DOOZY UI

COMPLETE UI MANAGEMENT SYSTEM

Owner's Manual

Contents

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.

Thank You

We would like to thank you for buying DoozyUI as it helps us develop this product even further.

Should you have any suggestions or find any bugs (or solutions) please let us know so that we can improve the system for you and anyone that uses it.

You can get in touch via email at support@doozyentertainment.com or through the Unity Forums at

https://forum.unity3d.com/threads/doozyui-complete-ui-management-system.397474

Thanks!
Doozy Entertainment

Social

Unity Forum: https://forum.unity3d.com/threads/doozyui-complete-ui-management-system.397474

YouTube Channel: http://www.youtube.com/c/DoozyEntertainment

Twitter: https://twitter.com/doozyplay

Facebook: https://www.facebook.com/doozyentertainment

What can DoozyUI do for you?

- ✓ Speed up UI design and implementation time by using a set of intuitive components.
- ✓ Manage all the UI from one location while giving you a plethora of options to interract with it.
- ✓ Manage automatically the UI navigation by creating a navigation history and handling it using the First In -> Last Out (FILO) system. You can also handle the navigation yourself, especially if you are using a FSM system like PlayMaker.
- ✓ Manage the loading and unloading of scenes by using the provided SceneLoader. It also supports EnergyBarToolkit, giving you the option to show any type progress bar while a scene is loading.
- ✓ Manage an in-game notification system, using a queue system. (Example: if the player earns 3 achievements at once, the notifications will appear one after another, not all at once)
- ✓ Trigger anything from anywhere by using UITriggers. They can react to button clicks or game events and trigger any method or set of methods you want, by using native Unity Events.
- ✓ Trigger from IN and OUT animations one or more methods, using native Unity Events, @START and/or @FINISH.
- ✓ Animate any uGUI component by using a responsive animator system that calculates the animation data, taking into account the actual screen size (be it Landscape or Portrait).
- ✓ Use premade or custom animation presets. Any presets you create ca be reused in future projects, as they are saved as .xml files.
- ✓ Handle without any code the 'Back' button event.
- ✓ Handle without any code Sound and Music state (ON/OFF), also saving and loading to and from PlayerPrefs their state.
- ✓ Handle without any code Pause/Unpause.
- ✓ Handle wihtout any code ApplicationQuit (or exit Play mode).
- ✓ Integrade PlayMaker, MasterAudio and EnergyBarToolkit in an elegant manner.
- ✓ Give you a better understanding of how the native Unity UI components work by watching the tutorial videos.
- ✓ Give you a performace boos on mobile. As it has a very low impact on performance being mobile friendly! ☺

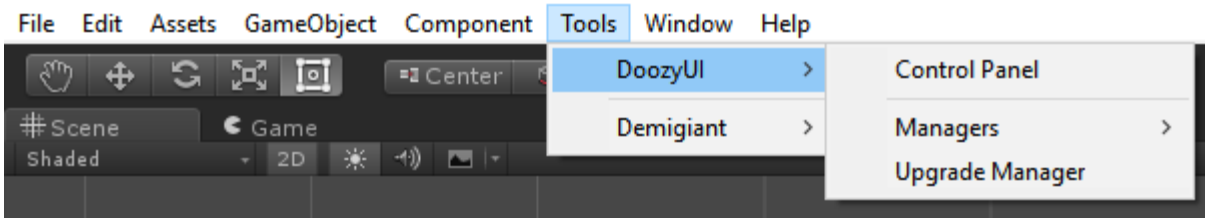
Quick Start

1. Import DOTween
2. Setup DOTween (from Tools > DOTween Utility Panel > [Setup DOTween])
3. Import DOOZY UI
4. Done! ☺

Upgrade Guide

If are upgrading from a version prior to 2.8 you need to do the following:

1. Import DOOZY UI
2. Wait for the Automated Upgrade Manager to process the databases
3. Open the Upgrade Manager from Tools > DoozyUI > Upgrade Manager



4. Open each scene that contains DoozyUI components and upgrade it with the help of the Upgrade Manager Window. Press [Upgrade Current Scene] for each scene.

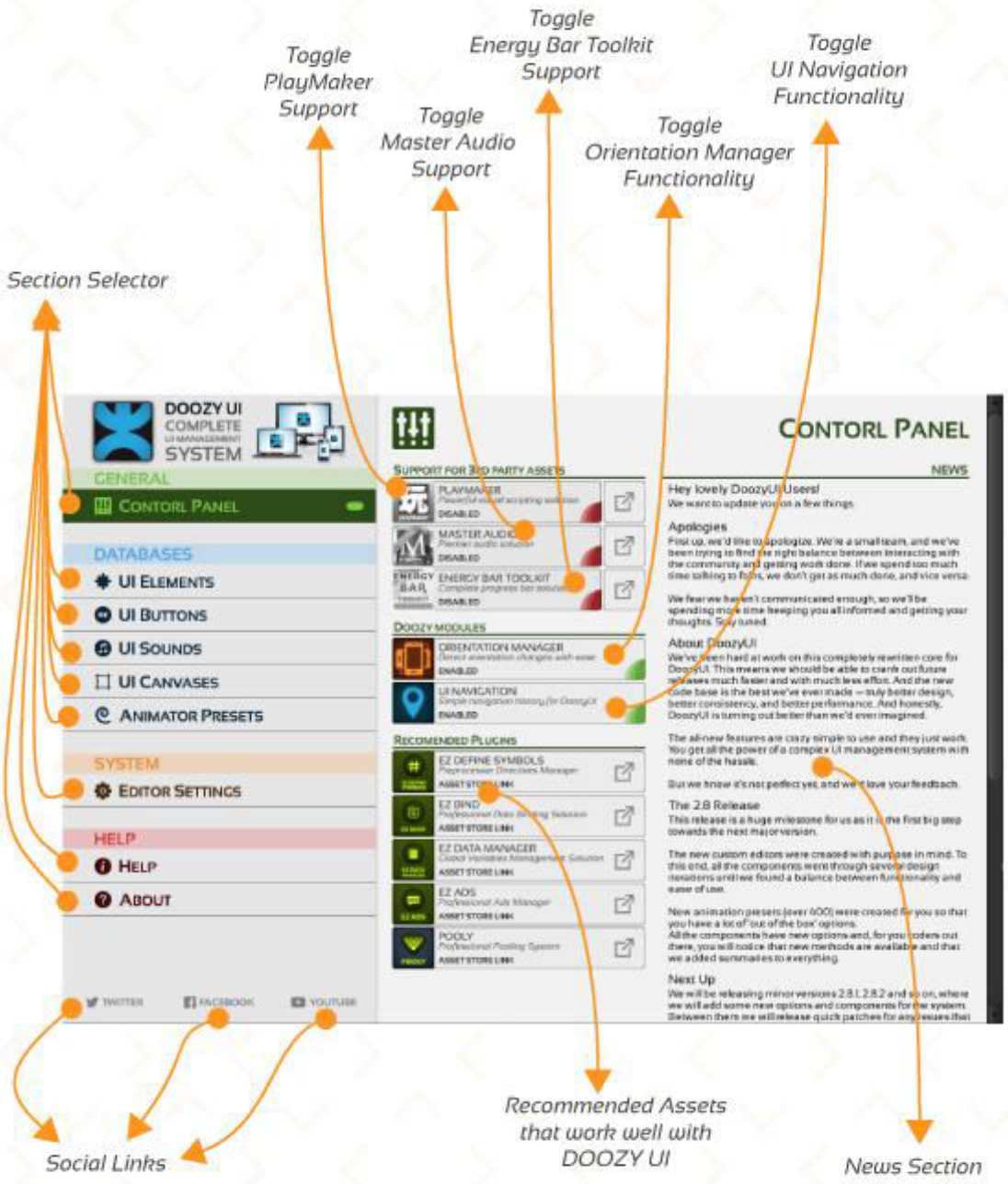
5. Once all the scenes have been upgraded you need to take a look at each component to be sure that all the settings look as expected. Since this is a new core with a new interface, the upgrade is not failproof and minor issues may arrise. Please understand that we did all we could in order to make the upgrade process as painless as possbile. So be prepared to work a bit in order to get all your settings as before the update.

Control Panel

The Control Panle is your main interface with the system. You will be using it a lot during development and it contains all the system settings and databases.

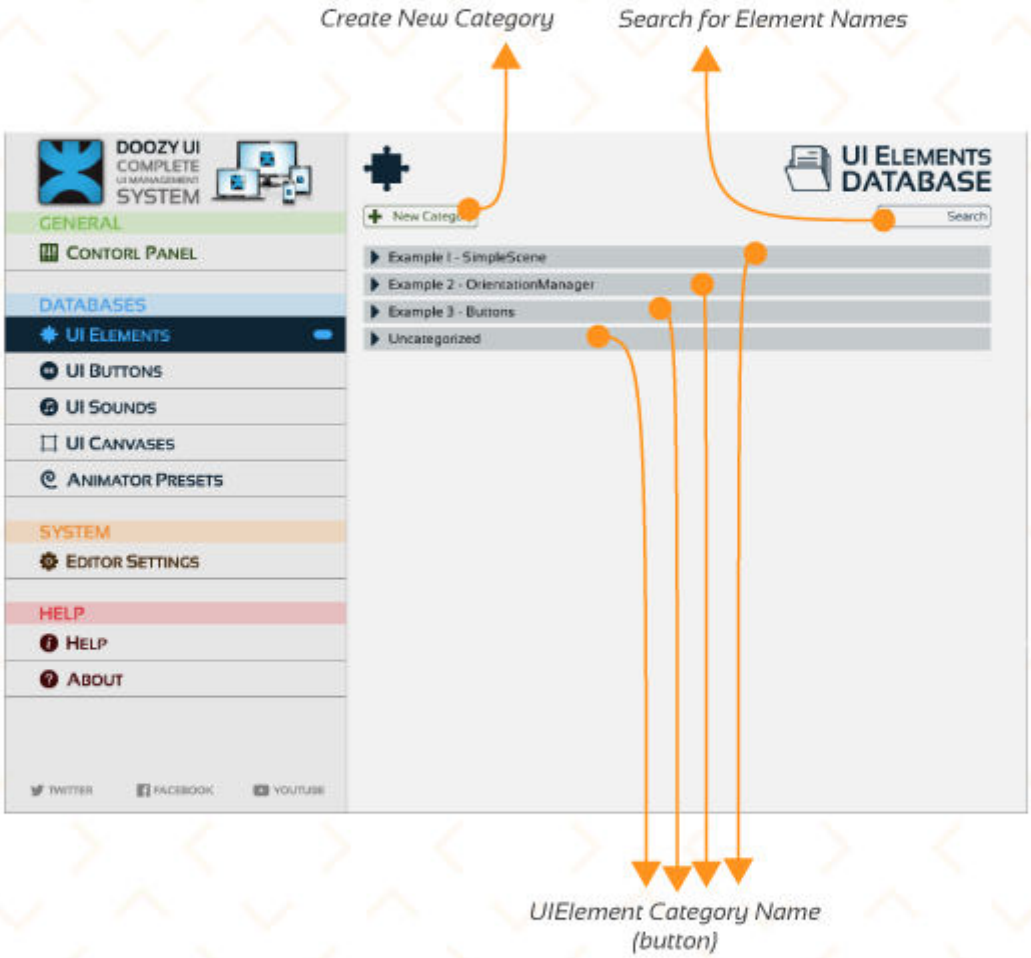
Control Panel

1. Toggle support for PlayMaker, Master Audio and Energy Bar Toolkit
2. Toggle the funtionality of the Orientation Manager and of the UI Navigation
3. Look at some recommended assets that work well with DoozyUI
4. Read the News about the system



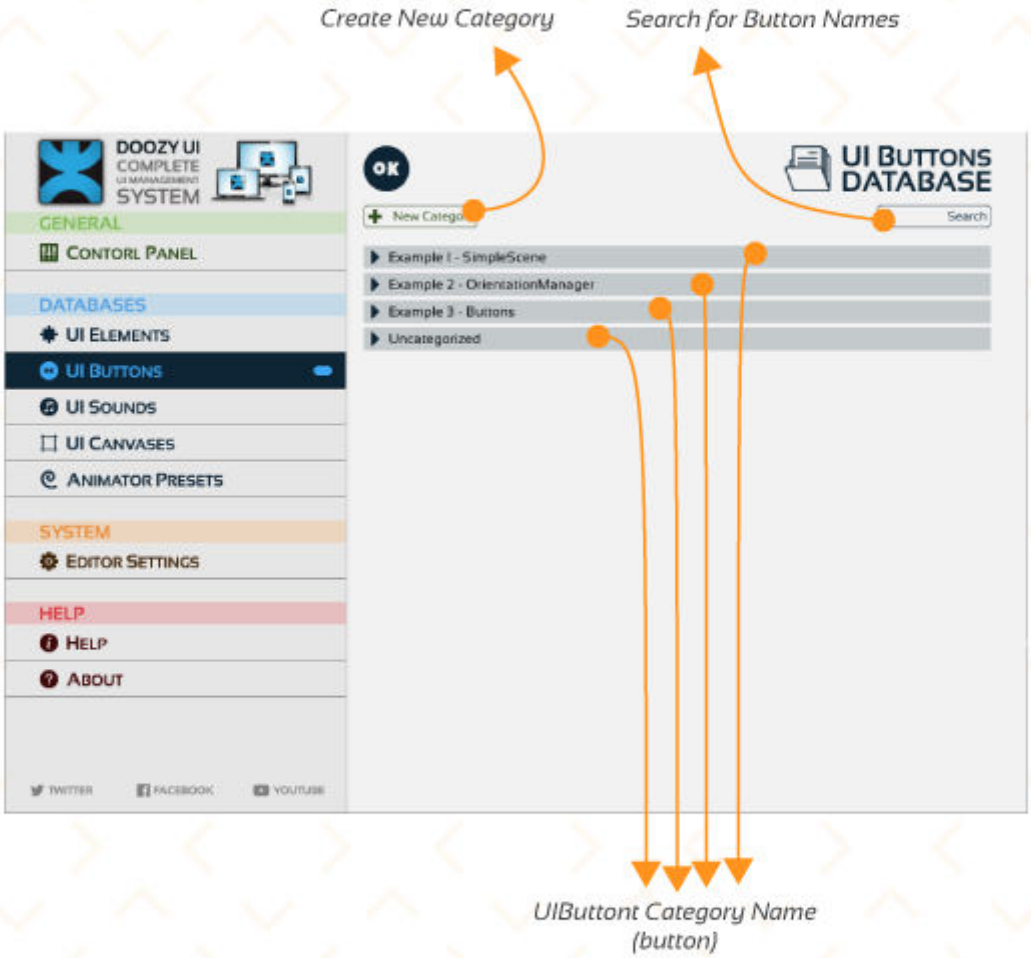
UIElements Database

Here you can create, edit and delete UIElement element categories and element names.



