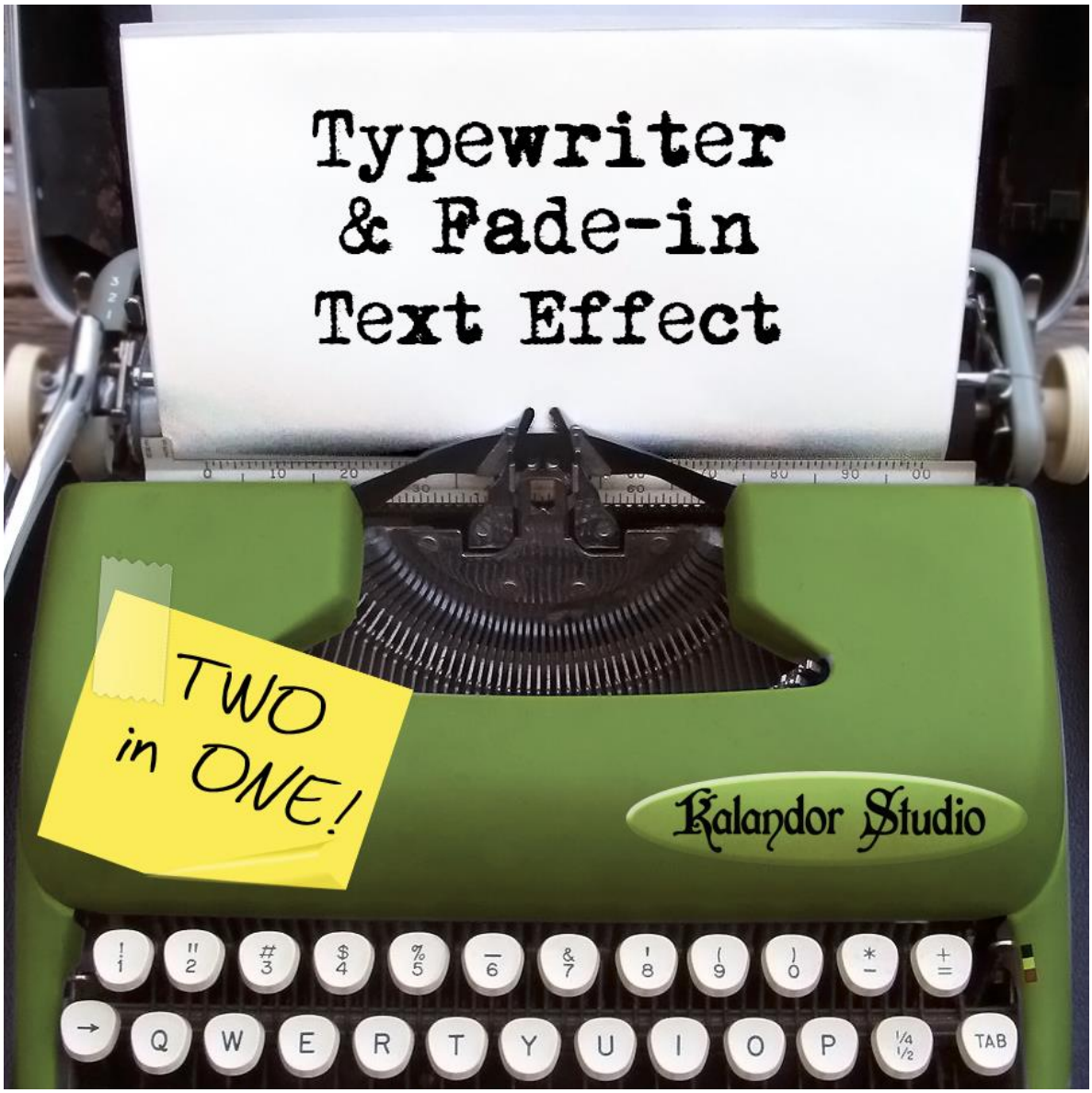


Documentation

Version 1.3.6

Table of Content

1. About Typewriter & Fade-in Text Effect

Who said that displaying text has to be boring? Show dialogues, mission objectives or even massive story fragments gradually with this easy-to-use text effect!

