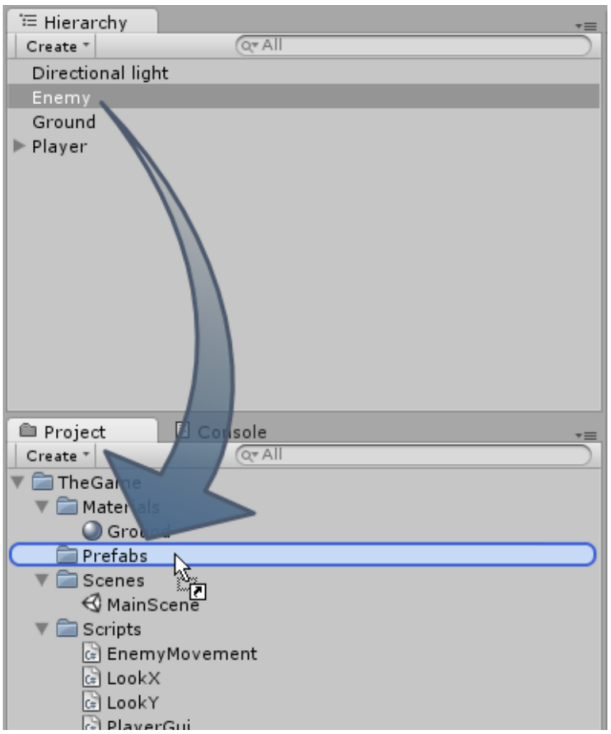
Unity Tutorial 4
CSC 3150/9010 - Game Development

---

1 Prefabs

Unity features an easy way to create templates out of any of your game objects. Unity calls this a "prefab".
Think of a prefab as a template, or a master copy of one of your game objects. Its that master copy that you keep on duplicating if you want more copies of it.

2 Making A Prefab Folder



Prefabs exist in the Project View. First let's make a folder to store all our prefabs. Inside the the game folder, make a new folder. Name it "Prefabs".

3 Identifying Prefabs

Now check the Enemy game object that you have on the scene. If you look at its entry in the Hierarchy View, its name is not in blue. This indicates that your Enemy game object is a prefab instance.

Think of a prefab instance as a "live" copy, whereas the prefab is the master copy that doesn't exist anywhere in the scene. A prefab's job is only to be duplicated into live copies to the scene. Prefabs are still the same game objects that you know. You can add or remove components from them. You can move them and edit their values like usual.

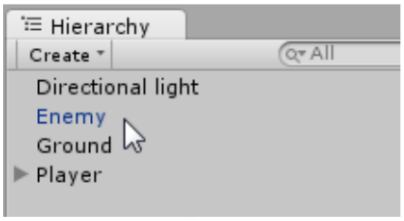


Illustration 1: Prefab instances are indicated by their blue name in the Hierarchy.

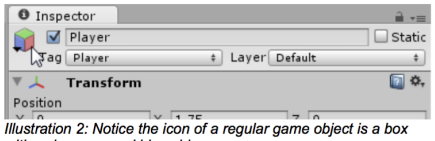


Illustration 2: Notice the icon of a regular game object is a box with red, green, and blue sides.

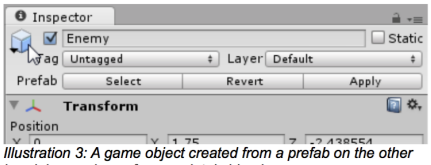
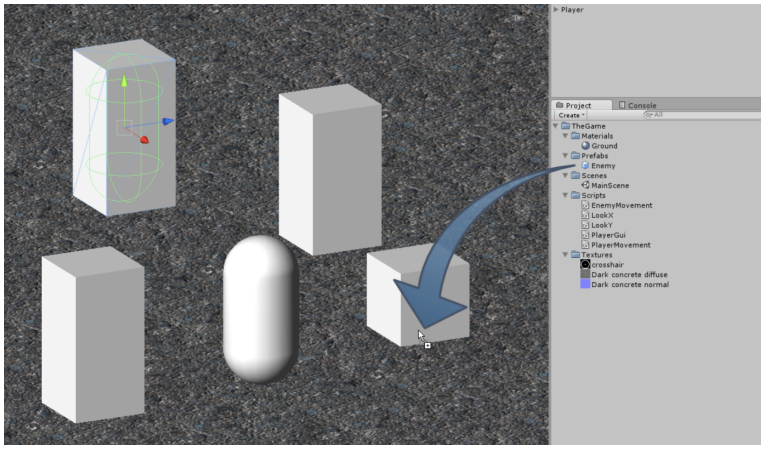


Illustration 3: A game object created from a prefab on the other hand, has an icon of a completely blue box.

