PRÁCTICA GUIADA
Desarrollo de aplicaciones de realidad virtual con HMDs

INTRODUCCIÓN

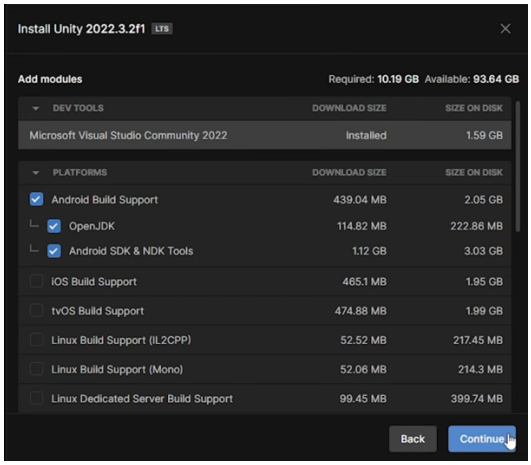
En el dinámico campo de la Realidad Virtual (RV), los Kits de Desarrollo de Software (SDKs) están en constante transformación, evolucionando a un ritmo impresionante. Dada esta rápida evolución, es un desafío abrumador abarcar completamente su complejidad en una sola práctica. Además, existe la posibilidad de que tanto las implementaciones actuales como los principios subyacentes sufran cambios significativos en un periodo relativamente corto. Enfocándonos en esta realidad, nuestra práctica se concentrará en la comprensión profunda de los principios fundamentales, aquellos que se espera permanezcan estables a lo largo del tiempo, utilizando el XR Interaction Toolkit, el SDK más destacado y recomendado por Unity para el desarrollo en realidad extendida. Este paquete ofrece una API versátil, que no depende de un hardware específico, brindando incluso la posibilidad de simular experiencias de RV con solo usar teclado y ratón.

En esta actividad será evaluada mediante un cuestionario de opción múltiple.

CREAR NUEVO PROYECTO Y CARGAR ESCENA DE EJEMPLO EN UNITY

Para esta práctica se recomienda la versión de Unity 3D - 2021.3.1f1 LTS o superior.

Cuando instales Unity, es crucial seleccionar ciertas opciones específicas dentro de la sección de soporte de construcción (Build Support) para asegurar la compatibilidad y el funcionamiento óptimo con dispositivos Android. Dentro de las opciones disponibles, asegúrate de incluir: **Android Build Support**. Esta opción habilita la capacidad de Unity para crear proyectos que se pueden ejecutar en dispositivos Android. Al seleccionar esta opción se seleccionan tambien **OpenJDK** (Unity incluye su propia versión del OpenJDK, que es necesario para el desarrollo de aplicaciones Android para evitar problemas de compatibilidad con Java) y **Android SDK & NDK Tools** (herramientas necesarias para la creación y optimización de tu proyecto para Android).



Si no has instalado las herramientas necesarias para el desarrollo en Android con Unity, sigue estos pasos:

1. Abre el Unity Hub.
2. Ve a "Installs" en el menú lateral para ver las versiones de Unity que tienes instaladas.
3. Selecciona la versión de Unity que deseas configurar y haz clic en los tres puntos al lado para abrir el menú de opciones. Elige "Add Modules".
4. En la lista de módulos disponibles, busca y selecciona "Android Build Support". Dentro de esta opción, asegúrate de marcar también "OpenJDK", "Android SDK & NDK Tools".
5. Haz clic en "Install" o "Next" para comenzar la instalación de estos componentes.

CREA UN NUEVO PROYECTO 3D O URP

Diferencia entre la plantilla 3D y URP:
La plantilla 3D es más versátil, pero la plantilla URP es más eficiente.
Si tu PC no es muy potente, la plantilla URP es la mejor opción.
Si necesitas todas las funcionalidades de Unity, la plantilla 3D te dará más flexibilidad.

Plantilla 3D: Cualquier plataforma, requisitos mínimos de PC.
Plantilla URP: Móviles, Oculus Quest, PC de baja potencia.
Plantilla VR: Experiencias VR completas, requisitos medios a altos de PC.
Plantilla HDRP: Gráficos de alta calidad, solo para equipos potentes.

Plantilla 3D:

Ventajas:
Más general, compatible con cualquier plataforma.
No requiere optimización específica.
Desventajas:
No está optimizada para el rendimiento.
Puede consumir más recursos de los necesarios.

Plantilla URP:

Ventajas:
Optimizada para móviles y Oculus Quest.
Menor consumo de recursos.
Mejor rendimiento en equipos menos potentes.
Desventajas:
Menos opciones y configuraciones que la plantilla 3D.
Puede no ser compatible con algunos plugins o assets.

