Nested Prefabs for Unity3d

Tutorial

Bento Studio

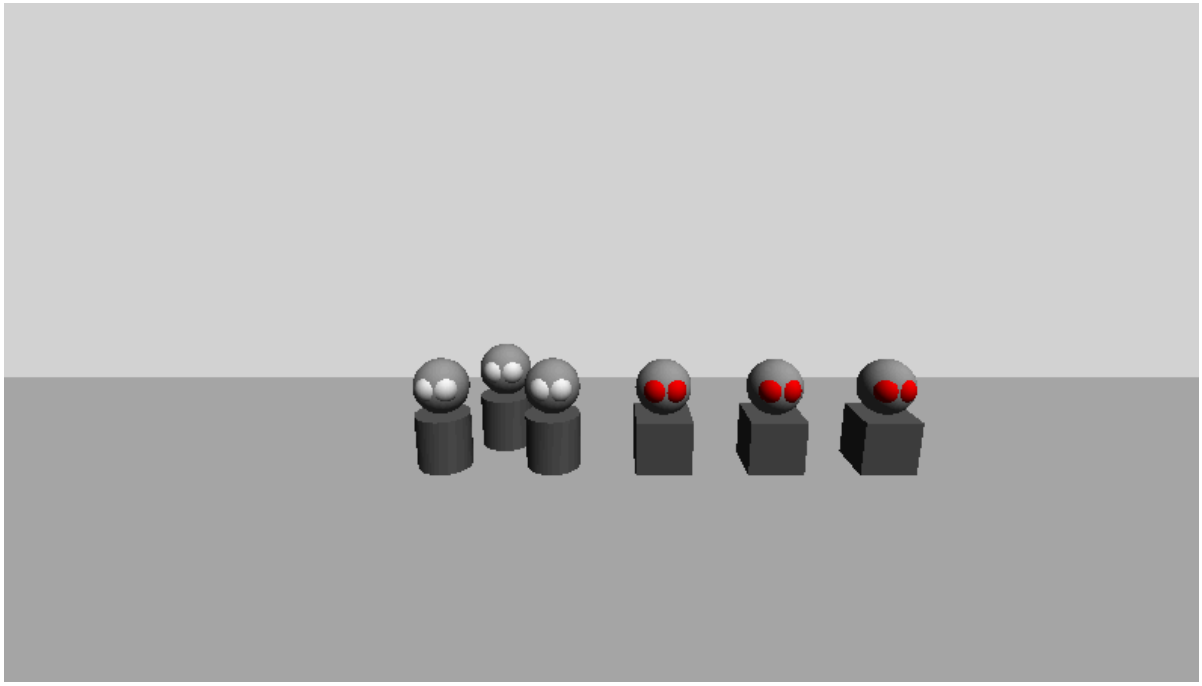
SUMMARY

This document is a tutorial to get you started with the Nested Prefab Editor.

Sample

If you want to see what will be accomplish during this tutorial, take a look at the sample scene.

You can find it at **NestedPrefab > Sample > scene_NestedPrefabSample.**



Reference

Once you have finished this tutorial it's recommended to look at the reference for more in depth information on the Nested Prefab Editor, its capabilities and limitations.

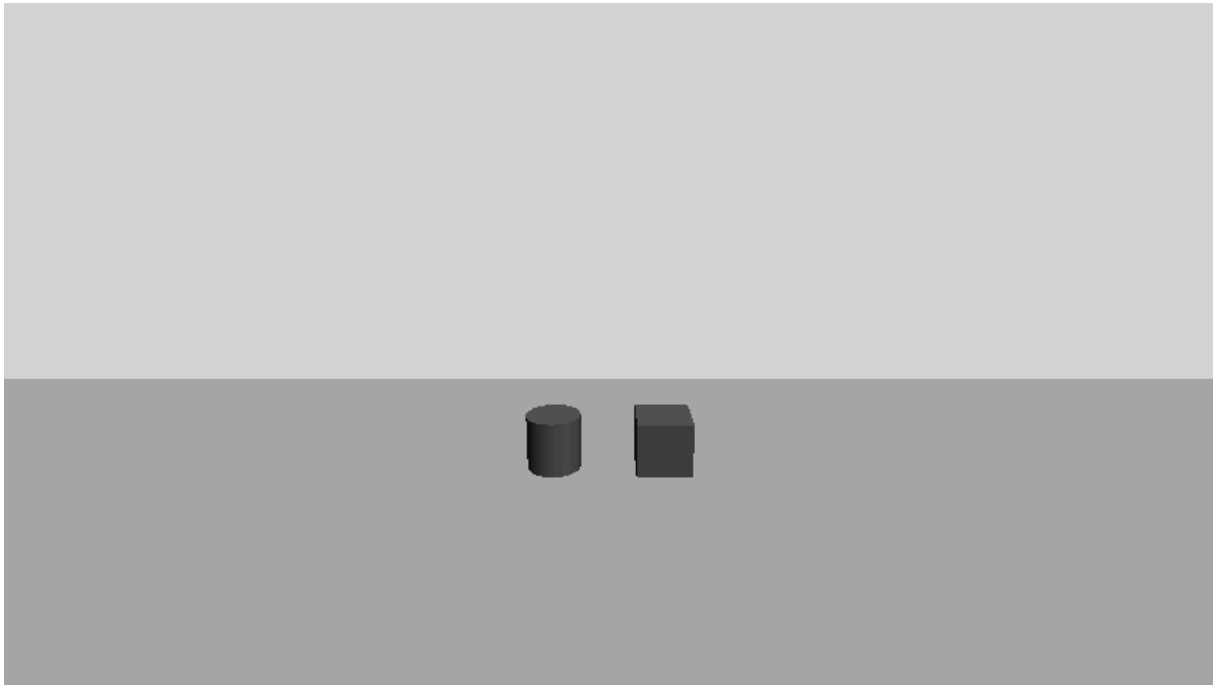
Edit a level with nested prefab

Create the scene

> Create a **Plane** for the Ground.

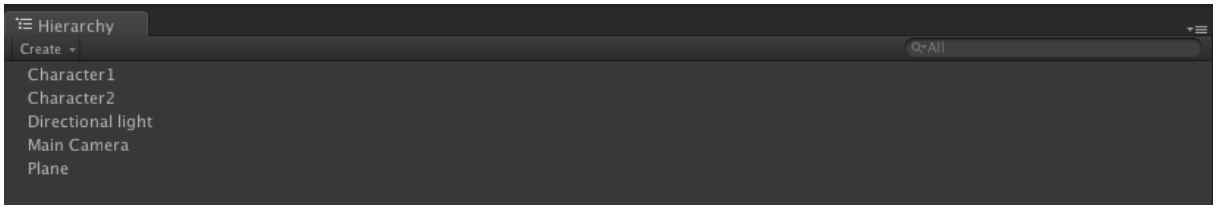
> Create a **Cylinder**.

> Create a **Cube**.



