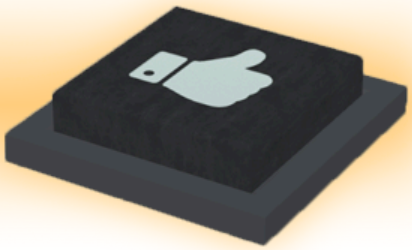
Immersive Interactions

by Janooba



v0.4.x

Welcome

Thank you for downloading Immersive Interactions. This project aims to provide the most comprehensive and physically accurate buttons and switches on the VRChat market.

If you'd like to test out the system, join the following world: Immersive Interactions

Contents

Quick Start

1. Install the ImmersiveInteractions package.
2. Drag the PlayerSkeletonInfo prefab located in **Packages/Immersive Interactions/Runtime/PlayerSkeletonInfo** into your scene.
    a. Ensure the **Bone Prefab** is set.
3. Drag one of the button prefabs located in **Packages/Immersive Interactions/Runtime/Button Prefabs** into your scene.
4. Once positioned where you'd like, select the child object with a **Pressable_Button** or **Flippable_Switch** component on it.
5. Click **Find Nearby Colliders To Ignore** under **// RUNTIME & DETECTION**. This will make sure the button doesn't get stuck in the wall or objects behind it. If this still happens, you may have to manually add the collider to the **Ignored Colliders** list.
6. Scroll to the **// EVENTS** section and add something to the list of receivers.
7. Toggle on one of the Send toggles and note the events that will be sent below.
8. Change the event name to match your own script's Events, or add a public event to your own Udon Behaviour that matches one of the events sent.

Reading The Gizmo

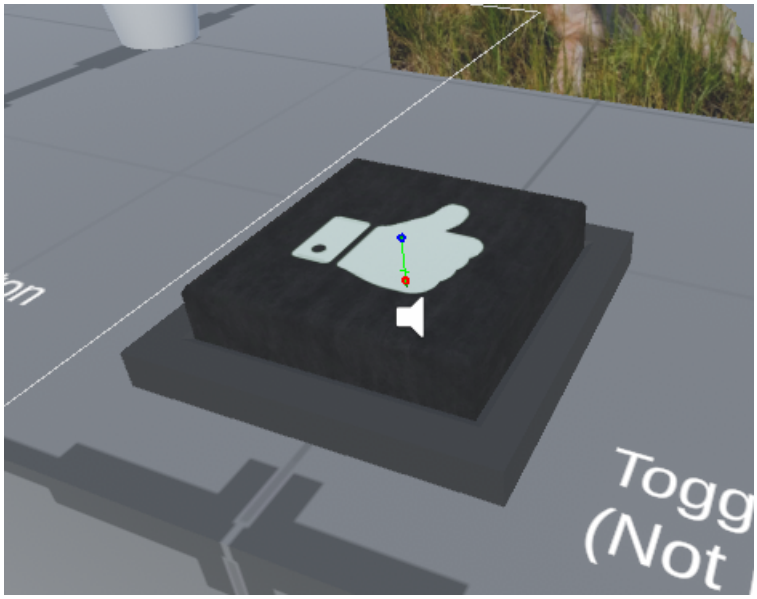
This system has an in-scene gizmo to help you set up your buttons.

