BW Tools

Release 2

Ben Wilson

May 21, 2022

CONTENTS

CHAPTER

ONE

HOW TO INSTALL

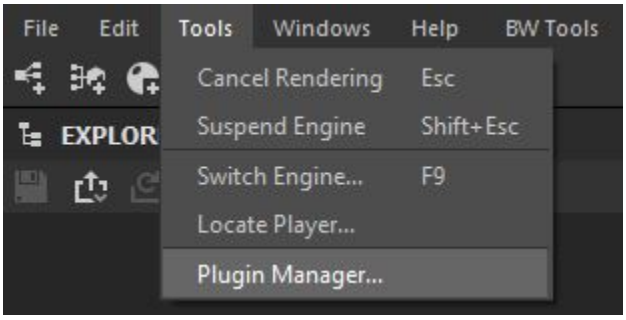
Have A Previous Version Installed?

You must first manually delete the previous version of bw_tools from your plugins folders.

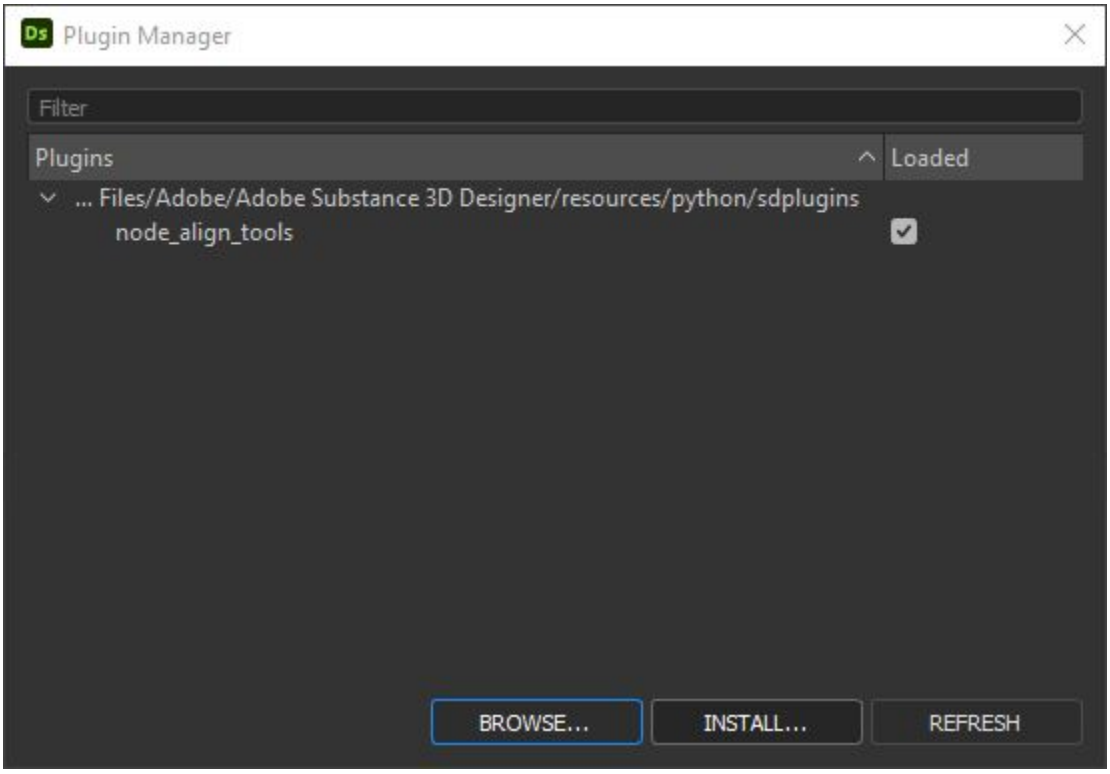
This is your documents folder by default `C:/Users/UserName/Documents/Adobe/Adobe Substance 3D Designer/python/sduserplugins`

Delete the previous bw_tools folder and restart Designer

To install, launch Substance Designer and to go Tools > Plugin Manager…



Click Install…



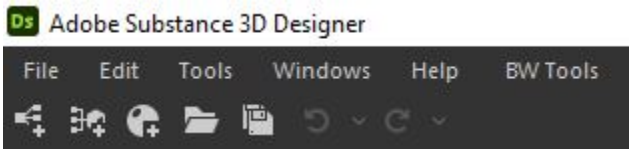
Select the downloaded .sdplugin file

CHAPTER

TWO

NOTABLE CHANGES FROM BWTOOLS 1.X

- Replaced the top toolbar icon with a menu bar to access the tools. This now lives next to the Help menu.



- Moved all graph related tools to the graph view toolbar. It can be toggled on and off by the BWT icon.

- Added a new module called *BW Framer*, which lets you reframe a selection quickly and provide default settings for the frame. No more manually adjusting frames.
- You can no longer run the tools on graphs types currently unsupported by the Designer API, which caused a crash. See *Unsupported Graph Types* for more info.
- Added more detailed tooltips.
- New documentation! You are reading it now.
- Code cleanup so it is a little more readable for this release.

2.1 BW Layout Tools Changes

- Considerable speed increase when running the layout tool.
- The mainline feature (which pushed nodes back) is now toggleable.
- Added multiple layout styles.
- Added a snap to grid option.
- Added option to run *BW Straighten Connection* after running layout.
- Several bug fixes in layout behavior. The default behavior may be slightly different in this release.
- See *BW Layout Graph* for full documentation.

2.2 BW Optimize Changes

- Removed features which are now handled natively by Designer.
- Fixed a bug which would incorrectly identify some nodes as duplicates.

2.3 BW PBR Reference Changes

- Can now be opened from the new menu bar.
- Removed custom color support in order to simplify the code for public release.

2.4 BW Straighten Connection Changes

- Now supports two different layout styles, see *BW Straighten Connection*.

CHAPTER

THREE

GENERAL GUIDE TO USING THE TOOLS

The design behind the tools is to aid in your day to day graph organisation and optimisation tasks. As such, the tools are best used frequently and on small tasks.

For example, when using the *BW Layout Graph* tools, it is often better to run them on small groups of nodes as a way to help speed up basic alignment tasks and compliment you personal layout style.

While the tool can run on large networks and node chains, every artist and/or material will have a unique structure and preference for graph layout, which the artist should still maintain.

This is especially true for *BW Optimize Graph* as this tool will delete duplicate node chains and thus modify the existing layout unexpectantly. If you run this tool at the end of your work and on a large graph, it may be difficult to see what was actually optimised away for example. So again, it is better to use the tool to optimise your work as you go.

BW Straighten Connection is very useful for visually tidying up your graph, but it does insert a lot of extra dot nodes. So make sure you do not forget about the *Remove Connected Dot Nodes Hotkey* feature to remove these dot nodes if you want.

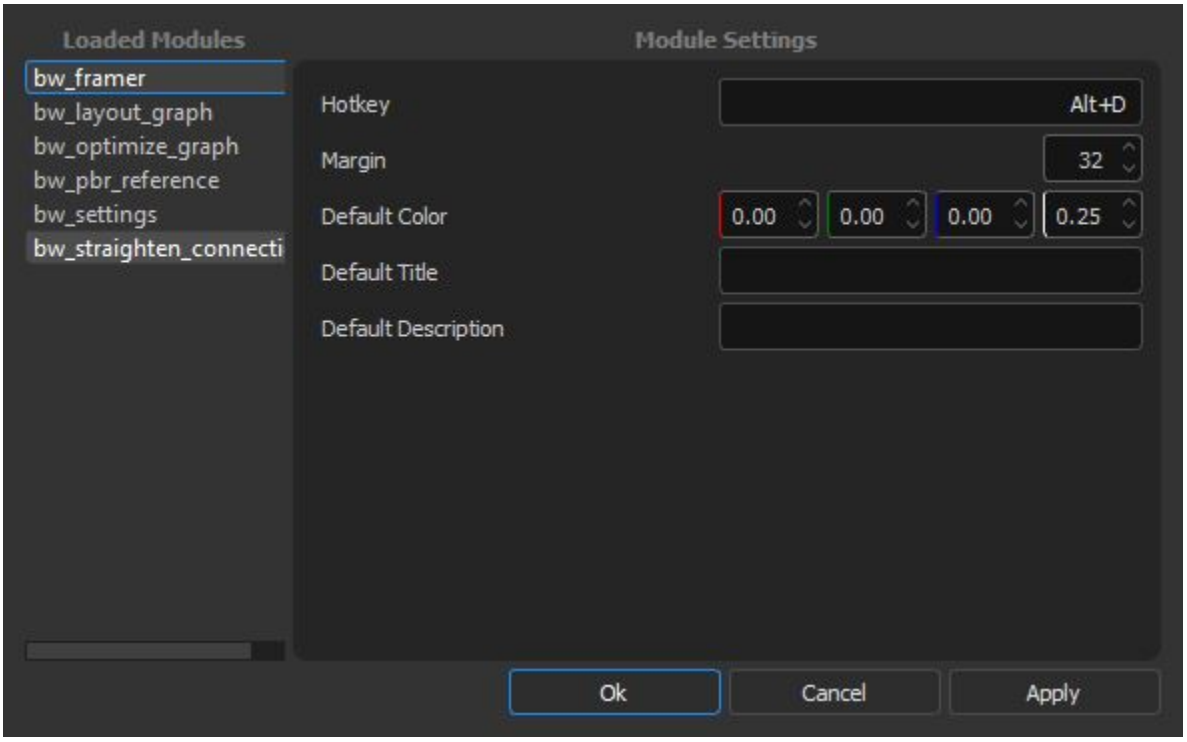
CHAPTER

FOUR

TOOL DOCUMENTATION

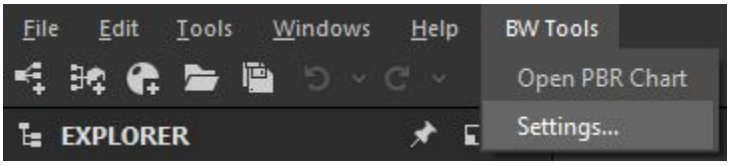
4.1 BW Settings

Window displaying all settings for all loaded modules



4.1.1 Settings Usage

Nativate to BW Tools > Settings…



4.1.2 Changing Hotkey Settings

Changing the hotkey for a setting requires a restart of Designer.

