**I RECOMMEND VIEWING THIS DOCUMENT IN WEB LAYOUT FOR EASE OF VIEWING**

Prior Knowledge for this tutorial

* Already have the Unity engine installed and know how to create a new project
* Already have Microsoft Visual Studio installed
* Know how to create new objects within a scene
* Know how to move between folders in Unity’s project window and create new empty scripts
* Know how to create a player movement scripts

In this tutorial, I will be showing you how to create a basic attack for a player character to use. This system will use triggers and tags within the Unity Engine. The aim for this tutorial is to create allow the player character to attack when the player presses the attack button and have this attack affect enemy characters or objects within the game.

To start we will need both a player character and an enemy for them to attack, so let’s add two cubes to our scene to act as these for now. We’ll call one “Player Cube” and the other “Enemy Cube”. While this isn’t required for a working system, I will be changing the colour of the enemy cube to brown to visually differentiate our two cubes.

A white box on a brown surface

Description automatically generated

Our enemy cube needs to be able to be hit by our attack’s hitbox so it needs a Box Collider on it. I should have one on it already but if it doesn’t then click on the enemy cube and in the inspector to the right click on the “Add Component” button and select “Box Collider”

A screenshot of a computer

Description automatically generatedA screenshot of a computer

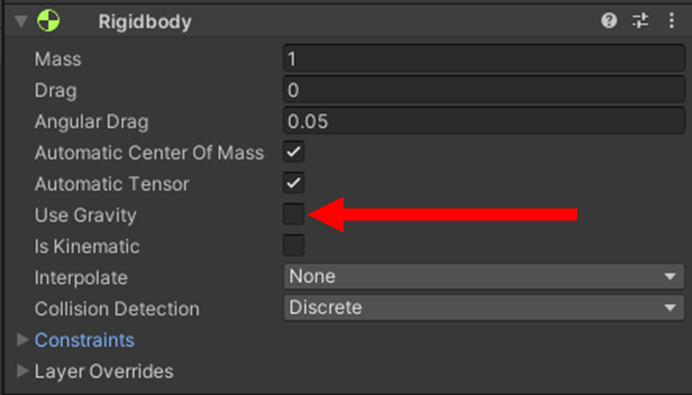
Description automatically generated

Now we also need a Rigidbody so let’s repeat this process, but this time select “Rigidbody”

A screenshot of a computer

Description automatically generated

Leaving the Rigidbody on its default settings means that our enemy cube will be affected by gravity, which we don’t want as without a floor in the scene the cube will fall forever. So let’s make sure that “Use Gravity” is unchecked in the inspector

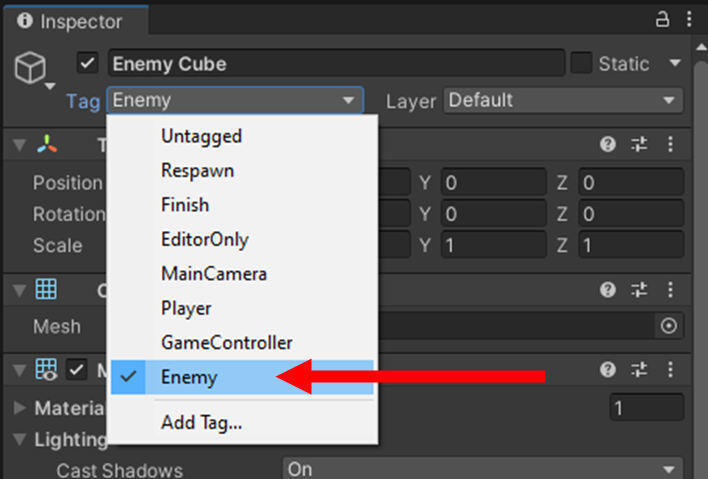


And finally, our enemy box will also need a tag so we can tell the code what to check for. Let’s select our cube again, open the Tags drop down and select “Add tag…”. We need to create a new tag so let’s create one called “Enemy” and then assign it to our cube

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

We now have our cubes but in order for one of them to attack we need an attack hitbox. This next step can also be accomplished by using an empty game object but for the sake of visual clarity I will be using another cube. You can resize this cube to your liking but keep in mind that this will be the size and range of your attack. Once you have sized the cube to your liking it will need the cube’s Box Collider to act as a trigger. Once again if your cube does not already have a box collider on it add one same way as before, from here we need to select our hitbox so that we can see its information in the Unity inspector, within the inspector we need to click “Add Component” and select “Box Collider” like before. But this time we need to look within the Box Collider tab and need to make sure that “Is Trigger” is ticked. Once again, to differentiate this box I am going to change its colour, this time let’s change it to red.

