INTRO

ProTex allows you to create and use procedural textures in Unity. Create your own textures using the visual node editor and apply them to your materials during runtime. You can also save them as normal textures in the Unity Editor to be used however you want.

Unlike normal textures, procedural textures take very little disk space to store (typically less than 20Kb), but they can be used to generate textures of up to 2048x2048 during runtime. All generated textures are guaranteed to be seamless, meaning that they can be placed side-by-side without creating a noticeable boundary.

Compose your textures using the ProTex Editor and the different types of nodes at your disposal:

- Noise generation: *FBM, Turbulence, Ridged, Cellular and Random.*
- Filters: *Blur, Sharpen, High Pass, Low Pass, Grayscale, Invert.*
- Math operations: *Blend, Interpolate, Add, Subtract, Multiply, etc.*
- Transformations: *Rotate, Repeat, Offset, etc.*
- Shape generation: *Circle, Triangle, Rectangle, Pentagon, etc.*

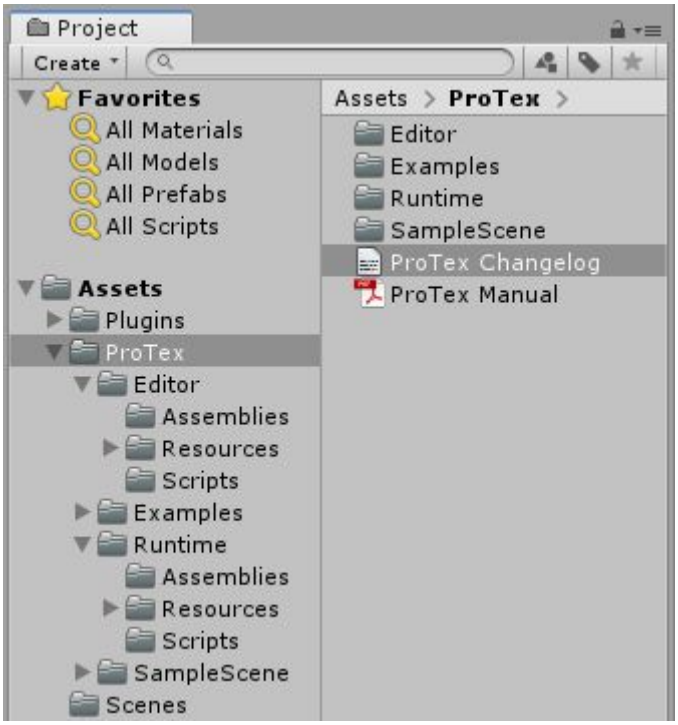
Each procedural texture can generate six different output textures (color, normal, height, metallic, occlusion and emission), and you can easily apply them to the Unity default materials using the ProTexMaterialBinder.

Nodes are implemented using compute shaders to reduce the generation time. If your computer does not support compute shaders, the software implementation will be used (but take into account that generation times will be increased).

Contact and support: protexeditor@gmail.com

PACKAGE FOLDER STRUCTURE

The ProTex package includes the editor, the runtime, the documentation, many procedural texture examples and a sample scene using the following folder structure:

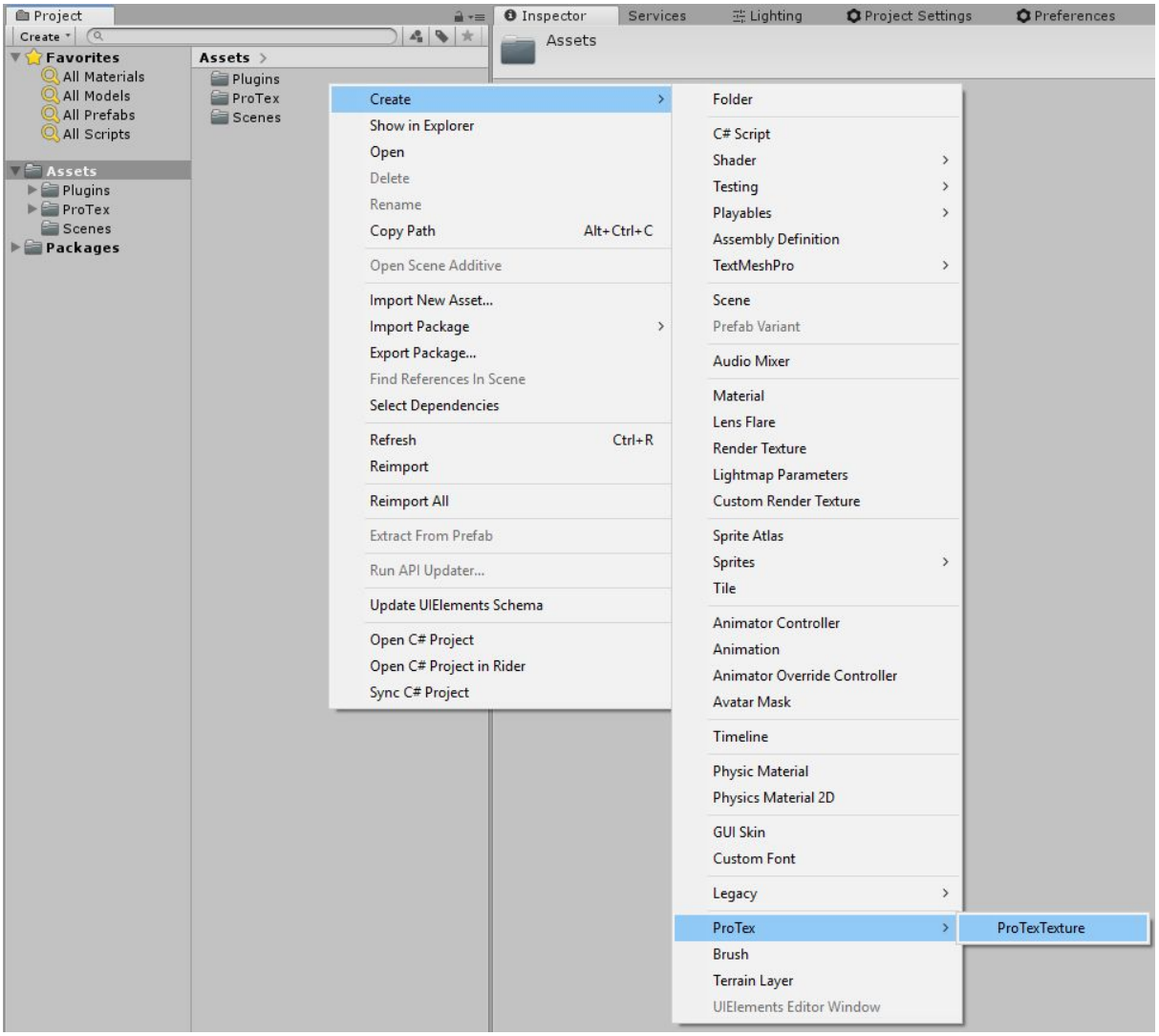


- **Editor**
  Includes the all the resources needed for the ProTex editor to work
  - **Assemblies**. ProTexEditor.dll
  - **Resources**. Resources needed by the ProTex Editor.
  - **Scripts**. *ProTexMaterialBinderInspector.cs* script
- **Examples**
  Procedural texture samples can be found here, ordered by the type of material.
- **Runtime**
  Code and resources needed for the runtime.
  - **Assemblies**. ProTex.dll
  - **Resources**. Resources needed during runtime
  - **Scripts**. *ProTexMaterialBinder.cs* script
- **SampleScene**
  Sample scene demonstrating how to use the *ProTexMaterialBinder* to generate and apply ProTex Textures during runtime.

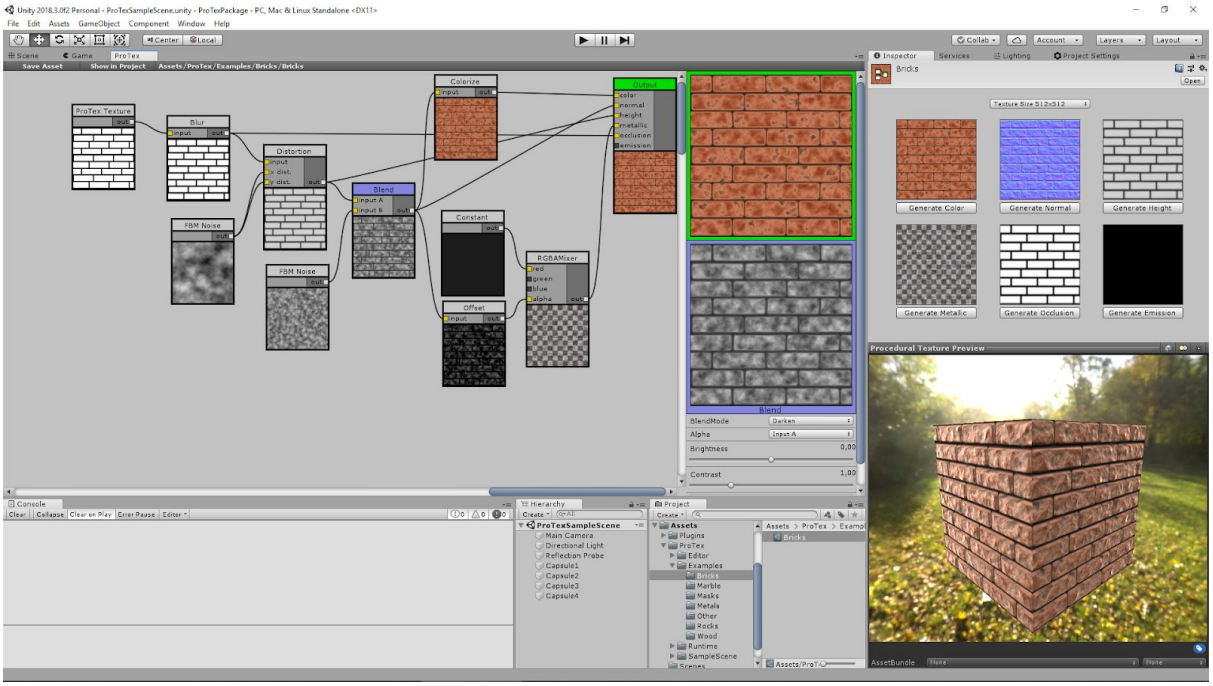
The root folder includes the ProTex documentation, including the *ProTex Manual.pdf* and the *ProTex Changelog.txt*.

CREATING PROTEX TEXTURES

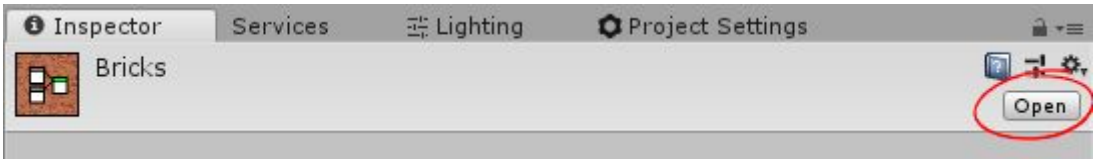
To create a new ProTex Texture, just right click in the Unity Project Window and select ProTex/ProTexTexture. Double clicking on the newly created ProTex Texture will launch the ProTex Editor.



PROTEX EDITOR



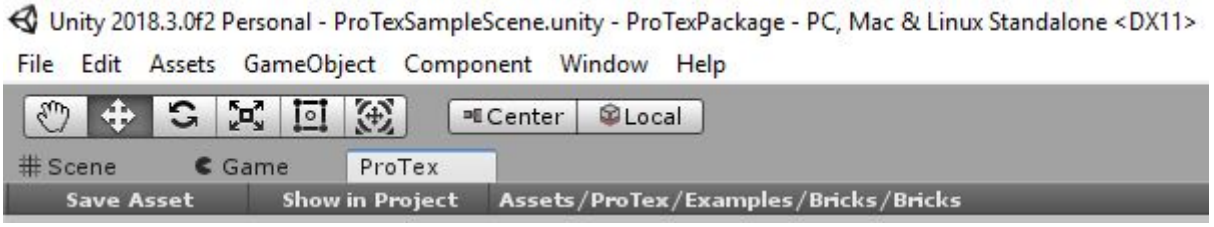
The **ProTex Editor** allows you to create your own procedural textures. Double clicking on a ProTex Texture in the Unity Project window will automatically open this editor, showing the nodes that compose that particular texture. Alternatively, you can click the open button in the Inspector window when you have a ProTex Texture selected.



