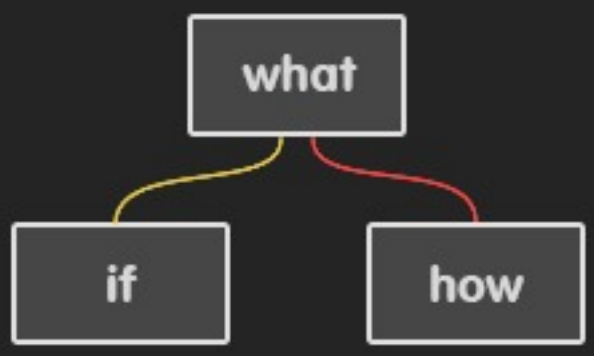
nodeCanvas



Behaviour Trees • FSM • Dialogue Trees

Hierarchical Node Graph Framework

For more thorough documentation please visit

www.nodecanvas.com

What is it...

NodeCanvas is a visual node graph editor to create Artificial Intelligence behaviours using Behaviour Trees and State Machines with ease and offers much power to both designers and programmers to work together in an efficient and flexible way.

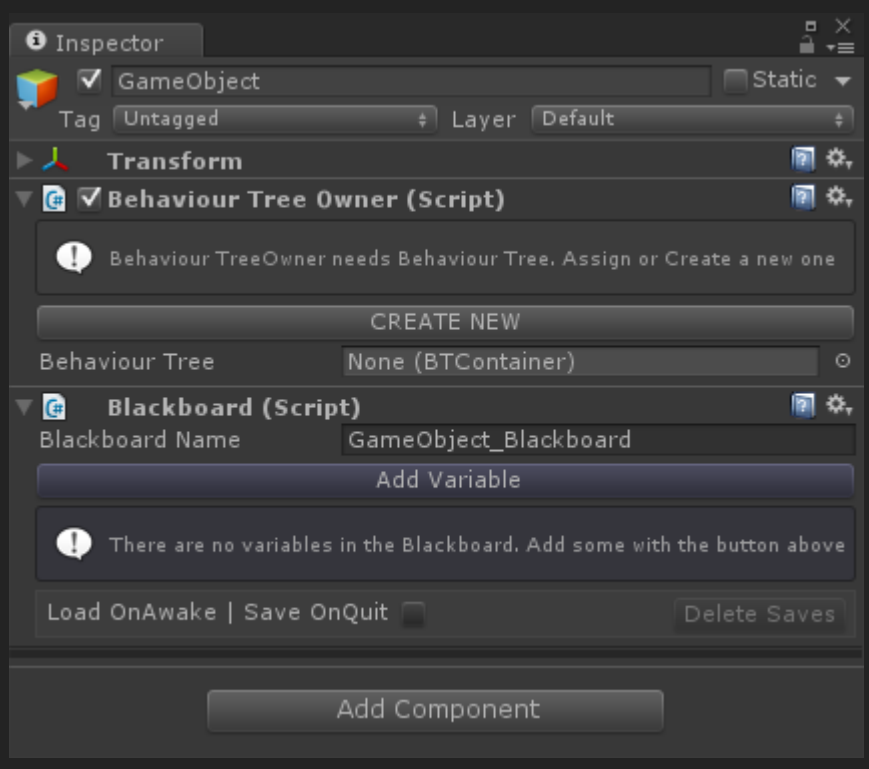
As an added bonus NodeCanvas also comes with a Dialogue Tree designer based on the same foundations of the framework.

