**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 change the scale and position of game objects within the unity editor

In this tutorial, I will be showing you how to create a basic health system using triggers and tags in the Unity Engine. The aim of this tutorial for a game objects health to be reduced when entering a trigger that damages them as well as increasing their health when they enter a trigger that heals them.

We will start by creating a cube within our scene. This will act as our player object whose health we will be adjusting for this tutorial, so let’s name this cube “Player” for simplicity.

Before our cube is ready to activate triggers, we need a Box Collider so that it can collide with triggers. Cubes are created with box colliders but if you ever have a cube that is missing one here is how to apply one. Click on our player cube and look over to the Inspector on the right, within the inspector click the “Add Component” button. That will open a smaller window, within that window we need to search for and select “Box Collider”

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

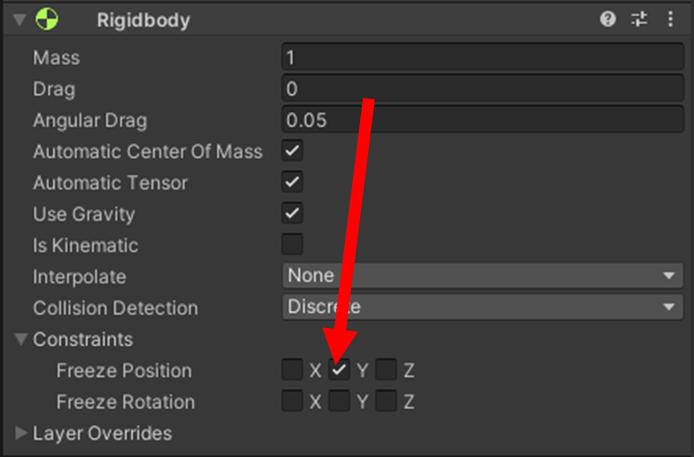
We’ll also need to add a Rigidbody to our cube so that it can activate triggers. So let’s add one of those as well. Same as before, click the “Add Component” button, but this time we want to select “Rigidbody”

A screenshot of a computer

Description automatically generated

Now that our cube has a Rigidbody it will be affected by gravity but as there is not ground below it within the scene it will just fall forever when we go to test our code later. We can avoid this by disabling “Use Gravity” on the Rigidbody within the inspector, alternativly, freezing its position on the Y axis will also work. Within the Inspector, either uncheck the box labled “Use Gravity” or open the “Constraints” dropdown under “Rigidbody” and tick the box labeled “Y” within “Freeze Position”.

A screenshot of a computer

Description automatically generated

Now that we have our cube set up, we can get started on the code. Let’s create a new script for the health system, you can name this script whatever you like but for this tutorial let’s call it “PlayerHealth”. Once you have created your script, double click it to open it in Visual Studio.

To start with our player needs a health value for us to edit so let’s create a variable to store their health. Let’s make the variable public so that we can edit it in the Unity inspector. As the player’s health will be stored as a numerical whole value we will be using an integer for this variable. We’ll also name the variable something simple so let’s go with the name “HP” and give it a default value of 10 for now.



To assign this to our cube let’s return to the Unity editor, select our Cube, then once again click “Add Component” and add our new script to it.

A screenshot of a computer

Description automatically generated

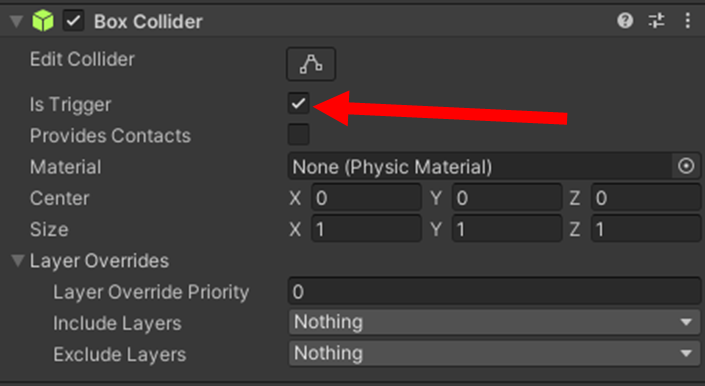
We now have our health value, but we need a way for it to change during gameplay, let’s start with taking damage. We now need a trigger that will damage our cube when it enters it so let’s return to our scene in Unity.

Within our scene, let’s create an empty game object that we will assign our trigger to, let’s call this one “Damage Trigger”. We need another Box Collider on this empty object so once again, select the new empty object and in the inspector for it we will click “Add Component” and assign it a “Box Collider”.

