FIMPOSSIBLE CREATIONS

F Texture Tools

USER MANUAL

About **F Texture Tools**

- F Texture Tools is group of tools for use only within Unity Editor (not on the build)

- Plugin offers different types of algorithms for editing texture files.

- Plugin is changing texture's source files, so if you're not sure of the result of the texture editing algorithm, make sure to create a backup of your source texture!

- Plugin is supporting saving image files with extensions like: .**PNG** .**JPG** .**TGA** .**EXR** But you can convert **any texture file** which Unity sees **into .PNG** format using this plugin's **quick converter function**.

- F Texture Tools is providing a different **customized inspector window** (GUI) to help use it without confusion since there are a **lot of parameters to play with**.

- You can use tools on any texture files in your project, **just hit the right mouse button on some texture file** in your project browser **-> Fimpossible Creations -> Texture Tools** and choose the option you're interested in.

- Package is providing source code, including base class for texture editing window, which makes texture editing algorithms easy to implement.

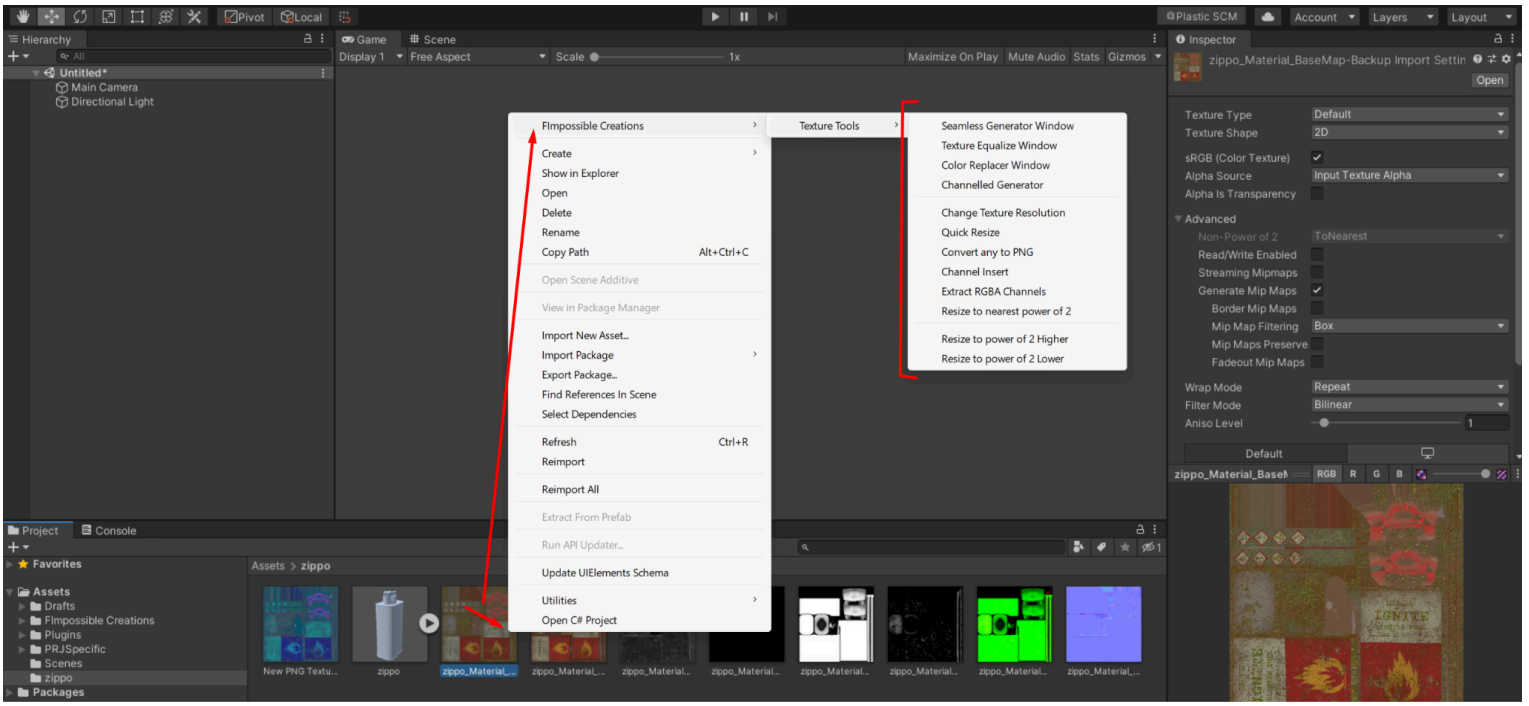
- Package also includes a quick scale (image resolution) tools.