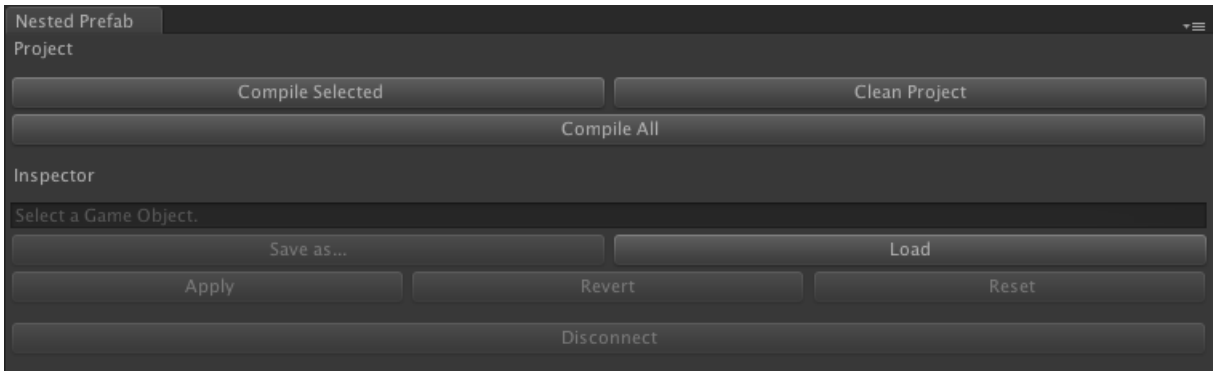
> Call the Cylinder **Character1**

> Call the Cube **Character2**



Open the editor

Click on **Nested Prefab > Open Editor** in the main bar to open the Nested Prefab editor.



Create your first hierarchical prefab

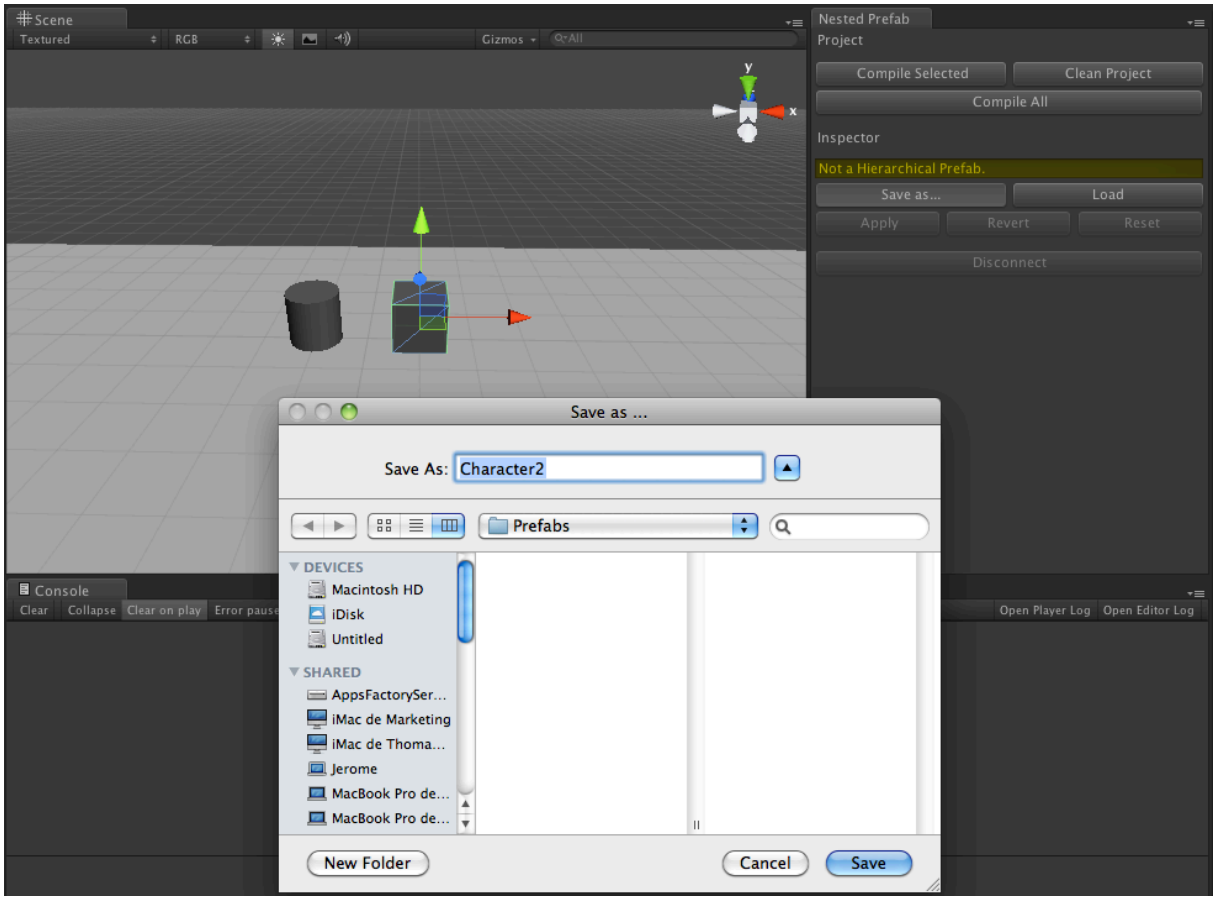
In order to nest prefab in each other you need special kind of prefab called hierarchical prefab.

> Select the **Character2** and click *Save as…*

> Choose the save location (Wherever you want, as long as it's under the Assets folder)

> Click Save

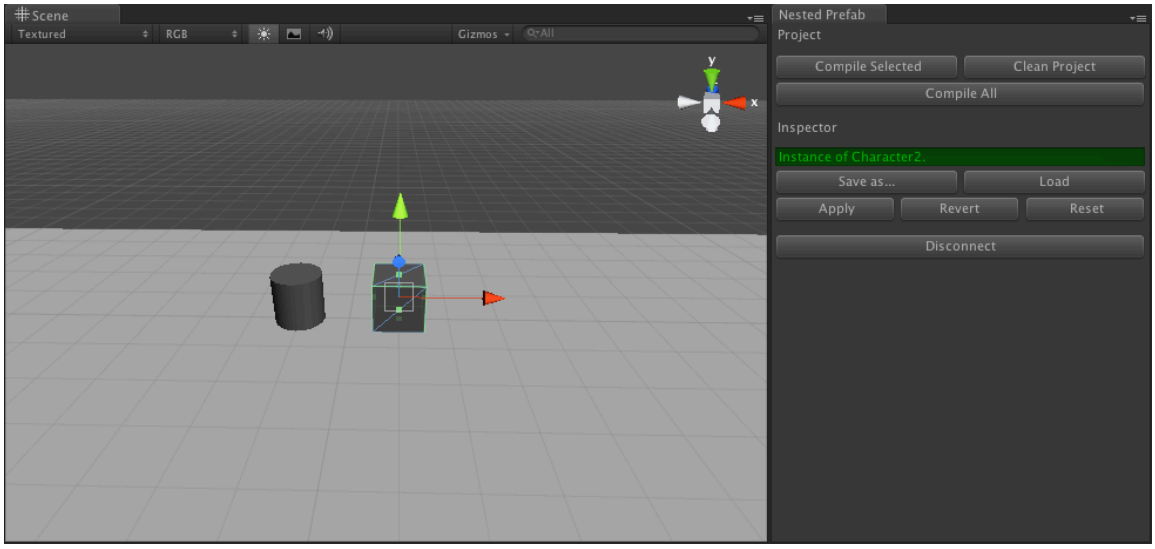
A hierarchical prefab look like a normal prefab except it has a **HierarchicalPrefabInstance** component.



As soon as you have saved an object into a hierarchical prefab it becomes an instance of this hierarchical prefab.

You can check whether a game object is a hierarchical prefab instance or not.

Just look at the **Nested Prefab Inspector**. If the selected object is a hierarchical prefab there is a **green text zone** that indicate which hierarchical prefab the object is an instance of.