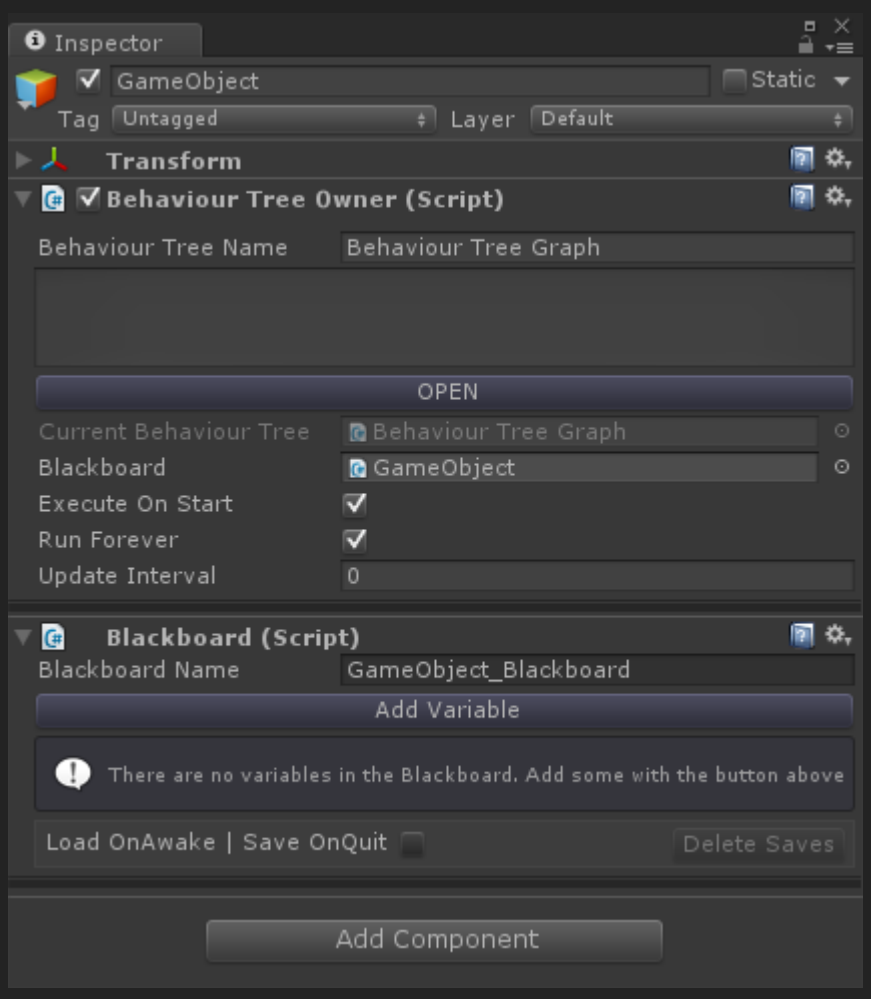
Last but not least you are also able to create your own such systems by extending the NodeCanvas framework as a whole, but that could be more advanced.

In this Quick Start guide we will just go through creating a very simple Behaviour Tree just for the shake of explaining some key concepts. By no means this is a real tutorial.

Let's Start...

Place the 'Behaviour Tree Owner' Component on any game object you'd like to behave based on a Behaviour Tree system. Click 'Create New' on that added component to create a new Behaviour Tree.

