TOOLKIT

Updated for Steam VR 2.0 This package is based on the Steam VR API and uses its scripts and functions for the actions to work. A basic understanding is required to use SteamVR in Unity but this documentation and the scene provided will make this easy to understand.

Table of Contents:

This package contains: A demo scene containing most of the actions to show them in use. Actions necessary to use SteamVR in playmaker. CC License models, including a gun.obj, bow.fbx arrow.fbx, boxingGloves.obj, and a punchingMachine.obj. Documentation listing the features and functions in this package. Extras folder.
**Any questions or comments please contact [*frametaleinfo@gmail.com*]**

Understanding SteamVR 2.0
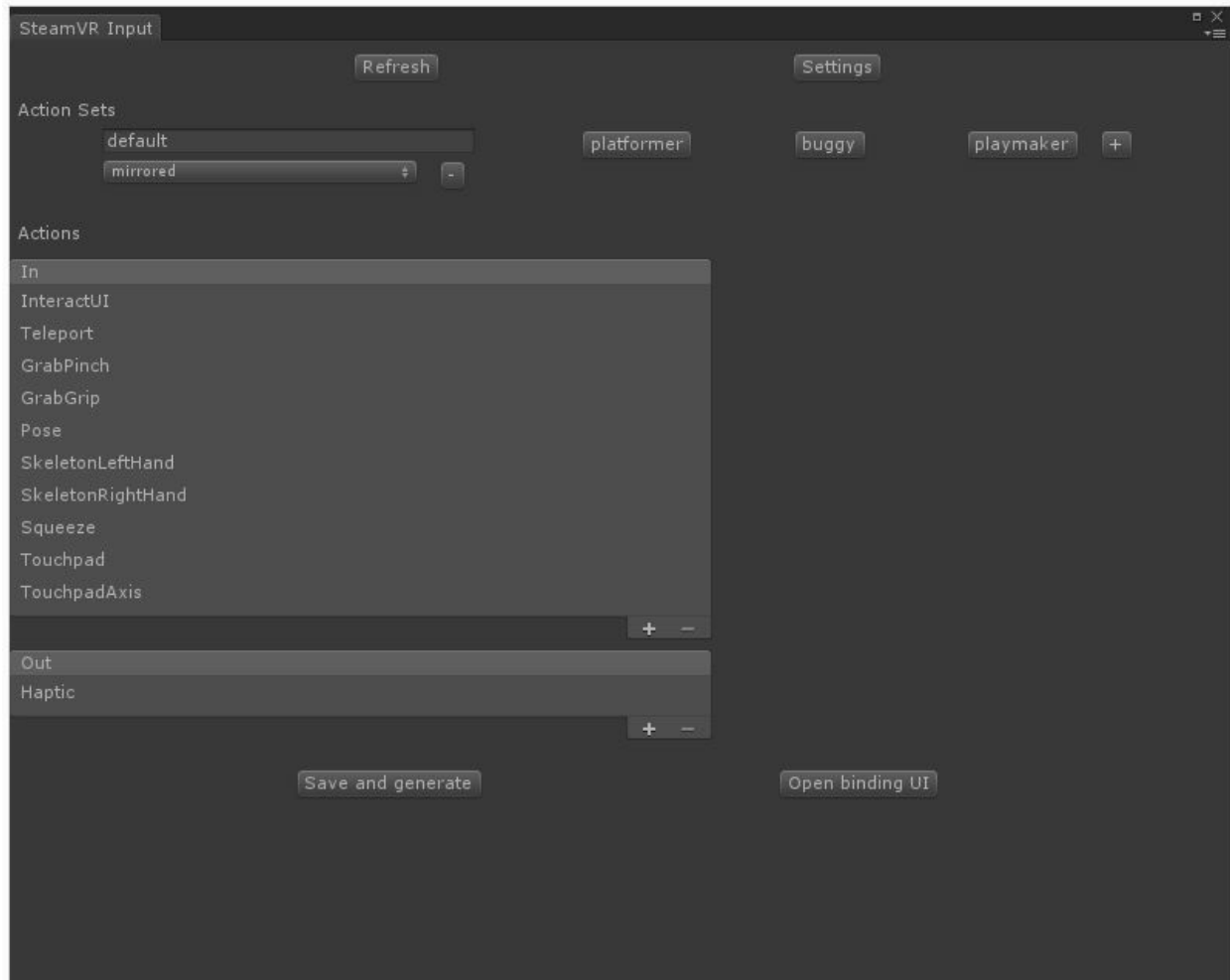
Steam VR 2.0 replaces version 1 Playmaker actions and introduces the default way to use SteamVR actions with the SteamVR Input system *(found under window)*. Inside the Input system are Actions Sets and Actions In and Out. These Actions are necessary when using the playmaker actions. *It is possible to use the Playmaker actions with any action sets.*

