bake-wrangler-docs

Oct 05, 2021

Contents

1 Install & Update **3**

2 Quick Start Guide **5**

3 Bake Passes **7**

4 Preferences **9**

5 Nodes **13**
- 5.1 Resolutions . . . 13
- 5.2 Objects . . . 14
- 5.3 Mesh . . . 15
- 5.4 Pass . . . 17
- 5.5 Output Image Path . . . 19
- 5.6 Batch Bake . . . 21

6 Examples **23**
- 6.1 Bake Pass vs Image . . . 23
- 6.2 Masking . . . 25

7 Bug Reports **29**

Bake Wrangler is an add-on for blender that provides a *nodes based* interface for texture and material baking.

Current Features:

- Easy to use node system
- Background batch baking
- Supports all the blender internal bake passes and multires
- Exposes separate Color, R/G/B and Value channels of a pass
- Allows assigning pass output to individual output channels (R/G/B/A)
- Masking feature allows layering of bakes and combining with external processes
- Objects are isolated within a pass preventing unwanted interference
- All standard image formats, color-spaces and bit depths are available
- Full set of 23 PBR passes (Albedo, Metallic, Smoothness, etc)
- Additional 11 passes not found in blender (Bevels, Cavity, Curves, Height, etc)
- Automatic cage generation options when baking down from other objects
- Material override system to bake different materials without changing your objects
- Modifier baking to unmodified base objects
- Still more planned...

The Future: There are three main areas of development planned for future releases:

- **User Interface (Current Focus)**
  - Productivity features for large/complex bakes to allow grouping of values so they can be modified from one location and a similar system of tokens for string replacement. This will be especially useful for recipe re-use in other projects to quickly set the project specific values.
  - Nodes and systems to better support specific work-flows that may not fit optimally within the current systems. Requires more user feedback.
- **Image Manipulation** - While not currently implemented, it would be possible to combine passes using a maths function. Similarly some post processing could be done on the images. There are some performance limitations imposed by Python and I'm not entirely sure how useful it would be. But it's certainly possible if users have a strong case for something.

Come visit blenderartists.org/t/bake-wrangler-node-based-baking-tool-set/ to get involved and make suggestions :)

Bake Wrangler is available from:

Gumroad:	gum.co/bake-wrangler
Blender Market:	blendermarket.com/products/bake-wrangler

to throw some coins into the *tip jar* and help fund more improvements. You can leave a comment with each donation telling me what **new feature** you would most like added.

CHAPTER 1

Install & Update

There are no special steps required to install the add-on. Simply open your blender preferences, navigate to the *Add-ons* tab (left column) and click the '*Install...*' button near the top right. In the '*File View*' that opens navigate to the location of the Bake Wrangler zip file and click '*Install Add-on*' with it selected or double click on the zip file.

Once installed you need to check the box on the add-on to enable it.

Updating:

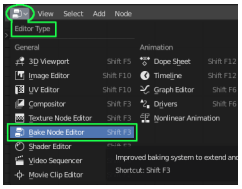
When updating follow the same procedure (making sure the '*Overwrite*' box is checked in the '*File Viewer*' when selecting the new version). Sometimes changes are made the require blender to be restarted, though I try to avoid this, it is a good idea to do so after updating if things don't seem to be working right.

If an update changes the node tree in some way, out of date trees will be updated when they are loaded. If a tree is already open when you update or load a .blend it won't be updated until something causes it to be re-read. Selecting it from the tree list or changing the tree both cause an update.

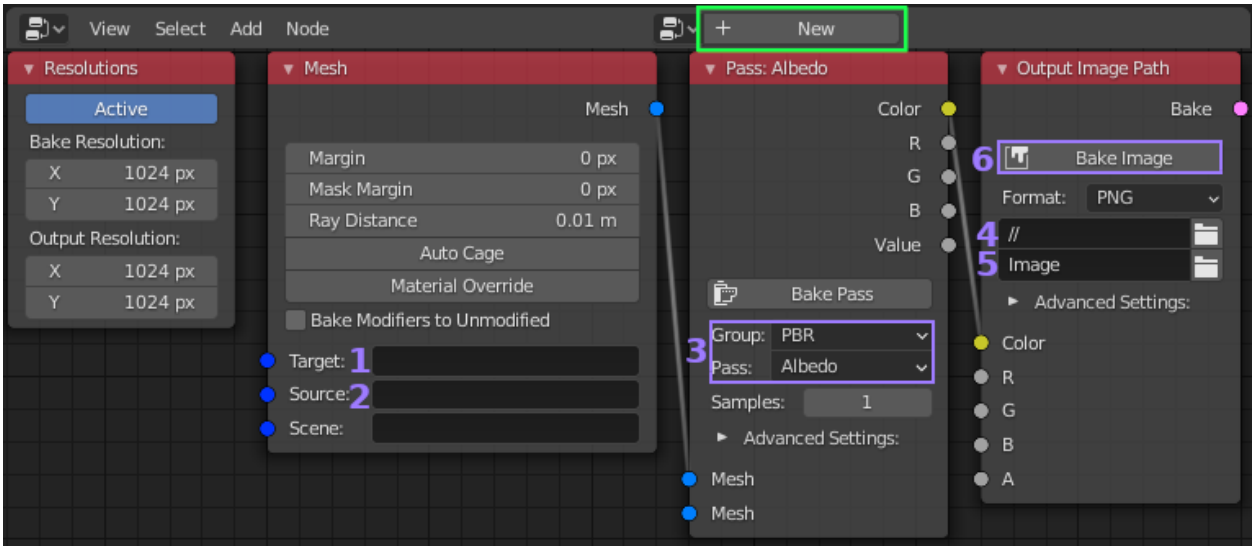
CHAPTER 2

Quick Start Guide

Once the add-on is installed and enabled a new '*Editor Type*' will become available (the drop down list in the top left title bar of every area) called '*Bake Node Editor*':



Switch to the *Bake Node Editor* and click '*New*' to create a tree. Your tree will start with the most simple bake node configuration loaded. This guide will cover this simple configuration, for details on each node take a look at the *Nodes Section* and for more complicated set ups hop on over to the *Examples*.



1. **Target:** is the object you want to **bake to**. It must be a '*MESH*' type object with a UV Map. A list of objects can also be connected here which would cause each object in the list to be baked with the same settings. This

field must be filled.

