Hello there!

Thanks for making **PixelBattleText** the new inhabitant of your project :)
Here's how to take advantage of it to animate pixel perfect damage counters, altered states and level ups.

First, take a look at the sample scene. On it, you will find a preview of the animations included in this asset pack.

Open the sample scene, you can find it at:
**Assets** > **PixelBattleText** > **_Example** > **DemoScene**

Hit Play and BE AMAZED!

Demo scene after you press the LEVEL UP button.

Getting Started

Now that you have experienced the power of the powerful and captivating animations that PixelBattleText can create, let's learn step by step how to put together a scene capable of displaying animated text.

Create a new scene and follow the instructions below:

- **1-** Create an empty object and name it *PixelTextController* (The name is not important; it is just to keep things organized)

- **2-** Select the new game object. In the inspector, press the **Add Component** button. Find the PixelBattleTextController component and click it.

> **Note**: There must always be a PixelBattleTextController in the scene for all animated text to work.
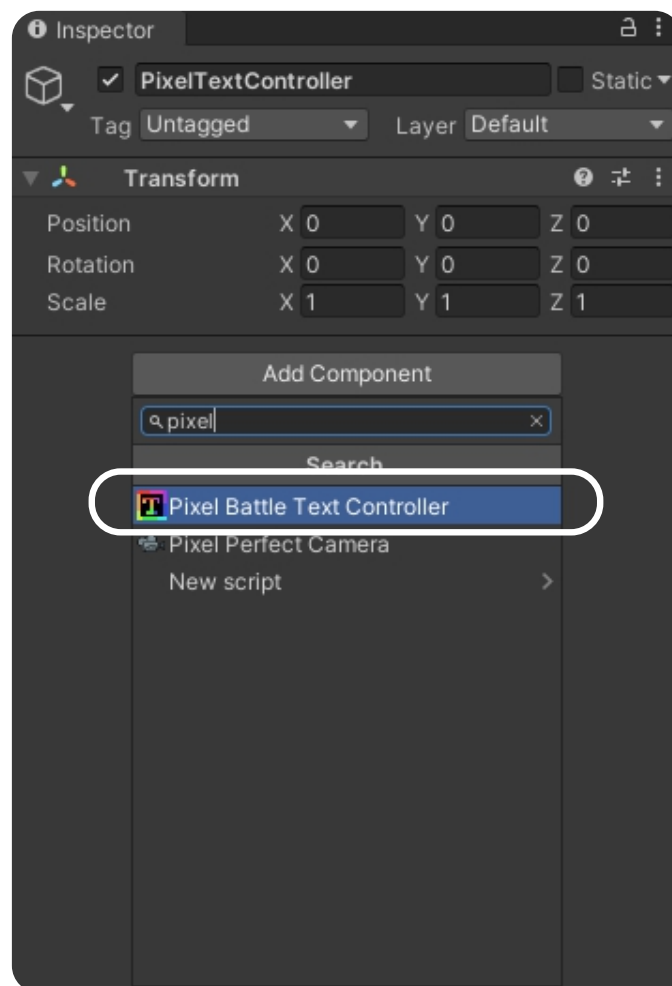
- **3-** Create a Canvas in the menu: **GameObject** > **UI** > **Canvas**, and drag it to the corresponding field in our ⊞**PixelBattleTextController**.
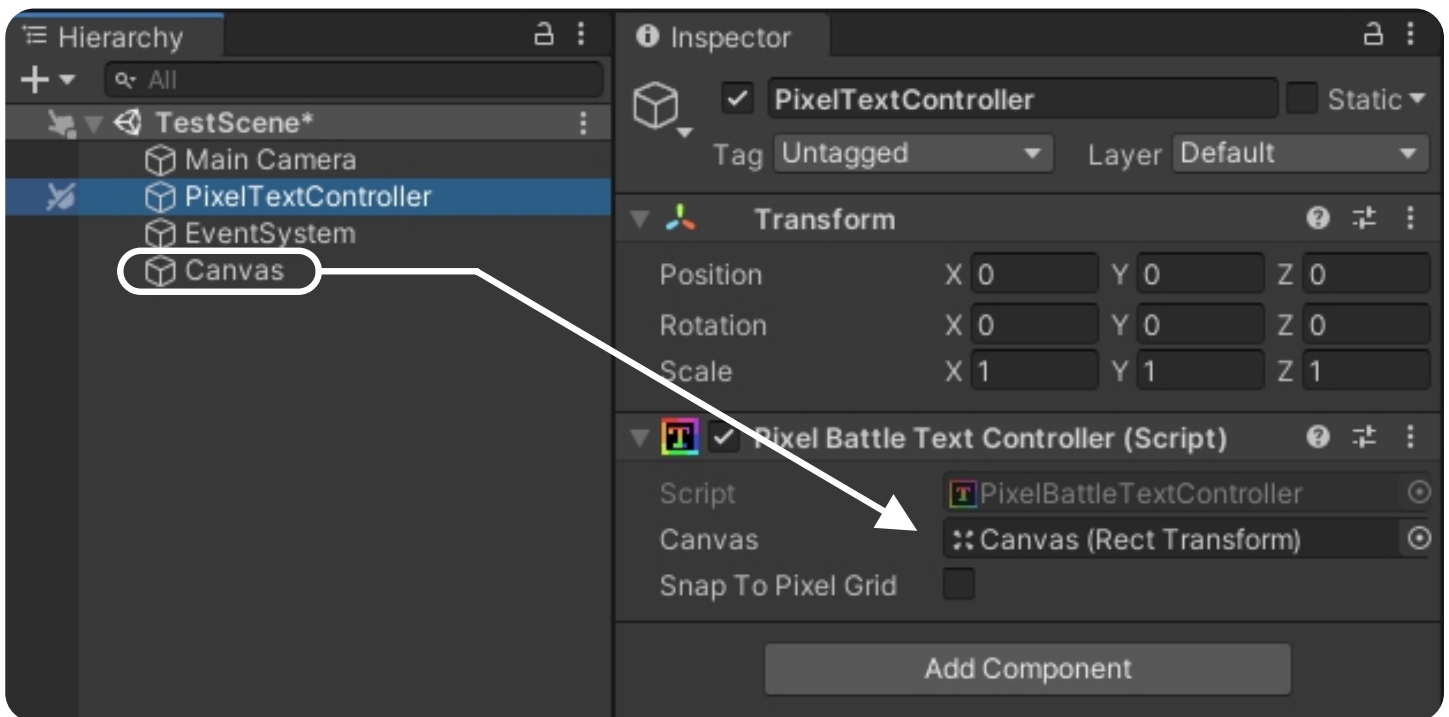

