Animation & Physically-Based Simulation

0368-3236, Spring 2019

Tel-Aviv University

Amit Bermano

Computer Animation

- Describing how 3D objects (& cameras) move over time

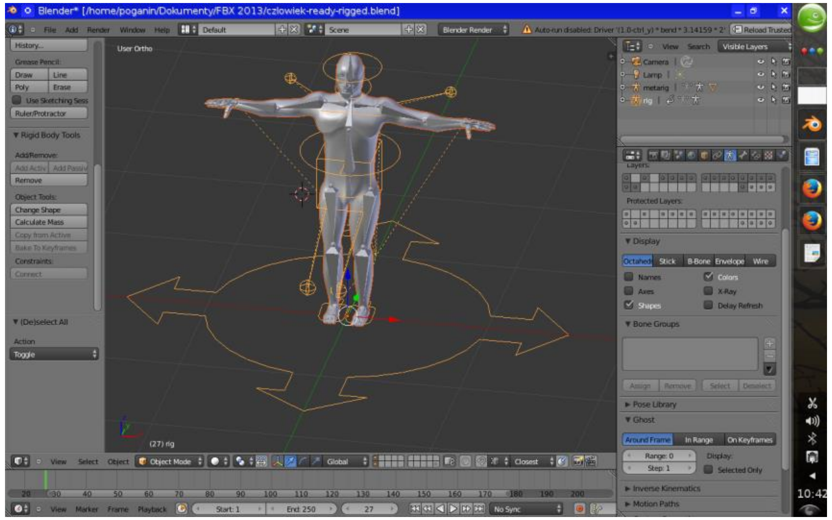
Pixar

Computer Animation

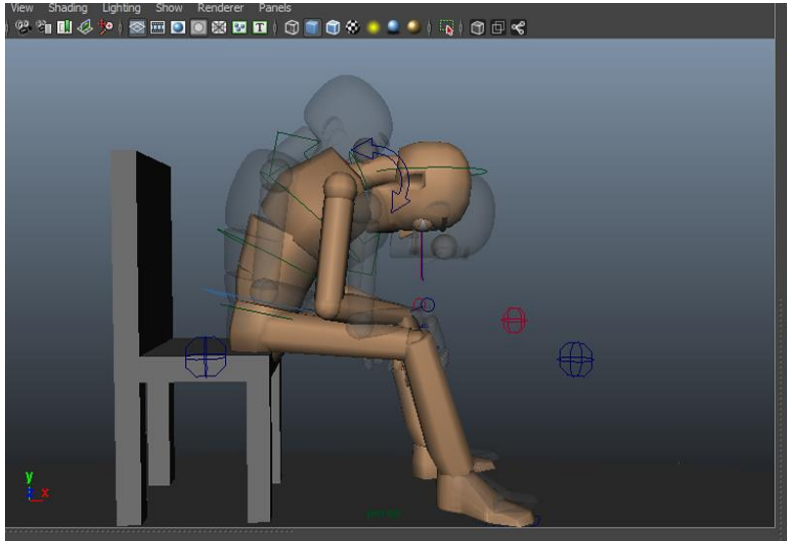
- Challenge is balancing between …
  - Animator control
  - Physical realism

Computer Animation

- Manipulation
  - Posing
  - Configuration control



https://blenderartists.org/

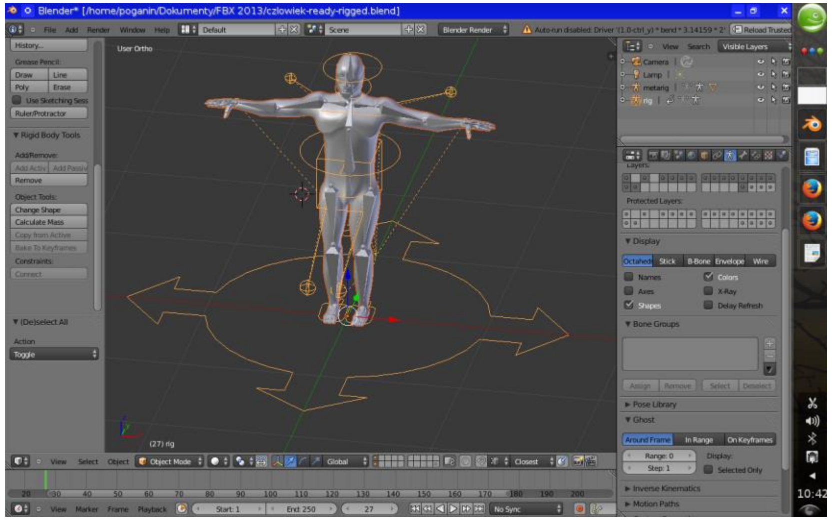


focus.gscept.com

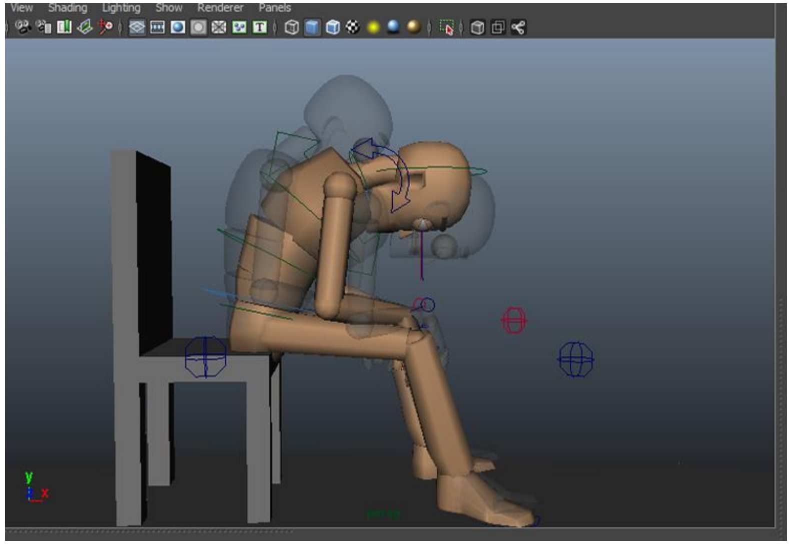
- Interpolation
  - Animation
  - In-betweening

Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation
  - Procedural



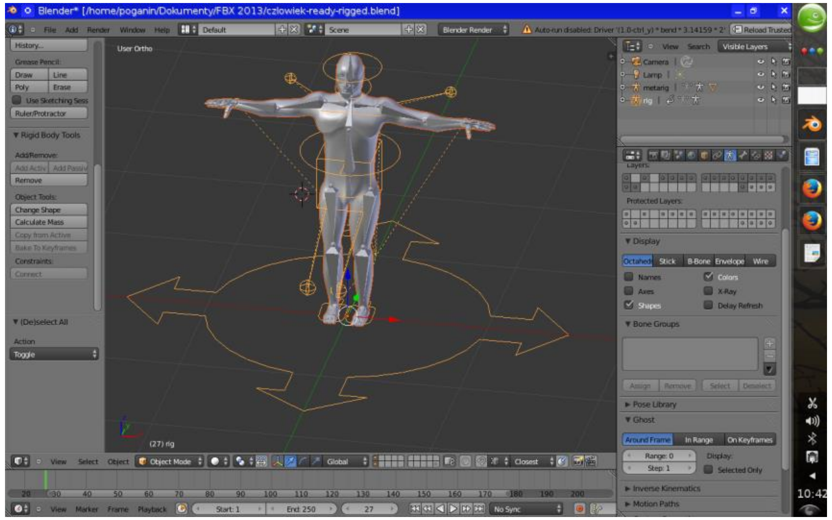
https://blenderartists.org/



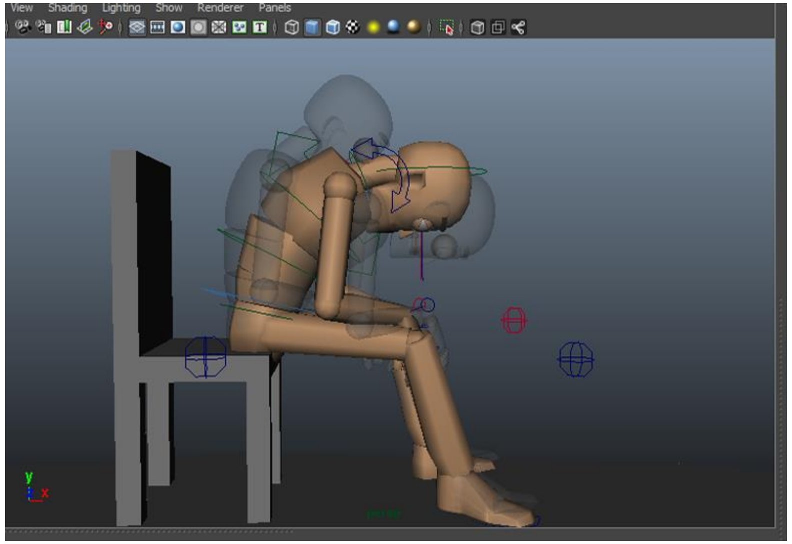
focus.gscept.com

Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation
  - Procedural



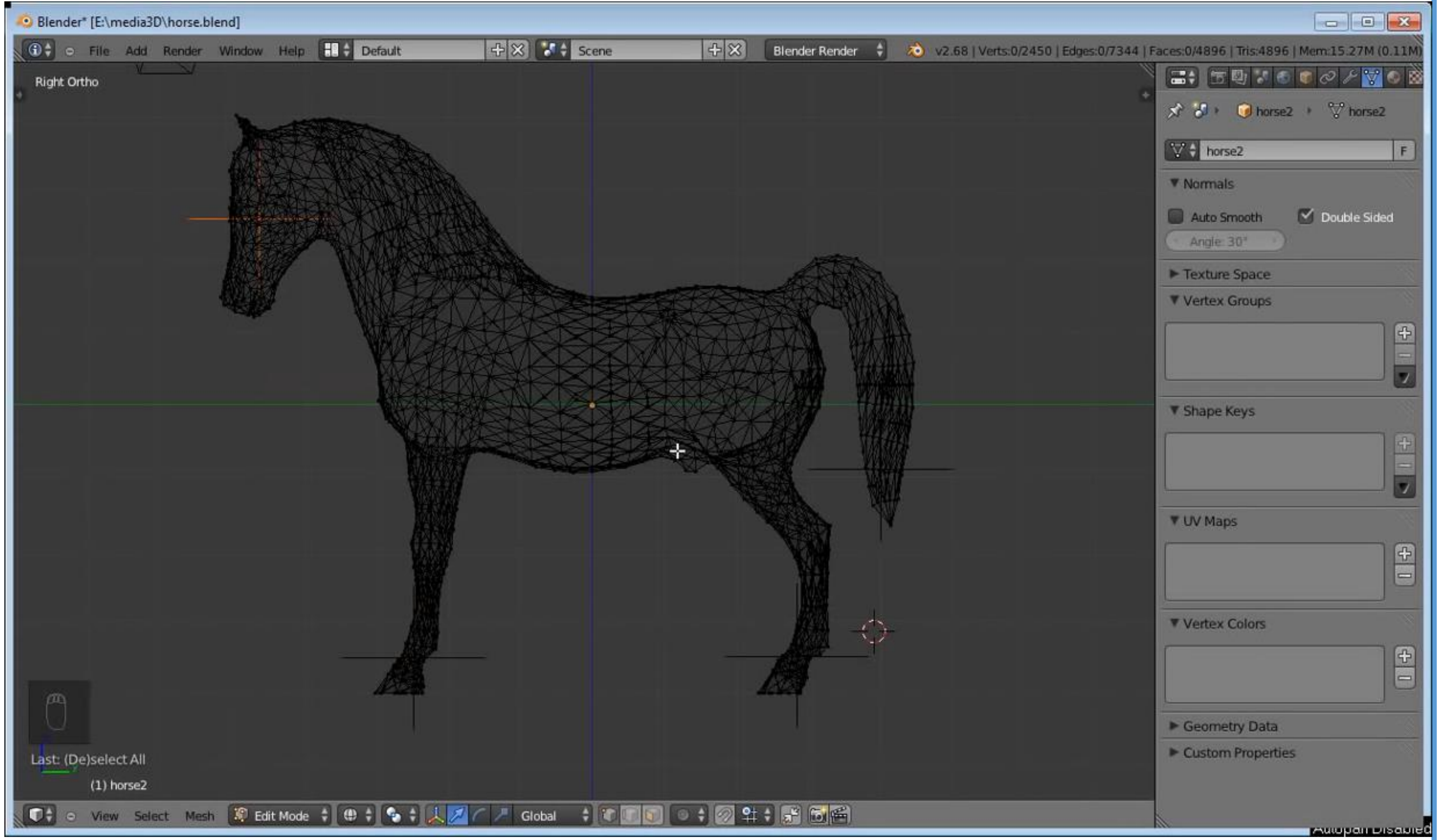
https://blenderartists.org/



focus.gscept.com

Deformation

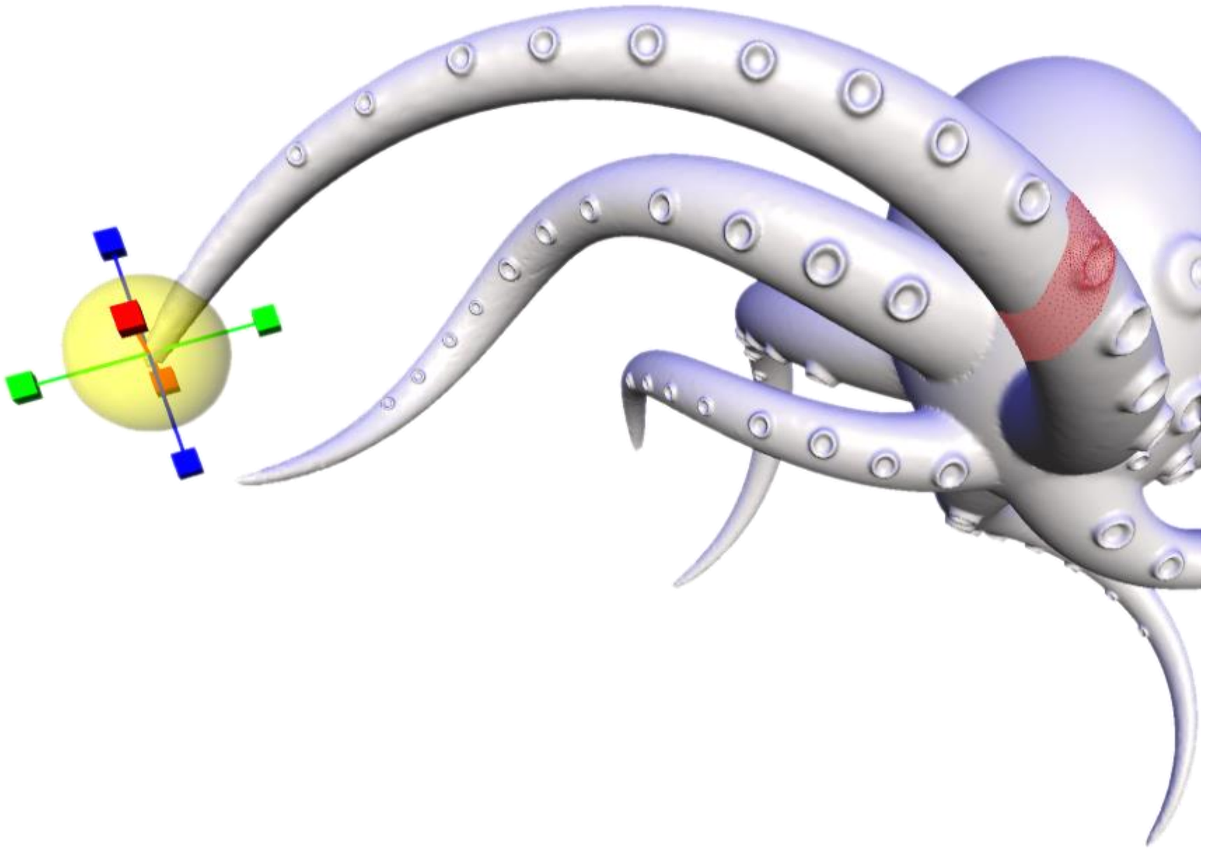
- How to change a character's pose?
  - Every vertex directly
  - Intuitive computation



https://www.youtube.com/watch?v=oxkf_N-QCNI

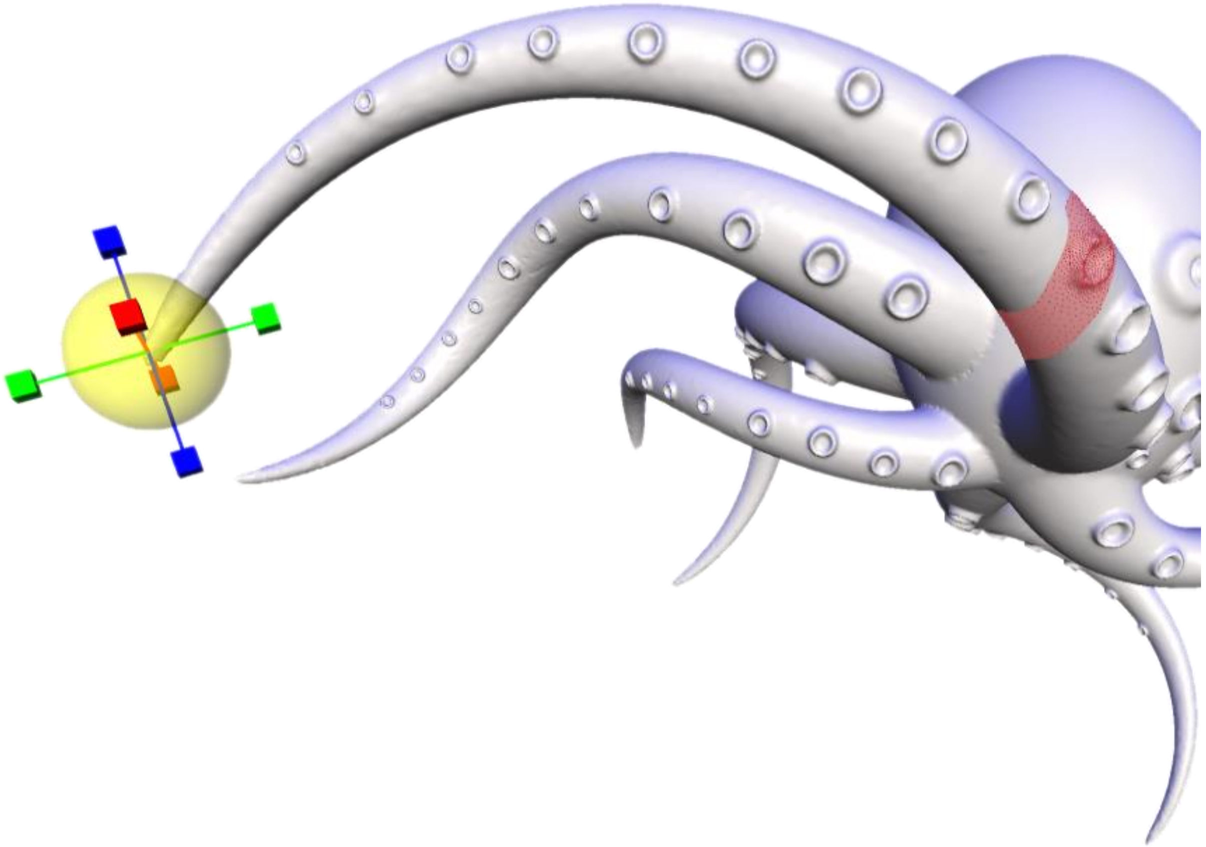
Deformation

- A HUGE variety of methods
  - Laplacian mesh editing
  - ARAP
  - CAGE Base
  - Barycentric coordinates
  - Heat diffusion
  - Variational
  - …



Deformation

- A HUGE variety of methods
  - Laplacian mesh editing
  - ARAP
  - CAGE Base
  - Barycentric coordinates
  - Heat diffusion
  - Variational
  - …

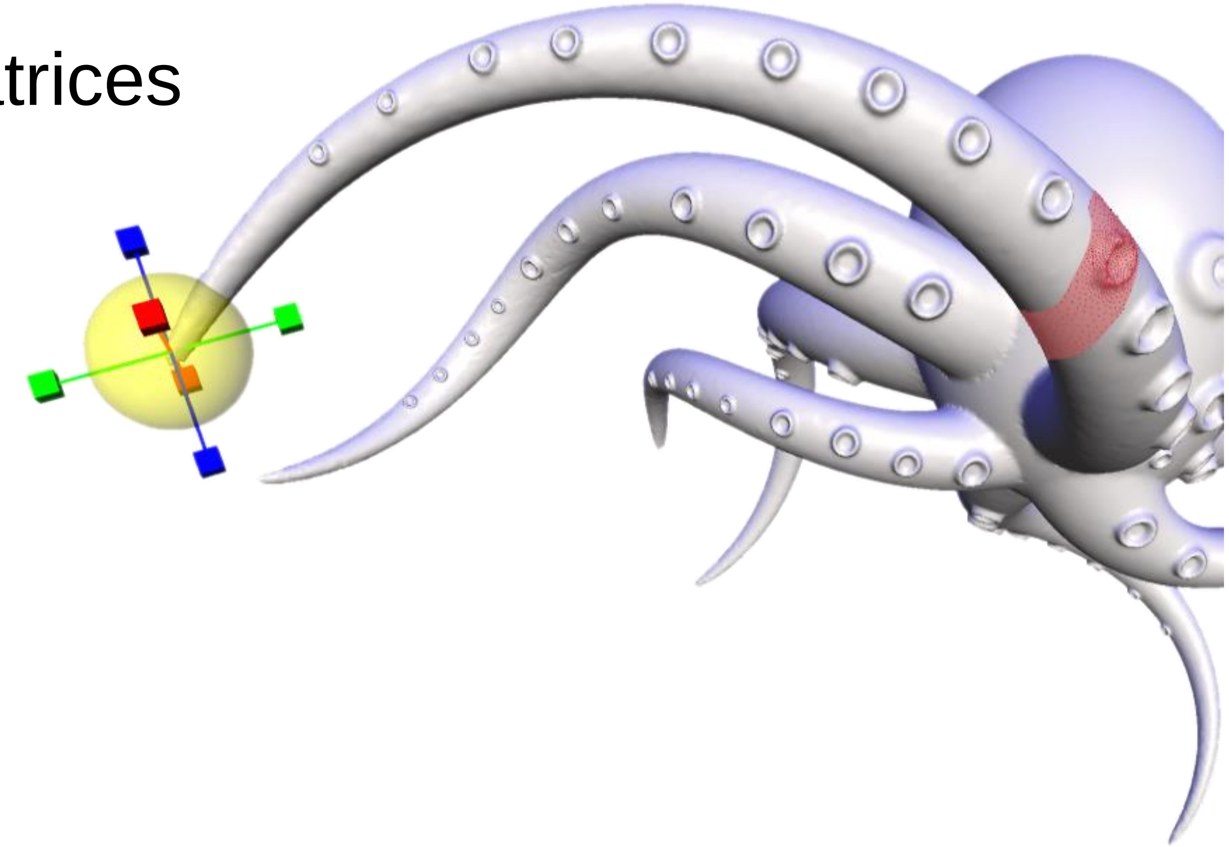


Laplacian Mesh Editing

- Local detail representation – enables detail preservation through various modeling tasks

- Representation with sparse matrices

- Efficient linear surface reconstruction



Overall framework

1. Compute differential representation

$$\delta_i = L(v_i) = v_i - \frac{1}{d_i} \Sigma_{j \in \mathrm{N}(i)} v_j$$

2. Pose modeling constraints

$$v_i' = u_i, \qquad i \in \boldsymbol{C}$$

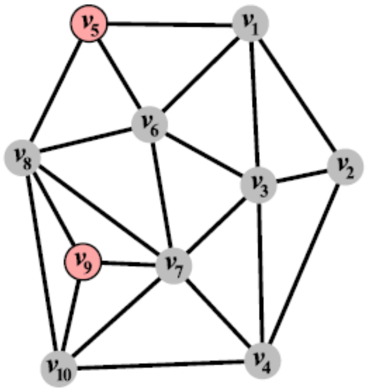
3. Reconstruct the surface – in least-squares sense

$$\binom{L}{L_c} \boldsymbol{V} = \binom{\boldsymbol{\delta}}{\boldsymbol{U}}$$

Differential coordinates?

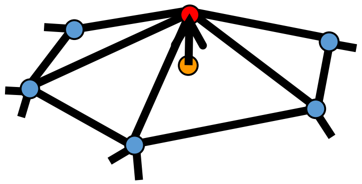
- In matricial form:

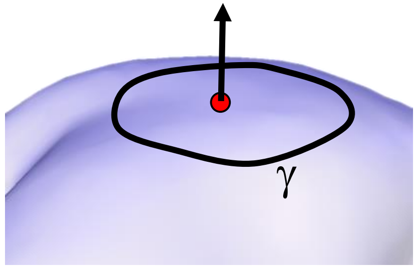
$$L_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \Sigma_{j \in 1_{ring_i}} w_{ij} & i = j \\ 0 & else \end{cases}$$



$$\begin{bmatrix} 4 & -1 & -1 & & -1 & -1 & & & & \\ -1 & 3 & -1 & -1 & & & & & & \\ -1 & -1 & 5 & -1 & & -1 & -1 & & & \\ & -1 & -1 & 4 & & & -1 & & & -1 \\ -1 & & & & 3 & -1 & & -1 & & \\ -1 & & -1 & & & 4 & -1 & -1 & & \\ & & -1 & -1 & & -1 & 6 & -1 & -1 & -1 \\ & & & & -1 & -1 & -1 & 6 & -1 & -1 \\ & & & & & & -1 & -1 & 3 & -1 \\ & & & -1 & & & -1 & -1 & -1 & 4 \end{bmatrix}$$

- They represent the **local** detail / local shape description
  - The direction approximates the normal
  - The size approximates the mean curvature





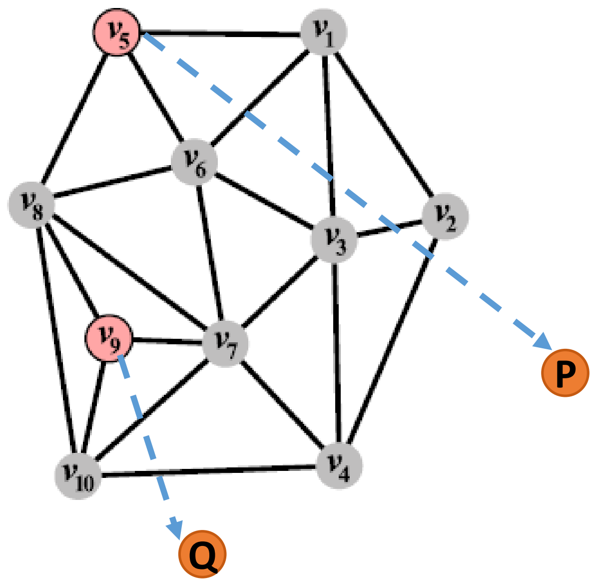
Adding constraints

- In matricial form:

$$L_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \sum_{j \in 1_{ring_i}} w_{ij} & i = j \\ 0 & else \end{cases}$$

$$\begin{bmatrix} 4 & -1 & -1 & & -1 & -1 & & & & \\ -1 & 3 & -1 & -1 & & & & & & \\ -1 & -1 & 5 & -1 & & -1 & -1 & & & \\ & -1 & -1 & 4 & & & -1 & & & -1 \\ -1 & & & & 3 & -1 & & -1 & & \\ -1 & & -1 & & & 4 & -1 & -1 & & \\ & & -1 & -1 & & -1 & 6 & -1 & -1 & -1 \\ & & & & -1 & -1 & -1 & 6 & -1 & -1 \\ & & & & & & -1 & -1 & 3 & -1 \\ & & & -1 & & & -1 & -1 & -1 & 4 \end{bmatrix}$$

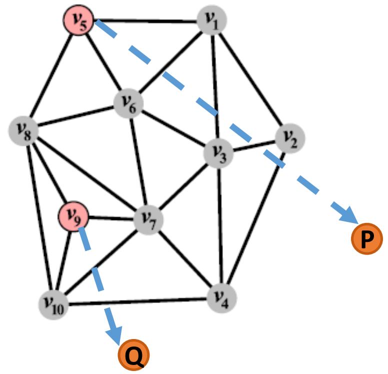
$$\begin{bmatrix} & & & & 1 & & & & & \\ & & & & & & & & 1 & \end{bmatrix}$$



Adding constraints

- In matricial form:

$$L_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \Sigma_{j\in 1_{ring_i}} w_{ij} & i = j \\ 0 & else \end{cases}$$



$$\begin{bmatrix} 4 & -1 & -1 & & -1 & -1 & & & & \\ -1 & 3 & -1 & -1 & & & & & & \\ -1 & -1 & 5 & -1 & & -1 & -1 & & & \\ & -1 & -1 & 4 & & & -1 & & & -1 \\ -1 & & & & 3 & -1 & & -1 & & \\ -1 & & -1 & & & 4 & -1 & -1 & & \\ & & -1 & -1 & & -1 & 6 & -1 & -1 & -1 \\ & & & & -1 & -1 & -1 & 6 & -1 & -1 \\ & & & & & & -1 & -1 & 3 & -1 \\ & & & -1 & & & -1 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} & & & & 1 & & & & & \\ & & & & & & & & 1 & \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} = \begin{pmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \\ \delta_9 \\ \delta_{10} \\ \boldsymbol{P_x} \\ \boldsymbol{Q_x} \end{pmatrix}$$

$$\begin{bmatrix} 4 & -1 & -1 & & -1 & -1 & & & & \\ -1 & 3 & -1 & -1 & & & & & & \\ -1 & -1 & 5 & -1 & & -1 & -1 & & & \\ & -1 & -1 & 4 & & & -1 & & & -1 \\ -1 & & & & 3 & -1 & & -1 & & \\ -1 & & -1 & & & 4 & -1 & -1 & & \\ & & -1 & -1 & & -1 & 6 & -1 & -1 & -1 \\ & & & & -1 & -1 & -1 & 6 & -1 & -1 \\ & & & & & & -1 & -1 & 3 & -1 \\ & & & -1 & & & -1 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} & & & & 1 & & & & & \\ & & & & & & & & 1 & \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \end{pmatrix} = \begin{pmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \\ \delta_9 \\ \delta_{10} \\ \boldsymbol{P_y} \\ \boldsymbol{Q_y} \end{pmatrix}$$

