Skinning

Character Animation

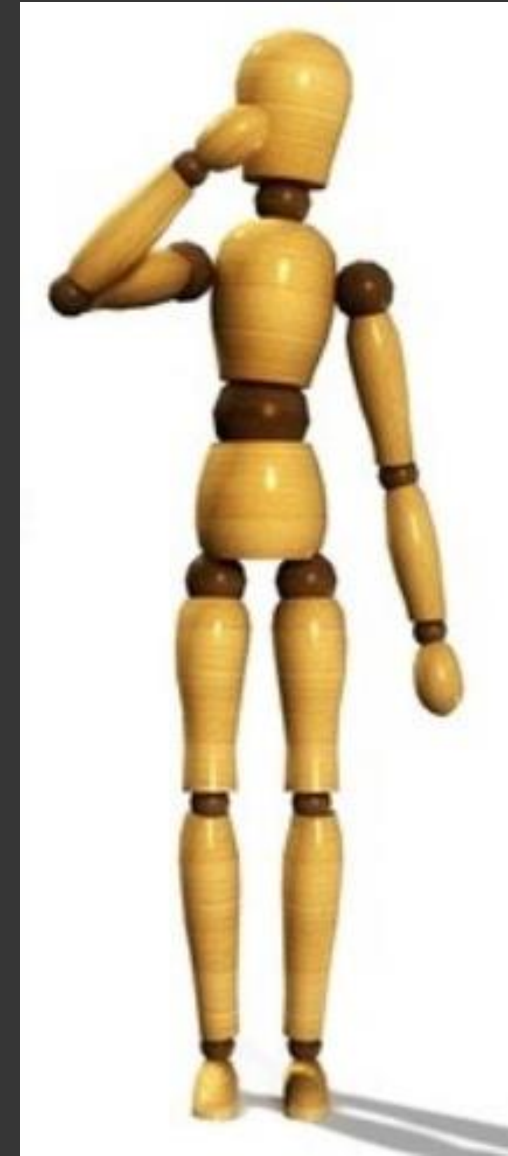
- The task of moving a complex, artificial character in a life-like manner
  - Animating characters is a particularly demanding area
  - Animated character must move and deform in a manner that is plausible to the viewer

Character Animation

- Animating a character model described as a polygon mesh by moving each vertex in the mesh is <u>impractical</u>
- Instead - specify the motion of characters through the movement of an internal articulated <u>skeleton</u>
  - Movement of the surrounding polygon mesh may then be deduced
- Mesh must deform in a manner that the viewer would expect, <u>consistent</u> with underlying muscle and tissue

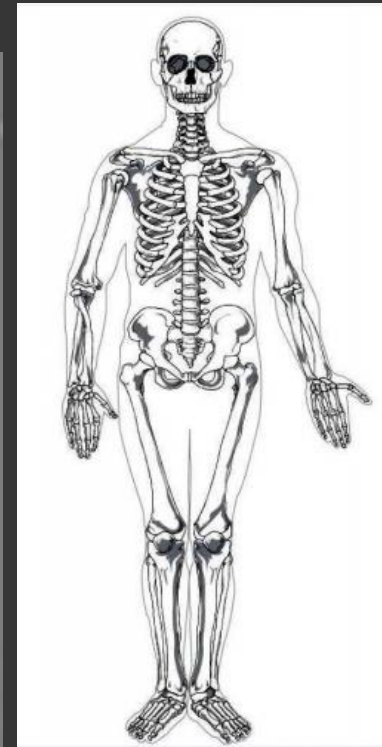
Articulated Models

- A collection of objects connected by joints in a hierarchical structure
- The objects and their relative connections define the static object skeleton
- The joint parameters (angles) define the stance of the model
- Animation is achieved by changing the joint parameters

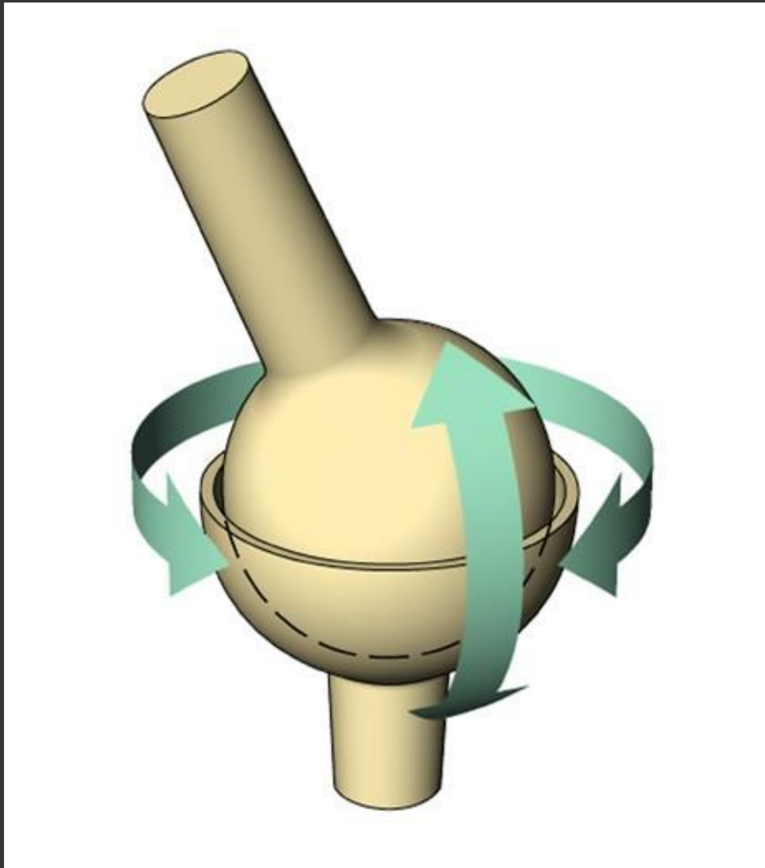


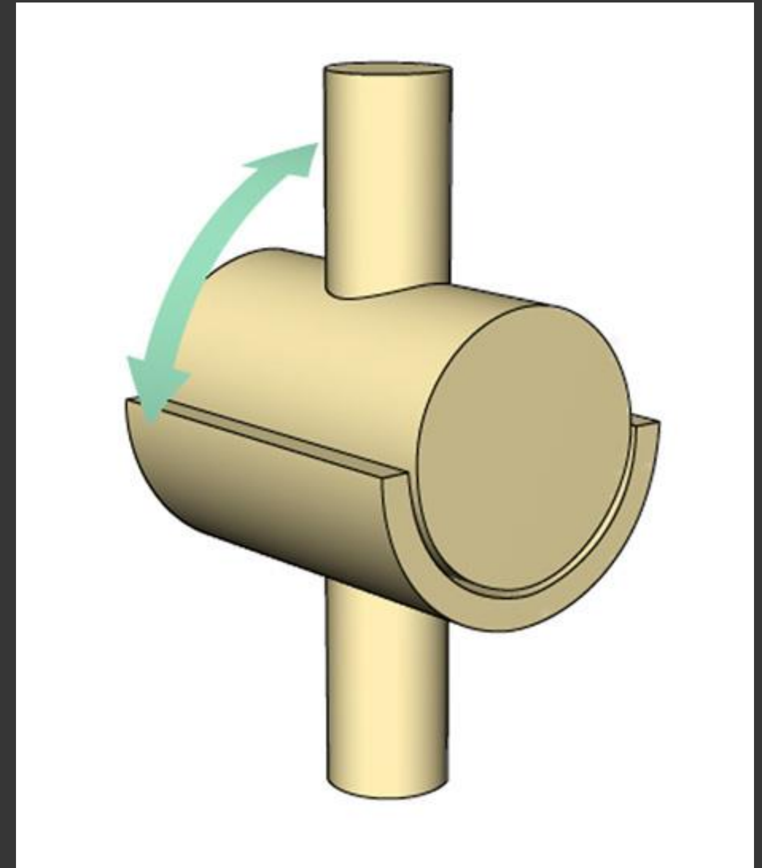
Human Skeleton

- Complex structure
  - Adult: 206 bones
- Spine: 33 vertebrae
  - Impractical to model each vertebra
  - Typically use 3/4 spine links
- Shoulders
  - Can translate as well as rotate
  - Wide range of motion
  - Prone to dislocation
- Fingers?



