UI Toolkit Text Animation Manual

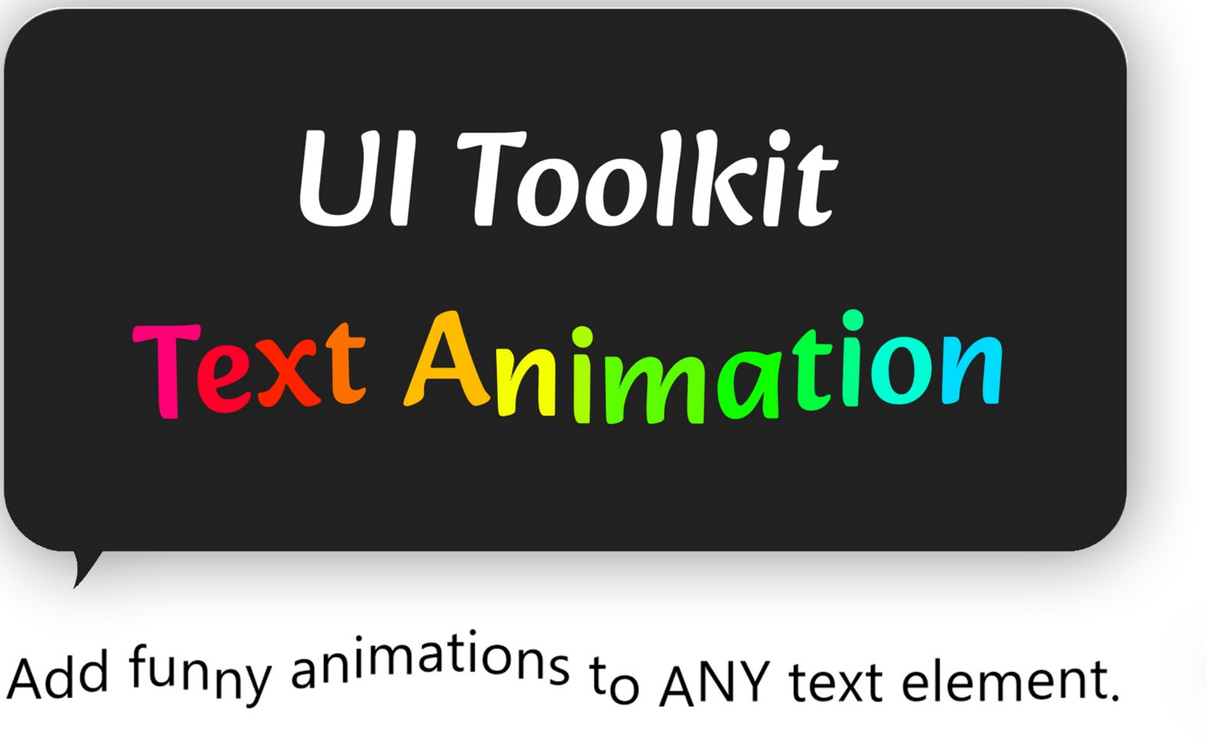


Table of contents

Requirements & Setup

Requirements

Unity 2022.3 or higher is required since that is the min version Unity allows for new assets in the store. However it may work just fine in older versions of Unity with minor changes.

Unity.Collections Package This asset uses some of the Utility classes from the Unity.Collections.LowLevel.Unsafe namespace. Thus it requires you to install the collecions package. Depending on you Unity version (Unity 6.0+) it may already be installed. If you have URP or HDRP installed then you will also have it installed already.

Introduction

What’s the key difference to all the other text animation solutions?

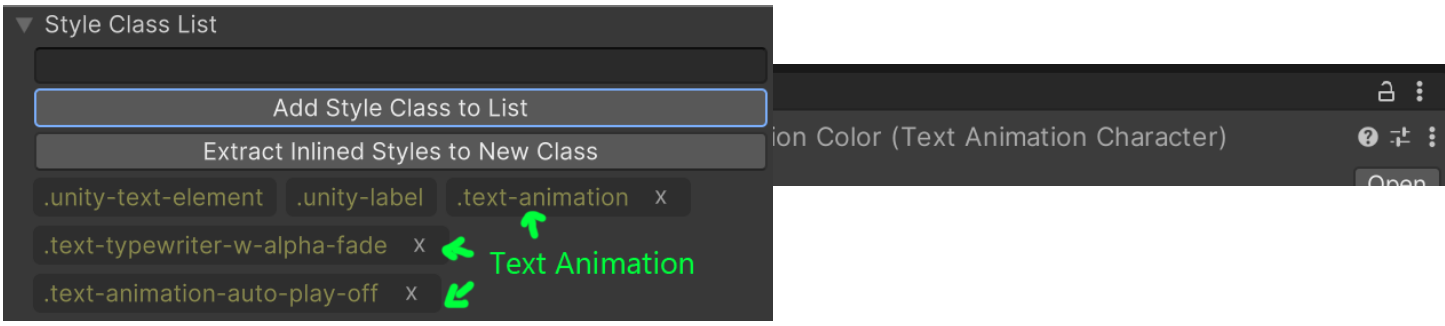
A) It’s for the UI Toolkit (not uGUI)
B) It is non-destructive. Which means you can add it to your project at any time (no custom label elements or similar). It works for any TextElement.

The system is based on manipulators. This means you can add or remove the animation from any text element in your project. It does not require you to use any special custom element. The advantage of this approach is that you can easily try it on your project without having to change anything about your UI structure.

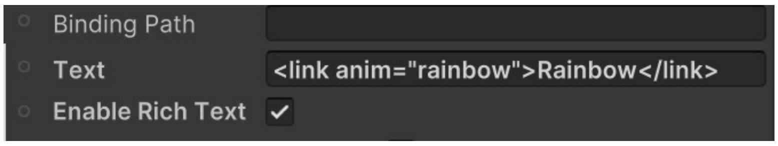
Sadly, this also has one big disadvantage. Manipulators can not add new attributes to your UI elements. This means all the configuration has to be done elsewhere. The asset uses three mechanisms for that:

1) **Scriptable Objects** (Animation Configs)
They are used to configure all the animation details like custom curves, colors, ...

2) **USS class names** to add typewriter animations.


Style Class List
Add Style Class to List
Extract Inlined Styles to New Class
.unity-text-element .unity-label .text-animation x
.text-typewriter-w-alpha-fade x
.text-animation-auto-play-off x
ion Color (Text Animation Character)

3) **<link anim=“ID“>** tags in the text to animate only some specific characters.


Binding Path
Text: <link anim="rainbow">Rainbow</link>
Enable Rich Text: ✓

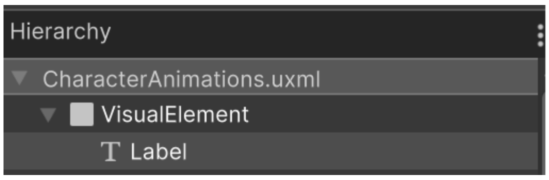
Info: Enable „Rich Text“ to use the tags.

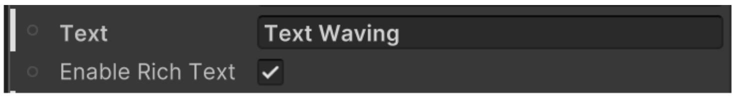
Please notice that Unity does not (yet) provide a public Text API and thus this uses some TextCore internal classes. This is mostly relevant for coders (if you want to extend the system).

Usage

Adding animations to existing text elements

Let's start from scratch with a default label:




Text: Text Waving
Enable Rich Text: ✓

Our goal is to animate the „Waving" part of the text. To do that we have to do three things:

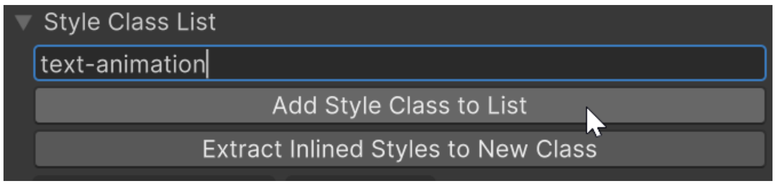
1) Enable text animation system for this label.

2) Configure the animation.

3) Add the TextAnimation system to the scene.

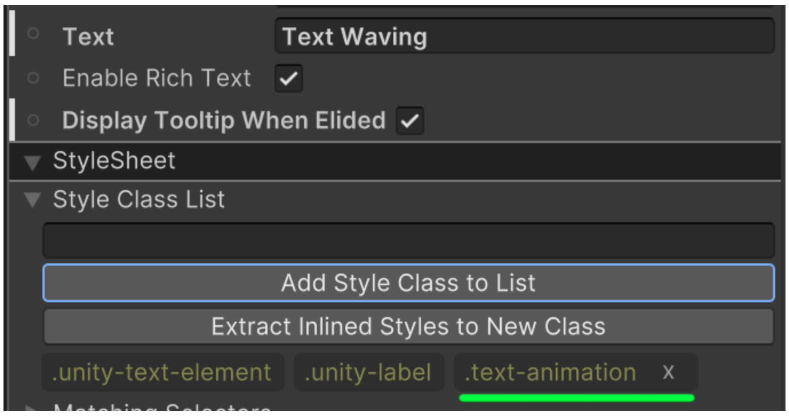
1) Enable text animations

To enable the text animation on a text element we have to make the animation system aware of that element. We do that by adding the „text-animation" USS class to the element:



text-animation

Once that class is added the text element is ready to be animated.

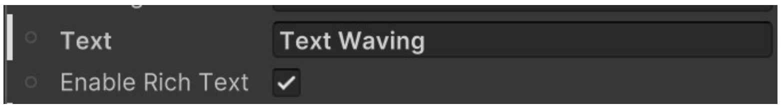

Text: Text Waving
Enable Rich Text: ✓
Display Tooltip When Elided: ✓

.unity-text-element .unity-label .text-animation X

HINT for coders:
This class name is available via the `TextAnimationManipulator.TEXT_ANIMATION_CLASSNAME` constant. There are also a lot of handy extension methods to add/remove this class (more on that later in the Scripting section).

