1. **Overview**

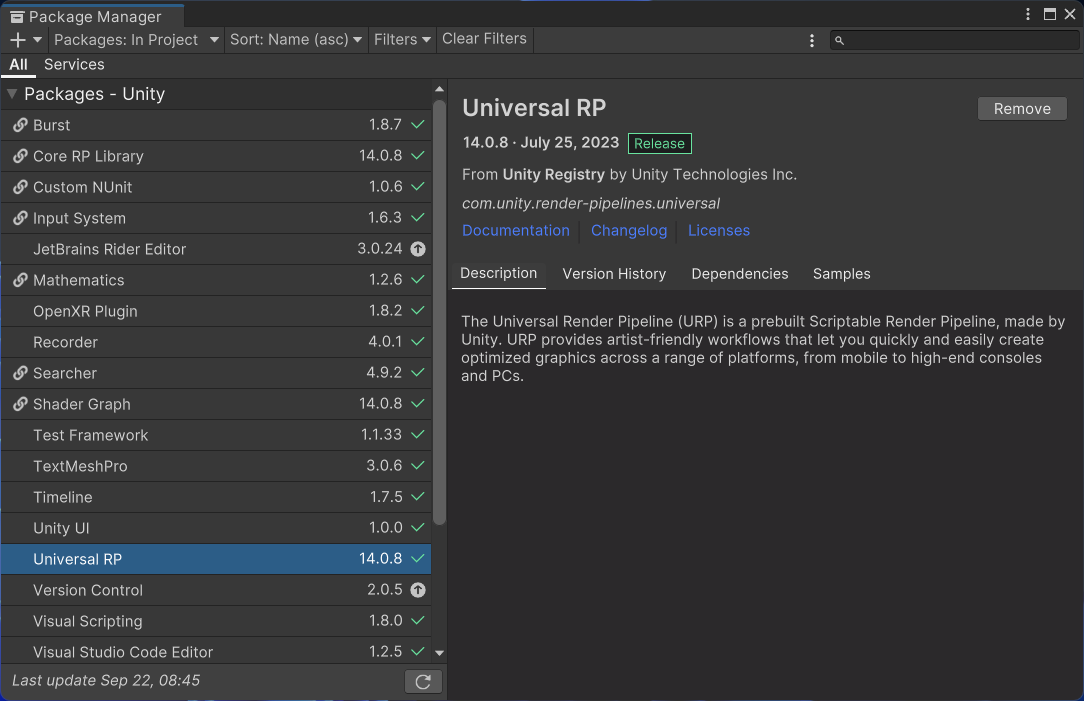
The plugin developed by smarteye.id aims to assist virtual reality developers in the application development process. This plugin suits developers with skill levels ranging from beginners to intermediate.

During the development process, two options are available to initiate this project. Firstly, you can utilize a pre-prepared project and replace its content to align with your specific requirements. The second alternative is to commence a project from scratch and add features already available in prefabs. The time required to create the application will be adjusted according to the complexity of the intended application.

1. **Requirements**
2. Utilize XR Interaction Toolkit version 2.4.3.
3. Implement Input System version 1.6.3.
4. It is highly recommended to employ OpenXR version 1.8.2
5. Incorporate TextMeshProUGUI.
6. Adopt the Universal Render Pipeline.
7. **Installation instructions**

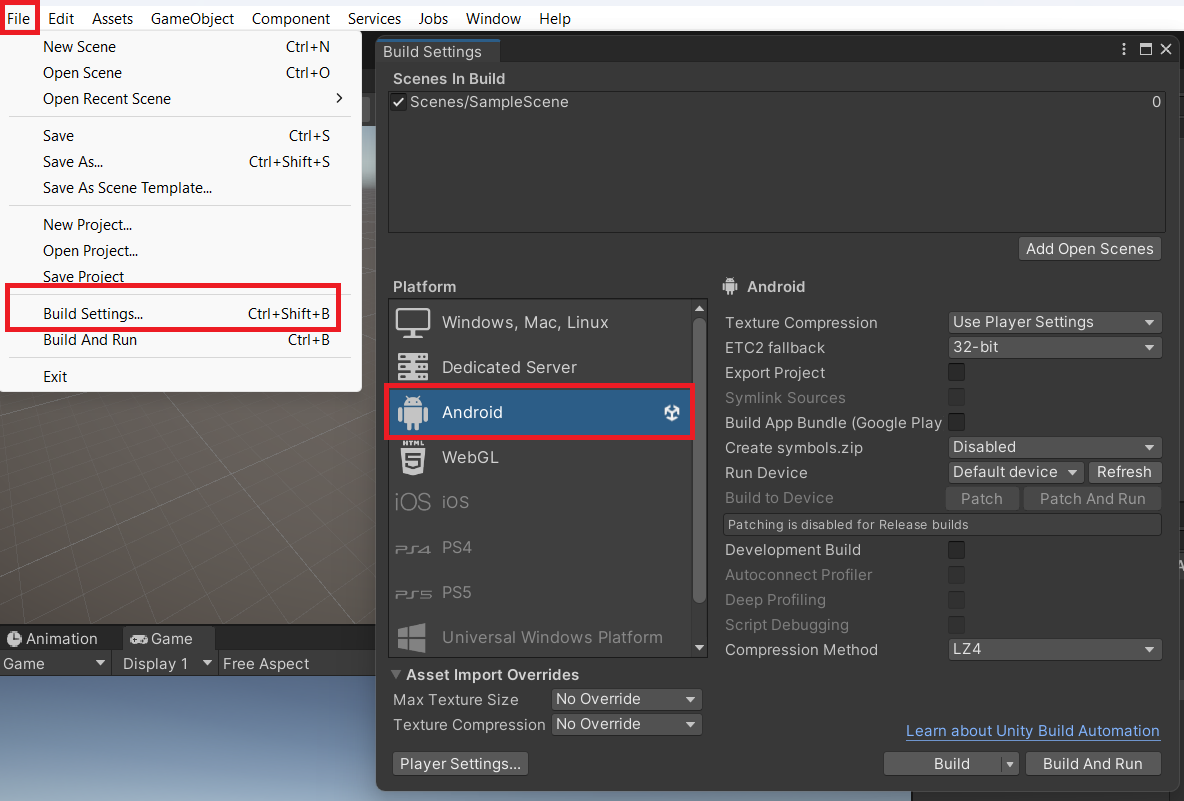
* **Utilize the Universal Render Pipeline**

When developing a VR application using the Smarteye framework, select a project supporting the Universal Render Pipeline