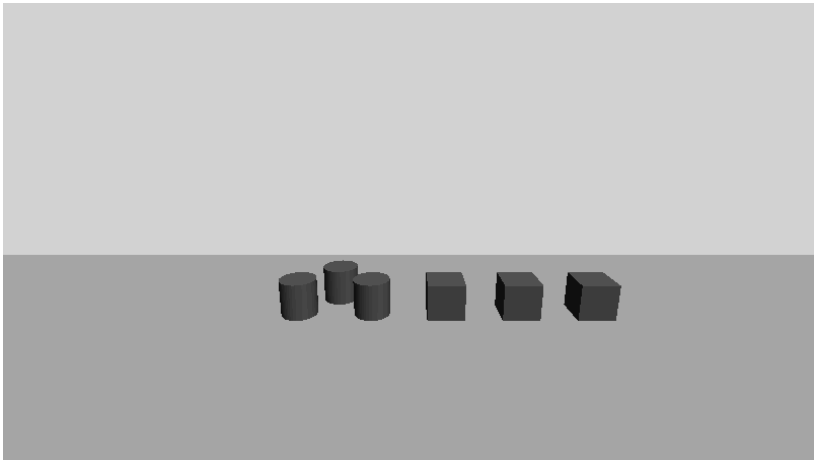
Duplicate the instances

Like the classic prefab you can duplicate the instances

> Try to press **Cmd-d** (or Ctr-d on windows) to duplicate a selected instance.

> Try to **Drag and drop** a hierarchical prefab from the **Project panel** to the **Hierarchy Panel**

We duplicate and move our instances to populate the scene, until it looks like the following screen.

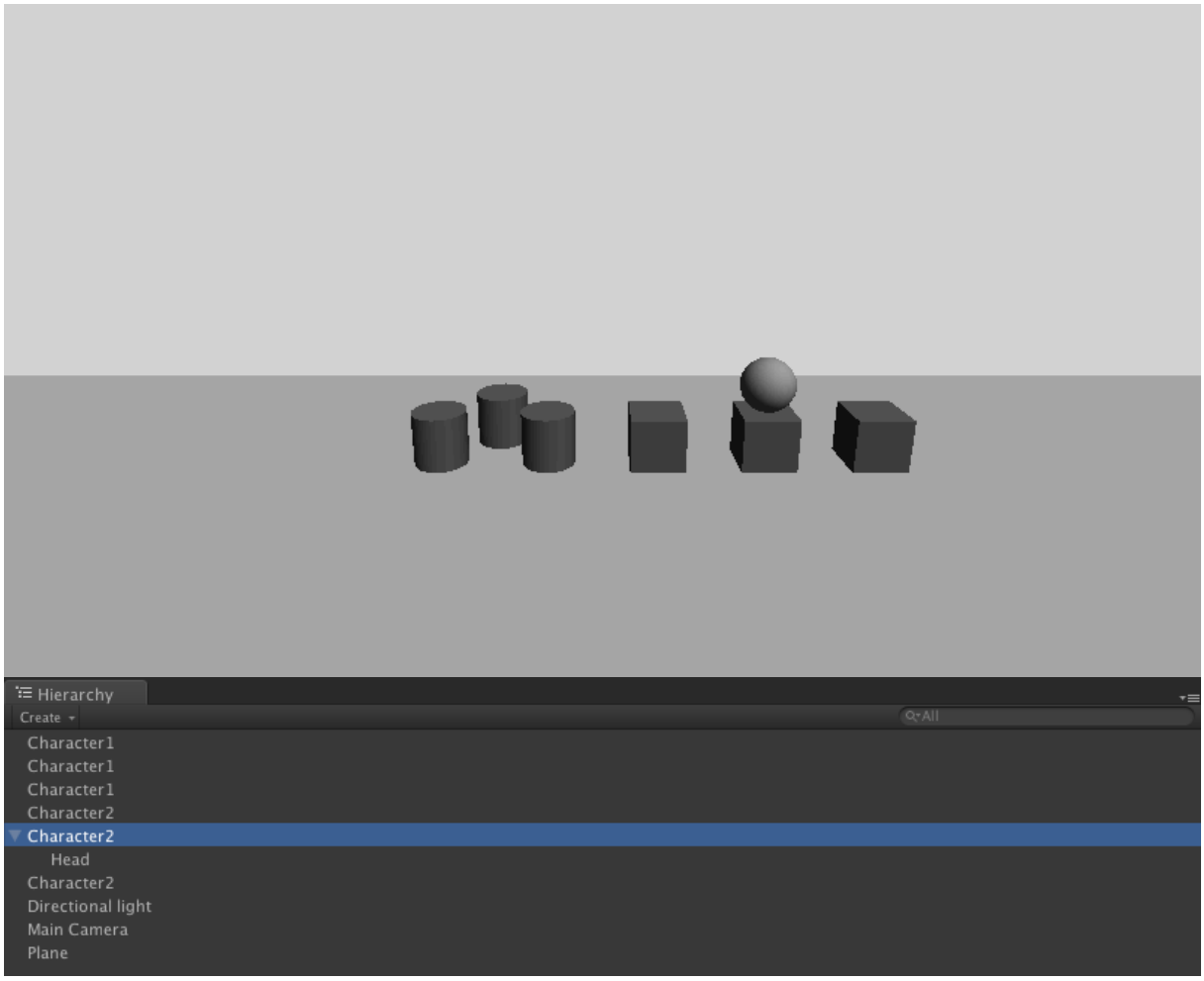


Propagate changes

The hierarchical prefab also has the welcomed ability to propagate his changes to all its instances.

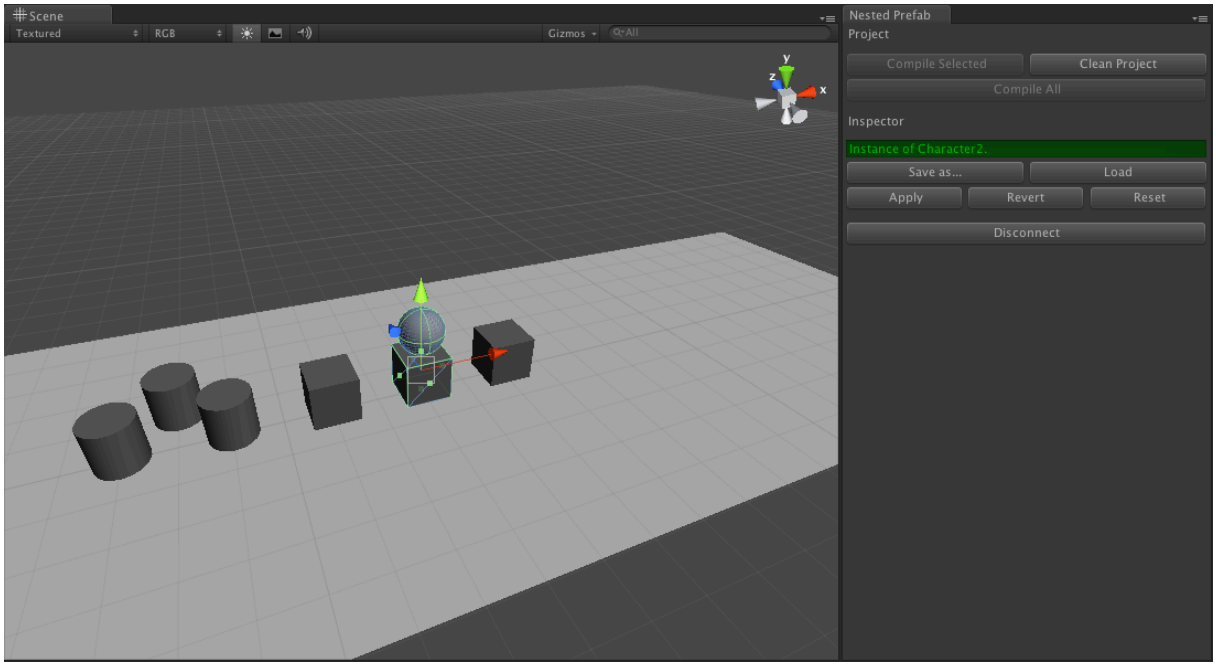
> Add a **Sphere** to an instance of **Character2**

