Arcadia Library - Documentation

Plugin version: 1.0

Unity version: 2017.3.0f3

Unity min version: 2017.1.0f3

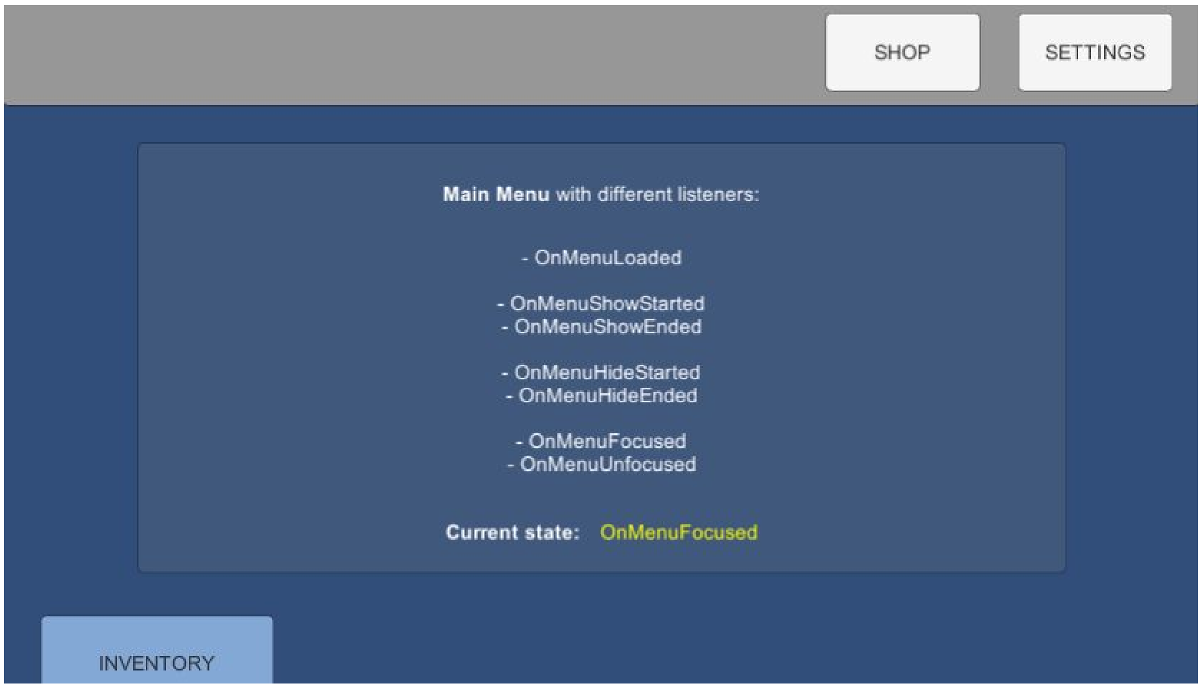
Support: info@gametroopers.net

Index

Introduction

Arcadia Library is a navigation solution for custom menus using Unity3D, allowing an easy flow between panels.

- This plugin allows the management of different menus with functions, storing history to back to the previous panels and dispatching events on each menu status, as: on menu loaded, show, hide, focused and unfocused.
- Supports Show and Hide functions.
- Menus are separated in ‘groups’, which means that you can initialize only the desired menus instead of maintain all of them on scene at the same time.
- ‘Layers’ concept. Only one menu can be shown simultaneously by layer, what it means that the current active menus on the same layer will be hidden to made easy the user flow.
- Also supports controller input, storing the last selected button/selectable on menus to make easy the navigation flow between panels.
- Default show and hide animations. In addition, animations can be fully customized by user.



Main Menu with different listeners:

- OnMenuLoaded

- OnMenuShowStarted
- OnMenuShowEnded

- OnMenuHideStarted
- OnMenuHideEnded

- OnMenuFocused
- OnMenuUnfocused

Current state: OnMenuFocused

Code features

- Based on two classes:
  - Menu: Each UI panel or menu inherits from this class.
  - MenuManager: Manages all the menus in the game, offering methods to initialize, show and hide them.
- All the menus can be loaded during game initialization or manually by group ID.
- Menus are indexed using the type of the classes that inherit from Menu.
- Each layer has an stack that stores all the opened menus. Closing a menu of a layer will automatically open the next menu on the stack, if any.
- The layer for each menu is defined on the inspector view.
- Buttons in the menus are loaded during the initialization and are disabled only during the hide animation of the menu.