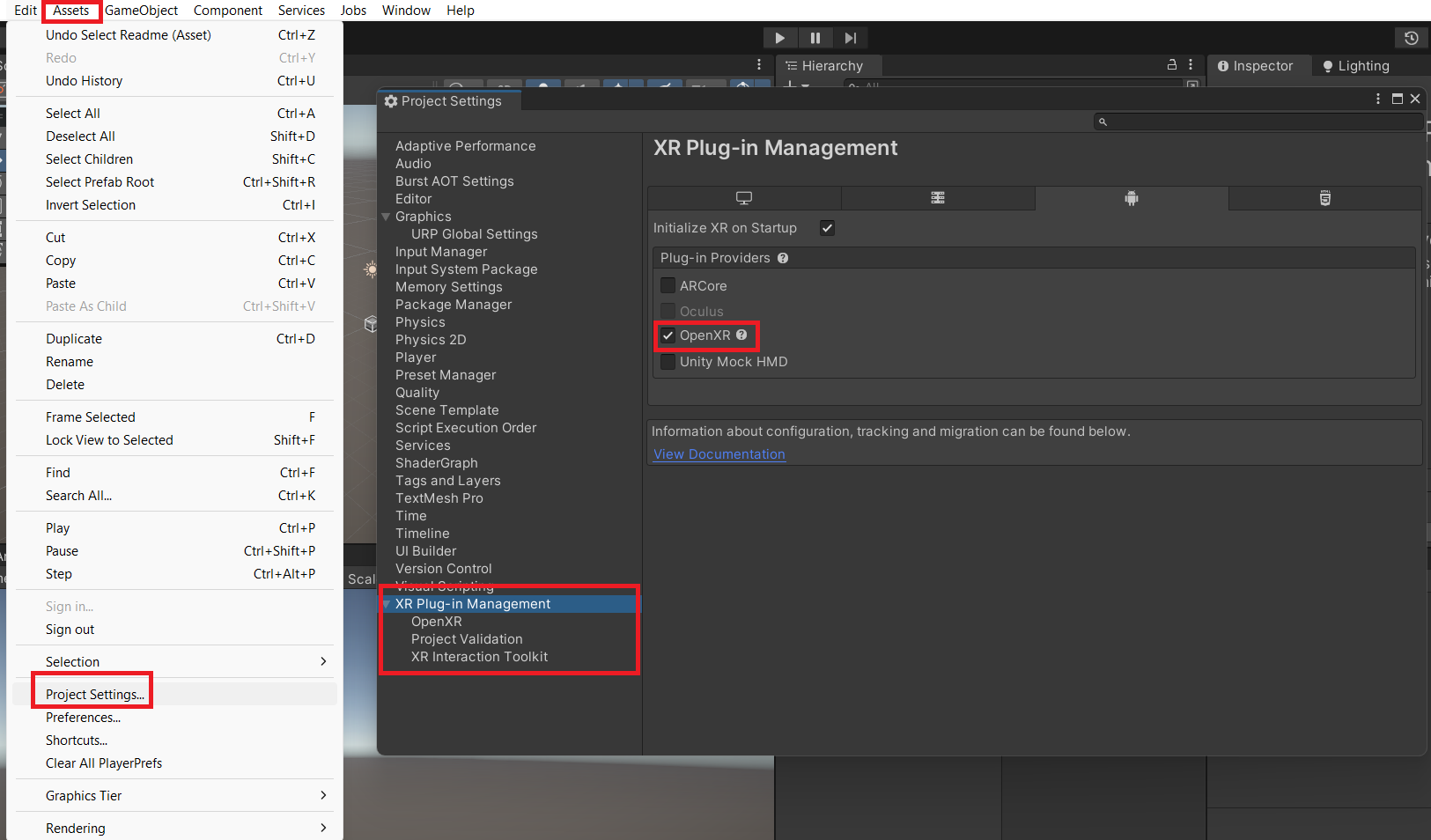
* **Change the Platform to Android**

Access the project settings menu and choose the option to modify the platform to Android.



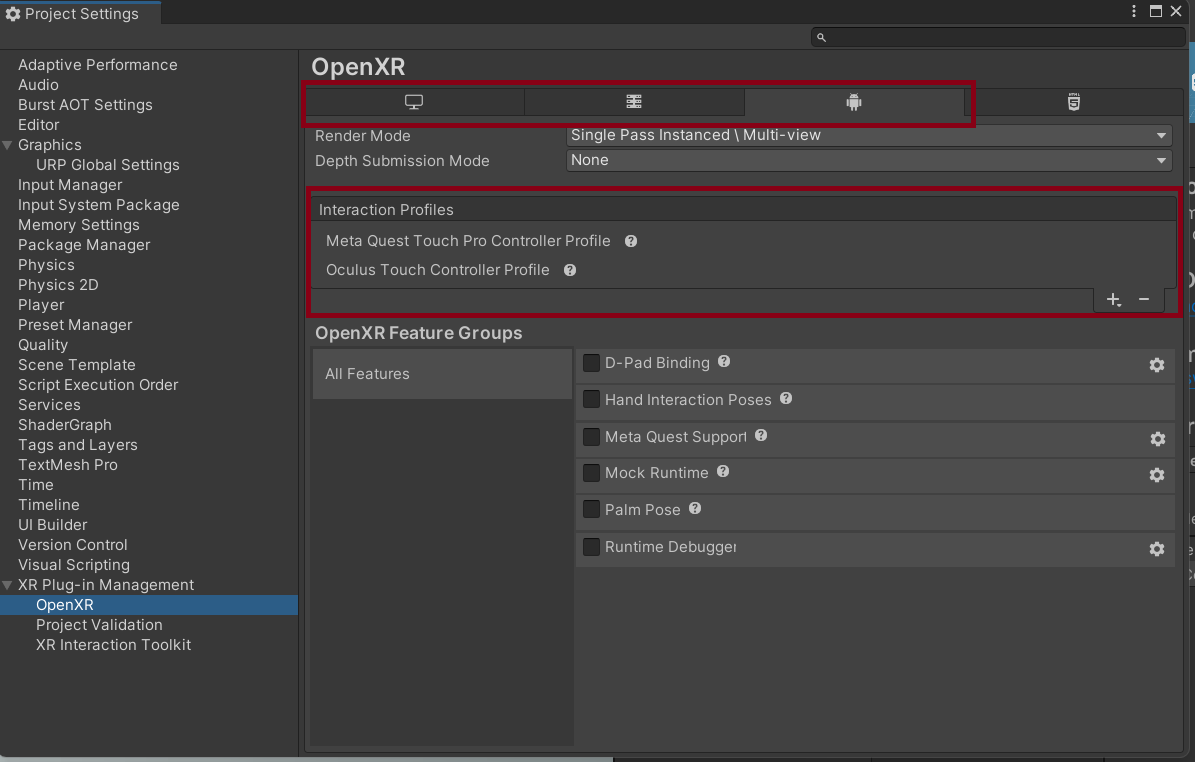
* **Install XR Plugin Management**.

Next, access the project settings (project settings) to install XR Plugin Management, which is essential for the project. You have the option to select a plugin provider, such as Oculus or Open XR, although we strongly recommend using Open XR.



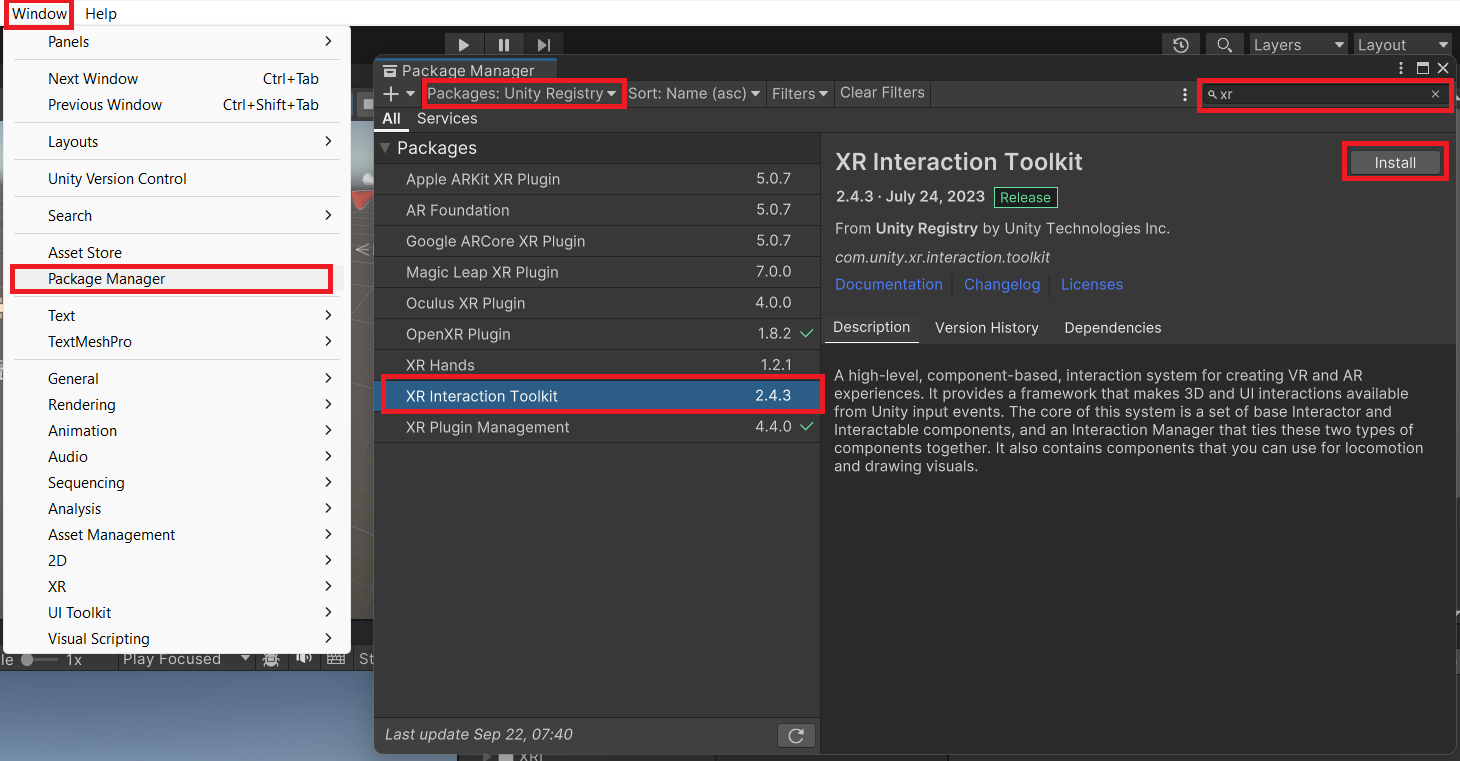
* **Add an Interaction Profile for Oculus Controllers**

Ensure that you have incorporated an appropriate interaction profile for Oculus controllers.