> *Gnome scientists have been struggling for centuries with a problem: no one bothered to read their wall-of-text publications as those were just too long, and intimidated everyone. Then they came up with the idea to gradually display text as the reader goes along, thus successfully keeping their attention. And it worked miraculously, simply because the human mind is curious by nature, and wants to know what's ahead!*

Main features:

- Display each character gradually or let them fade-in over time
- Rich Text support (letting you use multiple font styles and sizes on the same text)
- Highly customizable (speed, timescale independence etc.)
- Set up without scripting using the new Unity GUI-system
- Compatible with all GUI solutions thanks to its simple callback system*
- Thorough documentation, 3 sample scenes demonstrating some practical usage of the effect (a dialogue scenario, text fading, and working with other GUI-systems)
- Play and randomize sound effects and pitch
- Place triggers to invoke events anywhere along the text
- Utility scripts for further working with Rich Text Format
- Built-in Rich Text Editor
- All sources included (written in C#)
- Supports Unity 4.6+ and 5.0+ as well

* (Please note that Automatic Content Wrapping and Auto-Start only works with the new Unity GUI, and to use this asset to its full potential you need to enable Rich Text formatting in your text. It also means that if your custom GUI system doesn't support RTF, the fade-in effect won't be able to operate.)

A YouTube tutorial demonstrating how to set-up text fading in 2 minutes: http://youtu.be/fD-yoTiu_ns

Support and feedback

Should you have any questions, issues or suggestions regarding this plugin, please feel free to contact us at Kalandor Studio via Email: devsupport@kalandorstudio.com

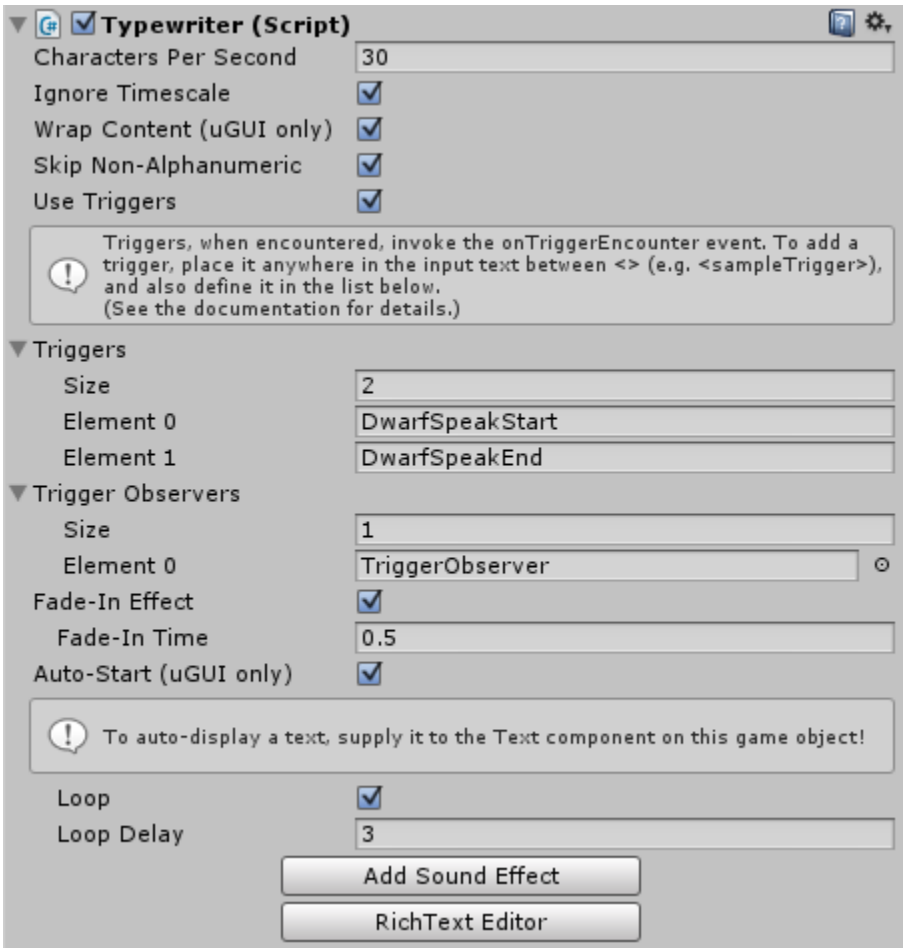
If you find this plugin to be useful, please leave a review or rate it on the Asset Store!