2. **Source:** is the object you want to **bake from**. The field is optional and if left empty the data will be taken from the target object. Most renderable objects can be used as the target and a list of objects can be connected. If multiple objects are selected then **all** of them will be applied to each target (if there are multiple targets).
3. **Group & Pass:** selects the type of data you want to bake. Some passes will display additional options below them when selected. In the example picture the '*Albedo*' pass in the '*PBR*' group has no additional options.
4. **Image Path:** selects where the baked image will be saved. Relative locations are supported and will be expanded on bake (*Eg. Using // for the current path*).
5. **Image Name:** the name of the baked image. File extensions will be added automatically based on the file format unless you specify your own extension and that extension is not a recognized format. In that case your unrecognized extension will be used.
6. **Bake Image Button:** will first validate the settings and then begin a background process to create the image. With default settings a window the same size as the bake editor will pop up, displaying any errors and progress information.

Most of the other settings a fairly self explanatory and can often be left at their defaults. More node types and details of each setting are covered in the *Nodes Section*

CHAPTER 3

Bake Passes

Following is a list of currently supported bake passes and a short description of their function:

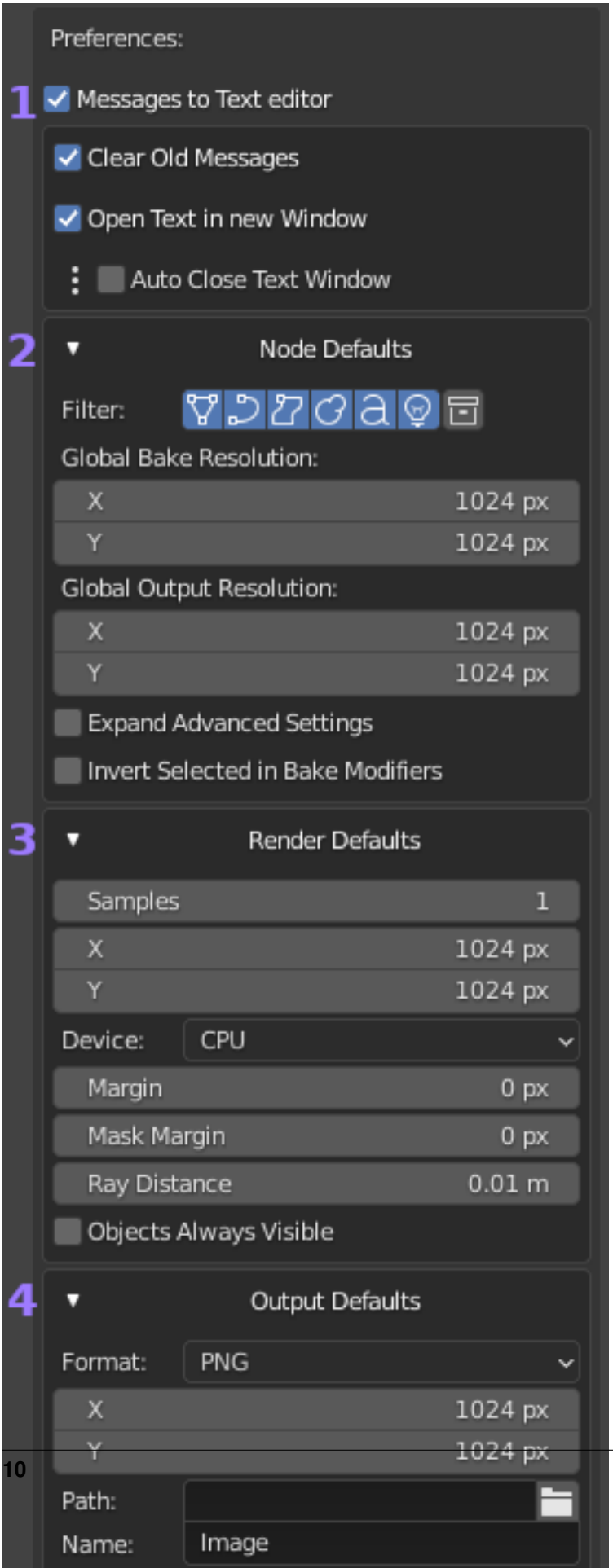
- **All Blender internal passes:** The standard passes and all their options are supported including the Multiresolution passes: Combined, AO, Shadow, Normal, UV, Roughness, Smoothness, Emit, Environment, Diffuse, Glossy, Transmission, Multiresolution Normals & Displacement.
- **Full set of PBR passes:** These passes all collect specific object information without influence from other inputs or lighting. The current list is: Albedo, Subsurface, Subsurface Radius, Subsurface Color, Metallic, Specular, Specular Tint, Roughness, Smoothness, Anisotropic, Anisotropic Rotation, Sheen, Sheen Tint, Clear Coat, Clear Coat Roughness, Clear Coat Normals, Transmission IOR, Transmission, Transmission Roughness, Emit, Alpha, Texture Normals (no geometry influence) and Object Normals (no texture influence).
- **Bevel Mask:** Generates a map with beveled areas masked in white.
- **Bevel Normals:** Creates a normal map with just the beveled areas (can be used with above mask to mix with other maps).
- **Cavity & Edges:** Used to get a grey scale map of cavities or edge locations.
- **Curvature:** Produces a grey scale map of surface angles with options for how values should be mapped.
- **Height Map:** Another grey scale map giving the distance between the target surface and other surfaces around it.
- **Island ID:** Outputs each UV island in a different flat color to differentiate them.
- **Material ID:** The areas covered by each material are output with a different color assigned to each material.
- **Object Color:** Simply outputs each object using the color assigned to it in the viewport settings.
- **World Position:** A tri-color vector map giving the location of the object from the origin.
- **Thickness:** Gets the distance between internal faces, representing how thick an area of the object is.
- **Vertex Colors:** Outputs whatever data is in the selected vertex colors block as a texture.

There is always the possibility of more passes being added by user request or additional features being adding to existing ones.

