

Link to Lab Resources

[google drive](#) OR <https://tinyurl.com/ta-comp376-daniel>

Introduction to Unity

Action Items

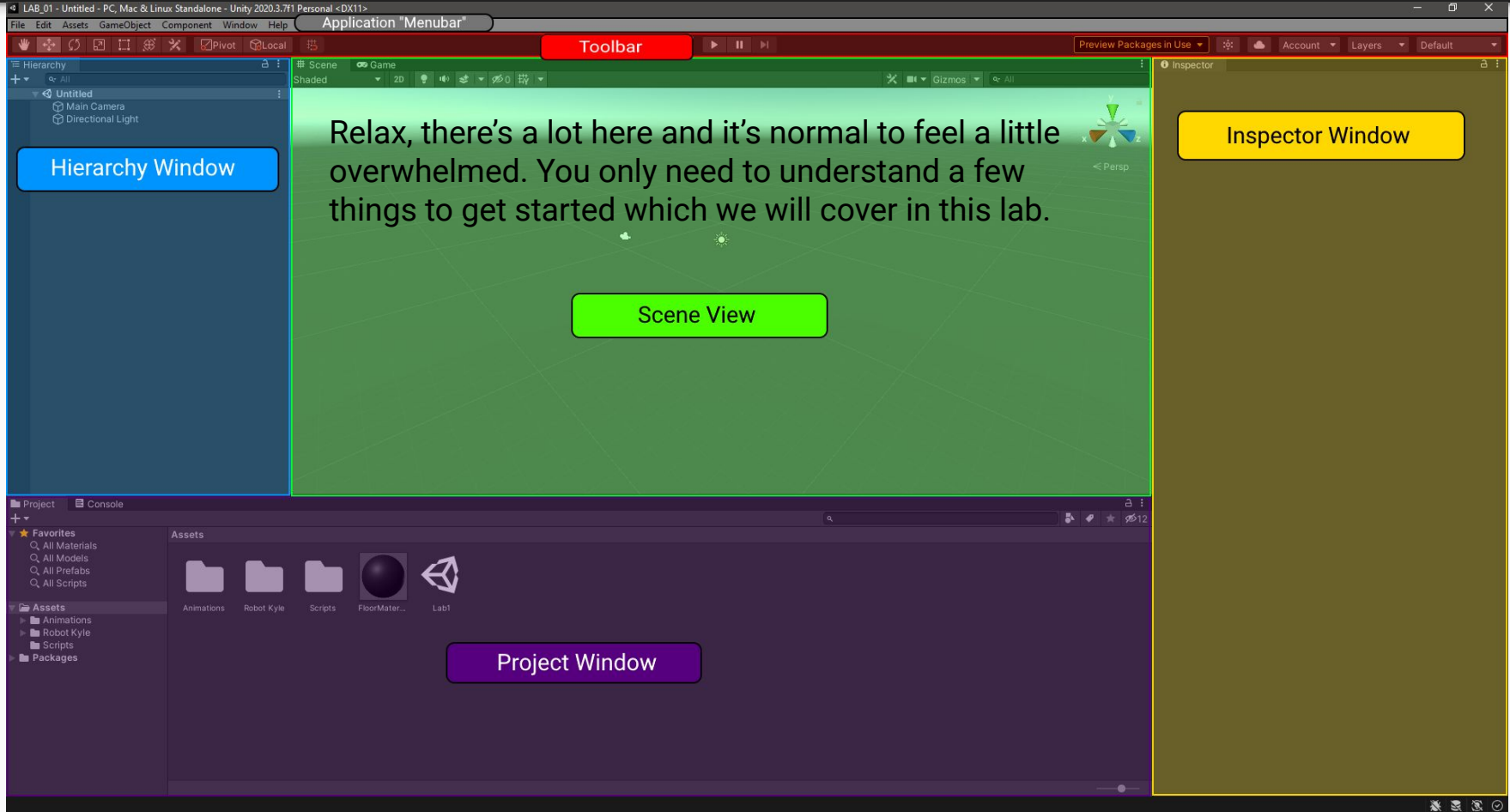
We will discuss :

- Explore the Unity Editor Interface using a new empty 3D project.
- Navigating in 3D space in the Scene view.
- Scripts, MonoBehaviour, Transform and GameObject.