The ProTex Editor is composed of several different elements:

- Menu Bar
- Workspace Area
- Preview Area
- Node Inspector Area
- ProTex Texture Inspector window

MENU BAR



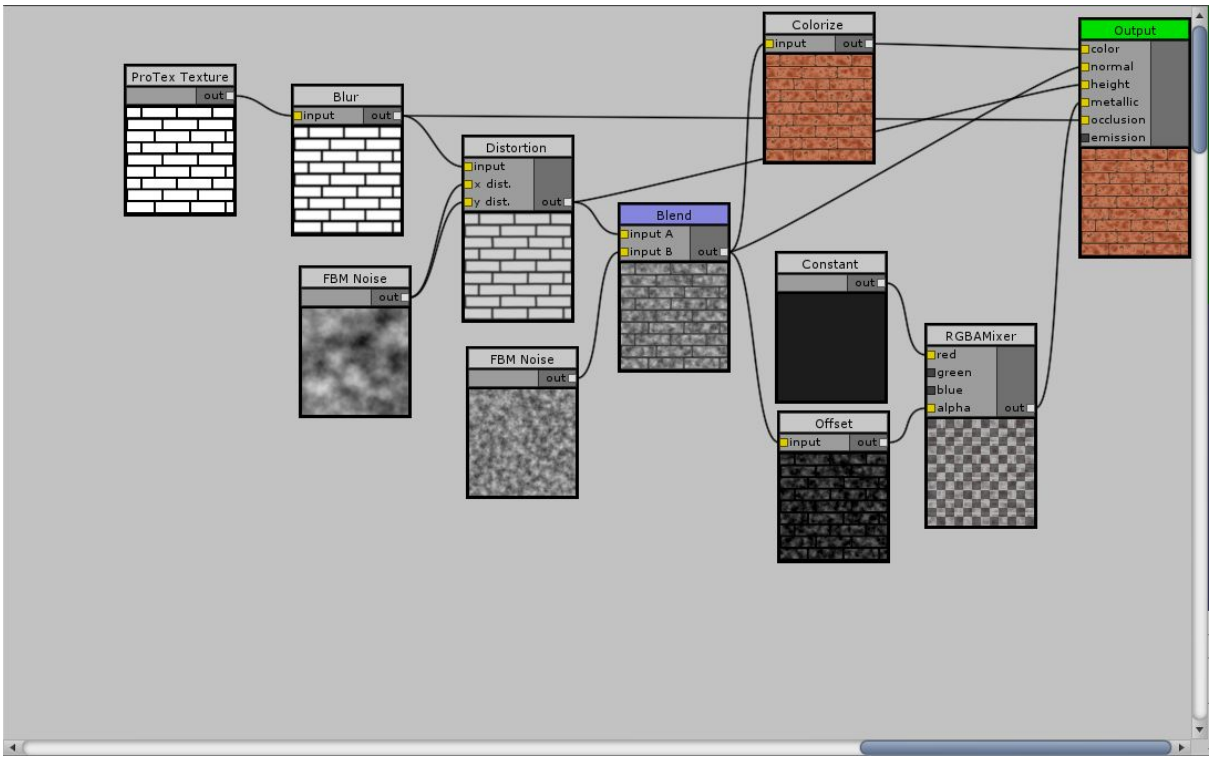
The **Menu Bar** sits on top of the ProTex window, and contains the *Save Asset* button, the *Show in Project* button and the current ProTex Texture path.

The *Save Asset* button will ensure the current ProTex Texture is saved. Note that it will be saved every time you save your project too.

The Show in Project button will locate the current ProTex Texture in the Unity Project window.

The rest of the menu bar will display the current ProTex Texture location path in the project, including its name.

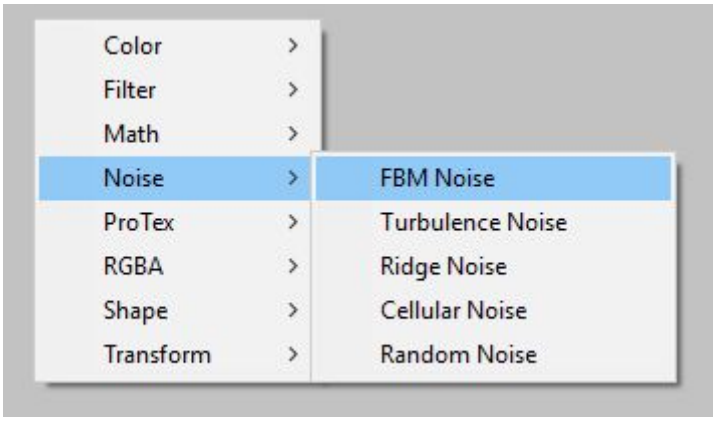
WORKSPACE AREA



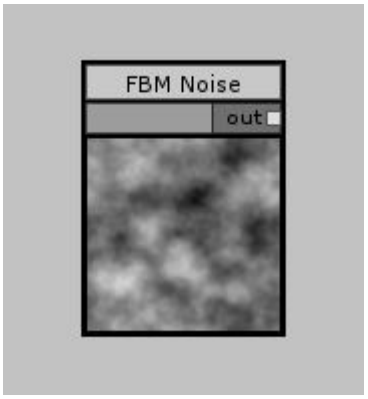
The **Workspace Area** is where you can actually arrange the different nodes that will compose the procedural texture, and link the different nodes inputs and outputs.

CREATING NEW NODES

To create a new node, right click on an empty position on the workspace area and the nodes menu will appear. Select the node you want to create, and the new node will appear in the workspace.

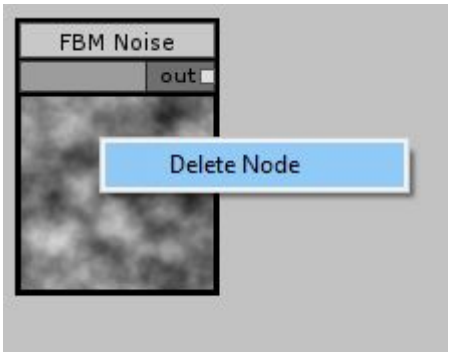


In the workspace, nodes are represented by a window with a title indicating the node type, a variable size are for the node inputs and outputs, and a fixed size texture preview, showing the output of that specific node.



DELETING NODES

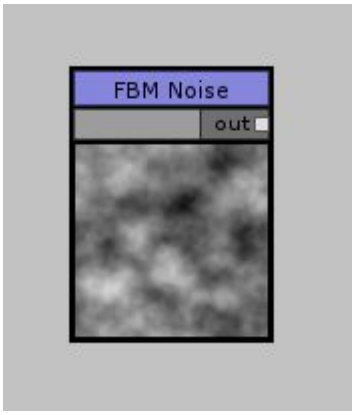
Right click on top on the node you want to delete and the delete node menu will appear. After deleting a node, all connected inputs and output links will disappear.



Alternatively, left click on the node to select it, and press delete key on the keyboard.

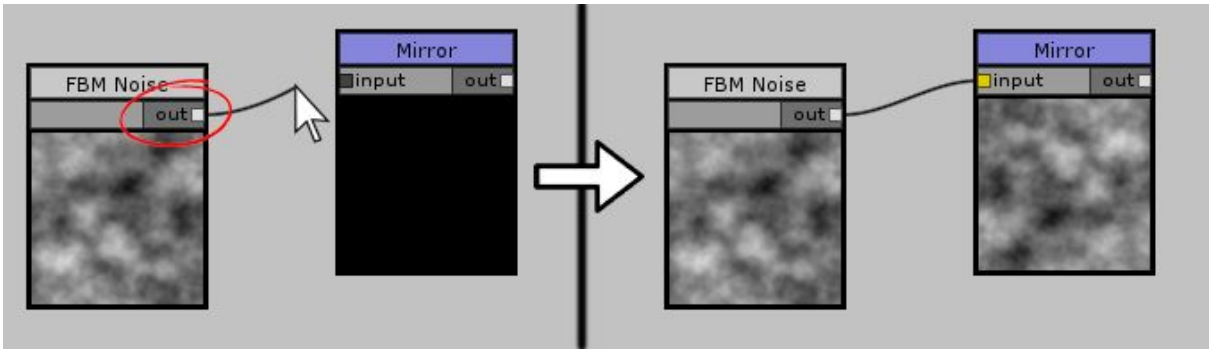
SELECTING NODES

Left click on a node window to select it. That will make that specific node parameters appear in the Node Inspector Area, so you can modify them. The selected node window title will change the color to reflect the selection.

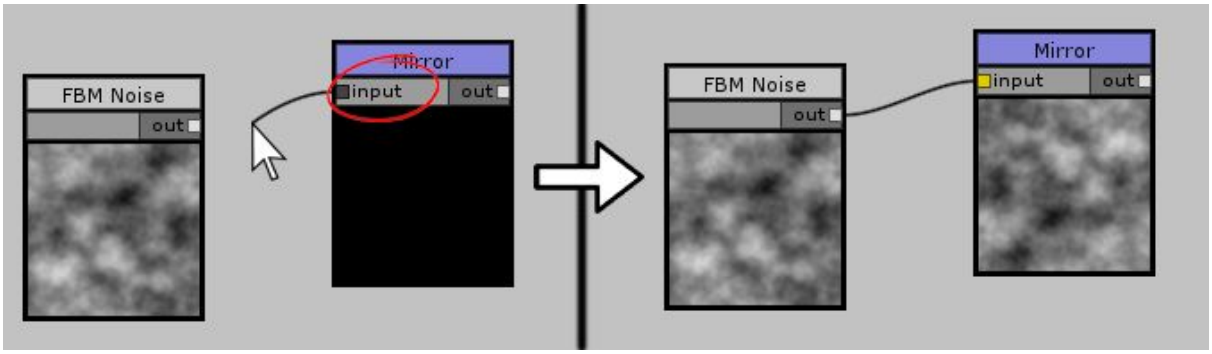


LINKING NODES

To link one node output to another node input, just drag the node output into the desired node input using the left mouse button. The link line will appear on the workspace while dragging. If the drag doesn't end on another node input, the link will be cancelled. If the destination input was already linked, the previous link will be removed.

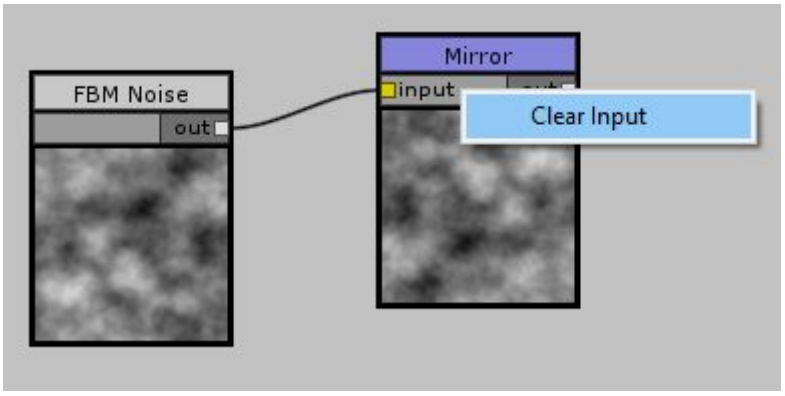


You can also create links starting from the input, dragging it to the desired output. In this case, if the destination output was already linked, the links will be maintained, as the same output can be linked with many different inputs.

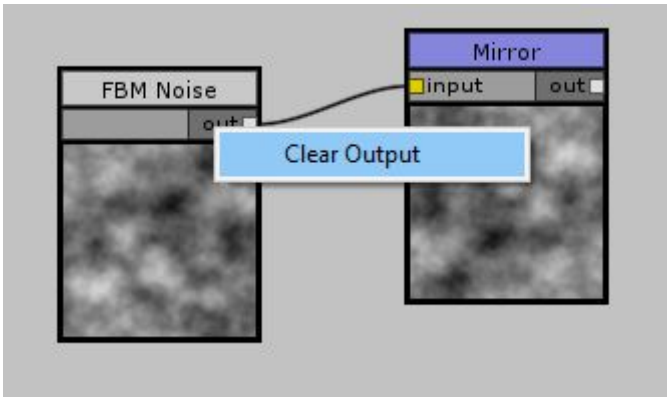


DELETING LINKS

To delete a link, right click on an input to show the clear input menu.



You can also right click on an output to show the clear output menu. Clearing an output will remove all the output connections.



MOVING NODES

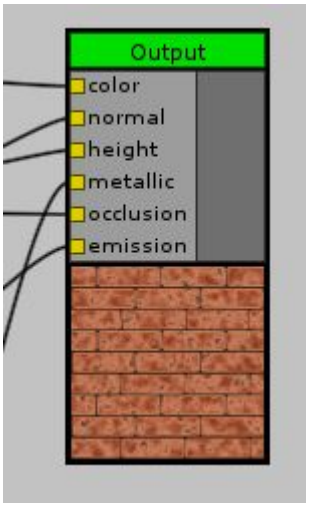
To move a node around the workspace, just drag it using the left mouse button. You have to start the drag on the title or the texture preview area.

MOVING AROUND THE WORKSPACE

The workspace area usually bigger that the space that can be displayed on the screen. You can use the scroll bars in the bottom and right sides of the workspace, or drag the workspace using the left or middle mouse button starting the drag on an empty workspace position.

THE OUTPUT NODE

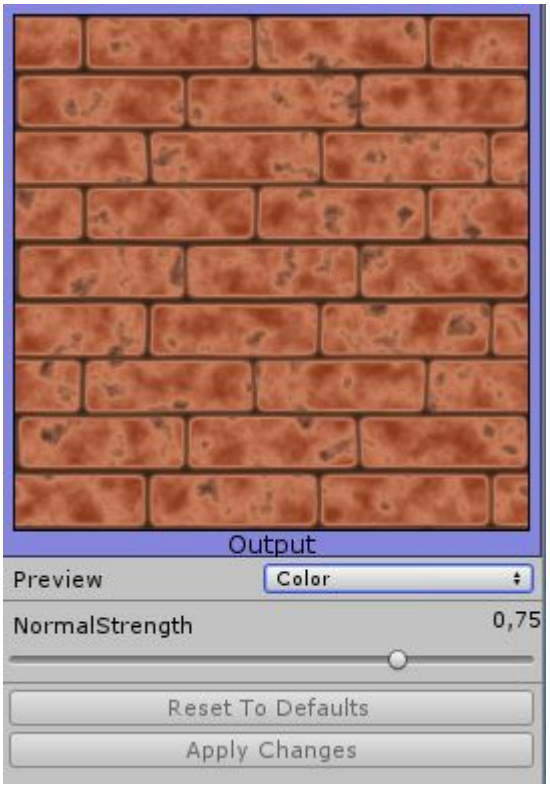
The Output node is a special type of node that will be always present in the workspace area, even the first time you open a new ProTex Texture after creating it. The output of this node will be the output of the ProTex Texture itself. It can't be deleted, and it's the only node that doesn't have an output. To distinguish this node easily from the others, the title bar has a green background.



Each one of the inputs of this node represents a different type of textures that the ProTex Texture can output (*color, normal, height, metallic, occlusion, emission*). As there are six inputs, the output node can have six different outputs and so does the ProTex Texture.