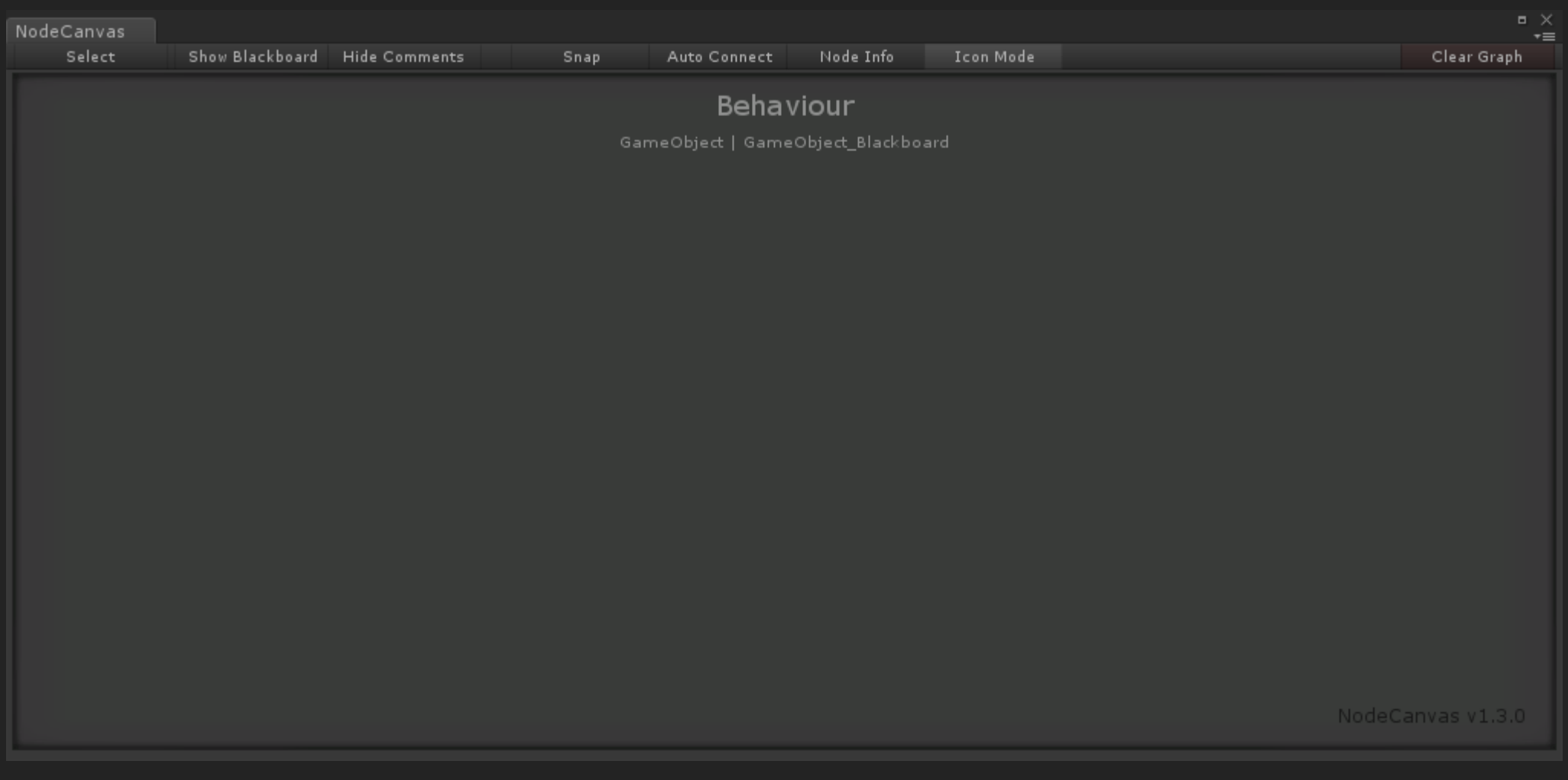




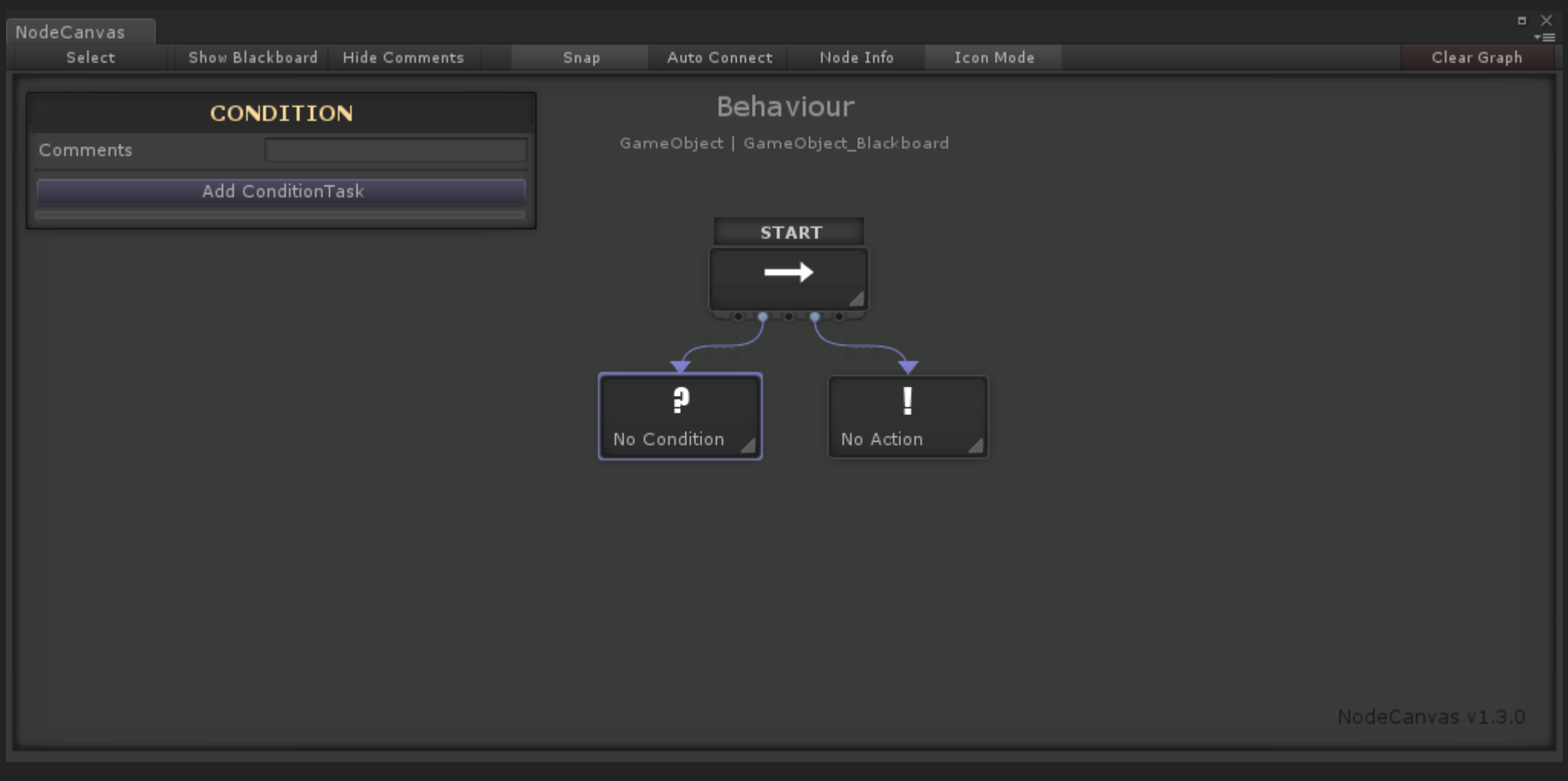
You will notice that a Blackboard Component is also automatically been added.
Blackboards are used to store variables and can be used from any action and condition tasks within the system to communicate data between one another. You can assign another one if you like say for example to have a common blackboard between two or more different AIs.

