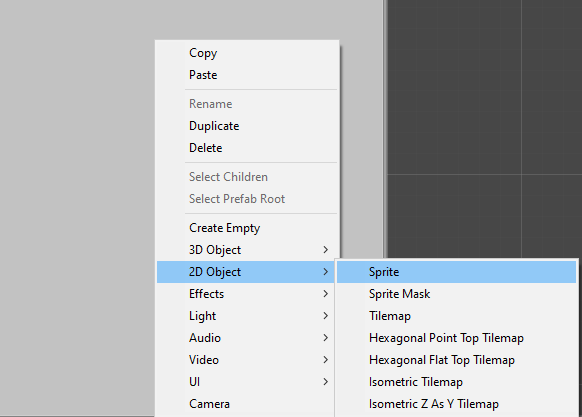
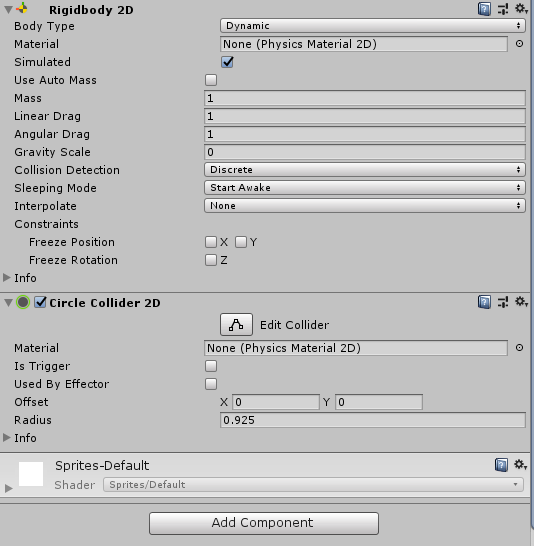
**HIT A MOVING TARGET BRIEF**

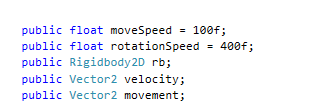
**BY SAMUEL PARSONS**

For the Moving target brief I started by creating a player that would use velocity to move around. To do this I made a sprite gameobject in the hierarchy and added a knob sprite.



I then added Rigidbody 2D and Circle Collider 2D components to the gameobject using the add component button and the bottom of the inspector on the right of the unity window. As well as a script component which I named PlayerMovement.

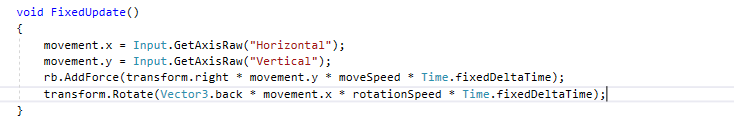
After that was all added I opened up the script and added a couple of variables. Firstly was two floats I named moveSpeed which was equal to 100f and rotationSpeed which I set to 400f. I then added a reference to the rigidbody of the Player gameobject as well as two Vector2 variables. The first Vector2 I named velocity and then the other I named movement.



I then created a *FixedUpdate* function which I used to create the physics based movement. I begin by making the x value of movement equal to the horizontal input value and the y value equal to vertical input value.



To add forward momentum to my player I then used the *rigidbody.AddForce* function utilizing the transform.right value. I then multiplied that value by the *movement.y* value as well as moveSpeed and *Time.fixedDeltaTime*, this all together made me able to propel the character forward at a speed that increased overtime. I continued to make a *transform.Rotate* function that used the gameobjects *Vector3.back* and multiplied it by the *movement.x*, rotationSpeed and *Time.fixedDeltaTime,* similar to the movement function this one rotated the player at a gradually increased rate overtime.



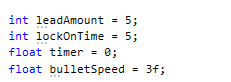
Now that the player movement was done I proceeded onto the targeting AI. For this I made another gameobject which I attached a fresh script to named EnemyTargeting. Within this script I would use math and velocity to determine where the player would be moving to and fire a projectile accordingly. I began by making 2 integers and 2 floatst:

-leadAmount which I set to equal 5 this would determine out much the projectile will lead its shot by

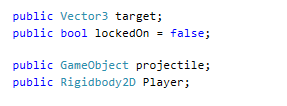
-lockOnTime which would be set to 10, this would be the time at which the enemy stops targeting and fires.

