



Documentation

Index

1. Basic Setup
2. Making/Using a Bitmap Font
3. Creating An Animation
4. Advanced Animations
5. Loops
6. Audio and Particle Effects
7. Scripting Reference
8. Best Practice Tips
9. FAQs
10. Feedback & Support

1 - Basic Setup

Index

- [Importing the Plugin](#)
- [Setting up a text animation object](#)

Importing the Plugin

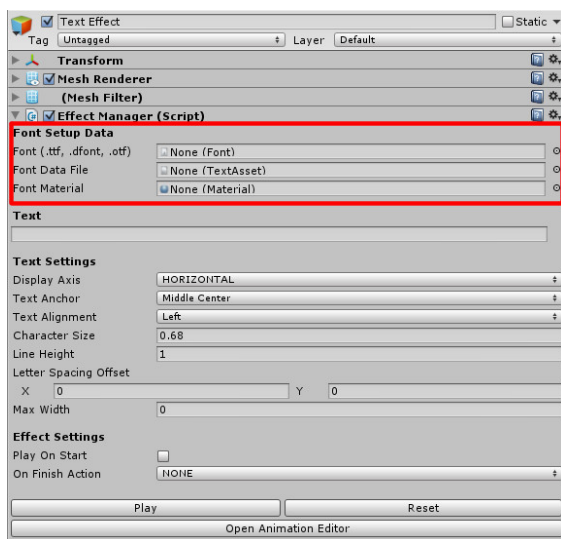
After purchasing the plugin through the Asset Store and importing the package into your project, you should have a **TextFX** folder in your project hierarchy.

Within this folder will be the folders **Editor** and **Scripts**. The **EffectManager.cs** script in the **Scripts** folder is the only script you'll need to use; it's the component script which handles the text animations.

There's a component menu shortcut for adding the EffectManager script to your object; **Component-> TextFx-> EffectManager** from the main Editor menu.

Setting up a text animation object

1. Create a new empty GameObject in your scene.
2. Apply the EffectManager script to the GameObject; **Component-> TextFx-> EffectManager**

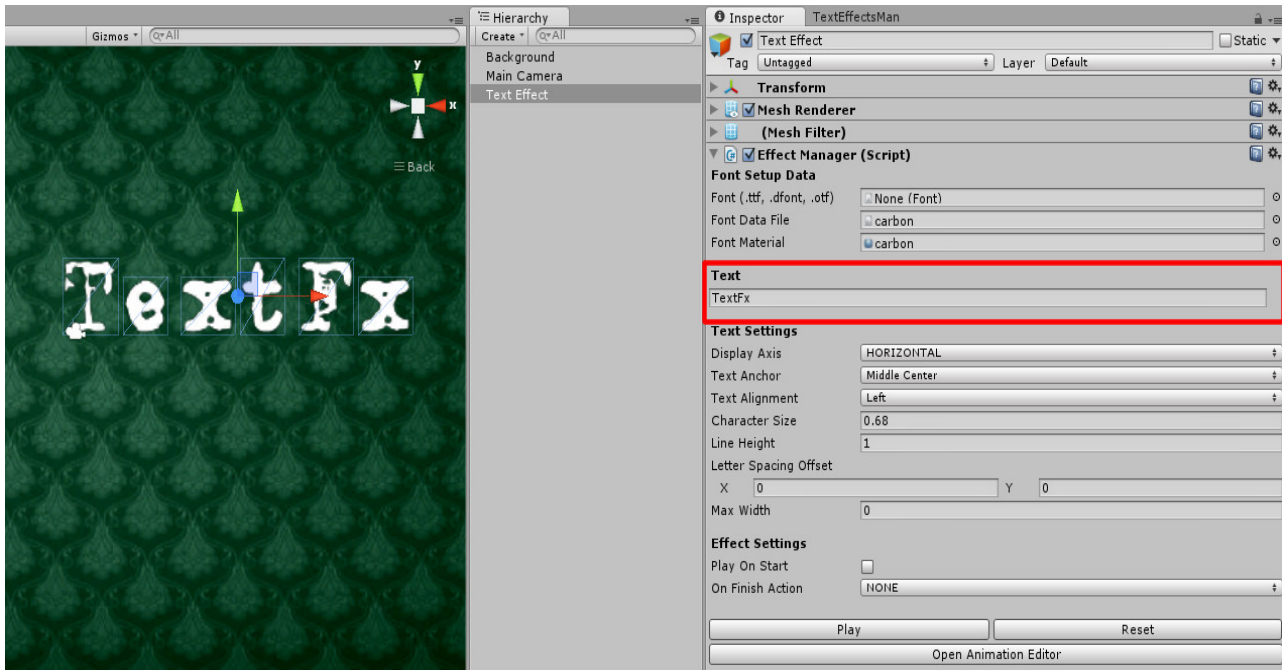


3. Assign your desired font file to the **Font** field (Unity 4.0 + only), or assign your Bitmap font text data file and it's associated Material to the **Font Data File** and **Font Material** fields respectively.

Note: For instructions on how to generate your compatible Bitmap Font read [these instructions](#).

Note: Try using one of the included Bitmap or TrueType fonts to get a quick start with your TextFx experimentation! Fonts found in 'Demo Scenes/Fonts'.

4. Type some text into the **Text** field to make sure your font assignment has worked correctly. You should see some text appear!



5. There are some settings available in the inspector for adjusting the basic layout/appearance of the text:
 - **Text** - The text to be displayed and animated in the effect.
 - **Display Axis** - Denotes whether the text should be written horizontally across the screen, or vertically down the screen.
 - **Text Anchor** - Denotes what part of the text object will act as the text anchor point.
 - **Text Alignment** - Denotes which side (left, right or center) the text will align to.
 - **Character Size** - Used for scaling the letters greater or less than their default display size. Default value is 1.
 - **Line Height** - Used for scaling the line height greater or less than their default display size. Default value is 1.
 - **Letter Spacing Offset** - Used for applying extra spacing between each letter.
 - **Max Width** - Maximum width that the text can spread before being forced onto a new line. Displayed in the editor scene window with red borders.
 - **Play On Start?** - Denotes whether the animation will start playing automatically or not, when the EffectManager object is first active in the scene.
 - **On Finish Action** - What should happen to the effect/object when the animation has finished.
 - Preview animation Control Buttons; **Play** and **Reset**. Used for previewing your animation in the Editor.
 - **Open Animation Editor** - Opens the TextFx Manager window.
6. When you're happy with how your text looks, and you've positioned the object to roughly where you'll want it in your scene, then you're ready to configure the animation.

Proceed to the [***Creating An Animation***](#) instructions.

2 - Making/Using A Bitmap Font

Index

- [Creating Fonts](#)
- [Using a Bitmap Font with TextFx](#)
- [Creating a Bitmap Font using BMFont](#)
- [Creating a Bitmap Font using Hiero](#)

Creating Fonts

TextFX supports the use of Bitmap fonts, which consist of a **Texture atlas** containing all of the required letters from the font, and a **text data file** describing the location and size of each letter on that texture atlas.

Using a Bitmap font has a few potential advantages; firstly that you can pre-select just the letters you'll be needing from that font, and therefore save on texture memory. Secondly, since you have access to the font sprite atlas, you can apply post-processing effects to the font, such as a glowing edge.

There are two free, third-party tools you can use to generate the Bitmap font files:

BMFont (Windows only) - <http://www.angelcode.com/products/bmfont/>

Hiero (All platforms - Java Executable) - <http://slick.cokeandcode.com/demos/hiero.jnlp>

Below are instructions for generating a font [using BMFont](#) and [using Hiero](#).

Using the Bitmap Font with TextFX

Once you've generated a Bitmap font data file and texture atlas and placed them in your Unity project Assets folder, you'll need to **setup a Material** for this Bitmap Font.

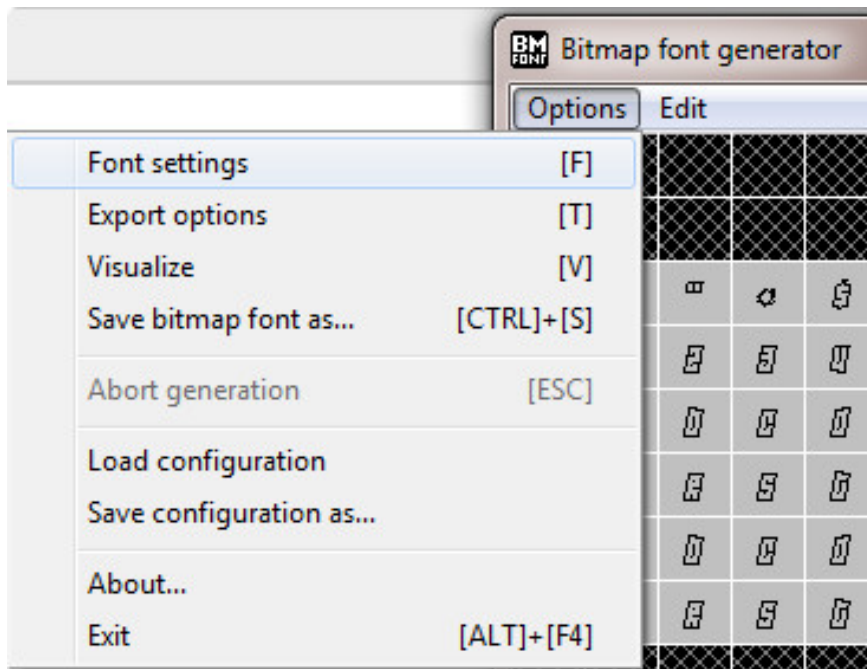
1. Make a new Material instance in your project hierarchy
2. Drag the your Font Texture onto the new materials main texture field in the Inspector to assign it.
3. **Set the shader** for the material to be **GUI->Text Shader**

Note : If you're using Unity 3.5 or less, you'll need to set the shader as TextFX->Text Shader instead, which adds VertexColor support to the default Unity Text Shader.

