For more thorough documentation please visit

www.nodecanvas.com

What is it...

NodeCanvas is a visual node graph editor to create dynamic and interesting 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 powerful Dialogue Tree designer based on the same foundations of the framework.

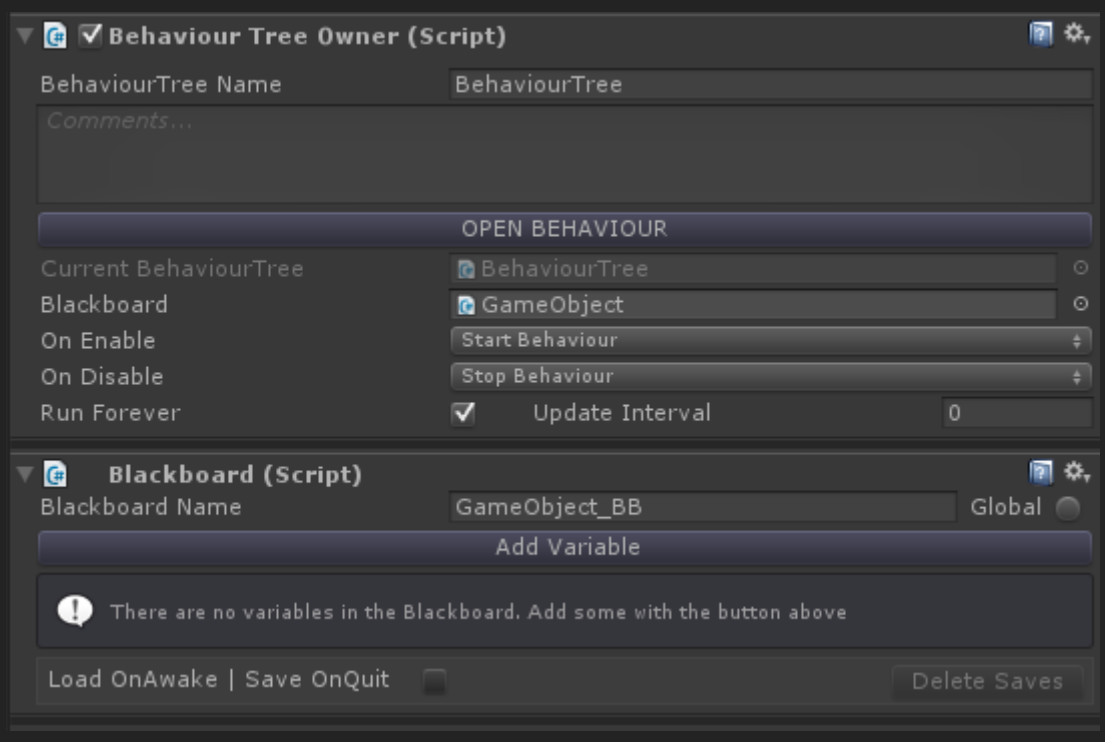
Last but not least you are also able to create your own such systems by extending the NodeCanvas framework as a whole, but that could be more advanced.

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. The BehaviourTreeOwner is responsible for executing a Behaviour Tree and is not the Behaviour Tree itself.

