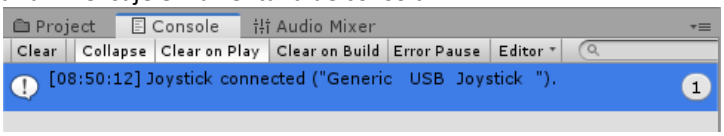
INTERFACES AVANZADAS

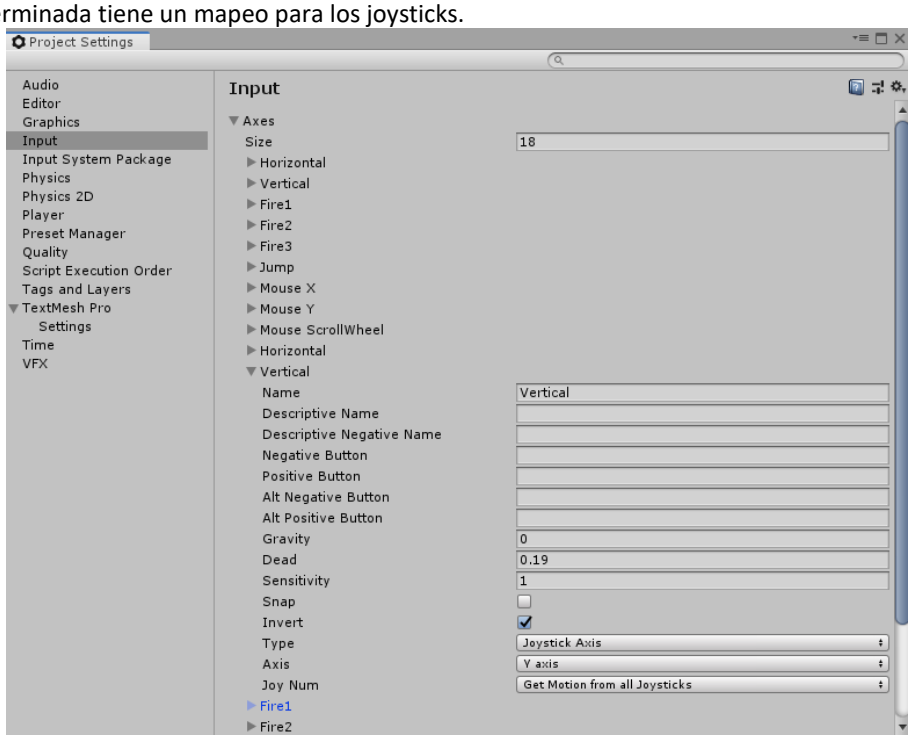
Interfaces avanzadas: Gamepad o mandos

A) Control de la paleta usando un Gamepad

1. En Unity de forma automática ya se puede usar un GamePad como un mando de Xbox o PlayStation o cualquier otro compatible con la computadora. Si lo conectamos a la computadora Unity lo debe reconocer y enviar un mensaje en la ventana de consola.



2. Esto es gracias a que el Input Manager (Menú Edit > Player Settings > Input) de forma predeterminada tiene un mapeo para los joysticks.



3. Al usar la forma *Input.GetAxis("Vertical")* en el script PaddleInput.cs Unity puede detectar de forma automática todos los Input Axes de nombre "Vertical", donde el primero es para el teclado y el segundo para cualquier GamePad joystick.

Interfaces avanzadas: Acelerómetro / Giroscopio

A) Control de la paleta usando el giroscopio del smartphone

1. Según Wikipedia el acelerómetro es un sensor que sirve para medir la aceleración de movimiento de un dispositivo que en este caso es el smartphone, de la misma forma al igual que el giroscopio, permite saber la posición y rotación exacta del dispositivo.
2. Tenemos que tener en cuenta los ejes del acelerómetro, en nuestro caso usaremos la vista Landscape o Horizontal, así que tenemos que tener en cuenta el eje X.

