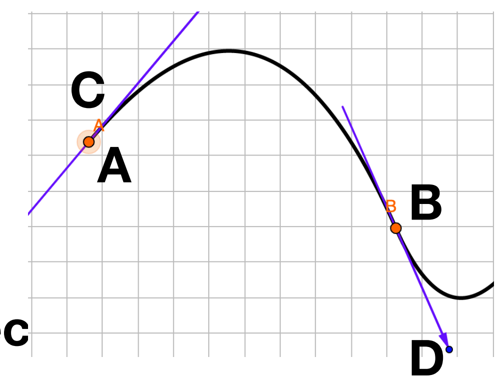
Lets look at cubic polynomial (max degree =3)

$$h(t) = a\,t^3 + b\,t^2 + c\,t + d$$

Let prepare actually 4 such polynomials

$$h_{00}(0) = 1 = d,$$
$$h_{00}(1) = 0 = a + b + c + d,$$
$$h_{00}'(0) = 0 = c,$$
$$h_{00}'(1) = 0 = 3a + 2b + c.$$

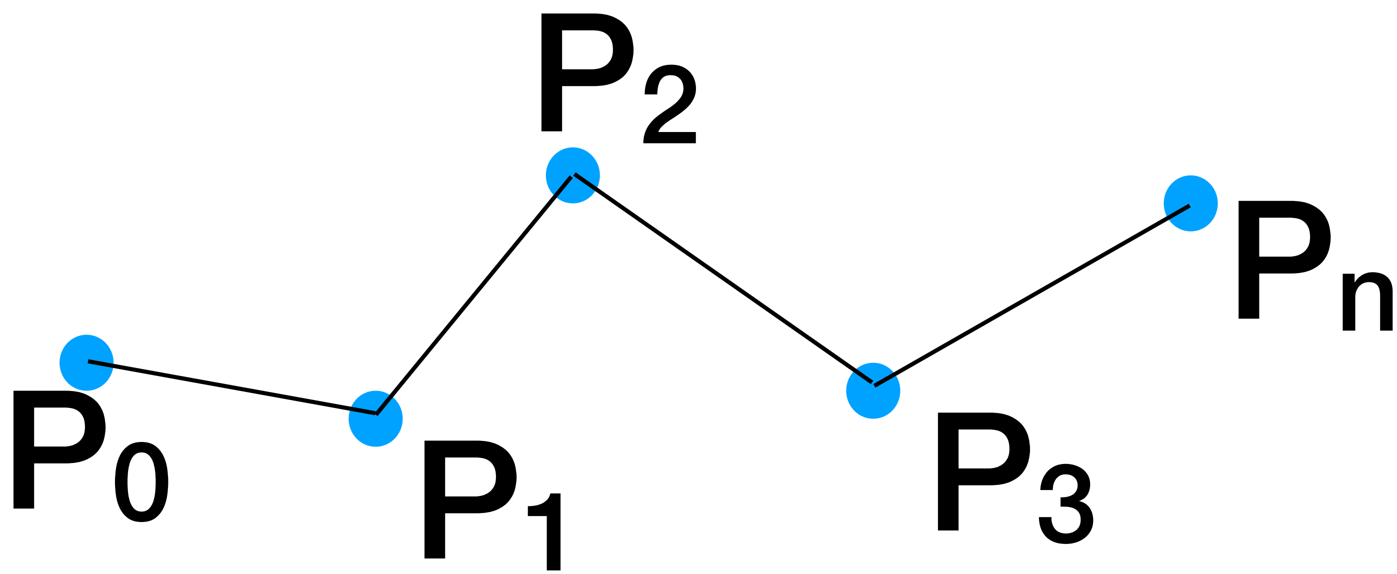
Curve	$h(0)$	$h(1)$	$h'(0)$	$h'(1)$
$h_{00}(t)$	1	0	0	0
$h_{01}(t)$	0	1	0	0
$h_{10}(t)$	0	0	1	0
$h_{11}(t)$	0	0	0	1



