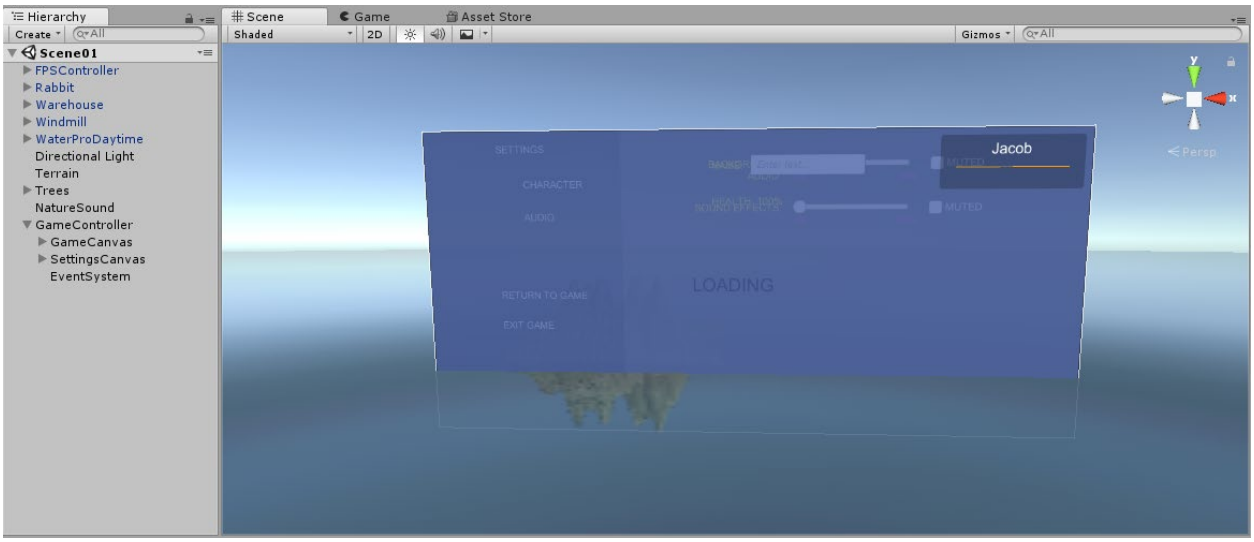
UI ELEMENTS FOR HUD AND GAME SETTINGS

This is a lengthy challenge, but one that you will need to complete before moving on to the next unit. It involves adding UI elements and functionality to the *GameSettings* canvas, which I renamed to SettingsCanvas.

You should download all of the scripts for this challenge at
burkecs.org/courses/3DGameDev/unit7/challenges/Scripts.zip

To ensure the scripts work correctly, you will need to organize and name GameObjects to match my scene.

The screenshot below shows the two canvases with all the panels overlaid. Notice both the *GameCanvas*, which is shown during gameplay, and the *SettingsCanvas*, which is used while the game is paused, are both children of the *GameController* GameObject.



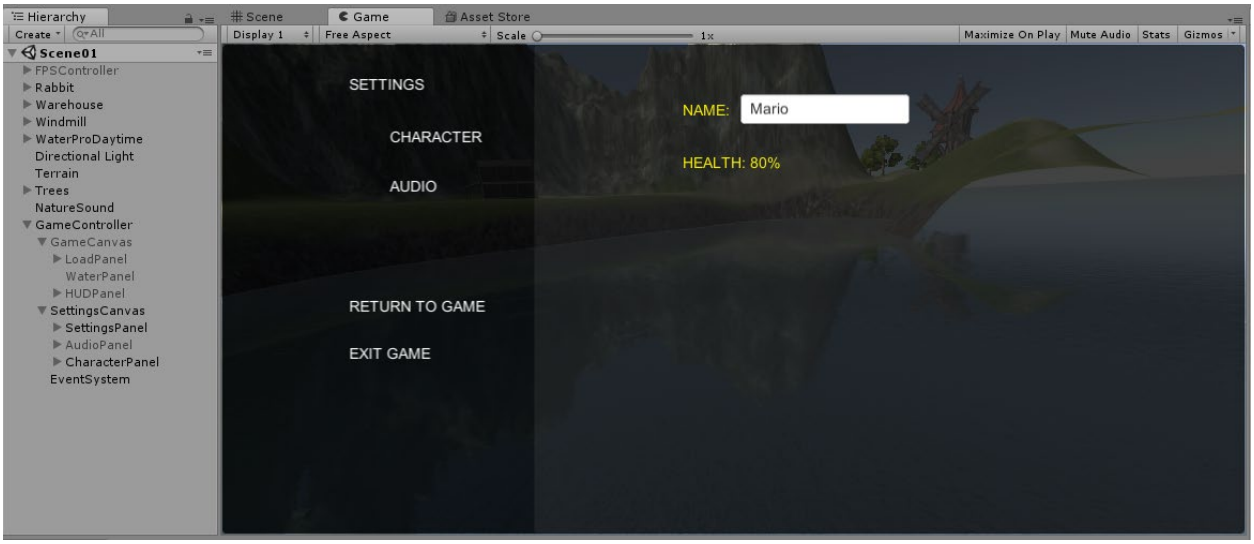
When you begin the game, the entire *SettingsCanvas* is disabled (grayed out in the hierarchy). I moved the *LoadPanel* into the *GameCanvas* because later we are going to set the entire *GameController* to be persistent across scenes. This will relieve us of having to make a copy of it in each scene, and then have several copies to edit every time a change is needed. Also, by keeping the *GameController* persistent across screens, we will be able to store data that persist across scenes. For example, the name and health of the character, which is displayed below in the *HUDPanel*, is a value that we would want to carry over when we load the next scene.