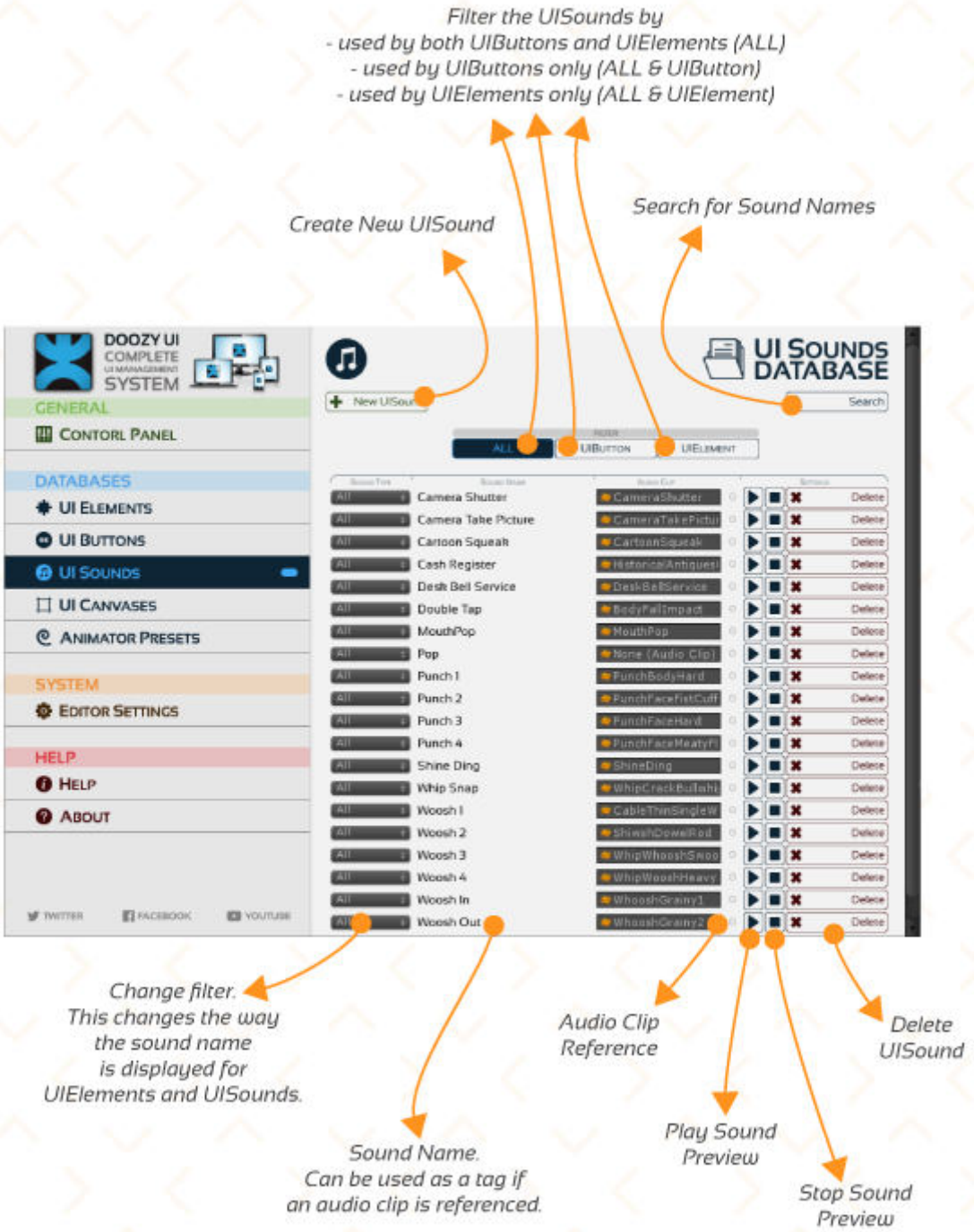
UIButtons Database

Here you can create, edit and delete UIButton button categories and button names.



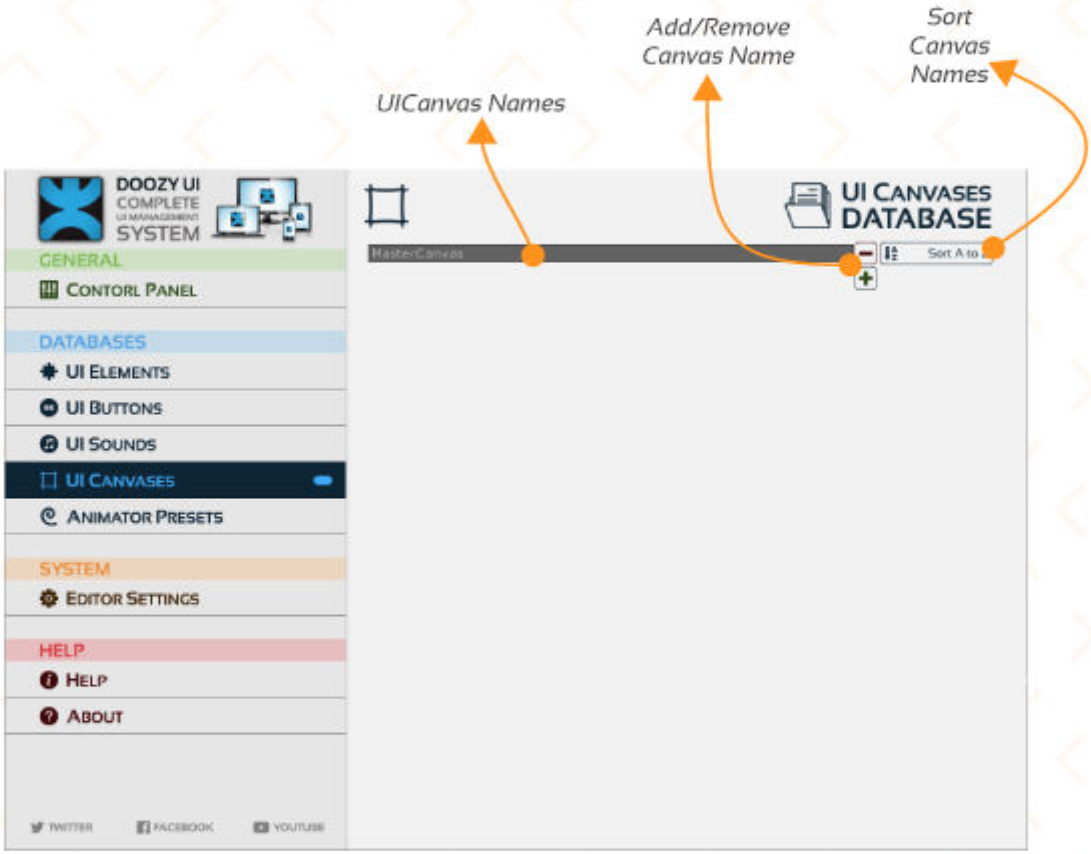
UISounds Database

Here you can create and delete UISounds. You also have a filter option available and sound preview buttons. Each UISound can have a name and an AudioClip reference. If an Audio Clip is not referenced, Soundy (the sound manager) will try to search for a soundfile under a Resources folder. If that fails, no sound will play.
You can use the sound name as a tag for an AudioClip.



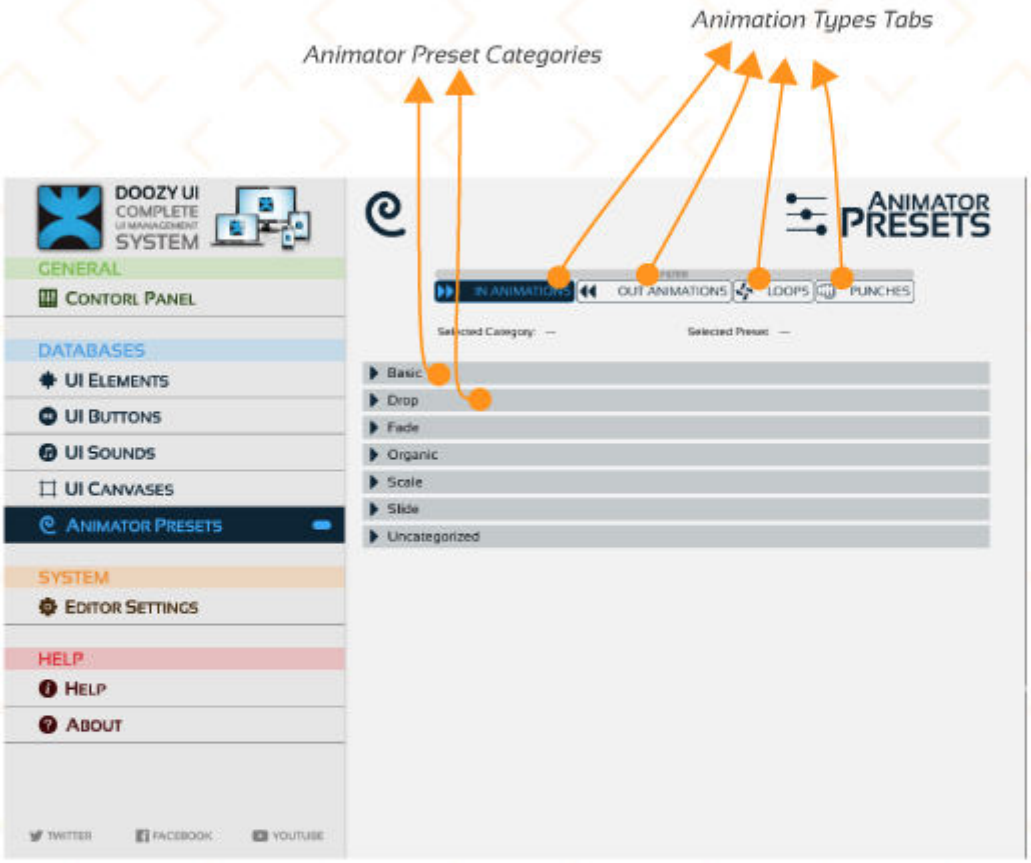
UICanvases Database

Here you can create and delete canvas names. These are the names used to identify UICanvases.
The canvas name is mostly used by the UINotifications so that they know what canvas to be shown on.
The UICanvas also helps with resetting the sorting layer names of all of its children gameobjects.



Animator Presets

The Animator Presets allow you to view and change any type of preset. You can only see on preset type at one time, and you can select the type from the top tabs (In Animations, Out Animations, Loops and Punches).
All the changes made here are saved in realtime for the loaded preset. Edit with care!

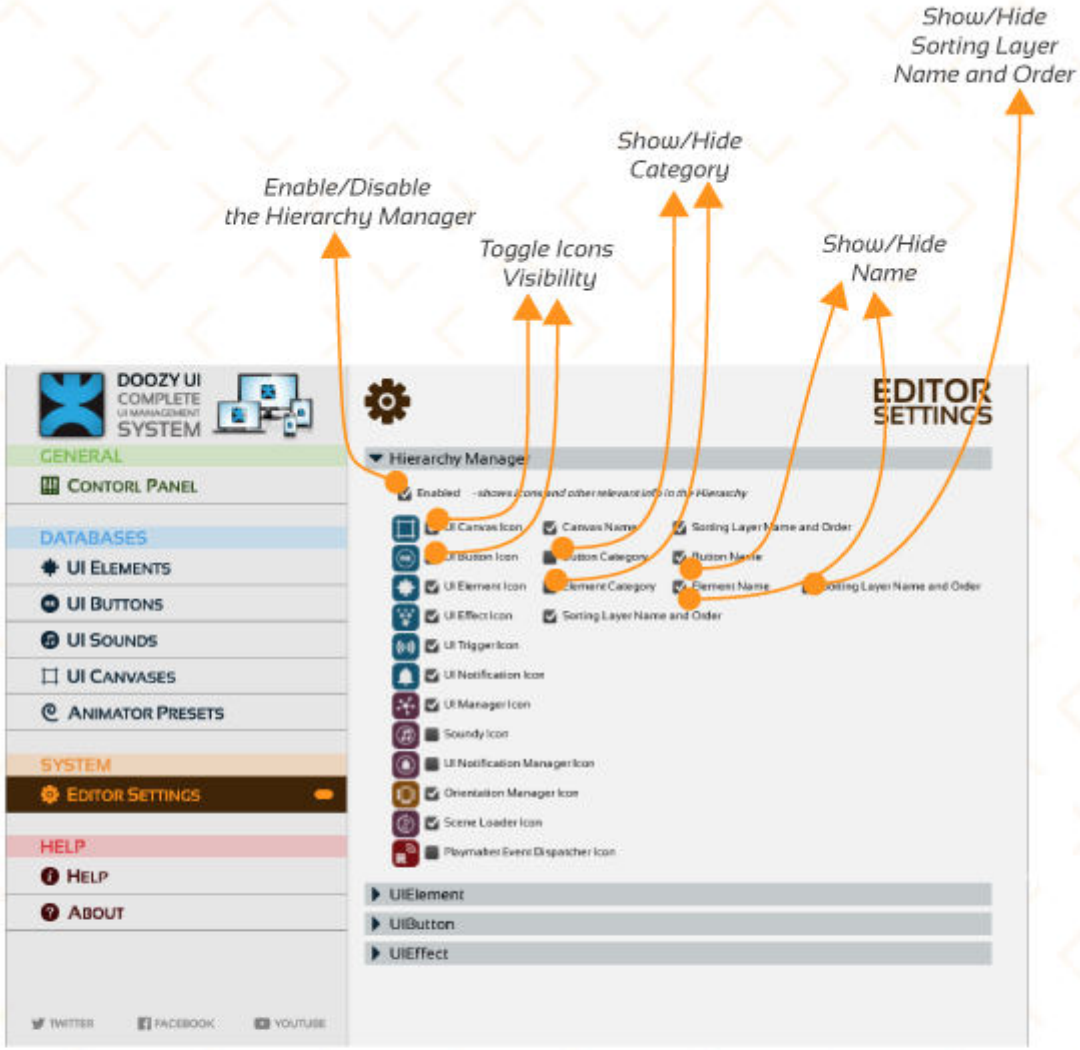


Editor Settings

These settings are used for the Unity Editor only as they are irrelevant for builds.