3. Finalmente, para poder controlar nuestra paleta de pong usando el acelerómetro vamos a editar el script ”PaddlInput.cs” de la siguiente forma.

4. Unity tiene una colección de directivas #define que pueden controlar partes del código que deben o no ser compiladas dependiendo de la plataforma de despliegue. De la misma forma se puede usar la directiva para dispositivos IOS
   Más información: https://docs.unity3d.com/Manual//PlatformDependentCompilation.html .
5. En C# podemos definir nuestras propias directivas usando #define SOME_NAME_X. Esto puede servir para controlar la compilación de mensajes de debug en la consola sin tener que borrarlos o comentarlos:

```
using UnityEngine;

[RequireComponent(typeof(Paddle))]
public class PaddleInput : MonoBehaviour
{
    private Paddle _paddle;

    private void Awake()
    {
        _paddle = GetComponent<Paddle>();
    }

    void Update()
    {
// Verifica que la plataforma de despliegue no sea Android, entonces ejecuta la lógica
normal
#if !UNITY_ANDROID
        var v = Input.GetAxis("Vertical");

        _paddle.Move(Vector3.forward * v);
// En el caso de Android usaremos el acelerómetro
#else
        var accel = Input.acceleration;
        _paddle.Move(Vector3.forward * accel.x);
#endif
    }
}
```

```
// Definir una directiva
#define DEBUG_PaddleInput
using UnityEngine;

[RequireComponent(typeof(Paddle))]
public class PaddleInput : MonoBehaviour
{
    private Paddle _paddle;

    private void Awake()
    {
        _paddle = GetComponent<Paddle>();
// Verificar si la directiva fue definida
#if DEBUG_PaddleInput
        Debug.Log("Awake(): Passing for Awake!", gameObject);
#endif
    }
```

6. Si comentamos la directiva el código dentro de #if no compilaría más:

```
// #define DEBUG_PaddleInput
```

Interfaces avanzadas: Leap Motion

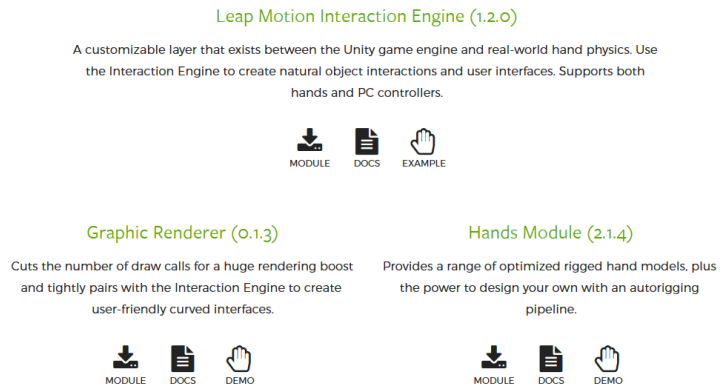
A) Instalación del plugin para Unity de Leap Motion

Leap Motion es un dispositivo de entrada compuesto por una cámara infraroja que puede detectar la anatomía de las manos y reproducirlas en modelos 3D dentro del motor de juego. Cabe destacar que el Leap Motion se puede utilizar con o sin un casco de realidad virtual o HMD.
https://www.leapmotion.com/where-to-buy/

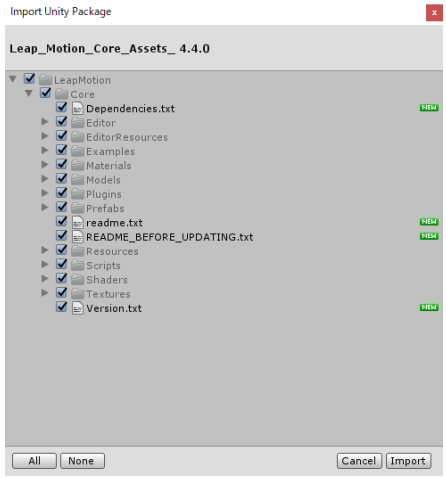
1. Para poder utilizarlo necesitamos instalar su software para Windows o MAC: https://developer.leapmotion.com/get-started,

