Getting Started with SRWorks Experience in Unity

Prerequisites

To work with this demo project, import all following packages into your project (Figure1):

(1) SteamVR
(2) Vive-SRWorks-*version* > Unity > Plugin
(3) Vive-SRWorks-*version* > Unity > Experience

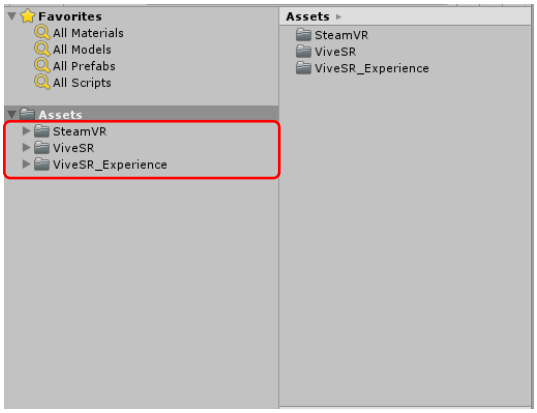

Figure 1. Project Window after Importing

Scenes

In **Experience_Unity > Assets > ViveSR_Experience > Scenes**, there are 9 scenes (Figure 2):

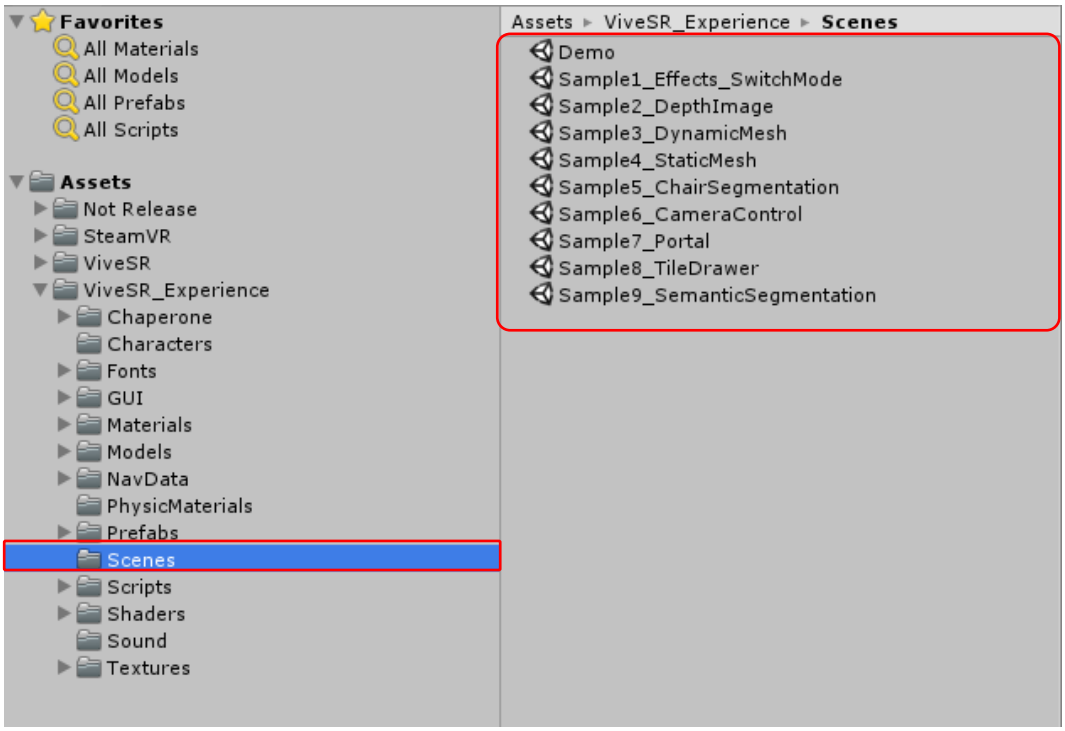

Figure 2. Nine Sample Scenes and Demo Scene

Demo.unity serves as an introduction scene that demonstrates most of **SRWorks**' features to new users in one go. If you spot some specific features you would like to use, you can check out the **sample scenes** (e.g. **Sample1_Effects_SwitchMode.unity**) to trace code. The sample scenes are breakdowns of all of the features shown in Demo.unity. Unlike Demo.unity, each sample scene targets only one or two features of SRWorks. Therefore, using the sample scenes will make it easier to find out how to call some specific features of SRWorks. This document will explain the structure of the sample scenes.

Sample1 – Effects Switch Mode

This sample demonstrates how to enable **visual effects** in see-through (Figure 3).

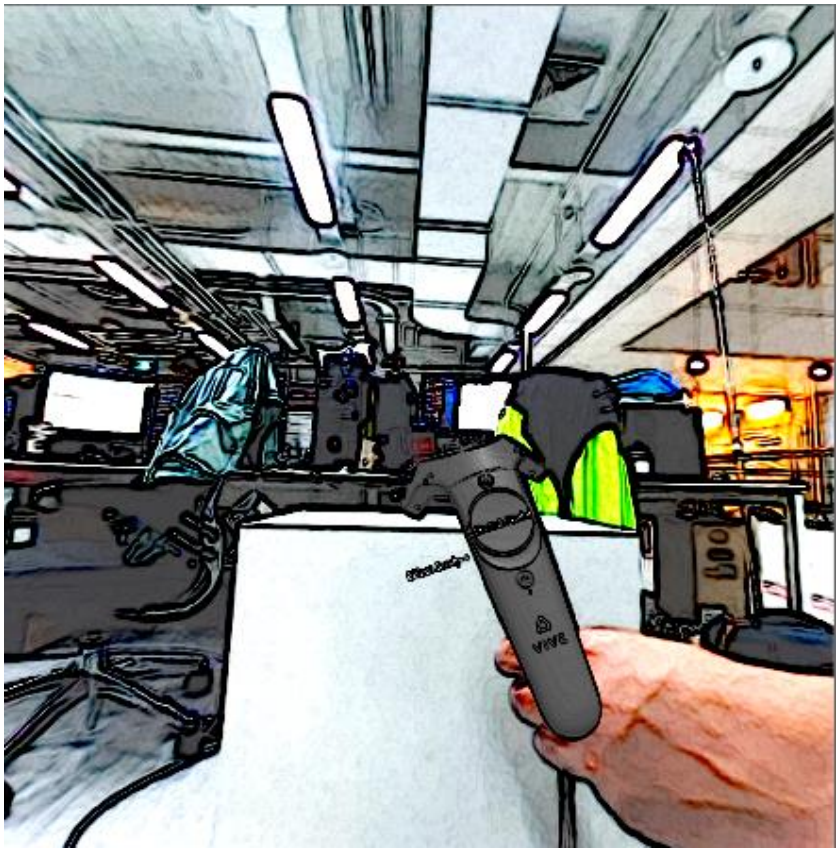

Figure 3. Sample1: Visual Effect

The structure of these sample scenes are simple and straightforward. When you open **Sample1_Effects_SwitchMode**, you can find a Gameobject that has the same name as the scene file (Figure 3).