Thanks for using Typewriter & Fade-in Text Effect!

2. Usage

Step 1 - Add the typewriter component to a game object

First you need to add the Typewriter script to a game object. If you're working with any other GUI system than the new Unity GUI (uGUI), you can place it on any game object. However, if using uGUI you should add Typewriter to the Text game object which you're planning to display the text with, or alternatively you can just add the Typewriter prefab (*Found in Assets/Typewriter Text Effect folder*) to the scene, which already contains a Text component.



Below are all the Typewriter options explained:

Characters Per Second	How many characters should be displayed in a second?
Ignore Timescale	Decides whether the effect should be real time, or if changing the *TimeScale* should also affect its speed.
Wrap Content	Whether the Typewriter should handle the wrapping of the text in cases when it would otherwise overflow horizontally. Only works with uGUI: in case of working with other GUI systems you need to write your own code to detect and handle when a line of text would overflow.
Fix Text Width	Set to true if you'd like the Text component to always have a fix size instead of growing as the text appears.

Skip Non-Alphanumeric	Whether non-alphanumeric characters (like symbols, periods etc.) should be displayed instantly.
Use Triggers	Check if you'd like to use triggers in the text. You can learn more about triggers here.
Triggers	The list of triggers to look for in the text.
Trigger Observers	Game objects that should receive the OnTypewriterTrigger_Encountered(string trigger) message when a trigger is encountered.
Auto-Start	If you'd like the Typewriter to start as soon as it's activated, specify these options.
Text	The text to display when starting automatically.
Loop	Should the Typewriter restart upon finish?
Loop Delay	How much time should the effect wait before restart? Requires Loop to be checked.
Add Sound Effect	Adds the TypewriterSound component to the game object. Learn more about this feature here.

Step 2 - Initialize the typewriter

There are three ways you can start the effect.

- **Automatically (with uGUI only):** By checking Auto-Start and specifying a text to display you can start the effect without writing a single line of code!
- **Calling the BeginDisplayMode method (with uGUI only):** This starts the Typewriter effect, displaying the text using the UnityEngine.UI.Text component of the same game object.
- **Calling the BeginCallbackMode method (for any GUI system):** Begins the Typewriter effect, but instead of displaying the text in a specified label, it uses events to notify the caller about every change in the text, giving you maximum control in handling the text. Call this method and listen to the event *onTextChanged* (see the Wall-of-Text example for sample code).

Step 3 (optional) - Specify sound effects

It is easy to play sound effect while the typing is in progress. This can be achieved in two ways:

- **Adding the TypewriterSound component:** Clicking the "Add Sound Effect" button on the *Typewriter* component adds a *TypewriterSound* component to your typewriter: specifying its audio clips is all it takes to configure it;
- **Passing clips as parameter:** just specify the *AudioClip* parameter when calling `Typewriter.BeginDisplayMode()` or `Typewriter.BeginCallbackMode()` methods. By doing so, the Typewriter will automatically add and initialize the TypewriterSound component.

TypewriterSound works with an AudioClip array, which means that if you specify more than one clips it will randomize them, making them sound less monotonous. You can also specify the *Pitch Random Factor* for every clip, making their pitch value to be randomized by the given value (for example a random factor 0.4 will result a value between 0.8 and 1.2.. To avoid pitch randomization, simply leave *Pitch Random Factor* at the default value of 0.