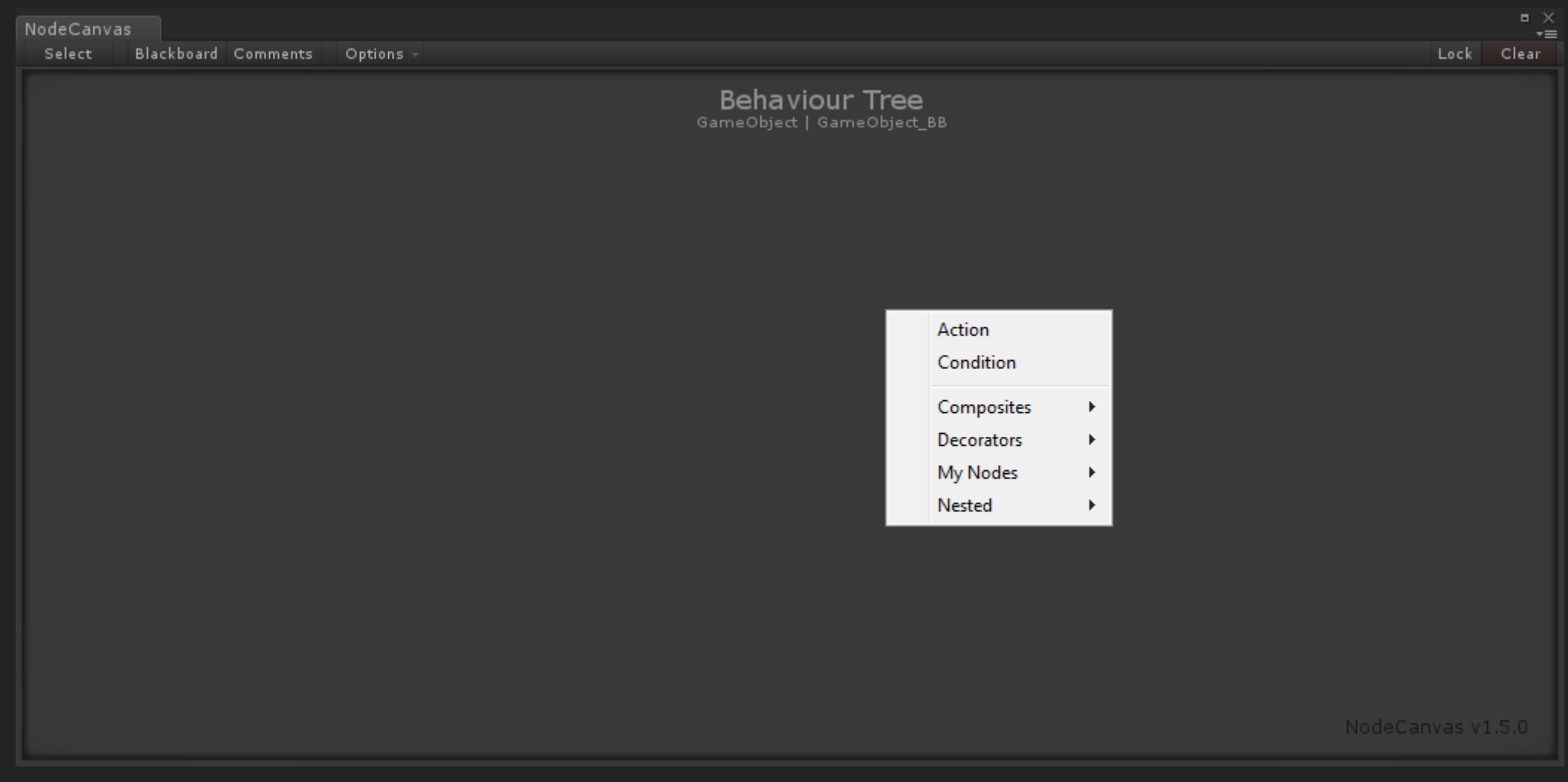
As such, it must be assigned a Behaviour Tree. Click 'Create New' on that added component to create and assign one automatically. Alternatively you could assign an already created Behaviour Tree.



