HW1: The Room (100pts + 10 EC pts)

By Nathaniel Myren, Shantanu Tulshibagwale, Elmeri Uotila, Paula Alavesa and Alessandro Nardi

Updated 10th of Jan 2024

Instructions

If you are working in the TS 135 lab, then all of the equipment will be provided to you. If you work from home or some other place, you need to make sure you have Unity3D Engine installed. If you don't have it already installed, download it from here Download Unity Hub. Note that there are many versions available. The latest LTS (Long Term Support) version should be a safe bet for maximum compatibility, but the latest stable release should also work. Unity Hub can be used to manage Unity installations, and is the recommended way to go about installing Unity.

As you go on completing your homework assignment, you are allowed to add extra features, make it more complex than required, or use the work here as a basis for your other future assignments. The following instructions are written so that you should be able to work independently and you can return your assignment early. If you are unfamiliar with programming, you can check out this C# tutorial before the exercises. If you are not using the headsets provided by us, you might need to somewhat adapt the instructions below.

Some materials for this homework are provided.

Now, proceed with the following step-by-step instructions to complete your assignment.

These instructions try to be as complete as possible but in order to keep the length approachable not everything is described in this document. When learning how to use Unity a good approach is using Youtube. Here is a list of channels that we find very useful, always pay attention to the fact that the solutions that they propose might be outdated or not best practice:
https://www.youtube.com/@ValemTutorials
https://www.youtube.com/@Brackeys

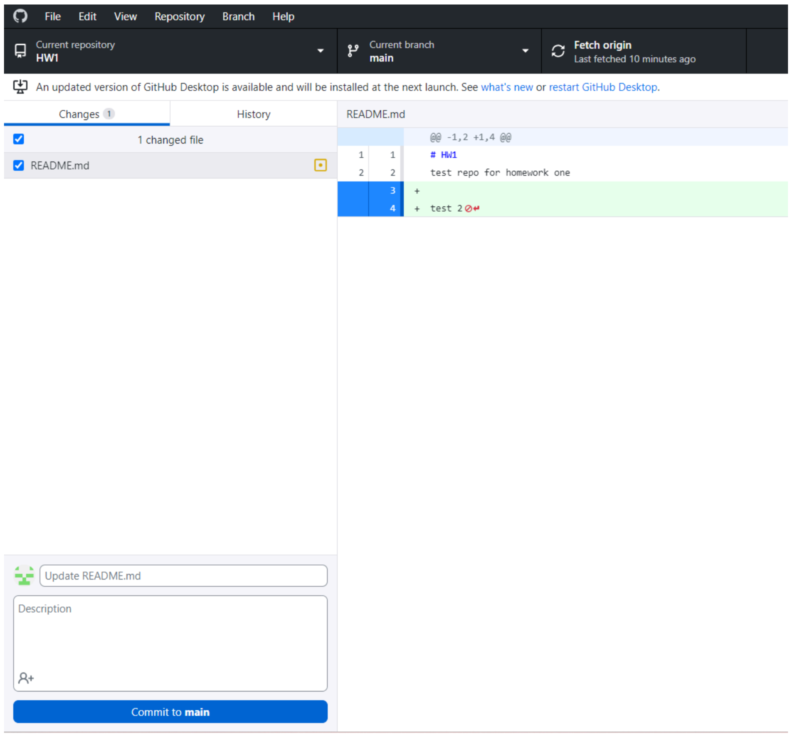
Add project on GitHub

GitHub, a web-based platform for version control and collaboration, is pivotal for Unity VR projects. It facilitates efficient code management and asset sharing, which is essential for structured VR development. We will now see how to Add your project to a git hub repository. If you don't already have it you have to create an account on their website.
For this course, we will use GitHub desktop but very often git commands are done through the command line interface.

Once you have an account proceed to create a new repository. Give it a name and make it public as you will have to share it with us. Tick the add README file option and choose the unity gitignore template (More information on what a gitignore file does here).

Once you have created a repository you can open GitHub desktop. Click on **file -> clone repository** choose the repository that you have previously created and choose a local path then press clone.

Now you have a local version of your git repo. Once you do any changes you will se on the right a list of the modified files.



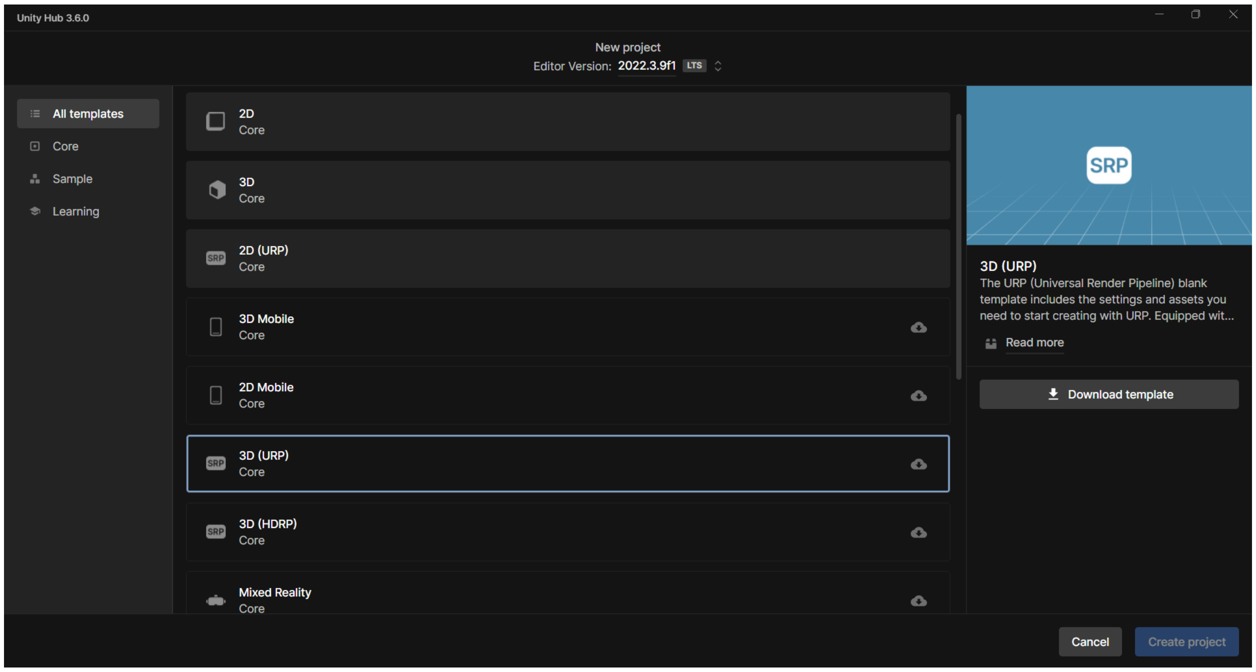
An updated version of GitHub Desktop is available and will be installed at the next launch. See what's new or restart GitHub Desktop.

```
@@ -1,2 +1,4 @@
1  1    # HW1
2  2    test repo for homework one
   3  +
   4  + test 2
```

Every time that you complete some changes and you want to create a point where you can restore the project if needed, write a title and a description of the changes and press on commit to main. Remeber that if you commit to main you have committed to the local copy of the repository and it doesen't mean that your changes are on the online repository. If you want to modify the online repo press "push origin".

Create a Project

Open Unity hub.To create a new project, click the New Project button on the top right. You should save the project inside the local repo folder. Pay attention to the fact that if you are working on a university computer all saves on PCs are deleted when they are rebooted. This means that before shutting down the computer you have to commit and push your project. We will use a 3D template, pay attention because different versions are available. We suggest to use the "3D(URP)" template. The different templates use different rendering pipelines we are gonna use the “**Universal rendering pipeline**”. If you want to know more about the topic you can refer to the Unity documentation



Editor Version: 2022.3.9f1 LTS

3D (URP)
The URP (Universal Render Pipeline) blank template includes the settings and assets you need to start creating with URP. Equipped wit...