Remember to change the file extension of your data file from a *.fnt filetype to a ***.txt**

Creating a Bitmap Font using BMFont

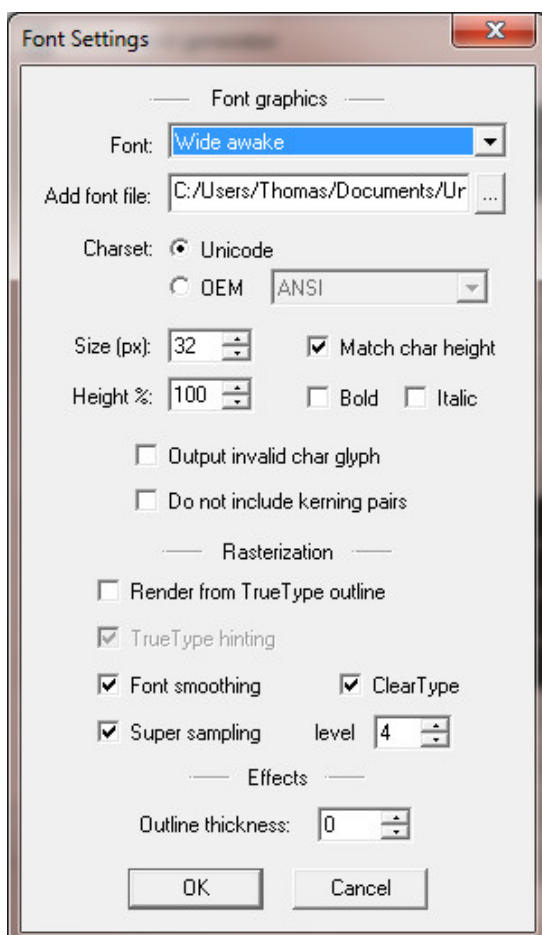
1. Run BMFont and open **Options->Font Settings**.



2. Select your desired font from the list of installed fonts on your PC, or add a font file that you have on your harddrive.

Set the pixel size of the font and check **Match char height**

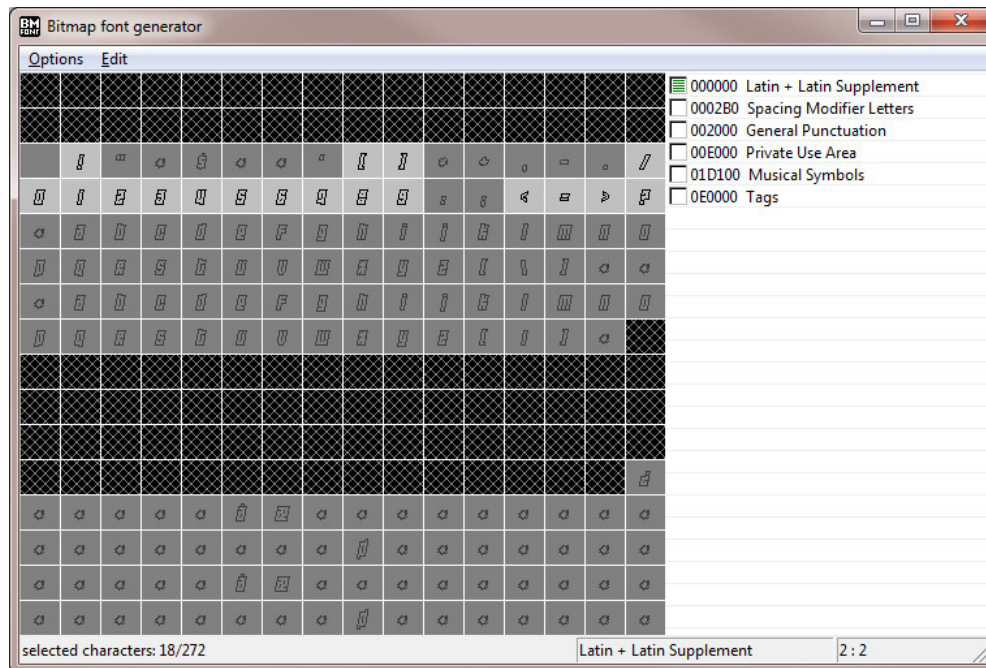
Check **Font Smoothing** and **Super Sampling** if you want the letters to be smooth and anti-aliased.



3. Then back in the main tool window you'll see all the letters contained within your chosen font.

Highlight all the letters you wish to be contained in the font, either individually by toggling on/off each letter, or by toggling subsections of letters from the list on the right.

Note: If you're selecting your own custom subset of characters, remember that you'll need to select a "space" character in order to use spaces in TextFX!



4. Next, open **Options->Export Options**

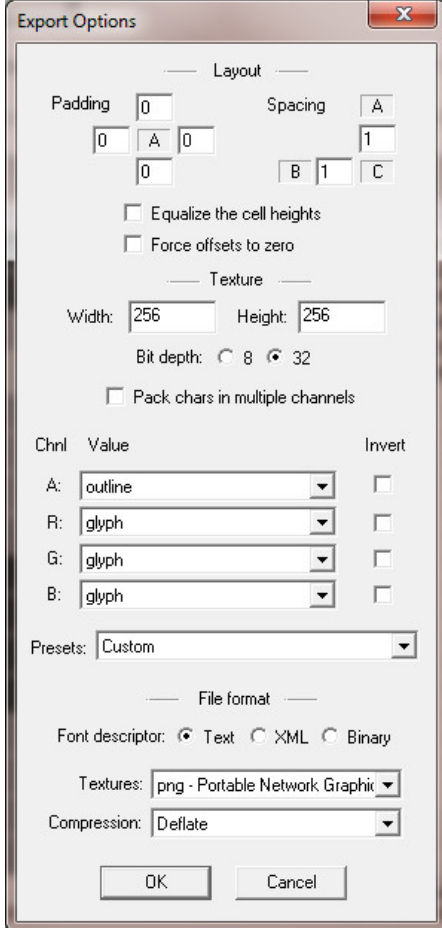
Depending on your chosen font, and whether you're adding any after effects to the texture atlas, you may want to add some letter padding to prevent unintentional artifacts on the letters in Unity. Bit of a trial and error process.

Set a Width and Height for the texture atlas. It needs to be big enough to fit all the letters onto one Texture. To check that it all fits on the chosen size, go back to the main window and press 'V' or go to **Options->Visualize**.

Note: Best to stick to power-2 dimensions for the atlas to keep Unity happy. ie. 2,4,8,16,32,64,128,256,512 etc.

Check 32 **Bit depth** to allow alpha channel.

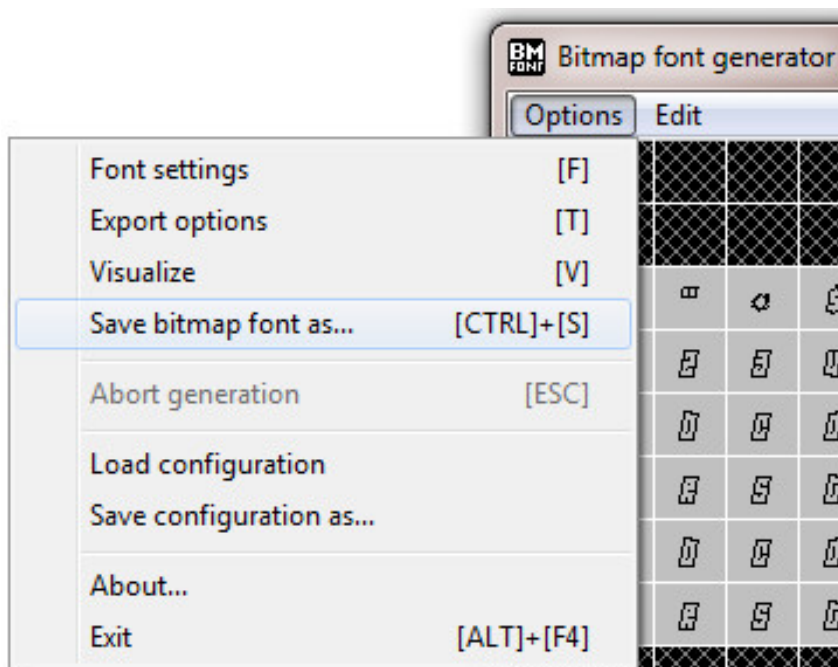
Check **Text** File Format, and **PNG** for the Texture format.