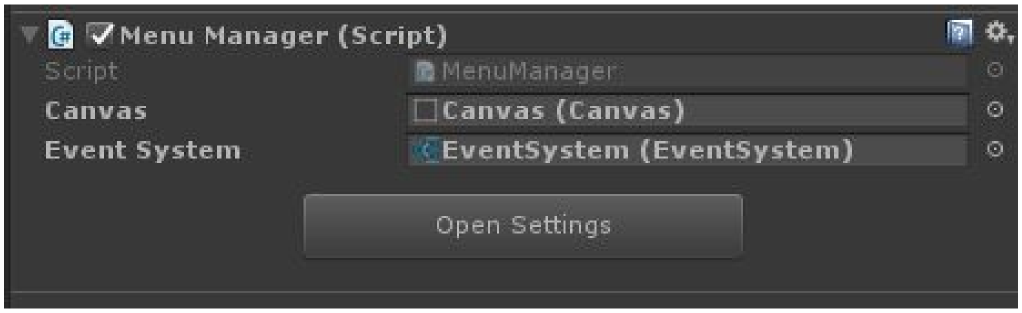
Setup

To start using the plugin, you should follow the next steps:

1. Create an empty gameObject on the scene and add it a MenuManager component. This gameObject won't be destroyed on load and may be referenced by your own data manager to access to the scripts methods (i.e: in a GameManager singleton).

2. Attach your scene canvas and event system into the MenuManager fields.

3. From the MenuManager, you can access to the UI Settings using the "Open Settings" button.

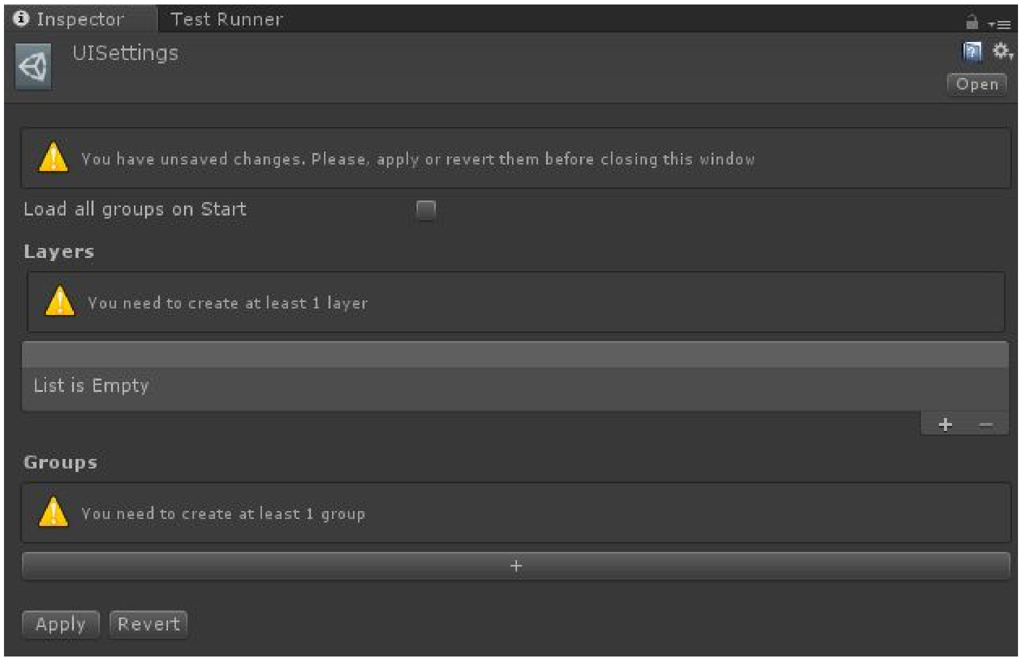


Menu Manager (Script)

