Style Profile

Intro

A style profile is a scriptable object containing variables to define a style. It was initially created for easy customization of the visuals in graphical user interfaces (GUI) but is now much more than just that. It can be used for any system thanks to its generic approach to variables. Beyond the customization it is non destructive thanks to its effect being applied at runtime. Any changes to your style profile will instantly be shown in the game making it more flexible and powerful in certain scenarios than a prefab based approach.

Core Concept

A style profile works in conjunction with style listeners. Style listeners are scripts that listen for changes of specified variables in a style profile and process the new variable value accordingly. An example is the color of an image component. For that we can use a color variable and a color based style listener. If the color variable on the profile changes the image color will be changed as well.

Creating A Style Profile

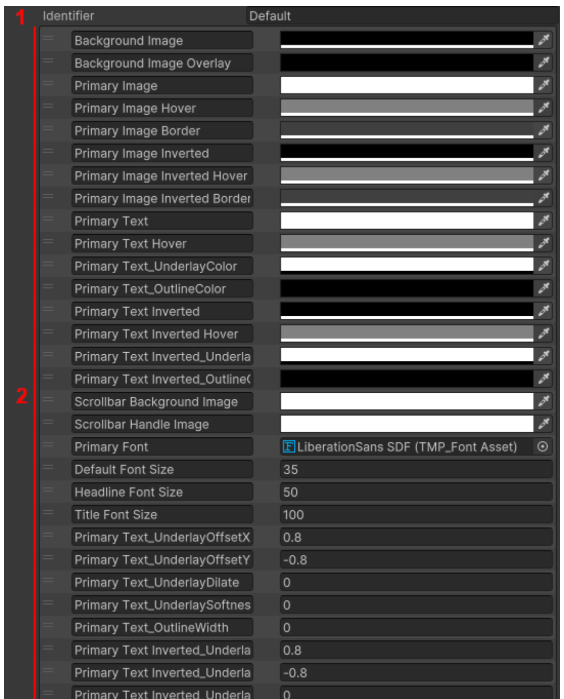
As the style profile is a scriptable object it can be created in the project window with:
Right Click > CitrioN > Style Profile > New Style Profile

Style Profile Overview

A style profile consists of an identifier and a list of variables with their associated values.

Identifier (1)

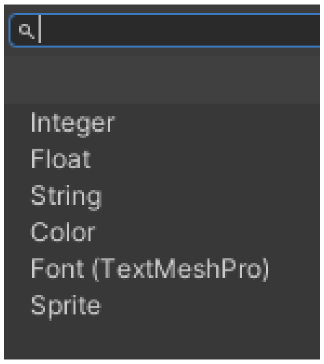
Any string you would like to represent your profile. Can be used to exclude or include a specific profile on a listener. You can for example have your listener only react to a profile that matches a specific identifier. It does not necessarily have to be unique. If multiple profiles have the same identifier they will both affect listeners that listen for profile changes with the specified name.



Identifier	Default
Background Image	⬛
Background Image Overlay	⬛
Primary Image	⬜
Primary Image Hover	▨
Primary Image Border	▨
Primary Image Inverted	⬛
Primary Image Inverted Hover	▨
Primary Image Inverted Border	▨
Primary Text	⬜
Primary Text Hover	▨
Primary Text_UnderlayColor	⬜
Primary Text_OutlineColor	⬛
Primary Text Inverted	⬛
Primary Text Inverted Hover	▨
Primary Text Inverted_Underla	⬜
Primary Text Inverted_Outline(	⬛
Scrollbar Background Image	⬜
Scrollbar Handle Image	⬜
Primary Font	LiberationSans SDF (TMP_Font Asset)
Default Font Size	35
Headline Font Size	50
Title Font Size	100
Primary Text_UnderlayOffsetX	0.8
Primary Text_UnderlayOffsetY	-0.8
Primary Text_UnderlayDilate	0
Primary Text_UnderlaySoftnes	0
Primary Text_OutlineWidth	0
Primary Text Inverted_Underla	0.8
Primary Text Inverted_Underla	-0.8
Primary Text Inverted_Underla	0

Variables (2)

Your list of variables are the heart of the style profile. They all have a key/identifier on the left and the associated value on the right. By default variables for Integer, Float, String, Color, TextMeshPro Font Asset and Sprite can be created. It is very easy to add more types so the profile can be compatible with anything you like in your project.

