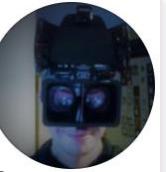


# Oculus integration SDK hand tracking in Unity

*tutorial for*

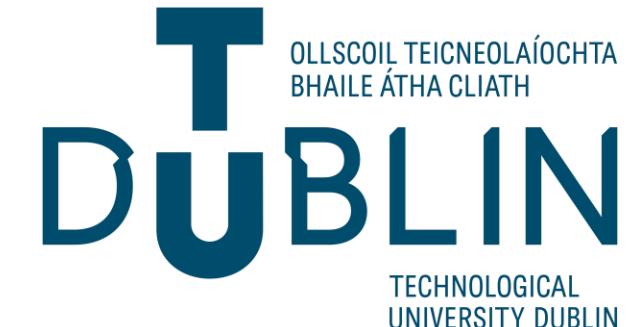
## Qualitative data collection in VR

by Krzysztof Szczurowski



Krzysztof.X.Szczurowski@mytudublin.ie

researchgate.net/profile/Krzysztof-Szczurowski-2



**Project files available at:**

<https://github.com/KrisITB/GEM2022tutorial>

# Setup

GEM2022

Unity Hub 3.3.0

Unity Hub 3.3.0

Learn

Featured Recommended Downloaded

Visit Unity Learn

Create with Code

In this official course from Unity, you will learn to Create with Code as you program your own exciting projects from scratch in C#. As you iterate with...

Read more

BEGINNER COURSE

Practical Game Accessibility

Explore the Unity Editor

Get started with the Unity Hub

Introduction to Visual Scripting

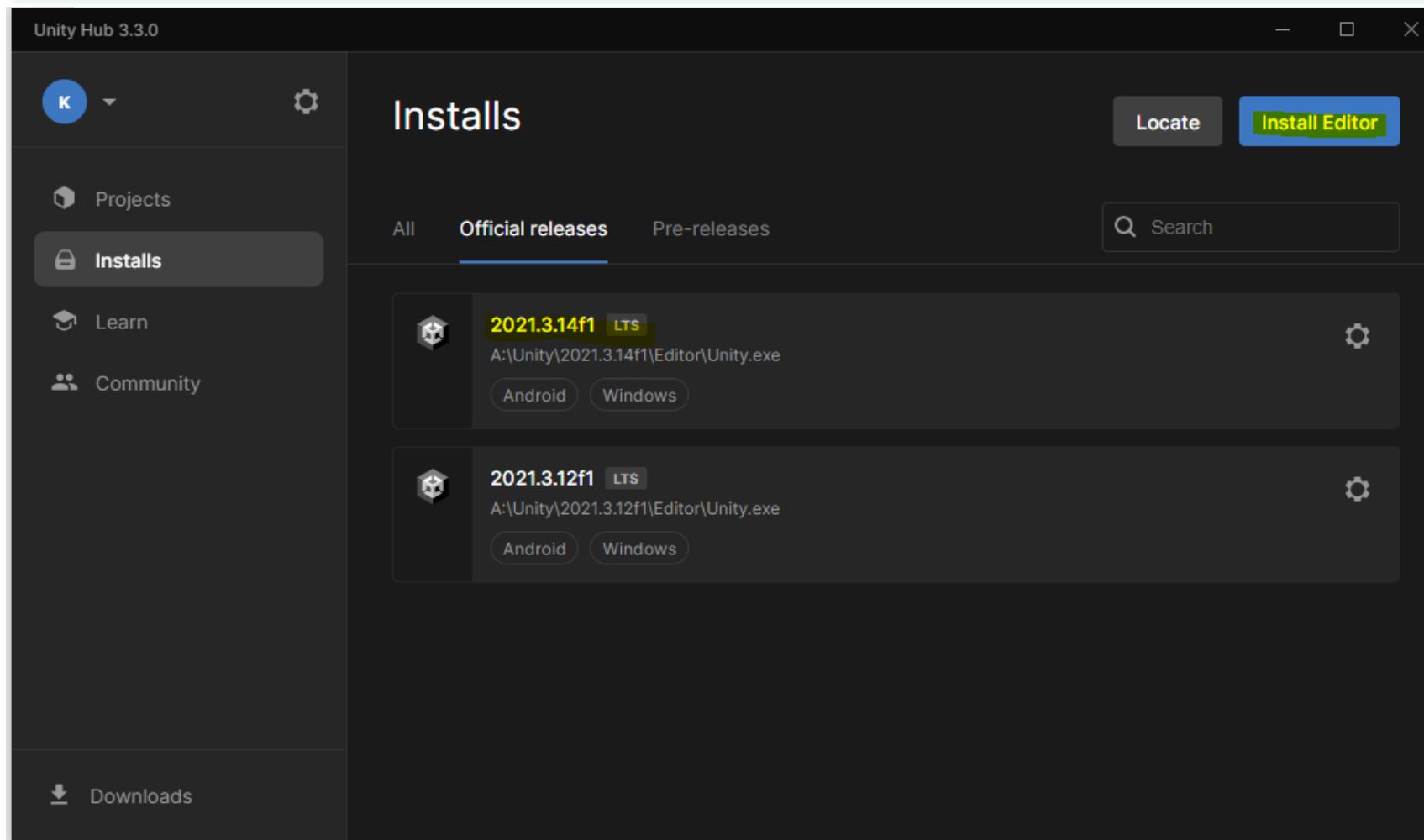
Create with VR

Downloads

# Setup

GEM2022

Unity 2021.3.14f1 (LTS)



# Setup

GEM2022

3D URP  
(Universal Render Pipeline)

Unity Hub 3.3.0

New project

Editor Version: 2021.3.12f1 LTS

All templates

Core

Sample

Learning

3D (URP)  
Core

3D (HDRP)  
Core

VR  
Core

AR  
Core



# Setup

GEM2022

Modules

Installs

Locate    Install Editor

Search

All    Official releases    Pre-releases

2021.3.11f1 LTS  
C:\Program Files\Unity\Hub\Editor\2021.3.11f1\Editor\Unity.exe

Add modules    Show in Explorer    Uninstall

PLATFORMS

Android Build Support

OpenJDK

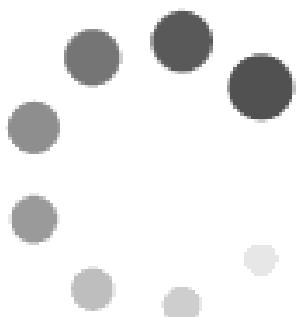
Android SDK & NDK Tools

Windows Build Support (IL2CPP)

# Setup

GEM2022

Questions?



# Setup



Oculus application (PC)  
Settings

Home

Store

Library

Events

Devices

Settings

Account   Privacy   Payment   **General**   Beta

Unknown Sources

Allow apps that have not been reviewed by Oculus to run. [Learn more](#).

OpenXR Runtime

Oculus is set as the active OpenXR runtime.

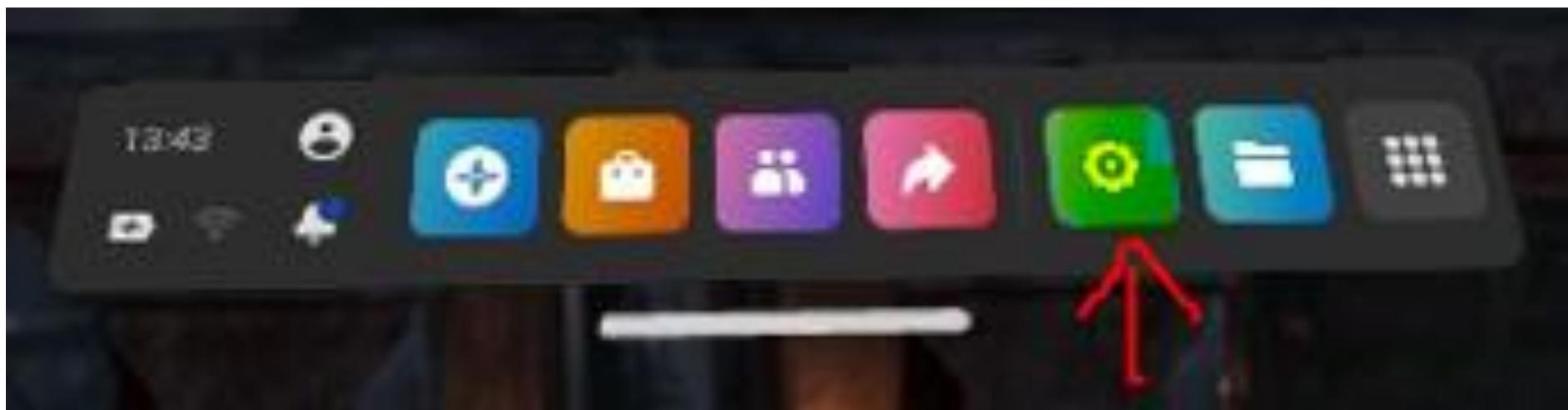
**Set Oculus as active**

Search

# Setup



Oculus (HMD)  
Settings



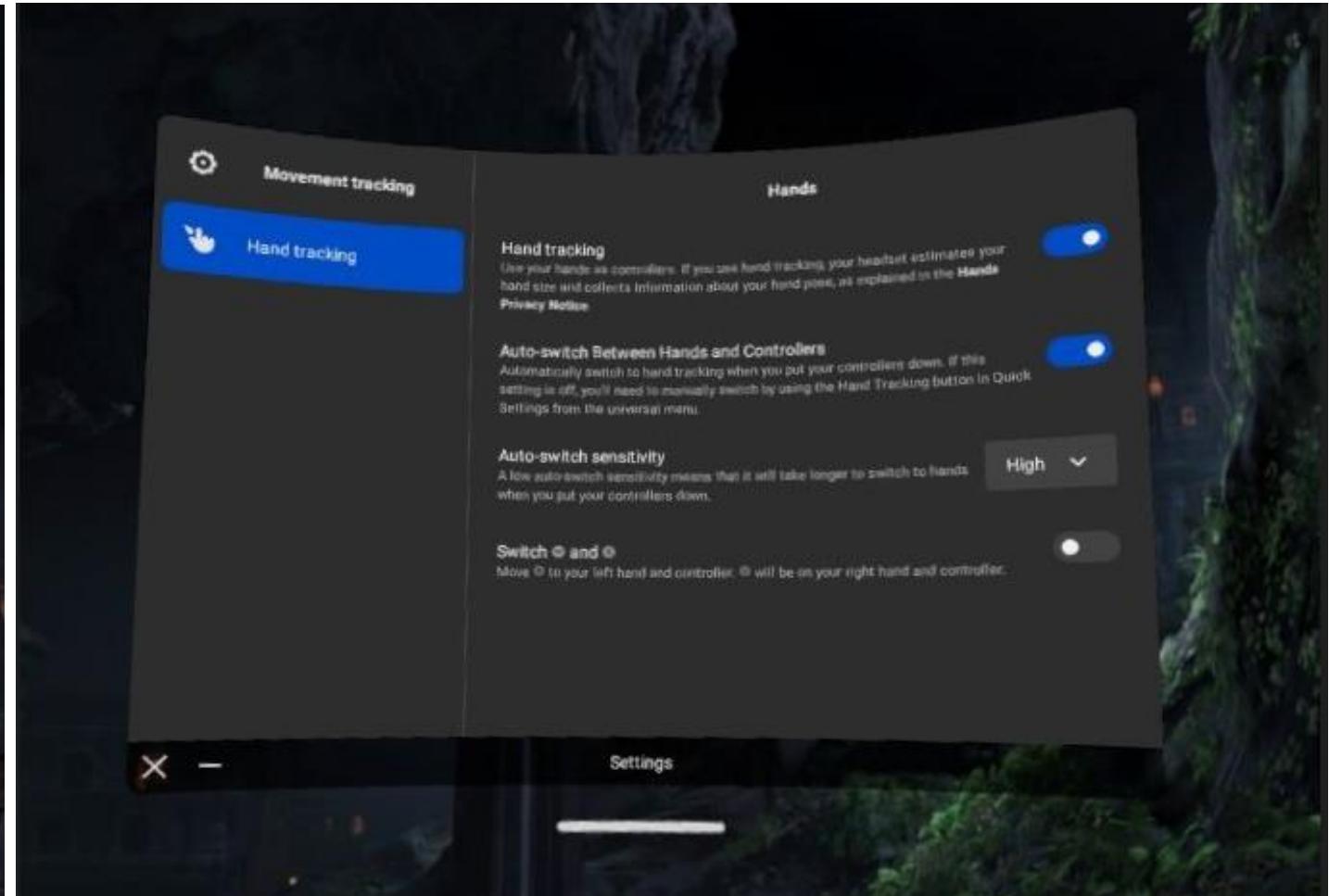
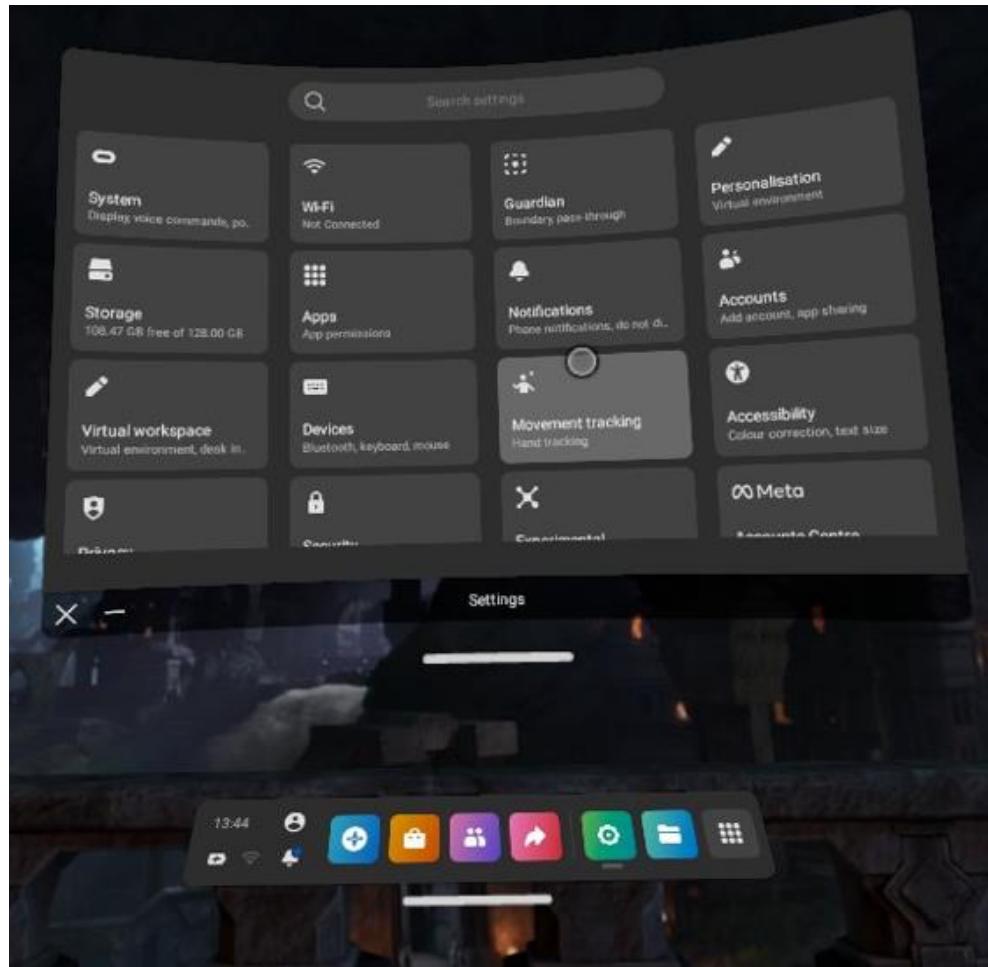
# Setup

GEM2022

Oculus (HMD)

Settings

Movement tracking -> Hand tracking



# Setup



Developer.oculus.com/  
downloads/package/  
oculus-developer-hub-win

The screenshot shows the Oculus Developer Hub interface. On the left is a sidebar with the following items:

- Device Manager (selected)
- Performance Analyzer
- File Manager
- App Distribution
- Downloads
- Meta Quest News
- Settings

The main area is titled "Devices" and contains the following sections:

- "Connect a Device" button
- "Custom Commands" section with a "Create an ADB Command" button

A modal window titled "Unverified Developer Account" is displayed at the bottom center. It contains the text: "To enable developer mode, join or create an organization, and then [verify your account](#)". Below this is a "Get Started" button.

[https://developer.oculus.com/  
documentation/unity/ts-odh/](https://developer.oculus.com/documentation/unity/ts-odh/)

# Setup



[developer.oculus.com/manage  
/organizations/create/](https://developer.oculus.com/manage/organizations/create/)

Meta Quest

Org Manager

Select Organization

Shortcuts

Create New Organization [?](#) X

**Recommendation**

Since you are an Oculus developer, we'd like to recommend custom ADB commands which can help you personalize development with your device.

[Try it out](#)

**Organization Name**

What is the legal name of your organization?  
Note: The name you choose here will be displayed publicly in the store.

Please enter a name (ex: Tuscany Studios)

We are happy to see that you're interested in developing for Oculus. Creating your developer organization for your Oculus account enables you access to features that are intended for the development of VR experiences. Remember that you have agreed to the Oculus Terms of Service and we may take action against violators. Please be a responsible member of the Oculus community.

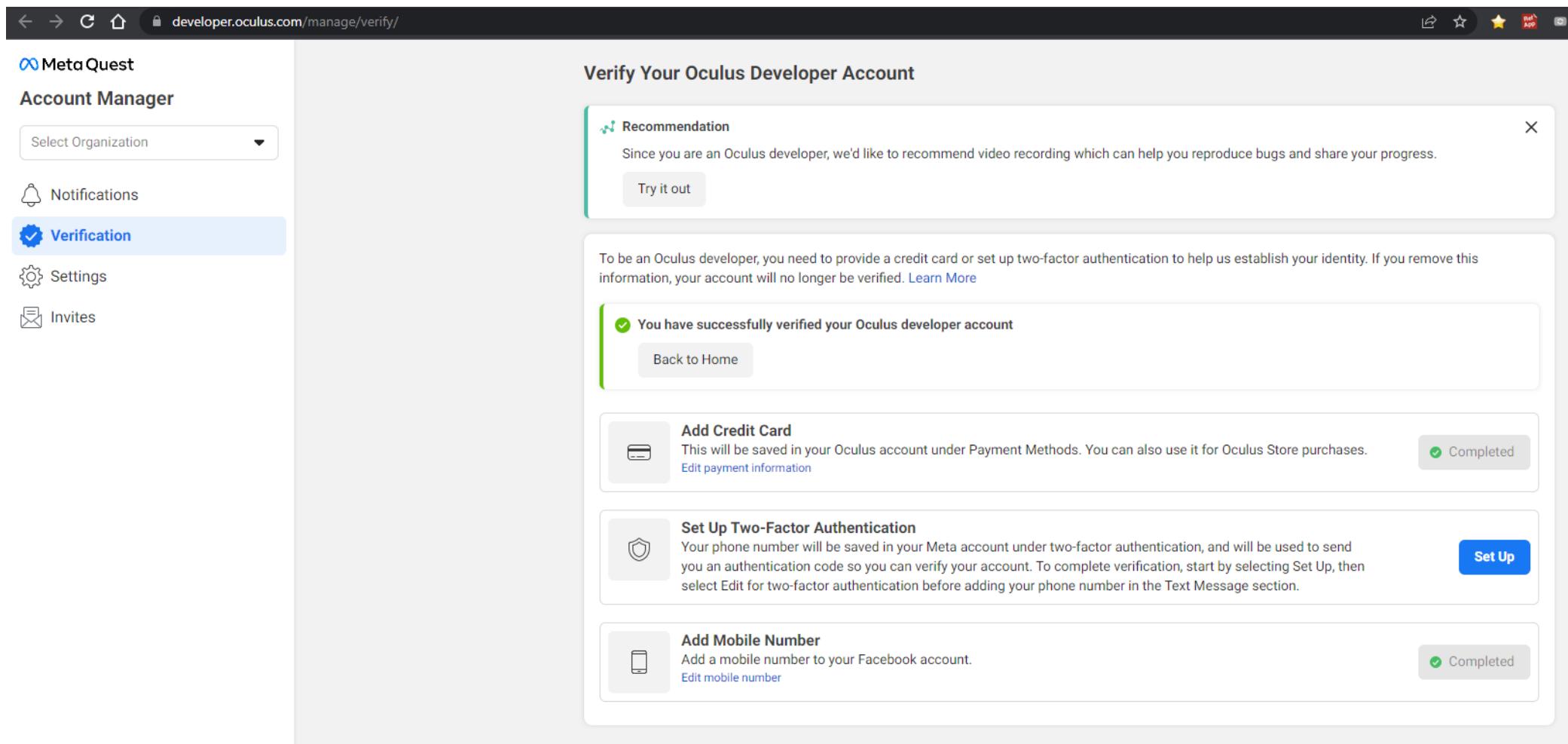
I understand  
Required

[Submit](#)

# Setup



developer.oculus.com/manage  
/verify/



The screenshot shows the 'Verify Your Oculus Developer Account' page on developer.oculus.com/manage/verify/. The left sidebar shows 'Meta Quest', 'Account Manager' (with 'Select Organization' dropdown), 'Notifications', 'Verification' (which is selected and highlighted in blue), 'Settings', and 'Invites'. The main content area has a heading 'Verify Your Oculus Developer Account'. It includes a 'Recommendation' section about video recording, a message about account verification requirements, and a success message: 'You have successfully verified your Oculus developer account'. Below these are three sections: 'Add Credit Card' (Completed), 'Set Up Two-Factor Authentication' (with a 'Set Up' button), and 'Add Mobile Number' (with a 'Completed' status).

Meta Quest

Account Manager

Select Organization

Notifications

**Verification**

Settings

Invites

Verify Your Oculus Developer Account

Recommendation

Since you are an Oculus developer, we'd like to recommend video recording which can help you reproduce bugs and share your progress.

Try it out

To be an Oculus developer, you need to provide a credit card or set up two-factor authentication to help us establish your identity. If you remove this information, your account will no longer be verified. [Learn More](#)

You have successfully verified your Oculus developer account

Back to Home

Add Credit Card

This will be saved in your Oculus account under Payment Methods. You can also use it for Oculus Store purchases.

[Edit payment information](#)

Completed

Set Up Two-Factor Authentication

Your phone number will be saved in your Meta account under two-factor authentication, and will be used to send you an authentication code so you can verify your account. To complete verification, start by selecting Set Up, then select Edit for two-factor authentication before adding your phone number in the Text Message section.

Set Up

Add Mobile Number

Add a mobile number to your Facebook account.

[Edit mobile number](#)

Completed

# Setup

GEM2022

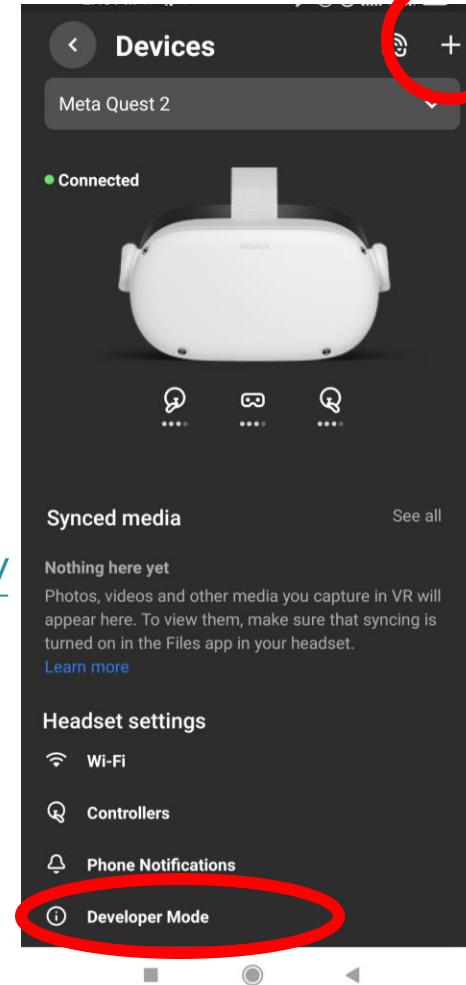
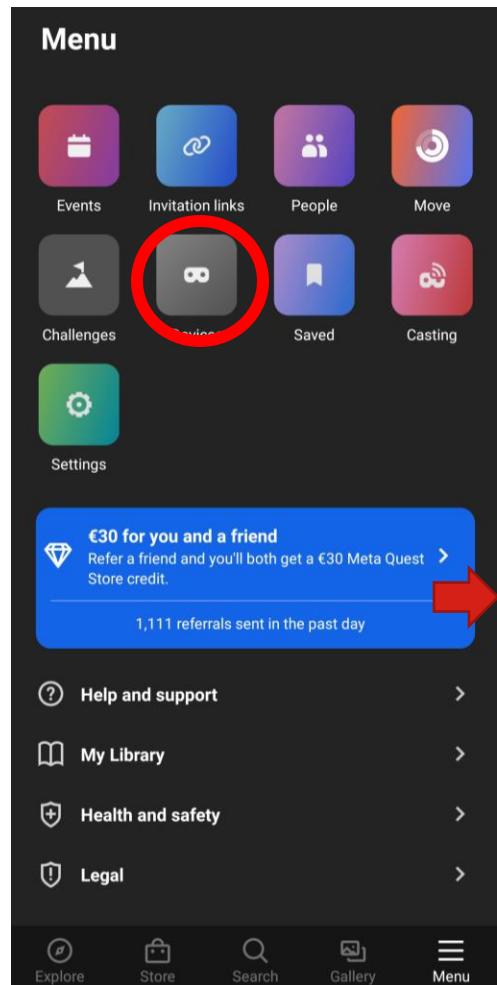
Questions?



# Setup



Oculus app



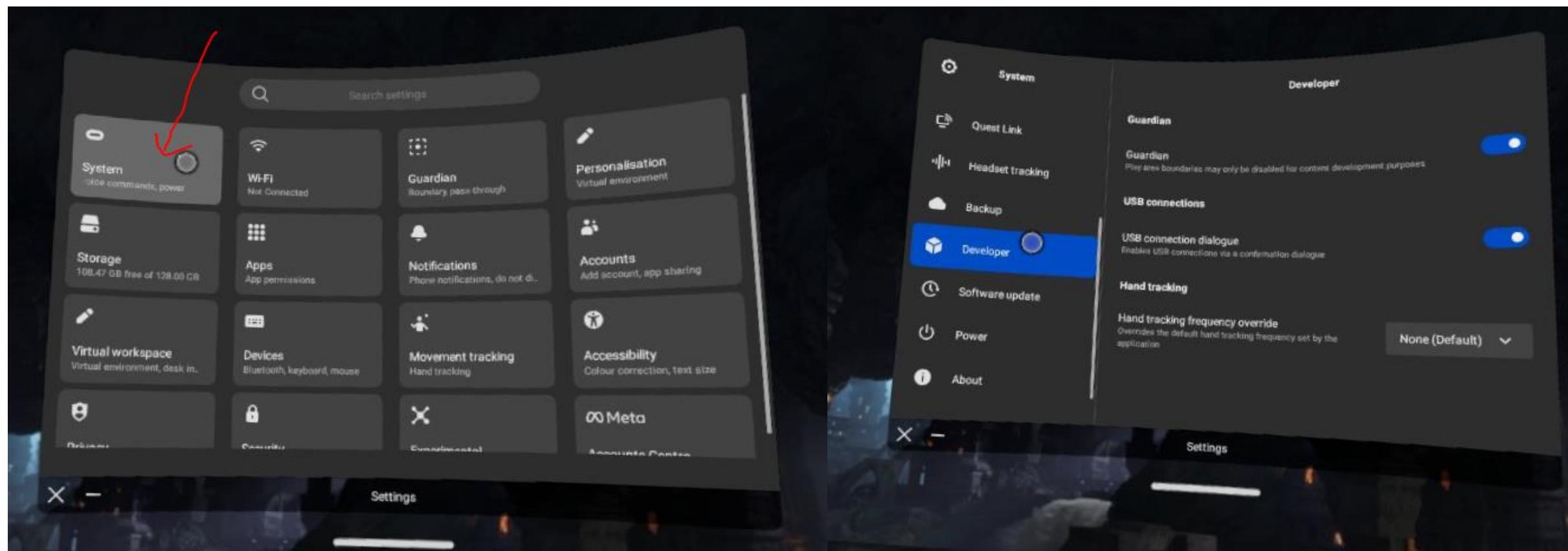
# Setup

GEM2022

Oculus (HMD)