Get acquainted with Unity’s basic features and the interface:

Interface Tutorial

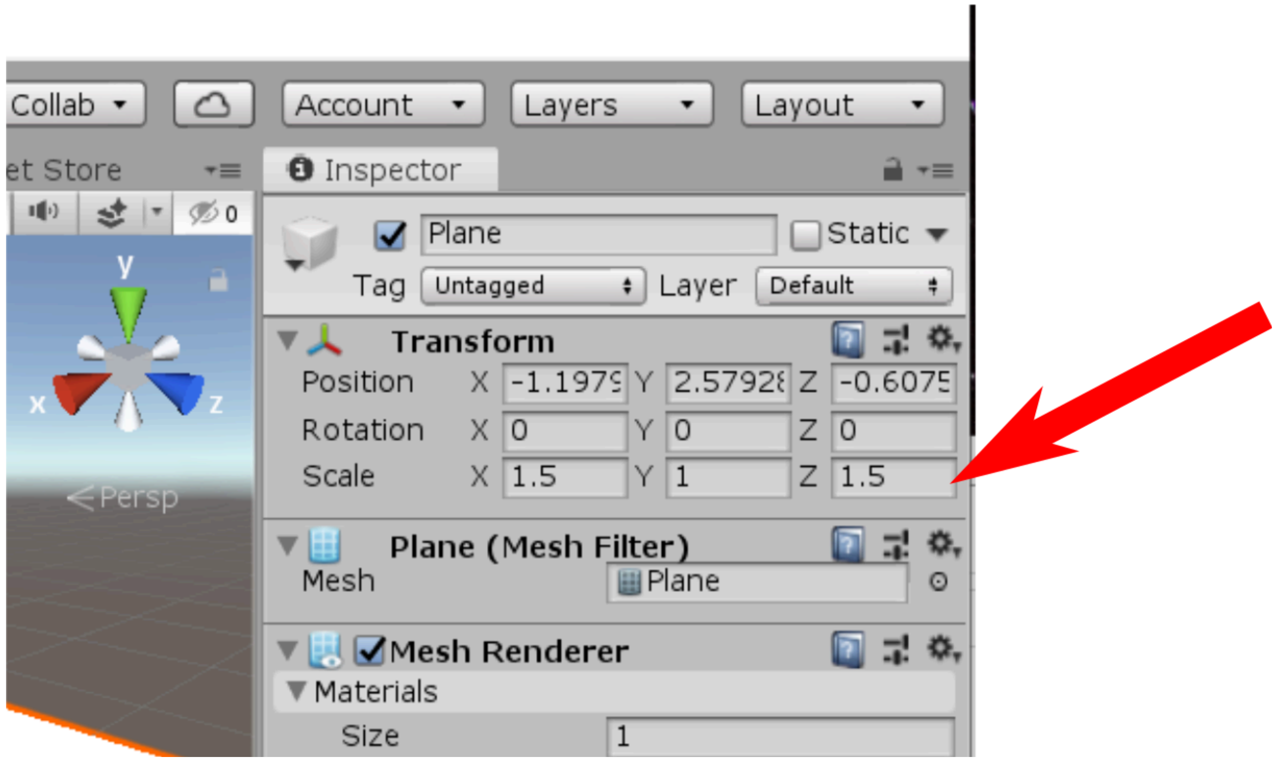
Part 1. Setting Up the Room and Learning Unity UI Basics (50pts)

1.1 The Room

Build a cubic room out of six planes. Make sure to orient these planes so the visible sides face inwards, and ensure that the player cannot walk through any of them. The room should be 15x15x15 Unity units (aka meters).

Create a plane (GameObject → 3D Object → Plane).

Unity’s default plane size is 10×10(X×Z) units. In order to make your room 15 units wide, you have to scale the plane. On the right side (in the default editor layout) you will find the Inspector window. This window provides details about the currently selected object. Select the plane in the Scene view, and the Inspector will fill with information and settings for said plane. Find the “Scale” option, and set the X and Z values to 1.5 to make your plane 15 units wide and long. Build the rest of the walls by adding new planes or duplicating the existing and adjusting scale, position and rotation. The plane has no thickness, so the value in Y can be any positive integer and remember your plane is now 15 units/meters wide.



Plane ☐ Static
Tag Untagged Layer Default

Transform
Position X -1.1979 Y 2.57928 Z -0.6075
Rotation X 0 Y 0 Z 0
Scale X 1.5 Y 1 Z 1.5

Plane (Mesh Filter)
Mesh Plane

Mesh Renderer
Materials
Size 1

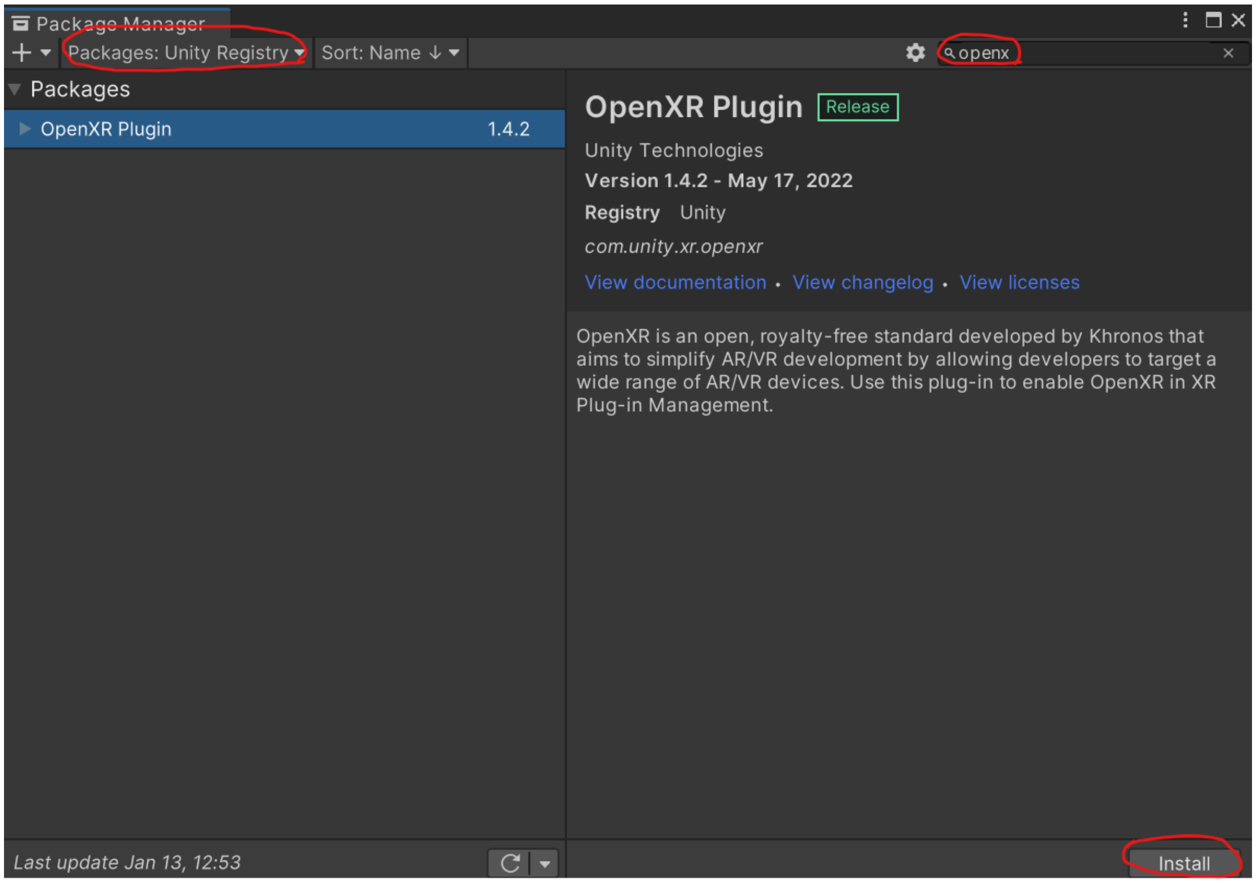
Note: The plane also rotated its axes, when you adjust rotation so the local axis changes. Make sure to account for that when rotating and moving objects!

Note: By default, your scene has a directional light in it, which illuminates your entire scene from a specified angle, from very far away, much like the sun. You'll notice that your planes do not block this light. That's because planes only block light (and render) from one side. Make sure that in your room, all six planes face inwards, and bear this in mind when creating objects in Unity in the future! For now, just delete the directional light. You will add more lights in later.

1.2 Player Controller in VR

Note: If you do not yet have access to an HMD, you should still be able to follow these steps, you will just be unable to test the results properly.