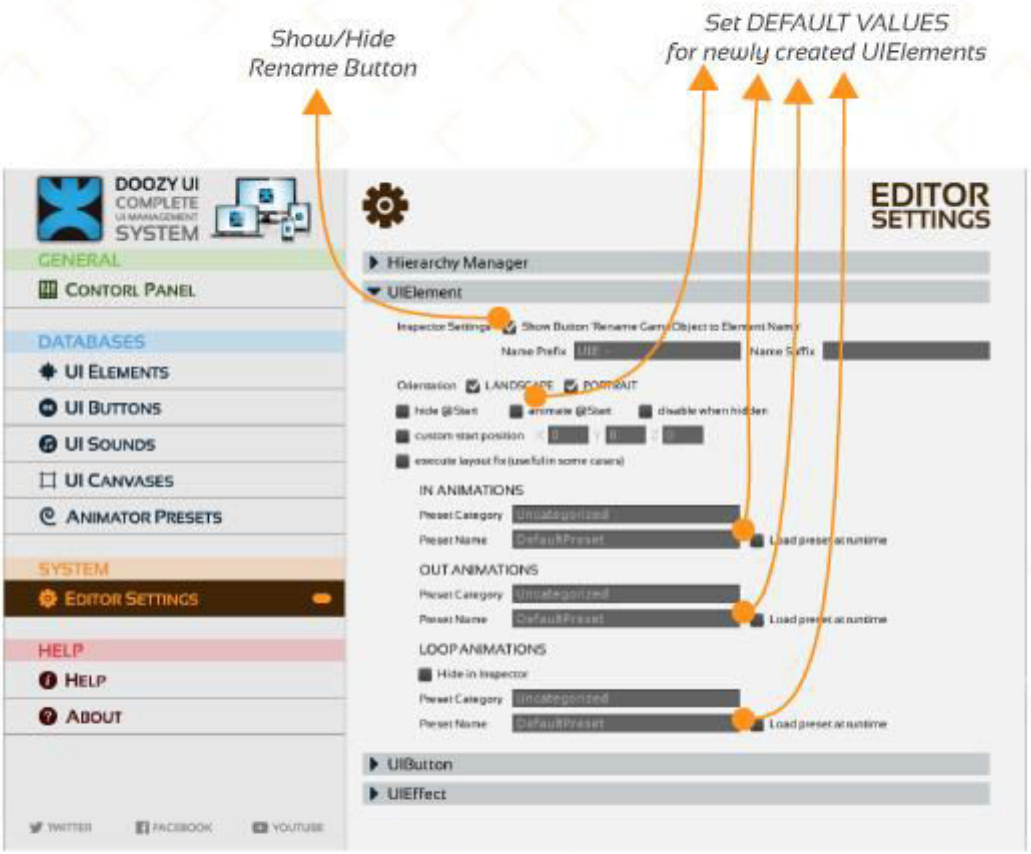
Hierarchy Manager

The Hierarchy Manager will show icons and other relevant informations to the right of any DoozyUI component in the Hierarchy View. You can toggle the manager on or off and any of its effects to suit your liking.



UIElement

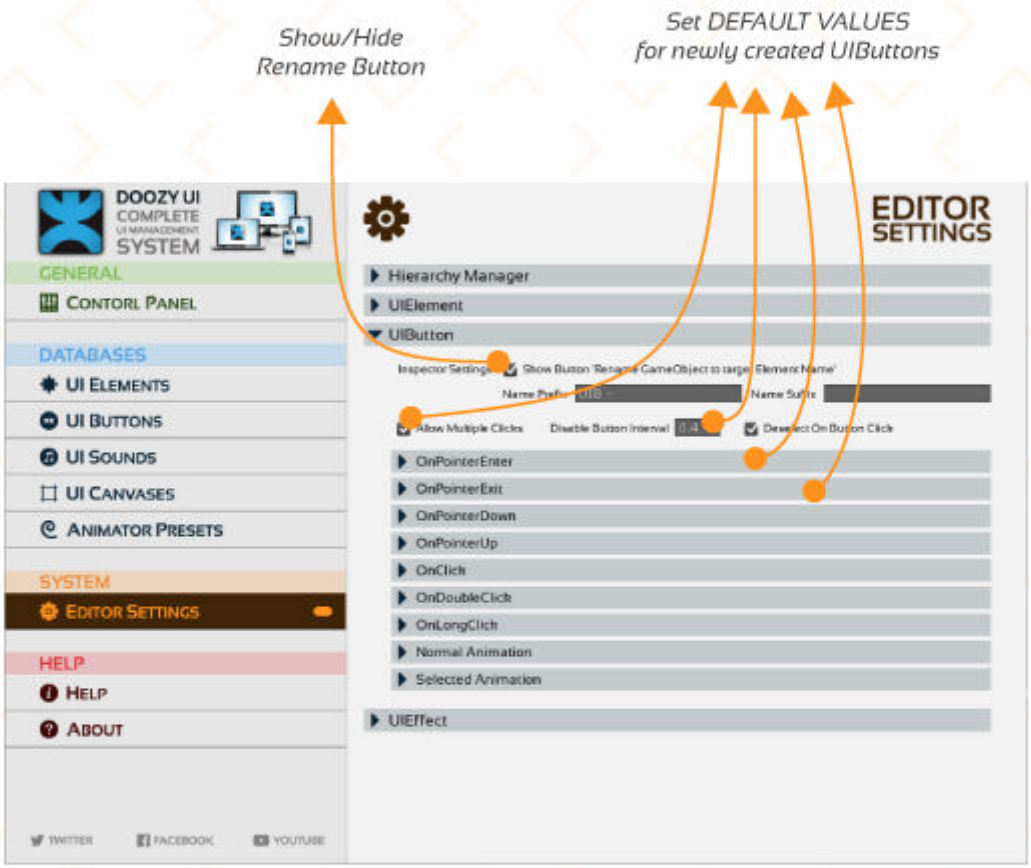
Here you can hide the Rename Button or the Loop Animations sections, should you not want to use them.
All the other settings are the Default Values that an UIElement will get when created. This will help you create your UI even faster as you may set the animation presets and other settings from the get go.



Show/Hide in Inspector
the Loop Animations Section

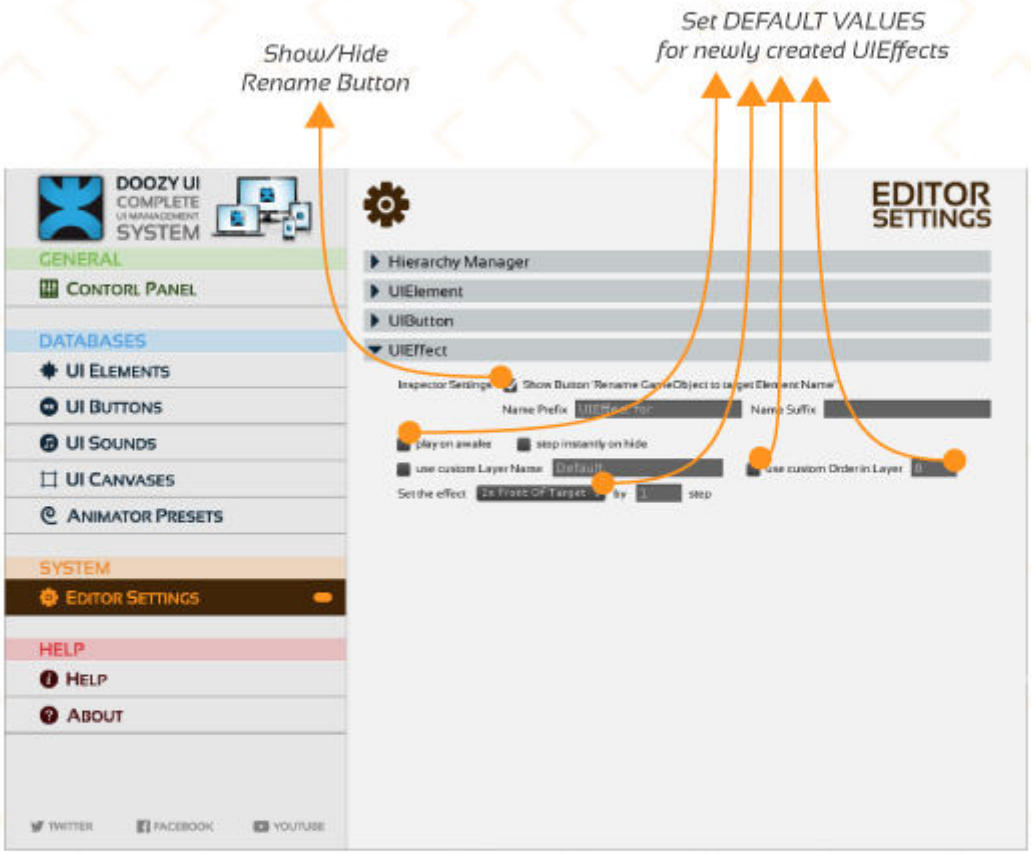
UIButton

Here you can hide the Rename Button or any unused tabs (like OnPointerEnter, OnPointerExit and so on…), should you not want to use them.
All the other settings are the Default Values that an UIButton will get 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.



UIEffect

Here you can hide the Rename Button, should you not want to use it.
All the other settings are the Default Values that an UIEffect will get 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.



UI Manager

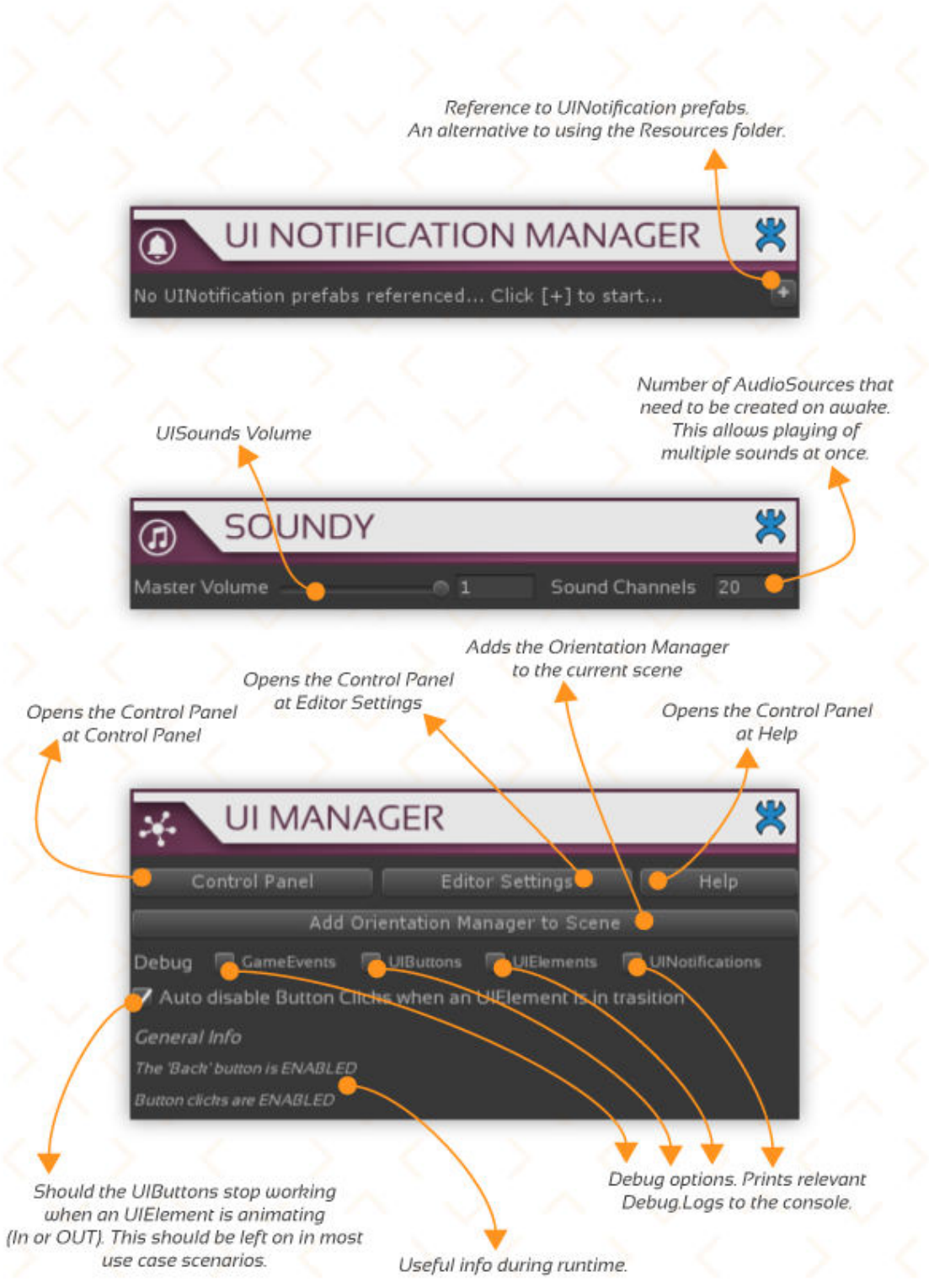
Is the core of DoozyUI as it 'communicates' with everything. You wil interract with it a lot in code and you should talke a look at the Api Documentation to see all the avaialble options (there are a lot).

Soundy

Is a simple sound manager that plays UISounds, Audio Clips and sound files located under a Resources folder.

UI Notification Manager

It keeps references to any UINotification prefabs, being an alternative to loading a notification from a Resources folder. Also it has all the logic to configure, load and show UINotifications.

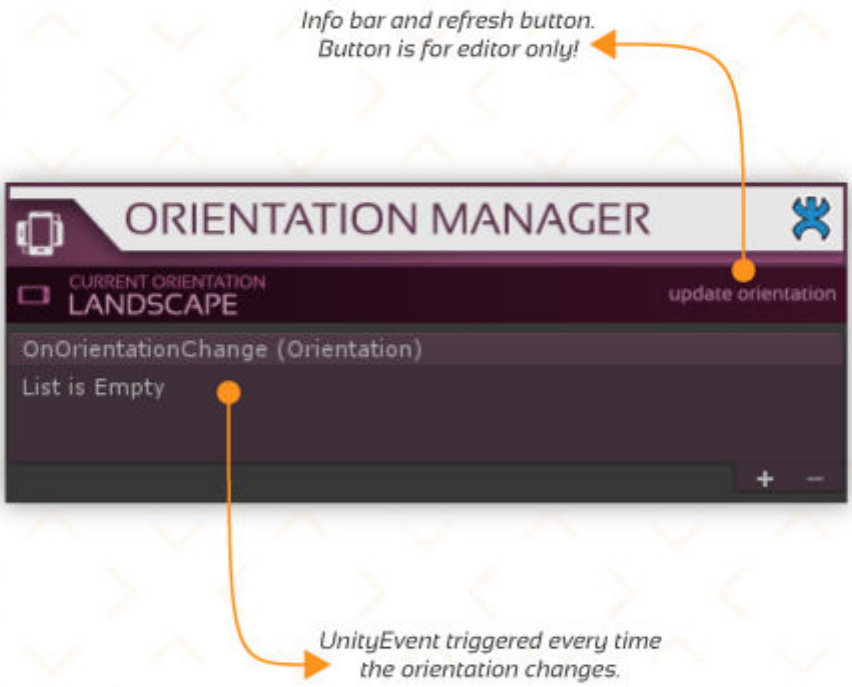


Orientation Manager

This is a simple, yet very efficient, implementation of an orientation detector. You just enable it from the Control Panel and forget about it. It will work like magic (with zero overhead).

Should you need to know when an orientation change occurs, then you can manually add it to the scene and hook yourself up to the provided UnityEvene.

Hint: click the bar to update the orientation in Edit Mode



Scene Loader

A simple implementation of a scene loading solution that is integrated with DoozyUI. You can use game events to trigger the loading and unloading of scenes.

If EnergyBarToolkit support is enabled another option will be available that will allow you to link progress bars. You can link as many loading bars you need to show the loading progress of the scenes.