The string format for a hotkey is <Modifiers>+<Hotkey>, not including any quotation marks.

for example (exclduing the quotes)

`C` `Alt+D` `Ctrl+Shift+A`

4.1.3 Reverting To Default Settings

If you want to revert your settings for a particular module or otherwise run into an problem, you can delete the <module_name>_settings.json file directly and restart Designer.

The <module_name>_settings.json is located in the plugin <install directory>/bw_tools_2_0_0/bw_tools_2_0_0/bw_tools/modules/<mod
For example: `C:\Users\User\Documents\Adobe\Adobe Substance 3D Designer\python\sduserplugins\bw_tools_2\bw_tools_2\bw_tools\modules\bw_framer\bw_framer_settings.json`

Upon restart, a fresh settings file will be created with default settings.

4.2 BW Framer

BW Framer lets you frame your selection with a Hotkey and default settings. It can be thought of as an extension to the existing framing tools. Unlike the existing tools however, BW Framer will recognise and update existing frames to your new selection, meaning you no longer need to make manual adjustments or delete them to refit the frame.

4.2.1 Drawing New Frames

You can draw a new frame by selecting your nodes and running the framer tool. Default hotkey Alt+D.

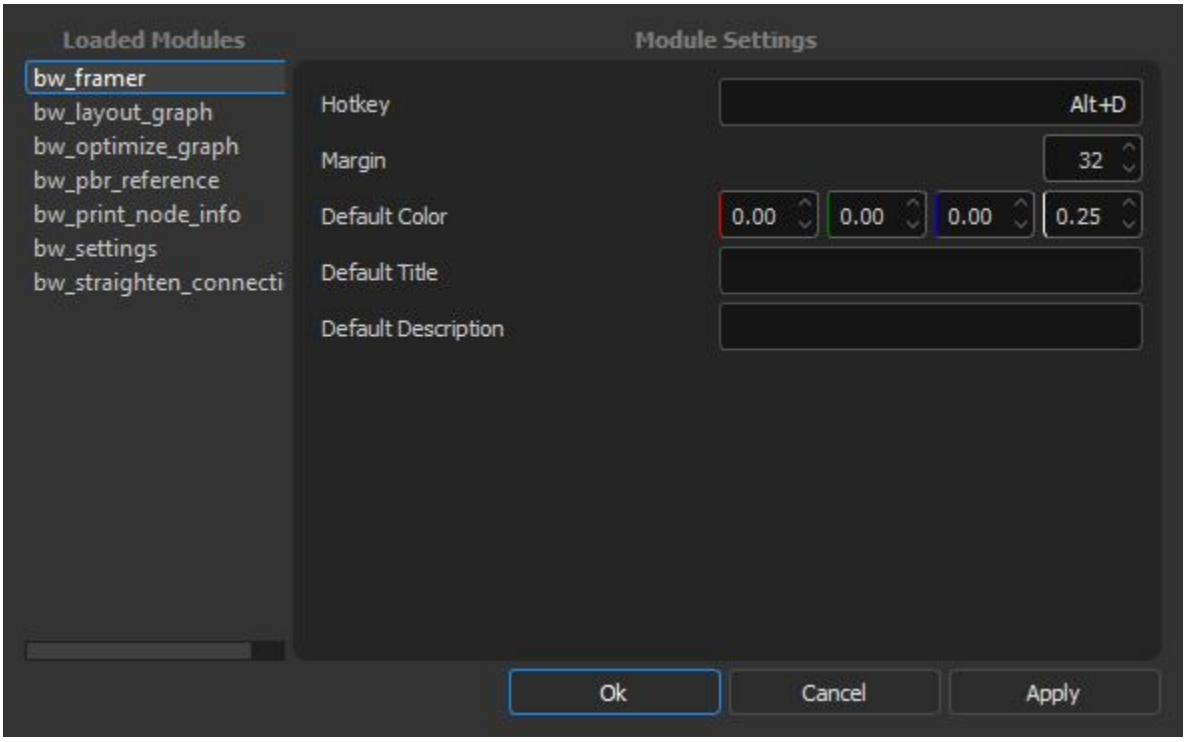
4.2.2 Redrawing Existing Frames

If you have a frame already selected, it will reuse that frame and refit it to your selection.

Multiple Frame Selections

If you select multiple frames in your selection, the left most frame will be used and the others deleted.

4.2.3 Framer Settings

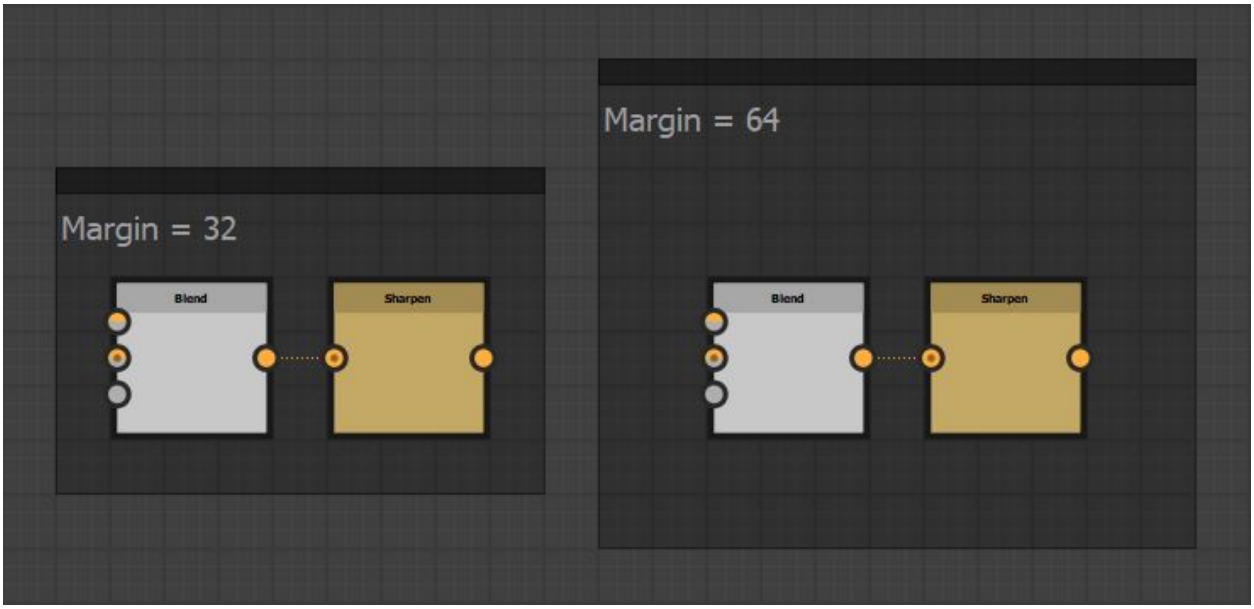


Framer Hotkey

The hotkey assigned to the run the tool, written as a string. Combine key combinations with "+". Requires a restart.

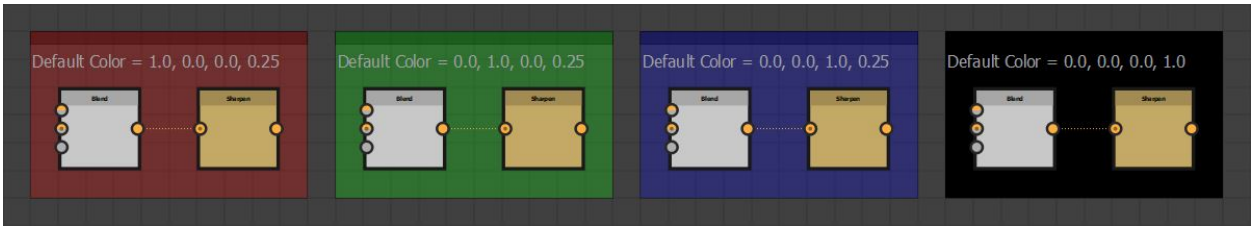
Margin

The spacing around the edges of the node.



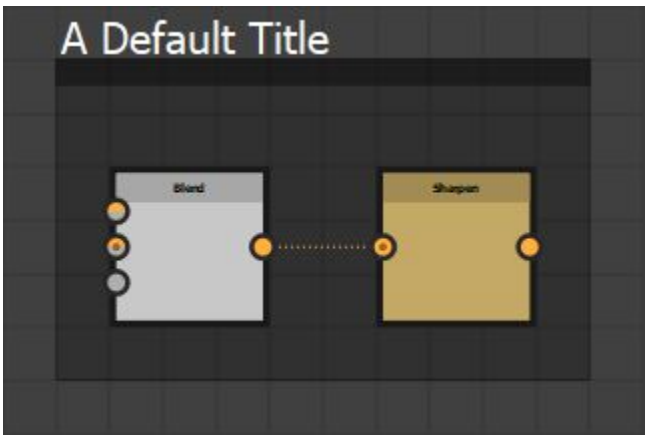
Default Color

The color of the frame in [RGBA].



