Runtime UI

UiToolkitMenu

Version 1.0.0

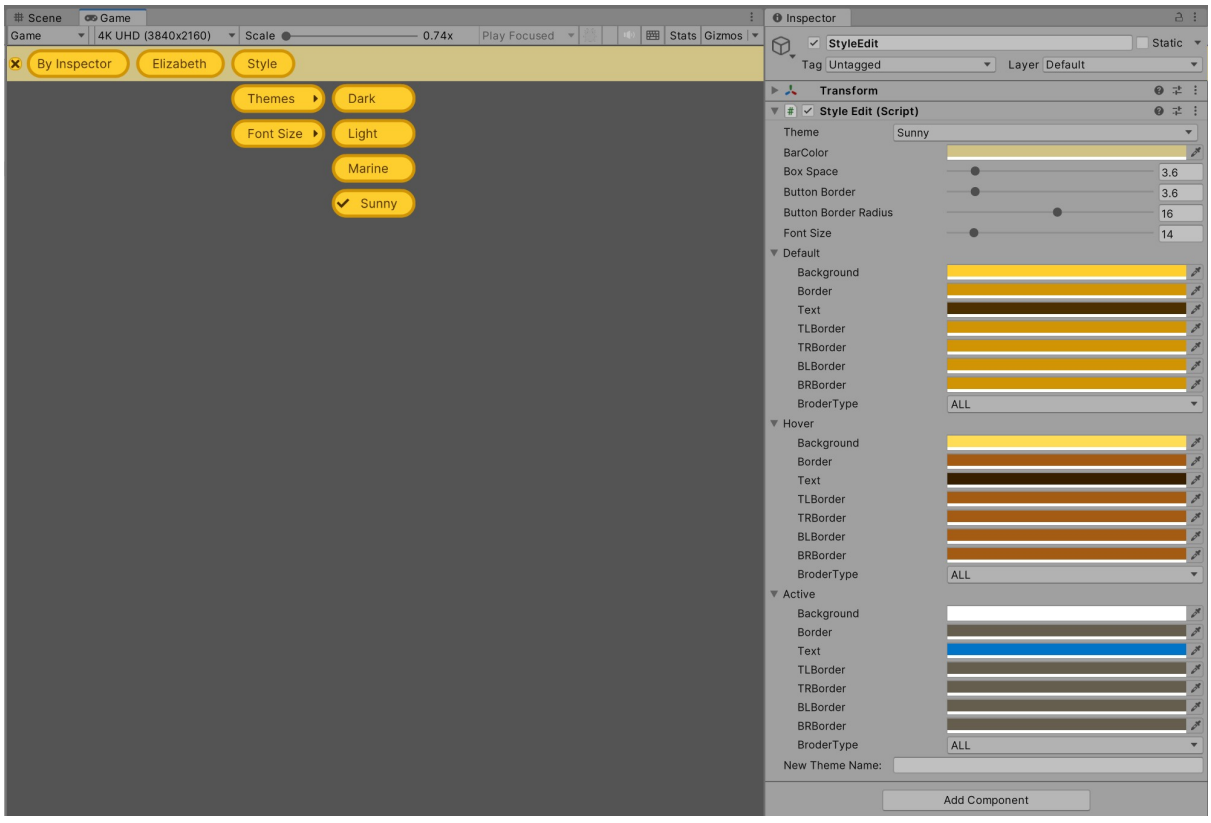

Table of contents

1. Overview

This asset let's you create a classic Menu Bar, by inspector or by code, using UI Toolkit (https://unity.com/features/ui-toolkit), the toolkit expected to become the recommended system for new UI development projects.

Menu Bar is often used in desktop apps and is for display an organized list of commands or options, and save space by hiding until the user needs them (https://learn.microsoft.com/en-us/windows/apps/design/controls/menus-and-context-menus). An example of a bar menu is shown in the following figure:

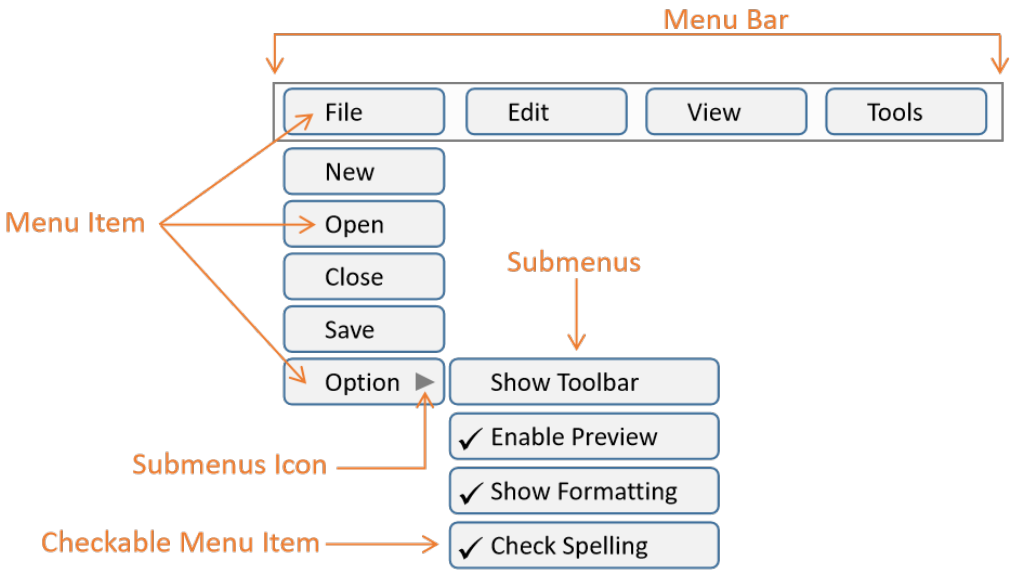

Figure 1 - A Menu Bar

2. Requirements

Unity 2022.3.0 or higher: previous versions report an incorrect layout format.

3. Features

- Hierarchical Structure: you can create main menu items and assign them children, grandchildren, ...
- Events: for each menu item you can define events for clicking, checking or un-checking
- Check Group: you can define a group of menu items that work like radio buttons
- Styling: you can define font size, border size, border-radius, and appearance for default, hover, and active states
- Themes: 4 themes are already implemented; you can create new ones and add them to the presets list
- Status Bar: you can report messages relating to the status of your app
- Display: you can show/hide or expand/contract, on the fly, both MenuBar and StatusBar
- Responsive: menu items adapt to any screen size, ensuring consistency across layouts

4. Using

1) Make sure there is an event system component in your scene; otherwise you can add one by selecting menu item GameObject > UI > EventSystem
2) drag and drop the UiToolkitMenu prefab which you can find in UiToolkitMenu/Prefabs

The menu items can be created by the inspector panel or by code.

4.1. Create menu items by inspector panel

Go to MenuBarController component of MenuBarDocument GameObject (Figure 1)

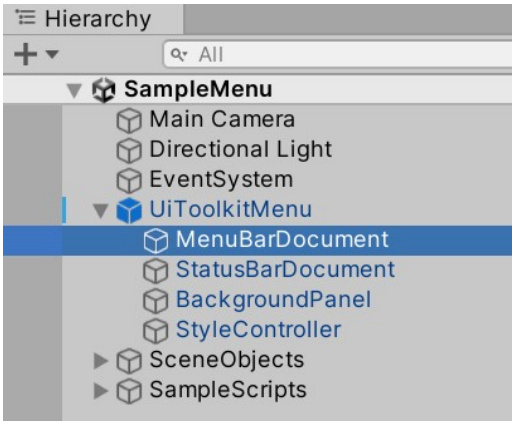

Figure 2

1) Enter, in the MenuItems field, the number of main menu items you want in your MenuBar.
2) Expand Element 0
3) Enter, in the Item Text field, the label you want to appear in this menu item (e.g. “By Inspector”)
4) Enter, in the Children field, the number of children menu items you want for “By Inspector”.
5) Expand Children Element 0
6) Initializes Item Text field and, optionally, Is Checkable, Checked, Check Group and Disabled.
7) Add, optionally, EventOnClick and or EventOnUnCheck
8) Repeat operations from 5 to 7 for each child of menu item “By Inspector”
9) Repeat operations from 2 to 7 for main menu items you want in your MenuBar.

