VR Capture Plugin

Version 1.0.3

Created by RockVR

http://vrcapture.rockvr.com/

Contact: dev@rockvr.com

Table of Contents

1. Introduction and Overview

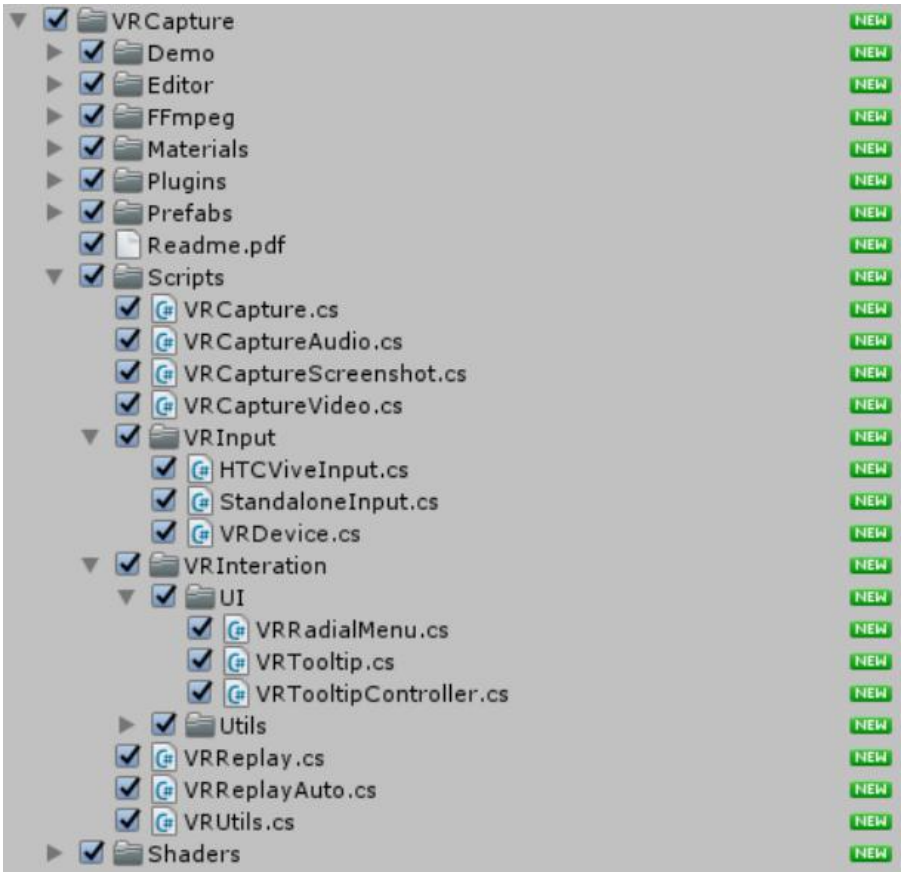
VR Capture is a plugin can help to create and share game play video. Many times there is no way to share your wonderful VR gameplay with friends in all angles. Now we provide a solution to solve your problems.

This plugin can be based on the requirements of the developer, recording the desired footage (flat or panorama video) in the VR game scene, and share with your friends (VOD or live streaming).

As we all know, record a 360 video from Unity game cost many resources and its impossible to do it in client side without hurt the game performance. However, performance means a lot for game especially VR game. How to enable 360 gameplay footage record and smooth VR game experience in the same time? VRCapture introduce a replay system VRReplay for Unity. During VR gameplay, VRReplay just record the game state and then send to our high performance rendering server to generate the 360 footage for customers, which totally avoid performance issue on the client side.

VR Capture include FFmpeg build, its a third party, open source, cross-platform tool that lets you easily convert video formats, and is bundled with VR Capture. You can learn more about FFmpeg through http://ffmpeg.org/.

When you import VR Capture plugin into your Unity project, the following assets will be added:



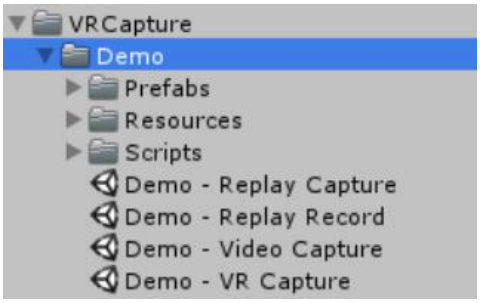
Demo/	Stores the scene file and all other assets for a fully functional demonstration of VR Capture. When importing to an existing project, you may choose to exclude this folder.
Editor/	Contains helper scripts and resources used in the Unity Editor and Inspector window.
FFmpeg/	Contains the FFmpeg binaries for Windows and Mac OSX. If you are only building for one target platform, you can exclude the file you don't need.
Plugins/	Contains the plugin platform depend native library.
Prefabs/	Contains useful prefabs can be drag and drop to your scene.
Scripts/	Contains the core logic scripts. Including capture, replay and interaction modules, etc.
Shaders/	Contains material's shader that required in plugin.

This guide covers integrating VR Capture to your own Unity project, and provides a detailed explanation on how the package works under the hood.

If you have any questions, feedback or having issues, please contact us directly at dev@rockvr.com. We will try our best to respond as quickly as possible.

