DOOZY UI

POWERED BY

DOTWEEN

SUPPORTS

MASTER AUDIO

TEXT MESH PRO

ENERGY BAR TOOLKIT

PLAYMAKER

Table of Contents

What can DoozyUI do for you! 4
Quick Start 5
General Info 5
Menu → DoozyUI/Control Panel 5
UI Camera 5
UI Container 5
UI Manager 5
Video Tutorials 6
About the DoozyUI components 6
3rd Party Plugins 6
The Navigation System 6
The Orientation Manager 6
Control Panel 7
UI Manager 8
Code Examples 9
UI Element 11
Code Examples 14
UI Button 15
Code Examples 17
UI Trigger 19
UI Effect 20
UI Notification 21
Code Examples 22
SceneLoader 24
Code Examples 25
PlaymakerEventDispatcher 26
PlayMaker Actions 27
Send Game Event Message 27
Send UI Button Message 27
Show Hide Elements 27
Show UI Notification 27
Application Quit or Close 28
Disable Back Button 28
Enable Back Button 28
Toggle Sound 28
Toggle Music 28
Toggle Pause 28

Navigation System 29
General Info 29
Special Buttons 29
Special Game Events 29
Sound Settings 30
General Info 30
Code Examples 30
Final Words 31

This plugin was created to help speed up the design, implementation and management any type of User Interface created with Unity. It ads functionality to all the native Unity UI components and it integrates seamlesly within the Unity environment.

We are are always open to hearing new ideas for improvements or suggestions and of any problems that you might encounter while using DoozyUI. You can email us any time at doozy.entertainment@gmail.com and we will respond shortly.

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).
- ✓ Seamless integration of MasterAudio, TextMeshPro, EnergyBarToolkit and PlayMaker with DoozyUI.
- ✓ Give you a better understanding of how the native Unity UI components work by watching the tutorial videos.
- ✓ Has a very low impact on performance. Is mobile friendly! ☺

Quick Start

1. Import DOTween or DOTween Pro before importing the DoozyUI package.
2. Setup DOTween (got to Tools -> DOTween Utility Panel -> Setup DOTween...)
3. Import DoozyUI
4. From the menu DoozyUI -> ControlPanel -> Add DoozyUI to the current scene
5. Done! ☺

Note 1: You should have a sorting layer named "UI" set up in the project. If you don't, create it in Tags&Layers.

Note 2: If you find Canvas components with "Sorting Layer <unknown layer>", even though you have created a sorting layer named "UI", you can select the UI Container gameObject (under the DoozyUI gameObject) and click on [Update Canvases] and [Update Renderers] buttons.

[Quick Start Video]

General Info

Menu → DoozyUI/Control Panel

- Activates/Deactivates support for MasterAudio, TextMeshPro, EnergyBarToolkit and Playmaker *(deactivated by default)*
- Activates/Deactivates the UI Navigation System *(active by default)*
- Manages the databases:
  - ElementNames: all the element names shown in the UIElement component
  - ElementSound: all the sounds an UIElement can play
  - ButtonNames: all the button names shown in the UIButton component
  - ButtonSounds: all the onClickSounds shown in the UIButton component
- Upgrades the current scene from DoozyUI 1.2d to DoozyUI 2.0 by creating the databases and populating them with proper records. Also it adds missing components required by the new system.
  - *The upgrade will not work for animation presets.*

UI Camera

- Set to orthographic mode
- Size 5.4 (means that the fixed height of the sceen in pixels should be 1080px, 1 unity unit = 100 pixels)
- This camera comes preconfigured to see only the UI layer
- Your other Cameras should not see the UI layer

UI Container

- This holds all the UI and any gameObject with a RectTransform should be under this gameObject.
- Canvas set to Screen Space – Camera linked to the [UI Camera]
- Canvas Scaler set to Scale With Screen Size
  - Reference Resolution set to 1920x1080px. Height is set to always be fixed at 1080px.
  - Reference Pixels Per Unit 100 (1 unity unit = 100 pixels)
- Has a component **UpdateSortingLayerNames** that helps fix the <unknown layer> issue (you need to have a SortingLayer named 'UI')

UI Manager

- The core of all the UI System
- Needs to be under the UI Container to get the screen size and adjust all the animation calculations
- Has all the methods that control the UI. All the methods are static
- Holds registries for all the DoozyUI components for efficient trigger routing
- Toggles support for MasterAudio, TextMeshPro, EnergyBarToolkit and PlayMaker *(default: disabled)*

- Toggles the Navigation System *(default: enabled)*

Video Tutorials

Doozy YouTube Channel Link - where you will find all the video tutorials; new tutorials are added on a weekly basis

About the DoozyUI components

- Quick Start - how to install and start using DoozyUI
- The Control Panel – general informations and how to use it
- UI Manager – general informations and how to use it
- UI Button – general informations, how to use it and usage examples
- UI Element – general informations, how to use it and usage examples
- UI Trigger – general informations, how to use it and usage examples
- UI Effect – general informations, how to use it and usage examples
- Scene Loader – general informations, how to use it and usage examples
- Playmaker Event Dispatcher – general informations, how to use it and usage examples

3rd Party Plugins

- How to enable support for MasterAudio
- How to enable support for TextMeshPro
- How to enable support for EnergyBarToolkit
- How to enable support for Playmaker

The Navigation System

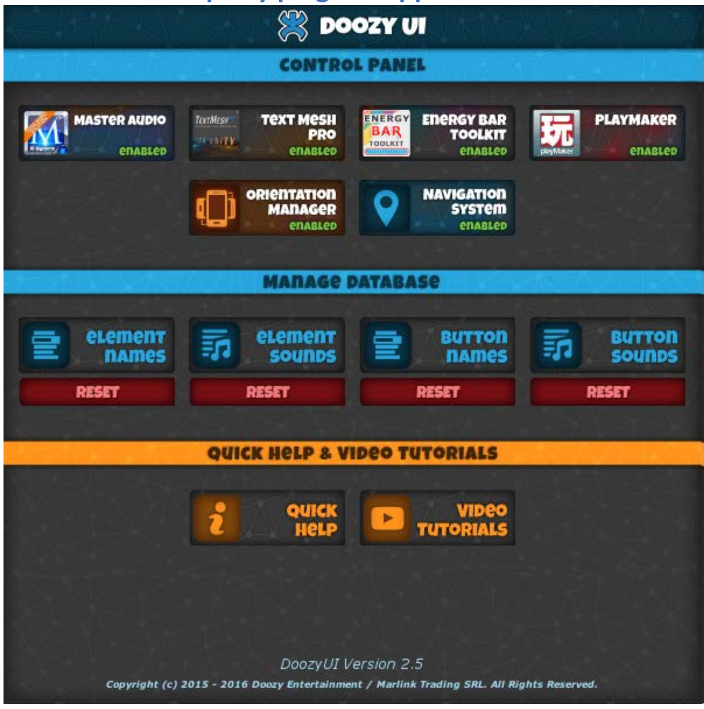
- How to disable the Navigation System

The Orientation Manager

- How to use the Orientation Manager

Control Panel

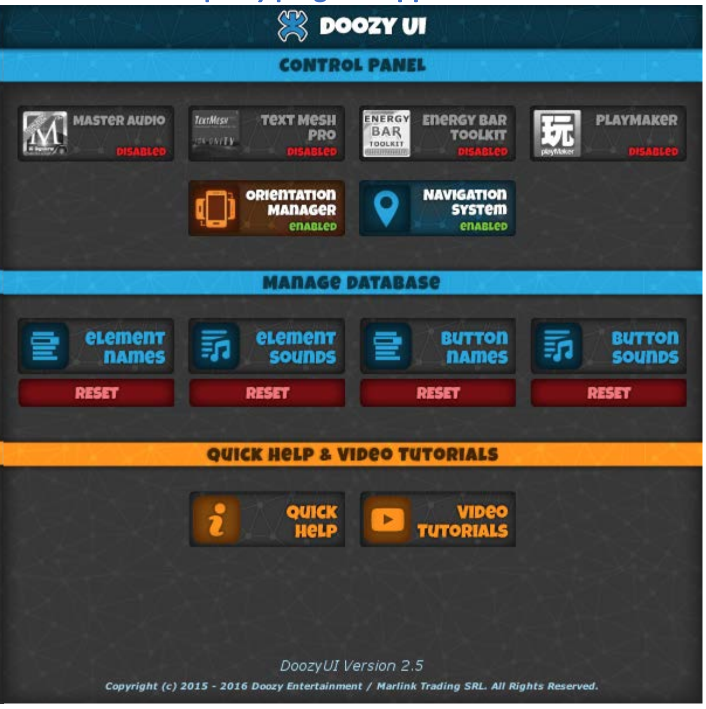
with 3rd party plugins support *enabled*



DoozyUI Version 2.5

Copyright (c) 2015 - 2016 Doozy Entertainment / Marlink Trading SRL. All Rights Reserved.

with 3rd party plugins support *disabled*



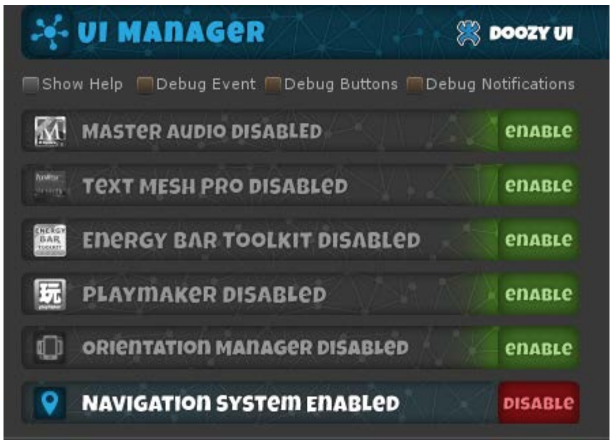
DoozyUI Version 2.5

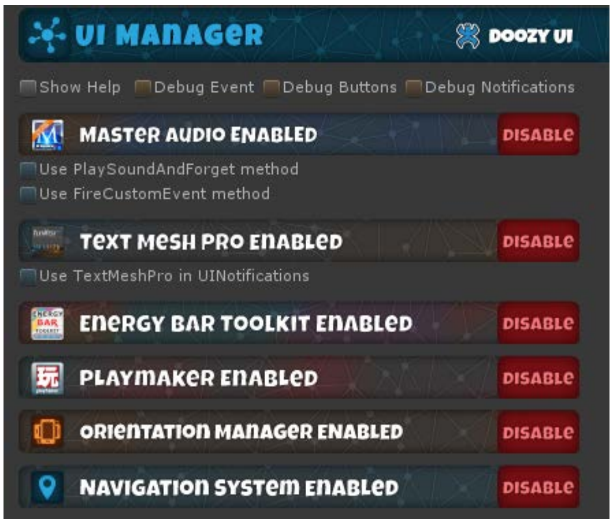
Copyright (c) 2015 - 2016 Doozy Entertainment / Marlink Trading SRL. All Rights Reserved.

The Control Panel will help you with the following things:

- Toggle 3rd party plugins support (for MasterAudio, TextMeshPro, EnergyBarToolkit, Playmaker)
- Toggle the Navigation System (enabled by default)
- Toggle the Orientation Manager (disabled by default)
- Manage the databases for
    - Element Names
    - Element Sounds
    - Button Names
    - Button Sounds
- Quick Help – has a link to this updated documentation and it will contain a FAQ section in future updates
- Video Tutorials – has a link to our YouTube channel (with a lot of tutorials) and direct links to the most relevant tutorials
- Upgrade Scene – this is an upgrade helper for users of DoozyUI 1.2d; it will populate all the databases with the relevant informations, collected from the current active scene.