> Call it **Head**

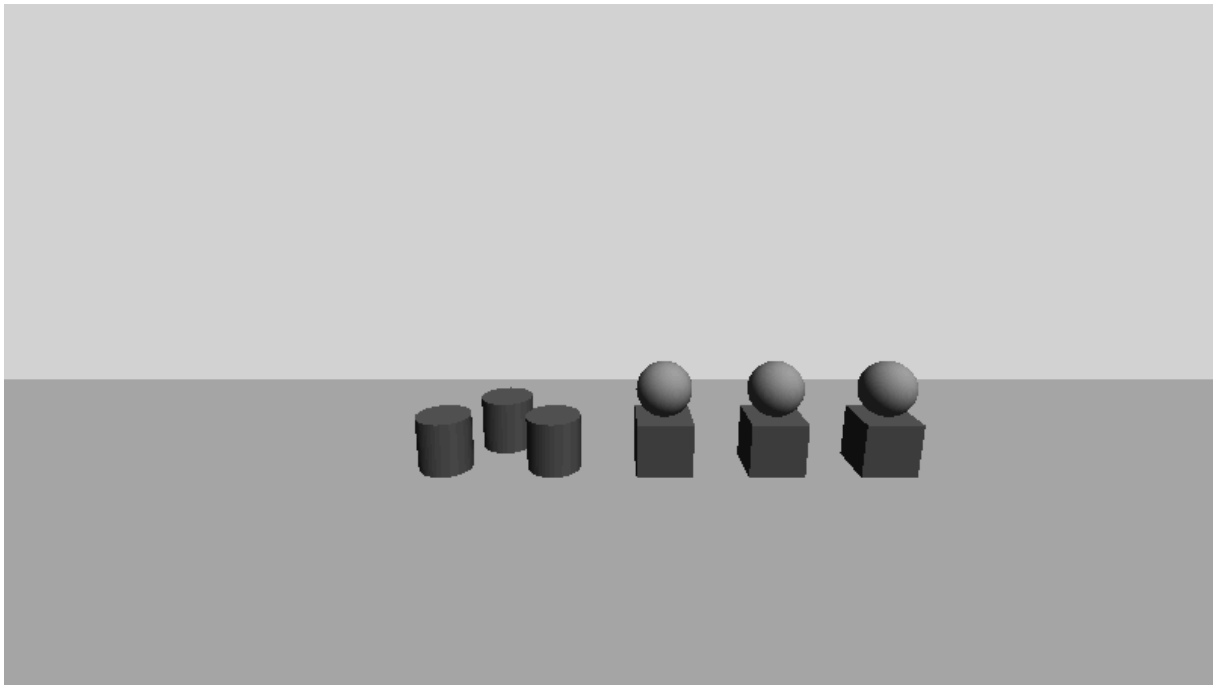


> Select the modified **Character2**

> Hit **Apply** on the **Nested Prefab Editor** (And **Not** on the classic Unity Apply button!)



As you can see all the instances of **Character2** have now a head.



Sharing the Head

Until now, all we have done can be done with normal Unity prefabs.

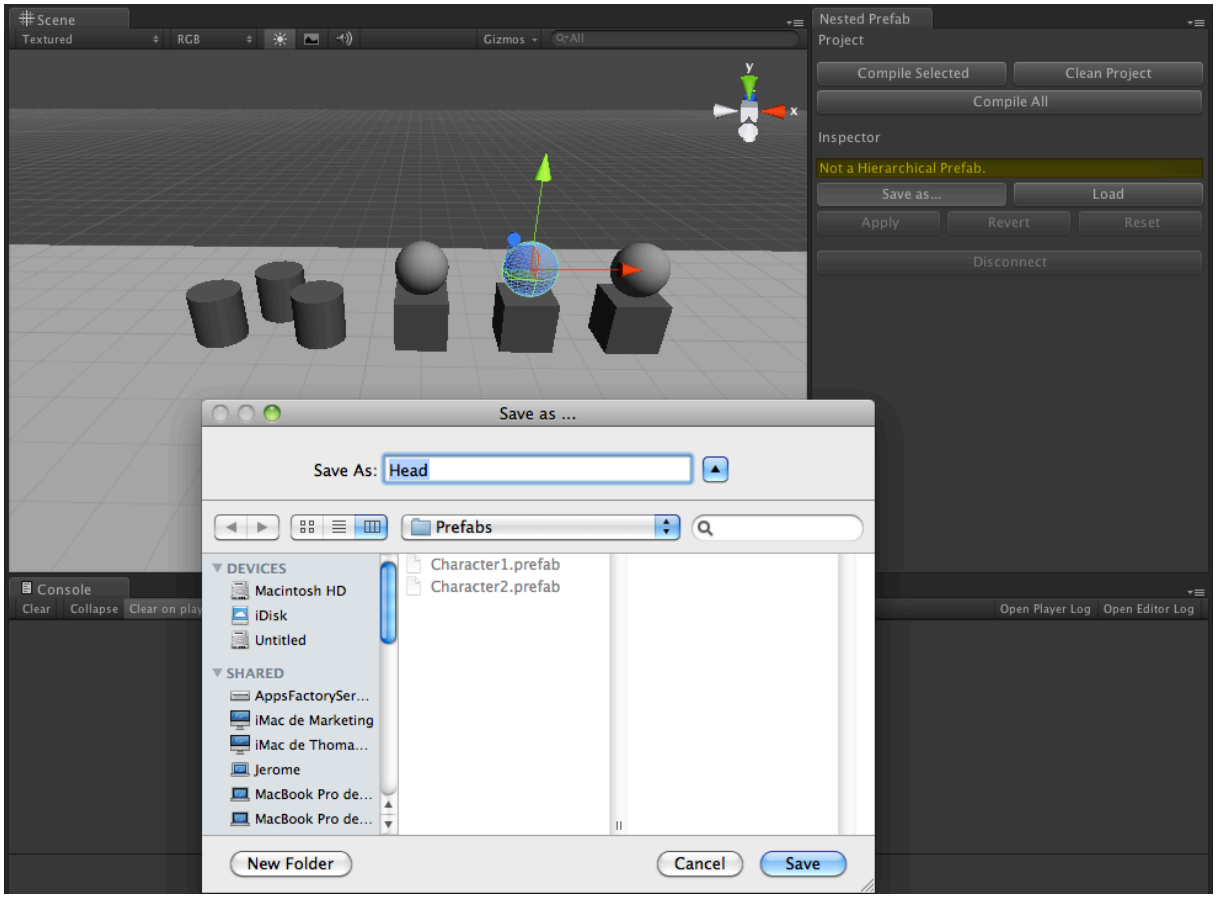
But what if we want to have a prefab for the head we have just create? A prefab that can be instantiated on **Character2** but also on **Character1.** So that we can change the head once and have the changes propagate to the **Character1 and Character2 instances!**

That isn't possible with the classic Unity Prefabs but it's what the Hierarchical Prefabs are for. They allow other prefabs to be nested into them.