2. y adicionalmente el SDK para Unity: https://developer.leapmotion.com/unity#5436356

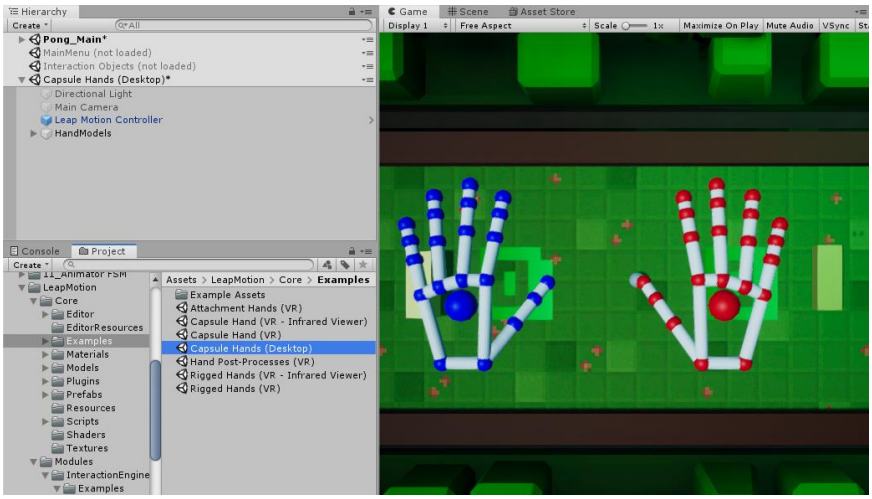
3. Adicionalmente Leap Motion cuenta con otros paquetes encargados de la interacción, gráficos y manos 3D. Cada paquete cuenta con ejemplos muy completos.