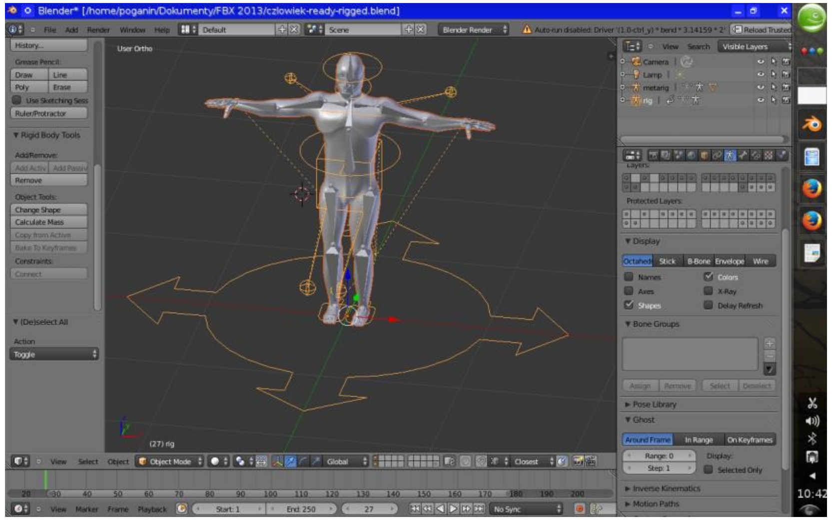
Demo

Laplacian Mesh Editing

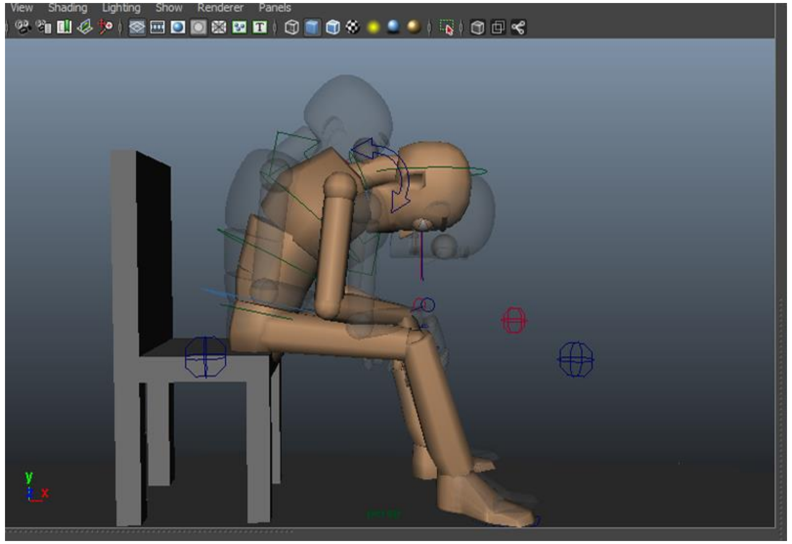
A short editing session
with the *Octopus*

Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation
  - Procedural



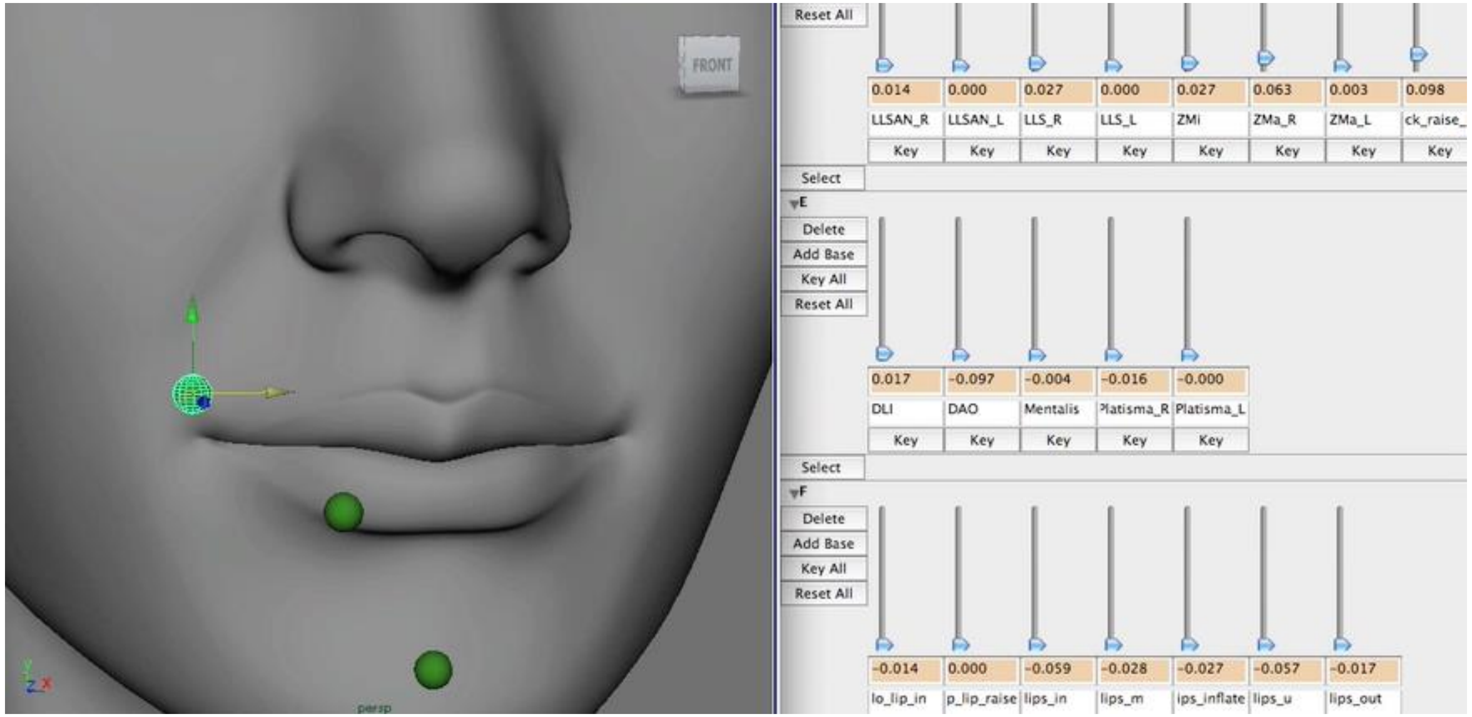
https://blenderartists.org/



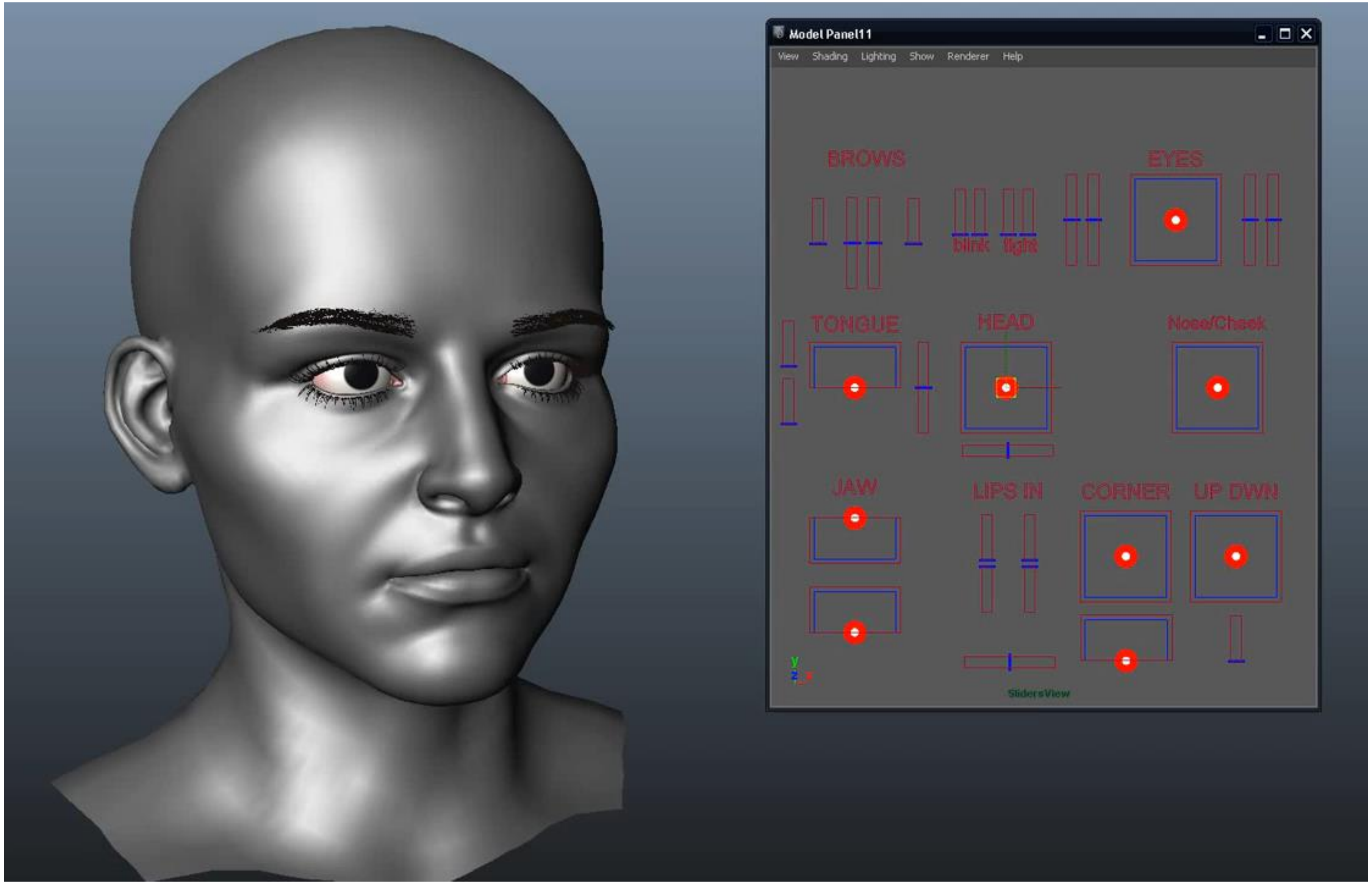
focus.gscept.com

Blendshapes

- Blendshapes are an approximate semantic parameterization
- Linear blend of predefined poses



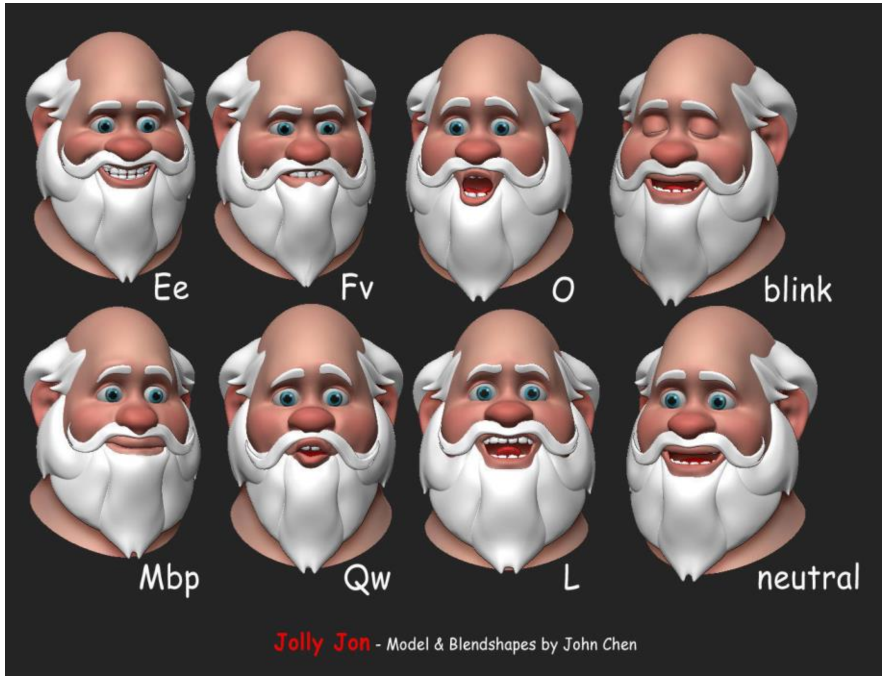
Blendshapes



https://www.youtube.com/watch?v=KPDfMpuK2fQ

Blendshapes

- Usually used for difficult to pose complex deformations
  - Such as faces
- Given:
  - A mesh $M = (V, E)$ with m vertices
  - n configurations of the same mesh, $M_b = (V_b, E), b = 1 \dots n$
- A new configuration is simply:
  - $M' = (\Sigma_{b=1\dots n} \mathrm{w_b V_b, E})$
- Delta formulation:
  - $M' = (\Sigma_{b=1\dots n} V_0 + \mathrm{w_b(V_b - V_0), E})$
  - A bit more convenient
- M_0 - the rest pose, w_b blend weights

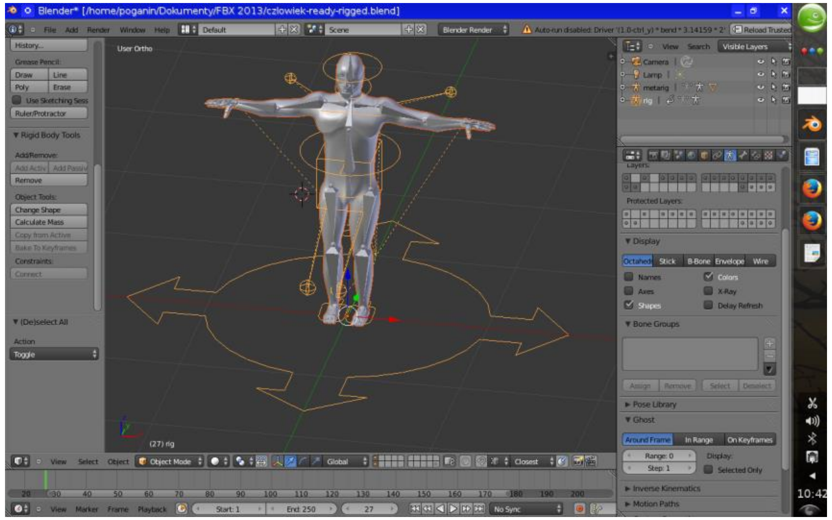


Blendshapes

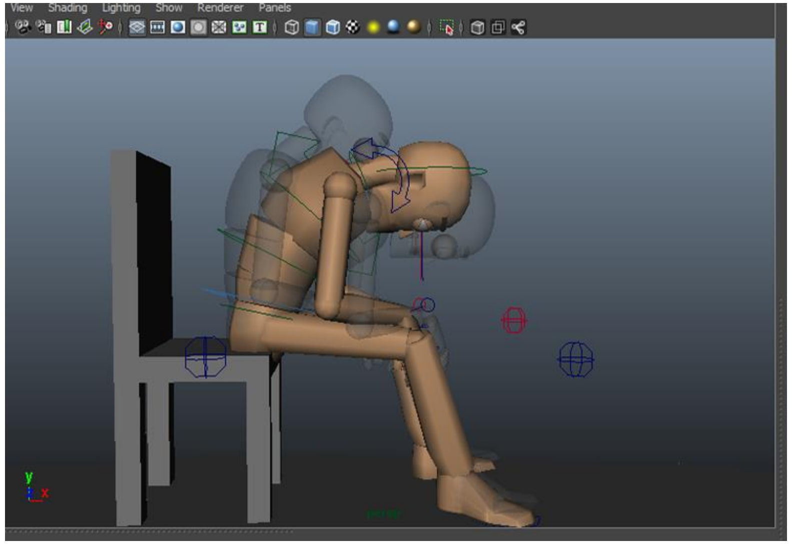
https://www.youtube.com/watch?v=jBOEzXYMugE

Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation
  - Procedural



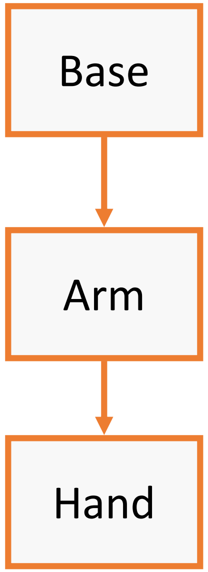
https://blenderartists.org/



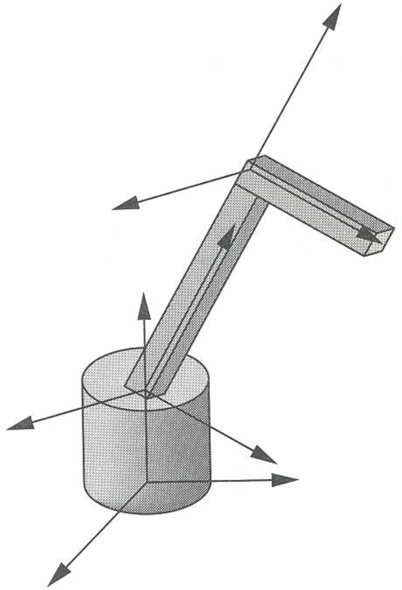
focus.gscept.com

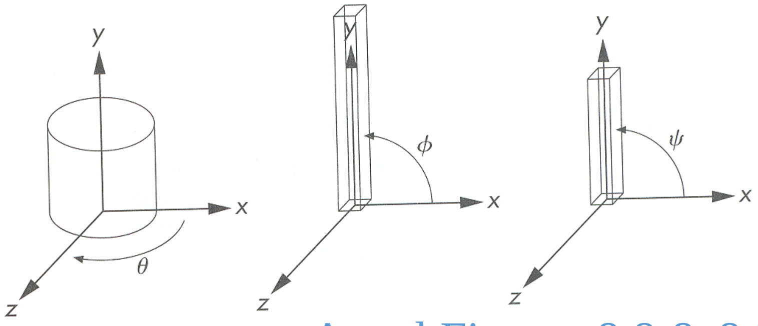
Articulated Figures

- Character poses described by set of rigid bodies connected by “joints”



Scene Graph

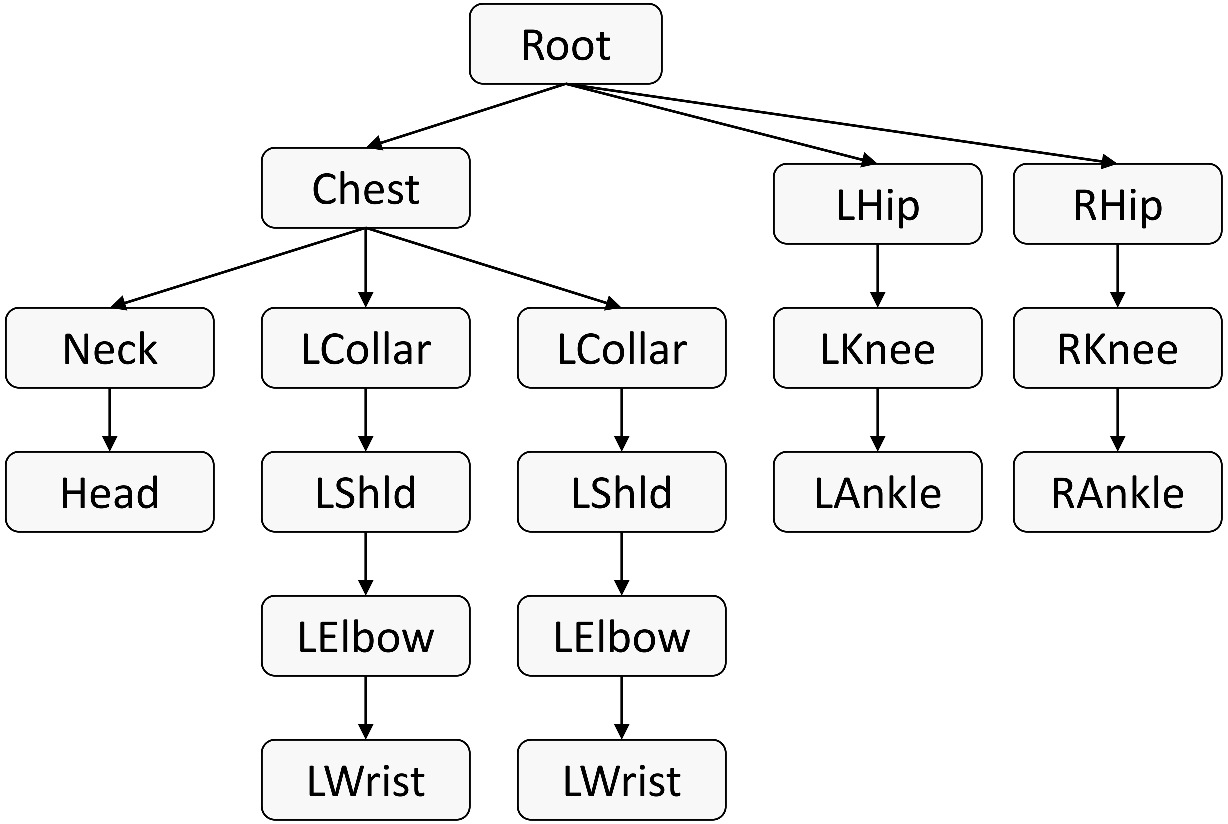


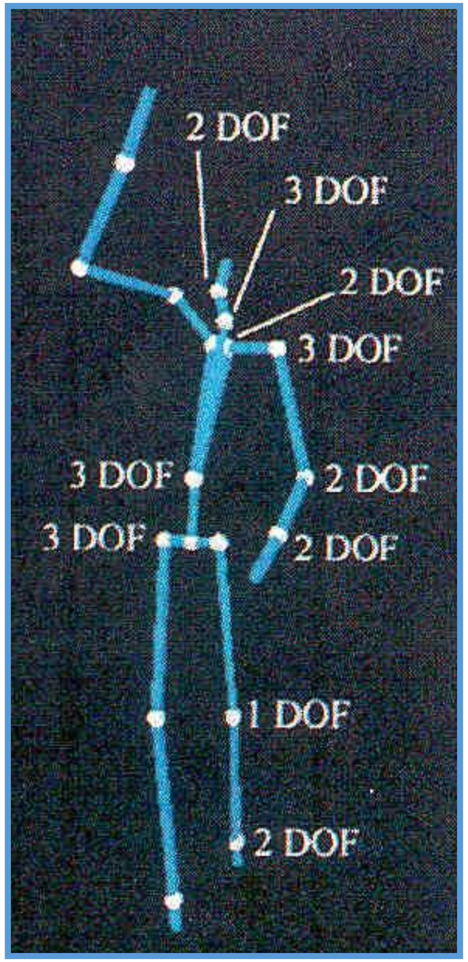


Angel Figures 8.8 & 8.9

Articulated Figures

- Well-suited for humanoid characters

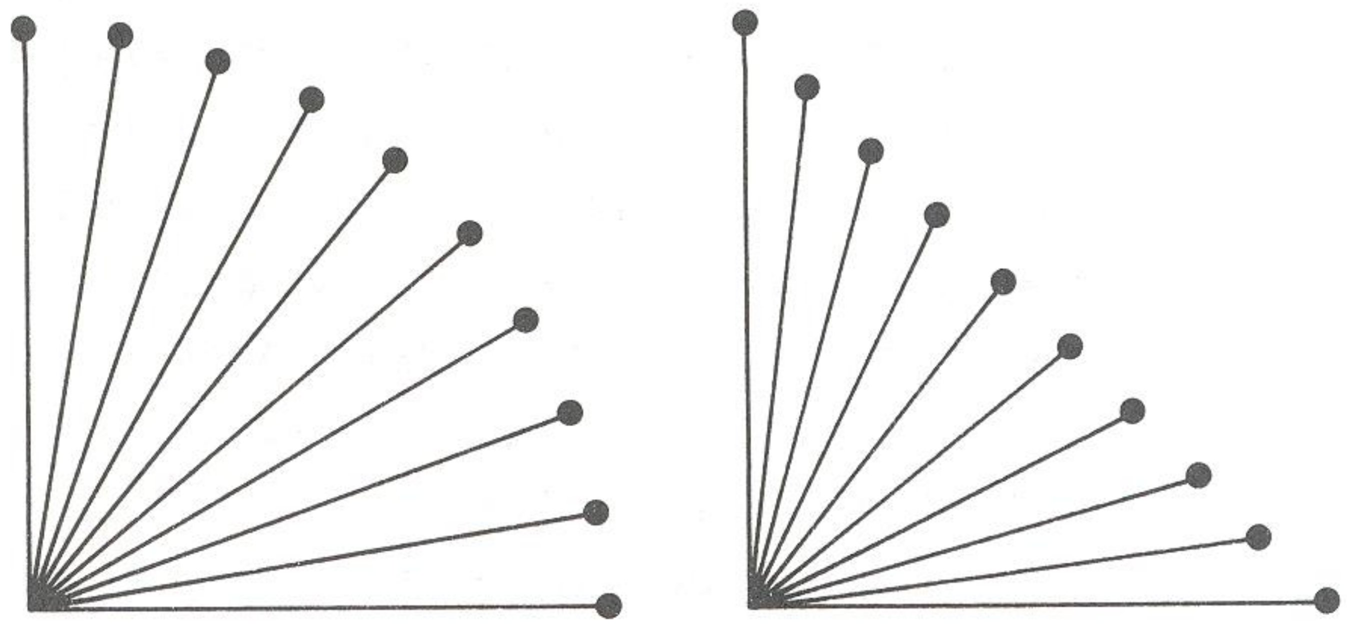




Rose et al. `96

Articulated Figures

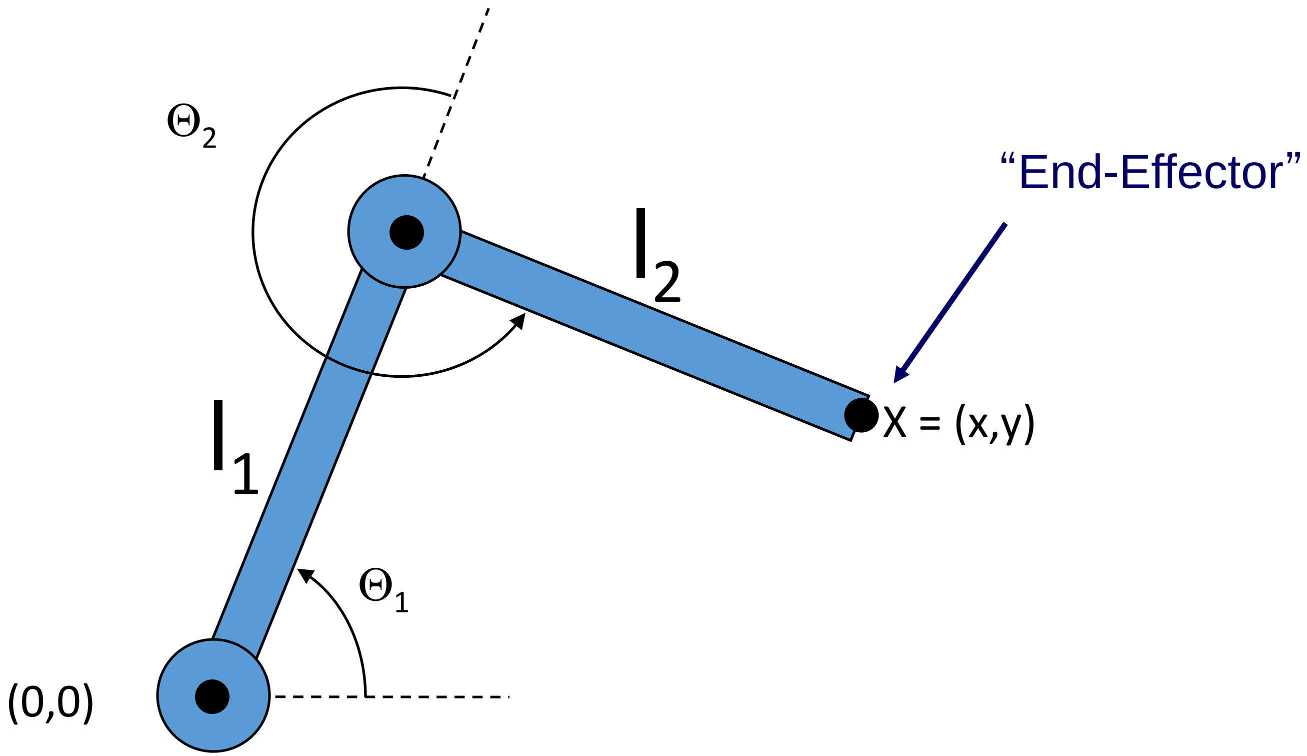
- Animation focuses on joint angles, or general transformations