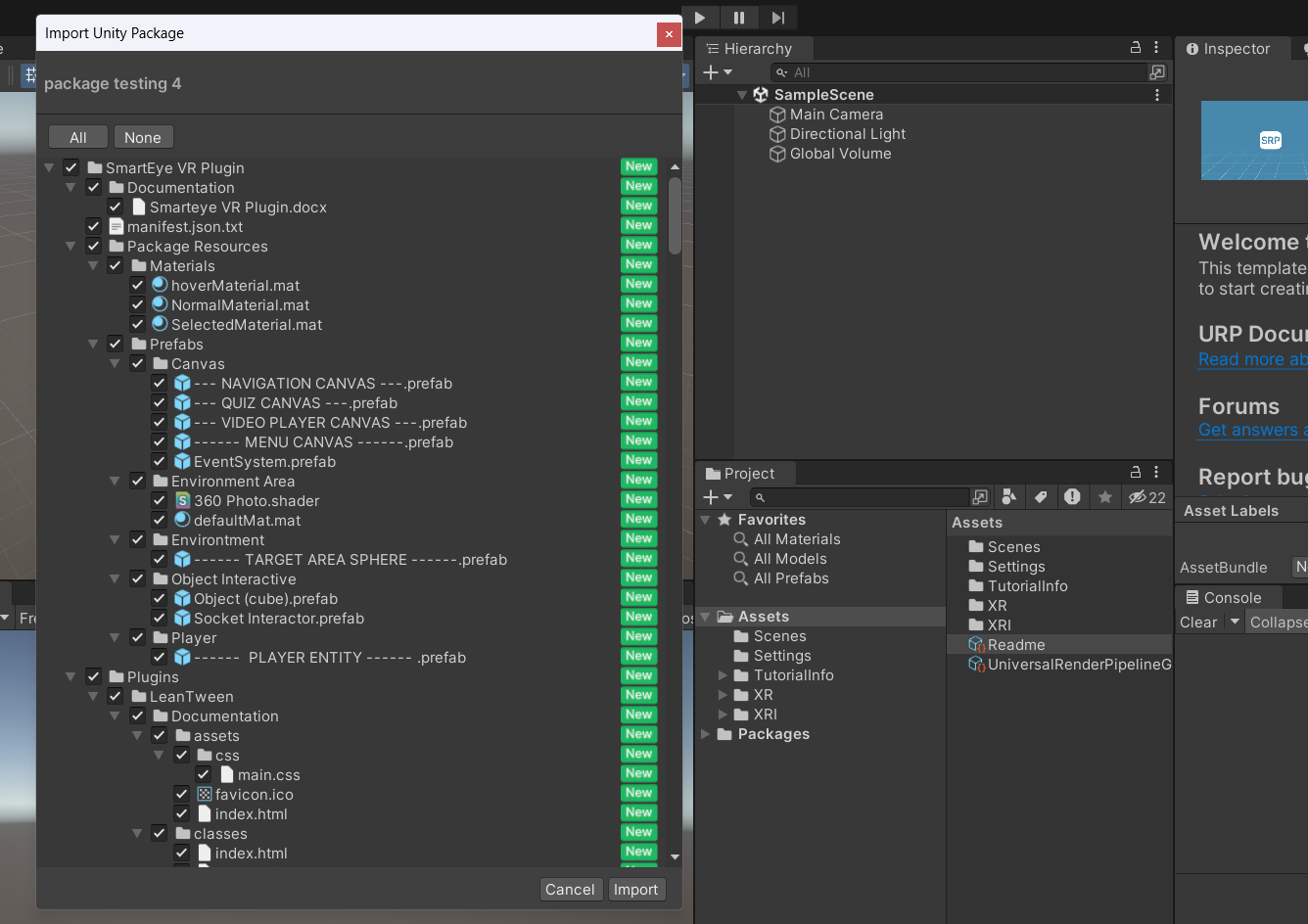


* **Install XR Integration Toolkit**

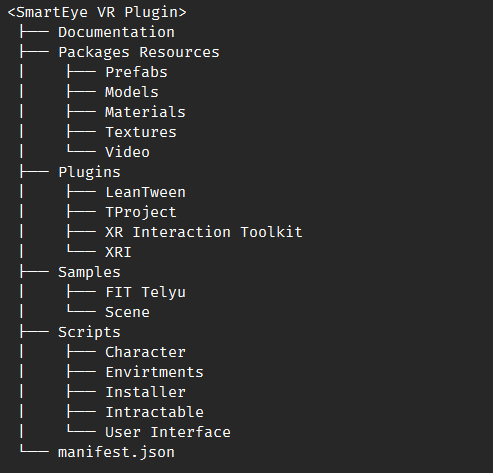
Open the Package Manager window, then install the XR Integration Toolkit available in the Unity registry.



* **Download and Import plugin from Asset Store**



1. **Folder Structure**

****

1. **Glossary**

**1. Player Entity** : A component that allows players to move characters or objects within a game or interactive environment.

**2. Object Interactor** : Objects directly interacting with the player can be held and moved.

**3. Socket Interaction** : An area where players place items; socket interaction includes trigger tools to detect these objects.

**4. Canvas Interaction**

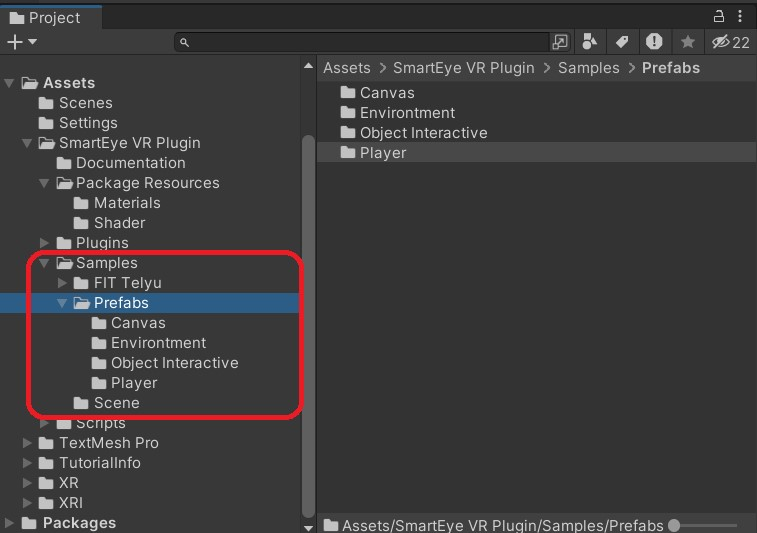
1. **Pop Up Canvas Information** : An element helpful in displaying information in the text. This pop-up information consists of buttons that enable the display of a box panel and content used to present this information.
2. **Menu Canvas** : A menu that can display player navigation, application exit options, and sound settings.
3. **Video Player Canvas** : A panel that can display videos using a Video Player.
4. **Quiz Canvas** : Can display multiple questions and illustrative images with four selectable answers.
5. **Navigation Canvas** : A technique used to manage and facilitate player navigation or movement in a VR environment.