Escena de ejemplo

Abre la **escena de ejemplo** que se proporciona en Aula Virtual y Google Drive: (Assets → Import Package → Custom Package y seleccionar el archivo .unitypackage.). Esta escena contiene simplemente un plano para el suelo, un cubo grande que hace de mesa y un par de cubos pequeños con los que podremos interactuar.

Si decides cambiar a la plantilla URP, recuerda hacer una copia de seguridad antes de comenzar.

Incorporar el URP en un proyecto existente

Para incorporar el URP en un proyecto existente, ve a Window -> Package Manager, buscar e instalar "Universal RP". Crea un asset URP mediante clic derecho en Assets -> Crear -> Rendering -> URP Asset (Con Universal Renderer). En Edit -> Project Settings -> Grphics, arrastrar el activo URP creado al espacio destinado para la configuración del Scriptable Render Pipeline. Eso se hace con botón derecho en los assets > Create > Rendering > URP Asset (with Universal Renderer). Cambiarle el nombre. Crea dos cosas diferentes uno es el asset y el otro es el render que usa el asset. Vamos a arrastrar el Asset al Scriptable Render Pipeline y click en Continuar.

Tras la integración del URP, es posible que los materiales de los objetos en escena aparezcan en color rosa, indicando incompatibilidad con el nuevo pipeline de renderizado. Para resolver esto, hay que cambiar manualmente el shader de los materiales afectados uno por uno yendo al material y en su Shader cambiar de Standart a "Universal Render Pipeline > Lit". Pero eso es un proceso tedioso por lo que se podrían seleccionar todos e ir a la opción Editar -> Rendering -> Materiales -> Convert Selected Built-in Materials to URP (Convertir Materiales Incorporados a URP) > Proceed para convertir todos los materiales deseados al URP.
Para proyectos de gran envergadura, se recomienda el uso Convertidor del Pipeline de Renderizado. Para ello ve a Window -> Rendering -> Render Pipeline Converter. Aparece una ventana donde hay que cambiar Convert Build-in 2D (URP) a Built-in to URP. Si el proyecto es muy grande esto puede tardar mucho tiempo. Hay que seleccionar todas las casillas y darle a Initialize Converters y luego Convert Assets. De ese modo se ha configurado todo en la Project Settings > Quality.

El URP ofrece diversas configuraciones para equilibrar entre rendimiento y calidad visual. Si buscamos en Assets Pip, veremos todos los nuevos Assets y renders que se han creado. También podemos ver que en Project Settings > Quality ya están implementados. Ahí es donde podemos crear una nueva opción para Quest 2, llamarla Quest y arrastrarle el Asset que habíamos creado antes a Render Pipeline Asset. Luego en Graphics también en Scriptable Render Pipeline utilizaremos el mismo.

Si seleccionamos el Renderer URP que creamos podemos ver en el inspector Rendering > Render Path y elegir el renderizado **Forward** para aprovechar el MSAA y suavizar los bordes de los objetos, aunque el renderizado **Deferred** puede ser adecuado para escenarios con muchas luces dinámicas. Para optimizar el proyecto podríamos deshabilitar la mayoría de los efectos, por ejemplo, Shadows > Transparent Recieve Shadows.
También podemos deshabilitar la mayoría de los efectos de post procesamiento para aliviar la carga sobre la CPU y GPU, excepto cuando sea necesario para mejorar la experiencia del usuario. Seleccionamos el Asset URP que creamos vamos a sus opciones en el inspector y desactivamos también HDR para evitar interferencias con MSAA. Ajustamos el Anti-Aliasing a 4x (Recomendado por Meta) para mejorar la nitidez de los objetos.

INSTALAR XR PLUGIN MANAGER Y HABILITAR OPENXR

Para crear un Proyecto de RV en Unity para una amplia gama de dispositivos, debes instalar el XR Plugin Manager que sirve para manejar todos los diferentes plugins y complementos.
Ve a: Edit→Project Settings→ **XR Plugin Management** e instálalo.
En la pestaña de PC, selecciona OpenXR (puede reiniciar el editor). Haz lo mismo en la pestaña de Android. Luego, tenemos que añadir el/los dispositivos para los que vamos a desarrollar la experiencia VR pulsando en el '+'. En nuestro caso solo necesitamos **Oculus Touch Controller Profile.** Lo hacemos en la pestaña de PC y también en la de Android. Echamos un vistazo en Project Validation y si hay algún problema pulsamos en Fix.

