8.1 Skinning/Enveloping

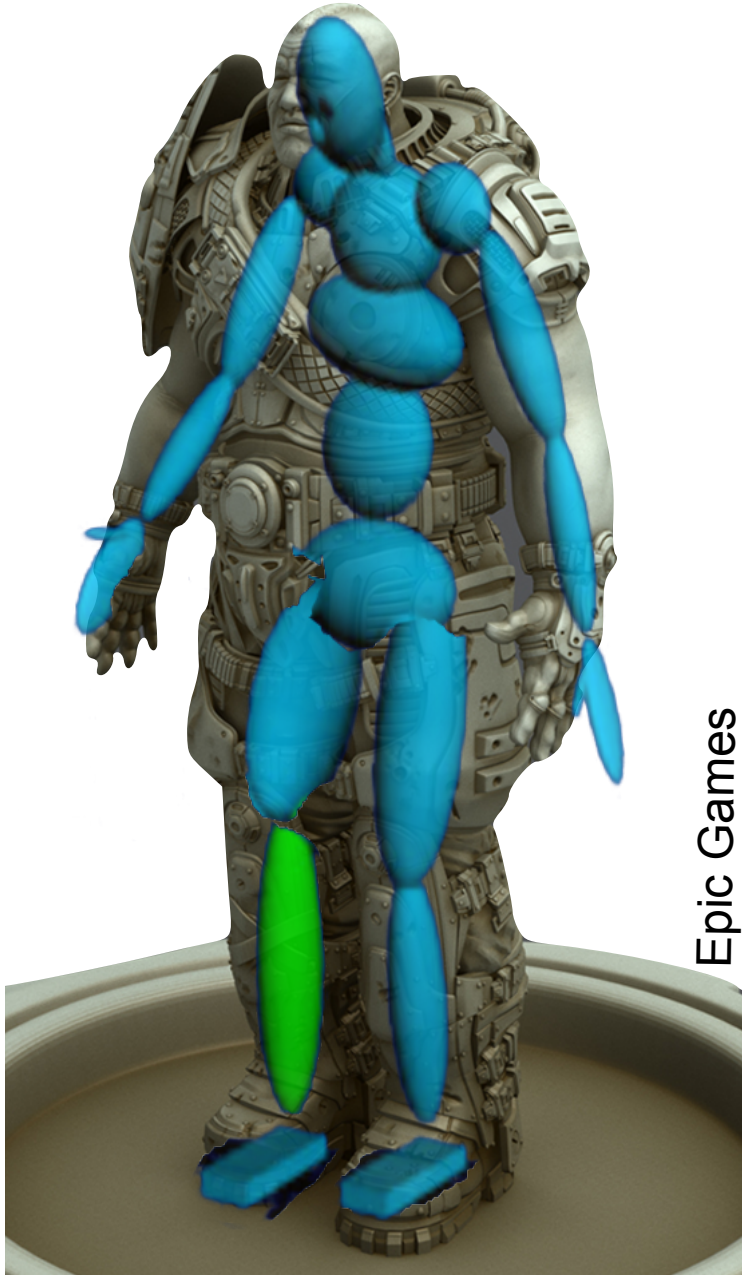
Epic Games / ign.com

In This Video

- Skinning: Animating articulated characters by matching a stick figure to a complex mesh
- Smoothly blending transformations

Skinning

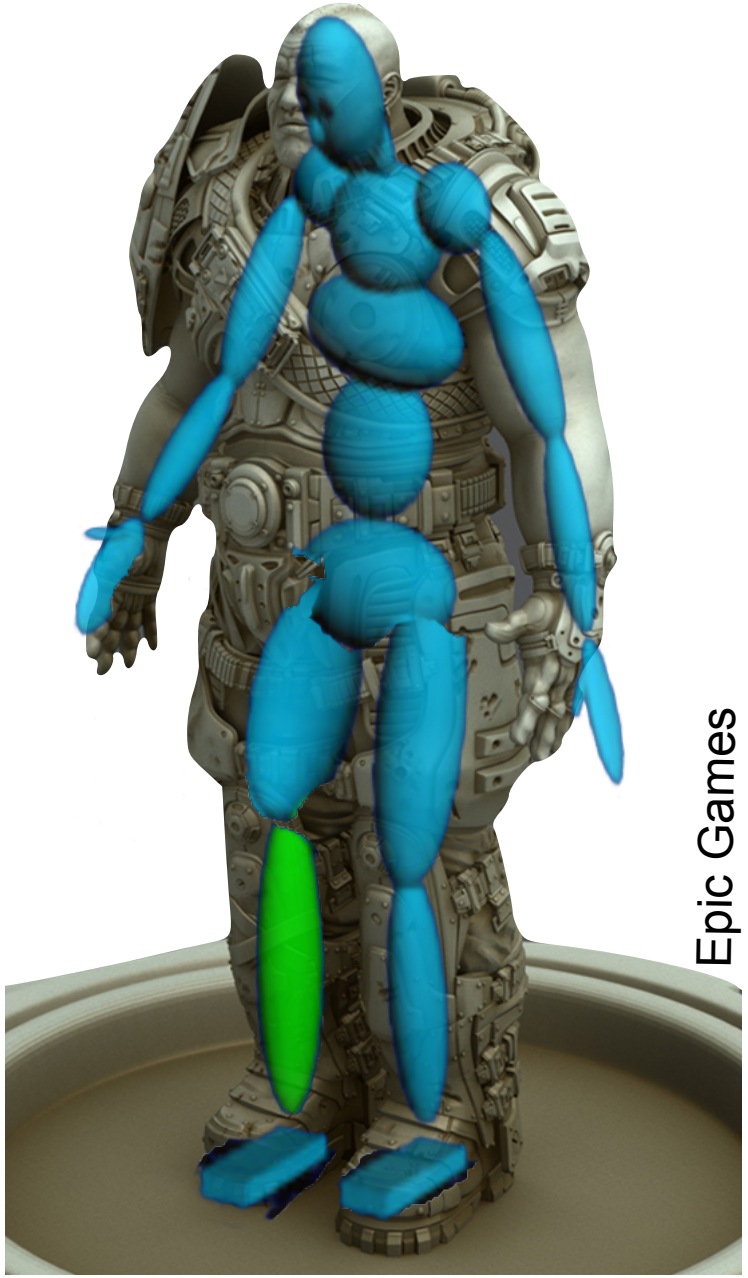
- We know how to animate a bone hierarchy
  - Change the joint angles, i.e., bone transformations, over time (keyframing)