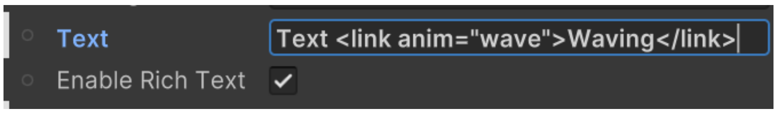
2) Configure what animation to use

Here you have two options. Either you add a <link> tag to animate only parts of the text -or- you add a typewriter USS class to animate all of the text. Our goal is to only animate the „Waving“ in our text so we choose the <link> tag option:



Text: Text Waving
Enable Rich Text: ✓

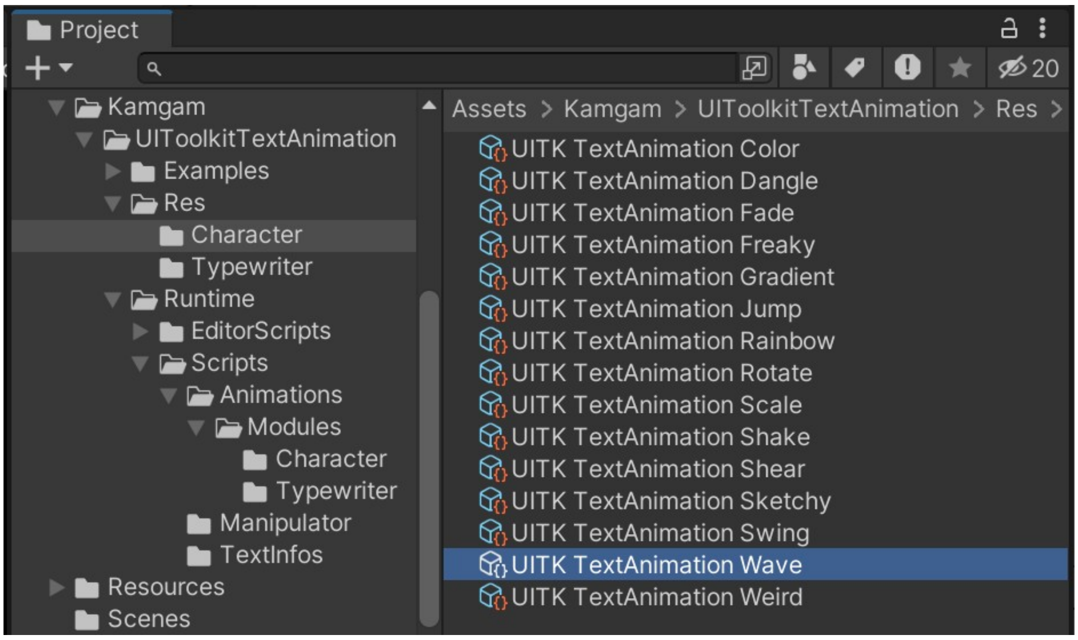
The format of the tag is <link anim="ID">TEXT</link> where ID is the id of the animation and TEXT is the text that should be animated. In our example it looks like this:



Text: Text <link anim="wave">Waving</link>
Enable Rich Text: ✓

But … how do we know what ID to use?

The ids are defined in the ScriptableObjects of the animations. You can use the predefined ones but you can also make your own (more on that later). The predefined ones can be found here:

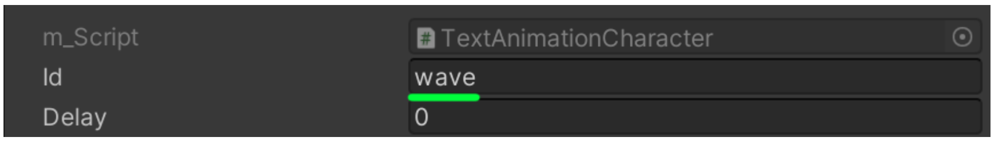


Assets > Kamgam > UIToolkitTextAnimation > Res >

UITK TextAnimation Color
UITK TextAnimation Dangle
UITK TextAnimation Fade
UITK TextAnimation Freaky
UITK TextAnimation Gradient
UITK TextAnimation Jump
UITK TextAnimation Rainbow
UITK TextAnimation Rotate
UITK TextAnimation Scale
UITK TextAnimation Shake
UITK TextAnimation Shear
UITK TextAnimation Sketchy
UITK TextAnimation Swing
UITK TextAnimation Wave
UITK TextAnimation Weird

NOTICE: Typewriter animations are in a separate folder (more on that later).

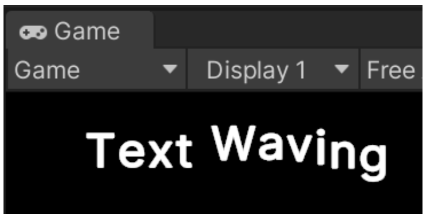
If you click on the object you will see all the details of this animation (more on that later). For now the only important thing is the Id field. That's where we will find the id we can use.



m_Script: TextAnimationCharacter
Id: wave
Delay: 0

INFO: Sadly <link> tags can not be nested (Unity`s tag parser auto-merges them). If you want to use multiple ids then you can combine them like this: <link anim="id1,id2,id3">…

If all went well (and if you have opened the scene which uses this layout) then you should already see the animation previewed in the game view:



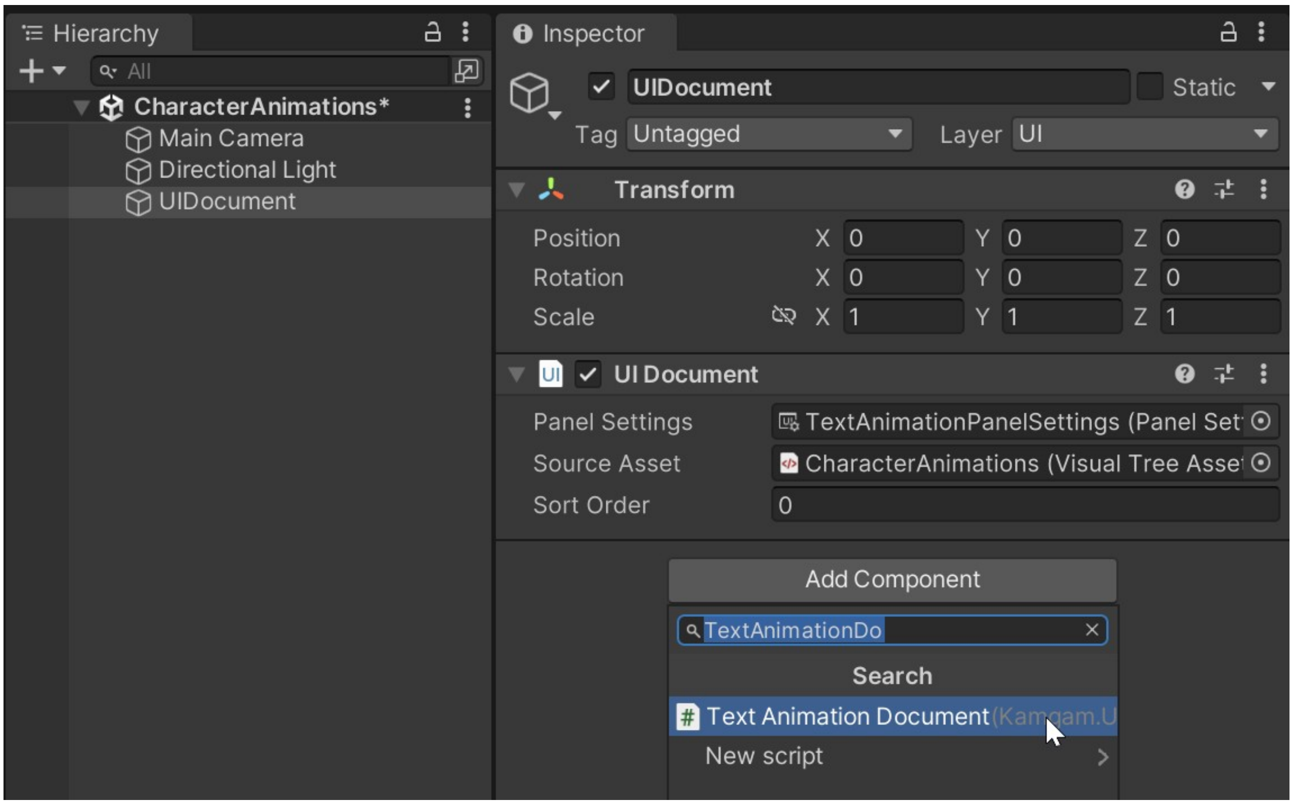
If not then maybe you have not yet added the TextAnimation system to your UI Document.

Usually this is done automatically once the first „text-animation“ USS class is added but sometimes it may not work (if you do no have the corresponding scene opened for example).

That’s where step #3 comes in.

3) Add the text animation system to your UI Document

It’s just a component called „TextAnimationDocument“. You have to add it to every UI Document you want to use the text animations in.



Hierarchy
- CharacterAnimations*
  - Main Camera
  - Directional Light
  - UIDocument

Inspector: UIDocument, Tag Untagged, Layer UI

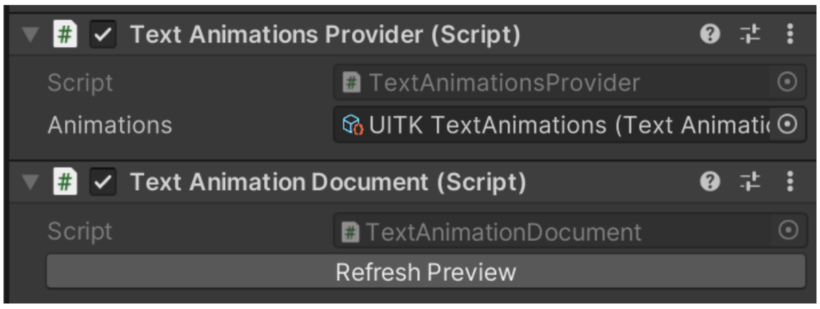
Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

UI Document

Panel Settings	TextAnimationPanelSettings (Panel Set
Source Asset	CharacterAnimations (Visual Tree Asse
Sort Order	0

Once added it should auto configure and add the TextAnimationProvider:



Text Animations Provider (Script)

