DOOZY UI COMPLETE UI MANAGEMENT SYSTEM

Owner's Manual

v 2.9

Contents

Introduction

DoozyUI is a complete UI management system for Unity. It manipulates native Unity components and takes full advantage of their intended usage. This assures maximum compatibility with uGUI, best performance and makes the entire system have a predictable behavior. Also, by working only with native components, the system will be compatible with any other asset that uses uGUI correctly.

Easy to use and understand, given the user has some basic knowledge of how Unity's native UI solution (uGUI) works, DoozyUI has flexible components that can be configured in a lot of ways. Functionality and design go hand in hand in order to offer a pleasant user experience (UX) while using the system.

Artists and designers can realize their creative vision without coding by creating blazing fast user interfaces with any animations they can imagine.

Programmers can add a powerful UI management system to their toolbox that can be interfaced with scripts or used alongside Playmaker for a code-free UI solution.

It comes with an UI Animator System that uses math-based tweens creating reliable resolution independent animations and an Orientation Manager that allows anyone to create views for both portrait and landscape modes and not have to worry about which view should be active.

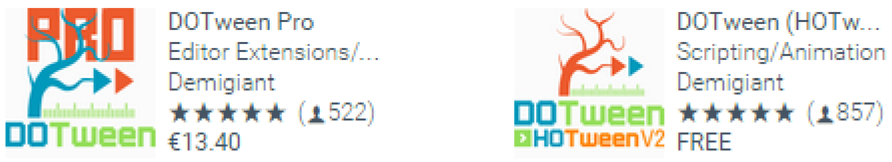
Eases the usage of Particle Systems within the UI layers and has a quick workflow for creating modal windows, named UI Notifications.

It supports all platforms and is extensively optimized to minimize its memory usage. Full C# source code, dedicated support and YouTube tutorials are available to help anyone understand and then master the system.

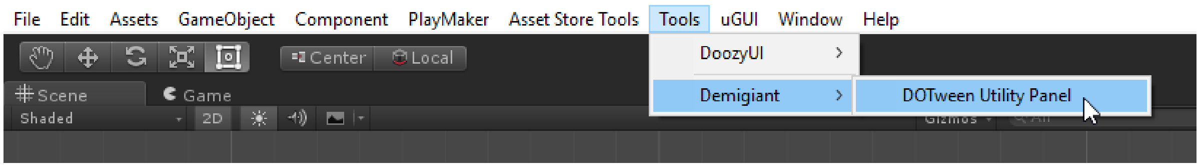
Quick Start

To start using DoozyUI in your project you need to do the following:

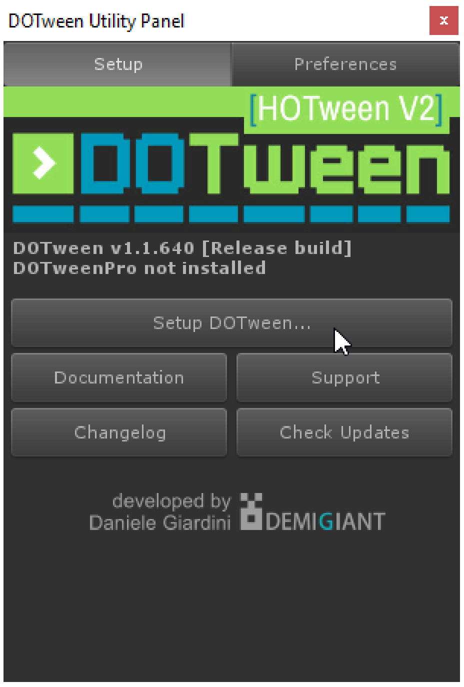
1. Import DOTween or DOTween Pro from the Unity Asset Store



2. Open DOTween Utility Panel



3. Setup DOTween



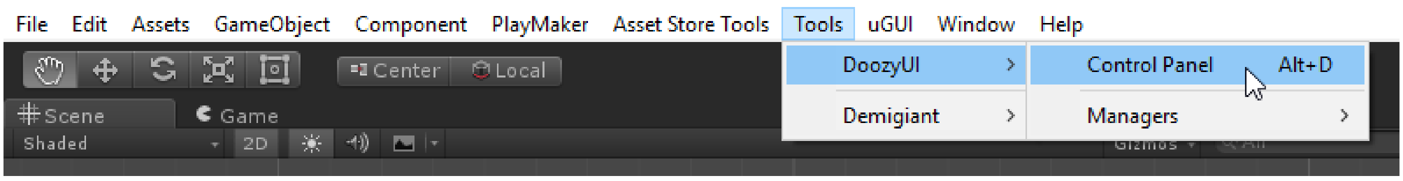
DOTween v1.1.640 [Release build]
DOTweenPro not installed

4. Import DoozyUI from the Unity Asset Store
5. Done! 😉

Control Panel

The Control Panel is the main location from where you can configure the way DoozyUI works, manage all the databases and tweak all the animations.

You can access (open) the Control Panel from the top bar *Tools* → *DoozyUI* → *Control Panel*. Or press Alt+D.



After the Control Panel has been opened, it can be closed from the [X] button or by pressing Alt+X.

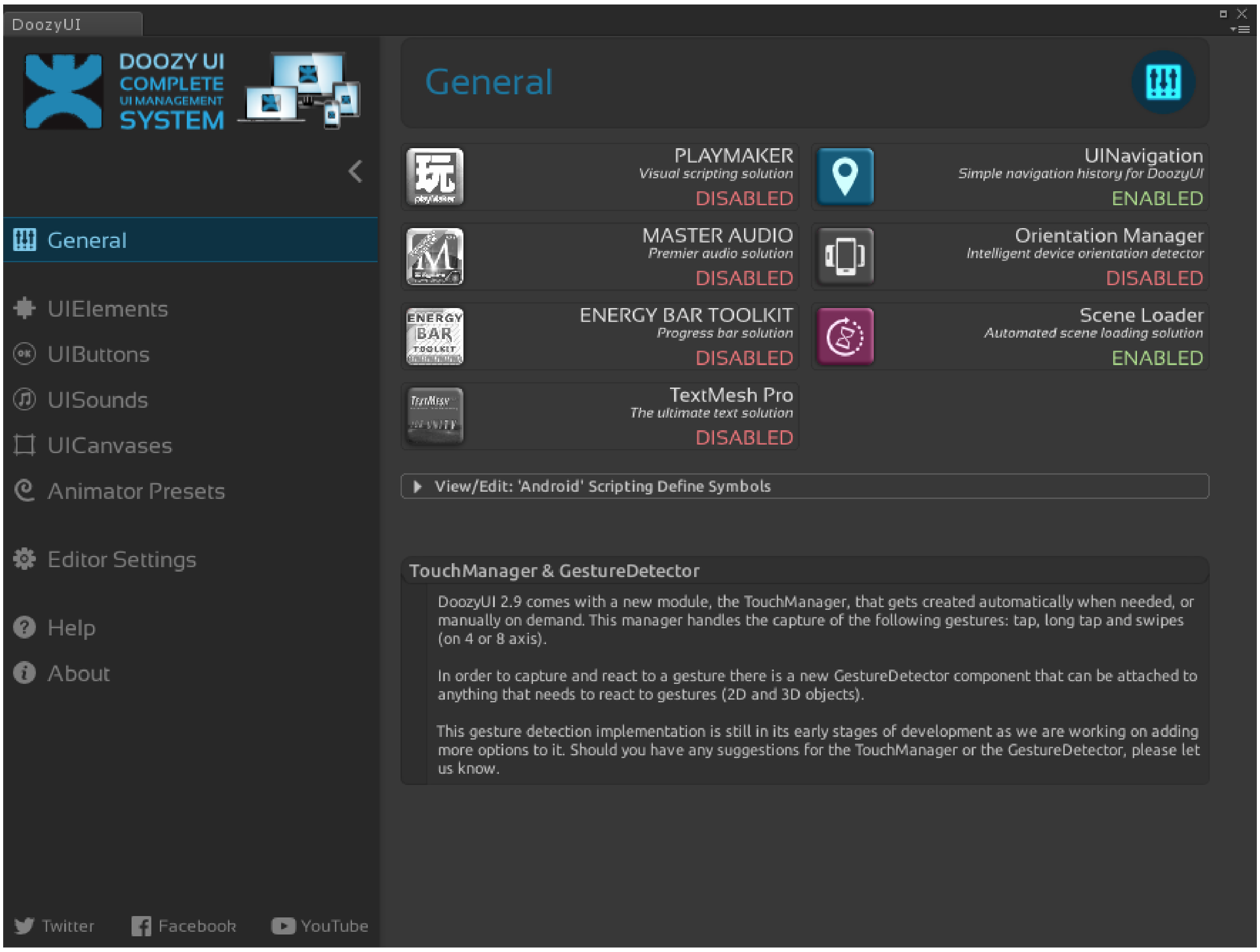
Collapse/Expand tab button names by clicking on the arrow below the logo or by pressing Alt+~

From here you can access the following tabs:

1. General
   - manage Scripting Define Symbols that toggle on/off DoozyUI extra options
   - read news about the system
2. UIElements
   - manage element categories and the element names they contain
3. UIButtons
   - manage button categories and the button names they contain
4. UISounds
   - manage UISounds and their visibility (for UIButtons, UIElements or both)
5. UICanvases
   - manage canvas names
6. Animator Presets
   - tweak UIElement animations: In, Out and Loop
   - tweak UIButton animations: Punch and State
7. Editor Settings
   - enable/disable the Hierarchy Manager that shows icon and/or info about DoozyUI components right in the Hierarchy View *(this feature might slow down the Editor)*
   - set default values for newly created UIElements
   - set default values for newly created UIButtons
   - set default values for newly created UIToggles
   - set default values for newly created UIEffects
8. Help
   - link to the online version of this manual
   - link to the DoozyUI Help Center
   - the support email address
9. About
   - view the installed DoozyUI version
   - read about DoozyUI

General Tab

Alt+1



General

PLAYMAKER
Visual scripting solution
DISABLED

UINavigation
Simple navigation history for DoozyUI
ENABLED

MASTER AUDIO
Premier audio solution
DISABLED

Orientation Manager
Intelligent device orientation detector
DISABLED

ENERGY BAR TOOLKIT
Progress bar solution
DISABLED

Scene Loader
Automated scene loading solution
ENABLED

TextMesh Pro
The ultimate text solution
DISABLED

View/Edit: 'Android' Scripting Define Symbols

TouchManager & GestureDetector

DoozyUI 2.9 comes with a new module, the TouchManager, that gets created automatically when needed, or manually on demand. This manager handles the capture of the following gestures: tap, long tap and swipes (on 4 or 8 axis).

In order to capture and react to a gesture there is a new GestureDetector component that can be attached to anything that needs to react to gestures (2D and 3D objects).

This gesture detection implementation is still in its early stages of development as we are working on adding more options to it. Should you have any suggestions for the TouchManager or the GestureDetector, please let us know.

Toggle support for:

- **Playmaker** (default: disabled)
- **Master Audio** (default: disabled)
- **Energy Bar Toolkit** (default: disabled)
- **TextMesh Pro** (default: disabled)

Toggle options for:

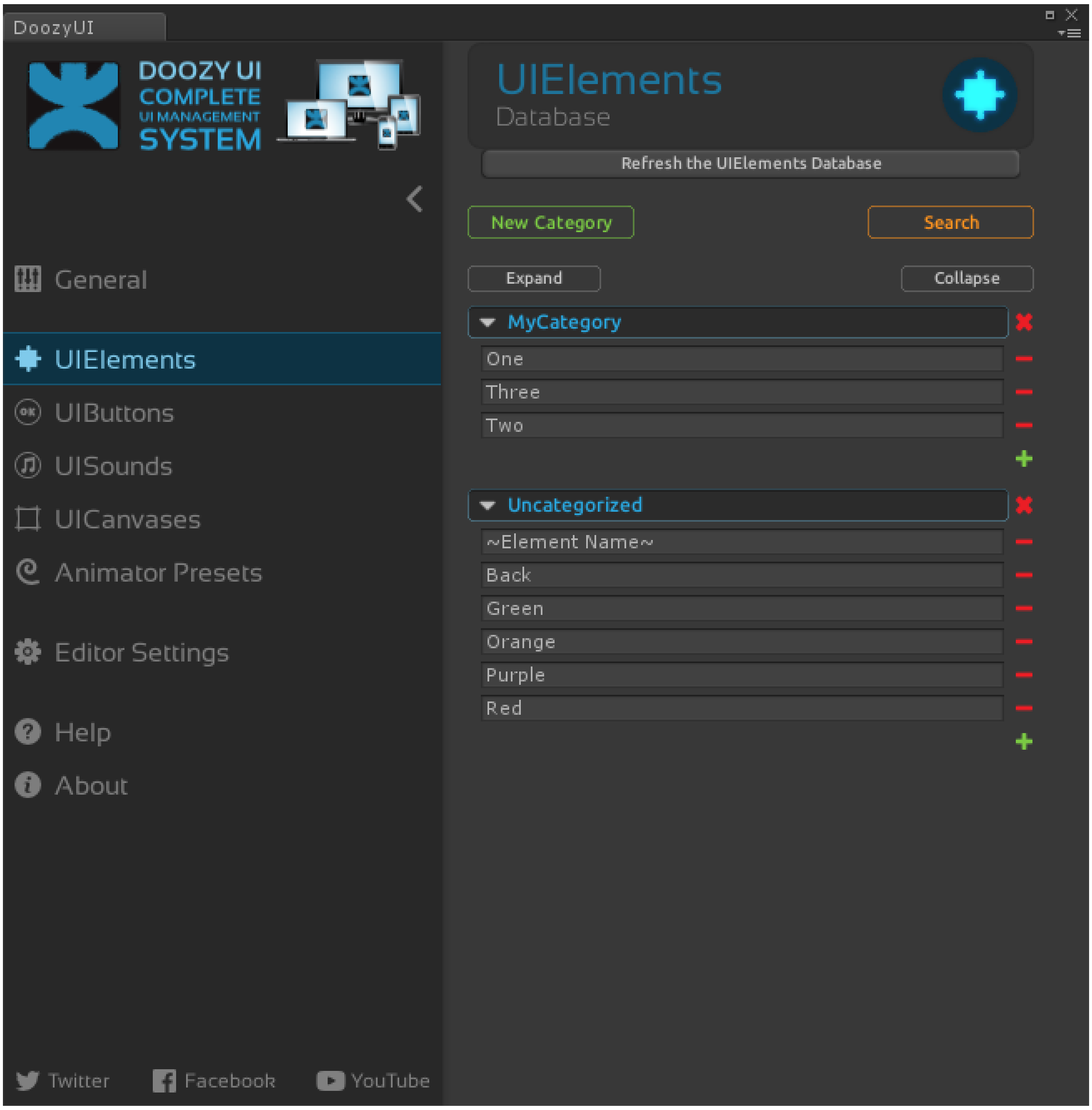
- **UINavigation** (default: enabled)
- **Orientation Manager** (default: disabled)
- **Scene Loader** (default: enabled)

View/Edit Scripting Define Symbols for the currently active platform.

Read news about DoozyUI.

UIElements Tab

Alt+2



Here you can add/remove element categories and add/edit/remove element names.

Refresh the UIElements Database → executes a scan for any new element categories

New Category → create a new element category (Alt+N)

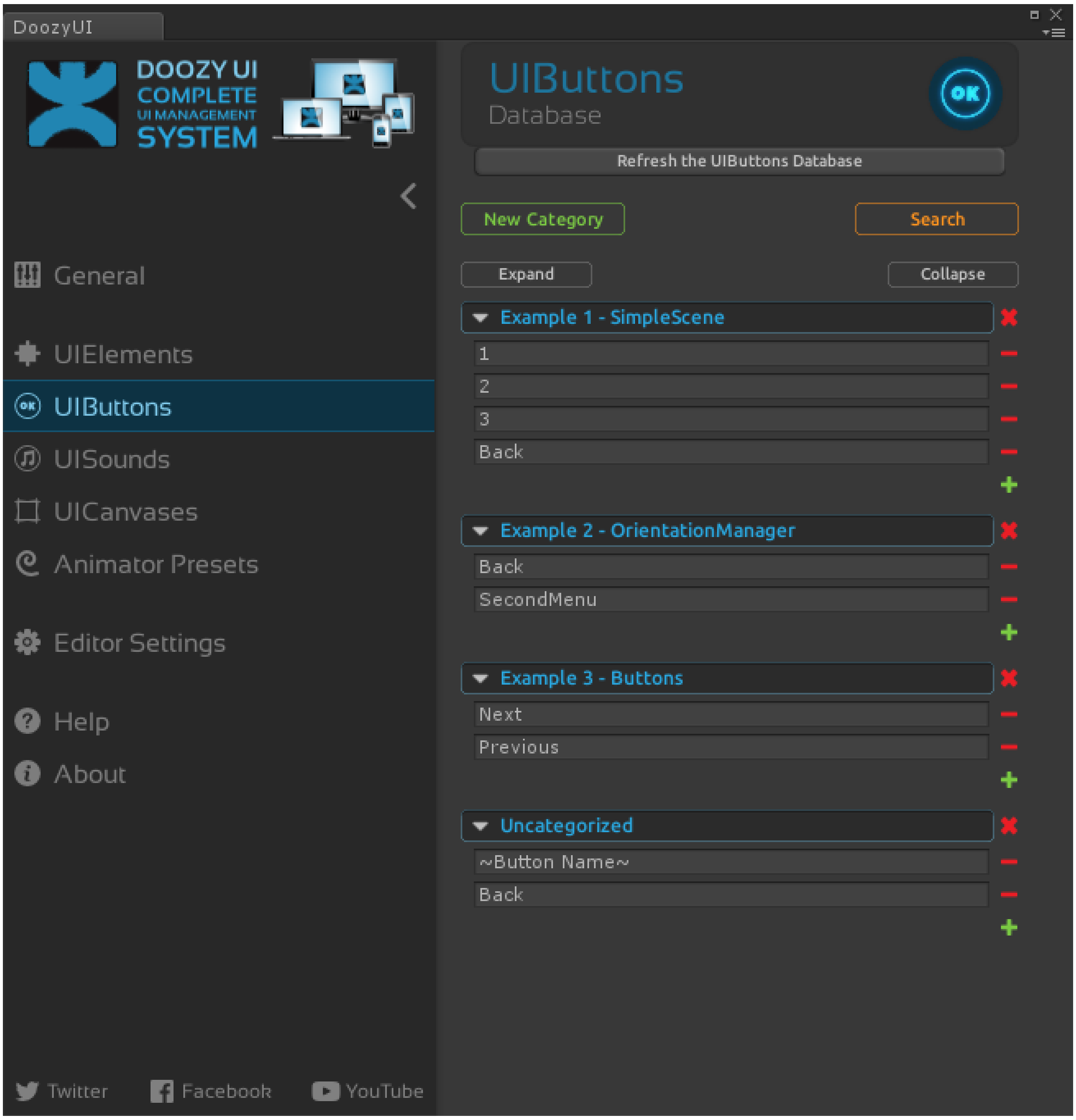
Search → Regex search for element names (Alt+S)

Expand → expand all the categories (Alt+E)

Collapse → collapse all the categories (Alt+C)

UIButtons Tab

Alt+3



Here you can add/remove button categories and add/edit/remove button names.

Refresh the UIButtons Database → executes a scan for any new button categories

New Category → create a new button category (Alt+N)

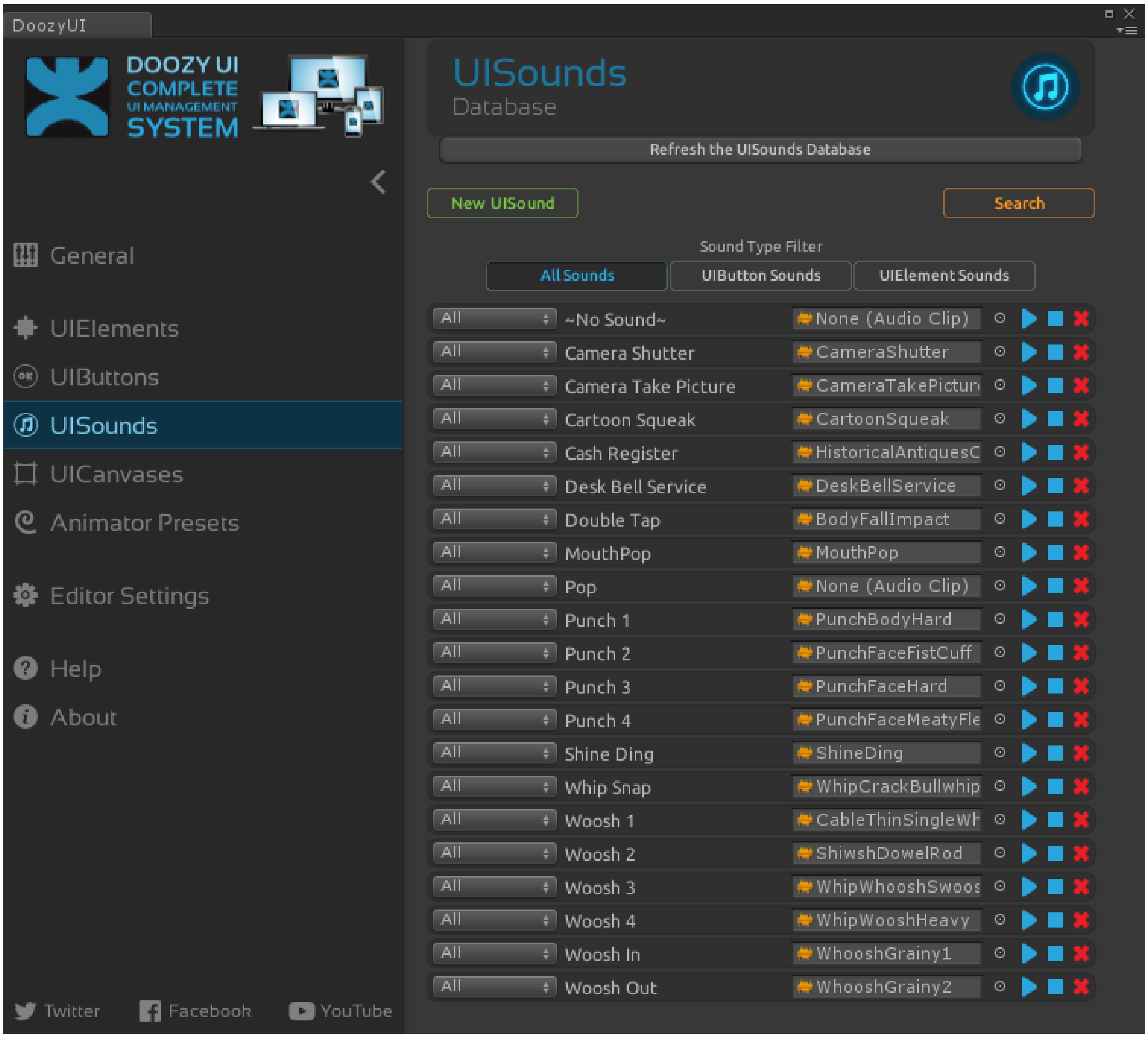
Search → Regex search for button names (Alt+S)

Expand → expand all the categories (Alt+E)

Collapse → collapse all the categories (Alt+C)

UISounds Tab

Alt+4



All	~No Sound~	None (Audio Clip)
All	Camera Shutter	CameraShutter
All	Camera Take Picture	CameraTakePictur
All	Cartoon Squeak	CartoonSqueak
All	Cash Register	HistoricalAntiquesC
All	Desk Bell Service	DeskBellService
All	Double Tap	BodyFallImpact
All	MouthPop	MouthPop
All	Pop	None (Audio Clip)
All	Punch 1	PunchBodyHard
All	Punch 2	PunchFaceFistCuff
All	Punch 3	PunchFaceHard
All	Punch 4	PunchFaceMeatyFle
All	Shine Ding	ShineDing
All	Whip Snap	WhipCrackBullwhip
All	Woosh 1	CableThinSingleWh
All	Woosh 2	ShiwshDowelRod
All	Woosh 3	WhipWhooshSwoos
All	Woosh 4	WhipWooshHeavy
All	Woosh In	WhooshGrainy1
All	Woosh Out	WhooshGrainy2

Here you can add/remove UISounds, reference AudioClips and preview the sounds.

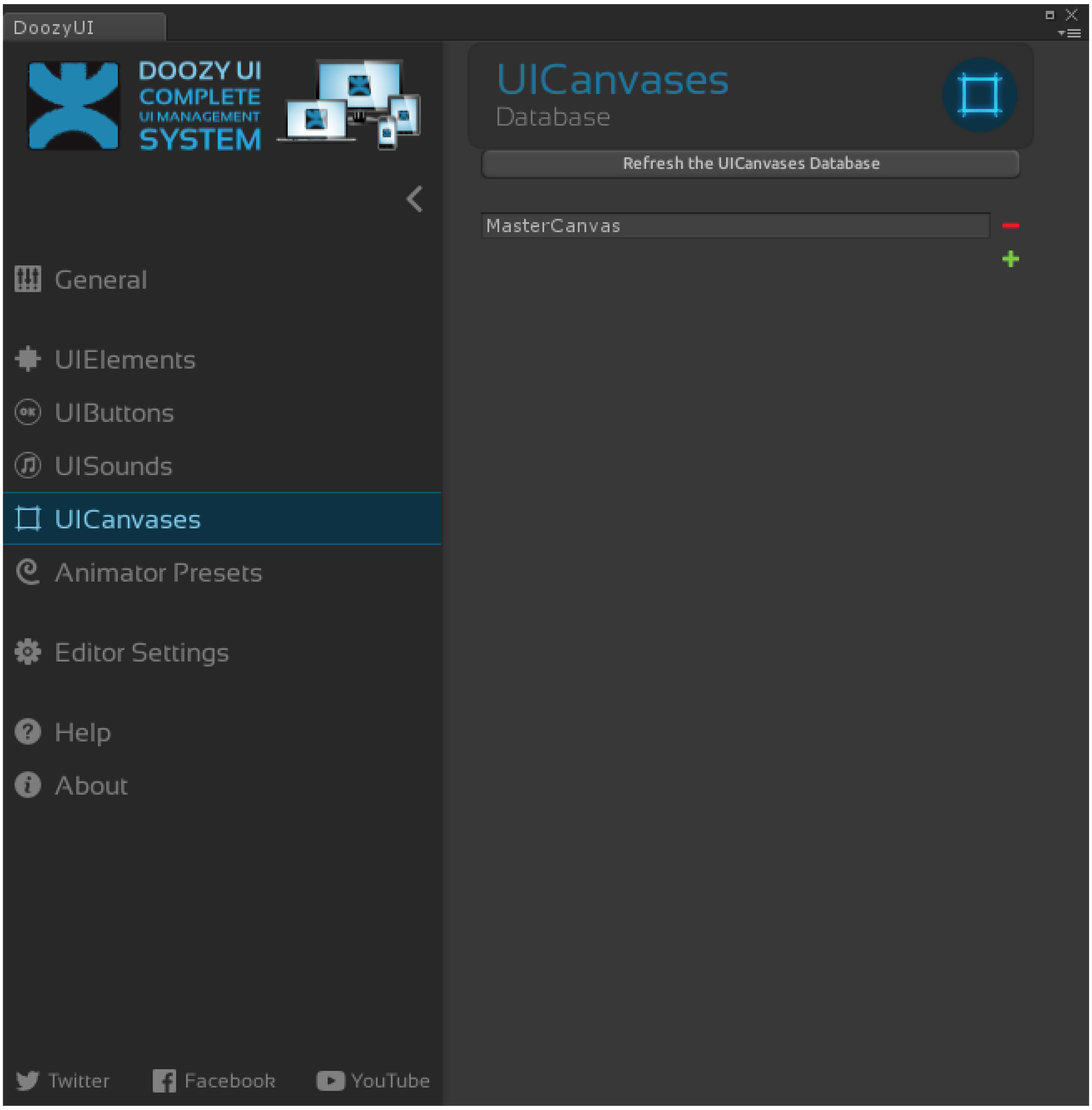
Refresh the UISounds Database → executes a scan for any new UISounds

New UISound → create a new UISound asset (Alt+N)

Search → Regex search for sound names (Alt+S)

UICanvases Tab

Alt+5

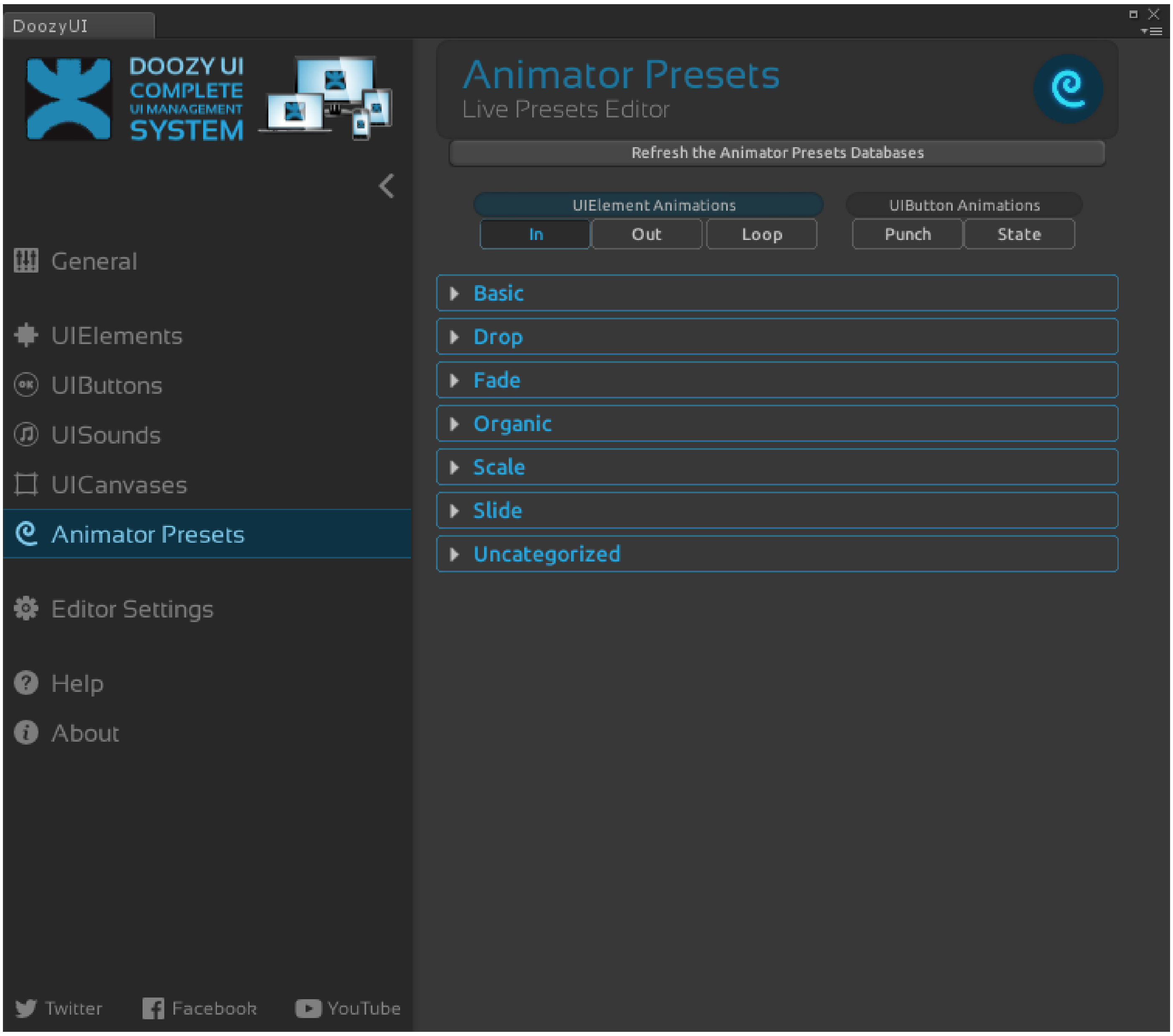


Here you can add/remove canvas names.