4. Una vez descargados lo importamos en el proyecto de Unity.

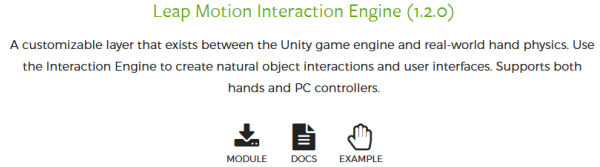


5. Agregamos la escena dentro de la carpeta Leap Motion > Core > Examples > Capsule Hands (Desktop).

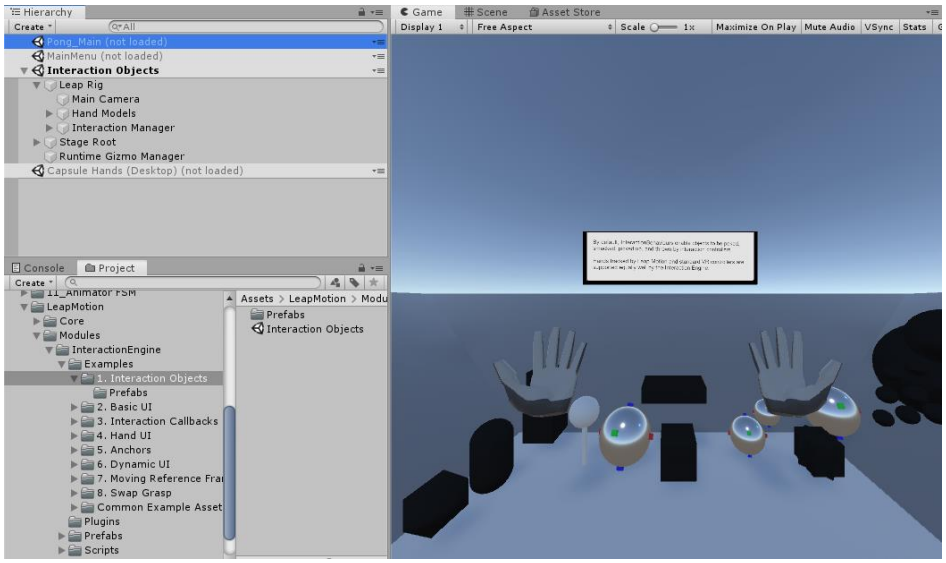


6. Al ejecutar usando el botón Play, aparecerían los modelos de las manos reconocidas por el Leap Motion.

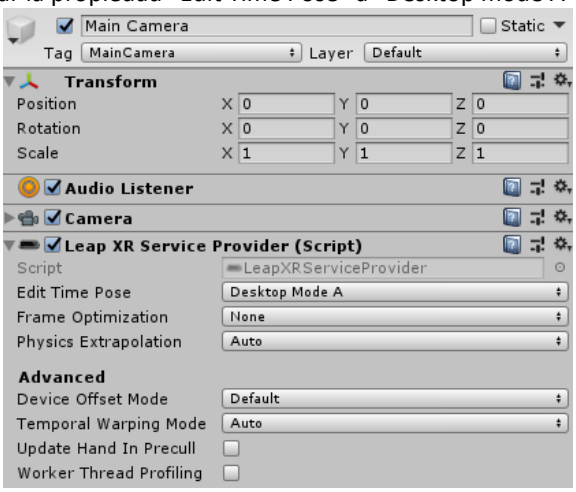
7. Para permitir la interacción con objetos necesitamos importar el Módulo de Motor de Interacción:



8. Una vez importado en el proyecto abrimos el ejemplo básico en la carpeta: Leap Motion > Modules > InteractionEngine > Examples > Interaction Objects



9. Para que funcione sin un casco de realidad virtual necesitamos seleccionar el GameObject “MainCamera” y cambiar la propiedad “Edit Time Pose” a “Desktop mode A”.



10. Hay varios ejemplos más en el módulo de interacción, como controles UI, callbacks, etc. Se recomienda probarlos para tener mayor alcance de todo lo que se podría hacer.

Sub tema 1.6 Interfaces avanzadas: Touch controls

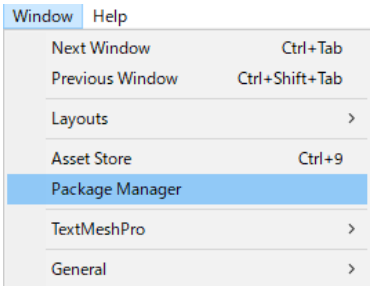
A) Instalación del SDK de Oculus

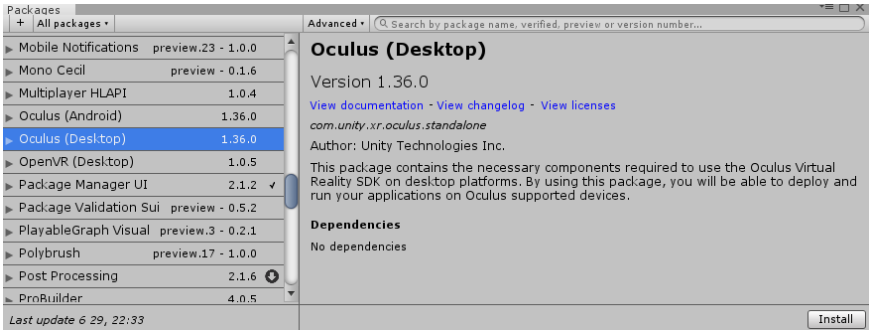
Oculus es un HMD (Head Mounted Display) o Visualizador Casco de Realidad Virtual, que incluye un par de controles touch, uno para cada mano.

Veamos como instalar el SDK para poder utilizarlo incluyendo el mapeo de la posición, rotación y entrada de los controles, en Unity.