**5. Environment Manager** : This manager controls the flow of the application, such as changing 360 images in various areas, managing transitions between areas, and activating game objects in each area.

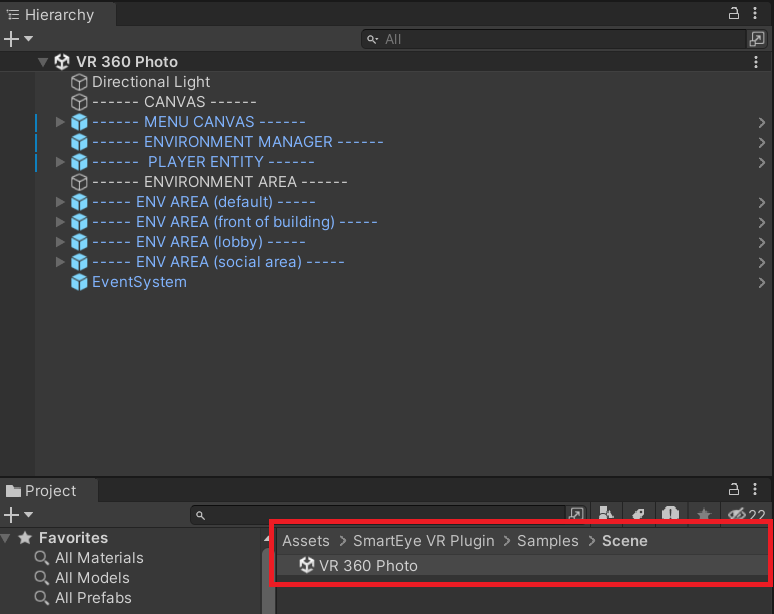
**6. Environment Area** : A component that houses area settings, such as interactive objects and 360-degree photo backgrounds.

**7. Sound Settings** : Settings for music and sound effects in the application

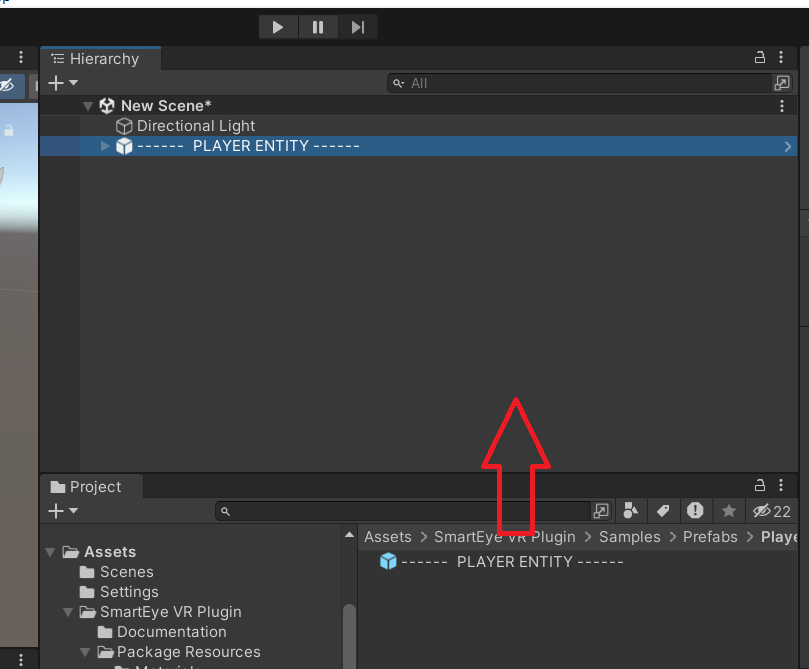
1. **Starting the Work**
   1. You can start creating the application in two ways: by using an existing scene as an example or by creating a new scene and inserting available functions in prefabs.

****

* 1. Here's how to start the project using a sample scene that's already available



* 1. If you're starting with a new project, you can use already available prefabs by incorporating them into the scene

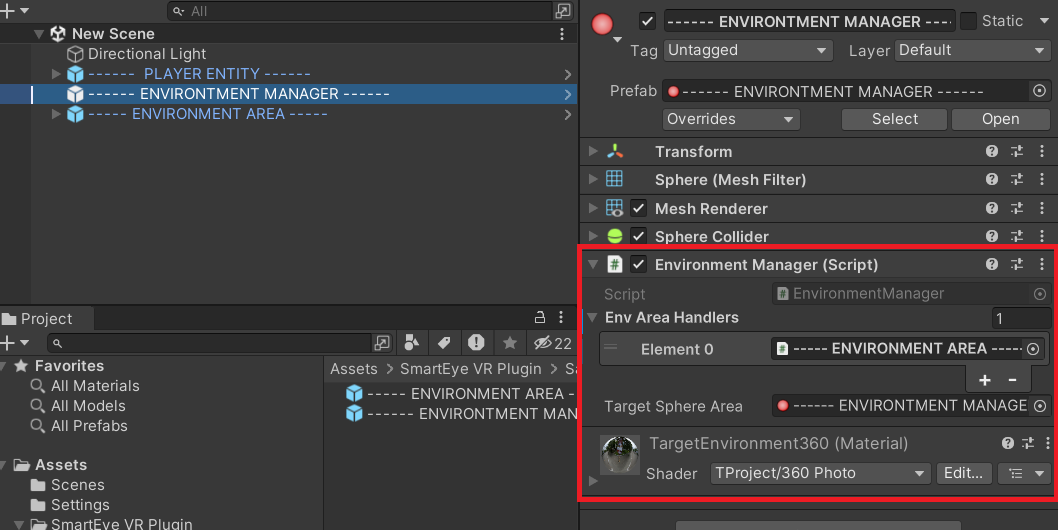


1. **Tools and Package**

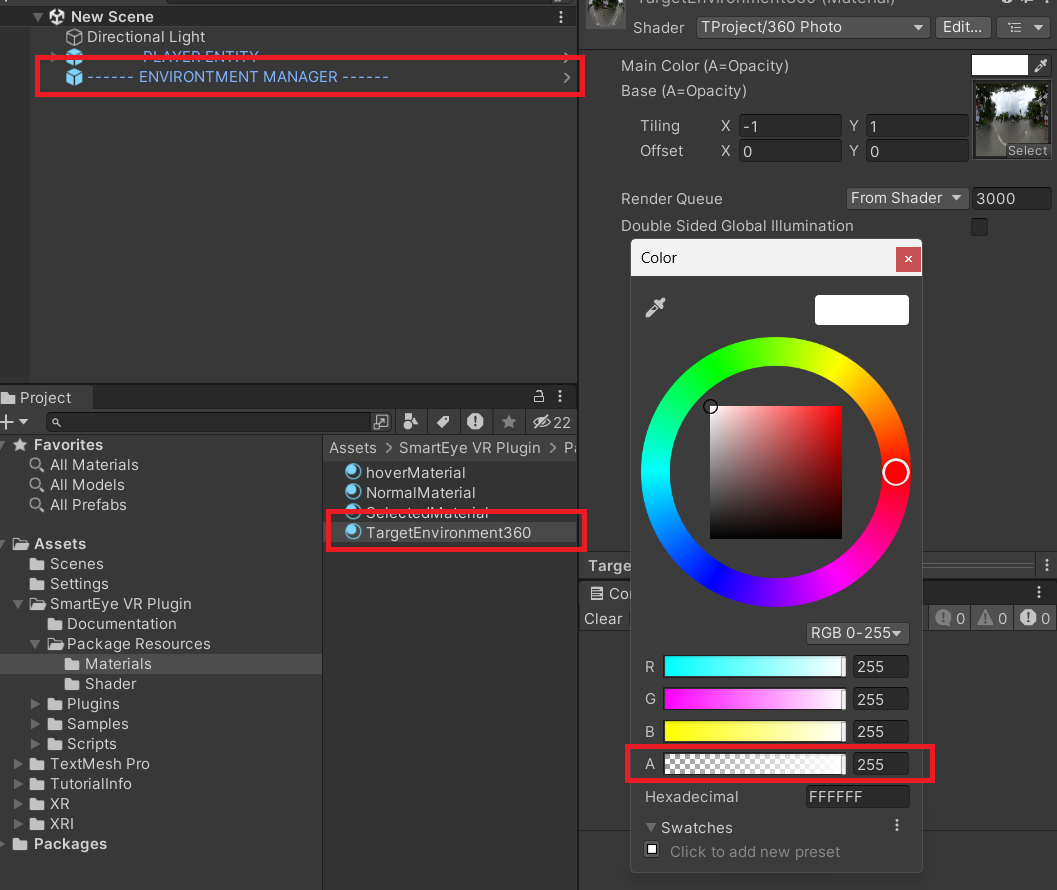
* **Player Entity**

Also commonly referred to as a character controller. By default, when using this component, players can move the headset and interact with objects using controllers.