OpenXR es una API (Interfaz de Programación de Aplicaciones) desarrollada por el Khronos Group, que sirve como un estándar abierto para el acceso unificado a una amplia gama de dispositivos de realidad virtual (VR) y realidad aumentada (AR). OpenXR facilita la creación de aplicaciones y juegos que pueden funcionar en múltiples plataformas y dispositivos sin necesidad de modificar el código para cada uno de ellos. Reduce significativamente el tiempo y los recursos necesarios para soportar diferentes dispositivos de VR y AR, incluido el Quest 2. Al adoptar un estándar abierto como OpenXR, nos aseguramos de que las aplicaciones sean compatibles con futuras tecnologías y dispositivos. Proporciona una serie de funcionalidades predefinidas para el manejo de dispositivos de entrada, seguimiento de movimiento, y otras tareas comunes en VR.

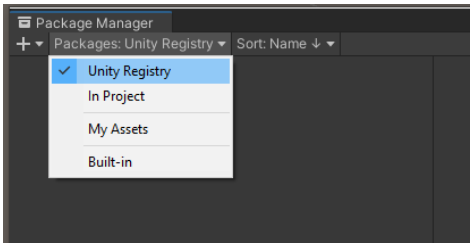
Build Settings: Switch Platform
El Oculus Quest 2 se basa en un sistema operativo Android. Por esta razón, cuando se desarrolla para Quest 2 utilizando Unity y OpenXR, es necesario configurar el proyecto para la plataforma Android, ya que esto garantiza que el software será compatible con el hardware y sistema operativo del Quest 2. Esto incluye el uso de APIs de Android y el cumplimiento de sus requisitos de rendimiento y compatibilidad.

Al configurar el proyecto y abrir la escena de ejemplo, Unity ya utiliza el dispositivo como una pantalla. Sin embargo, esta perspectiva aún no es dinámica; no sigue los movimientos de la cabeza. Para sincronizar los movimientos de la cabeza detectados por el dispositivo con la cámara en la escena, debemos incorporar un componente **'Tracked Pose Driver'** a la Main Camera.

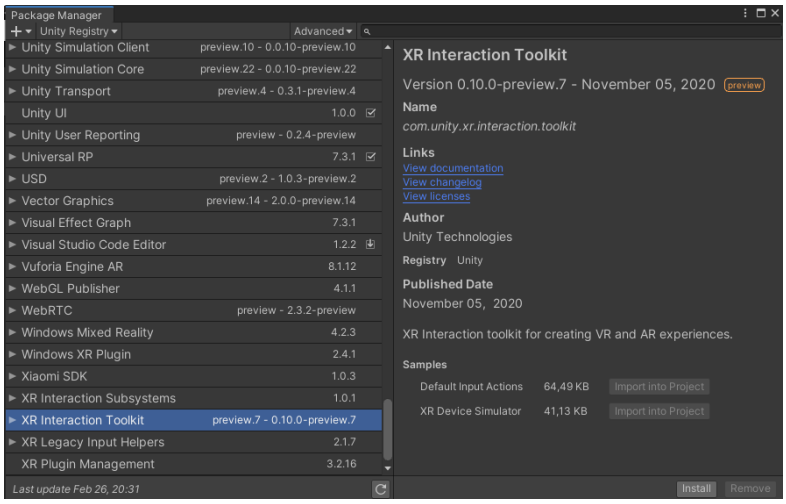
INSTALAR EL PAQUETE XR INTERACTION TOOLKIT

Vamos a emplear el paquete de herramientas más popular y recomendado por Unity, XR Interaction Toolkit.

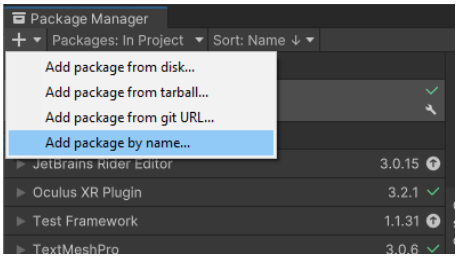
Ve a Window→Package Manager. Para ver los paquetes disponibles, tienes que seleccionar los Packages del **Unity Registry**:



Selecciona en la lista **XR Interaction Toolkit**, e instálalo:

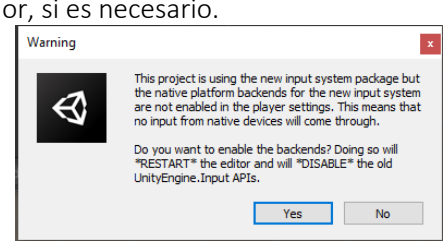


Si no aparece en la lista, puedes agregarlo directamente seleccionando la opción **Add Package by Name:**



Y escribiendo: **com.unity.xr.interaction.toolkit**

Si es la primera vez que se instala sale un aviso sobre el New Input System, que hay que aceptar y reiniciar Unity o el ordenador, si es necesario.