Watt & Watt

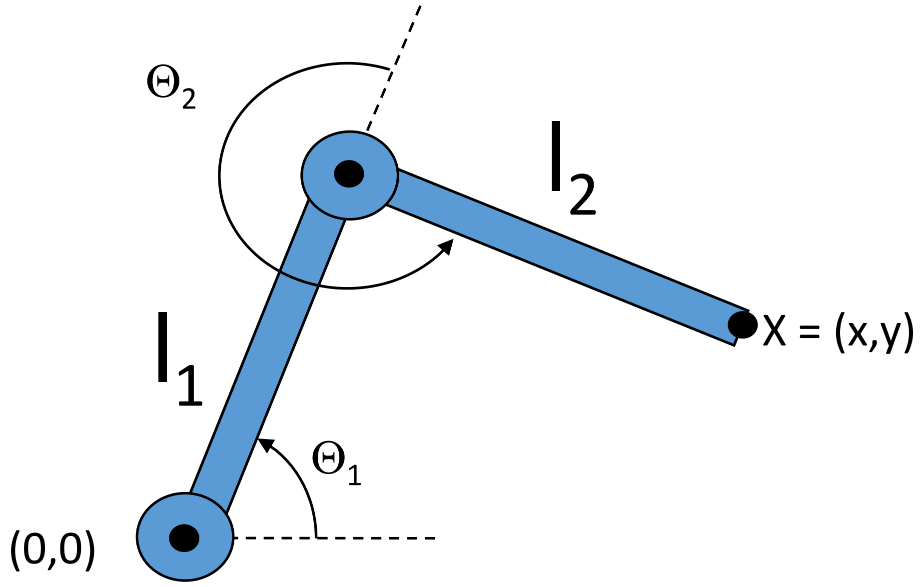
Forward Kinematics

- Describe motion of articulated character



Forward Kinematics

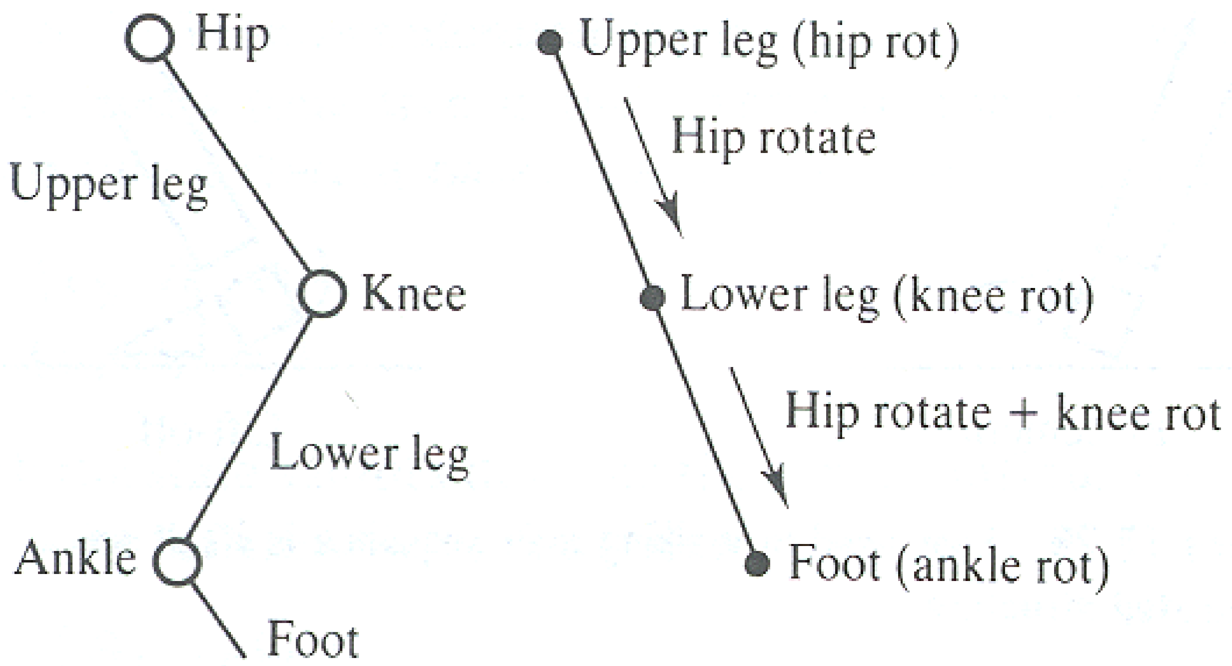
- Animator specifies joint angles: Θ_1 and Θ_2
- Computer finds positions of end-effector: X



$$X = (l_1 \cos\Theta_1 + l_2 \cos(\Theta_1 + \Theta_2), l_1 \sin\Theta_1 + l_2 \sin(\Theta_1 + \Theta_2))$$

Example: Walk Cycle

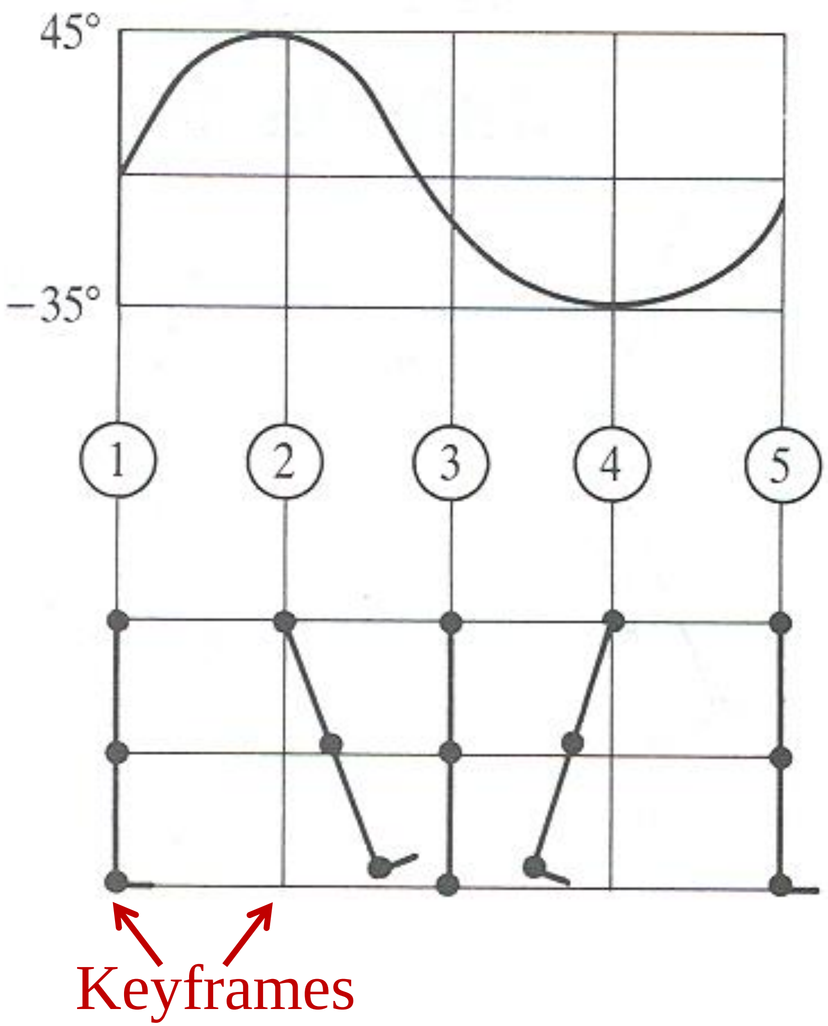
- Articulated figure:



Watt & Watt

Example: Walk Cycle

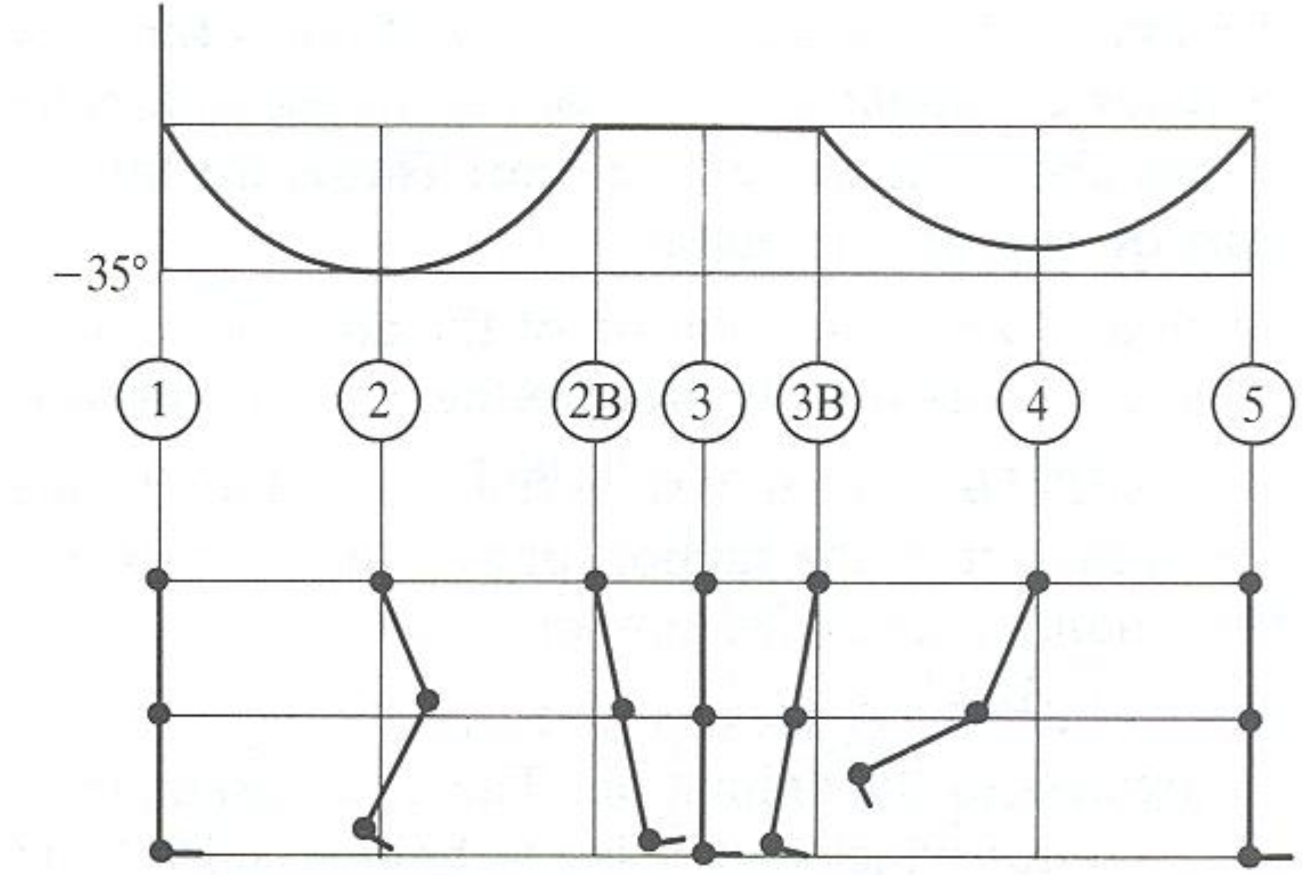
- Hip joint orientation:



Watt & Watt

Example: Walk Cycle

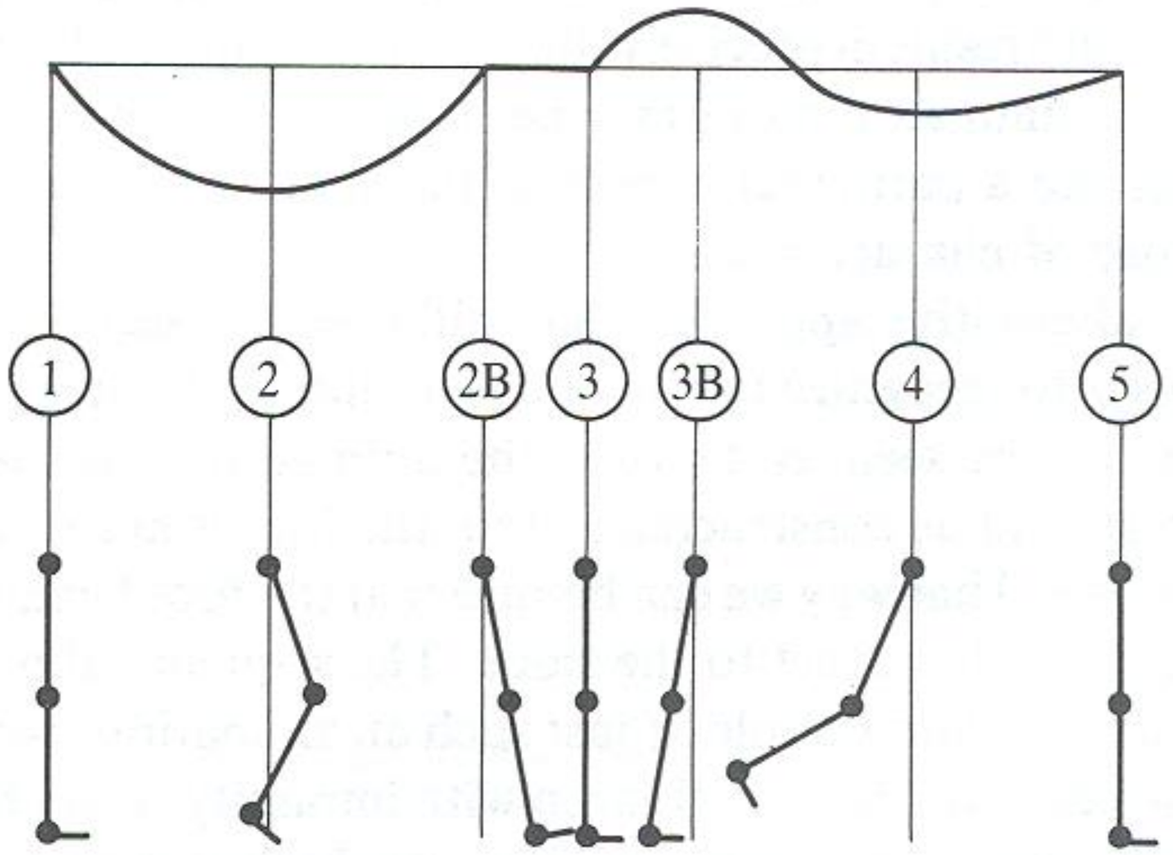
- Knee joint orientation:



Watt & Watt

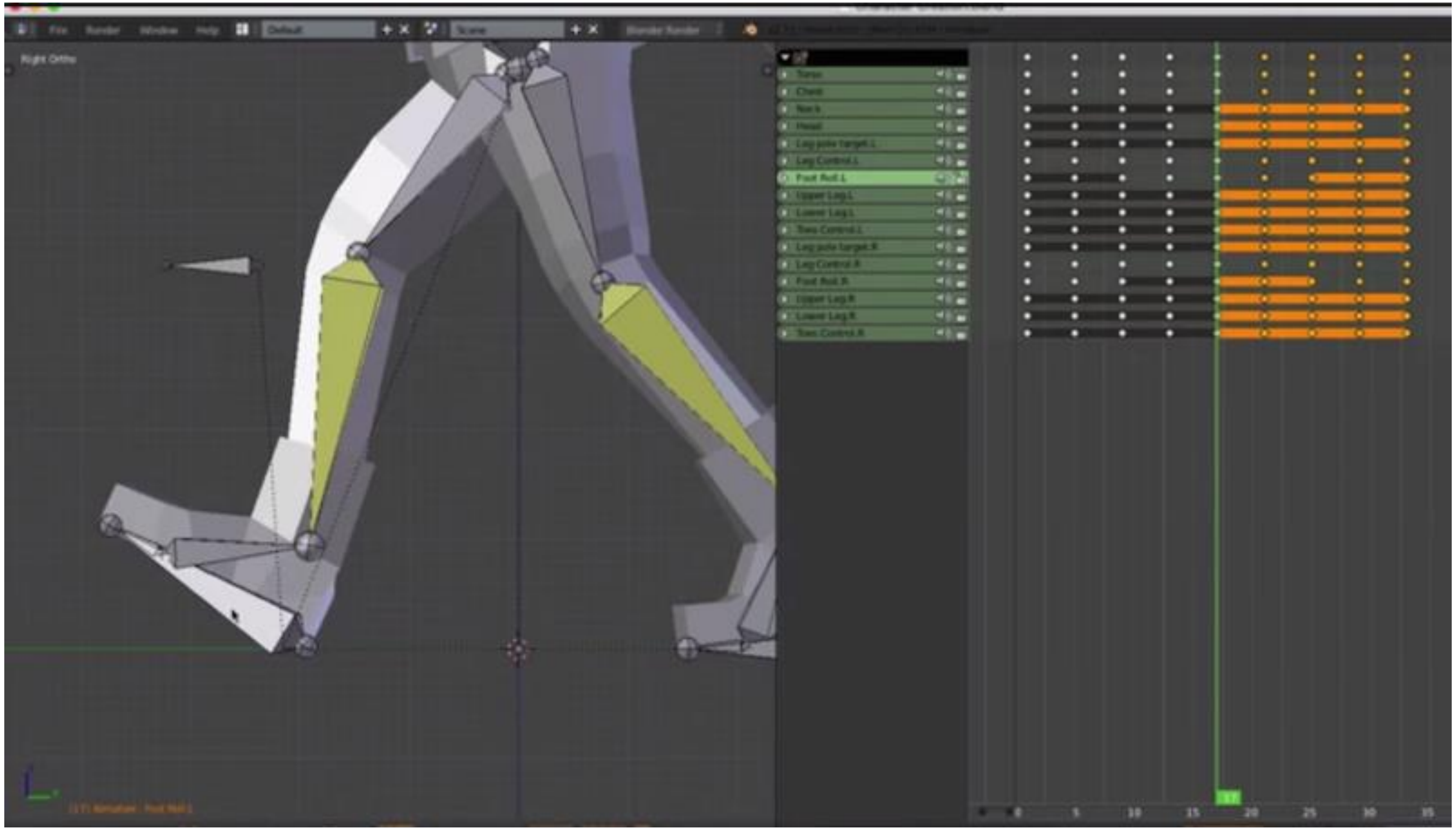
Example: Walk Cycle

- Ankle joint orientation:



Watt & Watt

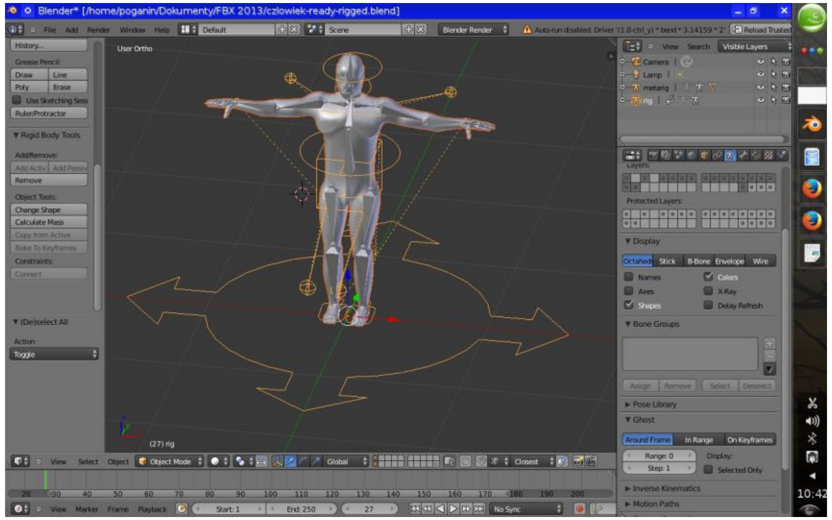
Example: walk cycle



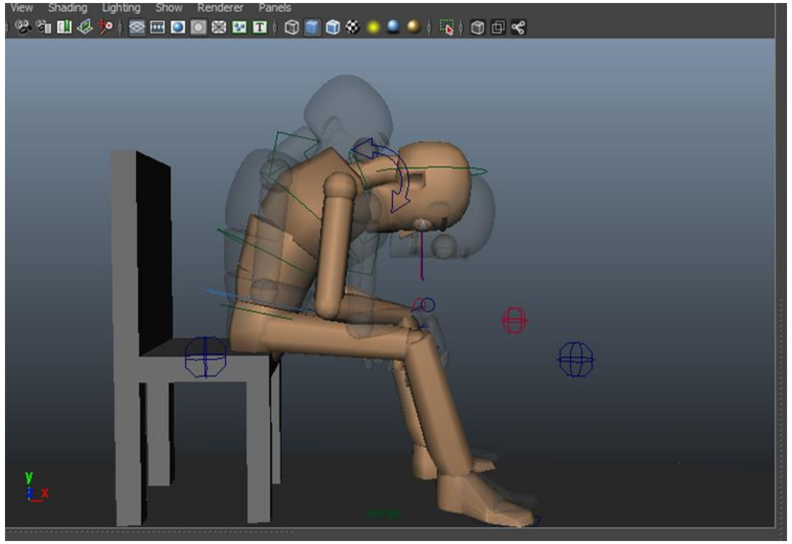
Lague: www.youtube.com/watch?v=DuUWxUitJos

Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation
  - Procedural



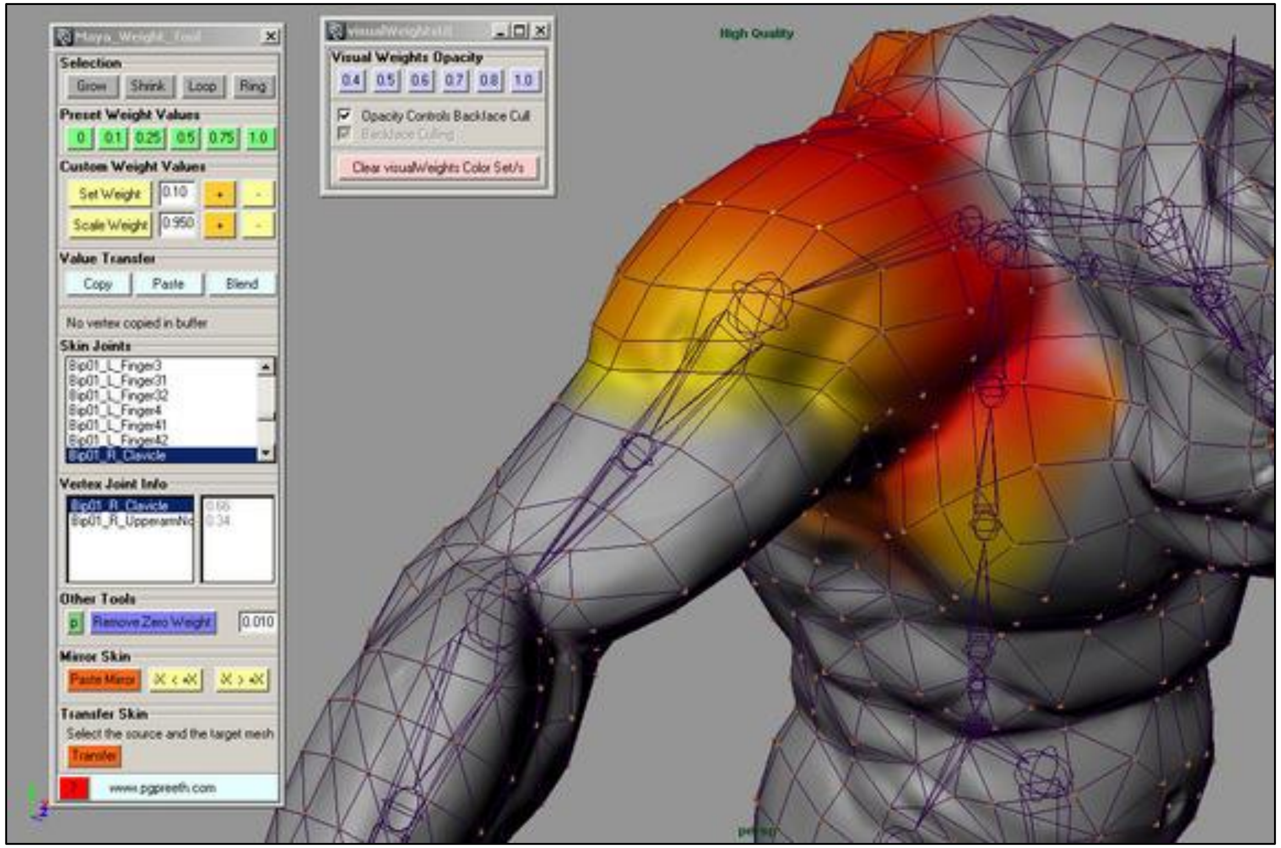
https://blenderartists.org/



focus.gscept.com

Beyond Skeletons…

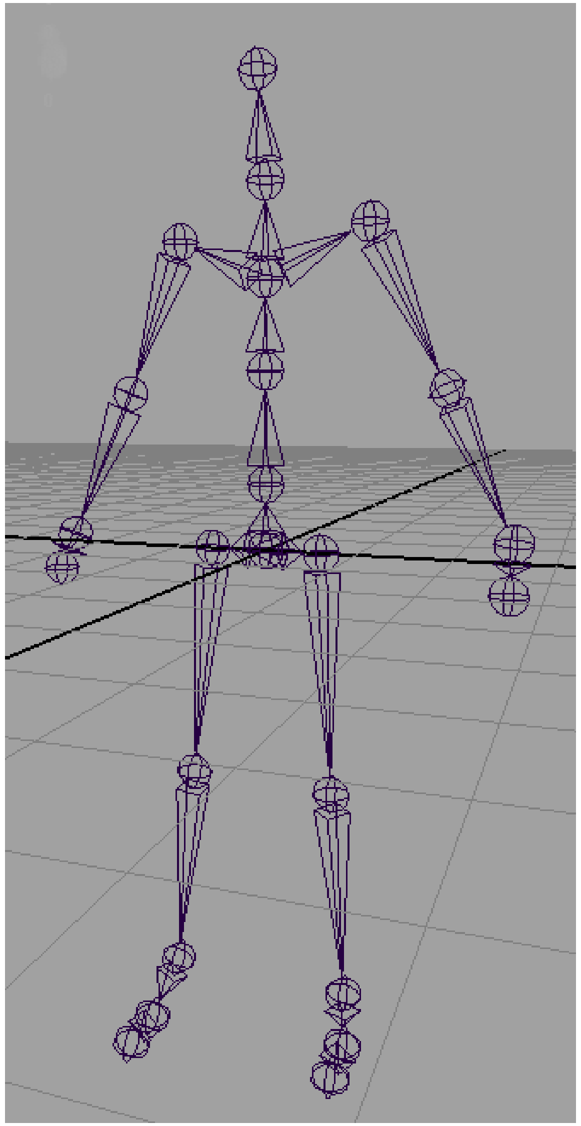
- Skinning



creativecrash.com

Kinematic Skeletons

- Hierarchy of transformations (“bones”)
  - Changes to parent affect all descendent bones
- So far: bones affect objects in scene or parts of a mesh
  - Equivalently, each point on a mesh acted upon by one bone
  - Leads to discontinuities when parts of mesh animated
- Extension: each point on a mesh acted upon by more than one bone



Linear Blend Skinning

- Each vertex of skin potentially influenced by all bones
  - Normalized weight vector $w^{(v)}$ gives influence of each bone transform
  - When bones move, influenced vertices also move

- Computing a transformation T_v for a skinned vertex
  - For each bone
    - Compute global bone transformation T_b from transformation hierarchy
  - For each vertex
    - Take a linear combination of bone transforms
    - Apply transformation to vertex in original pose

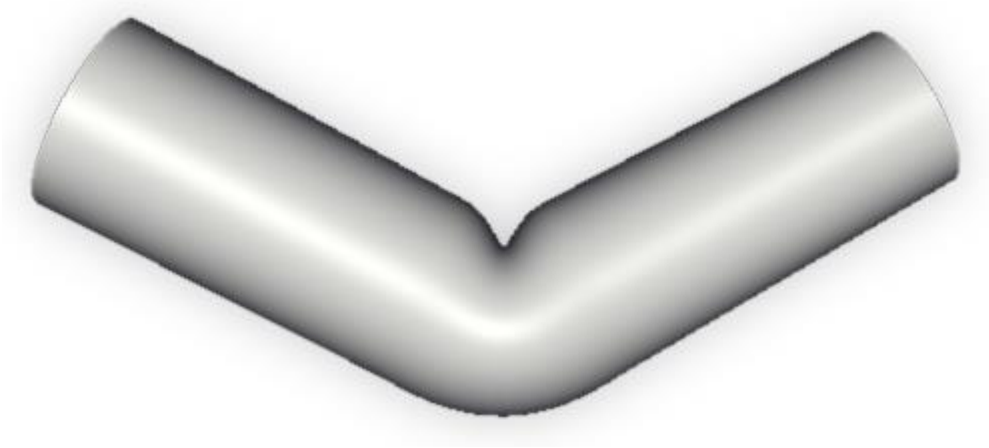
$$T_v = \sum_{b \in B} w_b^{(v)} T_b$$

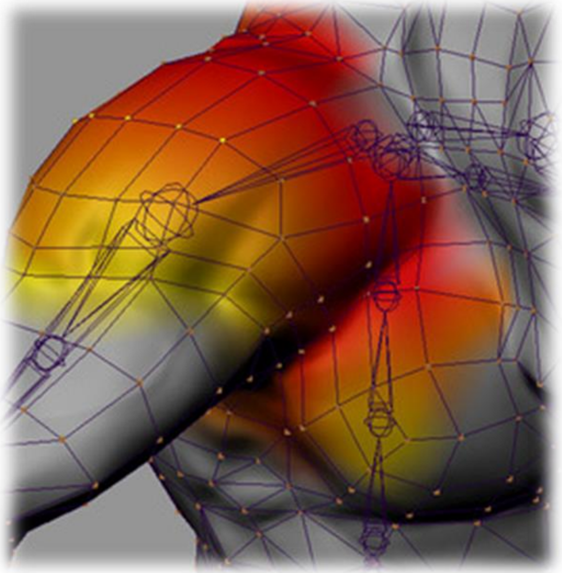
- Equivalently, transformed vertex position is weighted combination of positions transformed by bones

$$v_{transformed} = \sum_{b \in B} w_b^{(v)} \left(T_b v \right)$$

Assigning Weights: “Rigging”

- Painted by hand
- Automatic: function of relative distances to nearest bones
  - Smoothness of skinned surface depends on smoothness of weights!





