TOOLKIT

Requirements: *This package requires the latest version of Oculus and Unity to work.*
This document covers the custom actions of the Oculus Playmaker toolkit as well as the demo scene included. Actions provided below will get you started in creating your own Oculus title. This package is based the OVRInput API and uses its scripts and functions for the actions to work.

Table of Contents:

This package contains: A demo scene using most of the actions to show them in use. Actions necessary to use Oculus in playmaker. CC License models, including a robot.fbx, gun.obj,

Any questions or comments please contact [*frametaleinfo@gmail.com*]

boxingGloves.obj, and a punchingMachine.obj. Documentation listing the features and functions in this package.

Setting up and Understanding Oculus

These actions are based on the OVRInput api, which is the main api used for Unity, and is available in the Oculus developer page for Unity. These actions will make the process of development much easier and faster compared to the built-in input.

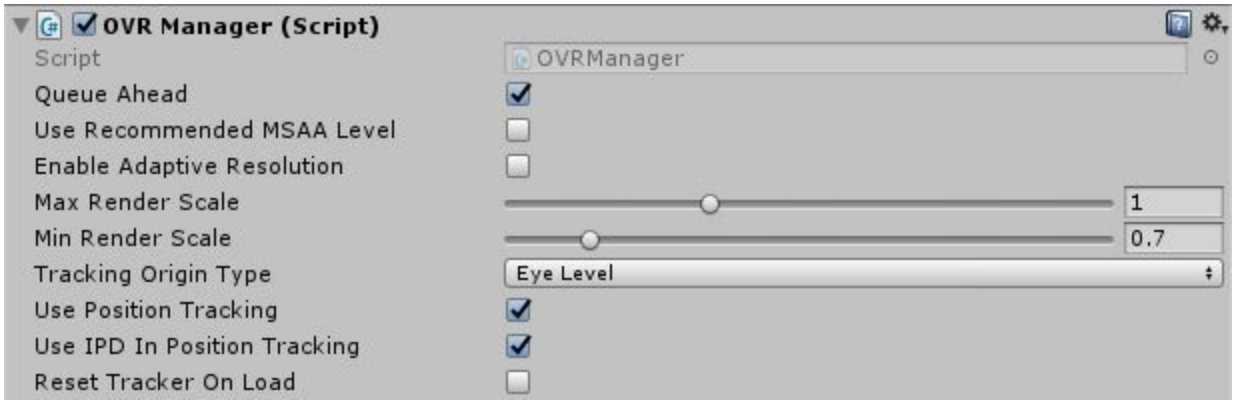
To begin using the Oculus actions in playmaker the OVR Manager (Script) must be placed on any game object in the scene. This script can be located in the Oculus Package located on the Oculus developer page for Unity.

https://developer.oculus.com/downloads/package/oculus-utilities-for-unity-5/

Once the Oculus package is imported you may place the OVRCameraRig prefab into your scene shown in ref.1. Or simply place the OVR Manager Script into a game object to get started.

[ref.1]

This OVRCameraRig contains a script (ref.2) which allows you to alter settings in Unity and Oculus.



[ref.2]

This script is the backbone for the asset and allows you certain features that affect Unity during play, including the resolution, the tracking type, tracking the position and the ability to reset the position of the headset on each load of a scene. For more information, please view the Oculus Unity utilities linked here.