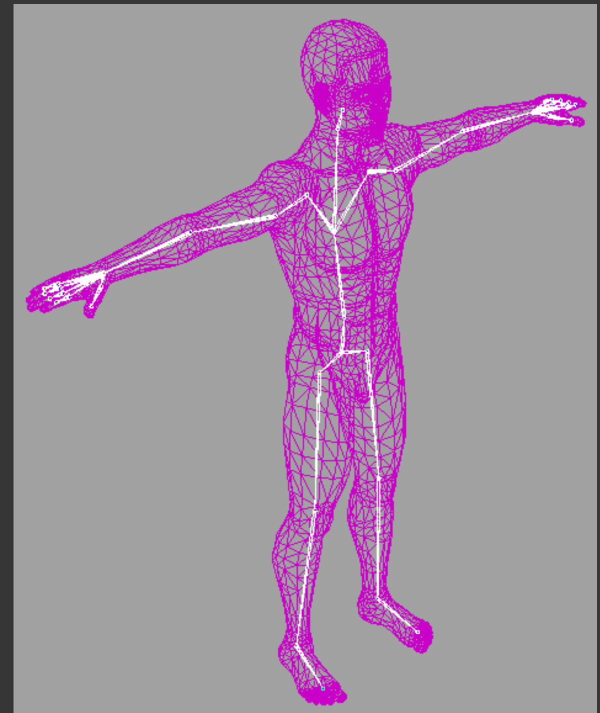
Joint Approximations



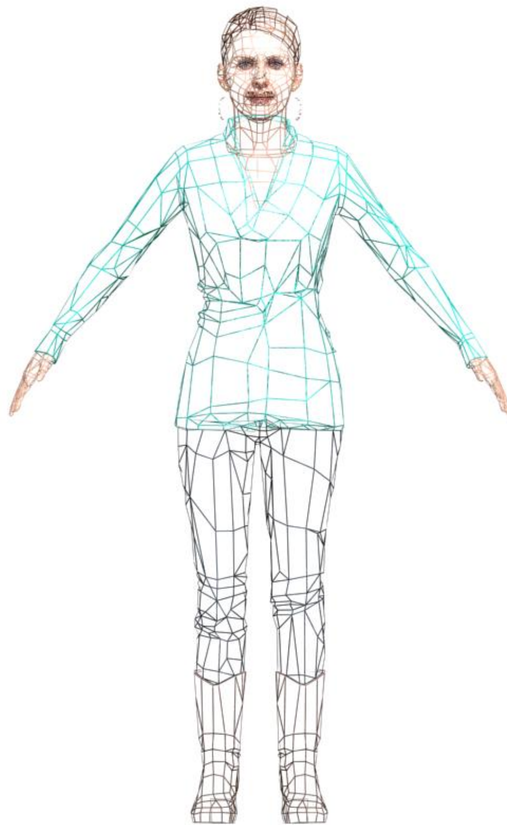


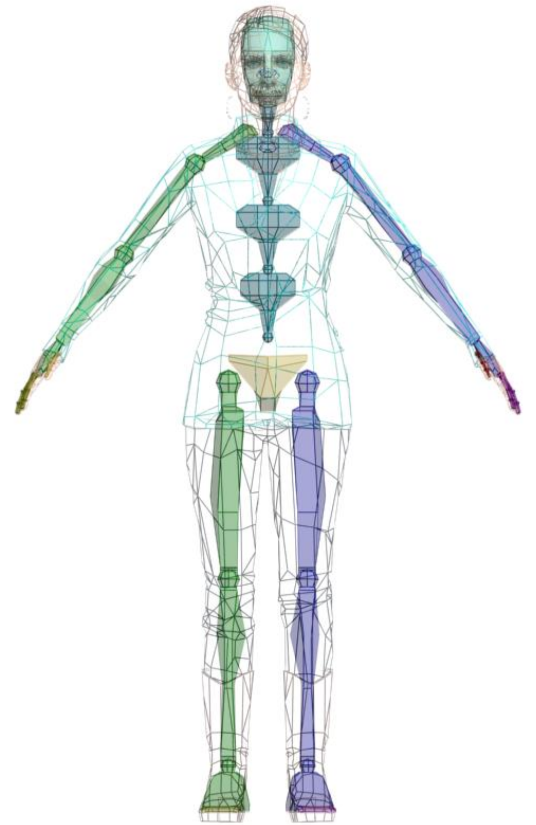
Character Animation - Skeleton

- These soft-bodied models appear to have hierarchies, much like rigid bodies
- So we can define an **independent** hierarchy of ***bones*** assumed to lie within the geometry
  - This is called a skeleton
    - Analogous to a human skeleton
  - The movement of the bones drives the overlaid geometry
  - Parts of the geometry bend and flex depending on the nearby bones



Skeleton





Skinning

- Skinning is the process of attaching a renderable skin to an underlying articulated skeleton.
- There are several approaches to skinning with varying degrees of realism and complexity.
- **Binding** refers to the initial attachment of the skin to the underlying skeleton and assigning any necessary information to the vertices

Rigid Skinning

- Set of rigid components
  - Each component attached to a single bone
  - Each component mesh is simply transformed into world space by the appropriate joint world matrix.
  - Robots and simple characters made up from a collection of rigid components can be rendered through classical hierarchical rendering approaches.
  - This results in every vertex in the final rendered character being transformed by exactly one matrix.

Rigid Skinning

- For every vertex, we compute the world space position by transforming the local space position by the appropriate joint world matrix
- Every vertex in each mesh is transformed from the joint local space where it is defined into world space, where it can be used for further processing such as lighting and rendering.
- Works fine for robots, mechanical characters, and vehicles
  - Not appropriate for organic characters with continuous skin.