Epic Games

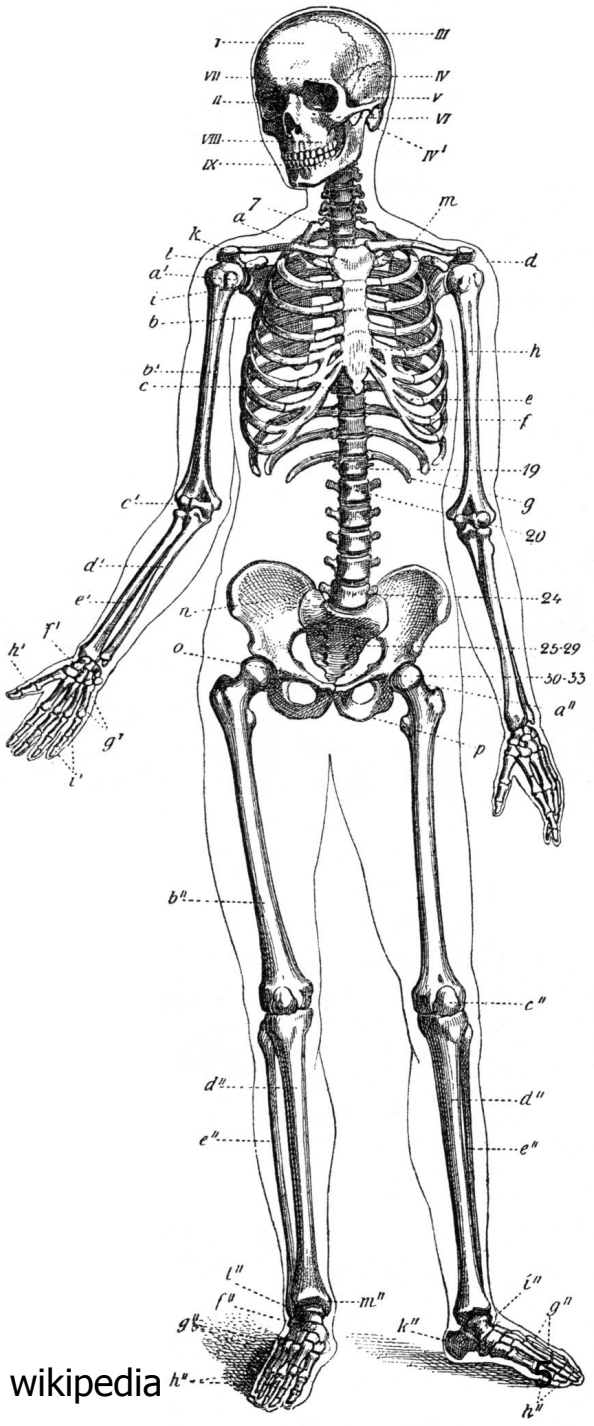
Skinning

- We know how to animate a bone hierarchy
  - Change the joint angles, i.e., bone transformations, over time (keyframing)
- Embed a skeleton into a detailed character mesh
- Bind skin vertices to bones
  - Animate skeleton, skin will move with it
  - But how?