UI Manager


UI MANAGER
DOOZY UI
Show Help · Debug Event · Debug Buttons · Debug Notifications
MASTER AUDIO DISABLED — enable
TEXT MESH PRO DISABLED — enable
ENERGY BAR TOOLKIT DISABLED — enable
PLAYMAKER DISABLED — enable
ORIENTATION MANAGER DISABLED — enable
NAVIGATION SYSTEM ENABLED — DISABLE


UI MANAGER
DOOZY UI
Show Help · Debug Event · Debug Buttons · Debug Notifications
MASTER AUDIO ENABLED — DISABLE
Use PlaySoundAndForget method
Use FireCustomEvent method
TEXT MESH PRO ENABLED — DISABLE
Use TextMeshPro in UINotifications
ENERGY BAR TOOLKIT ENABLED — DISABLE
PLAYMAKER ENABLED — DISABLE
ORIENTATION MANAGER ENABLED — DISABLE
NAVIGATION SYSTEM ENABLED — DISABLE

Show Help: shows inspector tooltips on each setting
Debug Event: prints to console all the Game Events
Debug Buttons: prints to console all the Button Clicks
Debug Notifications: prints to console all the shown notifications

MasterAudio:
- If enabled, it adds 'dUI_MasterAudio' to PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If disabled, it removes 'dUI_MasterAudio' from PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If use PlaySoundAndForget method is enabled, it will trigger the method with the same name from MasterAudio when playing a sound.
- If use FireCustomEvent method is enabled, it will trigger the method with the same name from MasterAudio when playing a sound.
- Use either PlaySoundAndForget or FireCustomEvent, but do not enable both of them at once.

TextMeshPro:
- If enabled, it adds 'dUI_ TextMeshPro' to PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If disabled, it removes 'dUI_TextMeshPro' from PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If enabled it will also consider that you are using TextMeshProUGUI components in your UI and it will not let you do scale animations on the z axis to avoid a TextMeshPro bug.
- If use TextMeshPro in UINotification is enabled, it will consider that the title, message and buttonTexts on the UINotification are TextMeshProUGUI, not Text.

EnergyBarToolkit:
- If enabled, it adds 'dUI_ EnergyBarToolkit' to PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If disabled, it removes 'dUI_EnergyBarToolkit' from PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If enabled, it will allow you to link loading bars to the SceneLoader.

PlayMaker:
- If enabled, it adds 'dUI_ PlayMaker’ to PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If disabled, it removes 'dUI_PlayMaker’ from PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If enabled, it will activate the PlaymakerEventDispatcher and all the DoozyUI Playmaker Actions.

OrientationManager:
- If enabled, it adds 'dUI_ UseOrientationManager’ to PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If disabled, it removes 'dUI_ UseOrientationManager’ from PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.

NavigationSystem:
- If enabled, it adds 'dUI_ NavigationDisabled’ to PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.
- If disabled, it removes 'dUI_ NavigationDisabled from PlayerSettings --> Scripting Define Symbols. It takes a few seconds to update.

- Enable it if you want to handle the navigation yourself (the back button and navigation options). This is useful if you are using a FSM system like PlayMaker in order to have a visual control over the UI. This switch will also modify the UIButton component and disable several other options.

Code Examples

*remember to add: ***using DoozyUI;***

Send a Game Event
```
string gameEvent = "gameEvent";
UIManager.SendGameEvent(gameEvent);
```

Send several Game Events at once
```
List<string> gameEvents = new List<string>() { "gameEvent_1" , "gameEvent_2", "gameEvent_3" };
UIManager.SendGameEvents(gameEvents);
```

Simulates a button click
```
UIManager.SendButtonClick(string _buttonName, bool _addToNavigationHistory = false, bool
_backButton = false, GameObject _gameObject = null, List<string> _showElements = null,
List<string> _hideElements = null, List<string> _gameEvents = null)
```

Check if the sound is on
```
UIManager.isSoundOn
```

Check if the music is on
```
UIManager.isMusicOn
```

Get the UICamera that comes with DoozyUI (returns a Camera)
```
UIManager.GetUICamera
```

Get Unity's default EventSystem that is active in the current scene (returns an EventSystem)
```
UIManager.GetEventSystem
```

Get the UIContainer Transfrom reference (this is the main Canvas of DoozyUI)
```
UIManager.GetUIContainer
```

Returns a List<string> or all the UIElement ElementNames that are visible on the screen (isVisible = true)
```
UIManager.GetVisibleUIElementElementNames()
```

Check the current orientation (if you have Orientation Manager enabled)
```
UIManager.currentOrientation
```

Checks the soundState in the PlayerPrefs and updates it accordingly
```
UIManager.SoundCheck()
```

Checks the musicState in the PlayerPrefs and updates it accordingly
```
UIManager.MusicCheck()
```

Toggles the soundState and saves it to the PlayerPrefs. Then sends a GameEvent 'UpdateSoundSettings'
```
UIManager.ToggleSound()
```

Toggles the musicState and saves it to the PlayerPrefs. Then sends a GameEvent 'UpdateSoundSettings'

```
UIManager.ToggleMusic()
```

Pauses or unpauses the application

```
UIManager.TogglePause()
```

Exits play mode (if in editor) or quits the application if in build mode

```
UIManager.ApplicationQuit()
```

The 'back' button was pressed (or escape key) – Simulates the Back button

```
UIManager.BackButtonEvent()
```

The 'back' button was pressed (or escape key) – Simulates the Back button

```
UIManager.BackButtonEvent()
```

Disables the 'Back' button functionality

```
UIManager.DisableBackButton()
```

Enables the 'Back' button functionality

```
UIManager.EnableBackButton()
```

Enables the 'Back' button functionality by resetting the additive bool to zero. backButtonDisableLevel = 0. Use this ONLY in special cases if something wrong happened and the back button is stuck in disabled mode.

```
UIManager.EnableBackButtonByForce()
```

Adds a navigation item to the Navigation History stack

```
UIManager. AddItemToNavigationHistory(Navigation navItem)
```

Removes the last item from the Navigation History stack

```
UIManager. RemoveLastItemFromNavigationHistory()
```

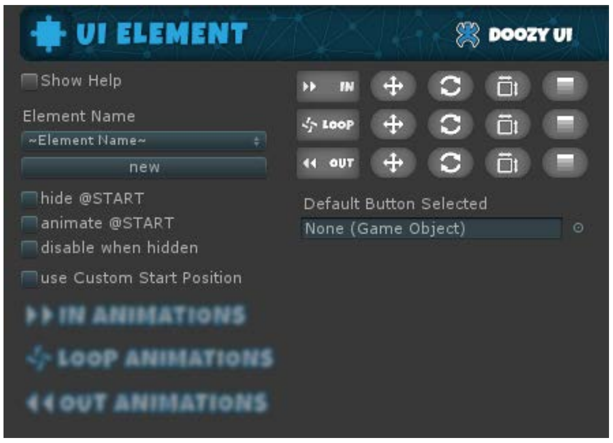
Returns the last item in the Navigation History stack. It removes the item from the stack by default.

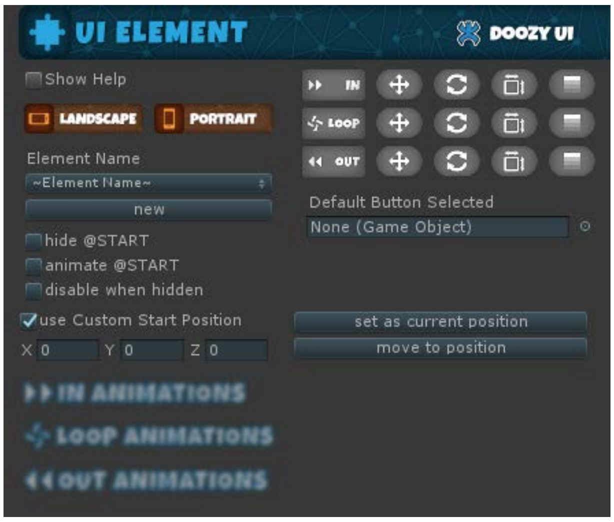
```
UIManager. GetLastItemFromNavigationHistory(bool removeFromStack = true)
```

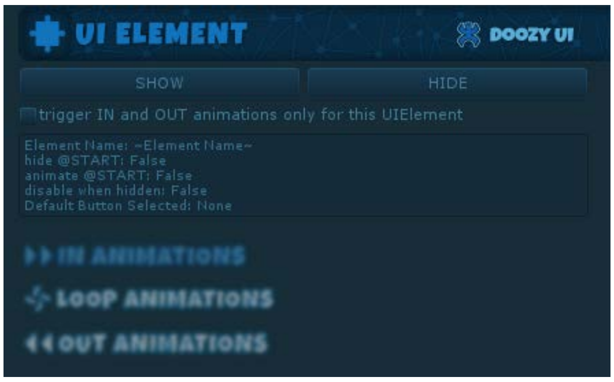
Cleares the Navigation History stack

```
UIManager. ClearNavigationHistory()
```

UI Element







```
Element Name: ~Element Name~
hide @START: False
animate @START: False
disable when hidden: False
Default Button Selected: None
```

Show Help: shows inspector tooltips on each setting

LANDSCAPE / PORTRAIT: enable this is you want this UIElement to be visible in landscape/portrait orientation. These buttons are enabled by default. They are visible only if the Orientation Manager is enabled.

Element Name: is used to trigger the show/hide animations for this gameObject.

- **new:** create a new element name in the ElementNames database
- **rename:** rename the selected element name in the ElementNames database
- **delete:** removes the selected element name from the ElementNames database and sets this UIElement's element name to the default value.

The right side mini menu
IN, LOOP, OUT – blue buttons toggle the visibility of the animation zones with the same name
MOVE (green), ROTATE (orange), SCALE (red), FADE (purple) – toggle the enabled/disabled states of the animations.

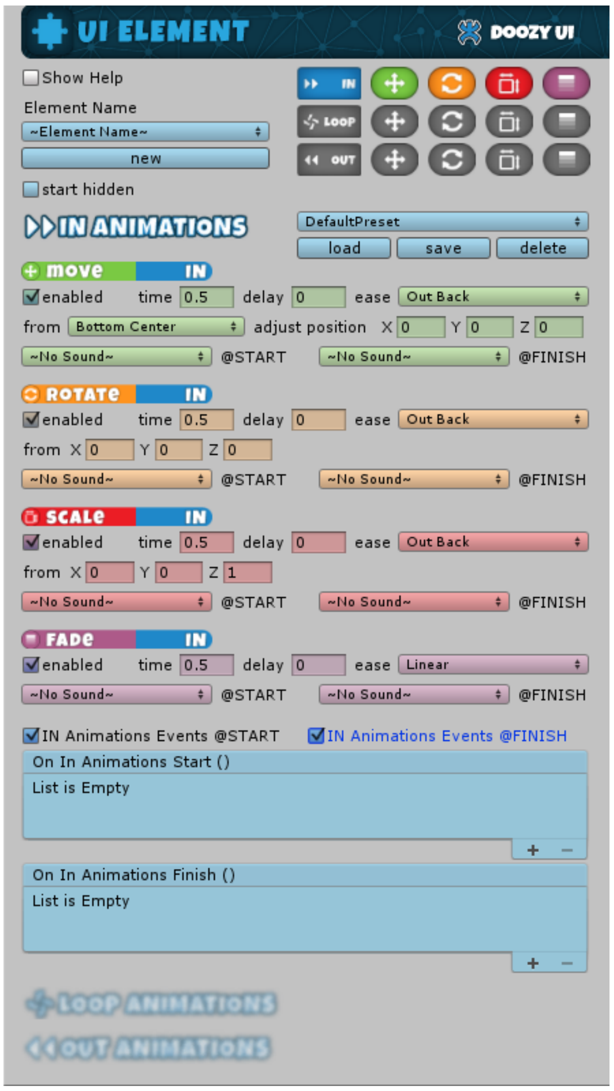
hide @Start: if enabled, it will auto hide this element when the app/game starts (an IN and an OUT animation should be setup beforehand) - this was previously known as 'start hidden'
animate @Start: if enabled, it will trigger automatically the IN animations OnAwake (an IN and an OUT animation should be setup beforehand)
disable when hidden: if enabled, it will disable - SetActive(false) - the UIElement when it is hidden (or set to Hide). This will help lower the draw calls for the UI.