Contact and other links you will find in Readme.txt file

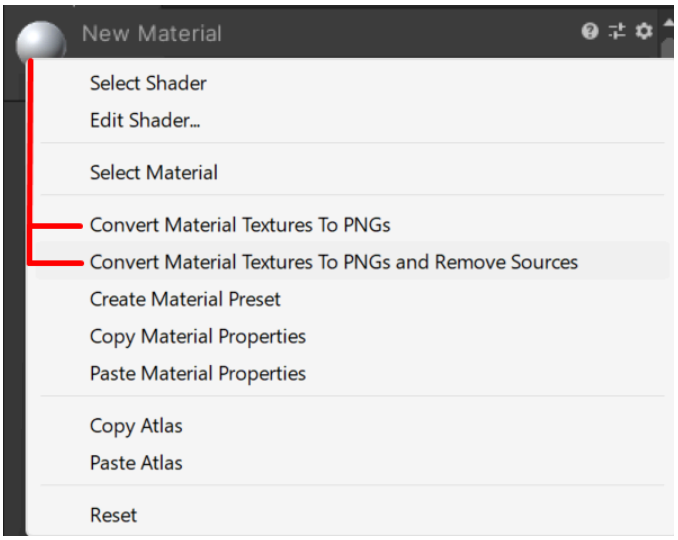
Index

1: How To Start Using Tools

After importing the plugin and compiling it, all you need to do to start using the tools is clicking with the **right mouse button** on any **texture file in your project browser**, then find "**Fimpossible Creations**" bookmark
(it can be placed on top or on the bottom, depending how unity will compile it)
it will reveal "**Texture Tools**" and here you will find most of the features of the package.



There is also extension menu for materials:



2: Seamless Texture Generator

If you're struggling with visible seams in your textures, this tool will be very handy. Seamless texture generator algorithm, is using edge stamp technique, in order to erase visible texture seams, and in result make it tile.
This technique will not tile brick like textures.

But don't overestimate this tool. It will make textures tile, but it will not solve repetitiveness. If your material is tiling a texture like 10x10 and the source image has big contrast, you will notice texture tiling easily.

Repetitiveness can be solved in some cases (not in all cases tho), you will read more about that in Texture Equalizer tool section.

To start using Seamless Texture Generator, simply click with right mouse button on any texture file in your project browser, find menu option:
Fimpossible Creations -> **Texture Tools** and select -> **Seamless Texture Generator**

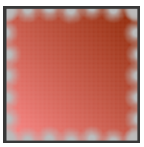
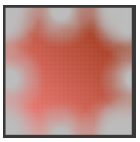
New window will pop up, and there you can prepare settings for stamping.

Seamless Texture Generator Parameters:

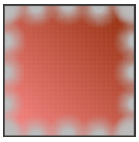
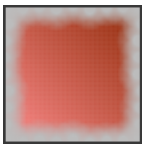
(To visualize parameters, there are used two different textures for main view (red) and for stamping (gray))

Seed: This value is changing random seed used for probing stamps from the source texture. If you change this value, stamps will be distributed in different ways.

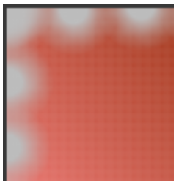
Stamp Radius: Changing size of the stamp and spacing between stamps.
In some cases, setting radius very high, can help reduce texture repetitiveness.

Radius = 0.25:  Radius = 1 

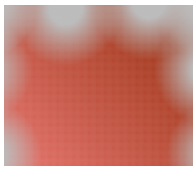
Stamp Density: Changing spacing between stamps.
Don't change it too low, it may prevent the algorithm from stamping seams!

Density = 0:  Density = 0.7: 

Stamp Hardness: Basically stamp's alpha falloff.

Hardness = 0:  Hardness = 1: 

Stamp Noise: Applying perlin noise to the stamp.
Warning! Increasing it above 1.5 value can reveal seams under stamps if using medium density, so watch out. Increasing this value above 1 will ignore the Stamp Hardness value!

Noise = 0:  Noise = 2: 

Randomize: Offsetting stamps in random directions with provided amount.