Script: MenuManager

Canvas: Canvas (Canvas)

Event System: EventSystem (EventSystem)



You have unsaved changes. Please, apply or revert them before closing this window

Load all groups on Start

Layers

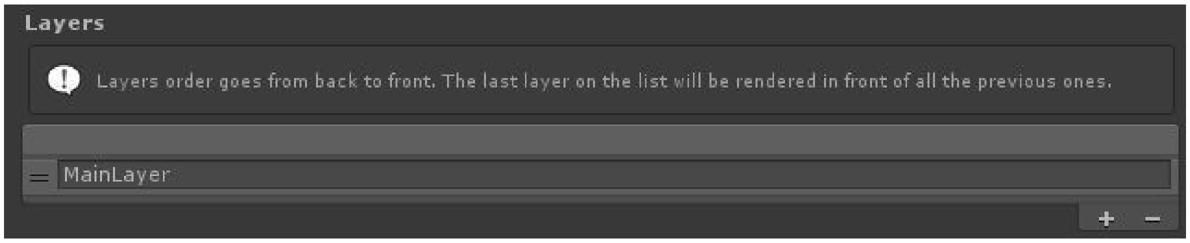
You need to create at least 1 layer

List is Empty

Groups

You need to create at least 1 group

4. You must add at least one Layer to setup the menus. You can add more layers using the buttons on the corner.



Layers

Layers order goes from back to front. The last layer on the list will be rendered in front of all the previous ones.

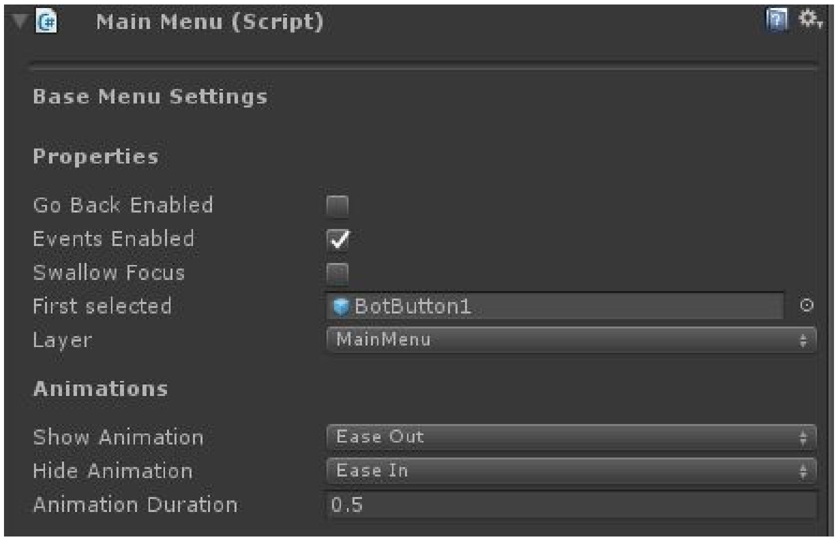
MainLayer

5. Now we're gonna setup our own menu. To create and manage your menus, each menu must inherit from Menu class to be accepted. A custom menu type should follow the next form:

```
using UnityEngine;
using GameTroopers.UI;

public class MainMenu : Menu
{
  private void Start() { }
}
```

Try to attach your menu type script in your menu gameObject and create a prefab. You need to have at least one layer created on the UI Settings to configure the menu settings. Once created, you will be able to customize the custom menu component from prefab and select a layer.