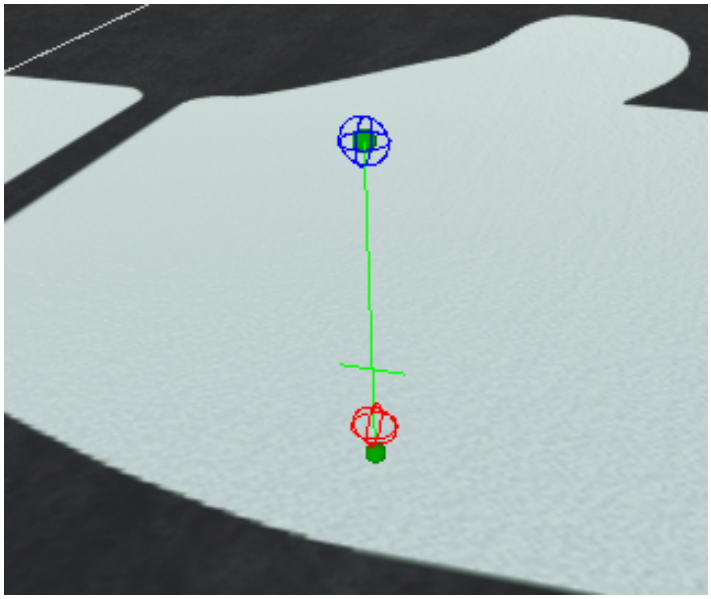
Green vertical line: The total travel distance.

Green Cross: The "toggled" stop point.

Blue wire sphere: The current position of the button.

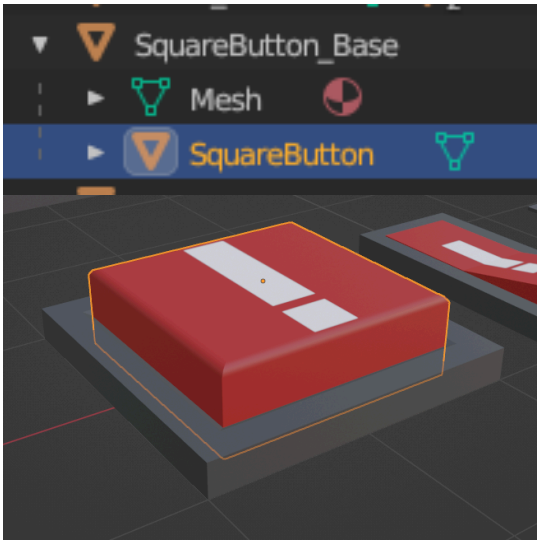
Red wire sphere: The "trigger" point, beyond which, the button will trigger.





Making a Button from Scratch

Step 1 - Make your button/switch model



Use your modeling software of choice to create the shape for your button. For buttons, the button itself (the part that moves) should have its origin at the very top surface of the button and it should be a child of a base object.

For switches, the origin should be at the pivot point. It will pivot around the X axis.

Position the button and switch in its resting or out/off position.

Keep your UVs in mind if you want to use the Texture swapping module.

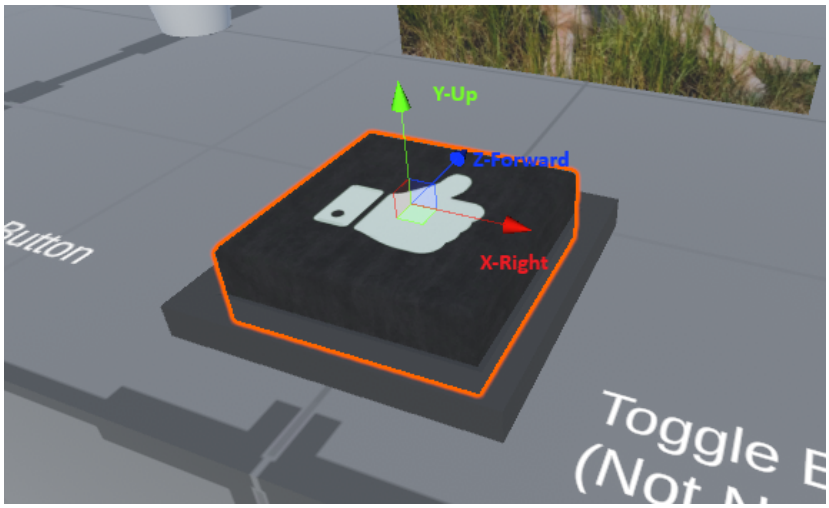
Step 2 - Setting up the prefab

Place your model in the world when you are ready. It should be a parent object, with the button as a child.

