

ZUI by Ziad Khalid

The best-easiest menu management tool!

Features

UI Element component to handle the visibility of your elements and animate them while they're switching (coming into, or going out of the screen).

Menus management system that manages switching and animating full screen menus and side menus.

Pop-ups management system that manages switching and animating pop-ups.

UI Animation component to give you full control over animating the elements by adding as many animation frames as you.

SFX system for every menu, side menu, pop-up and UI elements, to play audio when switching visibility.

UI assistance tools to help you create your UI.

Creation window to make it easier for you to use all of the above features

For support, please contact me at

contact@ziadkhalid.com

Feel free to send suggestions too!

Table of Contents

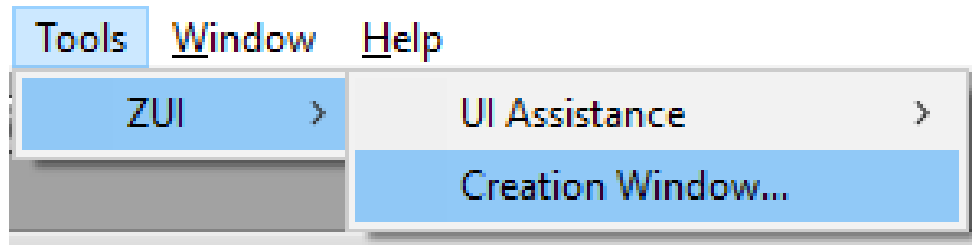
First Things First!	4
Creating Menus	6
Creating Animated Elements	12
Switching Between Menus	15
Multi-menus Animated Elements	18
Creating Pop-ups	20
Using Pop-ups	22
Creating Side-menus	24
Using Side-menus	26
Creating UI Element Groups	27
Using UI Element Groups	28
UI Animation	29
Using Sound Effects	31
Components In Details	32
ZUIManager	32
UIElement	37
Menu	42
Popup	45
SideMenu	47
UIElementsGroup	49
UIAnimation	52
SFXManager	53
UI Assistance Tools	54
Tips & Tricks	58
Animating the Entire Menu as One Block	58

Smooth Cartoonish Transitions	58
Persistent Back Button	58
One Pop-up for Many Reasons	59
Custom Animation Clips When Visibility Change.....	59
Goodbye! - Contact Me!	60

First Things First!

First thing you have to do is add the managers!

And the way to do this is:

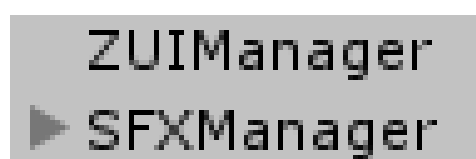


By going to the menu bar, *Tools>ZUI>Creation Window...*

This would open the creation window, here you can create the objects you need to control your UI.



What we're interested in, for now, is "Setup" button, once you click on it, it will create the managers you need.



"ZUIManager" is the main manager, while "SFXManager" is made to handle sound effects.

More details about the managers at ["ZUIManager In Details"](#) and ["SFXManager In Details"](#) sections.

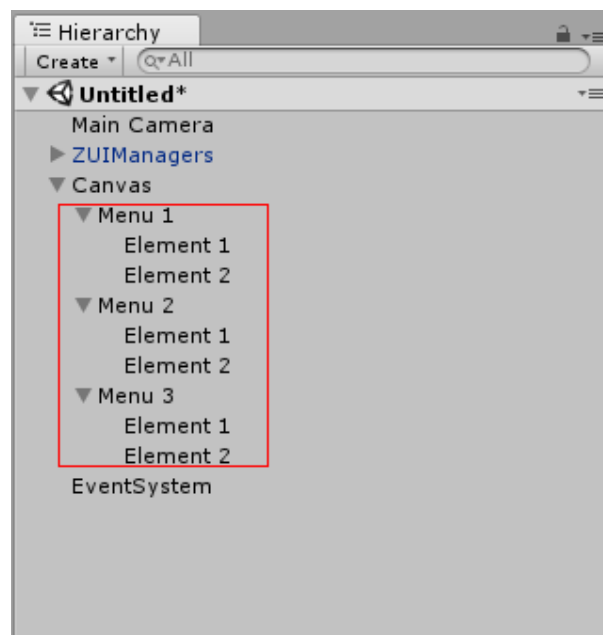
There's also another way that allows you to manually create them - just like the rest of the objects in the creation window - by creating an empty Game Object and put the necessary scripts on it, in this case you can find "ZUIManager.cs" and "SFXManager.cs" in the scripts folder at ZUI>Scripts .

Now that you have your managers ready, let's move to the next step.

Creating Menus

A menu holds and controls a number of UI Elements, by ordering those elements to animate whenever the menu is opening or closing (changing visibility). Only one menu can be visible at a time, whenever another menu is requested to be opened, the current one closes.

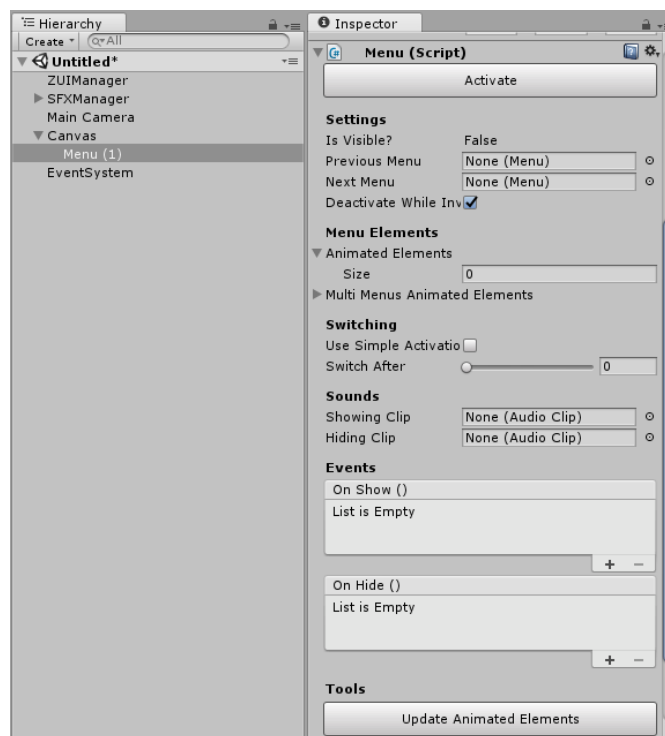
The structure is simple; each menu should have a Game Object that is the parent - which rectangle usually fills the entire screen - and a set of elements under it as children, those children are your actual UI elements that players would see (images, buttons, etc...). So here's an example of how menus would look like in the hierarchy:



And again, you can create a menu by using the "Creation Window" at the menu bar under *Tools>ZUI>Creation Window...*



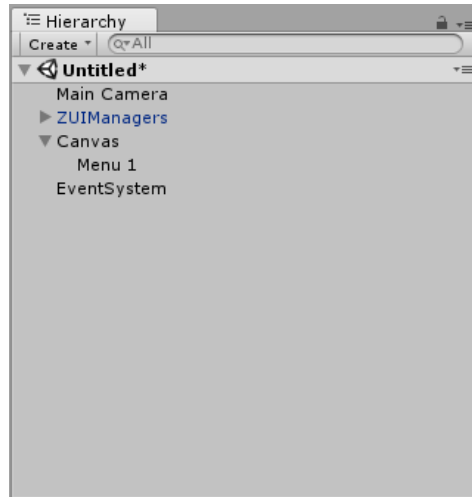
This will create a menu for you inside the current selected canvas (or the parent canvas of the current selected Game Object), or the last canvas you created if you're not selecting any.



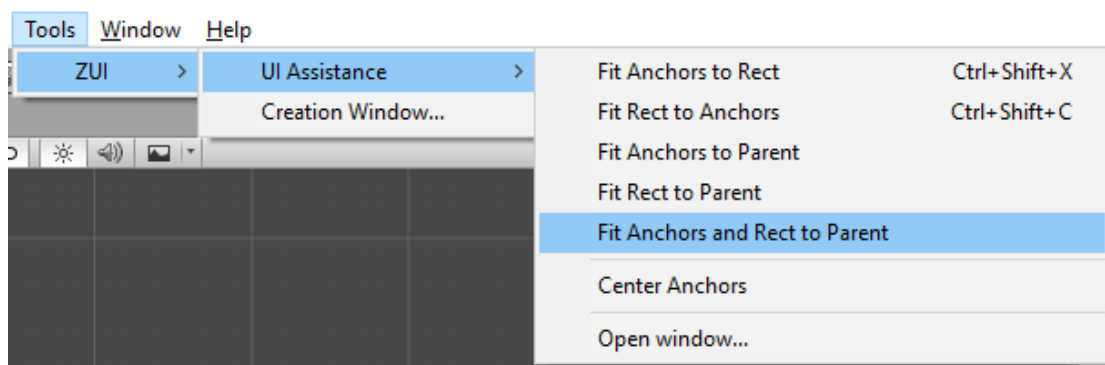
You now have your first menu ready to add elements to it.

But first, let's explain what just happened, by walking you through the steps of manually creating menus (rather than using the creation window).

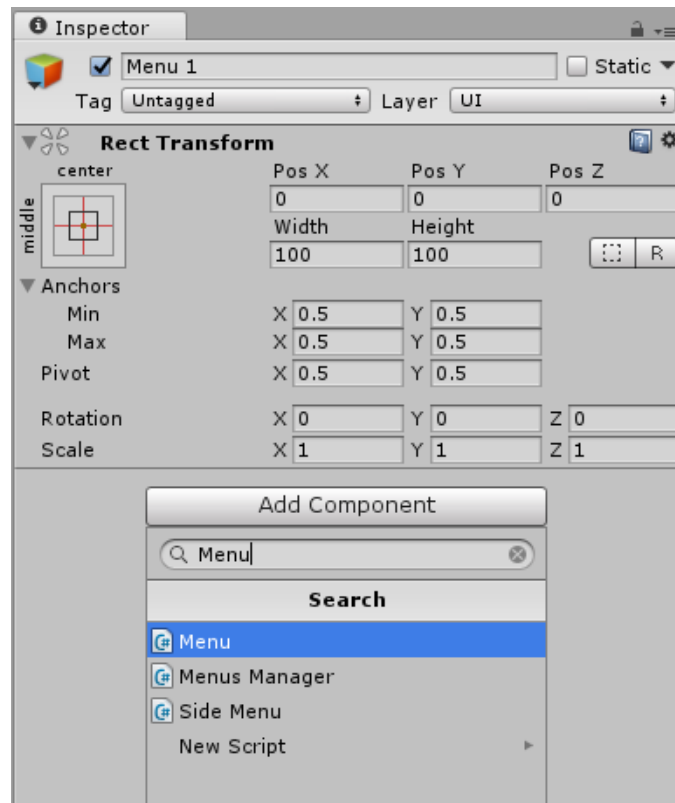
You're going to start off by creating a canvas, then create an empty Game Object under it, and give it a name.



And since menus usually fill the entire screen you have to resize the Rect and Anchors of the Game Object you just created. You can do this manually, or by simply clicking on *Tools>ZUI>UI Assistance>Fit Anchors and Rect to Parent*

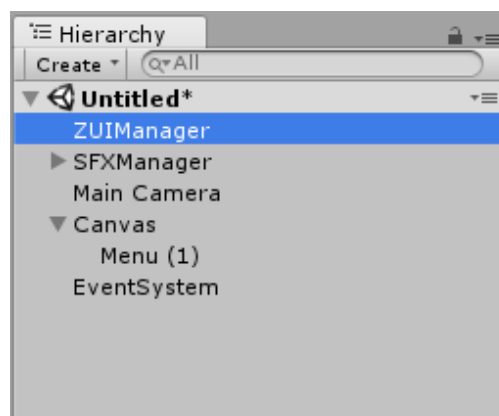


Now that you've the Game Object ready, you need to add "Menu" script to it. Select the menu Game Object, and in the inspector, click "Add Component", type "Menu", and click on it.

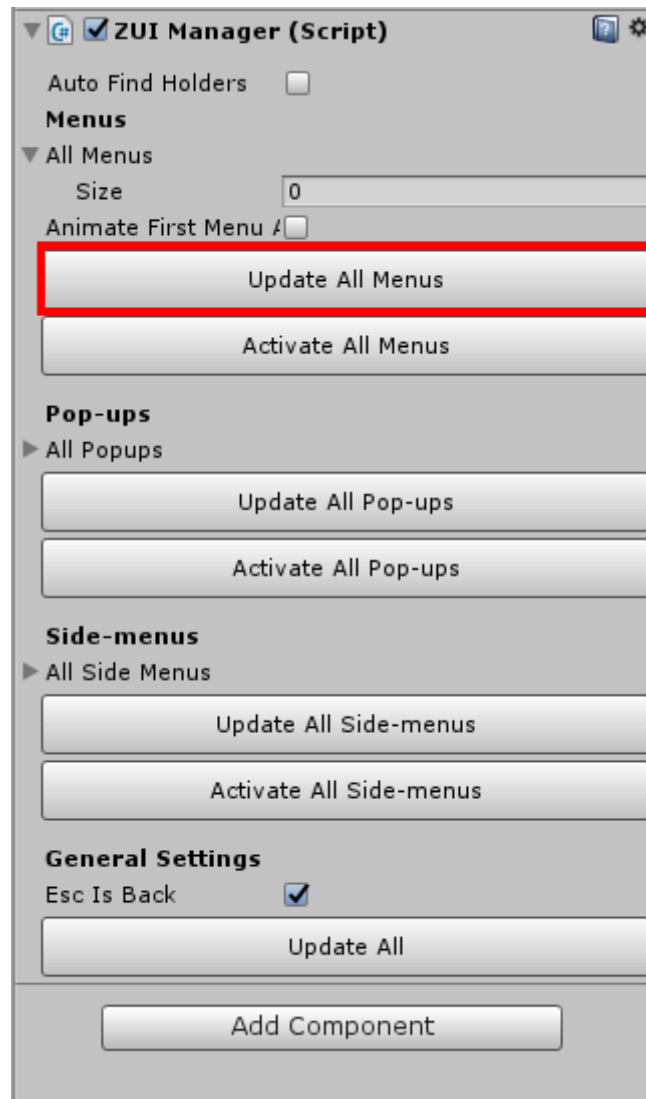


You can also find "Menu" script in your project folder under *Assets>ZUI>Scripts*.