Script	TextAnimationsProvider
Animations	UITK TextAnimations (Text Animati

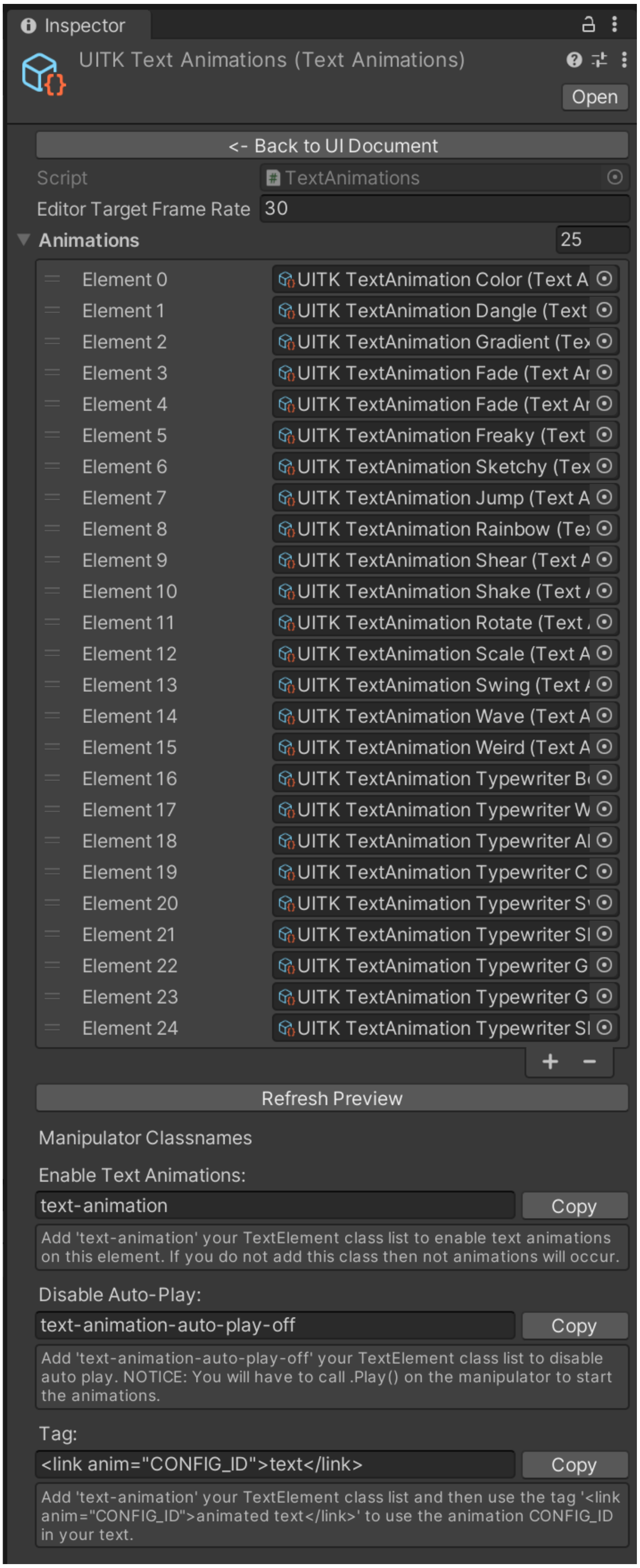
Text Animation Document (Script)

Script	TextAnimationDocument

If you click the Animations object it will take you to the global animations list.

This is where all the animations are referenced so they are usable at runtime.

Usually you will not have to touch this at all. If you create a new animation it will be added to this list automtically.


Inspector

UITK Text Animations (Text Animations)

Open

<- Back to UI Document

Script: TextAnimations

Editor Target Frame Rate: 30

Animations: 25

Element 0	UITK TextAnimation Color (Text A
Element 1	UITK TextAnimation Dangle (Text
Element 2	UITK TextAnimation Gradient (Tex
Element 3	UITK TextAnimation Fade (Text Ar
Element 4	UITK TextAnimation Fade (Text Ar
Element 5	UITK TextAnimation Freaky (Text
Element 6	UITK TextAnimation Sketchy (Tex
Element 7	UITK TextAnimation Jump (Text A
Element 8	UITK TextAnimation Rainbow (Te
Element 9	UITK TextAnimation Shear (Text A
Element 10	UITK TextAnimation Shake (Text /
Element 11	UITK TextAnimation Rotate (Text /
Element 12	UITK TextAnimation Scale (Text A
Element 13	UITK TextAnimation Swing (Text /
Element 14	UITK TextAnimation Wave (Text A
Element 15	UITK TextAnimation Weird (Text A
Element 16	UITK TextAnimation Typewriter B
Element 17	UITK TextAnimation Typewriter W
Element 18	UITK TextAnimation Typewriter Al
Element 19	UITK TextAnimation Typewriter C
Element 20	UITK TextAnimation Typewriter S
Element 21	UITK TextAnimation Typewriter Sl
Element 22	UITK TextAnimation Typewriter G
Element 23	UITK TextAnimation Typewriter G
Element 24	UITK TextAnimation Typewriter Sl

Refresh Preview

Manipulator Classnames

Enable Text Animations:

text-animation

Copy

Add 'text-animation' your TextElement class list to enable text animations on this element. If you do not add this class then not animations will occur.

Disable Auto-Play:

text-animation-auto-play-off

Copy

Add 'text-animation-auto-play-off' your TextElement class list to disable auto play. NOTICE: You will have to call .Play() on the manipulator to start the animations.

Tag:

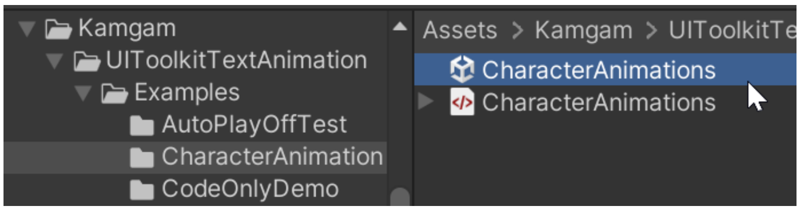
<link anim="CONFIG_ID">text</link>

Copy

Add 'text-animation' your TextElement class list and then use the tag '<link anim="CONFIG_ID">animated text</link>' to use the animation CONFIG_ID in your text.

Now we are done. If you hit play you should see the animation of the „waving" text. You should also see it in the game view without playing but sometimes Unity needs a single „Play" to catch up.

HINT: There is a demo scene in the asset that shows some of the default animations.



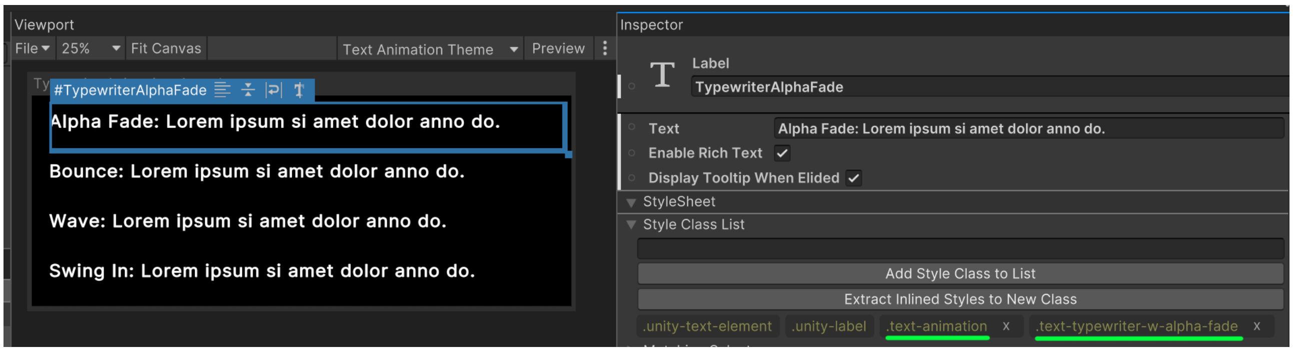
Adding typewriter animations

One thing we have not done yet is add a typewriter animation that fades in all the text. These are different from the <link> tag animations.

Character animations just loop forever and constantly animate your text.

Typewriter animations will go through your text and play the animation once for every character and then stop automatically. They are also added in a different way. We use USS classes to add them to a text element.

In fact we have to add two classes. First the default „text-animation" class to enable the element for text animation and second a typewriter class to specify what typewriter animation to use.



Alpha Fade: Lorem ipsum si amet dolor anno do.

Bounce: Lorem ipsum si amet dolor anno do.

Wave: Lorem ipsum si amet dolor anno do.

Swing In: Lorem ipsum si amet dolor anno do.

Label: TypewriterAlphaFade

Text: Alpha Fade: Lorem ipsum si amet dolor anno do.

Enable Rich Text ✓

Display Tooltip When Elided ✓

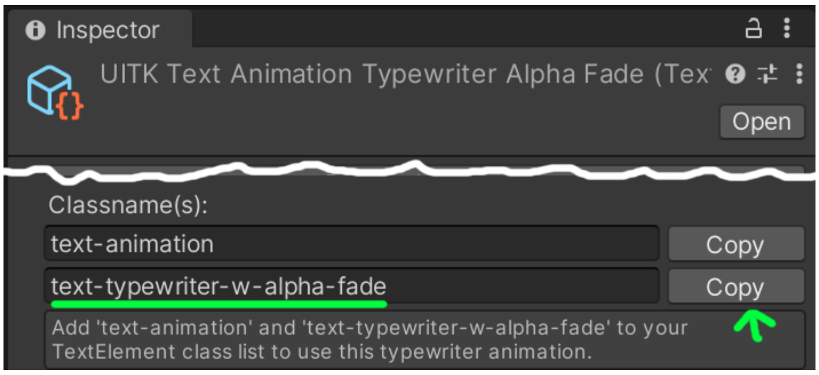
StyleSheet

Style Class List

.unity-text-element .unity-label .text-animation x .text-typewriter-w-alpha-fade x

INFO: I know that adding TWO classes is cumbersome and feels redundant. Improving this workflow is on the list of things to investigate (there are performance reasons for why it's not done yet).

The typewriter class name is a lot to type and mistakes are easily made. That's why each of the typewriter animation objects has a section at the bottom that allows you to copy the class quickly:


Inspector

UITK Text Animation Typewriter Alpha Fade (Tex

Open

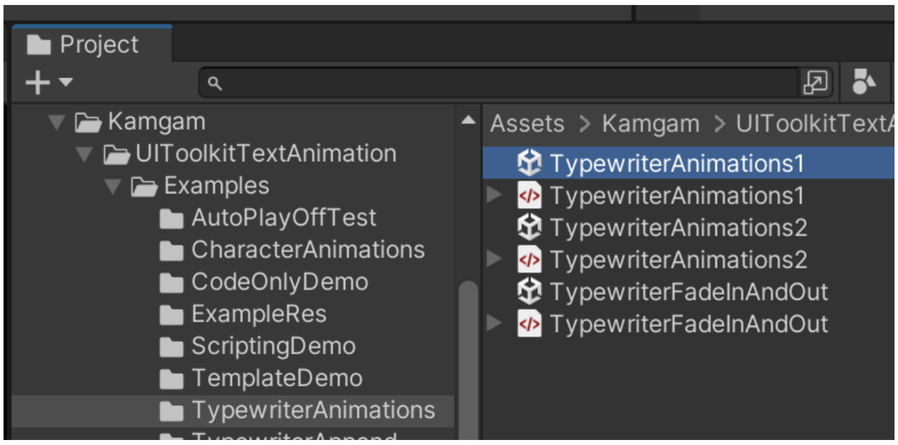
Classname(s):

text-animation — Copy

text-typewriter-w-alpha-fade — Copy

Add 'text-animation' and 'text-typewriter-w-alpha-fade' to your TextElement class list to use this typewriter animation.