The SettingsCanvas, like any Canvas UI element, will always stretch horizontally and vertically to fill the play window. You have little control over this. However, you do have control over the sizing and positioning of the Panels added to the Canvas.

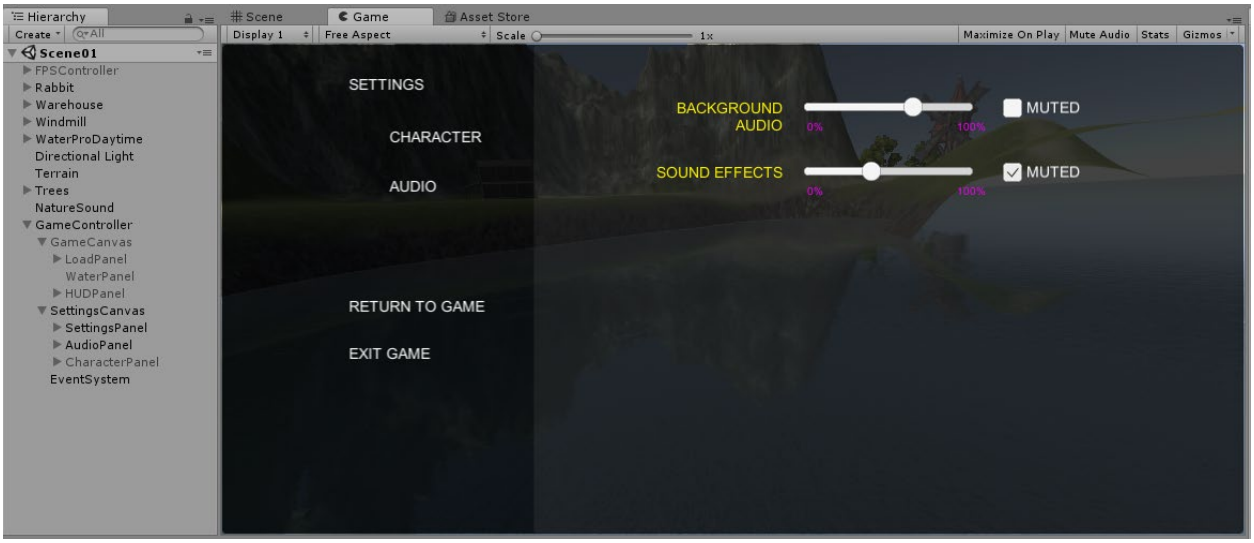
When created the panels of the SettingsCanvas, I did not leave the default settings that result in the panels stretching horizontally and vertically to fill the Canvas.

Instead, I clicked on the icon of the RectTransform Component and modified each of the panels as I wanted. I learned to do this by watching a 2014 demo by Unity.
www.youtube.com/watch?time_continue=152&v=0L-Y8dPJHDM&t=2m30s
The relevant bit is from 2m30s to about 6m30s.

