# GDD Homework 0

*Due Friday, September 22nd (9/22/17) at Noon (12 PM)*

This is an introductory homework designed to teach you the very basics of working in Unity3D. If you find that the assignment is taking you more than 4 hours, **please reach out to the teaching staff for help.**

For this assignment, you'll be making some tweaks to a small game where a cube bounces higher and higher off of a platform. The player will earn points by hitting the cube with their avatar (a lovely little capsule), timing it to move the capsule at the right time by pressing the SpaceBar.

## Initial Steps

- Install Unity 3D
    - You can find Unity at https://unity3d.com/
    - Don't worry too much about the build options for this class; you just need Unity itself.
    - If you already have Unity, update it to the most recent version (2017.1.1f1 at the time of writing).
- Install a C# IDE
    - If you took EECS 214, you'll be familiar with Visual Studios. This is the preferred option, and runs on Windows (dual-boot for Mac/Linux machines). Get it at https://www.visualstudio.com
    - The other accepted alternative is Rider (from JetBrains). Find it at https://www.jetbrains.com/rider/.
        - **note: we will not offer support for any other software in this class**
        - to be perfectly explicit, that means that if you decide to use any other software than Visual Studios or Rider, **you are on your own**
    - Both of the IDEs are pretty sizeable, offer excellent support for C#/C++ development, and are standard for industry development in Unity
- **If You Installed Visual Studios:** *(if you installed Rider, skip this step)*
    - Open up the Visual Studio Installer
        - this is the place to come to for installing language packs and various add-ons for Visual Studios
    - Under "Visual Studio Community 2017", click the button that says "Modify"
        - if there is an "Update" button instead, update Visual Studios before continuning
    - Scroll down to the section marked "Mobile Devices and Games"
    - Check the box for the Unity Development list item
    - In the bottom right, click on the "Modify" button to apply the relevant changes
    - Visual Studios can be extended with ReSharper, which adds a *lot* of functionality to the programs. Find ReSharper at https://www.jetbrains.com/resharper/ and install it.
- **If you Installed Rider:** *(if you installed Visual Studios, skip this step)*
    - Open Rider
    - Click the "Configure" button in the bottom right

  - Click "Plugins"
  - Scroll down until you see "Unity Support"
  - Check the box next to "Unity Support"
  - Click "OK" in the bottom right
  - Rider will restart

## Working with Unity

Now that you have Unity installed alongside a relevant IDE, let's take a look at some the fundamentals of working in the Unity editor.

- Unzip the Homework0 folder and delte the old .zip; you don't need it anymore
- Click the "Open" button in the top right and point Unity towards the folder you just unzipped (it should contain an "Assets" folder and a "ProjectSettings" folder)
- Open the "main" scene by double-clicking it at the bottom of the screen

## Configuring Unity for First-Time Use

Unity has a lot of settings designed for making building games easy. We're really only concerned with one of them for right now, though: having Unity open up your scripts in the right editor.

- Under the "Edit" menu (top left), click "Preferences"
- Under the "External Tools" tab, select the drop-down labeled "External Script Editor"
- Click "Browse" to find the editor you installed back in the introduction (Visual Studios or Rider)

Unity will now open your scripts in the correct editor.

> **Aside: GameObjects**
>
> GameObjects are Unity's way of representing *stuff* that exists in your game. Everything in the scene is a GameObject, even if it isn't visible to the player. Similarly, every GameObject has a Component (we'll talk about Components later) called a Transform. A Transform tells Unity *where* in the world a given GameObject is. It holds information like the position, rotation, and scale of a GameObject.

### Working with GameObjects

When you first open the scene, you should see a scene graph on the left-hand side. This is the list of all of the GameObjects currently in your game. If you hit the Play button in the top center (Ctrl-P on Windows; Cmd-P on Mac), the game will start playing. At this point, the all of the stuff in the game world should just sit still. We'll fix that.

First, we need to activate the Player. In the scene graph, you'll notice a GameObject named "Player" somewhere on the list. However, the name is grayed out. This is Unity's way of indicating that a given GameObject is *Inactive*. Inactive GameObjects don't do anything and the game engine ignores them for the most part when its running its calculations.

- Click on the "Player" GameObject
- In the Inspector tab (on the right-hand side), check the box next to the name "Player"
  - A checked box here indicates that a GameObject is active; an empty box, the opposite

A capsule (pill-like thing) should appear in the scene. When you hit the Play button, the capsule will now be drawn alongside the cube.

Let's get to actually moving the Player around on the screen. Open up the "Controller" script by double-clicking on it in the "Assets" section at the bottom of the Unity Editor screen.

There are a number of things going on here, but there are two that are of interest to us. First, the `Start` and `Update` magic functions. Unity has a handful of magic functions with specialized purposes. We aren't going to talk about all of them here, but just know that:

1. Spelling and capitalization matter. `OnCollisionEnter` is correct; `Oncollisionenter` is not.
2. Unity calls these functions whether you want it to or not.

The second thing that interests us is a function called `ShouldStartMoving`. Every frame, in the `Update` function, Unity calls this function. If it returns `true`, then the Player capsule starts moving if it isn't already in motion. If it returns `false`, nothing happens.

- In the body of `ShouldStartMoving`, change it so that the function returns `true` if the Space key has been pressed in the last frame.
  - We're not going to tell you the name of the relevant function-you have to find it yourself. A big part of this class is finding the right part of the Unity documentation to help you do what you want to do.

> **Aside: `Start` and `Update`**
>
> `Start` is a function that gets called at the beginning of a game. It gets called for every script that has a `Start` function. The order in which different `Start` functions is called is not easy to specify, and you shouldn't rely on any one `Start` getting called before any other.
>
> `Update` is a function that gets called once per frame. It's used to move things around, to check for input, and other important real-time actions. As with `Start`, it's not easy to specify which `Update` gets called first, so don't rely on a consistent ordering.

## Final Steps

Congratulations, you have yourself a game. It might not seem like much, but all good things have humble beginnings. There are a few easy tweaks, though. For one thing, the cube is pretty small, and hitting it with the Player capsule is kind of hard. Let's fix that by making the cube a little bigger.

- Open the Transform Component of the Cube in the Inspector tab by clicking on the Cube in the Scene graph
- In the section labeled "Scale," change the values from 1, 1, 1, to 1.5, 1.5, 1.5
    - This scales the GameObject by 1.5 on all axes, making it a good deal larger
- **Remember to save your scene!**

The cube should be far less frustrating to hit, and your work here is done.

## Submission

There are only a few things that we need to get your game working. Make a .zip file of **just** the "Assets" and "ProjectSettings" folders, and label it in the format: "NETID-Homework0"

For instance, my .zip file would contain a folder named "Assets" and a folder named "ProjectSettings," and the .zip file would be named "ecr867-Homework0".

Upload it to Canvas and you're all set!