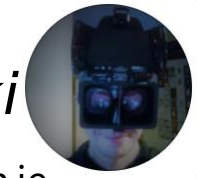
Oculus integration SDK **hand tracking** in **Unity**

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

GEM2022

IEEE

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH

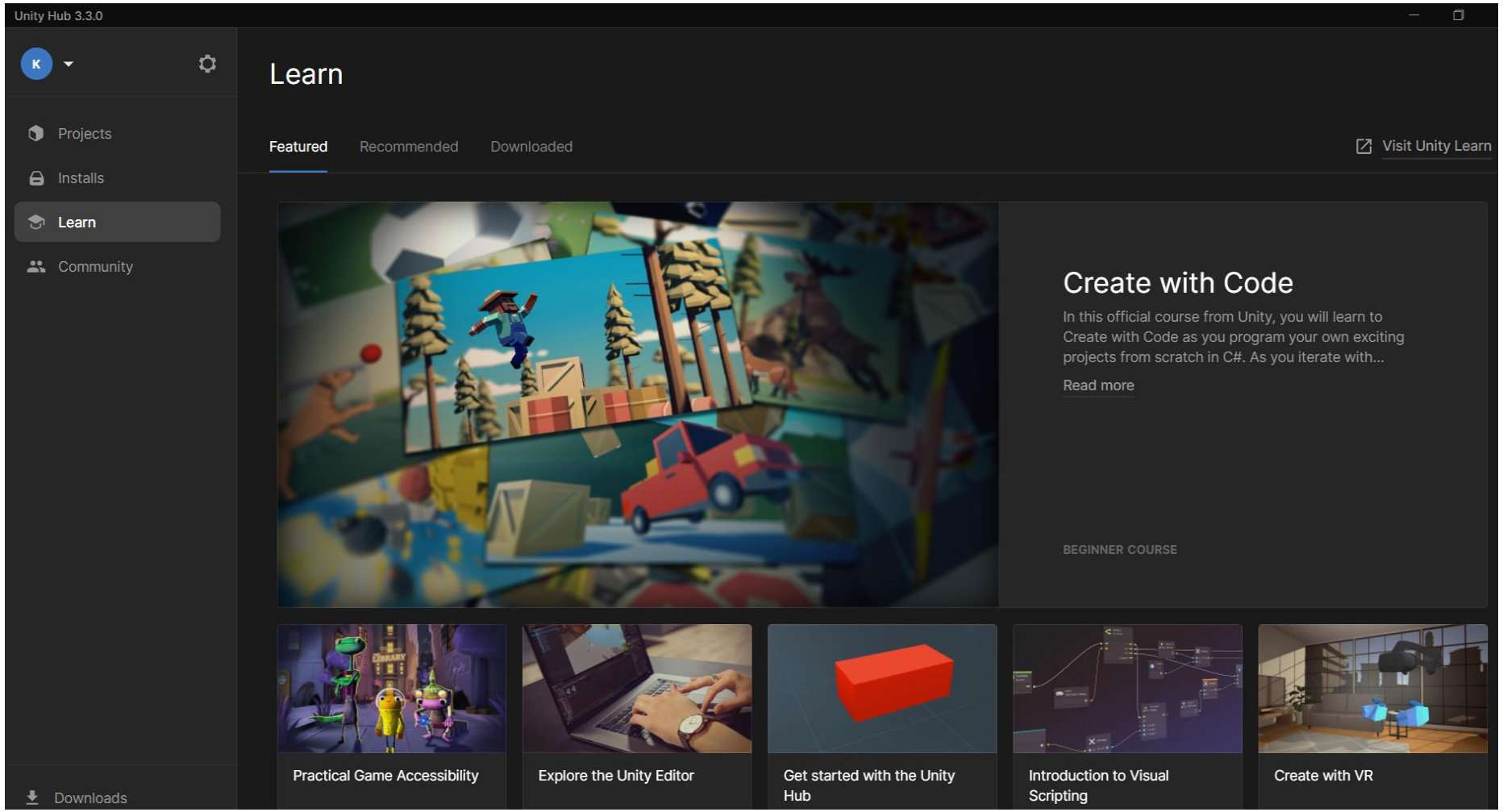
TU DUBLIN

TECHNOLOGICAL UNIVERSITY DUBLIN

Setup

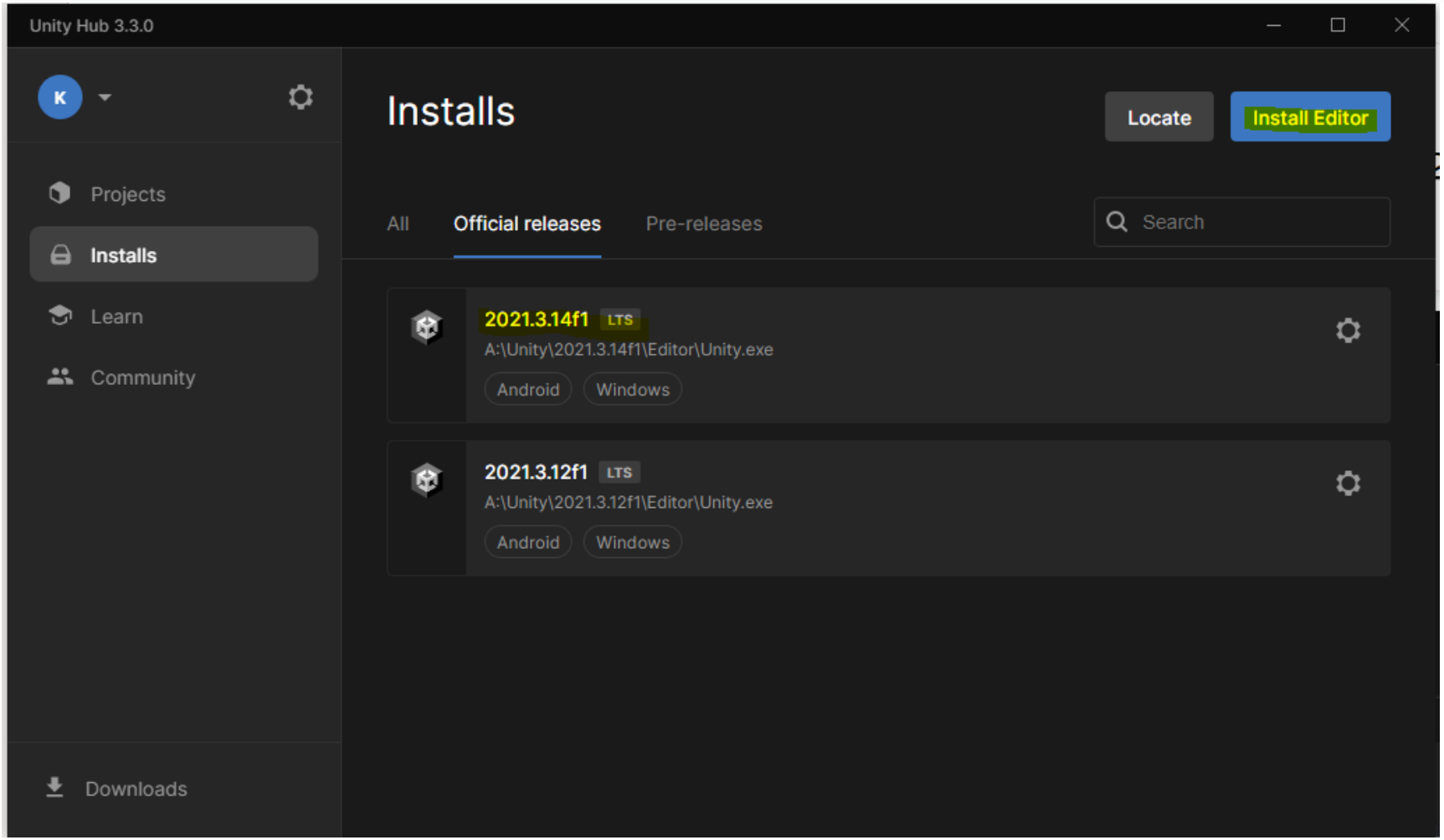
GEM2022

Unity Hub 3.3.0



Setup

Unity 2021.3.14f1 (LTS)

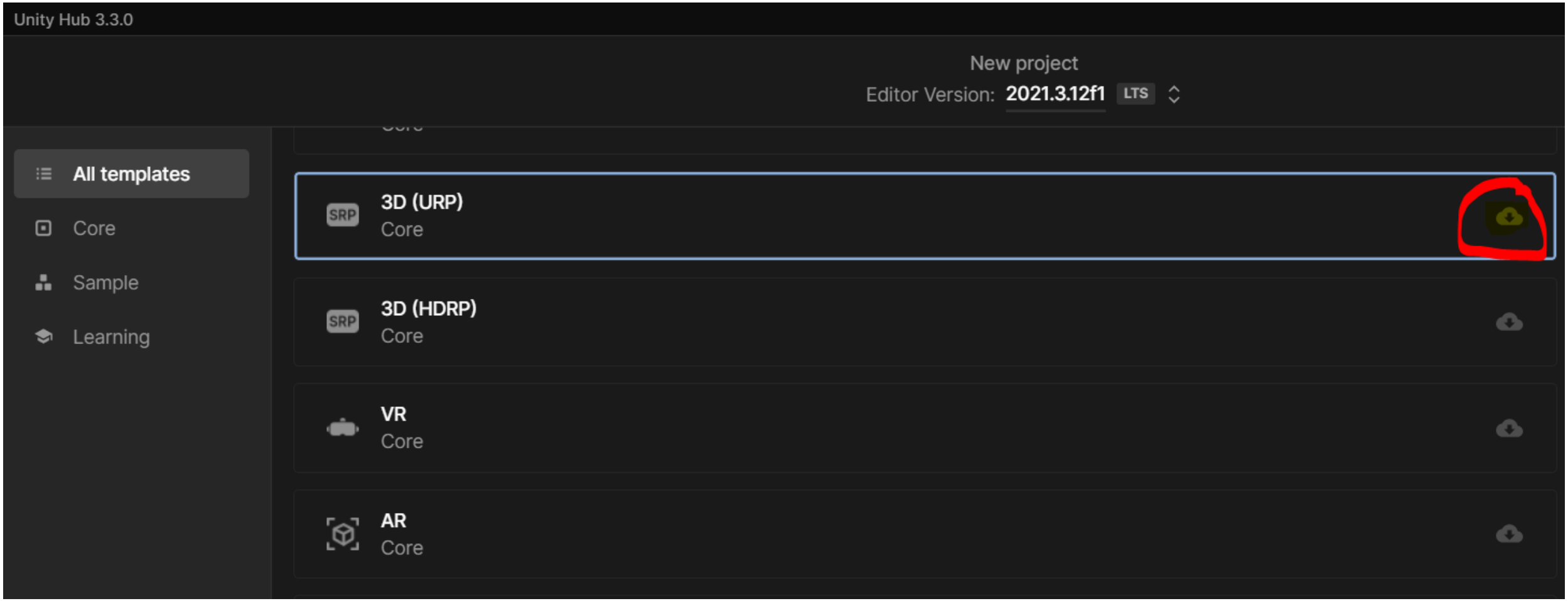


Setup

GEM2022

3D URP

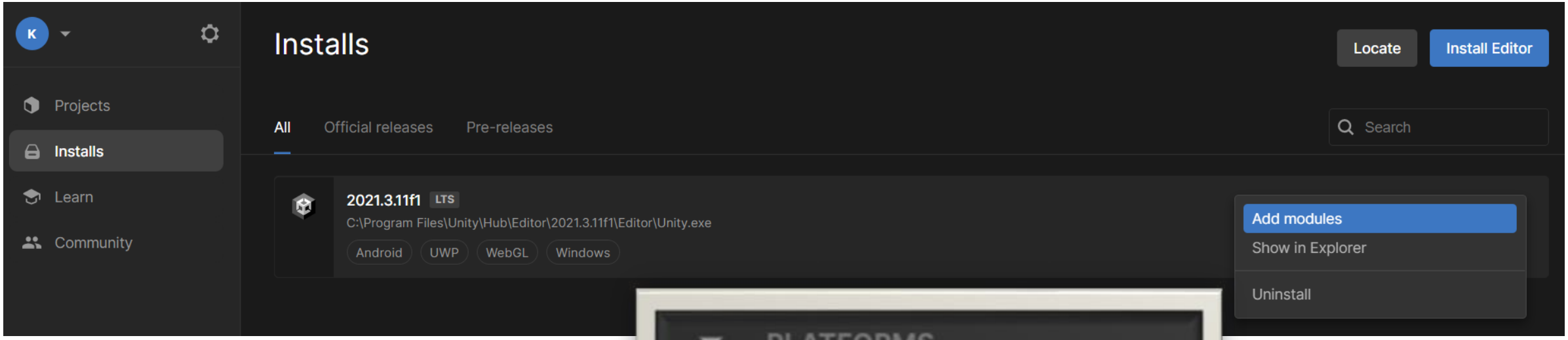
(Universal Render Pipeline)

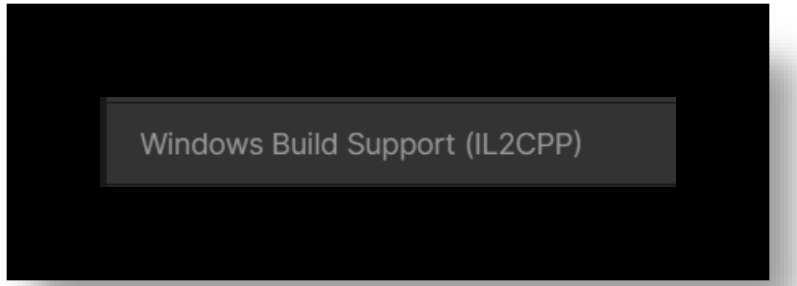


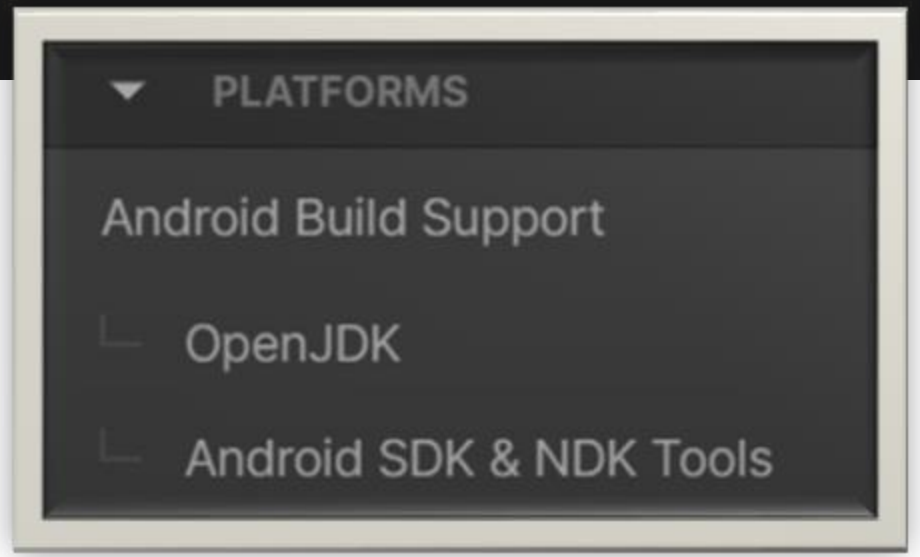
Setup

GEM 2022

Modules

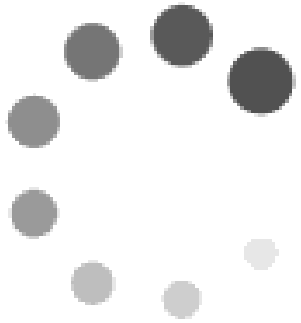






Setup

Questions?

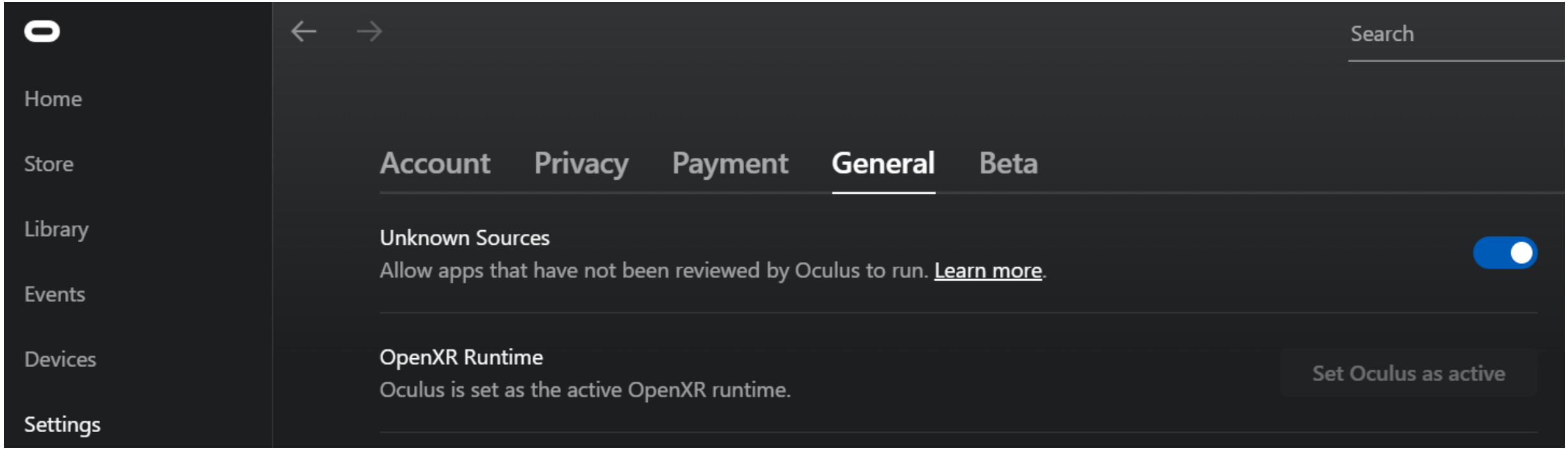


Setup

GEM2022

Oculus application (PC)

Settings

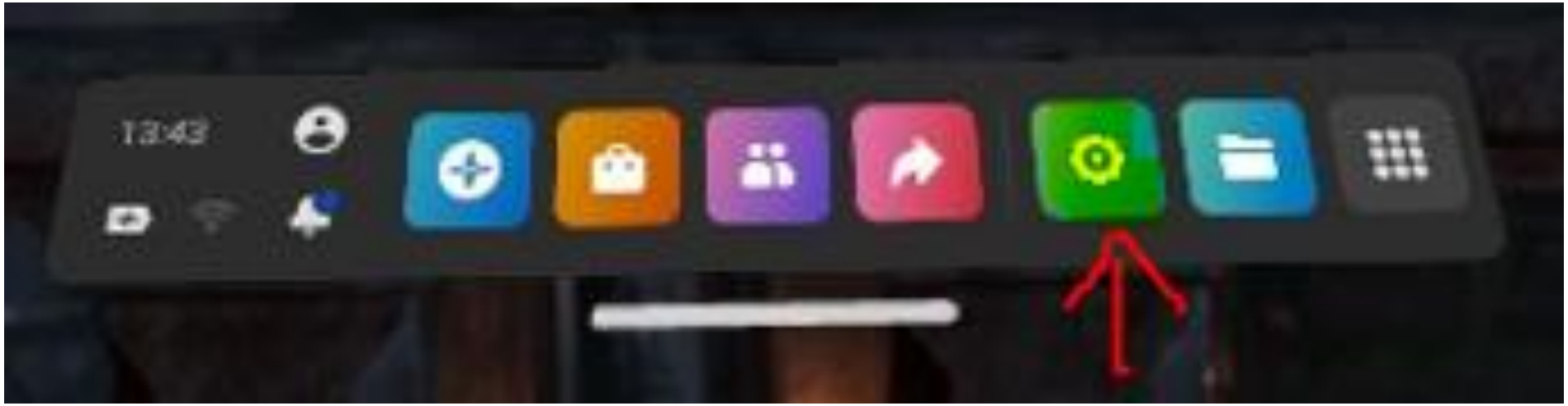


Setup

GEM2022

Oculus (HMD)

Settings



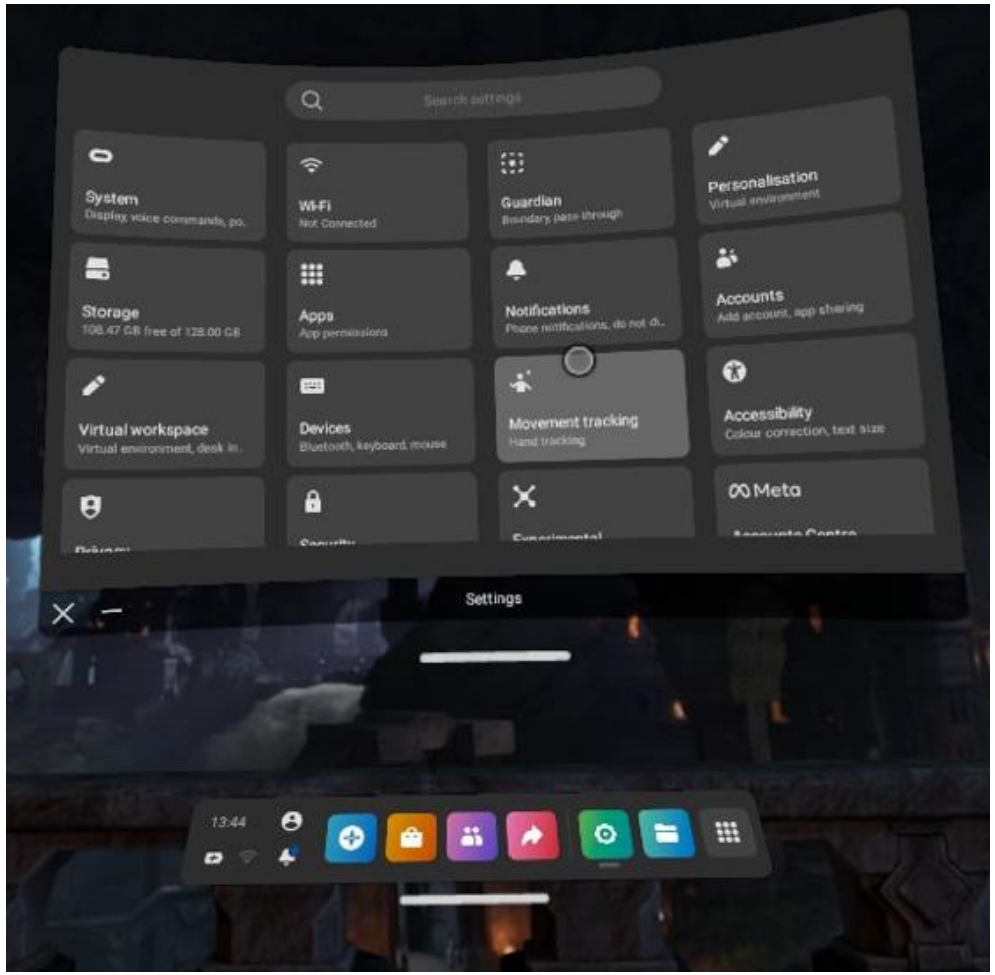
Setup

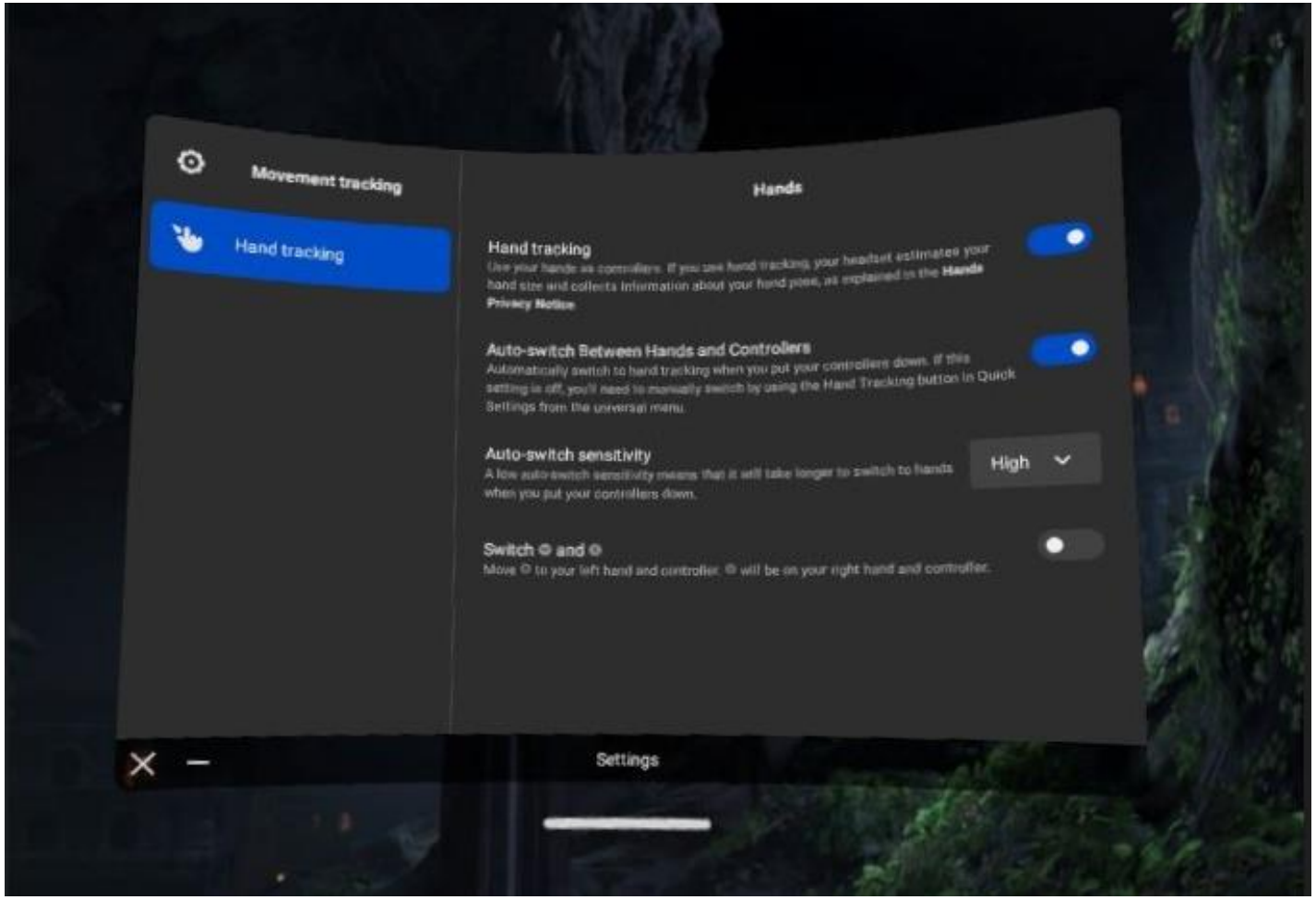
GEM2022

Oculus (HMD)

Settings

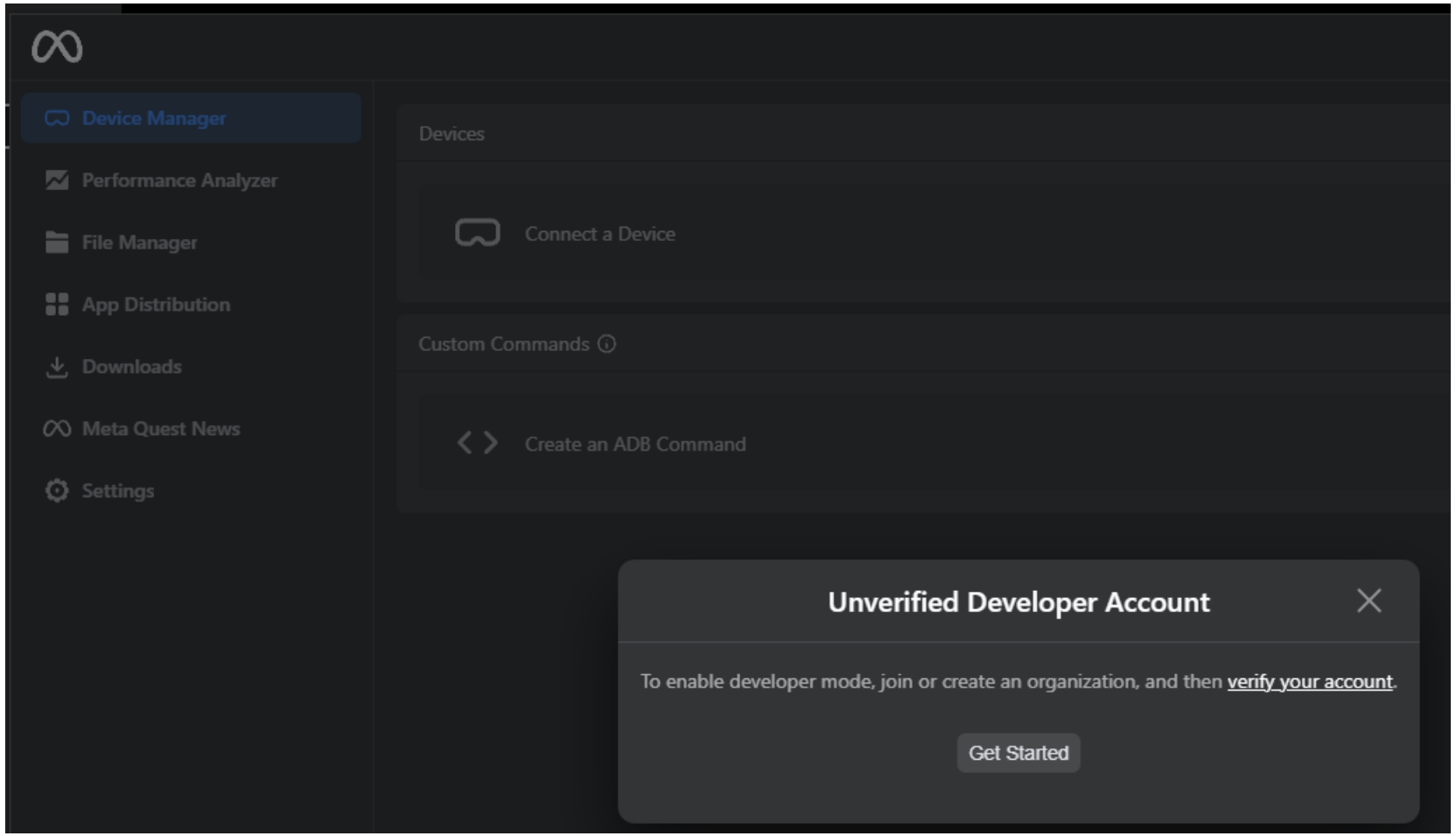
Movement tracking -> Hand tracking





Setup

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



https://developer.oculus.com/
documentation/unity/ts-odh/

Setup

developer.oculus.com/manage/organizations/create/

Meta Quest

Org Manager

Select Organization

Shortcuts Edit

Members Management - TU D...

Create An Organization

Create New Organization

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.

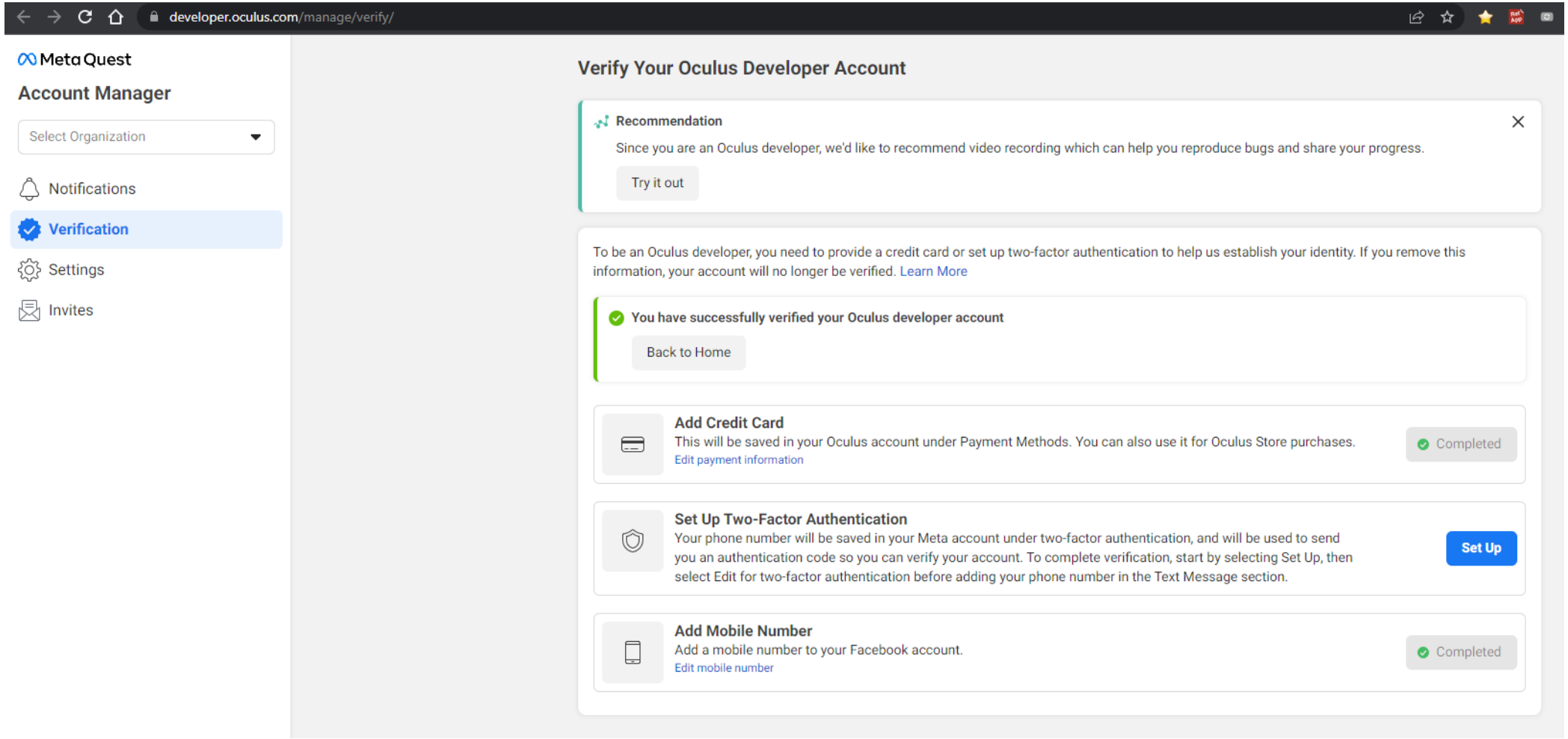
I understand

Required

Submit

Setup

developer.oculus.com/manage/verify/



Setup

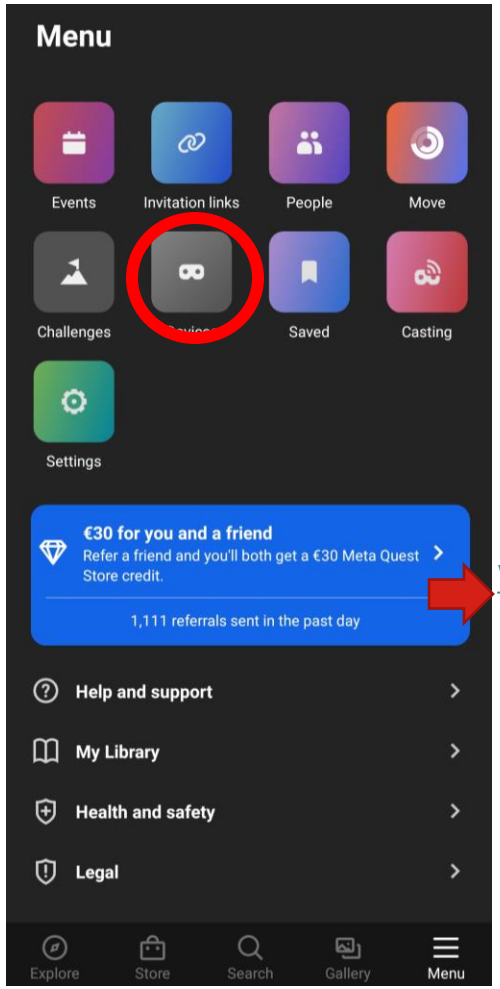
GEM2022

Questions?