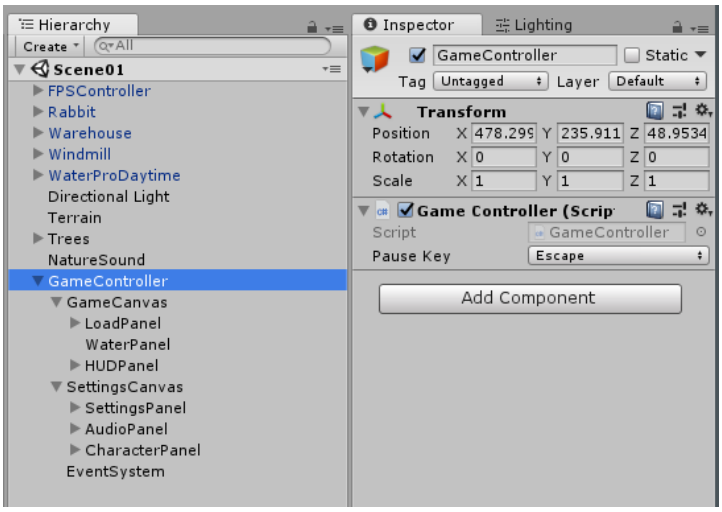
While playing the game, pressing the Escape key will pause the game and bring up the settings. You can see in the hierarchy that the *GameCanvas* and *FPSController* are now disabled (grayed out) and the *SettingsCanvas* has been enabled and two of its three panels are also enabled. The *SettingsPanel* is shown on the left of the game window and the *CharacterPanel* is shown on the right.



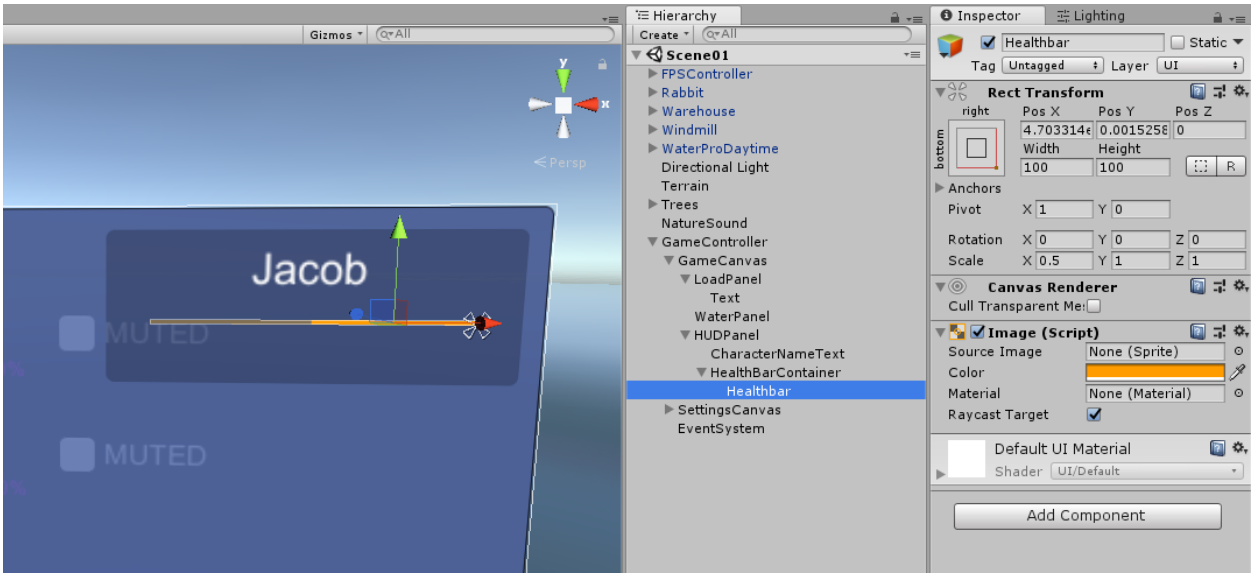
Clicking on the AUDIO button switches from the *CharacterPanel* to the *AudioPanel*.



