



Create with VR - 2021 LTS

All Lesson Plans

Getting Started - Tech Setup	3
Lesson 0.1 - VR Software Setup	3
Unit 1 - VR Basics	7
Lesson 1.1 - VR Project Setup	8
Lesson 1.2 - VR Locomotion	23
Lesson 1.3 - Grabbable Objects	30
Lesson 1.4 - Sockets	37
Challenge 1 - Architecture Review App	42
Lab 1 - VR Personal Project Basics	44
Quiz 1	48
Unit 2 - Events & Interactions	53
Lesson 2.1 - Audio and Haptics	54
Lesson 2.2 - Activation Events	60
Lesson 2.3 - Direct and Ray Interactors	67
Lesson 2.4 - User Interfaces	75
Challenge 2 - 3D Painting App	83
Lab 2 - Personal Project Events & Interactions	85
Quiz 2	87
Unit 3 - VR Ergonomics & Optimization	91
Lesson 3.1 - Comfort and Accessibility	92
Lesson 3.2 - Optimization	100
Lesson 3.3 - Lighting	109
Lesson 3.4 - Building and Sharing	117
Challenge 3 - Training Simulation App	121
Lab 3 - Personal Project Optimization & Lighting	123
Quiz 3	126
Other tutorials	129
VR Next Steps	129
How to set up a VR Project from Scratch	132


Course Description

In this official course from Unity, you will learn to design and develop your own Virtual Reality (VR) applications. You will create prototypes, attempt challenges, and complete quizzes to build and solidify your skill set. At the same time, you will be guided through creating your own unique VR project from start to finish, beginning with a blank design document and ending with a fully functional project. Whether you want to create an interactive walkthrough of an ancient ruin, a product configurator for a car manufacturer, a simulator for operating dangerous machinery, or any other experience, this course will help you bring those ideas to life in VR.

Thanks to Andrew, creator of the [VR with Andrew](#) YouTube channel, for collaborating with us on this course.

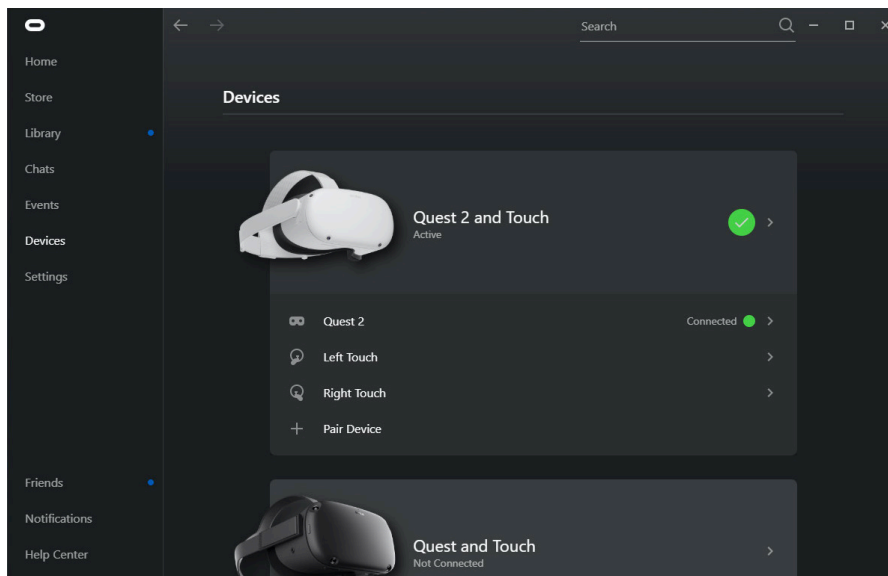
Getting Started - Tech Setup

Lesson 0.1 - VR Software Setup

Lesson Link	VR Software Setup - Unity Learn	
Length	1 Hour 30 Minutes	
Lesson Objective:		

Step	Instructions
1 - Overview	<p>In this lesson, you will get all the necessary software for your computer and your VR device installed and configured for VR development.</p> <p>First you will install the recommended version of Unity to be able to follow along with this course. Depending on which VR device you intend to use, you may also need to install additional export modules.</p> <p>You will also make sure that your VR headset is configured properly for development and testing, including downloading any additional device-specific software that is required.</p> <p>Note: If you do not have access to a VR headset, you can still benefit from this course, but your ability to fully test the VR experience will be somewhat limited.</p>
2- Course requirements	<p>Your Unity experience This course assumes you have at least a basic understanding of the Unity Editor. If you have never used Unity before, welcome! The Unity Essentials learning pathway has been designed to help you get set up and ready to create in the Unity Editor. We recommend you complete this pathway to learn the Unity basics before continuing with this course.</p> <p>Your VR experience You'll get the most out of this learning content if you have some previous basic experience with VR.. If you haven't explored any VR experiences before, try the basic tutorial content created for your VR hardware. You do not need any experience developing for VR to use this learning content.</p> <p>Your VR hardware This learning content is fully verified with Meta Quest model headsets. You can use any other headsets that support OpenXR, including the Valve Index, HTC Vive, and Windows Mixed Reality Headsets, but you are more likely to encounter obstacles along</p>

	<p>the way. Everything should function properly, but the learning content has not been fully tested and verified on these devices.</p> <p>Note: If you do not have access to a VR headset, you can still benefit from this course, but your ability to fully test the VR experience will be somewhat limited.</p>
3 - Before you begin	<p>Update the Unity Hub</p> <p>Before you begin to set up your Unity project, consider updating your Unity Hub to the latest release. If you are using an older version of the Hub, there may be differences between the guidance provided and your experience.</p> <p>Install Unity 2021.3 LTS</p> <p>Before you can follow along with this course, you must install a particular version of Unity and, if using an Quest, an additional Android Build Support module.</p> <ol style="list-style-type: none"> 1. Select Unity 2021.3 LTS: <ul style="list-style-type: none"> From Unity Hub, select to install Unity 2021.3 (LTS). <ul style="list-style-type: none"> This LTS (Long-Term Support) version of Unity is stable and will be fully supported for two years, making it ideal for development. 2. If you are using a Meta Quest, install the Android Build Support Module: <ul style="list-style-type: none"> When prompted to install additional export modules, select Android Build Support. <ul style="list-style-type: none"> This will enable you to build Android apps (.apk files), which run on Quest devices. If you are not using a Quest, you can leave all export modules unselected. 3. Download and install the Unity Editor: <ul style="list-style-type: none"> Follow the prompts in Unity Hub to continue downloading and installing Unity 2021.3 LTS, then wait for it to install. <p>You should now have the recommended version of Unity installed along with any additional required export modules.</p>
4 - Quest device setup	<p>If you are using a Quest or Quest 2, complete this step. Otherwise, skip this step.</p> <p>Before you can begin development with a Quest, there are a few steps required to make sure the device is configured appropriately.</p> <ol style="list-style-type: none"> 1. Put your device in Developer Mode: <ul style="list-style-type: none"> Follow the instructions on Device Setup from the Quest Developer page to put your device in Developer Mode. <ul style="list-style-type: none"> This will allow you to do testing and development on your device. 2. (Windows only) Install the Quest software and connect your device: <ul style="list-style-type: none"> If you are using a Mac, skip this step. From the Quest Setup page, download and install the correct application or software for your device. Within the application, follow the instructions to add your headset, either by Link cable or by Air Link.



3. (Mac only) Using Quest devices with a Mac:

- You won't need to download any additional software in order to use your device with a Mac. There will be further instructions on how to test your app with a Mac in a later tutorial.

You should now have your Quest device ready for development with Unity.

5 - Other device setup

If you are using a headset that is compatible with SteamVR, like the **HTC Vive**, **Valve Index**, or **Windows Mixed Reality** headset, complete this step. Otherwise, skip this step.

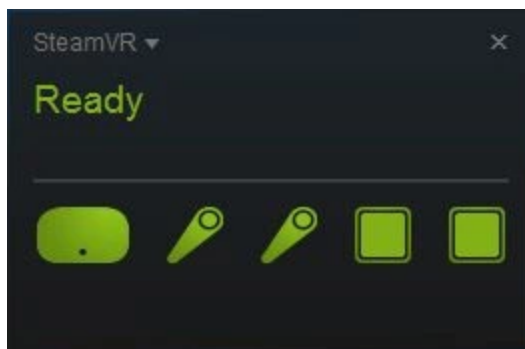
Before you can begin development with an HTC Vive, HTC Vive Pro, or Valve Index, you need to install Steam and SteamVR.

1. Install Steam:

- Go to the [Steam download page](#) and follow the instructions to download and install Steam.

2. Install SteamVR and link your device.

- Go to the [SteamVR download page](#) and follow the instructions to download and run SteamVR.
- Plug your headset into your computer and make sure it is linked and recognized by SteamVR.



You should now have SteamVR running with your device successfully connected and recognized.

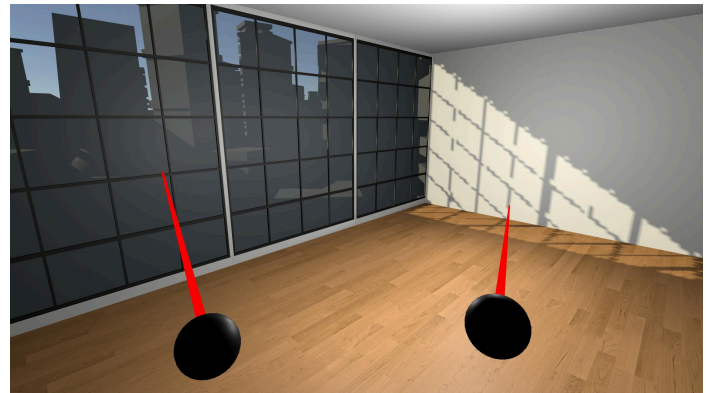
<p>*Optional* 6 - Set up version control</p>	<p>Using version control throughout this course is optional, but highly recommended to help keep a record of your work and make sure you don't lose any progress.</p> <p>If you are unfamiliar, version control (sometimes called source control) is a system that manages and tracks information. Version control software can be configured to store, manage, and track changes to any kinds of files, software, websites, or other data. If you want, check out this version control tutorial to learn more about it.</p> <p>We won't walk you through this process during the course, so it's up to you to do it on your own. Any time you're asked to set up a new project, you should set it up with version control. After every significant change, you should push an update to your cloud repository. This will make sure that you're able to recover previous versions of your project if you need to.</p> <p>Version control is important for your own development, but it's also important for getting jobs! Potential employers might even require that you submit a GitHub link in your application. So it's a good idea to get familiar with using version control now.</p>
<p>Recap</p>	<p>You have now set up your computer with the required version of Unity, the correct software for your device, and you have configured your device to be ready for development. With these tasks complete, you are ready to start working on your first VR project!</p>

Unit 1 - VR Basics

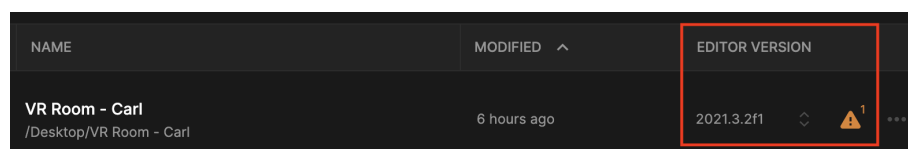


Lesson 1.1 - VR Project Setup

Lesson Link	1.1 - VR Project Setup - Unity Learn
Length	1 Hour 30 Minutes
Lesson Objective:	<p>By the end of this lesson, you will have a new unity project with a big empty room that you will experience in VR</p> <p>This unit is part of the Create with VR course.</p>



Step	Instructions	Teacher Notes
Step 1 - Open a new VR project	<p>The first thing you have to do is create a new Unity project configured for VR.</p> <ol style="list-style-type: none"> Download and extract the Create with VR starter project: <ul style="list-style-type: none"> Click to download the file: Create with VR Starter Project. To extract on Mac: double-click the downloaded file. To extract on Windows: right-click the downloaded file and select Extract All. Rename and relocate the project on your computer: <ul style="list-style-type: none"> Locate the folder called "VR Room Project" inside the extracted folder. If you want, rename the project folder to "VR Room - [Your Name]" Move the project folder to a logical directory on your computer (e.g. a folder called "Create with VR" on your desktop). Add the VR Room project to the Unity Hub: <ul style="list-style-type: none"> Open the Unity Hub. Add your VR Room Project to the list of projects in the Unity Hub. <p>If you are not sure how to do this, check out this tutorial on adding projects to the Unity Hub.</p> <ul style="list-style-type: none"> If you see a warning icon next to the Editor version of your project, don't worry. 	<p>Warning: Watch, then do</p> <p>Warning: Need to use correct version of Unity</p> <p>Explain: Starter project provided just has packages and settings configured - if you want to set up a project on your own, there is a link below. There is also a VR Template for LTS versions, but it doesn't quite have everything we need.</p> <p>Explain: What can you do with the package manager?</p> <p>Explain: Why URP? For mobile, VR, good optimized graphics. For higher end devices, might want to use HDRP, but that's much more complex, too.</p> <p>Warning: May take a while to open the first time- has to build the entire</p>



version of Unity that you don't have on your computer. As long as you have installed a version of Unity beginning with 2021.3, the project will run perfectly.

4. Open the project in Unity 2021.3 LTS (Long-Term Support):

- From the **Editor Version** dropdown, select a version of Unity beginning with 2021.3, then select **Open with 2021.3.X**.
- If a window pops up asking if you want to "Change Editor Version?", select **Change Version**.

Change Editor version?

If you change the Editor version of your project, the scripts might change and the project library might be rebuilt. Depending on the size of your project, this might take some time.

Are you sure you want to change the Editor version of your project?

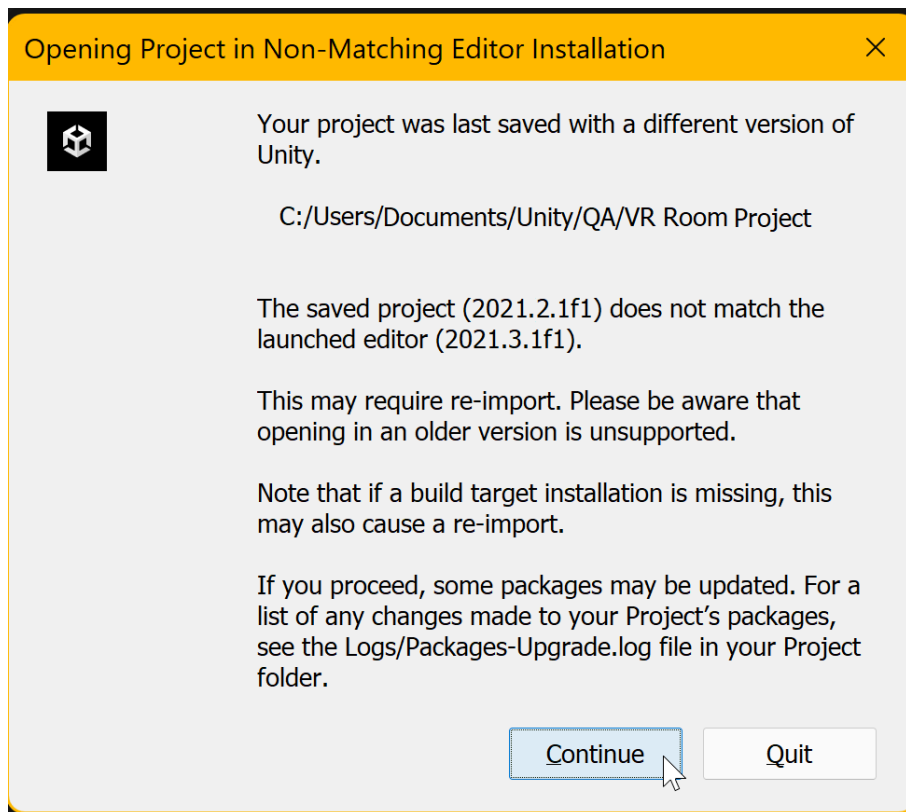
Cancel

Change version

- If a warning pops up that you are Opening Project in a Non-Matching Editor Installation", select **Continue**.

Library folder.

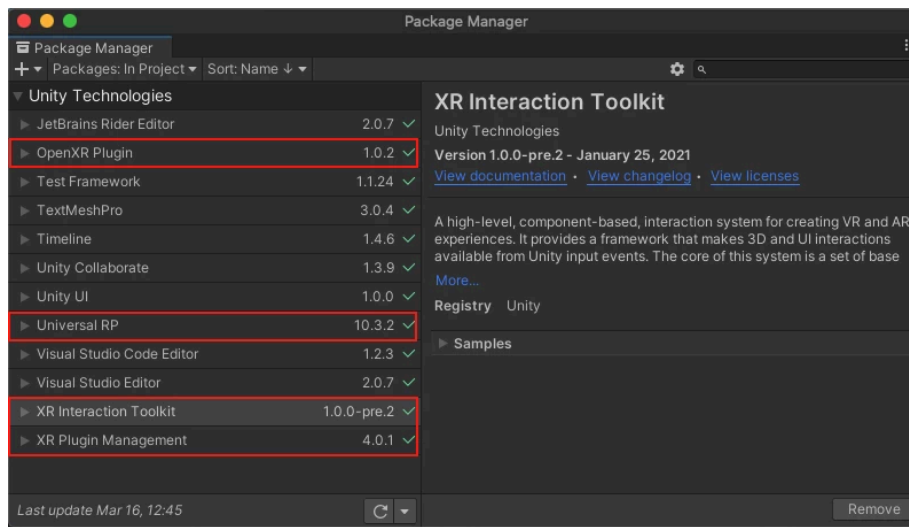
Explain: What does XR Stand for? Extended Reality, includes Mixed, Augmented, and Virtual Reality.



- Your project should now open in Unity.
- If there are yellow warning messages in the Console window about some of the assets, you can ignore them or clear them.

5. Explore the contents of the starter project and how it differs from a typical empty project:

- From the top menu, select **Window > Package Manager**.
- In the left panel of the Package Manager, locate the packages installed in this project (with checkmarks next to them), including:
 - XR Plugin Management
 - XR Interaction Toolkit
 - Universal RP (Render Pipeline)
- **Note:** to filter and view only packages currently installed in the project, from the top-left corner of the Package Manager window, select **Packages: In Project**.



You should now have your new VR starter project open to an empty scene and understand which packages allow for compatibility with VR.

Related Resources:

- [Blog about the Unity XR Plugin Framework](#)

Step 2 - Open and explore the starter scene

Before you run the project, you should choose a more interesting environment to experience in VR.

1. Rename and open the Starter Scene:

- From the **Project** window, open the **Scenes** folder.
- Rename the "Create-with-VR_Starter-Scene" as "[Your_Name] Room".
- Double-click on your renamed scene to open it.

2. Explore the contents of this scene and how it differs from a typical empty Scene:

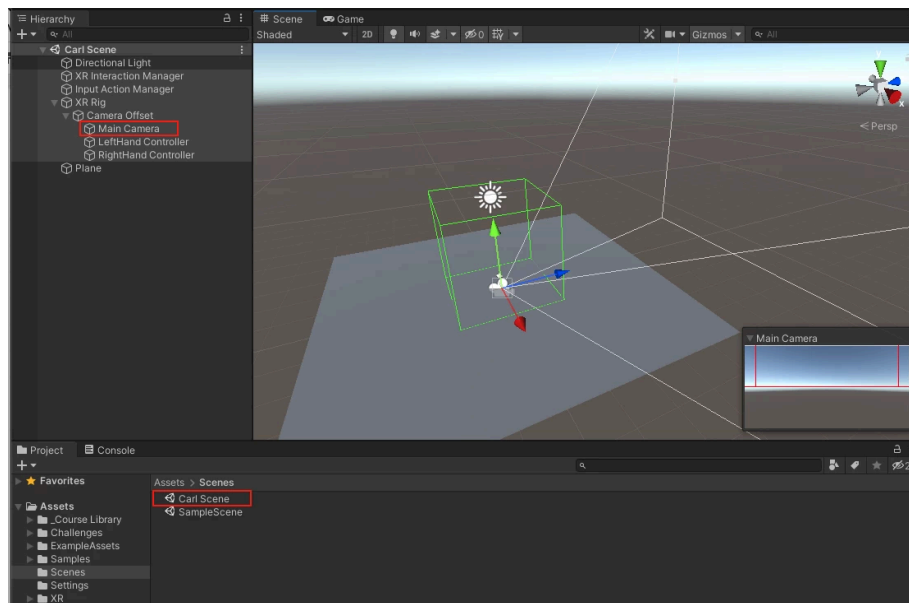
- From the **Hierarchy**, select the **XR Rig** object and inspect the XR Origin component.
- Select **XR Rig > Camera Offset > Main Camera** and inspect the Tracked Pose Driver component.
- Select either the **LeftHand Controller** or **RightHand Controller** object and inspect the XR Controller component.
- Select the **Input Action Manager** object and inspect the Input Action Manager component.

Explain: What is this course library?

What's included?

Demo: When you add an XR Rig to a default scene, it steals the Main Camera and makes it a child object.

Explain: Roomscale vs Stationary rig - show difference on Tracking Origin Mode for XR Rig (6 DOF vs 3 DOF)



You should now have your new scene open. You should also have a broad understanding of the components of the scene that make it different from a typical Unity scene.

Note: This scene and its settings have been pre-configured in this starter project, since they are required for VR development in this course. If you want to learn how to configure a VR-ready project similar to this from scratch, there are instructions in this tutorial: [Create a VR Starter Project from Scratch](#).

Step 3 - Add a room and background

This empty scene would be boring to explore in VR, so you should add a room that will be a bit more interesting and will provide more perspective when you're in VR.

1. Add a room to the scene:

- In the **Project** window, expand **Course Library > _Prefabs > Rooms**.
- Drag one of the **Room_[style]** prefabs into the Hierarchy.
- From the Hierarchy, delete the **Plane** object.

2. Add an environment outside the room's windows:

- Open the **Course Library > _Prefabs > Environments** folder
- Drag one **Foreground** object and one **Background** object into the Hierarchy.

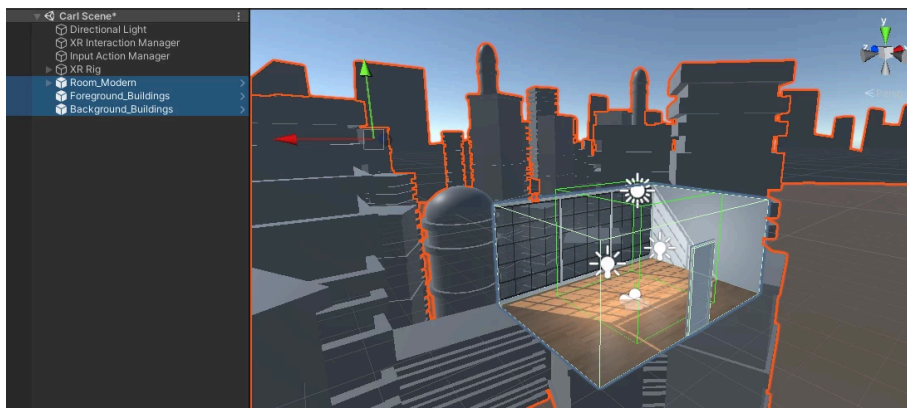
3. Adjust the sunlight in the room:

- Change the X and Y rotation of the **Directional Light** object to change the way sunlight enters your room.

Demo: Show the different styles of rooms.

Demo: Show the different environments (buildings vs hills) for foreground and background.

Demo: Different creative options for room with accompanying furniture for each one.



You should now have room, foreground, and background objects in your scene from the course library, with sunlight entering the room at the desired angle.

Step 4 - Run the app with the Device Simulator

You can test the scene with a **Device Simulator**. This simulator allows you to test the app in-editor using the mouse and keyboard, rather than having to connect to a device and put it on. This can be helpful for quick tests.

1. Add the Device Simulator to the scene:

- From the Project window, open **Samples > XR Interaction Toolkit > [version] > XR Device Simulator**.
- Drag the **XR Device Simulator** Prefab into the Hierarchy.
- Click **Play** to test the simulator.

2. Experiment with the keyboard and mouse controls:

- **Note:** To use the simulator effectively, a mouse with a clickable scroll wheel is required.
- To control the **camera**: hold right-click.
- To control the **left controller**: hold Left Shift or toggle with **T**.
- To control the **right controller**: hold the Spacebar or toggle with **Y**.
- To **pan around** with a device: move the mouse.
- To **rotate** a device: hold the middle mouse button.
- To **reset** the position and rotation of devices: press V.
- If you'd like a helpful guide, download the PDF of the [Rig Simulator Shortcut Cheat Sheet](#).

3. See more controls available for the device simulator:

- From the Hierarchy, select the **XR Device Simulator**.
- In the XR Device Simulator component, double-click on one of the Action variables to open the Action Editor.
- Expand the Actions to view their Bindings.
- From the left Action Maps panel, select either **Main** or **Input Controls** to view additional action mappings.

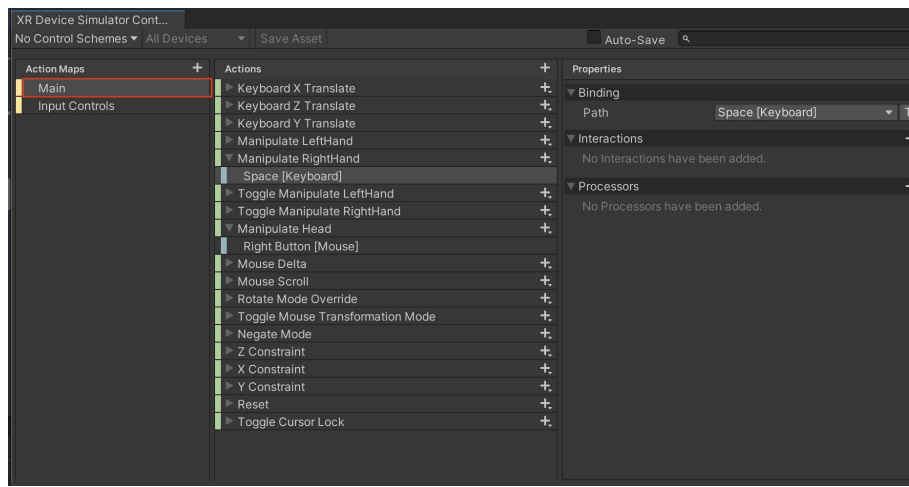
4. Important: disable the Device Simulator if you do not plan on using it:

- Select the Device Simulator object then disable it in the Inspector window.
- Having the device simulator active when running the project on your device will cause problems.

Warning: Using this is incredibly finicky and can be frustrating. You're trying to operate 3 devices, each with 6 degrees of freedom, thumbsticks, buttons, etc with one mouse and keyboard.

Warning: You really need a 3-button mouse (with a scroll wheel) to operate this.

Warning: Your head might start on the ground by default.



You should now be able to look around your room with your mouse and keyboard using the Device Simulator.

Step 5 - Test in VR through Unity

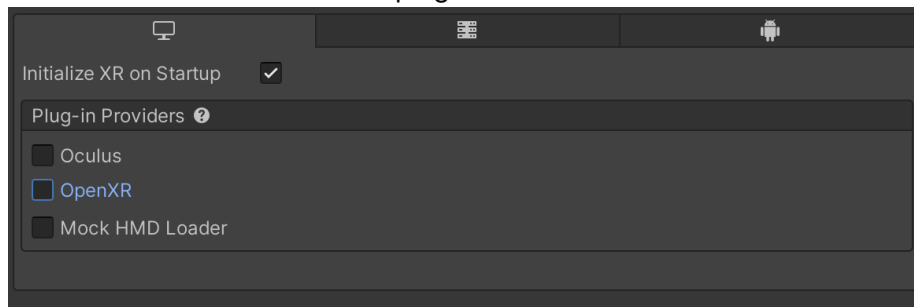
If you are using a **Windows** computer, follow this step to test on your device. Otherwise, skip this step.

If you are using a **Mac** for development, this step is not relevant. You will learn how to test your app in the next step when you build your app directly onto your device.

In order to run the project on your device through Unity, you need to install the correct plug-in for development. [OpenXR](#) is an open source API that connects Unity to supported VR hardware devices. It is recommended that you use this plug-in, since it allows you to deploy to multiple devices through a single, standardized API.

1. Install the OpenXR Plugin for desktop testing:

- From the top menu, select **Edit > Project Settings**, then select the **XR Plug-in Management** panel from the sidebar.
- In the **Windows, Mac, Linux** tab, select **OpenXR** from the list of available Plug-in Providers to install that plug-in.

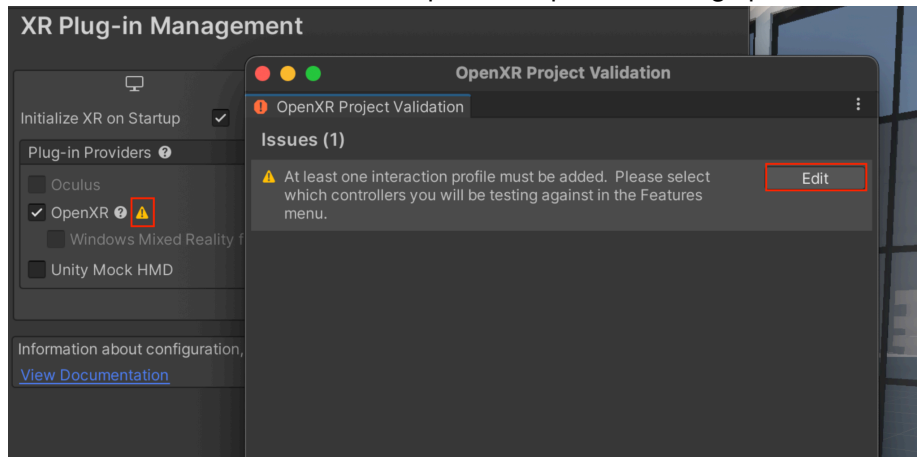


2. Resolve any warnings by setting up an interaction profile.

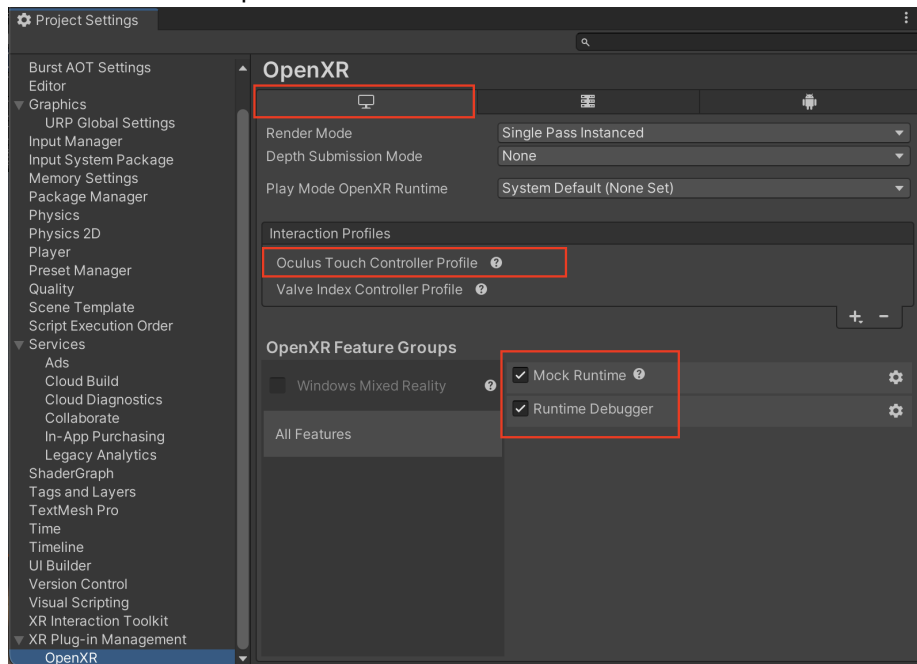
- After adding the OpenXR plugin, there may be a warning or error icon that appears next to the plugin Name.
- If there are no warnings or errors, you can skip to step 3 below to connect your device through the Quest Link Software.
- If there is a warning, click on the warning icon to open the **OpenXR Project Validation** window, which will tell you that you need to add an **interaction profile** for the device

you're using.

- Select the **Edit** button to open the OpenXR settings panel.



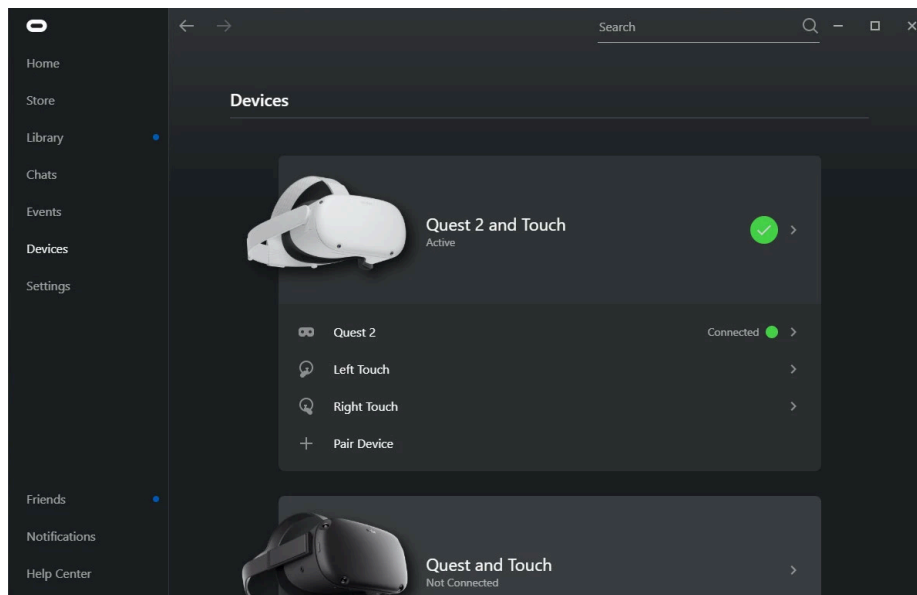
- In the **Windows, Mac, Linux** tab, make sure the **Oculus Touch Controller Profile** appears in the list of Interaction Profiles, then enable all available **OpenXR Feature Groups**.
- If you are using a different device, such as the Valve Index or HTC Vive, add its interaction profile instead.



- There should no longer be any warnings in the XR Plugin Management panel. If there are, select them and follow the recommended steps to resolve them.
- If you are having trouble resolving warnings or errors, or if you need help troubleshooting, check out this detailed [documentation on configuring the OpenXR plugin](#).

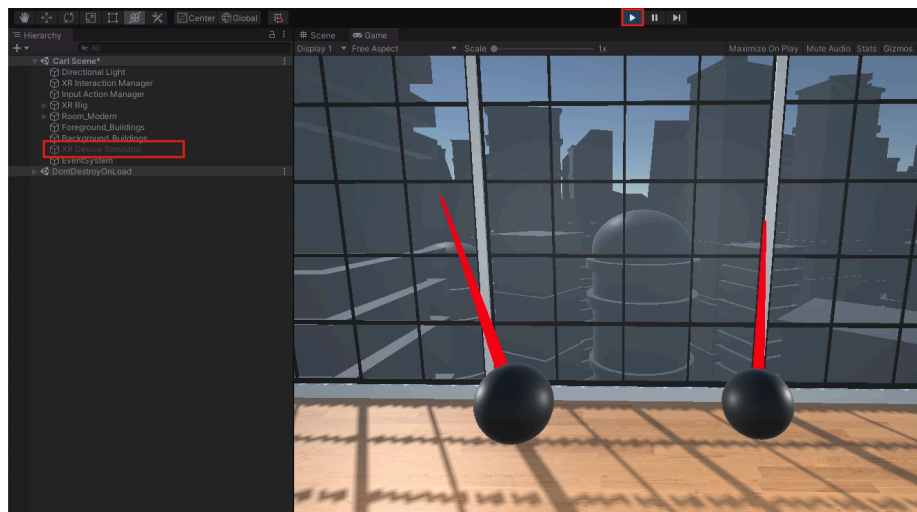
3. Connect your device through the Quest Link Software:

- Make sure your device is plugged into your computer using a compatible cable.
- Make sure the Quest App is running and has successfully recognized your device.
- If you have not yet downloaded the Quest App, you find it on the [Link Setup page](#).



4. Test the project on your device:

- Make sure the **XR Device Simulator** in your scene is disabled. If it is enabled, your head tracking will not work.
- Select **Play** in the Unity Editor and put on your headset.



On a Windows computer, you should now be able to quickly test your app on your device through Unity.

Step 6 - Build your app onto your Quest

If you are using a Quest, follow this step to build the app onto your device. Otherwise, skip this step.

Since the Quest is a standalone device (meaning it does not need to be connected to a computer), you can build an app and test it directly on the device, disconnected from your computer.

If you are on a Mac and cannot test your app through the Windows Quest software as in the previous step, this will be the primary way for you to test your app on your headset.

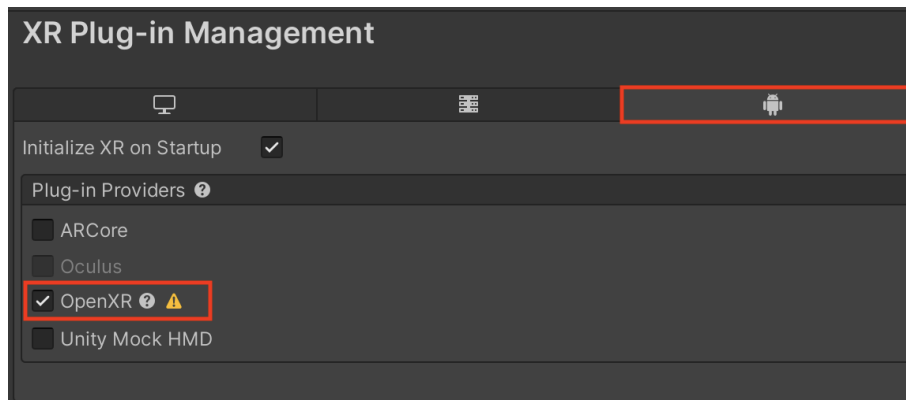
In order to build your app to the Quest headset, you will need to configure plug-ins for the Android build target.

1. Install the OpenXR for Android builds:

- From the XR Plug-in Management settings, select the **Android** tab, then select **OpenXR** from the list of available Plug-in Providers to install that plug-in.

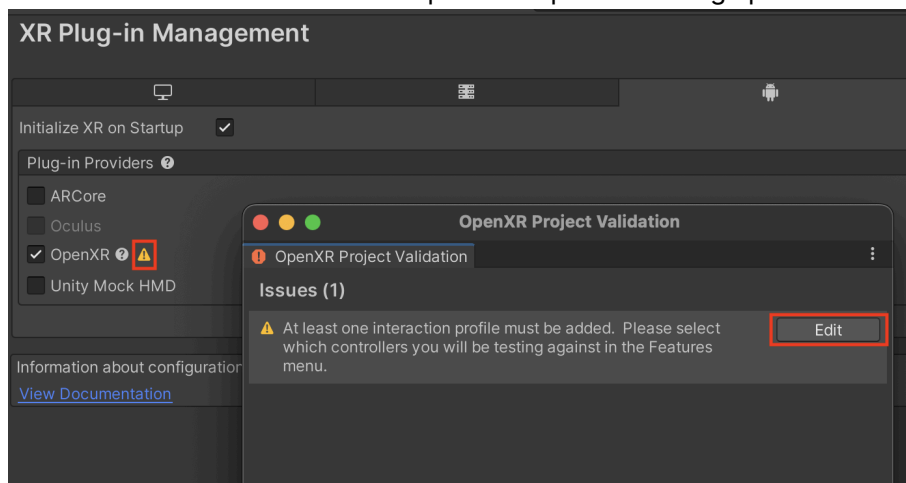
Note: You will only have an Android tab if you installed the Android export module.

- If you don't see an option for **OpenXR** in the Android tab, first go back to the Desktop tab and select OpenXR there, then return to the Android tab.



2. Resolve warnings by setting up an interaction profile.

- Click on the new warning or error icon that appears to open the **OpenXR Project Validation** window, which will tell you again that you need to add an **interaction profile**.
- Select the **Edit** button to open the OpenXR settings panel.



- In the **Android** tab, select the + button to add the appropriate

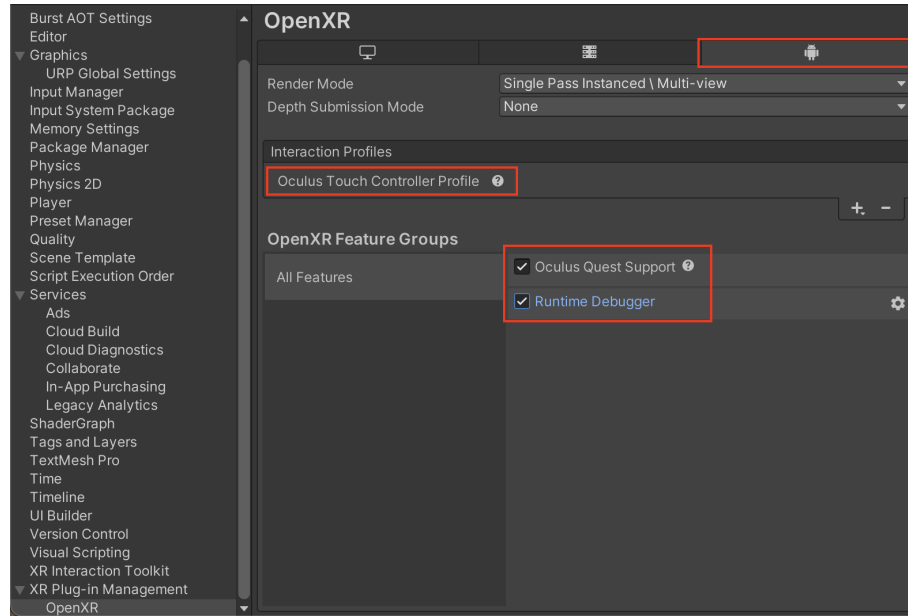
Explain: Why select Android? What does Android have to do with Oculus?