Rigid Skinning Limitations

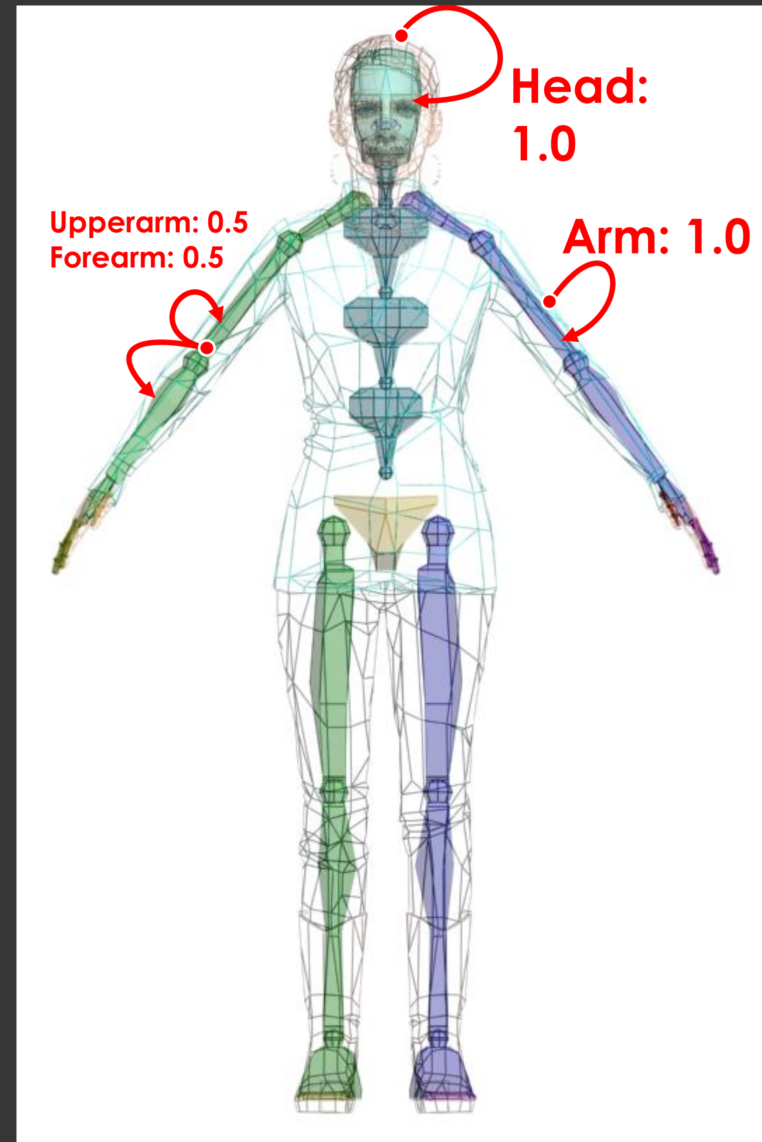
- Consider human joints
  - When they bend, the body shape bends as well
    - No distinct parts
  - We cannot represent this with rigid bodies
    - Or the pieces would separate, where there should be stretching or compression

Rigging

- Each vertex in the mesh can be attached to more than one joint
  - Each attachment affects the vertex with a different strength or weight.
- The final transformed vertex position is a weighted average of the initial position transformed by each of the attached joints.
- Many vertices will only need to attach to one or two joints and rarely is it necessary to attach a vertex to more than four.

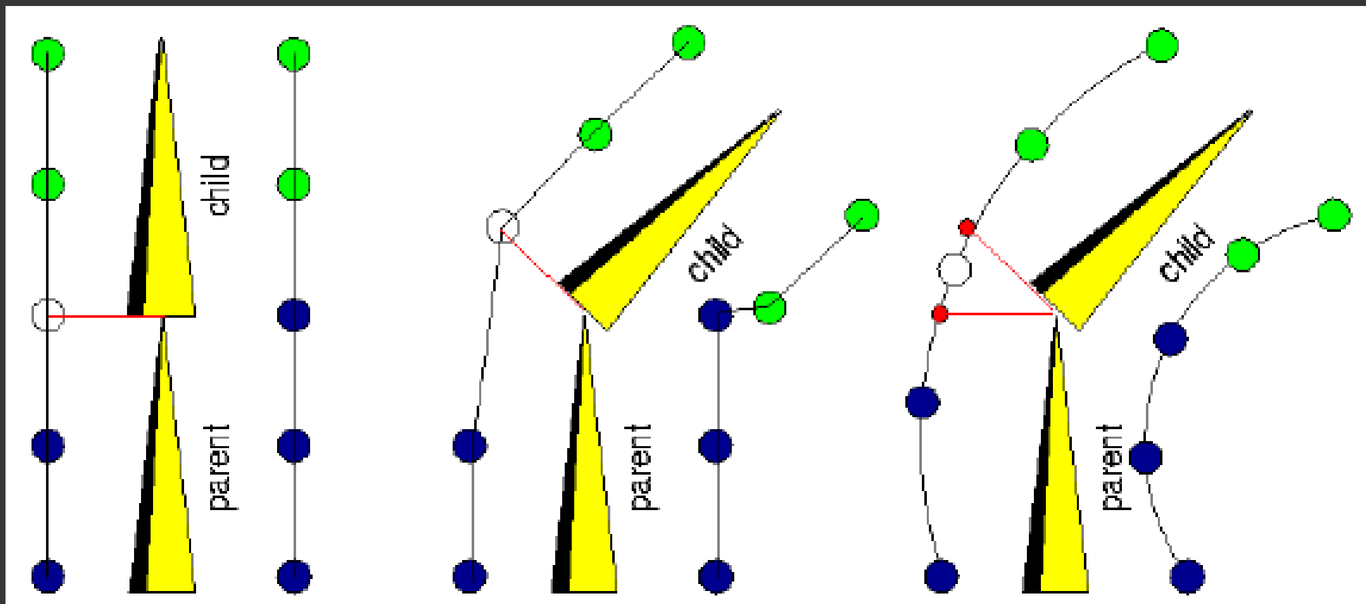
Rigging

- The artist manually creates a skeleton for the target model
  - Define correspondences between mesh and skeleton

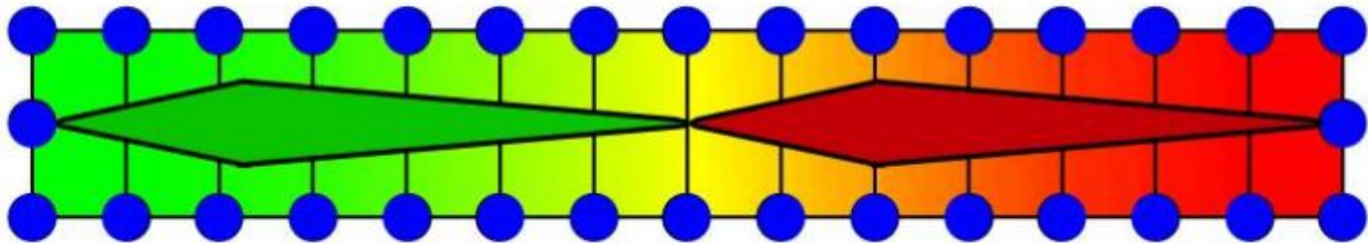


Rigging

- Associate each point with nearest link
  - When link moves, transform its points.
- Each point gets affected by several links
  - Take weighted average
  - Adjust the weights until it looks good

