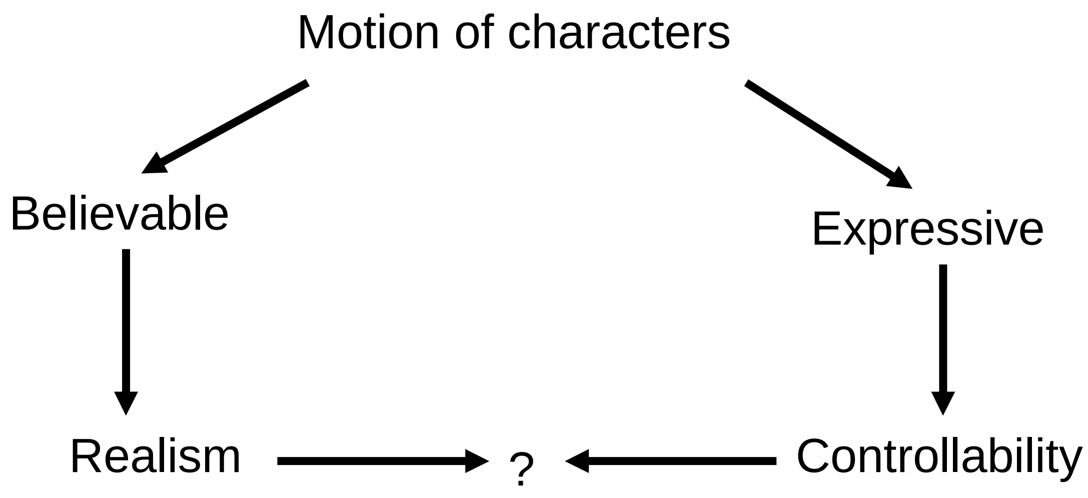
Kinematical Animation

lionel.reveret@inria.fr

2013-14

3D animation in CG

- Goal : capture visual attention

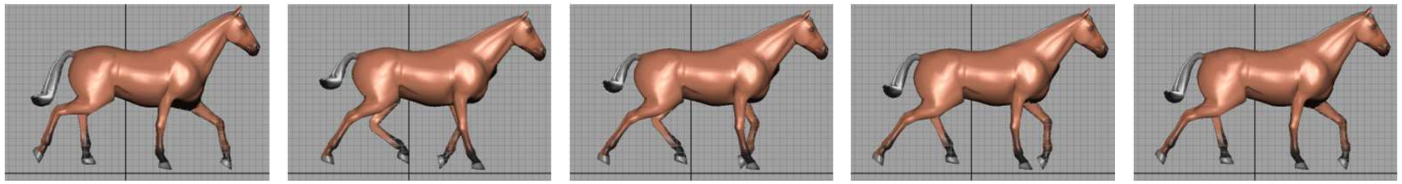


Limits of purely physical simulation :
- little interactivity
- high complexity for expressive characters

animation in 3D CG

- Two heritages
  - Cartoons
    - How to represent motion on a 2D visualization
  - Robotics
    - Mathematics foundation of 3D motion

Heritage from cartoon



- Sampled motion, film framework (24 images per second)

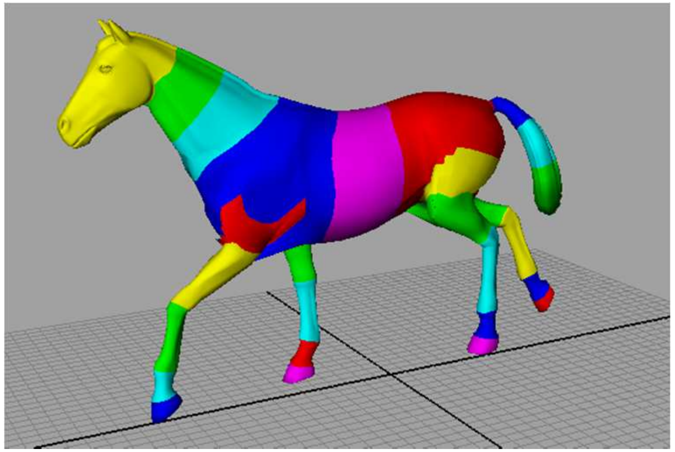
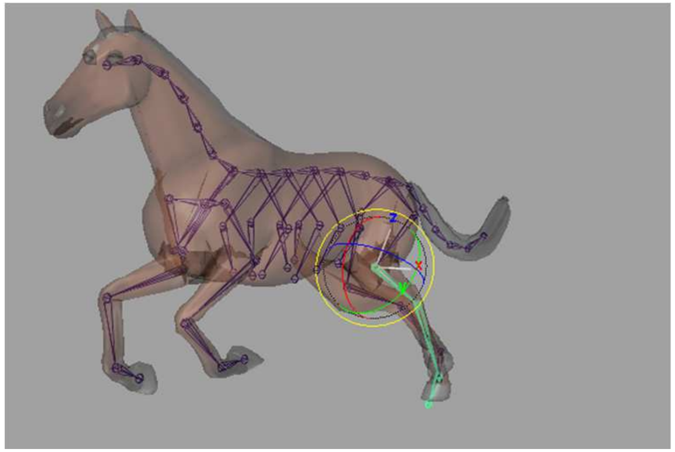
- Animation workflow from photographs (Muybridge, Marey)

