Overview

Chapters:

Oculus Rift support is now included in CRYENGINE. This article covers the programming side of interacting with a Rift headset. For information on basic use, see Oculus Rift.

Code

This section describes how to access VR in general, but specifically the Oculus from Game Code.

Check VR Setup

In order to check if the current VR setup is valid, you can check if the VR related pointers are set and valid:

C++

```
if (IHmdManager* pHmdManager = gEnv->pSystem->GetHmdManager()) // Check, if the HMD Manager exists
{
	if (IHmdDevice* pDevice = pHmdManager->GetHmdDevice()) // Check, if a valid HMD device is connected
	{
		if (pDevice->GetClass() == EHmdClass::eHmdClass_Oculus) // Check, if the connected device is
an Oculus device)
		{
			//your code
		}
	}
}
```

C#

```
IHmdManager pHmdManager = Global.gEnv.pSystem.GetHmdManager ();
if (pHmdManager != null) { // Check, if the HMD Manager exists
	IHmdDevice pDevice = pHmdManager.GetHmdDevice ();
	if (pDevice != null && pDevice.GetClass() == EHmdClass.eHmdClass_Oculus) { // Check, if a valid
Oculus device is connected
		// your code
	}
}
```

Get HMD Device Tracking State

You can retrieve the position and rotation of the HMD device. While the camera will usually be controlled by the ViewSystem directly, this might be useful to setup additional checks and game-play logic that takes the user's head position into consideration. In the following code samples we will assume pDevice to be set according to 'Check VR Support':

C++

```
// Get the current tracking state
HmdTrackingState state = pDevice->GetLocalTrackingState();
// Either use orientation and position directly, or build a TransformationMatrix from them
Matrix34 HMDtm = Matrix34(Vec3(1, 1, 1), state.pose.orientation, state.pose.position);
// You could not apply the values to an entity. For example, to position the entity 'pEntity' relative to
the entity 'pReference':
pEntity->SetWorldTM(pReference->GetWorldTM() * HMDtm);
pEntity->InvalidateTM();
```

C#

```
HmdTrackingState state = pDevice.GetLocalTrackingState();
// Either use orientation and position directly, or build a TransformationMatrix from them
Matrix34 HMDtm = new Matrix34(new Vec3(1, 1, 1), state.pose.orientation, state.pose.position);
// You could not apply the values to an entity. For example, to position the entity 'pEntity' relative to
the entity 'pReference':
pEntity.SetWorldTM(pReference.GetWorldTM() * HMDtm);
pEntity.InvalidateTM();
```

Get Controller Tracking State

You can get the controller's tracking states similar to the HMD's tracking state.

C++

```
// The IHmdController is handling all VR controllers associated with the current VR device.
const IHmdController* pController = pDevice->GetController();
// Make sure the desired controller is connected (the OpenVR implementation in CRYENGINE currently supports
controller ID 1 and 2)
if (pController->IsConnected(eHmdController_OpenVR_1))
{
	// Get the current tracking state
	HmdTrackingState state = pController->GetLocalTrackingState(eHmdController_OpenVR_1);
	// either use orientation and position directly, or build a TransformationMatrix from them
	Matrix34 controllerTM = Matrix34(Vec3(1, 1, 1), state.pose.orientation, state.pose.position);
	// You could not apply the values to an entity. For example, to position the entity 'pEntity'
relative to the entity 'pReference':
	pEntity->SetWorldTM(pReference->GetWorldTM() * controllerTM);
	pEntity->InvalidateTM();
}
```

C#

```
// The IHmdController is handling all VR controllers associated with the current VR device.
IHmdController pController = pDevice.GetController();
// Make sure the desired controller is connected (the OpenVR implementation in CRYENGINE currently supports
controller ID 1 and 2)
if (pController.IsConnected(EHmdController.eHmdController_OpenVR_1))
{
	// Get the current tracking state
	HmdTrackingState state = pController.GetLocalTrackingState(EHmdController.eHmdController_OpenVR_1);
	// either use orientation and position directly, or build a TransformationMatrix from them
	Matrix34 controllerTM = new Matrix34(new Vec3(1, 1, 1), state.pose.orientation, state.pose.position);
	// You could not apply the values to an entity. For example, to position the entity 'pEntity'
relative to the entity 'pReference':
	pEntity.SetWorldTM(pReference.GetWorldTM() * controllerTM);
	pEntity.InvalidateTM();
}
```

React to Controller Input

The Oculus Touch controller provides more than just tracking information. It has some buttons, a trigger and a touchpad, and you have several options should you want to react to inputs other than just tracking information:

- **Action Maps:** Please read **Setting Up Controls and Action Maps.** The relevant button names are:

Name	Type	Description
otouch_a	Button	

otouch_b	Button	
otouch_x	Button	
otouch_y	Button	
otouch_l3	Trigger	Left thumb button (stick).
otouch_r3	Trigger	Right thumb button (stick).
otouch_triggerl_btn	Button	Left trigger button.
otouch_triggerr_btn	Button	Right trigger button.
otouch_l1	Trigger	Left index trigger.
otouch_r1	Trigger	Right index trigger.
otouch_l2	Trigger	Left hand trigger.
otouch_r2	Trigger	Right hand trigger.
otouch_stickly	Axis	Left stick vertical motion.
otouch_stickry	Axis	Right stick vertical motion.
otouch_sticklx	Axis	Left stick horizontal motion.
otouch_stickrx	Axis	Right stick horizontal motion.