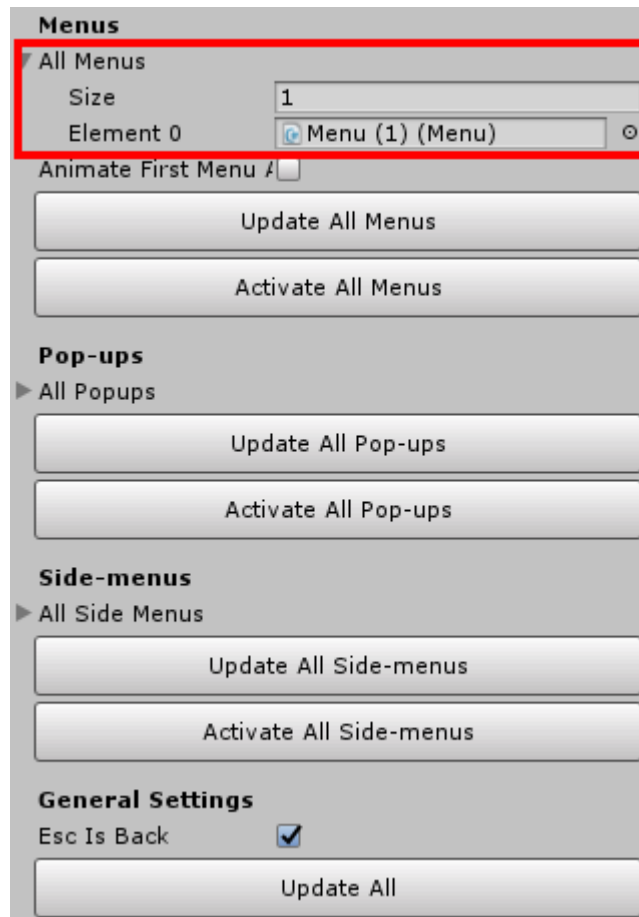
Last step in the process of creating the holder of the menu is to tell the manager (only if "Auto Find Holders" in "ZUIManager" is disabled), and you can do this by selecting the Game Object called "ZUIManager".



And simply click on "Update All Menus"



ZUIManager will then automatically find all the menus under all the canvases and put them in the list.



Note: Always remember that if "Auto Find Holders" is disabled then "ZUIManager" must know about all the menus, pop-ups and side-menus in your scene; because you will use it to manage them.

If you use the "Creation Window" it will automatically tell "ZUIManager" about them, otherwise you'll have to update the list yourself.

More details about menus at "[Menus In Details](#)" section.

Creating Animated Elements

Animated elements or UI Elements, are the core of the system, they are the objects that add life to your interface, they are the ones that animate!

Giving them an order to show or hide will start an animation based on the parameters you give them, this order can be given by a holder (Menu/Side-menu/Pop-up/UI Element Group), or you can have an independent element and change its visibility whenever you wish. For now we'll stick with having a holder control them for us.

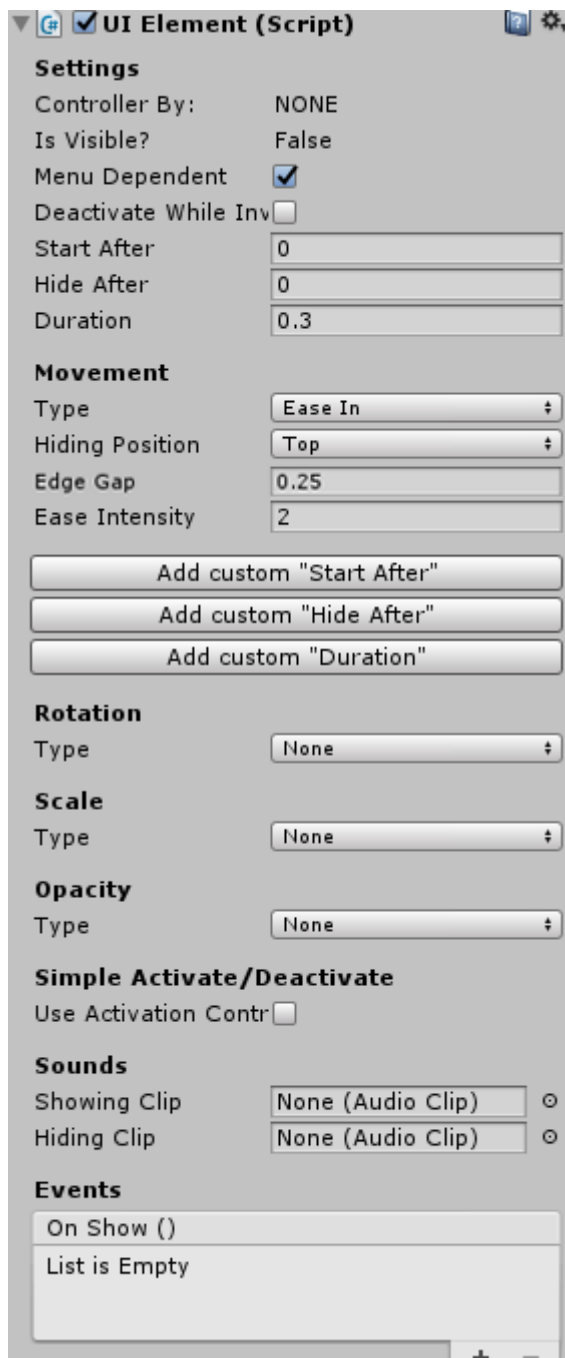
Each time a holder changes visibility, it tells all the elements it controls about it, so they change visibility too by playing an animation, whether it's showing or hiding.

Let's create a button under our menu (a child), by right clicking on it, and selecting *UI>Button*

Now we have to add "UIElement" script to it, you do this by selecting the button, clicking on "Add Component" and then type "UIElement", or drag and drop the script from *Assets>Scripts>UIElement* onto the button.

Note: Keep in mind that every element under the menu should have a UIElement script attached to it or to one of its parent, otherwise it will remain static and won't go out of or in to the screen when the menu is changing visibility.

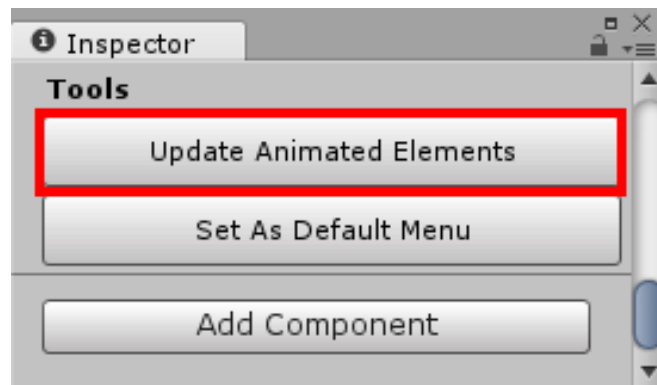
This is how the script look like:



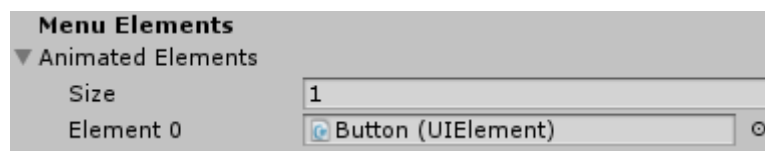
For now let's keep things simple, we will leave everything as it is by default, and we'll understand the details at "[UIElement In Details](#)" section.

Now that you've created a new "UIElement", You have to tell whatever is carrying it about it, which in this case is the menu we've created earlier.

Just like you did with "ZUIManager", select the menu Game Object, and click on "Update Animated Elements" in "Tools" section of the "Menu" component (or any holder other than a menu).



This will automatically update the list with all the "UIElements" under the menu Game Object (children), of course you can add elements by manually putting them inside the list too, instead of clicking the button.



Make sure that all the elements are in this list, otherwise they will remain on screen when they supposed to hide.

Note: you have to make sure that the element you want to put under the menu's control is marked "Menu Dependent" otherwise it will not be added to the list, more information in "UIElement In Details" section.

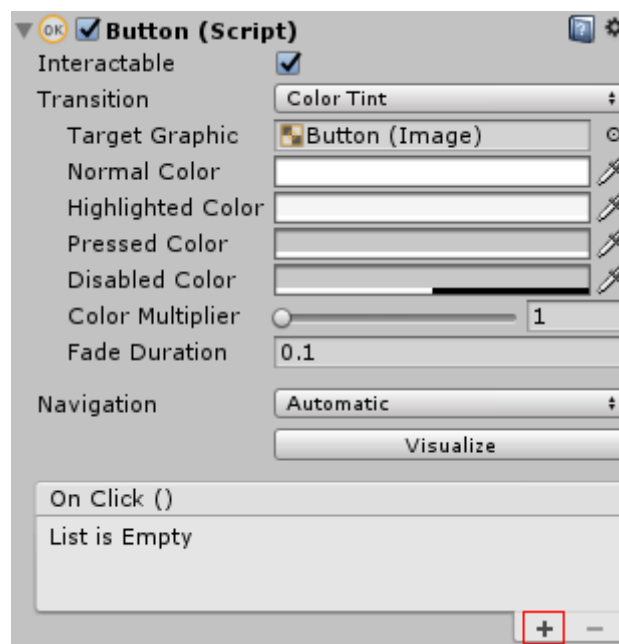
Switching Between Menus

Now we need to see the system in action, we need to switch between menus, but first, we need another menu to switch to.

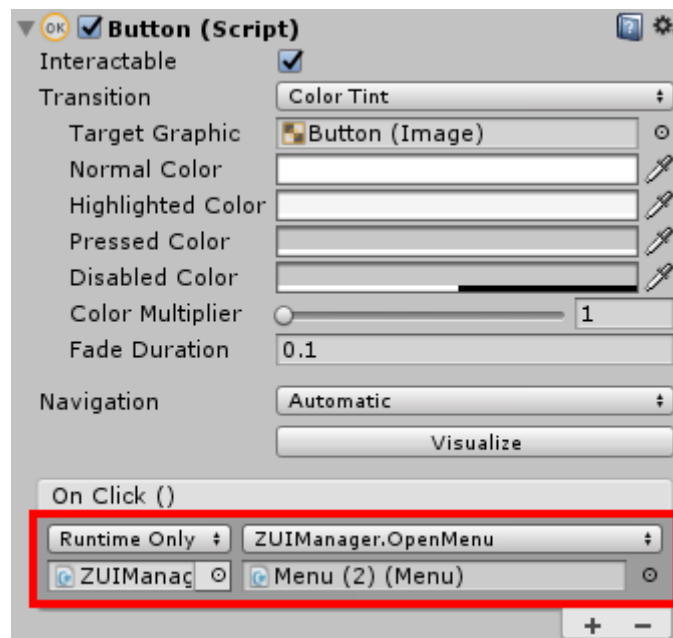
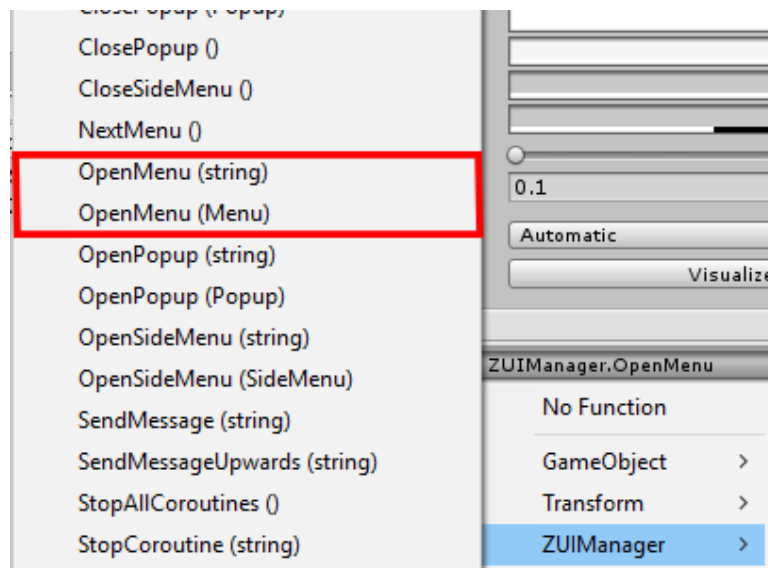
Create another menu, with the same steps mentioned before (click on "Menu" in "Creation Window"), add as many UI elements as you want, and make sure both menus look different just so you can clearly notice the change.