Early Mathematics of motion

- Etienne-Jules Marey (1830-1904)
  - – physiologist
  - – inventor of chronophotography (1882)
    - cinematography invented in 1895 (L. Lumière)

The “graphical method”:
one motion can be represented by a curve

Heritage from robotics

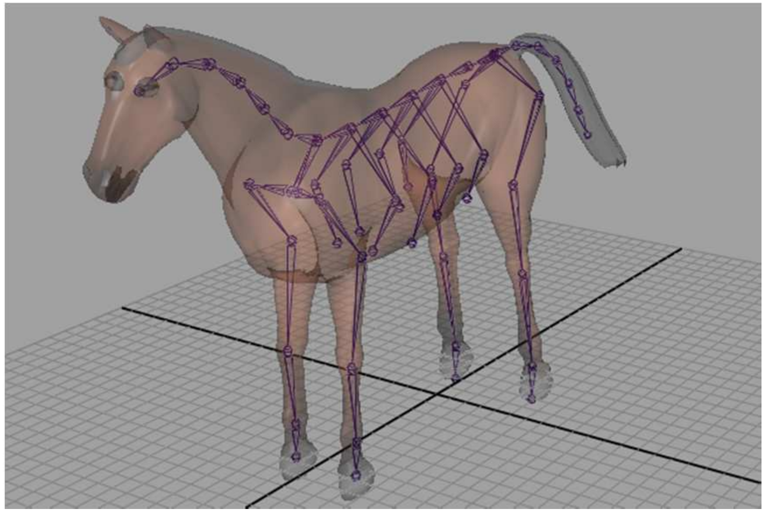
Chain of rigid articulations

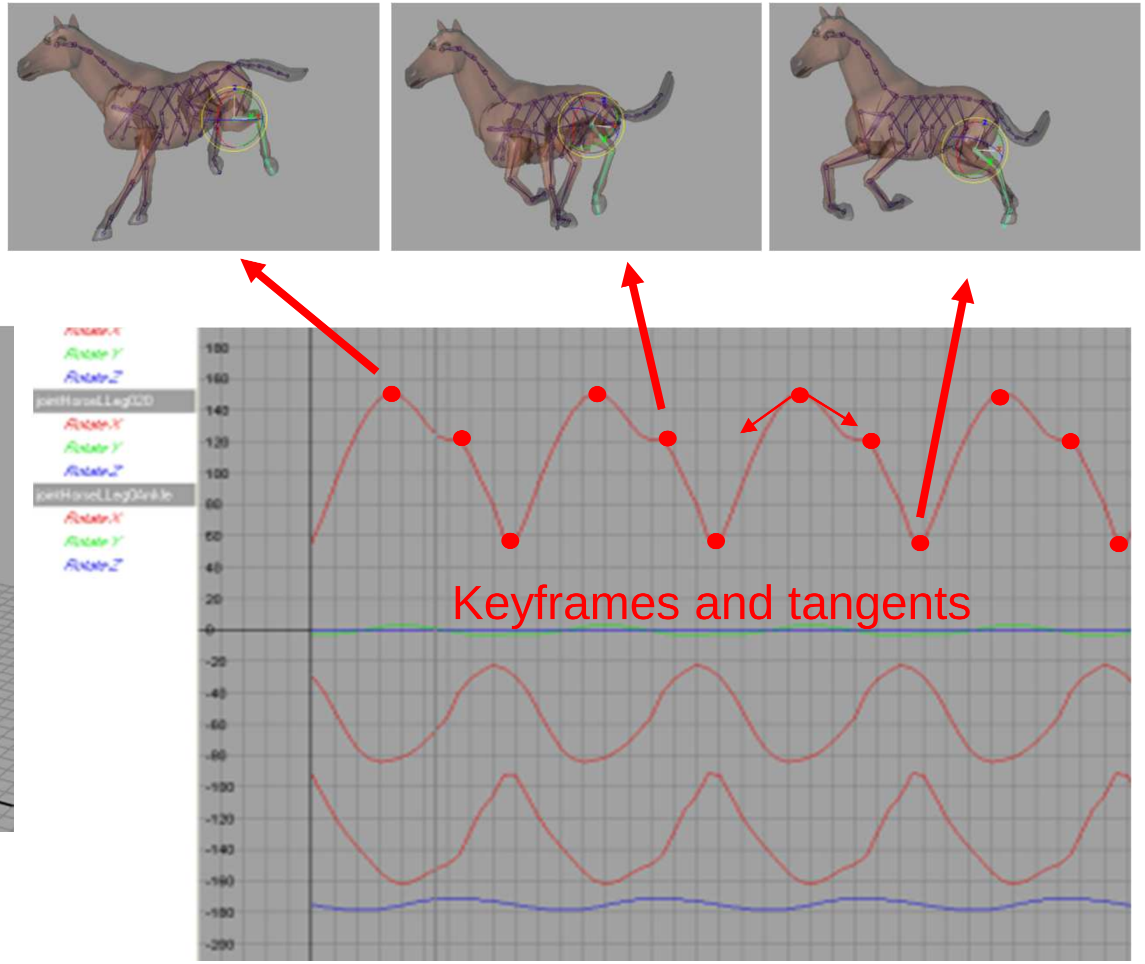
+ Forward kinematics
+ Inverse-Kinematics algorithm
+ Motion planning