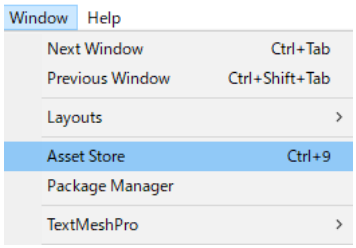
1. Para instalar el Oculus necesitamos hacerlo en 2 pasos, el primero es instalar el paquete de Oculus desde el Package Manager de Unity. Menú Window > Package Manager. E instalar el

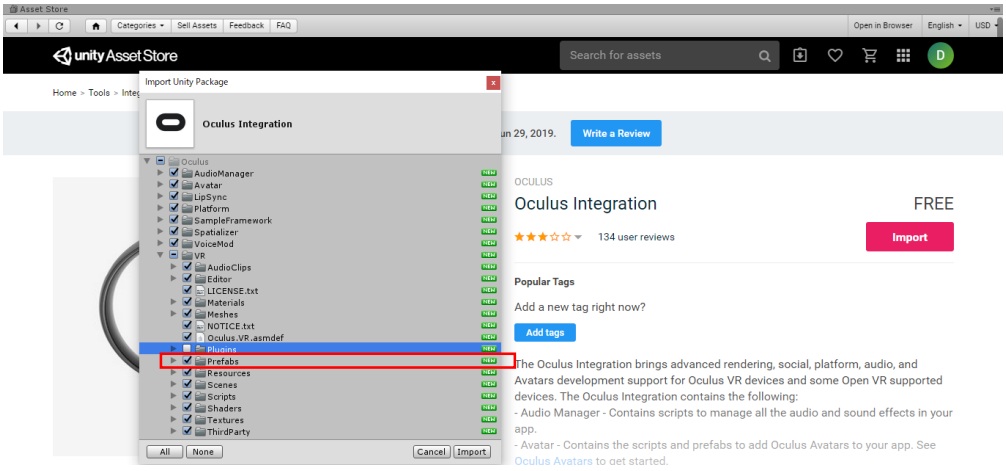
Oculus (Desktop), esta versión es compatible con los controles touch para las manos.





2. El segundo es instalar el paquete disponible del asset store: Window > Asset Store. Y Buscar por “Oculus Integration”



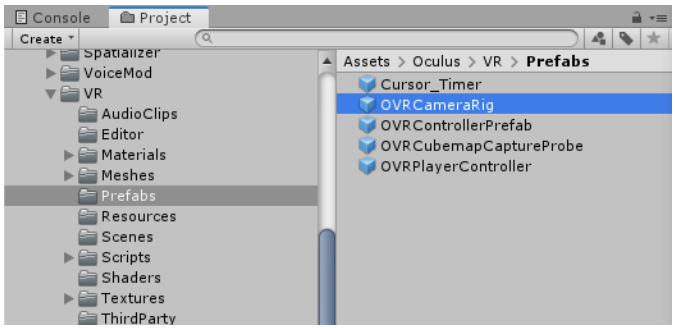


Nota: No se debe importar la carpeta “Plugins” dentro de la carpeta VR, ya que estos archivos .dll están incluidos en el paquete instalado desde el Package Manager.

3. Cabe destacar que además del paquete básico (carpeta VR), también integra varios otros paquetes como el Manejador de Audio, Avatar, Sincronización de labios, Spatializer (Algoritmo para control de Audio 3D), entre otros. Se recomienda leer su documentación para conocer mejor su aplicación: https://developer.oculus.com/documentation/ y tener un mayor alcance

