14. Character Animation

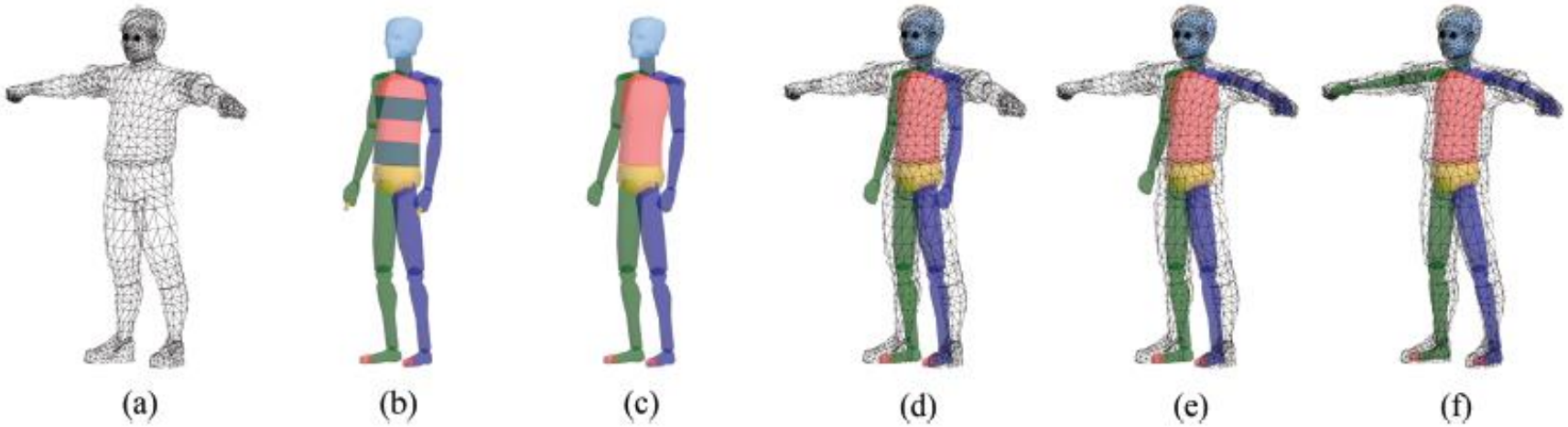
Advanced
Game Graphic Programming

Prof. HyeongYeop Kang
siamiz@khu.ac.kr

Skeleton

Skeleton?

- The most popular method to animate a character is to use a *skeleton*, which is an *articulated body* composed of rigid components, *bones*.
  - The initial pose of a character is called the default pose. (a).
  - 3ds Max provides a skeleton template named biped. (b).
  - Such a template can be edited. In (c), four spines shown in (b) are combined into one.
  - The skeleton is embedded into the character mesh. (d).
  - It is then made to fit to the default pose. (e) and (f).

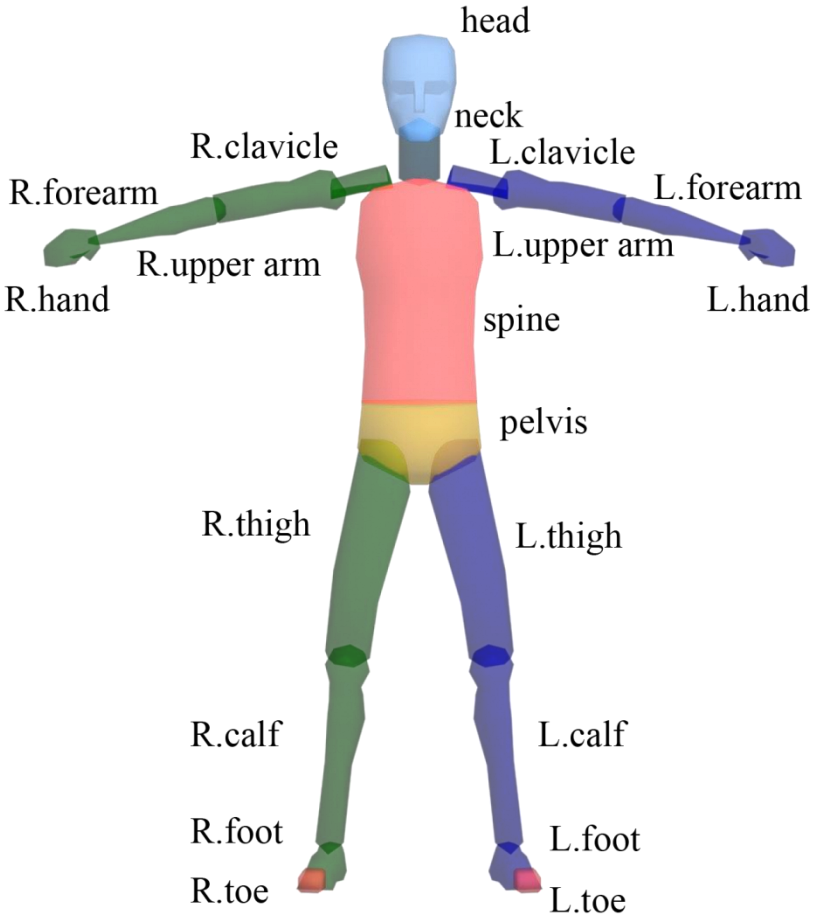


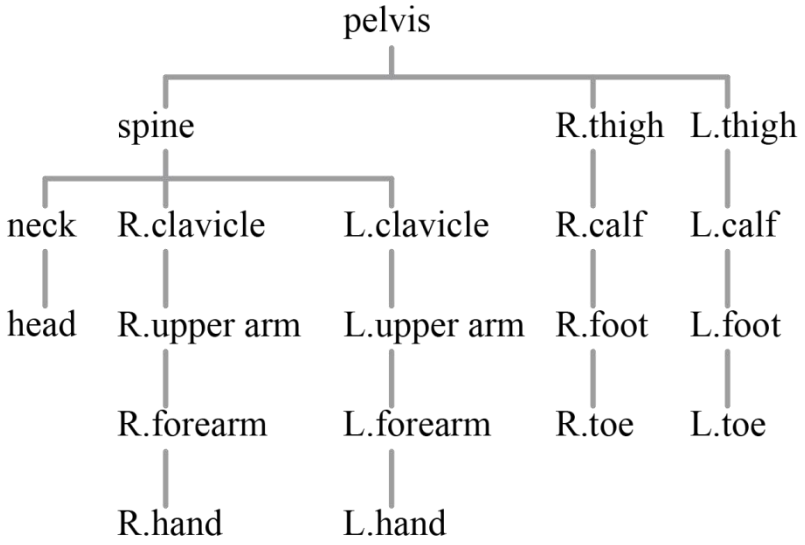
- The bones are depicted as if they were solids. However, they are not. When the skeleton is made to fit to the default pose, a *matrix* is computed per bone.

Skeleton

Skeletal hierarchy

- In a skeleton, the bones form a hierarchical structure with parent-child relationships.
- The pelvis is normally taken as the root node of the hierarchy.