Skinning

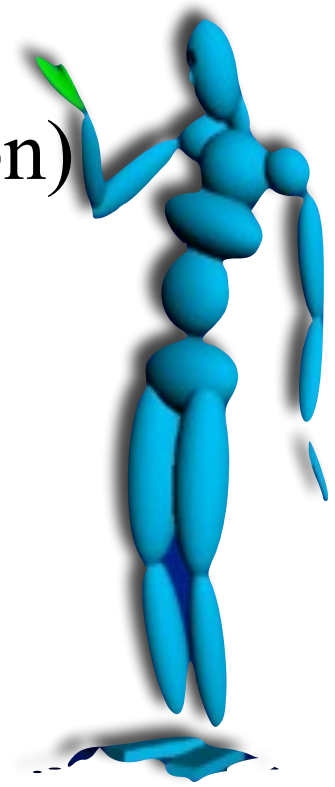
- Need to infer how skin deforms from bone transformations.
- Most popular technique: Skeletal Subspace Deformation (SSD), or simply *skinning*
  - Other aliases
    - vertex blending
    - matrix palette skinning
    - linear blend skinning



From wikipedia

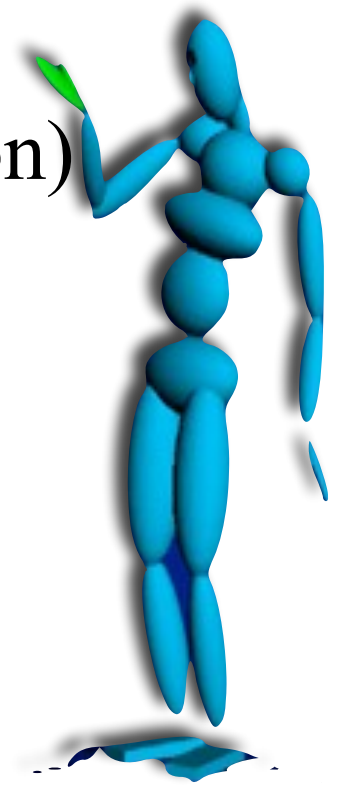
SSD / Skinning Mental Model

- Each bone “deforms the space around it” (rotation, translation)



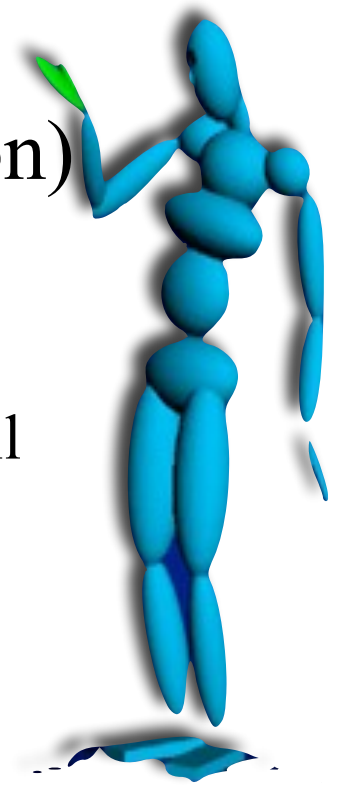
SSD / Skinning Mental Model

- Each bone “deforms the space around it” (rotation, translation)
  - What if we attach each vertex of the skin to a single bone?



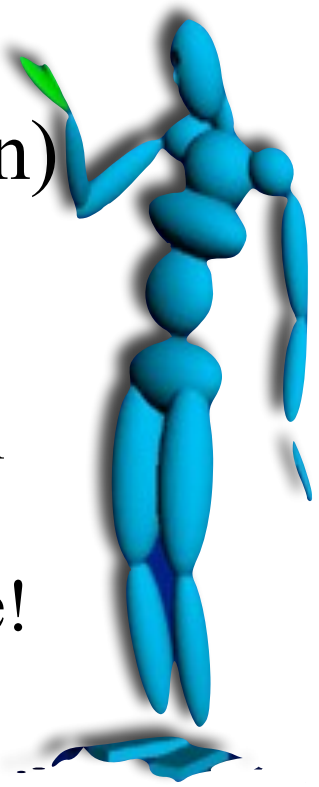
SSD / Skinning Mental Model

- Each bone “deforms the space around it” (rotation, translation)
  - What if we attach each vertex of the skin to a single bone?
    - Skin will be rigid, except at joints where it will stretch badly



SSD / Skinning Mental Model

- Each bone "deforms the space around it" (rotation, translation)
  - What if we attach each vertex of the skin to a single bone?
    - Skin will be rigid, except at joints where it will stretch badly
  - Let's attach a vertex to many bones at once!
    - In the middle of a limb, the skin points follow the bone rotation (near-)rigidly
    - At a joint, skin is deformed according to a "weighted combination" of the bones



Examples

Colored triangles are attached to 1 bone

Black triangles are attached to more than 1

Note how they are near joints

James & Twigg 2005

Examples

James & Twigg 2005

Colored triangles are attached to 1 bone

Black triangles are attached to more than 1

Note how they are near joints

Linear Blend Skinning: Basic Idea

- **Step 1**: Transform each vertex $\mathbf{p}_i$ with each bone as if it was tied to it rigidly.
- **Step 2**: Then blend the results using weights that associate vertices to bones.