Selecciona XR Interaction Toolkit en Package Manager y en la sección a Samples e importa los 'Starter Assets'.

AÑADIR UN XR ORIGIN A LA ESCENA

Al instalar el paquete del Toolkit, aparece un nuevo grupo de GameObjects. Agrega a la escena el objeto **XR→XR Origin (VR).** Un XR Origin es un GameObject que representa al avatar del usuario dentro del mundo virtual.

Seleccionando el componente XR Origin vemos que ya viene con el **Input Action Manager** incluido que sirve para gestionar y definir las acciones de entrada que podemos realizar desde los controladores. En Action Assets debería tener asignadas las acciones de entrada predeterminadas **XRI Default Input Actions**.

Podemos explorar otros parámetros en el inspector, referentes a objetos que necesita el XR Origin:

- **Origin Base Game object** sirve como punto de referencia o anclaje en el mundo virtual desde donde se calculan todas las posiciones y orientaciones de los elementos XR, como cámaras y controladores. el XR Origin se puede agregar como componente de cualquier game object.
- **Camera Floor Offset Object:** permite ajustar la altura inicial de la cámara en el mundo virtual. Es el único parámetro configurable. Hay dos posibles valores:
    - **Device**: pensado para dispositivos que sólo tienen tracking 3DoF (rotaciones de la cabeza, pero no posición) o bien para aplicaciones que se puedan ejecutar por igual estando de pie o sentado. El XR Origin se ubicará en (0,0,0), lo cual nos haría ver la escena como si estuviésemos a ras del suelo. Para evitar esto, podemos especificar una altura para la cámara (Camera Y Offset).
    - **Floor**: pensado para dispositivos 6DoF en aplicaciones en las que queramos tener libertad de movimientos. La referencia en este caso será la altura del suelo que hayamos especificado en el sistema guardián, por lo que apareceremos a diferentes alturas según estemos de pie o sentados. Por este motivo no es necesario especificar una altura (offset Y de la cámara), ya que se tomará la que realmente tengamos.

- **Camera:** la cámara principal está ahora dentro de la jerarquía del XR Origin, por lo que eliminamos la otra cámara de la escena.

Al añadir el XR Origin también se genera el Administrador de Interacción XR **(XR Interaction Manager).** Es el responsable de toda la interacción entre nosotros (interactor) y los objetos (interactuable).

INPUT CONTROLLERS

Actualmente coexisten dos sistemas de control de los dispositivos de entrada (como el tracking del HMD y de los mandos, y los diferentes botones):

- **Device-Based**: es el único que existía hasta hace unos años. Permite asociar los controles de los dispositivos hardware (botones del mando, etc.) a eventos como rotaciones, triggers, etc. Al trabajar de esta manera es necesario conocer los controles que tiene cada mando, aunque existen mapeos genéricos que facilitan el desarrollo de soluciones multiplataforma asignando un alias a cada control.

  Este sistema está cayendo en desuso en favor del nuevo sistema Action-based.

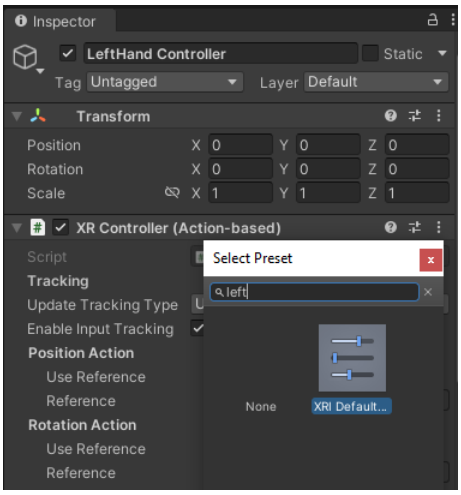
- **Action-based**: añade una capa de abstracción que hace el control de las entradas aún más independiente del hardware. En lugar de hablar de botones y otros controles, hablaremos de acciones. Los scripts no trabajarán directamente con los controles hardware, sino sólo con las acciones: en lugar de comprobar si se ha pulsado un botón, se comprueba si ha ocurrido una determinada acción.