The agent selection field, let us select a Component reference that will be passed through the graph, down the the nodes of that graph and at last down the the Action and Condition Tasks of those nodes that have such Tasks. So for example you can select a script you may have on the game object and receive it on an Action or Condition directly along with the blackboard. But, that's totally optional to use and you may as well leave it as is for now. For most purposes you can consider the word 'agent' to refer to the object that will perform something. (aka executor, operator etc.)

Leaving everything as is, click 'OPEN' to edit the Behaviour Tree.



Right clicking on the canvas, will show a menu of all available nodes for this system (in this case Behaviour Tree). Selecting any, will add it in the canvas at the mouse position.



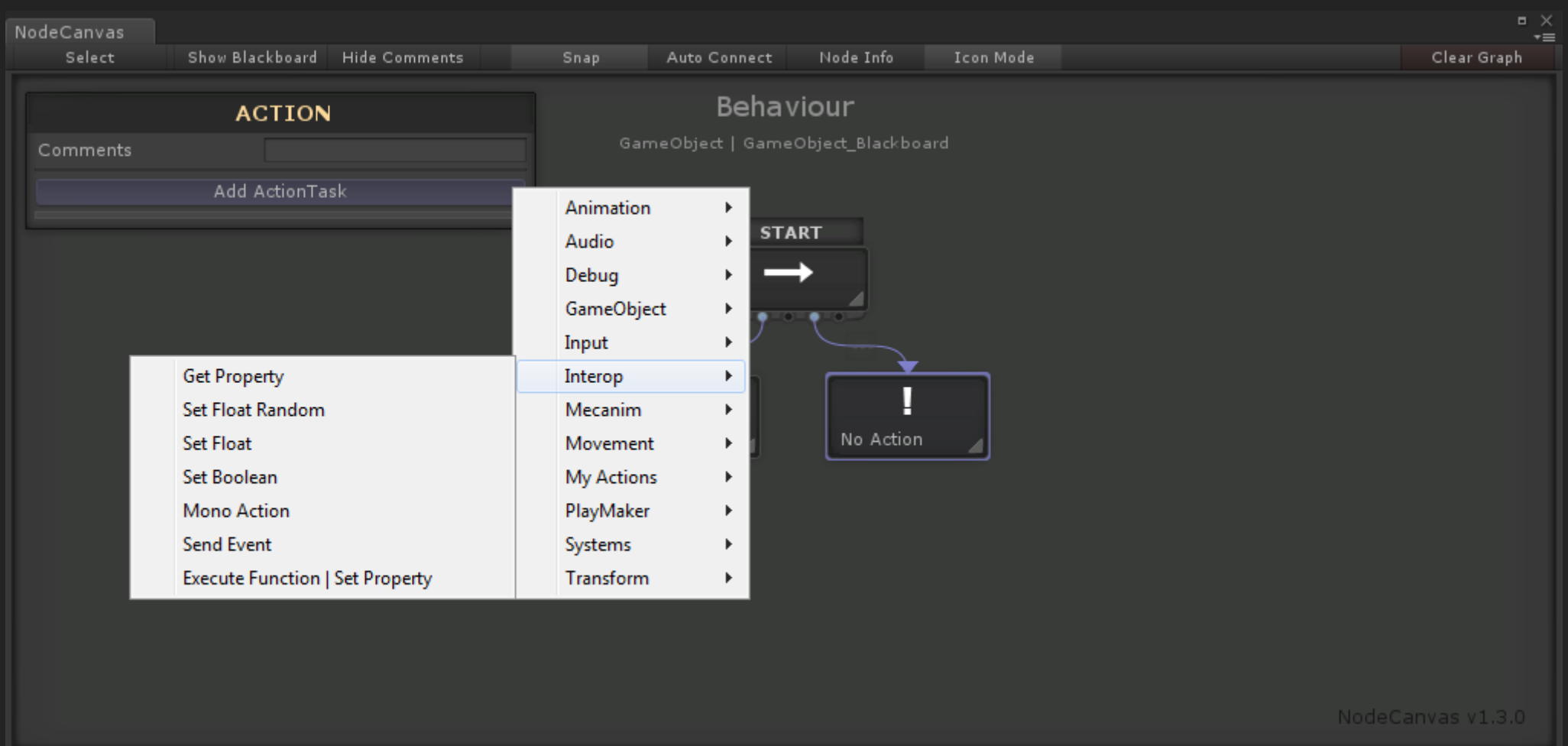
You will see that the first node been added to the canvas is marked as **Start**. That means that this will be the first node to be executed. You can specify another if you like by right clicking on a node and select ‘Make Start.