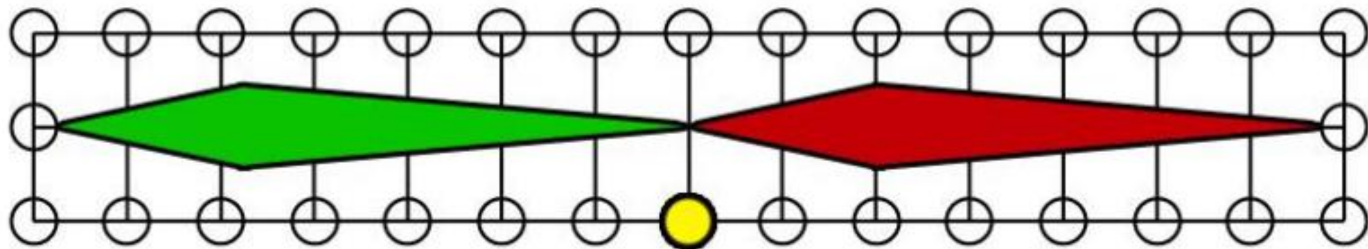


Rigging



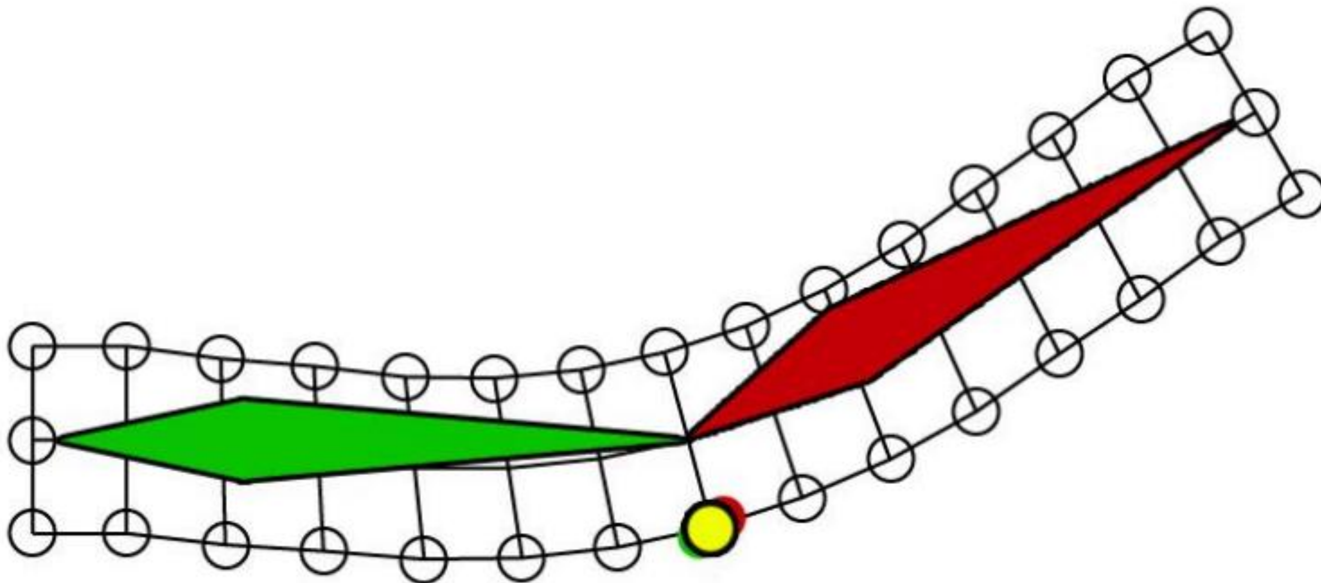
http://graphics.cs.uh.edu/ble/papers/2012sa-ssdr/SSDR-

Rigging



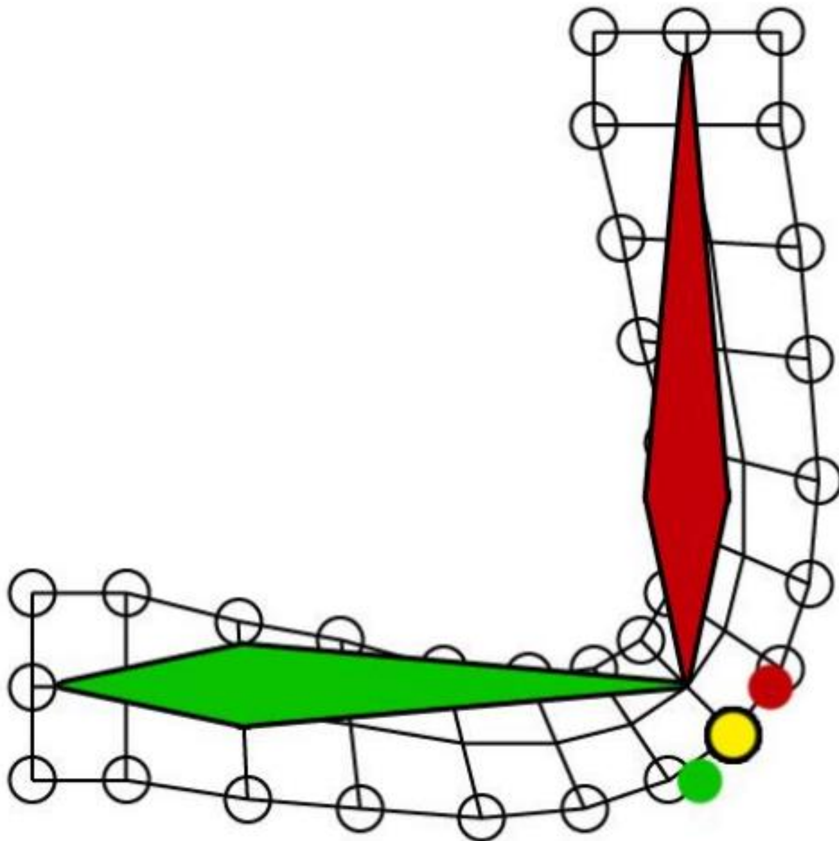
http://graphics.cs.uh.edu/ble/papers/2012sa-ssdr/SSDR-

Rigging



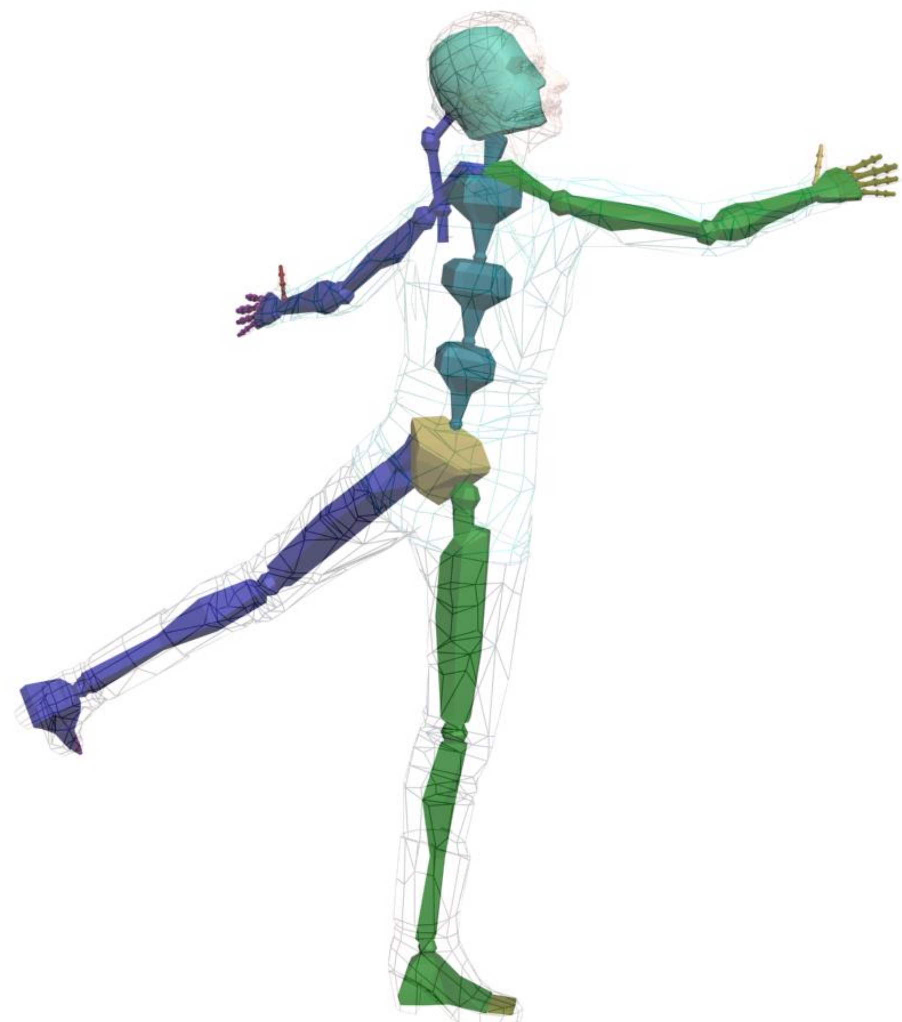
http://graphics.cs.uh.edu/ble/papers/2012sa-ssdr/SSDR-