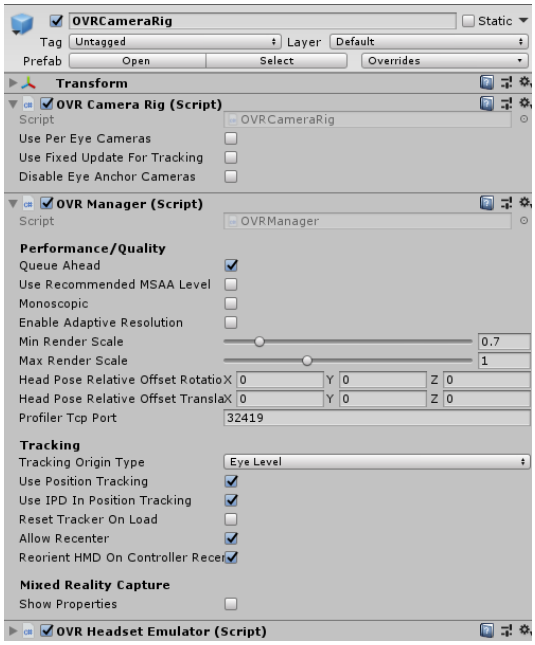
de cómo aprovechar esos paquetes.

4. Necesitamos un “CameraRig”. A diferencia de juegos tradicionales, una aplicación de realidad virtual generalmente está en vista en primera persona. El casco y los controles tienen una posición y rotación independientes de cada uno, pero los tres objetos necesitan tener un mismo GameObject padre que permita cambiarlos de posición en grupo. Este paquete cuenta con un prefabricado de un CameraRig listo para usarse: Carpeta Oculus > VR > Prefabs > OVRCameraRig



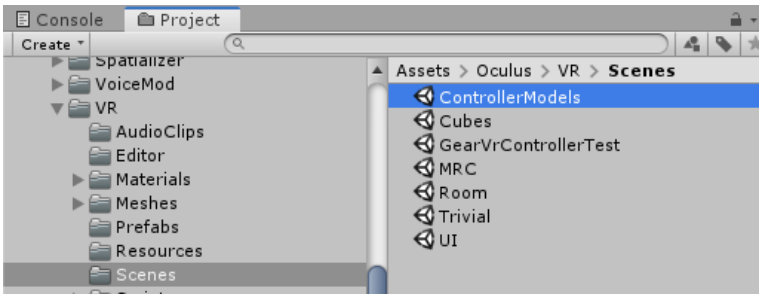
5. Si lo ponemos en la escena esto es lo que contiene:

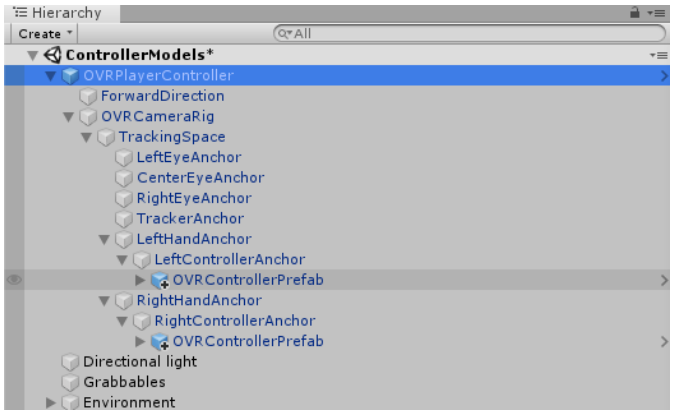
El script “OVRCameraRig” va a buscar entre sus GameObjects hijos los Transforms para el Casco o “Eye”, “LeftHand” y “RightHand” y actualizará su posición y rotación de forma automática. Basta con poner algunos objetos (cubos, modelos de manos 3D) como hijos dentro de la jerarquía para LeftHand o RightHand para mostrar un avatar simple para el jugador.