5. Select **Options->Save bitmap font as...**

Save the Bitmap font to your Unity project folder.

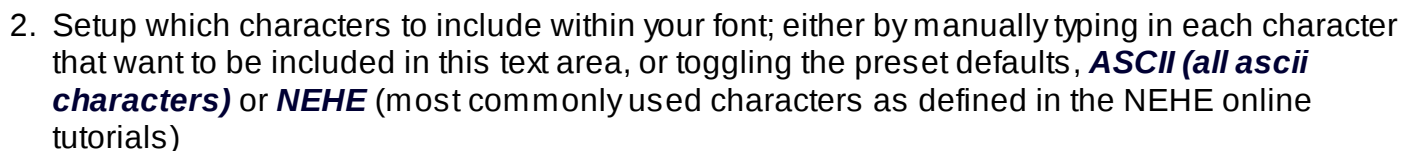
Change the file type of the ***.fnt** data file to be a ***.txt** file, so that Unity can interpret it as a TextAsset.



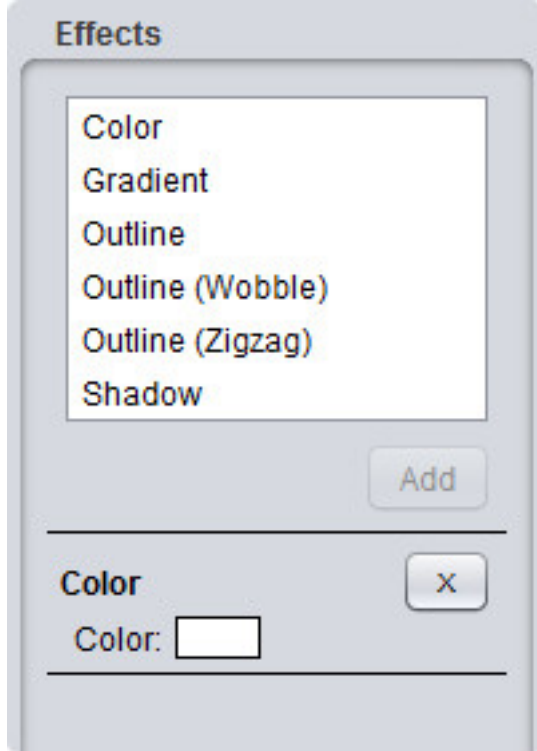
Creating a Bitmap Font using Hiero

1. Run the Hiero Java executable application, and select from the list of fonts installed on your

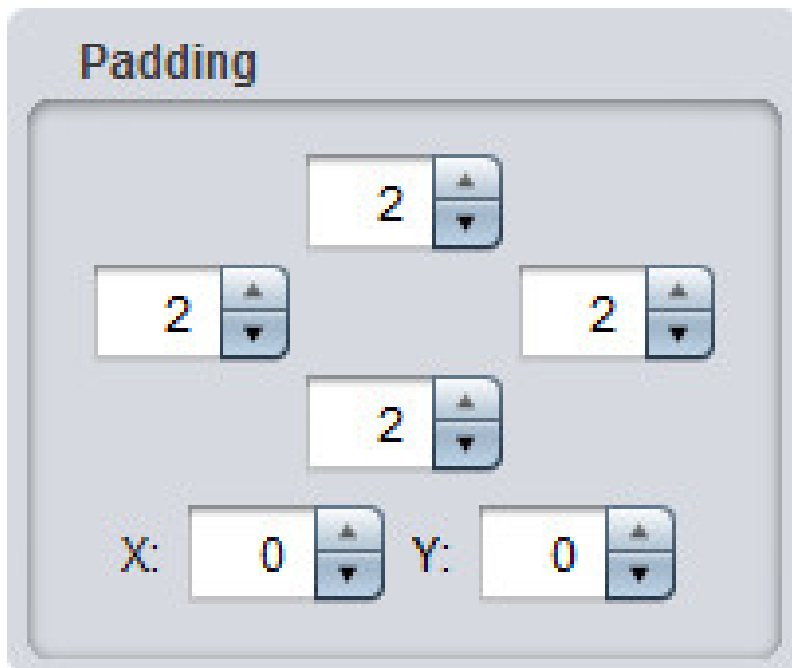
Select the pixel size of the Font and whether it should be **Bold** or *Italic*.

[illegible]

Note: To take full advantage of the VertexColor text colouring features available in TextFX, you'll want to keep your font base colour as white.



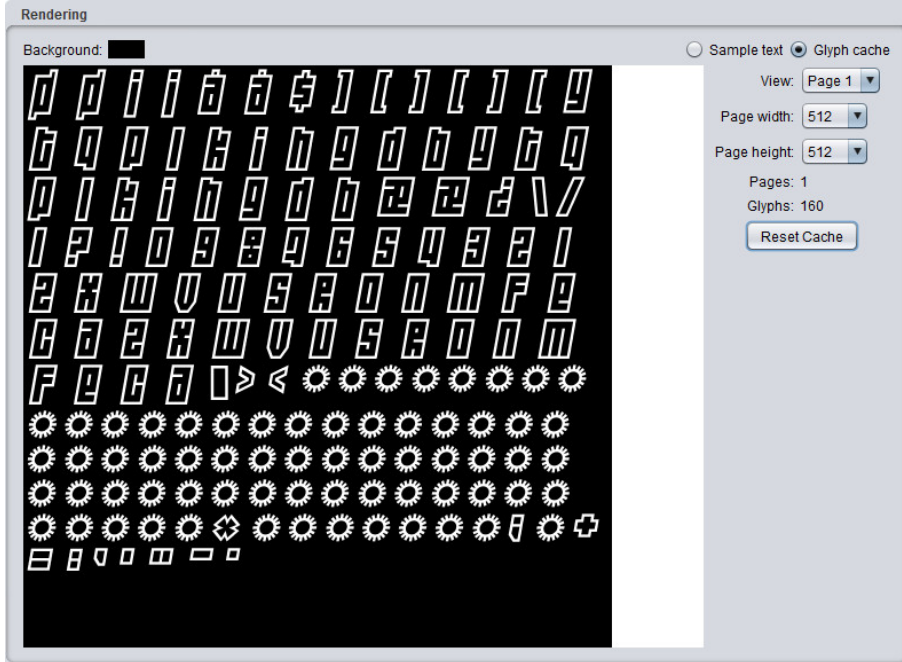
Depending on the effects you've applied, you may need to apply a bit of padding around each letter to accomodate for it and avoid the effect leaking into the adjacent letter's.



4. Once you've listed all of the characters you want to be included, applied the desired effects and added your letter padding, click the **Reset Cache** button in the **Rendering** panel to recalculate the letter placings on the texture atlas to the left.

TextFX only supports Bitmap fonts with one single Texture Atlas, so you'll need to adjust the dimensions of the Atlas using the **Page Width** and **Page Height** drop-downs until all the letters fit onto one page.