Settings

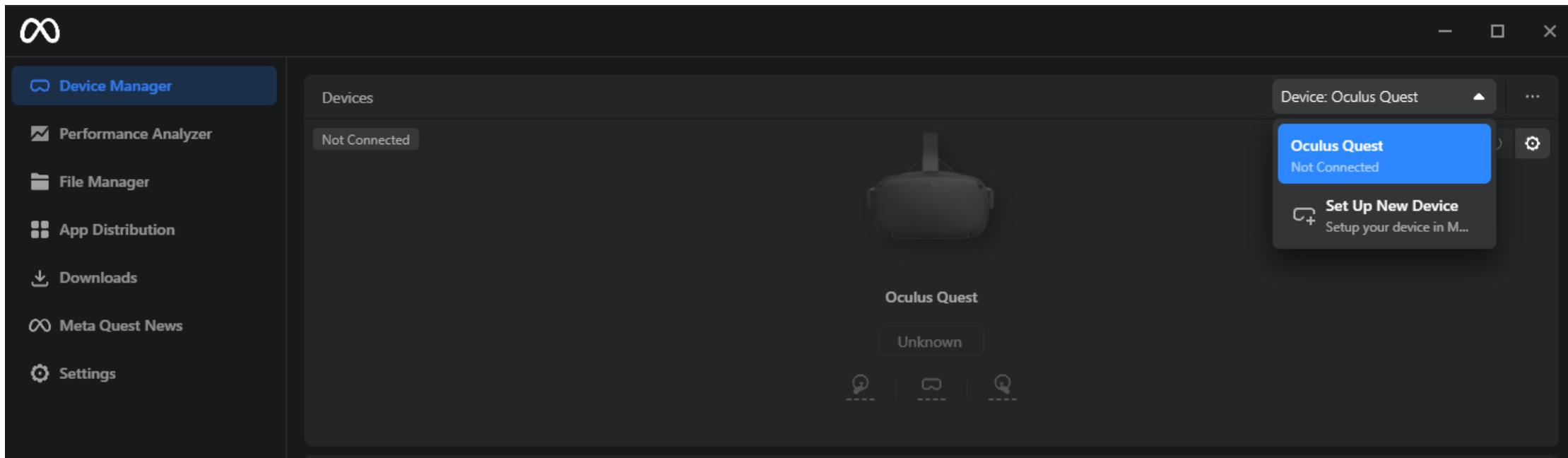
System -> Developer



# Setup



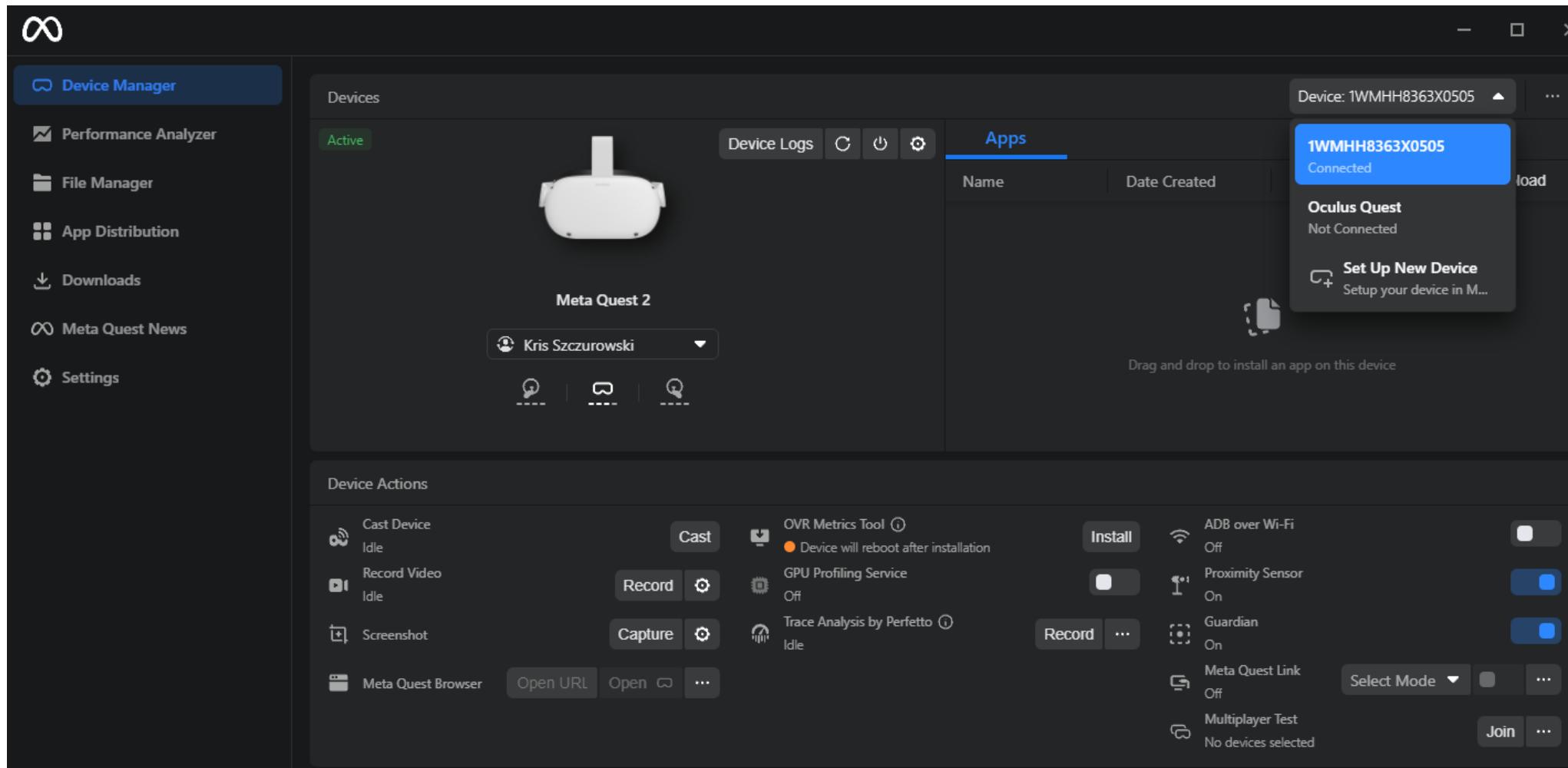
Meta Quest Developer Hub



# Setup

GEM2022

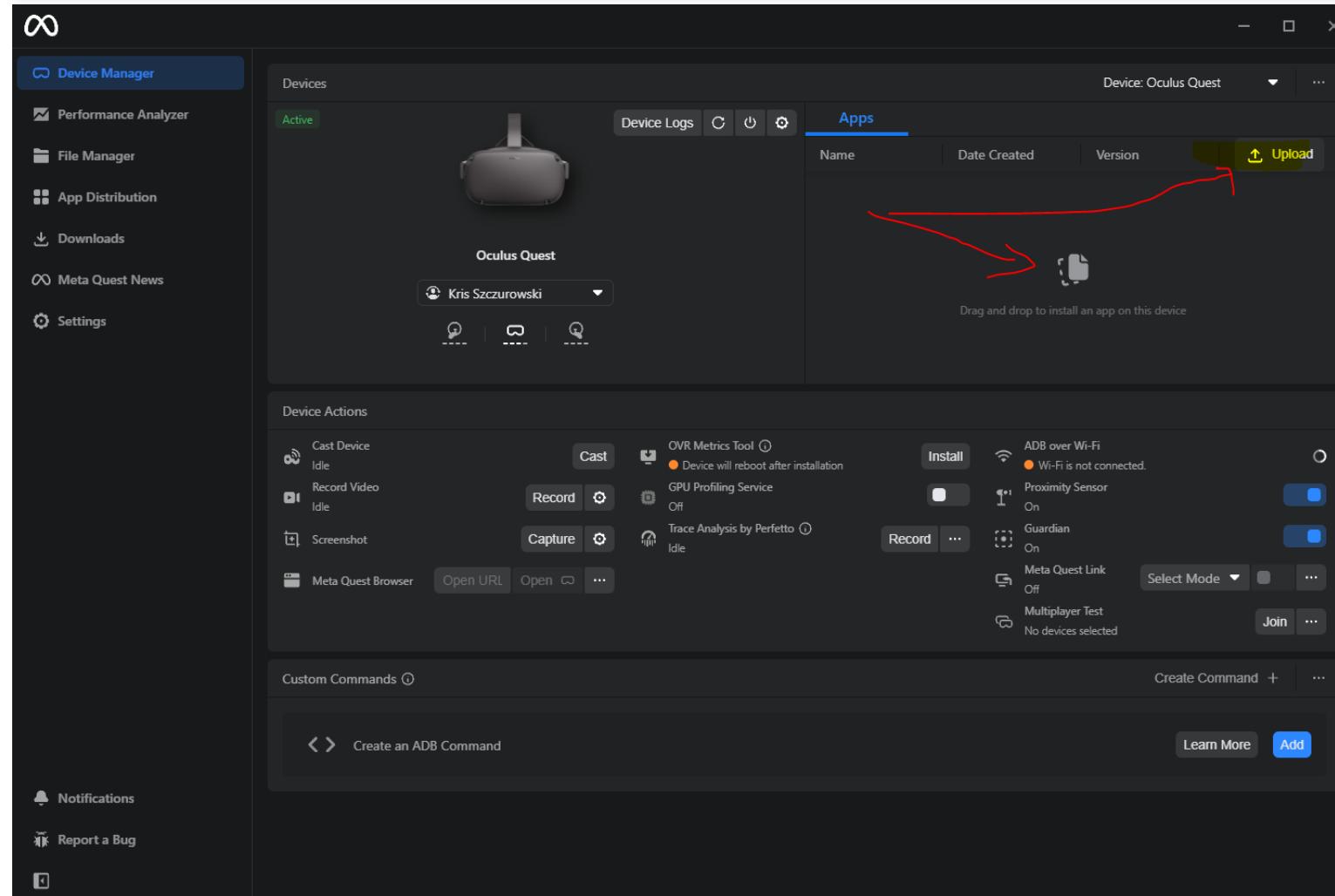
Meta Quest Developer Hub



# Setup

GEM2022

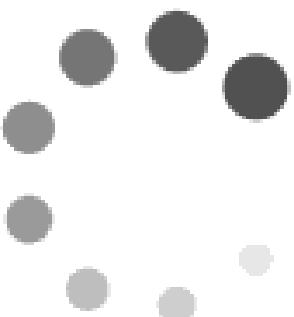
Meta Quest Developer Hub



# Setup

GEM2022

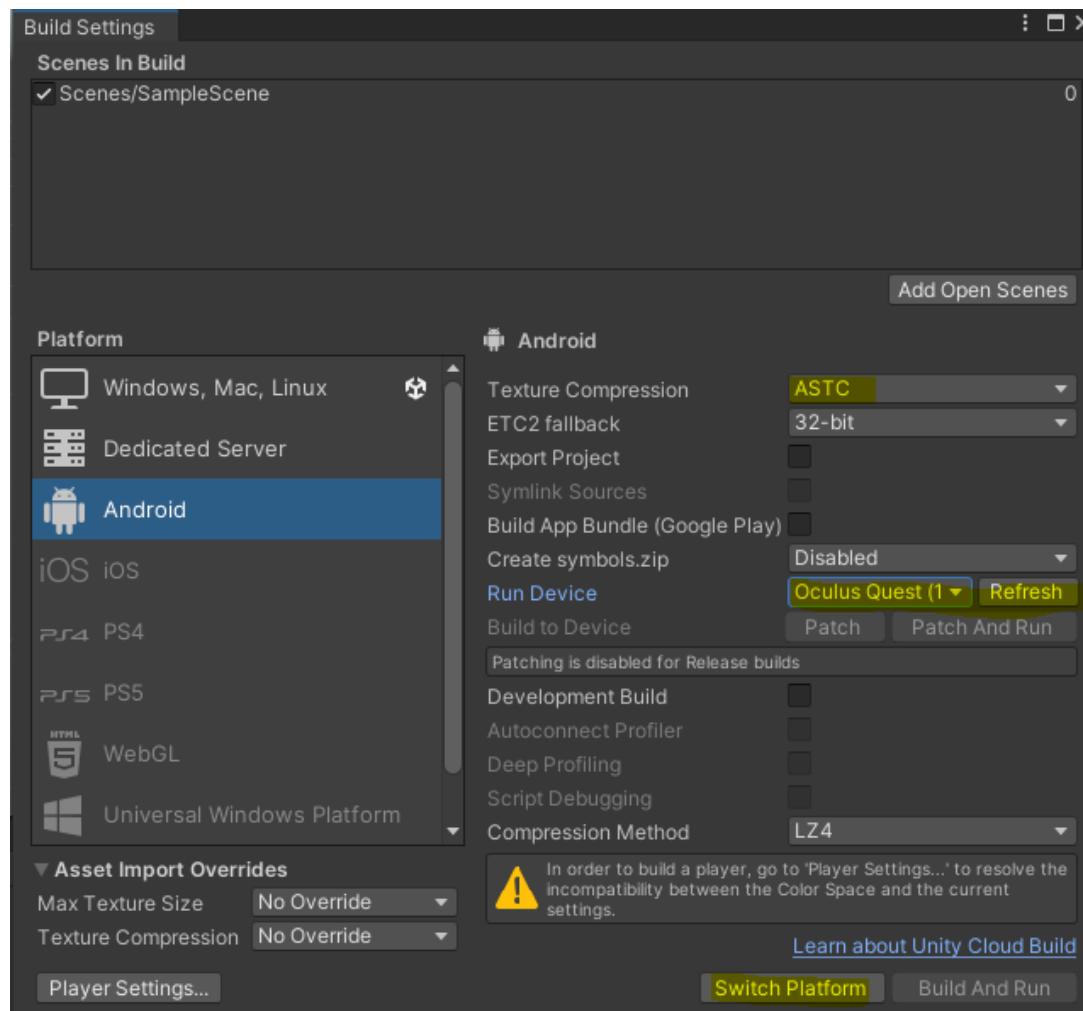
Questions?



# Setup

GEM2022

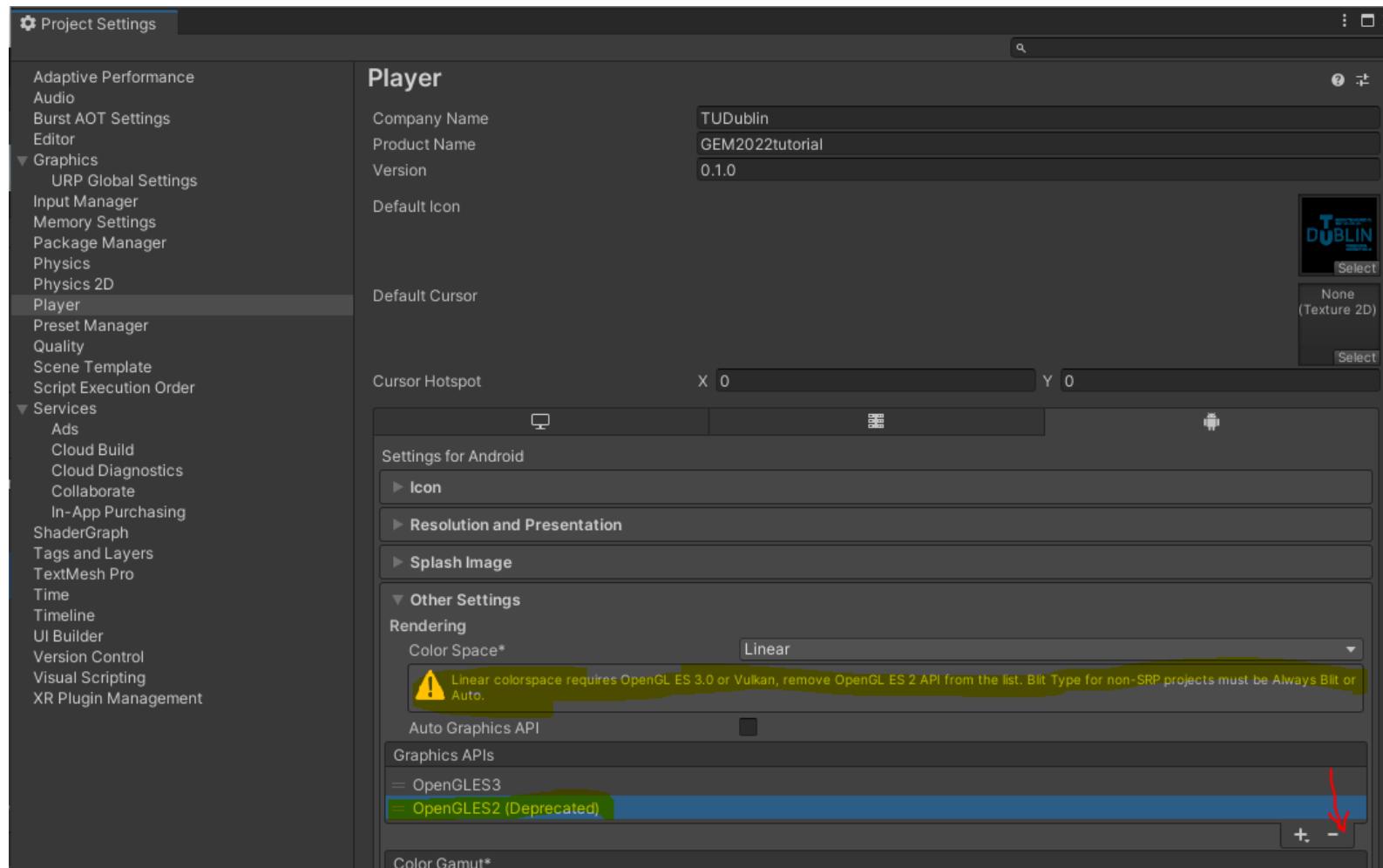
File-> Build Settings...



# Setup

GEM2022

Edit -> Project Settings...  
Player



# Setup

GEM2022

File-> Build Settings...  
Player

The screenshot shows the Unity Project Settings window with the 'Player' tab selected. The left sidebar lists various project settings categories. The 'Player' section is expanded, showing options like 'Icon', 'Resolution and Presentation', 'Splash Image', and 'Other Settings'. Under 'Other Settings', the 'Rendering' section is expanded, showing 'Color Space\*' set to 'Linear', 'Auto Graphics API' (disabled), and 'Graphics APIs' set to 'OpenGL ES3'. Below that are sections for 'Require ES3.1', 'Require ES3.1+AEP', 'Require ES3.2', 'Color Gamut\*', and 'Multithreaded Rendering\*'. The 'Multithreaded Rendering\*' section has several checkboxes checked: 'Static Batching', 'Compute Skinning\*', 'Graphics Jobs (Experimental)', and 'Texture compression format'. Under 'Texture compression format', 'ASTC' is selected. Other options include 'XYZ', 'Low Quality', and 'Lightmap Streaming'.

Project Settings

Adaptive Performance  
Audio  
Burst AOT Settings  
Editor  
Graphics  
URP Global Settings  
Input Manager  
Input System Package  
Memory Settings  
Package Manager  
Physics  
Physics 2D  
Player  
Preset Manager  
Quality  
Scene Template  
Script Execution Order  
Services  
Ads  
Cloud Build  
Cloud Diagnostics  
Collaborate  
In-App Purchasing  
ShaderGraph  
Tags and Layers  
TextMesh Pro  
Time  
Timeline  
UI Builder  
Version Control  
Visual Scripting  
XR Plug-in Management  
Oculus  
OpenXR

Player

Icon

Resolution and Presentation

Splash Image

Other Settings

Rendering

Color Space\* Linear

Auto Graphics API

Graphics APIs OpenGL ES3

Require ES3.1

Require ES3.1+AEP

Require ES3.2

Color Gamut\* sRGB

Multithreaded Rendering\*

Static Batching

Compute Skinning\*

Graphics Jobs (Experimental)

Texture compression format

ASTC

XYZ

Low Quality

Lightmap Streaming

# Setup



Edit -> Project Settings...  
Player

The screenshot shows the Unity Project Settings window for the 'GEM2022' project. The left sidebar lists various settings categories: Player, Preset Manager, Quality, Scene Template, Script Execution Order, Services (Ads, Cloud Build, Cloud Diagnostics, Collaborate, In-App Purchasing), ShaderGraph, Tags and Layers, TextMesh Pro, Time, Timeline, UI Builder, Version Control, Visual Scripting, and XR Plugin Management.

**Override Default Package Name**

- Package Name: com.TUDublin.GEM2022tutorial
- Version\*: 0.1.0
- Bundle Version Code: 1
- Minimum API Level: Android 6.0 'Marshmallow' (API level 23) (selected)
- Target API Level: Automatic (highest installed)

**Configuration**

- Scripting Backend: IL2CPP (selected)
- Api Compatibility Level\*: .NET Standard 2.1
- C++ Compiler Configuration
- Use incremental GC
- Assembly Version Validation (editor only): checked
- Mute Other Audio Sources\*

**Target Architectures**

- ARMv7
- ARM64 (selected)
- x86 (Chrome OS)
- x86-64 (Chrome OS)

# Setup



Edit -> Project Settings...  
XR Plug-in Management

Project Settings

Adaptive Performance  
Audio  
Burst AOT Settings  
Editor  
► Graphics  
Input Manager  
Memory Settings  
Package Manager  
Physics  
Physics 2D  
Player  
Preset Manager  
Quality  
Scene Template  
Script Execution Order  
► Services  
ShaderGraph  
Tags and Layers  
TextMesh Pro  
Time  
Timeline  
UI Builder  
Version Control  
Visual Scripting  
XR Plugin Management

## XR Plugin Management

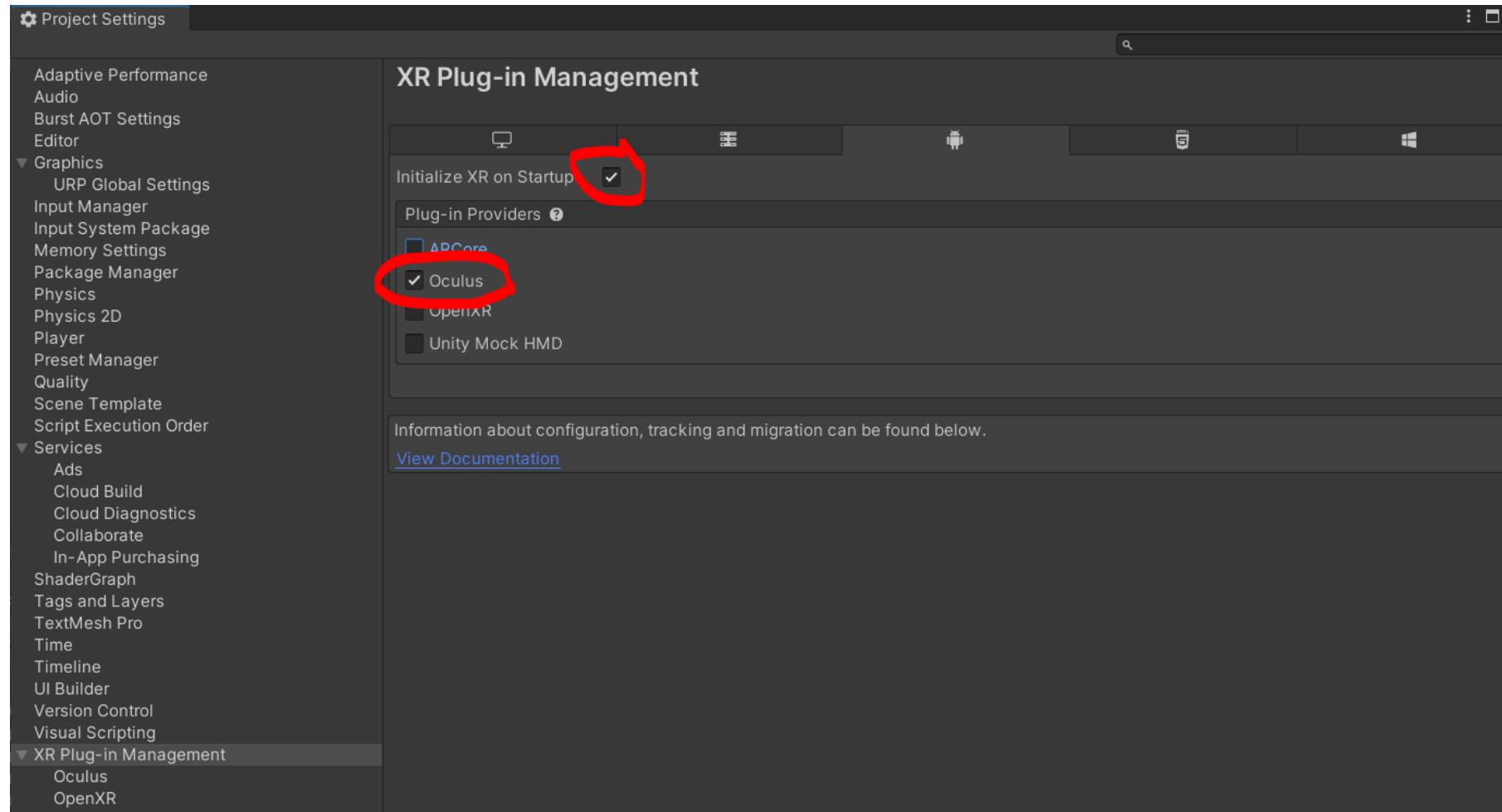
In order to use the new XR Plugin system you need to install the XR Plugin Management package. Clicking the button below will install the latest XR Plugin Management package and allow you to configure your project for XR.

[Install XR Plugin Management](#)

# Setup



Edit -> Project Settings...  
XR Plug-in Management



The screenshot shows the Unity Project Settings window with the "XR Plug-in Management" tab selected. The left sidebar lists various project settings categories. The main panel is titled "XR Plug-in Management" and contains two sections: "Initialize XR on Startup" (with a checked checkbox) and "Plug-in Providers". Under "Plug-in Providers", the "Oculus" provider is selected (indicated by a checked checkbox) and highlighted with a red circle. Other providers listed are "OpenXR" and "Unity Mock HMD". Below the provider list is a note about configuration and migration, followed by a link to "View Documentation".

# Setup



Project Settings

Adaptive Performance  
Audio  
Burst AOT Settings  
Editor  
▼ Graphics  
    URP Global Settings  
    Input Manager  
    Input System Package  
    Memory Settings  
    Package Manager  
    Physics  
    Physics 2D  
    Player  
    Preset Manager  
    Quality  
    Scene Template  
    Script Execution Order  
▼ Services  
    Ads  
    Cloud Build  
    Cloud Diagnostics  
    Collaborate  
    In-App Purchasing  
    ShaderGraph  
    Tags and Layers  
    TextMesh Pro  
    Time  
    Timeline  
    UI Builder  
    Version Control  
    Visual Scripting  
▼ XR Plug-in Management  
    Oculus

**Oculus**

	Multiview
Stereo Rendering Mode	<input checked="" type="checkbox"/>
Low Overhead Mode (GLES)	<input checked="" type="checkbox"/>
Optimize Buffer Discards (Vulkan)	<input checked="" type="checkbox"/>
Phase Sync	<input checked="" type="checkbox"/>
Symmetric Projection (Vulkan/Quest 2)	<input checked="" type="checkbox"/>
Subsampled Layout (Vulkan/Quest 2)	<input checked="" type="checkbox"/>

**Target Devices**

Quest	<input checked="" type="checkbox"/>
Quest 2	<input checked="" type="checkbox"/>

System Splash Screen  
None (Texture 2D)

► Experimental

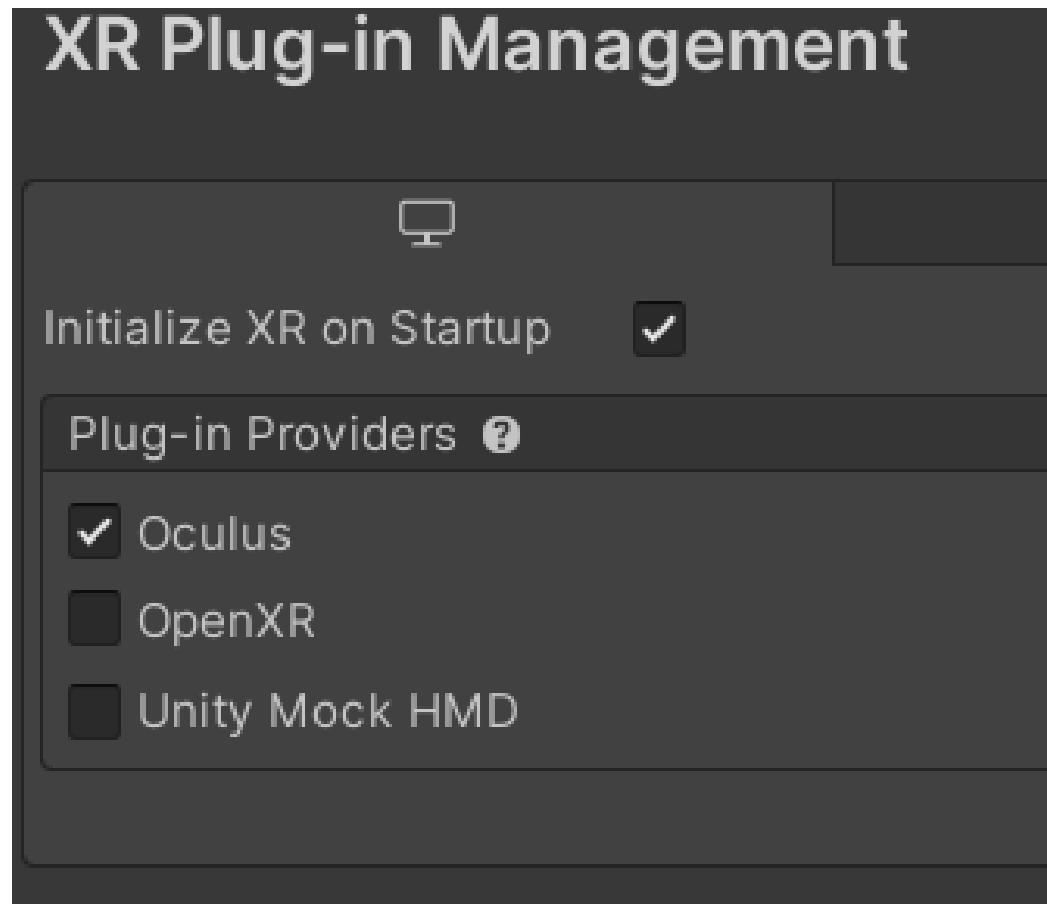
Edit -> Project Settings...  
XR Plug-in Management  
Oculus -> Android

# Setup



Edit -> Project Settings...  
XR Plug-in Management  
PC

For preview:



# Setup



Edit -> Project Settings...  
Quality

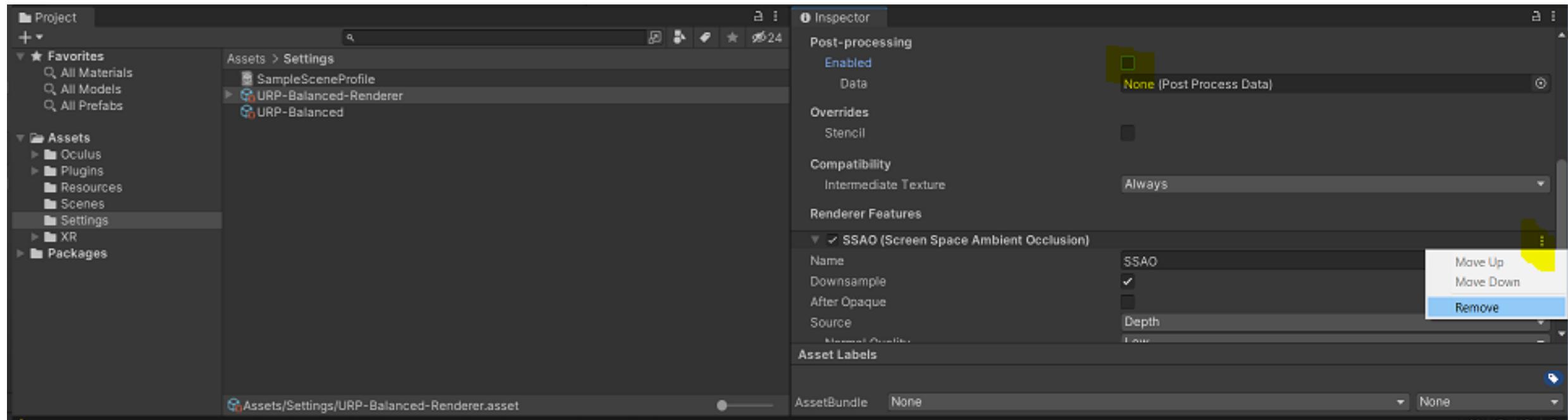
Project Settings

- Adaptive Performance
- Audio
- Burst AOT Settings
- Editor
- Graphics
  - URP Global Settings
  - Input Manager
  - Input System Package
  - Memory Settings
  - Package Manager
  - Physics
  - Physics 2D
  - Player
  - Preset Manager
  - Quality
- Scene Template
- Script Execution Order
- Services
  - Ads
  - Cloud Build
  - Cloud Diagnostics
  - Collaborate
  - In-App Purchasing
  - ShaderGraph
  - Tags and Layers
  - TextMesh Pro
  - Time
  - Timeline
  - UI Builder
  - Version Control
  - Visual Scripting
- XR Plug-in Management
  - Oculus
  - OpenXR

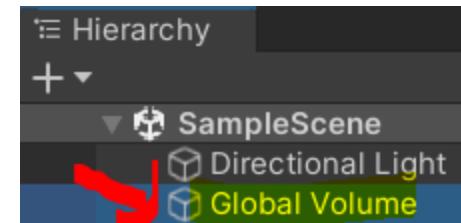
# Setup



Assets -> Settings  
URP-Balanced-Renderer



If no post-processing, remove:



# Setup



Assets -> Settings  
URP-Balanced

The screenshot shows the Unity Editor interface with the Project and Inspector tabs selected. In the Project tab, under Favorites, there are entries for SampleSceneProfile, URP-Balanced-Renderer, and URP-Balanced. Under Assets, the Settings folder is currently selected. The Inspector tab displays the URP-Balanced (Universal Render Pipeline Asset) settings. The Rendering section includes a Renderer List with one item: URP-Balanced-Renderer (Universal Renderer Data). The Quality section includes settings for HDR (4x), Anti Aliasing (MSAA), Render Scale (1), and Upscaling Filter (Automatic).

Project

Favorites

- All Materials
- All Models
- All Prefabs

Assets

- Oculus
- Plugins
- Resources
- Scenes
- Settings
- XR

Packages

Assets > Settings

- SampleSceneProfile
- URP-Balanced-Renderer
- URP-Balanced

URP-Balanced (Universal Render Pipeline Asset)

Rendering

Renderer List

- URP-Balanced-Renderer (Universal Renderer Data)

Quality

- HDR (4x)
- Anti Aliasing (MSAA)
- Render Scale (1)
- Upscaling Filter (Automatic)

# Setup



Project Settings - Graphics  
Assets -> Settings

The screenshot shows the Unity Project Settings window. The left sidebar lists various settings like Adaptive Performance, Audio, Burst AOT Settings, Editor, and Graphics. Under Graphics, there's a dropdown menu currently set to "None (Render Pipeline Asset)". The right panel displays the "Graphics" settings, which include "Scriptable Render Pipeline Settings" (set to "None"), a warning about a scriptable render pipeline being used, "Built-in Shader Settings", and a list of "Always Included Shaders" (Element 0 through Element 6). In the bottom-left corner, the Assets browser shows three assets: "SampleSceneProfile", "URP-Balanced-Renderer", and "URP-Balanced". The "URP-Balanced" asset is highlighted with a red box and has a red arrow pointing from it towards the "None (Render Pipeline Asset)" dropdown.