Default Button Selected: if a button is referenced, it will get selected automatically when the UIElement is shown. This feature has been introduced to facilitate the control of the UI through controllers and keyboard

use Custom Start Position: allows you to move the UIElement (window) anywhere in the scene. On Awake, it will automatically move to this position in order for the IN and OUT animations to work.
set as current position: sets the Custom Start Position as the UIElement's current anchoredPosition3D
move to position: moves the UIElement to the Custom Start Position using the anchoredPosition3D

NOTE: when you enter PLAY MODE the UIElement's Inspector changes and new options are now available to you.

Options visible only in PLAY MODE
SHOW: this will trigger the Show method for this elementName; because it works as a global command, it will affect all the UIElements with the same elementName.
HIDE: this will trigger the Hide method for this elementName; because it works as a global command, it will affect all the UIElements with the same elementName.
trigger IN and OUT animations only for this UIElement: if, for some reason, you want to Show or Hide ONLY THIS UIElement, and not trigger all the others (with the same elementName), you should set this toggle to TRUE. (default: FALSE)



Show Help
Element Name ~Element Name~
new
start hidden
DefaultPreset — load — save — delete

move IN: enabled ✓ time 0.5 delay 0 ease Out Back; from Bottom Center; adjust position X 0 Y 0 Z 0; ~No Sound~ @START ~No Sound~ @FINISH

ROTATE IN: enabled ✓ time 0.5 delay 0 ease Out Back; from X 0 Y 0 Z 0; ~No Sound~ @START ~No Sound~ @FINISH

SCALE IN: enabled ✓ time 0.5 delay 0 ease Out Back; from X 0 Y 0 Z 1; ~No Sound~ @START ~No Sound~ @FINISH

FADE IN: enabled ✓ time 0.5 delay 0 ease Linear; ~No Sound~ @START ~No Sound~ @FINISH

✓ IN Animations Events @START ✓ IN Animations Events @FINISH

On In Animations Start ()
List is Empty

On In Animations Finish ()
List is Empty

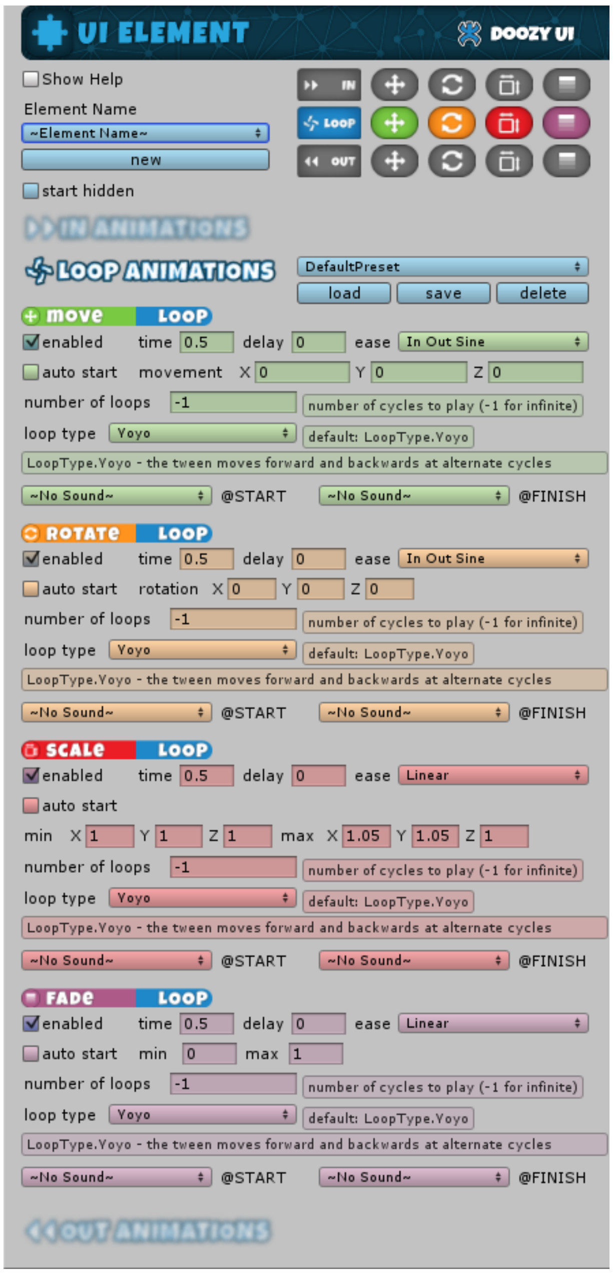
IN ANIMATIONS: toggles the visibility of the IN Animations zone.
Preset Zone
selector: select the preset animation settings you want to load
load: load the selected preset
save: save a new preset with the current animation settings (creates a .xml file)
delete: delete the selected preset (deletes the .xml file)
**the IN animations presets can be found in .xml format in DoozyUI/Presets/UIAnimations/IN*
*** all the animations can be reused in other projects by copying the .xml files*

Settings for all animation types
enabled: toggles the enabled/disabled state of each type of animation
time: amount (seconds) that the animation will take to finish
delay: amount (seconds) that the animation will wait before starting
ease: the rate of change of the animation over time (for more details about see: http://easings.net)
@START: the sound filename/event name/sound name (depending if MasterAudio is enabled or not) that will play when the animation starts
@FINISH: the sound filename/event name/sound name (depending if MasterAudio is enabled or not) that will play when the animation finished
IN Animation Events @START: the methods that you want triggered at the start of the animation (uses native UnityEvent)
IN Animation Events @FINISH: the methods that you want triggered when the animation finished (uses native UnityEvent)

Special settings for each animation type
Move IN
from: the direction that the gameObject enters the screen view from
adjust position: used in special cases to offset the object's position from the center
position from: custom option, activated when **from: 'Local Position'** is selected.

ROTATE IN
from: the angle (in degrees) that the gameObject rotate from (the z axis is usually used)

SCALE IN
from: the scale factor at the start for the animation (default: 0); 1 is 100%



Show Help
Element Name: ~Element Name~
new
start hidden

DefaultPreset
load | save | delete

move LOOP
enabled: ✓ time 0.5 delay 0 ease In Out Sine
auto start movement X 0 Y 0 Z 0
number of loops -1 (number of cycles to play (-1 for infinite))
loop type Yoyo (default: LoopType.Yoyo)
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles
~No Sound~ @START ~No Sound~ @FINISH

ROTATE LOOP
enabled: ✓ time 0.5 delay 0 ease In Out Sine
auto start rotation X 0 Y 0 Z 0
number of loops -1 (number of cycles to play (-1 for infinite))
loop type Yoyo (default: LoopType.Yoyo)
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles
~No Sound~ @START ~No Sound~ @FINISH

SCALE LOOP
enabled: ✓ time 0.5 delay 0 ease Linear
auto start
min X 1 Y 1 Z 1 max X 1.05 Y 1.05 Z 1
number of loops -1 (number of cycles to play (-1 for infinite))
loop type Yoyo (default: LoopType.Yoyo)
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles
~No Sound~ @START ~No Sound~ @FINISH

FADE LOOP
enabled: ✓ time 0.5 delay 0 ease Linear
auto start min 0 max 1
number of loops -1 (number of cycles to play (-1 for infinite))
loop type Yoyo (default: LoopType.Yoyo)
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles
~No Sound~ @START ~No Sound~ @FINISH

LOOP ANIMATIONS: toggles the visibility of the LOOP Animations zone.
Preset Zone
selector: select the preset animation settings you want to load
load: load the selected preset
save: save a new preset with the current animation settings (creates a .xml file)
delete: delete the selected preset (deletes the .xml file)
**the LOOP animations presets can be found in .xml format in DoozyUI/Presets/UIAnimations/LOOP*
*** all the animations can be reused in other projects by copying the .xml files*

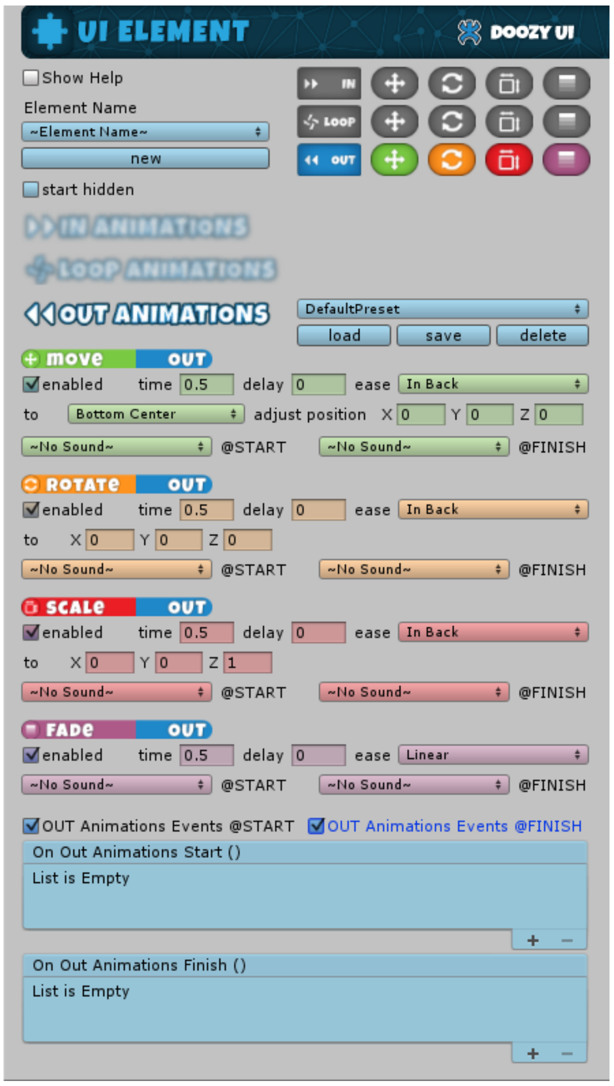
Settings for all animation types
enabled: toggles the enabled/disabled state of each type of animation
auto start: if there are no IN and OUT animations set up, you should enable auto start in order for the loop animation to work
time: amount (seconds) that the animation will take to finish
delay: amount (seconds) that the animation will wait before starting
ease: the rate of change of the animation over time (for more details about see: http://easings.net)
number of loops: number of cycle to play (-1 for infinite loops)
loop type: sets the looping options (Restart, Yoyo, Incremental) for the tween
LoopType.Restart: *When a loop ends it will restart from the beginning.*
LoopType.Yoyo: *When a loop ends it will play backwards until it completes another loop, then forward again, then backwards again, and so on and on and on.*
LoopType.Incremental: *Each time a loop ends the difference between its endValue and its startValue will be added to the endValue, thus creating tweens that increase their values with each loop cycle.*
@START: the sound filename/event name/sound name (depending if MasterAudio is enabled or not) that will play when the animation starts
@FINISH: the sound filename/event name/sound name (depending if MasterAudio is enabled or not) that will play when the animation finished

Special settings for each animation type
Move LOOP
movement: the distance to move on each axis (it is calculated startAnchoredPosition-movement for min and startAnchoredPosition+movement for max)

ROTATE LOOP
rotation: the amount (in degrees) that you want the object to rotate in time (back an forth if LoopType.Yoyo, or forward/backward if LoopType.Incremental).