Rigging



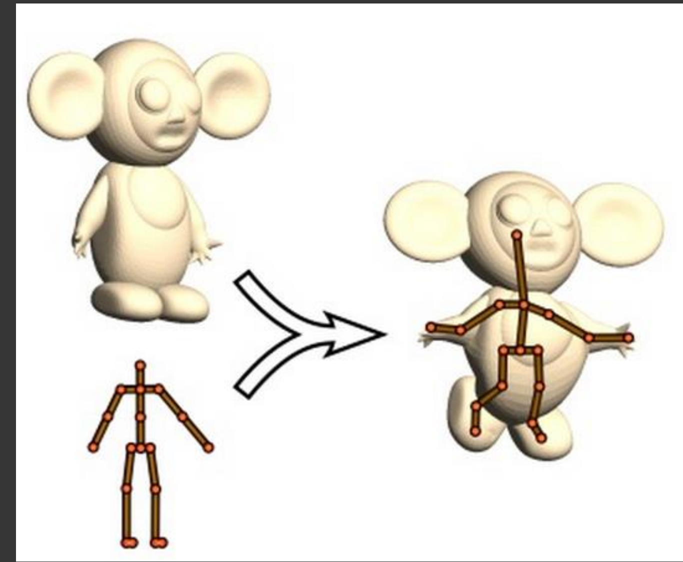
http://graphics.cs.uh.edu/ble/papers/2012sa-ssdr/SSDR-

Rigging



Automatic Rigging

- Rigging is time-consuming and tedious even for experienced animators



- The change in the behaviour of a vertex as its weights are changed is often counterintuitive and it may not be clear whether a value exists which gives the desired position.

- Some researches have looked at automatic rigging methods
  - e.g. "Automatic Rigging and Animation of 3D Characters" Baran & Popovic

- Online solutions, e.g., www.mixamo.com

Linear Blend Skinning

- Used in games, a.k.a.
  - Linear blend skinning
  - Skeletal subspace deformation
  - Enveloping
  - Vertex Blending
- Determines the new position of a vertex by **linearly combining** the results of the vertex transformed rigidly with each bone.
  - Each influencing bone is given a scalar weight w_i
  - Weighted sum gives the vertex's position in the new pose
  - Weights set such that sum of all weights for a vertex = 1

Hardware Skinning

- 3D artist supplies for each vertex
  - Index or indices of the joint(s) to which it is bound
    - 4 joint limit typical
  - A weighting factor for each joint describing how much influence that joint should have
    - Must sum up to 1
    - Last weight often omitted & calculated at runtime

Vertex Data Structure

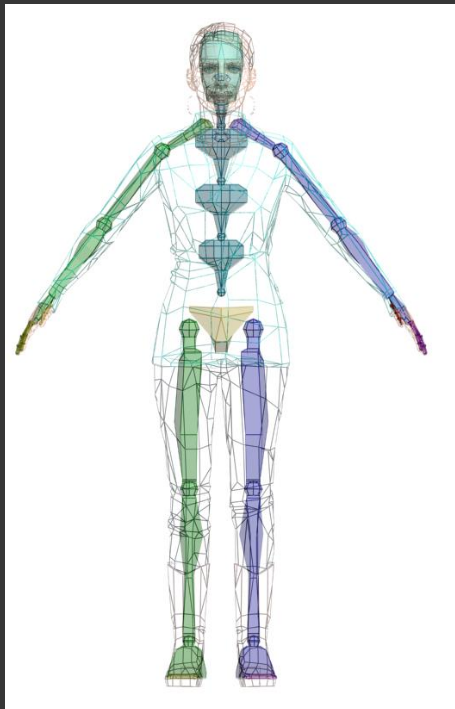
- struct **SkinnedVertex** {

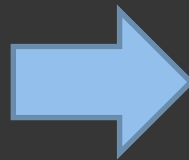
```
    float m_position[3]         // (Px, Py, Pz)
    float m_normal[3]           // (Nx, Ny, Nz)
    float m_u, M_v;             // texture coordinates
    int m_jointIndex[4]         // joint indices
    float m_jointWeight[3];     //joint weights, last omitted
}
```

Mathematics of Skinning

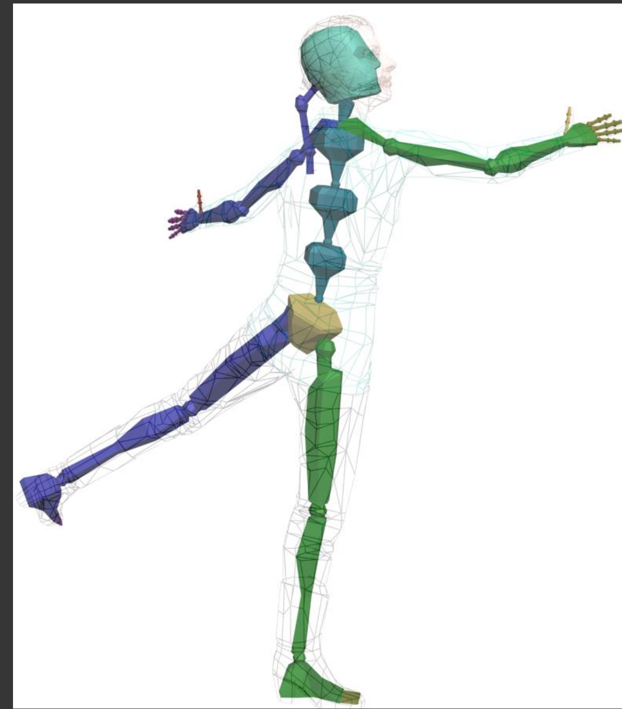
- Need a matrix to transform the vertices (in model space) of the mesh from *original* positions into *new* positions