Warning: May take a while to build the first time.

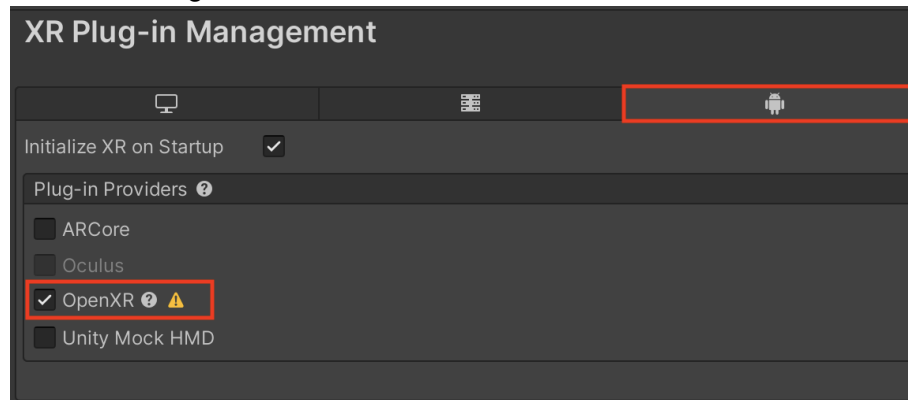
Warning: If you get an error saying that the Min API level must be set to something - can be set in Build Settings > Player Settings > Android > Other Settings > Minimum Api Level

Warning: Can be very frustrating if the device doesn't show up or if building doesn't work for some reason - may have to do some troubleshooting.

profile for your device, then enable all OpenXR Feature Groups.



- If any additional errors or warnings appear, click on the error or warning icons to resolve them.



3. Prepare your scene for building:

- Make sure the XR Device Simulator in your scene is disabled. If it is enabled, your head tracking will not work.
- From the top menu, click **File > Build Settings**.
- Click **Add Open Scenes** to add your scene.

4. Switch to the Android build platform:

- In the **Platform** section, select **Android**.
- Click **Switch Platform**, and wait for your project to switch to the Android platform.
- **Note:** Android will only show up as a possible build target if you successfully installed the Android Export Module during installation of the Unity Editor.

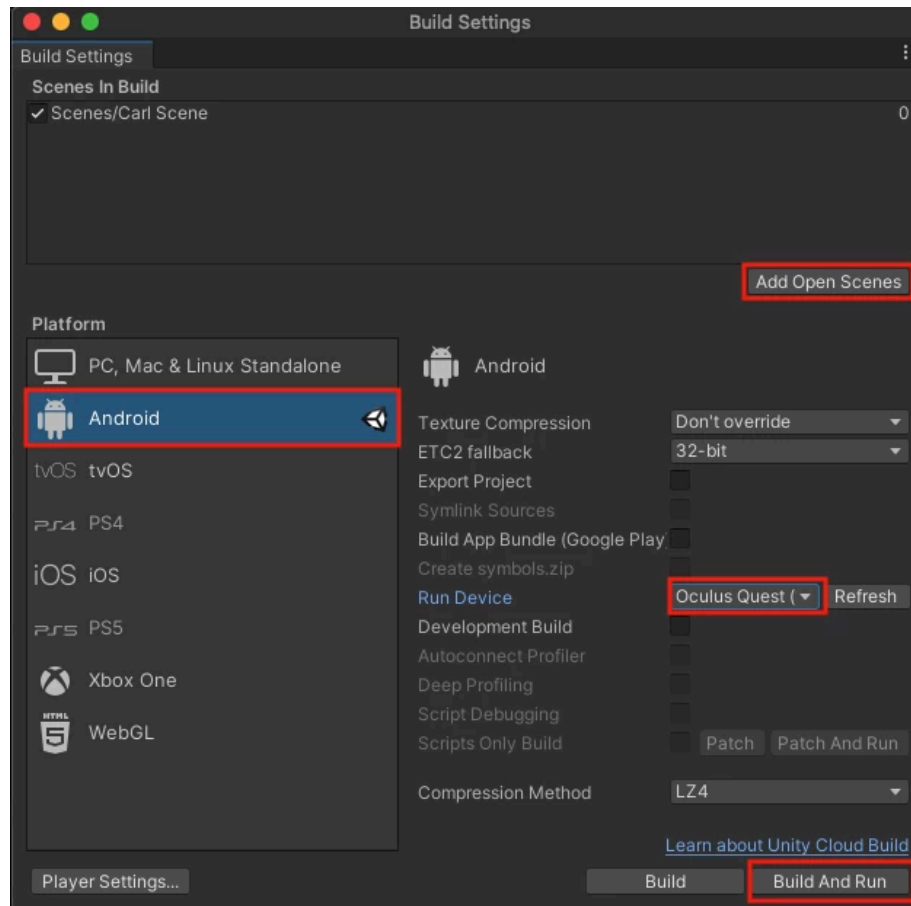
5. Select your device as the build target:

- Make sure your VR device is turned on and plugged in.
- Next to the **Run Device** drop-down, click **Refresh**, then select your VR device.
- **Note:** If your device does not show up in the list, try putting on the device. It may prompt you to **Allow USB debugging**. If it is still not showing up in the list, try restarting your device, making

sure it's in Developer Mode, or restarting Unity.

6. Build and run your project on your device:

- Click **Build and Run**.
- When prompted to choose a location, create a new "Builds" folder, then **Save** your project as "[YourName] - VR Room - 1.1".
- **Note:** When you build your app for the first time, it may take several minutes to compile.



You should now be able to test the app on your device, untethered from your computer.

Note: The app is automatically saved onto your device in the "Unknown Sources" section of your App Library. You may need to follow steps to [allow apps from unknown sources](#) in order to locate your app.

Troubleshooting tips:

- If the app runs, but it does not take up the entire field of view, it is likely a problem with plug-ins.
- If the app runs, but the entire scene moves along with your head, it is likely because you have the Device Simulator active or you do not have the Input Action Manager configured properly.

Step 7 - Test on other OpenXR-compatible devices	<p>If you are using a different OpenXR-compatible device, such as the Valve Index, HTC Vive, or a Windows Mixed Reality headset, follow this step to test on your device. Otherwise, skip this step.</p> <p>In order to test the project on one of these devices, you need to install the OpenXR plugin and enable the feature set for your specific device.</p> <ol style="list-style-type: none"> 1. Configure the OpenXR Plugin for your device: <ul style="list-style-type: none"> Follow the instructions on the OpenXR Manual page to install the OpenXR Plugin, enable it in XR Plugin Management, and configure it for your particular device. 2. Connect your device through the SteamVR app: <ul style="list-style-type: none"> Make sure your device is plugged into your computer using a compatible cable. Make sure the SteamVR app is running and has successfully recognized your device. 3. Test the project on your device: <ul style="list-style-type: none"> Make sure the Device Simulator in your scene is disabled. If it is enabled, your head tracking will not work. Select Play in the Unity Editor and put on your headset. <p>You should now be able to quickly test your app on your headset through SteamVR.</p>	
Recap	<p>New Features:</p> <ul style="list-style-type: none"> - Project is set up - Scene set up with Room - Tested with Device Simulator - Run on device <p>New Concepts & Skills:</p> <ul style="list-style-type: none"> - Pipelines for different types of hardware - Packages required for VR Development - VR scene vs typical Unity scene - Rig Simulator vs on-Device testing - Tethered vs Standalone testing <p>Next Lesson:</p> <ul style="list-style-type: none"> - VR Locomotion 	
Extensions	<p>If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below.</p> <p>Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.</p>	
Extension 1 - Directional Lighting	<p>1. Play with Directional Lighting [Easy]</p> <p>Experiment with the directional light to achieve the shadow and color you want in your room:</p> <ul style="list-style-type: none"> Try rotating the Directional Light about the X and Y axes Try adjusting the color of the Directional Light to match the color of light to the time of day 	



Extension 2 - Device Simulator Practice

2. Device Simulator Practice [Medium]

Get comfortable using the basic functions of the device simulator:

- Refer to the instructions in the step above for how to set up and use the device simulator.
- **Note:** Remember to remove or disable the Simulator before running the app in your device.



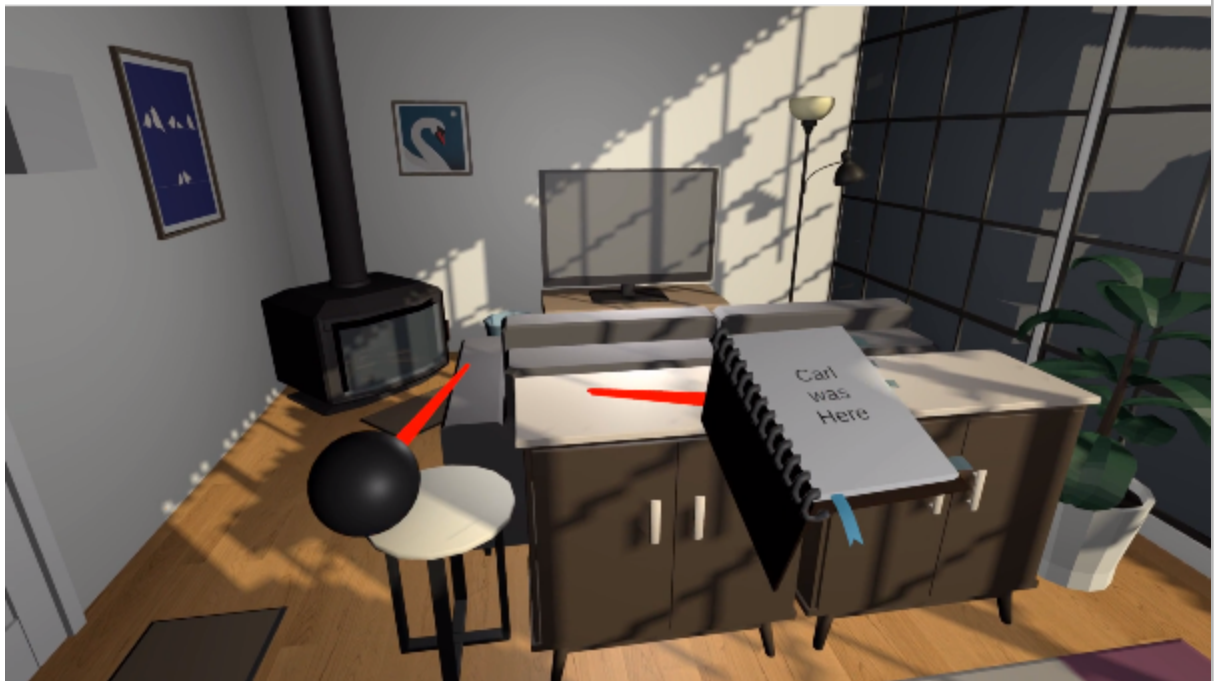
See lesson related videos [here](#)

Extension 3 - Add a functioning clock

3. Add a functioning clock [Expert] [Requires programming]


Add a clock on the wall that either (a) tells the current time for the user in their local timezone, or (b) tells the time you intend it to be your VR room world.

- Find a clock Prefab in the Course Library > Prefabs > Objects > Electronics folder.
- Hint: To access the system's current time, use the [DateTime.now](#) property.

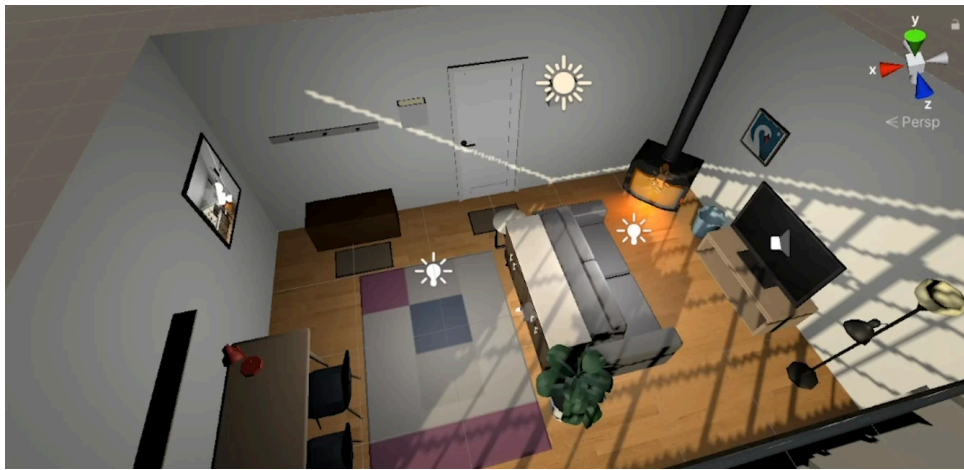


See lesson related videos [here](#)

Lesson 1.2 - VR Locomotion

Lesson Link	1.1 - VR Project Setup - Unity Learn
Length	1 Hour 30 Minutes
<p>Project Objectives: In this lesson, you will learn how to implement various types of locomotion in VR. By the end of this lesson, the user will be able to teleport around your newly furnished room to admire it from all angles.</p> <p>This lesson is part of the Create with VR course.</p> <p>Learning Objectives: By the end of this lesson, you will be able to:</p> <ul style="list-style-type: none"> Implement various techniques for locomotion in VR, including Roomscale, Teleportation, and Continuous movement. Recognize the advantages and disadvantages of different VR locomotion strategies in order to choose one that makes sense for your application. Appreciate the problem of "vection" in VR (the illusion of movement) in order to predict what might induce simulator sickness in users. 	

Step	Instructions	Teacher Notes
Step 1 - Fill the room with furniture	<p>You now have a room, but it's empty. You need to add spaces in the room where VR interactions can take place.</p> <ol style="list-style-type: none"> Locate the furniture prefabs: <ul style="list-style-type: none"> In the Project window, open Course Library > _Prefabs. Define an entryway: <ul style="list-style-type: none"> In one corner of the room, add a mirror object and hook object. Define a living area: <ul style="list-style-type: none"> In one half of the room, add a seating object, a television object, and a fireplace object. Define another area: <ul style="list-style-type: none"> Add a rug object and either a table object or storage object to provide a surface. 	<p>Warning: You might want to put the mirror on one of the side walls so that it doesn't have to render the background buildings</p> <p>Demo: Explain how the mirror works with a camera and render texture</p> <p>Warning: You can add lighting objects, but don't add any actual Unity Lights yet - we will do lighting later.</p>



You should now have: an entry area with a hook and a mirror; a seating area, a living area with a television and a fireplace; and another area with some surface space to hold objects.

Step 2 - Add snap turning

You will next set up the ability to turn around in place so that you can experience the room without physically moving and rotating.

1. Give your XR Rig locomotion capabilities:

- In the Hierarchy, select the **XR Rig** object and add a Locomotion System component.
- For the **XR Origin** property, select and drag the XR Origin component from the Inspector onto the slot that says "None".
- **Note:** You can also select the object picker and double-click the XR Rig object.

2. Allow your XR Rig to turn around:

- With the **XR Rig** object still selected, add a Snap Turn Provider (Action-based) component.
- For the **System** property, drag and drop the Locomotion System component onto the slot.

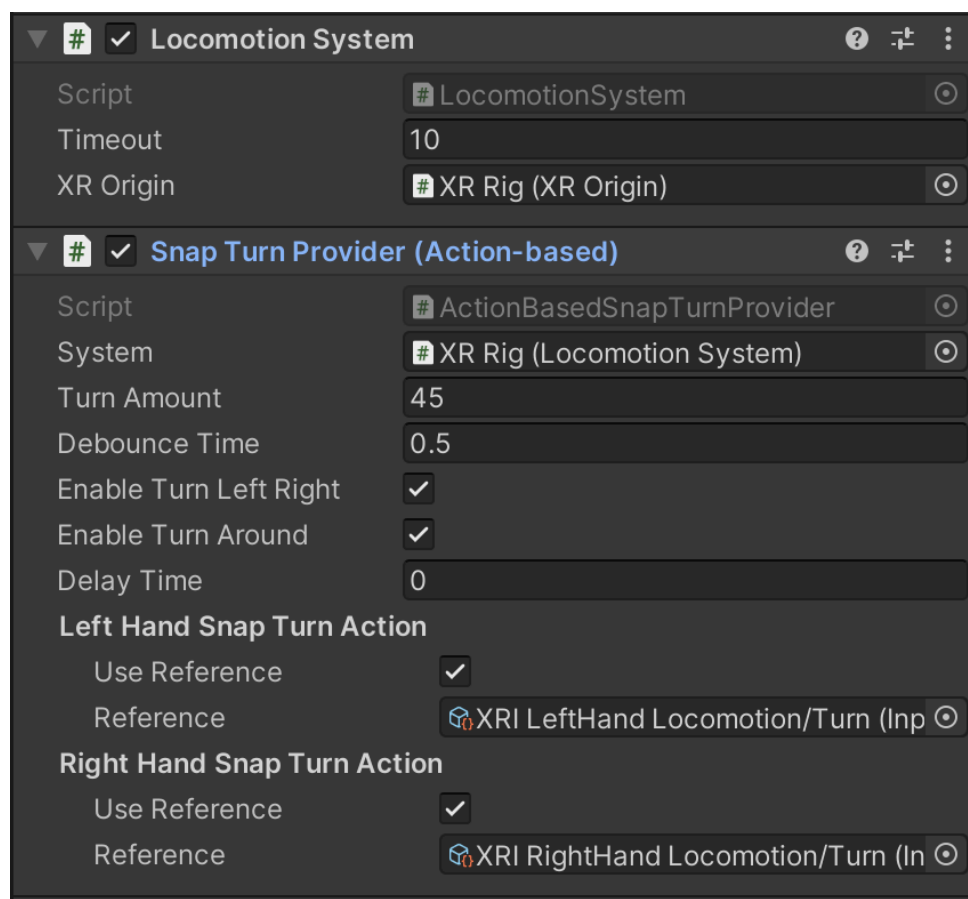
3. Fine-tune the snap turning experience:

- Experiment with the **Turn Amount** and **Debounce Time** properties.
- Select the **Enable Turn Around** property to allow the user to turn 180 degrees when they press the joystick down.

4. Experiment with continuous turning:

- Remove the Snap Turn Provider component and add a Continuous Turn Provider (Action Based) component.

Explain: What is "Action Based" vs "Device-based"? Action based uses an action-based input system which allows for device-agnostic actions to be used instead of a specific controller button. **Warning:** Continuous turning will make a significant percent of users motion sick.



You should now be able to use the joysticks on either hand controller to rotate by a certain number of degrees. This makes your project possible for a stationary experience.

Note: If you are using the XR Device simulator, you can simulate movement by using T or Y to toggle one of the controllers, then A or D to simulate snap turning.

Step 3 - Add a teleportation area

Now that you can rotate, you should also be able to teleport anywhere within a specific area.

1. Allow the user to teleport:

- Select the **XR Rig** object and add a Teleportation Provider component.
- For the **System** property, drag and drop your Locomotion System component to assign it to the property.

2. Make the rug a teleportation area:

- Select the **Rug** object and add a Teleportation Area component.
- At the bottom of the Teleportation Area component, for the **Teleportation Provider** property, drag and drop the **XR Rig** object from the Hierarchy to the empty slot to assign it to the property.

Now, when you point anywhere on the rug, the line renderer will turn white, and you can use the grip button on the controller to teleport there. The grip button is the lowermost button on the inside of the controller, usually pressed with the middle finger.

Note: If you are using the XR Device simulator, you can use T or Y to toggle one of the controllers, use the middle mouse button to aim your ray, then press G to simulate pressing the "Grab" button.

Explain: What is a teleportation area

Demo: If you double-click the XRI default input actions, you'll see the default button for "Select" is Grip

Demo: Show that there is also a continuous move provider, but that would likely make users nauseous.

Step 4 - Add teleportation anchors

Sometimes, you want the user to teleport to a specific spot, facing a specific direction.

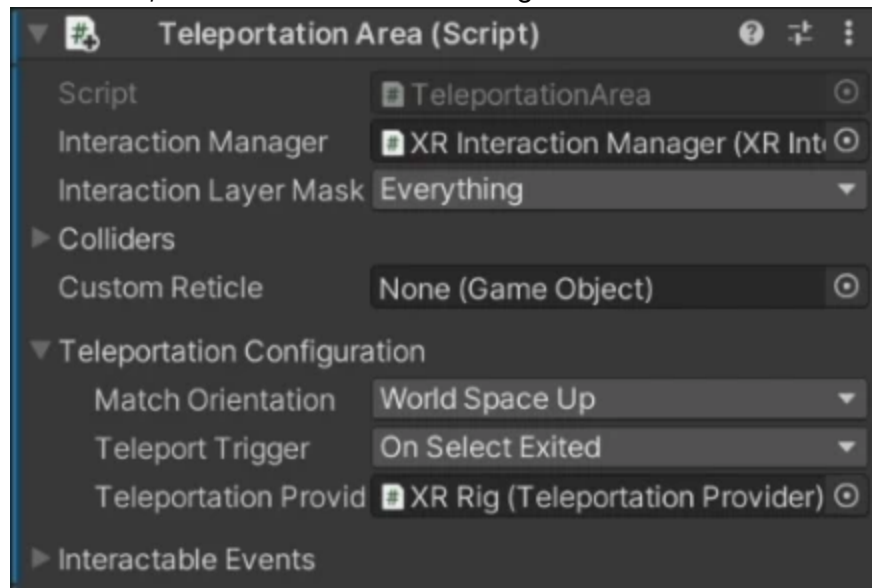
1. Add Teleportation Anchors to the scene:

- Go to **Course Library > Prefabs > Rugs**, then add smaller "Mat" objects at key locations around the room.
- Rotate these objects so that their local Z axis (blue) arrow is facing the direction you want the user to be when they teleport there.
- **Note:** You must be using **Local** coordinates to see the local forward direction of the objects.
- In Unity 2021.3 LTS, the Local/Global settings are represented by a Globe or Cube button drop-down in the top-left corner of Scene view:



2. Save changes to all of the mats at once in Prefab Mode:

- Select one of the mat objects and click the **Open** button at the top of the Inspector.
- To make sure your changes are saved in the top-right corner of the Scene view, enable the **Auto-save** setting.



3. Turn the mat prefab into a teleportation anchor:

- On the mat Prefab, add a new Teleportation Anchor component.
- **Note:** It's not actually necessary to assign the **Teleportation Provider** property - it is assigned automatically when the app runs.

4. Match the orientation of the anchor when you teleport there:

- In the Teleportation Anchor component, set the **Match Orientation** property to **Target Up And Forward**.

5. Exit Prefab Mode:

- Select the Back arrow in the top left corner of the Hierarchy.

You should now be able to teleport to the mats around the room, re-orienting to the direction of the mat when you arrive.

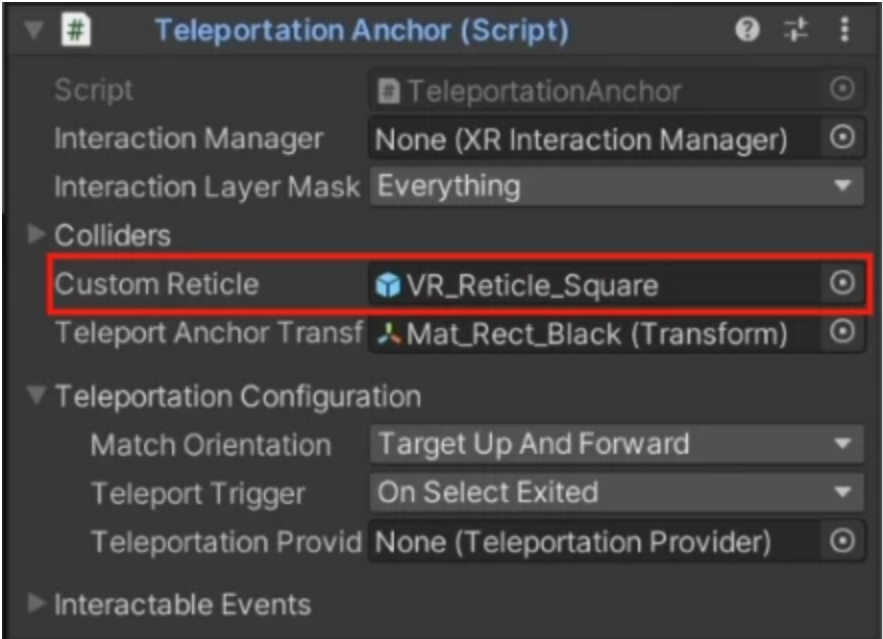
Note: To hide the colorful outlines of the objects in Scene view: in the top-right corner of the Scene view window, click the **Gizmos** button to hide the Scene view gizmos. You can also click the drop-down arrow next to the Gizmos button to show/hide specific elements.


Explain: How is a teleportation anchor different from an area?
Warning: Make sure you're using local coordinates
Demo: How to determine key locations?
Should be able to get everywhere in the room with easy access to objects.

Step 5 - Customize

The Line Renderers from your hands change color when you are able to teleport somewhere. You can provide more visual feedback to the user with

Explain: What is a reticle?

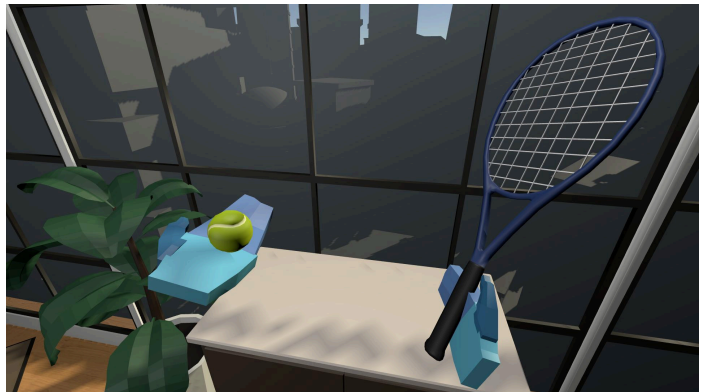
<p>the teleportation reticles</p>	<p>reticles.</p> <ol style="list-style-type: none"> Find the reticle you want to use: <ul style="list-style-type: none"> Go to Course Library > Prefabs > VR > Reticles, and determine which Reticle you prefer. Assign a custom reticle for the Teleportation Area rug: <ul style="list-style-type: none"> Select the large Rug object with a Teleportation Area component. At the bottom of the Teleportation Area component, for the Custom Reticle property, drag and drop your chosen Reticle to assign the property. Assign a custom reticle for the Teleportation Anchor mats: <ul style="list-style-type: none"> Open the prefab of one of your mat objects. In the Teleportation Anchor component, for the Custom Reticle property, assign the Reticle. Exit Prefab Mode.  <p>The custom reticle should now appear at the location where the line renderer meets the teleportation destination.</p>	<p>Basically a crosshair - something for aiming.</p> <p>Warning: Double-check that auto-save is enabled in prefab mode.</p>
<p>Recap</p>	<p>New Features:</p> <ul style="list-style-type: none"> - Designed room - Turning rig - Teleporting rig <p>New Concepts & Skills</p> <ul style="list-style-type: none"> - Types of locomotion (Room scale / Continuous / Teleport) - Vection & simulator sickness - Teleportation areas vs anchors <p>Next Lesson:</p> <ul style="list-style-type: none"> - Grabbable objects 	
<p>Extensions</p>	<p>If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.</p>	
<p>Extension - 1. Research</p>	<p>1. Research other VR movement [Easy]</p>	

other VR movement	<p>Look into other less common forms of VR movement, including projectile warping, scripted movement, projected avatar, world-pulling, etc:</p> <ul style="list-style-type: none"> • Check out this video about movement types from our Design, Develop, and Deploy for VR course • Google "VR movement types" to find other interesting strategies for VR locomotion.
Extension - 2. Create your own Art	<p>2. Create your own art [Medium]</p> <p>Add your own custom piece of art to the room to make it uniquely your own:</p> <ul style="list-style-type: none"> • You can create your own masterpiece or find one online that you have the rights to use (search for "creative commons 0") • Hint: You will need to create a new Material and assign your image as the Base Map of that material, then apply that material to an object. 
Extension - 3. Animated reticle	<p>3. Animate the reticle [Medium] [Requires Programming]</p> <p>Make the teleportation reticle spin around and/or scale up and down to make the experience feel more dynamic:</p> <ul style="list-style-type: none"> • Hint: Create a custom rotation script and apply it to the reticle prefab



See lesson related videos here: [Extension Activities Lessons](#)

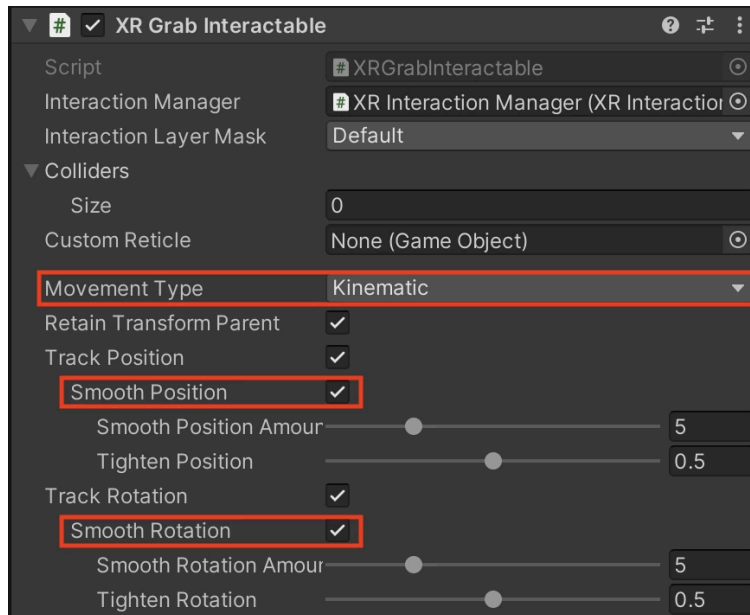
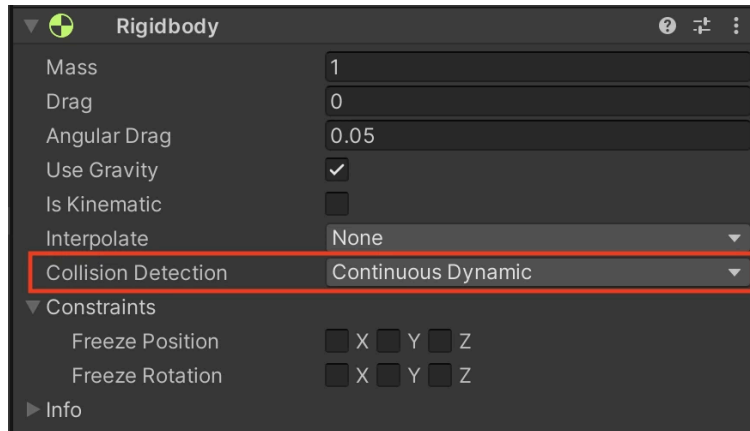
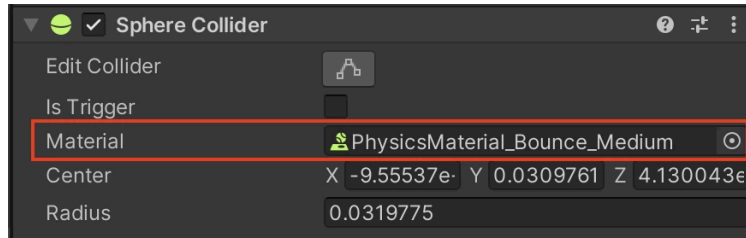
Lesson 1.3 - Grabbable Objects

Lesson Link	1.3 - Grabbable Objects - Unity Learn
Length	1 Hour 30 Minutes
Project Objective: In this lesson, you will learn how to configure objects for basic grabbable interactivity in VR. By the end of this lesson, users will be able to pick up objects in the scene and throw them around. This lesson is part of the Create with VR course . Learning Objectives: By the end of this lesson, you will be able to: <ul style="list-style-type: none">• Set up a grabbable object in VR with all required components and properties in order to be picked up and manipulated by a user.• Configure a grabbable object's attach point in order to ensure it is positioned and oriented properly when it is picked up by a user.• Use the relationship between units in Unity and real-world scale in order to ensure that objects in the scene are proportioned accurately.• Understand the properties of a grabbable object in order to ensure it behaves as desired in VR (e.g. movement type, position smoothing, etc).	

Step	Instructions	Teacher Notes
Step 1 - Choose hand models	<p>You can choose the models to represent the user's hands in VR rather than just using simple spheres.</p> <ol style="list-style-type: none">1. Browse options for hand models:<ul style="list-style-type: none">• In the Project window, open Course Library > Prefabs > VR > Hands, then determine which VR_Hand option you prefer.2. Assign a model to your left hand:<ul style="list-style-type: none">• In the Hierarchy, select XR Rig > Camera Offset > LeftHand Controller.• In the XR Controller Component, for the Model Prefab property, drag and drop your hand prefab of choice to assign it.3. Assign a model to your right hand:<ul style="list-style-type: none">• In the Hierarchy, select XR Rig > Camera Offset > RightHand Controller• Repeat the same step as above to assign a prefab for your right hand. <p>When you test your app, you should now see the hand models you selected instead of grey spheres.</p>	<p>Explain: These prefabs are instantiated when the app is run.</p> <p>Warning: Make sure you assign the correct left vs right hand prefabs to the left and right controllers</p>

Step 2 - Add a grabbable object	<p>Now that you can see your VR hands, you should add objects to grab with them.</p> <ol style="list-style-type: none"> 1. Add a ball to the scene: <ul style="list-style-type: none"> Go to Course Library > Prefabs > Objects > Sports. Drag a ball onto an accessible surface in the room. 2. Make the ball grabbable: <ul style="list-style-type: none"> Add an XR Grab Interactable component to the ball. Note: This will automatically add a Rigidbody component to the ball object as well. 3. Test your grabbable object: <ul style="list-style-type: none"> Reposition the XR Rig to be in front of the table with the grabbable ball object. Run your application. Test grabbing the object by aiming your ray at it and pressing the Grip button. 4. Make sure that the object is the correct scale: <ul style="list-style-type: none"> In the Hierarchy, create a new 3D Object > Cube object, and rename it "Measuring Stick". Use Google to determine the correct official diameter of the ball and then convert the dimensions you found into meters. In the Measuring Stick's Transform component, set the X and Z Scale to 0.01 and the Y scale to the length you calculated. Position the measuring stick next to your ball to make sure it's accurately sized. If it is not, use the Scale tool to correct it. <p>You should now be able to pick up the ball using the Grip button, release the grip button to throw it.</p> <p>Note: If you are using the XR Device simulator, you can toggle a controller with T or Y, rotate it to point at an object by holding the middle mouse button, then press and hold G to grab and hold an object.</p>	<p>Warning: It may go through the ground - that's OK.</p> <p>Warning: It will be a little weird to have the ball intersecting with your hand - we'll fix that.</p> <p>Demo: You can move the object towards and away from yourself with the joystick - we'll fix that.</p> <p>Explain: Unity to real-world scale - 1m = 1 Unity unit.</p>
Step 3 - Hide hands and disable anchor control	<p>Currently, the hand model overlaps the object you're holding and you are able to change that object's position in your hand with the controller. This is not optimal for interacting with the scene.</p> <ol style="list-style-type: none"> 1. Stop the user from moving the object with the left joystick: <ul style="list-style-type: none"> In the Hierarchy, select XR Rig > Camera Offset > LeftHand Controller object. In the XR Ray Interactor component, disable the Anchor Control setting. 2. Make your hand model disappear when you grab an object: <ul style="list-style-type: none"> In the XR Ray Interactor component, enable the Hide Controller on Select setting. 3. Apply these changes to your right controller: <ul style="list-style-type: none"> Repeat the steps above on the RightHand Controller object. <p>Your hands should now disappear when you pick up an object and you should no longer be able to move or rotate the object with the joystick.</p>	<p>Explain: Why would you want your hands to disappear? It just looks weird to have your hand intersecting with the object and it looks surprisingly OK to not have them there.</p> <p>Explain: When would you want to be able to move / rotate objects? Maybe when examining them, or in a creative / design app where you can position things.</p>
Step 4 - Fine-tune the throwing experience	<p>The object can be grabbed, but there are some minor aspects of its behavior that need to be tweaked.</p> <ol style="list-style-type: none"> 1. Make the ball bouncy: <ul style="list-style-type: none"> In the Ball object's Sphere Collider component, locate the Material property. Click the circular button to assign one of the provided Physics Materials. 2. Prevent a dropped object from going through the floor: <ul style="list-style-type: none"> In the Ball object's Rigidbody component, set the Collision 	<p>Explain: Kinematic vs velocity tracking vs Instantaneous movement</p> <p>Explain: What is the difference between the "Is kinematic" setting in the Rigidbody and the</p>

- Detection** setting to **Continuous Dynamic**.
- 3. Allow the ball's rigidbody physics to work while in your hand:**
 - In the XR Grab Interactable component, ensure that the **Movement Type** is set to **Kinematic**.
 - 4. Smooth the motion of the object:**
 - In the XR Grab Interactable component, enable the **Smooth Position** and **Smooth Rotation** properties.



The ball should now bounce, not fall through the floor, and move smoothly in your hand.

Kinematic movement type?
Explain: Go through other XR Grab interactable settings, including Attach Transform, Ease in time, Movement type, Interaction LayerMask, Smooth position/rotation, Throw on detach / etc
Demo: Tooltips are super helpful in inspector

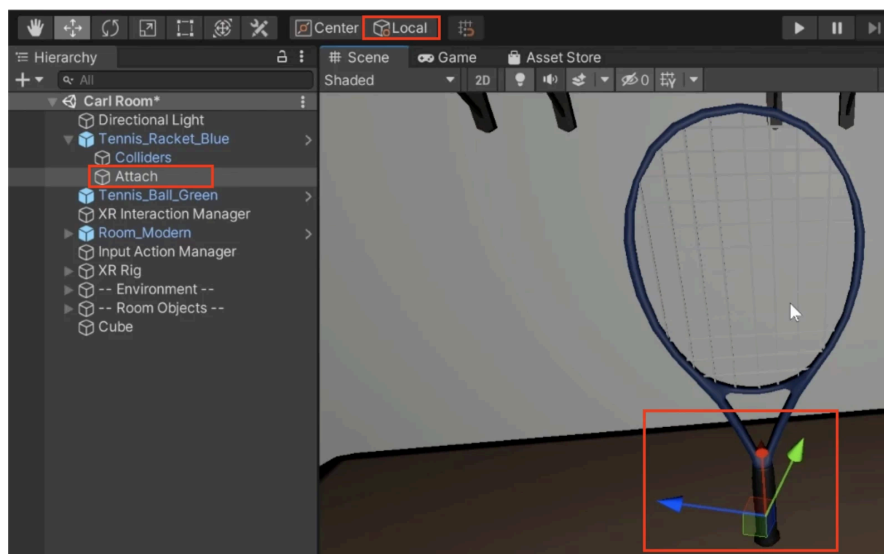
Step 5 - Add an object with a handle

- You have added a ball object that is held at its center and where orientation is not important. Now you will add an object that will be grabbed at a very particular point and orientation.
- 1. Add a sports racket, paddle, or bat into the scene:**
 - Go to **Course Library > Prefabs > Sports**

Explain: Imagine making a fist with a "thumbs up" gesture - the front of your knuckle should align

- Drag one of the sports implements onto the surface next to the ball.
- 2. Make the new object grabbable:**
 - Select the new object and add an XR Grab Interactable component.
 - In the XR Grabbable component, enable both the **Smooth Position** setting and the **Smooth Rotation** properties to decrease jitter.
 - In the Rigidbody component (which was added automatically), for the **Collision Detection** option, select **Continuous Dynamic** to prevent it from going through the floor.
 - 3. Create a specific attach point for the object:**
 - In the Hierarchy, right-click the object and create an **Empty** child object.
 - Rename this empty object "Attach".
 - Reposition and rotate the Attach object so that it matches the position and orientation your hand model should be when you grab the new object.
 - 4. Assign the attach point to your sports implement:**
 - In the Hierarchy, re-select the parent sports object.
 - In the XR Grab Interactable component, locate the **Attach Transform** property.
 - Drag and drop your new Attach object to assign it to the Attach Transform property.

with the z axis arrow your thumb should align with the y axis
Warning: This can be extremely counter-intuitive and may need to be done with trial and error



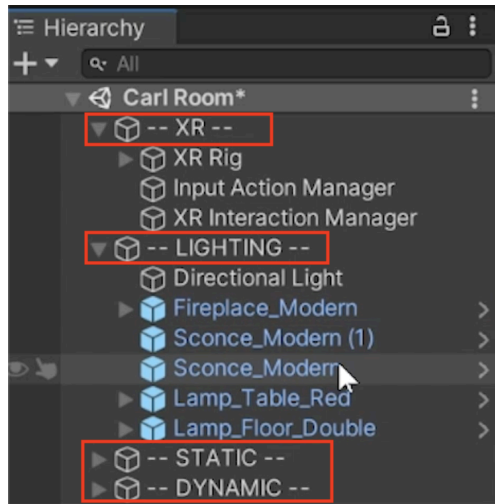
You should now be able to grab the sports implement object by the handle and hit the ball with it.