For example, whatever you link with the *emission* input will be the output of the ProTex Texture when a texture of type *Emission* is requested later on.

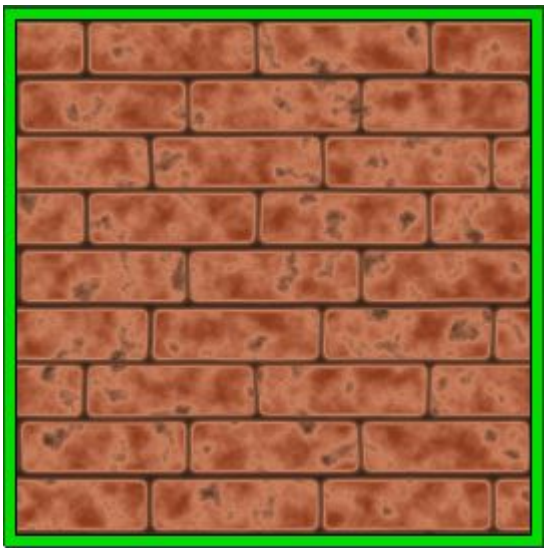
As there are six possible outputs but the node window can only preview one of them at a time, you can choose which one of those outputs will be shown in the Output node preview by selecting it changing the *Preview* parameter in the Node Inspector Area.



Changing the preview type will change the preview in the node window but also in the Preview Area of the editor.

The other parameter you can modify in the Output node is the *NormalStrength*, to alter how strong the generated normal map will be.

PREVIEW AREA

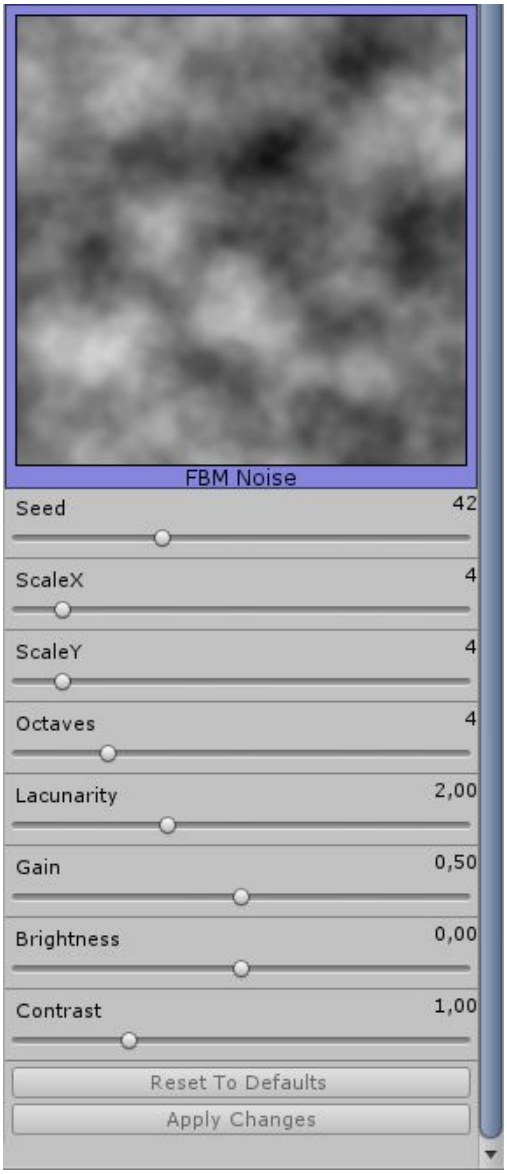


The Preview area is located in the top right corner of the ProTex Editor. It will always display the output texture of the Output node. As the Output node can have six different outputs (*color, normal, height, metallic, occlusion, emission*), the one that will be displayed will be the one selected by the Output node *Preview* parameter.

Like the Output node, the Preview area will be surrounded by a green border.

The Preview Area will always be visible in the ProTex Editor, and is not dependant on the node selected in the Workspace Area.

NODE INSPECTOR AREA



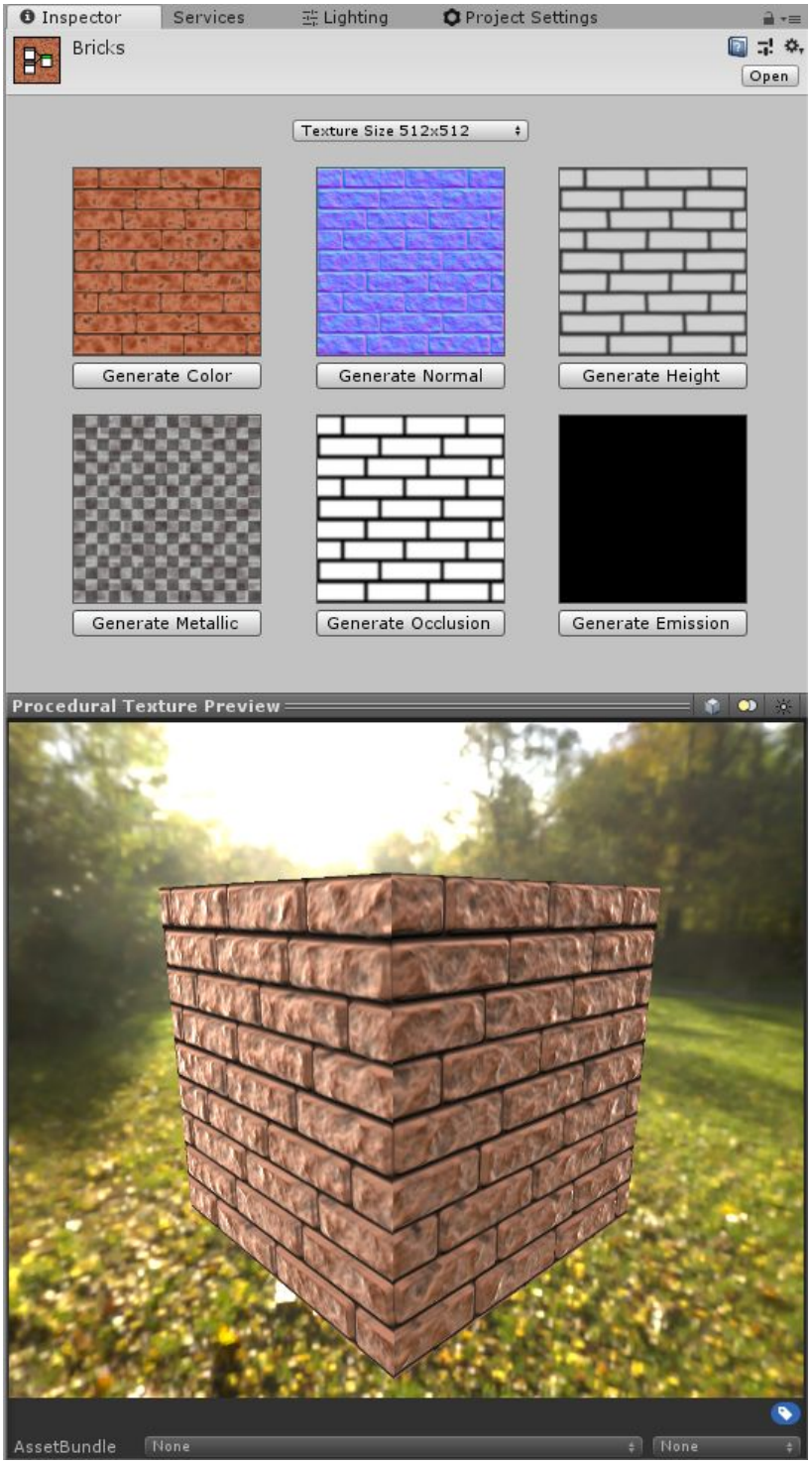
Once a node is selected, its parameters will be displayed in the Node Inspector Area. The upper part of the inspector will show the output texture of that node, while the bottom part will show the actual parameters and their current values.

Changing the parameters will modify the output texture, but the changes will not be automatically propagated to the selected node. **The changes will only be applied to the selected node when the *Apply Changes* button is pressed**. By default this button will be disabled, but it will be enabled in red color whenever a change is made.

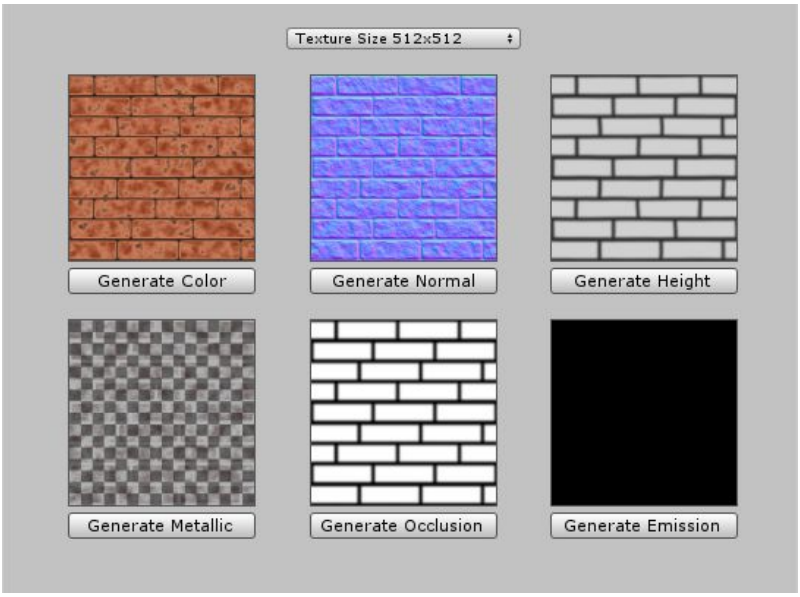
The Reset To Defaults button will reset the node to the default values, and similar to the Apply Changes, will only be active if any of the parameters is different from the default value.

PROTEX TEXTURE INSPECTOR WINDOW

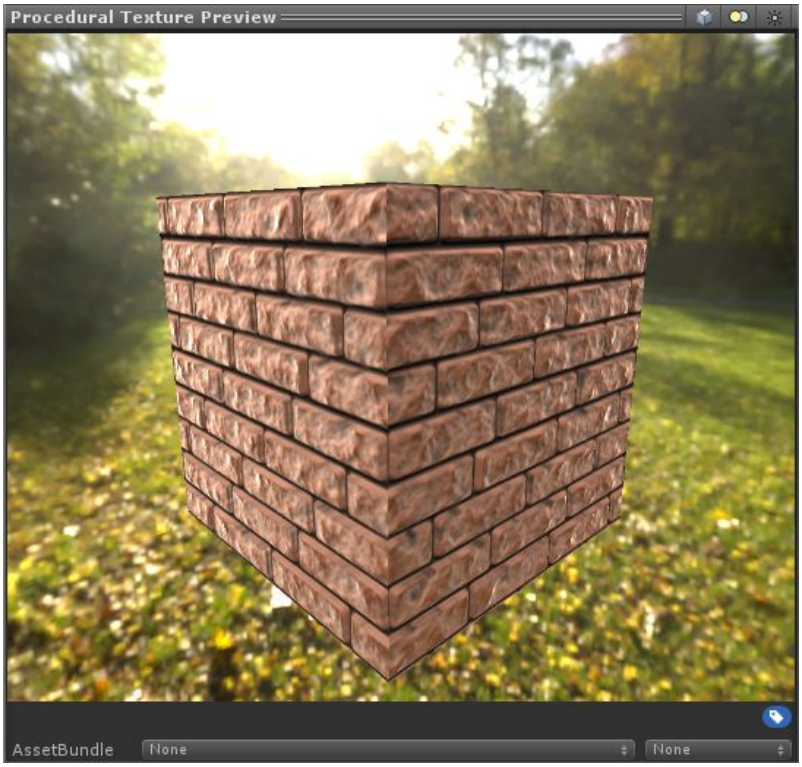
The ProTex Texture Inspector replaces the default Unity Inspector for ProTex Textures. It's not part of the ProTex Editor window, so you will see this inspector whenever you select a texture in the Unity Project window, even if the ProTex Editor is not opened. That also means that you can be editing one texture while inspecting a different one in this window.



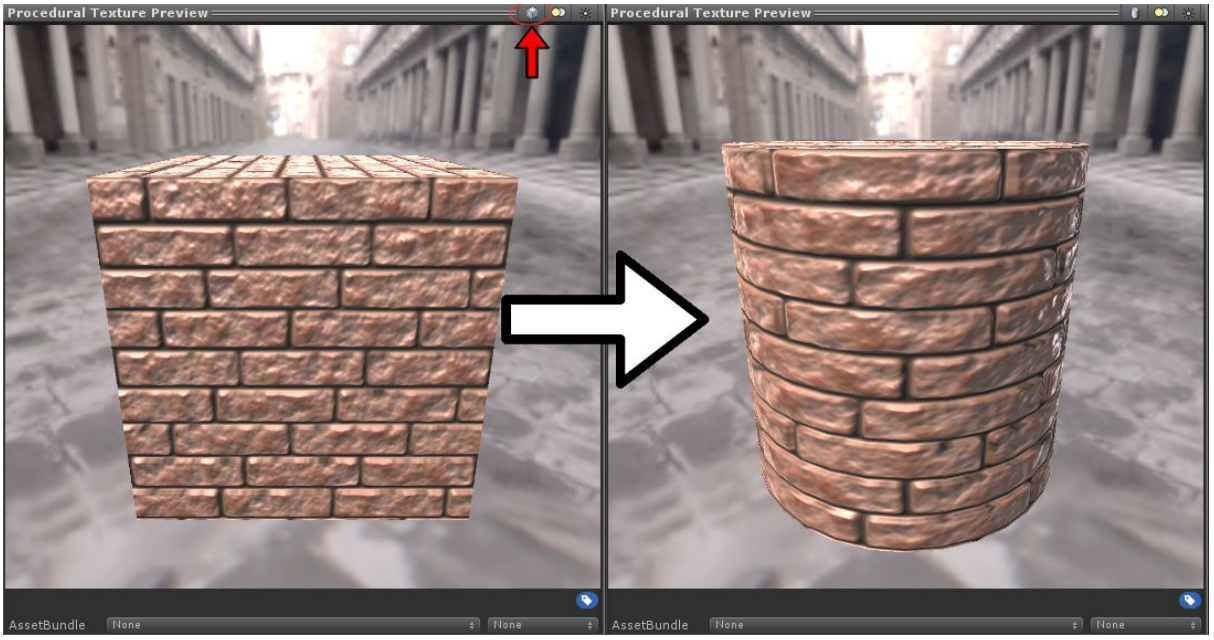
The upper part of the inspector shows a preview of each one of the different texture types that the procedural texture can output (*Color, Normal, Height, Metallic, Occlusion, Emission*). Below each one of those previews there is a "generate" button that allow us to generate the textures directly in the Unity Editor. The generation size can be selected using the drop down selector on top of the inspector. Textures will be saved in the same path and with the same name as the ProTex Texture, but with the texture type and size appended (e.g. if we have a ProTex Texture in the *Assets/ProTexTextures/Test* and we click the *Generate Color* button with a generation size of 521x512 selected, the generated texture will be saved as *Assets/ProTexTextures/Test_Color_512x512)*.