Before we continue, here are the editor controls:

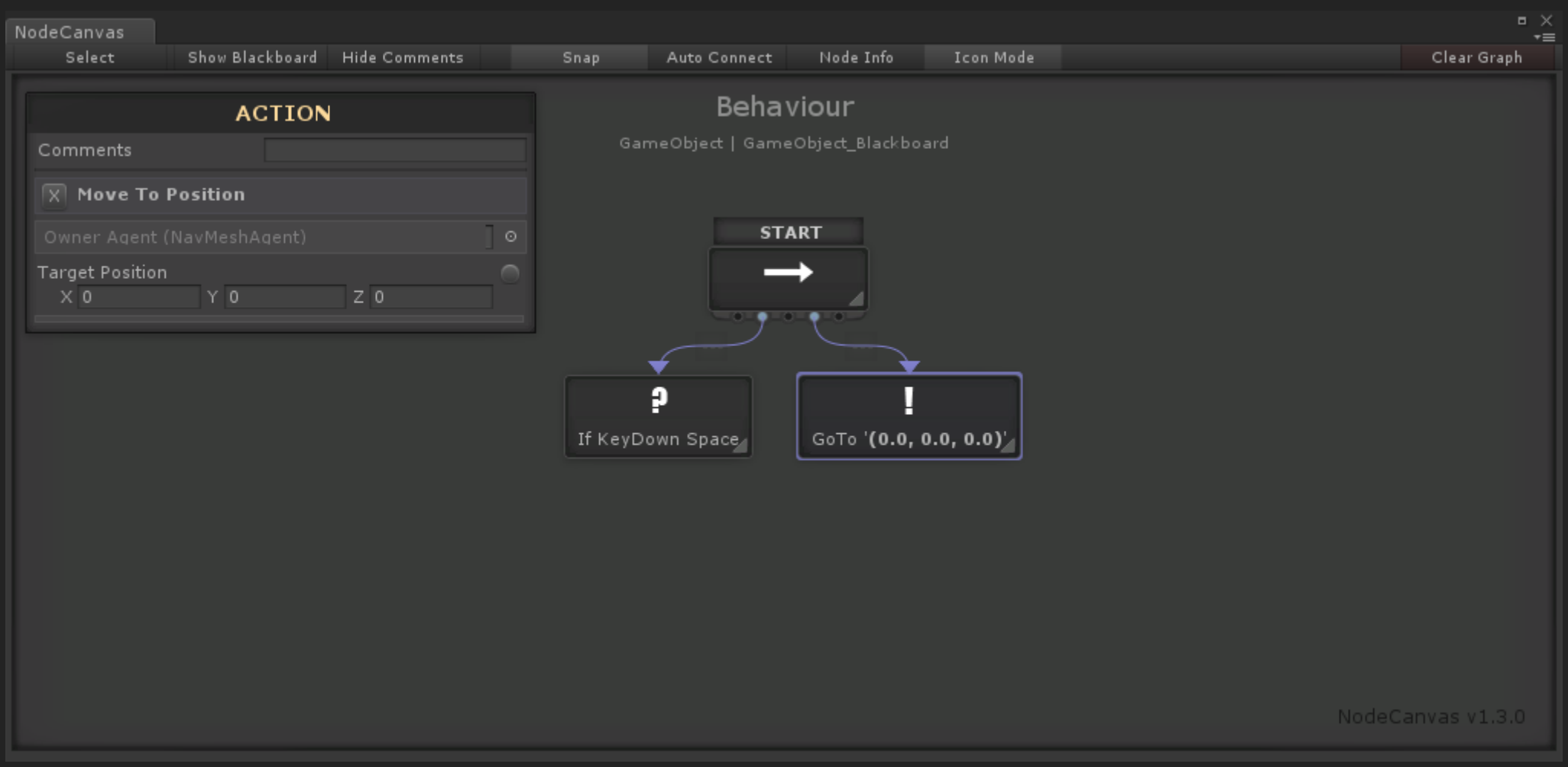
- To connect nodes together click and drag from an empty node port and release on top of the target node.
- You can disconnect nodes by right clicking on a connected port.
- You can delete nodes or connections after selecting them, by hitting ‘Delete’.
- You can pan all nodes around by middle click and dragging.
- **You can pan a node and all of it’s children together by CTRL + Click and dragging a node.**
- Pressing ALT + Q will contract all nodes to their minimum size.