CHAPTER 4

Preferences

Below are the current available preferences and their default settings:


Preferences:
1
Messages to Text editor
Clear Old Messages
Open Text in new Window
Auto Close Text Window
2
Node Defaults
Filter:
Global Bake Resolution:
X 1024 px
Y 1024 px
Global Output Resolution:
X 1024 px
Y 1024 px
Expand Advanced Settings
Invert Selected in Bake Modifiers
3
Render Defaults
Samples 1
X 1024 px
Y 1024 px
Device: CPU
Margin 0 px
Mask Margin 0 px
Ray Distance 0.01 m
Objects Always Visible
4
Output Defaults
Format: PNG
X 1024 px
Y 1024 px
Path:
Name: Image

1. **Message Settings:** Deal with where and how information is reported back to the user about progress and errors.
    - *Messages to Text Editor*: Causes messages and errors produced by Bake Wrangler to be written to a text file named '*BakeWrangler*' in the current project. When disabled these messages would only be visible in the console. This also enabled the three options below.
    - *Clear Old Messages*: Clears the text file prior to each bake, so that messages are only relevant to the current/last process.
    - *Open Text in new Window*: Will open a new window when a bake process starts, displaying the text file (which is continually updated). The size and location of the window will match the Bake Node Editor from which the bake was started. The intention is to allow you to create your own work space with the text file already open and so disable this pop-up.
    - *Auto Close Text Window*: Will close the above pop-up automatically when a bake completes successfully.
2. **Node Defaults:** Sets the default display state of some nodes. Here the initial filter settings of the *Objects* node can be set along with the default bake and output resolutions. You can also have nodes with collapsed advanced options start with them expanded instead. (Applies only to new nodes)
    - *Invert Selected in Bake Modifiers*: Causes the modifier selection method to be inverted when using this option in the *Mesh* node. Viewport hidden modifiers will be baked down instead of shown modifiers.
3. **Render Defaults:** Has settings related to the *Mesh* and *Pass* nodes, allowing you to configure their defaults. The resolutions set here are used when the global setting is overridden or if no global node is enabled in the scene. (Applies only to new nodes)
    - *Objects Always Visible*: Causes Bake Wrangler to ignore the visibility settings of an object in Blender. When enabled all objects selected as part of a bake will be made visible.
4. **Output Defaults:** Has settings related to the *Output Image Path* node, allowing you to configure the defaults for newly created nodes. Again the resolutions set here only apply when the global setting is overridden or no active global node is found.
    - *Create Paths*: Causes Bake Wrangler to attempt to create the output path if it doesn't exist.
    - *Save each Pass*: Causes the output to be saved after each bake pass instead of after all contributing passes. This will reduce performance in some cases (especially with Alpha being written every time), but will preserve pass data in the event of a later failure or memory shortage. Use it if some pass data seems to be getting lost or if you have very long pass bake times so as not to lose progress.
5. **Debug:** Adds more detailed messages to each process and if a bake fails with an error condition a complete process log will be opened in a new window. **Please post this log when reporting a bug.**

CHAPTER 5

Nodes

Detailed information on each node in the order they appear in the '*Add*' menu *(Note: All node settings have mouse over tool-tips with additional info)*:

Resolutions	Quickly sets global resolutions
Objects	Container for groups of objects (used for input)
Mesh	Settings of and relationship between inputs
Pass	Bake pass configuration
Output Image Path	Output file location and format
Batch Bake	Groups outputs into a single process

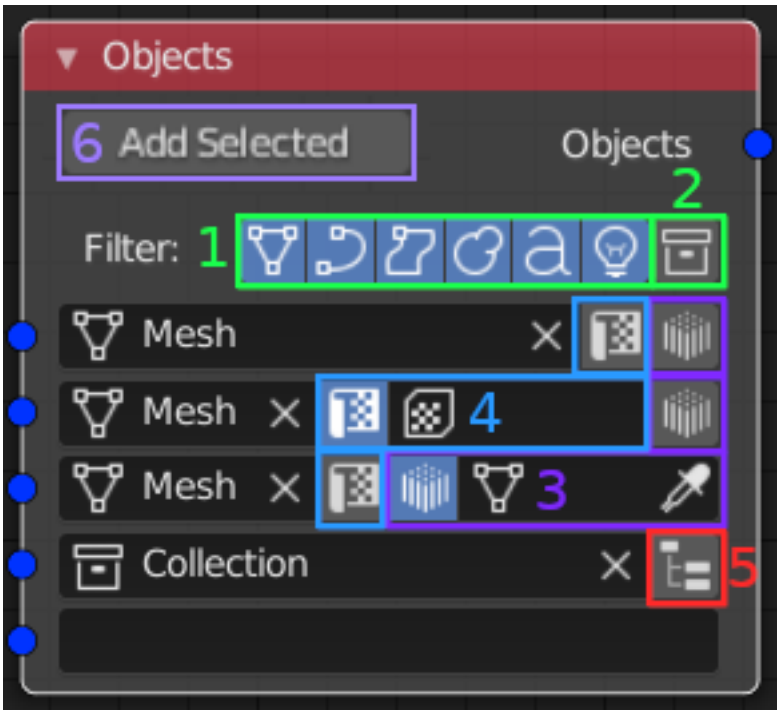
5.1 Resolutions

The *Resolutions* node allows you to globally set your bake and output resolutions rather than set each pass individually. Only one such node can be set *Active* at a time, but more than one can be present in a tree. Individual passes and outputs can have a local overriding value set if desired to ignore the global value. The local settings are also used if no global node is present or active.

5.2 Objects

The *Objects* nodes role is to contain lists of objects and collections which are used by the *Mesh* nodes input sockets. *Objects* nodes can be chained together and the node itself will continue to expand as objects are added, allowing for any number and combination of objects.

Some objects have additional options that can be set once they are added. This is where **Cages** and **UV Maps** are configured.



1. **Filter:** Selects what object types will be shown in the search boxes and starts with all types enabled.
2. **Collections:** Toggles between selecting objects or collections. The node can contain a mixture of both, but can only search for one or the other.

Note: If an object is contained in both a selected collection and specifically listed, the specific listing will take preference. It is also possible to list the same object multiple times with different settings (UV Map and/or Cage), in which case it will be evaluated multiple times.