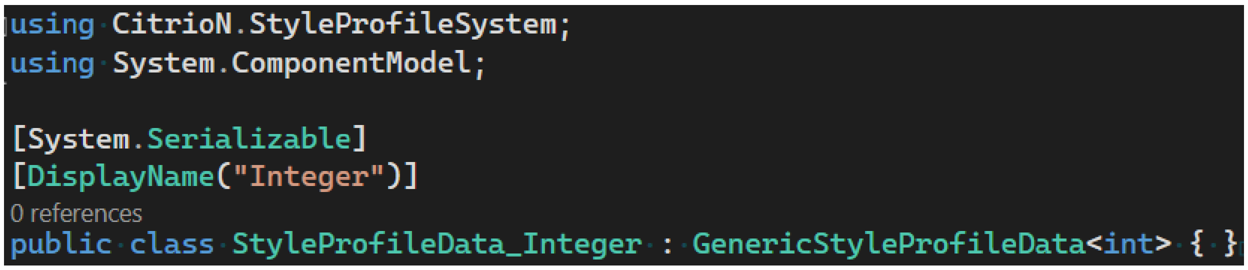


Add/Remove Buttons (3)

With the '- button' you can remove the selected variable entry from the list. The '+ button' will open an advanced dropdown window with options to choose your variable type. How to add new types to this is explained in the following section.

Adding Custom Variable Types

To make a new type compatible with a style profile you need to create a new script and create a derived class from GenericStyleProfileData with your type specified like shown below with the int type. Make sure to include the CitrioN.StyleProfileSystem namespace. If the namespace can not be found you are most likely missing the assembly definition reference in your own assembly definition file. To give your type a custom display name in the style profile you can use the DisplayName attribute. Your custom type should now be available in any style profile.



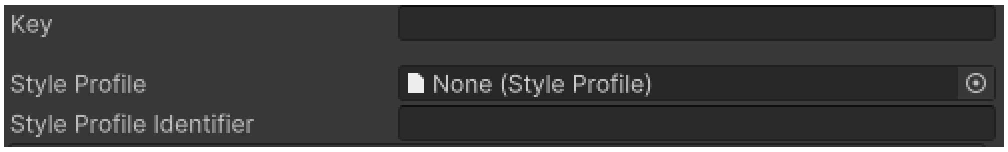
```
using CitrioN.StyleProfileSystem;
using System.ComponentModel;

[System.Serializable]
[DisplayName("Integer")]
0 references
public class StyleProfileData_Integer : GenericStyleProfileData<int> { }
```

If you want to have your type(s) inside a specific directory you can use the MenuPath attribute.

```
[CitrioN.Common.MenuPath("My Types")]
```

Style Listener Overview

A style listener is a script that can be attached to a GameObject. It will listen to changes to a specified variable and process the new variable. Example would be to change the color of an image or the text size of a text component.



Key

Style Profile: None (Style Profile)

Style Profile Identifier

Key

The key/identifier for the variable specified in your profile

Style Profile (Optional)

A reference to a style profile scriptable object. If specified only changes on the referenced profile to variables matching the key above will be processed by the listener.

Style Profile Identifier (Optional)

If no style profile is referenced the identifier can be used to specify which style profile(s) should be processed by the listener. Only changes in style profiles with the specified name will be processed. Using the identifier instead of a hard reference is more flexible because it makes replacing a style profile with another one easy if they have the same name. If neither a style profile nor an identifier is specified any profile will be valid for processing.

OnStyleChanged (Optional)

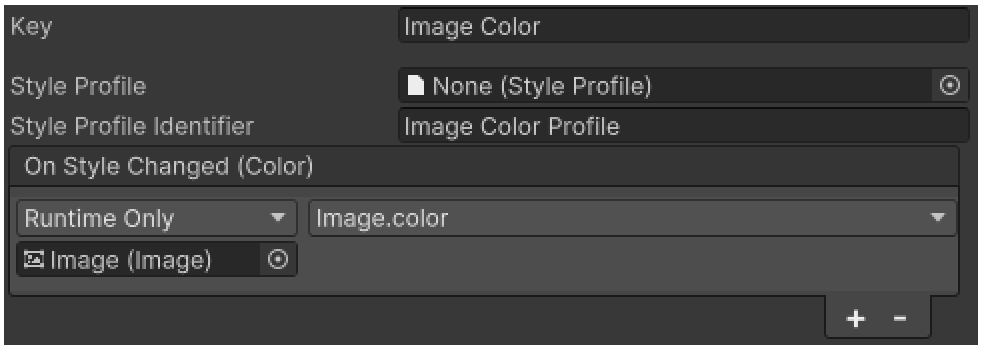
The OnStyleChanged Unity event allows you to specify your functionality to invoke when the respective variable on a matching style profile changes. You can find more information and an example for this in the next section.

