Session 8

Creating an Intro Scene in VR

Welcome to the session, Creating an Intro scene in VR

This session gives an introduction to creating an Intro scene in VR.

In this session, students will learn to:

* Explain UIHelpers
* Describe working with UI elements
* Explain how to create a .apk file for Oculus HMD
* Illustrate transferring of file to Oculus device

# 8.1 Introduction to UIHelpers

UIHelpers in virtual reality (VR) using Oculus Integration SDK provide a set of tools and components that assist in creating user interfaces (UI) for VR applications developed with Oculus platform. These UIHelpers make it easier to design and implement interactive menus, buttons, panels, and other UI elements within a VR environment.

Oculus Integration SDK is a software development kit provided by Oculus, designed to simplify the process of creating VR applications for Oculus VR devices, such as the Oculus Rift and Oculus Quest. It provides a wide range of features and functionalities, including input handling, rendering, tracking, and several UI components.

UIHelpers in the Oculus Integration SDK offer several benefits for VR developers, such as:

* **Interaction Design**: UIHelpers provide a set of tools specifically tailored for designing VR user interfaces. These tools enable developers to create intuitive and immersive interactions that are optimized for VR environments.
* **VR-specific Components**: The Oculus Integration SDK offers pre-built UI components, such as buttons, sliders, panels, tooltips, and more, that are designed to work seamlessly within VR. These components are optimized for performance and can be easily customized to fit the visual style and functionality of the VR application.
* **Input Handling**: UIHelpers assist in handling input from Oculus VR controllers, enabling developers to map controller inputs to UI actions. This allows users to interact with the VR UI using hand gestures, buttons, and triggers on the Oculus controllers.
* **Spatial UI**: UIHelpers facilitate the creation of UI elements that exist in 3D space, allowing developers to position menus, panels, and other UI elements within the virtual environment. This spatial UI enhances immersion and enables users to interact with UI elements by physically moving and manipulating them.
* **Event-driven Interactions**: UIHelpers utilize event systems to trigger actions and respond to user interactions. Developers can easily define event listeners and callbacks to handle user inputs, such as button clicks or menu selections, and update the UI accordingly.

# 8.2 Working with UI Elements

User interface in VR refers to the visual and interactive elements that enable users to navigate, interact, and receive feedback within a virtual environment. It serves as the bridge between the user and the virtual world, facilitating a seamless and intuitive user experience as shown in Figure 8.1.

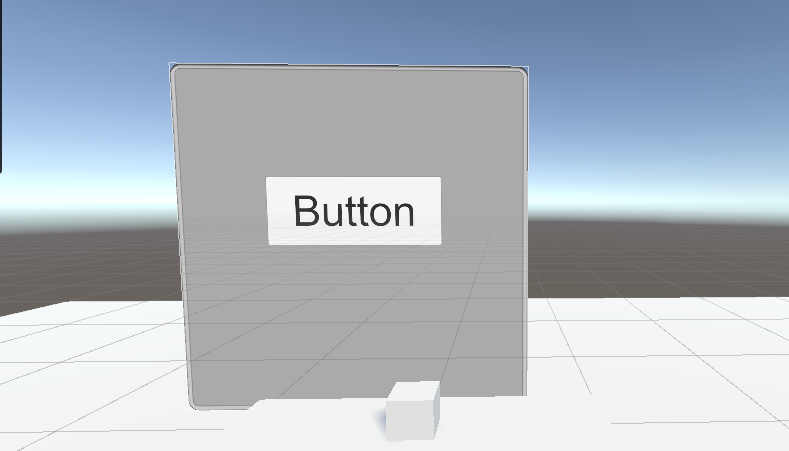
**Figure 8.1: Working with User Interface in Virtual Reality**

In VR, UI design takes on unique characteristics due to the immersive nature of the medium.

