Resource Wrangler

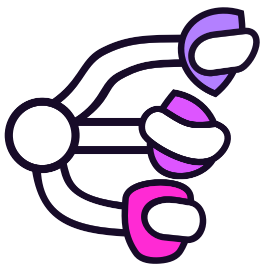


Figure 1: Logo

The Resource Wrangler is way to *visualize and build* resources and their interconnections.

In Godot, resources are everywhere—they are textures, meshes, shapes, animations and on and on. Resources also contain other resources.

The inspector in Godot is cool, but it's very hard to understand the nesting of resources by using it because it has to stick to a narrow column and an awkward way of showing data within data. This addon takes resources and *stretches* them out sideways in a node graph.

You can then manipulate which textures, say, are going into what materials and where those are going into meshes and so on. It's all visual—which is what nodes do well.

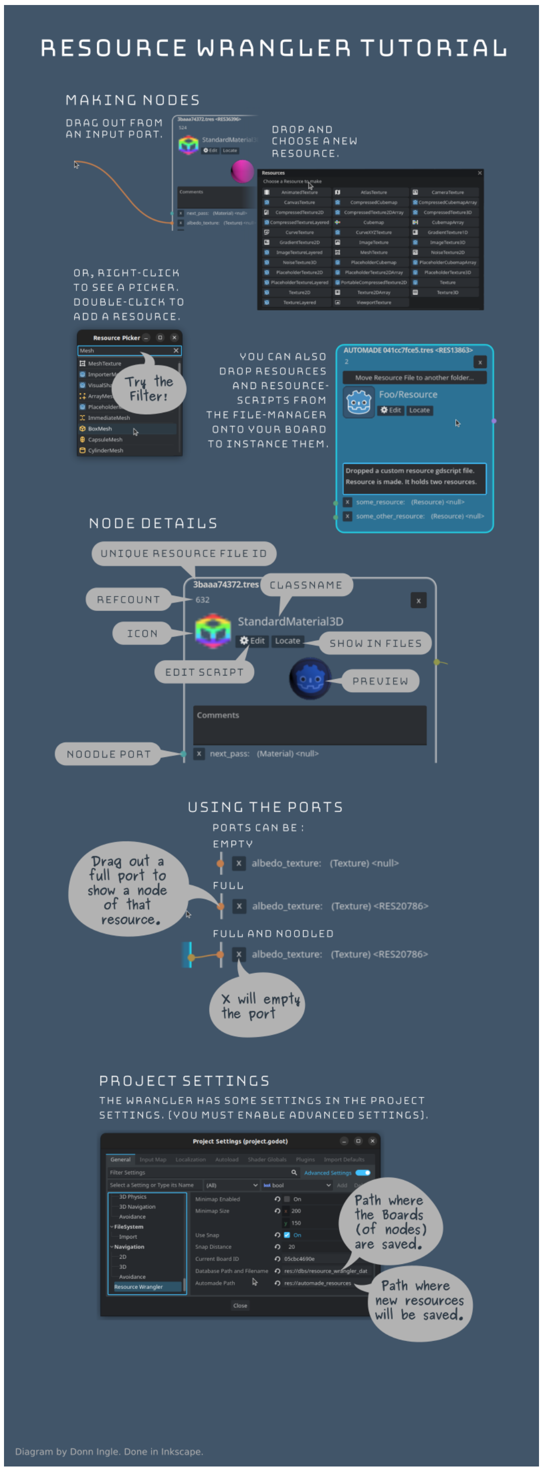
Once the Addon is activated (in Project Settings) you should see a new button named "Resource Wrangler" in the top-middle menu area, alongside the AssetLib.

Quick Tutorial

You can make new 'boards' (the blue graph screen) on the little toolbar. They are all saved in the database tres file whose path and name you can set in the Project Settings.

To make nodes, you can:

1. Drop resources (or resource-based custom classes) onto the blue board from the File Manager. New: You can also drag-drop resources from the inspector. Ymmv.
2. Right-click and choose from the Resource Picker.
3. From any node, you can drag from the **OUTPUT** ports to make a new input resource. Yes, it's weird, but think about it a bit... If the port was null, you'll see a list of suitable choices in a grid of buttons. If not, that



RESOURCE WRANGLER TUTORIAL

MAKING NODES

DRAG OUT FROM AN INPUT PORT.

DROP AND CHOOSE A NEW RESOURCE.