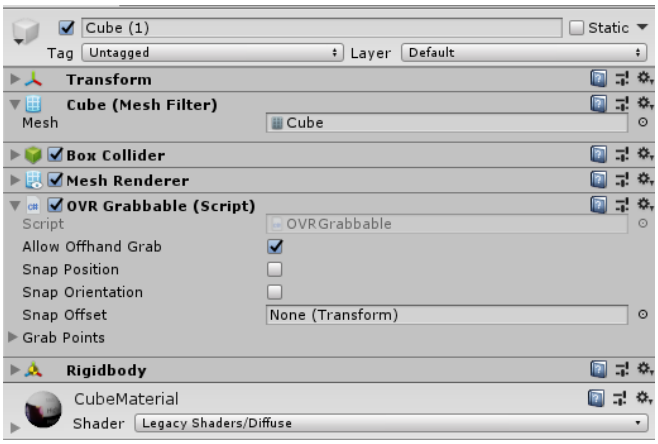
6. Al igual que Leap Motion, este paquete también contiene varios ejemplos de diversos casos de uso. Veamos la escena “ControllerModels” dentro de la carpeta: Oculus > VR > Scenes. Este

ejemplo contiene el OVRCameraRig y dentro de los Anchors para las manos, prefabricados de los modelos de los controles. Este ejemplo además permite la interacción de agarrar y lanzar objetos presionando los botones de puño (Grip) de los controles touch. .