HINT: There are some demo scenes that show typewriter animations:


Project

Kamgam
UIToolkitTextAnimation
Examples
AutoPlayOffTest
CharacterAnimations
CodeOnlyDemo
ExampleRes
ScriptingDemo
TemplateDemo
TypewriterAnimations

Assets > Kamgam > UIToolkitText

TypewriterAnimations1
TypewriterAnimations1
TypewriterAnimations2
TypewriterAnimations2
TypewriterFadeInAndOut
TypewriterFadeInAndOut

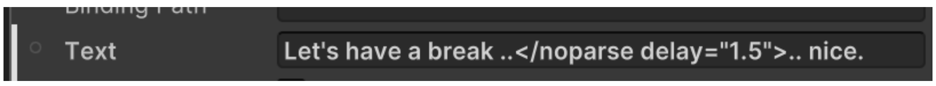
Additional Configuration Tags

Some things can be configured in the text via additional tags.

INFO: You may wonder why we are not creating our own tags for convenience. The simple answer is: Unity does not (yet) support custom text in UI Toolkit. Thus we have to piggy-back on some existing tags. One tag we use is the closing noparse (</noparse>). It is rarely used anyways so side-effects are minimal.

Delay

Syntax: </noparse delay="DELAY_IN_SEC">



Text: Let's have a break ..</noparse delay="1.5">.. nice.

You can just add it to your text where ever you want a delay and specify the duration in seconds.

NOTICE: The delays added via this tag are affected by the „Speed" parameter. If your animation has a speed of 2 then the delays will only be half as long as you have specified. That's different from the „Delay" you can specify in the animation configurations which is speed independent.

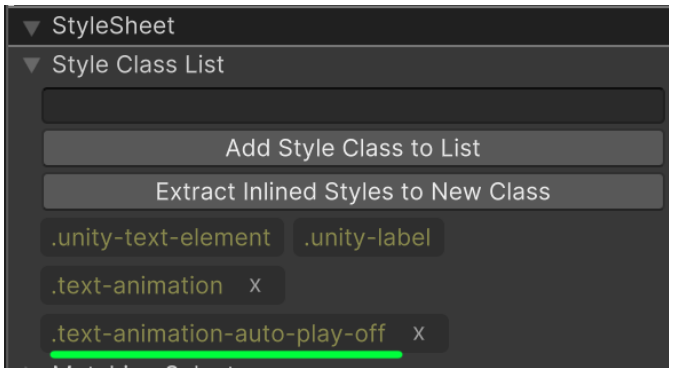
NOTICE: The delay is ignored in character animations. I am still contemplating how a delay would fit in there.

Additional Configuration Classes

Some parts can be configured on the element via extra USS classes.

Auto Play Off Class

By default any animation will start playing as soon as the element becomes visible. To disable this you can add the „text-animation-auto-play-off" class:



.unity-text-element .unity-label
.text-animation X
.text-animation-auto-play-off X

For typewriter animations this means the textfield will be completely invisible until you call .Play() on the manipulator.

HINT: There are some handy extension medthods to make this process easier:

```
// Work only if you have „Kamgam.UIToolkitTextAnimation.Extensions“ imported
label.TASetAutoPlay(false);
label.TAPlay();
```

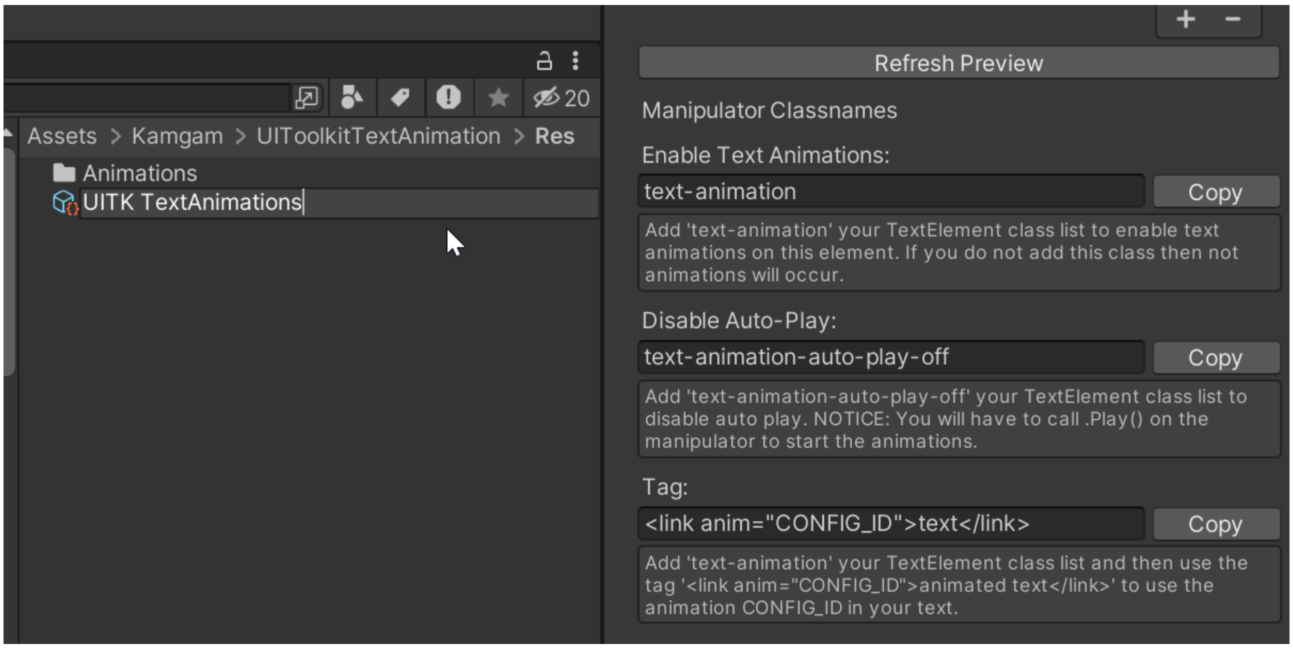
The animation class names are also constants in the TextAnimationManipulator class:

```
TextAnimationManipulator.TEXT_ANIMATION_CLASSNAME
TextAnimationManipulator.TEXT_ANIMATION_DISABLE_AUTO_PLAY_CLASSNAME
```

They are also listed on the main configs list for your convenience:

NOTICE: Since UI-Toolkit does not yet have a proper ClassNamesChanged event (source) you have to notify the animation system of any class list changes at runtime by calling:

```
element.TAUpdateAfterClassChange();
// There are extension methods to do this automatically:
```



Manipulator Classnames

Enable Text Animations:

text-animation

Add 'text-animation' your TextElement class list to enable text animations on this element. If you do not add this class then not animations will occur.

Disable Auto-Play:

text-animation-auto-play-off

Add 'text-animation-auto-play-off' your TextElement class list to disable auto play. NOTICE: You will have to call .Play() on the manipulator to start the animations.

Tag:

<link anim="CONFIG_ID">text</link>

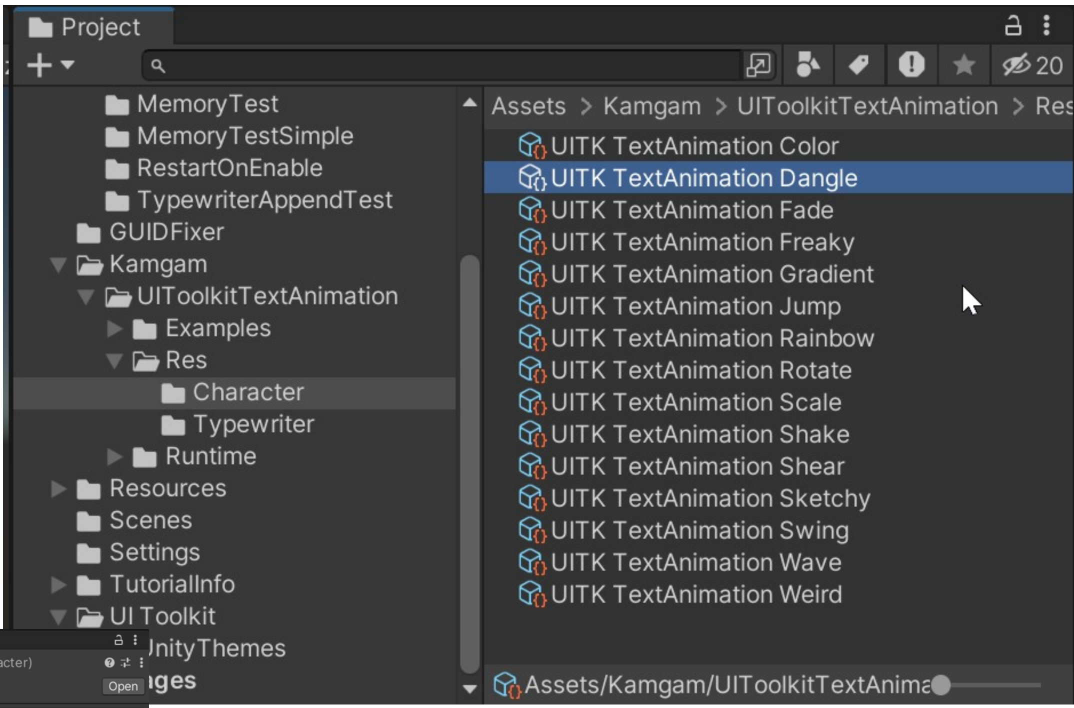
Add 'text-animation' your TextElement class list and then use the tag '<link anim="CONFIG_ID">animated text</link>' to use the animation CONFIG_ID in your text.

```
element.TAAddToClassList(className);
```

Configuring Animations

You may wonder how you can create new animations or configure them.