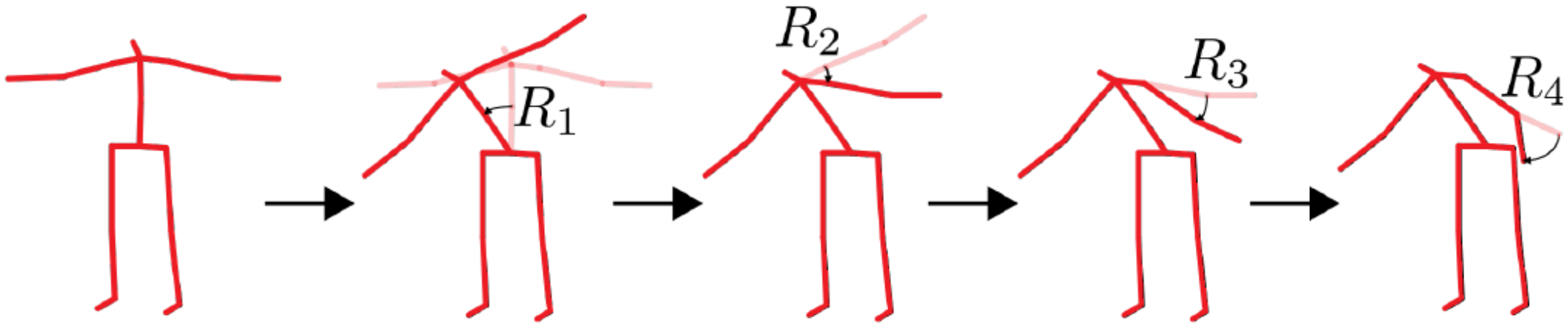




Skeletal Animation

Characteristics of skeletal animation

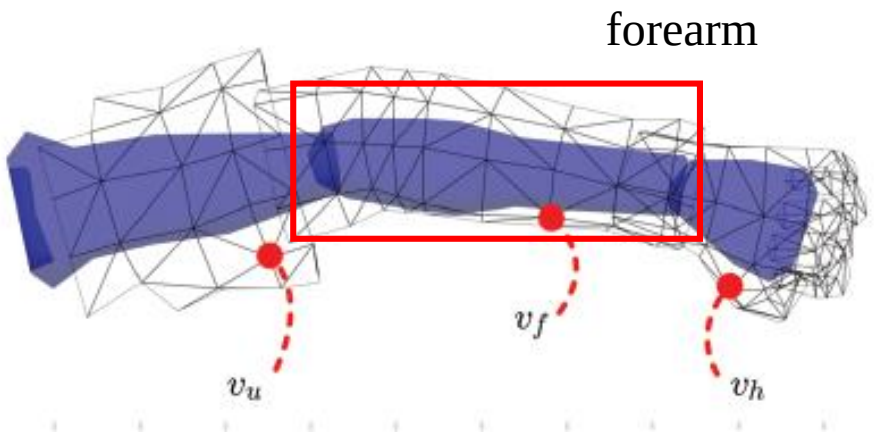
- Each bone has its own local motion, which is usually restricted to rotation.
- The motion of a bone affects all of its descendants.

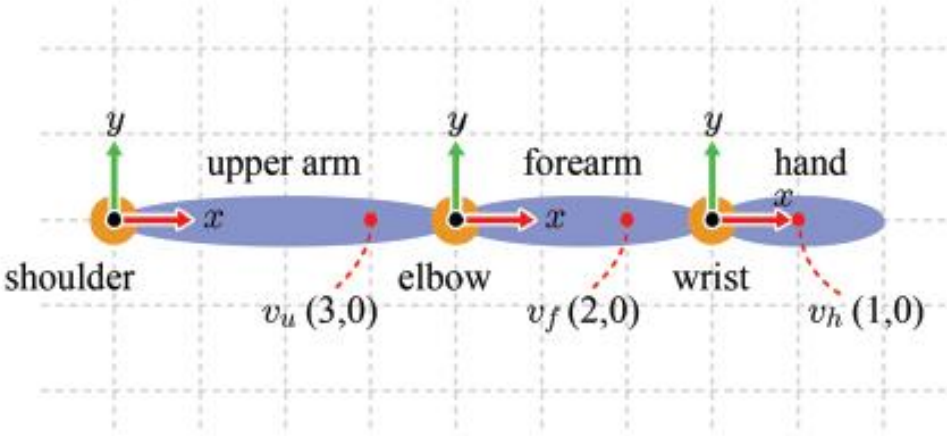


Space Change between Bones

Joints

- The bones are connected at *joints*.
- When the forearm moves, for example, v_f has to move accordingly.
  - It is simply achieved if v_f is defined in the forearm's object space.
- Initially, all vertices of the default pose are defined in the character's object space.
  - To avoid confusion with the object space of a bone, let's simply call the character's object space *character space*.
  - On the other hand, let's call the object space of a bone as *bone space*.
- Every character-space vertex of the default pose needs to be transformed into a bone space.
  - For example, v_f will be transformed into the forearm's bone space so as to have the coordinates (2,0).



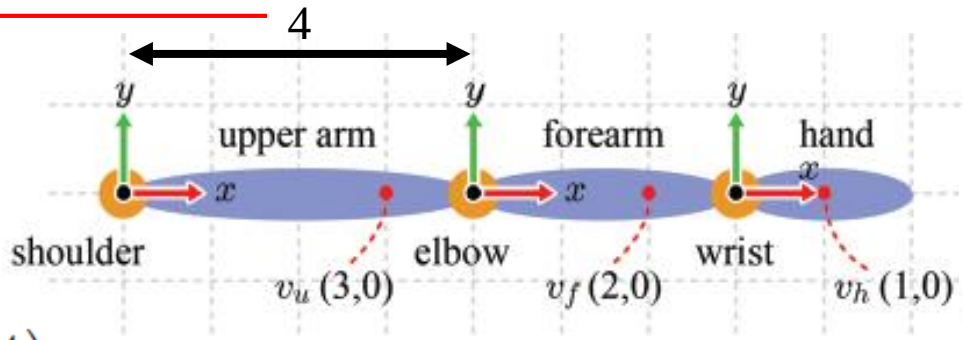


Space Change between Bones

Bone-to-character transform.

- If we can compute a *bone-to-character* transform, its inverse would be a *character-to-bone* transform.
- Consider the *to-parent* transform of the *forearm*. It transforms a forearm vertex to the bone space of its parent, the upper arm.

$$M_{f,p}v_f = \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \\ 1 \end{pmatrix}$$

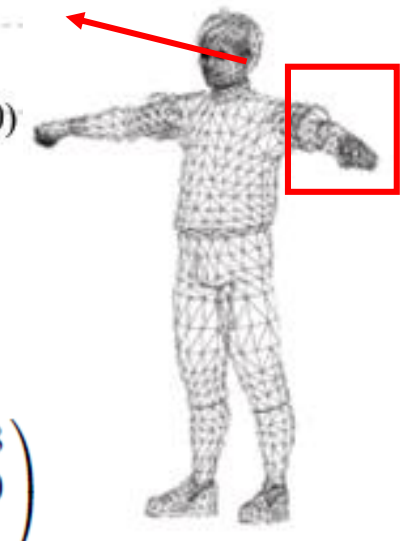


- To-parent matrix transforms a hand vertex v_h:

$$v'_h = M_{h,p}v_h = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

- v'_h defined in the forearm's space can be transformed into the upper arm's space.

$$M_{f,p}v'_h = \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 0 \\ 1 \end{pmatrix} \quad M_{f,p}v'_h = M_{f,p}M_{h,p}v_h = \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 0 \\ 1 \end{pmatrix}$$



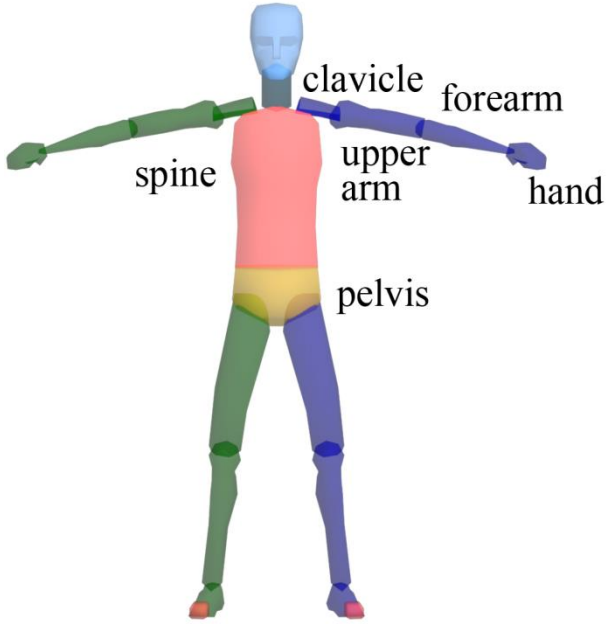
- Given a vertex at a bone, we can concatenate the *to-parent* matrices so as to transform the vertex into the bone space of any ancestor in the skeleton.

