

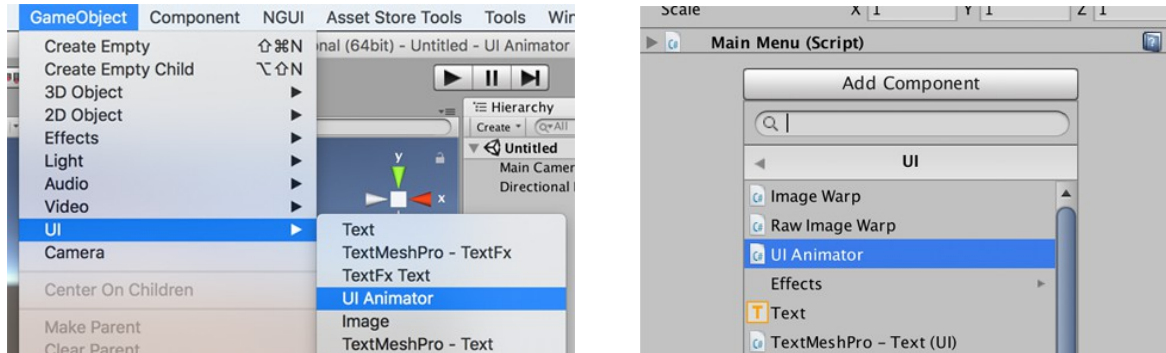
Table of Contents

Getting Started.....	3
Add a UI Animator component in your scene.....	3
Pick the animation type you want to setup.....	3
Drag & drop UI objects that you want to animate.....	3
Add some animation steps.....	4
Play around a bit!.....	5
Setup to play at runtime.....	5
Other Fundamentals.....	7
What Are 'Stages'?.....	7
What are 'Animation Instances'?.....	8
How to change the UI layout after setting up an Animation?.....	9
How to Animate a Group of Objects together?.....	10
How to Reset an Animation Step to Default Values.....	11
What Does 'Wait For All Finish?' do?.....	12
Further Topics.....	13
The Animation 'AutoPlay Behaviour Matrix'.....	13
Loop Animation Behaviour setting.....	14
Adding AudioClips to your Animation Steps.....	15
Assign an onStart / onFinish callback?.....	17
Trigger Animations OnPointerEnter / Exit.....	18
How Can I Reuse UI Animations in a Modular Way?.....	19
What is the 'Continuous' PlayMode, and when should I use it?.....	20
Scripting With UI Animator.....	21
The Basics.....	21
Modifying Animation Steps by Script.....	22
Scripting API – Class - UIAnimator.cs.....	24
Properties.....	24
Static Methods.....	24
Methods.....	24
Scripting API – Class - BaseAnimationStep.cs.....	26
Properties.....	26
Methods.....	26
Scripting API – Class - FadeAlphaAnimationStep.cs.....	26
Methods.....	26
Scripting API – Class - LayoutElementAnimationStep.cs.....	26
Properties.....	26
Methods.....	26
Scripting API – Class - PulseAnimationStep.cs.....	27
Properties.....	27
Scripting API – Class - ShakeAnimationStep.cs.....	27
Properties.....	27
Scripting API – Class - TransformAndFadeAlphaAnimationStep.cs.....	27
Methods.....	27

<u>Scripting API – Class - TransformAnimationStep.cs.....</u>	<u>28</u>
<u>Properties.....</u>	<u>28</u>
<u>Scripting API – Class - AnimStepVariableFloat.cs.....</u>	<u>28</u>
<u>Properties.....</u>	<u>28</u>
<u>Methods.....</u>	<u>28</u>
<u>Scripting API – Class - AnimStepVariablePositiveFloat.cs.....</u>	<u>28</u>
<u>Scripting API – Class - AnimStepVariableVector3.cs.....</u>	<u>29</u>
<u>Properties.....</u>	<u>29</u>
<u>Methods.....</u>	<u>29</u>
<u>Scripting API – Class - AudioClipData.cs.....</u>	<u>29</u>
<u>Properties.....</u>	<u>29</u>
<u>Methods.....</u>	<u>29</u>
<u>Scripting API – Enum – AnimSetupType.....</u>	<u>30</u>
<u>Scripting API – Enum – PlayTimeMode.....</u>	<u>30</u>
<u>Scripting API – Enum – LayoutElementParam.....</u>	<u>30</u>
<u>Scripting API – Enum – CLIP_TRIGGER_POINT.....</u>	<u>30</u>
<u>Support.....</u>	<u>31</u>
<u>Changelog.....</u>	<u>32</u>

Getting Started

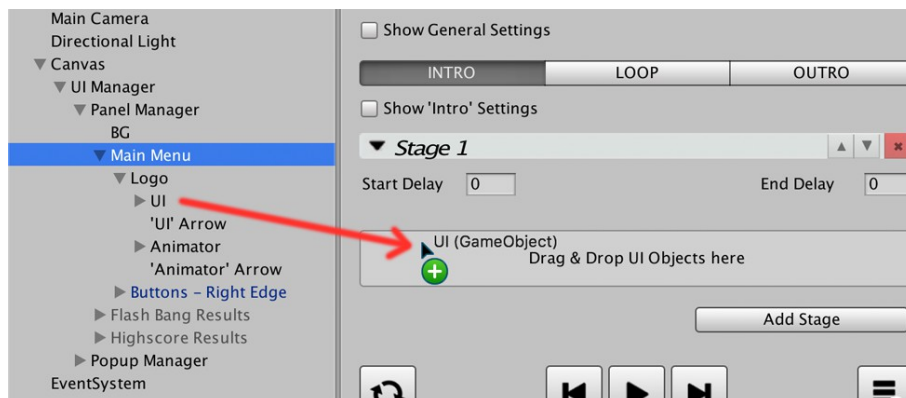
Add a UI Animator component in your scene.