OR, RIGHT-CLICK TO SEE A PICKER. DOUBLE-CLICK TO ADD A RESOURCE.

YOU CAN ALSO DROP RESOURCES AND RESOURCE-SCRIPTS FROM THE FILE-MANAGER ONTO YOUR BOARD TO INSTANCE THEM.

Dropped a custom resource gdscript file. Resource is made. It holds two resources.

NODE DETAILS

USING THE PORTS

PORTS CAN BE:

EMPTY

FULL

FULL AND NOODLED

PROJECT SETTINGS

THE WRANGLER HAS SOME SETTINGS IN THE PROJECT SETTINGS. (YOU MUST ENABLE ADVANCED SETTINGS).

Minimap Enabled: On
Minimap Size: x 200, y 150
Use Snap: On
Snap Distance: 20
Current Board ID: 05cbc4690e
Database Path and Filename: res://dbs/resource_wrangler_dat
Automade Path: res://automade_resources

Diagram by Donn Ingle. Done in Inkscape.

Figure 2: July 5, 2023

resource will be shown as a new node. **Note**: If the port is an Array[] then it will only let you make new entries, it won't drag them out like other ports. See Array section below for more.

Please note: Not all resource types can be made. I filter-out the ones that can be, so if you can't find a resource that's probably why.

Port Thoughts

A port represents an actual Resource property in some Object. That property can be empty (null) or full (a reference to a Resource Instance). When you disconnect a noodle (by closing a node, for example) it *does not* mean that the property will become empty. This is a design decision made to prevent losing work by mistake.

There is a little "X" for you to actually empty a port. Undo works.

New resources

When a new resource is made, it is automatically saved into the "Automade Path". You can also change that path in the settings.

Each resource is saved as a .tres file with a unique ID.

Automades

As you work with the nodes, you will often be making new resource files. These are saved into the "Automade Path" (see the settings). The files in that folder can become out of of date as you work, so there is a button on the top-right named "prune automades" which will try to match all actually used resource files and then move all the ones that are not being used (by any other nodes) into sub-folders named "have_deps" and "have_no_deps". You'll have to figure out how you want to handle things from there.

If you find any missing resources, (you'll see red-shaded nodes), the chances are a resource file has been moved into 'orphans'.

Moving Resource Files

If you a resource is in a board as a node (i.e. visually) then it is owned by the board database. The best way to move such a resource to a new path is to use the **Save As** button on the node.

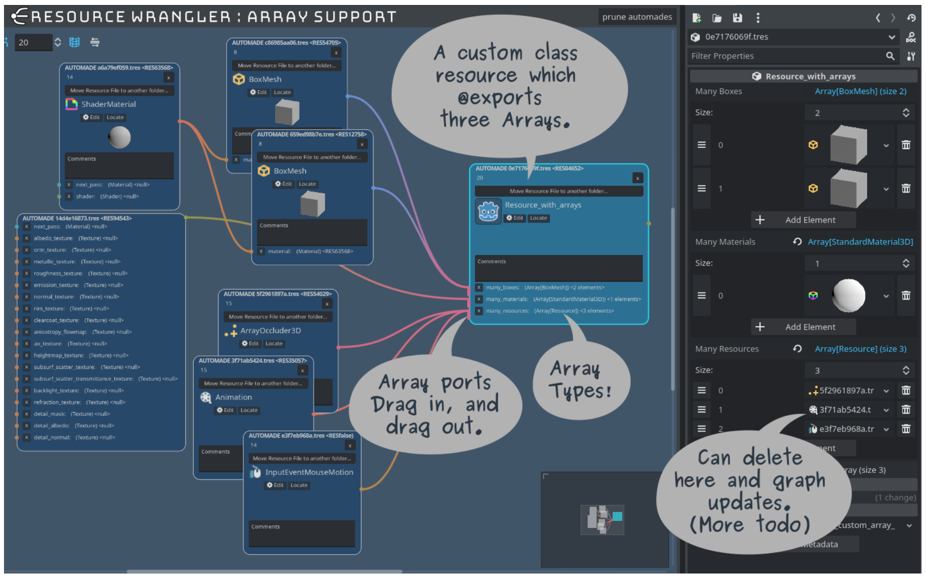

RESOURCE WRANGLER : ARRAY SUPPORT
A custom class resource which @exports three Arrays.
Array ports Drag in, and drag out.
Array Types!
Can delete here and graph updates. (More todo)

Figure 3: July 15, 2023

Arrays of Resources

You can use Arrays of *typed* resources, e.g. @export var A = Array[Resource], in your custom resource classes. When you instance one, it will draw with slightly larger ports (to indicate where the arrays are). You can add many noodles into one array port and dragging out of it will allow you to make a new resource instance that gets appended to the array.