Now the animator could decide about the location of A,B, and the direc (all vectors)

$$h_{00}(t)\mathbf{A} + h_{11}(t)\mathbf{B} + h_{01}(t)\mathbf{C} + h_{10}(t)\mathbf{D}$$

Now we could concatenate several termites, to form a curve to connects all the points



Hermite Cubic Basis (cont'd)

Curve	$h(0)$	$h(1)$	$h'(0)$	$h'(1)$
$h_{00}(t)$	1	0	0	0
$h_{01}(t)$	0	1	0	0
$h_{10}(t)$	0	0	1	0
$h_{11}(t)$	0	0	0	1

- Lets solve for $h_{00}(t)$ as an example.
- $h_{00}(t) = a\, t^3 + b\, t^2 + c\, t + d$ must satisfy the following four constraints:

$$\boxed{\begin{aligned} h_{00}(0) &= 1 = d, \\ h_{00}(1) &= 0 = a + b + c + d, \\ h_{00}'(0) &= 0 = c, \\ h_{00}'(1) &= 0 = 3a + 2b + c. \end{aligned}}$$

- Four linear equations in four unknowns.

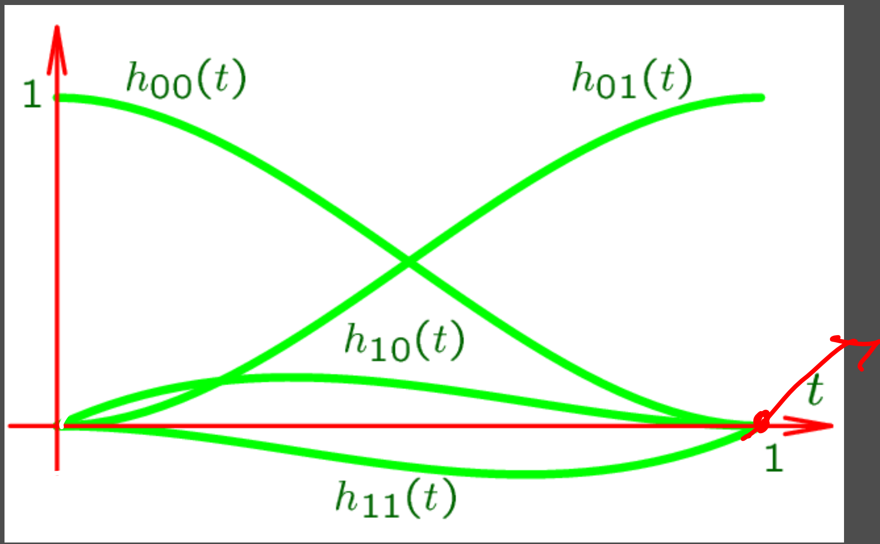
Hermite Cubic Basis

Curve	$h(0)$	$h(1)$	$h'(0)$	$h'(1)$
$h_{00}(t)$	1	0	0	0
$h_{01}(t)$	0	1	0	0
$h_{10}(t)$	0	0	1	0
$h_{11}(t)$	0	0	0	1

- The four cubics which satisfy these conditions are

$$h_{00}(t) = t^2(2t-3)+1 \qquad h_{01}(t) = -t^2(2t-3)$$
$$h_{10}(t) = t(t-1)^2 \qquad h_{11}(t) = t^2(t-1)$$

- Obtained by solving four linear equations in four unknowns for each basis function
- **Prove**: Hermite cubic polynomials are linearly independent and form a basis for cubics



$$\boxed{C(t) = P_0 h_{00}(t) + P_1 h_{01}(t) + T_0 h_{10}(t) + T_1 h_{11}(t)}$$

Hermite Cubic Basis (cont'd)

- Let $C(t)$ be a cubic polynomial defined as the linear combination:

- Then $C(0) = P_0,\ \ C(1) = P_1,\ \ C'(0) = T_0,\ \ C'(1) = T_1$

- To generate a curve through P_0 & P_1 with slopes T_0 & T_1, use

$$C(t) = P_0 h_{00}(t) + P_1 h_{01}(t) + T_0 h_{10}(t) + T_1 h_{11}(t)$$