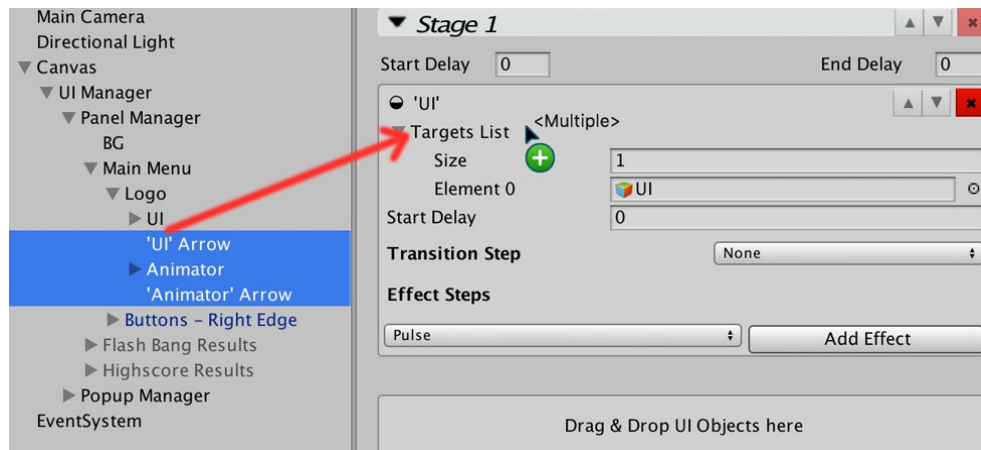
Pick the animation type you want to setup.



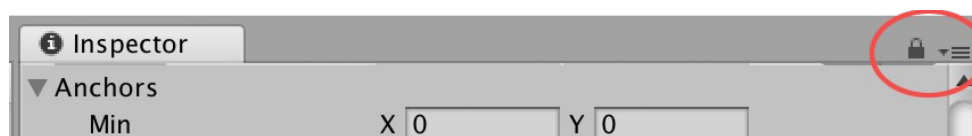
Drag & drop UI objects that you want to animate.



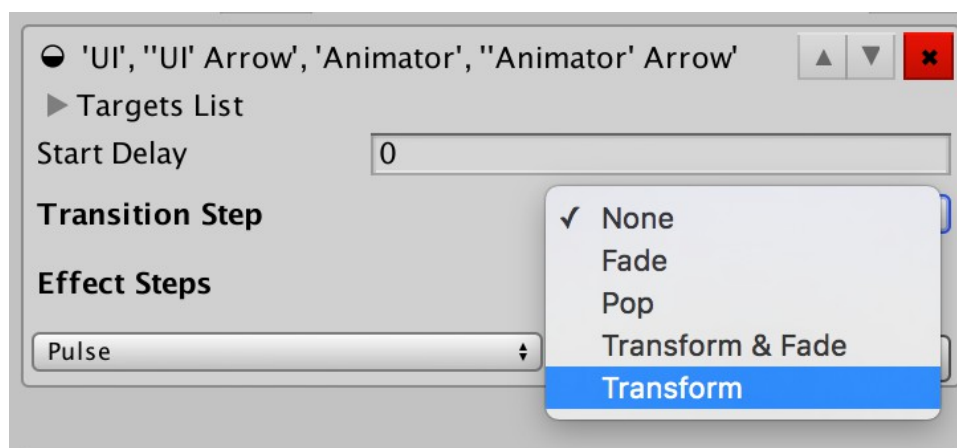
- You can drag more than one object to animate a group.
- You can always add/remove targets later from the targets list.



TIP: Remember to make use of the Inspector 'Lock'

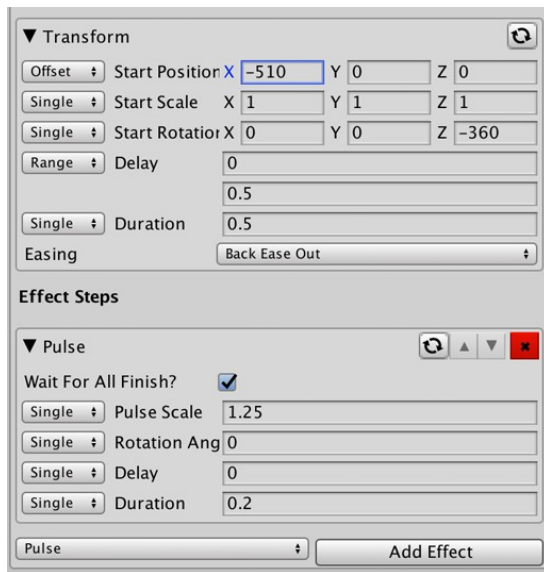


Add some animation steps.

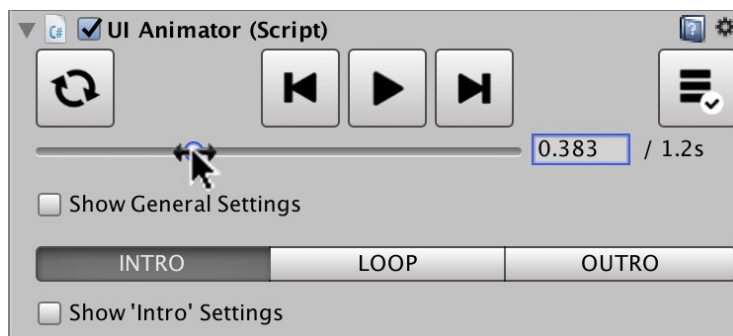


- There are two type of animation step; **Transition & Effect**.
 - **Transition Steps** – Meant for transitioning objects in/out
eg. Sliding a button in from off-screen left.
 - **Effect Steps** – Applies an animation effect, and always returns it to its original state
eg. a button *pulse* effect.
- Each animation type (**Intro, Loop, Outro**) is made up differently.
 - **Intro** – Zero or one *Transition* Step, followed by zero or more *Effect* steps.
 - **Loop** – Only *Effect* steps.
 - **Outro** – Zero or more *Effect* steps, followed by zero or one *Transition* step.

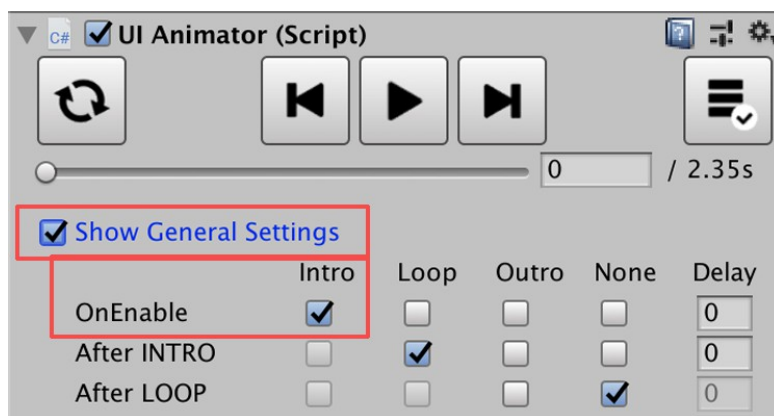
Play around a bit!



Preview your animation in the editor at any point, using the **playback controls** and the **animation slider**.



Setup to play at runtime.