Note that you can nest **hierarchical prefabs in hierarchical prefabs** but you can also nest **Classic Unity Prefabs in hierarchical prefabs**.

> **Save a Head** as a hierarchical prefab (You can also have made it a classic unity prefab)

> Select the **Character2** with the saved head and hit **Apply** on the **Nested Prefab Editor**

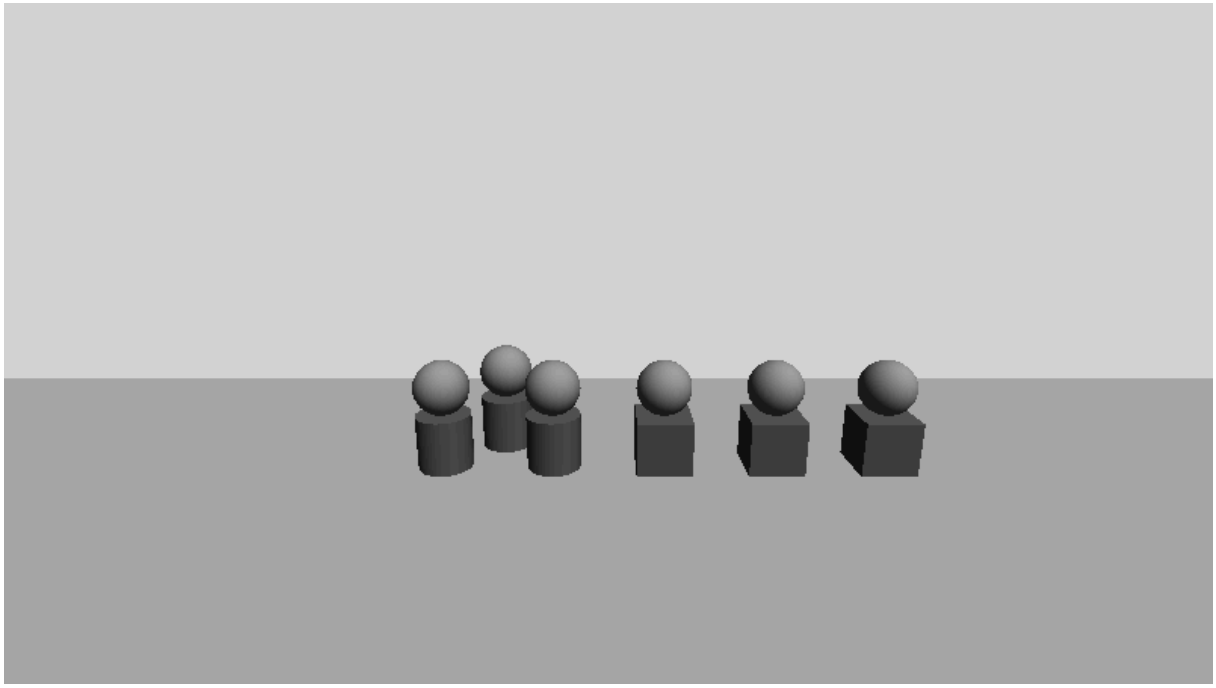


> Duplicate a Head of a **Character2**

> Put the duplicate Head on a **Character1**

> Select the **Character2** with the duplicated head and hit **Apply** on the **Nested Prefab Editor**

You must now have Heads on all your characters.



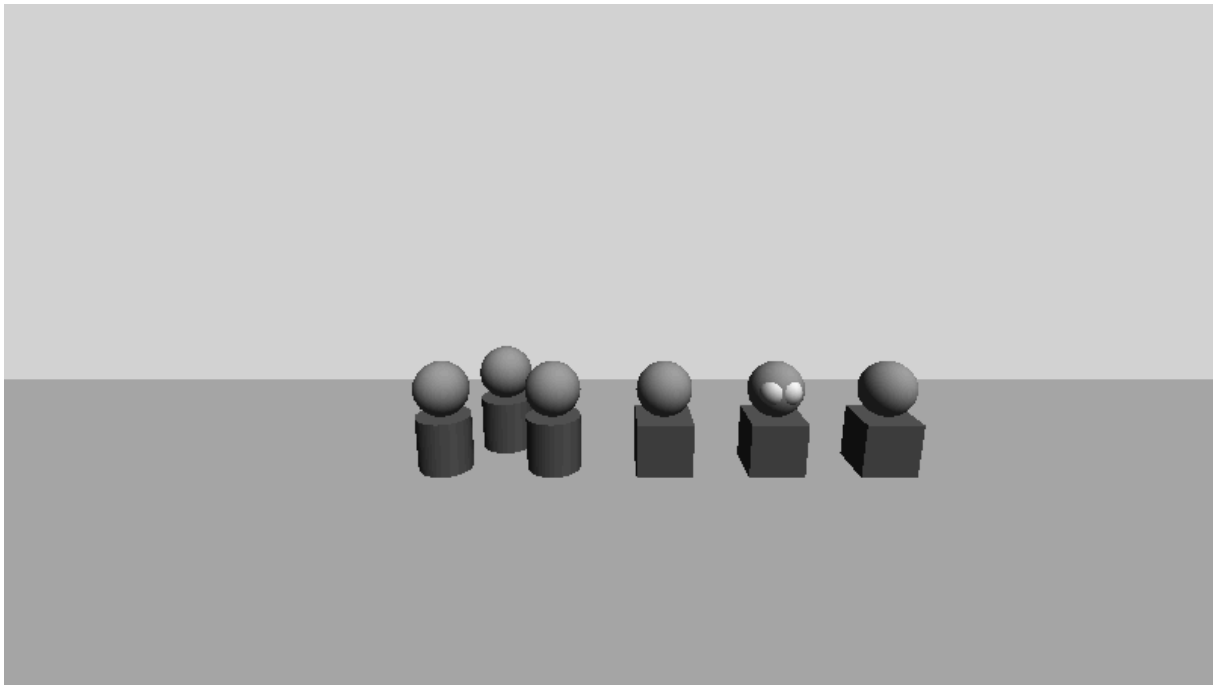
That doesn’t seem like it but the fact that all the heads are instances of the same prefab is a very powerful thing that makes level editing a lot easier.

Changing the Head

Let's add some eyes to the head.

