Game Development Journey Decisions

Character Controller vs Rigid Body:

Character Controller method isn’t responsive and will not take into account gravity so I will have to develop a completely new gravity system. One benefit to this method is much greater freedom with how our object behaves which is also the greatest downfall since you have to code every single part of that yourself and is unnecessary.

Rigid Body component doesn’t require a character controller. The one notable difference is the way physics operates with rigid bodies as it allows us to use a function called Fixed Update which I researched at the Unity API scripting Documentation. This function allows us to perform any physis simulating functions to be synced at the same interval. Furthermore, if more advanced physics dependent features need to be coded for the user then it is much more efficient to use the Rigid Body.

Debate on Character Controller vs Rigid Body - <https://medium.com/ironequal/unity-character-controller-vs-rigidbody-a1e243591483>

Update Function Unity API Scripting - <https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html>

Humble Object Pattern and Unit Testing:

Separate the logic used in MonoBehaviour framework. These MonoBehavious classes are sometimes too hard to instantiate since they are so coupled to their environment. We should use Monobehaviours as object just that are delegators for unity system calls to manage the logic.

This way we can unit test the actual logic of the program as we proceed

Humble Objects Video by Infallible Code - <https://www.youtube.com/watch?v=OecJvh8Zvc4>

Interfaces:

Start using interfaces where each interactable objects can implement and then the all the player class update method has to do is look for that specific object. This will avoid the code smell of classes are way too large which allows are code to be more scalable and easier to maintain. Now any class implementing this interface can be searched for without having to add a line of code for every single object.