Computer Animation and Games I

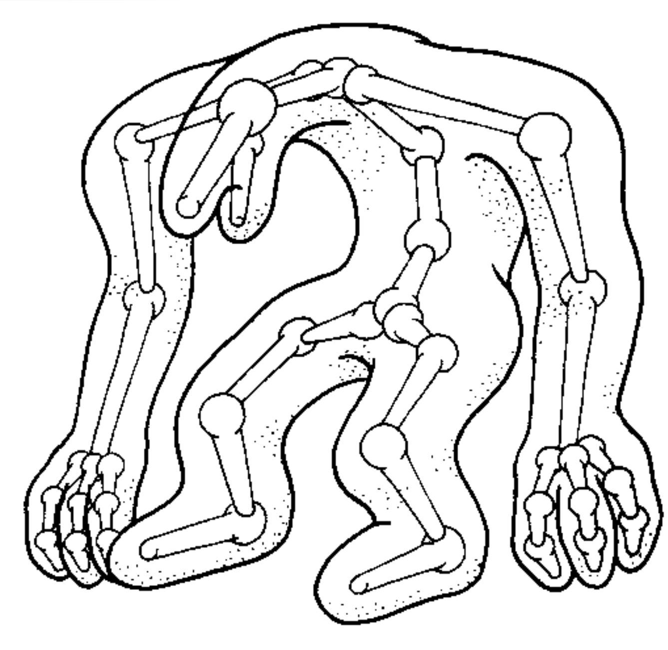
CM50244

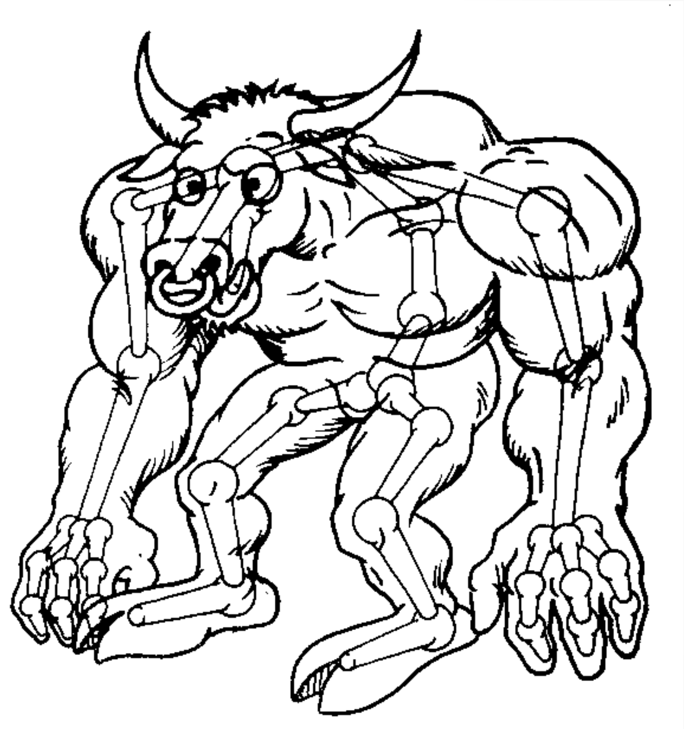
Skeleton-based Animation

Some slides from Charles Owen @ MSU

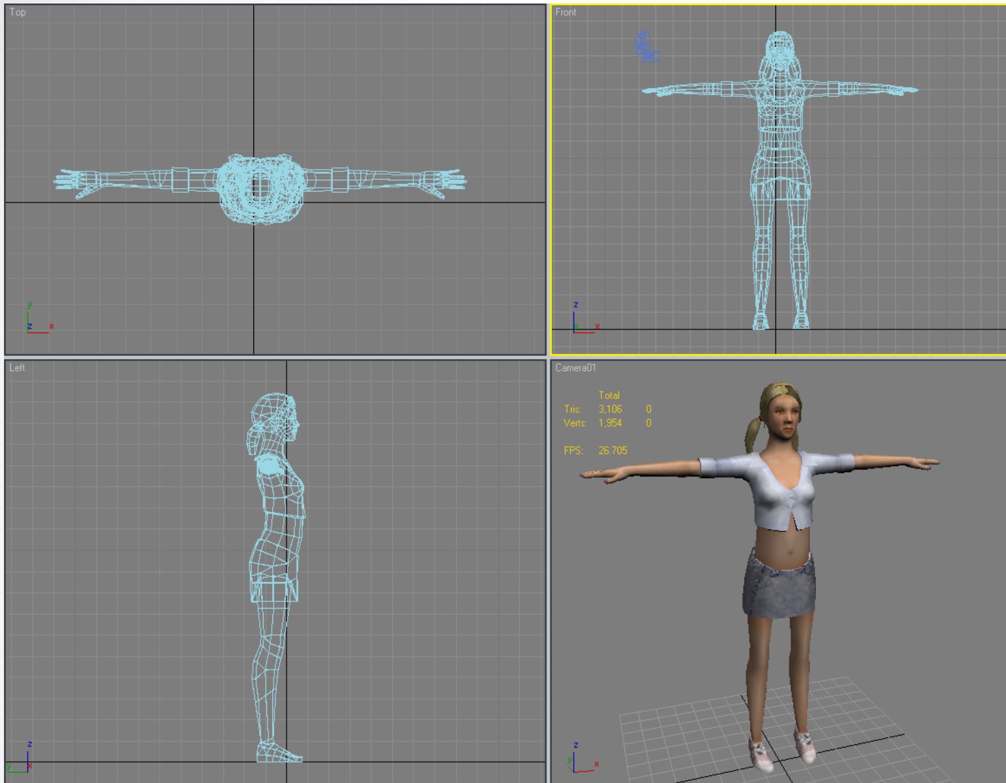
Recall: Skeleton-based Rigging

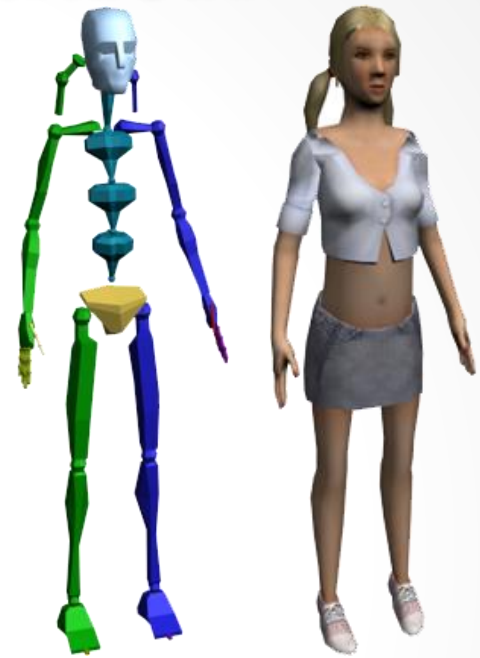
- Skin on top of the skeleton



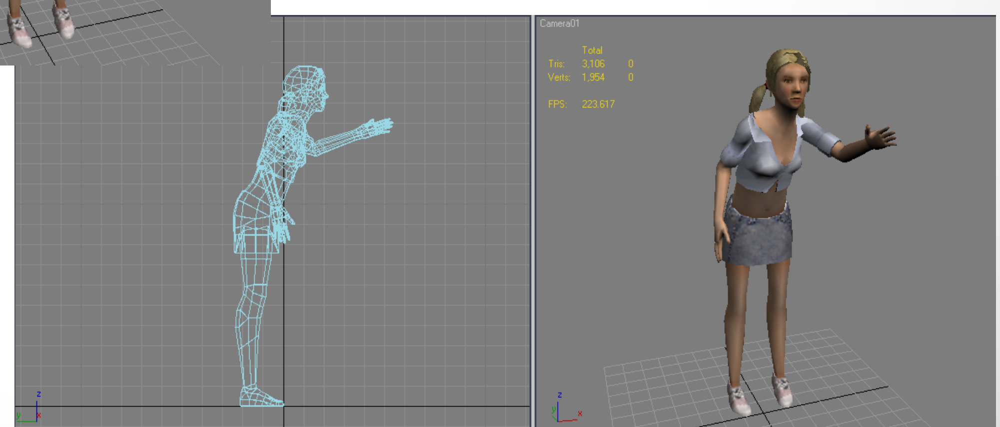


Skeletal Animation





Victoria



Overview