Note that Behaviour Tree nodes will get auto sorted from left to right

Condition and Action Tasks are being assigned to nodes that need them. In Behaviour Tree systems, the Condition Node requires a Condition Task and the Action Node an Action Task for example. This allows for much flexibility in design.



By pressing ‘Add Action Task’ (or Condition Task) a context menu of all available Action Tasks in the project will show up categorized respectively. When a task has been added, the node will read it’s information as of what it will do or check, as well as the node inspector on the top left will show that Task’s controls…



Note that you can toggle the ‘Node Info” at the top toolbar to display what each node does on the inspector panel.

You can also toggle ‘Icon Mode’ on and off if you prefer to work with text nodes instead of icons

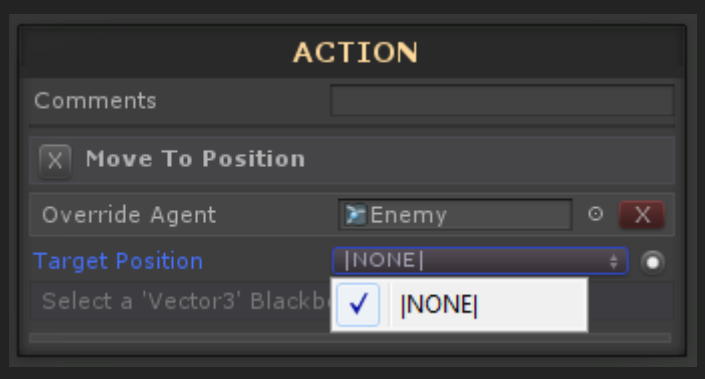
You can enable ‘Snap’ to snap the nodes verticaly and thus align them better.

You can enable ‘Auto Connect’ to connect any new node to the selected one if any.

First of all you can delete the current assigned Task by the "X" button and assign another one, without the need to delete the node of course. That allows to use placeholder actions and conditions while keeping the design non destructive until an action is implemented for example by the coder.