When creating a new action, there are different options that will be available under Action Details. The Full Action Path, Name, Type, Required, Localization, and Localized Strings. There are also currently **6** different types of Action types that may be used.
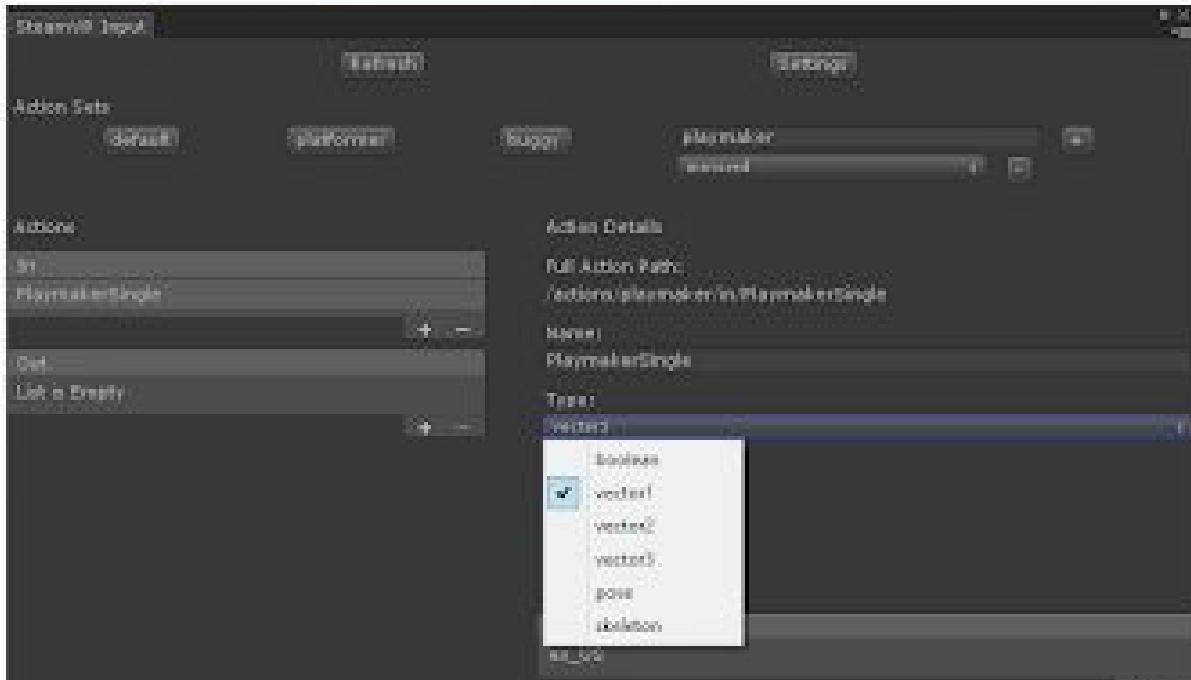
The 6 different types are Boolean, Vector 1 (Single), Vector 2, Vector 3, Pose and Skeleton. The playmaker actions are based on these different types.