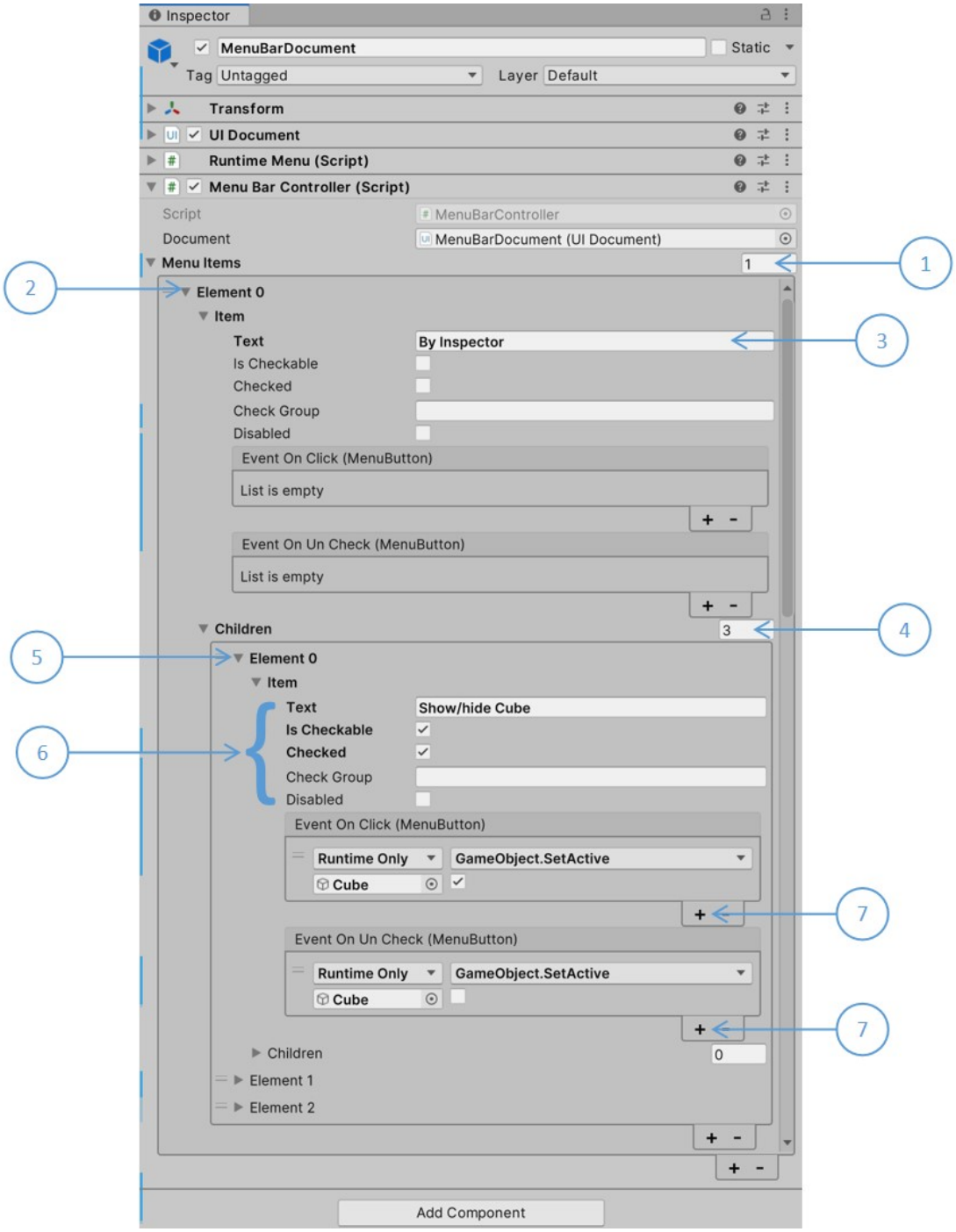

Figure 3

At the end the menu may look like this (Figure 3):

Figure 4

4.2. Create menu items by code

To create a menu item by code you need to use the RuntimeMenu class; many examples on how to create menu item by code can be found in “Examples/Script/EditMenu”: let's now report some steps of this script.

First of all you must be sure that the MonoBehavior RuntimeMenu has been successfully started before you can use it. To do this you can use:

```
IEnumerator Start()
{
    yield return new WaitWhile(() => RuntimeMenu is null);
    EditMenuBar();
    ....
}
```

