

Documentation & Quick Start

Welcome to **Text Animator for Unity!**

Thanks for purchasing my asset, thanks to you I'll be able to provide new updates, release more content and, well... survive <3

This asset was uploaded by
<https://unityassetcollection.com>

Useful Links

- **Documentation:** <https://www.textanimator.febucci.com/docs/>
Learn everything about the plugin in this short and simple documentation, containing images and GIFs for each use case.
- **Roadmap:** <https://www.textanimator.febucci.com/roadmap/>
Discover which features/tasks are currently in the work, and what's planned for the future.
- **Support** request: <https://www.textanimator.febucci.com/support/>
Look at the most common problems/question or ask for help directly.



The **example scenes** in this plugin will show you many features and use cases that you will find useful. They're located in the "Febucci/Text Animator/Example" folder, and you can delete it once you don't need them anymore.

Have fun using Text Animator for Unity!

*Please remember to write a review for the asset, it takes one minute but it helps.
Have a lovely day!*

Plugin Overview

If you're without an internet connection to read the online documentation, here's a quick overview to let you start right away using "TextAnimator for Unity".

Even if the asset contains multiple examples with each feature, you can learn more about Text Animator in this page as well.

Useful Links

Plugin Overview

1. [How to implement Text Animator](#)
 2. [How to add Effects to your texts](#)
 - [Behavior Effects](#)
 - [Appearance Effects](#)
 3. [Text Animator Players \(typewriter\)](#)
 - [A\) Via Code \(Recommended\)](#)
 - [B\) Via the "Easy Integration"](#)
 4. [Events and Actions](#)
 - [Events](#)
 - [Actions](#)
-

1. How to implement Text Animator

As the first step, you need to implement Text Animator in your project.

1. **Import the Text Mesh Pro package** (if not present): Browse the Project manager (*Unity* → *Window/PackageManager*), download and install the package named *TextMeshPro*.
2. **Import the Text Animator package**
3. **Add a TextAnimator Component:** add a `TextAnimator` component on the same GameObject that has a `TextMeshPro` component in your scene.

✓ *Your text is now ready for effects.*

2. How to add Effects to your texts

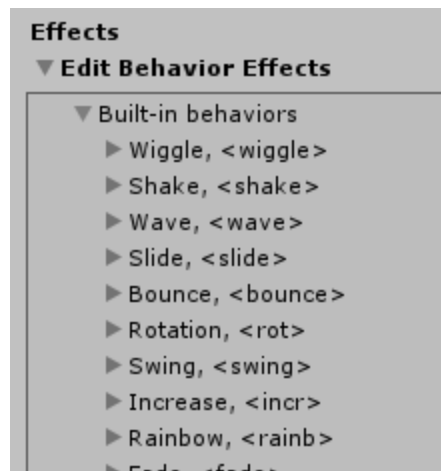
You can apply effects to your texts by using rich text tags.

Example, write in the TextMeshPro component: "It's `<wave>` windy `</wave>` out there".

Effects are divided in two **categories**, based on their functionality/application. They consist on:

- Behavior effects.
- Appearance effects.

You can read the full list of available built-in behaviors/appearances in the TextAnimator component itself (more than the online docs).



Behavior Effects



Behavior Effects animate letters continuously during time, after the letter has been shown.

You can apply behavior effects to your texts by using rich text tags. A Behavior Effect tag looks like this `<tag>`.

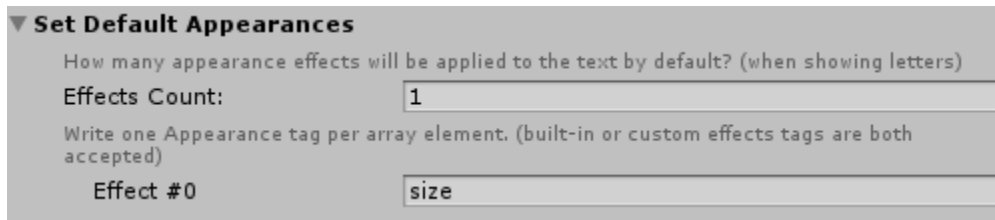
Example: "I'm `<wiggle>` joking `</wiggle>`. Now I'm `<shake>` cold `</shake>`".

Appearance Effects



Appearance Effects animate letters only when they're appearing on screen. For this reason, they're mostly used in combination with the typewriter, which shows letters one after another.

→ You can set which effect(s) will be applied to all letters by default in the `TextAnimator` component, without having to write appearance tags for each text.



