Introduction To Unity – Interface & Structure

Welcome to Unity!! ☺

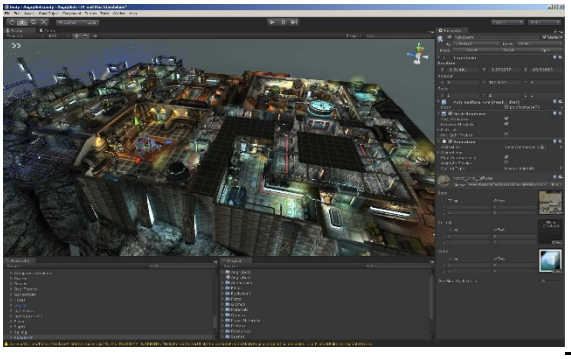
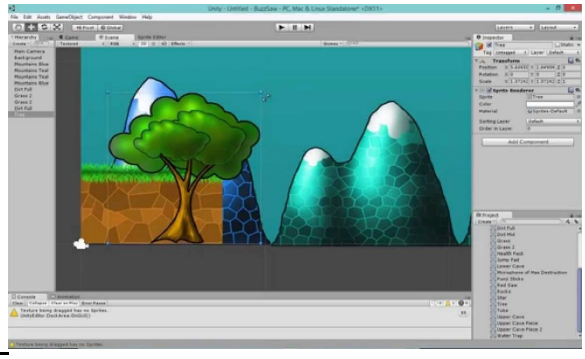
Defining Unity and its Capabilities:

1. It's a game development application defined by anything that has interactivity to it...like an RPG (Role playing games), first person shooter, educational games and VR/AR. It is not typically the application that you create assets in (although you can create some things), and by assets, I mean models, textures, animation (using rigs), scripts etc. It's the application that brings of that stuff together in most instances.

    a. To create 3D assets, we'll need to use applications like:
        i. 3ds Max, Maya, Zbrush etc...
    b. To create 2D assets (like textures), we'll need:
        i. Photoshop, Substance Painter, Quixel, etc

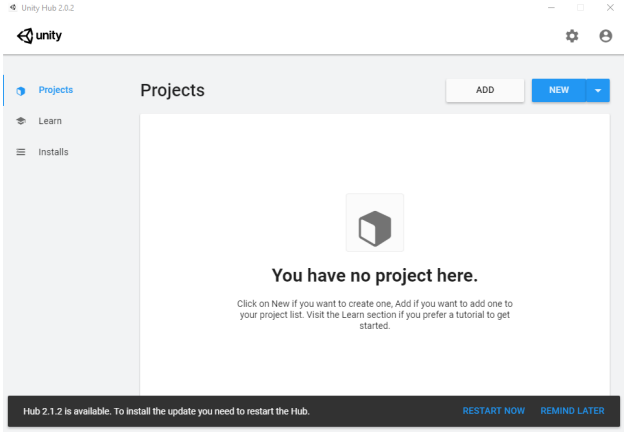
Once we have our assets, we will pull them into Unity scenes to create the interactive properties using C# scripts.

2. In addition to interactivity, we can also create lighting and particle systems in Unity.

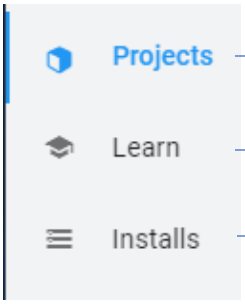
Unity Launcher:

1. If we open Unity for the first time - Starting with the Unity Hub (which is what pops up when you open Unity), we have a couple of options:



    a. To login using your personal Unity account and licensing (if needed) →

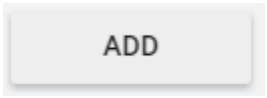
        i. Then sign in.
        ii. If you haven't created an account go to: https://id.unity.com/en/conversations/863f3b5a-5da0-422a-99ed-99a7bf03224e003f

b. 

With this selected, you'll see any current or past projects that you've worked on, available to select and open. To open a project simply double click on it.

This provides a series of learning projects, tutorials and links.

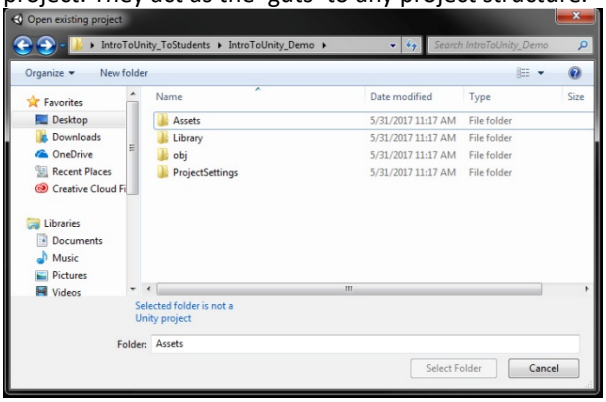
This shows the different versions of Unity you may have installed on your machine and allows you to select which you want to use. This becomes important once you start adding functionality to a scene. Sometimes things can break as you move from one version to the next, so you'll want to be aware of what you have going on in your project.

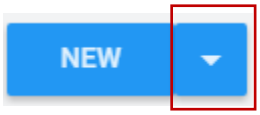
c. 

Allows us to add existing projects to our Unity Hub.

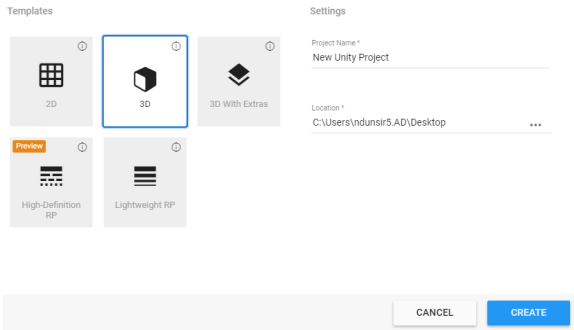
i. Once a project is brought in, you can select the Unity Version you want to use and the Target Platform you're outputting your application to.

ii. Unity will only recognize the folder that contains the '**Assets**' and '**Library**' within it, as these folders define the project. They act as the 'guts' to any project structure.



d. 

Allows us to create new projects



Note – we are going to use version 2019.03

i. **Project name**: name the folder you'd like your unity project to live in

ii. **Location**: click on the dots to navigate where you want your project to live

iii. Template - 3D vs. 2D

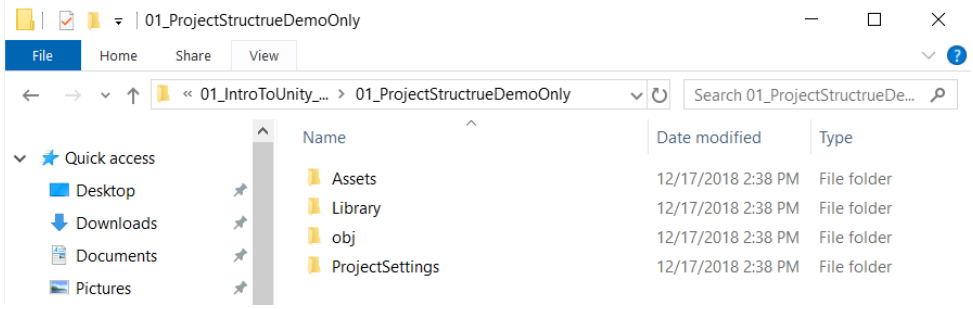
- Pick the style of game you want to create. It really doesn't matter which one you start with, as there aren't any limitations wither way. However, if you start in 2D, your interface layout will look a little different to begin with.

iv. Hit **Create**

e. If you already have Unity open and want to open a project:
   i. File → Open Project → Open → [navigate to your project structure as above]
      - When you open a new project, Unity will actually close and reopen. If you don't see anything in your viewport, simply double-click the scene that you want to open in the Project Tab
        - Assets → Scenes → 'file'.unity

Project Structure:

1. **Project** – is the folder structure that defines and holds all of the game assets as well as the connections made between those assets in Unity. It's highly important that we keep this clean and organized, most of which we'll do right through the Unity interface itself.



2. To examine the demo project that I have:
   a. My project structure is called: "01_ProjectStructureDemoOnly"
      i. When you create a new project, Unity will go in a create folders and files necessary to start your build. There are two main folders for this: "**Assets**" and "**Library**"

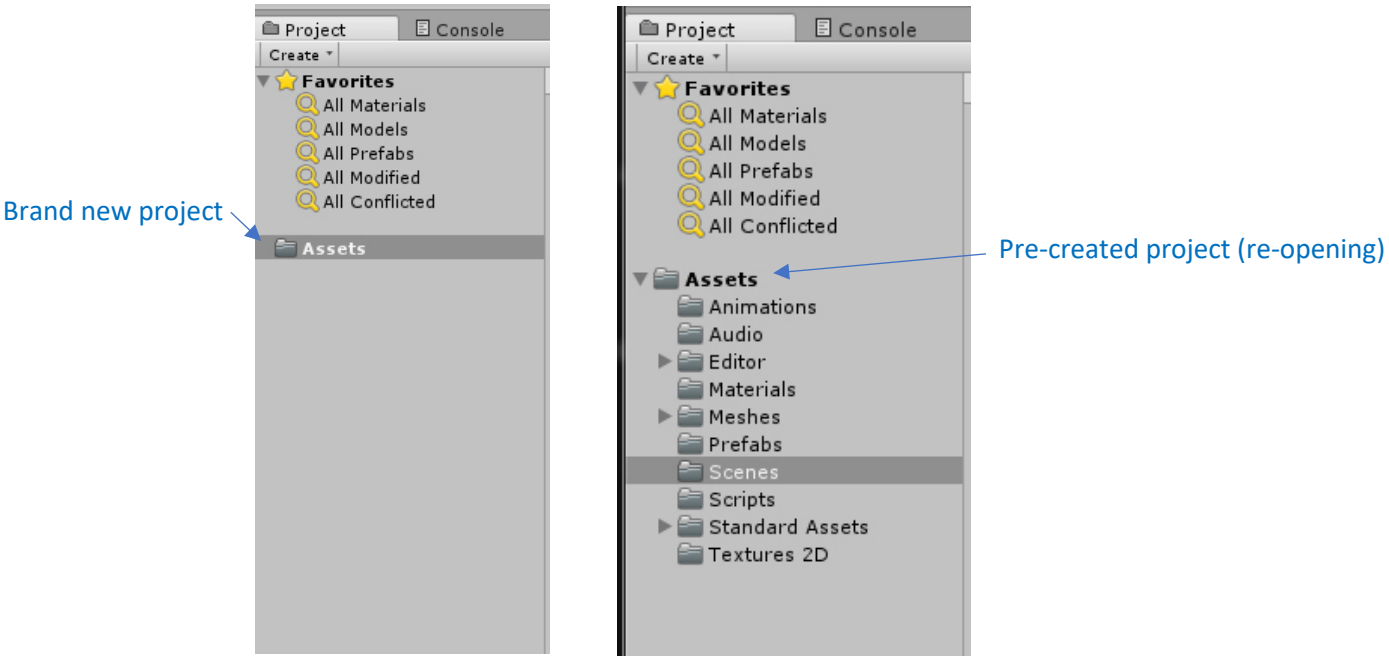
      ii. Later, when you try to open a project, Unity will only recognize this folder as the project structure b/c it holds the Assets and Library folders. Everything else will gray out.

   b. **Library Folder**:
      i. This is the brains of the project. We don't want to mess with the contents of this folder. This folder is the 'memory' of our game, remembering the connections being made between everything in our entire project. This should remain untouched!

   c. **Assets Folder**:
      i. This is where we'll work from, but we'll do it via the Unity interface. We don't want to edit our files using the windows explorer unless we have to b/c it can lead to breaks.