Main Menu (Script)

Base Menu Settings

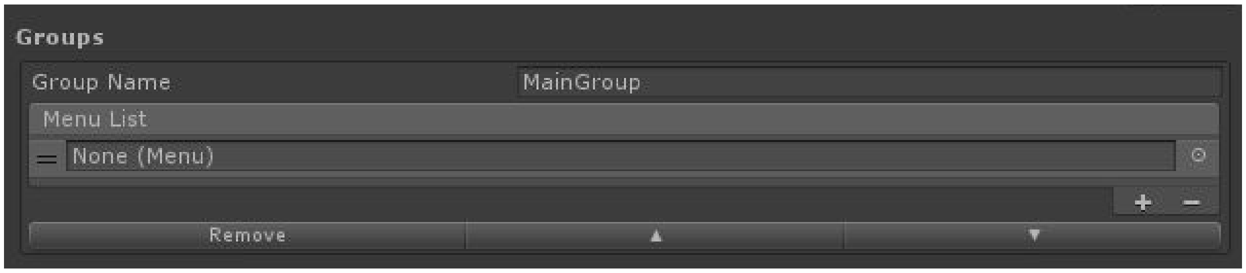
Properties

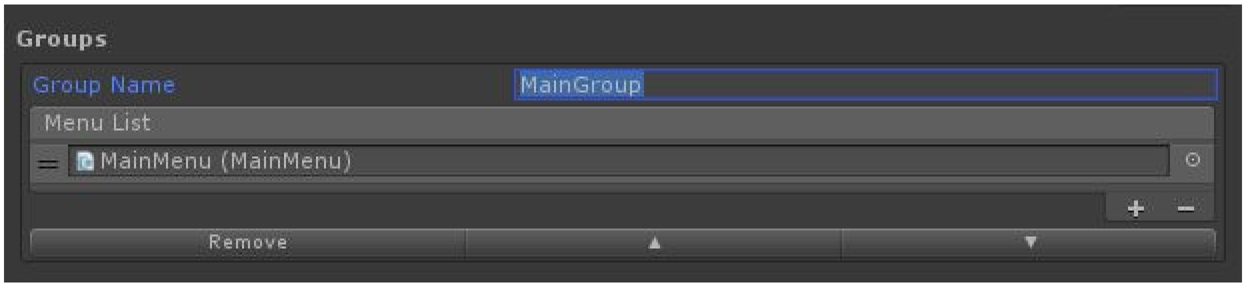
Go Back Enabled	☐
Events Enabled	✓
Swallow Focus	☐
First selected	BotButton1
Layer	MainMenu

Animations

Show Animation	Ease Out
Hide Animation	Ease In
Animation Duration	0.5

6. Now you can add the prefabs on the group section, again on the UI Settings asset. Create a new group with a unique Group ID and add a reference to the menu prefab.





7. To finish the setup, apply settings. Now you are ready to use the API on code.

First steps

Once the settings are configured, you should be able to open and hide your menu prefabs.

1. First of all, you need to load the group that contains the target menu, i.e: *MainGroup***.** After load, you can show and hide the target menu specifying the menu type from the *MenuManager*.

```
private void Start()
{
    m_menuManager.LoadGroup("MainGroup");

    m_menuManager.Show<MainMenu>();
}
```

Code documentation

Menu Manager

MenuManager is the responsible of create and organize menus and layers. Only one instance should exist on the scene, and already contains a DontDestroyOnLoad() call.

Exposed fields:

- **Canvas:**
    - The canvas where the layers and menus will be instantiated.
- **Event system:**
    - Needed to store the last selectable between menus and set it when the user navigates back in history.

Layer and group settings are stored in a UISettings asset file, which can be accessed from the MenuManager.

Public methods:

- **LoadGroup** (string groupName):
    - Loads and initialize a group and the containing prefabs. After the initialize for each method, a OnMenuLoaded event is dispatched.
- **UnloadGroup**(string groupName):
    - Unload the target group from the scene.
- **GetMenu**<T>():
    - Returns a reference to the menu with given Type.
- **ShowMenu**<T>(bool shouldPlayAnimation = true):
    - Generic method with a bool parameter to make instant transitions. Gets the menu with given type and shows it in the target layer, hiding the current menu in it, if any.
- **HideMenu**<T>(bool shouldPlayAnimation = true):
    - Generic method with a bool parameter to make instant transitions. Gets the last menu in the stack of the target layer and hides it. If there are more menus on the stack, opens the one on the top of it.
- **HideAll**(bool shouldPlayAnimation = true):