When all actions are created, **Save and Generate** will create the Actions into the Full Action Path. After the actions are created, the actions must be set to each controller in the **Open Binding UI.**

Binding the actions are necessary to ensure that the actions created are activated. The binding also works with different types of controllers, including Oculus. To bind an action, select the plus icon on the controller type *i.e. Grip,* choose the action *i.e. Button* and hit the check mark in the lower left of the controller type. To delete, simply press the trash bin icon on the right of the controller type window.

Note: In the current version, if your Action Sets name has a starting capital letter, it will not appear in the binding UI.

Once the SteamVR actions are binded you may add the [Camera Rig] prefab into your scene to use the headset and controllers.

The Camera Rig contains the [play area](#) script. The play area script determines the playspace and can be set to *calibrated* or fixed (between 3 options).

When played, the [SteamVR] prefab is automatically inserted into the scene which adds the scripts required to use .
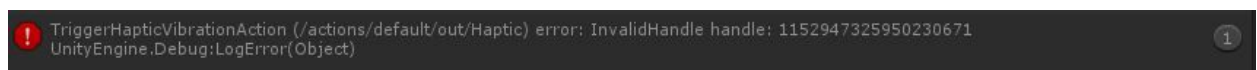

If you receive an error such as *The given key was not present in the dictionary.*

It may mean that the [SteamVR] scripts are not being loaded. Which requires either placing [Camera Rig] and [SteamVR] or just the [Camera Rig].

If you receive an error such as *InvalidHandle*

It may mean that SteamVR has not loaded yet in your scene (which usually takes 0.1 of a second).

Understanding the Prefabs and Actions

In this documentation, we will be using the HTC Vive for our examples. Inside the [CameraRig] are the 2 controller objects as well as a headset. In the interaction system, you may also find the [Player] prefab that contains the controller models and tools but this is not necessary to run the toolkit.

SteamVR 2.0 now uses Unity Camera to control the headset which contains the option to choose the **Target Display** and the **Target Eye**. The Target Display allows you to set which monitor or headset you which to project the scene, while the Target Eye allows you to set which lens to project the scene.

The following information in this documentation will cover the actual actions found in the toolkit. It is important to note that these actions may be placed anywhere in the scene and can control either controller by selecting the device. The only requirement is that the right SteamVR Action is selected from the Input system.
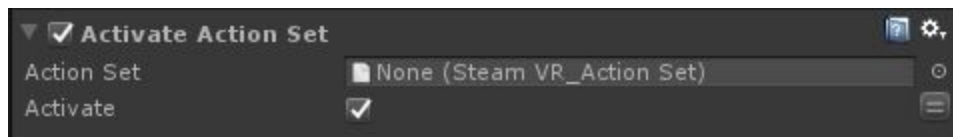
In most actions, there will be different types to choose from. The usual types are between the current state and last state. The *last* state returns a value based on the previous update, while the *current* state returns a value based on the most recent update.

Activate Action Set

Aside from the *default*, all action sets are set to Inactive. You can see an example of all actions and their current value with the *SteamVR Input Live View* under *window*.

The Activate Action Set can be used to select an action set and activate it based on the bool true.
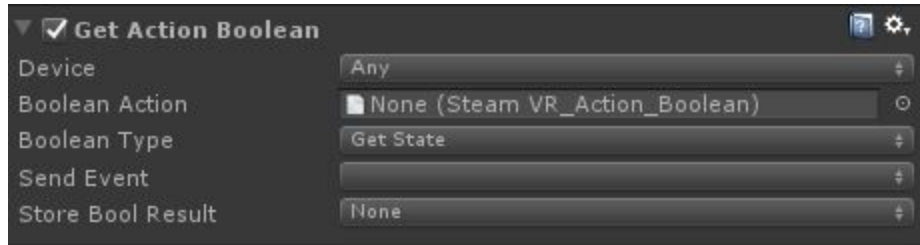

Action Set: Drag or click on the circle to the right of the field to insert the action set created in the Input system.
Activate: A boolean used to either Activate or Deactivate an action set.