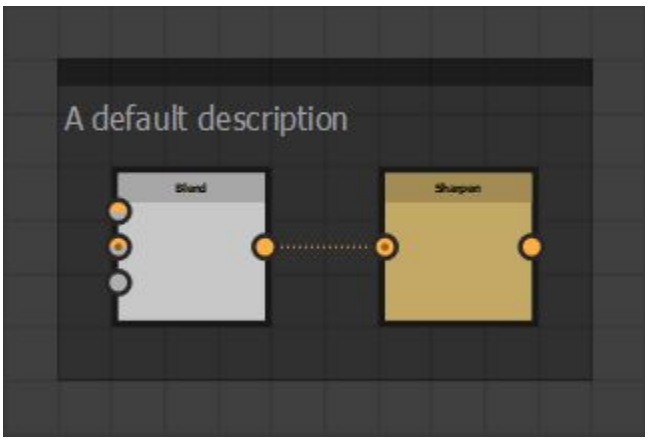
Default Title

The default title. Leave empty to remove the title.



Default Description

The default description. Leave empty to remove the description.



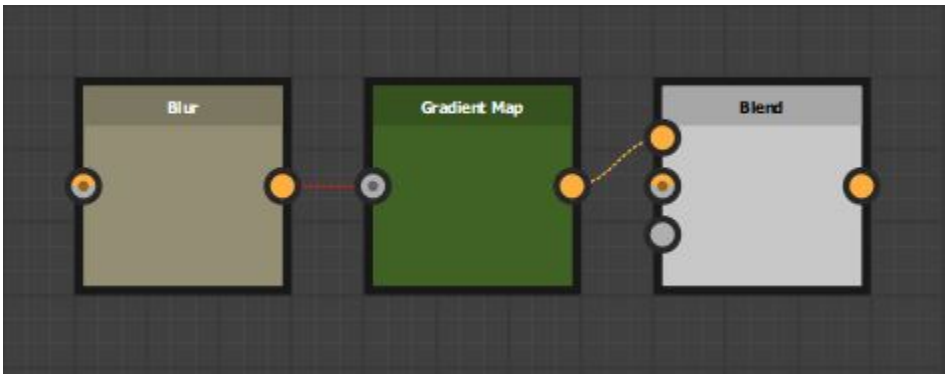
4.3 BW Layout Graph

Automatically align selected nodes based on their hierarchy, arranged to minimise overlapping. Align a given nodes inputs about their center point, stack them on top of each other or align them by their mainline. See *Mainline Concept*.

4.3.1 Node Placement Behavior

Simple Chain

Nodes are placed behind their output nodes.

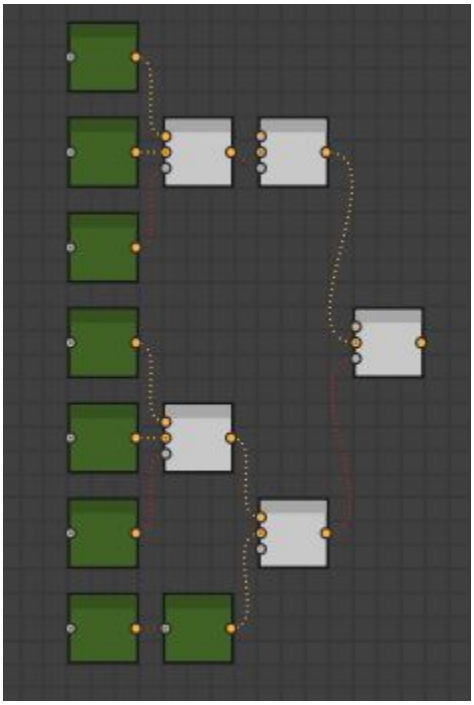


When a node has multiple outputs, it will align to the right most output. Visually creating the longest straight line.

Simple Hierarchy

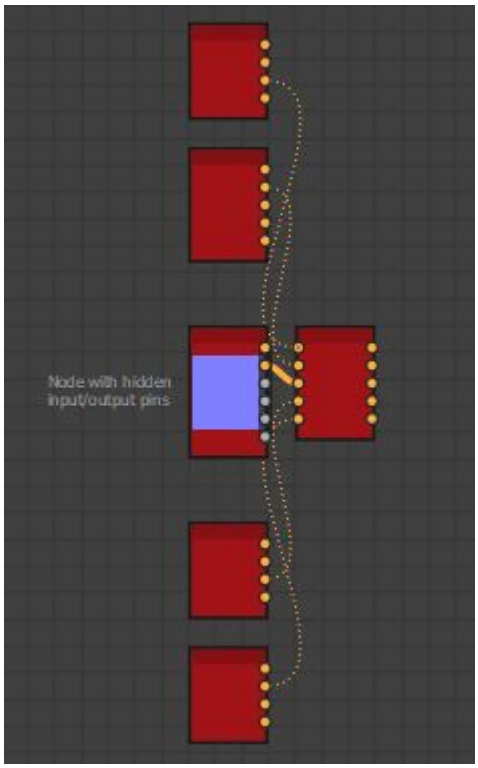
When a node has multiple inputs, they are aligned relative to the current position. Input nodes are stacked based on the input order and the specific alignment setting. See *Vertical Alignment Behavior*. Node heights are taken into account when aligning input nodes in a chain.

Where the hierarchy extends deeper, nodes will move to avoid any overlap.



Hidden Inputs/Outputs

Some nodes have hidden inputs or outputs, defined by the visibleif condition. Currently there is no way to access this information with the Designer API and as such they will be included in the height calculation. This means these node will have a height value as if all inputs and outputs were visible.



Root Nodes

Where nodes have no connected outputs, they are considered Root Nodes. When running the tool, these nodes will stay in place and all inputs will align to it

---

Root Node Behavior

That is, no connected outputs in your active selection. A Node may have connected outputs in your graph, but at the head of your selection, making it a root node.

If your selection contains multiple root nodes, they are considered as separate chains. Therefore, is important to provide enough spacing for the network to fully expand.

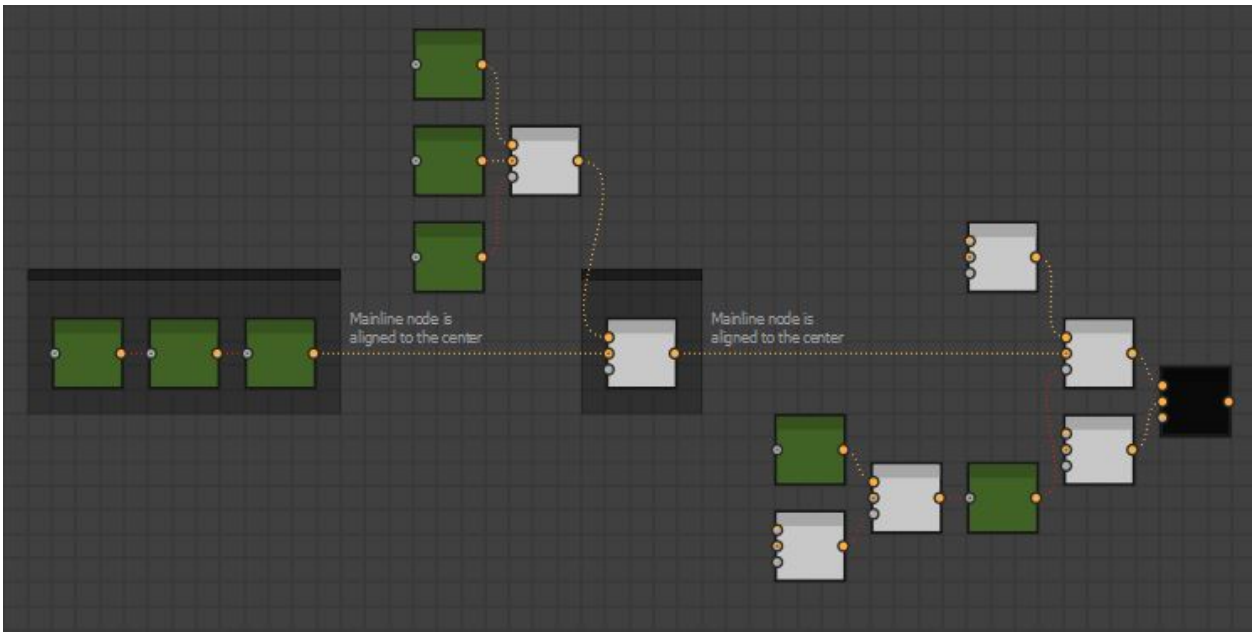
Vertical Alignment Behavior

Input nodes will align relative to their output. if a node has multiple outputs, it will align to the farthest output. See *Multiple Output Nodes*. Otherwise, alignment will be defined by the one of the following behaviors below.

- Mainline Alignment

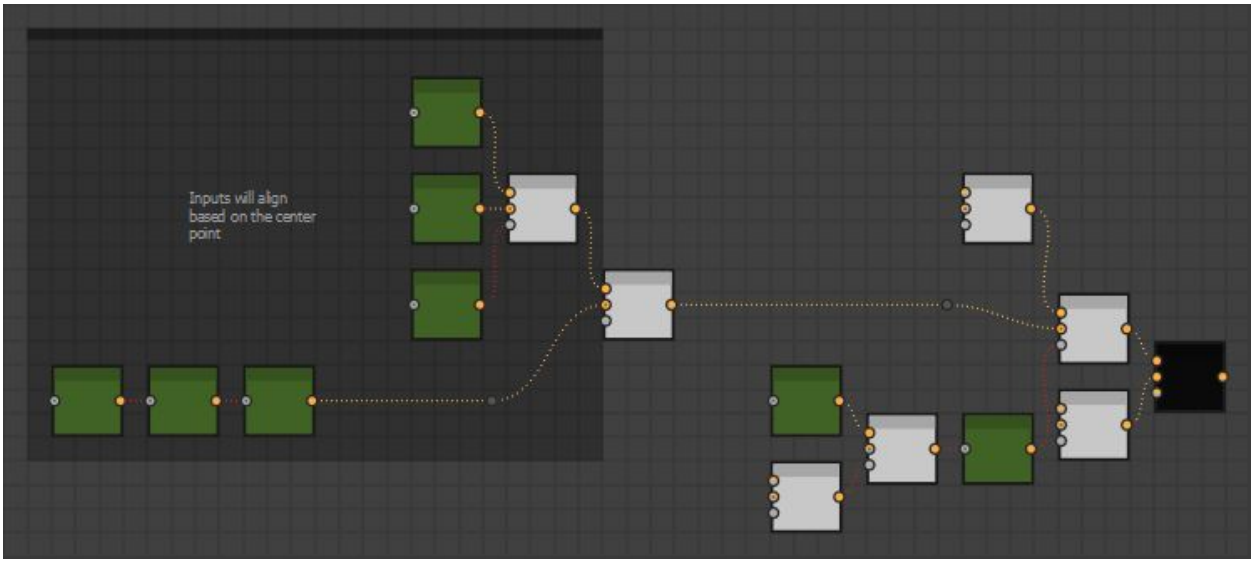
The mainline node will align to the center of the connected output node and siblings will position above or below accordingly. See *Mainline Concept* for information about what a mainline is.