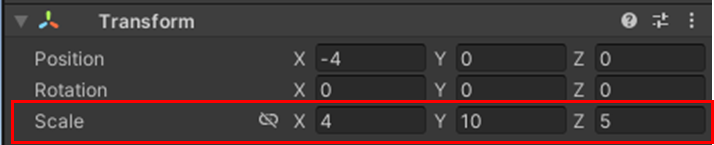
A screenshot of a computer

Description automatically generated

This time though we need to turn this collider into a trigger. Once we have our box collider on the object, we want to make sure that the “Is Trigger” checkbox is ticked otherwise the method that we will be using later in the code will not work.



Once a box collider has been assigned to the object, you’ll be able to see collider’s edges, use this to adjust the size of the object to your liking, for now let’s do this through adjust the scale in the inspector.

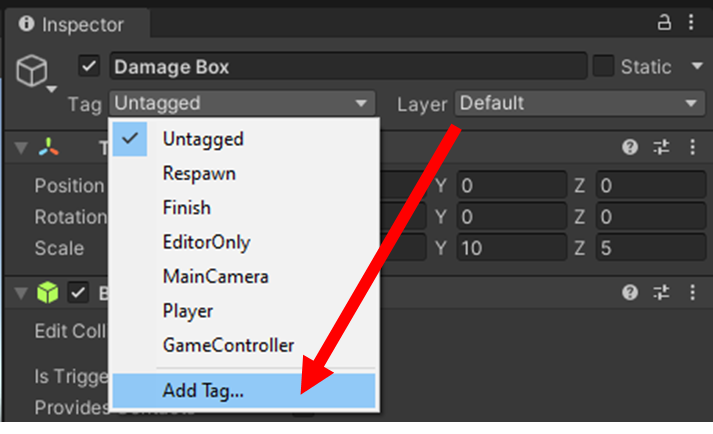
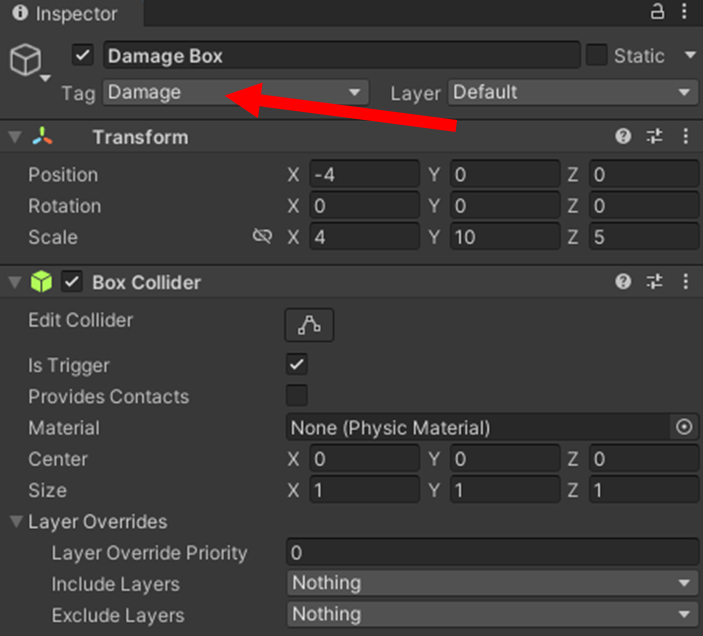


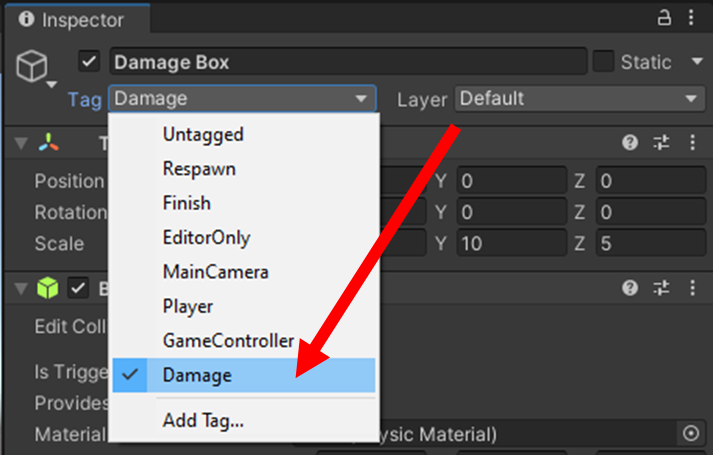
We also want to make sure that our cube does not start already within the trigger so let’s move it to the left of our cube for now.

A screenshot of a video game

Description automatically generated

Lastly, we will need to assign a tag to our trigger so that our code will know whether to reduce or increase the player’s health. Select our Damage Trigger, open the Tags drop down and select “Add Tag…”. Here we can assign a tag to our Trigger, in an actual game you would typically use tags named “Enemy” or “Projectile” for this purpose but for the sake of this tutorial, let’s just call this one “Damage” and assign it to our Trigger.