Step 6 - Organize the Hierarchy

- Before you get too many objects in the Hierarchy, it's good to set up an organizational system for your objects.
- 1. Make organizer objects in your Hierarchy:**
 - In the Hierarchy, create new empty GameObjects named "XR," "LIGHTING," "STATIC," and "DYNAMIC,"
 - Reset all of their positions to 0, 0 0.
 - 2. Organize your objects into categories:**
 - Drag objects from your Hierarchy onto the organizer objects - the advantage of this technique is that it keeps your Hierarchy nice and neat.

Explain: This will be very helpful down the line - especially with lighting
Demo: Use Ctrl/Cmd+Click to multi-select or Shift-click to select many objects in a row.

- Alternatively, drag your objects underneath these organizer objects - the advantage of this technique is that it keeps your objects easily accessible.



The Hierarchy should now be more usable, organized into categories.

Recap

New Features:

- Hand models
- Simple grabbable ball
- Grabbable tool with attach point
- Organized Hierarchy

New Concepts & Skills

- Unity vs Real-world scale
- Grabbable object properties
- Collision detection modes (discrete vs continuous dynamic)
- Movement types (kinematic, instantaneous, velocity-tracking)

Next Lesson:

- Sockets

Extensions

If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.

Extension - Fill your room with objects

1. Fill your room with objects [Easy]

Add fun decorative and grabbable objects to your scene to make it feel more interactive, even if those objects aren't fully functional yet:

- Browse through the Course Library > Prefabs > Objects folders for ideas.



Extension - Experiment with the Grab Interactable properties

2. Experiment with the grab interactable properties [Easy]

Add a new grabbable object to your scene and experiment with the various properties of the XR Grab Interactable component, including:

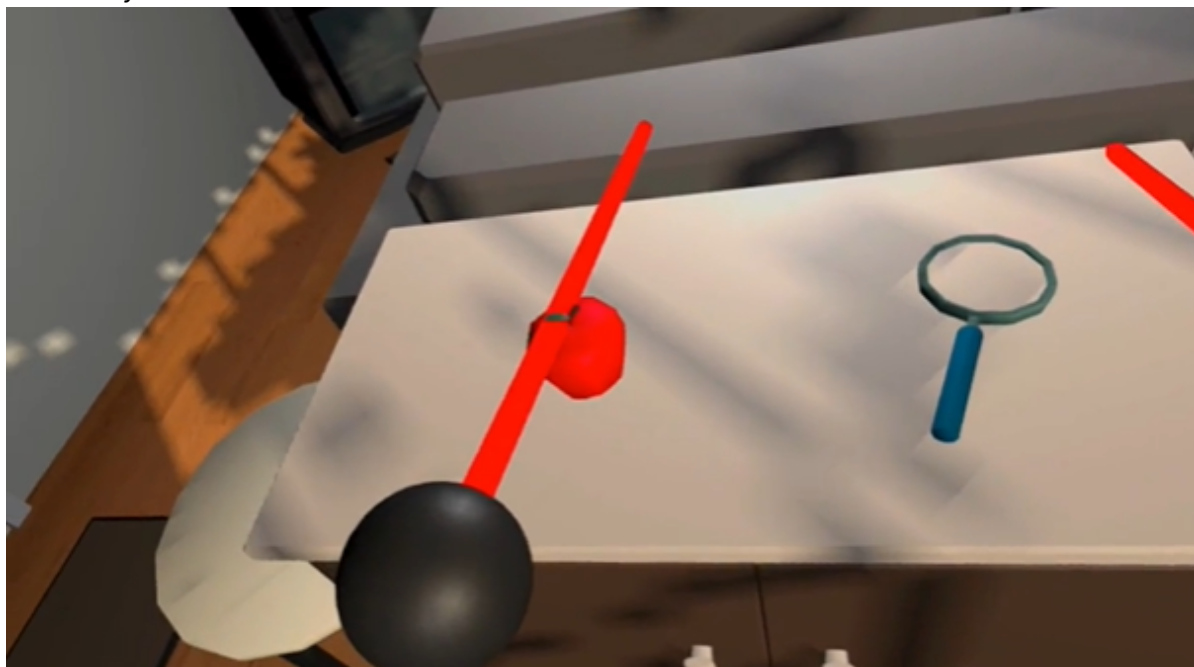
- Movement type: Kinematic vs Instantaneous vs Velocity Tracking
- Smooth Position/Rotation Amount and Tighten Position
- Throw on Detach properties

Extension - Magnifying Glass

3. Add a magnifying glass [Difficult]

Add a functional magnifying glass to your scene that you can use to pick up and examine objects:

- You could use a Camera Render Texture similar to the way the mirror works.
- Don't worry if the ray gets in the way - that will be resolved when we can toggle the rays on and off.



See lesson related videos [here](#)

Extension - Add a notebook that swings open

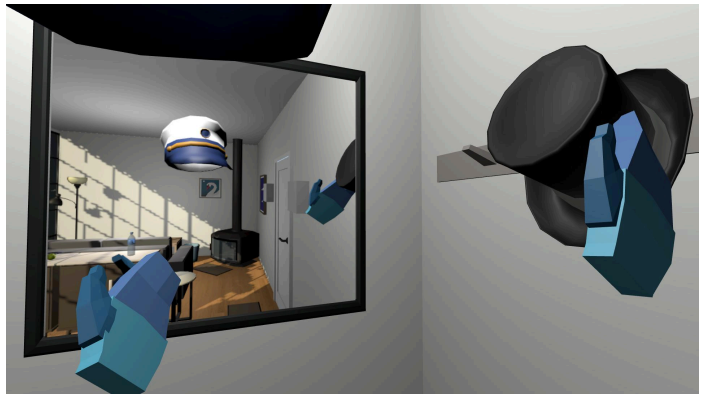
4. Add a notebook that swings open [Expert]

Add a notebook with a front cover that swings open when you grab it:

- Hint: The Notebook_Cover needs a [hinge joint](#) with its **Connected Body** property set to the notebook's Rigidbody.
- If you want, try adding some text to the notebook.

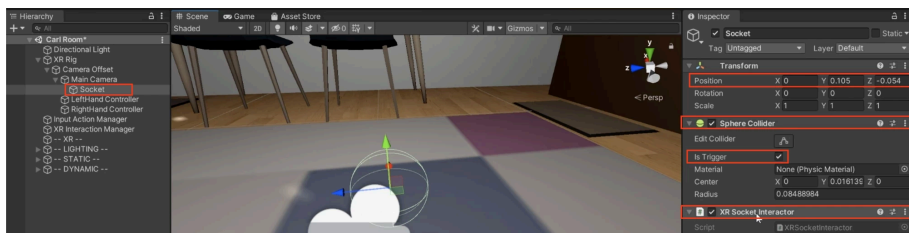
	See lesson related videos here
--	--

Lesson 1.4 - Sockets

Lesson Link	1.4 - Sockets - Unity Learn
Length	1 Hour 30 Minutes
Project Objective:	
In this lesson, you will learn how to configure sockets that objects can snap into in VR. By the end of this lesson, users will be able to hang hats up on hooks - and even wear the hats too!	
This lesson is part of the Create with VR course .	
Learning Objectives:	
By the end of this lesson, you will be able to:	
<ul style="list-style-type: none"> Describe the functionality of a socket in VR in order to propose potential use cases in an application. Configure a socket in order to have an object snap into a particular orientation if placed within a specific area. Use interaction layers in order to control which objects can and cannot interact with each other. 	

Step	Instructions	Teacher Notes
Step 1 - Add grabbable hats to the scene	<p>Before making a functional hook for hats, you'll first need some grabbable hats.</p> <ol style="list-style-type: none"> Some hats to your scene: <ul style="list-style-type: none"> Open the Course Library > Prefabs > Objects > Hats folder. Drag two hats you might want to wear onto a surface near the mirror. Note: If you do not already have a set of hooks in your scene, add one from the Course Library > Prefabs > Hooks folder. Make the hats obey physics: <ul style="list-style-type: none"> Use Ctrl or Cmd to select both hats in the Hierarchy. Add Rigidbody components to the selected hat objects. Set the Collision Detection property to Continuous Dynamic to prevent them from falling through the ground. Make the hats grabbable: <ul style="list-style-type: none"> Add XR Grab Interactable components to each hat. If you want, enable the Smooth Position and Smooth Rotation properties to reduce jitter. Prepare for testing: <ul style="list-style-type: none"> Reposition and reorient the XR Rig to be in front of the mirror within reaching distance of the hooks and the hats. <p>You should now be able to successfully pick up the hats.</p>	<p>Explain: You already know how to do all this - reminder that all you need for an object to behave like a grabbable object is a collider, rigidbody, and xr grab interactable component</p> <p>Warning: Don't get emotionally attached to the mirror - it's really bad for optimization and can only be kept if our app is well optimized - in fact, if your app already feels laggy, try disabling it and see how if that helps.</p>
Step 2 - Turn the hook into a socket	<p>To be able to hang the hat on a hook, the hook will need to behave like a socket.</p> <ol style="list-style-type: none"> Position the socket correctly: <ul style="list-style-type: none"> In the Hierarchy, right-click the Hook object and create an Empty child object. 	<p>Explain: What is a socket?</p> <p>Warning: Remember to check the "Is Trigger" checkbox</p>

	<ul style="list-style-type: none"> • Rename the child object "Socket 1". • Reposition the Socket 1 object to the tip of the hook, where the hat will rest. <p>2. Define the trigger detection area for the socket:</p> <ul style="list-style-type: none"> • Add a Sphere Collider component to the Socket object. • Click the Edit Collider button and reduce the Radius to 0.1. • Select the Is Trigger check box to prevent it from colliding with the hat. <p>3. Make the Socket 1 object behave like a socket:</p> <ul style="list-style-type: none"> • Add an XR Socket Interactor component to the object. <p>4. Make the hat point downwards when attached to the socket:</p> <ul style="list-style-type: none"> • Create an empty child object of Socket 1 and rename this child object "Attach". • Rotate the Attach object to face downward. • Assign the Attach object to the Attach Transform property of the Socket Interactor component. • Note: The local Z axis of the attach point should align with the desired forward direction of the hat and the local Y axis should align with its desired upward direction. <p>You should now be able to pick up and place any of your hats onto the socket you set up.</p>	<p>so that it can overlap with other objects.</p> <p>Demo: The Hover Mesh Material property - by default is assigned Blue and Red at runtime</p>
Step 3 - Duplicate the sockets and attach hats	<p>Now that you have one fully functional socket, you can duplicate it for the other hanging locations on the hook and have the hats start on hooks when the scene loads.</p> <p>1. Create sockets for all the hanging locations:</p> <ul style="list-style-type: none"> • In the Hierarchy, duplicate the Socket 1 object two times. • Rename the copied objects to "Socket 2" and "Socket 3". • Reposition the copied sockets for the other two hanging hook locations. <p>2. Have your hats start on the hooks by default:</p> <ul style="list-style-type: none"> • In each of the Socket object's XR Socket Interactor components, assign one of your hats to the Starting Selected Interactable property. • Note: If you only have 2 hats, one of the sockets should have "None" for its starting interactable property. <p>All 3 of your sockets should now be fully functional, with hats starting on the hooks at the start of the scene.</p>	<p>Demo: Might want to use isometric mode for easy positioning of the hooks</p>
Step 4 - Put hats on your head	<p>These hats and this mirror would really be wasted unless you could actually wear the hats on your head.</p> <p>1. Position the head socket object correctly:</p> <ul style="list-style-type: none"> • In the Hierarchy, right-click XR Rig > Camera Offset > Main Camera and create an Empty child object. • Rename this new child object "Socket" • Set the Y Position of the Socket object to 0.1 so that the hat appears above your eyes. <p>2. Add a trigger detection area for the socket:</p> <ul style="list-style-type: none"> • Add a Sphere Collider component to the Socket object. • Click the Edit Collider button and set the Radius property to 0.2. • Select the Is Trigger check box. <p>3. Allow the hats to snap to this socket:</p> <ul style="list-style-type: none"> • Add an XR Socket Interactor component. 	<p>Explain: The head/camera is on the ground by default - it gets pushed up when you enter the scene and your position is tracked.</p> <p>Explain: The distance from your eyes to where your hat should be centered is probably around 1/10th of a meter (0.1 units).</p>



You should now be able to put the hats on your head and see how they look in the mirror.

Step 5 - Prevent other objects from acting like hats

You can currently put all objects on your head or on the hooks as if they were hats. You can use Interaction layers to control which objects can and cannot interact with each other.

1. Add a new layer for "Hats":

- Select a hat object.
- In the XR Grabbable Interactable component, from the **Interaction Layer Mask** drop-down, select Add Layer.
- Add a new "Hats" layer in the first available empty slot.
- **Note:** Your hat will not yet be set to that layer. You have only created the layer at this point.

2. Put the hats on your new Hats layer:

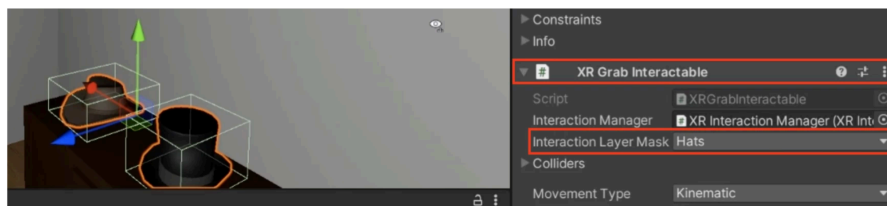
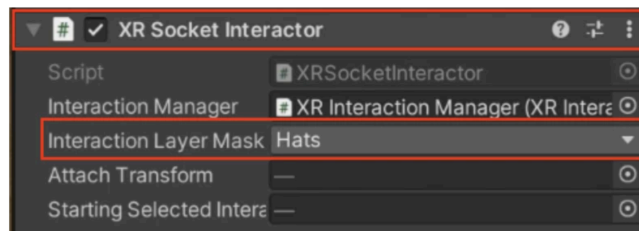
- In the Hierarchy, select all of your **Hat** objects using Ctrl/Cmd + Click.
- In the XR Grab Interactable component, from the **Interaction Layer Mask** drop-down, select **Nothing** to clear the layers.
- From the drop-down, select your new **Hats** layer to enable only that layer.

3. Make sure the hook and head sockets can only interact with objects on the "Hats" layer:


- Set each of the hook and head socket objects' **Interaction Layer Masks** property to only the **Hats** layer.

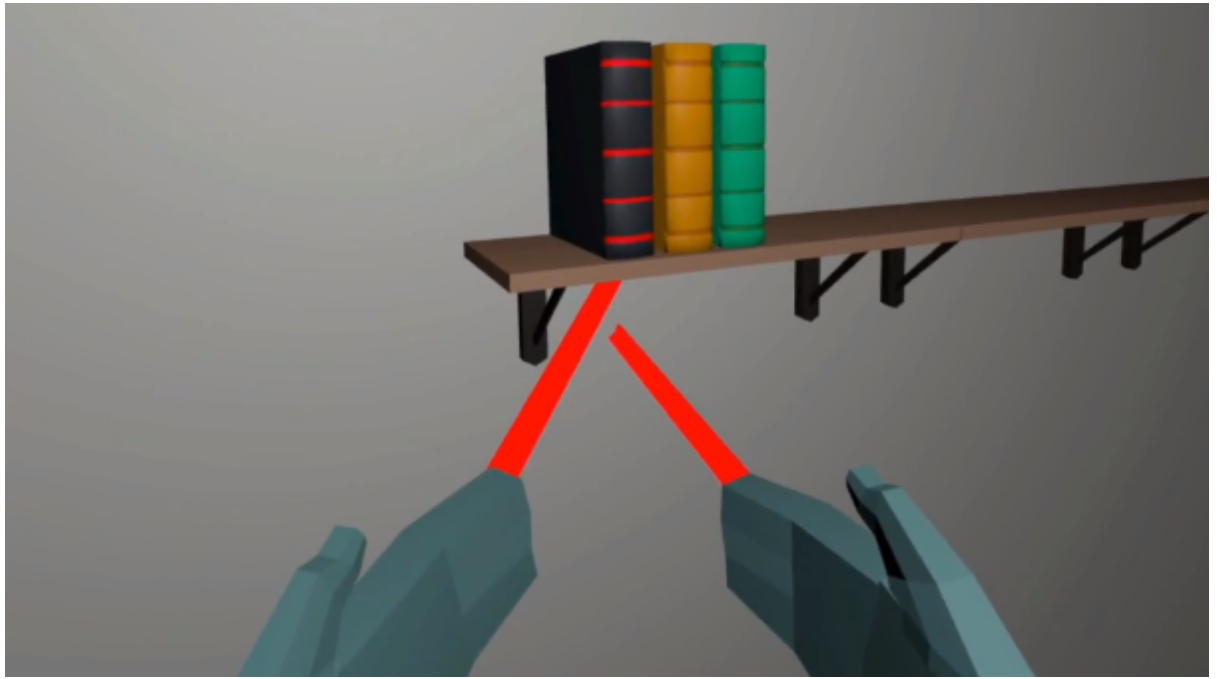
4. Make sure other objects cannot be used as hats:

- Select all of your other grabbable objects.
- From the **Interaction Layer Mask** drop-down, make sure only the **Default** layer is enabled.
- This will prevent them from interacting with any sockets on the **Hats** layer.




Explain: How do layers / interaction layers work? If you don't want things to interact, can't share ANY layers

	You should now only be able to put hats - and not any other grabbable objects - on your head and the hooks.	
Recap	<p>New Features:</p> <ul style="list-style-type: none"> - Added hats - Hang hats on hooks - Put hats on your head <p>New Concepts & Skills</p> <ul style="list-style-type: none"> - Socket interactors - Triggers to detect interaction - Interaction layer masks <p>Next Lesson:</p> <ul style="list-style-type: none"> - Audio and Haptics 	
Extensions	<p>If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.</p>	
Extension - Rearrangeable art	<p>1. Add rearrangeable art [Medium]</p> <p>Allow the user to swap the position of art hanging on the wall:</p> <ul style="list-style-type: none"> • To make a simple hook, create an empty object with a Socket Interactor component, then create a primitive cube child object, scaled and rotated appropriately, to represent the mesh. • To make the pieces of art hang at the correct height, you may need to add an Attach Transform located at the position on the frame the art would hang from.  <p>See lesson related videos here</p>	
Extension - Rearrangeable Bookshelf	<p>2. Add rearrangeable books [Difficult]</p> <p>Add books to a bookshelf that can be rearranged neatly in any order the user wants:</p> <ul style="list-style-type: none"> • In order to get the books to appear at the orientation you want, try creating an empty object with the correct orientation, then making the book object a child object. 	



See lesson related videos [here](#)

Challenge 1 - Architecture Review App


Lesson Link	Challenge 1 - Architecture Review - Unity Learn
Length	1 Hour
<p>In this challenge, you'll apply the skills you learned while making your VR Room in an Architecture review app. In this prototype, the user can examine a building at real-world scale, inspect a miniature 3d model and floor plan of that building, and even use a ruler to take measurements as they look around.</p> <p>This challenge will assess skills learned in the following lessons:</p> <ul style="list-style-type: none"> • VR locomotion • Grabbable objects • Sockets <p>This challenge is part of the Create with VR course.</p>	
	

Getting Started	<ol style="list-style-type: none"> 1. Open the broken architecture review prototype Scene: <ul style="list-style-type: none"> From the Project window, expand, Assets > Challenges > 01_Architecture > Scenes Double-click on the Architecture_Prototype_Broken Scene to open it. 2. Begin working on the challenge tasks: <ul style="list-style-type: none"> Work through the tasks outlined in the steps below. If you want to push your skills, attempt the optional bonus challenge tasks, as well. If you get stuck, there are hints for each task at the bottom of the page.
-----------------	---

Challenge	Task	Hint
1. The measuring stick and clipboard aren't attaching properly to the left wrist.	<ul style="list-style-type: none"> The clipboard should be flipped around and the measuring stick should snap to the spot just to the left of the clipboard. 	1. Look at the position and rotation of the Attach Transforms for your left hand sockets.
2. The house model is snapping to the pedestal from very far away.	<ul style="list-style-type: none"> The house model should snap to the pedestal when you place it on top of the pedestal. 	2. Look at the size of the Sphere Collider on the pedestal socket.
3. You can currently snap the ruler and clipboard to the pedestal and the house model the left wrist.	<ul style="list-style-type: none"> You should only be able to put the house model on the pedestal, while the clipboard and measuring stick should only be able to snap to the wrist. 	3. Look at the Interaction Layer Masks of the sockets and grabbable objects.
4. The dining table is too tall.	<ul style="list-style-type: none"> The table should be exactly 0.74 meters (or 29 inches) high. 	4. Try creating a measuring stick that is exactly the right height and putting it next to the table.

5. The teleportation anchor next to the pedestal puts the player upside-down and facing away from the pedestal.	<ul style="list-style-type: none"> Teleporting here should orient the player upright and towards the pedestal. 	5. Look at the local orientation of the anchor as well as the "Match Orientation" setting in its Teleportation Anchor component.
Bonus tasks		
6. The ray on your right hand uses all the default settings.	<ul style="list-style-type: none"> Change the shape of your ray to a smooth bezier curve to make teleporting easier and edit the valid and invalid colors to fade from transparent to opaque. 	6. Look in the XR Ray Interactor component to edit the line's shape and the XR Interactor Line Visual component to edit its colors.
7. The scene should have some ambient sound.	<ul style="list-style-type: none"> Add an ambient wind sound in the background and randomized bird sounds that seem to be coming from different locations around the house. 	7. Make sure your audio source is set to the 3D "Spatial Blend," then, if you want, use the OnTimeInterval and PlaySoundsFromList components to randomize the bird sounds.

Lab 1 - VR Personal Project Basics

Lesson Link	Lab 1 - VR Personal Project Basics - Unity Learn	
Length	2 Hours	
<p>In this first lab, you'll fill out a design document to lay out your concept, and then set up a basic VR scene using simple primitive shapes.</p> <p>This lab will draw on skills learned in the following lessons:</p> <ul style="list-style-type: none"> • VR Project Setup • VR locomotion • Grabbable objects • Sockets <p>This lab is part of the Create with VR course.</p>		

Step	Instructions	Teacher Notes
Step 1 - Fill out your design document	<p>Before you get working on your project, you need a plan for what you're actually going to create. The best way to do that is with a Design Document.</p> <ol style="list-style-type: none"> Get a blank copy of the VR Project Design Doc <ul style="list-style-type: none"> Click here to create a copy of a Google Doc Click here to download as a Word document Click here to download as a PDF Fill out Section 1 (App Info) of the document: <ul style="list-style-type: none"> Give your app a tentative title Place a checkmark (using copy and paste) next to the one or two categories that best describe your app. Fill out Section 2 (Pitch) of the document: <ul style="list-style-type: none"> Describe the goal of the app. Describe why it will be especially effective in VR. Describe what users will do during the app, at a high level. Specify whether your app will target devices with 3 or 6 degrees of Freedom Fill out Section 3 (Basics) of the document: <ul style="list-style-type: none"> Describe where the app will take place and how users will get around the environment. List the types of objects users will be able to grab and any socket interactors you will likely use. <p>You should now have the first 3 sections of your Design Document complete, providing enough direction to complete the rest of this lab.</p>	<p>Demo: Even though we're just filling out Sections 1-3, give them a quick tour of the whole document.</p> <p>Warning: Keep your concept simple!</p> <p>Explain: How will we use this design document throughout the process</p> <p>Warning: Don't do what the teacher does - this is a chance to come up with your own ideas.</p>
Step 2 - Open and	Just like you did at the start of the VR Room project, the first	Explain: Doing the

run a new VR project	<p>thing you have to do for your personal projects is create a new Unity project configured for VR.</p> <ol style="list-style-type: none"> Create a new Unity project for your personal project: <ul style="list-style-type: none"> Follow the instructions in the VR Project Setup tutorial. Move the extracted folder to the same directory as your VR Room project. Instead of naming your project "VR Room - [Your Name]" - name it "VR Personal Project - [Your Name]". Import the VR Course Library into your project: <ul style="list-style-type: none"> Follow the instructions in the VR Project Setup tutorial. Configure your project to run on your specific device: <ul style="list-style-type: none"> Follow the instructions in the VR Project Setup tutorial. Make sure you are ready to start developing your personal project: <ul style="list-style-type: none"> Rename and open the Starter Scene. Make sure you can successfully run your project on your VR device or with the Device Simulator. If you do not want the Challenges in your project: <ul style="list-style-type: none"> In the Project Window, right-click the Challenges folder and click Delete. <p>You should now have a new project open to a starter scene and be able to test it in VR or with the Device Simulator.</p>	<p>same thing you did with your room, but just naming things differently</p> <p>Explain: Why do you need the Course Library? There are scripts that will be helpful and it's sometimes useful to have assets as well.</p>
Step 3 - Set up the basic scene with primitives	<p>Rather than using assets from the course library, you can develop this basic scene using primitives and simple colored materials:</p> <ol style="list-style-type: none"> Create the ground: <ul style="list-style-type: none"> In the Hierarchy, select the Plane object. Reset the Plane's position to 0, 0, 0 in the Inspector. Rename it "Ground". Keep your materials organized: <ul style="list-style-type: none"> Create a new "Materials" folder inside the Assets directory. Create a material for the Ground: <ul style="list-style-type: none"> In the Materials folder, right-click > Create > Material. Rename the new material "Ground". Edit the Base Map color in the inspector. Drag the material onto your Ground object. Block out the rest of your basic scene: <ul style="list-style-type: none"> Continue creating and scaling additional primitive objects (cubes, spheres, planes, capsules, etc) and applying simple colored materials. Do not create grabbable objects yet - only critical environmental things like a room, a table, etc <p>Note: If you have objects that will be reused throughout the scene, you should create a "Prefabs" folder and drag those objects into it to make them prefabs.</p> <p>You should now have your basic scene blocked out with primitives at an appropriate scale.</p>	<p>Warning - Don't add grabbable objects yet - we'll do that in a bit.</p> <p>Explain: Why use primitives?</p> <p>Explain: Plane by default is 10m (units) x 10m, cube / sphere by default are 1, capsule is 2m</p> <p>Demo: How to use prefabs the use of prefabs, probably</p> <p>Warning: Keep an eye out for scale here!</p> <p>Demo: For more complex shapes, use empty objects and make primitives children of those - then probably make prefabs (e.g. trees)</p>
Step 4 - Set up basic locomotion	<p>Now that you have the basic scene blocked out, you need some way of getting around it.</p> <ol style="list-style-type: none"> Allow the user to move around the scene: <ul style="list-style-type: none"> Follow the instructions in the VR Locomotion tutorial to set up Teleportation Areas. It may be helpful to temporarily make your entire ground object a Teleportation Area for testing and development purposes. 	<p>Explain: Even if you don't intend on having any movement, it's kind of helpful to be able to teleport around, so err on the side of being able to teleport more.</p>


	<ul style="list-style-type: none"> Alternatively, it may be helpful to use continuous movement for testing and development purposes. <p>2. Allow the user to teleport to important locations in the scene:</p> <ul style="list-style-type: none"> Follow the instructions in the VR Locomotion tutorial to set up Teleportation Anchors. Instead of using the mats like you did in your Room, create 1 x 0.1 x 1 cubes named "Teleportation Anchor" Remember to set the Match Orientation property to match the anchor's forward direction. <p>You should now be able to move around your basic scene.</p>	<p>Explain: Probably good to use a prefab for your teleportation anchors.</p>
<p>Step 5 - Set up basic grabbable objects</p>	<p>The last thing you'll do is create any grabbable object prefabs you expect the user to interact with.</p> <p>1. To make more complex objects with primitives:</p> <ul style="list-style-type: none"> Create an Empty object. Rename the object. Create Primitive child objects to represent the mesh of the form of the object. <p>2. To make your objects grabbable:</p> <ul style="list-style-type: none"> Follow the instructions in the Grabbable Objects tutorial. Add components on the Empty parent object, not the primitive child objects. <p>3. To add sockets to your scene:</p> <ul style="list-style-type: none"> Follow the instructions in the Sockets tutorial. <p>4. To make your interactable objects into Prefabs:</p> <ul style="list-style-type: none"> Drag each new interactable object into the Prefabs folder. <p>You should now have basic functioning prefabs for the main grabbable objects in your scene.</p>	<p>Warning: Don't go crazy designing primitives - should just be representations of these objects - should just be enough so you know what it is..</p> <p>Demo: How to look up scale online and apply that scale - scale is critical for objects you'll hold in your hand</p> <p>Warning: Make sure you're adjusting the scale of the primitive child objects- NOT the parent object</p> <p>Explain Even if it kinda works without the attach point, it's good to have the control in case you need to adjust it down the line.</p>
<p>Step 6 - Organize your Hierarchy</p>	<p>Before your Hierarchy gets too crowded, you should take a moment to re-organize it</p> <p>1. To make organizer objects in your Hierarchy:</p> <ul style="list-style-type: none"> In the Hierarchy, create new empty GameObjects named "XR," "LIGHTING," "STATIC," and "DYNAMIC," Reset all of their positions to 0, 0 0. <p>2. To organize your objects into categories:</p> <ul style="list-style-type: none"> Drag objects from your Hierarchy either onto the organizer objects (to make child objects) or underneath the organizer objects (to use the organizer objects as headers) <p>Your Hierarchy should now be organized by category.</p>	<p>Explain what objects should go into "XR," "LIGHTING," "STATIC," and "DYNAMIC," and why.</p>
<p>Recap</p>	<p>New Functionality:</p> <ul style="list-style-type: none"> VR project set up Scene blocked out with primitives Locomotion around the scene Grabbable objects Socket interactors <p>New Concepts & Skills:</p>	

- | | |
|--|--|
| | <ul style="list-style-type: none">• Applying VR basics, such as locomotion and grabbable objects in your own VR app. |
|--|--|

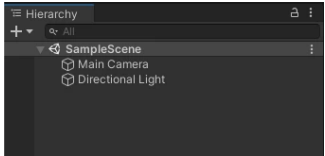
Next Lab:

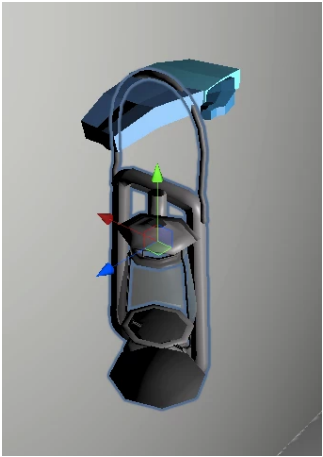


- Implement core functionality and interactions.

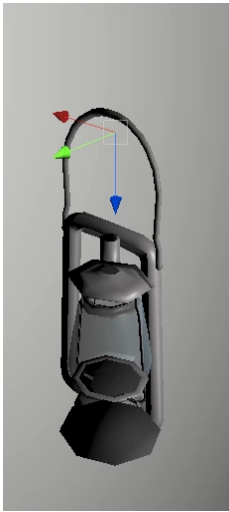

Quiz 1

Lesson Link	Quiz 1 - VR Basics - Unity Learn	
Length	15 Mins	
<p>In this quiz, you will test the knowledge and skills you learned in Unit 1 related to VR Basics.</p> <p>This quiz is part of the Create with VR course.</p> <p>This quiz will assess skills learned in the following tutorials:</p> <ul style="list-style-type: none"> • VR Project Setup • VR locomotion • Grabbable objects • Sockets 		

#	Question	Answers	Explanation
1	If you are targeting a mobile VR headset (as opposed to a tethered headset), which scriptable render pipeline (SRP) is most appropriate?	A. The Universal Render Pipeline (URP) B. The High-Definition Render Pipeline (HDRP) C. The VR Render Pipeline (VRRP) D. The Mobile Render Pipeline (MRP)	The Universal Render Pipeline (URP) is a prebuilt Scriptable Render Pipeline, made by Unity. URP provides artist-friendly workflows that let you quickly and easily create optimized graphics across a range of platforms, from mobile to high-end consoles and PCs. HDRP is not recommended for VR, since performance is so critical for user experience. Mobile and VR Render pipelines are not real.
2	When a VR Device claims to support "6 Degrees of Freedom" (6 DOF), what is "6" referring to?	A. The 6 degrees of freedom are: 1. Move head 2. Move left controller 3. Move right controller 4. Rotate head 5. Rotate left controller 6. Rotate right controller B. The 6 degrees of freedom are: 1. Move up 2. Move down 3. Move left 4. Move right 5. Move forward 6. Move backward C. The 6 degrees of freedom are: 1. Move along x axis 2. Move along y axis 3. Move along z axis	The degrees of freedom refer to the number of axes that are tracked for position and rotation. Older headsets were mostly 3DOF, since they would only track the rotation about the 3 axes, but not any of the position axis data.

		4. Rotate about x axis 5. Rotate about y axis 6. Rotate about z axis	
3	<p>Which of the following can be done through the Package Manager window?</p> <p>Select all correct answers.</p>	<p>A. Add the XR Interaction Toolkit package, which enables you to more easily develop for VR.</p> <p>B. Add the XR Plugin Management package, which allows you to deploy to different VR devices.</p> <p>C. Configure your Build Settings in order to deploy to your device vs the device simulator.</p> <p>D. Configure whether your scene is using either 3 DOF (Degrees of Freedom) or 6 DOF.</p>	<p>Both the XR Interaction Toolkit and the XR Plugin Management packages can be added through the package manager. Build settings are configured through the Build Settings window. Degrees of Freedom are configured through properties in your XR Rig.</p>
4	<p>You have an empty Unity scene with only a Directional Light and a Main Camera in it.</p>  <p>What would happen if you added an XR Origin object to that scene?</p>	<p>A. The Main Camera is duplicated so that there is one for each eye.</p> <p>B. A new object is added called an XR Origin and the Main Camera is made a child object of the XR Origin object.</p> <p>C. The Camera component on the Main Camera is replaced by an XR Camera component.</p> <p>D. The Directional Light is removed from the scene, since it cannot be rendered in VR.</p>	<p>When you add an XR Origin object, the Main Camera is automatically relocated in the Hierarchy as a child object of the XR Origin. A new Tracked Pose Driver component is added to the Camera, which applies the current Pose value of a tracked device (headset) to the transform of the GameObject.</p>
5	<p>Listed below are four types of locomotion that are possible in VR.</p> <p>Which two of these are least likely to induce simulator sickness?</p> <p>Select the two correct answers.</p>	<p>A. Physical (Room-scale) movement - the user physically moves in the real world in order to move in VR.</p> <p>B. Continuous (Controller) movement - the user uses a thumbstick or D-pad to smoothly move through the scene.</p> <p>C. Scripted movement - the user is automatically moved around the scene, controlled by a script.</p> <p>D. Teleport movement - the user clicks and is instantly transported to a new location.</p>	<p>Simulator sickness tends to occur when the visual field makes it appear as if the user is moving, but the user is not actually moving in the real world. This illusion is known as "vection". Both Continuous and Scripted movement have this quality.</p>
6	<p>You want to make a ball that the user can grab and throw around the scene. The ball should bounce off of the walls and floor.</p> <p>Which components and properties would the ball likely have?</p> <p>Select all correct answers.</p>	<p>A. A Sphere Collider with "Is Trigger" enabled.</p> <p>B. A Physics Material assigned to the collider.</p> <p>C. "Is Kinematic" enabled in the Rigidbody component.</p> <p>D. "Use Gravity" enabled in the Rigidbody component.</p> <p>E. An XR Grab Interactable component with "Kinematic" movement type.</p>	<p>If you enable "Is Trigger" for the collider, the collider will no longer be able to collide with other objects.</p> <p>If "Is Kinematic" is enabled, Forces (like gravity), collisions or joints will not affect the rigidbody anymore. The rigidbody will be under full control of animation or script control by changing transform.position.</p>

7	<p>You made a socket interactor for a phone charging dock in VR. However, the phone object is not snapping to the socket. What could possibly explain why this is happening?</p> <p>Select all correct answers.</p>	<p>A. The collider does not have “Is Trigger” enabled.</p> <p>B. The collider is a box collider instead of a sphere collider.</p> <p>C. The socket and the phone interactable only share one interaction layer.</p> <p>D. The socket has a Rigidbody component.</p>	<p>In order for a socket to detect another interactable object, its collider needs to have “Is Trigger” enabled. Otherwise, the collider will act like a solid object that cannot overlap with other objects.</p>
8	<p>In the screenshot below, you can see the way the hand should be oriented when it picks up the lantern. The local direction of the lantern is shown by the axis gizmos in the image.</p>  <p>Which of the following orientations for an Attach Point for this lantern would achieve the desired effect shown in the image?</p>	<p>A.  Green arrow pointing forwards and blue arrow pointing right.</p> <p>B.  Green arrow pointing left and blue arrow pointing forward.</p>	<p>If you are holding your right hand out in a fist with your thumb straightened as if you are giving a “thumbs up” signal, the front of your fist should align with the blue (Z axis) and your thumb should align with the Y (green) arrow.</p>

		 <p>C. Green arrow pointing forward and blue arrow pointing down.</p>  <p>D. Green arrow pointing forward and blue arrow pointing up.</p>	
9	<p>Which of the following is true about mobile vs tethered headsets?</p> <p>Select all correct answers.</p>	<p>A. A mobile headset can support higher-end graphics than tethered headsets.</p> <p>B. A tethered headset gets its processing power from a gaming console or computer.</p> <p>C. A tethered headset supports 6 degrees of freedom (6DOF), while mobile headsets do not.</p> <p>D. Some mobile headsets can be plugged in and operate as tethered headsets.</p>	<p>A tethered headset is plugged into a more powerful computer or console, which provides all of the processing power. This allows tethered headsets to support higher-end graphics.</p> <p>The lines between these types of devices is sometimes blurred with devices that can act as mobile and tethered devices.</p>
10	<p>What does "XR" stand for?</p>	<p>A. Extreme Reality</p> <p>B. Extra Reality</p> <p>C. External Reality</p> <p>D. None of the above</p>	<p>XR refers to any technology that blurs the line between real and digital worlds. The X is a placeholder letter for various types of these technologies, including VR (Virtual Reality), AR (Augmented Reality), and MR (Mixed Reality). There is debate</p>

			over what the X stands for, some claiming the X stands for eXtended Reality.
--	--	--	--

Unit 2 - Events & Interactions

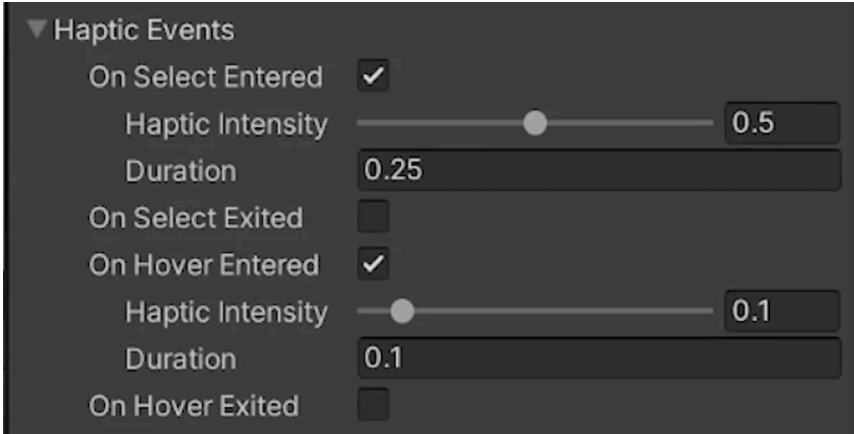
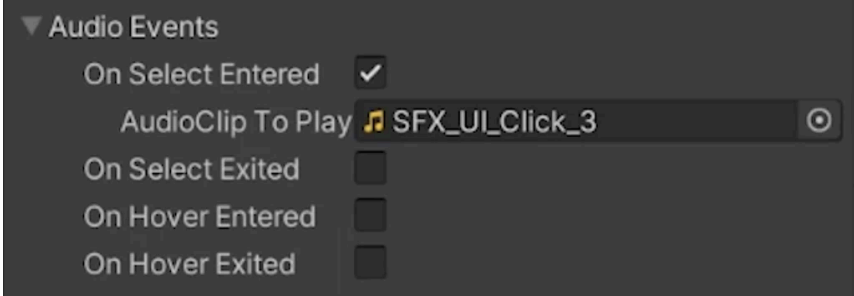


Unit Link	2 - VR Events and Interactions - Unity Learn
Length	9 Hours 15 Min
<p>In this unit, you will implement more complex interactions in VR based to make the experience more immersive. In these tutorials, you will:</p> <ul style="list-style-type: none">• Implement audio and haptic feedback• Apply new functionality to specific objects• Allow the user to toggle between multiple types of interactors• Add user interfaces to world <p>In the Challenge for this unit, you will apply your skills in a 3D painting prototype. Then, in the Lab, you'll implement the core functionality of your personal project. Finally, in the Quiz, you will test your new knowledge.</p> <p>This unit is part of the Create with VR course.</p>	

Lesson 2.1 - Audio and Haptics

Lesson Link	2.1 - Audio and Haptics - Unity Learn
Length	1 Hours 30 Mins
<p>Project Objective: In this lesson, you will learn how to increase the immersion of your project through touch and audio stimuli. By the end of this lesson, the user will receive haptic and auditory feedback when they hover over or grab an object. There will also be 3D spatial ambient sound in the scene.</p> <p>This lesson is part of the Create with VR course.</p> <p>Learning Objectives: By the end of this lesson, you will be able to:</p> <ul style="list-style-type: none"> • Implement haptic vibrations or sound effects on controller-based events in order to provide tactile or auditory feedback to the user in certain situations (e.g. when the user hovers over or selects something). • Understand the difference between 3d and 2d audio (diegetic vs non-diegetic) in Unity in order to distinguish scenarios where each might be appropriate. • Modify the spatial blend of an audio source in order to change it from 2D to 3D sound. • Modify the roll-off of an audio source in order to specify how it should change in volume as distance from the source increases. • Implement reverb zones in a specific area of a scene in order to better simulate the quality of sound in different environments (e.g. a cave vs a living room) • Understand the purpose of audio spatializer plugins in order to appreciate the importance of realistic audio in VR. 	