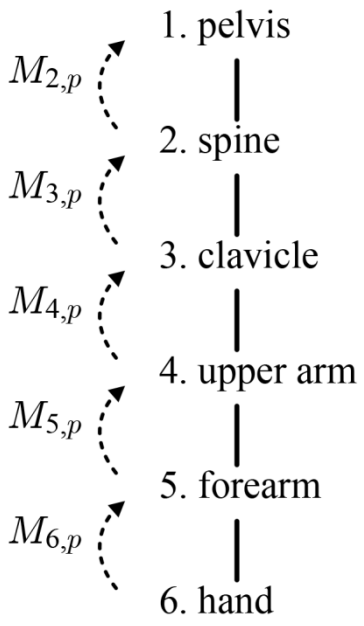
Character Space to Bone Space

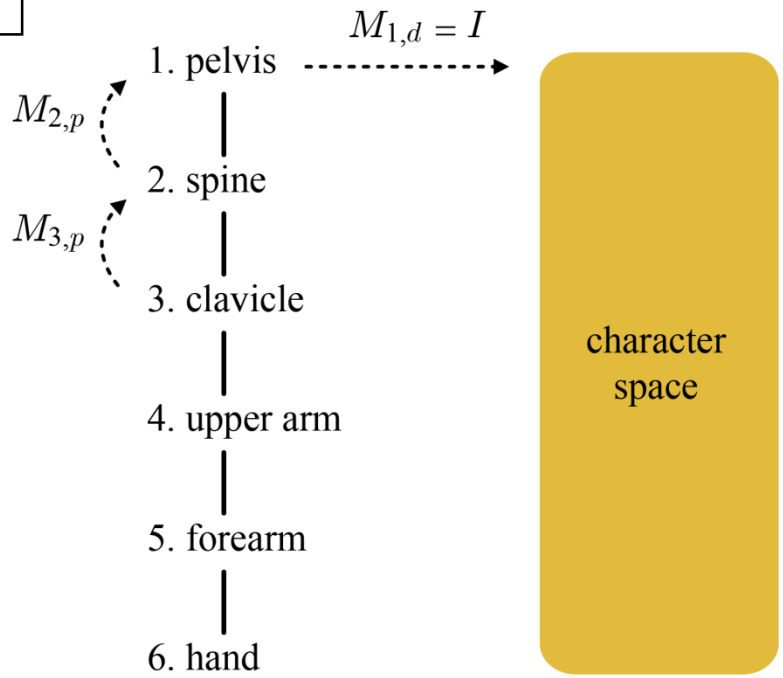
Bone-to-character transform.

- The to-parent matrix for the *i*-th bone is denoted by $M_{i,p}$.
- Let us denote the transform from the bone space to the character space by $M_{i,d}$, where *d* stands for default pose.

$$
\begin{aligned}
M_{2,d} &= M_{1,d}M_{2,p} \\
M_{3,d} &= M_{1,d}M_{2,p}M_{3,p} \\
&= M_{2,d}M_{3,p} \\
M_{i,d} &= M_{i-1,d}M_{i,p}
\end{aligned}
$$







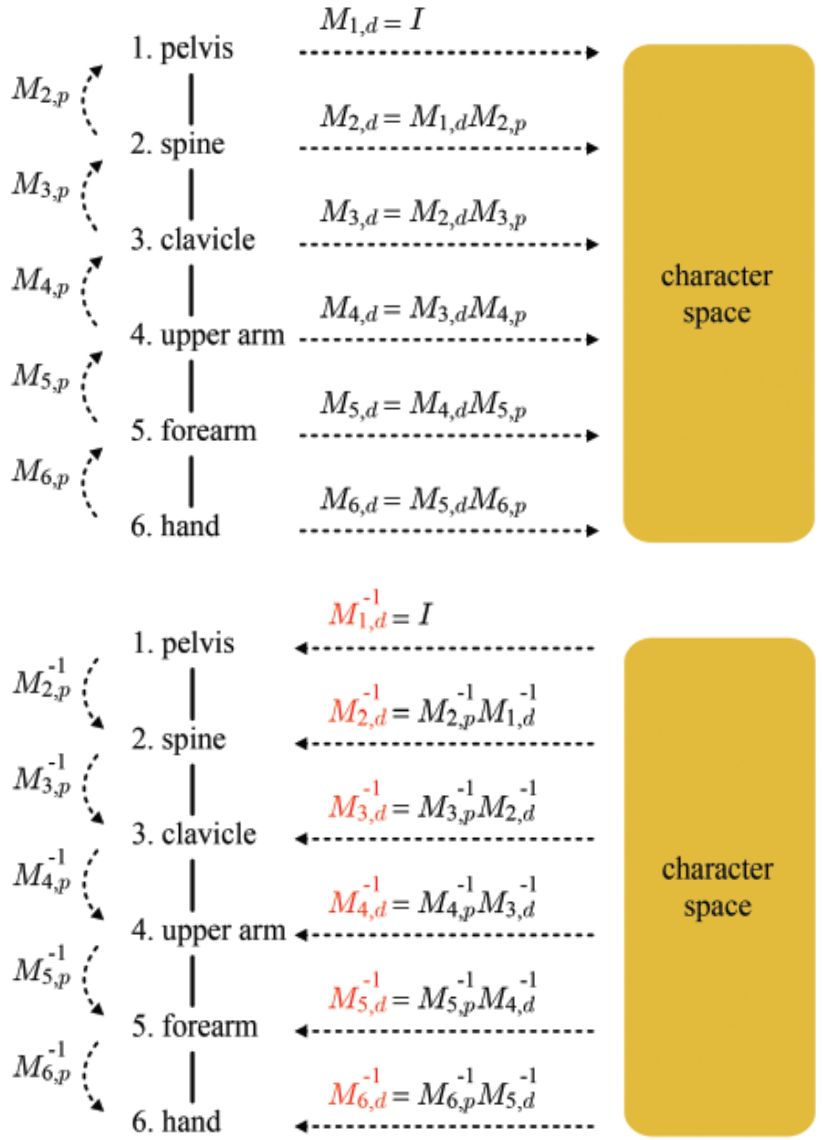
Character Space to Bone Space

Character-to-Bone transform.