    - Hides all the menus.
- **GetPreviousMenu**(string layerName):
    - Returns the previous menu to know where you come from.
- **InitMenus**(UISettings settings):
    - Loads an specific UISettings. Is not necessary call it manually; will be called automatically on MenuManager Start() with the default UISettings file. Cannot load a new UISettings if another one is already loaded.
- **GoBack**():
    - Calls the HandleGoBack method on all the top activated menus. Overwritable

Menu

Base class of all the UI panels that need to be managed by the MenuManager. This class is responsible of managing the lifecycle the menu and providing access to its properties and must be inherited by your own menu classes to be managed as such.

Exposed fields:

- **Go Back Enabled**:
    - If selected, the back event will be called on the menu (Esc key on keyboard, back button on mobile devices, Back button on Controller), hiding the menu and going back on history.
- **Events Enabled**:
    - If enabled, the current menu will receive the subscribed events.
- **Swallow Focus**:
    - By default, all the active menus on each layer will receive focus/input. If enabled, the focus will finish when this menu is shown and receiving all the focus over himself.
- **First Selected**:
    - A selectable which will be selected automatically when the menu gains focus (e.g: a button). If empty, no button will be selected.
- **Layer**:
    - Menu layer. At least one layer must be defined on the UI Settings.
- **Show Animation**:
    - Animation that will be player on show. Can be overwritten with a custom animation assigning the ShowAction action from code.
- **Hide Animation**:
    - Animation that will be player on hide. Can be overwritten with a custom animation assigning the HideAction action from code.
- **Animation Duration**:

    - Duration for both animations: show and hide.

<u>Public methods and properties:</u>

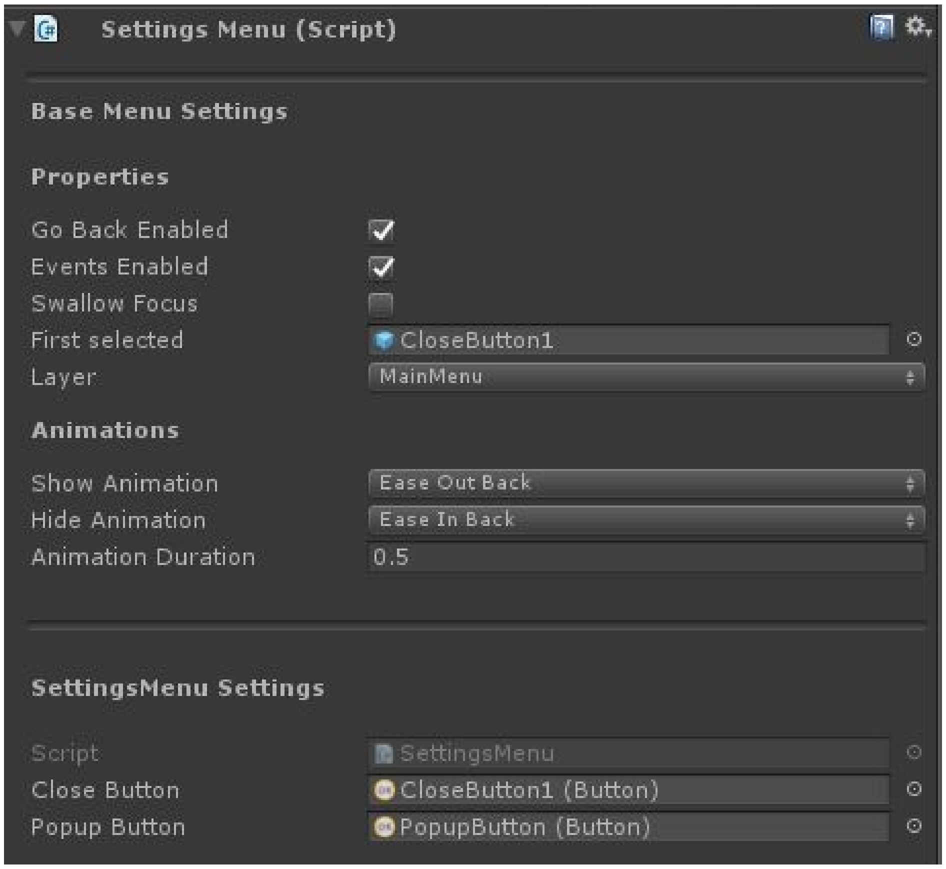
- **Layer**:
    - Layer name.
- **IsActive**:
    - Current state of the menu gameobject.
- **ShowAction**:
    - Virtual action which can be overwritten to set the show animation.
- **HideAction**:
    - Virtual action which can be overwritten to set the show animation.
- **GetMenuOnScreenPosition**():
    - Virtual method which can be overwritten to set the final position.
- **GetMenuOffScreenPosition**():
    - Virtual method which can be overwritten to set the start position.
- **HandleGoBack**():
    - Virtual method which can be overwritten to set the behaviour when the back event is triggered. By default, the current menu is hidden and will show the previous one in history (if possible).