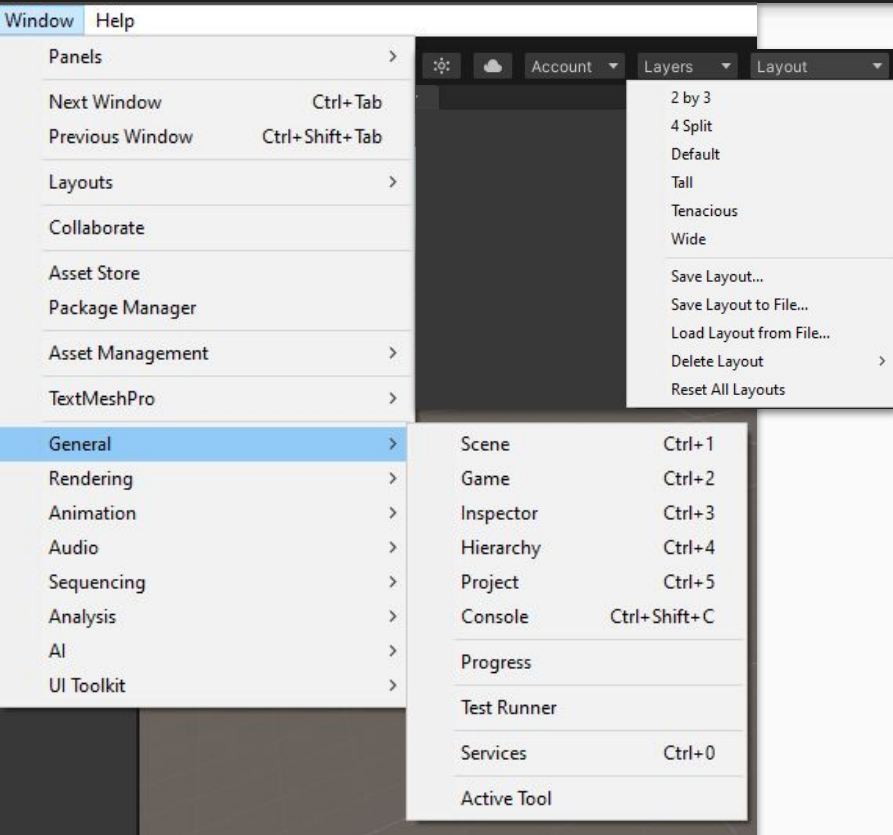
Tasks :

- Create and manipulate GameObjects. We will make a 3D avatar and a floor.
- Create a script that will allow us to move our avatar in 3D.
- Create a 2D version of our scene with similar functionality (just in 2D).
- Make a build and run it.