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");

LoadSceneAdditiveAsybc_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");

LoadSceneAdditiveAsybc_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");

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");

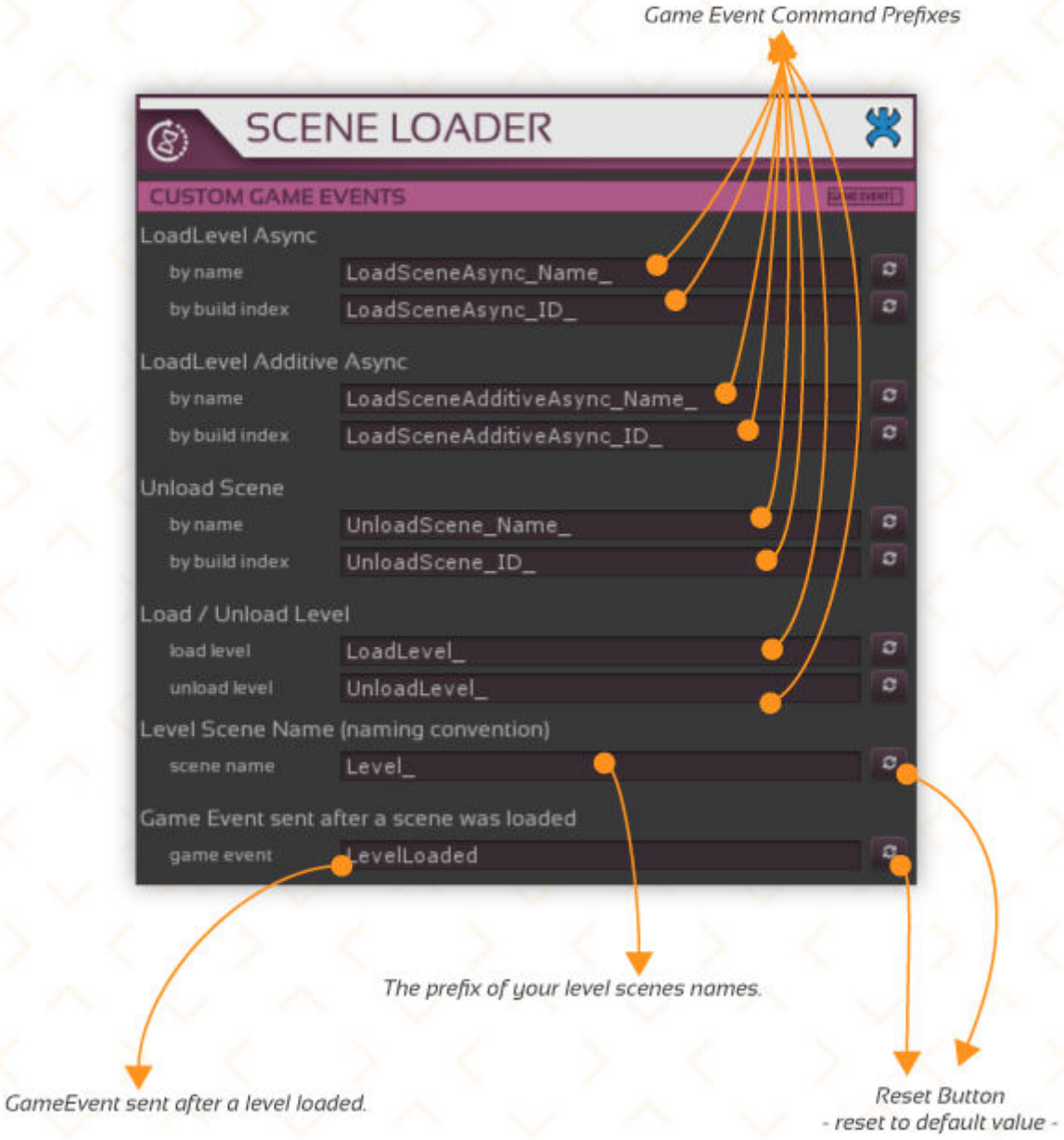
UnloadScene_ID_[buildIndex]
Usage example: To unload the 5th scene in your build index you need to send a game event with the command 'UnloadScene_ID_5', where 'UnloadScene_ID_' is the first part of the command and '5' is the build index number of the scene you want to unload.
Example code: UIManager.SendGameEvent("UnloadScene_ID_5");

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");

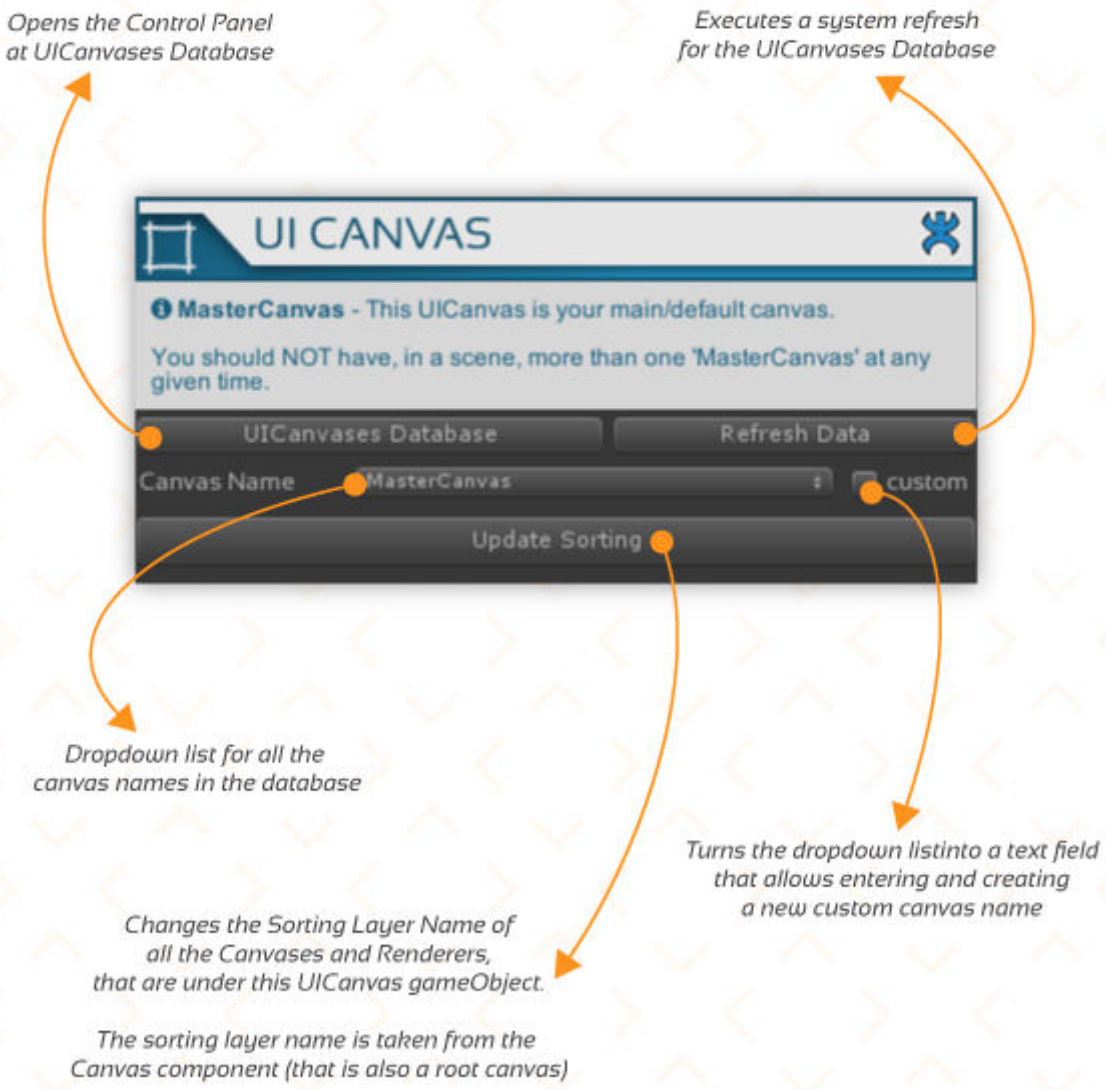
Level Scene Name
This is the name for your level scenes in build. Example: 'Level_1', 'Level_2' ... 'Level_100'
This is a simple, yet very efficient, implementation of an orientation detector. You just enable it from the Control

If you want to load a scene, you can do that just by sending a GameEvent. To load the scene in a certain way, you can build your game event string with a prefix. Here you can customize those prefixes.



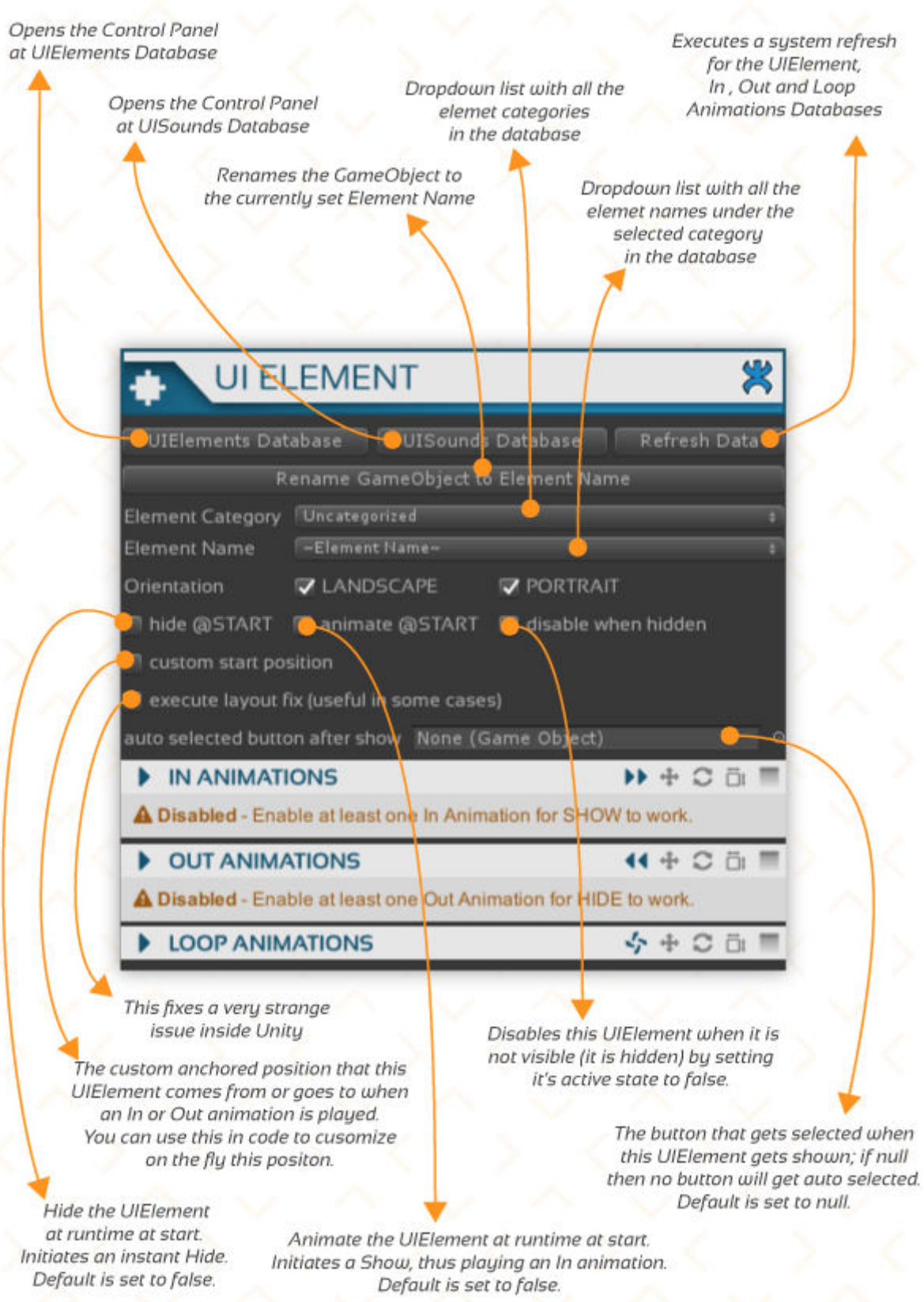
UI Canvas

This component's main purpose is to give the UINotification a target conainer when it is shown.
It does other fancy stuff as well, like when creating an UIElement, an UIButton or an UINotification, it will pe parented to the selected UICanvas or to the Master Canvas.
It also helps changing the sorting layer name of all of it's children.



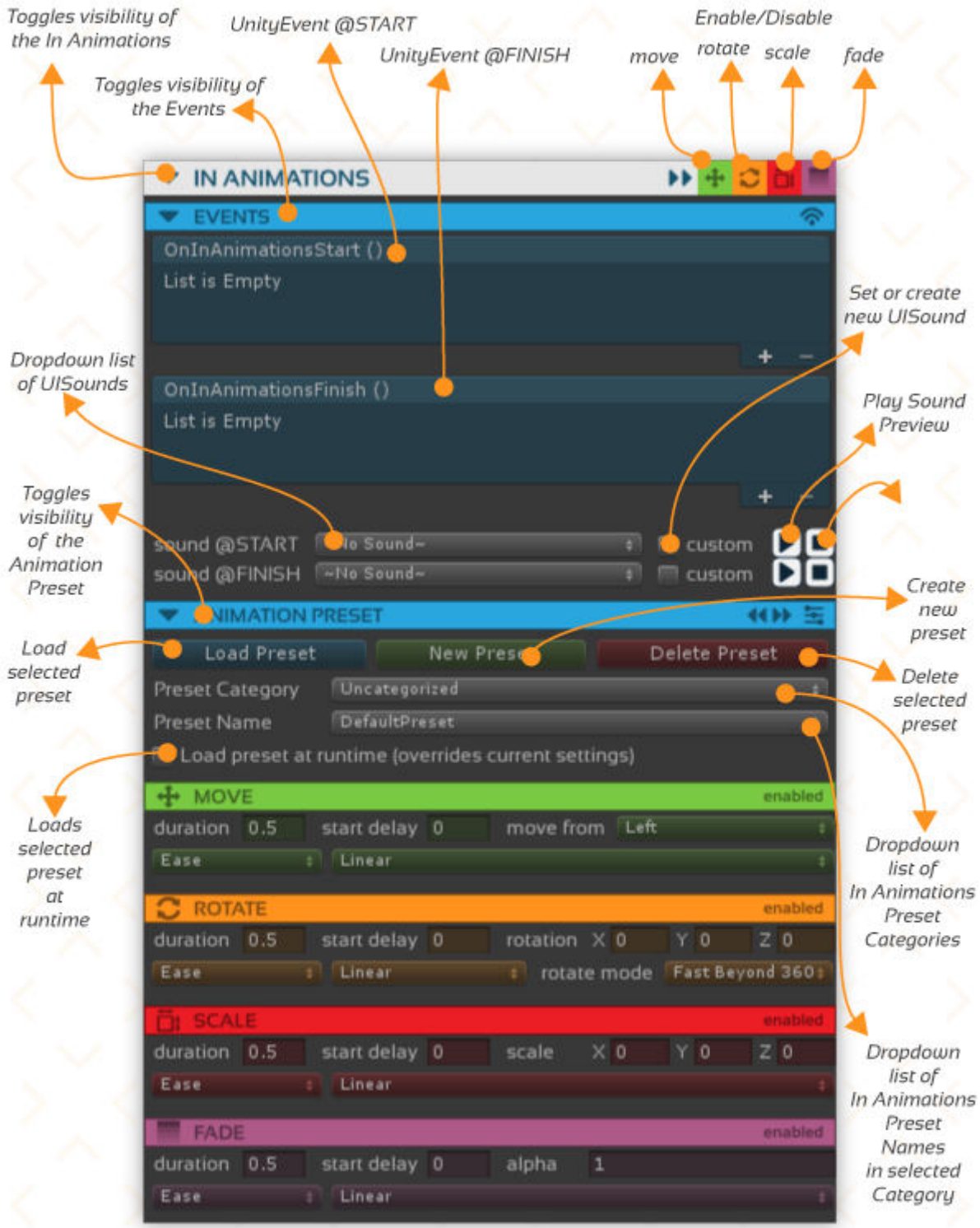
UI Element

This is one of the core components that you will be using extensively. It has a lot of options, but don't get scared, once you go through them, they are quite easy to use and understand.
A special option is the Orientation, LANDSCAPE / PORTRAIT are visible only if the Orientation Manager is enabled. You tick the orientation you want your UIElement to be visible on. It can be Landscape only, Portrait only or both (default).