Following are a description of UI in VR:

* **Immersive Interaction**: VR UI aims to create a sense of presence and immersion by allowing users to interact with virtual objects and menus using their hands, gestures, or dedicated input devices such as VR controllers. This direct and natural interaction fosters a more intuitive and engaging user experience.
* **Spatial Design**: VR UI exists within the 3D space of the virtual environment. It can be positioned and anchored to physical objects or appear as floating elements in the user’s field of view. Spatial UI design leverages depth, scale, and perspective to enhance the sense of depth and presence, making UI elements feel as if they truly exist in the virtual world.
* **Visual Feedback**: VR UI provides visual feedback to users, such as highlighting interactive elements when they are gazed at or activated. Visual cues and animations help users understand the state of UI elements, such as button press animations, loading indicators, or progress bars. This feedback reinforces the user’s actions and aids in understanding the system’s response.
* **Minimized Distraction**: VR UI design aims to minimize distractions and maintain the user’s focus on the virtual environment. UI elements should be non-intrusive, translucent, or appear only when required. Balancing the visibility and usability of UI elements while preserving the immersion is crucial to ensuring a seamless user experience.
* **Contextual and Adaptive**: VR UI adapts to the context and requirements of the user. It dynamically adjusts its appearance, size, and functionality based on the user’s actions or the virtual environment’s requirements. For example, UI elements might change based on the user’s proximity or context-sensitive information may appear based on the user’s gaze or interaction.
* **Clear and Legible**: VR UI prioritizes clarity and legibility to ensure that information and instructions are easily readable in the virtual space. Fonts, colors, and visual hierarchy are carefully chosen to provide optimal readability, even in stereoscopic 3D environments.
* **Accessibility Considerations**: VR UI design takes into account accessibility considerations, ensuring that users with different abilities can effectively interact with the virtual interface. This may involve options for adjusting text size, contrast, and providing alternative input methods to accommodate user requirements.

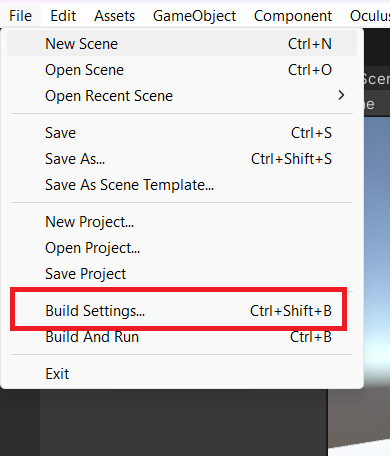
Overall, UI in VR combines the principles of traditional UI design with the unique affordances of Virtual Reality. It aims to create intuitive, immersive, and user-friendly interfaces that seamlessly integrate with the virtual environment, enhancing the overall VR experience. Figure 8.2 shows an UI created by adding a button.

**Figure 8.2: UI Created by Adding a Button**

# 8.3 Creating the Final Build in .apk Format

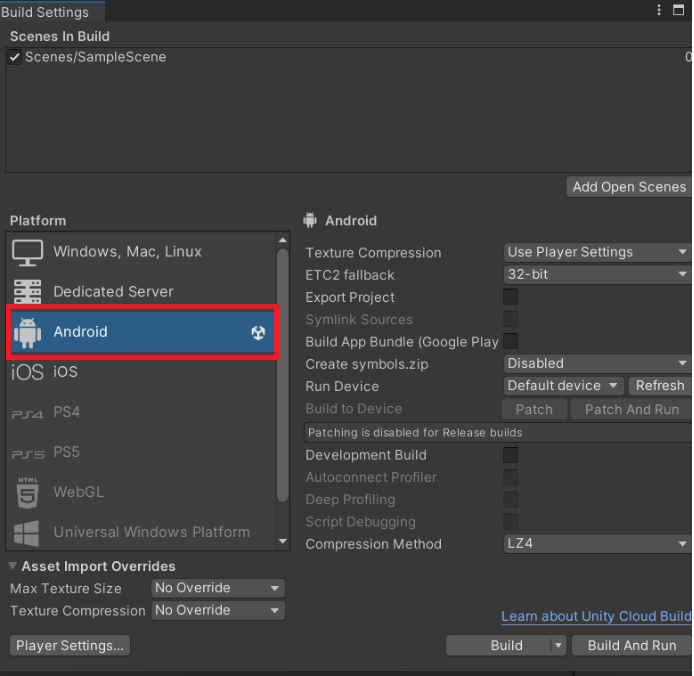
Following are the steps to create build settings:

**Step 1**:Go to **File** → **Build Settings** in Unity Editor as shown in Figure 8.3.



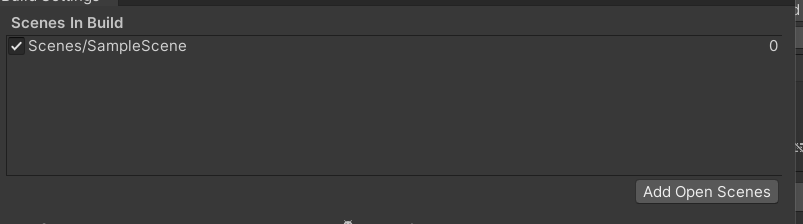
**Figure 8.3: Build Settings**

**Step 2**:In **Build Settings** window, select the target platform as **Android** as shown in Figure 8.4.



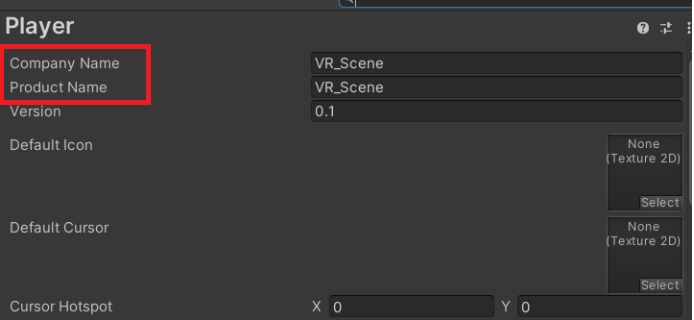
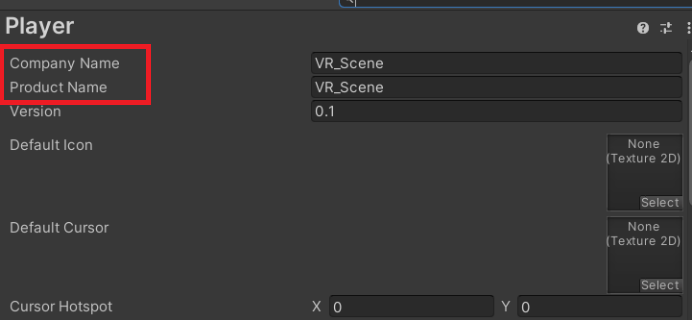
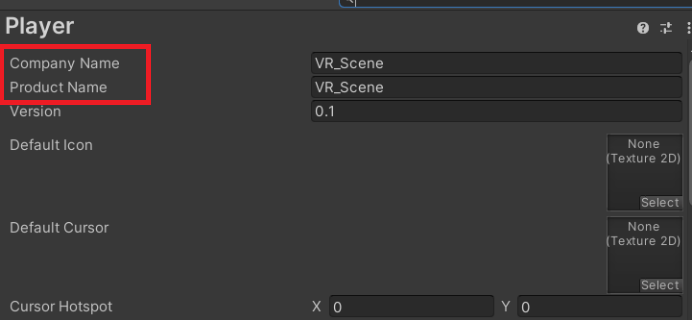
**Figure 8.4: Target Platform**

