

---

## CMSC 498M: Chapter 7c Modeling and Animation

**Reading:** (Not covered in our readings).

- Some material from Lecture Notes for by Steve Rotenberg at UCSD.

**Overview:**

- Animation basics and skeletal animation
- Skin and Bones
- Binding
- Motion blending

Chapter 7, Slide 1  
Copyright © David Mount and Amitabh Varshney

---

## Kinematics & Dynamics

**Forward Kinematics:**

- Given a hierarchical scene graph and **joint angles** for an articulated structure, find the **locations of all end points**.

**Inverse Kinematics:**

- Given the positions of the root, end points, and lengths, **find a self-consistent set of joint angles**.
- **Much harder than forward kinematics**, multiple solutions possible.

**Dynamics:**

- Given initial **positions and forces**, compute the **final positions**.

**Animation packages** (e.g., Maya, Lightwave, Kinetix):

- Support for **kinematics** and **dynamics**.
- Limited support for **inverse kinematics**.

Chapter 7, Slide 2  
Copyright © David Mount and Amitabh Varshney

## Skeletal Animation

---

### Skeleton:

- Objects represented as **hierarchy of bones and joints**.
- Joints and their **types** (e.g., hinge, universal, ball and socket).
- Joint **degrees of freedom** (rotation axes).
- **Limits** on movement may be given.

### Skin:

- Provides **smoothing, filling, and blending**.

### Binding:

- **Correspondence** between skeleton and skin geometries.

### Discussion:

- **Space efficient**, since only skeleton joints needs to be animated.
- Hierarchical structure of skeleton works well with **scene graphs**.
- Challenge is in making **skin movement appear natural**.

Chapter 7, Slide 3  
Copyright © David Mount and Amitabh Varshney

## Character Animation

---

Skeleton

Skin

Binding

Motion Blending

Chapter 7, Slide 4  
Copyright © David Mount and Amitabh Varshney

## Skeletal Animation

### Skeleton:

- A **pose-able framework of joints** arranged in a **tree structure**.
- An invisible **armature** upon which to attach the **skin** and other **geometric data** of the character.

### Joint:

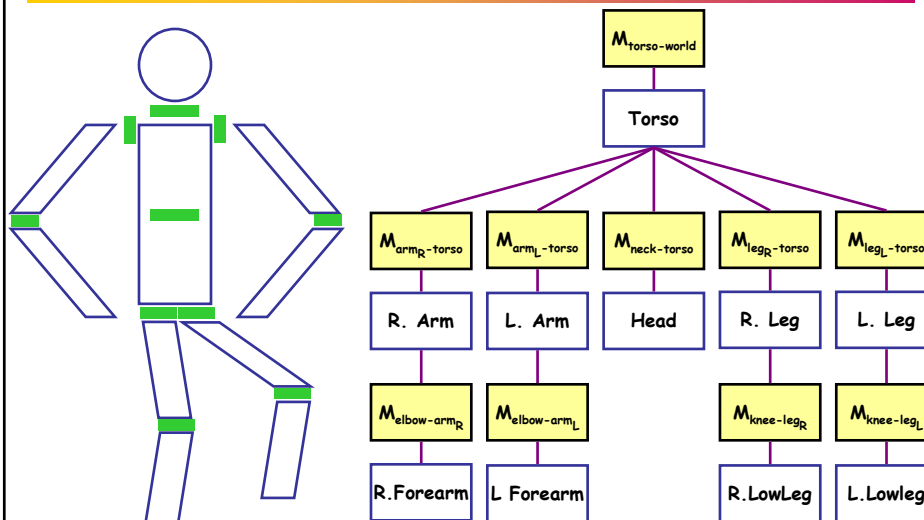
- A joint allows **relative movement** within the skeleton.
- Each joint is represented by a **transformation** (4x4 matrix).
- Joints can be **rotational**, **translational**, or some non-realistic types as well.

### Bone:

- Bones connect joints. Skin is **placed relative to the bones**.
- In many systems joints are the main entities, **bones are implicit**.
- For example, one might refer to the **shoulder joint** or **upper arm bone** (humerus) and mean the same thing.