Vamos a trabajar con el sistema action-based también porque es el que nos va a permitir emular el dispositivo y poder trabajar sin tener físicamente las Quest conectadas.

CREAR UN MAPA DE ACCIONES LIGADO A LOS MANDOS

XR Interaction Toolkit proporciona un mapeo de ejemplo que vamos a utilizar. Para ello, ve a la carpeta Samples dentro de Assets. Al hacer doble click sobre XRI Default Input Actions, nos abrirá el editor de Input Action Assets, donde hacemos el mapeo entre acciones y controles.

Si seleccionas uno de los controladores, verás que falta la lista de Acciones de Entrada. Para rellenarlas, necesitamos utilizar un mapa de Acciones de Entrada. Para usar los mapeos que nos proporciona XRI Toolkit, selecciona el controlador de la mano Izquierda y en el preset, el icono medio en la esquina superior derecha del componente XR Controller, selecciona XRI Default Left Controller. Repite los pasos para el Controlador Derecho.



INTERACTORS

Vemos que en lugar de manos tenemos unos controladores.
Los **Interactors** son los objetos que especifican cómo interactuamos con la escena y habitualmente van ligados a nuestras manos, mientras que los **Interactables** son los objetos de la escena con los que podemos interactuar.

Hay tres tipos principales de Interactors:

- **XR Ray Interactor:** pensado para acción a distancia. Por ejemplo, para UIs, juegos en los que no hay locomoción, etc.
- **XR Direct Interactor:** pensado para interacción directa con los objetos que estén justo en la posición de nuestras manos. Es la forma de interacción más natural.
- **XR Socket Interactor:** es el único que no está pensado para ir ligado a las manos sino a cualquier otro objeto. Por ejemplo, si queremos que una cerradura interactúe solo con su llave, la cerradura tendría un Socket Interactor.

Ahora configuraremos los interactores de manera individual para cada controlador.

Primero eliminamos los componentes añadidos por defectos de cada mano (XR Ray Interactor, XR Interactor Line Visual, Line Renderer, Sorting Group). Dejamos el XR Controller.

Haz clic derecho sobre **"LeftHand Controller"** y selecciona **XR > Direct Interactor (Action-based).** Como el objeto de la mano ya contiene un **XR Controller**, eliminaremos este componente del hijo Direct Interactor.

Repite este proceso para el "RightHand Controller", asegurándote de que todo sigue funcionando correctamente.

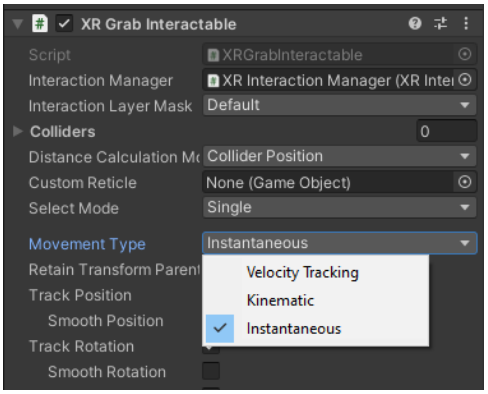
Ahora, para cada controlador, agrega de la misma forma un Ray Interactor (Action-based). También se puede añadir como nuevo componente pero hay que asegurarse de que este nuevo componente sea hijo del respectivo controlador. Dado que ya existe un XR Controller en el controlador, elimina el XR Controller del Ray Interactor, manteniendo sus tres componentes principales: el propio Ray Interactor, Line Renderer y XR Interactor Line Visual. Este último te permitirá configurar la apariencia de la línea del rayo.

La opción Interaction Layer Mask del XR Ray Interactor determina con cuáles de estas capas el "XR Ray Interactor" puede interactuar. Si un objeto está en una capa que está incluida en la máscara de interacción del interactor, entonces el rayo puede detectar e interactuar con ese objeto. Si el objeto está en una capa que no está incluida en la máscara, el interactor lo ignorará como si no estuviera allí.

GRAB INTERACTABLES

Para hacer que las cajitas en la mesa sean interactuables, tenemos que agregarles un componente de tipo Interactable. Vamos a agregarles el más habitual: **XR Grab Interactable**. Añade automáticamente un RigidBody (si no lo tenía ya).

Podemos modificar el parámetro **Movement Type** del XR Grab Interactable para hacer que el movimiento sea más realista.