Now that you have both menus ready and you've told "ZUIManager" about them (if "Auto Find Holders" is disabled), all you have to do now is select a button inside the first menu, and add an action to it, the action is to open the second menu.

You do this by adding a listener to the On Click event of the button

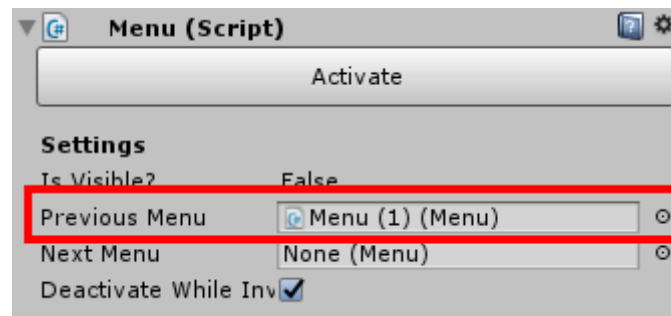


And the listener is *OpenMenu* function that you can find under the Game Object *ZUIManager>OpenMenu*, there are two variations of this function, one that takes the name of the menu as a parameter and the other takes the reference, We'll use the second one for this example.



Drag and drop the second menu, then test it out and you'll find that the menu switches once you click on the button and the animation you've made is playing (everything will move to the top of the screen, and come down of it if you left the default settings).

However, you'll find that there's no going back, once you open the second menu you can't get back to the previous one. This can be fixed by doing the same thing to a button inside the second menu or by assigning the variable called "Previous Menu", you do this by selecting the second menu, drag and drop the first menu in the "Previous Menu" field.



Now whenever the second menu is active, you can call `Back()` function on "ZUIManager" and it will then open the previous menu of the current active menu which is the first one, you can call `Back()` by making it a listener to a button, same way we did with `OpenMenu()`, or you can use the default key for back which is Escape (you can disable this by unchecking "Esc is back" in "ZUIManager" in "General Settings" section).

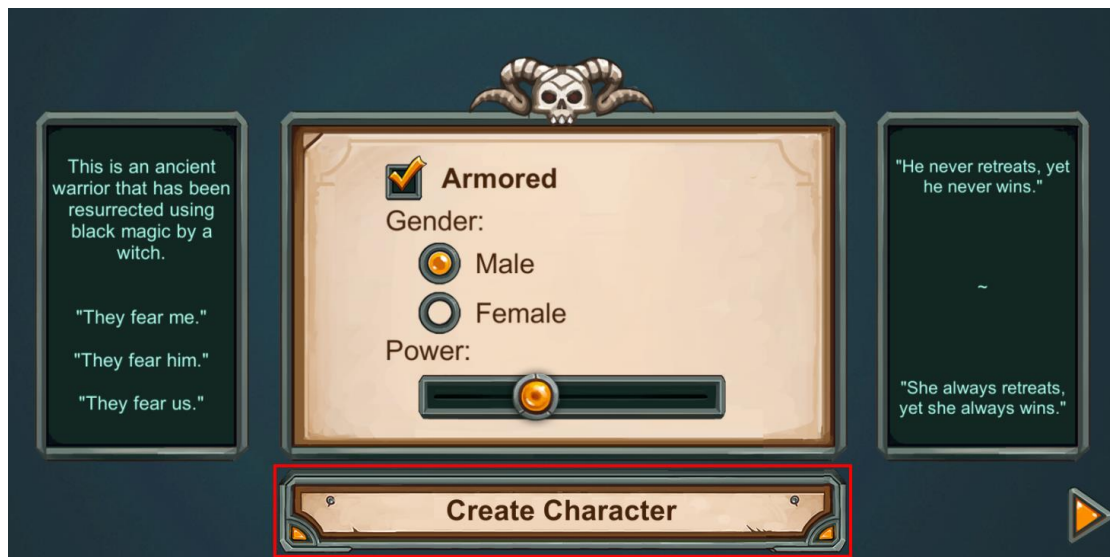
Now give it another test, click on the button to open the second menu, and then click "Esc" button on your keyboard to go back to the first one.

Note: `Back()` function will also close opened pop-ups or side-menus.

Multi-menu Animated Elements

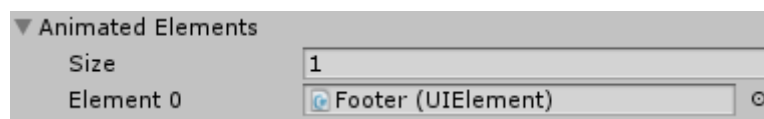
One of the smart features in this system is Multi-menu Elements.

Imagine you have a header or a footer that you want to stay on screen while the player is navigating between menus, and maybe you want it to get out of the screen only when he's inside the "Settings" menu for example.

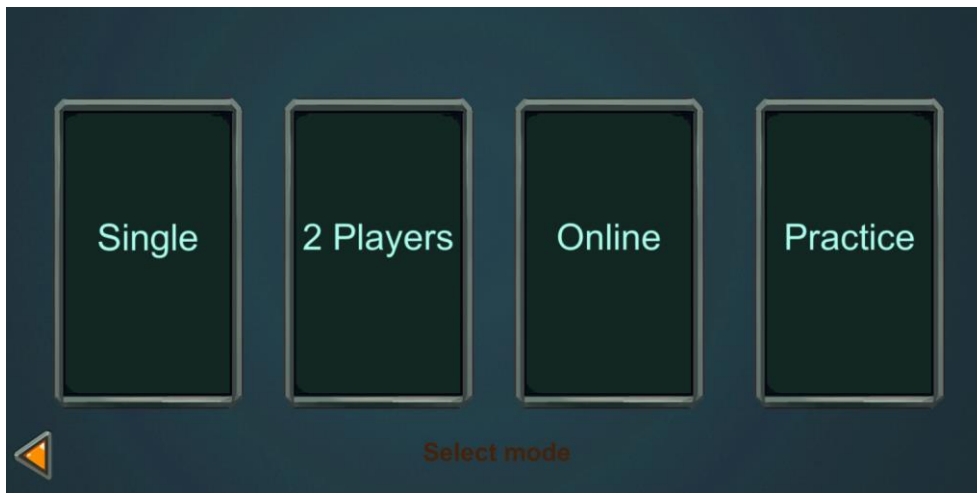


UI layout belongs to Ravenmore: <http://dycha.net>

Normally if you use "Animated Elements" list which is this one (inside the Menu script):

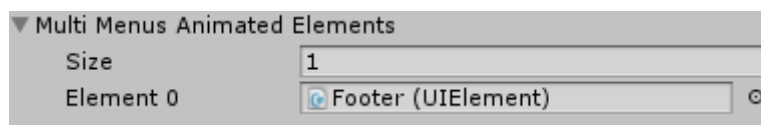


And you add the header/footer to it, it will simply disappear once you switch to another menu.

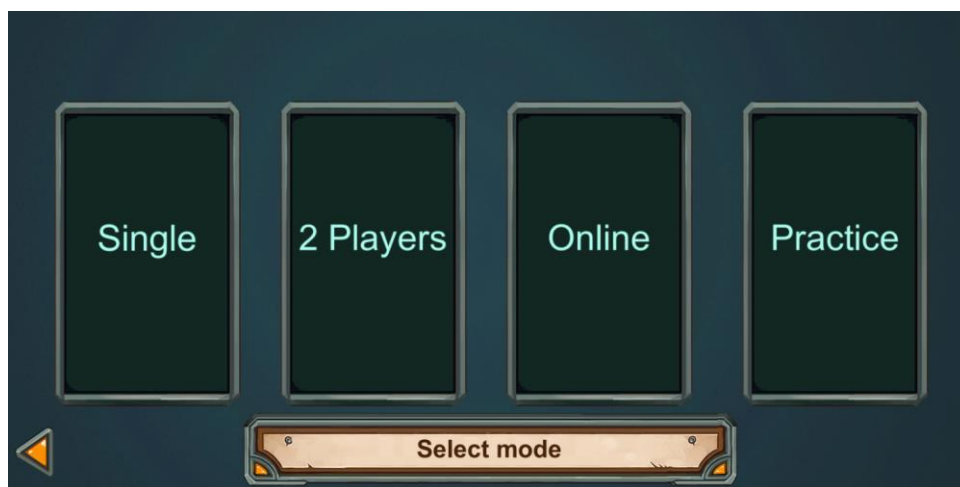


Note: notice how the text that supposed to be on the footer is put inside the normal "Animated Elements" list so that each menu shows its own name, because they only share the footer's background not the text on it.

however, there's a fix for that.



So instead of adding the footer to "Animated Elements" list, you add it to "Multi Menus Animated Elements", and you add the same footer for every single menu it should be visible inside, and of course don't add it to the "Settings" (or whatever) menu you don't want it to be seen there.

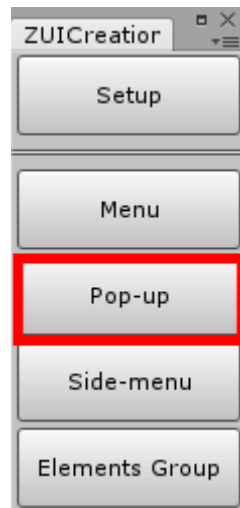


Note: you probably want multi-menu elements to not be children of any menu in the hierarchy so they don't disappear when its deactivated.

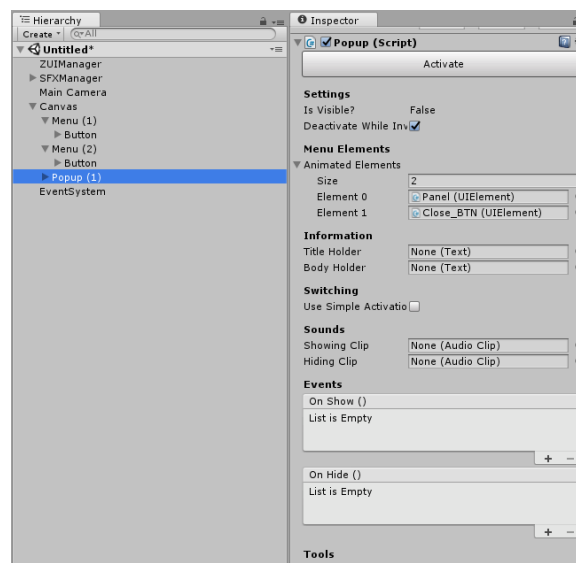
Creating Pop-ups

Pop-ups are another holder of UI elements, just like menus, however, it does not conflict with menus, you can have a menu opened, and a visible pop-up as well, but you can't have more than one pop-up visible at a time, requesting a pop-up to be opened will close the current opened one if there's any.

Pop-ups are similar to the menu when it comes to creation process, you either create them by going to the menu bar, *Tools>ZUI >Creation Window...* open the creation window and click on "Pop-up",

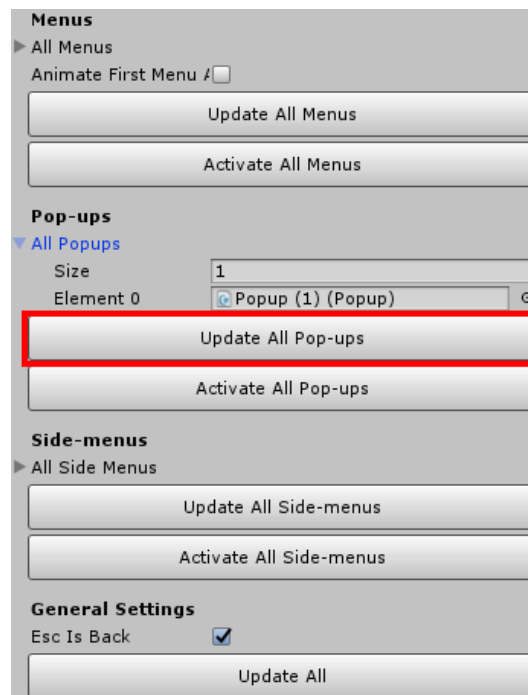


Or by creating the holder Game Object, and add "Popup" script to it.



And then you can create "UIElements" as children to the pop-up (refer to "[Creating Animated Elements](#)" section), and tell the pop-up about them by clicking "Update Animated Elements" button in "Popup" component.

