Assignment 6: VR Interaction and Animation

For this assignment we are going to have you trigger an animation using VR input. We have given you a model of The Tyrant to use to play animations. This requires assignment requires 3 parts. First you will set up the Animator Controller for the Tyrant. Next you will add sound sources. Last, you will write code to play your animations and sound. We are giving you a little less direction on this assignment so please don't hesitate to come to office hours and ask questions.

Part 1: Setting Up the Animator Controller

For this part of the assignment we will set up the animator controller for The Tyrant. What we will require is:

- You should add The Tyrant model somewhere into your scene
- You must create an Animatior Controller for The Tyrant model and assign it to the Animator component on The Tyrant in your scene
- In the animation window, set the default animation clip to **tyrant_idle,** and make sure that this clip loops
- Add the **tyrant_point** and **tyrant_make_fist** animations in the graph.
- Make an int parameter in the animator state machine called pose. The values of pose will correspond to animation clips in the state machine (0 = idle, 1 = point, 2 = make_fist)
- Make transitions between all of these clips. These clips should go both ways and use the **pose** parameter in their conditions
- For each transition, uncheck **Has Exit Time**

You can use the Butterfly Tutorial on the website for reference.

Part 2: Setting Up Sound

You will set up both a spatial and an ambient sound. We have provided 2 sound clips for you to use, but feel free to find your own if you like. The Requirements are as follows:

- Create an ambient sound source that will play on awake and loop
- Create a Spatialized sound source attached to The Tyrant that **does not** play on awake or loop. Make sure the Spatial Blend value is all the way to the 3D side
- Make sure to go into the Project Settings under Audio, and enable the Oculus Spatializer

Please use the Audio Tutorial for reference. To test the sound, we will have you go into the **Oculus->VR->Prefabs Folder** in your project, and put it into your scene where your main camera is. You should then disable your main camera to test it.

Part 3: Interaction Scripting

We have written a Gaze-Based Raycaster and object selection system for you to use. You may look at our default example and extend this to allow us to trigger animation. In order to use the Raycaster, go into the hierarchy and find the **OVRCameraRig.** Expand the OVR camera rig until you find the **Center Eye Anchor**. Add the **VRPointer** script to the Center Eye Anchor. You can look test this in the **Pointer Test** Scene Provided. You can use the trigger to select an object.

You can also test without a headset by adding a **PointerSelector** script to your main camera. This should work the exact same, but with your mouse instead

Let's take a brief look at the code we have for you:

- VRPointer – This component allows us to select anything that has a Selectable component attached to it. Use the trigger to select an object when you are looking at it. You can deselect an object by pulling the trigger while looking into space
- Selectable – This is an abstract class that outlines the methods needed for a selectable object. Let's go over the methods included
    - OnSelect() – This method is where you should include code on what will happen when the object is selected by our pointer
    - OnHover() – This method defines what happens when you "hover" over the object you want to select. By "hover" we mean your cursor (or gaze in this case) passes over an object you may want to select, but you haven't selected it yet. Similar to how you may "mouse-over" a button on a website, you will usually want to give some sort of indication that you are hovering over, such as changing color or playing a sound.
    - OnDeselect() – This method is what you should use to deselect an object. This should undo any changes that you have made on selection and/or hover and leave the object in its untouched state

    Because this is abstract, you cannot attach it to a game object as a component. You must instead write another class that inherits from it and define the behavior in the abstract methods.

You can see an example of this in the SelectableTile Class. SelectableTile simply changes the color of the tile when it is hovered over, selected, and deselected.

```
using UnityEngine;

public class SelectableTile : Selectable
{
    [SerializeField]
    private Material def, hover, selected;
    private bool isSelected;

    public override GameObject OnSelect()
    {
        Debug.Log("Selected");
        isSelected = true;
        gameObject.GetComponent<MeshRenderer>().material = selected;
        return this.gameObject;
    }

    public override GameObject OnHover()
    {
        if (gameObject.GetComponent<MeshRenderer>().material != hover && !isSelected)
        {
            gameObject.GetComponent<MeshRenderer>().material = hover;
        }
        return this.gameObject;
    }

    public override void OnDeselect()
    {
        gameObject.GetComponent<MeshRenderer>().material = def;
        isSelected = false;
    }
}
```

- First thing to note is at the top of the class. Any new class must inherit from Selectable using the YourNewSelectable : Selectable syntax
- We have a Boolean isSelected to keep track of our internal state
- We have three Materials that we will use to indicate our three states: Deselected, Hover, and Selected
- In our OnSelect method, we use the set selected to true, and get the objects MeshRenderer component and set its material to our Selected material. Then we return our game object so the selector can access its methods
- In our OnDeselect method, we simply set isSelected back to false, and again, we get the object's MeshRenderer component and set its material to our def (default) material
- Our OnHover Method is a little more complicated.
    - First we will check to see if our material is our Hover material. We do this to make sure that we don't set our material to hover if our object already is set to the hover material, because that would be unnecessary.
    - Next, we check to make sure that the object is not selected. We do this because in order to select an object we have to hover over it, so once we select it, our cursor may still be hovering over it. Because we have just selected it, we don't want to give it the hover material because we are now in the selected state.

For this assignment, You will fill in the code for our SelectableTyrantPlayer class. We'll give you some starter code, but you must figure out how to play the animations. Some tips:

- To change the materials, look at how it is done in the selectable tile example
- Look at the animator's SetInt(name, value) function from the butterfly tutorial to see how to set the int parameter
- Look at the AudioSource.Play() function to play the audio source

In order for this to work we have provided a Tyrant Controller prefab that sets most of this up for you. The additional setup you must do is to add this prefab to the scene and in the Animator component, drag and drop The Tyrant into it. You also need to add the tyrant's audio source in the **sound** slot.

You will need to Download our Asset Package from the website. This assignment will be due Wednesday 11/20 @ 9 PM. Please Tag this assignment as midterm_submission. To submit, push to github.