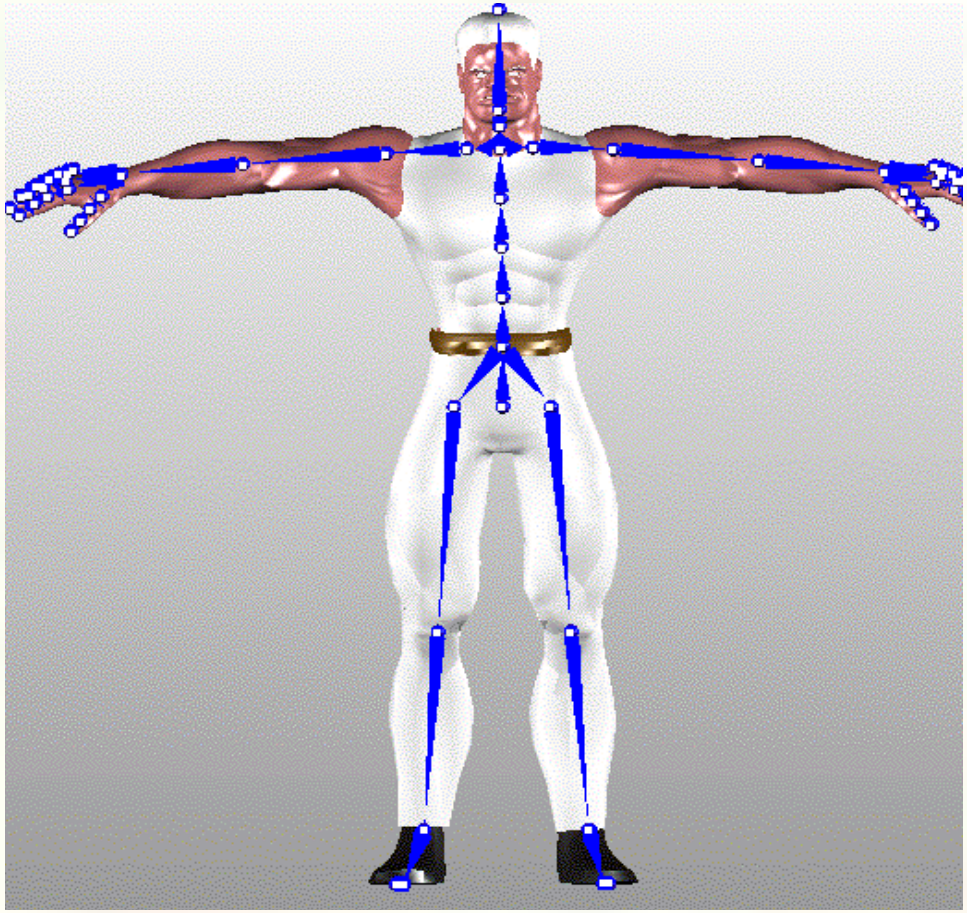
Character Animation and Skinning

Objectives

- Introduce the basics of character animation
- Introduce skinning
- Introduce basic linear blend skinning