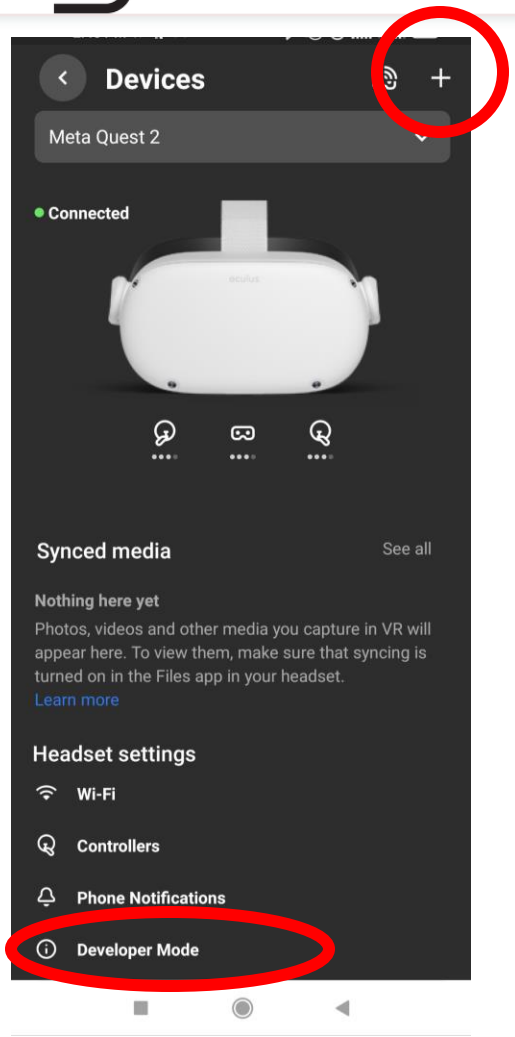
Setup

GEM2022

Oculus app



www.oculus.com/referrals/link/KrisITB/

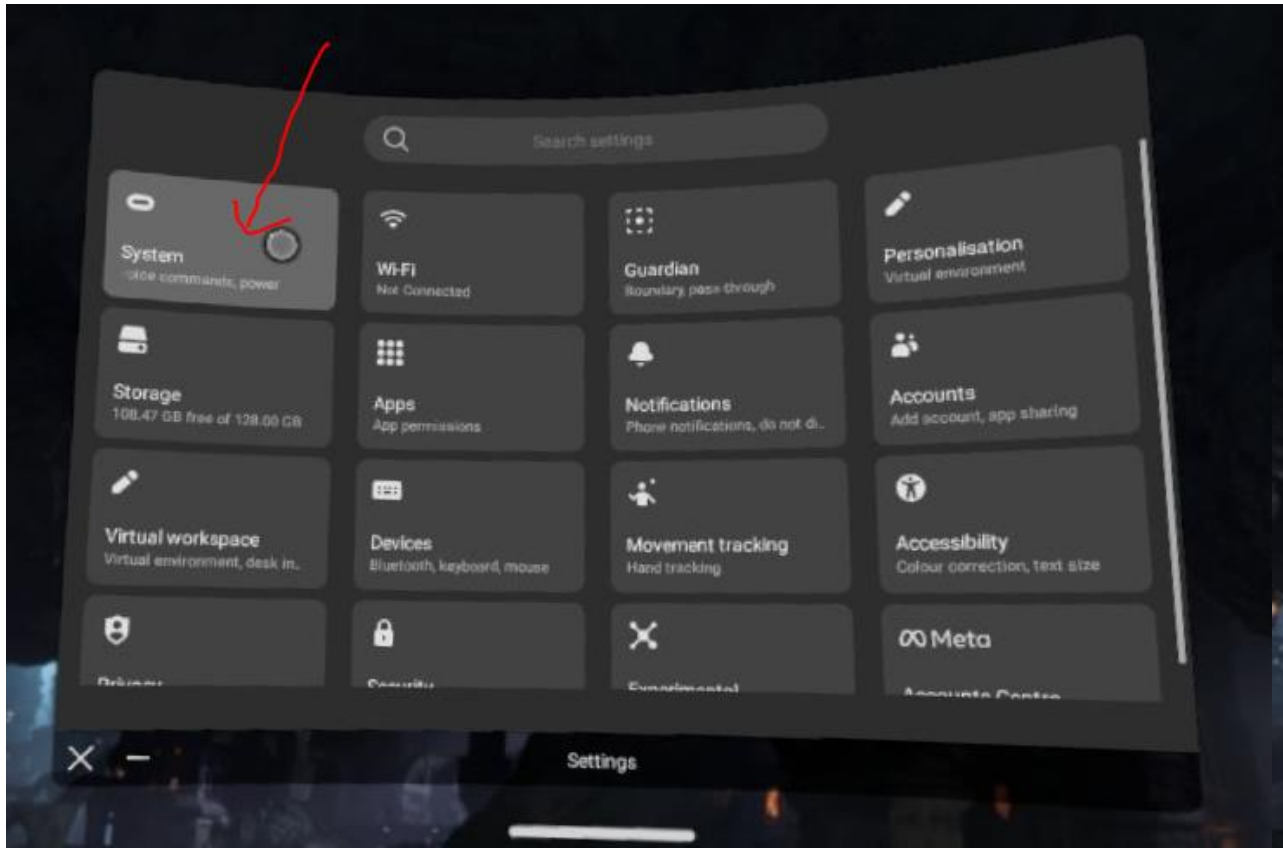


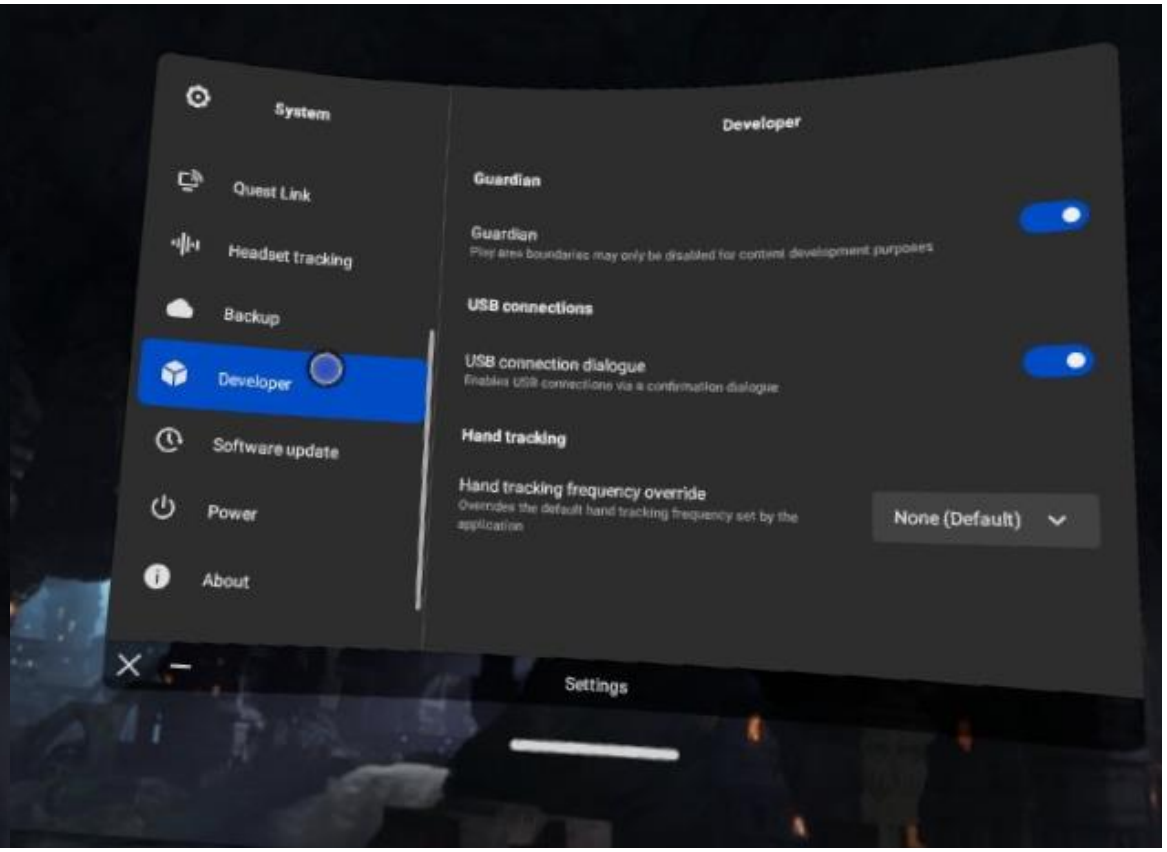
Setup

Oculus (HMD)

Settings

System -> Developer

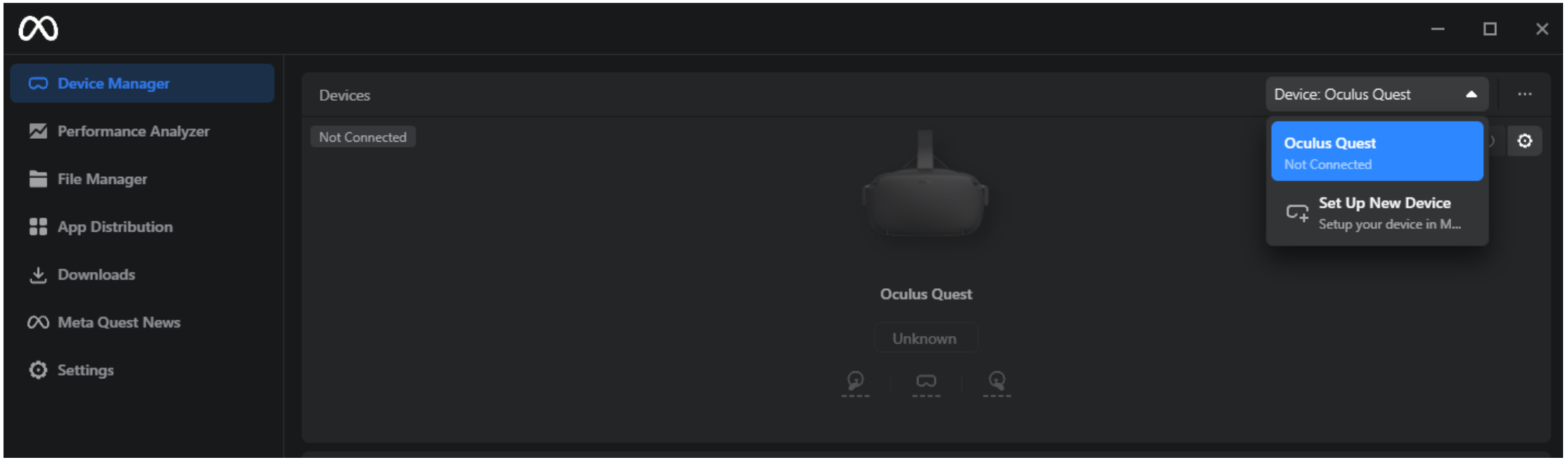




Setup

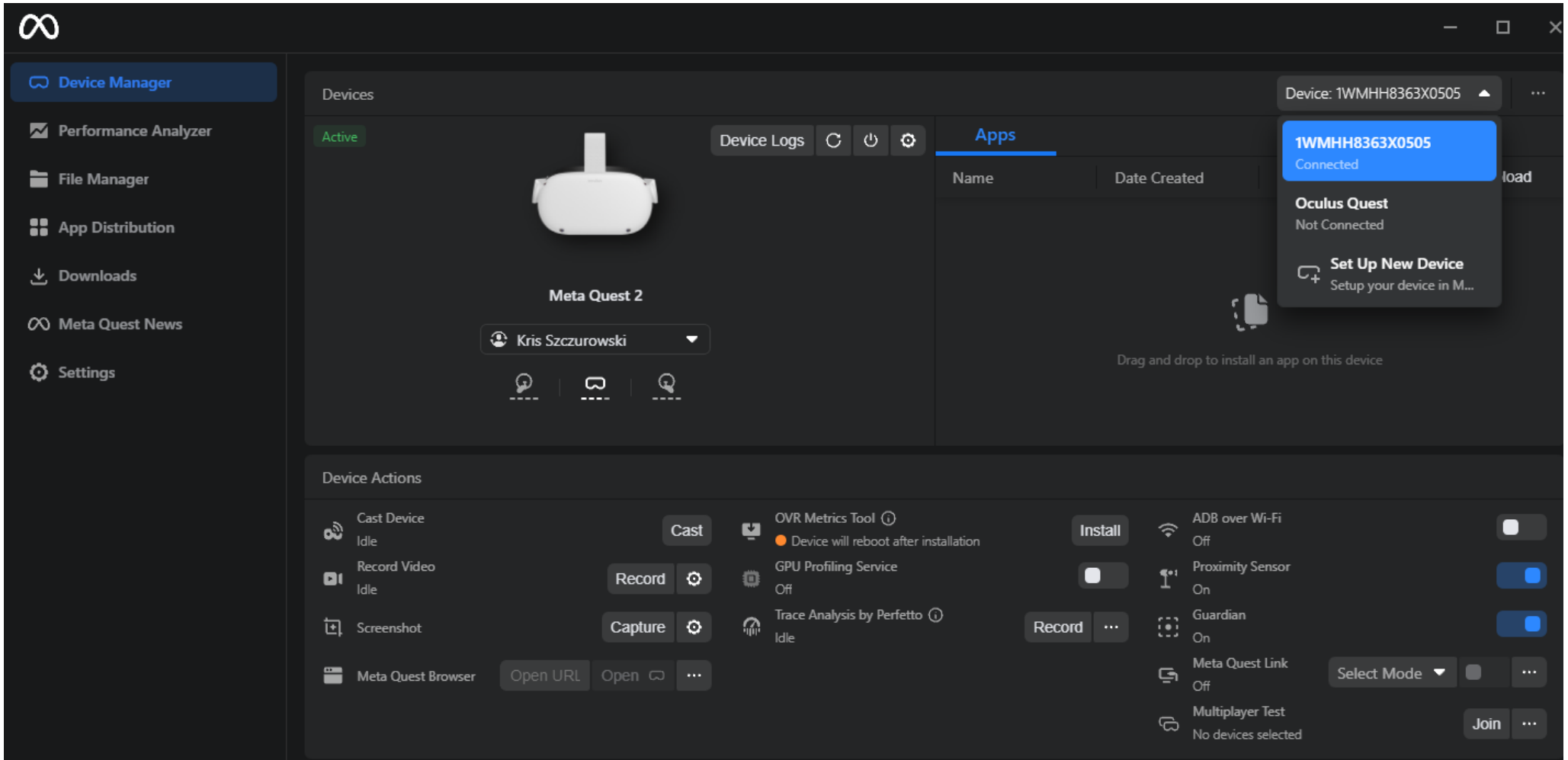
GEM2022

Meta Quest Developer Hub



Setup

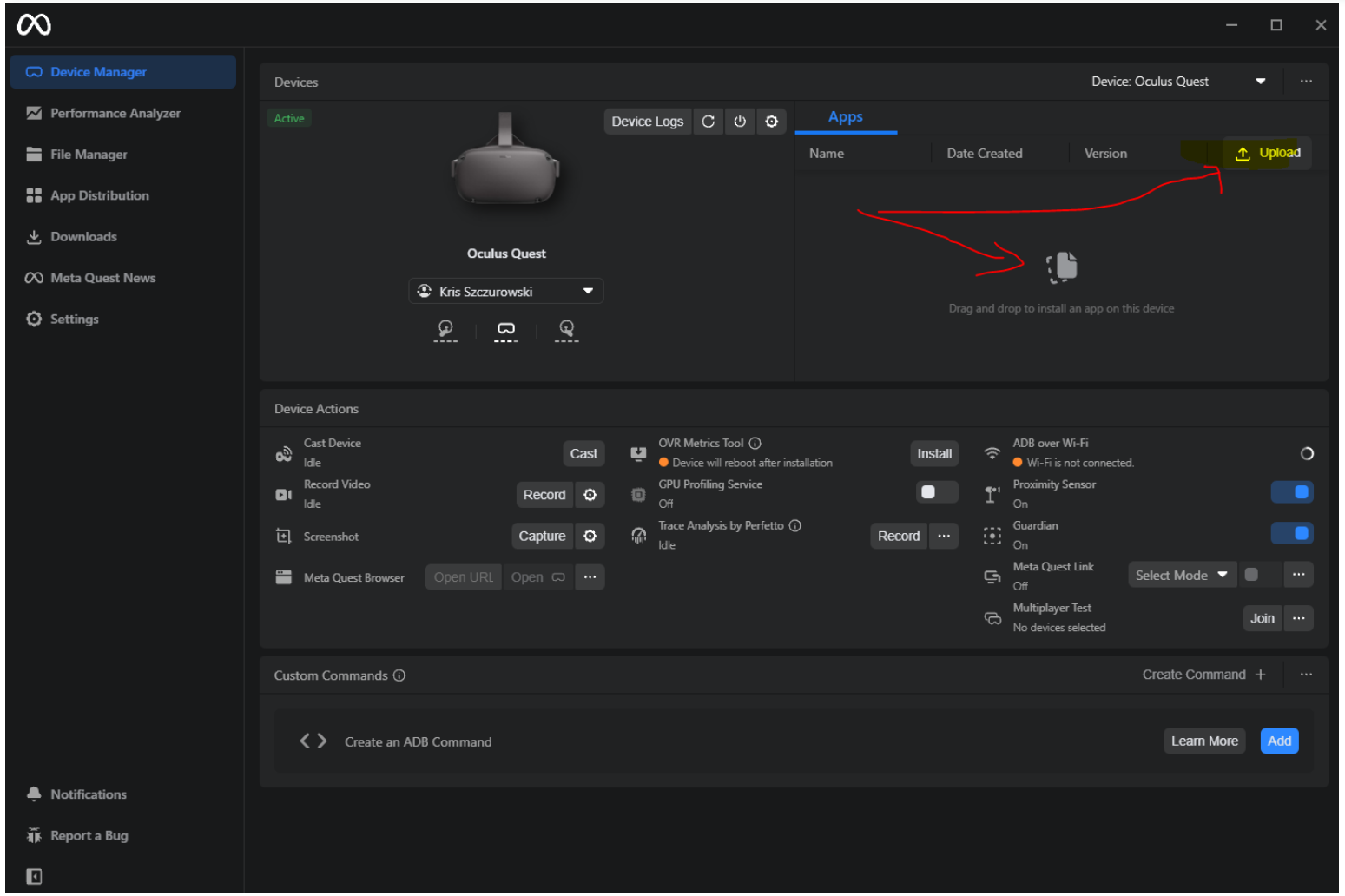
Meta Quest Developer Hub



Setup

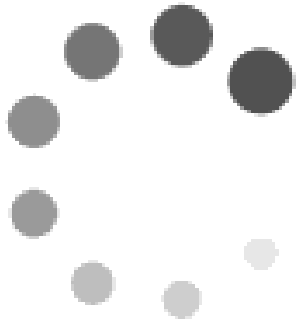
GEM2022

Meta Quest Developer Hub



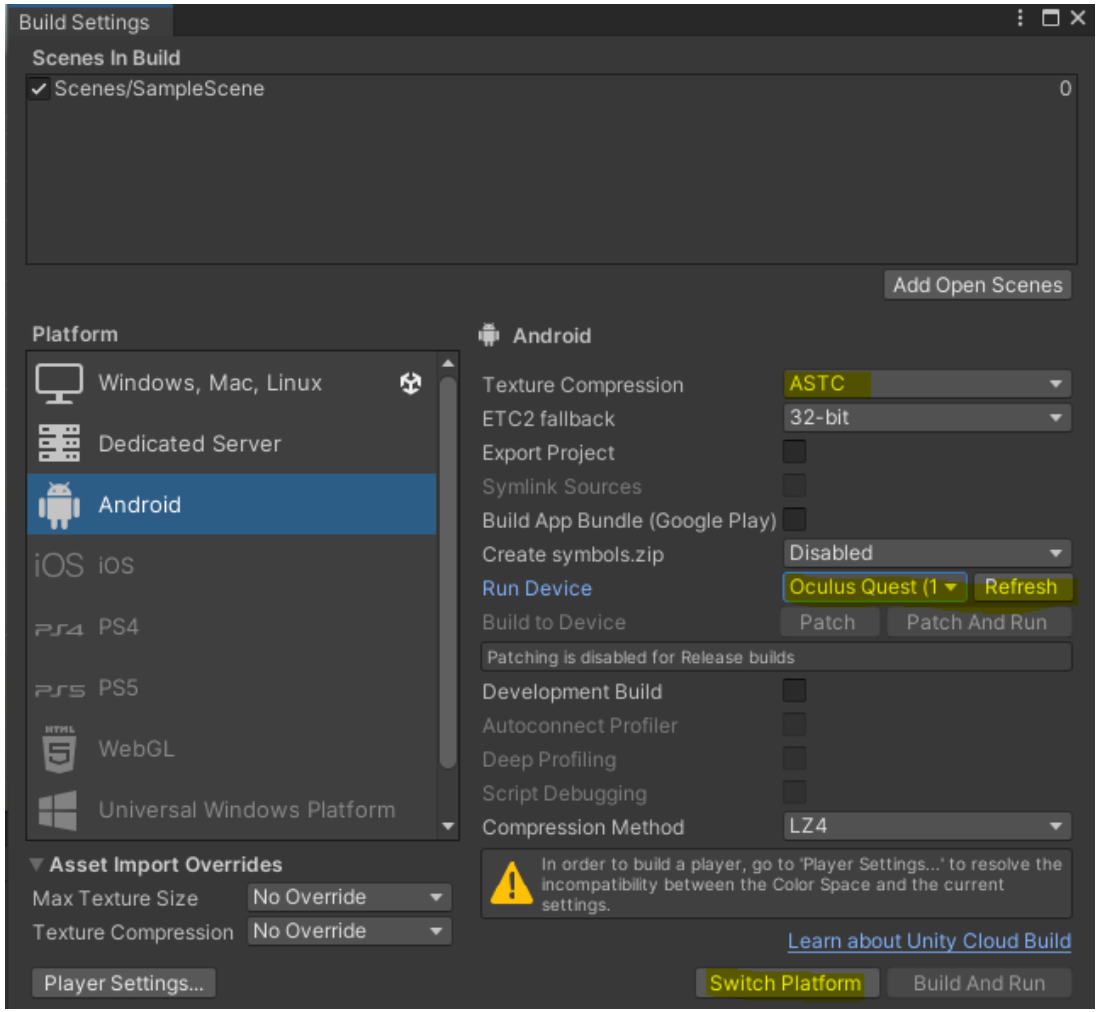
Setup

Questions?



Setup

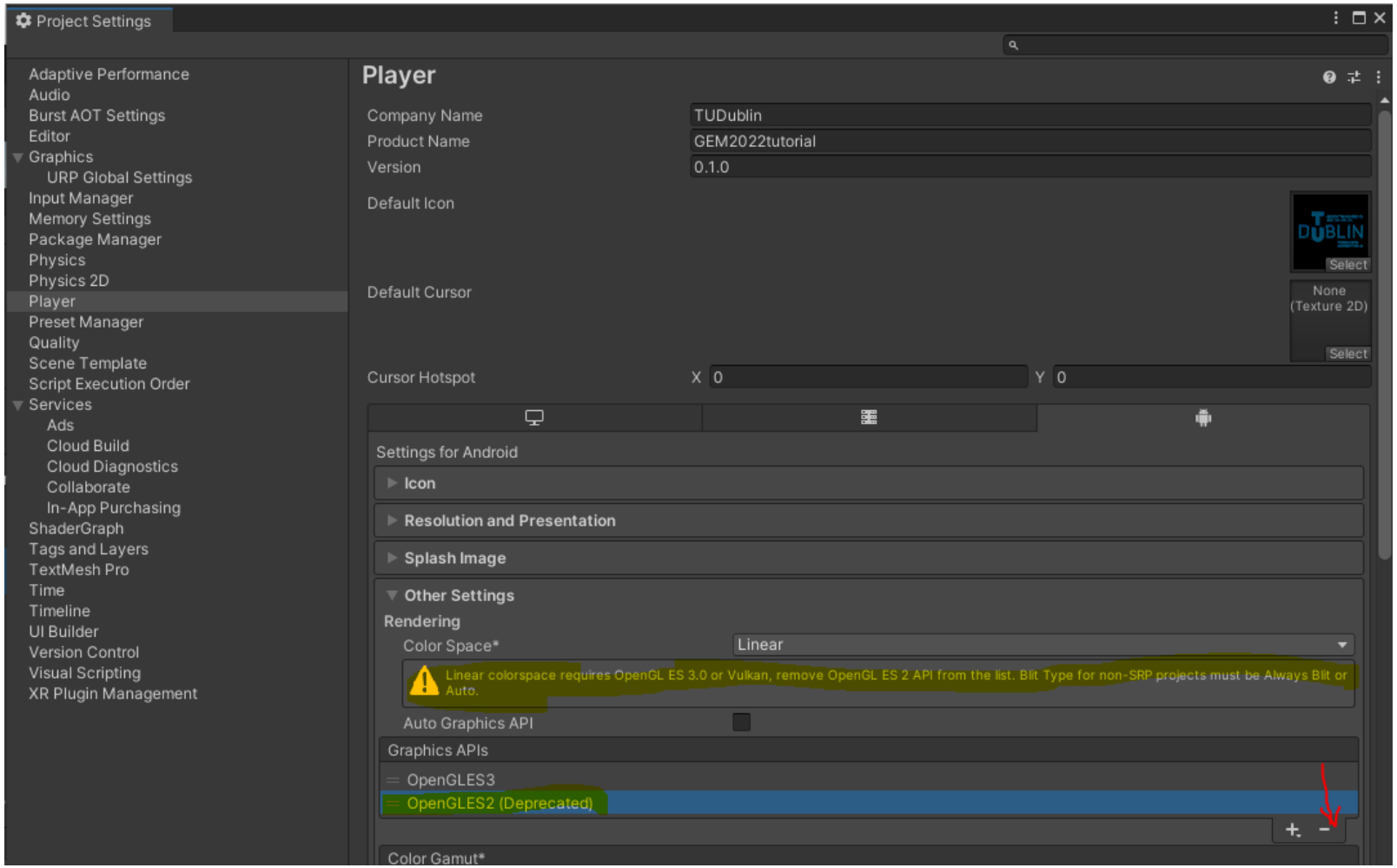
File-> Build Settings...



Setup

Edit -> Project Settings...

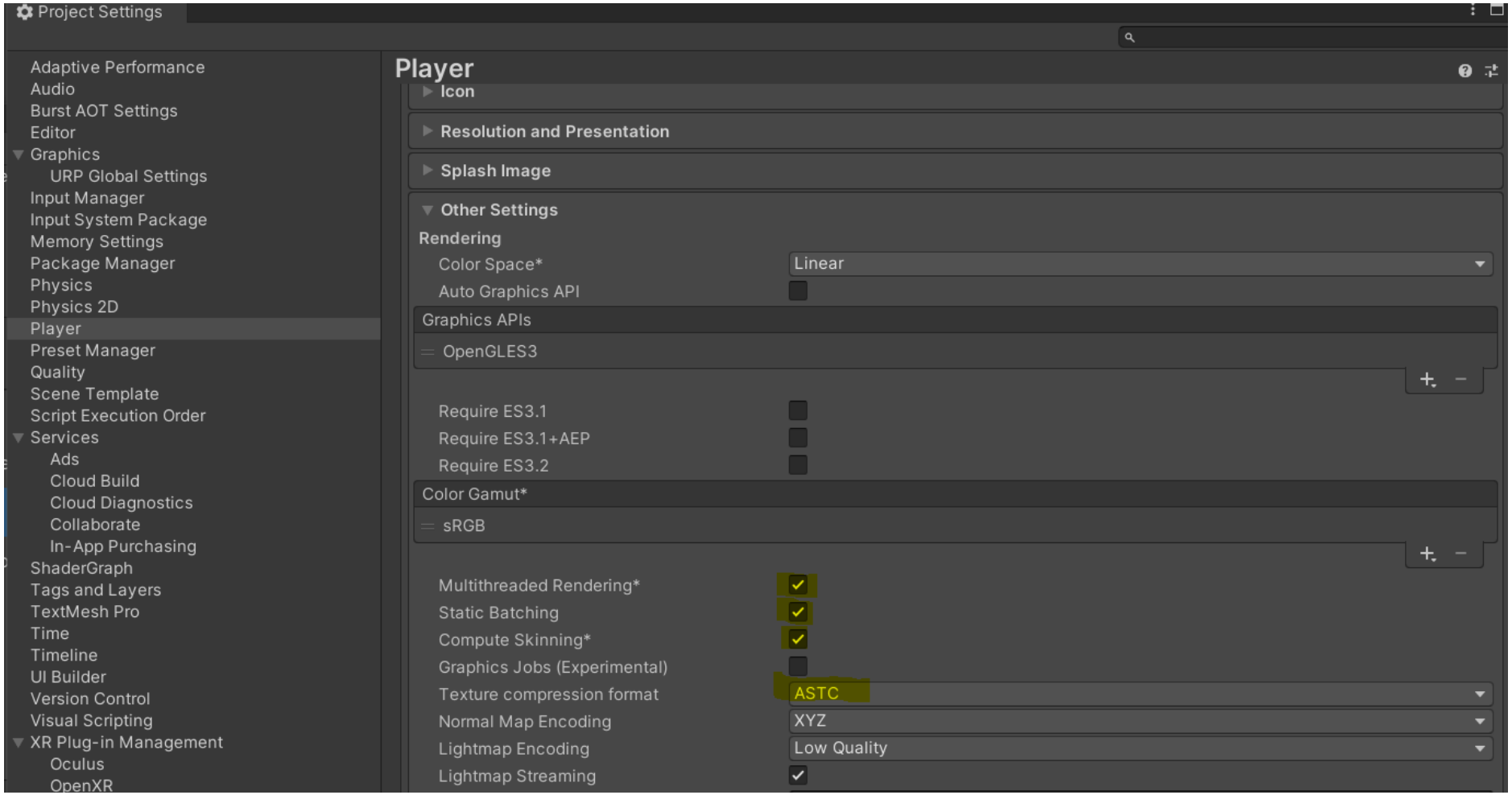
Player



Setup

File-> Build Settings...

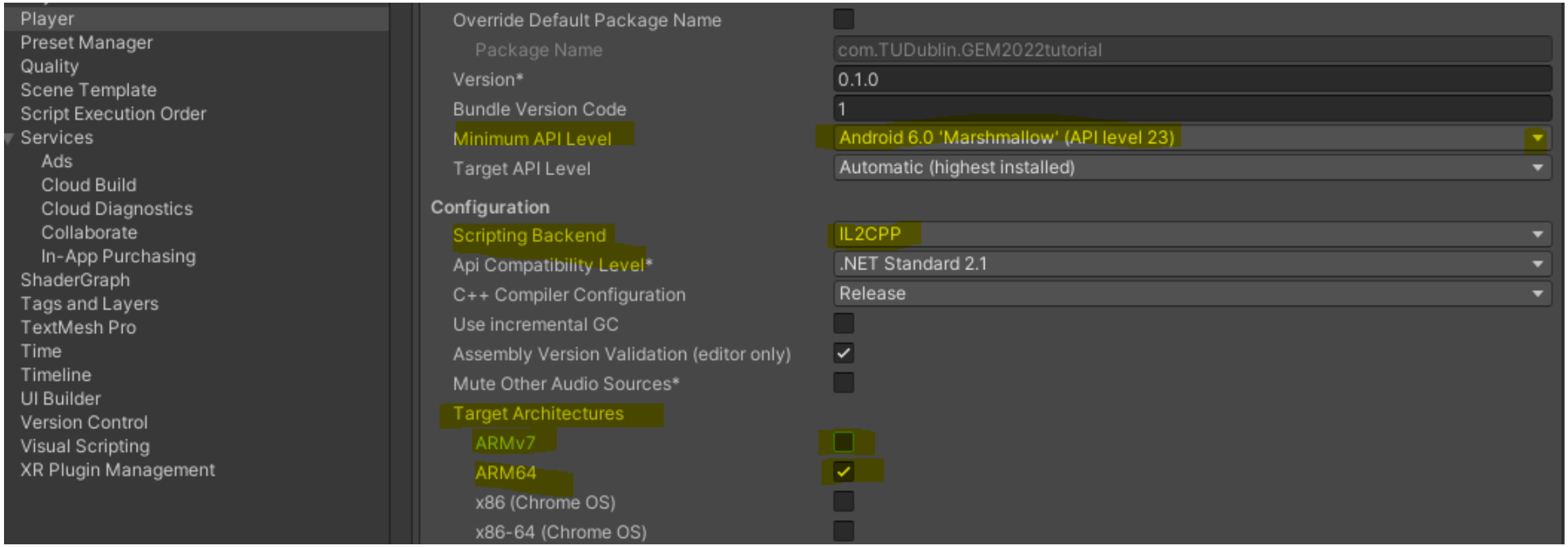
Player



Setup

Edit -> Project Settings...

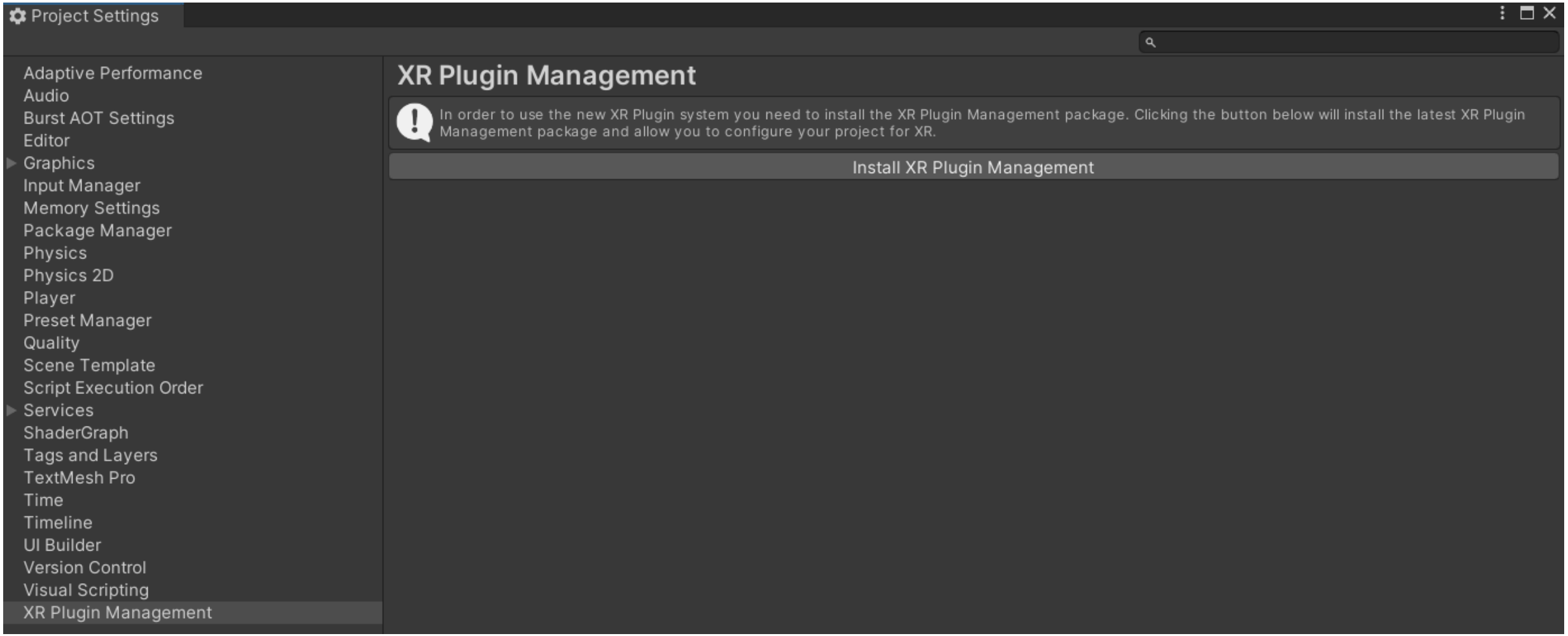
Player



Setup

Edit -> Project Settings...