- So far, we have considered the transform from the bone space to the character space. However, what is needed in an articulated-body animation is its inverse.

$$M_{2,d} = M_{1,d}M_{2,p}$$
$$M_{3,d} = M_{1,d}M_{2,p}M_{3,p} = M_{2,d}M_{3,p}$$
$$M_{i,d} = M_{i-1,d}M_{i,p}$$

$$M_{i,d} = M_{i-1,d}M_{i,p}$$
$$M_{i,d}^{-1} = M_{i,p}^{-1}M_{i-1,d}^{-1}$$

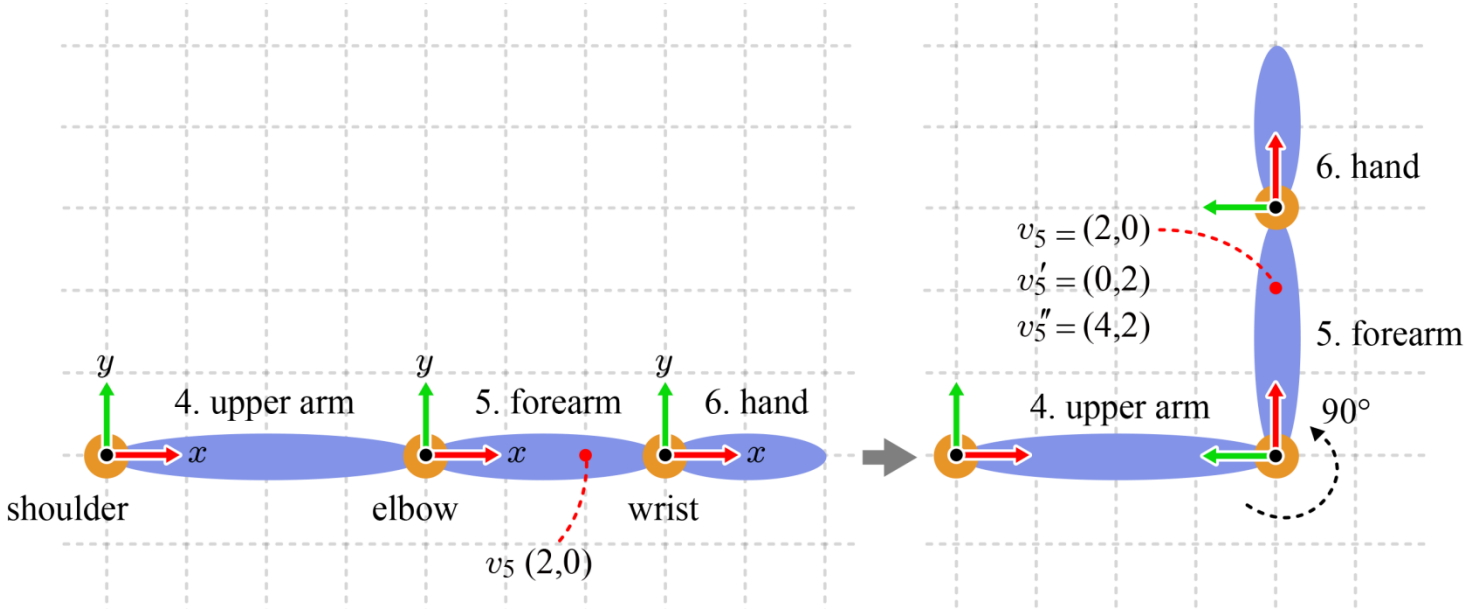


- Given the default pose, the to-parent transform of each bone, $M_{i,p}$, is immediately determined, and so is its inverse, $M_{i,p}^{-1}$.
- Computing $M_{i,d}^{-1}$ requires $M_{i-1,d}^{-1}$ to be computed in advance.
- As $M_{1,d}^{-1} = I$, we can compute $M_{i,d}^{-1}$ for every bone through the top-down traversal of the skeleton hierarchy.

Forward Kinematics

What if the bone moves?

- $M_{i,d}^{-1}$ transforms a character-space vertex into the *i*-th bone's space in the "default pose."
- Now the *i*-th bone is animated. Then, the vertices belonging to the bone are accordingly animated.



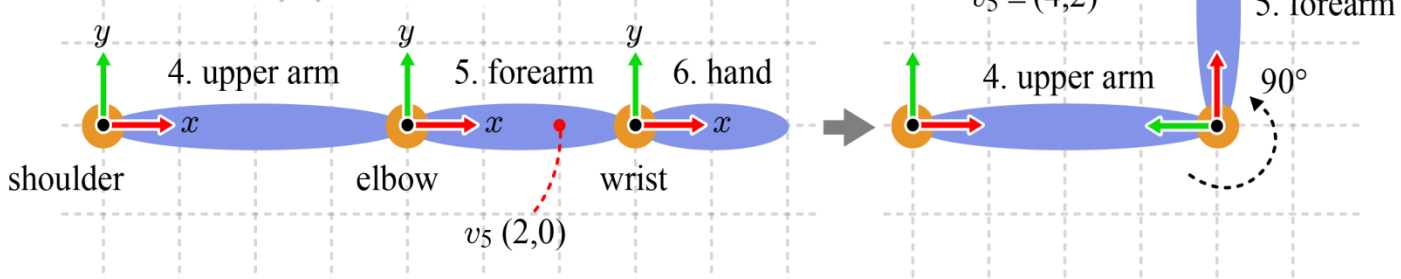
Forward Kinematics

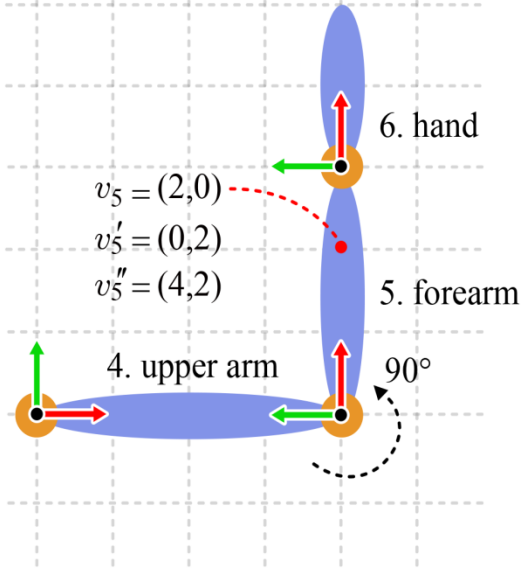
For the character rendering..

- The animated vertices should be transformed back to the character space so that they can be transformed to the world space, camera space and so forth, along the pipeline.
- Let us compute the matrix for animating v_5 and transforming it back to the character space. We name the matrix $M_{5,a}$.
- A bone's rotation is about often its *local* joint and is called local transform. For the forearm, it is denoted as $M_{5,l}$. It is applied to v_5. Whereas v_5 is fixed to (2,0), v_5' represents the coordinates with respect to the original bone space of the forearm "before rotation."
- The bone spaces of the upper arm and forearm "before rotation" are related by the to-parent matrix $M_{5,p}$ and therefore the coordinates of v_5' "in the upper arm's space" are computed using $M_{5,p}$.

$$M_{5,l} = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$v_5' = M_{5,l} v_5 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}$$





$$v_5'' = M_{5,p} v_5' = \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 1 \end{pmatrix}$$

$$v_5'' = M_{5,p} M_{5,l} v_5 = \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 1 \end{pmatrix}$$

Forward Kinematics

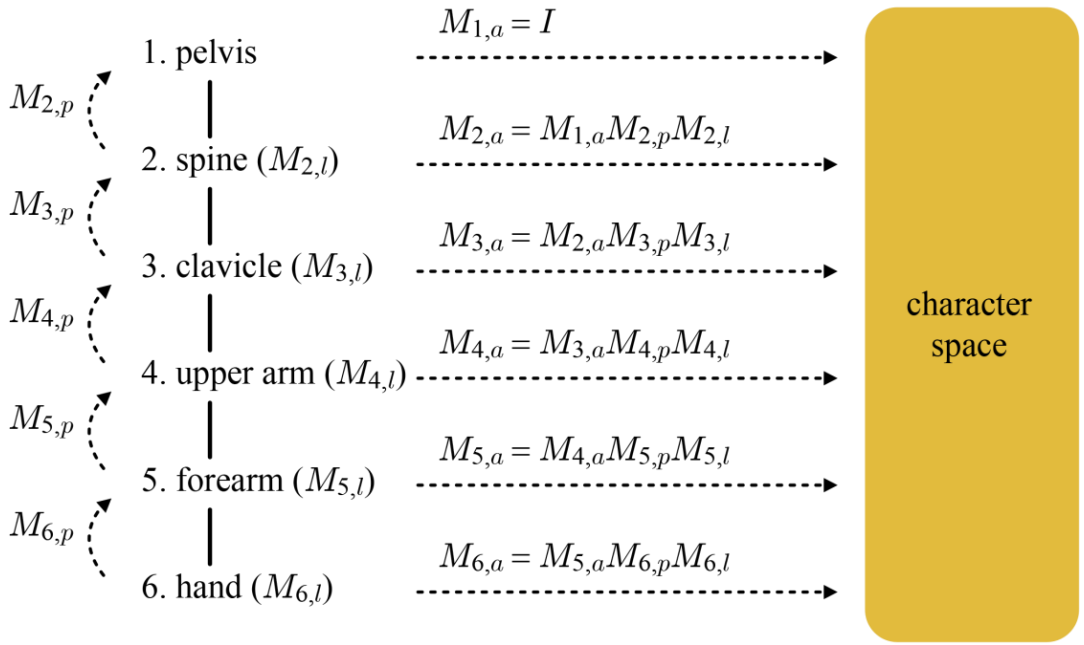
The transform from the bone space to the character space