Refresh the UICanvases Database → executes a scan for any new canvas names

Animator Presets Tab

Alt+6

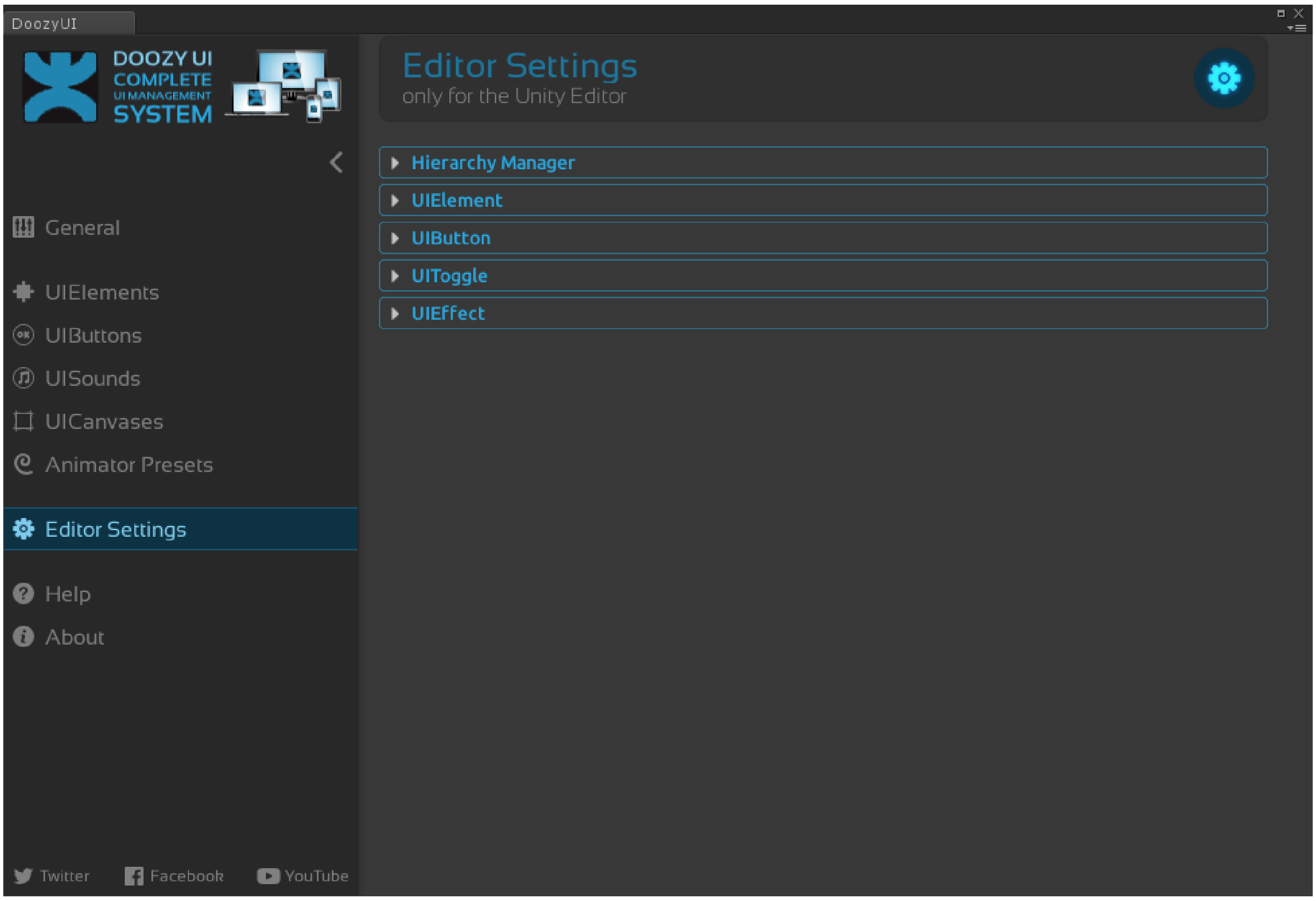


Here you can view/edit animation presets used by

- the UIElement component (In, Out and Loop animation presets)
- the UIButton component (Punch and State animation presets)
- the UIToggle component (Punch animation presets)

Editor Settings Tab

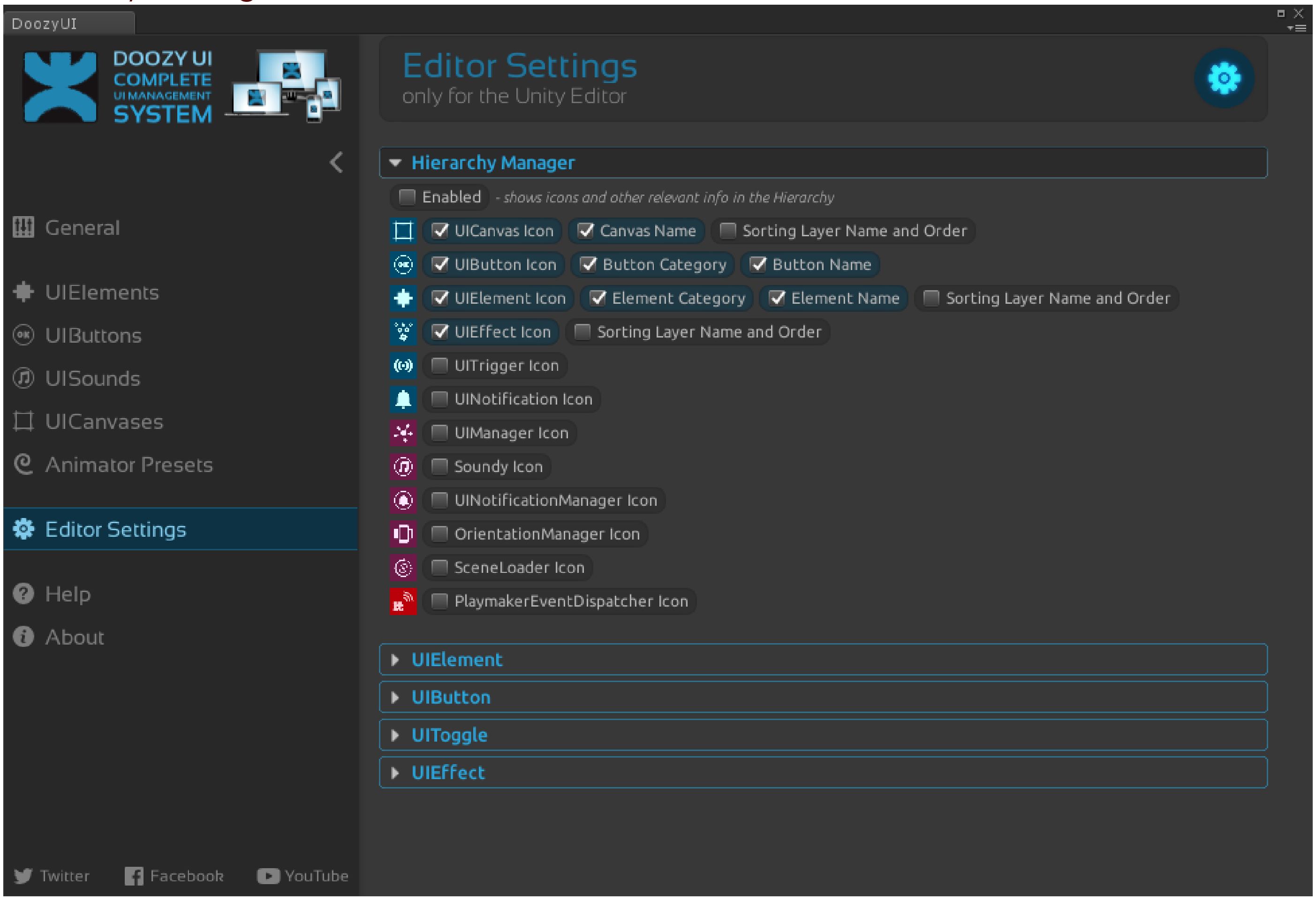
Alt+7



Here you can toggle on/off and configure the Hierarchy Manager *(this feature might slow down the Editor)*.

You can also configure default values for newly created UIElement, UIButtons, UIToggles and UIEffects.

Hierarchy Manager



Editor Settings
only for the Unity Editor

Hierarchy Manager

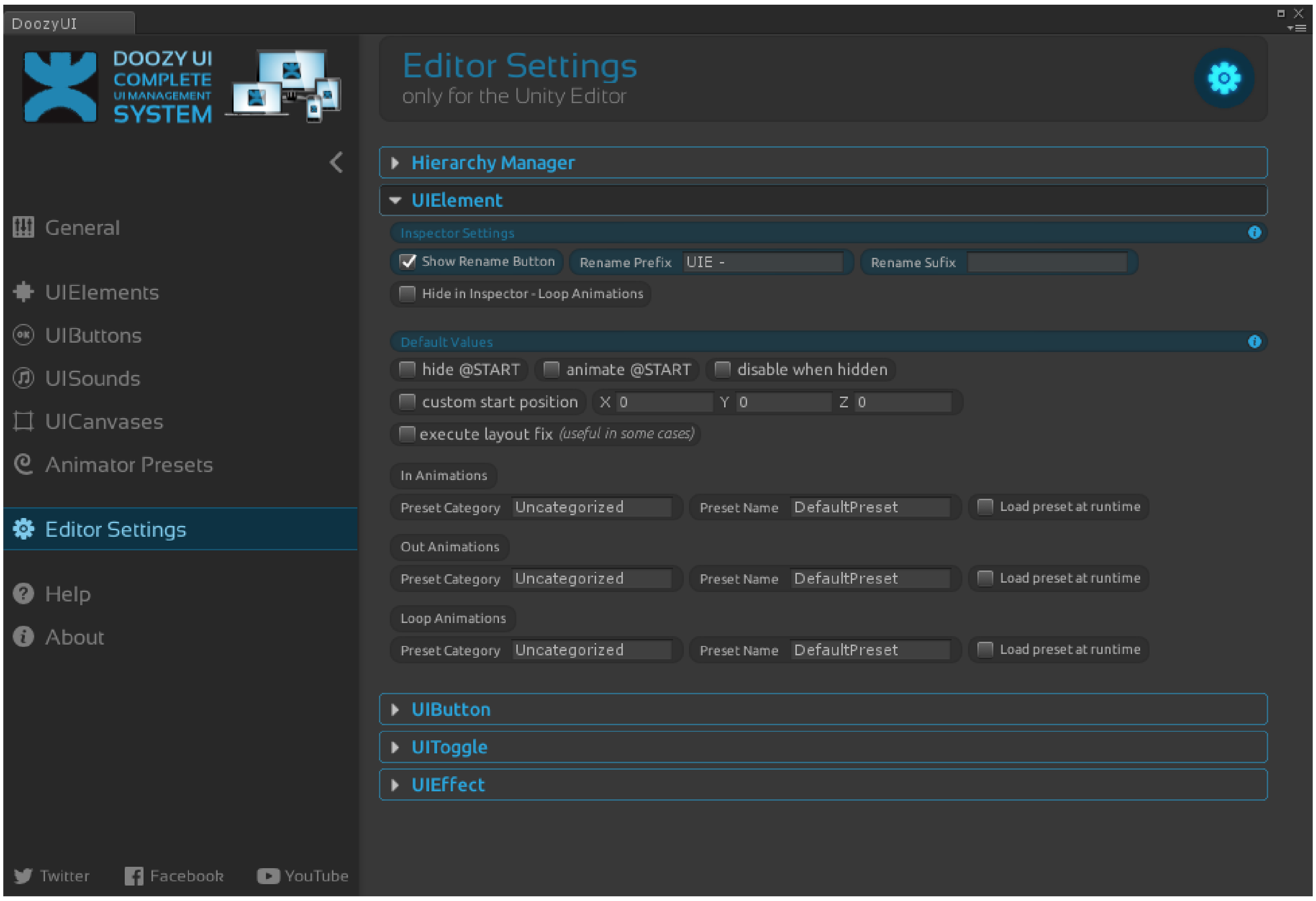
- [ ] Enabled - *shows icons and other relevant info in the Hierarchy*
- [✓] UICanvas Icon [✓] Canvas Name [ ] Sorting Layer Name and Order
- [✓] UIButton Icon [✓] Button Category [✓] Button Name
- [✓] UIElement Icon [✓] Element Category [✓] Element Name [ ] Sorting Layer Name and Order
- [✓] UIEffect Icon [ ] Sorting Layer Name and Order
- [ ] UITrigger Icon
- [ ] UINotification Icon
- [ ] UIManager Icon
- [ ] Soundy Icon
- [ ] UINotificationManager Icon
- [ ] OrientationManager Icon
- [ ] SceneLoader Icon
- [ ] PlaymakerEventDispatcher Icon

UIElement
UIButton
UIToggle
UIEffect

Here you can toggle on/off and configure the Hierarchy Manager *(this feature might slow down the Editor)*.

The Hierarchy Manager will show icons and other relevant info to the right of any DoozyUI component in the Hierarchy View. If enabled, the Hierarchy Manager will parse the all the objects in the currently opened scene and, if they are DoozyUI components, it will display any relevant info about them in the Hierarchy View.

UIElement – default values



Editor Settings
only for the Unity Editor

Inspector Settings

- [✓] Show Rename Button — Rename Prefix: UIE - — Rename Sufix:
- [ ] Hide in Inspector - Loop Animations

Default Values

- [ ] hide @START
- [ ] animate @START
- [ ] disable when hidden
- [ ] custom start position — X 0 — Y 0 — Z 0
- [ ] execute layout fix *(useful in some cases)*

In Animations: Preset Category: Uncategorized — Preset Name: DefaultPreset — [ ] Load preset at runtime

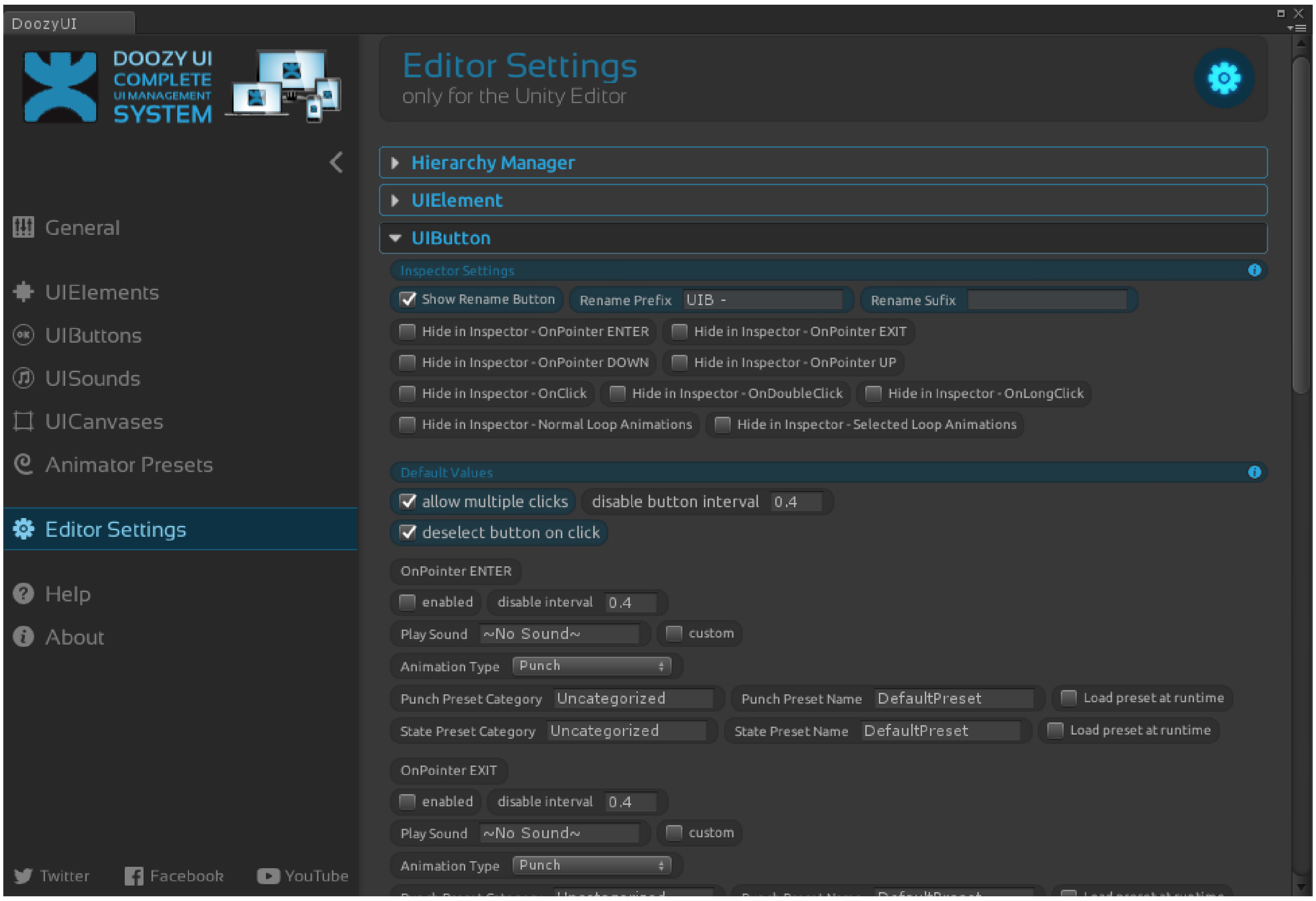
Out Animations: Preset Category: Uncategorized — Preset Name: DefaultPreset — [ ] Load preset at runtime

Loop Animations: Preset Category: Uncategorized — Preset Name: DefaultPreset — [ ] Load preset at runtime

Here you can hide the Rename Button or the Loop Animations sections, should you not want to use them.

All the other settings are the Default Values that an UIElement will have when created. This will help you create your UI even faster as you may set the animation presets and other settings from the get go.

UIButton – default values



Editor Settings
only for the Unity Editor

Hierarchy Manager
UIElement
UIButton

Inspector Settings
[✓] Show Rename Button   Rename Prefix UIB -   Rename Sufix
[ ] Hide in Inspector - OnPointer ENTER   [ ] Hide in Inspector - OnPointer EXIT
[ ] Hide in Inspector - OnPointer DOWN   [ ] Hide in Inspector - OnPointer UP
[ ] Hide in Inspector - OnClick   [ ] Hide in Inspector - OnDoubleClick   [ ] Hide in Inspector - OnLongClick
[ ] Hide in Inspector - Normal Loop Animations   [ ] Hide in Inspector - Selected Loop Animations

Default Values
[✓] allow multiple clicks   disable button interval 0.4
[✓] deselect button on click

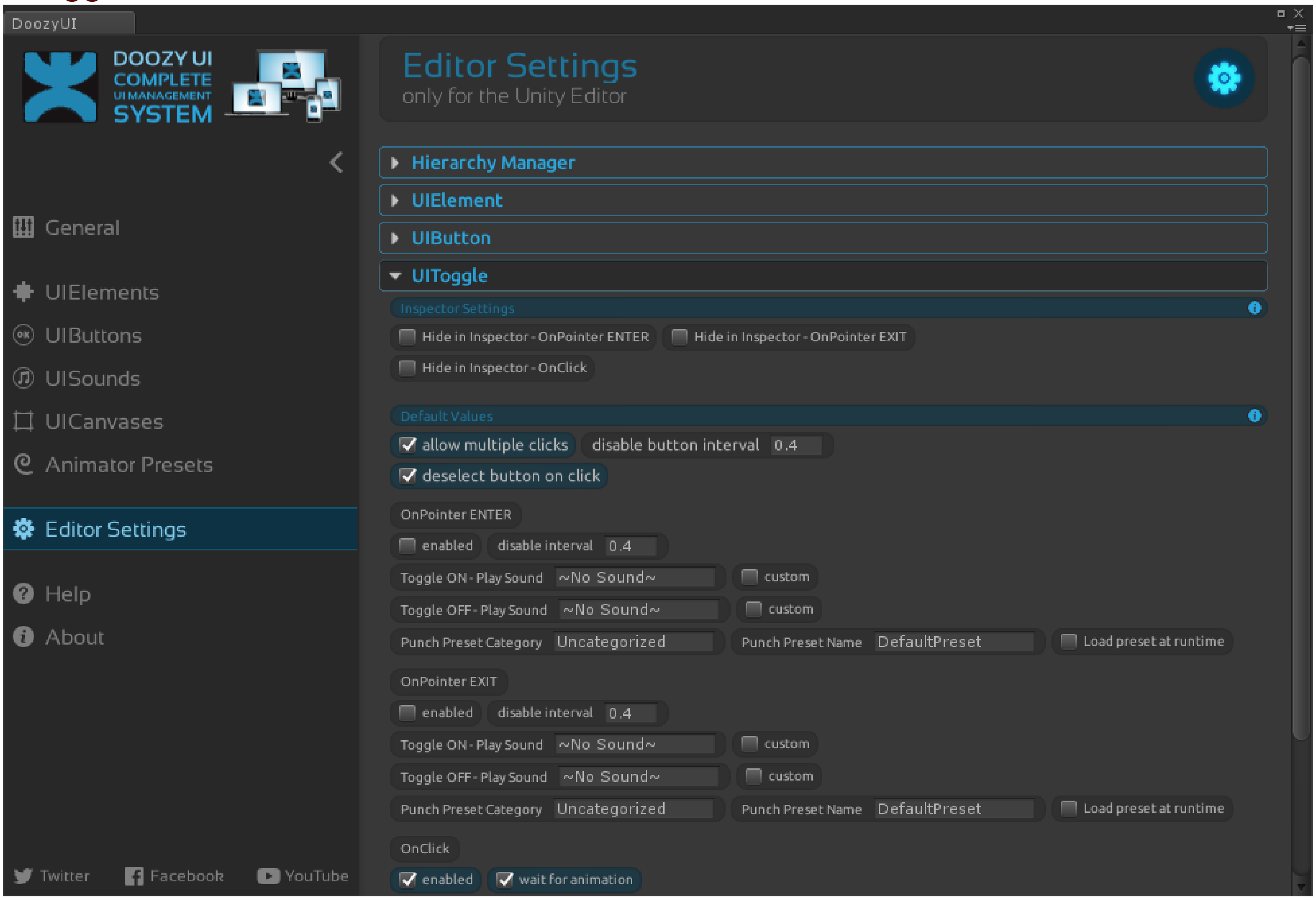
OnPointer ENTER
[ ] enabled   disable interval 0.4
Play Sound ~No Sound~   [ ] custom
Animation Type Punch
Punch Preset Category Uncategorized   Punch Preset Name DefaultPreset   [ ] Load preset at runtime
State Preset Category Uncategorized   State Preset Name DefaultPreset   [ ] Load preset at runtime

OnPointer EXIT
[ ] enabled   disable interval 0.4
Play Sound ~No Sound~   [ ] custom
Animation Type Punch

Here you can hide the Rename Button or any tabs that you might not need or want to use in your ongoing project (like OnPointerEnter, OnPointerExit and so on…).

All the other settings are the default values that an UIButton will have when created. This will help you create your UI event faster as you can even set the animation presets and other settings when creating new UIButtons.

UIToggle – default values


DoozyUI

Editor Settings
only for the Unity Editor

- General
- UIElements
- UIButtons
- UISounds
- UICanvases
- Animator Presets
- Editor Settings
- Help
- About

Twitter Facebook YouTube

▸ Hierarchy Manager
▸ UIElement
▸ UIButton
▾ UIToggle

Inspector Settings
- ☐ Hide in Inspector - OnPointer ENTER
- ☐ Hide in Inspector - OnPointer EXIT
- ☐ Hide in Inspector - OnClick

Default Values
- ☑ allow multiple clicks — disable button interval 0.4
- ☑ deselect button on click

OnPointer ENTER
- ☐ enabled — disable interval 0.4
- Toggle ON - Play Sound ~No Sound~ ☐ custom
- Toggle OFF - Play Sound ~No Sound~ ☐ custom
- Punch Preset Category Uncategorized — Punch Preset Name DefaultPreset — ☐ Load preset at runtime

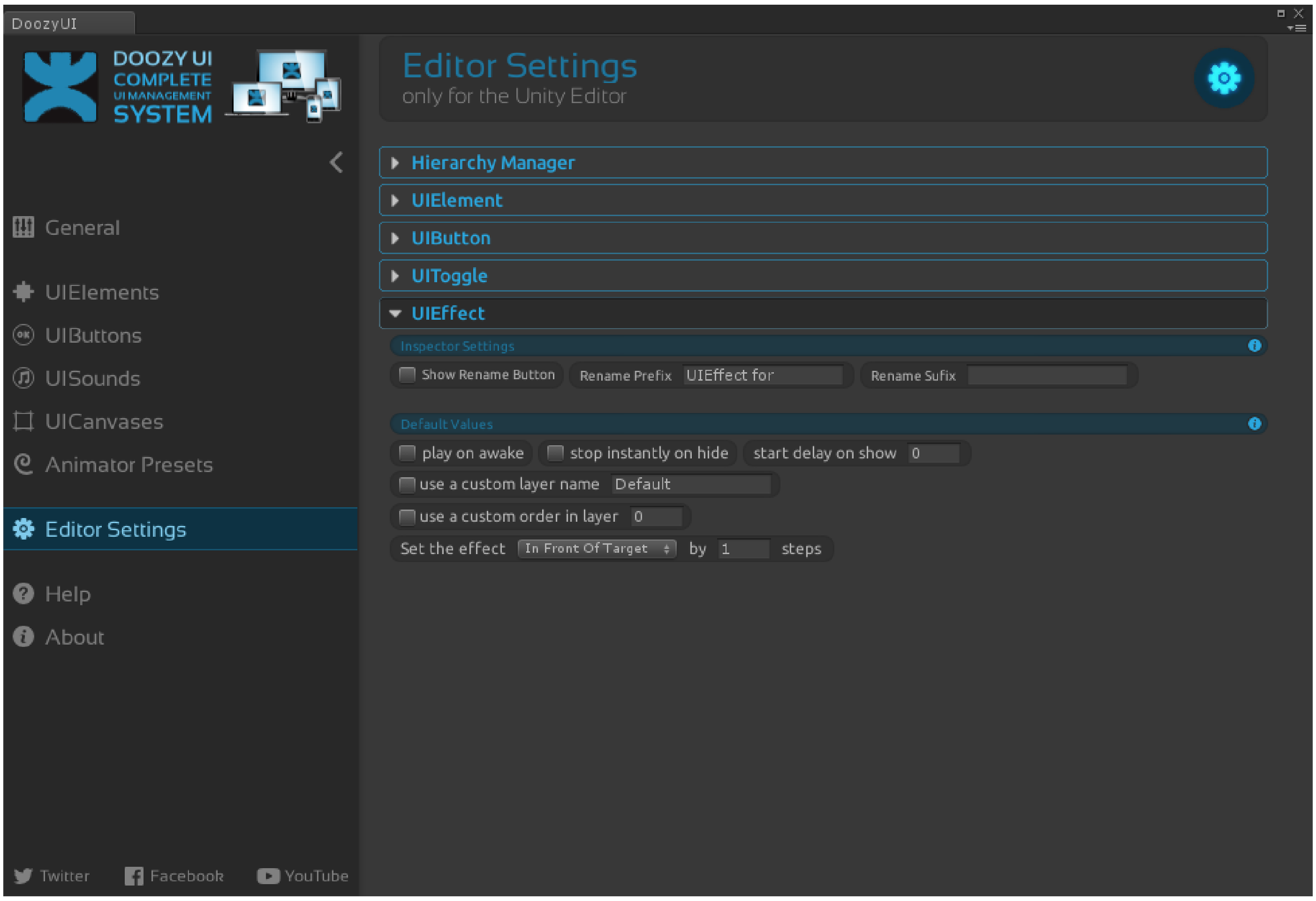
OnPointer EXIT
- ☐ enabled — disable interval 0.4
- Toggle ON - Play Sound ~No Sound~ ☐ custom
- Toggle OFF - Play Sound ~No Sound~ ☐ custom
- Punch Preset Category Uncategorized — Punch Preset Name DefaultPreset — ☐ Load preset at runtime

OnClick
- ☑ enabled ☑ wait for animation

Here you can hide any tabs that you might not need or want to use in your ongoing project (like OnPointerEnter, OnPointerExit and so on…).

All the other settings are the default values that an UIToggle will have when created. This will help you create your UI event faster as you can even set the animation presets and other settings when creating new UIToggles.

UIEffect – default values



Inspector Settings

Show Rename Button [ ] | Rename Prefix: UIEffect for | Rename Sufix:

Default Values

- [ ] play on awake
- [ ] stop instantly on hide
- start delay on show: 0
- [ ] use a custom layer name: Default
- [ ] use a custom order in layer: 0
- Set the effect: In Front Of Target, by 1 steps

Here you can hide the Rename Button or any tabs that you might not need or want to use in your ongoing project.

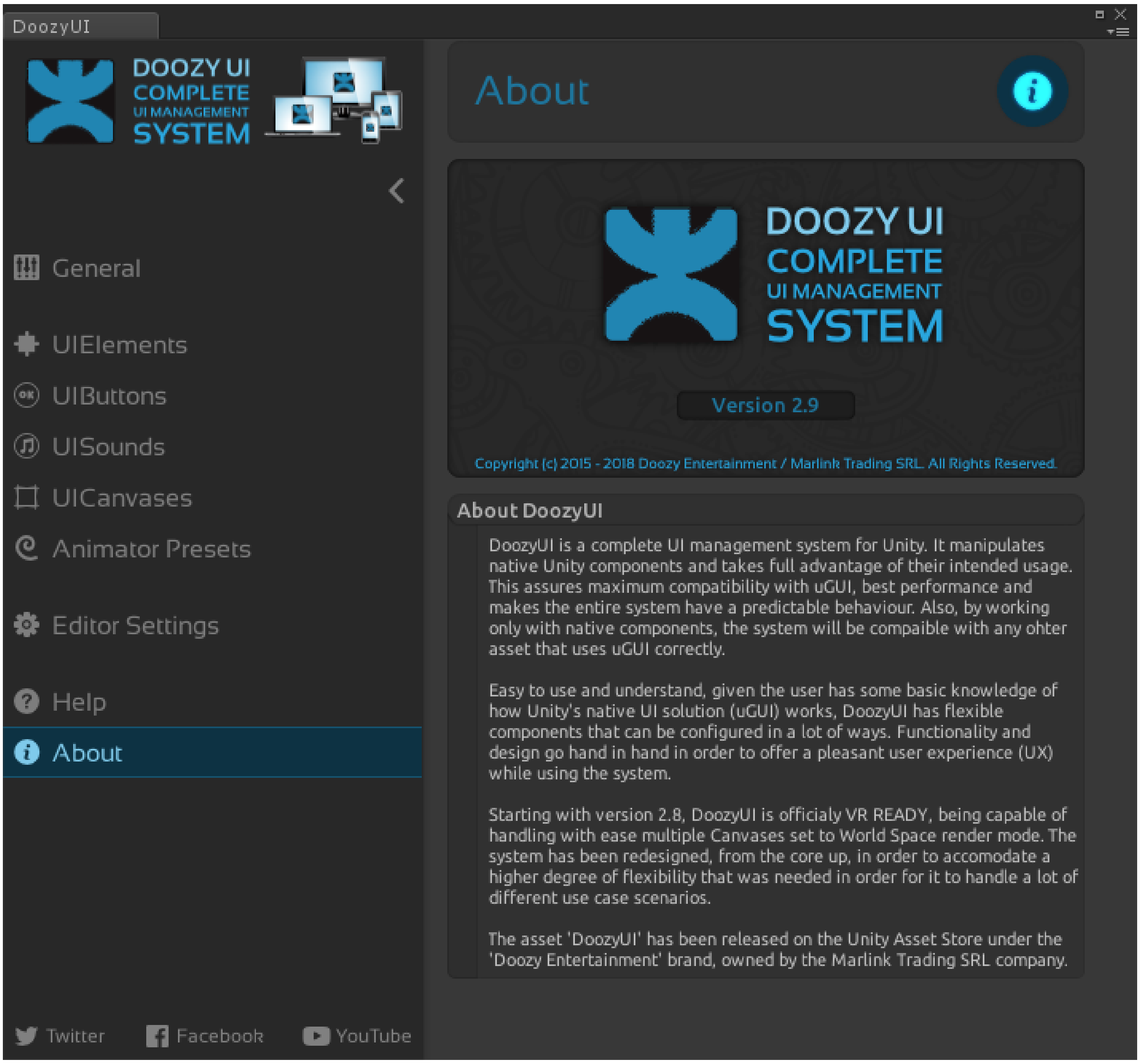
All the other settings are the default values that an UIEffect will have when created. This will help you create your UI even faster as you may set the sorting layer name, the order in layer and other settings from the get go.