Chapter 7, Slide 5  
Copyright © David Mount and Amitabh Varshney

## Character Skeleton

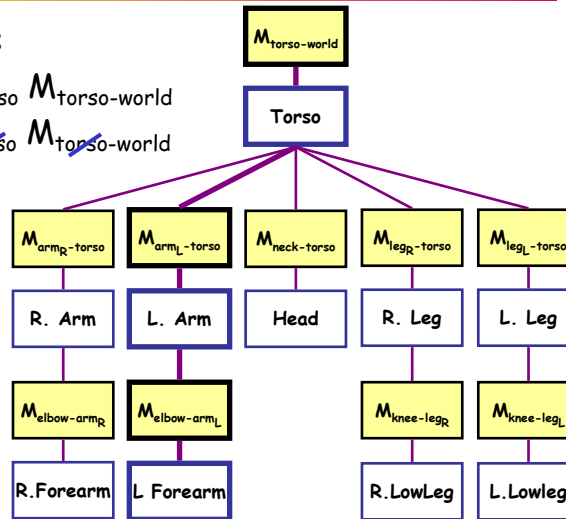


Chapter 7, Slide 6  
Copyright © David Mount and Amitabh Varshney

## Character Skeleton: Forward Kinematics

Position of left arm:

$$\begin{aligned}
 & M_{\text{elbow-arm}_L} M_{\text{arm}_L\text{-torso}} M_{\text{torso-world}} \\
 = & M_{\text{elbow-arm}_L} M_{\text{arm}_L\text{-torso}} M_{\text{torso-world}} \\
 = & M_{\text{elbow-world}}
 \end{aligned}$$



Chapter 7, Slide 7  
Copyright © David Mount and Amitabh Varshney

## Character Skeleton: Animation

**Moving the Joints:**

**Mapping Angles to Positions:** Each matrix is parameterized by its rotational **degrees of freedom**, each constrained to lie within some given **angular limits**:

- E.g.,  $M_{\text{knee-Rleg}} = R_x(\varphi)$ , where  $-180^\circ \leq \varphi \leq 0^\circ$ .
- Each joint matrix also encodes a **joint offset** (length of the bone), thus in fact,  $M_{\text{knee-Rleg}} = T(\text{len}) \cdot R_x(\varphi)$ .

**Animating Joints:** To generate the animation each joint's **motion parameter** (say the joint angle,  $\varphi(t)$ ) is changed from **frame to frame**.

**Joint Angles:** Changes (velocities, accelerations, etc) are specified by a **higher-level animation system** (e.g., keyframe, motion capture, or procedural).

Chapter 7, Slide 8  
Copyright © David Mount and Amitabh Varshney

## Character Skeleton: Animation

### Pose:

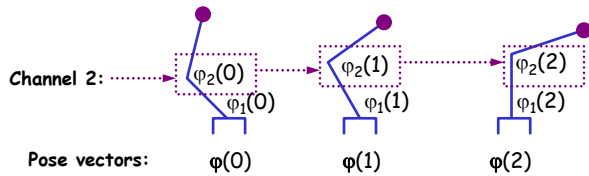
- A list of parameters  $\phi = (\phi_1, \phi_2, \dots, \phi_n)$  defining the various **joint angles** of the skeleton.

### Channel:

- A sequence of parameters for a **single joint angle**  $\phi_i(t)$ .
- Often use **parametric curves** (Bezier, B-spline, NURBS, ...) to edit, interpolate, approximate, or compress.

### Animation:

- An array of **poses**  $\phi(t)$  or an array of **channels**  $(\phi_1(t), \phi_2(t), \dots, \phi_n(t))$ .
- Tradeoff **memory access coherence** vs. **CPU computation**.



Chapter 7, Slide 9  
Copyright © David Mount and Amitabh Varshney

## Computer Animation

- Key-frame animation
- Procedural Animation
- Motion capture

Chapter 7, Slide 10  
Copyright © David Mount and Amitabh Varshney

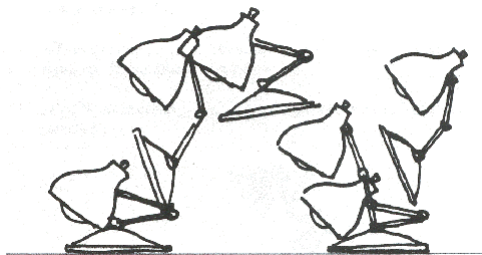
## Keyframe Animation

### Keyframe Animation:

- Main animator draws a few **key frames**.
- Others (artists or computers) draw **intermediate frames**.
- Drawing is simplified by drawing layers of the scene on **translucent cels** (celar animation), superimposing, and photographing them.
- Traditional approach to **animation in movies**.