- We obtained the coordinates of "animated v_5" in the upper arm's space.
- The upper arm can also be animated. Let $M_{4,a}$ denote the matrix that animates the upper arm's vertex and transforms it into the character space.
- Then, $M_{5,a}$ is defined in terms of $M_{4,a}$ together with $M_{5,p}M_{5,l}$.

$$M_{5,a} = M_{4,a}M_{5,p}M_{5,l}$$

- It is generalized as follows:

$$M_{i,a} = M_{i-1,a}M_{i,p}M_{i,l}$$

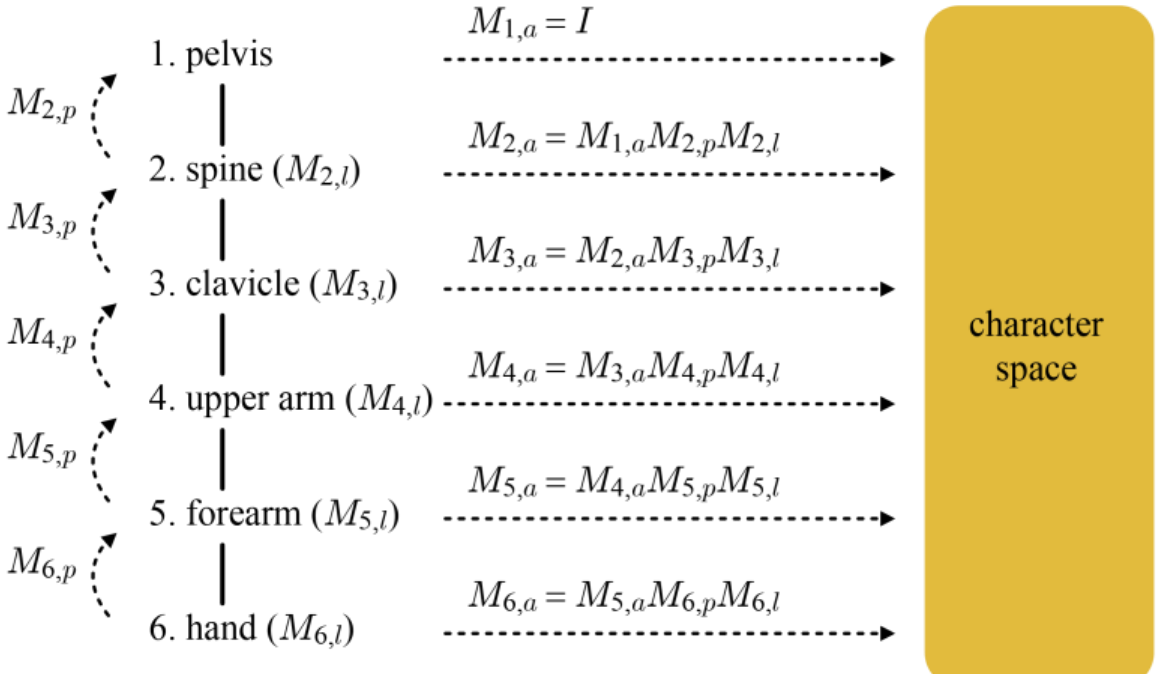


Forward Kinematics

The transform from the bone space to the character space

- Once the animator defines the animated pose of the *i*-th bone, $M_{i,l}$ is determined.
- $M_{i,p}$ was obtained from the default pose.
- So, computing $M_{i,a}$ simply requires $M_{i-1,a}$ to be computed in advance.
- Usually $M_{1,a} = I$ because the character space is identical to the pelvis' bone space. (On the other hand, the world transform of the character will be defined.)
- We can compute $M_{i,a}$ for every bone through the top-down traversal.

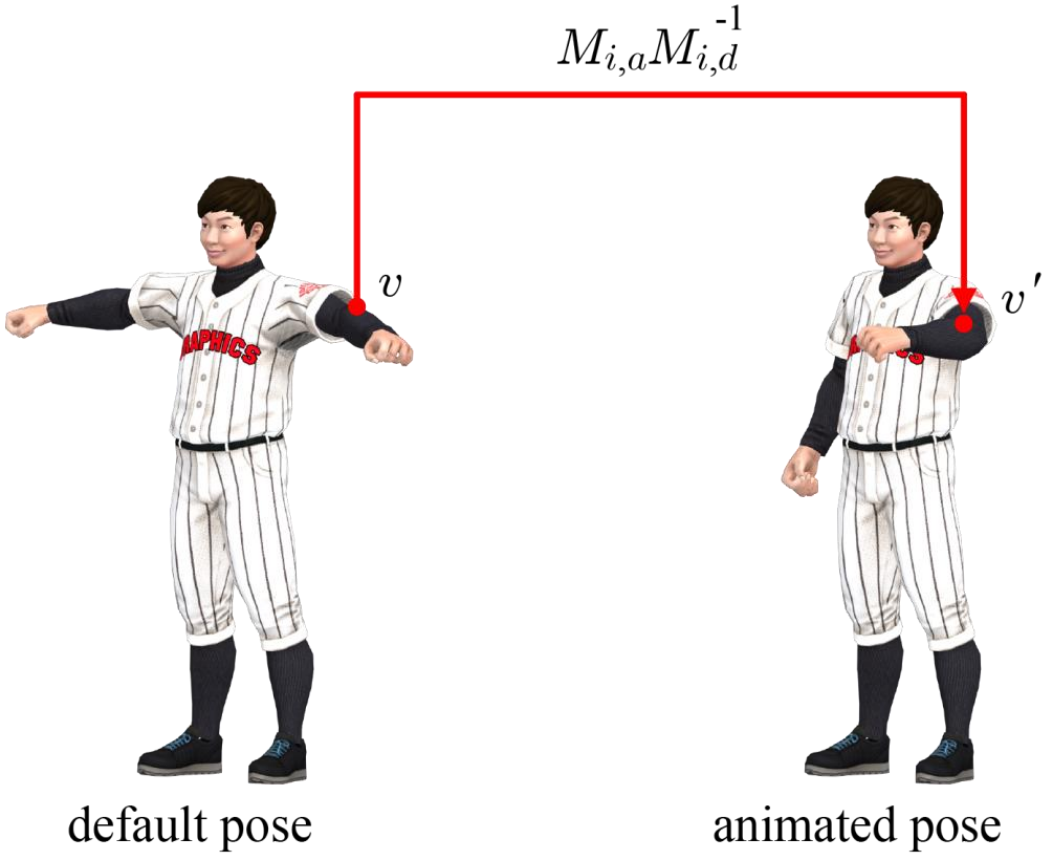
$$M_{i,a} = M_{i-1,a} M_{i,p} M_{i,l}$$



Forward Kinematics

Default pose vs. animated pose

- A character-space vertex in the "default pose" which we denote by v, is transformed to the bone space by $M_{i,d-1}$. Then, it is animated and transformed back to the character space by $M_{i,a}$ to define the character-space vertex v' in the "animated pose."