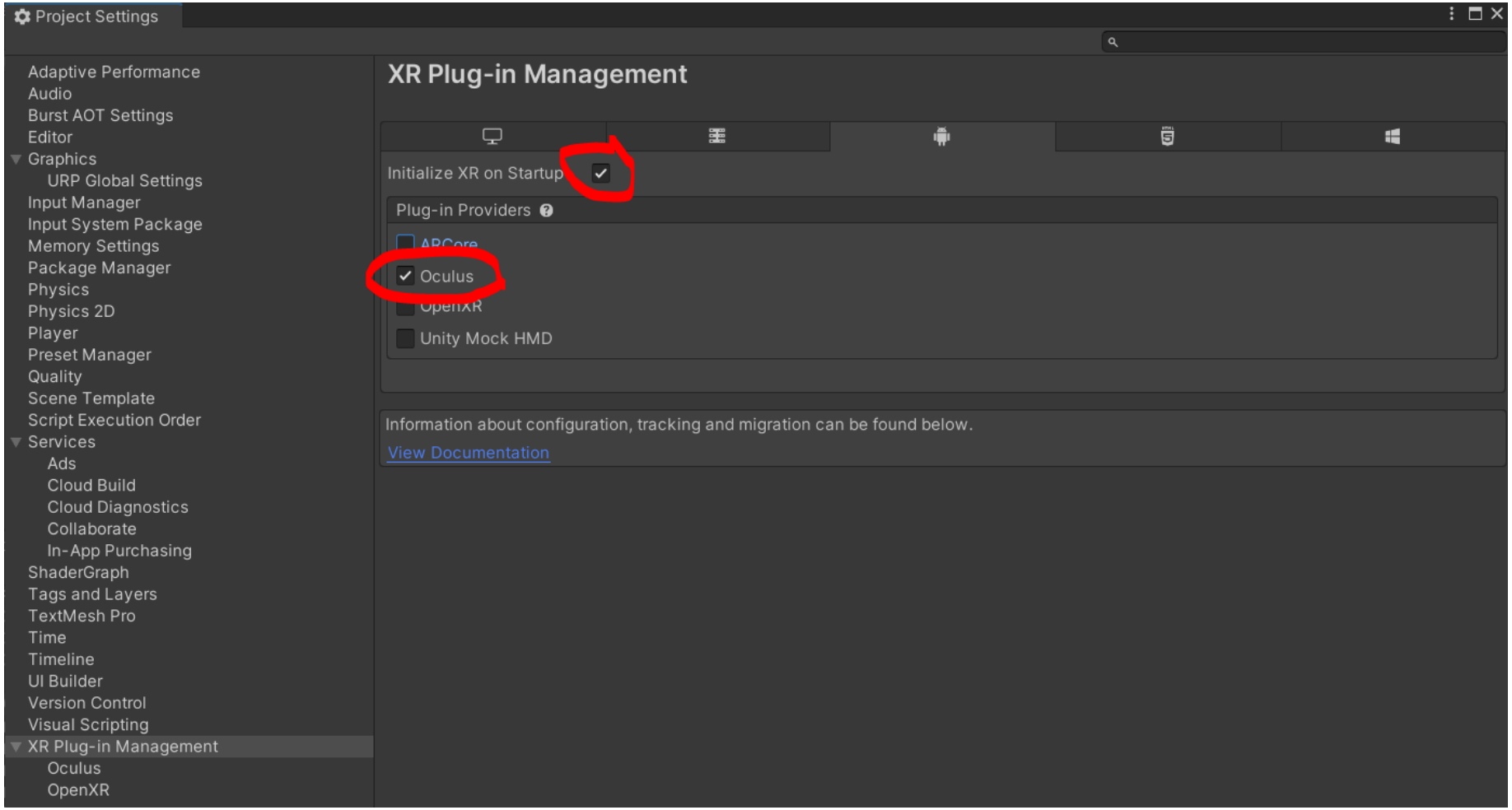
XR Plug-in Management



Setup

Edit -> Project Settings...

XR Plug-in Management

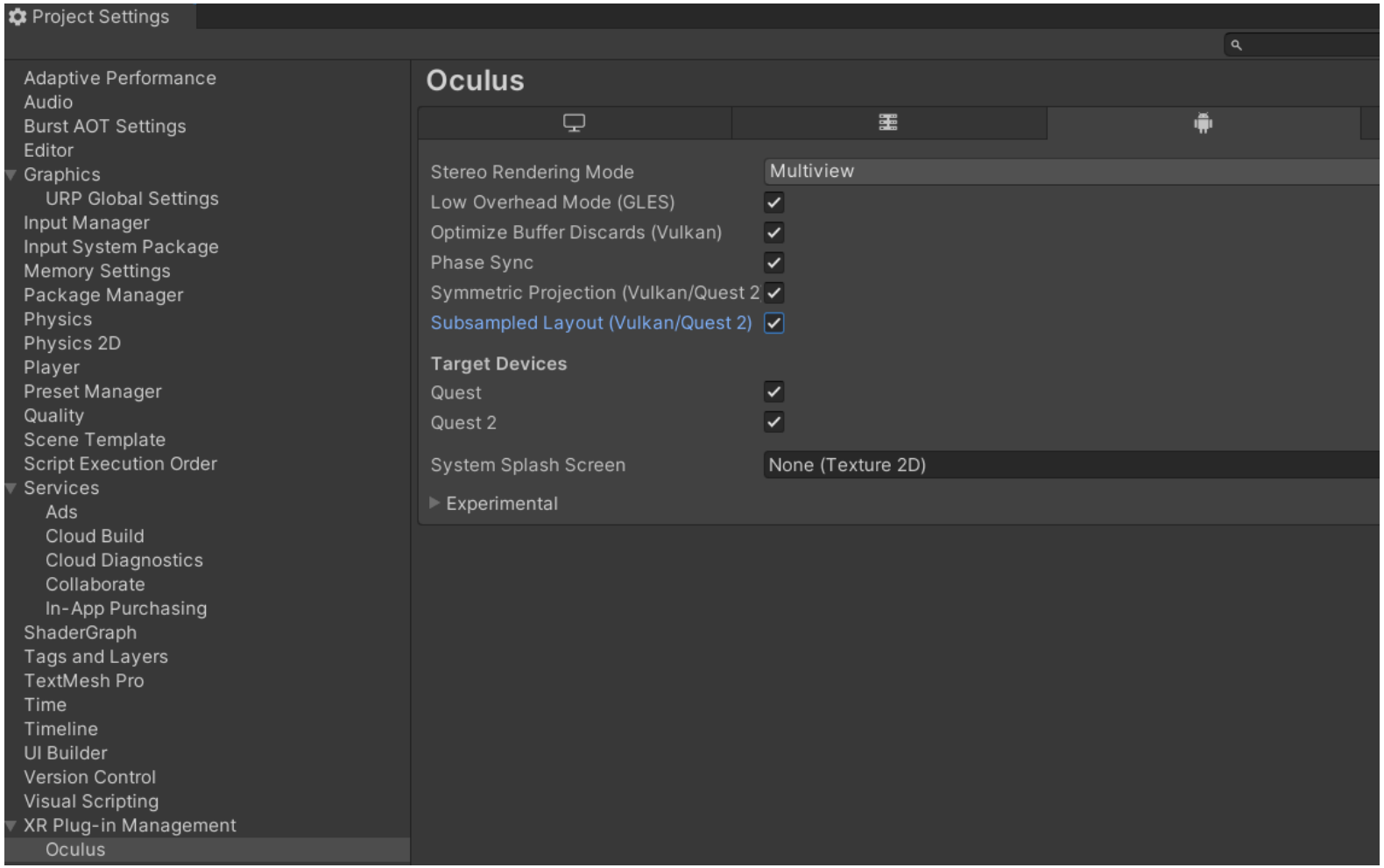


Setup

Edit -> Project Settings...

XR Plug-in Management

Oculus -> Android



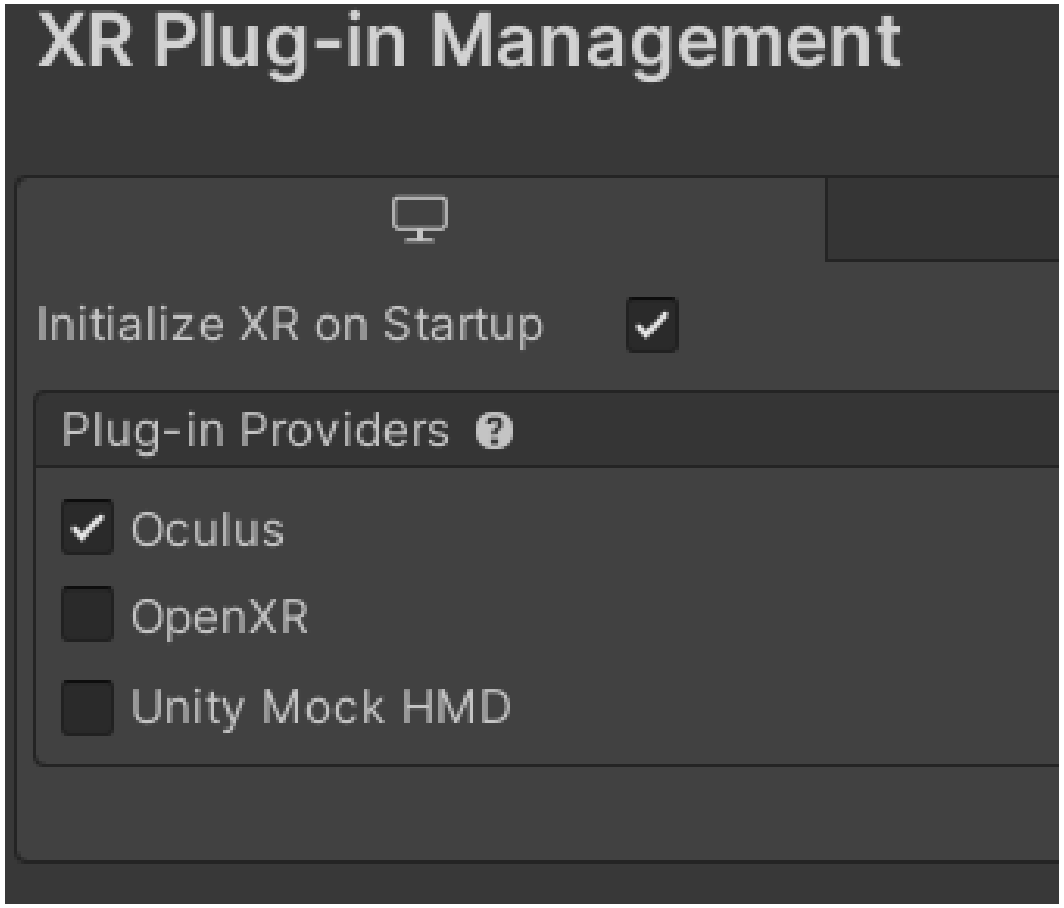
Setup

Edit -> Project Settings...

XR Plug-in Management

PC

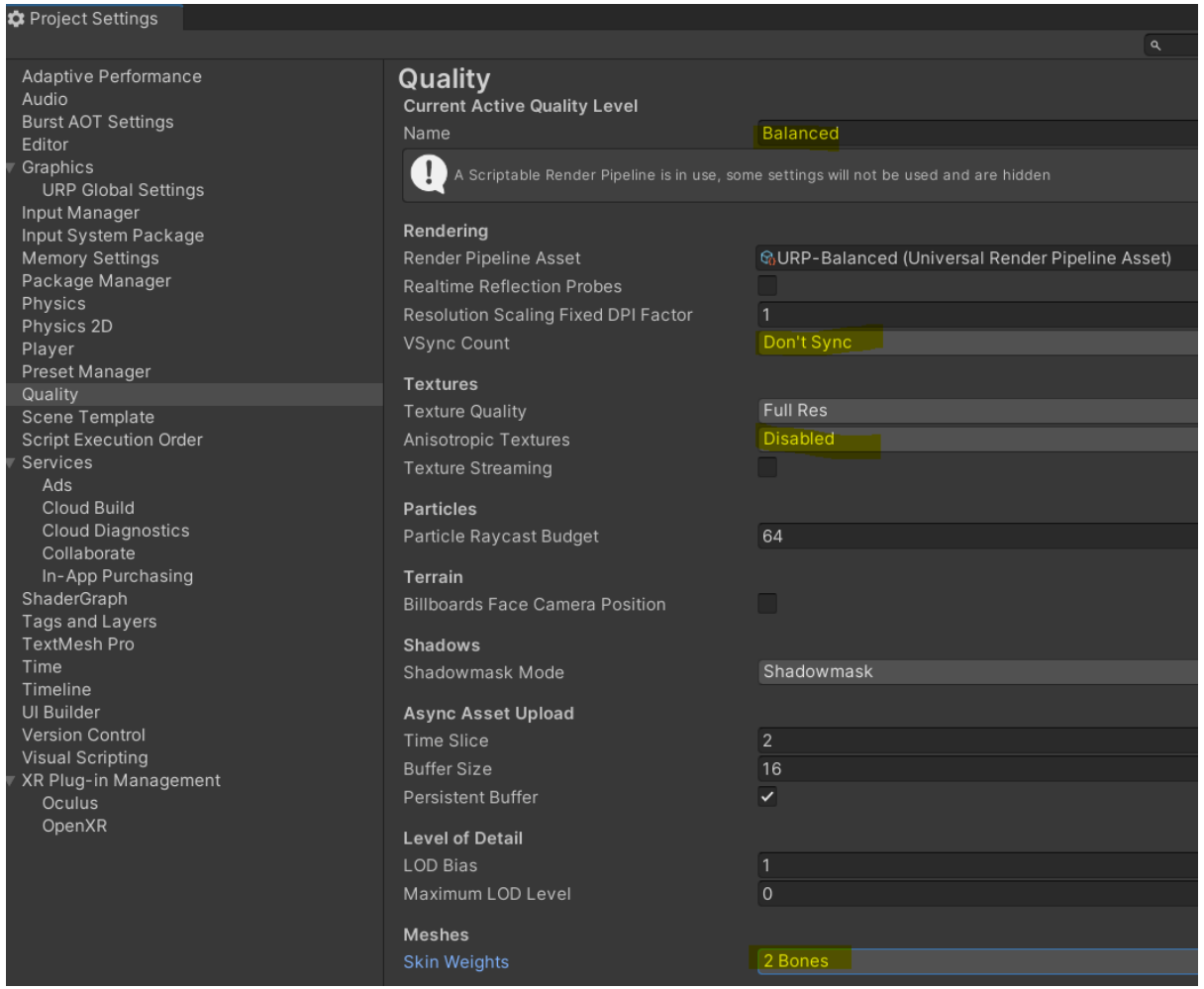
For preview:



Setup

Edit -> Project Settings...

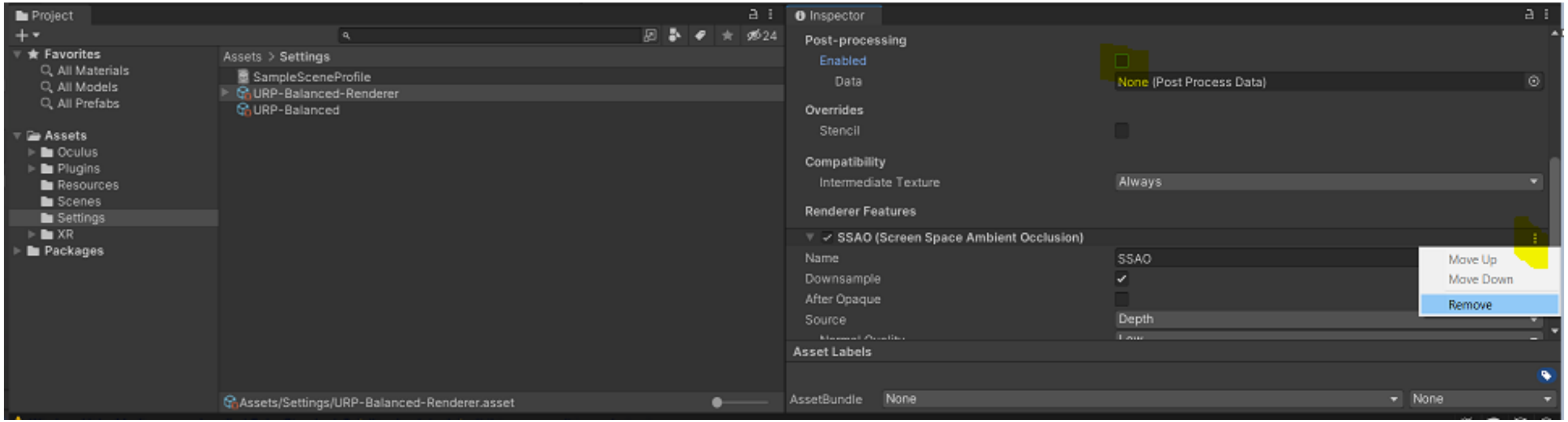
Quality



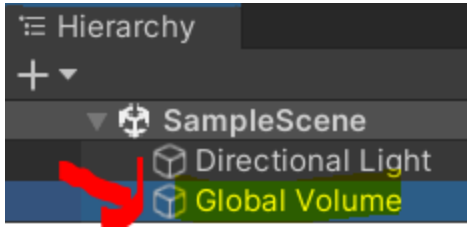
Setup

Assets -> Settings

URP-Balanced-Renderer



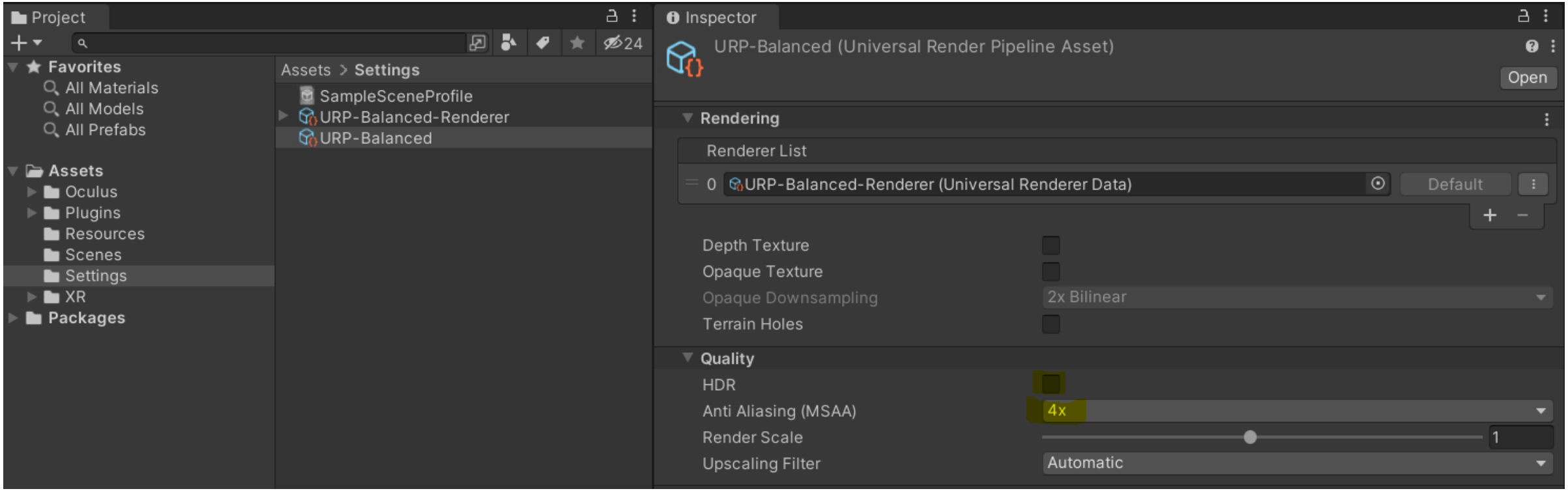
If no post-processing, remove:



Setup

Assets -> Settings

URP-Balanced

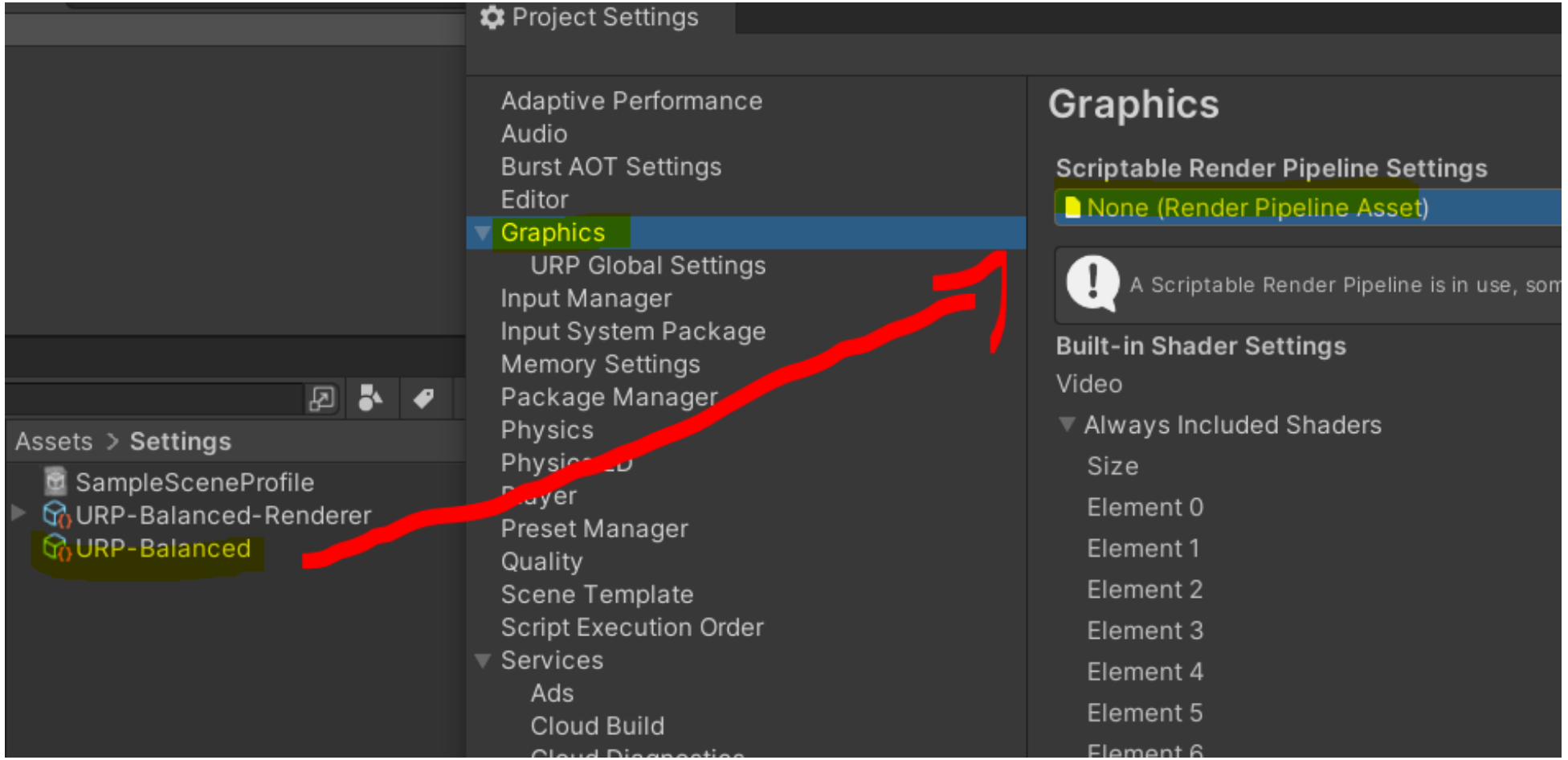


Setup

GEM2022

Project Settings - Graphics

Assets -> Settings

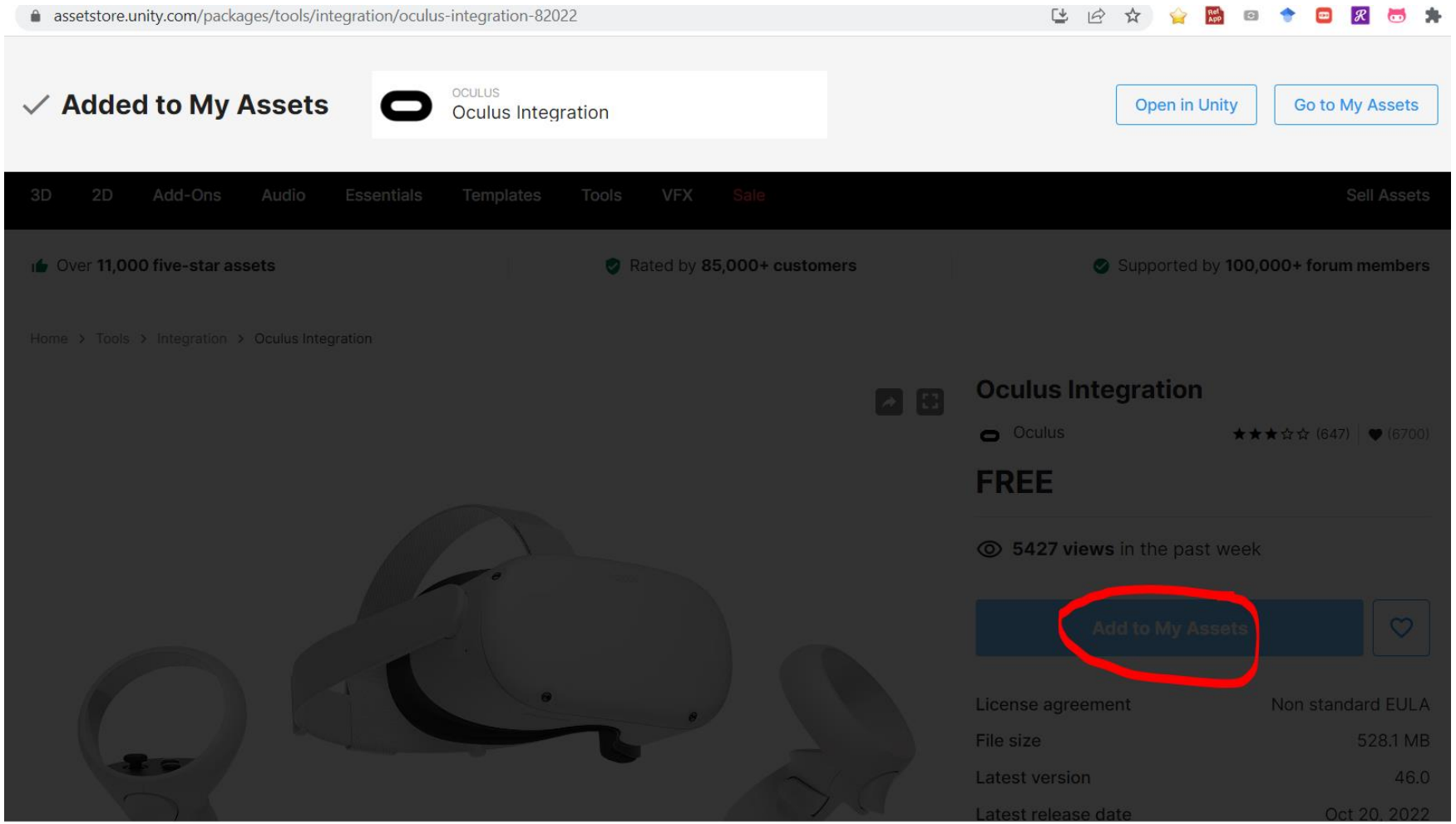


Setup

GEM 2022

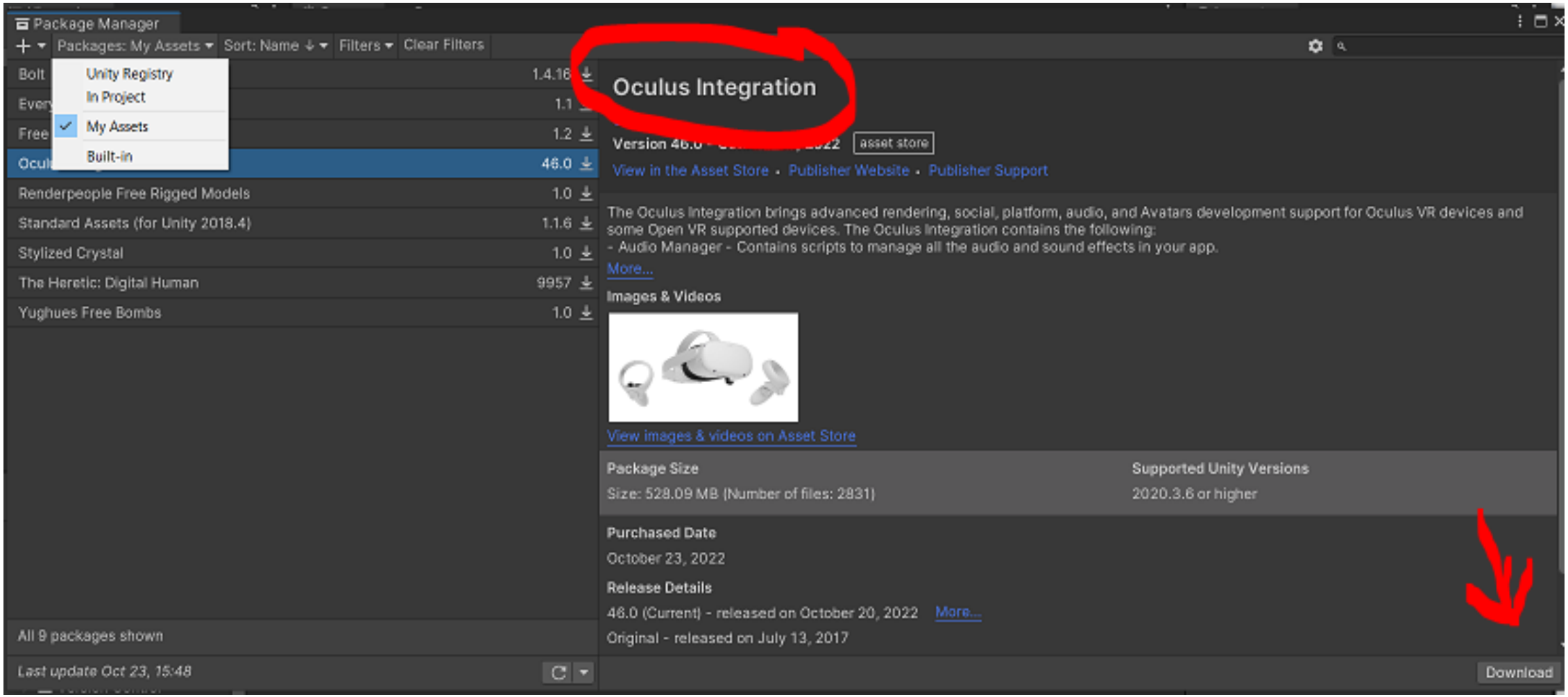
assetstore.unity.com

Oculus Integration Package



Setup

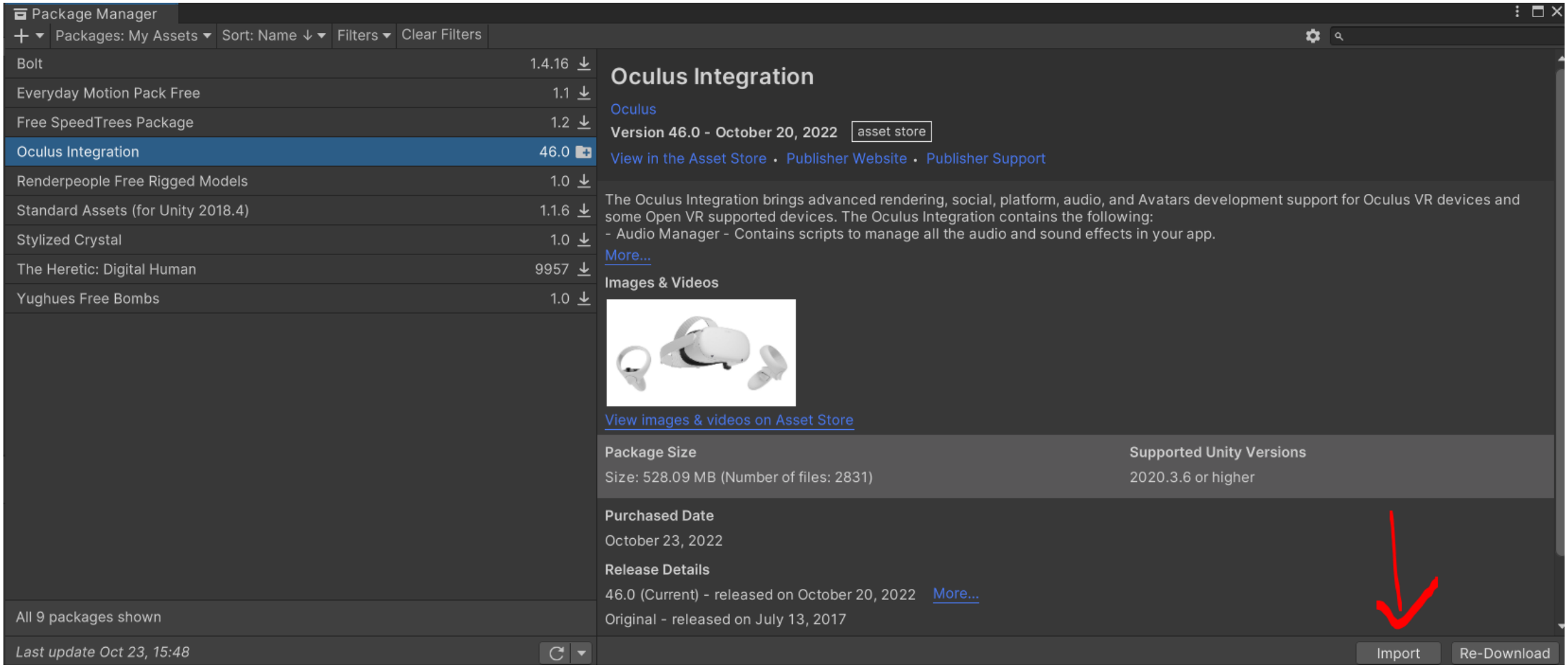
Window -> Package Manager



Setup

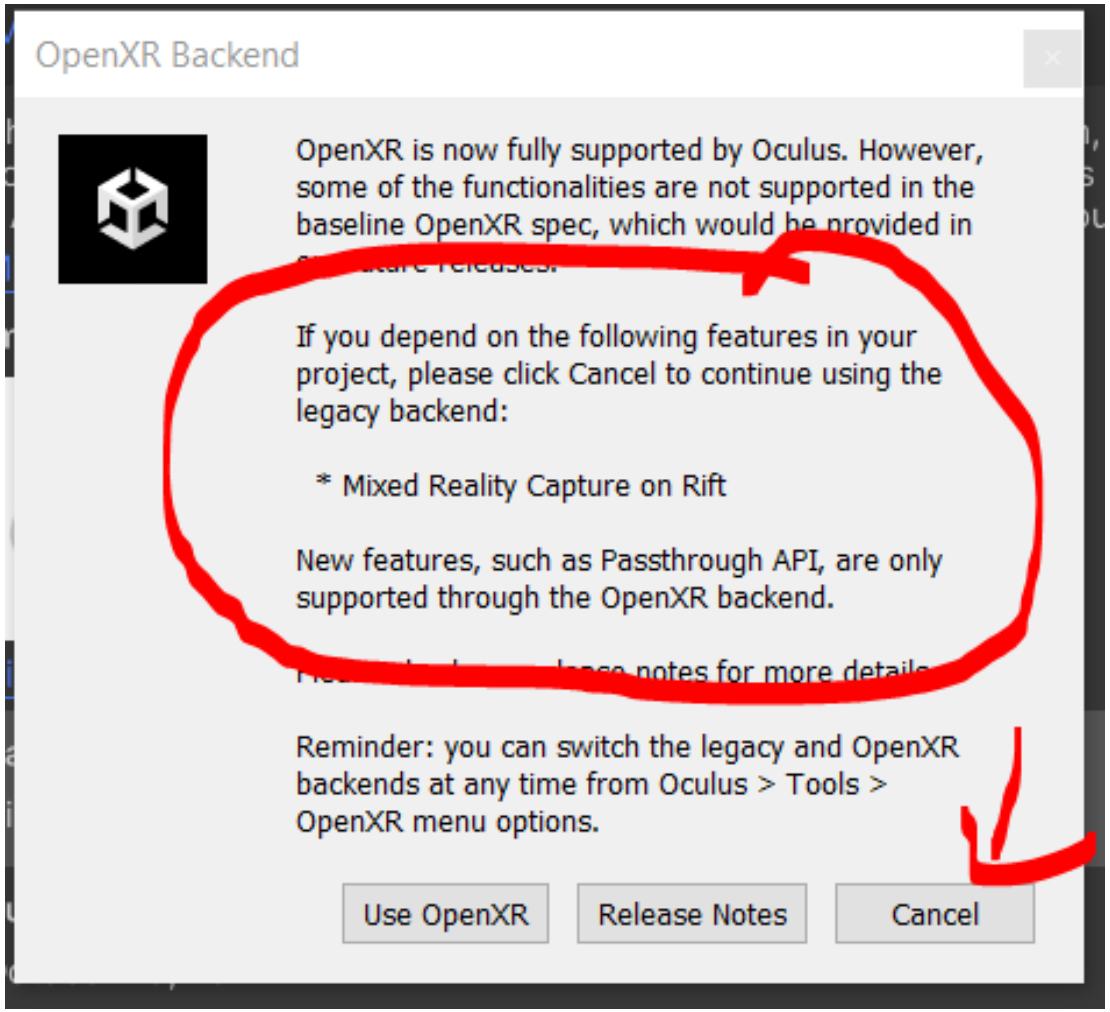
GEM2022

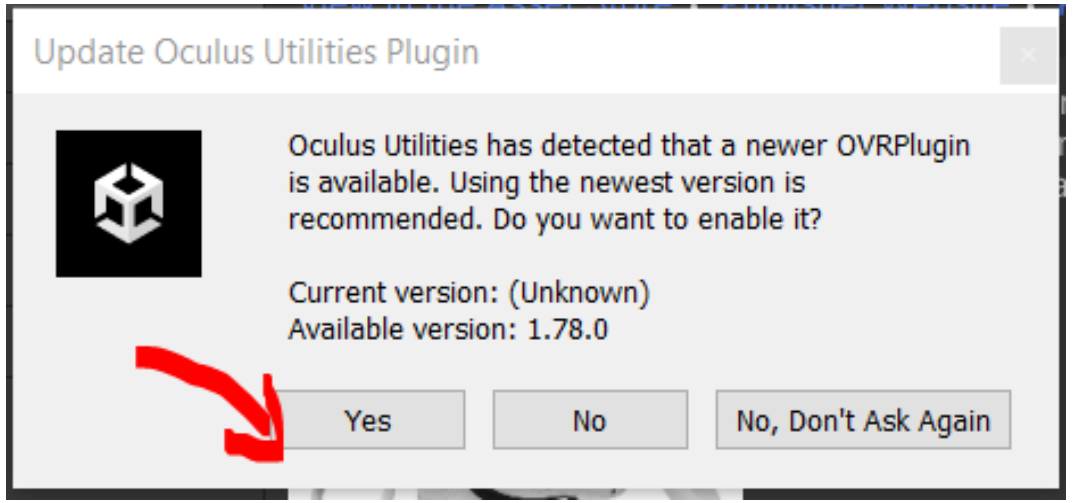
Window -> Package Manager



Setup

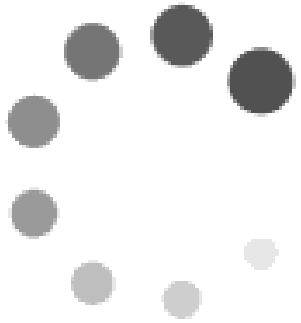
Window -> Package Manager





Setup

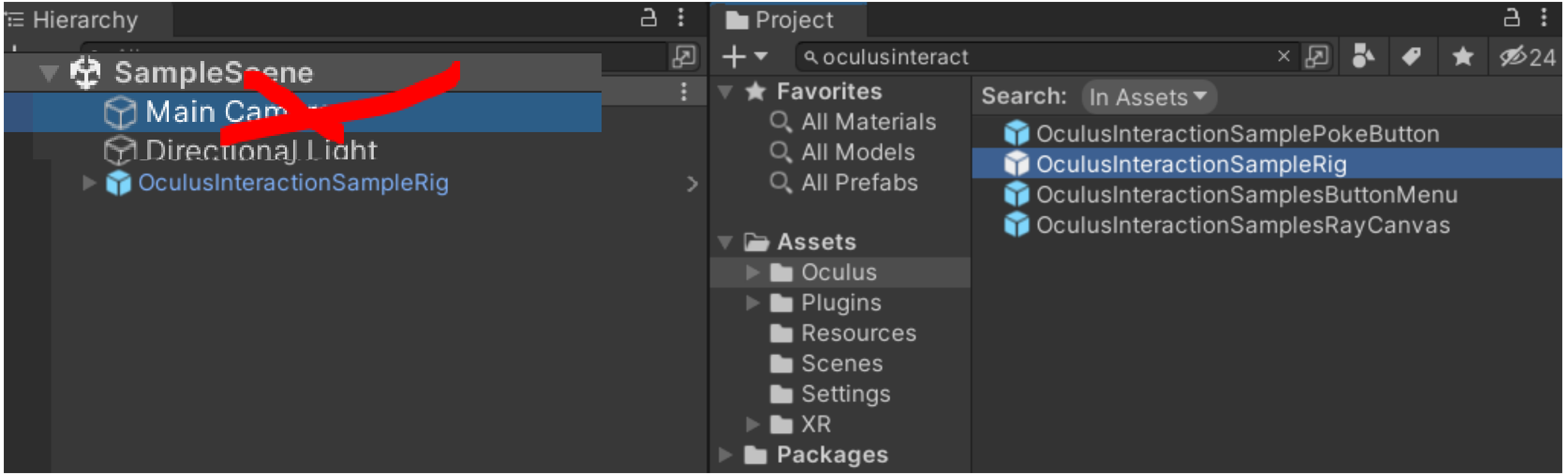
Questions?