- Skeletons and Poses
- Skeleton-based Mesh Deformation

Overview

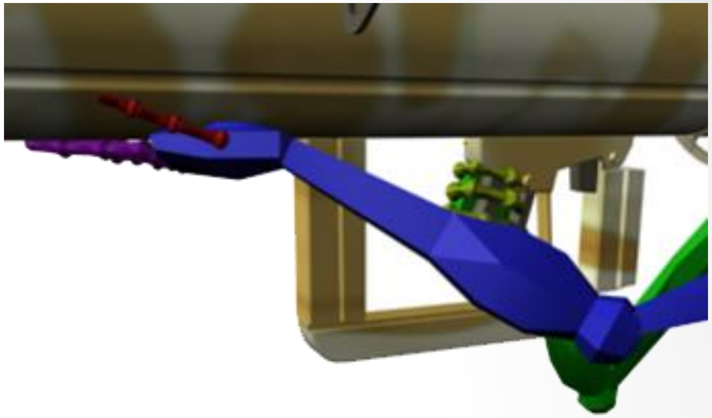
- **Skeletons and Poses**
- Skeleton-based Mesh Deformation

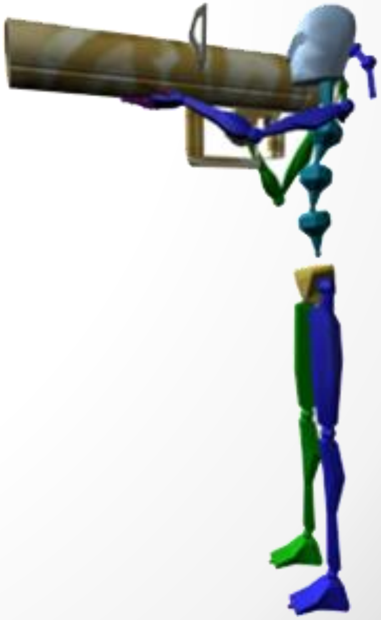
Skeletons

- Skeleton
  - A pose-able framework of **joints** and **bones** arranged in a tree structure.
  - An invisible structure to manipulate the skin and other geometric data of the character.
  - Does not actually render.

