Flexible Color Picker, free asset for Unity

Ward Dehairs
`http://wardddev.com`

1 Basic use guide

To integrate the color picker into your Unity project, simply add the provided prefab 'FlexibleColorPicker' to the appropriate canvas in your scene. Once added, you can access the output color of the FlexibleColorPicker script via the parameter 'color'. An extremely simple use case would for example be the following:

```
public FlexibleColorPicker fcp;
public Material material;

private void Update(){
    material.color = fcp.color;
}
```

1.1 Customization

All the 'picking images' of the color picker can be turned on or off and resized at will. 1 dimensional pickers, e.g. sliders will automatically orient themselves along the longest axis. All picker images are connected via a serialized array variable to the main script. It is not recommended to delete elements of the color picker, but rather turn the gameobjects off.

1.1.1 Naming

Picker images refer to 1D or 2D sliders that let the user change the selected colors. Color values are denoted by their starting letter: Red, Green, Blue, Hue, Saturation, Value and Alpha.

1.1.2 Markers

Markers pointing to the current value of a picker image are recognized by their name. They should contain the words 'marker' and 'hor' or 'ver' to indicate their function. 1 dimensional pickers have a horizontal and vertical marker included by default. These are automatically turned on or off based the detected direction. The 2 dimensional picker has only one marker. The markers can be freely customized, as long as the point they aim towards remains in the middle of the images and their names still contain the right keywords.

1.1.3 Main Picking Mode

Main picking mode can be set by the user or set by the developer ahead of time. This changes which color values are shown in the 2D picking image. The letters denote Hue, Saturation and Value.

1.1.4 Static Mode

Static mode will prevent the color picker from changing the picking textures when the user changes their selection. This provides a slight performance boost, but is mainly meant as a stylistic option. It is recommended not to use the main 2D picker or the saturation picker in static mode, since these two tend to look the most confusing without dynamically changing colors.

1.2 Event based access

The color picker provides a public Unity event named onColorChange, which can be used to respond to color changes. Note that this function can be called every frame while the user is controlling a slider, so any code added as a listener should be relatively lightweight. Use the standard Unity approach of *AddListener* for most use cases with the FCP.

1.3 Builtin saving

For more advanced projects it is recommended to have your own solution for saving colors picked by the FCP but for small projects and short-term convenience, the FCP package provides a persistence script which can automically save and re-apply the color picked on the FCP. Note that for this to work the persistence script along with its attached FCP must be activated and enabled in the scene. The script works with static variables which will break when recompiling. between session persistence can be achieved either by saving a dedicated file or by writing to playerprefs.

2 Scripting details

2.1 Using multiple color pickers

To achieve dynamic gradient colors with limited CPU overhead, the FCP makes use of custom shaders and materials referencing those shaders. To avoid cross contamination of shader variables between different instances of the FCP, the materials will be copied at runtime. This has some minor overhead, which is why the FCP provides the option to disable support for multiple pickers. This is not recommended generally, but may be useful for low-performance platforms.

2.2 Sprite mesh workings

To achieve dynamic gradient colors using the shader setup there is a little caveat, which is that the FCP must create a mesh for the sprites in the image components that actually render the FCP gradients. In general, the picker requires 2 vertices for most gradients, with only a hue gradient requiring 7 vertices in stead. This is because Unity breaks down the gradients into simple linear interpolations. A hue gradient can be accurately broken down into 6 gradients, requiring 7 vertex points. The 2D picker also suffers somewhat from lesser subdivisions, so it is recommended to provide it with at least 4 if it has no hue axis.
The Flexible Color Picker comes with a special editor script, called the FCP_ SpriteMeshEditor. You can add this script to a Gameobject in the editor, set the x and y values and add a sprite from your asset folder to it. This will change the mesh of the sprite so it has x and y vertex subdivisions in the horizontal and vertical direction. The goal is to keep the vertex count as small as possible, while providing enough detail to look good.
The 2D picker and hue picker come with a 7×7 sprite by default for the sake of being generally applicable.

2.3 BufferedColor

The Color Picker works internally with a BufferedColor object, this color can retain hue and saturation values even when these are singular for its color. This makes sure that these values do not automatically reset if the user select a singular color: e.g. pure gray has no valid hue value, which defaults to 0 indicating red. If the user desaturates blue it would suddenly jump to being red once completely desaturated.

2.4 private serialized parameters

- **pickers**; Contains data and connections for each picker images, which should be the 10 pickers in order of the *PickerType* enum.
- **hexInput**; Connects to the hexadecimal color input field.
- **modeDropdown**; Connects to the main picking mode selection dropdown.
- **mode**; Current main picking mode for the 2D picking image.
- **staticSpriteMain**; List of 6 static sprites for the main picker corresponding to the 6 main picking modes in order of the *MainPickingMode* enum.
- **startingColor**; Color value set upon Start()

2.5 Public access

- **color**; Main color getter/setter access
- **staticMode**; Should textures not change dynamically.
- **multiInstance**; Activate/deactivate support for displaying multiple FCP components consistently
- **onColorChange**; Unity event triggered when colors change

- **advancedSettings**; Change detail settings related to individual picker images

2.6 Public static functions

The Color Picker provides a few free functions that may be useful in further scripting: functions for sanitizing hex strings and for converting between rgb and hsv color formats.

2.7 Internal workings

These have been documented in the script itself where necessary, look for full-line comments for the structure.

2.8 Performance / optimization

The Flexible Color Picker was developed with clean and readable code in mind, in preference over performance. Due to complaints about performance, picking images have been reimplemented using a custom shader model in stead of generating them in-script. This should keep the FCP lightweight to the point where you should not be worried about its performance regardless of the application. If Performance does become a problem you can fall back to static mode. This is also recommended if compatibility is a problem, you can in such a case use platform-dependant code to activate static mode on problematic platforms.

3 FAQ

The gradients of the picker images won't render on a specific platform (often IOS)
As usual you want to check for any error messages being thrown up. If you cannot find relevant errors the most likely explanation is that the FCP gradient shader was not loaded on the target platform. You can try adding your shader to Unity's *always include* list. If that doesn't help, double check that the shader is working by adding an FCP material to a 3D object in your scene. In some cases doing this has outright fixed the problem.

I don't want the colors of the hue gradient to dissapear when choosing an unsaturated color (white, black or gray)
Go into advanced settings, open the dropdown for Hue (H) and check *override static.* this will make sure the Hue gradient is replaced by a premade, fully saturated static image which will not fade according to the selected color.