Rotate: Applying rotation to the stamps.
With value = 0, there is no rotation applied to the stamps.
With value = 1, there are applied random 90 degrees rotations per stamp.
With values greater than 1, in addition to 90 degrees rotation there are applied more randomized rotations. Warning! When value is greater than 1, the rotated stamps pixels are not interpolated (no antialiasing) so it's recommended to use Rotation = 1.

Dimensions to Loop: You can choose to stamp just the left/right side of the source texture (useful for wall textures), upper/bottom side (useful for pipes textures), or all sides.

Use Other Texture: With this option you can enable stamping with use of other texture files. It can be useful when you want to tile texture which has dark-crack lines in it and it appears in stamps, generating bad looking results. Then you can select different, more cleaner textures to stamp sides of the source texture.

In order to be able to stamp using another texture, you need to change its "read/write" toggle to true and you will see the button as a shortcut to do this. After stamping, it's recommended to switch "read/write" off again, to avoid bigger texture size in the memory.

Exporting result:

If all parameters are set, you can apply it to the target texture or selected textures.
To do it, you will use button on the bottom of the texture editor window:

Apply to file (No Undo): By hitting this button, algorithm will apply seamless texture generator algorithm, to the selected file and overwrite it. In result you will see effect immediately on the scene, and you will be able to do further editing of the texture in any image editing software.

Preview: By hitting this button, the algorithm will be applied to the Unity Editor texture instance, that means the source file will be untouched.
You will see preview on the scene but without mip-maps (texture can look sharper than default) You can undo these operations simply by clicking the right mouse button on the texture file and choosing 'Reimport'.
(Watch out! Don't hit Reimport All! It will trigger reloading the whole project!)

Result As New File: By hitting this button, the algorithm will generate a new image file and apply the result on it.

Generate Backup And Apply: By hitting this button, the algorithm will generate a copy of source texture then apply the result on the source file.
(good for instant previewing result with backup)

All these buttons above you will encounter in the Texture Equalize window and in Color Replacer Window. Same rules apply to them.

3: Texture Equalizer

If your texture has very bright / very dark areas, which are conflicting with PBR workflow, since the dark areas could easily be Ambient Occlusion maps and too bright areas could serve as the smoothness maps, you can use texture equalizer in order to reduce contrast, making image look more like Albedo map.

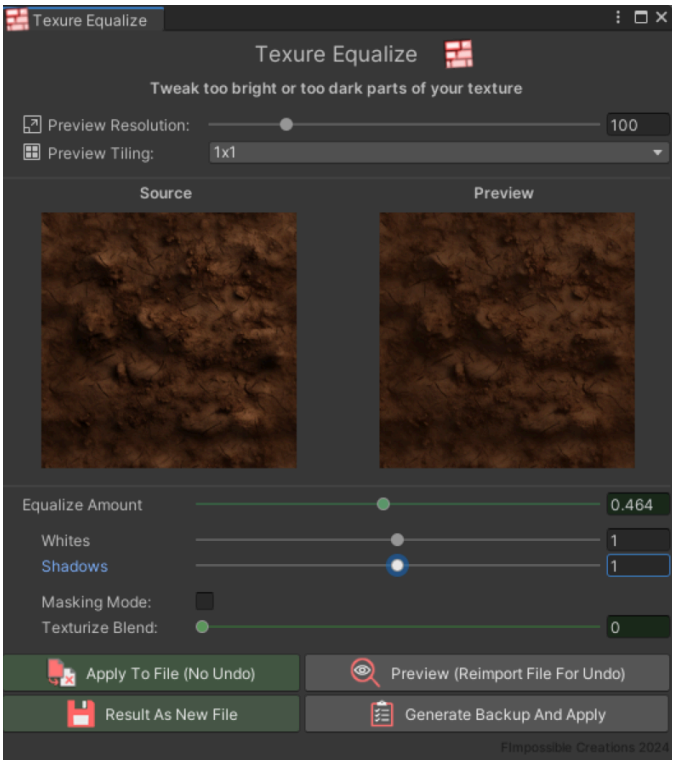
This tool can be used to reduce contrast on the image in the smart way, reducing the effect of the repetitiveness for tiled textures.

Since a simple brightness increase of dark shadow areas is causing loss of detail on the texture, texture equalizer offers a feature to apply another texture to the masked areas - like filling very dark shadow areas of brick textures with concrete texture instead.

You can use this feature for a few more purposes, not just delighting.