Skeletons

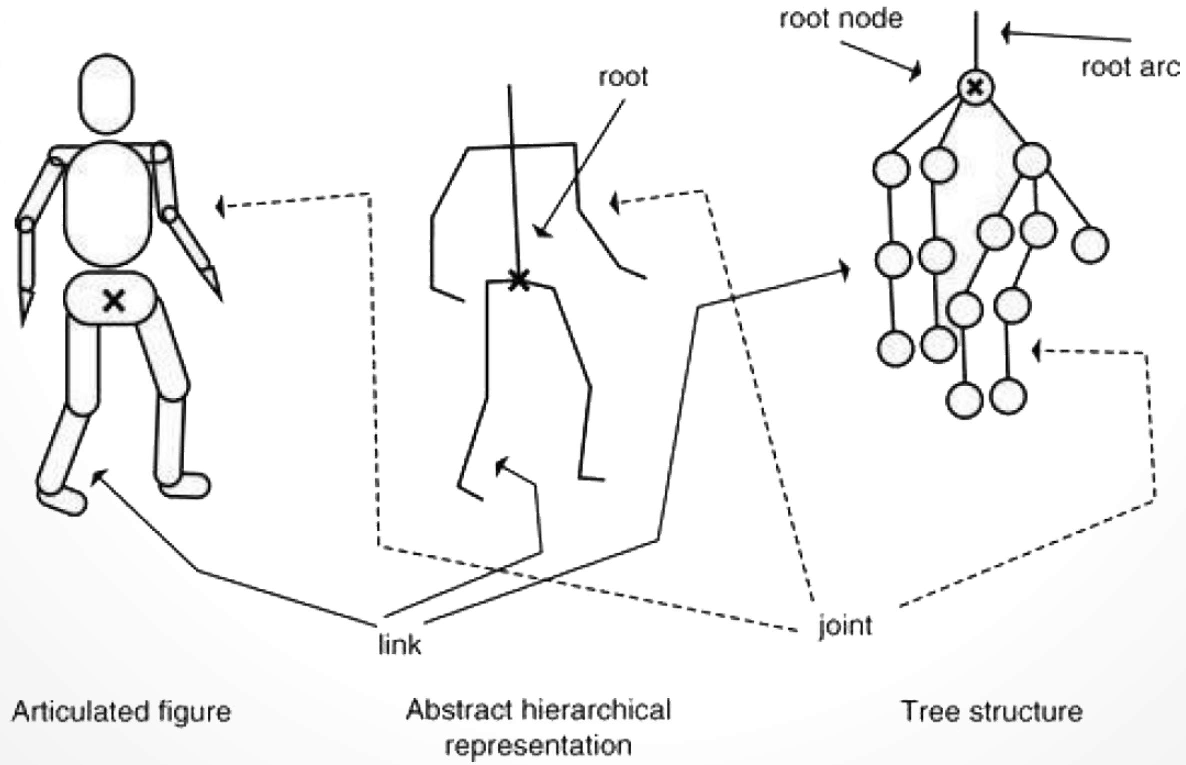
- Bones (links)
  - Correspond to body parts
- Joints
  - Allows relative movement between parts within the skeleton. (equivalent to 4x4 matrix transformations)