**Step 3:** Click the **Add Open Scenes** button to add Scenes in the application for build as shown in Figure 8.5.



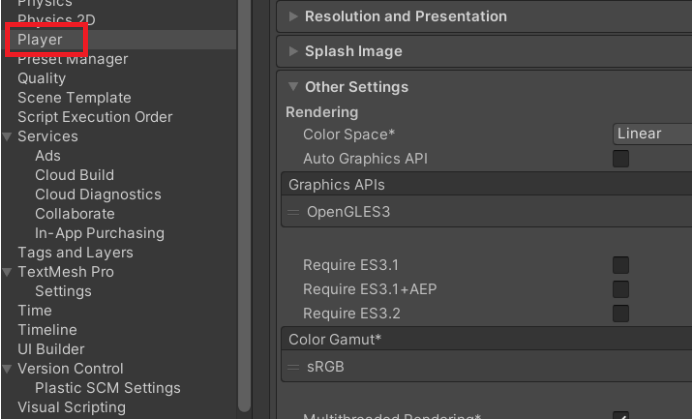
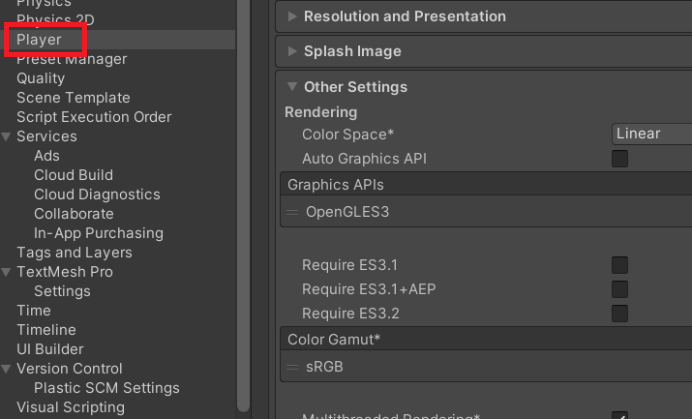
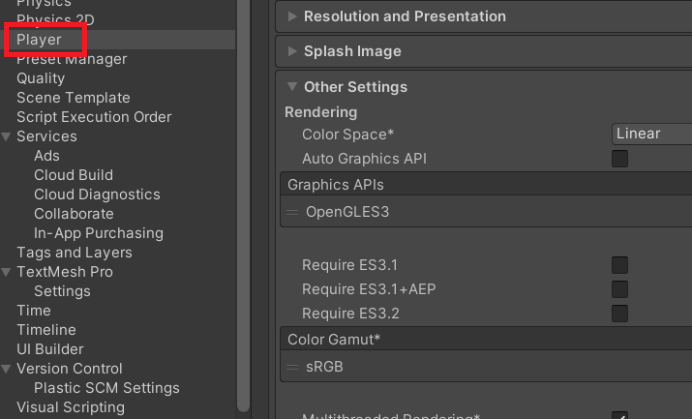
**Figure 8.5: Scenes Added**

**Step 4**:Go to **Player Settings**, and enter Company Name and Product Name. Select a unique name as shown in Figure 8.6.



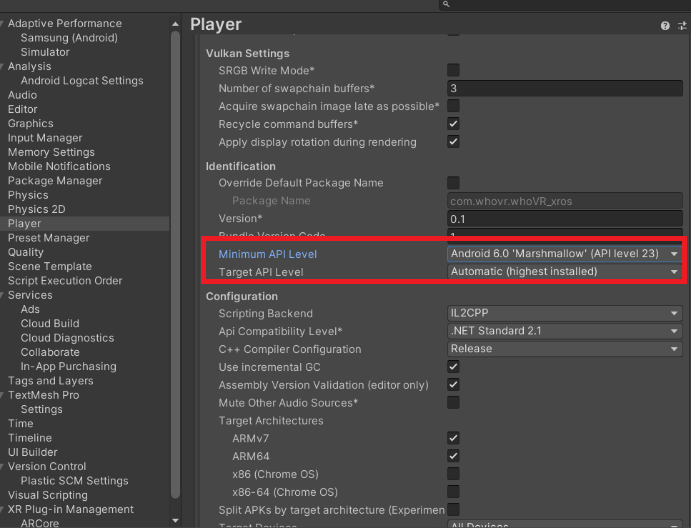
**Figure 8.6: Company Name and Product Name**

**Step 5**:Go to **Other Settings** in Player Settings as shown in Figure 8.7.

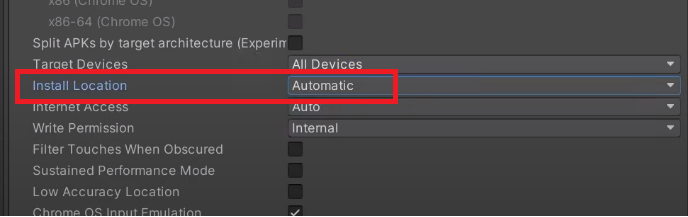
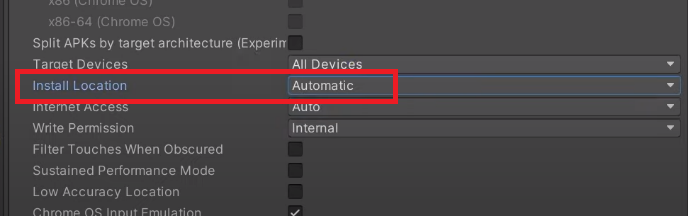
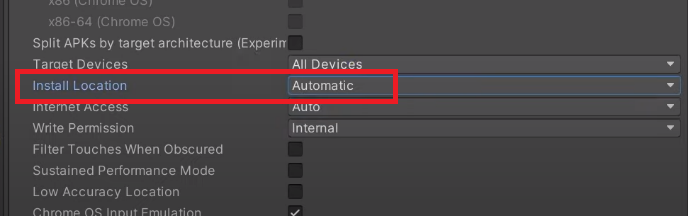