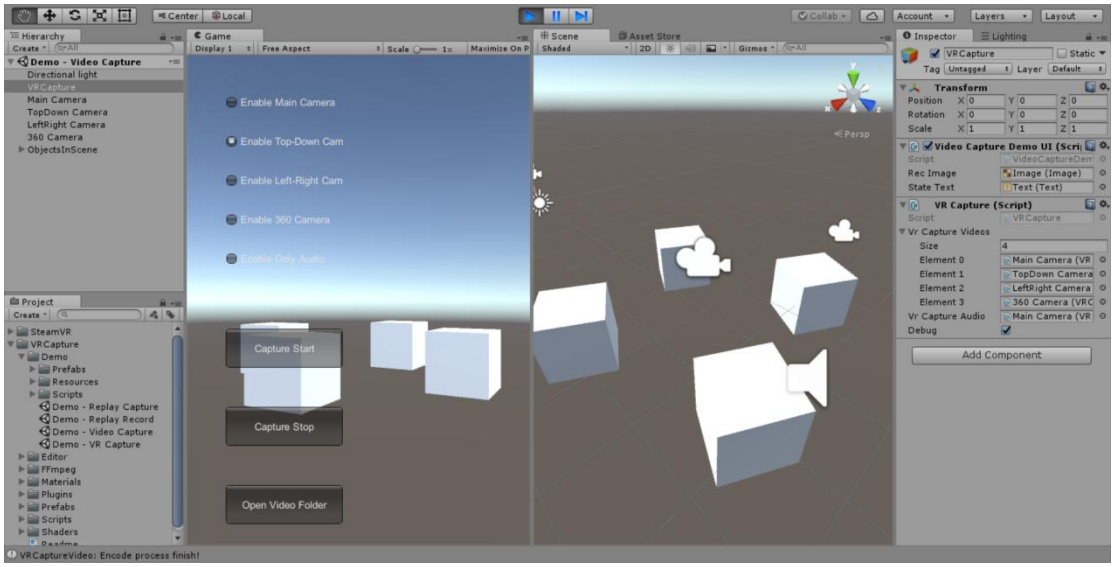
2. Demo Quick Start

VR Capture come with several demos to help you understanding functionality of plugin quickly. Start by creating creating a new project and importing all VR Capture package assets included demo scenes files.

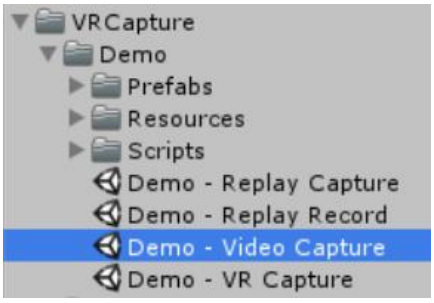


2.1 Video Capture Demo

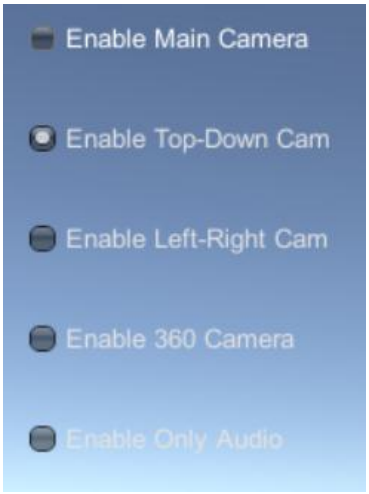
Video capture demo demonstrate video record functionality, you can test with multi-camera recording and 360 recording with it.



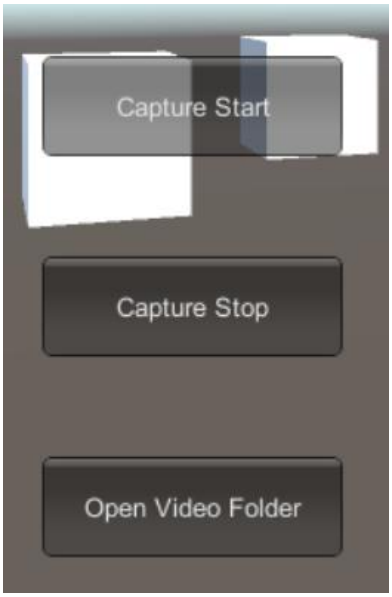
Step 1: Open the demo scene located in */Assets/VRCapture/Demo/Demo - Video Capture*:



Step 2: Play in editor, chose your desired camera:

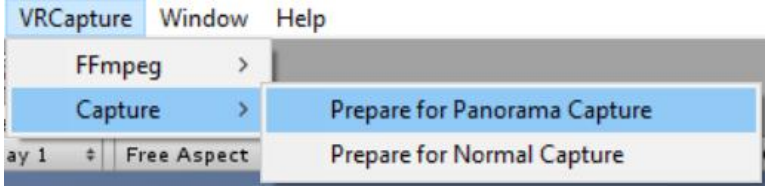


Step 3: Click *Capture Start* button, wait few minutes (depend how long you want record), and click *Capture Stop* button:



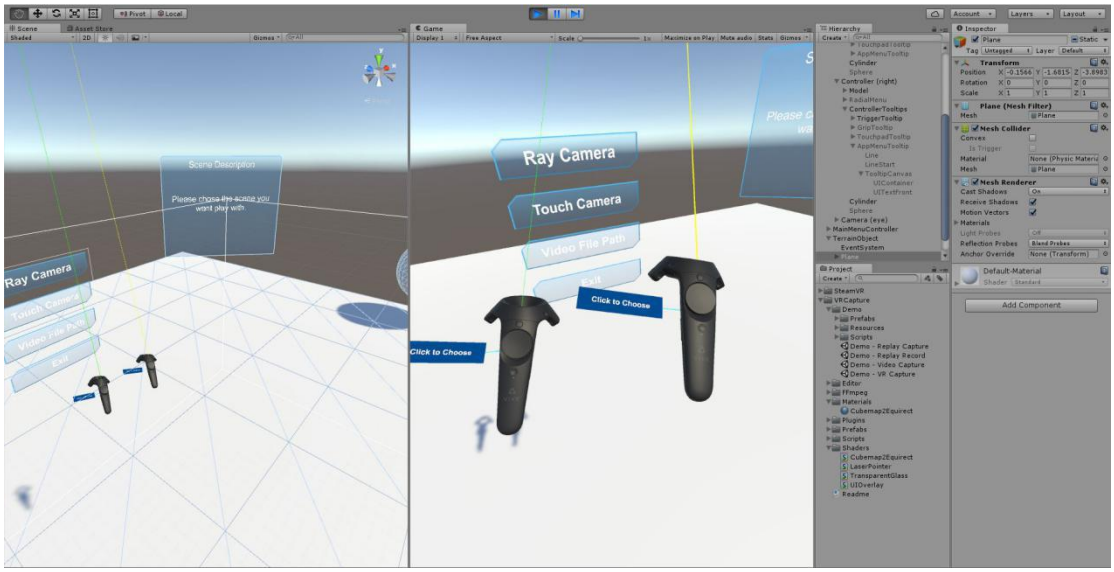
Step 4: Click *Open Video Folder* button to check out the video you just recorded.