In the event that a mainline node is not successfully found, alignment will revert to Center Alignment



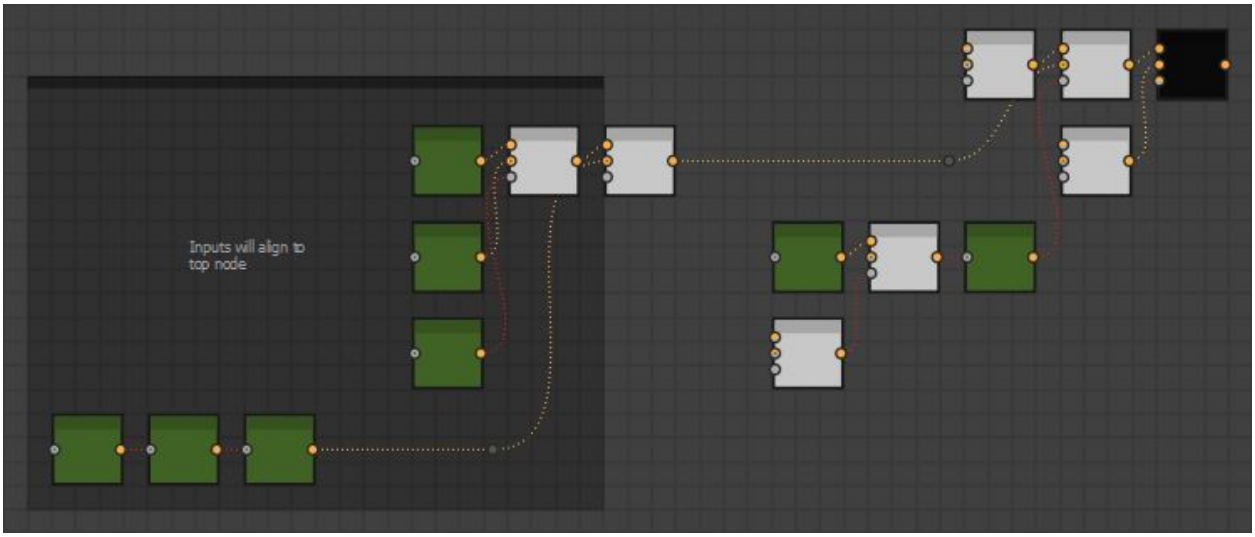
- Center Alignment

The center point of all the inputs will align to the connected output node.



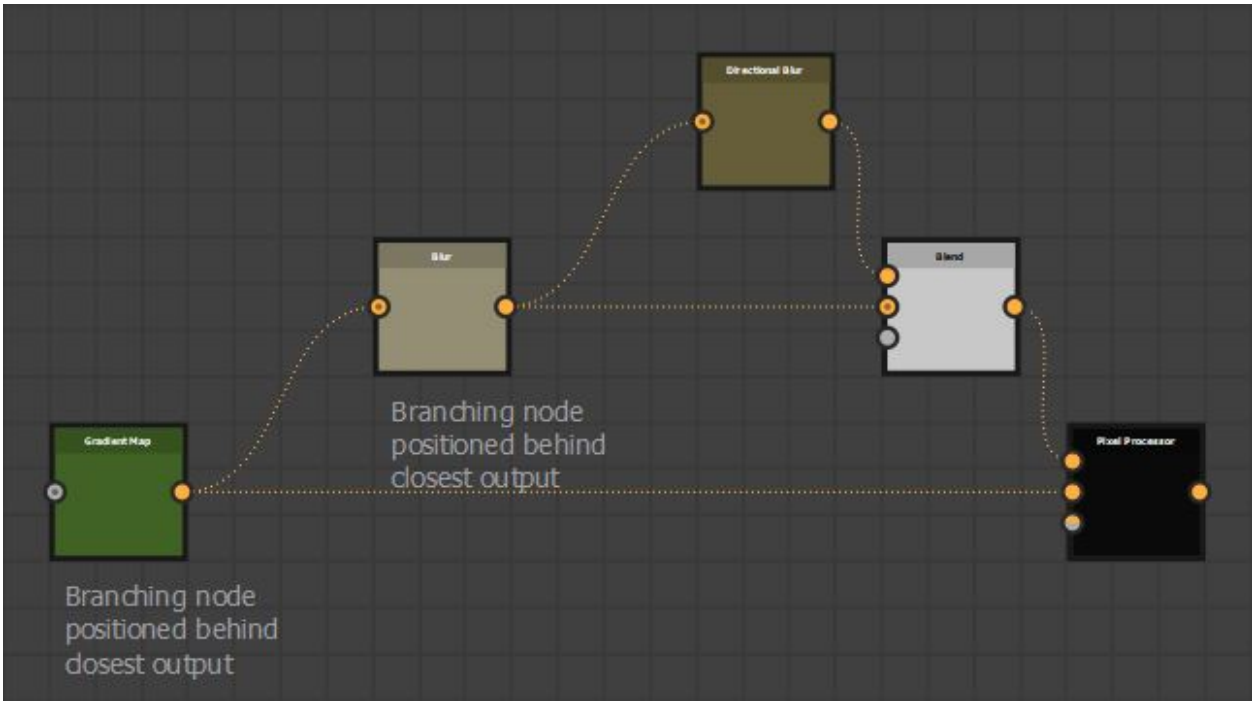
- Top Alignment

Input nodes are stacked ontop of each other and the top node is aligned to the connected output.



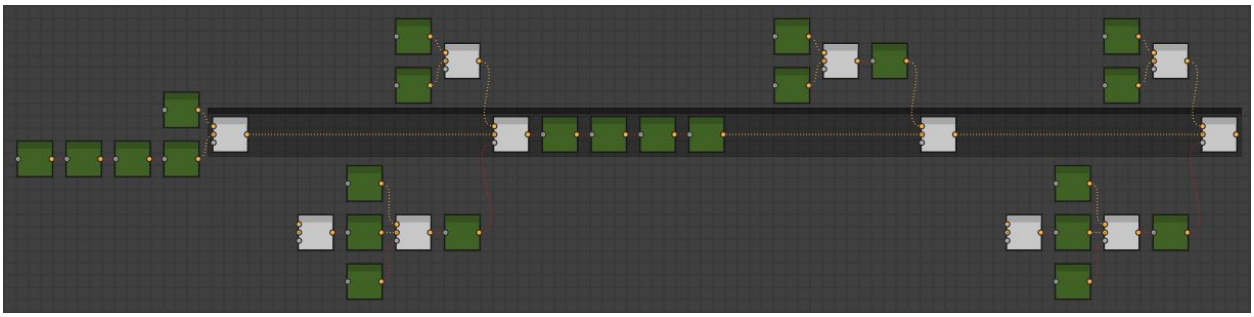
Multiple Output Nodes

Some nodes have multiple outputs and may connect to various points in the network. In these cases, the node will always be positioned behind the closest output and aligned with the farthest output.



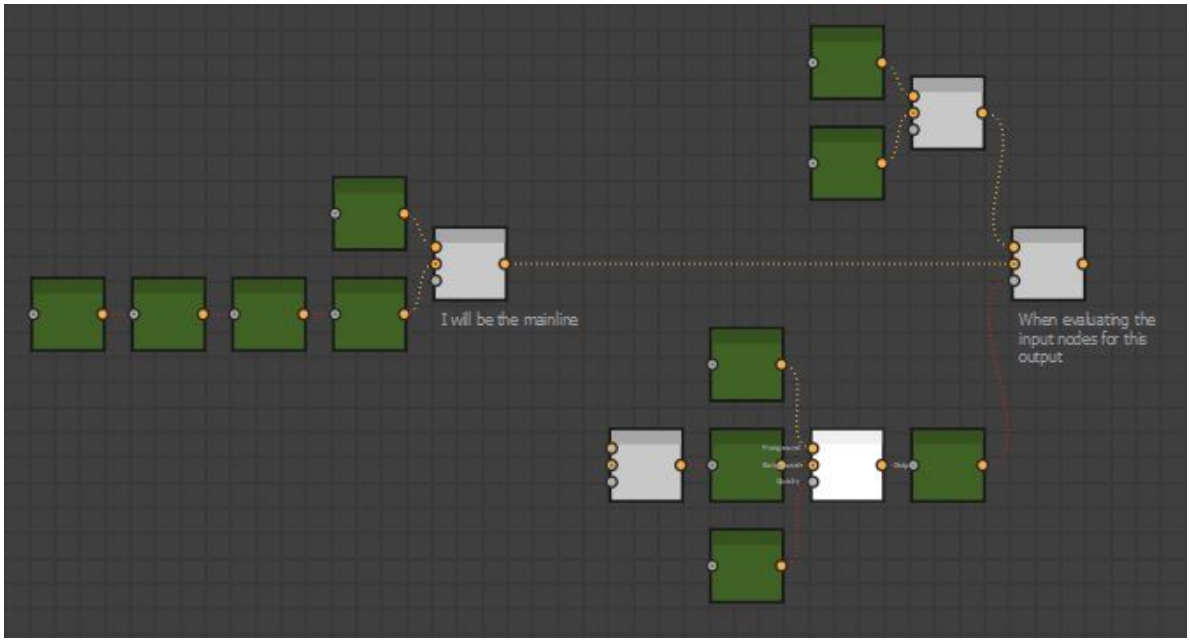
4.3.2 Mainline Concept

The tool will attempt to identify a mainline through the network. This is to mirror a common layout strategy used by many artists to organise their graphs, where a horizontal structure is defined and logical units connect into it.