* **Environment Manager**

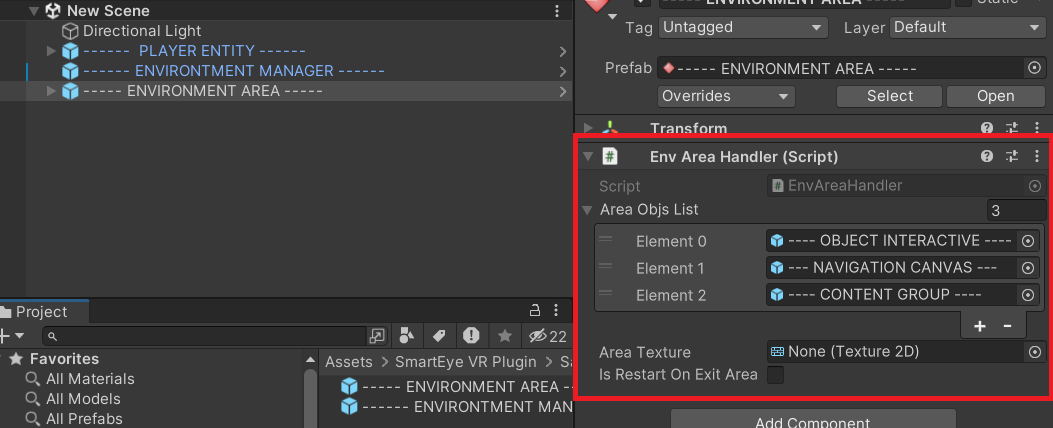
****

The environment manager stores multiple areas and the index numbers of "Env Area Handlers," which serve as identification numbers for each area. One of the uses of these indices is to switch between areas or perform teleportation.

****

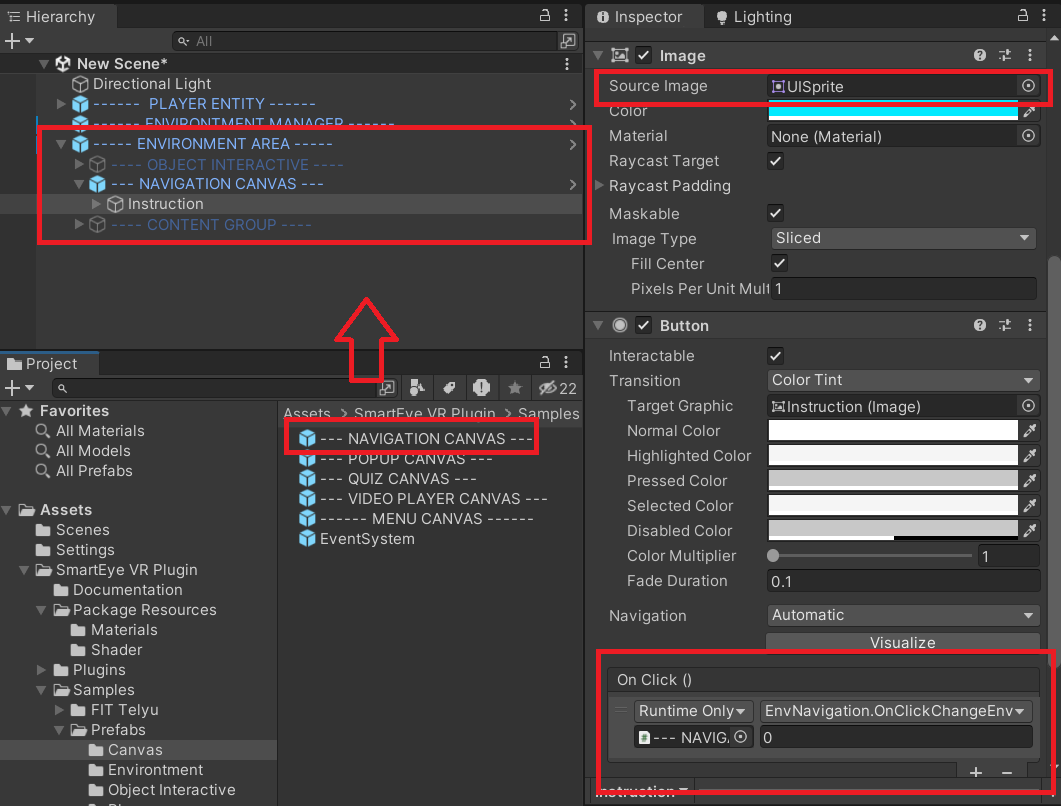
Fundamentally, the environment manager already includes materials that can be found in the package resources folder. You can adjust the alpha value to view 360-degree photo areas.

* **Environment Area**

****

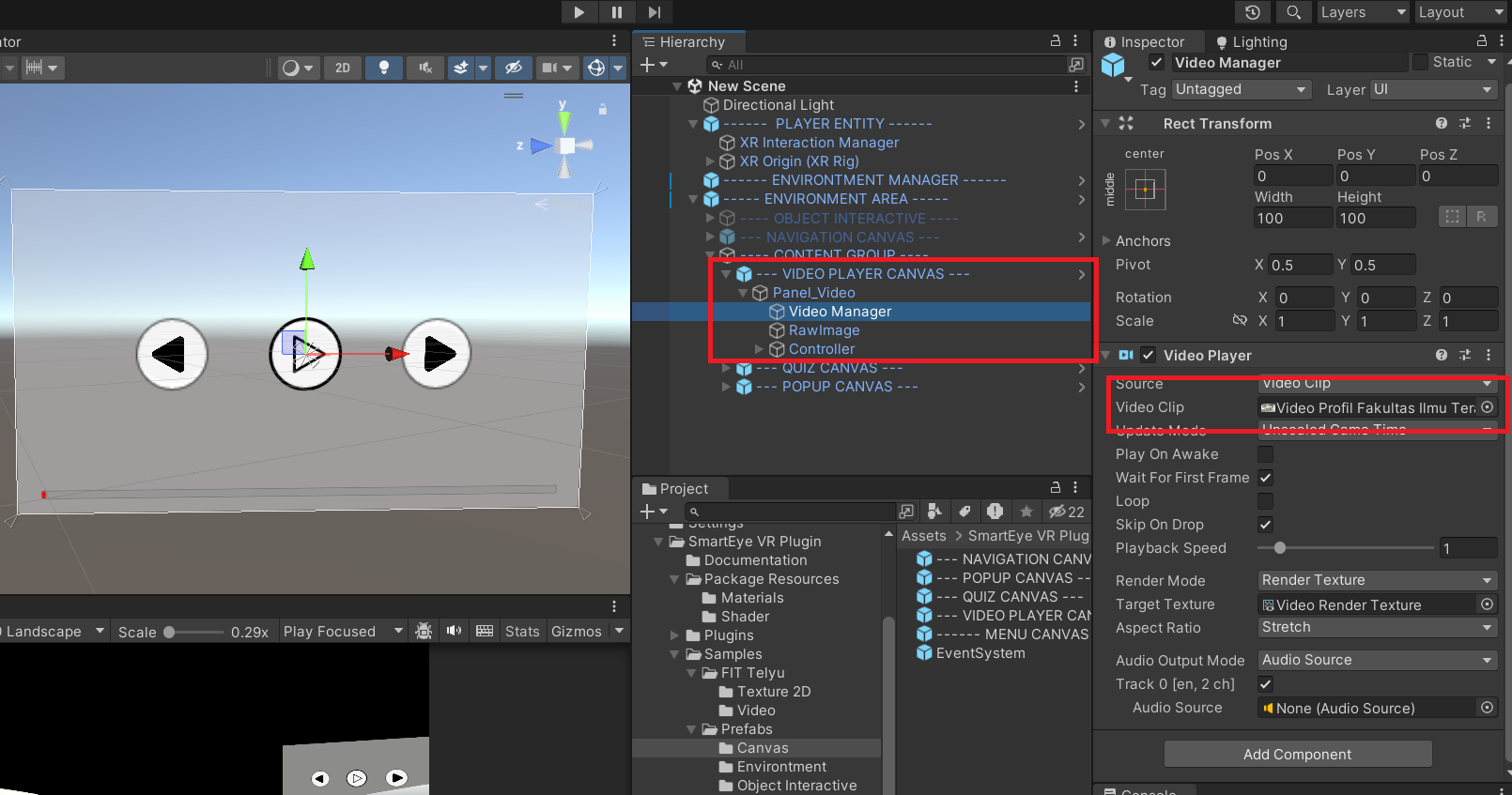
The environment area consists of lists of interactive objects within an area, followed by textures or 360-degree photo backgrounds. There is also validation regarding whether the area needs to be reset when transitioning to another area.