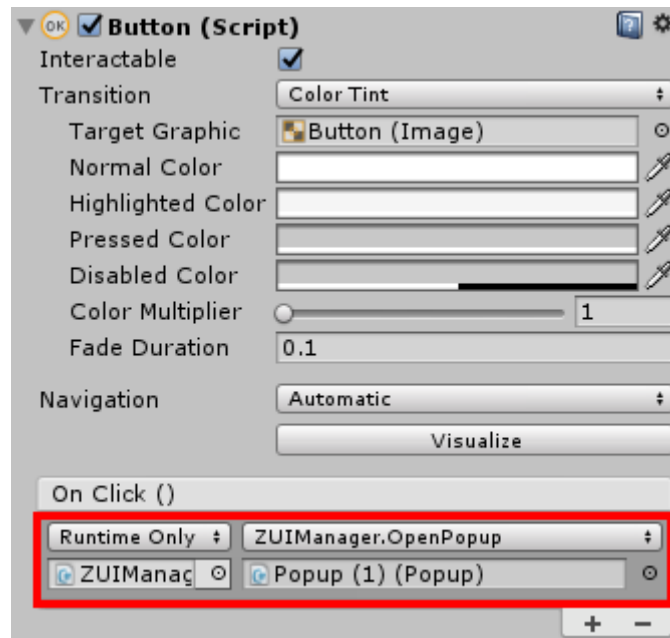
And the last step is to tell "ZUIManager" about the pop-up by clicking "Update All Pop-ups" (only if "Auto Find Holders" is disabled).



More details about pop-ups at "[Pop-ups In Details](#)" section.

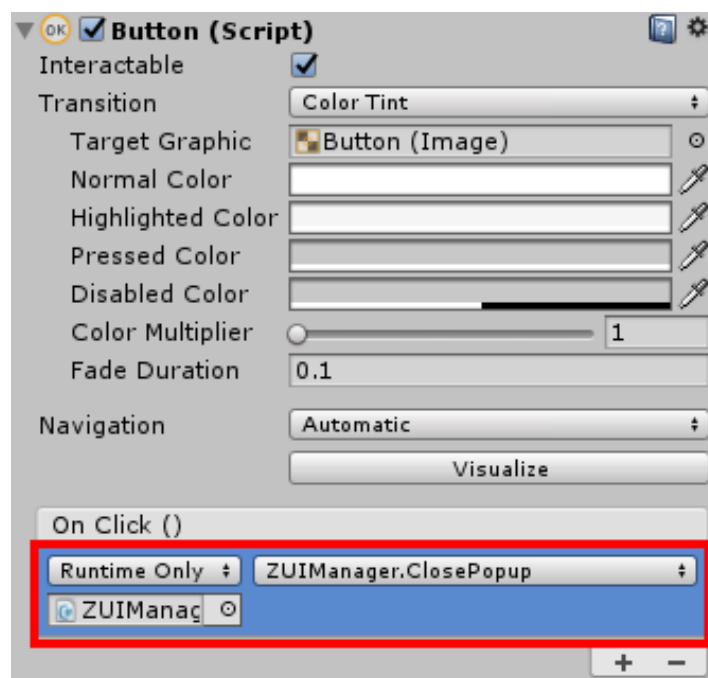
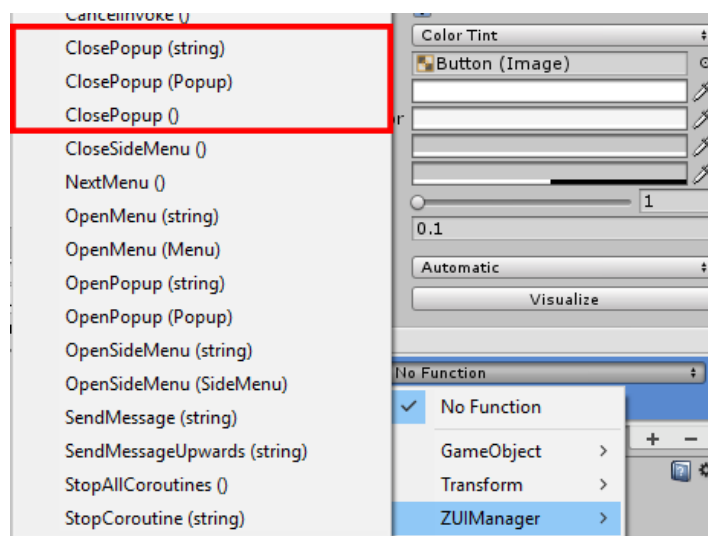
Using Pop-ups

Just like the menus, you have to call a function in "ZUIManager" that opens the popup, this function is called `OpenPop()`, and it also has two variations, one that takes the name of the menu as a parameter and the other takes the reference, We will, again, use the second one for this example. You can find it under the Game Object `ZUIManager>OpenPopup`, so create a button in one of your menus, and then add this function as a listener to it, and send the pop-up as a parameter.



The pop-up should open once you click the button, now we need to close it.

The function used to close pop-ups is `ClosePopup()`, you'll find it under the Game Object *ZUIManager*>*ClosePopup*, it has three variations, the first one takes the name of the popup as a parameter (closes it if only if its opened), second one takes a reference (closes it if only if its opened), and the last one closes the current opened popup (since only one would be active at a time). We'll go with the last one for this example, and probably this is the one you will use the most, so create a button inside your pop-up (and don't forget to add "UIElement" script to it and tell your pop-up about it) and then add this function as a listener to it.



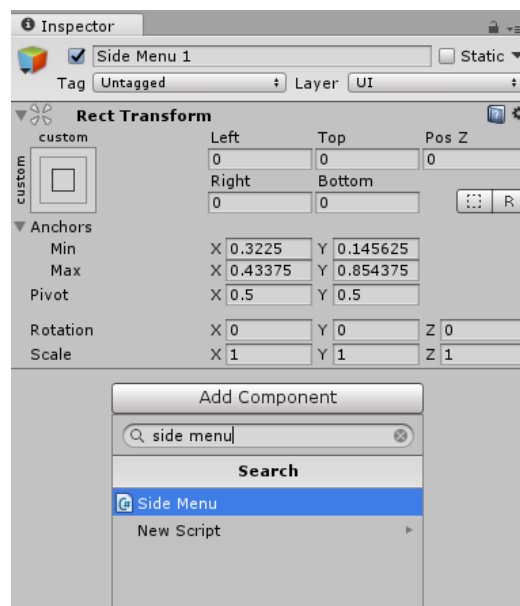
Creating Side-menus

Side-menus are yet another layer of menus, which means you can have only one side-menu active at a time, but with another menu and/or pop-up visible at the same time.

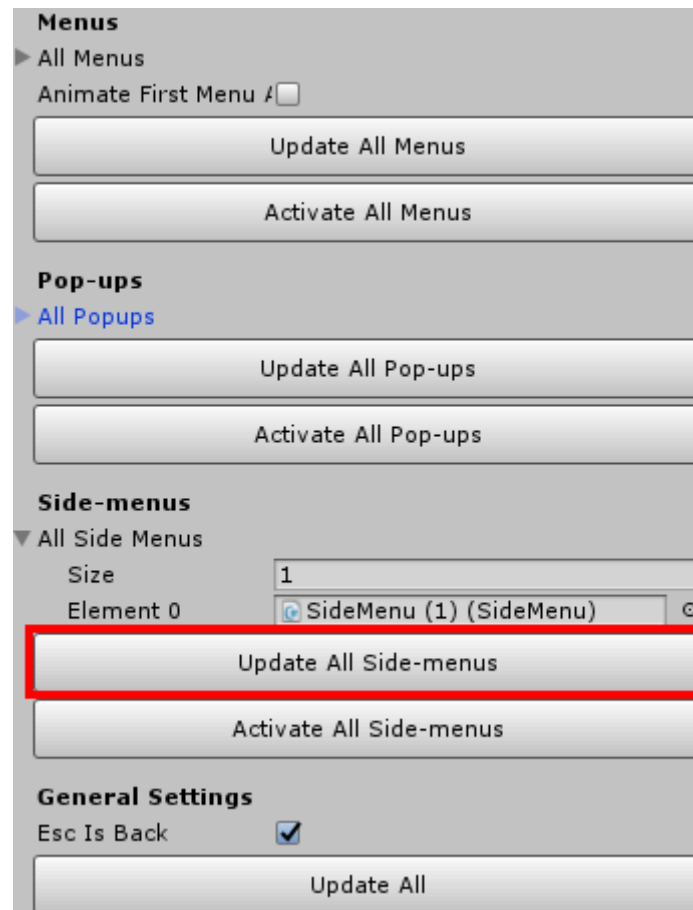
You can create one using "Creation Window".



Or you can create an empty Game Object as the holder of the side-menu, and then add "SideMenu" script to it.



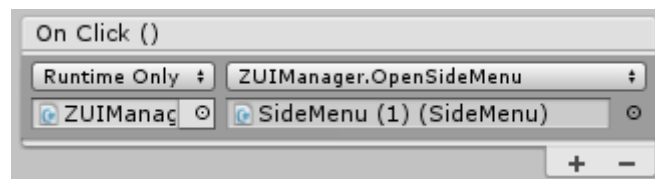
And of course create some UI Elements as children to it, tell the side menu about them, and then tell "ZUIManager" about the side menu (only if "Auto Find Holders" is disabled) by clicking "Update All Menus" button.



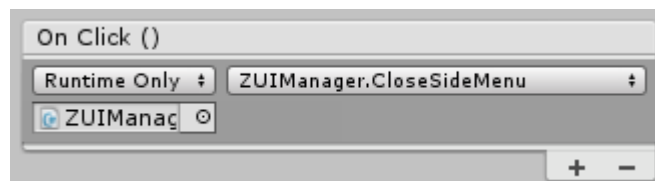
More details about side-menus at "[Side-menus In Details](#)" section.

Using Side-menus

The function to open side menus is `OpenSideMenu()`, which you will find under the Game Object *ZUIManager>OpenSideMenu*, two variations exist, one that takes the name of the side-menu as a parameter and the other takes a reference to it, and of course you need to make a button in one of your existing menus or popups to call this function just like before.



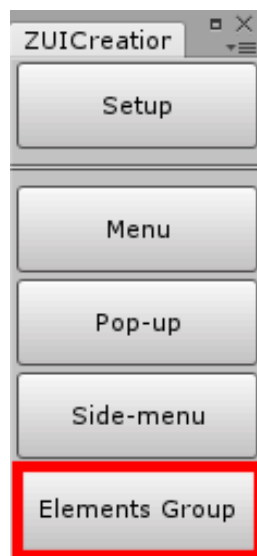
To close a side menu you have to call `CloseSideMenu()`, which is, again, under the Game Object *ZUIManager>CloseSideMenu*.



Creating UI Element Groups

UI Elements Group is a holder that carries a group of UI Elements, it does not have any constraints, which means you can have as many visible groups as you want, it cannot be directly added to an "Animated Elements" list of any controller (menu/pop-up/side-menu), and there's no manager that you need to inform about it.

As usual, creating a UI Elements Group can be done through the "Creation Window", by clicking "Elements Group"

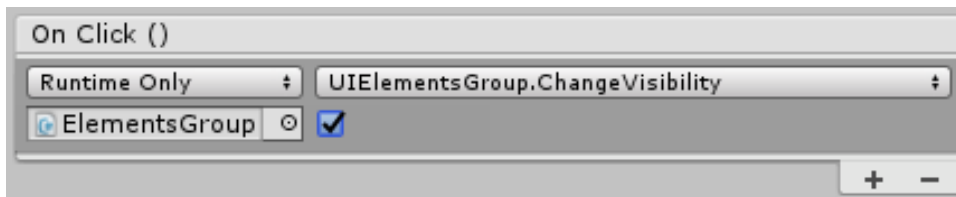


Or, of course, create an empty Game Object, and add "UIElementsGroup" script to it. There's no need to tell anyone about it, it's an independent holder.

More details about elements group at ["UI Elements Group In Details"](#) section.

Using UI Element Groups

Unlike other controllers, UI Element Groups are opened and closed by directly calling them instead of calling "ZUIManager", e.g. you can have a button, add "ChangeVisibility" function of the Group to its "OnClick" event listeners and send true as a parameter, this will make the entire group of UI Elements start playing showing animation once the button is clicked.



So basically, imagine you have 20 "UI Element" that you don't want to be controlled by one of the holders, but also you don't want to call a function on each one of them every time you wish to change their visibility, so you create a "UI Elements Group", put all of the 20 elements under it, add them to its list, and whenever you wish to change the 20 elements' visibility, do it to "UI Element Group".

To know more about the functions you can call, refer to ["UI Elements Group In Details"](#) section.

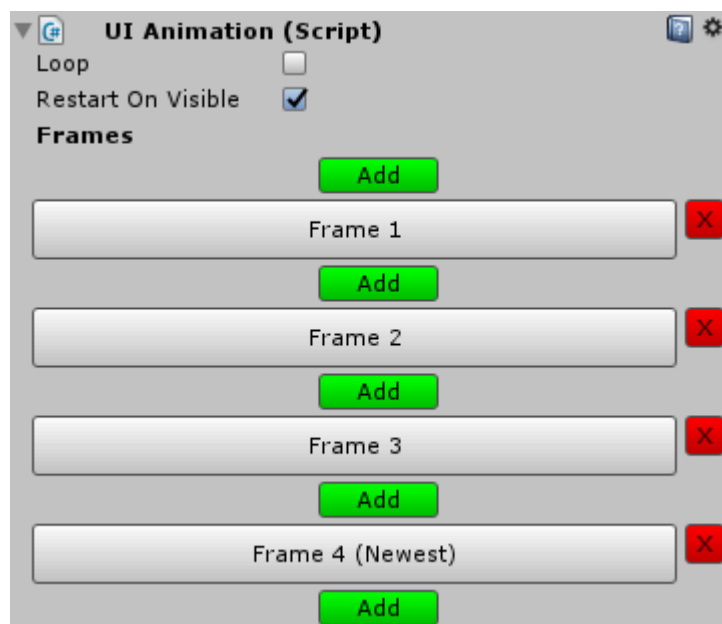
UI Animation

"UI Animation" is another useful feature, where it allows you to have huge control over element animations.