A screenshot of a computer

Description automatically generated

A screenshot of a video game

Description automatically generated

We now have our attack hitbox but we need to make it detect when it makes contact with a target and deal damage to the enemy cube. So, let’s create a new script for our hitbox and call it “AttackHitDetect”

Within our script we need to use the “OnTriggerEnter” command to tell our code to run when something enters the hitbox’s trigger so let’s write “void OnTriggerEnter(Collider target)”, this will detect when an object enters the hitbox and temporarily store said object under “target”.

A black screen with white text

Description automatically generated

Next, we want to check if that object has the enemy tag so we will need to add in if statement within our OnTriggerEnter. Let’s write “if (target.gameObject.CompareTag(“Enemy”))” which will check for the tag and only run the next part of the code if the object stored in target does have the enemy tag

A screen shot of a computer code

Description automatically generated

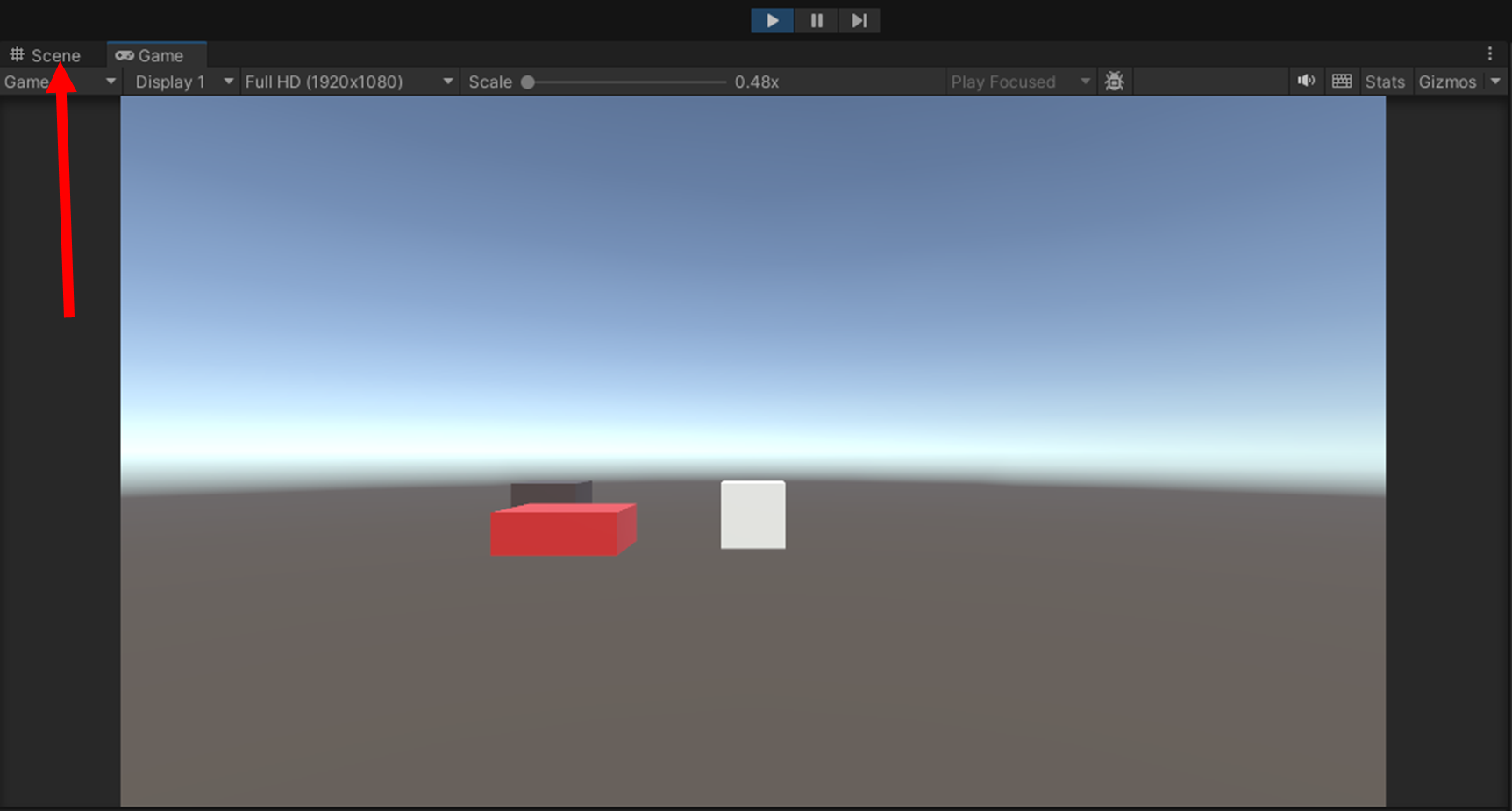
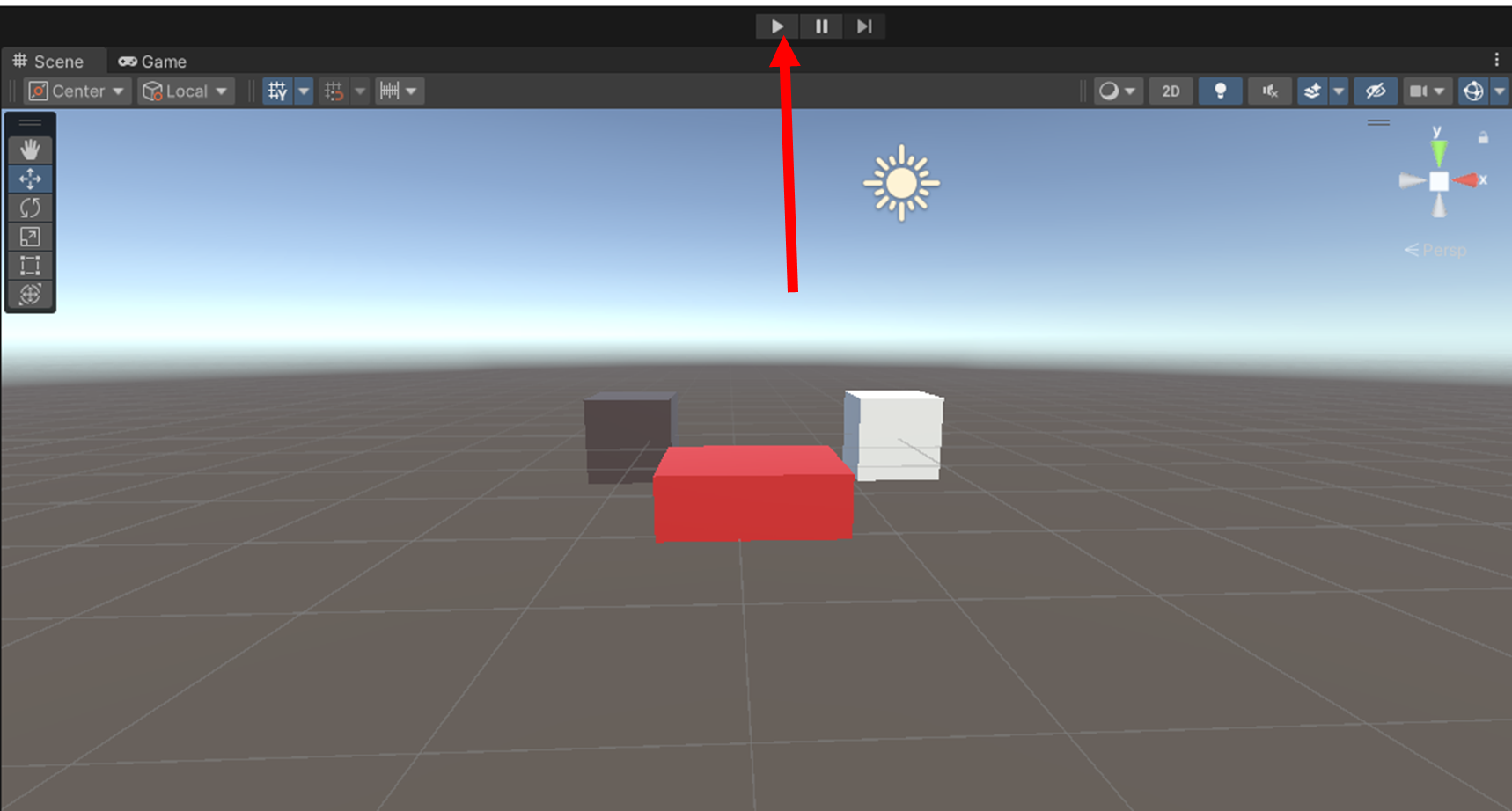
From here what you want to happen when the enemy is hit is completely up to you when you make your game, if you already know how to create a health system you could combine it with that to decrease the enemies health when they are hit but for the purpose of this tutorial I will just make the enemy cube get destroyed when it is hit by our attack hitbox so I am going to add “GameObject.Destroy(target,gameObject)” inside out if statement. Now it will deleted the object stored within target from our scene if it detect that it has the Enemy tag.