Animation skeleton :

appropriate degrees of freedom for expressivity

3D animation : interpolation+skeleton





Interpolation

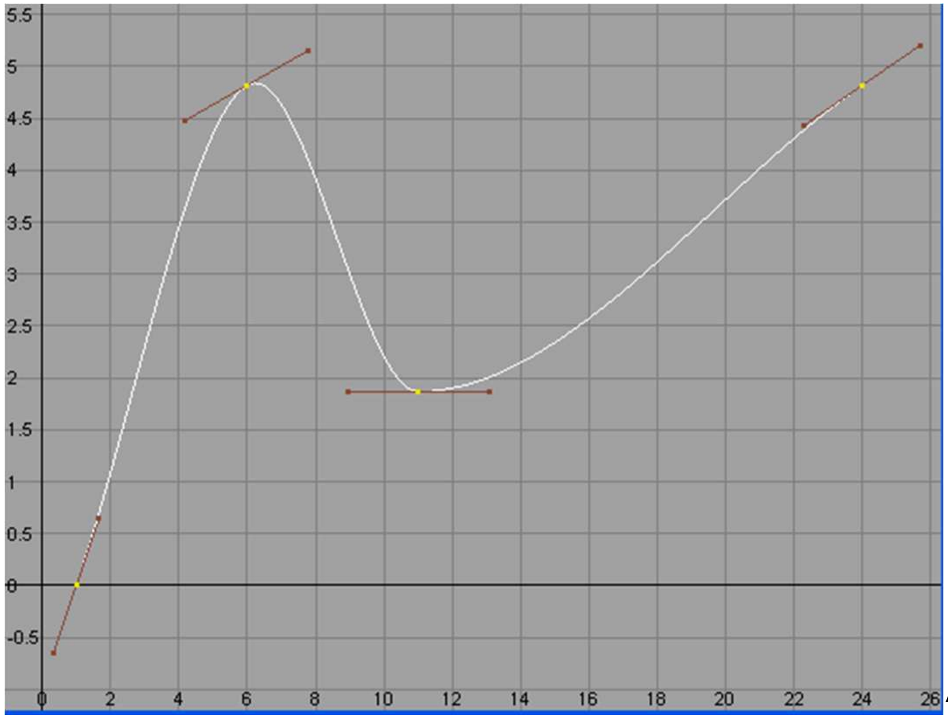
- Interpolation of a 1D-scalar value
  - One function p(u) with unknown parameters,
    - Typically cubic polynomial: $p(u)=\Sigma_{n=0..3}\ a_n\ u^n$

  - known constrains on points
    - $\{\ u_i\ ,\ (d^n p/du)(u_i\ )= b_i\ \}$
    - 4 are enough for exact cubic polynomial