**Figure 8.7: Player Settings**

**Step 6**:Make sure **Minimum API Level** is Android 6.0 ‘Marshmallow’ (Android 23) and Target API Level is Automatic (highest installed) as shown in Figure 8.8.

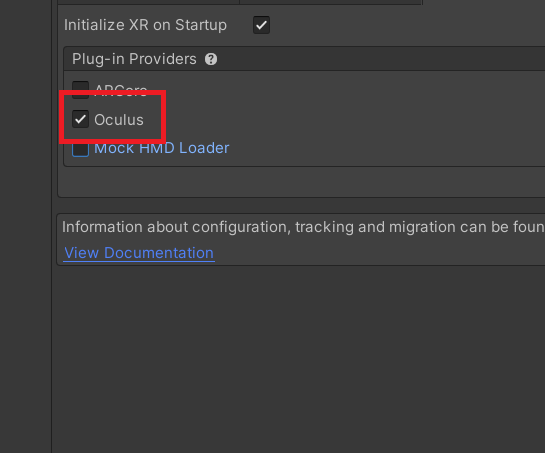
**Figure 8.8: Android APIs Selection**

**Step 7**: Turn **Install Location** to **Automatic** as shown in Figure 8.9.



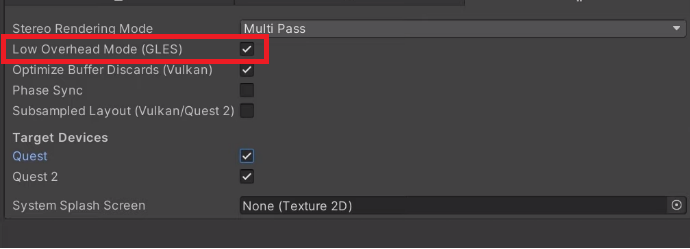
**Figure 8.9: Install Location to Automatic**

**Step 8**:Make sure **Oculus** checkbox is checked in **XR Plug-in Management** as shown in Figure 8.10.



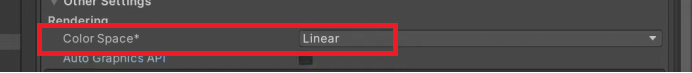
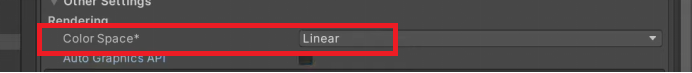
**Figure 8.10: Oculus Checkbox Checked**

**Step 9**:Make sure **Low Overhead Mode** is checked marked under XR Plug-in Management as shown in Figure 8.11.



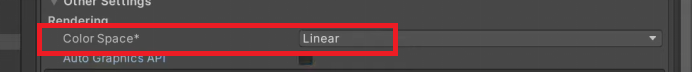
**Figure 8.11: Low Overhead Settings**

**Step 10**: Set the option as **Linear** for **Color Space** field, to make the build realistic as shown Figure 8.12.



**Figure 8.12: Set Color Space to Linear**

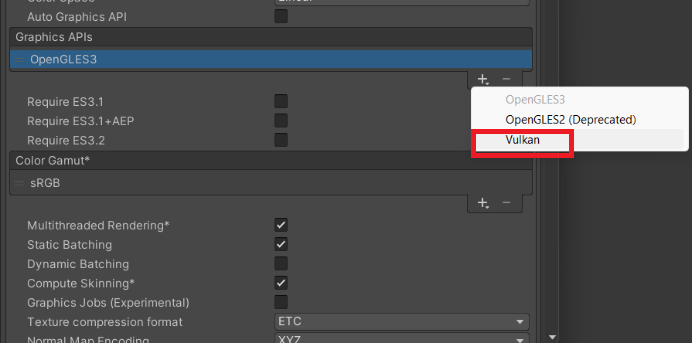
**Step 11**: Uncheck the **Auto Graphic API** checkbox as shown in Figure 8.13.