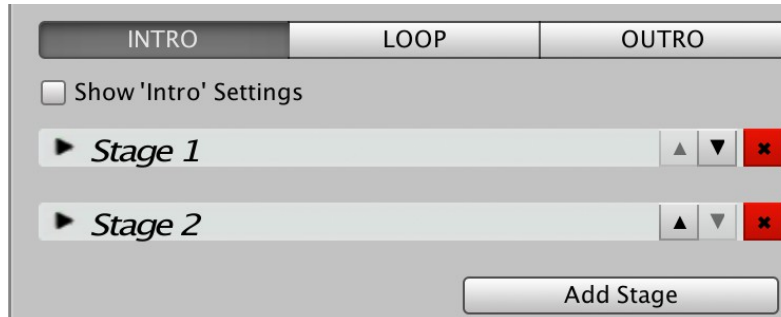
By default your new animations will be set to play the ***Intro*** animation when **OnEnable** is called, but this can be changed via the **General Settings**.

```
1 using UnityEngine;|
2 using UIAnimatorCore;
3
4 public class UiAnimatorTest : MonoBehaviour {
5
6     public UIAnimator m_uiAnimator;
7
8     void Start ()
9     {
10         m_uiAnimator.PlayAnimation (AnimSetupType.Intro);
11     }
12 }
```

- You could also trigger an animation via a scripting call to **PlayAnimation()**, like in the above example.
- See the ***Scripting API*** for full details on the scripting methods available.

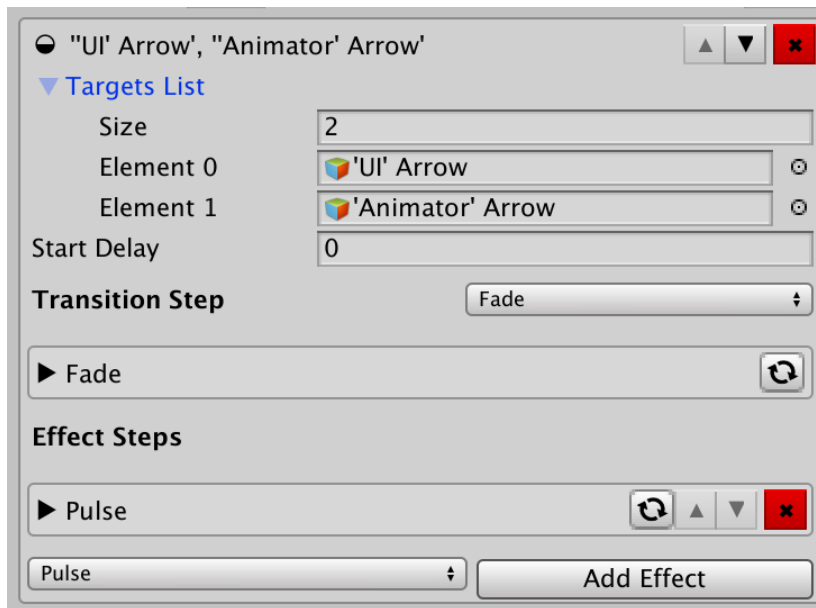
Other Fundamentals

What Are 'Stages'?



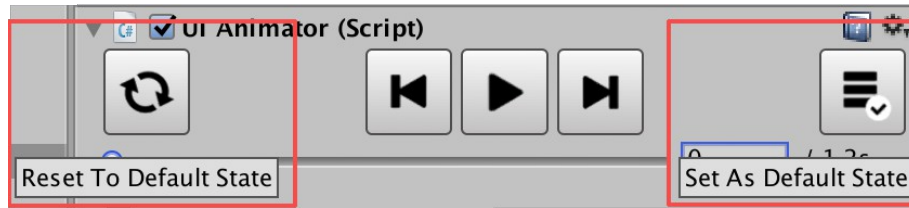
- A *Stage* is just a *collection* of *Animation Instances*.
- Each *Stage* waits for all of the contained *Animation Instances* to finish, before moving onto the next stage.
- You'll mostly only use one Stage.

What are 'Animation Instances'?



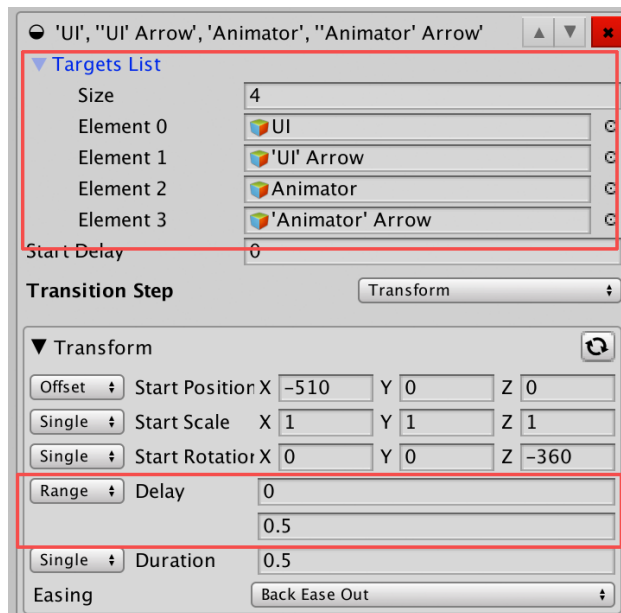
- An *Animation Instance* is a configuration of *Animation Steps* (Transition or Effect) which are to be applied to one or more *Target* UI gameObjects.
- Whether you are setting up an *Intro*, *Loop* or *Outro* animation, the setup of the *Animation Instance* will vary:
 - **Intro** – Zero or one *Transition* Step, followed by zero or more *Effect* steps.
 - **Loop** – Only *Effect* steps.
 - **Outro** – Zero or more *Effect* steps, followed by zero or one *Transition* step.

How to change the UI layout after setting up an Animation?



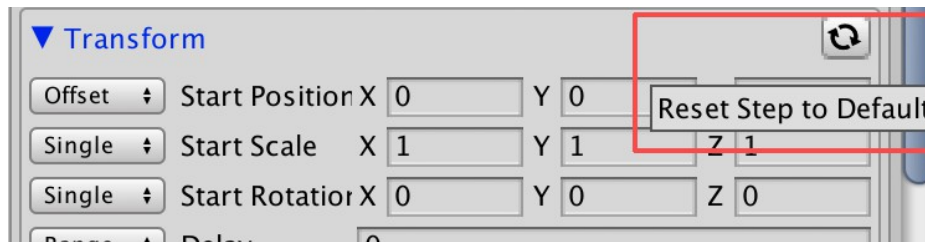
- Animation steps *cache the state* of the UI when you first add them, so that they know what to animate to/from.
- If you want to change the default UI layout that gets animated, you'll need to:
 - Make sure the animation is in its Default (Non-animated) state by **Resetting to Default State**. (See above image)
 - Make your changes to the UI layout.
 - Press the **Set As Default State** button. (See above). This will update the animation steps cached states to the current UI layout.