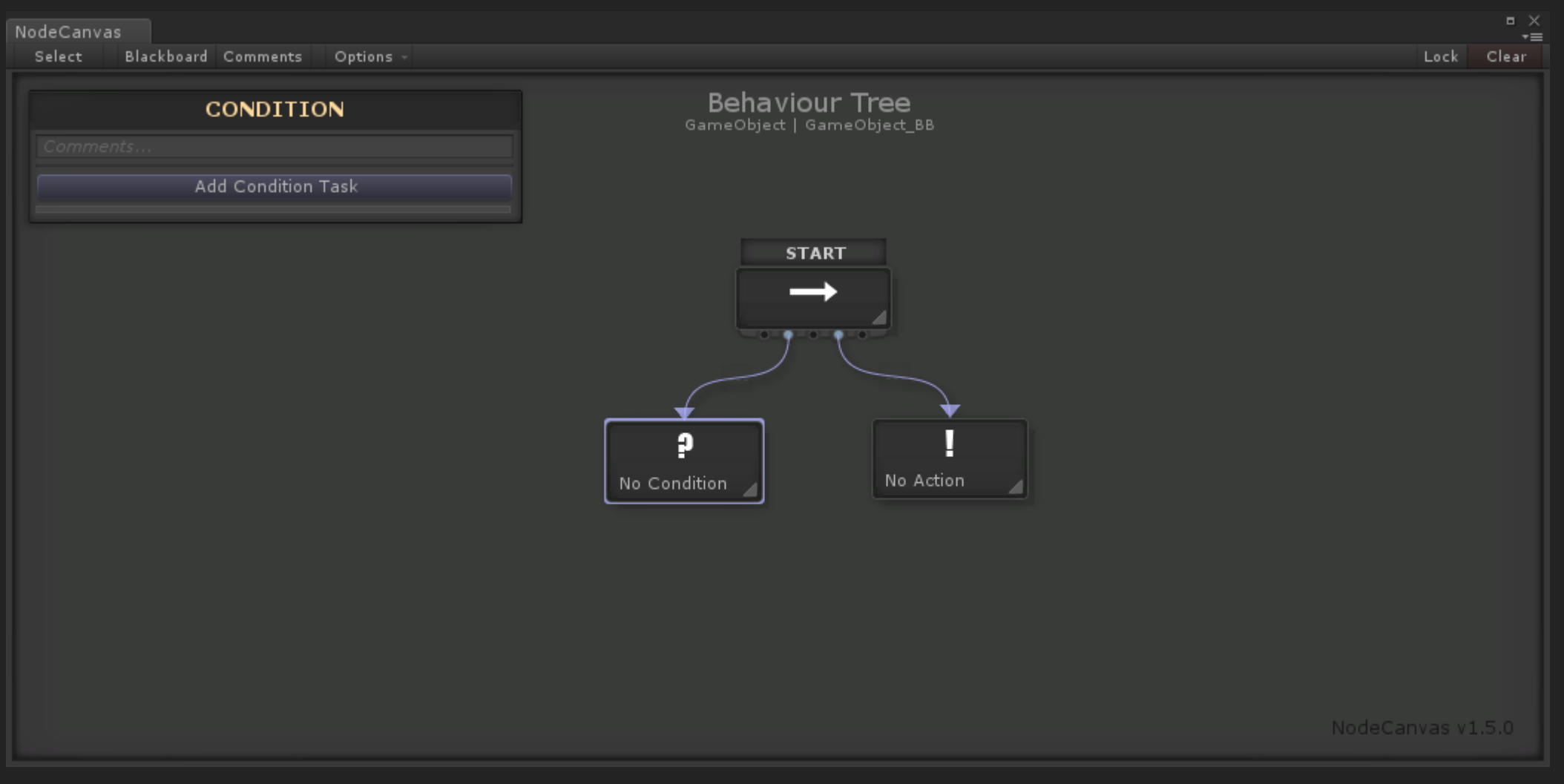
You will notice that a Blackboard Component is also automatically been added when the BehaviourTreeOwner was added. Blackboards are used to store variables and can be used from any action and condition tasks within the system to communicate data between one another. You can assign another blackboard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

You can select an option for what happens OnEnable as well as what happens OnDisable. By default, OnEnable the behaviour will start and OnDisable the behaviour will stop.

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; the entry point. You can specify another if you like by right clicking on a node and select 'Make Start'.