- Next up: how!

Vertex Weights

- We'll assign a weight w_{ij} for each vertex $\mathbf{p}_i$ for each bone $\mathbf{B}_j$.
  - w_{ij} = "How much vertex *i* should move with bone *j*"
  - $w_{ij} = 1$ means $\mathbf{p}_i$ is rigidly attached to $\mathbf{B}_j$.

Vertex Weights

- We'll assign a weight w_{ij} for each vertex $\mathbf{p}_i$ for each bone $\mathbf{B}_j$.
  - w_{ij} = "How much vertex *i* should move with bone *j*"
  - $w_{ij} = 1$ means $\mathbf{p}_i$ is rigidly attached to $\mathbf{B}_j$.

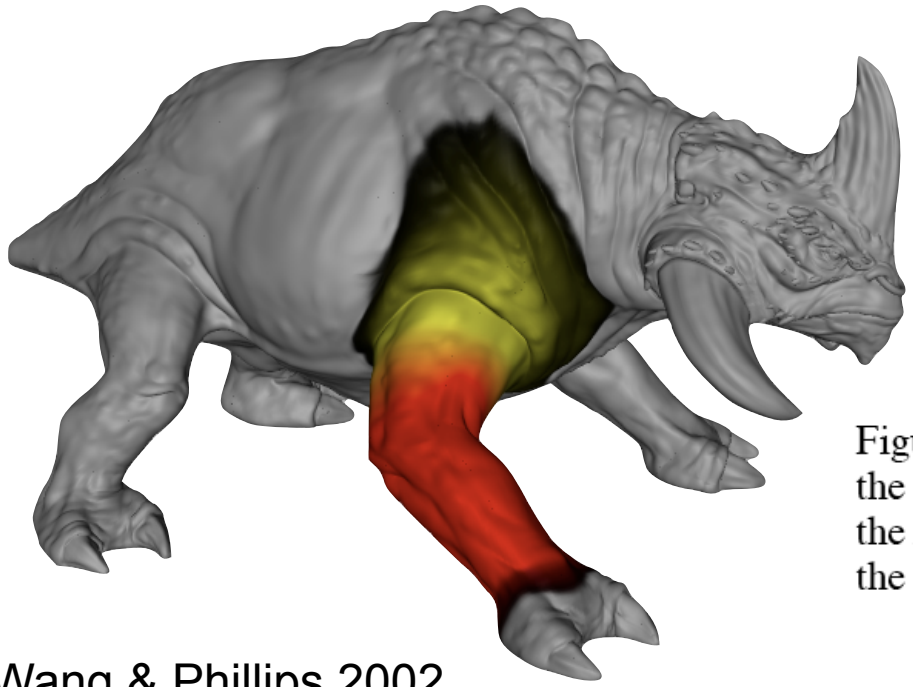


Figure 4: Color coded influences maps from two bones on the animal creature. The yellow area for the upper leg and the red area for the lower leg. The darker the area, the smaller the influence.

Wang & Phillips 2002

From Assignment 3

Figure 3: Left: Bone weights visualized on the vertices in bind pose. Middle: bind pose mesh lit using simple directional lighting. Right: Deformed skin, waist rotated to y=0.70.

Vertex Weights

- We'll assign a weight w_{ij} for each vertex $\mathbf{p}_i$ for each bone $\mathbf{B}_j$.
  - w_{ij} = "How much vertex *i* should move with bone *j*"
  - $w_{ij} = 1$ means $\mathbf{p}_i$ is rigidly attached to $\mathbf{B}_j$.
- Weight properties
  - Usually want weights to be non-negative Why?

Vertex Weights

- We'll assign a weight w_{ij} for each vertex $\mathbf{p}_i$ for each bone $\mathbf{B}_j$.
  - w_{ij} = "How much vertex *i* should move with bone *j*"
  - $w_{ij} = 1$ means $\mathbf{p}_i$ is rigidly attached to $\mathbf{B}_j$.
- Weight properties
  - Usually want weights to be non-negative
  - Also, *always* want the sum over all bones to be 1 for each vertex, i.e., $\sum_j w_{ij} = 1 \quad \forall i$ Why?

Vertex Weights

- We'll assign a weight w_{ij} for each vertex $\mathbf{p}_i$ for each bone $\mathbf{B}_j$.
  - w_{ij} = "How much vertex *i* should move with bone *j*"
  - $w_{ij} = 1$ means $\mathbf{p}_i$ is rigidly attached to $\mathbf{B}_j$.
- Weight properties
  - Usually want weights to be non-negative
  - Also, *always* want the sum over all bones to be 1 for each vertex, i.e., $\sum_j w_{ij} = 1 \quad \forall i$
    - This means translation independence.
    - (Again, a *partition of unity* – remember spline bases?)

Vertex Weights cont'd

- We'll limit the number of bones *N* that can influence a single vertex
  - *N*=8 bones/vertex is a usual choice
  - **Why?**

Vertex Weights cont'd

- We'll limit the number of bones N that can influence a single vertex
  - N=8 bones/vertex is a usual choice
  - Why? You most often don't need very many.
  - Also, storage space is an issue.
  - In practice, well store N (bone index j, weight w_{ij}) pairs per vertex.