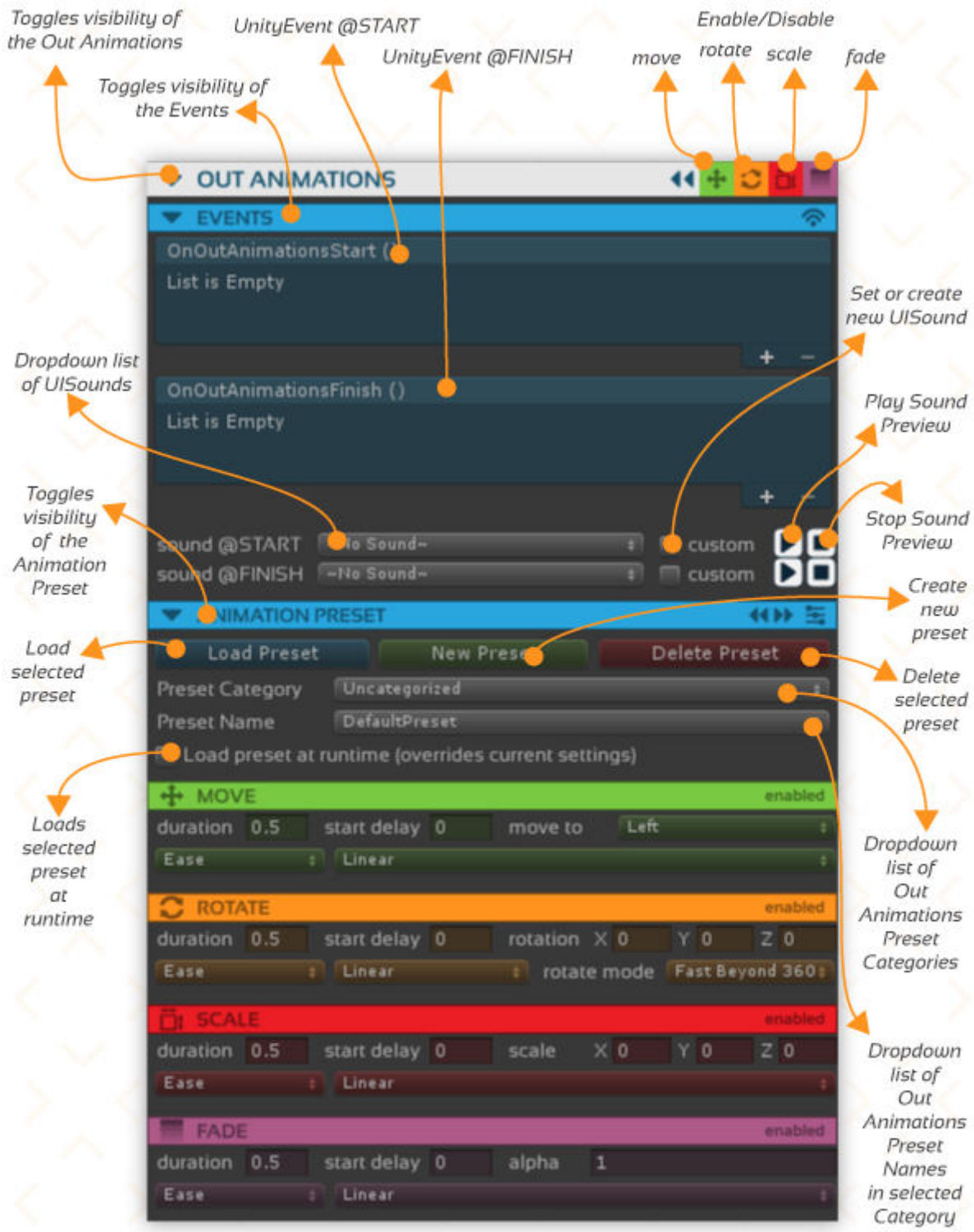
In Animations

These are the SHOW or ENTER animations. Look at the image below to get an idea of what everything does.



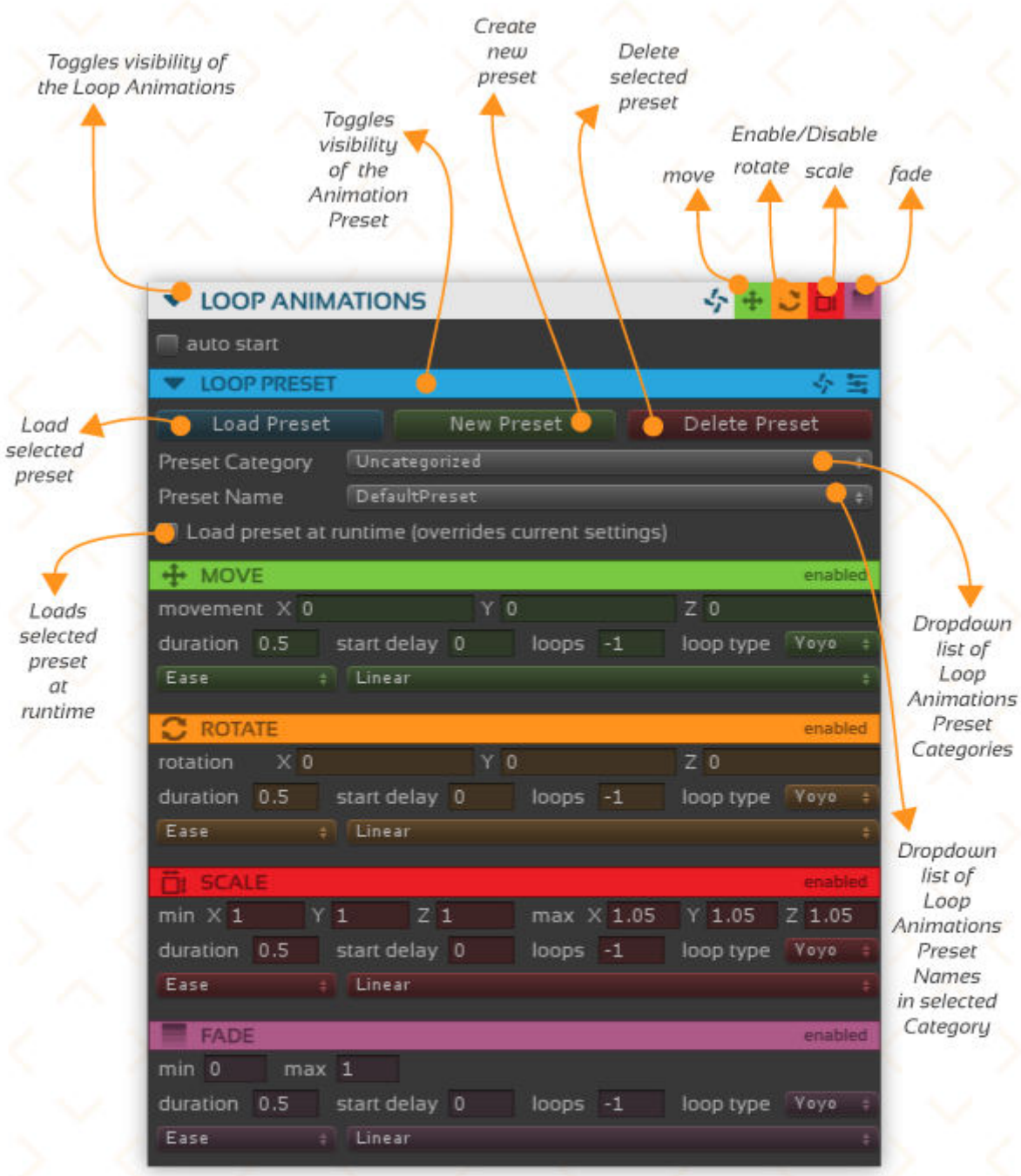
Out Animations

These are the HIDE or EXIT animations. Look at the image below to get an idea of what everything does.



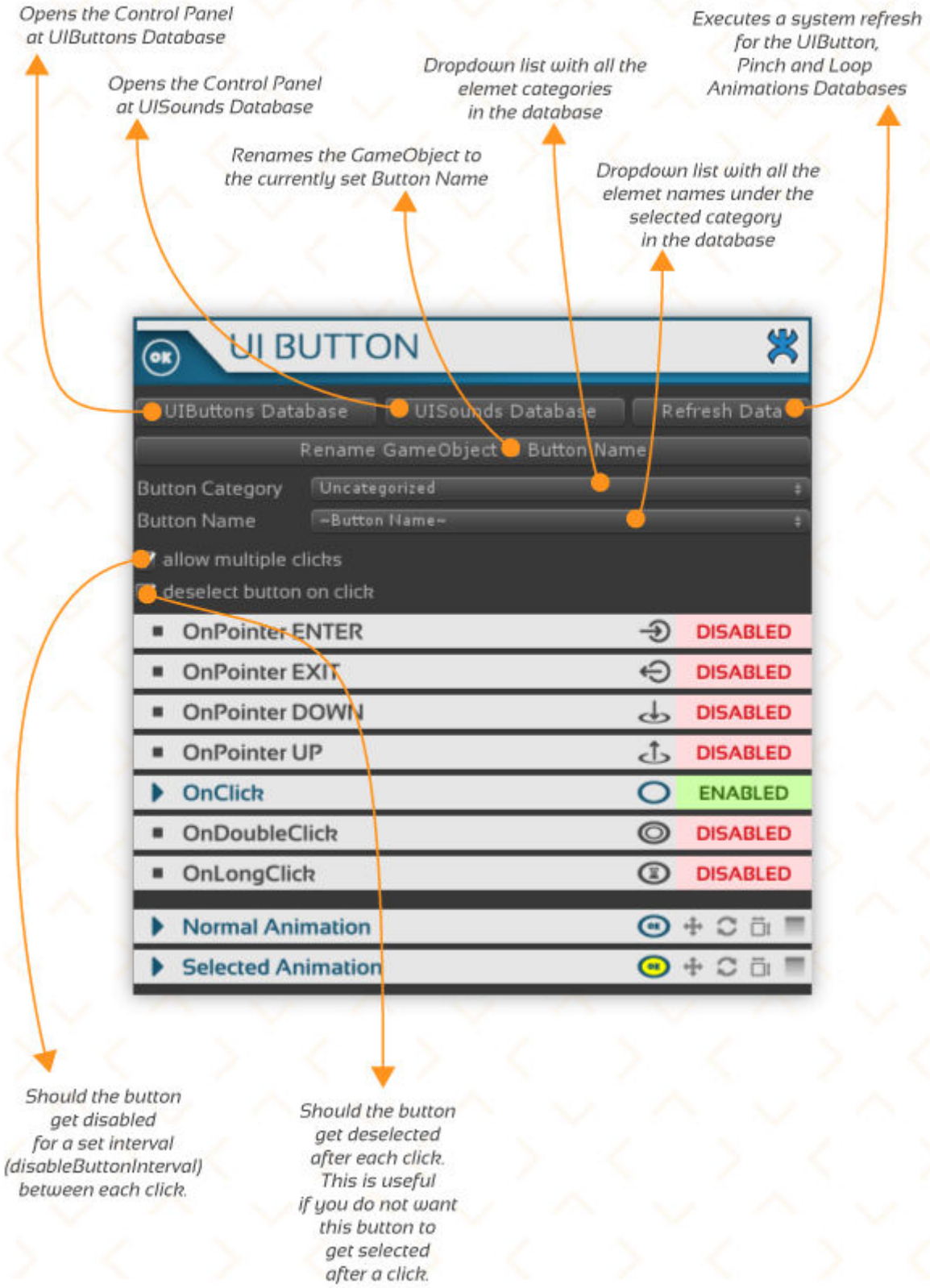
Loop Animations

These are the animations that run in a loop after the UIElement has been shown. They start automatically, if they are enabled, after a SHOW and stop automatically before a HIDE.
Should you need them to start by default (for example at runtime, for an UIElement that is already visible on screeen), just tick ‘auto start’ as true. This setting is also useful if you are not using In and Out animations.
Look at the image below to get an idea of what everything does.



UI Button

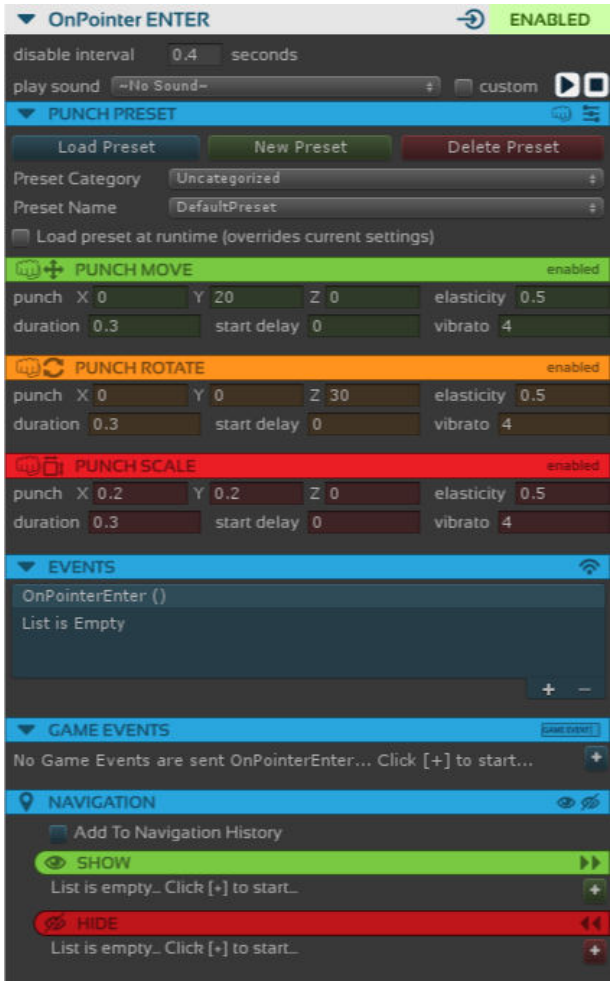
This is another one of the core components that you will be using extensively. It has a huge list of options, but don't get scared, once you go through them, they are quite easy to use and understand. And should you not need them all, you can hide them from the Control Panel > Editor Settings > UI Button.