Hint

If both a style profile and an identifier are assigned the listener will first check for the matching profile and then for the identifier if the profile didn't match.

How To Use A Style Listener

To use a style listener simply attach the style listener of your choice to a GameObject. Let's take the style listener for a color as an example:



Key	Image Color
Style Profile	None (Style Profile)
Style Profile Identifier	Image Color Profile

On Style Changed (Color)

Runtime Only — Image.color

Image (Image)

The key, style profile and style profile identifier fields were already covered in the previous section so we will focus on the event. The event will have the color from the style profile as the parameter. In this example we can simply drag an image component into the list and assign the image color to be changed. In our example only a color variable named **Image Color** on a style profile with the identifier **Image Color Profile** will affect this listener. You can use the same approach for all the other style listeners included. Some of them don't even require you to specify an event such as the listener for the Image Color which is an extension of the listener from our example with additional color modification options.

Available Style Listeners

Basic Types

There are listeners for all provided types. Those are *bool, int, float, string, Color, TMP Font Asset and Sprite*.

TextMeshPro Text - Float

Convenience script for float variables on a TMP text component. Can also be achieved using a basic float listener.

TextMeshPro Text - Color
Convenience script for changing the text color of a TMP text component.

Image - Color
Adds additional color changing options to the color received from the profile before applying it. This can be useful if you want to slightly lighten/darken or desaturate the received color.

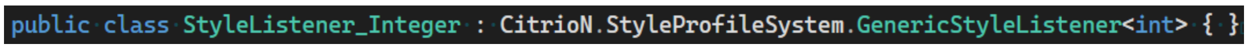
Image - Sprite
Convenience script for changing the sprite of an image component.

TextMeshPro Material - Float
Allows the change of a material float variable on a TMP text component such as the outline width.

TextMeshPro Material - Color
Allows the change of a material color variable on a TMP text component such as the outline color.

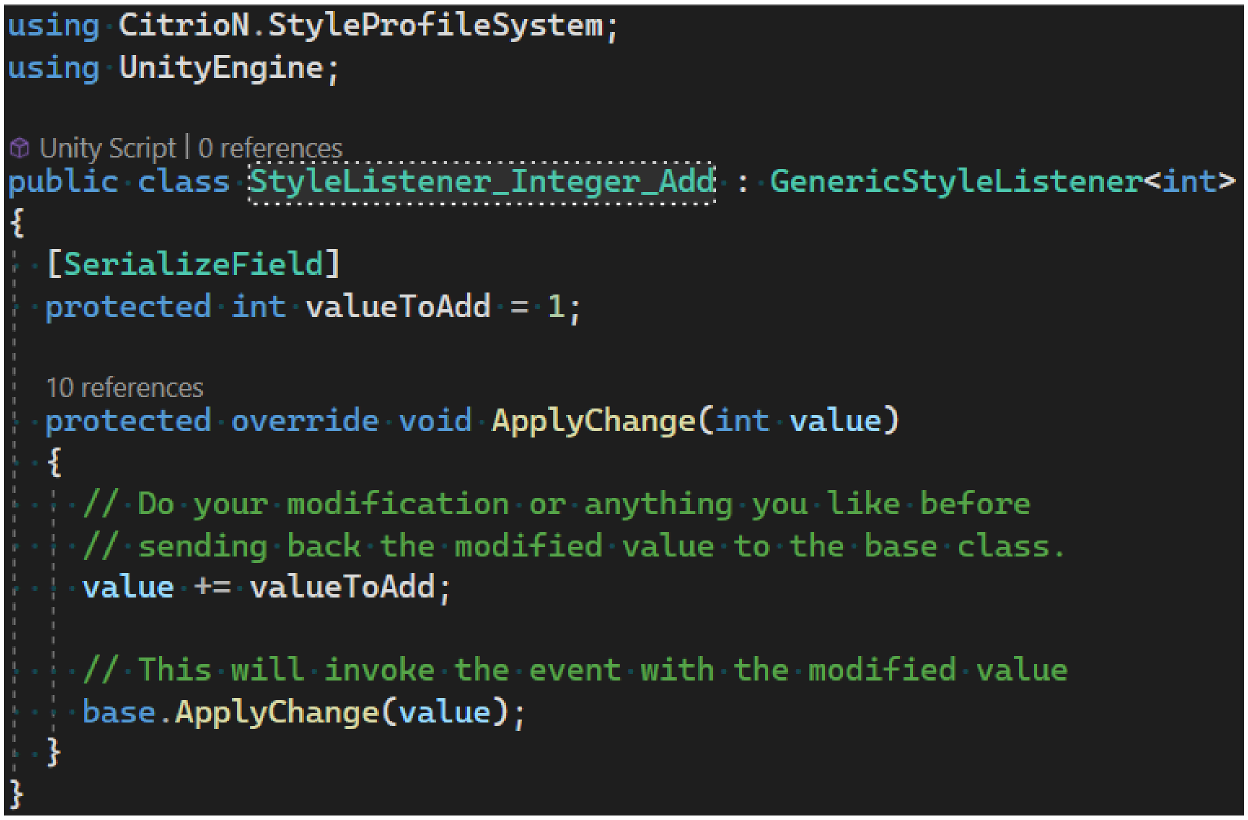
UI Toolkit
A style listener dedicated to the changing of variables on UI Toolkit visual elements.