-timer this would be set to 0 and will count up until it reaches the lockOnTime which it would be then reset

-bulletSpeed which was equal to 3, this would be the speed at which the bullet travels so I can get an accurate lead on the shot.

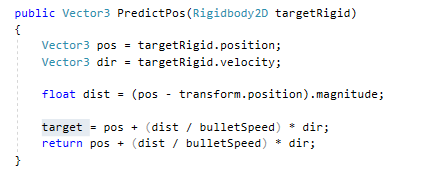


I then created a vector3 that would hold the position of the player for targeting as well as a bool that would be used for switching between locking on and off, this will be set to false to start with. Finally I added two variable references to a gameobject that would hold the projectile prefab and the players rigidbody2D which will be used for the players position.

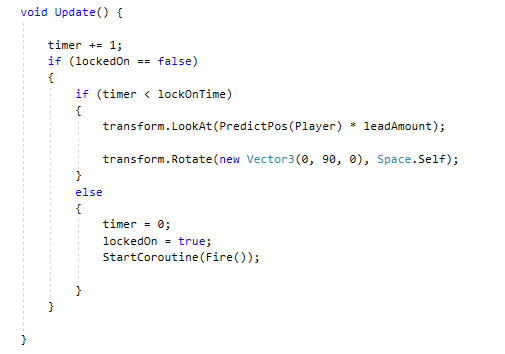


Next I attempted to create a function that would predict the players next position. To do this I created a Vector3 function that would take in a rigidbody and use that rigidbody to determine with its velocity and position where it will be next. To do this I stored the velocity and position of the target rigidbody in two separate variables I named pos (position) and dir(direction). I then made a float called distance that took the position of the player, took away the position of the AI and then returned the length of the vector using *.magnitude*.

I finally set the target variable to equal the position of the player plus the distance the bullet would travel which was divided by the bullet's speed and lastly multiplied by the direction the player was moving in, I lastly returned the same equation.

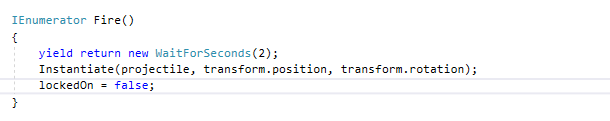


Now that I had the players predicted position all I had to do now was fire a bullet at that position. To do this I created a timer that would fire at intervals as well as make the AI turn to face the character(the rotation was just for visual aid in seeing if the AI was in fact aiming for the right position). For the timer I used the premade update void and made the timer go up by 1 every frame, I then made an if function that would check if the bool lockedOn was false, if so another if function would check if the timer had reached the lockOnTime. While the timer was below the lockOnTime the AI would continuously turn to face the player. Once the timer exceeds the lockOnTime the timer would be reset to 0, the lockedOn bool will be set to true and the Fire coroutine I made next would activate.



Next I worked on making the Fire coroutine, for this I used an IEnumerator which I named Fire. I first made a waitforseconds function that made the coroutine wait 2 seconds once called then after those 2 seconds it would instantiate the projectile from the position of the AI

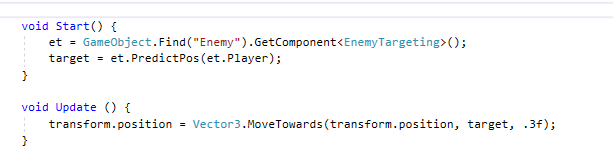
. Finally the coroutine would set the lockedOn bool to false.



Now that the predicted position was calculated at the projectile had been shot all that was left was to make the projectile land at the spot it was shot at. To do this I made one last 2D sprite gameobject with an empty script on which I named projectile. For this I simply made a reference to the gameobject that contained the targeting script as well as a vector3 that would hold the target position.



I then used the void start function to find the AI gameobject with the targeting script and then initiated the PredictPos function to give the position of the player. Then in the void update I made the projectile travel towards the static location using Vector3.MoveTowards.



Now that all the code was done all I had left to do was attach the rigidbodies and prefabs to the corresponding empty slots in the script components. I also made a few walls with colliders so that the player wouldn't go off screen and stay within the boundaries.