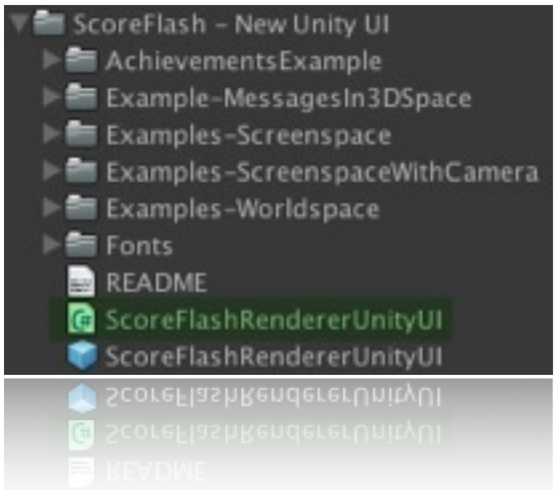
Using ScoreFlash with Unity UI

Starting with version 4.6.0, *ScoreFlash* comes with a folder `ScoreFlash - New Unity UI`. This folder contains everything you need to get you started using ScoreFlash with Unity's new UI system *Unity UI* (formerly known as *uGUI*).

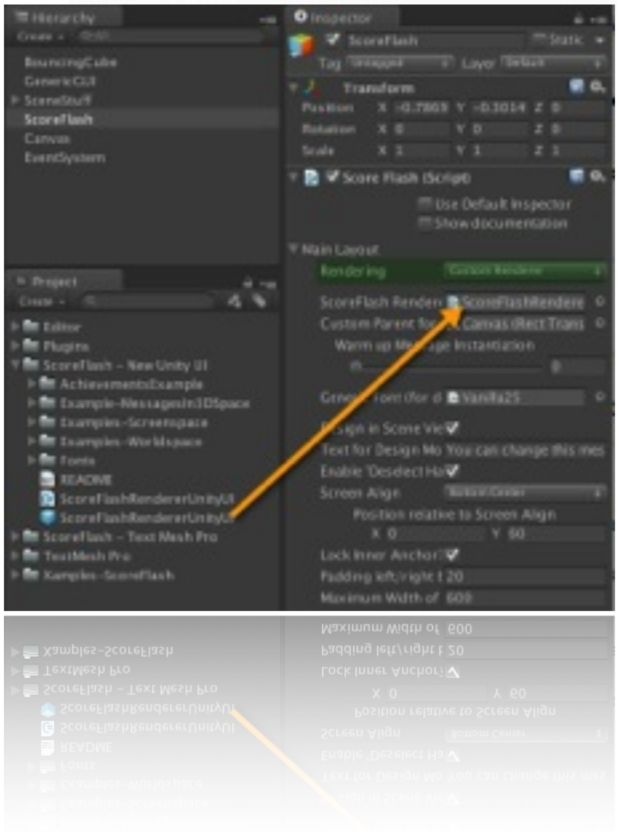
Basic Setup

Most important is the **class/component** `ScoreFlashRendererUnityUI` (see Screenshot) which you can attach to your own renderer prefabs if you want to build more complex setups than the ones delivered with the examples. Of course, you can also use `ScoreFlashRendererUnityUI.cs` as a base for your own implementation, in case you need some special custom handling, like it's done in the AchievementsExample (see the class `AchievementsCustomRendererUnityUI.cs`). For world space messages (or achievements), you might want to consider using `ScoreFlashRendererUnityUI3D` which you find under E`xample-MessagesIn3DSpace`.