and EditMenuBar() may have the following instructions:

```
private void EditMenuBar()
{
    /*
        Let's create the following menu hierarchy on the fly:

        Elizabeth
            Charles
                William
                Hanry
            Anne
                Peter
                Zara
            Andrew
                Beatrice
                Eugenie
            Edward
                Luise
                James
    */

    // Let's create the root MenuButton
    var Elizabeth = RuntimeMenu.Add("Elizabeth");

    // Let's add child "Charles" to Elizabeth using Add(string, MenuButton)
    RuntimeMenu.Add("Charles", Elizabeth);

    // Let's add child "William" to "Charles" using Add(string, string)
    RuntimeMenu.Add("William", $"Elizabeth{Constants.PathSeparator}Charles");

    // Let's add child "Hanry" to "Charles" using Add(string, string[])
    string[] path = { "Elizabeth", "Charles" };
    RuntimeMenu.Add("Hanry", path);

    var Anne = RuntimeMenu.Add("Anne", Elizabeth);
    RuntimeMenu.Add("Peter", Anne);
    var Zara = RuntimeMenu.Add("Zara", Anne);

    // Let's add onClick callback to Zara
    Zara.Item.EventOnClick.AddListener(StatusBarCallback);

    var Andrew = RuntimeMenu.Add("Andrew", Elizabeth);

    // Let's create MenuButton Beatrice without hierarchy
    var Beatrice = new MenuButton("Beatrice", isCheckable: true);
    // Let's add OnClick callback to Beatrice
    Beatrice.Item.EventOnClick.AddListener(StatusBarCallback);
```

```
    // Let's add OnUnCheck callback to Beatrice
    Beatrice.Item.EventOnUnCheck.AddListener(StatusBarCallback);
    // Let's add child Beatrice to Andrew using Add(MenuButton, MenuButton)
    RuntimeMenu.Add(Beatrice, Andrew);

    // Let's add child "Eugenie" to Andrew using Add(MenuButton, MenuButton)
    RuntimeMenu.Add(new MenuButton("Eugenie"), Andrew);

    var Edward = RuntimeMenu.Add("Edward", Elizabeth);
    // Let's add children "Luise" and "James" to Edward
    // using AddMenus(List<string>, MenuButton)
    RuntimeMenu.AddMenus(new List<string> { "Luise", "James" }, Edward);

    // Let's add a callback to MenuButton, of Elizabeth hierarchy,
    // which do not have any callback
    AddListnerToHierarchy(Elizabeth);

    // Let's create "Style" root MenuButtons
    var Style = RuntimeMenu.Add("Style");

    // Let's create "Themes" as parent of a group of checkable buttons
    var themes = new MenuButton("Themes");
    RuntimeMenu.Add(themes, Style);

    foreach (var name in Themes.Styles)
        AddTheme(name.Theme, themes);

    // Let's create "Font Size" as parent of a group of checkable buttons
    var fontSize = new MenuButton("Font Size");
    RuntimeMenu.Add(fontSize, Style);
    List<int> fontSizes = new() { 8, Constants.Menu.FontSize, 20, 26, 32 };
    foreach (var size in fontSizes)
    {
        // Let's create the "Theme" button
        MenuItem data = new()
        {
            Text = size.ToString(),
            IsCheckable = true,
            CheckGroup = "Font Size",
            Checked = size == Constants.Menu.FontSize,
        };
        data.EventOnClick.AddListener(_ =>
        {
            StyleController.Instance.FontSize(size);
            // Let's know the font size changed
            OnFontSizeChanged?.Invoke(size);
        });
        RuntimeMenu.Add(new MenuButton(data), fontSize);
    }
}
```

At the end the menu may look like this (Figure 4):