Normally when you're using "UI Element" component you only have control over one move of the object, for when its coming in, or going out of the screen.

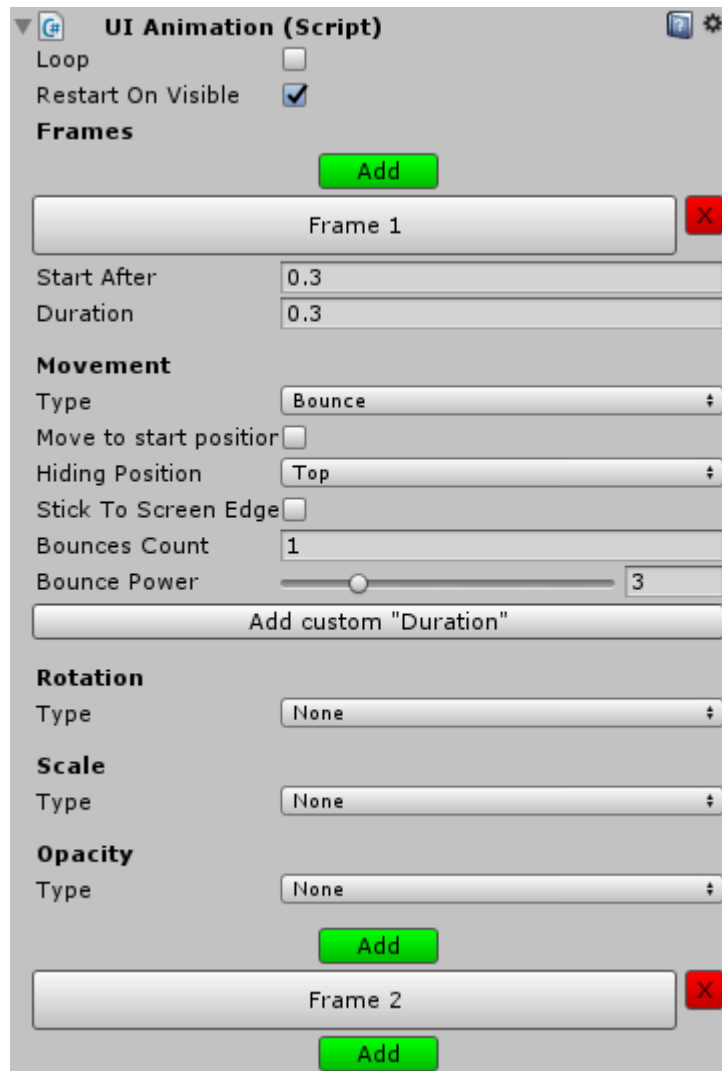
However, "UI Animation" is more flexible.

As you can see in the image below you can create the number of frames you want.



Note: "UI Animation" requires the object to have "UI Element" component.

Each one of these frames have almost the same properties presented in "UI Element" component.



Which means that you can, for example, have an image come into the screen, rotate 90 degrees, scale up 1 unit, fade out, fade in, scale down 1 unit, rotate -90 degrees, then get out of the screen again, and repeat.

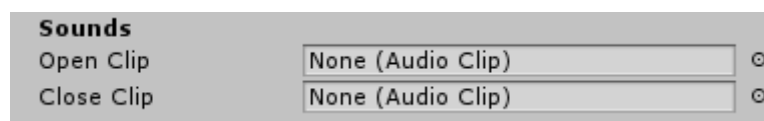
Basically have full control over the animation.

More details about UI Animation at "[UIAnimation In Details](#)" section.

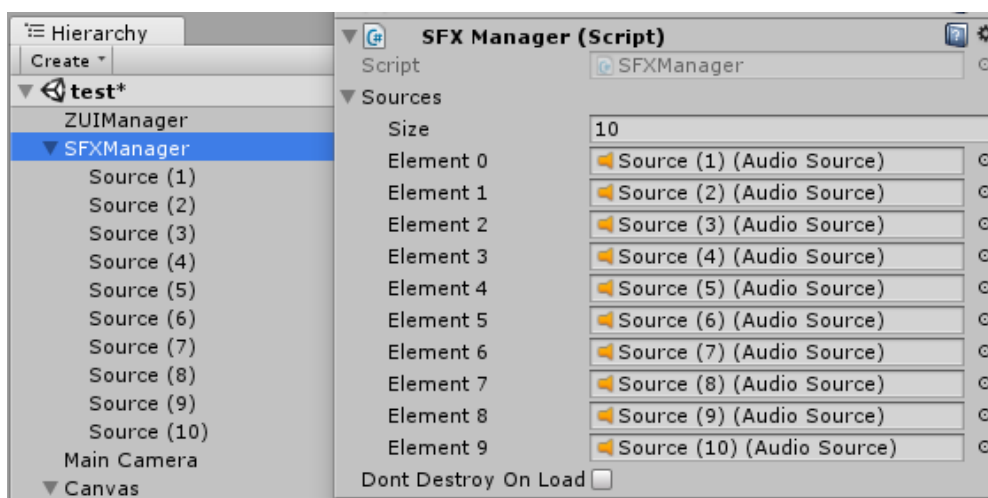
Using Sound Effects

Sound effects are essential to every game interface, that's why they are handled in the system.

Each Menu, Pop-up, Side Menu, UI Elements Group and UI Element have 2 fields that can contain an audio clip, one for when it shows up, and one for when it hides.



The audio clips are then sent to "SFXManager" to be played, which is one of the managers that will be created once you click "Setup" button in "Creation Window".



Notice that the "SFXManager" contains a list of 10 audio sources.

Each time "SFXManager" is told to play a certain audio clip, it plays it on one of the audio sources, and once another clip is requested to play it plays it on the next audio source, this to make sure that no clips would cancel each other. You can increase the number of audio sources if you have more than 10 clips to be played at the same time (hopefully you don't).

More details about "SFXManager" at "[SFXManager In Details](#)" section.

Components In Details

Here, we break out each component in the system, explaining all of its parameter, stating when it should be used, and how.

ZUIManager

"ZUIManager" is an essential component, the system will not work without it, it must be added to every scene you plan on using the system in. You can create it through "Creation Window", *Tools>ZUI >Creation Window...* by clicking on "Setup" button.

Auto Find Holders:

(Enabling this will hide all the holder sections below (Menus, Side-menus and Pop-ups))

Automatically find all the holders at the start of the scene, instead of having to update them like in the sections below.

Note: this may cause lag spike at the first frame, so it's advised to disable it before releasing a build, it's only here to make the creation process easier.

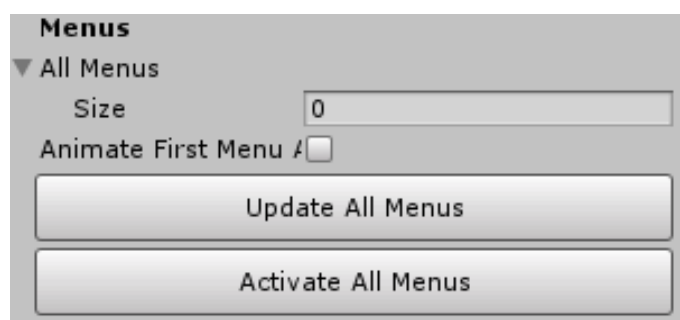


Menus

(Only available if "Auto Find Holders" is false)

All Menus:

A list of all menus in the scene, this MUST contain all menus, if a menu is not in this list it will not work correctly. Click "Update All Menus" button to automatically add all the menus in the scene.



Animate First Menu At Start:

Should the first menu play an animation once the scene starts (True)? Or should it start visible already (False)?

Update All Menus:

Finds all the menus in all the canvases in the scene and adds them to "All Menus" list.

Activate All Menus:

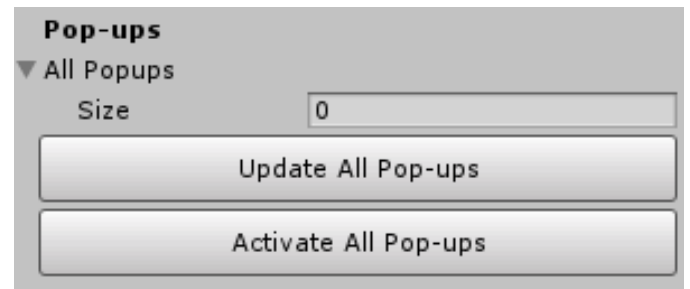
Activates all the Game Objects of all the menus in "All Menus" list.

Pop-ups

(Only available if "Auto Find Holders" is false)

All Popups:

A list of all pop-ups in the scene, this MUST contain all pop-ups, if a pop-up is not in this list it will not work correctly. Click "Update All Pop-ups" button to automatically add all the pop-up in the scene.



Update All Pop-ups:

Finds all pop-ups in all the canvases in the scene and adds them to "All Popups" list.

Activate All Pop-ups:

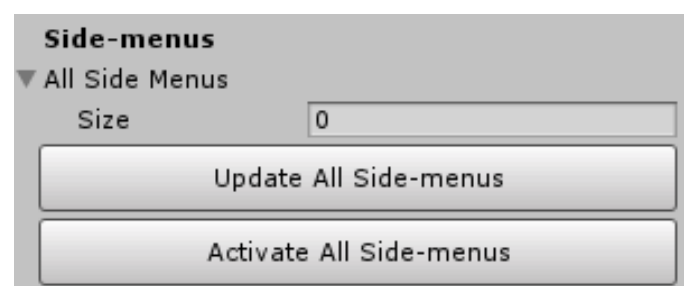
Activates all the Game Objects of all the pop-ups in "All Popups" list.

Side-menus

(Only available if "Auto Find Holders" is false)

All Side Menu:

A list of all side-menus in the scene, this MUST contain all side-menus, if a side-menu is not in this list it will not work correctly. Click "Update All Side-menus" button to automatically add all side-menus in the scene.



Update All Side-menus:

Finds all side-menus in all the canvases in the scene and adds them to "All Side Menus" list.

Activate All Side-menus:

Activates all the Game Objects of all the side-menus in "All Side Menus" list.

General Settings

General Settings

Animate First Menu At Start

Esc Is Back

Animate First Menu At Start:

Should the first menu play an animation once the scene starts (True)? Or should it start visible already (False)?

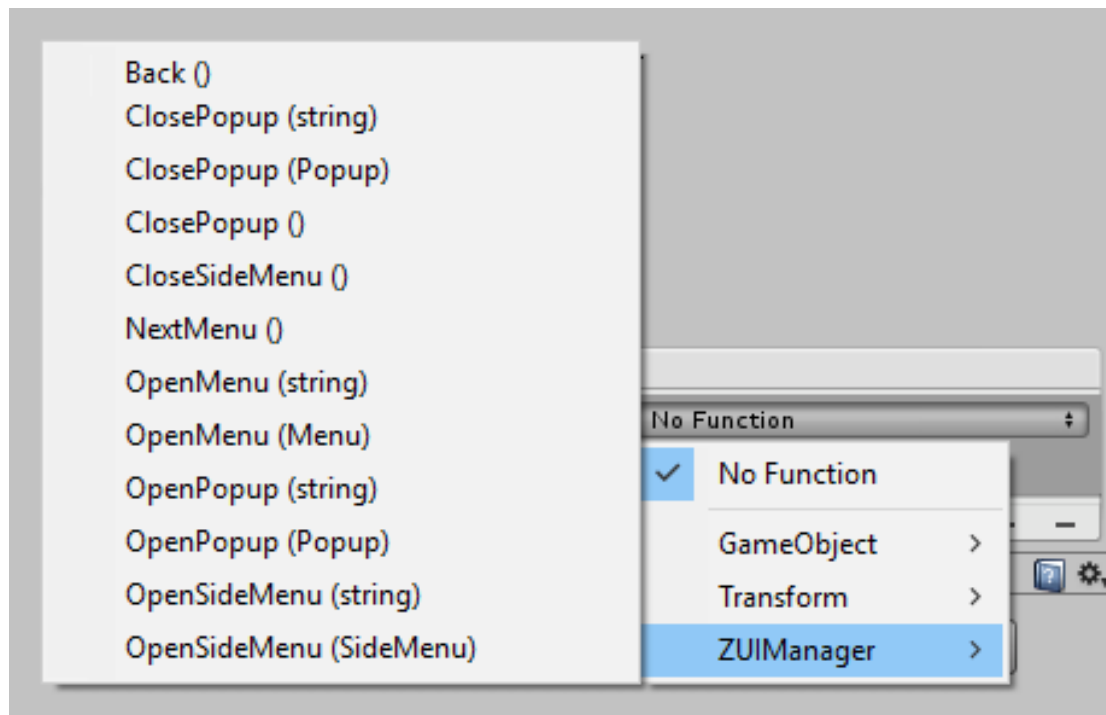
Esc Is Back:

Should "Cancel" button (Esc key/mobile back button by default, you can edit it via Edit>Project Settings>Inputs) on keyboards and mobile device act as a "Back" button?