Unity Editor Interface



Unity Editor - Layouts



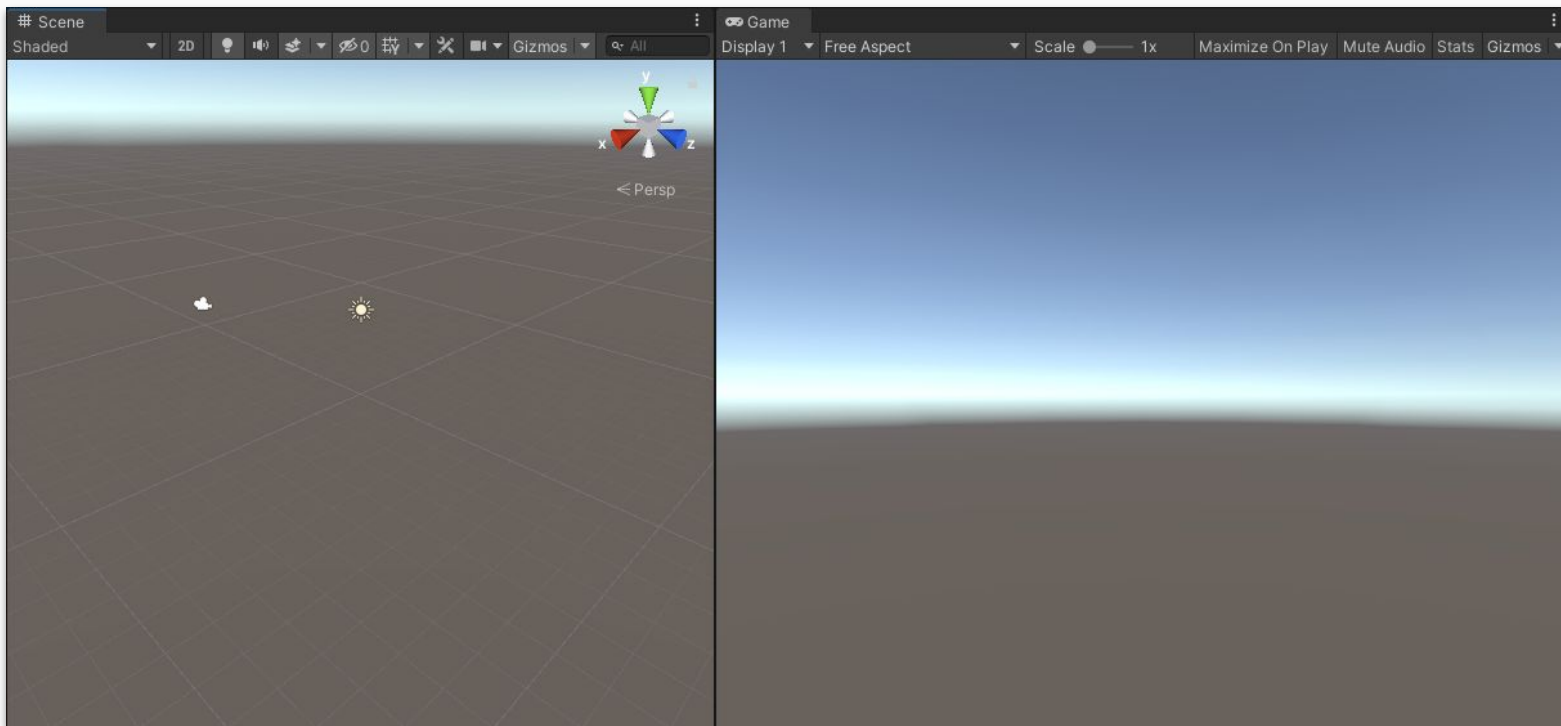
Here, when we talk about layouts within the context of Editor windows, we are talking about an arrangement/positioning of the windows in the Editor. Editor windows are separate sections of the Editor (see image above) that you can drag and drop and even snap to different positions/orientations by clicking on their appropriate tabs and dragging the mouse around.

- You can add more windows aside from the default by going to “Windows” in the **Application Menubar**.
- You can also use the Layout drop down menu in the **Toolbar** to change the arrangement of the windows. There are several preset layouts that come with Unity, and you can also save your own custom layouts.
- The best layout for the Editor depends on what you are doing and your own personal preferences.

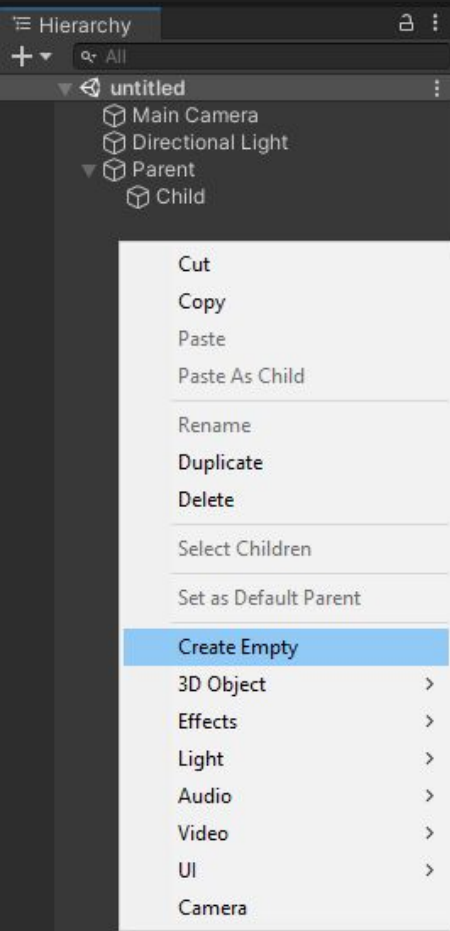
Unity Editor - Scene & Game View

In the center of the default Unity Editor layout is the **Scene view**. This is your interactive window into the world you are creating. You'll use the Scene view to manipulate objects and view them from various angles.