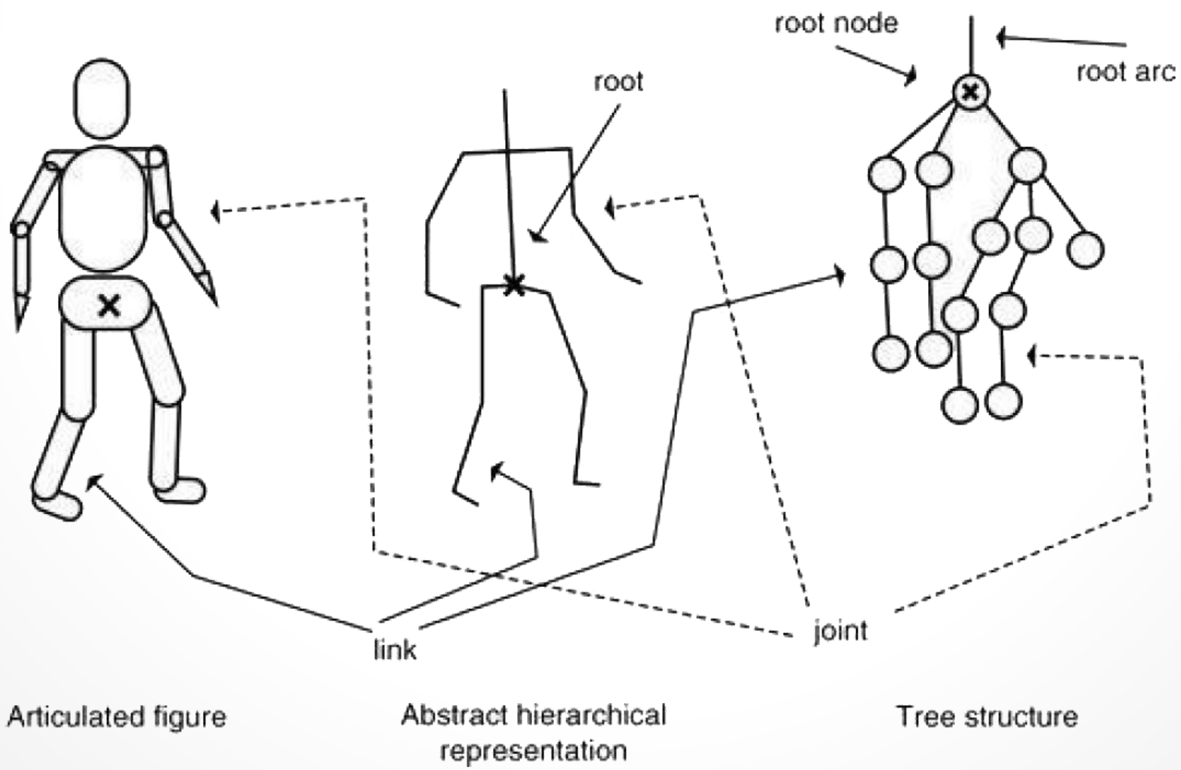
Hierarchical Tree Structure

- The bones and joints are organized in a hierarchical tree structure
  - nodes represent bones (links)
  - edges represent joints

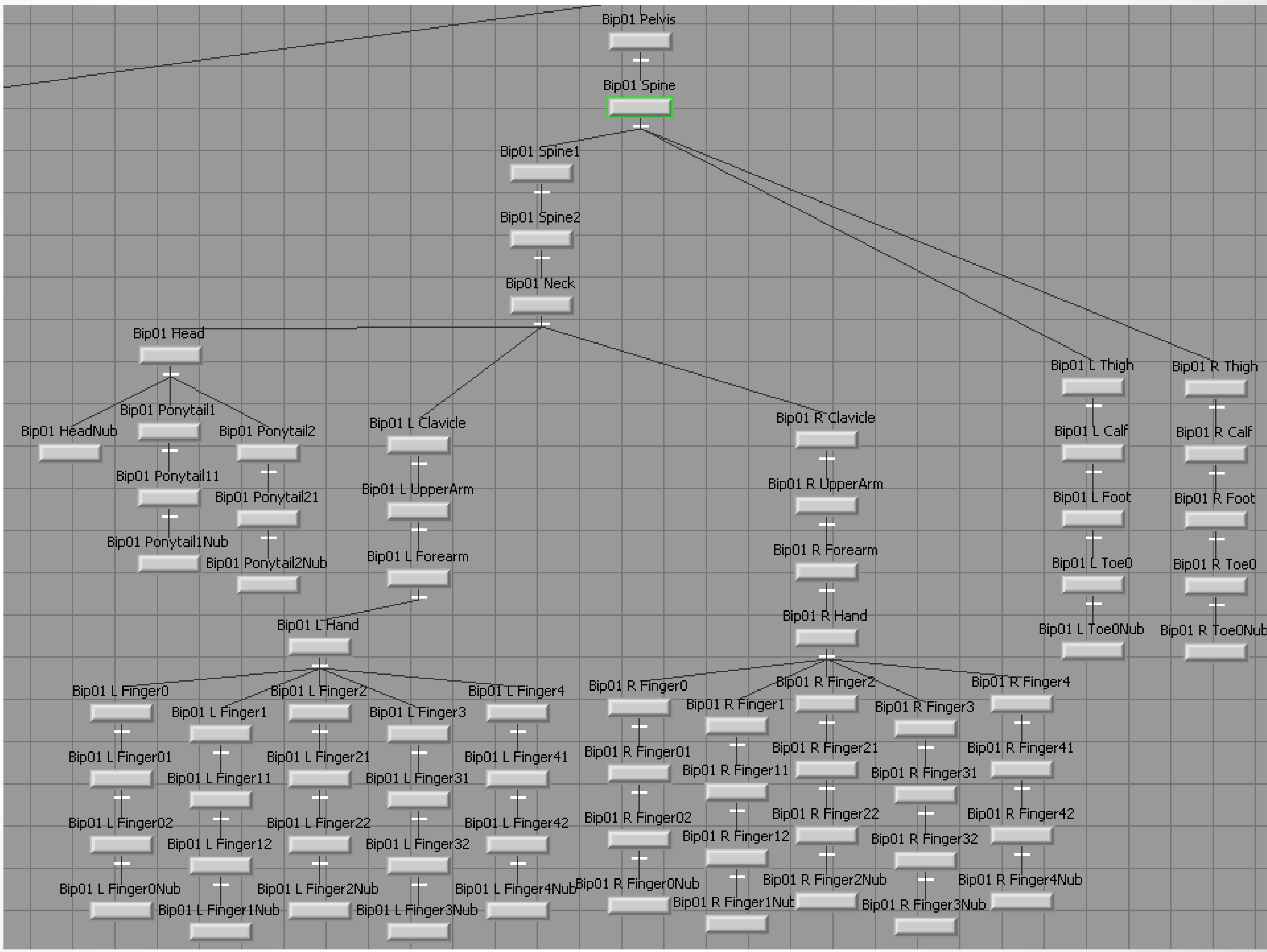


Hierarchical Tree Structure

- The bones and joints are organized in a hierarchical tree structure
  - joint movement will affect all the child nodes below it