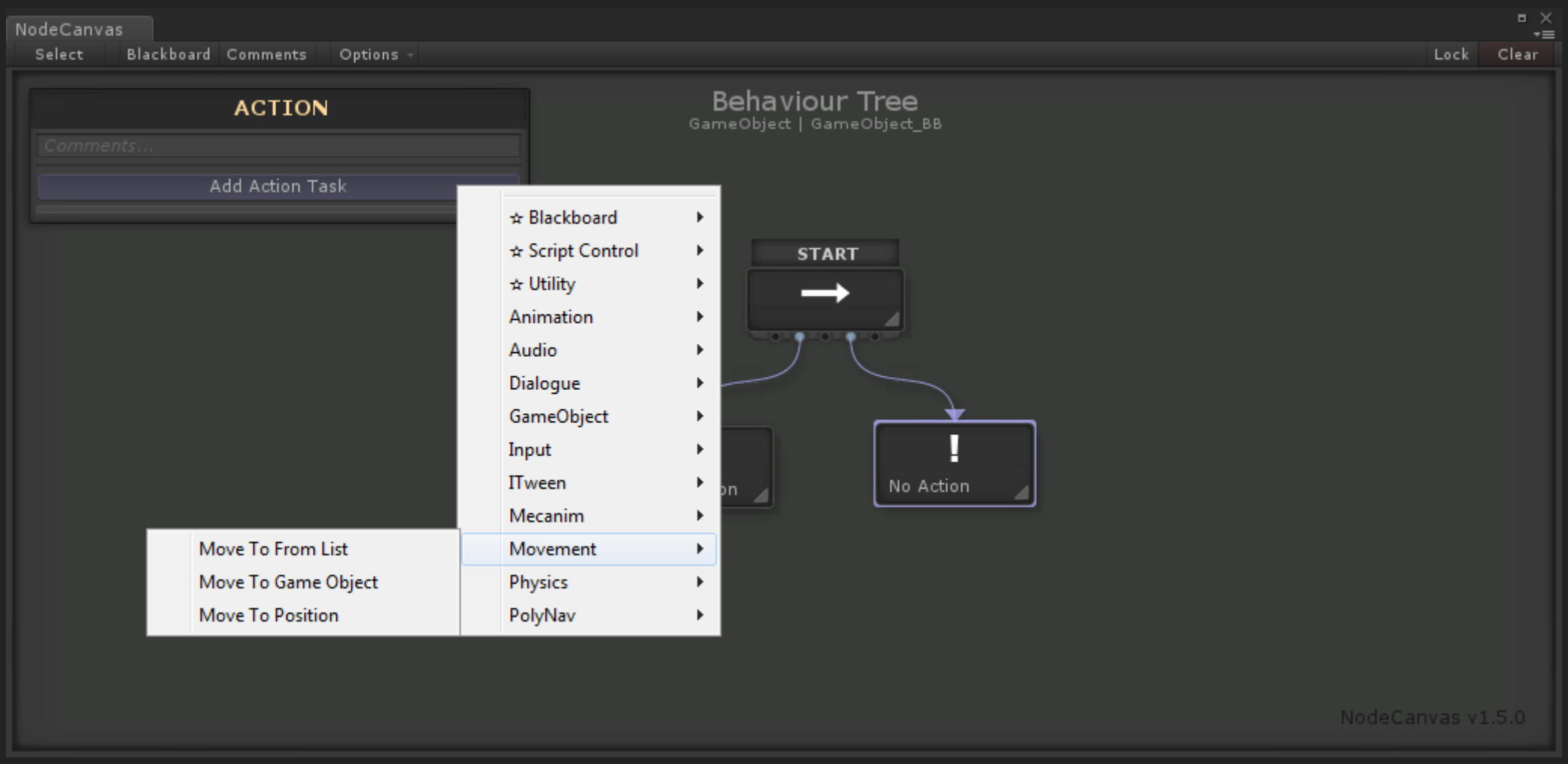
Before we continue, here are the editor controls:

- 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 + Shift and dragging a node.**
- Pressing ALT + Q will contract all nodes to their minimum size.