In the default window layout, the Game view also appears in this area as another tab; you'll use the Game view to playtest your game within the editor.



Unity Editor - Hierarchy Window



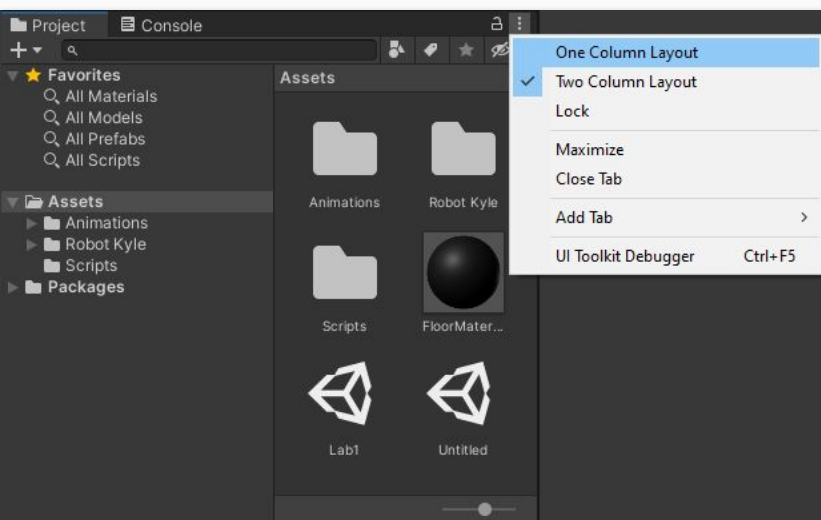
The **Hierarchy Window** is where you can organize all the things in your game world (and the world itself). These things are called **GameObjects** and this list is called the **Scene Hierarchy**.

- If you add a GameObject to a scene, it will be listed in that scene's Hierarchy. If you delete a GameObject from a scene, it will no longer be listed.
- You can add an empty gameobject to the scene by right clicking in the Hierarchy Window and selecting "Create Empty" from the **GameObject Context Menu**. There are also presets available (3D Object, Effects, UI, Camera, etc.). These are essentially the same as an empty gameobject but with a bunch of preset **components** attached to it.
- As the name suggests, the scene hierarchy is well, a hierarchy. This means that gameobjects start at the root (Ex: Main Camera, Directional Light, Parent) and they have no parent (null). Gameobjects that are nested within other gameobjects share a **child-parent relationship**. This will be important when we talk about **Transform** components later on.
- In the example on the right, there is only one scene loaded (untitled), but Unity also allows you to load multiple scenes at once if you wish.
- There is always only one currently "active" scene at any given time.

Unity Editor - Project Window

The **Project Window** is where you can find all the game files (assets) available for use in your project, whether you use them or not. The Project Window works like a file explorer, organized in folders.

- Note the difference between the Project and Hierarchy windows: the Hierarchy contains all the GameObjects in the current active scene, and the Project Window contains all the assets available to your entire project.
- You can drag assets (scripts, images, scene files, etc.) directly into the Project Window from your system's file explorer and Unity will detect and add them along with another file (one per file added) with the same name as the added file but with a **".meta"** extension. These **.meta** files are hidden in the Project Window and **should not be ignored** by **version control** software.



- If you do not like the default 2-column layout for the Project Window, you can click the 3 little dots in the top right corner of said window and select "One Column Layout" from the drop down.
- The root folder in the Project Window is always the project's **"Assets"** directory which is located at **PROJECT_ROOT/Assets/**

Unity Editor - Assets Directory & Special Folder Names

You can usually choose any name you like for the folders you create to organise your Unity project. However, there are [folder names](#) that Unity reserves for special purposes which you should know about in case you use them.