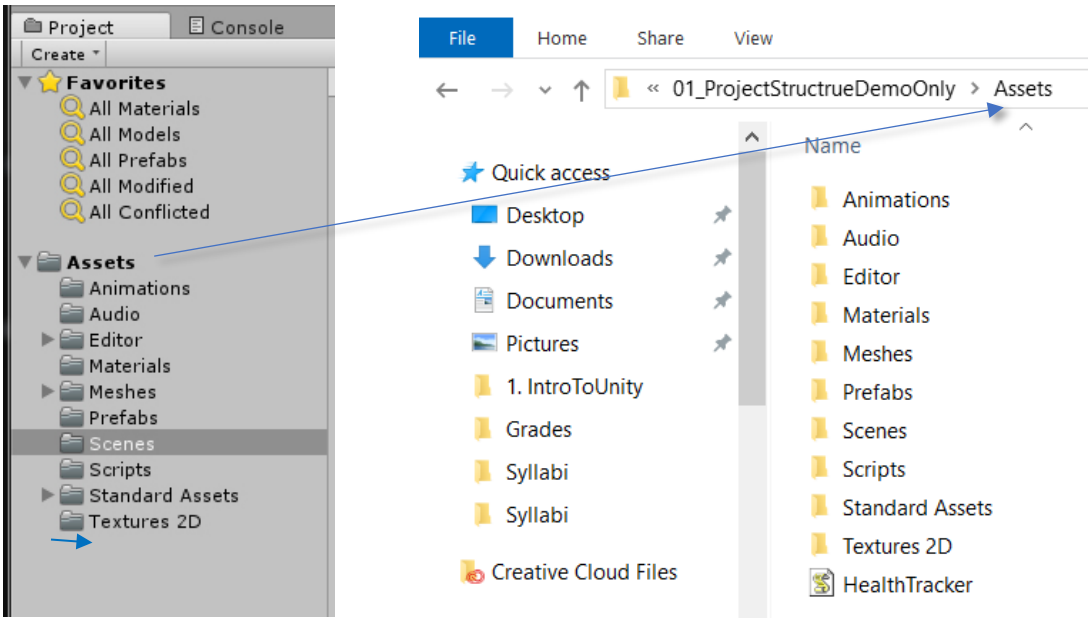
      ii. If we created a new project from scratch, this folder would be empty to start.

      iii. In Unity, this folder system is represented and found in the Project Tab:



iv. If you look at the **Project Tab** in Unity, you'll find that it's identical to what we see in the Assets folder. If we were to make a change to something in the project structure within Unity, like adjust the name of a material or a folder for example, Unity will go in and make all of the necessary updates to whatever connections it may have. It will maintain stability for us! If, however, we were to go into the file structure outside of Unity, using the Windows interface and make changes, connections may be lost. It's very similar to After Effects in this way (for those who have AE experience). You can always reestablish connections in Unity but doing that can be time consuming and unnecessary. **Organization is key!!**

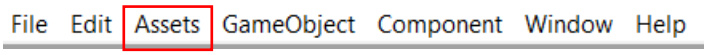
v. We can copy things into our assets folder (via the Windows interface) to pull into Unity for the first time, but once we start using it in Unity and we begin editing that new asset, we'll want to continue using the Unity interface for naming convention updates etc.



Test Scene and The Interface:

1. Let's create a new project:
   a. **Project name**: TestScene
   b. **Location**: click on the dots to navigate where you want your project to live
   c. **Template - 3D vs. 2D:** 3D
   d. **Asset Packages**...: Leave alone for now
   e. Hit **Create Project**

2. Creating assets in Unity:
   a. Assets (in the main menu) → Create → select the asset type you want to create



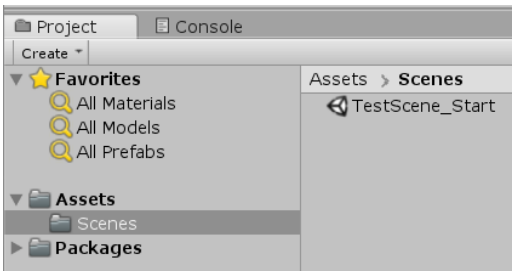
      i. Whatever you create will also be created in the Assets folder of the project structure in your Windows explorer.
      ii. To test this:
         • Assets → Create → Folder → Name it
            ➡ Note – now look in your project structure in the Windows explorer

3. **Scene File**:
   a. **IF** you don't see a Scenes folder nested under Assets, we need to create one. In newer versions of Unity, one is created automatically, and our file will be saved to it. To do this manually:

      i. Assets → Create → Folder → name it 'Scenes'

      ii. This folder holds all of the pieces of an individual part of our game. You could say that a scene file is like a 'level', but that's somewhat limiting. Scenes can be levels and they frequently are, but they can also just be various parts of the game, like a start-up screen, or a 'high scores' screen, or starting menus for example. Think of scene files as individual levels or individual parts of the game.

      iii. File → SaveAs →'TestScene_Start'.unity
         • If you see a 'SampleScene' in there, just delete it.

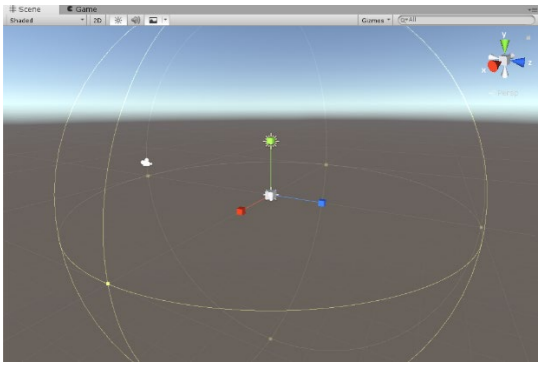


4. **Game Objects**:
   a. Everything that we see or bring into our scene is a 'game object', which is kind of like a container waiting to hold something that we define via the components...whether it's a mesh, or a light or an effect.

      i. For example, if we brought in a model cup, it's a game object with a mesh defined as its component. Game objects and their components are what really make up any game. They're just reading in assets that we may define.

      ii. As another example:
         • GameObject → Create Empty
            ➡ We can move this around etc....but there's nothing defining it, so it's invisible or 'empty'. With the empty object selected:

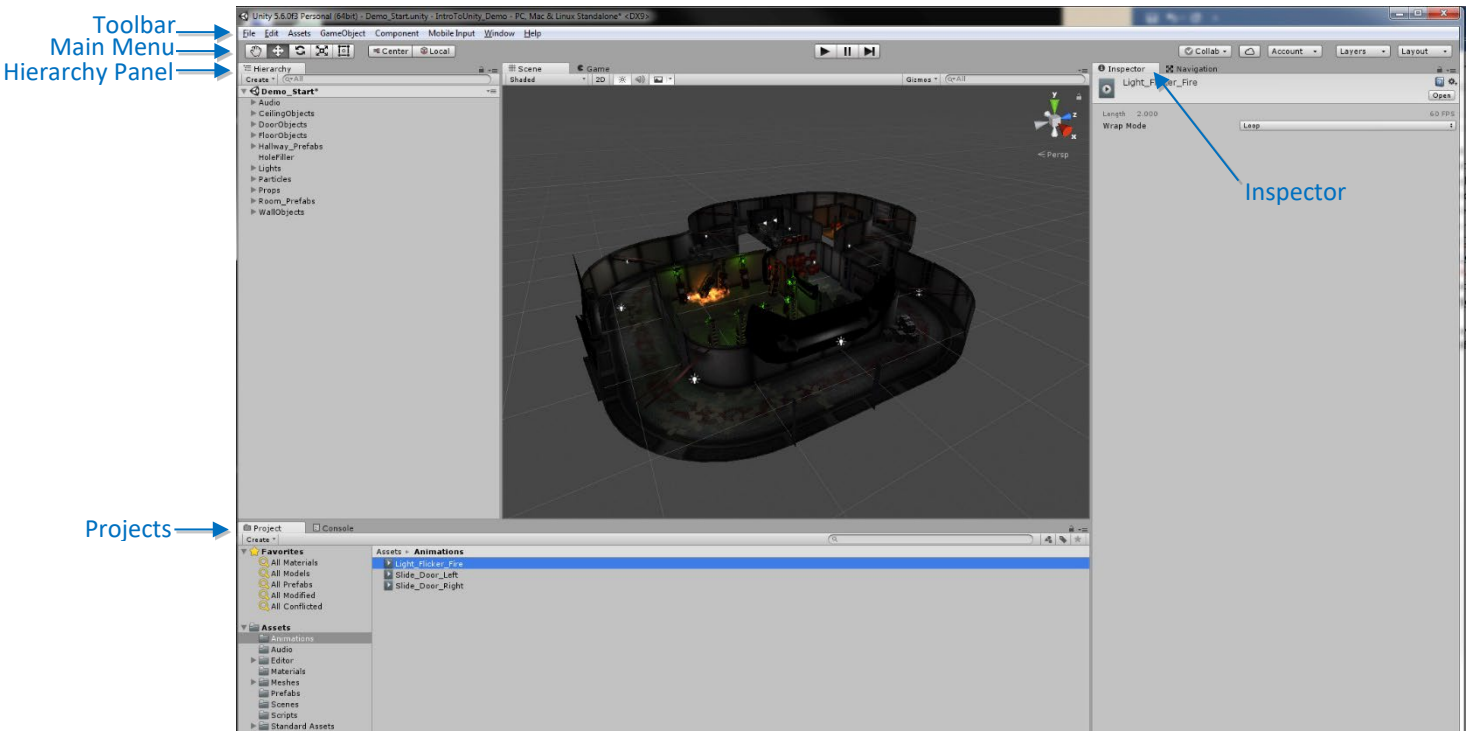
- Component → Add → Rendering → Light
  - Now there's a light component defining that game object.