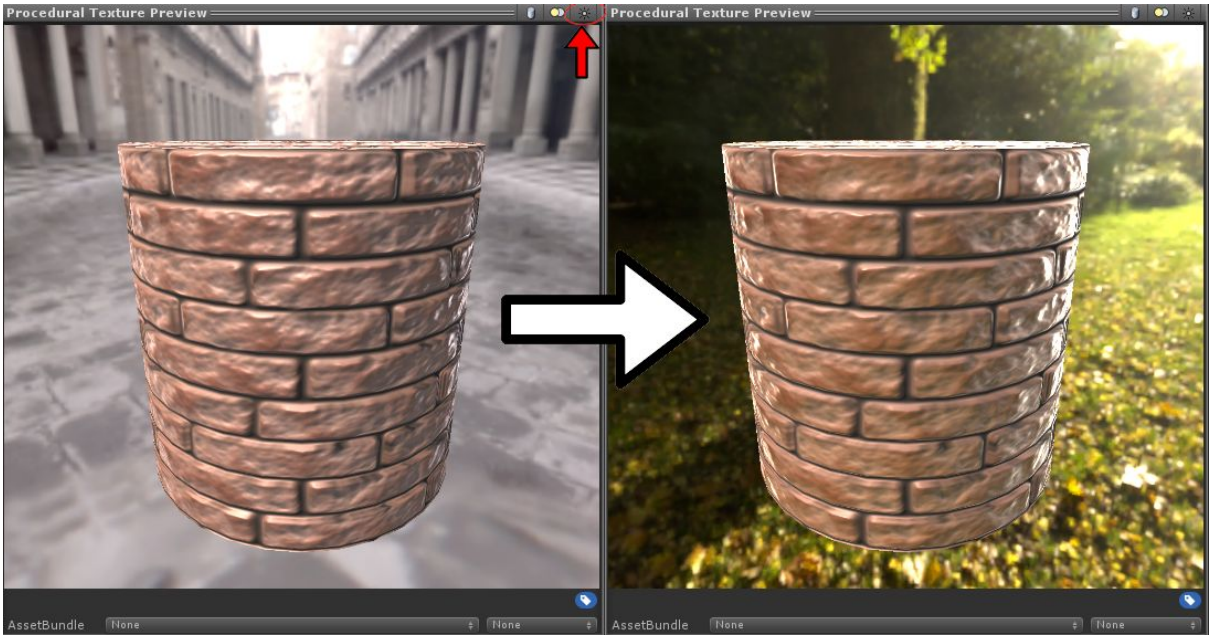
The bottom part of the inspector is dedicated to the Procedural Texture Preview. Similar to the standard Unity Material Preview, it's a quick way to preview how the textures generated by the ProTex Texture would look like if they were applied to the unity default PBR material.



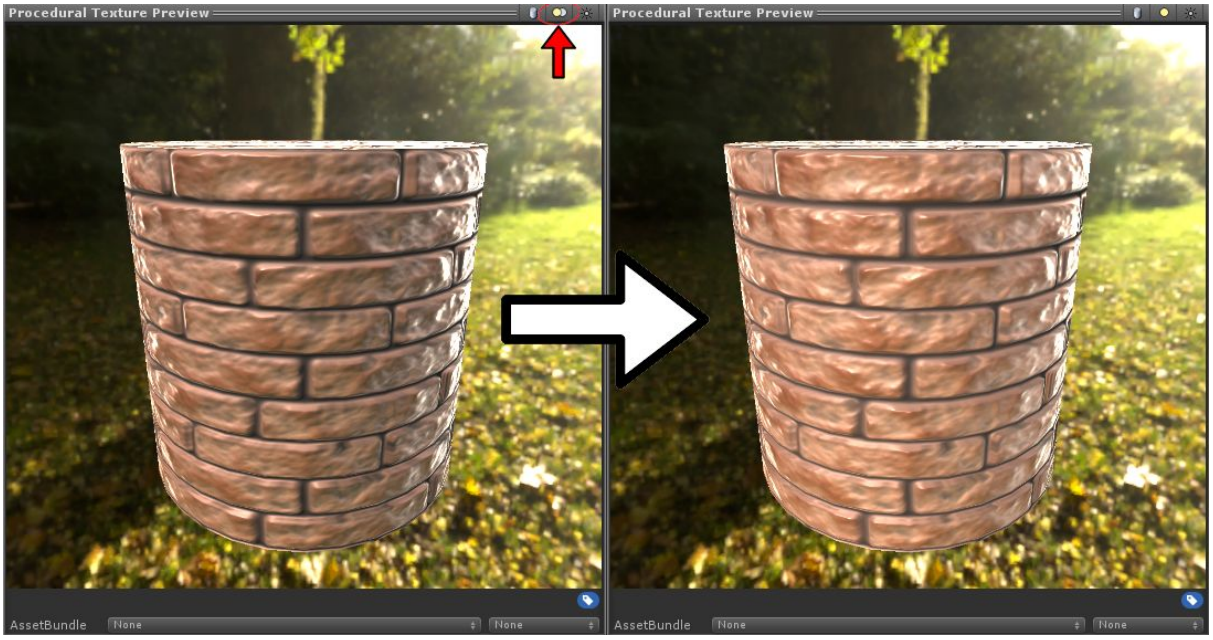
This is an interactive preview, so you can rotate around the preview object dragging with the mouse and change the preview object, the number of lights and the environment by using the preview menu bar. To change the type of object, click the object button in the preview menu. You can choose between sphere, cube, cylinder and torus.



You can also change the environment skybox. The first environment is the Uffizi gallery, the second one is a forest scene, the third one will use the skybox from you current Unity open scene and the last one will disable the skybox showing a gray background.



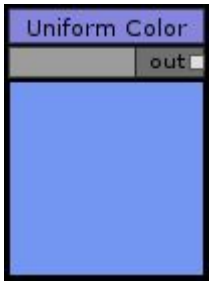
Finally you can change the light setup, choosing between one and two lights in the scene.



NODES REFERENCE

COLOR NODES

UNIFORM COLOR

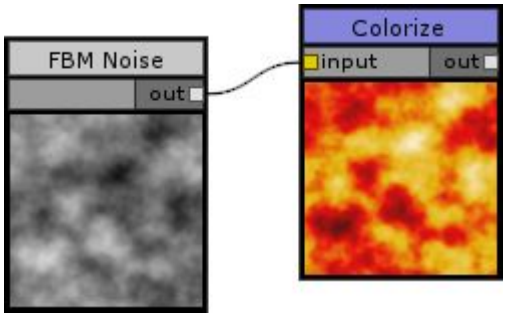


Generates a texture with a uniform color. If you intend to generate a grayscale texture, it's better to use the Constant node.

Parameters

- **Color**
  Shows a color picker to select the texture color.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

COLORIZE



Colorizes input texture based on a color gradient.

Inputs

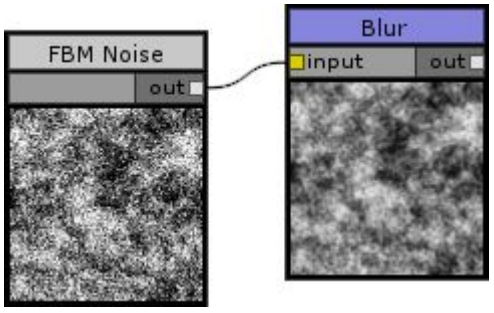
- **Input**
  Texture to colorize.

Parameters

- **Interpolator**
  Value that will be used to select the color for each pixel. Possible values are: *Luminance, Red Channel, Green Channel, Blue Channel, Alpha Channel, Max RGB, Min RGB, Average RGB*
- **Colors**
  Show a gradient color picker to select the desired colors.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

FILTER NODES

BLUR



Blurs the input texture.

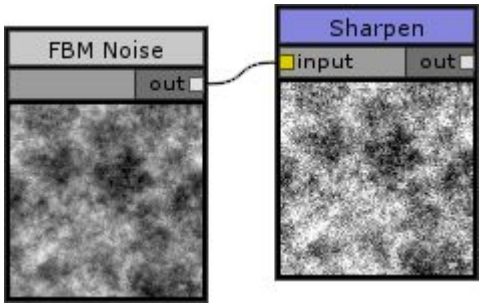
Inputs

- **Input**
  Texture to blur.

Parameters

- **Strength** [0, 1]
  Intensity of the filter effect.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

SHARPEN



Sharpens the input texture.

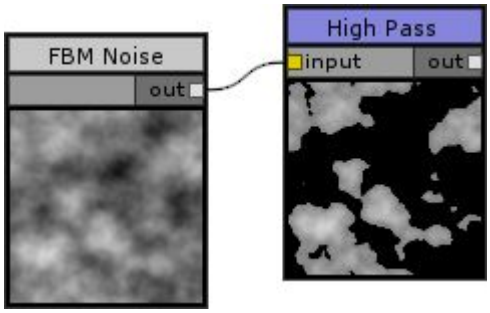
Inputs

- **Input**
  Texture to sharpen.

Parameters

- **Strength** [0, 1]
  Intensity of the filter effect.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

HIGH PASS



Discards input values lower than a threshold. Values lower than the threshold will be set to zero.

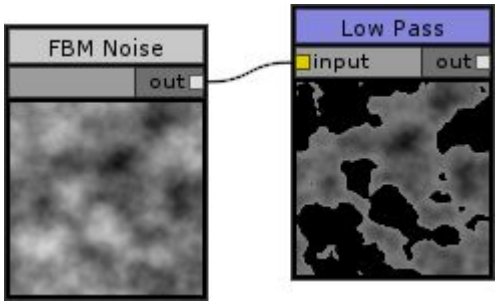
Inputs

- **Input**
  Texture to apply the filter to.

Parameters

- **Threshold**
  Shows a color picker to select the desired threshold. As the threshold is a color, the filter will be applied for each texture channel (RGBA).
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Original*: alpha channel will not be modified.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the filter will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

LOW PASS



Discards input values higher than a threshold. Values higher than the threshold will be set to zero.

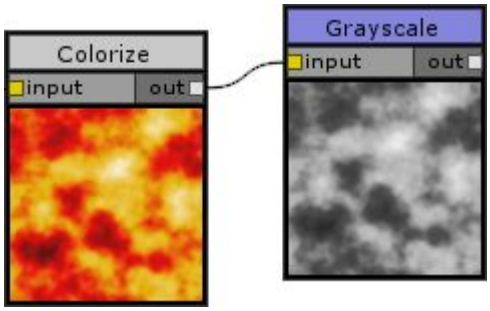
Inputs

- **Input**
  Texture to apply the filter to.

Parameters

- **Threshold**
  Shows a color picker to select the desired threshold. As the threshold is a color, the filter will be applied for each texture channel (RGBA).
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Original*: the alpha channel will not be modified.
  - *One*: the alpha channel will be set to one.
  - *Zero*: the alpha channel will be set to zero.
  - *Operation*: the filter will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

GRAYSCALE



Converts the input texture to grayscale.

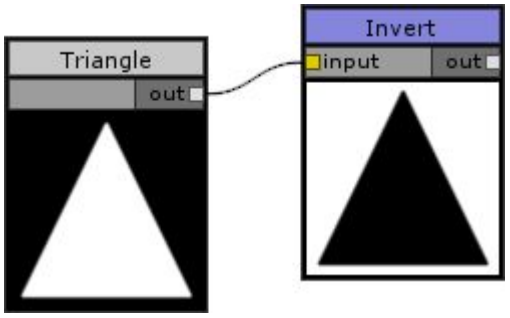
Inputs

- **Input**
  Texture to convert to grayscale.

Parameters

- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

INVERT



Inverts the input texture. The result of this node will be identical to the OneMinus node.

Inputs

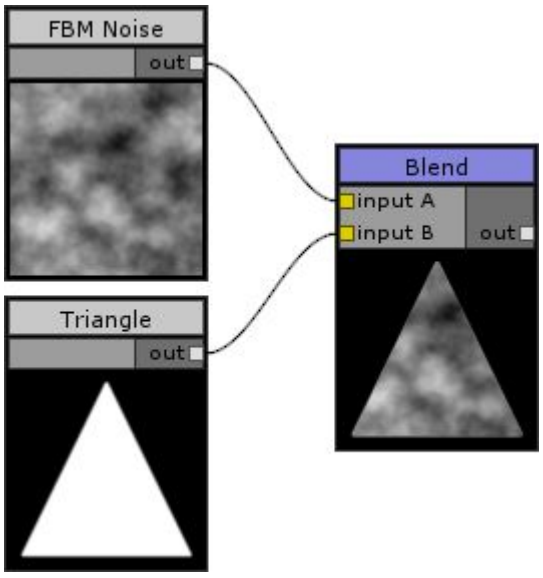
- **Input**
  Texture to invert.