Select your child button object, and add a **Pressable_Button** or **Flippable_Switch** component. This will automatically add a **Rigidbody** if needed, and set **Is Kinematic** to true.

The bare minimum needed is a **Collider** on the button itself and set it to **Is Trigger** (This is important for the system to work correctly). Primitive colliders are always preferred over mesh colliders for performance reasons. You can create children with colliders if you have a more complex button shape too, but if you do this, you MUST assign a fallback collider.

Set up the **Button Axis** according to how you modeled your button. The axis should point away from the button face in local space. So if **Y-Up** is your button facing, enter **X:0 Y:1 Z:0**.



If you want to add a collider to your button's parent/border, either make sure it doesn't cover the button completely, even when the button is pressed in, or add a fallback collider. This will ensure the button doesn't get stuck, and can be interacted with properly by anyone, VR or not. You will also need to add any parent colliders to the **Ignored Colliders** list, so that the button doesn't get "stuck" in it.

Drag your button into the Project panel to create a prefab. That's it! Refer to the Quick Start guide to see how to actually use this button.

Step 3 - (Optional) Setting Up Modules

> Audio

To give your button audio, all you need is an **Audio Source**, and clips for either press and release, or both. I personally like to put the **Audio Source** on a child object of the button base, for organization's sake. Assign the **Audio Source** to your **Button Source** field under Audio.

> Tint

The button system can change an arbitrary color on whatever renderer you choose by targeting a shader keyword. This will require you to know the shader keyword that needs to change in advance. You can often find this by enabling **Debug Mode** in the inspector while a material is selected, and looking through the values. By default, a lot of shaders use "**_Color**", "**_Tint**", or "**_EmissionColor**" for emission.

All you need to do is assign the target renderer that will have the material you want to change in the **Tint Renderer** field, and set the **Tint Shader Keyword** to the property you want to set.

I've included a Standard Shader called **Janooba/Button Standard** that should work for most people. The default inspector values will work for it.

> Texture

This option can be used to change a texture, such as an icon, depending on the button state. Much like the **Tint** module, This will require you to know the shader keyword needed to change the appropriate texture. Most shaders use "**_MainTex**" as their main diffuse texture.

All you need to do is assign the target renderer that will have the material you want to change in the **Texture Renderer** field, and set the **Texture Shader Keyword** to the property you want to set.

I've included a Standard Shader called **Janooba/Button Standard** that should work for most people. The default inspector values will work for it.

> Animation

Buttons are able to control an arbitrary **Animator** parameter with this module. This is useful for buttons that have a complex animation while being pressed. This system will set a float parameter to a value between 0 and 1, depending on how far the button is pressed. See the Button Prefab **Bumper Button** for an example of how this system can be set up. Typically, you will create a linear animation of the button being pressed, then assign the node's **Motion Time** to the controlled parameter.

> Handle (Switches only)

This module allows players to grab a handle and pull it around to rotate the switch. Useful for levers. To use, create a new trigger object with a **VRC Pickup** and a **Lever_Handle** component alongside the switch/lever and position it where you want the grab point to be. Assign the switch as **Lever** in the inspector, and then go back to your switch and assign the **Lever_Handle** to this module's **Handle** component.

The **Force Use Handle For Non VR** option is useful if you don't want desktop players to be able to just "click" the lever to activate it.

> Haptics

Not much to say about this. Default values work pretty well, but may need more tuning if you'd like a more clicky feel. Tweaking this is all about feel, so there isn't too much that can be explained.

Debugging Buttons

One of the quickest ways to debug buttons that may not be working as expected is to add the **DebugButtonCanvas** prefab to your scene, and assign the button to its **Button_Logger** script.

This will allow you to see what's going on in the button internally. Otherwise, refer to the Troubleshooting section for common hangups.

If still, you cannot seem to find a cause for the button malfunction, please create an issue on my github here.