Note: For 360 video record, it not support multi-camera record, and require running offline render mode during record session, that means it will move to next frame until the current frame record succeed. Before record start, please click *Prepare for Panorama Capture* item in VRCapture menu.

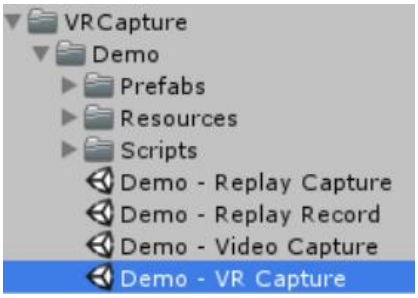


2.2 VR Capture Demo

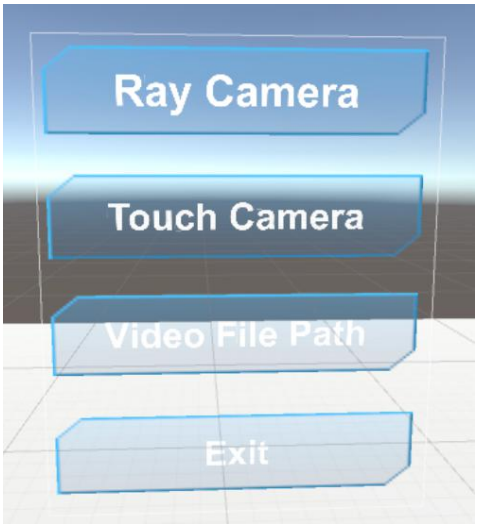
VR capture demo demonstrate how video capture component integrate with Steam VR interaction module. We provide several ways of interaction with recording camera, including touch, laser ray controller, etc.



Step 1: Open the demo scene located in */Assets/VRCapture/Demo/Demo - VR Capture*:

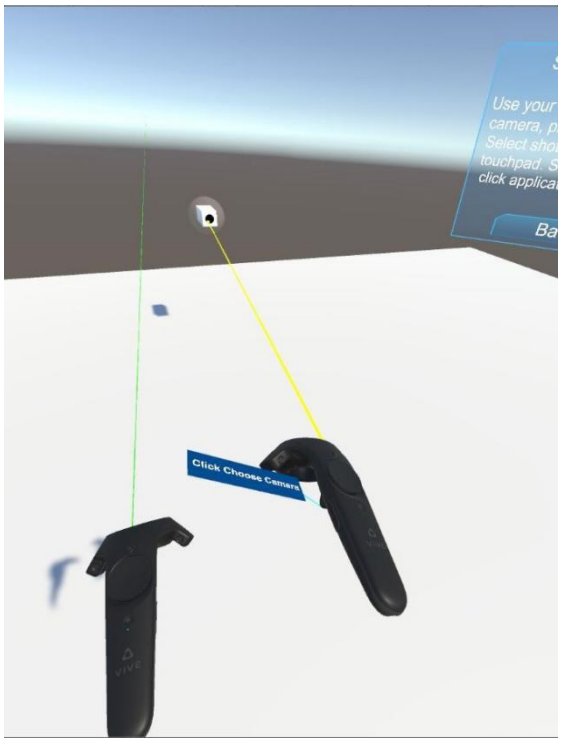


Step 2: Play in editor, chose your desired scene:

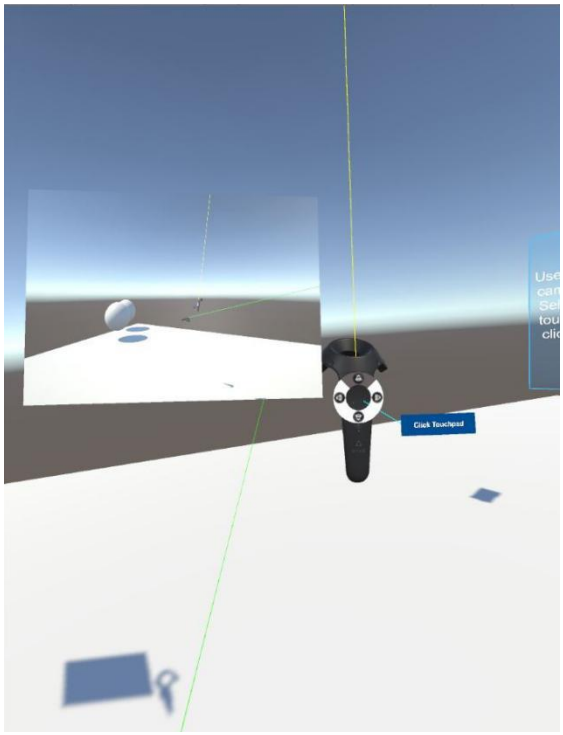


Ray Camera use laser to control camera and radial menu to set shooting position and angle;
Touch Camera use your controller to grab camera and adjust the shooting position and angle;
Video File Path will show you where are the recorded videos.