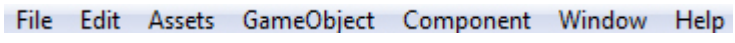
iii. So essentially everything that we do in Unity involves creating game objects, adding components to them and then modifying how they interact.

iv. Delete the Light
- Select it and hit Delete on your keyboard

5. **The Interface**:
   a. Once in the Unity interface there are several menus and panels to work with:



   b. **Main Menu**

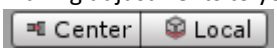


      i. This is where you have your standard menu items for saving, opening, making panels visible etc... You can also access game objects and create game objects from here.

   c. **Toolbar**

i. This holds the manipulator tools (scale, rotate, translate). Most of the time we will use hotkeys for these.

ii. Making adjustments to your pivot point (which we'll talk about in a bit).



iii. Layers and Layout:

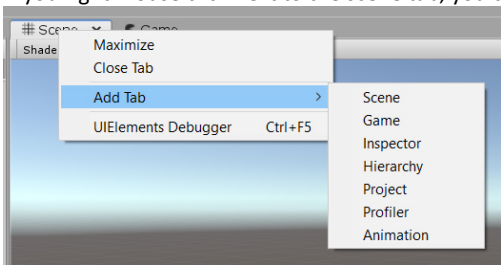
- **Layers**: This allows you to quickly show and hide layers of content in your game
- **Layout:** This is where you can change up what your default interface looks like if you need to move panels around. Any changes that you make you can save out for future use.
  - To detach or move tabs, simply click on the tab and drag it.
  - To add or hide tabs you can do one of three things:
    - Rt mouse click in the panel area → Close Tab OR Add Tab
    - Window → (select the panel you want)
    - Click on one of these:

f. **Hierarchy Panel**:

i. Shows us all of the game assets that are being used in the scene as we well as assets that are parented and/or grouped. Much like the Outliner in Maya. IF we delete content from this location, it will simply be removed from the scene not the project structure. IF we delete something from the Project tab, this DOES remove items from our project structure. So, we need to be careful.

g. **Scene Panel/Main Workspace**:

i. If you right mouse click next to the scene tab, you can pull up other tabs to toggle between.

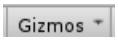


ii. **Scene Tab** - This is the main workspace

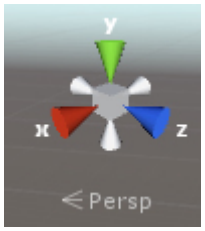
- **Shaded (drop down)**: allows you to see your scene in different modes (shaded, wireframe, wire on shaded, etc)
- **2D**: this where you'd switch to a 2D view

  - If you choose 2D at the beginning in the launcher, this is the view you'd see instead.
- **Light Icon**:

  - We can use this button to turn on/off our scene lights or default lights

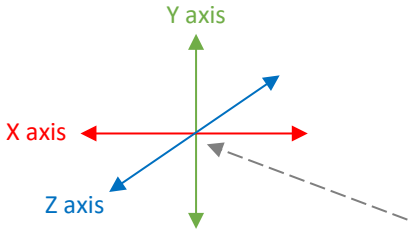
- **Audio Icon**:
  
  - Allow us to turn audio on/off if we have it in the scene
- **Gizmos** dropdown:
  

  - Allows you to turn things off/on in the scene… like the Grid or Selection Outline
- **Gizmo / Axis Handles**:



  - If you click on any of the arms, it'll switch your camera to that view. If you click on the middle square after you've clicked on one of the handles, it'll give you an absolute view or an orthographic view...meaning it'll take the perspective out. You may also hear this referred to as Isometric view.

iii. The grid:
- 0,0,0 is called the **Origin**
- The 3D Coordinate system is: **X, Y, Z** (always expressed in that order)

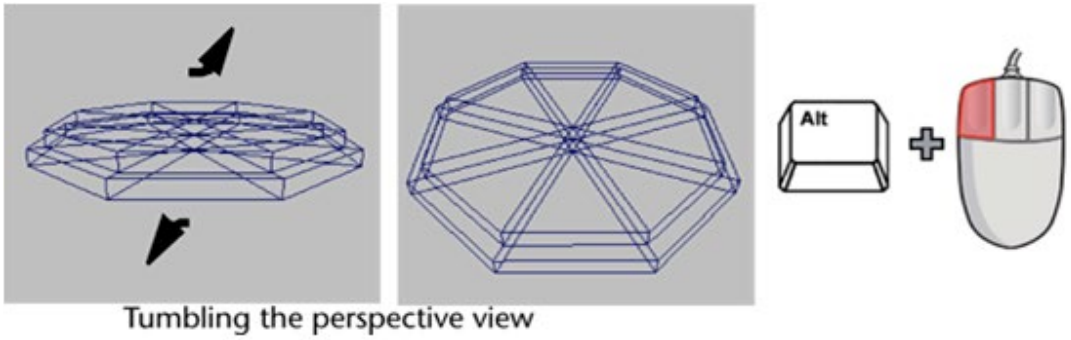


The Origin is where X, Y, Z intersect (0, 0, 0). It's the center of our scene.

Note -In Unity **Y** is up, in Maya **Y** is up but in 3dsMax and Zbrush, **Z** is up. This is important to understand when you move in and out of other software packages.

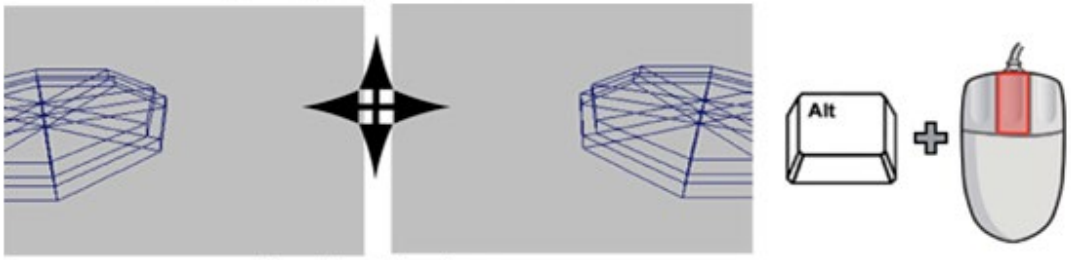
iv. Navigating within the scene:
- These are the same hotkeys used in Maya!

Rotate/Orbit



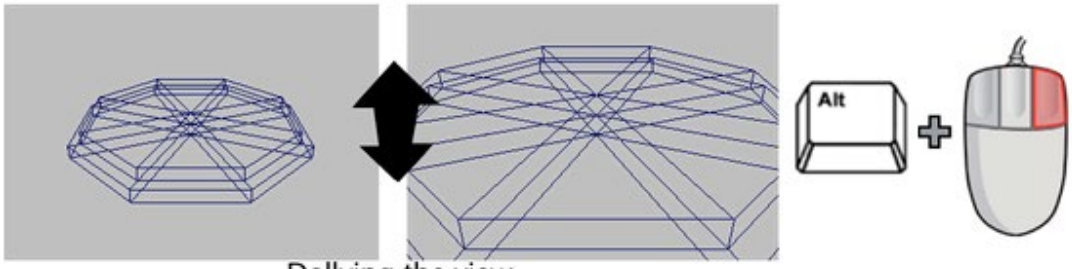
Tumbling the perspective view

Panning



Tracking the view

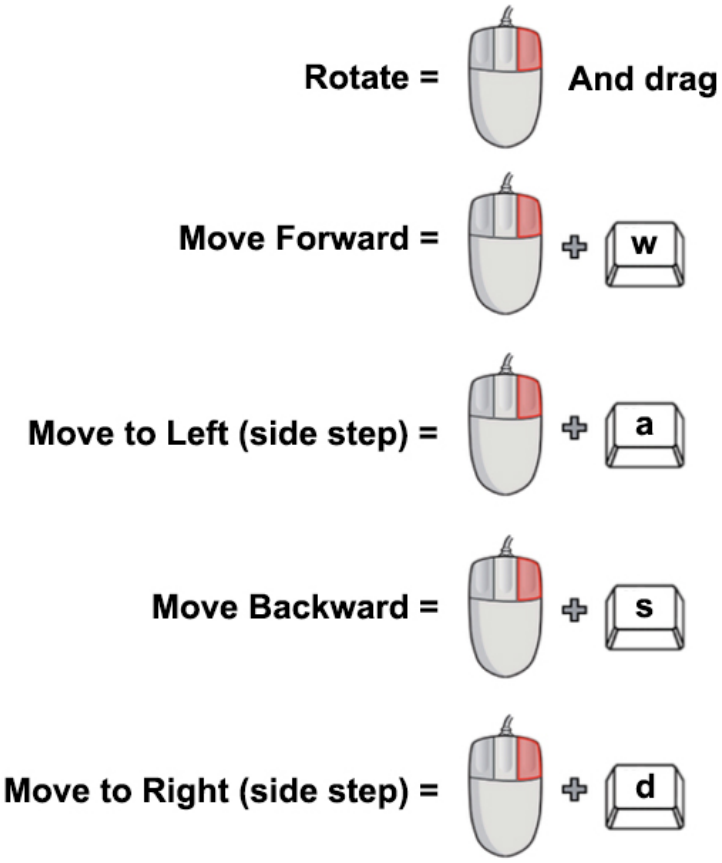
Zoom In/Out

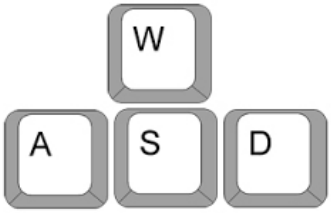


Dollying the view

- **Zoom in (of Focus) on a selected: Hotkey = f**
  - Also the same in Maya
  - Once you frame in on an object the viewer rotation gets set to that object.