Update All:

Finds all menus/pop-ups/side-menus in all the canvases in the scene and adds them to their related lists.

Actions (Public Functions)



These are the functions you can fire using a button, by scripting or other events.

OpenMenu(...)

OpenMenu(string): Opens a menu by name (the name of the Game Object holding "Menu" script).

OpenMenu(Menu): Opens a menu by reference (the Game Object holding "Menu" script).

OpenPopup(...)

OpenPopup(string): Opens a pop-up by name (the name of the Game Object holding "Popup" script).

OpenPopup(Popup): Opens a pop-up by reference (the Game Object holding "Popup" script).

OpenPopup(string, string, string) — *scripting only*: Opens a pop-up by name and apply title and body text.

OpenPopup(Popup, string, string) — *scripting only*: Opens a pop-up by reference and apply title and body text.

OpenSideMenu(...)

OpenSideMenu(string): Opens a side-menu by name (the name of the Game Object holding "SideMenu" script).

OpenSideMenu (SideMenu): Opens a side-menu by reference (the Game Object holding "SideMenu" script).

ClosePopup(...)

ClosePopup(): Close the currently active pop-up.

ClosePopup (string): Close the pop-up with the given name if it's currently active.

ClosePopup (Popup): Close the given pop-up if it's currently active.

CloseSideMenu()

Close the currently active side-menu.

Back()

If there's an active pop-up, close it. Or if there's an active side-menu, close it.

If no pop-up or side-menu are active, then go back to current menu's "Previous Menu" if found.

Next()

Go to current menu's "Next Menu" if found.

UIElement

UI Elements, are the visible elements (like a button or an image) that animate.

Giving them an order to show or hide will start an animation based on the parameters you give them.

Settings

Controlled By:

The holder controlling this UI Element. You must remove the controller of the UI Element if you wish to assign it to another controller.

Settings	
Controlled By:	NONE
Is Visible?	True
Menu Dependent	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Prewarm	<input checked="" type="checkbox"/>
Deactivate While Inv	<input type="checkbox"/>
Start After	0
Hide After	0
Duration	0.3

Is Visible:

Is this UI Element currently visible on the screen?

Menu Dependent:

Should this element be controlled by a holder? or it can switch on/off by external script/event instead.

Visible:

(Only available if "Menu Dependent" is false)

Should this element be visible when the game start?

Prewarm:

(Only available if "Menu Dependent" is false)

Should this element snap to its desired state at start (True)? or should it animate until it reaches it (False)?

Deactivate While Invisible:

Should this Game Object deactivate once it's invisible?

Start After:

The time in seconds before "Show" animation start playing.

Hide After:

The time in seconds before the "Hide" animation start playing.

Duration:

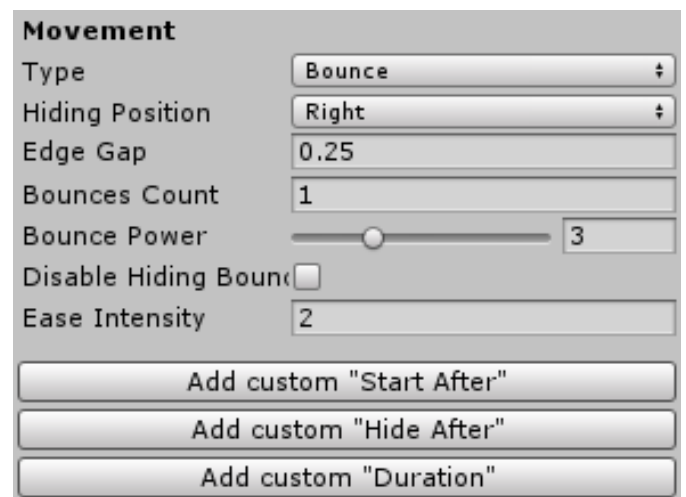
The duration of the animation in seconds (controls speed).

Movement/Rotation/Scale/Opacity

Each one of these sections have (almost) the same set of properties.

Type:

The type of the animation, whether its Ease In, Linear, Bounce or None if no animation is wanted in this section (if None is chosen no parameters will be shown).



The screenshot shows a 'Movement' settings panel with the following controls:

- Type: Bounce (dropdown menu)
- Hiding Position: Right (dropdown menu)
- Edge Gap: 0.25 (text input)
- Bounces Count: 1 (text input)
- Bounce Power: 3 (slider)
- Disable Hiding Bounce: (checkbox)
- Ease Intensity: 2 (text input)

At the bottom of the panel are three buttons:

- Add custom "Start After"
- Add custom "Hide After"
- Add custom "Duration"

Hiding Position:

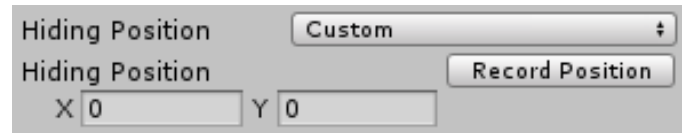
(Only available in "Movement" section)

The place out of the screen (or custom position) to move to when this element should be invisible. (Note that there's no option for start position because the position that the element is placed at when the game start is counted as the start position)

Custom:

If you chose "Custom" as Hiding

Position, the following parameters will be enabled:



Hiding Position (Vector 2): X and Y indicate the place on the screen this element should move to when it hides.

For X: 0 is the left side of the screen while 1 is the right side.

For Y: 0 is the bottom side of the screen while 1 is the top side.

You can click on "Record Position" button, it will turn red, and then you can move the element to the position you want, and if you click on the button again it will record this position as the hiding position and move the element back to its original position before clicking the button.

Edge Gap:

The gap left for safety between the element and the edge of the screen while it's hiding (in percentage of element's width or height) make this 0 if you want the element to stick to the edge of the screen.

Bounces Count:

(Only available if "Type" is "Bounce")

The count of bounces the element should make (if this number is increased it will increase the movement speed, its advised to keep it low).

Bounce Power:

(Only available if "Type" is "Bounce")

The power of the bounces.

Disable Hiding Bounce:

(Only available if "Type" is "Bounce")

Should this object not bounce when it's hiding?

Ease Intensity:

(Available if "Type" is "Ease In" or "Bounce")

The intensity of the ease animation?

Custom Properties:

Here you can add custom property to override the property in "Settings" section, but only for this specific section you added it to.

Simple Activate/Deactivate

Simple Activate/Deactivate

Use Activation Contr

Use Activation Control

Should this Game Object simply activate/deactivate when its visible/invisible instead of playing an animation?

Sounds

You MUST have "SFXManager" in the scene for sounds to work.

Showing Clip

The sound to play once the element is requested to be visible.

Sounds

Showing Clip

None (Audio Clip)



Hiding Clip

None (Audio Clip)



Hiding Clip

The sound to play once the element is requested to be invisible.

Events

On Show

Fires the moment this element is requested to be visible.

Events

On Show ()

List is Empty



On Hide

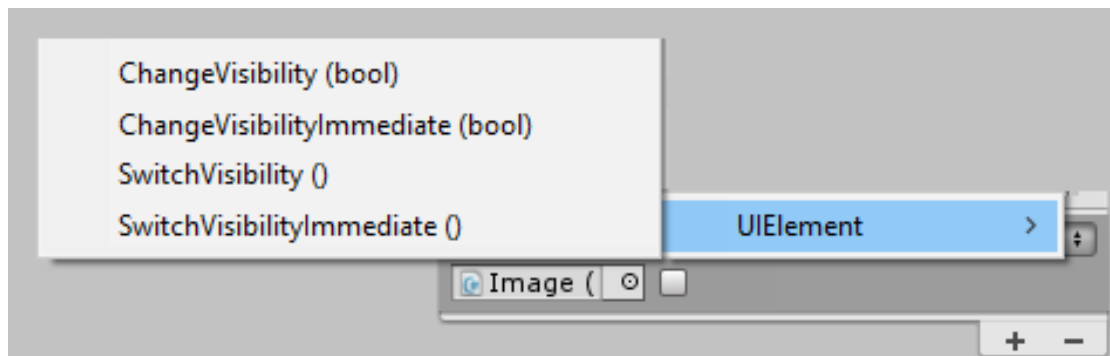
Fires the moment this element is requested to be invisible.

On Hide ()

List is Empty



Actions (Public Functions)



These are the functions you can fire using a button, by scripting or other events.

ChangeVisibility(bool)

Change the visibility of this element by playing the desired animation.

Takes a Boolean parameter (True: show, False: hide)

ChangeVisibilityImmediate(bool)

Change the visibility of this element instantly without playing animation.

Takes a Boolean parameter (True: show, False: hide)

SwitchVisibility()

Switch the visibility of the element by playing the desired animation, e.g. if the element is currently visible then it will be invisible once you call this function.

SwitchVisibilityImmediate()

Switch the visibility of this element instantly without playing animation.

Menu

Just like mentioned before, a menu holds and controls a number of UI elements, by giving them orders to animate whenever the menu is opening or closing (changing visibility). Only one menu can be visible at a time, whenever another menu is requested to be opened, the current one closes.

Activate:

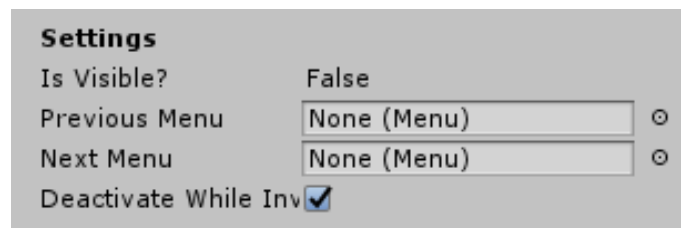
Activate this Game Object and deactivate all other menus' Game Objects.



Settings

Is Visible:

Is this menu visible?



Previous Menu:

The menu to open when this menu is opened and "Back()" functions is called on "ZUIManager".

Next Menu:

The menu to open when this menu is opened and "Next()" functions is called on "ZUIManager".

Deactivate While Invisible:

Should this Game Object deactivate once it's invisible?

Menu Elements

Animated Elements:

A list of all the elements only this menu controls.

Multi Menu Animated Elements:

A list of the elements this menu share with other menus, for further explanation, refer to "[Multi-menu Animated Elements](#)" section.



Switching

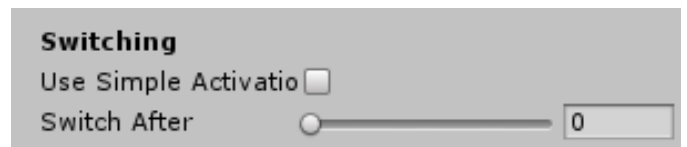
Use Simple Activation:

Simply activate/deactivate this Game Object instead of animating it's elements.

Switch After:

(Ranges from 0 ~ 1)

The percentage of animation time to pass before opening another menu. Increase this if the animation take some time to finish and you don't want it to conflict with the next menu's animation.



Sounds

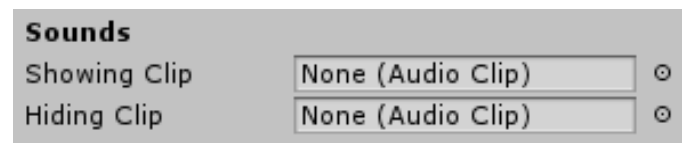
You MUST have "SFXManager" in the scene for sounds to work.

Showing Clip

The sound to play once the menu is requested to be visible.

Hiding Clip

The sound to play once the menu is requested to be invisible.



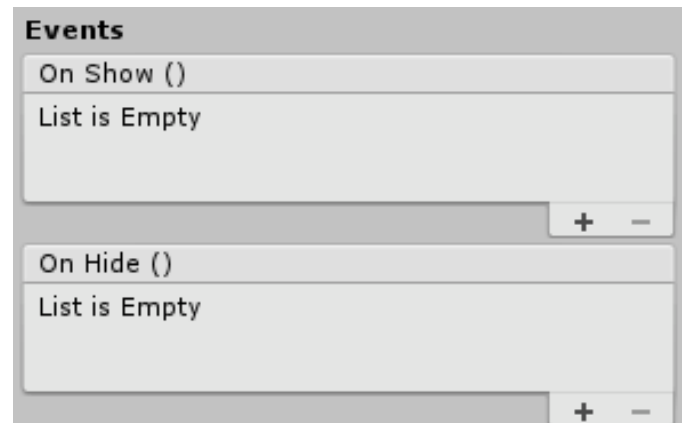
Events

On Show

Fires the moment this menu is requested to be visible.

On Hide

Fires the moment this menu is requested to be invisible.



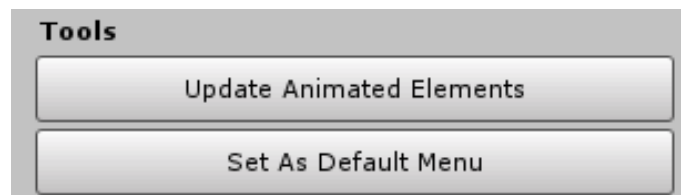
Tools

Update Animated Elements:

Find all the animated elements under this menu (children) and add them to "Animated Elements" list.

Set As Default Menu:

Make this menu the first one to open when the game start.



Actions (Public Functions)

DO NOT use menu's visibility functions directly unless you plan to make changes to the system. These actions are meant to be called by "ZUIManager" only, check "Actions" section at "ZUIManager In Details".