Help Tab

Here you can access the following:

- link to the online version of this manual
- link to the DoozyUI Help Center
- the support email address

About Tab



Copyright (c) 2015 - 2018 Doozy Entertainment / Marlink Trading SRL. All Rights Reserved.

About DoozyUI

DoozyUI is a complete UI management system for Unity. It manipulates native Unity components and takes full advantage of their intended usage. This assures maximum compatibility with uGUI, best performance and makes the entire system have a predictable behaviour. Also, by working only with native components, the system will be compaible with any ohter asset that uses uGUI correctly.

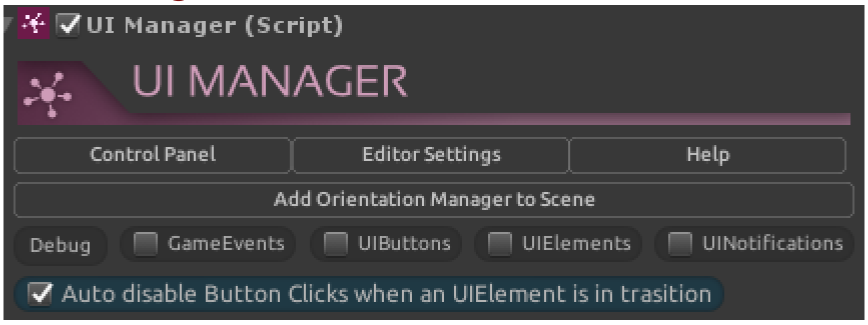
Easy to use and understand, given the user has some basic knowledge of how Unity's native UI solution (uGUI) works, DoozyUI has flexible components that can be configured in a lot of ways. Functionality and design go hand in hand in order to offer a pleasant user experience (UX) while using the system.

Starting with version 2.8, DoozyUI is officialy VR READY, being capable of handling with ease multiple Canvases set to World Space render mode. The system has been redesigned, from the core up, in order to accomodate a higher degree of flexibility that was needed in order for it to handle a lot of different use case scenarios.

The asset 'DoozyUI' has been released on the Unity Asset Store under the 'Doozy Entertainment' brand, owned by the Marlink Trading SRL company.

Here you can view what is the currently install version of DoozyUI and read about the system.

UIManager



The UIManager is the core of DoozyUI as it alone manages the entire system by binding all the relevant components together.

It has a lot of methods that will help you manage your designed UI structure.

The component (shown above) has the following options available:

Control Panel: opens the Control Panel window
Editor Settings: opens the Control Panel window at the Editor Settings Tab
Help: opens the Control Panel window at the Help Tab
Add Orientation Manager to Scene: ads the Orientation Manager gameObject to the currently opened scene
Debug GameEvents: prints to the console all the GameEvents that are being sent at runtime
Debug UIButtons: prints to the console all the button interactions that are captured by the system at runtime
Debug UIElements: prints to the console all the show and hide commands issued at runtime
Debug UINotifications: prints to the console all the relevant info about shown UINotifications at runtime
Auto disable Button Clicks when an UIElement is in transition: forces the system to ignore any button clicks that are performed then an UIElement is executing an In or an Out animation; this feature prevents the possibility that a click is issued when an UIElement transition is happening

CODE

To be able to access DoozyUI components in code, you need to add the following using on top of your classes.

```
using DoozyUI;
```

Fields

Name	*Description*
autoDisableButtonClicks	Should the system disable button clicks when an UIElement is in transition (an In or Out animation is running). Default is true.
currentGameTimeScale	Every time the user pauses the game, this variable stores the current Time.timeScale value. This is needed so that when the game needs to get unpaused, UIManager will know at what timescale should the game return to.
currentOrientation	Returns the current orientation of the device. Default is Orientation.Unknown because that triggers an orientation check/update.
debugGameEvents	Prints debug messages related to game events at runtime.
debugUIButtons	Prints debug messages related to UIButtons at runtime.
debugUICanvases	Prints debug messages related to UICanvases at runtime.
debugUIElements	Prints debug messages related to UIElements at runtime.
debugUINotifications	Prints debug messages related to UINotifications at runtime.
gamePaused	Returns true if the game has been paused (by the UIManager) and false otherwise.
isMusicOn	Returns true if the music is on and false otherwise. This variable knows only if the music is on for DoozyUI and not for anything else as it checks a PlayerPrefs int value named 'musicState'.
isSoundOn	Returns true if the sound is on and false otherwise. This variable knows only if the sound is on for DoozyUI and not for anything else as it checks a PlayerPrefs int value named 'soundState'.
useOrientationManager	Determines if the Orientation Manager should be used. This value is automatically set to true when the 'dUI_UseOrientationManager' Scripting Define Symbol has been added to the current active platform.
useOrientationManager	Determines if the Orientation Manager should be used. This value is automatically set to false when the 'dUI_UseOrientationManager' Scripting Define Symbol has not been added to the currently active platform.

useSceneLoader	Determines if the Scene Loader should be automatically loaded. This value is automatically set to false when the 'dUI_SceneLoaderDisabled' Scripting Define Symbol has been added to the current active platform.
useSceneLoader	Determines if the Scene Loader should be automatically loaded. This value is automatically set to true when the 'dUI_SceneLoaderDisabled' Scripting Define Symbol has not been added to the currently active platform.
usesTMPro	Global static variable that determines if the UINotification look for TextMeshProUGUI component instead of a Text componenet when looking for text. TextMeshPro support is currently in limbo as we wait to see what Unity does with it.

Properties

Name	*Description*
BackButtonDisabled	Returns true if the 'Back' button is disabled and false otherwise.
ButtonClicksDisabled	Returns true if button clicks are disabled and false otherwise. This is mosty used when an UIElement is in transition and, in order to prevent accidental clicks, the buttons need to be disabled.
ButtonClicksTriggerDatabase	Returns a registry of all the registered UITriggers that listens for button clicks.
ButtonDoubleClicksTriggerDatabase	Returns a registry of all the registered UITriggers that listens for button clicks.
ButtonLongClicksTriggerRegistry	Returns a registry of all the registered UITriggers that listens for button clicks.
CanvasDatabase	Returns a registry of all the registered UICanvases.
EffectDatabase	Returns a registry of all the registered UIEffects.
ElementDatabase	Returns a registry of all the registered UIElements.
GameEventsTriggerDatabase	Returns a registry of all the registered UITriggers that listens for game events.
Instance	Gets the instance.
IsNavigationEnabled	Returns true if the UI Navigation is enabled and false otherwise. It is set to false if Scripting Define Symbols, for the current active platform, contain the 'dUI_NavigationDisabled' symbol. In you want to handle the UI Navigation yourself just disable the UI Navigation from the Control Panel.
NotificationManager	Returns the UINotificationManager component.
OrientationManager	Returns the OrientationManager reference.
PlaymakerEventDispatcherDatabase	Returns a registry of all the registered PlaymakerEventDispatchers.
SceneLoader	Returns the SceneLoader reference.
Soundy	Returns the Soundy component.
TouchManager	Returns the TouchManager reference.

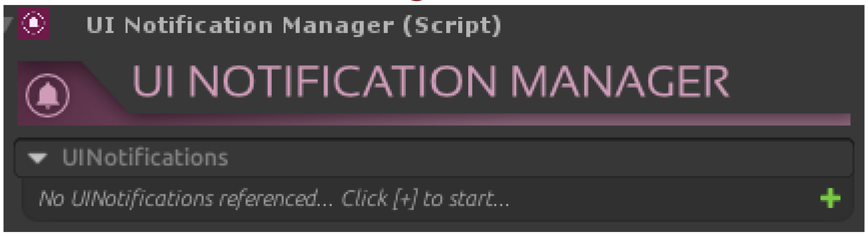
Methods

Name	*Description*
ApplicationQuit()	Exits play mode (if in editor) or quits the application if in build mode
BackButtonEvent()	The 'back' button was pressed (or escape key)
ChangeOrientation(Orientation)	Updates the current orientation to the new given one.
CreateCanvas(string)	Creates an UICanvas with the given canvas name and returns the reference to it.
DisableBackButton()	Disables the 'Back' button functionality
DisableButtonClicks()	Disables all the button clicks. This is triggered by the system when an UIElement started a transition (IN/OUT animations).
DispatchEventToPlaymakerEventDispatchers(string, DUI.EventType)	This method is obsolete, please use SendEventToPlaymaker instead.
EnableBackButton()	Enables the 'Back' button functionality
EnableBackButtonByForce()	Enables the 'Back' button functionality by resetting the additive bool to zero. backButtonDisableLevel = 0. Use this ONLY for special cases when something wrong happens, and the back button is stuck in disabled mode.
EnableButtonClicks()	Enables all the button clicks. This is triggered by the system when an UIElement finished a transition (IN/OUT animations).
EnableButtonClicksByForce()	Enables the button clicks by resetting the additive bool to zero. buttonClicksDisableLevel = 0. Use this ONLY for special cases when something unexpected happens and the button clicks are stuck in disabled mode.

GetCanvas(string, bool, bool)	Returns a reference to an UICanvas that has the given canvas name. It can also create the canvas you are searching for or just return the 'MasterCanvas' UICanvas.
GetMasterCanvas(bool)	Returns a reference to an UICanvas that is considered and used as a 'MasterCanvas'. If no such canvas exists, one will get created automatically by default.
GetUiEffects(string, string)	Returns a List of all UIEffects that are linked to an UIElement with a given name and category. If no UIEffect was found, it will return an empty list.
GetUiElements(string, string)	Returns a List of all UIElements that have a given name and category. If no UIElement was found, it will return an empty list.
GetUiTriggers(string, DUI.EventType)	Returns a list of all the UITriggers that are linked to the given triggerValue and of the given triggerType.
GetVisibleUIElements()	Returns a List of all the UIElements that are visible on the screen. An UIElement is considered visible if isVisible = true. If eDatabase is null or empty or if no UIElements are visible, it will return an empty list.
HideUiElement(string, string)	Hides all the UIElements that have the given name and category.
HideUiElement(string, bool)	Hides all the UIElements that have the given name and the DEFAULT CATEGORY name.
HideUiElement(string, string, bool)	Hides all the UIElements that have the given name and category.
MusicCheck()	Checks the musicState when the game starts in the PlayerPrefs
PlaySound(string)	Plays the given sound name, through Soundy. You can also use Soundy.PlaySound... Note: If support for MasterAudio is enabled it will play the sound name from the MasterAudio sounds database.
PlaySound(string, float)	Plays the given sound name, through Soundy. You can also use Soundy.PlaySound... Note: If support for MasterAudio is enabled it will play the sound name from the MasterAudio sounds database.
PlaySound(string, float, float)	Plays the given sound name, through Soundy. You can also use Soundy.PlaySound... Note: If support for MasterAudio is enabled it will play the sound name from the MasterAudio sounds database.
PlaySound(AudioClip)	Plays the given audio clip, through Soundy. You can also use Soundy.PlaySound...
PlaySound(AudioClip, float)	Plays the given audio clip at the given volume level, through Soundy. You can also use Soundy.PlaySound...
PlaySound(AudioClip, float, float)	Plays the given audio clip at the given volume and pitch levels, through Soundy. You can also use Soundy.PlaySound...
PlaySoundFromResources(string, float)	Plays the given sound name by searching the Resources folder for it, through Soundy. You can also use Soundy.PlaySound...
PlaySoundFromResources(string, float, float)	Plays the given sound name by searching the Resources folder for it, through Soundy. You can also use Soundy.PlaySound...
PlaySoundFromResources(string)	Plays the given sound name by searching the Resources folder for it, through Soundy. You can also use Soundy.PlaySound...
RegisterToNotificationQueue(UINotification.NotificationData)	Every notification that needs to enter the Notification Queue will be added to the notificatioQueue list as the last item.
SendButtonAction(string, UIButton.ButtonClickType)	Sends a button action with just a button name and what type of click it is. This method is used to simulate a button action since it does not have an UIButton reference.
SendButtonAction(UIButton, UIButton.ButtonActionType)	Sends a button action with a reference to the UIButton that sent it and what type of action it is.
SendButtonAction(string, UIButton.ButtonActionType)	Sends a button action with just a button name and what type of action it is. This method is used to simulate a button action since it does not have an UIButton reference.
SendButtonClick(string, bool, List<string>, List<string>, List<string>)	Use SendButtonAction instead.
SendEventToPlaymaker(string, DUI.EventType)	Sends an event that can be either a Game Event or Button Click to all the registered Playmaker Event Dispatchers.
SendGameEvent(string)	Sends the given game event.
SendGameEvents(List<string>)	Sends the given list of game events.
ShowNotification(GameObject, float, bool, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.

ShowNotification(GameObject, float, bool, string, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(GameObject, float, bool, string, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, string, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string, Sprite, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, string, Sprite, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(string, float, bool, string, string, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(string, float, bool, string, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(string, float, bool, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, string, Sprite, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(GameObject, float, bool, string, string, Sprite, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowUiElement(string, string, bool)	Shows all the UIElements that have the given name and category.
ShowUiElement(string, string)	Shows all the UIElements that have the given name and category.
ShowUiElement(string, bool)	Shows all the UIElements that have the given name and the DEFAULT CATEGORY name.
SoundCheck()	Checks the soundState when the game starts in the PlayerPrefs
ToggleMusic()	Toggles the musicState and saves it to the PlayerPrefs
TogglePause()	Pauses or Unpauses the application
ToggleSound()	Toggles the soundState and saves it to the PlayerPrefs
TriggerTheTriggers(string, DUI.EventType)	Triggers all the UITriggers that are listening for the given triggerValue and are of the given triggerType.
UnregisterFromNotificationQueue(UINotification.NotificationData)	Unregisters a notification, by removing the notification data that started it.
UpdateCanvasSortingLayerName(GameObject, string)	Updates the sorting layer for all the canvases on and under the target gameObject
UpdateRendererSortingLayerName(GameObject, string)	Updates all the sorting layer for all the renderers on and under the target gameObject

UINotificationManager


UI Notification Manager (Script)

UI NOTIFICATION MANAGER

UINotifications

No UINotifications referenced... Click [+] to start...

The UINotificationManager keeps references to all the UINotification prefabs that were not put under a folder named ‘Resources’.

Another very important function this manager performs is to show all UINotifications and to keep track of any of them that need to be shown sequentially (in a queue).

UINotifications can be shown just by using the prefab name, but they need to either be under a folder named 'Resources' or referenced to the UINotification Manager.

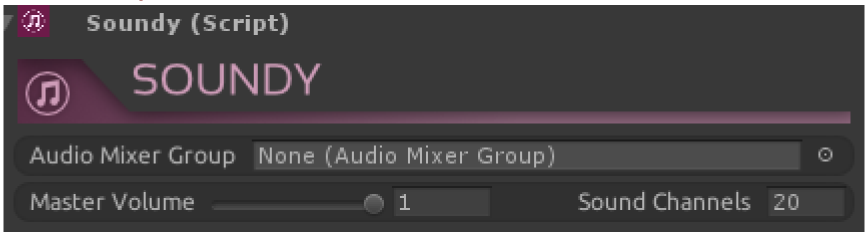
The Notification Queue is used when trying to show more than one UINotification at once. This way, notifications can be queued to appear one after another instead of being shown all at once as they may get stacked one over the other.

Methods

Name	*Description*
RegisterToNotificationQueue(UINotification.NotificationData)	Every notification that needs to enter the Notification Queue will be added to the notificatioQueue list as the last item.
UnregisterFromNotificationQueue(UINotification.NotificationData)	Unregisteres a notification, by removing the notification data that started it.
ShowNotification(GameObject, float, bool, string, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string, Sprite, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(GameObject, float, bool, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(string, float, bool, UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(string, float, bool, string, string, Sprite, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string, Sprite, string[], string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(GameObject, float, bool, string, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.

ShowNotification(GameObject, float, bool, string, string, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.
ShowNotification(string, float, bool, string, string, Sprite, string[], UnityAction[], UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(string, float, bool, string, string, Sprite, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(string, float, bool, string, string, UnityAction)	Show a premade notification with the given settings, using a prefabName.
ShowNotification(GameObject, float, bool, string, string, UnityAction)	Show a premade notification with the given settings, using a prefab GameObject reference.

Soundy


Soundy (Script)

SOUNDY

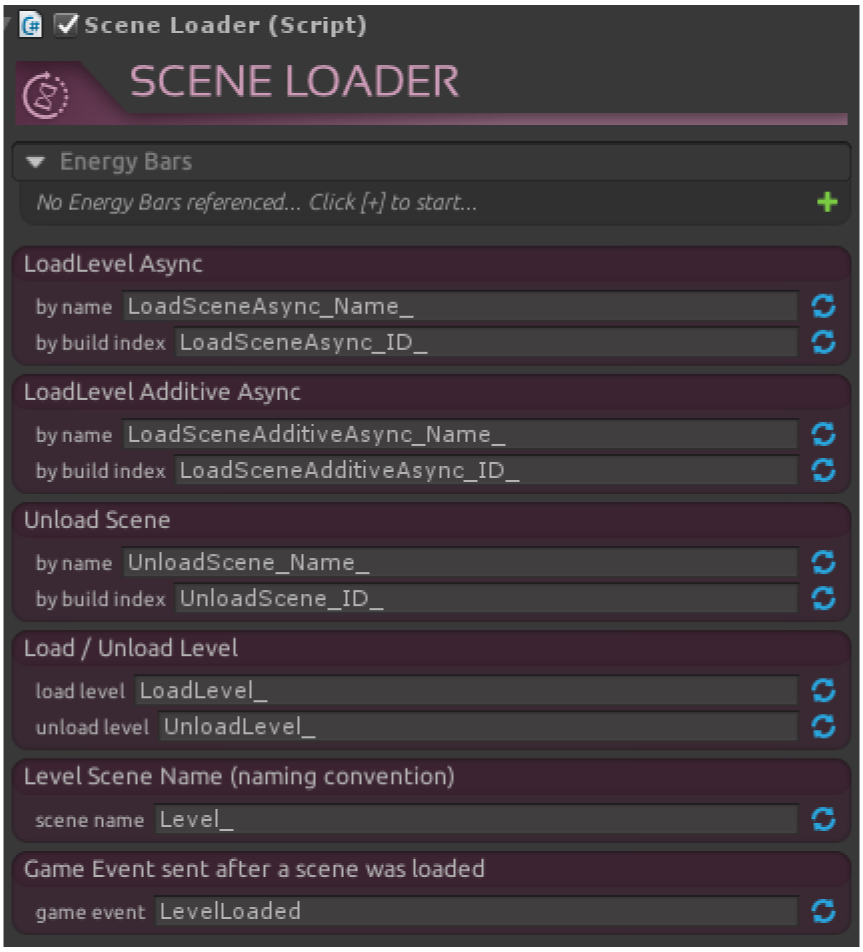
Audio Mixer Group None (Audio Mixer Group)

Master Volume 1 Sound Channels 20

Soundy is a simple sound manager that handles the playing of all the sounds (UISounds) used by the UI system.

Audio Mixer Group: if an audio mixer group is referenced, any sound (UISound) will get played (routed) through it
Master Volume: sets the default sound volume for all the played sounds
Sound Channels: a sound channel is an Audio Source and it sets how many sounds can be played at once

Scene Loader


Scene Loader (Script)

SCENE LOADER

Energy Bars

No Energy Bars referenced... Click [+] to start...

LoadLevel Async

by name LoadSceneAsync_Name_

by build index LoadSceneAsync_ID_

LoadLevel Additive Async

by name LoadSceneAdditiveAsync_Name_

by build index LoadSceneAdditiveAsync_ID_

Unload Scene

by name UnloadScene_Name_

by build index UnloadScene_ID_

Load / Unload Level

load level LoadLevel_

unload level UnloadLevel_

Level Scene Name (naming convention)

scene name Level_

Game Event sent after a scene was loaded

game event LevelLoaded

The Scene Loader helps with the loading and with the unloading of scenes. You can load or unload a scene just by sending a simple game event (a string) formatted in an easy to understand say.

The option to reference 'Energy Bars' will be available only if support for Energy Bar Toolkit is enabled. The referenced bars will get updated with the progress of the scene that is currently loading.

You can customize the game event formatting from the default one, just by editing the strings in the Inspector.

LoadLevel Async
LoadSceneAsync_Name_[sceneName]
Usage example: *To load the scene named 'MySceneName_5' you need to send a game event with the command 'LoadSceneAsync_Name_MySceneName_5', where 'LoadSceneAsync_Name_' is the first part of the command and 'MySceneName_5' is the name of the scene you want to load.*
Example code: UIManager.SendGameEvent("'LoadSceneAsync_Name_MySceneName_5");

LoadSceneAsync_ID_[buildIndex]
Usage example: *To load the 5th scene in your build index you need to send a game event with the command LoadSceneAsync_ID_5', where 'LoadSceneAsync_ID_' is the first part of the command and '5' is the build index number of the scene you want to load.*
Example code: UIManager.SendGameEvent("LoadSceneAsync_ID_5");

LoadLevel Additive Async
LoadSceneAdditiveAsync_Name_[sceneName]
Usage example: *To load the scene named 'MySceneName_5' you need to send a game event with the command 'LoadSceneAdditiveAsync_Name_MySceneName_5', where 'LoadSceneAdditiveAsync_Name_' is the first part of the comman and 'MySceneName_5' is the name of the scene you want to load.*
Example code: UIManager.SendGameEvent("LoadSceneAdditiveAsync_Name_MySceneName_5");

LoadSceneAdditiveAsync_ID_[buildIndex]

Usage example: *To load the 5th scene in your build index you need to send a game event with the command 'LoadSceneAdditiveAsync_ID_5', where 'LoadSceneAdditiveAsync_ID_' is the first part of the command and '5' is the build index number of the scene you want to load.*
Example code: `UIManager.SendGameEvent(“LoadSceneAdditiveAsync_ID_5”);`

Unload Scene

UnloadScene_Name_[sceneName]

Usage example: *To unload a scene named 'MySceneName_5' you need to send a game event with the command 'UnloadScene_Name_MyScene_5', where 'UnloadScene_Name_' is the first part of the command and 'MySceneName_5' is the name of the scene you want to unload.*
Example code: `UIManager.SendGameEvent(“UnloadScene_Name_MyScene_5”);`

UnloadScene_ID_[buildIndex]

Usage example: *To unload the 5th scene in your build index you need to send a game event with the command 'UnloadScene_ID_5', where 'UnloadScene_ID_' is the first part of the command and '5' is the build index number of the scene you want to unload.*
Example code: `UIManager.SendGameEvent(“UnloadScene_ID_5”);`

Load / Unload Level

LoadLevel_ [levelNumber]

shortcut command
Usage example: *To load level 5 you need to send a game event with the command 'LoadLevel_5', where 'LoadLevel_' is the shortcut command and '5' is the level you want to load.*
Example code: `UIManager.SendGameEvent(“LoadLevel_5”);`

UnloadLevel_[levelNumber]

shortcut command
Usage example: *To unload level 5 you need to send a game event with the command 'UnloadLevel_5', where 'UnloadLevel_' is the shortcut command and '5' is the level you want to unload.*
Example code: `UIManager.SendGameEvent(“UnloadLevel_5”);`

Level Scene Name (naming convention)

This is the name for your level scenes in build. Example: 'Level_1', 'Level_2' ... 'Level_100'.
To be clear, just name your scenes ‘Level_’ and level number. You do this in order to be able to use the ‘LoadLevel_’ and ‘UnloadLevel_’ commands.

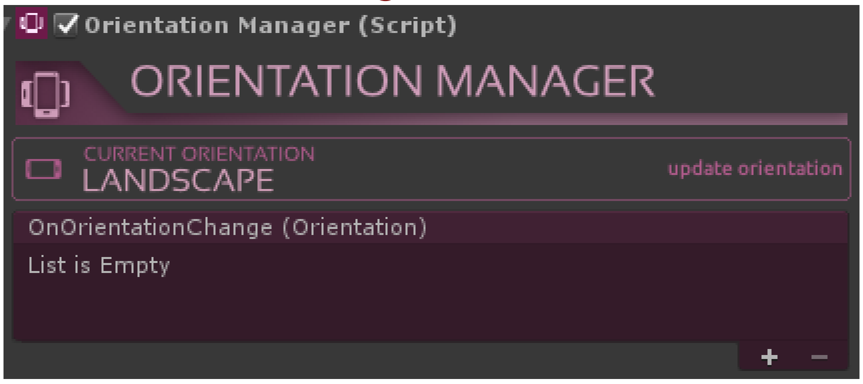
Game Event sent after a scene was loaded

After a scene has been loaded (using the Scene Loader), the system sends a game event named ‘LevelLoaded’. This game event is sent by default and you can change it should you want to. Just keep in mind that this is the game event you should listen for (with the help of an UITrigger) when you want to hook up to this event.

Methods

Name	*Description*
LoadLevel(int)	Loads the level.
LoadLevelAdditiveAsync(string)	Loads the level additive asynchronous.
LoadLevelAdditiveAsync(int)	Loads the level additive asynchronous.
LoadSceneAsync(string)	Loads the scene asynchronous.
LoadSceneAsync(int)	Loads the scene asynchronous.
OnGameEvent(string)	Method used by the UIManager to send the game events that trigger the loading and unloading of scenes.
UnloadLevel(int)	Unloads the level.
UnloadScene(string)	Unloads the scene.
UnloadScene(int)	Unloads the scene.

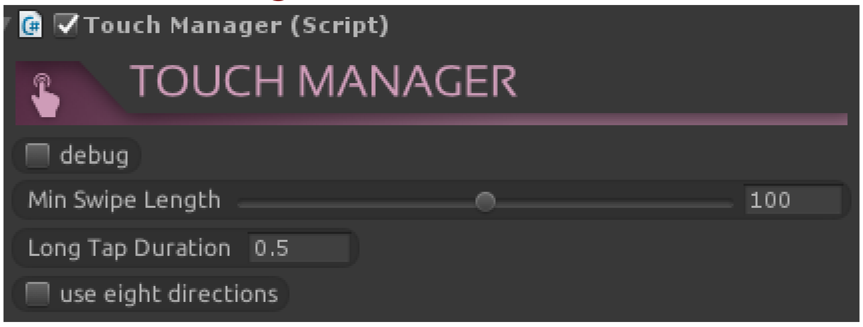
Orientation Manager



The Orientation Manager is responsible for detecting the current screen orientation of the target device. It is a simple, yet very efficient, implementation of an orientation detector (with zero overhead).

OnOrientationChange: an UnityEvent that sends an OrientaionManager.Orientation parameter when the device's orientation changes

Touch Manager


Touch Manager (Script)

TOUCH MANAGER

debug

Min Swipe Length 100

Long Tap Duration 0.5

use eight directions

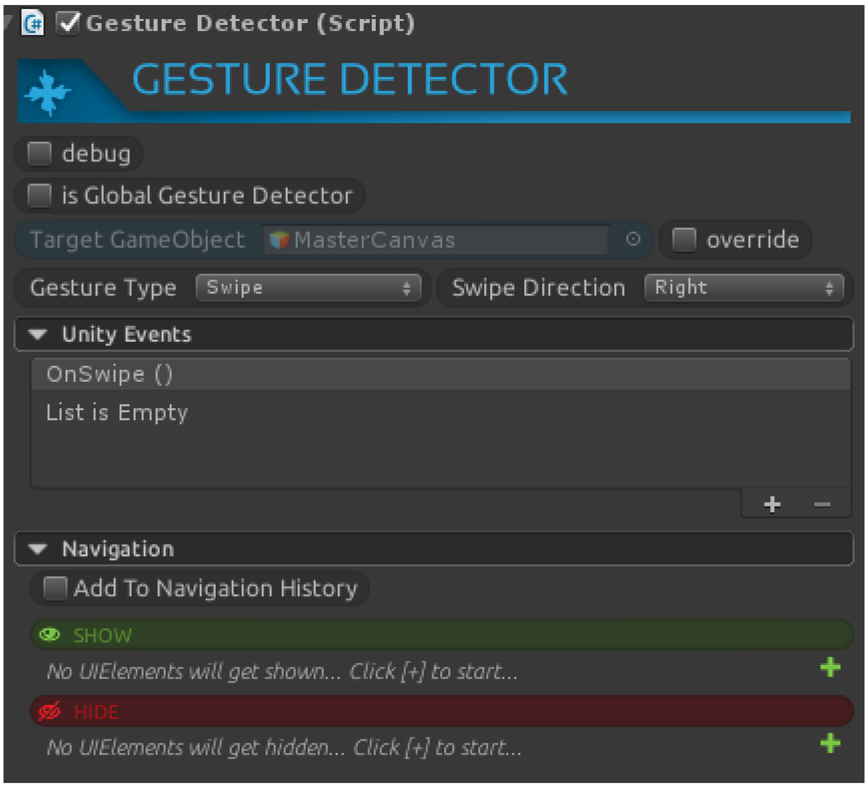
The Touch Manager is responsible for detecting touch gestures made on the target device. It will also simulate them, if in the Editor, by reacting to the mouse gestures.

It can detect the following gestures: Tap, Long Tap and Swipe

Tap Gesture: *is similar to a click, but it can be detected on any UI component, 2D object or 3D object.*
Long Tap Gesture: *is similar to a long click (or long press), but it can be detected on any UI component, 2D object or 3D object*
Swipe: *it can be detected in 4 directions (Left, Right, Up and Down) or it can be detected in 8 directions (Left, UpLeft, Up, UpRight, Right, DownRight, Down, DownLeft), on any UI component, 2D object or 3D object.*

debug: prints debug messages related to detected gestures at runtime
Min Swipe Length: the minimum swipe distance to be considered a swipe
Long Tap Duration: time period for a finger to be touching the target device, in order for the tap to be considered a long tap (long press)
use eight directions: sets if the TouchManager should detect swipes on eight or on four cardinal directions

Gesture Detector


Gesture Detector (Script)

GESTURE DETECTOR

debug

is Global Gesture Detector

Target GameObject MasterCanvas override

Gesture Type Swipe Swipe Direction Right

Unity Events

OnSwipe ()

List is Empty

Navigation

Add To Navigation History

SHOW

No UIElements will get shown... Click [+] to start...

HIDE

No UIElements will get hidden... Click [+] to start...

The Gesture Detector is an extension to the TouchManager. The TouchManager is the one that detects the touches (the gestures), but the Gesture Detector is the one that reacts to them, depending on its settings.