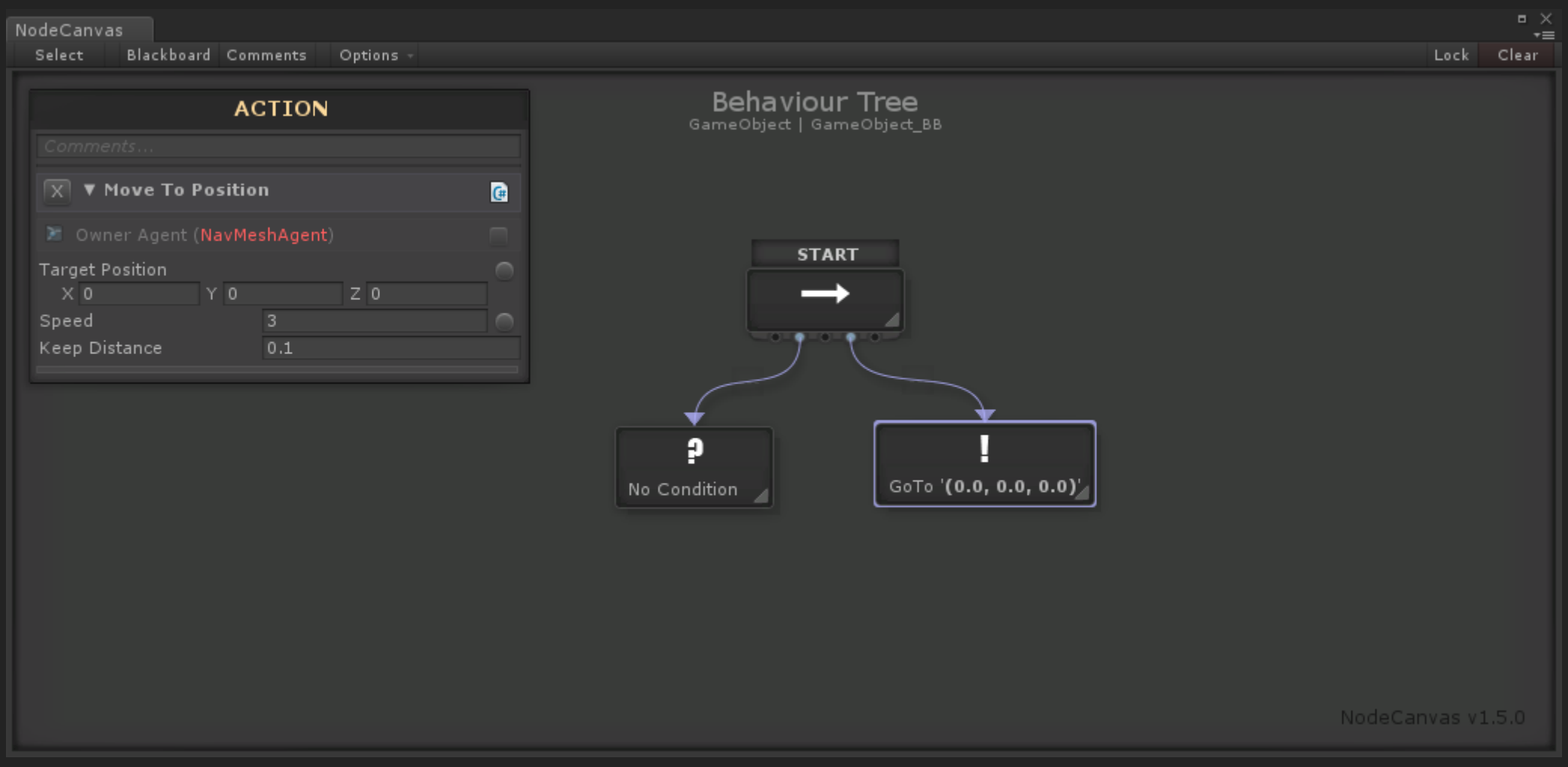
Note that Behaviour Tree nodes will get auto sorted depending on their position to the parent, left to right.

