Channel Mixer Documentation – 1.0v

Mixing 4 textures, into one.

Why is Channel Mixing important?

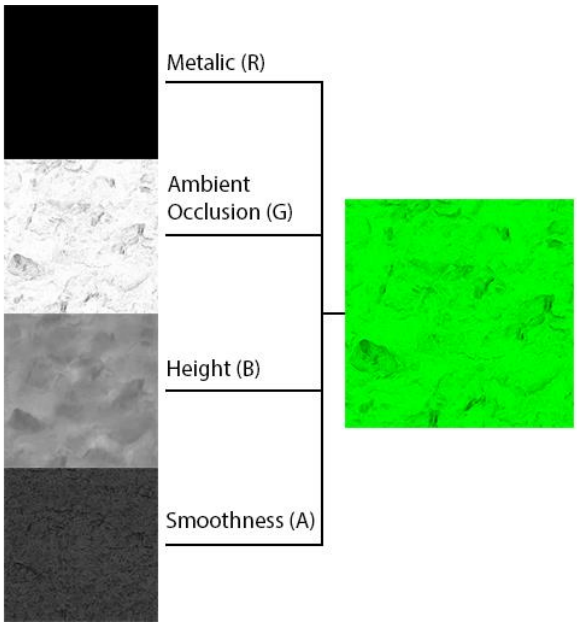
Channel Mixing has been with us a long time. If you take a look at an image, you can see it usually contains 3 to 4 channels, in other words, an image is made out of combination of 3 main colors, in which we can represent almost all of the colors. These colors or channels are Red, Green and Blue with an addition of the Alpha channel, which usually handles transparency of an image.

In order to represent the final image, each color channel has to have 8-bit width in the memory, giving it a range from 0-255, which in a RGB image, takes up 24-bits. If we also include the alpha channel that is a 32-bit image. For standard imagery you would not need to care much about this information, however, in 3D we use images to represent how a model looks in the 3d world.

For representing the values of the textures, we use Shaders, which is a type of a program that is used for shading to represent the appropriate levels of light, darkness, colour, etc. To fully represent an object we use Materials, which uses said shaders to fully define how a surface should be rendered, including information such as tiling, colour, and more, depending on the shader itself. [1]

To sum up, we use the textures in the material (which uses the shader) to achieve a look for an object. However, the more textures we use, the more memory we are using from both VRAM and RAM, which is crucial to avoid as a part of an optimization, to make the game run smoother.

In order to achieve that, we use Channel Mixing or Packing, which combines 4 textures that are usually in Grayscale, into one, to use in shaders that support the technique of using individual channels (Red, Green, Blue and Alpha) to represent a property with each channel, for example:

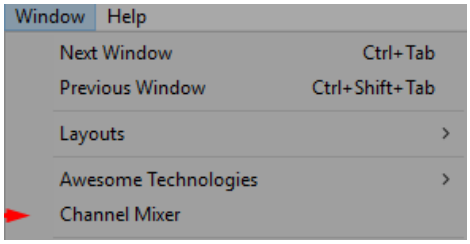


HDRP Mask Map Example:
Red – Metallic property
Green – Ambient Occlusion
Blue – Height Information
Alpha – Smoothness/Roughness information

In order to save on performance by not using 4 different textures.

With Channel Mixer you can do that, with any sort of combination you like, inside your Unity Project.

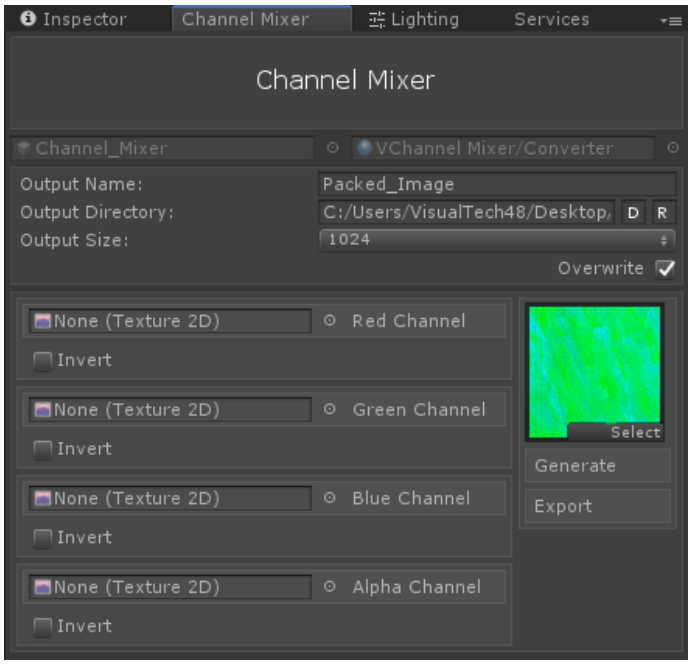
[1] https://docs.unity3d.com/560/Documentation/Manual/Shaders.html



Get started:

In order to start packing, firstly navigate to your unity's Windows tab, and find Channel Mixer:

This will open up the Channel Mixer next to where the Inspector is.



Start Packin'

Welcome to Channel Mixer. To start packing, click the Initialize button, will add 2 objects to your scene:

- Channel Mixer
  - Plane

These objects are crucial for Channel Packer as Channel packer unlike other solutions uses a half GPU half CPU approach, to maximize effectiveness, as GPU renders image a lot faster than a CPU can on which you can read here.

CPU is only used to actually copy the image to your PC.

As you can see Channel Mixer offers a lot to the control of output you want.

File Controls:

CONTROL	USAGE
OUTPUT NAME	Name of the texture that will be outputted/exported
OUTPUT DIRECTORY	Where will the texture be exported
[D]	Opens up a windows explorer button which allows you to choose where the texture will be exported
[R]	Reset to standard output directory, which is the projects *Assets/Channel Mixer/Exported*
OUTPUT SIZE	Size of the output texture. Available Sizes: [128, 256, 512, 1024, 2048, 4096] *Bigger image size will take longer to export
OVERWRITE	Should the pack overwrite the image, or add a new one with a prefix, starting from _01, onward.

Image Controls:

CONTROL	USAGE
RED/BLUE/GREEN/ALPHA CHANNEL	Assign a texture which will be represented in the Red/Green/Blue/Alpha Channel
INVERT	Inverts the assigned Texture
TEXTURE PREVIEW	On the right, you have a texture preview
GENERATE	Generates what will the texture look like in the texture preview
EXPORT	Exports the texture.

Support

For any questions or issues, please contact me through either the forums, or through email.
visualtech48@gmail.com

Thank you for purchasing the plugin!

Changelog:

1.0v – Initial release:

- Initial release of the plugin.