Pixar's "Luxo Jr."



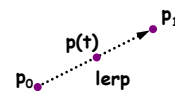
Chapter 7, Slide 11  
Copyright © David Mount and Amitabh Varshney

## Interpolating Key Frames

### Interpolation Methods:

**Linear Interpolation: (lerp)** Take weighted averages.

$$\mathbf{p}(t) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1, \quad \text{for } 0 \leq t \leq 1.$$



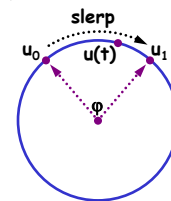
Appropriate for points, but not for angles.

**Spherical Interpolation: (slerp)** Let  $\varphi$  denote the angle between the unit vectors  $\mathbf{u}_0$  and  $\mathbf{u}_1$  ( $\varphi = \arccos(\mathbf{u}_0 \cdot \mathbf{u}_1)$ ). The spherical interpolation is given by:

$$\mathbf{u}(t) = \frac{\sin((1-t)\varphi)}{\sin\varphi} \mathbf{u}_0 + \frac{\sin(t\varphi)}{\sin\varphi} \mathbf{u}_1, \quad \text{for } 0 \leq t \leq 1.$$

**Normalized Lerp: (nlerp)** Do a lerp, then normalize to unit length. A quick-and-dirty cheat for slerp.

$$\mathbf{u}'(t) = (1-t)\mathbf{u}_0 + t\mathbf{u}_1; \quad \mathbf{u}(t) = \frac{\mathbf{u}'(t)}{\|\mathbf{u}'(t)\|}, \quad \text{for } 0 \leq t \leq 1.$$



Chapter 7, Slide 12  
Copyright © David Mount and Amitabh Varshney

## Interpolating Key Frames

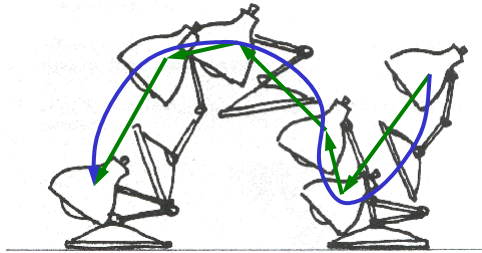
### Smoother Interpolation:

**Why:** A sequence of linear interpolations joined end-to-end can result in sudden **changes in speed** at joint points.

**Better:** Use **B-Spline** or **Bezier curves** to interpolate position.

**Objectives:** Local control, smooth motion, robustness.

**Challenge:** Maintain the right **balance** between interpolated position and timing (controlling velocity and acceleration). Almost an **art**.

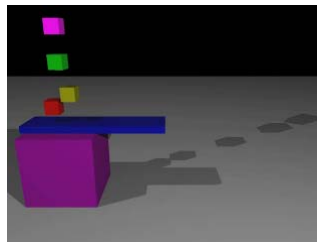


Chapter 7, Slide 13  
Copyright © David Mount and Amitabh Varshney

## Procedural Animation

### Procedural Animation:

- Specify the motion in terms in purely **mathematical terms** (e.g. equations, procedures, solution of differential equations).
- Usually best for **natural objects/scenes**:
  - Smoke, fire, explosions via **particle systems**.
  - **Physical simulations** like collisions.
  - **Periodical motion** like waves in water, blinking eyes, walking.
- Add **random noise** to periodic motion.
  - Players notice identical gait/blink cycles.
- Tricky to **fine tune** parameters to produce natural-looking motion.



Source: Kaufman, Edmunds, and Pai (SIGGRAPH 2005)

Chapter 7, Slide 14  
Copyright © David Mount and Amitabh Varshney

## Motion Capture

### Motion Capture:

- Attach **virtual reality trackers** on a person or use cameras and attach **markers** on the person.
- **Record tracker/marker motion** and play back the motion.