To start using Texture Equalizer, simply click with right mouse button on any texture file in your project browser, find menu option:
Fimpossible Creations -> **Texture Tools** and select -> **Texture Equalize Window**

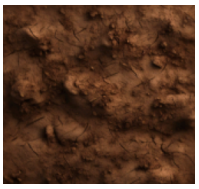
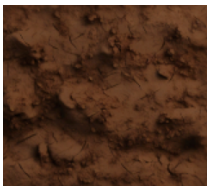
New window will pop up, and there you can prepare settings.



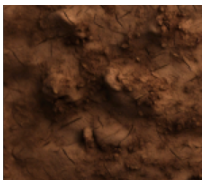
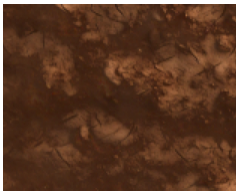
Texture Equalizer Parameters:

Equalize Amount: Basically 0-100% blend of the equalization effect.

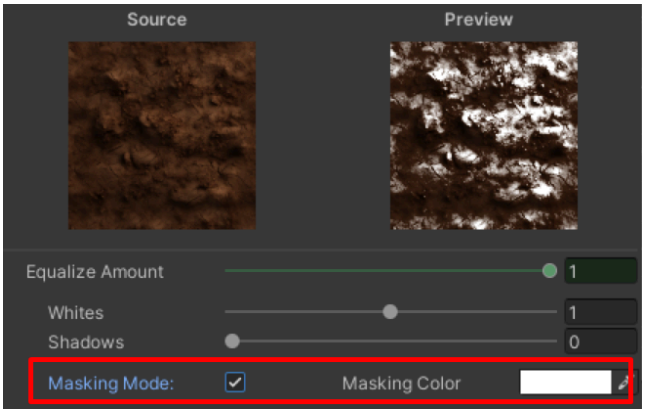
Whites: Sensitivity for detecting bright parts of the texture.

Whites = 0  Whites = 0.7 

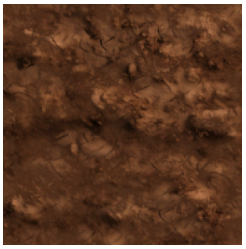
Shadows: Sensitivity for detecting dark parts of the texture.

Shadows = 0  Shadows = 2 

Masking Mode: Enabling feature, to apply custom color on the affected areas of the texture. It can be useful for making masks for shaders.



Texturize Blend: Increase this value to see more settings.
This value is an enabling feature to fill affected areas of the texture with another.

Texturize Blend = 0:  Texturize Blend = 1 

4: Color Replacer

Color replacer will help you replace / mask selective color on your texture.
It can be used not just for replacing color with new one, but also for masking to use with custom shaders.

To start using Color Replacer, simply click with right mouse button on any texture file in your project browser, find menu option:
Fimpossible Creations -> **Texture Tools** and select -> **Texture Equalize Window**

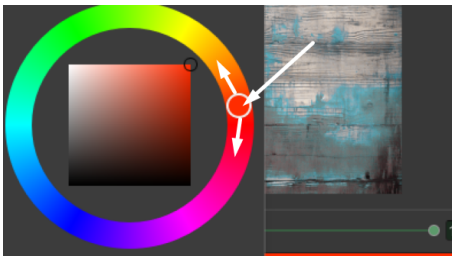
New window will pop up, and there you can prepare settings.

Color Replacer Parameters:

Effect Blend: Basically 0-100% blend of the replacement effect.

Color To Replace: Most important parameter. Reference color to be used by algorithm for hue matching.

It's useful to fine-tweak this value in the Unity's color circle and check the preview for best color match result:



Hue Matching: Another very important parameter. It's defining sensitivity of color similarity detection. Increase it to affect just very similar colors, lower to a smooth mask a bit.

Hue Matching = 0  Hue Matching = 1 

The '**Sat**' and '**Value**' parameters are responsible for extra saturation and value matching, but their effect is limited and it can help just in exceptional cases, or when using saturation/value boost under color settings below.

New Color Settings: There you will find **Hue** offset parameter to shift selected color into desired color, **Sat** to modify saturation of masked pixels and **value** to modify brightness.

