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 public array variable to the main script, this array should never be modified. 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 Using multiple color pickers

Currently, the flexible color picker works with custom shaders to achieve the gradients of the picking images. This has the annoying downside that the Unity UI does not have property color blocks, meaning that changes to one UI material will always change the look of all UI pieces using that material. In other words, if you use 2 color pickers, the picker images will cross update.
To get around this, you will simply have to copy the materials you want to use and assign the new batch to the secondary picker.

1.3 Performance

The main performance cost of the Flexible Color Picker lies in rendering the picking images. This performance cost is still relatively low, so for most practical applications you may skip this section.
The performance of the picker images can be improved by changing the mesh of the sprites in the picking images. 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 Scripting details

2.1 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.

2.2 Public access

- **PickerImages**; Should always contain the 10 types of picking images in order!
- **HexInput**; Connects to the hexadecimal color input field.
- **Dropdown**; Connects to the main picking mode selection dropdown.
- **mode**; Current main picking mode for the 2D picking image.
- **startingColor**; Color value set upon Start()
- **staticMode**; Should textures not change dynamically.
- **SetPointerFocus**; Set a specific picker image as the focused image for subsequent updates.
- **PointerUpdate**; Set new color value via a picking image.
- **TypeHex**; Set new color value via hex input.
- **FinishTypeHex**; idem, for finishing input.
- **ChangeMode**; Change main 2D picking mode (via button)

2.3 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.

2.4 Internal workings

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

2.5 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.