- **“Assets”** folder : The Assets folder is the **main folder** that contains all the Assets used by a Unity project.
- **“Editor”** folder : Any file that is a descendant of a folder called **“Editor”** are considered (Editor-Only). These files (usually scripts) add functionality to Unity during **development**, but **aren’t available in builds at runtime**. Note: Unity doesn’t allow components to be assigned to GameObjects if the scripts are a descendant of an Editor folder (more on components in the next slide).
- **“Resources”** folder : You can load Assets on-demand from a script instead of creating instances of Assets in a Scene for use in gameplay. You do this by placing the Assets in a folder called **“Resources”**. Load these Assets by using the [Resources.Load](#) function. **Important** : Unity **advises against using this folder** if possible for various [reasons](#), there is always another (most likely better) way to load assets on-demand.
- **“StreamingAssets”** folder : You may want an asset to be available as a separate file in its original format (although it’s more common to directly incorporate assets into a build). For example, you need to access a video file from the filesystem to play the video on IOS using the [Handheld.PlayFullScreenMovie](#) function. You can only have one **“StreamingAssets”** folder and it must be placed in the root of the **“Assets”** folder. Any file that is part of this folder remains unchanged when copied to the target machine, where it’s available from a specific folder.
- **Ignored files & folders** : During the import process, Unity **ignores** the following in the Assets folder : **hidden folders, files & folders which start with “.” or end with “~”, files & folders named “cvs”, files that end with “.tmp”**
- Other folder names to watch out for include: “Editor Default Resources”, “Standard Assets”, “Gizmos” ([more info here](#))

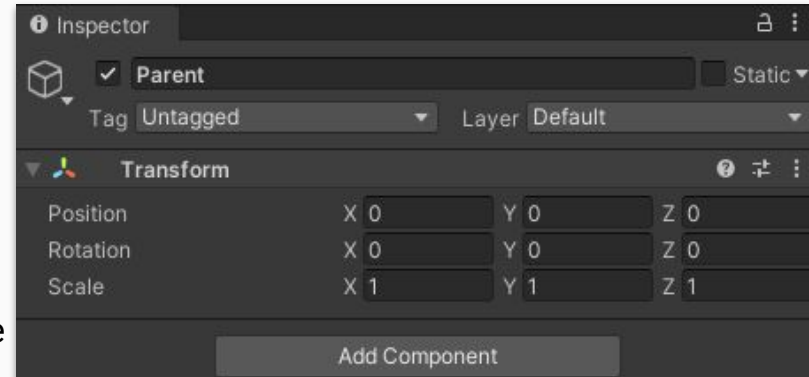
Unity Editor - Inspector Window

The **Inspector Window** is where you'll find and configure detailed information about GameObjects.

When you select a gameobject in Scene view or in the Scene Hierarchy, you'll see its components listed in the Inspector.

Components describe the properties and behaviors of gameobjects and are the level designer's main source of interaction/configuration with the behaviours that you associate with your gameobjects. In short, components are scripts (C# classes) that can be "attached" to gameobjects to give them custom desired behaviour.

- Any custom behaviours (scripts) that you create must derive from the [Component](#) class if you want to attach them to a gameobject. Although, when you create custom behaviours you will instead derive from the [MonoBehaviour](#) class as the Component class is not meant to be directly derived from by the programmer. The MonoBehaviour class inherits from Component.
- By default, every gameobject you create will have a non-null [Transform](#) component attached to it. A gameobject will always have exactly one Transform component attached to it at all times which you cannot detach.
- The Transform component exposes transformation information about the game object in the world (position, rotation, scale) and allows the level designer to modify these properties directly in the Inspector Window.
- You can **add a component** to a gameobject by **using the button** → or by **drag & drop from Project Window to the Inspector Window**




You can click the "Add Component" button to add built-in or custom components to a gameobject.