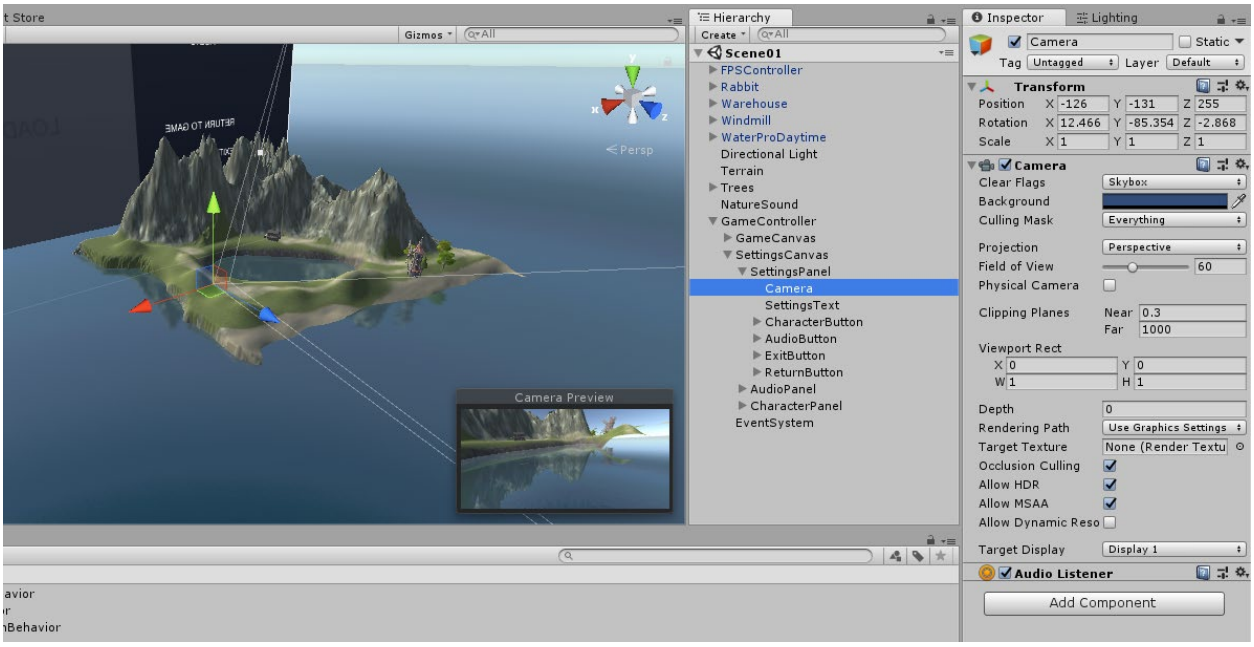
The *GameController* GameObject is not a Canvas, Panel, or any other UI element. It was created as an Empty GameObject to hold all GameObjects that we want to remain when another scene is loaded. The only Component added to the *GameController* is a script, also named *GameController*. This script was modified from the previous *GameSettings* script. I renamed the script because it quickly evolved to have a broader use beyond just the game settings. In fact, *GameController* will store any data that is used in more than one scene.



The *HealthBarContainer* and the *Healthbar* GameObjects are both Image UI elements. Notice the X of the scale is used for the health. In the screenshot below, the X scale is set to 0.5, so the *Healthbar* in the game window is taking up 50% of the *HealthBarContainer*. Of course, this can be changed during gameplay when damage is dealt to the character.