* **Navigation Canvas**

****

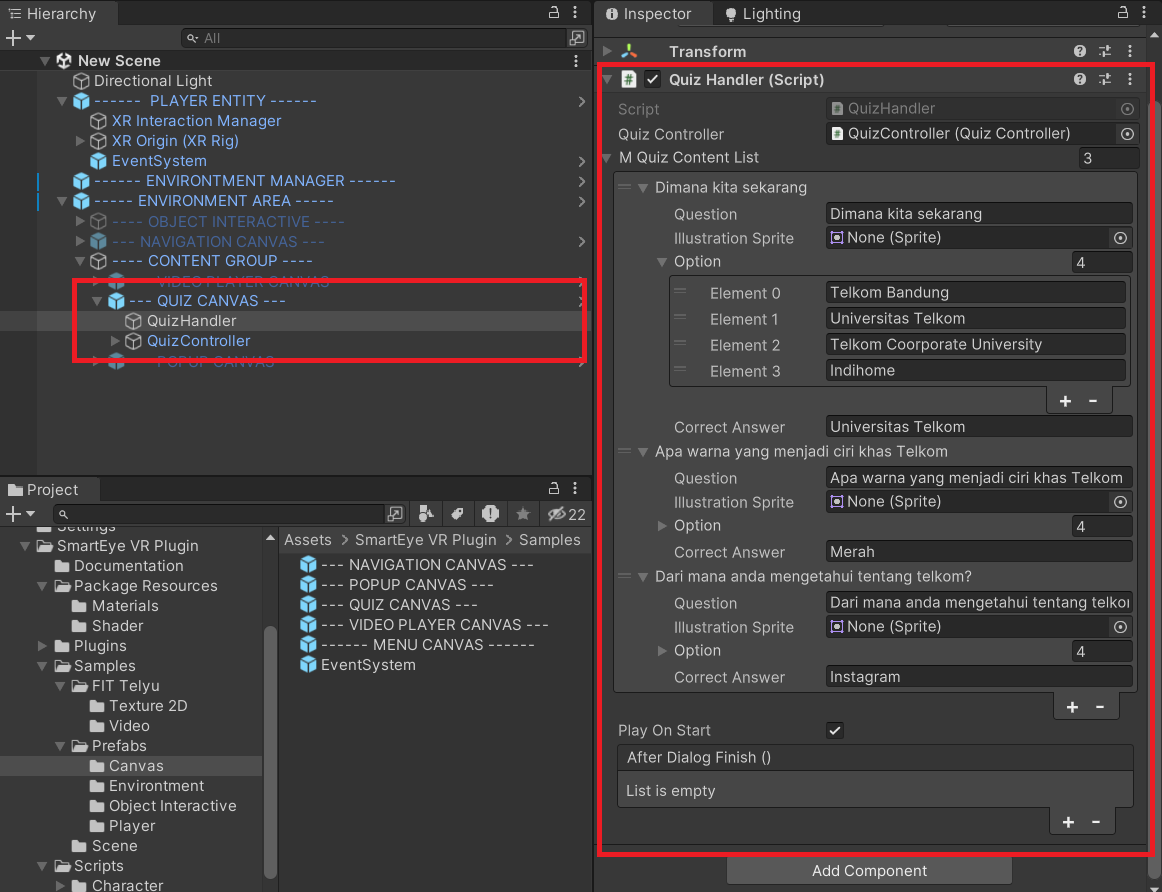
The navigation canvas is used to switch between areas, and you can place it within the environment area. To add images for button displays, you can replace the source image. You need to enter the index value of the target area, which is based on the area index available in the environment manager.

* **Video Player Canvas**

****

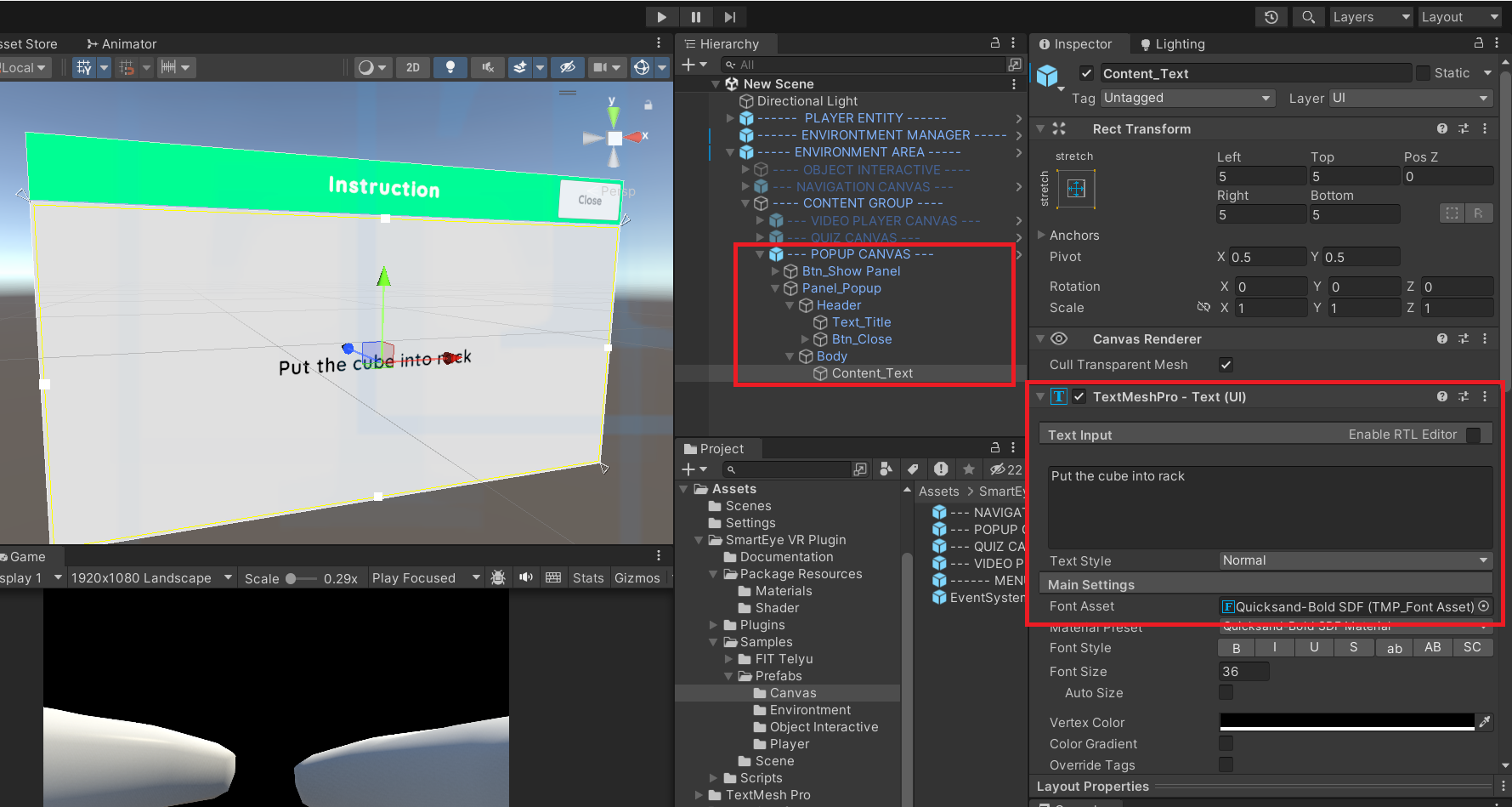
To display videos in the environment, you can use the video player canvas and insert videos in mp4 format into the video clip.