Unity Editor - Toolbar



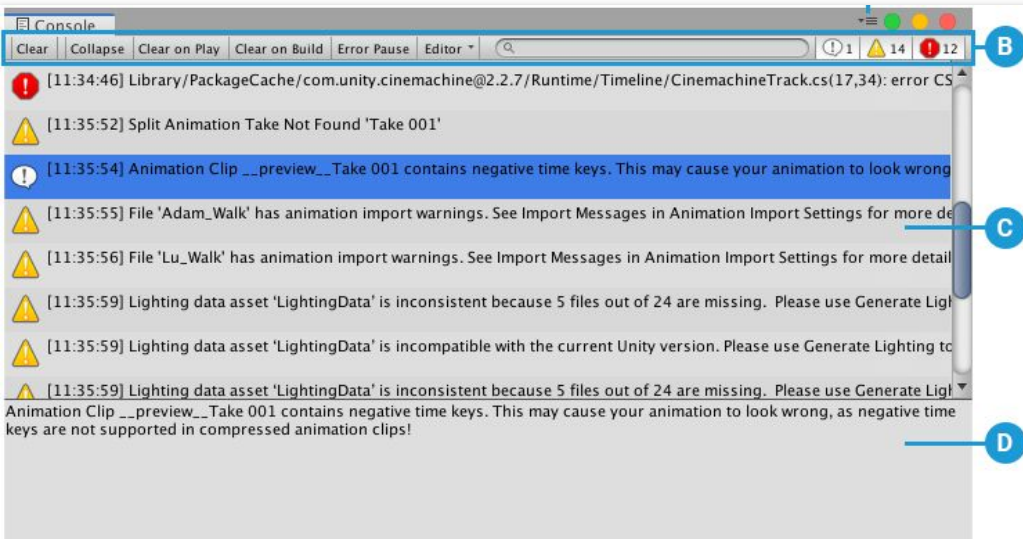
The **Toolbar** is always at the top of the Unity Editor Interface. Use the toolbar buttons to select and adjust GameObjects, change your point of view in the Scene, and start and stop **Play Mode**.

- The application is considered to be in Play Mode within the Unity Editor if the game is running and playing. To do so you can press the triangular shaped play button (▶) in the center of the toolbar. Press the button again when in Play Mode to stop running the game and return to **Edit Mode**.
- There are transform tools (located on the left side of the Toolbar) which allow you to change how you interact with the Transforms of one or more gameobjects that you have selected in the Hierarchy Window. You can quickly switch between these tools using QWERTY keys. 
- The first transform tool, the Hand Tool, allows you to pan around the Scene. Effectively modifying the transform of the scene view camera (this is not to be confused with the in-game camera).
- The others : Move, Rotate, Scale, Rect Transform and Transform tools allow you to modify the Transform of the currently selected gameobjects in the Hierarchy Window. You can guess which transformation properties the first 3 allow you to modify by their name. The Rect Transform Tool allows you to modify 2D Transforms (called **RectTransforms**) which we will discuss in another lab. Finally the last tool, the Transform Tool, is just a combination of the first three.
- More info on the other toolbar buttons can be found here : <https://docs.unity3d.com/Manual/Toolbar.html>

Unity Editor - Console

The **Console Window**, the must have tool in any editor for a programmer.

- The Console Window shows errors, warnings and other messages generated by Unity.
- You can also show your own messages in the Console using the [Debug](#) class.
- Everything that is written to the Console Window (by Unity, or your own code) is also written to a [Log File](#).
- To open the Console from the Unity application menubar, select Window > General > Console



B - The Console toolbar has options for controlling how messages are displayed, and for searching and filtering messages.

C - The Console list displays an entry for each logged message. Select a message to display its entire text in the detail area.

D - The detail area shows the full text of the selected message.

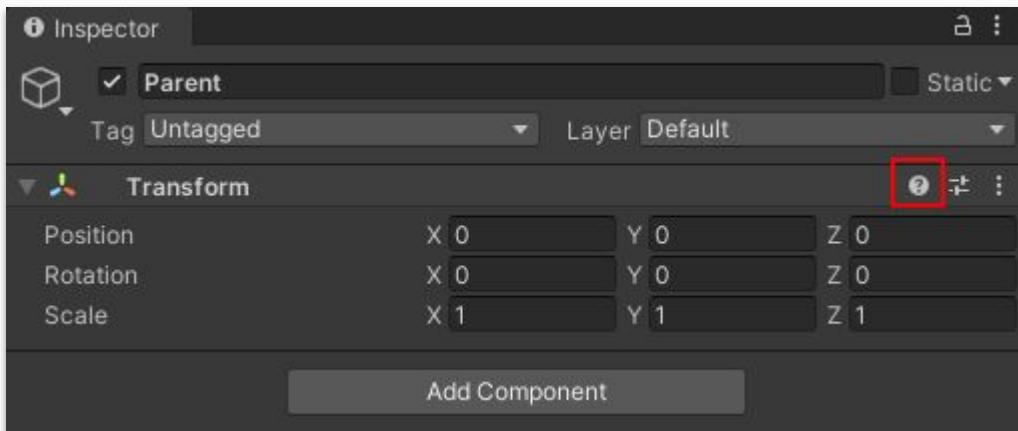
For more information about the Console Window visit the following link :

<https://docs.unity3d.com/Manual/Console.html>

Unity Manual

There is so much functionality packed into the Unity Editor that it's impossible to cover absolutely everything in these labs. Luckily Unity has a great community and tons of documentation!