If we try to show pixel perfect text using this canvas as it is, the letters would look very small. An 8 pixel tall font would take up exactly 8 vertical pixels of your screen (wich is very, very small) ⚠️

- **4-** We need to change the ⊞**CanvasScaler** in our *Canvas* game object.
  - Change **UI Scale Mode** drop down to **Scale With Screen Size**.
  - Set **Reference Resolution** to **320**x**180**.
  - Change **Screen Match Mode** drop down to **Match Width or Height**.
  - Set the **Match** slider to **1** (full height).
  - Set **Pixels per Unit** to **32.**
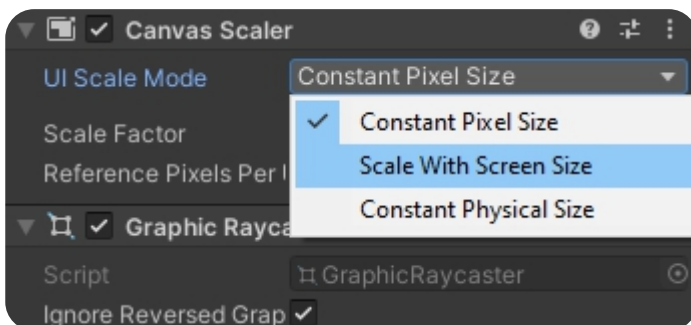
*1- Change the **UI Scale Mode** to **Scale With Screen Size.***
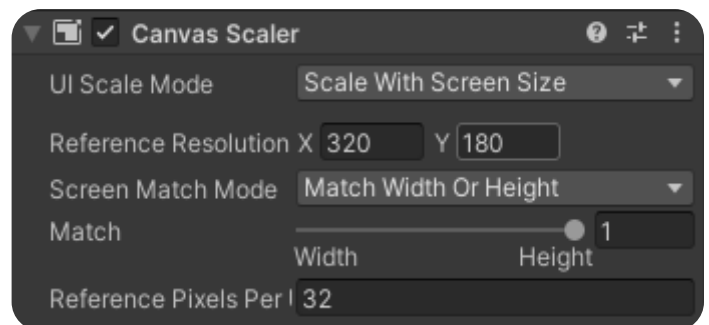
*2- Set your **CanvasScaler** component settings to match these.*