Popup

Pop-ups are holders of UI elements, but they have their own layer, meaning that they do not conflict with other types of holders. You can have only one pop-up visible at a time, requesting a pop-up to be opened will close the current opened one if there's any.

Activate:

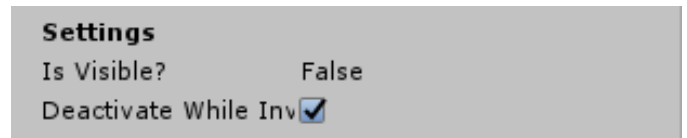
Activate this Game Object and deactivate all other popups' Game Objects.



Settings

Is Visible:

Is this pop-up visible?



Deactivate While Invisible:

Should this Game Object deactivate once it's invisible?

Menu Elements

Animated Elements:

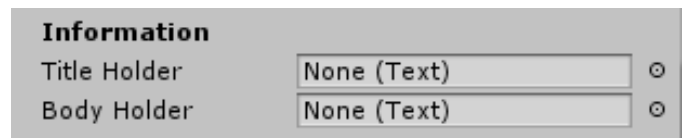
A list of all the elements this pop-up controls.



Information (Optional)

Title Holder:

The "Text" that holds the title of the pop-up.



Body Holder:

The "Text" that holds the body of the pop-up.

Switching

Use Simple Activation:

Simply activate/deactivate this Game Object instead of animating it's elements.

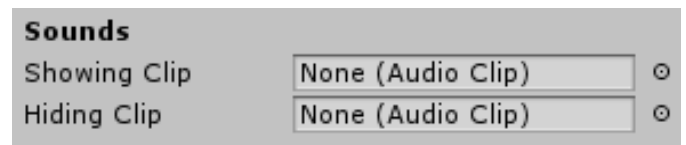


Sounds

You **MUST** have "SFXManager" in the scene for sounds to work.

Showing Clip

The sound to play once the pop-up is requested to be visible.



Hiding Clip

The sound to play once the pop-up is requested to be invisible.

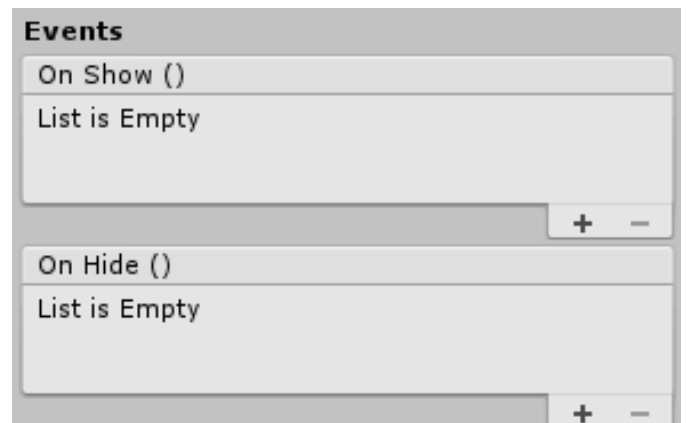
Events

On Show

Fires the moment this pop-up is requested to be visible.

On Hide

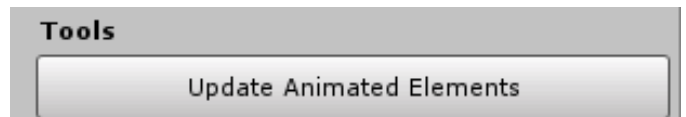
Fires the moment this pop-up is requested to be invisible.



Tools

Update Animated Elements:

Find all the animated elements under this pop-up (children) and add them to "Animated Elements" list.



Actions (Public Functions)

DO NOT use pop-up's visibility functions directly unless you plan to make changes to the system. These actions are meant to be called by "ZUIManager" only, check "Actions" section at "ZUIManager In Details".

.

SideMenu

Side-menus are holders of UI elements, having its own layer, meaning that it does not conflict with other types of holders. Only one side-menu can be visible at a time, requesting another side-menu to be opened will close the current opened one if there's any.

Activate:

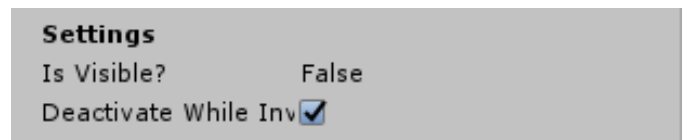
Activate this Game Object and deactivate all other side-menus' Game Objects.



Settings

Is Visible:

Is this side-menu visible?



Deactivate While Invisible:

Should this Game Object deactivate once it's invisible?

Menu Elements

Animated Elements:

A list of all the elements this side-menu controls.



Switching

Use Simple Activation:

Simply activate/deactivate this Game Object instead of animating it's elements.

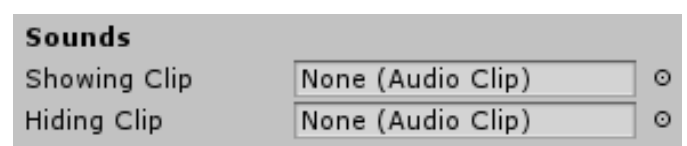


Sounds

You MUST have "SFXManager" in the scene for sounds to work.

Showing Clip

The sound to play once the side-menu is requested to be visible.



Hiding Clip

The sound to play once the side-menu is requested to be invisible.

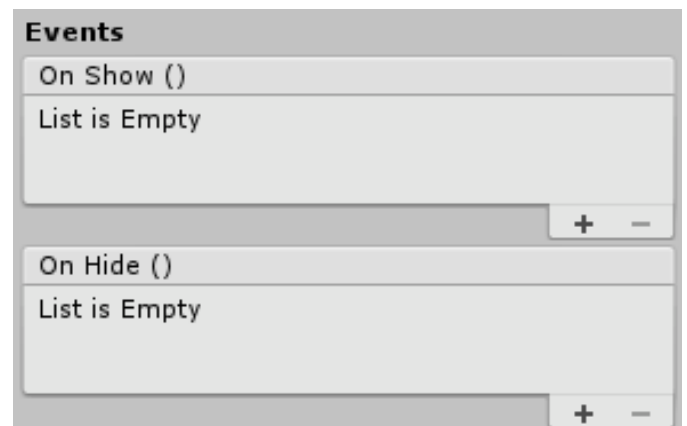
Events

On Show

Fires the moment this side-menu is requested to be visible.

On Hide

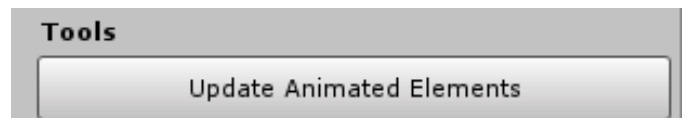
Fires the moment this side-menu is requested to be invisible.



Tools

Update Animated Elements:

Find all the animated elements under this side-menu (children) and add them to "Animated Elements" list.



Actions (Public Functions)

DO NOT use side-menu's visibility functions directly unless you plan to make changes to the system. These actions are meant to be called by "ZUIManager" only, check "Actions" section at "ZUIManager In Details".

.

UIElementsGroup

UI Elements Group is a holder that carries a group of UI Elements, it does not have any constraints, which means you can have as many visible groups as you want, it cannot be directly added to an "Animated Elements" list of any controller (menu/pop-up/side-menu), and there's no manager that you need to inform about it.

Settings

Is Visible:

Is this elements group visible?



Prewarm:

Should the element under this group snap to its desired state at start (True)? or should it animate until it reaches it (False)?

Deactivate While Invisible:

Should this Game Object deactivate once it's invisible?

Menu Elements

Animated Elements:

A list of all the elements this elements group controls.



Switching

Use Simple Activation:

Simply enable/disable this Game Object instead of animating it's elements.

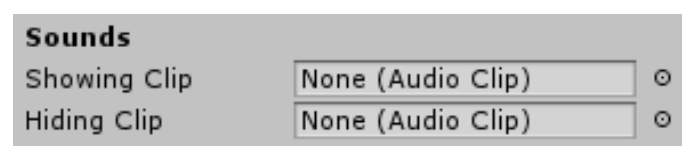


Sounds

You MUST have "SFXManager" in the scene for sounds to work.

Showing Clip

The sound to play once the elements group is requested to be visible.



Hiding Clip

The sound to play once the elements group is requested to be invisible.

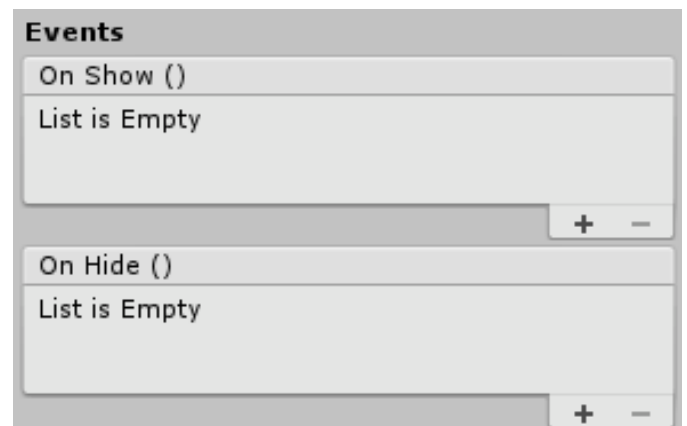
Events

On Show

Fires the moment this elements group is requested to be visible.

On Hide

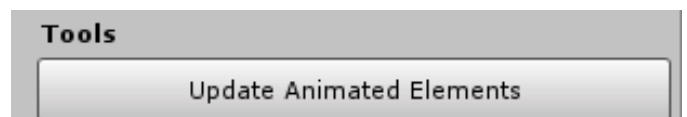
Fires the moment this elements group is requested to be invisible.



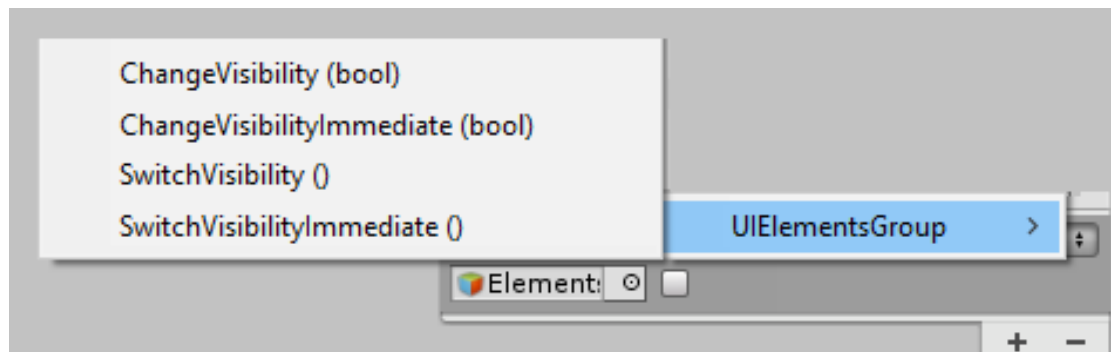
Tools

Update Animated Elements:

Find all the animated elements under this elements group (children) and add them to "Animated Elements" list.



Actions (Public Functions)



These are the functions you can fire using a button, by scripting or other events.

ChangeVisibility(bool)

Change the visibility of this group by playing the desired animation.

Takes a Boolean parameter (True: show, False: hide)

ChangeVisibilityImmediate(bool)

Change the visibility of this group instantly without playing animation.

Takes a Boolean parameter (True: show, False: hide)

SwitchVisibility()

Switch the visibility of the group by playing the desired animation, e.g. if the element is currently visible then it will be invisible once you call this function.

SwitchVisibilityImmediate()


Switch the visibility of this group instantly without playing animation.

UIAnimation

"UIAnimation" allows you to animate UI Elements freely, not just when they're switching visibility, and it requires the Game Object to have "UIElement" component, refer to "[UI Animation](#)" section for further explanation. The animation start once the object has switched to "Visible" state.

Loop:

Should this animation loop?



Loop
Restart On Visible

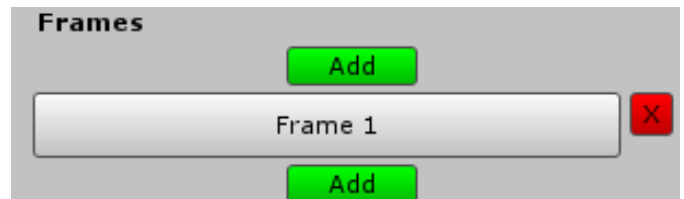
Restart On Visible:

Should this animation restart each time the element is visible?

Frames

Add:

Add a new frame at the desired place.



Frames

Add

Frame 1

Add

Frame:

Same properties as "UIElement", refer to "[UIElement In Details](#)" section.