How to Animate a Group of Objects together?



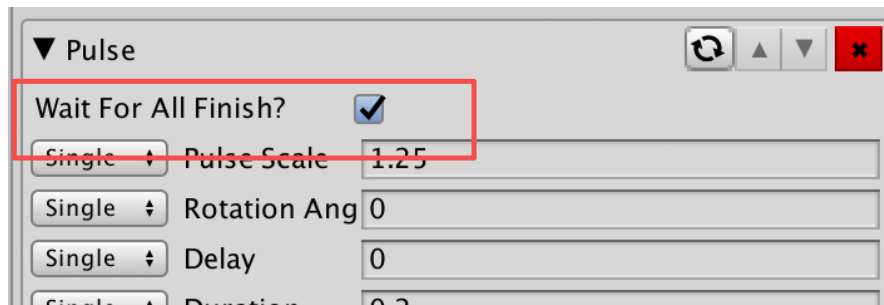
- You can assign multiple target objects to each animation setup.
- This helps to save time when you want a group of objects to animate in a similar way.
eg. a row of buttons popping into existence.
- As soon as an animation setup has more than one target, the interface changes to show options for each variable to either be:
 - **Single** - A single shared value for each target.
 - **Range** - A range of values, from/to, applied to each target in order.

How to Reset an Animation Step to Default Values.



- You can reset the values of any Animation Step by pressing the ***‘Reset Step to Default’*** button.

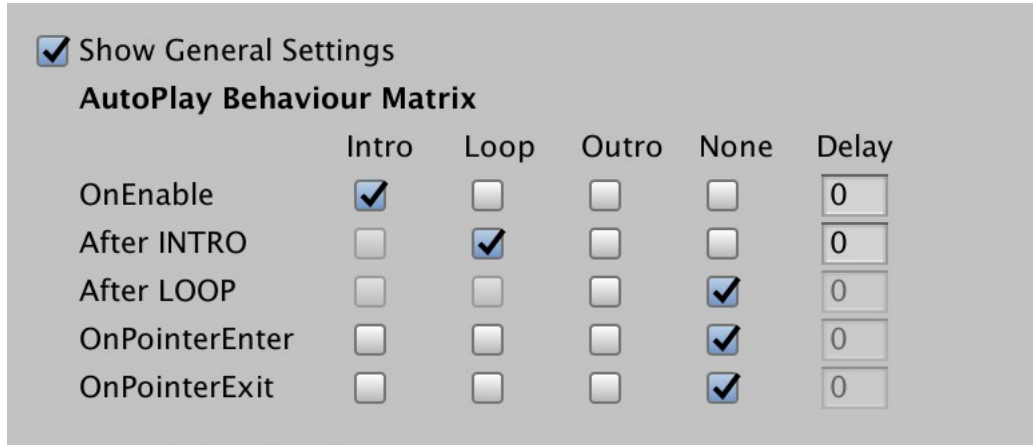
What Does 'Wait For All Finish?' do?



- If you're animating on a group of targets, then any Animation Step after the first will have the option '*Wait For All Finish?*'
- This option denotes whether the animation step should *wait for all targets of the previous steps* to be *finished*, or just start whenever any of the targets has finished.
- This only has an effect if you're setting a *range* of different *delays* or *durations*.

Further Topics

The Animation 'AutoPlay Behaviour Matrix'



The screenshot shows a settings panel with a checked checkbox for 'Show General Settings'. Below it is the 'AutoPlay Behaviour Matrix' section. It contains a table with five rows of events and six columns of options: Intro, Loop, Outro, None, and Delay. The 'Delay' column contains input boxes with the value '0'. The 'Intro' column has a checked checkbox for 'OnEnable'. The 'Loop' column has a checked checkbox for 'After INTRO'. The 'None' column has checked checkboxes for 'After LOOP', 'OnPointerEnter', and 'OnPointerExit'.

	Intro	Loop	Outro	None	Delay
OnEnable	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0
After INTRO	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0
After LOOP	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0
OnPointerEnter	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0
OnPointerExit	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0

- This is a time saving feature to reduce the need for coding calls to trigger animations.
- Using the AutoPlay Behaviour Matrix, you can configure your UI Animation to automatically play a particular animation state after certain events:
 - **OnEnable** – Set the animation to be played when the *UI Animator* gameObject is first active in the scene.
 - *By default this is set to play the 'Intro' animation onEnable.*
 - *Setting this option to 'None' will present an additional option to set which Animation State pose (Intro, Loop, Outro) the UI Animator instance should be set OnEnable.*
 - **After INTRO** - Set the animation to be played when the 'Intro' animation has finished playing.
 - **After LOOP** – Set the animation to be played when the 'Loop' animation has finished playing.
 - **OnPointerEnter** – Set the animation to be played when the *UI Animator* content registers an *OnPointerEnter* event.
 - **TIP:** *It's often useful to add an additional **transparent Image** object to be used as a consistent onPointer trigger area, which isn't animated.*
 - **OnPointerExit** – Set the animation to be played when the UI Animator content registers an *OnPointerExit* event.