- **5-** Create a script from which you will call the controller. You can do it directly in the same game *PixelTextController* object by pressing *Add Component* and choosing the option *New script*. Name it "*TestingPixelBattleText*" and press *Create and Add.*

- **6-** Open our new script and replace it's content with the following

```csharp
using UnityEngine;
using PixelBattleText;

public class TestingPixelBattleText : MonoBehaviour {
    public TextAnimation textAnimation;

    void Update() {
        if(Input.GetKeyDown(KeyCode.Space))
            PixelBattleTextController.DisplayText(
"Hello World!", textAnimation, Vector3.one * 0.5f);
    }
}
```

- **7-** Now your script should have a public field called *"Text Animation"*. Touch the circle to the right of the field and choose a ▣**TextAnimation** preset. (They are the same as in the demo scene)
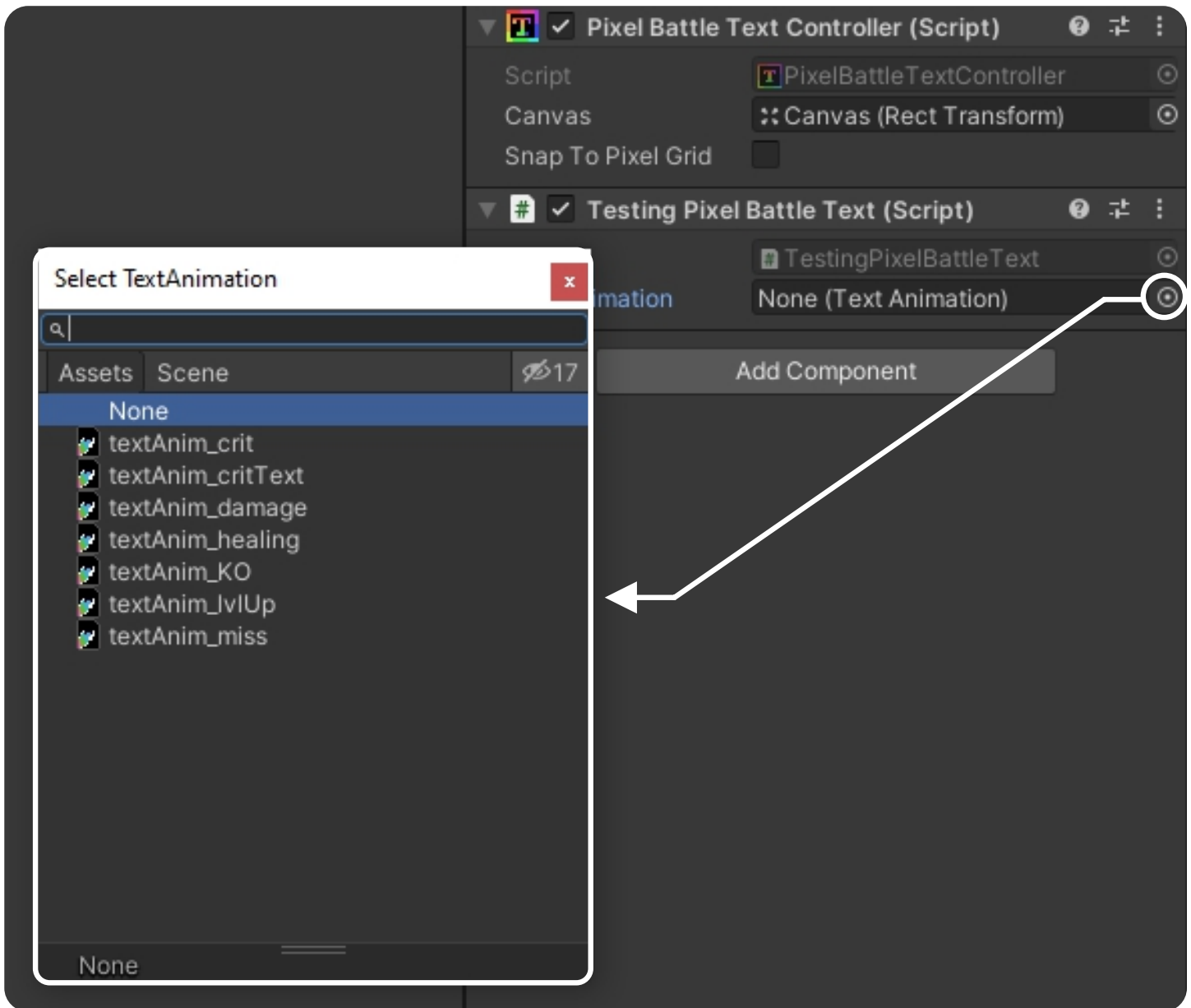
- **8-** With your ▣**TextAnimation** atached and ready, press **Play** in the Unity editor and hit your **space bar**. Pixel perfect text with the words ***"Hello World!"*** will appear in your Game View!