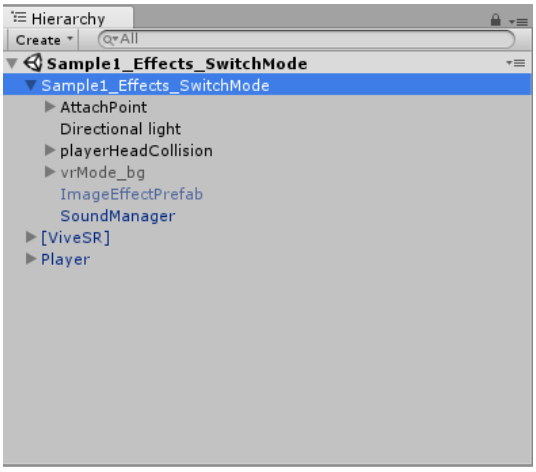

Figure 4. Sample1: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 4).

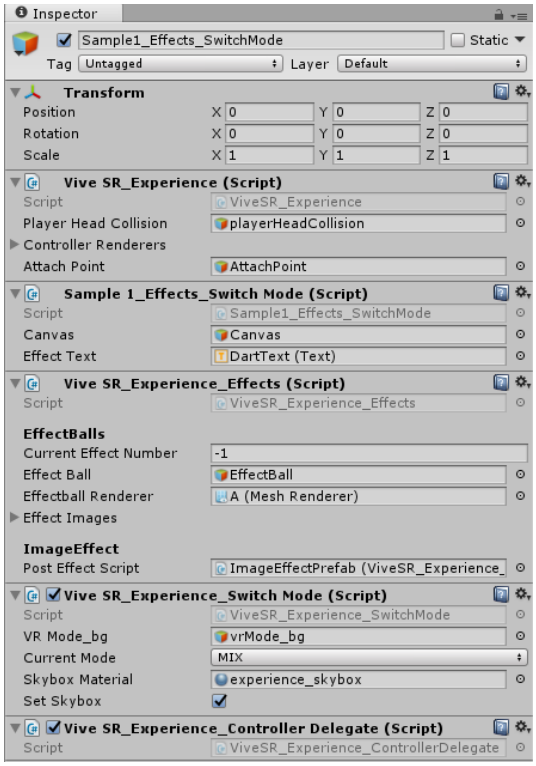

Figure 5. Sample1: Inspector Components

The following table explains what each Component does in the Sample1 (Table 1).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample1_Effects_SwitchMode	Determines how controller inputs work & handles UI display.
ViveSR_Experience_Effects	Utilizes SRWorks to provide functions for camera effects.
ViveSR_Experience_SwitchMode	Utilizes SRWorks to provide functions for mode switch.
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 1. Sample1 Components

Select **ImageEffectPrefab** component, you can tune each visual effect (Night Vision / Sharpen / Sketch / Thermal) strength in unity inspector (Figure 5).

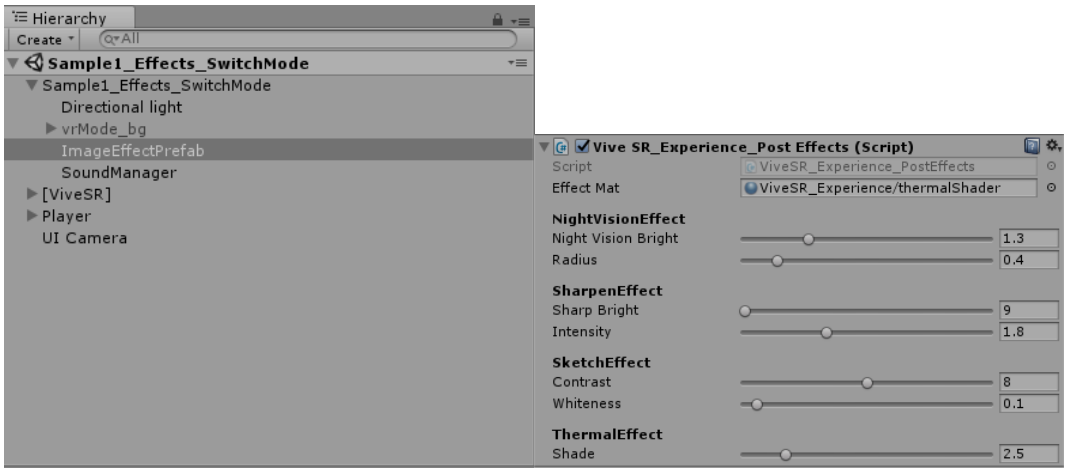

Figure 6. Sample1: Effect Strength Tuning

Sample2 – Depth Image

This sample demonstrates how to show and tune **Depth quality** (Figure 7).

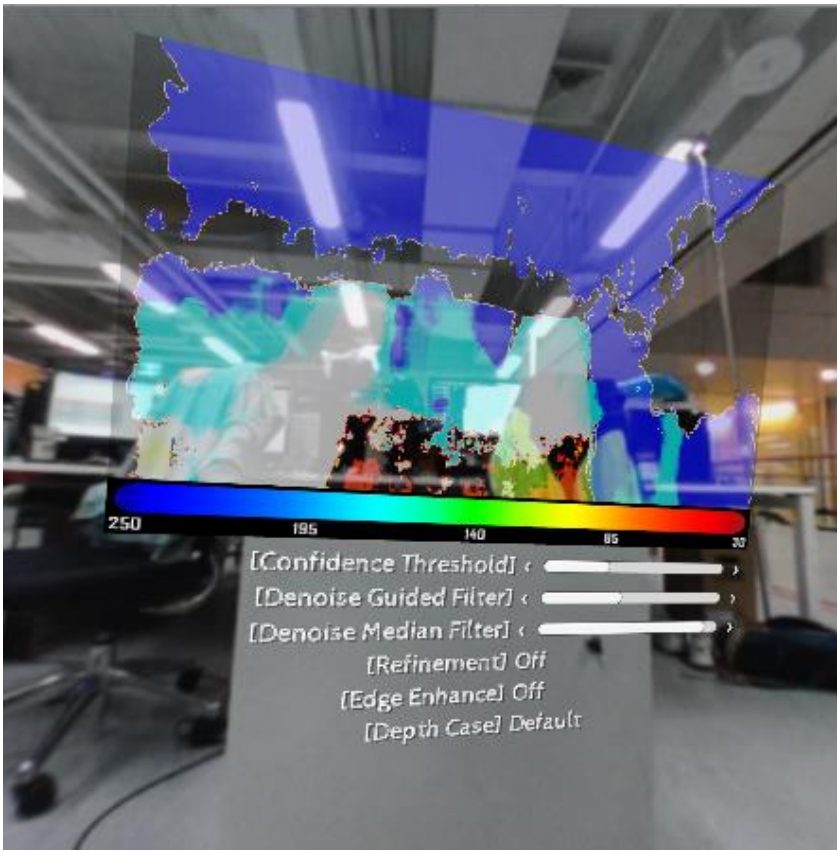

Figure 7. Sample2: Depth Image Control Window

The structure of these sample scenes are simple and straightforward. When you open **Sample2_DepthImage**, you can find a Gameobject that has the same name as the scene file (Figure 8).