# Setup



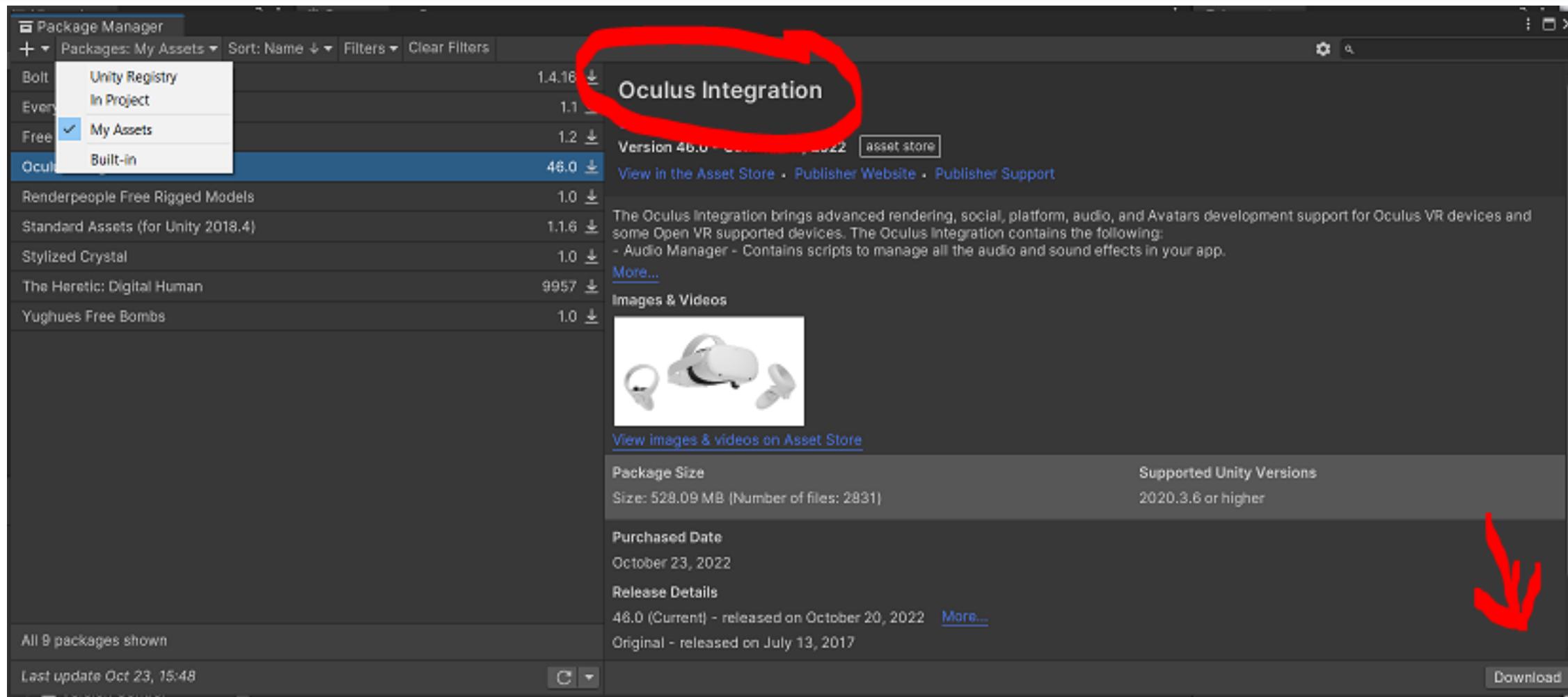
assetstore.unity.com  
Oculus Integration Package

The screenshot shows the Unity Asset Store page for the "Oculus Integration" package. At the top left, there's a message "✓ Added to My Assets". The package icon is an Oculus logo, and its name is "Oculus Integration". There are two blue buttons: "Open in Unity" and "Go to My Assets". A navigation bar below includes categories: 3D, 2D, Add-Ons, Audio, Essentials, Templates, Tools, VFX, Sale, and a "Sell Assets" button. Below the navigation bar are three status indicators: "Over 11,000 five-star assets", "Rated by 85,000+ customers", and "Supported by 100,000+ forum members". The main content area features a large image of an Oculus VR headset. To the right, the package title "Oculus Integration" is displayed with a "FREE" label, the developer "Oculus", and a rating of "★★★★☆ (647) | ❤️ (6700)". A red circle highlights the "Add to My Assets" button. Below this are sections for "License agreement", "File size" (528.1 MB), "Latest version" (46.0), and "Latest release date" (Oct 20, 2022).

# Setup

GEM2022

Window -> Package Manager



# Setup



Window -> Package Manager

Package Manager

Packages: My Assets ▾ Sort: Name ▾ Filters ▾ Clear Filters

Bolt	1.4.16	⤵
Everyday Motion Pack Free	1.1	⤵
Free SpeedTrees Package	1.2	⤵
Oculus Integration	46.0	⤵
Renderpeople Free Rigged Models	1.0	⤵
Standard Assets (for Unity 2018.4)	1.1.6	⤵
Stylized Crystal	1.0	⤵
The Heretic: Digital Human	9957	⤵
Yughues Free Bombs	1.0	⤵

**Oculus Integration**

Oculus  
Version 46.0 - October 20, 2022 [asset store]

[View in the Asset Store](#) • [Publisher Website](#) • [Publisher Support](#)

The Oculus Integration brings advanced rendering, social, platform, audio, and Avatars development support for Oculus VR devices and some Open VR supported devices. The Oculus Integration contains the following:

- Audio Manager - Contains scripts to manage all the audio and sound effects in your app.

[More...](#)

Images & Videos



[View images & videos on Asset Store](#)

Package Size  
Size: 528.09 MB (Number of files: 2831)

Supported Unity Versions  
2020.3.6 or higher

Purchased Date  
October 23, 2022

Release Details  
46.0 (Current) - released on October 20, 2022 [More...]  
Original - released on July 13, 2017

All 9 packages shown

Last update Oct 23, 15:48

C ⤵

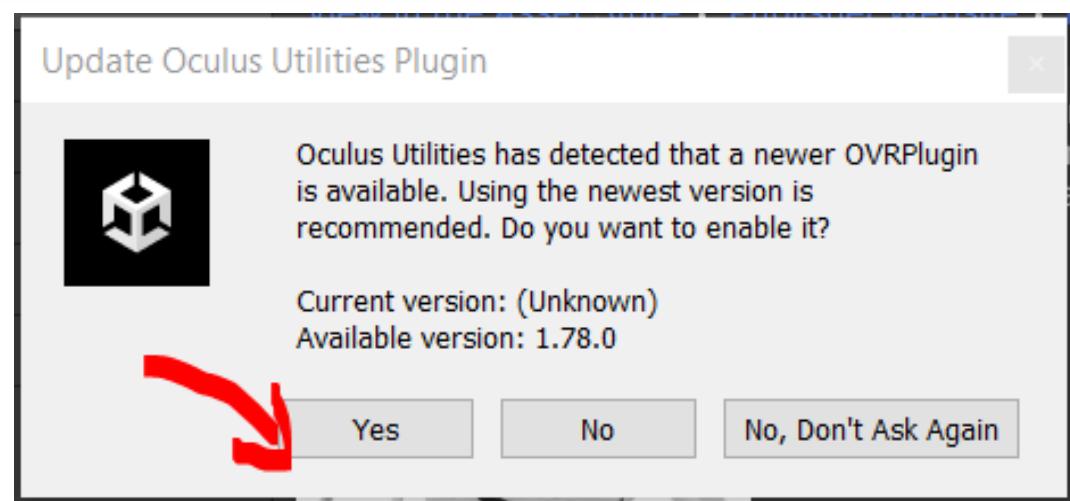
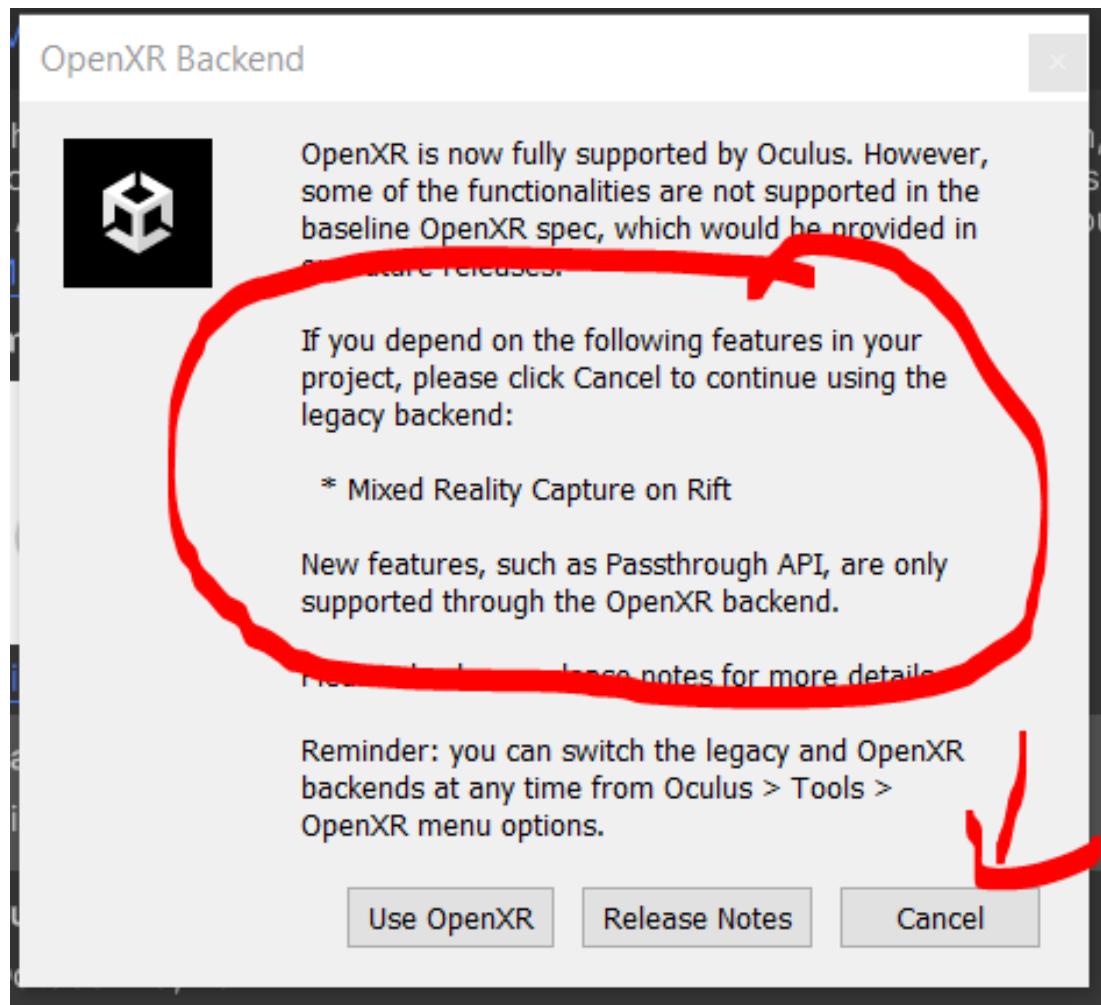
Import Re-Download



# Setup

GEM2022

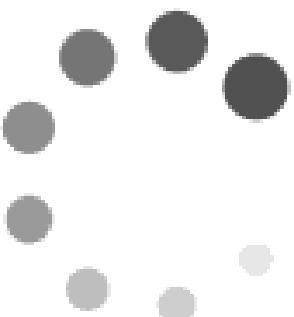
Window -> Package Manager



# Setup

GEM2022

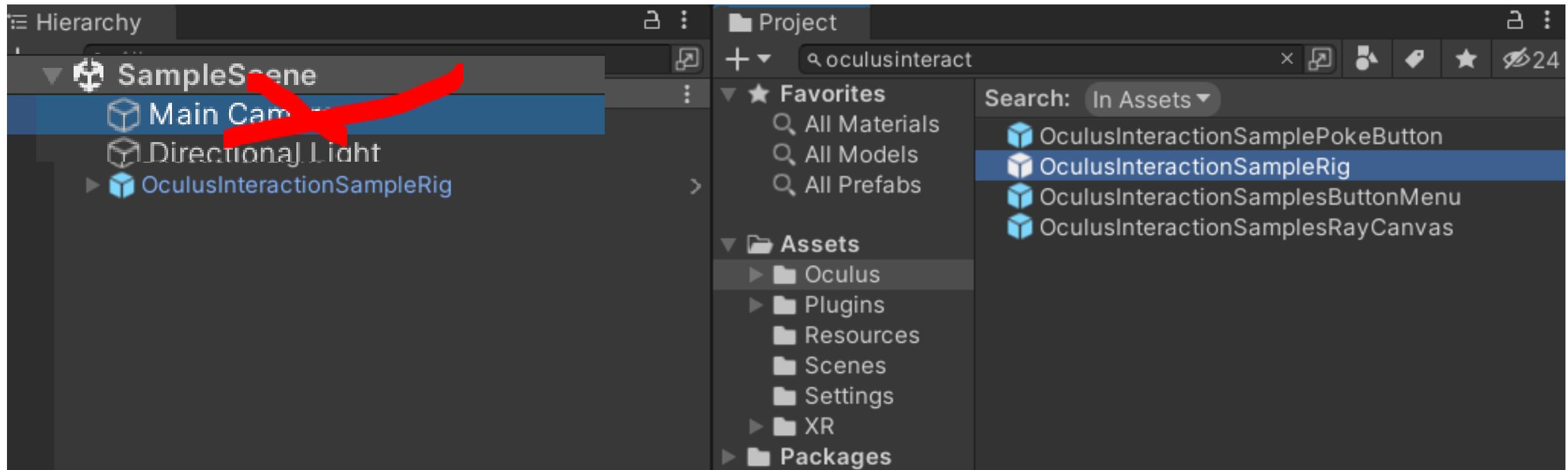
Questions?



# Setup

GEM2022

Project -> search:  
OculusInteraction



# Setup



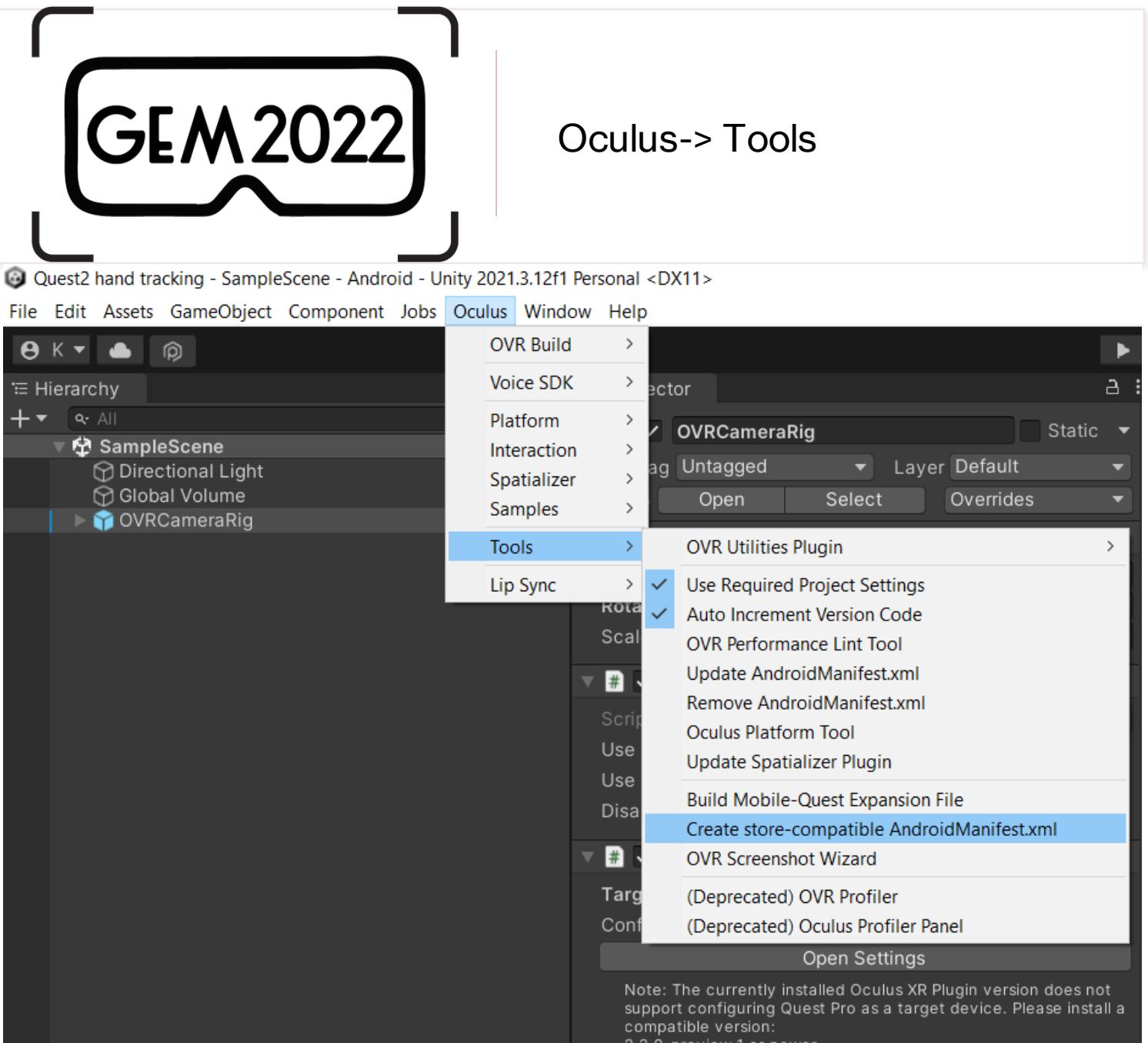
OculusInteractionSampleRig ->  
OVRCameraRig  
OVR Manager (Script)

The image shows the Unity Editor interface with two open windows: Hierarchy and Inspector.

**Hierarchy Window:** Shows the scene structure:

- SampleScene
  - Main Camera
  - Directional Light
  - OculusInteractionSampleRig
    - OVRCameraRig
    - InputOVR
      - Hmd
      - Controllers
        - LeftController
        - RightController
      - Hands
        - LeftHand
          - HandDataSource
          - LeftHandVisual
          - HandFeatures
          - HandInteractorsLeft
        - RightHand

# Setup



Oculus-> Tools

# Setup



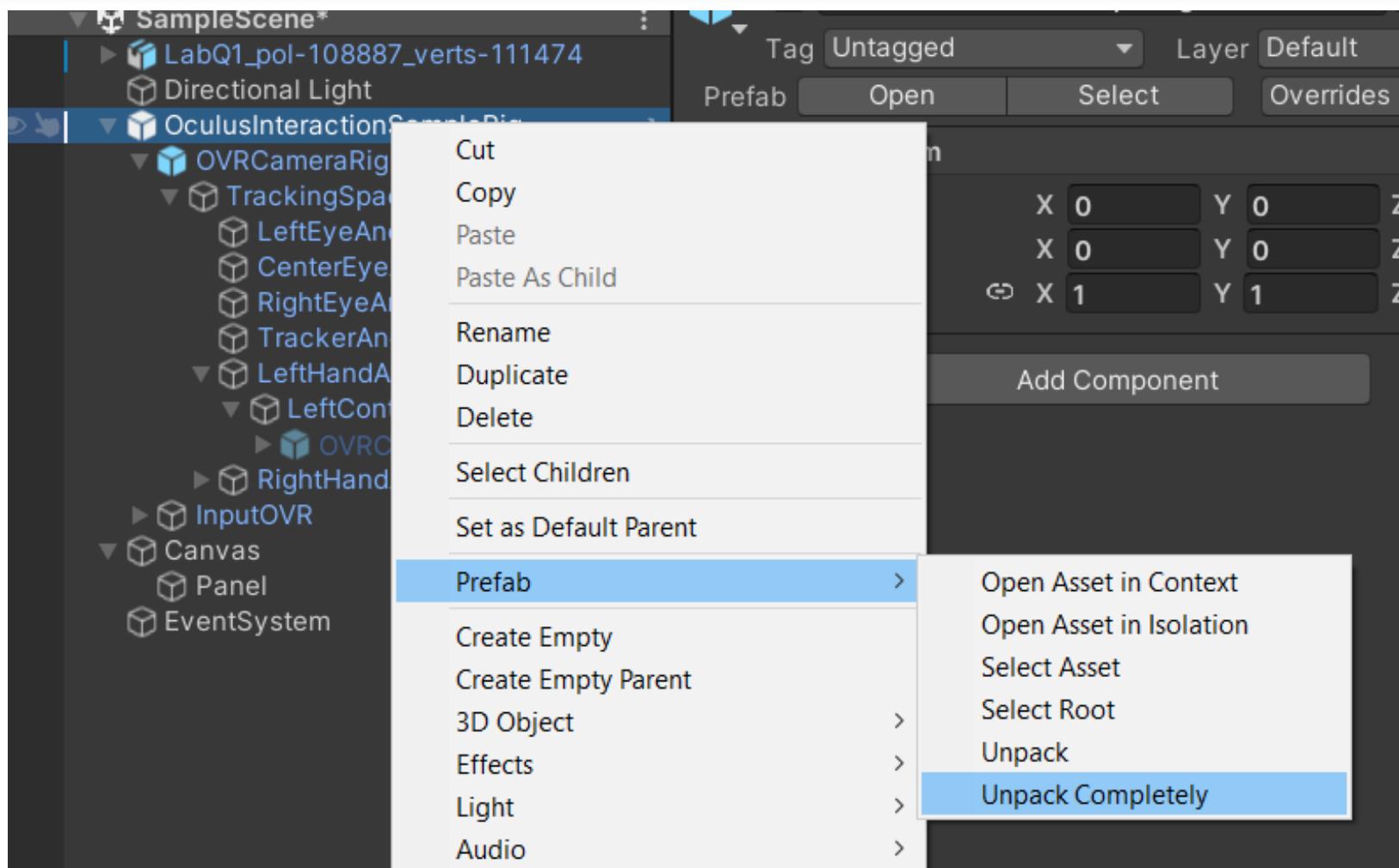
Oculus-> Tools

A screenshot of the Unity Asset Manager. On the left, there's a sidebar with sections like Favorites, Assets, Plugins, Resources, Scenes, Settings, XR, and Packages. The Plugins section is expanded, showing an Android folder which is also expanded, revealing an AndroidManifest file. The main panel shows the XML code for the AndroidManifest file. The code includes standard manifest tags like application, activity, intent-filter, category, and meta-data, along with specific ones for Unity and Oculus integration, such as android.hardware.vr.headtracking, oculus.software.handtracking, and com.oculus.permission.HAND\_TRACKING.

# Setup

GEM2022

Prefab -> Unpack Completely



# Setup



OculusHand  
Material

The screenshot shows the Unity Editor interface with the Project and Inspector tabs selected. In the Project tab, the Assets > Oculus > Interaction folder is open, displaying various Unity assets like DistanceLine, DistanceReticle, and OculusHand. The OculusHand asset is currently selected, which is also reflected in the Inspector tab.

**Inspector Tab (Oculus Hand (Material)):**

- Shader:** Interaction/OculusHand
- General:**
  - Color Top: Orange color swatch
  - Color Bottom: Red color swatch
  - Finger Glow Color: Yellow color swatch
- Opacity:** A slider set to 1.0
- Fresnel:**
  - Fresnel Power: Set to 1.75
- Outline:**
  - Outline Color: Red color swatch
  - Outline Joint Error Color: Yellow color swatch
  - Outline Width: Set to 0.00093
  - Outline Opacity: Set to 1.0
- Wrist:**
  - Wrist Fade: Set to 0.0
- Finger Glow:**

# Setup



Basic Hand Material

Quick fix for legacy shader

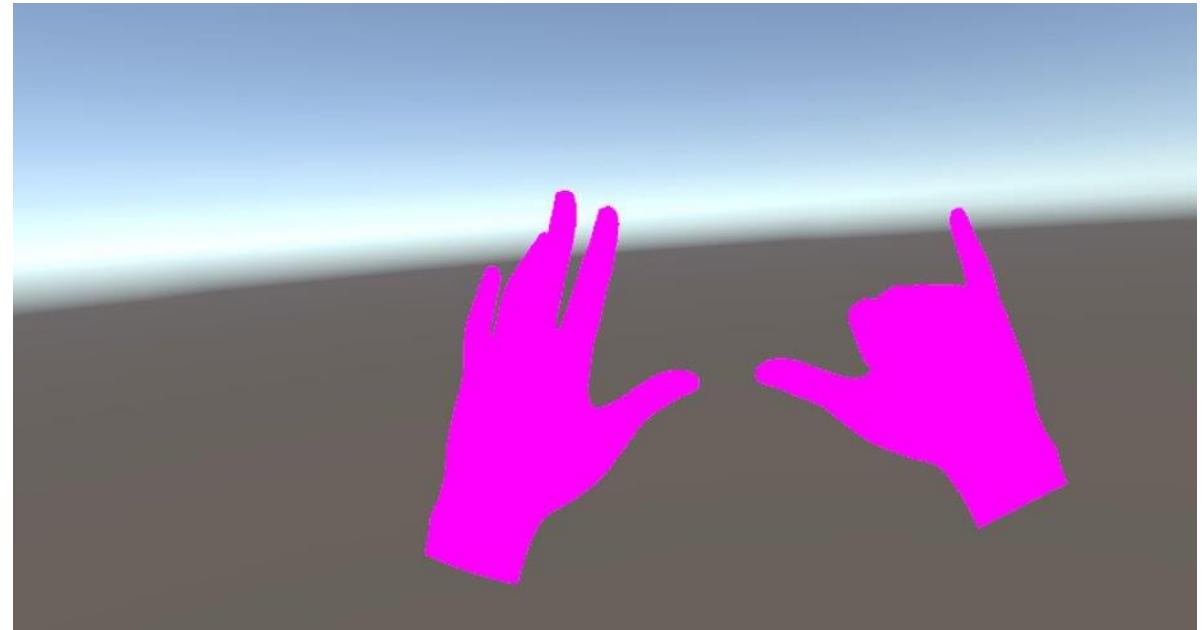
OVRHandPrefab

Material change to:

Universal Render Pipeline -> Complex lit

Base Map color -> Change (to match the skin color)

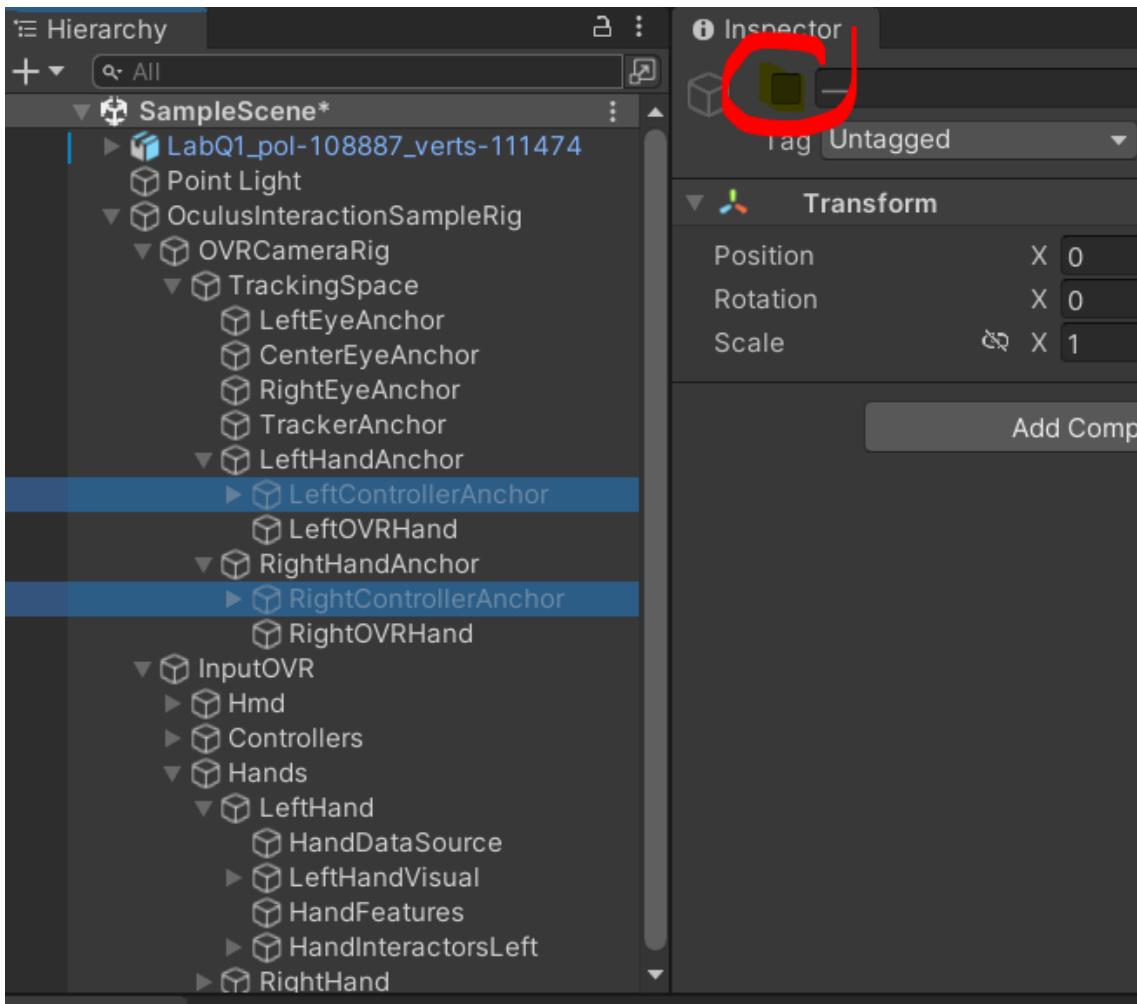
Smoothness -> 0



# Setup

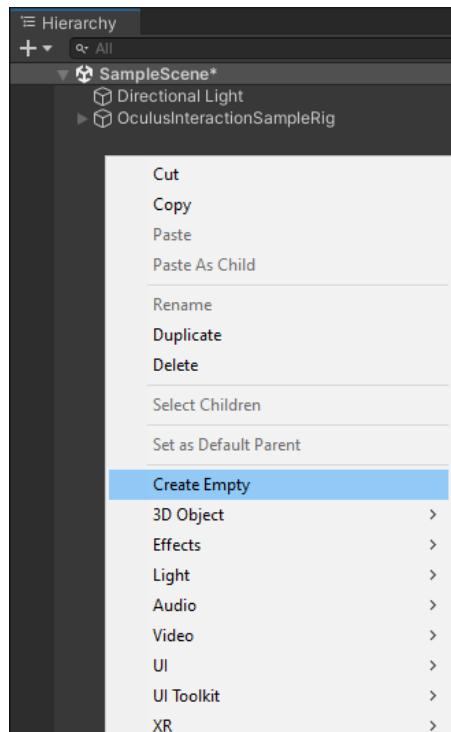
GEM2022

Disabling controllers' anchors

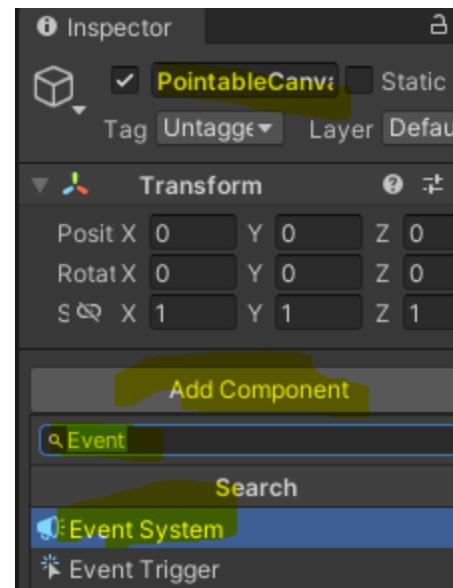


# Setup

1) Hierarchy (RMB)  
Create Empty  
rename:  
PointableCanvasModule

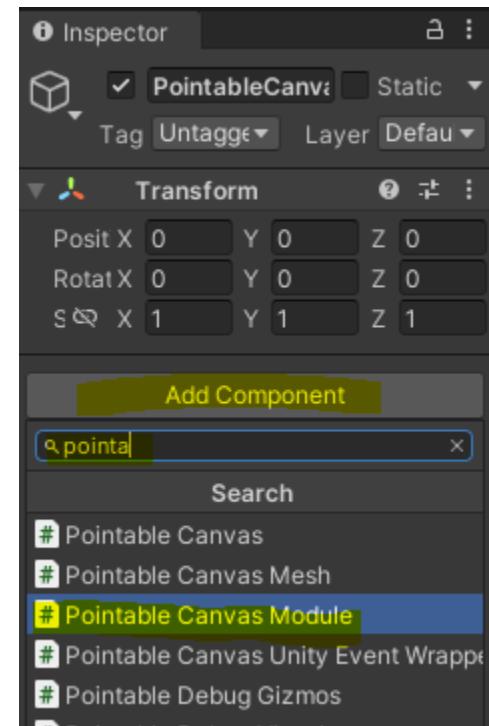


2) Add Component:  
Event System



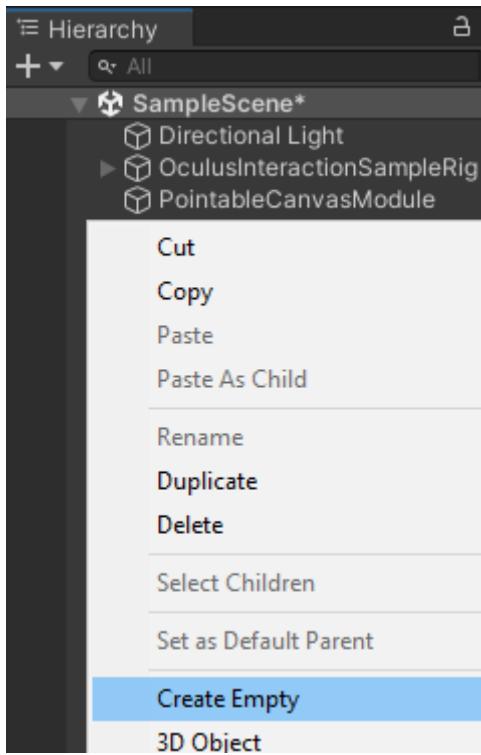
PointableCanvasModule

3) Add Component:  
Pointable Canvas Module (script)