SCALE LOOP
min: the scale factor at the start for the animation (default: 1); 1 is 100%
max: the scale factor when the animation finished (default: 1.05)

FADE LOOP
min: the alpha value at the start of the animation (default: 0)
max: the alpha value when the animation finished (default: 1)



Show Help
Element Name ~Element Name~
start hidden
DefaultPreset

move OUT
enabled time 0.5 delay 0 ease In Back
to Bottom Center adjust position X 0 Y 0 Z 0
~No Sound~ @START ~No Sound~ @FINISH

ROTATE OUT
enabled time 0.5 delay 0 ease In Back
to X 0 Y 0 Z 0
~No Sound~ @START ~No Sound~ @FINISH

SCALE OUT
enabled time 0.5 delay 0 ease In Back
to X 0 Y 0 Z 1
~No Sound~ @START ~No Sound~ @FINISH

FADE OUT
enabled time 0.5 delay 0 ease Linear
~No Sound~ @START ~No Sound~ @FINISH

OUT Animations Events @START OUT Animations Events @FINISH
On Out Animations Start ()
List is Empty
On Out Animations Finish ()
List is Empty

OUT ANIMATIONS: toggles the visibility of the OUT Animations zone.
Preset Zone
selector: select the preset animation settings you want to load
load: load the selected preset
save: save a new preset with the current animation settings (creates a .xml file)
delete: delete the selected preset (deletes the .xml file)
**the OUT animations presets can be found in .xml format in DoozyUI/Presets/UIAnimations/OUT*
*** all the animations can be reused in other projects by copying the .xml files*

Settings for all animation types
enabled: toggles the enabled/disabled state of each type of animation
time: amount (seconds) that the animation will take to finish
delay: amount (seconds) that the animation will wait before starting
ease: the rate of change of the animation over time (for more details about see: http://easings.net)
@START: the sound filename/event name/sound name (depending if MasterAudio is enabled or not) that will play when the animation starts
@FINISH: the sound filename/event name/sound name (depending if MasterAudio is enabled or not) that will play when the animation finished
OUT Animation Events @START: the methods that you want triggered at the start of the animation (uses native UnityEvent)
OUT Animation Events @FINISH: the methods that you want triggered when the animation finished (uses native UnityEvent)

Special settings for each animation type
MOVE OUT
to: the direction that the gameObject exits the screen view to
adjust position: used in special cases to offset the object's position from the center
position to: custom option, activated when **from: 'Local Position'** is selected.

ROTATE OUT
to: the angle (in degrees) that the gameObject to rotates to (the z axis is usually used)

SCALE OUT
to: the scale factor when the animation finished (default: 0); 1 is 100%

Code Examples

*remember to add: ***using DoozyUI;***

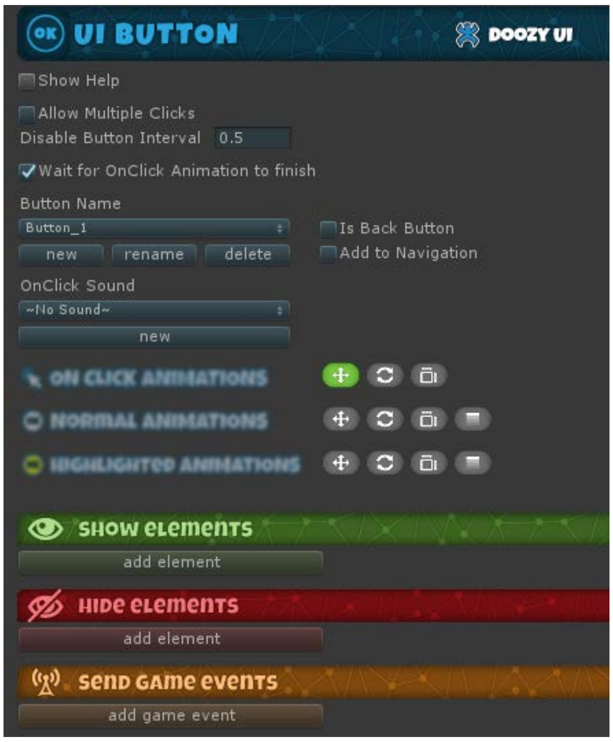
Show an UIElement from any script
```
string elementName = "elementName";
UIManager.ShowUiElement(elementName);
```

Show an UIElement instantly (in zero seconds) from any script
```
string elementName = "elementName";
UIManager.ShowUiElement(elementName, true);
```

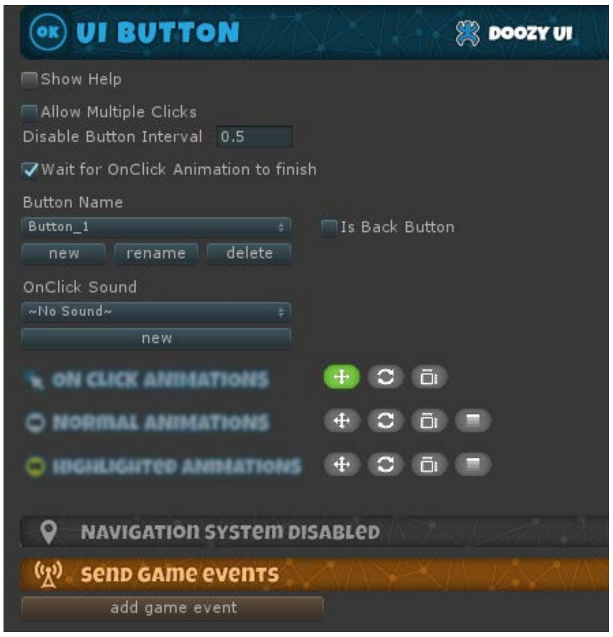
Hide an UIElement from any script
```
string elementName = "elementName";
UIManager.HideUiElement(elementName);
```

Hide an UIElement instantly (in zero seconds) from any script
```
string elementName = "elementName";
UIManager.HideUiElement(elementName, true);
```

UI Button



☐ Show Help
☐ Allow Multiple Clicks
Disable Button Interval 0.5
☑ Wait for OnClick Animation to finish
Button Name
Button_1
☐ Is Back Button
☐ Add to Navigation
OnClick Sound
~No Sound~



☐ Show Help
☐ Allow Multiple Clicks
Disable Button Interval 0.5
☑ Wait for OnClick Animation to finish
Button Name
Button_1
☐ Is Back Button
OnClick Sound
~No Sound~

Show Help: shows inspector tooltips on each setting

Allow Multiple Clicks: allows the user to press the button multiple times without restrictions. If you want to disable the button after each click for a set interval then disable the allow multiple clicks option.
Disable Button Interval: after each click the button is disabled for the set interval. Default is 0.5 seconds.

Wait for OnClick Animations to finish: enabled by default and visible only if at least one OnClick animation is enabled. 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 waitiong for the OnClick animation to finish.

Button Name: is the trigger for OnButtonClicked events

- **new:** create a new button name in the ButtonNames database
- **rename:** rename the selected button name in the ButtonNames database
- **delete:** removes the selected button name from the ButtonNames database and sets this UIButton's button name to the default value.

is back button: if the current button is a 'Back' button (hides current elements and shows the previous ones)

Add to Navigation: adds the visible elements to the Navigation History

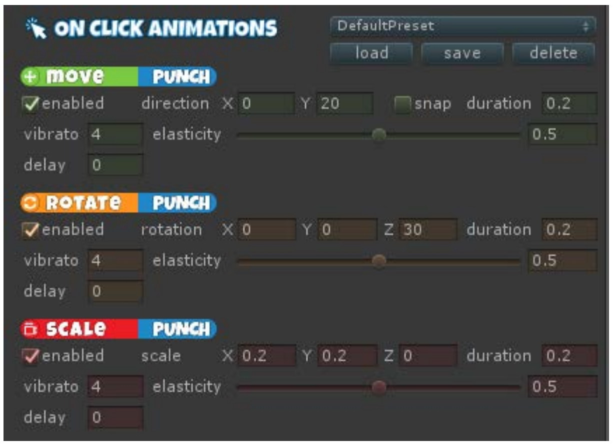
OnClick Sound: the sound filename/event name/sound name (depending if MasterAudio is enabled or not) that will play when the button is clicked

- **new:** create a new button sound in the ButtonSounds database
- **rename:** rename the selected button sound in the ButtonSounds database
- **delete:** removes the selected button sound from the ButtonSounds database and sets this UIButton'sOnClick Sound to the default value.

OnClick Animations: visibility toggle for the button animations that happen when it is clicked; the other 3 buttons are enable/disable toggles for MovePunch, RotatePunch, ScalePunch
Normal Animations: visibility toggle for the button animations that happen when the button is unselected; the other 4 buttons are enable/disable toggles for MoveLoop, RotateLoop, ScaleLoop, FadeLoop
Highlighted Animations: visibility toggle for the button animations that happen when the button is selected; the other 4 buttons are enable/disable toggles for MoveLoop, RotateLoop, ScaleLoop, FadeLoop

SHOW ELEMENTS: the list of elemets this button will show when clicked
HIDE ELEMENTS: the list of element this button will hide when clicked
SEND GAME EVENTS: the list of game events this button will trigger when clicked

NAVIGATION SYSTEM DISABLED: it informs you that the Navigation System is disabled (you can enable it form the Control Panel or from the UIManager) and that Add to Navigation, SHOW ELEMENTS and HIDE ELEMENTS are no longer available.



```
ON CLICK ANIMATIONS          DefaultPreset
                             load   save   delete
move PUNCH
✓enabled   direction X 0   Y 20          snap  duration 0.2
vibrato 4   elasticity                          0.5
delay   0
ROTATE PUNCH
✓enabled   rotation  X 0   Y 0    Z 30   duration 0.2
vibrato 4   elasticity                          0.5
delay   0
SCALE PUNCH
✓enabled   scale     X 0.2 Y 0.2  Z 0    duration 0.2
vibrato 4   elasticity                          0.5
delay   0
```

ON CLICK ANIMATIONS

Preset Zone

selector: select the preset animation settings you want to load
load: load the selected preset
save: save a new preset with the current animation settings (creates a .xml file)
delete: delete the selected preset (deletes the .xml file)
**the OnClick animations presets can be found in .xml format in DoozyUI/Presets/UIAnimations/OnClick*
*** all the animations can be reused in other projects by copying the .xml files*

Settings for all the PUNCH animation types

enabled: toggles the enabled/disabled state of each type of animation
duration: amount (seconds) that the animation will take to finish
vibrato: indicates how much will the punch vibrate
delay: amount (seconds) that the animation will wait before starting