To get started with setting up VR for your project, open Package Manager (right next to Asset Store as seen in the screenshot above). In Package Manager, look for packages from Unity Registry (top left in the window), search for **OpenXR** and **XR interaction toolkit** and install them. See screenshot below for how to navigate the Package Manager window. In addition, import the controller sample, which will appear in the window after installation. That will allow you to test your controllers, see how they work and also use it as an example.



OpenXR Plugin

Unity Technologies

Version 1.4.2 - May 17, 2022

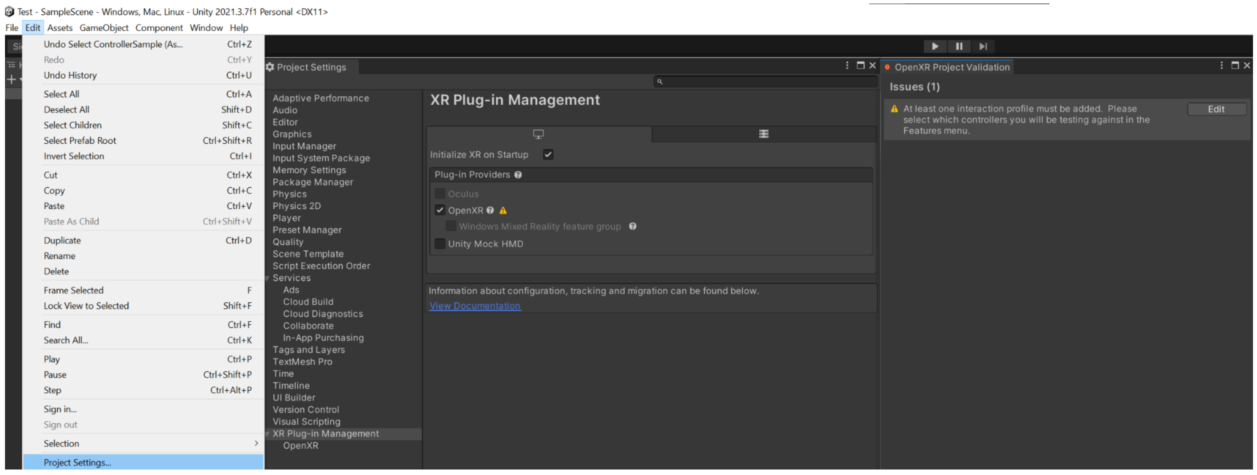
Registry Unity

com.unity.xr.openxr

OpenXR is an open, royalty-free standard developed by Khronos that aims to simplify AR/VR development by allowing developers to target a wide range of AR/VR devices. Use this plug-in to enable OpenXR in XR Plug-in Management.

Last update Jan 13, 12:53

You may then see a warning about an interaction profile. Next, we should navigate to Edit -> Project Settings -> XR Plug-in Management and select OpenXR. Although you probably have an Oculus device, OpenXR is a more universal solution, allowing your project to run on other devices as well, so we will use that.



XR Plug-in Management

Initialize XR on Startup ✓

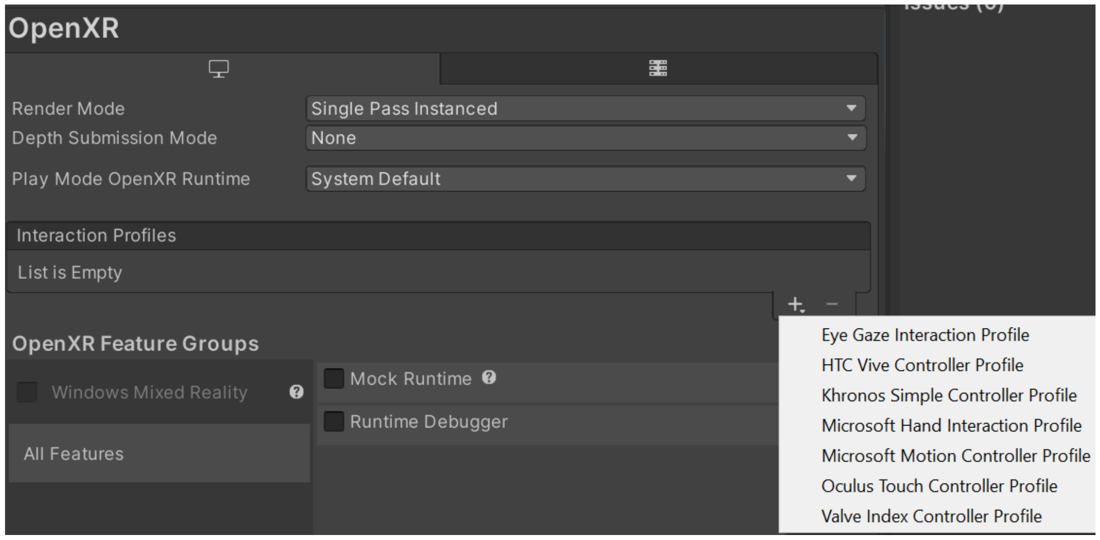
Plug-in Providers
- Oculus
- ✓ OpenXR
  - Windows Mixed Reality feature group
- Unity Mock HMD

Information about configuration, tracking and migration can be found below.

Issues (1)

At least one interaction profile must be added. Please select which controllers you will be testing against in the Features menu.

After that, select OpenXR below XR Plug-in Management and add an interaction profile. For most of you, Oculus Touch is the correct profile, so add that. If you have different controllers, choose a suitable option. Note that even when a specific profile is selected, many actions are shared between different controllers, so it will work with different controllers too.


Mock Runtime
Runtime Debugger
All Features
Eye Gaze Interaction Profile
HTC Vive Controller Profile
Khronos Simple Controller Profile
Microsoft Hand Interaction Profile
Microsoft Motion Controller Profile
Oculus Touch Controller Profile
Valve Index Controller Profile

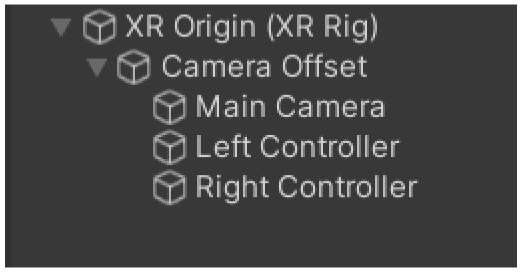
OpenXR

Render Mode: Single Pass Instanced
Depth Submission Mode: None
Play Mode OpenXR Runtime: System Default

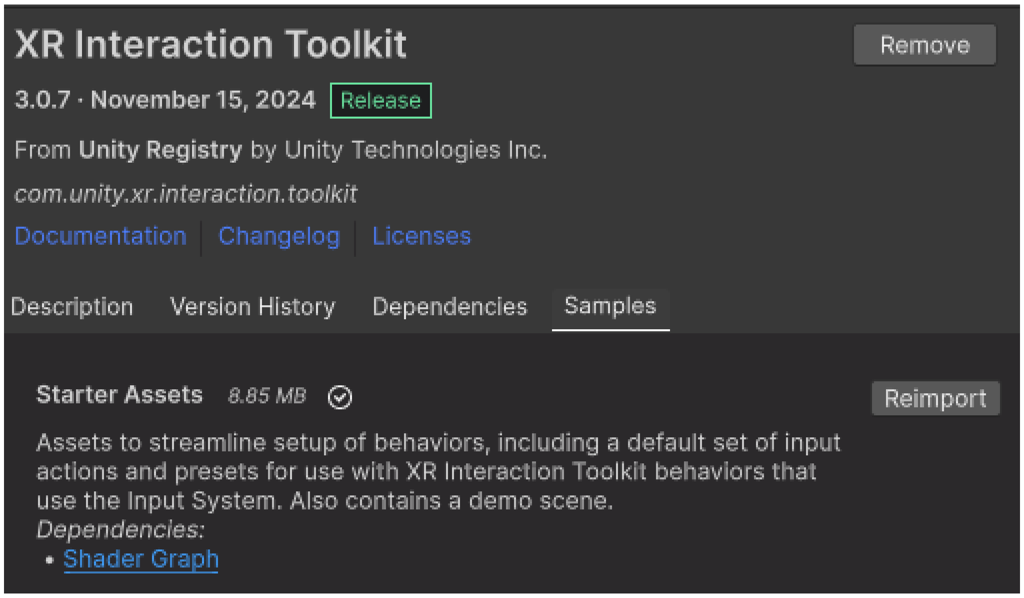
Interaction Profiles
List is Empty

OpenXR Feature Groups
Windows Mixed Reality

After that, you should be ready to start working on your project. You should be able to see the hierarchy tab on the left side of the screen. Inside it there should be an object called Main Camera which you should delete as we will use an XR Rig instead.
Click on the plus button at the top of the hierarchy and then go to XR -> Device based -> XR origin (VR). This will spawn an XR origin in your scene. This is a complex object composed of multiple game objects. By expanding it, you should see a hierarchy as in the picture.



