# Coordinating Full-body Character Controllers With Shadows

**Nathan Marshak, Alexander Shoulson, Mubbasir Kapadia, and Norman Badler**

**SIG Center for Computer Graphics, University of Pennsylvania**

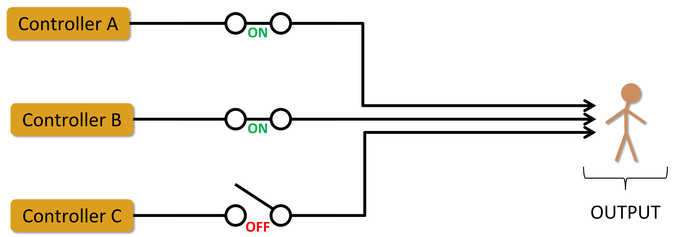
**{nmarshak, shoulson, mubbasir, badler}@seas.upenn.edu**

## Problem

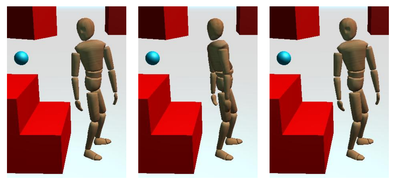
- Agents in the ADAPT platform are animated with controllers. Controllers modify subsets of the character's transform hierarchy, to satisfy some objective. Examples include gazing, locomotion, and reaching.
- Agents that can perform many tasks must have many controllers attached to them. When the number of controllers becomes large, adding new controllers without breaking the older ones is challenging. In addition, adding many controllers without introducing snapping or jerkiness can be difficult.

## Previous Approach

- Use controller scripts that follow the typical paradigm for controllers in ADAPT: Directly set the transforms of the character.

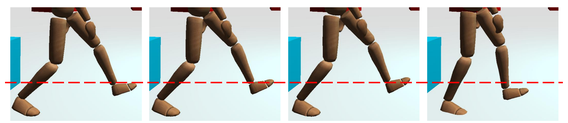



- Since all controllers write to the same data, synchronization problems can arise, tempting the programmer to make individual controllers communicate. If all controllers need to communicate directly, adding new controllers becomes an $n^2$ problem.
- Activating and de-activating controllers produces snapping and jittering artifacts as there is no easy way to blend between arbitrary controller definitions.



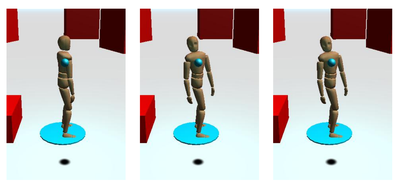
Frame 1350 Frame 1351 Frame 1352

The character's body twists even though he should ALWAYS be gazing at the ball. This happens when the sitting and locomotion controller are toggled.



Frame 3098 Frame 3099 Frame 3100 Frame 3101

Leg snap visibly – difference between first three frames is small. Difference between third and fourth frame is larger.

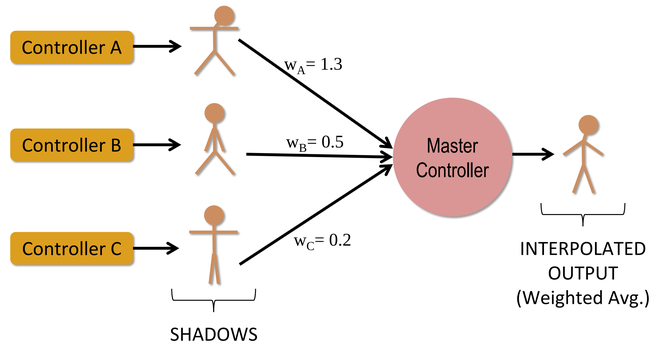


Frame 1424 Frame 1425 Frame 1426

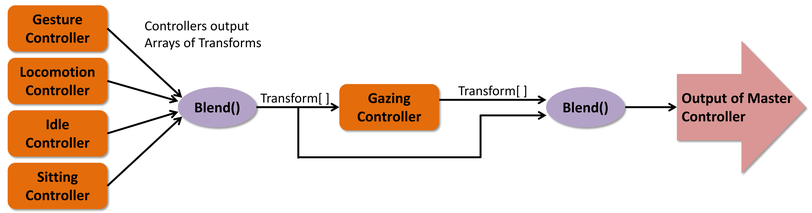
When gazing is toggled on and off, the character snaps from the original pose to the gazing pose in a single frame. This looks jerky.

## Using Shadows to Coordinate Controllers

- Each controller is instantiated with its own skeleton that matches the character's skeleton. In other words, each controller has its own "personal" skeleton. We call this "personal skeleton" a SHADOW.