Get Action Boolean

The Get Action Boolean can be used to send an event based on a Boolean Type. The Boolean action can be used for any button on a controller. It can also treat triggers as buttons as well based on the binding.

Device: You may select between Any, Left Hand and Right Hand.
Boolean Action: Drag or click on the circle to the right of the field to insert the binded boolean action created in the Input system.
Boolean Type: Choose between which Boolean type to trigger.
Send Event: Sends an event based on the Boolean Type.
Store Bool Result: Will set to true or false depending on the Boolean Type and controller input.

Note: There are currently six different types of Booleans to choose from.
- Get State
- Get State Down
- Get State Up
- Get Last State
- Get Last State Down
- Get Last State Up

Get State is similar to the last version of Steam VR with Get Press. It returns true or false depending on the current state of the button press.

Get Action Pose

The Get Action Pose can be used to get the Velocity, Angular Velocity, Position or Rotation of any Device.
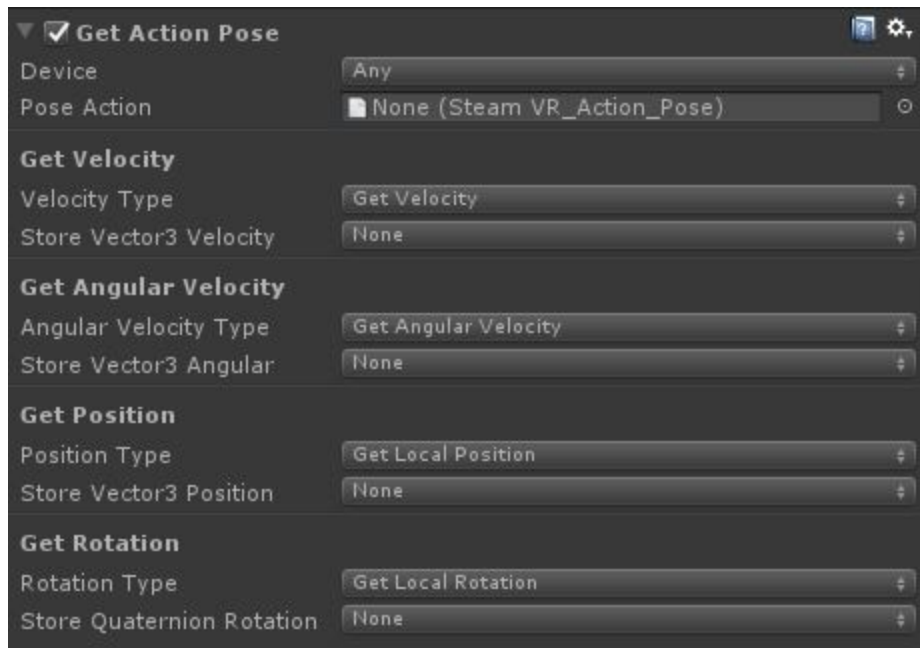
Device: You may select between Any, Left Hand and Right Hand.

Pose Action: Drag or click on the circle to the right of the field to insert the binded Pose action created in the Input system.

Velocity Type: Choose between Get Velocity and Get Last State Velocity type to trigger.

Store Vector3 Velocity: Sets the Vector 3 Velocity based on type.

Angular Velocity Type: Choose between Get Angular Velocity and Get Last State Angular Velocity type to trigger.

Store Vector3 Angular: Sets the Vector 3 Angular Velocity based on type.

Position Type: Choose between the Local Position or Last Local Position.

Store Vector3 Position: Sets the Vector 3 position based on type.