Parameters

- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

MATH NODES

BLEND



Blends input A and input B using the selected blend mode.

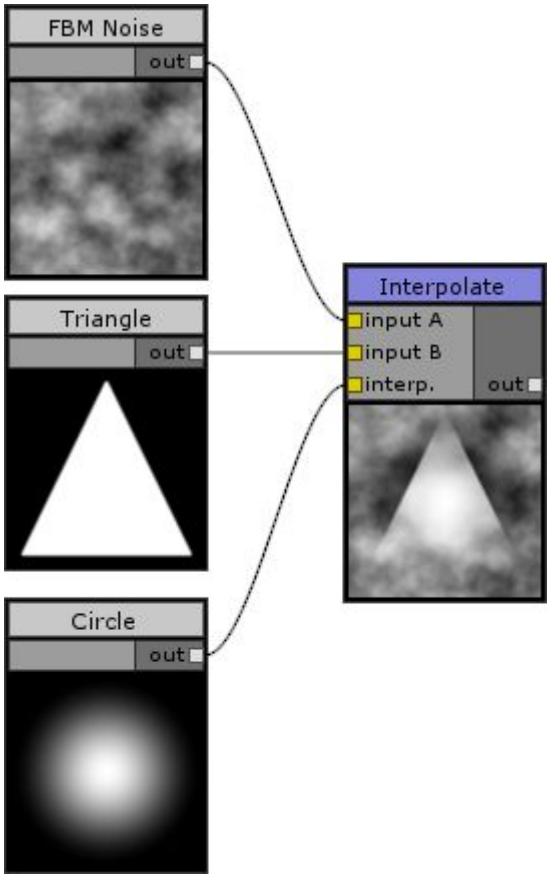
Inputs

- **Input A**
  First texture to blend.
- **Input B**
  Second texture to blend.

Parameters

- **Blend Mode**
  Blend mode used to blend the two textures. Possible values are:
  Darken, Multiply, ColorBurn, LinearBurn, Lighten, Screen, ColorDodge, LinearDodge, Overlay, SoftLight, HardLight, VividLight, LinearLight, PinLight, HardMix, Difference, Exclusion.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Input A*: the alpha channel of the input A will be used.
  - *Input B*: the alpha channel of the input B will be used.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the alpha channel will be blended too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

INTERPOLATE



Does a linear interpolation between input A and input B based on the value of interp. (shortcut for interpolator).

Inputs

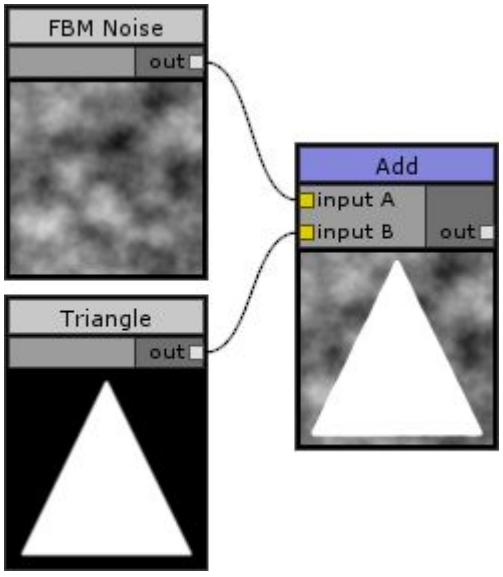
- **Input A**
  First texture to blend.
- **Input B**
  Second texture to blend.
- **Interp.**
  Controls the interpolation. Dark pixels will select input A, light colors will select input B

Parameters

- **Offset** [-1, 1]
  Offset the interpolation result towards input A (negative) or input B (positive)
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Input A*: the alpha channel of the input A will be used.

    - *Input B*: the alpha channel of the input B will be used.
    - *One*: alpha channel will be set to one.
    - *Zero*: alpha channel will be set to zero.
    - *Operation*: the alpha channel will be interpolated too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

ADD



Adds input A and input B.

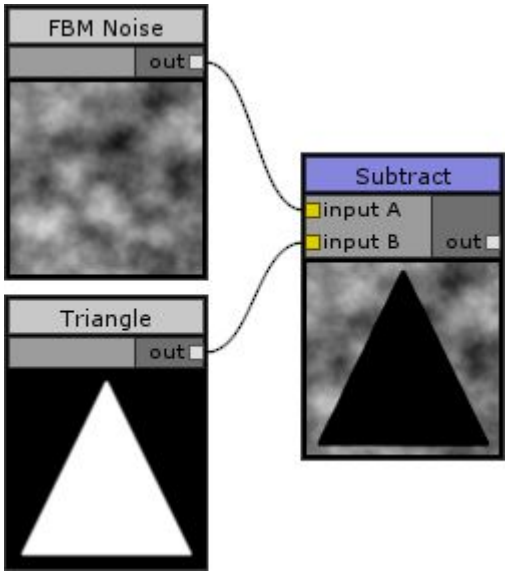
Inputs

- **Input A**
  First texture to add.
- **Input B**
  Second texture to add.

Parameters

- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Input A*: the alpha channel of the input A will be used.
  - *Input B*: the alpha channel of the input B will be used.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

SUBTRACT



Subtracts input B from input A.

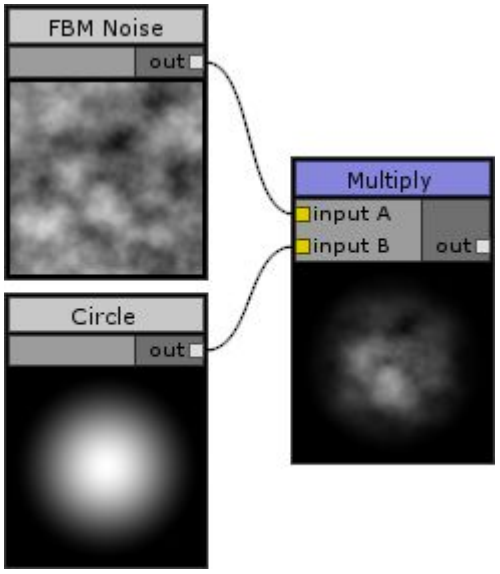
Inputs

- **Input A**
  Base texture.
- **Input B**
  Texture to subtract to Input A.

Parameters

- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Input A*: the alpha channel of the input A will be used.
  - *Input B*: the alpha channel of the input B will be used.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

MULTIPLY



Multiplies input A by input A.

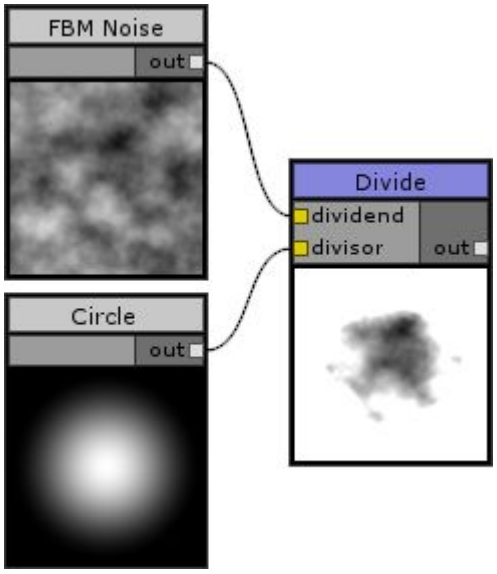
Inputs

- **Input A**
  First texture to multiply.
- **Input B**
  Second texture to multiply.

Parameters

- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Input A*: the alpha channel of the input A will be used.
  - *Input B*: the alpha channel of the input B will be used.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

DIVIDE



Divides dividend texture values by divisor texture values.

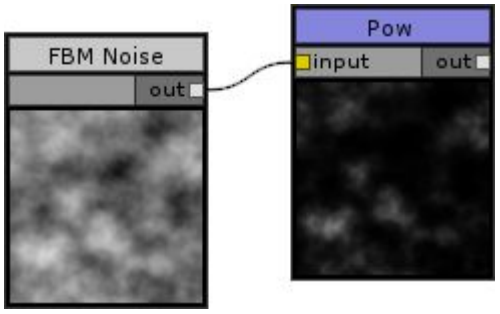
Inputs

- **Dividend**
  First texture to multiply.
- **Divisor**
  Second texture to multiply.

Parameters

- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Input A*: the alpha channel of the input A will be used.
  - *Input B*: the alpha channel of the input B will be used.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

POW



Raises input texture to Power value.

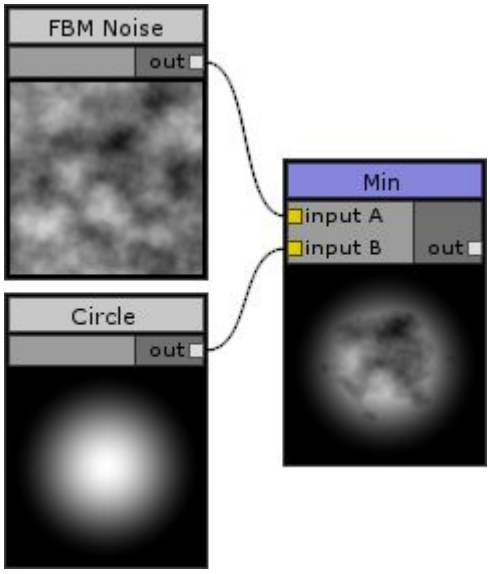
Inputs

- **Input**
  Texture to raise.

Parameters

- **Power** [1, 32]
  Value to raise the input texture to.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Original*: the alpha channel will not be modified.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

MIN



Selects the minimum value between input A and input B.

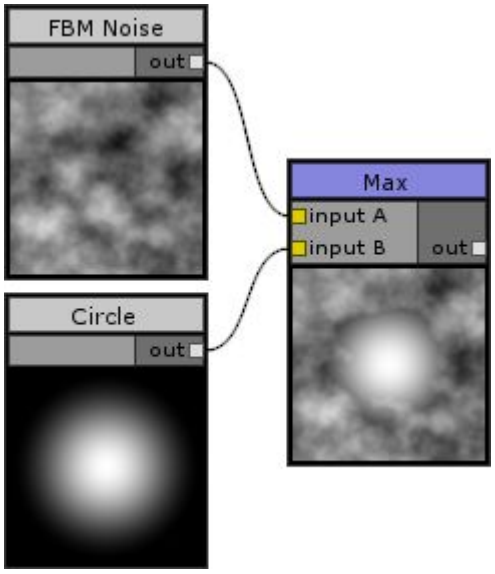
Inputs

- **Input A**
  First texture to evaluate.
- **Input B**
  Second texture to evaluate.

Parameters

- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Input A*: the alpha channel of the input A will be used.
  - *Input B*: the alpha channel of the input B will be used.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

MAX



Selects the maximum value between input A and input B.

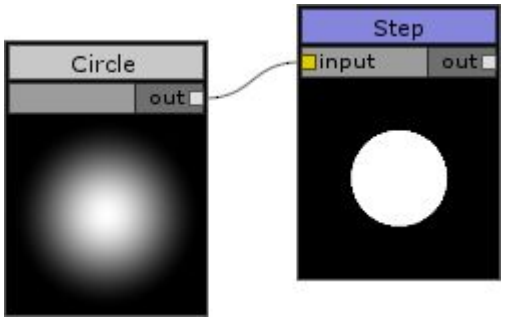
Inputs

- **Input A**
  First texture to evaluate.
- **Input B**
  Second texture to evaluate.

Parameters

- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Input A*: the alpha channel of the input A will be used.
  - *Input B*: the alpha channel of the input B will be used.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

STEP



Generates a step function by comparing the input texture values to a threshold. It returns 0 if the value is lower than the threshold, and 1 otherwise.

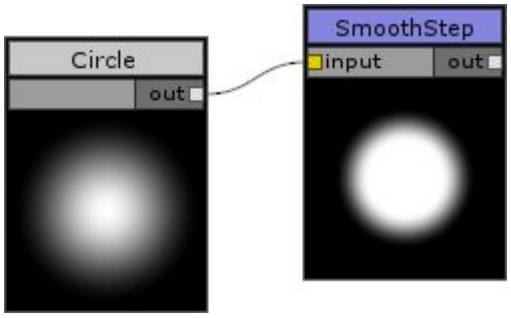
Inputs

- **Input**
  Texture with the values to compare with the threshold.

Parameters

- **Threshold** [0, 1]
  Value to compare the input texture values with.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Original*: the alpha channel will not be modified.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

SMOOTHSTEP



Performs a smooth Hermite interpolation between 0 and 1 when *ThresholdMin* < input < *ThresholdMax*. Similar to the Step node, but with a smooth transition

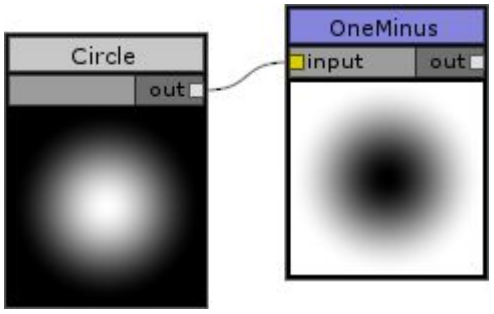
Inputs

- **Input**
  Texture with the values to compare with the thresholds.

Parameters

- **ThresholdMin** [0, 1]
  Lower edge of the Hermite function. This value can't be bigger than *ThresholdMax*.
- **ThresholdMax** [0, 1]
  Upper edge of the Hermite function. This value can't be smaller than *ThresholdMin*.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Original*: the alpha channel will not be modified.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

ONEMINUS



Subtracts the input texture values from 1. The result of this node will be identical to the Invert node.