So here we've added a **Sequencer** as well as a **Condition** and an **Action** node. You will notice that when adding an Action or a Condition node we don't specify beforehand the actual action or condition that will take place, but instead we simply adding a placeholder for an action or a condition. So we are free to design the behaviour irrelevantly of whether or not an action or condition has been implemented yet. This is very powerful in the long run.

So these two nodes get assigned an actual Action and Condition, collectively known as Tasks.



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 will show that Task's controls from there after. Here we've added the Move To Position Action Task...



Note that you can toggle the 'Node Info" at the top toolbar Options 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 also in the toolbar Options.

You can enable 'Snap' to snap the nodes vertically and thus align them better.

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

Finally there is a Connection Mode options to choose between Curved or Stepped connection Mode.

Lets take a closer look at the node Inspector now and in regards to the current Action Task assigned.

First of all, we can simply remove the Action Task and assign another without deleting the node. Simply press that "X" button and the "Add Action Task" button will show up again to add another.

Next comes that 'Owner Agent' control which also reads in red the NavMeshAgent component type and has a toggle control at the far right. This tells us that for this action to happen, the agent's game object needs to have that type of component. Right now the 'Owner Agent' refers to our BehaviourTreeOwner we've added at first and since it was a newly created game object, it doesn't have the NavMeshAgent component added. As soon as we add that component, the type will no longer show in red.

But let's do something else. You can override the agent and specify another one to execute this action task by pressing that toggle at the far right.

The control will now change and allow us to specify directly a NavMeshAgent component. But, let's not do that for now and toggle off the override and instead let's simply go ahead and add a NavMeshAgent component to our game object.