SFXManager

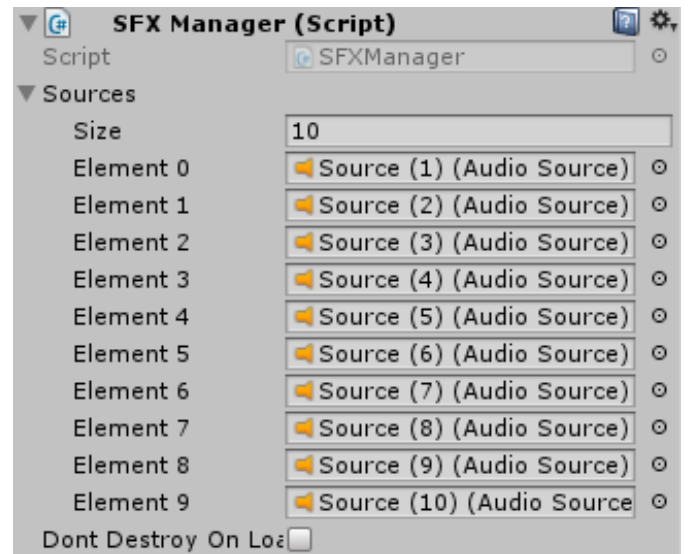
Add "SFXManager" to the scene you wish to have sound effects in.

Sources:

A list of audio sources to play the sounds on, by default there are 10 children to "SFXManager" you can change this count and update this list if you wish.

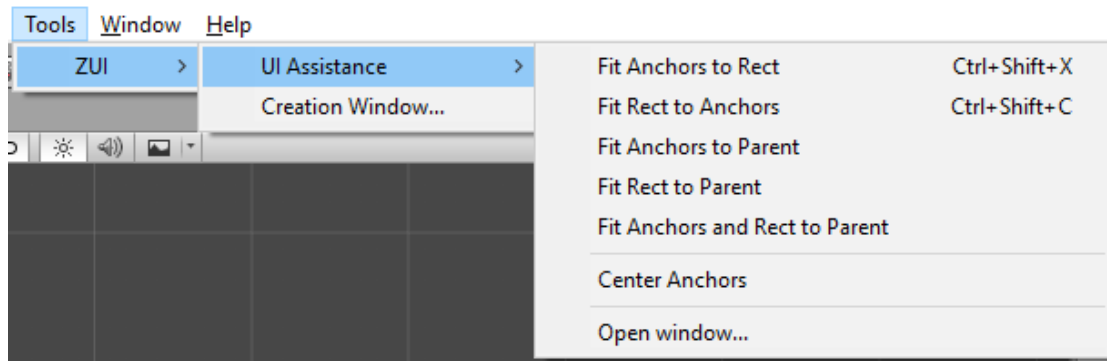
Don't Destroy On Load:

Enable this if you wish to have this manager in all of your scenes. This will protect it from destroying when a new scene is called.

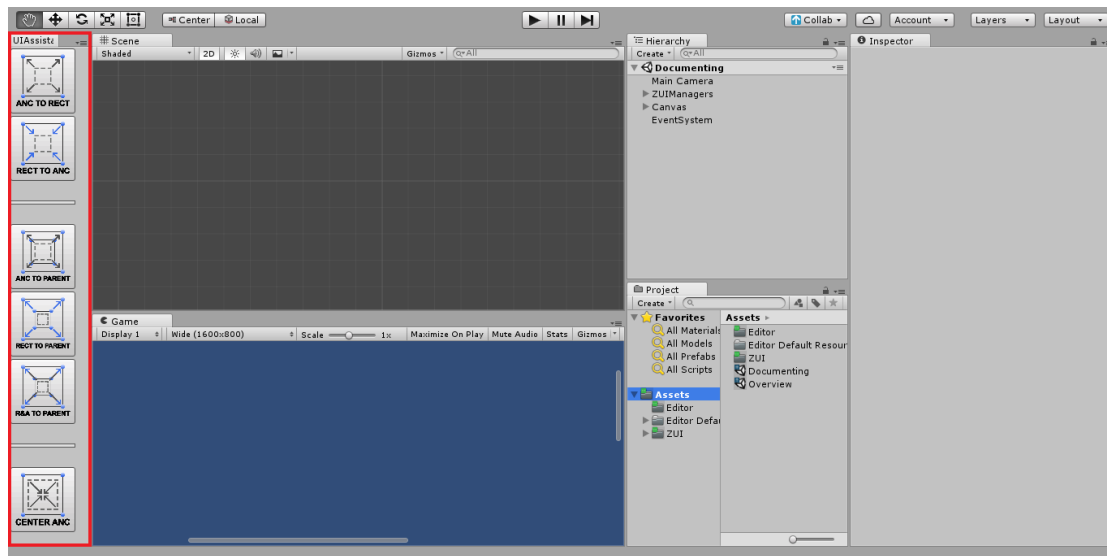


UI Assistance Tools

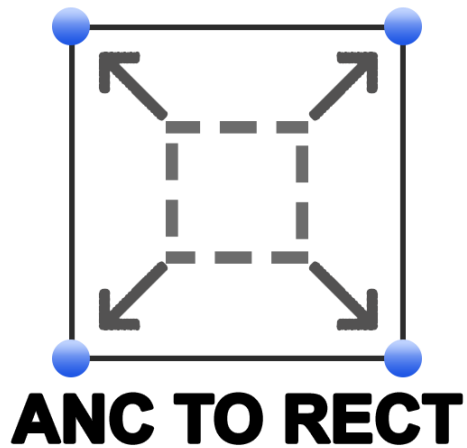
ZUI comes with a set of tools to help you create your UI, you can find those tools at the menu bar under *Tools>ZUI>UI Assistance*.



You can also open a menu that contains all these tools if you plan to use them often, by clicking on "Open window..." and dock it wherever you wish.

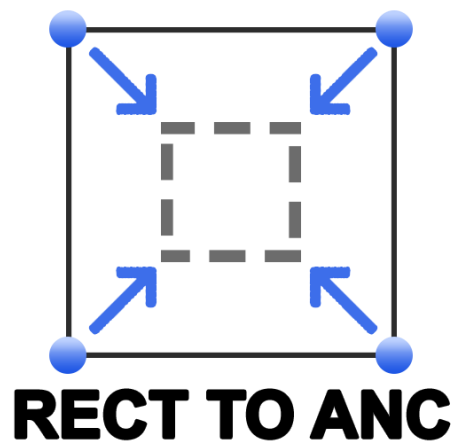


Fit Anchors to Rect



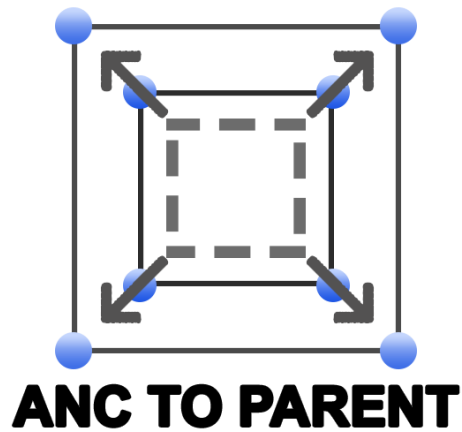
This one is my personal favorite, as it automatically fits the anchors to the rectangle of the UI element. (shortcut: Ctrl+Shift+X)

Fit Rect to Anchors



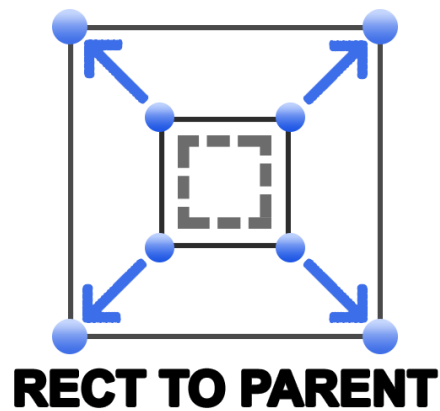
Opposite of the previous one, fits the rectangle of the UI element to the anchors. (shortcut: Ctrl+Shift+C)

Fit Anchors to Parent



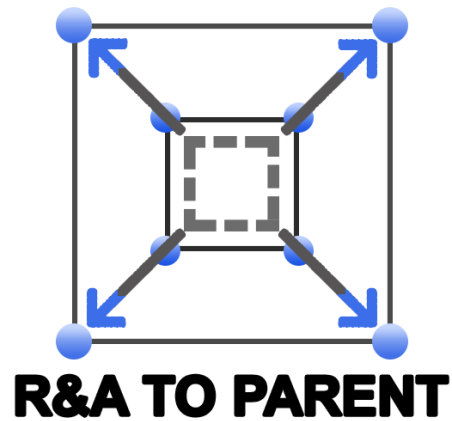
Fits the anchors of the UI element to the rectangle of its parent.

Fit Rect to Parent



Fits the rectangle of the UI element to the rectangle of its parent.

Fit Rect and Anchors to Parent



The last 2 tools combined, fits the rectangle and the anchors of the UI element to the rectangle of its parent, this is specifically useful when creating menu holders, background images, and whatever element you want to fit the entire screen (if its direct parent is the canvas) or his holder.

Tips & Tricks

Animating the Entire Menu as One Block

Sometimes you may want to simply fade the entire menu/pop-up/side-menu/elements-groups in or out, or move it out of the screen as one block. In this case it won't be efficient to apply "UI Element" component to each element inside the holder and animate its properties, instead, you can apply "UI Element" component to the holder itself, and do the animation you want there (and don't forget to tell the holder component about it by clicking "Update Animated Elements").

Note: to achieve the fading effect you need to add "Canvas Group" component to the holder before animating its opacity, so the component can then animate the alpha property.

Smooth Cartoonish Transitions

You can adjust the "Start After" and "Hide After" properties of the "UI Element", so the elements would show or hide one after the other, this adds smoothness and fluidity to the transitions, as presented in the demos in this package.

Persistent Back Button

Some game interfaces have a single back button on one of the corners of the screen that remains there all the time, so players can use it to open the previous menu.

You can have this by adding the back button to every menu's "Multi-menus Animated Elements" list, and add the Back() function (you'll find it in "ZUIManager") as a listener to the back button's OnClick event.

One Pop-up for Many Reasons

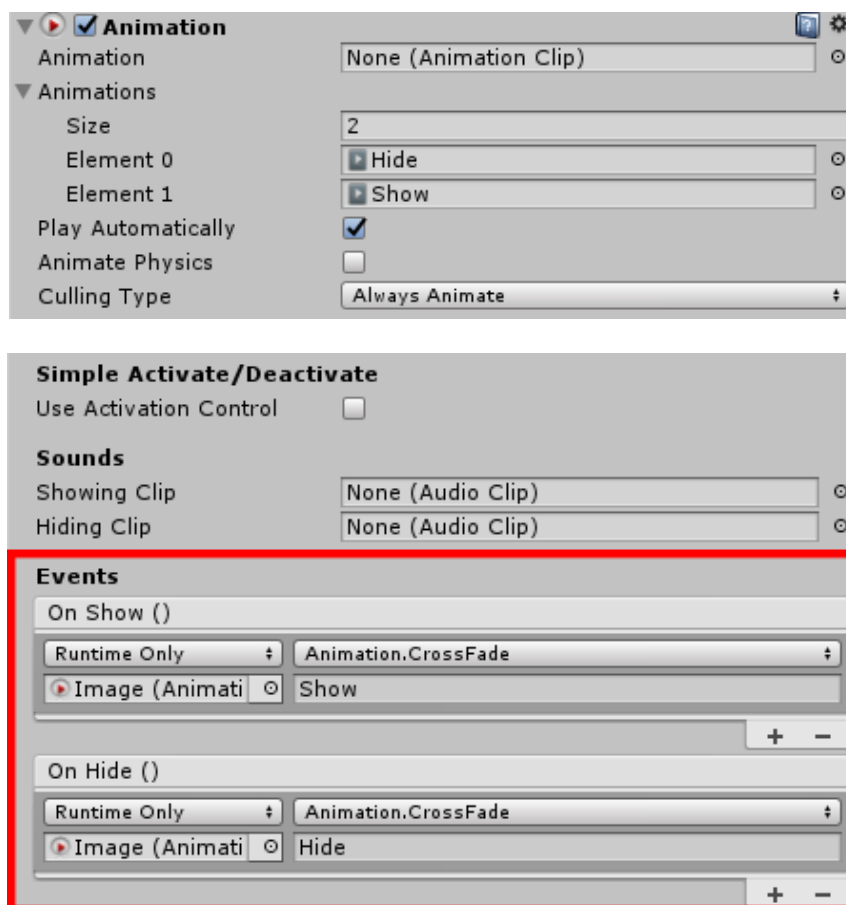
You can have only one pop-up that changes title and body text by opening it using the function "OpenPopup(Popup, string, string)" the second two strings in the parameters are title and body.

This will give you the ability to create a dynamic pop-up, e.g. a pop-up that tells the player "You have consumed all your energy" and the same pop-up tell him in another situation "You have refilled your energy successfully!".

But make sure you've updated the information section of the Pop-up!

Custom Animation Clips When Visibility Change

If you have more complicated animation you would like to play instead of the preset animations, you can create Animation Clips inside Unity, and use the UI Element's event properties to play them.



And of course, you can use Mechanim if you wish.

Goodbye! - Contact Me!

Lastly, it's been my pleasure walking you through this documentation, and I hope the system meets your expectations.

Do not hesitate contacting me through my:

Website: <http://ziadkhalid.com/>

E-mail: contact@ziadkhalid.com

Facebook page: <https://www.facebook.com/zIndie.GameDeveloper/>

YouTube channel: <https://www.youtube.com/channel/UCr14L6llbr4y--jJ9Fjkwpg>

Twitter: <https://twitter.com/ShinyZM>

License

You may use this package personally or commercially, **but you cannot resell or redistribute the package on any store or site.**