Step	Instructions	Teacher Notes
Step 1 - Add haptics on hover and select	<p>You can add haptic on hover enter and select enter to provide physical feedback to the player's hands.</p> <ol style="list-style-type: none"> Edit both controller objects at the same time: <ul style="list-style-type: none"> • In the Hierarchy, expand XR Rig > Camera Offset. • Ctrl/Cmd+select both the RightHand Controller and LeftHand Controller so they are both selected. Locate the haptic events: <ul style="list-style-type: none"> • At the bottom of the Controller objects' XR Ray Interactor components, expand the Haptic Events fold-out. Create subtle haptic feedback when the user hovers over an object: <ul style="list-style-type: none"> • Click the On Hover Entered check box to enable it. 	<p>Warning: If you make the vibrations long, it stops being meaningful and also gets very annoying.</p>

	<ul style="list-style-type: none"> Set the Intensity to a low value (e.g. 0.1-0.5). Set the Duration to a very low value (e.g. 0.1 seconds). <p>4. Create more noticeable feedback when the user grabs an object:</p> <ul style="list-style-type: none"> Select the On Select Entered check box. Set the Intensity and Duration values to slightly higher values (e.g. 0.25 seconds).  <p>The controller should now vibrate subtly when you hover over an object and slightly more so when you select an object.</p>	
<p>Step 2 - Add audio on hover or select</p>	<p>Now that you have haptic feedback, you can enhance the User Experience (UX) with audio feedback.</p> <p>1. Select the sound you want to use:</p> <ul style="list-style-type: none"> From the Course Library > Audio > FX, determine which subtle UI sound effect you want for the hover or select event. To preview the sound, you may need to drag up the Preview panel from the bottom of the Inspector window. <p>2. Apply this sound effect to both controllers:</p> <ul style="list-style-type: none"> From the Hierarchy, select both the RightHand Controller and LeftHand Controller objects. In the XR Ray Interactor component, expand the Audio Events fold-out. Click to enable either the On Select Entered or the On Hover Entered setting. and then assign your chosen sound effect.  <p>You should now hear a sound effect either when you hover over an object, or when you select an object.</p>	<p>Warning: Want this to be subtle and not get in the way - you just want to alert the player that something has happened.</p> <p>Demo: Might need to pull up the preview window to play sound.</p> <p>Warning: Do not apply a sound for hover AND select - it will likely be annoying for the user.</p>
<p>Step 3 - Add 3D audio from the fireplace</p>	<p>In addition to audio feedback on events, you can add 3D spatial audio into the scene emanating from the fireplace to increase immersion.</p>	<p>Explain: Difference between 2d and 3d sound.</p>

1. Give the fire a sound effect:

- Add an Audio Source component to the particle object.
- For the **AudioClip** property, assign the SFX_Loop_Fire sound effect from the **Course Library > Audio > FX** folder.

2. Make the sound play automatically and loop:

- In the Audio Source component, make sure both the **Play on Awake** and **Loop** settings are enabled.

3. Configure this component as a 3D sound:

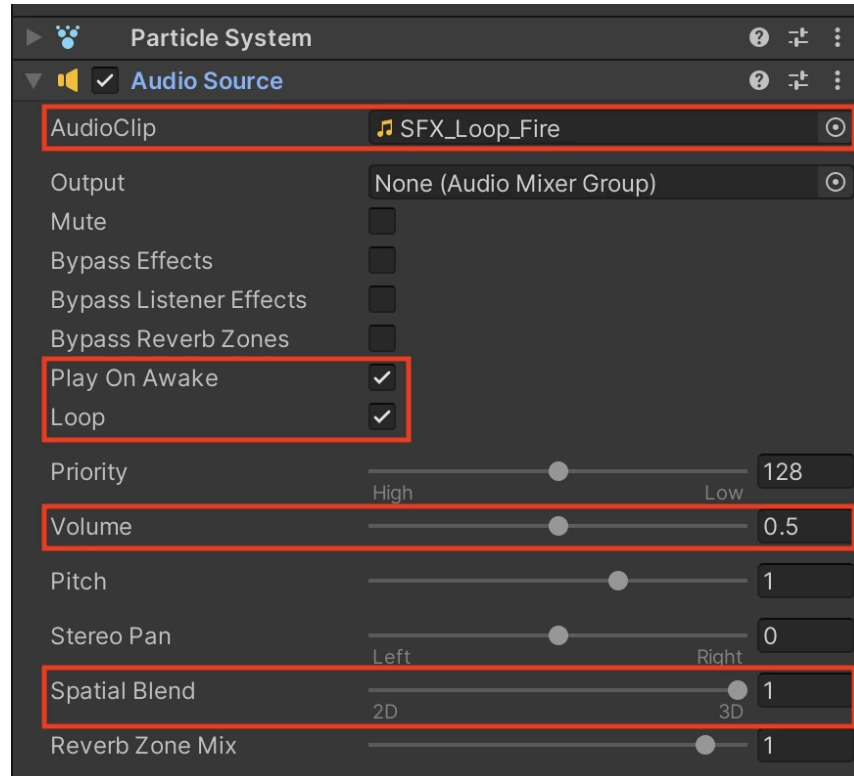
- Adjust the **Spatial Blend** property by dragging the slider all the way to the right **3D** setting (or by setting the slider to 1).

4. Edit the overall volume and volume roll-off of the fireplace:

- Edit the **Volume** property.
- Expand the 3D Sound Settings fold-out, then adjust the **Min Distance** and **Max Distance** values.

Explain - you can mess with the Volume Rolloff, but the default tends to be fine.

Explain: Min Distance is the distance you can be from the Audio Source before it starts to decrease.



The fireplace should now make a crackling sound, which should change in volume and left/right balance depending on where you are located and oriented in the scene.

Related resources:

- [Audio Source Settings](#)

Step 4 - Add a reverb zone

To maintain a sense of presence in VR, it's especially important to make sure audio behaves the way it would in the real world. This includes considering how sounds might reverberate around different kinds of rooms.

1. Add a new Audio Reverb Zone as a child of the Room object:

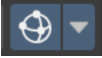
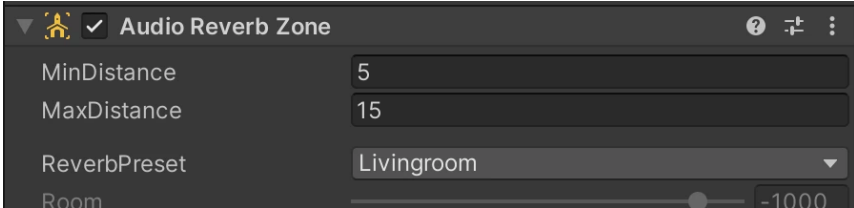
- In the Hierarchy, right-click the **Room_[style]** object.
- Click **Audio > Audio Reverb Zone**.

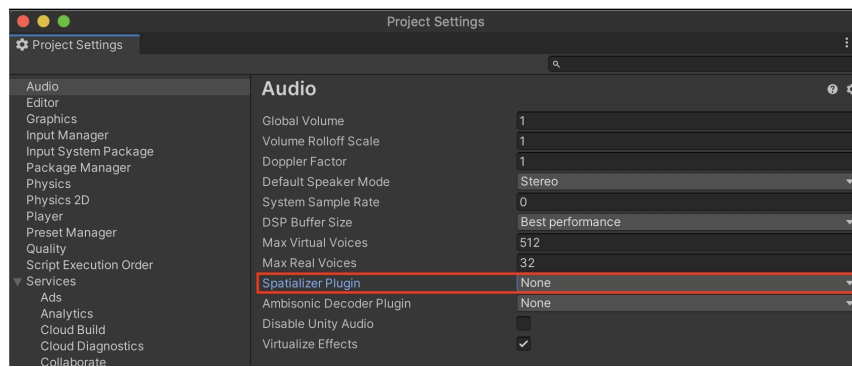
2. Make sure the reverb zone encompasses your whole room:

- In the Audio Reverb Zone component, make sure the **Min Distance** property is set to at least the width of your room (10).

Explain: What is a reverb zone?

Explain - We don't need a transition from reverb zones, so the Max value isn't important - we just care that the min value is larger than the size of the room.

	<ul style="list-style-type: none"> • Note: To visualize the zone, you may need to turn on gizmos and zoom out. In Unity 2021 LTS, the Gizmos button is represented by a sphere icon with dots on its vertices:  <p>3. Make your reverb zone match your room:</p> <ul style="list-style-type: none"> • Use the Reverb Presets drop-down to experiment with different styles of reverb. • Select a setting that matches your room (Room or Living Room).  <p>You should now have a reverb zone set up in your room to make the 3D sound even more realistic.</p> <p>Related resources:</p> <ul style="list-style-type: none"> - Audio Reverb Zones 	<p>Demo: To visualize the zone, you may need to turn on gizmos and zoom out.</p>
<p>Step 5 - [Optional] Experiment with spatializer plugins</p>	<p>[Optional] In addition to Unity's built-in 3D sound, which changes the volume in the left and right ears depending on player distance and orientation, there are additional spatializer plugins that you can install to make the sound even more realistic.</p> <ol style="list-style-type: none"> 1. Download and import a plugin: <ul style="list-style-type: none"> • From the links below, download the relevant spatializer plugin for your platform. • Import the package into your project. 2. Enable the plugin: <ul style="list-style-type: none"> • Click Edit > Project Settings. • Select the Audio section from the left panel. • Use the drop-down next to Spatializer Plugin to select your plugin. 3. Implement the plug-in on your fire particle: <ul style="list-style-type: none"> • Follow the instructions relevant to your device to implement spatialized audio. 	<p>Warning: This is not necessary - for the most part, the default 3d setting will get the job done.</p> <p>Explain: How is this different from the default setting? If you put a microphone in someone's ear, it sounds different from if the mic were outside the ear - that diff is the HRTF (head related transfer function).</p>



You should now have even more realistic sounding audio in your scene, taking into account the user's Head Related Transfer Function (HRTF).

Related resources:

- [Unity Audio Spatializer](#)
- Downloads
 - [Oculus Spatializer](#)
 - [Microsoft HRTF Spatializer](#) (supports UWP and PC running Windows 10)
- Oculus Spatializer Info
 - [Oculus 3D Audio Spatialization Guide](#)
 - [Oculus Spatializer Features](#)
 - [Apply Spatialization in Unity](#)

Recap

New Functionality:

- Haptic feedback on controllers
- Audio feedback on controllers
- Realistic 3D Audio from fireplace

New Concepts & Skills:

- Importance of haptics and audio for immersion
- 3d vs 2d audio
- Reverb zones
- Spatialized audio with head-related transfer function (HRTF)

Technical Skills:

- Audio & Haptic events

Extensions

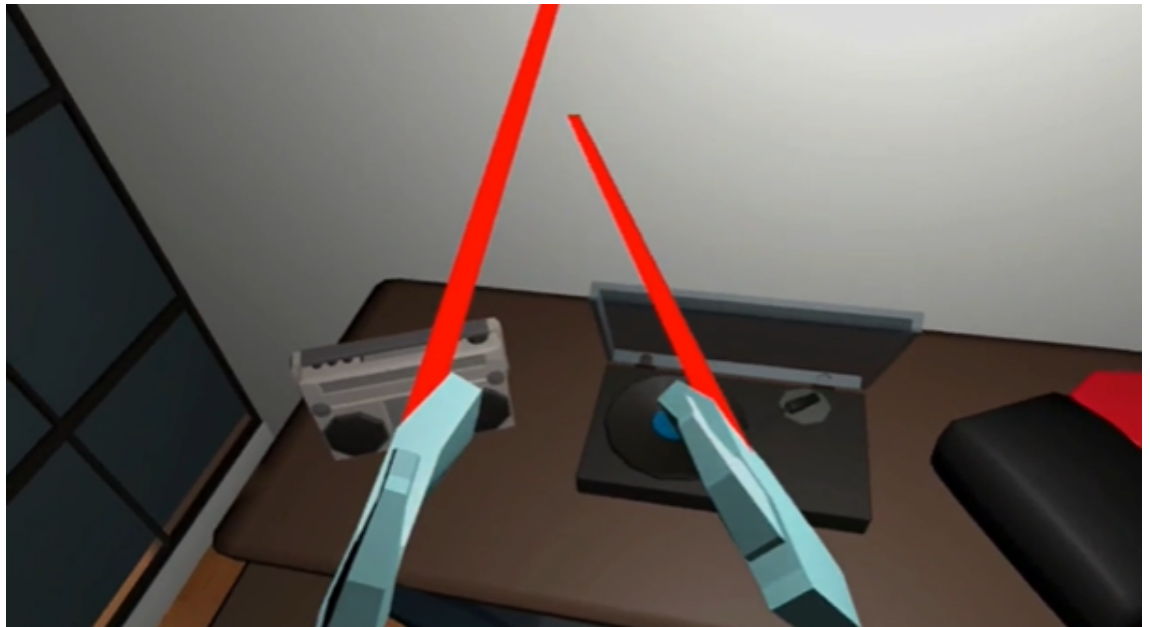
If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.

Extension - Add a speaker

1. Add a speaker [Easy]

Add a speaker to the room that plays music - or even a record player that actually rotates:

- Find a speaker from the Course Library > Prefabs > Objects > Speakers folder.
- Find music options from the **Course Library > Audio > Music** folder.
- Bonus - If you use the Record Player, you'll need a simple script to rotate the platter the record spins on.



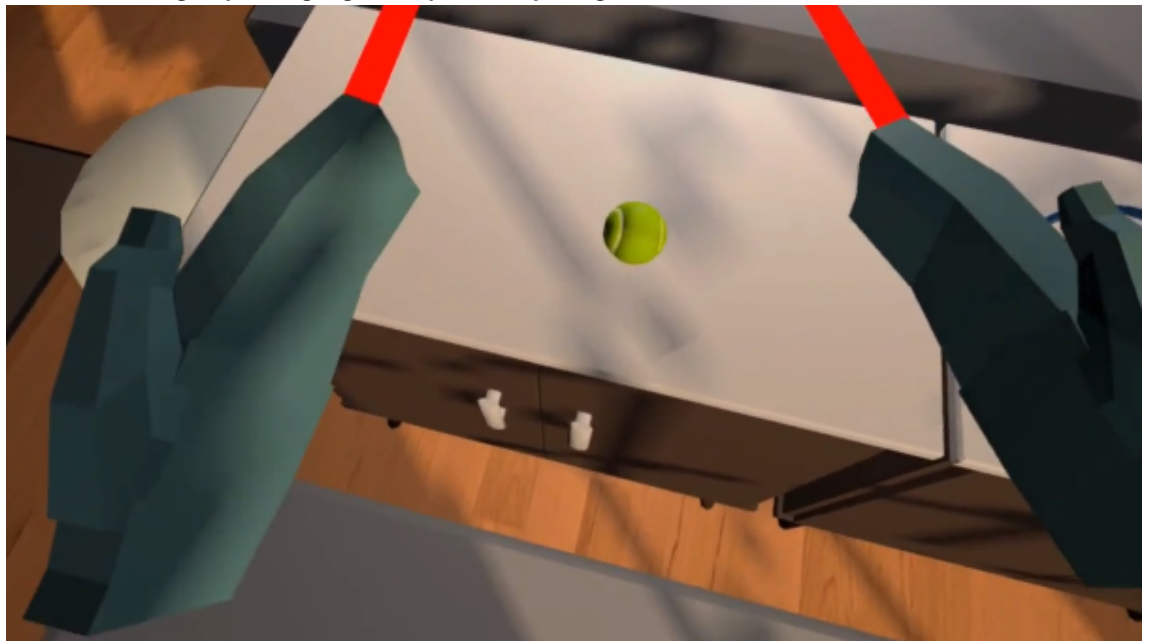
See lesson related videos [here](#)

Extension - Ball bouncing sounds

2. Add realistic bouncing sounds [Medium] [Requires programming]


Add a bouncing sound for your ball so that you hear it bounce every time it hits the floor:

- You may want to download and import a new sound effect
- Add a 3d audio source to your ball just like you did for your fireplace.
- To make it play a sound when it hits the ground, you will probably want to use the OnCollisionEnter function and PlayOneShot method.
- Bonus - if you want to make the volume change based on the speed the ball is moving, try using `rigidbody.velocity.magnitude`.

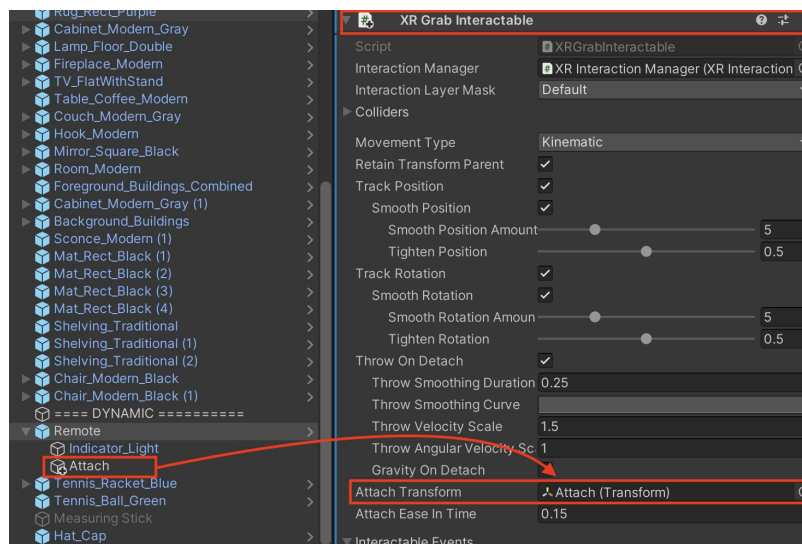


See lesson related videos [here](#)

Lesson 2.2 - Activation Events

Lesson Link	2.2 - Activation Events - Unity Learn
Length	1 Hours 30 Mins
<p>Project Objective: In this lesson, you will learn how to add unique functionality to an object when the user interacts with it in a certain way. By the end of this lesson, users will be able to pick up a remote and press a button on a controller to turn on a TV.</p> <p>This lesson is part of the Create with VR course.</p> <p>Learning Objectives: By the end of this lesson, you will be able to: Trigger actions based on interactions with a specific object in order to add additional functionality to an object (e.g. when an object is selected, when it is activated, etc).</p>	

Step	Instructions	Teacher Notes
Step 1 - Add a grabbable remote control	<p>Up to this point, you have only been able to pick up and put down objects. Now, you will add an object that you can <i>do</i> something with - or "activate" - once it's in your hand.</p> <ol style="list-style-type: none"> Add a remote control to your scene: <ul style="list-style-type: none"> Go to Course Library > Prefabs > Objects > Electronics, Drag the "Remote" prefab object onto a surface in your scene. Allow the object to be picked up: <ul style="list-style-type: none"> Add a Rigidbody component with Collision Detection set to Continuous Dynamic so it doesn't fall through the floor. Add an XR Grab Interactable component. Position the remote correctly in the player's hand when it is picked up: <ul style="list-style-type: none"> In the Remote object, create a new, empty "Attach" child object. Position and orient the Attach object appropriately. Assign the Attach object to the Attach Transform property in the XR Grab Interactable component. 	<p>Explain: You probably want the remote to be more vertical in your hand rather than pointing straight ahead so that you have to tilt your hand forward to use it.</p> <p>Warning: It's very easy to forget to actually assign the attach point to the XR Grab Interactable.</p>



You should now be able to pick up the remote in a way that looks natural.

Step 2 - Play a sound when the remote is activated

Now you will make the remote make a click sound when you are holding it and press the trigger.

1. Allow the remote to emit sound:

- On the Remote object, add an AudioSource component
- In the AudioSource component, slide the **Spatial Blend** property all the way to full **3D** ("1") to allow for 3D audio.

2. Give the remote the ability to play a sound:

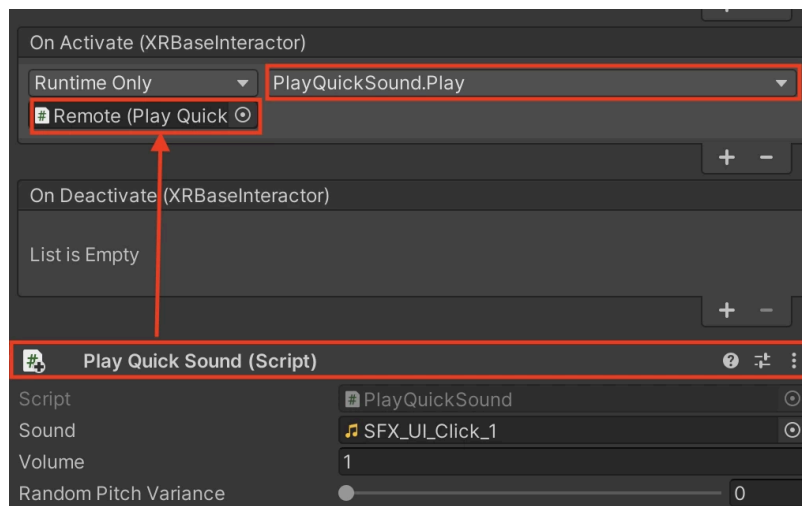
- Add a Play Quick Sound component to the remote.
- For the **Sound** property, assign an audio clip you want from the **Course Library > Audio** folder.

3. Locate the remote's "Activated" event:

- At the bottom of the remote's XR Grab Interactable component, expand the **Interactable Events** fold-out to see all of the event options.
- Locate the **Activated** event

4. Play a sound when the remote is activated:

- In the **Activated** event, click the **+** to add a new action.
- Drag and drop the **Remote** object to the empty Object slot in order to access its components.
- Click the **No Function** drop-down to select a function, then select **Play Quick Sound > Play ()**.



Demo: What are the other interactable events (On Hover, Select Enter, Activate, etc)?

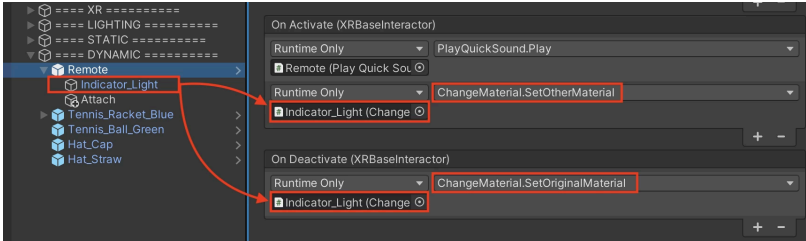
Explain:

PlayQuickSound is one of the custom scripts that we provided for this project - all of these are located in the Course Library > Scripts > Actions folder.

Demo: If you are familiar with programming, check out the scripts - you can use them as a template for creating your own. If you're not a programmer, no problem - you can still read through them and read comments to see what's going on.

Demo: Open the PlayQuickSound script and show them that there are functions in the script that we'll get to choose from

Demo: If you open the XRI Default Input Actions objects, you can see that the activate action is binded to the trigger button.

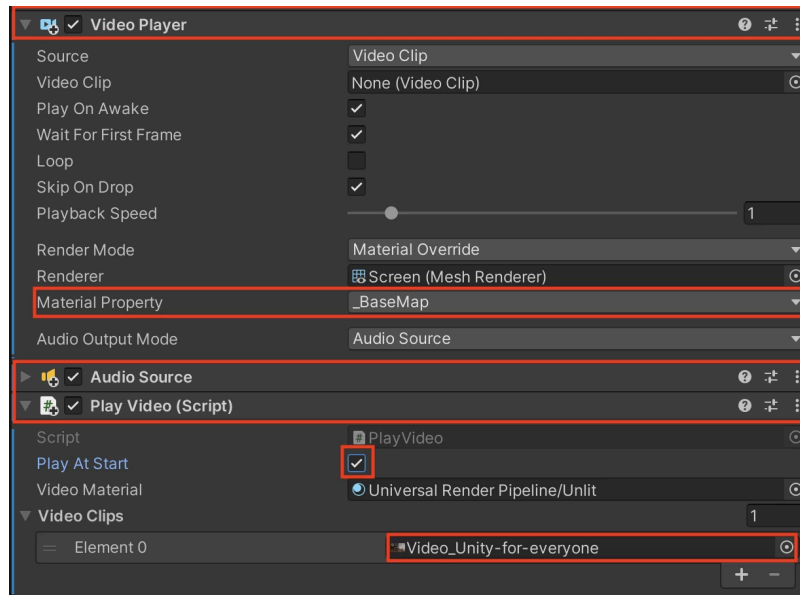
	<p>With the remote in your hand, you should now be able to press the activate button, the controller trigger, and hear a sound.</p>	
<p>Step 3 - Flash a red light when the remote is activated</p>	<p>To provide additional feedback that the remote has been pressed, you can also turn the indicator light red when the remote is activated.</p> <ol style="list-style-type: none"> 1. Add the capability for the indicator light to turn red: <ul style="list-style-type: none"> In the Hierarchy, expand the Remote object and select the Indicator_Light child object. Add a Change Material component. Assign a Red material to the "Other Material" property. 2. Change the material color to red when the trigger is pressed: <ul style="list-style-type: none"> On the Remote parent object, at the bottom of the XR Grab Interactable component, expand the Interactable Events fold-out to see all of the event options. In the Activated event, click the + to add a new function. Assign the Indicator Light child object to the Object property. Click the No Function drop-down to select a new function, then click Change Material > SetOtherMaterial () function. 3. Make the material revert when the activate button is released: <ul style="list-style-type: none"> In the Deactivated event, click the + to add a new action. Assign the Indicator_Light object, then select the Change Material > SetOriginalMaterial () function.  <p>When you activate the remote, the indicator light should turn red and then revert to its original color when the activate button is released.</p>	<p>Explain: You could toggle an actual red light, but you should avoid realtime light changes whenever possible.</p> <p>Demo: Open the ChangeMaterial.cs script and show them that there are functions in the script that we'll get to choose from</p> <p>Explain: Deactivated - this is when the trigger is released.</p>
<p>Step 4 - Allow the TV to play video</p>	<p>Before you get the button set up to turn the TV on, you should make sure the TV in the scene can successfully play video.</p> <ol style="list-style-type: none"> 1. Allow the tv to emit 3D audio: <ul style="list-style-type: none"> In the Hierarchy, select the Screen child object of the Television. Add an Audio Source component. Drag the Spatial Blend property to full 3D (1.0) to enable spatial audio. 2. Allow the TV to play video: <ul style="list-style-type: none"> On the Screen child object, add a Video Player component. Change the Material Property to _BaseMap (instead of _MainTex) to project the video to its material rather than its texture. 	<p>Explain: These videos are low resolution to keep the size of the app down. You could get your own videos if you want - you just need to make sure you have the rights to use it.</p> <p>Explain: The PlayVideo script is one we supply for you to add additional play/pause/next functionality.</p>

3. Add additional video control functionality:

- On the Screen child object, add a Play Video component.
- Select the **Play At Start** check box to play the video automatically.

4. Select the video to play on the screen:

- In the Play Video component, expand the **Video Clips** fold-out.
- Set the list **Size** property to 1 and press Enter to reveal an empty Video Clip slot.
- Assign one of the videos from the **Course Library > Videos** to the Video Clip property.



Now when you run your application, the video should play automatically on the TV Screen with 3D audio.

Step 5 - Control the TV with the remote

Now that the TV is working, you need to be able to control it with the remote.

1. Stop the video from playing automatically:

- In the Play Video component, clear the **Play At Start** check box.

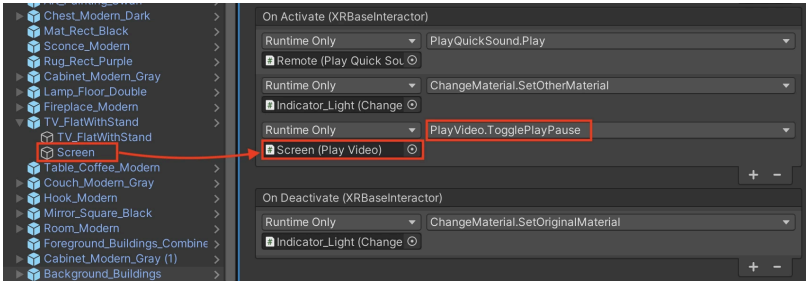
2. Locate the Remote's Interaction events:

- Select the **Remote** GameObject.
- In the XR Grab Interactable component, expand the **Interactable Events** fold-out to see all of the event options.

3. Trigger the TV when the remote is activated:

- In the **Activated** event, click the + to add a new action.
- Assign the **Screen** object to the empty Object property, since it has the Play Video script.
- Click the **No Function** drop-down and select the **PlayVideo > TogglePlayPause ()** function, or the **PlayVideo > TogglePlayStop ()** function, depending on whether you want it to pause or stop when pressed a second time.

Demo: Open the PlayVideo script to see the functions within.

	 <p>When you activate the remote, the video on the TV should alternate between playing and pausing or stopping.</p>	
Recap	<p>New Functionality:</p> <ul style="list-style-type: none"> - Grabbable remote - Play sound action - Change material action - Play video action - <p>New Concepts and Skills:</p> <ul style="list-style-type: none"> - Object activation - Events & actions - Scripts & functions <p>Next Lesson:</p> <ul style="list-style-type: none"> - Direct and Ray Interactors 	
Extensions	<p>If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.</p>	
Extension - Add a phone that plays video	<p>1. Add a phone that plays video [Easy]</p> <p>Make a phone that plays a video when you press the trigger:</p> <ul style="list-style-type: none"> • You can find phone prefabs from the Course Library > Prefabs > Objects > Electronics folder. • You will need to make a new separate “screen” child object with a material similar to the way the TV screen is set up. • Bonus: Try making the phone cycle through multiple videos using the Play Video > NextClip () function. 	



See lesson related videos [here](#)

Extension - Add a functioning flashlight

2. Add a functioning flashlight [Medium]

Add a working flashlight into the scene that toggles on and off when you press the trigger, including making a clicking sound to provide audio feedback:

- You will need to add a child SpotLight object to the flashlight.
- Try using the ToggleLight > Flip () function to flip the light on and off.

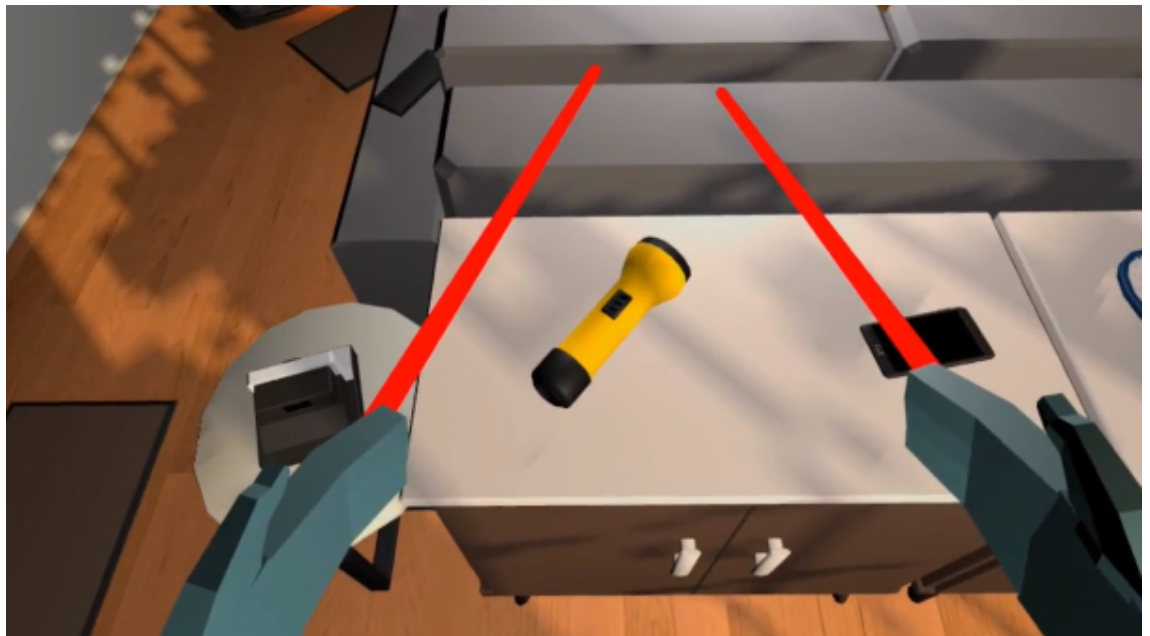
See lesson related videos [here](#)

Extension - Add a dart gun or stapler projectile

3. Add a dart gun or stapler projectile [Expert]

Add a dart gun that shoots darts or a stapler that shoots staples when you press the trigger:

- Locate the prefabs for the objects and projectiles in the Course Library > Prefabs > Objects > Launchers folders
- Use the LaunchProjectile script provided in the Course Library
- If you want, use the DestroyObject script provided in the Course Library to make the projectile disappear after a certain amount of time.
- **Bonus:** Make the stapler swing open with a hinge joint
- **Bonus:** Add sounds for when the object is launched



See lesson related videos [here](#)

Extension - Add a functioning polaroid camera

4. Add a functioning polaroid camera [Expert] [Requires Programming]

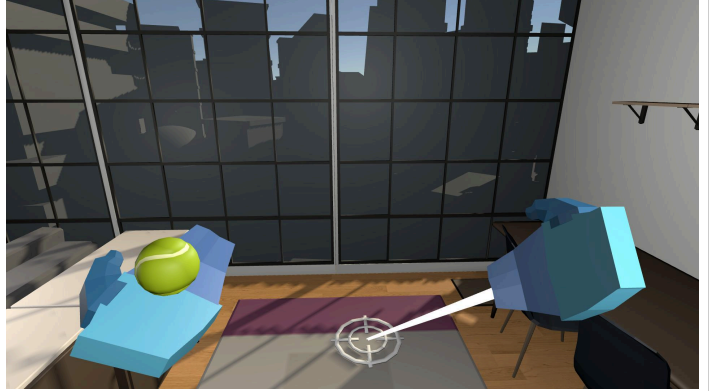
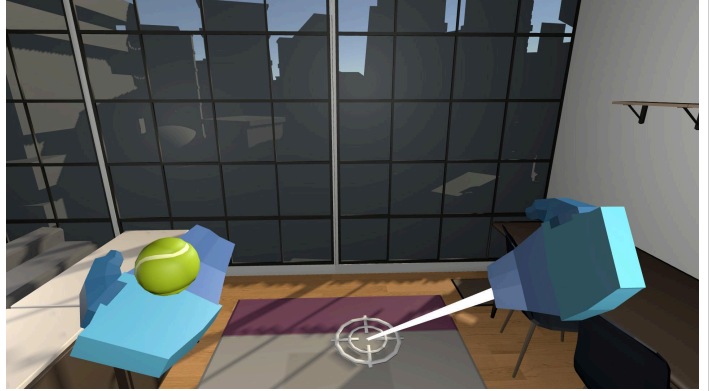
Add a functioning polaroid camera that allows you to preview on a viewfinder screen and print a photo using the trigger:

- For the viewfinder screen, look at how the mirror is set up with a render texture.
- Look at the documentation for [Camera.Render\(\)](#).
- Don't worry if the camera is currently rendering your rays for now - we'll learn how to hide your rays in the next tutorial.
- If you want some help, download an example script for the [polaroid camera](#) and the [photo](#)

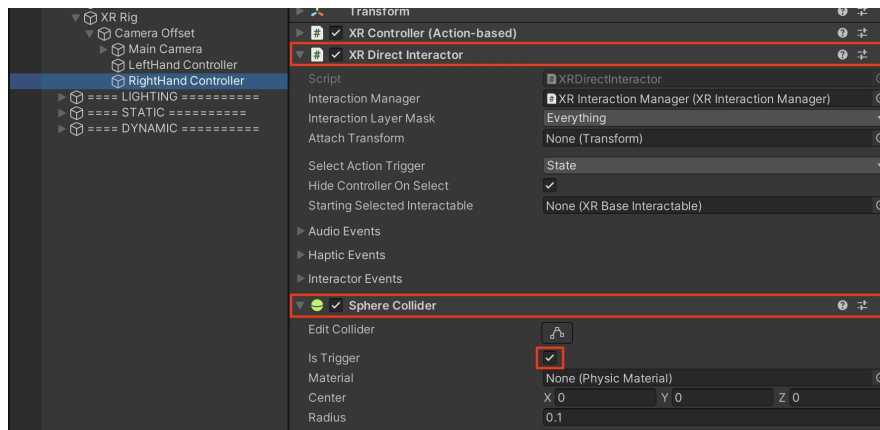


See lesson related videos [here](#)

Lesson 2.3 - Direct and Ray Interactors

Lesson Link	2.3 - Direct and Ray Interactors - Unity Learn
Length	1 Hours 30 Mins
Project Objective:	
In this lesson, you will learn how to implement Direct Interactors and how to switch between interactors on the same controller. By the end of this lesson, the user will be able to grab things directly with their hands, and then toggle a ray to point at things when they need to.	
This lesson is part of the Create with VR course .	
Learning Objectives:	
By the end of this lesson, you will be able to:	
<ul style="list-style-type: none"> Implement direct interactors on a controller in order to allow a user to pick things directly with their hands. Understand the need for a variety of interactor types in VR in order to allow the user to interact with the environment in different ways. 	
Toggle between multiple types of interactors on one controller in order to expand the range of actions a user can do with a single hand (e.g. switch between a ray and direct interactor).	

Step	Instructions	Teacher Notes
Step 1 - Add a direct interactor to the right controller	<p>Rather than using the ray interactors that come by default with the rig, you can set up a "Direct" interactor for one of the controllers.</p> <ol style="list-style-type: none"> Remove the Ray Interactor from the right controller: <ul style="list-style-type: none"> In the Hierarchy, select the RightHand Controller object. Remove the three components related to the ray interactor: the XR Interactor Line Visual component, the Line Renderer component, and the XR Ray Interactor component. Enable direct interactions on the right controller: <ul style="list-style-type: none"> Add a XR Direct Interactor component to the RightHand Controller. Select the Hide Controller on Select check box to have the object you pick up replace your hand visual. Enable detection of the Direct Interactor: <ul style="list-style-type: none"> Add a Sphere Collider component. Select the Is Trigger check box. Reduce the Radius to 0.05. 	<p>Explain: What is a Direct interactor?</p> <p>Explain: We'll be doing this for both hands, we're just starting with one for now so we can make sure we get all the settings right.</p> <p>Demo: It won't let you add a Direct Interactor on the same controller as the Ray Interactor</p> <p>Warning: You have to remove the components from bottom to top or Unity will get mad at you because of dependencies</p>



You should now be able to pick up all the grabbable objects in your scene using the hand with the direct interactor.

Step 2 - Add haptics and audio to the direct interactor

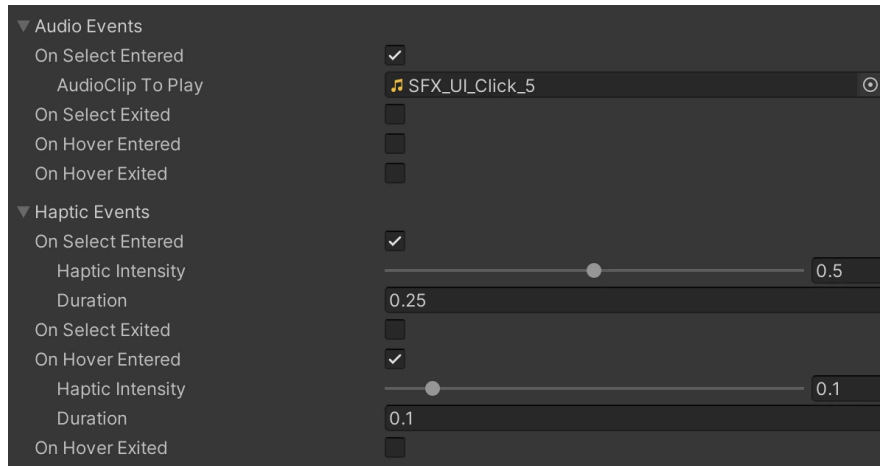
Just like you did for the ray interactor, you should add the same haptic and audio feedback for hovering over and selecting objects.

1. Recall how you set up haptic and audio feedback originally:

- From the Hierarchy, select the **LeftHand Controller** object (which should still have a Ray Interactor component).
- In the XR Ray Interactor component, expand the **Audio Events** and **Haptic Events** foldouts.
- Note the intensity and duration of your haptic events and the audio clip you selected for your Audio events.

2. Apply haptic and audio event settings to your new Direct Interactor:

- From the Hierarchy, select the **RightHand Controller** object.
- In the XR Direct Interactor component, apply the same haptic and audio event settings to match your other controller.



Your Direct Interactor should now have haptic and audio feedback on hover and select events.

Explain: Unfortunately can't paste component values from ray to direct - you just have to do it manually.