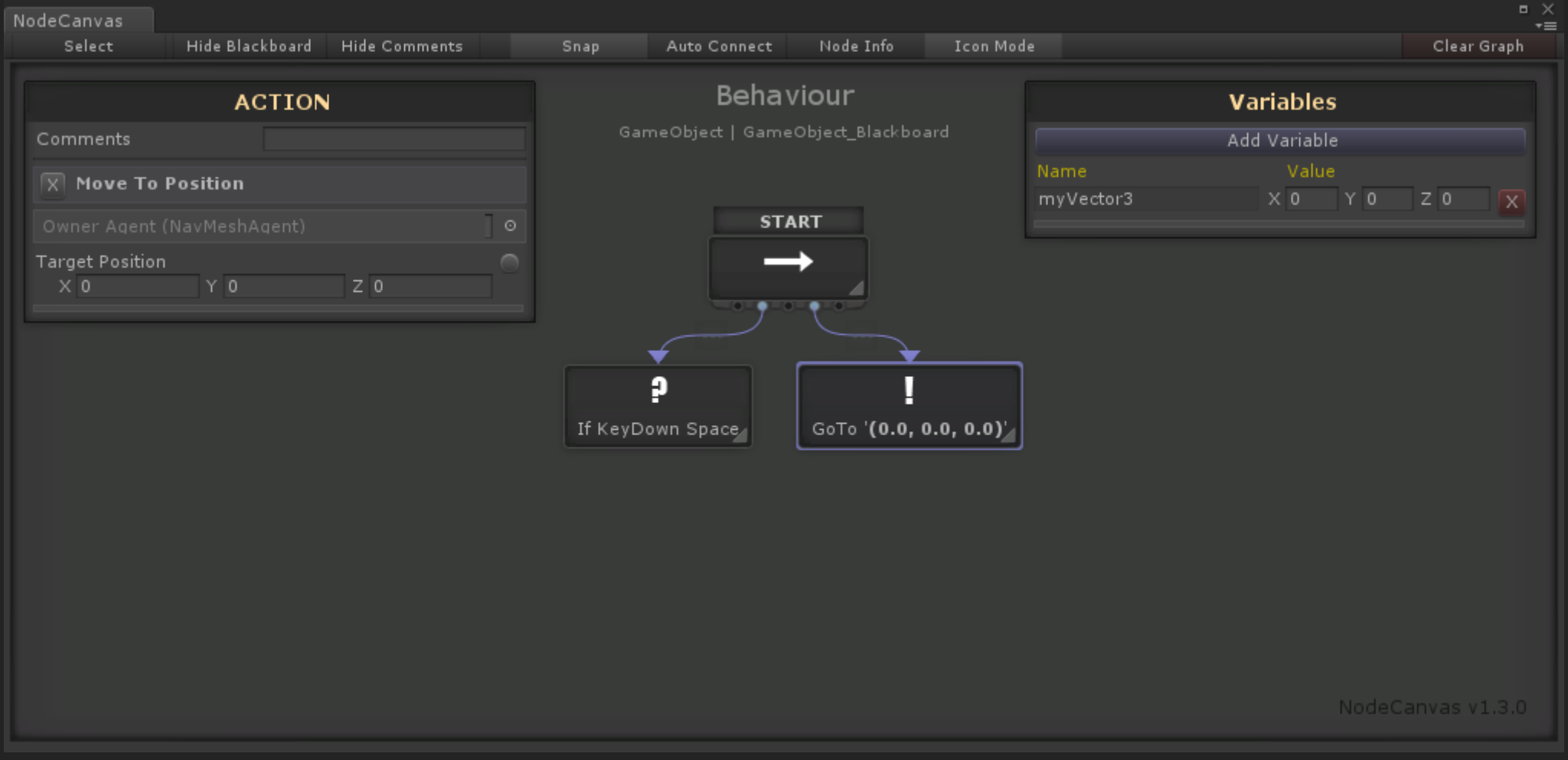
The first thing you will notice is a field reading 'Owner Agent' and then a Type. This means that the Task (in this case Move To Position) requires a Component of the type specified to be present on the Owner Agent game object, or in other words the game object that we've added the 'Behaviour Tree Owner' for now. If we want the action to be performed by another 'Agent', we can override this by picking a new one through the object picker.

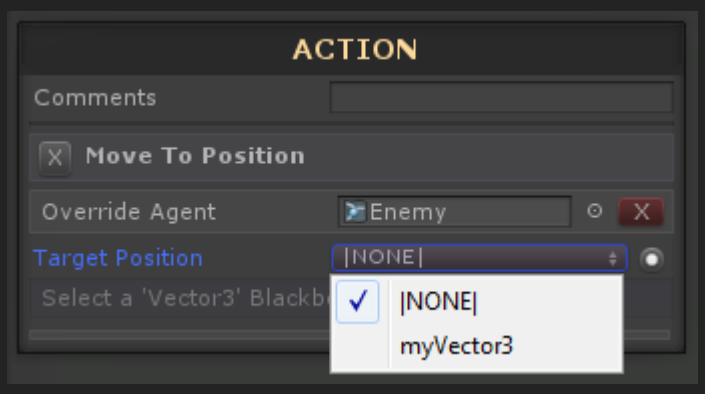
The next thing to notice in this action will be the Target Position field which comes with a radio button on the right...
This is a special kind of field, that allows us to either input a value directly or select a blackboard variable to get the value from.

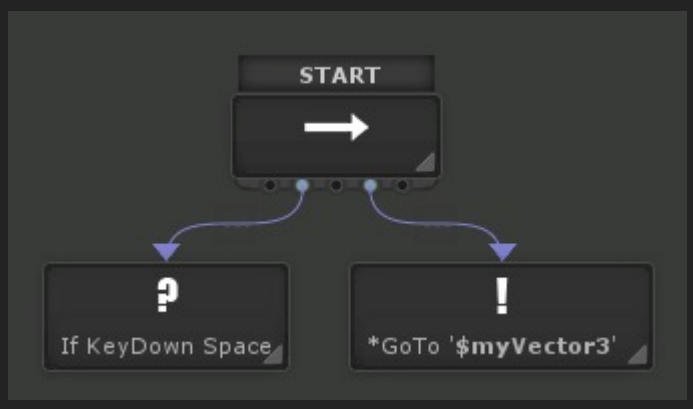


Clicking on that radio button will change the control to allow us to select a blackboard variable of the type required. Currently we have no such variable on the blackboard, so let's create one...