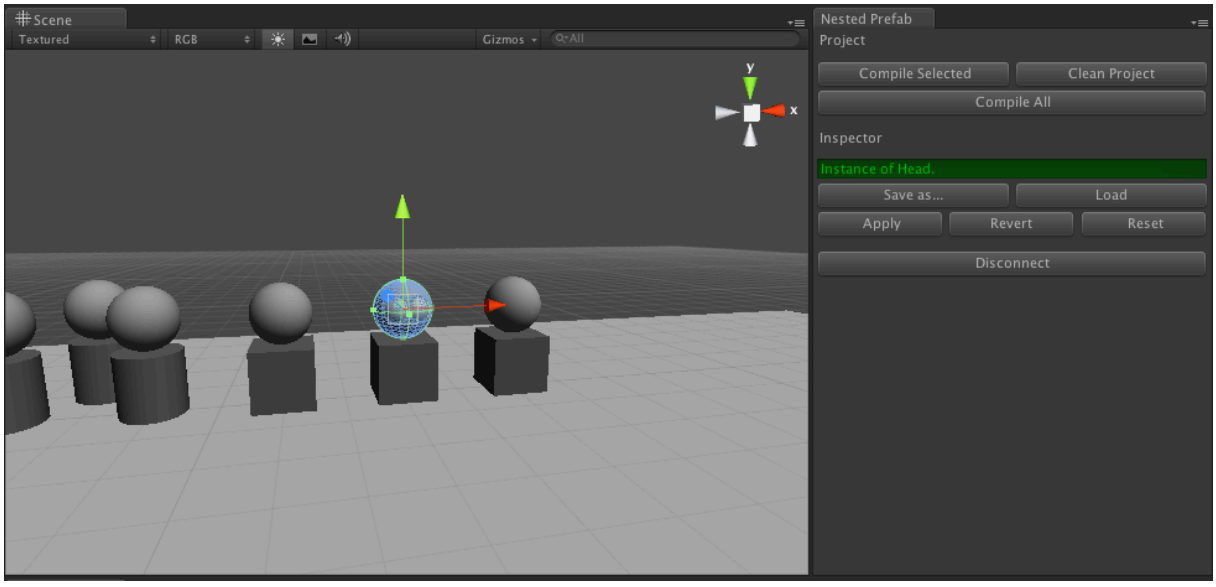
> Create 2 **Spheres.**

> Add them as child of a **Head** (it doesn't matter which one).

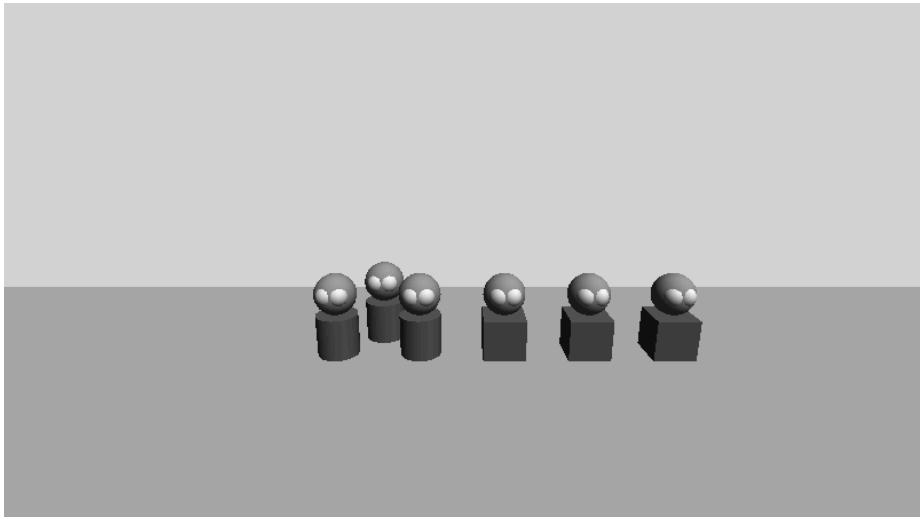


> Select the Head with the eyes

> Hit **Apply** on the **Nested Prefab Editor**



Now, as expected, all the heads have eyes!



Overriding the eyes color

Another powerful prefab feature is the ability the instance has to override his prefabs properties.

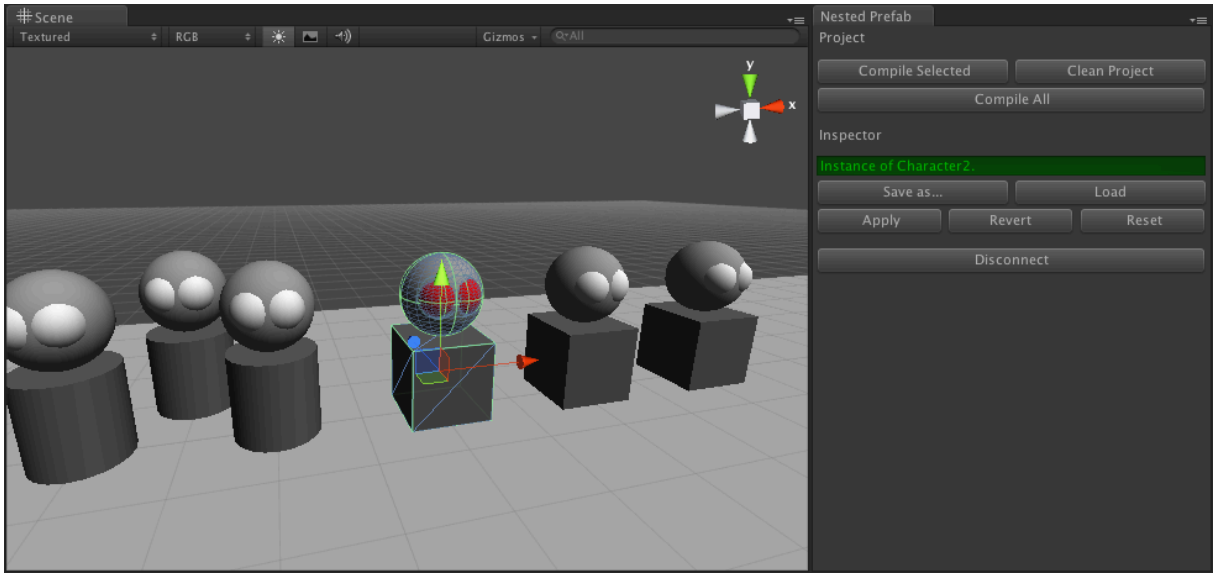
Hierarchical prefab instance can override their properties exactly like classic prefab. In addition they can also override the properties of their nested prefabs.

To demonstrate this, we will give red eyes to the **Character2**

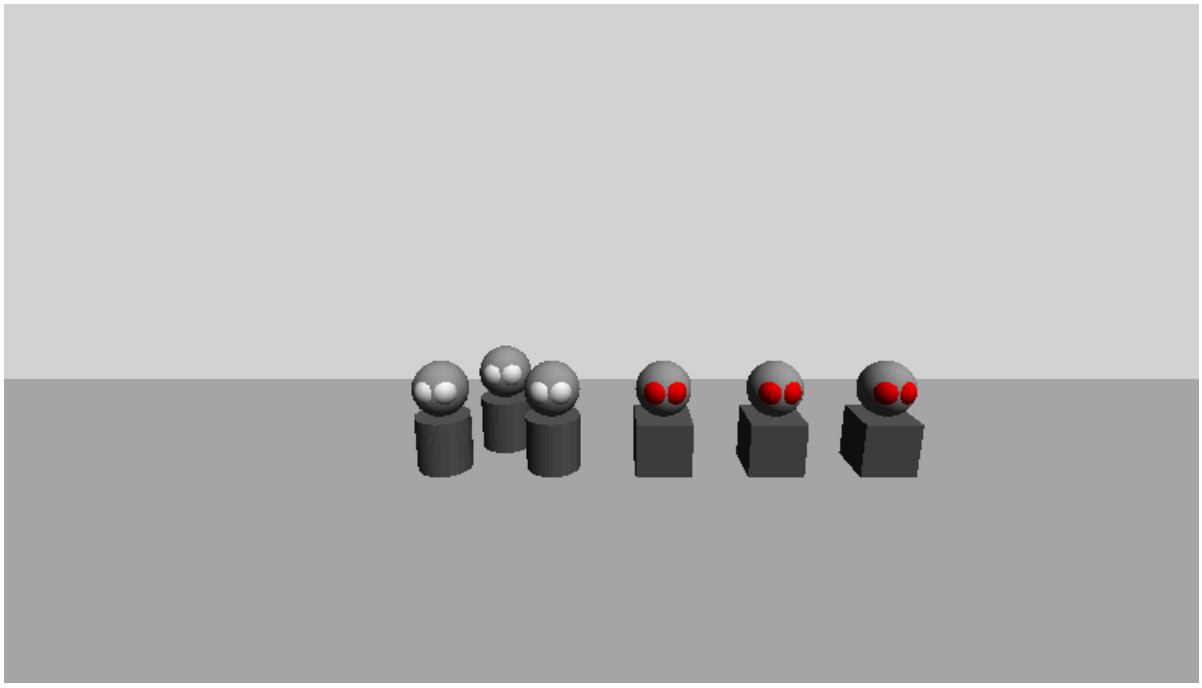
> **Apply another material** on the eyes of one of the **Character2** instance.