Character Animation

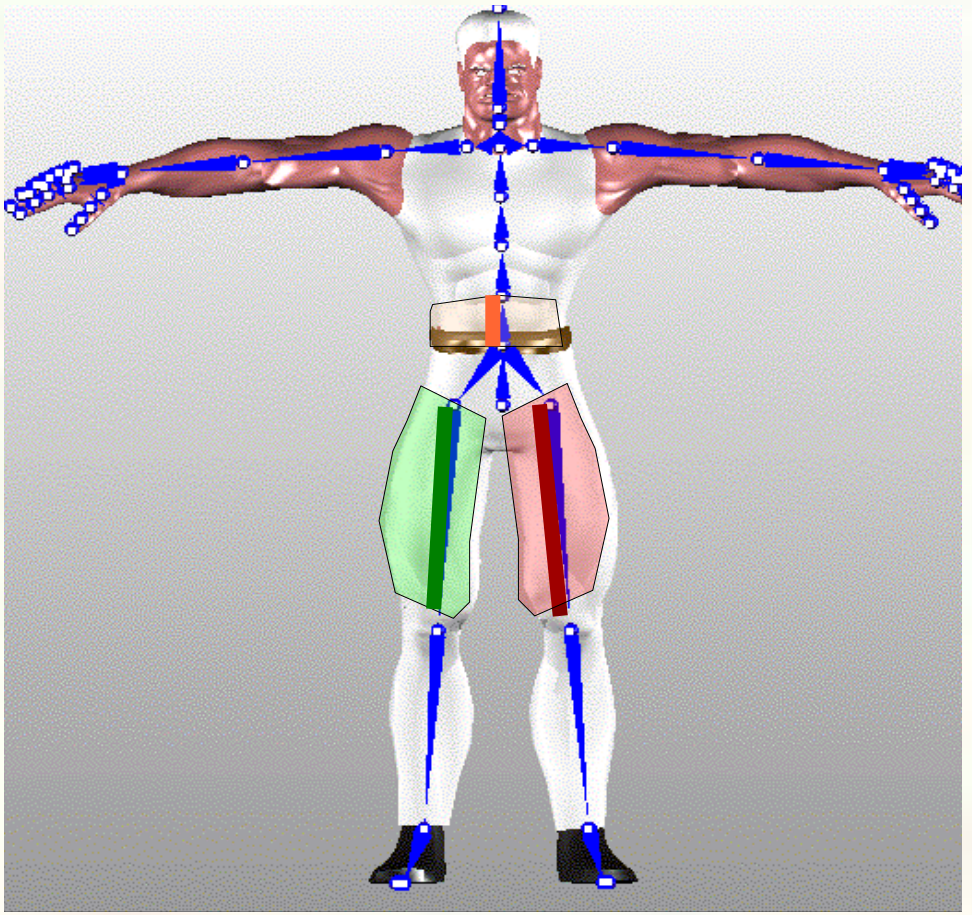
- Skeletons and skin
  - **skeleton** – a hierarchy of bones or joints
  - note arrows pointing from parent to child joint
  - **skin** – the polygon mesh defining the body surface



http://www.okino.com/conv/skinning.htm

Binding

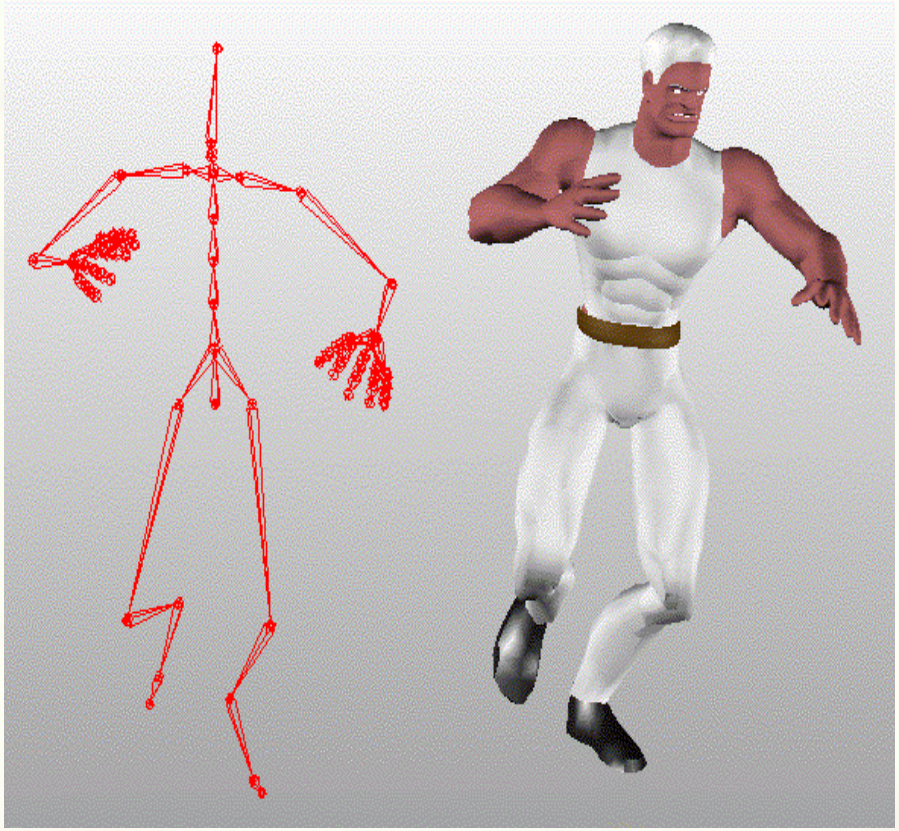
- Define transform between joint and skin spaces in **rest** or **bind** pose
- Associate skin vertices to subset of the joints