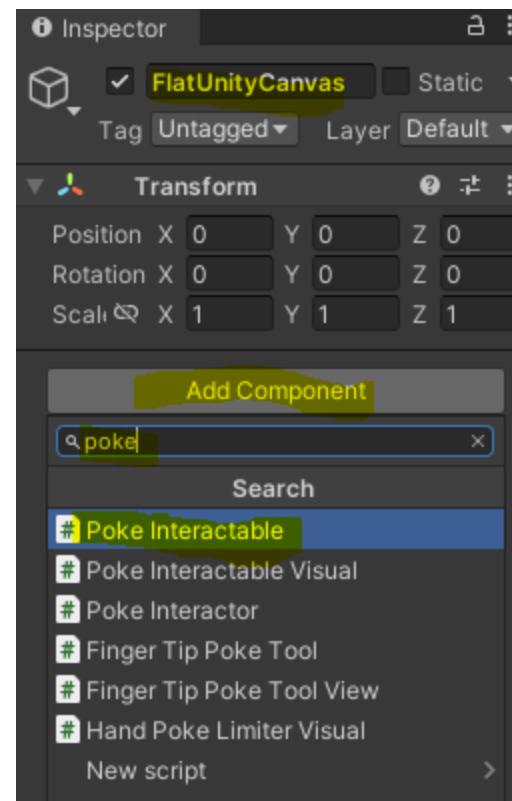


# Setup

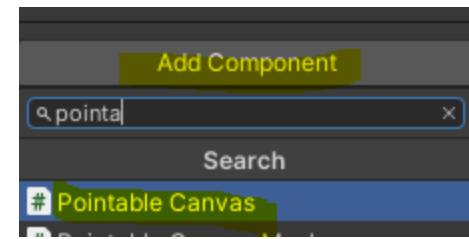
1) Hierarchy (RMB)  
Create Empty  
rename:  
**FlatUnityCanvas**



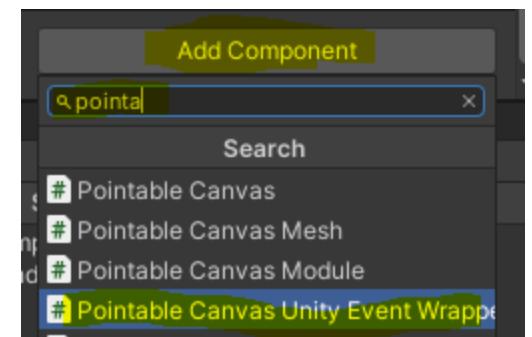
2) Add Component:  
**Poke Interactable**



3) Add Component:  
**Pointable Canvas**  
(script)



4) Add Component:  
**Pointable Canvas**  
**Unity Event**  
**Wrapper**  
(script)



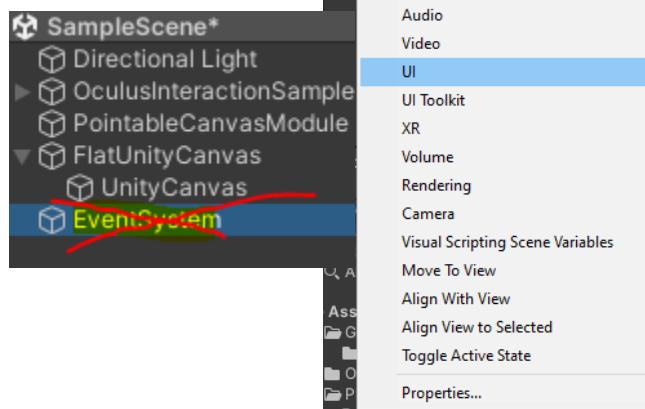
**FlatUnityCanvas**



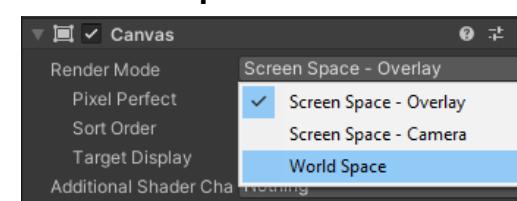
# Setup

1) Hierarchy ->  
FlatUnityCanvas (RMB)  
UI -> Canvas  
rename:  
UnityCanvas

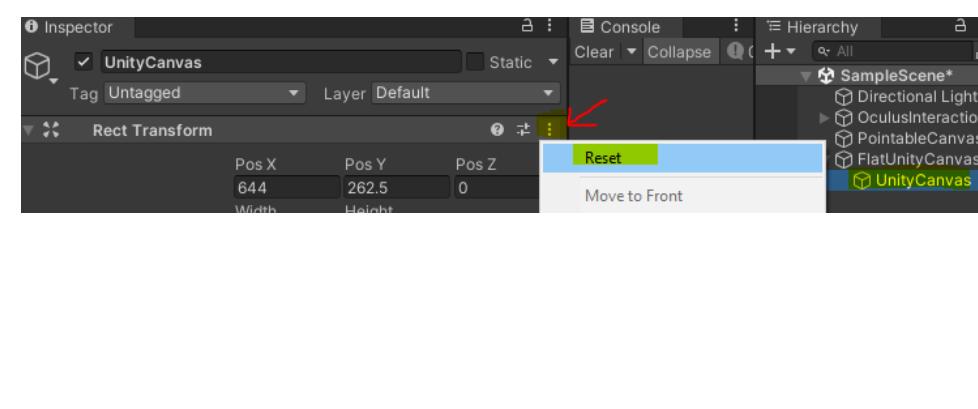
1.5) Delete:  
EventSystem



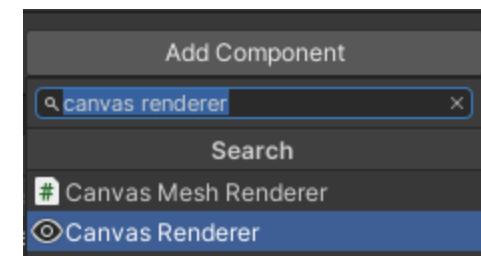
2) In UnityCanvas  
Canvas  
Change Render Mode:  
World Space



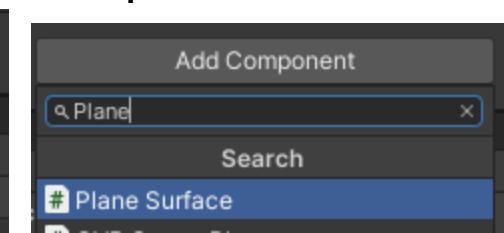
\*) Reset position



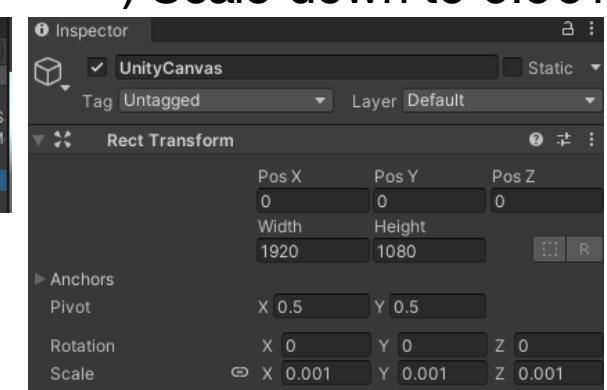
3) Add Component:  
Canvas Renderer



4) Add Component:  
Plane Surface  
(script)



\*\*) Resize:  
Width 1920 Height 1080  
\*\*\*) Scale down to 0.001

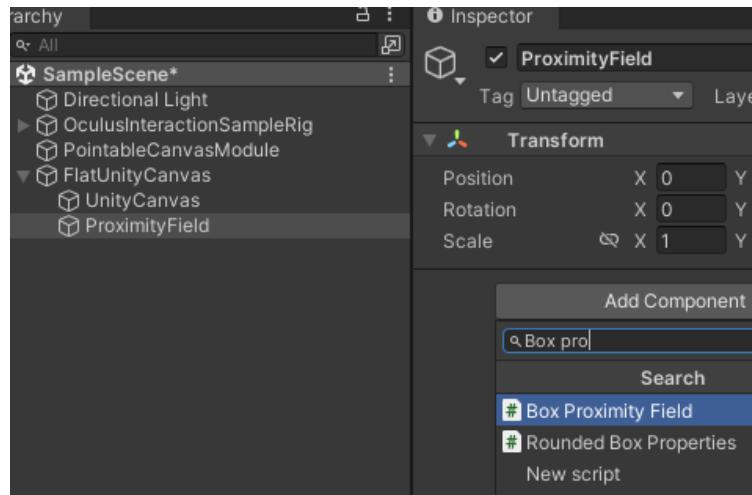
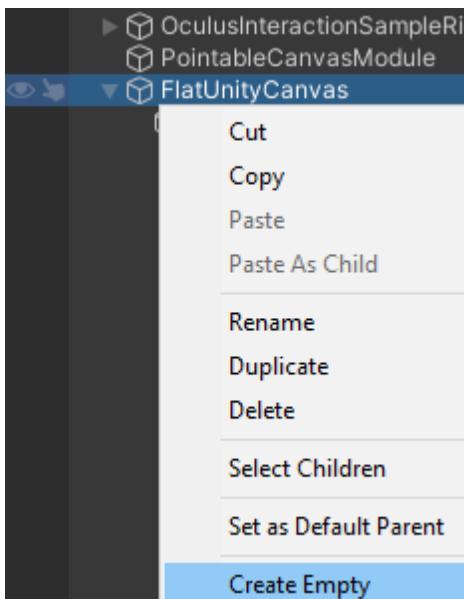


UnityCanvas

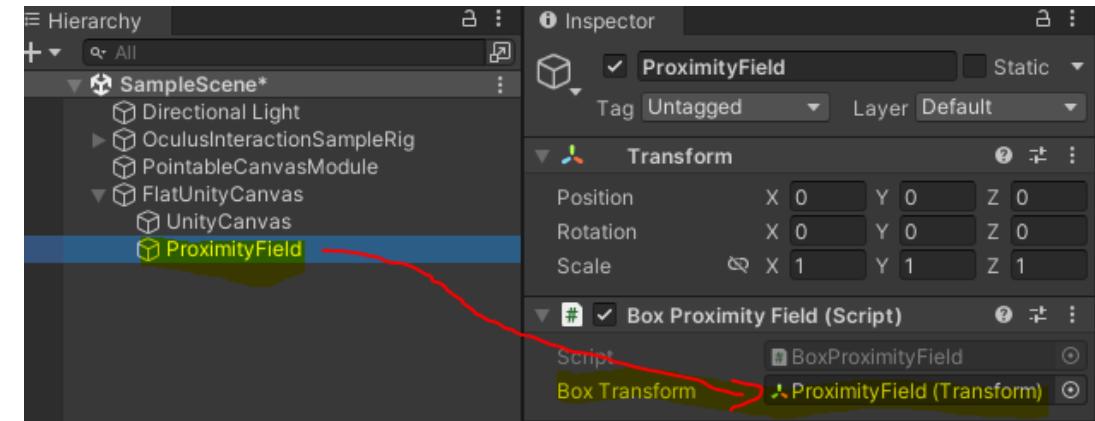


# Setup

1) Hierarchy ->  
FlatUnityCanvas (RMB)  
Create empty  
  
rename:  
ProximityField



2) Add Component:  
Box Proximity Field  
(script)



ProximityField

3) Drag and drop  
ProximityField(GO)  
to  
Box Transform  
in  
Box Proximity Field

# Setup



FlatUnityCanvas

Drag and drop:

in Poke Interactable:

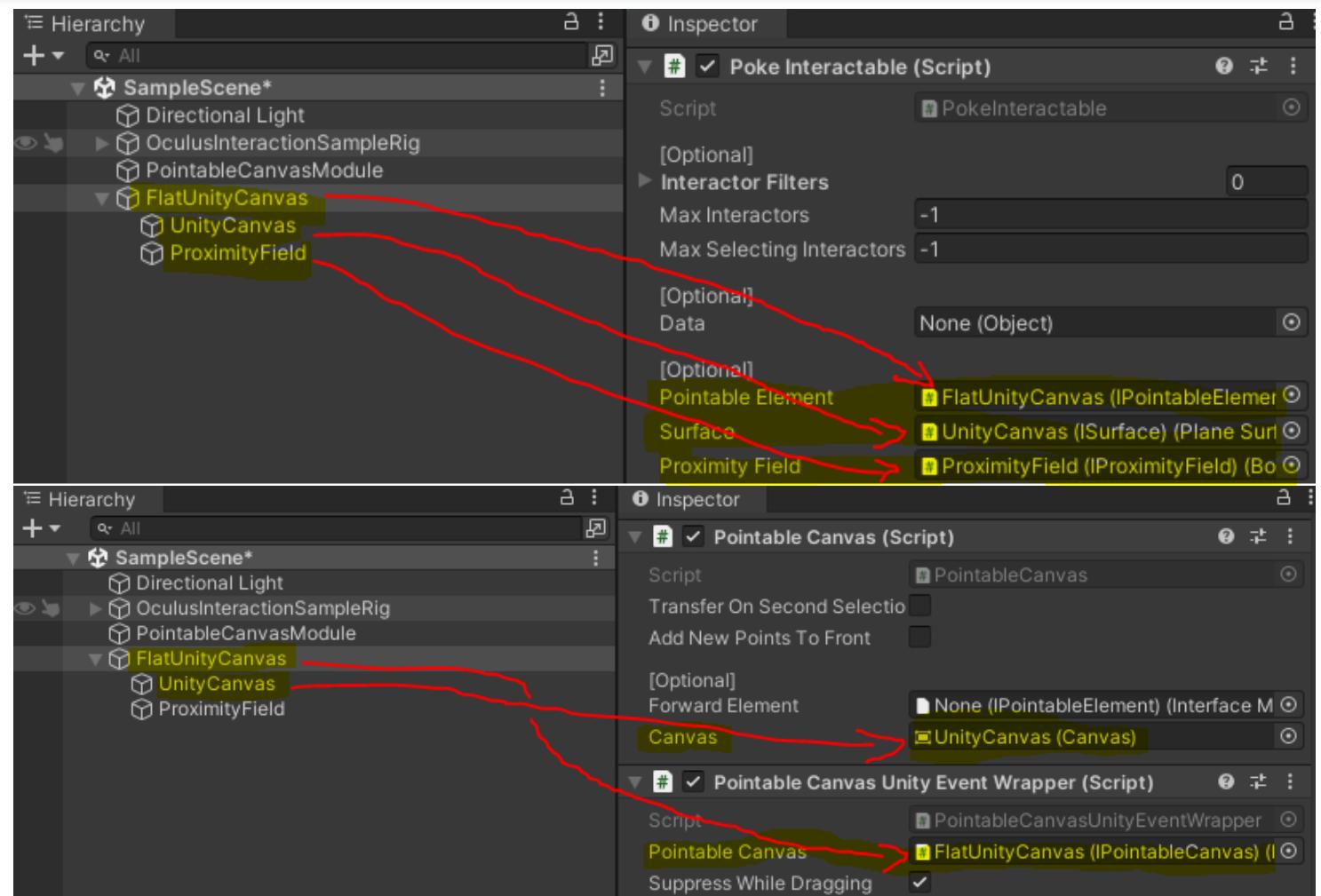
- FlatUnityCanvas (GO) to Pointable Element
- UnityCanvas (GO) to Surface
- ProximityField (GO) to Proximity Field

in Pointable Canvas:

- UnityCanvas (GO) to Canvas

in Pointable Canvas Unity Event Wrapper

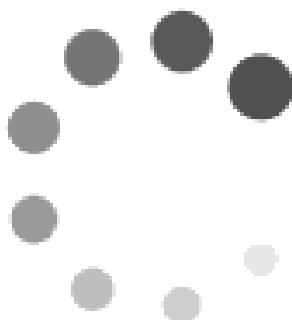
- FlatUnityCanvas (GO) to Pointable Canvas



# Setup



Questions?

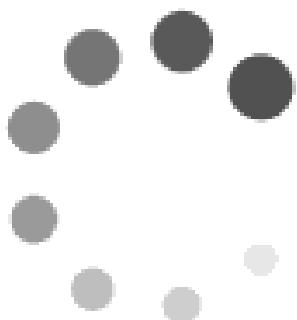


>>> if needed open Scenes -> **Step1\_setupComplete** scene

# Setup



Status



>>> Congratulations!

# Setup

GEM2022

Status: **COMPLETE!**

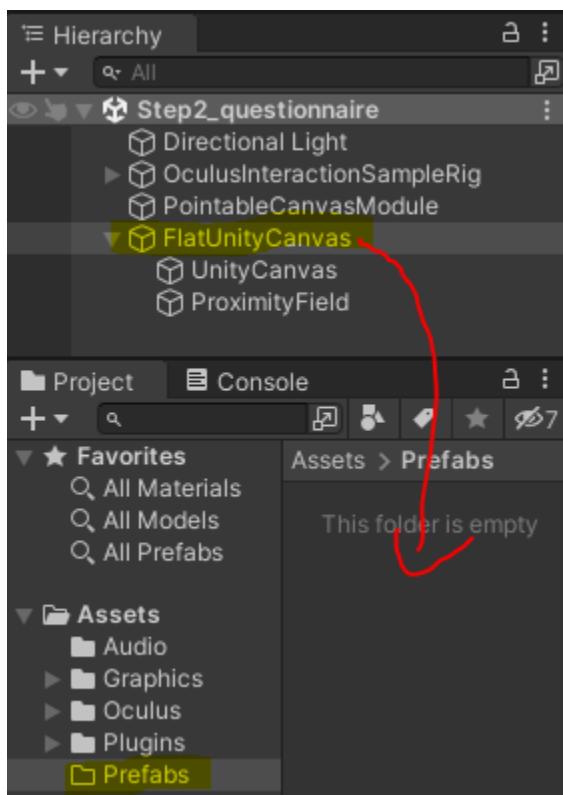


>>> Setup complete!

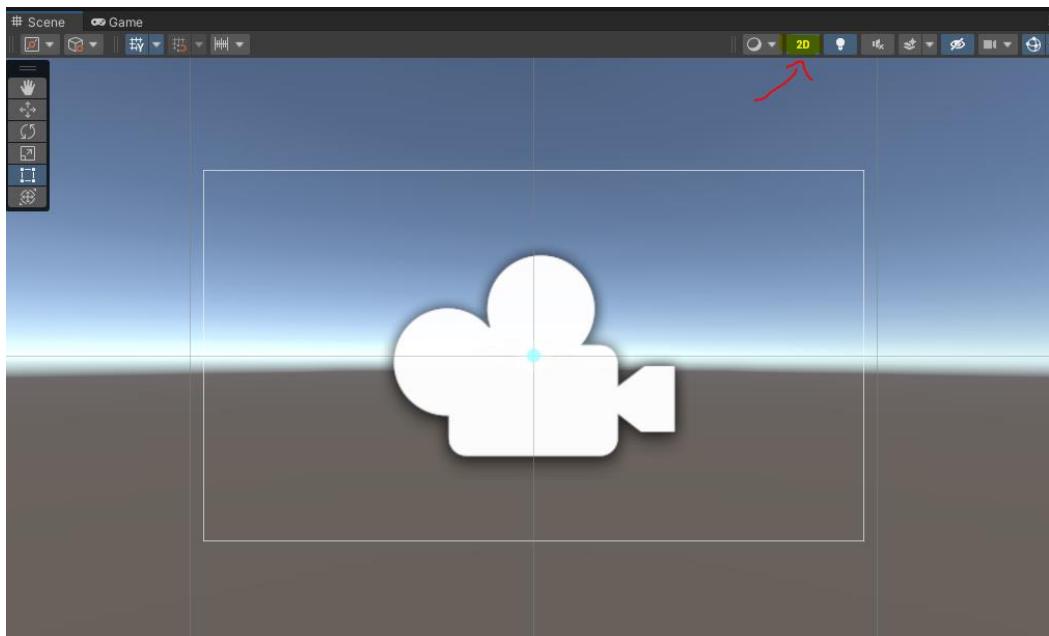
# Questionnaire

GEM2022

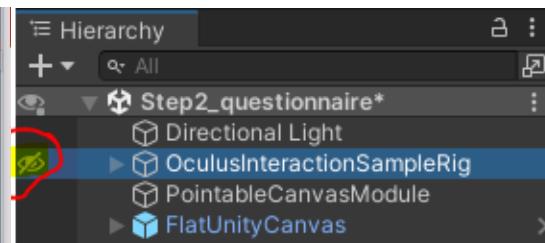
- 1) Project (RMB) Create -> Folder Rename Repeat:
  - Scenes
  - Prefabs
  - Scripts
- 2) File -> Save As -> whatever ... in Scenes folder



- 3) In Project -> Open prefabs Drag and drop FlatUnityCanvas from Hierarchy to that folder in editor



- 4) In the Scene window click on 2D Next select FlatUnityCanvas in Hierarchy, press F



Adding folders:  
Scenes & Prefabs & Scripts

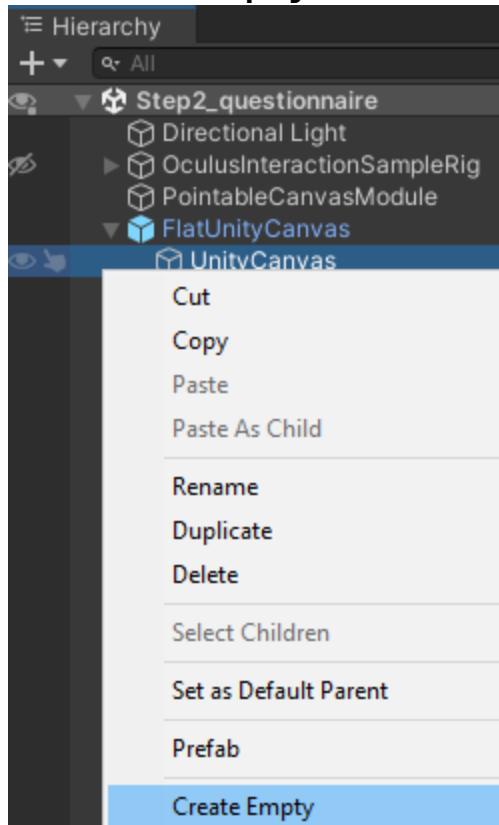
- 5) In Hierarchy click on the eye icon beside OculusInteraction SampleRig game object

# Questionnaire

GEM2022

Adding the first button

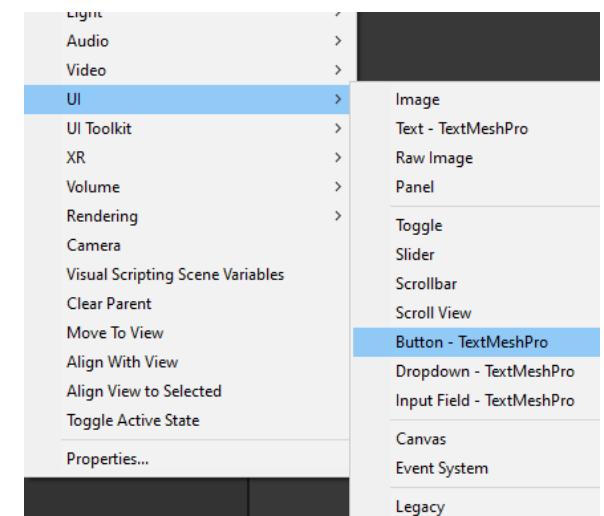
1) Hierarchy (RMB) on UnityCanvas (GO) -> Create Empty (rename)



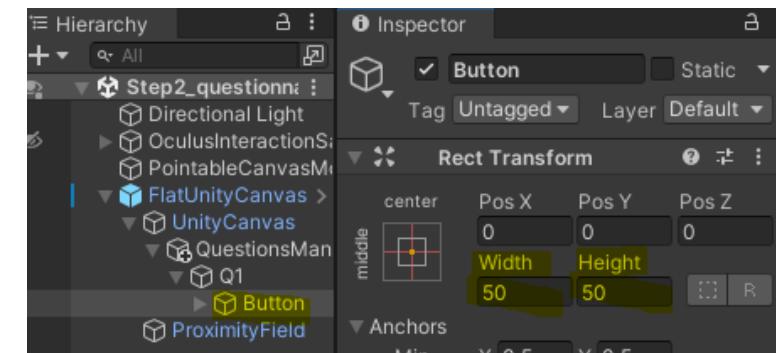
2) On that new object create another child (rename to Q1)



3) RMB on Q1 -> UI -> Button - TextMeshPro



4) In the Inspector panel when this Button is selected in Hierarchy change Width and Height to 50



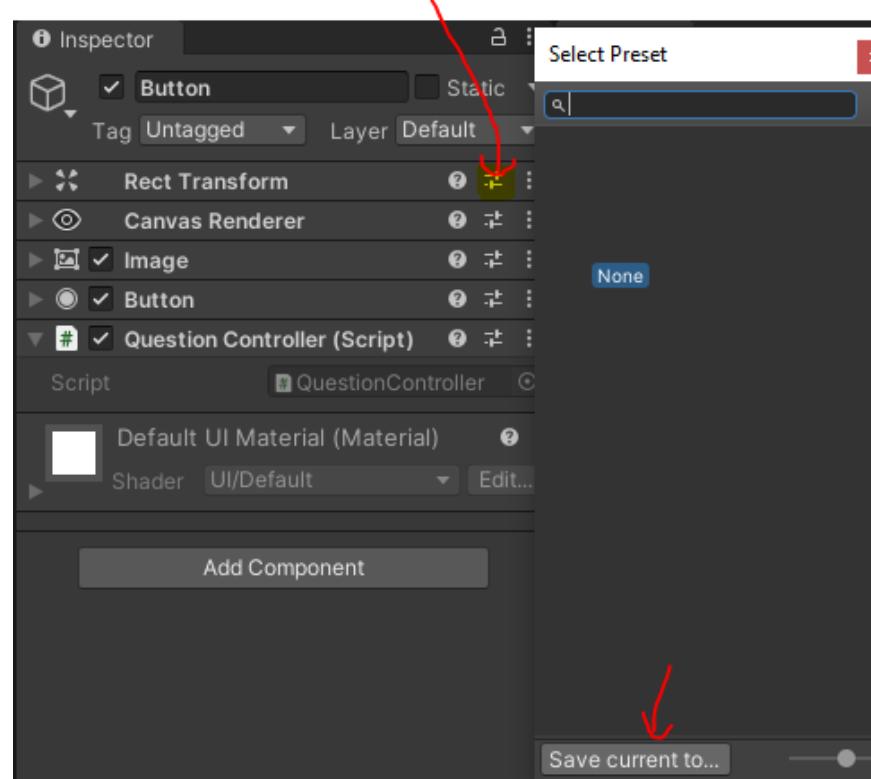
# Questionnaire



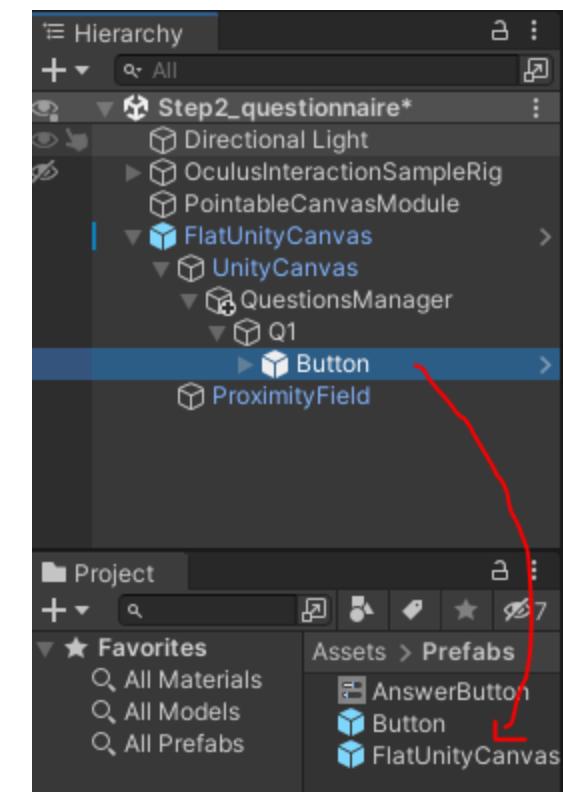
Create first preset

1) In the Inspector  
click the icon  
 on Rect Transform  
component

Save current to...  
AnswerButton



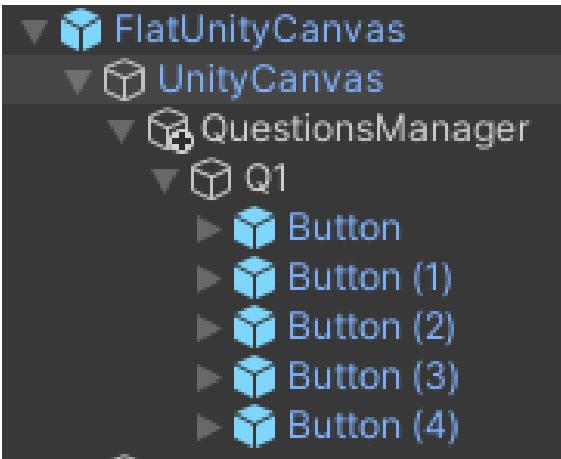
2) Drag and drop Button  
to Prefabs folder



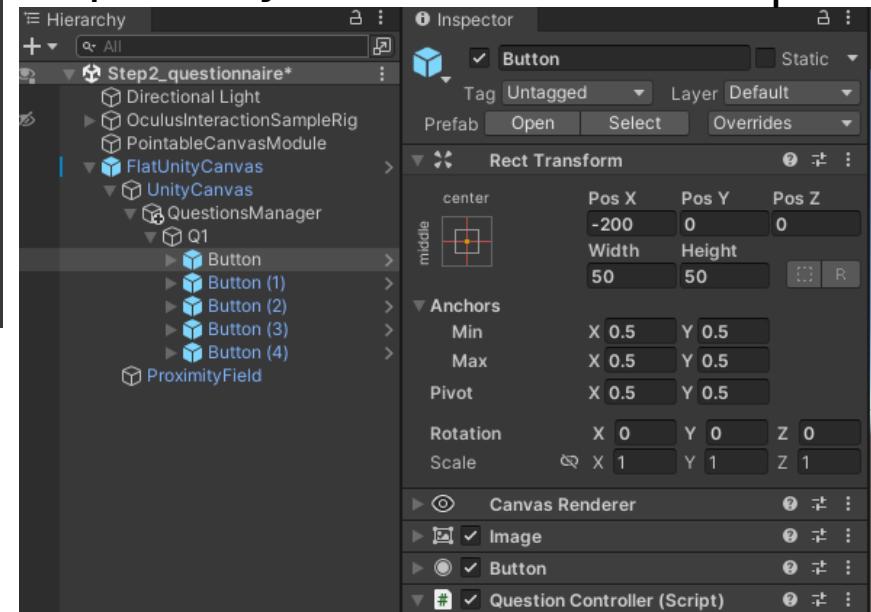
# Questionnaire

GEM2022

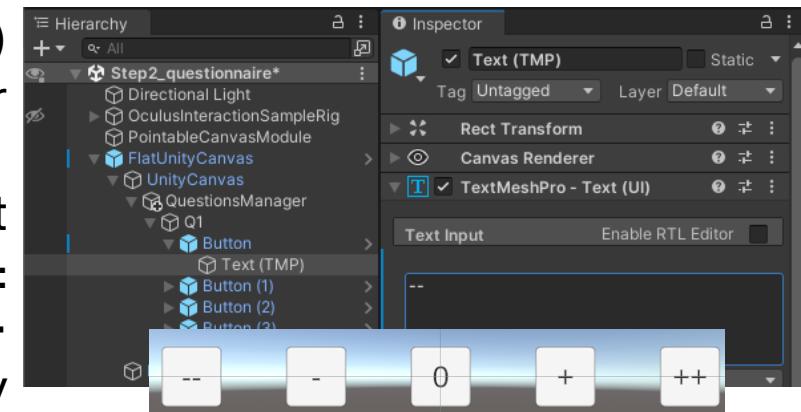
1) Drag and drop from  
Prefabs Button prefab  
onto Q1 game object  
(repeat 4 times)



