Don’t Destroy Objects on Load

By Nathan Stalley

In this tutorial, we will be making a universal script that can be applied to any game object, the script will keep the game object between different scenes and will not destroy it when a new scene is loaded up.

To start off with, right click in your assets folder and create a new C# script, we will call this one “DontDestory”. It is important to name it something that you will remember as we will be referencing the name of this script.

```
[HideInInspector]
public string objectID;

Unity Message | 0 references
private void Awake()
{
    objectID = name + transform.position.ToString() + transform.eulerAngles.ToString();
}
```

We will start by making a ‘public string’ and naming it ‘objectID’ this is going to be used to determine the name of the game object that we have this script attached to as we can use this script on many different game objects.

We will also use [HideInInspector] just above this public string, this is because we do not need to see this string in the inspector, but we need it to be public so that it can be accessed from outside of this script.

We then need to make a ‘private void Awake()’ function above the start() function. In here, we will type out the following:

```
objectID = name + transform.position.ToString() + transform.eulerAngles.ToString().
```

After we have this typed out, we need to move to our ‘private void Start()’ function and type out the following:

```
private void Start()
{
    for (int i = 0; i < Object.FindObjectsOfType<DontDestroy>().Length; i++)
    {
        if (Object.FindObjectsOfType<DontDestroy>()[i] != this)
        {
            if (Object.FindObjectsOfType<DontDestroy>()[i].objectID == objectID)
            {
                Destroy(gameObject);
            }
        }
    }
}
```

This piece of code is what is used to keep track of game objects and to not destroy them on load.

To start off with, we need to search the scene for any game objects with this script attached to them. We do that by using this 'for' statement, this is used to search through game objects and look for the script named DontDestroy, therefore it is important to name the script to something that's easy to remember.

```
for (int i = 0; i < Object.FindObjectsOfType<DontDestroy>().Length; i++)
```

Next, we need to destroy the game object, we do this by using 'Destroy(gameObject)'. The problem with this is that the game object will end up destroying itself as it uses the same name, so to get around this, we use the first if statement with '!= this'.

This basically says 'not equal to this script' this will then find a game object without the same name, the if statement under that then checks for the object ID and then proceeds to destroy the game object.

```
if (Object.FindObjectsOfType<DontDestroy>()[i] != this)
{
    if (Object.FindObjectsOfType<DontDestroy>()[i].objectID == objectID)
    {
        Destroy(gameObject);
    }
}
```

Once all of this code is written out, you can attach this script to a game object and that game object will persist different scenes and will not spawn any duplicate objects when you load between different scenes.