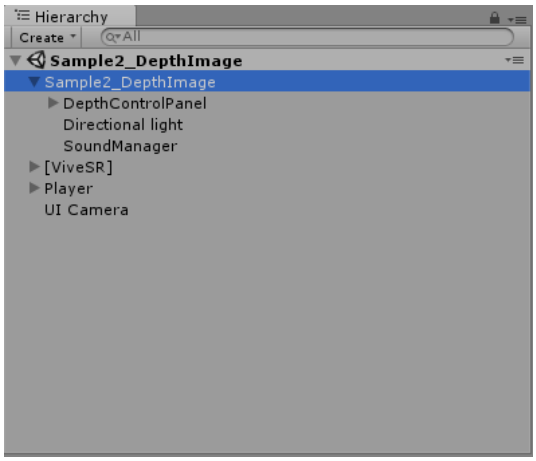

Figure 8. Sample2: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 9).

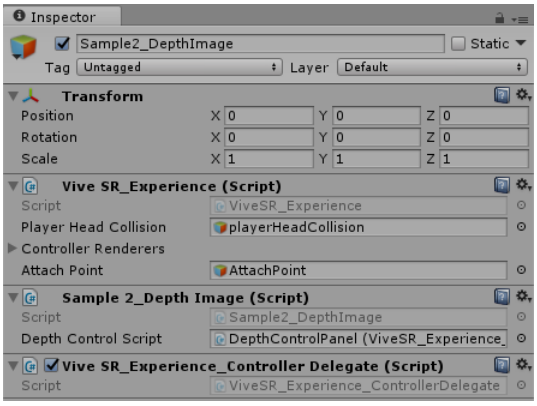

Figure 9. Sample2: Inspector Components

The following table explains what each Component does in the Sample2 (Table 2).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample2_Depth Image	Determines how controller inputs work & handles UI display.
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 2. Sample2 Components

Sample3 – Dynamic Mesh

This sample demonstrates how to use real-time **Dynamic mesh collider** (Figure 10).

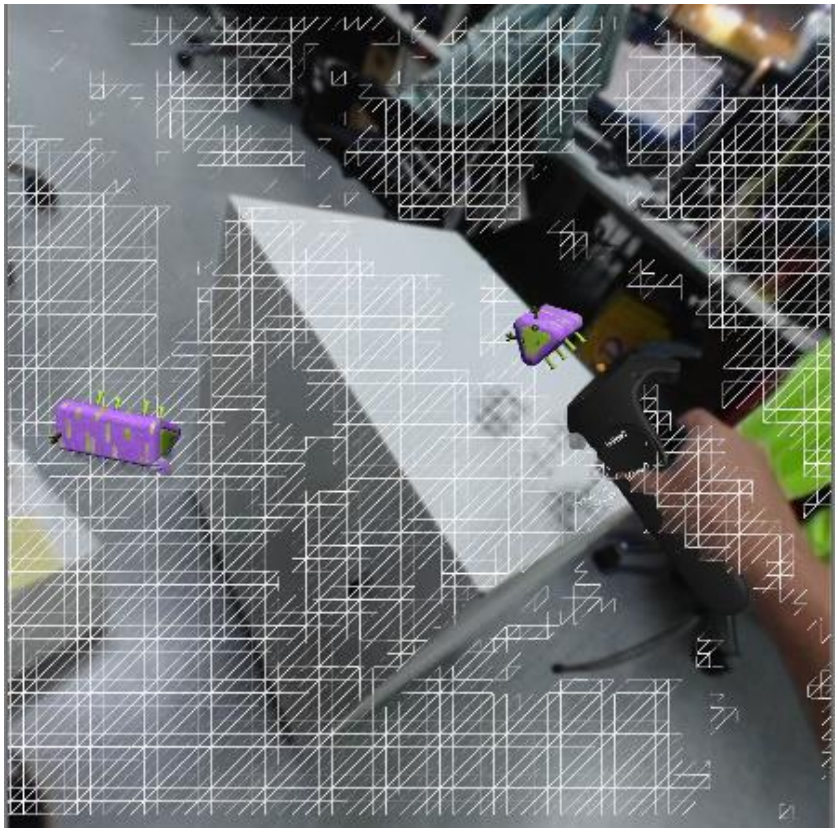

Figure 10. Sample3: Real-time interaction using dynamic mesh collider

The structure of these sample scenes are simple and straightforward. When you open **Sample3_DynamicMesh**, you can find a Gameobject that has the same name as the scene file (Figure 11).

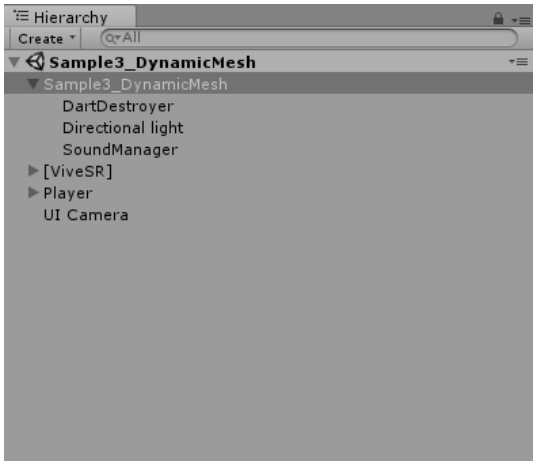

Figure 11. Sample3: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 12).