Assigning Weights: “Rigging”

- Painted by hand
- Automatic: function of relative distances to nearest bones
  - Smoothness of skinned surface depends on smoothness of weights!
  - Other problems with extreme deformations
    - Many solutions

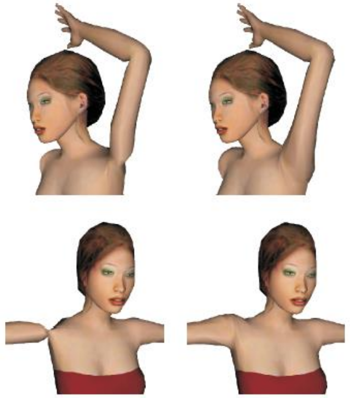
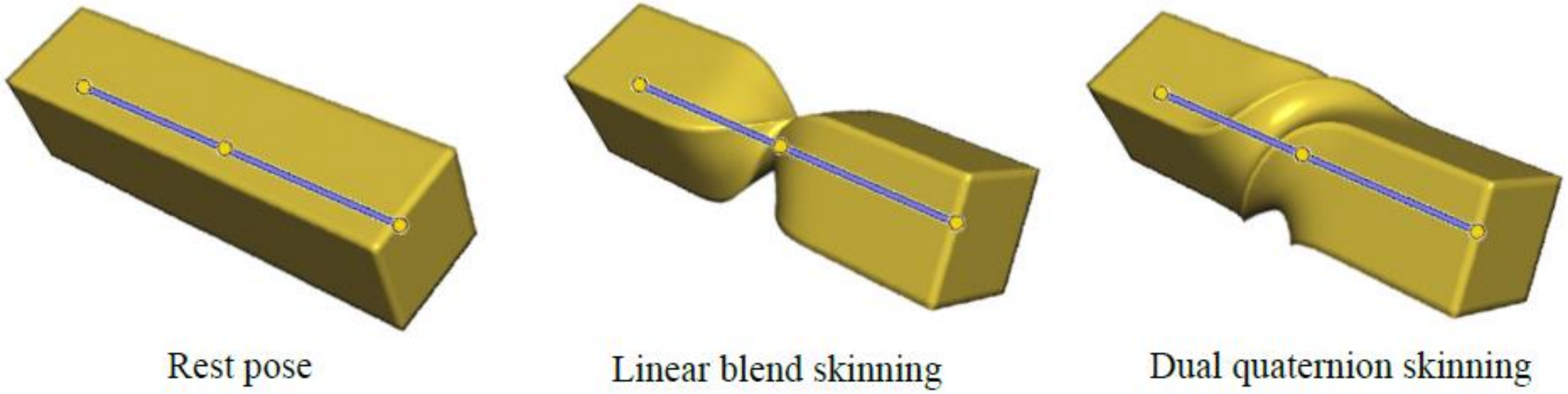


Figure 14: Comparison of linear (left) and dual quaternion (right) blending. Dual quaternions preserve rigidity of input transformations and therefore avoid skin collapsing artifacts.

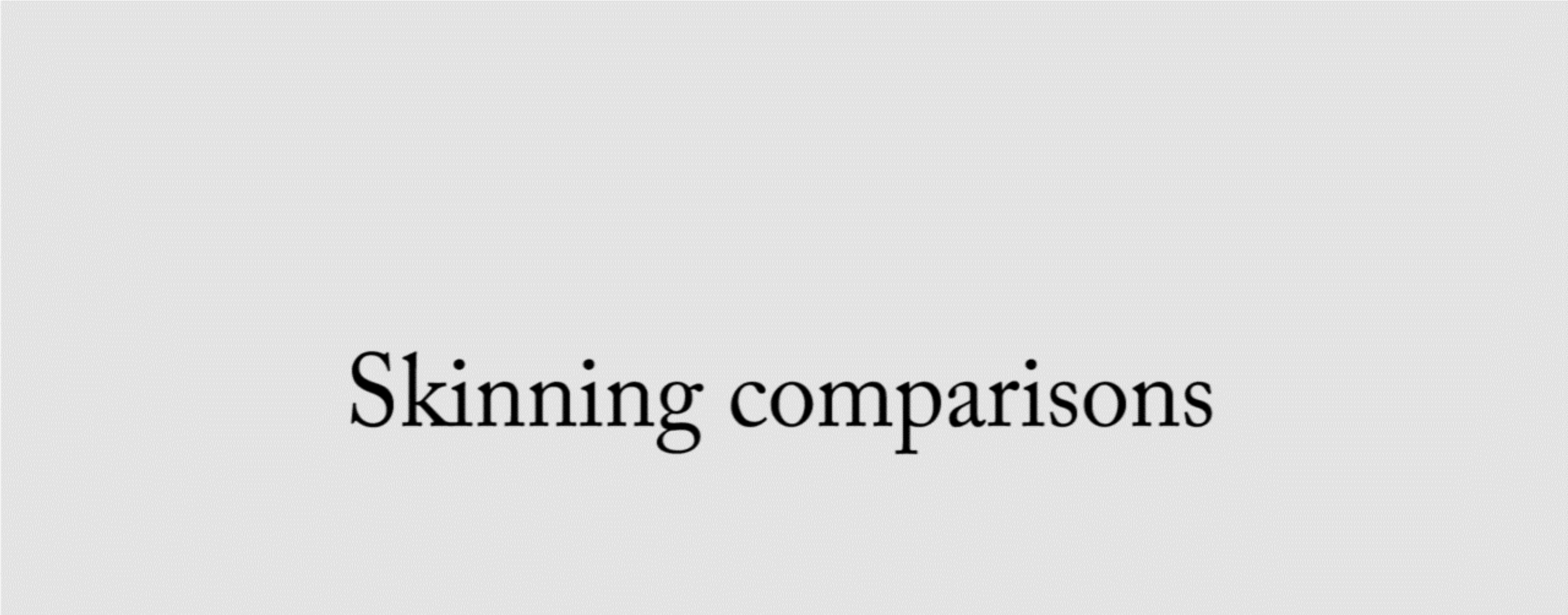
[Kavan et al. SG '08]



[Kavan, SG'14 course]

Assigning Weights: “Rigging”

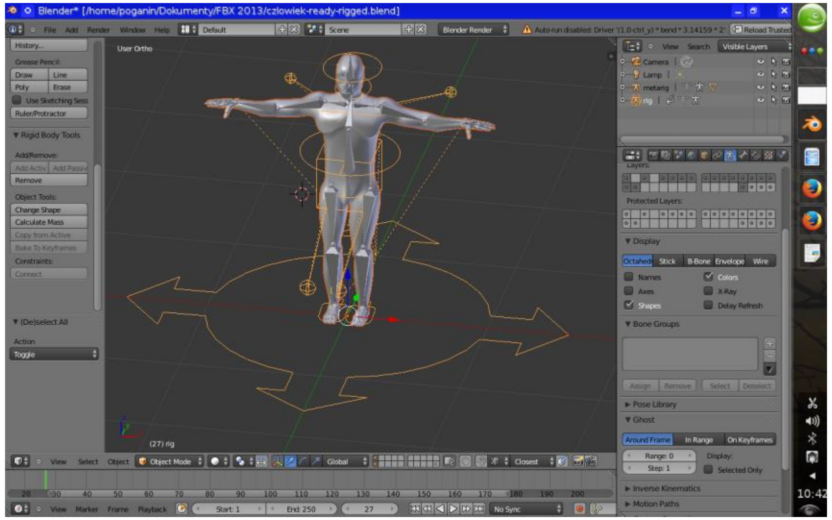
- Painted by hand
- Automatic: function of relative distances to nearest bones
  - Smoothness of skinned surface depends on smoothness of weights!
  - Other problems with extreme deformations



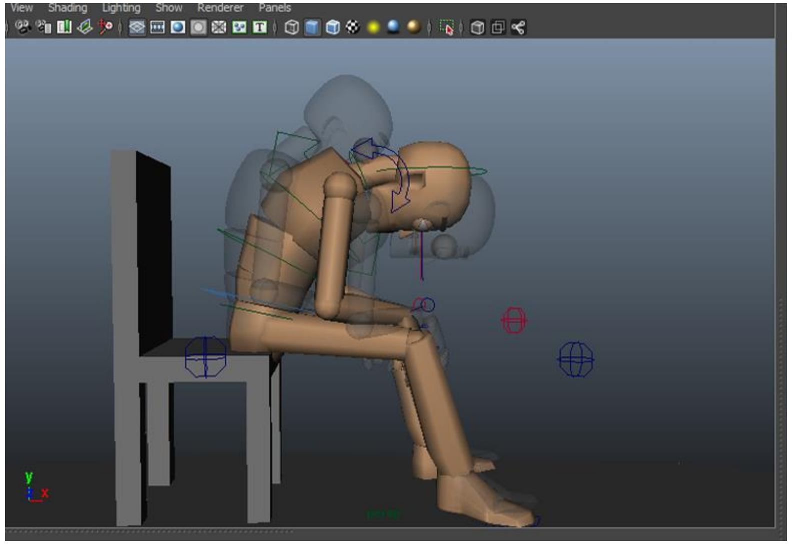
https://cgl.ethz.ch/publications/papers/paperOzt13.php

Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation
  - Procedural



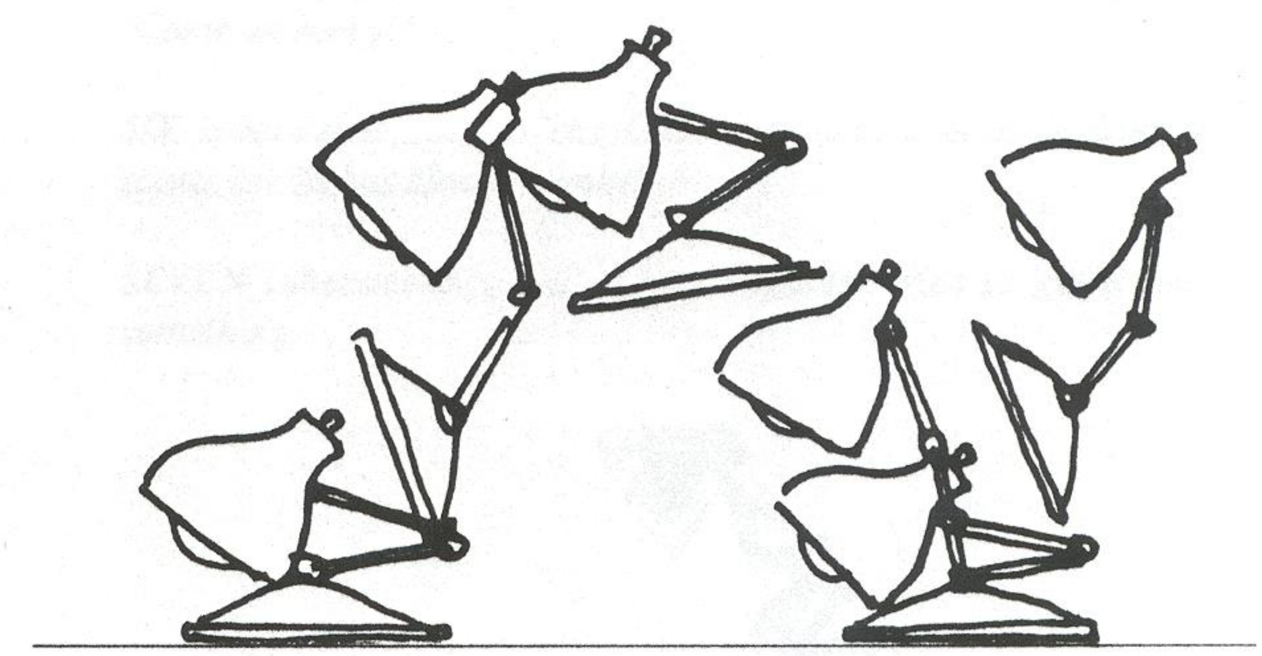
https://blenderartists.org/



focus.gscept.com

Keyframe Animation

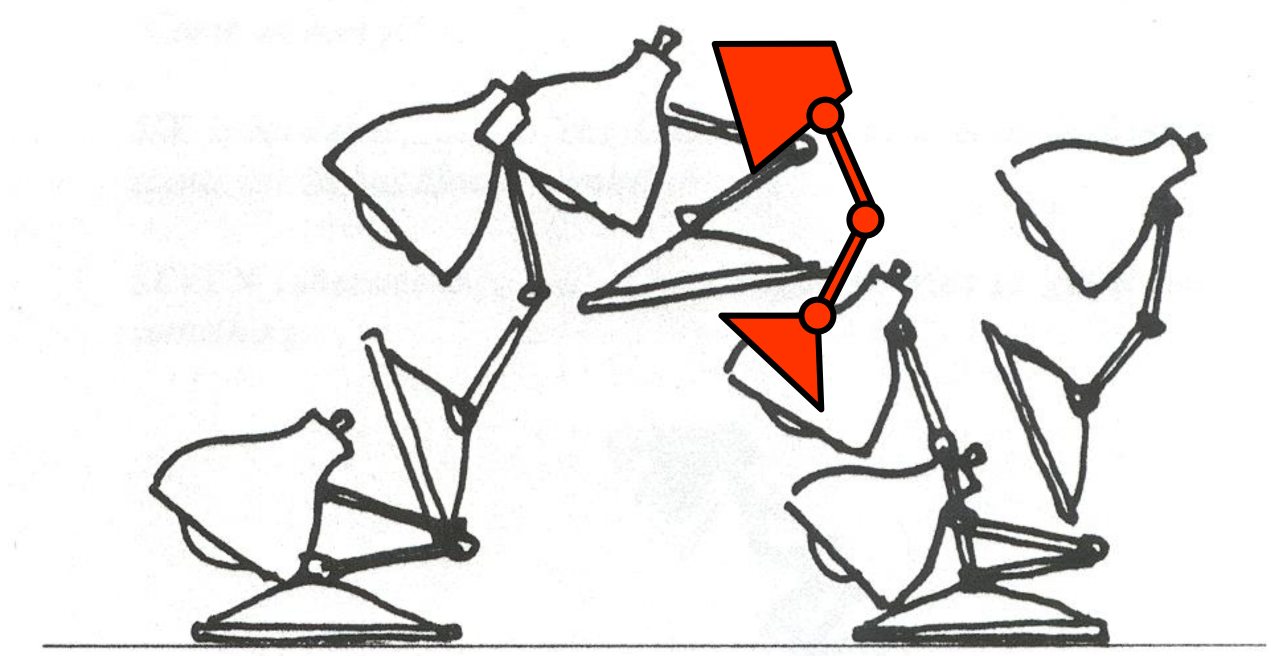
- Define character poses at specific time steps called “keyframes”



Lasseter `87

Keyframe Animation

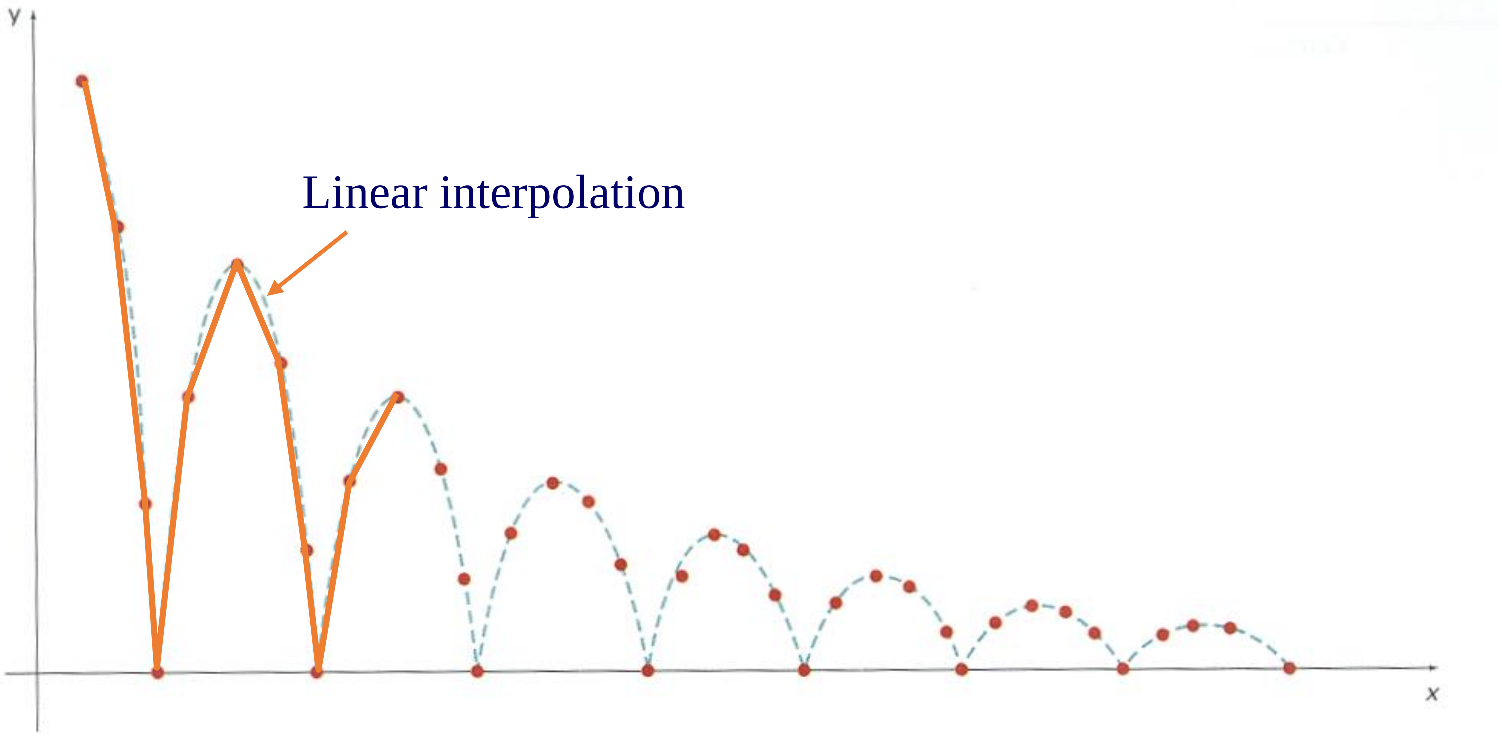
- Interpolate variables describing keyframes to determine poses for character in between



Lasseter `87

Keyframe Animation

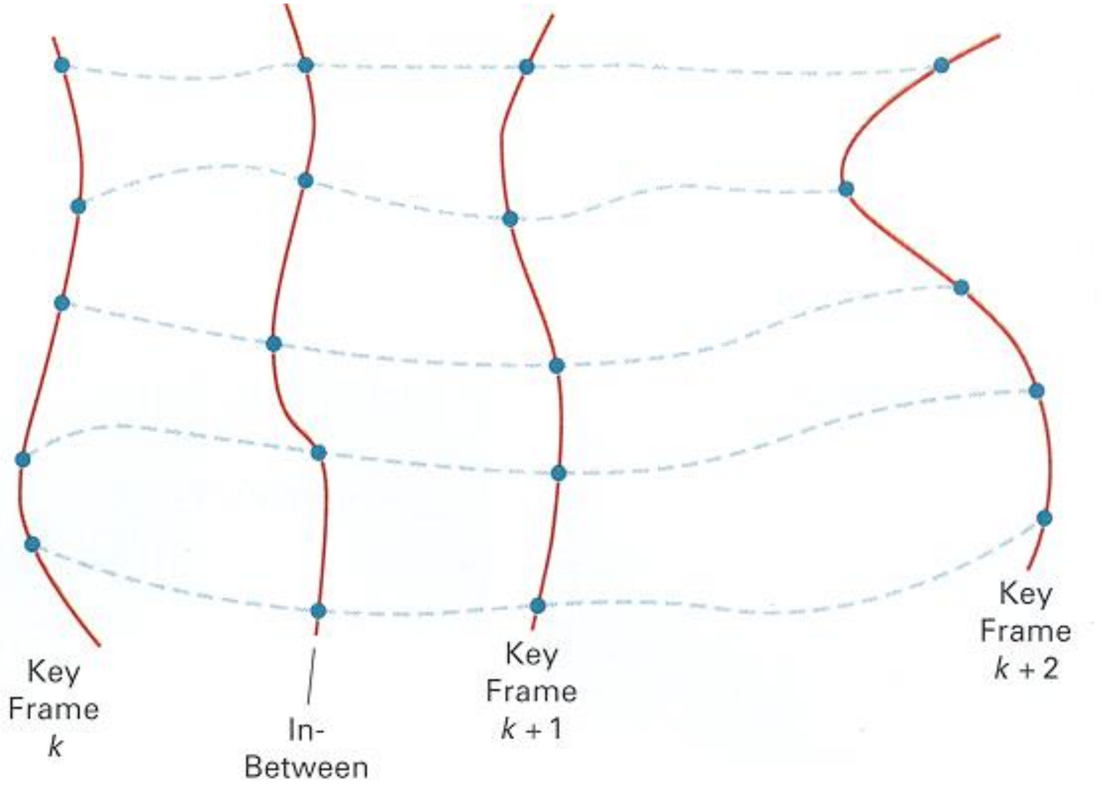
- Inbetweening:
  - Linear interpolation - usually not enough continuity



H&B Figure 16.16

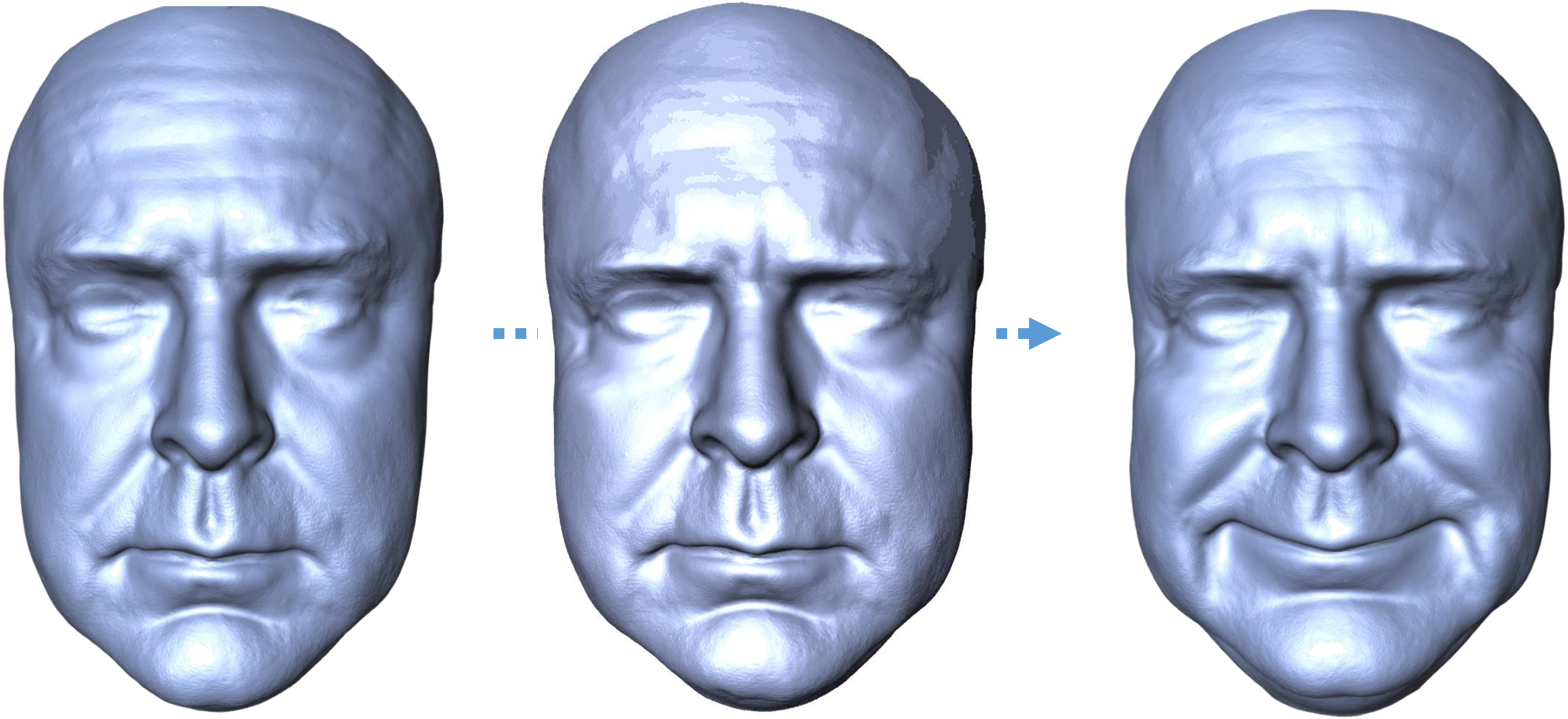
Keyframe Animation

- Inbetweening:
  - Spline interpolation - maybe good enough



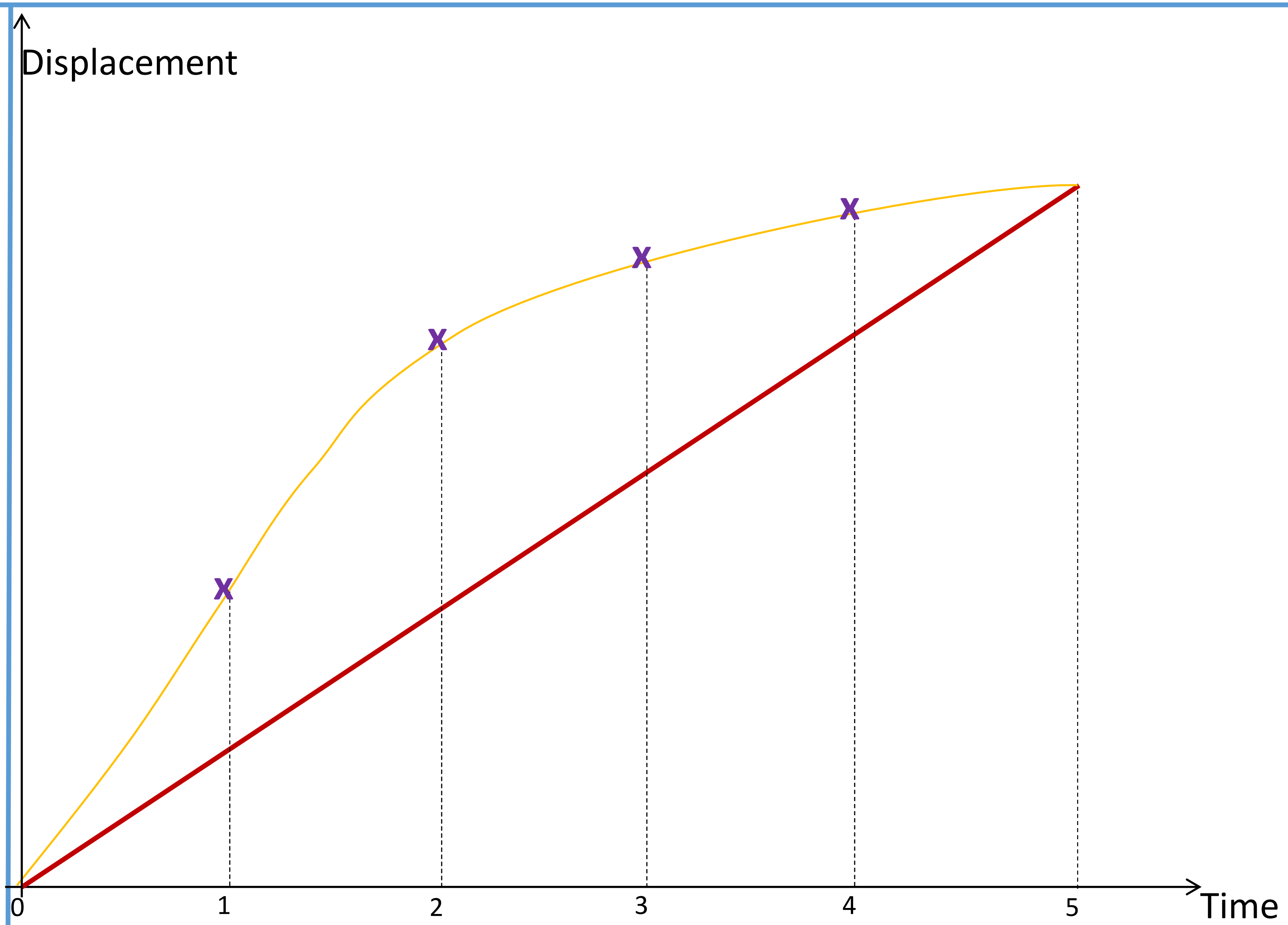
H&B Figure 16.11

Temporal Enhancement



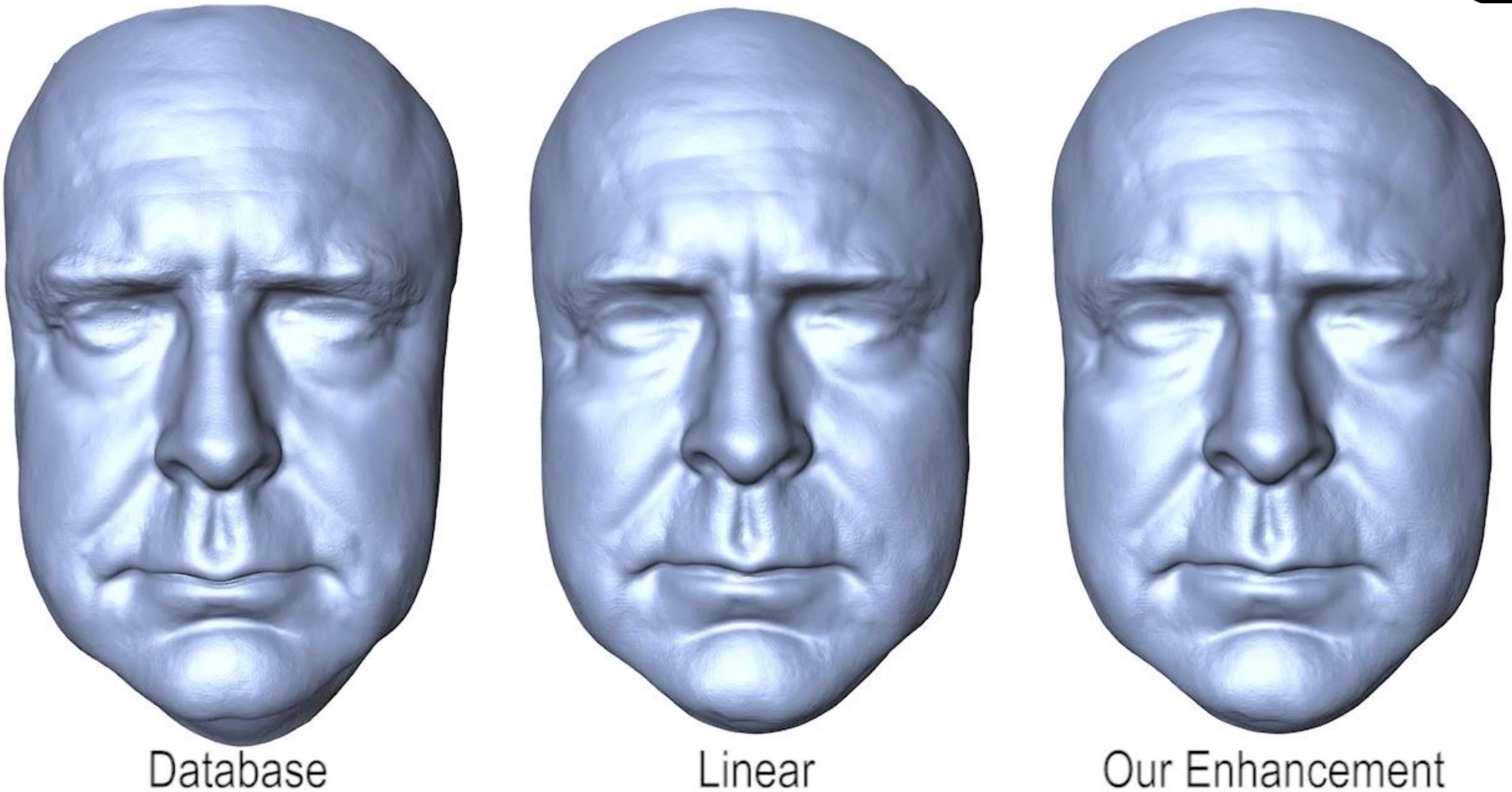
Start frame End frame

Temporal Enhancement



Interpolation
Database

Results – Hand Animated Rig



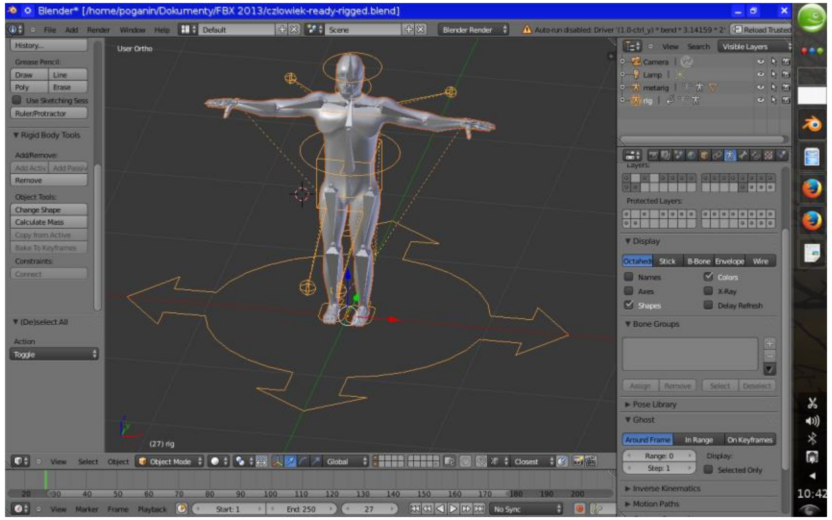
Example: Ball Boy

“Ballboy”