- **First Person Controls** (kind of like you're standing in place and looking through the camera rather than looking through and rotating or moving the camera):





- Holding down **Shift** with any of these will speed things up. This is the same as first person controls for a PC game.

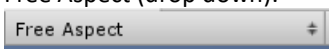
h. **Game Panel/Tab:**

i. Rt mouse click next to the scene tab and pull up the **Game Tab** (if it's not already up)**:**

- Game tab:

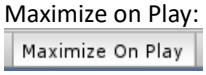
  - Allows us to look at our game in player mode
  - This is going to allow us to see through the camera that's in the scene. If we don't have a camera created, we'll need to create one before using the Game Tab.
  - If you were to hit any of the player controls, it will automatically switch you to this panel.

- Free Aspect (drop down):



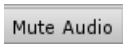
  - This is where you can change the aspect ratio of your game

- Maximize on Play:



  - With this on, our view will maximize to our screen size upon hitting the play button.

- Mute Auto:

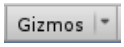


  - This will mute any game audio

- Stats:

  - This turns on information about our game, like Frames Per Second (FPS). This becomes extremely important, especially if we're creating for immersive environments.
    - 30 fps is the bare minimum for ok game play
    - 60 fps is good for PC or console games
    - 90 fps+ is really needed for VR games in order to avoid motion sickness

- Gizmos:



  - Turns the Gizmos off/on during game play testing

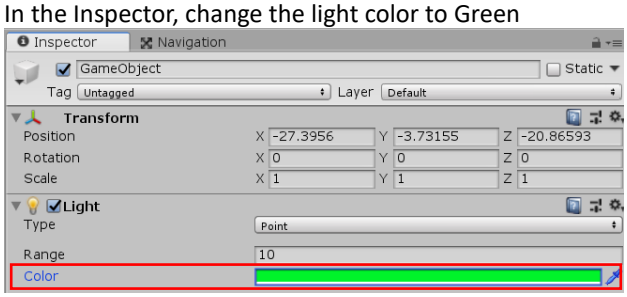
ii. Player controls for testing the game inside the editor:

Use the standard PC controls to navigate – Rt mouse click with **w, s, a, d, spacebar**

- To get out of game mode, you can click on the play button again or use the keyboard shortcut **cntrl p**.

- To pause or unpause the game you can hit the pause button or **cntrl shift p.** You would do this to frame in on a spot you want to test fix or do something to. This is super useful b/c you can edit stuff while you're play testing. The hiccup with this however is that as soon as you get out of play test mode, those changes won't be saved. That's b/c Unity is using that mode to allow you to experiment without potentially creating breaks in the game. So, the best way to utilize this would be to only make a couple changes at a time while in play mode, write those changes down and then get out of play mode to actually make the adjustments. Don't make the mistake of making a ton of edits while in this mode b/c you'll lose them as soon as you hit the play button off. So, this is basically a way to test edit before you make the actual changes.

- A good way to see this:
  - Create an empty game object
  - Add a light component
  - Click the play button and then the pause button
  - In the Inspector, change the light color to Green



  - Then unclick the pause and play buttons
    - Notices that the change doesn't stay

iii. 

- The step play button allows us to play frame by frame. This is useful when you're debugging something in your scene. For example, if a light's intensity value is animated, we can see what the value is frame by frame by looking in the inspector or attributes panel.

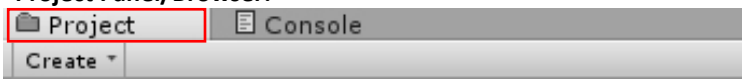
i. **Inspector Panel**

i. Anytime you have something selected this is going to display the attributes or properties that you can adjust for that asset.

ii. Select the default camera or a light to see the attributes.

- Add Component:
  - Components are things that modify the game object that you have selected (just like we did earlier). For example, you could add effects or dynamics to a particular object...This is exactly the same as using the Component drop down in the main menu.

j. **Project Panel/Browser:**



i. This holds all of the assets that are in the project (i.e. living in the project structure to be brought in for use...like textures and models)

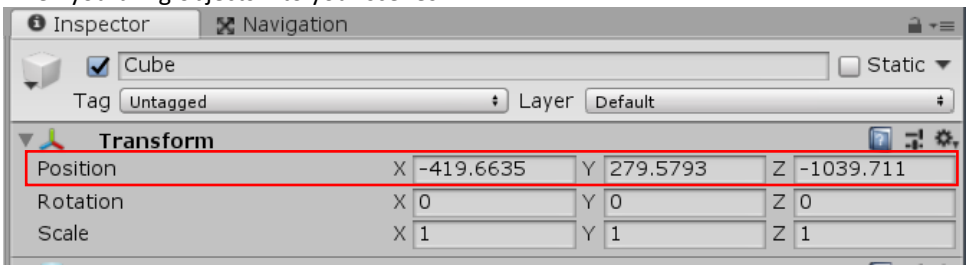
ii. It's going to be very important that we keep this organized!!

k. **Console:**

i. This is what will help you debug the game you're creating by showing error and warning messages.

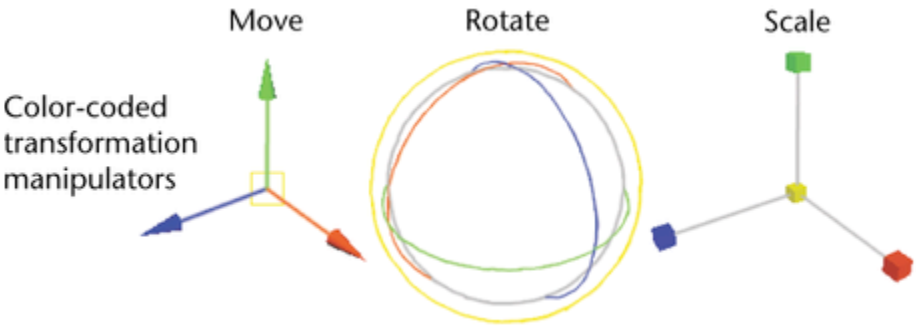
Working with Assets:

1. In the Main Menu:
   a. GameObject → 3D Object → Cube
      i. This will create a cube in our scene. Notice that in the Inspector under Transform, that the Position values may not be at the Origin. This often has to do with where our viewport is looking when we create the object. We'll want to reposition the object at the Origin to make it easier to work with. This is something to be mindful of when you bring objects into your scenes.



      ii. To adjust this to the origin:
         - Click on the gear icon in the upper right corner of the position values: 
         - **Reset**
            - This will snap our object to the Origin in our scene

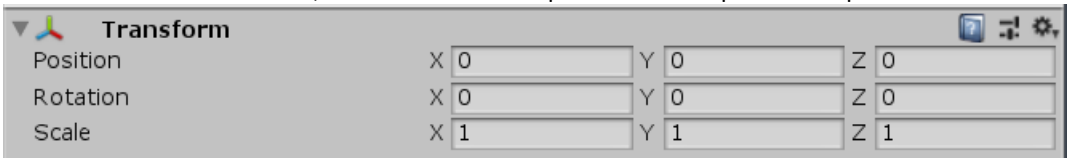
2. Unity's gizmo handles are color coordinated:



   - X is always red
   - Y is always green
   - Z is always blue

3. Hotkeys:
   a. **W** – Move/Translate
   b. **E** - Rotate
   c. **R** – Scale
   d. You can move in multiple axis's at once by grabbing the manipulator in the middle (yellow), however it's good practice to only move, rotate and scale on a specific axis at one time to ensure you're getting the result that you want.

   e. You can also use the Position, Rotation and Scale options in the Inspector to input exact coordinates.

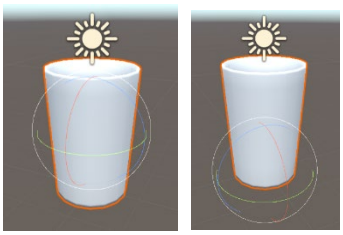


f. Making adjustments to your pivot point:

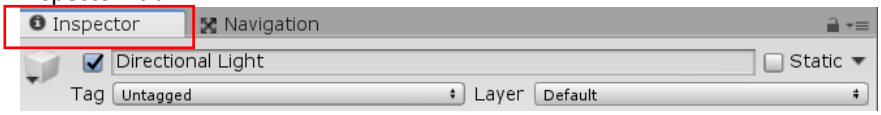
This is extremely helpful in dictating where our obj moves from! Where the pivot is placed (locally) is largely dictated by how you have it setup in your 3d file.

i. **Center** will center the pivot in the middle of the object. **Pivot** will utilize the pivot location set within the 3D package it was created in.
ii. **Global** will align the gizmo with the scene, **Local** will align it with the object's local values. You'll flip back and forth between these two depending on what you want to do.

iii. To see this:
- Rt mouse click over the Assets folder in the Project Tab → Create → Folder
  - Name it "Meshes"
- Outside of Unity, navigate to the folder I've given you called "02_ExternalAssets"
  - Find the cup file
  - Drag if from the folder into the Meshes folder within Unity (we can drag right from our desktop or drives like this).
    - Notice that when we do that, Unity recognizes this as an asset we want to use in our project. So, it makes a duplicate of that cup file in our project structure, not only in Unity but in our TestScene folder.
- In Unity, double click on the Meshes folder
  - Drag the cup into the scene
  - Using the steps described above, reset its position to the Origin
  - Now click on **Center** and **Pivot** to see the difference
  - You won't see much shift between **Global** and **Local**, but you will for more complex objects



4. Attributes:
   a. Select the light in the scene
   b. Inspector Tab:



i. Anywhere there's an attribute you can:
- Enter in values
- Move sliders
- Lft mouse drag over an attribute name and slide from side to side, in order to make adjustments

ii. You can lock the Inspector to whatever obj you have selected. Once you lock it, you can click on other objects in your scene, but the inspector will stay locked on the object you locked it on. I personally don't use this a great deal.

c. Debug mode:
   i. Rt mouse click over the word Inspector → switch from Normal to Debug
   - This will show you some hidden attributes along with our other attributes in a more compact setting. I don't recommend doing your editing in this area for now. But just know that it's an option.