MOVE PUNCH

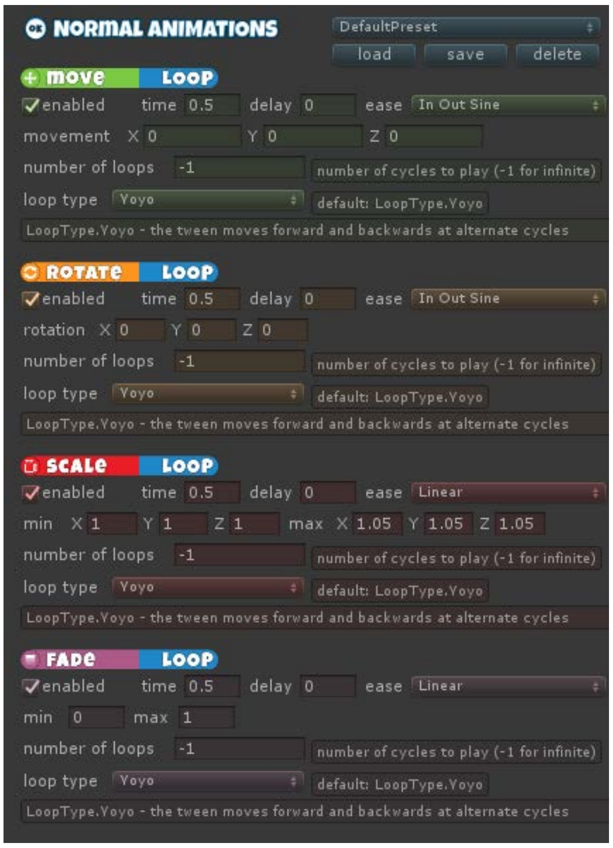
Punches the button's anchoredPosition towards the given direction and then back to the starting one as if it was connected to the starting position via an elastic.
direction: the direction and strength of the punch (added to the Transform's current position)
snap: if TRUE the tween will smoothly snap all values to integers
elasticity: represents how much (0 to 1) the vector will go beyond the starting position when bouncing backwards. 1 creates a full oscillation between the punch direction and the opposite direction, while 0 oscillates only between the punch and the start position

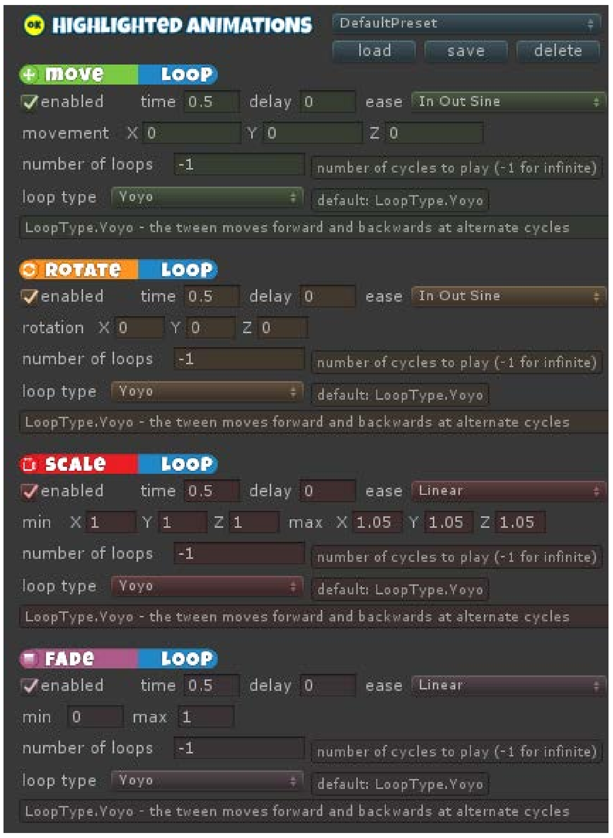
ROTATE PUNCH

Punches a button's localRotation towards the given size and then back to the starting one as if it was connected to the starting rotation via an elastic.
rotation: the punch strength (added to the Transform's current rotation)
elasticity: represents how much (0 to 1) the vector will go beyond the starting rotation when bouncing backwards. 1 creates a full oscillation between the punch rotation and the opposite rotation, while 0 oscillates only between the punch and the start rotation

SCALE PUNCH

Punches a button's localScale towards the given size and then back to the starting one as if it was connected to the starting scale via an elastic.
scale: the punch strength (added to the Transform's current scale)
elasticity: represents how much (0 to 1) the vector will go beyond the starting size 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


```
NORMAL ANIMATIONS                DefaultPreset
                                 load   save   delete
move LOOP
enabled     time 0.5   delay 0    ease In Out Sine
movement    X 0        Y 0        Z 0
number of loops  -1          number of cycles to play (-1 for infinite)
loop type   Yoyo             default: LoopType.Yoyo
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles

ROTATE LOOP
enabled     time 0.5   delay 0    ease In Out Sine
rotation  X 0  Y 0  Z 0
number of loops  -1          number of cycles to play (-1 for infinite)
loop type   Yoyo             default: LoopType.Yoyo
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles

SCALE LOOP
enabled     time 0.5   delay 0    ease Linear
min  X 1  Y 1  Z 1    max  X 1.05  Y 1.05  Z 1.05
number of loops  -1          number of cycles to play (-1 for infinite)
loop type   Yoyo             default: LoopType.Yoyo
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles

FADE LOOP
enabled     time 0.5   delay 0    ease Linear
min  0   max  1
number of loops  -1          number of cycles to play (-1 for infinite)
loop type   Yoyo             default: LoopType.Yoyo
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles
```


```
HIGHLIGHTED ANIMATIONS           DefaultPreset
                                 load   save   delete
move LOOP
enabled     time 0.5   delay 0    ease In Out Sine
movement    X 0        Y 0        Z 0
number of loops  -1          number of cycles to play (-1 for infinite)
loop type   Yoyo             default: LoopType.Yoyo
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles

ROTATE LOOP
enabled     time 0.5   delay 0    ease In Out Sine
rotation  X 0  Y 0  Z 0
number of loops  -1          number of cycles to play (-1 for infinite)
loop type   Yoyo             default: LoopType.Yoyo
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles

SCALE LOOP
enabled     time 0.5   delay 0    ease Linear
min  X 1  Y 1  Z 1    max  X 1.05  Y 1.05  Z 1.05
number of loops  -1          number of cycles to play (-1 for infinite)
loop type   Yoyo             default: LoopType.Yoyo
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles

FADE LOOP
enabled     time 0.5   delay 0    ease Linear
min  0   max  1
number of loops  -1          number of cycles to play (-1 for infinite)
loop type   Yoyo             default: LoopType.Yoyo
LoopType.Yoyo - the tween moves forward and backwards at alternate cycles
```

NORMAL & HIGHLIGHTED ANIMATIONS
Preset Zone
selector: select the preset animation settings you want to load
load: load the selected preset
save: save a new preset with the current animation settings (creates a .xml file)
delete: delete the selected preset (deletes the .xml file)
**the Normal and Highlighted Animations have shared presets; meaning that if you create a Normal Animations preset, it will be available in the Highlighted Animations preset list as well; same goes for deleting a preset*
***the presets are named ButtonLoops can be found in .xml format in DoozyUI/Presets/UIAnimations/ButtonLoops*
**** all the animations can be reused in other projects by copying the .xml files*

Settings for all animation types
enabled: toggles the enabled/disabled state of each type of animation
time: amount (seconds) that the animation will take to finish
delay: amount (seconds) that the animation will wait before starting
ease: the rate of change of the animation over time (for more details about see: http://easings.net)
number of loops: number of cycle to play (-1 for infinite loops)
loop type: sets the looping options (Restart, Yoyo, Incremental) for the tween
LoopType.Restart: *When a loop ends it will restart from the beginning.*
LoopType.Yoyo: *When a loop ends it will play backwards until it completes another loop, then forward again, then backwards again, and so on and on and on.*
LoopType.Incremental: *Each time a loop ends the difference between its endValue and its startValue will be added to the endValue, thus creating tweens that increase their values with each loop cycle.*

Special settings for each animation type
MOVE LOOP
movement: the distance to move on each axis (it is calculated startAnchoredPosition-movement for min and startAnchoredPosition+movement for max)

ROTATE LOOP
rotation: the amount (in degrees) that you want the object to rotate in time (back an forth if LoopType.Yoyo, or forward/backward if LoopType.Incremental).

SCALE LOOP
min: the scale factor at the start for the animation (default: 1); 1 is 100%
max: the scale factor when the animation finished (default: 1.05)

FADE LOOP
min: the alpha value at the start of the animation (default: 0)
max: the alpha value when the animation finished (default: 1)

Code Examples

*remember to add: ***using DoozyUI;***

Simulate a button click from any script
```
string buttonName = "buttonName";
UIManager.SendButtonClick(buttonName);
```

Disables this button by making it non-interactable.

```
UIButton.DisableButtonClicks()
```

Enables this button by making it interactable.

```
UIButton.EnableButtonClicks()
```

Add a game event to this UIButton's gameEvents list.

```
UIButton.AddGameEvent(string eventName)
```

Remove a game event from this UIButton's gameEvents list.

```
UIButton.RemoveGameEvent(string eventName)
```

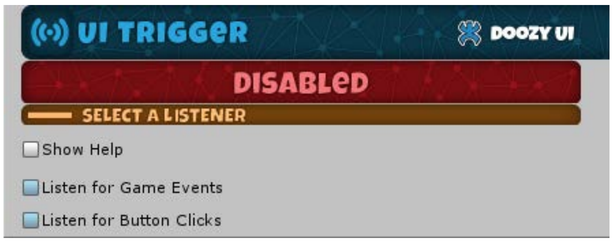
Executes the button click by playing the button sound (if set), starting the OnClick animation (if enabled) and sending the ButtonClick and GameEvents to the UIManager

```
UIButton.ExecuteButtonClick()
```

Sends the ButtonClick and the GameEvents to the UIManager without starting the OnClick animation (if enabled) and playing the button sound (if set)

```
UIButton.SendButtonClickAndGameEvents()
```

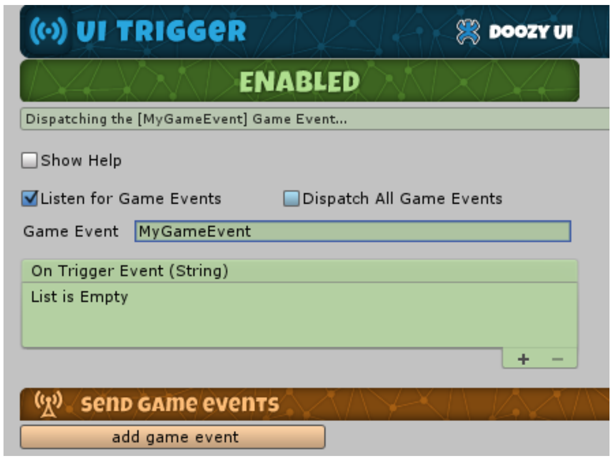
UI Trigger



☐ Show Help

☐ Listen for Game Events

☐ Listen for Button Clicks



Dispatching the [MyGameEvent] Game Event...