Component Reference

Player Skeleton Info

Field	Type	Description
Bone Prefab	GameObject	Should be set for you. The prefab that will be used for tracked bones. Don't change this unless you really know what you're doing.
Force Fallback	Bool	Will force all buttons to use the basic VRChat interact system instead of tracking the skeleton. Will not disable physics interactions. Can be changed at runtime. Useful for giving the player performance options.
Include Head	Bool	Will add a tracked sphere for the head.
Include Hips	Bool	Will add a tracked sphere for the hips.
Include Feet	Bool	Will add a tracked capsule for feet.
Include Full Hands	Bool	Will track all fingers and palms. If this is off, only the index finger will be tracked.
Show Debug	Bool	Shows the bone colliders as blue transparent shapes.
Debug Status	TextMeshPro UGUI	An optional TMP text component to display debug information.
All Bones	Array (PlayerBone)	For debug use. Will be filled with all created and tracked player bones.

Pressable Button

Field	Type	Description
// RUNTIME & DETECTION		
Is Locked	Bool	If the button is locked, it cannot be moved or pressed.
Detect Players	Bool	Whether or not to detect the player skeleton.
Detect Rigidbodies	Bool	Whether or not dynamic objects with rigidbodies are detected.
Detect Static	Bool	Whether or not static objects (like the world) are detected. Useful if your button is on a dynamic object.
Find Nearby Colliders To Ignore	Button	Will try to automatically find objects that may obstruct the button as it's pressed in, so they can be ignored.
Clean Up Nulls	Button	Will remove null entries from the Ignored Colliders list.
Ignored Colliders	Array (Colliders)	All colliders in here will be completely ignored.

// GENERAL SETTINGS		
Button Axis	Vector3	Which direction does the button face? Points away from the button.
Button Thickness	Float	How far the button travels before bottoming out.
Trigger Zone	Float (Slider)	Percentage of the thickness the button needs to travel before being considered “pressed”. Where 0 is fully unpressed, and 1 is fully pressed.
Cooldown	Float	Minimum amount of time passed, in seconds, before the button will register a new press.
Return Rate	Float	How fast the button will return to its unpressed position.
Limit Backpressing	Bool	Limits how quickly a button can move in a frame so that it doesn’t snap down from presses coming from behind it.
Is Toggle Button	Bool	Will the button stay pressed until pressed again.
- Start Toggled On	Bool	Should the button be on when initialized?
- Toggle In Position	Float	The position to set as the new “top” in travel distance percentage when pressed in. The button will rest at this point until toggled off.
Fallback Collider	Collider	Optional. Will be used for desktop users and fallback if the normal button collider isn’t accessible. This can happen if the button is pushed inside of another collider.
Fallback Press Time	Float	How long to stay pressed when using a fallback/non vr press before popping back up.
// EVENTS		
Send Pressed Event	Bool	This event will be sent to receivers when pressed. Will have “_On” or “_Off” appended if the button is a toggle button.
Send Released Event	Bool	This event will be sent to receivers when released. Will have “_On” or “_Off” appended if the button is a toggle button.
Force Send Stateless	Bool	This forces the system to send the pressed and released events without “_On” or “_Off” as well.
Udon Receivers	Array (GameObject)	A list of Udon Behaviours to send events to.
// MODULES		
Network Sync		Enables network syncing for this button.
Master Only	Bool	If on, the button will only be interactable by the instance master. Even by physics.
Transfer Receiver Ownership	Bool	Attempts to transfer ownership to the person pushing, even with physics. In most cases you want this on, as it may lead to players that don’t own the button not being able to press it.
Disable When Network Clogged	Bool	In certain circumstances, the network may get clogged by too much traffic (which may or may not be related to buttons). In order to help the network recover, this option will lock buttons so they cannot be pressed until the network has recovered.