Now let’s add this script onto our hitbox. Select the hitbox, return to “Add Component” and select our script

A screenshot of a computer

Description automatically generated

Let’s quickly test this. Enter play mode in Unity by hitting the play button at the top, this will automatically put us in Game view but since we have no movement controls on our hitbox we need to go back to scene view. From there click on the hitbox and use the arrows to move it so that it collides with the enemy cube. If successful so far then the enemy cube should be deleted.



A screenshot of a computer

Description automatically generated

Also try moving the hitbox into the player cube. As it doesn’t have the Enemy tag, you can see that the hitbox won’t delete the player.

A screenshot of a video game

Description automatically generated

So, our hitbox works but our player cube still isn’t attacking. We won’t our cube to attack when the player presses an attack button, that means we need to make the player cube be able to spawn the hitbox.

For that we need to make our hitbox a prefab, let’s create a new folder within our Assets folder in the project window and call it “Prefabs”

A screen shot of a phone

Description automatically generated

Now drag the hitbox object from the scene hierarchy into the Prefabs folder which will automatically turn our hitbox into a prefab. This way our hitbox does not need to be in the scene at the start but we can still spawn it in. You can now delete the current hitbox from the scene if you wish.

Let’s create another new script, this time for our player controls. Let’s call this one “PlayerController”.

The easiest way to test our code later on would be to already have the player cube move based on player input, however as this is not the focus of this tutorial I will not be explaining how to do that here but if you aren’t sure he is some code you can use for basic movement that will allow your player cube to move left and right when the arrow keys are pressed. Simply copy this into your “PlayerController” script.

A screen shot of a computer program

Description automatically generated

Now, for attacking. First, we need to set up an if statement that will detect the player has pressed the attack button. Let’s use the spacebar as the attack button for now, so we should write “if (Input.GetKeyDown(KeyCode.Space))” within the Update function. Which will check for when the player pressed the spacebar.

A black screen with white text

Description automatically generated

Make sure you use “GetKeyDown” and not “GetKey” as we only want the attack to trigger once when the spacebar is pressed whereas “GetKey” will result in an attack triggering for each frame the spacebar is held down.

Now we need to be able to call the attack each time this key is pressed. In theory we could simply write our code for that within this if statement however it will be much easier to code this as a separate function that we can call within the if statement.

We will need a couple of variables for this so let’s return to the top of our code and add some. We need to write “public GameObject playerBody;” and “public GameObject attack;”. These variables will allow us to assign objects within our scene to be stored within these variables via the inspector. We will use playerBody to store the player cube and attack to store the hitbox prefab.



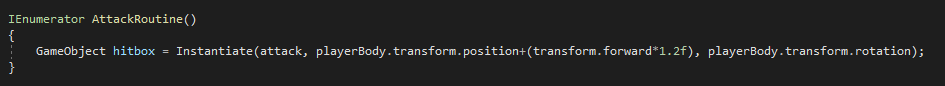
Now go back to the bottom of our code and outside of the update function let’s create an IEnumerator function for us to call when attacking. We’ll simply call this one “AttackRoutine”

A black screen with white text

Description automatically generated

The first thing we want out IEnumerator to do is to spawn in the hitbox which we are storing within attack for this we will use Instantiate but as the hitbox is designed for a melee attack we want to make sure it spawns just in front of the player cube.