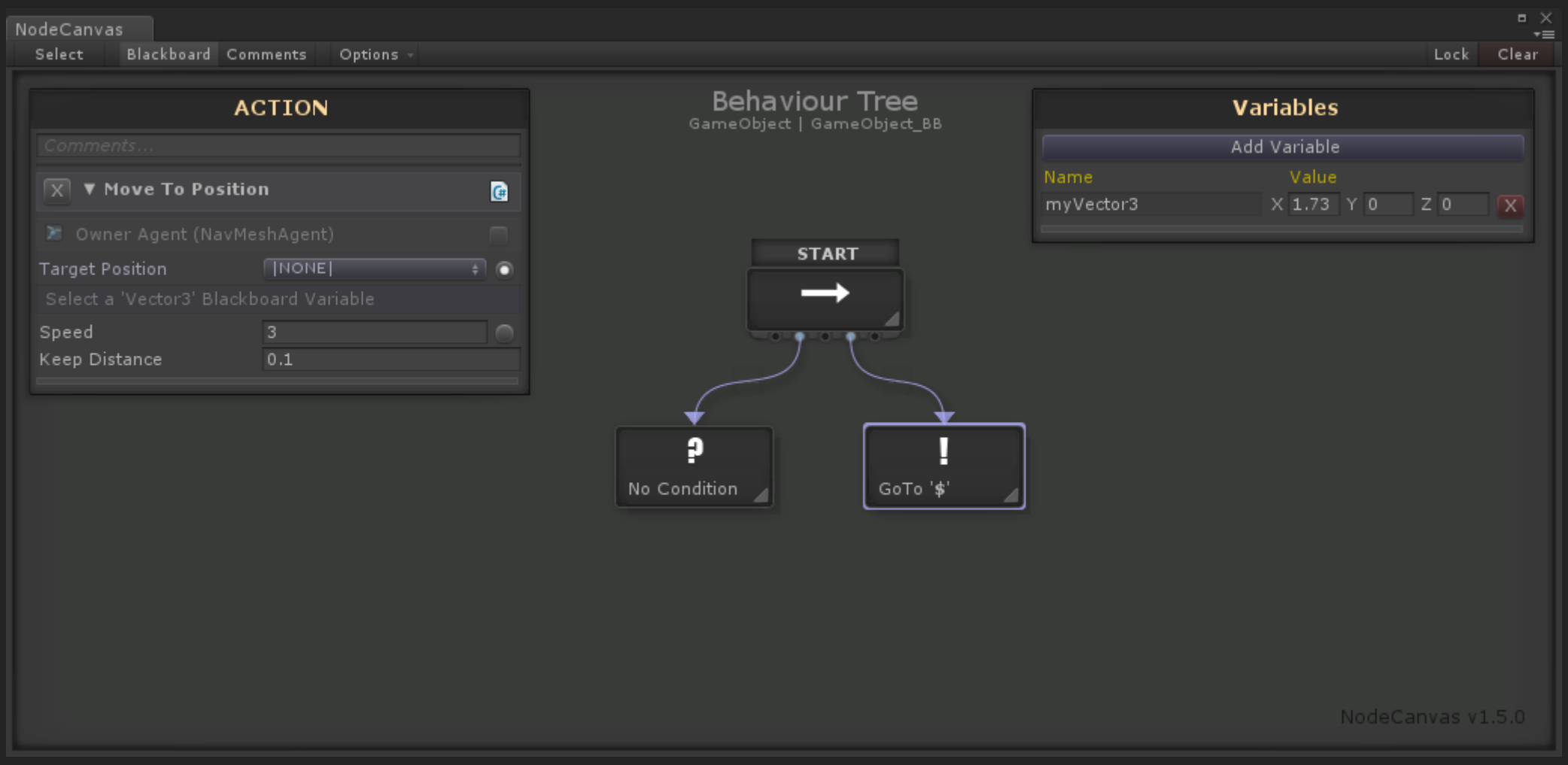
Make sure you've also baked a navigation mesh

Next come the settings for the action. In this case the target position for the agent to move, the speed at which to move and the distance to keep from the target position. The first two of the settings have this radio button on the far right.