First let's find the location of the animation configs:

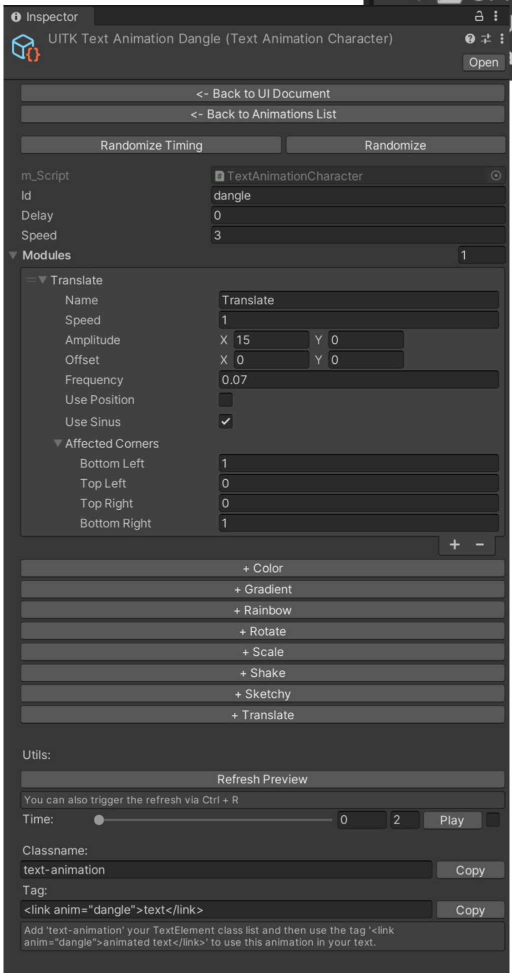


MemoryTest
MemoryTestSimple
RestartOnEnable
TypewriterAppendTest
GUIDFixer
Kamgam
UIToolkitTextAnimation
Examples
Res
Character
Typewriter
Runtime
Resources
Scenes
Settings
TutorialInfo
UI Toolkit
UnityThemes

UITK TextAnimation Color
UITK TextAnimation Dangle
UITK TextAnimation Fade
UITK TextAnimation Freaky
UITK TextAnimation Gradient
UITK TextAnimation Jump
UITK TextAnimation Rainbow
UITK TextAnimation Rotate
UITK TextAnimation Scale
UITK TextAnimation Shake
UITK TextAnimation Shear
UITK TextAnimation Sketchy
UITK TextAnimation Swing
UITK TextAnimation Wave
UITK TextAnimation Weird

Assets/Kamgam/UIToolkitTextAnima

If you click one of them you will see a lot of configurable parameters in the inspector



UITK Text Animation Dangle (Text Animation Character)

m_Script	TextAnimationCharacter
Id	dangle
Delay	0
Speed	3
Modules	1
Translate	
Name	Translate
Speed	1
Amplitude	X 15 Y 0
Offset	X 0 Y 0
Frequency	0.07
Use Position	☐
Use Sinus	✓
Affected Corners	
Bottom Left	1
Top Left	0
Top Right	0
Bottom Right	1

Utils:

You can also trigger the refresh via Ctrl + R

Time: 0 2

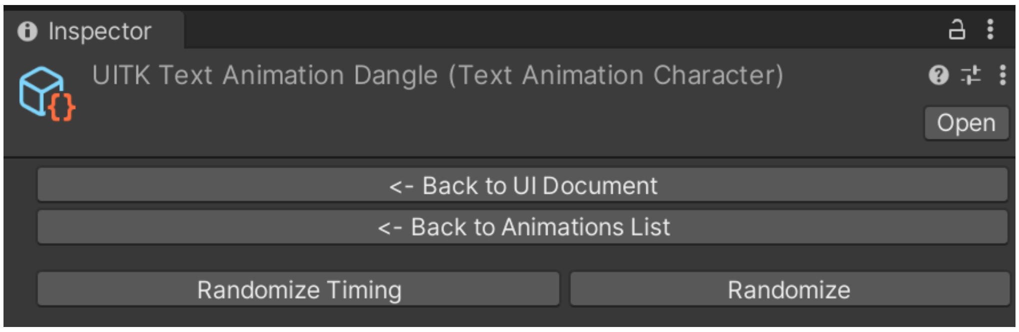
Classname:
text-animation

Tag:
<link anim="dangle">text</link>

Add 'text-animation' your TextElement class list and then use the tag '<link anim="dangle">animated text</link>' to use this animation in your text.

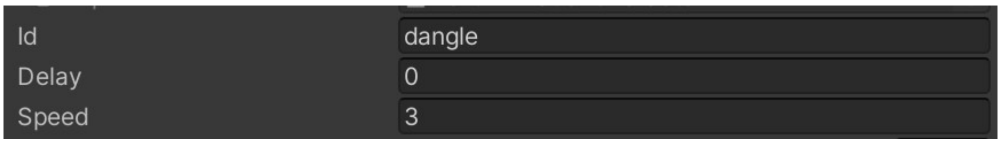
I know it looks a bit daunting at first but don't worry I will walk you through it step by step.

First are some helpful button for convenience. The „<-“ buttons will help you to navigate between the scene and the configs easier.


Inspector
UITK Text Animation Dangle (Text Animation Character)
Open
<- Back to UI Document
<- Back to Animations List
Randomize Timing
Randomize

The „Randomize“ button are useful for prototyping. If pressed they will randomize the parameters of the config (the do support Undo/Redo so don’t be afraid to give them at try later).

In the next section we have the most important field, the „Id“. This is what you have to use in your animation tags or classes to tell the system that you want it to use this animation.

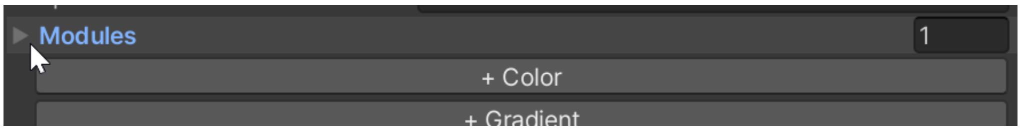


Id dangle
Delay 0
Speed 3

The delay is the starting delay of the animation and the speed, well it’s how fast your animation will be playing.

NOTICE: The speed does <u>not</u> influence the start delay.

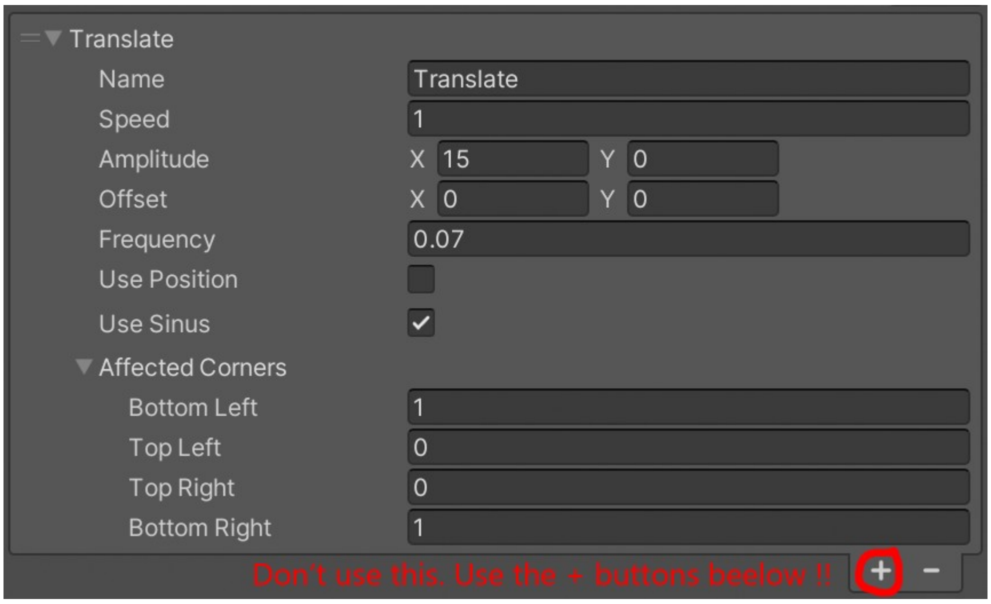
Next are the modules. If the are not folded yout yet please click the „> Modules“



So what the heck are modules?

Simply put, this is where the actual animation is done. Each module takes one character and modifies the position and/or color.

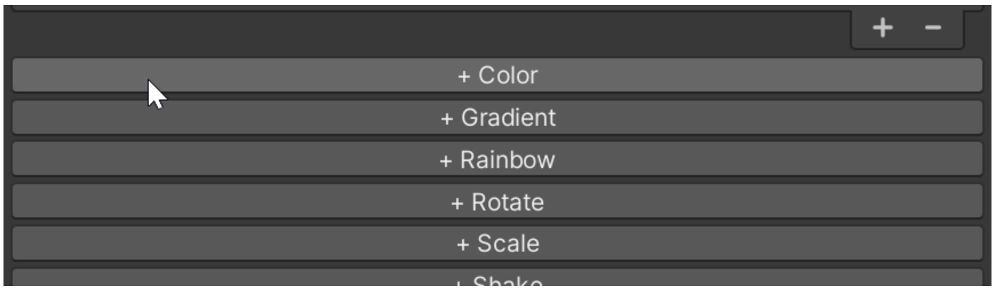
If you have clicked the modules you should see one in the inspector. In this case it is a „Translate" module which means it will translate the character (hence the name).



Translate

Name	Translate
Speed	1
Amplitude	X 15 Y 0
Offset	X 0 Y 0
Frequency	0.07
Use Position	☐
Use Sinus	✓
Affected Corners	
Bottom Left	1
Top Left	0
Top Right	0
Bottom Right	1

You may have noticed the „+ …" buttons at the bottom. These are all the different animation modules you can add to this animation.

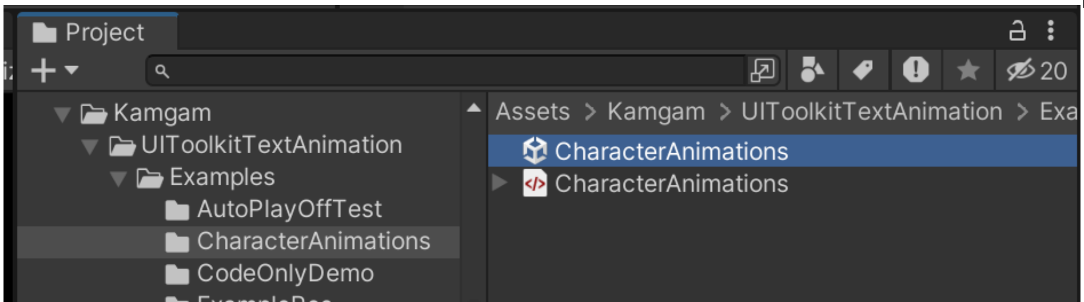


HINT: You can use the same module multiple times.

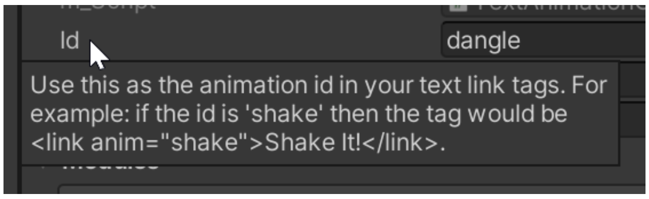
HINT for coders: You can create custom modules easily by deriving from the TextAnimationCharacter or TextAnimationTypewriter class. New buttons will appear automatically after compilation (it fetches the new types from the assembly via reflection).

Now, explaining all the parameters of each module may be a bit cumersome to read. I think it is better if you just give it a try and see what happens.