Setup

Project -> search:

OculusInteraction

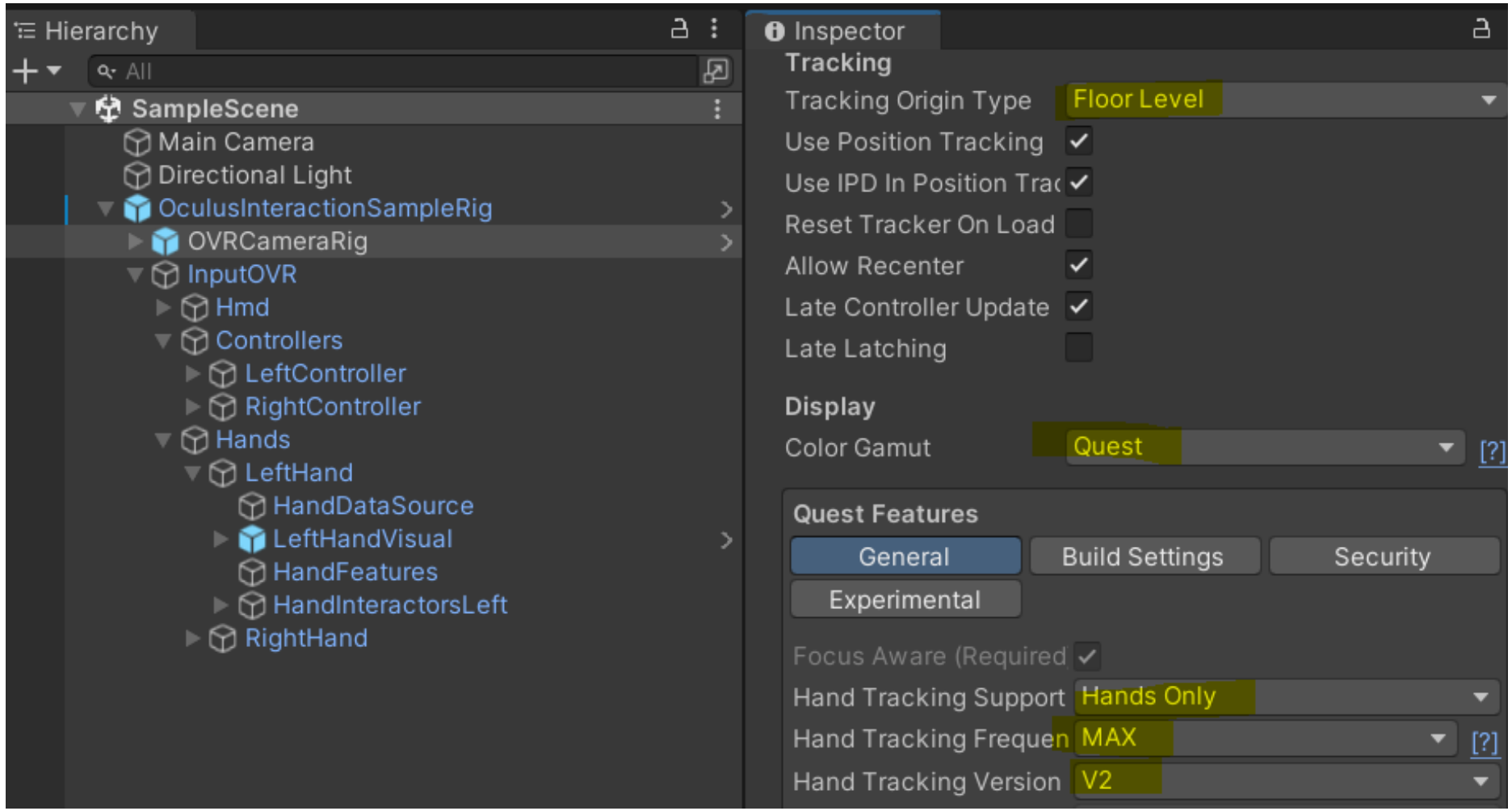


Setup

OculusInteractionSampleRig ->

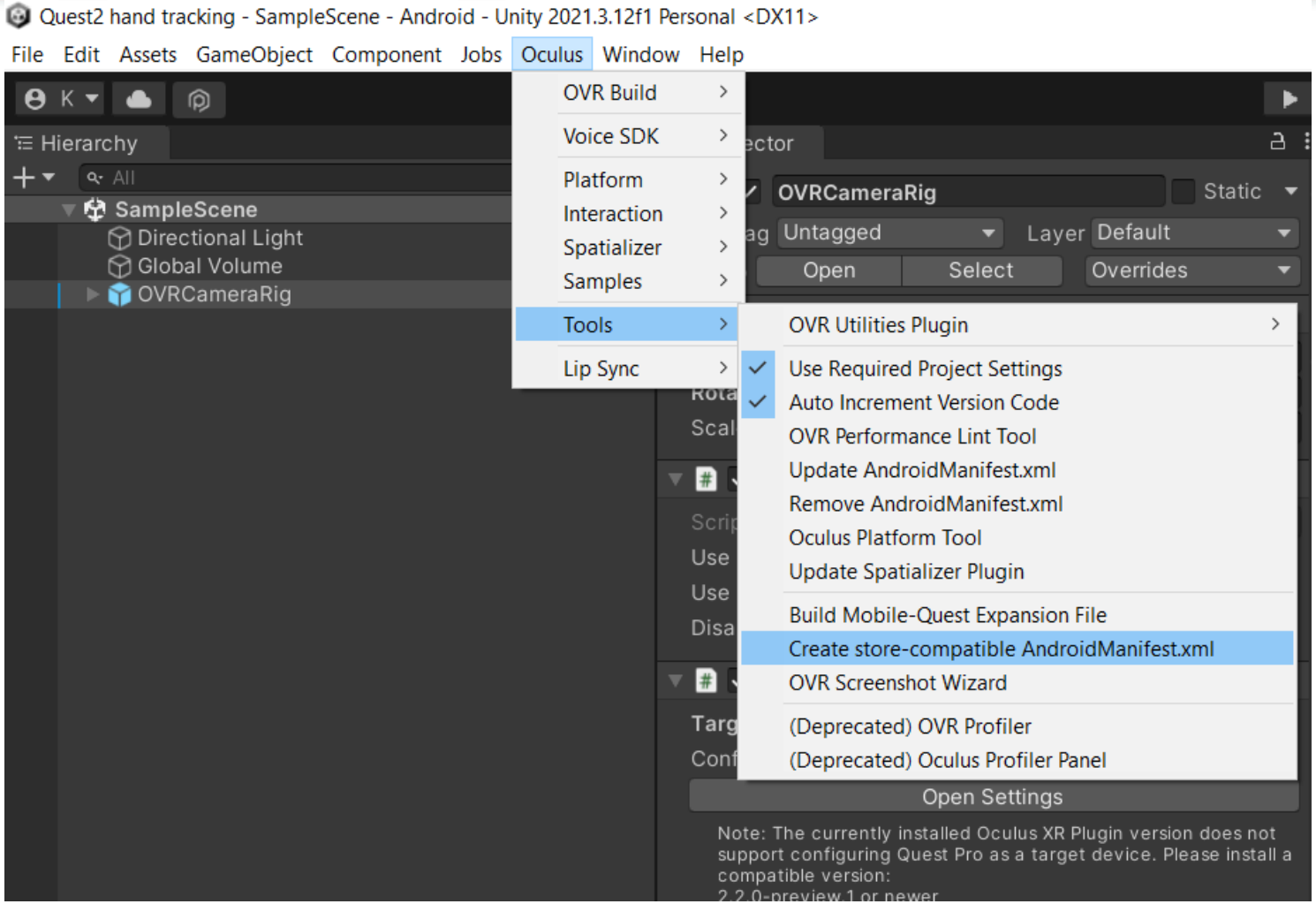
OVRCameraRig

OVR Manager (Script)



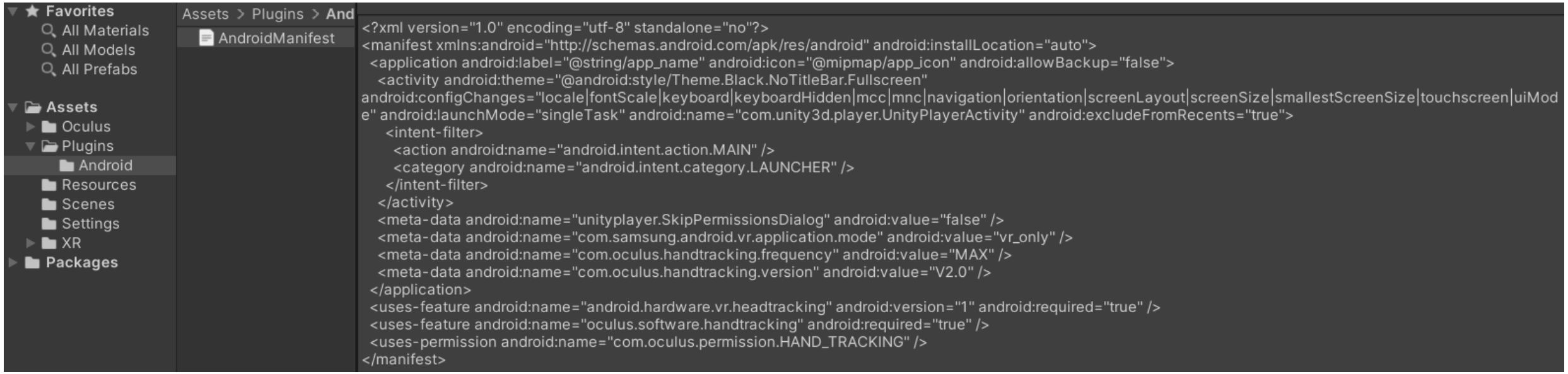
Setup

Oculus-> Tools



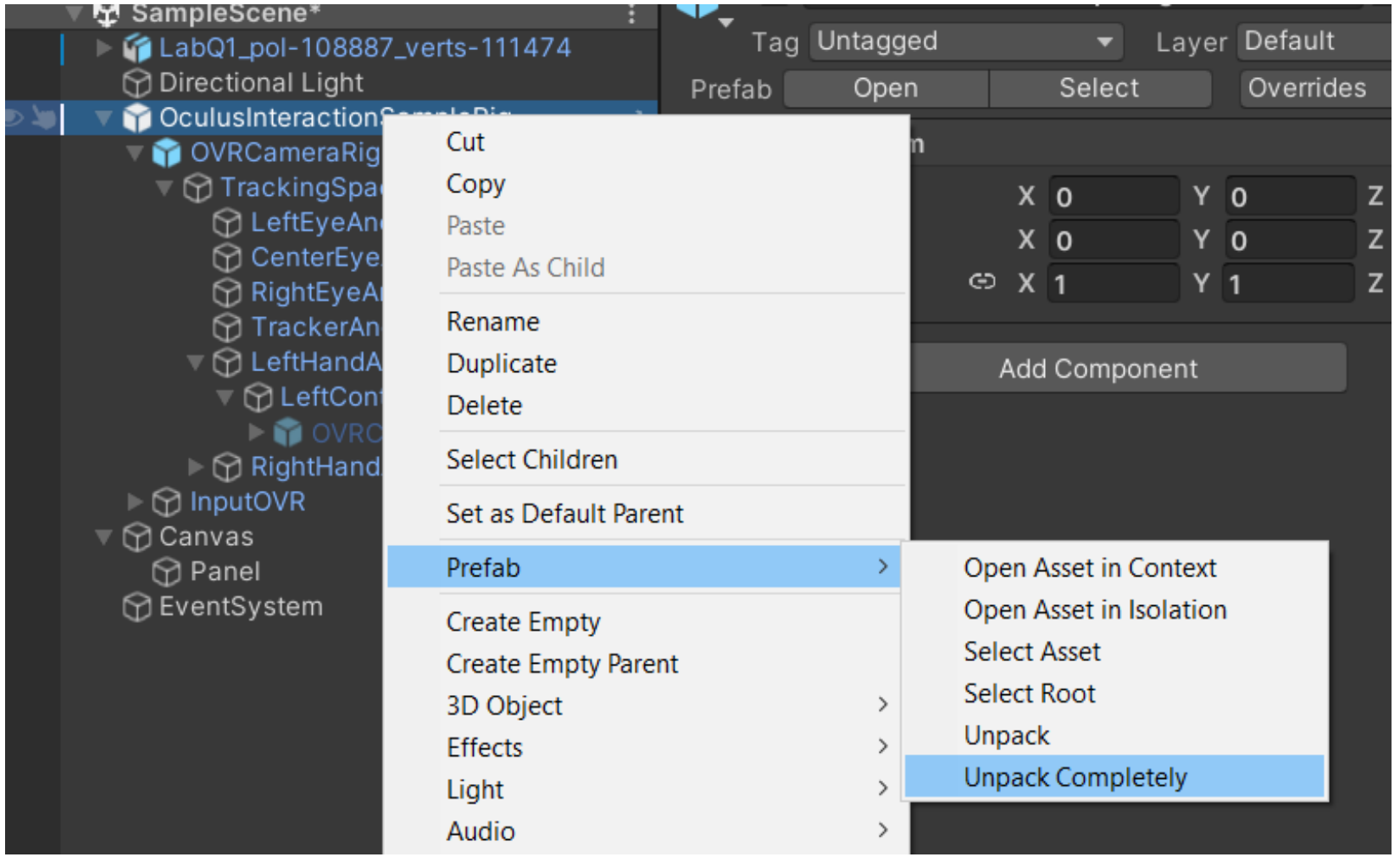
Setup

Oculus-> Tools



Setup

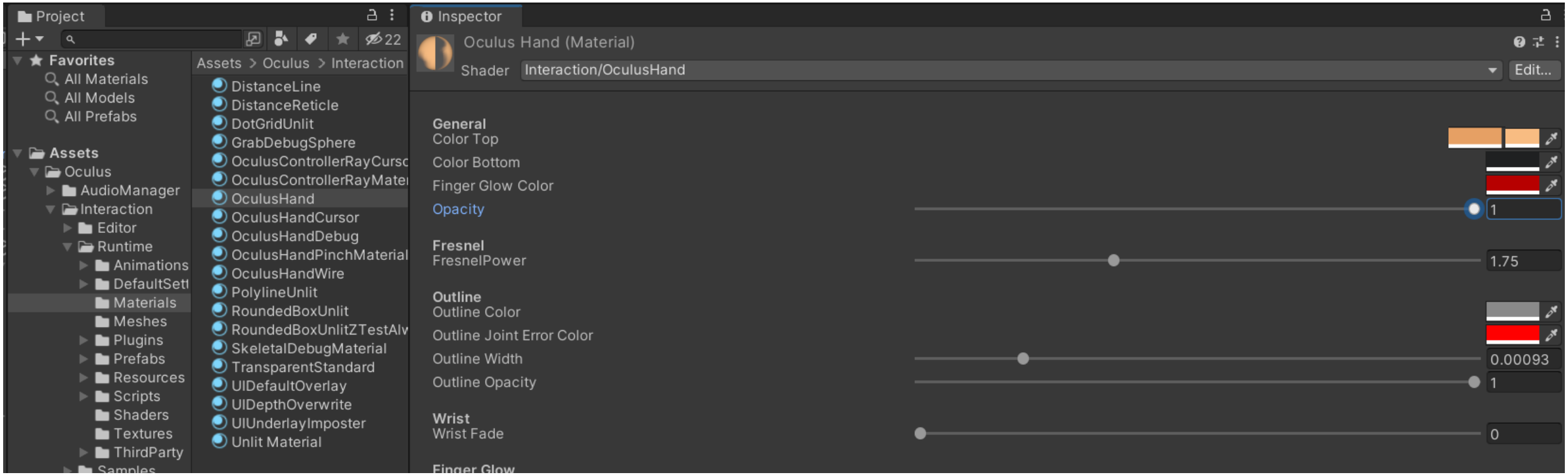
Prefab -> Unpack Compeltly



Setup

OculusHand

Material



Setup

Basic Hand Material

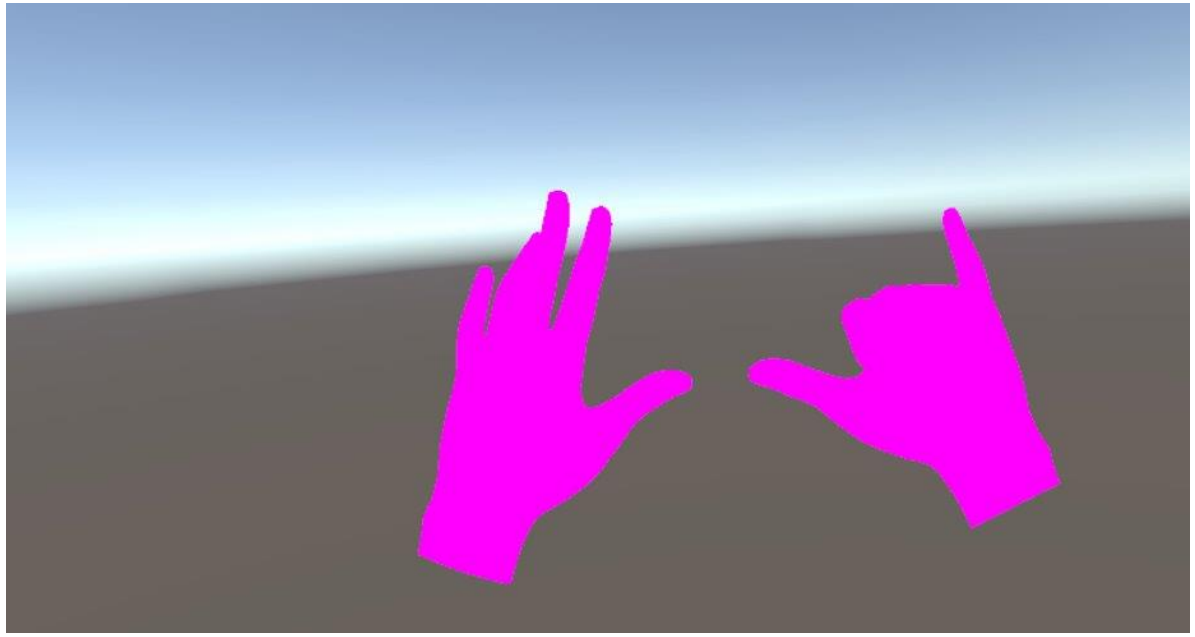
Quick fix for legacy shader

OVRHandPrefab

Material change to:
Universal Render Pipeline -> Complex lit

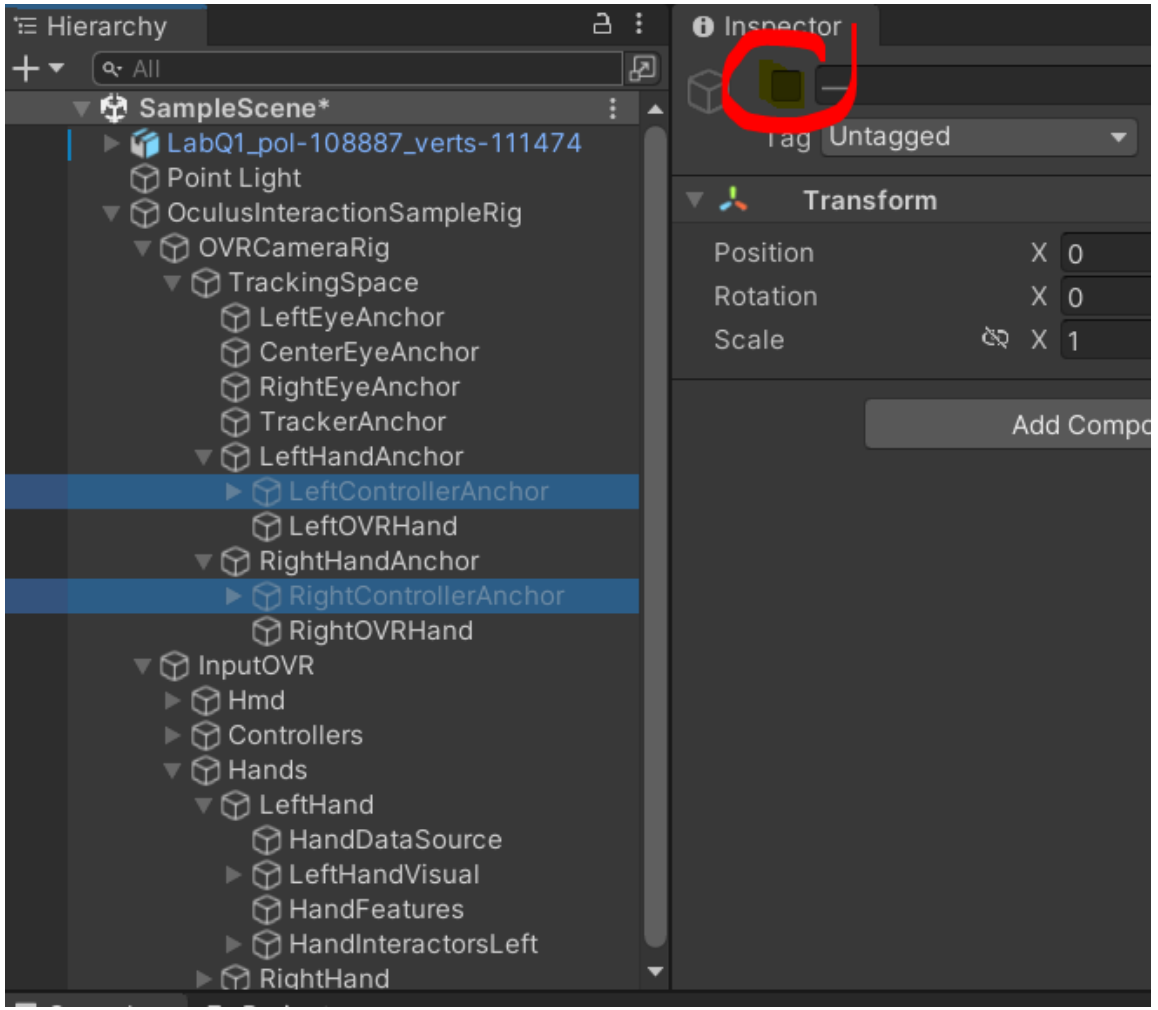
Base Map color -> Change (to match the skin color)

Smoothness -> 0



Setup

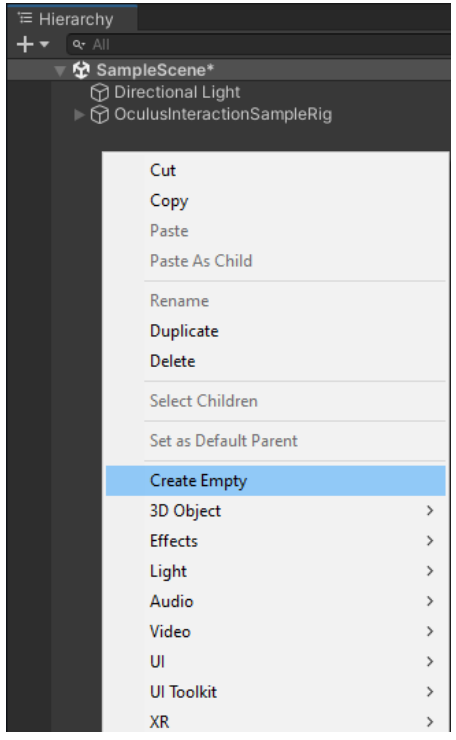
Disabling controllers' anchors



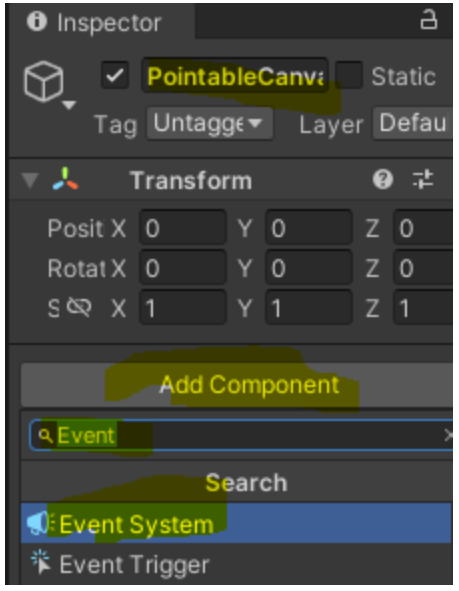
Setup

PointableCanvasModule

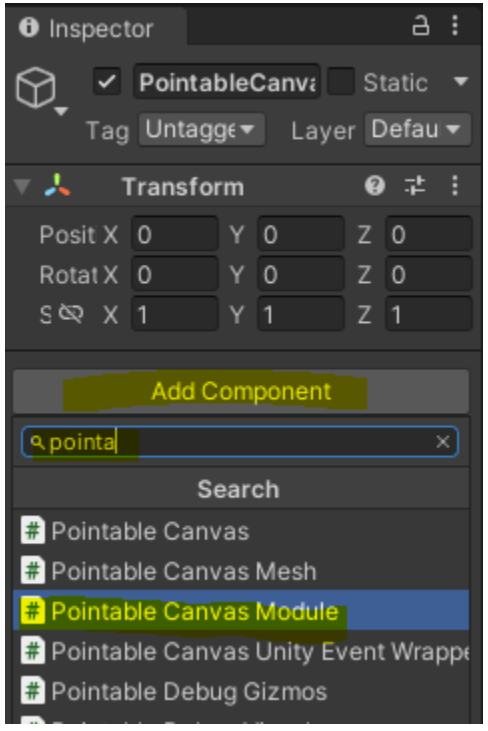
1) Hierarchy (RMB)
Create Empty
rename:
PointableCanvasModule



2) Add Component:
Event System



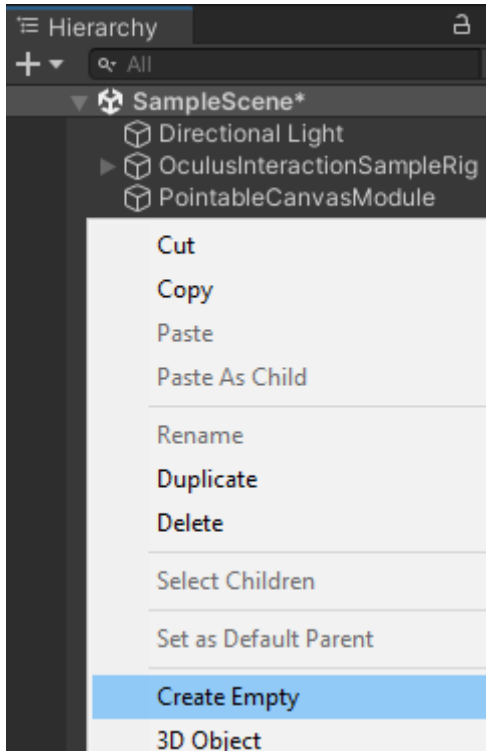
3) Add Component:
Pointable Canvas Module (script)



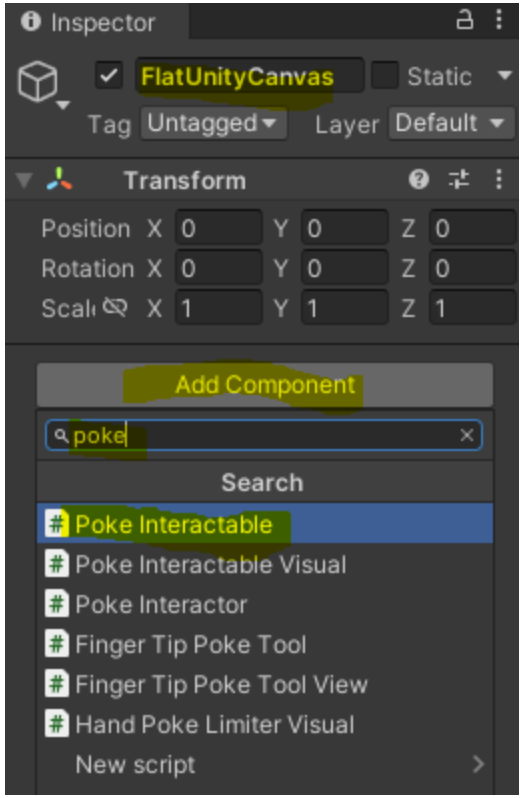
Setup

FlatUnityCanvas

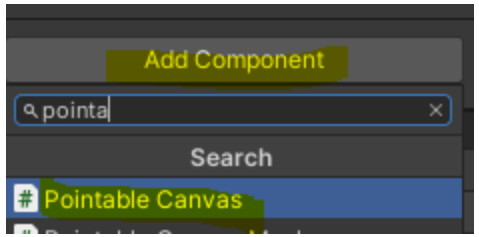
1) Hierarchy (RMB) Create Empty rename: FlatUnityCanvas



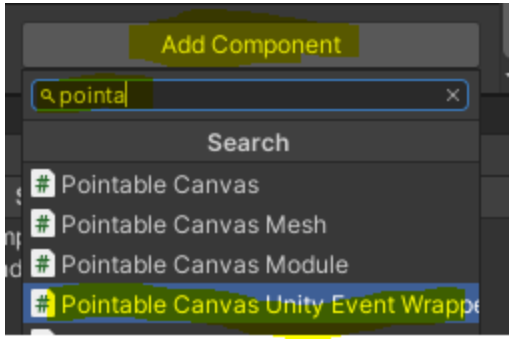
2) Add Component: Poke Interactable



3) Add Component: Pointable Canvas (script)



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

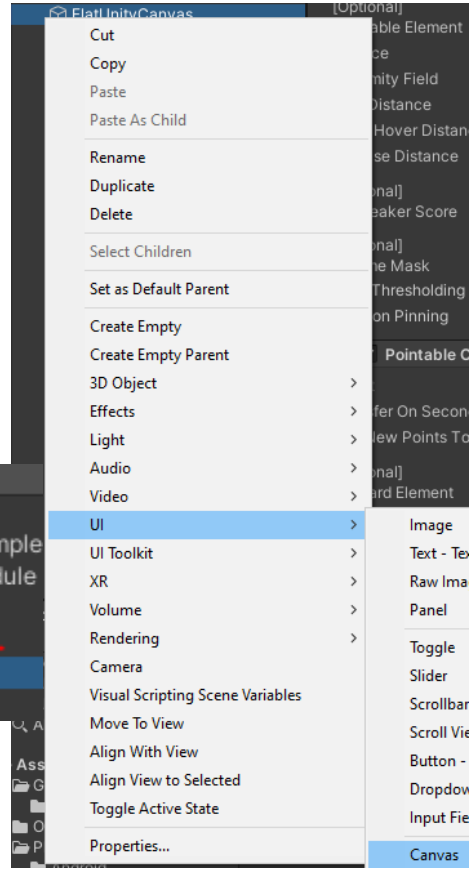


Setup

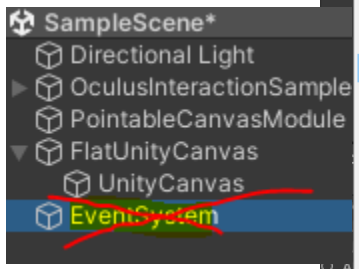
GEM2022

UnityCanvas

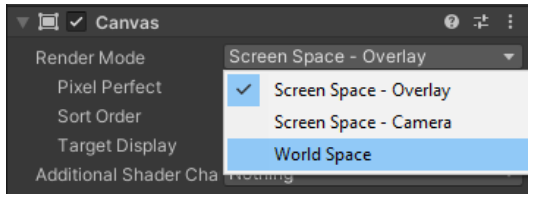
1) Hierarchy -> FlatUnityCanvas (RMB)
UI -> Canvas
rename:
UnityCanvas