**Rotation Type:** Choose between the Local Rotation and Last Local Rotation.

Store Quaternion Rotation: Sets the Quaternion of the rotation based on type.

All the actions have two options between the current state and the last state.

Get Action Single

The Get Action Single (Also known as Get Action Vector 1) can be used to get a float from the controller. This is typically used for the triggers on the controllers.
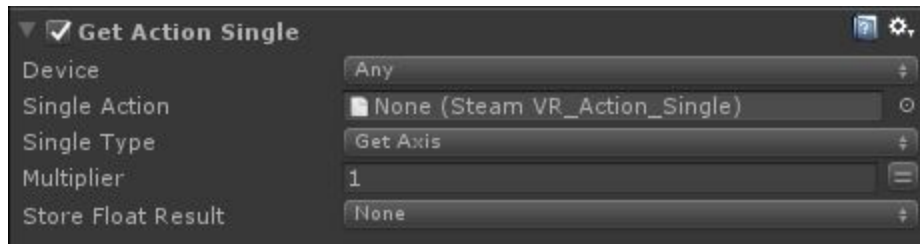
Device: You may select between Any, Left Hand and Right Hand.
Single Action: Drag or click on the circle to the right of the field to insert the binded Single action created in the Input system.
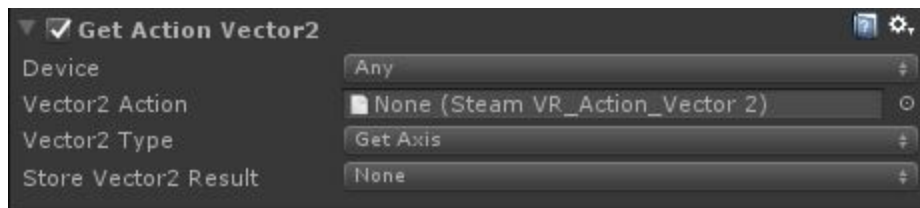**Single Type:** Choose between Get Axis, Get Axis Delta, Get Last Axis, and Get Last Axis Delta.
Multiplier: Adds a multiplier to the float.
Store Float Result: Sets the float variable based on controller input.

Get Action Vector 2

The Get Action Vector 2 can be used to get a Vector 2 from the controller. This is typically used for the touchpad or joysticks.

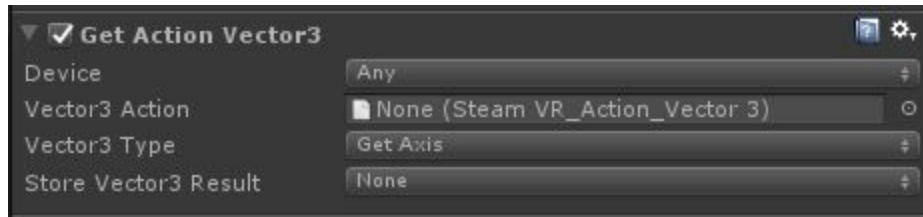**Device:** You may select between Any, Left Hand and Right Hand.
Vector 2 Action: Drag or click on the circle to the right of the field to insert the binded Vector 2 action created in the Input system.
Vector 2 Type: Choose between Get Axis, Get Axis Delta, Get Last Axis, and Get Last Axis Delta.
Store Result: Sets the Vector 2 variable based on controller input.

Get Action Vector 3

The Get Action Vector 3 can be used to get a Vector 3 from the controller.