HINT: There are some demos included. Go check them out and play around with the parameters.

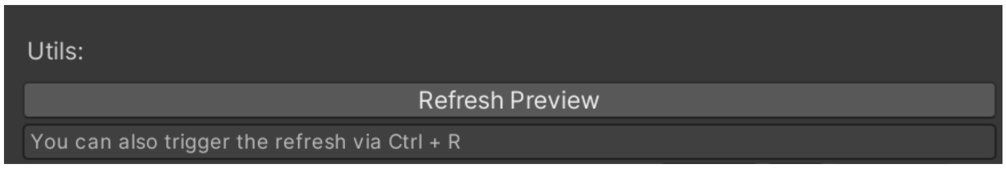


HINT: Most parameters have Tooltips if you hover over them. I did my best explaining what each parameter does there.


Id dangle

Use this as the animation id in your text link tags. For example: if the id is 'shake' then the tag would be <link anim="shake">Shake It!</link>.

Finally the Utils section. Since the Unity Editor often re-creates the UI Toolkit elements at edit time it happens that the animation configs get disconnected from the game view. To fix this (if you notice your changes have no effect) you can press the „Refresh Preview" button. I did my best to automate this but still. It is very useful.


Utils:

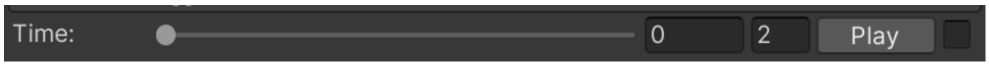
Refresh Preview

You can also trigger the refresh via Ctrl + R

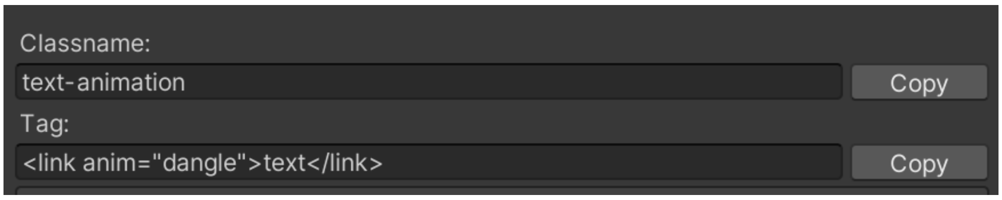
The time controls:

The slider will make the animation freeze at a certain point in time (very handy for tweaking animations). The „2" on the right is the time range. You may have to increase it for longer animations. The play button on the right will restart the animation an play it.

If you tick the box to the right of it then it will repeat playing every # seconds (where # is the „2" value from the field before). Very handy while tweaking parameters so you don't have to go down and click Play all the time.



And last but not least we have some convenience fields for copying class names and tags (`cause I just hate typing that stuff repeatedly).


Classname:
text-animation
Copy
Tag:
<link anim="dangle">text</link>
Copy

Maybe a bit overkill for animation tags but once you have to deal with typewriter classes you will thank me for it ;-)

And that’s about it. Happy animating!!!!

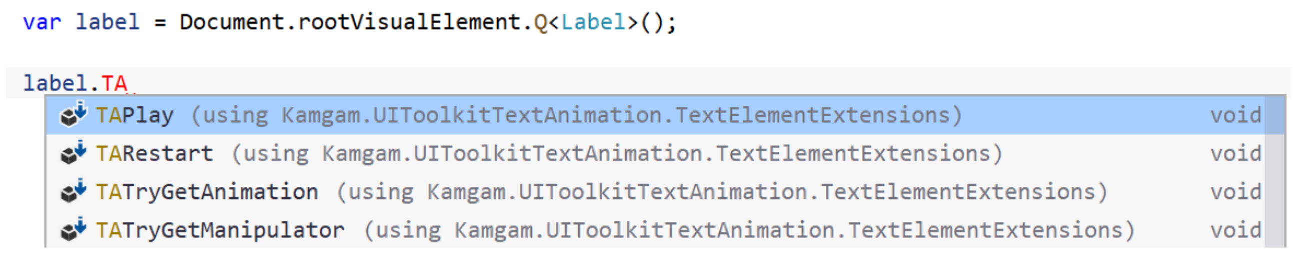
Scripting API

Extension methods

If you import the extension methods like this:

```
using Kamgam.UIToolkitTextAnimation.Extensions;
```

then you can use methods directly on your text elements. They all start with TA…



```
var label = Document.rootVisualElement.Q<Label>();

label.TA
 TAPlay (using Kamgam.UIToolkitTextAnimation.TextElementExtensions)                void
 TARestart (using Kamgam.UIToolkitTextAnimation.TextElementExtensions)             void
 TATryGetAnimation (using Kamgam.UIToolkitTextAnimation.TextElementExtensions)     void
 TATryGetManipulator (using Kamgam.UIToolkitTextAnimation.TextElementExtensions)   void
```

HINT: There are also extensions for QueryBuilders.

Async Initialization

Sadly UI Toolkit has no official API for character vertex transformation (yet) and thus we had to use a lot of internal APIs. One of the main downsides of this is that if a UI Document is shown for the first time the text information is acutally not yet available in the first frame.

This has far reaching implications. For example: You can get the manipulator of a label in frame 0 after UI Document creation BUT the animations and animation modules will not exist yet. The code has to wait for Unity to finish parsing the „<link anim=…>" tags. To work around this you have to wait until TextAnimationManipulator.HasCharacterAnimations() is true.

NOTICE: If your text does not have any <link anim…> tags then this will always be false.

NOTICE: You should not change the animations or modules on a manipulator anyways! Why? See next two sections…

Ephemeral Animations and Modules

Since the text of a label can change at any time so can the animations that are connected to that label. If the text of a label is changed then the system will clear all animations and get fresh copies based on the animations you defined in the scriptable objects (Res/.. folder).

INFO: Objects are re-used (pooled) for efficiency but for simplicity please assume they are destroyed and re-created after every text change.

Thou shall NOT change animations or modules on the manipulator!!!

Before every frame the parameters of each animation and animation module (except for time) are copied from the ScriptableObject base configs.

If you want to make changes to an animations or modules then do NOT change them on the manipulator. They will be overwritten or re-created in the next frame anyways. Instead change the parameters on the base configs. The only exception are timing related things (pause/play) because the timing is done locally.

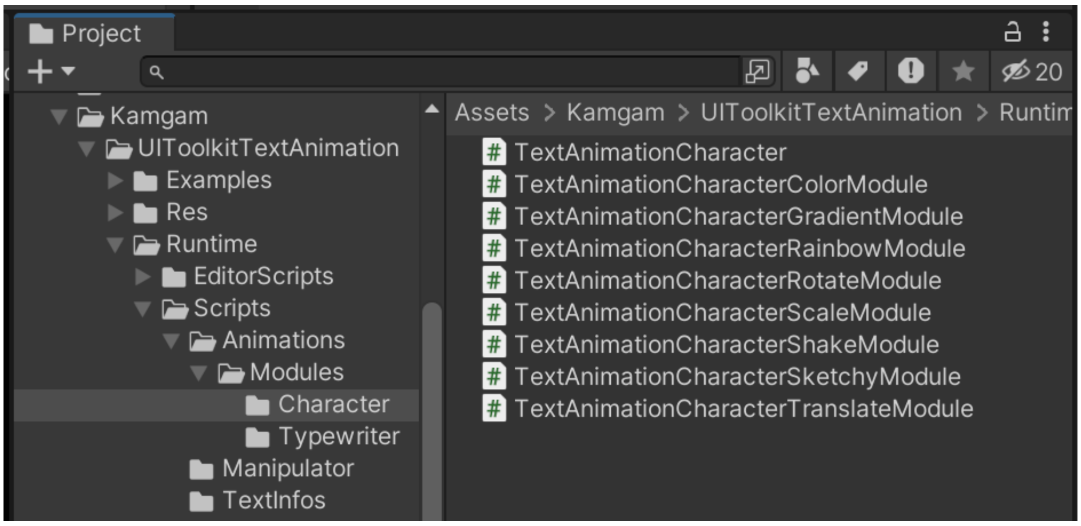
You can use the `TextAnimationsProvider.GetAnimation(string id)` method to access the base animation configs.

HINT: You can control the playback for all animations on the manipulator as well which usually is the best option.

Adding custom animation modules

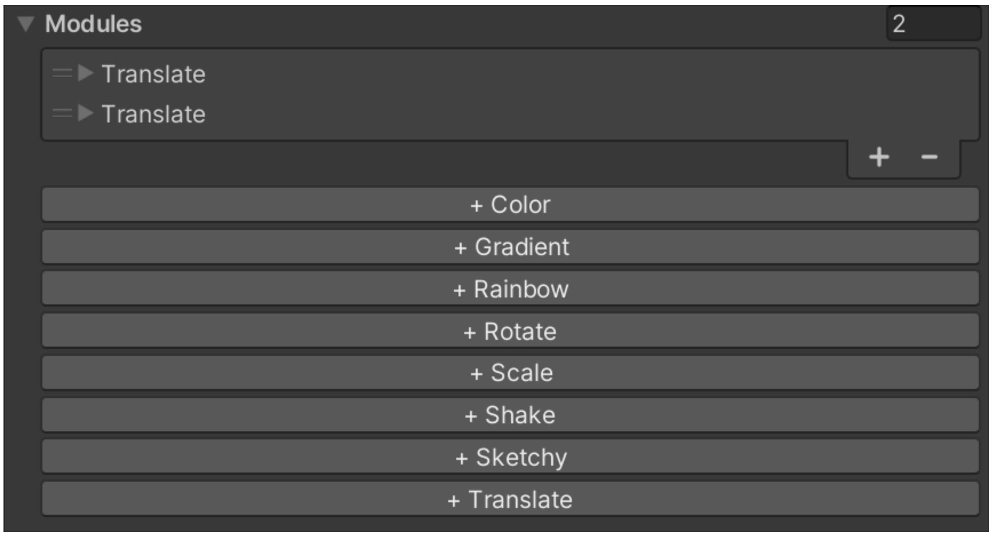
Sometimes the existing animation modules are not enough. The recommendation would be to copy the one that is closest to what you want and go from there.

You can find all the default module scripts in in the Animations/Modules/... folder:


Project
20
Kamgam
UIToolkitTextAnimation
Examples
Res
Runtime
EditorScripts
Scripts
Animations
Modules
Character
Typewriter
Manipulator
TextInfos
Assets > Kamgam > UIToolkitTextAnimation > Runtim
TextAnimationCharacter
TextAnimationCharacterColorModule
TextAnimationCharacterGradientModule
TextAnimationCharacterRainbowModule
TextAnimationCharacterRotateModule
TextAnimationCharacterScaleModule
TextAnimationCharacterShakeModule
TextAnimationCharacterSketchyModule
TextAnimationCharacterTranslateModule

There are two types of Animations: Character and Typewriter. Typewriter animations show/hide all the text in a text element and are added via USS class names. Character animations are added via the „<link anim="ID">text</link>" tag.