Quick Tip: (You can change the "**Text Animation**" field while the game is running and preview the change by pressing space bar)

Chapter Summary

- You need a **PixelBattleTextController** in order to show cool, animated text in your scene

- Remember that you will need a canvas with a correct scale to display the text in a suitable size.

- All you need to show animated text in, is to call:

```
PixelBattleTextController.DisplayText("meh", animation, position);
```

from anywhere in your scene.

Note: The position we give to the **PixelBattleTextController** is *"relative"*. It means that the position goes from (0,0) to (1,1). X = 0; means the leftmost pixel, X=1 means the rightmost pixel. Y=0 means bottom, while Y=1 means top. If we give the function the position (0.5, 0.5) the text will be displayed in the center of the canvas.

Making your own TextAnimation preset

In case our juicy animation presets are not enough to express your game developer creativity, you can always make your own :)

You can create a TextAnimation asset in the **Project** tab by:

Right Click > **Create** > **PixelBattleText** > **TextAnimation**

You can easily recognize **a TextAnimation** asset by its icon:

As you can see in the figure below, the **TextAnimation** properties are divided into several groups: **Font Controls**, **Time Controls**, **Spacing Animation**, **Offset Animation** and **Color animation**.

In this short chapter we will learn the role of each field in a **TextAnimation** asset and how to "calculate" pixel pivots in order for better understanding pixel perfect rendering for text.
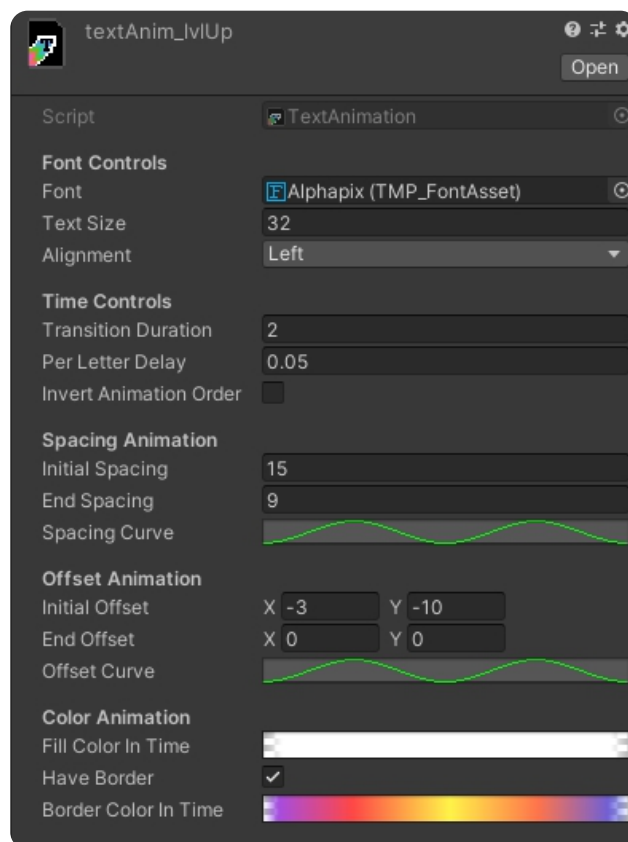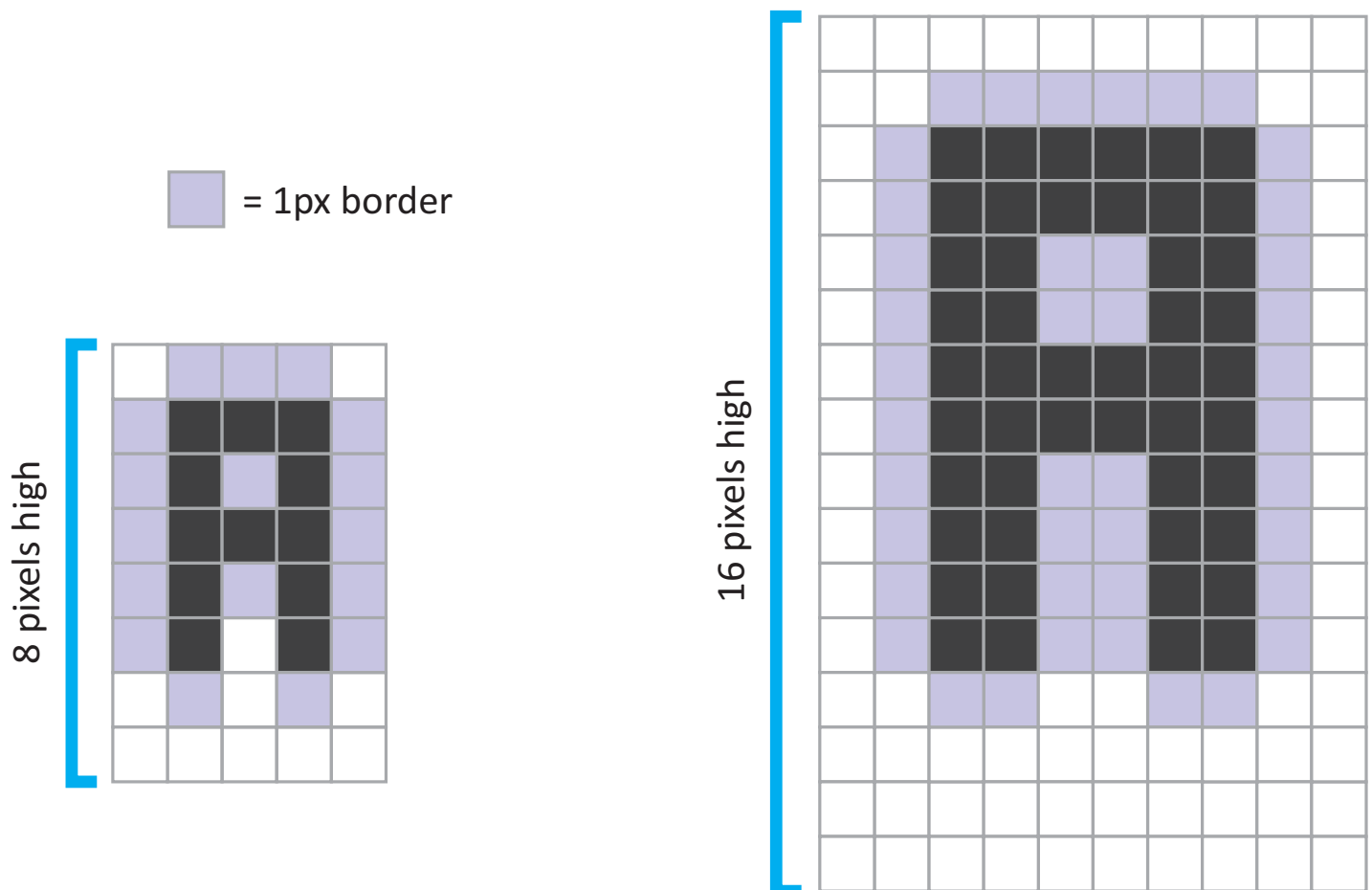
- Font Controls

- **Font**: The **TMP_FontAsset** file for displaying in this animation. this allows to have different animations, each with a different font.