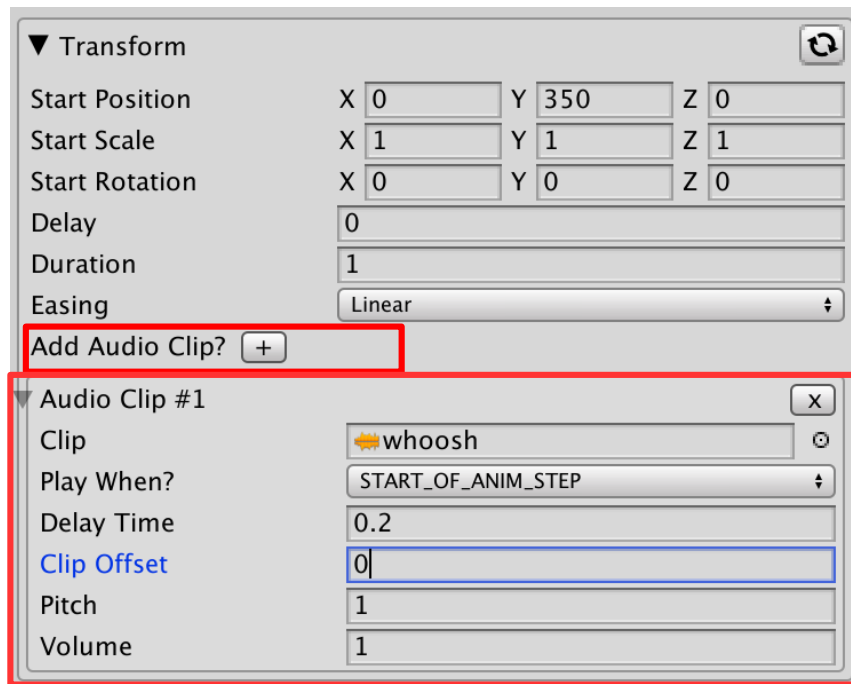
Loop Animation Behaviour setting

Loop Animation Behaviour
Play Infinitely? ☒

Loop Animation Behaviour
Play Infinitely? ☐ Num Iterations

- A setting found under the '*Show General Settings*' option.
- When enabled, the 'Loop' animation will continue to play infinitely until manually told to stop or change. *This is the default setting.*
- When disabled, you can specify how many times the 'Loop' animation should play before it automatically stops playing.
 - **Note:** If you want the animation to play the '*Loop*' for a fixed number of times and then automatically lead into the '*Outro*' animation, use this setting in combination with the '*[AutoPlay Behaviour Matrix](#)*'.

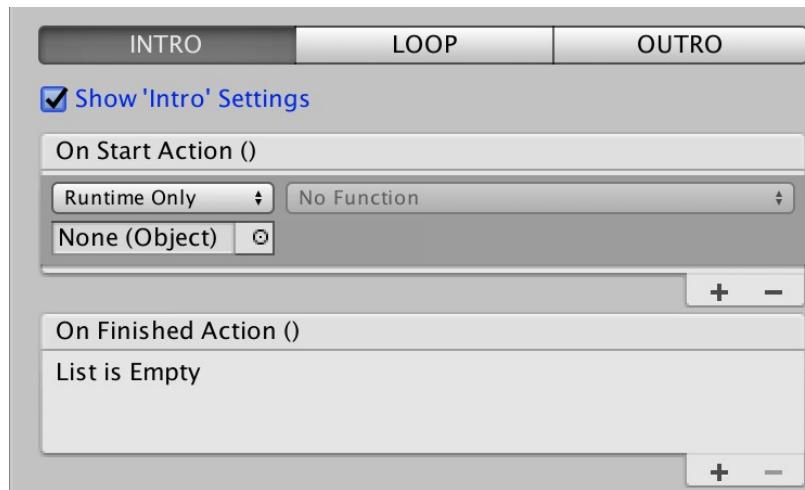
Adding AudioClips to your Animation Steps



- Any Animation Step (Transition or Effect), can have AudioClips assigned to it to triggered at certain points.
- Click the Add ('+') button next to the '*Add Audio Clip?*' option at the bottom of the animation step settings.
- You can assign as many AudioClips as you like.
- Simply assign an AudioClip currently imported into your Unity project, and then setup when and how it should play:
 - **Play When?** - Set whether the clip should play at the start or the end of the animation step.
 - **Delay Time**
 - If the clip is set to play from the *Start*, this will be the time delay in seconds from when the animation step starts playing.
 - If the clip is set to play from the *End*, this will be the time in seconds from the end of the animation step, that the clip should start playing.
 - eg. '0' would be at the end of the animation step. '0.5' would be 0.5 seconds before the end of the animation step.
 - **Clip Offset** – How after into the clip (in seconds) should the clip start playing.
 - **Pitch** – The pitch that the clip should be played at.
 - '1' is the default value.
 - **Volume** – The volume that the clip should be played at.

- UIAnimator will automatically create a *pool of AudioSource gameObjects* as children of the UIAnimator gameObject.
 - **Note:** In order to avoid potential runtime performance spikes, it's recommended that you allow UIAnimator to create these AudioSource child objects **in edit mode**, and save it in your scene. This way UIAnimator can re-use the already created AudioSources at runtime, saving the creation of new GameObjects and components at runtime, which can be computationally expensive.

Assign an onStart / onFinish callback?

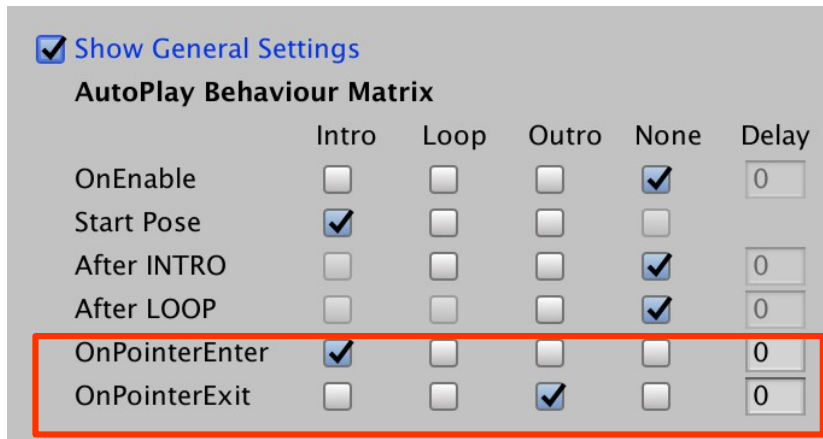


- Each animation type (*Intro*, *Loop*, *Outro*) has optional **OnStart** & **OnFinish** callback events which can be configured in the inspector.

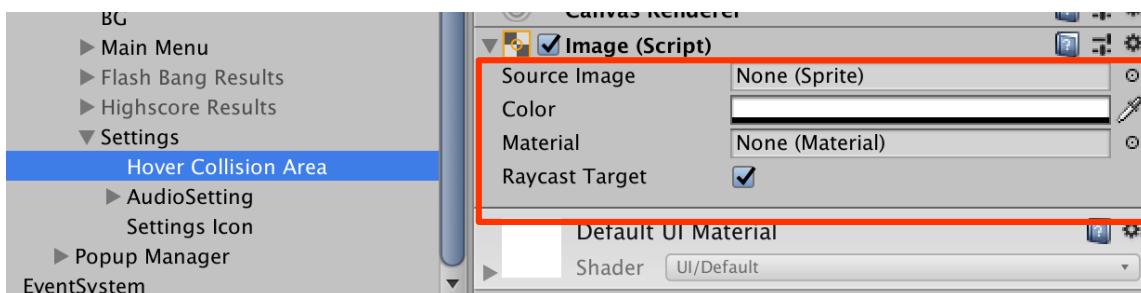
```
1 using UnityEngine;
2 using UIAnimatorCore;
3
4 public class UiAnimatorTest : MonoBehaviour {
5
6     public UIAnimator m_uiAnimator;
7
8     void Start ()
9     {
10         m_uiAnimator.PlayAnimation (AnimSetupType.Intro, a_onFinish: () => {
11
12             Debug.Log("UI Animation finished");
13
14         });
15     }
16 }
```

- You can also provide a callback method to the **a_onFinish** parameter of *PlayAnimation()*.