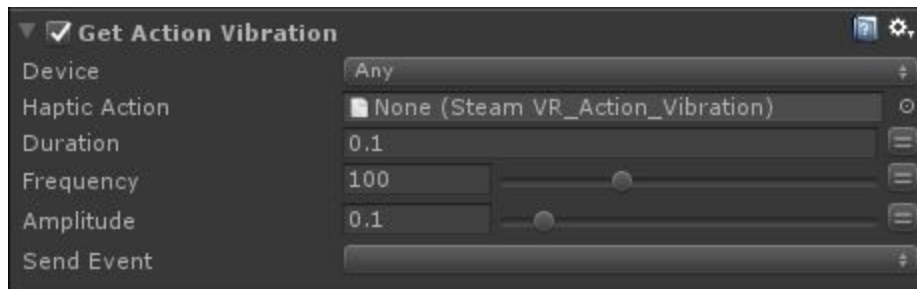**Device:** You may select between Any, Left Hand and Right Hand.
Vector 3 Action: Drag or click on the circle to the right of the field to insert the binded Vector 3 action created in the Input system.
Vector 3 Type: Choose between Get Axis, Get Axis Delta, Get Last Axis, and Get Last Axis Delta.
Store Vector 3 Result: Sets the Vector 3 variable based on controller input.

Get Action Vibration

The Get Menu can be used to send an event based on a Trigger type.

Device: You may select between Any, Left Hand and Right Hand.
Haptic Action: Drag or click on the circle to the right of the field to insert the binded Haptic action created in the Input system.
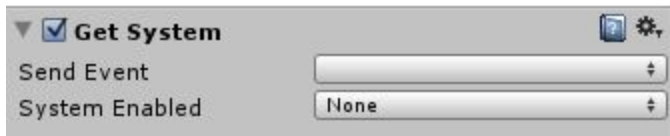**Duration:** The amount of time the haptic action will last.
Frequency: How often the haptic motor should bounce.
Amplitude: How intense the haptic action should be.
Send Event: Event to send when the duration is completed.

Get System

The Get System can send an event or set bool value based on button press.

**Send Event:** Event to send when the System button is used.
System Enabled: Set a bool to true when System button is pressed.

This action uses the script SteamVR_Render (Ref.2) to determine when to send an event or set bool value based on the system button press. This action is useful for setting a pause screen when the system button has been activated.

Get Gaze

Bool is set to true if object is in range between Gaze In Cutoff and Gaze Out Cutoff.
Note: This action looks for the Main Camera. Make sure your camera being used for VR has the tag *Main Camera*.
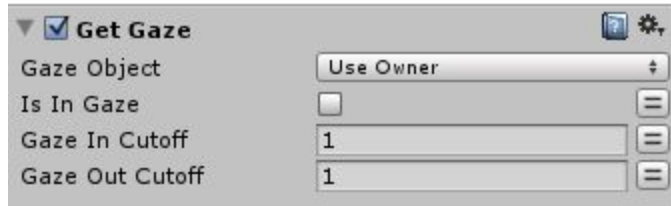
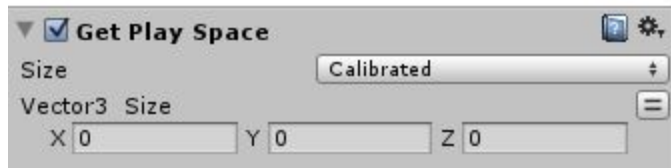**Gaze Object:** The gameobject used to keep in view of headset.
Is in Gaze: Bool to note when gaze is in range.
Gaze in Cutoff: Cutoff closest to gameObject.
Gaze out Cutoff: Cutoff furthest from gameObject.

Get Playspace

Gets the Play Space determined by size, set to a Vector3.

Size: Select size to grab Vector3.
Vector3 Size: Sets Vector3 size determined by the size option.