Events

The following events are dispatched for all the menus that inherit the corresponding interface. You can add it on each custom menu class and generating the corresponding method, which be called on dispatch.

- **OnMenuLoaded**:
    - This callback will be executed when the menu is loaded into memory. This can happen after a LoadGroup call or during the game initialization if UISettings.loadAllOnStart option is selected on settings. Interface: *IOnMenuLoaded.*
- **OnMenuShowStarted**:
    - This callback will be executed when a menu is about to start playing its show animation, so it is still visible at this point. Interface: *IOnMenuShowStarted.*
- **OnMenuShowEnded**:
    - This callback will be executed when the hide animation of a menu has finished and it is not longer visible. Interface: *IOnMenuShowEnded.*
- **OnMenuHideStarted**:

    - This callback will be executed when a menu is about to start playing its hide animation, so it is still visible at this point. Interface: *IOnMenuHideStarted*.
- **OnMenuHideEnded**:
    - This callback will be executed when the hide animation of a menu has finished and it is not longer visible. Interface: *IOnMenuHideEnded*.
- **OnMenuFocused**:
    - This callback will be executed when a menu gains the input and is ready to use, always after the OnMenuHideEnded Interface: *IOnMenuFocused*.
- **OnMenuUnfocused**:
    - This callback will be executed when a menu lose the input, always before the OnMenuShowStarted. Interface: *IOnMenuUnfocused*.


Settings Menu (Script)

Base Menu Settings

Properties

Go Back Enabled	✓
Events Enabled	✓
Swallow Focus	☐
First selected	CloseButton1
Layer	MainMenu