  - Direct solving for a_n
    - p(u) is thus known for every u

Interpolation

- Practical case : Hermit polynomial (spline)
  - C^1 continuity between sets of 2 points
    - 2 positions and tangents at these positions



$$p(u) = [\ a_0\ a_1\ a_2\ a_3\ ]\ [\ 1\ u\ u^2\ u^3\ ]^t = A^t\ Q(u)$$

$$\begin{aligned} p(0) &= A^t\ Q(0) = b_0 \\ p'(0) &= A^t\ Q'(0) = b_1 \\ p(1) &= A^t\ Q(1) = b_2 \\ p'(1) &= A^t\ Q'(1) = b_3 \end{aligned}$$

$$A^t\ [\ Q(0)\ Q'(0)\ Q(1)\ Q'(1)\ ] = [\ b_0\ b_1\ b_2\ b_3\ ]^t$$

$$A^t\ Q_{4x4} = B^t \quad \Rightarrow \quad A^t = B^t\ Q^{-1}$$

$$\boxed{\begin{array}{l} p(u) = B^t\ Q^{-1}\ Q(u) \\ \\ p(\ u=(t-t0)/(t-t1)\ ) \end{array}}$$

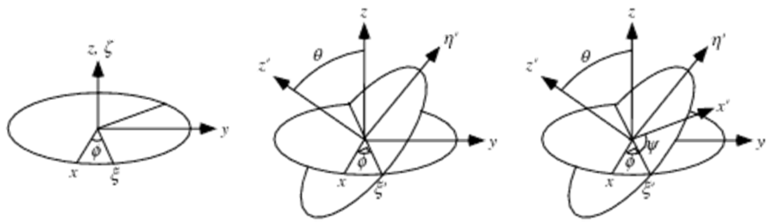
Interpolation

- Interpolating 3D rotation
  - Canonic representation : SO(3) matrix
    - but $M_0, M_1 \in SO(3) \not\Rightarrow (1-\alpha)M_0 + \alpha M_1 \notin SO(3)$
  - Represent SO(3) matrix with Euler angles
    - any rotation in R^3 can be represented by 3 angles
    - $M = R_{x,\psi}\ R_{y,\theta}\ R_{z,\Phi} =$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplication order matters !

    - Local transformation
      - Rzyx : yaw, pitch and roll
      - can be $R_{z,\psi}\ R_{y,\theta}\ R_{z,\varphi}$