**Figure 8.13: Uncheck Auto Graphic API Checkbox**

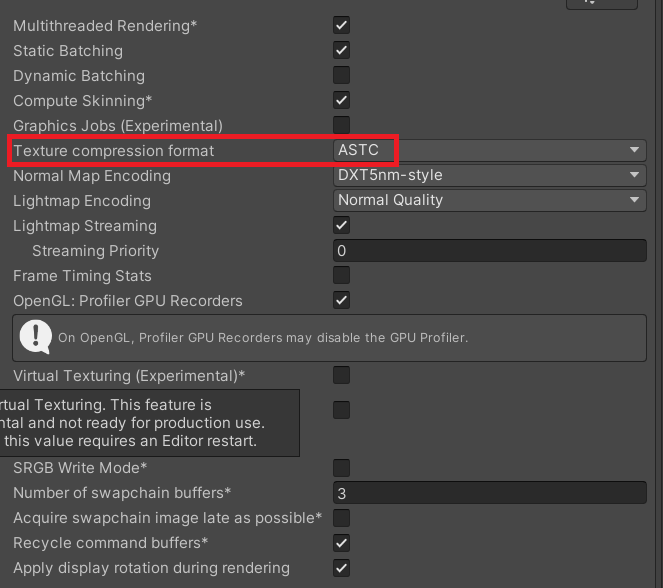
**Step 12**:Remove the Vulkan Settings by clicking the Graphic API’s then click on (-) sign to remove thye vulcan settings as shown in Figure 8.14.

The main reason we remove vulcan settings is that **it pushes a lot of responsibilities on to the application, including memory allocation, workload dependency management, and CPU-GPU synchronization**

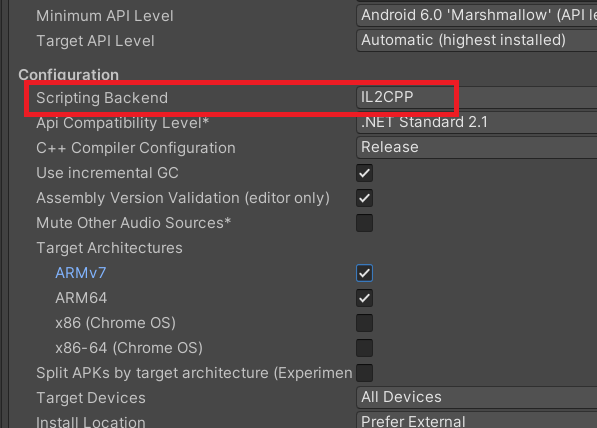


**Figure 8.14: Vulkan Settings**

**Step 13**: Set **Texture compression** **format** as **ASTC** as shown in Figure 8.15.

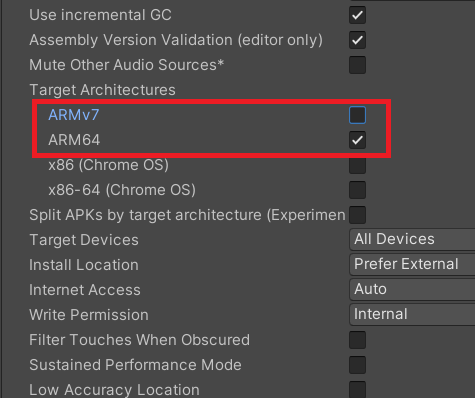
 **Figure 8.15: Texture Compression Set as ASTC**

**Step 14**:Go to **Configuration** section and change the option of **Scripting Backend** from **Mono** to **IP2CPP** as shown in Figure 8.16.