Step 4 (optional) – Triggers

Triggers are special tags placed in the text, that – when encountered – invoke events that notify you, making it possible to run your event handler when the text processing reaches the trigger. This can be useful for example to play a sound effect when the display reaches a certain part, to play an animation etc. (To see Triggers in action, check out the Fade-in Example!)

4.1 - Place the trigger in the text

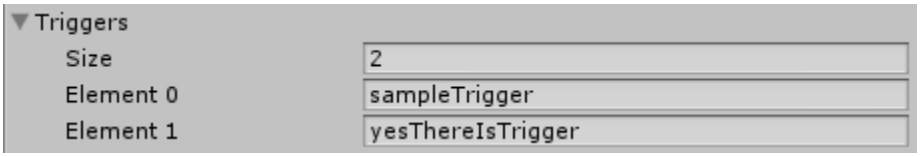
To add a trigger, place it anywhere in the input text just like an HTML tag, between <> characters (e.g. <sampleTrigger>). For example:

> *This is a sample text<sampleTrigger> which contains a trigger in the middle, and I suspect there may be another in the end<yesThereIsTrigger>!*

Please note that the trigger expressions could have been anything else, the only criteria is that they shouldn't be a valid Rich Text expression.

4.2 - Add the trigger to the list of triggers

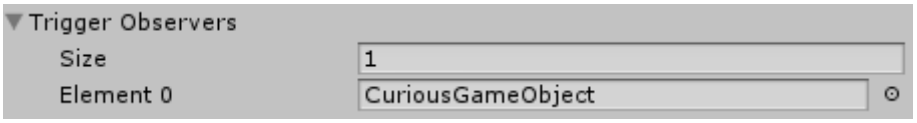
Now that you've placed the triggers in the text, you should add them to the list of Triggers:



4.3 - Specify a trigger observer

All that's left is to make sure someone actually gets notified when a trigger is encountered!

One way to achieve this is to specify the Trigger Observers: game object with a method called `OnTypewriterTrigger_Encountered(string trigger)`, as seen below.



Another way would be to subscribe to the `Typewriter.onTriggerEncountered` event before starting the effect.

Step 5 (optional) - Rich Text formatting

Rich Text is a great way to play with font styles, colors and sizes without having to use different text elements to display the differently styled texts. Luckily, both of Unity's GUI systems, the legacy and uGUI support Rich Text Formatting. Below is an RTF markup example and how it looks in action:

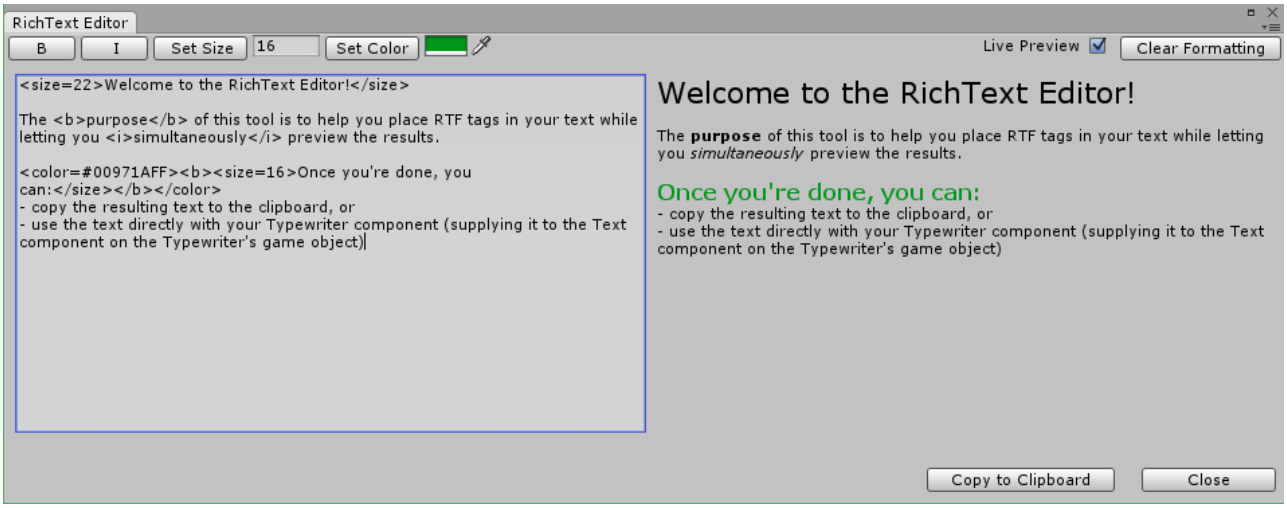
> *With RTF you can distract the readers like a <color=green>charm</color>!*