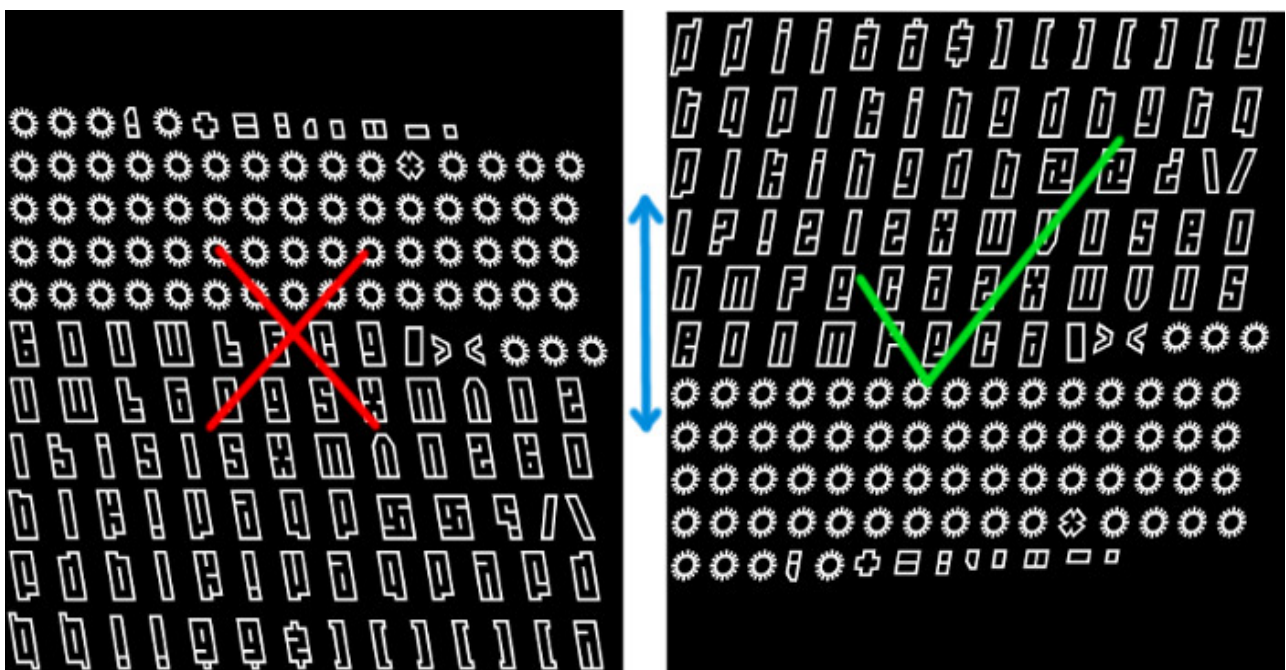
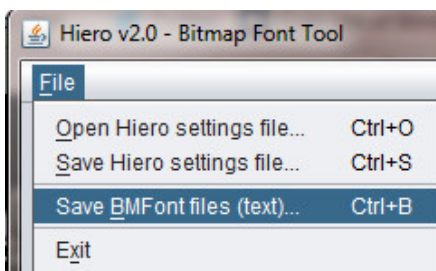
(Check that only one page is listed in the **View** drop-down)



- Next, select **File->Save BMFont files (text)...** and save the Bitmap font to your Unity project folder.

Change the file type of the ***.fnt** data file to be a ***.txt** file, so that Unity can interpret it as a TextAsset.

Important: You'll need to flip the generated texture atlas vertically using an image editing software before it'll work correctly with TextFX in Unity.



3 - Creating An Animation

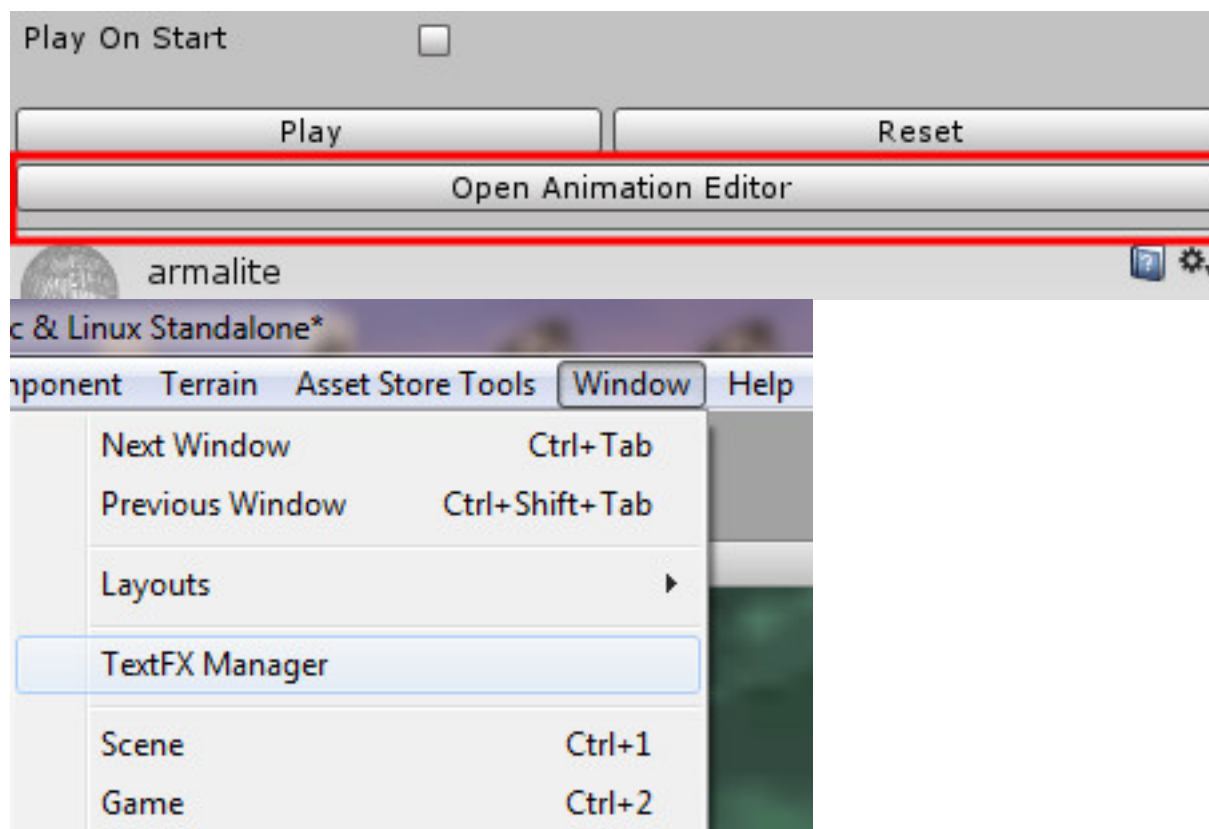
Index

- [Opening the TextFX Manager](#)
- [General Settings](#)
- [Adding An Animation](#)
- [TextFX Actions](#)
- [Advanced Action Settings](#)
- [More Than One Action](#)
- [Offset Variables From Last State](#)
- [Per-Axis Easing Function Overrides](#)

Opening the TextFX Manager

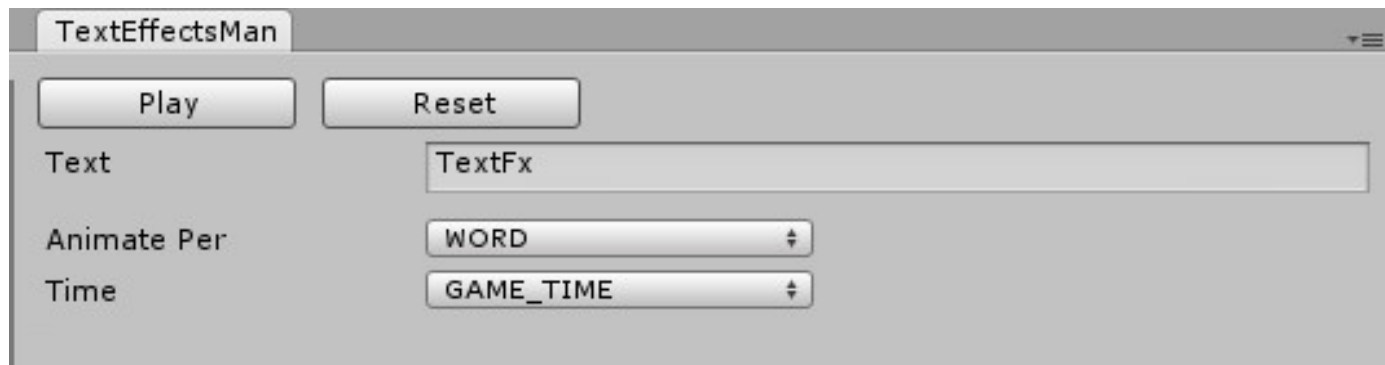
Assuming that you've followed the [Basic Setup instructions](#) and have an EffectManager object instance in your project, you now need to open a TextFx Manager window to set up your animation.

You can open the TextFX Manager from the button at the bottom of the EffectManager inspector, **or** by going to ***Window->TextFX Manager***



General Settings

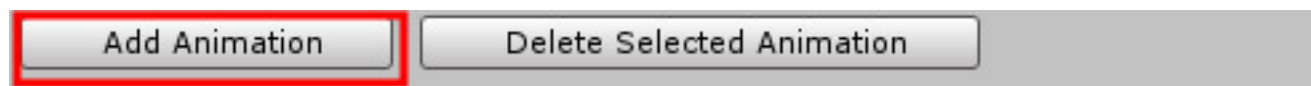
The basic settings:



- Preview animation Control Buttons; **Play** and **Reset**. Used for previewing your animation in the Editor.
- **Text** - The text to be displayed and animated in the effect.
- **Animate Per** - Denotes the default AnimatePerOption; this says whether the text will animate on a per-letter, per-word or per-line basis. This value can be overridden on individual cases in your animation.
- **Time** - Denotes whether the animation will use Unity's in-built Time scale (Time.deltaTime), or instead use realtime.

Adding an Animation

By default, a new EffectManager object will have no animations setup, so you'll need to add one by clicking the **Add Animation** button.



It's possible to have more than one animation for your Text Effect. Use the **Anim (N)** tabs to select between them.

Each letter can only be controlled by one animation though, so you can specify a subset of letters for the animation to apply to via the **Animate On** field.



TextFX Actions

Every TextFX animation is a sequence of one or more **Actions**.

