**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 spawn and destroy objects within your scene on a timer through the use of floats and variables. The aim for this tutorial is to spawn in items at set intervals and have them destroy themselves after a set amount of time.

Let’s start by creating a cube within our scene, this will act as the object we want to spawn in. We will need to apply a script to this cube later but for now we don’t need to do anything to it just yet so simply name it what you want and save it as a prefab so that we can spawn it in later.

To do this, create a new folder within Assets in our project window and name the folder “Prefabs”.

A screen shot of a phone

Description automatically generated

Now simply drag the cube from the hierarchy into the prefabs folder to save it as a prefab for later. You can remove the cube from the scene if you wish as we no longer need it there.

Now let’s create an empty object within the scene, this will hold our spawning script so let’s call it “Spawn Manager”.

Now we need to make the script for spawning the cubes into our scene. Let’s create a new script, call it “Spawn Timer” and open it.

First things first we need the script to know what it is supposed to spawn in so let’s create a new variable for a game object by typing “public GameObject spawnItem;”



Our aim is to set up a timer so that the cubes will spawn at intervals rather than every frame so we do not want to place our Instantiate command directly within the Update() function.

Let’s instead create some additional float variables to set up the time. Let’s call them “spawnRate” and “spawnTimer” with spawnRate being a public variable and spawnTimer being a private one.

A black background with white text

Description automatically generated

spawnTimer is what we will use to count the time between spawning in objects to we should set it to 0. spawnRate will be how often we want cubes to spawn in, I have set mine to 3 meaning cubes will spawn roughly every 3 seconds, with some variation depending on how many FPS the game is running. spawnRate is a public variable so if you want to create multiple spawn points with different spawn rates you can adjust the value within Unity’s inspector.

So now we have the first part of our timer set up but we need the timer to actually tick up.

Within out Update() function let’s gradually increase the value of spawnTimer.

A black background with white text

Description automatically generated

This line of code will gradually increase spawnTimer as the by one every frame, this isn’t exactly what we want but we will use this to test that our spawning it is working and then adjust it later.

In the next line we want to type out an if statement that will check if spawnTimer has reached the value of spawnRate.

A computer screen with white text

Description automatically generated

This if statement will only run the code within it if spawnTimer greater than or equal to the value of spawnRate. As this is checked every frame and spawnTimer is only increasing by 1 each time, “spawnTimer == spawnRate” should result in the same outcome however I prefer to use >= as a failsafe just in case an unforeseen problem occurs, this way the code will still run even if spawnTimer is for some reason greater than spawnRate.

Within this if statement we want to instantiate our cube. We will write a line code to instantiate the cube at default position.

A computer screen shot of a program code

Description automatically generated

We also need to reset the spawnTimer otherwise it will now continue to spawn more cubes every frame. This is simple enough and just requires assigning spawnTimer the value of 0 again

A computer screen with white text

Description automatically generated

Let’s go test our code. But first we need to make sure everything in the scene is set up properly

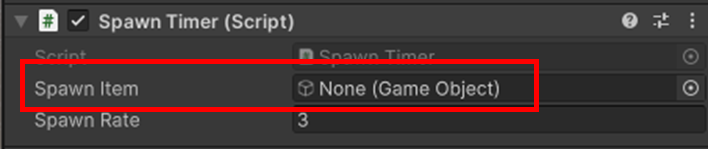
First let’s attach script to the Spawn Manager. Select Spawn Manager in the hierarchy, look over at the inspector to the right and click on “Add Component”, then select our script.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Our script still has nothing assigned to the Spawn Item variable as you can see through the inspector



Let’s fix that by assigning our cube prefab to it. Simply drag it from the project window onto the spawn item variable in the inspector, once done is should look like this

A screenshot of a computer

Description automatically generated

Now it’s time to test by clicking the play move button above our scene

A screen shot of a video game

Description automatically generated

If the code work you should see a cube in your scene BUT if you look to the Hierarchy you’ll see that there are actually a LOT of cubes in the scene and have all spawned in the same spot making it look like there is only one cube

A computer screen shot of a white cube

Description automatically generated

This is because with how our code is currently set up a cube is being spawned every three frames, while this is technically at set intervals which is what we set out to accomplish, this is not sustainable. 3 frames is a small fraction of a second resulting in many cubes even within a single second and with these cubes being spawned continuously it could very easily overwhelm the system and crash unity which we don’t want. So let’s go back and fix our timer.

This is a simple fix, on our line of code where we are increasing spawnTimer’s value, let’s just change 1 to Time.deltaTime. This way spawnTimer will still increase but it will do so gradually. Time.deltaTime refers to the time between the current frame and the last in seconds and with how fast frames go by, the value of Time.DeltaTime is normally incredibly small. In Theory, increasing the spawnTimer by Time.deltaTime as opposed to by 1 each frame means that the value of spawnTimer should not match the value of spawnRate until roughly 3 seconds have passed, spacing out our spawns by a lot more.