3. **Cage:** When the selected object can support a cage (*'MESH' type objects*), this button will appear at the end of its row. Toggling the button will enable using a cage for that object when baking (*if using a cage makes sense for the bake*). The cage object must have the same number of vertex as the original after all modifiers are taken into account. On click it will automatically search for and select an object that has the same name as the original plus a spacing character follow by 'cage' (*Eg. cube.cage*).
4. **UV Map:** Allows selection of a specific *UV Map* to use for the object. This option only appears when the object is able to support multiple maps (*Currently that is 'MESH' type objects*). When left blank or disabled the currently active map is used. Removing or renaming the referenced map on the object will break the reference and must be manually updated before baking can start.
5. **Recursive Selection:** When the selected item is a collection this option will appear. Enabling it will cause any and all sub collections of the primary selection to also be included.
6. **Add Selected:** Clicking this button will add all currently selected objects to the node, respecting the current filter. Duplicate items will not be added unless SHIFT is held while clicking. It is not possible to add collections in this way.

5.3 Mesh

The *Mesh* node is responsible for mesh specific settings and the relationship between objects. It can take *Objects* nodes as any of its inputs, and outputs to a *Pass* node. Only objects that are linked to this node will be included in a bake pass and will be completely isolated from all other objects in the .blend file.

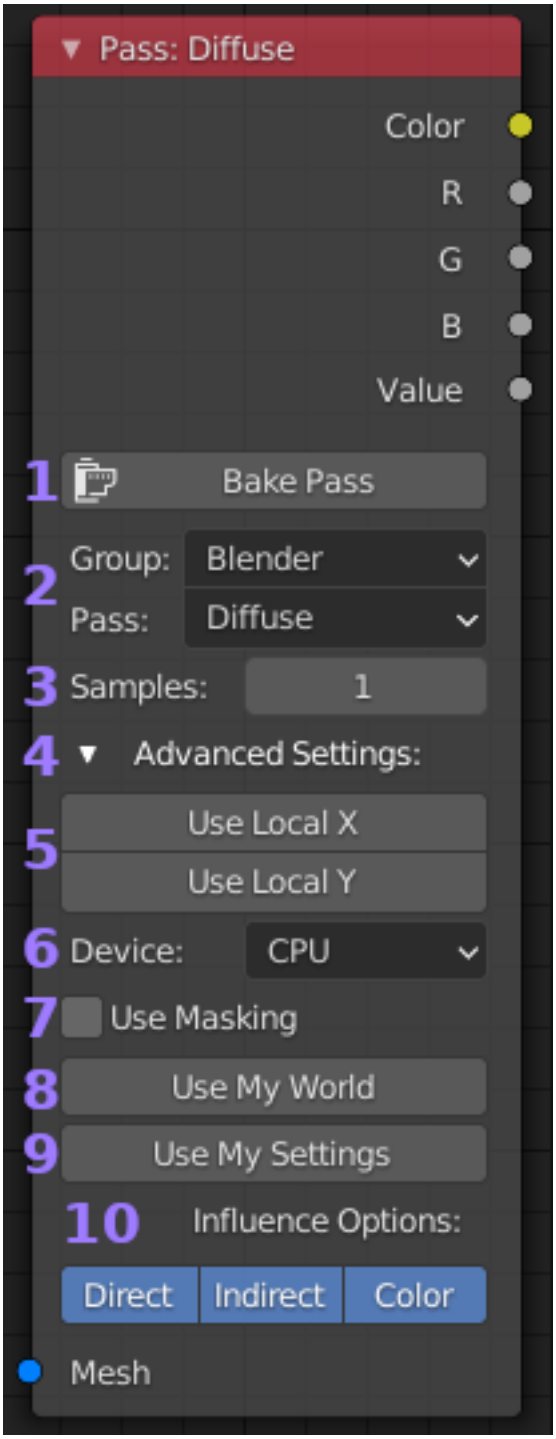
1. **Margin:** Number of pixels to extend out around the edges of baked UV islands. This is used to prevent bleeding of background colors into texture island edges when a coordinate lies on a pixel boundary or when a texture is re-sampled at a different resolution.
2. **Mask Margin:** If masking is enabled this sets how many pixels the mask should extend beyond the UV islands margin. Normally this can be set to zero, but is provided in case a mask is cutting off edges.
3. **Ray Distance:** Sets the distance in blender units that baking rays should be cast from above the objects surface. This only applies when baking from one or more *Source* objects. It needs to be far enough away to hit any details that extend above the surface of the *Target*. When finer control is required a *cage* object should be specified on the *Target* object.
4. **Auto Cage:** This will create a cage for every target object that doesn't have it's own cage specified. When enabled an expansion and smoothing angle value will appear. The expansion value determines how far out (or in with a negative value) the cage will be created from the original object. The smoothing angle is the angle below which normals will be smoothed. Complicated objects may not result in a good cage and small values for expansion should be used.
5. **Material Override:** This will replace all materials on your objects with the selected material during baking. Use this to either bake alternative materials without modifying your scene or to bake your own material based passes (such as OSL shaders).
6. **Bake Modifiers to Unmodified:** This can be used to bake the effect of a modifier onto the base object. Mainly this is for convenience and achieves the same result as duplicating an object, stripping the modifiers you wanted to bake and then baking the original object onto the stripped version. This can be useful to bake a bevel modifier for example. It is also possible to exclude modifiers by disabling their viewport visibility (this behavior can be inverted in the preferences). If for example you had a mirror modifier followed by a bevel you would hide the mirror (it will now get applied to both objects before baking) and leave the bevel visible so that only the

difference created by the bevel would be baked.

7. **Target:** May be a single 'MESH' type object or taken from a *Objects* node. When a list of objects is used, only 'MESH' type objects will be considered with other types ignored. Each *Target* will be baked in its own pass but shares the *Mesh* nodes settings. At least one valid object must be selected for a bake to valid.
8. **Source:** Optional field that can take a single object of most types or a list of objects from a *Objects* node. All of *Source* objects will have their surface data projected onto to the *Target(s)* (if sensible for the selected bake pass). Normally a ray distance greater than zero is needed to capture everything correctly.
9. **Scene:** Optional field that can take a single collection or a list of objects from a *Objects* node. This is the only input that will consider lights as valid objects. This input is used to set up lighting, shadow casting objects and anything that indirectly influences the pass but isn't directly mapped to the *Target*. Generally unless you are trying to capture lighting information this input is not needed.