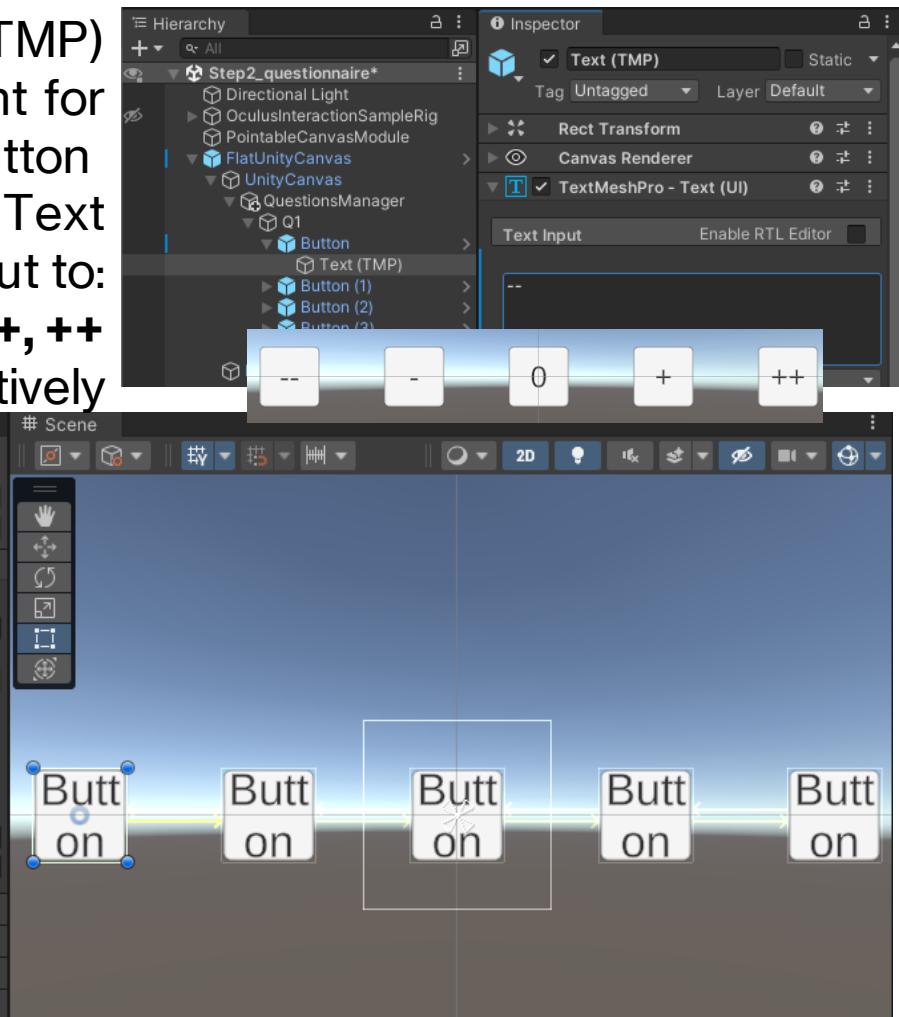
2) In Rect Transform  
component change  
Pos X to:  
**-200, -100, 0, 100, 200**  
respectively



3) In Text (TMP)  
component for  
each button  
change Text  
Input to:  
**--, -, 0, +, ++**  
respectively



Populating Q1 with buttons



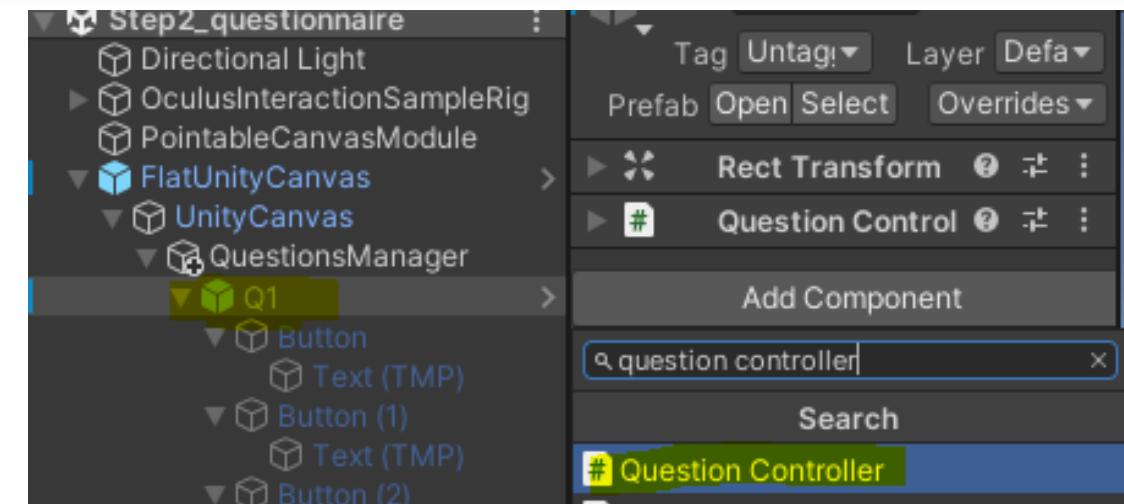
# Questionnaire

GEM2022

Create first script

1) In Scripts folder  
(RMB)  
Create -> C# Script  
name it:  
QuestionController

2) Select the Q1 in  
Hierarchy (child of  
QuestionsManager)  
- Scroll down to the  
button: Add Component  
in the Inspector panel  
and click on  
QuestionController



3) Double click on the QuestionController file in Scripts  
folder to open it

# Questionnaire

GEM2022

Create first script

Add the following code inside QuestionController file:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; //ADD to support Button class

// Unity Script (2 asset references) | 0 references
public class QuestionController : MonoBehaviour
{
    [SerializeField] List<Button> buttons; //Buttons in question holder
    public int QuestionID = 0; //Question number holder

    // References
    public void NewAnswer(int answer) //method for highlighting last pressed button
    {
        foreach (Button button in buttons)
        {
            button.GetComponent<Image>().color = Color.yellow;
        }
        buttons[answer - 1].GetComponent<Image>().color = Color.green;
    }
}
```

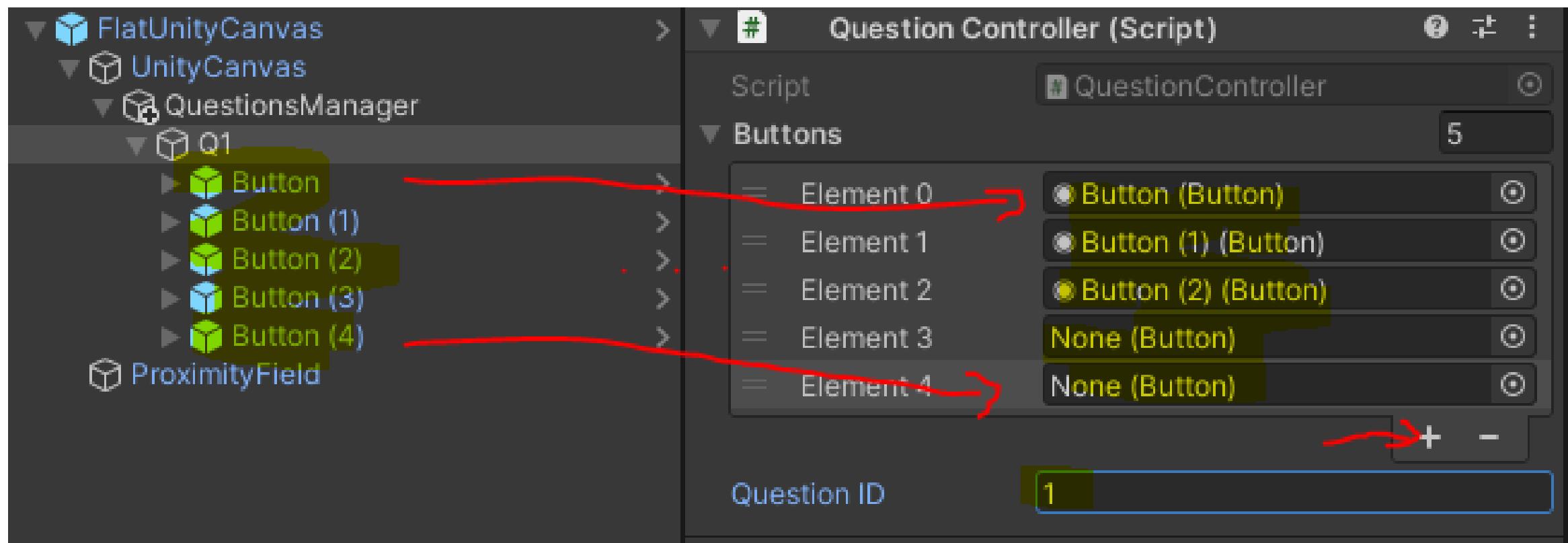
Save!.. save often... here and in Unity

# Questionnaire

GEM2022

Drag and dropping exercise...

Drag and drop each Button to Buttons list



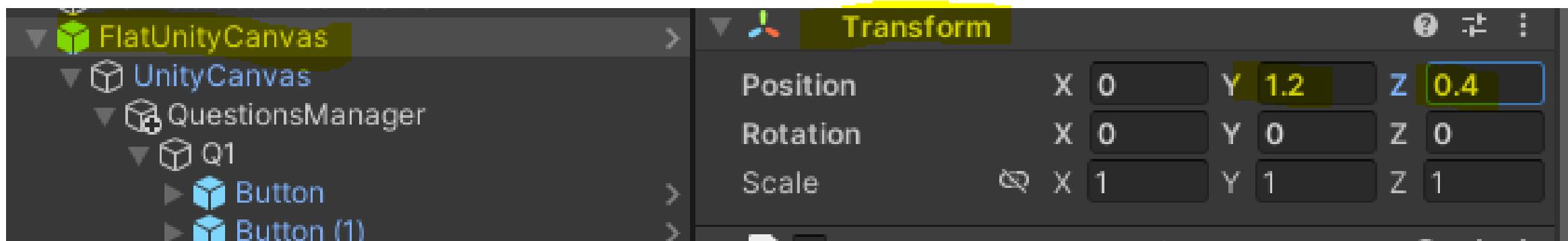
Change Question ID to 1

# Questionnaire

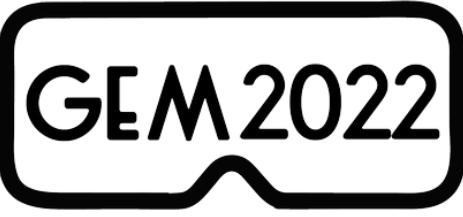


Move that canvas elsewhere!

Change the Transform Y and Z position of the FlatUnityCanvas (GO) to somewhere more user friendly

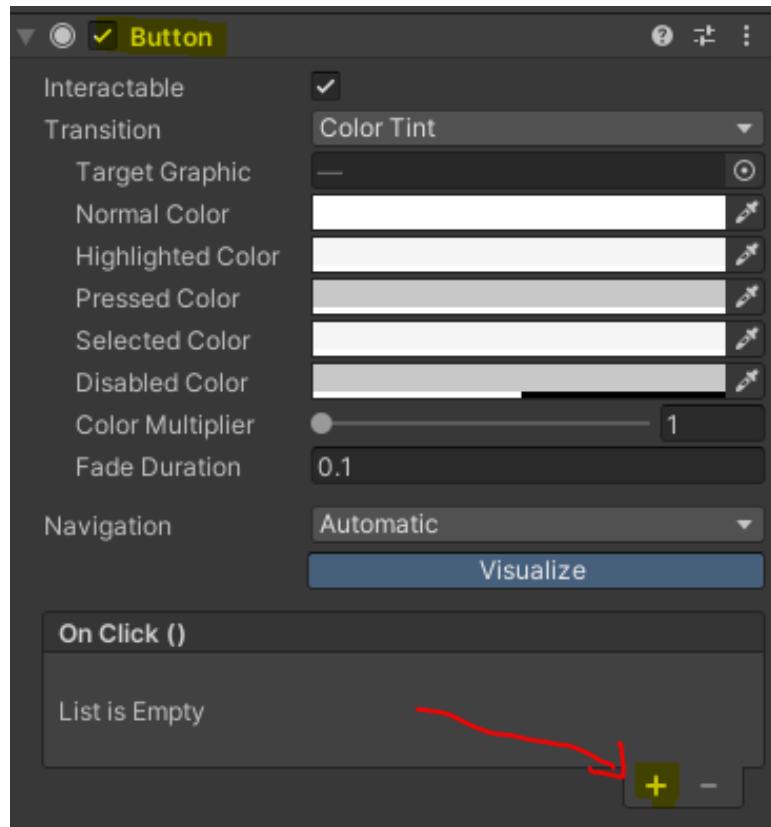


# Questionnaire

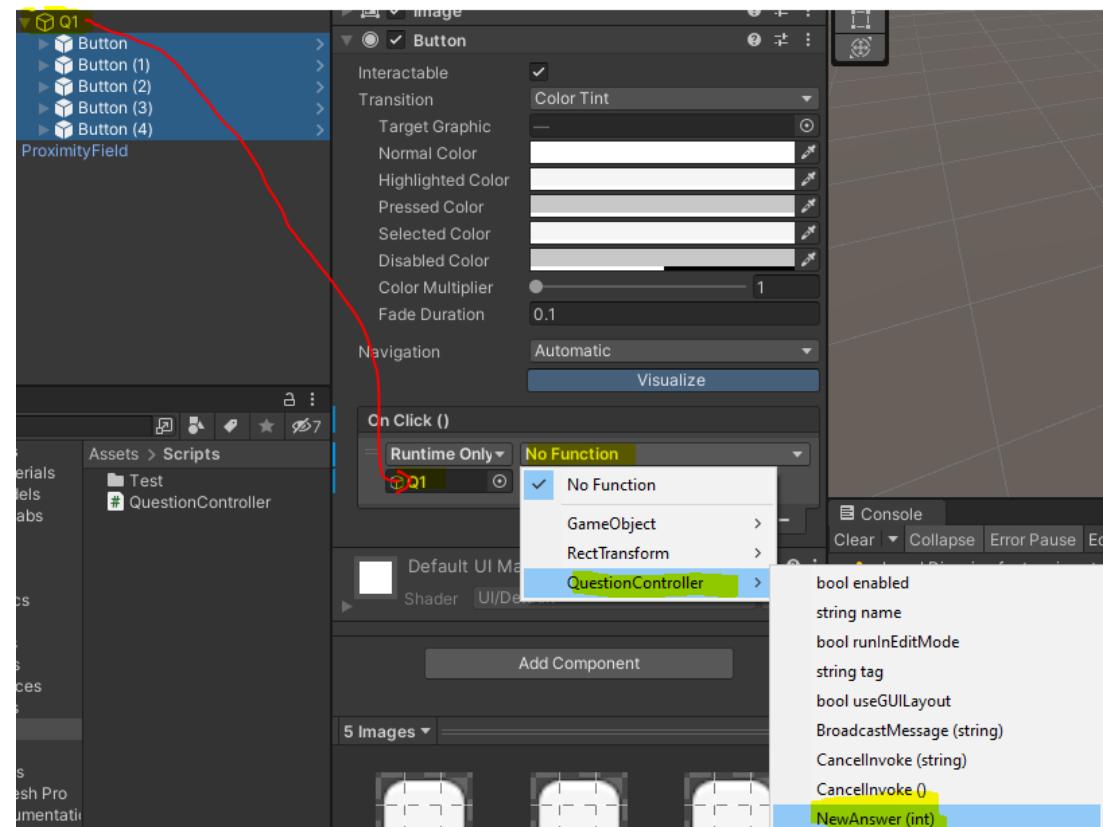


Lets make some action!

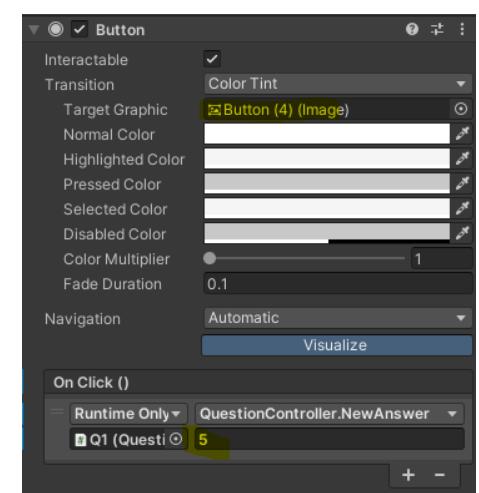
1) Select all the buttons, and  
in Button class click the +



2) Drag and drop the Q1 object



3) Select: QuestionController ->  
NewAnswer



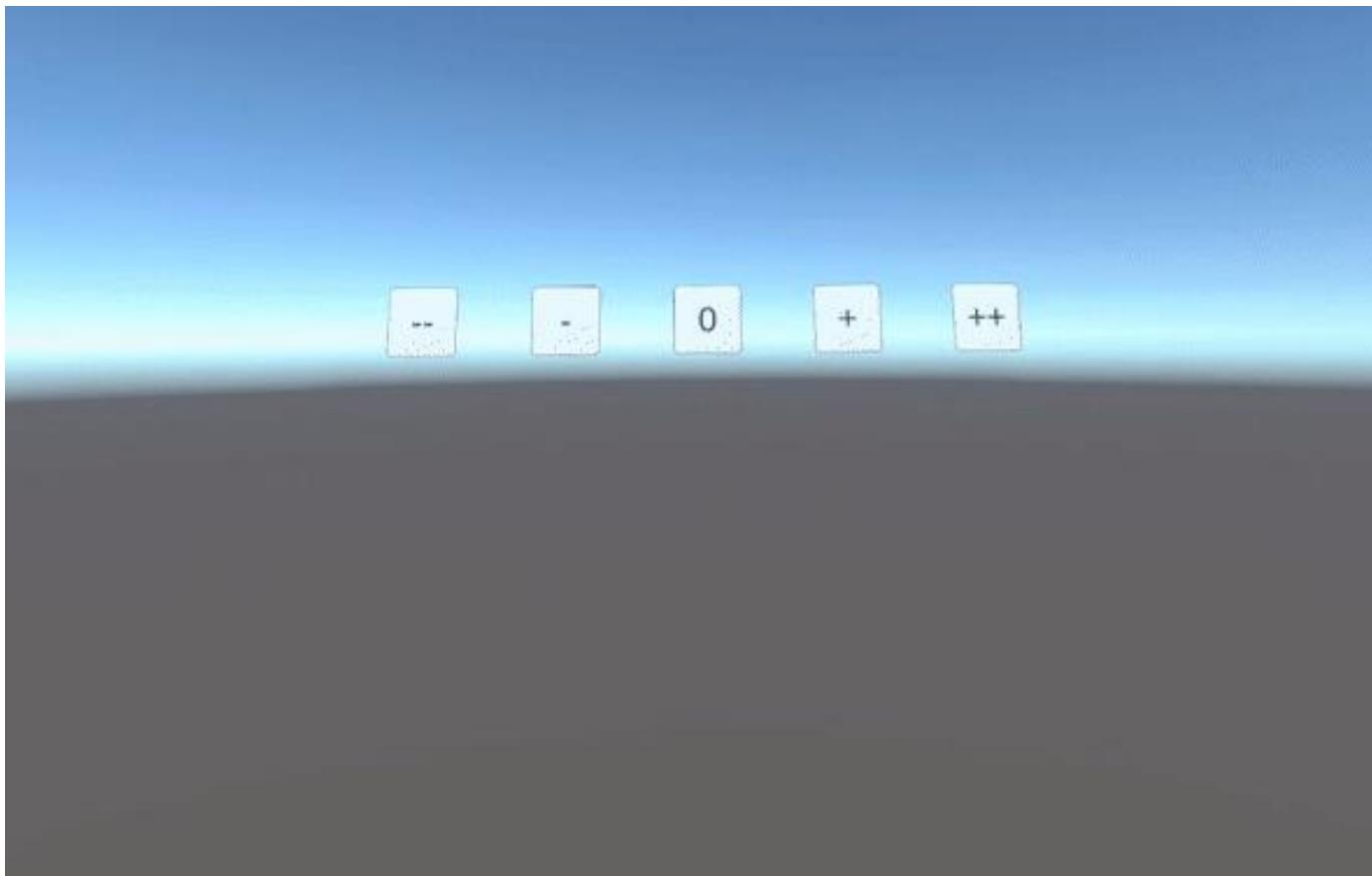
4) Change 0 to:  
1, 2, 3 ,4, 5  
respectively for  
each of these  
buttons

# Questionnaire

GEM2022

Status: **TESTING!**

Press play and see if it works!



# Questionnaire

GEM2022

Hard to answer lack of questions

1) Click (RMB) on Q1

-> UI

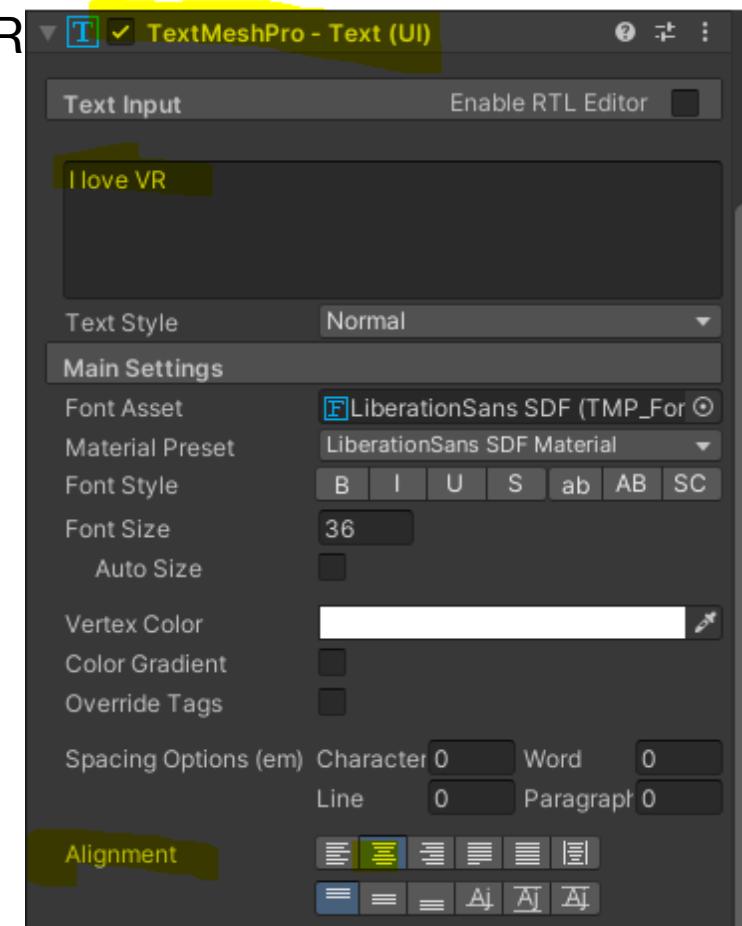
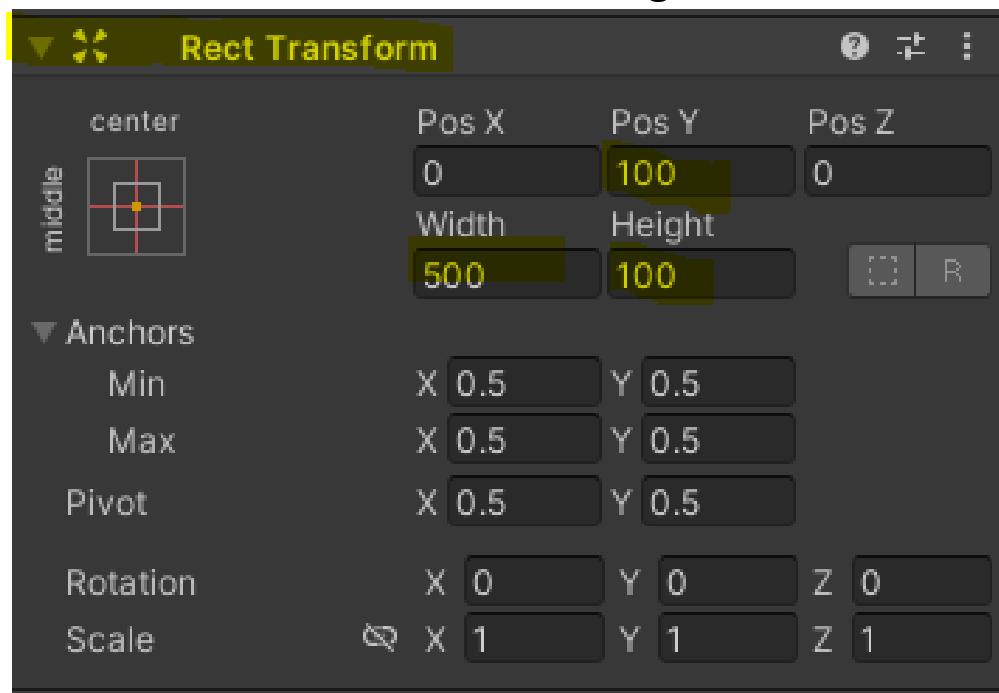
-> Text - TextMeshPro

2) In Rect Transform  
of that object change:

- Pos Y to 100
- Width to 500
- Height to 100

3) In TextMeshPro - Text (UI)

- Text input to : I love VR
- Alignment to: Center and middle



# Questionnaire

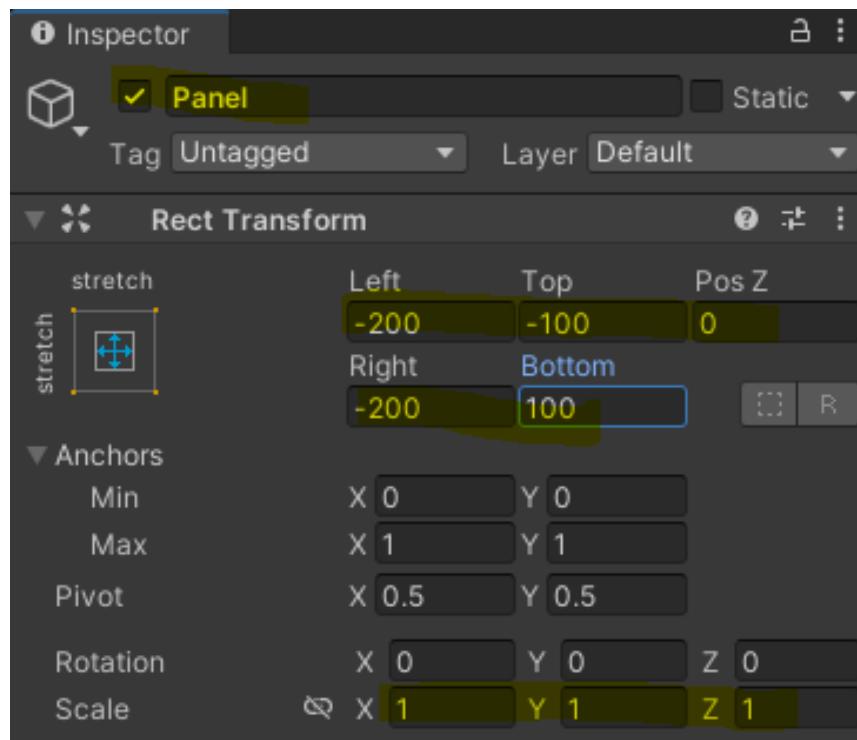
GEM2022

...and see them without background

1) Click (RMB) on Q1

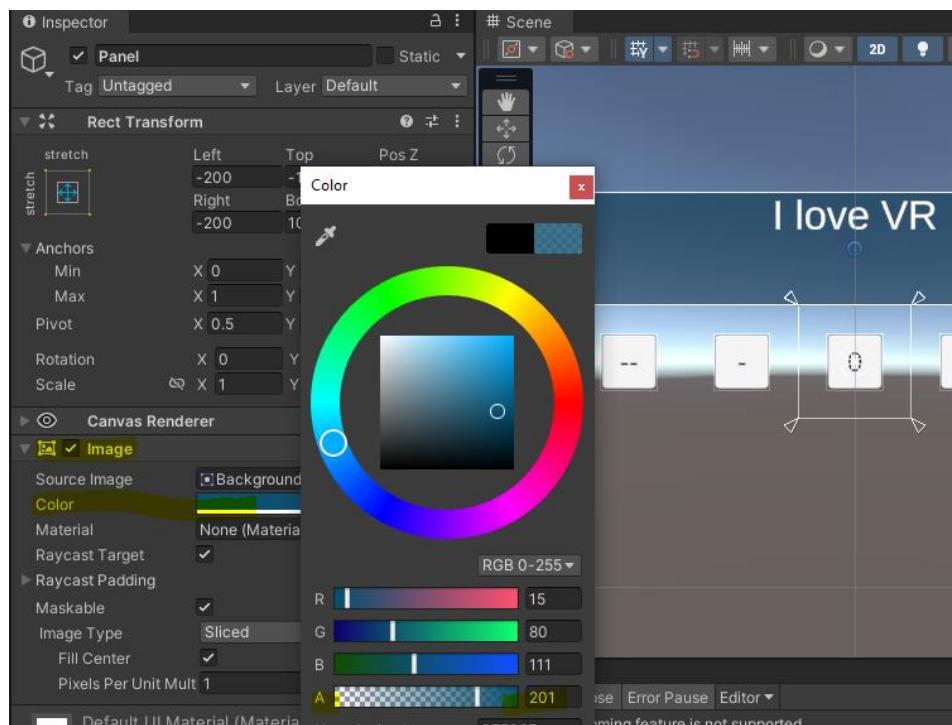
-> UI

-> Panel



2) In Rect Transform  
of that object change  
- Left to -200  
- Right to 200  
- Top to -100  
- Bottom to 100

3) In Image pick-up  
some nice color



4) Move that Panel  
above Text (TMP)



# Questionnaire

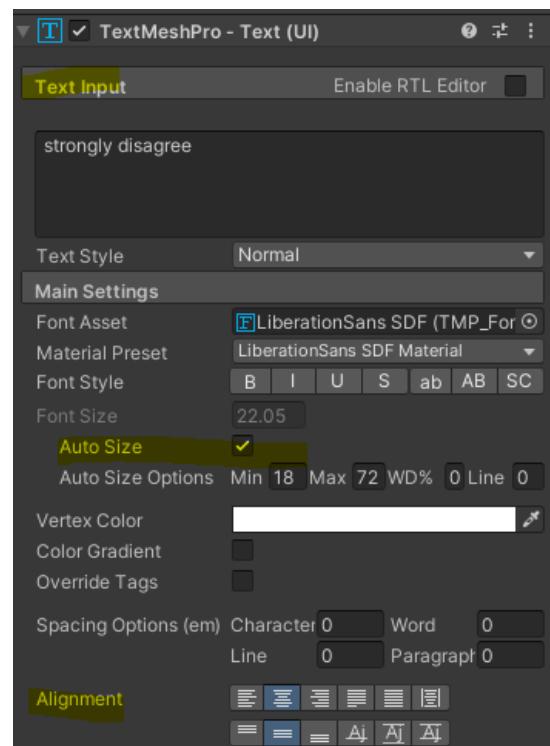
GEM2022

Pluses and minuses everywhere

1) Add an empty game object to Q1 and rename it to labels, change its Pos Y to -60

2) Add a TextMeshPro(TMP) childr to Labels game object

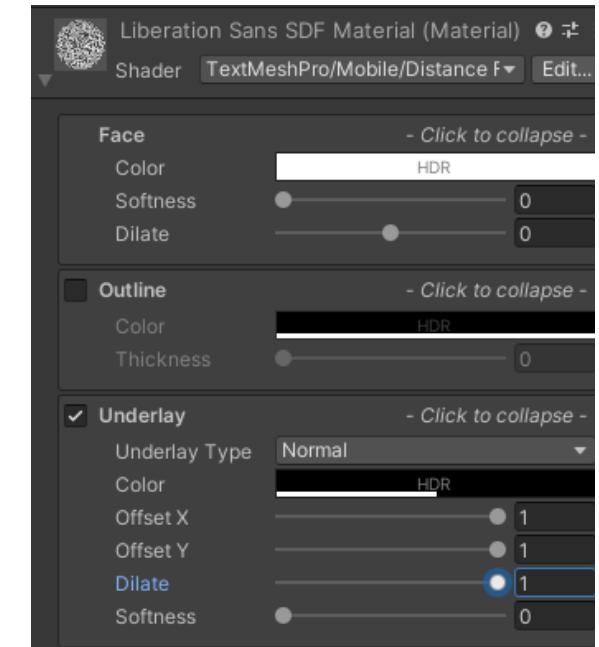
3) Change its  
- width to 90  
- alignment to center and middle  
- enable Auto Size



4) Select that text object and press <ctrl / cmd + d> 4 times

5) Change Pos X to:  
-200, -100, 0, 100, 200  
Respectively

6) Change text to match classic Likert scale



7) Maybe have some fun with the shader



# Questionnaire

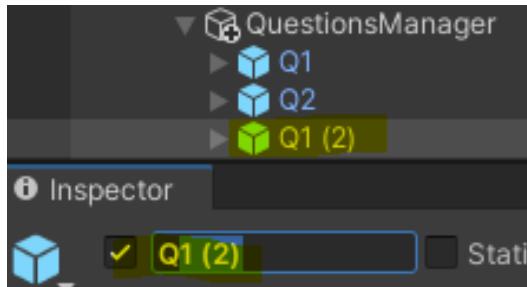
GEM2022

Adding questions

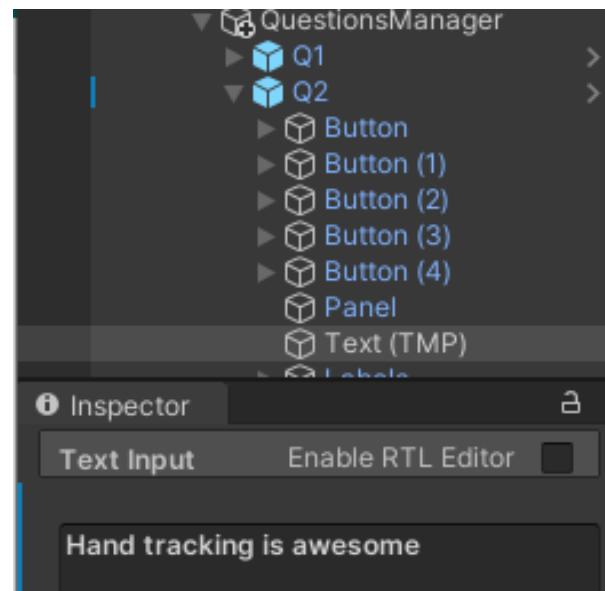
1) Drag and drop Panel UI Game Object to from Q1 to Question Manager

2) Select Q1 in Hierarchy and press <ctrl / cmd> + d twice

3) Rename these copies to Q2 and Q3



4) In Q2 -> Text (TMP): change Text Input to Hand tracking is awesome



5) In Q2 -> Text (TMP): change Text Input to This is the best conference ever

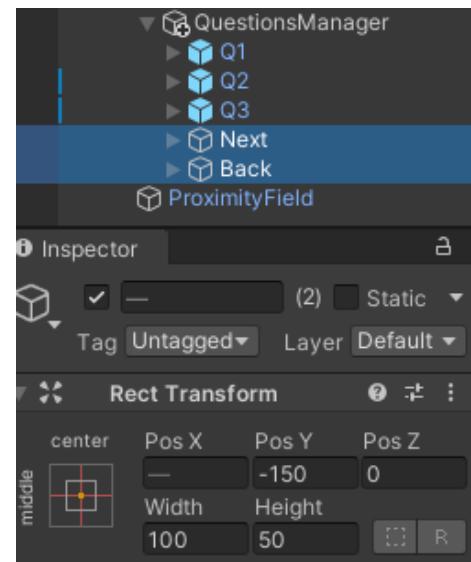
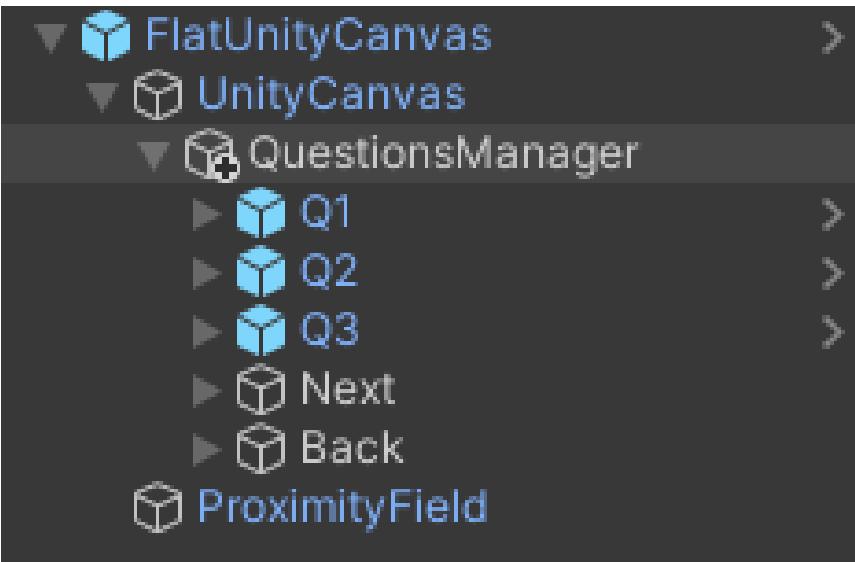


# Questionnaire

GEM2022

NEXT!

- 1) Add two new buttons to QuestionsManager:  
(RMB) -> UI -> Button - TextMeshPro  
and either repeat or duplicate it
- 2) Rename them to: Next and Back

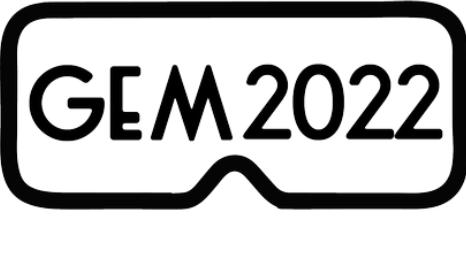


- 3) Change their Width to 100 and Height to 50
- Pos Y to -150
- and Pos X to 150 and -150 respectively

- 4) Change their Text Input on child Text (TMP) to Next and Back respectively

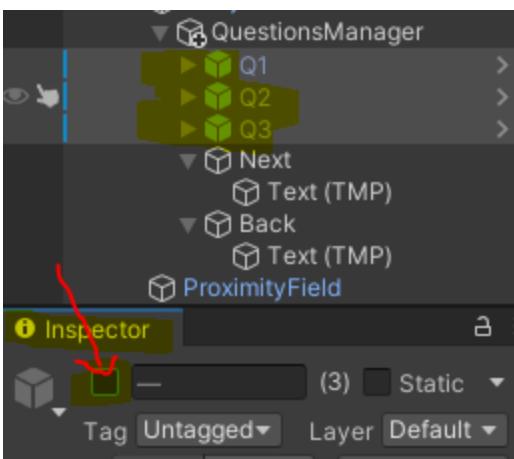


# Questionnaire

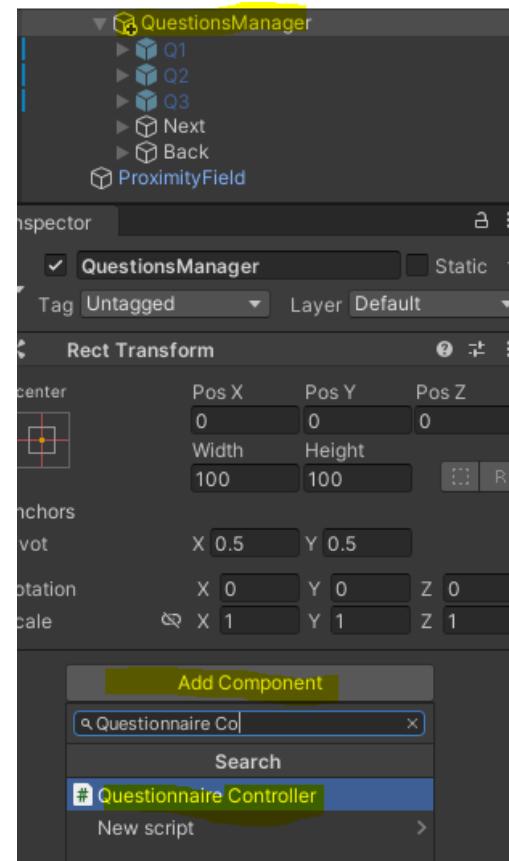


Adding some logic

- 1) Disable Q1, Q2 & Q3 in the inspector



- 3) Add it to the QuestionsManager



- 2) In Scripts folder create:  
QuestionnaireController

- 4) Open  
QuestionnaireController  
for edition

# Questionnaire

GEM2022

QuestionnaireController

1) Add using UnityEngine.UI for support of Button class

2) We need some index for current question

3) List of game objects that store these questions

4) References to next and back buttons

\*in this class we will use serialized fields to have access to these variables in editor but keeping them private...good practice...but can go public if you prefer

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class QuestionnaireController : MonoBehaviour
7  {
8      private int currentQuestion = 0;
9      [SerializeField] List<GameObject> questions;
10
11     [SerializeField] Button next;
12     [SerializeField] Button back;
```