Trigger Animations OnPointerEnter / Exit

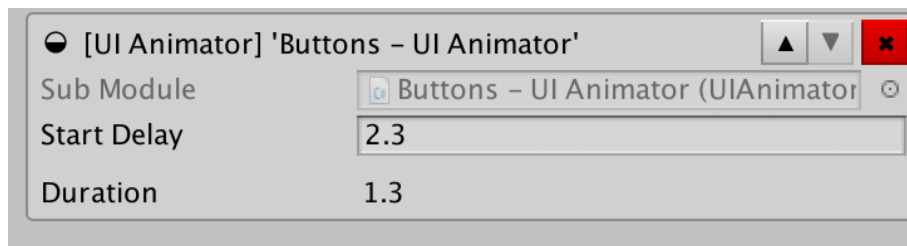
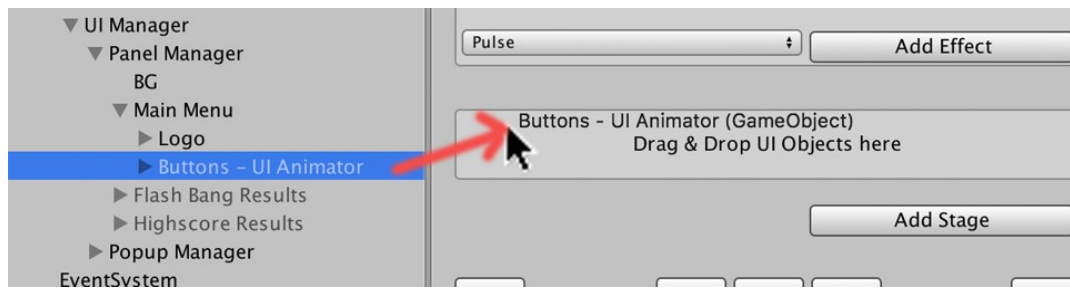


- Sometimes you may want to trigger a UI Animation when the user hovers the pointer over a particular area of your UI.
 - eg. *Hovering over a settings icon in the top left of the screen could reveal additional UI elements, and then hide them away when the pointer leaves the area. **See the UI Animator Demo Scene for an example of this!***
- This functionality can be achieved by setting up the *OnPointerEnter / Exit autoplay behaviours* under the *General Settings* for your UI Animator instance.
 - Set which animation you'd like to be triggered on each event, and specify a delay if required.
 - It's as easy as that!



- **TIP:** Create a non-animated transparent Image, to use as the pointer collision area.
 - Make sure to set this as a '**Raycast Target**' (set to **true**), so that it receives the pointer enter/exit events.
 - Set **all other non-interactive UI elements** to **not** be 'Raycast Target's (set to **false**), so that they don't accidentally trigger enter/exit states when you don't want them to!
 - See the 'Settings' UI elements in the **Demo Scene** for an example of this.

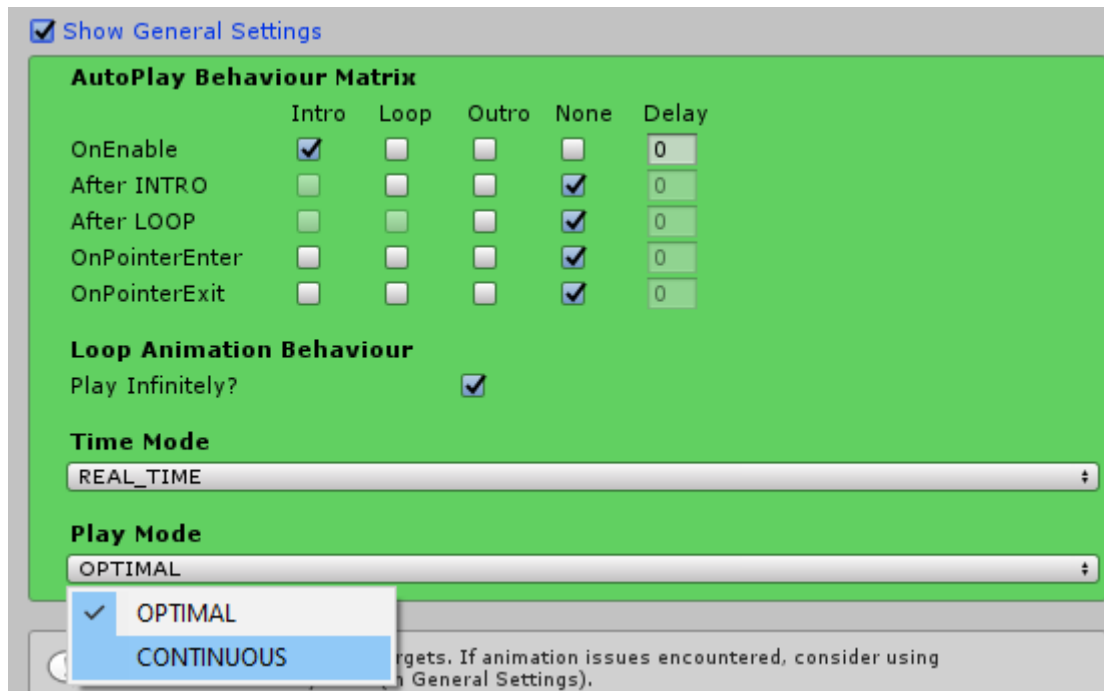
How Can I Reuse UI Animations in a Modular Way?



- You can drag other **child** UI Animator instances onto a parent UI Animator, and then control it as part of its animation sequence.
- This way you can have a **shared UI Animation prefab** which you assign to multiple parent UI Animation instances, and save having to setup and maintain the animation in multiple places.

What is the 'Continuous' PlayMode, and when should I use it?

PlayMode is an option found within the General Settings on a UIAnimator component.