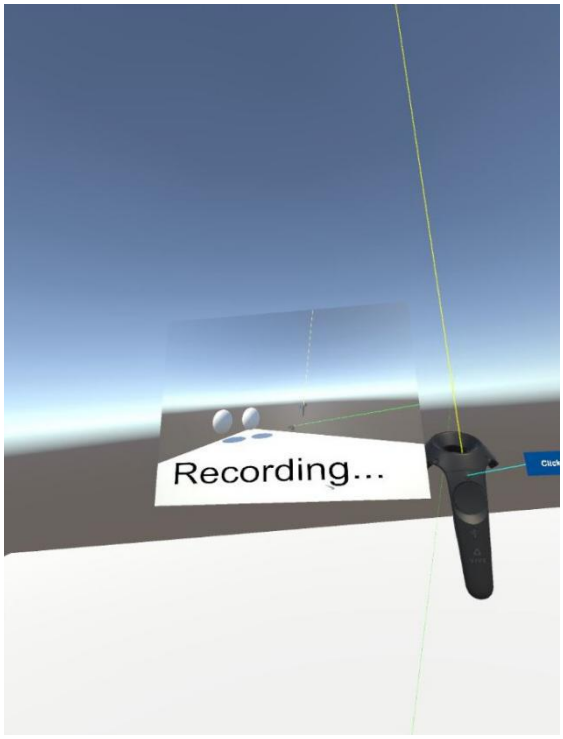
Step 3: Interact with recording camera. If you chose *Ray Camera* scene, you need using laser ray to select the camera:



Step 4: Once you grab the camera, you can use radial menu to chose the shooting position with pre-set value:

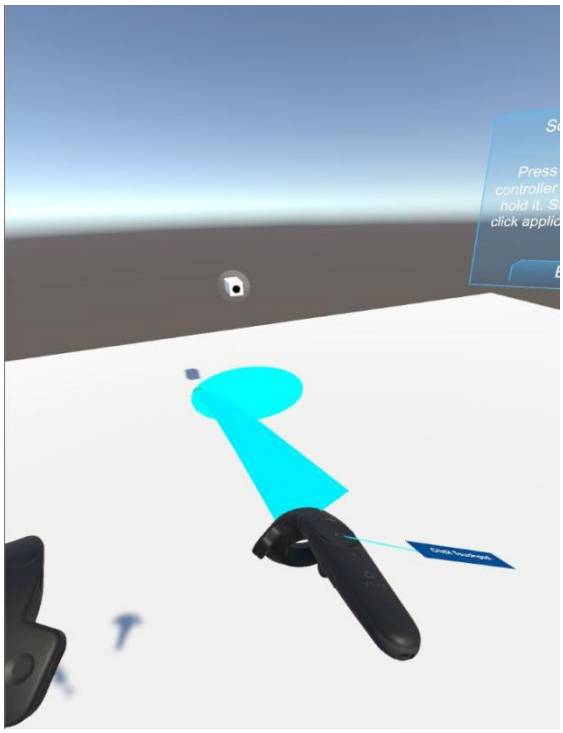


Step 5: After your perfect camera position set, press trigger to start video recording session:

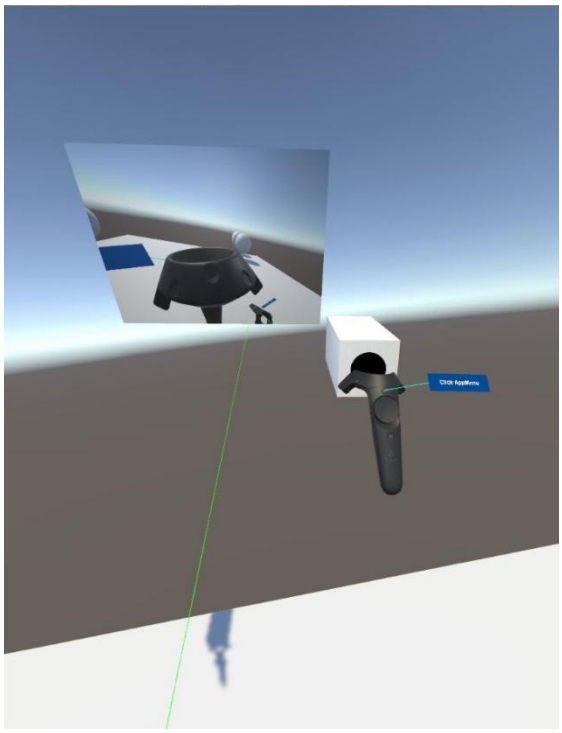


Checkout recorded video location by enter *Video File Path* scene.

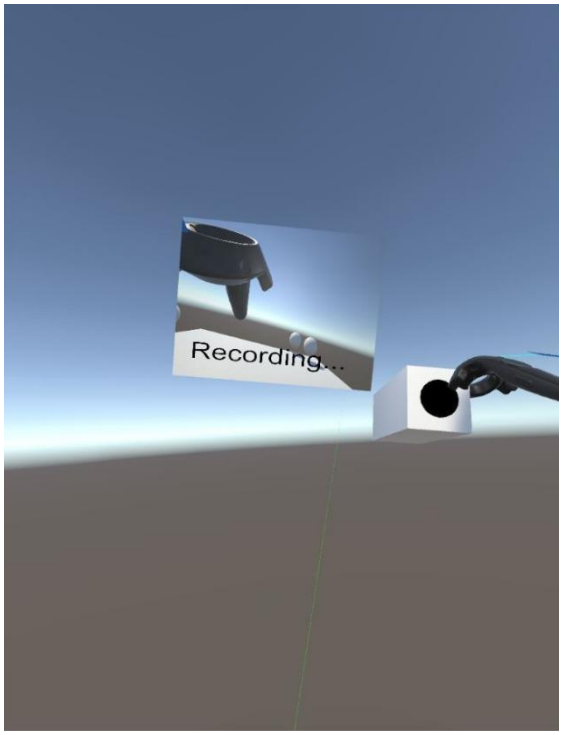
Note: If you chose *Touch Camera*, the process should be similar, please see steps below.
Step 3: Use teleport functionality to move close to camera object:



Step 4: Grab the camera and adjust desired position and angle:

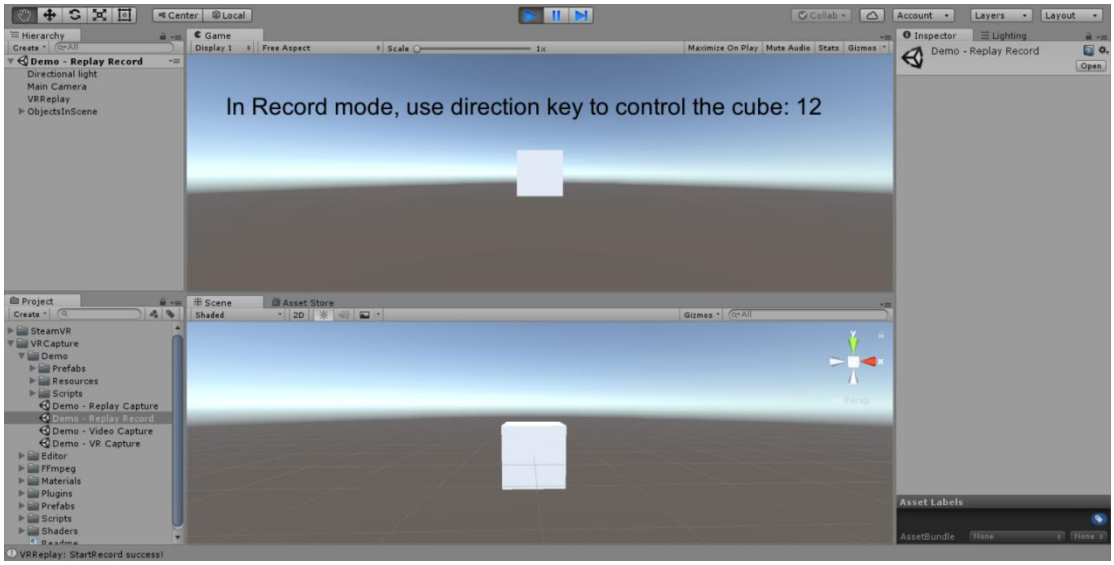


Step 5: Press trigger to start video recording session, also you can do recording while grab camera and move it around:

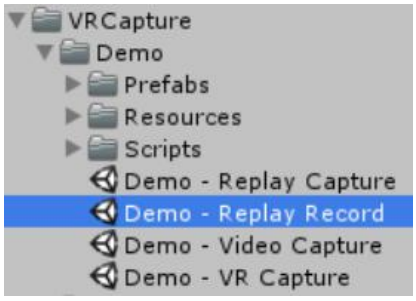


2.3 Replay Record / Capture Demo

Replay demo demonstrate how use VRReplay module to enable a simple Unity replay system. In this demo, *Replay Record* scene record user's input, and *Replay Capture* scene use that input data to re-run the same game.

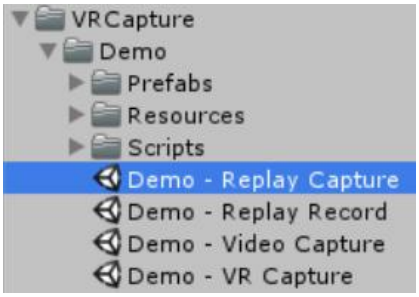


Step 1: Open the record demo scene located in */Assets/VRCapture/Demo/Demo - Replay Record*:

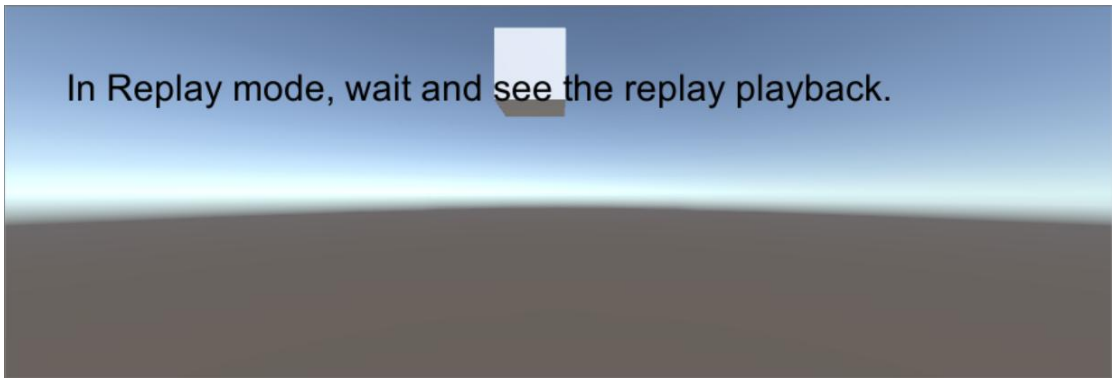


Step 2: Play in editor, the replay record will auto start, use direction key to control the cube, and it will auto stop in 15 seconds.

Step 3: Open the playback demo scene located in */Assets/VRCapture/Demo/Demo - Replay Capture*:



Step 4: Play in editor, the replay playback will auto start, you should see the cube move exactly same as you controlled previous:



3. Integration Guide

3.1 Integrate with VRCapture Module

VRCapture module implemented the core function of video/audio processing.