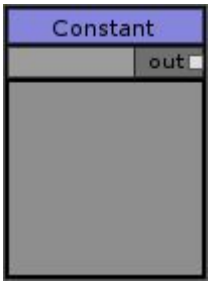
Inputs

- **Input**
  Texture with the values to compare with the thresholds.

Parameters

- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *Original*: the alpha channel will not be modified.
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the math operation will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

CONSTANT



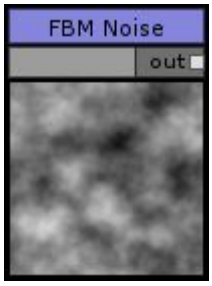
Generates a texture with a constant value. As the same value will be applied to all RGB channels, this node will always generate a grayscale texture. If you want to generate a texture with a constant color, use the Uniform Color node.

Parameters

- **Constant** [0, 1]
  Constant value to use.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

NOISE NODES

FBM NOISE

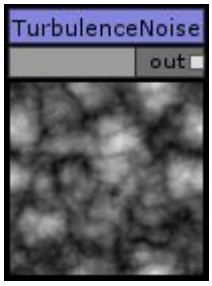


FBM (Fractal Brownian Motion) Noise is a fractal noise generator. Very similar to Perlin Noise, but based on the Open Simplex Noise implementation.

Parameters

- **Seed** [1, 128]
  Changes the noise pattern by changing the original noise seed.
- **ScaleX** [1, 32]
  Changes the noise horizontal scale.
- **ScaleY** [1, 32]
  Changes the noise vertical scale.
- **Octaves** [1, 16]
  Number of iterations used to generate the noise. Increasing this number increases the node generation time, so it's recommended to use the default value (4) unless you are generating a big texture and want the maximum possible noise detail.
- **Lacunarity** [1, 4]
  Frequency multiplicator for each one of the octaves. The default value of 2 means that each octave will double the frequency of the previous one.
- **Gain** [0, 1]
  Amplitude multiplicator for each one of the octaves. The default value of 0.5 means that each octave will contribute to the final texture half than the previous one.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

TURBULENCE NOISE

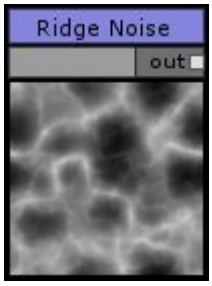


Fractal noise generator.

Parameters

- **Seed** [1, 128]
  Changes the noise pattern by changing the original noise seed.
- **ScaleX** [1, 32]
  Changes the noise horizontal scale.
- **ScaleY** [1, 32]
  Changes the noise vertical scale.
- **Octaves** [1, 16]
  Number of iterations used to generate the noise. Increasing this number increases the node generation time, so it's recommended to use the default value (4) unless you are generating a big texture and want the maximum possible noise detail.
- **Lacunarity** [1, 4]
  Frequency multiplicator for each one of the octaves. The default value of 2 means that each octave will double the frequency of the previous one.
- **Gain** [0, 1]
  Amplitude multiplicator for each one of the octaves. The default value of 0.5 means that each octave will contribute to the final texture half than the previous one.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

RIDGE NOISE

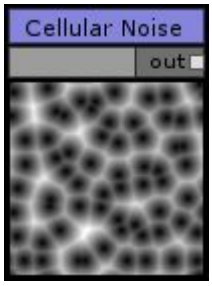


Fractal noise generator.

Parameters

- **Seed** [1, 128]
  Changes the noise pattern by changing the original noise seed.
- **ScaleX** [1, 32]
  Changes the noise horizontal scale.
- **ScaleY** [1, 32]
  Changes the noise vertical scale.
- **Octaves** [1, 16]
  Number of iterations used to generate the noise. Increasing this number increases the node generation time, so it's recommended to use the default value (4) unless you are generating a big texture and want the maximum possible noise detail.
- **Lacunarity** [1, 4]
  Frequency multiplicator for each one of the octaves. The default value of 2 means that each octave will double the frequency of the previous one.
- **Gain** [0, 1]
  Amplitude multiplicator for each one of the octaves. The default value of 0.5 means that each octave will contribute to the final texture half than the previous one.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

CELLULAR NOISE

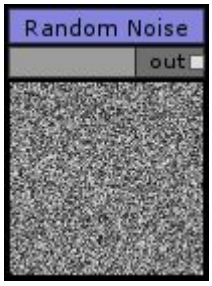


Worley Voronoi-type noise generator.

Parameters

- **Seed** [1, 64]
  Changes the noise pattern by changing the original noise seed.
- **ScaleX** [1, 8]
  Changes the noise horizontal scale.
- **ScaleY** [1, 8]
  Changes the noise vertical scale.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

RANDOM NOISE



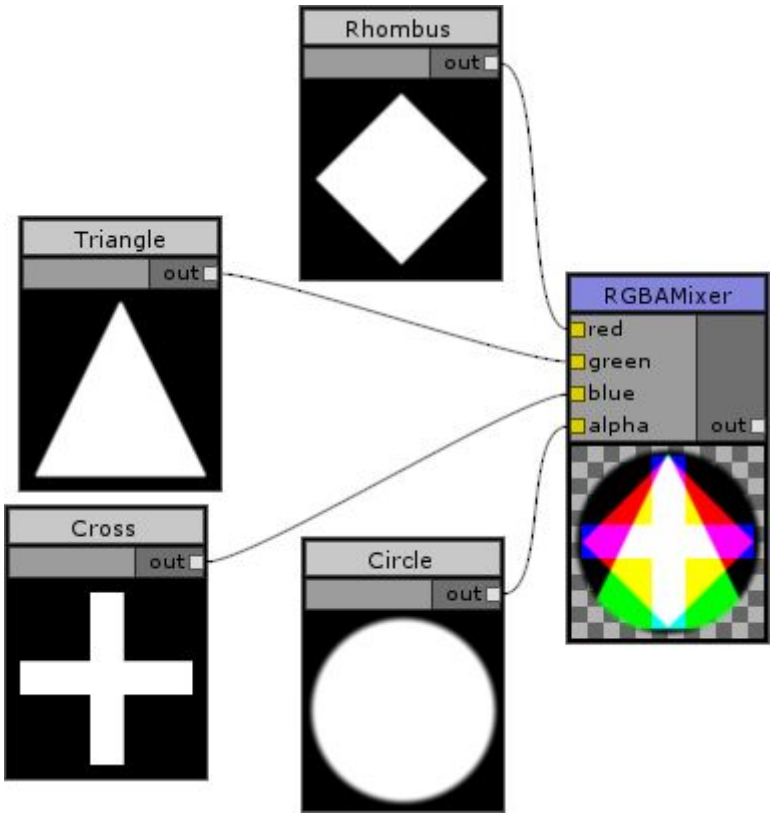
Random noise generator.

Parameters

- **Seed** [1, 64]
  Changes the noise pattern by changing the original noise seed.
- **ScaleX** [1, 9]
  Changes the noise horizontal scale.
- **ScaleY** [1, 9]
  Changes the noise vertical scale.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

RGBA NODES

RGBA MIXER



Combines the selected color channels from the input textures into one texture.

Inputs

- **Red**
  Texture from where to extract the red channel of the final texture.
- **Green**
  Texture from where to extract the green channel of the final texture.
- **Blue**
  Texture from where to extract the blue channel of the final texture.
- **Alpha**
  Texture from where to extract the alpha channel of the final texture.

Parameters

- **RedInput**
  Channel of the *Red* input texture to be used to generate the red channel in the final texture. Possible values are: *Red, Green, Blue, Alpha*
- **GreenInput**
  Channel of the *Green* input texture to be used to generate the green channel in the final texture. Possible values are: *Red, Green, Blue, Alpha*
- **BlueInput**

Channel of the *Blue* input texture to be used to generate the blue channel in the final texture. Possible values are: *Red, Green, Blue, Alpha*

- **AlphaInput**
  Channel of the *Alpha* input texture to be used to generate the alpha channel in the final texture. Possible values are: *Red, Green, Blue, Alpha*
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

PROTEX NODES

PROTEX TEXTURE

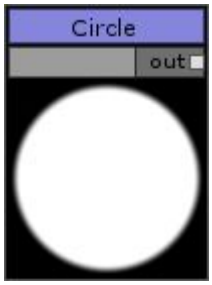
References another ProTex Texture. The output of this node will be the selected output of the referenced ProTex Texture.

Parameters

- **ProTexTexture**
  Shows an object picker to select the ProTex Texture.
- **OutputTexture**
  Selects the desired output texture from the ProTex Texture. Possible values are:
  Color, Normal Source, Height, Metallic, Occlusion, Emission.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

SHAPE NODES

CIRCLE

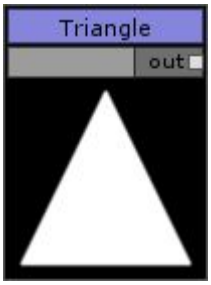


Generates a circle shape.

Parameters

- **Radius** [0, 1]
  Radius of the circle.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

TRIANGLE

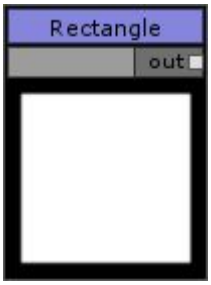


Generates a triangle shape.

Parameters

- **Width** [0, 1]
  Width of the triangle.
- **Height** [0, 1]
  Height of the triangle.
- **Roundness** [0, 1]
  How rounded the corners will be, with a value of 0 being not rounded at all.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

RECTANGLE

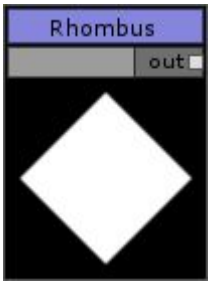


Generates a rectangle shape.

Parameters

- **Width** [0, 1]
  Width of the rectangle.
- **Height** [0, 1]
  Height of the rectangle.
- **Roundness** [0, 1]
  How rounded the corners will be, with a value of 0 being not rounded at all.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

RHOMBUS

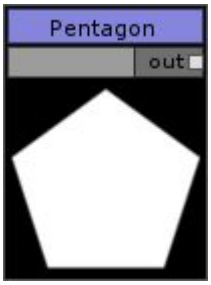


Generates a rhombus shape.

Parameters

- **Width** [0, 1]
  Width of the rectangle.
- **Height** [0, 1]
  Height of the rectangle.
- **Roundness** [0, 1]
  How rounded the corners will be, with a value of 0 being not rounded at all.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

PENTAGON

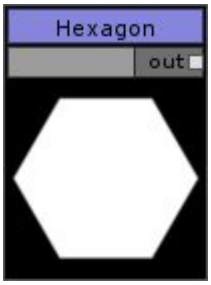


Generates a pentagon shape.

Parameters

- **Radius** [0, 1]
  Radius of the pentagon.
- **Roundness** [0, 1]
  How rounded the corners will be, with a value of 0 being not rounded at all.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

HEXAGON

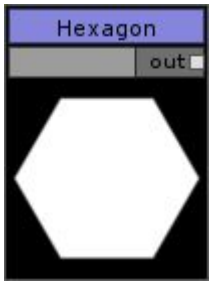


Generates a hexagon shape.

Parameters

- **Radius** [0, 1]
  Radius of the hexagon.
- **Roundness** [0, 1]
  How rounded the corners will be, with a value of 0 being not rounded at all.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

OCTAGON

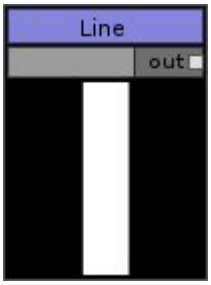


Generates a octagon shape.

Parameters

- **Radius** [0, 1]
  Radius of the octagon.
- **Roundness** [0, 1]
  How rounded the corners will be, with a value of 0 being not rounded at all.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

LINE

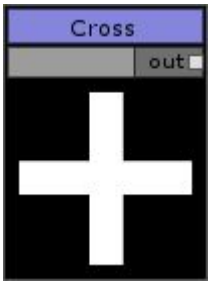


Generates a line shape.

Parameters

- **Direction**
  Selects the line direction. Possible values are:
  Vertical, Horizontal.
- **Thickness** [0, 1]
  Thickness of the line.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

CROSS



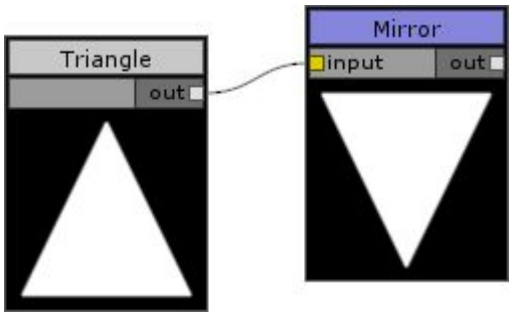
Generates a cross shape.

Parameters

- **Size** [0, 1]
  Size of the cross.
- **Thickness** [0, 1]
  Thickness of the cross.
- **RepeatX** [1, 16]
  Repeats the shape horizontally.
- **RepeatY** [1, 16]
  Repeats the shape vertically.
- **Fallof** [0, 1]
  How smooth the shape border will be, with a value of 0 being a hard border.
- **Alpha**
  How the alpha channel will be calculated. Possible values are:
  - *One*: alpha channel will be set to one.
  - *Zero*: alpha channel will be set to zero.
  - *Operation*: the Constant value will be applied to the alpha channel too.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

TRANSFORM NODES

MIRROR



Mirrors the input texture.

