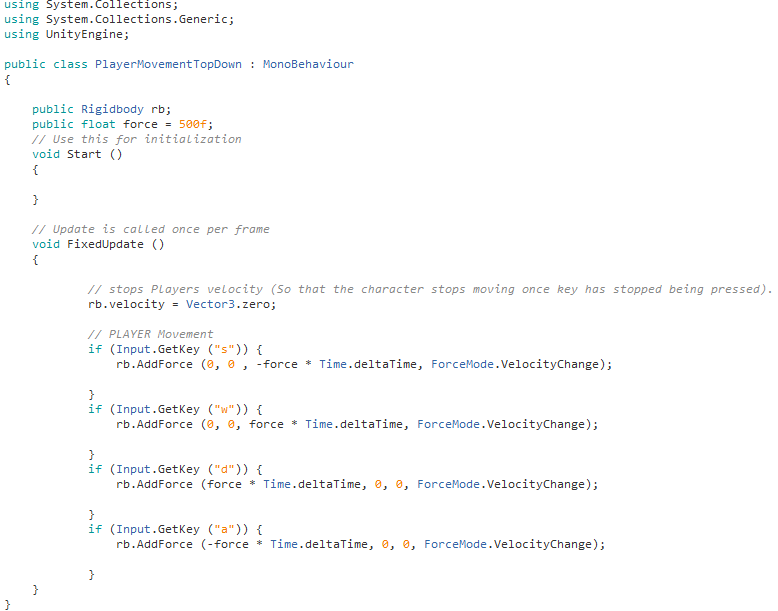
Coding Script Tutorials

# Player Movement Script



Placing this script on the player object will allow the player to control the object using the “WASD” Keys.

1. We’ll want to start by declaring the Object’s Rigidbody as a public so that we can drag the Objects Rigidbody upon it.
2. Next, we’ll be declaring the force which will be applied to the object which again will be made public so that we can easily change this value in the editor if it feels to quick or slow for the game.
3. You’ll want to edit the “*void Update*” to “*void FixedUpdate*”
4. You’ll now want to start with an IF statement ‘*Input.GetKey (“”)*’ and place your input key within the quotation marks.
5. Next, you’ll want to want to use the Rigidbody that was declared earlier to specify the object you will be adding force to.

The force which is being applied to the object is based on 3 different directions the “(X, Y, Z)” and is represented by (0, 0, 0), for an example we’ll use the first if statement.

Since we want the player to move backwards, we added force to the Z axis by using the float we declared before (“*force*”) instead of inserting a raw number. And because we want it to go backwards on the Z axis we must make the number a negative so “*-force*”. After inserting the force we’ll want to times it by the games’ framerate (“*Time.deltaTime*”) so that it’ll run the same no matter the framerate being used.

Lastly, we have “*ForceMode.VelocityChange*” which sets it so that the velocity is reset when you move to a different direction.

1. Repeat part 4) -5) for each IF statement.

# Player shield Script

# 

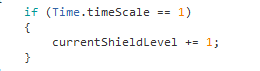
This script will create a shield around the player when the player holds down the shift key and disappears once the player has depleted its energy before slowly charging back up.

1. We’ll be declaring multiple variables and GameObjects as shown below

* The Object we’re going to use as the Shield
* A slider to show how much energy we have and depleting (if you choose to use one)
* Our current shield value
* Our maximum shield value
* A bool to decide when the shield should recover (should be set as true)

1. We’ll want to start with the “*Void Start*” and set it so that the “*currentShieldLevel*” = “*maxShieldLevel*”
2. Outside of both “void update” and “void Start” you’ll want to create a float called “*CalculateShield(){ }*” and within the curly brackets you’d want to “*return currentShieldLevel / maxShieldLevel*;” this will calculate the value of the “shieldBar”.
3. Return to the “*void Start*” and insert “*shieldbar.Value = CalculateShield*” this will display the shields bar value if there was a UI.
4. In the “void update” you’ll want to a “*shieldbar.Value = CalculateShield*” this will constantly update the value as you’re playing the game.
5. The following if statement makes it so that if the shield recover bool is true and the shield is less than or equal to the maximum value it will use the following code.

Within the next if statement we have another if statement which is, if the game isn’t paused the shield recovers 1 point

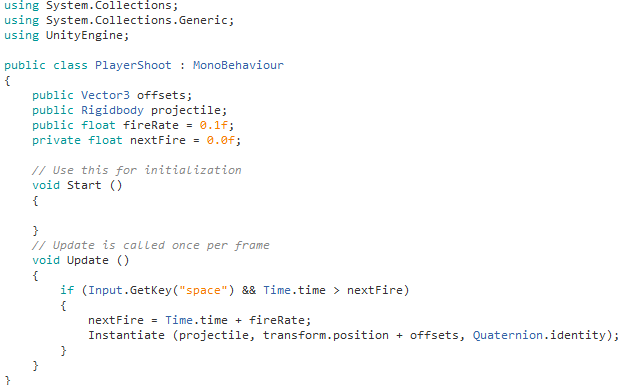


1. In another IF statement outside the previous one we have a statement which when the “left shift” and if the current shield level is less than or equal to 0.

Within this statement we set the “*shieldRecover*” to false so that the shield doesn’t recover as its being used. also, instead of doing “*currentShieldLevel += 1*” which adds to the shield value we instead do “*currentShieldLevel -= 3*” which will subtract from it decreasing its value. Lastly, we set the shield to active “*Shield.setActive (true*);” which makes the shield appear around the player.

1. The next IF statement is rather simple. Once you remove your finger from the shiftkey it’ll turn on the shield recovery by setting “*shieldRecover*” to true and remove the shield by setting the “*shield.SetActive*” to false.
2. Lastly, we have the final IF statement within “void Update” which is IF the current shield value is less than or equal to 0 it’ll deactivate the shield (The shield must be a parent to the player and sit on top of it)

# Player Shoot Script



This script allows the player to shoot a projectile with a rate of fire by holding down the space key.

1. As before we’ll start by declaring the variables

* A public vector3 Offset
* The Rigidbody of the projectile we’re shooting
* A float of the rate in which the player will shoot
* And a float of the next fire

1. Skipping “void Start” we’ll go straight into the “void Update”. We’ll start off by adding a IF statement which when the spacebar is down and the Time is greater than next fire (which is 0).
2. Within the IF statement we change the value of the “*nextFire*” to the “*Time*.*time*” + “*fireRate*” adding a grace period between the next time the player can shoot.
3. Next to finish it up we Instantiate the projectile by doing “instantiate (What We’re Instantiating, the players position + the offsets which you change in the editor to fit your model length and size, the rotation of this object so the projectile faces forward).

(The projectile will have to be a prefab for it to work. To make the object a prefab you’re going to have to drag the projectile object with the script into a prefab folder in the asset list)

# Player’s Projectile Script

# 

This is the script that you’ll be placing on your projectile to make it move and delete itself

1. Not many variables to declare this time around, only two.

* The Rigidbody of the object (projectile)
* The forward force it will move at

1. We’ll begin by including “Destroy (gameObject, 1)” to the “void Start”. This will destroy the game object after 1 second.
2. Outside of “void Start” and “void Update” we’ll create a new void called “*void* *OnCollisionEnter(UnityEngine.Collision collisionInfo*){}” and within this new void we’ll have a IF statement which sets it so that if the object collides with a tag with the name == X in this case “*EnemyProjectile*” the player projectile will delete itself.
3. Within “void Update” we add force to the projectile similarly to how we add force to make our player move