How to compute vertex positions?

Epic Games

Basic Idea Recap

- **Step 1**: Transform each vertex $\mathbf{p}_i$ with each bone as if it was tied to it rigidly.
- **Step 2**: Then blend the results using the weights.

Computing Vertex Positions

- **Step 1**: Transform each vertex $\mathbf{p}_i$ with each bone as if it was tied to it rigidly.
- **Step 2**: Then blend the results using the weights.

“

$$\boldsymbol{p}'_{ij} = \boldsymbol{T}_j \, \boldsymbol{p}_i$$

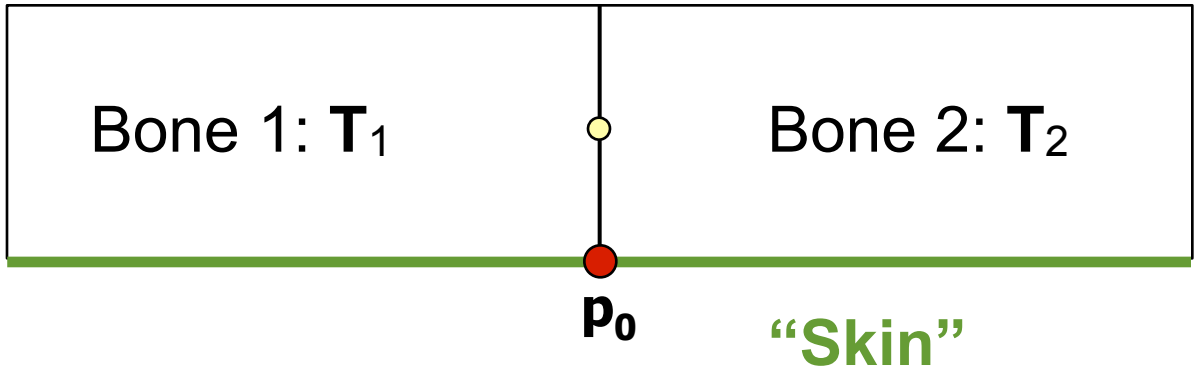
$$\boldsymbol{p}'_i = \sum_j w_{ij} \boldsymbol{p}'_{ij}$$

”

$\mathbf{p}'_{ij}$ is the vertex i transformed using bone j.
$\mathbf{T}_j$ is the current transformation of bone j.
$\mathbf{p}'_i$ is the new skinned position of vertex i.