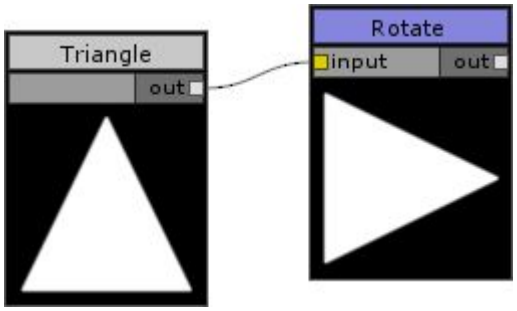
Inputs

- **Input**
  Texture to mirror.

Parameters

- **Horizontal**
  Enable to mirror the input texture horizontally.
- **Vertical**
  Enable to mirror the input texture vertically.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

ROTATE



Rotates the input texture.

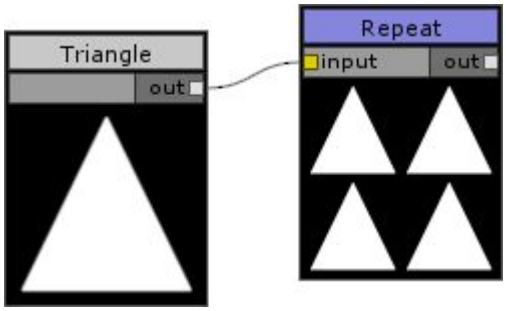
Inputs

- **Input**
  Texture to rotate.

Parameters

- **Rotation**
  Sets the rotation angle. Possible values are (cw stands for clockwise):
  CW_90, CW_180, CW_270.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

REPEAT



Repeats the input texture.

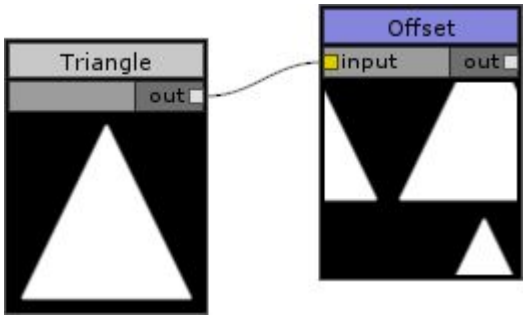
Inputs

- **Input**
  Texture to repeat.

Parameters

- **RepeatX** [1, 16]
  How many times to repeat the texture horizontally.
- **RepeatY** [1, 16]
  How many times to repeat the shape vertically.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

OFFSET



Displaces the input texture.

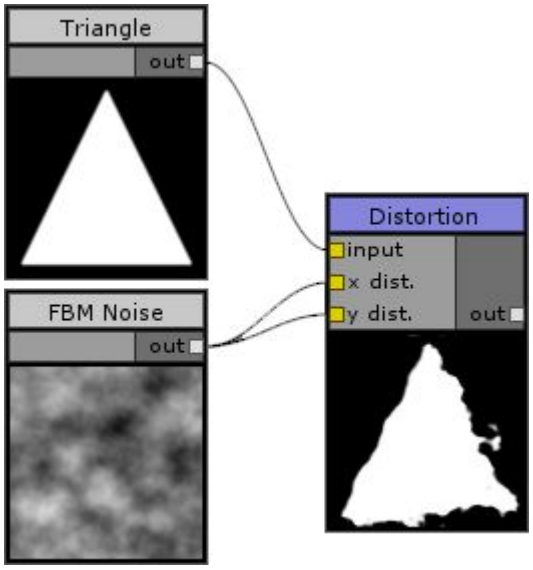
Inputs

- **Input**
  Texture to offset.

Parameters

- **OffsetX** [0, 1]
  How much to displace the input texture horizontally.
- **OffsetX** [0, 1]
  How much to displace the input texture vertically..
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

DISTORTION



Distorts the input texture based on x dist. (horizontally) and y dist. (vertically).

Inputs

- **Input**
  Texture to distort.
- **X dist.**
  Horizontal distortion.
- **Y dist.**
  Vertical distortion.

Parameters

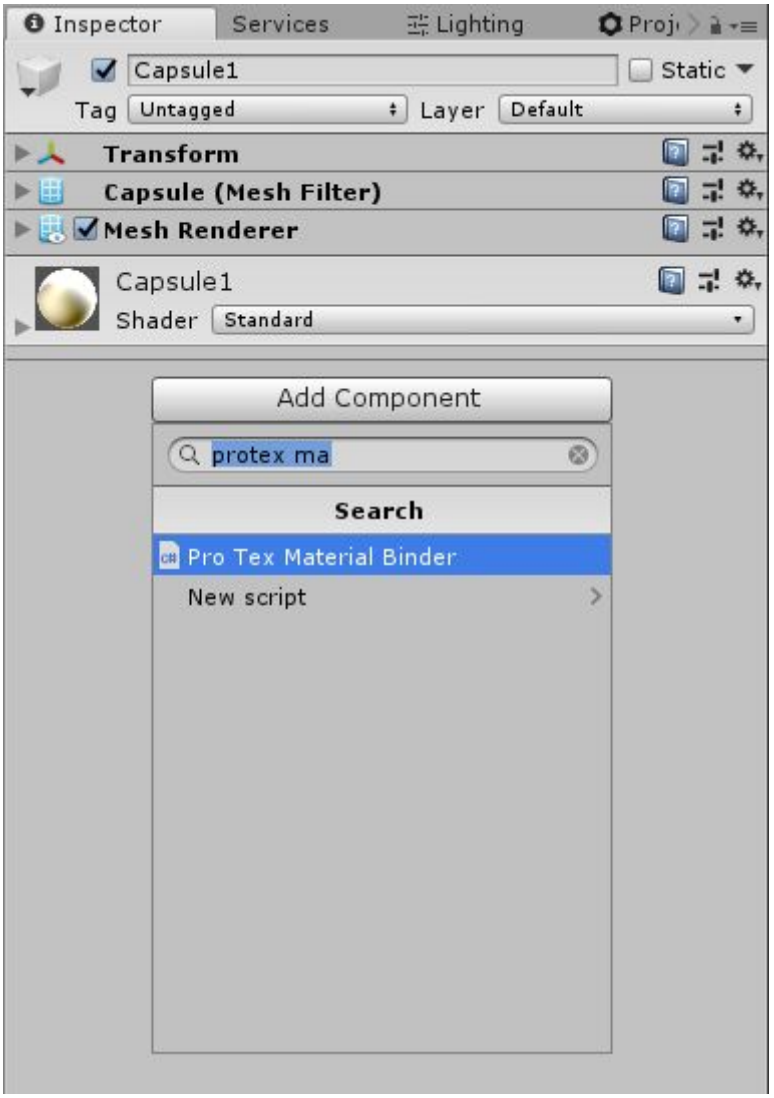
- **ChannelX**
  What channel of the x dist. texture to use. Possible values are:
  Red, Green, Blue, Alpha.
- **DistortionX** [0, 1]
  Strength of the horizontal distortion.
- **ChannelY**
  What channel of the y dist. texture to use. Possible values are:
  Red, Green, Blue, Alpha.
- **DistortionY** [0, 1]
  Strength of the vertical distortion.
- **Brightness** [-1, 1]
  Brightness modificator of the generated texture.
- **Contrast** [0, 4]
  Contrast modificator for the generated texture.

PROTEX MATERIAL BINDER

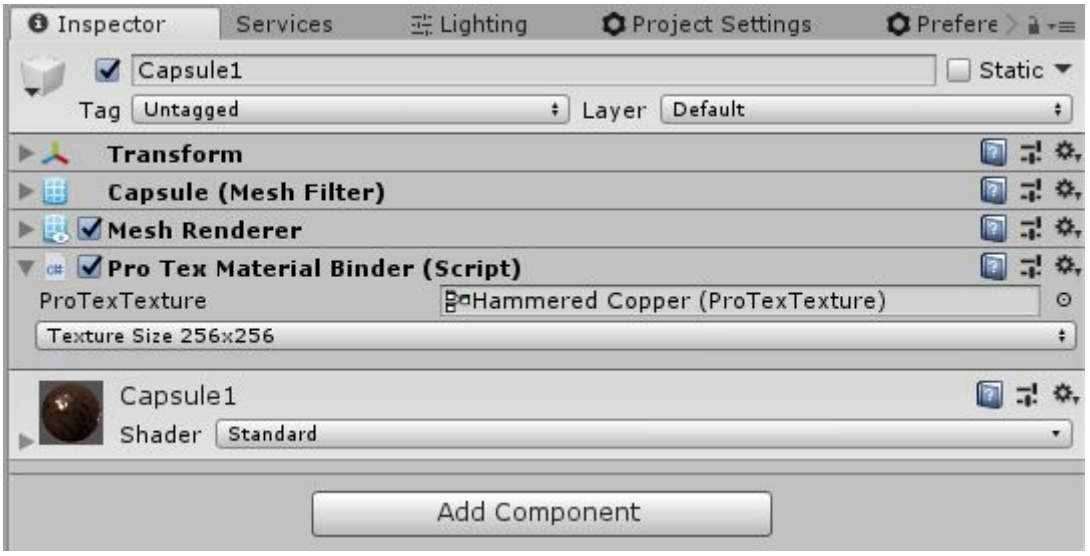
The *ProTexMaterialBinder* is a C# script component that you can add to any game object with a MeshRenderer and a material assigned. This component allows you to assign a ProTex Texture to it, so the texture will be generated during runtime and applied to the material.

The binder will work with the default materials in the standard render pipeline, the High Definition Render Pipeline and the Lightweight Render Pipeline. If you create a custom material and want to apply a ProTex Texture to it, you may need to modify the *ProTexMaterialBinder* or create your own binder. For that, refer to the Using ProTex in Scripts section.

To start using it, add the *ProTexMaterialBinder* component to your game object. You can search by name or go to Scripts/ProTex/ProTexMaterialBinder.



Once the component is added, assign the desired ProTex Texture by dragging and dropping it or using the object picker.



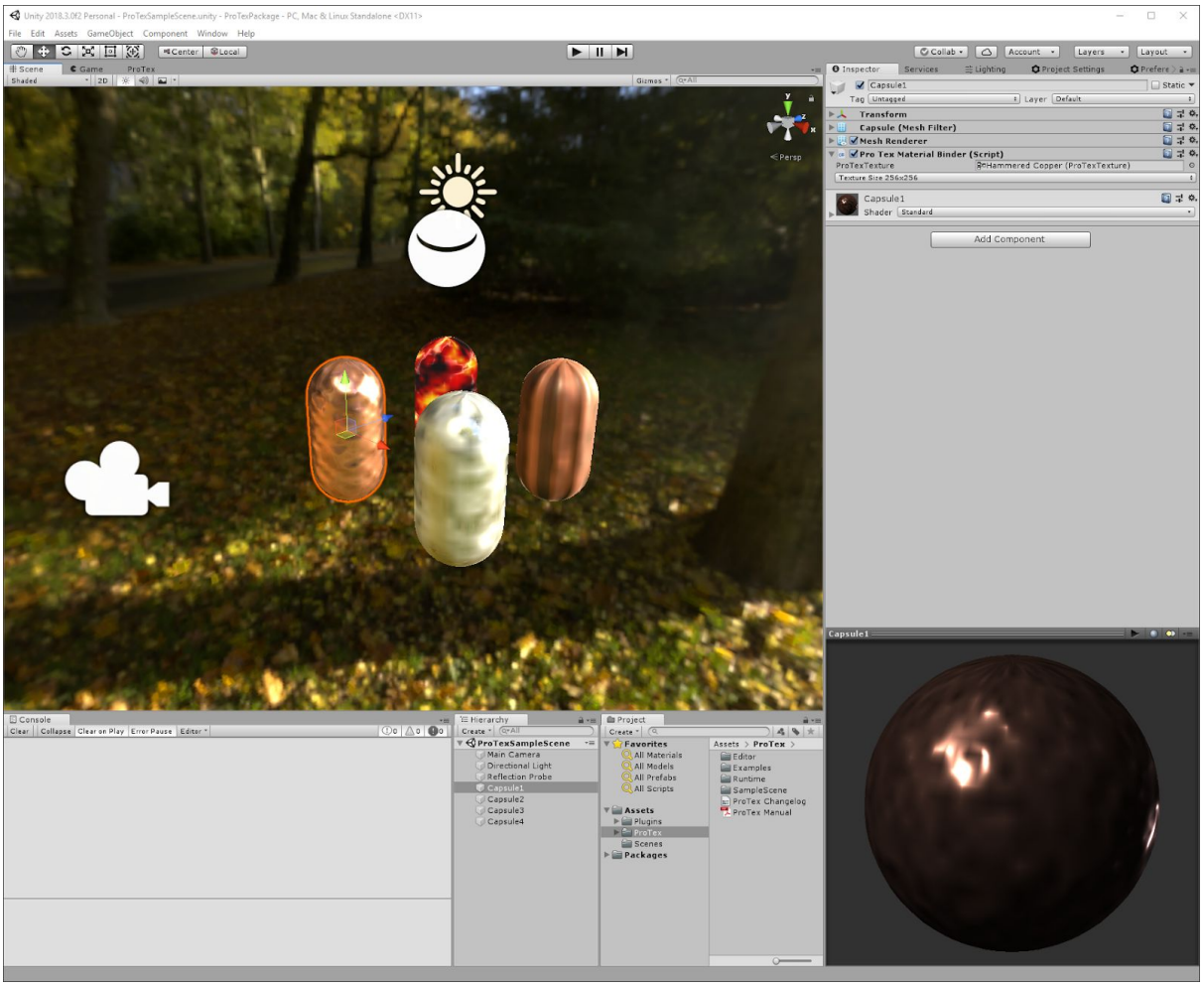
Once a ProTex Texture is selected, a preview of the textures will be automatically generated and applied to the material. As this is just a preview, the texture sizes will be very small (16x16), to generate them quickly and to avoid increasing the scene size.

The actual textures will be generated with the desired size once the game is launched, in the Start method of the *ProTexMaterialBinder.* The generation size can be changed in the dropdown menu, ranging from 64x64 to 2048x2048.