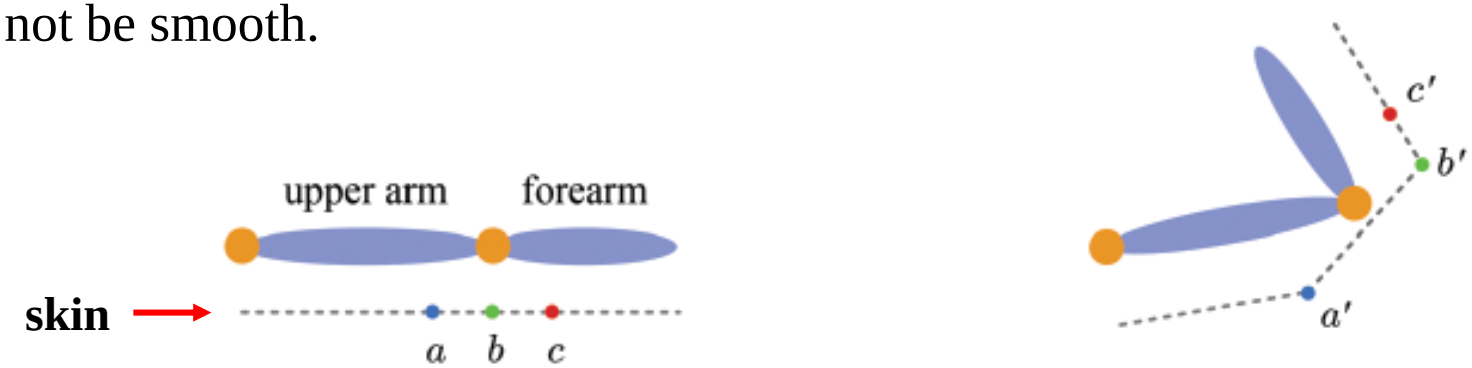
Skinning

Skin

- In character animation, the polygon mesh defined by the skeletal motion is often called a skin.
- Skin is often attached to the character skeleton.

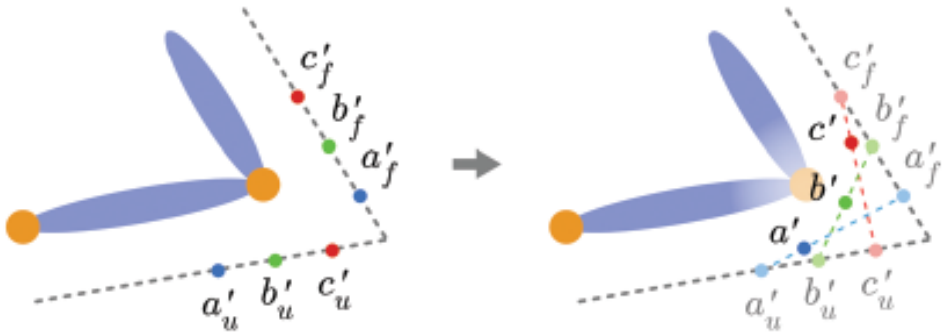
Skin deformation

- Suppose that vertex *a* belongs to the upper arm whereas *b* and *c* belong to the forearm. Then, the deformed skin may not be smooth.



- The problem can be alleviated by making multiple bones affect a vertex and then blending the results, instead of binding a vertex to a bone.

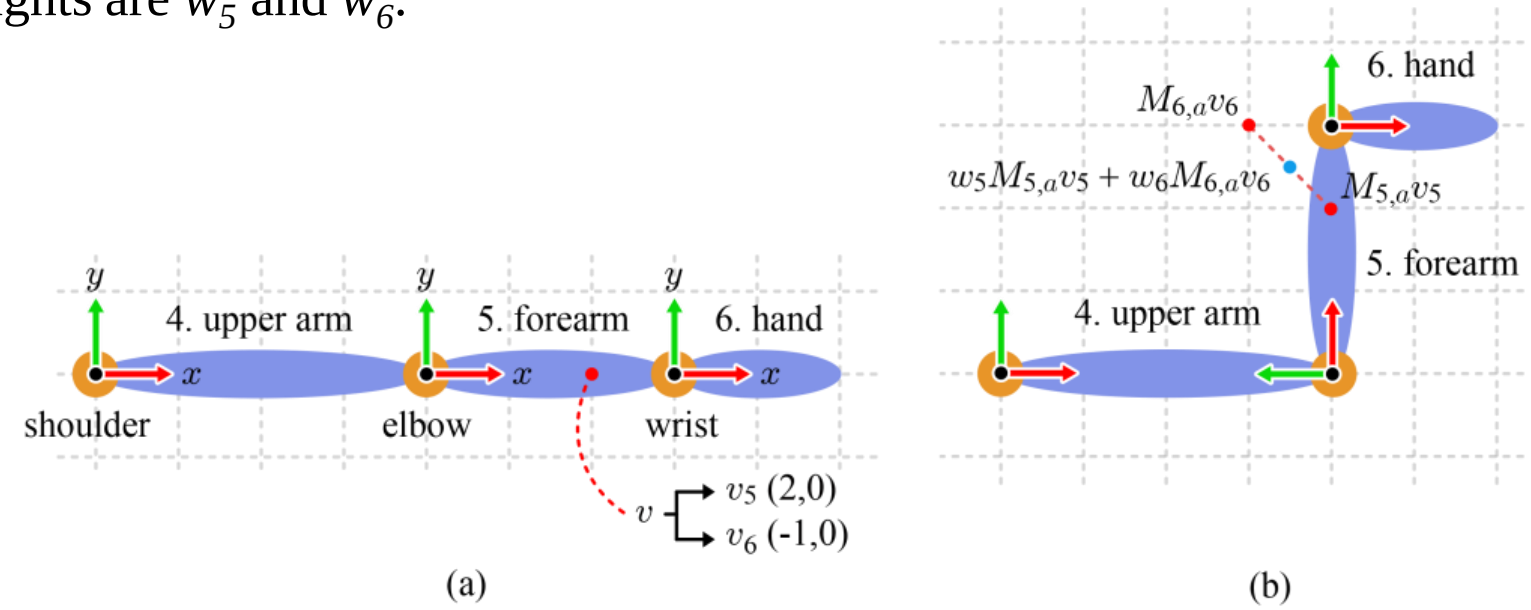
	upper arm	forearm
a	0.7	0.3
b	0.5	0.5
c	0.3	0.7



Skinning

Skin deformation

- Suppose that v is affected by both forearm and hand.
- Their weights are w_5 and w_6.



