Animation Documentation

In order to get my animation system started I decided to first get the 2 bone animation working within my engine. I tackled this problem by following each step that was presented during class. However, I had to tweak a few variables and settings within my engine in order to get all my previous work from last quarter and earlier this quarter to play nice with the new animation system. Once that was done, I had a working animation system that properly utilized a PCS Tree to process each bone as it transformed over time.

From there, I decided to take the lazy mans approach that was presented during the last lecture to get the humanoid animation working. It turns out that that approach was not as lazy as I thought it was going to be. Since the humanoid animation did not have a set bind pose that I could simply copy out of the converter I had to start thinking about how to derive it from the key frame data. Unfortunately, I struggled to wrap my head around the problem. Eventually, I decided to abandon the problem for now since I knew from previous experiments that the dude animation had a set bind pose I could grab from the converter.

Once I decided to go with the dude animation I went into my converter and added all the modified code presented during the final lecture so that it printed out properly formatted c++ code that would create my hierarchy and add my key frame data. After that I tackled how to reformat how the converter was spitting out the bind pose data so that it printed out the matrix I needed to set the bind pose. Turns out all I needed to do was alter a few lines of code within the DisplayPose function within DisplayPose.cxx. I also added another DisplayHierarchy function within DisplayHierarchy.cxx so that it printed out the code I needed to actually use the previously mentioned matrices to set the bind pose.

From here I copied all the output from the converter and pasted it into the proper functions within my engine. Unfortunately, once I ran the code I noticed that only about five of the fifty-eight bones were being displayed. After spending some time debugging I noticed that the converter was spitting out key frame data that claimed that most of the bones had no translation or rotation. I’m not entirely sure why, but it turns out that this particular model stores its key frame data a bit differently than what the humanoid model does. When the converter checked to see if there was an animation curve it was coming up false despite there obviously being some animation going on. So, I commented out that section of code, ran the code again, and I received data that at least seemed correct.

I copied my updated key frame data into my engine, ran the code again, and noticed that all the bones but the arms were displayed and animating correctly. So, I was moving in the right direction, but I still had several hours worth of debugging in front of me. After some major sleuthing I found out that my slerp function was sometimes returning an undefined resultant quaternion that was preventing the arms from being displayed. Turns out that I forgot to cover two special cases. First, I forgot to cover the rare special case where a key frame and the key frame after it contained the same quaternion. Thus, my calculated phi was zero, and I was getting a division by zero in my slerp equation. I fixed this problem by putting in a conditional that returned a zero quaternion if phi happened to be zero. The second problem occurred whenever one of my temporary variables turned out to be a little greater than one. This caused phi to become undefined, which in turn caused the resultant quaternion to be undefined. I corrected this problem by adding another conditional that checked if my temporary variable was just a smidgen above 1.0. If it was I reset its value to one. After I solved those two issues I ended up with a fully displayed and animating figure, shouted to the heavens in glee, and called it good after that.