Bind Pose



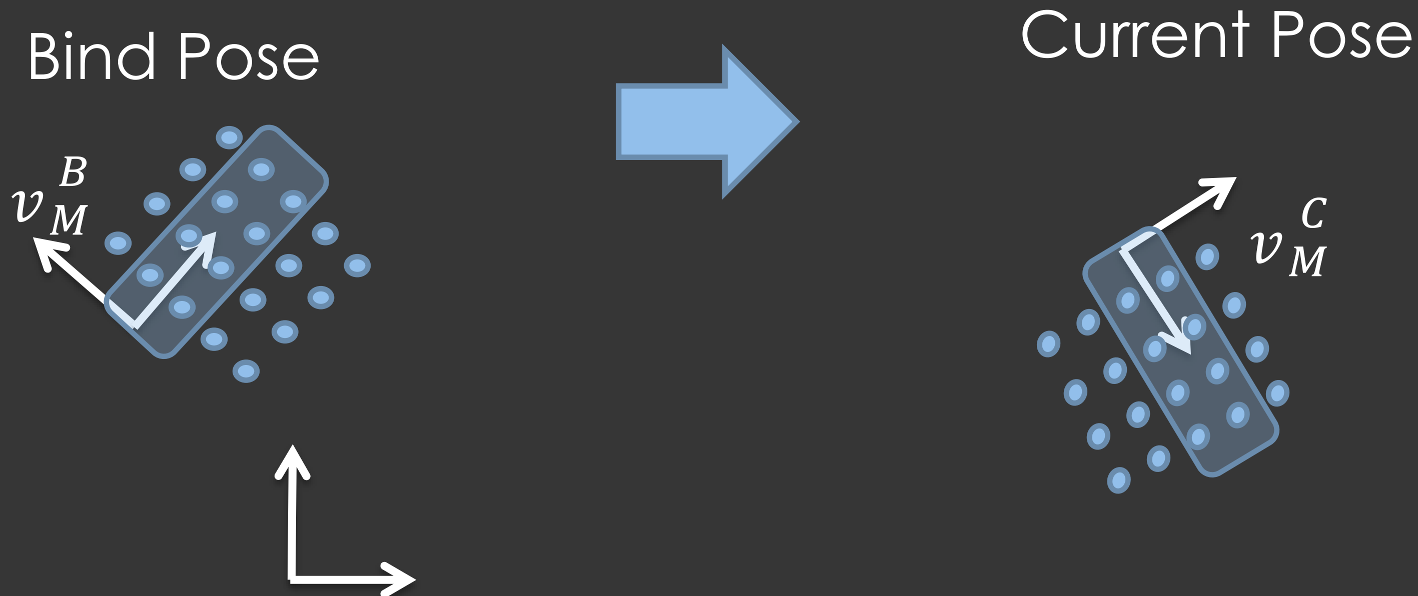


Current Pose



Simple Example

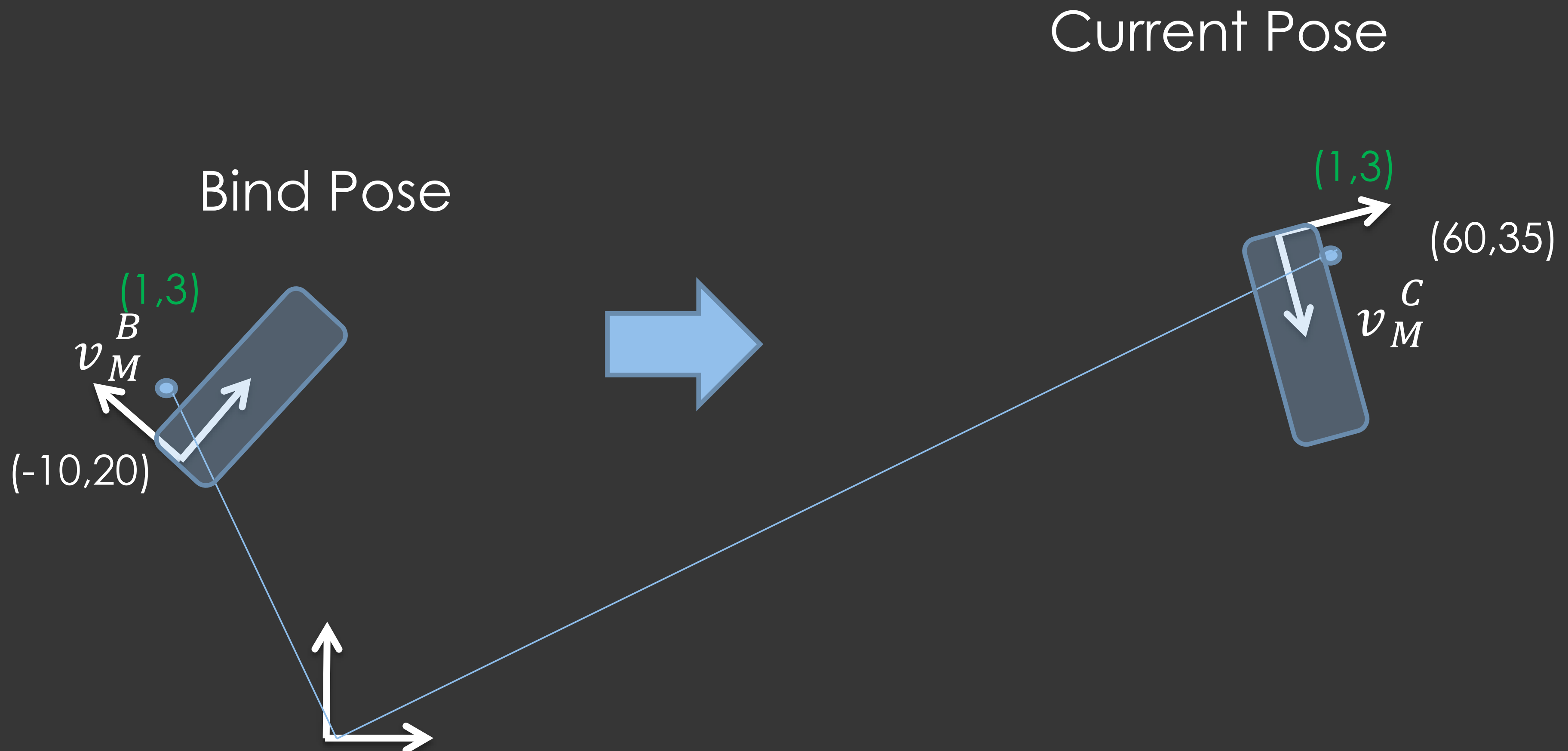
- Lets take a one-jointed skeleton example
  - Model space – M
  - Joint space – J
  - Bind pose – B
  - Current pose – C (new pos & ori in model space)



Simple Example

- Trick: position of a vertex bound to a joint is *constant* when expressed in that joints coordinate system
  - Take bind-pose position of vertex
    - convert to joint space
    - move joint to current pose
    - convert back to model space