A screenshot of a computer

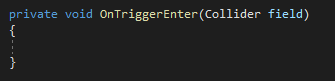
Description automatically generated

This part is not required for the code to work but to make things visually easier I am going to place a red plane below our Damage Trigger so that we can see where it is, feel free to do this too if you wish.

A screenshot of a computer

Description automatically generated

Returning to our script, we can now write the code to reduce the player’s health. First, we will need our player to know when they have entered the trigger and then reduce the so for this we will use “void OnTriggerEnter()” and within the brackets we will add “Collider field”. With this code whenever our cube enters the trigger it run the code written within the function and will temporarily store it under “field”, so we can reference it within said code.



First, we will need to check the tag on the trigger that we just entered before we reduce the cube’s health so let’s check that using an if statement. By using “if(field.gameObject.CompareTag(“Damage”))” the code will check the tag on the collider that is currently stored within “field”, which in this case is our Damage Trigger, and will only run the code within the if statement if it finds that the collider’s tag is our “Damage” tag.

A screen shot of a computer

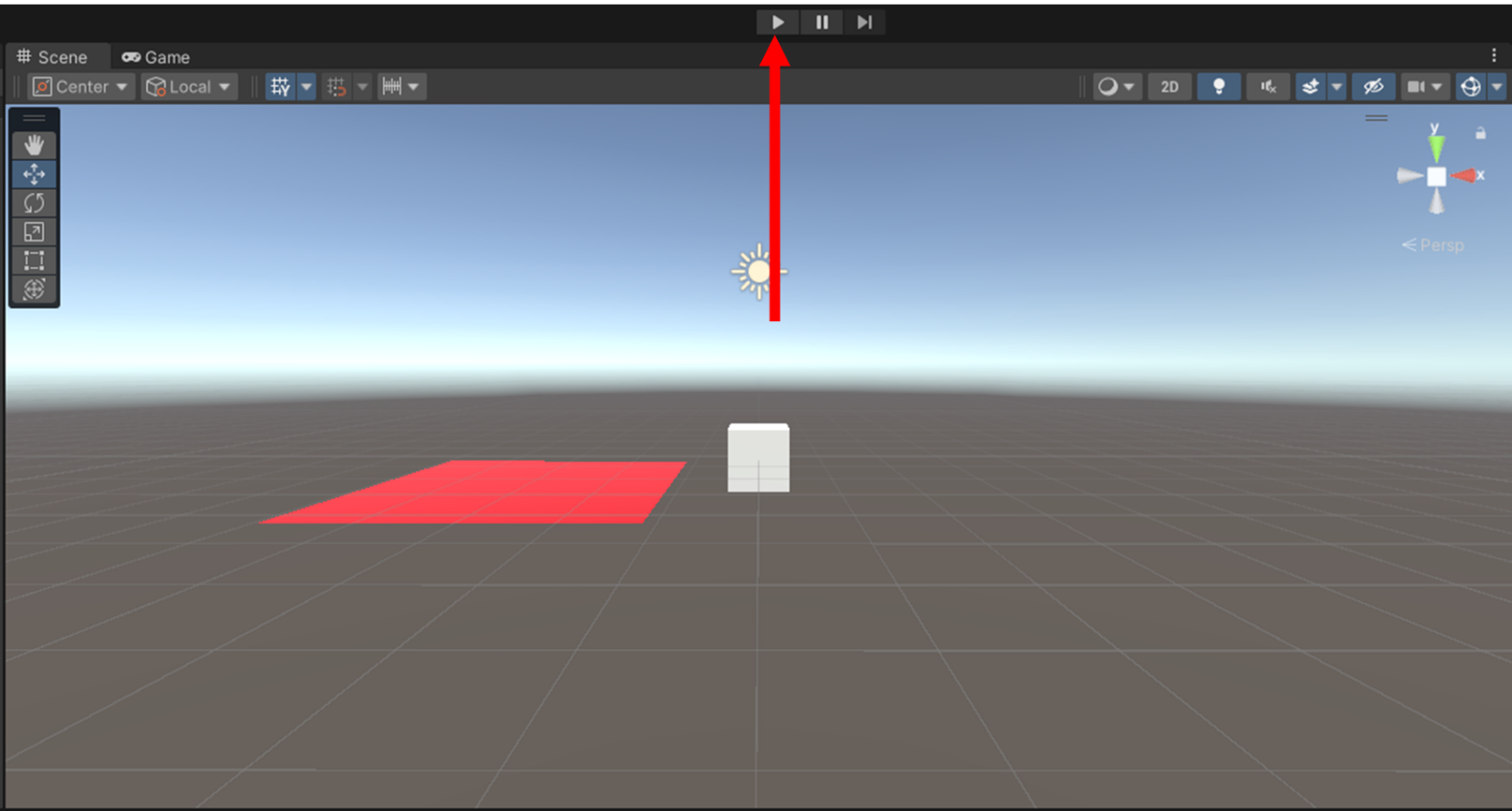
Description automatically generated

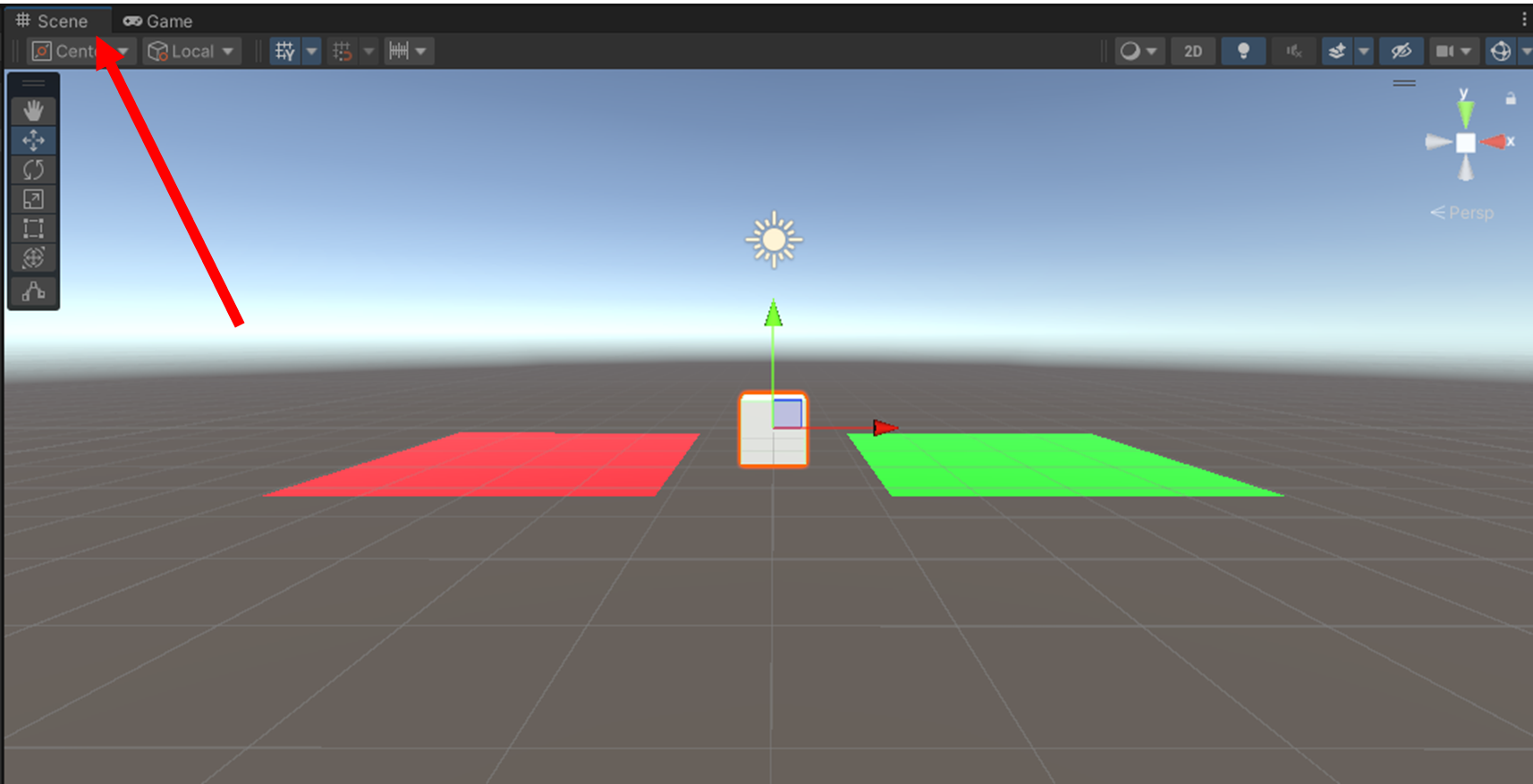
Then we will need to reduce the cube’s health by one which we can do by writing “HP -= 1;” within the if statement.