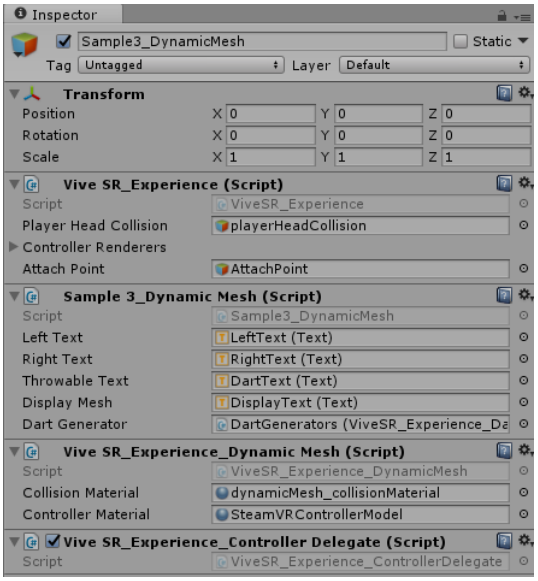

Figure 12. Sample3: Inspector Components

The following table explains what each Component does in the Sample3 (Table 3).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample3_Dynamic Mesh	Determines how controller inputs work & throw virtual objects.
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 3. Sample3 Components

Sample4 – Static Mesh

This sample demonstrates how to scan and load 3D mesh(Figure 13).

Figure 13. Sample4: Static mesh scanning

Please follow the instructions below (Figure 14):

1. <Scan>: 3D scanning the environment.
2. <Save>: Save 3D environment mesh.
3. <Stop>: Stop to scan.
4. <Load>: Load 3D mesh and mesh collider.
5. After loading mesh, you can "Trigger Throw Item" to throwing virtual objects.
6. Press "View Collider" → Show all mesh collider.

Figure 14. Sample4: Controller interface

The structure of these sample scenes are simple and straightforward. When you open Sample4_StaticMesh, you can find a Gameobject that has the same name as the scene file (Figure 15).

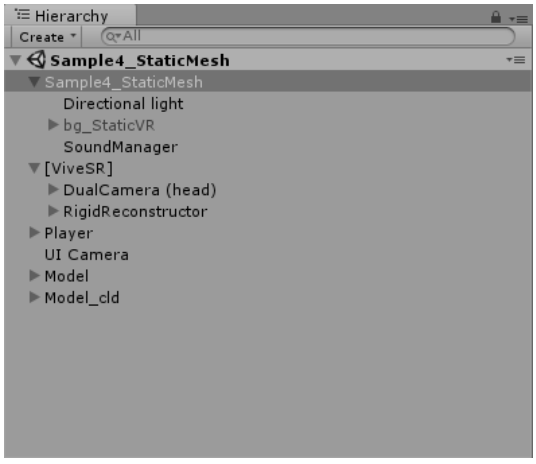

Figure 15. Sample4: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 16).

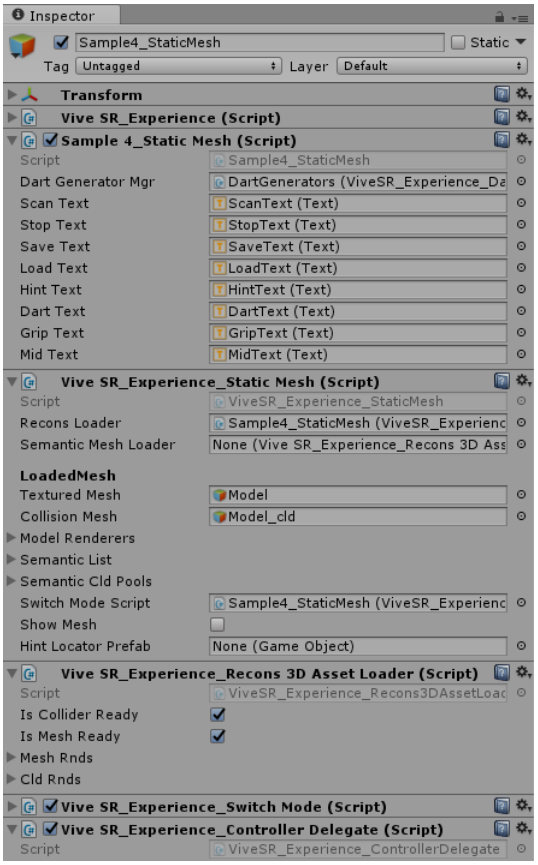

Figure 16. Sample4: Inspector Components

The following table explains what each Component does in the Sample4 (Table 4).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample4_Static Mesh	Determines how controller inputs work & throw virtual objects.
ViveSR_Experience_Static_Maesh	Render 3D scene mesh
ViveSR_Experience_Recons 3D Asset Loader	Runtime mesh loader controller
ViveSR_Experience_Switch Mode	Switch MR or VR scene mode
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 4. Sample4 Components

Sample5 – Chair Segmentation

This sample demonstrates how to let fairy to find a seat in the real world (Figure 17):

Demo Video: https://youtu.be/eMB_nApqb6w

Please follow the instructions below (Figure 17-19):

1. <Scan>: 3D scanning the environment.
2. <Save>: Save 3D environment mesh and chair information (seat position and seat forward direction).
3. <Test>: pre-test to find the chairs.
4. <Play>: Load chair information and 3D scene mesh. Start to play the fairy finding seat.

Figure 17. Sample5: Static mesh scanning

Figure 18. Sample5: Press “Test” to generate Chair Hint

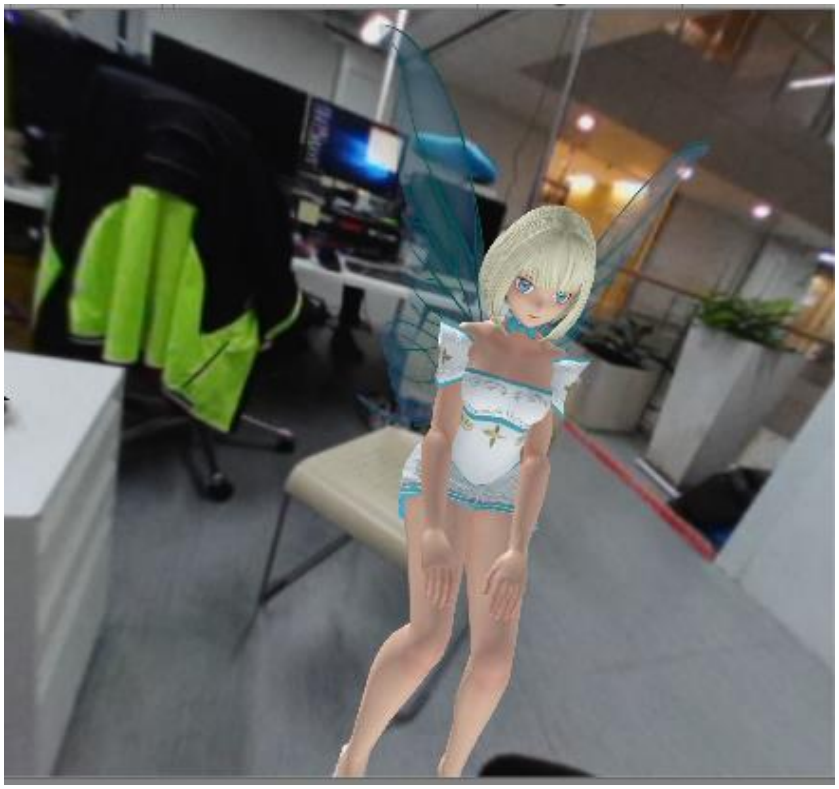

Figure 19. Sample5: Press “Test” to generate Chair Hint

The structure of these sample scenes are simple and straightforward. When you open **Sample5_ChairSegmentation**, you can find a Gameobject that has the same name as the scene file (Figure 15).