An **Action** defines the transition of the text from a start state to an end state.

You setup a start and end **colour**, **position**, **rotation** and **scale**, and the **duration** of the transition.

For each **Action** you have the following options:

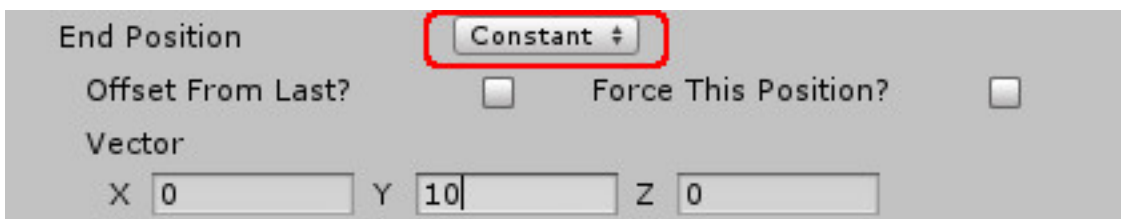
1. **Action Type** - Denotes whether this action is a normal **animation sequence**, or a **break** action for stopping the animation for a given amount of time.
2. **Letter Anchor** - The anchor point used on each letter for this Action - ie. what part of the letter to rotate around and scale from/to.
3. **Ease Type** - The easing function to use for this Action - changes how the state progresses over time between the start and end state.
4. **Start/End Colour Gradients?** - Toggles between using one flat colour for the start/end state, or using four colours for each corner of the letter mesh which blend into one another.
5. **Start/End Colour** - The colour(s) of the letters at the start and end of the action.
6. **Set Position Axis Ease?** - Allows you to override the action 'Ease Type' for each position axis individually to create a more unique movement.
7. **Start/End Position** - The position of the letters at the start and end of the action.
8. **Force This Position?** - If toggled, this position variable will be the forced position of all letters in the text, without using the default character spacings. Ie. It'll pile up all the letters at one position in the scene.
9. **Set Rotation Axis Ease?** - Allows you to override the action 'Ease Type' for each rotation axis individually to create a more unique movement.
10. **Start/End Euler Rotation** - The euler rotation of the letters at the start and end of the action.
11. **Set Scale Axis Ease?** - Allows you to override the action 'Ease Type' for each scaling axis individually to create a more unique movement.
12. **Start/End Scale** - The scale of the letters at the start and end of the action.

13. **Force Same Start?** - Forces all letters in the text to start this action at the same time.
14. **Delay** - How long to delay before starting the Action.
15. **Duration** - How long the Action will take to complete.
16. **Audio OnStart/OnFinish** - Audio clip to play when action starts/ends. **Covered in more detail in the Audio and Particle Effects section.**
17. **Emitter OnStart/OnFinish** - Particle Emitter animation to play when action starts/ends. **Covered in more detail in the Audio and Particle Effects section.**

Advanced Action Settings

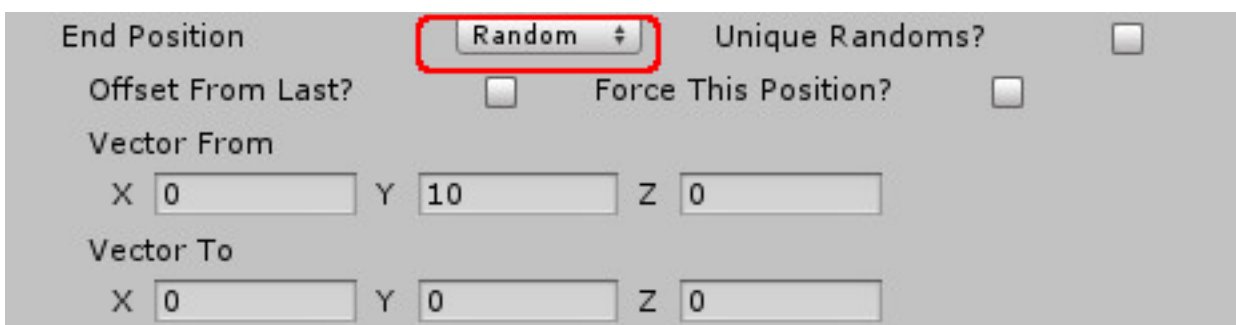
Each of the main transition and timing variables on an Action have additional settings to define the spread of values across each letter. These help to add a lot of variation to your effects!

- **Constant variables** - All letters will use this same value



The screenshot shows the 'End Position' settings for a 'Constant' variable. The 'Constant' button is highlighted with a red box. Below it, there are checkboxes for 'Offset From Last?' and 'Force This Position?'. The 'Vector' section has input fields for X (0), Y (10), and Z (0).

- **Random variables** - Letters will each use a unique value randomly chosen between the **From** and **To** bounds.



The screenshot shows the 'End Position' settings for a 'Random' variable. The 'Random' button is highlighted with a red box. To its right is a 'Unique Randoms?' checkbox. Below the button are checkboxes for 'Offset From Last?' and 'Force This Position?'. The 'Vector From' section has input fields for X (0), Y (10), and Z (0). The 'Vector To' section has input fields for X (0), Y (0), and Z (0).

- **Eased variables** - The first letter of your text will have the **From** value, and the last letter will have the **To** value, with the letters in between being set an eased value between the two based on its position in the text.

You can set the easing function to alter the progression of values across the letters.

End Position **Eased** Function : Linear 3rd? ☐

Offset From Last? ☐

Vector From
X 0 Y 10 Z 0

Vector To
X 0 Y 0 Z 0

- **3-way Eased variables** - Similar to the previous 2-way Eased variable, except the values can ease from the start to a middle point value, and then ease towards a third value for the second half of the letters.

End Position **Eased** Function : Linear 3rd? ☒

Offset From Last? ☐

Vector From
X 0 Y 10 Z 0

Vector To
X 0 Y 0 Z 0

Vector Then
X 0 Y 10 Z 0

More Than One Action

When adding additional Actions to your TextFX animation, you'll notice it appears slightly different.

In order to save time setting up TextFX animations, your second and all subsequent Actions can be set to be an **Offset** from the last Action, using the **Offset Prev?** checkbox.

This will hide all of the **Start** state variables since this action will be starting from whatever state the previous action ended in; so you only need to define how it transitions from there.

You can uncheck this option to define your own unique starting states again, but beware that this may result in a jarring/jumpy animation.

▶ **Action 0**

AddDeleteDownReset

▼ **Action 1**

Offset Prev? ☒AddDeleteUpReset

Letter Anchor

Middle Center

Ease Type

Linear

Use Colour Gradients?

☐

End Colour

Constant

Offset From Last?

☐

Colour

End Position

Eased

Function :

Linear

3rd? ☒

Offset From Last?

☐

Vector From

X

0

Y

10

Z

0

Vector To

X

0

Y

0

Z

0

Vector Then

X

0

Y

10

Z

0

End Euler Rotation

Constant

Offset From Last?

☐

Vector

X

0

Y

0

Z

0

End Scale

Constant

Offset From Last?

☐

Vector

X

1

Y

1

Z

1

Force Same Start?

☐

Delay

Constant

Value

0

Duration

Constant

Value

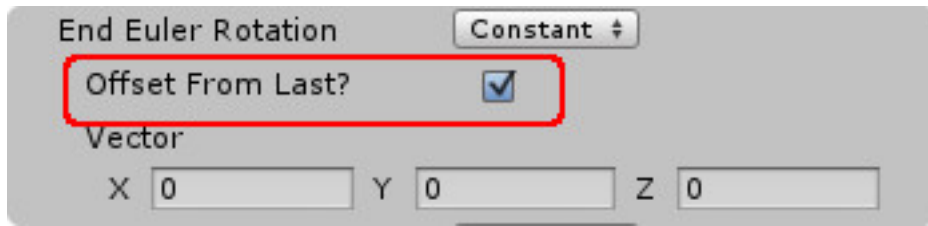
0

Offset Variables From Last State

Another time saving feature available on all variable states except the very first, is the option to have that variable be an offset from the previous state value using ***Offset From Last?***.

When set, the variable value will be added on to the last state value; ie. If set to zero, the value will remain the same as it was in the previous state.