http://www.okino.com/conv/skinning.htm

Animation

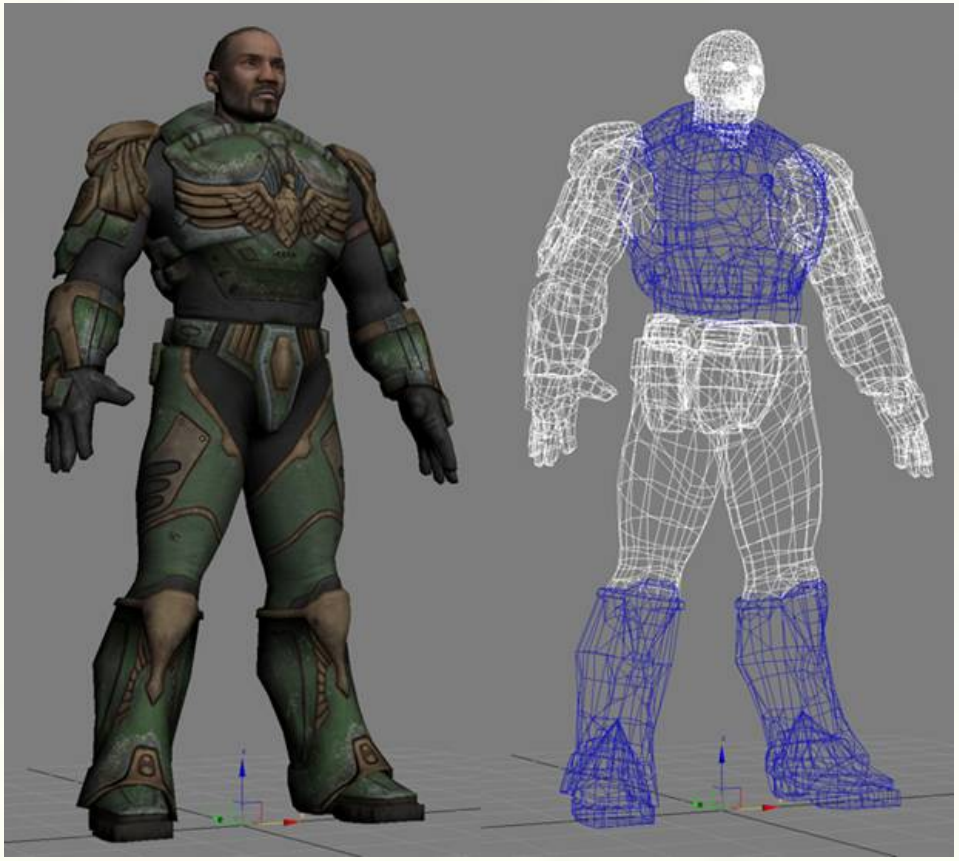
- Move the joints and the skin moves with them
- This deforms the mesh from its rest position



http://www.okino.com/conv/skinning.htm

Skin

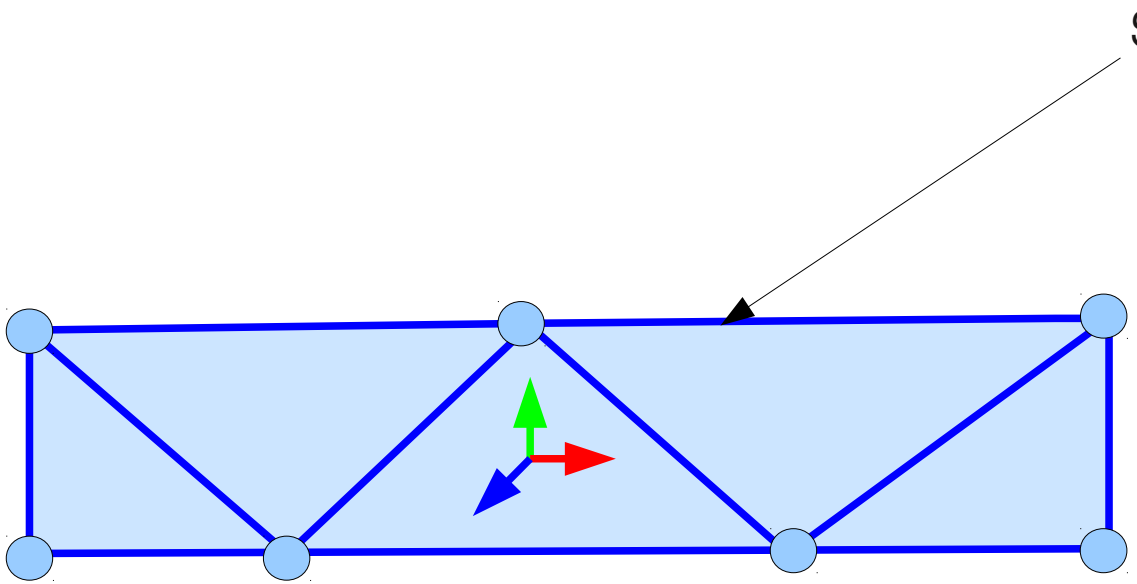
- Skin is a set of polygonal meshes
- A mesh is a collection of (connected) polygons