Audio		Allows this button to make press and release sounds.
Button Source	Audio Source	The audio source that button sounds will come from.
Press Clip	Audio Clip	The sound to play when the button has been pressed.
Release Clip	Audio Clip	The sound to play when the button has been released.
Tint		Changes a renderer's tint depending on button state. Will target all materials on a renderer with a MaterialPropertyBlock, and set a shader keyword.
Tint Renderer	Renderer	The renderer to target for tinting.
Tint Shader Keyword	String	The shader keyword on the materials that should be changed.
Off Tint	Color	Color for the default "out" state of the button.
In Tint	Color	Color displayed while the button is pressed in, but not released.
On Tint	Color	Color displayed if the button is a toggle button, and it is currently toggled on.
Locked Tint	Color	Color displayed when the button is locked.
Texture		Changes a renderer's texture depending on button state. Will target all materials on a renderer with a MaterialPropertyBlock, and set a shader keyword.
Texture Renderer	Renderer	The renderer to target for texture swapping.
Texture Shader Keyword	String	The shader keyword on the materials that should be changed.
Off Texture	Texture 2D	Texture for the default "out" state of the button.
In Texture	Texture 2D	Texture displayed while the button is pressed in, but not released.
On Tint	Texture 2D	Texture displayed if the button is a toggle button, and it is currently toggled on.
Locked Texture	Texture 2D	Texture displayed when the button is locked.
Animation		Controls an animator's parameter with the press progress. Useful if you want a button to have a unique animation for pressing.
Animator	Animator	The animator that will be controlled by this button.
Progress Parameter	String	The parameter that will be set to the button progress (0 to 1).
Haptics		Trigger haptic feedback for users pressing this button.
Haptics Duration	Float	How long to vibrate the controller.
Haptics Amplitude	Float (Slider)	How strong the vibration should be.
Haptics Frequency	Float	How fast the vibration should be in Hz.

Flippable Switch

Field	Type	Description
// RUNTIME & DETECTION (See Pressable Button)		
// GENERAL SETTINGS		
Max Rotation	Float	How far to rotate the switch in degrees.
Trigger Zone	Float (Slider)	Percentage of the rotation the button needs to travel before being considered “pressed”. Where 0 is fully unpressed, and 1 is fully pressed.
Cooldown	Float	Minimum amount of time passed, in seconds, before the button will register a new press.
Return Rate	Float	How fast the button will return to its unpressed position.
Is Toggle Button	Bool	Will the button stay pressed until pressed again.
- Start Toggled On	Bool	Should the button be on when initialized?
Fallback Collider	Collider	Optional. Will be used for desktop users and fallback if the normal button collider isn’t accessible. This can happen if the button is pushed inside of another collider.
Fallback Press Time	Float	How long to stay pressed when using a fallback/non vr press before popping back up.
// EVENTS (See Pressable Button)		
// MODULES (See Pressable Button)		
Handle		Attach a handle to this switch/lever for easy pulling
Handle	Lever_Handle	The lever handle object to use.
Force Use Handle For Non VR	Bool	If true, Non VR players will not be able to click this switch and must use the handle.

Lever Handle

Field	Type	Description
References		
Lever	Flippable_Switch	The switch to control with this lever.
Pickup	VRC Pickup	This handle’s Pickup.

Public Scripting API

Method / Field	Description
CurrentUnitProgress	Field. A 0 to 1 value that represents how far pressed this button is.
IsSleeping	Field. Whether this button is sleeping (Not being updated).
IsTriggered	Field. Will be true while the button is beyond the trigger threshold.
IsToggled	Field. Will be true when the button is toggled, if this button is a toggle button.
Pressed / Pressed_On / Pressed_Off	Methods. Can be used to manually press this button as if a person pressed it.
Lock_On / Lock_Off	Methods. Locks or Unlocks the button.
Wake / Sleep	Methods. Will force the button awake or to sleep. May be useful if you are modifying things like its progress outside of the API.
NonVR_Interact	Method. Can be used to manually press this button as if a NonVR user pressed it.

Troubleshooting

Issue	Solution
My button gets stuck when I press it.	Ensure the surfaces behind the button are added to the collider ignore list.
My buttons are sometimes not registering presses or moving.	Make sure the network isn’t clogged! This can stop buttons from working correctly.