Inheritance in Unity is the biggest challenge I’ve faced with the engine. For the past couple of weeks I’ve consistently had issues with GameObject components referencing the wrong object at runtime. This caused all of my inputs for my second character to go to the first, and animations for the second character would not play yet physics would still be applied normally since that was specifically coded for that character. I solved this by making a reference in every script I use that will store a reference to the GameObject in which the script storing it resided (this.gameObject). All components initialized now follow this reference to the gameObject they are attached to.

Simultaneously this helped me to understand how damage should be dealt properly when inheritance is involved, as I heavily rely on inheritance to emulate the “class” of character.

This brings me to being halfway done with my structure and execution of damage over time also as I ideally separate them into unseen objects with their own timers. They should periodically deal damage to a target referenced player. Also, my game manager has become little more than a manager for these objects since I gave movement control and attack key references to the player objects themselves to better control them.

Coming back to the animations, I nearly perfected how they should be for my “Warrior” character. I realized that AnyState in the animator should only be used for animations representing actions that should be able to be performed no matter the situation, such as blocking midair, on the ground, or in the middle of an already initiated attack (animation cancelling). I’ve found a way to accurately grab animation time from the animator itself using the AnimationController class, so now the time in which player actions should last should be better reflected in the code.