Learning Journal

I have already known how to make a basic CharacterController in Unity, but I never fully understood how to make the CharacterController have control in the air. After looking at several Unity Forums, I have found that the solution would be to use 2 different Vector3 “move directions”. The code would use the first Vector3 to define the constant movement of the player (including gravity) and if the player was not “Grounded” then it would use the second Vector3 to update the X and the Z positions but use the Y (gravity) of the previous Vector3. This change allowed me to have a simple and smooth CharacterController.

While creating the Shooting tutorial I discovered how useful and important modularity in code/components are. Using the same “health” code would allow me to quickly drop the code onto any object with a collider and have them take damage from the “bullets”, what ever that object may be; a player, an enemy or just an obstacle in the level. This is done by either a) on collision with bullet checking if the collided object has the “health” script attached, if so, apply damage to the health variable in the code, or b) in the health script create a separate void that is a receiver, and upon the bullet colliding with an object, the bullet would BroadCastMessage to the hit gameObject with the damage, and if a piece of code has that specific receiver, it would apply damage to the object.

Creating the AI tutorial taught me how to create the “NavMesh” and how simple it is to create a basic AI in a video game. In the past I thought that to make a basic enemy it would take hundreds of lines of code, but the “NavMesh” has simplified everything a lot. All that had to be done was; create the NavMesh, create and assign the NavMeshAgent to the “enemyAI”, set target of the “enemyAI” which in this case was the player.