http://udn.epicgames.com/Three/UT3CustomCharacters.html

Skin

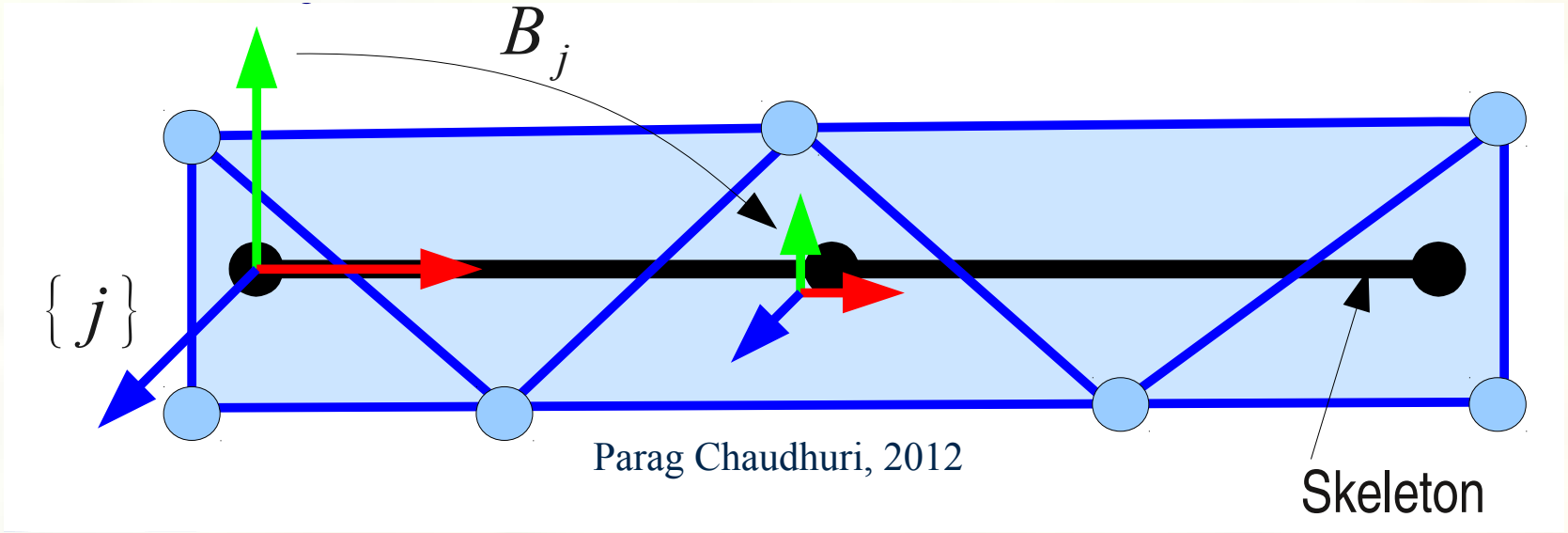
A skin mesh is defined in its own local frame