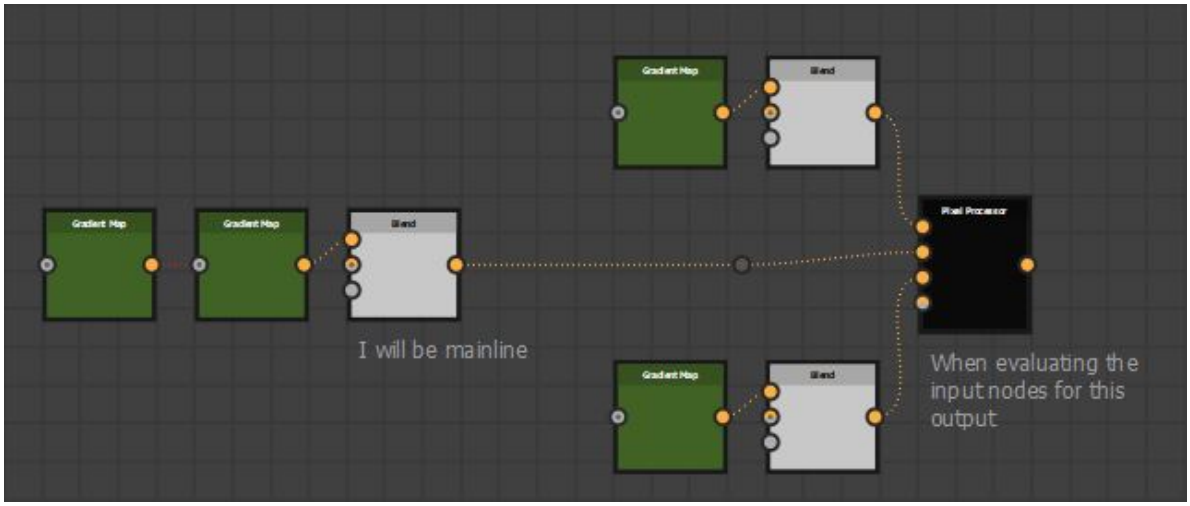


When evaluating the alignment for a given nodes inputs, the tool will use the following criteria to define which node is the part of the mainline.

1. From a given nodes inputs, the one deepest in the network will become the mainline.



2. If all nodes are in line with each other, the input node with the deepest chain will become the mainline.



3. If all input node chains are of the same length, it will declare no mainline node could be found.

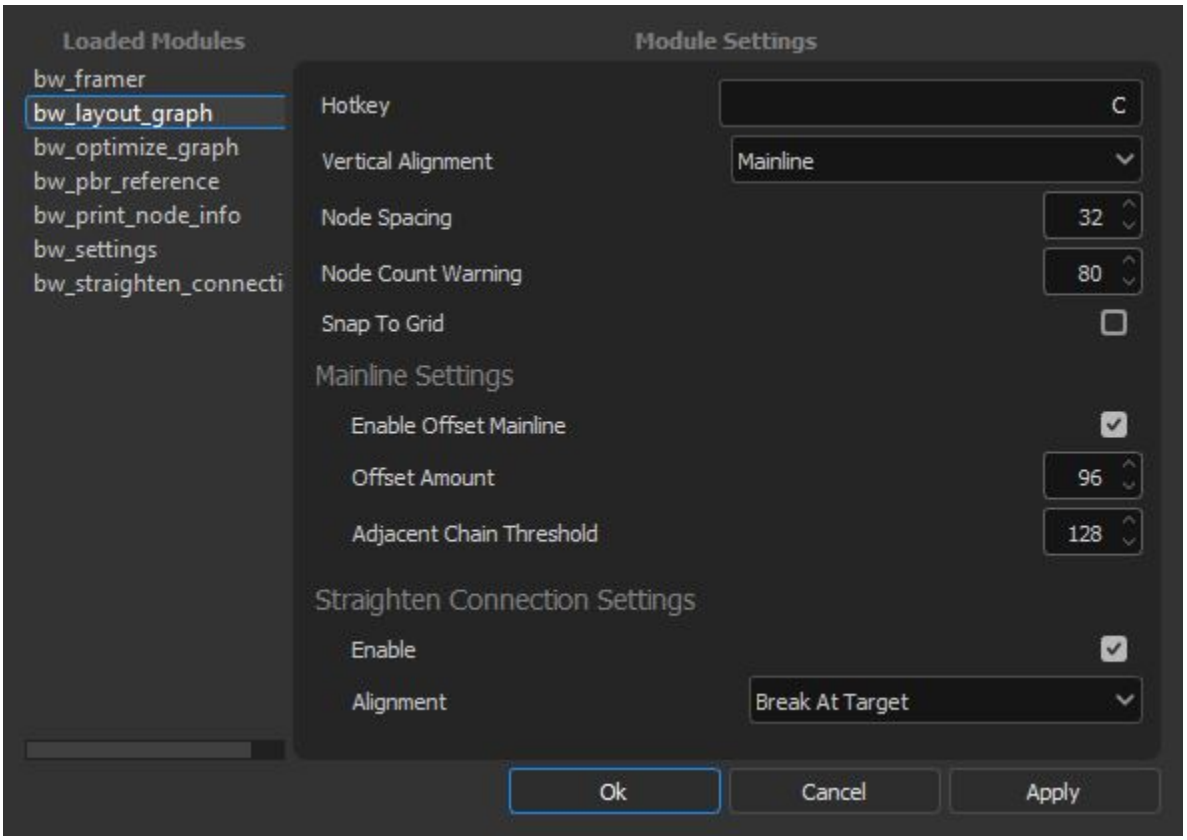
When a mainline node was successfully found, it will either be moved back, based on the *Mainline Settings* or not. If a mainline node is not found, no nodes will be moved back. In either case the *Vertical Alignment Behavior* will define how the input nodes align vertically.

4.3.3 Evaluation Order

The tool runs in several passes and is evaluated from left to right, top to bottom, meaning the deepest part of the network is solved first.

1. Sort nodes by their hierarchy.
2. Apply mainline offsets, if Enable Offset Mainline is on.
3. Align nodes vertically basee on their *Vertical Alignment Behavior*.
4. Apply node snapping, if Snap To Grid is on.
5. Insert dot nodes, if *Straighten Connection* is turned on.

4.3.4 Layout Settings



Layout Hotkey

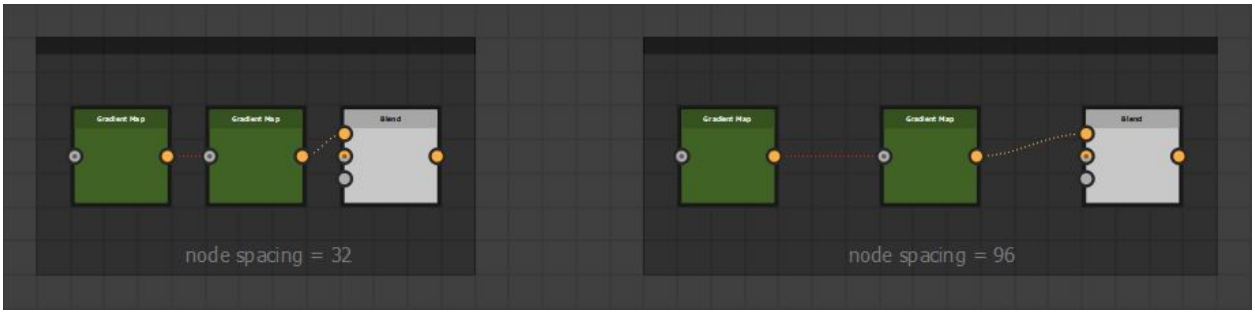
The hotkey assigned to the run the tool, written as a string. Combine key combinations with "+". Requires a restart.

Vertical Alignment

The vertical alignment behavior to use. See *Vertical Alignment Behavior* for more information.

Node Spacing

The spacing between nodes, given in absolute units of the graph grid.



Node Count Warning

Whether or not to prompt the user if more node than the given threshold are selected when running the tool. This is helpful to alert the user that the tool could potentially take some time to run.

Snap To Grid

Whether or not to snap nodes to the grid after running the tool. This setting does not apply to any dot nodes inserted by the tool.