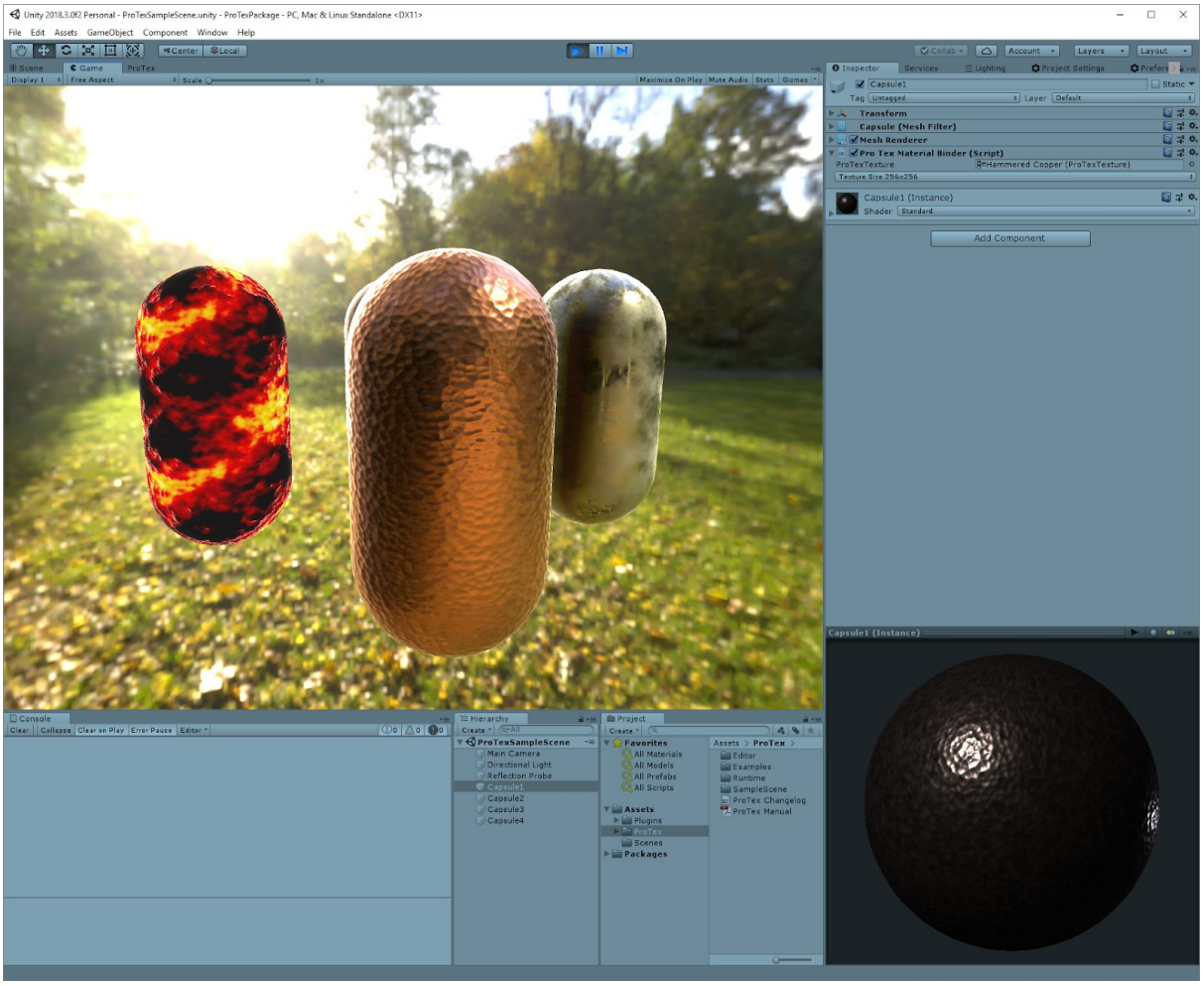
The source code for the *ProTexMaterialBinder* component and the custom inspector *ProTexMaterialBinderInspector* are included in the ProTex Package.

SAMPLE SCENE

The included sample scene shows 4 capsules with a ProTexMaterialBinder component and 4 different ProTex Textures applied to them (*Hammered Copper, Scuffed Iron, Lava* and *Wood Floor 01*). In the Unity Editor, only the preview textures are shown, so they have a small size (16x16).



Pressing the Play button will launch the scene, and the procedural textures will be automatically generated with the size selected (256x256 by default). You can change the texture sizes in the *ProTexMaterialBinder* component, and check the different quality levels. Keep in mind that higher texture sizes will require more memory and time to generate, so try to keep them as small as possible.



USING PROTEX IN SCRIPTS

If you have a reference to a ProTex Texture in a C# script, you can use it to procedurally generate the output textures whenever you want. The ProTexTexture class exposes the following methods:

public Texture2D **GenerateTexture**(int width, int height)
This method will generate the ProTexTexture default output texture. The default output texture is the one selected by the Preview parameter of the Output node. **Parameters**: • **width**: width of the texture to generate • **height**: height of the texture to generate

```
public class Test
{
   public ProTexTexture proTexTexture;

   private void TestGenerateDefaultTexture()
   {
      Texture2D generatedTexture = proTexTexture.GenerateTexture(256, 256);
   }
}
```

Instead of using the default texture output, we can also select the type of texture we want to generate:

public Texture2D **GenerateTexture**(int width, int height, TextureType textureType)
This method will generate the desired ProTexTexture output texture. **Parameters**: • **width**: width of the texture to generate • **height**: height of the texture to generate • **textureType**: type of the texture we want to generate

```
public class Test
{
   public ProTexTexture proTexTexture;

   private void TestGenerateTextures()
   {
      Texture2D colorTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Color);
      Texture2D normalTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Normal);
      Texture2D heightTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Height);
      Texture2D metallicTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Metallic);
      Texture2D occlusionTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Occlusion);
      Texture2D emissionTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Emission);
   }
}
```

```
public enum TextureType
{
    Color,
    Normal,
    Height,
    Metallic,
    Occlusion,
    Emission,
    NormalSource
}
```

The different types of textures that a ProTexTexture can generate.

- *Color*: albedo texture
- *Normal*: normal map texture
- *Height*: height map texture
- *Metallic*: metallic texture, typically used with PBR materials
- *Occlusion*: occlusion texture, typically used with PBR materials
- *Emission*: emission texture
- *NormalSource*: the original texture in the normal input of the Output node.

```
public class Test
{
    public ProTexTexture proTexTexture;

    private void TestGenerateTextures()
    {
        Texture2D colorTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Color);
        Texture2D normalTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Normal);
        Texture2D heightTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Height);
        Texture2D metallicTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Metallic);
        Texture2D occlusionTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Occlusion);
        Texture2D emissionTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Emission);
    }
}
```

Finally, we can check if the procedural texture has a specific type of output texture. If the procedural texture doesn't have a type of texture and we generate it anyway we will just get a black texture with alpha set to 1.

public bool **HasTexture**(TextureType textureType)

Returns true if the procedural texture has that type of output texture.

Parameters:

- **textureType**: type of the texture we want to check

```
public class Test
{
    public ProTexTexture proTexTexture;

    private void TestGenerateTextures()
    {
        Texture2D colorTexture;
        if (proTexTexture.HasTexture(TextureType.Color))
        {
            colorTexture = proTexTexture.GenerateTexture(256, 256, TextureType.Color);
        }
    }
}
```

The *ProTexMaterialBinder.cs* source code is included here to serve as a reference of how to use a ProTex Texture to generate different textures and apply them to a material.

```
using UnityEngine;

namespace ProTex
{

public class ProTexMaterialBinder : MonoBehaviour
{
   //-------------------------------------------------------------------------------------------------------------------
   public ProTexTexture proTexTexture;
   public int runtimeTextureSize = 256;

   //-------------------------------------------------------------------------------------------------------------------
   private const int EditorPreviewTextureSize = 16;

   //-------------------------------------------------------------------------------------------------------------------
   void Start()
   {
      if (proTexTexture != null)
      {
         var meshRenderer = gameObject.GetComponent<MeshRenderer>();
         var material = (meshRenderer != null) ? meshRenderer.material : null;
         if (material != null)
         {
            UpdateMaterial(material, runtimeTextureSize);
         }
         else
         {
            Debug.LogError("No material detected. GameObject [" + gameObject.name + "]");
          }
      }
   }

   //-------------------------------------------------------------------------------------------------------------------
   public void SetEditorPreviewTextures()
   {
      if (proTexTexture != null)
      {
         var meshRenderer = gameObject.GetComponent<MeshRenderer>();
         if ((meshRenderer == null) || (meshRenderer.sharedMaterial == null))
         {
            Debug.LogError("No material detected. GameObject [" + gameObject.name + "]");
         }
         else
         {
            var material = new Material(meshRenderer.sharedMaterial);
            meshRenderer.sharedMaterial = material;

            UpdateMaterial(material, EditorPreviewTextureSize);
         }
      }
   }

   //-------------------------------------------------------------------------------------------------------------------
   private bool IsDefaultMaterial(Material material)
   {
      return
         material.HasProperty("_MainTex") &&
         material.HasProperty("_BumpMap") &&
         material.HasProperty("_ParallaxMap") &&
         material.HasProperty("_MetallicGlossMap") &&
         material.HasProperty("_OcclusionMap") &&
         material.HasProperty("_EmissionMap");
      }

   //-------------------------------------------------------------------------------------------------------------------
   private bool IsHighDefinitionRenderPipelineMaterial(Material material)
```

```
    {
        return
            material.HasProperty("_BaseColorMap") &&
            material.HasProperty("_NormalMap") &&
            material.HasProperty("_HeightMap") &&
            material.HasProperty("_MaskMap") &&
            material.HasProperty("_EmissiveColorMap");
    }

    //-----------------------------------------------------------------------------------------------------------------
    private bool IsLightweightRenderPipelineMaterial(Material material)
    {
        return
            material.HasProperty("_MainTex") &&
            material.HasProperty("_BumpMap") &&
            material.HasProperty("_MetallicGlossMap") &&
            material.HasProperty("_OcclusionMap") &&
            material.HasProperty("_EmissionMap");
    }

    //-----------------------------------------------------------------------------------------------------------------
    private void UpdateMaterial(Material material, int textureSize)
    {
        if (IsDefaultMaterial(material))
        {
            UpdateDefaultMaterial(material, textureSize);
        }
        else if (IsHighDefinitionRenderPipelineMaterial(material))
        {
            UpdateHighDefinitionRenderPipelineMaterial(material, textureSize);
        }
        else if (IsLightweightRenderPipelineMaterial(material))
        {
            UpdateLightweightRenderPipelineMaterial(material, textureSize);
        }
        else
        {
            Debug.LogError("Shader [" + material.shader.name + "] not supported. Game object [" +
gameObject.name + "]");
        }
    }

    //-----------------------------------------------------------------------------------------------------------------
    private void UpdateDefaultMaterial(Material material, int textureSize)
    {
        SetTexture(material, TextureType.Color, textureSize, "_MainTex", "", "_Color");
        SetTexture(material, TextureType.Normal, textureSize, "_BumpMap", "_NORMALMAP", "");
        SetTexture(material, TextureType.Height, textureSize, "_ParallaxMap", "_PARALLAXMAP", "");
        SetTexture(material, TextureType.Metallic, textureSize, "_MetallicGlossMap", "_METALLICGLOSSMAP",
"");
        SetTexture(material, TextureType.Occlusion, textureSize, "_OcclusionMap", "", "");
        SetTexture(material, TextureType.Emission, textureSize, "_EmissionMap", "_EMISSION",
"_EmissionColor");
    }

    //-----------------------------------------------------------------------------------------------------------------
    private void UpdateHighDefinitionRenderPipelineMaterial(Material material, int textureSize)
    {
        SetTexture(material, TextureType.Color, textureSize, "_BaseColorMap", "", "_BaseColor");
        SetTexture(material, TextureType.Normal, textureSize, "_NormalMap", "_NORMALMAP", "");
        SetTexture(material, TextureType.Height, textureSize, "_HeightMap", "_HEIGHTMAP", "");
        SetTexture(material, TextureType.Metallic, textureSize, "_MaskMap", "_MASKMAP", "");
        SetTexture(material, TextureType.Emission, textureSize,"_EmissiveColorMap",
"_EMISSIVE_COLOR_MAP", "_EmissiveColor");
    }

    //-----------------------------------------------------------------------------------------------------------------
    private void UpdateLightweightRenderPipelineMaterial(Material material, int textureSize)
    {
        SetTexture(material, TextureType.Color, textureSize, "_MainTex", "", "_Color");
```

```
        SetTexture(material, TextureType.Normal, textureSize, "_BumpMap", "_NORMALMAP", "");
        SetTexture(material, TextureType.Metallic, textureSize, "_MetallicGlossMap",
"_METALLICSPECGLOSSMAP", "");
        SetTexture(material, TextureType.Occlusion, textureSize, "_OcclusionMap", "_OCCLUSIONMAP", "");
        SetTexture(material, TextureType.Emission, textureSize, "_EmissionMap", "_EMISSION",
"_EmissionColor");
    }

    //-----------------------------------------------------------------------------------------------------------
    private void SetTexture(
        Material material,
        TextureType textureType,
        int textureSize,
        string textureName,
        string keyword,
        string colorName)
    {
        if (keyword.Length > 0)
        {
            material.EnableKeyword(keyword);
        }

        bool hasTexture = proTexTexture.HasTexture(textureType);

        if (hasTexture)
        {
            material.SetTexture(textureName, proTexTexture.GenerateTexture(textureSize, textureSize,
textureType));
        }
        else
        {
            material.SetTexture(textureName, null);
        }

        if (colorName.Length > 0)
        {
            material.SetColor(colorName, hasTexture ? Color.white : Color.black);
        }
    }
}
}
```