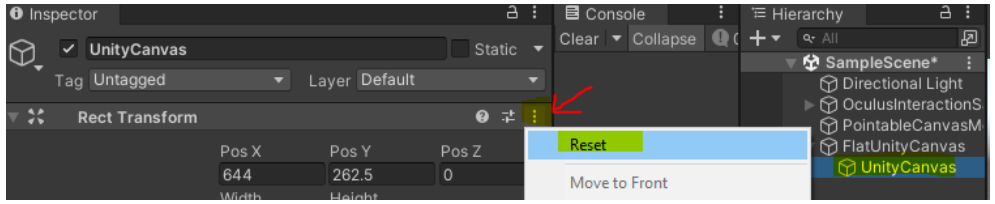
1.5) Delete:
EventSystem



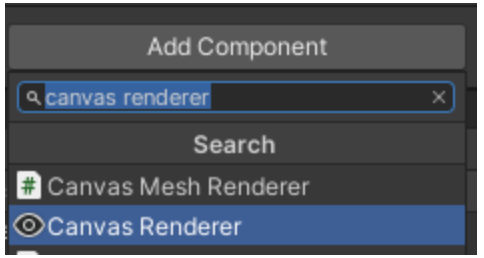
2) In UnityCanvas
Canvas
Change Render Mode:
World Space



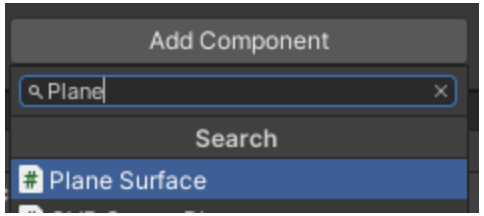
*) Reset position



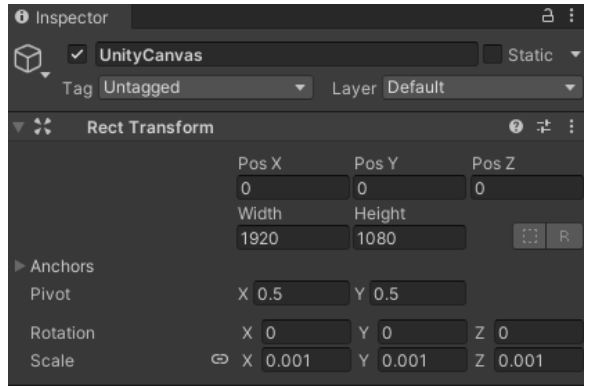
3) Add Component:
Canvas Renderer



4) Add Component:
Plane Surface
(script)



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



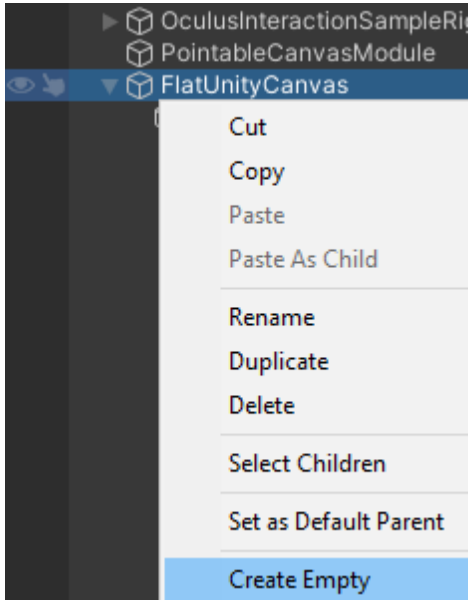
Setup

GEM2022

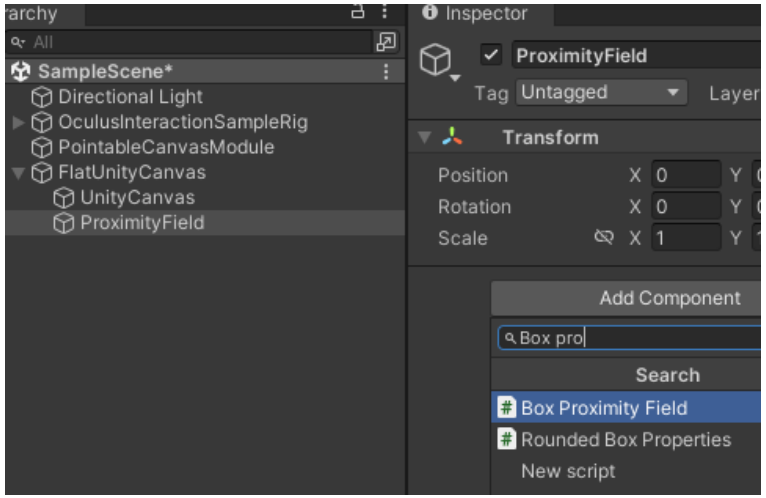
ProximityField

1) Hierarchy -> FlatUnityCanvas (RMB) Create empty

rename: ProximityField

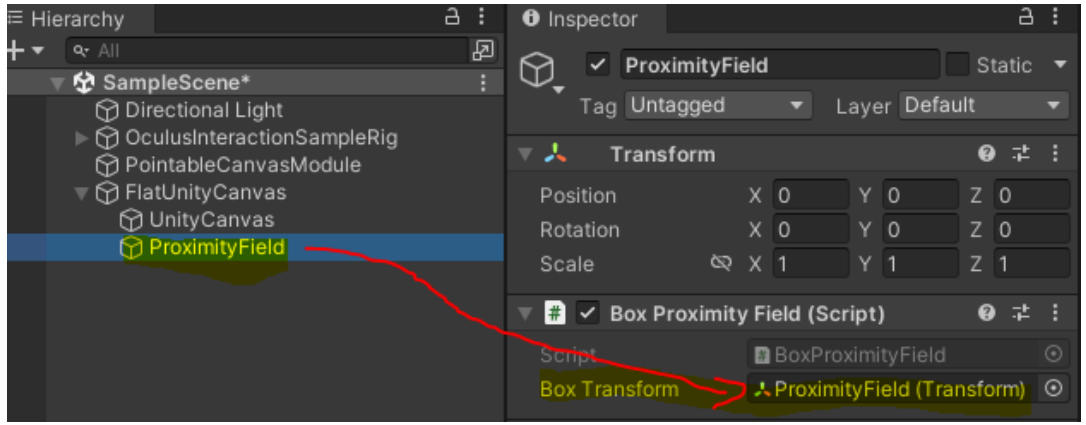


2) Add Component: Box Proximity Field (script)



3) Drag and drop

ProximityField(GO) to Box Transform in Box Proximity Field



Setup

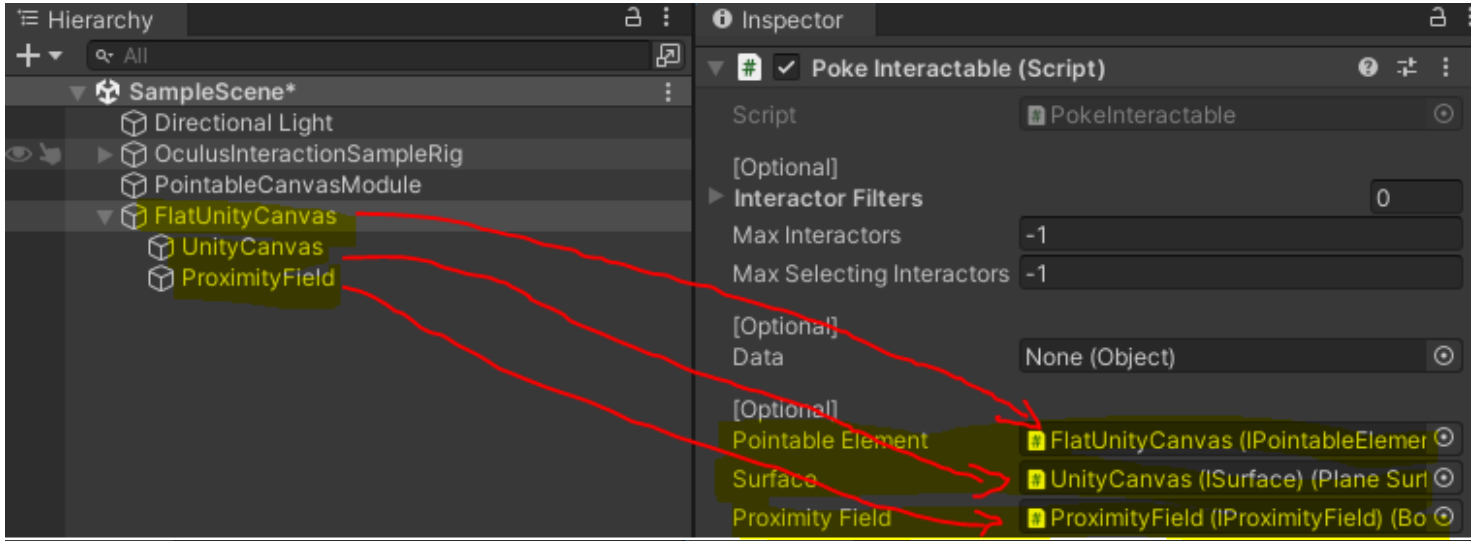
FlatUnityCanvas

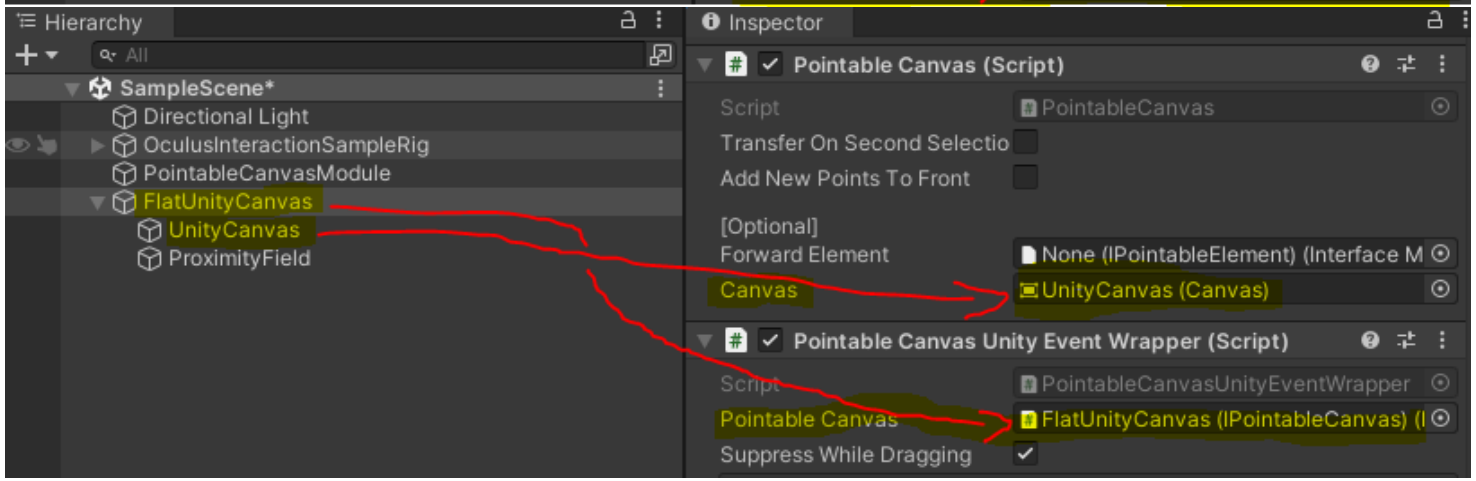
Drag and drop:

in Poke Interactable:
-- FlatUnityCanvas (GO)
to Pointable Element
-- UnityCanvas (GO)
to Surface
-- ProximityField (GO)
to Proximity Field

in Pointable Canvas:
-- UnityCanvas (GO)
to Canvas

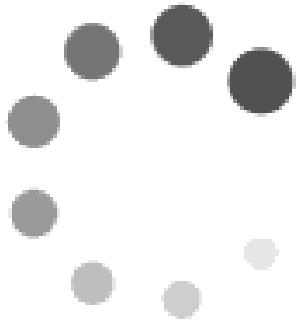
in Pointable Canvas Unity Event Wrapper
-- FlatUnityCanvas (GO)
to Pointable Canvas





Setup

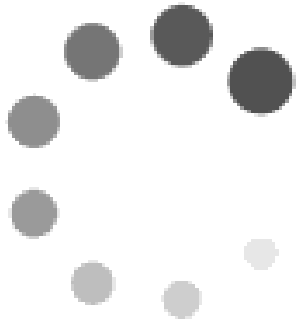
Questions?



>>> if needed open Scenes -> **Step1_setupComplete** scene

Setup

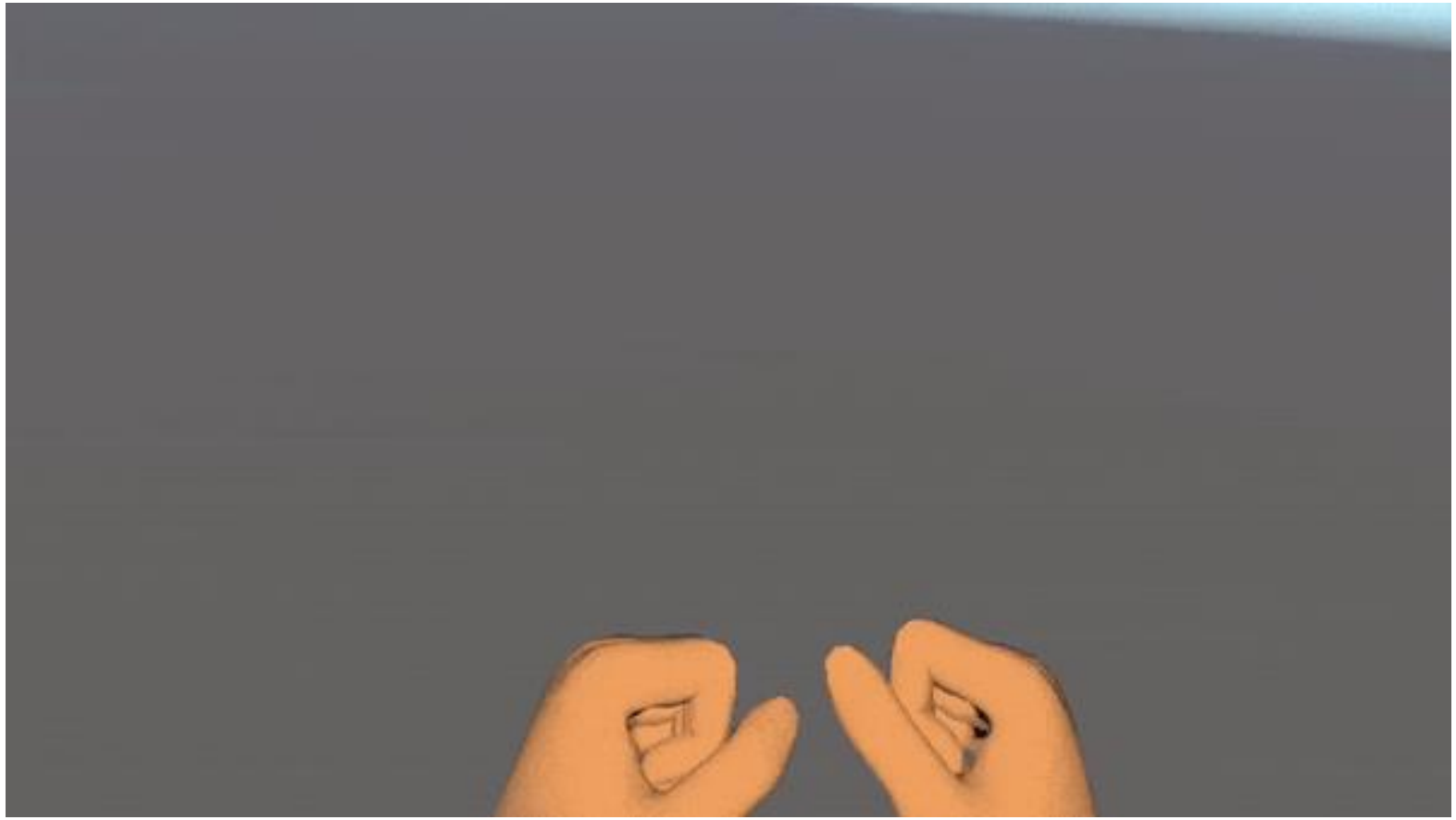
Status



>>> Congratulations!

Setup

Status: **COMPLETE!**



>>> Setup complete!

Questionnaire

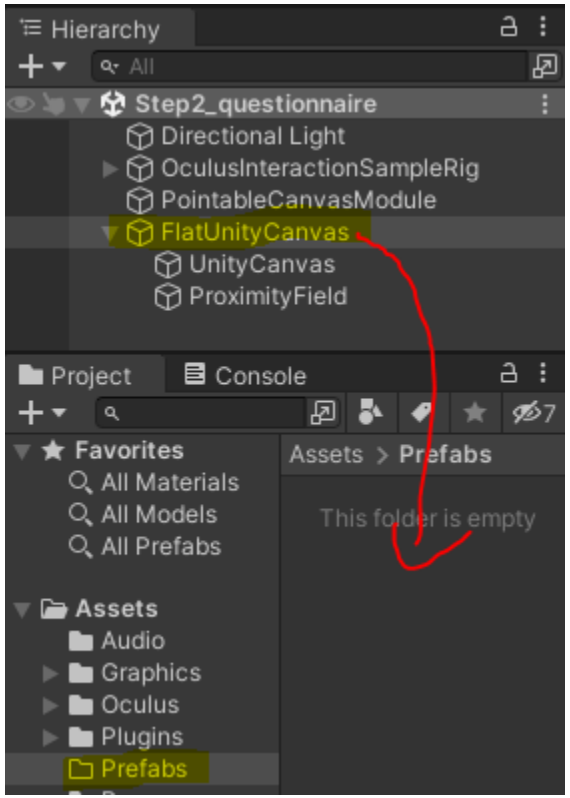
GEM2022

Adding folders:

Scenes & Prefabs & Scripts

1) Project (RMB) Create -> Folder
Rename
Repeat:
- Scenes
- Prefabs
- Scripts

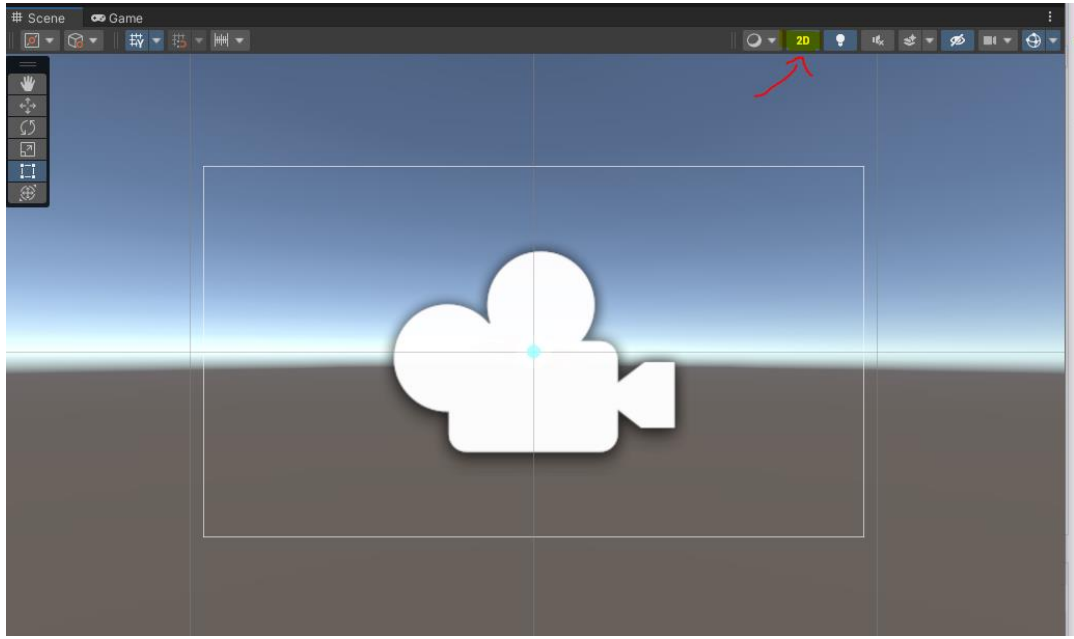
2) File -> Save As -> whatever
... in Scenes folder



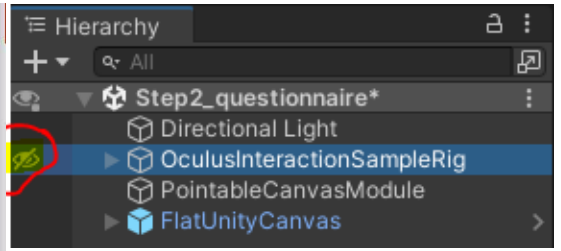
3) In Project -> Open prefabs
Drag and drop FlatUnityCanvas from Hierarchy to that folder in editor

4) In the Scene window click on 2D

Next select FlatUnityCanvas in Hierarchy, press **F**



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

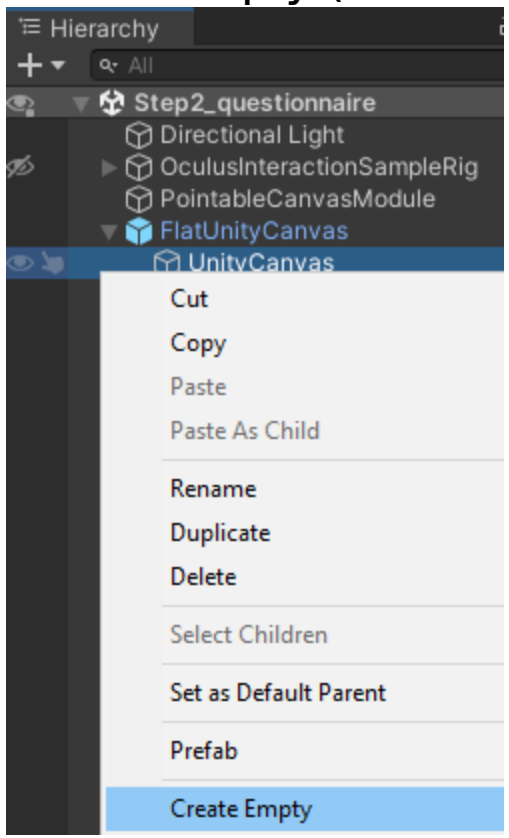


Questionnaire

GEM2022

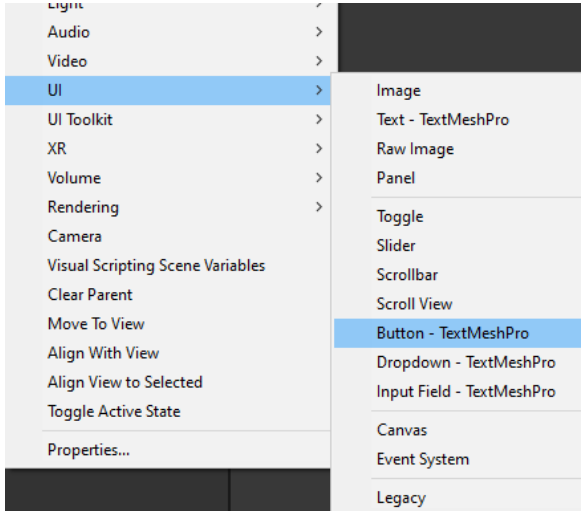
Adding the first button

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

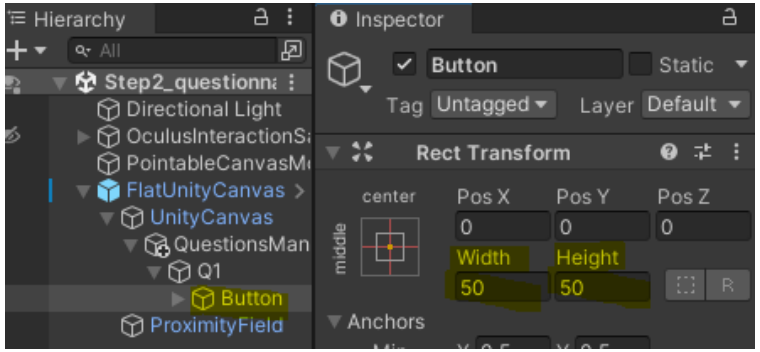


2) On that new object create another child (rename to Q1)

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



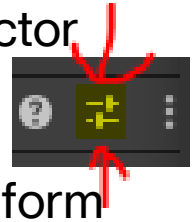
4) In the Inspector panel when this Button is selected in Hierarchy change Width and Height to 50



Questionnaire

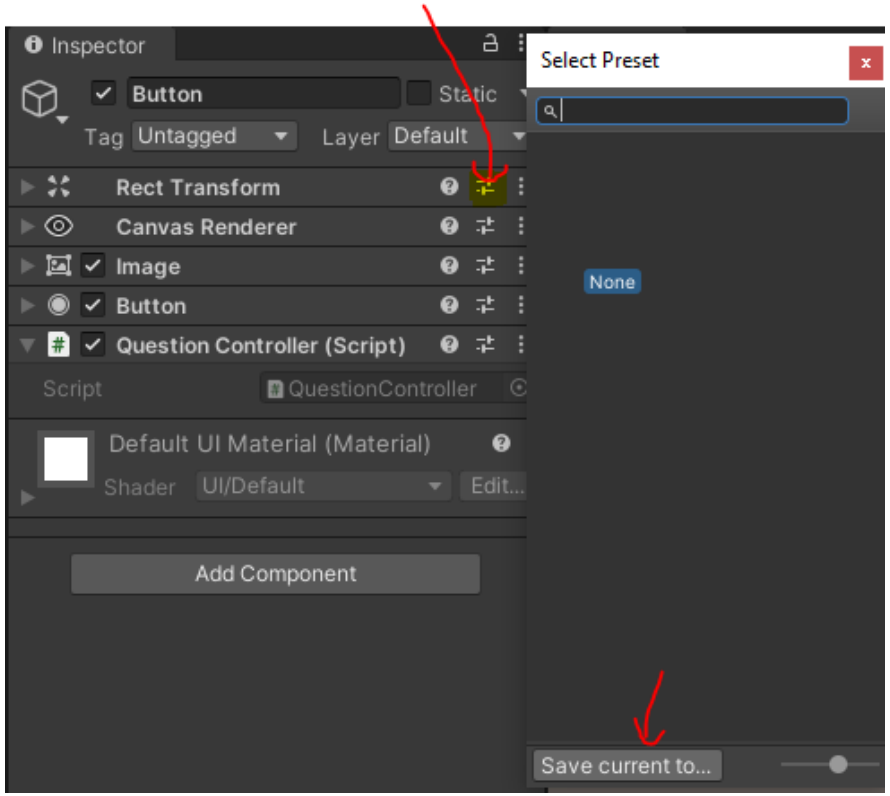
GEM2022

Create first preset

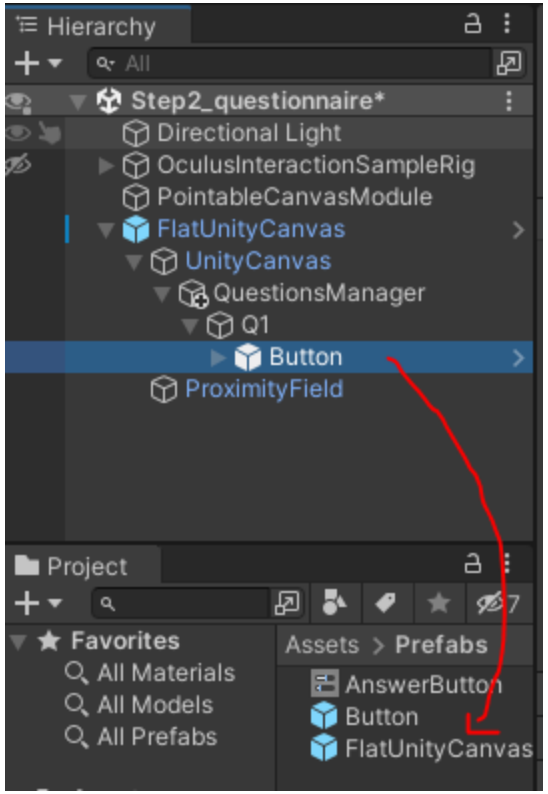
1) In the Inspector click the icon 

on Rect Transform component

Save current to... AnswerButton



2) Drag and drop Button to Prefabs folder

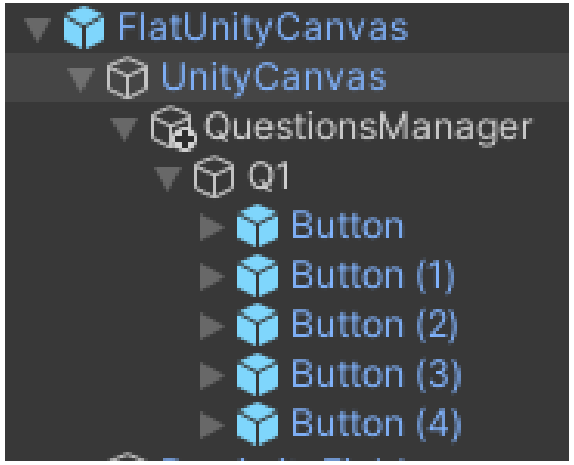


Questionnaire

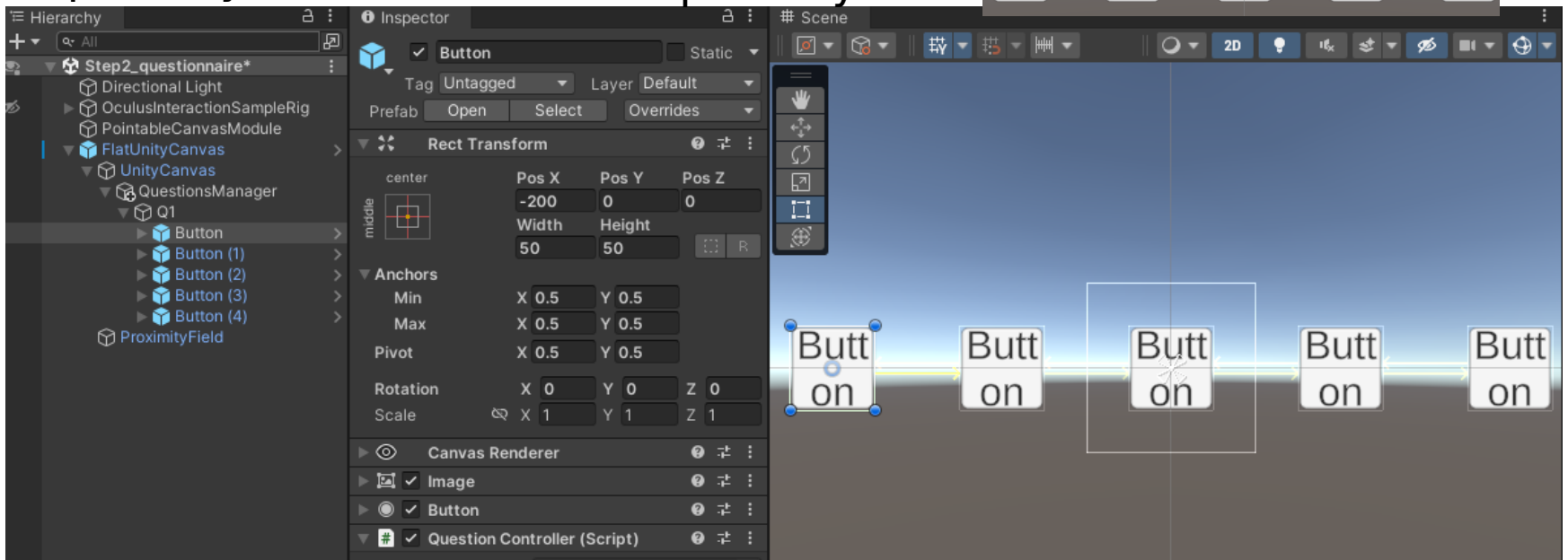
GEM2022

Populating Q1 with buttons

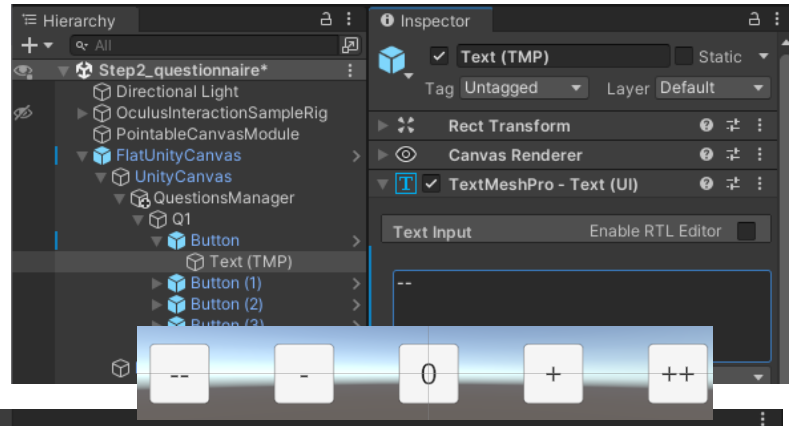
1) Drag and drop from Prefabs Button prefab onto Q1 game object (repeat 4 times)