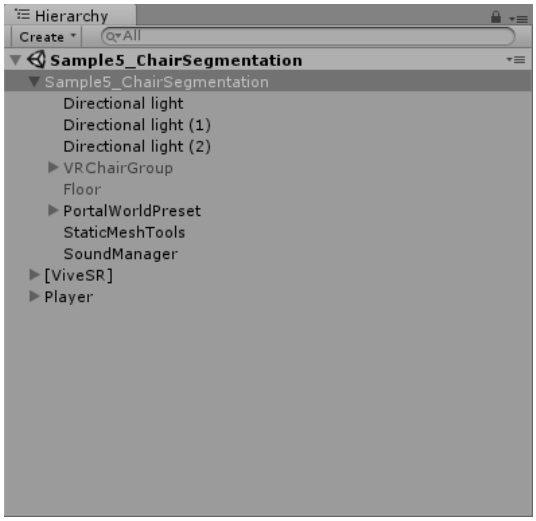

Figure 20. Sample5: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 21).

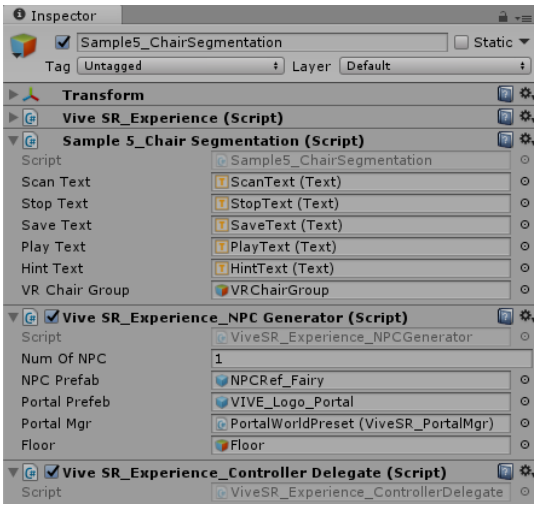

Figure 21. Sample5: Inspector Components

The following table explains what each Component does in the Sample5 (Table 5).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample5_ Chair Segmentation	Determines how controller inputs work & finding chair
ViveSR_Experience_NPC Generator	Generate NPC (Fairy) and portal navigation.
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 5. Sample5 Components

Sample6 – Camera Control

This sample demonstrates how to show and tune **Camera Control** (Figure 22).

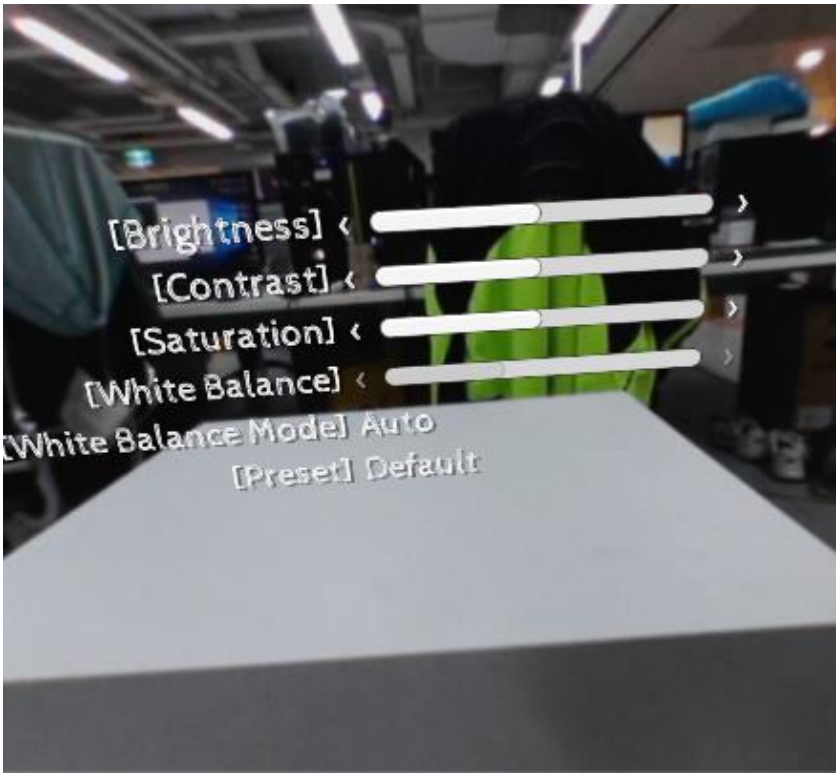

Figure 22. Sample6: Camera Control Window

The structure of these sample scenes are simple and straightforward. When you open **Sample6_CameraControl**, you can find a Gameobject that has the same name as the scene file (Figure 23).

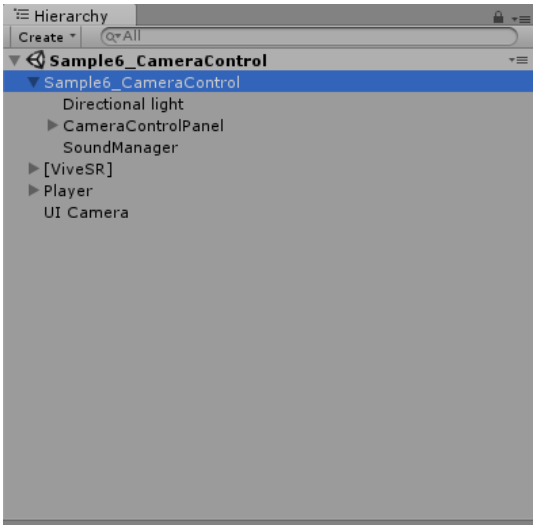

Figure 23. Sample6: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 24).