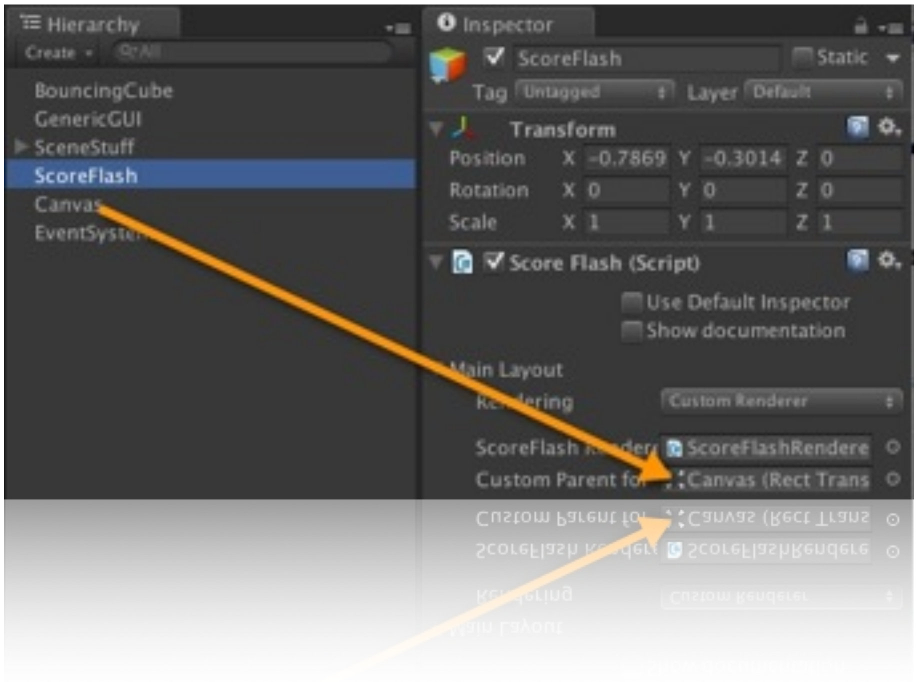


ScoreFlash – New Unity UI
AchievementsExample
Example-MessagesIn3DSpace
Examples-Screenspace
Examples-ScreenspaceWithCamera
Examples-Worldspace
Fonts
README
ScoreFlashRendererUnityUI
ScoreFlashRendererUnityUI

For the most common scenarios, all you need is a copy of the **prefab** `ScoreFlashRendererUnityUI` which you can set up according to your needs (Font, Font Size, effects if needed). You can even add particle systems to your renderer prefabs, as is illustrated by the prefabs and scenes in `Examples-ScreenspaceWithCamera`. To use the new *Unity UI* with ScoreFlash, the first step is to pull your prefab of choice into the slot `ScoreFlash Renderer` (of course, you also need to make sure `Rendering` is set to `Custom Renderer)`. See also the screenshot to the right side of this paragraph.



Furthermore, you also need to assign the correct `Custom Parent for Renderer Instances`, which will usually be your main **`Canvas`**. As is illustrated by the next screenshot:



Using ScoreFlash with different Render Modes

In general, ScoreFlash works best, or at least as one would usually expect it, when using `Render Mode = Screen Space - Overlay` and having `Constant Pixel Size` for `Ui Scale Mode` with `Scale Factor = 1` in the `Canvas Scaler`. However, the *ScoreFlash Unity UI Addon* also has code included that makes adjustments in case you use **scaling**. This has been tested with `Scale With Screen Size` as well as `Constant Physical Size` and different (reasonable) `Scale Factors` for `Constant Pixel Size`. You just need to be cautious that the scaling doesn't make your texts too large or too small.

Of course, you can also use ScoreFlash with `Screen Space - Camera` (in that case you just need to make sure to set the plane distance up properly) or even `World Space`. With `World Space`, however, you need to understand that ScoreFlash simply has no way to properly map the locations of 3D objects to its messages, so `ScoreFlashFollow3D` will usually not work in that case - at least not properly. But you can achieve some really interesting effects using *World Space* canvases, as is illustrated by the examples in `Examples-Worldspace`.

3D-Space Messages

Finally, *ScoreFlash* now also supports **real 3D-space messages** as you would usually use them in VR-environments. The way this is done is that the rendering prefab includes the Canvas. For this purpose, a different implementation of the rendering component is needed: Usually, you'll simply use `ScoreFlashRendererUnityUI3D` (which you find under `Example-MessagesIn3DSpace`). Notice in the example prefabs (`ScoreFlashRendererUnityUI3D-Close` and `ScoreFlashRendererUnityUI3D-Distant`) that you might have to set up the scale appropriately to prevent scaling artefacts. Notice also that in this case, you don't need to assign a `Custom Parent for Renderer Instances` because ScoreFlash can simply add those objects as a child of the main ScoreFlash instance.

Building Renderer Prefabs from Scratch

When building your own prefabs from scratch, there's a few things to be aware of: Usually, you'll want a *top left anchor* and *pivot* for the main object of the prefab. When checking `Fix Pivot` and `Fix Alignment`, this will be automatically corrected in case it was set up incorrectly and it will also be adjusted according to the layout set up in `ScoreFlash` or `ScoreFlashLayout`. The exception to this rule is 3D space messages which work better with a *middle center anchor* and a *centered pivot*.

If your prefab is primarily just a `Text` object, you'll usually want to have a `ContentSizeFitter` with `Horizontal Fit` set to `Unconstrained` and `Vertical Fit` set to `Preferred Size`. When you're adding backgrounds (like in the Achievements example), you'll usually want to have your own implementation of `ScoreFlashRendererBase` that implements `GetSize()` according to your specific needs. In those cases, the width set up in `ScoreFlash` (or `ScoreFlashLayout` or `ScoreFlashFollow3D`) will usually be ignored.

If there's anything you're struggling with - please let me know:
Email: issues@narayana-games.net
Unity Forum: http://bit.ly/ScoreFlashUnityForum