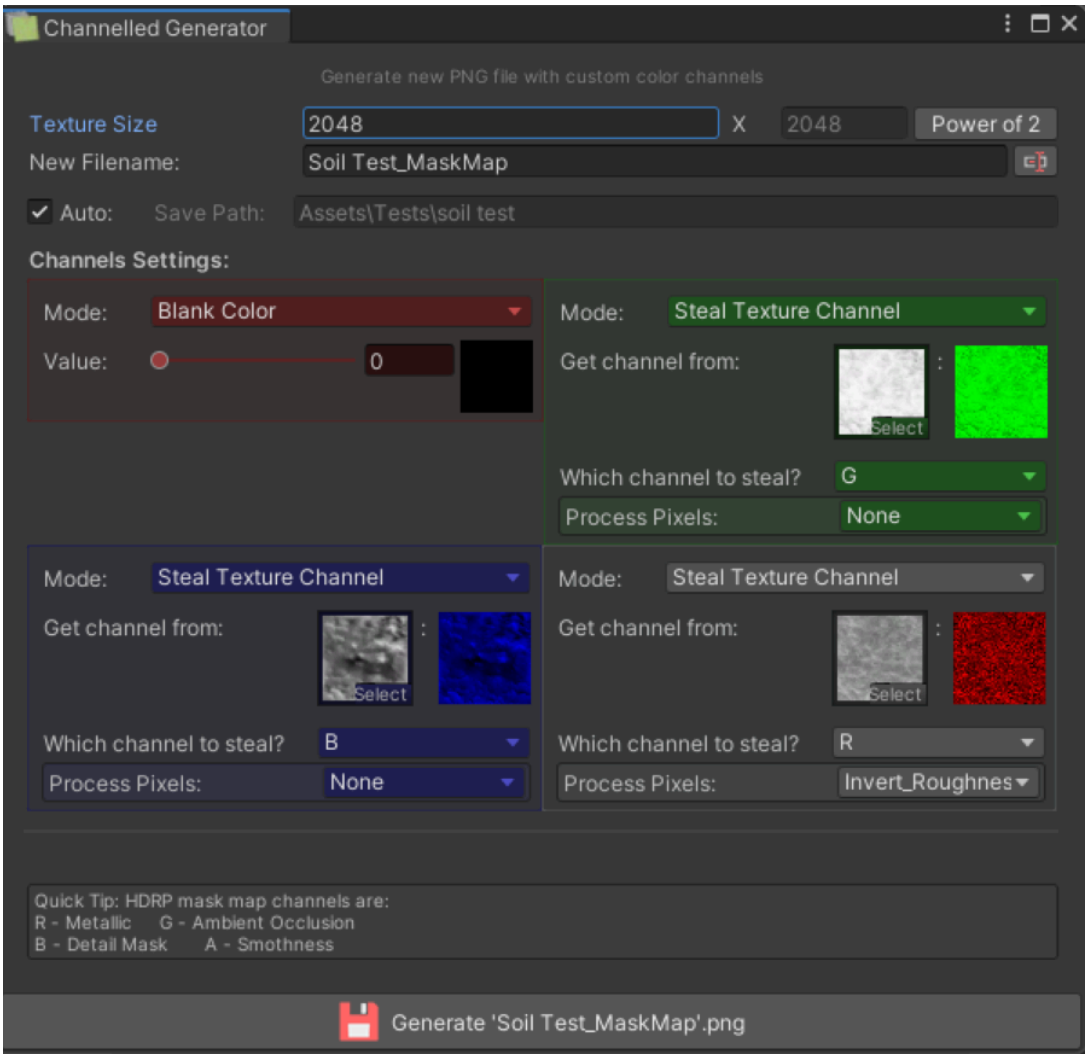
To execute the result, check page 7 (Exporting Result) of this manual for export buttons description.

5: Channelled Generator

Channelled Generator will help you generate texture files with selective masks for each single RGBA channel. It also offers pixel processors for example to change roughness map into smoothness map, brighten it etc.

To start using Channelled Generator, simply click with right mouse button on any texture file in your project browser, find menu option:
Fimpossible Creations -> **Texture Tools** and select -> **Channelled Generator**

New window will pop up, and there you can prepare settings.