- [Unity Manual](#)
- [Scripting API Documentation](#)
- [Unity Answers](#) : Unity's own kind of version of stack overflow with a very active community
- Protip : If you have any questions about a given component in Unity and want to quickly access the documentation, you can click the ? icon in the top right area of any component.



Action Items

- ~~Explore the Unity Editor Interface using a new empty 3D project.~~
- Navigating in 3D space in the Scene view.
- Scripts, MonoBehaviour, Transform and GameObject.
- Create and manipulate GameObjects. We will make a 3D avatar and a floor.
- Create a script that will allow us to move our avatar in 3D.
- Create a 2D version of our scene with similar functionality (just in 2D).
- Make a build and run it.

Navigating the Scene View

When working in Unity Editor, navigating in the Scene view is very important. One way to think about navigating in this window is like operating a drone camera — it lets you examine your GameObjects from any angle or distance. With practice, you can learn to navigate with ease. There are also more general settings you can use to configure the Scene view.

Let's quickly review the basics:

- **Pan:** Select the Hand tool in the Toolbar, and click and drag in the Scene view to move your view. You can also middle click and drag.
- **Zoom:** Holding Alt (Windows) or Option (macOS), right-click and drag in the Scene view to zoom. You can also just use the scroll wheel.
- **Rotate:** Right-click and drag in the Scene view to rotate the scene view camera (this option is not available in 2D mode).
- **Orbit:** Holding Alt (Windows) or Option (macOS), left-click and drag to orbit around the current pivot point (this option is not available in 2D mode).
- **Focus (Frame Select):** When a GameObject is selected, select F with your cursor in the Scene view to focus your view on that GameObject. Note: If your cursor is not in the Scene view, Frame Select will not work.

Navigating the Scene View

Flythrough mode

You can also use Flythrough mode to navigate in the Scene view by flying around in first person, which is common in many games.

To do this, click and hold the right mouse button then :

- Use **WASD** to move the view **left/right/forward/backward**.
- Use **Q** and **E** to move the view **up and down**.
- Press and **hold Shift** to **move faster**.

Note: Flythrough mode is **not available in 2D mode**. Instead, **holding the right mouse button** down and dragging **pans** around the Scene view.

For further guidance, you can review a complete overview in the [Unity Manual](#)

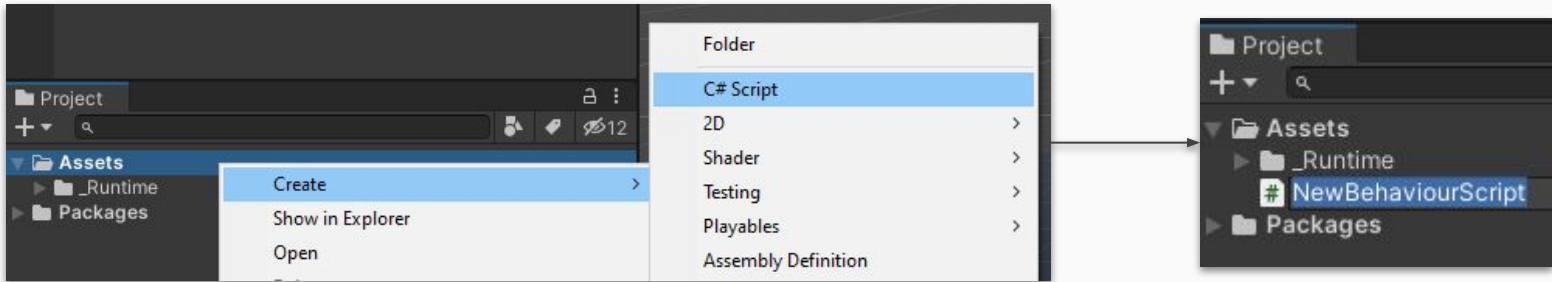
Action Items

- ~~Explore the Unity Editor Interface using a new empty 3D project.~~
- ~~Navigating in 3D space in the Scene view.~~
- Scripts, MonoBehaviour, Transform and GameObject.
- Create and manipulate GameObjects. We will make a 3D avatar and a floor.
- Create a script that will allow us to move our avatar in 3D.
- Create a 2D version of our scene with similar functionality (just in 2D).
- Make a build and run it.