If you do not see the controllers, it might be because Unity 6 changes some things. In that case, try getting the starter assets for XR Interaction Toolkit and grab the XR Origin (XR Rig) prefab from Assets/Samples/XR Interaction Toolkit/{version number}/Starter Assets/Prefabs and place it in your scene instead of any other XR Rigs or cameras.


XR Interaction Toolkit

3.0.7 · November 15, 2024 Release

From **Unity Registry** by Unity Technologies Inc.

com.unity.xr.interaction.toolkit

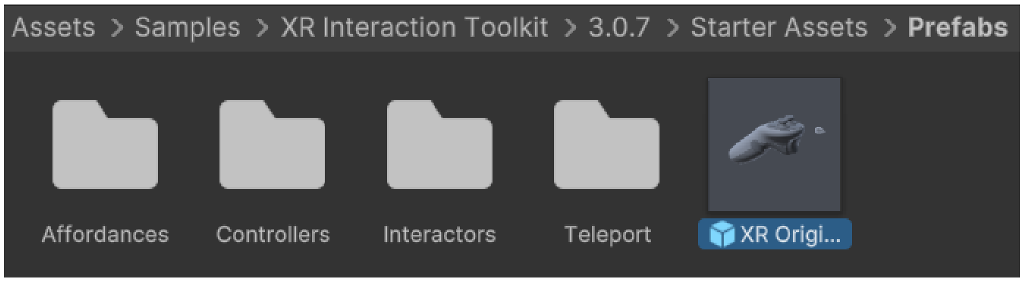
Documentation | Changelog | Licenses

Description Version History Dependencies Samples

Starter Assets *8.85 MB*

Assets to streamline setup of behaviors, including a default set of input actions and presets for use with XR Interaction Toolkit behaviors that use the Input System. Also contains a demo scene.

Dependencies:
- Shader Graph


Assets > Samples > XR Interaction Toolkit > 3.0.7 > Starter Assets > **Prefabs**

You can now press play and put on the headset. If you did not use this prefab, the controllers may not be visible when playing. In that case, press the + button and 3D object -> Cube and drag it under “Left Controller” and do the same for “Right Controller”. Make sure that the position is (0,0,0) so the position is always at the tracked position of the controller and scale it down so it isn’t too big (0.1 is appropriate). As long as you have any 3D model tracking the position of the controllers in-game, feel free to proceed.

If you have 2022 version of Unity and there are issues with getting the controllers or tracking to work, you may also want to get the XR Interaction Toolkit sample as shown above, but it will be an older version, and the XR Rig prefab may be found in Assets/Samples/XR Interaction Toolkit/2.6.3/Starter Assets/Prefabs/XR Interaction Setup. You can try placing that in your scene instead of any other camera or XR Rig.

1.3 Create the first build

When creating a project in unity it is very useful to test it in preview but before shipping is necessary to build the project in order to create a version that doesn't require Unity. A smart approach to this is to make an incremental this allows to keep the build time contained and simplifies catching errors. Always remember if it runs in preview mode it doesn't mean that it will build successfully, **do not wait the last day to build a project for the first time.** A build of a large project can take up to several hours to be completed and if it fails at 10 minutes from the end you might not have the time to understand what went wrong.

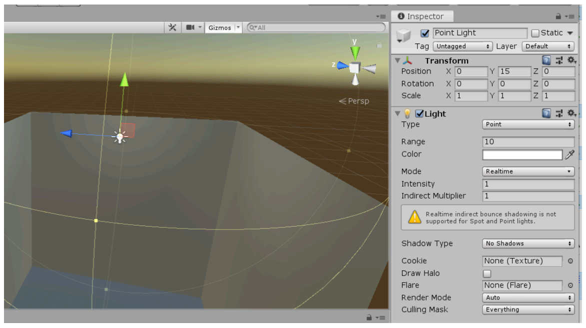
To build a project simply go to File -> build settings check that you ticked all the scenes that you want to build and then press build and run. Select a folder to contain the build and press ok. Then wait until completion.

This will create an executable (.exe) for running the build, a folder containing your scene data, a "Mono" folder, and UnityPlayer.dll.

This is a very basic build but depending on you project and platform some of you might have to make some changes to the build setting for more information check the Unity documentation.

1.4 Lighting

Create a point light (GameObject → Light → Point Light), and place it at (0,15,0). The inspector tab should have a Light component like below:



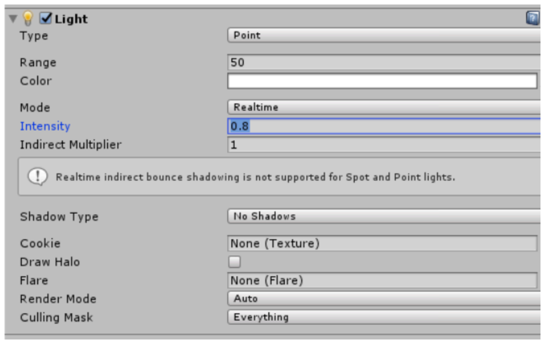
Point Light ☐ Static
Tag Untagged Layer Default

Transform
Position	X	0	Y	15	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Light
Type	Point
Range	10
Color	
Mode	Realtime
Intensity	1
Indirect Multiplier	1

Realtime indirect bounce shadowing is not supported for Spot and Point lights.

Shadow Type	No Shadows
Cookie	None (Texture)
Draw Halo	☐
Flare	None (Flare)
Render Mode	Auto
Culling Mask	Everything



Light
Type	Point
Range	50
Color	
Mode	Realtime
Intensity	0.8
Indirect Multiplier	1

Realtime indirect bounce shadowing is not supported for Spot and Point lights.

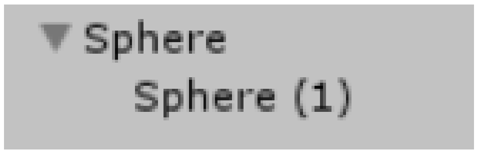
Shadow Type	No Shadows
Cookie	None (Texture)
Draw Halo	☐
Flare	None (Flare)
Render Mode	Auto
Culling Mask	Everything

Of primary importance are the range (the radius of your light), color, and intensity values. Set the shadow type to "soft shadows", and the mode to "Realtime". Set your range and intensity so that your room is brightly lit.

We recommend reading the Unity manuals for general lighting, shadows, and lighting modes. In later homework, and your course project, these settings can impact performance significantly.

1.5 Planet and Moon

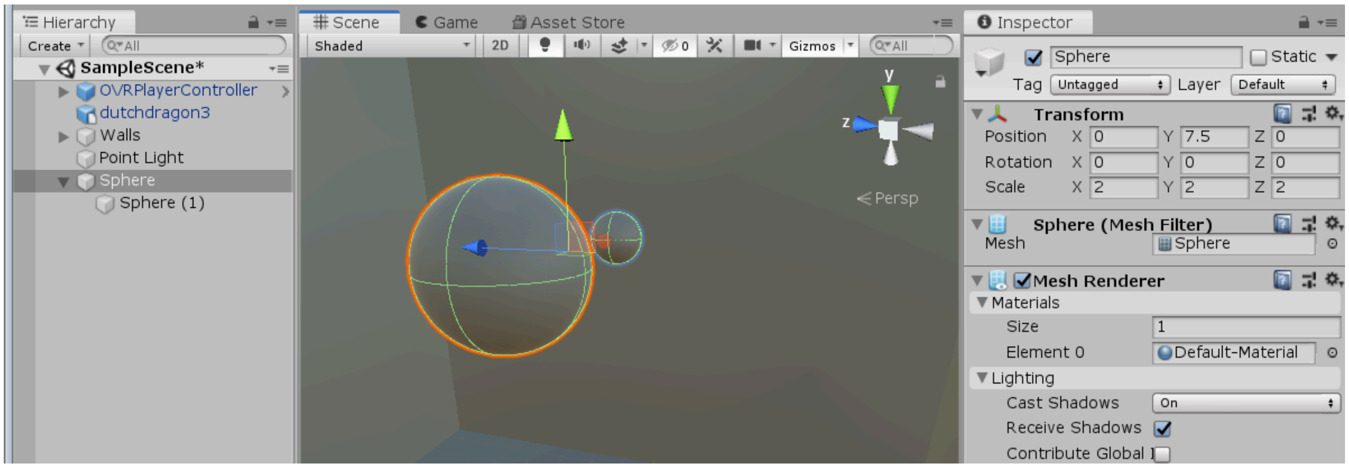
Create two spheres (GameObject → 3D → Sphere). Scale the first sphere to 2 in all directions, and place it in the center of your room. In the Hierarchy view, drag the second sphere onto the first. The result should look like this:



▼ Sphere
    Sphere (1)

Now, the second sphere is a child of the first sphere. When you change the position, rotation, or size of the parent sphere, its child will also undergo the same movement, rotation, or scaling. The (0,0,0) origin position of the child is now its parent's position, *not* the global (0,0,0) origin. That is, the child's position is an offset from the parent's position. Finally, if the parent rotates, then the child will rotate about its parent's axes, not its own axes (this will make more sense later). See the Unity hierarchy manual for more information.

Set the position of the child sphere to be (2,0,0), which is four units from the parent sphere on the X-axis. For why this is, think about the scaling of the parent sphere.



▼ SampleScene*
  ▶ OVRPlayerController
    dutchdragon3
  ▶ Walls
    Point Light
  ▼ Sphere
      Sphere (1)

Sphere Static
Tag Untagged Layer Default

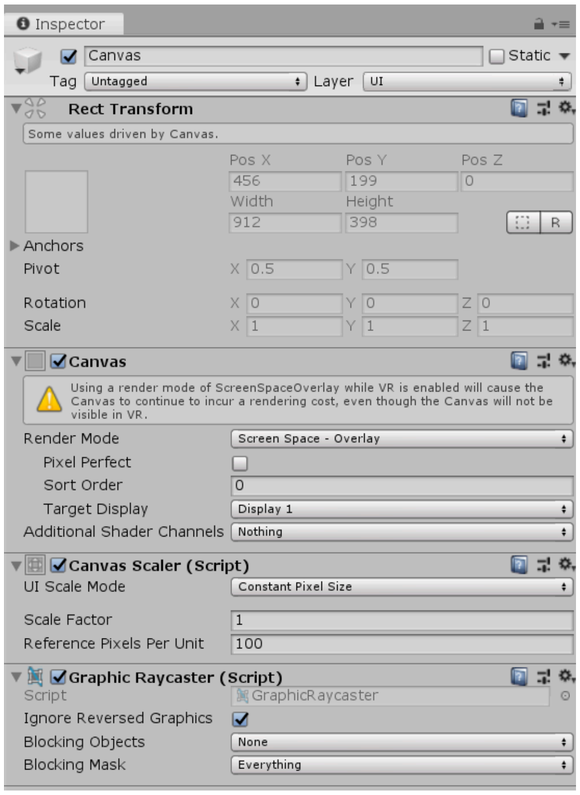
Transform
Position X 0 Y 7.5 Z 0
Rotation X 0 Y 0 Z 0
Scale X 2 Y 2 Z 2

Sphere (Mesh Filter)
Mesh Sphere

Mesh Renderer
Materials
Size 1
Element 0 Default-Material
Lighting
Cast Shadows On
Receive Shadows ✓
Contribute Global I

1.6 Text

Check out the Unity tutorial on Creating Worldspace UIs. Create a canvas (GameObject → UI → Canvas) and then create a text object under it (GameObject → UI → Text). If you create a text object directly, it will be parented under an existing canvas, which may not be desired since our PlayerRig prefab uses its own canvases. The canvas should look like this in the inspector tab:


Inspector

Canvas · Static

Tag: Untagged · Layer: UI

Rect Transform

Some values driven by Canvas.

	Pos X	Pos Y	Pos Z
	456	199	0
	Width	Height	
	912	398	

Anchors

Pivot: X 0.5 · Y 0.5

Rotation: X 0 · Y 0 · Z 0

Scale: X 1 · Y 1 · Z 1

Canvas

Using a render mode of ScreenSpaceOverlay while VR is enabled will cause the Canvas to continue to incur a rendering cost, even though the Canvas will not be visible in VR.

Render Mode: Screen Space - Overlay

Pixel Perfect: ☐

Sort Order: 0

Target Display: Display 1

Additional Shader Channels: Nothing

Canvas Scaler (Script)

UI Scale Mode: Constant Pixel Size

Scale Factor: 1

Reference Pixels Per Unit: 100

Graphic Raycaster (Script)

Script: GraphicRaycaster

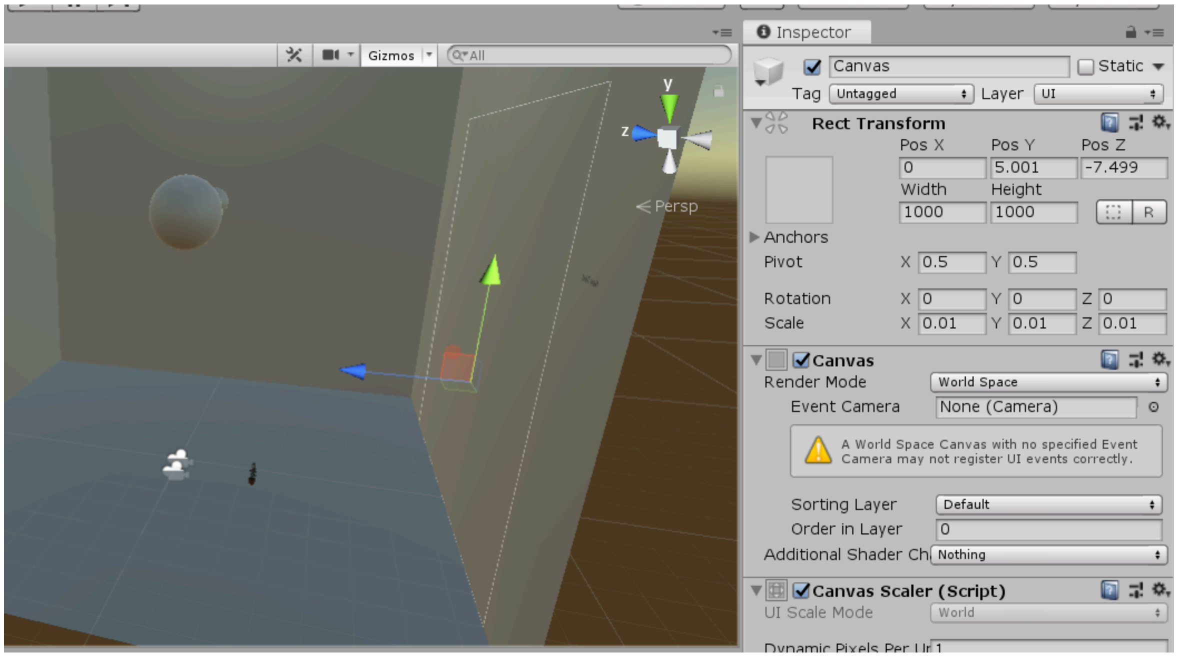
Ignore Reversed Graphics: ✓

Blocking Objects: None

Blocking Mask: Everything

First, change the Render mode from Screen Space - Overlay, to World Space. This changes the canvas from a UI element glued to the camera, to an object that is stationary in the world. Traditional UIs do not work well in VR, and we strongly discourage sticking any UI elements to the camera in your future MPs and Projects. Always attach UI elements to something in the world (see this).

Now that the canvas is a world space object, we can make it a more reasonable size. However, since the “Rect Transform”’s width and height are in units of pixels, not world units, we must first set the resolution of the canvas. Set the width and height to 1000 (that is 1000×1000 pixels). Now, shrink the canvas by setting the scale. Multiplying the canvas width and height by the scale factors gives the actual size of the canvas in world space. For example, since we set the canvas to be of size 1000×1000 pixels, using scaling factors of 0.01 would make the canvas 1000∗0.01=10 units large in world space. Adjust your text’s Rect Transform (position and with/height) to be the same as on the parent canvas, but leave the scale as it is, 1. Now, you can place your canvas against one of the walls.