- **Text Size**: The size of the font to display. It changes the **Size** property in the **TMP_Text** component used to represent the animated text

Size changes the height of the font. This includes the empty space below the baseline or above the letter. **Size** is set in "canvas pixels", independent of the actual pixels the font would take on screen. The border will always take one (1) "canvas pixel" on each side, independent of the font size.

The pixel border is kept constant, independent of the font size

= 1px border

8 pixels high

16 pixels high

- Alignment: It controls the horizontal alignment of the font. and the position of the font pivots.

⊕ = letter pivot

Right Alignment

Center Alignment

Left Alignment
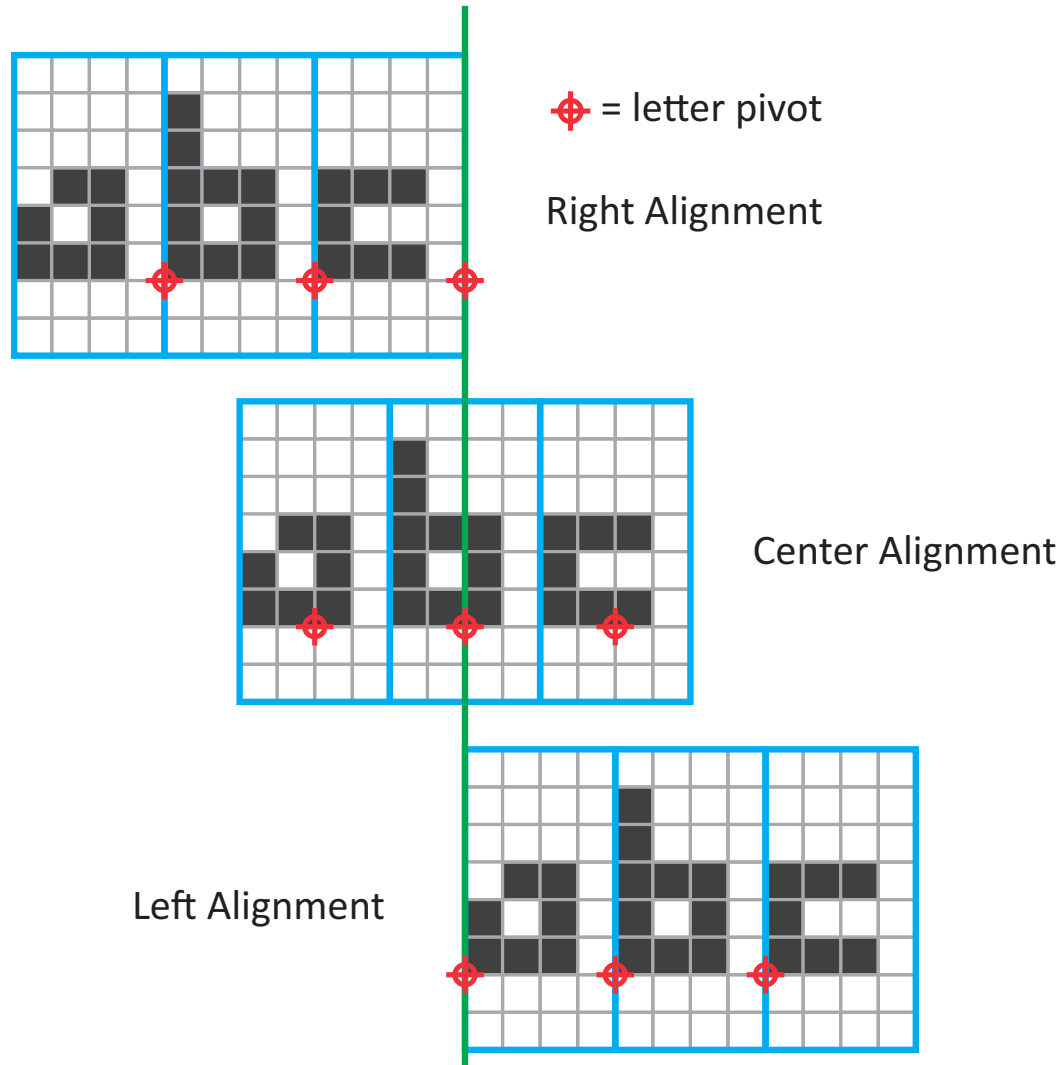
- Time Controls

- Transition Duration: The time it takes a single letter to finish the animation.

- Per Letter Delay: The time it takes for each letter to begin its animation. Each letter must wait for the delay of those in front to end and its own, to begin to animate. This is independent of the duration of the transition. Each letter can start its animation as soon as its delay ends, even if other letters ahead have not finished their animation.