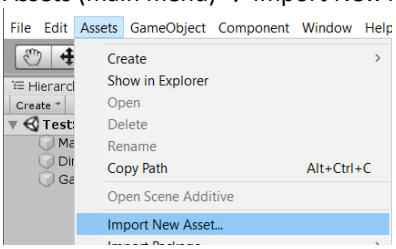
Importing Assets:

1. Method 01:
   a. As we've already seen, we can drag assets into our Unity project right from the desktop. When we do this, we can drag directly to the Assets folder and our project structure will update accordingly.

   b. You can create and have your asset files organized outside of the Unity project structure (which I will often have to keep things clean and accessible), but once you pull them in, Unity will copy them to the project structure wherever you designate them to be.... if you're pulling in models, you'll probably put them in the 'meshes' or 'models' folder. Once they're in the project structure, make sure that any changes you make to the location or naming convention is done through the Unity interface so that things don't break.

2. Method 02:
   a. We can drag the new assets into the appropriate folders in the windows project structure. Just make sure that if you're doing this, it's for assets that you haven't done anything with in Unity yet. Once you do this you should see the new assets in the Project Panel. If you don't, right mouse click in the panel → Refresh.

3. Method 03:
   a. Assets (main menu) → Import New Asset

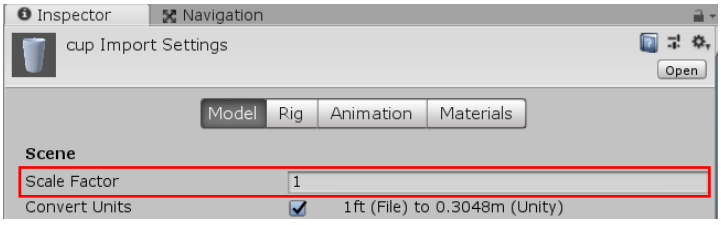


      i. In my example: External Assets → Cup

4. Two ways to place items in the scene:
   a. Drag the obj to the Hierarchy tab
      i. You can drag them into existing GameObjects (that can act like groups)
      ii. Or they can live independently
   b. Drag the objects to the viewport directly

5. **Scale**: What happens if your object comes in and its scale isn't correctly? How do you fix that?
   a. 1st way: you could just scale it in your scene **(Independent Scaling)**
      i. The issue with this is that it will only scale that instance of the model. If we want to have the scale address every instance of that model in the scene, we'll want to scale the asset in our Project Tab.

   b. Select the obj (in this case the cup) in the Project tab (**Universal Scaling**)
      i. Inspector tab → Model → Scene section



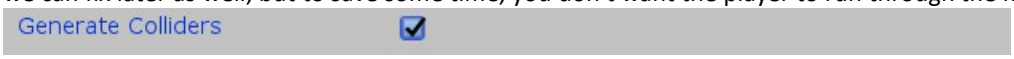
- **Scale Factor**: set to 1 if it isn't or whatever scale factor you want.
  - Scale factor is used for compensating difference in units between Unity and 3d modeling tools - it rescales the whole file. Normally you can simply set it to 1. Note that Unity's Physics Engine is scaled as 1 unit equals 1 meter. It is important that if you want to have correct physical behavior you should have the model correctly scaled in the original modeling application. If this cannot be done, or you do not have control over the modification of the mesh, the scale of the model can be adjusted here.

- Once you enter a value and then click somewhere else in Unity a dialogue box will pop up and ask you to confirm the change. **Hit Apply**.

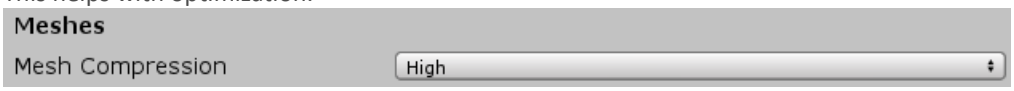
- You can also **hit Apply** in the lower right corner of the Inspector attributes area.

  - This change will affect any instance of that cup we have in the scene (ie. It's a global change)
  - It's good practice if you're working in a team, to agree upon a scale factor at the beginning when you're creating your assets so that you don't have to make as many adjustments.

- **Generate Colliders**: Check On
  - Another adjustment that you can and probably should make to your object that may save you a lot of time is to make it collidable on import. By default, this is checked off, so if we were to go into player mode, we could just walk through meshes that didn't have collidable properties on. This is something we can fix later as well, but to save some time, you don't want the player to run through the mesh



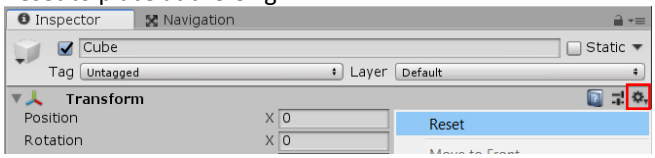
- **Mesh compression:**
  - Increasing this value will reduce the file size of the mesh but might introduce irregularities. It's best to turn it up as high as possible without the mesh looking too different from the uncompressed version. This helps with optimization.



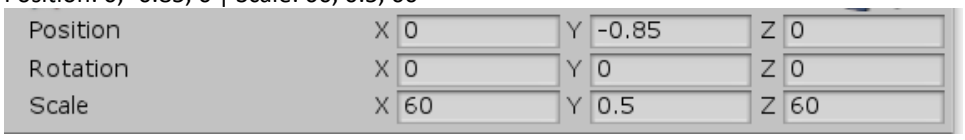
  - Hit **Apply**

Setting up a Character (using a standard package):

1. To set this up, let's create a basic environment to place our character in.
   a. GameObject (Main Menu) → 3D Object → Cube
   b. Inspector →
      i. Transform:
         - Reset to place at the Origin

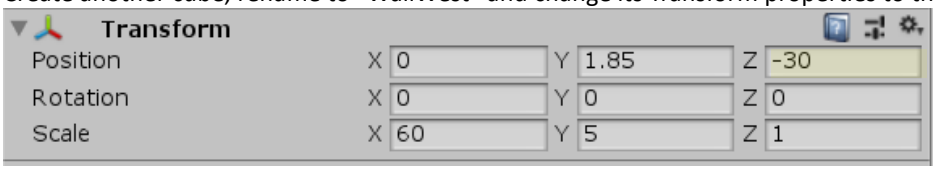


         - Position: 0, -0.85, 0 | Scale: 60, 0.5, 60

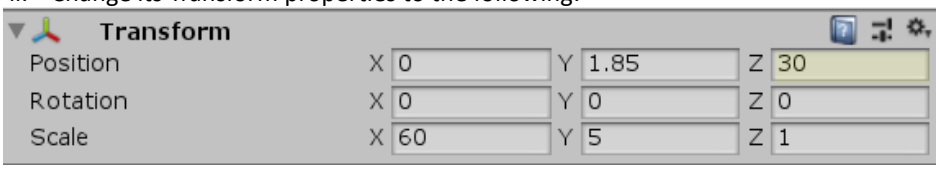


   c. Rename the Cube to "Floor". You can do this in Three ways:
      i. Click on it in the Hierarchy tab
      ii. Rt mouse click over it in the Hierarchy tab → Rename
      iii. Change the name in the Inspector

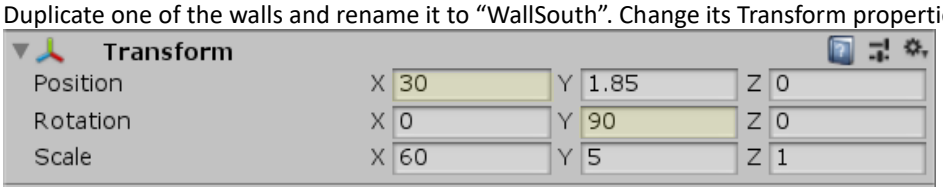
d. Create another cube, rename to “WallWest” and change its Transform properties to the following:



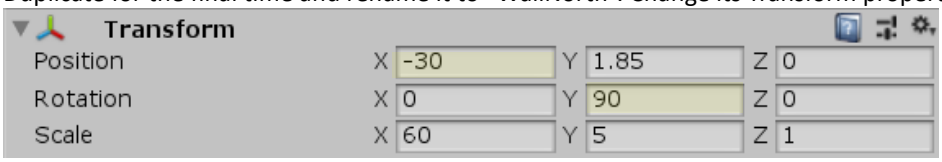
e. Create a third cube by **Duplicating** the first, rename it to “WallEast”, to duplicate:
   i. Select the first wall and then hit **Cntrl d**
   ii. Change its Transform properties to the following:



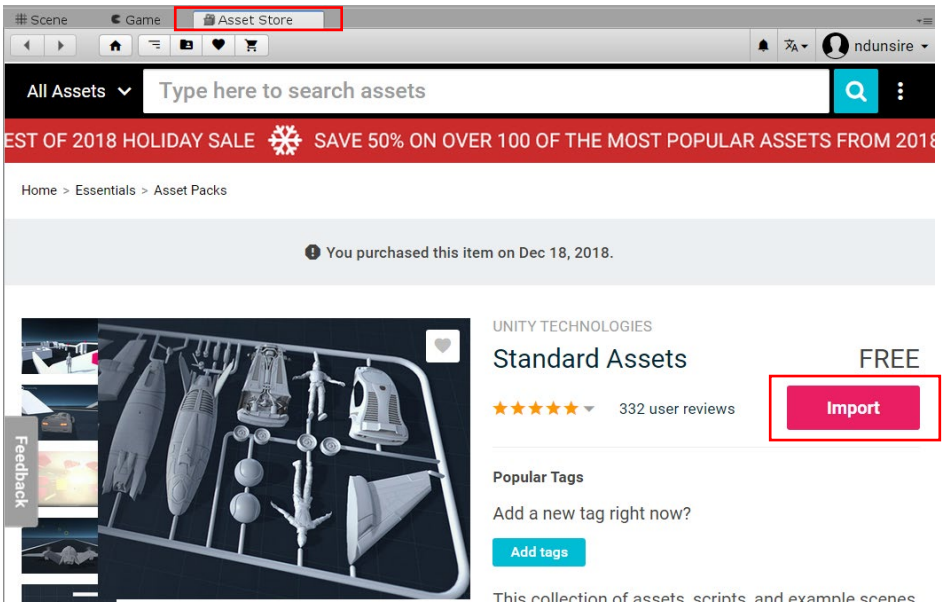
f. Duplicate one of the walls and rename it to “WallSouth”. Change its Transform properties to the following:



g. Duplicate for the final time and rename it to “WallNorth”. Change its Transform properties to the following:

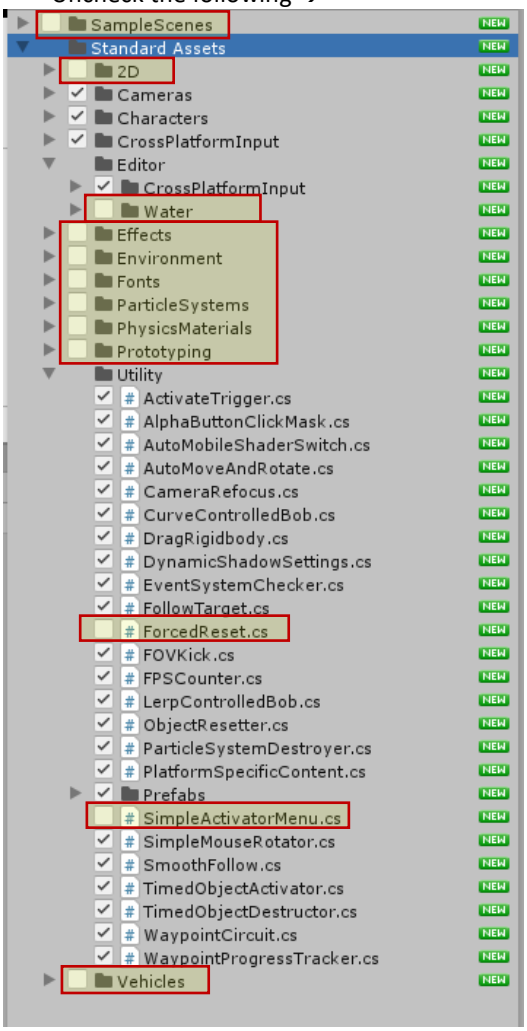


2. We would like to bring in a standard character package to play with.
   a. Window → **Assets Store**
      i. This will open up the Asset Store Tab of your viewport within Unity.
      ii. Search for “Standard Assets”



iii. **Hit Import** and a dialogue box will appear:

- Uncheck the following →



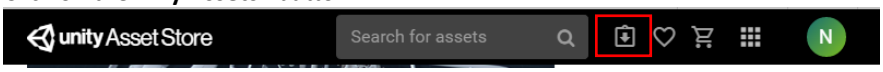
- This will import everything we need for our preset character into our Assets folder.

b. The next time you want to import this asset package to a project:

i. Go to the **Asset Store** tab of your viewport

- If you don't see it, remember that you can Rt mouse click over one of the open tabs and choose other tabs to pull up, including this one

ii. Click on the "**My Assets**" button



- This will show you the assets that you've downloaded or purchased from the Asset Store

iii. Next to the Standard Assets Package option choose **Import**

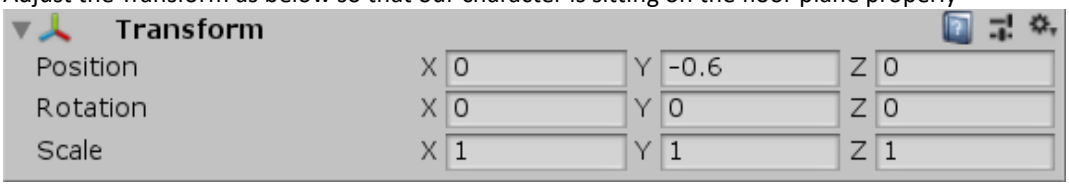
iv. Again, the dialogue box will appear, choose what you want and hit Import

c. With that all said, once you have an Asset package imported, you can Export it and save it locally so that you don't have to get online to access it. This is something I've already done and have given you. So, if you don't have a Unity account or internet access, you can import this way:

i. Rt mouse click over Assets in your Project Panel

- Import package → Custom Package
  - Navigate to the "StandardAssetsPackage" unity file that I've given you and hit Import
  - The dialogue box will appear, and you can choose the options you want as above

3. Assets → Standard Assets → Characters →
   a. **ThirdPersonCharacter** → Prefabs Folder → Drag 'ThirdPersonController' into the scene
   b. Reset its position so that It's at the origin
   c. Adjust the Transform as below so that our character is sitting on the floor plane properly



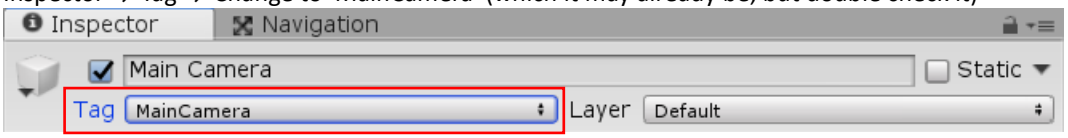
4. Select the **MainCamera** in the Hierarchy
   a. Using your move and rotate hotkeys, Position it back behind the character in a manner that you'd want to see it in play mode. Use the Camera Preview window to help position it.

   b. **Hit Play** and see the result
      i. Right now, we can:
         - See the animation of our character in idle mode. This is an animation done in a program like Maya or 3dsMax. The file is then saved as an .fbx (which is a file format that holds animation information) In Unity, the attributes for this character are setup to play this idle animation file, while the character isn't in motion.

         - Try using the controls: **WASD Spacebar**. Like before there are different animation being played and movement happening depending on what keyboard keys are pressed. BUT the camera doesn't follow which is a problem right now.

ii. To address this:

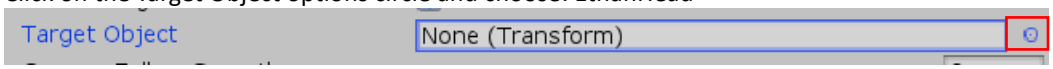
- Get out of Play mode
- Select the Camera
  - Inspector → Tag → Change to 'MainCamera' (which it may already be, but double check it)



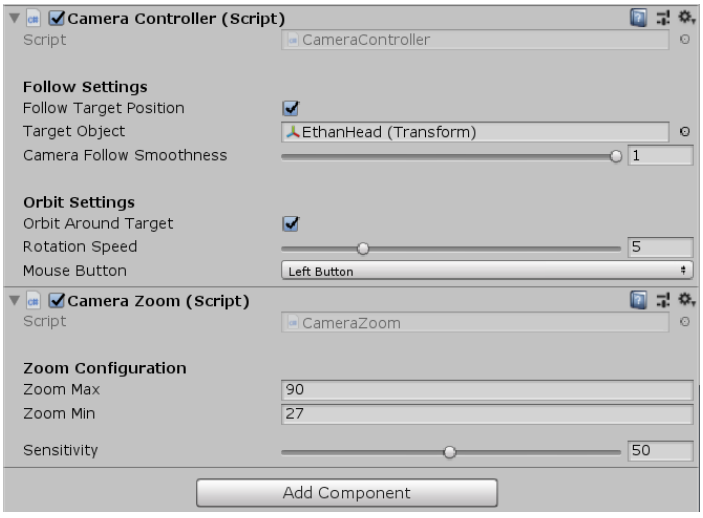
- Now, we need to add some scripts to this so that the camera follows the character as he moves AND so that we can rotate / orbit. Because this is fairly typical movement, this is something we can also find in the Assets Store, assuming you don't have a background in C# coding. If you do have the ability to write code, then you can write you own. For now, I'm going to give you scripts to work with and test out.

  - Rt Mouse click over Assets in the Project Tab → Create → Folder
    - Name it "Scripts"
  - Drag and drop the two scripts that I'm giving you to this folder
    - **Camera Controller (C#)**
    - **CameraZoom (C#)**

- Select the Main Camera
- In the Scripts folder drag the **CameraController script** onto the camera in the scene. Then drag the **CameraZoom script** onto the camera. We should now see them in the Inspector:
  - Camera Controller (Script)
    - Click on the Target Object options circle and choose: EthanHead



    - Camera Follow Smoothness: set to 1.0
    - Rotation Speed: 5
    - Mouse Button: Left Button
  - Camera Zoom (Script)
    - Zoom Max: 90
    - Zoom Min: 27
    - Sensitivity: 50



- Let's test. Hit the Play button
- Now your Lft mouse button controls the camera rotation and the camera follows the character.
  - Use **W, S, A, D, Spacebar** on your keyboard
  - If you hold down **shift** at the same time, the character will go into walk mode.
    - Play with the attributes for desired results (remembering that now that we have a script driving the camera, we're at the whim what control options the script writer gave us in terms of placement etc).
- Note – we could have also added these scripts by hitting the Add Component button in the Inspector and navigating to the scripts there. I just find that the drag and drop method is a little easier and faster.

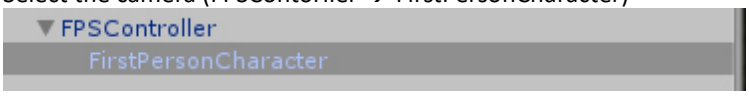
5. Now let's try **FirstPersonCharacter**
   a. Hierarchy → Delete
      i. ThirdPersonController
      ii. Camera

   b. Project structure → StandardAssets → Characters → FirstPersonCharacter → Prefabs → drag the **FPSController** into the scene
      i. Reset its position
      ii. Adjust the Y Position to 0.25 (just up off the ground a bit)

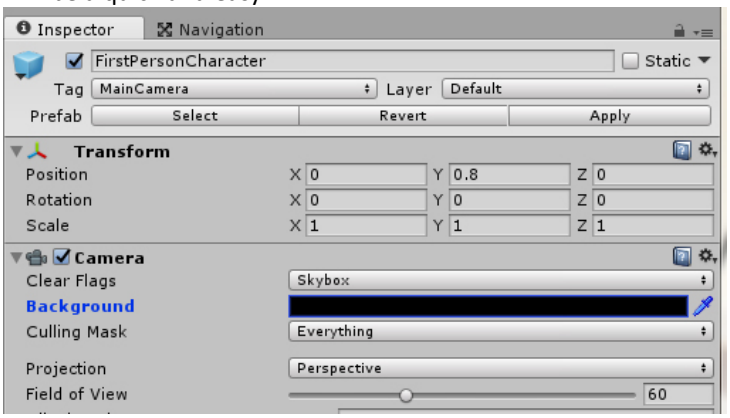
   c. Hit play to see results.
      i. If you select the FPSController, you have a lot of attributes in the inspector that you can play with to get the settings that you want....
      ii. Don't forget that **Cntrl p** gets you out of play mode

   d. If you're finding that as you move around your scene, you're seeing small gaps in the geometry, one way to fix that quickly:

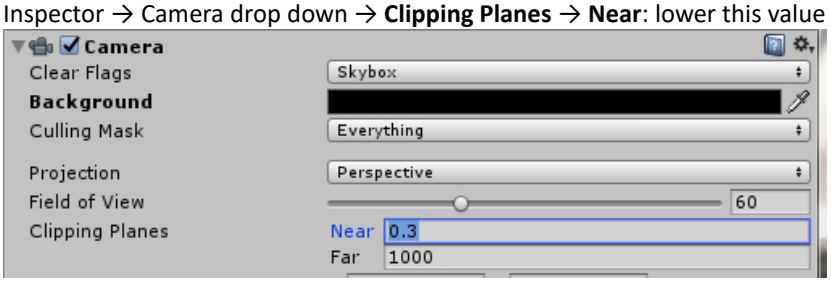
      i. Select the camera (FPSContorller → FirstPersonCharacter)



      ii. Inspector → Camera drop down → **Background** → Change to whatever color your geometry is.
         - It's just a quick fake. You'd do something like this if you're bringing in modular geometry and something was slightly off. Rather than taking the time to go back into the 3d application or trying to adjust in Unity, this can be a quick and easy fix.