Hay tres posibles valores para este parámetro:

- **Instantenous** (por defecto): mientras el objeto está agarrado, los movimientos de la mano modifican directamente el Transform del objeto. Pierdes toda la física, pero proporciona el menor coste computacional y visualmente muestra un agarre totalmente firme del objeto.
- **Kinematic**: mientras agarremos el objeto, éste se comportará como un rigid body cinemático e interactuará con otros rigid bodies. En cambio, atravesará objetos que no tengan ligado un rigid body. La diferencia fundamental con Instantenous se ve cuando tocas a otro objeto, aunque también se puede apreciar que el agarre no es tan firme.
- **Velocity tracking**: nuestros movimientos afectarán a las velocidades (lineal y rotacional) del rigidbody, por lo que sería el comportamiento más correcto físicamente pero no siempre el más deseable. Se ve afectado por la física de colisión con otros objetos, no pudiendo atravesarlos, aunque no tengan un rigid body. Provocará el agarre menos firme del objeto, lo cual puede ser un comportamiento deseable o no según la aplicación. El comportamiento es muy similar al método Virtual Proxy que veremos en el tema de Renderizado Háptico.

INTERACTABLES PERSONALIZADOS

XR Simple Interactable

Este componente permite hacer que cualquier objeto en Unity sea interactuable, habilitando interacciones básicas y sencillas. Sus características principales son:

- **Eventos Personalizables**: puedes asignar respuestas específicas a diferentes tipos de interacción. Por ejemplo, puedes configurar un objeto para que cambie de color, emita un sonido o realice una acción específica cuando el usuario lo agarra o lo toca.
- **Fácil Configuración**: Para hacer que un objeto sea interactuable, simplemente necesitas agregar el componente XR Simple Interactable a dicho objeto en el Editor de Unity. Luego, puedes personalizar su comportamiento mediante la configuración de las propiedades y eventos del componente.

- **Integración con el Ecosistema XR**: XR Simple Interactable está diseñado para trabajar de la mano con otros componentes del XR Interaction Toolkit, como los XR Interactors (por ejemplo, XR Direct Interactor, XR Ray Interactor) y los XR Targets. Esto facilita la creación de experiencias de interacción ricas y variadas dentro de tus proyectos.

El **XR Simple Interactable** es ideal para objetos que requieren una interacción básica pero efectiva, como botones, palancas, objetos que se pueden recoger o cualquier otro elemento interactuable en un entorno XR. Su simplicidad lo hace accesible para desarrolladores de todos los niveles, permitiendo concentrarse más en la experiencia del usuario y menos en la complejidad técnica.

XR Tint Interactable Visual

Si queremos cambiar el aspecto de un objeto para dar una idea de selección, existe un tipo de Interactable específico que es el **XR Tint Interactable Visual.**

Para ello elige el objeto interactivo en la jerarquía de Unity al que quieres aplicar el efecto. Con el objeto seleccionado, ve al Inspector y haz clic en "Add Component". Busca y selecciona "XR Tint Interactable Visual". En el componente, ajusta las propiedades como "Tint Color" para elegir el color de tinte. Decide cuándo se aplicará el tinte: en "hover" (Tint On Hover) y/o en selección (Tint On Selection). El campo 'Tint Renderers' en el componente 'XR Tint Interactable Visual' permite seleccionar específicamente qué partes (renderizadores) del objeto cambiarán de color al interactuar (opcional). Si necesitas especificar renderizadores particulares para el tinte, agrégalos en la sección "Tint Renderers", arrastrando y soltando los objetos Renderer desde tu jerarquía o escena al campo "Tint Renderers". Otra opción es hacer clic en el círculo pequeño al lado del campo de renderizador y seleccionar el renderizador deseado de una lista.
Esta configuración permitirá que el objeto cambie visualmente cuando se interactúe con él, sin necesidad de escribir código.

LOCOMOCIÓN POR MOVIMIENTO CONTINUO

Podríamos agregar los componentes que necesitamos directamente al XR Origin, pero para mantener un cierto orden, vamos a crear un objeto vacío como hijo y lo llamaremos **Locomotion System.** Luego le añadiremos un componente **Locomotion System**. En el inspector, en el campo de XR Origin referenciamos a nuestro XR Origin (XR Rig) arrastrándolo.

El Locomotion System puede gestionar diferentes **Providers de locomoción**, para tener flexibilidad por ejemplo de movernos tanto con movimiento continuo como con teletransporte.

Ahora, creamos un objeto vacío como hijo de **Locomotion System** y lo llamamos Movimiento o Move. Le agregamos un provider (Add Component) de movimiento continuo: **Continuous Move Provider (Action-based)**. Arrastramos en el campo **System** el **Locomotion System** (Locomotion System).