Canvas Static
Tag Untagged Layer UI

Rect Transform

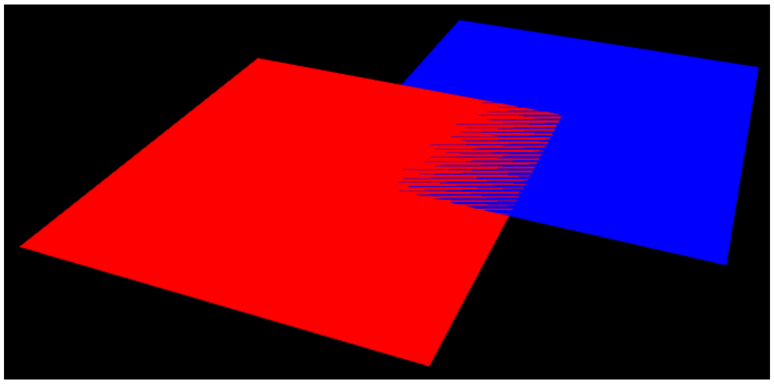
	Pos X	Pos Y	Pos Z
	0	5.001	-7.499
	Width	Height	
	1000	1000	

Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 0.01 Y 0.01 Z 0.01

Canvas
Render Mode World Space
Event Camera None (Camera)
A World Space Canvas with no specified Event Camera may not register UI events correctly.
Sorting Layer Default
Order in Layer 0
Additional Shader Ch Nothing

Canvas Scaler (Script)
UI Scale Mode World
Dynamic Pixels Per Ur 1

Offset it a small amount (like, 0.001) off of the wall it is against to avoid Z-fighting, which happens when two objects have the same depth, and Unity can't figure out which one to render. Below is an example of Z-fighting:

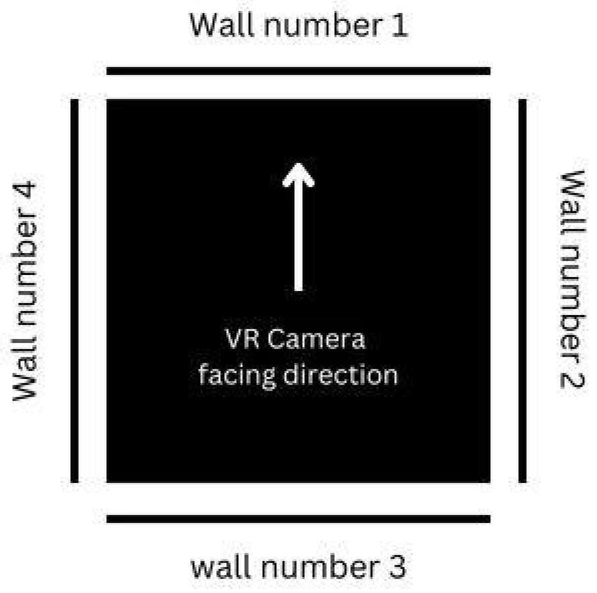


Now, you can set your text color, size, font, width, whether it wraps or overflows, etc. You can check how your project looks in VR by just clicking play on top of your scene view.

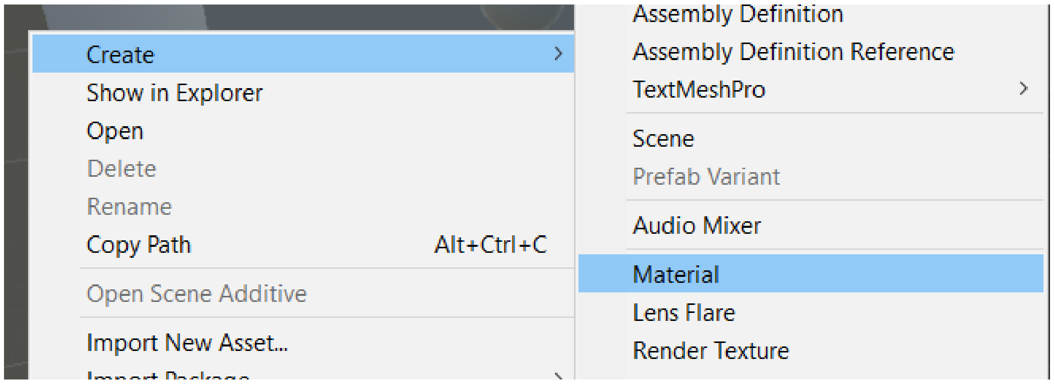
Later when we start scripting, make sure your text has the controls for your game. Make sure the text is big enough for us to read. If the text appears blurry or jagged, then increase the width and height of the canvas and text (to increase the resolution), and scale them down further.

1.7 Material

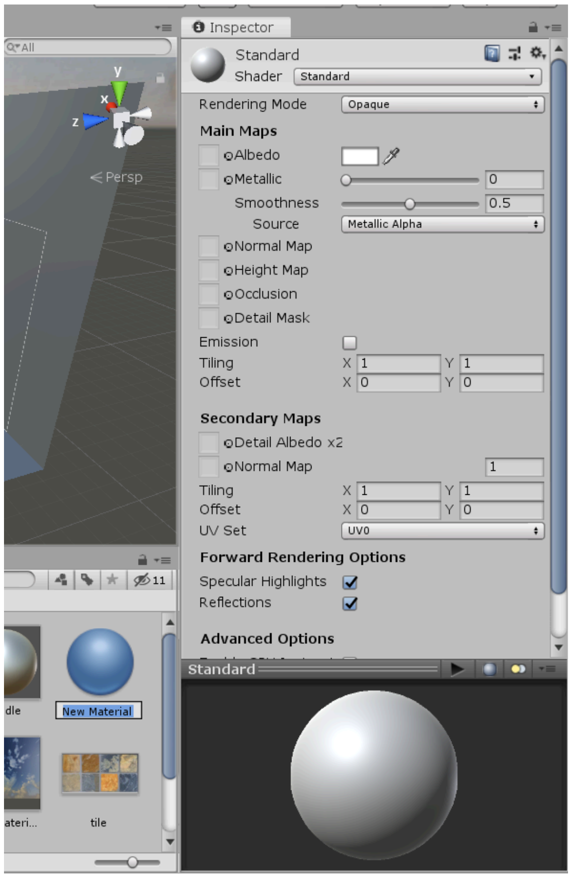
To easily understand how to complete the next section consider the walls in the room as numbered in this way:



Read the Unity materials, shaders and textures manual, focusing mainly on the Materials, for now. A base texture (tile.png), and a normal map (the weird purplish image tile-normal.png) can be found from the provided materials. To create a material, go to Assets → Create → Material.



This will generate a default material, which should show up in the inspector tab like so:


Inspector

Standard
Shader Standard

Rendering Mode Opaque

Main Maps
Albedo
Metallic 0
Smoothness 0.5
Source Metallic Alpha
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling X 1 Y 1
Offset X 0 Y 0

Secondary Maps
Detail Albedo x2
Normal Map 1
Tiling X 1 Y 1
Offset X 0 Y 0
UV Set UV0

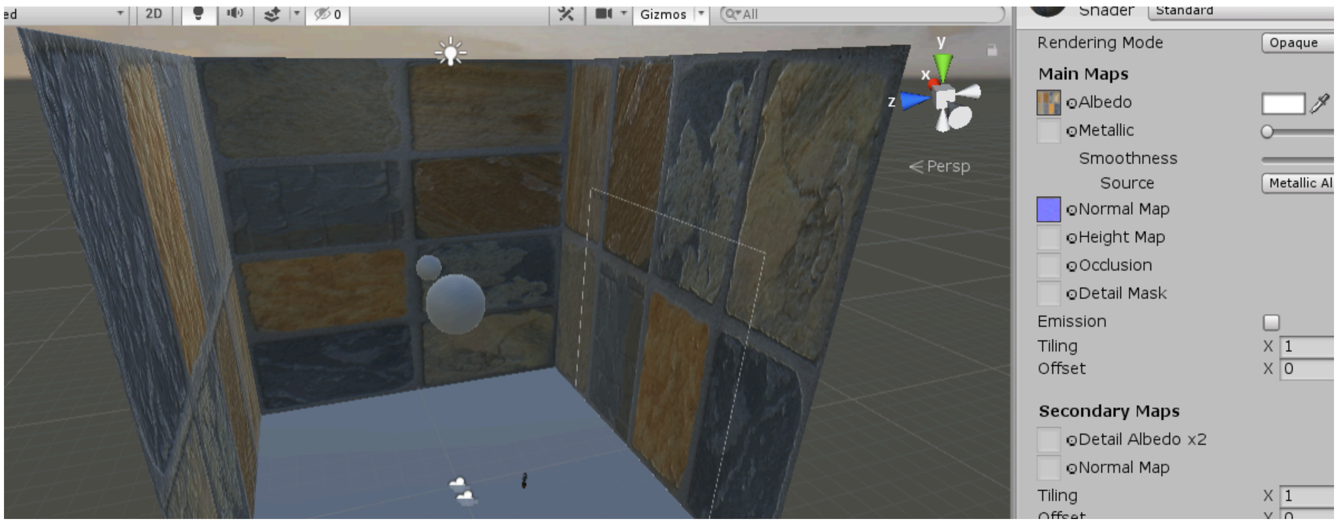
Forward Rendering Options
Specular Highlights ✓
Reflections ✓