Here’s what we need tov write within the IEnumerator: “GameObject hitbox = Instantiate(attack, playerBody.transform.position+(transform.forward\*1.2f), playerBody.transform.rotation);”



Now this looks like a lot so let’s break it down really quickly.

The “Instantiate” command is spawning the object stored in the attack variable, which in this case is our hitbox, and temporarily storing it under a variable called “hitbox”

However, we still need to tell that hitbox where to spawn which is what the rest of the code within “Instantiate” is for. “playerBody.transform.position” will spawn it at the same position as the object stored within playerBody which will be our player cube. However, this will spawn it directly on top of the player cube which we don’t want so we add “+(transform.forward\*1.2f)” to spawn it just in front of the cube instead (You may need to adjust 1.2f to a different value depending on the size of your player cube, the size of your hitbox and how far in front of the player you want the hitbox to spawn).

Including “playerBody.transform.rotation” will ensure that the hitbox always spawns in front of the player cube regardless of which direction the player cube is facing.

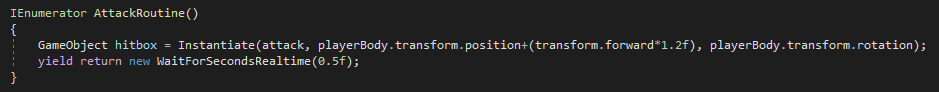
Now we need to actually call this IEnumerator to run when the attack button is pressed so let’s return to our if statement from earlier and within it write “StartCoroutine(“AttackRoutine”);”

A black screen with white text

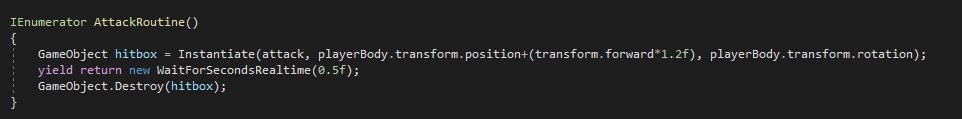
Description automatically generated

This would normally be a good point to test out code however the IEnumerator will give an error if it does not return a value so just this once let’s skip testing and just finish the code right now

The next line we need to add is within the IEnumerator and is “yield return new WaitForSeconsRealTime(0.5f);” which will cause the code to wait at this step for the specified amount of time (which is in seconds) before running the next line. You can adjust the numerical value if you wish. Just keep in mind that the number you put will be how long the hitbox is active in the scene for.



And finally, we want our hitbox to be destroyed from the scene after that time is up so let’s add one final line to the IEnumerator. “GameObject.Destroy(hitbox);” will deleted the object stored in hitbox, which in this case is the hitbox that it just spawned, from the scene and thanks to the prior line of code it will only delete it after the amount of time specified has passed.



Now let’s test the code again but first we need to make sure it is set up correctly.

Select the player cube and select “Add Component” and then attach our new script to it.

A screenshot of a computer

Description automatically generated

Make sure you drag the player cube object from the hierarchy into the player body variable listed under the script in the inspector. Also be sure to do the same with the hitbox by dragging it from the prefabs folder into the attack variable. If done correctly it should look something like this

A screenshot of a video game menu

Description automatically generated

If you test this code now you should find that when the spacebar is pressed the hitbox will spawn in front of the player and de-spawn again after a set amount of time. You can now move the player over towards the enemy cube and attack it to delete it.

A group of cubes in a flat area

Description automatically generated

A white and red cubes in a flat area

Description automatically generated

You may notice that the hitbox is static and will not move alongside the player for its duration after it is spawned. It is up to you whether you want the hitbox to move or not but if you do I recommend simply using the same movement script on the player cube for the hitbox so it’s movements will match the player objects. It is better to do this by copying the movement code into either a new script and applying it to the hitbox or putting the code in the “AttackHitDetection” script already assigned to the hitbox as attaching the “PlayerController” to the hitbox will result in the hitbox spawning more hitboxes when the spacebar is pressed.

Congratulations. You know have a functioning attack. As mentioned, before you could combine this system with a health system to create enemies that take multiple hits to defeat or experiment with different types of attacks with varying size hitboxes and different amounts of damage.

Thanks for reading.