Animations

Show Animation	Ease Out Back
Hide Animation	Ease In Back
Animation Duration	0.5

SettingsMenu Settings

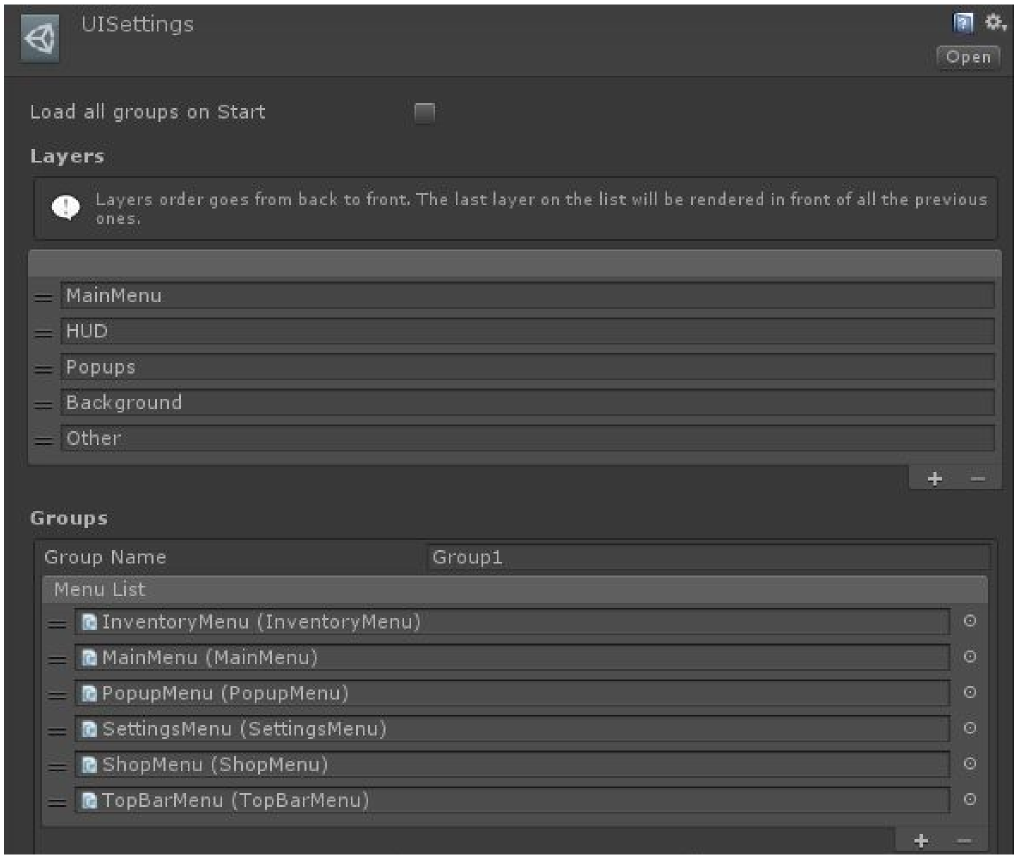
Script	SettingsMenu
Close Button	CloseButton1 (Button)
Popup Button	PopupButton (Button)

UI Settings

The asset file stores the menu settings, including the layers and groups composition and prefabs.

Exposed fields:

- **Load all groups on start:**
    - Groups and menus can be loaded from the beginning. Is this check is disabled, you have to load groups manually from code using the unique string identifier (group name). Please take in mind that loading menus can be a heavy operation, so we recommend to use loading screens during the load and unload operations.
- **Layers:**
    - Different layers where the menus will be instantiated. From the top to the bottom, the lower layers will overlap the other ones (e.g: commonly, popups should be the last layer). Only one menu type allowed.
- **Groups**:
    - Menu prefabs separated by groups. Each group represents a block of menus that will be loaded at the same time, e.g: TitleMenu, Gameplay, EditMode, VisitMode.



Load all groups on Start

Layers

Layers order goes from back to front. The last layer on the list will be rendered in front of all the previous ones.

MainMenu
HUD
Popups
Background
Other

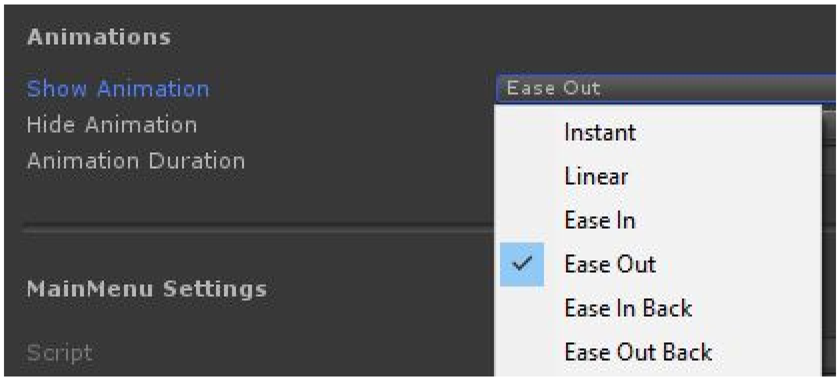
Groups

Group Name: Group1

Menu List
InventoryMenu (InventoryMenu)
MainMenu (MainMenu)
PopupMenu (PopupMenu)
SettingsMenu (SettingsMenu)
ShopMenu (ShopMenu)
TopBarMenu (TopBarMenu)

Integrating DOTween

Arcadia Library allows default dotween animations on menus. Linear and easing transitions needs DOTween configured on project to work correctly.