- Invert Animation Order: Determine if the order in which the letters proceed to animate starts with the first or last. It is always taken by "first letter", the one that is further to the left. Regardless of the lineup. Reversing the order of the animation would therefore cause the rightmost letter to always start animating first.

- Spacing Animation

- **Initial Spacing**: Determines the initial space (in canvas pixels) between the pivots of the letters. This space is taken in the animation as the starting point for the position of the letters.

- **End Spacing**: Determines the final space (in canvas pixels) between the pivots of the letters. This space is taken in the animation as the end point for the position of the letters

- **Spacing Curve**: Determines the progress of the animation from initial spacing to end spacing following a curve from 0 to 1, where 0 is the initial spacing and 1 the end.
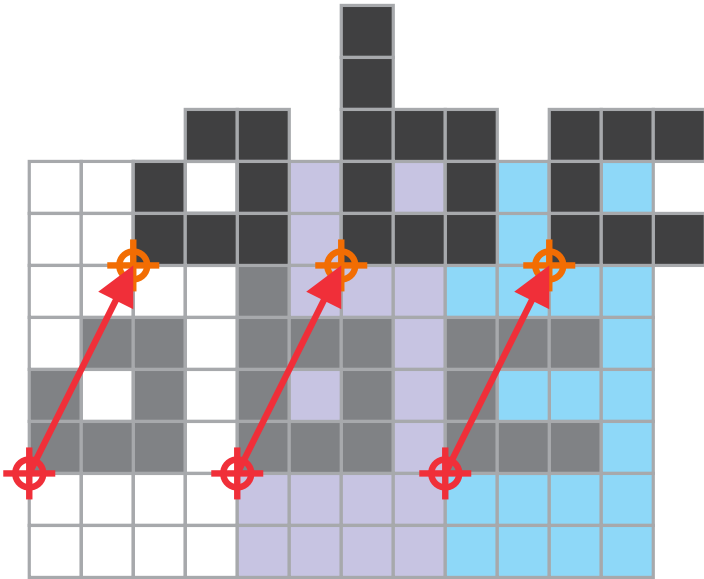