Advanced Options

Rename it **WallMaterial1.**

Drag the tile.png image to the box labeled “Albedo”. Now, drag this material from the assets folder onto the wall 1in the Scene view. It probably doesn’t look too good. Don’t worry, it’ll get better. Drag the tile-normal.png image to the box labeled “Normal Map”. Notice how it changes the perceived material. A normal map is a trick used to give the illusion of depth on a flat surface, by telling the engine to reflect light as if there were these little bumps and pits in the material. Apply this material to a wall by dragging it onto a wall.

Create a new material, called **WallMaterial2**, apply the same albedo and normal maps as you did WallMaterial1, and apply it to the wall number 2. Right above the “Secondary Maps” subheading is the “Tiling” option, which has an option for x and y. Tiling causes a material to repeat itself on the same object, rather than covering the whole thing. So, changing tiling x to 2, means that the material will repeat once, and show up twice, in the x direction on the wall. Play with the tiling until you like the look of it. Below is an example of non tiled and tiled walls side by side.



Shader Standard
Rendering Mode Opaque
Albedo
Metallic
Smoothness
Source Metallic Al
Normal Map
Height Map
Occlusion
Detail Mask
Emission
Tiling X 1
Offset X 0
Detail Albedo x2
Normal Map
Tiling X 1
Offset X 0

In the inspector tab, right below the albedo option, there are metallic and smoothness sliders. Play around with these, and see how they affect the material. The metallic slider adjusts the reflections, and smoothness helps to enhance or subdue the normal map. Modify the metallicity and smoothness of the two wall materials so that they are clearly visually different.

Finally, create a material that has no albedo or normal map, call it **WallMaterial3**. Next to the albedo option is a small color box. Since this material has no albedo, the material will be this flat reflection color. Try and see what happens when you change the color of a material with an albedo. Apply this flat color onto the third wall in the room.

1.8 Adding Skybox

Follow the instructions in the Unity skybox manual to set up the skybox. Since the room is currently enclosed, you won't be able to see the skybox from the room. We will remedy that in the scripting section.

The skybox is from mgsvevo.

Part 2. Scripting, C# (50pts)

2.1 Basic Scripting

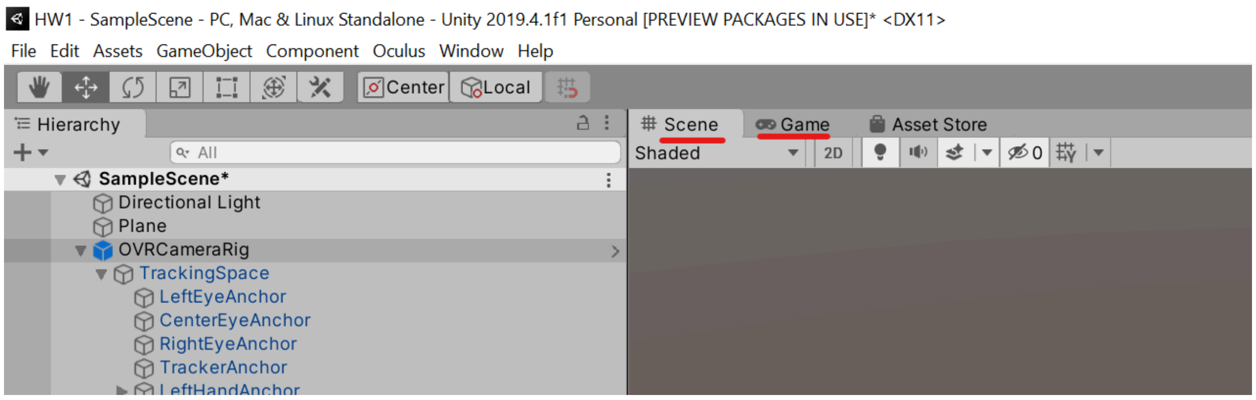
Unity scripts use the C# language. If you are unfamiliar with programming, you can check out this C# tutorial. You'll only need the basics of objects, classes, and variables for now.

We recommend reading the Unity scripts manual and the script object API.

Unity's Scripting API Reference is a useful source of information for when you are scripting new objects. For the first script, we will detail the functions that we recommend, but for the others, we expect that you will refer to the API reference if needed.

If you have access to VR HMD and controllers, use those controllers as the input device. You may test the following scripts with keyboard inputs until you get access to VR hardware. However, the returned assignment should function with the motion controllers of a VR device.

For testing without VR hardware, you may also find it useful to use the **scene view**. When you run the scene in the Unity editor, the view defaults to "Game", but it can be switched to "Scene" right after, allowing you to move the rig, simulating movement, or otherwise manipulate the scene. You can also monitor the whole scene this way and ensure everything is functioning as intended.



Some documentation about input systems is linked below. However, you may find it sufficient to simply take example from the controller sample scene.

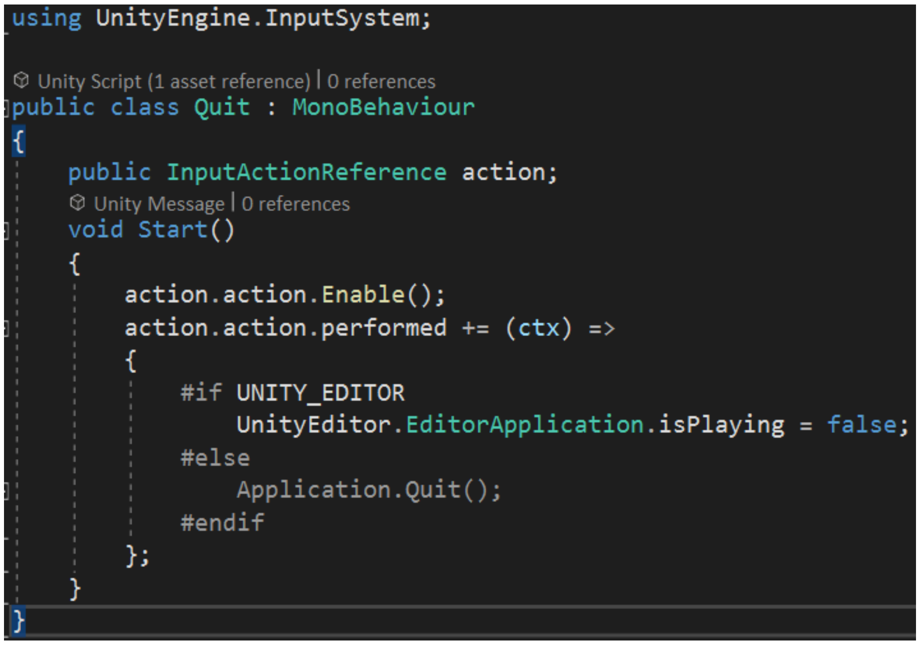
- XR Input, Unity New Input System Documentation
- Unity Legacy Input

2.2 Quit Key

Create a new script using the Assets menu (Create → C# script). To attach the script to an object, select the object, then drag the script from the assets tab to the inspector tab. (Alternatively, you can create and attach a script in one step using Add Component → New Script.) For this script, the exact object attached is unimportant.

When a Unity script is attached to a GameObject, that script will run when the game is started. Furthermore, the "this" reference in the script will refer to the object to which the script is attached.