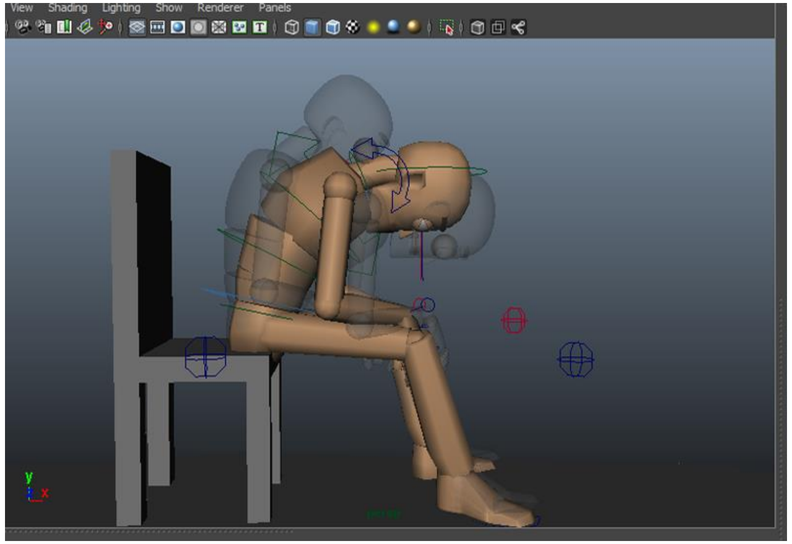
Fujito, Milliron, Ngan, & Sanocki
Princeton University

Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation
  - Procedural



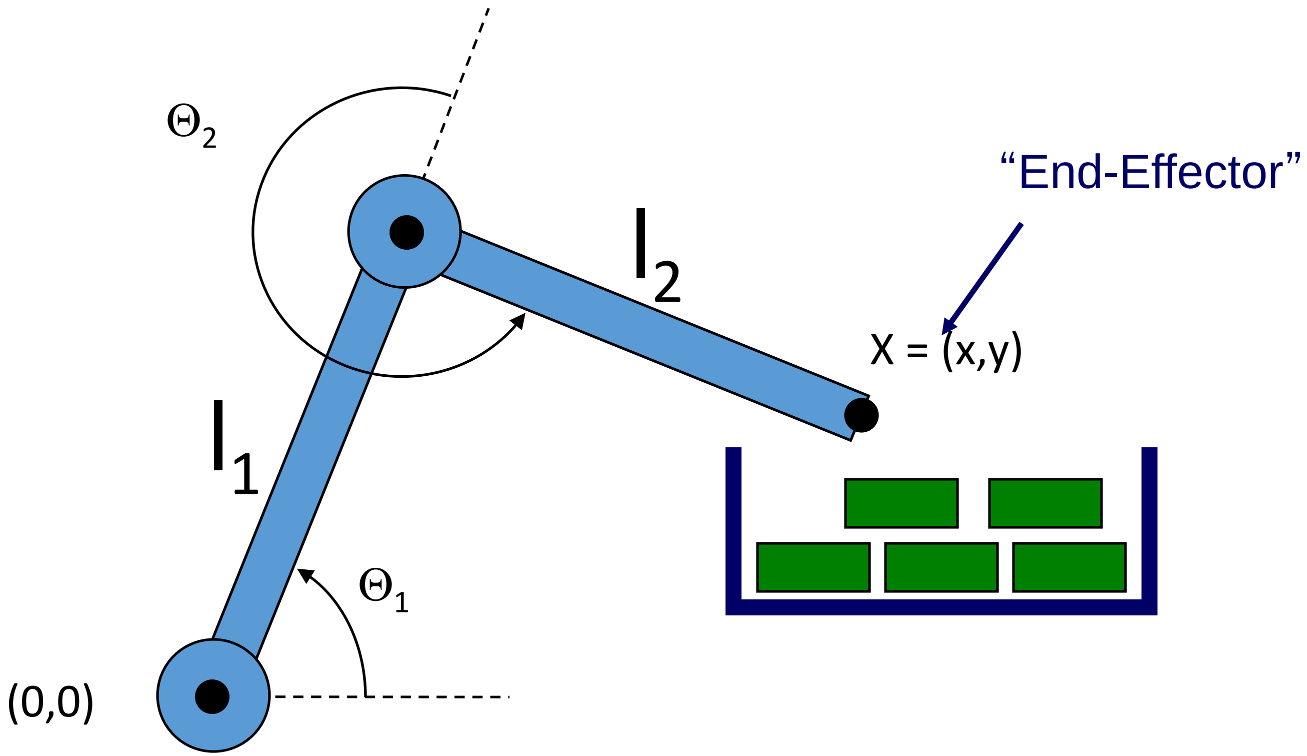
https://blenderartists.org/



focus.gscept.com

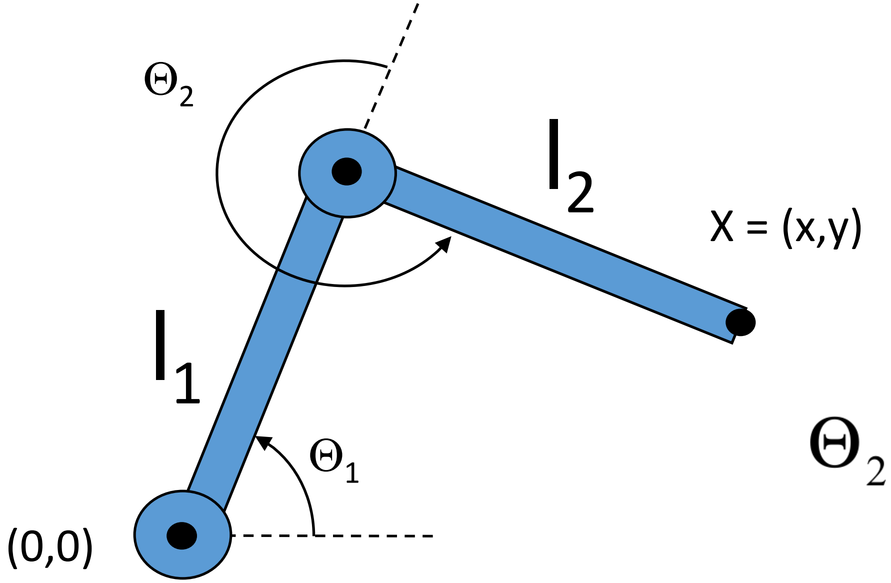
Inverse Kinematics

- What if animator knows position of “end-effector”?



Inverse Kinematics

- Animator specifies end-effector positions: X
- Computer finds joint angles: Θ_1 and Θ_2:

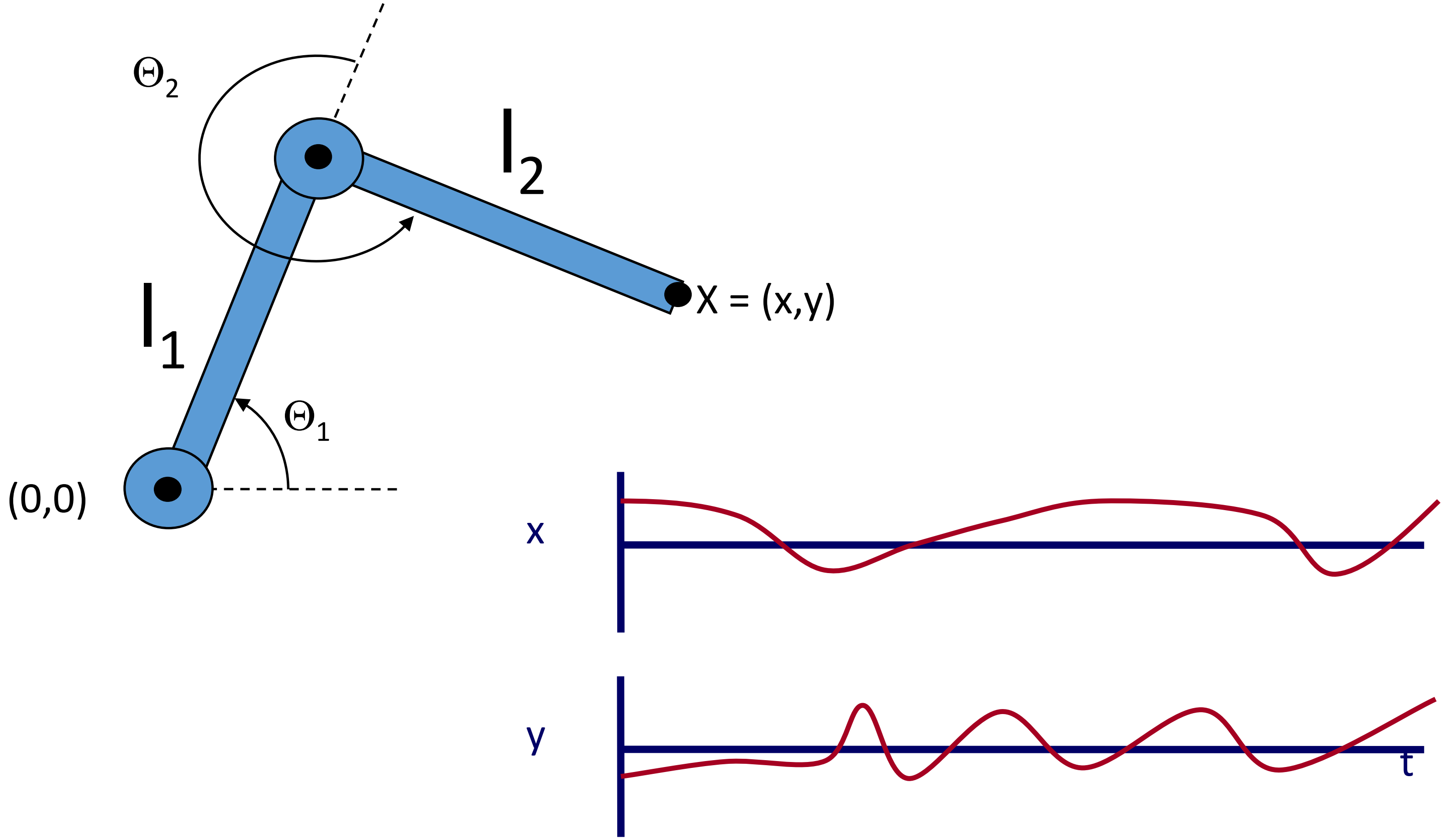


$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + x^2 - l_1{}^2 - l_2{}^2}{2 l_1 l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin(\Theta_2)x + (l_1 + l_2 \cos(\Theta_2))y}{(l_2 \sin(\Theta_2))y + (l_1 + l_2 \cos(\Theta_2))x}$$

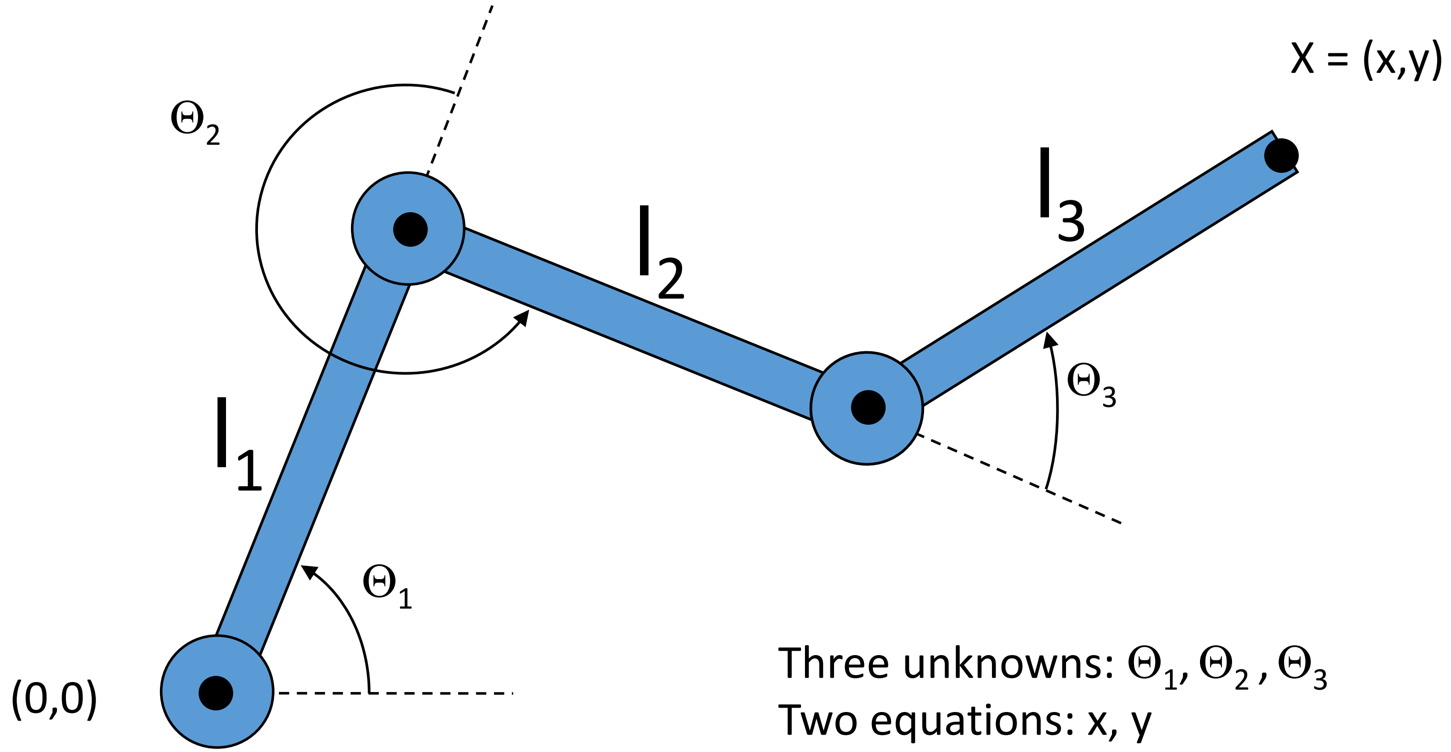
Inverse Kinematics

- End-effector postions can be specified by spline curves



Inverse Kinematics

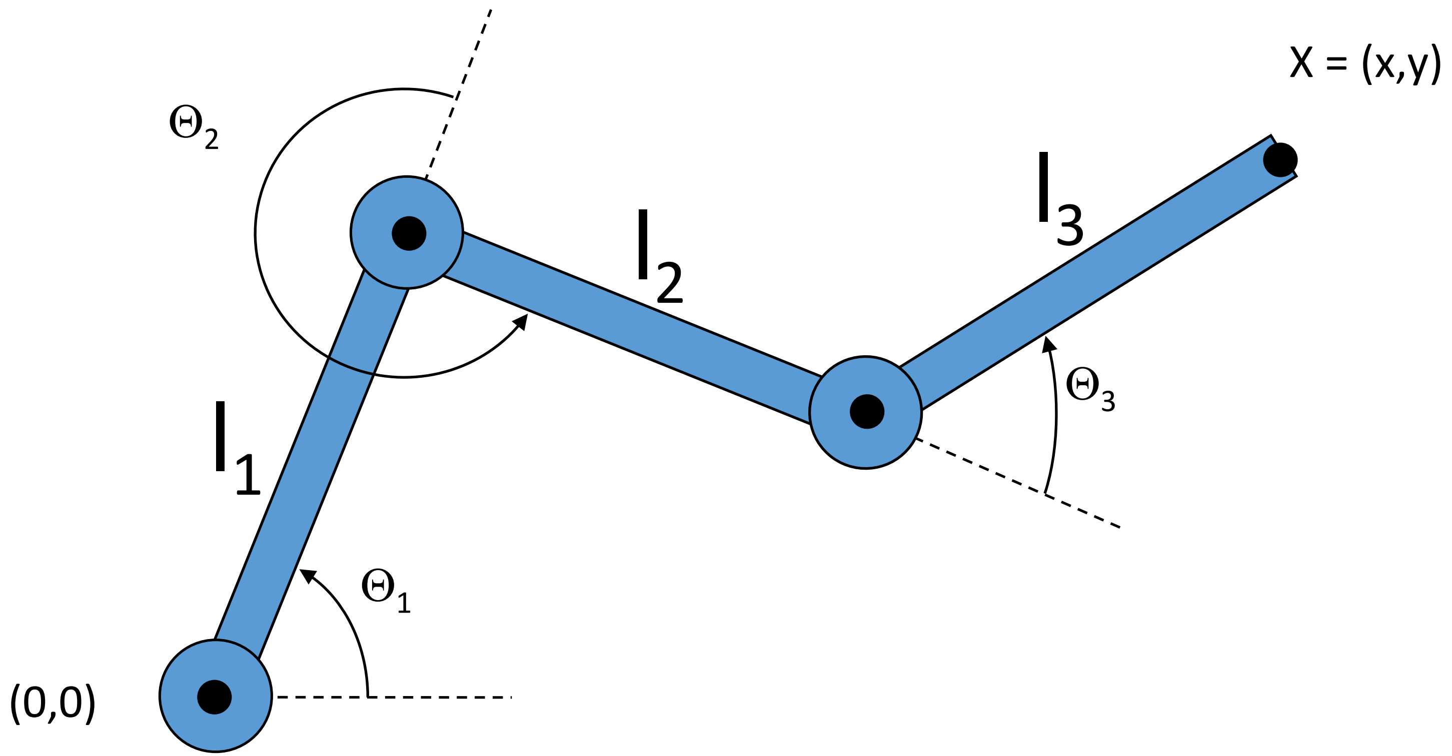
- Problem for more complex structures
  - System of equations is usually under-constrained
  - Multiple solutions



Three unknowns: $\Theta_1, \Theta_2, \Theta_3$
Two equations: x, y

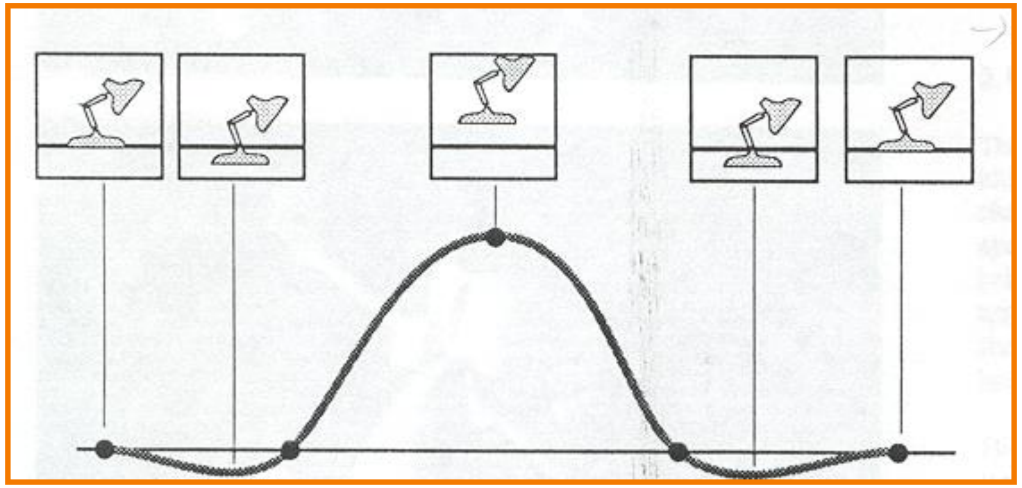
Inverse Kinematics

- Solution for more complex structures:
  - Find best solution (e.g., minimize energy in motion)
  - Non-linear optimization



Kinematics

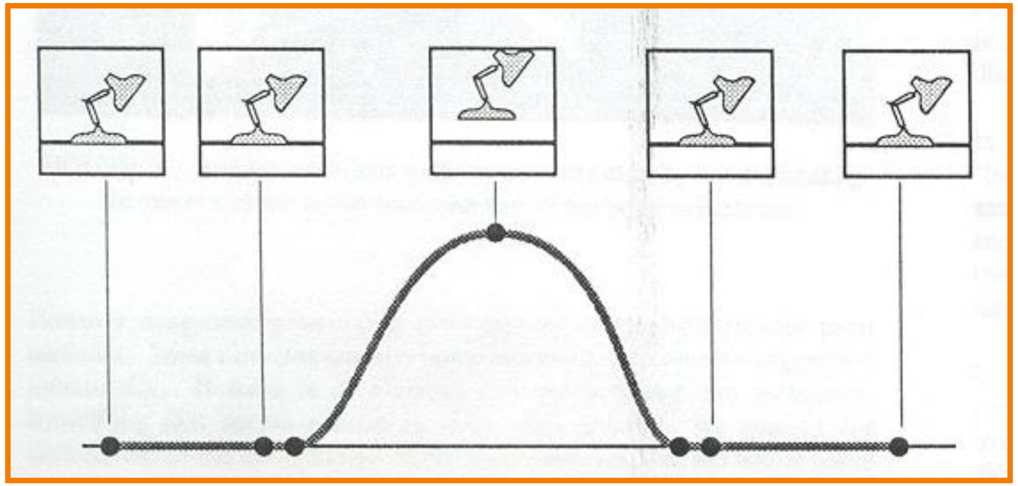
- Advantages
  - Simple to implement
  - Complete animator control
- Disadvantages
  - Motions may not follow physical laws
  - Tedious for animator



Lasseter `87

Kinematics

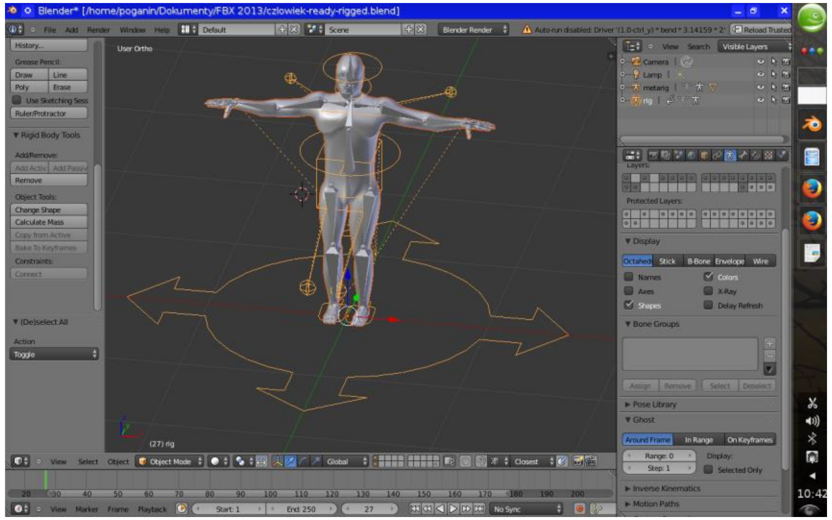
- Advantages
  - Simple to implement
  - Complete animator control
- Disadvantages
  - Motions may not follow physical laws
  - Tedious for animator



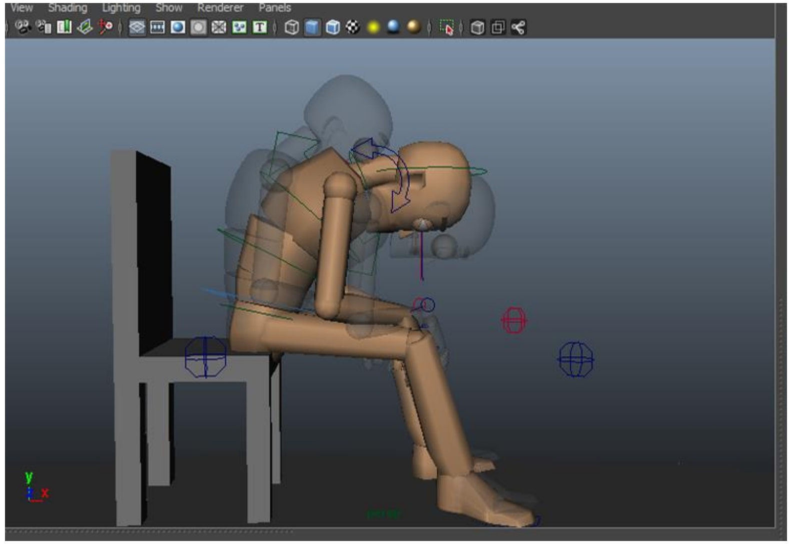
Lasseter `87

Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation
  - Procedural



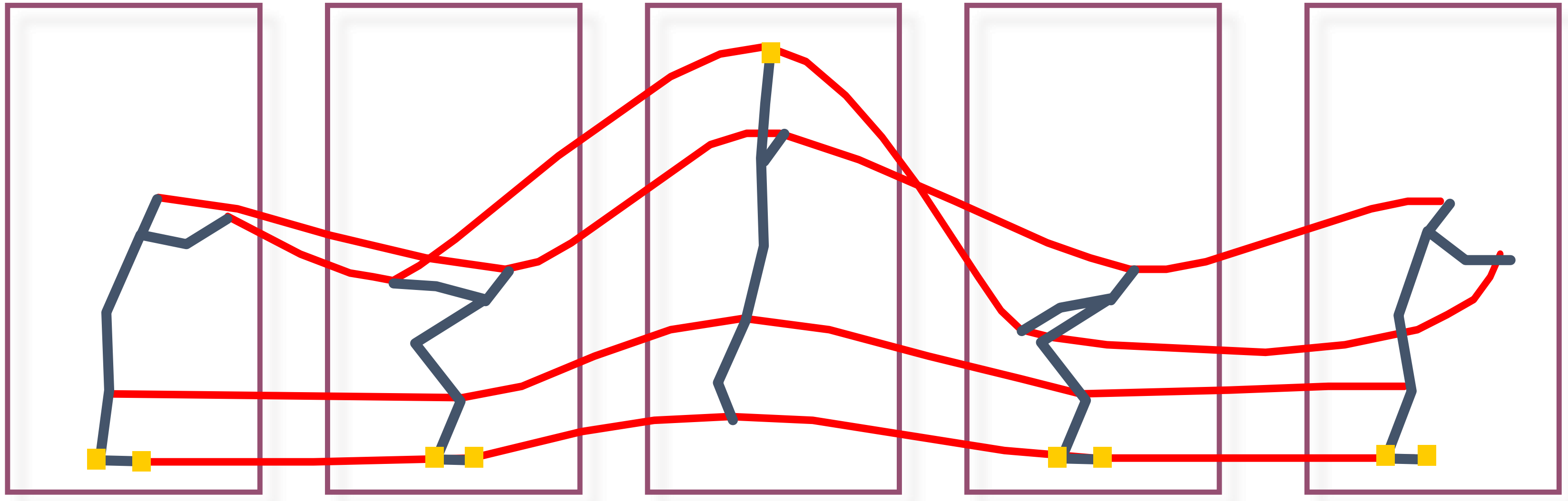
https://blenderartists.org/



focus.gscept.com

Motion Capture

- Measure motion of real characters and then simply “play it back” with kinematics