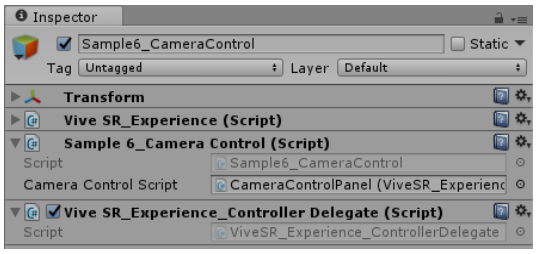

Figure 24. Sample6: Inspector Components

The following table explains what each Component does in the Sample6 (Table 6).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample6_Camera Control	Determines how controller inputs work & handles UI display.
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 6. Sample6 Components

Sample7 – Portal

This sample demonstrates how to use **Portal** (Figure 25).

Please see more detail in **"SRWorks Unity Portal Guideline.docx"**.

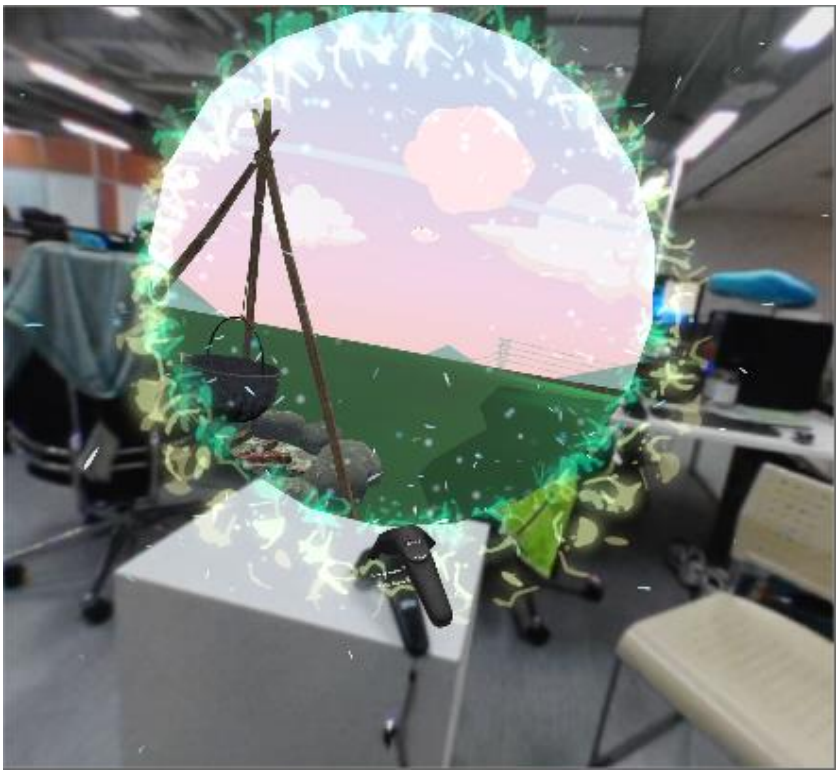

Figure 25. Sample7: Generate Portal

The structure of these sample scenes are simple and straightforward. When you open **Sample7_ Portal**, you can find a Gameobject that has the same name as the scene file (Figure 26).

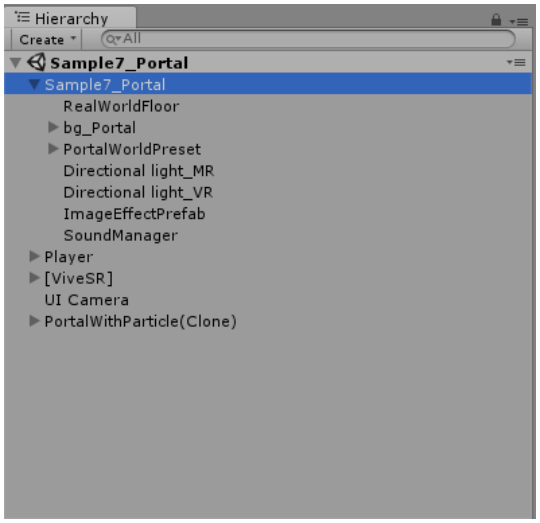

Figure 26. Sample7: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 27).

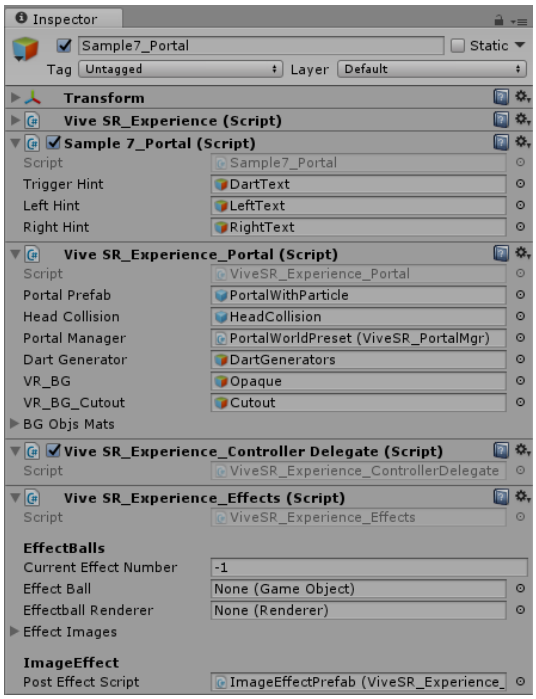

Figure 27. Sample7: Inspector Components

The following table explains what each Component does in the Sample7 (Table 7).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample7_Portal	Determines how controller inputs work & Portal control
ViveSR_Experience_ Portal	Generate Portal and MR interaction
ViveSR_Experience_Effects	Utilizes SRWorks to provide functions for camera effects.
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 7. Sample7 Components

Sample8 – Tile Drawer

This sample demonstrates how to lay tiles on a specified plane and the effect of Depth Occlusion. (Figure 28-29)

Please follow the instructions below:

1. Scan and load the environment.
2. Push '**<**' and '**>**' to adjust the orientation of the tile.
3. Hold trigger to specify a plane. Release trigger to lay tiles.
4. Push '**Occlude ON/OFF**' to enable/disable Depth Occlusion.

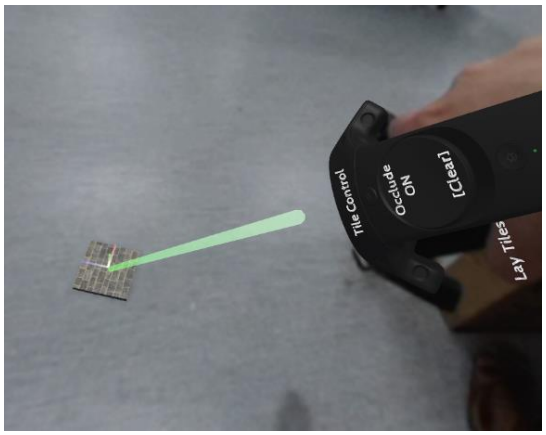

Figure 28. Screenshot of specifying a plane

Figure 29. Comparison between Depth Occlusion disabled and enabled.