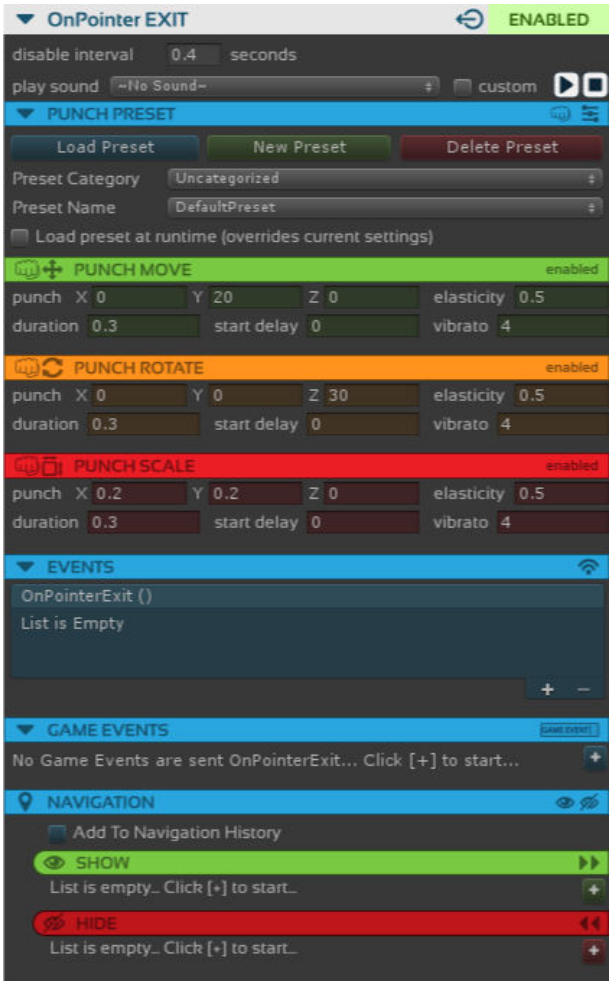


On Pointer Enter / On Pointer Exit

Please see image for details.

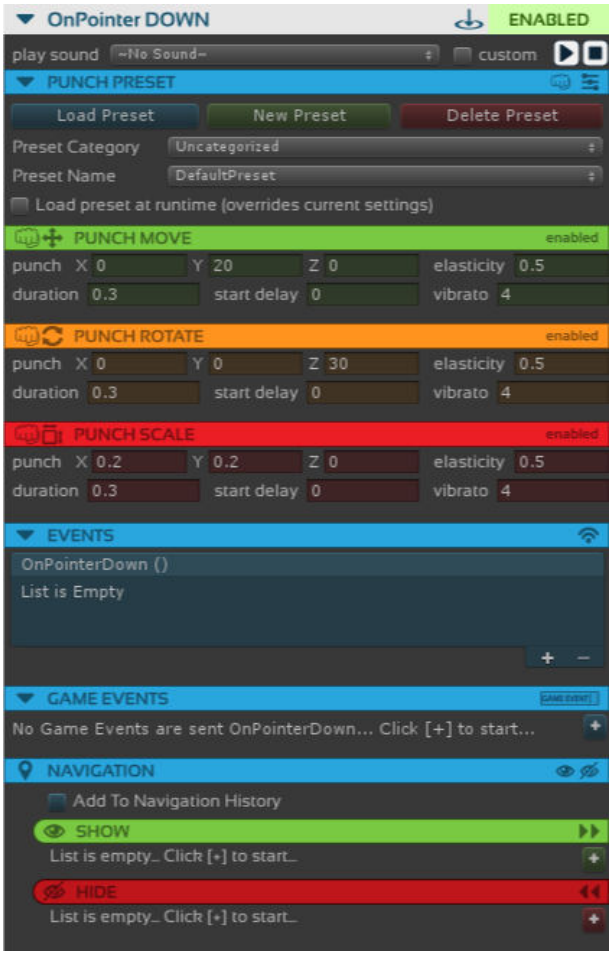
disable interval: after the pointer enters/exits, the functionality is disabled for the set number of seconds.

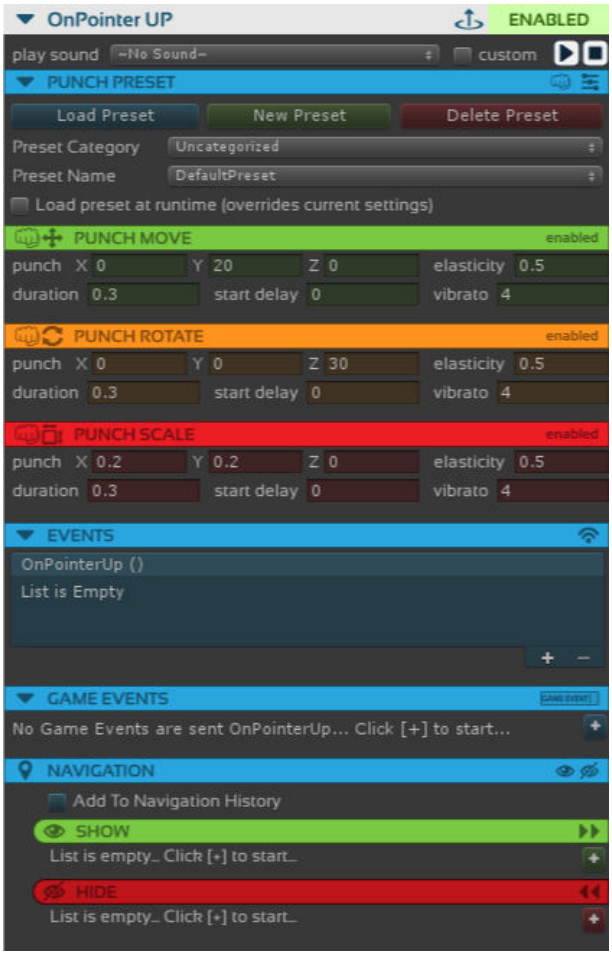




On Pointer Down / On Pointer Up

Please see image for details.



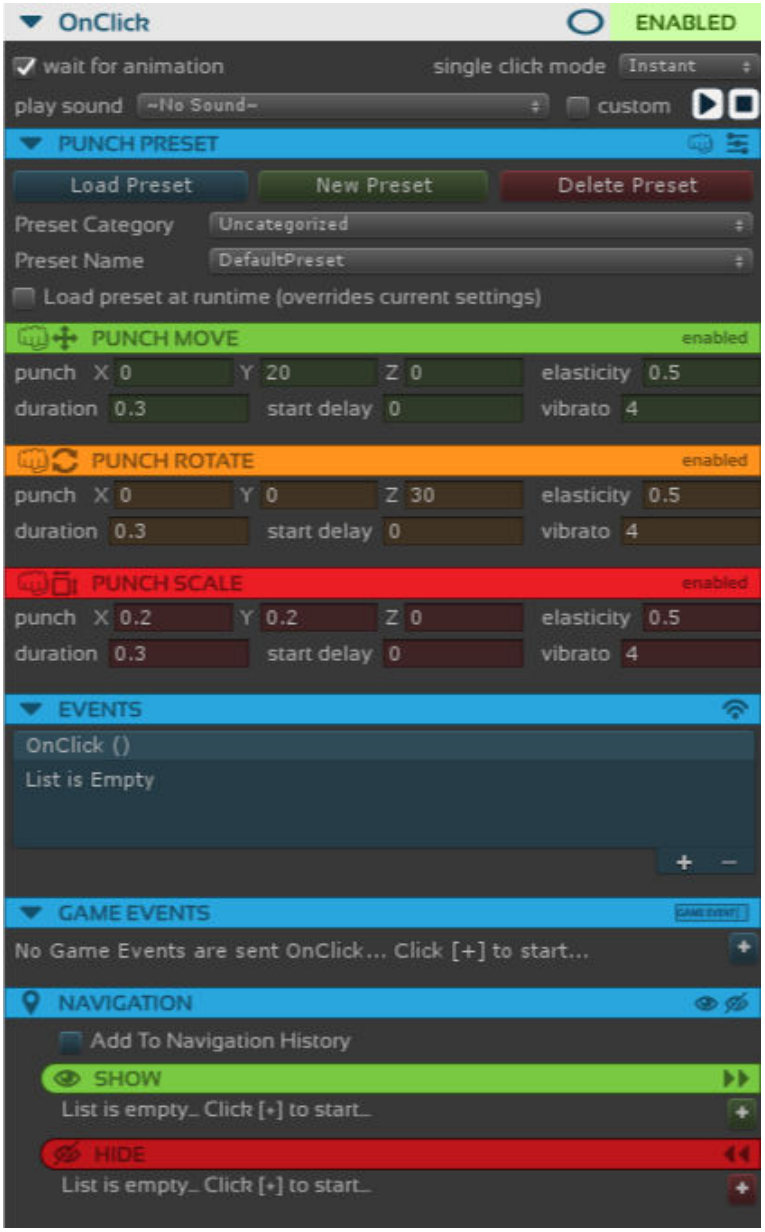


On Click

Wait for animation is enabled by default.. If true it will wait until the OnClick animation finised playing before sending the OnClick command (includes the show/hide and send game events actions). If false, it will send the OnClick command instantly without waiting for the animation to finish. Depends on the effect you are looking for.

Single click mode marks if the click should be registered instantly without checking if it's a double click or not.

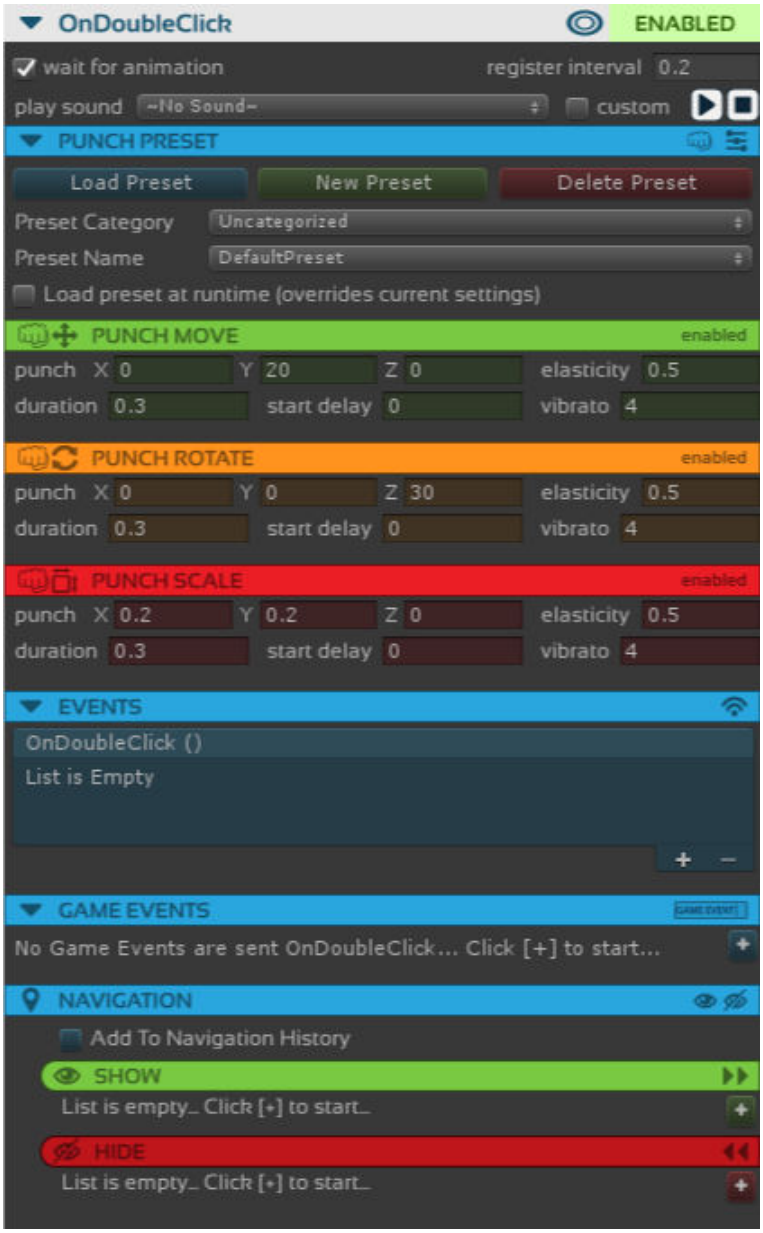
- ***Instant*** - the click will get registered **instantly** without checking if it's a double click or not. This is the normal behaviour of a single click in any OS. Use this if you want to make sure a single click will get executed before a double click (dual actions). Usage example: SingleClick - selects, DoubleClick - executes an action.
- ***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 donwside is that there is a delay when executing the single click (the delay is the doulbe click register interval), so make sure you take that into account



On Double Click

Wait for animation is enabled by default.. If true it will wait until the OnDoubleClick animation finised playing before sending the OnDoubleClick command (includes the show/hide and send game events actions). If false, it will send the OnDoubleClick command instantly without waiting for the animation to finish. Depends on the effect you are looking for.

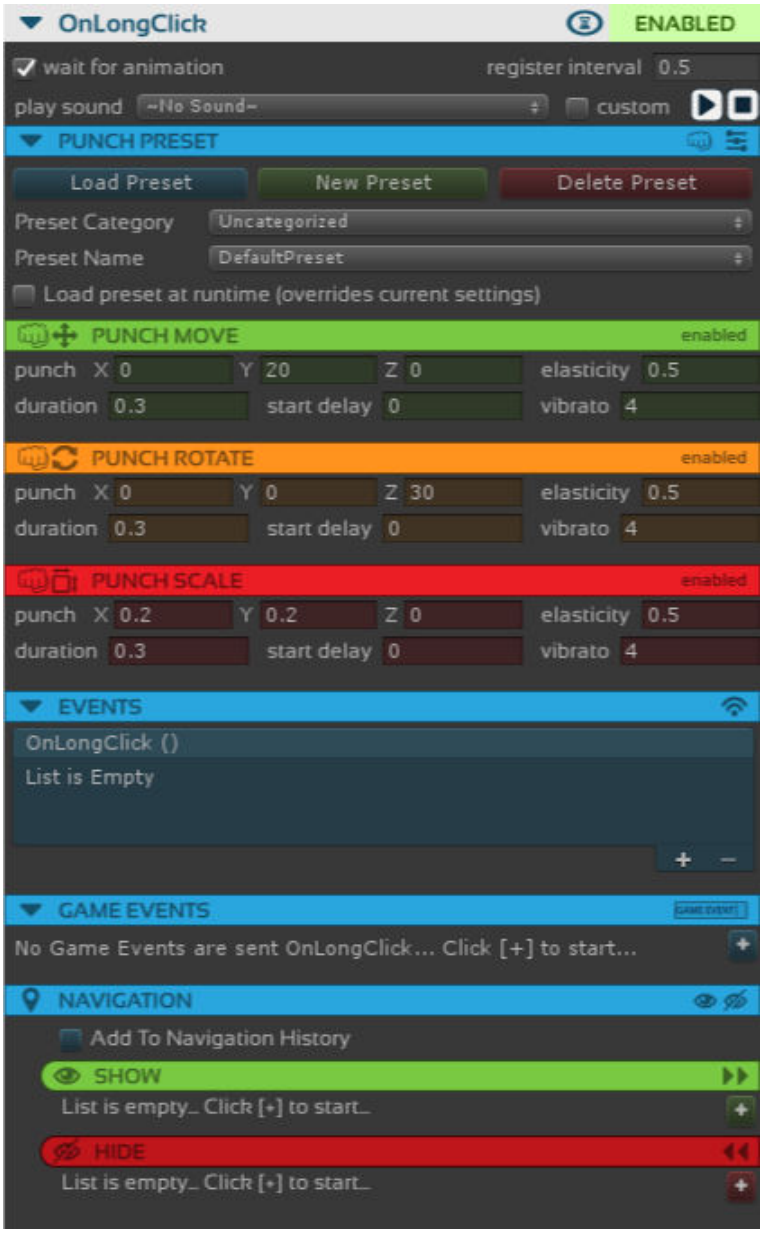
Register interval is the time interval used to register a double click. This is the time interval calculated between two sequencial clicks to determine if either a double click or two separate clicks occured.