=> Animation by interpolating angles

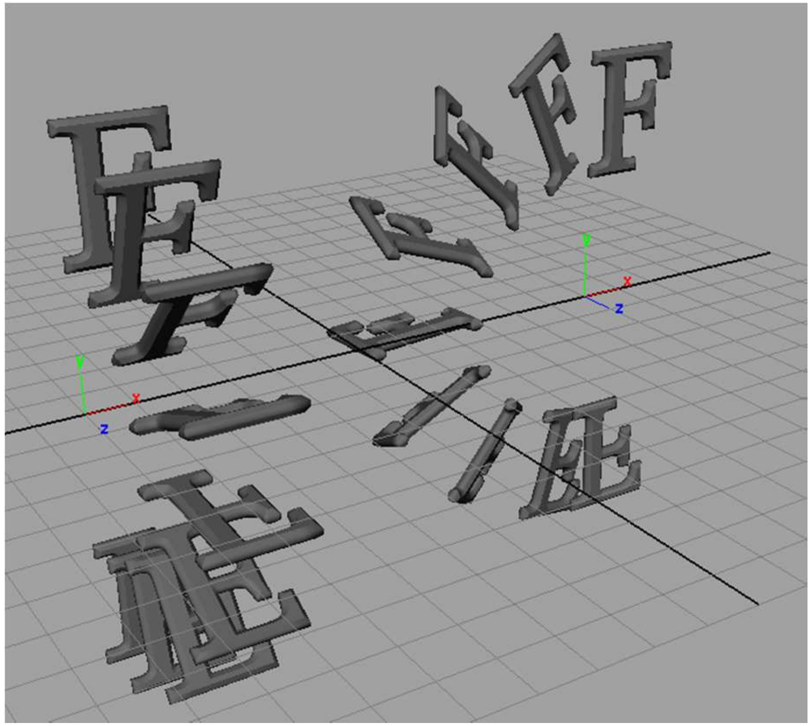
Interpolating 3D rotation

- Limitations of Euler angles
  - Non-uniqueness of position and path

$[\theta_x,\theta_y,\theta_z]=[0,0,0]$

$[\theta_x,\theta_y,\theta_z]=[\pi,0,0]$

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$



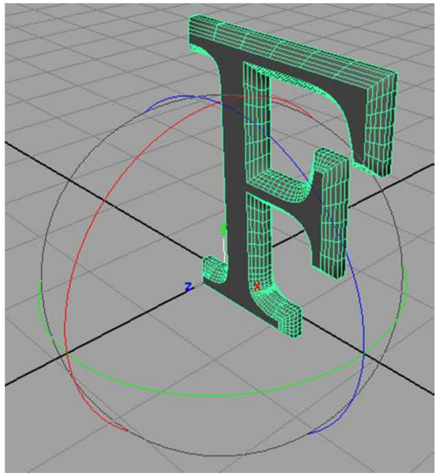
$[\theta_x,\theta_y,\theta_z]=[0,0,0]$

$[\theta_x,\theta_y,\theta_z]=[0,\pi,\pi]$

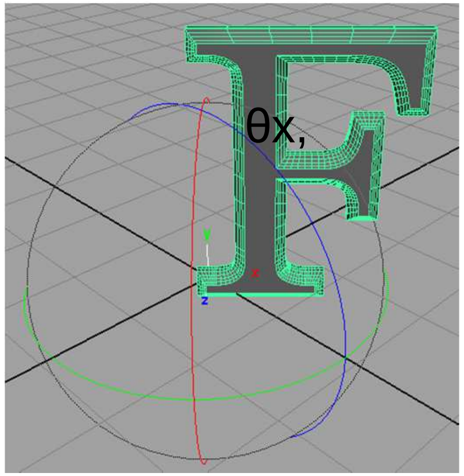
$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Interpolating 3D rotation

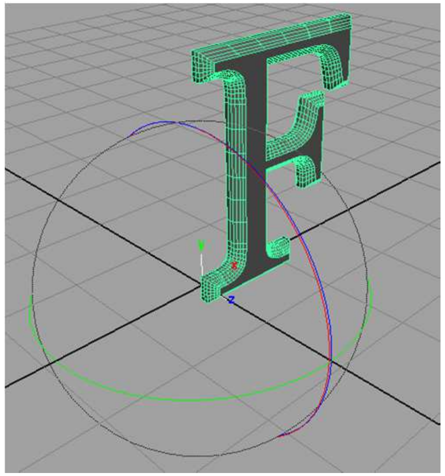
- Limitations of Euler angles
  - Gimbal lock



$[\theta_x,\theta_y,\theta_z]=[0,0,0]$



$[\theta_x,\theta_y,\theta_z]=[0,\pi/4,0]$

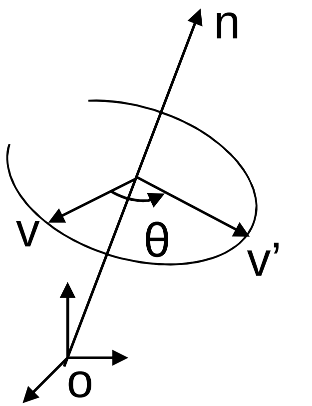


$[\theta_x,\theta_y,\theta_z]=[0,\pi/2,0]$

1 degree of freedom is lost : change in $\theta_x \Leftrightarrow$ change in θ_z

3D rotation

- Axis-angle
  - any rotation in R^3 is a planar rotation around an axis
  - strong link with quaternion



$$v' = R_{\theta,n} v = \cos\theta\, v + \sin\theta\, n \times v + (1-\cos\theta)(v \cdot n)\, n$$

$$R_{\theta,n} = \cos\theta\, \mathbf{I} + \sin\theta\, [n]_\times + (1-\cos\theta) n n^t$$

Rodrigues formula

$$[n]_\times = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

$$R_{\theta,n} = \exp([\theta n]_x) = I + \Sigma_{k=1\ldots\infty} [\theta n]_x^k / k!$$

Quaternion

H: Extension of standard complex

$q = [\ s, x, y, z\ ] = s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z = [\ s, \mathbf{n}\ ]$

with $\mathbf{i}^2=\mathbf{j}^2=\mathbf{k}^2=\mathbf{ijk}=-1$

$q.q' = [\ ss' - \mathbf{n.n}', s'\ \mathbf{n} + s\ \mathbf{n'} + \mathbf{n} \times \mathbf{n'}\ ]$