5.4 Pass

The *Pass* node is where the type of bake (*pass*) is selected and all rendering options are set. It can take any number of *Mesh* nodes as input (adding inputs will cause it to keep expanding to accept more). The outputs expose the combined *Color* and *Value* of pixels as well as the individual *Red*, *Green* and *Blue* values. Outputs can be connected in any combination to any number of *Output Image Path* nodes.



1. **Bake Pass Button:** Will validate settings and then bake this nodes pass. Any linked outputs will be updated, but *only* the specific linked channels. See the *Bake Pass vs Image* example for how this works.

2. **Group & Pass:** Drop down list to select bake *pass*. Some passes may have additional options which will appear below when selected. Tool-tips are used to explain their functions. Additionally some passes require a *Principled BDSF* based material on the objects, the tool-tip on the *pass* should list any requirements.

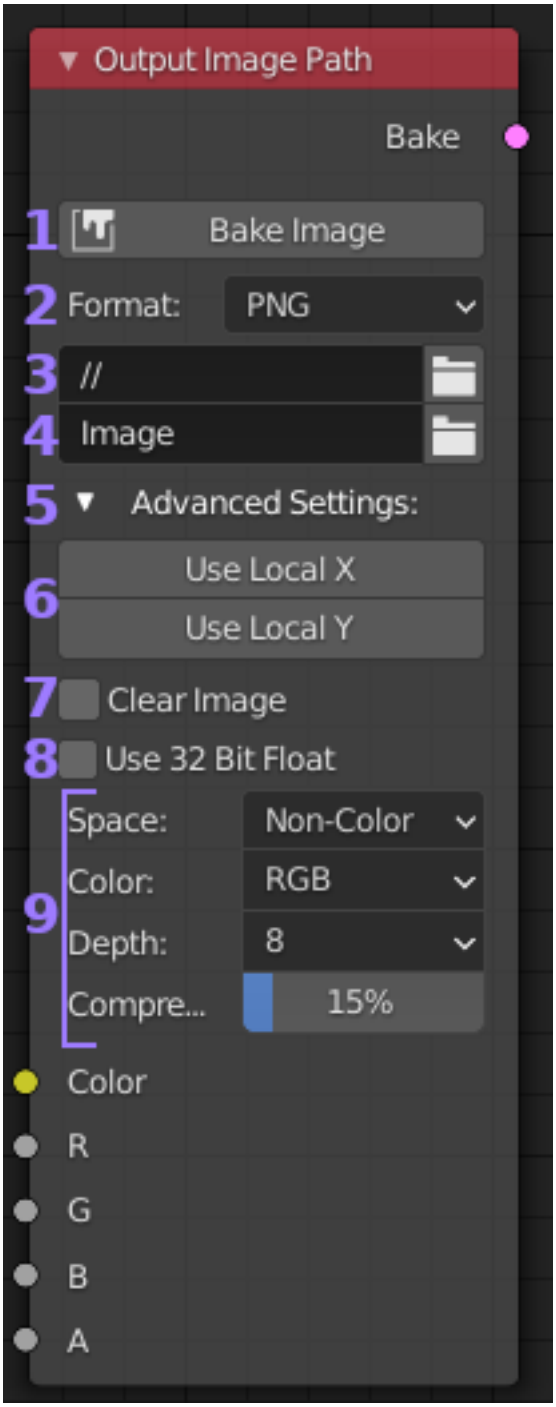
3. **Samples:** This is the number of samples taken for each pixel in the bake. For most *passes* (any PBR map, normals, etc) **one** sample is sufficient. When lighting information is needed (eg. AO *pass*) more samples will be

needed to produce a good result. If in doubt start with one sample and increase if the result isn't good enough.

4. **Advanced Settings:** Collapses or expands the more advanced or less used settings to reduce clutter and node footprint.
5. **Local X and Y Resolution:** These are used only to override any *Resolutions* nodes or when no global values are set or active.
6. **Device:** Simple choice between *CPU* or *GPU* as rendering device. When the node is places this will default to your scenes currently selected device. While not usually an issue when baking, if a render exceeds the available memory of your *GPU* it will fail. This isn't the case for *CPU* making it the *safe* option.
7. **Use Masking:** Enabling this option will create a second image of the same size, where white pixels (*1s*) map to the UV islands of the bake and black pixels (*0s*) map to pixels that aren't part of an island. This map will automatically be used when writing to the final output to leave unmasked pixels unchanged. While many bakes don't require this, it allows for much greater flexibility in layering *passes* and objects into a single final image. The time taken to generate the mask is negligible, take a look at the *Masking* for an example.
8. **Use My World:** If you want to use lighting information from your scenes world in the *pass*, you need to enable this option and select the '*World*' you want to use (if left blank, but enabled the currently active scenes world is used). By default Bake Wrangler uses a plain 100% white world background.
9. **Use My Settings:** Enabling this causes Bake Wranglers default rendering settings to be replaced with the settings from the '*Scene*' you select (if left blank, but enabled the currently active scenes settings are used). The default render settings used by Bake Wrangler are optimized to quickly render maps without lighting information and complex data like hairs and caustics. If you need to bake lights, hairs or rays that are changed by passing through objects (glass, fog, etc) then you will need to use this setting with appropriate *Cycles* values set in your scene. For any PBR maps and most data maps this should be disabled for best performance.
10. **Additional Pass Settings:** If the selected pass has additional options they are displayed here. Use the tool-tips to see what they do, or if in doubt leave at default.

5.5 Output Image Path

The *Output Image Path* node specifies where an image should be saved, the name to use and all the file format information. Currently it is the only output path for a bake. It takes color or value data from one or more *Pass* nodes as input. Each input is written to the final image in the order they appear, such that if you used the *Color* input and the *R* input (assuming none of the input data had *Masks* assigned) the *R* input would override every *Red* value of the *Color* input. Writing to the *A* (Alpha) channel requires an image format that supports alpha to be chosen or it will be ignored. Writing alpha also takes slightly longer and is performed in a separate pass once colors are written. An output is provided for linking to one or more *Batch Bake* nodes if desired.



1. **Bake Image Button:** After validating settings this will cause all connected *passes* to be performed, mixed together and saved to the output file. This is the primary method for starting a non-batch bake. For the differences with the similar *Bake Pass* button take a look at the *Bake Pass vs Image* example.

2. **Format:** Drop down list of supported image formats. This will default to the output format selected for your current scene when placing the node. The options in section *[7]* will change depending on the format chosen.