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



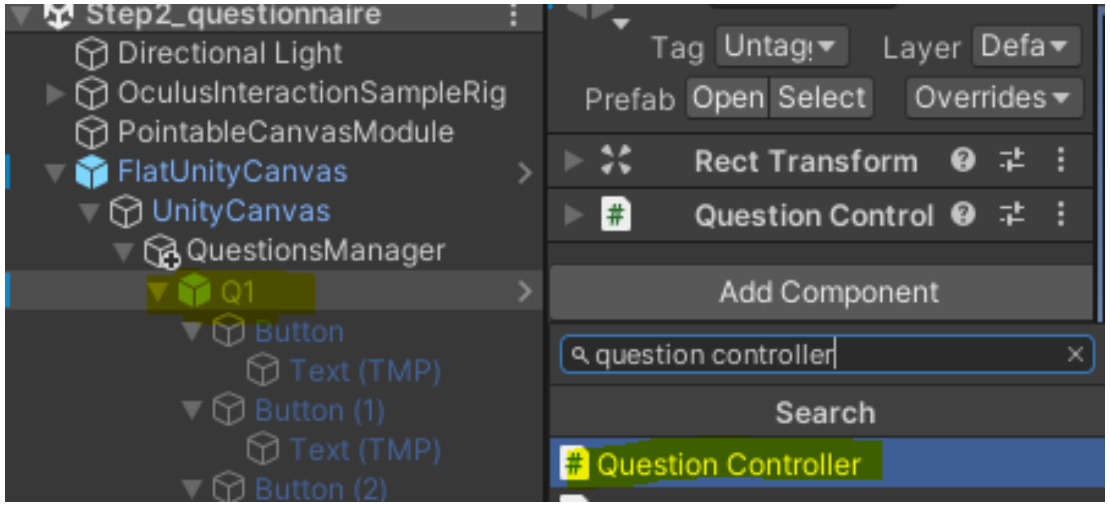
Questionnaire

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

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

    0 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;
    }
}
```

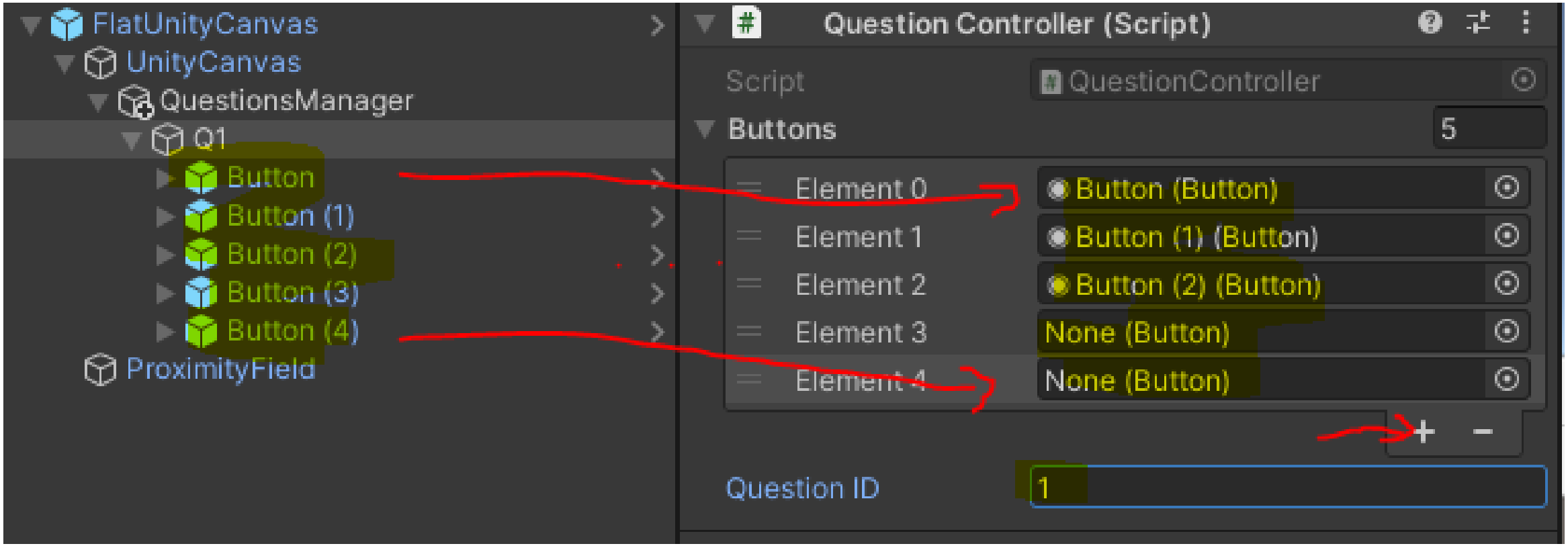
Save!.. save often... here and in Unity

Questionnaire

GEM2022

Drag and dropping exercise...

Drag and drop each Button to Buttons list

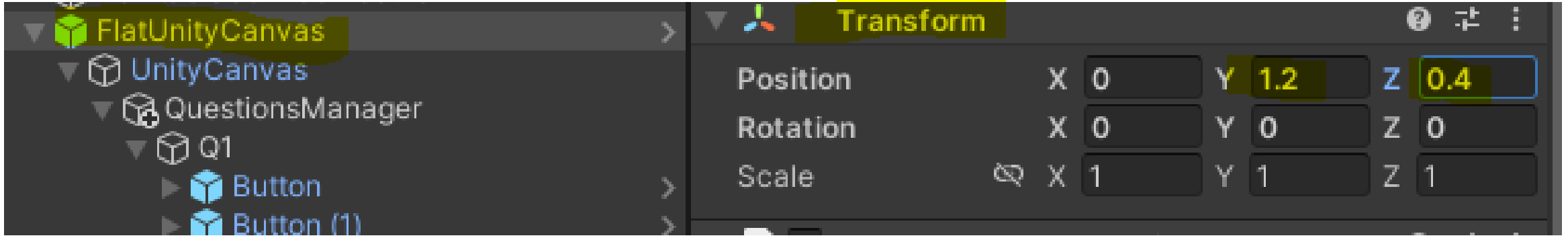


Change Question ID to 1

Questionnaire

Move that canvas elsewhere!

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

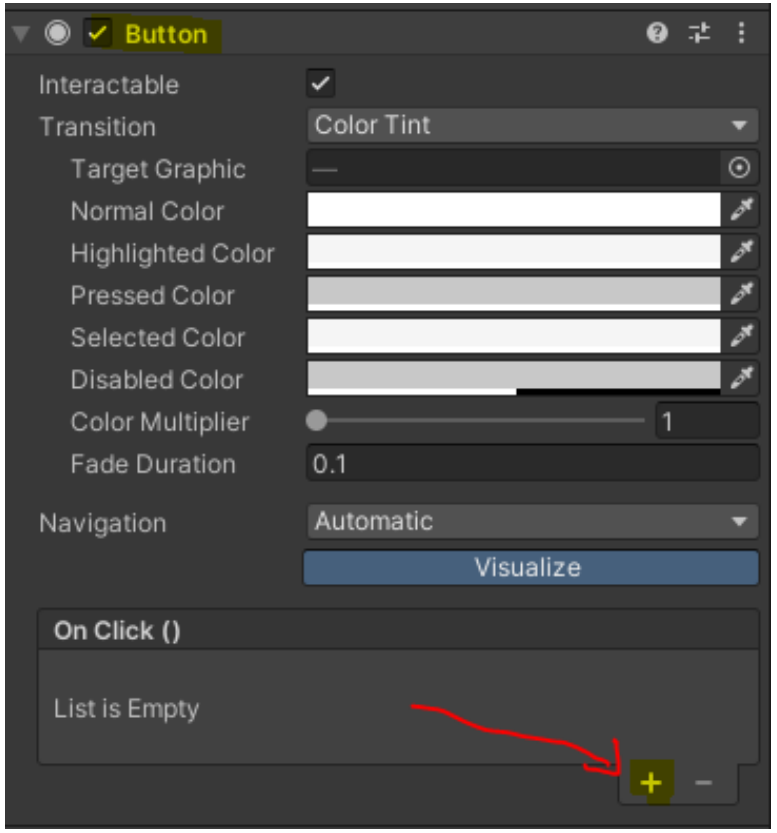


Questionnaire

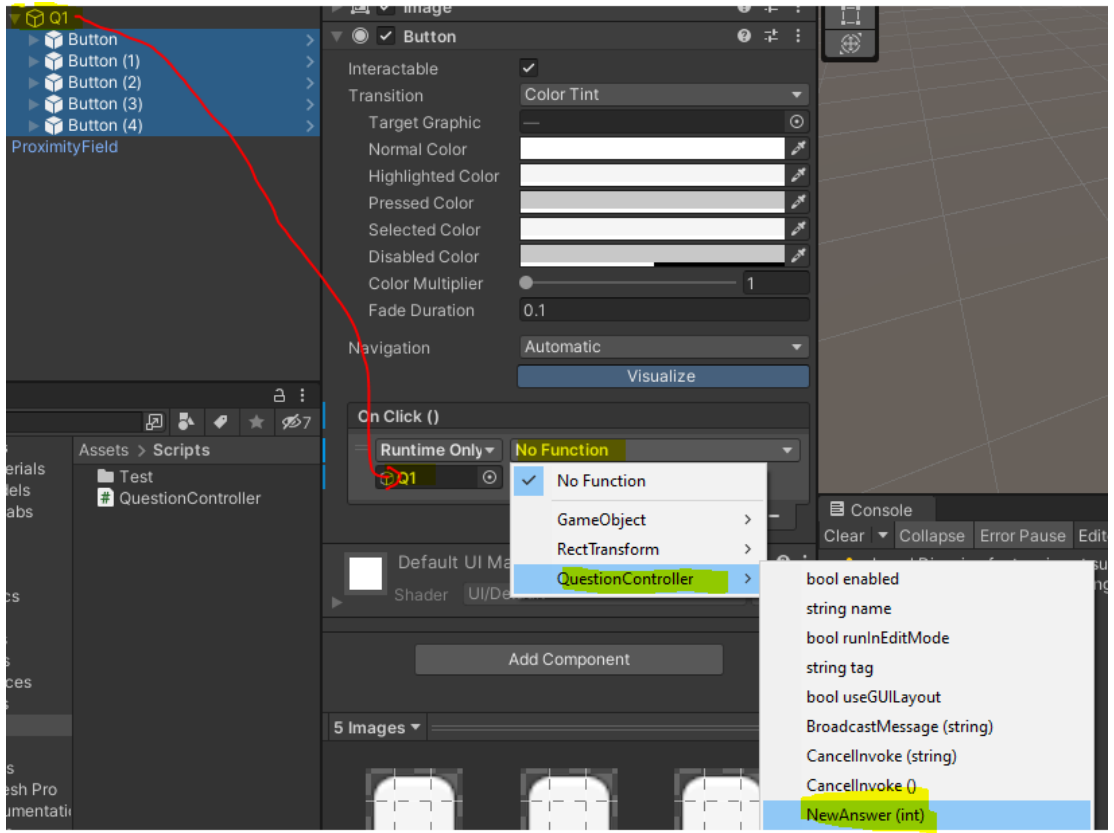
GEM2022

Lets make some action!

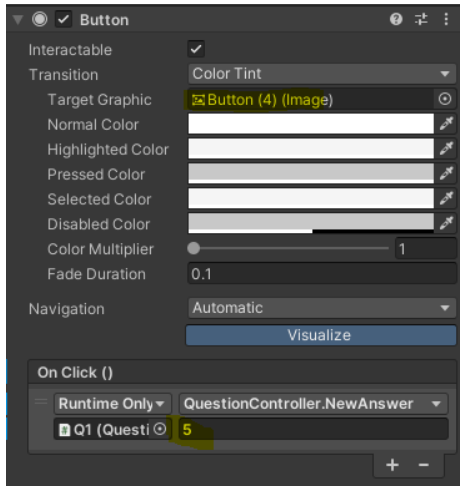
1) Select all the buttons, and in Button class click the +



2) Drag and drop the Q1 object



3) Select: QuestionController -> NewAnswer



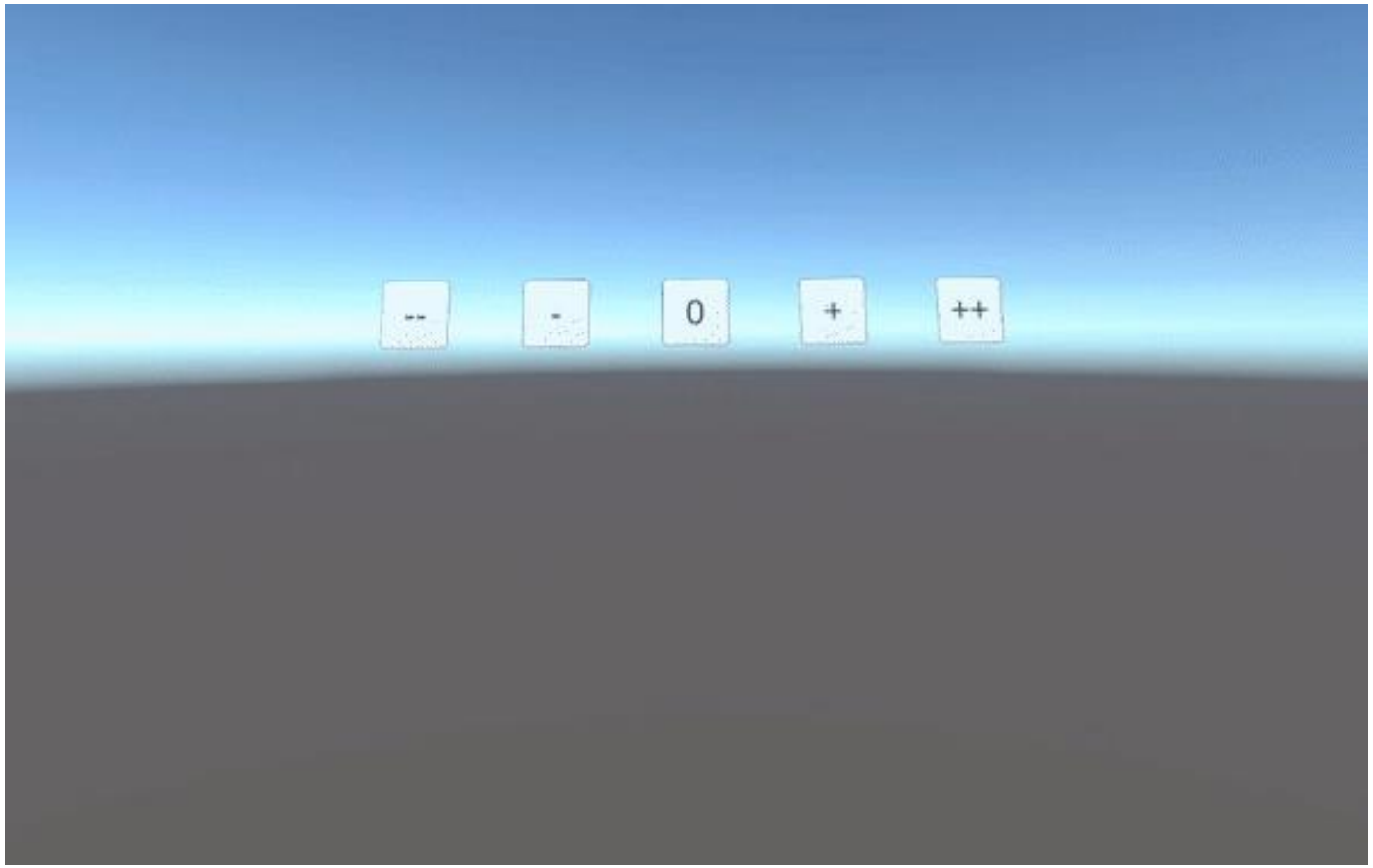
4) Change 0 to: 1, 2, 3 ,4, 5 respectively for each of these buttons

Questionnaire

GEM2022

Status: **TESTING!**

Press play and see if it works!



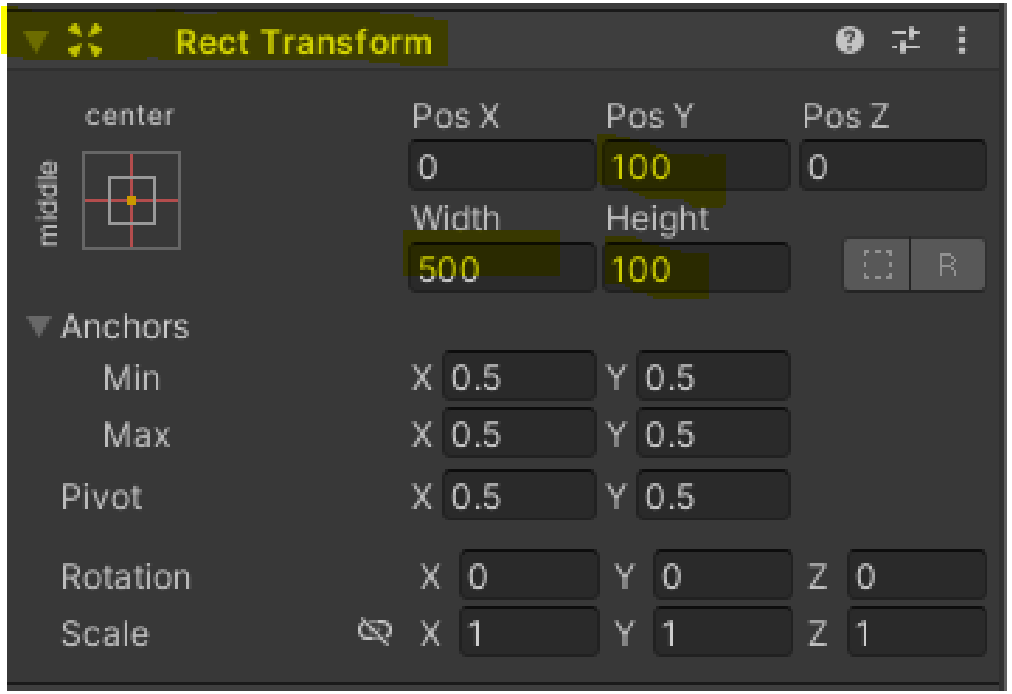
Questionnaire

GEM2022

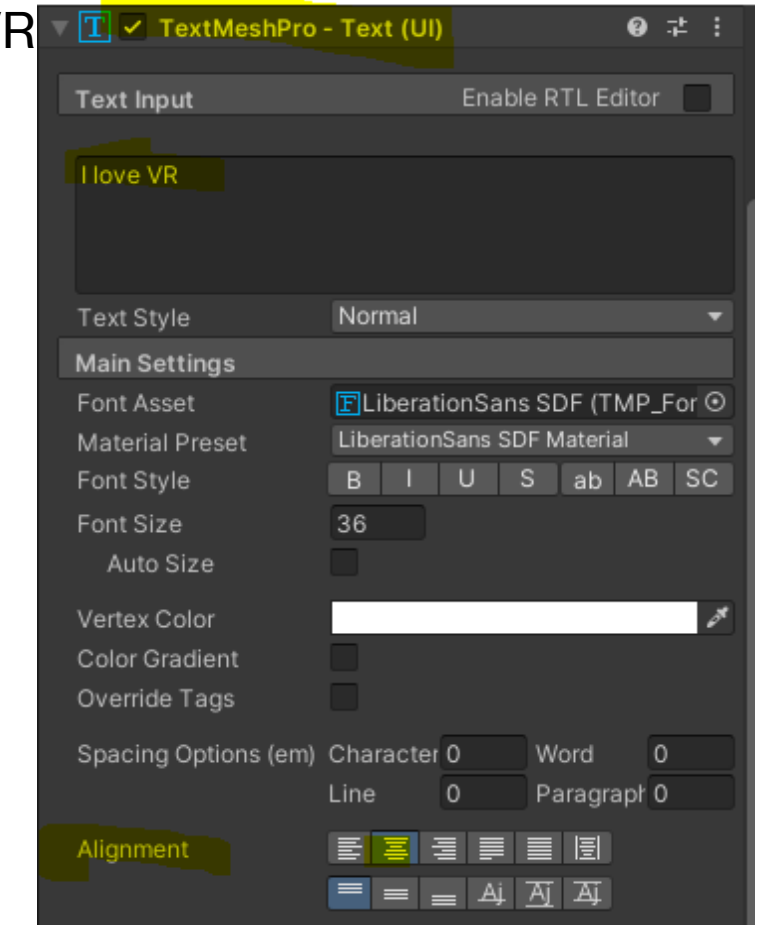
Hard to answer lack of questions

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

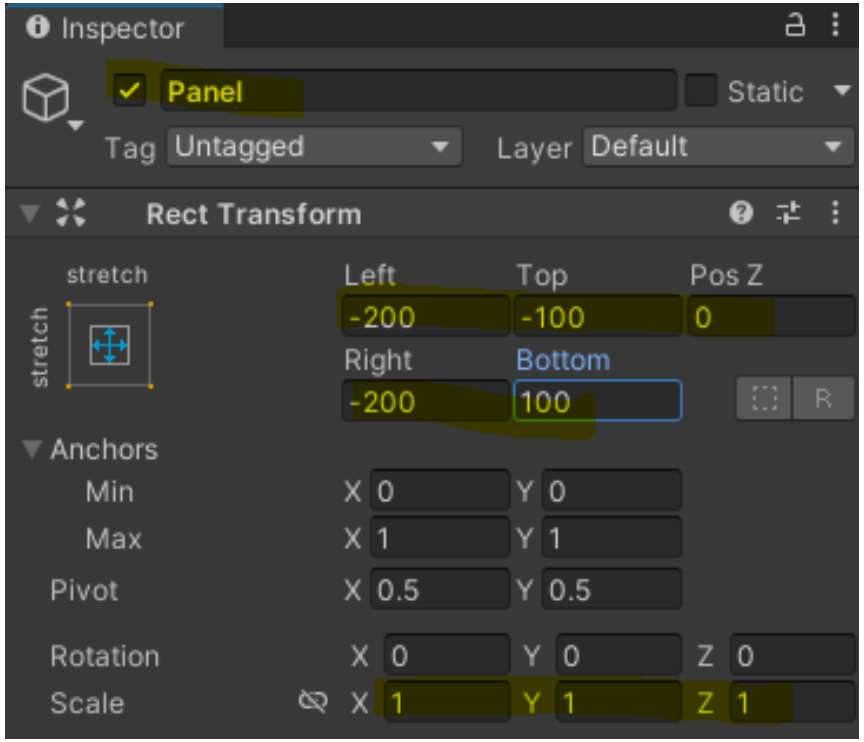


Questionnaire

GEM2022

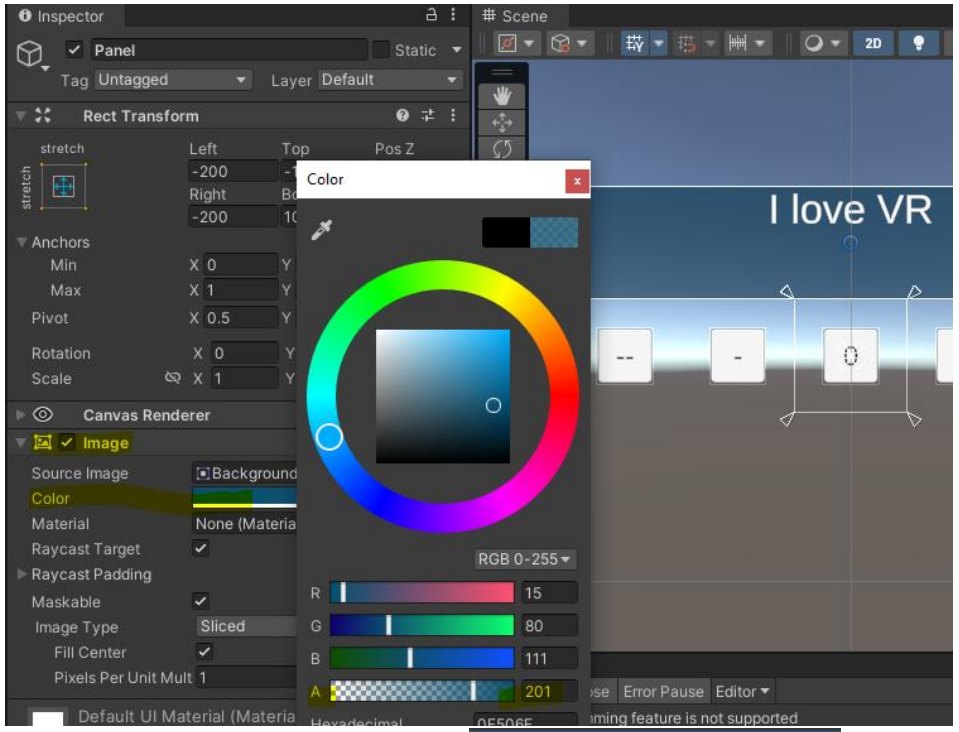
...and see them without background

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

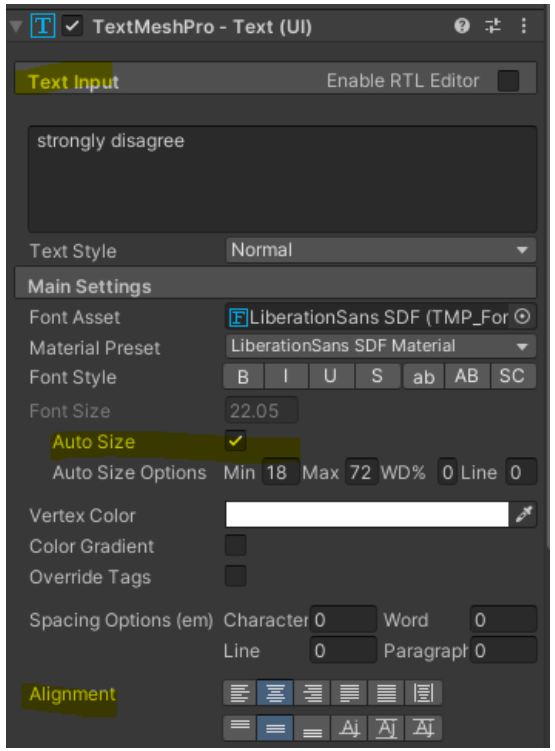
GEM2022

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

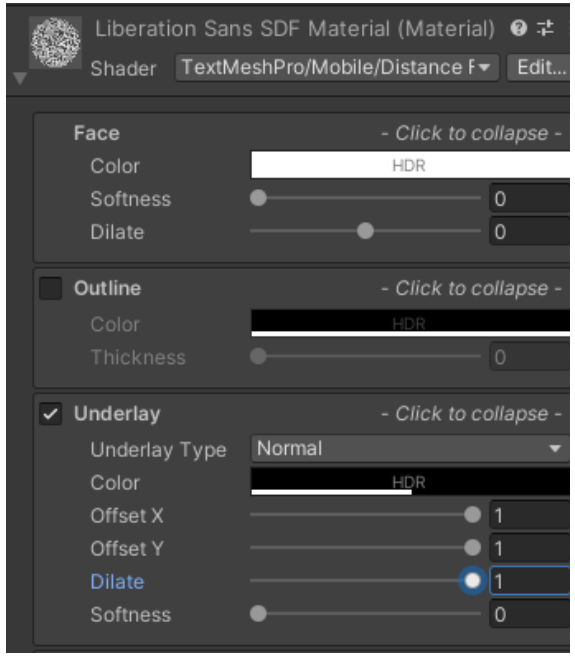


4) Select that text object and press <ctrl / cmd + d> 4 times

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

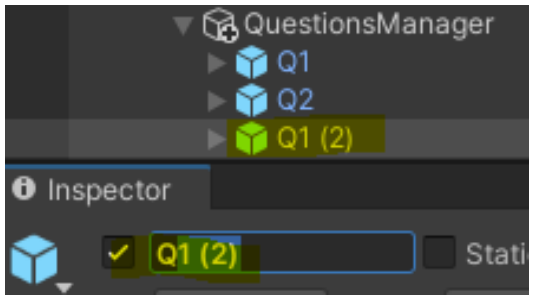
GEM2022

Adding questions

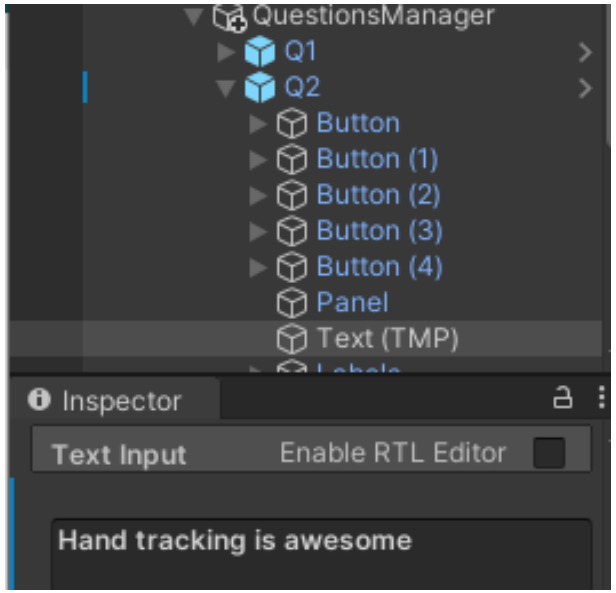
1) Drag and drop Panel UI Game Object to from Q1 to Question Manager

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

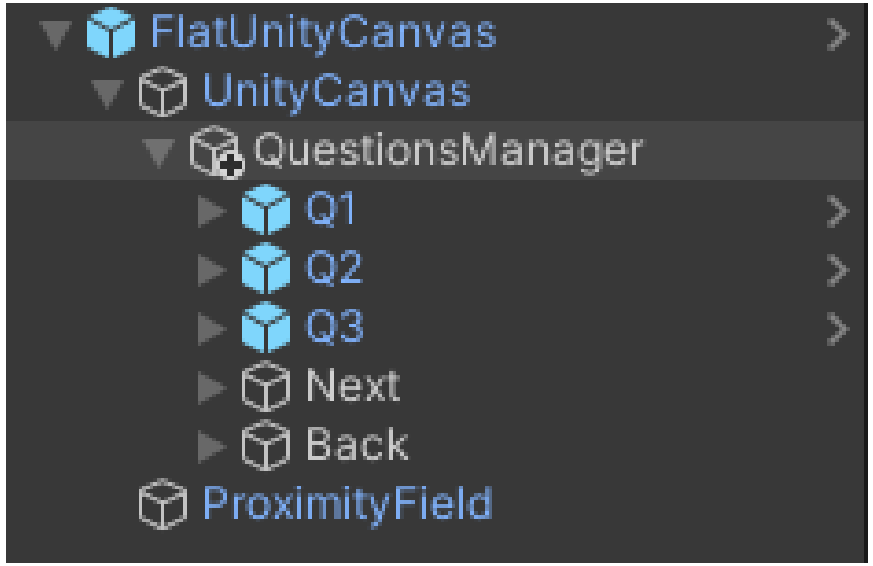
Questionnaire

GEM2022

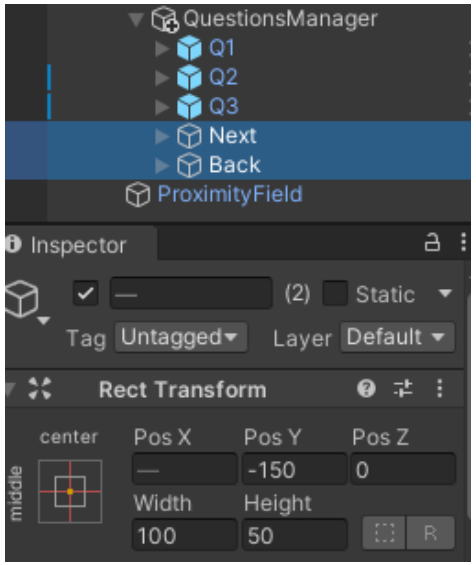
NEXT!

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

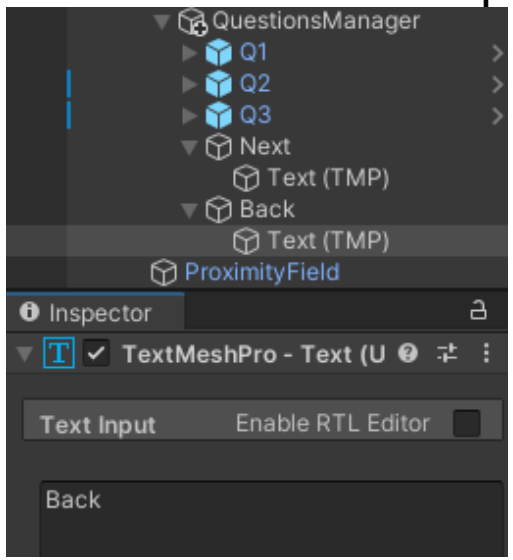
2) Rename them to: Next and Back

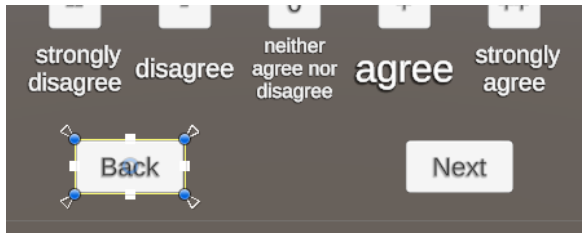


3) Change their
Width to 100 and
Height to 50
Pos Y to –150
and Pos X to 150 and –150
respectively



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



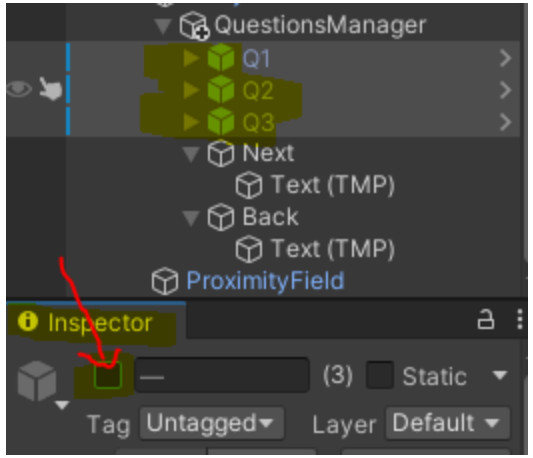


Questionnaire

GEM2022

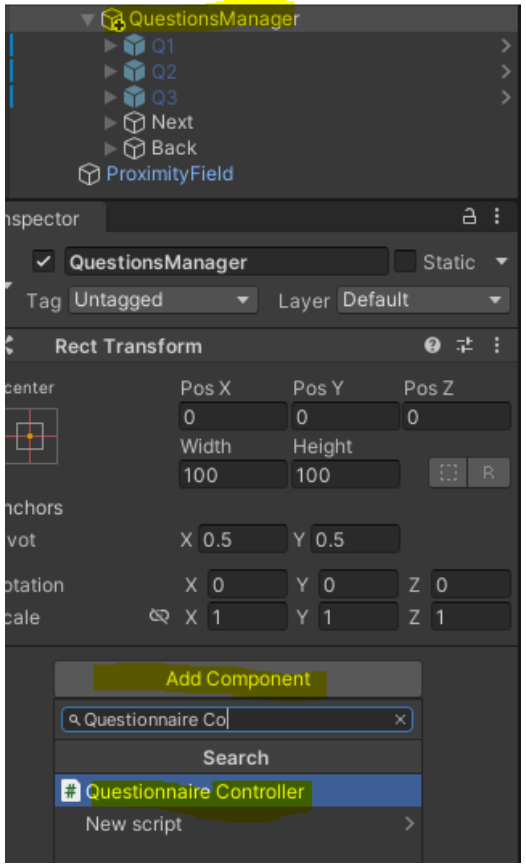
Adding some logic

1) Disable Q1, Q2 & Q3 in the inspector



2) In Scripts folder create: QuestionnaireController

3) Add it to the QuestionsManager



4) Open QuestionnaireController for edition

Questionnaire

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
    Unity Script | 0 references
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;
```

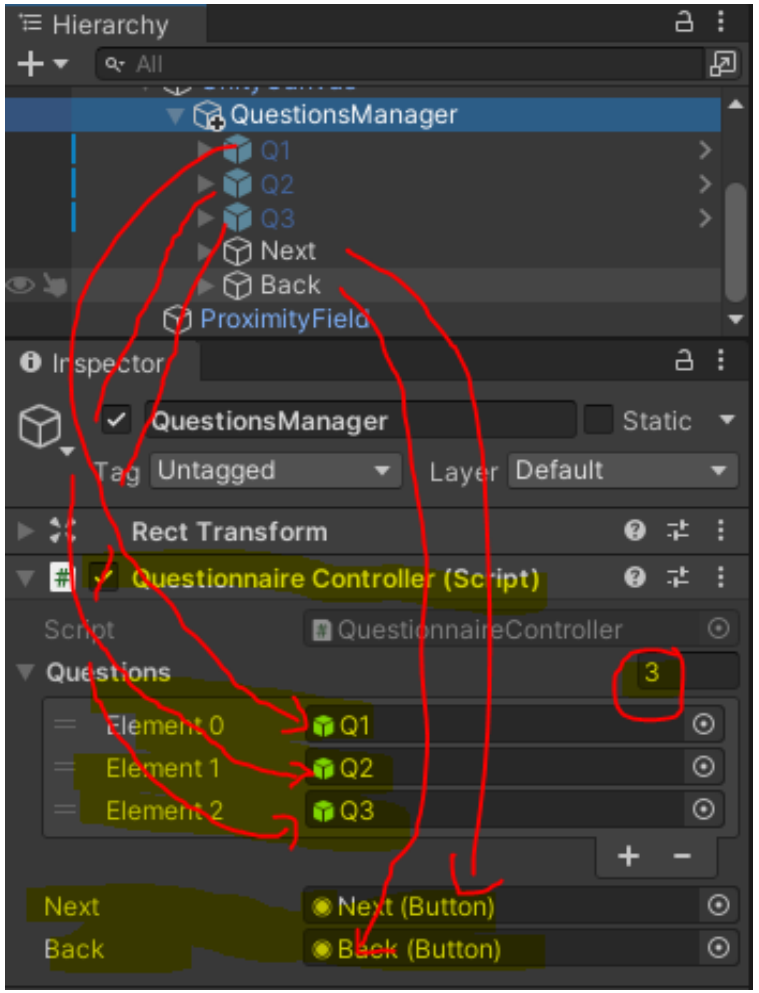
Questionnaire

QuestionnaireController

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

Change questions number to 3

Drag and drop



Questionnaire

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

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

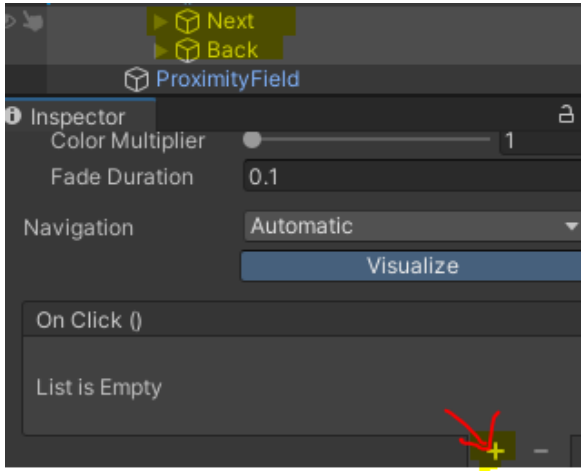
```
private void Start()
{
    currentQuestion = 1;
    SetActiveQuestion(currentQuestion);
}
```

