**Character System V2**

The current way that characters are scripted and built has elements of newer design strategies, older design strategies, and prototyping mentality. I will need a new architecture in order to continue building characters in a more expedient and easily manipulated/data driven way.

**V1 System Notes:**

* GetMoveMagnitude() gets the “speed” at which a character is currently moving. It is strangely located in CharacterBehaviour and should be extracted/removed ASAP
  + This is utilized by Character Animation Handler class
* All the character classes are very heavily dependent on direct function calls rather than event driven dependencies
* Some systems mix how they work
* Thankfully, at least the classes themselves are adequately defined
  + Character Behaviour
  + Character Move Controller
  + Character Animation Handler
  + Damageable
  + NPC Vision

**V2 System Notes:**

We need to now develop a new means of controlling the character, moving the character, processing damage, and changing animation states.

* Character Behaviour
  + Should be the “brain” of the character
  + Calculates “intent” to move around
  + Calculates “intent” to attack/interact
  + Handles behavior when Damageable component sends messages
* Character Move Controller
  + Should handle actually moving the character around
  + This also includes handling physics interactions (in the case of CharacterController) based characters
  + Should listen to intent from CharacterBehaviour as well as messages from Damageable component
* Character Animation Handler
  + Dependencies on Behaviour, Move Controller, and Damageable
  + Listens to behavior on state changes
  + Listens to move controller for movement
  + Listens to damageable for damage messages
  + Set variables based on other component states to determine own animation state
* NPC State Machine system
  + On Taking Damage
    - When taking damage, the NPC should respond accordingly depending on the context
      * If not already engaged, it should check for the attacker
      * If it can see the attacker, it will engage in combat
      * If the attack produces a knockback effect, wait for the animation(s) to end and handle next transition