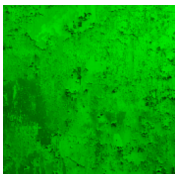
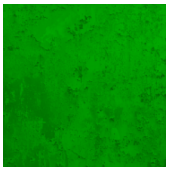


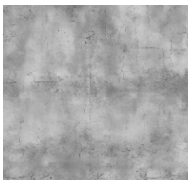
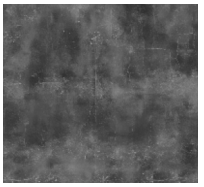
(if texture is grayscale, you can simply steam R or any other channel, I selected G and B for more intuitive insight)

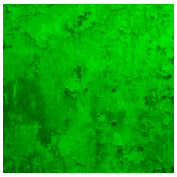
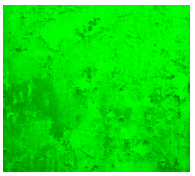
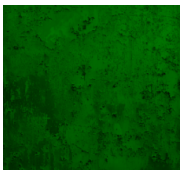
This window will generate a new image file, so you need to define its resolution, its filename and save path.
The save path is gathered automatically out of the currently selected file in the project browser.

Below you can choose how each color channel of the new image file should be filled. You can choose **Blank Color** which will make the channel be filled with a single value in each pixel.
Steal Texture Channel will copy selected color channel of the provided texture.
None can be used on the alpha channel in order to not include it in the file.

Pixel Processors allows you to modify the channel before pasting it into a new file.

Reset Contrast = 0  Reset Contrast = 0.6 

Invert - Roughness to Smoothness Off  On 

Add = 0  Add = 0.3  **Multiply** = 0.5 

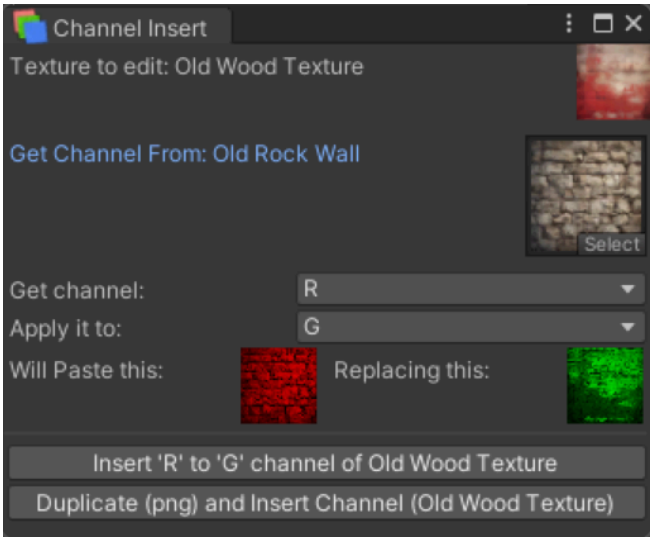
Hit the “ Generate New File” button to create a new image file in the selected directory and name.

6: RGBA Channels Tools

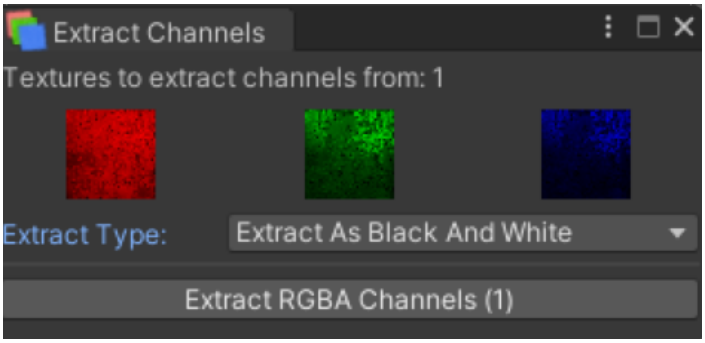
Package currently is offering channel tools like:

- Channelled Generator (Page 12)
- Channel Inserter
- Channels Extractor
- PNG Converter (channel swizzle possibility)

Channel Inserter allows you to quickly insert a channel of one texture into choosed channel of another texture. You can open the channel inserter window by selecting it in the image file menu (Page 3).



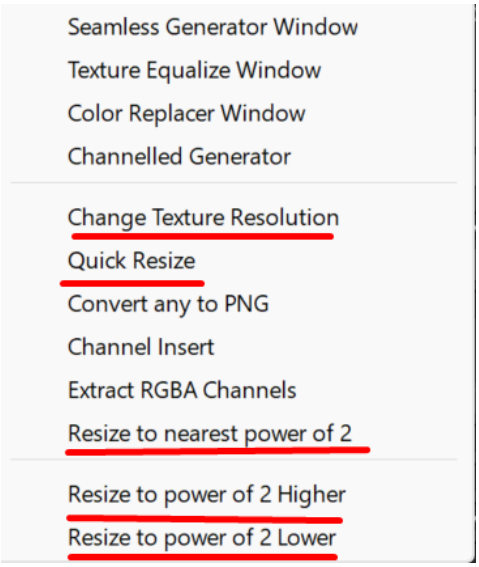
Channels Extractor allows you to break an image file into each channel as a separate grayscale or color image file. You can open the channel inserter window by selecting it in the image file menu (Page 3) it's called **RGBA Channels Extractor**.