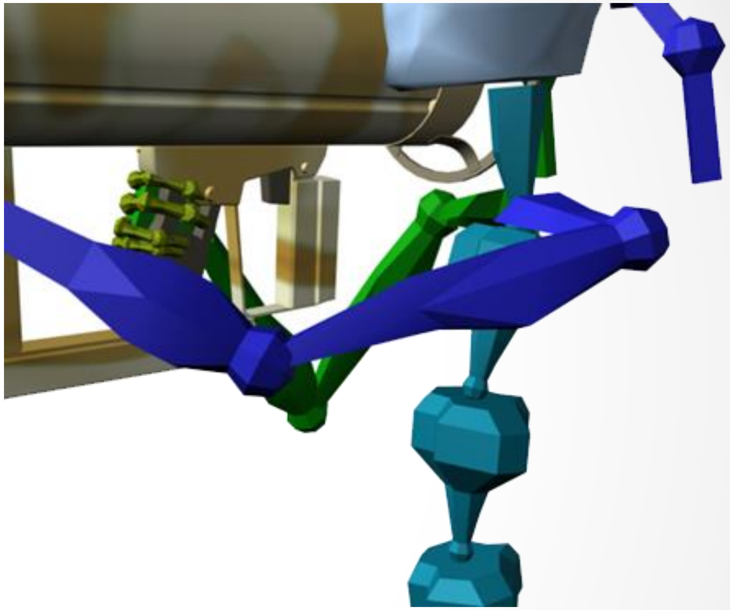


Victoria in MotionBuilder



DOFs

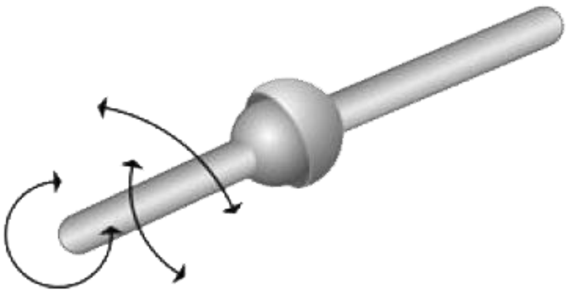
- Degree of Freedom (DOF)
  - A variable φ describes one dimension of movement within a joint
  - Joints typically have around 1-6 DOFs ($\varphi_1 \ldots \varphi_N$) Can have more (up to 9 for affine – without translation)
  - Changing the DOF values will generate different poses



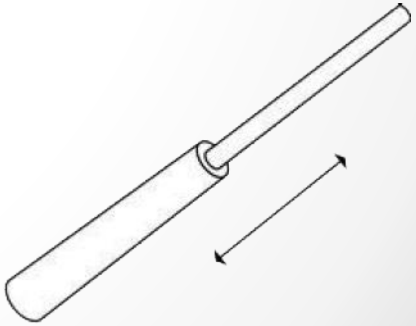
Rigid body transformations (rotation+translation): 6DOF
Arbitrary rotations: 3DOF

Joint Types

- Rotational
  - Hinge: 1-DOF
  - Universal: 2-DOF
    - Around 2 axis
  - Ball & Socket: 3-DOF
    - Euler Angles
    - Quaternions
- Translational
  - Prismatic: 1-DOF
  - Translational: 3-DOF (or any number)
- Non-Rigid
  - Scale
  - Shear
  - Etc.
- Design your own...



Ball-and-socket joint



Prismatic joint

Forward Kinematics

- Each joint computes a local matrix **M** based on the DOFs:

$$\text{Local matrix } \mathbf{M} = \mathbf{M}_{\text{joint}}(\varphi_1, \varphi_2, \ldots, \varphi_N)$$

- Then, world matrix **W** is computed by combing **M** with the world matrix of the parent joint $\mathbf{W}_{\text{parent}}$ (local-global conversion)

$$\text{World matrix } \mathbf{W} = \mathbf{W}_{\text{parent}}\mathbf{M}$$

- The recursive process iterates until the root joint is reached

Skeleton Posing Process

1. Specify DOF values for the skeleton (joints)

2. Traverse the hierarchy using **forward kinematics** to compute the world transformation matrices of individual skeleton bones