$$v_5 = M_{5,d}^{-1} v$$

$$v_6 = M_{6,d}^{-1} v$$

$$v' = w_5 M_{5,a} v_5 + w_6 M_{6,a} v_6$$

$$v' = w_5 M_{5,a} M_{5,d}^{-1} v + w_6 M_{6,a} M_{6,d}^{-1} v$$

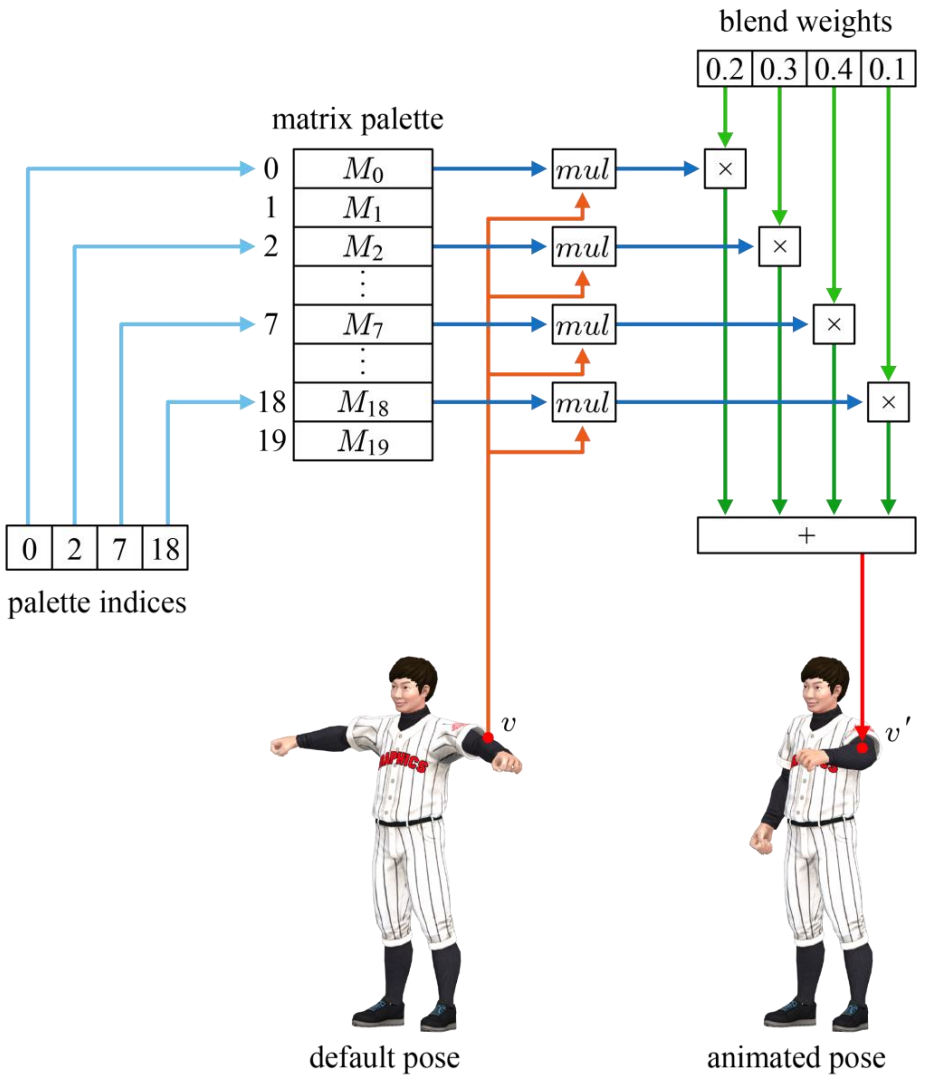
$$v' = \sum_{i=1}^{m} w_i M_{i,a} M_{i,d}^{-1} v$$

$$v' = \sum_{i=1}^{m} w_i M_i v$$

Skinning

Implementation

- In real-time 3D applications such as games, the number of bones affecting a vertex, *m*, is typically limited to four (or less) for programming efficiency.
- In general, the skinning algorithm is implemented by the vertex shader.
- If our character has 20 bones, we should update 20 instances of M_i for each frame.
- These instances are often stored in a table named the *matrix palette* and provided as a uniform variable.
- In contrast, the palette indices and the blend weights are stored in the vertex array.



Keyframe Animation

In $M_{i,a}M_{i,d}^{-1}$, which is to be updated for "every frame" of animation,

- $M_{i,d}^{-1}$ is computed *once* for the default pose and remains fixed, but
- $M_{i,a}$ for an in-between frame is updated by interpolating $M_{i,a}$ in the keyframes.

In $M_{i,a}$, which is defined as $M_{i\text{-}1,a}M_{i,p}M_{i,l}$,

- $M_{i,l}$ is (almost) a rotation, and
- $M_{i,p}$ is a space-change matrix composed of a translation and a rotation.

Let $M_{i,p,l}$ denote $M_{i,p}\,M_{i,l}$. It is in $[\boldsymbol{R}|\boldsymbol{t}]$.

- For a keyframe, $\boldsymbol{R}$ is stored as a quaternion, and $\boldsymbol{t}$ is stored as a vector. They form the key data of the i-th bone.
- For each in-between frame of animation, the skeleton hierarchy is traversed in a top-down fashion to compute

$M_{i,a}$ for each bone.

- The quaternions and translational vectors in the keyframes are interpolated.
- The interpolated quaternion is converted into a matrix.
- The interpolated translation vector fills the fourth column of the matrix.
- The matrix is combined with $M_{i\text{-}1,a}$ to complete $M_{i,a}$.

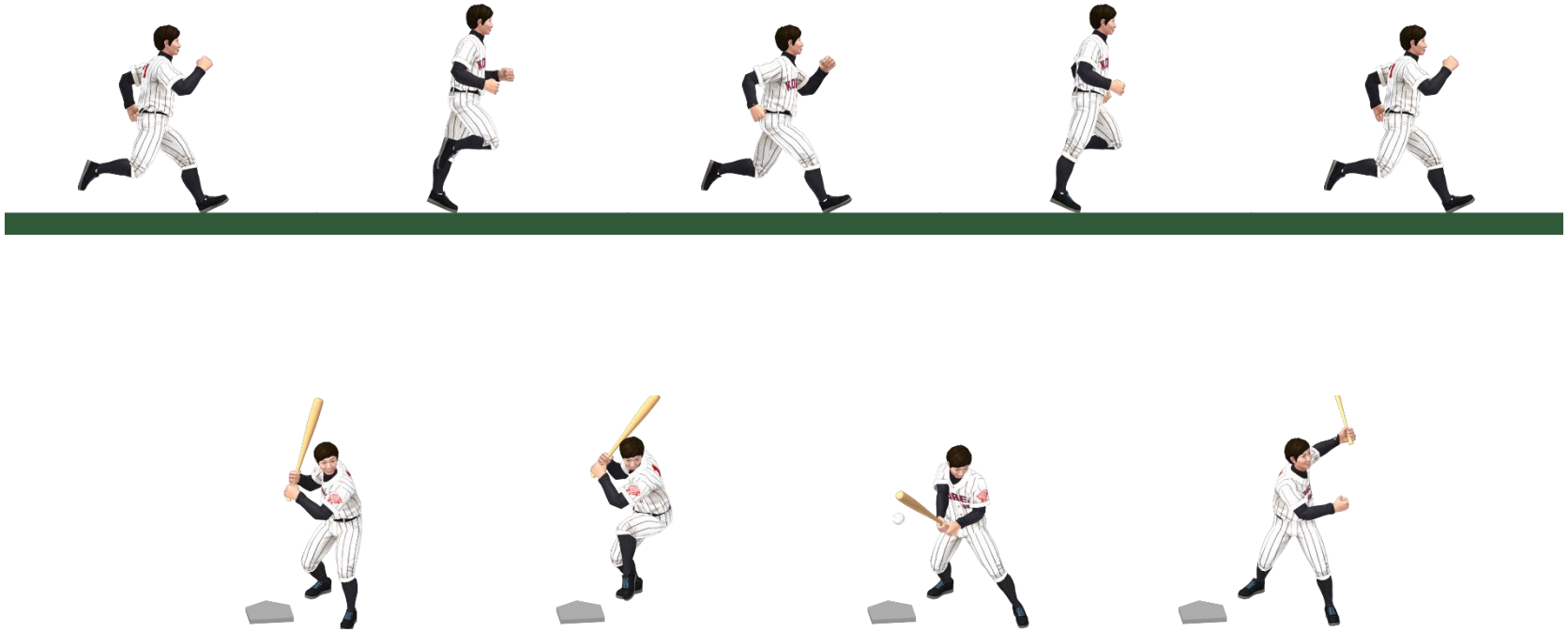
Keyframe Animation

```
for each bone // default pose
    load M_{i,d}^{-1}

for each frame
    for each bone i // animated pose
        interpolate key data
        compute M_{i,a}
        combine M_{i,a} with M_{i,d}^{-1} to define M_i
        store M_i into the matrix palette
    invoke vertex shader for skinning
```