$q^* = [s, -x, -y, -z\ ]$

$||q||^2 = q\ .\ q^* = s^2 + x^2 + y^2 + z^2$

$q^{-1} = q^*/||q||^2$

$||q||=1 \Rightarrow q = [\ \cos\theta, (\sin\theta)\mathbf{n}\ ] \in H_1$

with $\mathbf{n}\in R^3$ and $||\mathbf{n}||=1$

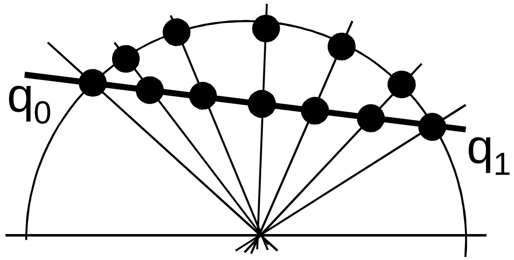
Any rotation in R^3 can be represented in H_1

$x\in R^3, x'= R_{\theta,n}x \Leftrightarrow [0,x'] = q[0,x]q^{-1}$

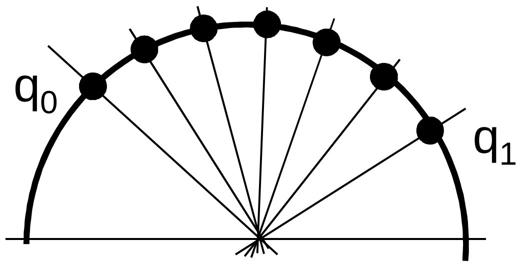
with $q = [\ \cos(\theta/2), \sin(\theta/2)\mathbf{n}\ ]$

Interpolating 3D rotation

- Quaternion
  - Linear interpolation in H does not work well
    - $q(t) = (1-t).q_0 + t.q_1$
    - Angular velocity is not constant



  - Spherical linear interpolation is fine (SLERP)



Interpolating 3D rotation

Log and exp in H_1

$q = [\cos\theta, (\sin\theta)\mathbf{n}] = \exp([0,\theta\mathbf{n}])$

$\log(q) = [0,\theta\mathbf{n}]$

$q^t = \exp(t.\log(q))$

$dq^t/dt = \log q \; q^t$

$\| dq^t/dt \| = \|\log q\| = \| [0,\theta\mathbf{n}] \| = |\theta|$

Application to SLERP

(q_i : unit quaternion, p_i : geometrical point on sphere)

$\mathrm{SLERP}(q0,q1,t) = (q_1 q_0^{-1})^t q_0$

$\mathrm{SLERP}(p0,p1,t) = (\sin[(1-t)\Omega]p_0 + \sin[\Omega t]p_1)/\sin\Omega$

with $\cos\Omega = p_0 \cdot p_1$

Kinematic Animation

- Two fundamental cases
  - Unconstrained motion
    - waving arms, nodding head, etc

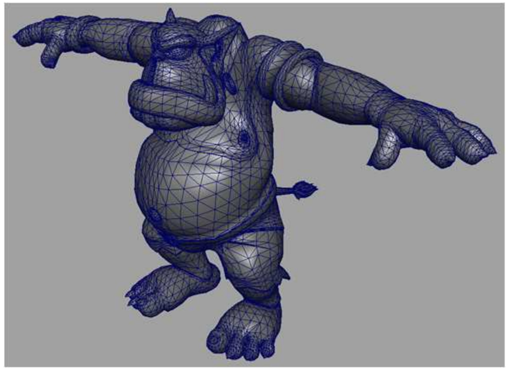
    => Forward Kinematics (FK)
  - Constrained motion
    - grasping an object, walking on the ground, etc