Parag Chaudhuri, 2012

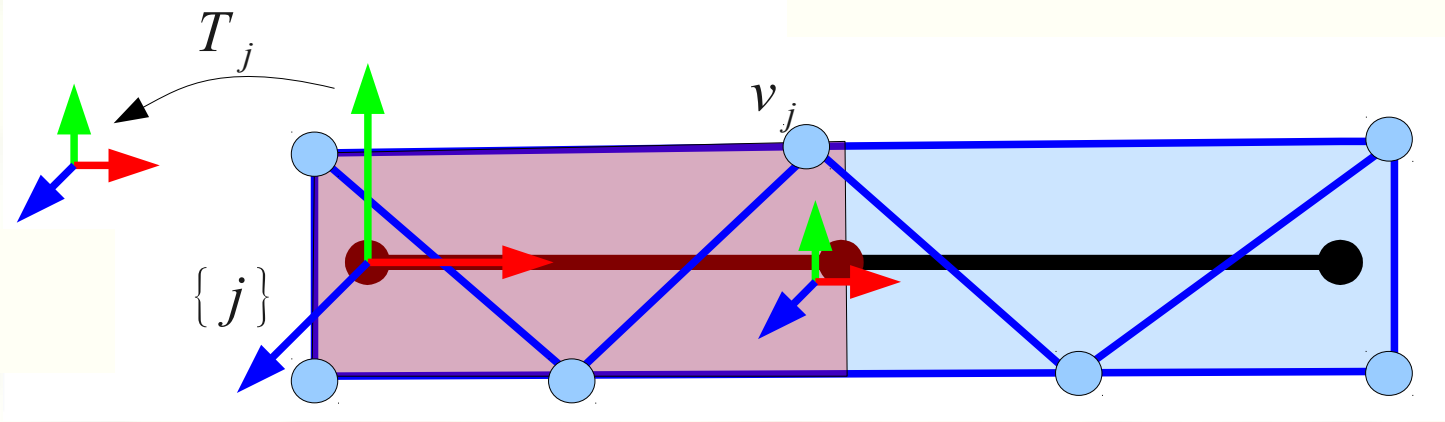
Binding

- Each joint (bone) has its own local frame
- Let B_j be the transformation from local joint frame j to the skin mesh local frame in the binding pose.
- B_j is represented by a **binding matrix**



Rigid skinning – basic idea

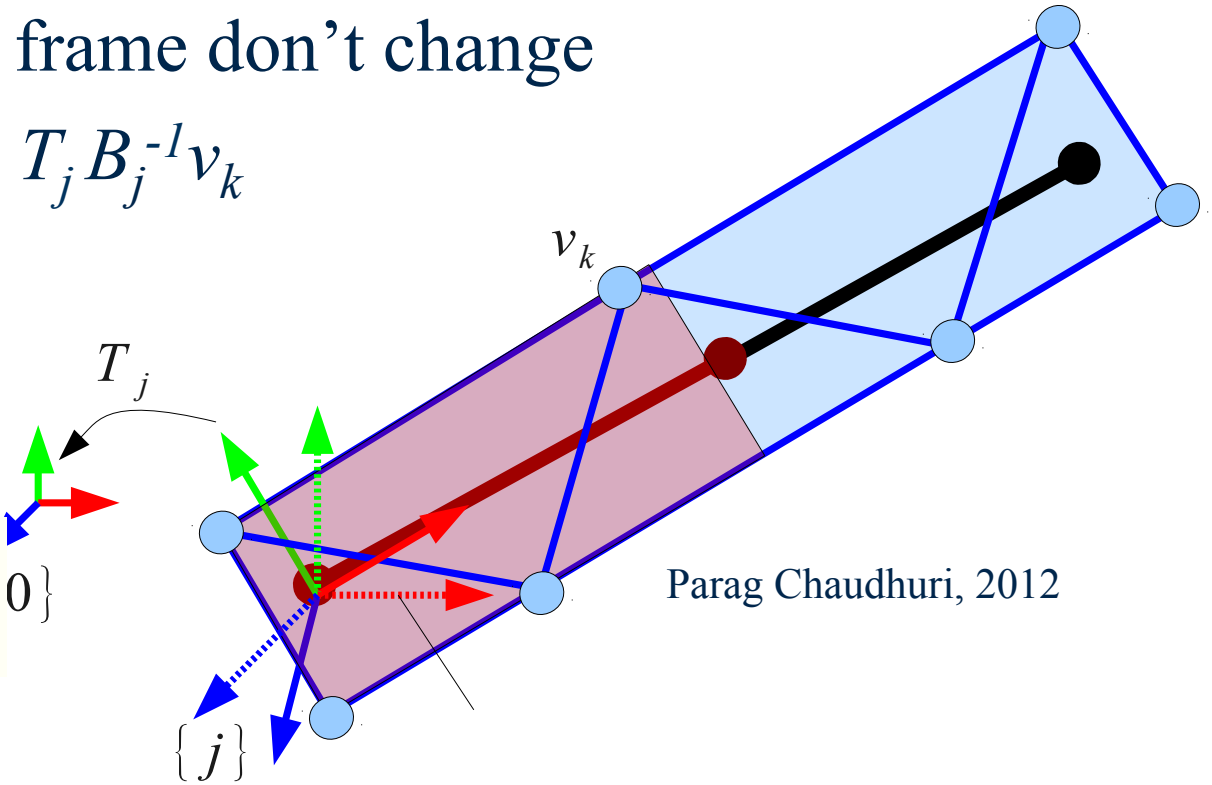
- Associate a group of vertices to a single joint j
- Let T_j be the transformation from joint j local space to world space
- Then the skin vertex transform to world space for vertices v_k associated with joint j is $v'_k = T_j B_j^{-1} v_k$