animation

4 px spacing

2 px spacing
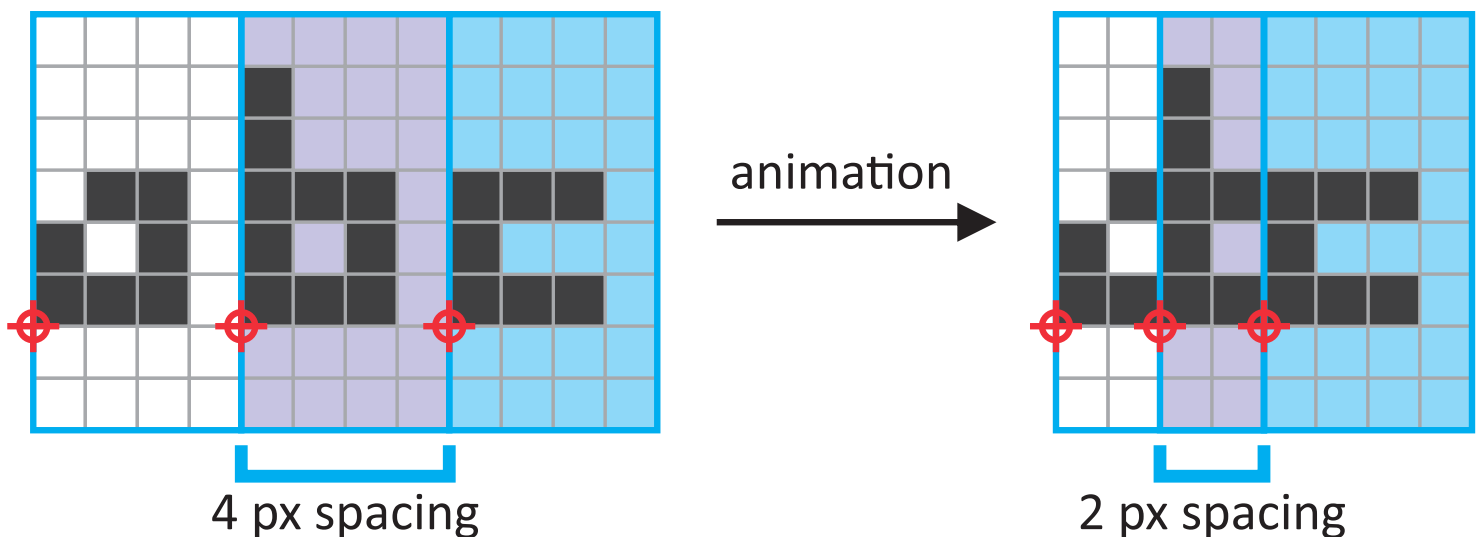
- Offset Animation

- **Initial Offset**: Determines a vector (in canvas pixels) to be added to the pivot at the begining of the animation.

- **End Offset**: Determines a vector (in canvas pixels) to be added to the pivot at the end of the animation.

- **Offset Curve**: Determines the progress of the animation from **Initial Offset** to **End Offset** following a curve from 0 to 1, where 0 is the initial spacing and 1 the end.

Offset changing final position of letters with no delay

Offset changing final position of letters at different moments in time beacuse of letter individual delays
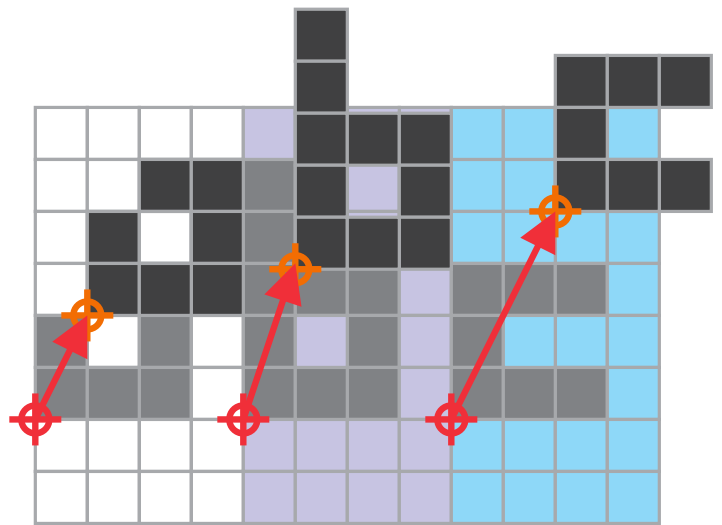
- Color Animation

- **Fill Color In Time**: Determines a color gradient that represents the changing of the letter´s inner color during the animation.

- **Border Color In Time**: Determines a color gradient that represents the changing of the letter´s border color during the animation.

- **Has Border**: Determines whether the edge of the letters should be animated or not even displayed.

Aaaand... That's all folks

Now you have the necessary tools to make the battle text of your next Final Fantasy clone (or a Dragon Quest clone if you are one of those)

An Official
PSYCHIC FURR
Product