Step 1: Attach *VRCapture.cs* script to a game object (or you can just create a new empty object) in your scene, this script is used to manager the work of video and audio processor.

Step 2: Attach *VRCaptureAudio.cs* script to the main camera (which has Audio Listener) to enable the audio record function:

Step 3: Drag and drop needed camera prefab into your scene which located in */Assets/VRCapture/Prefabs/*, we provided 3 different video capture camera prefabs this time.

DedicatedCameraCapture - Used to capture perspective different with main camera.

MainCameraCapture - Used to capture main camera's perspective, the original main camera should be replaced with this prefab.

PanoramaCameraCapture - Used to capture 360 degree panorama video.

Step 4: Configure *VRCaptureVideo* component's properties.

Common properties:

Live Streaming - You can set video destination to remote RTMP server by check live streaming check box, you need fill out the remote RTMP server address.

Frame Size - Resolution of recorded video, the higher size, the better video quality, but more performance loss.
The available size are: 640x480, 720x480, 960x540, 1280x720, 1920x1080, 2048x1080, 3840x2160 and 4096x2160.

Encode Quality - Lower quality will decrease file size on disk and video bit rate. Available quality are: Low (1000 bit/s), Medium (2500 bit/s) and High (5000 bit/s).

Anti Aliasing - Set anti aliasing factor for frame captured, higher anti aliasing will increase video quality.

Target Framerate - Set target frame rate for recorded video, to avoid performance loss, use lower target framerate.

Dedicated Camera - Set false for main camera, and true for individual camera.

Enabled - Set true if you want use this camera, if set false it will be ignored during video capture process.

Flat video capture properties:

Format Type - Set as NORMAL for flat video capture.

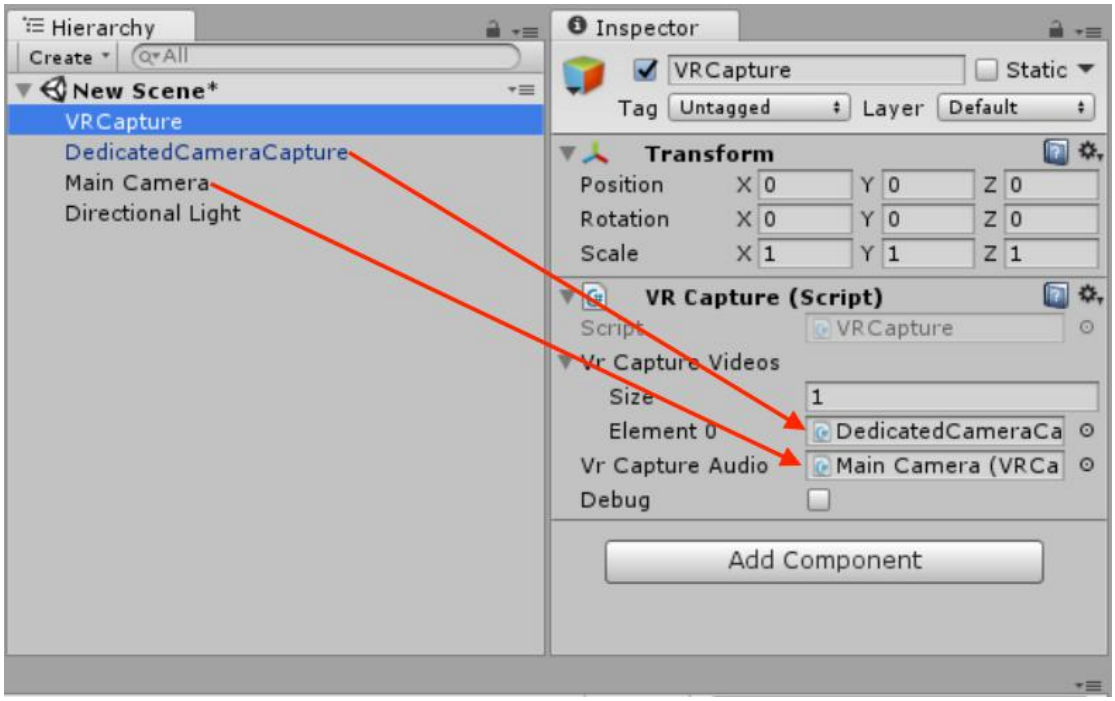
Panorama video capture properties:

Format Type - Set as PANORAMA for 360 video capture, `360 video capture camera always use dedicated camera.`

Projection Type - Currently you can chose EQUIRECTANGULAR or CUBEMAP, most video platform support equirectangular format, like Youtube, etc. However, cubemap format can reduce bitrate for generated video.

Cubemap Size - Square pixel size of frame captured by each direction camera. If use CUBEMAP type, *Frame Size* will not work, the size will be (3 x *Cubemap Size*) x (2 x *Cubemap Size*).

Step 5: Configure *VRCapture* component's public properties, drag the camera prefab (*VRCaptureVideo*) you selected before into VRCapture videos property (you can set multiply cameras), drag the main camera which attached *VRCaptureAudio* script into VRCapture audio property:



Step 6: Enable video capture function by code, *VRCapture* provide API to start or stop video recording, you can call those functions according to your requirements:

```
20              // Start video capture.
21              VRCapture.Instance.StartCapture();
22
23              // Game logic...
24
25              // Stop video capture.
26              VRCapture.Instance.StopCapture();
```