# Questionnaire

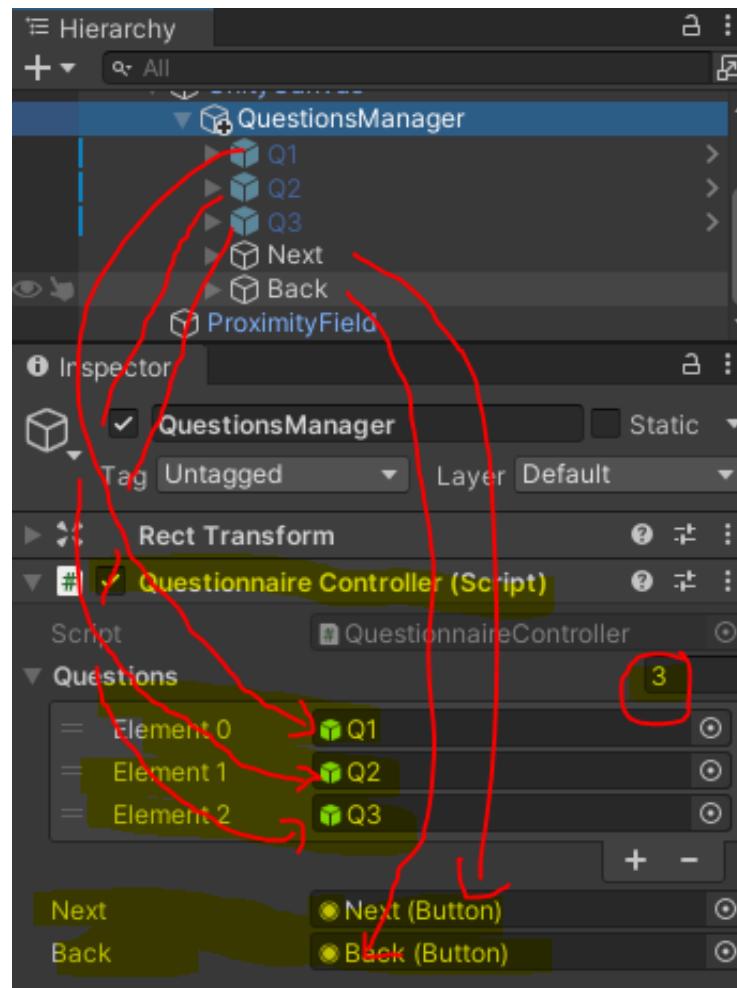


QuestionnaireController

5) Add these game objects (Q1, Q2, Q3 and Next, Back buttons) references:

Change questions number to 3

Drag and drop



# Questionnaire

GEM2022

QuestionnaireController

6) we need a method for setting active question

This method will first go through all questions in the questions list and set them as not active

After that is done it will use the current question index to set this one active

7) We will also need to set buttons making back not interactable on first question and last not interactable on last question

This method will be called at the end of setting buttons

```
private void SetActiveQuestion(int questionNumber)
{
    foreach (GameObject question in questions)
    {
        question.SetActive(false);
    }
    questions[questionNumber - 1].SetActive(true);
    SetButtons(currentQuestion);
}

private void SetButtons(int questionNumber)
{
    if (questionNumber == 1)
    {
        back.interactable = false;
    }
    else if (questionNumber == questions.Count)
    {
        next.interactable = false;
    }
    else
    {
        next.interactable = true;
        back.interactable = true;
    }
}
```

# Questionnaire

GEM2022

QuestionnaireController

- 8) We also need some method for moving between questions that will be called when next and back buttons are activated

\*for controlling direction we will use +1 or -1

```
public void ChangeQuestion(int direction)
{
    currentQuestion += direction;
    SetActiveQuestion(currentQuestion);
}
```

- 9) and we will set currentQuestion to 1 at the start and run SetActiveQuestion with this variable as a parameter

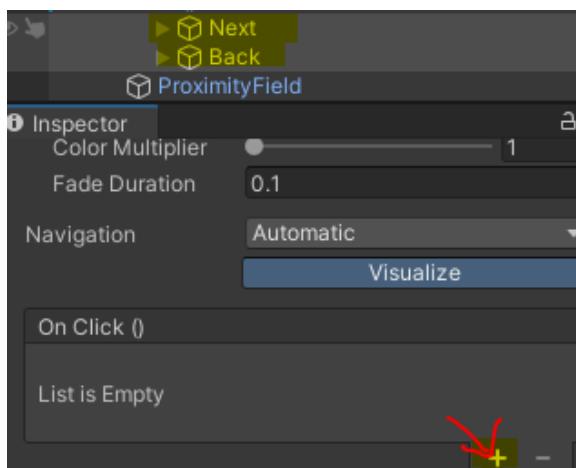
```
private void Start()
{
    currentQuestion = 1;
    SetActiveQuestion(currentQuestion);
}
```

# Questionnaire

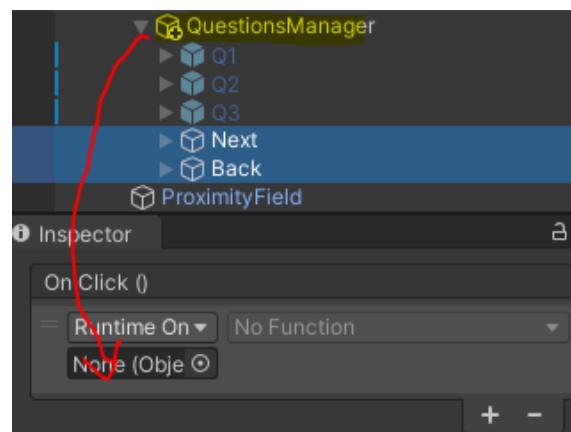


QuestionnaireController

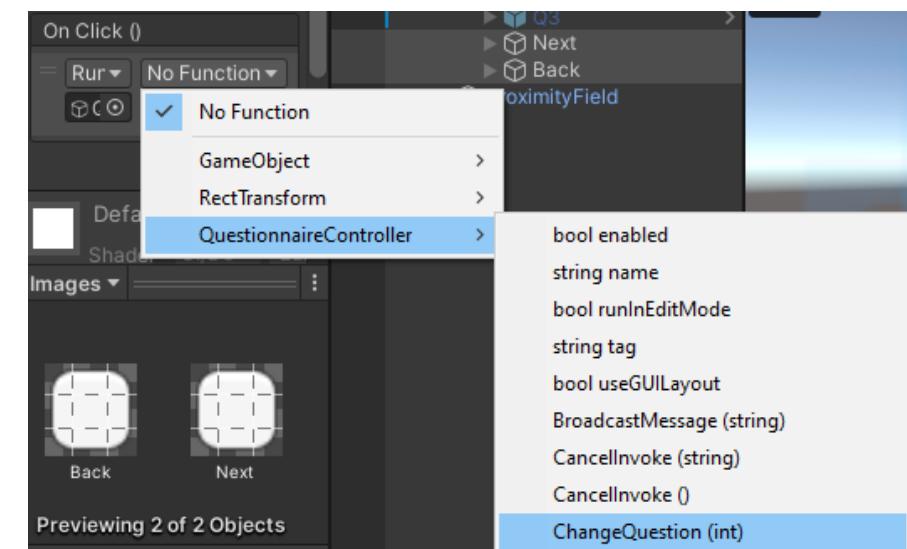
11) Select Next and Back  
in On Click hit +



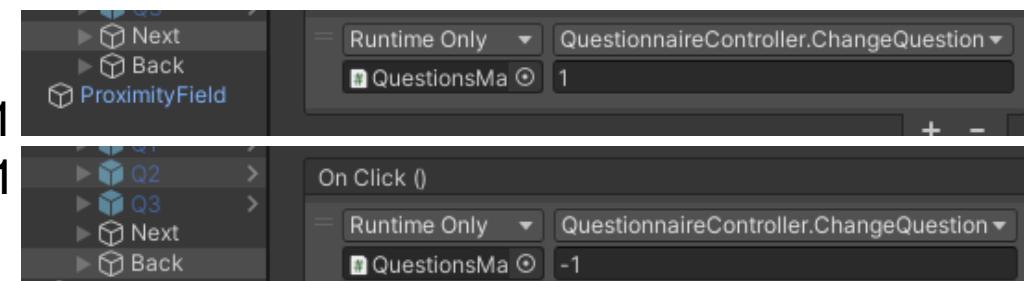
12) Drag and drop  
QuestionManager to that  
none space



13) Select the function ChangeQuestion



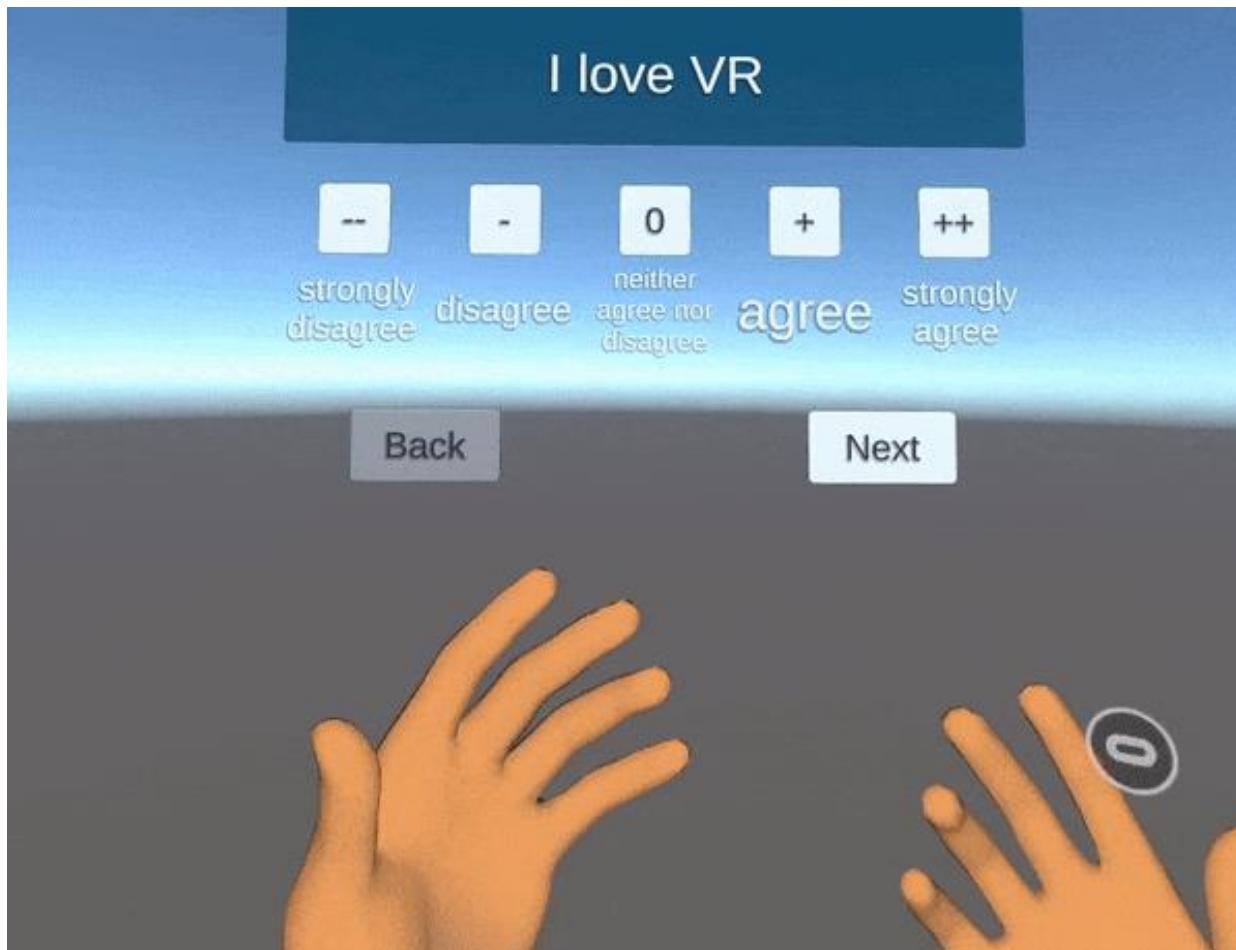
14) Change the int value for Next button to 1  
& for Back button to -1



# Questionnaire

GEM2022

Test time!



# Questionnaire

GEM2022

DataHolder

1) We need to record this answers somewhere

Use a static class for that: create a new C# script in your Scripts folder and name it DataHolder

It won't be attached to any object, but we will be using it during saving later on so we will serialize it

We will also need to access DateTime class later to use timestamp as an ID of the sample

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using System;
5
6  [System.Serializable]
7  public static class DataHolder
8  {
9  }
10
11
```

# Questionnaire

GEM2022

DataHolder

2) string ID will be used to store the timestamp

[SerializeField] public static string ID;

3) array of int Answers will store answers

[SerializeField] public static int[] Answers = { 0, 0, 0 };

4) constructor of this static class will assign timestamp to ID and format it to a string that can be used for naming files if needed

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

[System.Serializable]
1 reference
public static class DataHolder
{
    [SerializeField] public static string ID;
    [SerializeField] public static int[] Answers = { 0, 0, 0 };

    0 references
    static DataHolder()
    {
        ID = DateTime.UtcNow.ToString().Replace("/", "-").Replace(" ", "_T_").Replace(":", "-")
            + "." + System.DateTime.UtcNow.Millisecond.ToString();
    }
}
```

```
static DataHolder()
{
    ID = DateTime.UtcNow.ToString().Replace("/", "-").Replace(" ", "_T_").Replace(":", "-")
        + "." + System.DateTime.UtcNow.Millisecond.ToString();
}
```

# Questionnaire

GEM2022

Edit: QuestionController

- 1) To update that DataHolder we need to add one line to NewAnswer method in QuestionController

```
public class QuestionController : MonoBehaviour
{
    [SerializeField] List<Button> buttons; //Buttons in question holder
    public int QuestionID = 0; //Question number holder

    void References()
    {
        foreach (Button button in buttons)
        {
            button.GetComponent<Image>().color = Color.yellow;
        }
        buttons[answer - 1].GetComponent<Image>().color = Color.green;
        DataHolder.Answers[QuestionID - 1] = answer;
    }
}
```

# Questionnaire

GEM2022

SaveData

1) This data will be saved to CSV file by another static serialized class

Create a new C# script in your Scripts folder and open it

System namespace will be used for timestamps and System.IO for writing files using StreamWriter class

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using System.IO;
```

```
[System.Serializable]
2 references
public static class SaveData
```

# Questionnaire

GEM2022

SaveData

2) We need to specify where to save this csv file, for that we will write a method GetPath

String fileName will hold the name of the file

Method returns a data path to the folder where the file will be stored concatenated with the file name in a safe way to avoid system errors

```
private static string GetPath()
{
    string fileName = "questionnaire.csv";
    return Path.Combine(Application.persistentDataPath, fileName);
}
```

# Questionnaire

GEM2022

SaveData

3) SaveFile method will first call GetPath method and assign its value to string path variable

Next it will check if the file exists already, and if not it will create one using StreamWriter class writing column names

Next instance of StreamWriter will use second parameter set to true to specify that file will be overwritten

Retrieves ID from DataHolder as the first part of the string that will be saved, loops through all Answers in DataHolder to add them to that string separated by commas and writes them into the file before it **closes**

```
public static void SaveFile()
{
    string path = GetPath();

    if (!File.Exists(path))
    {
        string columns = "time, question 1, question 2, question 3";
        using (StreamWriter columnsWriter = new StreamWriter(path))
        {
            columnsWriter.WriteLine(columns);
        }
    }

    StreamWriter sw = new StreamWriter(path, true);
    string answers = DataHolder.ID;
    foreach (int answer in DataHolder.Answers)
    {
        answers += "," + answer;
    }
    sw.WriteLine(answers);
    sw.Close();
}
```

# Questionnaire

GEM2022

Edit: QuestionnaireController

- 1) QuestionnaireController rules them all so needs an option to save data

Add method Submit to that class that will call the SaveFile method from the SaveData static class

```
public void Submit()
{
    SaveData.SaveFile();
}
```

# Questionnaire



Final page

1) there is a lot of ways to trigger, but to be polite final page of this questionnaire must be added to say thanks and give an option to submit answers or go back and change them

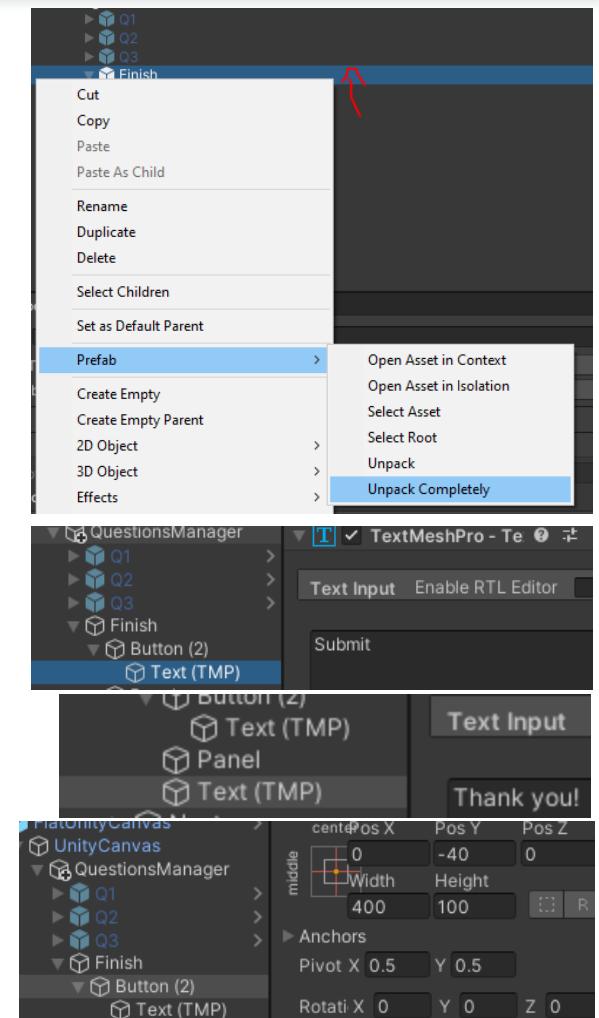
2) to make it quick add another instance of Q1 prefab by dragging and dropping it to QuestionsManager game object. Rename it to Finish



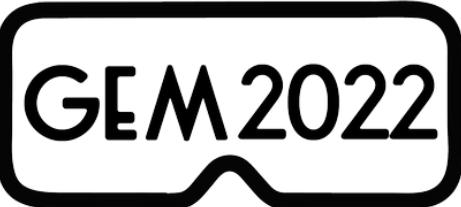
- 2) Move it just below Q3 and Unpack Completely
- 3) Delete all the buttons except the middle one [Button (3)] and Labels



- 4) On the child Text (TMP)  
Change Text Input to Submit
- 5) Change Text (TMP) Text Input to Thank you
- 6) In Button (2) Rect Transform  
change Width to 400 Height to 100  
And Pos Y to -40

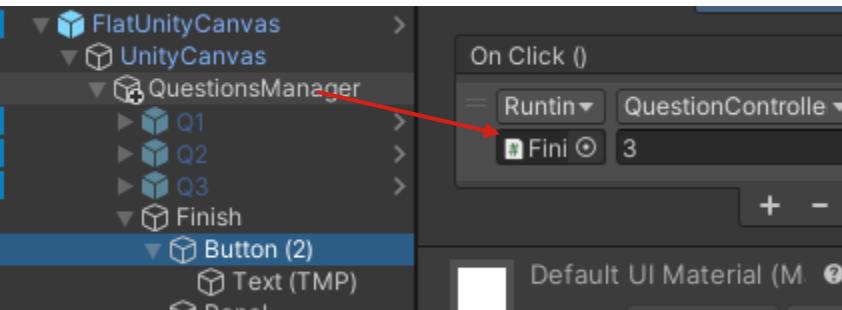


# Questionnaire

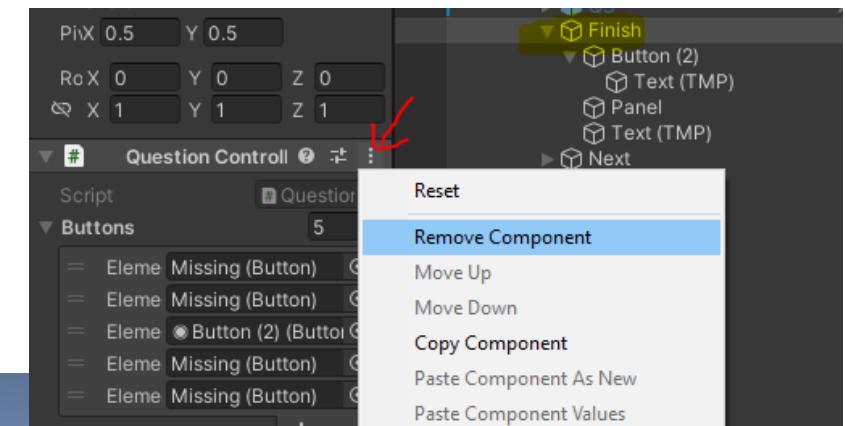
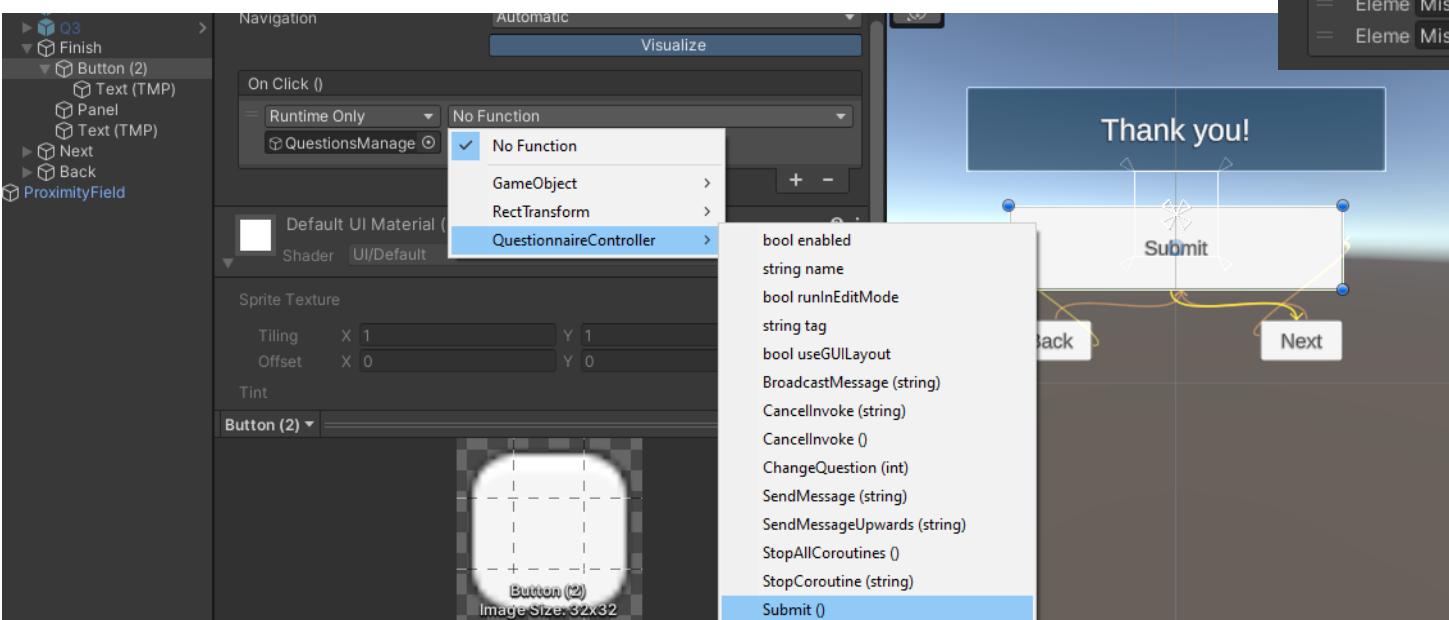


Final page

7) Drag the QuestionsManager To replace Finish game object in On Click field on Button (2)



8) Select Submit()



9) In Finish GameObject

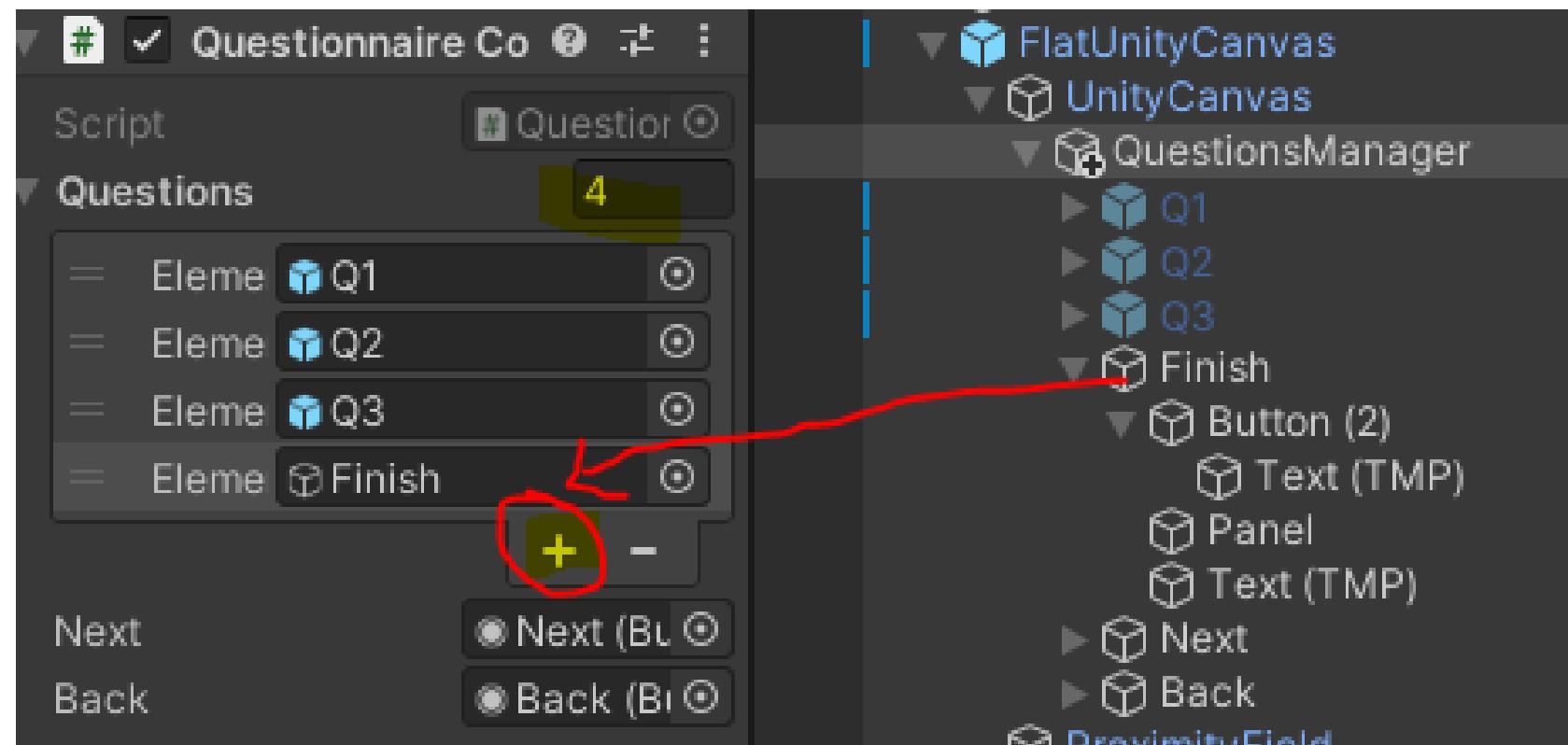
Remove:  
QuestionController  
from the

# Questionnaire

GEM2022

Final page

10) In QuestionsManager click on the + to add one more question and drag and drop that Finish GameObject to that field



# Questionnaire

GEM2022

Edit: SaveData

- 1) To reset the questionnaire after answer was submitted add namespace

using UnityEngine.SceneManagement;

and at the end of SaveFile method add:

```
public static void SaveFile()
{
    string path = GetPath();

    if (!File.Exists(path))
    {
        string columns = "time, question 1, question 2, question 3";
        using (StreamWriter columnsWriter = new StreamWriter(path))
        {
            columnsWriter.WriteLine(columns);
        }
    }

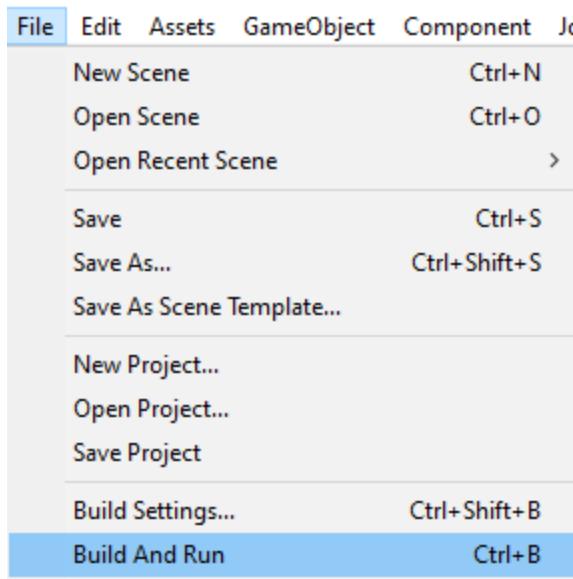
    StreamWriter sw = new StreamWriter(path, true);
    string answers = DataHolder.ID;
    foreach (int answer in DataHolder.Answers)
    {
        answers += "," + answer;
    }
    sw.WriteLine(answers);
    sw.Close();
}

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
```

# Questionnaire

GEM2022

File -> Build And Run  
To test it



CSV should be in:

[your pc]\Quest\Internal shared storage\Android\data\com.TUDublin.GEM2022tutorial\files

# Wearable button



Making something to press

1) Add an Empty GO:  
In Hierarchy (RMB) ->  
Create Empty

Rename it to  
CylinderButton

4) Add one more  
Rename it to  
Visual

2) Add an Empty GO as a  
child of CylinderButton

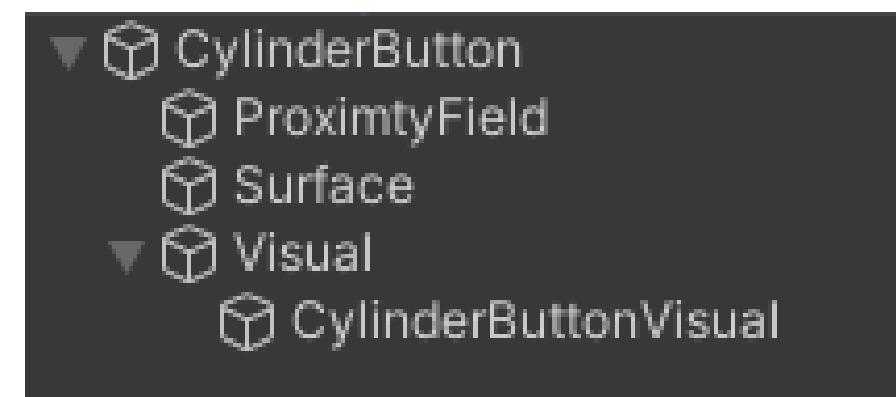
Rename it to  
ProximityField

5) Add one child (RMB) ->  
3D Object -> Cylinder  
to Visual GO

Rename it to  
CylinderButtonVisual

3) Add another Empty  
GO as a child of  
CylinderButton

Rename it to  
Surface

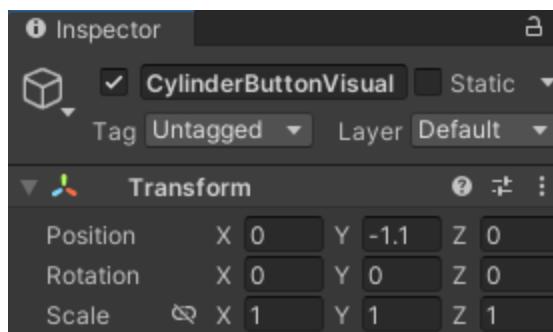


# Wearable button



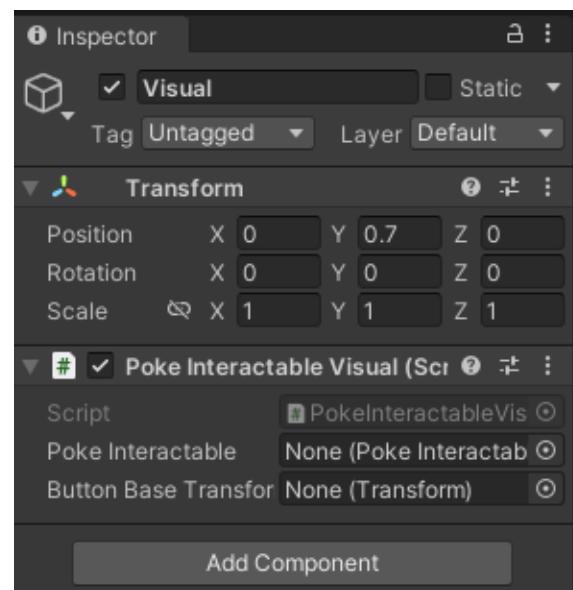
Making something to press

- 1) in CylinderButtonVisual  
Change Position Y to -1.1



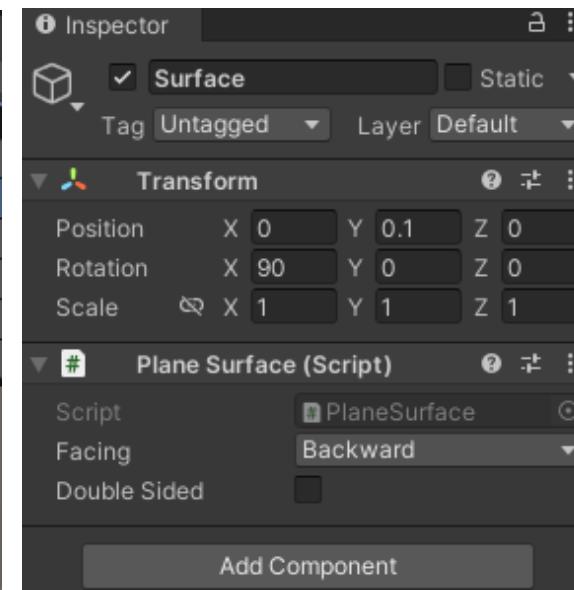
- 2) Change Visual  
Position Y to 0.7

Add Component:  
Poke Interactable Visual



- 3) Change Surface  
Position Y to 0.1  
Rotate X to 90

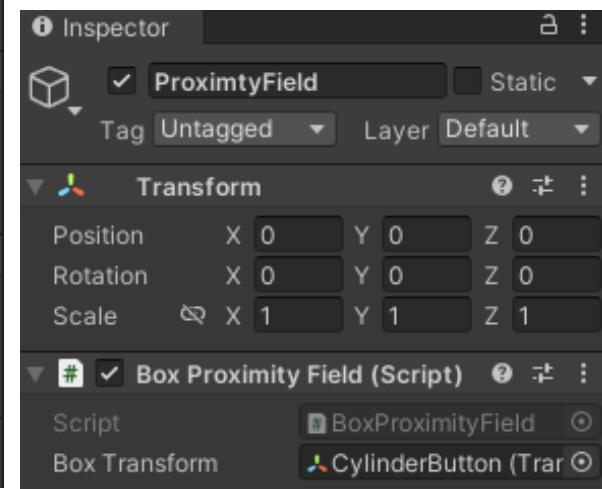
Add Component:  
Plane Surface



- 4) To ProximityField

Add Component:  
Box Proximity Field

Drag and drop  
CylinderButton GO to  
Box Transform Field



# Wearable button



Making something to press

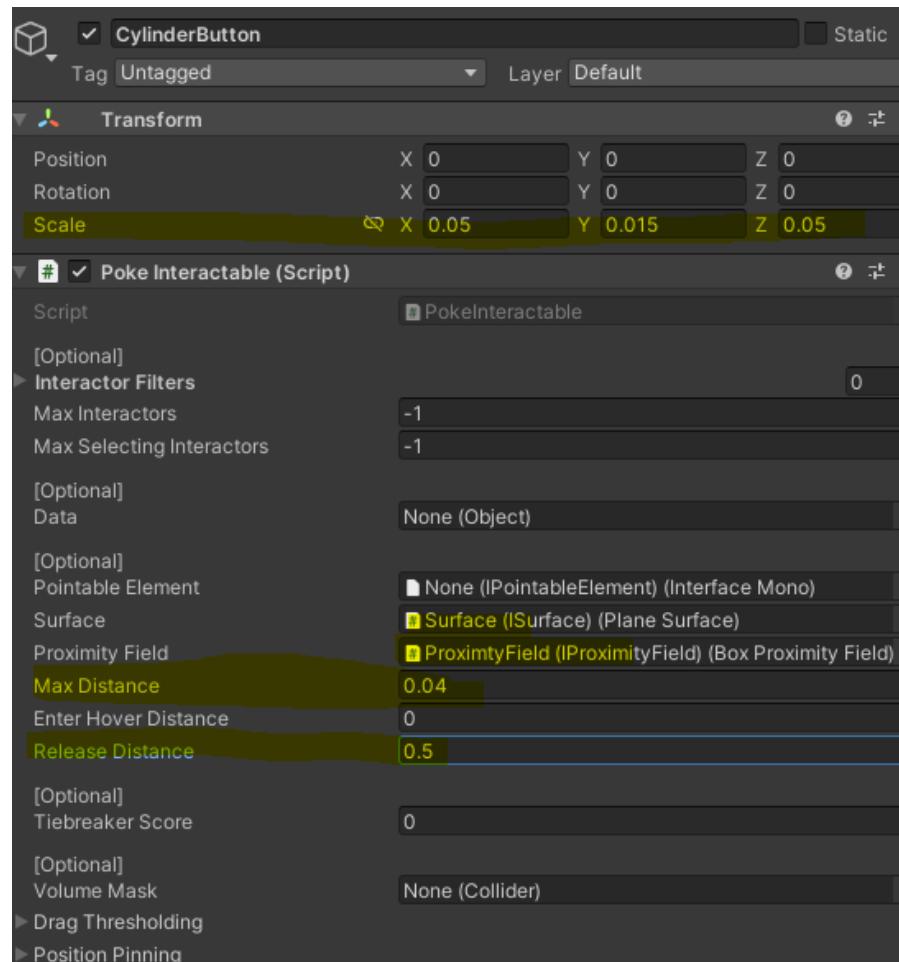
5) To CylinderButton GO

Add Component:  
Poke Interactable

Drag and drop  
ProximityField and Surface to  
corresponding fields

Change Scale to  
X: 0.05 ; Y: 0.015 ; Z: 0.05

Change Max Distance to 0.04  
and Release Distance to 0.5



# Wearable button

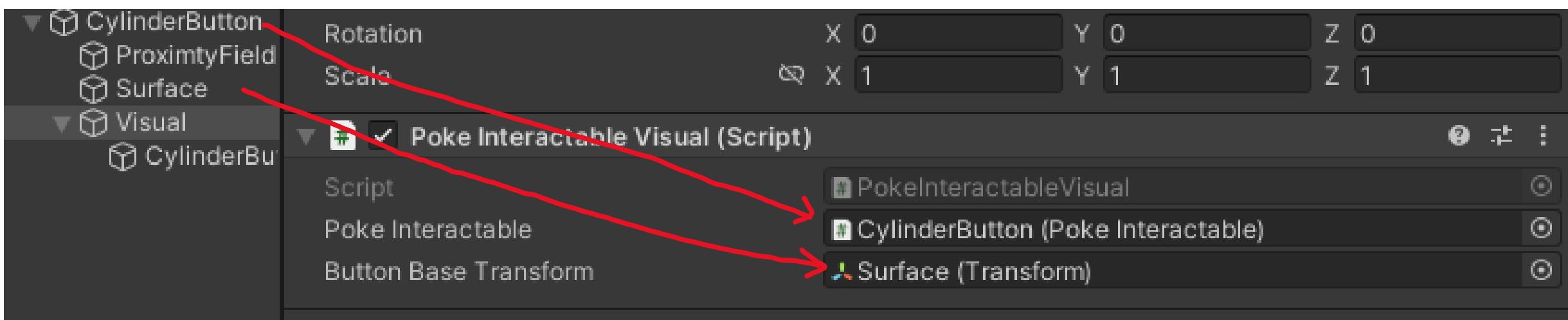
GEM2022

Making something to press

## 5) In Visual GO

Drag and drop  
CylinderButton GO to Poke Interactable field

And Surface to Button Base Transform



# Wearable button



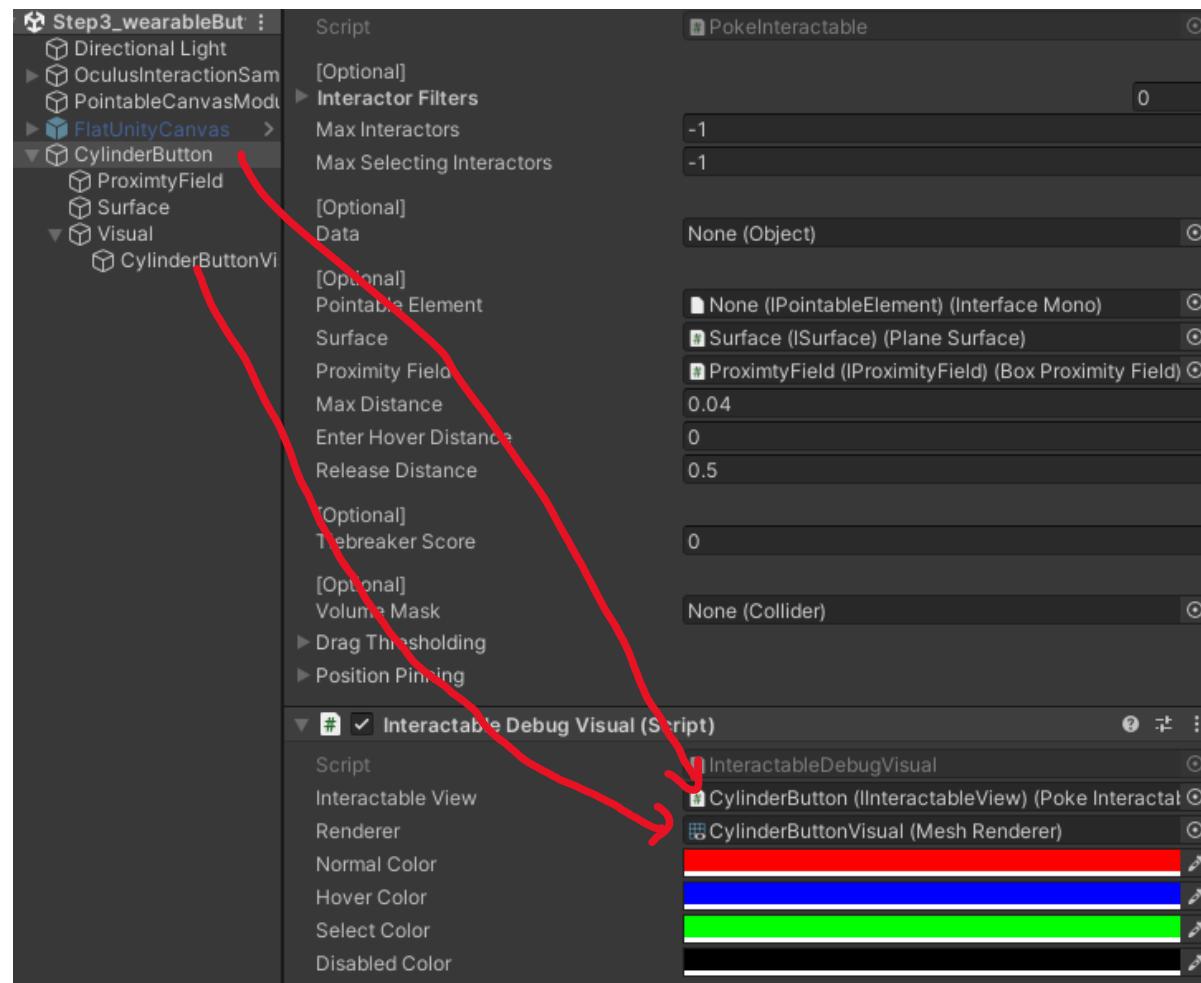
Making something to press

1) to test if it works

Add Component  
Interactable Debug Visual  
To CylinderButton GO

Drag and drop  
CylinderButton to  
Interactable View

CylinderButtonVisual to  
Renderer

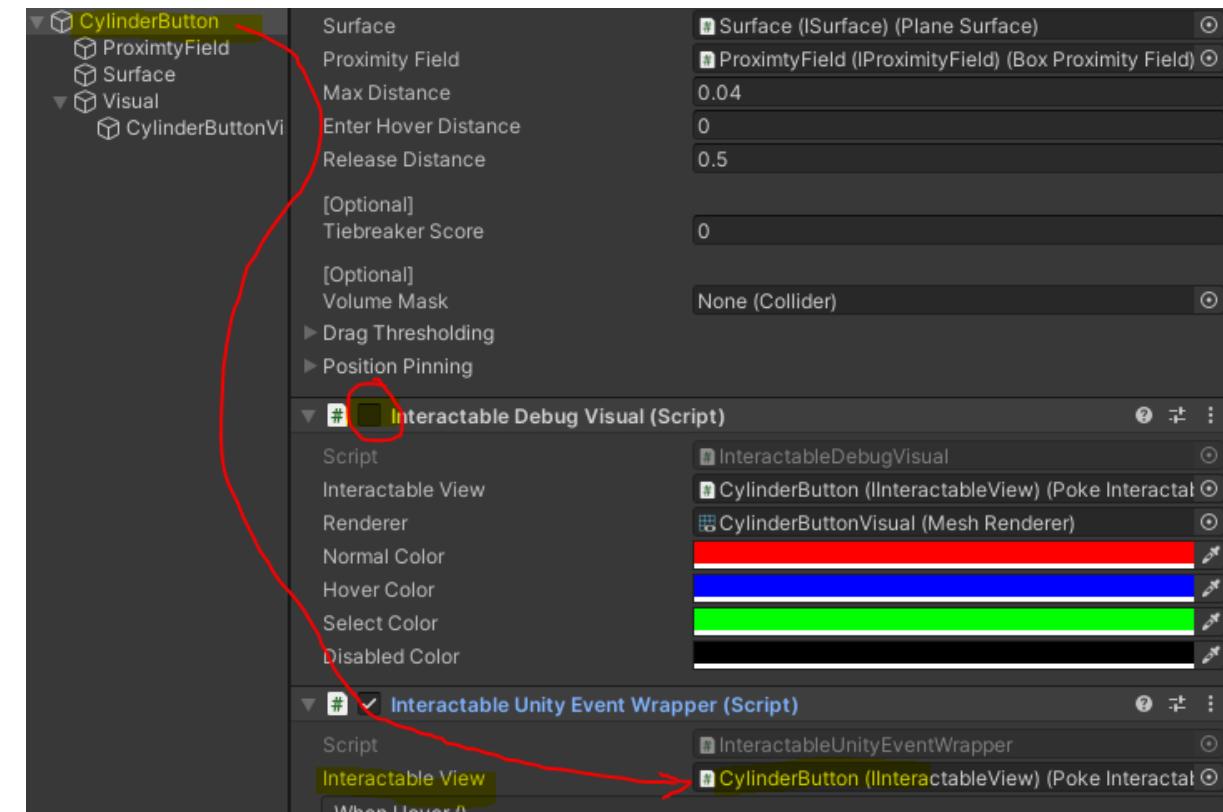


# Wearable button



Making something to press

- 1) Add Interactable Unity Event Wrapper to CylinderButton
- 2) Drag and drop CylinderButton to Interactable View
- 3) After testing Disable Interactable Debug Visual

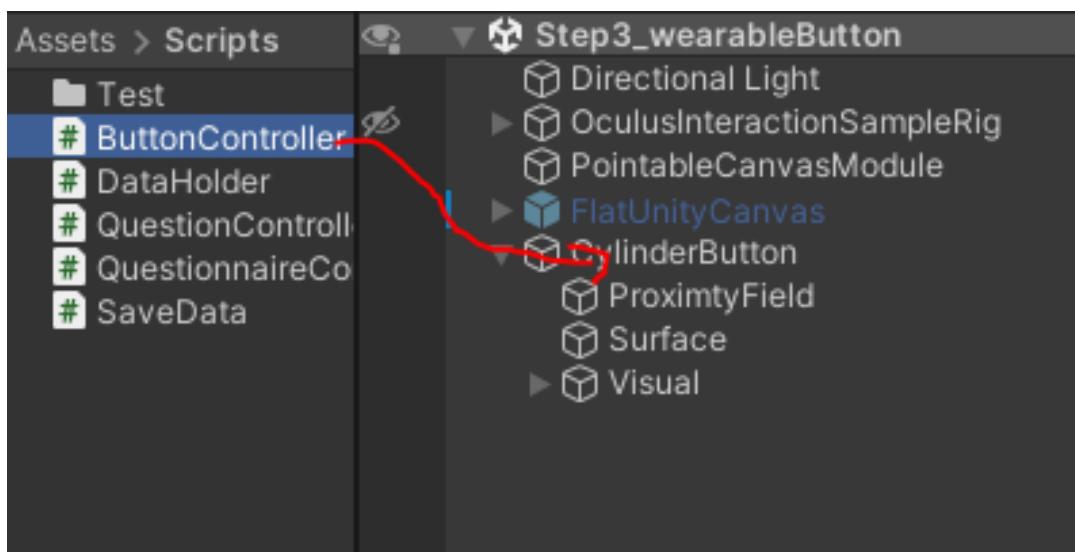


# Wearable button

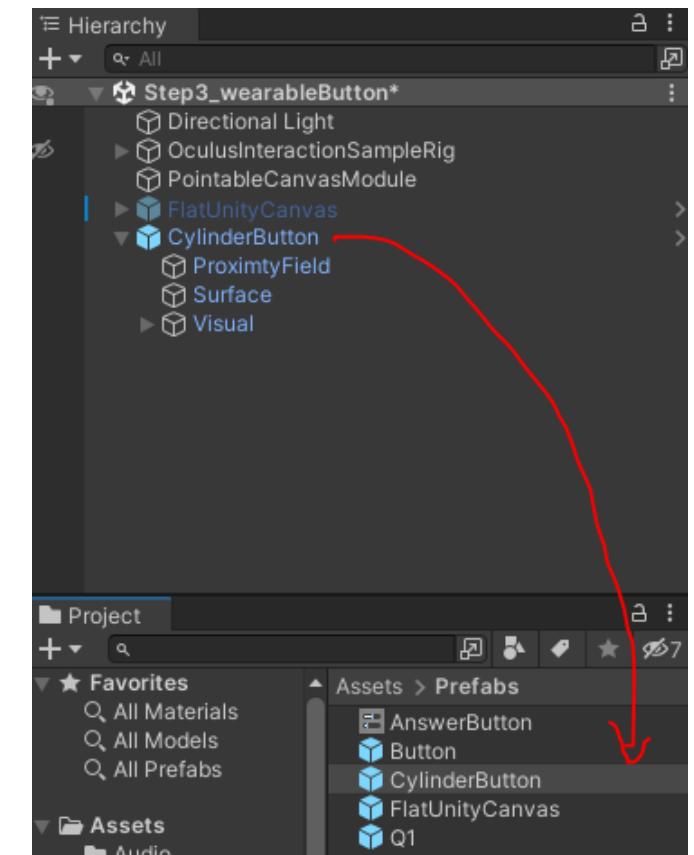
GEM2022

Prefabricate it

- 1) Create a new C# script in your Script folder  
Drag and drop it onto that CylinderButton GO



- 2) Drag and drop CylinderButton GO into Prefabs folder



# Wearable button

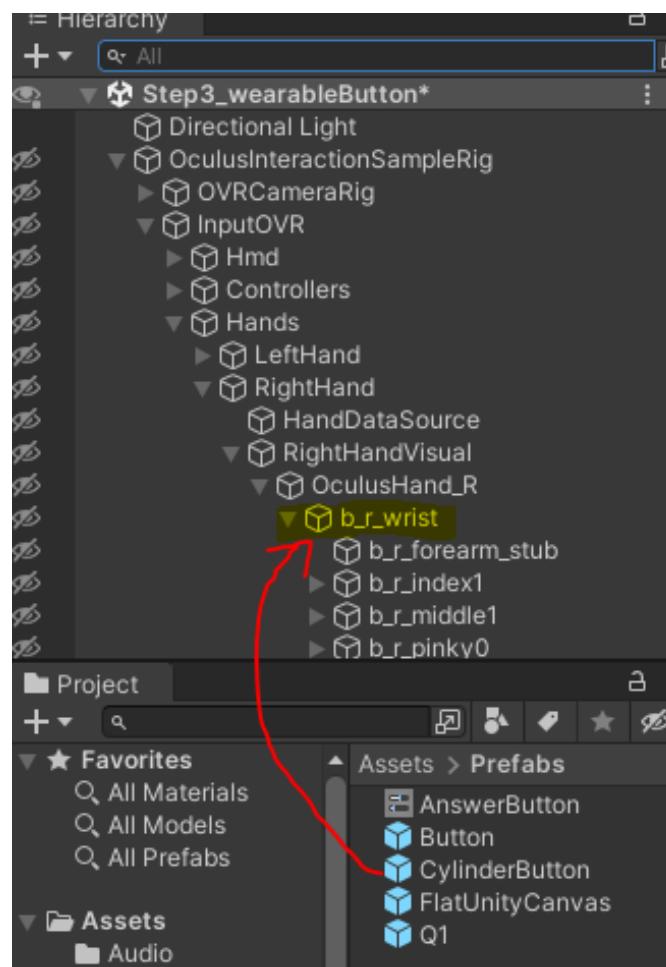


Mount the button!

- 1) In Hierarchy search for b\_r\_wrist click on it and click x on search



- 2) Drag and drop CylinderButton prefab on that b\_r\_wrist GO



# Wearable button

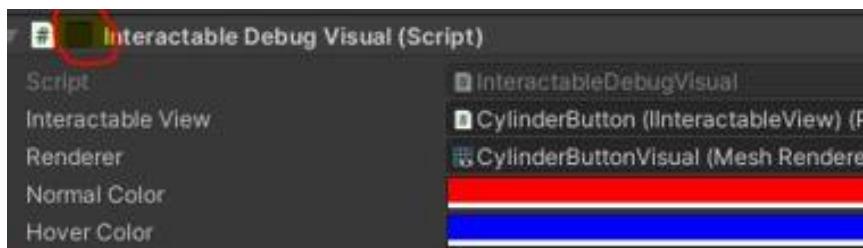
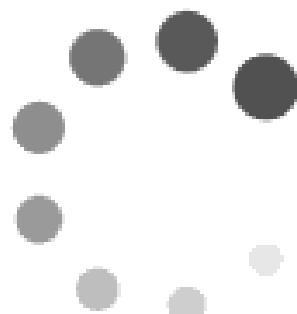


Making something to press  
Any questions?

Experiment with Y position on that Visual to make sure it's pressable

Y: 2 looks best for me

After this test disable Interactable Debug Visual



# Wearable button

GEM2022

Adding a pinch of logic

## 1) Open DataHolder script

Add one line to hold  
Boolean variable TurnedOn

```
public static class DataHolder
{
    [SerializeField] public static string ID;

    [SerializeField] public static int[] Answers = { 0, 0, 0 };

    [SerializeField] public static bool TurnedOn = false;

    0 references
    static DataHolder()
    {
        ID = DateTime.UtcNow.ToString().Replace("/", "-").Replace(" ", "_T_").Replace(":", "-")
            + "." + System.DateTime.UtcNow.Millisecond.ToString();
    }
}
```

# Wearable button

A 3D rendering of a glowing blue button with the text "GEM2022" on it.

And some colors

- 1) Open ButtonHolder script

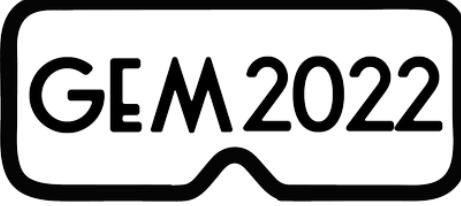
Add two Color variables:  
turnedOnColor  
turnedOffColor

And a MeshRenderer variable to  
MeshRenderer

```
public class ButtonController : MonoBehaviour
{
    [SerializeField] Color turnedOnColor;
    [SerializeField] Color turnedOffColor;

    private MeshRenderer meshRenderer;
```

# Wearable button



And some colors

2) in the start method assign MeshRenderer component found in children of this GO to meshRenderer

3) next assign the value of turnedOffColor variable to the color of the material of the first component in the array of MeshRenderers

4) add a public method called Selected where turnedOnColor will be switched when button is pressed