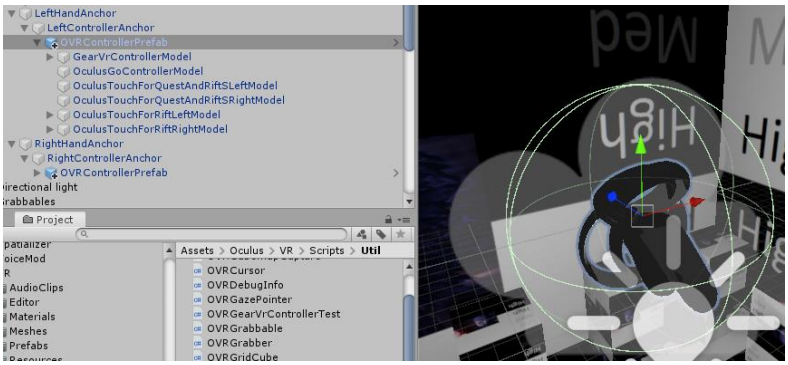


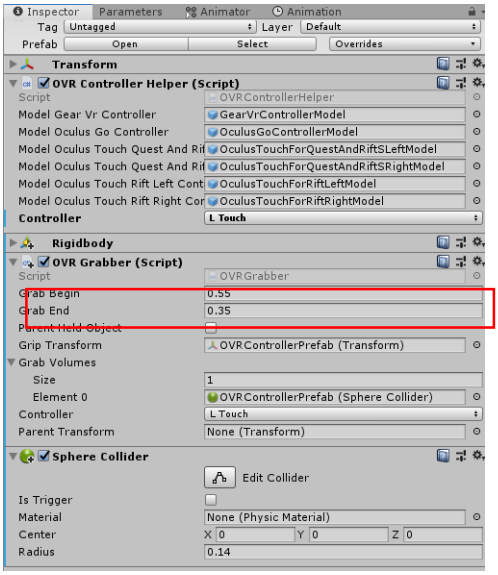


7. Cada objeto interactivo necesita el componente “OVRGrabbable”



8. Y cada objeto origen de interacción (manos) necesita el componente OVRGrabber

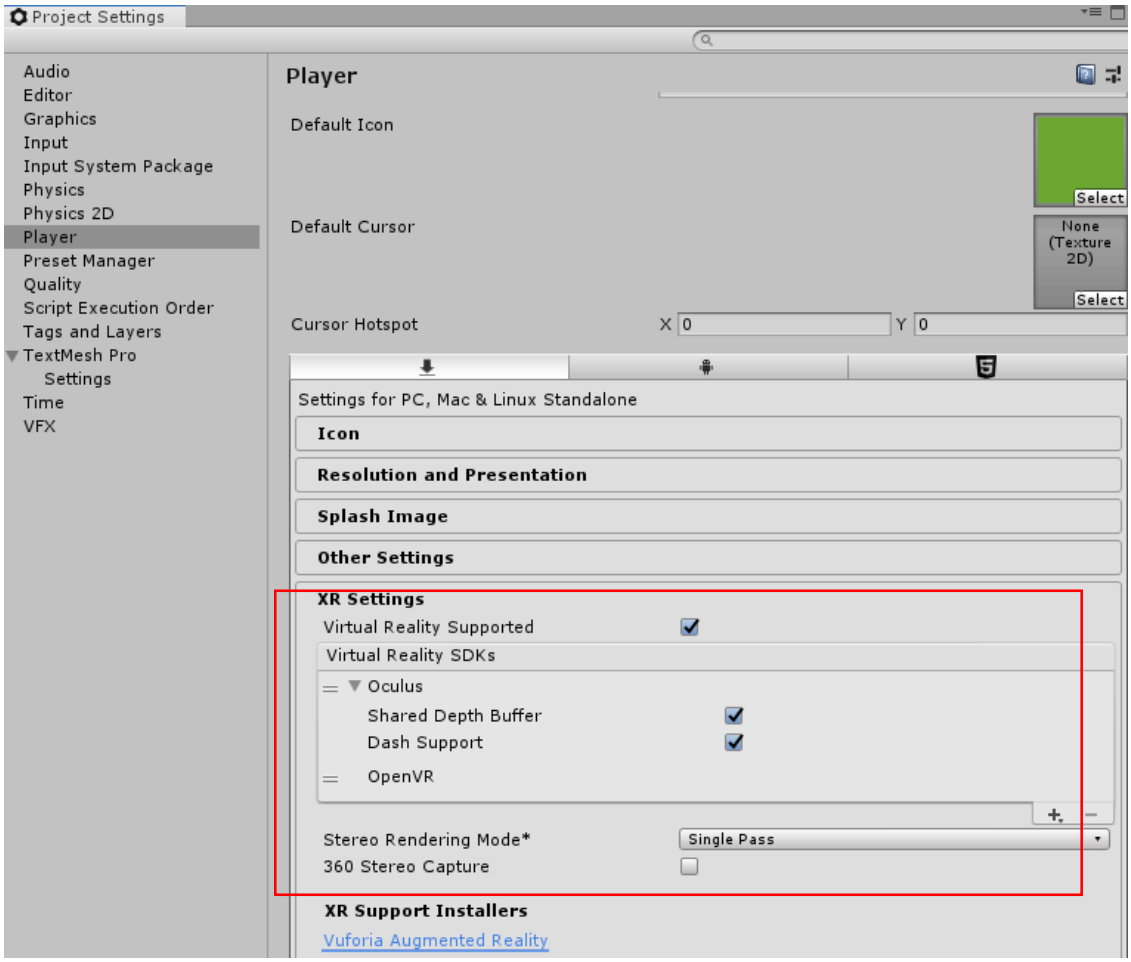




Nota: Las propiedades GrabBegin y GrabEnd definen en que punto se considera el comienzo o fin de agarre. El botón de puño (grip) al igual que los demás botones pueden enviar un Axis, un valor entre 0 y 1.

9. Finalmente, para que funcione el HMD, necesitamos verificar que Virtual Reality este permitido en Player Setting: Menú Edit > Project Settings > Player

   Nota: Cuando es la primera vez que se prueba en el Oculus después de activar está opción, ocasionalmente se necesita que Unity se cierre y se abre para que funcione correctamente.



10. El mapeo completo de todos los botones los podemos encontrar en la siguiente dirección: https://docs.unity3d.com/Manual/OculusControllers.html. Adicionalmente, el paquete de Oculus > VR, tiene un script muy útil para acceder a estos botones con enumeraciones (estructura de datos de tipo enum en C#): https://developer.oculus.com/documentation/unity/latest/concepts/unity-ovrinput/