On Long Click

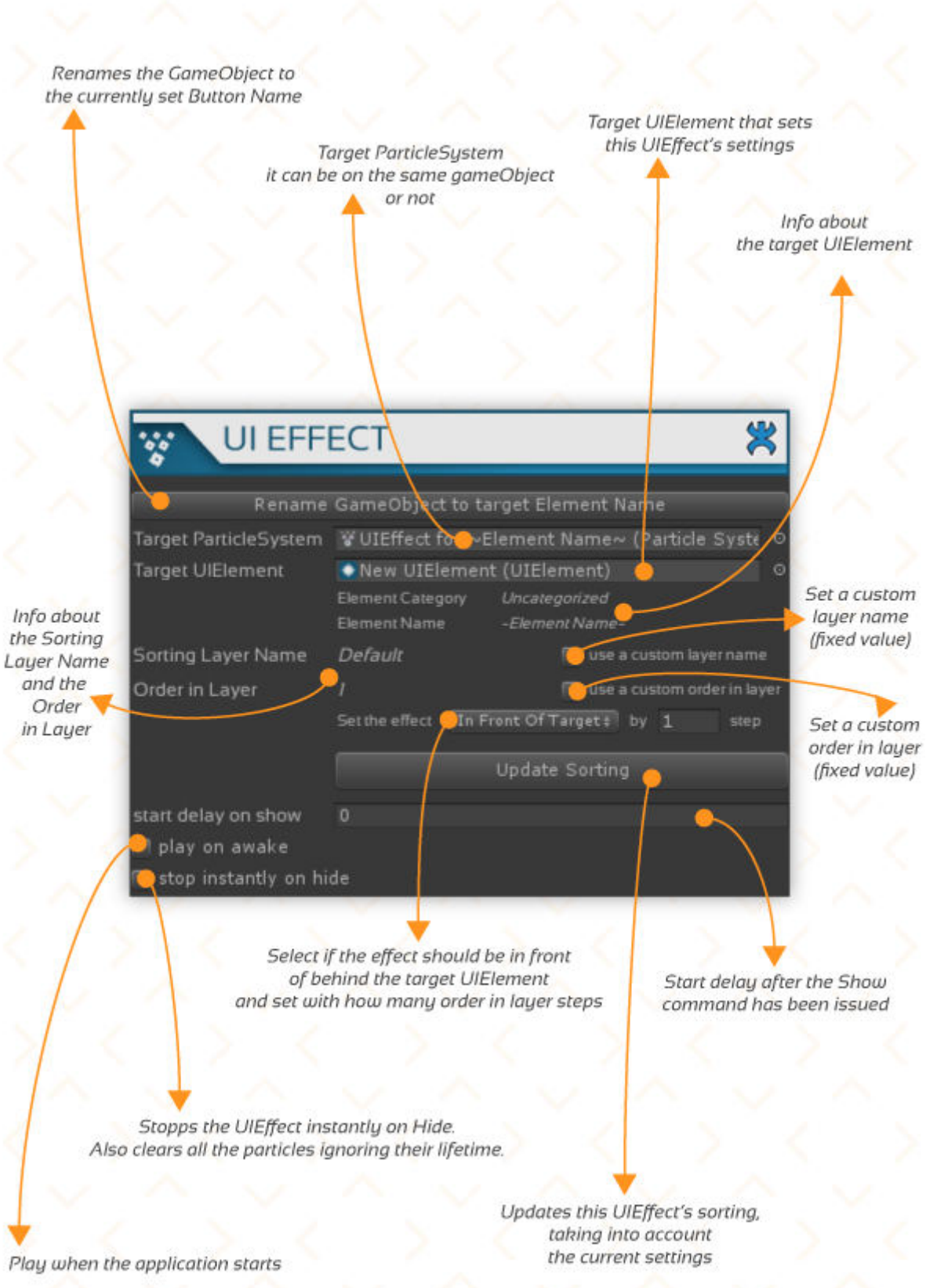
Wait for animation is enabled by default.. If true it will wait until the OnLongClick animation finised playing before sending the OnLongClick command (includes the show/hide and send game events actions). If false, it will send the OnLongClick command instantly without waiting for the animation to finish. Depends on the effect you are looking for.

Register interval is the 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.



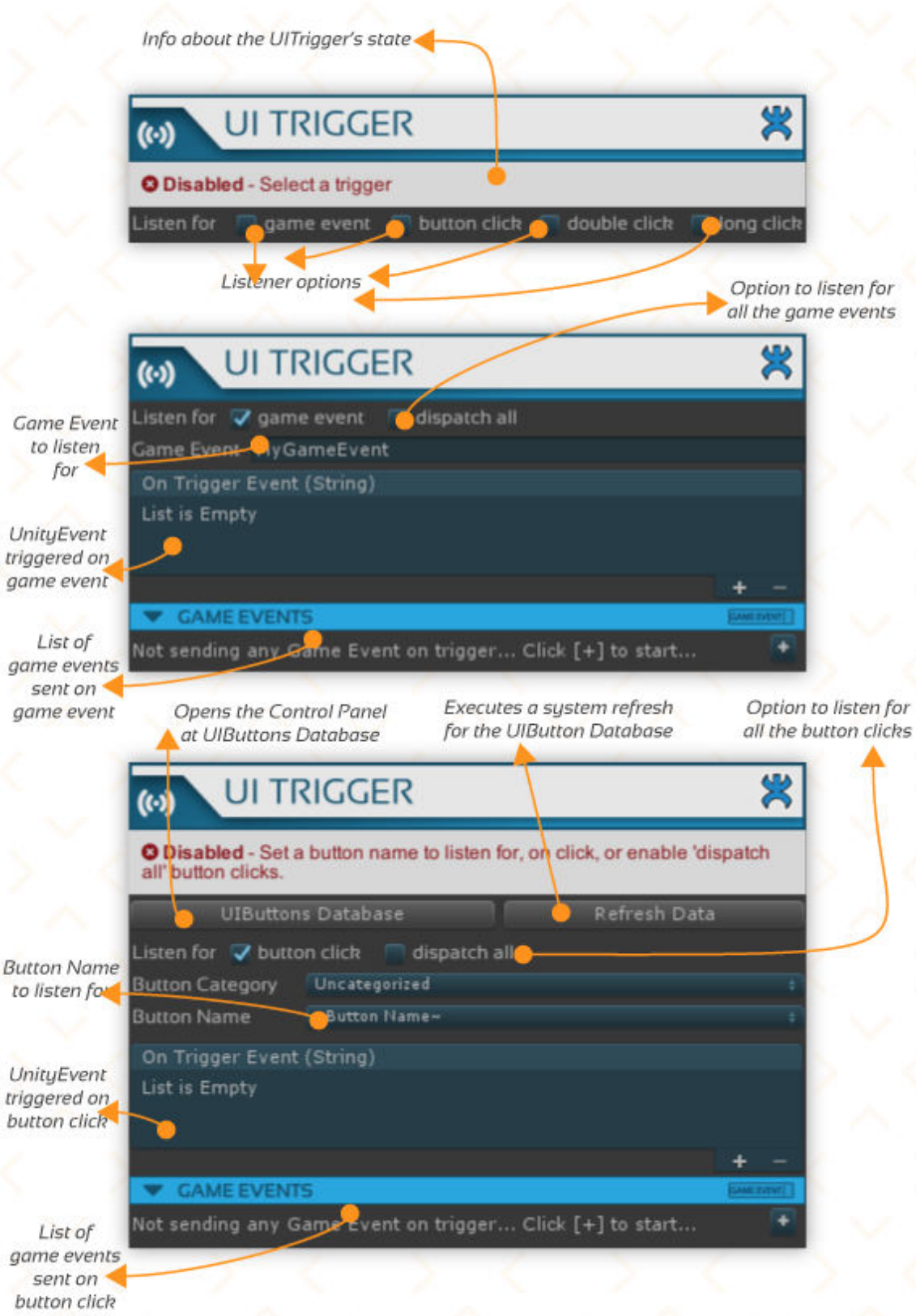
UI Effect

Helps you use native ParticleSystem components with the uGUI. It does the proper setup of the PartilcleSystem in order to get show in front or behinnd an UIElement.
It is also linked to an UIElement and it automatically starts emitting particles when that UIElement is shown and then stops the emision of particles when the UIElement is hidden.



UI Trigger

It registers itself to the UIManager and in return it gets notified whenever the a game event or button click (depending of what listener is activ) has been sent.
Upon being triggered, it invokes the UnityEvent and sends out any game events that have been set up.
It's quite a simple system and because of the implementation you can have as many as you want as you won't see any performance drops.
Dispatch all will listen and dispatch ALL game events or button clicks. *You should be aware that in order for it to work as intended you must call a function that has a string as a paramater and it is VERY IMPORTANT that you select, in the 'On Trigger Event (String)' section, from the right drop down list, a function that is located in the 'Dynamic string' section (on top) and not the 'Static parameters' section (on bottom).*



UI Notification

This is a special class that shows and configures modal windows that can be used in countless ways. From a simple tooltip, to an animated modal window with several buttons on it.

Notification container is a chld gameObject of the notification, with an UIElement component attached, that will serve as the main container of all the other notification elements.

Background overlay shoud be a fullscreen overlay / color tint. An Image with an apha.

Title is the text that will serve a the notification's title; a gameobject that should have either a Text or a TextMeshProUGUI component attached. For now Text Mesh Pro support has been removed. It will work in upgraded projects though.

Message is the text that will serve a the notification's message; a gameobject that should have either a Text or a TextMeshProUGUI component attached. For now Text Mesh Pro support has been removed. It will work in upgraded projects though.

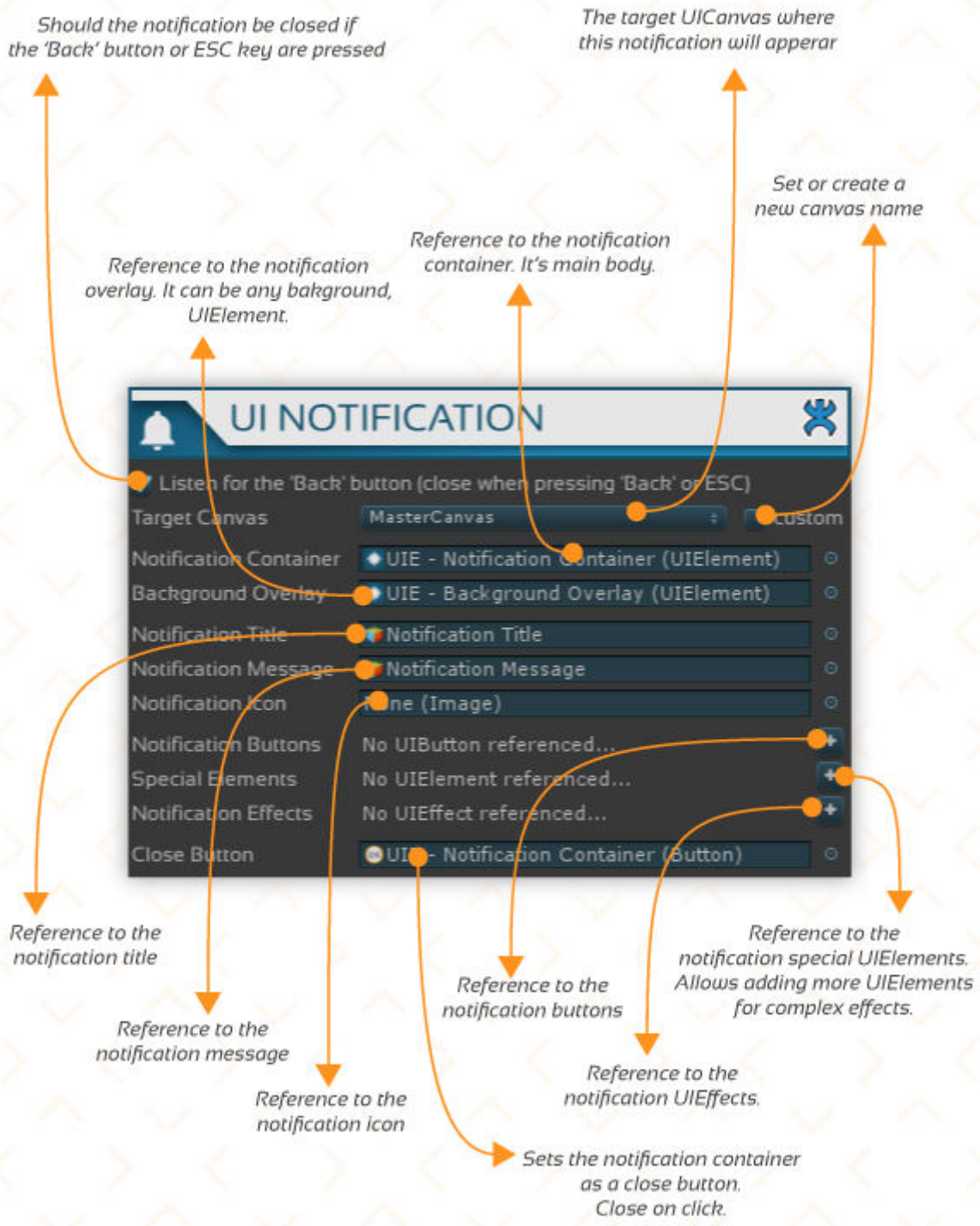
Should you want to show a custom **icon** every time you show the notification, reference an Image component here and pass in the icon Sprite when you call the ShowNotification method.

Buttons are the UIButtons you would like to be available for this notification. They can be turned on or off when you call the ShowNotification method.

Special elements are other UIElements (for extra panache). If you added stars or anything else with an UIElement attached, you need to link it here. This list of UIElements allows you to create fancy animations with a lot of objects.