Size has a selection between:
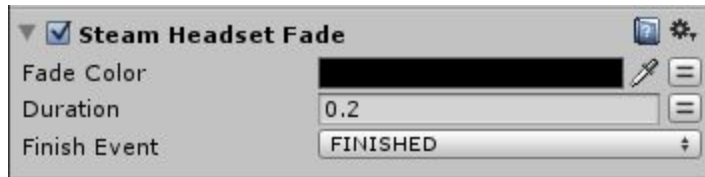-Calibrated (Use to determine players custom playspace.)
-200x150
-300x225
-400x300

Steam Headset Fade

Uses the Steam Fade script to Fade the headset for a determined duration.
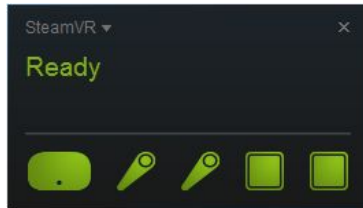*Requires the script SteamVR_Fade to be placed on Camera to work.*

Fade Color: The color used to fade the headset.
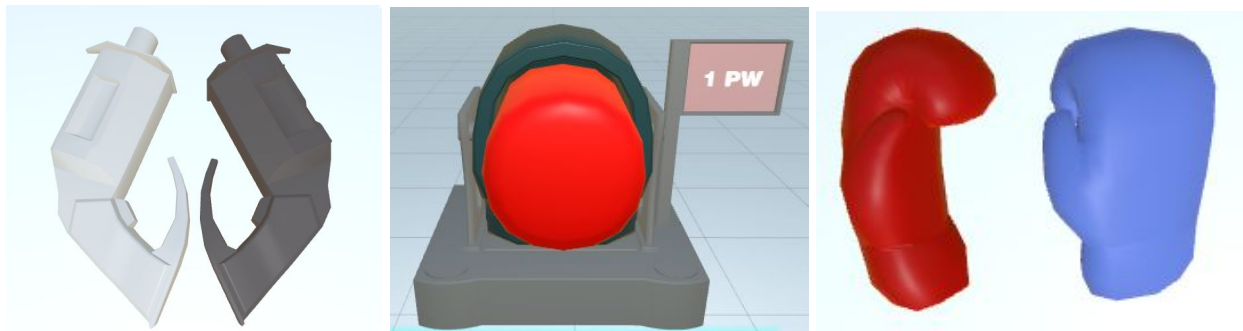**Duration:** The time taken to complete the fade.
Finish Event: Event to send when the fade has completed.

Demo Scene

When playing the scene, make sure you have SteamVR running.

In the demo scene are examples using most of the actions with detailed FSM's and states as well as 3 custom made objects.

1. The Gun.
2. Boxing Machine.
3. Boxing Gloves.

*Objects are subject to a Creative Commons License.

The scene demonstrates the power of the actions with the Get Action Boolean used to grab and switch between controllers.
Get Action Pose to determine the speed of impact when using the boxing gloves on the machine.
Get Vector 2 to switch the material color of the sphere between red and blue using the Y axis.
Get Action Boolean to reset the position of the sphere.
Get Gaze to change the material of the giant sphere when gaze is range.
Get Action 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 when the gun is available, a giant sphere to *gaze* and a physics sphere to grab and throw.
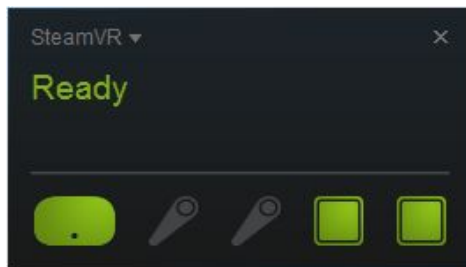
Errors

Errors that you may encounter in this package and how to resolve them.

Index OutOfRangeException: Array index is out of range.
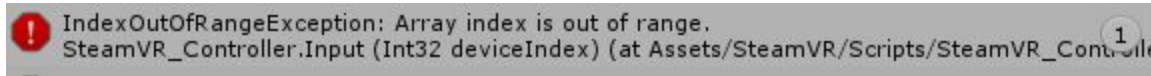
If this error appears in the console, an action or script can not find the controller it is expecting to find. In most cases a controller is not found in the steamVR status.

The type or namespace name `FsmStateAction' could not be found.

If this error appears, Playmaker can not be found in the assets folder.

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