A computer screen with white text

Description automatically generated

We can now test our code again, this time there should not be any cubes in the scene when we first click on the play button, but wait three seconds and one will spawn in. If you keep your eye on the hierarchy and wait another three seconds you’ll see that another cube has spawned in.

A computer screen shot of a white cube

Description automatically generated

We now have cubes that spawn at set intervals. But before we move on, we want to actually be able to see that there are multiple cubes spawning within the scene and not just the hierarchy right? We could spawn the cube at a different position each time but that requires more code and I have a separate tutorial on that so instead let’s do this a different way.

Let’s edit our cube prefab. Simply double click on the cube in the project window to open up the prefab editor, any changes here will be saved to the prefab and apply to all cubes spawned.

A screen shot of a computer

Description automatically generated

Now let’s add a rigid body to this cube. Simply click the “Add Component” button in the inspector again and select Rigidbody.

A screenshot of a computer

Description automatically generated

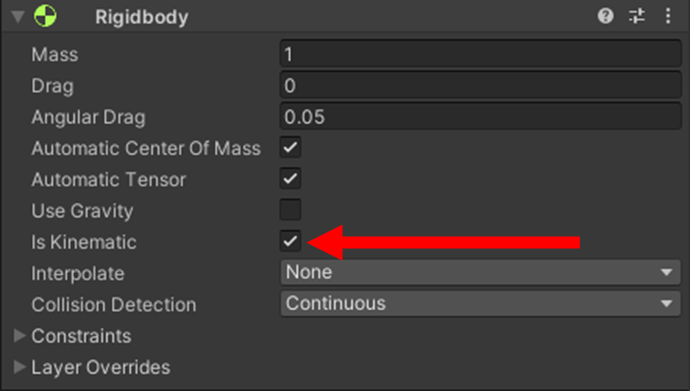
This will make our cubes effected by gravity and also prevent them from overlapping with each other so we can see each one drop from where it spawns, amusing right? But if they’re affected by gravity then they will just fall infinitely, and we’ll lose sight of them.

Let’s return to our scene add a plane to the scene to act as ground to stop them before they fall off screen. We also want to assign a rigidbody to this plane as well, same way as before. But this time let’s turn off gravity so that the plane won’t fall down as well. We can do this by selecting the plan and looking at the rigidbody settings in the inspector and unchecking “Use Gravity”.

A screenshot of a computer

Description automatically generated

Also make sure to check the Is Kinematic box so that the plane does not get pushed by the cubes



Lastly, we’ll need to adjust where our cubes spawn in the code so that they spawn above the plane. The following code will now spawn the cubes just above the centre of the scene where the plane is positioned. “Quaternion.identity” affects the item’s rotation, it is not important to understand for this tutorial however make sure you include it in your command.

A screen shot of a computer

Description automatically generated

Now if you test again. The cubes should fall onto the plane and you’ll be able to individually see each one.

A computer screen shot of a white cube

Description automatically generated

So we can spawn cubes at set intervals but once a cube is spawned it just sits there, eventually we’ll still have a lot of cubes. What if we want to get rid of a cube after it gets a little old? Well, we can accomplish this with the same timer setup that we just made.

Let’s create another new script and call it “DestroyTimer”. The let’s open it.

We’ll need some similar variables to the other script so let’s create two more float variables, a public one called “Cube Life” and a private one called “Destroy Timer”.



cubeLife will refer to how long a cube will live for before we destroy it, while destroyTimer is our timer and serves the same purpose as spawnTimer did, meaning we needs to increase it within the Update() function the same way we increase spawnTimer.

A computer screen with white text

Description automatically generated

Now we need another if statement, like before this one will check to see if the destroyTimer has reached the value of cubeLife.

A computer screen with white text

Description automatically generated

This time we don’t want to spawn an item once the cubeLife has been reached, we want to destroy an item so let’s write a line of code to destroy the item that the script is attached to

A screen shot of a computer program

Description automatically generated

This time we won’t need to reset the timer as the object it is attached to will be destroyed anyway. Now we need to attached this script to our cube so open the cube in a prefab editor once again and attach this script to it via the ”Add Component” button

A screenshot of a computer

Description automatically generated

Now if you run the code one last time and wait 10 seconds after the first cube has spawned you should see that the cube will delete itself from the scene.

Congratulations. You can now spawn and destroy object within your scene at set intervals. If you have read my other tutorial on spawning items in random locations you can combine this tutorial with it to create items that spawn at set intervals but in random locations. These timers can also be used of other things to such as making enemies in a game jump at set intervals.

Thanks for reading.