This is a useful when the previous variable state was randomly chosen, and you then want to continue from that random offset with a fixed value



End Euler Rotation Constant ▾

Offset From Last? ☒

Vector

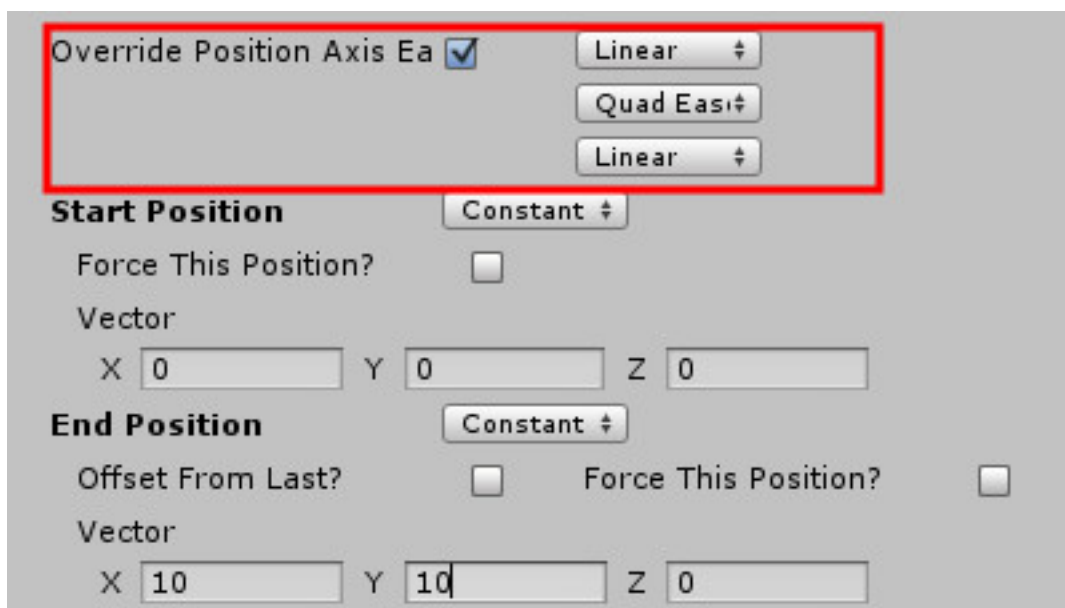
X Y Z

Per-Axis Easing Function Overrides

For any of the transformations in your Action, you can choose to set unique Easing Functions for each axis individually.

This allows you to create more dynamic looking movements

Try overriding the axis easing for your positional translation! It looks cool.



Override Position Axis Ea ☒ Linear ▾

Quad Easi ▾

Linear ▾

Start Position Constant ▾

Force This Position? ☐

Vector

X Y Z

End Position Constant ▾

Offset From Last? ☐ Force This Position? ☐

Vector

X Y Z

4 - Advanced Animations

Index

- [Break-State Actions](#)
- [Custom Animation Letters](#)

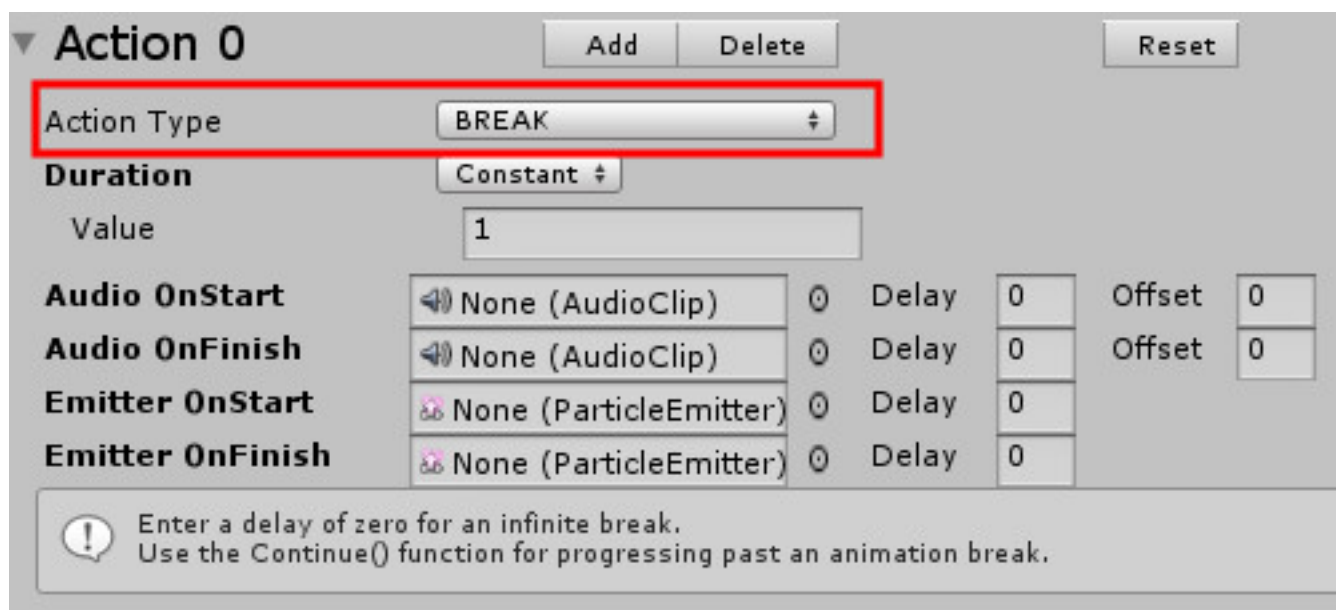
Break State Actions

You can create a paused state in your animation by adding a **Break-State** action. Setting the **Action Type** of an action to **BREAK** will pause this animation for the specified duration of time.

Setting a break duration of zero or less will mean that the animation gets paused at this point indefinitely.

An animation in a paused state (infinite or finite) can be continued by calling the **ContinueAnimation** scripting function, which will force the break to finish, and continue the animation to the next action.

See the [Scripting Reference](#) page for information on all the scripting functions available.



Action 0 [Add] [Delete] [Reset]

Action Type: **BREAK**

Duration: Constant

Value: 1

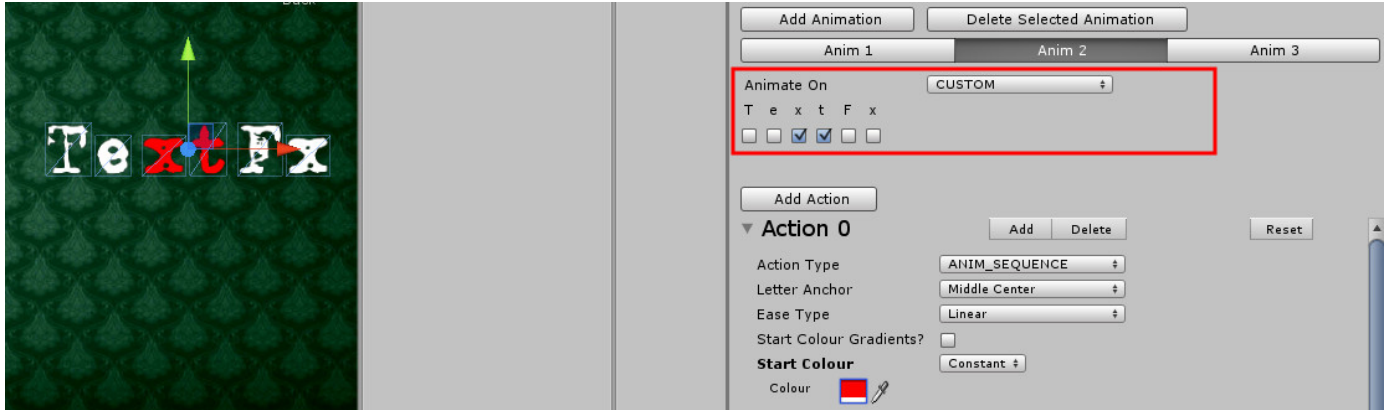
Audio OnStart	None (AudioClip)	0	Delay	0	Offset	0
Audio OnFinish	None (AudioClip)	0	Delay	0	Offset	0
Emitter OnStart	None (ParticleEmitter)	0	Delay	0		
Emitter OnFinish	None (ParticleEmitter)	0	Delay	0		

! Enter a delay of zero for an infinite break.
Use the Continue() function for progressing past an animation break.

Custom Animation Letters

Each animation has the option to specify which letters it will apply to within the text, using the **Animate On** field. By default an animation will apply to all letters in the text, but you can change it to be a subset of the letters.

Note: Each letter can only be controlled by one animation. If you're using more than one animation, make sure none of the other animations try to *Animate On* the same letters.



5 - Loops

Index

- [Loops Menu](#)
- [Adding A Loop](#)

Loops Menu

With your TextFX animation, you're able to set one or more Actions to loop over themselves however many times you'd like.

Once a loop is finished, the animation will continue linearly from wherever the loop ended

The left hand section of the TextFX Manager window is where you'll find the **Loops Menu**.

Here you'll see a list of all the current loops applied to your animation, and a form for inputting new Loops

The screenshot shows the 'Loops [0]' panel in the TextFX Manager. It features a 'Hide' button at the top left. Below it, the 'Active Loops' section is visible. This section contains a table with columns: 'From', 'To', '#Loops', 'Type', and 'DFO [?]'. The table is currently empty. To the left of the table, there are labels 'A0', 'A1', and 'A2'. To the right of the table, there is a 'New' button and an 'Add' button. The 'Add' button is located to the right of the '#Loops' column, which currently contains the value '0'.

	From	To	#Loops	Type	DFO [?]
A0					
A1					
A2					

Adding A Loop

A loop is defined by a **start action index** and an **end action index**, and then the **number of loop iterations**.