Captured Motion

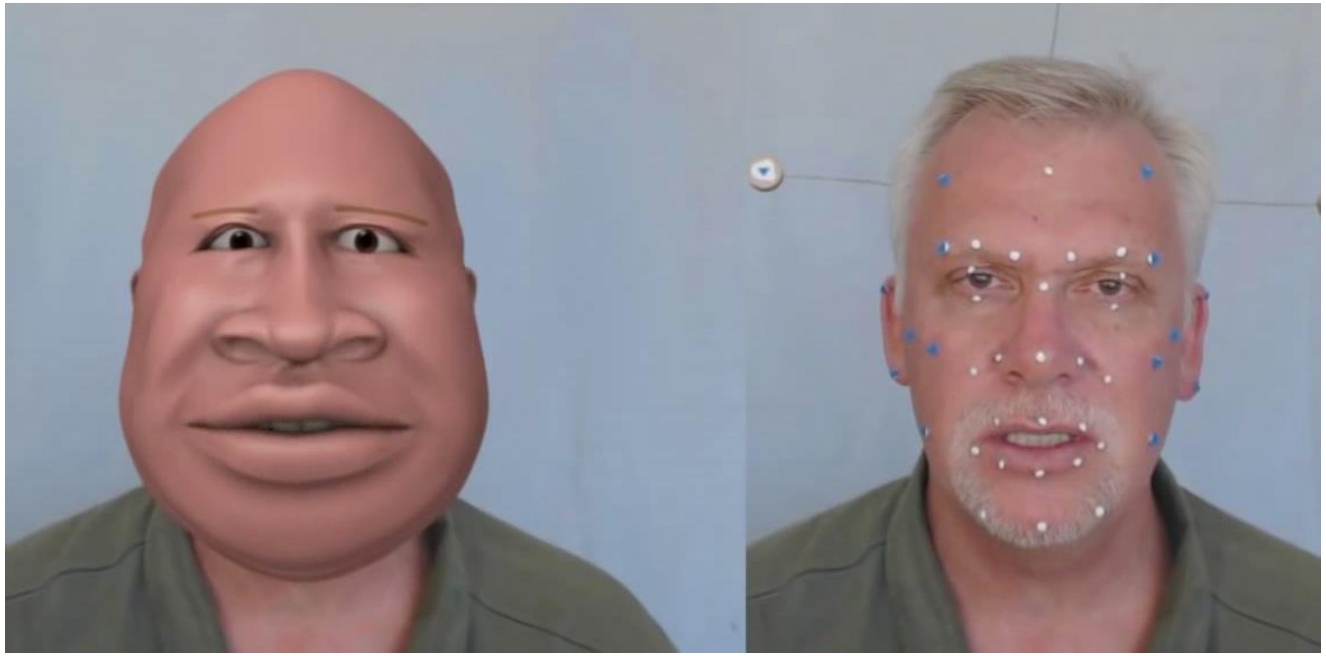
Motion Capture

- Measure motion of real characters and then simply “play it back” with kinematics

https://www.youtube.com/watch?v=MVvDw15-3e8

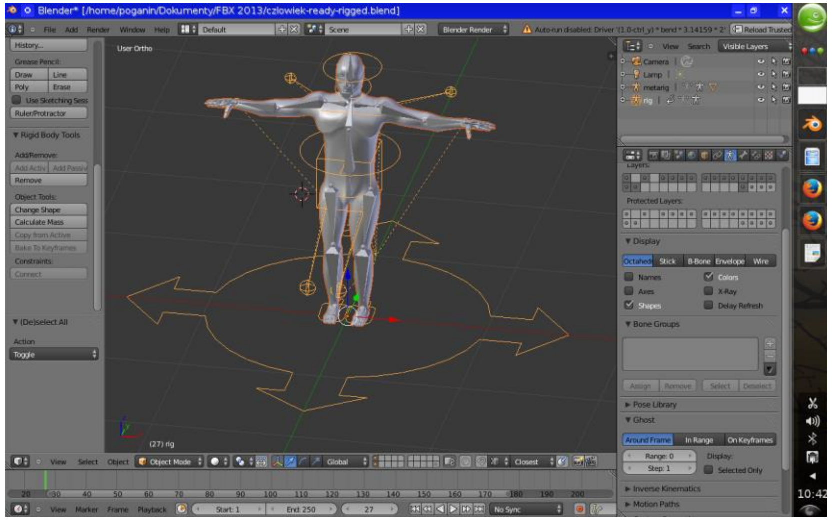
Motion Capture

- Could be applied on different parameters
  - Skeleton Transformations
  - Direct mesh deformation
- Advantage:
  - Physical realism
- Challenge:
  - Animator control

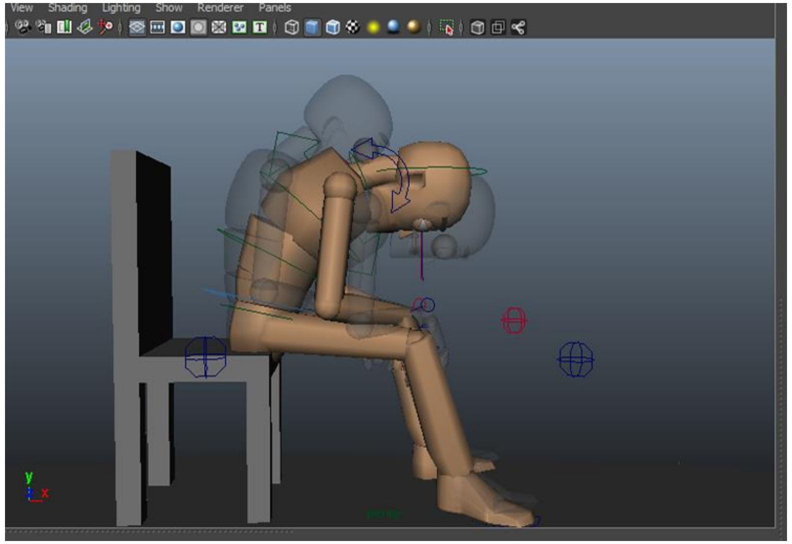


Character Animation Methods

- Modeling (manipulation)
  - Deformation
  - Blendshape rigging
  - Skeleton+Envelope rigging
- Interpolation
  - Key-framing
    - Kinematics
  - Motion Capture
  - Energy minimization
    - Physical simulation

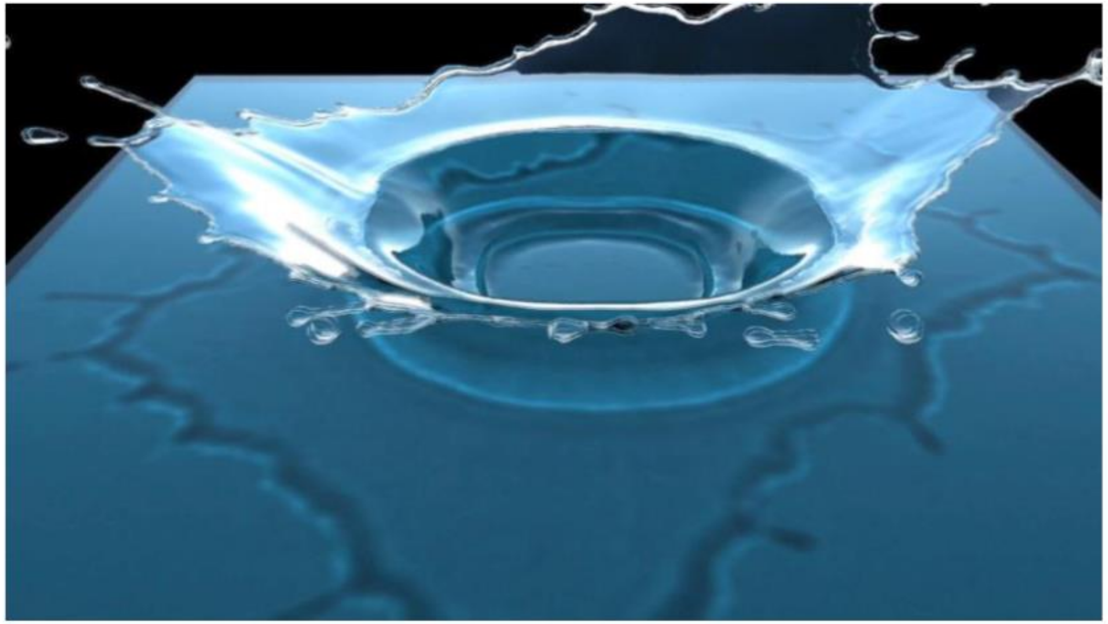


https://blenderartists.org/



focus.gscept.com

Animation Techniques



Keyframing

- good for characters and simple motion
- but many physical systems are too complex

Physically Based Simulation

- Equations known for a long time
  - Motion (Newton, 1660)
  - Elasticity (Hooke, 1670)
  - Fluids (Navier, Stokes, 1822)

$$d/dt(m\mathbf{v}) = \mathbf{f}$$

$$\boldsymbol{\sigma} = \mathbf{E}\boldsymbol{\varepsilon}$$

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = -k\nabla\rho + \rho\mathbf{g} + \mu\nabla^2\mathbf{v}$$

- Simulation made possible by computers

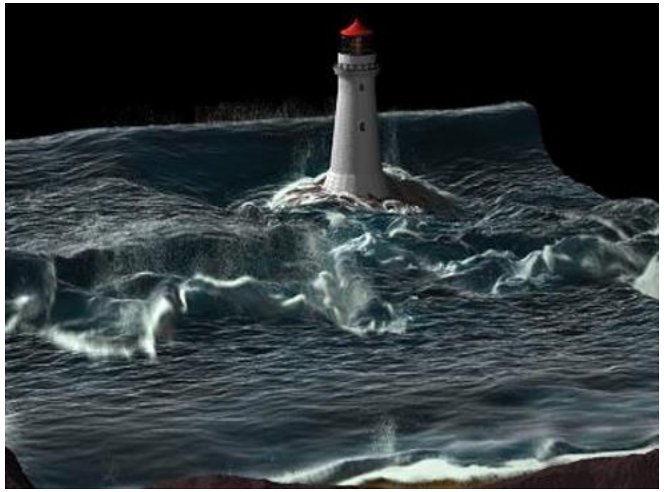
1938: Zuse Z1

2014: Tianhe-2 @ NUDT (China)

PBS and Graphics

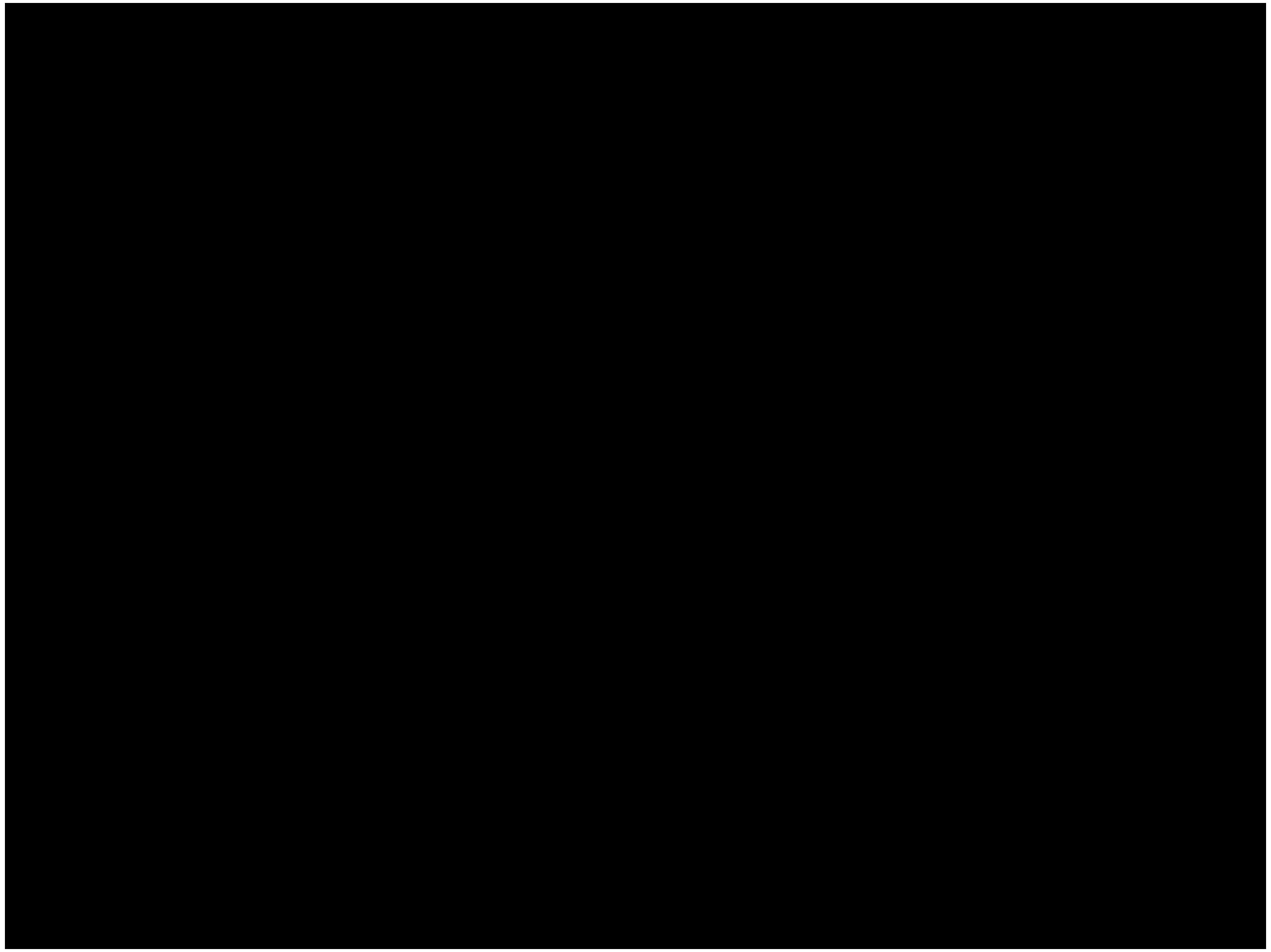
Physically-based simulation

- Computational Sciences
  - **Reproduction** of physical phenomena
  - Predictive capability (accuracy!)
  - Substitute for expensive experiments
- Computer Graphics
  - **Imitation** of physical phenomena
  - Visually plausible behavior
  - Speed, stability, art-directability



Simulation in Graphics

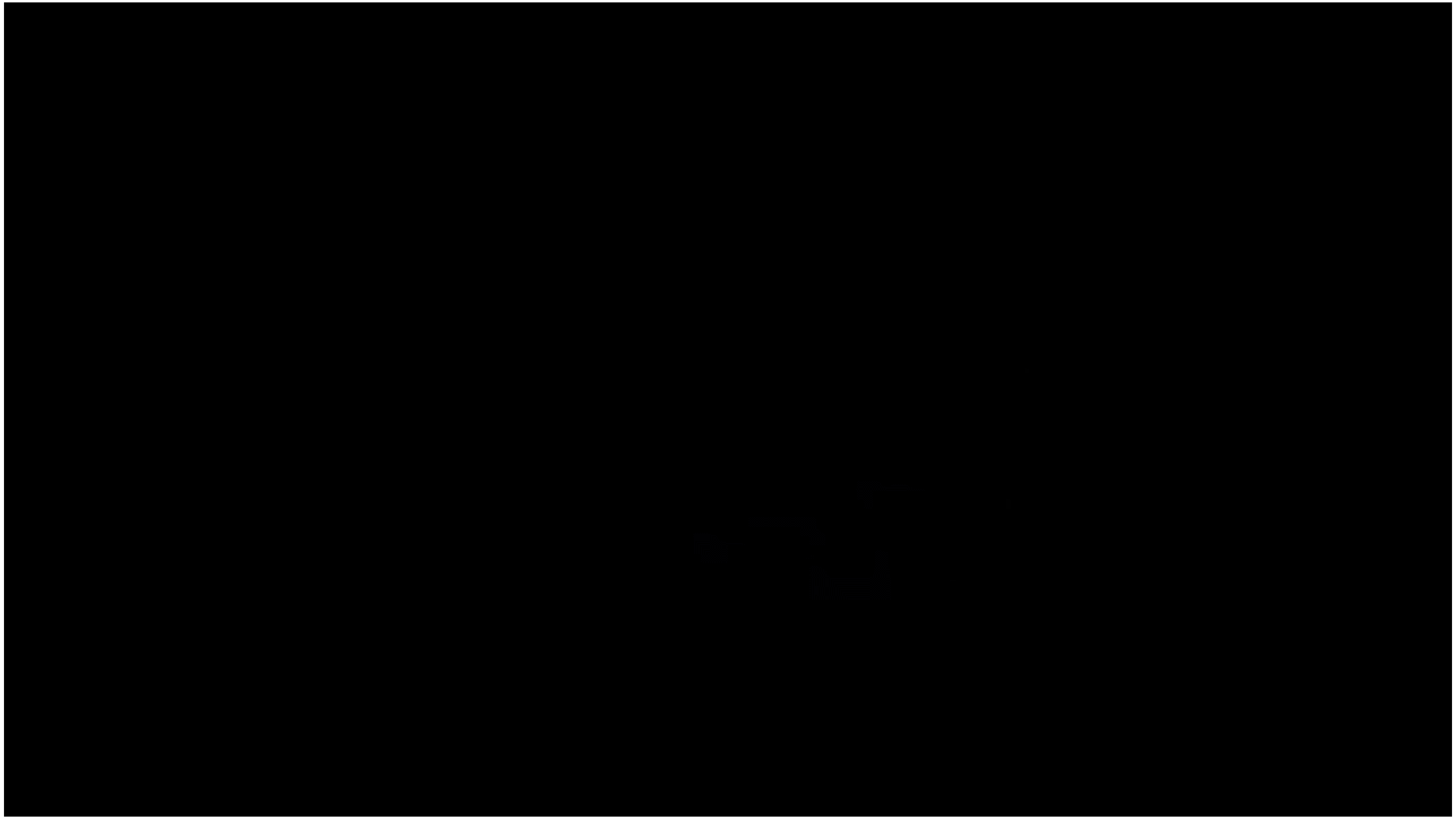
- Art-directability



Simulation in Graphics

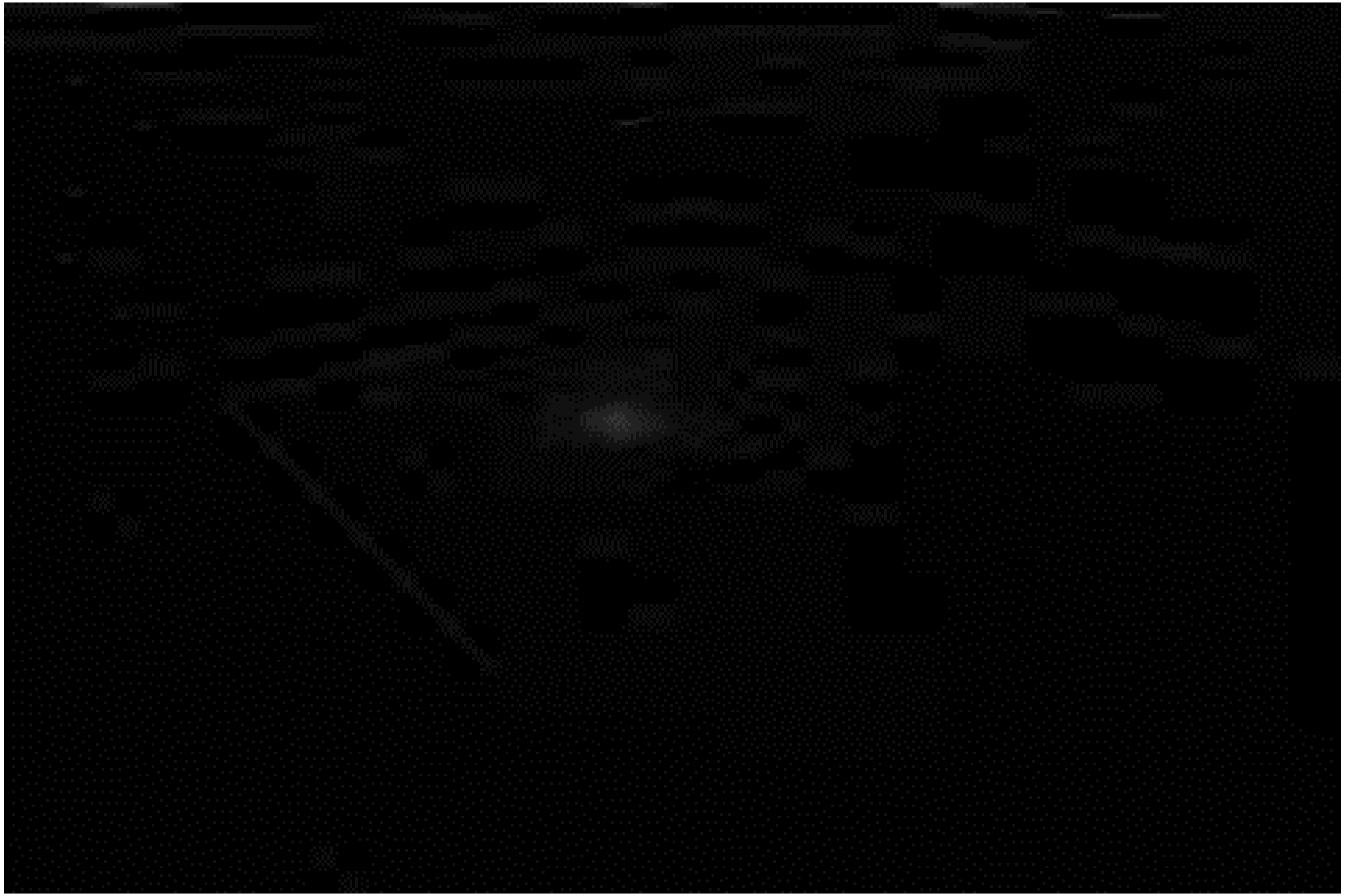
- Speed

https://www.youtube.com/watch?v=-x9B_4qBAkk



Simulation in Graphics

- Stability



https://www.youtube.com/watch?v=tT81VPk_ukU

Applications in Graphics

Offline physically-based simulation

- Visual quality
- Controllability

Advertisement

Animated movies

Live-Action

Applications in Graphics

Real-time physically-based simulation

- Performance and stability
- Accuracy according to application

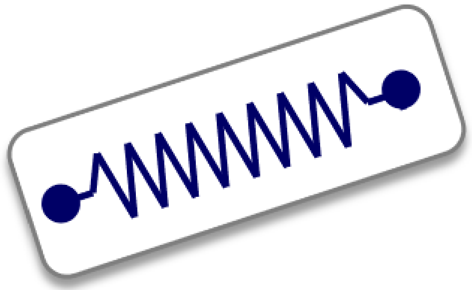
Surgery trainers

Virtual Reality

Computer Games

Mass – Spring Systems

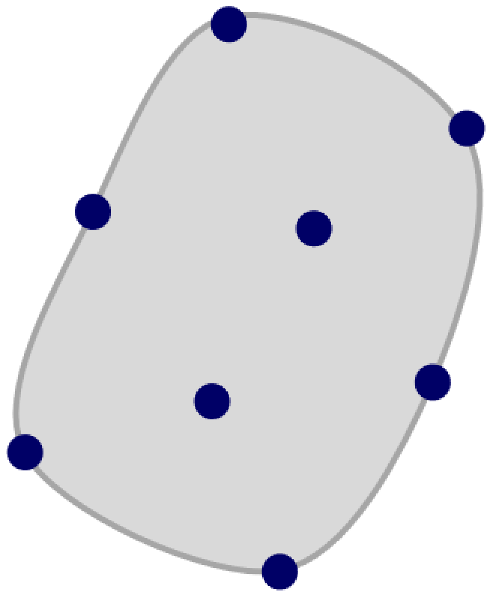
Mass-Spring Systems



Steps towards simulation

1. **Spatial discretization**: sample object with mass points
2. **Forces**: define internal (*springs!*) and external forces
3. **Dynamics**: set up equations of motion
4. **Temporal discretization**: solve equations of motion

Spatial Discretization



Sample object with mass points

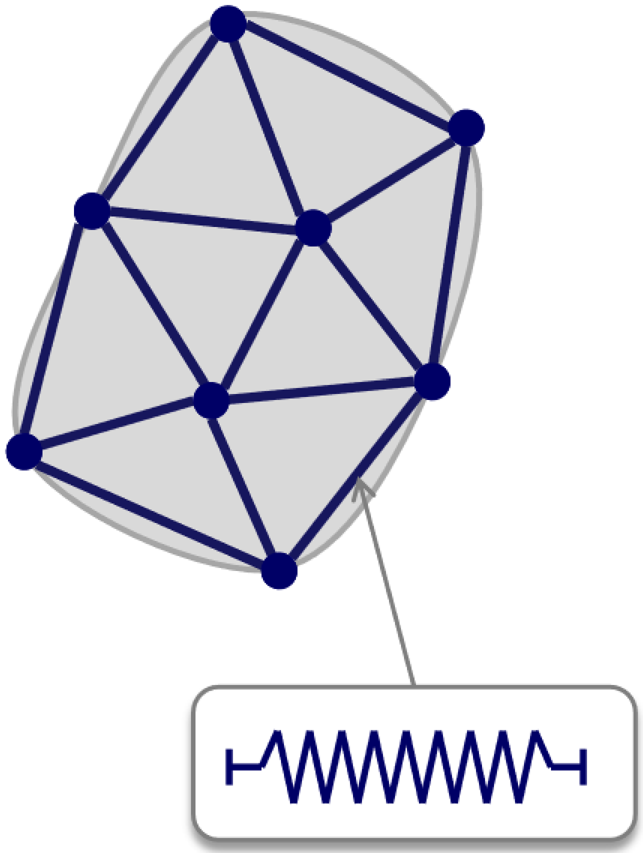
- Total mass of object: M
- Number of mass points: n
- Mass of each point: $m = M/n$
  (*uniform distribution*)

Each point holds properties

- Mass m_i
- Position $\mathbf{x}_i(t)$
- Velocity $\mathbf{v}_i(t)$

Forces

What are the forces that act on particle i?