To be clear, the TouchManager detects a gesture and the GestureDetector reacts to it. Because the listening for gestures needs to happen in the Update() method, by having only the TouchManager as the only listener and all the Gesture Detectors getting fired by the TouchManager, we have an efficient setup.

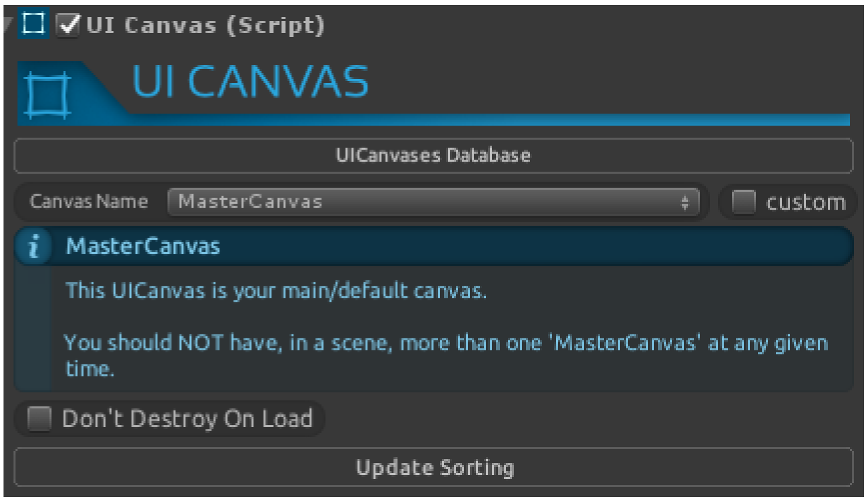
debug: prints debug messages related to detected gestures at runtime
is Global Gesture Detector: if true, it will trigger this Gesture Detector regardless of the targetGameObject
Target GameObject: only gestures performed on the target game object will trigger this Gesture Detector
override target: allows you to set another targetGameObject reference (in the Inspector); other than the gameObject this component is attached to
Gesture Type: the gesture type this Gesture Detector will react to
Swipe Direction: the swipe direction this Gesture Detector will react to; this works only if the gestureType is set to GestureType.Swipe

Only the 'active' UnityEvent is visible in the Inspector (in order to reduce clutter)
OnTap: UnityEvent invoked on tap
OnLongTap: UnityEvent invoked on long tap
OnSwipe: UnityEvent invoked on swipe (it has to be the proper swipe direction)

onTapAction: action invoked on tap
onLongTapAction: action invoked on long tap
onSwipeAction: action invoked on swipe (it has to be the proper swipe direction)

Only if the UINavigation is enabled, the Navigation options will be available.
Add to Navigation History: if there are any UIElements set to be shown or hidden, the action will be added to the Navigation History
SHOW: a list of pairs of element category and element name that will get shown when this Gesture Detector gets triggered
HIDE: a list of pairs of element category and element name that will get hidden when this Gesture Detector gets triggered

UICanvas



MasterCanvas

This UICanvas is your main/default canvas.

You should NOT have, in a scene, more than one 'MasterCanvas' at any given time.

The UICanvas main purpose is to give any UINotification, you want to show, a target container.

It is also used for other Inspector related actions that help you out in designing your UI. For example, when you want to create a new UIElement, it will get automatically parented to the selected UICanvas gameObject, or if no UI related gameObject is selected, it will get parented to the UICanvas named 'Master Canvas'.

You can consider the gameObject that has an UICanvas attached as your main UI container. Of course, you will also find a native Canvas component attached to the same gameObject, that is your rootCanvas.

Canvas Name: the name of this canvas (used in order to be able to identify it)
custom Canvas Name: is used by the custom inspector to allow you to type a canvas name instead of selecting it from the Canvas Names Database; this option can also be used to add a new canvas name and you can do that in four easy steps:

1. Enable custom name
2. Enter a new name
3. Disable custom name
4. A display dialog will appear asking you if you want to add the new name to the database (click yes)

Don't Destroy on Load: makes the UICanvas gameObject not get destroyed automatically when loading a scene; this allows your UI to persist across scenes and it is quite useful when working (loading/unloading) a lot of scenes

Fields

Name	*Description*
canvasName	The name of this canvas.
customCanvasName	Used by the custom inspector to allow you to type a canvas name instead of selecting it from the Canvas Names Database.
dontDestroyOnLoad	Makes this UICanvas gameObject not get destroyed automatically when loading a new scene.
MASTER_CANVAS_NAME	Default name given to a new canvas. The name is 'MasterCanvas' and you should have ONLY ONE per scene as this is considered your main/default canvas.

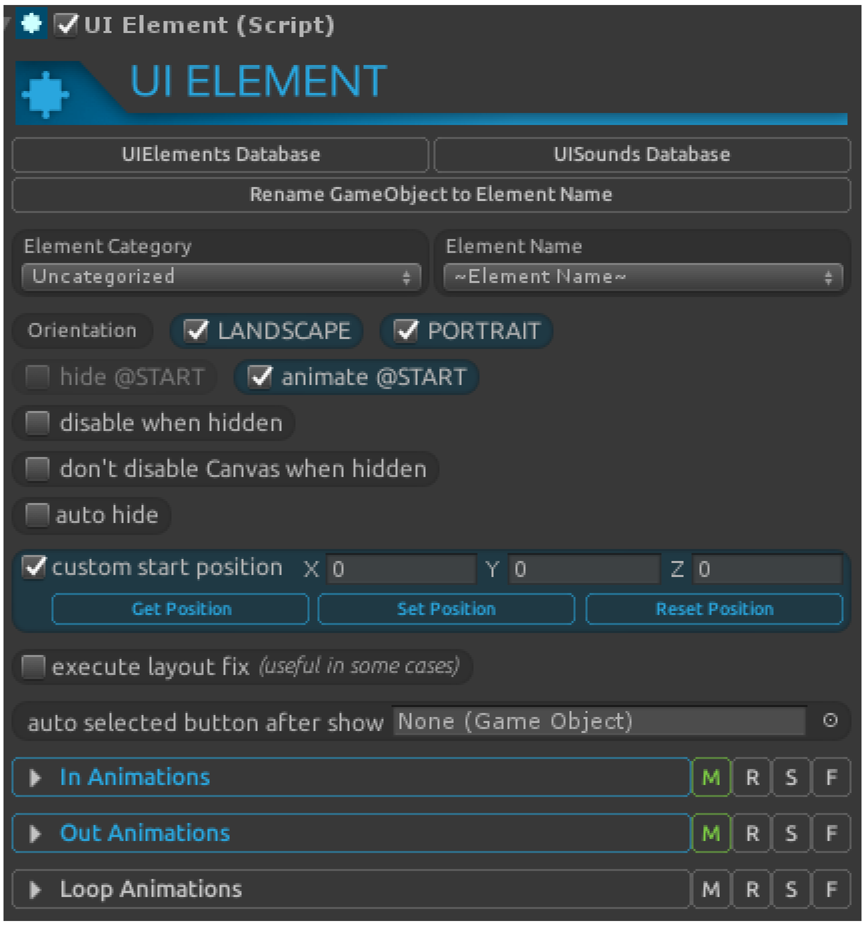
Properties

Name	*Description*
Canvas	Returns the Canvas component.
IsMasterCanvas	Returns true if this canvas name is 'MasterCanvas' and if it has been registered to the UIManager as the MasterCanvas
RectTransform	Returns the RectTransform component.

Methods

Name	*Description*
RegisterToUIManager()	Registers this UICanvas to the UIManager.
UnregisterFromUIManager()	Unregisteres this UICanvas from the UIManager.

UIElement



The UIElement is one of the core components that you will be using extensively. Because it is such an important component, it also comes with a lot of settings, but don't worry, once you know what they do you'll see that it's a piece of cake to configure.

All the settings have meaningful names and have been carefully arranged in an intuitive manner.

UIElements Database: opens the Control Panel window at the UIElements Tab
UISounds Database: opens the Control Panel window at the UISounds Tab
Rename GameObject to Element Name: renames the gameObject that this UIElement is attached to 'UIE – element name'; this button can be hidden from the Control Panel → Editor Settings Tab → UIElement

In order to be able to identify the UIElements, the system uses a category + name pair in order to trigger the In (show) or the Out (hide) animations. To this end, you can create your own categories that can contain any number or names. The categories and their respective names are available (as a dropdown list) right in the custom inspector.

Element Category: the element category
Element Name: the element name

You can use the dropdown lists, by selecting a category and then one of its names, or you can set a custom name. In order to do that, just select the ~Custom Name~ category and you'll be able to enter a custom name that has not been added to the database.

You can add a new name to any category (from the custom inspector) as follows:
1. Set the Element Category to ~Custom Name~
2. Enter a custom element name
3. Select the category that you want the custom element name to be added to
4. A display dialog will appear asking you if you want to add the new name to the database (click yes)

A new category can only be created from the Control Panel → UIElements Tab.

Orientation

If the Orientation Manager is enabled from the Control Panel → General Tab, then you will have the option of setting on which orientation is this UIElement meant to work (to be visible and for show/hide to work).
By default, an UIElement has both **LANDSCAPE** and **PORTRAIT** enabled. This means that the UIElement will be available (visible) on both orientations.
If you want to have different UIElements visible for each orientation you need do the following:

1. Create two UIElements that have the same element category and element name
2. Set the first UIElement to be enabled on **LANDSCAPE** and disabled on **PORTRAIT** (this will be visible when the orientation is in LANDSCAPE mode)
3. Set the second UIElement to be disabled on **LANDSCAPE** and enabled on **PORTRAIT** (this will be visible when the orientation is in PORTRAIT mode)
4. Design both UIElements contents for each orientation type
5. At runtime, the proper UIElement will be available, depending to the current orientation; and, on orientation change, they will be swapped one with the other as is the case

hide @START: hides the UIElement at start by initiating an instant hide (plays the Out animation in zero seconds)
animate @START: shows the UIElement at start by initiation a show (plays the In animation)

disable when hidden: disables the gameObject, this UIElement is attached to, by settings its active state to false, when it is not visible (on screen); we recommend that you enable this option only if there are scripts that you need stopped when the UIElement is hidden; by default the system disables the Canvas and Graphic Raycaster components, attached to the same gameObject as this UIElement, when hidden (this lowers the drawcalls)
don't disable Canvas when hidden: disables the automated disabling of the Canvas and Graphic Raycaster components, attached to the same gameObject as this UIElement, when hidden; do not enable this option unless you know what you are doing as this might increase the drawcalls count

auto hide: automatically issues a 'Hide' command for this UIElement after it has been shown and after the autoHideDelay duration has passed

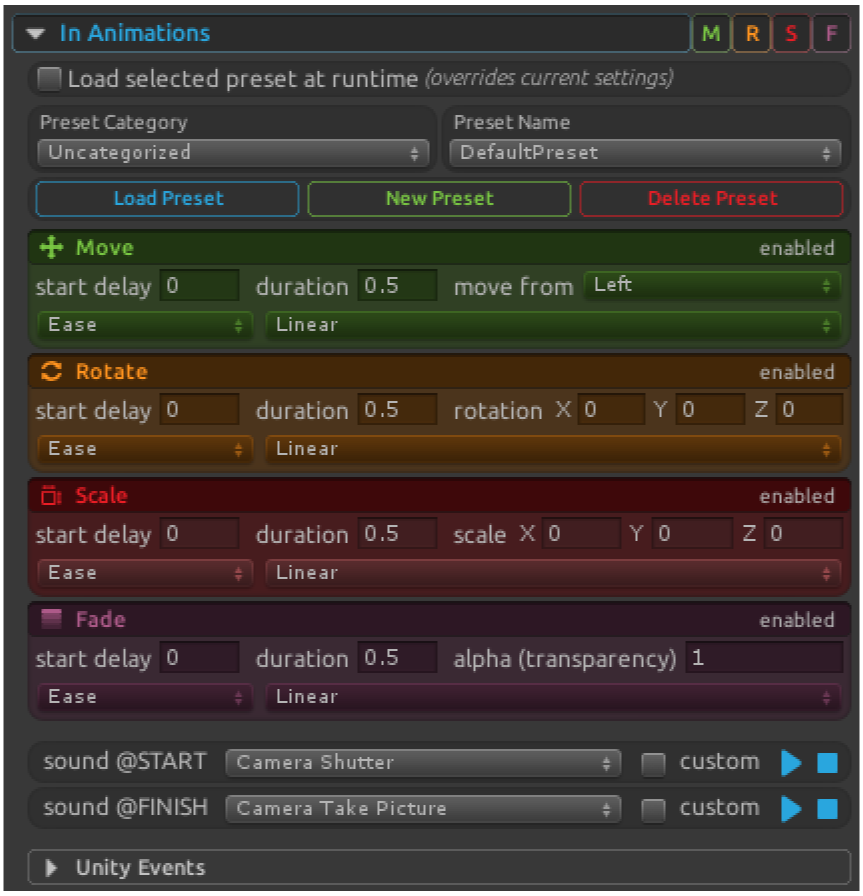
(use) custom start position: enables the option to use a custom start position for the In and Out animations
custom start (anchored) position: anchored position (used by the UI) that this UIElement uses to calculate its In and Out animations; if modified at runtime, it allows you to change the animations show and hide locations; this is also useful when designing your UI as you don't need to stack UIElements over another and it works as follows:

- create an UIElement in the position you want it to be visible at runtime
- enable the custom start position
- click the Get button to copy the current RectTransform anchored position values to the custom start position
- drag and drop the newly created UIElement anywhere in the scene
- after performing the aforementioned steps, at runtime, the UIElement will 'snap' to the custom start position for In or Out animations

execute layout fix: initiates a layout reset for this UIElement (may fix some issues)

auto selected button after show: the button (Game Object) that gets selected when this UIElement gets shown; if nothing is referenced, no button will get auto selected; default is set to null

In Animations



In Animations

M R S F

Load selected preset at runtime *(overrides current settings)*

Preset Category: Uncategorized

Preset Name: DefaultPreset

Load Preset | New Preset | Delete Preset

Move — enabled
start delay 0 duration 0.5 move from Left
Ease Linear

Rotate — enabled
start delay 0 duration 0.5 rotation X 0 Y 0 Z 0
Ease Linear

Scale — enabled
start delay 0 duration 0.5 scale X 0 Y 0 Z 0
Ease Linear

Fade — enabled
start delay 0 duration 0.5 alpha (transparency) 1
Ease Linear

sound @START Camera Shutter custom

sound @FINISH Camera Take Picture custom

Unity Events

The In Animations are a set of separate animations (Move, Rotate. Scale and Fade) that get played together when a 'Show' command has been issued. Because these animations are math based, any value change will result in a different animation.

Presets are different configuration sets that result in various animations.

There are two ways of setting up the animation. You can just select a preset, load its values (Load Preset) and either leave them like that or tweak them, or you can select a preset and set it to load at runtime (this will automatically load the preset values at runtime, overriding the settings from the Inspector).

Load selected preset at runtime: loads, at runtime, the animation preset that has the selected preset category and preset name; this will override any values set in the inspector

Preset Category: the selected animation preset category

Preset Name: the selected animation preset name (found in the selected preset category)

You can create new presets with ease, by creating new preset categories and preset names, right from the Inspector.

MOVE
start delay: start delay for the animation
duration: the duration of the animation
move from: the position this UIElement will animate from
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

ROTATE
start delay: start delay for the animation
duration: the duration of the animation
rotation: the rotation this UIElement will animate from
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

SCALE
start delay: start delay for the animation
duration: the duration of the animation
scale: the scale value this UIElement will animate from
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

FADE
start delay: start delay for the animation
duration: the duration of the animation
alpha (transparency): the alpha value this UIElement will animate from
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

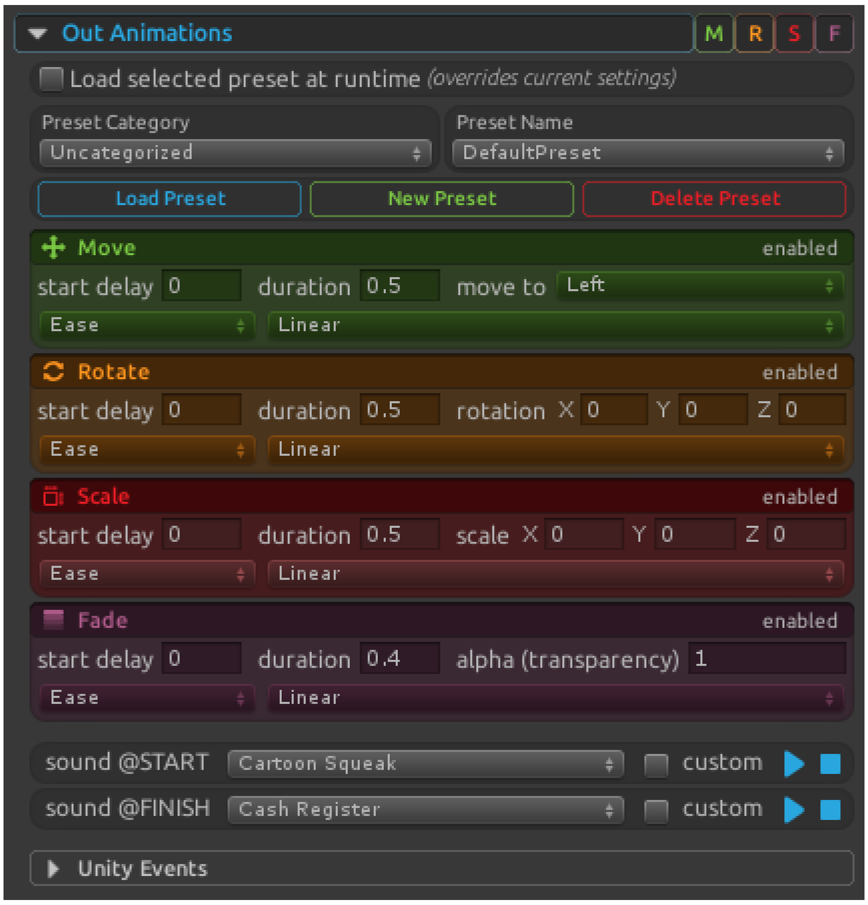
sound @START: the sound name of the UISound that gets played when the animation starts

sound @FINISH: the sound name of the UISound that gets played when the animation finished

You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

Unity Events
OnInAnimationStart: UnityEvent invoked when the animation starts
OnInAnimationsFinish: UnityEvent invoked when the animation finished

Out Animations



Load selected preset at runtime *(overrides current settings)*

Preset Category: Uncategorized
Preset Name: DefaultPreset

Move — enabled
start delay 0 duration 0.5 move to Left
Ease Linear

Rotate — enabled
start delay 0 duration 0.5 rotation X 0 Y 0 Z 0
Ease Linear

Scale — enabled
start delay 0 duration 0.5 scale X 0 Y 0 Z 0
Ease Linear

Fade — enabled
start delay 0 duration 0.4 alpha (transparency) 1
Ease Linear

sound @START Cartoon Squeak custom
sound @FINISH Cash Register custom

The Out Animations are a set of separate animations (Move, Rotate. Scale and Fade) that get played together when a 'Hide' command has been issued. Because these animations are math based, any value change will result in a different animation.

Presets are different configuration sets that result in various animations.

There are two ways of setting up the animation. You can just select a preset, load its values (Load Preset) and either leave them like that or tweak them, or you can select a preset and set it to load at runtime (this will automatically load the preset values at runtime, overriding the settings from the Inspector).

Load selected preset at runtime: loads, at runtime, the animation preset that has the selected preset category and preset name; this will override any values set in the inspector

Preset Category: the selected animation preset category

Preset Name: the selected animation preset name (found in the selected preset category)

You can create new presets with ease, by creating new preset categories and preset names, right from the Inspector.

MOVE
start delay: start delay for the animation
duration: the duration of the animation
move to: the position this UIElement will animate to
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

ROTATE
start delay: start delay for the animation
duration: the duration of the animation
rotation: the rotation this UIElement will animate to
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

SCALE
start delay: start delay for the animation
duration: the duration of the animation
scale: the scale value this UIElement will animate to
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

FADE
start delay: start delay for the animation
duration: the duration of the animation
alpha (transparency): the alpha value this UIElement will animate to
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

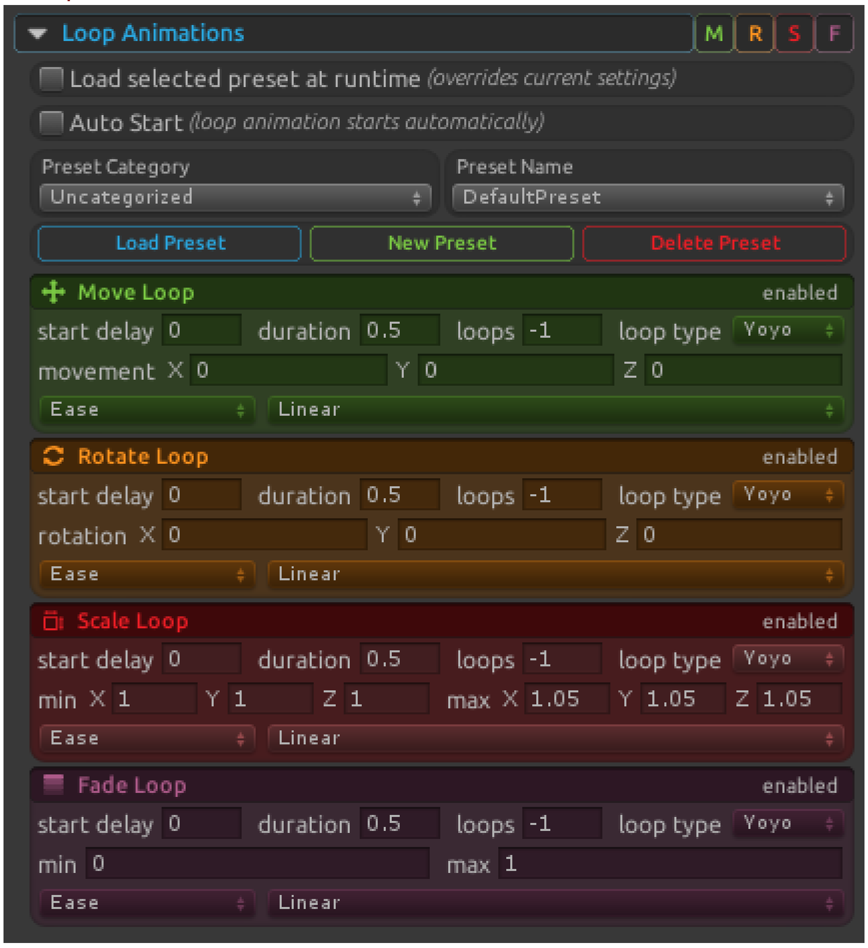
sound @START: the sound name of the UISound that gets played when the animation starts

sound @FINISH: the sound name of the UISound that gets played when the animation finished

You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

Unity Events
OnOutAnimationStart: UnityEvent invoked when the animation starts
OnOutAnimationsFinish: UnityEvent invoked when the animation finished

Loop Animations



Loop Animations

☐ Load selected preset at runtime *(overrides current settings)*

☐ Auto Start *(loop animation starts automatically)*

Preset Category: Uncategorized
Preset Name: DefaultPreset

Move Loop — enabled
start delay 0 duration 0.5 loops -1 loop type Yoyo
movement X 0 Y 0 Z 0
Ease Linear

Rotate Loop — enabled
start delay 0 duration 0.5 loops -1 loop type Yoyo
rotation X 0 Y 0 Z 0
Ease Linear

Scale Loop — enabled
start delay 0 duration 0.5 loops -1 loop type Yoyo
min X 1 Y 1 Z 1 max X 1.05 Y 1.05 Z 1.05
Ease Linear

Fade Loop — enabled
start delay 0 duration 0.5 loops -1 loop type Yoyo
min 0 max 1
Ease Linear

The Loop Animations are a set of separate animations (Move, Rotate. Scale and Fade) that get played together after an In animation finished and get stopped before an Out animation starts. The auto start option will start playing the loop animation without waiting for an In animation to finish. Because these animations are math based, any value change will result in a different animation.

Presets are different configuration sets that result in various animations.

There are two ways of setting up the animation. You can just select a preset, load its values (Load Preset) and either leave them like that or tweak them, or you can select a preset and set it to load at runtime (this will automatically load the preset values at runtime, overriding the settings from the Inspector).

Load selected preset at runtime: loads, at runtime, the animation preset that has the selected preset category and preset name; this will override any values set in the inspector

Auto Start: sets if the animation should start from the get go (after being initialized) or on demand (after an In animation finished playing)

Preset Category: the selected animation preset category

Preset Name: the selected animation preset name (found in the selected preset category)

You can create new presets with ease, by creating new preset categories and preset names, right from the Inspector.

MOVE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)

loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
movement: movement is calculated startAnchoredPosition-movement for min and startAnchoredPosition+movment for max
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

ROTATE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
rotation: rotation is calculated startRotation-rotation for min and startRotation+rotation for max
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

SCALE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
min: the minimum values for the scale factor of the scale loop animation (default: 1)
max: the maximum values for the scale factor of the scale loop animation (default: 1.05)
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

FADE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
min: the minimum alpha value for the fade animation loop (default: 0)
max: the maximum alpha value for the fade animation loop (default: 1)
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

Fields

Name	*Description*
animateAtStart	Animate the UIElement at runtime at start. Initiates a Show, thus playing an In animation. Default is set to false.
autoHide	Automatically issue a 'Hide' command for this UIElement after being shown and after the autoHideDelay duration has passed.
autoHideDelay	If this UIElement is set to automatically autoHide after being shown. A 'Hide' command will get issued by this UIElement, after the set delay duration.
autoRegister	Used by the UINotification. If this element is linked to a notification, then the notification should handle its registration process in order to use an auto generated name. Do not change this value yourself.
customInAnimationsSoundAtFinish	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customInAnimationsSoundAtStart	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOutAnimationsSoundAtFinish	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOutAnimationsSoundAtStart	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customStartAnchoredPosition	The custom anchored position that this UIElement comes from or goes to when an In or Out animation is played. You can use this in code to customize on the fly this position.
disableWhenHidden	Disables this UIElement when it is not visible (it is hidden) by setting it's active state to false. Use this only if you have scripts that you need to disable. Otherwise you don't need it as the system handles the drawcalls in an efficient manner.
dontDisableCanvasWhenHidden	This will disable the optimization that disables the Canvas and GraphicRaycaster when the UIElement is hidden. Do not enable this unless you know what you are doing as, when set to TRUE, this will increase your draw calls.
elementCategory	The category this element name belongs to. The category is important when showing or hiding an UIElement as it is considered.
elementName	The name of this element. The name is important when showing or hiding an UIElement as it is considered.
executeLayoutFix	This fixes a very strange issue inside Unity. When setting a VerticalLayoutGroup or a HorizontalLayoutGroup, the Image bounds get moved (the image appears in a different place). If you have this issue, just set this to true. Default is set to false. If you are curious about what this does, look at the ExecuteLayoutFix method.
inAnimations	In Animation Settings
inAnimationsPresetCategoryName	Out Animations Preset Category Name
inAnimationsPresetName	Out Animations Preset Name
inAnimationsSoundAtFinish	The sound name of the sound that gets played when the in animations finished.
inAnimationsSoundAtStart	The sound name of the sound that gets played when the in animations start.
isVisible	Keeps track if this UIElement is visible or not. Do not change this value yourself.
LANDSCAPE	Use this UIElement for LANDSCAPE orientation. Default is true. If Orientation Manager is disabled, this setting does nothing.
linkedToNotification	If this UIElement is linked to an UINotification then it will have an auto-generated element name. Do not change this value yourself.
loadInAnimationsPresetAtRuntime	Should the system load, at runtime, the Animation Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadLoopAnimationsPresetAtRuntime	Should the system load, at runtime, the Loop Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOutAnimationsPresetAtRuntime	Should the system load, at runtime, the Animation Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
LOOP_ANIMATIONS_ID	This is an extra id tag given to the tweener in order to locate the proper tween that manages the loop animations.
loopAnimations	Loop Animation Settings
loopAnimationsPresetCategoryName	Loop Animations Preset Category Name
loopAnimationsPresetName	Loop Animations Preset Name
OnInAnimationsFinish	UnityEvent invoked when In animations finished.
OnInAnimationsStart	UnityEvent invoked when In animations start.
OnOutAnimationsFinish	UnityEvent invoked when Out animations finished.
OnOutAnimationsStart	UnityEvent invoked when Out animations start.
outAnimations	Out Animation Settings
outAnimationsPresetCategoryName	Out Animations Preset Category Name

outAnimationsPresetName	Out Animations Preset Name
outAnimationsSoundAtFinish	The sound name of the sound that gets played when the out animations finished.
outAnimationsSoundAtStart	The sound name of the sound that gets played when the out animations start.
PORTRAIT	Use this UIElement for PORTRAIT orientation. Default is true. If Orientation Manager is disabled, this setting does nothing.
selectedButton	The button that gets selected when this UIElement gets shown; if null then no button will get auto selected. Default is set to null.
startHidden	Hide the UIElement at runtime at start. Initiates an instant Hide. Default is set to false.
useCustomStartAnchoredPosition	Should this UIElement come from or go to a set custom position every time an In or Out animation is played? Default is set to false.

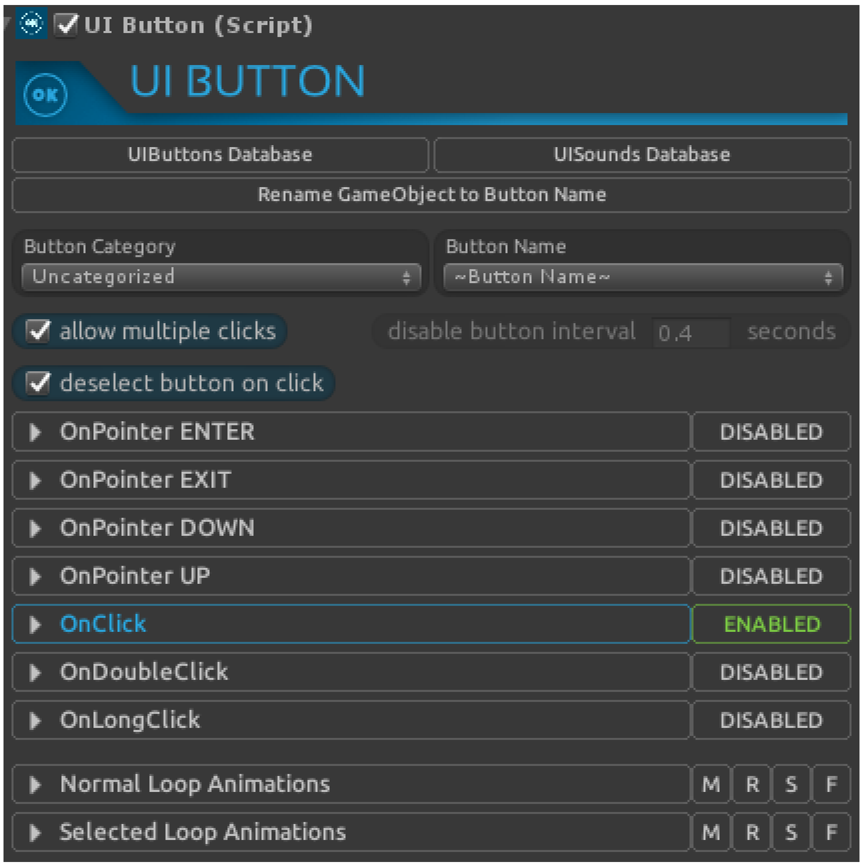
Properties

Name	*Description*
Canvas	Returns the Canvas component.
CanvasGroup	Returns the CanvasGroup component.
GraphicRaycaster	Returns the GraphicRaycaster component.
InAnimationsEnabled	Returns true if at least one In animation is enabled. This means that if either move or rotate or scale or fade are enabled it will return true and false otherwise.
LoopAnimationsEnabled	Returns true if at least one Loop animation is enabled. This means that if either move or rotate or scale or fade are enabled it will return true and false otherwise.
OutAnimationsEnabled	Returns true if at least one Out animation is enabled. This means that if either move or rotate or scale or fade are enabled it will return true and false otherwise.
RectTransform	Returns the RectTransform component.