Setting the number of loop iterations to be **zero or less** will cause the loop to running **infinitely** (represented in the Editor by a "~").

There are two types of loop available:

- **Normal Loop** : Loops through the actions in order from the start index to the end index, and then returns to the start index for the next loop.
- **Reverse Loop** : Loops through the actions in order from the start index to the end index, and then reverses the animation back from the end state of the end action, to the start state of the start action.

The last setting is **DFO (Delay First Only)** which will cause your loop to only apply action timing delays for the first forward pass through the loop. This stops the letters getting more and more out of sync with each loop iteration. Delays which are constant across all letters **will** still be applied, as these won't affect the sequencing between letters.

When you add a loop, you'll see a visual representation of the loop on the Action tree on the left of the loop menu.

Note: You can also add loops by clicking on the Action nodes on the left hand side. Click and drag from one node to the other to create a loop across several actions.

Note: Loops can be embedded within each other, but they can't intersect other loops. For instance having a loop from Action 0 -> 2 and then trying to add a loop from Action 1 -> 3, would not be allowed!

The screenshot shows the 'Loops [3]' interface. On the left, a vertical action tree displays nodes A0, A1, A2, and A3. A red vertical line with a red box containing the number '3' spans from A1 to A2, indicating a loop. A0 has a blue box with '2' and A3 has a blue box with a tilde symbol. At the top left is a 'Hide' button. To the right is a table titled 'Active Loops'.

	From	To	#Loops	Type	DFO [?]	
Loop 1	0	0	2	LOOP	<input type="checkbox"/>	x
Loop 2	3	3	0	LOOP	<input type="checkbox"/>	x
Loop 3	1	2	3	LOOP_REVERSE	<input checked="" type="checkbox"/>	x
New	0	0	Add			

In the example shown above, the animation progress will be as follows:

- Loop over Action 0 twice.
- Reverse Loop over Action 1 and Action 2 three times.
- Loop over Action 3 infinitely.

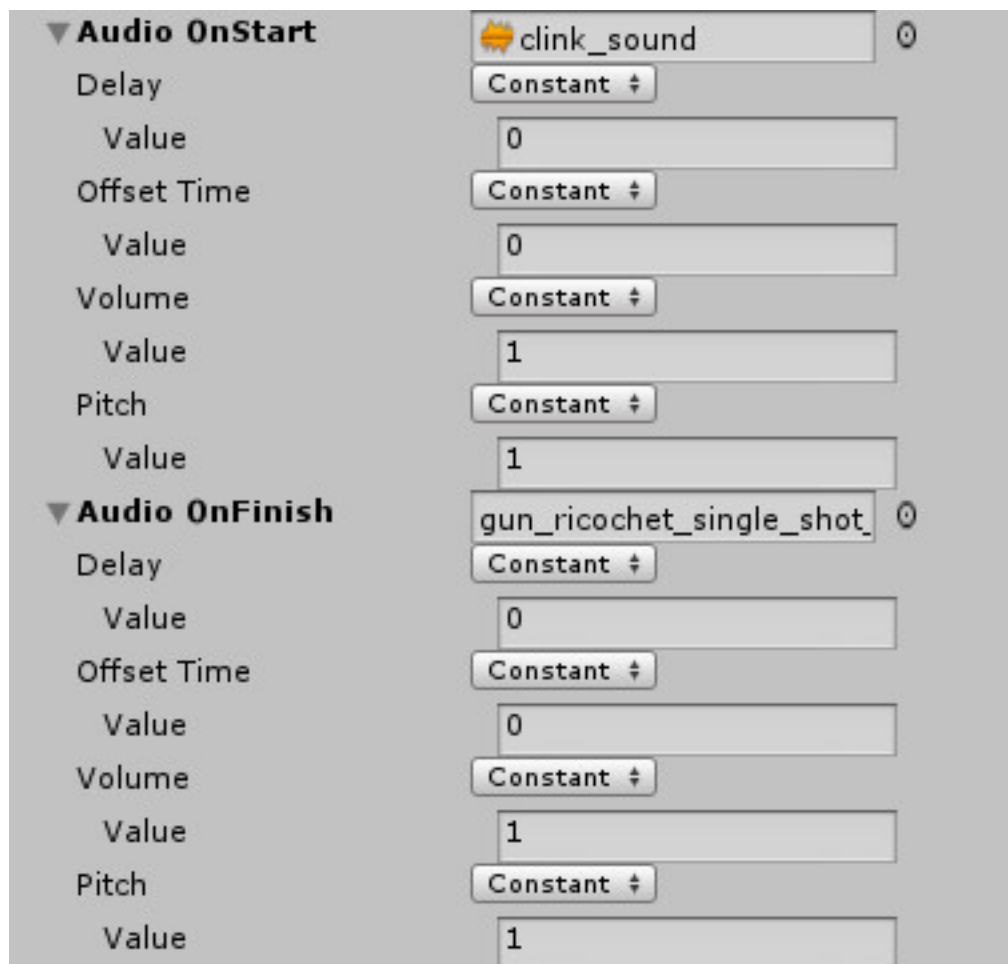
6 - Audio And Particle Effects

Index

- [Adding an Audio Clip](#)
- [Adding A Particle Effect](#)

Adding an Audio Clip

Each Action you add to your animations has the following options at the bottom to add audio clips triggered when the action starts and ends.



The screenshot displays the Unity animation editor's settings for an animation action. It is divided into two main sections: 'Audio OnStart' and 'Audio OnFinish'. Each section has a dropdown menu to select an audio clip, followed by fields for 'Delay', 'Value', 'Offset Time', 'Volume', and 'Pitch'. Each field has a 'Constant' button and a value input box. The 'Audio OnStart' section is currently set to 'clink_sound' with a value of 0. The 'Audio OnFinish' section is currently set to 'gun_ricochet_single_shot' with a value of 0.

Section	Audio Clip	Delay	Value	Offset Time	Volume	Pitch
Audio OnStart	clink_sound	Constant	0	Constant	0	Constant
Audio OnFinish	gun_ricochet_single_shot	Constant	0	Constant	0	Constant

Assign a Unity supported audio clip to the **Audio OnStart** or **Audio OnFinish** field to have the audio play. You can preview the effect in the editor.

Note: It is recommended that you preview your animation in the editor, so that it creates the required number of AudioSource instances outside of runtime. See 'Best Practice Tips' for more info!

The following settings are available for each Audio clip you assign.

- **Delay** - The amount of time (in seconds) the audio clip will be delayed before playing.

- **Offset Time** - What time within the audio clip (in seconds) should the clip start playing.
- **Volume** - The volume that the audio clip should be played at. Value between 0 and 1.
- **Pitch** - The pitch that the audio clip should be played at.

Note: If all the letters are starting or finishing the action at the same time, and there is a constant time delay for your audio clip, then only one audio clip will be played for all of the letters. Else, if the letters are out of sync or if an eased or random time delay is specified for the audio clip, then an audio source will be played for each letter.

Adding a Particle Effect

Each Action you add to your animations has the following options at the bottom to add Particle Effects triggered when the action starts and ends.

The screenshot displays the 'Emitter OnStart' and 'Emitter OnFinish' settings in the TextFx animation editor. Both sections have identical controls:

- Emitter OnStart / Emitter OnFinish:** A dropdown menu set to 'None (ParticleEmitter)' with a '0' icon.
- Effect Per Letter?:** A checked checkbox.
- Delay:** A 'Constant' button and a text input field with '0'.
- Duration:** A 'Constant' button and a text input field with '0'.
- Follow Mesh?:** An unchecked checkbox.
- Position Offset:** A 'Constant' button.
- Vector:** Three text input fields for X, Y, and Z, all containing '0'.

On the right side of each section, there is a tooltip that reads: 'zero == "One Shot"'.

Note: Currently, only the legacy Unity particle emitter effects are supported by TextFx. Support for Shuriken Particle System effects will be added in a future update.

Assign a ParticleEmitter effect instance to the **Emitter OnStart** or **Emitter OnFinish** field to have an instantiated instance of that effect played. You can preview the effects in the editor.

Note: It is recommended that you preview your animation in the editor, so that it creates the required number of particle effect instances outside of runtime. See 'Best Practice Tips' for more info!

The following settings are available for each particle emitter you assign.

- **Effect Per Letter?** - Whether to play an effect for each letter or just one effect.
- **Delay** - The amount of time (in seconds) the particle effect will be delayed before playing.
- **Duration** - How long to play the effect for. If set to zero, the effect will play as a "One Shot" effect.
- **Follow Mesh?** - Should the particle effect follow the mesh of the letter it is assigned to?
- **Position Offset** - What position should the particle effect be placed relative to the center of the letters mesh it is assigned to?