Note: This setting uses Designer's native snap to grid script

4.3.5 Mainline Settings

Enable Offset Mainline

Whether or not to offset the mainline node. When set to true, the mainline node is only moved back if one is successfully found in a nodes inputs. See *Mainline Concept*

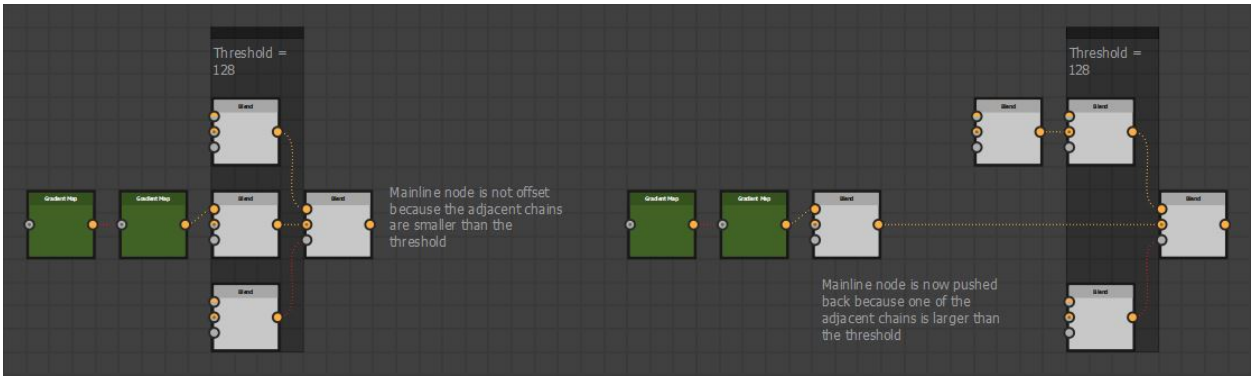
Offset Amount

The amount to offset the mainline node, given in absolute units of the graph grid.

Note: This is in addition to the node spacing value.

Adjacent Chain Threshold

When a mainline node is found, this setting defines the threshold by which the sibling chains depth must be larger in order to apply the offset.



4.3.6 Straighten Connection

Information about the Straighten Connection can be found *BW Straighten Connection*.

Enable

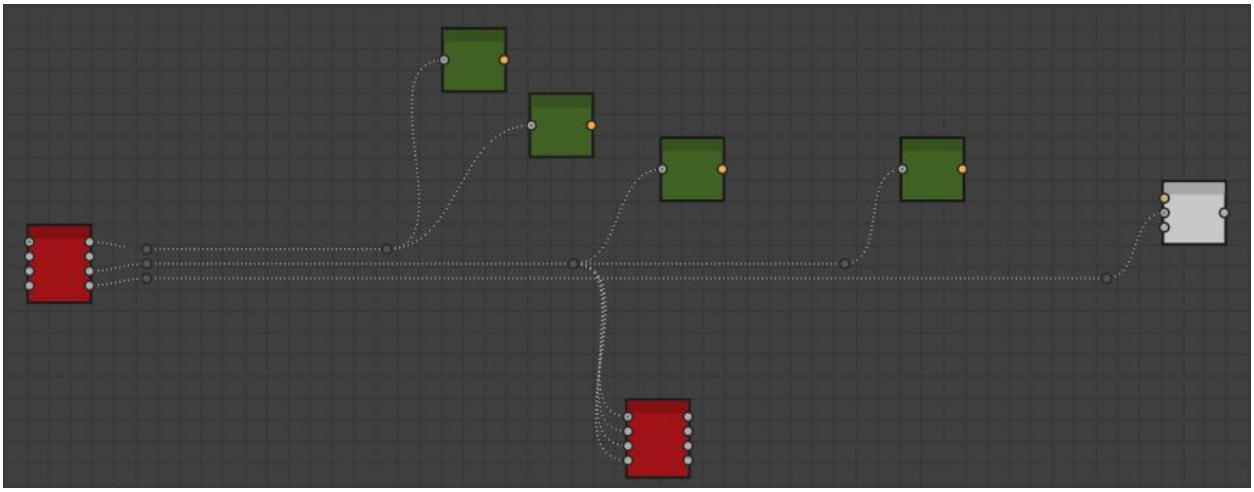
Whether or not to run the straighten connection tool after running the layout tool.

Alignment

The algorithm to use when running straighten connection after running the layout tool. See

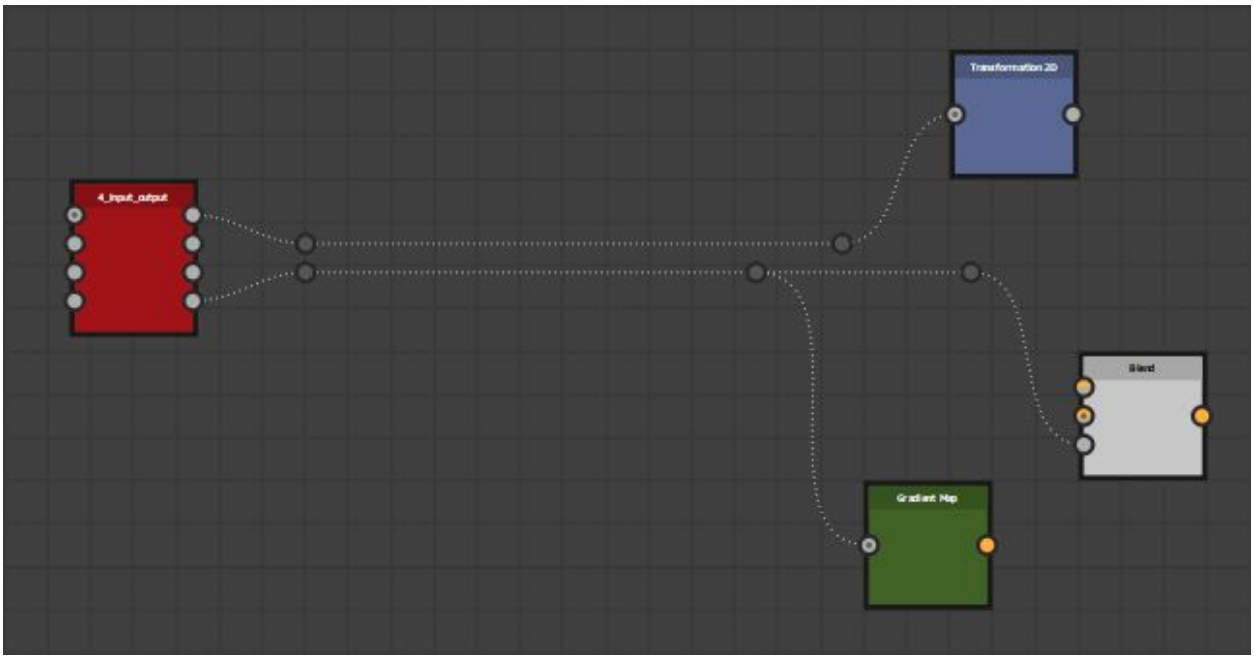
4.4 BW Straighten Connection

Straightens connection from selected nodes to all outputs by inserting dot nodes into the connection. There are two modes of operation which defines the visual behavior. *Break At Target* and *Break At Source*.



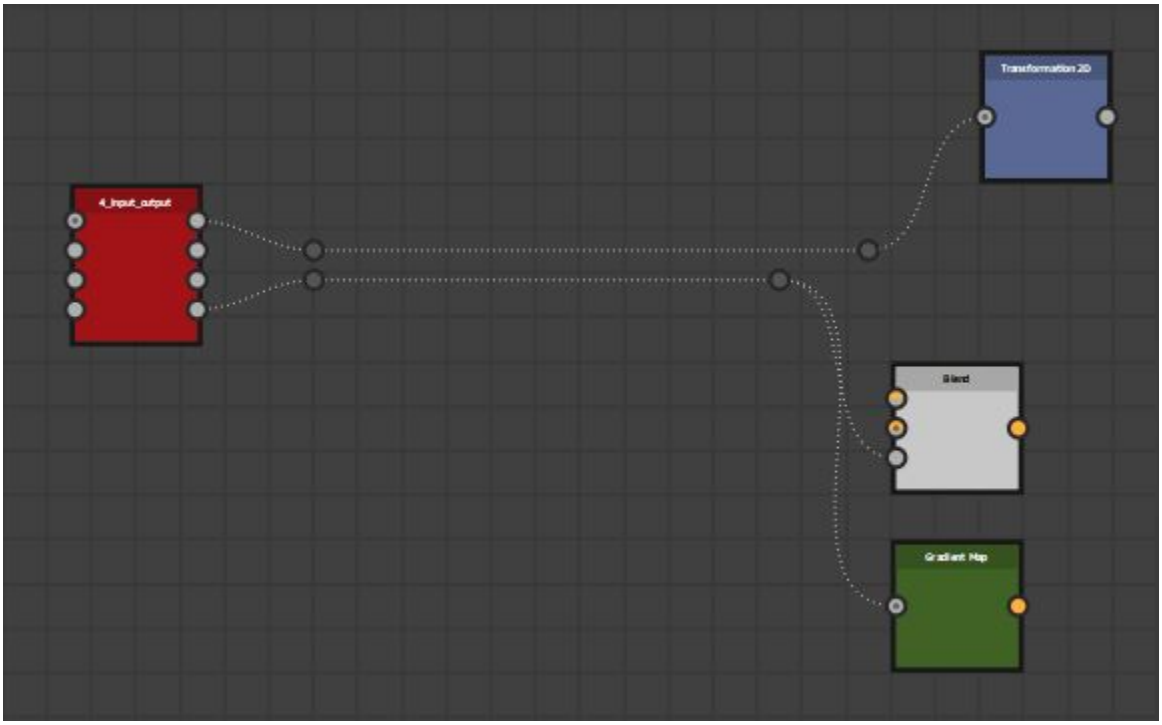
4.4.1 Break At Target

This mode will align dot nodes horizontally inline from the source node and break away when they reach the target.



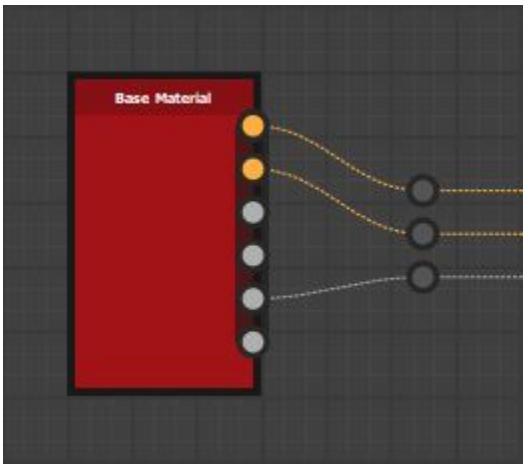
If a given output has more than one connection, the dot nodes will chain together.