Methods

Name	*Description*
AutoHide(bool, float)	Initiates an automated auto hide after the UIElement has been shown.
CancelAutoHide()	If an AutoHide has been initiated. This method will cancel it.
Hide(bool)	Hides the element.
Hide(bool, bool)	Hides the element.
RegisterToUIManager()	Registers this UIElement to the UIManager.
Show(bool)	Shows the element.
UnregisterFromUIManager()	Unregisters this UIElement from the UIManager.

UIButton


UI Button (Script)

UI BUTTON

UIButtons Database | UISounds Database

Rename GameObject to Button Name

Button Category: Uncategorized

Button Name: ~Button Name~

☑ allow multiple clicks — disable button interval 0.4 seconds

☑ deselect button on click

OnPointer ENTER	DISABLED
OnPointer EXIT	DISABLED
OnPointer DOWN	DISABLED
OnPointer UP	DISABLED
OnClick	ENABLED
OnDoubleClick	DISABLED
OnLongClick	DISABLED
Normal Loop Animations	M R S F
Selected Loop Animations	M R S F

The UIButton is another core component that you will be using extensively. Because it is such an important component, it also comes with a lot of settings, but don't worry, once you know what they do you'll see that it's a piece of cake to configure.

All the settings have meaningful names and have been carefully arranged in an intuitive manner.

UIButtons Database: opens the Control Panel window at the UIButtons Tab
UISounds Database: opens the Control Panel window at the UISounds Tab
Rename GameObject to Button Name: renames the gameObject that this UIButton is attached to 'UIB – button name'; this button can be hidden from the Control Panel → Editor Settings Tab → UIButton

In order to be able to identify the UIButton that has been clicked, the system listens for the 'button name'. To help you out manage all the button names, the UIButtons database allows you to sort all button names into categories. To this end, you can create your own categories that can contain any number or names. The categories and their respective names are available (as a dropdown list) right in the custom inspector.

Button Category: the button category (set in the database)
Button Name: the element name (used to identify what button was clicked)

You can use the dropdown lists, by selecting a category and then one of its names, or you can set a custom name. In order to do that, just select the ~Custom Name~ category and you'll be able to enter a custom name that has not been added to the database.

You can add a new name to any category (from the custom inspector) as follows:

1. Set the Button Category to ~Custom Name~
2. Enter a custom button name
3. Select the category that you want the custom button name to be added to
4. A display dialog will appear asking you if you want to add the new name to the database (click yes)

A new category can only be created from the Control Panel → UIButtons Tab.

allow multiple clicks: if disabled, after each click, the button will get disabled for a set interval (disableButtonInterval); by default, the user is allowed to press the button multiple times without getting disabled (default: true)

disable button interval: is the duration that this button will get disabled after each click

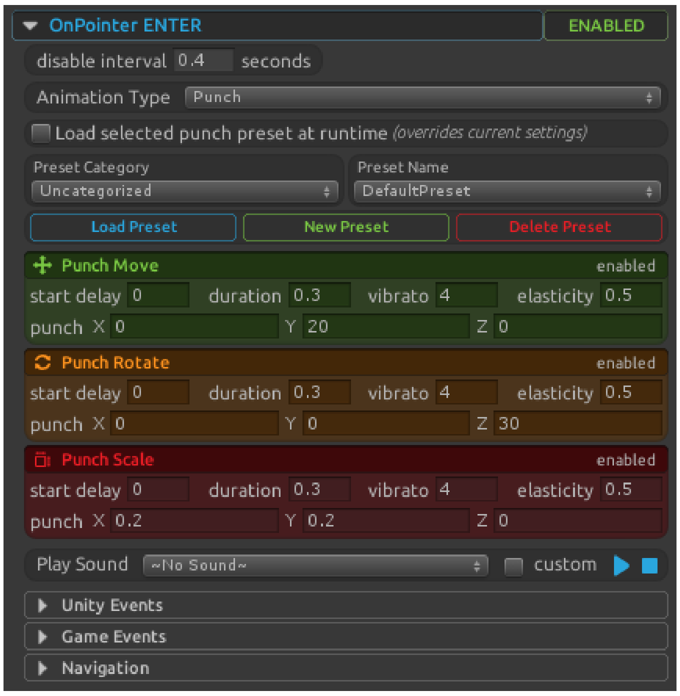
deselect button on click: makes the button get deselected after each click; this is useful if you do not want the button to get selected after each click (maybe not to trigger the selected loop animation)

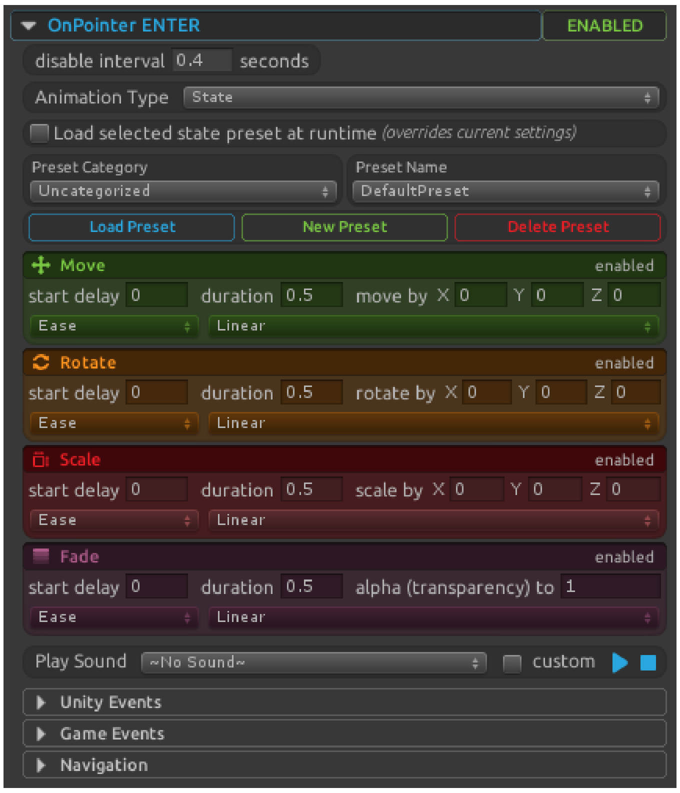
The OnPointer ENTER, OnPointer EXIT, OnPointer DOWN, OnPointer UP, OnClick, OnDoubleClick and OnLongClick functionalities can execute only one type of animation, either a punch animation or a state animation. They work in different ways and, if you are a beginner, it is recommended that you start by using only punch animations, as state animations require being set up on two opposing functionalities in order to reset the button's properties.

BUTTON ANIMATION TYPES	
PUNCH ANIMATION this type of animation means that the position (move), rotation (rotate) and size (scale) will return to their initial values after the animation	**STATE ANIMATION** this type of animation means that the position (move), rotation (rotate), size (scale) and alpha (fade) will NOT return to their initial values after the animation; you need to have another animation (on another functionality) that resets the button's values
Load selected punch preset at runtime: loads, at runtime, the punch animation preset that has the selected preset category and preset name; this will override any values set in the inspector	**Load selected state preset at runtime:** loads, at runtime, the state animation preset that has the selected preset category and preset name; this will override any values set in the inspector
Preset Category: the selected punch animation preset category **Preset Name** the selected punch animation preset name (found in the selected preset category) You can create new presets with ease, by creating new preset categories and preset names, right from the Inspector.	**Preset Category:** the selected state animation preset category **Preset Name** the selected state animation preset name (found in the selected preset category) You can create new presets with ease, by creating new preset categories and preset names, right from the Inspector.
PUNCH MOVE **start delay:** start delay for the animation **duration:** the duration of the animation **vibrato:** indicates how much will the punch vibrate **elasticity: r**epresents how much (0 to 1) the vector will go beyond the starting values when bouncing backwards. 1 creates a full oscillation between the punch position and the opposite position, while 0 oscillates only between the punch position and the start position. **punch:** the punch strength (added to the Transform's current position)	**MOVE** **start delay:** start delay for the animation **duration:** the duration of the animation **move by:** the position this UIButton will animate relative to the start position (if set to zero – it will reset to the start position) **ease (type):** select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time **ease:** if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net **animation curve:** if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time
PUNCH ROTATE **start delay:** start delay for the animation **duration:** the duration of the animation **vibrato:** indicates how much will the punch vibrate **elasticity: r**epresents how much (0 to 1) the vector will go beyond the starting values when bouncing backwards. 1 creates a full oscillation between the punch rotation and the opposite rotation, while 0 oscillates only between the punch rotation and the start rotation. **punch:** the punch strength (added to the Transform's current rotation)	**ROTATE** **start delay:** start delay for the animation **duration:** the duration of the animation **rotate by:** the rotation this UIButton will animate relative to the start rotation (if set to zero – it will reset to the start rotation) **ease (type):** select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time **ease:** if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of

	a parameter over time; to see how default ease curves look, check out easings.net **animation curve:** if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time
PUNCH SCALE **start delay:** start delay for the animation **duration:** the duration of the animation **vibrato:** indicates how much will the punch vibrate **elasticity: r**epresents how much (0 to 1) the vector will go beyond the starting values when bouncing backwards. 1 creates a full oscillation between the punch scale and the opposite scale, while 0 oscillates only between the punch scale and the start scale. **punch:** the punch strength (added to the Transform's current scale)	**SCALE** **start delay:** start delay for the animation **duration:** the duration of the animation **scale by:** the scale value this UIButton will animate relative to the start scale (if set to zero – it ill reset to the start scale) **ease (type):** select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time **ease:** if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net **animation curve:** if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time
---	**FADE** **start delay:** start delay for the animation **duration:** the duration of the animation **alpha (transparency) to:** the alpha value this UIButton will animate to **ease (type):** select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time **ease:** if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net **animation curve:** if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

OnPointer ENTER


OnPointer ENTER — ENABLED
disable interval 0.4 seconds
Animation Type: Punch
Load selected punch preset at runtime *(overrides current settings)*
Preset Category: Uncategorized — Preset Name: DefaultPreset
Load Preset | New Preset | Delete Preset
Punch Move — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 20 Z 0
Punch Rotate — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 0 Z 30
Punch Scale — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0.2 Y 0.2 Z 0
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation


OnPointer ENTER — ENABLED
disable interval 0.4 seconds
Animation Type: State
Load selected state preset at runtime *(overrides current settings)*
Preset Category: Uncategorized — Preset Name: DefaultPreset
Load Preset | New Preset | Delete Preset
Move — enabled
start delay 0 duration 0.5 move by X 0 Y 0 Z 0
Ease: Linear
Rotate — enabled
start delay 0 duration 0.5 rotate by X 0 Y 0 Z 0
Ease: Linear
Scale — enabled
start delay 0 duration 0.5 scale by X 0 Y 0 Z 0
Ease: Linear
Fade — enabled
start delay 0 duration 0.5 alpha (transparency) to 1
Ease: Linear
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation

The OnPointer ENTER functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer enters its boundaries.

disable interval: time interval when the functionality gets disabled after it has been triggered; it is useful in certain cases

Animation Type: sets what type of animation this functionality uses; it can be a Punch Animation (see left image) or a State Animation (see right image)

Since you can have/use only one animation type per functionality, if you set the animation type to Punch, only the settings for Punch Animations will be available (visible) and, if you see the animation type to State, only the settings for the State Animations will be available(visible).

Play Sound: the sound name of the UISound that gets played when this functionality gets triggered

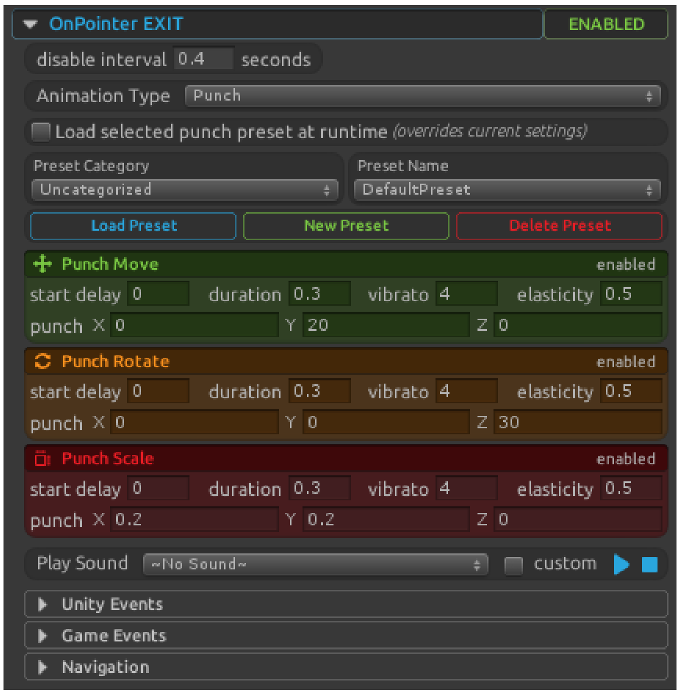
You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

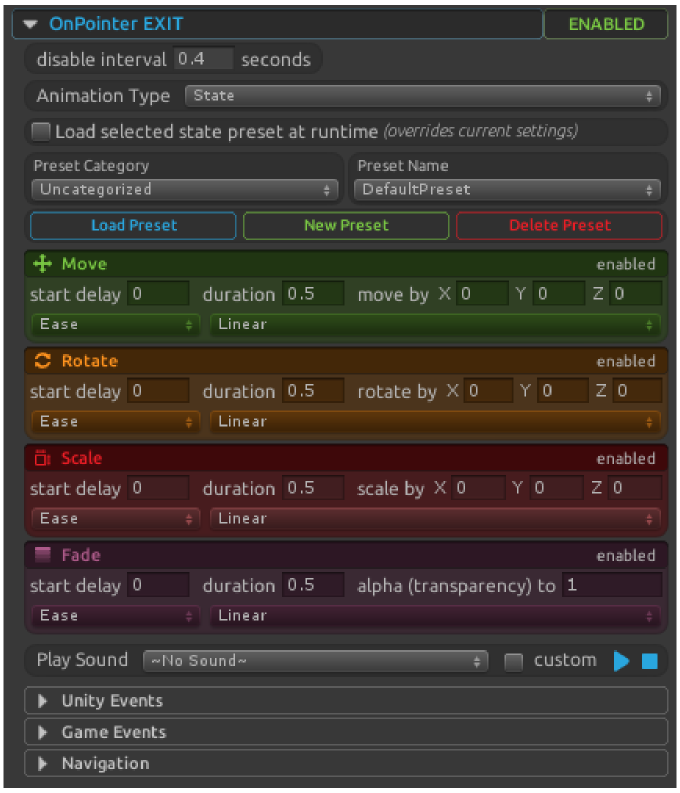
Unity Events
OnPointerEnter: UnityEvent invoked when the functionality gets triggered

Game Events
List of Game Events sent when the functionality gets triggered.

Navigation
If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden.
Add to Navigation History: if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back').
Show: is a list of pairs of element category & element name that will get shown when this functionality gets triggered
Hide: is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

OnPointer EXIT


OnPointer EXIT — ENABLED
disable interval 0.4 seconds
Animation Type: Punch
Load selected punch preset at runtime *(overrides current settings)*
Preset Category: Uncategorized — Preset Name: DefaultPreset
Load Preset | New Preset | Delete Preset
Punch Move — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 20 Z 0
Punch Rotate — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 0 Z 30
Punch Scale — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0.2 Y 0.2 Z 0
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation


OnPointer EXIT — ENABLED
disable interval 0.4 seconds
Animation Type: State
Load selected state preset at runtime *(overrides current settings)*
Preset Category: Uncategorized — Preset Name: DefaultPreset
Load Preset | New Preset | Delete Preset
Move — enabled
start delay 0 duration 0.5 move by X 0 Y 0 Z 0
Ease Linear
Rotate — enabled
start delay 0 duration 0.5 rotate by X 0 Y 0 Z 0
Ease Linear
Scale — enabled
start delay 0 duration 0.5 scale by X 0 Y 0 Z 0
Ease Linear
Fade — enabled
start delay 0 duration 0.5 alpha (transparency) to 1
Ease Linear
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation

The OnPointer EXIT functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer exits its boundaries.

disable interval: time interval when the functionality gets disabled after it has been triggered; it is useful in certain cases

Animation Type: sets what type of animation this functionality uses; it can be a Punch Animation (see left image) or a State Animation (see right image)

Since you can have/use only one animation type per functionality, if you set the animation type to Punch, only the settings for Punch Animations will be available (visible) and, if you see the animation type to State, only the settings for the State Animations will be available(visible).

Play Sound: the sound name of the UISound that gets played when this functionality gets triggered

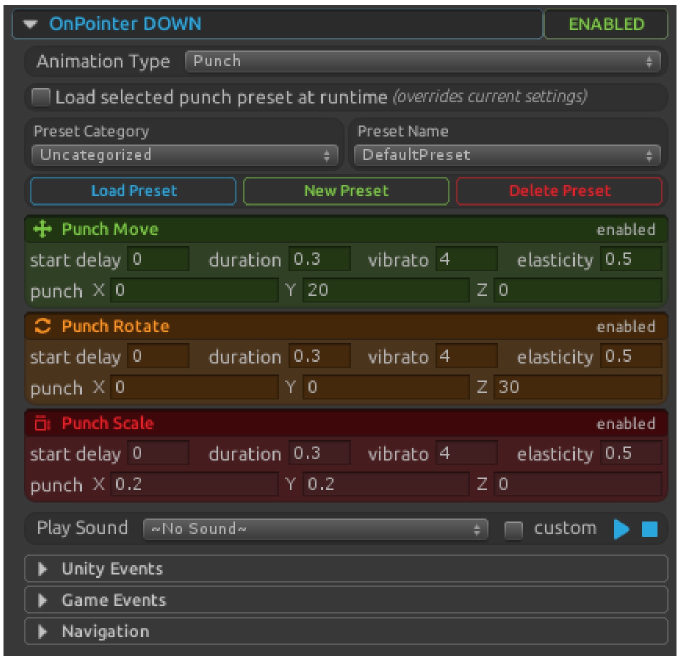
You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

Unity Events
OnPointerExit: UnityEvent invoked when the functionality gets triggered

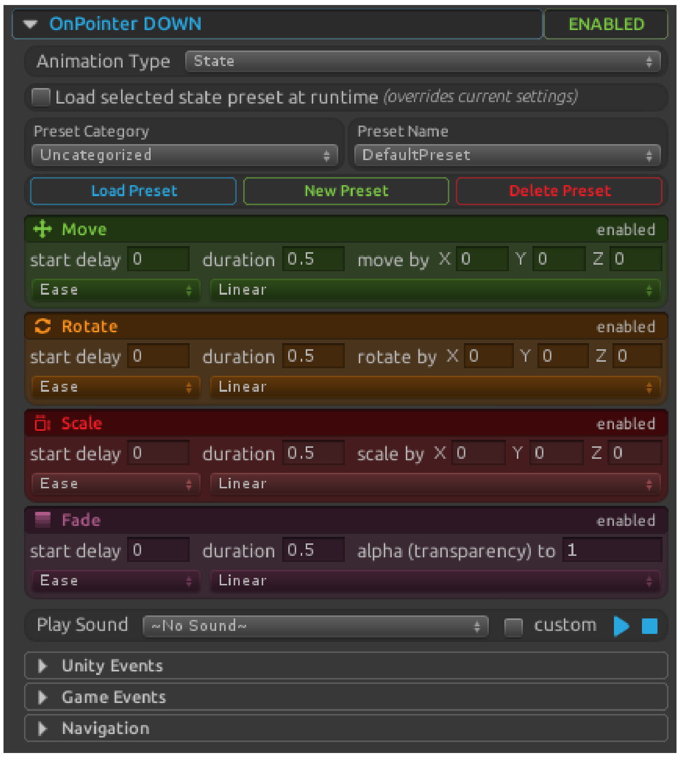
Game Events
List of Game Events sent when the functionality gets triggered.

Navigation
If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden.
Add to Navigation History: if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back').
Show: is a list of pairs of element category & element name that will get shown when this functionality gets triggered
Hide: is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

OnPointer DOWN



OnPointer DOWN — ENABLED
Animation Type: Punch
Load selected punch preset at runtime *(overrides current settings)*
Preset Category: Uncategorized — Preset Name: DefaultPreset
Load Preset | New Preset | Delete Preset
Punch Move — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 20 Z 0
Punch Rotate — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 0 Z 30
Punch Scale — enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0.2 Y 0.2 Z 0
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation



OnPointer DOWN — ENABLED
Animation Type: State
Load selected state preset at runtime *(overrides current settings)*
Preset Category: Uncategorized — Preset Name: DefaultPreset
Load Preset | New Preset | Delete Preset
Move — enabled
start delay 0 duration 0.5 move by X 0 Y 0 Z 0
Ease — Linear
Rotate — enabled
start delay 0 duration 0.5 rotate by X 0 Y 0 Z 0
Ease — Linear
Scale — enabled
start delay 0 duration 0.5 scale by X 0 Y 0 Z 0
Ease — Linear
Fade — enabled
start delay 0 duration 0.5 alpha (transparency) to 1
Ease — Linear
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation

The OnPointer DOWN functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer is down and over its boundaries.

Animation Type: sets what type of animation this functionality uses; it can be a Punch Animation (see left image) or a State Animation (see right image)

Since you can have/use only one animation type per functionality, if you set the animation type to Punch, only the settings for Punch Animations will be available (visible) and, if you see the animation type to State, only the settings for the State Animations will be available(visible).

Play Sound: the sound name of the UISound that gets played when this functionality gets triggered

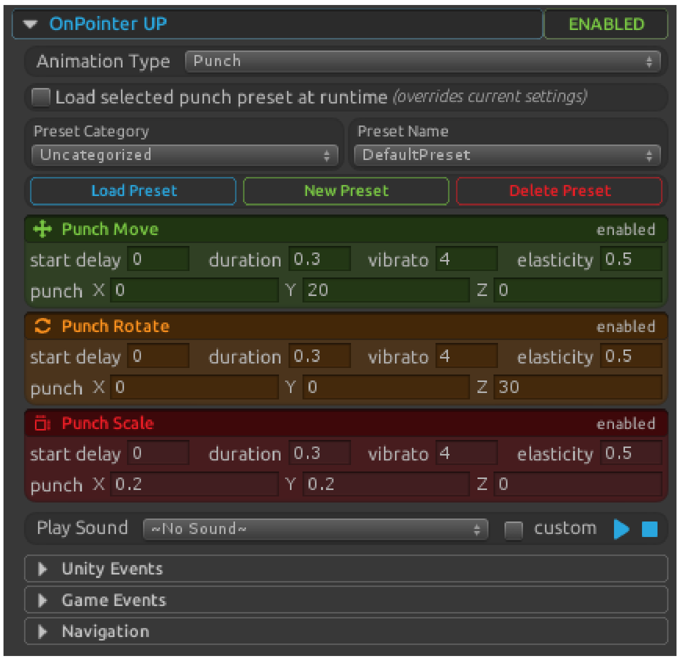
You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

Unity Events
OnPointerDown: UnityEvent invoked when the functionality gets triggered

Game Events
List of Game Events sent when the functionality gets triggered.

Navigation
If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden.
Add to Navigation History: if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back').
Show: is a list of pairs of element category & element name that will get shown when this functionality gets triggered
Hide: is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

OnPointer UP



OnPointer UP — ENABLED

Animation Type: Punch

Load selected punch preset at runtime *(overrides current settings)*

Preset Category: Uncategorized | Preset Name: DefaultPreset

Load Preset | New Preset | Delete Preset

Punch Move — enabled
start delay 0 | duration 0.3 | vibrato 4 | elasticity 0.5
punch X 0 | Y 20 | Z 0

Punch Rotate — enabled
start delay 0 | duration 0.3 | vibrato 4 | elasticity 0.5
punch X 0 | Y 0 | Z 30

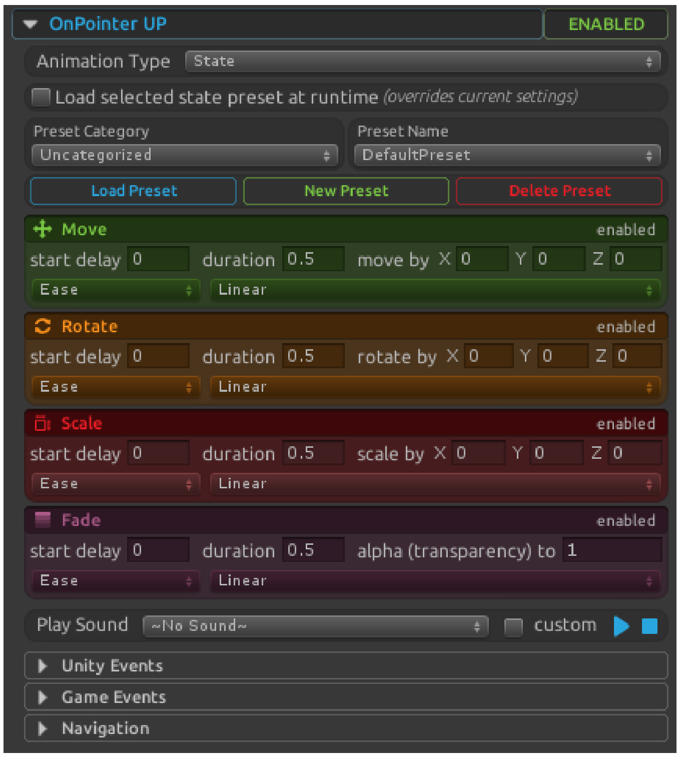
Punch Scale — enabled
start delay 0 | duration 0.3 | vibrato 4 | elasticity 0.5
punch X 0.2 | Y 0.2 | Z 0

Play Sound: ~No Sound~ | custom

Unity Events

Game Events

Navigation



OnPointer UP — ENABLED

Animation Type: State

Load selected state preset at runtime *(overrides current settings)*

Preset Category: Uncategorized | Preset Name: DefaultPreset

Load Preset | New Preset | Delete Preset

Move — enabled
start delay 0 | duration 0.5 | move by X 0 | Y 0 | Z 0
Ease | Linear

Rotate — enabled
start delay 0 | duration 0.5 | rotate by X 0 | Y 0 | Z 0
Ease | Linear

Scale — enabled
start delay 0 | duration 0.5 | scale by X 0 | Y 0 | Z 0
Ease | Linear

Fade — enabled
start delay 0 | duration 0.5 | alpha (transparency) to 1
Ease | Linear

Play Sound: ~No Sound~ | custom

Unity Events

Game Events

Navigation

The OnPointer UP functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer is up (happens only after OnPointer DOWN).

Animation Type: sets what type of animation this functionality uses; it can be a Punch Animation (see left image) or a State Animation (see right image)

Since you can have/use only one animation type per functionality, if you set the animation type to Punch, only the settings for Punch Animations will be available (visible) and, if you see the animation type to State, only the settings for the State Animations will be available(visible).

Play Sound: the sound name of the UISound that gets played when this functionality gets triggered

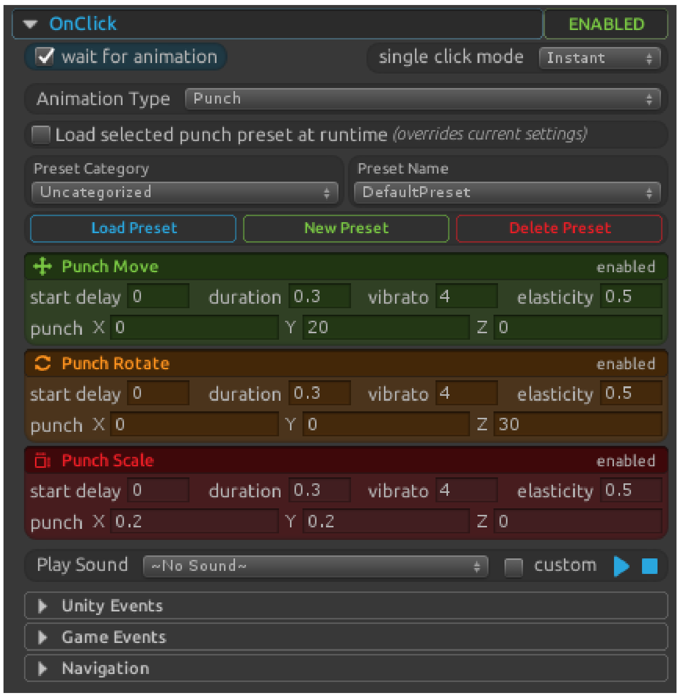
You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

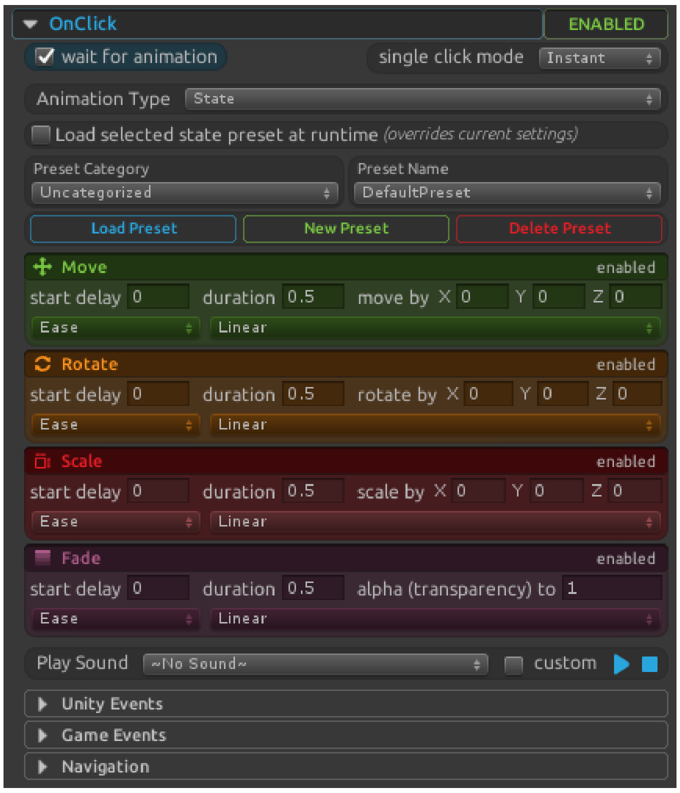
Unity Events
OnPointerUp: UnityEvent invoked when the functionality gets triggered

Game Events
List of Game Events sent when the functionality gets triggered.

