Amar Saini - Computer Animation/Simulation Project 2 Report

Project Description and Goal

Topic: A Move-Graph with Keyboard Control.

Project Description: The exact keys are: ‘a’ to turn left, ‘d’ to turn right, ‘q’ to toggle sprint on or sprint off, ‘w’ to begin walking when in the standing/initial pose.

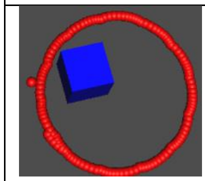
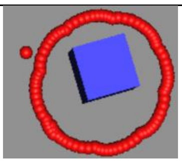
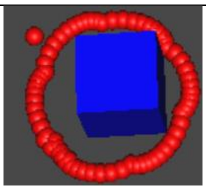
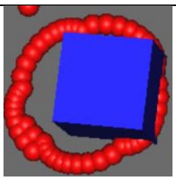
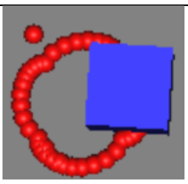
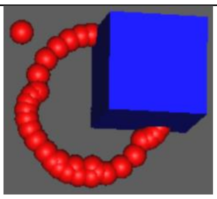
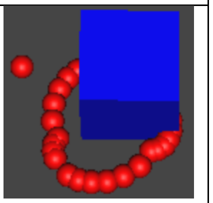
Toggle Sprint Implementation: The transitions in the move graph is basically implemented in move_graph.cpp. To toggle sprint on or off, my_viewer.cpp listens for the keyboard event “q” to occur. If q is pressed, it’ll call a method in move_graph.cpp that updates the skeleton’s move graph. Essentially, the toggle spring feature is updating the move graph to perform running left/right steps instead of walking left/right steps. Move graph corner cases are considered, such as “if I’m walking on left step and sprint is then toggled on, then we need to transition from left step to running right step.”

Turning Animation Implementation: To have the character turn, I tried 2 different methods. The first and more effective method is to simply add a global root rotation to “drift” the character. This is done by getting the orientation quaternion for the global root (the hips joint of the skeleton) and multiplying a small drift value to it (around the y-axis). So every time ‘a’ or ‘d’ is pressed, a drift value is multiplied into the orientation quaternion. The higher the drift value, the sharper the turn at the cost of animation smoothness. The second method was to blend part of the turn180.bvh animation into the walking/running animations, to get a smooth animation and a change in orientation. The part of the turn180.bvh animation that was used for blending was a trimmed version of the turn180.bvh file that only consists of the frames that actually capture the “turning” animation. Basically, the extra animation parts, such as walking to a stop before turning, were cut out. For the opposite turning direction, we would use turn180_mirror.bvh instead of the turn180.bvh. (2 different blending animations for 2 different turn directions).

Project Goal: Find a way to turn the character that is very responsive, yet well animated.

Evaluation and Results

Note: N represents the Drift Increment, or the amount of drift/rotation we use per key press for a turn. For example, a drift increment of 0.1 means we will add a 0.1 rotation around the y-axis for the orientation of the skeleton, every time ‘a’ is pressed, (-0.1 if ‘d’ is pressed). The Drift Increment can be changed via the slider during the animation. The Drift Increment slider has a minimum value of 0.05, and a maximum value of 0.2. The Drift Increment is initialized to 0.1. The Box in the figure was of size 20.0 x 20.0 x 20.0. To trace the trajectory, a red sphere was drawn at the skeleton’s position every time a turn key was pressed. The red sphere was tied to the skeleton’s root position, which in this case would be the Hips joint’s position. This is why the trajectories aren’t perfect circles, because the hips of the skeleton move a little bit due to the .bvh animations (the hips wobble, which is caused from the realistic motion capture). Also, blending was disabled to monitor the turns more accurately.

N = 0.050	N = 0.075	N = 0.100	N = 0.125	N = 0.150	N = 0.175	N = 0.200
						
Very small turn radius. Best Animation.	Small turn radius. Still really good animation.	Turn radius is small enough to *barely* circle the object. Animation is decent.	Turn radius is a bit large. It cuts the object at the corners. Animation isn’t great.	Turn is large, it penetrates the object more. Animation is poor.	Turn radius is pretty large, penetrates the object and comes out of an adjacent face, rather than the opposite face.	Very large turn radius. Worst Animation, very choppy.

Conclusion

As expected, I found that as we increase the drift increment, the turn radius increases at the cost of animation. The optimal value I found for the drift increment that maximizes the turn radius while minimizing the animation being choppy, was 0.100. As for sharper turns, (what if we had a bigger object), then we would need to sacrifice the animation smoothness for a bigger drift increment for a larger turn radius. Another option would be to attempt to blend a sharp turn animation with a walking/running animation. (such as using the turn180.bvh, but maybe capture only the frames that show a 90 degree turn). I tried this, and it achieves sharper turns, but the resulting animation didn’t look too great. The animation appeared to be very jagged, (the character taking a step then throwing it’s hand out and turning in a very sharp manner, then returning back to its original walking/running posture). If we had a turn180.bvh file that had a simple animation of just turning the character (without throwing hands out or anything), then the blending may look smoother.