The structure of these sample scenes are simple and straightforward. When you open **Sample8_ TileDrawer**, you can find a Gameobject that has the same name as the scene file (Figure 30).

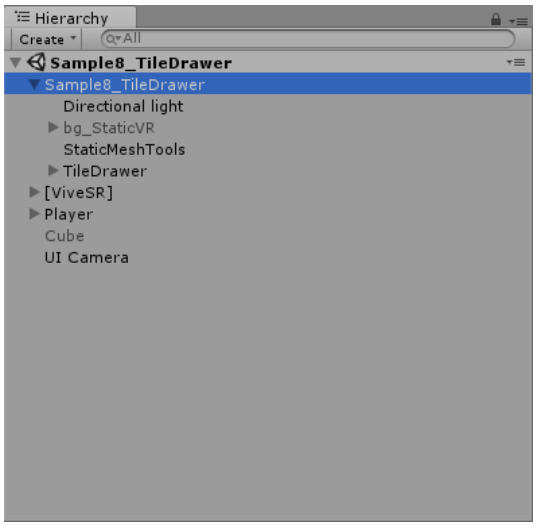

Figure 30. Sample8: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 31).

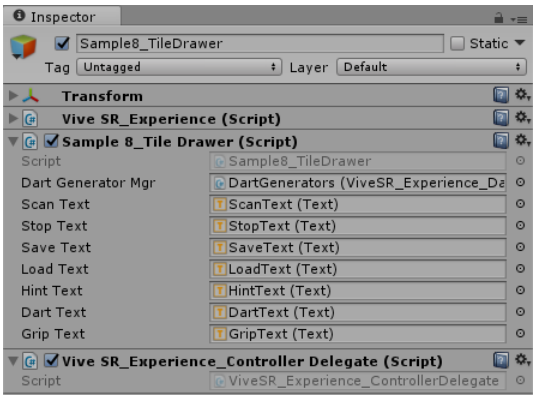

Figure 31. Sample8: Inspector Components

The following table explains what each Component does in the Sample8 (Table 8).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample7_ Tile Drawer	Determines how controller inputs work & lay tiles on a specified plane and the effect of Depth Occlusion
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 8. Sample8 Components

Sample9 – Semantic Segmentation

This sample demonstrates how to recognize scene objects. (Figure 32-34)

Demo Video: https://youtu.be/94uKZCV5fH4

Please follow the instructions below:

1. Scan and load the environment.
2. Select object.(Figure 32.)
   (1) Push '<' and '>' to show each type of objects.
   (2) Hold trigger. If the line attached to the controller becomes green, some object is detected.

3. Push '**Show All**' to show all recognized scene objects.
4. Push '**Clear**' to show hide all objects.

Figure 32. Screenshot of Selecting a Chair.

5. Show usable positions of any recognized object that some objects can be placed on it. The distance between those positions are 40cm
   (1) Check '**Enable Placer**' in the inspector as Figure 33.
   (2) Select object as **Step 2**. You will see the usable positions.

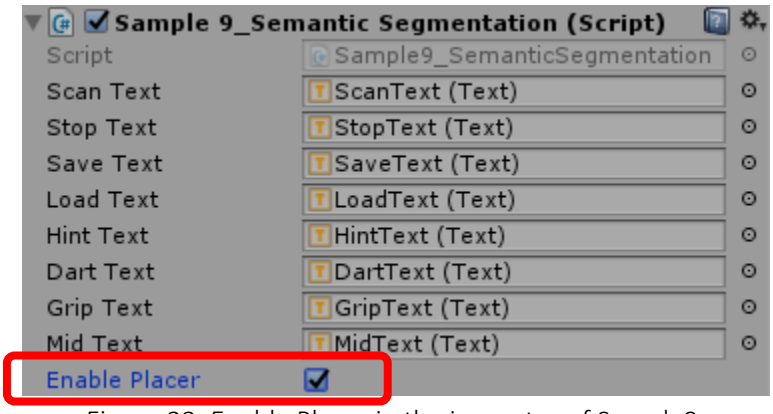

Figure 33. Enable Placer in the inspector of Sample9.

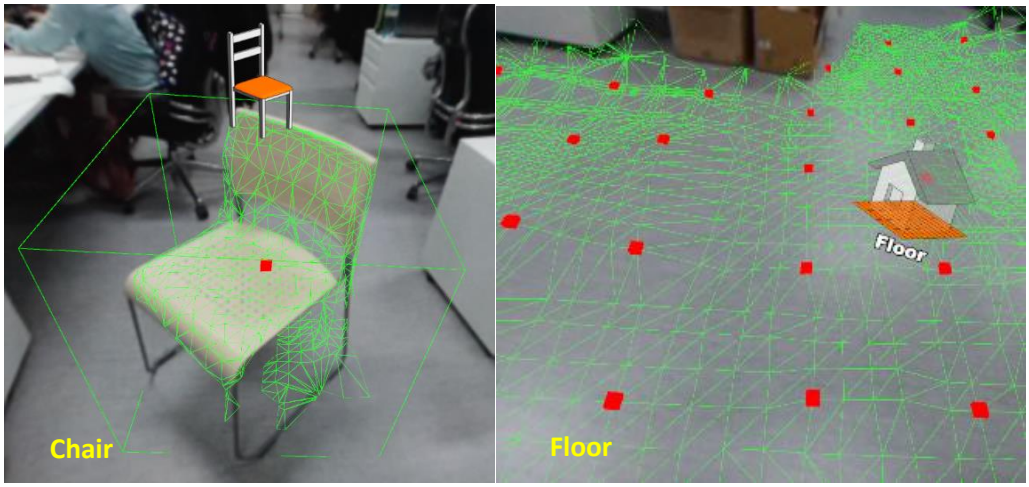

Figure 34. Usable Positions (the Red Points) of Chair and Floor.

The structure of these sample scenes are simple and straightforward. When you open **Sample9_SemanticSegmentation**, you can find a Gameobject that has the same name as the scene file (Figure 35).