3. Use world matrices of the bones to deform skin & render

Poses

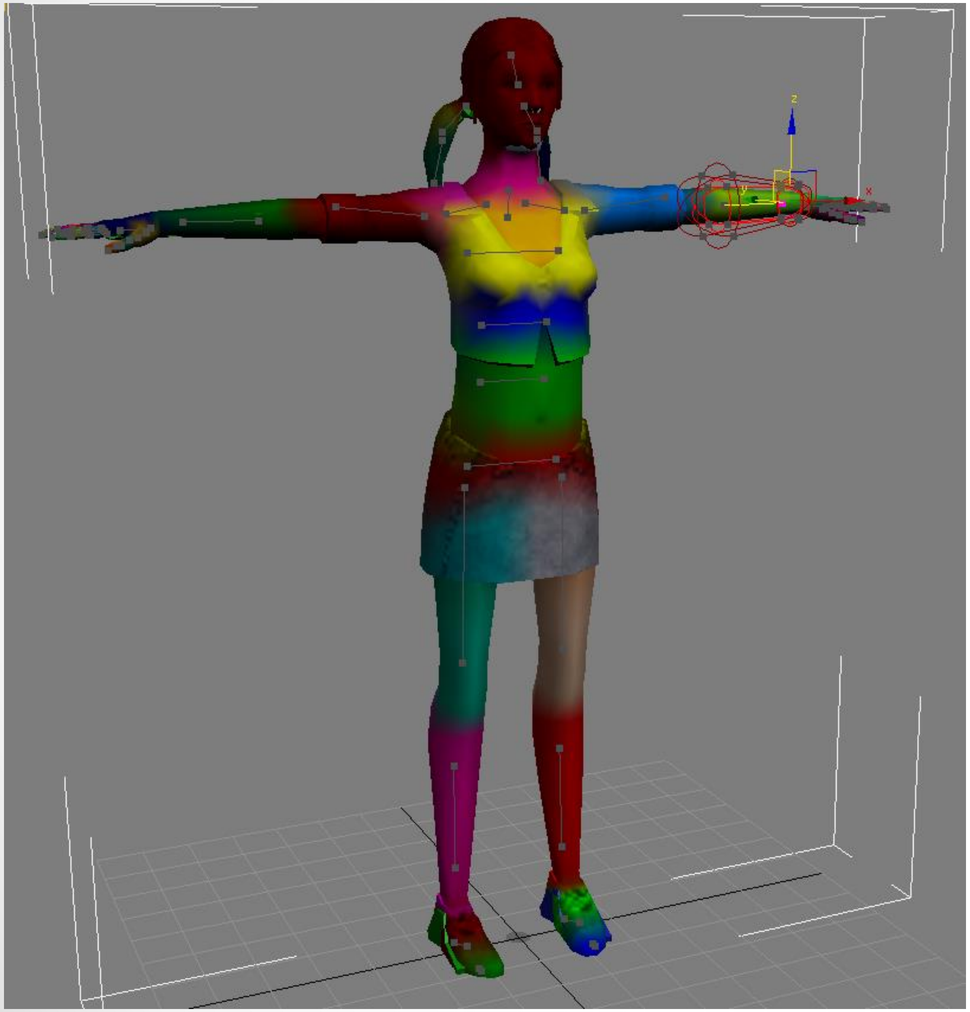
- Adjust DOFs to specify the pose of the skeleton
- We can define a pose Φ by setting up a vector of N DOFs in the skeleton

$$\Phi = [\varphi_1\ \varphi_2\ \dots\ \varphi_N]$$

Overview

- Skeletons and Poses
- **Skeleton-based Mesh Deformation**

Smooth Skin Algorithm

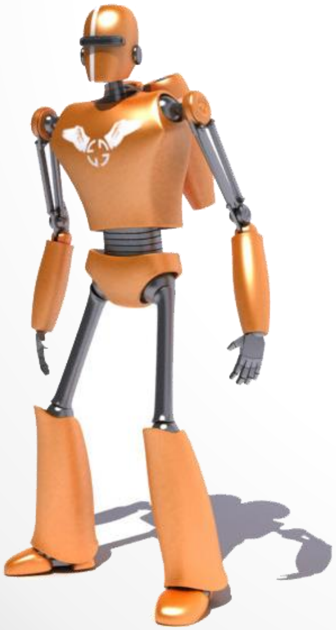


Rigid Parts are Easy

- Robots and mechanical creatures
    - Rigid parts, no smooth skin

- Every part (vertex) of the character's geometry is transformed by exactly one (global) matrix of a single bone