* **Quiz Canvas**

****

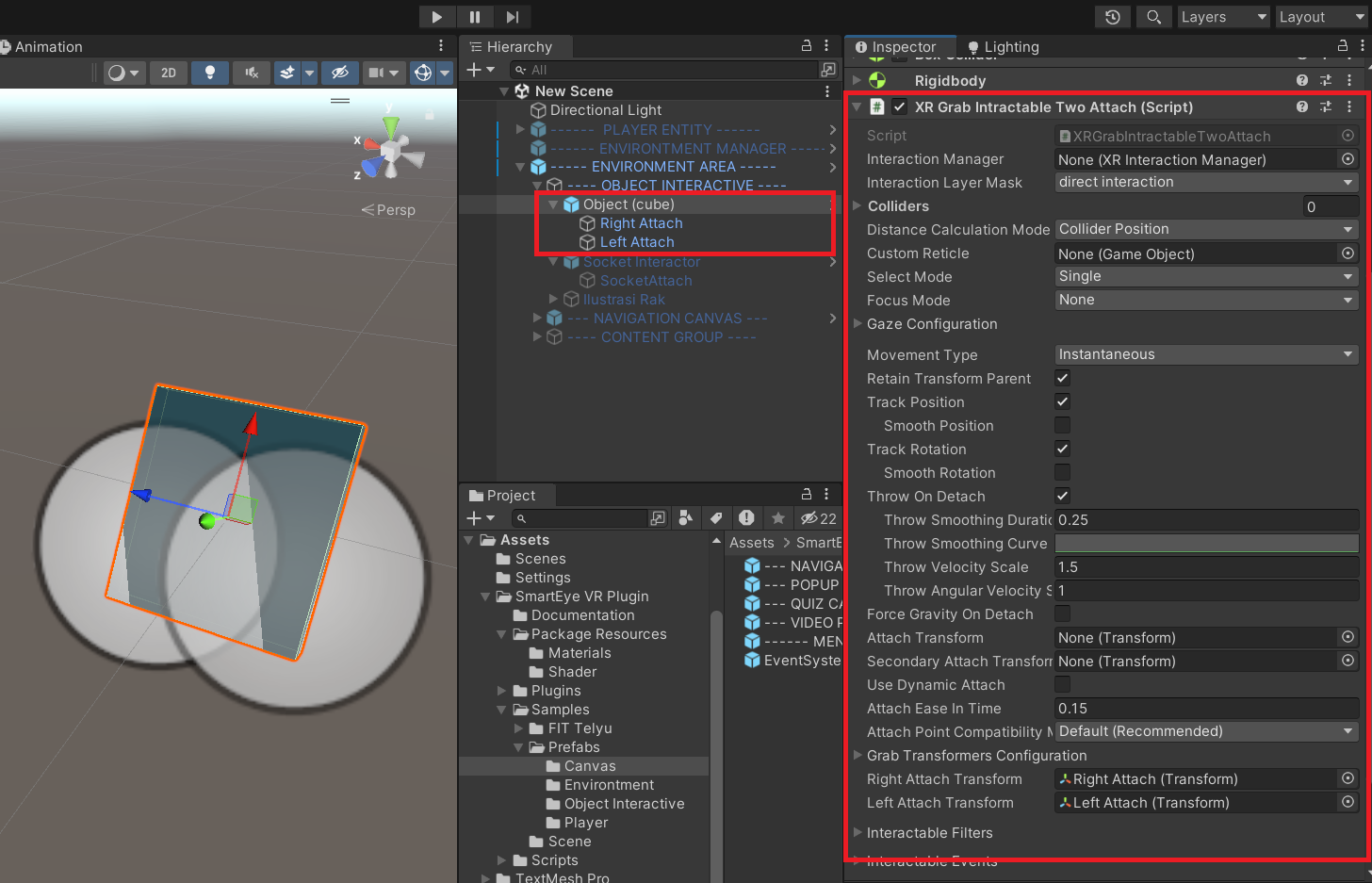
You can add questions and answers using the script component Quiz Handler. There are illustrative images available if you want to include pictures. The options consist of a maximum of 4 answers, with the correct answer specified in the valid answer field.

* **Pop Up Canvas**

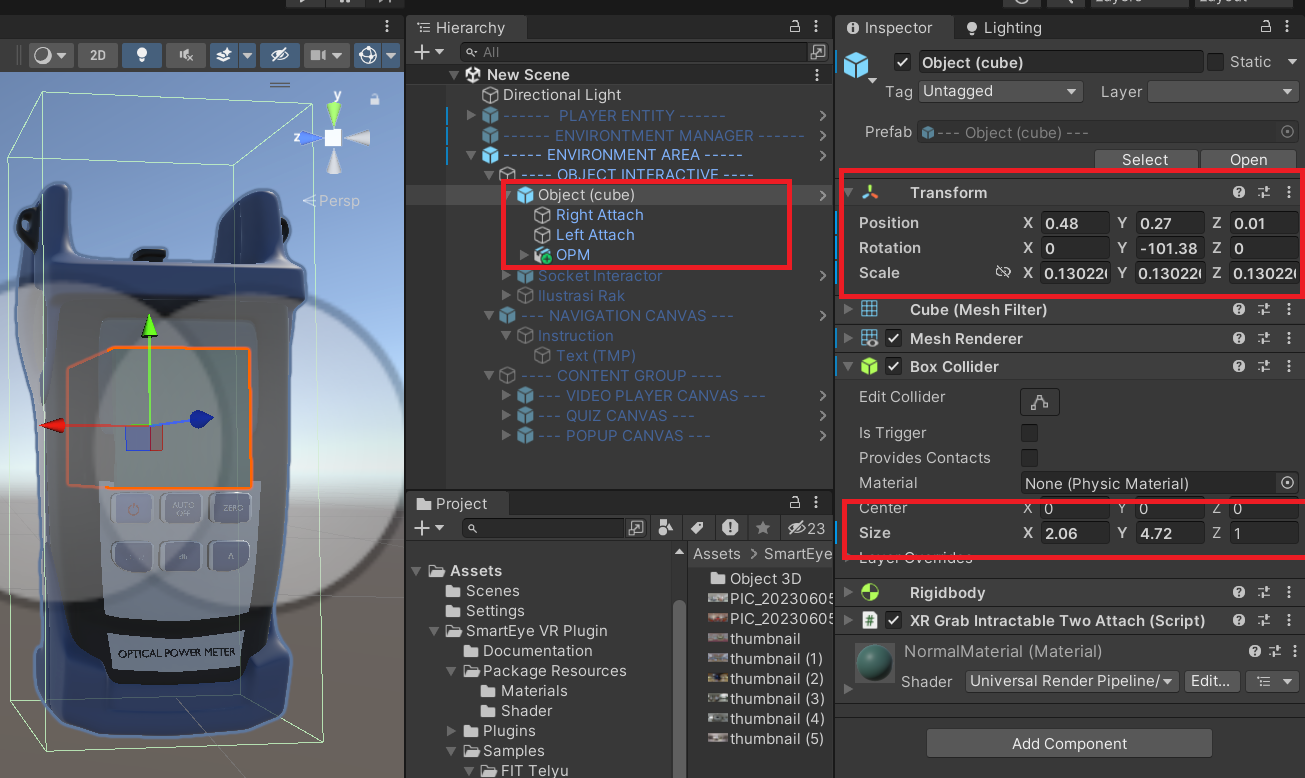
****

You can add pop-up text and modify content using the TextMeshPro component within the GameObject body.