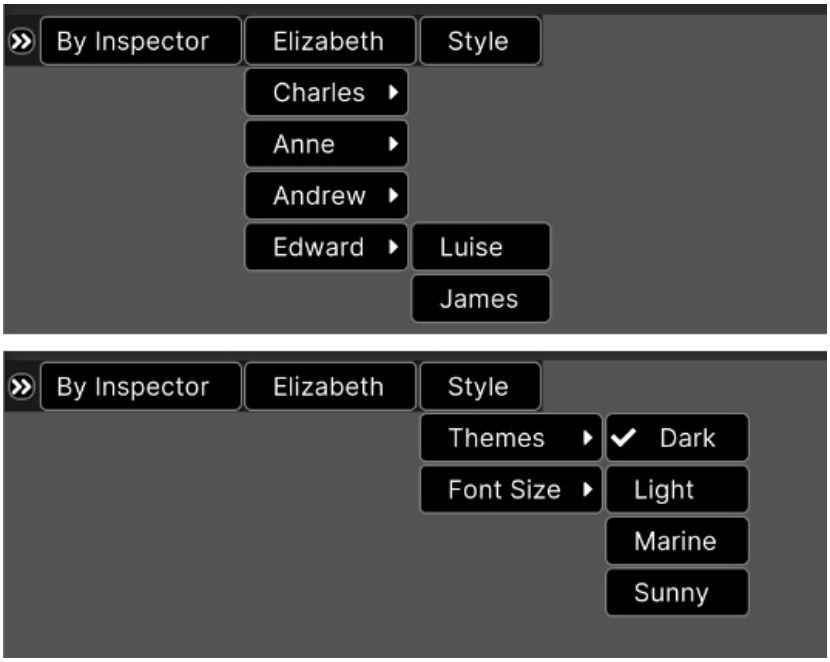

Figure 5

4.3. Themes and Styles

In Figure 4 you can see that the Themes menu is designed in Dark mode; the asset comes with 4 themes implemented: Dark, Light, Marine and Sunny; you can create new ones and add them to the presets list.

Let's now show Figure 4 after setting the other themes.

Figure 6 - Light mode

Figure 7 - Marine mode

Figure 8 - Sunny mode

You can change, runtime, font size, border size, border-radius, and appearance for default, hover, and active states of menu item; to do this you can insert the StyleEdit component into the scene and obtain a preview of the changes by operating in the inspector panel of this component. The next Figure 8 shows an example of a modified style for the sunny theme.

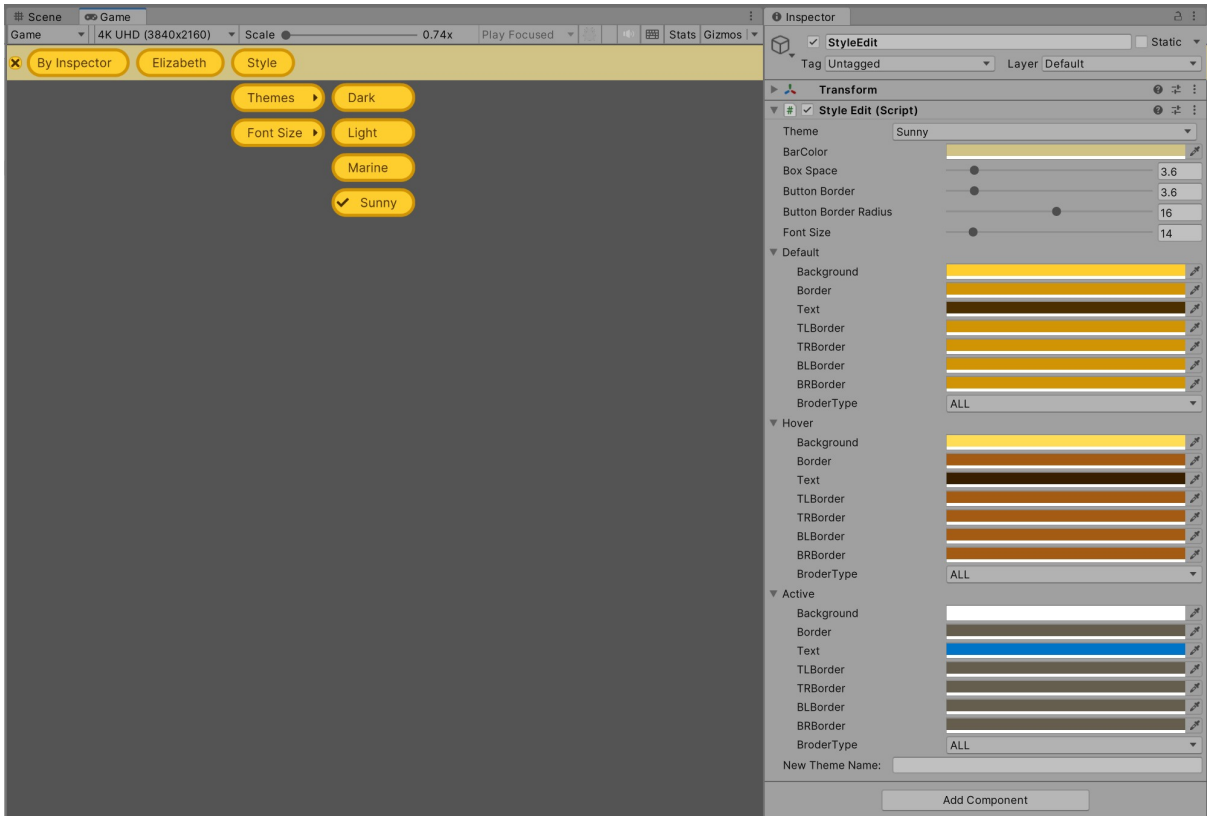

Figure 9 - Runtime style edit

To create a new one, type the name of the new theme in the inspector field of the same name: an add button will appear and, by pressing it, we will store the new theme (Figure 10).