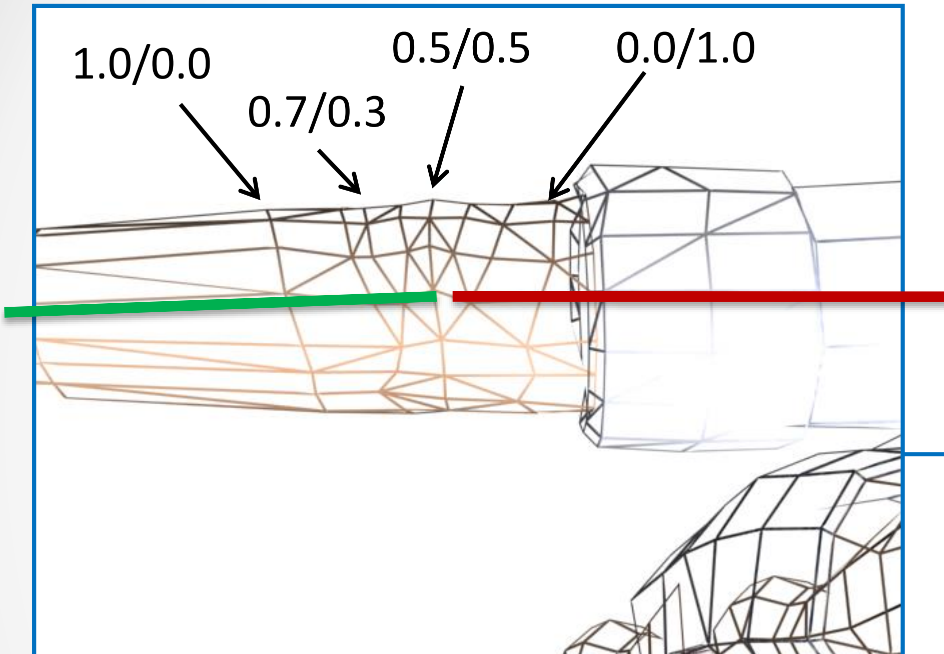
$$\mathbf{v}' = M\mathbf{v}$$

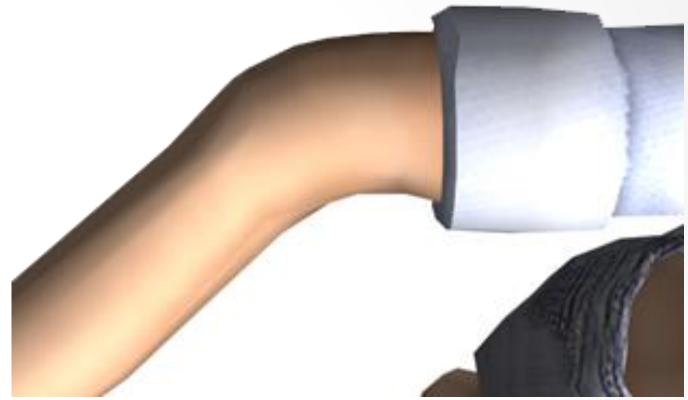


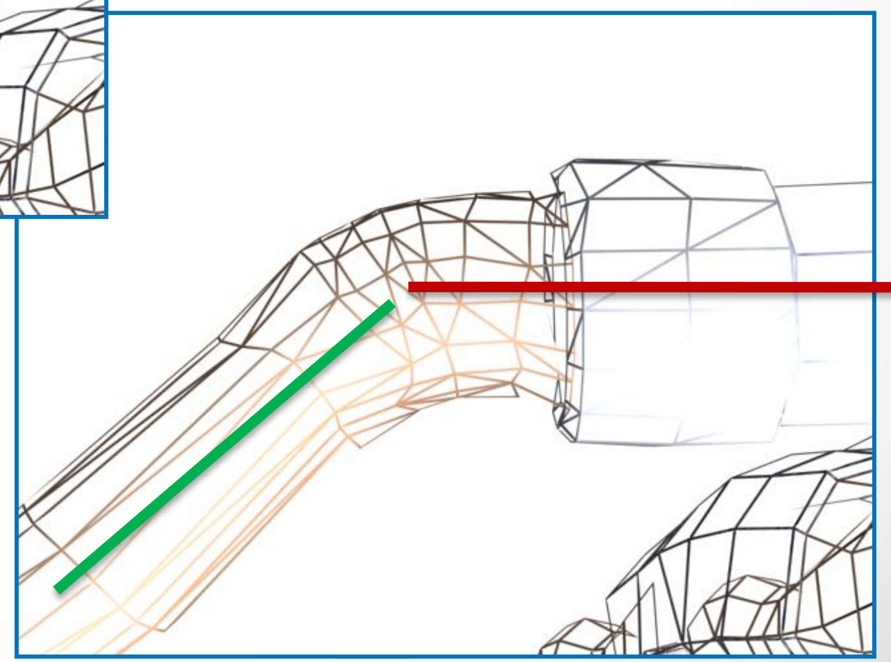
What happens with Skinned Humans?

The mesh is deformed by the bones, but not "rigidly". Instead, it is a flexible bend.

The Basic Concept







Each vertex can be moved by 1-4 bones, with each bone having a **weight**.

Mathematics of Mesh Skinning

- Each vertex is multiplied by several "weighted" transformation matrices and the results are added together.

$$\mathrm{v}' = \sum_{i}^{n} w_i M_i \mathrm{v} \quad \text{with} \quad \sum_{i} w_i = 1$$

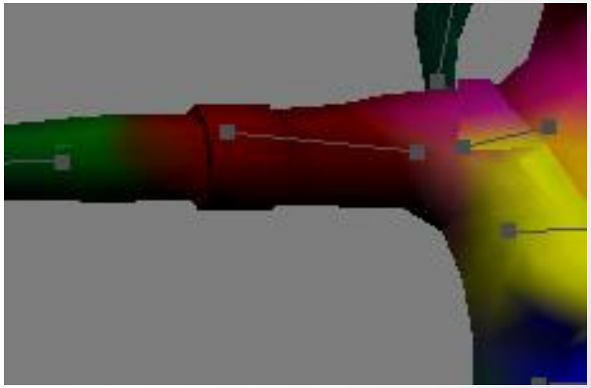
Where:

n is the number of bones.

v is the vertex position.

w_i is the weight associated.

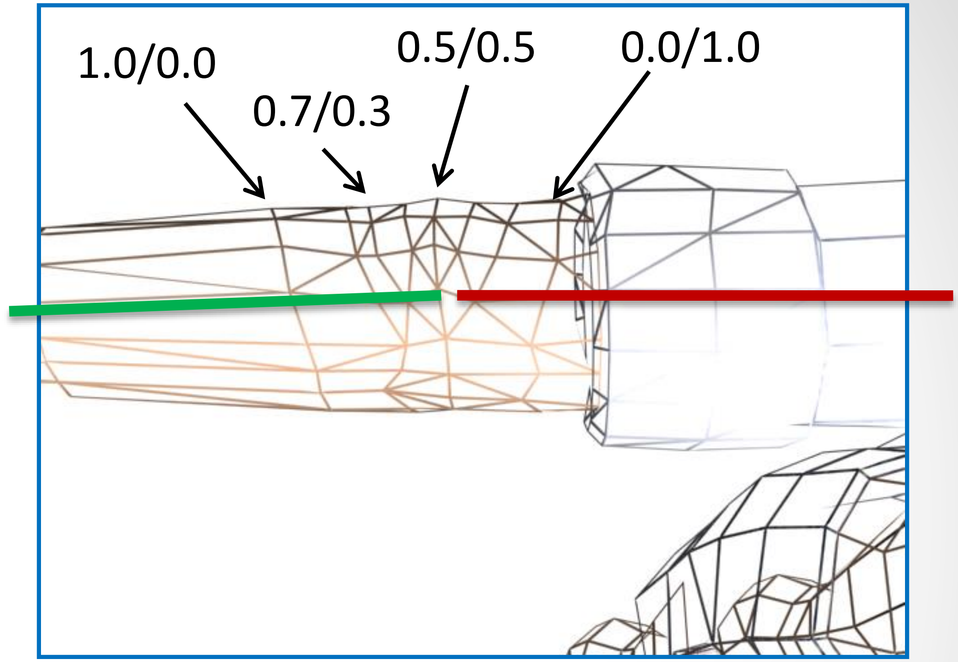
M_i is a transformation matrix.