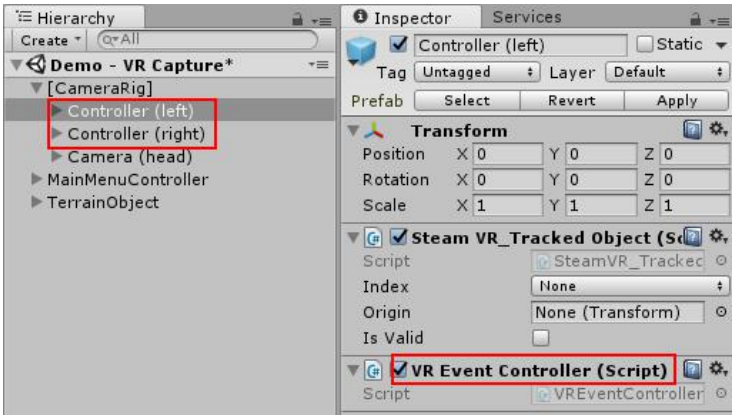
Default the video will be saved to, for more details please check *VRUtils.cs*:

```
23              string saveFolder = Environment.GetFolderPath(
24                  Environment.SpecialFolder.MyDocuments) + "/VRCapture/";
```

3.2 Integrate with VRInteration Module

VRInteration module implemented the core function of VR interaction control.

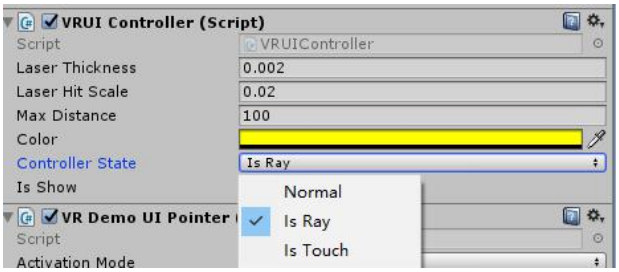
Step 1: Attach *VREventController* script to the Steam VR devices controller in the scene you want to control.



Note: *VREventController* is based on the *HTCViInput* and SteamVR plugin, it is a event script to control the Vive handle devices.

Step 2: Adding different interactive features of the script to the Steam VR devices after the first step.

VRUIController - Set the interaction patterns, choose Ray or touch to interacting.



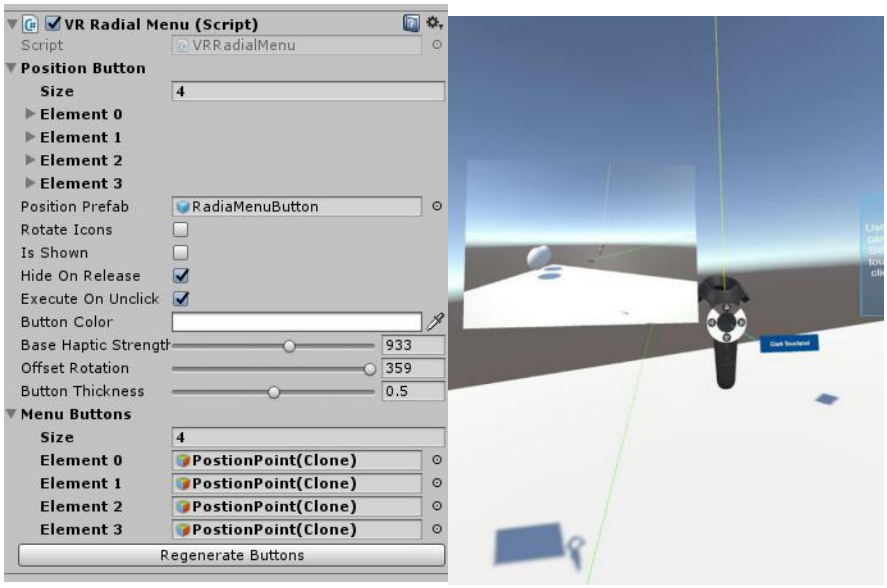
VRUIPointer - This script is used to take response for interaction event, wrapper for eventSystem. *VRInputModule* is designed to work as you would expect a SteamVR controller input to work. Events for button presses, dragging, and similar are sent in response to input.

```
72      private void Hover(VRUIPointer pointer, List<RaycastResult> results)
112
113     private void Click(VRUIPointer pointer, List<RaycastResult> results)
141
142     private void Drag(VRUIPointer pointer, List<RaycastResult> results)
```

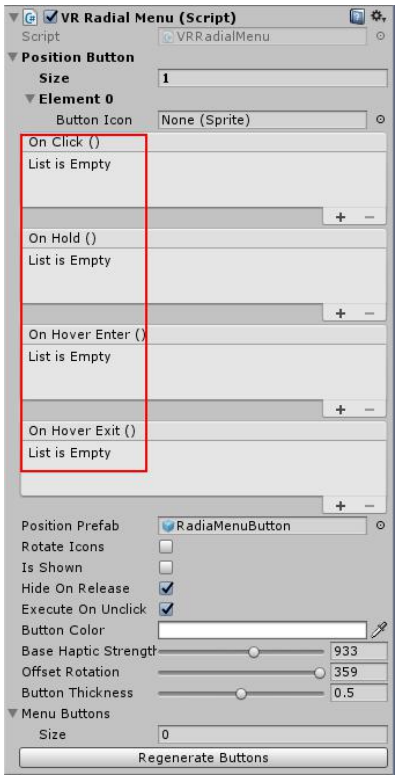
VRTeleport - Implemented functionality of teleport in VR scene. Enable SeachDownPoint or SureDownPoint function to implemented teleport.

```
233  public void SureDownPoint();
234  public void SeachDownPoint();
```

VRRadiaMenu - Set radial menu attribute, and binding keys to listener events.



Add different events to different objects in different RadiaMenuButtons state.



Step 3: Create a control management script to manage the handle events.