If you want to see individual nodes from within an array, then look in the **Inspector**—you can **drag and drop** any of them to the graph to see that resource. If you connect that node by noodle to the array port, it *will not append* because it's *already* in your array.

Settings

Turn on Advanced Settings and look on the left-bottom for "Resource Wrangler". In there you will find the settings where you can change some paths and stuff.

Custom Resource Nodes

Basic Resource Nodes

Look in "res://addons/resource_wrangler/components/nodes/basic_nodes/"

These scripts show how you can make your own basic Resource nodes. Try drop any of them onto a board to see what happens.

1. "custom_basic_resource.gd" shows a basic Resource.
2. "custom_basic_resource_to_test_missing_class_name.gd" Shows what can go wrong without a class_name keyword.
3. "custom_basic_resource_array.gd" Shows how to use Arrays.

Extended Nodes

See "res://addons/resource_wrangler/components/nodes/extended_nodes/extended_node.gd" for the example.

If you want to 'glue' extra controls onto a node, you can use this process:

Code at the top

```
@tool #make sure this is used
class_name ExtendedResourceNode # Your name here
extends DbatResourceBase #and then extend this class
```

TIP: If you get weird errors like Resource has no method "show_node_gui" then restart the project. Godot is iffy like that.

Your Exports

As you please:

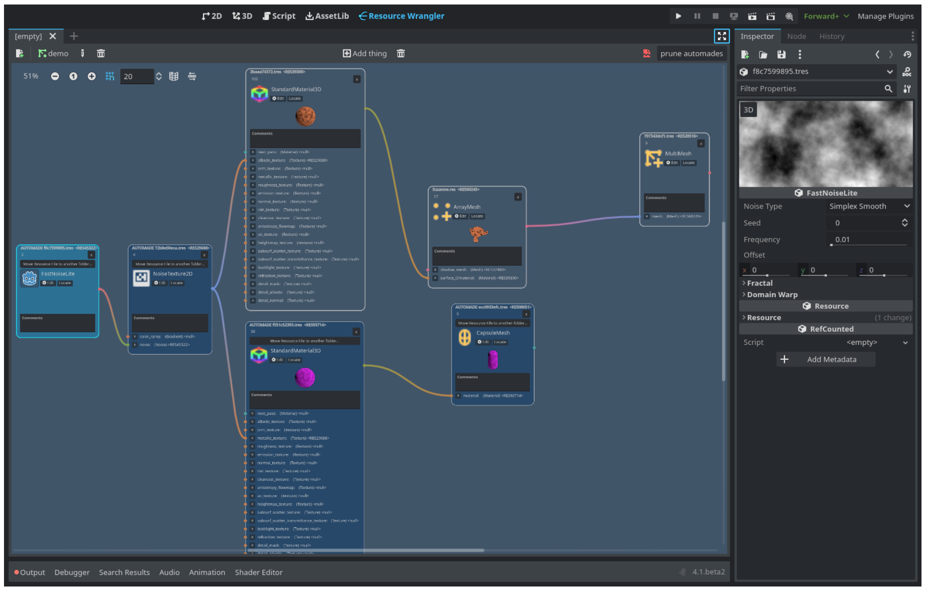
```
@export_group("Meshes")
@export var my_meshes:Array[Mesh]
```

Supply a Scene and Override show_node

```
## This is a scene that is your gui
const SCENE_PRELOAD = preload("sample_extended_ui.tscn")

## Now implement this func:
## Calls parent which handles adding the gui
## Then does the necc to show the data in the gui
## _nop just means no operation, i.e. ignore it; leave as-is
func show_node_gui(graphnode:GraphNode, data: Dictionary, _nop) -> void:
    pass # Actual body up to you..
```

Screenshots



FastNoiseLite

Noise Type Simplex Smooth
Seed 0
Frequency 0.01
Offset x 0 y 0 z 0
Fractal
Domain Warp
Resource
Resource (1 change)
RefCounted
Script <empty>
Add Metadata

Figure 4: July 3rd, 2023