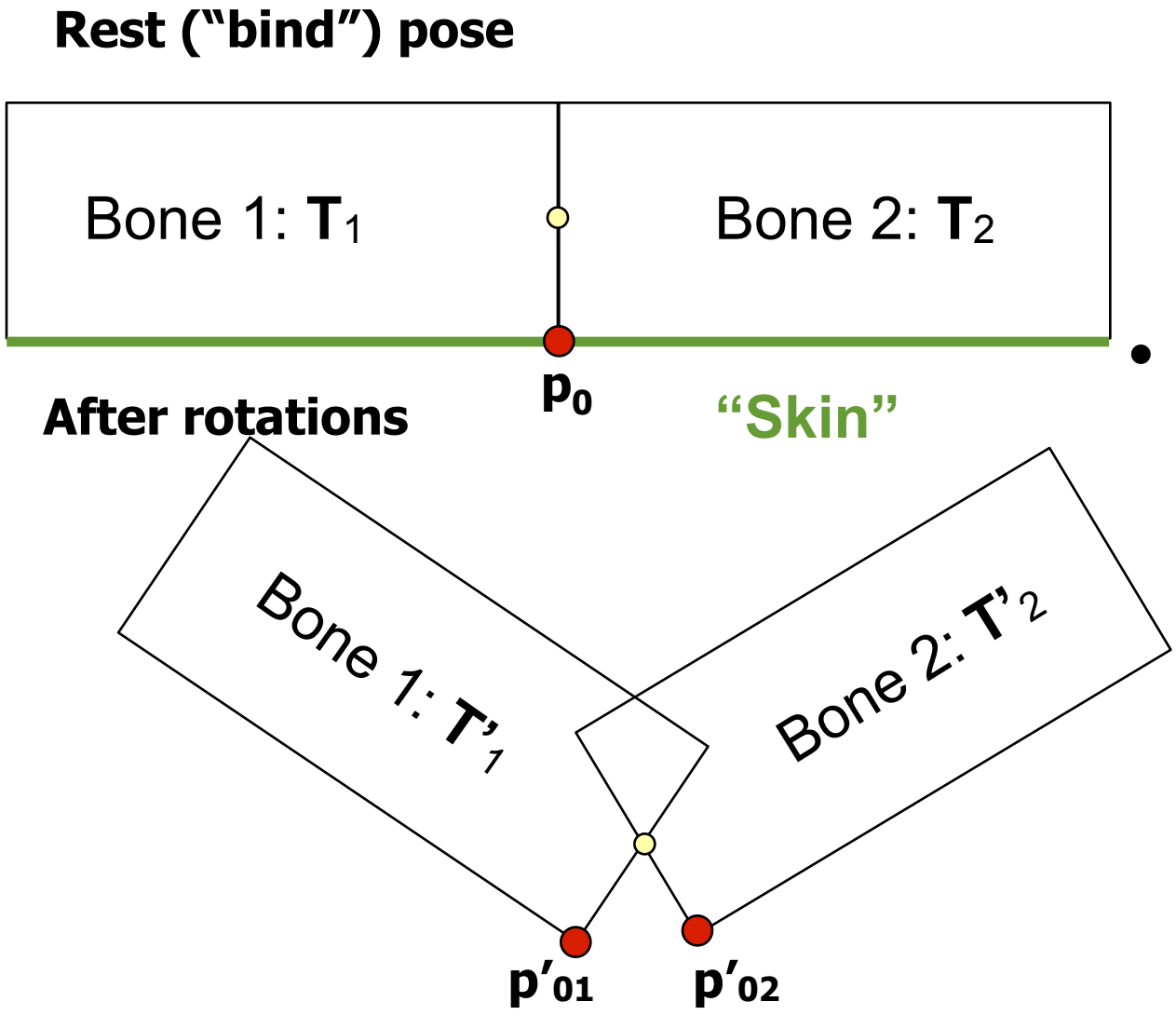
Computing Vertex Positions

Rest ("bind") pose



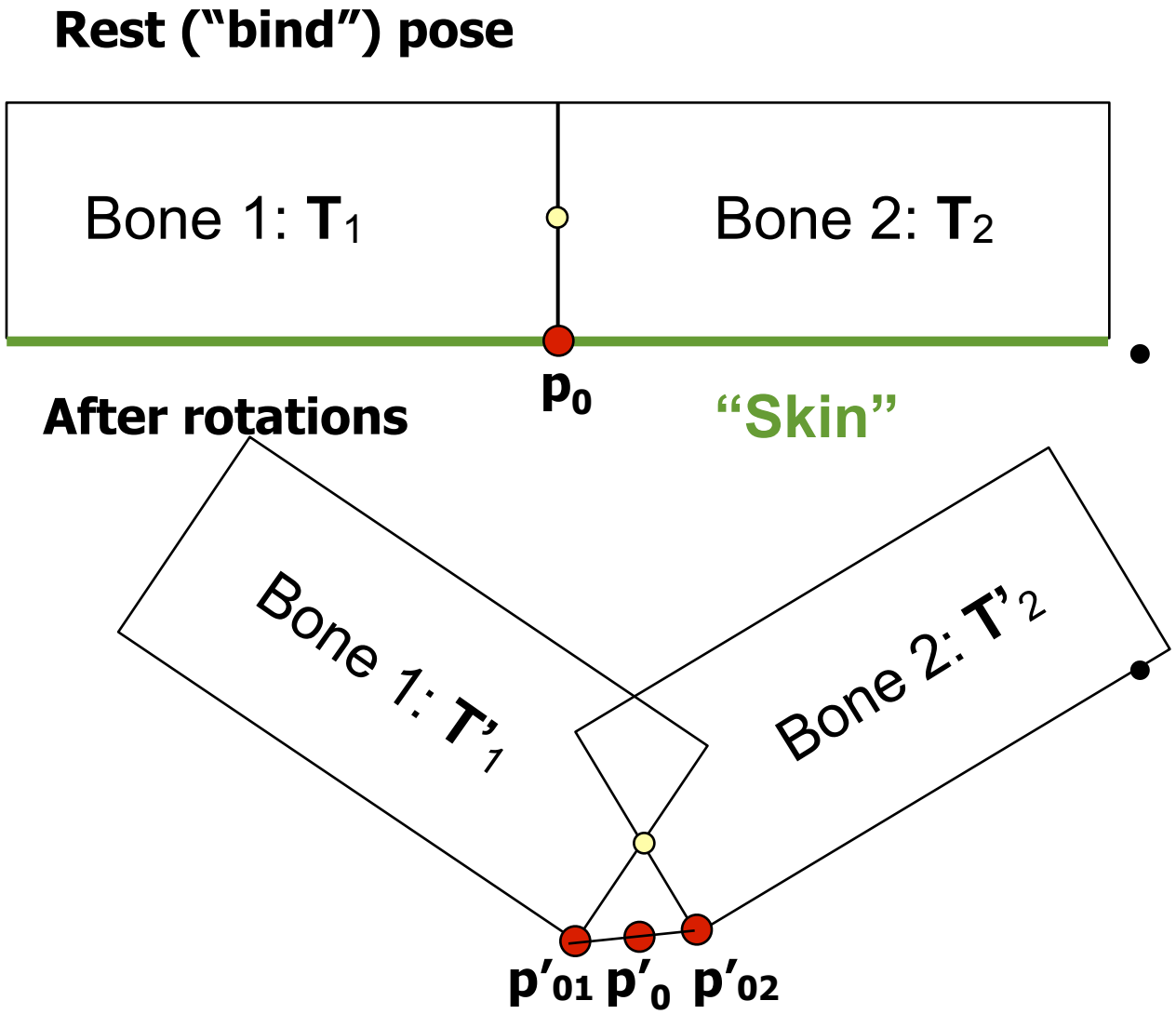
- Vertex $\mathbf{p}_0$ has weights $w_{01}=0.5$, $w_{02}=0.5$