External forces

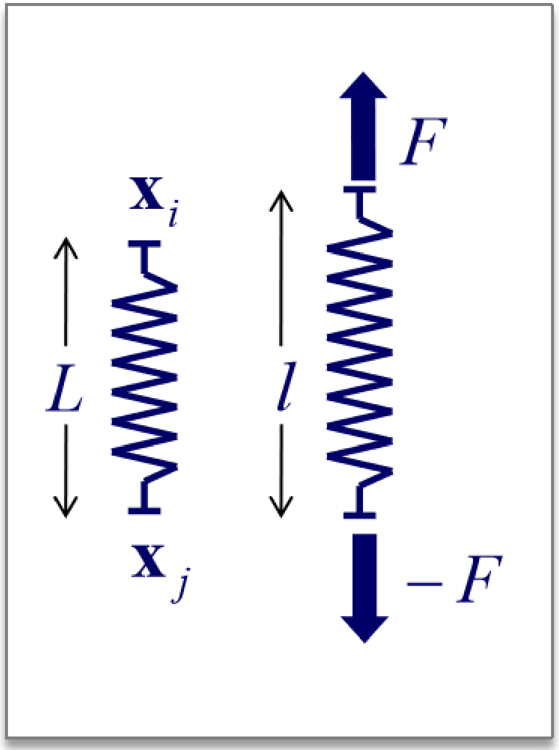
– Gravity $\mathbf{F}_i^g = m_i \begin{pmatrix} 0 \\ 0 \\ 9.81 \end{pmatrix} \frac{m}{s^2}$

Internal forces

– Elastic spring forces

– Viscous damping forces

Forces



Force in 1D

$$F = -k(l - L)$$ Hooke's Law

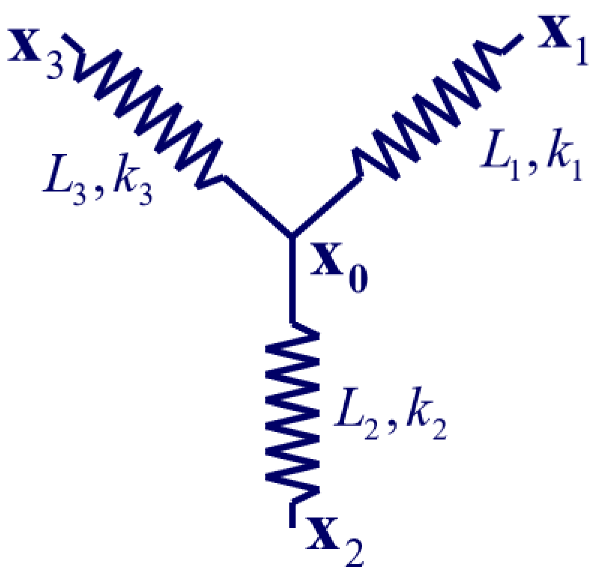
Force in 3D

$$\mathbf{F}_i = -k\left(\|\mathbf{x}_i - \mathbf{x}_j\| - L\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}$$

Initial spring length L
Current spring length l
Spring stiffness k

Forces

Internal forces $\mathbf{F}^{\text{int}}$



External forces $\mathbf{F}^{\text{ext}}$

- Gravity
- Contact forces
- All forces that are not caused by springs

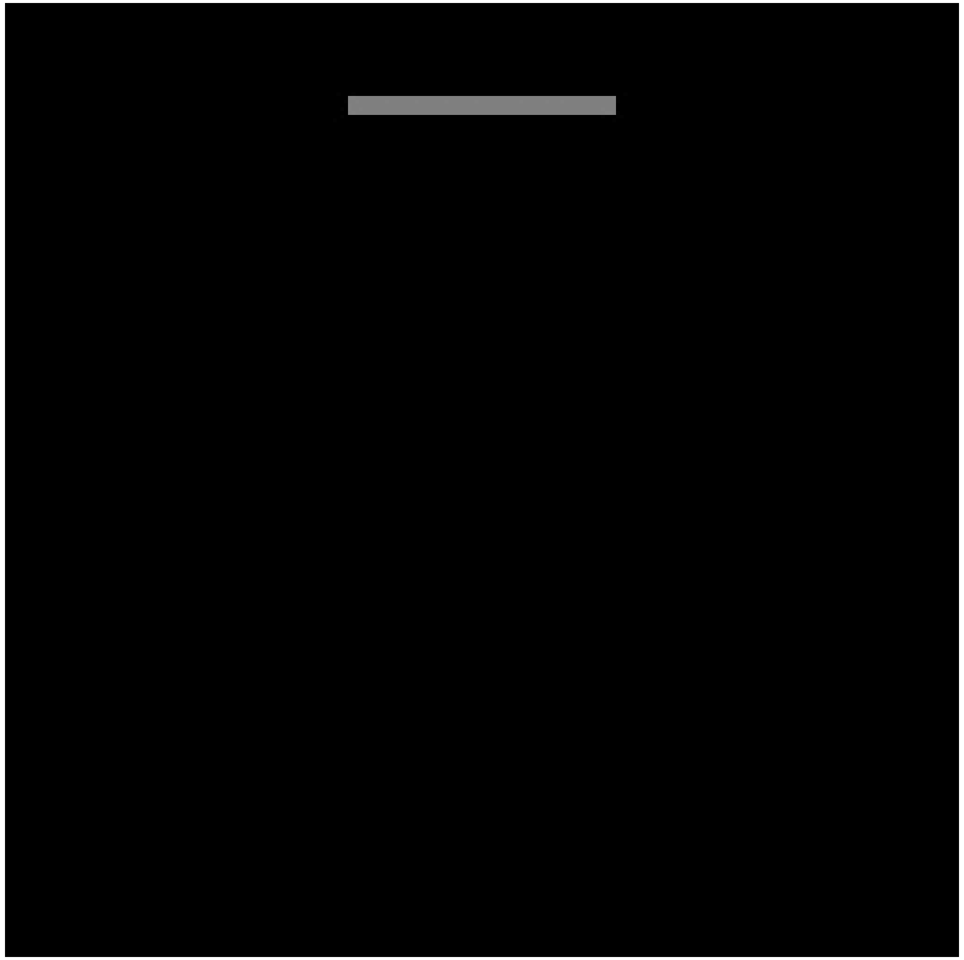
Total spring force

$$\mathbf{F}_0^{\text{int}} = -\sum_{i \mid i \in \{1,2,3\}} k_i (l_i - L_i) \frac{\mathbf{x}_i - \mathbf{x}_0}{l_i}$$

Resulting force at point i

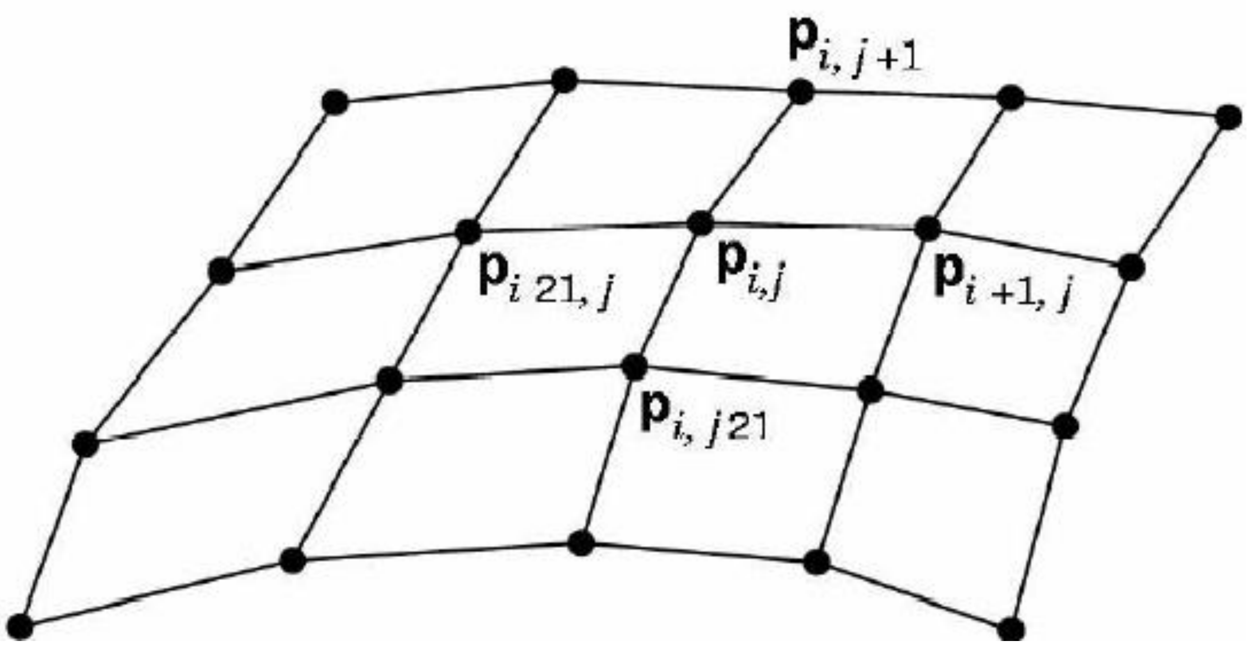
$$\mathbf{F}_\mathrm{i} = \mathbf{F}_\mathrm{i}^{\text{int}} + \mathbf{F}_\mathrm{i}^{\text{ext}}$$

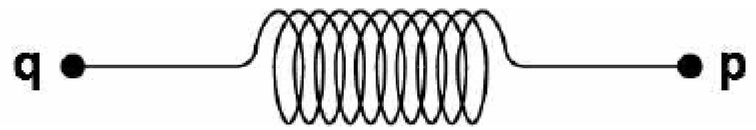
Example: Rope



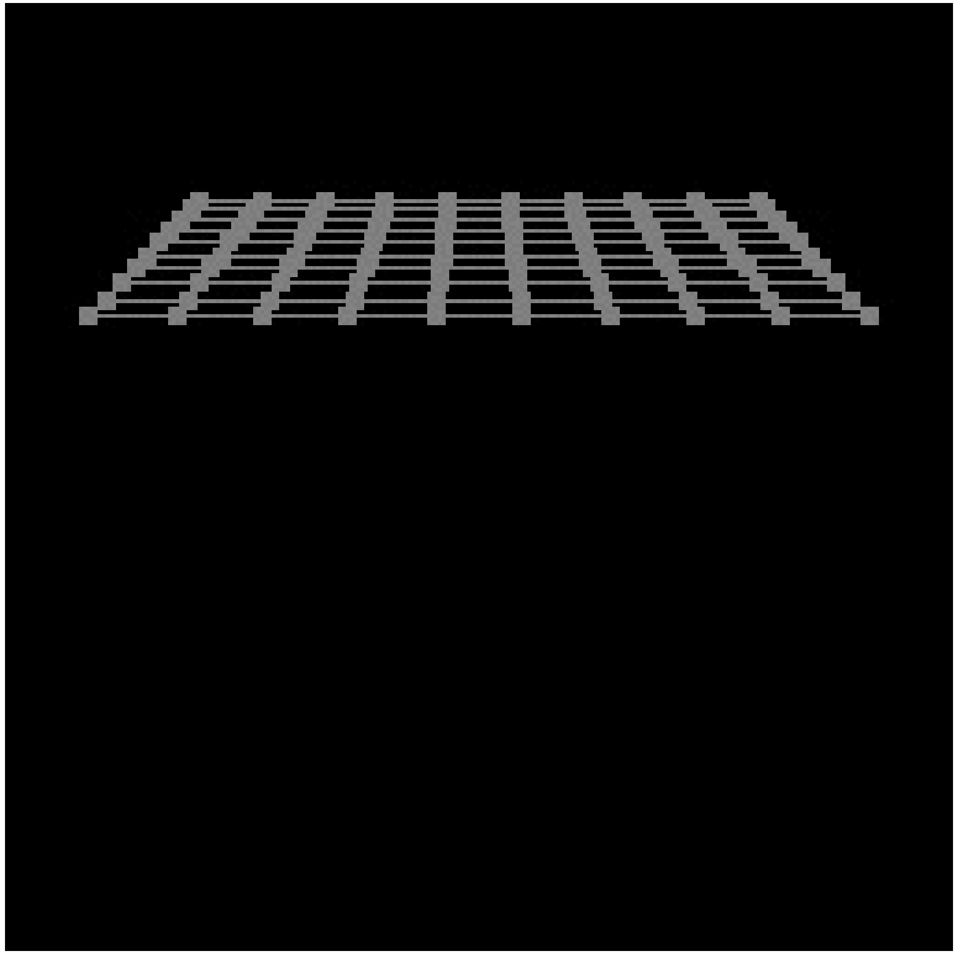
Particle System Forces

- Spring-mass mesh





Example: Cloth



Demo

Equations of Motion

- Newton's Law for a point mass
  - f = ma

- Computing particle motion requires solving second-order differential equation

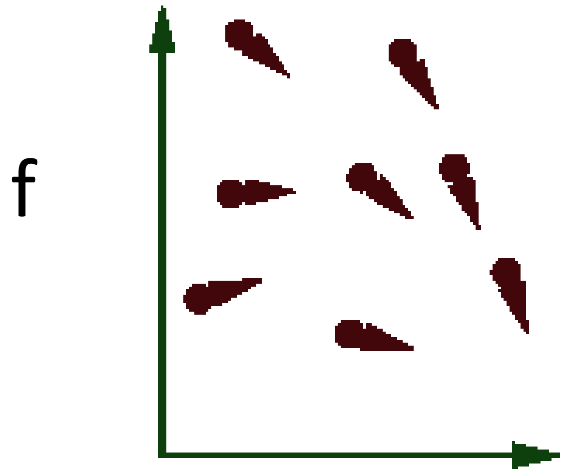
$$\ddot{x} = \frac{f(x, \dot{x}, t)}{m}$$

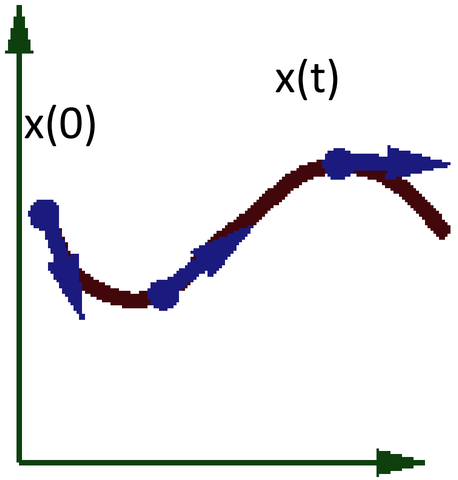
- Add variable v to form coupled first-order differential equations: "state-space form"

$$\begin{cases} \dot{x} = v \\ \dot{v} = \dfrac{f}{m} \end{cases}$$

Solving the Equations of Motion

- Initial value problem
  - Know x(0), v(0)
  - Can compute force (and therefore acceleration) for any position / velocity / time
  - Compute x(t) by forward integration





Solving the Equations of Motion

- Forward (explicit) Euler integration

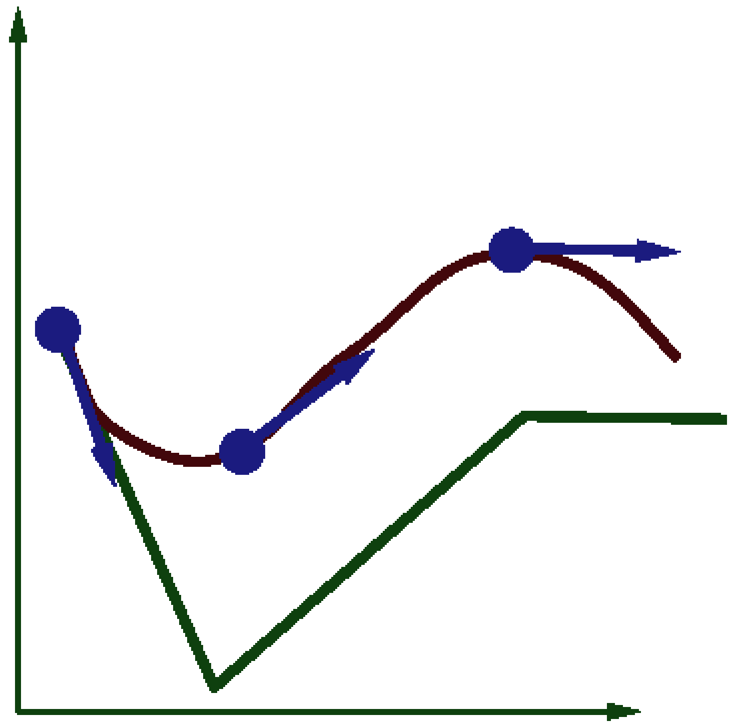
Euler Step (1768)

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

- **Idea**: start at initial condition and take a step into the direction of the tangent.
- Iteration scheme: $y_n \rightarrow f(t_n, y_n) \rightarrow y_{n+1} \rightarrow f(t_{n+1}, y_{n+1}) \rightarrow \ldots$

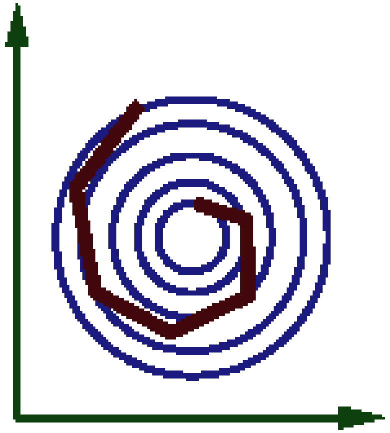
Solving the Equations of Motion

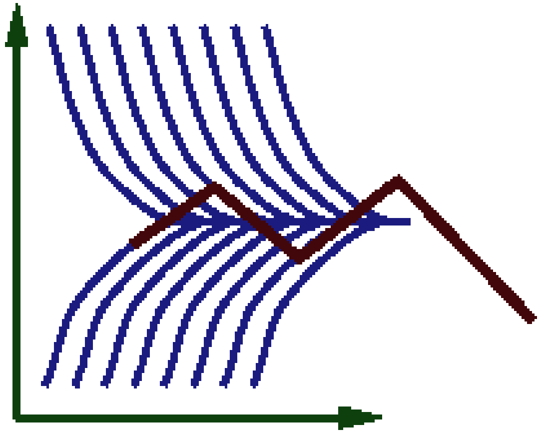
- Forward (explicit) Euler integration
  - $x(t+\Delta t) \leftarrow x(t) + \Delta t\, v(t)$
  - $v(t+\Delta t) \leftarrow v(t) + \Delta t\, f\big(x(t), v(t), t\big) / m$



Solving the Equations of Motion

- Forward (explicit) Euler integration
  - $x(t+\Delta t) \leftarrow x(t) + \Delta t\, v(t)$
  - $v(t+\Delta t) \leftarrow v(t) + \Delta t\, f\big(x(t), v(t), t\big) / m$
- Problem:
  - Accuracy decreases as Δt gets bigger

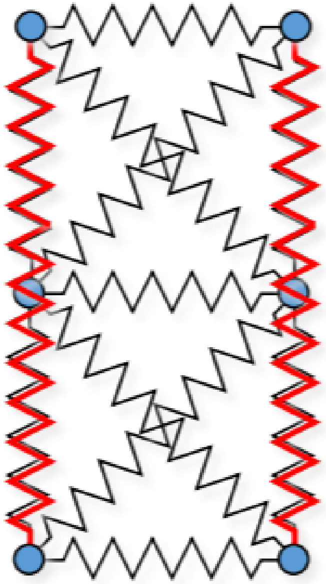




Single Particle Demo

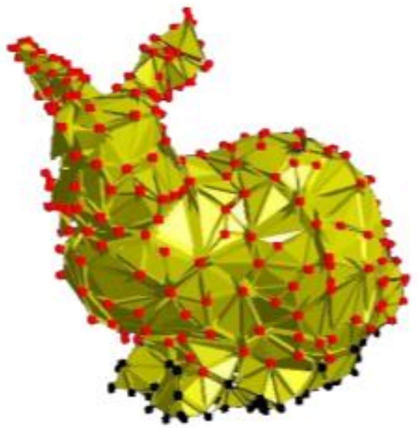
Mass Spring Systems

- Can be used to model arbitrary deformable objects, and are easy to understand and implement, but…
  - Behavior depends on mesh tessellation
    - Find good spring layout
    - Find good spring constants
    - Different types of springs interfere
  - Limited accuracy
    - No explicit volume or area preservation



Alternative

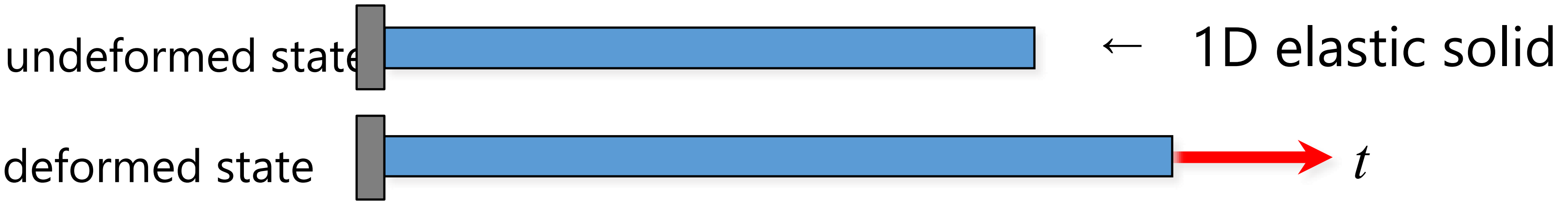
- Start from continuum mechanics principles
- Discretize with Finite Elements
  - Decompose model into simple elements
  - Setup & solve system of algebraic equations
- Advantages
  - Accurate and controllable material behavior
  - Largely independent of mesh structure



Continuum Mechanics

https://www.youtube.com/watch?v=BOabEZXm9IE

1D Continuous Elasticity

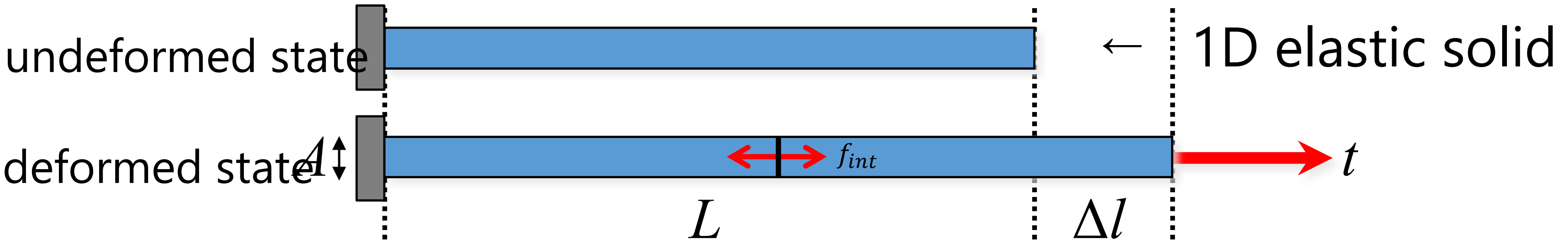


Given t, how to determine deformed configuration?

Principle of minimum potential energy

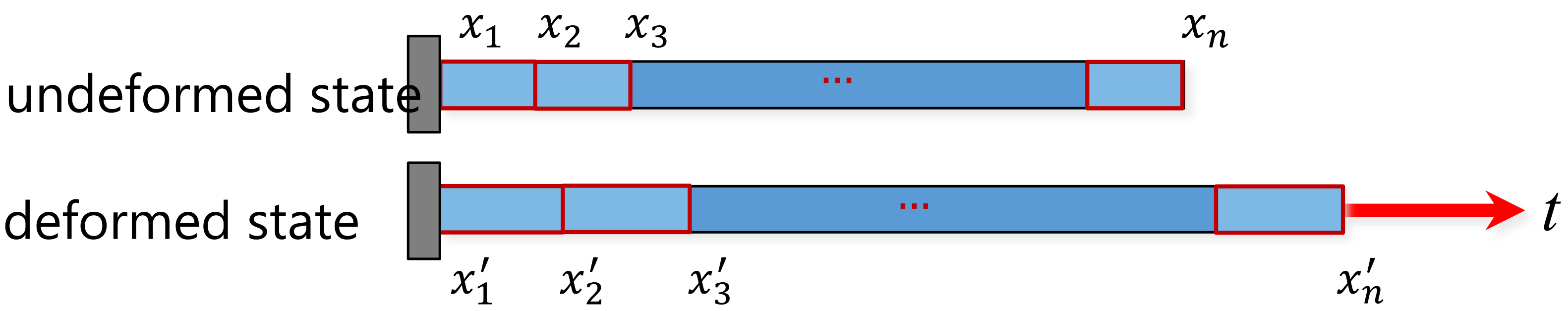
A mechanical system in static equilibrium will assume a state of minimum potential energy.

1D Continuous Elasticity



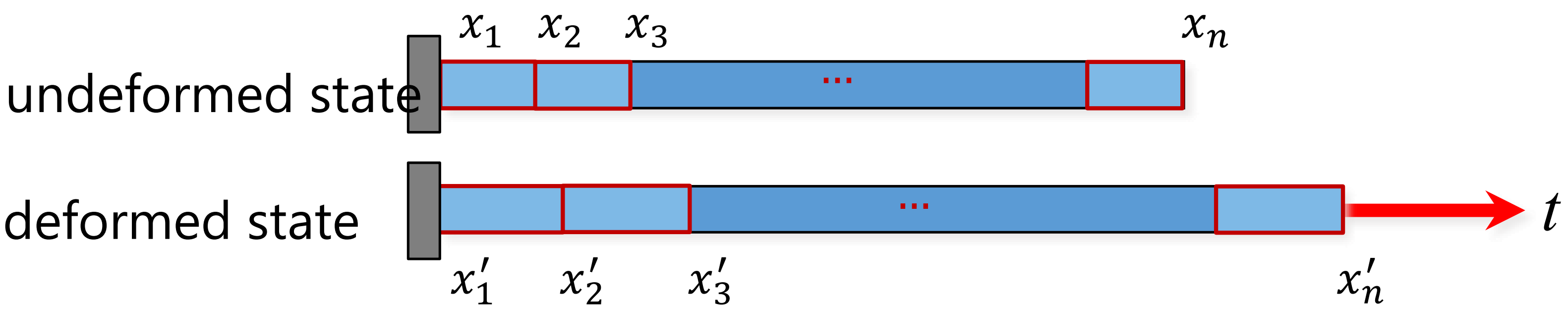
- Strain: $\varepsilon = \frac{\Delta l}{L}$ (*relative stretch*)
- Stress: $\sigma = \frac{f_{int}}{A}$ (*internal force density*)
- Hooke's law: $\sigma = k\varepsilon$ (k *material constant*)
- Strain energy density: $\Psi = \frac{1}{2}k\varepsilon^2$ (*postulate via* $\sigma = \frac{\partial\Psi}{\partial\varepsilon}$)

1D Continuous Elasticity



- Discretize domain into elements
- Element strain: $\varepsilon_i = \frac{x'_{i+1} - x'_i - L_i}{L_i}$ with $L_i = x_{i+1} - x_i$
- Element strain energy: $W_i = \Psi_i \cdot L_i = \frac{1}{2} k \varepsilon_i^2 \cdot L_i$
- Total strain energy: $W = \sum W_i$

1D Continuous Elasticity

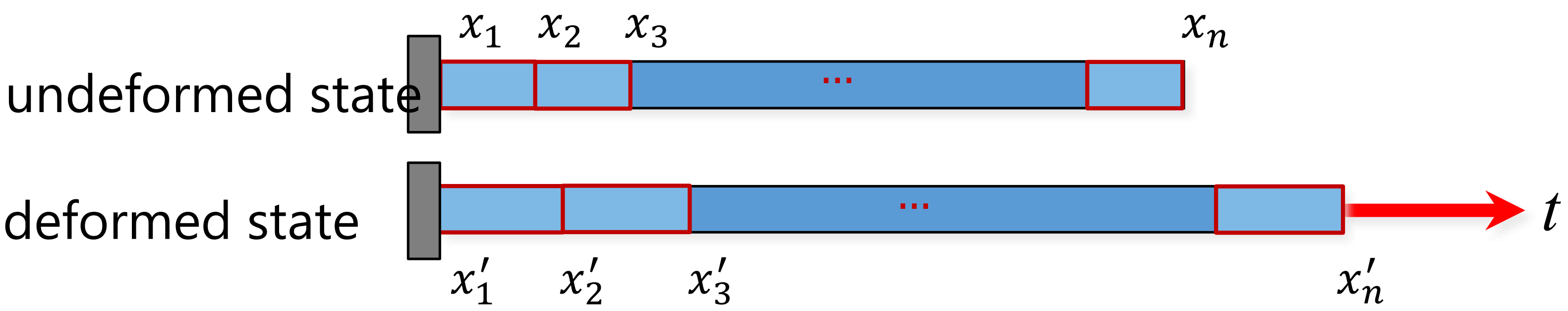


Minimum energy principle: at equilibrium

- system assumes a state of minimum total energy
- total forces vanish for all nodes

- $W_i = \frac{1}{2}k\varepsilon_i^2 \cdot L_i$ and $\varepsilon_i = \frac{x'_{i+1} - x'_i - L_i}{L_i}$
- $f_i = -\frac{\partial W}{\partial x'_i}$
- $f_1 = k\varepsilon_1$ and $f_n = -k\varepsilon_{n-1}$

1D Continuous Elasticity



Equilibrium conditions
$$f_i = \begin{cases} 0 & \forall i \in 2 \ldots n-1 \\ t & i = 1 \\ -t & i = n \end{cases}$$

→ *n-2* linear equations for *n-2* unknowns x'_i

→ solve linear system of equations to obtain deformed configuration.

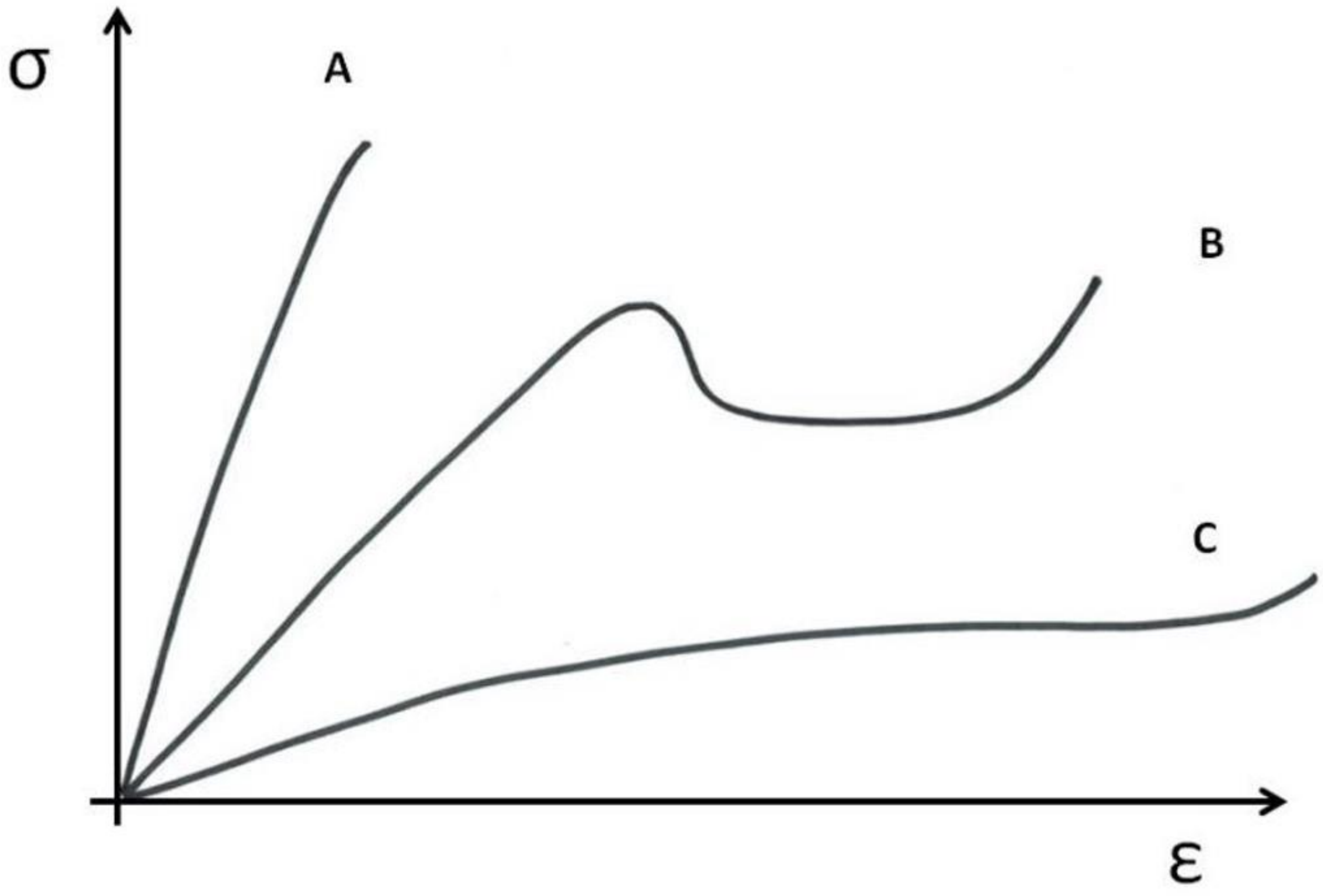
Stress Strain Curve

- What do these represent?