With **RTF** you can *distract* the readers like a **charm**!

To learn more about RTF and its special tags, please check out the Unity documentation.

5.1 Built-in Rich Text Editor

With version 1.2.0 of Typewriter & Fade-in Text Effect arrives the built-in Rich Text Editor. The purpose of this editor window is to help you place RTF tags in your text, which you can then paste anywhere or use directly with your Typewriter component.



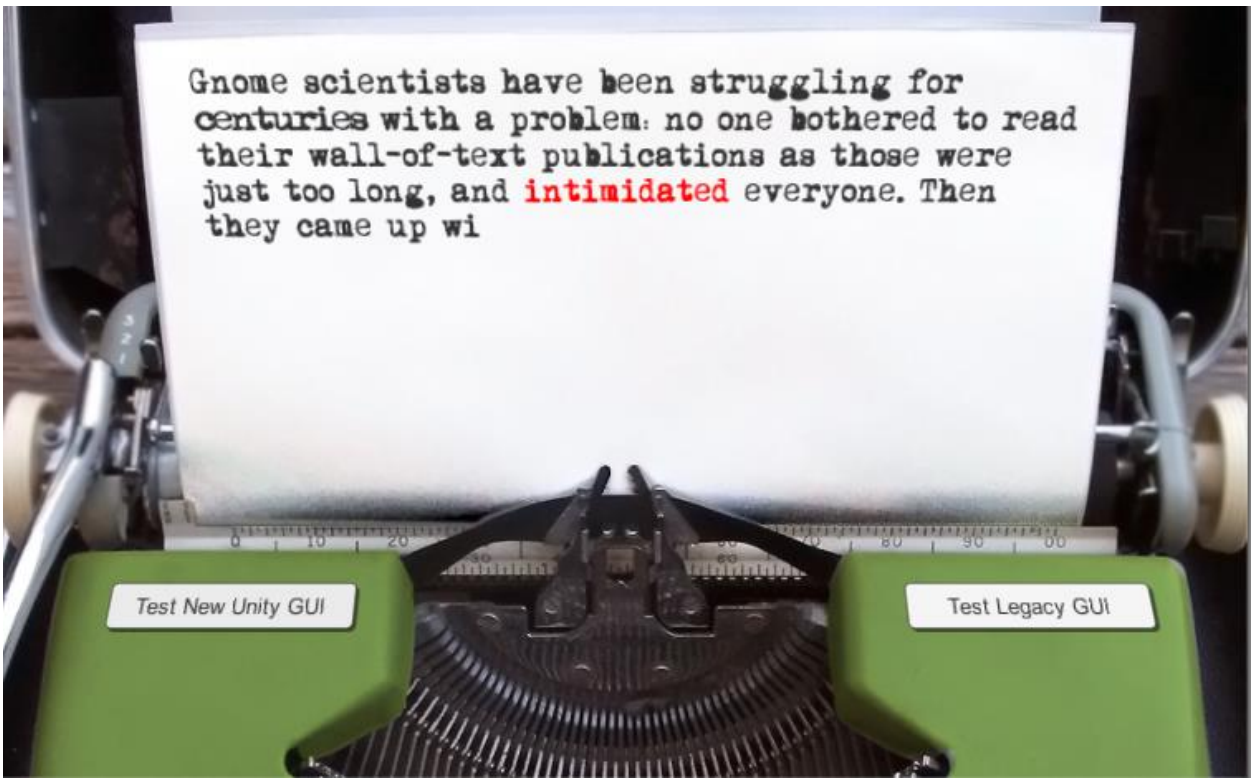
You can open the RichText Editor either via the *Window menu -> RichText Editor*, or by clicking the *RichText Editor* button at the bottom of the Typewriter inspector.

