Notes from Computer games CS 294 seminar on 2/10/04
Last class in the motion module

Topics covered:

- Paper 1: Kovar, Gleicher, Schreiner. *Footskate Cleanup for Motion Capture Editing,* SCA 2002
- Paper 2: Zordan, Hodgins. *Motion Capture-driven Simulations that Hit and React,* SCA 2002.
- Wrap-up of the motion module
- Demo of the latest Zelda game

First part of the class: Kovar's & Gleicher's footskate paper (presented by Pushkar Joshi)

Description of the paper:

Assumptions:

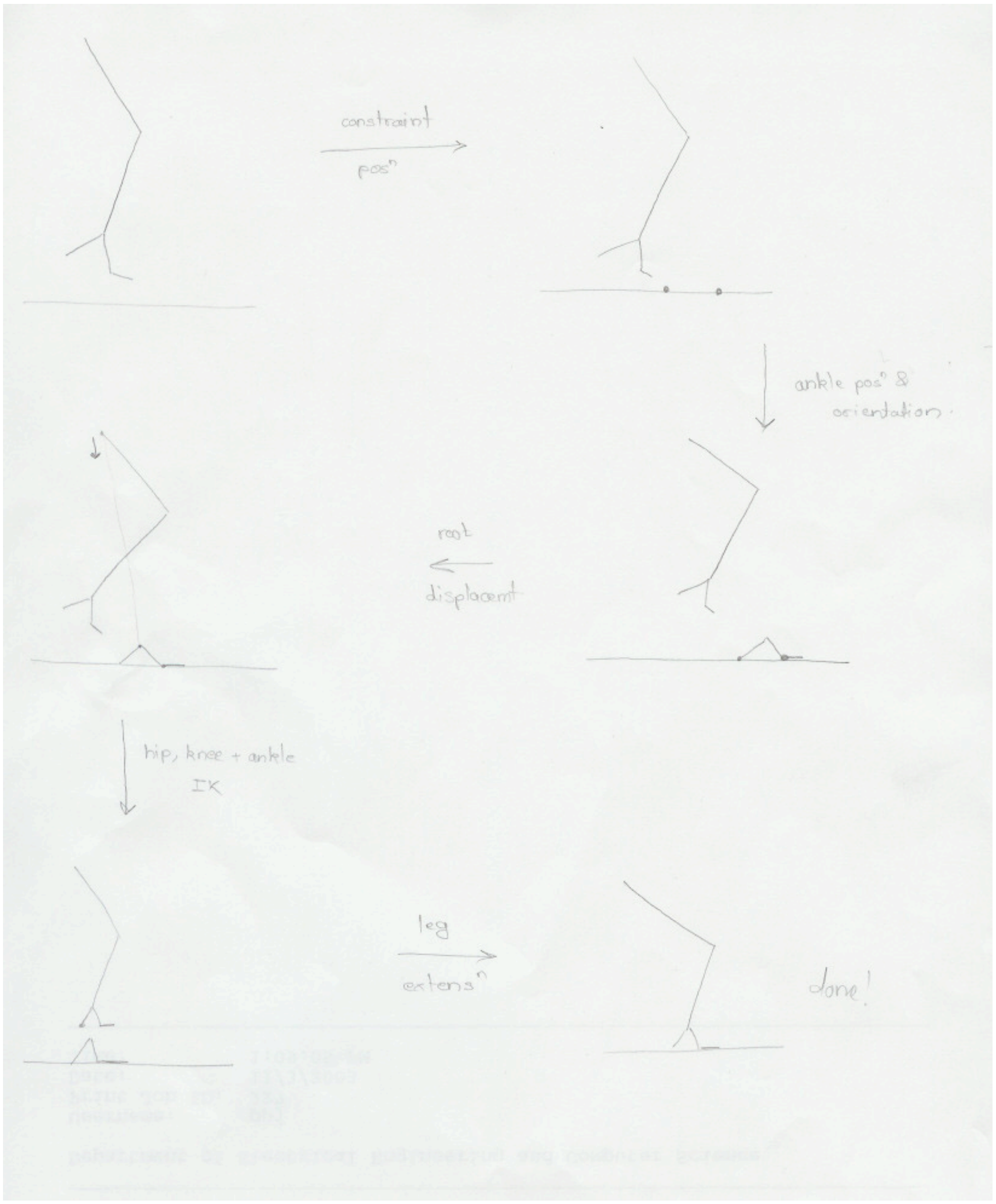
1) You already know the frames where you want the footplant to be constrained.
2) The two types of footplant constraints are the heel and the ball.

Per-frame operations (refer to Pushkar's drawings after this section):

1) Figure out footplant constraint positions for heel and ball.
    a. If same constraint in previous frame, maintain position.
    b. If not, average position over x next frames
2) Figure out ankle position and orientation
    a. Doubly-constrained (i.e. both heel and ball have plant constraints): try to keep ankle orientation unchanged, and move ankle from current position to desired position. The foot may rotate about the vector that goes from the heel constrained location to the ball constrained location.
    b. Singly-constrained: translate ankle to meet constraint. Determine orientation by blending if there are doubly-constrained frames within a window of frames (using slerp to blend). If there is floor penetration, rotate ankle slightly.
    c. Unconstrained: affected only by smoothing from nearby constraints.
3) Place the root so that we can reach the constraints
    a. If root is too far and only one ankle is constrained, project the root onto the sphere of radius *l_target* (the distance from the target to the root) centered about the constrained position of the ankle minus the hip offset.
    b. If both ankles are constrained, project onto the intersection of two spheres.

    c. Note that as constraints switch between legs, the root can pop. So, take a weighted average over a block of constrained frames. (Smoothing may place the root too far away from the target, but that's handled later on.)