Opciones:

- La opción **"Enable Strafe"** se refiere a permitir el movimiento lateral. Al habilitar esta opción, nos podemos mover hacia la izquierda o hacia la derecha además de hacia adelante y hacia atrás, utilizando los controles de entrada definidos.
- **Enable Fly**
- **Use Gravity**. Hay dos opciones para usar la gravedad: **"Gravity Application Mode": Immediately:** Al elegir esta opción, la gravedad se aplica al avatar del jugador inmediatamente, sin importar otras condiciones o acciones en curso. Esto significa que la fuerza de gravedad actúa sobre el jugador en todo momento, lo que podría resultar en una experiencia más realista, ya que refleja cómo la gravedad funciona constantemente en la vida real. Esta inmediatez garantiza que el jugador experimente la gravedad desde el momento en que no hay soporte, como estar en el aire después de un salto. **Attempting Move:** Esta opción indica que la gravedad solo se aplicará cuando el jugador esté intentando moverse. Sería útil para subir una escalera. Esto puede crear un comportamiento donde el jugador siente la gravedad principalmente durante los movimientos activos, lo que podría utilizarse para ciertos estilos de juego o mecánicas donde se desea tener un control más directo o condiciones específicas bajo las cuales la gravedad influye en el jugador.
- **Forward Source** especifica la fuente que determina la dirección hacia adelante para el movimiento continuo. Usar la **Main Camera** como Forward Source puede ser ideal para experiencias de exploración o juegos en los que se espera que el jugador se mueva principalmente hacia donde está mirando. En situaciones donde el movimiento del jugador debe ser independiente de la dirección de la mirada, podría ser necesario configurar un Forward Source diferente o ajustar la mecánica de movimiento.

Referenciamos este movimiento a una sola mano o a las dos, activando la casilla de la mano correspondiente y eligiendo el preset: XRI LeftHand Locomotion/Move o XRI RightHand Locomotion/Move.

MOVIMIENTO CONTINUO CON GIRO

El Toolkit incorpora dos Providers para girar: Snap (giros en ángulos discretos) y Continuous (giro en ángulos continuos).

Ahora, creamos un objeto vacío como hijo de **Locomotion System** y lo llamamos Giro o Turn. Le agregamos un provider (Add Component) de movimiento continuo: **Continuous Turn Provider (Action-based).** Arrastramos en el campo System el Locomotion System (Locomotion System). Ahora, si nos movemos con la izquierda, querremos girar con la derecha y al revés. Así que de momento, activamos la casilla de la mano derecha y la referenciamos buscando Turn: XRI RightHand Locomotion/Turn (Input Action Reference). El giro continuo marea incluso más que el movimiento lineal continuo.
Es posible que el giro no funcione bien, esto se puede solucionar deshabilitando el **Strafe** (movimiento lateral) en Continuous Move Provider.

Ahora eliminamos o desactivamos el Continuous Turn Provider y agregamos un **Snap Turn Provider**. De nuevo referenciamos el Locomotion System y, en este caso, utilizamos la acción

predefinida **Snap Turn**. Si notas algún problema en el giro al ir hacia atrás, desactiva la opción Enable Turn Around.

MOVIMIENTO CONTINUO CON TUNNELING

Si queremos, podríamos utilizar Tunneling como una forma de reducir el mareo provocado por el movimiento continuo. Toolkit incluye un prefab que nos permite hacer tunneling de forma muy sencilla. Desde el Package Manager importamos el ejemplo del Toolkit **Tunneling Vignette.**

Buscamos el prefab Tunneling Vignette y lo arrastramos debajo de Main Camera, emparentándolo como hijo. Podemos ver la viñeta del túnel si cambiamos en el inspector, en el campo **Preview in Editor** a Default Parameters.

Podemos asociar el túnel a cualquier provider de locomoción, agregando el provider a la lista **Locomotion Vignette Providers** (en nuestro caso por ejemplo a Move) del componente **Tunneling Vignette Controller**.

Si ejecutas ahora la escena verás que el túnel está siempre activo. Esto es debido a un pequeño bug que hace que, al tener un personaje con gravedad, considere que constantemente nos estamos moviendo hacia el suelo. La solución es en el Continuous Move Provider, cambiar el Gravity application mode a Attempting move.

LOCOMOCIÓN POR TELETRANSPORTE

El teletransporte es un método muy común de movimiento en VR, además del movimiento continuo.

Para implementar el teletransporte se requiere un Proveedor de Teletransporte (**Teleportation Provider**), un Interactor de Teletransporte (**Teleportation Interactor**), y un Área de Teletransporte (**Teleport Area**).