Understanding the Controller and Headset

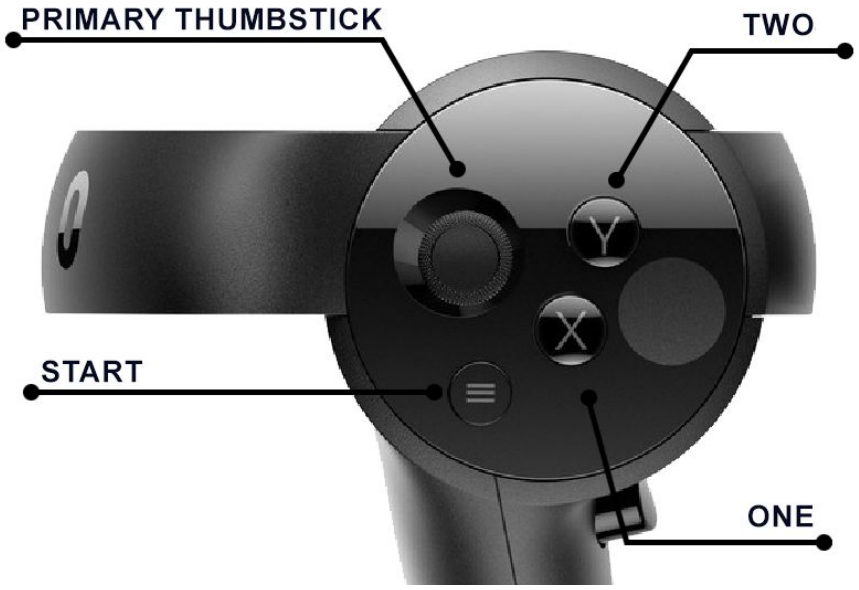
Inside the Ovr Camera Rig (ref.1) are anchors for the camera and both controllers. The LeftHandAnchor and RightHandAnchor are game objects that track the position and rotation of the controllers and can be used to set child objects to simulate the placement.

The CentralEyeAnchor contains the camera and audio listener. If the **Use Per Eye Cameras** is enabled on the OVR Camera Rig (script), the LeftEyeAnchor and the RightEyeAnchor will be the dual cameras while the CentralEyeAnchor will be used as the Audio Listener.

Both the left controller and right controller have their own set of buttons which will be demonstrated in the sections below. There are some buttons that are reserved such as the volume controls on the remote and the oculus button, that cannot be called.

Note: It is important when dealing with the Oculus that the controllers are activated when the headset is on the user's head as the scene is playing. This is especially true for any rumble haptic features as well.

Get Touch Button



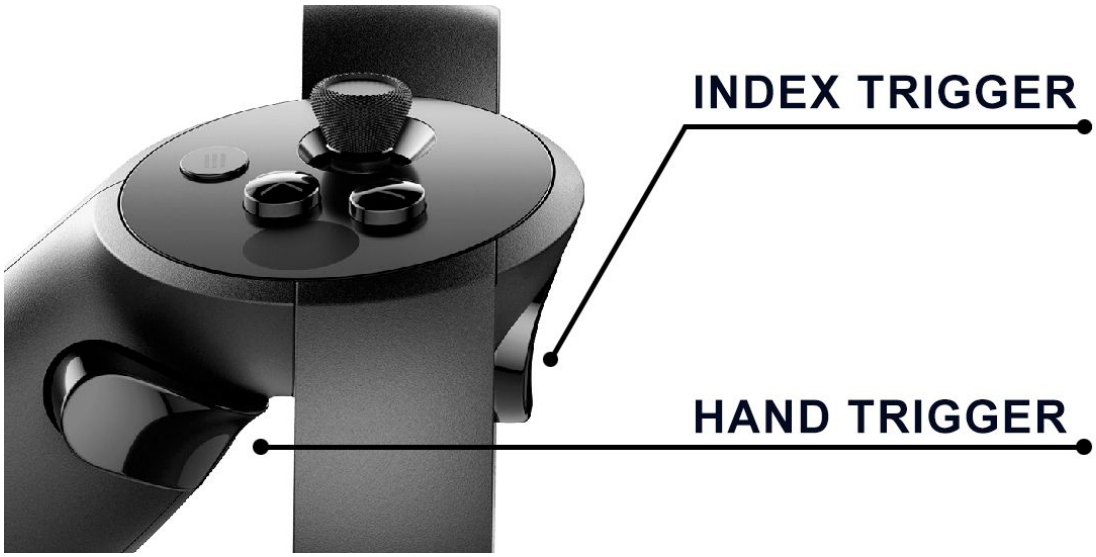
The Get Touch Button action can be used to send an event based on a button press. Aside from the Start button, the setup is the same for both controllers. The **Oculus Button** on the left controller is reserved in the API and can not be currently used.