Simple Example

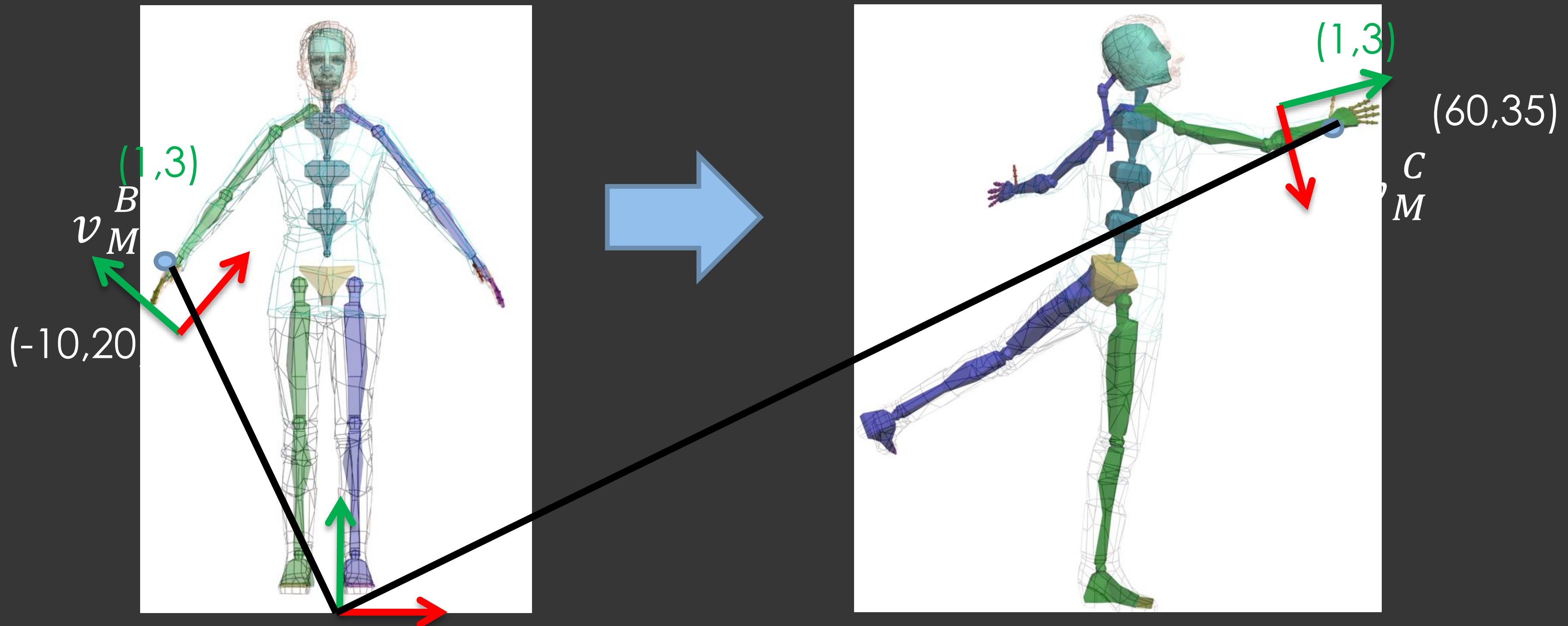


Simple Example

- Bind pose of joint in model space:
  - $B_{j \to M}$
  - This matrix transforms point from J to M
- Now, consider v_M^B, need in joint space
  - Convert using *inverse bind pose matrix*
  - $(B_{j \to M})^{-1}$
  - $v_j = (B_{j \to M})^{-1} \, v_M^B$
- If $C_{j \to M}$ is the joints *current pose*
  - Convert v_j back to model space
  - $v_M^C = C_{j \to M} (B_{j \to M})^{-1} \, v_M^B$

Skinning matrix
K_j

Complex Example



More Complex Example

- Previous example only considered one joint
- Extend to multiple joint skeleton
  - Make sure that $B_{j \to M}$ and $C_{j \to M}$ calculated properly for joint in question (concatenate transformation of parents)
  - Calculate an array of skinning matrices, one for each joint called the *matrix palette*.
- Matrix palette is passed to rendering engine when rendering a skinned mesh
  - Used to transform vertex from bind pose to current pose

Matrix Management

- $C_{j \to M}$ changes every frame as the character assumes different poses over time
- $(B_{j \to M})^{-1}$ constant throughout the game
  - Generally cached with skeleton
  - Not calculated at runtime
- Animation engines calculate local poses for each joint $C_{j \to P(j)}$ then convert into global poses $C_{j \to M}$ before multiplying by corresponding cached inverse bind pose matrix
- This gives us a K_j for each joint

Model-to-World Transform

- Need to transform each vertex to world space
- Pre-multiply palette of skinning matrices by the object's model-to-world transform
  - Save rendering engine one matrix multiply per vertex
- $(K_j)W = M_{M\to W} C_{j\to M} \left(B_{j\to M}\right)^{-1}$
- Why not bake this transform?
- Ok sometimes but sometimes not?

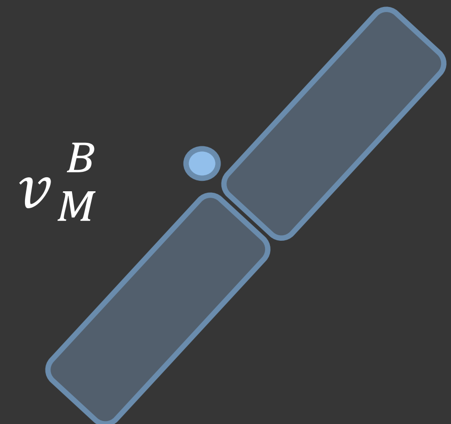
Animation Instancing

- For crowds
  - Keep model-to-world transforms separate
  - Share single matrix palette across all characters

Multiple Joints per Vertex

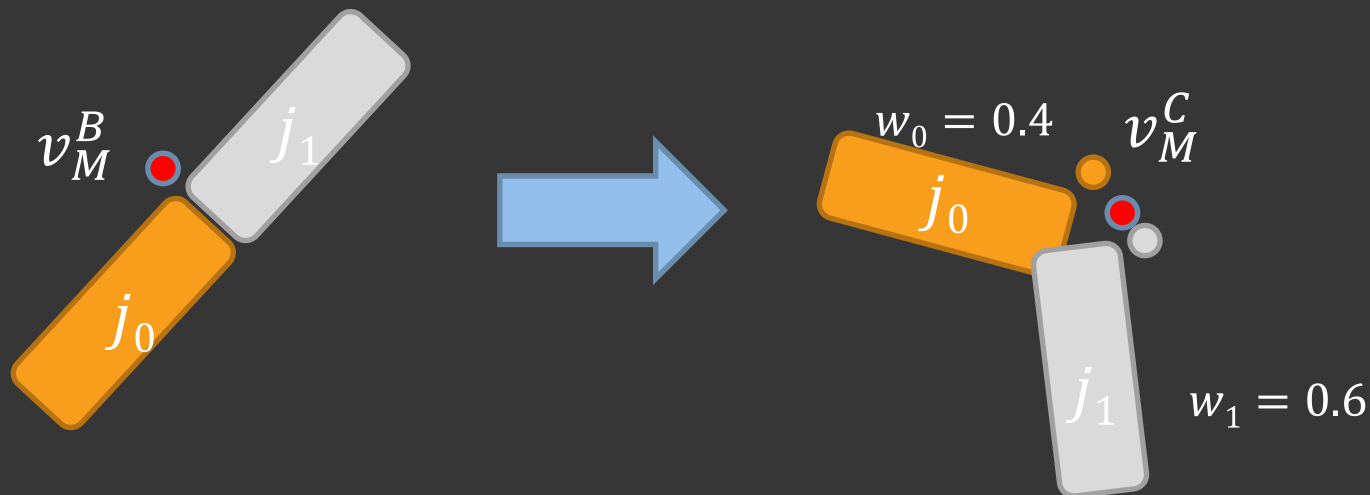
- When a vertex is skinned to more than one joint
  - Calculate model space position for each joint
  - Take a weighted average of resulting positions
  - Weights provided by artist (must sum to 1)
- For a vertex skinned to N joints, with indices j_0 to j_{N-1} & weights w_0 to w_{N-1}, equation:

$$v_M^C = \sum_{i=0}^{N-1} w_{ij} K_{j_i} v_M^B$$



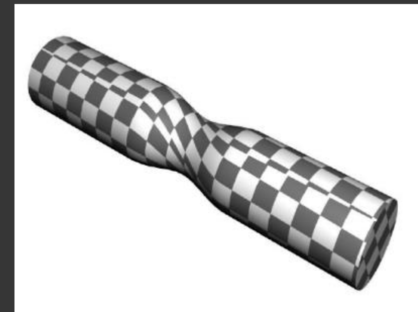
Multiple Joints per Vertex

$$v_M^C = \sum_{i=0}^{N-1} w_i K_{j_i} v_M^B$$



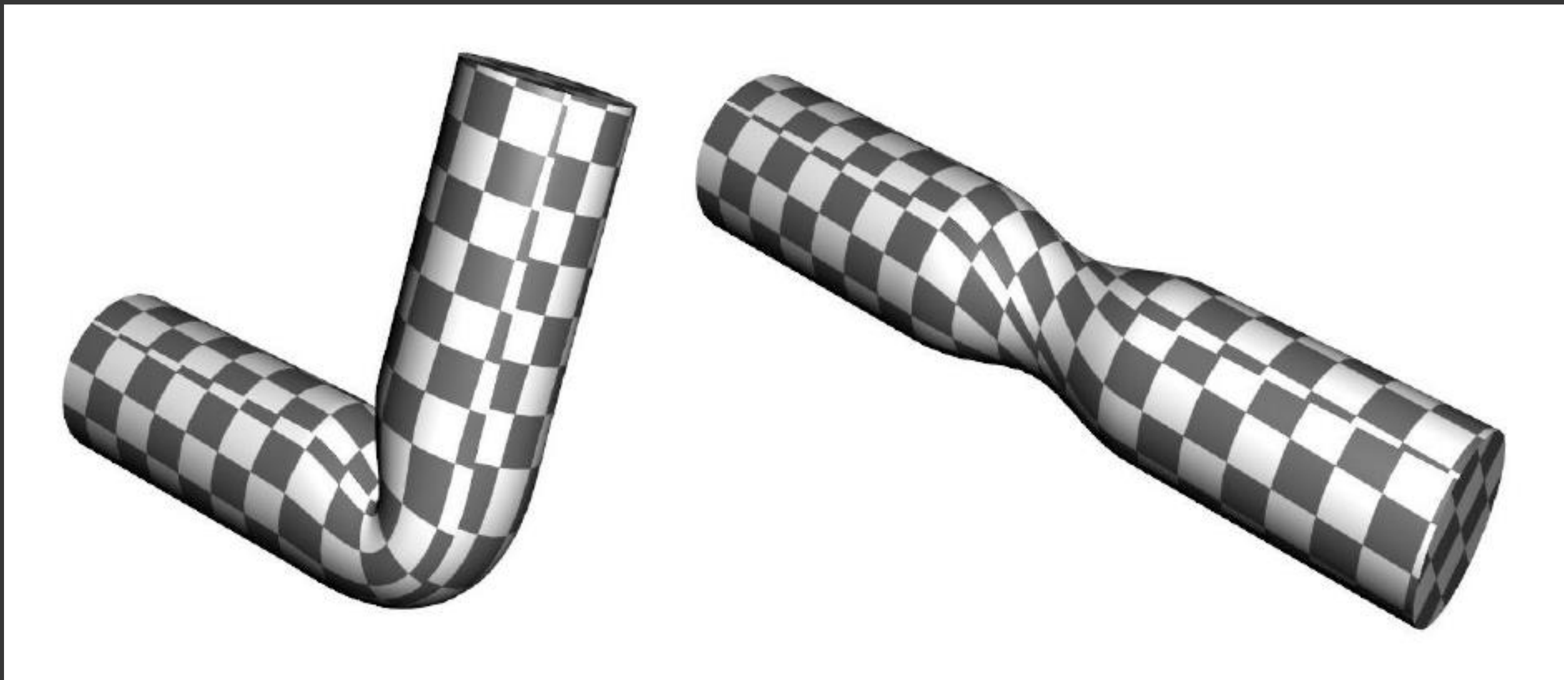
Problems

- Volume loss as joints rotated to extreme angles
  - Collapsing elbow joint
  - Candy wrapper effect on wrist
- Due to lack of flexibility in the framework
- Linear interpolation of transformation matrices is not equivalent to linear interpolation of their rotations

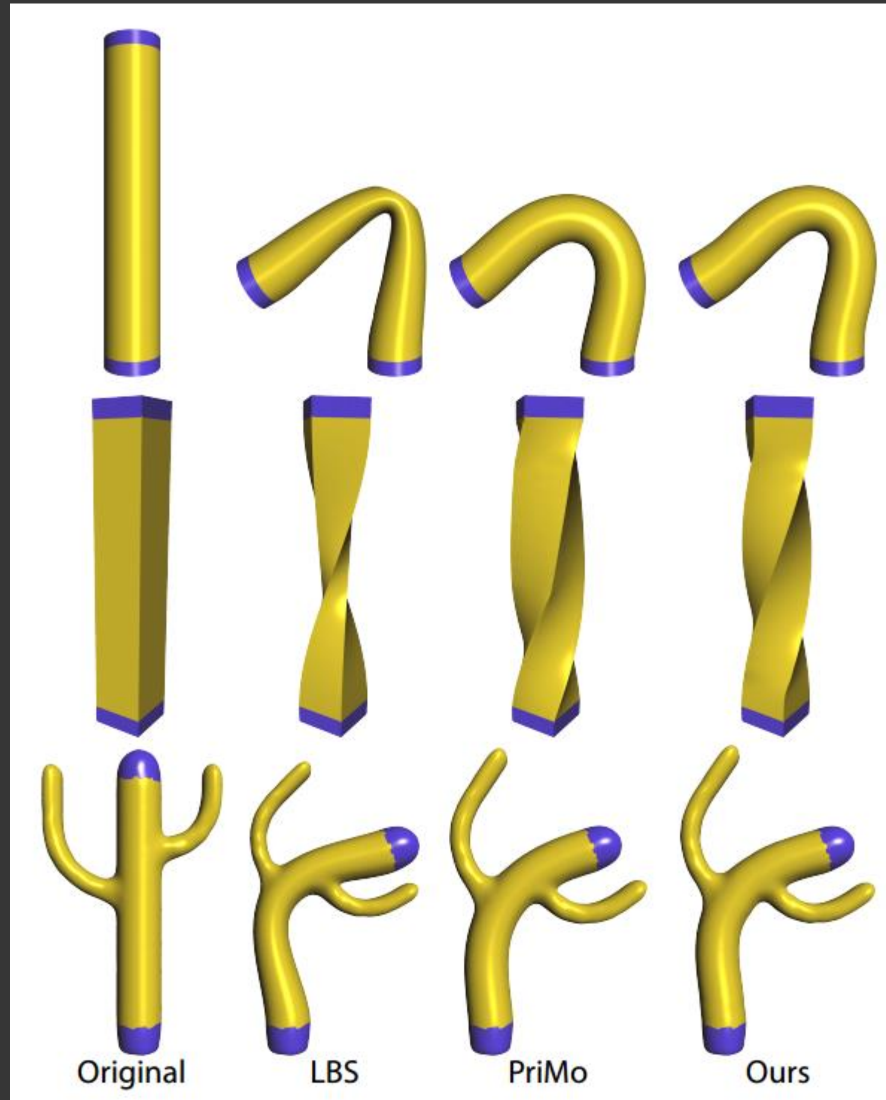


Popularity

- Simplicity and computational efficiency



Other approaches exist



Solutions

- Avoid poses with big variation angles
- Add extra bones
- Use a more sophisticated linear blend skinning solution
  - Dual quaternions
- Use a non-linear blend skinning solution

Muscle-Based Models