Create *VREventController* object.

```
23          public VREventController eventController;
```

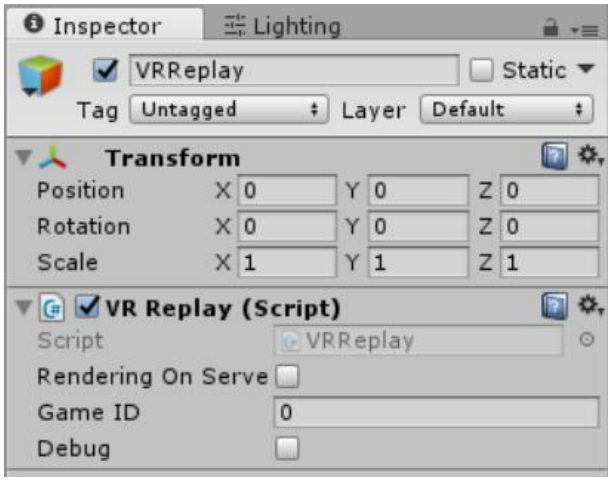
Register to create a delegated events.

```
59          void OnEnable() {
60              if(eventController != null) {
61                  eventController.OnPressApplicationMenuDown += OnPressApplicationMenuDown;
62                  eventController.OnPressTrigger += OnPressTrigger;
63                  eventController.OnPressTouchpadUp += OnPressTouchpadUp;
64                  eventController.OnSwipeLeft += OnSwipeLeft;
65                  eventController.OnSwipeRight += OnSwipeRight;
66                  eventController.OnPressTriggerUp += OnPressTriggerUp;
67                  eventController.OnPressTouchpad += OnPressTouchpad;
68                  eventController.OnTouchPadTouch += OnTouchPadTouch;
69                  eventController.OnTouchPadTouchUp += OnTouchPadTouchUp;
70                  eventController.OnPressTouchpadDown += OnPressTouchpadDown;
71              }
72          }
```

3.3 Integrate with VRReplay Module

VRReplay module implemented the core function of record/replay game states.

Step 1: Attach *VRReplay* script to a game object in the scene you want to record gameplay.



Note: Rendering on server feature require integrate with our server side rendering server, please contact us if you do want this feature.

Step 2: Using *VRReplay* Code API to record game state:

StandaloneInput - Record user input for standalone platform, provide the same API as Unity's Input. Replace the input API usage withe *VRReplay* API as required, example usage:

```
18              // Replace
19              if (Input.GetKey(KeyCode.UpArrow)) {
20                  transform.Translate(new Vector3(0f, 0.1f, 0f));
21              }
22              // with:
23              if (StandaloneInput.GetKey(KeyCode.UpArrow)) {
24                  transform.Translate(new Vector3(0f, 0.1f, 0f));
25              }
```

HTCViveInput - Record user input for HTC Vive device, provide the same API as SteamVR_Controller.Device. Replace the input API caller with *VRReplay* API as required, example usage:

```
130             int trackedIndex = (int)this.GetComponent<SteamVR_TrackedObject>().index;
131             // Replace
132             SteamVR_Controller.Device steamDevice =
133                 SteamVR_Controller.Input(trackedIndex);
134             if (steamDevice.GetPress(EVRButtonId.k_EButton_SteamVR_Trigger)) {
135                 // Do action when pressed trigger...
136             }
137             // With:
138             HTCViveInput.HTCViveDevice replayDevice =
139                 HTCViveInput.Input(this.gameObject, trackedIndex);
140             if (replayDevice.GetPress(EVRButtonId.k_EButton_SteamVR_Trigger)) {
141                 // Do action when pressed trigger...
142             }
```

Object Transform - If the game object is not controlled by user input, that's also can be recorded by transform record system, example usage:

```
18              // Add current game object for recording its transform.
19              VRReplay.Instance.AddTransform(this.transform);
```

Custom Information - In addition, VRReplay support record custom information required during replay, example usage:

```
18              // Save float sequence:
19              VRReplay.Instance.RecordSequenceFloat("PositionY", 1f);
20              VRReplay.Instance.RecordSequenceFloat("PositionY", 2f);
21              // Query float sequence:
22              float first = VRReplay.Instance.QuerySequenceFloat("PositionY");
23              float second = VRReplay.Instance.QuerySequenceFloat("PositionY");
```

After setup record user input, game object transform and custom info, etc. We can start to record and replay our gameplay, *VRReplay* provide useful API to achieve those functionality easily, for recording, just call:

```
18              VRReplay.Instance.StartRecord();
```

Try play your game for a while, then quit the game. The replay file should be saved to default save folder, for more details please check *VRUtils.cs*:

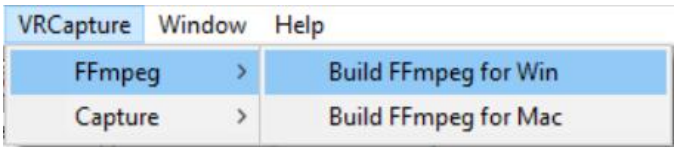
```
22              string saveFolder = Environment.GetFolderPath(
23                  Environment.SpecialFolder.MyDocuments) + "/VRCapture/Replays/";
```

Step 3: Using *VRReplay* Code API to replay game by replay file:

```
20              if (VRReplay.Instance.LoadReplay(
21                  VRReplayUtils.SaveFolder + "xxxx-xx-xx-xx-xx-xx.txt")) {
22                  VRReplay.Instance.StartReplay();
23              }
```

4. Prepare Build

To enable video record functionality in your game build, its require add FFmpeg build. Based on your platform, click corresponding option in VRCapture menu:



5. Feedback

If you have any feedback to VR Capture plugin, please email us directly, and you can help us to improve our product by fill out the Feedback Questionnaire, your suggestion will be very valuable to us. If you plan integrate a plugin into your game, please contact us and we will provide more help to let you share your awesome game more efficient.