▼ Set Default Appearances

How many appearance effects will be applied to the text by default? (when showing letters)

Effects Count:

Write one Appearance tag per array element. (built-in or custom effects tags are both accepted)

Effect #0

👍 If you don't want any appearance effect, simply set the effects' count to zero.

→ You can also apply Appearance Effects to specific parts of your text by using rich text tags, overriding the default Appearance Effect (if any). An Appearance Effect tag looks like this `{tag}`.

If you close an appearance tag, the letters will be affected again by the default effect (if any is set).

Example: "default default `{fade}` fade fade fade `{/fade}` default default"

3. Text Animator Players (typewriter)



You can use `TextAnimatorPlayers` to show letters dynamically (like a typewriter), choosing different pauses for any kind of characters (punctuation, letters, [...]) and more.

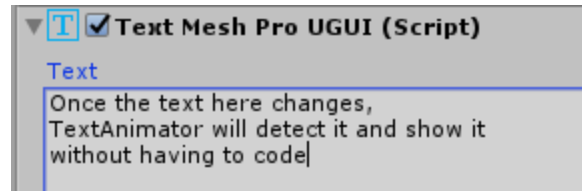
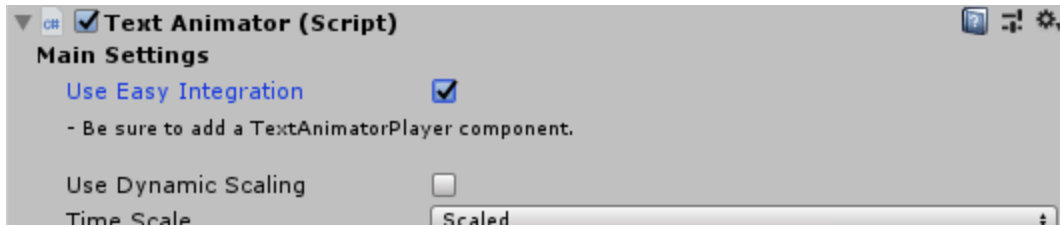
You can tell TextAnimator to use the typewriter in two ways.

A) Via Code (Recommended)

Replacing "`tmproText.text = textValue;`" with `textAnimatorPlayer.ShowText(textValue);` in your code (referencing a `TextAnimatorPlayer` component).

B) Via the "Easy Integration"

Enabling the "`Use Easy Integration`" option in the TextAnimator component and let the plugin automatically check for text changes in your TextMeshPro's component.



You can choose when to start the typewriter (the default setting is "automatically").

You can also track when it finished showing letters by subscribing to the "OnTextShowed" event.

4. Events and Actions

If you have the typewriter enabled, you can use events and actions.

Events



Events are special tags that let you send messages (string) to any listener script, once the typewriter has reached a specific part of the text.

You can write events in your text by using rich text tags (they're case sensitive!).

Event's messages are preceded by a question mark, like this: `<?eventMessage>`.

Example: To call an event named 'shakeCamera', write: `<?shakeCamera>`

Listening to events example:

```
//Inside your script

public TextAnimator textAnimator;

//Manage the event subscription

private void Awake()
{
    textAnimator.onEvent += OnEvent;
}

private void OnDestroy()
{
    textAnimator.onEvent -= OnEvent;
}

//Do things based on messages
void OnEvent(string message)
{
    switch (message)
    {
        case "something":
            //do something
            break;
    }
}
```

Actions



You can perform actions once the typewriter reaches a specific position in the text.

Example: waiting for X seconds or waiting for the player input.

You can add actions in your text by using rich text tags.

Actions' formatting follows this formula: “<actionID>” or “<actionID=attribute1,attribute2,...>” for eventual parameters/attributes.

▲ Actions are case sensitive, <waitfor> and <waitFor> will produce different results.

Some built-in actions:

- **Wait for Seconds**

- Tag: waitfor
- Description: Waits for X seconds before continuing to show the text.
- Attributes: float (wait duration)
- Usage Example: <waitfor=1.5>

- **Wait for Input**

- Tag: waitinput
- Description: Waits for the player input"
- Usage Example: <waitinput>

- **Speed**

- Tag: speed
- Description: Multiplies the typewriter speed
- Attributes: float (speed multiplier)
- Usage Example: <speed=2>

Here was an overview of the main "Text Animator for Unity" features.

You can discover more in the online documentation, including custom effects, external plugin integrations and more.

Have fun using Text Animator for Unity!

I'm always available in case you need any help (support page at the top). Cheers!