Creamos otro objeto vacío como hijo de nuestro Locomotion System y lo llamamos Teleportation Provider. Le agregamos (Add Component) un nuevo componente llamado Teleportation Provider y arrastramos al campo System el nuestro sistema de locomoción: Locomotion System.

Puedes ajustar en las propiedades del "Teleportation Provider", "Delay Time" para agregar un tiempo de retraso al teletransporte, donde puede aparecer la viñeta.

Ahora busca en la carpeta de prefabs de XR Interaction Toolkit el "Teleportation Interactor". Arrastra el prefab "Teleportation Interactor" al controlador de la mano derecha, que parece la mano más natural para ello. Si le has asignado el movimiento continuo a esta mano conviene quitarlo porque prevalecerá ante el teletransporte. Puedes dejar el movimiento continuo para la mano izquierda y teletransporte para la derecha.

Ahora selecciona el "Teleportation Interactor" y, en el panel "Inspector", busca el campo "Interaction Layer Mask". Asegúrate de que esté configurado para interactuar con la capa "teleport". Si no existe, ve a "Edit > Project Settings" > "Tags and Layers" y crea una nueva capa llamada "teleport".

Al configurar el interactor de teletransporte, es posible que no esté configurado para interactuar con una capa llamada "teleport", ya que esta capa no se añade automáticamente con el XR Interaction Toolkit. Para resolver esto, debes crear manualmente la capa "teleport" en las configuraciones del proyecto.

Ve a "Edit > Project Settings > Tags and Layers en Unity.
Busca la capa "User Layer 31" y nómbrala "teleport".

Ahora, regresa al interactor de teletransporte y en el panel "Inspector", bajo "Interaction Layer Mask", asegúrate de que la capa "teleport" esté seleccionada.

Es una buena práctica verificar si hay errores relacionados con las capas de interacción. Puedes hacerlo en "Project Settings > Project Validation". Si encuentras un error que indica que la capa 31 debe estar configurada para teletransporte, simplemente haz clic en "Fix" para resolverlo automáticamente.

Esta capa es esencial para que el sistema de teletransporte funcione correctamente, permitiendo que el interactor de teletransporte interactúe únicamente con los objetos asignados a esta capa.

Pero todavía no podemos teletransportarnos porque no tenemos a dónde.

TELEPORTATION AREA

Ahora vamos a definir el área a la que nos podemos teleportar. Para ello empezamos seleccionando el suelo y añadiéndole un componente **Teleportation Area**. Un Teleportation Area es un tipo de **Interactable** que interactua con el Ray Interactor. Nos aseguramos que en la capa tiene asignada Teleport. También podemos hacer que toda una zona sea teletransportable seleccionando, por ejemplo, un plano, en nuestro caso cada uno de los elementos de la plataforma, añadir el componente "Teleport Area" y configurarlo con El Interaction Manager y la capa Teleport.

En su lista de **Colliders** arrastramos la malla del propio suelo.

Ejecuta la escena y comprueba que te puedes teleportar a cualquier punto del suelo al que apuntes, utilizando el control de **Grip,** ya que por defecto el teletransporte se asocia a la acción Select.

En **Teleportation Configuration** puedes personalizar la configuración del área de teletransporte, como la orientación del jugador. Por ejemplo, si **Match Orientation** está configurado en "target up and forward", el jugador siempre se orientará según el eje Y (up) y el eje Z (forward) del ancla.

Puedes hacer que toda una zona sea teletransportable seleccionando, por ejemplo, un plano y añadiendo el componente "Teleport Area". Asegúrate de configurar correctamente la capa de interacción y desactivar la capa por defecto.

Aparte de este modo, también tenemos un prefab de XRI Toolkit para eso: Teleport Area y Teleport Anchor que podemos utilizar directamente.

DESTINOS DE TELETRANSPORTE PREFIJADOS (TELEPORT ANCHOR)

Por último, vamos a ver un nuevo tipo de Interactable que nos permite definir destinos de teletransporte prefijados: los **Teleport Anchor**. Podemos detallar una posición (y orientación) concreta a la que nos teleportaremos, para tener objetivos prefijados en posiciones de interés para nuestro juego/aplicación.

Podemos agregar un Teleport Anchor como componente, pero lo que vamos a hacer es utilizar el prefab de XRI Toolkit: Teleport Anchor. Vemos que ya tiene asignada la capa y le podemos arrastrar el XR Interaction Manager a su XR Interaction Manager.

Para definir el punto y rotación concretos de teleportación a la plataforma puedes definir un **Transform**.