Navigation
If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden.
Add to Navigation History: if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back').
Show: is a list of pairs of element category & element name that will get shown when this functionality gets triggered
Hide: is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

OnClick


OnClick ENABLED
wait for animation ✓ single click mode Instant
Animation Type Punch
Load selected punch preset at runtime *(overrides current settings)*
Preset Category Uncategorized Preset Name DefaultPreset
Load Preset New Preset Delete Preset
Punch Move enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 20 Z 0
Punch Rotate enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 0 Z 30
Punch Scale enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0.2 Y 0.2 Z 0
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation


OnClick ENABLED
wait for animation ✓ single click mode Instant
Animation Type State
Load selected state preset at runtime *(overrides current settings)*
Preset Category Uncategorized Preset Name DefaultPreset
Load Preset New Preset Delete Preset
Move enabled
start delay 0 duration 0.5 move by X 0 Y 0 Z 0
Ease Linear
Rotate enabled
start delay 0 duration 0.5 rotate by X 0 Y 0 Z 0
Ease Linear
Scale enabled
start delay 0 duration 0.5 scale by X 0 Y 0 Z 0
Ease Linear
Fade enabled
start delay 0 duration 0.5 alpha (transparency) to 1
Ease Linear
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation

The OnClick functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer executed a click, or a finger tap happened over it.

wait for animation: if enabled, the button action and game events are sent after the set animation has finished playing; this is useful if you want to be sure the uses see the button animation

single click mode: determines if on click is triggered instantly or after it checks if it's a double click or not; depending on your use case, you might need the Instant or Delayed mode; default is set to Instant

Single Click Mode – Instant: The click will get registered instantly without checking if it's a double click or not. This is the normal behavior of a single click in any OS. Use this if you want to make sure a single click will get executed before a double click (dual actions). Usage example: SingleClick - selects, DoubleClick - executes an action)

Single Click Mode – Delayed: The click will get registered after checking if it's a double click or not. If it's a double click, the single click will not get triggered. Use this if you want to make sure the user does not execute a single click before a double click. The downside is that there is a delay when executing the single click (the delay is the double click register interval), so make sure you take that into account

Animation Type: sets what type of animation this functionality uses; it can be a Punch Animation (see left image) or a State Animation (see right image)

Since you can have/use only one animation type per functionality, if you set the animation type to Punch, only the settings for Punch Animations will be available (visible) and, if you see the animation type to State, only the settings for the State Animations will be available(visible).

Play Sound: the sound name of the UISound that gets played when this functionality gets triggered

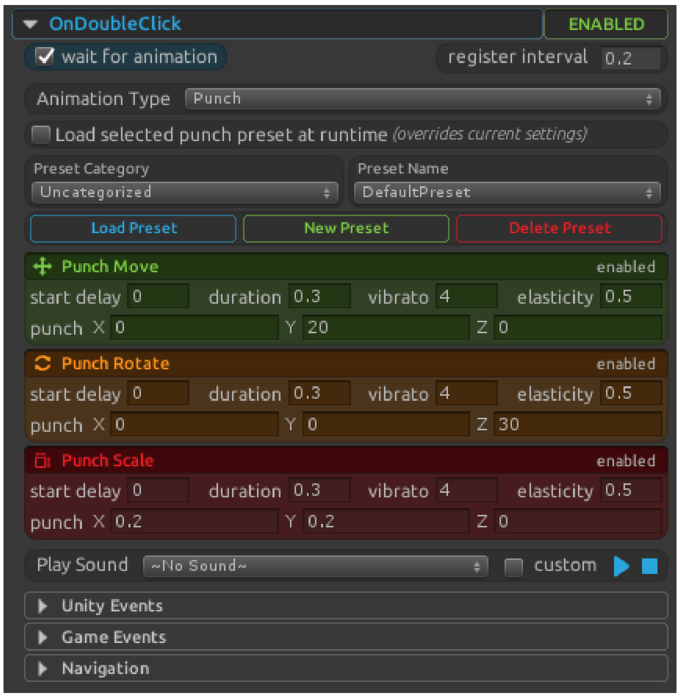
You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

Unity Events
OnClick: UnityEvent invoked when the functionality gets triggered

Game Events
List of Game Events sent when the functionality gets triggered.

Navigation
If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden.
Add to Navigation History: if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back').
Show: is a list of pairs of element category & element name that will get shown when this functionality gets triggered
Hide: is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

OnDoubleClick

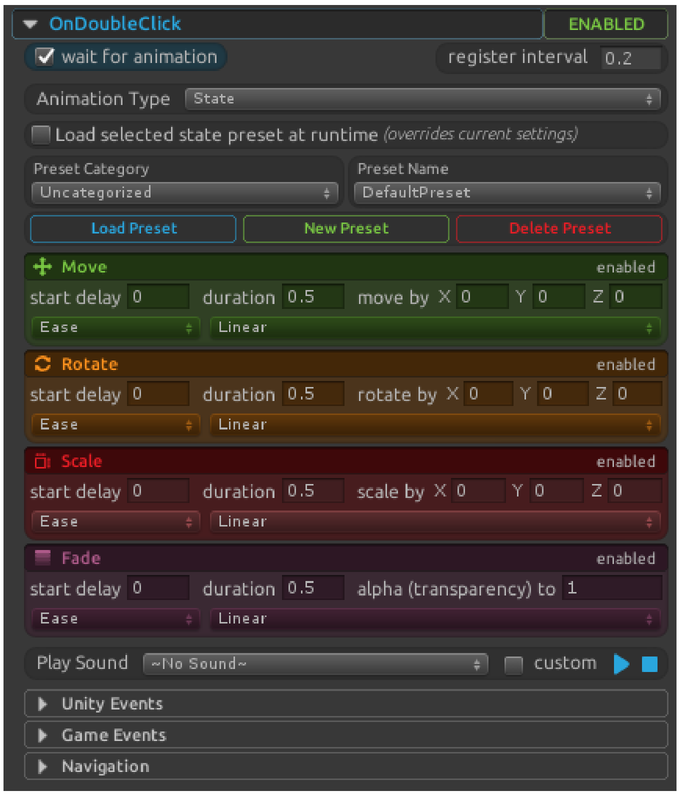

OnDoubleClick ENABLED
wait for animation ✓ | register interval 0.2
Animation Type: Punch
Load selected punch preset at runtime *(overrides current settings)*
Preset Category: Uncategorized | Preset Name: DefaultPreset
Load Preset | New Preset | Delete Preset

Punch Move — enabled
start delay 0 | duration 0.3 | vibrato 4 | elasticity 0.5
punch X 0 | Y 20 | Z 0

Punch Rotate — enabled
start delay 0 | duration 0.3 | vibrato 4 | elasticity 0.5
punch X 0 | Y 0 | Z 30

Punch Scale — enabled
start delay 0 | duration 0.3 | vibrato 4 | elasticity 0.5
punch X 0.2 | Y 0.2 | Z 0

Play Sound: ~No Sound~ | custom
Unity Events
Game Events
Navigation


OnDoubleClick ENABLED
wait for animation ✓ | register interval 0.2
Animation Type: State
Load selected state preset at runtime *(overrides current settings)*
Preset Category: Uncategorized | Preset Name: DefaultPreset
Load Preset | New Preset | Delete Preset

Move — enabled
start delay 0 | duration 0.5 | move by X 0 | Y 0 | Z 0
Ease | Linear

Rotate — enabled
start delay 0 | duration 0.5 | rotate by X 0 | Y 0 | Z 0
Ease | Linear

Scale — enabled
start delay 0 | duration 0.5 | scale by X 0 | Y 0 | Z 0
Ease | Linear

Fade — enabled
start delay 0 | duration 0.5 | alpha (transparency) to 1
Ease | Linear

Play Sound: ~No Sound~ | custom
Unity Events
Game Events
Navigation

The OnDoubleClick functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer executed a double click, or a finger double tap happened over it.

wait for animation: if enabled, the button action and game events are sent after the set animation has finished playing; this is useful if you want to be sure the uses see the button animation

register interval: time interval used to register a double click; this is the time interval calculated between two sequential clicks to determine if either a double click, or two separate clicks occurred

Animation Type: sets what type of animation this functionality uses; it can be a Punch Animation (see left image) or a State Animation (see right image)

Since you can have/use only one animation type per functionality, if you set the animation type to Punch, only the settings for Punch Animations will be available (visible) and, if you see the animation type to State, only the settings for the State Animations will be available(visible).

Play Sound: the sound name of the UISound that gets played when this functionality gets triggered

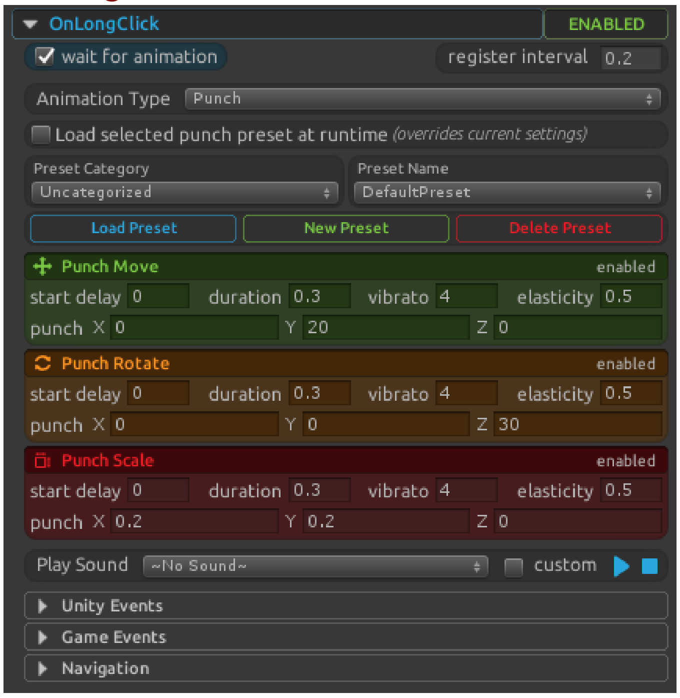
You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

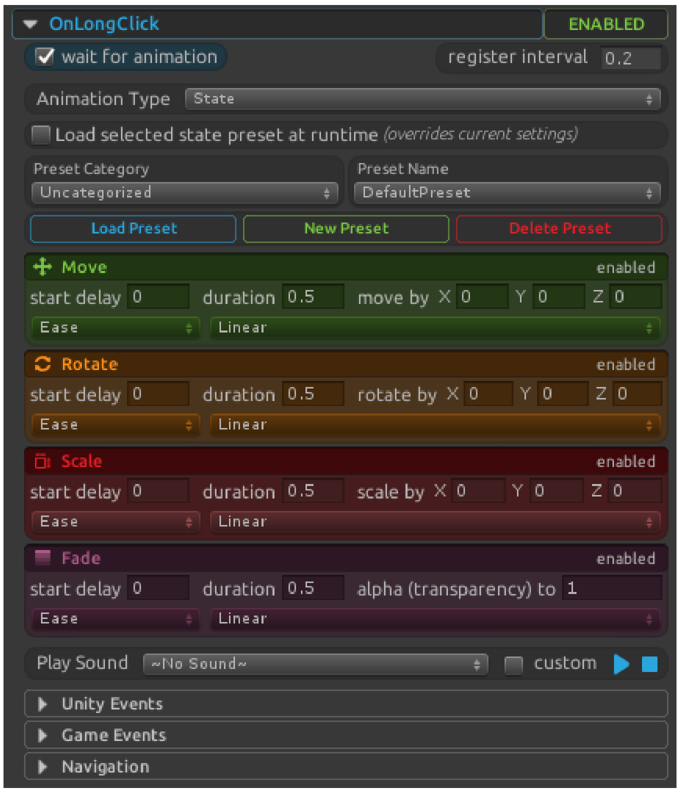
Unity Events
OnDoubleClick: UnityEvent invoked when the functionality gets triggered

Game Events
List of Game Events sent when the functionality gets triggered.

Navigation
If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden.
Add to Navigation History: if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back').
Show: is a list of pairs of element category & element name that will get shown when this functionality gets triggered
Hide: is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

OnLongClick


OnLongClick ENABLED
wait for animation ✓ register interval 0.2
Animation Type Punch
Load selected punch preset at runtime *(overrides current settings)*
Preset Category Uncategorized
Preset Name DefaultPreset
Load Preset | New Preset | Delete Preset
Punch Move enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 20 Z 0
Punch Rotate enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0 Y 0 Z 30
Punch Scale enabled
start delay 0 duration 0.3 vibrato 4 elasticity 0.5
punch X 0.2 Y 0.2 Z 0
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation


OnLongClick ENABLED
wait for animation ✓ register interval 0.2
Animation Type State
Load selected state preset at runtime *(overrides current settings)*
Preset Category Uncategorized
Preset Name DefaultPreset
Load Preset | New Preset | Delete Preset
Move enabled
start delay 0 duration 0.5 move by X 0 Y 0 Z 0
Ease Linear
Rotate enabled
start delay 0 duration 0.5 rotate by X 0 Y 0 Z 0
Ease Linear
Scale enabled
start delay 0 duration 0.5 scale by X 0 Y 0 Z 0
Ease Linear
Fade enabled
start delay 0 duration 0.5 alpha (transparency) to 1
Ease Linear
Play Sound ~No Sound~ custom
Unity Events
Game Events
Navigation

The OnLongClick functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer executed a long click (press), or a finger long tap (press) happened over it.

wait for animation: if enabled, the button action and game events are sent after the set animation has finished playing; this is useful if you want to be sure the uses see the button animation

register interval: time interval used to register a long click; this is the time interval a button has to be pressed down to be considered a long click

Animation Type: sets what type of animation this functionality uses; it can be a Punch Animation (see left image) or a State Animation (see right image)

Since you can have/use only one animation type per functionality, if you set the animation type to Punch, only the settings for Punch Animations will be available (visible) and, if you see the animation type to State, only the settings for the State Animations will be available(visible).

Play Sound: the sound name of the UISound that gets played when this functionality gets triggered

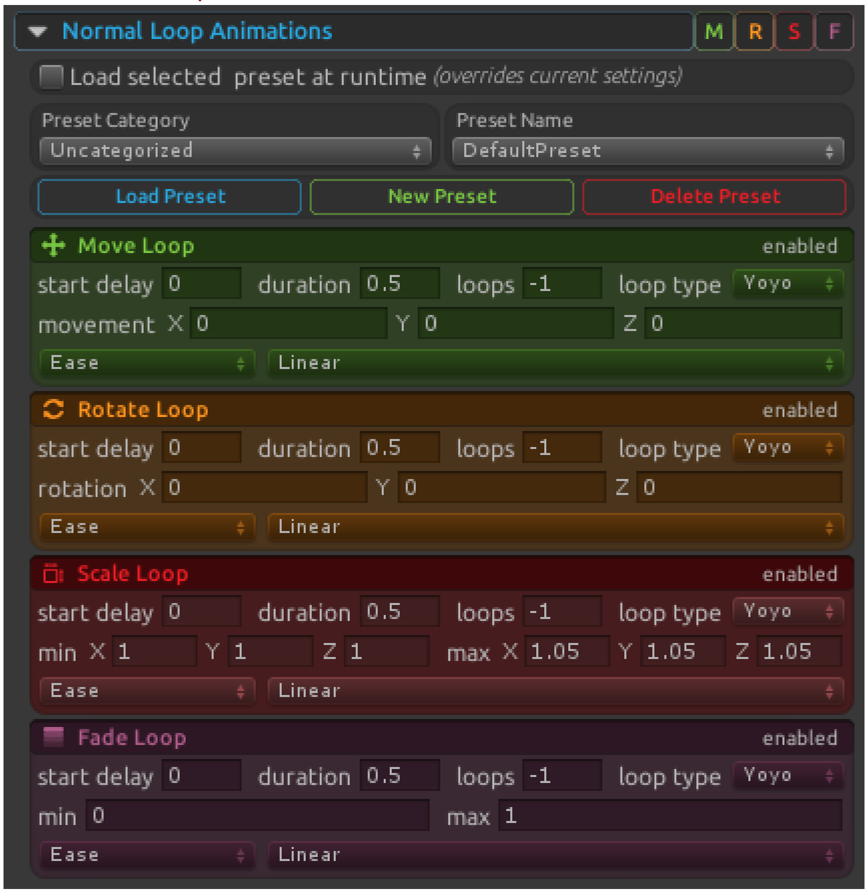
You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the same sound name) should be under a folder named 'Resources' in order for the sound to get played.

Unity Events
OnLongClick: UnityEvent invoked when the functionality gets triggered

Game Events
List of Game Events sent when the functionality gets triggered.

Navigation
If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden.
Add to Navigation History: if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back').
Show: is a list of pairs of element category & element name that will get shown when this functionality gets triggered
Hide: is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

Normal Loop Animations



Load selected preset at runtime *(overrides current settings)*

Preset Category: Uncategorized
Preset Name: DefaultPreset

Move Loop — enabled
start delay 0 duration 0.5 loops -1 loop type Yoyo
movement X 0 Y 0 Z 0
Ease Linear

Rotate Loop — enabled
start delay 0 duration 0.5 loops -1 loop type Yoyo
rotation X 0 Y 0 Z 0
Ease Linear

Scale Loop — enabled
start delay 0 duration 0.5 loops -1 loop type Yoyo
min X 1 Y 1 Z 1 max X 1.05 Y 1.05 Z 1.05
Ease Linear

Fade Loop — enabled
start delay 0 duration 0.5 loops -1 loop type Yoyo
min 0 max 1
Ease Linear

The Normal Loop Animations are triggered when the button is not selected (a button is selected when EventSystem currentSelected gameObject is the one that this UIButton component is attached to).

Because these animations are math based, any value change will result in a different animation.

Presets are different configuration sets that result in various animations.

There are two ways of setting up the animation. You can just select a preset, load its values (Load Preset) and either leave them like that or tweak them, or you can select a preset and set it to load at runtime (this will automatically load the preset values at runtime, overriding the settings from the Inspector).

Load selected preset at runtime: loads, at runtime, the animation preset that has the selected preset category and preset name; this will override any values set in the inspector

Preset Category: the selected animation preset category

Preset Name: the selected animation preset name (found in the selected preset category)

You can create new presets with ease, by creating new preset categories and preset names, right from the Inspector.

MOVE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
movement: movement is calculated startAnchoredPosition-movement for min and startAnchoredPosition+movment for max
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net

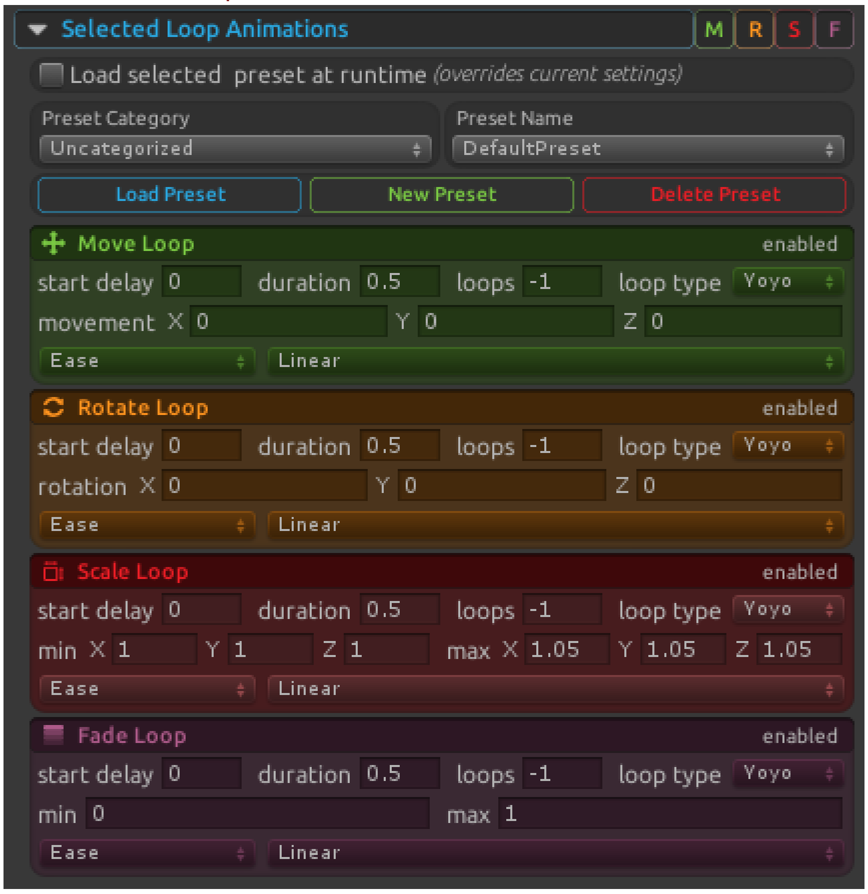
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

ROTATE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
rotation: rotation is calculated startRotation-rotation for min and startRotation+rotation for max
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

SCALE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
min: the minimum values for the scale factor of the scale loop animation (default: 1)
max: the maximum values for the scale factor of the scale loop animation (default: 1.05)
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

FADE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
min: the minimum alpha value for the fade animation loop (default: 0)
max: the maximum alpha value for the fade animation loop (default: 1)
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

Selected Loop Animations



The Selected Loop Animations are triggered when the button is selected (a button is selected when EventSystem currentSelected gameObject is the one that this UIButton component is attached to).

Because these animations are math based, any value change will result in a different animation.

Presets are different configuration sets that result in various animations.

There are two ways of setting up the animation. You can just select a preset, load its values (Load Preset) and either leave them like that or tweak them, or you can select a preset and set it to load at runtime (this will automatically load the preset values at runtime, overriding the settings from the Inspector).

Load selected preset at runtime: loads, at runtime, the animation preset that has the selected preset category and preset name; this will override any values set in the inspector

Preset Category: the selected animation preset category

Preset Name: the selected animation preset name (found in the selected preset category)

You can create new presets with ease, by creating new preset categories and preset names, right from the Inspector.

MOVE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
movement: movement is calculated startAnchoredPosition-movement for min and startAnchoredPosition+movment for max
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net

animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

ROTATE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
rotation: rotation is calculated startRotation-rotation for min and startRotation+rotation for max
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

SCALE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
min: the minimum values for the scale factor of the scale loop animation (default: 1)
max: the maximum values for the scale factor of the scale loop animation (default: 1.05)
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

FADE LOOP
start delay: start delay for the animation
duration: the duration of the animation
loops: number of loops this animation performs until it stops (-1 = infinite loops)
loop type: type of loops (Restart: each loop cycle restarts from the beginning / Yoyo: the tween moves forward and backwards at alternate cycles)
min: the minimum alpha value for the fade animation loop (default: 0)
max: the maximum alpha value for the fade animation loop (default: 1)
ease (type): select if the animation should use an Ease or an Animation curve in order to calculate the rate of change of the animation over time
ease: if easeType is set to Ease then this sets the ease of the tween; easing functions specify the rate of change of a parameter over time; to see how default ease curves look, check out easings.net
animation curve: if easeType is set to AnimationCurve, this curve will be used in order to calculate the rate of change of the animation over time

Fields

Name	*Description*
allowMultipleClicks	Should the button get disabled for a set interval (disableButtonInterval) between each click. By default, we allow the user to press the button multiple times.
BETWEEN_CLICKS_DISABLE_INTERVAL	Default value used to disable button after each click. Used when allow multiple clicks is set to false.
buttonCategory	The category this button name belongs to. The category is used only for database sorting purposes only. It does not matter when registering a button action.
buttonName	The name of this button. This is the value the system looks at when this button issues an action.
customOnClickSound	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnDoubleClickSound	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnLongClickSound	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnPointerDownSound	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnPointerEnterSound	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnPointerExitSound	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnPointerUpSound	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
debug	Enables debug logs.
DESELECT_BUTTON_DELAY	Special time interval added when deselecting a button. It fixes some anomalies.
deselectButtonOnClick	Should the button get deselected after each click. This is useful if you do not want this button to get selected after a click.
disableButtonInterval	If allowMultipleClicks is false, then this is the interval that this button will be disabled for between each click.
DOUBLE_CLICK_REGISTER_INTERVAL	Default time interval used to register a double click. This is the time interval calculated between two sequential clicks to determine if either a double click or two separate clicks occurred.
doubleClickRegisterInterval	Time interval used to register a double click. This is the time interval calculated between two sequential clicks to determine if either a double click or two separate clicks occurred.
loadNormalLoopPresetAtRuntime	Should the system load, at runtime, the Loop Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnClickPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnClickStatePresetAtRuntime	Should the system load, at runtime, the State Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnDoubleClickPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnDoubleClickStatePresetAtRuntime	Should the system load, at runtime, the State Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnLongClickPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnLongClickStatePresetAtRuntime	Should the system load, at runtime, the State Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerDownPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerDownStatePresetAtRuntime	Should the system load, at runtime, the State Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerEnterPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerEnterStatePresetAtRuntime	Should the system load, at runtime, the State Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerExitPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerExitStatePresetAtRuntime	Should the system load, at runtime, the State Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerUpPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerUpStatePresetAtRuntime	Should the system load, at runtime, the State Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadSelectedLoopPresetAtRuntime	Should the system load, at runtime, the Loop Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.

LONG_CLICK_REGISTER_INTERVAL	Default time interval used to register a long click. This is the time interval a button has to be pressed down to be considered a long click.
longClickRegisterInterval	Time interval used to register a long click. This is the time interval a button has to be pressed down to be considered a long click.
NORMAL_LOOP_ID	This is an extra id tag given to the tweener in order to locate the proper tween that manages the normal loop animations.
normalLoop	Loop Animation Settings
normalLoopPresetCategory	Loop Animation Preset Category Name
normalLoopPresetName	Loop Animation Preset Name
ON_POINTER_ENTER_DISABLE_INTERVAL	Default value used to disable the on pointer enter capture functionality after it has been triggered. Useful for certain cases.
ON_POINTER_EXIT_DISABLE_INTERVAL	Default value used to disable the on pointer exit capture functionality after it has been triggered. Useful for certain cases.
OnClick	UnityEvent invoked when on click has been captured by the system.
onClickAnimationType	Selects what type of animation this button is using OnClick.
onClickGameEvents	A list of game events that are sent when on click has been triggered.
onClickNavigation	UINavigation settings.
onClickPunch	Punch Animation Settings
onClickPunchPresetCategory	Punch Animation Preset Category Name
onClickPunchPresetName	Punch Animation Preset Name
onClickSound	The sound name of the sound that gets played on click.
onClickState	State Animation Settings
onClickStatePresetCategory	State Animation Preset Category Name
onClickStatePresetName	State Animation Preset Name
OnDoubleClick	UnityEvent invoked when on double click has been captured by the system.
onDoubleClickAnimationType	Selects what type of animation this button is using OnDoubleClick.
onDoubleClickGameEvents	A list of game events that are sent when on double click has been triggered.
onDoubleClickNavigation	UINavigation settings.
onDoubleClickPunch	Punch Animation Settings
onDoubleClickPunchPresetCategory	Punch Animation Preset Category Name
onDoubleClickPunchPresetName	Punch Animation Preset Name
onDoubleClickSound	The sound name of the sound that gets played on click.
onDoubleClickState	State Animation Settings
onDoubleClickStatePresetCategory	State Animation Preset Category Name
onDoubleClickStatePresetName	State Animation Preset Name
OnLongClick	UnityEvent invoked when on long click has been captured by the system.
onLongClickAnimationType	Selects what type of animation this button is using OnLongClick.
onLongClickGameEvents	A list of game events that are sent when on long click has been triggered.
onLongClickNavigation	UINavigation settings.
onLongClickPunch	Punch Animation Settings
onLongClickPunchPresetCategory	Punch Animation Preset Category Name
onLongClickPunchPresetName	Punch Animation Preset Name
onLongClickSound	The sound name of the sound that gets played on click.
onLongClickState	State Animation Settings
onLongClickStatePresetCategory	State Animation Preset Category Name
onLongClickStatePresetName	State Animation Preset Name
OnPointerDown	UnityEvent invoked when on pointer down has been captured by the system.
onPointerDownAnimationType	Selects what type of animation this button is using OnPointerDown.
onPointerDownGameEvents	A list of game events that are sent when on pointer down has been triggered.
onPointerDownNavigation	UINavigation settings.
onPointerDownPunch	Punch Animation Settings
onPointerDownPunchPresetCategory	Punch Animation Preset Category Name
onPointerDownPunchPresetName	Punch Animation Preset Name
onPointerDownSound	The sound name of the sound that gets played on pointer down.
onPointerDownState	State Animation Settings
onPointerDownStatePresetCategory	State Animation Preset Category Name
onPointerDownStatePresetName	State Animation Preset Name
OnPointerEnter	UnityEvent invoked when on pointer enter has been captured by the system.
onPointerEnterAnimationType	Selects what type of animation this button is using OnPointerEnter.
onPointerEnterDisableInterval	Time interval when the on pointer enter functionality is disabled after it has been triggered. Useful in certain cases.
onPointerEnterGameEvents	A list of game events that are sent when on pointer enter has been triggered.
onPointerEnterNavigation	UINavigation settings.
onPointerEnterPunch	Punch Animation Settings
onPointerEnterPunchPresetCategory	Punch Animation Preset Category Name
onPointerEnterPunchPresetName	Punch Animation Preset Name
onPointerEnterSound	The sound name of the sound that gets played on pointer enter.
onPointerEnterState	State Animation Settings
onPointerEnterStatePresetCategory	State Animation Preset Category Name
onPointerEnterStatePresetName	State Animation Preset Name
OnPointerExit	UnityEvent invoked when on pointer exit has been captured by the system.

onPointerExitAnimationType	Selects what type of animation this button is using OnPointerExit.
onPointerExitDisableInterval	Time interval when the on pointer exit functionality is disabled after it has been triggered. Useful in certain cases.
onPointerExitGameEvents	A list of game events that are sent when on pointer exit has been triggered.
onPointerExitNavigation	UINavigation settings.
onPointerExitPunch	Punch Animation Settings
onPointerExitPunchPresetCategory	Punch Animation Preset Category Name
onPointerExitPunchPresetName	Punch Animation Preset Name
onPointerExitSound	The sound name of the sound that gets played on pointer exit.
onPointerExitState	State Animation Settings
onPointerExitStatePresetCategory	State Animation Preset Category Name
onPointerExitStatePresetName	State Animation Preset Name
OnPointerUp	UnityEvent invoked when on pointer up has been captured by the system.
onPointerUpAnimationType	Selects what type of animation this button is using OnPointerUp.
onPointerUpGameEvents	A list of game events that are sent when on pointer up has been triggered.
onPointerUpNavigation	UINavigation settings.
onPointerUpPunch	Punch Animation Settings
onPointerUpPunchPresetCategory	Punch Animation Preset Category Name
onPointerUpPunchPresetName	Punch Animation Preset Name
onPointerUpSound	The sound name of the sound that gets played on pointer up.
onPointerUpState	State Animation Settings
onPointerUpStatePresetCategory	State Animation Preset Category Name
onPointerUpStatePresetName	State Animation Preset Name
SELECTED_LOOP_ID	This is an extra id tag given to the tweener in order to locate the proper tween that manages the selected loop animations.
selectedLoop	Loop Animation Settings
selectedLoopPresetCategory	Loop Animation Preset Category Name
selectedLoopPresetName	Loop Animation Preset Name
singleClickMode	Determines if on click is triggered instantly or after it checks if it's a double click or not. Depending on your use case, you might need the Instant or Delayed mode. Default is set to Instant.
useOnClickAnimations	Toggles the OnClick functionality. Not recommended to be disabled. If you disable this functionality, do some tests to be sure that the button behaves as you want it to.
useOnDoubleClick	Toggles the OnDoubleClick functionality.
useOnLongClick	Toggles the OnLongClick functionality.
useOnPointerDown	Toggles the OnPointerDown functionality.
useOnPointerEnter	Toggles the OnPointerEnter functionality.
useOnPointerExit	Toggles the OnPointerExit functionality.
useOnPointerUp	Toggles the OnPointerUp functionality.
waitForOnClickAnimation	If enabled, the button action and game events are sent after the on click punch animation has finished playing. This is useful if you want to be sure the uses see the button animation.
waitForOnDoubleClickAnimation	If enabled, the button action and game events are sent after the on double click punch animation has finished playing. This is useful if you want to be sure the uses see the button animation.
waitForOnLongClickAnimation	If enabled, the button action and game events are sent after the on long click punch animation has finished playing. This is useful if you want to be sure the uses see the button animation.

