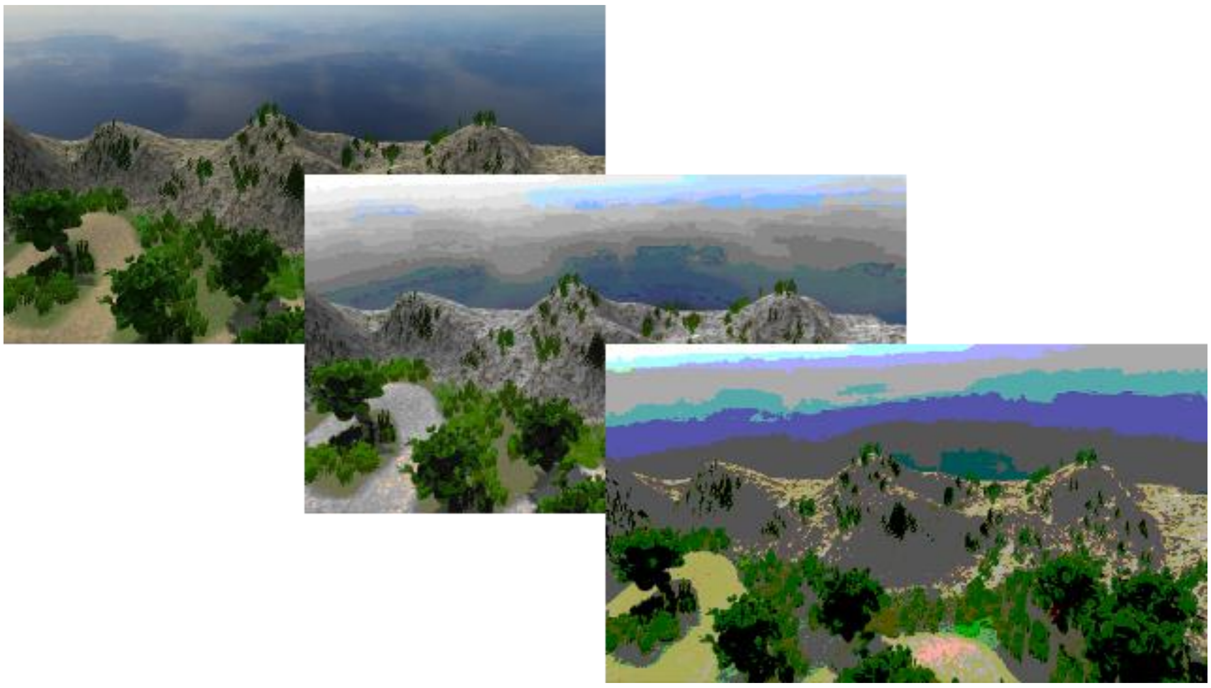
Palette Shader (Version 1.0.1)

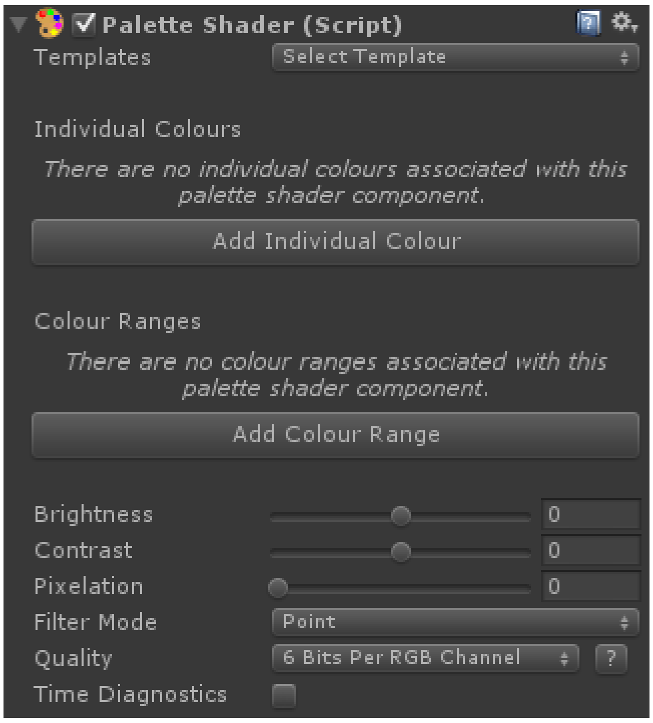
Palette Shader is a full-screen camera shader for restricting the palette of available colours that a camera can render. Any number of colours, or ranges of colours, can be specified via a Palette Shader component (which has its own Unity Editor GUI) and the shader will display each pixel in whichever palette colour is nearest to the pixel's original colour. This can be used, for example, to restrict the number of on-screen colours to produce a more retro feel:



No coding is required – it's simply a case of adding the Palette Shader component to a game object that has a camera and setting up the palette details in the Unity Editor. Palette Shader gives full control over image brightness, contrast and pixelation and has adjustable quality settings to tweak the trade-off between quality and speed/memory. It has full prefab support and includes 21 template colour palettes to get you started. Here's how it works:

Adding a Palette Shader Component

To use Palette Shader, add a Palette Shader component to a game object. Palette Shader requires a Camera component – one will be added to the game object automatically if it doesn't already have one. The basic Palette Shader component looks like this:


Palette Shader (Script)

Templates: Select Template

Individual Colours

There are no individual colours associated with this palette shader component.