By default the field will be set to '**OPTIMAL**', but there may be times when you wish/need to switch it to use '**CONTINUOUS**'.

Most UIAnimator animations will be configured to include some sort of delay, whether it's a literal delay between one Animation Step playing and the next, or a more subtle delay '*Range*' setting being used to stagger the animation of multiple UI objects.

When the **PlayMode** is in '**OPTIMAL**', any UI element which is currently *waiting* for a delay to finish will not have it's state set at all, and therefore is more optimal in the calculations and UI setup it has to do each frame.

When the **PlayMode** is in '**CONTINUOUS**' however, all UI elements are having their state updated every frame, even if they are currently not doing anything (waiting for a delay).

Where '**CONTINUOUS**' mode might be required is in cases where your target UI elements are having their transform position and scale **driven** by a Unity UI **LayoutGroup** (*HorizontalLayoutGroup* or *VerticalLayoutGroup* for example), and therefore will require continuous updating of the object states to prevent the LayoutGroup from re-setting them to their default positions.

Our advise is to use '**OPTIMAL**' until you start seeing re-positioning problems, then switch.

Scripting With UI Animator

The Basics

- Include the *UIAnimatorCore* namespace.
- Keep a reference of a *UIAnimator* component from your scene.
- Call to play an animation, and optionally listen for the onFinish callback event.

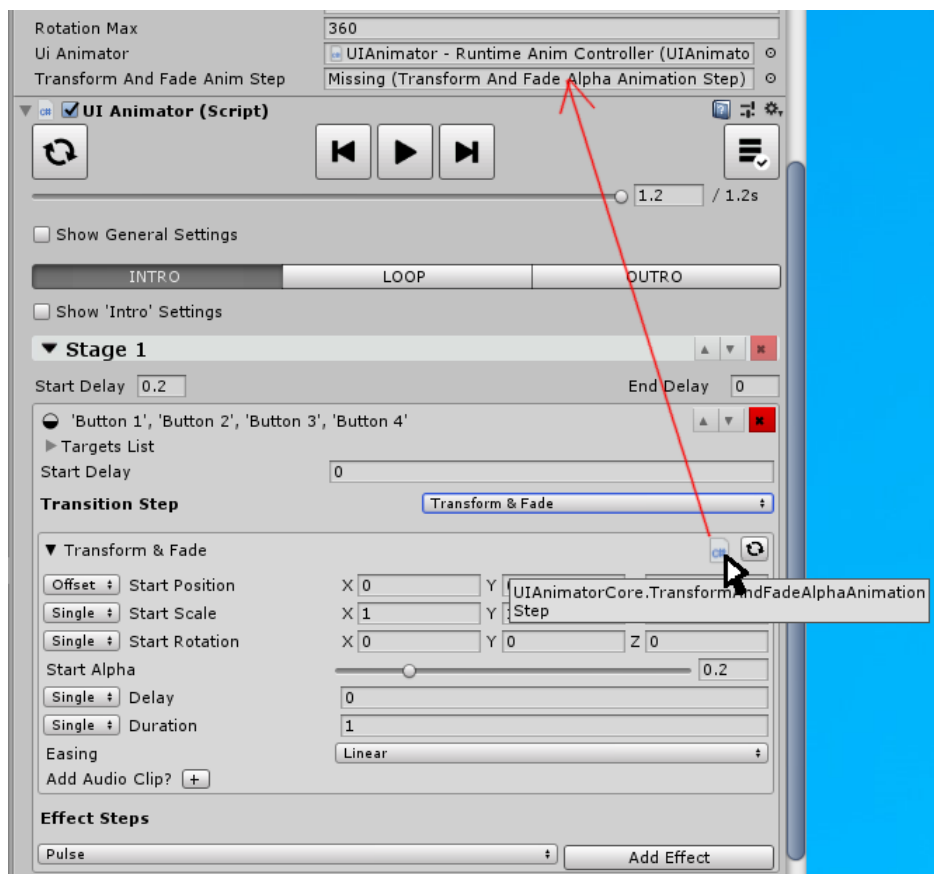
```
1 using UnityEngine;
2 using UIAnimatorCore;
3
4 public class UiAnimatorTest : MonoBehaviour {
5
6     public UIAnimator m_uiAnimator;
7
8     void Start ()
9     {
10         m_uiAnimator.PlayAnimation (AnimSetupType.Intro, OnIntroAnimationFinished);
11     }
12
13     void OnIntroAnimationFinished()
14     {
15         Debug.Log ("Intro finished");
16     }
17 }
```

Modifying Animation Steps by Script

All animation steps, whether they are a *Transition Step* or an *Effect Step*, are inheriting from a ***BaseAnimationStep*** class.

BaseAnimationStep is a MonoBehaviour (which has been set to be hidden in the inspector), so you can grab a direct reference to the Animation Step by dragging and dropping the '**c#**' icon next to the step names.

To see the full class name of the Animation Step, try hovering the cursor over the '**c#**' icon to see it displayed in a tooltip. (See below)



So you can add a **BaseAnimationStep** public variable to your component script, drag any of the AnimationStep's on your UIAnimation as a reference in the inspector, and then start changing things by script.

Below is an example of some scripting logic for setting the AudioClip of an animation at runtime, and the scale of a Transform & Fade animation step.

```
1  using UnityEngine;
2  using UIAnimatorCore;
3
4  public class UIAnimatorRuntimeSetter : MonoBehaviour {
5
6      public AudioClip m_clip1;
7      public AudioClip m_clip2;
8
9      public BaseAnimationStep[] m_genericAnimationSteps;
10
11     public TransformAndFadeAlphaAnimationStep m_transformAndFadeAnimStep;
12
13     void Update ()
14     {
15         if (Input.GetKeyDown (KeyCode.Alpha1))
16         {
17             m_genericAnimationSteps[0].AudioClipDatas[0].SetAudioClip(m_clip1);
18             m_genericAnimationSteps[0].ParentUIAnimator.PlayAnimation (AnimSetupType.Intro);
19         }
20
21         if (Input.GetKeyDown (KeyCode.Alpha2))
22         {
23             m_genericAnimationSteps[1].AudioClipDatas[0].SetAudioClip(m_clip2);
24             m_genericAnimationSteps[1].ParentUIAnimator.PlayAnimation(AnimSetupType.Intro);
25         }
26
27         if (Input.GetKeyDown(KeyCode.Space))
28         {
29             m_transformAndFadeAnimStep.Scale.SetValue(new Vector3(1, Time.time % 1, 1));
30             m_transformAndFadeAnimStep.ParentUIAnimator.PlayAnimation(AnimSetupType.Intro);
31         }
32     }
33 }
34
35
36
```

Check out Demo Scene, '*04 - Runtime Animation Changes by Script*' with main script '*RuntimeAnimationController.cs*', for a working example.

