Unit 04b: GetComponent

Introduction

Some parts of working with Unity need their own unit. `GetComponent` is one of those parts. It's a critical piece of code to understand.

Goal

This unit is solely to make sure you understand the `GetComponent` method.

Process

As we've seen, objects in Unity consist of a **GameObject**. Attached to the gameObject are **Components**. Components are all pieces of *functionality*, defined by code. Sometimes, as with the standard Unity components like colliders or rigidbodies, that code isn't visible. But it's still code.

And the code we make are components, too. When we attach a script to a gameObject, it becomes a component.

Referencing components

Transform is the standard component attached to all GameObjects, and as such we can access it just by referencing `transform`, e.g. `transform.Translate()`.

But for all other components, we need to manually get a reference to before we can manipulate them in code. That's where `GetComponent` comes in -- it's the method we use to connect to components in the current gameObject and any other gameObject in the Scene. It looks like this:

```
gameObject.GetComponent<BoxCollider>();
```

> Remember that lowercase `gameObject` is the gameObject the script is currently attached to.
>
> If we wanted to get the component of a different gameObject, we call it from that gameObject:

> ```
> otherGameObject.GetComponent<HealthScript>();
> ```
>
> Easy as.

Let's break down the `GetComponent` method a little more.

You may not have seen the **carats** used yet -- the "<>" symbols. In C#, they are used to indicate a **Type**. So it's a way to specify the particular *type* of component you're getting. It comes *before* the parentheses because it's technially a **type parameter**, not a method parameter.

You can put whatever component inside those carets -- including your own scripts. Technically, you can also put the Transform in there, too -- if you wanted to be consistent you can use `GetComponent<Transform>()` instead of just `transform`.

Saving the Reference

Sometimes you'll want to save the returned component, so you can perform multiple things with it. You can do this easily by defining a variable of the same type as the component you're getting:

```
HealthScript enemyHealthScript = enemy.GetComponent<HealthScript>();

enemyHealthScript.DealDamage(14);
enemyHealthScript.HealDamage(2);
```

Sometimes you'll want to get a reference to a component to be used throughout the lifecycle of that object. In this case, you'll often want to make a class variable, and set its value during the `Start` or `Awake` methods:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerInput : MonoBehaviour
{
    public HealthScript healthScript;

    void Start()
    {
        healthScript = GetComponent<HealthScript>();
    }
}
```

Method Chaining

A cool aspect of C# is the ability to **chain methods** -- the ability to add methods together into one call, one after the other. You'll sometimes do this with `GetTransform`, where you want to just make one method call and not save the reference. In this case, you just add a dot and the method onto the end of the `GetComponent` call:

```
enemy.GetComponent<HealthScript>().DealDamage(16);
```

Wrap-Up

Go forth and get components!

Further Material

- GetComponent in Unity reference
- C# Types in Microsoft Reference