> Select the modified **Character2** instance. (**And not the head**, because we want to change the eyes colors of one Character type not the default eye color of all the heads)

> Hit **Apply** on the **Nested Prefab Editor**.



As you can see all the **Character2** instances have red eyes but not the **Character1** instances that have kept their default color.



Instantiate a Hierarchical Prefab at Runtime

Add a Rigid Body

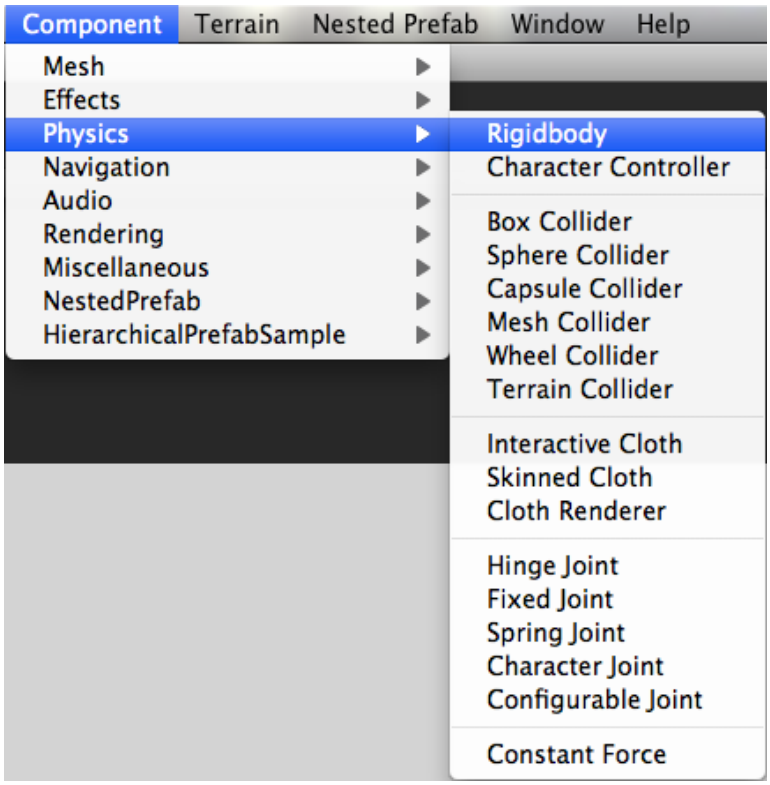
We will add a rigid body to all the characters. It's not a necessary step at all for hierarchical prefab instantiation but it will make the spawn more visual for our example.

> Select a **Character1** instance

> Click on **Component > Rigidbody**

> Hit **Apply** on the **Nested Prefab Editor**

> Do the same with **Character2**



Create the spawner

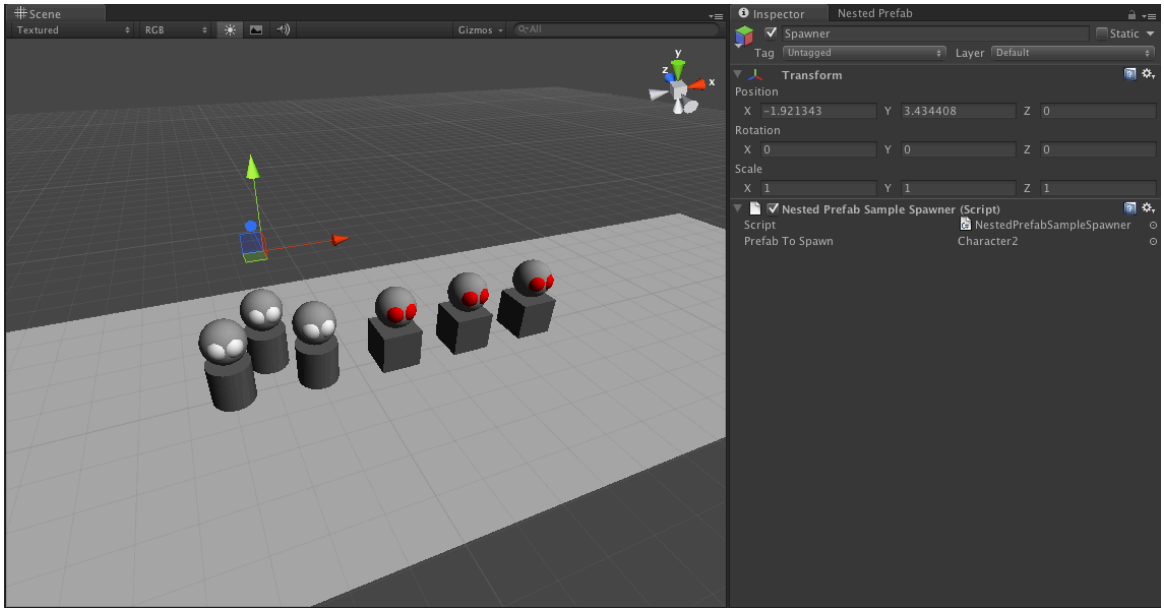
> Create an **Empty GameObject**

> Call it **Spawner**

> Place it over the ground in front of the camera

> Add to it the sample spawner component (**Component > NestedPrefabSample > NestedPrefabSampleSpawner)**

> Drag the **Character2** prefab from the **hierarchy panel** to the component **Prefab To Spawn** object field.



This sample script just calls the **HierarchicalPrefabUtility.Instantiate** method when the user clicks.

Note that you are compelled to use this method to instantiate Hierarchical Prefab via script. The classic **GameObject.Instantiate** will yield an error.

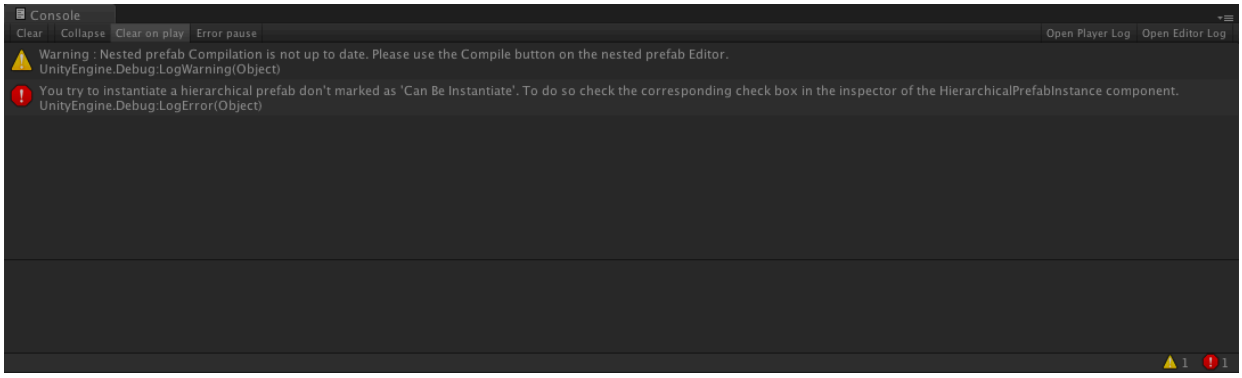
Nevertheless it's possible to call the **HierarchicalPrefabUtility.Instantiate** method to instantiate classic Unity Prefab Object as well. So you can replace all your Instantiate method by this one.

```
using UnityEngine;
using System.Collections;

[AddComponentMenu("HierarchicalPrefabSample/HierarchicalPrefabSampleSpawner")]
// Sample class used to spawn a hierarchical prefab with the special instantiate method
public class NestedPrefabSampleSpawner : MonoBehaviour
{
    // The prefab to spawn
    public GameObject prefabToSpawn;

    // Update is called once per frame
    private void Update()
    {
        // If the mouse is clicked
        if(Input.GetMouseButtonUp(0))
        {
            // Spawn the prefab at the spawner position
            HierarchicalPrefabUtility.Instantiate(prefabToSpawn, transform.position, transform.rotation);
        }
    }
}
```