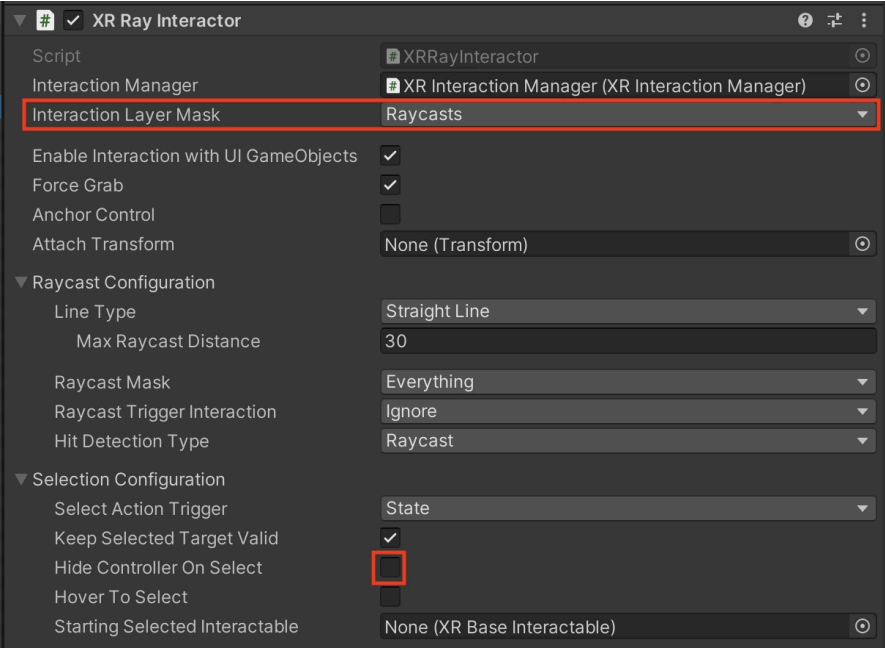
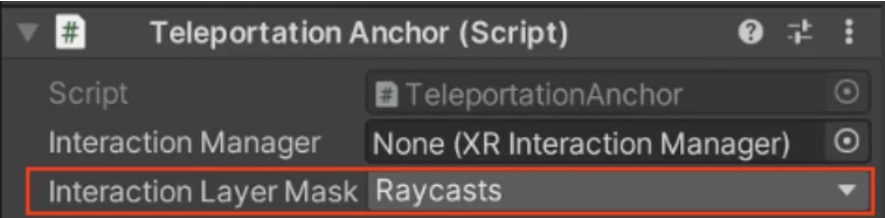
Step 3 - Prevent ray from picking up objects

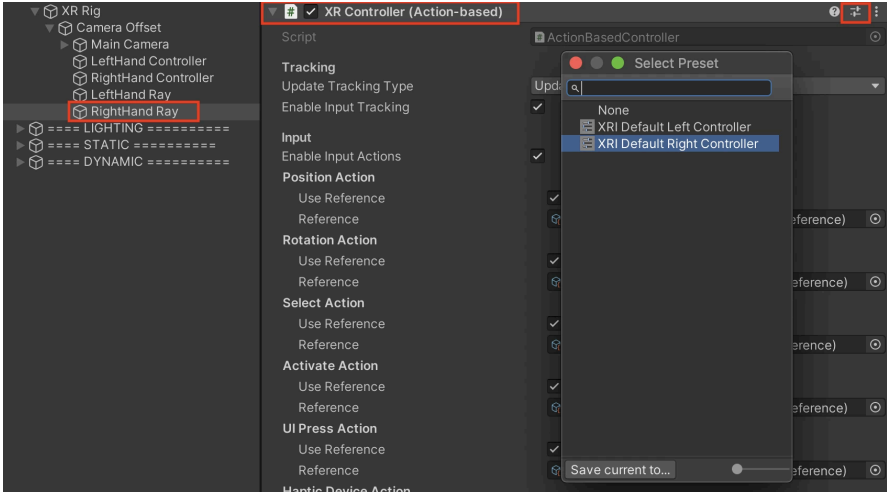
Right now, you can pick up everything with your ray, which doesn't feel natural. You should only be able to teleport with your ray and pick up objects with your hand directly.

1. Add a new layer for your Rays:

- In your other hand's **XR Ray Interactor** component, from the **Interaction Layer Mask** drop-down, select **Add Layer**.

Explain: Let's not use Ignore Raycast because it's a negative (double-negative is confusing) - if we're using the affirmative

	<ul style="list-style-type: none">•• Add a new "Raycasts" layer in the first available empty slot.• Note: Your Ray Interactor will not yet be set to that layer. You have only created the layer at this point. <p>2. Prevent your ray from interacting with your grabbable objects:</p> <ul style="list-style-type: none">• Still on your controller object's XR Ray Interactor component, for the Interaction Layer Mask property, assign only the Raycasts layer.• To do this, you need to first select "Nothing" and then re-select the "Raycasts" layer.• <p>3. Make sure your ray can interact with teleport objects:</p> <ul style="list-style-type: none">• For each Teleportation Anchor and Teleportation Area object, in the Interaction Layer Mask property, make sure the Raycasts layer is enabled. <p>4. Prevent your hand model from disappearing when you teleport:</p> <ul style="list-style-type: none">• In the XR Ray Interactor component, disable the Hide Controller On Select setting. <p>Note: If you can still use your ray to pick up objects, make sure your grabbable objects are not on the Raycasts Interaction Layer.</p>   <p>Your ray interactor should no longer be able to pick up objects, but should be usable for teleporting.</p>	<p>"Hats" - let's be consistent with our layering system.</p>
<p>Step 4 - Create a direct and ray</p>	<p>In order to be able to pick up objects with both hands, you should have direct interactors on both hands. Then you can just enable the ray when the joystick is moved. To achieve this, you will need to create the Direct and Ray interactors for each hand.</p>	<p>Explain: This is a very common interaction in VR - joystick toggles Ray.</p>

<p>interactor for each hand</p>	<ol style="list-style-type: none"> 1. Make alternative ray controllers for each hand: <ul style="list-style-type: none"> Duplicate the controller with the Ray Interactor two times. Renaming these objects "LeftHand Ray" and "RightHand Ray". 2. Control each ray with the correct hand: <ul style="list-style-type: none"> For your new RightHand Ray and LeftHand Ray objects, in the XR Controller component, make sure the Input Action References preset is set correctly to Left or Right for each object. To edit the preset, in the top-right of the component, click the Presets button and select the appropriate preset for the hand. 3. Make the LeftHand Controller behave as a Direct Interactor: <ul style="list-style-type: none"> On the LeftHand Controller object, remove the components related to the Ray interactor (the XR Ray Interactor component, the Line Renderer component, the XR Interactor Line Visual component). Copy the Direct Interactor and Sphere Collider components from the RightHand Controller to the LeftHand Controller. <p>Note: To copy a component:</p> <ul style="list-style-type: none"> Select the context menu in the top-right of that component and click Copy Component. To paste the component on another object, select a context menu of one of the receiving object's components and click Paste Component As New. <ol style="list-style-type: none"> 4. Make sure that you only have one hand model for each hand: <ul style="list-style-type: none"> On your new LeftHand Ray and RightHand Ray objects, in the XR Controller component, change the Model Prefab object to None. 	<p>Explain: This is just a temporary setup until we get the toggle functionality.</p> <p>Demo: Show that both types of interactors are now usable on both hands simultaneously.</p>
	 <p>Each hand should now be able to teleport with their Ray Interactors and grab objects with their Direct Interactors.</p>	<p>Step 5 -Toggle ray with button press</p> <p>Now that you have both Direct and Ray Interactors on each hand, you will add the ability to toggle the Ray on and off when a button is pressed.</p> <ol style="list-style-type: none"> 1. Add the ability to toggle the left ray: <ul style="list-style-type: none"> On the LeftHand Ray object, add a Toggle Ray component. For the Direct Interactor property, assign the LeftHand Controller object. 2. To be able to detect if a button is pressed: <ul style="list-style-type: none"> On the LeftHand Controller object, add an On Button Press component.

3. Specify the button to bind to this action:

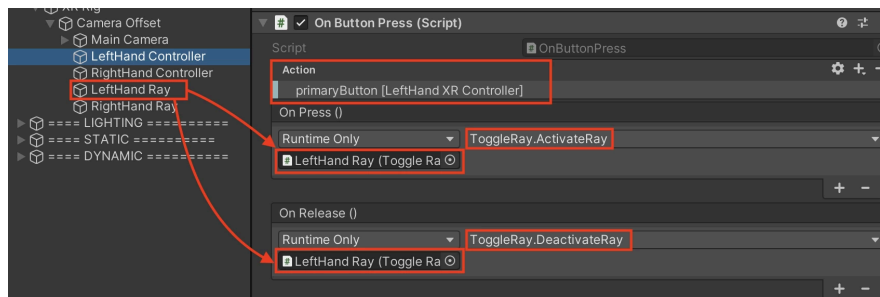
- In the On Button Press component, in the **Action** box, click the **+** and choose **Add Binding (Note: It is very easy to miss the Action box - look closely)**.
- Double-click where it says **No Binding**.
- In the **Path** drop-down, select **XR Controller > XR Controller (Left Hand) > Optional Controls > primaryButton**.

4. Activate the ray when the button is pressed:

- In the **On Press** event, click the **+** to add a new action.
- Assign the **LeftHand Ray** object, since it has the Toggle Ray component.
- From the function drop-down, select the **ToggleRay > ActivateRay ()** function.

5. Deactivate the ray when the button is released:

- In the **On Release** event, click the **+** to add a new action.
- Assign the **LeftHand Ray** object.
- Select the **ToggleRay > DeactivateRay ()** function.



The ray on the left hand should now only appear while the primary button is touched and disappear when the button is released.

Step 6 - Add an additional binding and repeat for other hand

One controller is working with one button. You can give the player more options of buttons to push depending on their preference, and then apply this functionality to the other hand.

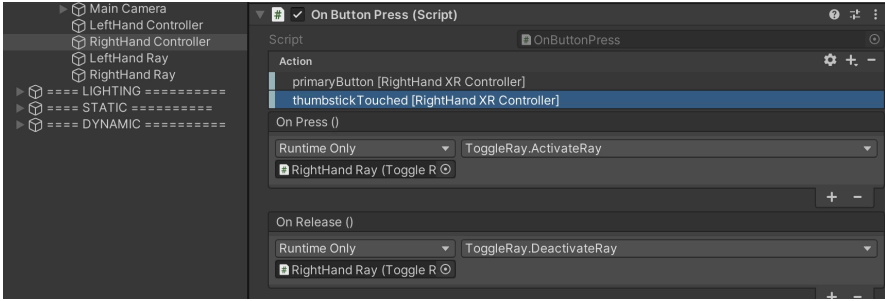
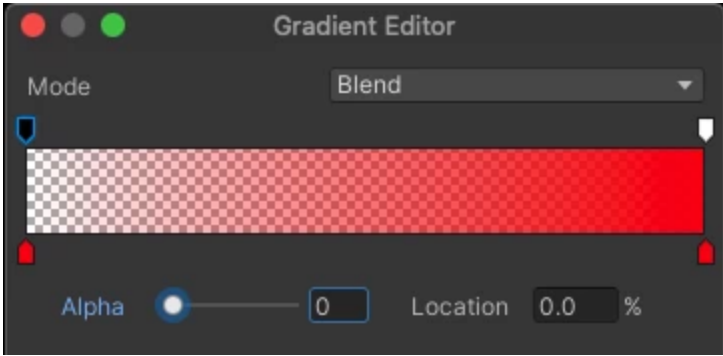
1. Add an additional binding that detects the thumbstick:

- In the On Button Press component, in the Action box, click the **+** button to add an additional binding to this action.
- Follow the same instructions as the previous step to select a binding, but select **XR Controller > XR Controller (Left Hand) > Optional Controls > thumbstickTouched** instead of **primaryButton**.

2. Add this same ray toggling functionality for your right hand:

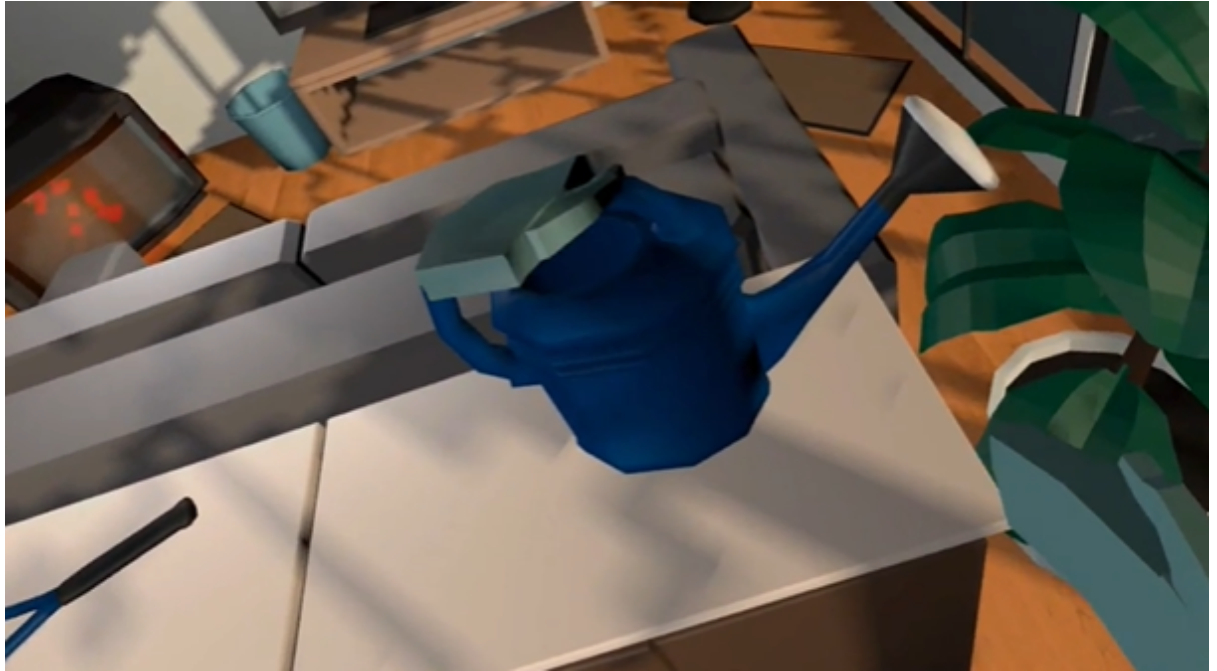
- Repeat all of the instructions from the previous step, but for your RightHand Controller and RightHand Ray:
 - Add a Toggle Ray component to the **RightHand Ray** and assign the RightHand Controller to the Direct Interactor property.
 - Add an On Button Press component to the **RightHand Controller** and add bindings for the Right Primary Button and Right Thumbstick
 - In the **On Press** event, call the **Toggle Ray > ActivateRay()** function on the **RightHand Ray** object.
 - In the **On Release** event, call the **Toggle Ray > DeactivateRay()** function on the **RightHand Ray** object.

Warning: If your controller batteries are low or your joystick is malfunctioning, this behavior may not work as expected.

	 <p>The rays on both hands should now only appear while the primary button or thumbstick are touched.</p>	
Recap	<p>New Functionality:</p> <ul style="list-style-type: none"> - Direct interactors on each hand - Toggle direct interactors on/off with button <p>New Concepts and Skills:</p> <ul style="list-style-type: none"> - Direct vs Ray interactors - Detect button press <p>Next Lesson:</p> <ul style="list-style-type: none"> - User Interfaces 	
Extensions	<p>If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.</p>	
Extension - Customize the ray interactor lines	<p>1. Customize the ray interactor lines [Easy] Edit the Ray Interactor line colors and transparency colors, including gradients and transparency:</p> <ul style="list-style-type: none"> • It looks good to make the start of the ray transparent so it doesn't overlap with your hands. • To customize the colors of the rays, look in the XR Interactor Line Visual components Valid and Invalid color gradient properties • In the Gradient Editor window, use the markers below the color bar to edit the color and the markers above the color bar to edit the transparency (Alpha) value.  <p>See lesson related videos here.</p>	
Extension - Watering can	<p>2. Add a functional watering can [Medium] Add a watering can that generates a water particle when you tilt it:</p> <ul style="list-style-type: none"> • Use the OnTilt script provided from the Course Library > Scripts > Conditions folder to 	

detect tilting.

- Find a Particle_Water object from the Course Library > Prefabs > Particles folder
- Use the ToggleParticle > Play() and Stop() functions to control the particle.
- Bonus - Add in pouring sound effect using the PlayContinuousSound script.
- Bonus - Use the OnVelocity script to also add a splashing sound when you shake it - Hint: add a separate audio source for this to not interfere with the pouring sound.



See lesson related videos [here](#).

Extension - Door hinges

3. Add hinges to cabinet doors [Expert]


Allow the user to open cabinet doors or chest lids using Unity's physics hinges:

- You will need to add a [Hinge Joint](#) and rigidbody to the door, and customize its angular limits
- You will need to add an XR Grab Interactable component to the handle and may want to add a [Fixed Joint](#) connected to your door's rigidbody.
- Make sure your door's colliders do not overlap any other colliders, which could cause it to get stuck.
- Warning: joints are very tricky and can be frustrating.

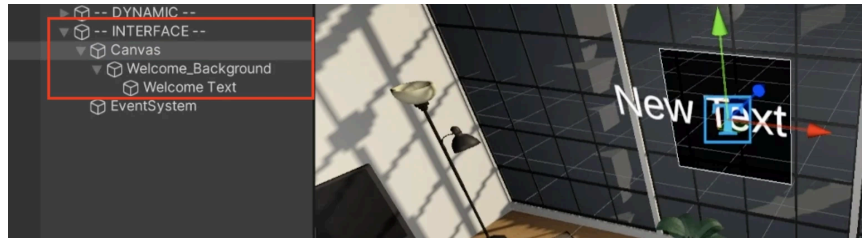


See lesson related videos [here](#).

Lesson 2.4 - User Interfaces

Lesson Link	2.4 - User Interface - Unity Learn
Length	1 Hours 30 Mins
<p>In this lesson, you will learn how to implement world space user interfaces in VR. By the end of this lesson, the user will be greeted by a welcome screen that provides basic instructions. They will also be able to bring up a simple reset menu that allows them to reload the scene.</p> <p>This lesson is part of the Create with VR course.</p> <p>Learning Objectives: By the end of this lesson, you will be able to:</p> <ul style="list-style-type: none"> Recall best practices for UI in VR in order to design an experience that is comfortable and intuitive for the user. Add a simple world space UI into the scene with text and a button in order to allow the user to read and interact with a user interface. 	
	

Step	Instructions	Teacher Notes
Step 1 - Add a world space UI into the room	<p>To create some welcome screen text, the first thing you need to do is add a canvas with text and position it in the room.</p> <ol style="list-style-type: none"> Add a new Canvas to your scene: <ul style="list-style-type: none"> Add a new empty object in your Hierarchy, reset its position, and rename it "INTERFACE". Create an XR > UI Canvas object within that organizer object. Position the canvas appropriately in the room: <ul style="list-style-type: none"> In the Canvas' Rect Transform component, set the XYZ Scale to 0.01. Set the Pos Y and Pos Z properties so the canvas is at the back of the room Add a semi-transparent black background for the UI: <ul style="list-style-type: none"> In the Hierarchy, create a UI > Image and rename it "Welcome Background". In the Image component, click the Color box. Change the color to black. Use the Alpha (A) slider to reduce the color's transparency. If your windows are rendering on top of your text: <ul style="list-style-type: none"> In the Canvas component, set the Order in Layer property to "1". Add text to the panel as a child object: <ul style="list-style-type: none"> In the Hierarchy, right-click Welcome Background and create UI > Text - TextMeshPro as a child object. Note: you may have to click "Import TMP Essentials" if prompted. 	<p>Explain: This will add an Event System object.</p> <p>Explain: Since it's an XR canvas, it will be "Worldspace" by default (instead of "Screen space")</p> <p>Explain: Screen space - overlay vs camera vs worldspace.</p> <p>Explain - default text vs TextMeshPro for lower resolutions - always choose TextMeshPro.</p>



You should now have a semi-transparent box in the room with huge off-center text that says "New Text".

Related Resources:

- [Canvas Render Modes](#)

Step 2 - Edit the welcome text

Now that you have the canvas in place, it is time to make a more useful welcome message.

1. Edit the shape and placement of the background image:

- In the image object's Rect Transform component, edit the **Width**, **Height**, **Pos X**, and **Pos Y** properties.

2. Edit the appearance of the text:

- On the text object, in the TextMeshPro - Text (UI) component, change the **Font Asset**, **Font Size**, **Vertex Color**, and **Alignment** properties.

3. Change the contents of the text:

- In the TextMeshPro - Text (UI) component, change "New Text" to a short, but informative welcome message about the scene.

4. Edit the size and spacing of the text box:

- In the Rect Transform component, set the **Width**, **Height**, **Pos X**, and **Pos Y** to position in your text's Rect Transform component.
- **Note:** Make sure to leave some margins and space for an "OK" button.



Your welcome text should now look the way you want it in the scene.

Explain: There's lots of other settings you can play with as well in the TMP component.

Step 3 - Add an OK button to the panel

You don't want the welcome text displayed forever, so you will add an "OK" button that hides it.

1. Add a button to the panel as a child object:

- In the Hierarchy, right-click the **Welcome Background** and click **UI > Button - TextMeshPro**.

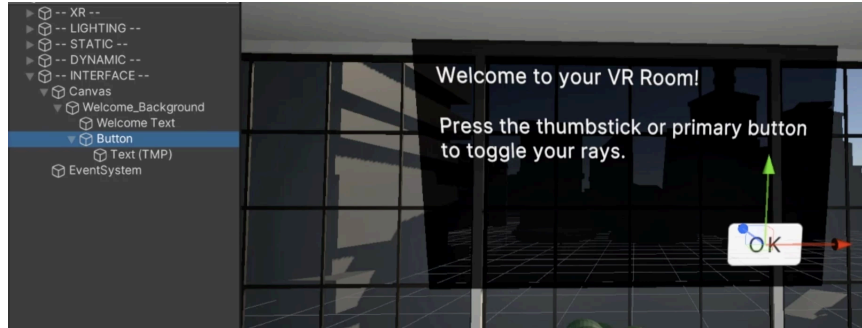
2. Edit the button's text:

- Select the button's child Text object.
- Change the text to "OK" or "Got it!".

Explain: How does the color tint work?

Warning: The button won't work yet - you will click it and nothing will happen - we will

- Edit the Text settings to match your other text.
- 3. Position the button appropriately:**
 - In the Rect Transform component, edit the button's position, width, and height.
 - 4. If you want to change the color of the button as you interact with it:**
 - In the Button component, edit the Normal, Highlighted, and Pressed Color properties.
 - 5. Prevent your ray from appearing active when it is over the background panel:**
 - In the Background Image component, disable the **Raycast Target** setting.
 - In the **Text** object's TextMeshPro component, in the Extra Settings section, disable the **Raycast Target** setting.



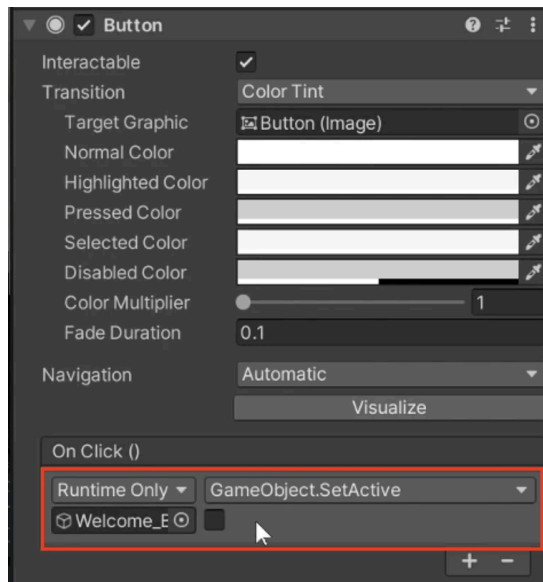
You should now have an "OK" button that changes color when it is highlighted and clicked with the trigger.

fix that soon.
Demo: If you wanted to change the image, could do that in the image component, too

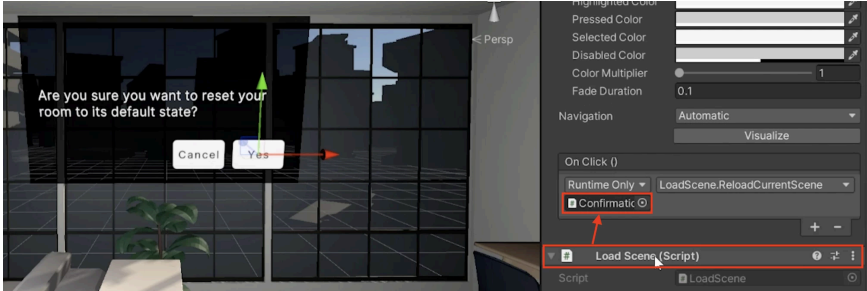
Step 4 - Make the OK button remove the panel

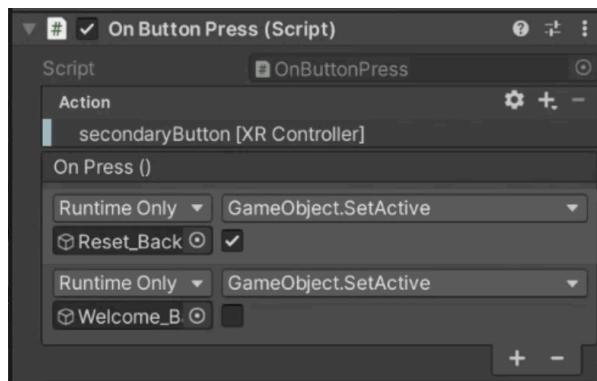
Now that the button looks the way you want, you should make it remove the welcome panel when clicked.

- 1. Locate the On Click event for the button.**
 - On the **Button** object, at the bottom of the Button component, locate the **On Click** event.
- 2. Deactivate the welcome panel:**
 - In the **On Click** event, click the **+** button to add a new action.
 - Assign the Welcome Background to the Object slot.
 - From the "No Function" drop-down, select **GameObject > Set Active (bool)** function and make sure the check box is deselected so that it is deactivated.



Explain: SetActive is a bool, which can be true or false. If you leave it unchecked, that is false, which means it will be set as INACTIVE.

	<p>When you select the OK button with the trigger, the welcome panel should disappear.</p>	
<p>Step 5 - Make a reset UI canvas</p>	<p>Now that the welcome UI works perfectly, it will be easy to duplicate and repurpose the UI elements to make a Reset UI that reloads the scene.</p> <ol style="list-style-type: none"> 1. Make new text for your reset screen: <ul style="list-style-type: none"> In the Hierarchy, duplicate the Welcome_Background object and rename the new object "Reset_Background". Temporarily deactivate the Welcome Background To be able to edit it more easily. Edit the text to be about resetting the scene to its original state. 2. Add a second button for your reset screen: <ul style="list-style-type: none"> Duplicate the button, then reposition it next to the original. Rename the buttons and edit their text to make one a "Cancel" button and the other an affirmative "Yes" button. 3. Make the Yes button actually reset the scene: <ul style="list-style-type: none"> Add a Load Scene component to the Yes button object. In the On Click event, assign the Load Scene component and choose the Load Scene > ReloadCurrentScene () function.  <p>From the Reset canvas, you should now be able to be able to click Cancel to hide the panel or click Yes to reload the current scene.</p>	<p>Explain: The "cancel" button should already work as expected, since it just disables the panel.</p> <p>Explain: LoadScene is a custom script we provided for you.</p>
<p>Step 6 - Show reset panel on button press</p>	<p>There's currently no way to return to the reset screen. To execute a reset, you can teleport the player back to the starting point when they press the secondary button.</p> <ol style="list-style-type: none"> 1. Only show the welcome panel at start: <ul style="list-style-type: none"> Re-enable the Welcome_Background object and disable the Reset_Background object. 2. Test for a Secondary Button press: <ul style="list-style-type: none"> On the XR Rig object, add an On Button Press component. Click + to add a new Action Binding. Assign the binding path to XR Controller > Optional Controls > secondaryButton. 3. Show the reset panel on the button press: <ul style="list-style-type: none"> In the On Press event, click the + to add a new action. Assign the Reset_Background object. Select the function GameObject > Set Active (bool), and make sure the check box is enabled. 4. Hide the welcome panel on the button press: <ul style="list-style-type: none"> Click the + to add a new action. Assign the Welcome_Background object. Select the GameObject > Set Active (bool) function, and make sure the check box is disabled. 	<p>Explain: What is the secondary button?</p> <p>Demo: Why hide the welcome panel? You don't want them on top of each other</p> <p>Explain: On Button Press is a custom script.</p>

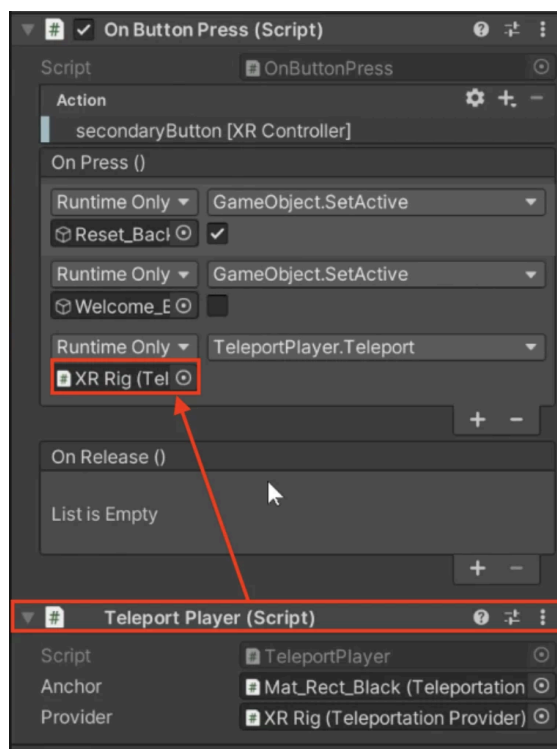


When you press the secondary button, the welcome panel should disappear and the reset panel should appear.

Step 7 - Teleport to the starting position for reset UI

- If the player is not at the starting position when they press the secondary button, they need to be teleported back to the starting position to view the Reset panel.
- 1. Make sure you have a teleportation anchor at the start of your scene:**
 - If you don't already have one, duplicate and reposition a teleportation anchor at the position you want your player to start in the room.
 - 2. Make your player start at the correct location:**
 - Match the **position** and **rotation** of the **XR Rig** object to the position and rotation of the starting anchor.
 - 3. Add a new teleporting behavior to the XR Rig:**
 - On the **XR Rig** object, add a Teleport Player component.
 - For the **Teleport Destination** property, assign the teleportation anchor at the start of your scene.
 - For the **Provider** property, assign your Teleportation Provider.
 - 4. Connect the teleportation action to the button click:**
 - In the **XR Rig** object's On Button Press component, for the **On Press** event, click the **+** to add a new function.
 - Assign the **XR Rig** object.
 - Select the **TeleportPlayer > Teleport ()** function.

Demo: Why would you want to teleport back to the starting position? Need to control where player is relative to UI in case they're right next to it and can't read it or don't see it.
Explain: Teleport Player is a custom script.



When you press the secondary button, the player should now teleport to the starting position of the scene in order to view the Reset UI.

Recap

New Functionality:

- Welcome screen with OK button
- Reset screen to reload scene

New Concepts and Skills:

- VR UI best practices
- Worldspace vs Screenspace UI

Next Lesson:

- Comfort & Accessibility

Extensions

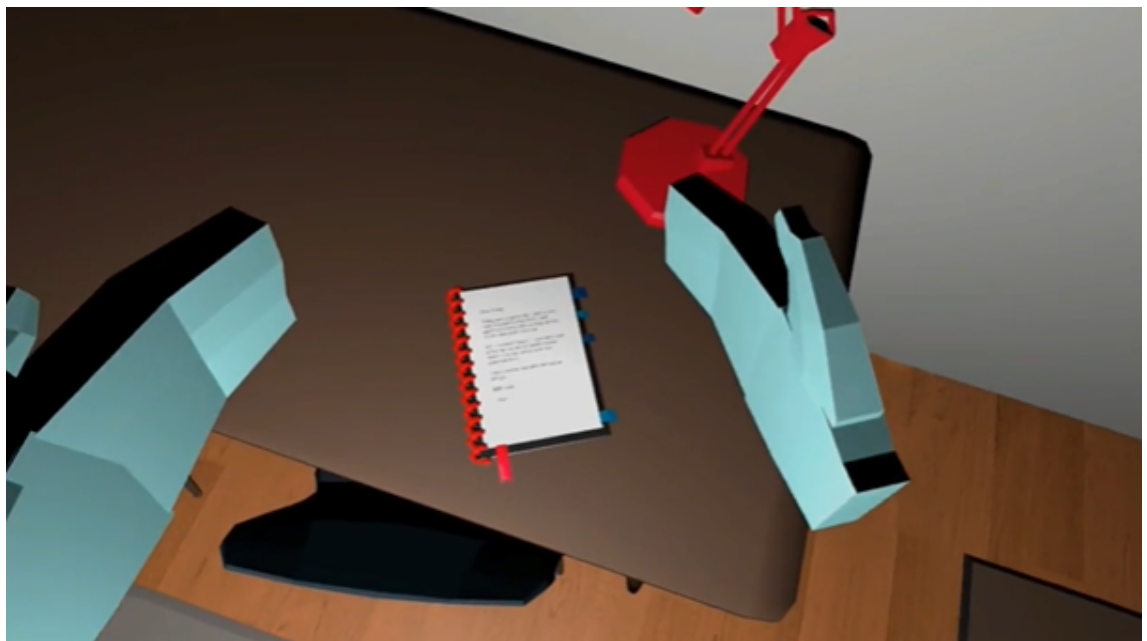
If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.

Extension - Real-world text

1. Add real-world text to the scene [Easy]

Add text onto a real-world object like a notebook to provide additional information to the user:

- Make a "TexMeshPro - Text" object a child object of whatever object you want it to display on.
- Make sure the Canvas is set to world space Render Mode
- Bonus - add audio narration that reads the text when you pick up the object.



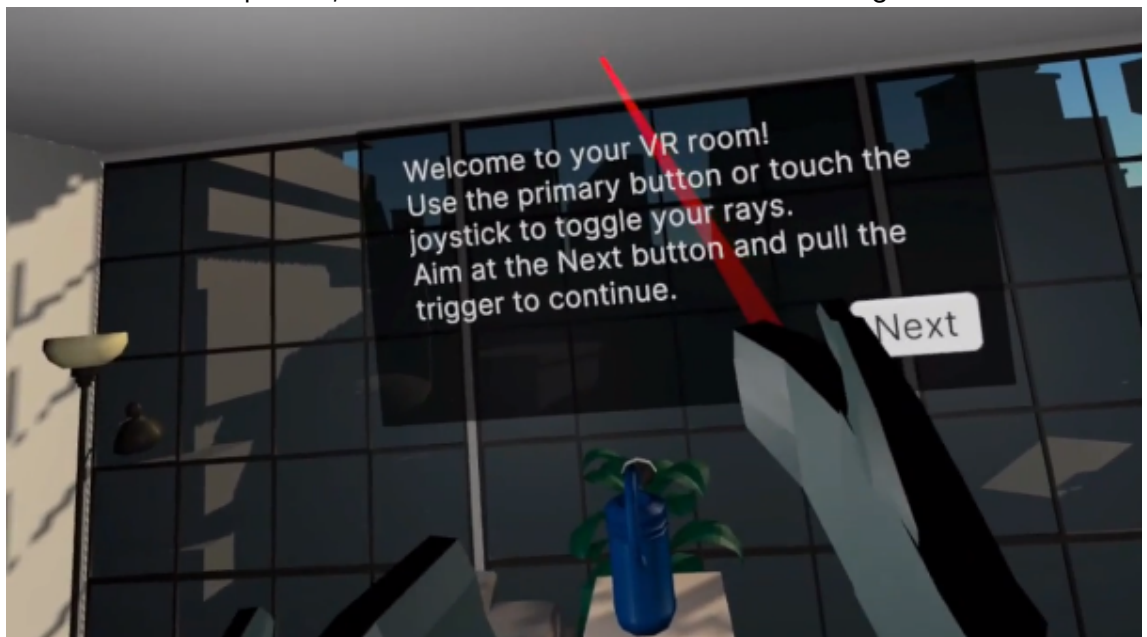
See lesson related videos [here](#)

Extension - Multi-step UI

2. Add multi-step user interfaces [Medium]

Instead of having a single paragraph of instructions at the start of your VR room with an "OK button," add a "Next" button that cycles through a few shorter strings of text:

- On your "TextMeshPro - Text" component, add the ShowMessagesFromList component, which was provided in the Course Library, then use the NextMessage() function to cycle through them. Note the "On Complete ()" event on this component, which fires after there are no more messages to show.



See lesson related videos [here](#)

Extension - Contextual Tooltips

3. Add contextual tooltips [Hard]

Add a small UI screen to guide the user in a particular task in your room:


- The UI screen should only appear when it is relevant.
- Bonus - if it is a multi-stage task like grabbing a hat and putting it on your head, try displaying multiple messages on the UI, depending on what stage the user is on.

- Bonus - ideally, the UI should only display the first time you complete that task.



See lesson related videos of contextual tooltips on Learn [here](#)

Challenge 2 - 3D Painting App

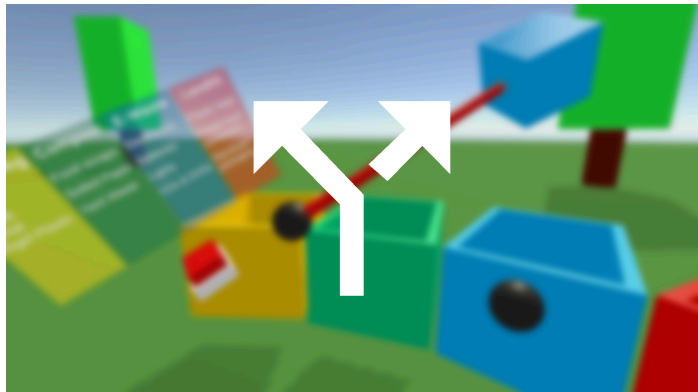
Lesson Link	Challenge 2 - 3D Painting - Unity Learn	
Length	1 Hours 30 Mins	
<p>In this challenge, you'll apply the skills you learned while making your VR Room in a 3D painting app. In this app, the user can choose the shape they want to paint on, select the color and size of their brush on a handheld palette, and then create their masterpiece!</p> <p>This challenge will review your skills learned in the following lessons:</p> <ul style="list-style-type: none"> • Audio & Haptics • Activation Events • Direct & Ray Interactors • User Interface <p>This challenge is part of the Create with VR course.</p>		

Getting started	<ol style="list-style-type: none"> 1. Open the broken 3D painting prototype Scene: <ul style="list-style-type: none"> • From the Project window, expand, Assets > Challenges > 02_3DPainting > Scenes • Double-click on the 3DPainting_Prototype_Broken Scene to open it. 2. Begin working on the challenge tasks: <ul style="list-style-type: none"> • Work through the tasks outlined in the steps below. • If you want to push your skills, attempt the optional bonus challenge tasks, as well. • If you get stuck, there are hints for each task at the bottom of the page.
-----------------	--

Challenge	Task	Hint
1. The Canvas that allows you to change the shape on the pedestal follows the camera.	<ul style="list-style-type: none"> • This canvas should be stationary next to the pedestal. 	1. The canvas render mode needs to be in "World Space" - not "Screen Space".
2. The toggled ray and palette canvas are tracking the opposite hands.	<ul style="list-style-type: none"> • The palette should move with the left hand and, when you press the thumbstick or primary button on the right controller, the toggled ray should emit from the right hand. 	2. The canvas render mode needs to be in "World Space" - not "Screen Space".
3. The paint brush activation is backwards, starting when the trigger is released and stopping when it's pressed.	<ul style="list-style-type: none"> • The paintbrush should create a trail while the trigger is being held, stopping when the trigger is released. 	3. Look at the input mapping in the XR Controller components for the LeftHand and the RightRay.

4. The medium brush size button turns invisible when pressed and makes the trail gigantic.	<ul style="list-style-type: none"> The button should turn the same grey color as the other buttons when pressed and change the trail to a size in between the small and large brush sizes. 	4. Look at the functions in the Activated and Deactivated events in the paintbrush's XR Grab Interactable component.
5. The paint brush sound effect stays the same, regardless of how far it is from you.	<ul style="list-style-type: none"> The paint brush sound should be at full volume close to your face, but about 50% volume when your arm is outstretched 	5. Make sure to check the alpha (transparency) value of the button tint. Look at the functions on the small and large brushes for references on the ideal trail width.
Bonus		
6. There is no audio feedback from the UI.	<ul style="list-style-type: none"> Add sound effects for all of the buttons in the UI and a continuous sound effect that plays while the paint brush is in use. 	6. Try using either the PlayQuickSound or the PlaySoundsFromList scripts.
7. There is no way to hide the pedestal completely.	<ul style="list-style-type: none"> Add a toggle UI to the menu next to the pedestal that shows/hides the pedestal object when toggled on/off. 	7. Use the Toggle() function in the ToggleVisibility script.
8. There is no way to save your work. [Expert]	<ul style="list-style-type: none"> Add a "Print" button that creates a miniature version of your 3d creation on a table next to you. 	8. No hints available. Use Google!
9. There is no way to paint on the back or sides of the object. [Expert]	<ul style="list-style-type: none"> Add a slider to the UI that rotates the entire pedestal, giving you the ability to paint the back as well. 	9. No hints available. Use Google!