Animations

Show Animation: Ease Out

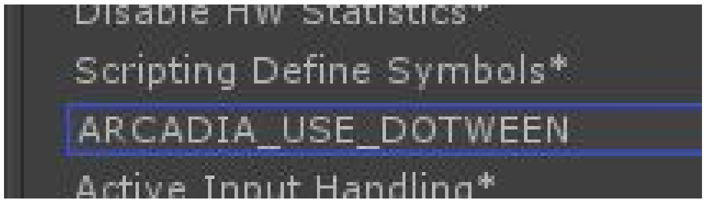
Hide Animation

Animation Duration

MainMenu Settings

Script

To enable and use these animations, *ARCADIA_USE_DOTWEEN* symbol must be set on the Scripting Define Symbols field, on ***Edit -> Project Settings -> Player -> Other Settings.***


Disable HW Statistics*

Scripting Define Symbols*

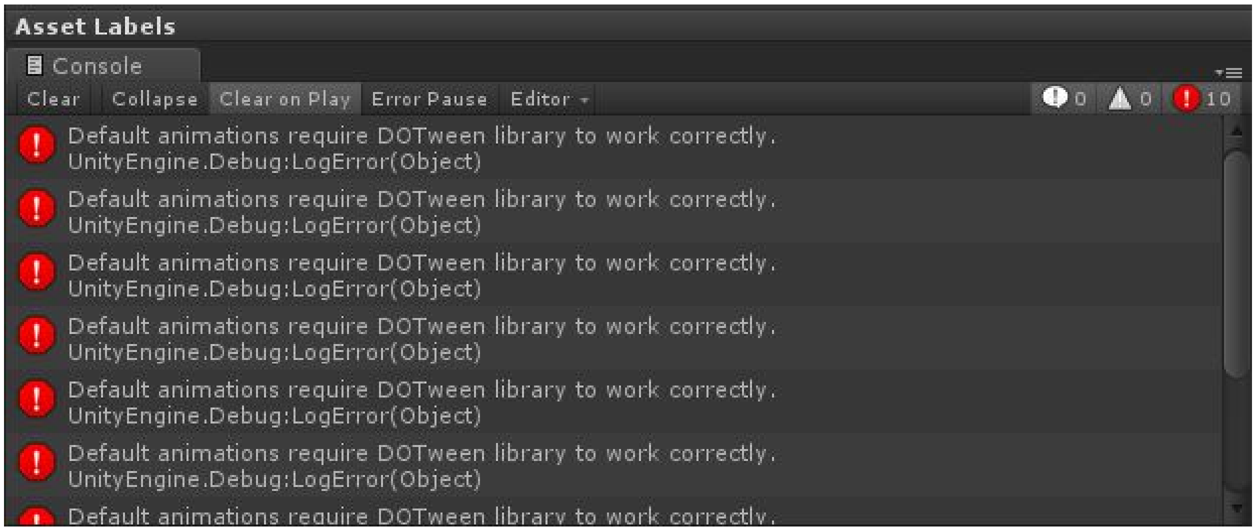
ARCADIA_USE_DOTWEEN

Active Input Handling*

FAQ

- *Why I'm getting DOTween errors?*

Default animations (including default prefabs from main scene) use dotween animations. Please ensure that DOTween is correctly installed. If this error persist, probably you are missing the DOTWEEN symbol (check [Integrating DOTween]()).



```
Default animations require DOTween library to work correctly.
UnityEngine.Debug:LogError(Object)
Default animations require DOTween library to work correctly.
UnityEngine.Debug:LogError(Object)
Default animations require DOTween library to work correctly.
UnityEngine.Debug:LogError(Object)
Default animations require DOTween library to work correctly.
UnityEngine.Debug:LogError(Object)
Default animations require DOTween library to work correctly.
UnityEngine.Debug:LogError(Object)
Default animations require DOTween library to work correctly.
UnityEngine.Debug:LogError(Object)
Default animations require DOTween library to work correctly.
```

- *But I don't want to use DOTween.*

Then just replace the default easing animation types by a instant or custom type