Parag Chaudhuri, 2012

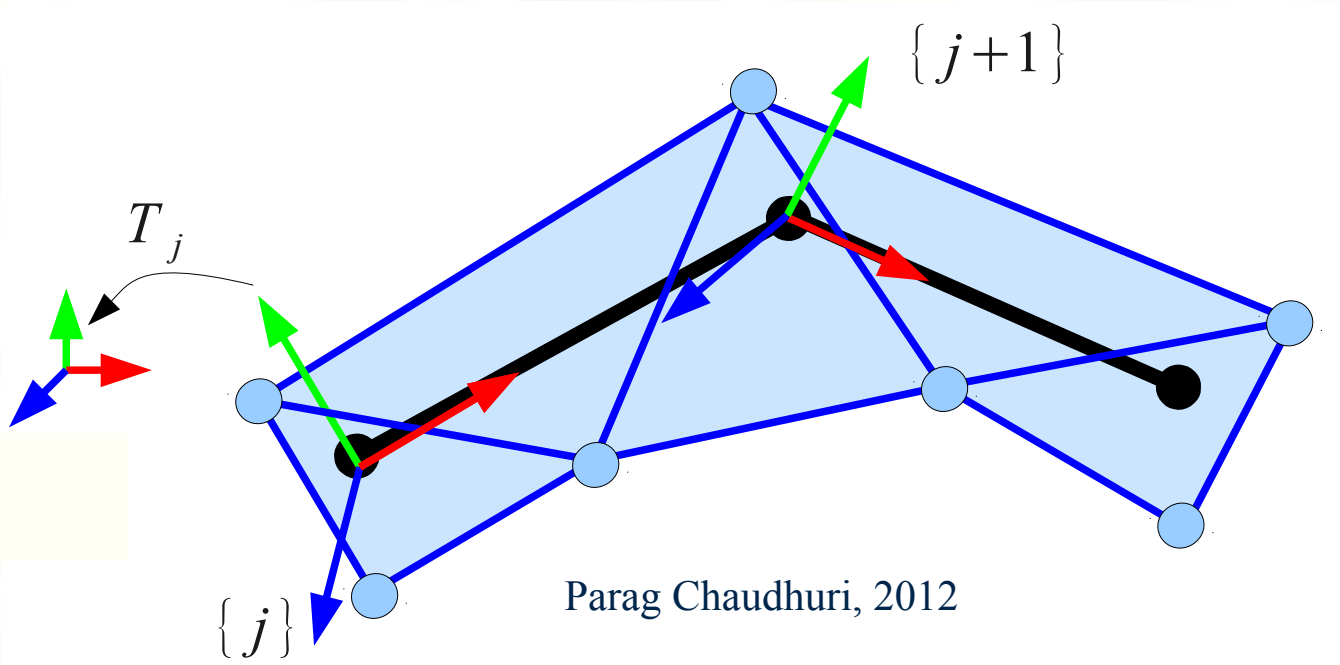
Joint motion

- When joint j moves, T_j changes and the skin vertices move with it.
- The relative positions of the vertices in the local joint frame don't change
- $v'_k = T_j B_j^{-1} v_k$



Problems with rigid skinning

- Simple but low quality because large distortions happen when bends form at joints