Lab 2 - Personal Project Events & Interactions

Lesson Link	Lab 2 - Personal Project Events & Interactions - Unity Learn
Length	2 Hours
<p>By the end of this lab, your personal project will have most of its core functionality.</p> <p>This lab will draw on skills learned in the following lessons:</p> <ul style="list-style-type: none"> • Audio & Haptics • Activation Events • Direct & Ray Interactors • User Interface <p>This challenge is part of the Create with VR course.</p>	
	

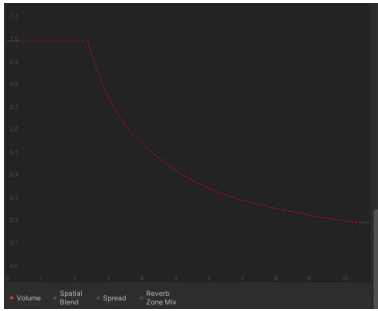
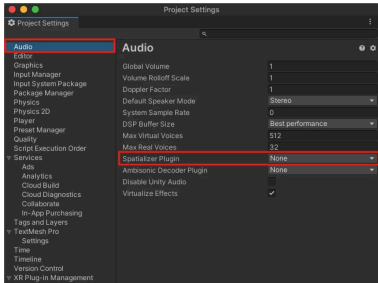
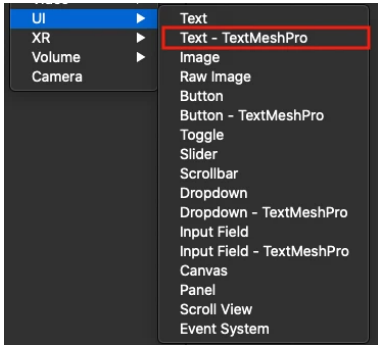
Step	Instructions	Teacher Notes
<p>By the end of this lab, your personal project will have most of its core functionality.</p> <p>This lab will draw on skills learned in the following lessons:</p> <ul style="list-style-type: none"> • Audio & Haptics • Activation Events • Direct & Ray Interactors • User Interface <p>This challenge is part of the Create with VR course.</p>		
Step 1 - Fill in Design Document for events & interactions	<p>Before you resume work on your project, you need a plan for what you're actually going to work on. You will continue filling out your design document so you have a clear plan of action for this lab.</p> <ol style="list-style-type: none"> Pick up where you left off with your design document: <ul style="list-style-type: none"> • Re-open the document you created during Lab 1. Fill out Section 4 (Events and Interactions) of the document: <ul style="list-style-type: none"> • Determine if you will be using direct interactors, ray interactors, or both, including how you would toggle between them. • Describe the haptic / audio feedback, as well as any other 3D sounds you expect in your scene • List specific object Activation functionality, where if the user is holding an object and presses the trigger, something will happen. • Describe the main menu of the app, including what can be done from that menu. • If relevant, describe other UI elements in the scene. (Optional) Add more specific details to your design document: <ul style="list-style-type: none"> • Fill out section 6 (Other features) • Fill out section 7 (Sketch) • Fill out section 8 (Timeline) <p>You should now have section 4 of your Design Document complete, providing direction for this lab where you will be implementing your app's core functionality .</p> <p>Related resources:</p> <ul style="list-style-type: none"> - Oculus Best Practices 	

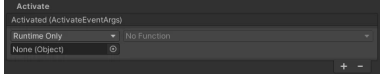
Step 2 - Add haptics and audio	<p>For all of the interactors set up on our controllers, add audio and haptic feedback.</p> <ol style="list-style-type: none"> 1. Add haptic feedback to your controllers: <ul style="list-style-type: none"> Follow the instructions in the Audio and Haptics tutorial. 2. Add audio feedback to your controllers: <ul style="list-style-type: none"> Follow the instructions in the Audio and Haptics tutorial. 3. Add 3D / ambient feedback to the environment, including the use of Reverb Zones: <ul style="list-style-type: none"> Follow the instructions in the Audio and Haptics tutorial. <p>You should now experience haptic and/or audio feedback on desired events with your controllers. There should also be 3D environmental sound in your scene.</p>
Step 3 - Implement event-based interactions	<p>If you have any event-based interactions planned for your app (e.g. when you press the trigger, when the scene loads, when you tilt an object, etc), take the time to get those functional now.</p> <ol style="list-style-type: none"> 1. To add Activation events for certain objects in your app: <ul style="list-style-type: none"> Follow the instructions in the Activation Events tutorial. 2. To add additional event-based functionality to the scene: <ul style="list-style-type: none"> Add actions to other interaction events such as OnHoverEntered, OnSelectEntered, etc. Add new events from the Course Library > Scripts folder such as OnButtonPress, OnSceneLoad, OnTilt, etc. <p>You should now have core event-based functionality implemented in your app.</p>
Step 4 - Set up controller interactors and layers	<p>If you plan on using a combination of Ray and Direct Interactors, you need to set those up for each hand, and if relevant, the ability to toggle between them.</p> <ol style="list-style-type: none"> 1. To implement interactors for each hand: <ul style="list-style-type: none"> Follow the instructions in the Direct and Ray Interactors tutorial. 2. To configure layers so that each interactor can only interact with specific interactables: <ul style="list-style-type: none"> Follow the instructions in the Direct and Ray Interactors tutorial. <p>You should have the appropriate interactors implemented for each hand, including the ability to toggle between them. Using layers, these interactors should only be able to interact with appropriate objects (i.e. sockets can only receive specific objects, rays not being able to grab certain objects, etc).</p>
Step 5 - Create a main menu UI	<p>One User Interface (UI) nearly all apps should have is some kind of main menu UI.</p> <ol style="list-style-type: none"> 1. Create a simple main menu UI: <ul style="list-style-type: none"> Follow instructions in the User Interface tutorial. 2. To add a reset screen UI: <ul style="list-style-type: none"> Follow instructions in the User Interface tutorial. 3. To add additional in-world UI screen: <ul style="list-style-type: none"> Create additional world space canvases and display them in relevant contexts for the user. <p>You should now have at least one main menu UI, which the user can click out of, a reset screen, and any additional UIs required for the functionality of your app.</p>
Recap	<p>New Functionality:</p> <ul style="list-style-type: none"> Haptic and Audio feedback 3D environmental audio Event-based interactions such as activating objects Direct and Ray interactors on controllers Worldspace UI screens <p>New Concepts & Skills:</p> <ul style="list-style-type: none"> Applying more complex interactions in your own VR app. <p>Next Lab:</p> <ul style="list-style-type: none"> Optimization & Lighting

Quiz 2

Lesson Link	Quiz 2 - VR Events & Interactions - Unity Learn	
Length	15 Mins	
<p>In this quiz, you will test the knowledge and skills you learned in Unit 2 related to VR Events and Interactions. This quiz is part of the Create with VR course.</p> <p>This Quiz will assess the skills learned in the following tutorials:</p> <ul style="list-style-type: none"> • Audio & Haptics • Activation Events • Direct & Ray Interactors • User Interface 		

#	Question	Answers	Explanation
1	<p>If you set the Spatial Blend property of an Audio Source to be fully "3D", which two of the following things will happen?</p> <p>Select the two correct answers.</p>	<p>A. Sound volume will be louder in the user's ear that is closest to the audio source.</p> <p>B. Sound volume will increase as the user gets closer to the Audio Source.</p> <p>C. Sounds coming towards the user will be at a higher pitch and sounds going away from the user will be at a lower pitch.</p> <p>D. The quality of the audio will change, depending on the materials of the space.</p>	<p>Making an audio source "3D" makes it change in volume in the left and right ears, depending on the user's position and orientation relative to that sound, just like in real life. Sounds changing pitch depending on their direction relative to the user is the Doppler effect, which is another setting. To edit the quality of the audio to reflect a specific type of space, you can use Reverb Zones.</p>
2	<p>This red line on this graph shows the logarithmic volume Rolloff of an Audio Source. Distance (spread) is on the X axis and Volume is on the Y axis.</p> <p>Approximately how loud will the volume be when the user is about 2 meters away from the source?</p>	<p>A. 0% volume (silent)</p> <p>B. 25% volume</p> <p>C. 50% volume</p> <p>D. 75% volume</p> <p>E. 100% volume (full volume)</p>	<p>You can see that the red line is straight at 1.0 for the first 2 units of distance (meters). The minimum distance of this audio source is set to 2.0, which means that it will be at full 100% volume if the user is anywhere within 2 meters, and then drop as distance increases beyond 2 meters.</p>

	 <p>See high-res image here</p>		
3	<p>In addition to the Spatial Blend property in an Audio Source component, there are additional Audio Spatializer plugins that you can import and enable through the Project Settings.</p>  <p>See high-res image here.</p> <p>What do these spatializer plugins aim to do?</p>	<p>A. Apply subtle effects to the volume, quality, and timing of the audio between your left and right ears to make it sound even more like real-world audio.</p> <p>B. Allow you to play sounds from the controllers in addition to the headset, which allows users to better locate where sounds are coming from.</p> <p>C. Allow you to add a larger number of background ambient sound effects that increase the sense of immersion in the scene.</p>	<p>A sound is different when you hear it directly with your ears than when you hear a recording of that same sound. Audio Spatializers attempt to simulate the changes in audio that occur when you hear sounds directly in the world in order to allow the VR user to better locate the origin of sounds in 3D virtual space.</p>
4	<p>Which of the following is NOT true about using TextMeshPro Text objects over legacy Text objects?</p>  <p>See high-res image here</p>	<p>A. TextMeshPro provides improved clarity & readability at lower resolutions.</p> <p>B. TextMeshPro provides more control over text formatting and layout.</p> <p>C. TextMeshPro objects do not require a Canvas object to appear in your scene.</p>	<p>Just like the legacy text objects, TextMeshPro objects also require a Canvas object.</p>
5	<p>Which of the following definitions of Canvas Render modes is NOT correct?</p>	<p>A. World Space: the Canvas follows the user wherever they are in the world.</p> <p>B. Screen Space Camera: the Canvas is placed a given distance in front of a specified Camera.</p> <p>C. Screen Space Overlay: places</p>	<p>In the World Space render mode, the Canvas will behave as any other object in the scene. The size of the Canvas can be set manually using its Rect Transform, and UI elements</p>

		Canvas UI elements on the screen rendered on top of the scene.	will render in front of or behind other objects in the scene based on 3D placement. This is useful for UIs that are meant to be a part of the world. This is also known as a "diegetic interface".
6	<p>Below you can see the Activated event of a grabbable object.</p> <p>This event will fire when the user presses a particular button while holding this object.</p> <p>What should be assigned to the property that currently says "None (object)"</p>  <p>See high-res image here</p>	<p>A. The controller object that will be holding the object.</p> <p>B. The XR Rig object that will be affected by the object.</p> <p>C. The object with the script component that contains the function you want to run.</p> <p>D. It should be left blank - it will be assigned automatically when the app is run.</p>	When you are adding actions to an event, you first must assign the object that contains the script you want to access. After the object is assigned, you can select a script / component and the function with that script / component you want to run.
7	<p>You have just added a Direct Interactor component to one of your controller objects, expecting to be able to pick up objects directly with your hand.</p> <p>However, when you try to press the grip button near an object, it doesn't work.</p> <p>Which of the following explanations might be causing this issue?</p> <p>Select all correct answers.</p>	<p>A. You forgot to assign a Model Prefab to represent the hand for that XR controller.</p> <p>B. You forgot to check the "Is Trigger" check box on the Sphere Collider component.</p> <p>C. The Interaction Layer Mask property is set to "Nothing"</p> <p>D. The object you're trying to grab does not have a collider on it.</p>	For a Direct Interactor to function, it requires a collidere with "Is Trigger" enabled, it must share an interaction layer with the object it's interacting with, and the object it's interacting with must also have a collider. Having a hand model is not required for this interaction to work.
8	<p>True or False:</p> <p>In order to add a user interface button that works in VR, from the Hierarchy, you should click Create > XR > XR Button.</p>	<p>A. True</p> <p>B. False</p>	The standard button object works in VR and responds to Raycasts just like it would in a typical screen-based experience. You just need to click Create > UI > Button (TextMeshPro).
9	<p>What does the word "haptics" refer to?</p>	<p>A. The use of tactile or touch sensations.</p> <p>B. Tracking of hand movement in VR.</p> <p>C. Tracking hand rotation in VR.</p> <p>D. Audio feedback from user interfaces.</p>	Haptics, in general, is anything related to the sense of touch. Specific to VR or mobile technology, haptics usually refers to

			simulating touch sensations through subtle physical feedback and vibrations.
10	Which of the following statements about VR User Interface best practices is true?	<p>A. Most user interfaces should use the World Space render mode.</p> <p>B. Most user interfaces should use the Screen Space render mode.</p> <p>C. A button's Normal Color, Pressed Color, and Highlight Color should all be the same.</p> <p>D. The interface and text should be as large as possible, even if that requires the user to bend their neck to see it.</p>	Most UI in VR should be in "world space". Objects in "screen space" can be distracting and difficult to interact with. A button should have different interaction colors so the user knows when they've interacted with it. The text should be comfortable to read within the user's field of view, not requiring them to bend their necks in order to see the whole thing.

Unit 3 - VR Ergonomics & Optimization



Unit Link [3 - VR Optimization and Lighting - Unity Learn](#)

Length 9 Hours15 Mins

In this unit, you will focus on ergonomics and optimization for VR in order to make your app as accessible and comfortable as possible..


In these tutorials, you will:

- Improve comfort and accessibility
- Evaluate key performance metrics
- Implement optimized baked lighting
- Configure and build your app for sharing

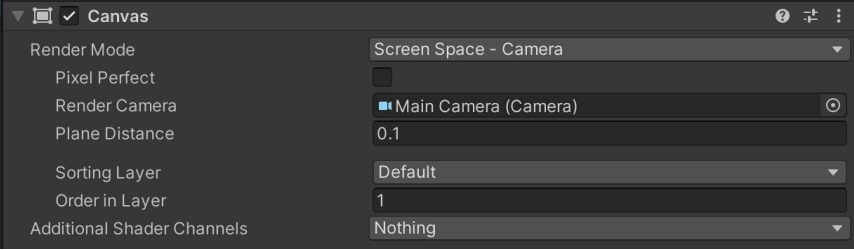
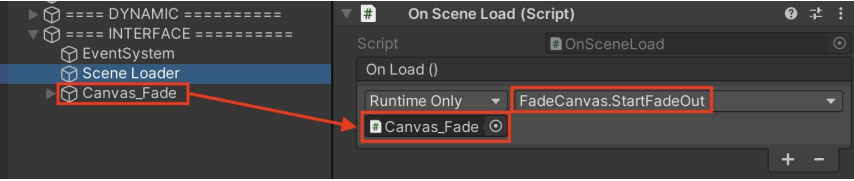
In the Challenge for this unit, you will apply your skills in an industrial training simulation prototype. Then, in the Lab, you'll implement the core functionality of your personal project. Finally, in the Quiz, you will test your new knowledge.

This unit is part of the [Create with VR course](#).

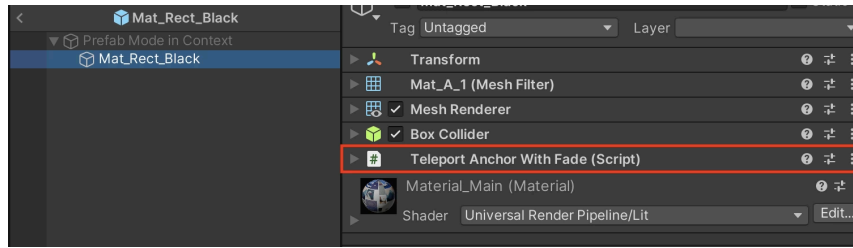
Lesson 3.1 - Comfort and Accessibility

Lesson Link	3.1 - Comfort and Accessibility - Unity Learn
Length	1 Hour 30 Mins
Project Objectives:	
In this lesson, you will learn how to reduce the risk of simulator sickness and increase the inclusivity of your app, allowing it to be enjoyed by as many people as possible. By the end of this lesson, your app will be more comfortable and accessible.	
This lesson is part of the Create with VR course .	
Learning Objectives:	
By the end of this lesson you will be able to:	
<ul style="list-style-type: none"> Explain the importance of limiting a user's field of view in order to reduce the risk of simulator sickness, especially as it relates to locomotion. Recognize opportunities for improving accessibility in an app in order to make it more inclusive, including motor, audio, visual, and cognitive accessibility. Modify a project in order to make it more comfortable or accessible in order to allow it to be enjoyed by as many people as possible. 	

Step	Instructions	Teacher Notes
Step 1 - Create a black canvas for fading	<p>When the scene loads, it can be a little jarring to just appear in a new world. It is generally good to fade into new environments.</p> <ol style="list-style-type: none"> Make a new Screen Space canvas to fade in from: <ul style="list-style-type: none"> In the Hierarchy, Create > UI > Canvas and rename it "Fade_Canvas". Make the canvas render directly in front of the camera: <ul style="list-style-type: none"> In the Canvas object, in the Canvas component, set the Render Mode to Screen Space - Camera. Set the Render Camera property to your Main Camera. Make the canvas black: <ul style="list-style-type: none"> Right-click your Fade_Canvas object and create a UI > Image. Set the Color property to black. Set the X Scale and Y Scale properties to a large number (e.g. 100) to make sure the canvas covers your whole field of view. Make the canvas appear directly in front of the Main Camera and in front of all other objects: <ul style="list-style-type: none"> In the Canvas component, set the Plane Distance to 0.1. Then set the Order in Layer to 1. 	<p>Explain: Screen space Camera vs Overlay vs world space?</p> <p>Warning: No need to make an XR Canvas - just simple canvas will work.</p> <p>Warning: You don't want the plane distance too small or you might clip through it.</p>

	 <p>The black canvas should now completely cover the field of view when you run the program.</p>	
<p>Step 2 - Fade in on scene load</p>	<p>Now that you have the black canvas set up, you need to fade from it on scene load.</p> <ol style="list-style-type: none"> 1. Allow the canvas and all of its elements to fade: <ul style="list-style-type: none"> On the Fade_Canvas object, add a Canvas Group component and a Fade Canvas component. 2. Make an event that detects when the scene has loaded: <ul style="list-style-type: none"> In the Hierarchy, add an empty object and rename it "Scene Loader". It will hold your scene loading events. Add an On Scene Load component to the Scene Loader object. 3. Fade in from black on Scene Load: <ul style="list-style-type: none"> On the Scene Loader object, in the On Load event, click the + to add a new function. Assign the Canvas_Fade object. Select the FadeCanvas > StartFadeOut () function. 4. Hide the black image in scene view by default: <ul style="list-style-type: none"> Set the Canvas Group component's Alpha value to 0. <p>The scene should now appear to fade in from black when the scene loads. You can test this using the reset scene UI screen you made.</p>  <p>Related Resources:</p> <ul style="list-style-type: none"> Screen Fader with custom shader tutorial 	<p>Explain: What is a canvas group component? It allows you to control all elements within a canvas.</p> <p>Explain: The FadeCanvas script is a custom script.</p>
<p>Step 3 - Teleporting with fade</p>	<p>Another opportunity to improve player comfort through fading is during teleporting. Typically, there is a short fade to reduce the illusion of movement ("vection") when teleporting and improve comfort.</p> <ol style="list-style-type: none"> 1. Add fading functionality to your Teleportation Area (rug): <ul style="list-style-type: none"> On your Rug prefab, remove the default Teleportation Area component and replace it with the custom Teleport Area With Fade component. At the bottom of the new component, assign the same Interaction Layer Mask, Custom Reticle and Match Orientation properties to match the original component. 2. Add fading functionality to your Teleportation Anchor (mats): <ul style="list-style-type: none"> Enter Prefab Mode on one of your mats in order to apply changes to all mats at once. Remove the default Teleportation Anchor component and replace it with the custom Teleport Anchor With Fade component. 	<p>Explain: Why is this more comfortable?</p> <p>Explain: How does this work? It's looking for the object with the fade canvas component in the scene.</p>

- Edit the new component's properties to match the original component.
- **Note:** make sure that you have applied the changes to all mat prefabs or prefab variants.



Your teleportation anchors and teleportation area should now quickly fade to black when you teleport.

Step 4 - Add a settings panel

In order to add some settings, you will add a new panel that can be accessed from the main menu panel.

1. Add a new settings button:

- In the Hierarchy, right-click **Welcome Background** object and select **UI > Button - TextMeshPro** to add a button.
- Rename it "Settings Button".
- Set the **Source Image** property to the gear icon
- Delete the default Text so that it has no text on it.
- Resize and reposition the button so it fits nicely on in the bottom-right corner of the menu.

2. Create a new settings screen:

- In the Hierarchy, **duplicate** the **Welcome Background** object.
- Rename the copied object "Settings Background"
- Temporarily deactivate the welcome panel so you can only see the settings screen for easier editing.

3. Add placeholder content to the settings screen:

- Replace the contents of the panel with three placeholder text objects ("Setting 1" / Setting 2" / "Setting 3")
- Add a **Toggle** object next to each setting, deleting the default text that is generated by the toggle.
- You may want to parent each text object to group them together.
- **Note:** Use TextMeshPro Text for the labels instead of the default Text label that comes with the toggle.

4. Replace the settings gear icon with an "X" icon:

- In the Gear icon's Image component, change the **Source Image** property from **Icon_Gear** to **Icon_Close**.

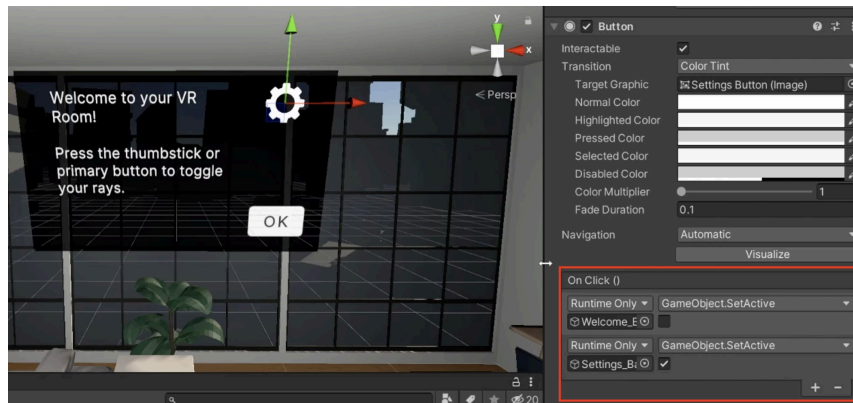
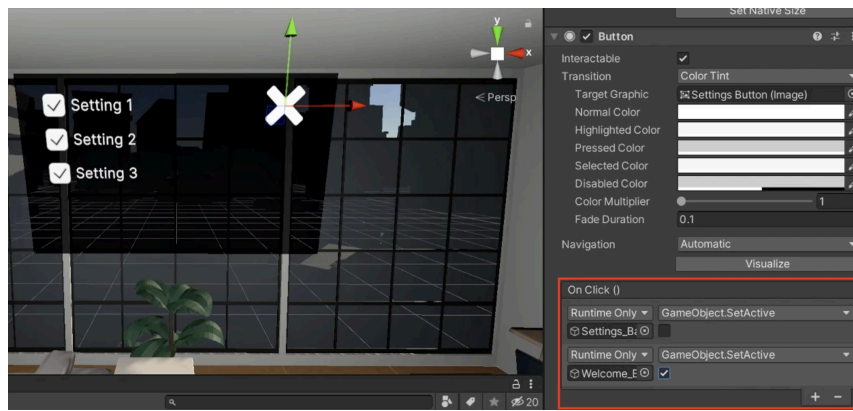
5. Allow the user to go back and forth between these menus:

- In the Settings button's **On Click** event, set the settings screen as active and the welcome screen as inactive.
- In the Close button's **On Click** event, set the settings screen as inactive and the welcome screen as active.

6. Make the reset menu hide the settings menu when it appears:

- In the XR Rig's **On Button Press** event, add a new action that disables the settings screen.

Demo: You may need to re-organize your welcome panel a bit to accommodate the settings button.



You should now be able to access the Settings menu from a gear icon in the main menu, which has 3 placeholder toggle settings. You should also be able to return to the welcome panel from the settings by pressing a Close button.

Step 5 - Add a snap turning setting

One of the easiest ways to increase accessibility is allowing for more ways to navigate the environment. You will allow the user to enable or disable snap turning.

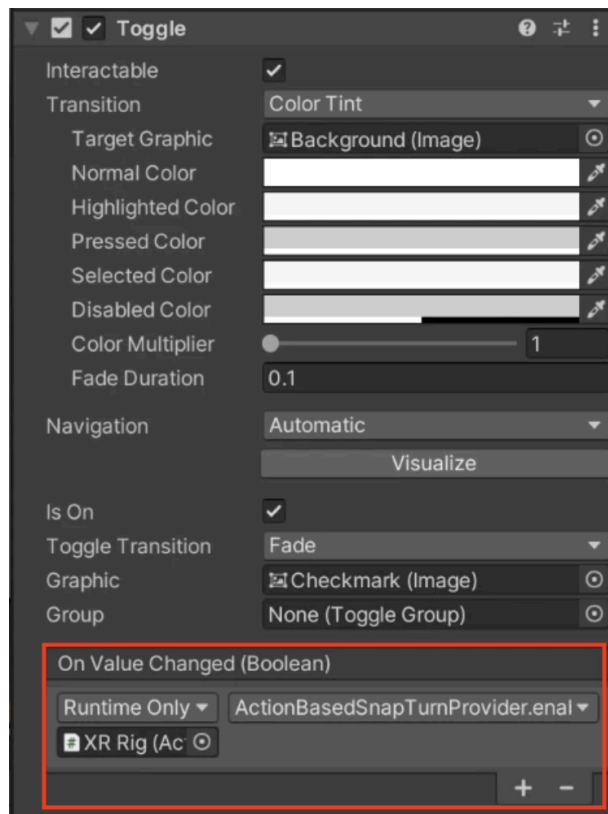
1. Replace the placeholder setting:

- Edit the first setting text to say "Snap Turning".
- In the Toggle component, make sure the **Is On** property is enabled, since Snap Turning is enabled by default.

2. Link the toggle to the Snap Turn component:

- In the Toggle component, for the **On Value Changed** event, click the **+** to add a new action.
- Assign the XR Rig to the Object slot, since it contains the Snap Turn provider component.
- Select the **Action Based Snap Turn Provider > enabled (Dynamic Bool)** function .
- **Note:** Make sure to select the "enabled" property that is a dynamic bool (boolean) - this means that it will be affected dynamically by the state of the toggle.

Explain: Why use the dynamic bool? Because that's tied to the value that's passed in by the toggle.



You should now have a functional check box that enables and disables snap turning.

Step 6 - Add a distance grab setting

Another setting that can drastically increase accessibility is the ability to grab objects at a distance using rays.

1. Set up how your setting should appear by default:

- In the Toggle component, enable or disable the **Is On** property.
- Change the text of the Toggle to "Distance Grab".

2. Make sure all of your grabbable objects can be accessed on the same layer:

- In each object's XR Grab Interactable component, make sure the **Interaction Layer Mask** property includes the **Default** layer.

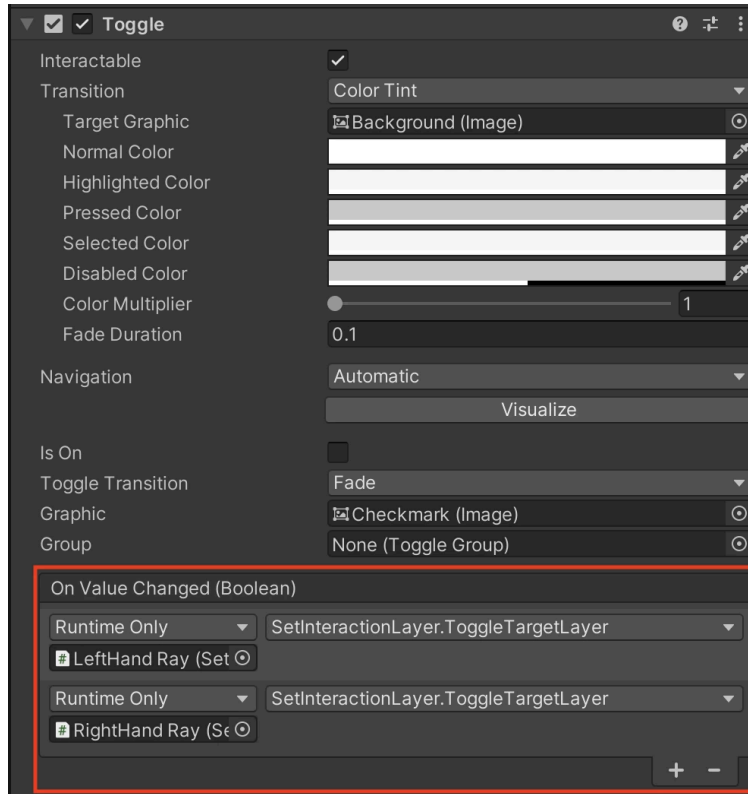
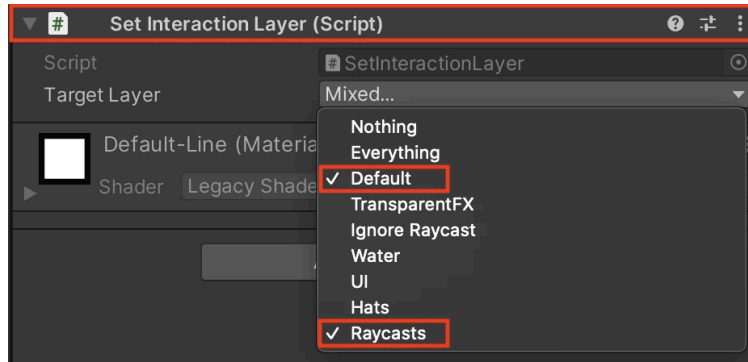
3. Give your rays the ability to change to a different set of layers:

- In the Hierarchy, select both the **LeftHand Ray** and **RightHand Ray** objects.
- Add a **Set Interaction Layer** component to both objects.
- Set the **Target Layer** property to include the **Default** layer and the **Raycasts** layer, which would allow them to grab objects.

4. Make the "Distance Grab" toggle trigger these target layers:

- In the Toggle component, in the **On Value Changed** event, click the **+** two times to add two new actions (one for each hand).
- Assign the **RightHand Ray** to one object slot and the **LeftHand Ray** to the other object slot.
- For both actions, select the **SetInteractionLayer > ToggleTargetLayer (Dynamic bool)** function.
- **Note:** You must select the **Dynamic bool** option in the upper section of the list in order to tie the function to the value of the toggle.

Explain: Other ideas for settings include continuous movement, closed captions, rig height, tooltips, etc



You should now have a toggle that enables and disables the ability to grab objects with rays from a distance.

Related resources:

- [VR Accessibility](#)

Recap

New Functionality:

- Fade in from black
- Teleportation fading
- Settings menu

New Concepts and Skills:

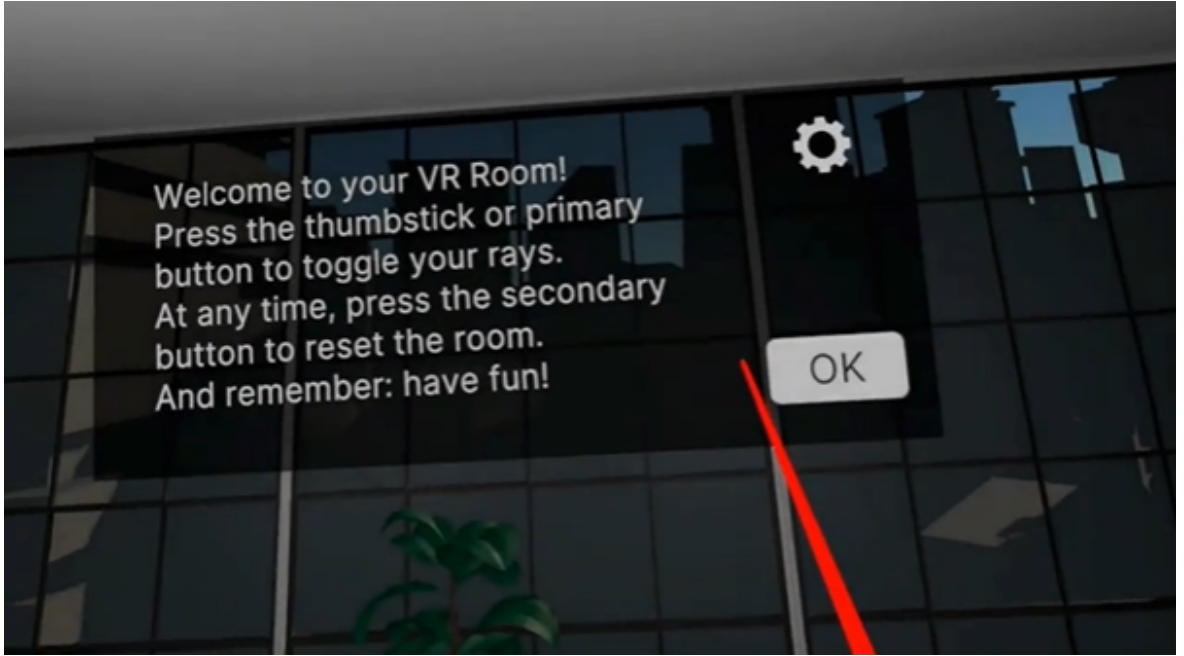
- VR comfort
- VR accessibility

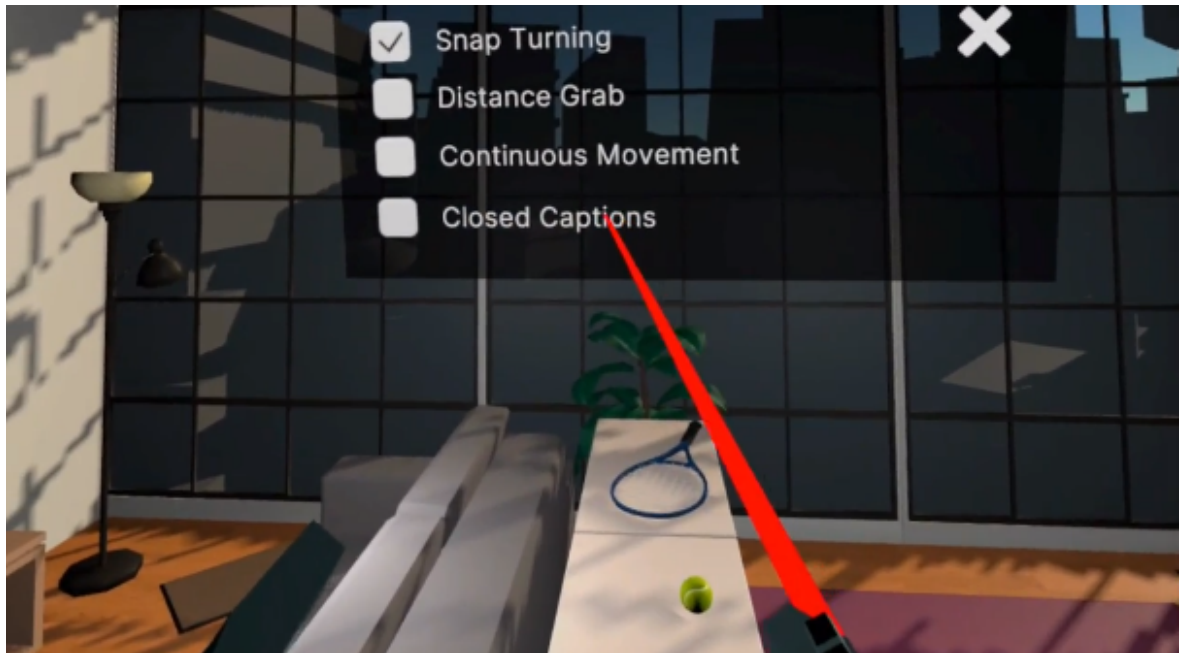
Next Lesson:

- Optimization

Extensions

If you want to further develop your skills, explore new concepts, and improve your project,

	<p>check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.</p>
Extension - Voiceover narration	<p>1. Add voiceover narration [Easy]</p> <p>Add audio narration that plays at the start of your scene, reading the text on the main menu:</p> <ul style="list-style-type: none"> • Use the OnSceneLoad script to trigger this • Bonus - the audio should only play  <p>See video of audio narration in action here.</p>
Extension - Closed captions setting	<p>2. Add a closed captions setting [Medium]</p> <p>Add closed captions to improve audio accessibility:</p> <ul style="list-style-type: none"> • For ambient sound like the fireplace, add closed captions to indicate what the sound is at the sound's location. • Bonus - have the closed caption "billboard" towards the player, so that it's always facing towards the player and is therefore always readable (use the "LookAtPlayer" script from the Course Library or write your own script). • Bonus - for your TV, use a video with words in it and add closed captions that go along with the words of the video (the Unity-shape-the-world video provided has voiceover) [Requires programming].



See lesson related videos [here](#).

Extension - Continuous movement setting

3. Add a continuous movement setting [Medium]


Add a setting to toggle on and off continuous movement:

- Use the same technique you did to toggle snap turning.
- You will need to add a Continuous Move Provider component and a Continuous Turn Provider component to the XR Rig.
- Bonus - You could have a single toggle or drop-down that alternates between (a) continuous movement and turning vs (b) snap turning and teleporting.



See lesson related videos [here](#).

Lesson 3.2 - Optimization

Lesson Link	3.3 - Lighting - Unity Learn	
Length	1 Hour 30 Mins	
Project Objectives:	<p>In this lesson, you will learn about each of the key performance metrics for VR (fps, polycount, and draw calls) and how to ensure these metrics are optimized. By the end of this lesson, your app will be more optimized for performance.</p> <p>This lesson is part of the Create with VR course.</p>	
Learning Objectives:	<p>By the end of this lesson you will be able to:</p> <ul style="list-style-type: none">Recall the key metrics for performance, including fps, tris, and draw calls, in order to locate performance targets for your device.Explain the importance of fps as it relates to comfort of a VR experience in order to prioritize it appropriately in development.Identify key performance metrics in the Game Stats window in order to determine whether or not your app is currently hitting its targets.Understand factors that contribute to high draw calls in your scene in order to locate potential areas for optimization (including static batching, sharing materials, etc).Understand factors that contribute to high polycount in your scene in order to locate potential areas for optimization (including particles, high-poly meshes, etc).Understand factors that contribute to the performance of textures in your scene in order to locate potential areas for optimization (including texture atlases, mip maps, anisotropic filtering, etc.).Understand the impact of particle effects and post-processing on performance in order to avoid optimization issues.Set up anti-aliasing in your scene in order to improve the appearance of jagged curves or angles in the scene.	
		

Step	Instructions	Teacher Notes
Step 1 - Take stock of current fps	Before you start optimizing your code, you will want to identify your performance goals and how to evaluate if your app is meeting them. 1. Determine your device's target frames per second (FPS):	Explain: Importance of optimization - similar to mobile, but

- Use the links below as reference and write down your target FPS.

2. Determine the frames per second (fps) running on your device:

- From the **Course Library > Prefabs > Testing**, drag the **FPS Overlay UI** into your Hierarchy.
- Assign your **Main Camera** to the **Render Camera** property.
- Run your application to see your FPS.
- **Note:** If you are running the app through the Unity editor, this will not be an accurate prediction of the FPS.

3. View the FPS in the Unity Stats window:

- In **Game View**, in the top-right corner, enable the **Stats** window, and locate the FPS number.

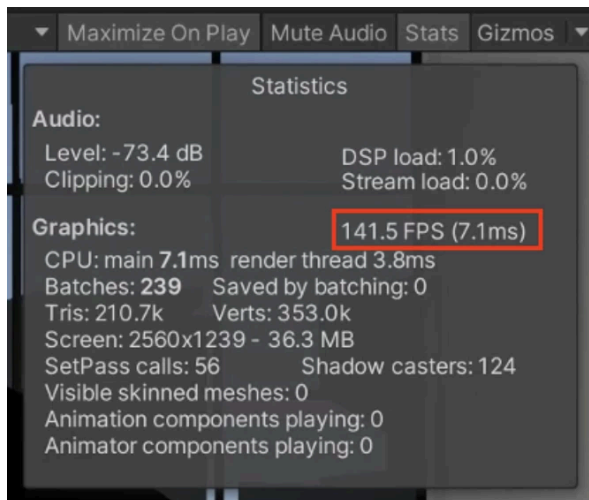
4. Explore the Unity Profiler:

- Click the Play button.
- From the top menu bar, select **Window > Analysis > Profiler**.
- Click the red circular record button in the Profiler window.

5. To test the true frames per second running on a mobile headset:

- Build and run the app on your device with no connection to the computer.

You should now know where to find target metrics for your device and how to find the frames per second on your device.

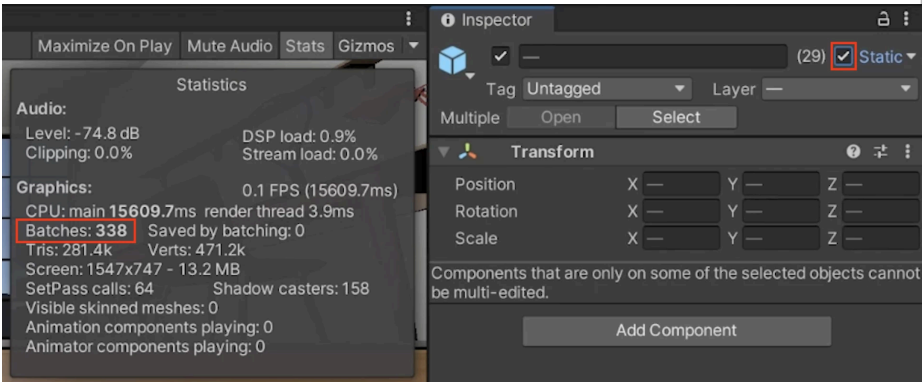


have to render 2x and need high frame rate.