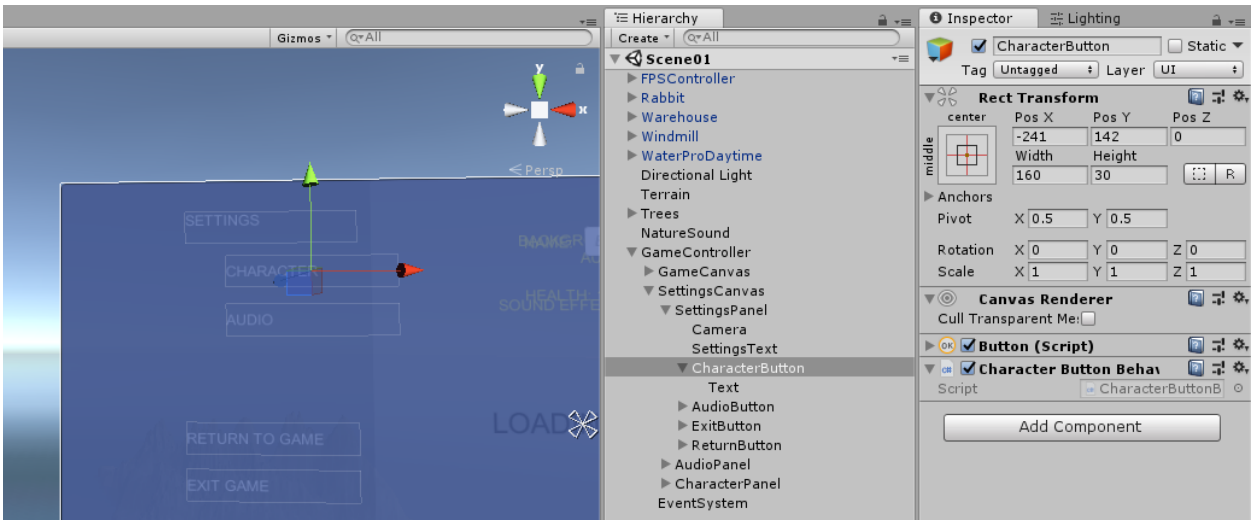


Notice a *Camera* has been added as a child of the *SettingsPanel*. This is needed because we are turning off the FPSController, which contains our in-game camera, when the game is paused. By turning off the FPSController, we ensure all functionality of that player is also disabled.

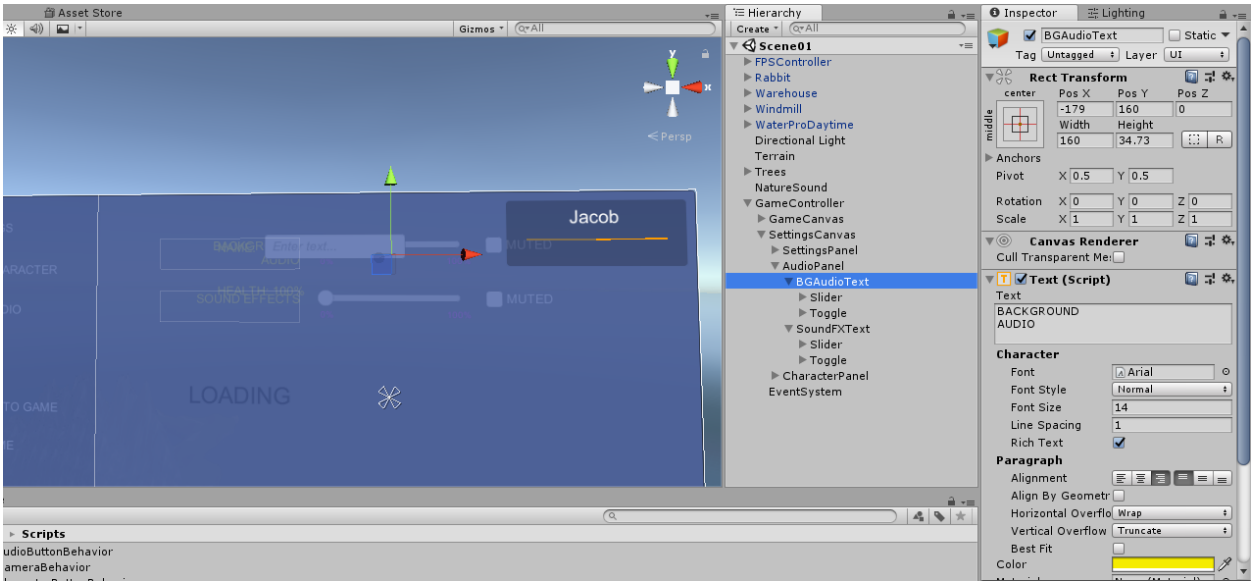


There are other advantages of using a different camera while the SettingsPanel is enabled. First, you can set up the camera to point at what you want to see while the game is paused. This is especially useful if your *SettingsPanel* is partially transparent. The other advantage, and the original reason I added a second camera, was that to solve a bug I was having with losing the cursor when any UI element was clicked on. Code on the FPSController was interfering with the cursor, so I needed to disable it while the game was paused.