4 Creating Prefab Instances

To create new instances of a prefab, drag the prefab from the Project View into anywhere in the scene. Alternatively, you can drag the prefab from the Project View into the Hierarchy View. Go



ahead and drop at least two more prefabs onto the scene.

5 Modifying Prefabs

Now, if we wanted duplicates of our Enemy, actually, we could have just copy-pasted it without bothering with prefabs. Why are we bothering with prefabs then? The convenience of a prefab is that you can change its master copy, and all live copies will have the changes propagated to them. (The most important reason you'll want prefabs is that you can create live copies of a prefab on-the-

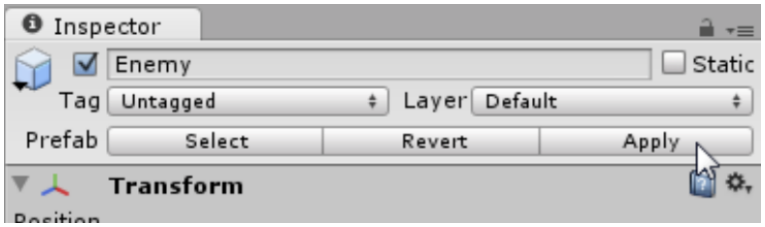
fly during runtime, but we'll tackle that later.)
Now, we'll try modifying the prefab. Click on the Enemy prefab in your Project View. The Inspector will show the Enemy prefab's properties, just like a regular game object.
Change the material color to green. If you select any of your Enemy game objects in the scene, you'll see it now has a green material also!

6 Overriding Prefab Values

How about if you want only one of the enemies to have a different color? You can still edit each Enemy game object individually. When you modify only one of them, the value will show in bold letters. That indicates that you've overriden its default prefab value.
If you ever want to revert a property back to its default value, right-click on it and choose "Revert Value to Prefab".

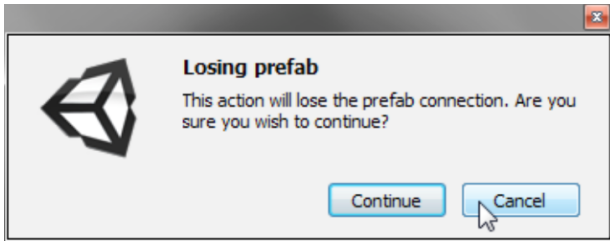
7 Turning An Override Value Into Its Default Value

If you ever decide that the override value you gave to one of your prefab instances should be applied to all copies of that prefab, you can do so by pressing the "Apply" button found at the top of the Inspector.



8 Changing Components In Prefabs

If you add a new component to one of your prefab instances, you'll be warned that it will break the "link" to your prefab.



Select one of your enemies in the scene. Add an Audio Source component (Component > Audio > Audio Source) to your Enemy.
Go ahead and click "Add" on the warning message that appears.
What happened is, since you added an Audio Source, your edited prefab instance will then no longer be "linked" to the original prefab. You just need to "link" it back. If you "link" it back, the prefab

(master copy) will now also have an Audio Source.
Simply press the “Apply” button in the top part of the Inspector to do so:
When that’s done, all copies of that prefab (and the prefab itself) will now have an Audio Source component.

9 Instantiate

We can create the prefabs on the fly using the Instantiate command. Create a public variable in your script with the type transform. This will allow any object to be assigned to this variable that has a transform component. Using the variable, Instantiate the object at a given Vector3 position and rotation (typically Quaternion.idenity).

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Build : MonoBehaviour {
5
6     public Transform brick;
7     // Use this for initialization
8     void Start () {
9         Instantiate(brick,new Vector3(0,0.5f,0),Quaternion.identity);
10    }
11
12    // Update is called once per frame
13    void Update () {
14
15    }
16 }
17
```