We can either go back to our Graph Owner by pressing 'Select' on the top left of the editor, or click the 'Show/Hide Blackboard' on the top of the editor to show the blackboard reference within the canvas. Clicking Add Variable and selecting Vector3 will add such a variable in the blackboard. We can of course specify a name as well.

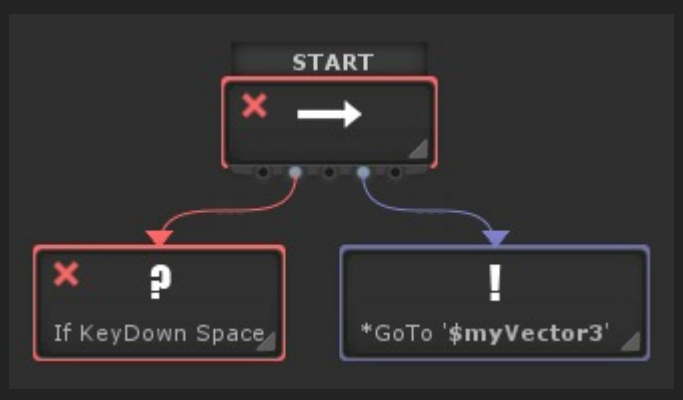


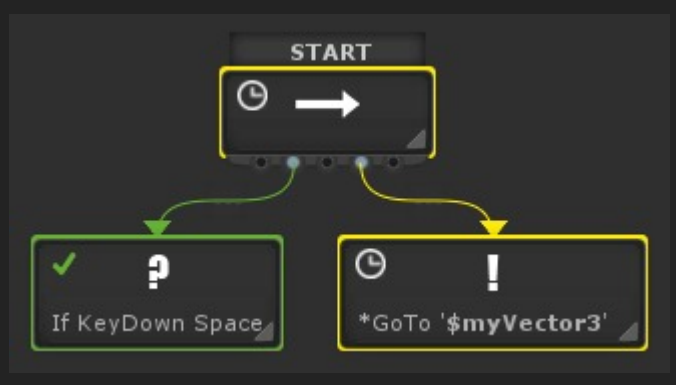




Back to our action settings, the variable will now show up to be selected. You will also notice that the node will now read that selected variable name instead of the actual value with an '$' symbol on the start and in **bold** to designate that this is a blackboard variable.

So, we can now hit play and watch it happen...

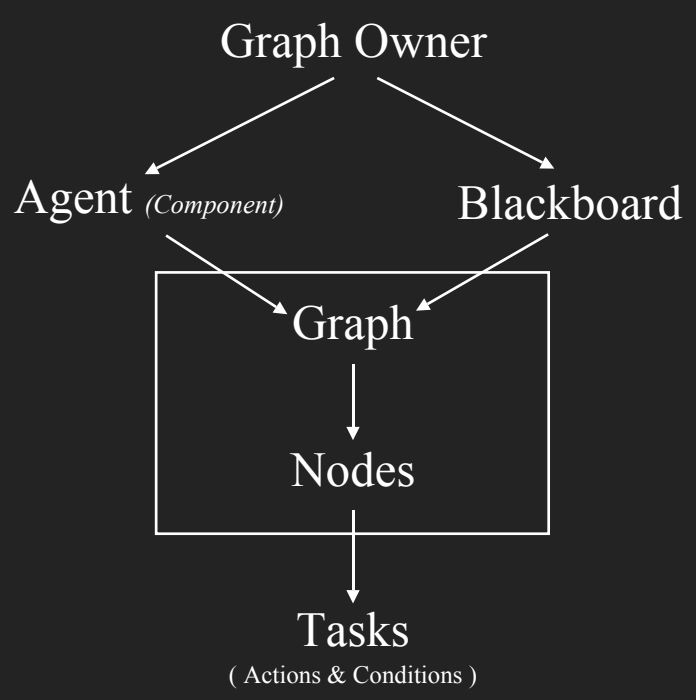




You will notice that the nodes will now read their return status along with an icon that represents that status, as well as the connection colours which also represent that status.

- So in this example the Sequencer will check the first child as it should, which will return Failure (red) since the space key is not pressed down.
- As soon as the space key is pressed down, the Condition will return Success(green) and as such the Sequencer will continue to the next child as it should.
- Since the Action 'Move To Position' requires some time to complete, it will enter a running(yellow) state until it is finished at which point it will return Success and the Behaviour Tree will reset and start a new cycle.

That's it for this very brief Quick Start to NodeCanvas and Behaviour Trees, but of course there are a lot more to it! Please do visit www.nodecanvas.com to read much more thorough documentation and learn how to create you own Action and Condition Tasks as well as how to use the State Machine and Dialogue Tree systems included.
Please understand that this is a very brief quick start guide.



www.nodecanvas.com