Explain: Everything that you'd want to google for "mobile optimization" is basically applicable here, too

Explain: Relationship between frame rate and ms per frame - just inverse (divide 1000 over fps).

Explain: You can also download / install apps that monitor FPS / GPU / CPU (e.g. ovr metrics).

	<p>Related resources:</p> <ul style="list-style-type: none"> Target performance metrics <ul style="list-style-type: none"> HTC Vive performance targets Oculus performance targets Unity Tools <ul style="list-style-type: none"> Stats window Frame debugger Profiler 	
<p>Step 2 - Optimize draw calls</p>	<p>In order to render a single frame, Unity “draws” things to the screen in “batches” of objects. Every time Unity “calls” on the engine to “draw” something to the screen is called a “draw call”. If you can minimize draw calls, you can improve performance.</p> <ol style="list-style-type: none"> Determine your device’s target draw calls (batches): <ul style="list-style-type: none"> Consult your platform’s website to find your target “Draw Call” metric. Find the number of draw calls in your scene: <ul style="list-style-type: none"> Open the Game Stats window and locate the “Batches” (draw call batches) value. Visualize how batching and draw calls can affect frame rate: <ul style="list-style-type: none"> In the top menu, select Window > Analysis > Frame Debugger. Click Enable. Use the left/right arrows in the Frame Debugger window and watch in the Game view to see the step-by-step creation of a single frame. Make sure “Static Batching” is enabled: <ul style="list-style-type: none"> From the top menu, select Edit > Project Settings > Player panel. In the Other Settings foldout, make sure Static Batching is enabled. Improve batching by marking unmoving objects as “Static”: <ul style="list-style-type: none"> Make sure your Hierarchy is organized so that all static objects can be selected easily. Select all static objects (including lighting objects like lamps and chandeliers). At the top of the Inspector select the Static checkbox. When prompted, choose “Yes, change children”. Notice the effect this has on Batches when you click Play. If your Batches number is still very high (over 175): <ul style="list-style-type: none"> Consider removing the mirror object from the scene. 	<p>Explain: What are draw calls?</p> <p>Explain: Why “batches” instead of draw calls? - Unity batches things when it draws them to be more efficient</p> <p>Explain: What does dynamic batching do? Good for particle systems bc dynamic geometry can be built into shared vertex buffers.</p> <p>Demo: Importance of sharing materials over a large number of meshes. Batching can only be done with shared materials. If you don’t, draw calls go up.</p> <p>Demo: Look at what happens when you remove the mirror .</p>
	 <p>You should have now reduced the number of draw calls through batching objects with shared materials.</p> <p>Related resources:</p> <ul style="list-style-type: none"> - Draw call batching 	

Step 3 - Minimize polycount

The most straight-forward performance metric is the poly (polygon) count. The poly count is often measured in "Tris" counting triangles or "Verts" counting vertices.

1. Determine your device's target polycount (triangles):

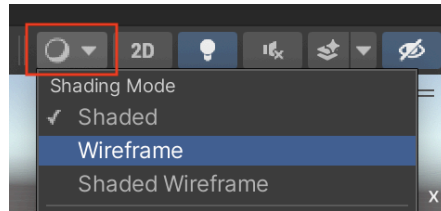
- Consult your platform's website to find your target "triangles" metric.

2. Find the number of triangles in your scene:

- Open the Game Stats window and locate the "tris" (triangles) value.

3. Visualize your triangles more clearly:

- From the **Scene** view, change from the "Shaded" draw mode to the **Wireframe** draw mode.
- **Note:** In Unity 2021 LTS: the draw mode dropdown is represented by a circular icon at the top of Scene view:



4. Determine the number of triangles for an individual object:

- Select an object.
- In its Mesh Filter component, click its Mesh.
- Select that Mesh in the Project window to view its triangle count in the Inspector.
- **Note:** Anything under 1,000 tris is pretty "low poly"

5. Notice how your tris change depending on what objects are in frame:

- Monitor the stats window as you rotate the XR Rig around.

6. If you are not hitting the suggested target for Tris:

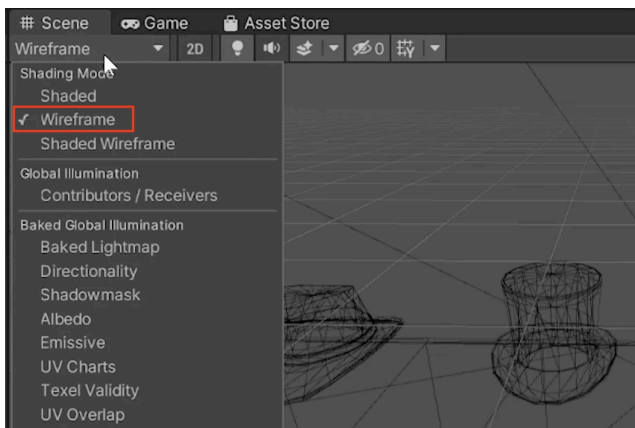
- Remove or replace assets with high polygon counts until you reach your target
- You can also try removing the mirror from your scene.



Demo: Why triangles? In wireframe mode you can see how Unity converts everything into triangles.

Demo: Duplicate a plant a bunch of times (like 100 times), run your app on device and look at the fps drop as you look away from vs looking at them.

Demo: You can also quickly see the effect on tris by disabling objects one at a time to see what impact it has.



You should now be hitting your target for Tris and have an understanding of where you are spending your tri budget.

Related resources:

- [Occlusion culling](#)

Step 4 - Optimize textures

In addition to evaluating the draw calls Unity makes, it is also important to consider the size of the textures used in your scene.

1. Add a painting to the scene to better analyze the impact of texture settings:

- From the **Course Library > Prefabs > Art**, add a piece of Art to a wall in your scene.

2. Access the material for the objects in your scene:

- In the painting's Mesh Renderer component, expand the **Material** section at the bottom of the inspector..

3. Locate the texture atlas used for the material:

- From the material's **Surface Inputs** foldout, click the thumbnail next to **Base Map** to locate the texture in the Project window.
- Click this texture to view the texture's **Import Settings**.

4. Experiment with the texture settings:

- At the bottom of the texture's settings, adjust the **Max Size**, **Resize Algorithm**, **Format**, and **Compression** properties.
- Click **Apply** to see the impact each setting has on the visuals in your scene.
- Once you have observed the difference, revert back to the original settings.

5. Enable "mip maps": to automatically reduce the resolution of textures seen from a distance :

- Locate and select the InteriorColorSwatch texture in the Project window.
- In the Inspector, expand the **Advanced** fold-out, and select **Generate Mip Maps**.
- Click **Apply**. This will generate and apply lower quality textures on farther-away objects.

6. Enable "Anisotropic Filtering" to improve the visuals of textures when viewed at a shallow angle:

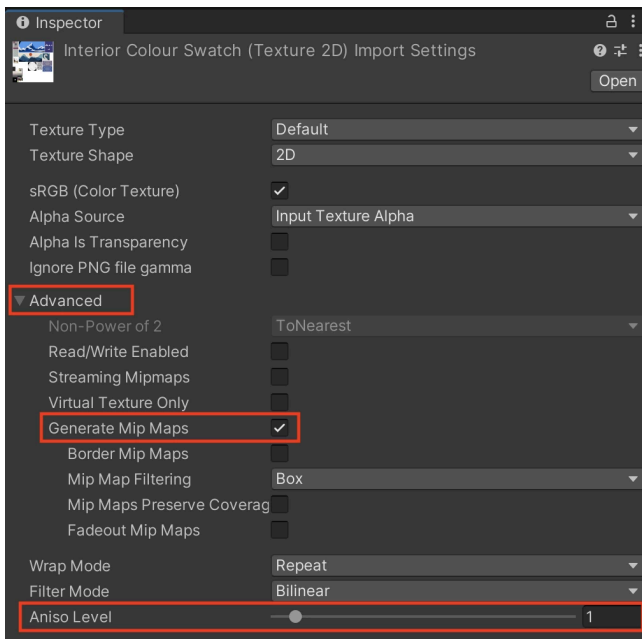
- Locate and select the **WoodFloor_Texture** in the Project window.
- In the Inspector, drag to increase the **Aniso Level** slider.
- Click **Apply**. This will increase Anisotropic Filtering to improve texture visuals when viewed at shallow angles.

You should now know which settings exist to optimize textures for performance and/or visual quality.

Explain: Texel Density = pixels per world unit. You need higher texel density for detailed objects you can view from close up (e.g. a strawberry you can grab vs a building in the background)

Explain: What are mipmaps? Lists of progressively smaller versions of an image. Unity automatically uses a smaller version of the texture when it is far away from the Camera.

Explain: Textures and materials are not a huge problem in this project because we've set up the assets properly - but could be a HUGE problem if you download other assets.



Related resources:

- [Importing Textures](#)
- [Texture import settings \(and mip maps\)](#)
- [Anisotropic Filtering](#)

Step 5 - Optimize particles and post-processing

Particles and post-processing also impact performance. You will look at how these can affect frames per second and ensure they are not causing performance issues in the project.

1. Optimize the particle's shader so that it does not respond to dynamic lights:

- Select the Particle_Fire object.
- Locate the **Shader** property drop-down at the bottom of the Inspector.
- Make sure your particle is using the **Universal Render Pipeline > Particles > Unlit** shader.

2. Experiment with Post-processing:

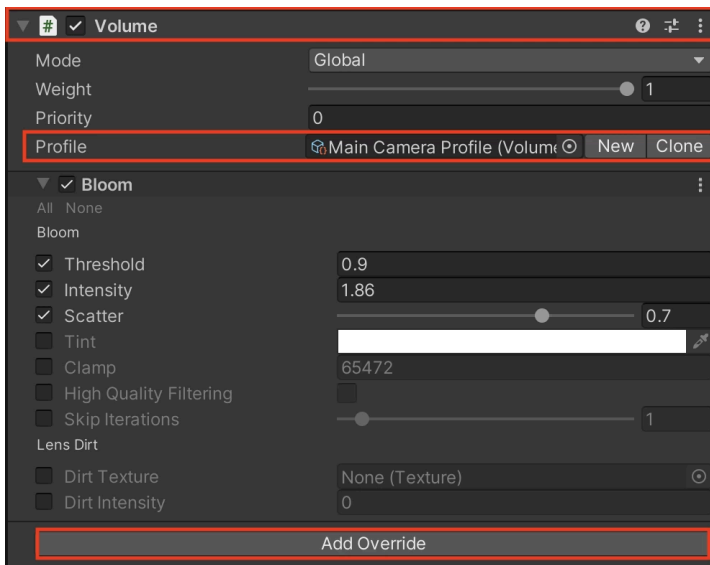
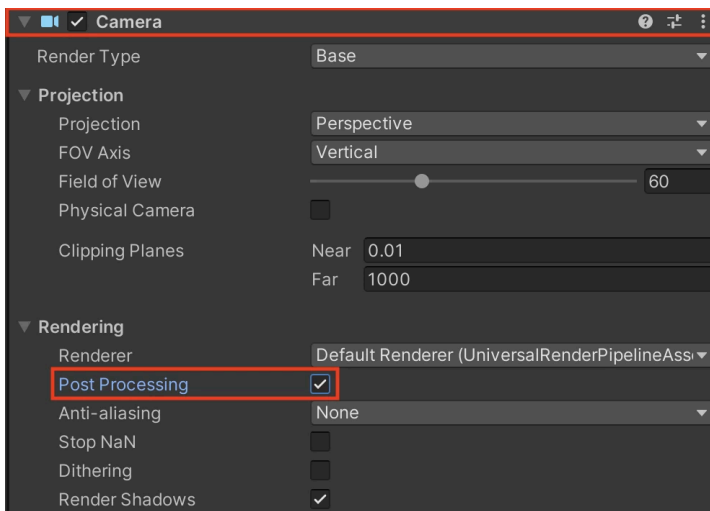
- In your **Main Camera** object's Camera component, enable the **Post-processing** setting.
- Add a Volume component.
- Click the **New** button to create a new Post-Processing profile.
- You can then experiment by clicking the **Add Override** button and enabling new effects.
- **Warning:** If you are going to add any post-processing effects, color adjustments are a good option. Lens distortions and motion blurs induce cyber sickness; avoid using these effects.

You should now have appropriate particles and post-processing for your app.

Explain: Utilize mobile or unlit shaders over the standard shader if possible

Demo: Focus camera on fire, then increase emission > rate over time (with max particles way up) and watch tris spike.

Warning: Post-processing can make FPS drop dramatically.



Related resources:

- [Choosing shaders in URP](#)
- [Post-processing in the Universal Render Pipeline](#)
- [Screen space processing](#)

Step 6 - Enable anti-aliasing

You may have noticed that the edges of some of the 3D objects appear jagged. This can be immersion-breaking in VR. We can smooth these jagged edges with anti-aliasing.

1. Access the render pipeline settings:

- In the Project window, search for "UniversalRenderPipelineAsset" and select it to see its properties in the inspector.

2. Enable anti-aliasing in the Universal Render Pipeline (URP) asset:

- In the **Quality** fold-out, change the **Anti-Aliasing** setting from Disabled to **4x**.

3. If you want to add additional anti-aliasing on the main camera:

- On your **Main Camera** object, in the Camera component, expand the **Rendering** fold-out.
- Change the **Anti-Aliasing** setting to "Fast Approximate Anti-Aliasing (FXAA)".
- Make sure the **Post-Processing** setting is enabled to be able to see the effect.

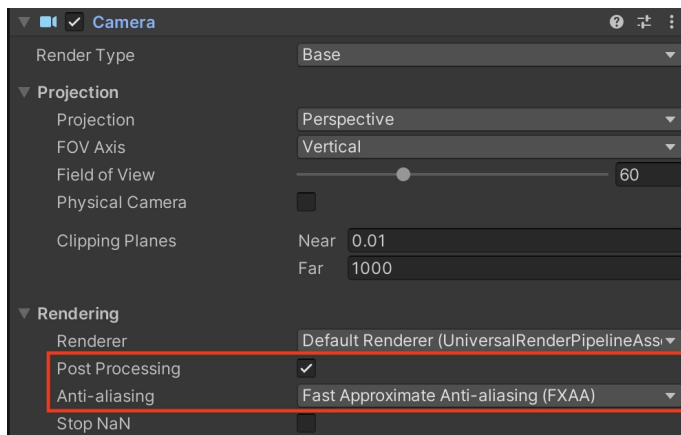
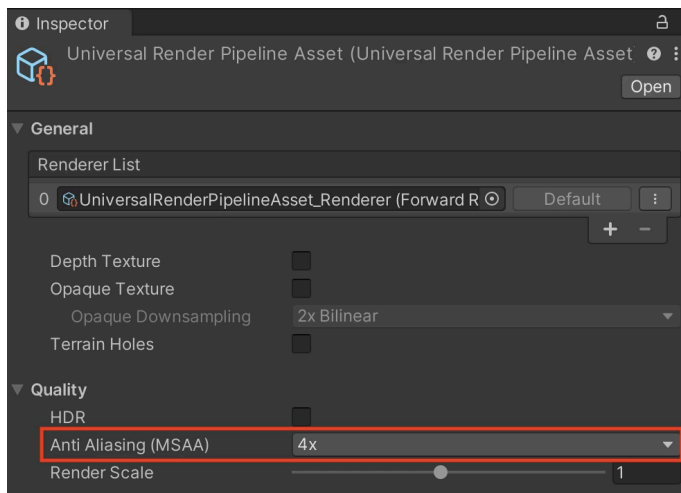
Explain: What is aliasing? What is anti-aliasing?

Warning: Make sure to select the `_Asset` file, not the `_Asset_Renderer`.

Demo: In Game view, look at the edge of an object and see the difference of when it's enabled vs disabled

Explain: SMAA (Subpixel Morphological Anti-Aliasing) vs FXAA (Fast

- **Note:** This may have a significant impact on FPS.
- 4. Apply anti-aliasing to the mirror:**
- Locate the "Mirror_Texture" in your project.
 - Change the **Anti-Aliasing** property from "None" to "4 samples".



Your main camera and your mirror should now have much smoother edges than they did before.

Related resources:

- [Anti-Aliasing](#)

approximate anti-aliasing) - just a different algorithm for calculating the pixel colors. FXAA is better for performance.

Recap

New Functionality:

- Higher frame rate

New Concepts and Skills:

- Target metrics for VR
- Polycount
- Draw calls
- Texture and materials optimization
- Particles and FX optimization
- Anti-aliasing

Next Lesson:


- Lighting

Extensions

If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below.

	Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.
Extension - Learn more about VR optimization	1. Learn more about VR optimization [Medium] Read more about optimization techniques and strategies for VR. Note that a big part of optimization is Lighting, which we will tackle in the next lesson: <ul style="list-style-type: none"> • Seven Stages of Optimization for Mobile VR • Optimizing your VR/AR Experiences • Optimizing Graphics Performance
Extension - Platform-specific tools	2. Research platform-specific tools [Hard] Depending on what platform you're using, there may be specific tools available for monitoring performance on your device: <ul style="list-style-type: none"> • If it requires a separate application download, you may need to "sideload" the app onto your device rather than downloading it from an official store (e.g. using SideQuest for Oculus). • Examples: <ul style="list-style-type: none"> ◦ Oculus - Oculus Developer Hub and the OVR Metrics Tool ◦ HTC Vive - Android Systrace ◦ Valve / SteamVR - Frame Timing Tool
Extension - Using the Profiler window.	3. Explore the Unity Profiler [Expert] Get more comfortable with the Unity Profiler window. <ul style="list-style-type: none"> • Unity Profiler Window manual • Optimization guide using the Profiler • Diagnosing performance issues

Lesson 3.3 - Lighting

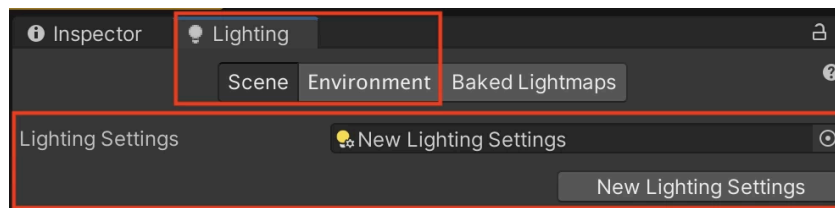
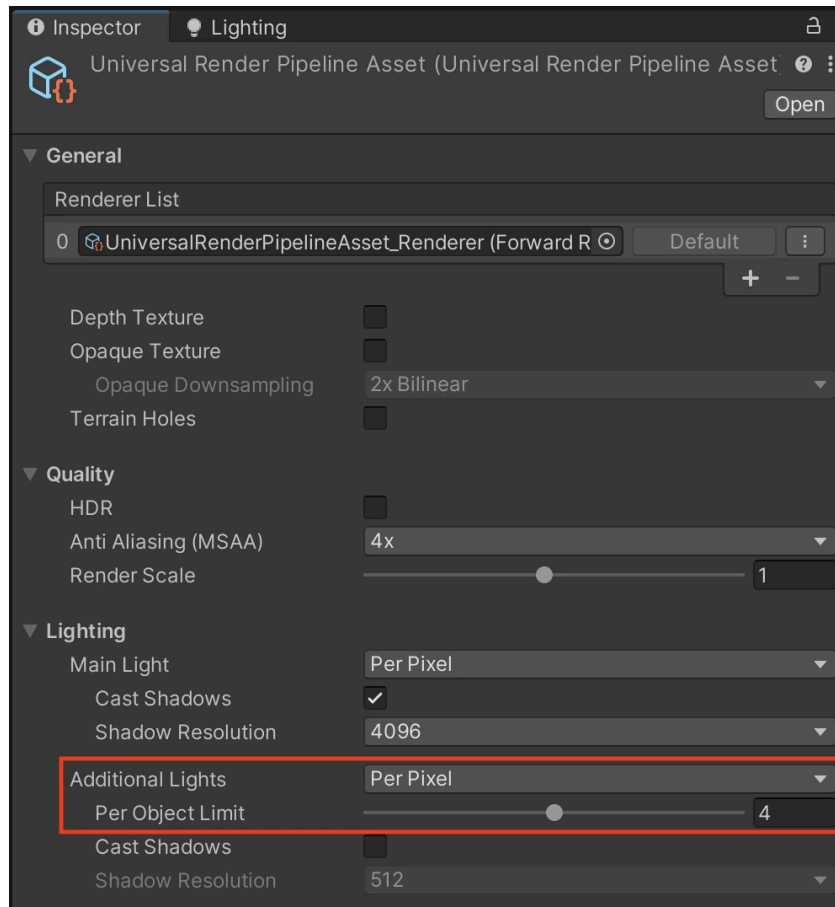
Lesson Link	3.3 - Lighting - Unity Learn
Length	1 Hour 30 Mins
<p>Project Objectives: In this lesson, you will learn how to use light modes, lightmapping, and light probes in order to optimize the lighting for VR. By the end of this lesson, the lighting in your scene will be both beautiful and performant.</p> <p>This lesson is part of the Create with VR course.</p> <p>Learning Objectives: By the end of this lesson you will be able to:</p> <ul style="list-style-type: none"> • Understand the differences between real-time, mixed, and baked lighting in order to select a lighting strategy that makes sense for your app. • Take the necessary steps required for lightmapping in order to bake the lighting in your scene, (i.e. setting light mode to baked and setting objects as static). • Add basic environmental lighting using point lights and spot lights in order to give your scene the desired appearance. • Understand the effect that mixed lighting has in a scene in order to achieve specific real-time lighting effects (e.g. real-time shadows). • Implement Light Probes in order to create optimized real-time lighting effects from baked lights. 	

Step	Instructions	Teacher Notes
Step 1 - Experiment with the lighting settings	<p>Before you start editing the lighting of the scene, you will create a new Lighting Settings asset and get familiar with the settings at your disposal.</p> <ol style="list-style-type: none"> 1. Access the render pipeline settings: <ul style="list-style-type: none"> • In the Project window, search for and select the "UniversalRenderPipelineAsset". 2. Experiment with the render pipeline settings: <ul style="list-style-type: none"> • In the Inspector for the URP Asset, in the Lighting section, play with the settings. • Be sure to include experimenting with the Per Object Limit slider to appreciate the limitations it places on complex lighting. 3. Create new Lighting Settings for your scene: <ul style="list-style-type: none"> • Click Window > Rendering > Lighting to open the Lighting window. • Dock the window next to the Inspector, then click the New Lighting Settings button to create a new profile. 	<p>Demo: At the start of this, demo a before/after realtime vs baked lighting and how with baked lights, shadows don't change when you move things - don't make shadows for dynamic objects</p> <p>Warning: This is complex and is an entire discipline in and of itself - we won't be going over every setting, but will be lots of additional links for you to explore if you want.</p> <p>Explain: The per object</p>

4. Experiment with the environmental lighting in your scene:

- At the top of the Lighting window, select the **Environment** tab
- Experiment with the settings to see the impact they have on your scene.

You should now be able to locate and adjust the lighting settings from within the Lighting window and the URP Asset.



Related resources:

- [Universal Render Pipeline Asset](#)

limit drastically affects how realistic the lighting looks, but is also extremely expensive - so you want the realistic light combinations, but without the cost.

Warning: The Lighting settings asset will be generated in whatever folder you're currently in - would recommend in the Scenes folder.

Step 2 - Prepare and bake a basic lightmap

The time has come to attempt your first lightmap bake. There are just a few steps you need to take beforehand.

1. Allow all of your lights to be baked:

- Locate and select each light in the scene that will not move or turn on and off. You can do this by searching "t:light" in the Hierarchy search bar.
- In their Light components, inside the **General** fold-out, change the **Mode** setting from **Realtime** to **Baked**.

2. Allow objects in your scene to have light baked onto them:

Demo: What is realtime vs baked lighting - look at how objects shadows / colors brightness changes as you move them around the scene
Demo: Look how long it takes when you try to bake with background

- In the Hierarchy, select all non-moving, static objects.
- At the top of the Inspector, make sure the **Static** check box is selected.
- **Note:** Remember to select the objects in your Lighting section (e.g. lamps, sconces, chandeliers, etc)

3. Remove the background and foreground objects from your lightmap to reduce bake times:

- Select each large background object.
- In its Mesh Renderer component, for the **Lightmapping** section, reduce the **Scale in Lightmap** property to 0.

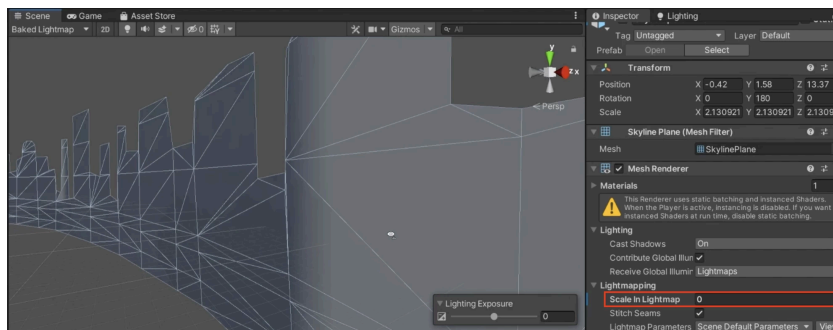
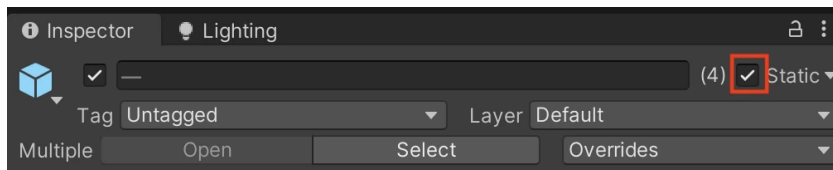
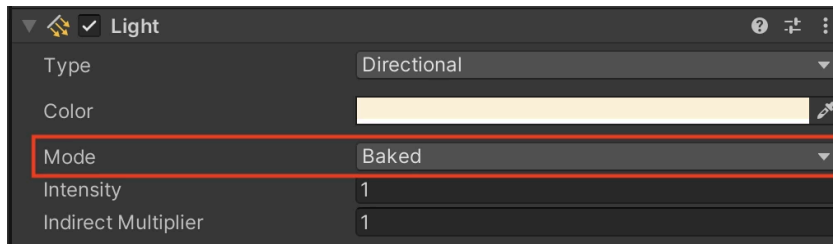
4. Try baking your lights:

- Navigate to the **Lighting** window.
- At the bottom of the **Scene** tab, click **Generate Lighting**.

5. Temporarily minimize the time to generate your lightmap:

- Reducing the **Lightmap Resolution** to a value between 5-10.
- Reduce some of the other values which contribute to lightmap quality (samples, bounces, etc).

You should now have a low resolution baked lightmap in your scene.



Related resources:

- [Light modes: Baked, Mixed, Realtime](#)

buildings, then change lightmap scale to 0.

Demo: Look how when you make something static, Lightmapping settings appear in Mesh Renderer component, allowing you to edit.

Demo: Use the drop-down in the top-left of Scene view to select the "Baked Lightmap" view - notice how the lightmap disappears when an object is not marked static.

Step 3 - Adjust scene lighting and re-bake


Now that you have attempted your first bake, you can take some time to properly light your scene and try again.

1. Clear your lightmap and view lighting changes in real time:

- At the bottom of the Lighting window, click the **dropdown** next to Generate Lighting.
- Select **Clear Baked Data**.

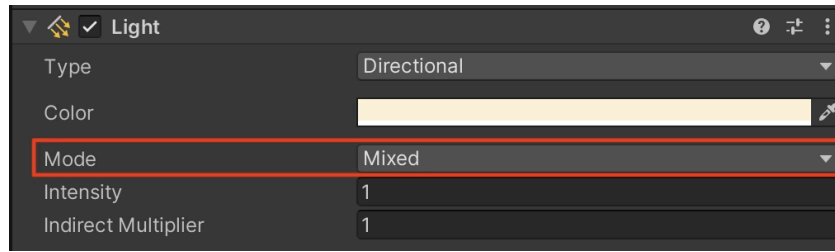
2. View the effect of as many lights as possible:

Warning: This is an extremely finicky and challenging art form. It may take hours of fiddling with values to get it looking the way you want.

	<ul style="list-style-type: none"> • Select the UniversalRenderPipelineAsset. • Temporarily increase the Additional Light Per Object Limit to the maximum. <p>3. Remove the default lights that came with the Room prefab:</p> <ul style="list-style-type: none"> • In the Hierarchy, expand the Room_[style] prefab. • Delete the two child Point Lights. • Note: you will have to open the room prefab in order to delete the lights. <p>4. Add more realistic lighting from actual light sources in your scene:</p> <ul style="list-style-type: none"> • From the Course Library > Prefabs > Lighting folder, make sure you have enough light-emitting objects to light your scene. • To add child Light objects to those lighting elements, right-click them in the Hierarchy and select Light > Point / Spot Light. <p>5. Get more even, balanced lighting in the room:</p> <ul style="list-style-type: none"> • On your Directional Light, try increasing the Intensity and Indirect Multiplier values. • In your Lighting settings, try increasing the Indirect Intensity value. <p>6. Test how your lighting looks:</p> <ul style="list-style-type: none"> • Click Generate Lighting again. • If you have time, increase the Lightmap Resolution and other Light settings values to see how it might look in its final state. • Note: if you are getting splotchy, pixelated shadows: from the Lighting settings, set the Lightmap Compression property to None. <p>You should now have rough lighting baked into your scene looking approximately the way you want.</p>  <p>Related resources:</p> <ul style="list-style-type: none"> • Configuring Lightmaps 	<p>Demo - If you forget to change even a single realtime light to baked, it could have a huge impact on performance (e.g. the fireplace).</p>
<p>Step 4 - Add mixed lighting</p>	<p>If you want your objects to cast shadows in real time, but also get the benefits of the performance from baked lightmaps, you can use "Mixed" lighting.</p> <p>1. Get real-time shadows on top of your baked lightmaps:</p> <ul style="list-style-type: none"> • Change the Mode of your Directional Light to "Mixed" (instead of "Baked"). <p>2. See the new mixed lighting take effect:</p> <ul style="list-style-type: none"> • In the Lighting window, click Generate Lighting to re-bake your lighting 	<p>Explain: When to use baked vs mixed vs realtime? If you're not going to interact with something - definitely bake it.</p> <p>Explain: If you want any of your dynamic objects to have shadows - and</p>

- Notice what effect the mixed light has on the lighting and shadows of your dynamic objects.

You should now have real-time shadows on your dynamic objects coming from your directional light.



Related resources:

- [Mixed Lighting](#)

have baked lights - need to set some lighting to Mixed.

Step 5 - Add light probes

In order to get more realistic real-time lighting from all of your baked lights in the scene, you need Light Probes.

1. Add a new Light Probe Group as a child of the Room object:

- In the Hierarchy, right-click the **Room_[style]** object and select **Light > Light Probe Group**.

2. To be able to edit the light probes:

- Make sure **Gizmos** with **3D Icons** are enabled from the top-right corner of the Scene view.
- Click the **Edit Light Probes** button in the Inspector of the Light Probe Group.

3. Duplicate and reposition the light probes around the room:

- Drag **selection** boxes around probes to select them.
- Use the **Move** tool to position the probes.
- Use **Ctrl/Cmd+D** to duplicate selected probes.
- **Note:** It may be helpful to switch to Isometric view by clicking on the small icon below the axis gizmo in the top-right corner of the Scene view.

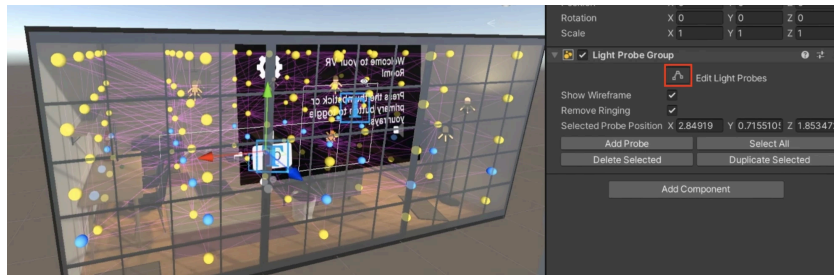
4. To position your light probes properly:

- Make sure you have probes in areas where lighting changes drastically either in color or intensity.
- Make sure your probes are not inside objects.

5. See the effect of your light probes:

- From the Lighting window, re-generate your lighting and test to see the lighting on your dynamic objects.

You should now have approximate dynamic lighting in your scene based on your baked lighting calculated by your light probes.



Related resources:

- [Light Probes](#)
- [Light Probe Groups](#)
- [Configuring Light Probes](#)

Explain: What are light probes? How do they work?

Demo: Show them before and after light probes - maybe duplicate an object where one has light probes Off and the other has it to "Blend Probes".

Demo: Recommend using isometric view for duplicating probes.

Warning: It's very easy to forget to click the "Edit Probes" button.

Warning: Don't position probes inside objects.

Step 6 - Finalize lighting

Now that your lighting is configured approximately the way you want, you are ready for a more final higher resolution lightmap.

1. Make sure your lighting is optimized appropriately for your target device:

- Test your app in the manner you expect it to be used by users (i.e. if you are targeting a mobile device, test it on the standalone device, not connected to your computer).
- Make sure you're hitting your target frame rate.
- For instructions on how to build and run your app on the actual device, refer to the instructions in the [VR Project Setup tutorial](#).

2. Generate a single lightmap file rather than several smaller ones:

- Increase the **Max Lightmap Size** to 4096.
- It is recommended to have a single lightmap file rather than several smaller ones.

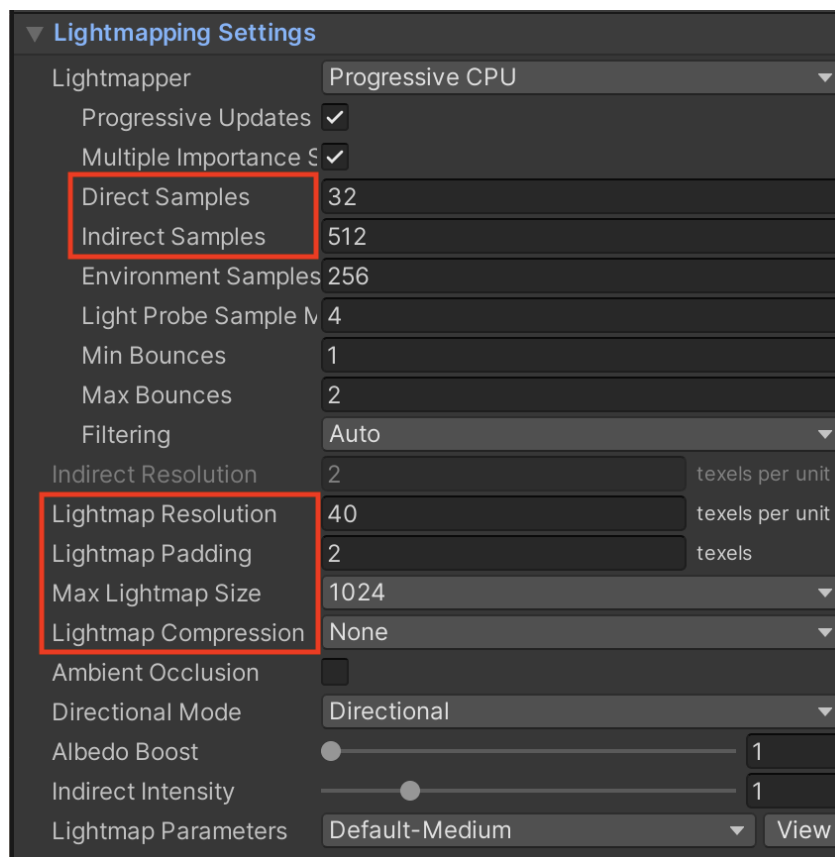
3. If you are getting splotchy or pixelated shadows:

- Set the **Lightmap Compression** property to **None**.

4. If you are happy with the general appearance of your lighting and are ready to generate your final lightmap:

- In the **Lighting** window, increase the Lightmap Resolution to somewhere between 20 and 80.
- If you want, continue to adjust other lighting settings (samples, bounces, etc) to get the desired result.

Your lighting should now look the way you want and be optimized for performance.





Related resources:

- [Setting up Lighting in Unity](#)

Recap

New Functionality:

	<ul style="list-style-type: none"> - Complex performant lighting <p>New Concepts and Skills:</p> <ul style="list-style-type: none"> - Baked vs Realtime lighting - Lightmapping - Light probes <p>Next Lesson:</p> <ul style="list-style-type: none"> - Publishing
Extensions	<p>If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.</p>
Extension - Add a new skybox	<p>1. Add a new skybox [Easy] Create or download a new skybox to enhance the look of your scene:</p> <ul style="list-style-type: none"> • Learn more about Skyboxes • You can download skyboxes on the Unity Asset Store - search for Free assets • Add your skybox from the Lighting window in the Environment tab • Warning: if you have a high resolution skybox, and have your skybox set as the Source of your Environment Lighting in the Environment tab, Lightmapping may take a very long time. You can change your Source to a Gradient to avoid this.  

**Extension -
Continue
tweaking
lighting**

2. Continue tweaking lighting [Medium]

Continue to fine-tune the lighting in your scene and the lighting settings to make it as beautiful and performant as possible:

- Try adjusting values in the [The Lighting Window](#) to see what kind of results you get.
- Learn more about lighting in Unity:
 - [Intro to Lighting and Rendering](#)



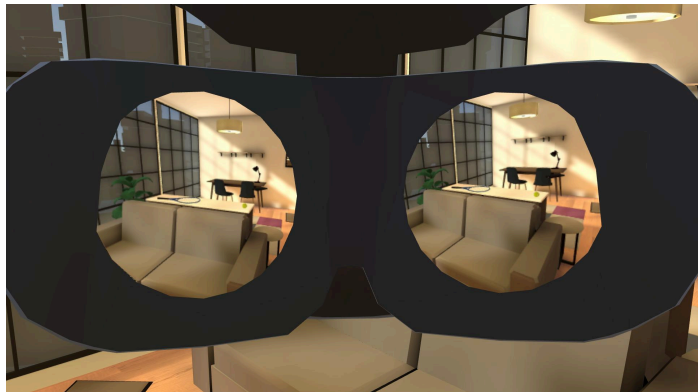
**Extension -
Reflection
Probes**


3. Explore Reflection Probes [Hard]

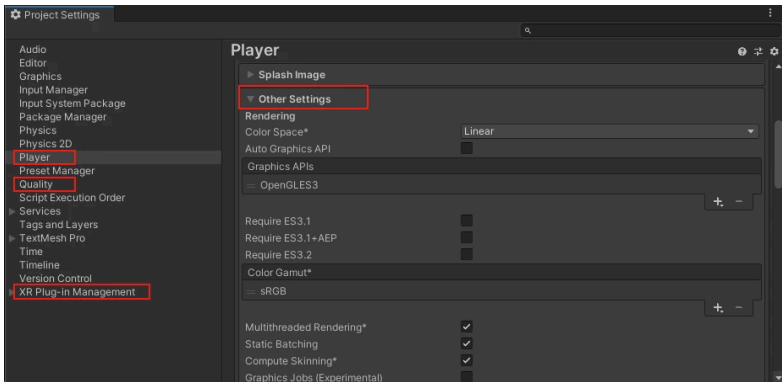
Add reflection probes into your scene in order to get more realistic reflections from metallic objects:

- [Reflection Probes Manual](#) and all about [using reflection probes](#)
- You should probably duplicate the Material_Main to create a new "Material_Main_Shiny", increase the material's metallic property, then assign that material to a shiny object in your scene that should be reflective.

Lesson 3.4 - Building and Sharing

Lesson Link	2.4 - User Interface - Unity Learn
Length	1 Hours 30 Mins
<p>Project Objectives: In this lesson, you will follow guidelines for best practices and explore options for sharing your work. By the end of this lesson, you'll configure your app appropriately and build a version of it that is ready to share.</p> <p>This lesson is part of the Create with VR course.</p> <p>Learning Objectives: By the end of this lesson you will be able to:</p> <ul style="list-style-type: none"> Adjust an app's settings in alignment with recommended best practices in order to publish it to a particular platform. Understand the landscape of VR app marketplaces in order to evaluate the appropriate distribution method for your app. 	