A screenshot of a computer program

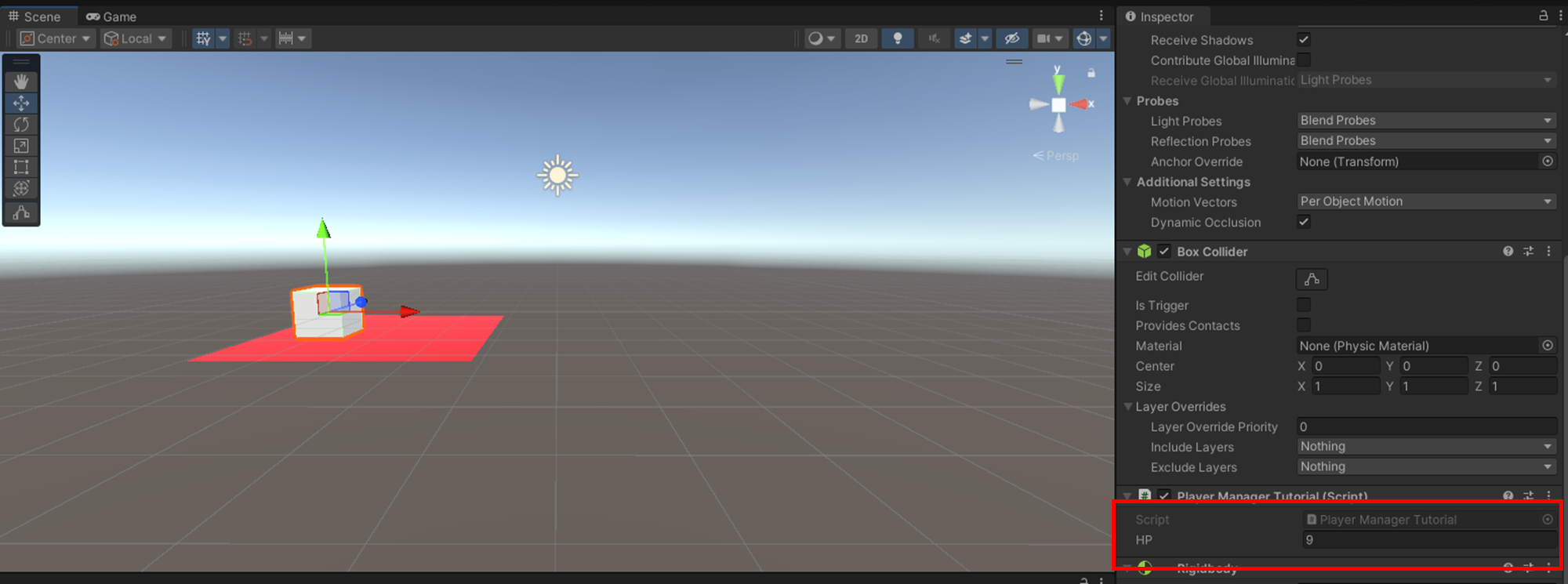
Description automatically generated

Let’s quickly test if this works by saving our code, returning to Unity, entering play mode and moving our cube within the Damage Trigger. Take a look in the inspector for the cube, if the code works as intended you should be able to see the integer stored in HP go down by one when the cube is moved into the Damage Trigger. (If you already know how to make an object move through player input feel free to do this step that way but if you don’t you can still test this but switching from game view back to scene view, clicking the cube and then moving it by clicking and dragging the arrows)









Now let’s work on restoring the cube’s health. For this we will need a new trigger so let’s create one exactly the same as before up until we assign it a tag. Once again, create an empty object, we’ll call this one “Heal Trigger”. On our new empty object, let’s “Add Component”, select “Box Collider” and make sure “Is Trigger” is checked. You can also move and resize this one as you wish so let’s move it to the right of our cube and I will also add another plane below the trigger for better visuals, this time I’ll make it green.

A screenshot of a video game

Description automatically generated

Now our new trigger also needs a tag, but we can’t use the same tag as before for this one otherwise it will just reduce the cube’s health again, so we need to create a new one. Following the same steps as before, let’s create another new tag and call this one “Heal”. Then we assign it to our “Heal Trigger”.