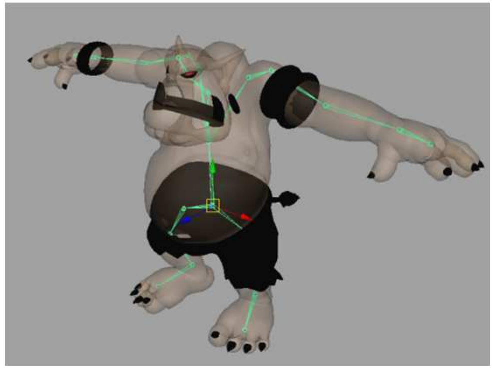
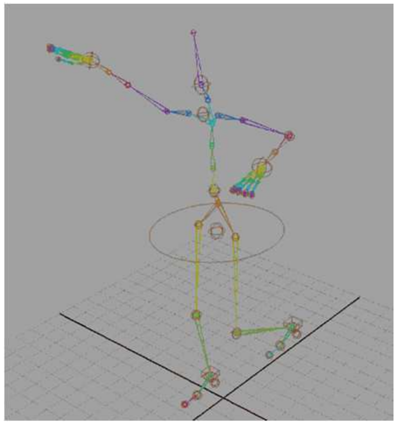
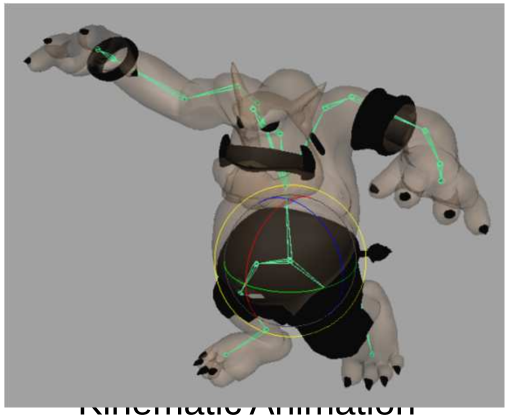
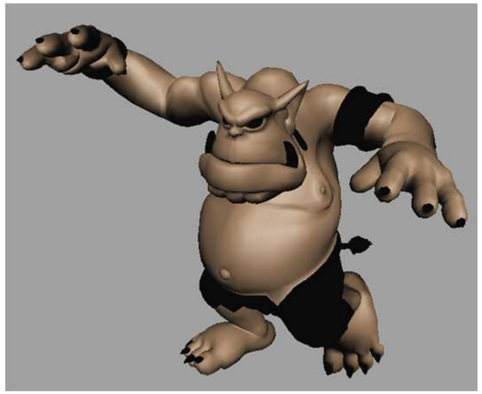
    => Inverse Kinematics (IK)

lionel.reveret@inria.fr

Forward Kinematics

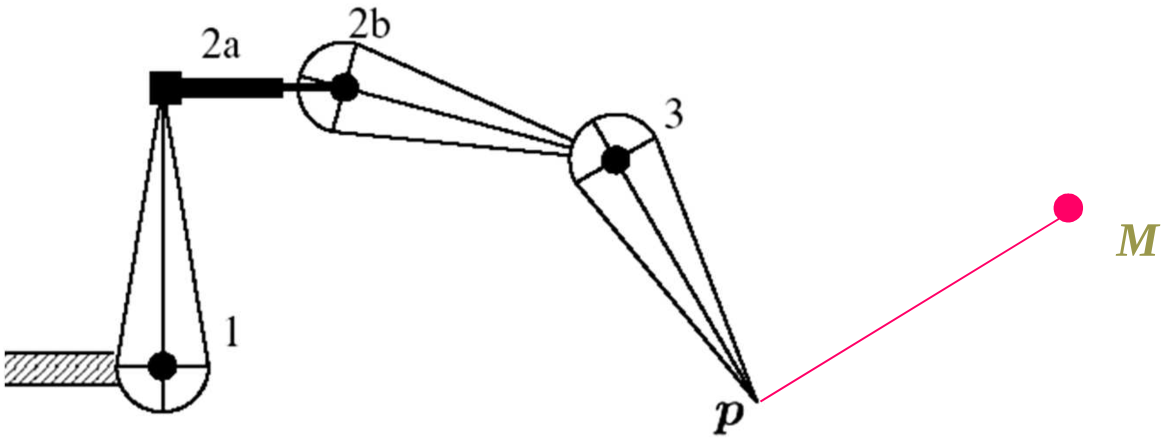
- Direct application of 3D framework



Inverse Kinematics

- Articulated object
  - Translational and rotational links
  - Goal to reach



Inverse Kinematics

- input : goal to reach (M)
- model parameter :

  $\Theta = (\theta_1, \theta_2, \ldots, t_1, t_2, \ldots)$, model parameters

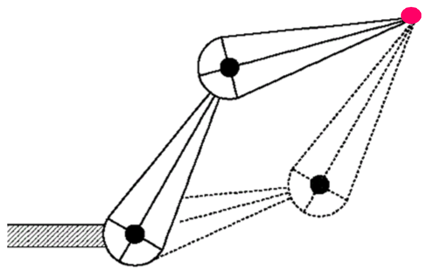
  $f(\Theta)$ position of kinematic chain end

$\Rightarrow$ Find $\Theta^* / M = f(\Theta^*)$

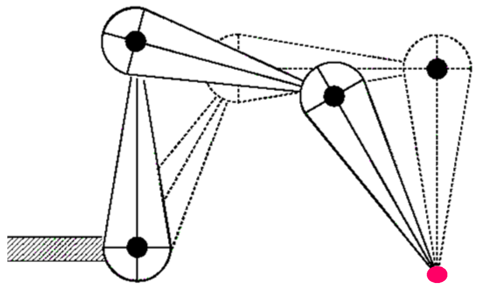
- 2 or 3 rotations: direct computation in R^2 or R^3
- N articulations : ?