Touch Controller: The Tracked Object required to run event.
Button: Select the button to trigger the event.
Button Type: Select between type to trigger an event.
Send Event: Event to send when the specified Button Type is used.
Store Result: Set a bool to true when Button Type is used.

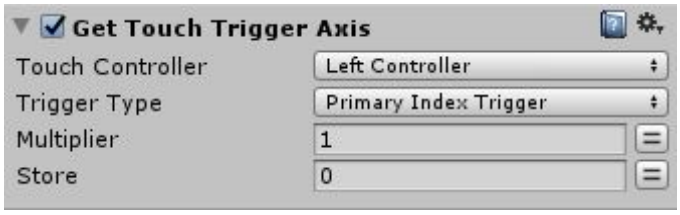
Tip: Do you need the stored result to notify if the button is either down (true) or up (false)? This action runs Every Frame and if set to Get Button, it will flip from **true** to **false** depending on press.

Example Usage:
Get Button Down and Get Button Up are great for UI buttons in game. Use Get Button Down to start a UI button animation, and stop that animation with Get Button Up.

Get Touch Trigger Axis



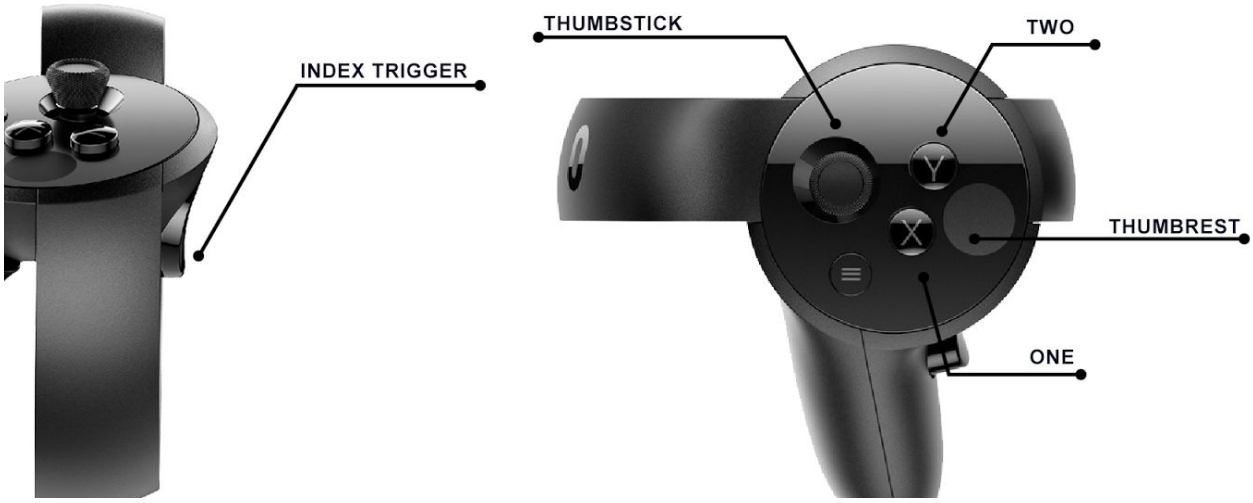
The Get Touch Trigger Axis can be used to send an event based on a Trigger type and Touch Controller.



Touch Controller: The controller required to run event.
Trigger Type: Select between type to trigger an event.
Multiplier: Multiplies the trigger value by this amount. Default set to 1.
Store: Stores the amount of pressure being held down on the trigger type.

Example Usage:
Primary Index Trigger - Great for firing weapons or drawing back bows.
Primary Hand Trigger - A popular usage is for picking up objects.

Get Touch Touchpad



The Get Touch Touchpad can be used to send an event based on a Touchpad type. The event is based on the touch of the button itself and not the press.

Touch Controller: The touch controller required to run event.
Button Type: Select between type to trigger an event.
Button: Select the button to trigger an event.
Send Event: Event to send when the specified Touchpad Type is used.
Store Result: Set a bool to true when Touchpad Type is used.

Note: The buttons have their own sensors that activate by simply touching the button, they also have sensors based on proximity to the button. The latter is activated using Get Touch Near Touchpad.