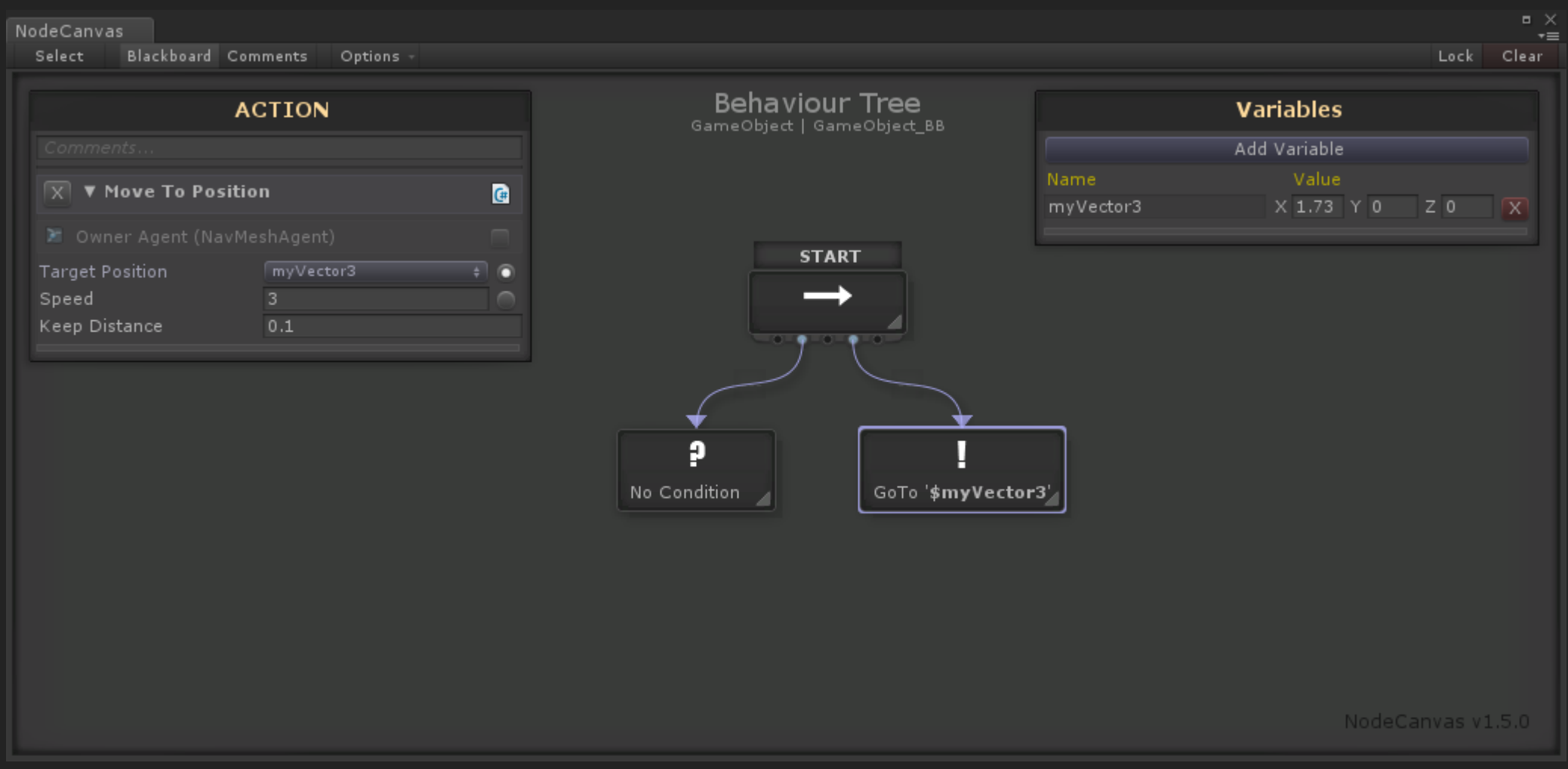
Whenever that radio button is show, it means that it is possible to select a value from the blackboard variables instead of directly providing one. Pressing the radio button for the Target Position will turn the control into this...

The drop down (currently reading |NONE|) will show all available variables on the blackboard and of the type required. Currently we have no such variables on the blackboard so lets create one.

We can either go back to our BehaviourTreeOwner by pressing ‘Select’ on the top left of the editor, or click the ‘Blackboard’ button on the top of the editor toolbar 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.



For the shake of this demo, lets add a Check Keyboard Condition Task to the condition node, so that if we press a key, then the action will happen.

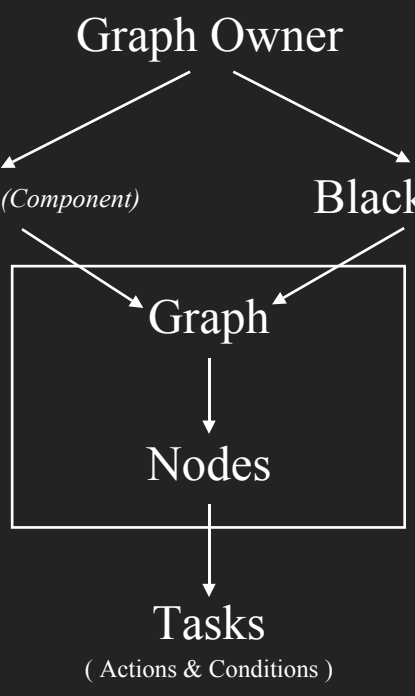
We do this in the very same way we’ve added the action. Select the Condition node, click ‘Add Condition Task’ and select the Check Keyboard condition under the Input category.

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