Add Individual Colour

Colour Ranges

There are no colour ranges associated with this palette shader component.

Add Colour Range

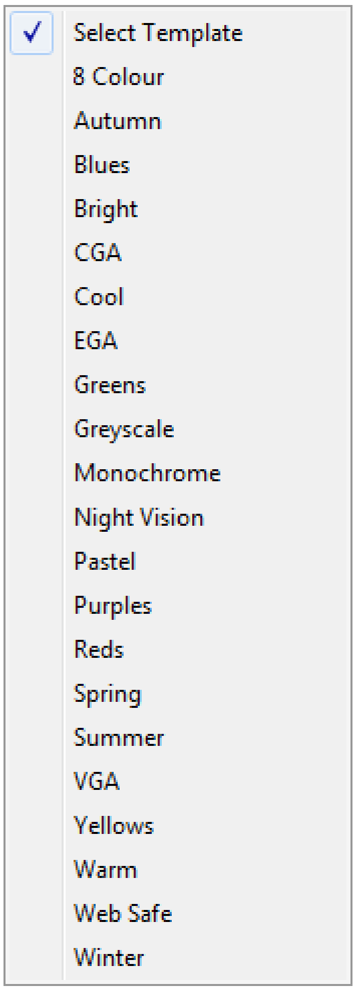
Brightness	0
Contrast	0
Pixelation	0
Filter Mode	Point
Quality	6 Bits Per RGB Channel
Time Diagnostics	☐

Each of the properties is described below.

<u>Templates</u>

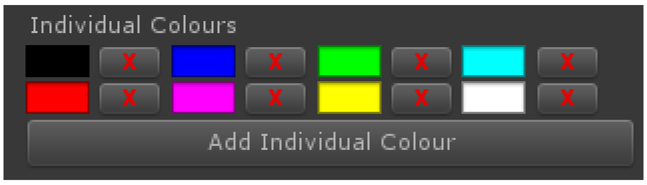
Here you can choose from one of 21 pre-set templates. Selecting a template will change all the Palette Shader settings to match that template, after which they can be tweaked further to your liking. Templates range from colour palettes used in old graphics adapters to give your game a more retro feel, to warm/cool/seasonal palettes, to greyscale/monochrome. You do not have to select a template – Palette Shader components can just as easily be built from scratch.

The list of templates is:



Individual Colours

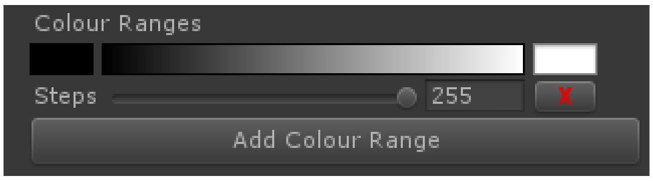
This property specifies the individual colours that are included in the colour palette. When the scene is run, each pixel that the camera displays is shown in whatever its nearest palette colour is. The more colours you include in the palette, the more colours your final image will have.



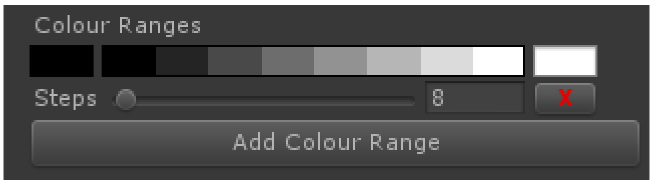
Colours can be added by clicking the **Add Individual Colour** button, changed by clicking on the colour squares themselves or deleted by clicking their respective **X** buttons. Note that you must set at least one individual colour or one colour range (see below) to create a valid palette.

Colour Ranges

Colour ranges are a quick way of creating a smooth transition from one colour to another. For example the greyscale palette template contains a single colour range from black to white:



This results in black, white and all levels of grey in between being included in the palette, and saves you having to define 256 individual colours. The **Steps** property can be adjusted to make colour ranges display colours in steps, rather than as a smooth transition, for example:

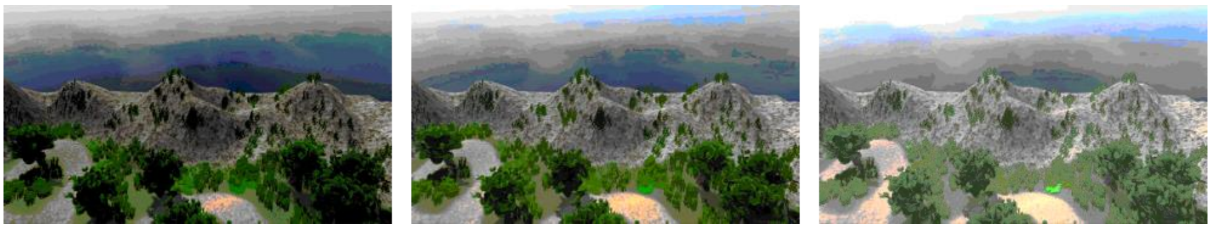


As with individual colours you can have as many colour ranges as you like. The **Add Colour Range** button can be clicked to add another one, and the red **X** buttons can be clicked to delete them.

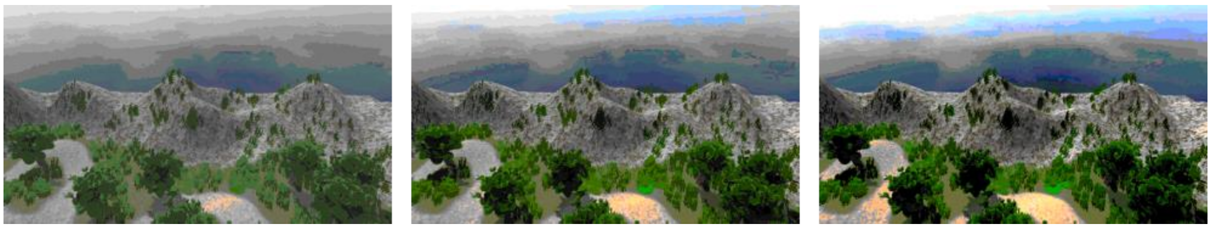
Brightness and Contrast

The brightness and contrast levels work the same way as they would on a television or in a graphics program. They change the original image, not the palette, but doing so will change the palette colours that the individual pixels in the original image point to. These values can be changed at run-time and will have an instant effect.

Here is an example of brightness increasing, from the left to the right image:

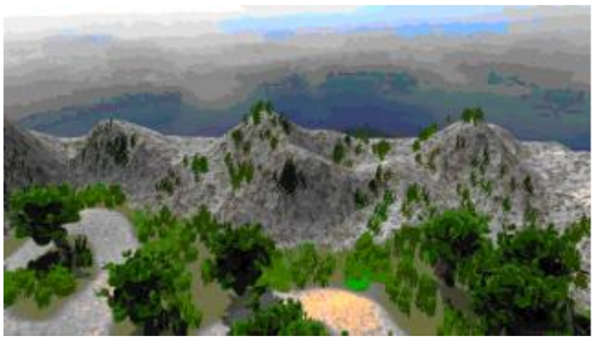


And contrast increasing:



Pixelation

This property controls how much the screen is pixelated, with 0 meaning no pixelation and 1 meaning the entire screen is merged into a single pixel. The image on the right, below, is a pixelated version (at a level of 0.4) of the one on the left:



Filter Mode

Palette Shader creates a reference texture to map individual pixel colours to palette colours (see Design-time and Run-time below), and this property specifies the way the shader samples that texture. Basically, **Point** displays the palette colour with the exact closest match in the palette, whereas **Bilinear** and **Trilinear** apply some merging at colour boundaries (and may result in colours that are part-way between two palette colours, meaning the output image is not necessarily restricted to the just the specified palette colours).

Quality

This property specifies the size of the 3D reference texture to generate. Higher bits per pixel will result in a larger texture that will take up more memory and take more time to generate, but will be more accurate. Usually 6 bits per RGB channel is more than enough – you should go for the lowest possibly quality setting that doesn't compromise the output image quality. You can click the **?** button to the right of this property in the inspector for more information on how the quality setting works.

Time Diagnostics

This is a Unity Editor-only feature that displays in the console window the time taken to generate the reference texture when the scene runs. Higher quality settings can cause Palette Shader to take a long time generating the reference texture – if this is the case you may wish to consider reducing the quality level, often at negligible cost to the final image quality. The default setting of 6 bits per RGB channel rarely takes more than a few milliseconds to generate, even for complex palettes. Note that the time diagnostics flag has no effect in a game build – the timing code is only compiled when running a scene in the Unity Editor.

Design-time and Run-time

Palette Shader works by generating a 3D reference texture when the scene is run. The red, green and blue components of any given colour represent the three dimensions of the texture, with each pixel pointing to its closest palette colour. For this reason, palettes themselves must be set up at design-time and cannot then be modified at run-time. This means the **Individual Colours**, **Colour Ranges** and **Quality** parameters must be initialised at design-time and will have no effect if they are changed at run-time.

However, **Brightness**, **Contrast**, **Pixelation** and **Filter Mode** are just shader properties, so these can be changed at run-time, programmatically if required. The Palette Shader component will handle passing them to the shader.