3. **Image Path:** Simply used to select the path where the image will be saved. Relative paths may be used (*eg. // to refer to the path the .blend file is in*).

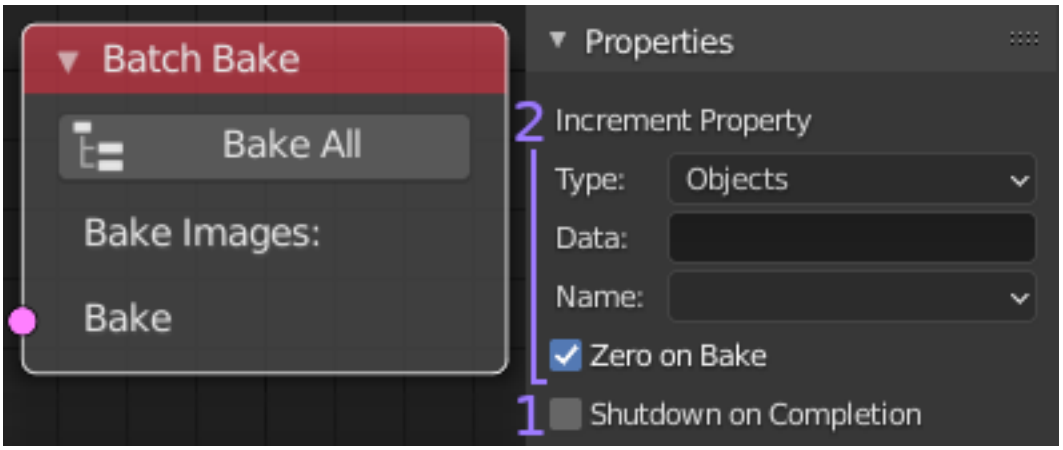
4. **Image Name:** The name of the image with or without the extension. The extension will be added for the

currently selected file format unless you have given the file an extension that isn't recognized (*eg. if format was set to 'JPEG' both 'img' and 'img.png' would be saved as 'img.jpg' but 'img.myext' would not be changed*).

5. **Advanced Settings:** Collapses or expands the more advanced or less used settings to reduce clutter and node footprint.
6. **Local X and Y Resolution:** These are used only to override any *Resolutions* nodes or when no global values are set or active.
7. **Clear Image:** When enabled and if the target image already exists, it will clear the image to black (and transparent if supported by image settings) before writing bake data.
8. **Use 32 Bit Float:** Normally blender renders images using 8bits per channel per pixel (24bpp or 32bpp with Alpha). If you want to save your output in a format with more data per pixel than that you need to enable this option. All contributing *passes* will then use 92bpp (128bpp with Alpha). This uses up 4x more memory than standard, but can be useful for *data* maps (eg. normals) to allow more variations in color (and hence a more accurate representation of the data). For plain color maps it generally doesn't provide any advantages. You must also use an image format and bit depth with higher than 8bpp or the extra data will be lost and colors may appear different than expected as they are remapped to a lower bit depth.
9. **Format Options:** This group of options are specific to the chosen image format and will change accordingly. Almost all formats support '*Color Space*', if you are unsure what to pick use '*sRGB*' for color information and '*Non-Color*' for data maps like normals. For any other settings either use the defaults or check the tool-tip for more information if you are unsure.

5.6 Batch Bake

The *Batch Bake* node simple takes any number of *Output Image Path* nodes as input (it will continue to expand as sockets are connected). Pressing the '*Bake All*' button will validate and process all the attached images, generating them in the order they are connected. It also has some additional properties in the side bar explained below.



1. **Shutdown on Completion:** Will attempt to power off the system once the batch has completed. Useful when running very long processes unattended. It should work fine on Windows systems, but MacOS and Linux require the user to be able to run 'sudo shutdown' without entering a password.
2. **Increment Propery:** This is a somewhat advanced feature. Essentially you proide a custom number property which Bake Wrangler will increase by 1 each time it completes one of the batch inputs. The check box will set it to zero at the start if enabled. This would let you change things in the scene between each batch input using drivers connected to the supplied property. This allows for a large degree of automation.

CHAPTER 6

Examples

Bake Wrangler is capable of many complex configurations and provides two paths to generate output. The following examples better illustrate how some features work and give some ideas of what is possible:

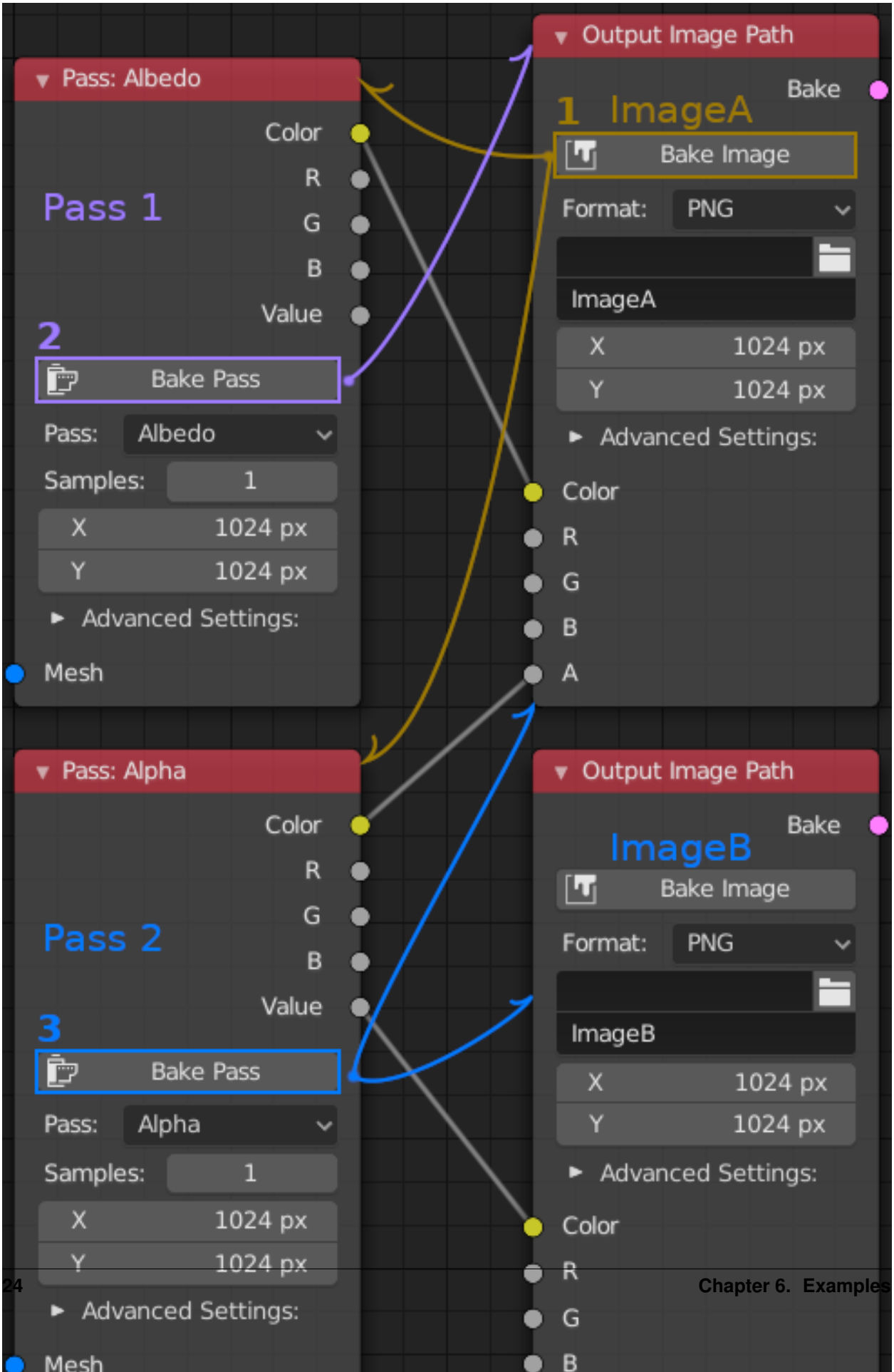
Bake Pass vs Image	Details '*Bake Pass*' vs '*Bake Image*' buttons
Masking	Using the '*Masking*' feature