If you add a new class the derives from either the TextAnimationCharacter or TextAnimationTypewriter class then these will automatically show up in the „+" section of the scriptable objects (we simply use reflection to find all types and add one button for each).



HINT: Check out then TextAnimation.StaticAPI.cs file. In there you will find some static helper methods that make transforming the characters easier like: TextAnimation.GetQuadCenter(..), RotateQuadAround(..), ScaleQuad(..), TranslateQuad(..), … . There also are a lot of randomization helper methods.

Also check out the EasingUtils class. It is very handy for animations.

Updates are always based on the default position/color

One important thing to know about adding custom animations is that the „UpdateCharacter()" method is called for each quad every frame and the vertices do NOT retain their previous state.

They always start at the default position and color (unless another animation has changed them before). You should think of it as performing relative changes to the character (quad).

INFO: The reason for this is that vertex data is not always persistent in the Unity text core (it may change any time). Maybe this will change in future Unity versions once we have a proper API but for now we have to assume all vertices are reset to default before we animate them.

NOTICE: While a single animation is always relative to the original text position and color. Multiple animations (modules) are stacked and based on each others output. This enables you to easily combine them.

HINT: They are combined in the order you define in the anim attribute. Example: <link anim="wave,dangling,..">. Modules are also executed in order (top to bottom).

Randomness

Randomness in animation is tricky. In the text animation system each character position is updated by each animation in every frame.

INFO: The reason for this is that vertex data is not always persistent in text core. Maybe this will change in future Unity versions but for now we have to assume all vertices are reset to default before we animate them.

This makes randomness tricky, especially if we want to „pause" randomness in animations like shaking (position change every N frames). To do this we use the deterministic nature of the RandomNumberGenerator. We control the seed (state) so we control when the random sequence changes. That way we can keep it stable for some frames and have it spit out new random values in other frames.

Check out animations that use randomness like the Shake or the Sketchy to see how to handle randomness & persistence

Here is an example:

```
[System.NonSerialized]
protected float m_lastChangeTime = 0f;

[System.NonSerialized]
protected int m_lastFrame = -1;

[System.NonSerialized]
Random.State m_rndState;

public bool UpdateCharacter(
    TextElement element, TextInfoAccessor.AnimationTagInfo tagInfo, int characterIndex,
    float time, float unscaledTime, float unscaledDeltaTime,
    TextInfoAccessor.QuadVertexData quadVertexData, float delay, float speed)
{
    // Ensure it is reset if the animation is restarted.
    if (m_lastChangeTime > unscaledTime)
        m_lastChangeTime = unscaledTime;

    // New random seed after a delay (controlled by frequency).
    if (unscaledTime - m_lastChangeTime > 1 / (speed * Speed))
    {
        m_lastChangeTime = unscaledTime;
        Random.InitState((int)System.DateTime.Now.Ticks);
        m_rndState = Random.state;
    }

    //...
}
```

INFO: If you are an experienced dev the you may worry that these seed modifications will influence Unitys random code in other areas. Don't worry, all the random state modifications are reset after each frame was rendered so this will have no effect on your random code in the rest of your project.

Typewriter Animations

The typewriter animations work quite similar to normal (character) animations. They have slightly different parameters (quads instead of chars) and there is some logic that optimizes when each quad is updated but generally it is the same. You have a normalized progress (0 to 1) that tells you the progress of each quads visibility. Based on that progress value you can advance your animation.

Hint: If you can add the AffectedCorners parameter to your custom animations. It is very handy to achieve all sorts of effects. Also it's fun to play around with.

Easing

There is an EasingUtils class included. You can use it for all sorts of easing animations. Check out TextAnimationConfigTypewriterTranslate.cs for an example.

Pooling

To reduce memory allocations almost all objects in the text animation system are pooled. This means that objects are reused constantly. Usually this does not concern you as a user but if you want to write your own custom animation class then you have to take care of it. If you add hidden properties or fields to your config then make sure to override the Reset() method and reset those. Take a look at the existing implementations for reference.

HINT: Also don't forget to copy your values in the CopyValuesFrom() method. If you don't then you will wonder why the changes in the SO will have no effect (I speak from experience).

Editor Features

If you make your own custom animation you can also implement a Randomize() method. This should randomize all your parameters so the user has a nice and easy way of exploring the possibilities of your animation.

All the default implementations support this so you can take a look at how it's done.

HINT: The TextAnimation has many helpful static methods called „GetRandom…".

Frequently Asked Questions

What's the key difference to all the other text animation solutions?

What's the key difference to all the other text animation solutions?

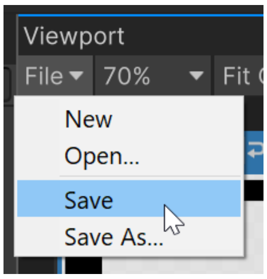
A) It's for the UI Toolkit (not uGUI)
B) It is non-destructive. Which means you can add it to your project at any time (no custom label elements or similar). It works for any TextElement.

Does it have events?

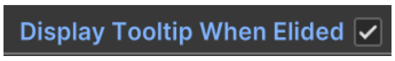
No yet, but they are right on top of the TODO list.

Adding or removing the „text-animation" class in the UI Builder window does nothing

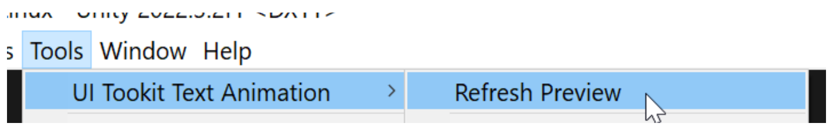
Usually it should pick up on class changes by default. If it does not then please make sure you hit File -> Save afterwards to trigger the change in the game view:



NOTICE: It may take 1-2 seconds for the game view to pick up on the change.

HINT: Instead of hitting Save you can also just update any value in the UI Builder panel to trigger the update. I like to use  because it is right above the classes and does not change any visuals effects.

HINT: You can also call a refresh it via the tools menu:



If all else fails hitting the play mode once should clear up all stuck animations.

Adding or removing the „text-animation" class at runtime does nothing?

Since UI-Toolkit does not yet have a proper ClassNamesChanged event (source) you have to notify the animation system manually after modifying the class list:

```
TextAnimationManipulator.AddOrRemoveManipulator(textElement);
```

This will then add or remove the animations as needed.

HINT: You can also use the TA extension methods like this:

```
var label = new Label();
label.AddToClassList("text-animation")
label.TAUpdateAfterClassChange();

// There are extension methods to do this automaticallyon one go:
element.TAAddToClassList(className);
```

HINT: Typewriter configs have a GetClassName() method that you can use.

Are nested tags like <link> <link></link> </link> supported?

No, sadly the Unity rich text parser does not properly report nested tags and therefore they can not (yet) be supported. As soon as that changes they can be supported. Sadly I have no info when or if that will happen.

If you want to mix multiple animations then you can list multiple ones in one link tag like this:

```
This is some<link amin="shake,rainbow">Shaking rainbow text</link>. Yay!
```

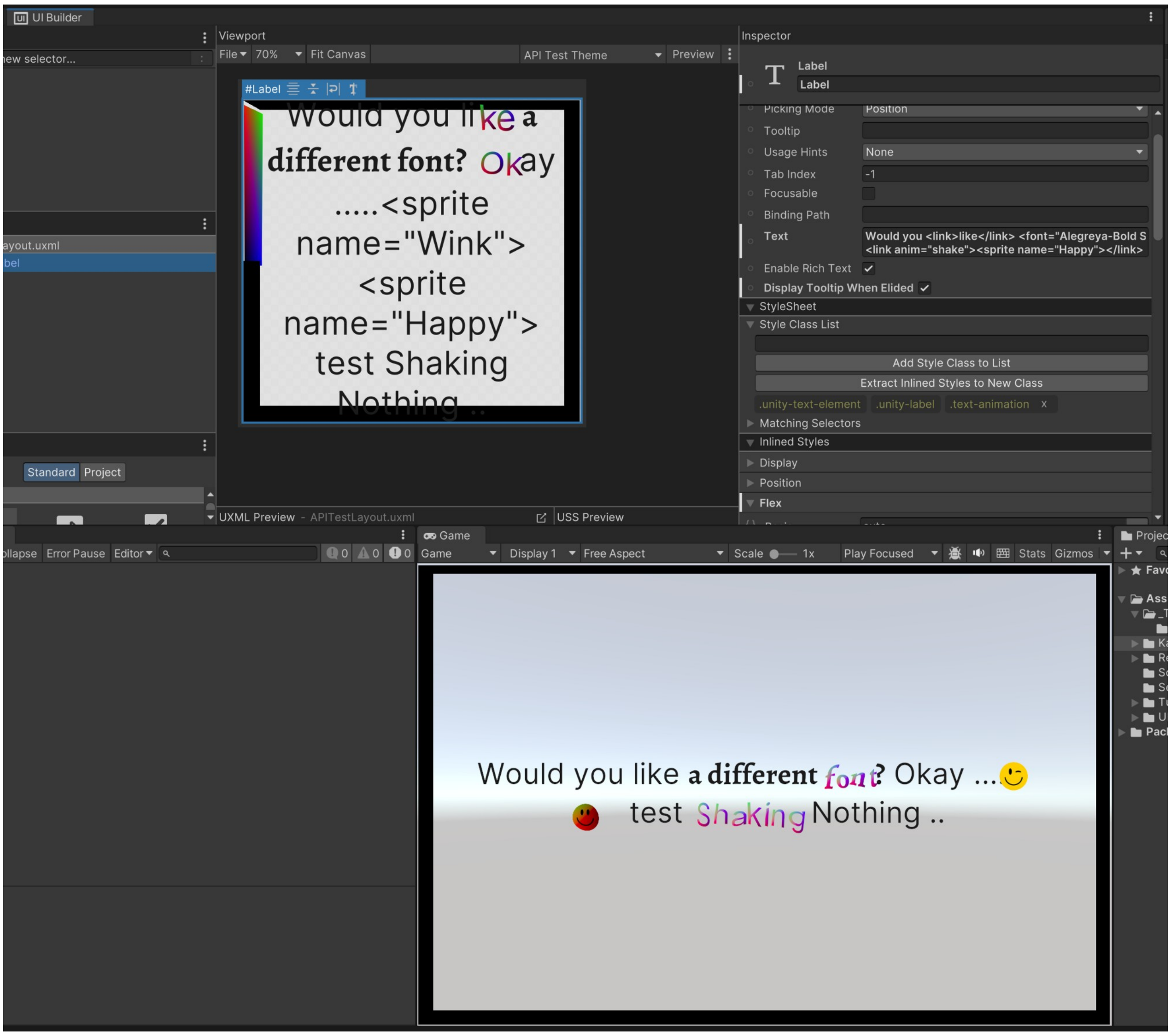
Notice the „`shake,rainbow`" part. Multiple animation config ids are separated by a comma.