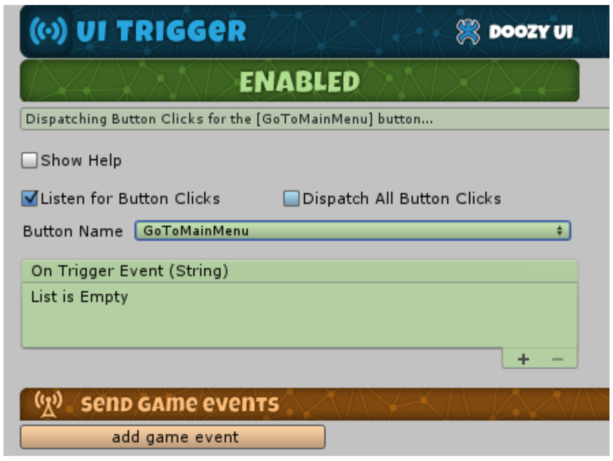
☐ Show Help

☑ Listen for Game Events ☐ Dispatch All Game Events

Game Event: MyGameEvent

On Trigger Event (String)

List is Empty



Dispatching Button Clicks for the [GoToMainMenu] button...

☐ Show Help

☑ Listen for Button Clicks ☐ Dispatch All Button Clicks

Button Name: GoToMainMenu

On Trigger Event (String)

List is Empty

Show Help: shows inspector tooltips on each setting

Listen for Game Events: activates the games events listener options

- **Dispatch All Game Events:** will listen and dispatch ALL game events.
  - 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).
- **Game Event:** the game event you want this trigger to listen for.

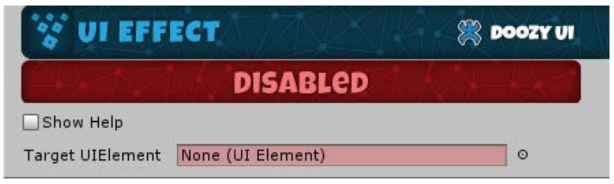
Listen for Button Clicks: activates the button clicks listener options

- **Dispatch All Button Clicks:** will listen and dispatch ALL 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).
- **Button Name:** the button name you want this trigger to listen for.

On Trigger Event *(String)*: the methods that you want triggered (uses native UnityEvent)

SEND GAME EVENTS: the list of game events this trigger will send when triggered either by a button click or a game event

UI Effect

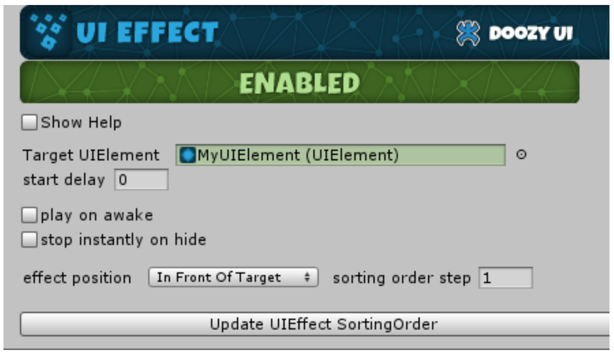

UI EFFECT

DOOZY UI

DISABLED

Show Help

Target UIElement None (UI Element)


UI EFFECT

DOOZY UI

ENABLED

Show Help

Target UIElement MyUIElement (UIElement)

start delay 0

play on awake

stop instantly on hide

effect position In Front Of Target sorting order step 1

Update UIEffect SortingOrder

Show Help: shows inspector tooltips on each setting

Target UIElement: the UIElement this effect will be linked to for the show and hide events

start delay: : after the show event, for the target element, has been issued, you can set a start delay before the effect plays

play in awake: just like the UIElement's 'start hidden' option, the play on awake should be used if this effect will be visible when the scene strts. Otherwise you should let this option unchecked

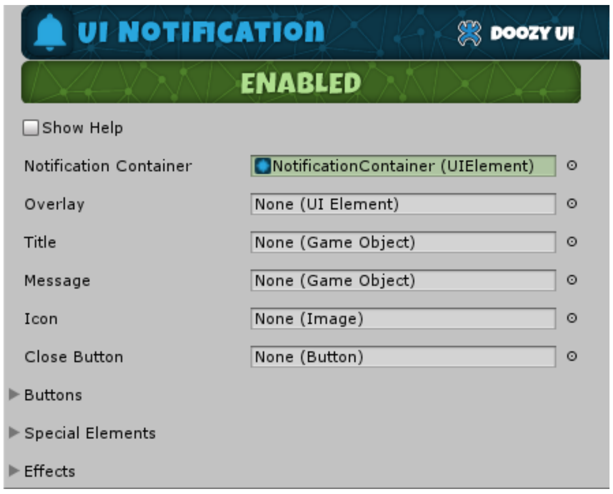
stop instantly on hide: when the hide event, for the target element, has been issued, you can stop and clear the effect instantly (it will dissapear) or you can let it fade out (for the lifetime duration of the particles)

effect position: this option adjusts the sorting order 'moving' the effect in front or behind the target UIElement's canvas sorting order

sorting order step: this option goes hand in hand with the previous one, by selecting how many steps/levels should the effect sorting order be ajusted with. Example: if the target UIElement’s canvas sorting order is 100 and we have a sorting order step of 5 then the effect can be either at 95 (if behind) or 105 (if in front)

Update UIEffect SortingOrder: this option helps you adjust/update the sorting order on the fly whenever you want (this also updates any child objects with ParticleSystem components)

UI Notification



Show Help

Notification Container	NotificationContainer (UIElement)
Overlay	None (UI Element)
Title	None (Game Object)
Message	None (Game Object)
Icon	None (Image)
Close Button	None (Button)

Buttons

Special Elements

Effects

The UI Notification 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.

Show Help: shows inspector tooltips on each setting

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

- It should contain everything besides the overlay
- To play sounds when it appears and when it dissapears you can add them in the attached UIElement IN and OUT animations

Overlay: the fullscreen overlay / color tint

Title: the text that will serve a the notification's title; a gameobject that should have either a Text or a TextMeshProUGUI component attached; If you used a TextMeshProUGUI componenet make sure TextMeshPro it is enabled in the UIManager or the Control Panel.

Message: the text that will serve a the notification's message; a gameobject that should have either a Text or a TextMeshProUGUI component attached; If you used a TextMeshProUGUI componenet make sure TextMeshPro it is enabled in the UIManager or the Control Panel.
Icon: every notification can have a customized icon or you can leave it null. Link the Image component here and pass in the icon Sprite when you call the ShowNotification method

Close Button: This 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)

Buttons: These 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. They can also have either a Text or a TextMeshProUGUI component attached, that you can set in ShowNotification.

Special Elements: 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.

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

Code Examples

*remember to add: ***using DoozyUI;***

Show an UINotification with no title, message, icon or buttons from any script

```
string prefabName = "prefabName";
float lifetime = 3f;
bool addToNotificationQueue = true;
UIManager.ShowNotification(prefabName, lifetime, addToNotificationQueue);
```

OR

Show an UINotification with no title, message, icon or buttons from any script

```
GameObject prefab;
float lifetime = 3f;
bool addToNotificationQueue = true;
UIManager.ShowNotification(prefab, lifetime, addToNotificationQueue);
```

Show an UINotification with title and message, but without icon and buttons from any script

```
string prefabName = "prefabName";
float lifetime = 3f;
bool addToNotificationQueue = true;
string title = "myTitle";
string message = "myMessage";
UIManager.ShowNotification(prefabName, lifetime, addToNotificationQueue, title, message);
```

OR

Show an UINotification with title and message, but without icon and buttons from any script

```
GameObject prefab;
float lifetime = 3f;
bool addToNotificationQueue = true;
string title = "myTitle";
string message = "myMessage";
UIManager.ShowNotification(prefab, lifetime, addToNotificationQueue, title, message);
```

Show an UINotification with title, message and icon, but without buttons from any script

```
string prefabName = "prefabName";
float lifetime = 3f;
bool addToNotificationQueue = true;
string title = "myTitle";
string message = "myMessage";
Sprite icon = myIcon; //this is a referenced sprite
UIManager.ShowNotification(prefabName, lifetime, addToNotificationQueue, title, message, icon);
```

OR

Show an UINotification with title, message and icon, but without buttons from any script

```
GameObject prefab;
float lifetime = 3f;
bool addToNotificationQueue = true;
string title = "myTitle";
string message = "myMessage";
Sprite icon = myIcon; //this is a referenced sprite
UIManager.ShowNotification(prefab, lifetime, addToNotificationQueue, title, message, icon);
```

Show an UINotification with title, message, icon and buttons from any script

```
string prefabName = "prefabName";
float lifetime = 3f;
bool addToNotificationQueue = true;
string title = "myTitle";
string message = "myMessage";
Sprite icon = myIcon; //this is a referenced sprite
string[] buttonNames = new string[] { "button1", "button2", "button3" };
string[] buttonTexts = new string[] { "Yes", "No", "Later" };
UIManager.ShowNotification(prefabName, lifetime, addToNotificationQueue, title, message, icon, 
buttonNames, buttonTexts);
```

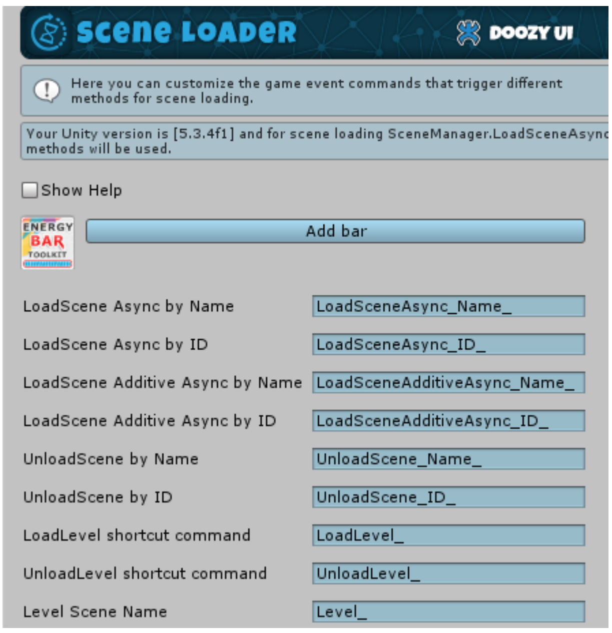
OR

Show an UINotification with title, message, icon and buttons from any script

```
GameObject prefab;
float lifetime = 3f;
bool addToNotificationQueue = true;
string title = "myTitle";
string message = "myMessage";
Sprite icon = myIcon; //this is a referenced sprite
string[] buttonNames = new string[] { "button1", "button2", "button3" };
string[] buttonTexts = new string[] { "Yes", "No", "Later" };
UIManager.ShowNotification(prefab, lifetime, addToNotificationQueue, title, message, icon, 
buttonNames, buttonTexts);
```

NOTE: You can **show a notification** in any combination you want, as **there are 24 overloads available**. Just be sure that the notification you are trying to show has all the elements attached (there will not be any NullReferenceExceptions, but for example, if you want show a notification and set a title, but the notification does not thave a Text or a TextMeshProUGUI component linked in the title field, then you won't see that title).

SceneLoader

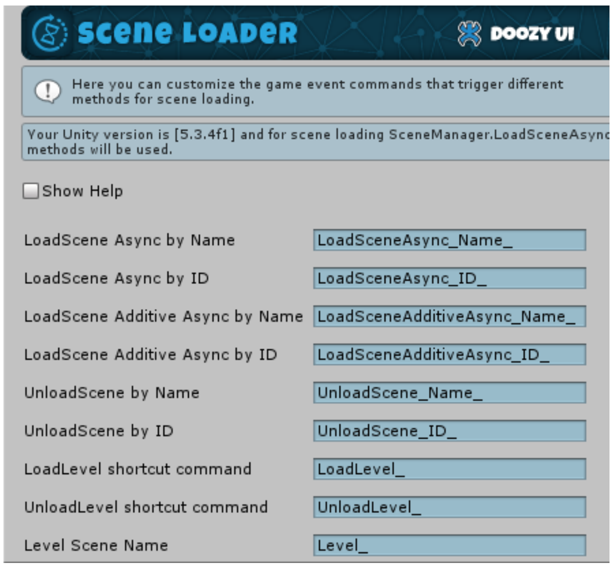


Here you can customize the game event commands that trigger different methods for scene loading.

