**OVERVIEW**

In this lab you will learn how to programmatically map and use the left- and right-hand HMD Odyssey controllers. You’ll be required to perform some basic operations such as grabbing, and dropping, and interfacing with a button via your controllers. You’ll also be required to create, in addition to a project runnable via Unity, a standalone and installable project build.

**IN-CLASS ASSIGNMENT**

* **You must use the PCs in Little 231 for this assignment**
* **This is an in-class assignment that is due by the end of the lab.**
* Setup the Odyssey with two controllers via the HMD Portal
* Launch and run the provided OdysseyInput Unity project, and show the Professor that you can interact with all the controller controls.
  + This project along the information provided in the following link(s) will provide you with enough information to programmatically access, control and track each controller.
    - Unity XR Input documentation: <https://docs.unity3d.com/Manual/xr_input.html>
* Create a UWP Unity Project for Windows Mixed Reality, and import the Lab2 Package. Use the scene provided in the Package.

**Note: For this lab your headset (camera) and two controllers need to be structured so as to be children of an Empty Game Object denoted as Player**

* Setup the left controller.
  + Create a controller prefab consisting of an appropriately sized cylinder with a material applied to it, and a visible location that grabbed objects will attach themselves to.
    - Structure the preFab as follows:  
       ***Controller prefab (Empty Game Object)  
       |--- Cylinder Model  
       |--- Visible Hold Objects Point***
  + Add a script to your controller preFab. Your script should
    - Real-time track, position and orient the controller
      * If the controller clips too early as it gets closer to your face fix it, and make things look more realistic. Clipping is not a problem in itself, as long as the clipping doesn’t look unrealistic. For example, a controller shouldn’t clip one foot away from your face.
      * Output the Vector2 horizontal and vertical joystick values to the console window and demonstrate to the Professor that you have this working properly

**LAB REQUIREMENTS**

**Setup and add the right controller to your Player**

1. Continuing on with the Class Assignment above add your controller prefab tracked, positioned, and oriented correctly, to your project and map it to the right-hand controller
   * Both controllers should now be part of your project.
   * Your prefab script should correctly
     + Real-time track, orient and position the right-hand controller  
         
       **Implement Grab Mechanic**
2. In your scene you’ll see an object on top of a pedestal. The grip button can be used to grab objects at the controller’s “grab point”.

* When the grip button is released, assuming an object is currently being grabbed, the object should be dropped.
* Restriction - only one controller can grab one object at any given time.
* Requirement – The object that is being grabbed should know it’s being grabbed, and also which controller is grabbing it.
* The object while grabbed should mirror the orientation and position of the controller that is holding it at the “grab point” on the controller.
* Helpful Tutorial with some usable ideas:  
  // HOW TO PICK UP AND PLACE / THROW OBJECTS WITH C# - Mini Unity   
  // Tutorial - EASY & FUN

<https://www.youtube.com/watch?v=IEV64CLZra8>

**Button Interaction**

1. Each controller should generate Game view, visible raycasts that are always active.
2. If the raycast falls on the Respawn button, and the trigger button is pressed:
   * Do nothing if the object is already on the pedestal.
   * Reposition the object onto the pedestal if it’s not being grabbed, and its currently not on the pedestal.

IMPORTANT: If the object falls off any platform it will automatically be destroyed. In that case, you will have the option to either respawn the object back onto the center platform automatically or only do so after a respawn button press. Key to all this is that only one grabbable object is allowed to be in the scene at any given time.

1. Implement a way to kill the game for both a standalone project or when in Unity via the controller. How to exit the application instructions need to be visible in the game window.

**Extra Credit**

1. Change the color\material of the respawn button when a raycast intersects with the respawn button
2. Implement a throw mechanic and try to hit the platforms not part of the main platform.

Here’s some helpful tutorials that have some good ideas that can be adapted to the HMD Odyssey in regards to throwing:

// Vive Minute Tutorials in Unity - Throwing Objects – 6:31 minutes

<https://www.youtube.com/watch?v=ltxoFzXcGQA>

// Unity link pertaining to Odyssey controller linear and angular velocities

<https://docs.unity3d.com/ScriptReference/XR.CommonUsages.html>

1. Optionally add a visible score field and turn this into a timed throw game.
   * Consider displaying time left, hits and misses, and a Game Over button

**SUBMISSION**

Submit to Canvas a zip file with the following

* Remove the old Lab01 standalone (installable) build folder
  + The size of your project will quickly get unwieldly if you don’t remove this folder.
* Structure your submission folder as follows  
   ***You turn in folder  
   |--- Standalone, installable Lab02 build (Only latest version)  
   |--- Unity project ready to build and run from within Unity***
* Remove files that can be rebuilt from the Unity project to reduce the size of the project folder
* Name you zipfile like this <YourName>\_AG231\_Lab2