4. Example Scenes

There are three sample scenes included, each showcasing different practical usages of the Typewriter & Fade-in Text Effect.

4.1 - Wall-of-Text Example

This demo demonstrates how to start Typewriter from code, and how to use it with uGUI and a different GUI-solution (the legacy Unity GUI in our case).



If you're planning to use the Typewriter with any other GUI systems than uGUI, just check out how it's done in *LegacyGUISample.cs*:

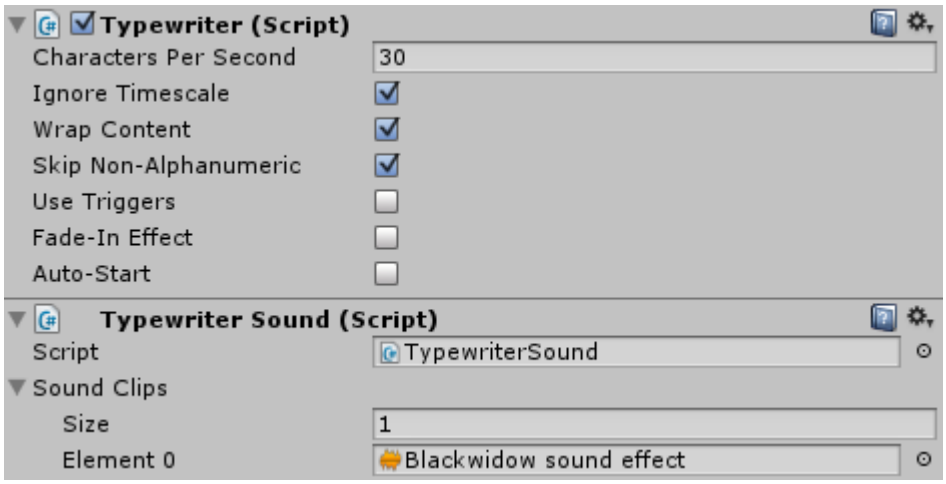
```
// Subscribing to events before doing anything else
typewriter.onTextChanged += OnTypeWriterText_Changed;
typewriter.onFinished += OnTypeWriter_Finished;
// Starting the typewriter
typewriter.BeginCallbackMode(text);
```

4.2 - Dialogue Example

The Typewriter effect has a great use in dialogue systems, as this example tries to demonstrate using uGUI. To check out how this example works, you might want to look into *DialogueSample.cs*, located on the "Dialogue Sample" game object.