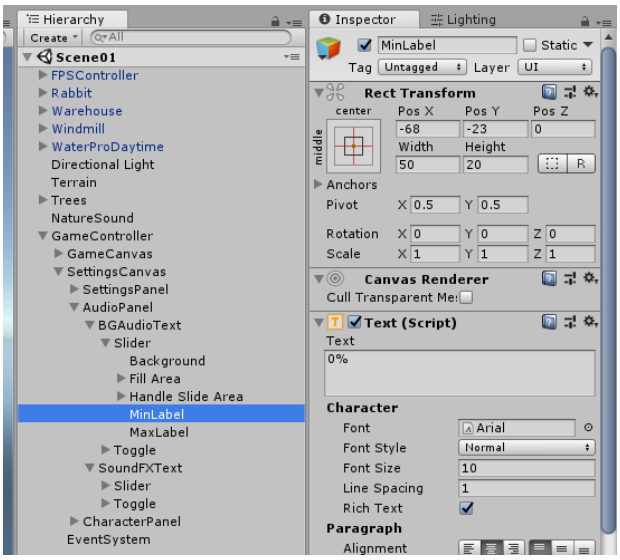
You can see in the game window that most of the UI elements on the *SettingsPanel* appear to just be Text elements. Actually though, this is only true of *SettingsText*, which is simply used as a label that you cannot interact with. To ensure the other elements were clickable, they were actually added as Buttons. The reason they do not appear as a button is because the Image component of the Button object was removed. However, because they have a Button script as a component, they can still behave as a button.



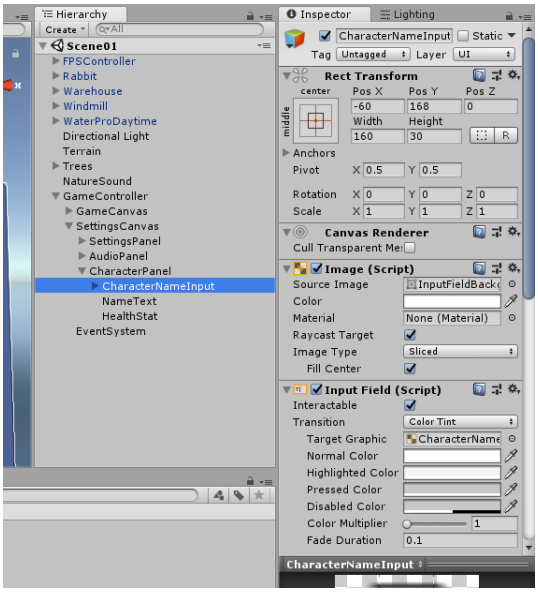
In the AudioPanel, you see I created *BGAudioText* and *SoundFXText* to use as labels for the audio categories. For each of those Text UI elements I added a Slider UI element and a Toggle UI element. The Slider is used to change the volume between 0 and 1, and the Toggle displays a checkbox that we use to mute or unmute the audio.



I added Text UI elements named MinLabel and MaxLabel as children of the Sliders to show the 0% and 100% at either end.



As a child of the *CharacterPanel*, I added an Input Field UI Element named *CharacterNameInput*.



Download the scripts provided and move them into your game project's **Scripts** folder.

Attach CharacerButtonBehavior.cs to the *CharacterButton* of the *SettingsPanel*.

Attach ExitButtonBehavior.cs to the *ExitButton* of the *SettingsPanel*.

Attach ReturnButtonBehavior.cs to the *ReturnButton* of the *SettingsPanel*.

Attach StartButtonBehavior.cs to the *StartButton* of the *SettingsPanel*.

Attach AudioButtonBehavior.cs to the *AudioButton* of the *SettingsPanel*.

Attach GameController.cs to the *GameController*.

Attach VolumeModifier.cs to every GameObject in your scene that has an AudioSource Component and set the public property type to SoundFX or BGAudio using the dropdown in the Inspector. In my game, I attached this script to:

- The NatureSound with the type set to BGAudio
- The Rabbit with the type set to SoundFX
- The FPSController with the type set to SoundFX

Note, we are now setting the volume of the FPSController based on player defined game settings. This means the code we previously added to the FirstPersonController.cs script needs to be removed.

After removing the code, you should look at the FPSController's AudioSource in the inspector and use the public volume property set the volume at a level that is balanced with all the other sound effects in the game. The player will have control over the volume of all audio of type SoundFX and of type BGAudio, but they will not have control over individual audio sources.

Most likely you will make at least some errors as you set this all up, resulting in your code not compiling or the functionality not behaving as expected. This is a great opportunity for debugging. As the scripts have all been tested, no modification to the code should be needed as long as the GameObjects name and organization are correct.

I may have forgotten a step or two, so this document will be updated as I am made aware of any issues.

Happy coding!