If two more more output nodes are close together, they will reuse an existing dot node.



Multiple Outputs

Where nodes have multiple outputs, but fewer connections, dot nodes will align to the center of the source node.



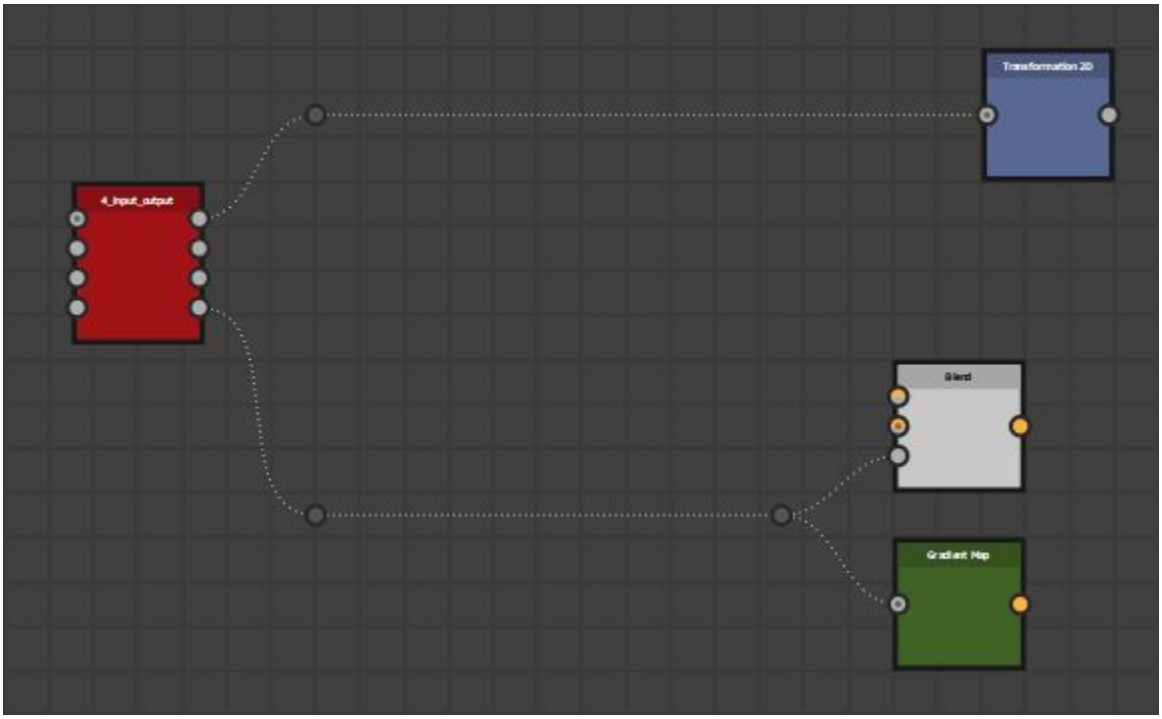
4.4.2 Break At Source

This mode will break away at the source node and align horizontally with the target.

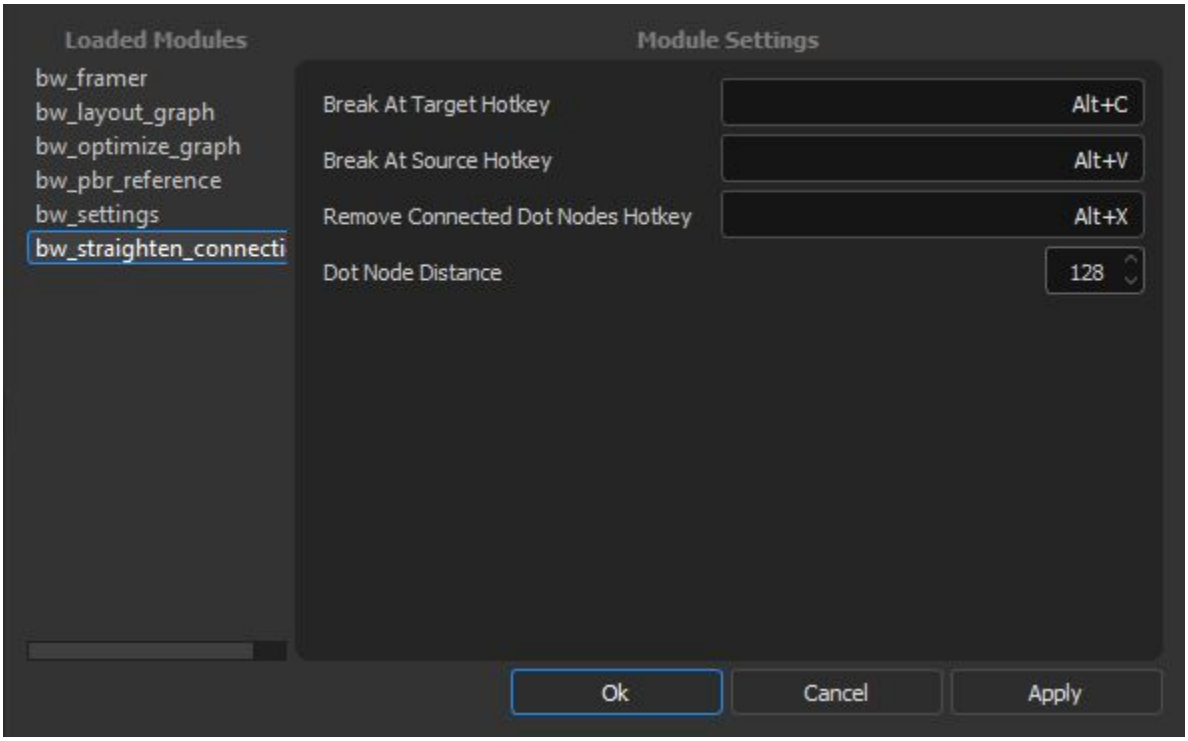
If a given output has more than one connection, the dot nodes will chain together.

If a given output has more than one connected node, the dot nodes will align to the mid point of those output nodes.

If two more more output nodes are close together, they will reuse an existing dot node.



4.4.3 Straighten Connection Settings



Break At Target Hotkey

The hotkey assigned to the run *Break At Target*, written as a string. Combine key combinations with "+". Requires a restart.

Break At Source Hotkey

The hotkey assigned to the run *Break At Source*, written as a string. Combine key combinations with "+". Requires a restart.

Remove Connected Dot Nodes Hotkey

The hotkey assigned to remove dot nodes connected to a node, written as a string. Combine key combinations with "+". Requires a restart.

Useful for when you wish to clean up the dot nodes created from running the other tools.

Dot Node Distance

The distance from the nodes a dot node will be inserted.

4.5 BW Optimize Graph

Attempts to optimise your selection for graph performance by removing duplicate nodes and setting appropiate output sizes.

4.5.1 Removing Duplicate Nodes

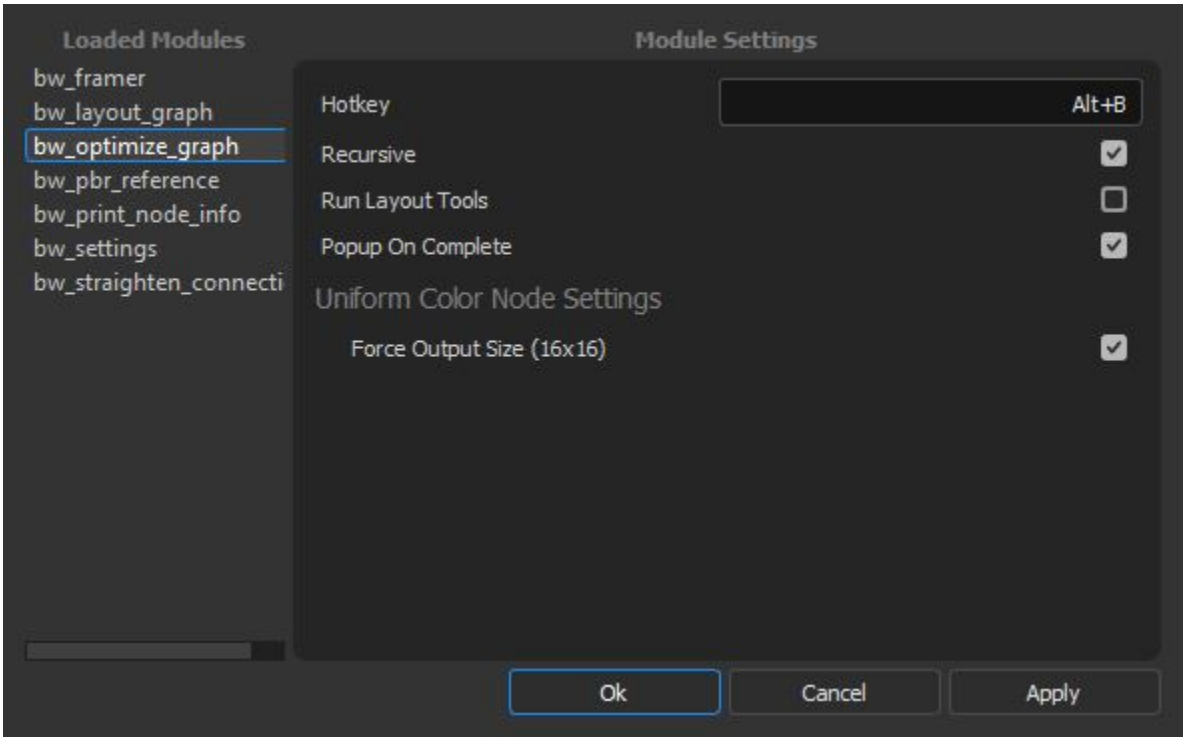
The tool analyzes each node in your selection to identify duplicates nodes. A node is considered a duplicate if all parameter settings and inputs are the same. If recursive is checked, will continue to run the tool on your selection, until no more duplicate nodes are found. This is useful to find and remove duplicate chains of nodes.