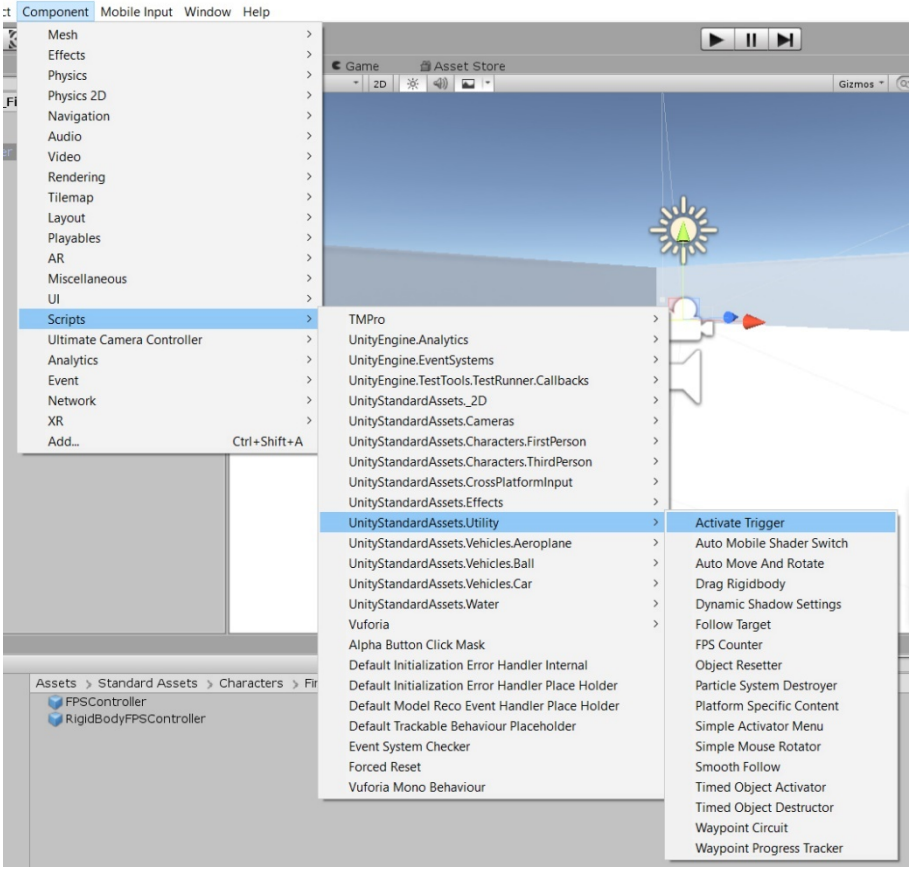
e. If you're finding that as you move close to walls you can start to see through them, you can fix this by:
   i. Selecting the camera (FPSContorller → FirstPersonCharacter)
   ii. Inspector → Camera drop down → **Clipping Planes** → **Near**: lower this value



Using Trigger Scripts to create interactivity:

1. What we'd like to do is have some things happen as the character interacts with objects in the scene. For example, we'd like a set of doors to open. In order to do most things in Unity, we'll have to use scripting. But the nice thing about Unity is that it does give us some options to start with:

   a. Component → Scripts → UnityStandardAssets.Utility → **ActivateTrigger** (this is what we're going to use)

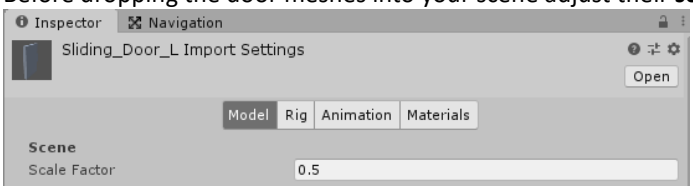


   i. **ActivateTrigger** is a multi-purpose tool to begin creating interactivity.

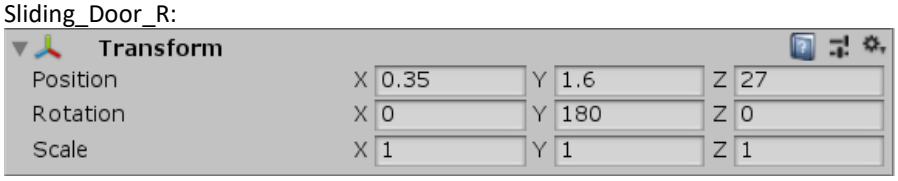
   ii. Before we add this to anything, we want to decide how this trigger is going to be activated, or how the character is going to interact. That interaction could be a key press, or the character moving into some other object, etc... In this case we want doors to open when the character gets close to them. So, we're going to use a trigger, with a collider.

2. To start:
    a. Starting with our FPS Controlled character scene
    b. Create a folder called “Meshes” and another called “Animation”
    c. From the External Assets folder that I gave you:
        i. Doors → drag the following into the Meshes folder of your project structure:
            • Doors Folder → Sliding_Door_L & Sliding_Door_R
            • Animations Folder → Slide_Door_Left.anim & Slide_Door_Right.anim
                ➡ (If you want to make an Animation folder in your project structure and place these here you can)

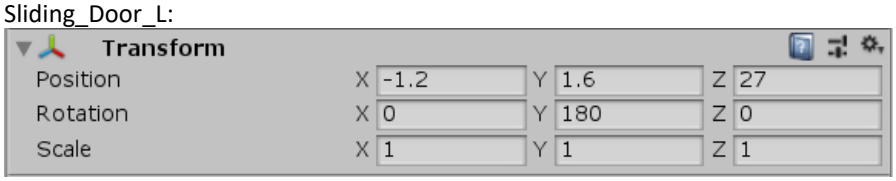
    d. Before dropping the door meshes into your scene adjust their **scale factors to 0.5**. Don’t forget to **hit Apply**!



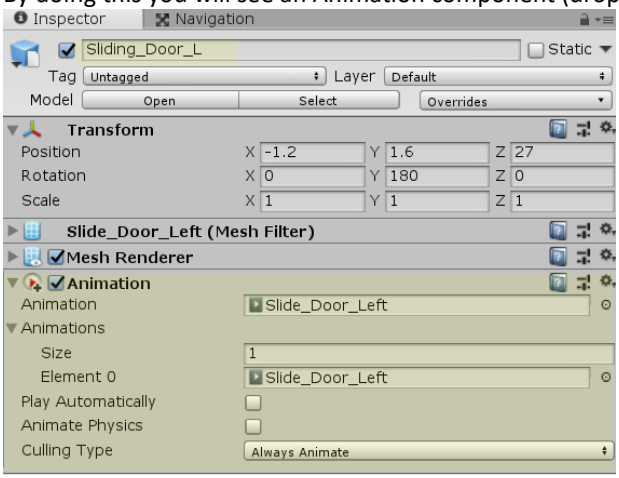
    e. Drop Sliding_Door_L & Sliding_Door_R into your scene
        i. Reset their Transforms
        ii. Adjust their positioning:
            • Sliding_Door_R:



            • Sliding_Door_L:

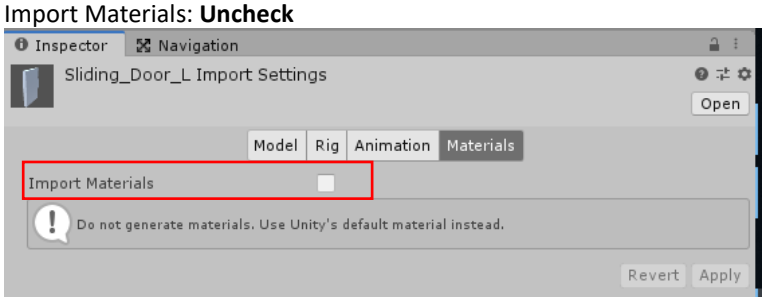


    f. Drag the Slide_Door_Left anim file ONTO Sliding_Door_L in the scene
        i. By doing this you will see an Animation component (drop down) now present in the Inspector for that door

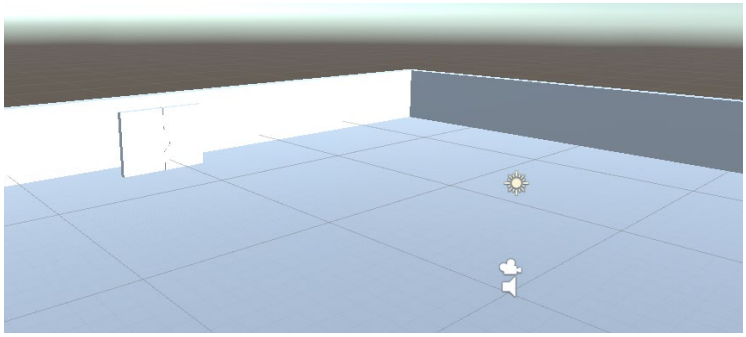


    g. Drag the Slide_Door_Right anim file ONTO Sliding_Door_R in the scene

h. Reset the Material values for the doors (we'll address materials and textures later):
   i. In the Meshes Folder of the Project Tab
   ii. Select Sliding_Door_L
   iii. In the Inspector, click on the Materials Tab
      - Import Materials: **Uncheck**