For a list of all of the public API methods available for altering the animation step states, check out the '*Scripting API*' section.

Scripting API – Class - UIAnimator.cs

Properties

AnimSetupType	CurrentAnimType { get; }
bool	IsPlaying { get; }
bool	Paused { get; set; }
PlayTimeMode	TimeMode { get; set; }
float	Timer { get; }

Static Methods

void	SetUIAudioState (bool a_audiolsPlaying);
------	---

Methods

void	ForceStopAllAudioSources ();
float	GetAnimationDuration ();
float	GetAnimationDuration (AnimSetupType a_animType);
void	PlayAnimation (AnimSetupType a_animType);
void	PlayAnimation (AnimSetupType a_animType, float a_delay, System.Action a_onFinishCallback);
void	ResetToEnd ();
void	ResetToEnd (AnimSetupType a_animType);
void	ResetToDefault ();
void	ResetToStart ();
void	ResetToStart (AnimSetupType a_animType);
void	SetAnimationTimer (float a_timerValue);

void	SetAnimationTimer (AnimSetupType a_animType, float a_timerValue);
void	SetAnimType (AnimSetupType a_animType);
void	SetPlayOnEnable (bool a_playOnEnable, AnimSetupType a_animToPlay, float a_delay = 0);
void	SetPlayAfterIntro (bool a_playAfterIntro, AnimSetupType a_animToPlay, float a_delay = 0);
void	SetPlayAfterLoop (bool a_playAfterLoop, AnimSetupType a_animToPlay, float a_delay = 0);
void	SetPlayOnPointerEnter (bool a_playOnPointerEnter, AnimSetupType a_animToPlay, float a_delay = 0);
void	SetPlayOnPointerExit (bool a_playOnPointerExit, AnimSetupType a_animToPlay, float a_delay = 0);
void	UpdateState (float a_deltaTime);

Scripting API – Class - *BaseAnimationStep.cs*

Properties

AudioClipData[]	AudioClipDatas { get; }
AnimStepVariablePositiveFloat	Delay { get; }
AnimStepVariablePositiveFloat	Duration { get; }
UIAnimator	ParentUIAnimator { get; }

Methods

bool	IsCompletelyFinished ();
void	SetEasing (EasingEquation a_easing);

Scripting API – Class - *FadeAlphaAnimationStep.cs*

Methods

void	SetAlpha (float a_alpha);
------	------------------------------------

Scripting API – Class - *LayoutElementAnimationStep.cs*

Properties

AnimStepVariablePositiveFloat	WidthScale { get; }
AnimStepVariablePositiveFloat	HeightScale { get; }

Methods

void	SetWidthParamToAnimate (LayoutElementParam a_param);
------	---

void	SetHeightParamToAnimate (LayoutElementParam a_param);
------	--

Scripting API – Class - *PulseAnimationStep.cs*

Properties

AnimStepVariablePositiveFloat	PulseScale { get; }
AnimStepVariableFloat	RotationAngle { get; }

Scripting API – Class - *ShakeAnimationStep.cs*

Properties

AnimStepVariablePositiveFloat	ShakeAmount { get; }
---	-----------------------------

Scripting API – Class - *TransformAndFadeAlphaAnimationStep.cs*

Inherits from *TransformAnimationStep.cs*

Methods

void	SetAlpha (float a_alpha);
------	------------------------------------

Scripting API – Class - TransformAnimationStep.cs

Properties

AnimStepVariableVector3	Position { get; }
AnimStepVariableVector3	Scale { get; }
AnimStepVariableVector3	Rotation { get; }

Scripting API – Class - AnimStepVariableFloat.cs

Properties

float	Value { get; }
float	ToValue { get; }

Methods

void	SetValue (float a_value);
void	SetValues (float a_fromValue, float a_toValue);

Scripting API – Class - AnimStepVariablePositiveFloat.cs

Inherits from *AnimStepVariableFloat.cs*

Only difference is that the values are capped at zero.

Scripting API – Class - AnimStepVariableVector3.cs

Properties

Vector3	Value { get; }
Vector3	ToValue { get; }

Methods

void	SetValue (Vector3 a_value);
void	SetValues (Vector3 a_fromValue, Vector3 a_toValue);

Scripting API – Class - AudioClipData.cs

Properties

AudioClip	Clip { get; }
AnimStepVariablePositiveFloat	Delay { get; }
AnimStepVariablePositiveFloat	OffsetTime { get; }
AnimStepVariablePositiveFloat	Pitch { get; }
CLIP_TRIGGER_POINT	TriggerPoint { get; }
AnimStepVariablePositiveFloat	Volume { get; }

Methods

void	SetAudioClip (AudioClip a_audioClip);
void	SetPlayWhen (CLIP_TRIGGER_POINT a_triggerPoint);

Scripting API – Enum – AnimSetupType

Intro

Loop

Outro

Scripting API – Enum – PlayTimeMode

GAME_TIME

REAL_TIME

Scripting API – Enum – LayoutElementParam

MIN

PREFFERED

FLEXIBLE

Scripting API – Enum – CLIP_TRIGGER_POINT

START_OF_ANIM_STEP

END_OF_ANIM_STEP

Support

Support email: fenderrio@gmail.com

Thank you for buying UI Animator! :)

Changelog

v1.0.0 – 02/04/2018

- First release

v1.1.0 – 08/09/2018

- Added option to trigger AudioClips during UI Animations.
- Tidied up the 'AutoPlay' functionality; now an easy-to-use Behaviour Matrix (in General Settings)
- Added support for triggering animations OnPointerEnter / Exit
- Tidied up and improved the User Guide; now a proper PDF with hyperlinked contents table.
- Bug fixes.

v1.2.0 – 23/10/2019

- Added new animation step '**LayoutElement Animator**' for animating the width/height of LayoutElements.
- Added a 'CONTINUOUS' play mode, to help with animations fighting against UI LayoutGroups.
- Added new demo scenes
- Fixed up some Editor inspector styling issues and handled editor warnings
- Fixed some minor bugs

v1.3.0 – 20/02/2020

- Added ability to drag & drop AnimationStep component references to your scripts
- Added extensive new scripting API calls to manipulate all animation step settings by script.
- Added a new demo scene '*Runtime Animation Changes by Script*'
- Fixed a few bugs