Properties

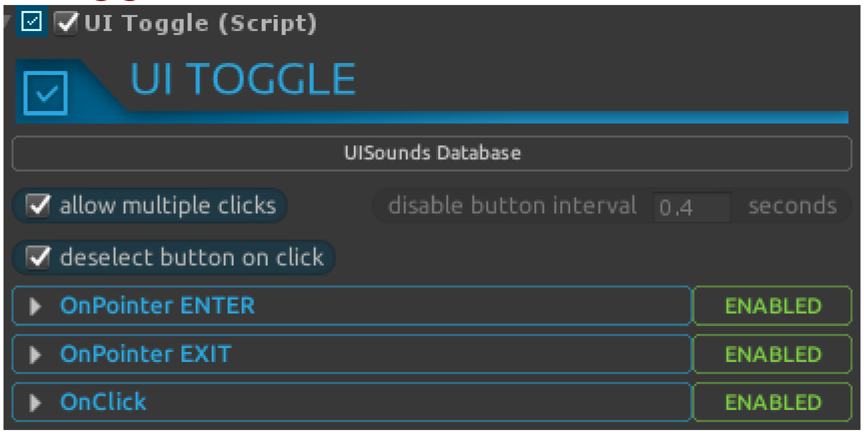
Name	*Description*
Button	Returns the Button component.
Interactable	Returns true if the button's Button component is interactable. This also toggles this button's intractability.
IsBackButton	Returns true if this button's name is 'Back'
IsSelected	Returns true if this button is selected, by checking the EventSystem.current.currentSelectedGameObject
RectTransform	Returns the RectTransform component.

Methods

Name	*Description*
AddGameEvent(string, ButtonActionType)	Add a game event to the target action type gameEvents list.
DisableButton()	Sets Interactable to false.
DisableButton(float)	Sets Interactable to false for the set duration. After that it sets Interactable to true.
EnableButton()	Sets Interactable to true.

ExecuteClick(bool)	Executes the OnClick trigger. You can force an execution of this trigger (regardless if it's enabled or not) by calling this method with forced set to TRUE
ExecuteDoubleClick(bool)	Executes the OnDoubleClick trigger. You can force an execution of this trigger (regardless if it's enabled or not) by calling this method with forced set to TRUE
ExecuteLongClick(bool)	Executes the OnLongClick trigger. You can force an execution of this trigger (regardless if it's enabled or not) by calling this method with forced set to TRUE
ExecuteOnPointerEnter(bool)	Executes the OnPointerEnter trigger. You can force an execution of this trigger (regardless if it's enabled or not) by calling this method with forced set to TRUE
ExecutePointerDown(bool)	Executes the OnPointerDown trigger. You can force an execution of this trigger (regardless if it's enabled or not) by calling this method with forced set to TRUE
ExecutePointerUp(bool)	Executes the OnPointerUp trigger. You can force an execution of this trigger (regardless if it's enabled or not) by calling this method with forced set to TRUE
ExecutePunch(Punch, bool, bool)	Executes a given punch animation.
ExecuteState(Anim, bool, bool)	Executes a given state animation.
GetButtonActionType(ButtonClickType)	Converts enum type from ButtonClickType to ButtonActionType
GetButtonClickType(ButtonActionType)	Converts enum type from ButtonActionType to ButtonClickType
OnDeselect(BaseEventData)	Used by IDeselectHandler.
OnSelect(BaseEventData)	Used by ISelectHandler.
RemoveGameEvent(string, ButtonActionType)	Remove game event from the target action type gameEvents list.
ResetAnimations()	
SendButtonClick()	Simulates this button's click action, without playing the set on click sound and punch animation.
SendButtonClick(bool, bool, bool, bool)	Simulates this button's click action and plays the set on click sound and punch animation.
SendButtonDoubleClick(bool, bool, bool, bool)	Simulates this button's double click action and plays the set on double click sound and punch animation.
SendButtonDoubleClick()	Simulates this button's double click action, without playing the set on double click sound and punch animation.
SendButtonLongClick(bool, bool, bool, bool)	Simulates this button's long click action and plays the set on long click sound and punch animation.
SendButtonLongClick()	Simulates this button's long click action, without playing the set on long click sound and punch animation.

UIToggle



☑ allow multiple clicks

disable button interval 0.4 seconds

☑ deselect button on click

OnPointer ENTER	ENABLED
OnPointer EXIT	ENABLED
OnClick	ENABLED

The UIToggle is a variation to the UIButton. It does not need nor use a button category or button name, but it does use Punch Animations.

All the settings have meaningful names and have been carefully arranged in an intuitive manner.

UISounds Database: opens the Control Panel window at the UISounds Tab

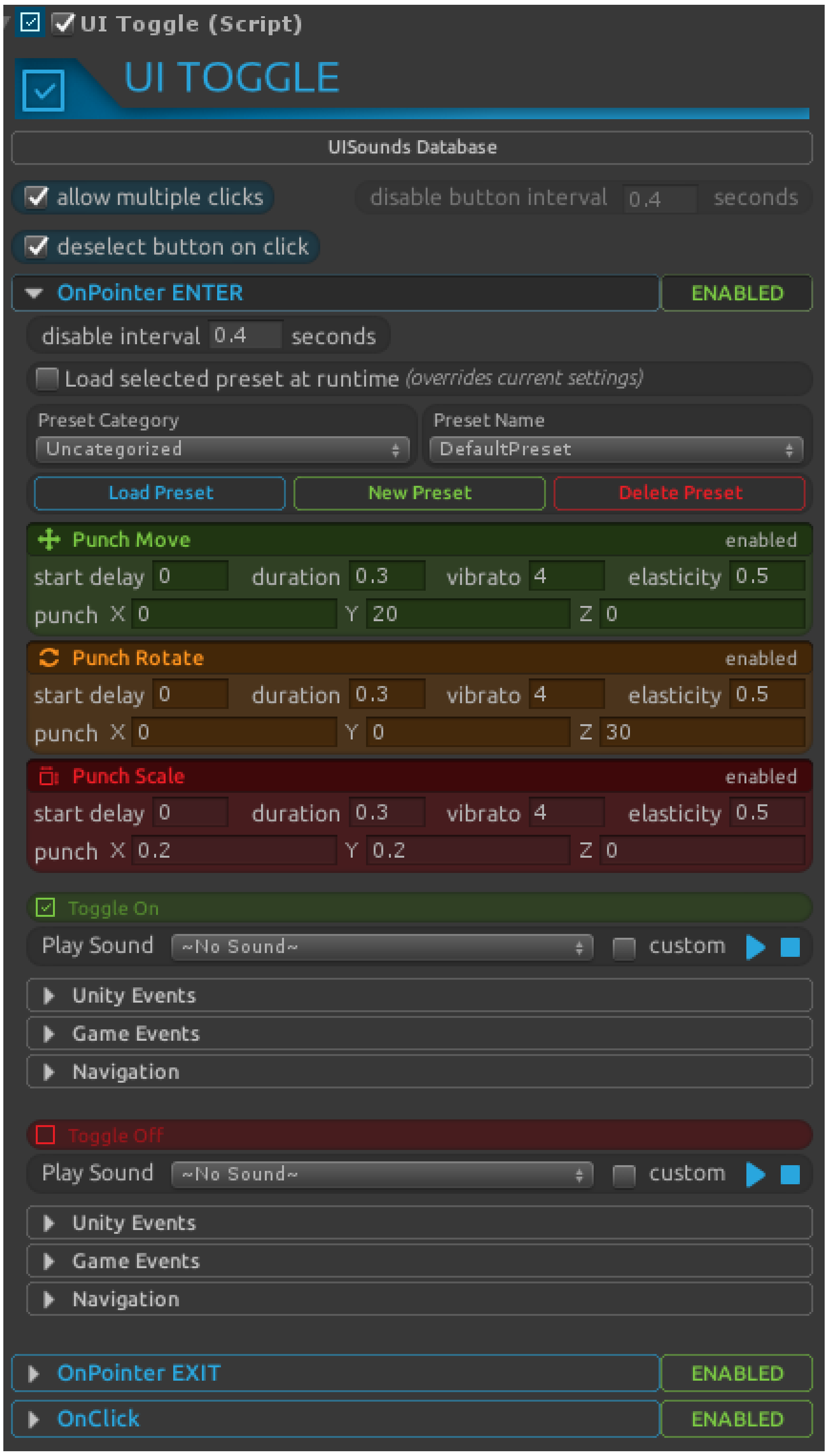
allow multiple clicks: if disabled, after each click, the button will get disabled for a set interval (disableButtonInterval); by default, the user is allowed to press the button multiple times without getting disabled (default: true)

disable button interval: is the duration that this button will get disabled after each click

deselect button on click: makes the button get deselected after each click; this is useful if you do not want the button to get selected after each click (maybe not to trigger the selected loop animation)

The OnPointer ENTER, OnPointer EXIT and OnClick functionalities can execute only one type of animation: a punch animation, that is the same as the one described for the UIButton component.

OnPointer ENTER


UI Toggle (Script)

UI TOGGLE

UISounds Database

☑ allow multiple clicks — disable button interval 0.4 seconds

☑ deselect button on click

OnPointer ENTER — ENABLED

disable interval 0.4 seconds

☐ Load selected preset at runtime *(overrides current settings)*

Preset Category: Uncategorized — Preset Name: DefaultPreset

Load Preset | New Preset | Delete Preset

Punch Move — enabled
start delay 0, duration 0.3, vibrato 4, elasticity 0.5
punch X 0, Y 20, Z 0

Punch Rotate — enabled
start delay 0, duration 0.3, vibrato 4, elasticity 0.5
punch X 0, Y 0, Z 30

Punch Scale — enabled
start delay 0, duration 0.3, vibrato 4, elasticity 0.5
punch X 0.2, Y 0.2, Z 0

☑ Toggle On
Play Sound ~No Sound~ ☐ custom
Unity Events
Game Events
Navigation

☐ Toggle Off
Play Sound ~No Sound~ ☐ custom
Unity Events
Game Events
Navigation

OnPointer EXIT — ENABLED

OnClick — ENABLED

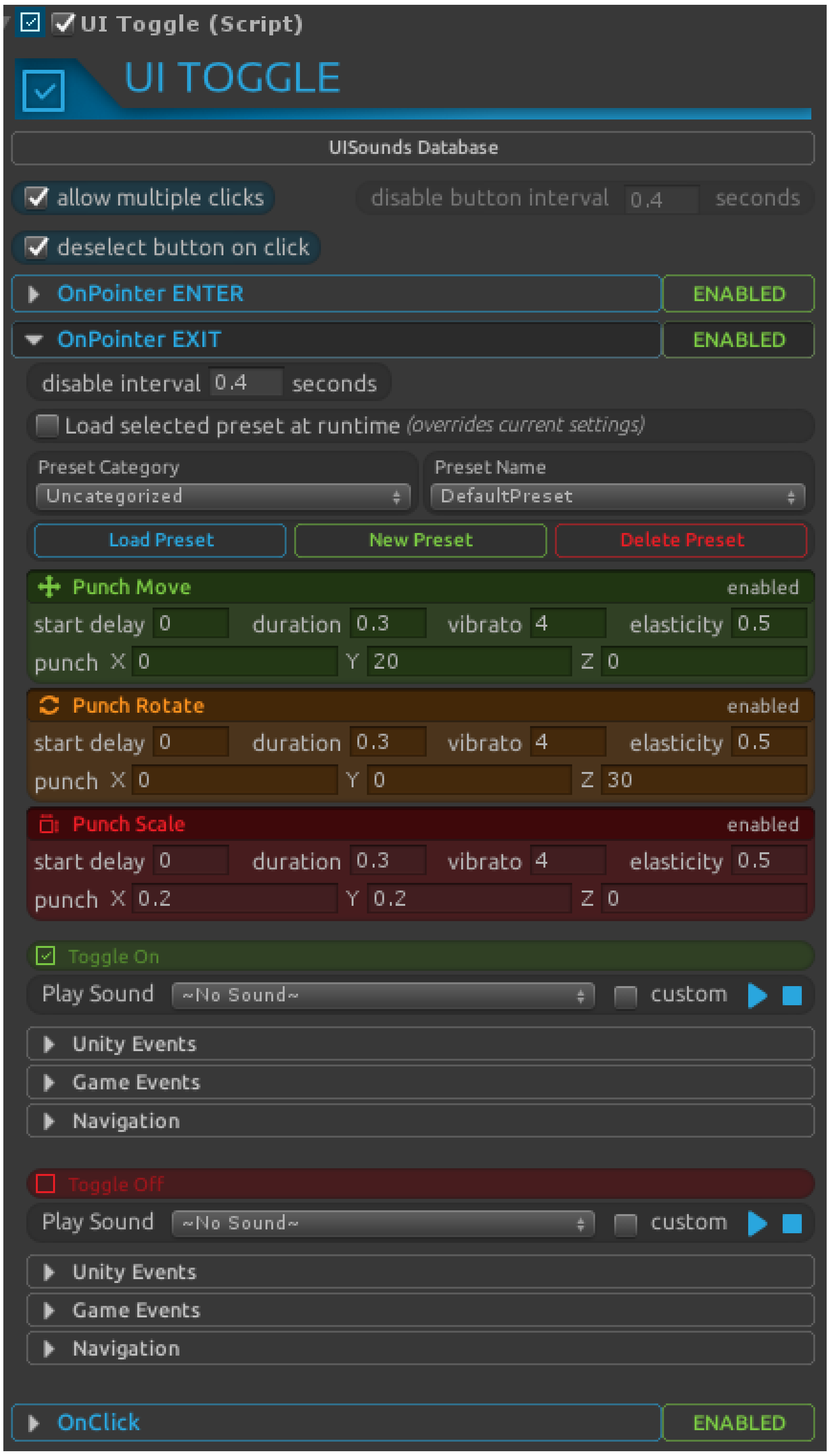
The OnPointer ENTER functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer enters its boundaries.

disable interval: time interval when the functionality gets disabled after it has been triggered; it is useful in certain cases

TOGGLE ON	TOGGLE OFF
Play Sound on Toggle On: the sound name of the UISound that gets played when this functionality gets triggered You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the	**Play Sound on Toggle Off:** the sound name of the UISound that gets played when this functionality gets triggered You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the

same sound name) should be under a folder named 'Resources' in order for the sound to get played.	same sound name) should be under a folder named 'Resources' in order for the sound to get played.
Unity Events **OnPointerEnterToggleOn:** UnityEvent invoked when the functionality gets triggered	**Unity Events** **OnPointerEnterToggleOff:** UnityEvent invoked when the functionality gets triggered
Game Events on Toggle On List of Game Events sent when the functionality gets triggered.	**Game Events on Toggle Off** List of Game Events sent when the functionality gets triggered.
Navigation on Toggle On If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden. **Add to Navigation History:** if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back'). **Show:** is a list of pairs of element category & element name that will get shown when this functionality gets triggered **Hide:** is a list of pairs of element category & element name that will get hidden when this functionality gets triggered	**Navigation on Toggle Off** If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden. **Add to Navigation History:** if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back'). **Show:** is a list of pairs of element category & element name that will get shown when this functionality gets triggered **Hide:** is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

OnPointer EXIT



UI Toggle (Script)

UI TOGGLE

UISounds Database

☑ allow multiple clicks   disable button interval 0.4 seconds

☑ deselect button on click

OnPointer ENTER   ENABLED

OnPointer EXIT   ENABLED

disable interval 0.4 seconds

☐ Load selected preset at runtime *(overrides current settings)*

Preset Category	Preset Name
Uncategorized	DefaultPreset

Load Preset   New Preset   Delete Preset

Punch Move   enabled

start delay	0	duration	0.3	vibrato	4	elasticity	0.5
punch X	0	Y	20	Z	0		

Punch Rotate   enabled

start delay	0	duration	0.3	vibrato	4	elasticity	0.5
punch X	0	Y	0	Z	30		

Punch Scale   enabled

start delay	0	duration	0.3	vibrato	4	elasticity	0.5
punch X	0.2	Y	0.2	Z	0		

Toggle On

Play Sound ~No Sound~   ☐ custom

Unity Events

Game Events

Navigation

Toggle Off

Play Sound ~No Sound~   ☐ custom

Unity Events

Game Events

Navigation

OnClick   ENABLED

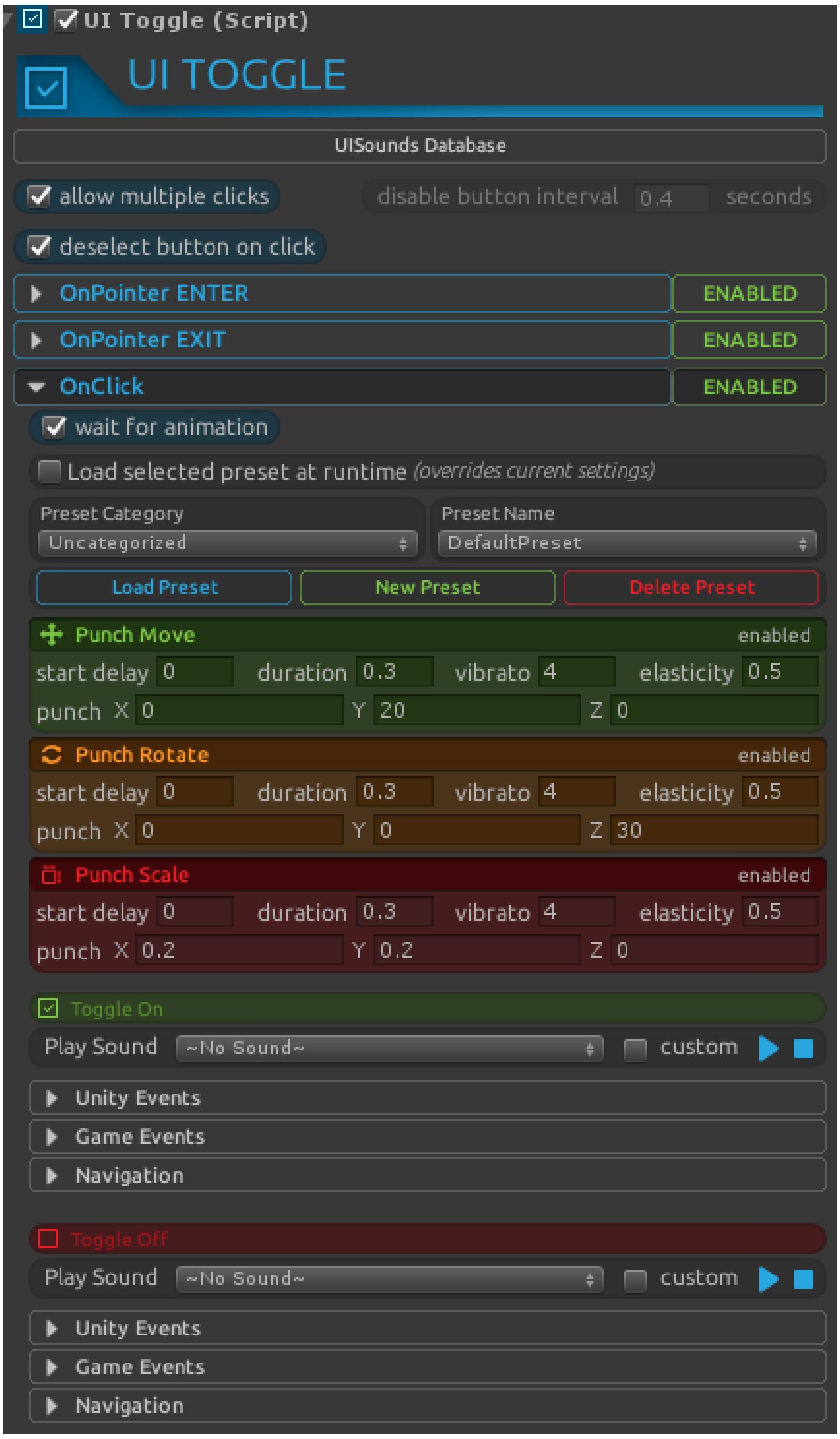
The OnPointer EXIT functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer exits its boundaries.

disable interval: time interval when the functionality gets disabled after it has been triggered; it is useful in certain cases

TOGGLE ON	TOGGLE OFF
Play Sound on Toggle On: the sound name of the UISound that gets played when this functionality gets triggered You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the	**Play Sound on Toggle Off:** the sound name of the UISound that gets played when this functionality gets triggered You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the

same sound name) should be under a folder named 'Resources' in order for the sound to get played.	same sound name) should be under a folder named 'Resources' in order for the sound to get played.
Unity Events **OnPointerEnterToggleOn:** UnityEvent invoked when the functionality gets triggered	**Unity Events** **OnPointerEnterToggleOff:** UnityEvent invoked when the functionality gets triggered
Game Events on Toggle On List of Game Events sent when the functionality gets triggered.	**Game Events on Toggle Off** List of Game Events sent when the functionality gets triggered.
Navigation on Toggle On If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden. **Add to Navigation History:** if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back'). **Show:** is a list of pairs of element category & element name that will get shown when this functionality gets triggered **Hide:** is a list of pairs of element category & element name that will get hidden when this functionality gets triggered	**Navigation on Toggle Off** If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden. **Add to Navigation History:** if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back'). **Show:** is a list of pairs of element category & element name that will get shown when this functionality gets triggered **Hide:** is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

OnClick



UI Toggle (Script)

UI TOGGLE

UISounds Database

✓ allow multiple clicks — disable button interval 0.4 seconds

✓ deselect button on click

OnPointer ENTER — ENABLED

OnPointer EXIT — ENABLED

OnClick — ENABLED

✓ wait for animation

Load selected preset at runtime *(overrides current settings)*

Preset Category	Preset Name
Uncategorized	DefaultPreset

Load Preset · New Preset · Delete Preset

Punch Move — enabled

start delay	0	duration	0.3	vibrato	4	elasticity	0.5
punch X	0	Y	20	Z	0		

Punch Rotate — enabled

start delay	0	duration	0.3	vibrato	4	elasticity	0.5
punch X	0	Y	0	Z	30		

Punch Scale — enabled

start delay	0	duration	0.3	vibrato	4	elasticity	0.5
punch X	0.2	Y	0.2	Z	0		

✓ Toggle On

Play Sound ~No Sound~ — custom

Unity Events

Game Events

Navigation

Toggle Off

Play Sound ~No Sound~ — custom

Unity Events

Game Events

Navigation

The OnClick functionality needs to be enabled before it can be configured and used. The button will react when the mouse pointer executed a click, or a finger tap happened over it.

wait for animation: if enabled, the button action and game events are sent after the set animation has finished playing; this is useful if you want to be sure the uses see the button animation

TOGGLE ON	TOGGLE OFF
Play Sound on Toggle On: the sound name of the UISound that gets played when this functionality gets triggered You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the	**Play Sound on Toggle Off:** the sound name of the UISound that gets played when this functionality gets triggered You can enter a custom sound name (that was not added to the UISounds database) just by enabling the custom option. But remember that the audio file (with the

same sound name) should be under a folder named 'Resources' in order for the sound to get played.	same sound name) should be under a folder named 'Resources' in order for the sound to get played.
Unity Events **OnPointerEnterToggleOn:** UnityEvent invoked when the functionality gets triggered	**Unity Events** **OnPointerEnterToggleOff:** UnityEvent invoked when the functionality gets triggered
Game Events on Toggle On List of Game Events sent when the functionality gets triggered.	**Game Events on Toggle Off** List of Game Events sent when the functionality gets triggered.
Navigation on Toggle On If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden. **Add to Navigation History:** if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back'). **Show:** is a list of pairs of element category & element name that will get shown when this functionality gets triggered **Hide:** is a list of pairs of element category & element name that will get hidden when this functionality gets triggered	**Navigation on Toggle Off** If the UINavigation is enabled from the Control Panel → General Tab, then you will have the option of setting what UIElements should be shown and/or hidden. **Add to Navigation History:** if enabled and if this functionality has any entries in the Show or in the Hide lists, they will be added to the Navigation History (that allows for reverse navigation upon pressing a button named 'Back'). **Show:** is a list of pairs of element category & element name that will get shown when this functionality gets triggered **Hide:** is a list of pairs of element category & element name that will get hidden when this functionality gets triggered

Fields

Name	*Description*
allowMultipleClicks	Should the button get disabled for a set interval (disableButtonInterval) between each click. By default, we allow the user to press the button multiple times.
BETWEEN_CLICKS_DISABLE_INTERVAL	Default value used to disable button after each click. Used when allow multiple clicks is set to false.
customOnClickSoundToggleOff	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnClickSoundToggleOn	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnPointerEnterSoundToggleOff	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnPointerEnterSoundToggleOn	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnPointerExitSoundToggleOff	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
customOnPointerExitSoundToggleOn	Used by the custom inspector to allow you to type a sound name instead of selecting it from the UISounds Database.
debug	Enables debug logs.
DESELECT_BUTTON_DELAY	Special time interval added when deselecting a button. It fixes some anomalies.
deselectButtonOnClick	Should the button get deselected after each click. This is useful if you do not want this button to get selected after a click.
disableButtonInterval	If allowMultipleClicks is false, then this is the interval that this button will be disabled for between each click.
loadOnClickPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerEnterPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
loadOnPointerExitPunchPresetAtRuntime	Should the system load, at runtime, the Punch Preset with the set Preset Category and Preset Name. This overrides any values set in the inspector.
ON_POINTER_ENTER_DISABLE_INTERVAL	Default value used to disable the on pointer enter capture functionality after it has been triggered. Useful for certain cases.
ON_POINTER_EXIT_DISABLE_INTERVAL	Default value used to disable the on pointer exit capture functionality after it has been triggered. Useful for certain cases.
onClickGameEventsToggleOff	A list of game events that are sent when on click has been triggered and the toggle is off.
onClickGameEventsToggleOn	A list of game events that are sent when on click has been triggered and the toggle is on.
onClickNavigationToggleOff	UINavigation settings when the toggle is off.
onClickNavigationToggleOn	UINavigation settings when the toggle is on.
onClickPunch	Punch Animation Settings
onClickPunchPresetCategory	Punch Animation Preset Category Name

onClickPunchPresetName	Punch Animation Preset Name
onClickSoundToggleOff	The sound name of the sound that gets played on click and the toggle is off.
onClickSoundToggleOn	The sound name of the sound that gets played on click and the toggle is on.
OnClickToggleOff	UnityEvent invoked when on click has been captured by the system and the toggle is off.
OnClickToggleOn	UnityEvent invoked when on click has been captured by the system and the toggle is on.
onPointerEnterDisableInterval	Time interval when the on pointer enter functionality is disabled after it has been triggered. Useful in certain cases.
onPointerEnterGameEventsToggleOff	A list of game events that are sent when on pointer enter has been triggered and the toggle is off.
onPointerEnterGameEventsToggleOn	A list of game events that are sent when on pointer enter has been triggered and the toggle is on.
onPointerEnterNavigationToggleOff	UINavigation settings when the toggle is off.
onPointerEnterNavigationToggleOn	UINavigation settings when the toggle is on.
onPointerEnterPunch	Punch Animation Settings
onPointerEnterPunchPresetCategory	Punch Animation Preset Category Name
onPointerEnterPunchPresetName	Punch Animation Preset Name
onPointerEnterSoundToggleOff	The sound name of the sound that gets played on pointer enter and the toggle is of.
onPointerEnterSoundToggleOn	The sound name of the sound that gets played on pointer enter and the toggle is on.
OnPointerEnterToggleOff	UnityEvent invoked when on pointer enter has been captured by the system and the toggle is off.
OnPointerEnterToggleOn	UnityEvent invoked when on pointer enter has been captured by the system and the toggle is on.
onPointerExitDisableInterval	Time interval when the on pointer exit functionality is disabled after it has been triggered. Useful in certain cases.
onPointerExitGameEventsToggleOff	A list of game events that are sent when on pointer exit has been triggered and the toggle is off.
onPointerExitGameEventsToggleOn	A list of game events that are sent when on pointer exit has been triggered and the toggle is on.
onPointerExitNavigationToggleOff	UINavigation settings when the toggle is off.
onPointerExitNavigationToggleOn	UINavigation settings when the toggle is on.
onPointerExitPunch	Punch Animation Settings
onPointerExitPunchPresetCategory	Punch Animation Preset Category Name
onPointerExitPunchPresetName	Punch Animation Preset Name
onPointerExitSoundToggleOff	The sound name of the sound that gets played on pointer exit and the toggle is off.
onPointerExitSoundToggleOn	The sound name of the sound that gets played on pointer exit and the toggle is on.
OnPointerExitToggleOff	UnityEvent invoked when on pointer exit has been captured by the system and the toggle is off.
OnPointerExitToggleOn	UnityEvent invoked when on pointer exit has been captured by the system and the toggle is on.
useOnClick	Toggles the OnClick functionality. Not recommended to be disabled. If you disable this functionality, do some tests to be sure that the button behaves as you want it to.
useOnPointerEnter	Toggles the OnPointerEnter functionality.
useOnPointerExit	Toggles the OnPointerExit functionality.
waitForOnClick	If enabled, the button action and game events are sent after the on click punch animation has finished playing. This is useful if you want be sure the uses sees the button animation.

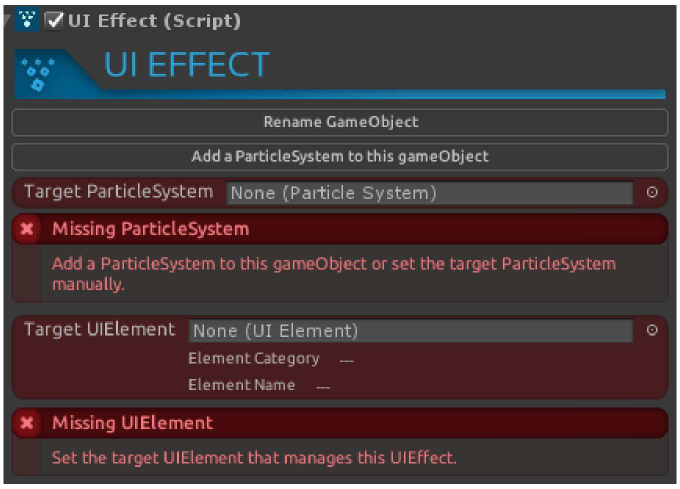
Properties

Name	*Description*
Interactable	Returns true if the button's Toggle component is interactable. This also toggles this button's interactability.
IsOn	Returns true if the toggle is on and false otherwise. This also toggles this toggle's on/off state.
IsSelected	Returns true if this button is selected, by checking the EventSystem.current.currentSelectedGameObject
RectTransform	Returns the RectTransform component.
Toggle	Returns the Button component.

Methods

Name	*Description*
AddGameEvent(string, bool, ToggleActionType)	Add a game event to the target action type gameEvents list.
DisableButton(float)	Sets Interactable to false for the set duration. After that it sets Interactable to true.
DisableButton()	Sets Interactable to false.
EnableButton()	Sets Interactable to true.
ExecuteClick(bool)	Executes the OnClick trigger. You can force an execution of this trigger (regardless if it's enabled or not) by calling this method with forced set to TRUE
ExecuteOnPointerEnter(bool)	Executes the OnPointerEnter trigger. You can force an execution of this trigger (regardless if it's enabled or not) by calling this method with forced set to TRUE
OnDeselect(BaseEventData)	Used by IDeselectHandler.
OnSelect(BaseEventData)	Used by ISelectHandler.
RemoveGameEvent(string, bool, ToggleActionType)	Remove game event from the target action type gameEvents list.

UIEffect


UI Effect (Script)

UI EFFECT

Rename GameObject

Add a ParticleSystem to this gameObject

Target ParticleSystem None (Particle System)

Missing ParticleSystem

Add a ParticleSystem to this gameObject or set the target ParticleSystem manually.