A: Paper

B: Fabric / unfilled plastic

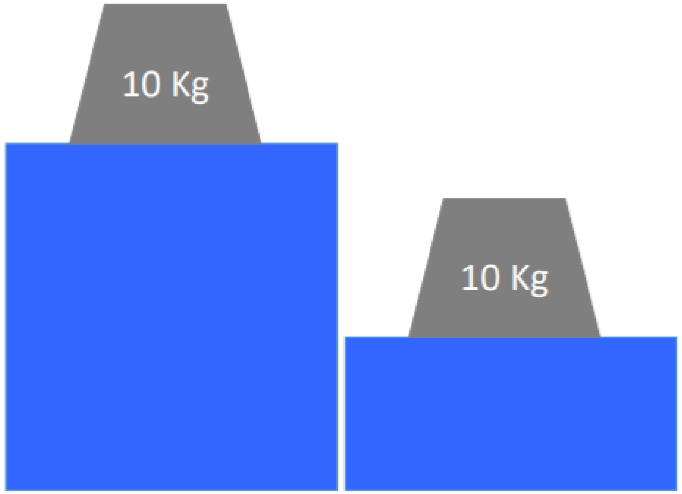
C: Rubber



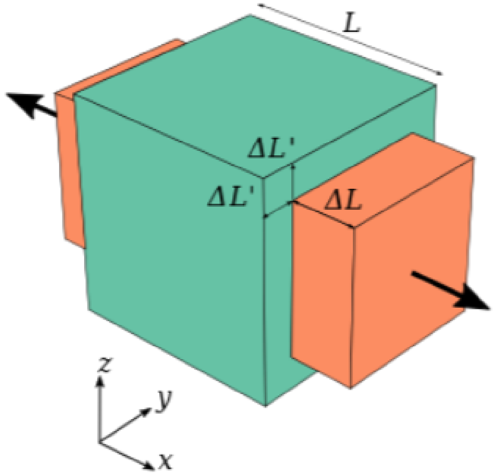
Material Models – Linear

$$\Psi = \frac{1}{2}\lambda \mathrm{tr}(\varepsilon)^2 + \mu tr(\varepsilon^2)$$

Lame parameters λ and μ are material constants related to the fundamental physical parameters: Poisson's Ratio and Young's modulus (http://en.wikipedia.org/wiki/Lamé_parameters)



Young's modulus (E), measure of stiffness

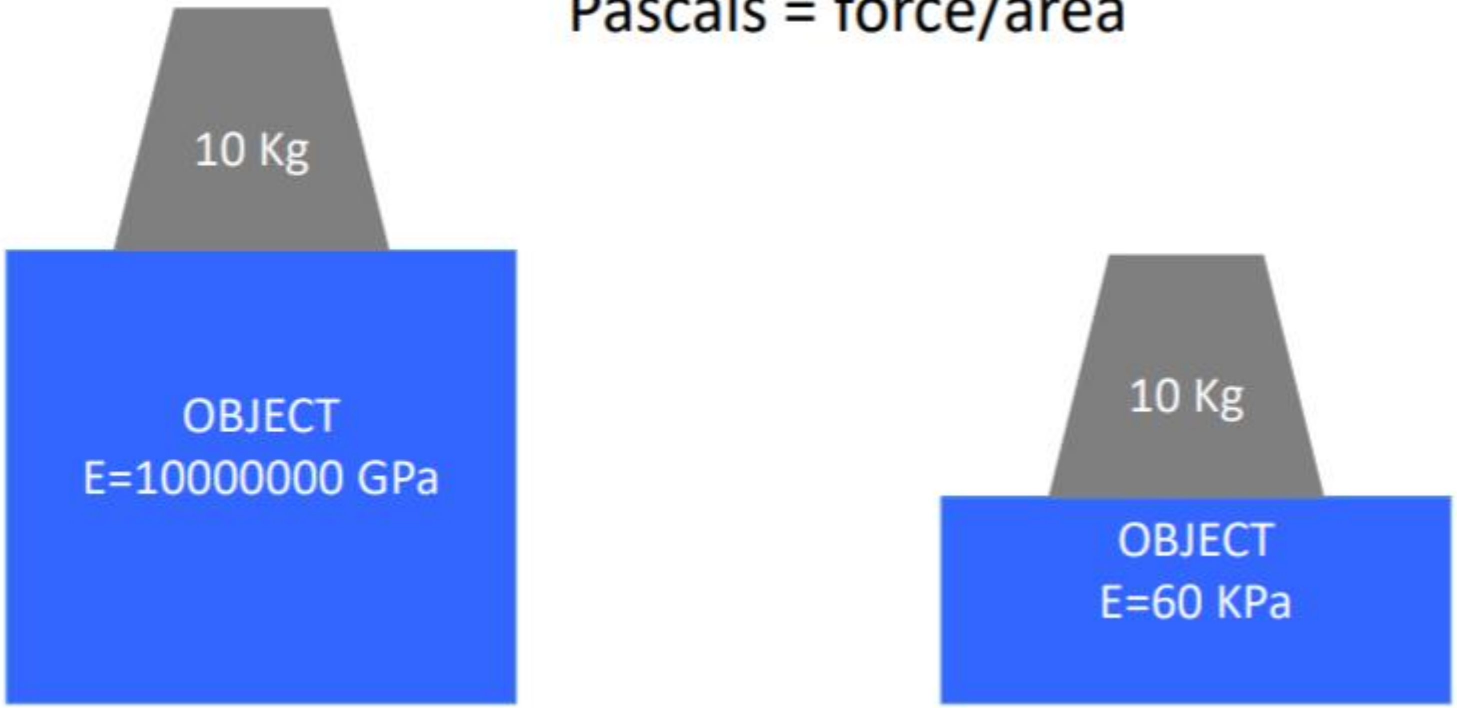


Poisson's ratio (v), captures transverse deformation relative to axial deformation

Material Models – Linear

- Stiffness is pretty intuitive

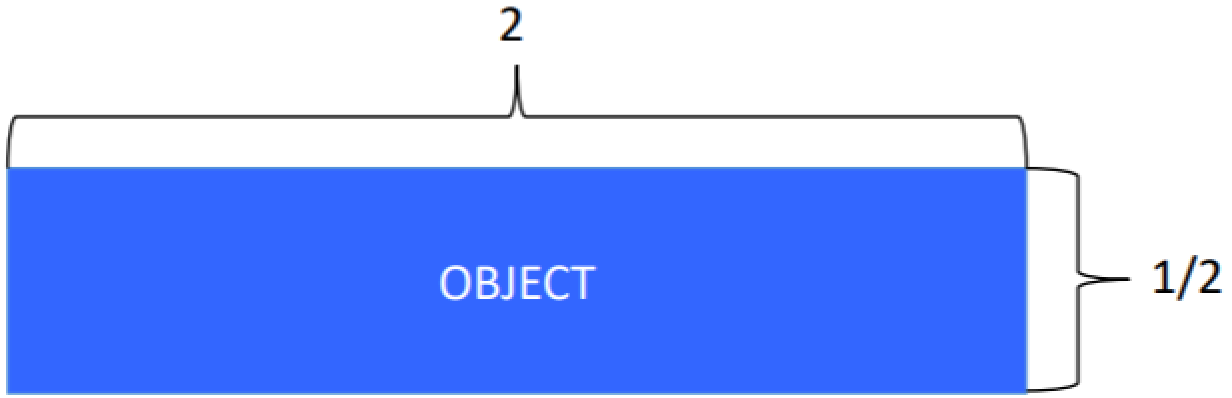
Pascals = force/area



Material Models – Linear

Poisson's Ratio controls volume preservation

$$\mu = -\frac{\Delta y}{\Delta x} = 0.5$$



Negative Poisson's Ratio

https://www.youtube.com/watch?v=5wpRszZZhYQ

Nonlinear Elasticity

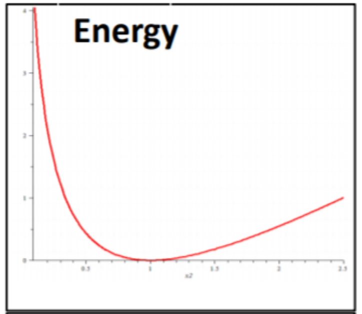
- Idea: replace Cauchy strain with Green strain

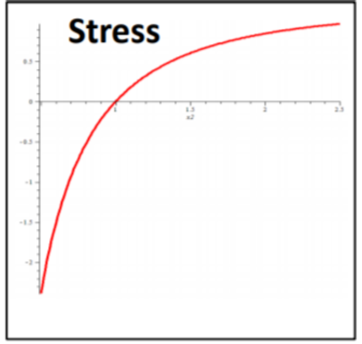
  $\rightarrow$ *St. Venant-Kirchhoff material* (StVK)
- Energy $\Psi_{StVK} = \frac{1}{2}\lambda \mathrm{tr}(\mathbf{E})^2 + \mu \mathrm{tr}(\mathbf{E}^2)$
- Component l of force on node k is $\mathbf{f}_{kl}^e = -\frac{\partial W^e}{\partial \mathbf{x}_k} = -\sum_{ij} \frac{\partial W^e}{\partial \mathbf{F}_{ij}^e} \frac{\partial \mathbf{F}_{ij}^e}{\partial \mathbf{x}_{kl}}$
- Note:
  - Energy is quartic in $\mathbf{x}$, forces are cubic
  - Solve system of nonlinear equations

Nonlinear Elasticity

- Green Strain $\mathbf{E} = \frac{1}{2}(\mathbf{F}^t\mathbf{F} - \mathbf{I}) = \frac{1}{2}(\mathbf{C} - \mathbf{I})$
- Split into *deviatoric (i.e. volume-preserving shape changing/distortion)* and *volumetric (dilation, volume changing)* deformations

  Volumetric: $J = \det(\mathbf{F})$ Deviatoric: $\mathbf{C} = \mathbf{F}^t\mathbf{F}$





- *Compressible* Neo-Hookean material:

$$\Psi_{NH} = \frac{\mu}{2}(\mathrm{tr}(\mathbf{C}) - 3) - \mu \ln J + \frac{\lambda}{2}\ln(J)^2$$

Observations:

- the first term penalizes all deformations equally (since $\mathrm{tr}(\boldsymbol{C}) = |\boldsymbol{F}|_F^2$)
- the third term goes to infinity for increasing compression (*faster than the second*)
- the stress-strain behavior is initially linear, but goes into plateau for larger deformations
- Rule of thumb: NH is good for deformations of up to 20%

Limits

- Real-world materials are not perfectly (hyper)elastic
  - Viscosity (*stress relaxation, creep*)
  - Plasticity (*irreversible deformation*)
  - Mullins effect (*stiffness depends on strain history*)
  - Fatigue, damage, …

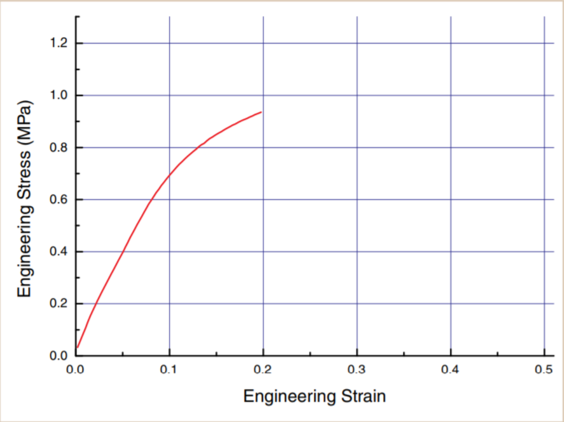


Figure 11, 1st Loading of a Thermoplastic Elastomer

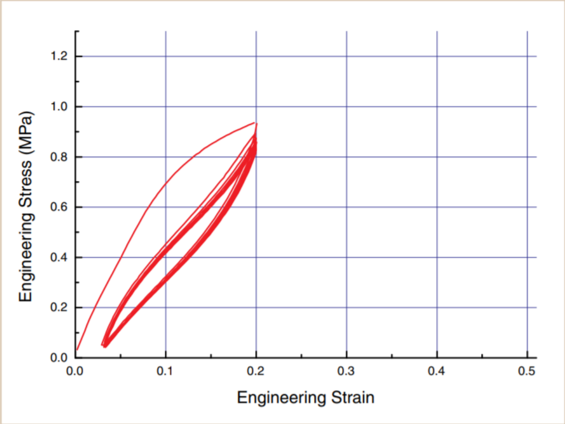


Figure 12, Multiple Strain Cycles of a Thermoplastic Elastomer

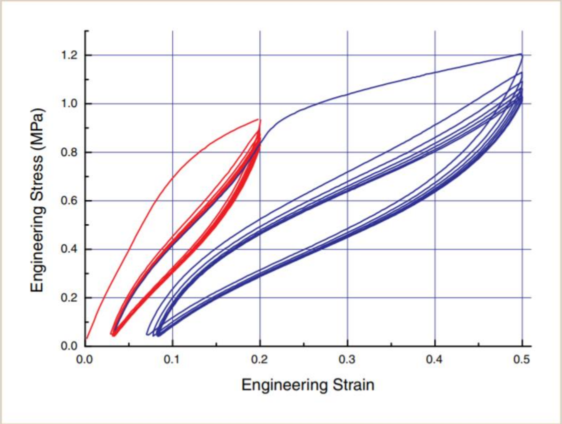


Figure 13, Multiple Strain Cycles of a Thermoplastic Elastomer at 2 Maximum Strain Levels

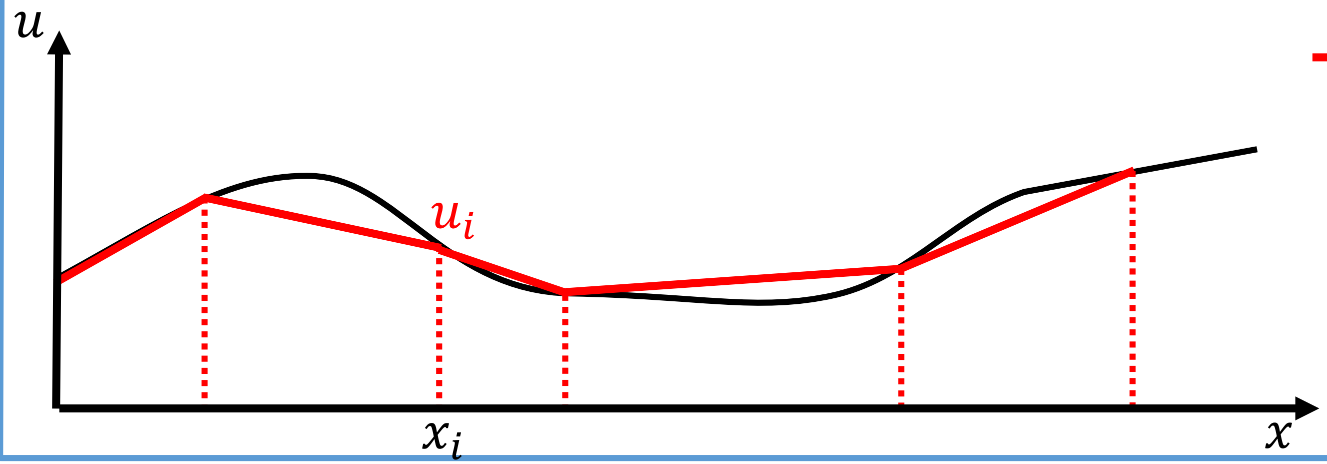
Finite Elements

What is a finite element?

A finite element is a triplet consisting of

- a closed subset $\Omega_e \subset \boldsymbol{R}^d$ (*in d dimensions*)
- n vectors of nodal variables $\bar{\boldsymbol{x}}_i \in \boldsymbol{R}^d$ describing the reference geometry
- n nodal basis functions, $N_i\colon \Omega_e \to \boldsymbol{R}$

$\to n$ vectors of degrees of freedom (e.g., deformed positions $\boldsymbol{x}_i$)



$$= u_{i+1} \cdot \frac{x - x_i}{x_{i+1} - x_i} + u_i \cdot \frac{x_{i+1} - x}{x_{i+1} - x_i}$$

$$x \in [x_i, x_{i+1}), i = 0 \ldots n-1$$

$$= \sum_i u_i \cdot N_i(x)$$

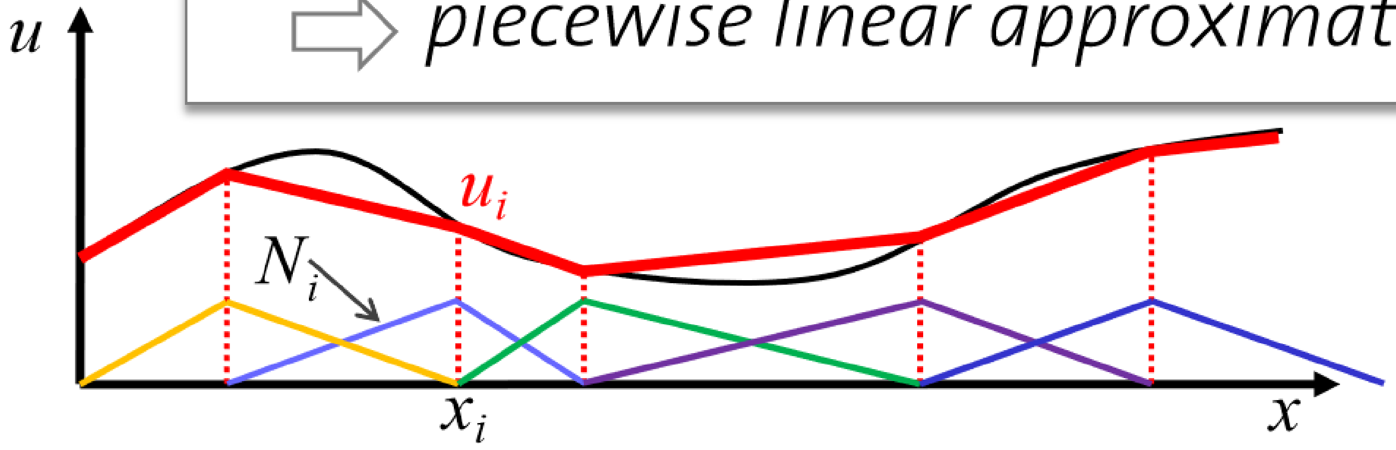
FEM – 1D – Basis Functions

What basis functions should be used in $u(x) = \sum_i u_i N_i(x)$

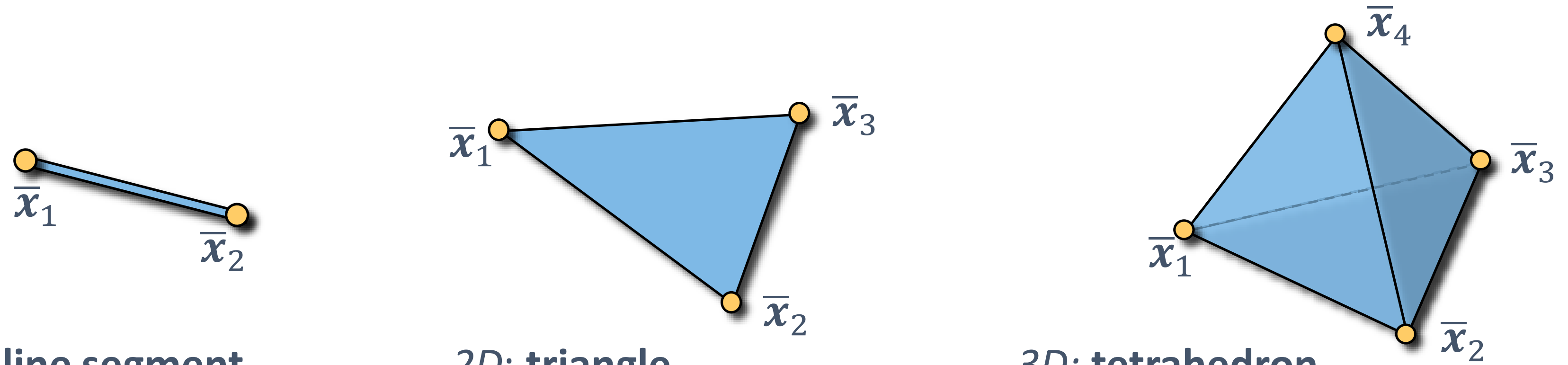
- Smooth enough -> *once differentiable*
- Simple -> *polynomial functions*
- Interpolation -> $N_i(\mathbf{x}_j) = \delta_{ij}$
- Compact support -> *defined piecewise on simple geometry*

Use piecewise linear basis functions

⇨ *piecewise linear approximation* $u(x)$



Linear Simplicial Elements



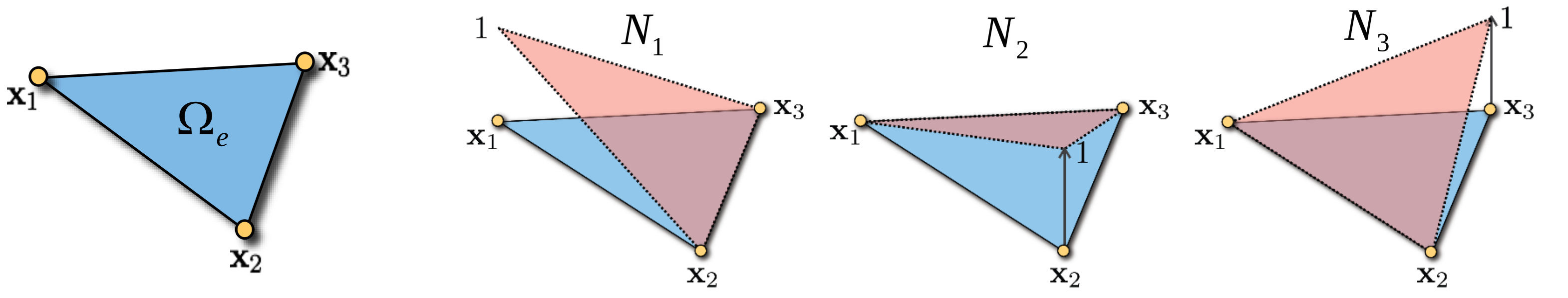
1D: **line segment** *2D*: **triangle** *3D:* **tetrahedron**

- Simplicial elements admit linear basis functions
- Basis functions are uniquely defined through
- reference geometry $\bar{\boldsymbol{x}}_i$ and
- interpolation requirement $N_i(\bar{\boldsymbol{x}}_j) = \delta_{ij}$

$\bar{\boldsymbol{x}}_i = \bar{x}_i$	in 1D
$\bar{\boldsymbol{x}}_i = (\bar{x}_i, \bar{y}_i)$	in 2D
$\bar{\boldsymbol{x}}_i = (\bar{x}_i, \bar{y}_i, \bar{z}_i)$	in 3D

Computing Basis Functions – 2D

Example: 3-node elements with linear basis functions

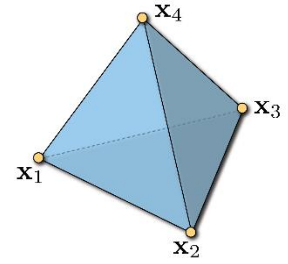


- Basis functions are linear: $N_i(x,y) = a_i x + b_i y + c$
- Due to $N_i(\mathbf{x}_j) = \delta_{ij}$, we have

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} = \begin{bmatrix} \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \end{bmatrix} \Rightarrow \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \end{bmatrix}$$

Computing Basis Functions – 3D

4-node tetrahedron with 4 linear basis functions

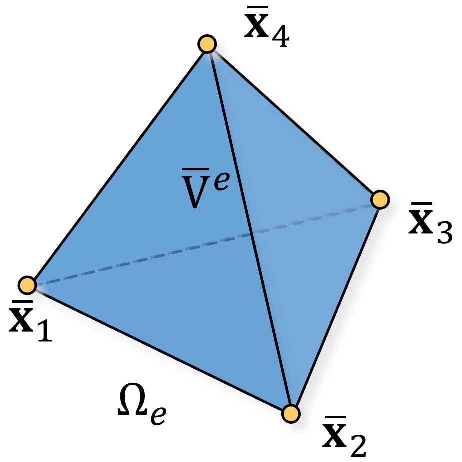


… … … …

- Basis functions are linear, $N_i(\bar{x}, \bar{y}, \bar{z}) = a_i\bar{x} + b_i\bar{y} + c_i\bar{z} + d_i$
- From $N_i(\bar{\boldsymbol{x}}_j) = \delta_{ij}$ we obtain

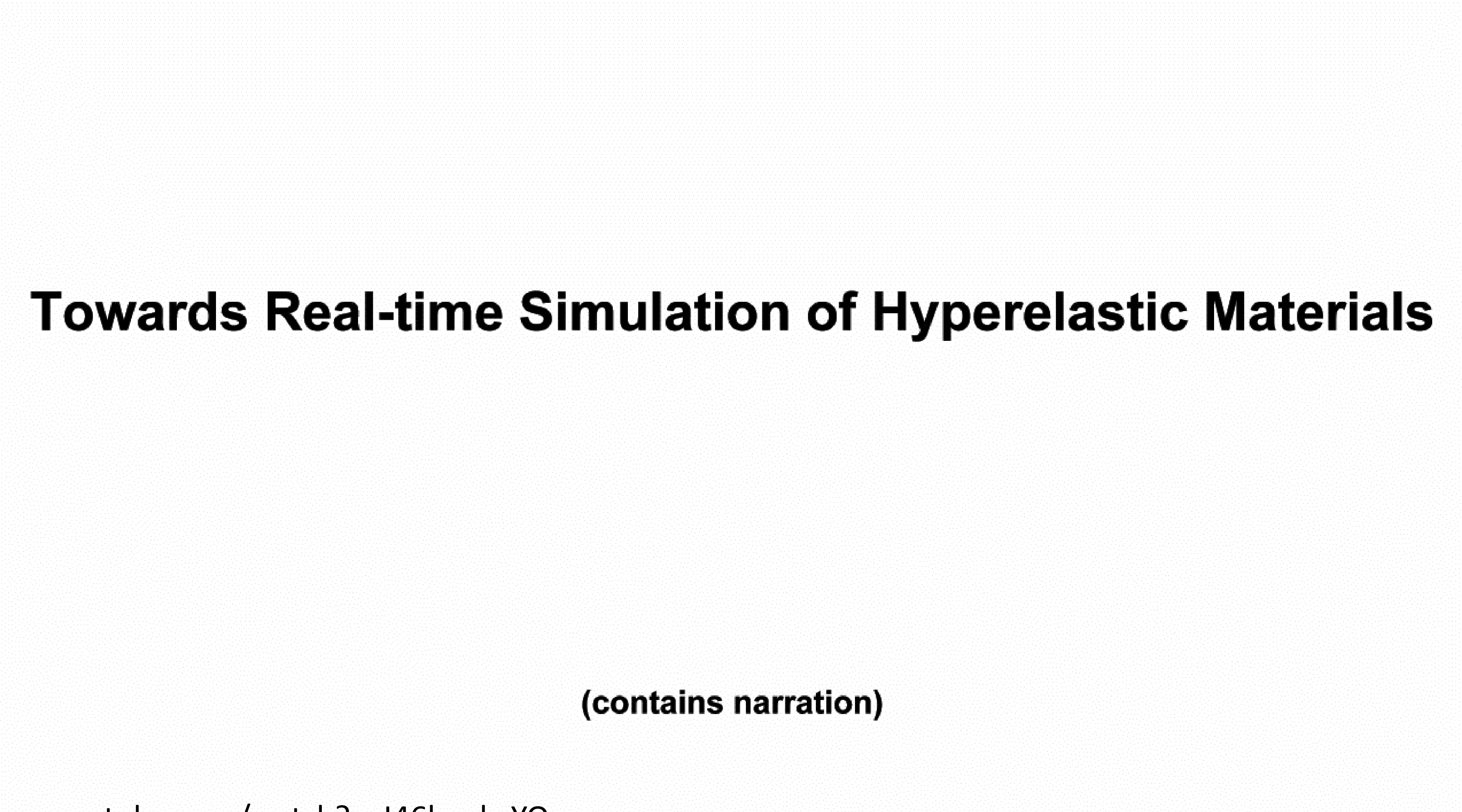
$$N_i(\bar{x}, \bar{y}, \bar{z}) = a_i\bar{x} + b_i\bar{y} + c_i\bar{z} + d_i$$

$$\begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{pmatrix} \begin{pmatrix} a_i \\ b_i \\ c_i \\ d_i \end{pmatrix} = \begin{pmatrix} \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ \delta_{4i} \end{pmatrix}$$



110

Example



https://www.youtube.com/watch?v=I46ly-ubzYQ

Example

Stable Neo-Hookean Flesh Simulation

Breannan Smith	Pixar Animation Studios
Fernando de Goes	Pixar Animation Studios
Theodore Kim	Pixar Animation Studios

https://vimeo.com/245424174