A screenshot of a computer

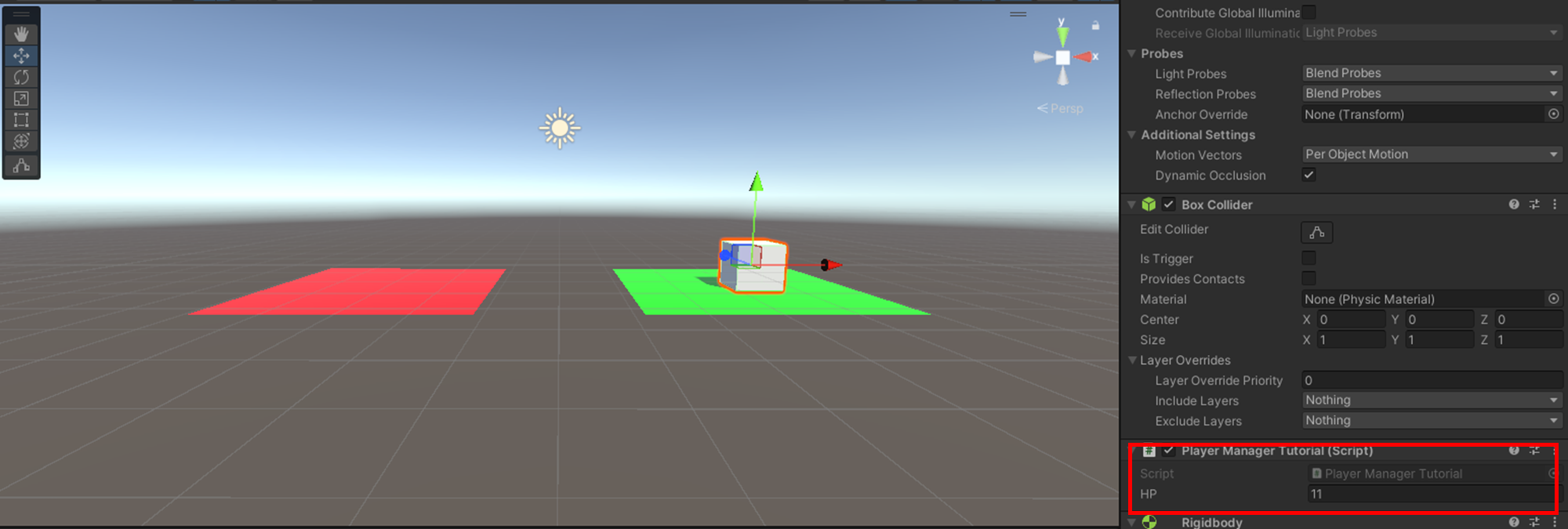
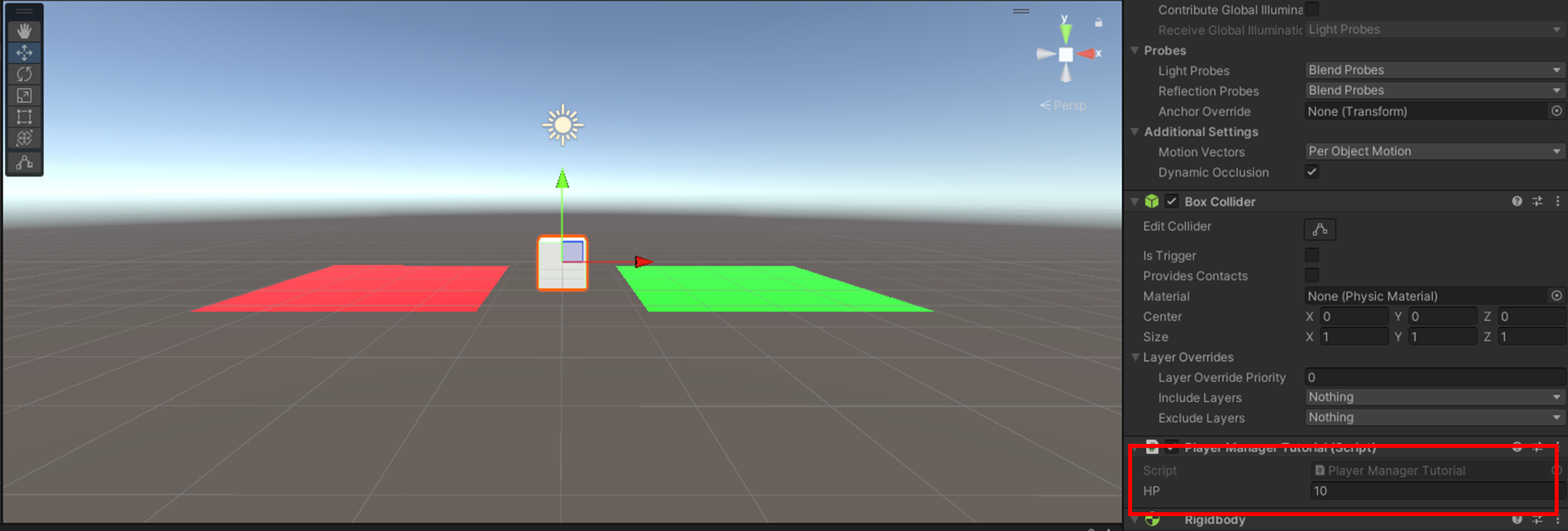
Description automatically generated

Now we need to add another if statement within our code to check for the new tag. So let’s repeat the same code as before but this time we should replaces “Damage” with “Heal” and “HP -= 1;” with “HP += 1;”

A computer screen shot of a program

Description automatically generated

Once again let’s return to our scene and test if this works the same way we did before. If it does, the HP value should increase when the box enters the Health Trigger.



Great, it works. But now we have a new problem. You can see that our player’s HP has increased to 11 but our initial HP value was 10. We now have more health than we started with. Under normal circumstances, in a game a player shouldn’t be able to increase their current health over their maximum HP so let’s implement that now, shall we?