How To Create New Style Listeners
Creating a new style listener for any type is as simple as creating a new class like this:



```
public class StyleListener_Integer : CitrioN.StyleProfileSystem.GenericStyleListener<int> { }
```

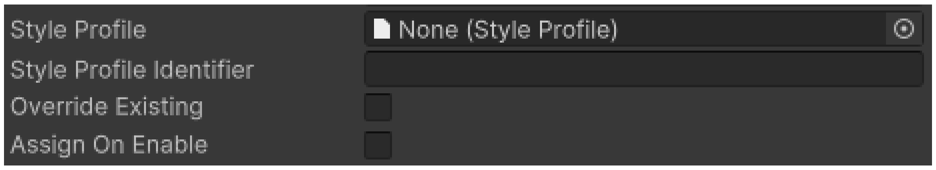
You can of course go much further than this and add your own variables to your listener class and modify the received value in any way you like before the events are invoked.
In the following example we simply add the value specified in the valueToAdd variable to our value received from the style profile. The OnStyleChanged events will now use the modified value.



```
using CitrioN.StyleProfileSystem;
using UnityEngine;

Unity Script | 0 references
public class StyleListener_Integer_Add : GenericStyleListener<int>
{
    [SerializeField]
    protected int valueToAdd = 1;

    10 references
    protected override void ApplyChange(int value)
    {
        // Do your modification or anything you like before
        // sending back the modified value to the base class.
        value += valueToAdd;

        // This will invoke the event with the modified value
        base.ApplyChange(value);
    }
}
```

Other Useful Scripts

AssignStyleProfileToListenersInHierarchy



Style Profile	None (Style Profile)
Style Profile Identifier	
Override Existing	☐
Assign On Enable	☐

Can specify a style profile or style profile identifier to be added to any style listener in its hierarchy. This is useful if you don't want to specify any profile or identifier on your prefabs. By using this script you can assign your style profile to listen for at runtime for all its children. This approach is recommended because having an identifier or profile reference assigned in your prefabs makes it cumbersome to change later.

Style Profile
The style profile to assign to all style listeners in the child hierarchy of this script.

Style Profile Identifier
The style profile identifier to assign to all style listeners in the child hierarchy of this script.

Override Existing
Can be used to override or skip already existing profiles and identifiers on style listeners.

Assign On Enable
If the profile/identifier should be assigned when this script gets enabled.

RegisterStyleProfile
Registers a style profile for runtime queries, essentially making it available to style listeners. If working with direct profile references this script is not needed. But in the case of using style profile identifiers (which is the recommended approach) it may be required for your profile to be registered before your listeners attempt to query values from it.

Style Profile
The style profile to register for runtime queries.