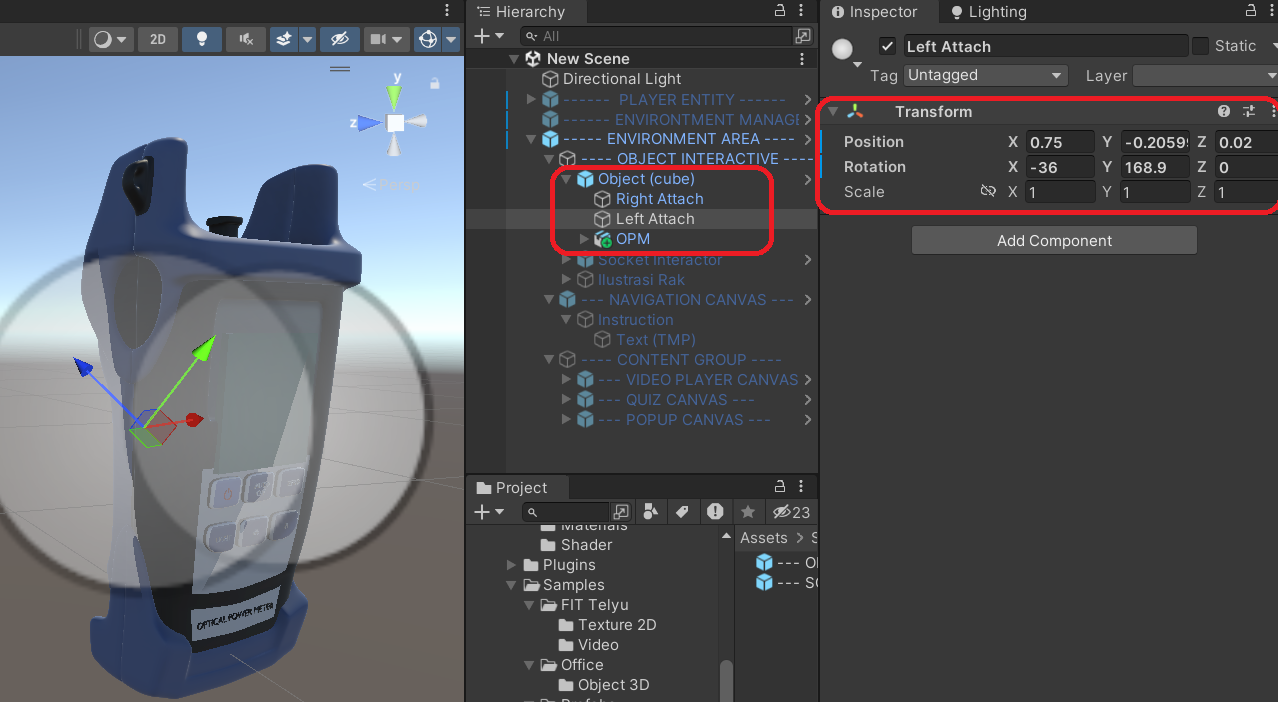
* **Object Interactive**

****

The Object Interactive (cube) is a standard asset to place interactive objects. There is an attachment used to adjust hand positions when holding things.

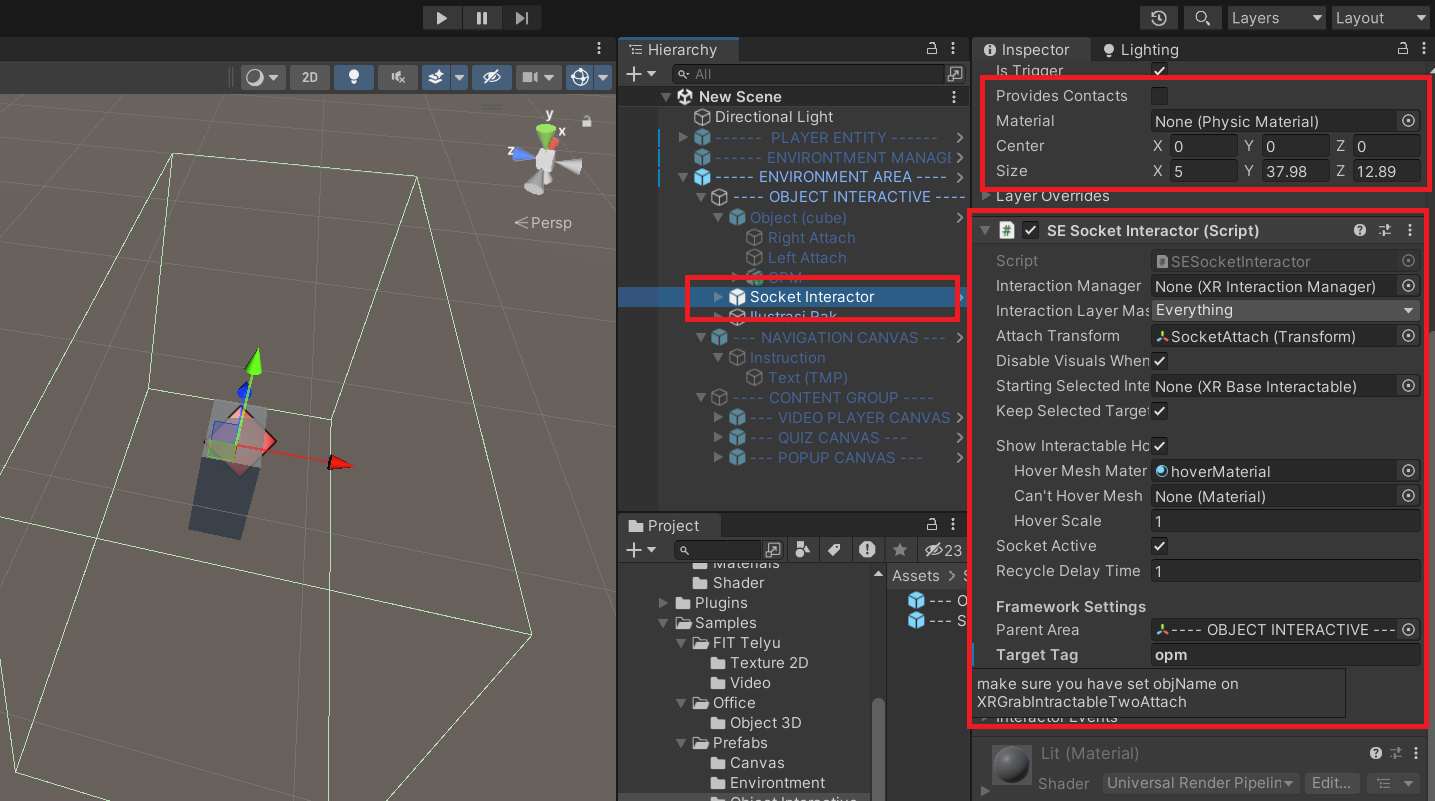


To add a new object, insert an FBX object as a child of the Object prefab, then adjust the object's size and position. Remember to set up the box collider to match the trigger area. To hide the cube's mesh, you can disable the mesh renderer.



You can adjust the transform on the object attachment if you want to change hand positions when the player grips an object.

* **Socket Interactor**

****

To create a socket or place for players to put objects, you can use the socket interactor. You can adjust the box collider to define the area that can detect objects. Set the transform position to change the object's status once the player has placed it.

1. **Tutorial**
2. **Use Universal Render Pipeline**

When creating a VR application using the Smarteye framework, choose a project supporting the Universal Render Pipeline

1. **Change the Platform to Android**

Open the project settings menu and select the option to change the platform to Android.

1. **Instal XR Plugin Management**

Next, open the project settings to install XR Plugin Management, which is required for the project.

1. **Select a Plugin Provider**

You can choose a plugin provider, such as Oculus or Open XR, but we recommend using Open XR.

1. **Add Interaction Profiles for Oculus Controllers**

Ensure you have added appropriate interaction profiles for Oculus controllers

1. **Instal XR Integration Toolkit**

Open the Package Manager window, then install the XR Integration Toolkit available in the Unity registry.

.

1. **Download and Import Smarteye Framework Package**

Download the package from the asset store. Then, you can install it via the package manager or do it manually from a local folder on your computer.