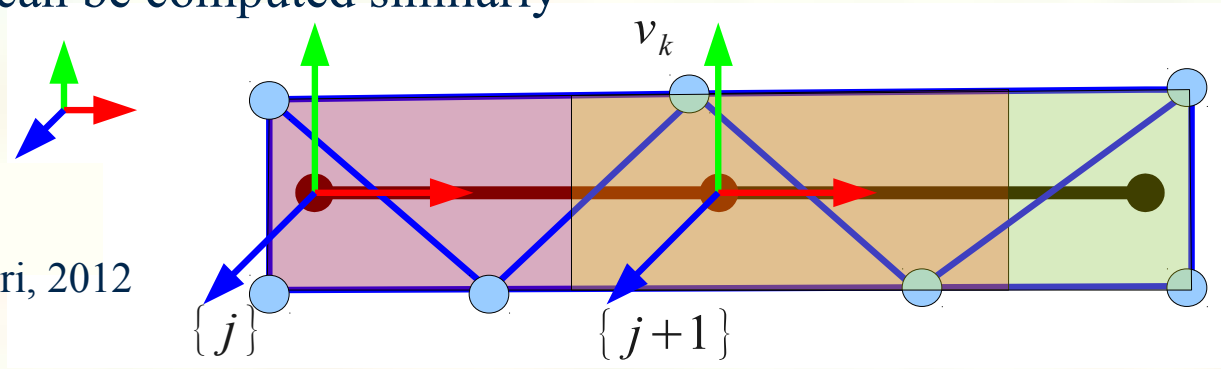
Linear Blend Skinning

- Adds flexibility to fix artifacts but still simple and fast
- Commonly used in games
- Vertices associated with multiple joints, not just one
- Vertex transform is a linear combination of the transforms associated with its joints. Each vertex has weights for this linear combination assigned to it

$$v'_k = \sum_i w_{i,k} T_i B_i^{-1} v_k$$

$$\forall k \sum_i w_{i,k} = 1 \text{ and } 0 \le w_{i,k} \le 1$$

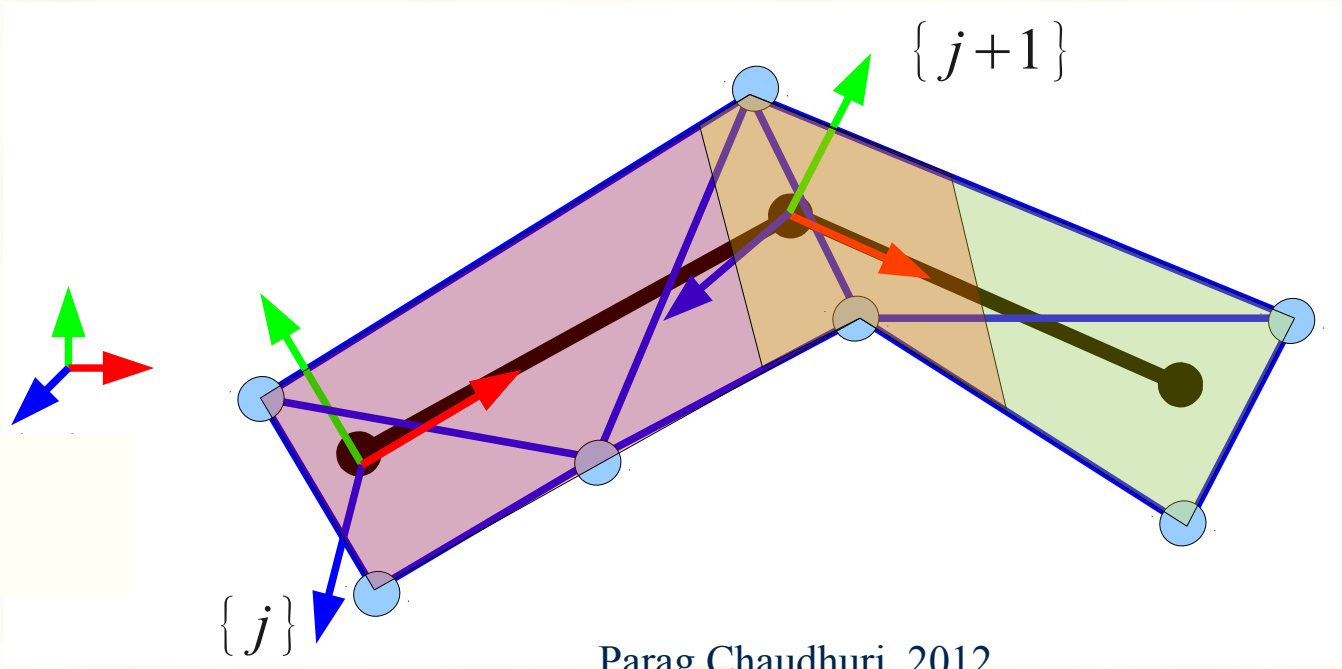
- Vertex normals can be computed similarly



Parag Chaudhuri, 2012

Fewer artifacts

- With proper weights many but not all artifacts are eliminated or improved



Parag Chaudhuri, 2012

Linear blend skinning algorithm

Skin::Update()

Compute $M_i = T_i B_i^{-1}$ for each joint. Note that B_i^{-1} can be precomputed and stored.

