Developer Log

Starting with Player Controller script, there is quite a few declarations and code they I didn't end up using and forgot to scrap from the script. Primarily what I end up using that is meant for this script is setting the transform.localscale to a certain size because when running the game. The character sprite was too small, and I didn't want to change the scale on the inspector tab because I wanted it to be code-able where I can change it at a later time. That was stated in the Start method so that the size is set immediately. In the update method I have two other methods called, Walking() and Jumping(). The Walking() is set to take key inputs, left and right, to move the character. I set the transform.position to be += to Vector3.left/right * speed * time.deltaTime because I wanted the character to be able to stop on a dime without any kind of momentum. This is so that the player has complete control over the character and won't have any difficulty jumping platform to platform.

Moving onto PlayerManager Script, there is an enum code black that has white and black, which are the two forms of which the characters sprites encapsulate. In the Start() I have it set so that the characters starting position is at a location of gameobject. So whenever the level loads in or if the character dies. It'll automatically restart in that same exact location. I create a GameObject Scene in order to access a variable called currentSceneColor that is in the lvlManager script held with in the Scene gameobject. I could have made public lvlManager lvl to get to the variable with less code, but that's also more variables being made public which I didn't really want to happen. AppleCount, which is the counter of how many apples the player collected, is set to 0. This is because when the player dies, the counter restarts back to 0 and the apples reset, so the level actually resets. In the update method is where I keep the switch case for the enum State of switch betweening the black sprite and white sprite. I should have moved into it's own method but didn't think of it till now. In the same Update method I have the CurrentScene method that takes the currentSceneColor mentioned earlier as tracks what the scene's color currently is at so that it can change character sprite. Another method that is not called in Update() is Respawn(). This method takes the location of the GameObject Start and moves the character to that location. This method is called by another at a later time when the character dies.

Bound is a script that takes the rect size of the camera and places those values as boundaries for the player to move in. So, the player is unable to move left and right pass the boundaries of the camera. However, the player can drop below the camera view, which will indicate the player "fell off the map" forcing a restart on the level. I placed an if statement for the HitSideBound() code block because on the last level, I want the camera to follow the player, but keeping that method active presents the player from moving passed the side camera view. When the player falls passed the bottom of the camera view, the player dies, initiating the PlayerCollide.PlayerDead = true. I should have moved the PlayerDead code a part of PlayerManager rather than PlayerCollide since it makes more sense for it to be part of the manager rather than a part of a collision script.

I feel as though a lot of my code is very designated for this game only, and I never really thought about using code that can be used for other games. I think I was more fixated on making the game work rather than coding a video game. Looking at how my other classmates made their games made realize that I still need to learn what a Game Programmer actually is. I never once made a child class. I never

made a InputHandler. I only once made an actual enum of states. There is still so much more I can learn on what programming a video game actually means.