> Click Play

> Click on the screen

The **Spawner** will fail to instantiate. Look at the console to see why.

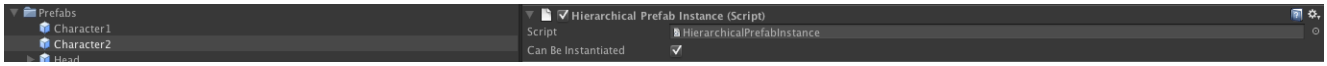


There are two things you have to take care of before you can instantiate a hierarchical prefab at runtime. First you need to tell which prefab can be instantiated, and then you need to compile these prefabs.

Mark the Character2 As “Can Be Instantiated”

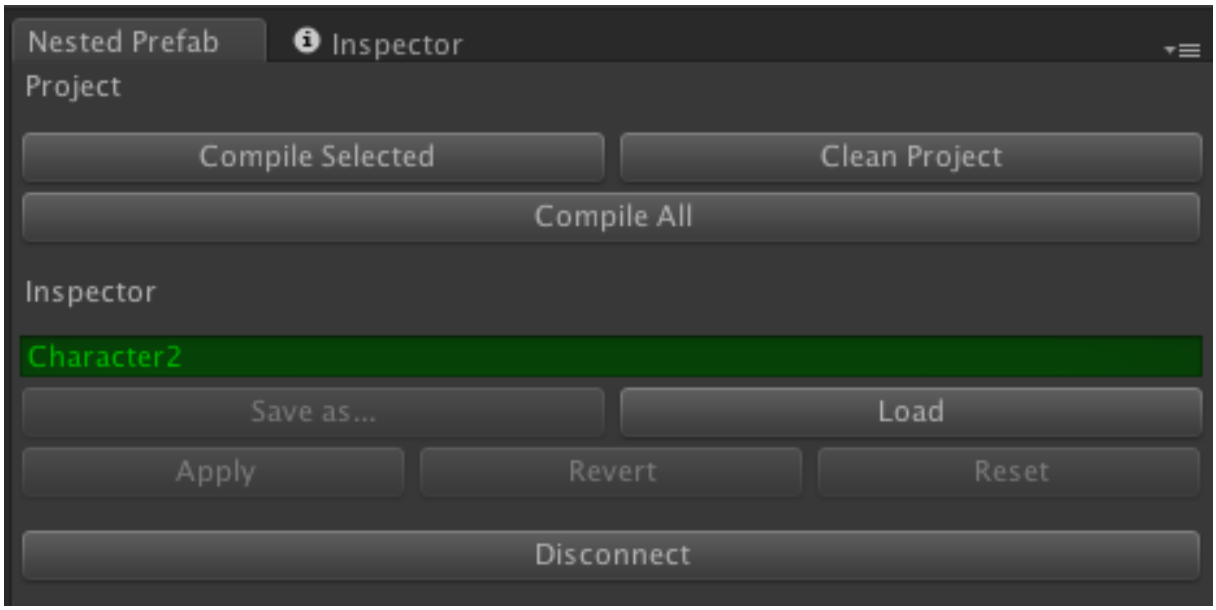
> Select the **Character2** prefab

> Check the **Can Be Instantiated** box**.**



Compile the Hierarchical Prefabs

> Hit the **Compile All** button in the **Nested Prefab Editor**



You can see all the compiled Hierarchical Prefab in the folder **NestedPrefab > Compiled**

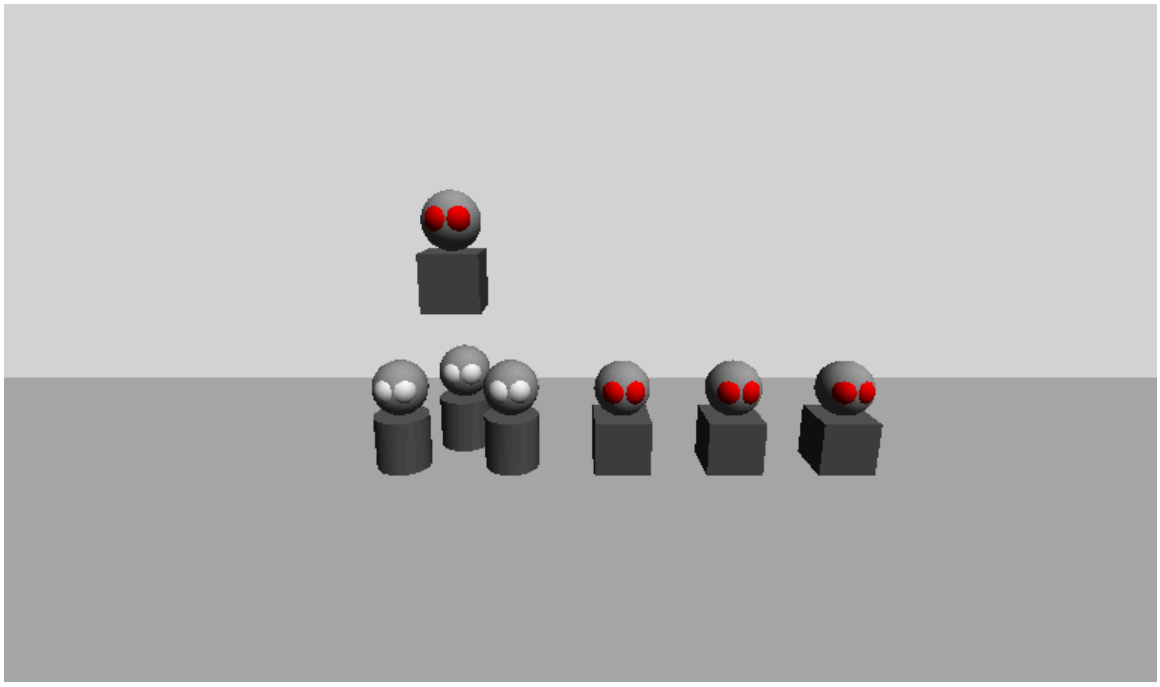
As you can see only **Character2** has been compiled. This is normal as only the prefab that can be instantiated in runtime need to be compiled.

Red eyes aliens invasion

You can now spawn as much **Character2** instances as you want in runtime.

> Click Play

> Click on the screen



> Click, Click, Click, Click, Click, Click, Click, Click, Click, Click, Click, Click, Click, Click, Click