top-down traversal

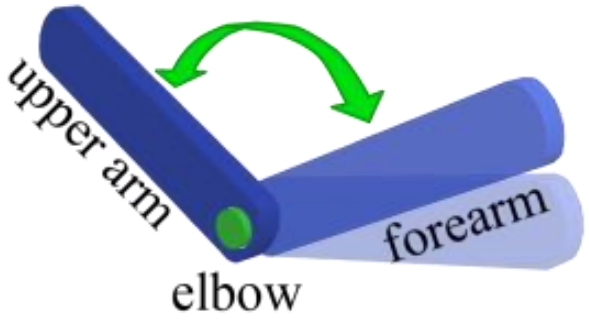
Keyframe Animation



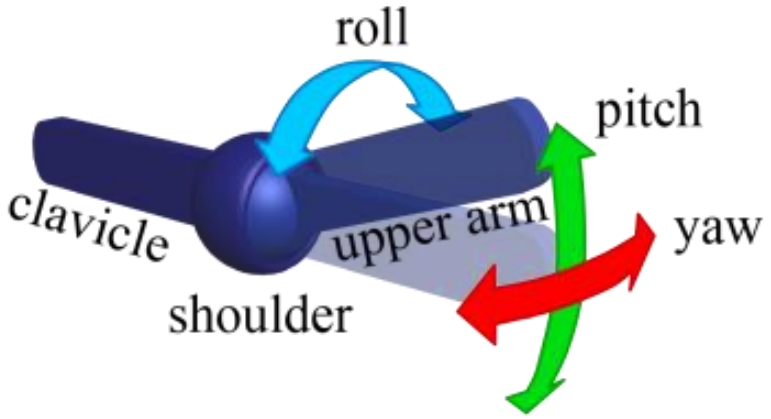
Inverse Kinematics

What is inverse kinematics?

- In robotics, an end effector refers to the device located at the end of a robotic arm.
- Forward kinematics computes the pose of the end effector as a function of the joint angles of the robotic arm.
- The reverse process is called inverse kinematics (IK). Given the desired pose of the end effector along with the initial pose of the robotic arm, the joint angles of the final pose are calculated.
- The number of independent variables defining the state of an object is called the degrees of freedom (DOF).



(a)

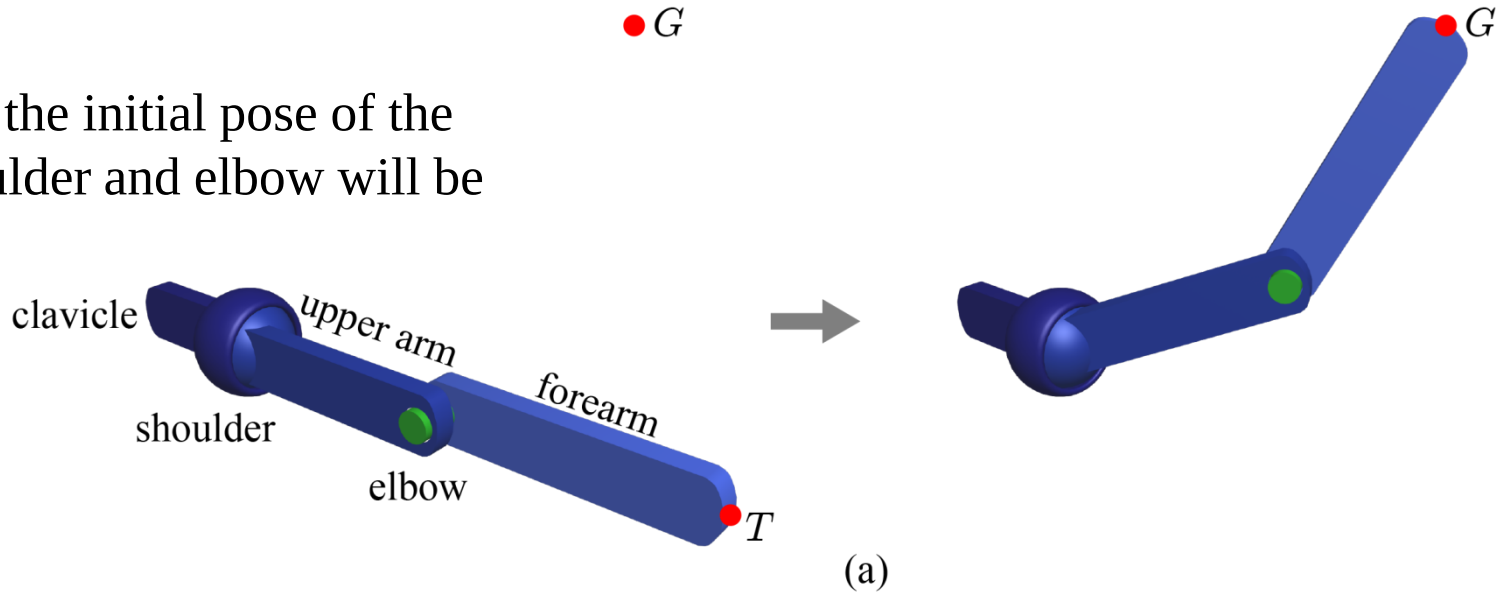


(b)

Inverse Kinematics

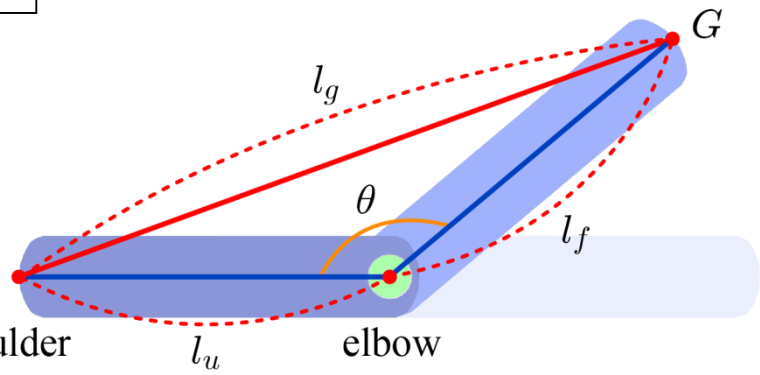
Analytic solution

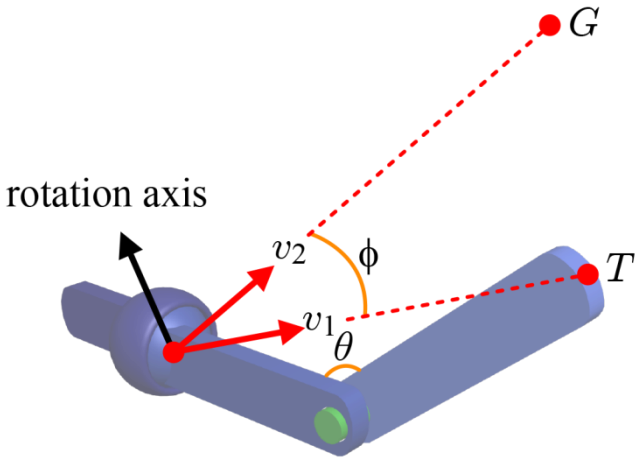
- Consider the two-joint arm.
- Given the goal position, G, and the initial pose of the arm, the joint angles of the shoulder and elbow will be automatically calculated.
- We can obtain the θ.
- Then, Φ can be computed by $\arccos(v_1 \cdot v_2)$



$$l_g^2 = l_u^2 + l_f^2 - 2l_u l_f cos\theta$$

$$\theta = acos\frac{l_u^2 + l_f^2 - l_g^2}{2l_u l_f}$$





Inverse Kinematics