Tutorial #3 Melee attacks

**Setting up the scene**  
Melee attacks could be an essential part of a game that someone would want to work on, so how would someone want to make an attack system that actually deals some form of damage? In this tutorial we’ll be taking a look at Melee attacks and how we can get damage to be detected by running a debug.Log function to tell us this. First things first, we’ll be setting up our scene once again.

Graphical user interface, application

Description automatically generated

A picture containing text, electronics, screenshot

Description automatically generated

Next what we want is to do the same thing as before and make two squares so right click in the hierarchy and click 2D Object>Sprites>Squares. One square be big and stretched out as our ground seeing as we’ll be needing one and one a standard size as our player. You should have something that looks like this. Graphical user interface, application

Description automatically generated

Once we have this be sure to add a Rigidbody 2D to the player as well as a Box Collider 2D component to both objects in our scene so they function well. To do so just select an object and go into the object’s inspector on the right and click Add Component, once that’s done add a “RigidBody2D” and a “Box Collider2D” to the player sprite and a collider.   
  
Seeing as we’ll be making our player attack something let’s go ahead and make an enemy sprite in the same way, we’ll be using them to see if our attack manages to register or not.   
  
  
  
Now we have a scene we can use with another character. We need to start by creating a custom tag for the enemy. First things first, select the enemy object we’ve just created in our scene. Once selected go to the top right and go the “Tag” drop down menu. Graphical user interface

Description automatically generated

Next hit the drop-down menu and select on the “Create Tag” option. Now let’s make a brand-new tag and let’s name this Enemy. Once done go back and change the tag to “Enemy”. It should look like this.   
  
A picture containing text, device, clipart, screenshot

Description automatically generated

Now we can start coding. In the assets folder, right click and create a new C# script. Let’s start by naming it “Enemy” as this is what we’ll be using on our Enemy, when once hit it should trigger a debug message that tells us that the player was hit, here’s the script.  
A screenshot of a computer

Description automatically generated with medium confidence   
It’s fairly simple and straight forward, the script is telling Unity that when the enemy has been hit by the player then we need to make sure that we display the “Damage TAKEN” message on our main console.   
  
Now what’s next is that we make a script that goes along with our player script, seeing as we need something to initiate the attack for there to be results. Go back into the Assets folder and make a brand new script named “Attack”.   
Open it up and begin writing the script, here’s a screenshot of what the code looks like: Text

Description automatically generated

Here we can clearly see that the code is asking for several things, first going down we’re asking Unity to look for various things for example a “WhatIsEnemies” tag, the attack position (which is where the attack will come from) as well as the attack range (this being how much space the attack can cover.

The Private void OnDrawGizmosSelected section allows us to have a visual indicator of how big our hitbox will be when we adjust the attack range.

In terms of the functionality, in order to initiate an attack I’ve chosen to press the “P” key, to get an attack out. This then will call the other script we have that will detect if the enemy’s been hit if so our debug message should appear on the bottom left of the screen.   
  
Next we need to make sure that we have everything set up. First things first let’s drag the attack script on to our player. Now here you’ll see a lot of public variables in the inspector when you select the player.

Below you can see that there’s Attack range, Attack Position, Damage and WhatIsEnemy. Look familiar? They should that’s what we implemented in our code. So let’s go step by step to work out how to get each component working.

Graphical user interface

Description automatically generated

First “WhatIsEnemy” let’s click on the Enemy sprite that we have, go into the “Layers” tab in the inspector.

Graphical user interface

Description automatically generated

Now click the drop-down menu and change make a new Layer and name it WhatIsEnemy. When you go back, make sure to change it in the inspector. Then head back to the player object and look back in the Attack script area from earlier, and where it says “WhatisEnemies” change the layer to “WhatIsEnemy” now we have enemy detection.

**Graphical user interface, application

Description automatically generated**

Now Let’s make an attack position, this is where we’ll have our area of attack, so in order to make a point Right click in the hierarchy, and press “Create Empty”.

Graphical user interface, application, Teams

Description automatically generated

Now we’ll have an empty game object A picture containing chart

Description automatically generated

Graphical user interface, application

Description automatically generated

Next what you need to do is go into the inspector for this empty Gameobject, and select the 3D cube, these give you some visible tags you could use.

Graphical user interface, application

Description automatically generated

Select any one of these options.

A screenshot of a computer

Description automatically generated with medium confidence

Now move the Gameobject just outside the player sprite now we have an invisible point of attack. So when we press “P” we have a registered attack.

Graphical user interface, application

Description automatically generated

However, that’s not the case we need to assign it to it’s area first. Select the player and go back to the Attack script area. You can see a section that says “AttackPos” from our Hierarchy drag the Game object we made and put it into the AttackPos slot.

A picture containing diagram

Description automatically generatedUp next we now need to make sure we can see the range for our attack and this is the final one. So here we need to select the player and go back to the Attack Script area and if we drag the mouse left or right whilst hovering over the “Attack Range” we can change its size or we could type it in too. Let’s change it to 0.93 as it seems like a decent size. When we do we should see a massive red circle. This came as a result of the Gizmo function that we called earlier in our attack script. Allowing us to see a hitbox for our attack.

Graphical user interface, application

Description automatically generated

Everything seems to be in place so now we can test to see if it works. One thing we need to make sure is that the empty Game object we made earlier for the Attack position needs to be a child of the Player object this is because then the Attack point will move along side our player allowing for attacks anywhere the player is. To do this simply select the Game object in the hierarchy, and drag it underneath the player object so now it should look something like this.

Now everything seems to be in place let’s test this out, remember in order to see if this works we’re looking for the “damage Taken” prompt on the bottom left of A screenshot of a computer

Description automatically generated with medium confidenceour screens let’s see if it shows up.   


Excellent it works! And we can even see the small debug message in the corner too meaning this was a successful task.