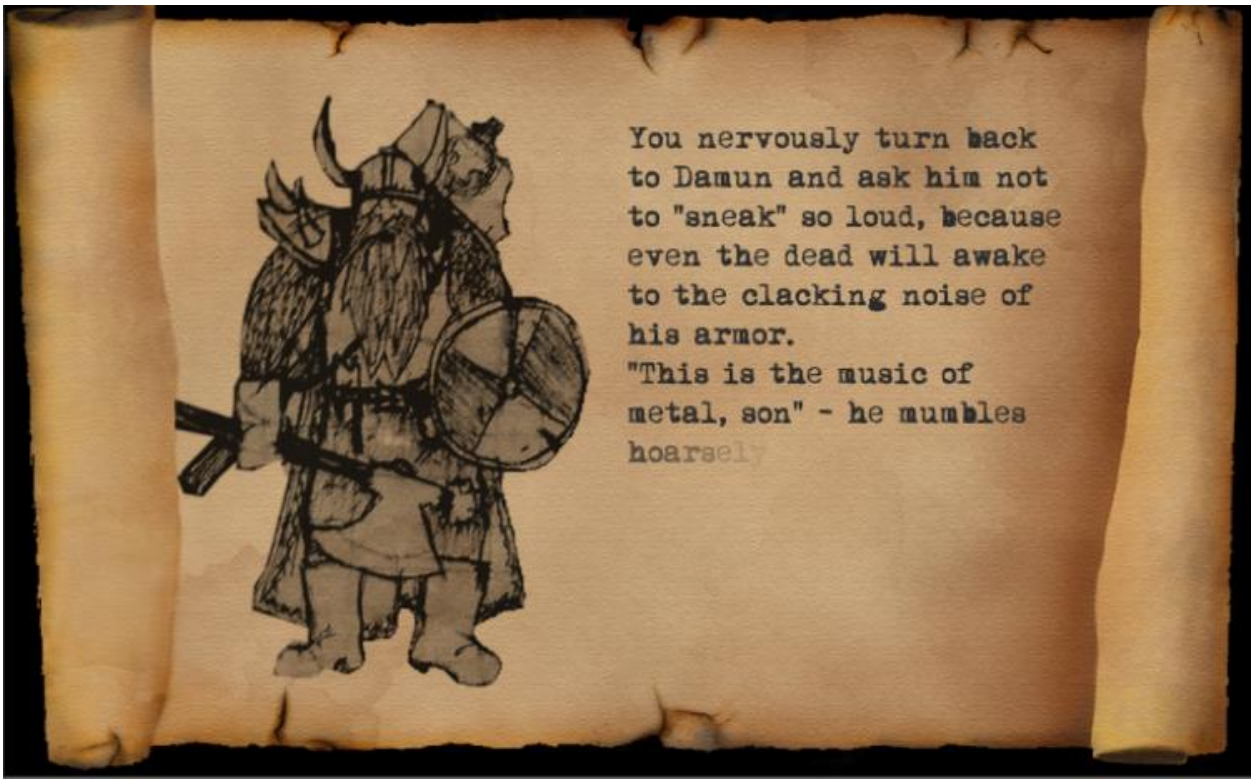
There are two useful tricks to see here:

- Finishing the effect upon clicking: If you click on the text while the typewriter is in progress, it immediately displays the whole text. This is a good tool to give a chance to the less patient users to progress faster with your dialogues.
- Playing sound effects: Since the Typewriter used for displaying the dialogue's nodes has a TypewriterSound component attached, it plays the specified sound effect while the effect is in progress.



4.3 - Fade-in Example

While the classic typewriter effect is quite modern, fading the text gradually fits games with even a more historical setting or atmosphere. Check out the Fading Example to see this effect in action!



The techniques used in this demo:

- All it took to configure the fading effect was checking the Fade-In Effect checkbox and specifying the Fade-In Time.
- Since Auto-Start and the text was specified, no coding was required to get this typewriter working!
- Because triggers and their observer were specified we were able to get notified (as seen in the console while playing the effect) when the text reached any of the triggers. The observer simply handles the triggers with the following method (as seen in TriggerObserverSample.cs):

```
public void OnTypewriterTrigger_Encountered(string trigger)
{
    Debug.Log(string.Format("Encountered trigger '{0}'.", trigger));
}
```

5. Updating

If a new version of the asset is available, the safest way to update is the following:

1. (Optional) Create a backup of your folder „Typewriter & Fade-in Text Effect" and place it somewhere outside of the project folder (in case for some reason the new version wouldn't work out for you).
2. Open a new, empty scene (Ctrl + N).
3. Delete the folder „Typewriter & Fade-in Text Effect".
4. Import the new package, and place it wherever you'd like in your project structure.

6. Credits

- Old typewriter sound effect by pakasit21:
- Typewriter photo by Lipah-WritersBlock:
- Mom's Typewriter Font by Christoph Mueller