Back to our code, we now need another variable for maximum HP but we also need to make sure our HP always starts at max when the game starts. We could simply make their values the same in our code but what if your game has a levelling system and the player’s max HP increases when they level up? If we just write the value in our code the current HP won’t increase with it unless you write more code to increase that as well. Instead let’s simply write some code assign the value of max HP to the current HP when the game is started.

To prevent confusion with our variable names let’s rename our current HP variable to “maxHP” and then create a separate new variable for “HP”. This time we won’t assign a value to HP immediately and instead leave it blank for now.



But if HP remains blank then the player starts with 0 HP. So within our “Start()” we should right “HP = maxHP;”. This means that everyime the game starts, the code will assign the player’s maxHP as their current HP.

A black background with white text and black text

Description automatically generated

A screenshot of a video game menu

Description automatically generated

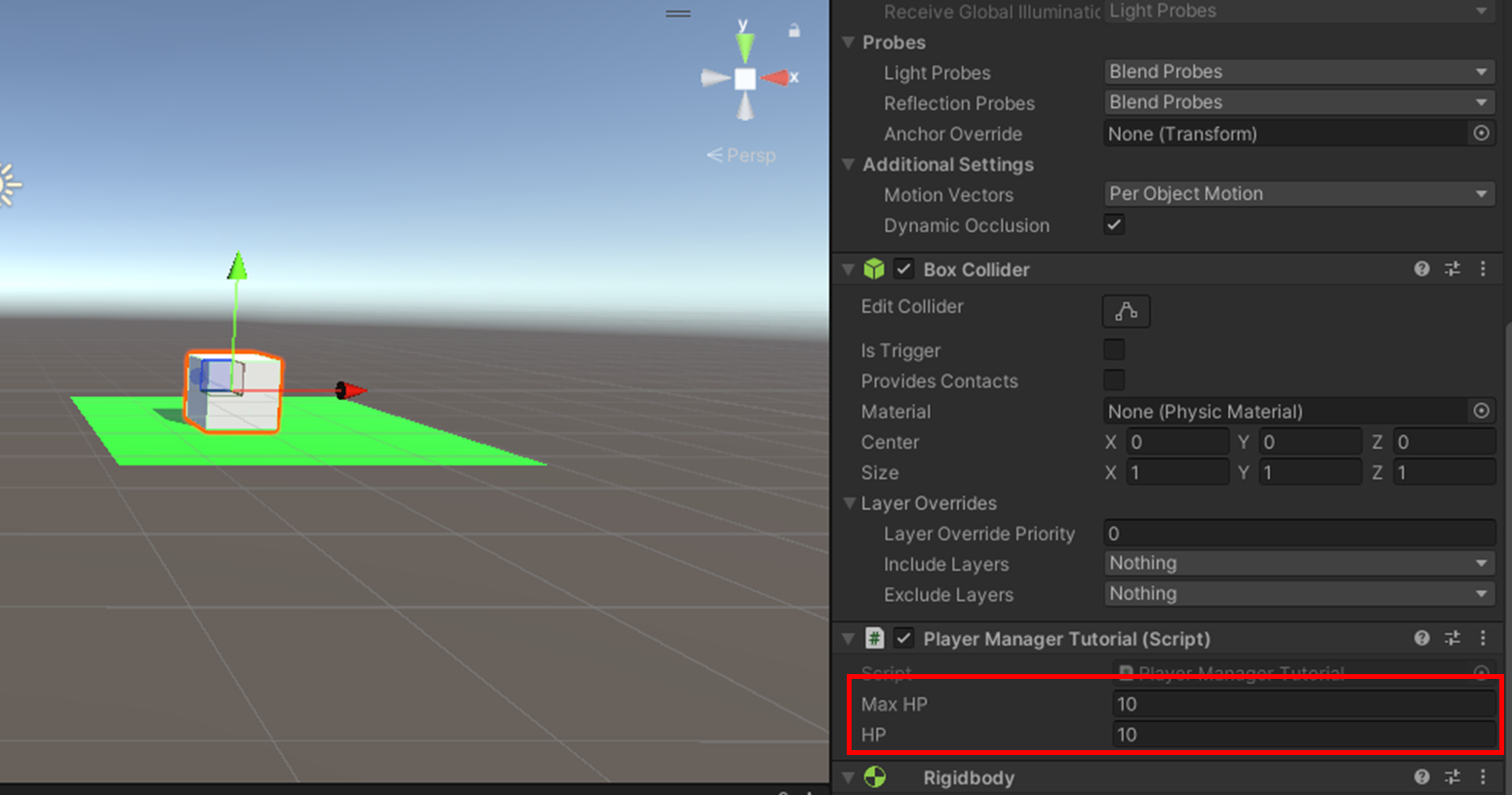
As long as the code we wrote previously is still decreasing or increasing HP and not maxHP, our player can still heal and take damage while their maxHP will remain unaffected, feel free to go back to our scene and try it out to see for yourself.