Your Unity version is [5.3.4f1] and for scene loading SceneManager.LoadSceneAsync methods will be used.

Show Help

LoadScene Async by Name	LoadSceneAsync_Name_
LoadScene Async by ID	LoadSceneAsync_ID_
LoadScene Additive Async by Name	LoadSceneAdditiveAsync_Name_
LoadScene Additive Async by ID	LoadSceneAdditiveAsync_ID_
UnloadScene by Name	UnloadScene_Name_
UnloadScene by ID	UnloadScene_ID_
LoadLevel shortcut command	LoadLevel_
UnloadLevel shortcut command	UnloadLevel_
Level Scene Name	Level_



Here you can customize the game event commands that trigger different methods for scene loading.

Your Unity version is [5.3.4f1] and for scene loading SceneManager.LoadSceneAsync methods will be used.

Show Help

LoadScene Async by Name	LoadSceneAsync_Name_
LoadScene Async by ID	LoadSceneAsync_ID_
LoadScene Additive Async by Name	LoadSceneAdditiveAsync_Name_
LoadScene Additive Async by ID	LoadSceneAdditiveAsync_ID_
UnloadScene by Name	UnloadScene_Name_
UnloadScene by ID	UnloadScene_ID_
LoadLevel shortcut command	LoadLevel_
UnloadLevel shortcut command	UnloadLevel_
Level Scene Name	Level_

INFO

After a scene has been loaded the SceneLoader sends a GameEvent named ‘LevelLoaded’. You can listen (with a trigger) for it to know when the loading has been completed.

NOTE: *For Unity versions 5.1 and 5.2 for scene loading Application.LoadLevel methods will be used. For Unity versions 5.3 and up for scene loading SceneManager.LoadSceneAsync methods will be used.*

Show Help: shows inspector tooltips on each setting

If EnergyBarToolkit support is enabled

Add bar: adds the option to link as many loading bars you need to show the loading progress of the scenes.

SceneLoader commands (game events)

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]

not available in Unity 5.1 & 5.2

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]

not available in Unity 5.1 & 5.2

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]
not available in Unity 5.1 & 5.2 | 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'

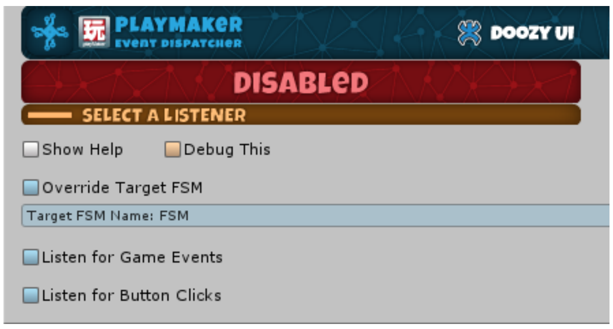
Code Examples

*remember to add: ***using DoozyUI;***

SceneLoader methods

```
SceneLoader.Instance.LoadSceneAsync(sceneName); //string

SceneLoader.Instance.LoadSceneAsync(sceneBuildIndex); //int

SceneLoader.Instance.LoadLevelAdditiveAsync(sceneName); //string

SceneLoader.Instance.LoadLevelAdditiveAsync(sceneBuildIndex); //int

SceneLoader.Instance.LoadLevel(levelNumber);

SceneLoader.Instance.UnloadLevel(levelNumber); //available only for unity 5.3 and up
```

PlaymakerEventDispatcher



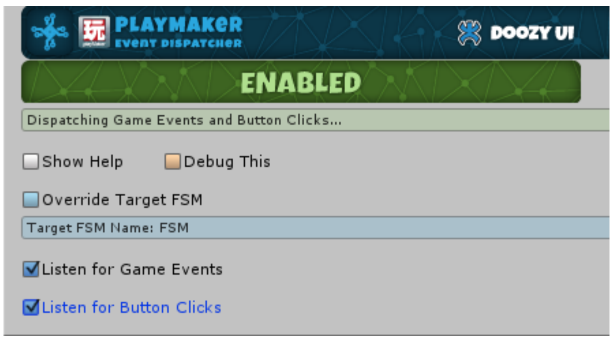
Show Help Debug This

Override Target FSM

Target FSM Name: FSM

Listen for Game Events

Listen for Button Clicks



Dispatching Game Events and Button Clicks...

Show Help Debug This

Override Target FSM

Target FSM Name: FSM

✓ Listen for Game Events

✓ Listen for Button Clicks

Show Help: shows inspector tooltips on each setting
Debug This: prints to Debug.Log all the Game Events and/or Button Clicks this dispatcher sends to the targetFSM

***Note:** this dispatcher auto targets the first FSM on this GameObject. You can override that and reference the FSM you want to target*

Override Target FSM: allows for setting as target FSM a different FSM than the auto selected one.

*Notice the **Target FSM Name:** help box. It shows the name of the FSM that this dispatcher sends messages to.*

Listen for Game Events: dispatches all the game events to the target FSM
Listen for Button Clicks: dispatches all the button clicks to the target FSM

Notes

***

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.

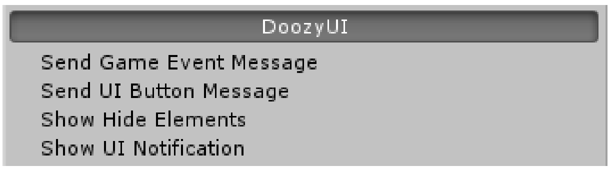
****

To dispatch Game Events, you have to create, in the FSM, events named exactly as the Game Event commands you wants to catch.

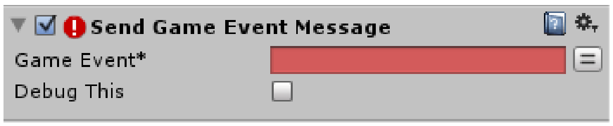
*****

To dispatch Button Clicks, you have to create, in the FSM, events named exactly as the buttonNames you wants to catch.

PlayMaker Actions

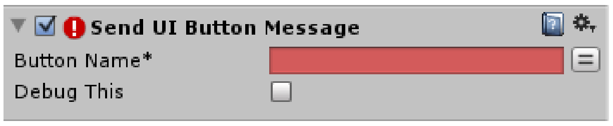

DoozyUI
Send Game Event Message
Send UI Button Message
Show Hide Elements
Show UI Notification

Send Game Event Message


Send Game Event Message
Game Event*
Debug This

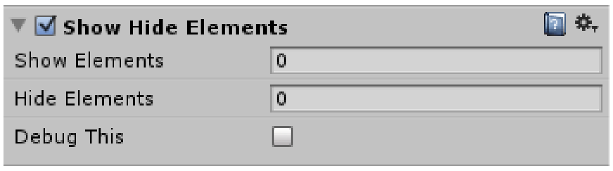
Game Event: the game event you want to send
Debug This: prints to Debug.Log whenever this action is triggered

Send UI Button Message


Send UI Button Message
Button Name*
Debug This

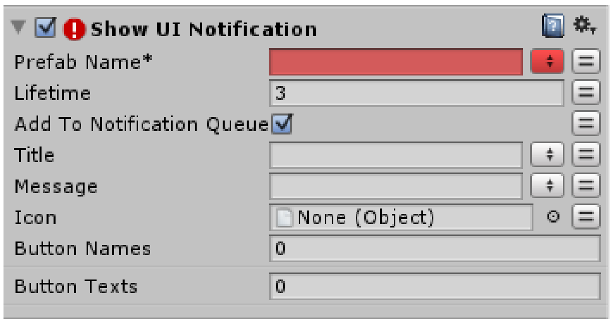
Button Name: the button click you want to simulate
Debug This: prints to Debug.Log whenever this action is triggered

Show Hide Elements


Show Hide Elements
Show Elements 0
Hide Elements 0
Debug This

Show Elements: a list of all the UIElements you want to show
Hide Elements: a list of all the UIElements you wan to hide
Debug This: prints to Debug.Log whenever this action is triggered

Show UI Notification


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

Prefab Name: the name of the notification prefab you want to show; it needs to be in a Resources folder

Lifetime: how long will the notification be on the screen. Infinite lifetime is -1.(default: 3)
** lifetime doesn't take into account the IN and OUT animations time of the Notification.*

Add To Notification Queue: should this notification be added to the NotificationQueue or shown right away. (default: true)

Title: the text you want to show in the title area (if linked); it can be either a Text or a TextMeshProUGUI component. (default: string.Empty)

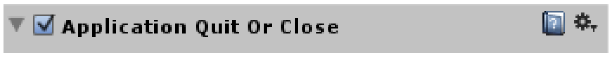
Message: the message you want to show in the message area (if linked); it can be either a Text or a TextMeshProUGUI component. (default: string.Empty)

Icon: the sprite you want the notification icon to have (if linked) (default:null)

Button Names: the button names you want the notification to have (from left to right). These values are the ones that we listen to as button click. (default: null)

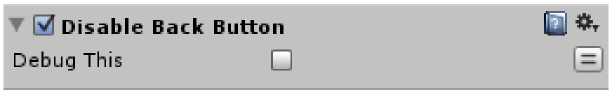
Button Texts: the text on the buttons (example: 'OK', 'Cancel', 'Yes', 'No' and so on) (default: null)

Application Quit or Close


Application Quit Or Close

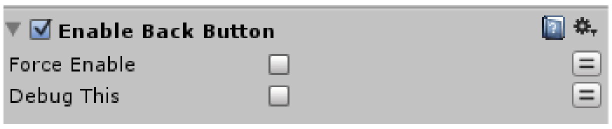
When the ApplicationQuit command has been issued we either exit play mode (if in editor) or we quit the application if in build mode.

Disable Back Button


Disable Back Button
Debug This ☐

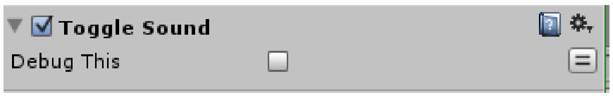
Disable the back button by adding another disable level to the additive bool.
Debug This: prints to Debug.Log whenever this action is triggered

Enable Back Button


Enable Back Button
Force Enable ☐
Debug This ☐

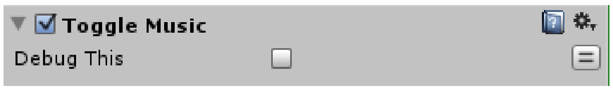
Enables the back button by subtracting another disable level from the additive bool. Forced enable cleares the additive bool levels.
Force Enable: enables the back button by resetting the bool level to 0
Debug This: prints to Debug.Log whenever this action is triggered

Toggle Sound


Toggle Sound
Debug This ☐

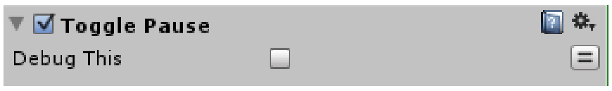
Toggles the sound
Debug This: prints to Debug.Log whenever this action is triggered

Toggle Music


Toggle Music
Debug This ☐

Toggles the music
Debug This: prints to Debug.Log whenever this action is triggered

Toggle Pause


Toggle Pause
Debug This ☐

Toggles pause through timescale adjustment
Debug This: prints to Debug.Log whenever this action is triggered

Navigation System

General Info

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

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.