Step	Instructions	Teacher Notes
Step 1 - Enter the app's name and version number	<p>First, you want to check that the name, developer/company, and thumbnail aren't the default placeholders.</p> <ol style="list-style-type: none"> Access the Player Settings: <ul style="list-style-type: none"> Click Edit > Project Settings. On the left side, select the Player panel. Change the default app information: <ul style="list-style-type: none"> Change the Company Name, Product Name, and Version Number to reflect the current state of your app. If you want to edit the Splash Image that appears when the app starts up: <ul style="list-style-type: none"> Expand the Splash Image fold-out and edit its settings. You will need at least a Pro license to disable the Splash Image.  <p>Your App should now display on your device with its proper name, and should load with the splash image you want.</p>	<p>Explain: By "Player" settings - they don't mean the "Player" as in "User" - they mean the Device's app "player" (e.g. video player, audio player).</p> <p>Explain: Can just use a made up company name - or your name.</p> <p>Explain: Version numbers - Major#.Minor#.Revision# - maybe 1.0.1 is appropriate?</p> <p>Explain: No need to do icons here - that would be submitted separately through dev portal or other platform - won't show up on device in developer mode anyway.</p>

	<p>Related resources:</p> <ul style="list-style-type: none"> - Naming Best Practices - Icon Best Practices 	
<p>Step 2 - Confirm recommended publish settings</p>	<p>In addition to optimizing the performance of your app, generally, there are also a few platform-specific recommendations that will make your app build and perform at its best.</p> <p>Since specific recommendations are platform-specific and are changing all the time, the best way to find up-to-date and accurate information is searching online.</p> <p>1. Find guidance on recommended settings for VR publication:</p> <ul style="list-style-type: none"> • Explore the links under “Related Resources” below to find specific recommendations. • Google “Recommended Unity settings for [insert_device]”. <p>2. Configure Player Settings:</p> <ul style="list-style-type: none"> • Click Edit > Project Settings, then click Player from the left panel. • Scroll down to the Other Settings fold-out. <p>3. Configure Quality settings:</p> <ul style="list-style-type: none"> • Click Quality from the left panel of Project Settings. <p>4. Configure XR settings:</p> <ul style="list-style-type: none"> • Click XR Plug-in Management from the left panel of Project Settings. <p>5. Configure rendering settings:</p> <ul style="list-style-type: none"> • Locate and select the UniversalRenderPipelineAsset from the Project window. <p>6. Confirm shadow settings:</p> <ul style="list-style-type: none"> • For any Realtime or Mixed Lights, ensure you are using appropriate shadows. <p>7. To confirm Build Settings:</p> <ul style="list-style-type: none"> • Click File > Build Settings.  <p>Your app should now abide by all of the recommended settings for publishing to your target platform.</p> <p>Related resources:</p> <ul style="list-style-type: none"> - Configuration Settings - Blog: Unity Settings for Mobile VR - Stereo Rendering in VR / AR 	<p>Warning: There's lots of stuff we're going to gloss over - we'll have links if you want to learn more. Since technology and these settings are always changing, better to consult online articles to get the latest info for your specific platform.</p> <p>Warning: Might take a while to build after you edit these settings.</p>
<p>1. To configure Player Settings:</p>		

- Click Edit > Project settings, then click **Player** from the left panel.
 - Scroll down to the **Other Settings** fold-out.
 - In the **Rendering** section:
 - Set the **Color Space** to Linear.
 - Confirm the correct **Graphics API** (Open GL ES 3).
 - Confirm **Multithreaded Rendering** is enabled.
 - In the **Identification** section:
 - Confirm the correct **Minimum API Level** (Marshmallow API Level 23). This is only relevant for Android builds.
 - In the **Configuration** section:
 - Set the **Scripting Backend** to IL2CPP.
 - In the **Optimization** section:
 - Enable the **Prebake Collision Meshes** setting.
 - Enable the **Keep Loaded Shaders Alive** setting.
2. In **Quality** settings, in the **Rendering** section:
 - Confirm **Texture Quality** is set to "Full Res"
 - Confirm Anisotropic Textures are set to "Per Texture"
 3. In **XR Plug-in Management** settings:
 - If building to Android, in the **Android** tab, set **Stereo Rendering Mode** to Multiview and enable the **Low Overhead Mode** setting .
 - If building to desktop, in the **Desktop** tab, set **Stereo Rendering Mode** to Single Pass Instanced.
 4. In the **UniversalRenderPipelineAsset**:
 - Confirm **HDR** is disabled.
 - Confirm **Anti Aliasing** is set to 4x.
 - Confirm **Additional Light** is set to "Per Pixel" and the **Per Object Limit** is set to 1.
 5. For any **Realtime or Mixed Lights**:
 - Ensure you are using either **Hard Shadows** or **No Shadows**.
 6. In Build Settings:
 - set the **Texture Compression** to "ASTC".

**Step 3 -
(Optional) Look
into sharing or
publishing your
app**

Up to this point, you have only been building and running the app through Unity. Take a look at what it would take to distribute your app outside of Unity.

1. Explore the requirements for submitting your app to an official store:

- Use the links below to read more about the process.
- Google "Submit app to [insert_store_name]" (e.g. Steam, Oculus Store).

2. Explore alternative platforms for distributing your App:

- Explore requirements for submitting your app to sites like [SideQuest](#) and [Itch.io](#).

You should now have a sense of the various platforms available for distributing your app and an understanding of what it might take to have an app approved for one of those platforms.

Related resources:

- [Submit your app to the Oculus Store](#)
- [Lifecycle of an Oculus App](#)
- [Oculus Best Practices](#)
- [Oculus VRCs](#)

Explain: How can you load your apk if your app isn't on the oculus store? You might have to "side load" it.

Recap

New Functionality:


- App & Company name
- Recommended settings

New Concepts and Skills:

- Best practices per device
- Distribution options

	Next Lesson: <ul style="list-style-type: none"> - What's next
Extensions	<p>If you want to further develop your skills, explore new concepts, and improve your project, check out some of the optional extension activities below. Each one is tagged as [Easy], [Medium], [Difficult], or [Expert] and will also include a [Requires Programming] tag if some coding is required.</p>
Extension - Publication process	<p>1. Research the publication process [Easy]</p> <p>Research the requirements for submitting an app to your preferred platform:</p> <ul style="list-style-type: none"> • This could include an official app marketplace like the Oculus Store or Steam. • You could also submit your app to an unofficial platform like SideQuest or Itch.io.
Extension - App artwork	<p>2. Create some artwork for your app [Medium]</p> <p>Take epic screenshots, make an app icon, hero image, or other materials to showcase your app:</p> <ul style="list-style-type: none"> • You can use the Unity Recorder to capture high quality images of your app. • Follow guidelines from your relevant app platform on how to make the best icon or imagery <ul style="list-style-type: none"> ◦ E.g. Oculus Blog on How to Prep App for Submission ◦ E.g. Sidequest requirements on image sizes for banner images and icons

Challenge 3 - Training Simulation App

Lesson Link	Challenge 3 - Training Simulation - Unity Learn	
Length	1 Hour	
<p>In this challenge, you'll apply the skills you learned while making your VR Room in an industrial training simulation. In this app, using buttons, knobs, levers, and joysticks, the user has to collect crates dropped into the factory and stack them on a nearby platform.</p> <p>This challenge is part of the Create with VR course.</p> <p>This challenge will assess skills learned in the following lessons:</p> <ul style="list-style-type: none">• Comfort & Accessibility• Optimization• Lighting		
		

Getting started	<ol style="list-style-type: none"> 1. Open the broken Training prototype Scene: <ul style="list-style-type: none"> • From the Project window, expand, Assets > Challenges > 03_Training > Scenes • Double-click on the Training_Prototype_Broken Scene to open it. 2. Begin working on the challenge tasks: <ul style="list-style-type: none"> • Work through the tasks outlined in the steps below. • If you want to push your skills, attempt the optional bonus challenge tasks, as well. • If you get stuck, there are hints for each task at the bottom of the page.
-----------------	--

Challenge	Task	Hint
1. The black fade in screen appears in the middle of the factory.	<ul style="list-style-type: none"> • The black fade in screen should cover the user's entire field of view. 	1. To better visualize the fade canvas, temporarily set its Canvas Group Alpha value to 1, then make sure it's not too far from the camera.
2. The "Console Labels" toggle makes the entire console disappear.	<ul style="list-style-type: none"> • Only the labels on the console should disappear when the toggle is deactivated. 	2. Make sure the correct object is being Activated/Deactivated by the toggle.
3. The lighting on the floor looks really bad.	<ul style="list-style-type: none"> • Make the floor lighting look as detailed as the other surfaces. 	3. From the top of Scene view, change the Draw Mode from Shaded to Baked Lightmap - you will see that

		the lightmap resolution on the floor is very low compared to other objects.
4. The pile of boxes in the back left corner does not have realistic baked lighting and shadows.	<ul style="list-style-type: none"> The shadows on all the piles of boxes should be the same. 	4. In order for objects to be baked into the lightmap, they need to be marked as "static".
5. The console is not responding to the baked light above it - it looks very dark.	<ul style="list-style-type: none"> Keeping the light above the console baked for optimization, the console should appear to be lit by the light above it.. 	5. Light probes allow dynamic objects to respond to baked lighting conditions.
Bonus		
6. There is no visual indicator on the magnet when it is active vs inactive.	<ul style="list-style-type: none"> The magnet should turn a different color when it is active vs inactive. 	6. You can use the same function you used to change the indicator light on the remote.
7. The user has access to their ray and direct interactors throughout the entire app.	<ul style="list-style-type: none"> The user should not be able to use the console while reading the instructions and their rays should disappear when they have clicked through the instructions. 	7. In the ShowMessageFromList component on the welcome Text object, there is an On Complete () event. On this event, you should toggle off the rays and toggle on the console's functionality, which should be disabled by default.
Expert Bonus		
8. There is no way for the user to be able to control the speed of the magnet. [Expert]	<ul style="list-style-type: none"> There should be a setting that controls the speed of the magnet. 	8. No hints available. Use Google!
9. There is no way for the user to track or save their progress in this training. [Expert]	<ul style="list-style-type: none"> There should be some user interface that displays how many boxes have been successfully lifted and how much time has elapsed. 	9. No hints available. Use Google!

Lab 3 - Personal Project Optimization & Lighting


Lesson Link	Lab 3 - Personal Project Ergonomics & Optimization - Unity Learn
Length	2 Hours
<p>By the end of this lab, your personal project will be filled with beautiful art, optimized for performance, and ready to share.</p> <p>This lab will draw on skills learned in the following lessons:</p> <ul style="list-style-type: none"> • Comfort & Accessibility • Optimization • Lighting • Building & Sharing <p>This lab is part of the Create with VR course.</p>	
	

Step	Instructions	Teacher Notes
Step 1 - Fill in Design Document for ergonomics & optimization	<p>Before you resume work on your project, you need a plan for what you're actually going to work on. You will continue filling out your design document so you have a clear plan of action for this lab.</p> <ol style="list-style-type: none"> Pick up where you left off with your design document: <ul style="list-style-type: none"> Re-open the document you created during Lab 1 and continued working on during Lab 2. Fill out Section 5 (Optimization & Publishing) of the document: <ul style="list-style-type: none"> List the ways you intend to make the user experience more comfortable or accessible. Write down the target performance metrics for your target device(s). Describe the lighting strategy you will implement. (Optional) Add more specific details to your design document: <ul style="list-style-type: none"> Continue to fill out section 6 (Other features) Continue to fill out section 7 (Sketch) Continue to fill out section 8 (Timeline) <p>You should now have section 4 of your Design Document complete, providing direction for this lab where you will be implementing your app's core functionality.</p> <p>Related resources:</p> <ul style="list-style-type: none"> - Oculus Best Practices 	
Step 2 - Replace primitive objects with 3D art	<p>Before we begin optimizing our app and improving the user experience, it's time to replace some of our primitive assets with some real 3D art.</p> <ol style="list-style-type: none"> Browse assets on the asset store: <ul style="list-style-type: none"> Go to assetstore.unity.com. Sign in to the Unity Asset Store. Search for "low poly" assets. <p>Tip: You can also filter by "Free".</p> Import an asset pack you like: <ul style="list-style-type: none"> Click Add to My Assets > Open in Unity. From the Package Manager, click Download. Then click Import to add the assets to your project. To create your own 3D models inside Unity using Probuilder: 	<p>Explain: Why use Low Poly?</p> <p>Warning: This is a very involved step - you could spend an entire day on this!</p> <p>Explain: You don't have to replace every asset right now - just get started.</p>

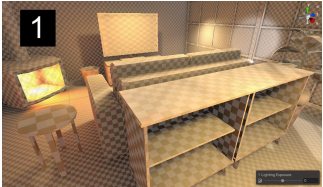
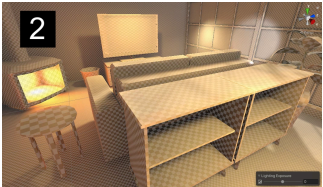
	<ul style="list-style-type: none"> • From the Package Manager, install the Probuilder package. • From the top menu in Unity, click Tools > Probuilder > Probuilder Window. <p>4. Replace the primitive shapes in your scene with custom or imported 3D art:</p> <ul style="list-style-type: none"> • Add the 3D models as child objects of empty parent objects in your scene. • To achieve appropriate scale, adjust the scale of the child object rather than the scale of the parent object. • As necessary, add colliders to your objects to make them collide appropriately. • As necessary, adjust the attach points of your objects so that they are grabbed appropriately. • If possible, try to use the same shaders to reduce draw calls. <p>Your scene should now look a lot nicer with some real 3D art.</p> <p>Related resources:</p> <ul style="list-style-type: none"> - About Probuilder / Probuilder Docs 	
Step 3 - Improve comfort and accessibility	<p>You should take time to consider how comfortable and accessible your app will be. This will allow it to be enjoyed by as many people as possible and make sure fewer people are excluded from using your app due to design decisions you've made.</p> <p>1. Make sure your app's locomotion is as comfortable as possible.</p> <ul style="list-style-type: none"> • Follow the instructions in the Comfort and Accessibility tutorial. <p>2. Add settings and menu options that increase the accessibility of your app:</p> <ul style="list-style-type: none"> • Follow the instructions in the Comfort and Accessibility tutorial. <p>Your app should now be more comfortable and accessible, allowing it to be enjoyed by more people than it was before.</p>	
Step 4 - Optimize draw calls and polycount	<p>Now that the app is largely functional and has some real 3d assets, you can attempt to optimize your app's performance to hit your target metrics.</p> <p>1. Determine your app is currently performing, including fps, draw calls (batches), and polycount (tris):</p> <ul style="list-style-type: none"> • Follow the instructions in the Optimization tutorial. <p>2. Reduce polycount (tris):</p> <ul style="list-style-type: none"> • Follow the instructions in the Optimization tutorial, including replacing high-poly assets with lower-poly ones. <p>3. Reduce draw calls (batches):</p> <ul style="list-style-type: none"> • Follow the instructions in the Optimization tutorial, including marking objects as static and using shared materials. <p>4. Improve jagged lines with anti-aliasing:</p> <ul style="list-style-type: none"> • Follow the instructions in the Optimization tutorial. <p>5. Quickly experiment with changes to your app's performance:</p> <ul style="list-style-type: none"> • Locate the UniversalRenderPipelineAsset and tweak its settings to see what has an impact. <p>You should now have an understanding of how your app is performing and, if it is performing poorly, what the cause of that poor performance might be. You should not be discouraged if your frames per second is still too high, though, since you have not yet optimized lighting.</p>	

	Related resources: <ul style="list-style-type: none"> • Seven stages of optimizing mobile VR content in Unity • Optimizing your VR/AR Experiences • Unit 9 - Optimizing VR Apps • VR Best Practice • Optimizing graphics performance 	
Step 5 - Optimize lighting	<p>In VR, performance is critical to the user experience and lighting is one of the biggest components of performance. In order to optimize performance for VR, baking most of the lighting is highly recommended, using a minimal number of real-time or mixed lights.</p> <p>1. Add and bake environmental lighting in your scene:</p> <ul style="list-style-type: none"> • Follow the instructions in the Lighting tutorial. <p>2. Improve realtime lighting with mixed lights and light probes:</p> <ul style="list-style-type: none"> • Follow the instructions in the Lighting tutorial. <p>You should now have baked lighting in your scene with realtime effects on dynamic objects from mixed lights and light probes.</p>	/ - This is not your final lighting - just a first broad attempt
Step 6 - Confirm settings and build your app	<p>The final step before building a version of an app that you would consider sharing or publishing is double-checking that you are using all of the recommended build settings for your target platform.</p> <p>1. Confirm that your app's project settings are following best practices for VR:</p> <ul style="list-style-type: none"> • Follow the instructions in the Building and Sharing tutorial, including Player settings, Quality settings, Build settings, Render Pipeline Asset properties, Lighting and shadow settings <p>2. Build a version of your app that can be shared or published:</p> <ul style="list-style-type: none"> • Follow the instructions in the Building and Sharing tutorial. <p>You should now have a built app, ready for distribution, following all best practices for build settings.</p>	•
Recap	<p>New Functionality:</p> <ul style="list-style-type: none"> • New art • Improved comfort and accessibility • Optimized key metrics • Improved, baked lighting • Built app <p>New Concepts & Skills:</p> <ul style="list-style-type: none"> • Optimizing and publishing your own VR app. <p>Next Lab:</p> <ul style="list-style-type: none"> • Next Steps and the rest of your VR development life! 	

Quiz 3

Lesson Link	Quiz 3 - VR Ergonomics & Optimization - Unity Learn	
Length	15 Mins	
<p>In this quiz, you will test the knowledge and skills you learned in Unit 3 related to VR Lighting and Optimization.</p> <p>This quiz is part of the Create with VR course.</p> <p>This quiz will assess skills learned in the following lessons:</p> <ul style="list-style-type: none"> • Comfort & Accessibility • Optimization • Lighting • Building & Sharing 		

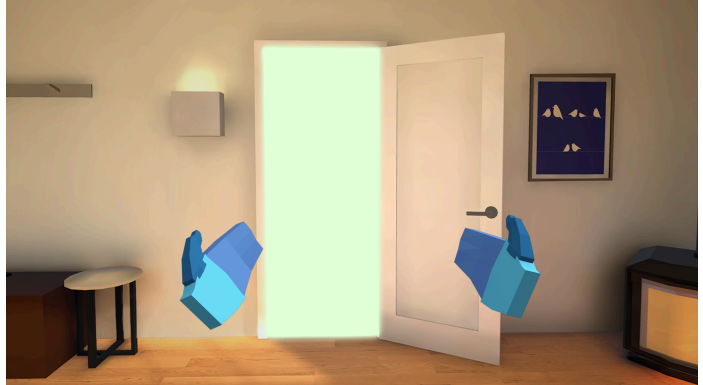
#	Question	Answers	Explanation
1	Which of the following would make your app LESS comfortable for users?	<p>A. During continuous motion, apply a black vignette around the edge of the screen to reduce the field of view.</p> <p>B. During teleportation, fade to black to reduce the sense ofvection.</p> <p>C. Slowly fade in at the start of the scene so that the user does not suddenly find themselves in a new room.</p> <p>D. Decrease the frames per second (FPS) so that the image does not change as quickly.</p>	Decreasing the FPS will make your app much less comfortable. If the user's brain is not "tricked" into thinking they are looking at a real visual field, they are likely to get simulator sickness.
2	True or False: Providing users options to choose between several types of locomotion increases accessibility.	<p>A. True</p> <p>B. False</p>	It is hard to anticipate how people will use your app. Providing more options for how users can engage with your app increases the likelihood that it will be accessible to more people.
3	Which one of the following tends to improve performance?	<p>A. Post-processing</p> <p>B. Real-time lighting</p> <p>C. Static batching</p> <p>D. Fog</p>	Static batching combines static (not moving) GameObjects into big Meshes, and renders them in a faster way.
4	Which one of the following Unity tools is NOT used to diagnose performance problems.	<p>A. The Rendering statistics window</p> <p>B. The Profiler</p> <p>C. The Frame Debugger</p> <p>D. The Timeline</p>	Timeline is used to create cinematic content, game-play sequences, audio sequences, and complex particle effects. All the other tools can be used for

			diagnosing performance problems.
5	<p>Which of the following techniques would likely DECREASE draw calls (batches)?</p>	<p>A. Sharing one material over many objects.</p> <p>B. Using anti-aliasing.</p> <p>C. Using anisotropic filtering.</p> <p>D. Increasing your number of triangles.</p>	<p>One of the best ways to reduce draw calls is to use a shared texture atlas for many objects and to have those objects also share the same material and shader. Unity cannot batch objects if they are using different shaders.</p>
6	<p>Which of the following are true about lighting modes?</p> <p>Select all correct answers.</p>	<p>A. Objects must be set to static for lightmapping.</p> <p>B. Lights must be set to Realtime mode for lightmapping.</p> <p>C. Mixed Lighting allows for real-time shadows.</p> <p>D. To see the effect of Light Probes, you have to bake your lighting.</p>	<p>Lightmapping is only possible with static objects. Realtime lights cannot be baked. You can use Mixed Lights to combine dynamic shadows with baked lighting from the same light source. Similar to lightmaps, light probes store "baked" information about lighting in your scene. The difference is that while lightmaps store lighting information about light hitting the surfaces in your scene, light probes store information about light passing through empty space in your scene.</p>
7	<p>Looking at the two images below, what can you tell about their lighting settings?</p>  <p>See high-res image here</p>  <p>See high-res image here</p>	<p>A. Image 1 has a lower Lightmap Resolution than Image 2.</p> <p>B. Image 1 will take longer to bake than Image 2.</p> <p>C. Image 1 has a lower number of Direct Samples than Image 2.</p> <p>D. Image 1 has a lower number of Bounces than Image 2.</p>	<p>Increasing the Lightmap Resolution will produce a more detailed lightmap, which will take longer to bake. From this image alone, you cannot tell anything about the Direct Samples or Bounces.</p>
8	<p>Which of the following are true about graphic quality optimization in Unity?</p> <p>Select all correct answers.</p>	<p>A. Mip maps should be used for high-resolution textures that might be seen from far away.</p> <p>B. Anisotropic filtering should be used to improve the quality of textures seen at an angle.</p> <p>C. Texture Atlases help make images clearer when they are moving at high speeds.</p> <p>D. Using lower resolution images can</p>	<p>Mip maps are smaller versions of the texture that get used when the texture is very small on screen. Anisotropic filtering makes textures look better when viewed at a shallow angle, but comes at a performance cost in the graphics hardware. A Texture atlas is a larger texture containing many smaller textures or color swatches</p>

		decrease the number of triangles in your scene.	- this does not have anything to do with fast-moving objects. Changing the resolution of your images will not affect your polycount number of triangles).
9	What does Anti-aliasing do?	<p>A. It optimizes your textures to improve frame rate.</p> <p>B. It simplifies far-away objects to reduce the number of triangles.</p> <p>C. It optimizes draw calls by batching together objects that share the same texture.</p> <p>D. It smooths out jagged edges on curved and diagonal lines.</p>	Anti-aliasing reduces the prominence of these jagged lines by surrounding them with intermediate shades of color. Although this reduces the jagged appearance of the lines, it also makes them blurrier.
10	Which of the following would actually DECREASE the accessibility of your VR app?	<p>A. For mobility / motor accessibility, make sure all actions require two hands.</p> <p>B. For hearing / audio accessibility, add closed captions to any critical audio instructions.</p> <p>C. For vision accessibility, make text instructions large and clear.</p> <p>D. For cognitive accessibility, include directions or tutorials for how to use your app.</p>	Requiring two-handed interactions would actually decrease motor accessibility. Ideally, actions could be completed with either or both hands.

Other tutorials

VR Next Steps

Lesson Link	VR Next Steps - Unity Learn	
Length	45 Mins	
<p>In this lesson, you will be provided with potential next steps, now that you have completed this course. Whether you are more interested in art or programming, there is so much more you can learn in order to take your skills to the next level and create even more custom, unique VR experiences.</p> <p>If you have completed either the Unity Certified User: Programmer exam or the Unity Certified User: Artist exam, you are also eligible to take the Unity Certified User: VR Developer exam.</p> <p>This lesson is part of the Create with VR course.</p>		

Step	Instructions	Teacher Notes
Continue working on your Personal Projects	<p>Maybe the most effective way to gain new VR development skills is actually building something new in VR. Luckily, you already have a personal project in development, which is the ideal opportunity for you to expand your skills.</p> <p>Dream up some new ambitious features for your personal project and try to implement them. As you do this, not only will you improve the project that you can share and add to your portfolio, you will also very quickly take your VR skills to the next level.</p> <p>As you continue developing your project, please share it with the Unity community! We'd love to see what you're working on.</p>	
Learn more about Programming for VR	<p>In this course, you used the default behaviors included in the XR Interaction Toolkit and some pre-built scripts that were provided to you in the course. However, if you want to continue developing unique interactions in VR, especially if you are interested in doing VR development professionally, you should be comfortable programming and debugging custom VR interactions.</p> <p>1. Solidify your programming fundamentals: If you are not already confident with your programming, you may find programming for VR somewhat challenging since it is a rapidly evolving technology with more limited documentation. For that reason, it is recommended that you start with the Junior Programmer Pathway, which will give you the foundational skills needed to jump-start your VR development.</p> <p>2. Implement some VR-specific functionality: Once you are already comfortable with programming, you should attempt to follow some VR development tutorials to apply your programming skills on custom VR interactions. We highly</p>	

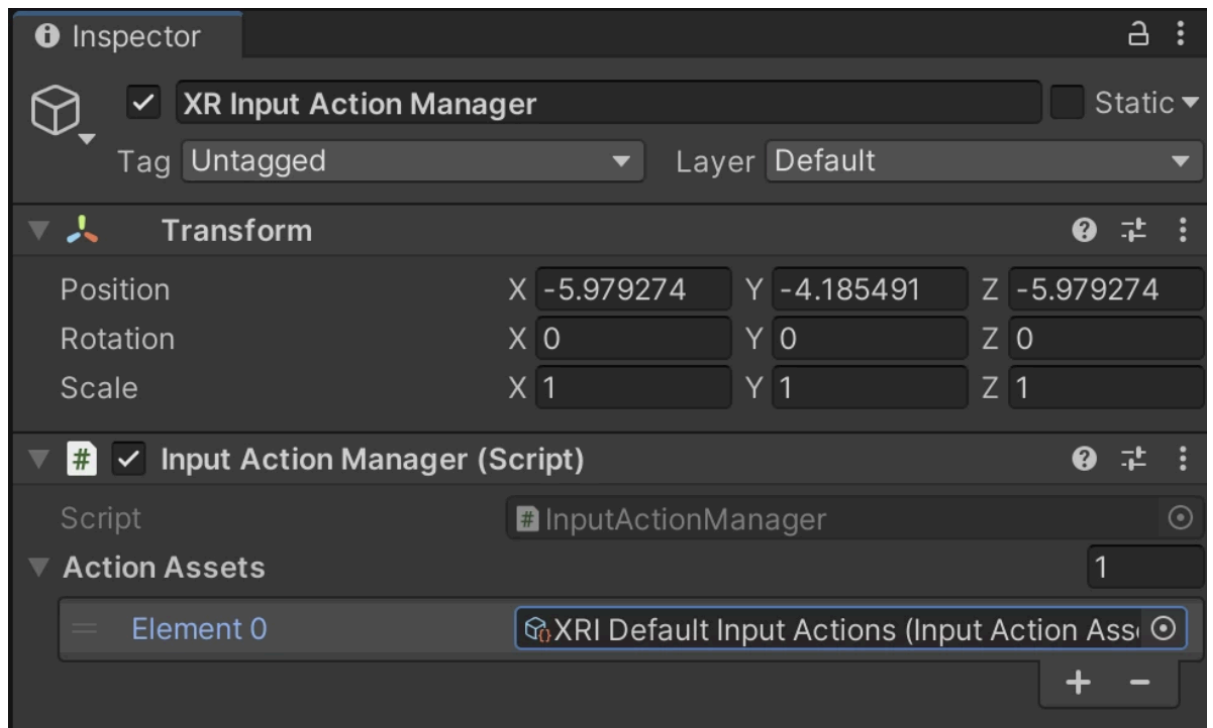
	<p>recommend the VR with Andrew YouTube channel.</p> <p>3. Get comfortable with the Profiler: Since performance is so critical for VR, it is important for any VR developer to be comfortable using the Profiler window. This is a very complex tool, which allows you to pinpoint things like how your code, assets, scene settings, camera rendering, and build settings affect your application's performance.</p> <p>4. Complete the VR Development Pathway This learning pathway is designed for anyone interested in learning to create experiences for VR using their knowledge of programming. By the end of this pathway, you will be ready to get a job in VR development.</p> <p>If you are interested in getting an entry level job doing development for VR, what's most important is a portfolio showcasing your experience developing custom VR interactions. As you expand your skills, develop small prototypes that you can share with potential employers or institutions.</p>
Learn more about Art for VR	<p>In this course, all of the art assets and environments were pre-built for you, optimized for performance on mobile VR headsets. If you want to create environments in VR that are completely your own or to get a job creating art for a VR project, there is a lot more you can learn to build your skills.</p> <p>1. Understand the Universal Render Pipeline: The Universal Render Pipeline (URP) provides artist-friendly workflows that let you quickly and easily create optimized graphics across a range of platforms, from mobile to high-end consoles and PCs. For this reason, it is incredibly useful for VR development, where you will be developing for devices with a wide range of processing power.</p> <p>2. Develop Low Poly assets: Since performance is so critical in VR, it is important that you are comfortable developing nice, low poly assets that will perform well on mobile VR headsets.</p> <p>3. Create custom materials and shaders: You did briefly explore materials and shaders in this course, but if you are going to develop your own custom VR art, it's important that you can develop your own materials and shaders that are optimized for VR.</p> <p>4. Learn more about lighting: You learned some of the basics for optimized lighting in this course, but that was only the surface of a very deep and complex topic. It is recommended that you explore lighting more thoroughly to be able to achieve desired lighting in the most performant way possible.</p> <p>These are just some of the most important skills for art in VR. As you actually attempt to implement your art into VR environments, you will learn additional critical skills in the VR art pipeline.</p>
Take the VR Developer certification exam	<p>If you have already earned the Unity Certified User: Programmer or the Unity Certified User: Artist certification, you are eligible to take the The Unity Certified User VR Developer certification exam. You can find the objectives for the exams here.</p> <p>This certification can be used to prove to potential employers or educational institutions that you are capable of developing VR experiences. It tests your ability to create VR experiences in Unity and assumes you have accumulated about 200 hours of experience with Unity.</p> <p>If you completed the entirety of this course, you should be able to pass this exam. However, if you want to ensure you are ready for it, there is additional courseware you can purchase to help with your preparation.</p>

Consider publishing your work	<p>Research the requirements for submitting an app to your preferred platform: This could include an official app marketplace like the Quest Store or Steam. You could also submit your app to an unofficial platform like SideQuest or Itch.io.</p>
Submission: Show us what you're working on	<p>If you have continued to develop your personal project, developed new prototypes, or created custom art for VR, please share it with us here.</p> <p>Take screenshots, videos, or post links of what you're working on!</p>

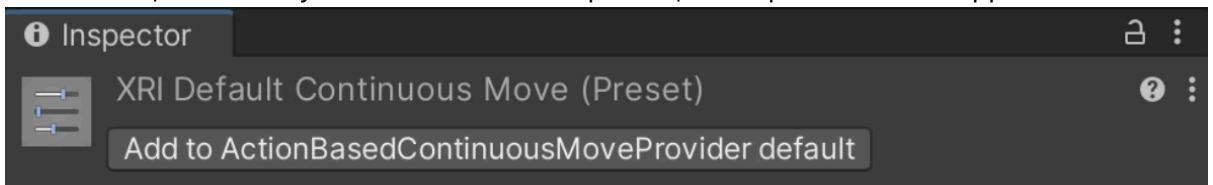
How to set up a VR Project from Scratch

Step 1 - Install the correct version of Unity	<p>Before you make a project, make sure you have the correct version of Unity installed, which can support stable VR development:</p> <ol style="list-style-type: none">1. From Unity Hub, install Unity 2021 LTS (long-term support).2. If developing for mobile VR (e.g. Oculus Quest), make sure to download the Android export module. <p>You should now have the correct Long-Term Support version of Unity installed with the appropriate export modules.</p>
Step 2 - Create a new URP Project	<p>Now that we have the correct version of Unity installed, we need to make a new project on that version configured for VR.</p> <ol style="list-style-type: none">1. Open the Projects tab in Unity Hub2. Select New Project3. Make sure the Editor Version of your new project is set to the recommended version from the previous step.3. From the list of templates, select 3D (URP), then set the Project Name, and the Location to save it.4. Click Create and wait for your project to open. <p>You should now have a new empty URP project open.</p>
Step 3 - Create a new empty scene.	<p>Before we do anything related to VR, let's set ourselves up with a simple empty scene.</p> <ol style="list-style-type: none">1. Make a new Scene:<ul style="list-style-type: none">• In the Project window, right-click on the Scenes folder and select Create > Scene.• Rename the new scene "VR Starter Scene".• Open the new scene2. Add a simple ground object:<ul style="list-style-type: none">• In the Hierarchy, right-click to create a 3D > Plane object.• Rename the Plane object "Ground."• In the Inspector, make sure its XYZ coordinates are set to X=0, Y=0, Z=0. <p>You should now have a new empty scene open with a simple ground object.</p>
Step 4 - Install XR plugin packages.	<p>Before we can begin developing for a particular device, we need to make sure we can interface with different devices.</p> <ol style="list-style-type: none">1. Enable XR plug-ins:<ul style="list-style-type: none">• From the top menu select Edit > Project Settings.• In the XR Plugin Management panel, select the Install XR Plugin Management button.2. Install the OpenXR package:<ul style="list-style-type: none">• From within the XR Plugin Management panel, in the PC, Mac, Linux & Standalone tab, select OpenXR from the list of available Plug-in Providers to install the plug-in.• This will allow you to publish to a wide range of devices using a single plugin.• If you are prompted to restart your project in order to switch to the new input system, select Yes.3. Resolve warnings by setting up an interaction profile.<ul style="list-style-type: none">• After adding the OpenXR plugin, notice the warning or error icon that appears next to the plugin Name• Click on the warning or error icon to open the OpenXR Project Validation window, which will tell you that you need to add an "interaction profile" for the device you're using, select the Edit button to do that. This will open the OpenXR settings panel.• In the Windows, Mac, Linux tab, make sure the Oculus Touch Controller Profile appears in the list of Interaction Profiles, then enable all available OpenXR Feature Groups.• If you are using a different device, such as the Valve Index or HTC Vive, select that interaction profile instead.• There should no longer be any warnings in the XR Plugin Management panel. If there are, select them and follow the recommended steps to resolve them.

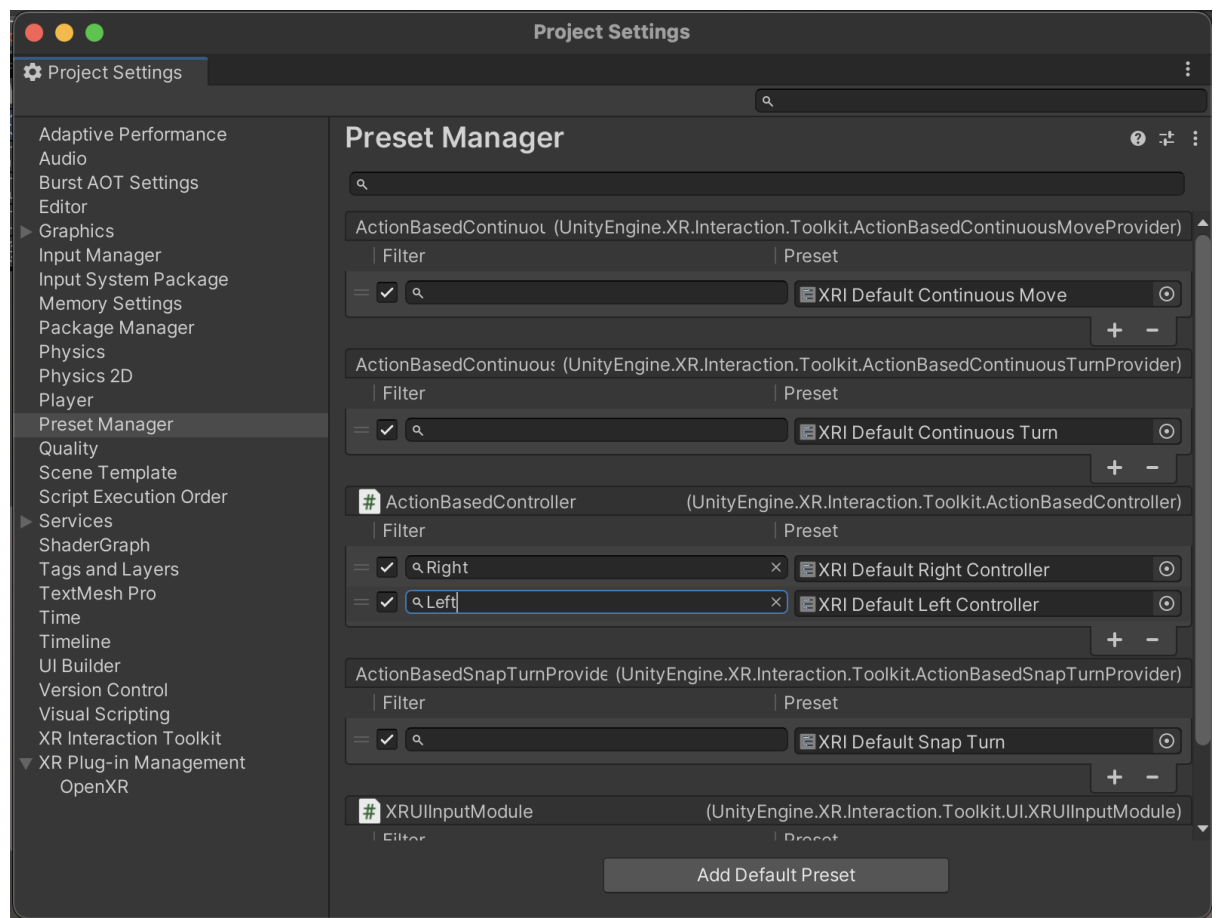
	<ul style="list-style-type: none"> If you are having trouble resolving warnings or errors, or if you need help troubleshooting, check out this detailed documentation on configuring the OpenXR plugin. <p>You should now have the appropriate plugins installed for your target device.</p>
Step 5 - Install the XR Interaction Toolkit and samples	<p>It would technically be possible to deploy to a device now if we already had a functioning project, but in order to actually develop the project, we need to install the XR Interaction Toolkit XRIT.</p> <ol style="list-style-type: none"> Allow the installation of "Pre-release" packages: <ul style="list-style-type: none"> In the Project Settings window, in the Package Manager panel, select the option to Enable Pre-release Packages and then select "I Understand". Browse preview packages in the Package Manager: <ul style="list-style-type: none"> From the top menu, select Window > Package Manager. To be able to search through all packages, from the drop-down in the top-left, change the scope from "In Project" to "Unity Registry". Install the XR Interaction Toolkit package: <ul style="list-style-type: none"> Search for and select the XR Interaction Toolkit package. If you don't see it in the list, you can select the + button and choose to Add Project by Name, then enter "com.unity.xr.interaction.toolkit". Click Install. This will allow you to quickly develop XR interactions. Install helpful samples for quick development: <ul style="list-style-type: none"> After the XR Interaction Toolkit package has installed, a new Samples fold-out should appear at the bottom of the description. Click to Import the Starter Assets. <ul style="list-style-type: none"> This provides a set of Default Input Action presets for mapping specific buttons to actions using the action-based input system Click to Import the XR Device Simulator <ul style="list-style-type: none"> This allows you to test your VR app using keyboard and mouse controls rather than an actual headset. <p>You should now have the XR Interaction Toolkit package installed, including helpful samples, and have a new XR option when creating new objects in the Hierarchy.</p>
Step 6 - Add an Action-based XR Rig to the scene.	<p>Now that you have all of the packages and samples you need to get started, you can set up your scene for XR Development. If you require additional information after trying this step, check out the Samples Documentation.</p> <ol style="list-style-type: none"> Set up the action-based input system in your scene: <ul style="list-style-type: none"> In the Hierarchy, right-click to create a Empty GameObject, then rename it "XR Input Action Manager" Add an Input Action Manager component to the object. Expand the Action Assets fold-out and click the + to add a new asset. Select the object picker for Element 0 and assign the XRI Default Input Actions preset.



2. Apply action presets automatically when you add XR components:
 - In the Project window, navigate to **Samples > XR Interaction Toolkit > [version] > Starter Assets**.
 - Select each default action preset in this folder and, at the top of the Inspector for each one, select **Add to [component] Default**.
 - Now, whenever you add a new XR component, these presets will be applied.



3. Make sure that new XR GameObjects use the appropriate presets:
 - From **Project Settings**, open the **Preset Manager** panel, then locate the **ActionBasedController** section
 - For the XRI Default Right Controller, type the Filter value **"Right"**
 - For the XRI Default Left Controller, type the Filter value **"Left"**



4. Add an action-based XR Rig to your scene:

- In the Hierarchy, right-click to create an **XR > XR Origin (VR)**.
 - This object contains the camera and controllers for the user.
- In the Hierarchy, expand **XR Origin > Camera Offset**, then select one of the controller objects.
 - Notice how the Action references are already assigned by default.

You should now have an XR Origin in your scene set up with action-based inputs already configured.

Step 7 - Configure other project settings.

Before running your project, you should also make sure the project's quality and rendering settings are appropriate for VR.

1. If you are building to Android (for a Quest):

- Still in the **Project Settings** window, in the **Player** settings panel, in the **Android** tab, in the **Other settings** fold-out, locate the **Minimum API Level**.
- Set the Minimum API Level to Android 10 (API level 29) (or whatever is recommended in [this Quest Documentation](#)).

2. Make sure your graphics quality settings are appropriate for VR.

- Still in Project Settings, click the **Graphics** panel on the left.
- For the **Scriptable Render Pipeline Settings** property, use the object picker to select the **URP-Balanced** option.
 - This will ensure your graphics are not too intensive for mobile VR.

3. Remove post-processing from your scene:

- In the Hierarchy, select the **Global Volume** GameObject and set it as Inactive.
- This object adds post-processing to your scene, which you will learn about in later tutorials - to start and to keep things as performant as possible, it's best to remove post-processing for now.

You should now have critical settings configured properly for running VR projects.

Step 8 - Import the

With your project now completely set up for VR, you can import the course-specific assets and scenes.

course assets	<ol style="list-style-type: none">1. Click to download the zip file that contains the assets.2. Extract the zip file and open the folder inside to locate the .unitypackage within.3. Import the .unitypackage into your project. <p>You should now see _Course Library and Challenges folders in your assets</p>
Next steps	<p>You now have the project set up for VR development.</p> <p>Next, you can continue on to Lesson 1.1 - VR Project Setup Step 2 in order to add the appropriate device-specific plug-ins and build your project for the first time.</p>