Computing Vertex Positions



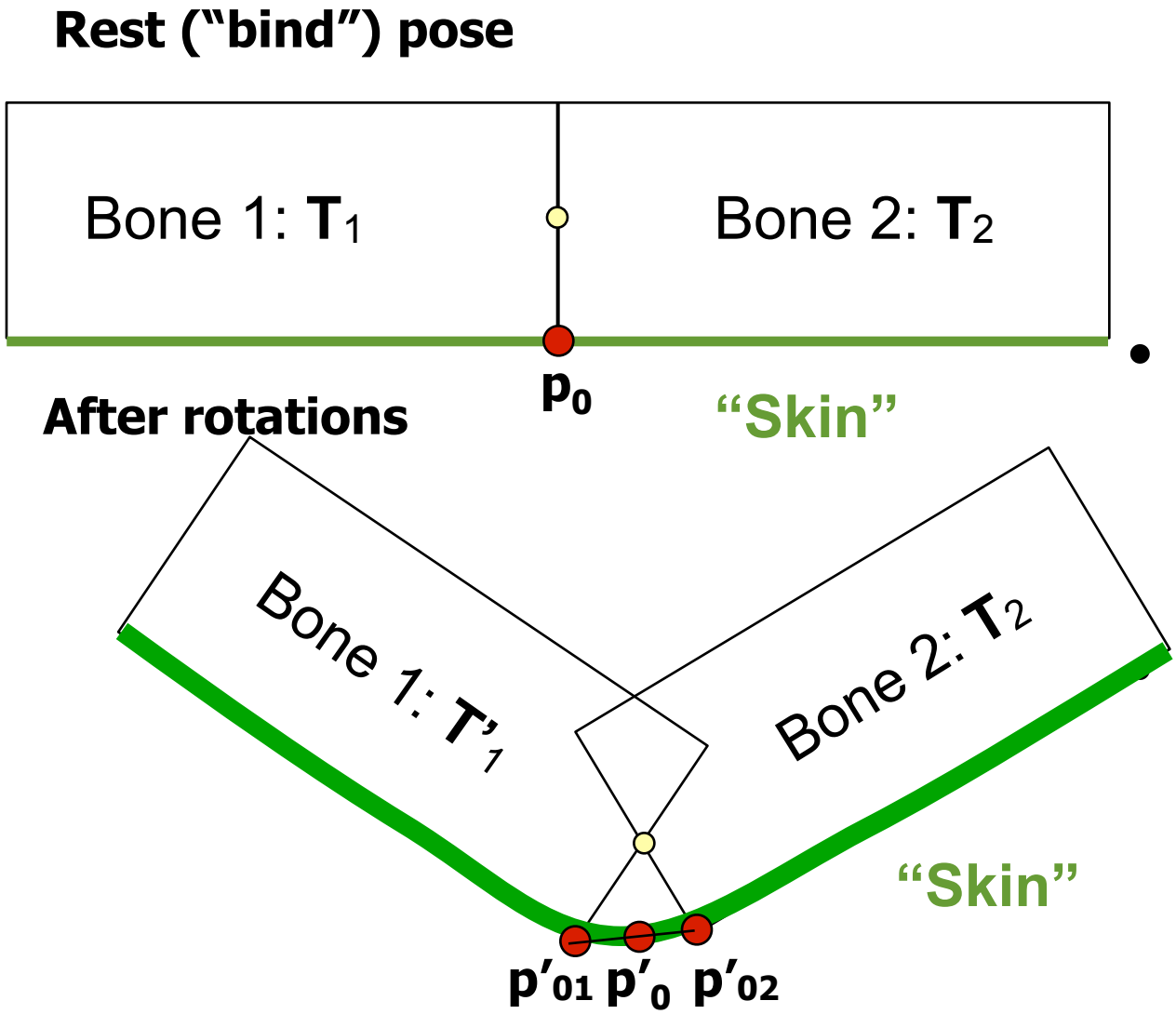
- Vertex $\mathbf{p}_0$ has weights $w_{01}=0.5$, $w_{02}=0.5$
- Transform by $\mathbf{T'}_1$ and $\mathbf{T'}_2$ yields $\mathbf{p'}_{01}$, $\mathbf{p'}_{02}$