Questionnaire

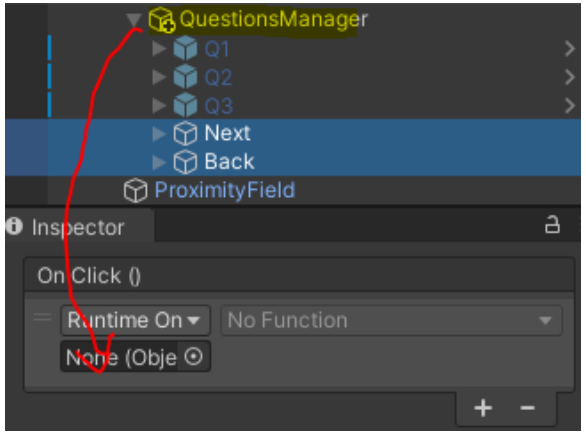
GEM2022

QuestionnaireController

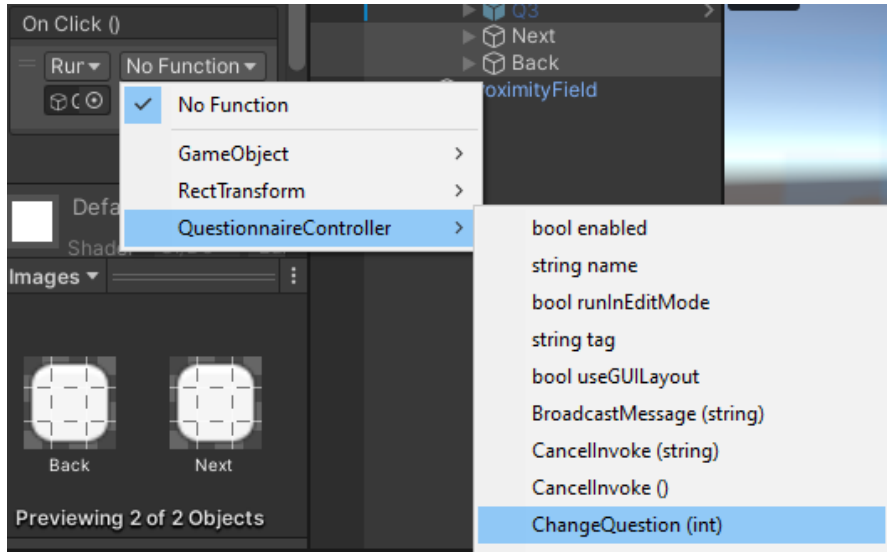
11) Select Next and Back in On Click hit +



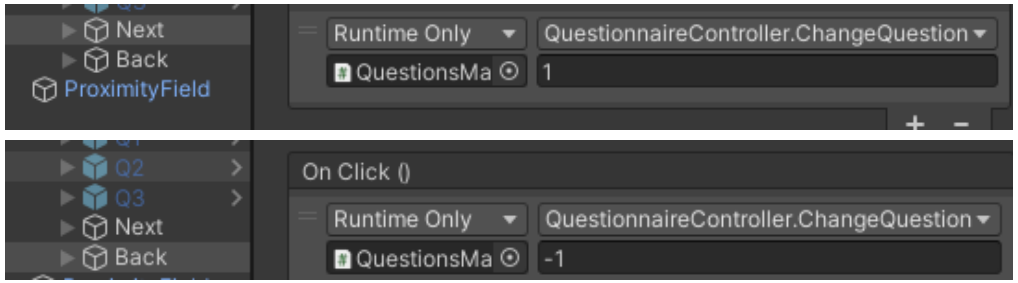
12) Drag and drop QuestionManager to that none space



13) Select the function ChangeQuestion



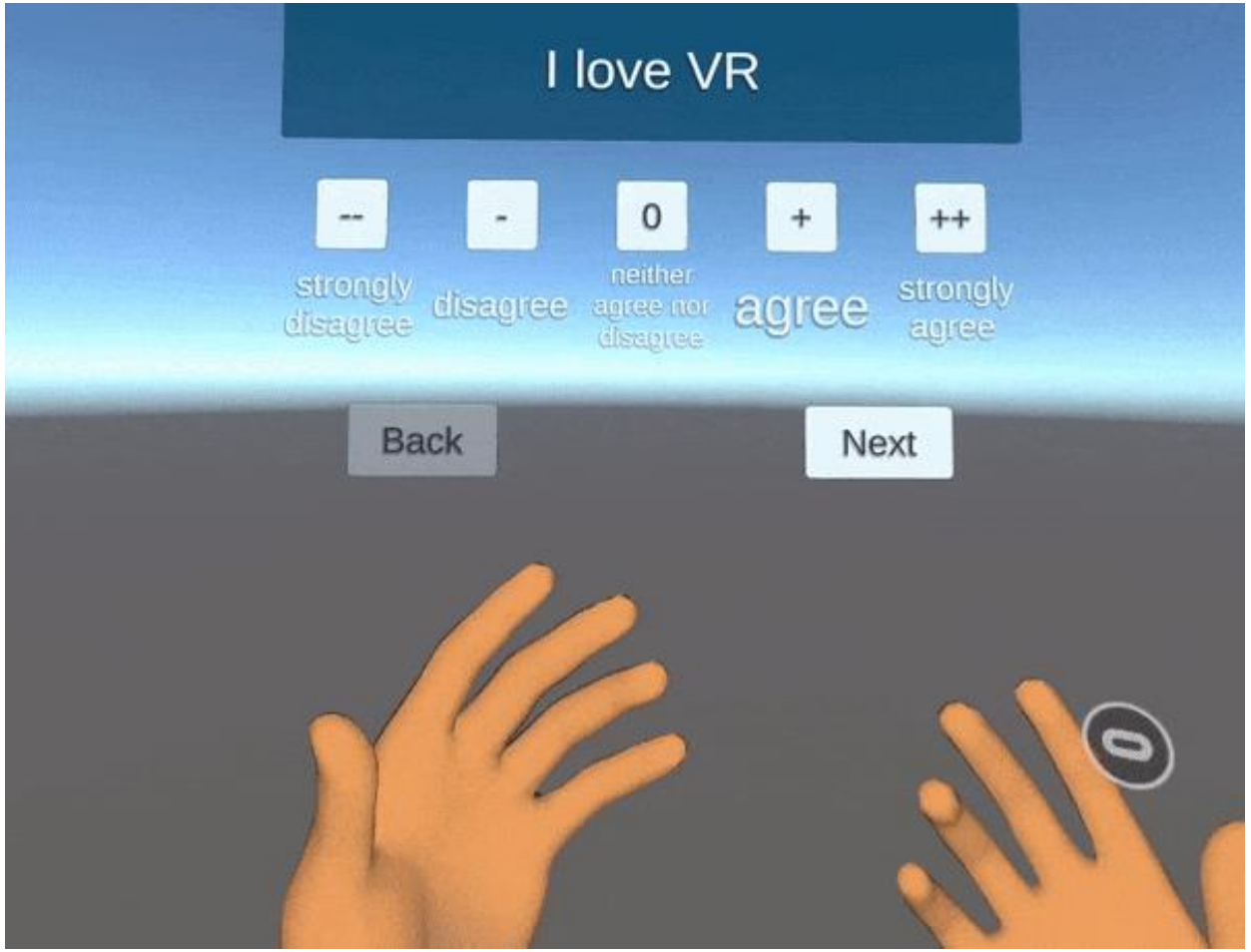
14) Change the int value for Next button to 1 & for Back button to -1



Questionnaire

GEM2022

Test time!



Questionnaire

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]
    0 references
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

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

    0 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;

        DataHolder.Answers[QuestionID - 1] = answer;
    }
}
```

Questionnaire

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

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

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

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();
}
```

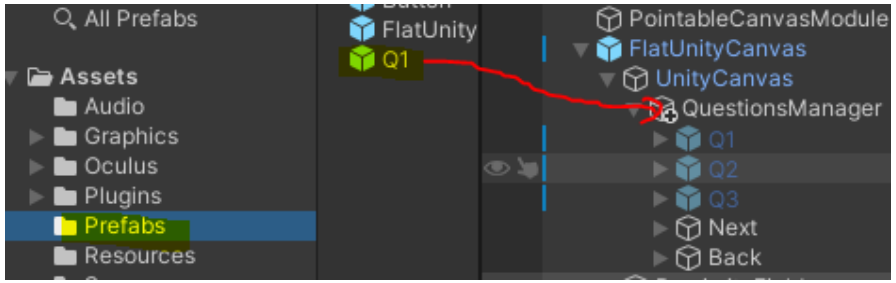
Questionnaire

GEM2022

Final page

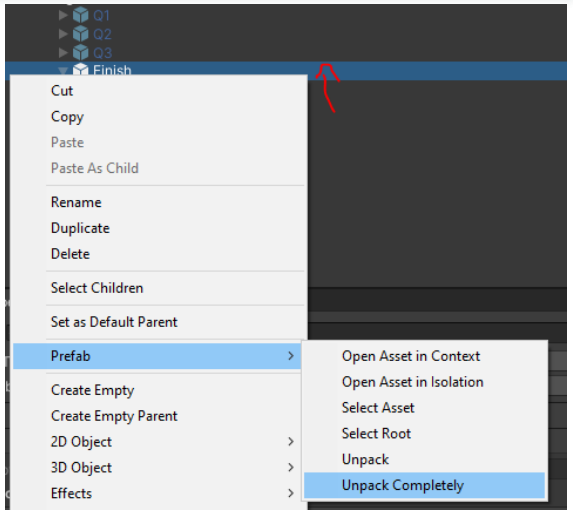
1) there is a lot of ways to trigger, but to be polite final page of this questionnaire must be added to say thanks and give an option to submit answers or go back and change them

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 Completly

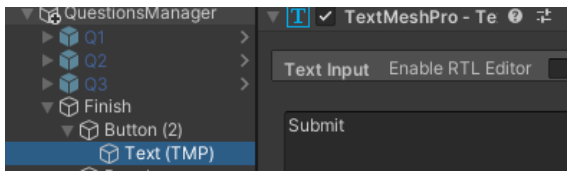
3) Delete all the buttons except the middle one [Button (3)] and Labels

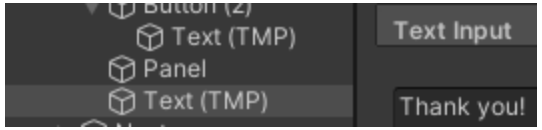


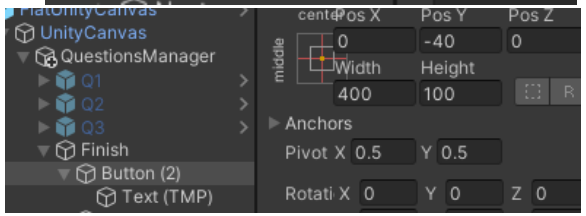
4) On the child Text (TMP) Change Text Input to Submit

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





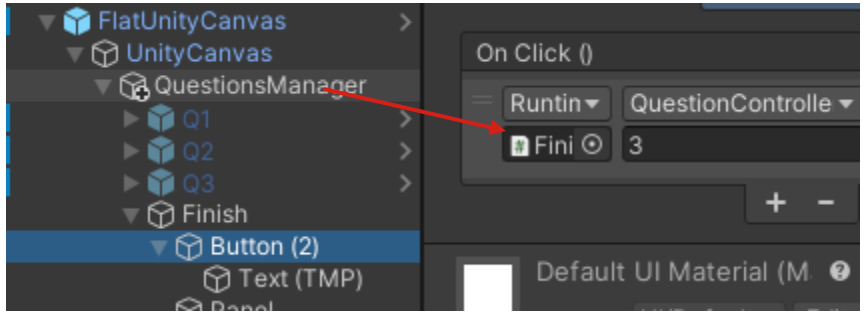


Questionnaire

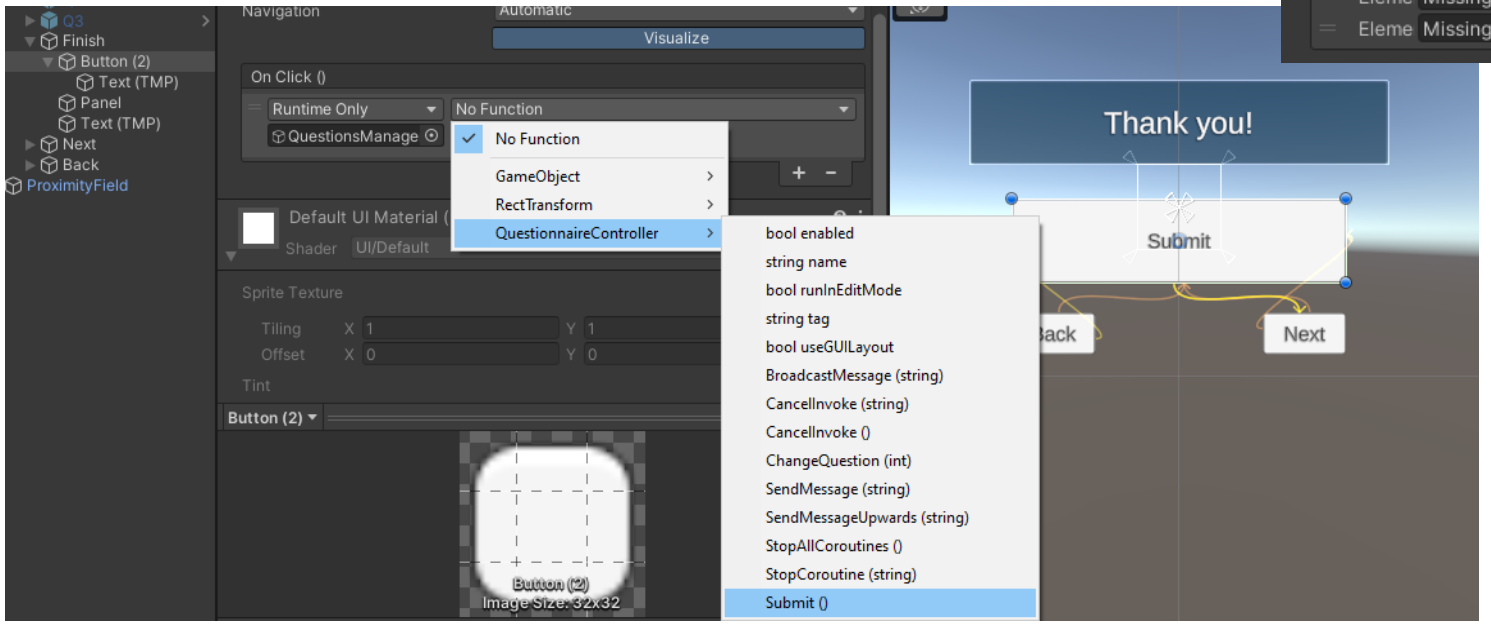
GEM2022

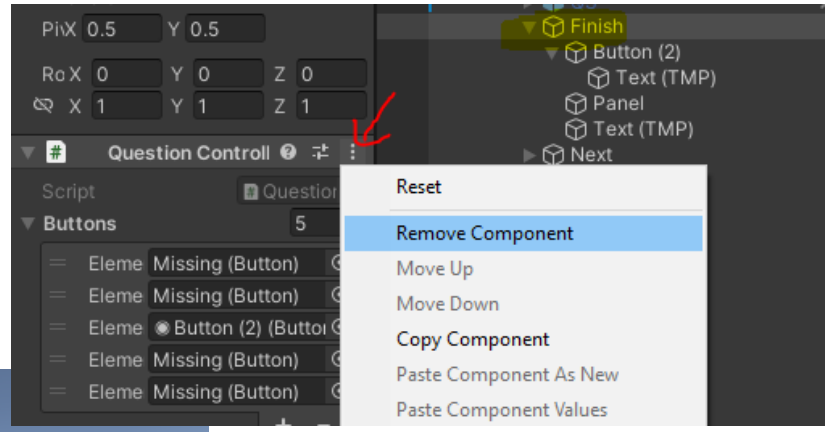
Final page

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



8) Select Submit()





9) In Finish GameObject

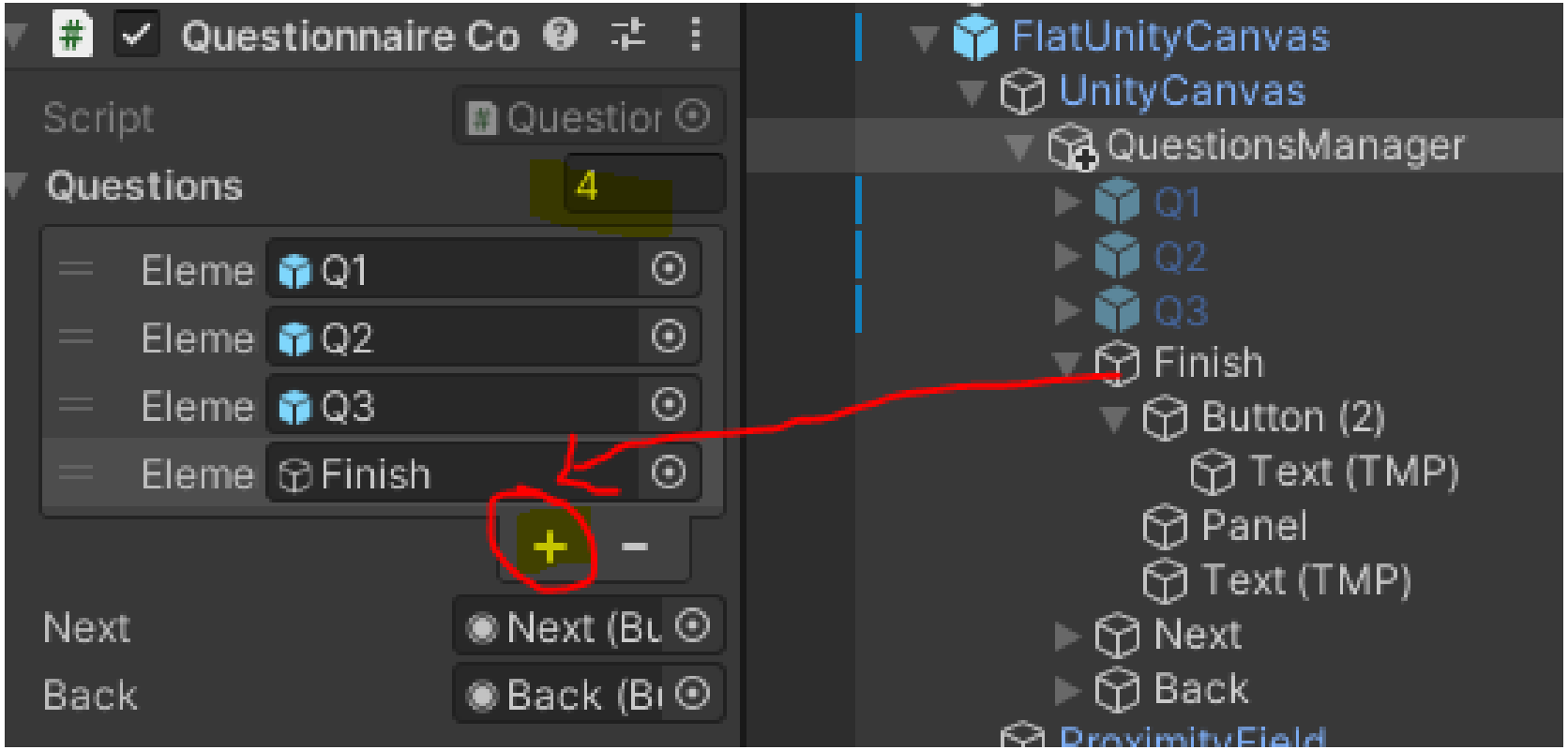
Remove: QuestionController from the

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;
```

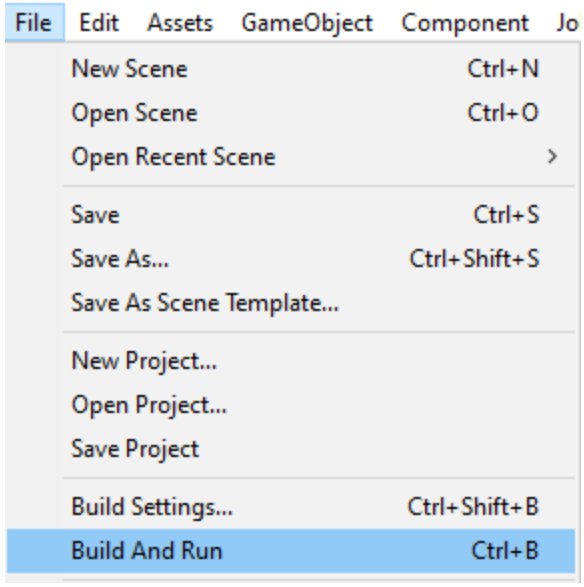
and at the end of SaveFile method add:

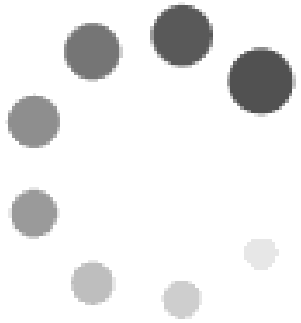
```
public static void SaveFile()
{
    string path = GetPath();