**Pros:** Real-time, easy to capture **complex human motions**.

**Cons:** May **not be feasible**. May be **hard to edit** if changes needed.



Chapter 7, Slide 15  
Copyright © David Mount and Amitabh Varshney

## Character Animation

Skeleton

Skin

Binding

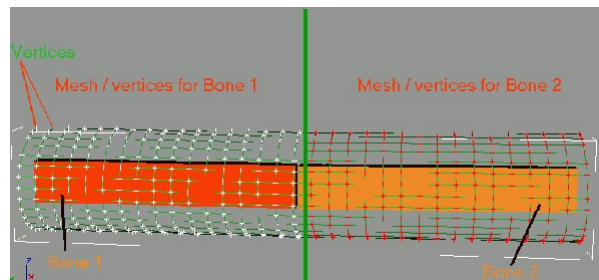
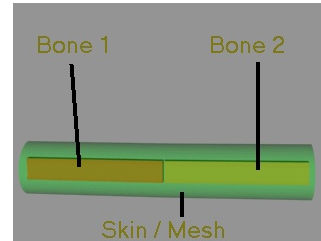
Motion Blending

Chapter 7, Slide 16  
Copyright © David Mount and Amitabh Varshney

## Skin and Bones

### Skin:

- A **mesh** attached to each bone.
- Each **vertex** of this mesh is placed relative to (one or more) bones.



Source: [http://www.flipcode.com/articles/article\\_dx8shaders.shtml](http://www.flipcode.com/articles/article_dx8shaders.shtml)

Chapter 7, Slide 17  
Copyright © David Mount and Amitabh Varshney

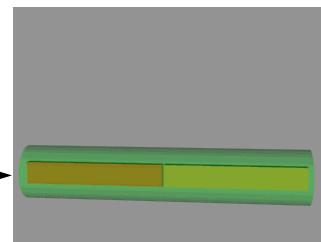
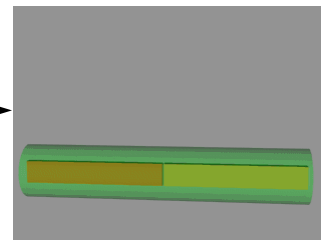
## Skin and Bones

### Problems:

- When joints flex, vertices attached to them **move rigidly**.
- While this definition is simple, it is **not very realistic**.
- Flexing of joints cause mesh to **stretch** and/or **self-intersect**.

### Smooth Skin:

- Want a system that **flexes naturally**.



Source: [http://www.flipcode.com/articles/article\\_dx8shaders.shtml](http://www.flipcode.com/articles/article_dx8shaders.shtml)

Chapter 7, Slide 18  
Copyright © David Mount and Amitabh Varshney

## Rigid vs. Smooth Skin

### Rigid Skin:

- Every vertex is associated with **exactly one joint**. It is expressed relative to the **joint's local coordinate system**:

$$\mathbf{v}'(t) = \mathbf{M}_{\text{joint}}(t) \cdot \mathbf{v}$$

### Smooth Skin:

- Each vertex is associated with **multiple joints** (usually two).
- It is expressed as a **weighted average** of joint coordinates:

$$\mathbf{v}'(t) = w_1 \mathbf{M}_{\text{joint1}}(t) \cdot \mathbf{v} + w_2 \mathbf{M}_{\text{joint2}}(t) \cdot \mathbf{v} + \dots + w_n \mathbf{M}_{\text{jointn}}(t) \cdot \mathbf{v},$$

where  $w_1 + w_2 + \dots + w_n = 1$ .

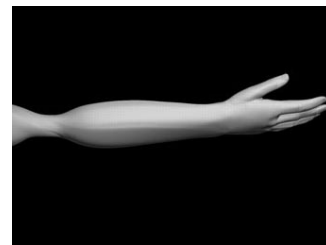
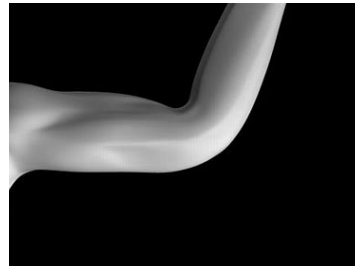
- This is called **geometry blending**.
- **Direct3D** supports geometry blending. (I don't know about OpenGL).