- The master controller interpolates the skeletons based on per-controller and per-joint blend weights (e.g. $w_A$, $w_B$, $w_C$). It can crossfade controllers by adjusting blend weights. Controllers can now be crossfaded rather than turned on and off naively, eliminating snapping.
- The master controller sends messages to each individual controller, thus there is no need for controllers to communicate with each other. As a result, adding new controllers is less likely to turn into an $n^2$ problem.
- Controllers can OUTPUT their shadow as an ordered array of transforms. Controllers can take these arrays as INPUT, modify the transforms, and apply it to the shadow. This allows us implement blend trees INSIDE the master controller. See diagram below for an example.



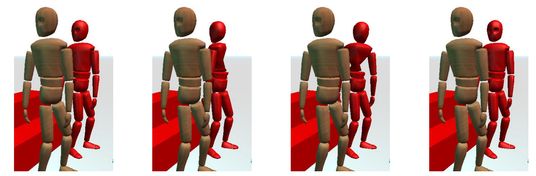

- Blend trees allow us to specify the order in which controllers act on the skeleton's state. In the example above, four controllers are blended FIRST, and then the gazing controller procedurally twists the output. This allows us to twist the character from an arbitrary pose. The gazing controller is blended with the non-twisted skeleton, which allows us to fade the controller in and out.
- Existing controllers are modified for use with shadows by inheriting from a C# interface, and by changing pointers from a character's skeleton (e.g. `this.transform.root`) to pointers to the shadow's skeleton (e.g. `this.shadowMan.root`).

## Acknowledgements

- Project developed as an extension of the Agent Development And Prototyping Testbed (ADAPT).
- Ari Shapiro and Wei-Wen Feng for support with SMARTBody (www.smartbody-anim.org).
- Parts of this research were supported by U.S. Army MURI "SUBTLE" and U.S. Army Robotics Collaborative Technology Alliance. The opinions expressed are solely those of the authors and not the sponsors.
- Poster template modified from "Approximate Correspondences in High Dimensions" by Kristen Grauman and Trevor Darrell. Template found in Ben Sapp's slides. Switch symbol from Wikipedia.

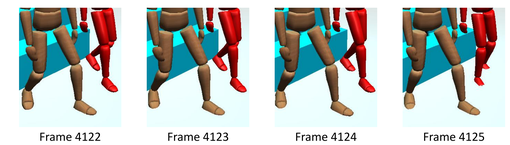
## Results

- The snapping exhibited earlier has been eliminated since we can crossfade between controllers.
- Character in **red** uses the OLD set of controllers. Character in **brown** uses NEW set.



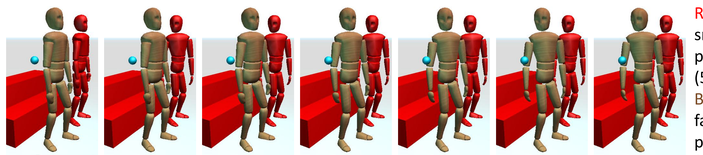
Frame 932 Frame 933 Frame 934 Frame 935

Red: The character's body twists even though he should ALWAYS be gazing at a target. This happens when the sitting and locomotion controller are toggled.
Brown: Controllers are crossfaded, thus no twisting happens.



Frame 4122 Frame 4123 Frame 4124 Frame 4125

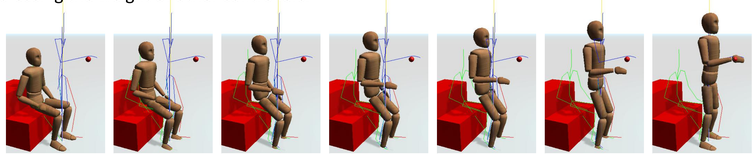
Red: Legs snap. First three frames are very similar, last frame is different.
Brown: Difference between any two frames is small, since locomotion controller is faded out smoothly.



Frame 582 Frame 583 Frame 589 Frame 595 Frame 601 Frame 607 Frame 612

Red: Character snaps to gazing pose in one frame (582-583).
Brown: Character fades to gazing pose over many frames.

- Our test case for integrating a complex controller was fading in SMARTBody, using it to perform an example-based reach, then fading it out.
- Below, the SMARTBody skeleton is in **blue**, and the sitting controller in **green**. We can fade from an arbitrary position to the reaching position by increasing SMARTBody's weight, and decreasing the weight of other controllers.



wSMART = 0, wSit = 20 | wSMART = 4.8, wSit = 20 | wSMART = 12.5, wSit = 20 | wSMART = 20, wSit = 20 | wSMART = 20, wSit = 13.4 | wSMART = 20, wSit = 5.4 | wSMART = 20, wSit = 0

## Future Work

- The master controller must be modified when new controllers are added. This could be automated with a messaging interface in which controllers are registered with the master controller.
- A GUI could be written to simplify authoring of blend trees like the one in described in "Using Shadows to Coordinate Controllers".
- Support for blending skeletal hierarchies different from the character's should be added. E.g. some controllers may only control a partial skeleton, others expect an additional root node.
- Adding complex controllers can be painful due to the number of references that need to be changed. Some of this could be automated with features like reflection in C#.