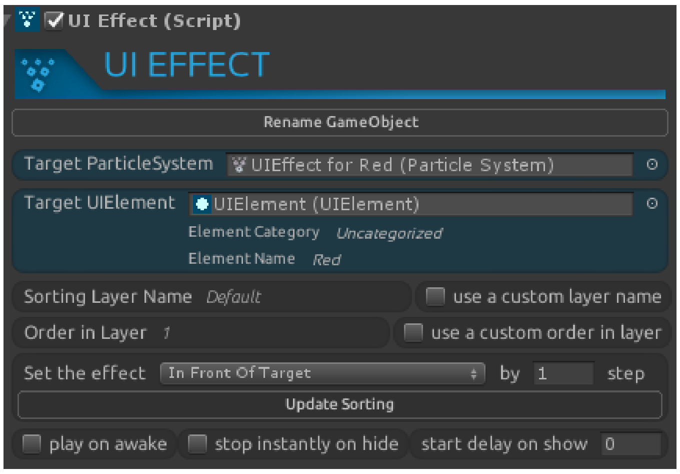
Target UIElement None (UI Element)

Element Category ---

Element Name ---

Missing UIElement

Set the target UIElement that manages this UIEffect.


UI Effect (Script)

UI EFFECT

Rename GameObject

Target ParticleSystem UIEffect for Red (Particle System)

Target UIElement UIElement (UIElement)

Element Category *Uncategorized*

Element Name *Red*

Sorting Layer Name *Default* ☐ use a custom layer name

Order in Layer *1* ☐ use a custom order in layer

Set the effect In Front Of Target by 1 step

Update Sorting

☐ play on awake ☐ stop instantly on hide start delay on show 0

The UIEffect helps you use native ParticleSystem components with the UI. It does the proper setup for the ParticleSystem component by making the effect be shown in front or behind the target UIElement by a specified number of order in layer steps.

Note: in order to be able to sort the ParticleSystem components with the UI, the root canvas Render Mode needs to be set up to be either Screen Space – Camera or World Space

Another feature of this component is that it gets linked to an UIElement and, when the target UIElement gets shown, this UIEffect start emitting particles, and,

On the left image there is an UIEffect that has no ParticleSystem referenced, nor a target UIElement. Thus, it is disabled and will not work. Note that there are Info Messages that show and tell you exactly what to do in order to make this UIEffect operational (look at the right image).

Rename GameObject: renames the gameObject that this UIEffect is attached to 'UIEffect for 'target element name''; this button can be hidden from the Control Panel → Editor Settings Tab → UIEffect

Add a ParticleSystem to this gameObject: this button will attach a ParticleSystem component to this gameObject and reference it as the target ParticleSystem; this button is visible only if a target ParticleSystem has not been referenced;

Target ParticleSystem: the ParticleSystem that this UIEffect is controlling

Target UIElement: the UIElement that controls this UIEffect

Sorting Layer Name: the name of the sorting layer that the ParticleSystem belongs to

use a custom layer name: allows entering a custom sorting layer name value

Order in Layer: the sorting order in the set sorting layer

use a custom order in layer: allows entering a custom order in layer value

Set the effect – In Front Of Target: this will set the Order in Layer value by adding the number of steps to the order in layer of the target UIElement; makes the target ParticleSystem visible in front of the target UIElement

Set the effect – Behind Target: this will set the Order in Layer value by subtracting the number of steps from the order in layer of the target UIElement; makes the target ParticleSystem visible behind the target UIElement

Update Sorting: recalculates the Order in Layer value and updates the sorting for the ParticleSystem component

play on awake: should the effect start playing on awake or not (default: false)

stop instantly on hide: should the effect stop instantly and clear, after the 'Hide' command has been issued (for the target UIElement), or should it just stop and let the particle disappear (linger) after their set lifetime (default: false)

start delay on show: time interval to wait to play this effect, after the 'Show' command has been issued (for the target UIElement)

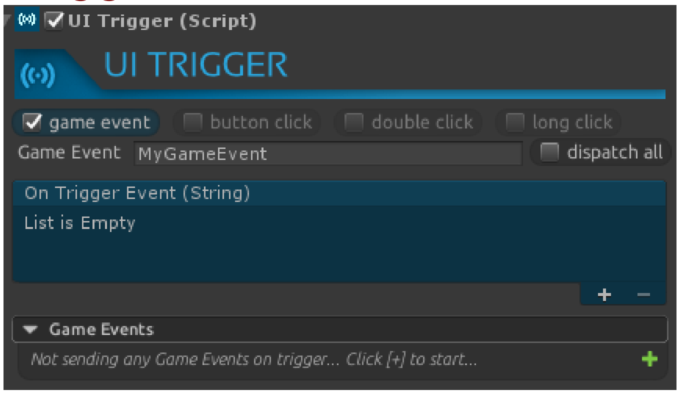
Fields

Name	*Description*
autoRegister	Used by the UINotification. If this effect is used by a notification, then the notification should handle it's registration process in order to use an auto generated name. Do not change this value yourself.
customOrderInLayer	Used by the custom inspector to set your custom order in layer. Use this only if you know what you are doing.
customSortingLayerName	Used by the custom inspector to set your custom layer name. Use this only if you know what you are doing.
DEFAULT_CUSTOM_ORDER_IN_LAYER	Default order in layer value.
DEFAULT_CUSTOM_SORTING_LAYER_NAME	Default sorting layer name value.
DEFAULT_DEFAULT_SORTING_ORDER_STEP	Default sorting order step value.
effectPosition	Determines the order in layer by adding (if InFrontOfTarget) or subtracting (if BehindTarget) the set number of sorting order steps to the order in layer value.
isVisible	Keeps track if this UIEffect is visible or not. Do not change this value yourself.
playOnAwake	Should this effect start playing on awake or not. Default is set to false.
sortingOrderStep	Considering the target's [Canvas][Order in Layer][value] - we adjust the [ParticleSystem][Renderer][Order in Layer][value] with this sorting step (by adding, if set to InFrontOfTarget or subtracting, if set BehindTarget)
startDelay	Time interval to wait to play this effect, after the show command has been sent for the target UIElement.
stopInstantly	Should the effect stop instantly and clear, after the hide command has been sent, or should it stop and let the particles disappear after their set lifetime. Default is set to false.
targetParticleSystem	The ParticleSystem that this UIEffect is controlling.
targetUIElement	The UIElement that controls this UIEffect.
useCustomOrderInLayer	Used by the custom inspector to allow you to type a order in layer instead of getting it automatically set by this UIEffect. Use this only if you know what you are doing.
useCustomSortingLayerName	Used by the custom inspector to allow you to type a layer name instead of selecting it from the layers dropdown list. Use this only if you know what you are doing.

Methods

Name	*Description*
Hide(bool)	Hides this UIEffect (similar to the Hide method of the UIElement).
RegisterToUIManager()	Registers this UIEffect to the UIManager.
Show(bool)	Shows this UIEffect (similar to the Show method of the UIElement).
UnregisterFromUIManager()	Unregisteres this UIEffect from the UIManager.
UpdateSorting()	Updates the sorting of this effect to the set and calculated values.

UITrigger


UI Trigger (Script)

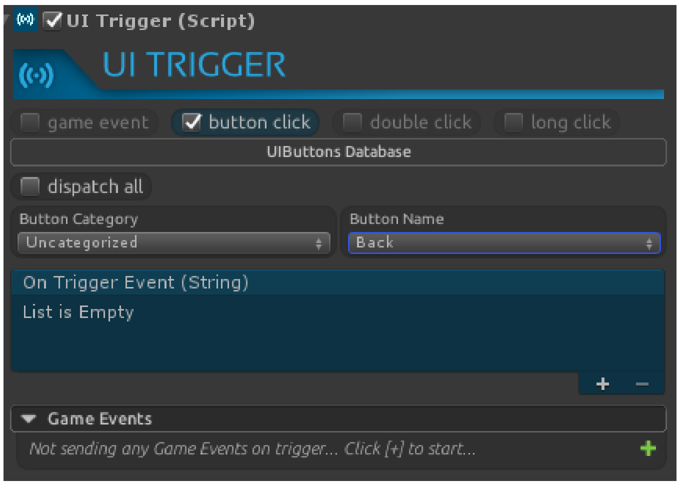
UI TRIGGER

✓ game event ☐ button click ☐ double click ☐ long click

Game Event: MyGameEvent ☐ dispatch all

On Trigger Event (String)
List is Empty

Game Events
Not sending any Game Events on trigger... Click [+] to start...


UI Trigger (Script)

UI TRIGGER

☐ game event ✓ button click ☐ double click ☐ long click

UIButtons Database

☐ dispatch all

Button Category: Uncategorized
Button Name: Back

On Trigger Event (String)
List is Empty

Game Events
Not sending any Game Events on trigger... Click [+] to start...

The UITrigger component listens for one or all game events or button clicks and then invokes an UnityEvent and sends any game events that have been set up.

It can listen for any of the following:

1. game event
2. button click
3. button double click
4. button long click

Game Event

Here you have the option to either listen for a single game event (you enter it in the inspector), or you can enable **dispatch all** and then this UITrigger will get triggered every time a game event is sent (regardless of what it is).

Button Click / Double Click / Long Click

Regardless of the type of click that this UITrigger is set to react to, you will have the option of selecting a button category and a button name from the UIButtons database (you can also open the database using the 'UIButtons Database' button).

If **dispatch all** is enabled, then this UITrigger will get triggered every time a button click / double click / long click (depending on the listener) has been registered by the system.

OnTriggerEvent: UnityEvent invoked when this UITrigger gets triggered

Game Events: list of Game Events sent when this UITrigger gets triggered

Fields

Name	*Description*
buttonCategory	Used by the custom inspector to allow you to select a button name from the UIButtons Database.
buttonName	If any of triggerOnButtonClick or triggerOnButtonDoubleClick or triggerOnButtonLongClick are true, this is the button name value that will make this UITrigger execute its actions.
dispatchAll	If dispatch all is set to true, game event and button name are set to a special value that make this UITrigger execute its actions on every game event or button click/double click/long click.
gameEvent	If triggerOnGameEvent is true, this is the game event value that will make this UITrigger execute its actions.
gameEvents	List of game events that are sent by the UITrigger when it executes its actions.
onTriggerEvent	UnityEvent invoked when the UITrigger has been triggered.
triggerOnButtonClick	Should this UITrigger execute its actions on button click? Default is false.
triggerOnButtonDoubleClick	Should this UITrigger execute its actions on button double click? Default is false.

triggerOnButtonLongClick	Should this UITrigger execute its actions on button long click? Default is false.
triggerOnGameEvent	Should this UITrigger execute its actions on game event? Default is false.

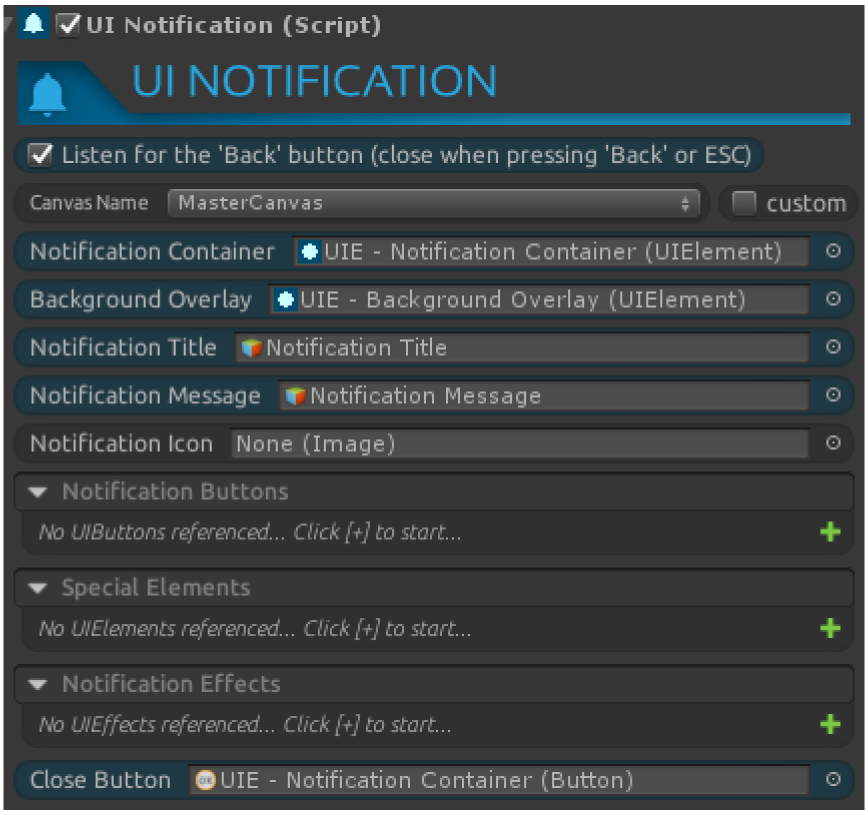
Properties

Name	*Description*
Enabled	Returns true if this UITrigger has proper settings set up and is operational.
ListeningFor	Returns the type of event that this UITrigger is listening for.

Methods

Name	*Description*
RegisterToUIManager()	Registers this UITrigger to the UIManager.
TriggerTheTrigger(string)	Triggers the UITrigger to execute its actions.
UnregisterFromUIManager()	Unregisters this UITrigger from the UIManager.

UINotification


UI Notification (Script)

UI NOTIFICATION

☑ Listen for the 'Back' button (close when pressing 'Back' or ESC)

Canvas Name	MasterCanvas ☐ custom
Notification Container	UIE - Notification Container (UIElement)
Background Overlay	UIE - Background Overlay (UIElement)
Notification Title	Notification Title
Notification Message	Notification Message
Notification Icon	None (Image)

Notification Buttons

No UIButtons referenced... Click [+] to start...

Special Elements

No UIElements referenced... Click [+] to start...

Notification Effects

No UIEffects referenced... Click [+] to start...

Close Button: UIE - Notification Container (Button)

The UINotification is a complex component that can be used in countless ways. You can think of an UINotification as a modal window. It can be a simple tooltip, or a complex window with several panels (UIElements) and buttons (UIButtons) inside.

It can manage (show/hide) an UIElement container (that can house several other UIElements), a background overlay (another UIElement that can be animated and used as a background) and several other (special) UIElements, UIButtons and UIEffects.

Listen for the 'Back' button: should this notification listen for the 'Back' button? If yes, upon pressing the 'Back' button, the notification will close by automaically calling Hide on itself. Default is true.

Canvas Name: the target (UICanvas) canvas name where the UINotification will get shown

Notification Container: reference to a child UIElement that can be considered the body of the notification (in most cases)

Background Overlay: reference to a child UIElement that might have an Image component attached and that can be used as an animated background

Notification Title: reference to a child GameObject that has a Text or a TextMeshPro component attached (depending on your setup) that will be used as the notification title (when showing the notification, you can pass a title string)

Notification Message: reference to a child GameObject that has a Text or a TextMeshPro component attached (depending on your setup that will be used as the notification message (when showing the notification, you can pass a message string)

Notification Icon: reference to a child GameObject that has an Image component attached that will be used as the notification icon (when showing the notification, you can pass a sprite reference that will be set as the Image's sprite – the icon)

Notification Buttons: an array of references to UIButtons (that are under the UINotification), that will be used (configured) as the notification's buttons

Special Elements: an array of references to any other child UIElements that need to be controlled by this notification. It allows for a lot of flexibility design wise. For example, if you have 3 stars with different animations, you can create an UIElement gameObject for each, set up their respective animations and reference them to this array

Notification Effects: an array of references to any child UIEffects used by this notification

Close Button: this is a reference to a Button component, that is attached by default to the notification's gameObject. Upon clicking the notification, it will auto close (by calling 'Hide' on itself).

Fields

Name	*Description*
buttons	An array of references of child UIButtons that will be used as the notification's buttons.
closeButton	This is a reference to a Button component, that is attached by default to the notification's gameObject. Upon clicking the notification, it will auto close (by calling Hide on itself).
customTargetCanvasName	Used by the custom inspector to allow you to type a canvas name instead of selecting it from the Canvas Names Database.
data	Used by the notification when it gets set up.
DEFAULT_ADD_TO_NOTIFICATION_QUEUE	Default behavior if a notification should be added to the notification queue or be shown right away.
DEFAULT_BUTTON_NAMES	Default notification array of button names (the button name is used to distinguish buttons one from the other).
DEFAULT_BUTTON_TEXT	Default notification array of button texts (the button texts are the text values shown on the buttons).
DEFAULT_ICON	Default notification icon.
DEFAULT_LIFETIME	Default time interval of how long a notification should be seen on screen before the Hide command is automatically issued.
DEFAULT_MESSAGE	Default notification message.
DEFAULT_TITLE	Default notification title.
effects	An array of references to any child UIEffects used by this notification.
icon	Reference to a child GameObject with an Image attached. This Image component will get its sprite value set to the notification's icon.
listenForBackButton	Should this notification listen for the 'Back' button? If yes, upon pressing the 'Back' button, the notification will close by automaically calling Hide on itself. Default is true.
message	Reference to a child GameObject that has a Text component attached. This Text component will get its text value set to the notification's message.
notificationContainer	Reference to the main UIElement that holds everything.
overlay	Reference to an UIElement that can be used as a background image.
specialElements	An array of references to any other child UIElements that need to be controlled by this notification. It allows for a lot of flexibility design wise. For example, if you have 3 stars with different animations, you can create an UIElement gameObject for each, set up their respective animations and reference them to this array.
targetCanvasName	The target canvas where this notification will be shown.
title	Reference to a child GameObject that has a Text component attached. This Text component will get its text value set to the notification's title.

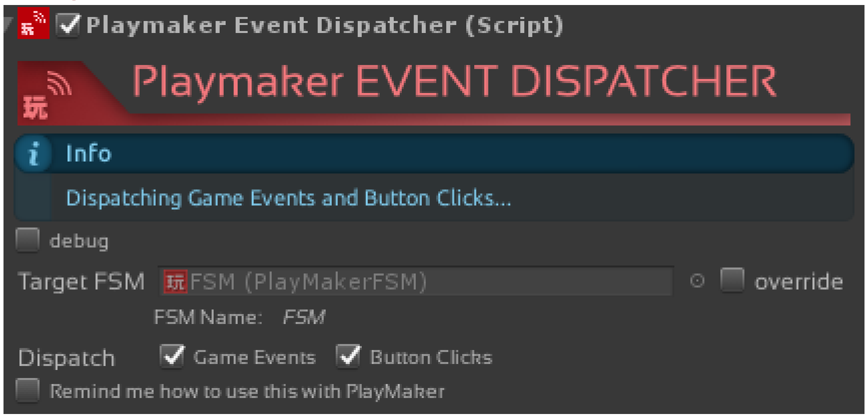
Properties

Name	*Description*
RectTransform	Returns the RectTransform component.

Methods

Name	*Description*
HideNotification(bool)	Hides the notification with a destroy option. Default notification behavior is to get automatically destroyed.
Initialize()	Executes the initial setup of this notification.
ShowNotification(NotificationData, UICanvas)	Shows the notification considering the NotificationData value.

Playmaker EVENT DISPATCHER



Playmaker Event Dispatcher (Script)

Playmaker EVENT DISPATCHER

Info

Dispatching Game Events and Button Clicks...

☐ debug

Target FSM: FSM (PlayMakerFSM) ☐ override

FSM Name: FSM

Dispatch: ✓ Game Events ✓ Button Clicks

☐ Remind me how to use this with PlayMaker

This component is available only if you have the Playmaker asset installed and if you enabled Playmaker support from Tools → Control Panel → General Tab.

The role of the Playmaker EVENT DISPATCHER is to send game events and/or button clicks (button names of the buttons that were clicked) to any Playmaker FSM.

When attached to a gameObject, the component ill search for a Playmaker FSM and set it as its target FSM (the target FSM is the one that receives the game events and/or button names).

debug: prints to the console all the game events or button clicks (names) that are being sent at runtime

Target FSM: the target FSM that will receive all the game events and/or button clicks (names)

override Target FSM: used by the custom inspector to allow you to override the target FSM reference.

dispatch Game Events: if enabled, all the game events will get dispatched to the target FSM

dispatch Button Clicks: if enabled, all the button clicks (names) will get dispatched to the target FSM

Remind me how to use this with PlayMaker: if enabled, it will display a quick primer on how the Playmaker EVENT DISPATCHER works and how to use it.

Fields

Name	*Description*
debug	Prints to the console all the game events and/or button clicks (names) that are being sent at runtime.
dispatchButtonClicks	If enabled, all the button clicks (names) will get dispatched to the target FSM.
dispatchGameEvents	If enabled, all the game events will get dispatched to the target FSM.
overrideTargetFSM	Used by the custom inspector to all you to override the target FSM reference.
targetFSM	The target FSM that will receive all the game events and/or button clicks (names)

Properties

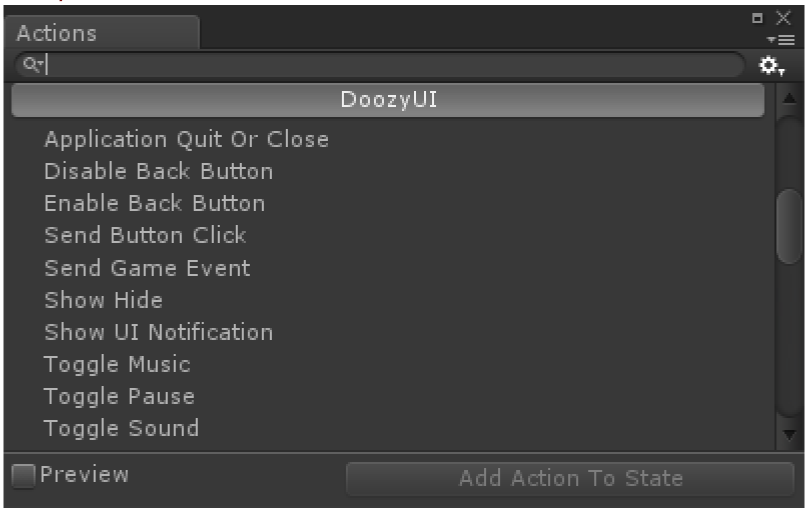
Name	*Description*
Enabled	Gets a value indicating whether this PlaymakerEventDispatcher is enabled.

Methods

Name	*Description*
DispatchEvent(string, DUI.EventType)	Dispatches an event that can be either a game event or button click (name).

RegisterToUIManager()	Registers this PlaymakerEventDispatcher to the UIManager.
UnregisterFromUIManager()	Unregisters this PlaymakerEventDispatcher from the UIManager.

Playmaker Actions

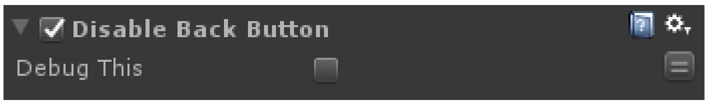

Actions

DoozyUI

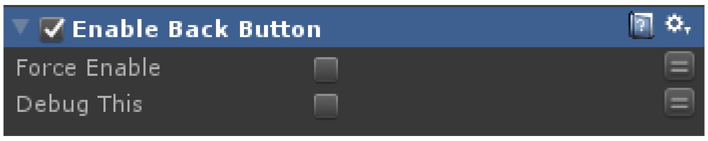
- Application Quit Or Close
- Disable Back Button
- Enable Back Button
- Send Button Click
- Send Game Event
- Show Hide
- Show UI Notification
- Toggle Music
- Toggle Pause
- Toggle Sound

These are the Playmaker Actions that will be available if the Playmaker asset has been installed and the Playmaker support has been enabled from Tools → DoozyUI → Control Panel → General Tab.

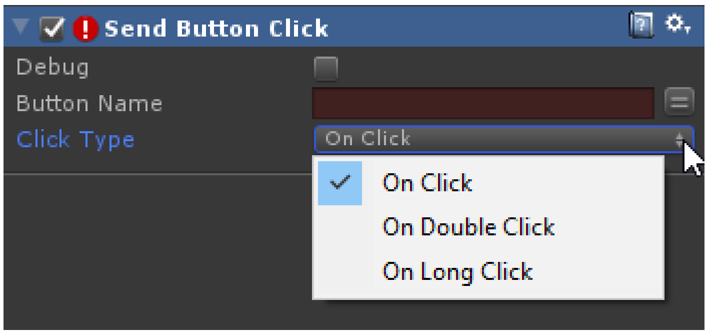
When the 'ApplicationQuit; game event has been issued it either exits play mode (if in editor) or quits the application (if in build).



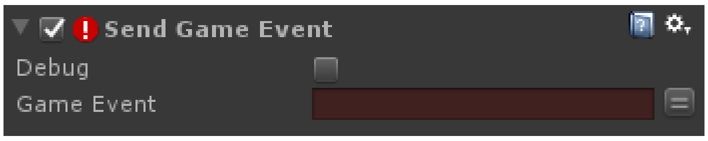
Disables the 'Back' button by adding another disable level to the additive bool.



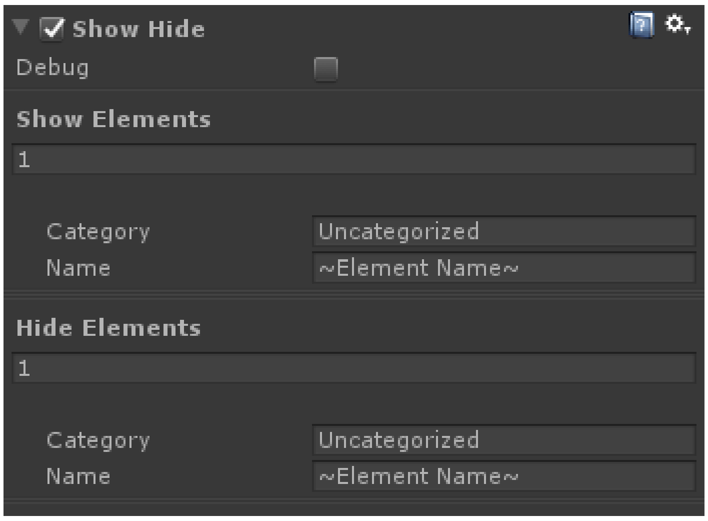
Enables the back button by subtracting another disable level from the additive bool. Forced enable clears the additive bool levels by resetting the bool level to 0.



Simulates a button click / double click / long click by sending the selected event along with the button name.

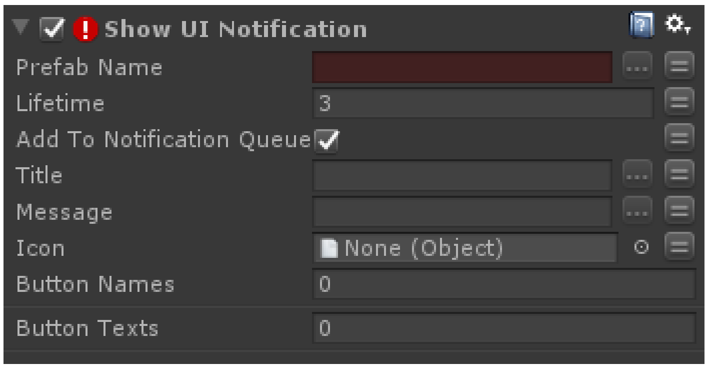


Sends a game event.

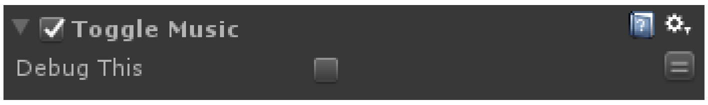

Show Hide
Debug
Show Elements
1
Category: Uncategorized
Name: ~Element Name~
Hide Elements
1
Category: Uncategorized
Name: ~Element Name~

Shows the pairs of category and name entered in the Show Elements array.

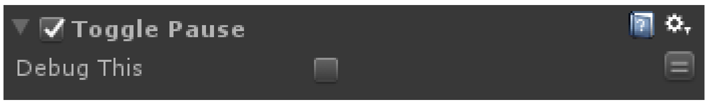
Hides the pairs of category and name entered in the Hide Elements array.


Show UI Notification
Prefab Name
Lifetime: 3
Add To Notification Queue: ✓
Title
Message
Icon: None (Object)
Button Names: 0
Button Texts: 0

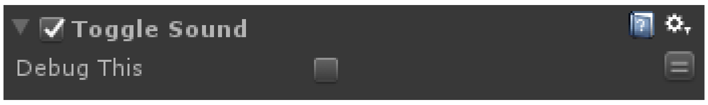
Shows an UINotification with the given settings. The UINotification needs to exist as a prefab either under a folder named 'Resources' or referenced to the UINotificatinManager.


Toggle Music
Debug This

Toggles the static bool used to determine if the music is on.


Toggle Pause
Debug This

Toggles the pause by changing the timescale automatically.


Toggle Sound
Debug This

Toggles the static bool used to determine if the sound is on.

UINavigation

The UINavigation is enabled by default and it can be disabled from Tools → DoozyUI → Control Panel → General Tab.

Its purpose is to keep track of all the shown and hidden UIElements that were added to the 'Navigation History'. It does that using the First-In-Last-Out (FILO) pattern (like a stack). With its help, by clicking a button named 'Back' or pressing the Escape key (also used as the back button on the Android platform), one can go back to previously shown and hidden UIElements.

There are special **button names** that will perform specific actions, as follows:

- **Back:** will go back in the Navigation History
- **TogglePause:** toggles the application's pause state (timescale)
- **ToggleSound:** toggles the static bool used to determine if the sound is on
- **ToggleMusic:** toggles the static bool used to determine if the music is on
- **ApplicationQuit:** exits play mode (if in editor) or executes Application.Quit

There are special **game events** that will perform specific actions, as follows:

- **ClearNavigationHistory:** clears the 'Navigation History'
- **DisableBackButton:** disables the back button functionality
- **EnableBackButton:** enables the back button functionality (enabled by default)
- **SoundCheck:** executes a sound check and sets the static bool to its previously saved value
- **MusicCheck:** executes a music check and sets the static bool to its previously saved value

Properties

Name	*Description*
IsNavigationEnabled	Returns true if the UINavigation is enabled and false otherwise. It is set to false if Scripting Define Symbols, for the current active platform, contain the 'dUI_NavigationDisabled' symbol. In you want to handle the UINavigation yourself just disable it from the Control Panel.

Methods

Name	*Description*
AddItemToHistory(NavigationPointerData)	Adds a navigation item to the end of Navigation History (FILO - First In Last Out).
ClearNavigationHistory()	Clears the Navigation History.
GetLastItemFromNavigationHistory(bool)	Returns the last item in the Navigation History. It removes the data from History by default.
Hide(List<NavigationPointer>, bool, bool)	Executes the Hide for the given list of Navigation Pointers.
PopToSpecificUIElement(string, string)	Pops to specific UIElement.
RemoveLastItemFromHistory()	Removes the last item from the Navigation History (FIFLO - First In Last Out).
Show(List<NavigationPointer>, bool)	Executes the Show for the given list of Navigation Pointers.
UpdateTheNavigationHistory(NavigationPointerData)	Updates the Navigation History while showing and hiding the relevant UIElements.

Final Words

YouTube Channel

You can find all the videos related to DoozyUI, that were created by Doozy Entertainment, at youtube.com/c/DoozyEntertainment

Twitter Feed

The Twitter handle used by Doozy Entertainment is **'doozyplay'**. You can view our Twitter feed by going to twitter.com/doozyplay

Facebook Page

The Facebook page used by Doozy Entertainment is **'doozyentertainment'**. You can view our Facebook page by going to facebook.com/doozyentertainment

Help Center

The Help Center for DoozyUI can be found at doozyentertainment.zendesk.com.

There you will be able to access an online (searchable) **Documentation**, **Tutorials** and **FAQ** sections.

Also, there is a Community area where you can submit **Feature Requests** or open **General Discussion** topics at doozyentertainment.zendesk.com/hc/en-us/community/topics

Support Ticket

You can open a **support ticket** by sending an email to support@doozyentertainment.com