4) Meet target ankle configuration using hip, knee, and ankle rotation.
    a. Adjust knee angle.
    b. Adjust hip angle and try to stay close to the ankle orientation found in step 2.
    c. Rotate ankle to get the correct ankle orientation in step 2.
    d. Damp knee angle to prevent knee popping when the leg is almost fully extended.
    e. Change leg length if necessary.
5) Clean up the motion
    a. Smooth out adjustments so there are no discontinuities when constraints appear or disappear.
    b. Fix foot-floor penetration for unconstrained feet.
    c. Fix toe-floor penetration.



Major discussion points:

Can one automatically figure out the frames in which a footplant should occur by looking at the kinematics of the body? Also, can one figure out frames where the foot should be sliding? Think of this problem either as a classification problem or as a physics-based problem.

Feet do not tend to lock onto a particular position (as this technique assumes). You may get some small amount of sliding (imagine for example wearing dancing shoes on a polished floor).

This technique assumes the constraints it's given are good. If the constraints used are wrong, the resulting motion may look bad. For example, if a footplant constraint lasts for a few frames too long, the motion of the character when she finally picks up her foot may look unnatural (there is speculation that there was an example of this in the movie that goes with the paper).

This technique uses a lot of smoothing. There was speculation that perhaps smoothing is not the best solution to this problem.

In this technique, the authors are implicitly trusting the footplant constraints more than the position of the root. Is that valid? If you consider motion capture as a measurement technique, the position of the root may be fairly reliable (barring marker sliding). Or at least, particular markers may not be reliable (i.e. the root may slide around), but you should be able to trust groups of markers. It's suspect if you have to move all the markers (which is what averaging does, and thus, what this paper does).

Next: Victor Zordan's Motion capture tracking paper (presented by Okan Arikan)

Summary of the paper:

This paper presents a method for merging motion capture with physical simulation where the character can accomplish specified tasks and react to collision forces. It proposes a technique for animating a physically-based character by having that character track a motion capture signal until it experiences an external force (like a blow from another character). At that point, the physical simulation takes over to allow the character to react to the force. The simulation then blends the character's motion back onto the motion capture signal.

Technique:

Use pd controllers to control joints when there are no external forces:

Torque = k_t * (_$_{desired}$ – _') – g_t * (d_/dt)

Qualities of this controller:

- Will cause character's joint to go close to correct _ and stay there
- Will not give you long-term zero error (to achieve this, you would need to add an integral of the error from time 0 to time t)

When character is hit and needs to produce a reaction force:

1) For any joints affected by the hit for some time t_c, set stiffness term k of the appropriate pd controller to k_r, where $k_r < k_t$
2) Linearly increase stiffness from k_r to k_t for time t_e
3) At time $t_c + t_e$, you will have returned to original stiffness

For lower-body, use balance controller:

1) Project body's COM onto the ground
2) Apply torque to leg joints to move the projected COM towards the center of support, which is computed from the polygon created by the feet (i.e. the support polygon)

Major discussion points:

Consider the balance-controller for the lower body. It will try to bring the projected COM to the center of support within the support polygon. But, it doesn't consider where the projected COM should be within the support polygon. The location really matters. Say I'm standing (on my feet), and want to launch a heavy attack to the front. One of my legs is in front of the other. My projected COM will be towards the back of my support polygon because my weight is on my back foot so that I'm ready to spring forward. In other words, I do not always want my COM to project to my center of support.

It seems as though this technique is actually a low-pass filter. Figure 3 in the paper is a graph plotting the shoulder angle versus time for the original motion capture data and for the simulation data this technique produced. Note that the simulation's rise-time is significantly slower than the original motion capture. Also, note that there is a time-delay between the two. This technique will miss some of the wiggles in the original motion capture data, so it can be considered a low-pass filter.

Why don't they play back the original motion capture data instead of the simulation data?

James suggests one way to make the simulation more stable and still set the gain on the p-d controller high: Compute a time-step. The closer your computation is to the original motion signal, the larger you set the gain on the controller. Average the answer the integrator gives you with the original motion signal, where the closer the answer the integrator gives you to the original signal, the greater weight you give the original signal in the average. This gives you a very stable integration scheme where you can set the gain high.

Demo of Zelda (played by Egon Pasztor)

This is a GameCube game, and is the latest Zelda title to come out.

Recap of motion module (David Forsyth)

Research projects (e.g. final class projects) that could be based on motion:

- Variant of control
- Automatic detection of footplant constrained frames (i.e. automatic motion clean-up)
- Infering motion from a given motion path using combinatorial motion synthesis – i.e. given a motion path, pull out little pieces of motion from a mocap database that describe the same motion of the root, and figure out how to put them together using dynamic programming.
- Consider the Pullen/Bregler technique as a method to generate motion from regression. This provides a framework for thinking about how one might extend their work.

Next time, we'll start talking about game engines.

Notes scribed by Leslie Ikemoto (lesliei@eecs)