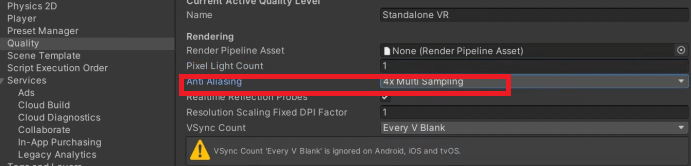
**Figure 8.16: Set Scripting Backend to IP2CPP**

**Step 15**: Uncheck ARMv7 and check ARM64 as shown in Figure 8.17.



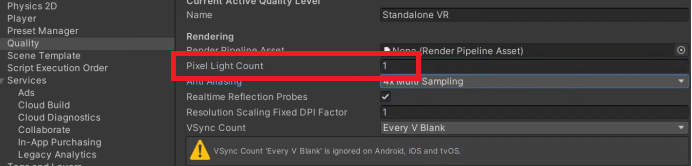
**Figure 8.17: Mark ARM64 Settings**

**Step 16**: Go to **Quality** and change **Anti Aliasing** to **4× Multi Sampling** as shown in Figure 8.18.



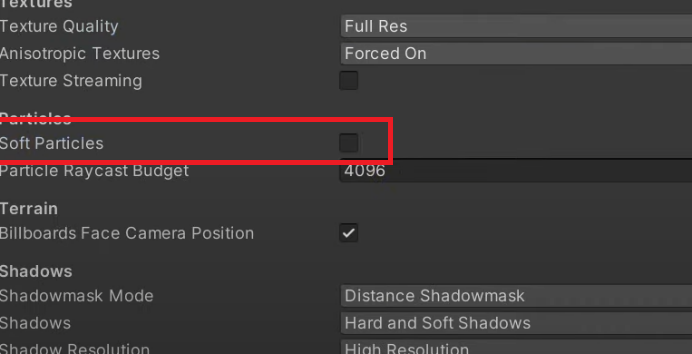
**Figure 8.18: Change Anti Aliasing to 4× Multi Sampling**

**Step 17**: **Change Pixel Light Count** to one (1) as shown in Figure 8.19.



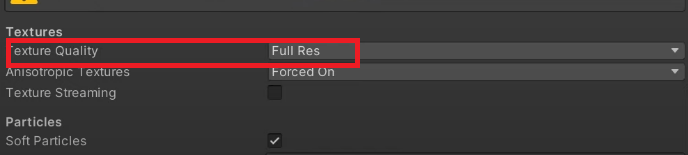
**Figure 8.19: Change Light Count to 1**