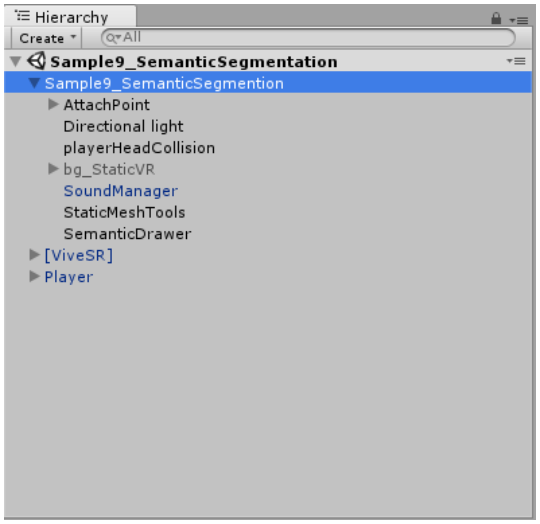

Figure 35. Sample9: Hierarchy Structure

In the inspector, there are already some Components attached to this Gameobject (Figure 36).

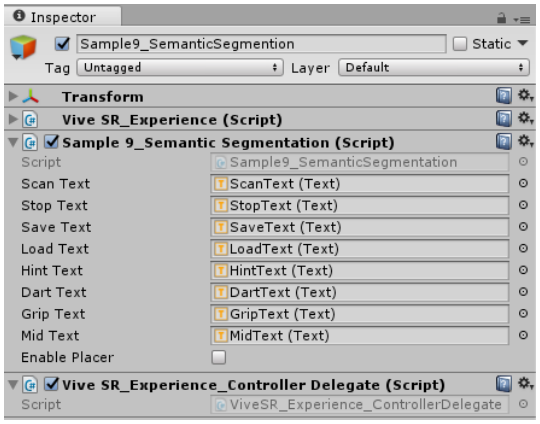

Figure 36. Sample9: Inspector Components

The following table explains what each Component does in the Sample9 (Table 9).

Component Name	Function
ViveSR_Experience	Provides basic functions such as checking if SteamVR and ViveSR are ready.
Sample7_ Semantic Segmentation	Determines how controller inputs work & recognize scene objects
ViveSR_Experience_ControllerDelegate	Updates controller inputs.

Table 9. Sample9 Components

NOTE: If you intend to build standalone executable for your project incorporating **AI Vision** module, you need to manually copy the deep learning models under /Assets/ViveSR/Plugins/model to /YOUR_STANDADLONE_EXE/Plugins/model. Still, you can override the path to the model files. Please refer to the included Unity sample script.

Specific Sample – Chaperone

This sample demonstrates how to create an OVERLAY application which can runs when another game is working. The Chaperone will detect whether there is any person in front of you and display an images overlaying your application (Figure 39).

In **Experience_Unity > Assets > ViveSR_Experience > Chaperone > Scenes** (Figure 37)

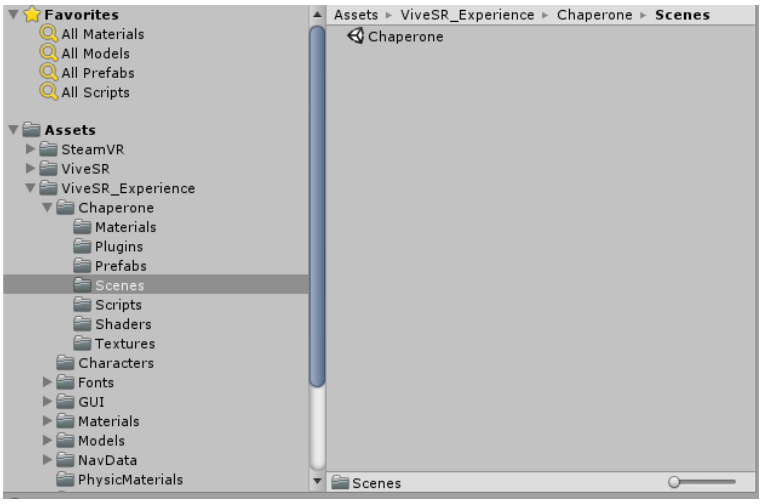

Figure 37. Chaperone scene

Vive log icon shows whether the environment is able to launch Chaperone.
Shield icon shows whether Chaperone is working.
Gray means Idle, blue means OK and red means Error.

Figure 38. Game view of Chaperone

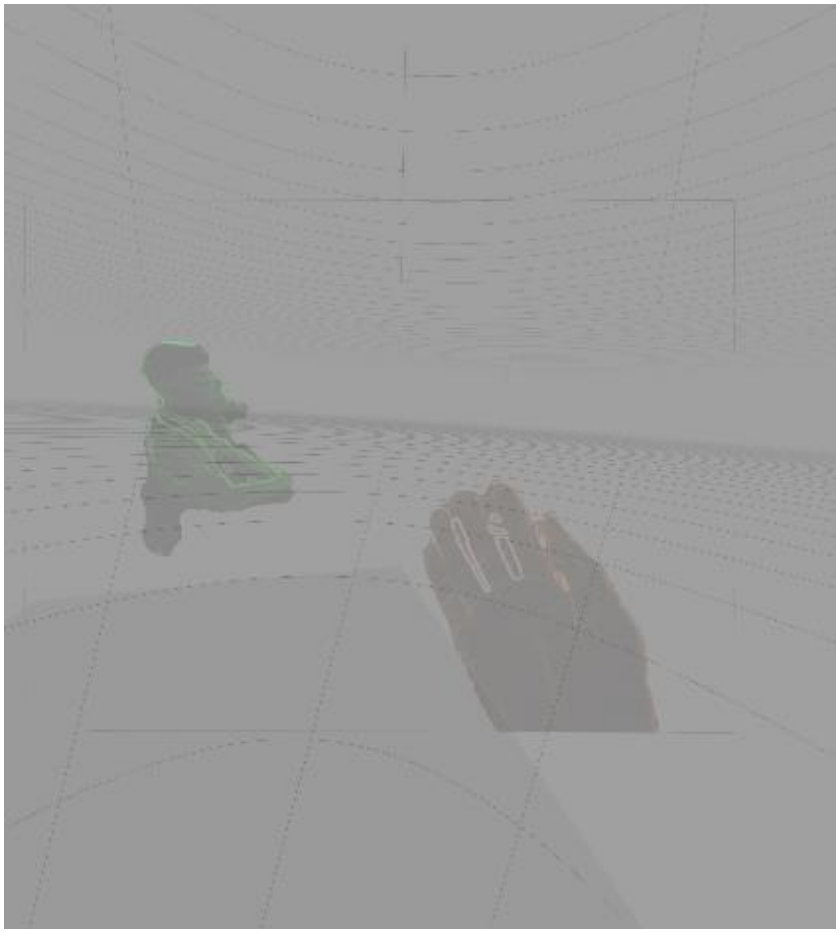

Figure 39. Mirror of HMD