```
public class ButtonController : MonoBehaviour
{
    [SerializeField] Color turnedOnColor;
    [SerializeField] Color turnedOffColor;

    private MeshRenderer meshRenderer;

    private void Start()
    {
        meshRenderer = this.GetComponentInChildren<MeshRenderer>();
        meshRenderer.material.color = turnedOffColor;
    }

    public void Selected()
    {
        meshRenderer.material.color = turnedOnColor;
    }
}
```

# Wearable button

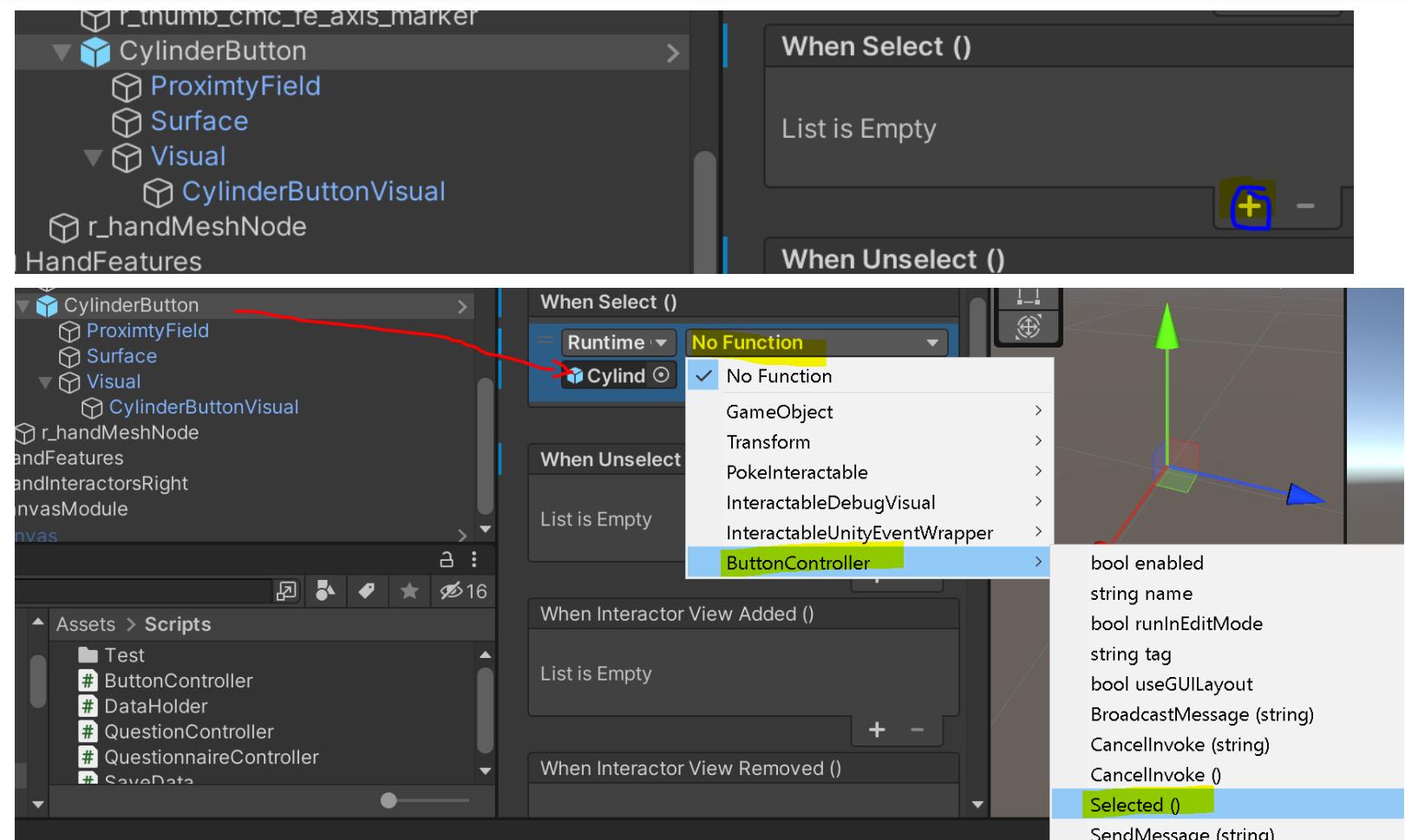


Hook it up

1) Select CylinderButton  
GO, scroll down to When  
Selected

2) Drag and drop  
CylinderButton GO to  
empty (None) field that you  
just added

3) From available functions  
-> ButtonController ->  
Selected()

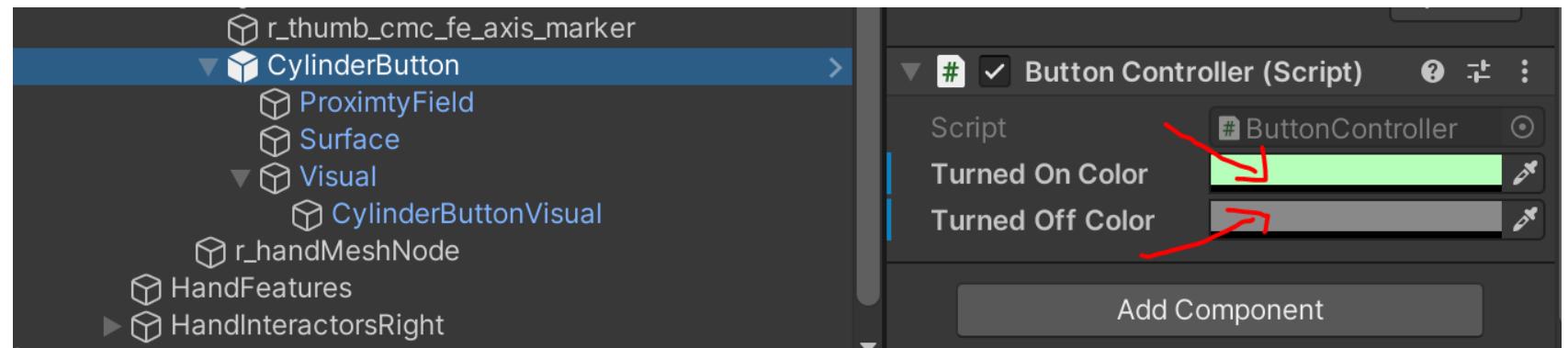


# Wearable button



Choose your colors wisely

- 1) On CylinderButton GO scroll down to ButtonController and change the colors for turned on and off variables as you want



# Wearable button



To on or off,  
that is the question

1) to Selected method in  
ButtonController add a  
switch that will change  
TurnedOn in DataHolder to  
its opposite value

2) add an if statement based  
on value of TurnedOn in  
DataHolder to change colors  
correctly

```
public void Selected()
{
    DataHolder.TurnedOn = !DataHolder.TurnedOn;
    if (DataHolder.TurnedOn)
    {
        meshRenderer.material.color = turnedOnColor;
    }
    else
    {
        meshRenderer.material.color = turnedOffColor;
    }
}
```

# Wearable button

GEM2022

SaveFile

- 1) in SaveData class we need to add another way of saving timeseries data.

Add a ts Boolean variable as a parameter to GetPath method, and before returning path add a small if statement overriding the file name using ID value stored in DataHolder

```
private static string GetPath(bool ts)
{
    string fileName = "questionnaire.csv";
    if (ts)
    {
        fileName = DataHolder.ID + ".csv";
    }
    return Path.Combine(Application.persistentDataPath, fileName);
}
```

# Wearable button

The logo consists of the letters "GEM" stacked vertically above the year "2022". The entire logo is enclosed in a rounded rectangular frame with a thick black border.

SaveFile

2) we will also add same parameter to SaveFile method but this time make it optional set to false as default

3) Pass local ts to GetPath method

4) override columns string if SaveFile is executed to save timeseries data

```
public static void SaveFile(bool ts = false)
{
    string path = GetPath(ts);

    if (!File.Exists(path))
    {
        string columns = "time, question 1, question 2, question 3";
        if (ts)
        {
            columns = "time, button";
        }
        using (StreamWriter columnsWriter = new StreamWriter(path))
        {
            columnsWriter.WriteLine(columns);
        }
    }
}
```

# Wearable button

GEM2022

SaveFile

2) and fork the code to save as before if ts is false or use the timestamp as an index and value stored in DataHolder TurnedOn variable as a value for button (second) column

```
StreamWriter sw = new StreamWriter(path, true);

string answers;
if (!ts)
{
    answers = DataHolder.ID;
    foreach (int answer in DataHolder.Answers)
    {
        answers += "," + answer;
    }
}
else
{
    answers = DateTime.UtcNow.ToString() +
        "." + DateTime.UtcNow.Millisecond.ToString() +
        "," +
        DataHolder.TurnedOn;
}

sw.WriteLine(answers);
sw.Close();
```

# Wearable button



GEM2022

Ready to save? Fire!

- 3) Finally, call that SaveFile method from ButtonController when Selected method is executed

```
public void Selected()
{
    DataHolder.TurnedOn = !DataHolder.TurnedOn;
    if (DataHolder.TurnedOn)
    {
        meshRenderer.material.color = turnedOnColor;
    }
    else
    {
        meshRenderer.material.color = turnedOffColor;
    }
    SaveData.SaveFile(true);
}
```

# Wearable button

A digital watch face with the text "GEM2022" displayed on its screen.

Color doesn't stay?

Add an if statement to start method in ButtonController to check if the color should match state of TurnedOn in DataHolder

```
private void Start()
{
    meshRenderer = this.GetComponentInChildren<Renderer>();

    if (DataHolder.TurnedOn)
    {
        meshRenderer.material.color = turnedOnColor;
    }
    else
    {
        meshRenderer.material.color = turnedOffColor;
    }
}
```

# Wearable button



Saving takes time

And to SaveFile function in  
SaveData add an if  
statement to make sure we  
the scene is not reloaded  
every time the button is  
pressed.

```
if (!ts)
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
}
```

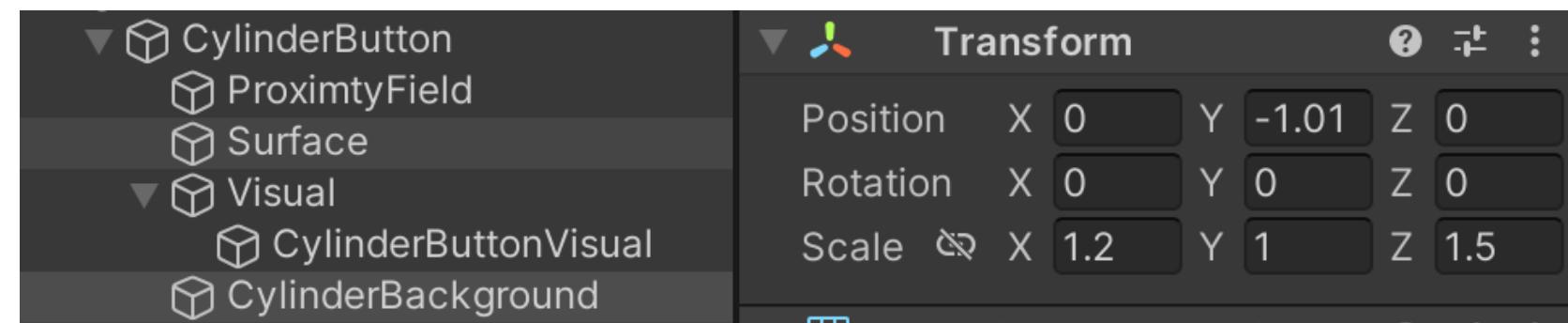
# Wearable button



Button base

1) Add a 3D cylinder as a child to CylinderButton GO

Make sure it's the last child in the stack



Change Transform

parameters:

- Position Y to -1.01
- Scale X to 1.2 Z to 1.5

# Wearable button



Time to test if it works  
Any questions?

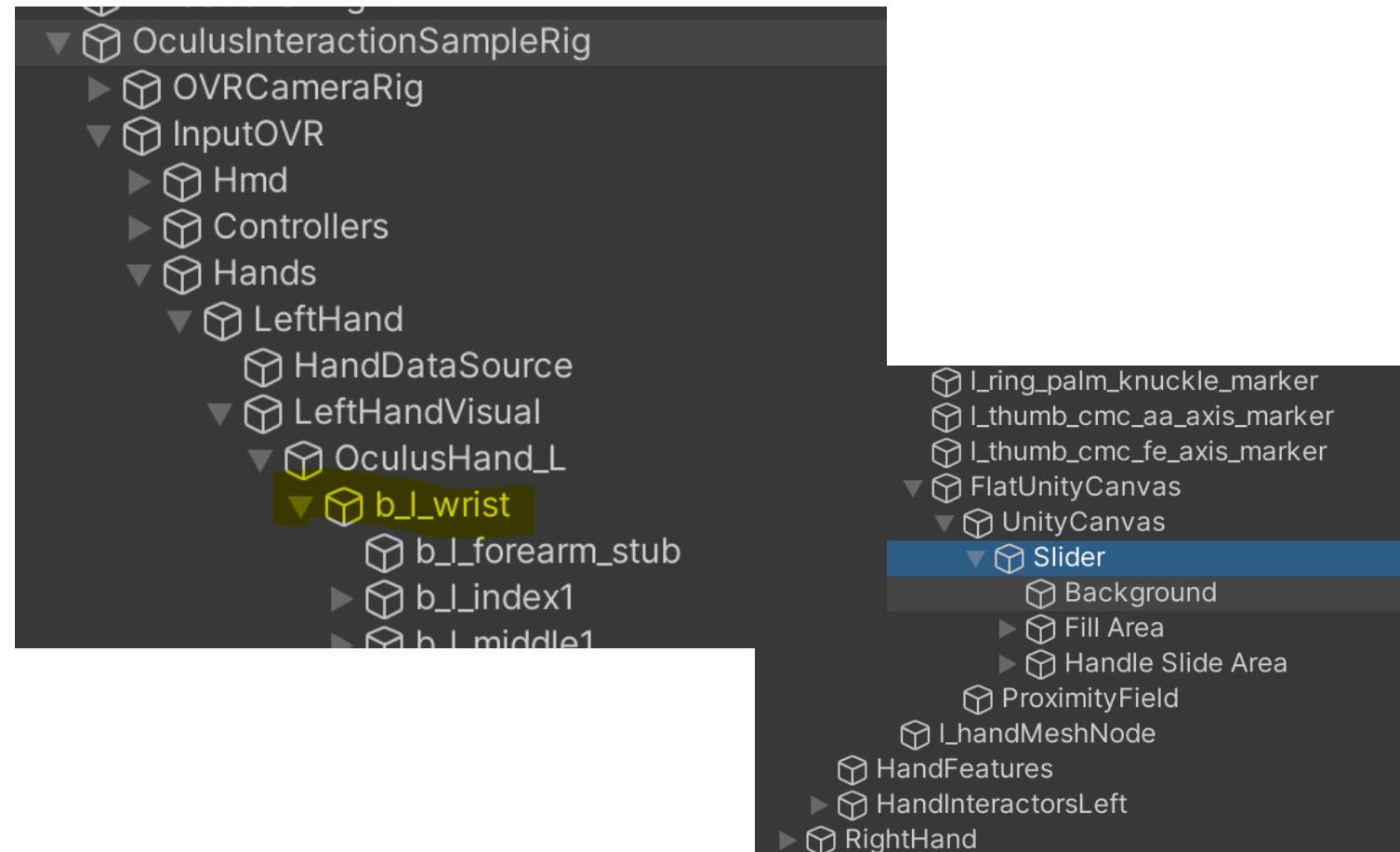


# Slider



Time to slide

- 1) Drag and drop on b\_L\_wrist GO
- 2) Unpack it
- 3) RMB -> UI -> Slider
- 4) Move that Slider GO to UnityCanvas  
delete and the standard Canvas



# Slider

5) on FlatUnityCanvas  
In Transform  
Change  
Position y to -0.05  
Rotation X to 90  
Rotation Y to -180

In Poke Interactable  
Change  
Release distance to  
0.025



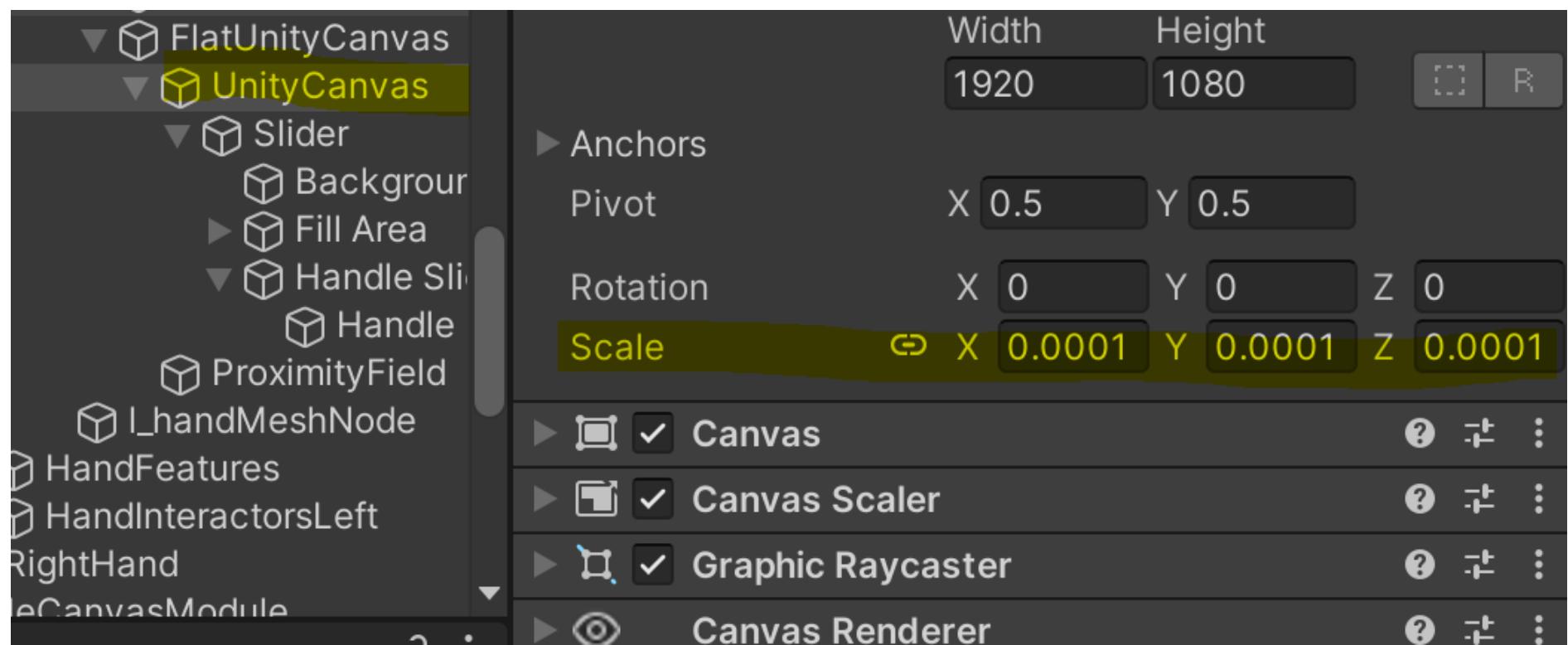
Time to slide

# Slider



Time to slide

6) on UnityCanvas  
RectTransform  
Change  
Scale to 0.0001



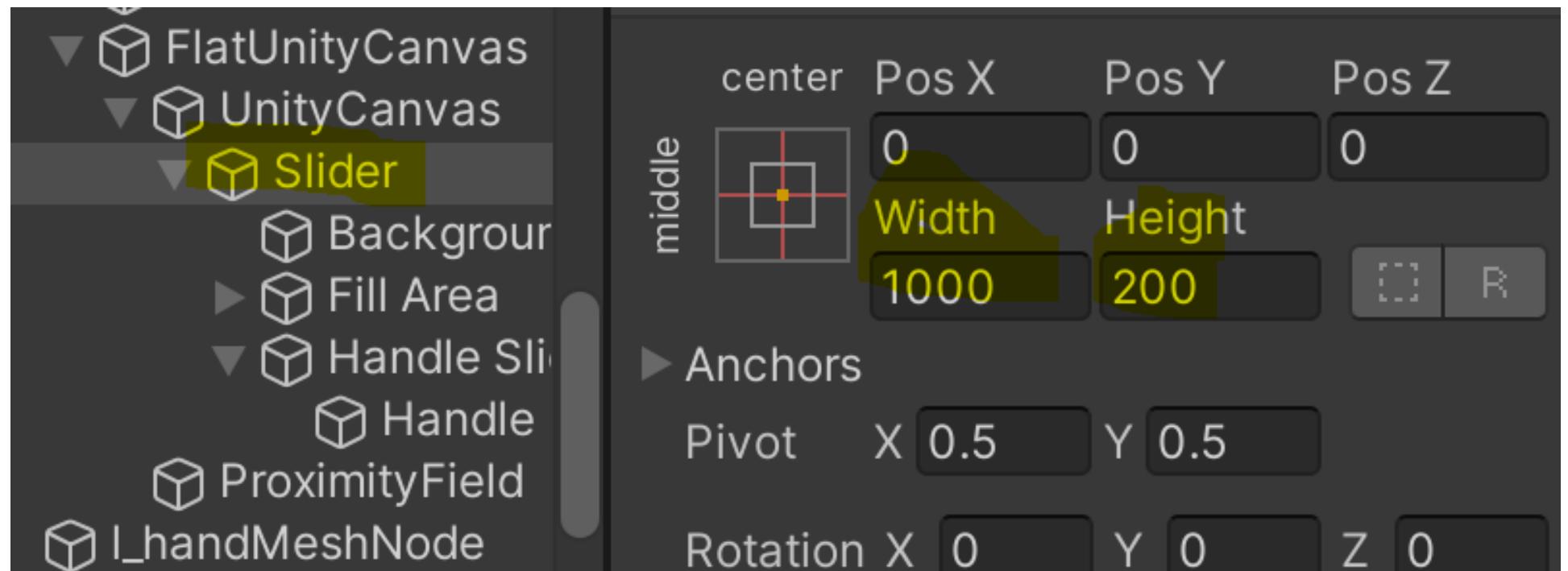
# Slider



Time to slide

7) on Slider  
Change

Width to 1000  
Height to 200



# Slider

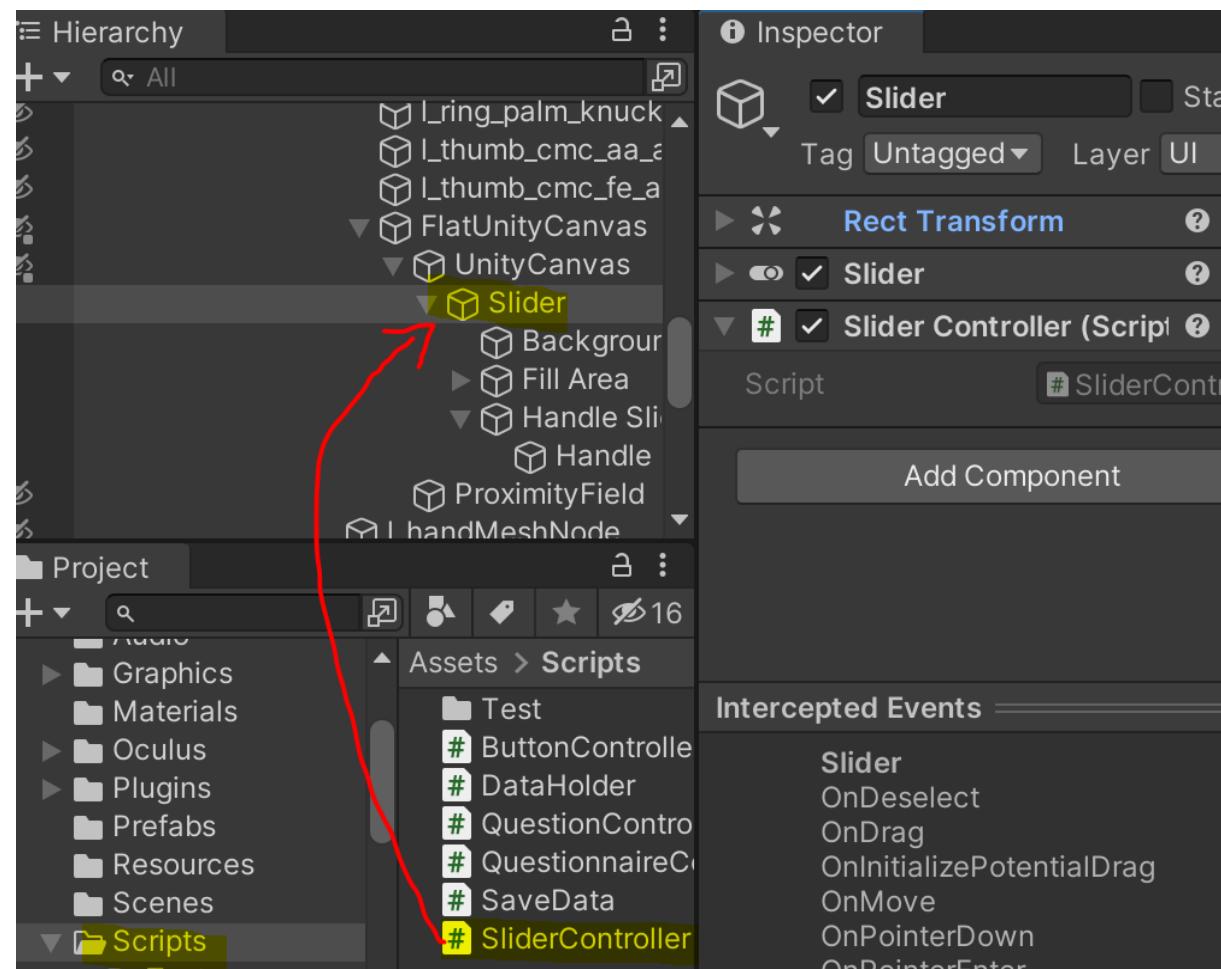


SliderController

7) Create a new C# script in Scripts folder named SliderController

8) Drag and drop it on the Slider GO

9) Open that script to edit it



# Slider

- 1) Add UI namespace
- 2) Create fields slider and an array of Images
- 3) at the start assign Slider class of this component to slider variable  
And all its Image components to sliderImages array
- 4) public method NewValue will loop through this list when value will change and lerp between blue and red



Make it hot!

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class SliderController : MonoBehaviour
{
    Slider slider;
    Image[] sliderImages;
    // Start is called before the first frame update
    void Start()
    {
        slider = this.GetComponent<Slider>();
        sliderImages = slider.GetComponentsInChildren<Image>();
    }

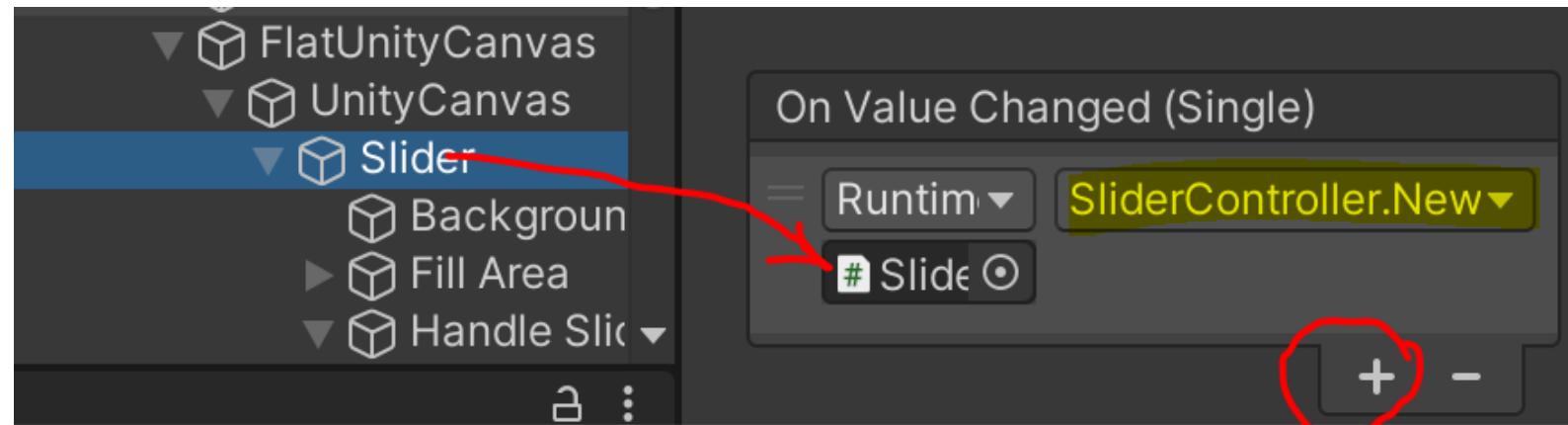
    public void NewValue()
    {
        foreach(Image image in sliderImages)
        {
            image.color = Color.Lerp(Color.blue, Color.red, slider.value);
        }
    }
}
```

# Slider



Make it hot!

5) Hit + for On Value  
Changed, drag and drop  
Slider GO to this empty field  
select NewValue method  
from available functions



# Slider



Make it SUPER hot!..saving data

- 1) To DataHolder add a float SliderValue

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

[System.Serializable]
public static class DataHolder
{
    [SerializeField] public static string ID;

    [SerializeField] public static int[] Answers = { 0, 0, 0 };

    [SerializeField] public static bool TurnedOn = false;

    [SerializeField] public static float SliderValue = 0f;

    static DataHolder()
    {
        ID = DateTime.UtcNow.ToString().Replace("/", "-").Replace(" ", "_T_").Replace(":", "-")
            + "." + System.DateTime.UtcNow.Millisecond.ToString();
    }
}
```

# Slider

GEM2022

Make it SUPER hot!..saving data

2) In SaveData add column slider to columns for timeseries

columns = "time, button, slider";

```
using System;
using System.IO;
using UnityEngine.SceneManagement;

[System.Serializable]
public static class SaveData
{
    public static void SaveFile(bool ts = false)
    {
        string path = GetPath(ts);

        if (!File.Exists(path))
        {
            string columns = "time, question 1, question 2, question 3";
            if (ts)
            {
                columns = "time, button, slider";
            }
            using (StreamWriter columnsWriter = new StreamWriter(path))
            {
                columnsWriter.WriteLine(columns);
            }
        }

        StreamWriter sw = new StreamWriter(path, true);

        string answers;
        if (!ts)
```

# Slider



Make it SUPER hot!..saving data

3) ...and add DataHolder  
SliderValue to the string that  
will be saved in case of  
timeseries after ,

```
StreamWriter sw = new StreamWriter(path, true);

string answers;
if (!ts)
{
    answers = DataHolder.ID;
    foreach (int answer in DataHolder.Answers)
    {
        answers += "," + answer;
    }
}
else
{
    answers = DateTime.UtcNow.ToString() +
    "." + DateTime.UtcNow.Millisecond.ToString() +
    "," +
    DataHolder.TurnedOn +
    "," +
    DataHolder.SliderValue;
}

sw.WriteLine(answers);
sw.Close();

if (!ts)
```

# Slider

GEM2022

Make it SUPER hot!.saving data

4) to SliderController  
add UnityEngine.Events  
namespace

```
StreamWriter sw = new StreamWriter(path, true);

string answers;
if (!ts)
{
    answers = DataHolder.ID;
    foreach (int answer in DataHolder.Answers)
    {
        answers += "," + answer;
    }
}
else
{
    answers = DateTime.UtcNow.ToString() +
    "." + DateTime.UtcNow.Millisecond.ToString() +
    "," +
    DataHolder.TurnedOn +
    "," +
    DataHolder.SliderValue;
}

sw.WriteLine(answers);
sw.Close();

if (!ts)
```

# Slider

GEM2022

Make it SUPER hot!.saving data

4) to SliderController  
add UnityEngine.Events  
namespace

```
StreamWriter sw = new StreamWriter(path, true);

string answers;
if (!ts)
{
    answers = DataHolder.ID;
    foreach (int answer in DataHolder.Answers)
    {
        answers += "," + answer;
    }
}
else
{
    answers = DateTime.UtcNow.ToString() +
    "." + DateTime.UtcNow.Millisecond.ToString() +
    "," +
    DataHolder.TurnedOn +
    "," +
    DataHolder.SliderValue;
}

sw.WriteLine(answers);
sw.Close();

if (!ts)
```

**Done!**



Mission accomplished!

**Enable FlatUnityCanvas**

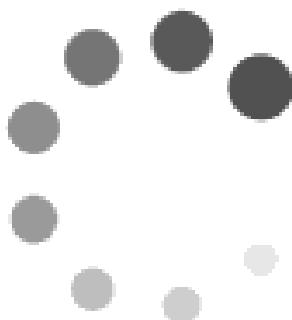
**Build and run**

**Done!**

**GEM2022**

Mission accomplished!

**Any questions?**



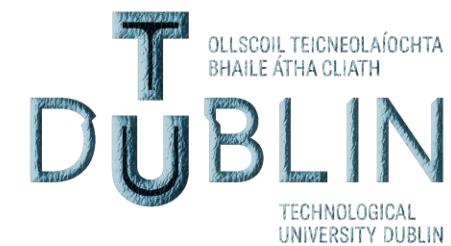
**Done!**



Mission accomplished!

**Thank you,  
feel free to contact me at  
[Krzysztof.X.Szczurowski@mytudublin.ie](mailto:Krzysztof.X.Szczurowski@mytudublin.ie)**

Project files available at:  
<https://github.com/KrisITB/GEM2022tutorial>



# Next?



Final page

**Check if all answers completed?**

**Audio?**

====

**Curved canvas?**

**IK?**

=====

**Web XR?**