7 - Scripting Reference

Classes

- [EffectManager](#)

EffectManager Class

Public Member Functions

void	SetText (string text) Sets up the animation for the supplied text. Can't be called while animation is running.
------	---

void	PlayAnimation () Plays the configured animation on this EffectManager.
------	--

void	PlayAnimation (float delay) Plays the configured animation on this EffectManager. delay : time in seconds to delay the starting of the animation.
------	--

void	PlayAnimation (float delay, OnAnimationFinish animation_callback) Plays the configured animation on this EffectManager. delay : time in seconds to delay the starting of the animation. animation_callback : a callback method which is called when the animation has completely finished. Must be a method with no parameters
------	---

void	PlayAnimation (OnAnimationFinish animation_callback) Plays the configured animation on this EffectManager. animation_callback : a callback method which is called when the animation has completely finished. Must be a method with no parameters
------	---

void	ResetAnimation () Stops and resets the animation back to its starting state.
------	--

void	SetEndState () Sets animation to its end state.
------	---

void	ContinueAnimation () Continues the state of all animations.
------	---

void	ContinueAnimation (int animation_index) Continues the state of the specified animation.
------	--

Public Attributes

<i>[Read Only]</i> bool	Playing A read-only attribute to denote whether the animation is currently playing.
-------------------------	---

bool	Paused Getter/Setter for denoting the pause state of the animation.
------	---

AnimatePerOptions	m_animate_per Denotes the default AnimatePerOption; this says whether the text will animate on a per-letter, per-word or per-line basis. This value can be overridden on individual cases in your animation. [LETTER, WORD, LINE]
-------------------	--

bool	m_begin_on_start Denotes whether the animation is started automatically when the object is first active in the scene.
------	---

float	m_character_size The size of each character (This scales the whole text)
-------	--

TextDisplayAxis	m_display_axis Denotes which axis to draw the text on. [HORIZONTAL, VERTICAL]
-----------------	--

Font	m_font Reference to the Font file to use for the text in the animation. <i>Only available in Unity 4.0 or greater.</i>
-------------	---

TextAsset	m_font_data_file Reference to the bitmap font data text file.
------------------	---

Material	m_font_material Reference to the accompanying font Material to use with the bitmap font data file specified by m_font_data_file .
-----------------	--

float	m_line_height Used for scaling the line height greater or less than their default display size. Default value is 1.
-------	--

float	m_max_width Maximum width that the text can spread before being forced onto a new line. Displayed in the editor scene window with red borders.
-------	---

Vector2	m_px_offset Addition spacing to apply to each letter in your animations text.
---------	---

string	m_text Current text value used for the animation. To change the text of your animation, please call SetText() instead.
--------	---

TextAnchor	m_text_alignment Denotes where the text will align to (Left, Center, Right).
-------------------	--

TextAnchor

m_text_anchor

Which point of the text shares the position of the Transform.

AnimationTime

m_time_type

Denotes whether to use Unity's game time (affected by Time.timeScale), or use realtime, when animating the text. **[GAME_TIME, REAL_TIME]**

8 - Best Practice Tips

Index

- [Running Your Effects From Script](#)
- [Audio And Particle Effect Pre-Computation](#)

Running Your Effects From Script

If you're going to be handling the triggering of your TextFx animations by script, then please consider the following optimisations to give you the best possible performance.

Note: These tips should be applied to any audioclip, particle effect, model animation that you use in your Unity projects, not just TextFx animations.

- **Preload your effects!** - You should instantiate your effects during the loading of your project/scene, and keep a reference to them in code in order to play as and when you need them. This avoids a potential frame rate drop (especially on mobile devices!) which is caused by playing an animation in the same frame as it was loaded into memory.

Wrong way to handle your effects:

```
(Instantiate(text_effect_prefab) as EffectManager).PlayAnimation();
```

Correct way to handle your effects:

```
public EffectManager my_text_effect_prefab;
EffectManager my_text_effect;

// Preload your EffectManager effects in Start function or other
loading function
void Start()
{
    my_text_effect = Instantiate(my_text_effect_prefab) as
    EffectManager;
}

void PlayMyTextEffect()
{
    my_text_effect.PlayAnimation();
}
```

- **Cache the references to your effects!** - Like in the previous tip, you should already have a reference to your EffectManager object in the scene when you wish to play it. You don't want to be searching through your entire scene heirarchy for your effect, in the same frame that you want to play it, otherwise you risk seeing initial frame rate drops, especially on mobile devices.

An example of what **not** to do when you play your animations:

```
GameObject.Find("my_text_animation").PlayAnimation();
```

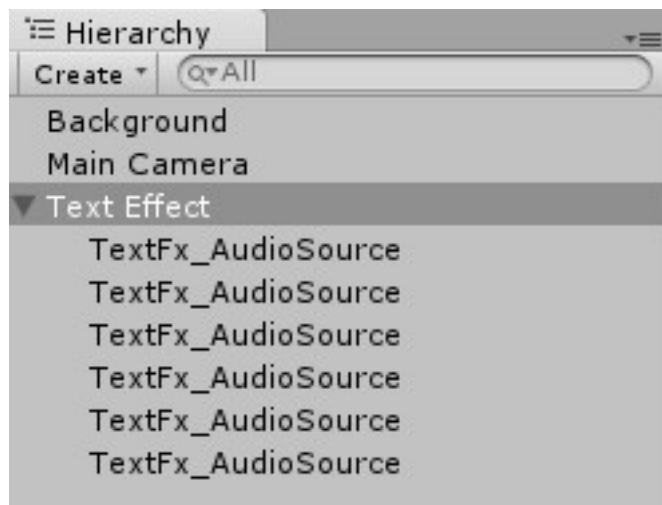

Audio And Particle Effect Pre-Computation

When adding audio and particle effects to your text animation, it is best to run the animation a few times in the editor first.

This allows the correct number of audioclip and particle emitter objects to be created in advance, and stored as child objects of your EffectManager gameobject.

If the audioclips and particle emitters are left to be created at runtime, this can cause frame-rate drops when playing the animation.

If making a prefab of your animation, be sure to make the prefab after it has created the number of child audio and emitter objects it needs for your animation



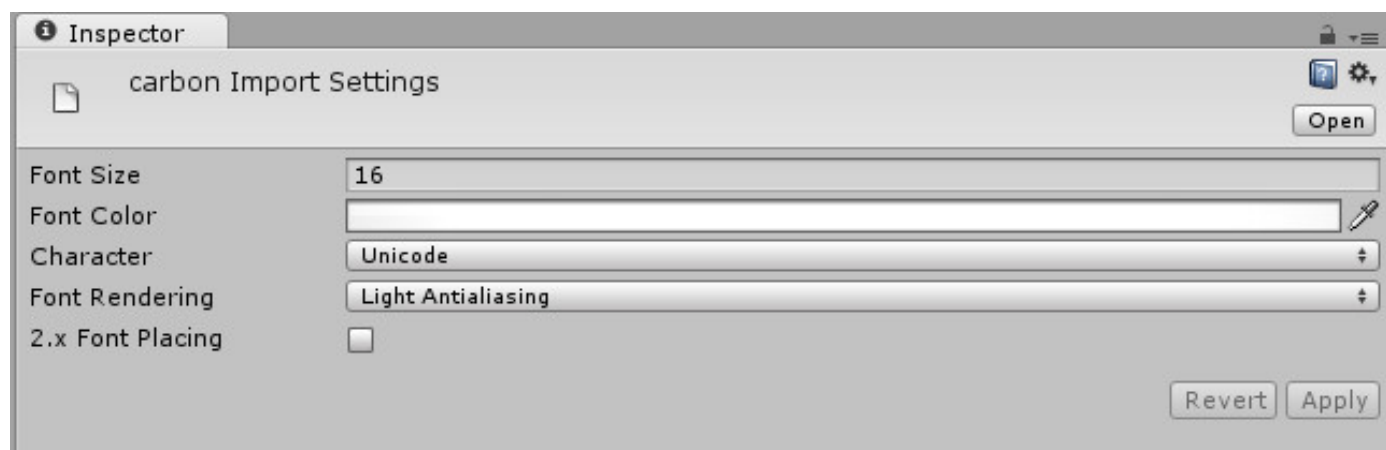
9 - FAQs

Index

- [How do I increase the resolution of my font?](#)

How do I increase the resolution of my font?

If you're using a TTF or other font file type supported by Unity, go to the fonts Import Settings and increase the **Font Size** field. Access a font's Import Settings by clicking on the Font file in the Editor, and looking in the Inspector panel.



If you're using a Bitmap Font, you'll need to generate a new bitmap font based on a larger font size, so that the character texture atlas is larger/higher resolution.

10 - Feedback/Support

Contact Details

If you'd like to do any of the following:

- Offer feedback
- Request a new feature
- Report a bug
- Ask for help with something

Please contact us here: fenderrio@gmail.com

Help A Developer

If you'd like to **show your love for TextFx** and help out the developer massively, you can **leave a rating and review** for the plugin in the AssetStore.

Just access the AssetStore through your Unity Editor, find TextFx, and write down some loveliness...

That makes him one happy cat!