**Step 18**:Under Particles, uncheck **Soft Particles** checkbox as shown in Figure 8.20.



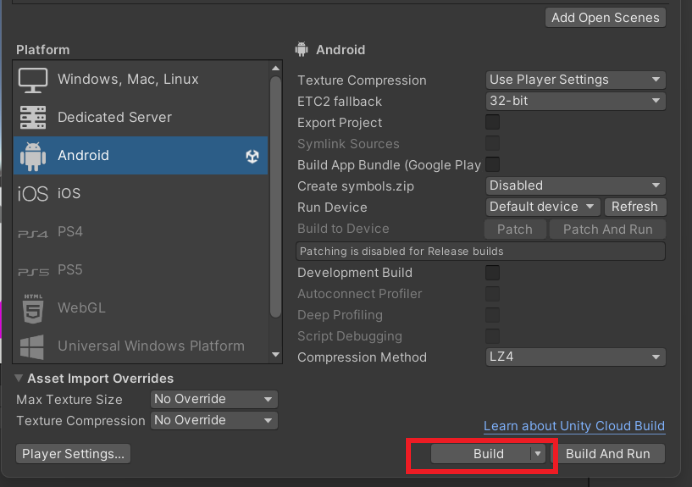
**Figure 8.20: Unmark Soft Particles**

**Step 19**:Make sure **Texture Quality** is set to **Full Res** as shown in Figure 8.21.



**Figure 8.21: Texture Quality to Full Res**

**Step 20**: Select Android and click Build as shown Figure 8.22.

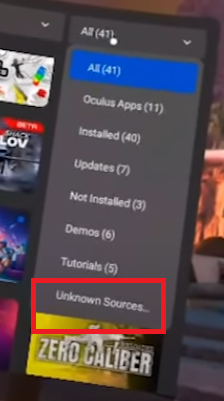


**Figure 8.22: Build apk**

# 8.4 Transferring .apk File to Oculus Device

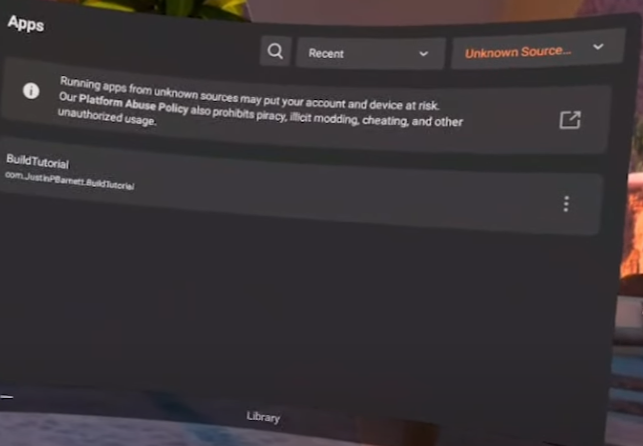
Following are the steps to transfer a file to Oculus device:

**Step 21**: Plug-in Oculus HMD. Go to Unknown Sources as shown in Figure 8.22.

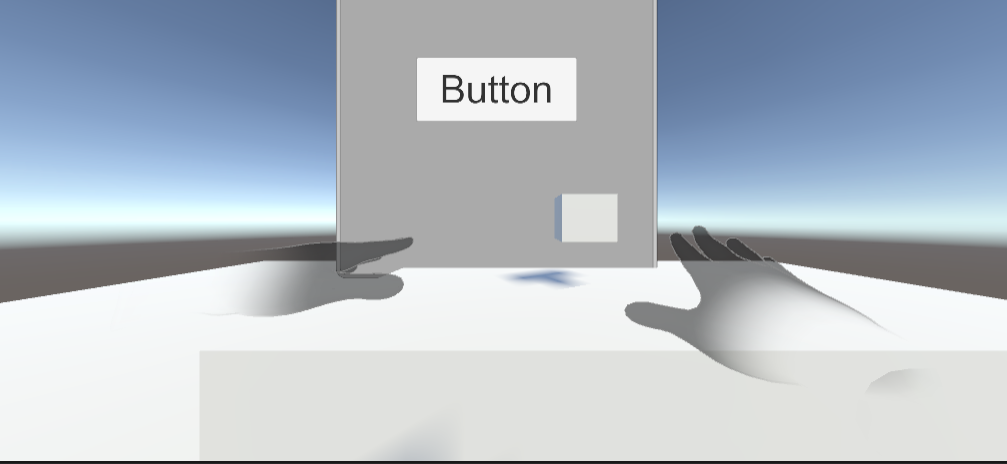


**Figure 8.22: Unknown Sources(This is Oculus HMD Screen)**

**Step 22**: Click the Application Build .apk build as shown in Figure 8.23.

**Figure 8.23: Install Build APK on Oculus HMD**

**Step 23**:Open and test the scene in Oculus HMD as shown in Figure 8.24.



**Figure 8.24: Final Build**

8.5 Summary

* UIHelpers assist in handling input from Oculus VR controllers, enabling developers to map controller inputs to UI actions.
* Spatial UI in VR, integrates User Interfaces into the 3D virtual environment, enabling intuitive interactions with UI elements as if they were physical objects, enhancing immersion and usability.
* To transfer the apk build to VR, you must turn ON the Unknown Settings option in Oculus HMD.
* Minimun API level for development should be 6.1 or Android 23 and Target API Level is Automatic (highest installed)

# 8.6 Check Your Progress



1. Which element in VR is used for creating intuitive interfaces in virtual reality with optimized components and performance?
2. OVRPlayerController
3. OVRCameraRig
4. UIHelper
5. UIDesign
6. What should be minimum API level of Build for Oculus VR application?
7. Marshmallow 6.1
8. Lollipop 5.1
9. 7.1 Nougat
10. 8.0 Orea
11. Which option is turned Linear to make Build look more realistic in VR Scene?
12. Custom Design
13. Color Space
14. Graphic API
15. Texture Quality
16. Which Scripting Backend is supported by Oculus VR?
17. Mono
18. IP2CPP
19. IPOP22
20. IPCPO
21. Which option is required to open .apk file in Oculus HMD
22. Unknown Source
23. Known Source
24. Download
25. None of these

## 8.6.1 Answers

|  |  |
| --- | --- |
| 1 | c |
| 2 | c |
| 3 | b |
| 4 | b |
| 5 | a |

# 8.7 Try It Yourself

* Create Interactive UI using UIHelper.
* Build a VR apk Build.