Difficulties

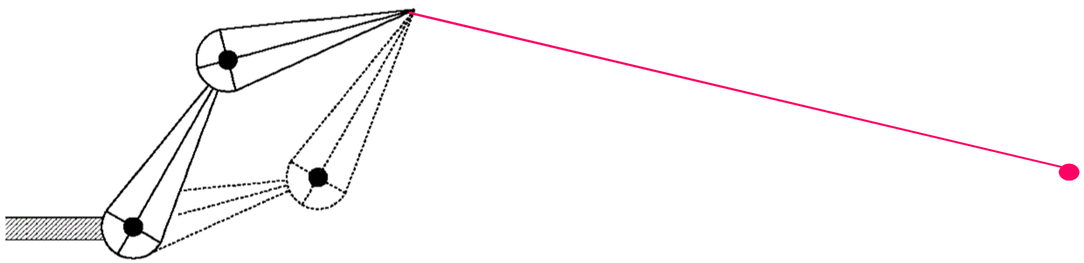
- Two solutions :



- Range of solutions :



- No solutions :



f : direct kinematics

- Concatenation of matrix transforms

- $f\,(\Theta) = \mathbf{R}_1(\theta_1)\mathbf{T}_1\boldsymbol{(}t_1\boldsymbol{)}\mathbf{R}_2\,(\theta_2)\,\mathbf{T}_2\boldsymbol{(}t_2\boldsymbol{)}\ldots M_0$
  - M_0 : position in rest pose (no rotations)

- Non linearity because of rotations

Zero of non-linear function

- Find Θ / $f(\Theta)$-M = 0

- Linearization :
  - given a current Θ and error E = $f(\Theta)$-M
  - find h / E = $f(\Theta+h) - f(\Theta) = f'(\Theta)h$

  => $h = f'(\Theta)^{-1} E$

  => $\Theta := \Theta + h$

  - iterate

Linearization

- Taylor series :

$$f(\Theta + h) = f(\Theta) + f'(\Theta)h + f''(\Theta)h^2 + \ldots$$

- Multivariate case :

$$f(\Theta + h) = f(\Theta) + \mathbf{J}(\Theta)h + {}^t h\mathbf{H}(\Theta)h + \ldots$$

- **J** Jacobian of *f*, linear form in h
- **H** Hessian of *f*, quadratic form in h

Jacobian

- Matrix of derivatives of several functions with severable variables :

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

J:3xN matrix => not squared

Use Pseudo-inverse for inversion :

$$J^{+} = J^{t}(JJ^{t})^{-1} \quad \text{if } N>3$$

or $$J^{+} = (J^{t}J)^{-1}J^{t} \quad \text{if } N<3$$

Algorithm

```
inverseKinematics()
  {
    start with current Θ;
    E := target - computeEndPoint();
    for(k=0; k<k_max && |E| > eps; k++){
      J := computeJacobian();
      solve J h = E;
      Θ := Θ + h;
      E := target - computeEndPoint();
    }
  }
```

Joint limits

- Joint may have limits of variation
  - For example realistic elbow is limited
- To enforce limits :
  - test for limitation violation
  - cancel parameter if violation
    - in practice, remove column in J
  - compute new J and J^+
  - compute new h

Adding constrains

1. if Ker$J \neq 0$, degrees of freedom left to enforce a new constrains Ω

   $\mathbf{J}\Omega = 0$

2. if Θ solves $\mathbf{J}\Theta = E$, thus $\Theta+\Omega$ is also solution

   $\mathbf{J}(\Theta+\Omega) = \mathbf{J}\Theta + \mathbf{J}\Omega = E + 0 = E$

3. if general constrain C, need to project on KerJ:

   $C_p = (J^+J - I)\, C$

   check: $J\, C_p = J\,(J^+J - I)\, C = (J - J)\, C = 0$

lionel.reveret@inria.fr

Example: preferred angle

- Value : Θ_{pref}
- Constraint C :
  – $C_i = \Theta_i - \Theta_{\text{pref}}$

- Modified algorithm :
  – use $\mathbf{h} = \mathbf{J}^{+}E + (\mathbf{J}^{+}\mathbf{J} - \mathbf{I})C$
  => preserve convergence

Inverse Kinematics

- Other methods
  - use J^t instead of J^+
    - theory of infinitesimal works
    - $h = J^t E$
  - use several 1D optimization
    - Cyclic Coordinate Descent

=> faster but less accurate

Inverse Kinematics

- A good source:

  http://billbaxter.com/courses/290/html/