    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

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

GEM2022

Making something to press

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

Rename it to CylinderButton

2) Add an Empty GO as a child of CylinderButton

Rename it to ProximtyField

3) Add another Empty GO as a child of CylinderButton

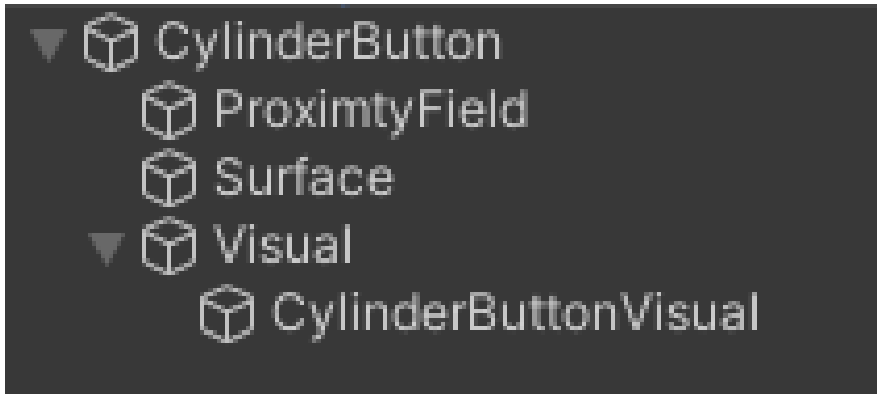
Rename it to Surface

4) Add one more

Rename it to Visual

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

Rename it to CylinderButtonVisual

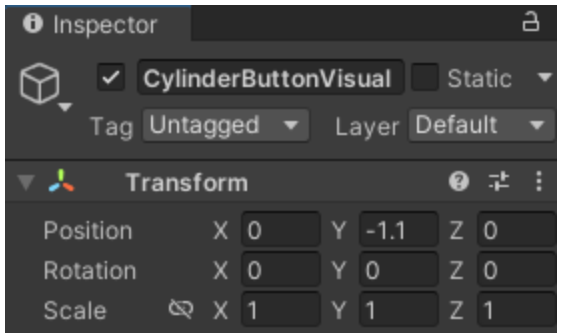


Wearable button

GEM2022

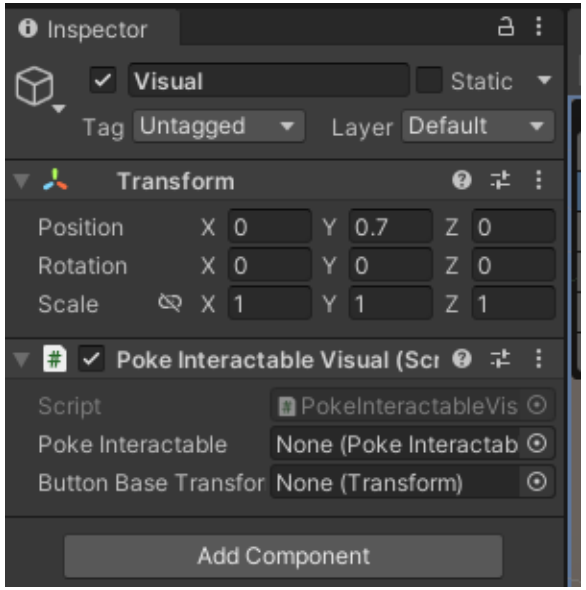
Making something to press

1) in CylinderButtonVisual
Change Position Y to –1.1



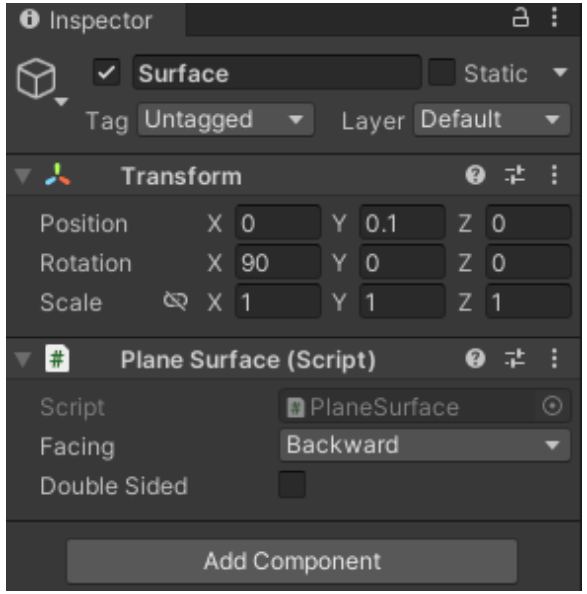
2) Change Visual
Position Y to 0.7

Add Component:
Poke Interactable Visual



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

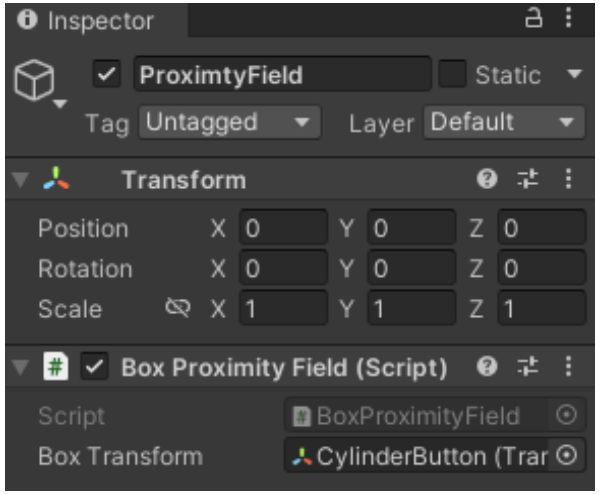
Add Component:
Plane Surface



4) To ProximityField

Add Component:
Box Proximity Field

Drag and drop
CylinderButton GO to
Box Transform Field



Wearable button

GEM2022

Making something to press

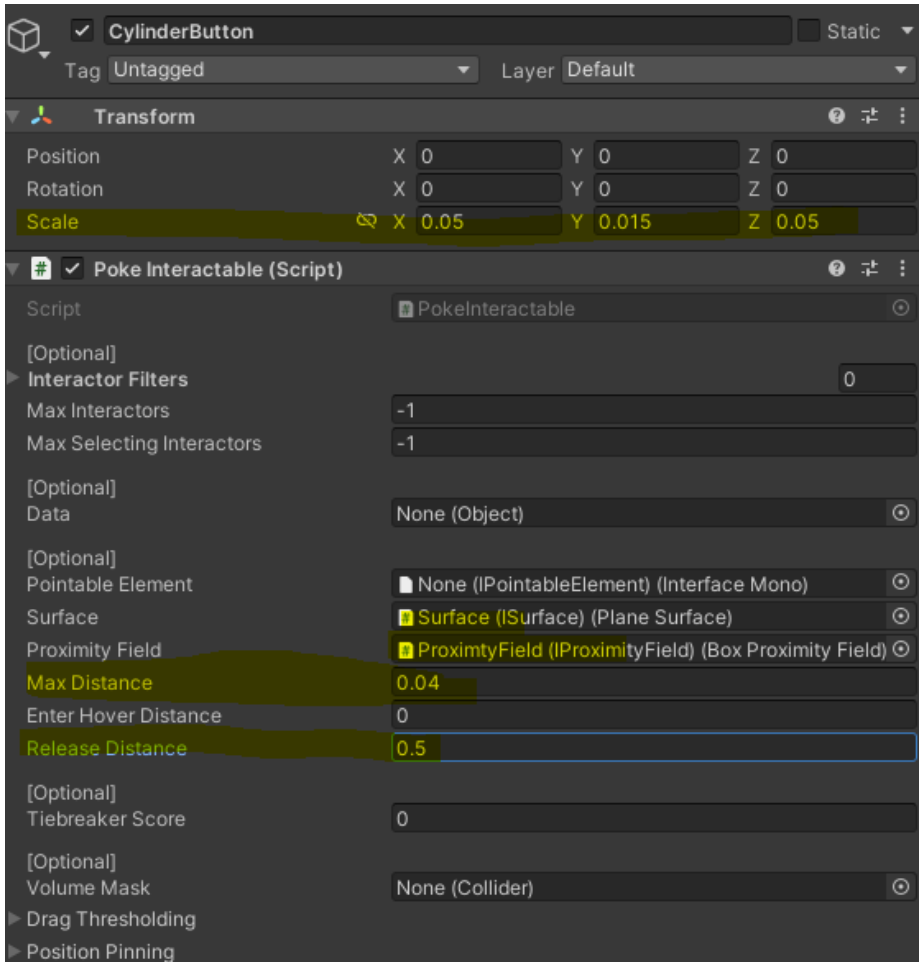
5) To CylinderButton GO

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

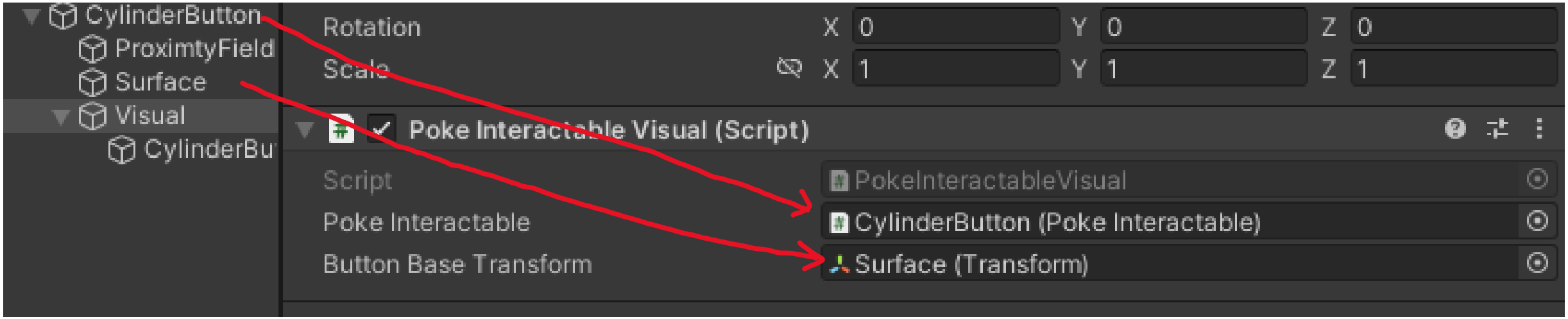
GEM2022

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

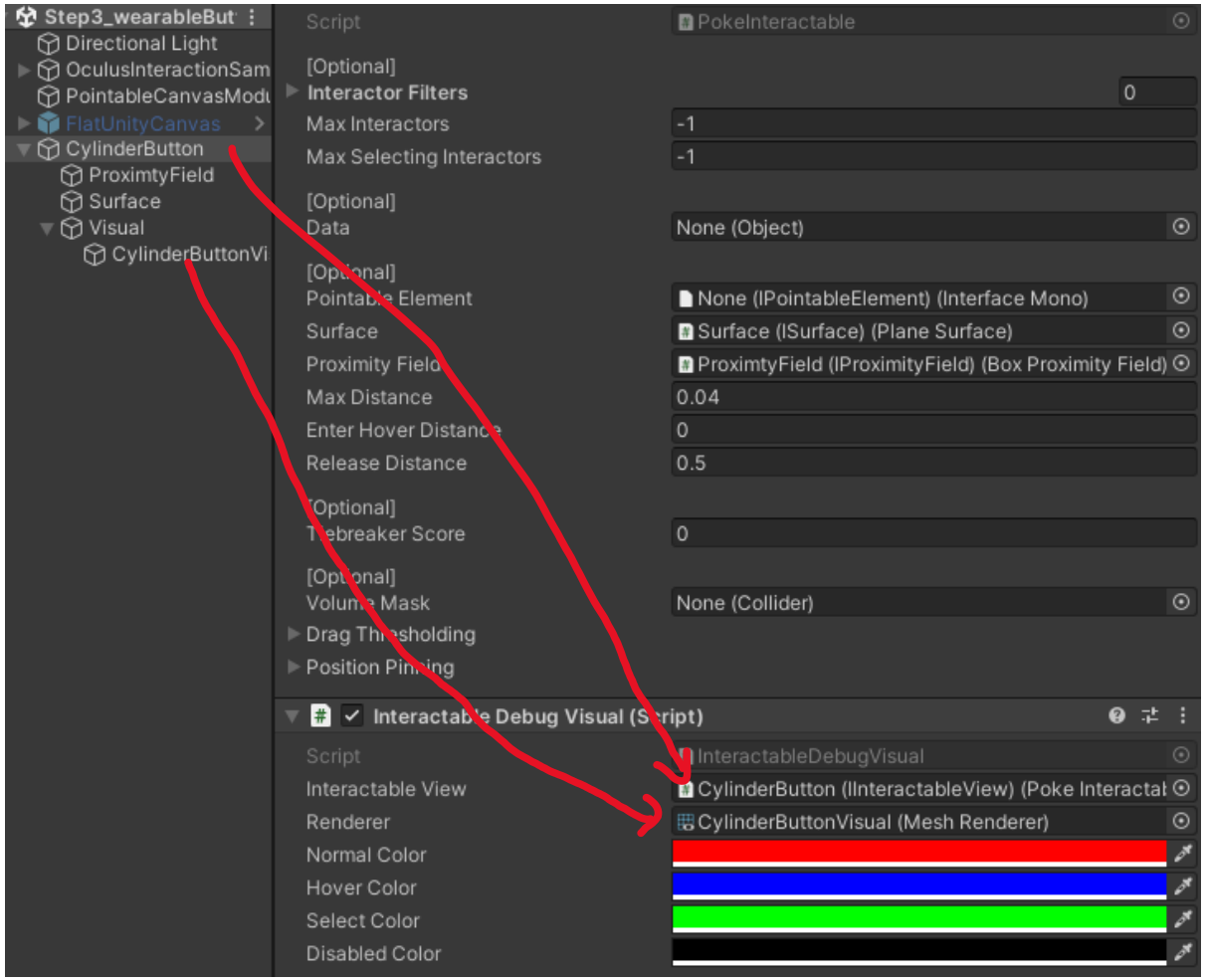
GEM2022

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

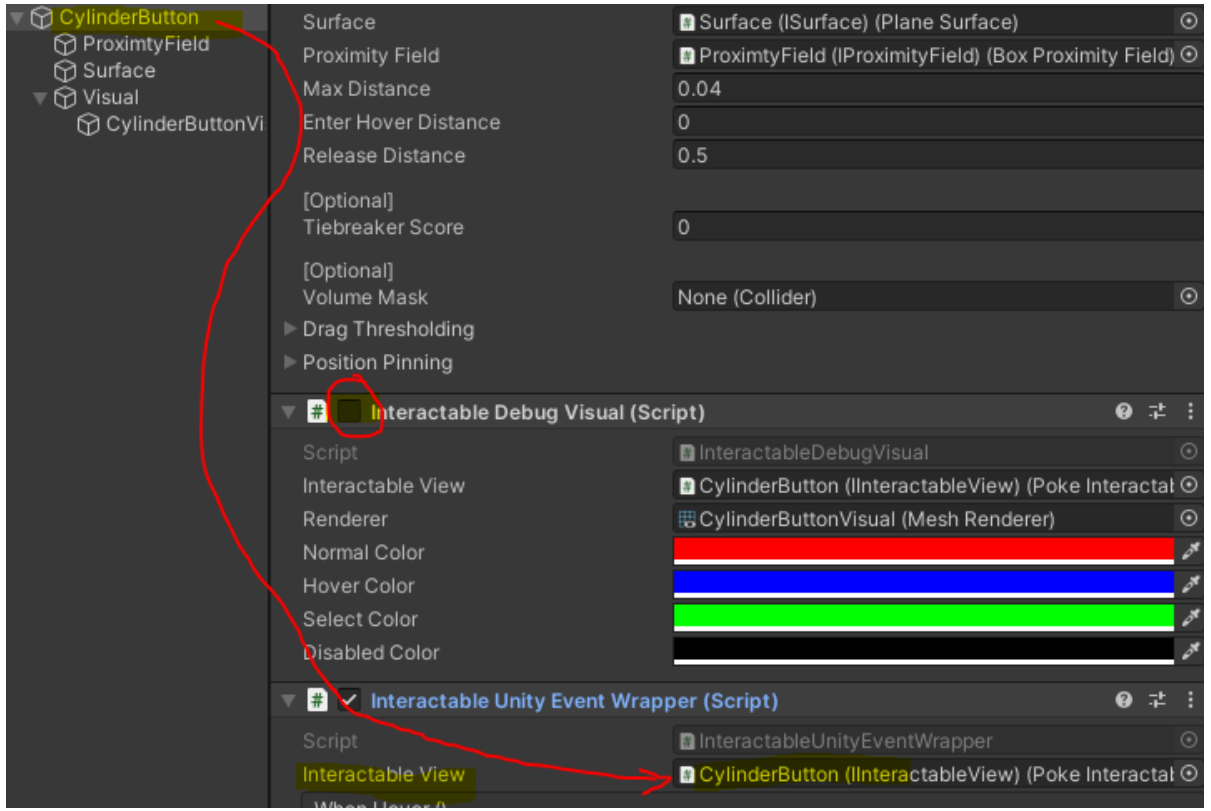


Wearable button

GEM2022

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

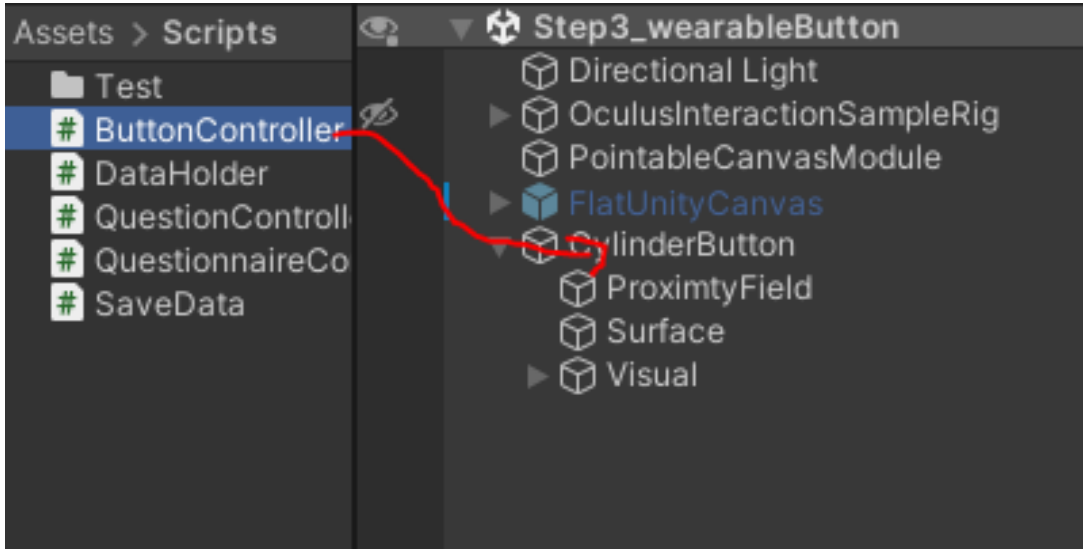


Wearable button

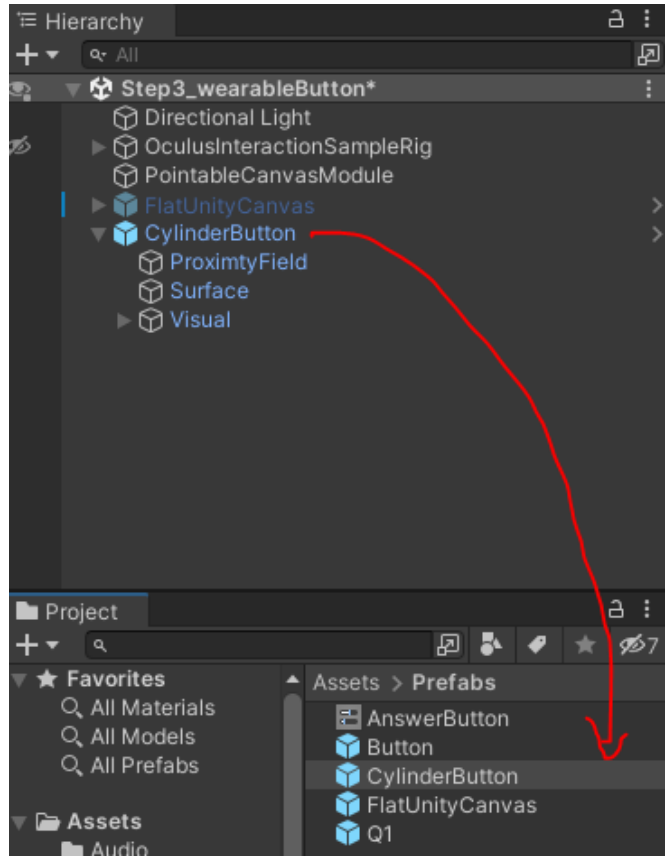
GEM2022

Prefabricate it

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



2) Drag and drop CylinderButton GO into Prefabs folder

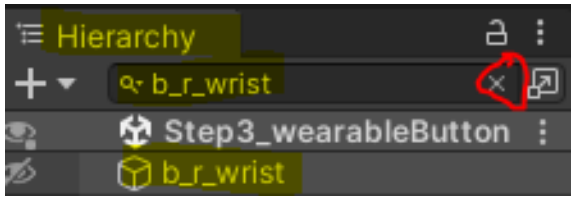


Wearable button

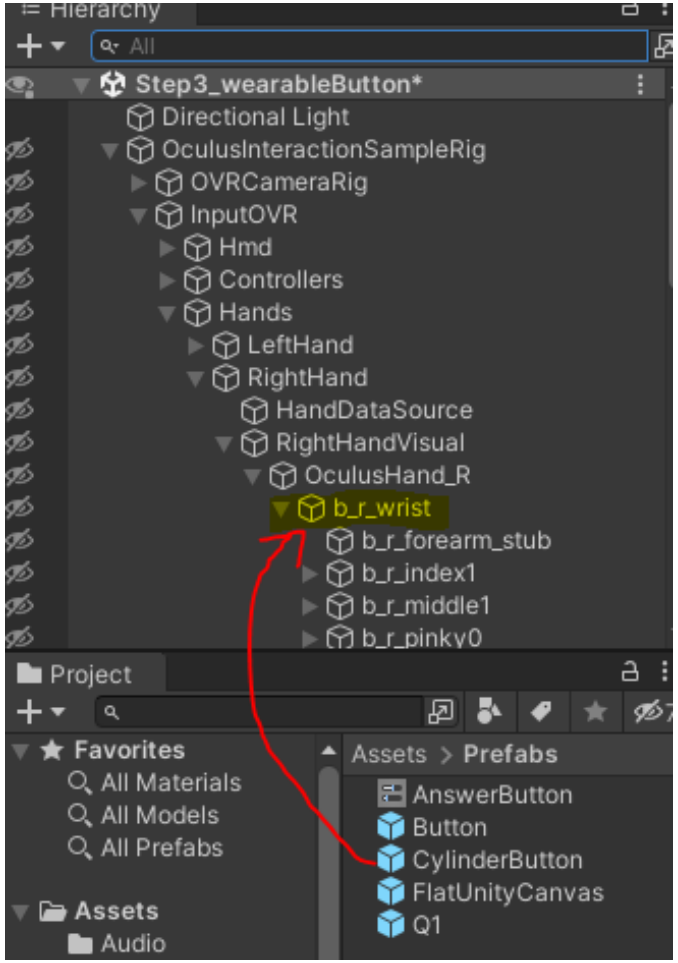
GEM2022

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

GEM2022

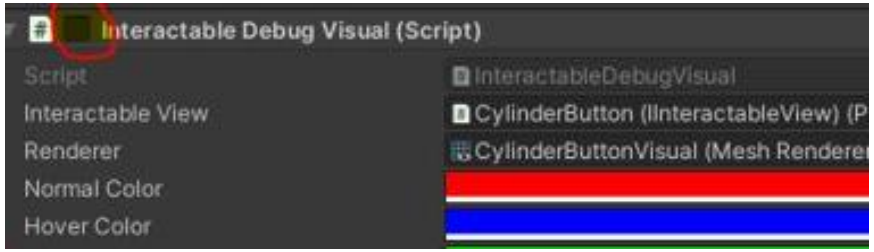
Making something to press

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

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

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;

    Unity Message | 0 references
    private void Start()
    {
        meshRenderer = this.GetComponentInChildren<MeshRenderer>();
        meshRenderer.material.color = turnedOffColor;
    }
    0 references
    public void Selected()
    {
        meshRenderer.material.color = turnedOnColor;
    }
}
```

Wearable button

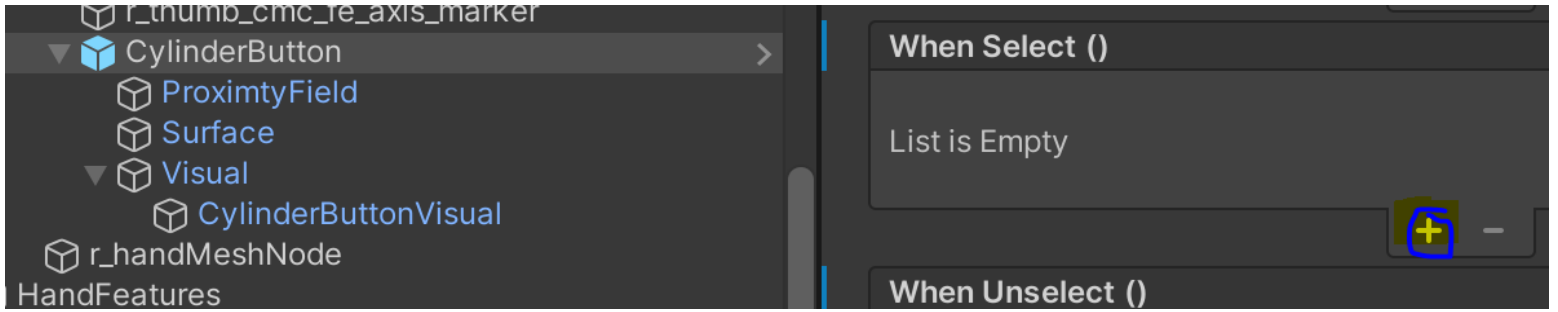
GEM2022

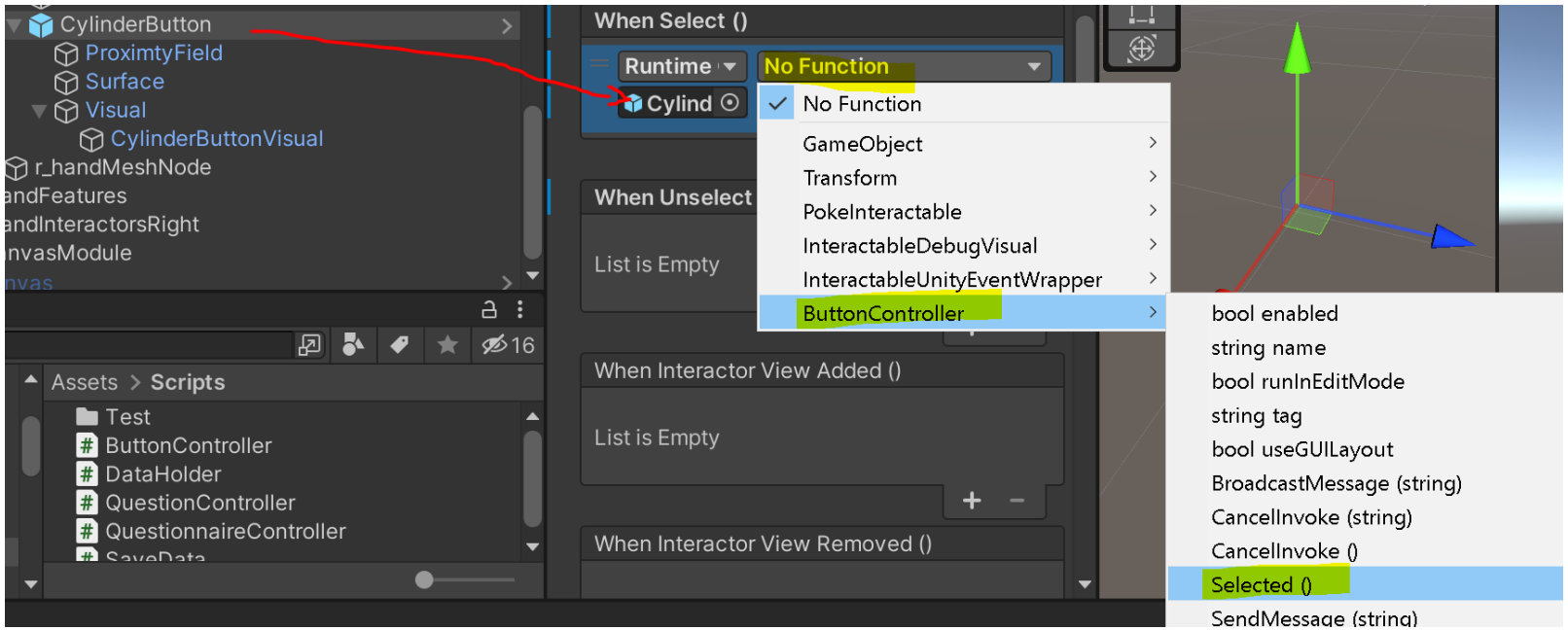
Hook it up

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()



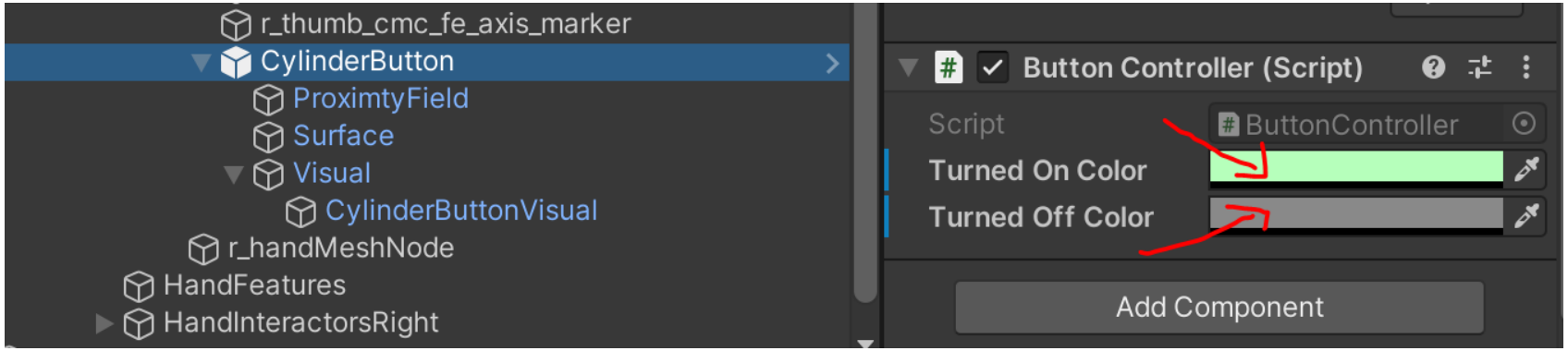


Wearable button

GEM2022

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

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

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

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

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

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

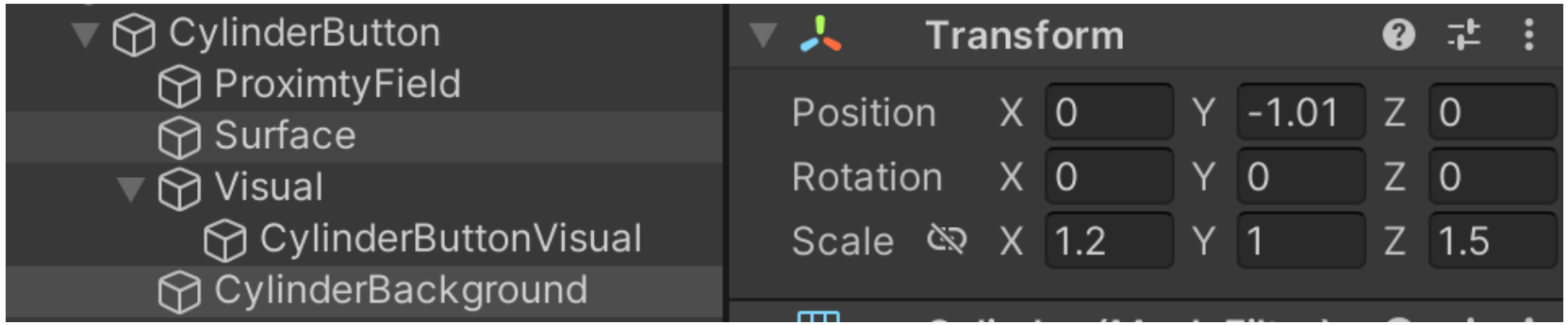
GEM2022

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

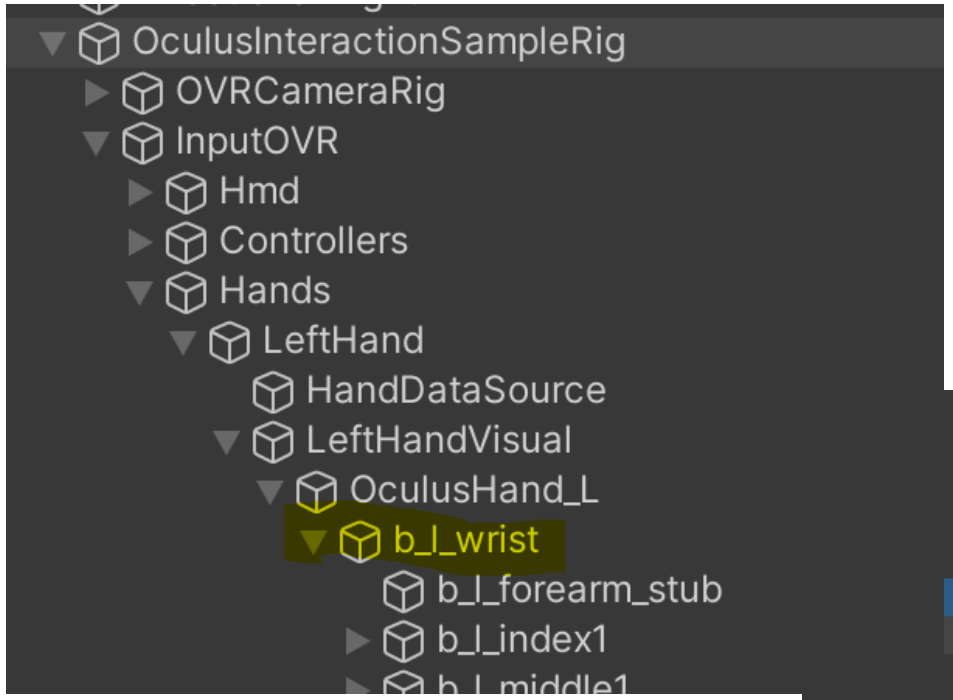
Time to slide

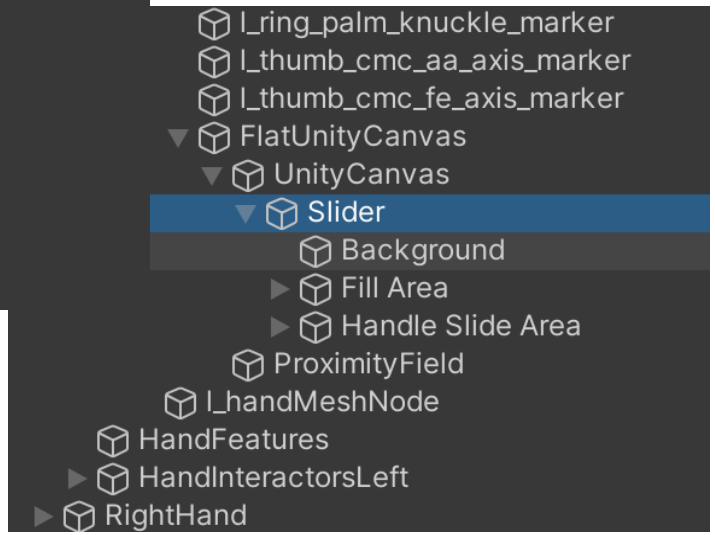
1) Drag and drop on b_l_wrist GO

2) Unpack it

3) RMB -> UI -> Slider

4) Move that Slider GO to UnityCanvas delete and the standard Canvas



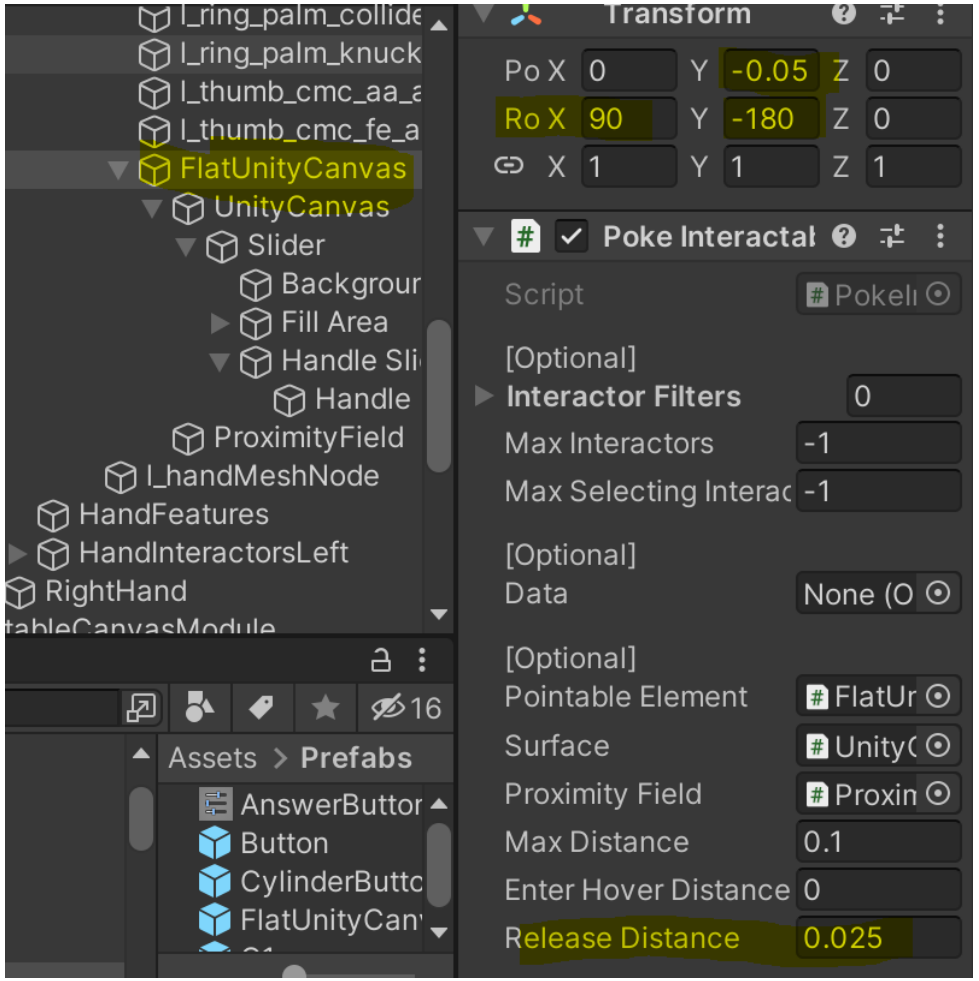


Slider

Time to slide

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

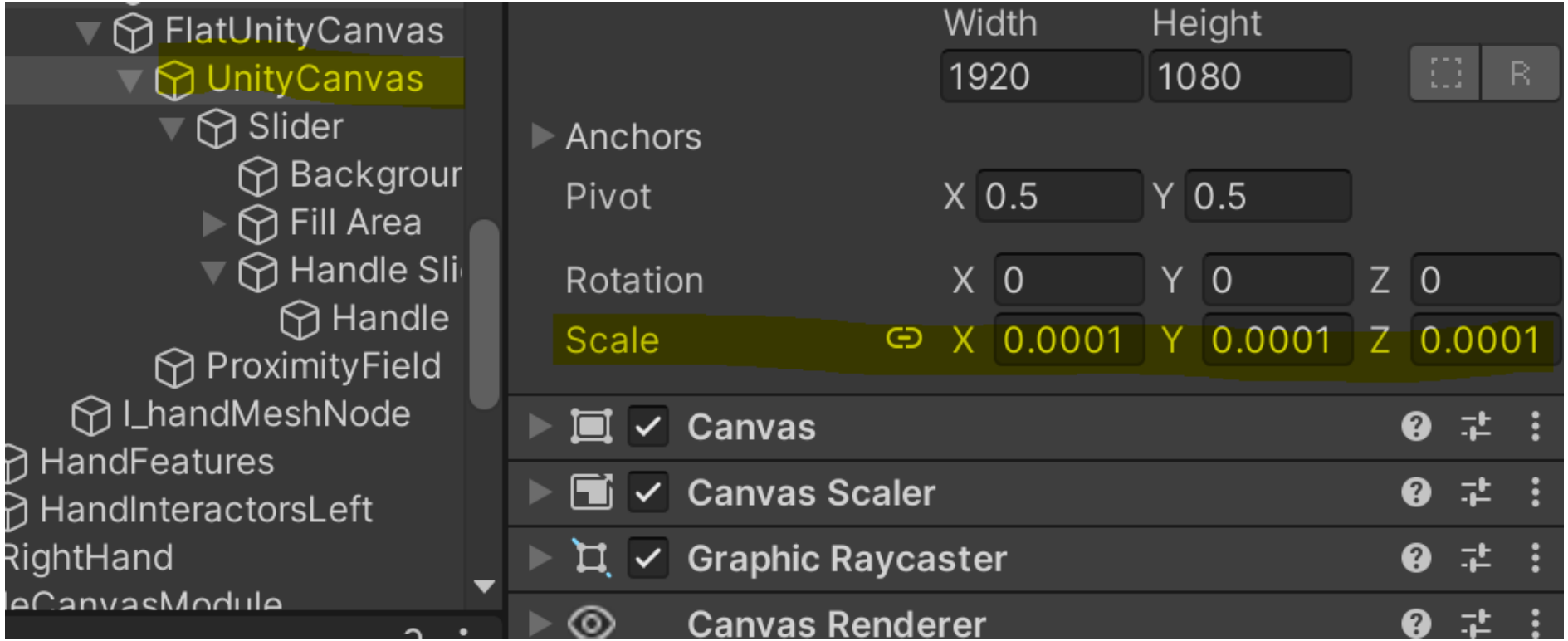


Slider

GEM2022

Time to slide

6) on UnityCanvas
RectTransform
Change
Scale to 0.0001



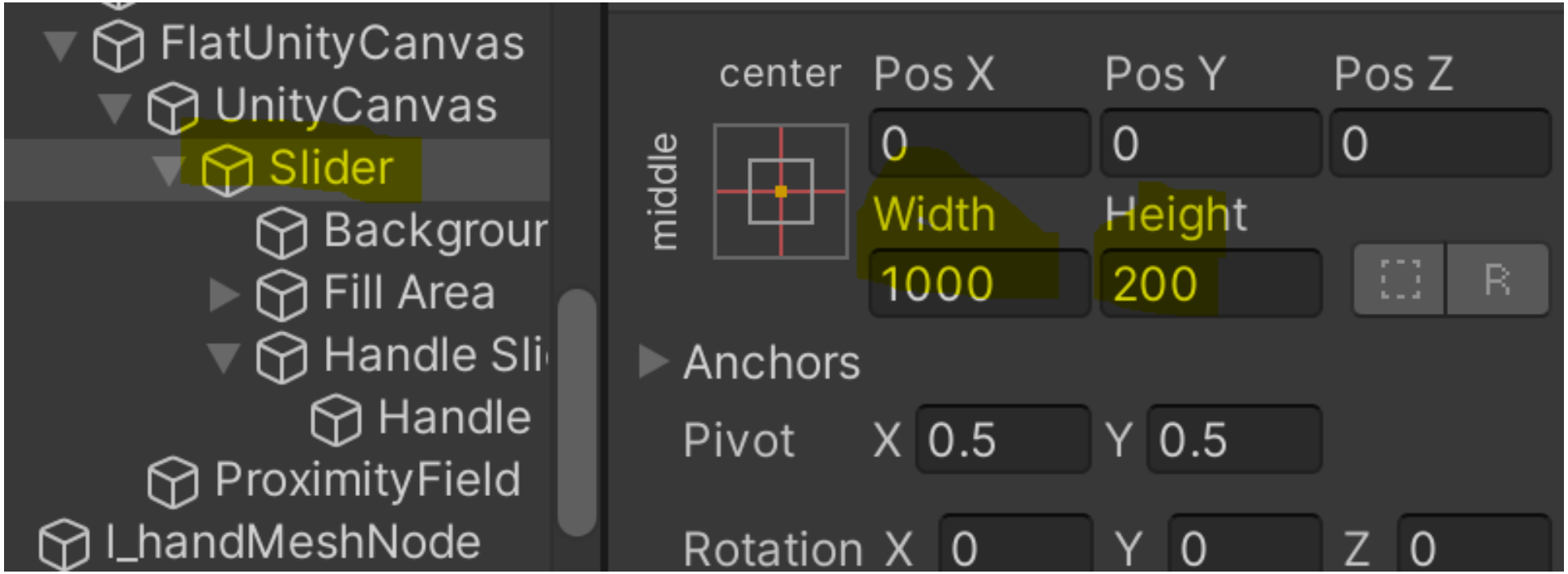
Slider

GEM2022

Time to slide

7) on Slider Change

Width to 1000
Height to 200



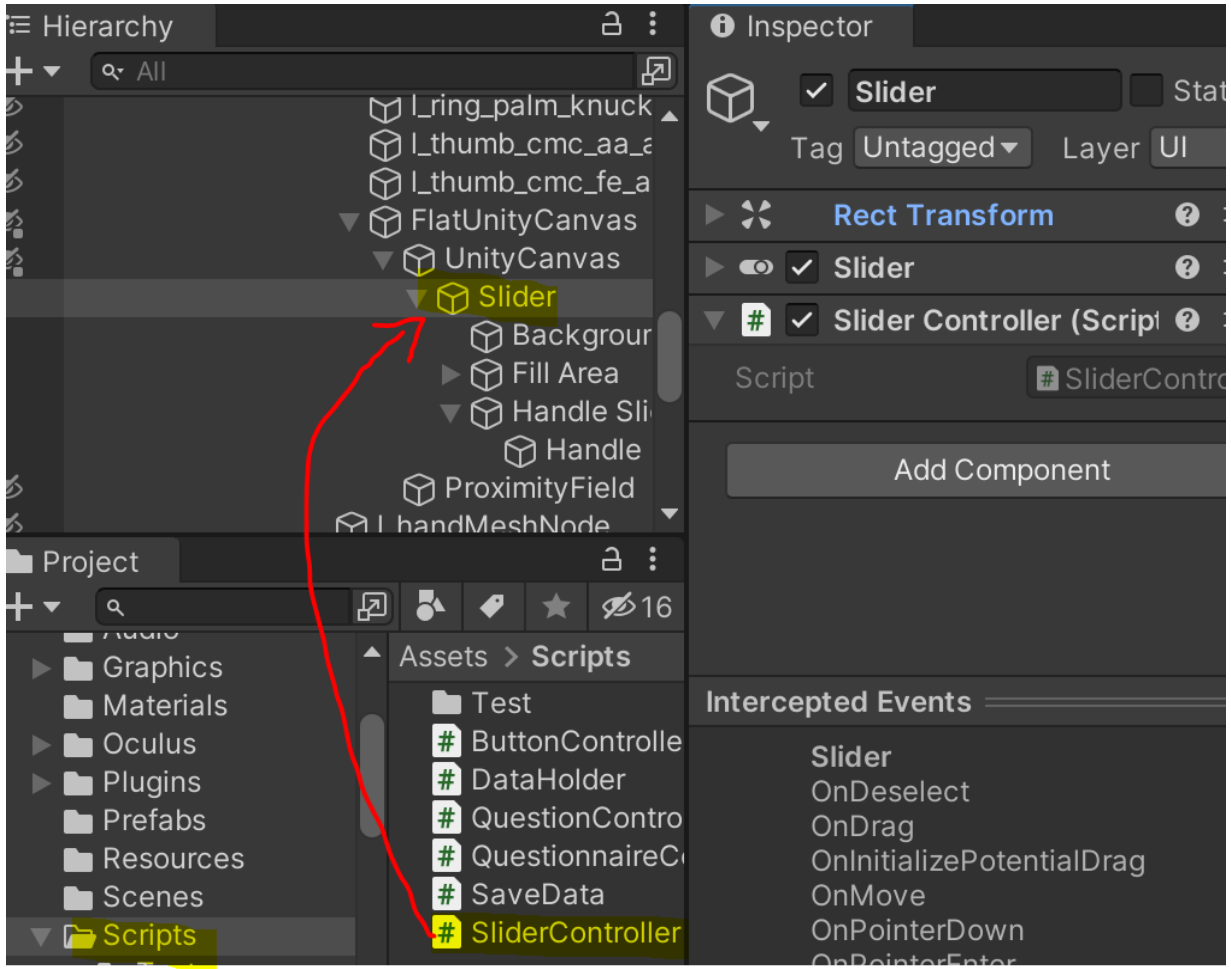
Slider

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

Make it hot!

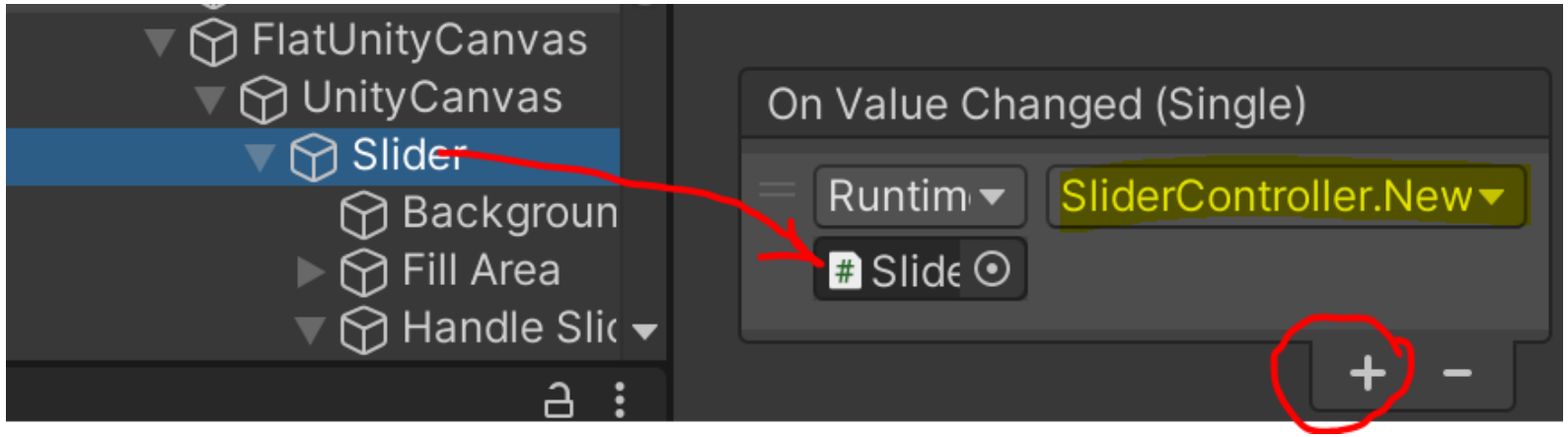
1) Add UI namespace
2) Create fields slider and an array of Images
3) at the start assign Slider class of this component to slider variable
And all its Image components to sliderImages array
4) public method NewValue will loop through this list when value will change and lerp between blue and red

```
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

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

Make it SUPER hot!..saving data

4) to SliderController
add UnityEngine.EventSystems 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

Make it SUPER hot!..saving data

4) to SliderController
add UnityEngine.EventSystems 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!

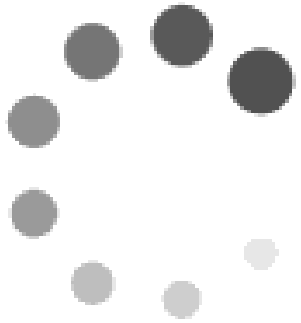
Enable FlatUnityCanvas

Build and run

Done!

Mission accomplished!

Any questions?



Done!

Mission accomplished!

Thank you,
feel free to contact me at
Krzysztof.X.Szczurowski@mytudublin.ie

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

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

Next?

Final page

Check if all answers completed?
Audio?
===
Curved canvas?
IK?
======
Web XR?