6.1 Bake Pass vs Image

You may have noticed there are two ways (other than the Batch node) to begin a bake. Both the *Pass* and *Output Image Path* nodes have a button to start baking.

In a simple tree where a *Pass* node connects to only one *Output Image Path* node there is essentially no difference. However if a *Pass* node connects to more than one *Output Image Path* node or a *Output Image Path* has more than one *Pass* connected things are different.

This will be illistrated in the below example. But the rule is that the '*Bake Pass*' button (found on the *Pass* node) will generate only the *channels* of the outputs they are connected to, not the whole image. While the '*Bake Image*' button (found on the *Output Image Path* node) will generate only that *image*, even if the passes used would normally contribute to multiple images.


Output Image Path
Bake
1 ImageA
Bake Image
Format: PNG
ImageA
X 1024 px
Y 1024 px
Advanced Settings:
Color
R
G
B
A
Pass: Albedo
Color
R
G
B
Value
Pass 1
2
Bake Pass
Pass: Albedo
Samples: 1
X 1024 px
Y 1024 px
Advanced Settings:
Mesh
Pass: Alpha
Color
R
G
B
Value
Pass 2
3
Bake Pass
Pass: Alpha
Samples: 1
X 1024 px
Y 1024 px
Advanced Settings:
Mesh
Output Image Path
Bake
ImageB
Bake Image
Format: PNG
ImageB
X 1024 px
Y 1024 px
Advanced Settings:
Color
R
G
B

1. Pressing the '*Bake Image*' button on the *ImageA* (Brown) node would cause *Pass1* and *Pass2* to both be processed. However *ImageB* would not be changed even though *Pass2* connects to it.
2. Pressing '*Bake Pass*' on the *Pass1* (Purple) node would generate **only** the *Color* channel of *ImageA*, leaving everything else unchanged.
3. While prssing '*Bake Pass*' on the *Pass2* (Blue) node would generate **only** the *Alpha* channel of *ImageA* (leaving *Color* unchanged). As it is the only input to *ImageB*, it would also completely generate *ImageB*.

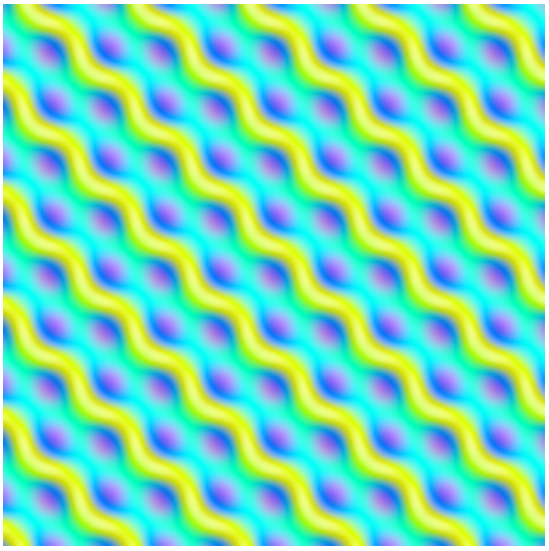
This can be very useful when you have data packed into different channels of an image and only want to update a specific subset of data. Saving doing unnecessary bakes. It also allows for generating a specific image, while leaving others alone.