The animations are not previewed in the UI Builder window

If I enable the preview in UI Builder it looks weird?

The preview in the UI Builder is disabled by default. You can enable it via the static TextAnimationManipulator.AddToUIBuilderElements flag. However, in older versions of Unity it may apply the animation to the wrong characters since some old versions of Unity do not propery support <sprite> or <link> tags.

Example:



Would you like a different font? Okay .....<sprite name="Wink"> <sprite name="Happy"> test Shaking Nothing ..

Text: Would you <link>like</link> <font="Alegreya-Bold S
<link anim="shake"><sprite name="Happy"></link>

Enable Rich Text ✓
Display Tooltip When Elided ✓

Style Class List: .unity-text-element .unity-label .text-animation

Would you like a different font? Okay ... test Shaking Nothing ..

My recommendation: Keep it turned off.

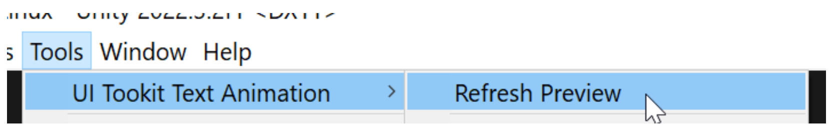
The preview in the GameView does not always work in Unity 2023

Yes, sorry about that but Unity 2023 is a mess in regards to that. It will work in play mode and in builds but sometimes in the Editor it glitches out.

Since 2023 is no longer supported by Unity itself I will also no longer invest time into it. Please, if you can, upgrade to Unity 6. Thank you for understanding.

If all else fails, hit the play mode once. That should clear up any stuck animations.

HINT: You can also call a refresh via the tools menu:



Is Advanced Text Generator supported?

Not yet, but once it is the standard in Unity I will take a look.

Is right-to-reft text supported?

Sadly no since the default UI Toolkit test implementation does not support it either. There is a push for better support in Unity 6+ with the Advanced Text Generator but this is not yet the standard tool. Once this is the default I may look into supporting it.

Links:

* https://discussions.unity.com/t/announcing-full-rtl-language-support/1544214

* https://docs.unity3d.com/6000.0/Documentation/Manual/UIE-advanced-text-generator.html

If I append text to a textfield with a typewriter then the old character animations are completed immediately.

While it's not pretty it is „by design" because the typewriter does not store individual character animation states (progress). Instead it stores which characters have been completed and if new ones are added it will fast-forward to start animating the newly added text which automatically completes the other characters. If your per-character animations are short enough this will hardly be noticable.

Appending text to a typewriter that has completed is not animating.

This happens if you have AutoPlay disabled on the element. If you have then you need to calls Play() on the manipulator of the element manually since it is not playing by default

Are the animations memory allocation free?

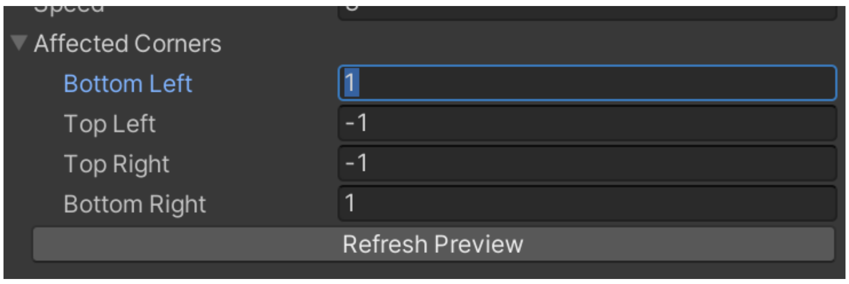
Sadly no since the text animation has to trigger a text mesh re-creation (due to a lack of proper APIs to access text vertex data). This generates some GC allocations in Unitys internal code (~0.5 kBytes per frame). This also depends on the used Unity version. The text animation code itself is GC free after initial allocations for new animations but rebuildig the text via Text Core is not. This will of course be optimized once there is an official API or other changes allow me to get rid of it (please check the profiler on your project).

Changes to my custom config parameters are ignored.

If you have copied the animation config class then you may have forgotten to update the CopyVValues() method. It has to copy all the custom parameters to propagate changes. Also notice the typecast before assigning the values.

Changes to config values in the inspector have no effect in Editor Mode?

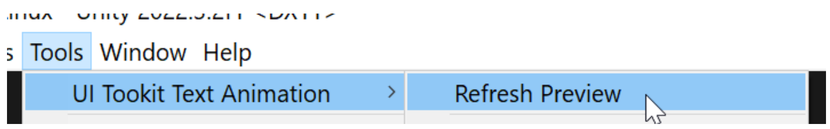
I this happens then the configs have lost the connection to the modifiers. This can happen in the editor (especially after restarting Unity or re-compiles). To fix this please click the „Refresh Preview" button. It will refresh all modifiers and reconnect them to the configs.



Affected Corners

Bottom Left	1
Top Left	-1
Top Right	-1
Bottom Right	1

You can also call it via the tools menu:



Typwriter and Character configs not found by id

The „Id“ of the configs have to be unique across all configs (typewriters and animations). If you have duplicate ids then the first that was created will be used and the other one will be ignored. You may have to press „Refresh Preview“ after chaning Ids for them to take effect.

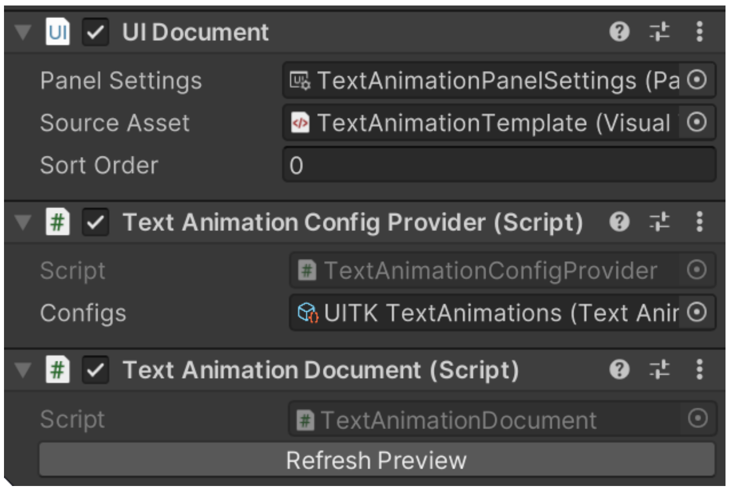
That's also why all the demo typewriter ids are prefixed with a „w-“ to ensure unique ids.

I have added the classes but nothing changed?

Press the „Refresh Preview“ button on the config or under Tools > UI Toolkit Text Animation > Refresh Preview.

The animations shows in the Editor but not at play-/runtime?

Did you add the „TextAnimationDocument“ to your UI Document? The Editor has some special code to preview the animations but at runtime it only works if that component is added to the UI Document.


UI Document
Panel Settings: TextAnimationPanelSettings (Pa
Source Asset: TextAnimationTemplate (Visual
Sort Order: 0
Text Animation Config Provider (Script)
Script: TextAnimationConfigProvider
Configs: UITK TextAnimations (Text Anir
Text Animation Document (Script)
Script: TextAnimationDocument

Are templates supported?

Yes, though in the UI Builder you can not easily change the classes of instances so you may have to do that via code. Though that's true for any template and is not text-animation related.

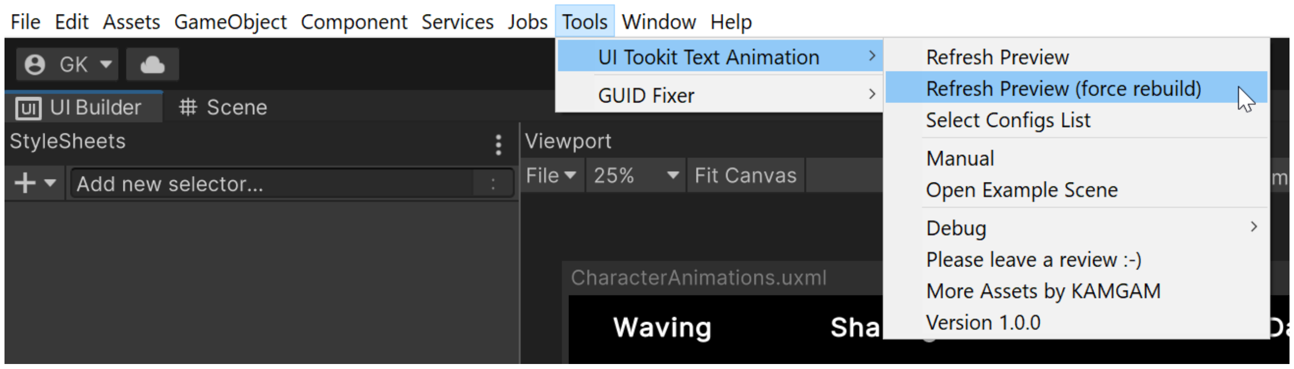
There is a demo scene called „TemplateDemo“ that uses templates.

The animation does not restart autmoatically if the „visibility“ style is changed.

Yes, sadly there is no reliable event (yet) in UI Toolkit to detect this. However, it will restart if you change the „display“ style. HINT: You can always use the TARestart() extension method to start it.

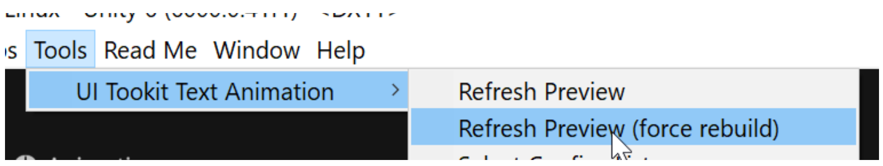
The preview in the game view looks weird while editing.

It rarely happens but usually you can fix it by opening the UI Builder and call „Refresh Preview (force rebuild)“. Also in Unity 6 this should no longer happen.



After adding delay tags the delays are inserted at the wrong time.

It sometimes happens that in the editor the textfield delays do not refresh properly. You can fix it by manually forcing a refresh or if all else fails by hitten the play mode once.



It is under investigation.

The animations don’t show on my Button?

The most likely cause is that „Enable Ritch Text“ is disabled on your button. Enable it to and press refresh and you should start seeing your animation.