But this still doesn’t fix our initial problem. What we need to do now is make it so that touching the health trigger won’t increase our health above the max. So we need to add another if statement to check that HP is not already at the maximum. Within our previous if statement for healing we should add another that says “if (HP < max)” and then move our “HP += 1;” line into that new if statement. It should now look like this.

A screen shot of a computer

Description automatically generated

Now if you return to the scene and try again, our HP should not increase when you move the cube into the health trigger unless HP is below maxHP



That’s that problem fixed. Except we actually have a similar issue on the other side. As the code is currently, our player can keep taking damage even when their HP is already at 0 and their HP can go into the negatives, so we need to fix that too

Unlike when HP is at maxHP, we don’t want the player to be able to continue when their HP hits 0 as that is supposed to be their lose condition. What exactly you want to happen when the player gets a game over is up to you but for now let’s just destroy our player cube when they lose.

In our code let’s add another if statement within our previous damage detection one, this time however we want to place our new if statement after we decrease the player HP as we still want the HP to go down and then check if it’s either at or below 0 so let’s write “if (HP <= 0)”(in this scenario is should never be below as we are only decreasing the HP by one each time but in an actual game with attacks that can deal more than 1 damage to the player the HP may go below 0)

A computer screen with white text

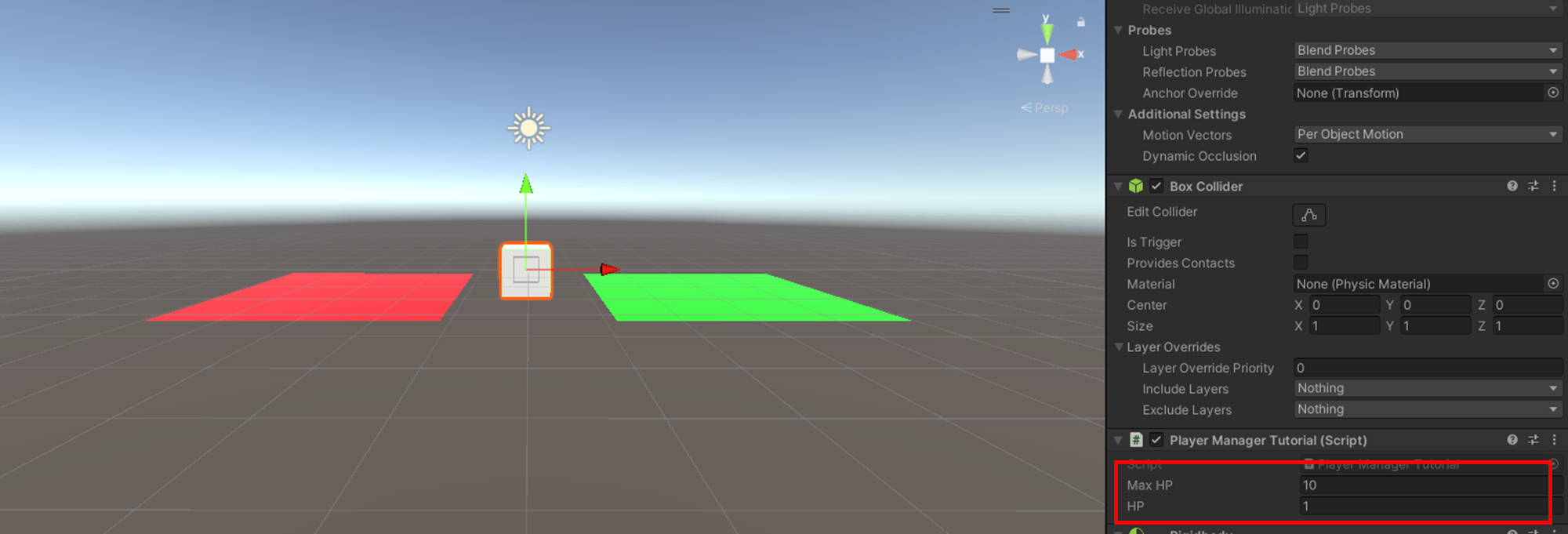
Description automatically generated

Within that if statement, let’s add a line of code that destroys the player if their health is at 0 or below. We want to type “GameObject.Destroy(gameObject);” which will destroy the object that the script is attached to, which in this case is our player. When you’ve finished your entire script should look like this.

A screen shot of a computer program

Description automatically generated

Now let’s test this one last time and see if the player is destroyed.



A screenshot of a computer

Description automatically generated

As you can see, as soon as the player’s health hits 0 they are deleted.

Congratulations. You know have a working basic health system. Of course, you can expand upon this and apply it to actual game objects like enemy bullets or health pickups and have them deal different amounts of damage or restore different amounts of HP. If you are confident with UI you can also use this system to create a health bar but that’s a topic for another time.

Thanks for reading.