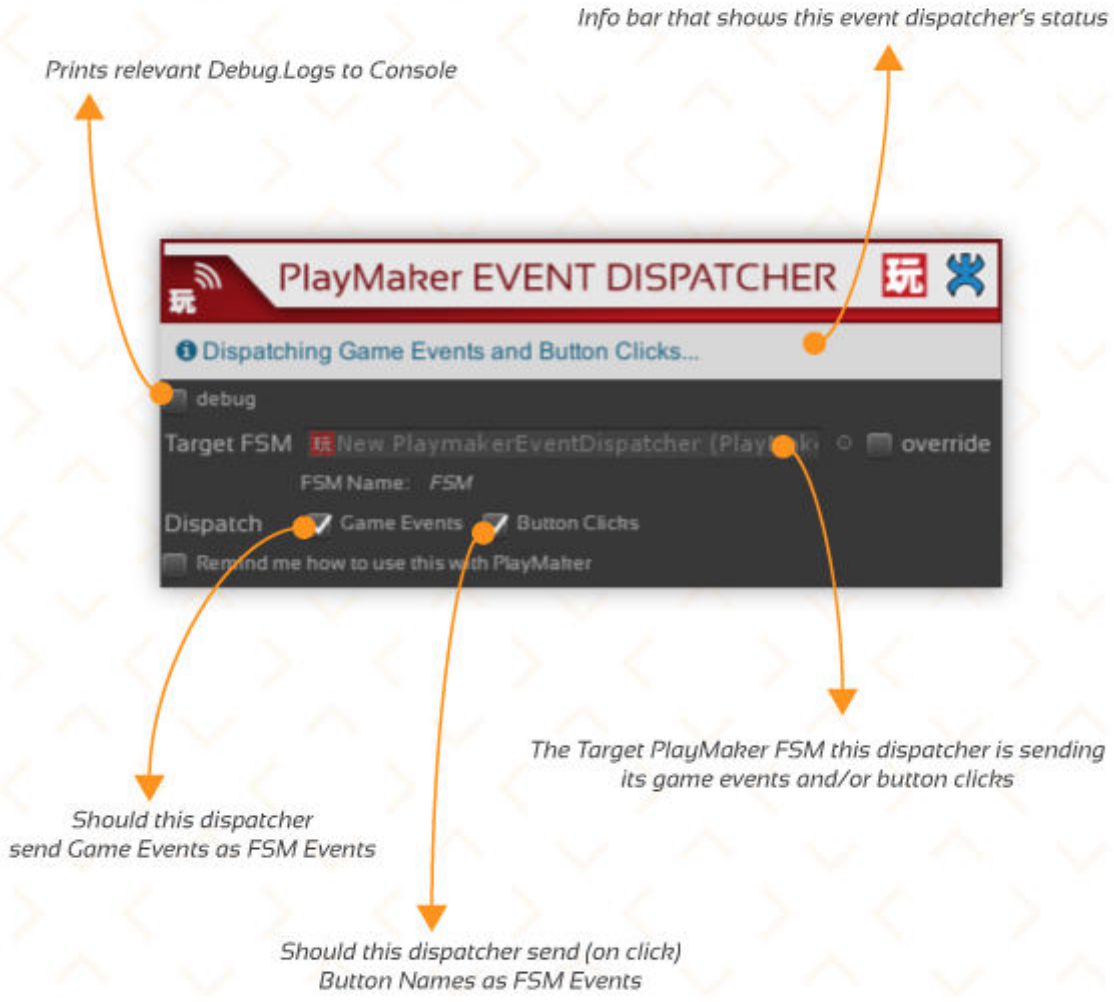
If you added any **effects** with UIEffect attached, you need to link them here so that they can work as intended.

Close button is a special button as it coveres the entire notification area. This allows the user to dismiss at once the notification, just by touching/clicking on it. (TIP: you can attach a Button component to the Overlay, link it here and have a full screen close button)


Should the notification be closed if the 'Back' button or ESC key are pressed
The target UICanvas where this notification will apperar
Set or create a new canvas name
Reference to the notification container. It's main body.
Reference to the notification overlay. It can be any bakground, UIElement.
UI NOTIFICATION
Listen for the 'Back' button (close when pressing 'Back' or ESC)
Target Canvas
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 UIButton referenced...
Special Elements
No UIElement referenced...
Notification Effects
No UIEffect referenced...
Close Button
UIE - Notification Container (Button)
Reference to the notification title
Reference to the notification message
Reference to the notification icon
Reference to the notification buttons
Reference to the notification special UIElements. Allows adding more UIElements for complex effects.
Reference to the notification UIEffects.
Sets the notification container as a close button. Close on click.

PlayMaker Event Dispatcher

For this dispatcher to work in Playmaker, you have to create FSM events named exactly as the Game Event commands or buttonNames that you want to listen for and react to, in the FSM. The event names are case sensitive.



UI Navigation

The system is turned on by default. It's enabled state can be toggled from the Control Panel.

The Navigation System keeps track of all the registered navigation actions, using the First-In Last-Out (FILO) pattern, in a stack. To register an action all you need to do is check the 'Add to Navigation' option on any UIButton that shows or hides elements.

Special Buttons Names

- **Back**
  - simulates the 'Back' button on android and it goes back to the previous menu activating the Navigation System
  - to set an UIButton as a 'Back' button you need to check the 'Is Back Button' option
- **TogglePause**
  - Toggles the timescale pausing or unpausing the game
  - When pausing it tweens the timescale from the current value to zero in 0.25 seconds
  - When unpausing it tweend the timescale from zero to the initial timescale in 0.25 seconds
- **ToggleSound**
  - Toggles the sound on/off while saving the value to PlayerPrefs
  - On enable sound it also triggers show 'SoundON' and hide 'SoundOFF' elementNames
  - On disable sound it also triggers show 'SoundOFF' and hide 'SoundON' elementNames
  - This is useful since you can have 2 buttons one over another and make them look like a toggle
- **ToggleMusic**
  - Toggles the music on/off while saving the value to PlayerPrefs
  - On enable music it also triggers show 'MusicON' and hide 'MusicOFF' elementNames
  - On disable sound it also triggers show 'MusicOFF' and hide 'MusicON' elementNames
  - This is useful since you can have 2 buttons one over another and make them look like a toggle
- **ApplicationQuit**
  - exits Play Mode (if in Unity Editor) or it executes Application.Quit

Special Game Events

- **DisableBackButton**
  - disables the 'Back' button functionality
- **EnableBackButton**
  - enables the 'Back' button functionality (the 'Back' button is enabled by default)
- **SoundCheck**
  - does a sound check and shows the proper state for the sound button toggle
- **MusicCheck**
  - Does a music check and shows the proper state for the music button toggle

Sound Settings

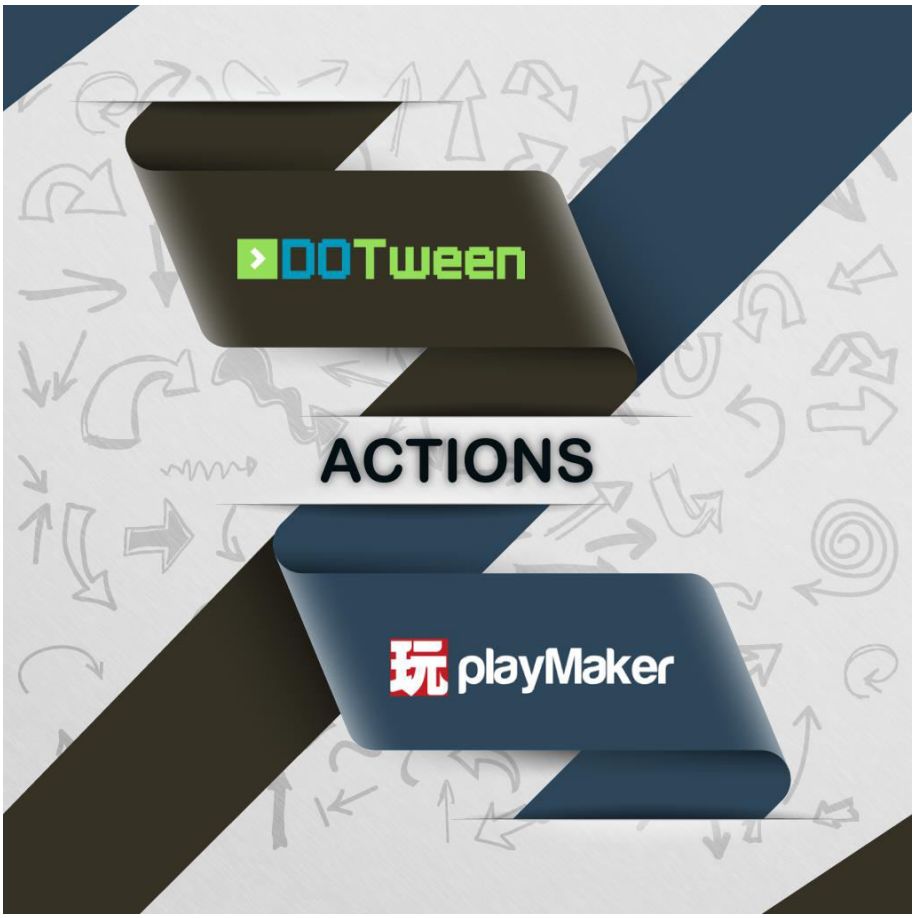
At runtime, on Start, UIManager does a SoundCheck and a MusicCheck loading the settings saved in PlayerPrefs.

You may have noticed that UISounds require a strings to trigger. That is because we are loading the sound files from Resources folder, thus we don't need any reference to them. This is useful since we will never get a NullReferenceException nor do we need to keep references to each sound. On the other hand, should you want to use references, just go ahead and reference your audio clips in the Control Pannel > UI Sounds database section.

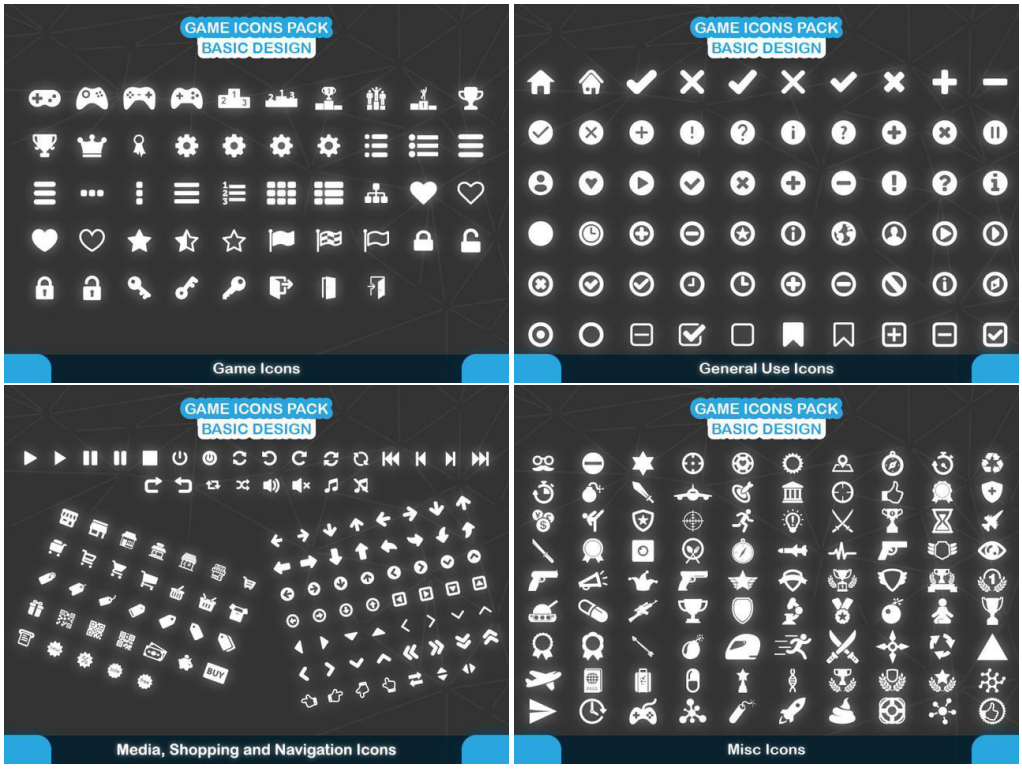
If you enabled the Master Audio support Soundy will try to play the sounds using Master Audio. You need to have the sounds set up in Master Audio in oder for the system to work.

Final Words

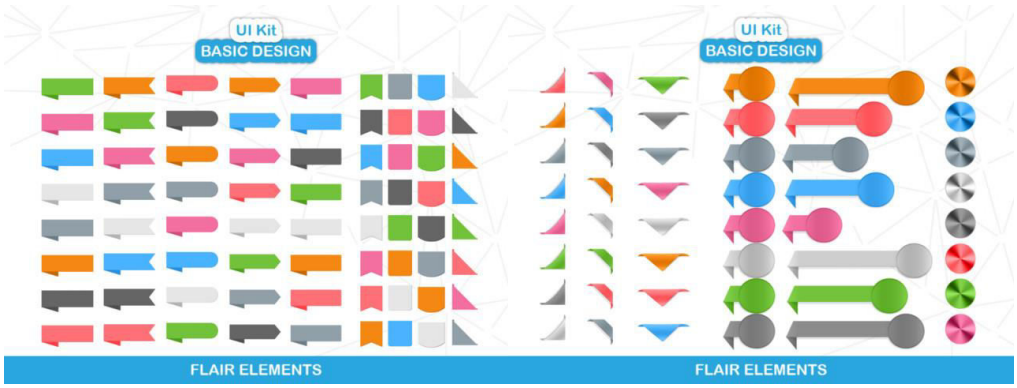
- Make sure you check out our other assets such as:
  - Playmaker Actions for DOTween by Doozy - http://u3d.as/kRs



  - Game Icons Pack - Basic Design - http://u3d.as/crV

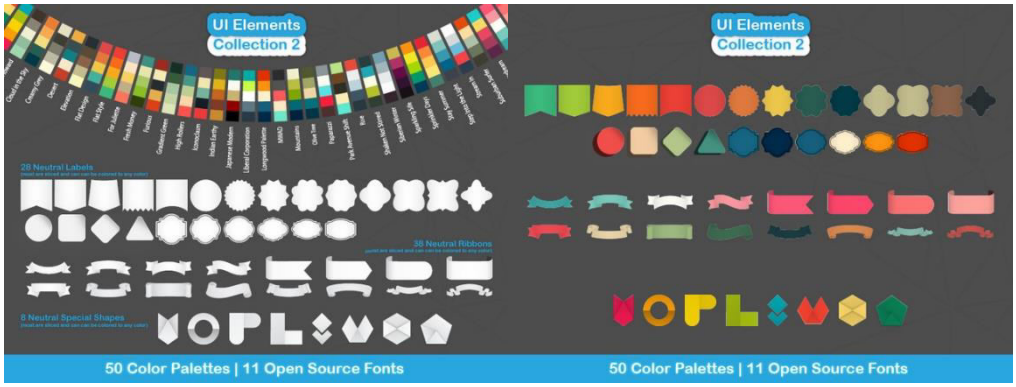


- UI Kit - Basic Design - http://u3d.as/fyv



- UI Elements - Collection 1 - http://u3d.as/g4y

- UI Elements - Collection 2 - http://u3d.as/ghU



- and others - https://goo.gl/kEADpX