6.2 Masking

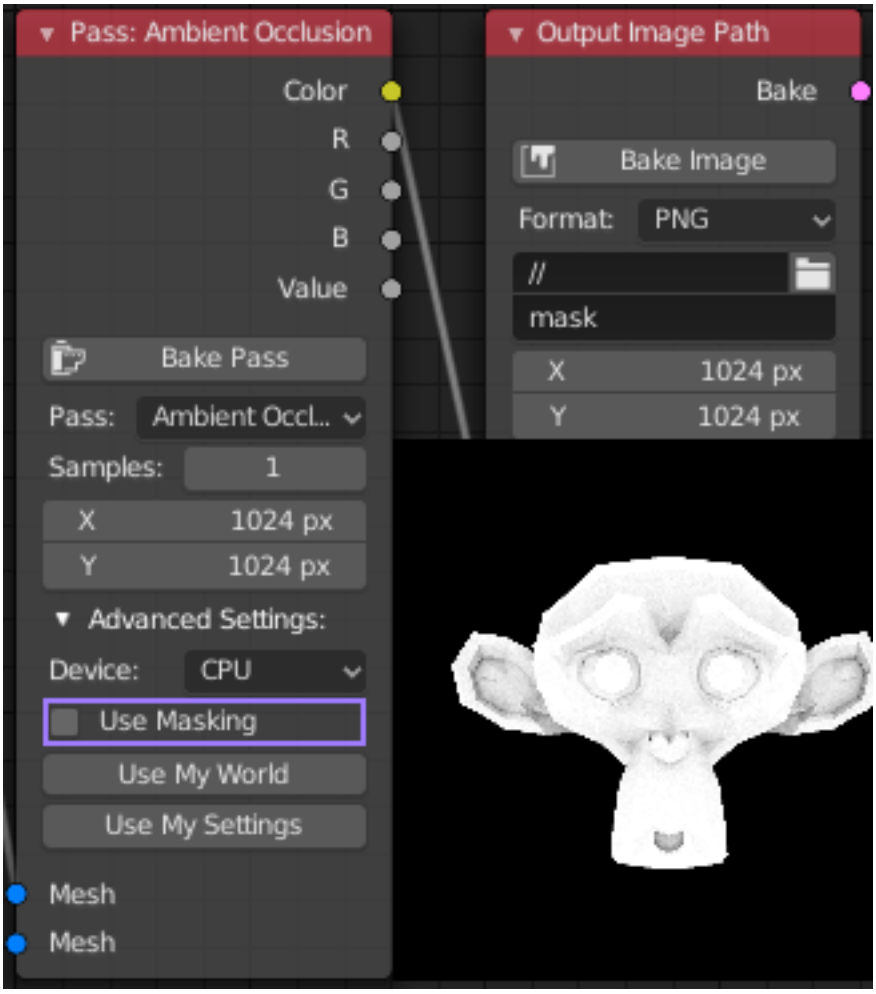
One of the advanced options on the *Pass* node is to use *Masking*. For most tasks there is no need to use this setting, but it allows for many configurations that would otherwise be impossible.

When enabled a simple black and white map of the UV space used by the bake *Pass* will be created and used to determine what pixels in the output get modified. This simple example will demonstrate what this looks like.

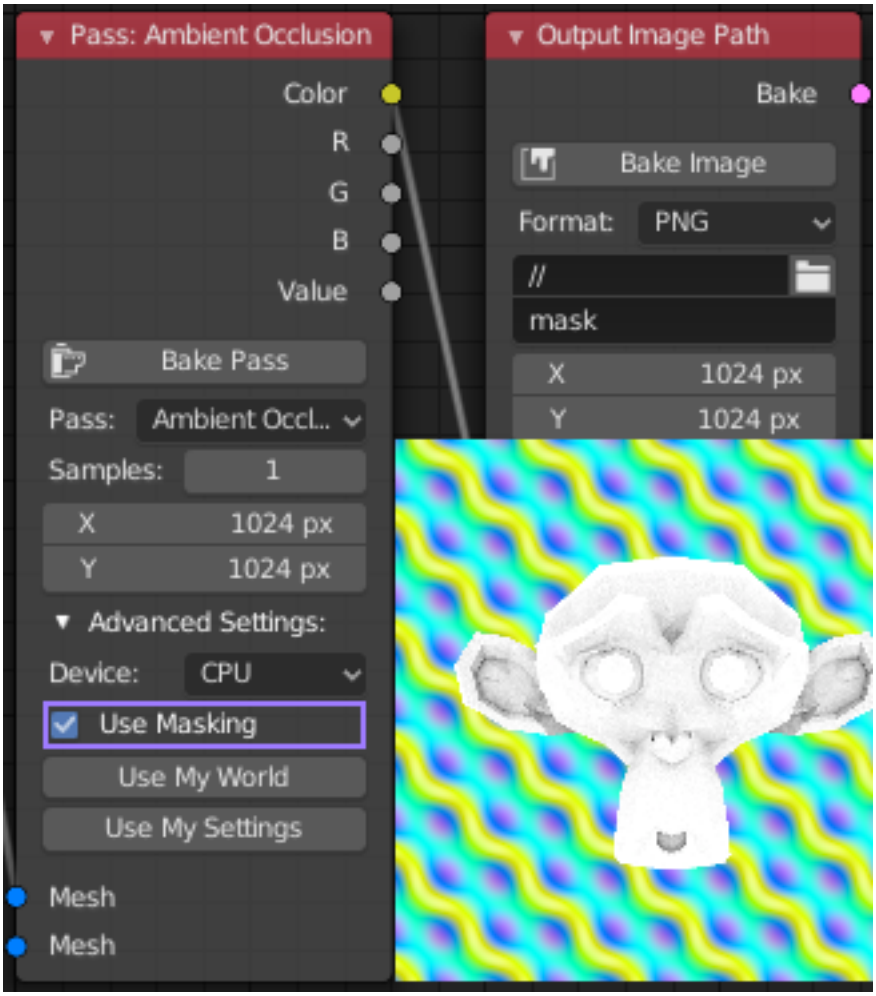
We will start with the following image and then bake the AO *Pass* of a Suzanne over the top of it with *Masking* turned off and then on:



With *Masking* disabled, the original image/channel is completely replaced:



However when *Masking* is enabled, only the pixels which contribute to the *Pass* are replaced, leaving the original unchanged in the rest of the image:



There are many ways in which this can be used, either with images created by external software that you want to add to. Or with layering of *passes* of different types or with different settings. But it is completely unnecessary when layering *passes* of the same type with the same settings.

CHAPTER 7

Bug Reports

Have you found a bug? *Please* report it here (blenderartists.org/t/bake-wrangler-node-based-baking-tool-set/) following these guidelines:

Always post the steps to reproduce the bug.

User interface bug: Along with describing what you did, post a screen shot if possible/relevent and any error messages produced in the console.

Bake process bug: Make sure to enable '*Debug*' in the *Preferences*. This should cause a complete log of the process to open in a new window when the process fails. Please post this text with your report. If a log doesn't open you may need to post your .blend file.

Bake output bug: If the results of a bake pass don't match what is expected, first verify your node tree set up and check the tool-tip for the pass you are using to see if it has any special requirements. If the result is still incorrect post your node tree along with the expected and actual outputs. I will likely need your .blend file also.

Thanks for helping to improve Bake Wrangler!