7: Resize Tools 

Resize tools are basic utility tools, which can be used for quickly resizing textures to power of 2 (256 - 512 - 1024 etc.) which are interpreted by computer graphics faster. You can resize images to any size you want, but the target is power of 2.

You can also use it to scale down textures if some of the assets you use have much bigger textures than you need. Can come very handy for managing texture files in the git repositories.



You will find 5 scaling options in the menu. **Change texture resolution** will open new window for resizing images.

Quick Resize is similar but simplified.

Resize to nearest power of 2 will immediately trigger resizing the texture to fit its resolution to power of 2. Nothing will happen if it's already power of 2 resolution.

Resize to power of 2 Higher/lower will immediately trigger resizing texture.

8: Creating Custom Algorithm

Since F Texture Tools comes with source code, you can use the base class which is used by Seamless Texture Generator, Texture Equalizer and Color Replacer windows, to create your own tool!

Here example how you can do it:
(keep in mind you need to know how to code unity editor GUI and know how to operate on the pixels)

We will do a basic grayscale conversion window.

First, create script file in the directory which has access to F Texture tools, for example do it under *Assets\FImpossible Creations\Editor\Editor Tools\F Texture Tools*
Let's call it GrayscaleConvertWindow.cs

You will require usings as:

```
using FIMSpace.FTextureTools;
using UnityEditor;
using UnityEngine;
```

Inherit from **FTextureProcessWindow**, so:

```
public class GrayscaleConvertWindow : FTextureProcessWindow
```

Now you need to define method which will bring window to display:

```
[MenuItem("Examples/Grayscale Convert Window")] // Now you can turn on window through the unity top toolbar
public static void Init()
{
    GrayscaleConvertWindow window = (GrayscaleConvertWindow)GetWindow(typeof(GrayscaleConvertWindow));
    window.titleContent = new GUIContent("Custom Texture Edit");
    window.previewScale = FEPreview.m_1x1;
    window.drawPreviewScale = true;
    window.position = new Rect(200, 100, 600, 800);
    window.Show();
}
```

Let's define variable to control effect blend and let's draw it in the window GUI:

```
public float GrayscaleBlend = 0.5f;

protected override void OnGUICustom()
{
    GUILayout.Space(4); // Basic GUI Spacing

    GrayscaleBlend = EditorGUILayout.Slider(new GUIContent("Grayscale Blend"), GrayscaleBlend, 0.0f, 1f);

    GUILayout.Space(8);
}
```

Now let's write a basic algorithm to make pixels grayscale.
We will compute the average value of R G B and apply it to all channels.
To do it, you need to override

```
protected override void ProcessTexture(Texture2D source, Texture2D target, bool preview = true)
{
    Color32[] sourcePixels = source.GetPixels32(); // Get edited texture pixels
    Color32[] newPixels = source.GetPixels32(); // Prepare new pixels to be applied to the texture

    Vector2 dim = new Vector2(source.width, source.height);

    for (int x = 0; x < source.width; x++) // Iterate through all pixels
    {
        for (int y = 0; y < source.height; y++)
        {
            // Get current pixel index
            int px = GetPX(x, y, dim);

            Color sourcePx = sourcePixels[px];

            // Calculate grayscale
            Color grayscalePX = sourcePx;

            float average = (grayscalePX.r + grayscalePX.g + grayscalePX.b) / 3f;
            grayscalePX.r = average;
            grayscalePX.g = average;
            grayscalePX.b = average;

            // Apply with blend
            newPixels[px] = Color32.LerpUnclamped(sourcePixels[px], grayscalePX, GrayscaleBlend);
        }
    }

    target.SetPixels32(newPixels);
    target.Apply(false, false); // Finalize texture
}
```

And that's all, now you can open you texture window and do first image conversion!

If you like this package please visit my [asset store page] for more or write a review for this asset ;)