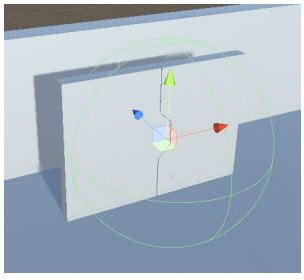


   iv. Repeat for Sliding_Door_R

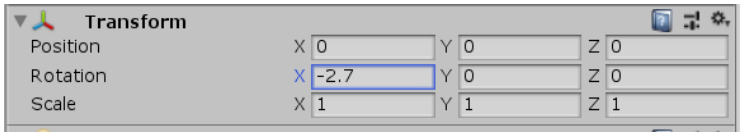


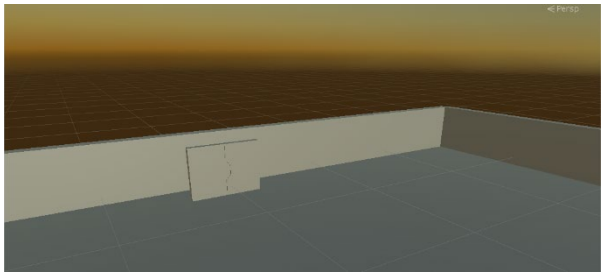
i. GameObject → Create Empty
   i. Move it out to right in front of the doors
   ii. Right now, this is a game object that does nothing. We need to tell it, that it's going to be a colliding trigger

j. With the GameObject selected:
   i. Inspector → Add Component → Physics → Sphere Collider
      - **Radius**: increase to say 3.0
      - Position it in front of the door as needed
      - Name the GameObject: DoorOpen_Collider
      - Now we have an invisible object that our character can collide with

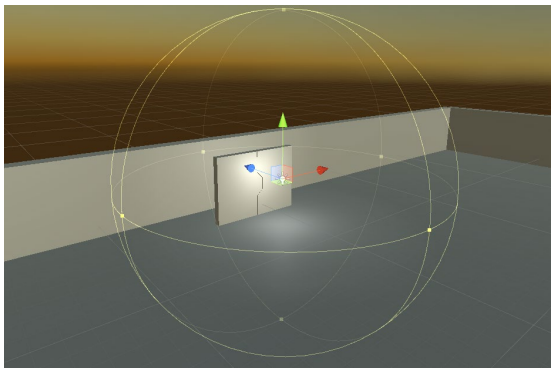


k. Let's also adjust our lighting so that as the door opens a light comes on:
   i. Select the Directional Light in the scene
      - Rotate it so that we get more of a dusk
      - Inspector →
        - Intensity: 0.2
        - Shadow Type: No Shadows

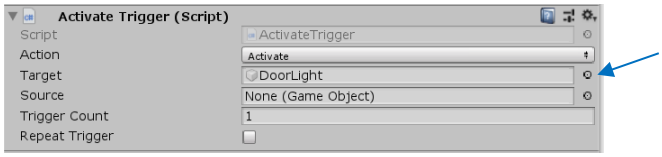




   ii. GameObject → Light → Point Light
      - Place this in front of the doors and rename to "DoorLight"

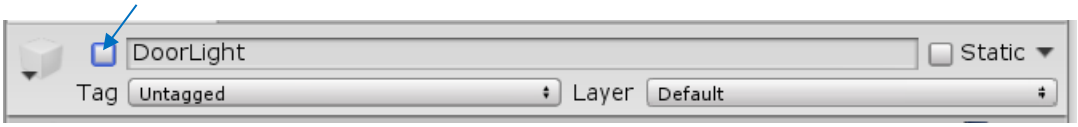


l. Select the DoorOpen_Collider game object we created
m. Inspector → Add Component → **Scripts** → **UnityStandardAssets.Utility** → **ActivateTrigger**
   i. The options that we're seeing are the variables that the script is exposing to us for use.
   ii. Action:
      - Activate – good to use for turning things on (like lights for example)
        - Inspector → **ActivateTrigger** (script) drop down
          - **Action**: Activate
          - **Target**: select the DoorLight



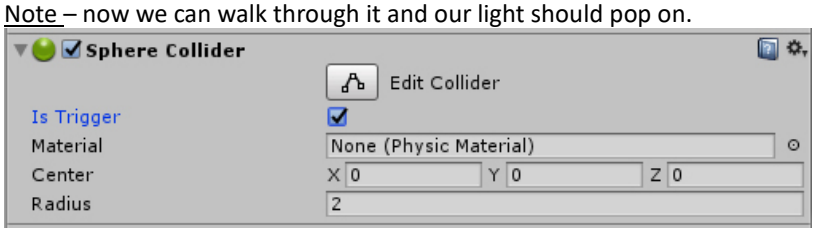
      - Note – we can either use the option box as we've been doing, or we can drag it from the Hierarchy directly to the option box. This can be easier and faster.

- Select the DoorLight (let's turn it off to start)
  - Inspector → Uncheck the box at the top next to the name box



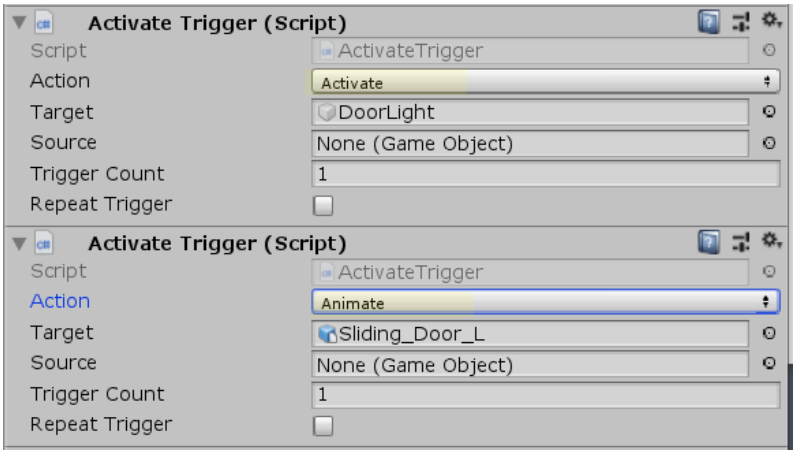
  - Now we'd like for the light to come on when the character hits the collider, so we need to tell that collider that it's a trigger...

➢ Select the DoorOpen_Collider/game object
- Inspector → Sphere Collider drop down →
  - **Is Trigger**: Check ON
  - Note – now we can walk through it and our light should pop on.

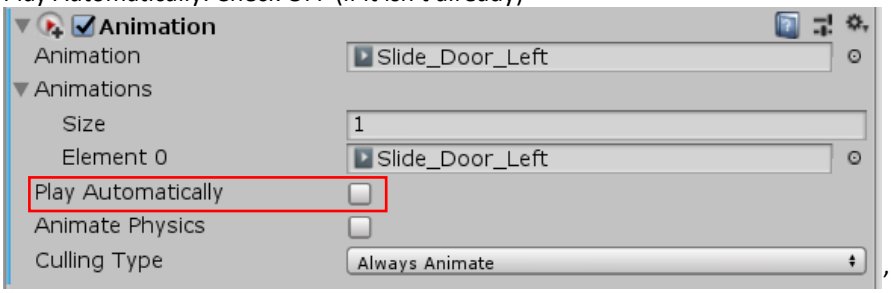


n. Now let's work with the doors:

i. Select the DoorOpen_Collider
- Add Component → Scripts → UnityStandardAssets.Utility → ActivateTrigger
  - **Action**: Animate
  - **Target**: select (or drag over) Sliding_Door_L



ii. Select the Sliding_Door_L object
- Inspector → Animation drop down →
  - Play Automatically: Check OFF (if it isn't already)



,

iii. Select the DoorOpen_Collider
- Add Component → Scripts → UnityStandardAssets.Utility → ActivateTrigger
  - **Action**: Animate
  - **Target**: select (or drag over) Sliding_Door_R

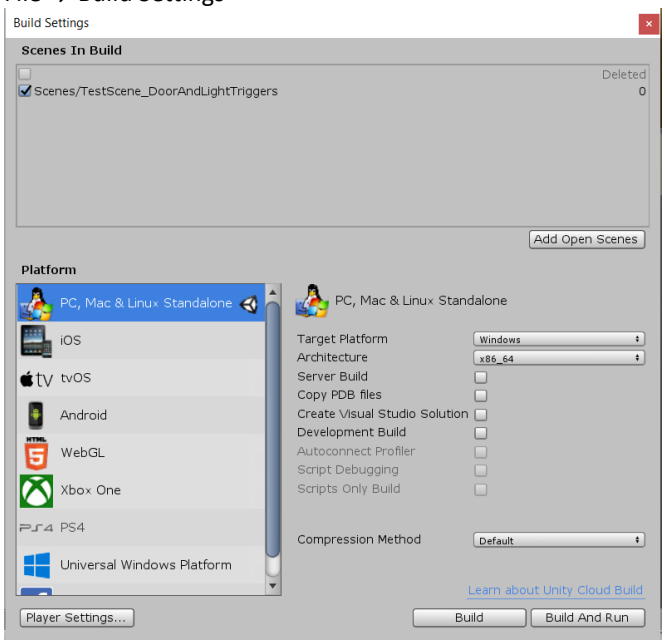
iv. Select the Sliding_Door_R object
- Inspector → Animation drop down →
  - Play Automatically: Check OFF (if it isn't already)

v. Play to test
- We should now see our door open and the light turn on as the character approaches.

i. Note -
- Select the DoorOpen_Collider
  - **Source**: set to none means that anything can trigger this action right now. If we only wanted it to be applicable to the character, we could designate that here...
  - **Trigger Count**: we can limit the number of times the trigger can happen with this...

Compiling & Finalizing for Distribution:

1. File → Build Settings



a. **Platform**: PC and Mac Standalone

b. Scenes In Build:
   i. This is the list of the scenes/levels we want to build out.
   ii. The first scene listed should be the first thing that opens up when your game starts...this is likely a start or splash screen etc.
   - Drag and drop the scenes from your Project Tab, putting them in order from top to bottom
   - In this case we have just the one scene so we can hit 'Add Open Scenes' (if we don't already see it in there)

c. **Target Platform**: Windows

d. **Architecture**: x86_64

e. **Player Settings**:
   i. You can go in here and play with company naming, icons etc.

f. Once everything is setup the way you want Hit **Build**
   i. Save the package to where you'd like
   ii. Note – it'll create an .exe file and a Data folder with the necessary files needed to run the .exe. You won't be able to run the .exe without the Data folder