Scripts - MonoBehaviour

The behavior of GameObjects is controlled by the components that are attached to them. Although Unity's built-in components can be very versatile, you will soon find you need to go beyond what they can provide to implement your own gameplay features. Unity allows you to create your own custom behaviors using C# scripts. These allow you to trigger game events, modify component properties over time and respond to user input in any way you like.

- To create a new script, **right click under the Assets folder** in the Project Window and select **Assets > Create > C# Script** from the **Create Asset Menu** that pops up. The new script will be created in whichever folder you have selected in the Project Window. The new script file's name will be selected, prompting you to enter a new name.



- It is a good idea to enter the name of the new script at this point rather than editing it later. The name that you enter will be used as the name of the C# class that will be generated for you. It is **important** that the **filename matches the class name**.

Scripts - MonoBehaviour

The default template for a script created by the Unity Editor looks like this ----->
Where “MyClassName” is the name of the file you created.

There are two functions created for you called “**Start**” and “**Update**”. These are known as “**magic methods**” (Unity’s terminology). Just like with the folder names in the Assets folder, there are special function names as well.

These magic methods are called at different times during a given **frame**. Your game automatically has a gameplay loop built into it and it’s being called an arbitrary amount of times per second. Each time the loop is called, this is referred to as a **frame**. Most games nowadays run at > 60 **FPS**.

So when you include a magic method in your script, Unity will detect this and call it at the appropriate time. This allows developers to call their own logic at different times during a given frame. The full list of magic methods, including their execution order and other various information is documented here :

<https://docs.unity3d.com/2020.3/Documentation/Manual/ExecutionOrder.html>

(BOOKMARK THIS LINK!)

You may also notice that our class inherits from [UnityEngine.MonoBehaviour](#) . You can think of this class as a kind of blueprint for creating a new component type that can be attached to GameObjects. These custom MonoBehaviours are what developers use to tie their custom behaviors to a gameobject. **Unity does not have a “main method”** like you may be used to in other languages. So you must use the magic methods to call your custom logic at the right time during the gameplay loop. You must attach your script to a gameobject for it to work (See previous slide on components in the Editor)

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class MyClassName : MonoBehaviour
5  {
6      // Use this for initialization
7      void Start()
8      {
9      }
10
11     // Update is called once per frame
12     void Update()
13     {
14     }
15
16 }
17
```

Scripts - Transform & GameObject

Recall : Every gameobject in a Scene has **exactly one Transform** component attached to it. It's used to store and manipulate the position, rotation and scale of the object. Since every gameobject can have a parent, this information has to be stored somewhere. In Unity, the parent-child hierarchy information is stored in the Transform component which every gameobject has a reference to. This is the hierarchy seen in the Hierarchy Window.

- You can access the [Transform](#) component of a gameobject from within a MonoBehaviour script by accessing it's **transform** property which it inherited from the Component class.
- Similarly, you can access the [GameObject](#) instance itself from within a MonoBehaviour script by accessing it's **gameObject** property which it also inherited from the Component class.
- You can search Unity's [Scripting API](#) for documentation about any class provided by Unity. Also recall that you can click the ? button at the top right of a component to access it's documentation in the [Unity Manual](#).
- If you need access, at runtime, to the instance of another component attached to a gameobject from within your custom scripts, you can use the methods listed under "Public Methods" that **start with "GetComponent"** for either the Transform or GameObject classes.

Action Items

- ~~Explore the Unity Editor Interface using a new empty 3D project.~~
- ~~Navigating in 3D space in the Scene view.~~
- ~~Scripts, MonoBehaviour, Transform and GameObject.~~
- Create and manipulate GameObjects. We will make a 3D avatar and a floor.
- Create a script that will allow us to move our avatar in 3D.
- Create a 2D version of our scene with similar functionality (just in 2D).
- Make a build and run it.

Thank You

Link to Lab Resources

[google drive](#) OR <https://tinyurl.com/ta-comp376-daniel>

Other links

<https://docs.unity3d.com/2020.3/Documentation/Manual/index.html>

<https://docs.unity3d.com/2020.3/Documentation/Manual/ExecutionOrder.html>