Get Touch Near Touchpad

The Get Touch Near Touchpad can be used to send an event based on finger/thumb near the buttons.

Touch Controller: The touch controller required to run event.
Button Type: Select between type to trigger an event.
Button: Select the button to trigger an event.
Send Event: Event to send when the specified Touchpad Type is used.
Store Result: Set a bool to true when Touchpad Type is used.

Example Usage: Near Touchpad is used in the Oculus examples for pointing when the Index finger is off the Index trigger.

Get Touch Thumbstick Axis

The Get Touch Thumbstick Axis grabs the X and Y value based on the player's thumbstick position.

Controller: The Tracked Object required to grab axis.
Multiplier: Adds a multiplier to the X and Y. Default set to 1.
Store X: Sets the float X thumbstick position into a variable.
Store Y: Sets the float Y thumbstick position into a variable.

Get Touch Controller Vibration

Activates the Haptic Feedback on a controller.

Touch Controller: The Tracked Object required to set haptic feedback.
Frequency: Set the Frequency. Value range from 0 to 1
Amplitude: Set the Amplitude. Value range from 0 to 1
Duration: Set the length of the haptic feedback on the controller.
Finish Event: Event to send when the specified duration is finished.

Note [Required]: The headset needs to be on the players head for the vibration to work.

Tip: Amplitude is the strength of the Vibration while the Frequency would be the intensity of the vibration. The lower the frequency the lower the "pitch" of the vibration. Test between 0 - .5 for a lower vibration compared to 5.1 - 1.

Tip: If the Duration is set to -1, the Vibration will continue as long as the state is active.

Get Touch Controller Velocity

Determines the current speed of the controller.

Touch Controller: The touch controller required to obtain amount of acceleration.
Controller Acceleration: The local Vector3 value of the acceleration.

Get Touch Controller Angular Velocity

Determines the current angular speed of the controller.

Touch Controller: The touch controller required to obtain amount of acceleration.
Controller Acceleration: The local Vector3 value of the acceleration.

Note: For the Oculus there are two similar actions to determine the speed of the controller. Velocity and Acceleration. The Acceleration calculates the velocity divided by time, while the Velocity calculates the displacement divided by time.

Example Usage: Velocity and Angular Velocity are necessary for throwing and spinning objects into the air.

Get Touch Controller Acceleration

Determines the acceleration of the controller.

Touch Controller: The touch controller required to obtain amount of acceleration.
Controller Acceleration: The local Vector3 value of the acceleration.

Get Touch Controller Angular Acceleration

Determines the current angular acceleration of the controller.

Touch Controller: The touch controller required to obtain the amount of angular acceleration.
Controller Angular Acceleration: The local Vector3 value of the angular acceleration.

TIP: Acceleration receives a higher value compared to Velocity, and is best used to determine larger changes over time.

Example Usage: The damage from a swinging a bat or a sword may yield better results from Acceleration compared to Velocity based on a higher accuracy in value.

Get Touch Controller Position

Determines the current local position of the controller.

Touch Controller: The controller required to obtain the local position.
Controller Position: Vector3 based on local position.

Get Touch Controller Rotation

Determines the current local rotation of the controller.

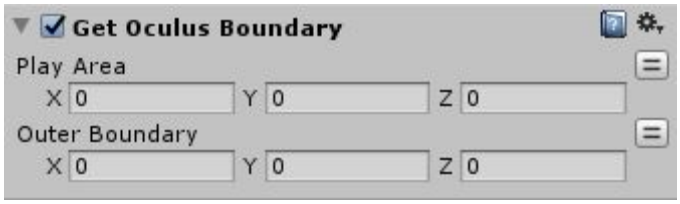
Touch Controller: The controller required to obtain the local rotation.
Controller Position: Vector3 based on local rotation.

Get Oculus Recenter

Recenters the Oculus headset when the state with this action is activated.