Each transformation matrix indicates how one bone has been moved.

Smooth Skin

- A vertex can be attached to more than one joint/bone with adjustable weights that control how much each joint affects it
  - Rarely more than 4

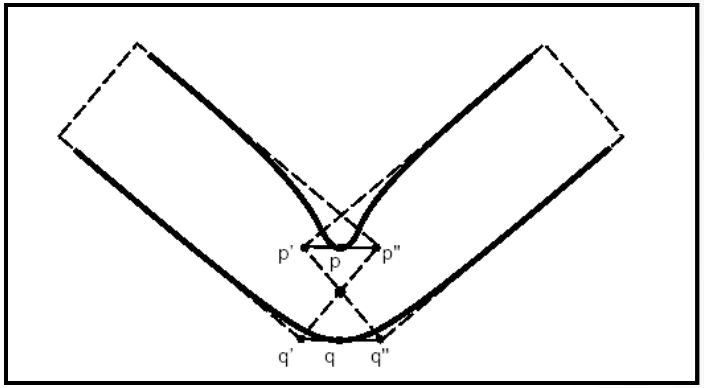


- Result is a blending of the n transformations

- Algorithm names
  - blended skin, skeletal subspace deformation (SSD), multi-matrix skin …

Limitations of Smooth Skin

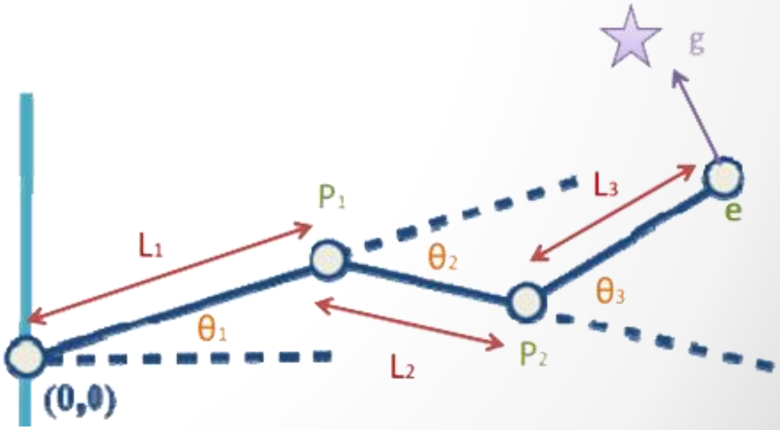
- Smooth skin is very simple and quite fast, but its quality is limited
  - Joints tend to collapse as they bend more



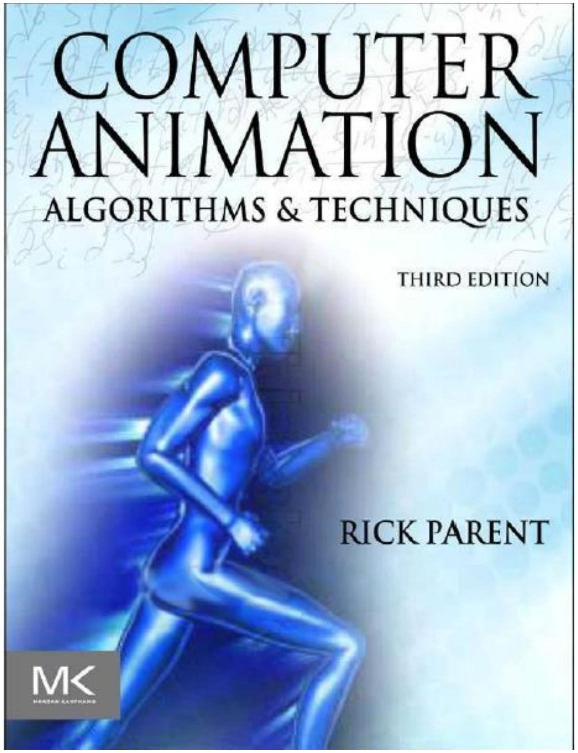
- Still, it is common in animation and games
- If nothing else, it is a good baseline upon which more complex schemes can be built

Inverse Kinematics

- Forward Kinematics
  - Generate pose by manually adjusting pose parameters
  - Trial-and-error process
- Inverse Kinematics
  - Only specify the target position/orientation of the end effector
  - The pose parameters are automatically computed
  - **Coursework part 1**



More about Inverse Kinematics



Chapter 5

Coursework

- Coursework 1
  - individual project on Inverse Kinematics
  - deadline: Friday November 24th 2017
  - code (50%) + report (50%)
  - demo session

Coursework

- Coursework 2
  - literature survey on related topics
    - Implicit modelling and its uses in animation and games
    - Freeform deformation and its uses in animation and games
    - Modelling and rendering of fur and hair
    - ...
  - ask for approval of your own topics
  - deadline: Friday January 19th 2017
  - 3000-4000 words