By mimicking the input code in the controller sample scene, we can make it something like this.

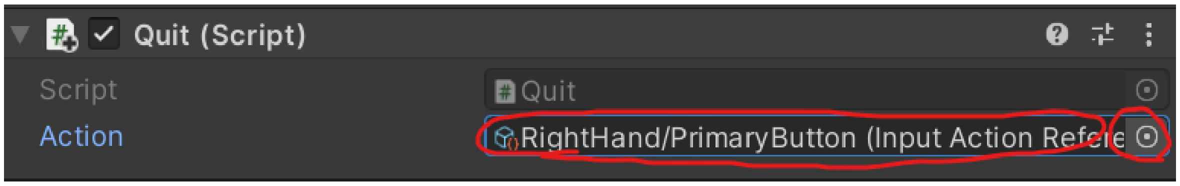


```
using UnityEngine.InputSystem;

Unity Script (1 asset reference) | 0 references
public class Quit : MonoBehaviour
{
    public InputActionReference action;
    Unity Message | 0 references
    void Start()
    {
        action.action.Enable();
        action.action.performed += (ctx) =>
        {
            #if UNITY_EDITOR
                UnityEditor.EditorApplication.isPlaying = false;
            #else
                Application.Quit();
            #endif
        };
    }
}
```

In addition, remember to set the desired action in the Unity editor. The controller sample should come with all the necessary actions to use all the input options of your controllers, and you can find and select these by pressing the circle icon on the right, but if you want to add other actions or edit them in some way, you can find the input actions file by clicking on the text after selecting an action, or you could just right click on the project assets window, Create -> Input Actions.

This documentation may also help you understand what action corresponds to what button.



Quit (Script)

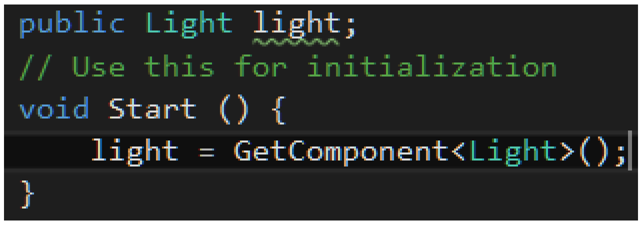
Script: Quit

Action: RightHand/PrimaryButton (Input Action Refere

Application.Quit() quits a Unity application, but it will not stop a game running in the editor. Thus, we use an #if and a Unity-specific preprocessor directive that adjusts its behavior depending on whether the project is within the editor, or a standalone executable.

2.3 Light Switch

Make a script and attach it to the point light. Our first step is to get the Light component of our point light GameObject. Read the Controlling GameObjects using Components tutorial, then add these lines to your script:

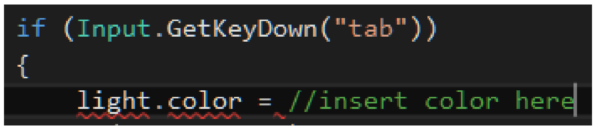


```
public Light light;
// Use this for initialization
void Start () {
    light = GetComponent<Light>();
}
```

The Start function runs as the game initially starts. This saves a reference to the Light component for later use.

Alternatively, you can declare public GameObject variables. Save your script, then navigate to the script in the inspector pane. The variable will show up in the inspector pane, and you can assign GameObjects to it by dragging them from the hierarchy tab into the variable slot! This works similarly for other variable types. You can read more about this in the Variables and the Inspector tutorial.

To change the light, write something similar to the following, but remember to do it according to the input system you are using rather than just copying the code below.



```
if (Input.GetKeyDown("tab"))
{
    light.color = //insert color here
```

For the actual light color, you can either create a new color using the new Color(red, green, blue) constructor, or one of the predefined colors. How you change the light is up to you, but pressing the controller button you designate must visibly change the light color.

2.4 Orbiting Moon

Make the moon orbit the planet. The easiest way to do this is to have the planet object constantly rotate. Since the moon object is a child of the planet object, it will also rotate around the planet. You can control the rotation and position of a GameObject with the transform object variable, an instance of the aptly named Transform class. Use the functions of this class to rotate the planet system around the Y-axis. Keep in mind that the Update function runs every frame, but frames often vary in real time length. Thus, using static rotation amounts with the Transform class will make the apparent rotation amount depend on framerate. This is generally not desirable. To use real time frame times instead, use the Time.deltaTime variable.

2.4.1 Break Out

Set up a button on the controllers to switch the player's position between an external viewing point and the room.

Like all GameObjects, the player object has a Transform. With that in mind, make a script that switches the player between a room and a new external viewing point, e.g. a small new plane a moderate distance from the room. The player should still start within the room. Pressing the button for the first time must move the player to the external viewing point. After that, pressing the button must alternate between the two locations.

You can control the rotation and position of a GameObject with the transform object variable, an instance of the aptly named Transform class. Use the functions of this class to rotate the planet system around the Y-axis.

Part 3. A More Interesting Room (10pts Extra Credit)

Instead of the cube room we laid out above, use a modeling tool to create more complicated room geometry, like a curved roof, slanted windows, multiple levels, et cetera. Please write down what you created. Your creation still needs to have the other features of the room we described above.

Blender is a free and powerful 3D modeling software, but you may also use something else if you prefer.

The default Unity modeling tools are extremely limited, so we highly recommend you familiarize yourself with another tool. It will assist greatly in your final project. You can use the room as a basis of your course project.

Another way to make a more complex room is to download and use assets from the internet. Here there are some websites that you can use:

Blender-models.com
Blend Swap
TurboSquid
Thingiverse
Sketchfab
3D Oulu and University Campus assets

Contents of the Assignment and Grading

The Room, 5pts	5	Complete and enclosed room
Lighting, 5pts	5	Bright point light in center of room ceiling
Planet and Moon, 5pts	5	Planet and moon in center of room
Text, 5pts	5	Text is sharp, clearly visible in VR and displays controls
Skybox, 5pts	5	A skybox is present from the external vantage point
Material Maps, 5pts	5	Two walls have the provided albedo and normal maps
Material Tiling, 5pts	5	Two walls have distinct tiling settings
Material Properties, 5pts	5	Two walls have altered smoothness and/or metallicity
Flat Color Material, 5pts	5	One wall has a flat color material
VR Tracking, 5pts	5	VR camera and controller tracking is working correctly
Controller inputs, 10pts	10	VR controller inputs are used
Quit Key, 10pts	10	A controller button exits the game
Light Switch, 10pts	10	A controller button changes the light color
Orbiting Moon, 10pts	10	The moon orbits the planet at a steady rate
Break Out, 10pts	10	A controller button switches the player to and from an external vantage point
More Complex Room, 10pts	10	Extra credit assignment
Total points	110	

Submission Instructions

Check the content of the assignment table above to make sure that everything required is in the delivered homework.

Step1: Record a video of the homework

1. Start play in unity (or start the builded project)
2. Press **windows key + alt + R** to record the screen.
3. Slowly show all the features required for the homework in this order:
    a. The cubes that represent the hands
    b. The light at the centre and the fact that it can change color
    c. The planet and the moon at the centre
    d. The text
    e. The walls in the order specified in the point 1.7
    f. Switch to the outside point of view and show the sky box.
    g. Now show any additional work you did to make the room more interesting
    h. Pres the key to exit and then finish the recording

Step 2: Create a Standalone Build

1. Save the project to C:\Users\<your netid>\<project name> temporarily, rather than the EWS U: drive. Local storage is faster when building.
2. Go to *File → Build Settings*.
3. Open the scene with your work.
4. Click "Add Open Scenes". You must have saved the scene to the assets folder for this to work.
5. Click "Build".
6. This will create an executable (.exe) for running the build, a folder containing your scene data, a "Mono" folder, and UnityPlayer.dll.

Step 3: Zip the Files and Submit Through Classroom Stream

1. Create a zip file containing the following items:
    - The video created at point 1
    - The .exe, .dll, Mono, AND **DATA FOLDER** created in Step 2
    - A README.txt file containing the link to the GitHub repo and any special instructions or notes you think are relevant for evaluating your assignment (like if you added anything to make the room more interesting).
2. **Make sure the executable in your submission folder runs correctly on your VR system before submitting.**

DO NOT SUBMIT YOUR ENTIRE PROJECT FOLDER!