If a property has a function graph connected, it will not be considered a duplicate, even if all other parameters are.

4.5.2 Uniform Color Nodes

All uniform color nodes in your selection will have their output size forced to 16x16 (the optimal output size for Designer) and the connected outputs set to relative to parent. If the uniform color node has a function graph applied to the output size property, it will not be changed.

4.5.3 Optimize Settings



Optimize Hotkey

The hotkey assigned to the run the tool, written as a string. Combine key combinations with "+". Requires a restart.

Recursive

When checked, will recursively run the tool until no more nodes have been optimized.

Run Layout Tools

Whether or not to run the layout graph tools after running the optimizer. See *BW Layout Graph*.

Popup On Complete

When checked, a popup will prompt the user with information about what was done to the graph. The same information is printed to the console regardless of this setting.

4.5.4 Uniform Color Node Settings

Force Output Size (16x16)

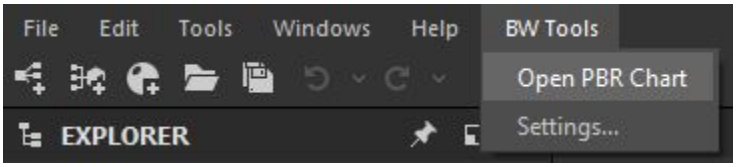
Whether or not to optimize uniform color nodes output size.

4.6 PBR Reference

A convenient PBR color chart built directly into Designer. It provides various pbr values, based on DONTNOD 2014 pbr chart, to quickly and easily reference without the need to having a downloaded color chart opened in windows.

4.6.1 PBR Reference Usage

To open the pbr chart, navigate to BW Tools > Open PBR Chart



4.6.2 Color Picking

The window is always displayed ontop of the designer viewport, making it easy to color pick from

4.6.3 Dragable Swatches

Swatches are select-able, drag-able and hide the UI to allow for easy comparison with your texture. Useful to click and drag over the 2d view to visually compare values directly.

4.7 Unsupported Graph Types

Currently, the Designer API has many bugs with the new graph types added in 11.0.0 which causes the tools to error or crash. For this reason, I have disabled the tools from running in these unsupported graph types. I hope to support all graph types in future when the Designer API has fixed the bugs.

Current supported graph types are

1. Compositing Graph
2. Function Graph

Current unsupported graph types are

1. Model Graph
2. MDL Graph
3. FX Map Graph

CHAPTER

FIVE

SUPPORT

If you are having issues with the tool, please check the following:

1. You are using a compatible version of Designer, check *Compatibility*
2. Check you only have one version of the tool installed. See *How To Install* for more info.
3. You may be trying to use a tool on an *Unsupported Graph Types*

Otherwise, please contact me directly on Artstation.

CHAPTER

SIX

FOR DEVELOPERS

If you wish to browse or extend the code, go to git hub page

Here is some general information which will be useful to know.

6.1 Working With Modules

You can add new modules by defining a folder and .py file with the same name inside `./bw_tools/modules`.

For example:

```
bw_tools
├── modules
    ├── my_new_module
        ├── __init__.py
        ├── my_new_module.py
```

You must define a on_initialize(api: BWAPITool) function inside my_new_module.py for the plugin to automatically load your module.

```
def on_initialize(api: BWAPITool):
    # Do any initialization here
    ...
```

All logic to initialize your module must be done from this function.

6.2 Adding Actions To The Menu Bar

You should add new actions to the BW Tool menu bar in the on_initialize function, using the api tool.

```
def on_initialize(api: BWAPITool):
    action = api.menu.addAction("My Action")
    action.triggered.connect(func_to_run)
```

6.3 Adding Actions To The Graph View Toolbar

The plugin will handle creating and managing the graphview toolbar itself, but you must add your actions to it in the on_initialize function.

Use the api tool to register a new graph view created callback which adds your new actions to the graph view toolbar, also accessed through the api tool.

```
def on_graph_view_created(graph_view_id, api: BWAPITool):
    # Get the toolbar for the new graph view
    toolbar = api.get_graph_view_toolbar(graph_view_id)

    # Create a new action
    action = QAction()
    action.triggered.connect(func_to_run)

    # Add action to graph view toolbar
    toolbar.add_action("my_tool_name", action)


def on_initialize(api: BWAPITool):
    api.register_on_graph_view_created_callback(
        functools.partial(on_graph_view_created, api=api)
    )
```

6.4 Working With Setting Files

Module settings are stored in json and must be named `<module>_settings.json` and live along side the module.py file

```
bw_tools
├── modules
    ├── my_new_module
        ├── __init__.py
        ├── my_new_module.py
        ├── my_new_module_settings.json
```

The json file must follow a specific format in order for the plugin to automatically generate UI for your module.

You define the settings name with the key of a dictionary, the value of which then define the properties for the setting.

```
{
    "Hotkey": {
        "widget": 1,
        "value": "Alt+C"
    }
}
```

6.4.1 Setting properties

Properties are defined with the value or a diction key.

- widget - Enum int to define which widget the UI should use. Refers to the `WidgetTypes` Enum inside `bw_tools/modules/bw_settings/settings_loader.py`.
- value - The value for the setting.
- list - The list of possible values when populating a combobox. Only available for combobox widget types.
- content - The content of a groupbox widget, the value of which should be a dictionary containing the settings inside the groupbox.

```
{
    "My String Setting": {
        "widget": 1,
        "value": "my string value"
    },
    "My Int Setting": {
        "widget": 2,
        "value": 32
    },
    "My Float Setting": {
        "widget": 3,
        "value": 32.0
    },
    "My Bool Setting": {
        "widget": 4,
        "value": true
    },
    "My Combobox Setting": {
        "widget": 5,
        "list": [
            "Option 1",
            "Option 2",
            "Option 3"
        ],
        "value": "Option 1"
    },
    "My RGBA Setting": {
        "widget": 6,
        "value": [
            1.0,
            1.0,
            1.0,
            1.0
        ]
    },
    "My Group Box": {
        "widget": 0,
        "content": {
            "My Sub Setting": {
                "widget": 4,
                "value": true
```

(continues on next page)

(continued from previous page)

```
            }
        }
    }
}
```

6.4.2 Providing Default Settings

To provide default settings for a module, you must define a get_default_settings function which returns a dict inside the main module.py file.

If your module declares this function, the plugin will automatically generate a module_settings.json file if one was not found.

```
def get_default_settings() -> Dict:
    return {
        "My Hotkey": {"widget": 1, "value": "Alt+D"},
        "My Value": {"widget": 2, "value": 32},
    }
```

6.5 General Helper Classes

There are some classes inside bw_tools/common to help with general API tasks.

6.5.1 BWAPITool Class

This helper class is to simplify the interface with Designer's API. It handles the Designer application related tasks such as getting the package manager or adding UI elements to the UI, shown in *Adding Actions To The Menu Bar*.

```
api_tool = BWAPITool()

# Get the current node selection
sd_nodes = api_tool.current_node_selection

# Get the current graph
sd_graph = api_tool.current_graph

# Get the Designer main window widget
designer_main_window = api_tool.main_window
```

6.5.2 BWNode Class

This node is a wrapper around the Designer API SDNode type which makes accessing properties a little easier. Most of the modules that come with bw_tools make use for BWNode.

This class is designed to work with the *BWNodeSelection Class*, which handles creating these nodes, allowing you to query a nodes inputs and outputs within the selection.

```
api = BWAPITool()
sd_nodes = api.current_node_selection

# Create a BWNode
node = BWNode(sd_nodes[0])

# Print a nodes label
print(node.label)

# Print a nodes position
print(node.pos)
```

6.5.3 BWNodeSelection Class

This node lets you define a selection of BWNode's, allowing you to query a nodes inputs and outputs.

When initializing a BWNodeSelection, the input and outputs of a given node are only added if they are in same selection. This behavior differs from the Designer API which always returns all connected nodes, regardless of selection.

```
api = BWAPITool()

selection = BWNodeSelection(api.current_node_selection, api.current_graph)

# Get a particular node from the selection
bw_node = selection.node(identifier)

# Because we got the BWNode through BWNodeSelection Class,
# we can now query only the connections in the selection
output_nodes = bw_node.output_nodes

# Confirm a node is in the selection
selection.contains(output_nodes[0])
```

6.6 Running Unit Tests

The unit tests are written to be run inside Designer, using the built in Python Editor.

Dependencies

The unit tests depend on PIL so you must pip install Pillow into the designer install directory!

To run the tests, open run_unit_tests.py in the Designer Python Editor and run. This will load a number of Designer graphs used by the unit tests and execute them automatically.

CHAPTER

SEVEN

DOWNLOAD

Download here

CHAPTER

EIGHT

OVERVIEW

BWTools is a series of plugins for Substance Designer, aimed at speeding up graph organisation tasks. The plugins come as a package which neatly slots directly into Designers UI, where the user can set hotkeys for the various tools along with their associated settings.

If you have used an older version, please check out *Notable Changes From BWTools 1.x*.

The plugin is split into a series of modules, each designed to solve a particular problem. Check out *Tool Documentation* for a more detailed look at each of the tools.

I recommend looking at *General Guide To Using The Tools* to get the most out of the tools.

If you are looking to work with the code or want to understand more, check out my git hub page

CHAPTER

NINE

INSTALLING THE TOOLS

To get the tools installed, follow this guide *How To Install*

CHAPTER

TEN

COMPATIBILITY

Compatible with Substance Designer 11.2.0 onwards

CHAPTER

ELEVEN

LICENSE