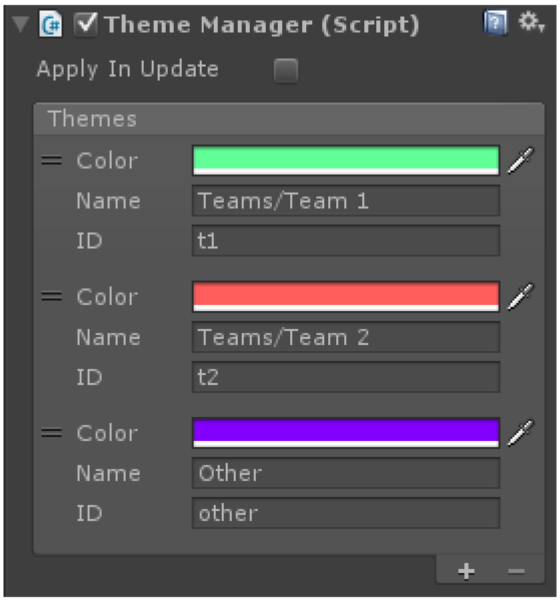
THEMO

GETTING STARTED

Themo allows you to define color themes, and then use those themes however you see fit throughout your game, for things like team or UI colors. Themo's backbone consists of two easy to use MonoBehaviours.

Theme Manager


Theme Manager (Script)

Apply In Update ☐

Themes

Color	🟩
Name	Teams/Team 1
ID	t1
Color	🟥
Name	Teams/Team 2
ID	t2
Color	🟪
Name	Other
ID	other

The **Theme Manager** is where you can define your themes. As you can see, they are kept in a reorderable list, and each Theme has 3 fields.

Color: Color of your Theme. Duh.

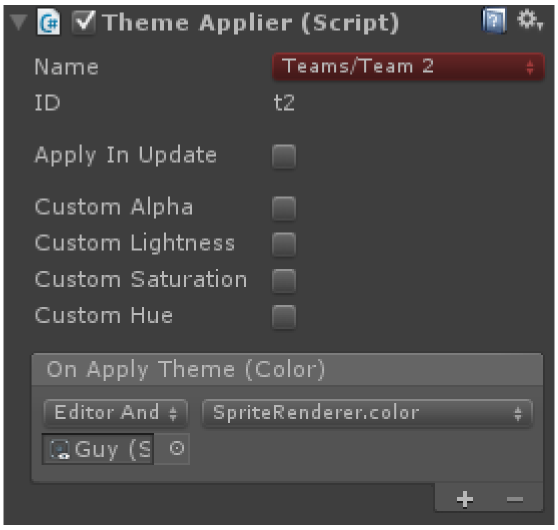
Name: The visible name of your Theme. This name is used for the popup list in the **Theme Applier** (see below). Thanks to how Unity's popup lists work, you can easily create submenus by using forward slashes!

ID: This is the actual identifier that is used by **Theme Applier**s while fetching their base colors before calculations. The reason Themes have both a Name and an ID is for mere organizational convenience. Once you create a Theme in the **Theme Manager**, you can drag it around in the list, and rename it as many times as you like, but an ID should always stay the same. For your convenience, for each new Theme you add to the list, a unique ID will be created for you. You can use it, or define your own ID. An ID doesn't have to be complex. All that matters is that they be unique. Even “1, 2, 3” will do!

As you can see, there's a final field in the editor.

Apply In Update: The “Apply In Update” option, when true, will apply all changes made to every Theme Applier in the scene.

Theme Applier



Theme Applier (Script)

Name	Teams/Team 2
ID	t2
Apply In Update	☐
Custom Alpha	☐
Custom Lightness	☐
Custom Saturation	☐
Custom Hue	☐

On Apply Theme (Color)

Editor And | SpriteRenderer.color

Guy (S

A **Theme Applier** is what actually takes the colors from the **Theme Manager**, and applies them to something. This component has many fields you can tweak to your desire.

Name: This is where you can pick the Theme you previously defined in the **Theme Manager**. Note that there will be an error box here if you do not have a **Theme Manager** in the scene.

ID: You will not be changing the ID manually unless there is an error. This is merely here to let you know which ID the Name field corresponds to.

Custom [------]: These fields allow you to adjust the final output of your Theme's color. So each **Theme Applier** can have variants of one theme color.

Apply In Update: By default, changes you make during runtime to the Theme Applier will not be applied immediately. The “Apply In Update” option, when true, will apply all changes made to this particular Theme Applier.

OnApplyUpdate: This is a UnityEvent. Thanks to how UnityEvents work, you can subscribe literally anything that takes a Color argument! Make sure that when you add a listener, the call mode is set to “Editor And Runtime.”

API

Both **Theme Manager** and **Theme Applier** have small, and self-explanatory interfaces. There are two important functions you should know about if you are going to be using Themo through scripts at runtime. By default, Themo assumes that you define your themes in the editor, and don't change them during runtime. If you wish to modify any themes during runtime, you must either call the ApplyThemes() function of the **Theme Manager**, or the ApplyTheme() function of the particular **Theme Applier**

`ThemeApplier.ApplyTheme()`
This function will apply any changes you have made to the **Theme Applier**. If you set ApplyInUpdate to true, this will be called for you in every Update loop.

`ThemeManager.ApplyThemes()`
This will call the ApplyTheme() function of all **Theme Applier**s in the scene.