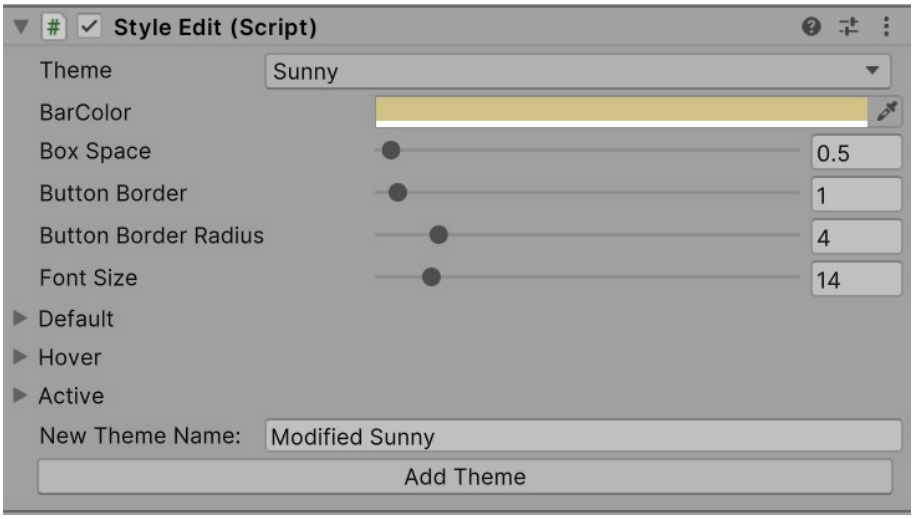

Figure 10 - Adding a new theme

Update and remove operations can be carried out on themes created by the user (Figure 11).

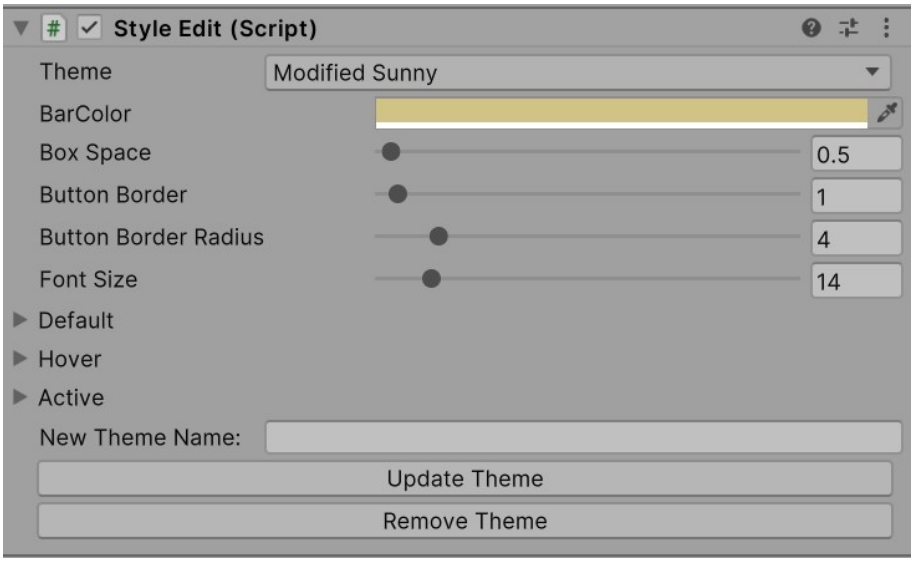

Figure 11 - Updating and removing theme

The themes created by the user remain available for subsequent work sessions: they are stored in a json file in the Application.persistentDataPath folder.

5. Examples

The asset comes with the SampleMenu scene example (UiToolkitMenu/Examples/Scenes/SampleMenu): by loading this scene you will be able to find all the methods of use explained in the chapter “4 Using”.

6. Known Issues

The console gives the following warning:

```
Serialization depth limit 10 exceeded at 'UnityEngine.Events::ArgumentCache.m_ObjectArgument'.
There may be an object composition cycle in one or more of your serialized classes.
```

This is due to the MenuData class which implements the Children variable which is, in turn, a list of MenuData; the aforementioned warning does not affect the correct functioning of the asset.