Computing Vertex Positions



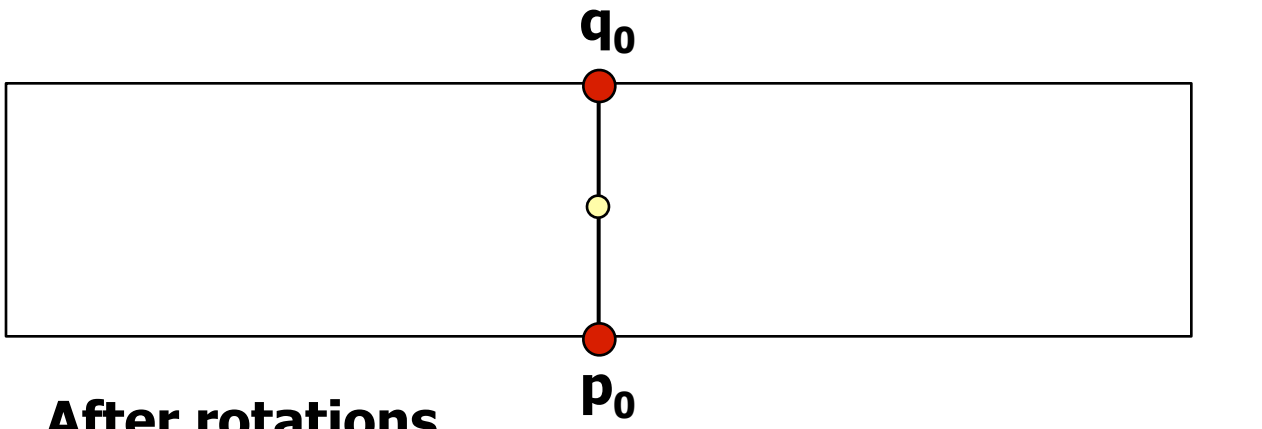
- Vertex $\mathbf{p}_0$ has weights w_{01}=0.5, w_{02}=0.5
- Transform by $\mathbf{T'}_1$ and $\mathbf{T'}_2$ yields $\mathbf{p'}_{01}$, $\mathbf{p'}_{02}$
- the new position is $\mathbf{p'}_0$= 0.5*$\mathbf{p'}_1$ + 0.5*$\mathbf{p'}_2$

Computing Vertex Positions

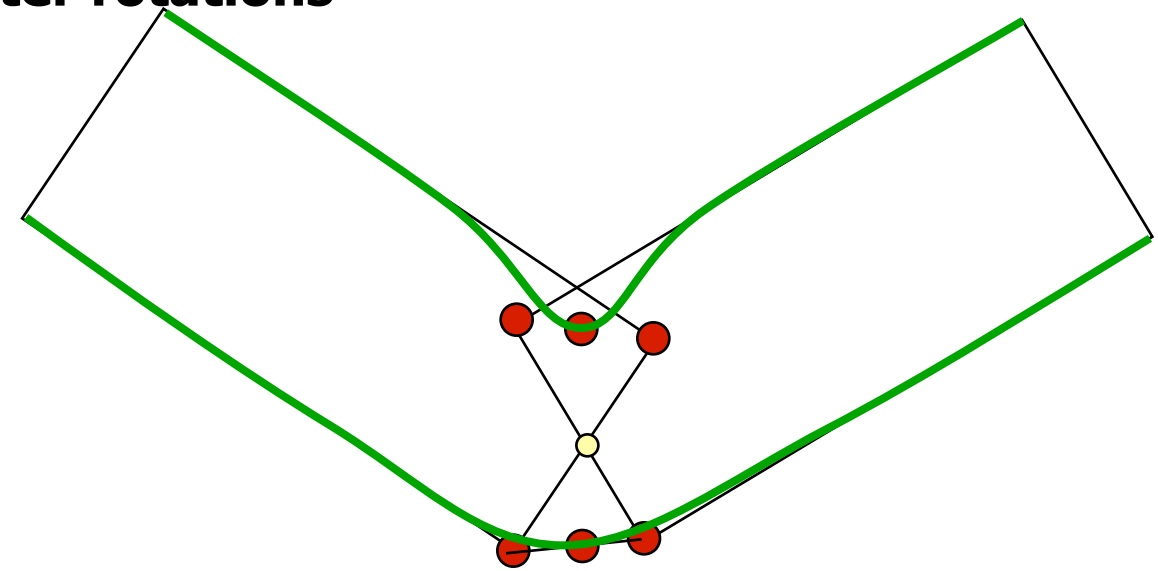


- Vertex $\mathbf{p}_0$ has weights w_{01}=0.5, w_{02}=0.5
- Transform by $\mathbf{T'}_1$ and $\mathbf{T'}_2$ yields $\mathbf{p'}_{01}$, $\mathbf{p'}_{02}$
- the new position is $\mathbf{p'}_0$= 0.5*$\mathbf{p'}_1$ + 0.5*$\mathbf{p'}_2$

SSD is Not Perfect



After rotations



On To Part 2…

Remedy Entertainment, Microsoft Game Studios, IGN.com