UIManager is the one keeping all the information about what to show and what to hide. It also listens for button clicks and reacts accordingly.

Special Buttons

- **GoToMainMenu**
    - goes directly to the MainMenu and clears the Navigation History
    - it is useful if you are in game and you need to jump to the main menu without being able to go back
- **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

- **ClearNavigationHistory**
    - cleares the NavigationHistory
    - be aware though that if the player presses the 'Back' button, when the Navigation History is empty, it will trigger show 'QuitMenu'
- **DisableBackButton**
    - disables the 'Back' button functionality
- **EnableBackButton**
    - enables the 'Back' button functionality (the 'Back' button is enabled by default)
- **SoundCheck**

- o does a sound check and shows the proper state for the sound button toggle

- **MusicCheck**
    - o Does a music check and shows the proper state for the music button toggle

Sound Settings

General Info

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

You may have noticed that ElementSounds and ButtonSounds require a strings to trigger. That is because we are loading the sounf 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.

If you enabled the MasterAudio support then all you have to do is reference the sounds to the MasterAudio manager and call them by soundName or eventName. There are 2 methods available:

1. PlaySoundAndForget – will call that exact method and play the sound with the given name.
2. FireCustomEvent – will call that exact method and trigger the MasterAudio event that plays whatever sound you set up.

Code Examples

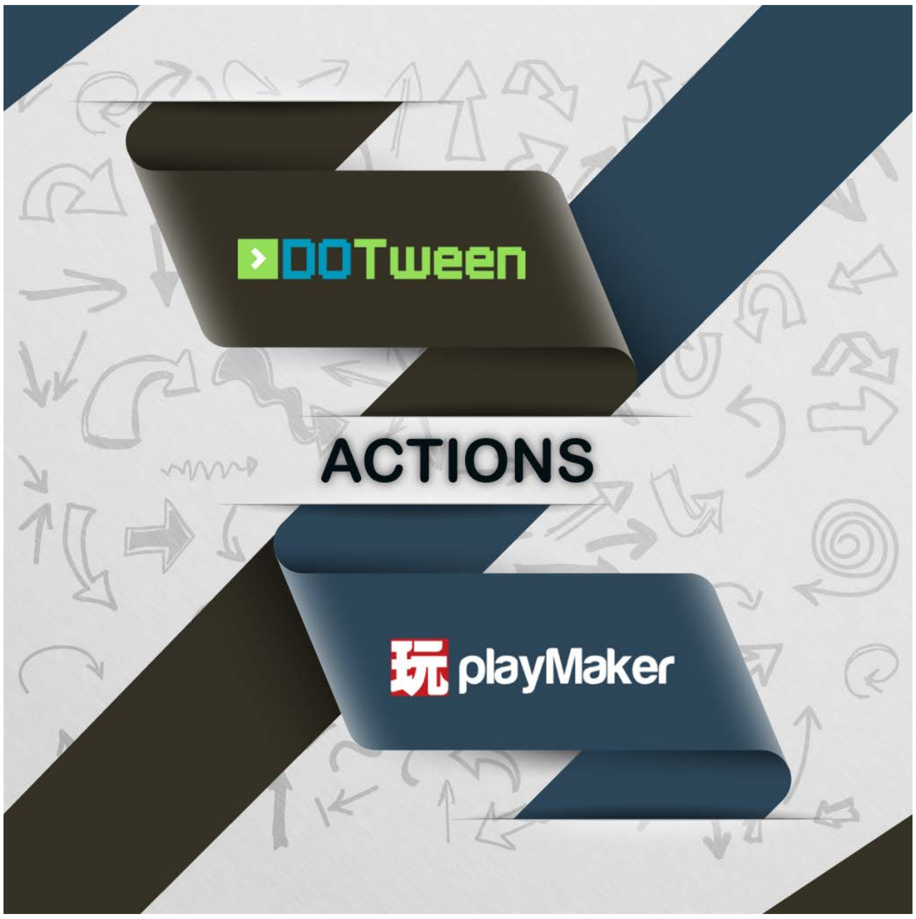
*remember to add: ***using DoozyUI;***

Play a sound from any script through DoozyUI

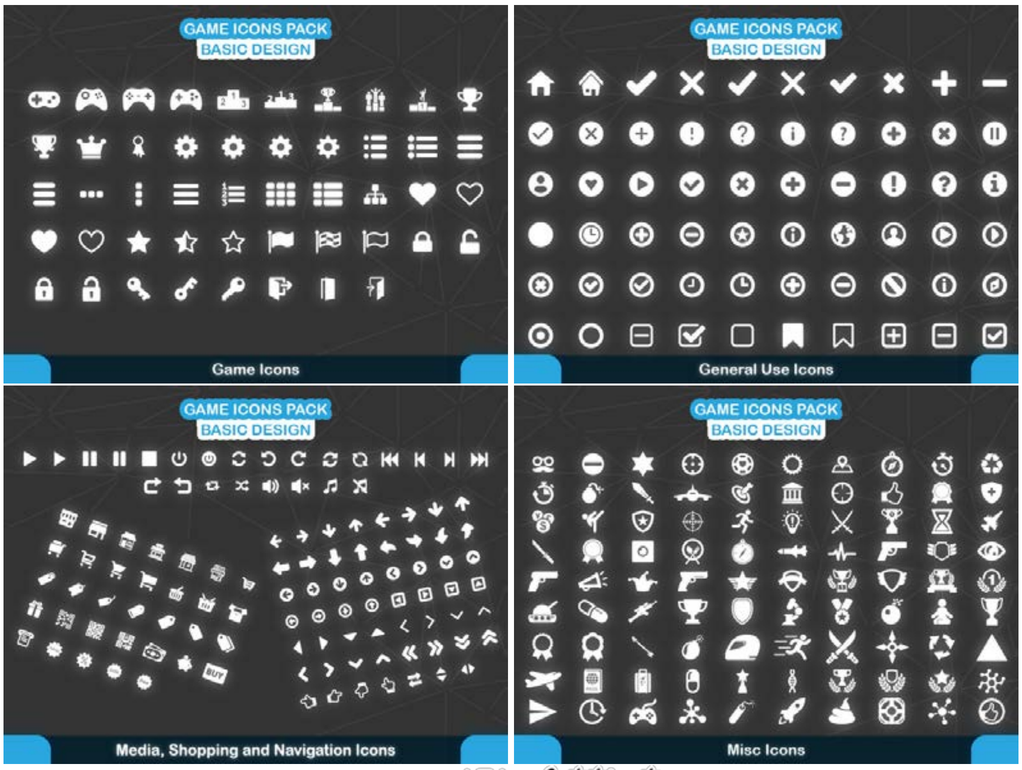
```
string soundName = "soundName";
UIAnimator.PlaySound(soundName);
```

Final Words

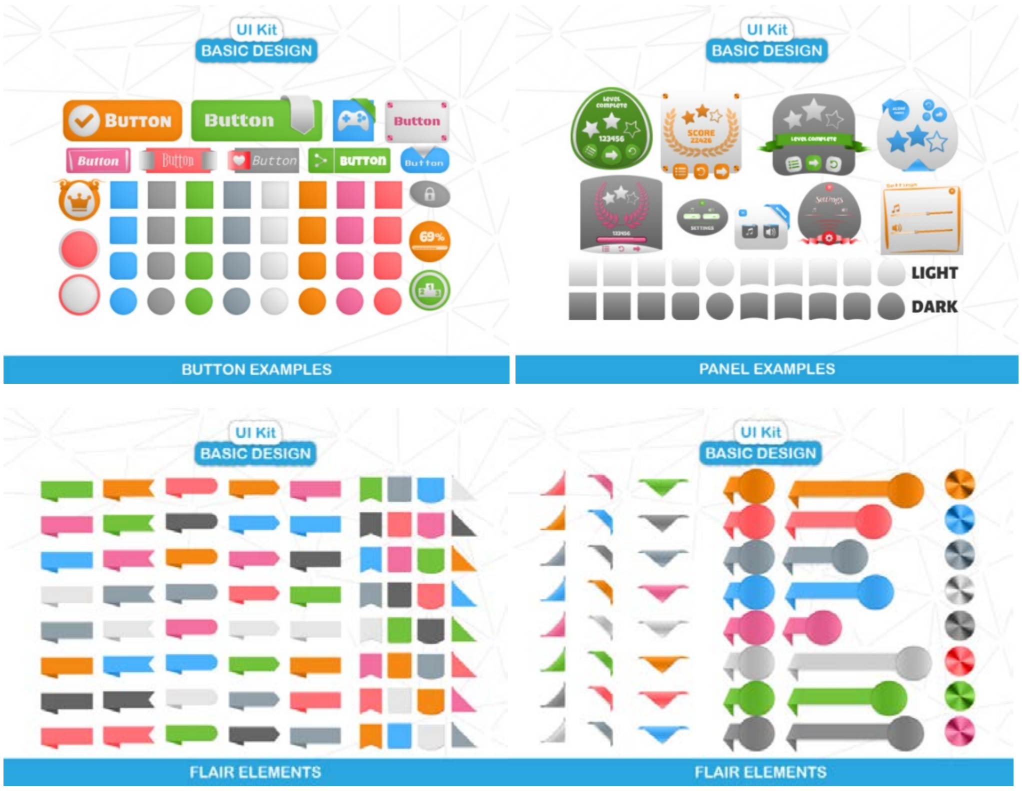
- Support is available by emailing doozy.entertainment@gmail.com.
- Make sure you check out our other assets such as:
  - Playmaker Actions for DOTween by Doozy - http://u3d.as/kRs



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



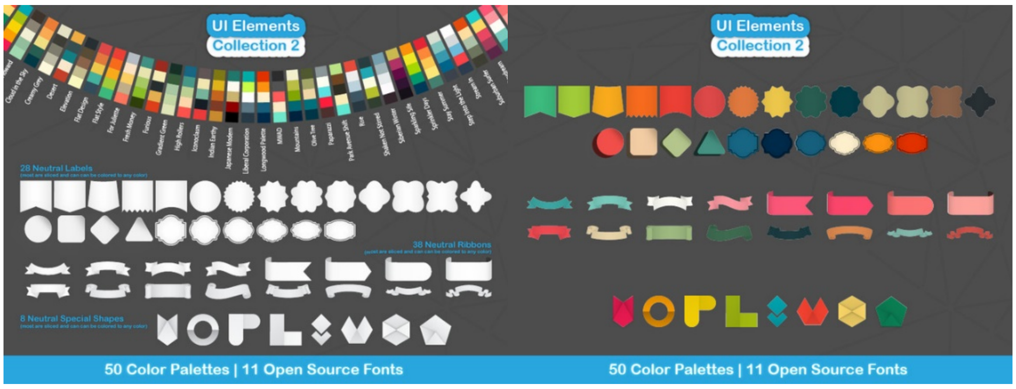
- UI Kit - Basic Design - http://u3d.as/fyv



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

UI Elements
Collection 1
10 Neutral Ribbons
7 Neutral Panels
10 Retro Frames (5 gold, 5 silver)
...and several other UI Designs and Elements
Several other UI Designs and Elements
40 Color Palettes | 11 Open Source Fonts
40 Color Palettes | 11 Open Source Fonts
POLE DESIGN
PUZZLE DESIGN

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



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