Chapter 7, Slide 19  
Copyright © David Mount and Amitabh Varshney

## Smooth Skin

### Problems with Smooth Skin:

- Although smooth skin produces smooth results, it is still **only a heuristic**.
- Flexing and twisting can result in unnatural **collapsing** at joints.
- **More sophisticated** approaches are needed to achieve better realism.



Source: Lewis, Cordner, Fong, SIGGRAPH 00

Chapter 7, Slide 20  
Copyright © David Mount and Amitabh Varshney

## Character Animation

---

Skeleton

Skin

Binding

Motion Blending

Chapter 7, Slide 21  
Copyright © David Mount and Amitabh Varshney

## Binding Skin with Bones

---

### Layering:

- Animation is usually easiest to do with a **skeleton**.
- Final appearance involves additional **layers**: skin, muscles, clothes, ...
- Character layers are modeled in a **natural pose** (e.g., standing).
- This pose **may not** correspond to the skeleton's **zero pose** ( $\varphi_i = 0$ ).

### Binding: A three-step process:

1. Pose the skeleton in a natural pose, called the **binding pose**.
2. Establish **correspondence** (and weights) between **skin vertices** and the **skeleton** (several heuristics).
3. Transform **skin vertices** to **local coordinate system** of joint(s).

### Establishing Correspondences:

**By Proximity:** Assign each skin vertex to the **closest** bone(s).

**Manually:** Create **bounding volumes** (spheres, ellipsoids, cubes) for each bone to enclose skin vertices that belong to it.

Chapter 7, Slide 22  
Copyright © David Mount and Amitabh Varshney

## Character Animation

---

Skeleton

Skin

Binding

Motion Blending

Chapter 7, Slide 23  
Copyright © David Mount and Amitabh Varshney

## Blending: Unique to Games

---

### Need for Blending:

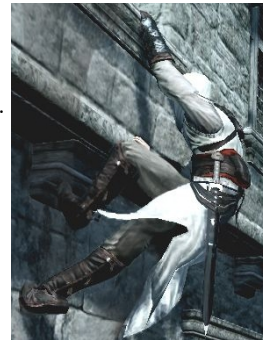
- Characters in games have a **library of actions** that need to be sequenced (generally seamlessly) **on demand**, at interactive rates.

### - Examples:

- **General:** Sitting, walking, running, climbing.
- **Battle:** Sword fighting, punching.
- **Sports:** Throwing a ball, swinging a bat, kicking.

### Motion blending:

- Interpolate between:
  - **ending pose of previous action** and
  - **starting pose of next action.**



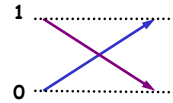
Assassin's Creed

Chapter 7, Slide 24  
Copyright © David Mount and Amitabh Varshney

## Cross Dissolve

### Cross Dissolve:

- **Lerp** or **Slerp** from one pose to the next.
- The **simplest** type of blend.
- The term "**cross**" comes from increasing one weight while decreasing the other.
- Unfortunately, does **not** always produce **realistic results**.



### Challenges: How can we produce more realistic transitions?

**Ensuring phase synchronization:** E.g., running to kicking.

**Adapting to changes in velocity:** E.g., walking to running.

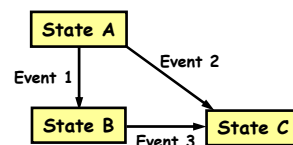
**Mixing rotations and translations:** E.g., sitting to walking.

Chapter 7, Slide 25  
Copyright © David Mount and Amitabh Varshney

## Animation State Machine

### Animation State Machine:

- A **finite state machine** where:
  - States:** represent **animation clips**.
  - Transitions:** represent **motion blends**.



### Features:

- Enables **complex motion sequences**:
  - Sitting to walking should have an intermediate stage of standing.
- **Object-oriented** way to build complex motions:
  - Encapsulate simpler motions (state machine) as a **single state** of a more complex motions.
- Allows greater **control of transitions** between selected states.
- **Simplifies design** of animated games:
  - Transitions triggered by **user events**, game AI, randomness, ...

Chapter 7, Slide 26  
Copyright © David Mount and Amitabh Varshney

## Summary

---

### Summary:

- Animation basics and skeletal animation
- Skin and Bones
- Binding
- Motion blending

### What's Next?

- Procedural modeling