- **InputListener:** Please read **CryInput.** Check for DeviceType **eIDT_Motion** and KeyIds:

- eKI_Motion_OculusTouch_A
- eKI_Motion_OculusTouch_B
- eKI_Motion_OculusTouch_X
- eKI_Motion_OculusTouch_Y
- eKI_Motion_OculusTouch_L3
- eKI_Motion_OculusTouch_R3
- eKI_Motion_OculusTouch_TriggerBtnL
- eKI_Motion_OculusTouch_TriggerBtnR
- eKI_Motion_OculusTouch_L1
- eKI_Motion_OculusTouch_R1
- eKI_Motion_OculusTouch_L2
- eKI_Motion_OculusTouch_R2
- eKI_Motion_OculusTouch_StickL_Y
- eKI_Motion_OculusTouch_StickR_Y
- eKI_Motion_OculusTouch_StickL_X
- eKI_Motion_OculusTouch_StickR_X
- eKI_Motion_OculusTouch_Gesture_ThumbUpL
- eKI_Motion_OculusTouch_Gesture_ThumbUpR
- eKI_Motion_OculusTouch_Gesture_IndexPointingL
- eKI_Motion_OculusTouch_Gesture_IndexPointingR

- **Polling:** Additionally, you can poll the current controller state. A nice bonus of this approach, is that you will not only get the information if a button is pressed, but also (in most cases) if the button is just touched:

C++

```
// The IHmdController is handling all VR controllers associated with the current VR device.
const IHmdController* pController = pDevice->GetController();
// Make sure the desired controller is connected (the OpenVR implementation in CRYENGINE currently supports
controller ID 1 and 2)
if (pController->IsConnected(eHmdController_OpenVR_1))
{
        // Is the Application A button pressed?
        bool bAppMenuPressed = pController->IsButtonPressed(eHmdController_OculusLeftHand,
eKI_Motion_OculusTouch_A);
        // or is it touched?
        bool bAppMenuTouched = pController->IsButtonTouched(eHmdController_OculusLeftHand,
eKI_Motion_OculusTouch_A);
        // Is the L2 pressed?
        bool bGripPressed = pController->IsButtonPressed(eHmdController_OculusLeftHand,
eKI_Motion_OculusTouch_L2);
        // or is it touched?
        bool bGripTouched = pController->IsButtonTouched(eHmdController_OculusLeftHand,
eKI_Motion_OculusTouch_L2);
        // Get Trigger value
        float fTrigger = pController->GetTriggerValue(eHmdController_OculusRightHand,
eKI_Motion_OculusTouch_TriggerBtnR);
        // Get the Gesture state
        Vec2 vTouchPad = pController->IsGestureTriggered(eHmdController_OculusRightHand,
eKI_Motion_OculusTouch_Gesture_ThumbUpR);
}
```

C#

```
// The IHmdController is handling all VR controllers associated with the current VR device.
IHmdController pController = pDevice.GetController();
// Make sure the desired controller is connected (the OpenVR implementation in CRYENGINE currently supports
controller ID 1 and 2)
if (pController.IsConnected(EHmdController.eHmdController_OpenVR_1))
{
        // Is the Application A button pressed?
        bool bAppMenuPressed = pController.IsButtonPressed(EHmdController.eHmdController_OculusLeftHand,
EKeyId.eKI_Motion_OculusTouch_A);
        // or is it touched?
        bool bAppMenuTouched = pController.IsButtonTouched(EHmdController.eHmdController_OculusLeftHand,
EKeyId.eKI_Motion_OculusTouch_A);
        // Is the L2 pressed?
        bool bGripPressed = pController.IsButtonPressed(EHmdController.eHmdController_OculusLeftHand, EKeyId.
eKI_Motion_OculusTouch_L2);
        // or is it touched?
        bool bGripTouched = pController.IsButtonTouched(EHmdController.eHmdController_OculusLeftHand, EKeyId.
eKI_Motion_OculusTouch_L2);
        // Get Trigger value
        float fTrigger = pController.GetTriggerValue(EHmdController.eHmdController_OculusRightHand, EKeyId.
eKI_Motion_OculusTouch_TriggerBtnR);
        // Get the Gesture state
        Vec2 vTouchPad = pController.GetThumbStickValue(EHmdController.eHmdController_OculusRightHand,
EKeyId.eKI_Motion_OculusTouch_Gesture_ThumbUpR);
}
```

Oculus HRTF Plugin Support in Wwise

To activate HRTF plugin support in Wwise, set the *AUDIO_HRTF* option in CMake.