Character Animation

Christian Miller
CS 354 - Fall 2011
Exam 2 Grades

- Avg = 74.4, std. dev. = 14.4, min = 42, max = 99
• Everything is important in an animation
• But people are especially sensitive to characters
• We’re biologically primed to recognize faces, expressions, and life-like motions
• It’s worth spending a huge amount of time and effort to get them to look right
Appearance

• The first thing to do is create the character mesh
• Authored in a static, neutral pose
Making it move

- A static mesh is of little use
- We want to be able to pose the character like an action figure
  - And have it deform nicely
- We can do that with rigging and skinning
Rigging & Skinning

- Put a skeleton of rigid “bones” inside your mesh
  - This process is called rigging
  - Do all your animation on that skeleton
- Assign “skin weights” for every vertex on the mesh to every bone, which control how attached that part of the mesh is to a given bone
  - This process is called skinning
  - Do some math at runtime to deform the mesh
R&S Example
Rigging

• Create a hierarchy of bones inside the character
• Put joints where you want to bend the mesh
Computing bone poses

• The hierarchy is just a parameterized scene graph
• You have a set of bones, which are connected together with joints
• Each joint has an angle or orientation that can change over time
• Once you have joint angles, compute the whole hierarchy with forward kinematics (FK)
Forward Kinematics

- Start at the root node (normally the hips)
- Compute the joint transform, push it on your matrix stack
- Multiply in the bone transform to the next joint
- Recurse on all children
- Pop matrix stack on the way back up
- Gives you world transforms for every bone
Skinning

- Skinning (or weight binding) is the process of assigning a skin weight to every vertex, for every bone
- Basically, one weight map per bone
- The skin weight determines how “attached” a vertex is to a particular bone
- Normally done by an artist with a painting tool
• Example, an arm with two bones
Skinning

- Skin weights are painted on for each bone
- Can blend weights across joints
Skinning video
Computing deformations

- Given a skeleton, a pose, and a set of skin weights, how do we compute what the mesh should be?
- The simplest technique is called Linear Blend Skinning (LBS)
  - Transform each vertex as if it were rigidly attached to each bone
  - Blend together those results using skin weights
LBS math

$$s_i = \sum_{j=1}^{n_b} w_{ij} T_j v_i$$

- $s_i$ is the skinned result vertex
- $v_i$ is the original vertex (in the bind pose mesh)
- $T_j$ is the transformation matrix for bone j (from FK)
- $w_{ij}$ is the weight for bone j on vertex i
- Assumes all $w$ for a vertex sum to 1, and are $\geq 0$
• Creates relatively nice blending where skin weights change smoothly
LBS Problems

- LBS will create “candy-wrapper” errors around joints that are very different from the bind pose.

- Dual Quaternion Skinning is a relatively new way to get rid of the problem.
LBS vs. DQs

- There are methods that simulate DQs using a bunch of very small LBS bones at each joint
What about faces?

- Skeleton-based animation works well for things with skeletons in them
- Faces are squishy and more complicated
- Worse, we’re extremely good at picking up on facial detail
Blendshapes

- Create a bunch of face poses by explicitly moving the mesh around
- Store these as differences from a base mesh
At runtime, linearly combine the blendshapes with some weights to create blends between expressions, then add them to the neutral mesh.
Animation

• We can pose and deform characters and faces now, how do we actually make an animation?

• We’ve reduced it to the problem of specifying joint angles and blendshape weights over time

• 3 main ways to do this
• Define parameters at particular times (keyframes), then interpolate between them using splines.
Mocap

- Record performances and extract frame-by-frame values for all the parameters
Mocap

- Reconstruction process is nontrivial, and can be prone to error
Simulation

- Run the skeleton (or any set of bodies) through a physics simulator

[Halo 3]
Practically speaking

- All 3 techniques have their uses
- Simulation used everywhere for very specific things
- Mocap used pervasively in movies and games
- Keyframing required for everything you can’t mocap, as well as cleanup and refinement
Good animation

• Depends on what you’re trying to do

• Disney figured out the basics a long time ago, and technology makes things easier (sometimes)

• One data point: Pixar refuses to use mocap, sticks with keyframing and simulation alone
Figures courtesy...

- Real-Time Rendering, 3rd edition [RTR]
- Unreal Engine 3 UDK [UE]
- Blender documentation [BL]
- Kavan et al., Automatic Linearization of Nonlinear Skinning, I3D 2009 [LK]
- Other tutorials scattered around the internet