Get Oculus Boundary

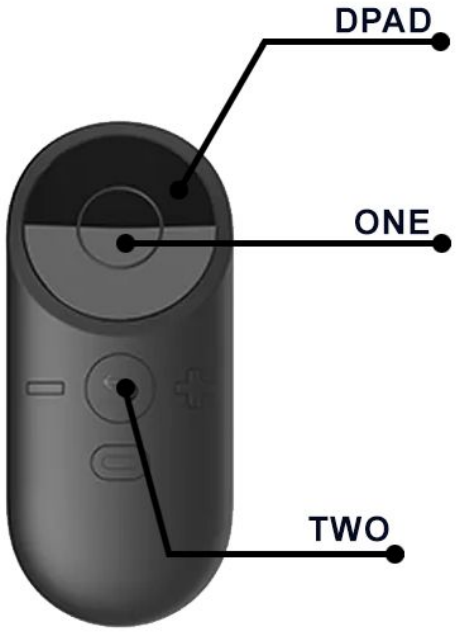
Gets the size of the Play Area and the Outer Boundary set to a Vector3.



Play Area: Gets the Vector3 size of the players play area.
Outer Boundary: Gets the Vector3 size of the current outer boundary.

Example Usage: Knowing the Boundary is important to know how large or short the play area should be depending on the player's space.

Get Rift Remote Button



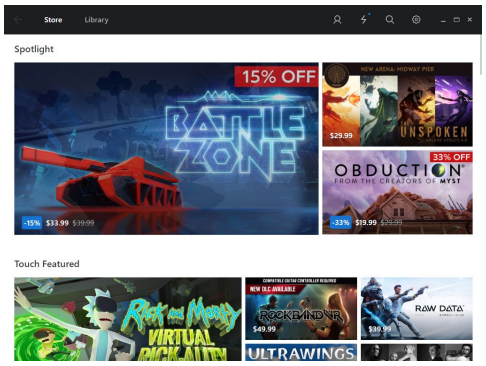
The Get Rift Remote Button action can be used to send an event based on a button press. The **Oculus Button, along with the Plus and Minus Button** are reserved and cannot be configured in the API and can not be currently used.

Button: Select the button to trigger the event.
Button Type: Select between type to trigger an event.
Send Event: Event to send when the specified Button Type is used.
Store Result: Set a bool to true when Button Type is used.

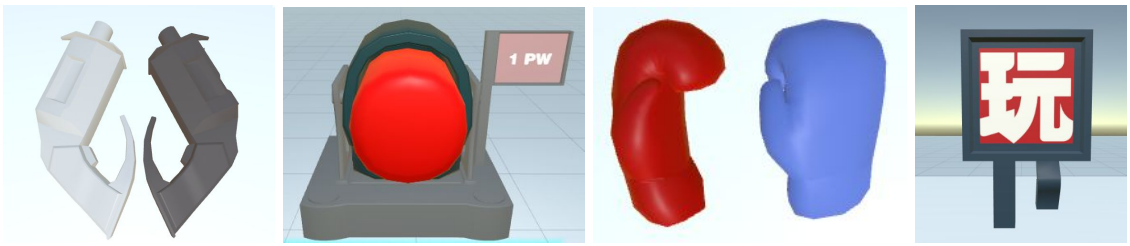
Example Usage:
Dpad is a great button type when selecting options or in a menu for a more passive selection.

Demo Scene

When playing the scene, make sure you have Oculus running (Oculus store should be active). For the Touch Controllers, also make sure you have the headset on when testing the controllers.



In the demo scene there are examples using most of the actions with detailed FSM's and states as well as 4 custom made objects.



*Objects are subject to a Creative Commons License.

- The scene demonstrates the power of the actions with the Get Touch Trigger Axis used to grab and switch between controllers.
- Get Touch Controller Velocity to determine the speed of impact when using the boxing gloves on the machine.
- Get Touch Touchpad and Get Touch Thumbstick Axis to switch the material color of the sphere between red and blue using the Y axis.
- Get Touch Trigger Axis to reset the position of the sphere.
- Get Touch Controller Vibration to set the haptic feedback when firing the weapon, using the punching machine and switching between controllers.
- The scene also features targets to shoot at when the gun is available and a physics sphere to grab and throw.
- Using Get Touch Near Touchpad to point at the little robot and Get Touch Thumbstick Axis for movement.

*Oculus Touch Controller Model is part of the SteamVR folder and package.