For each vertex compute world position and normal.

Skin::Draw()

Initialize ModelView matrix.

Draw skin polygons using global positions and vertices.

Problems

- Skin collapse at bends

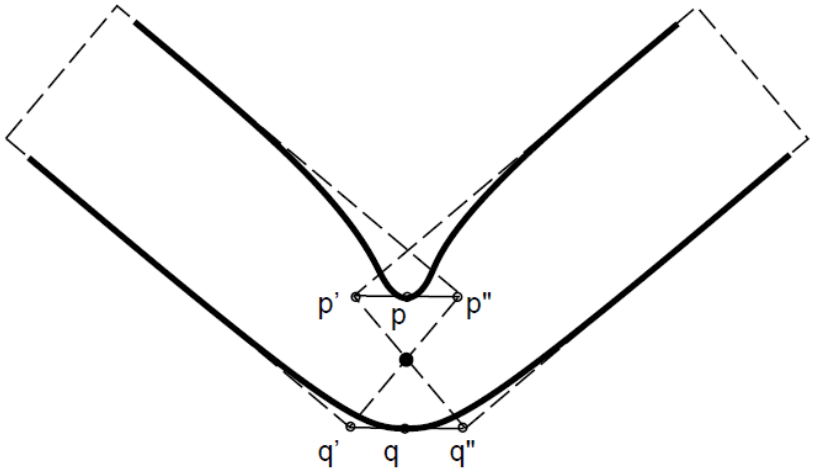
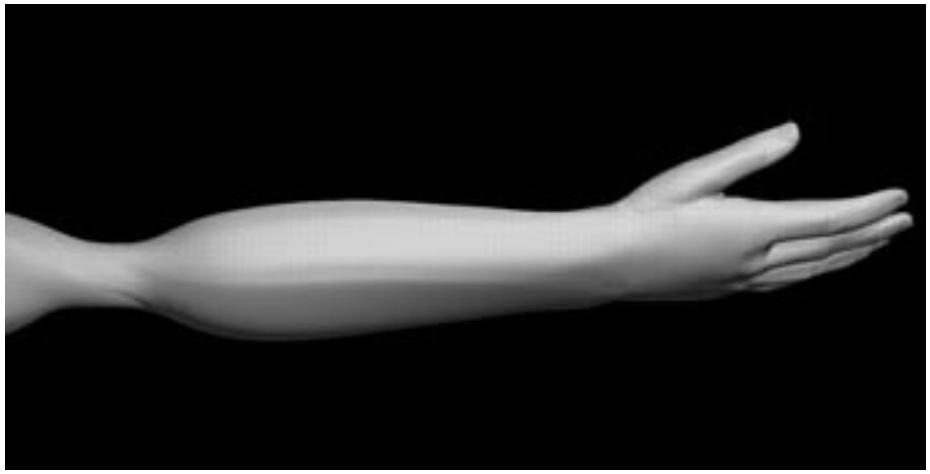


Figure 1: The skeleton subspace deformation algorithm. The deformed position of a point p lies on the line $p'p''$ defined by the images of that point rigidly transformed by the neighboring skeletal coordinate frames, resulting in the characteristic 'collapsing elbow' problem (solid line).

Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation, Lewis, Cordner and Fong, SIGGRAPH 2000

Problems

- Skin collapses at twists



Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation, Lewis, Cordner and Fong, SIGGRAPH 2000

Dual Quaternion Skinning

- Better solution, nearly as fast




Geometric Skinning with Approximate Dual Quaternion Blending, Kavan Collins, Zara and O'Sullivan, ACMTOG 2008

Linear Blend Skinning

- Problems
  - Binding is difficult – what joints should each vertex be associated with?
  - Weight assignment is not intuitive and very time-consuming
  - Still have collapse with linear blend skinning
- Advantages
  - Simple
  - Fast
  - Easy GPU implementation