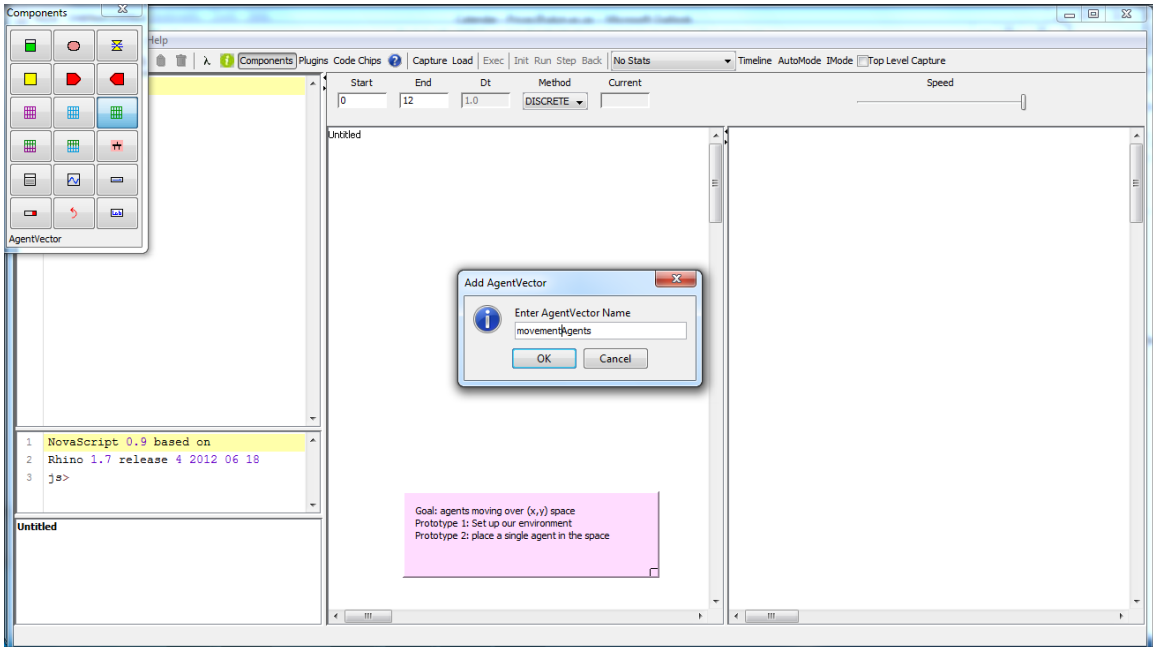
Tutorial of how to make a moving agent in NOVA

Notes taken by C. S. Price during N. Sippl-Swezey's tutorial, July 2014

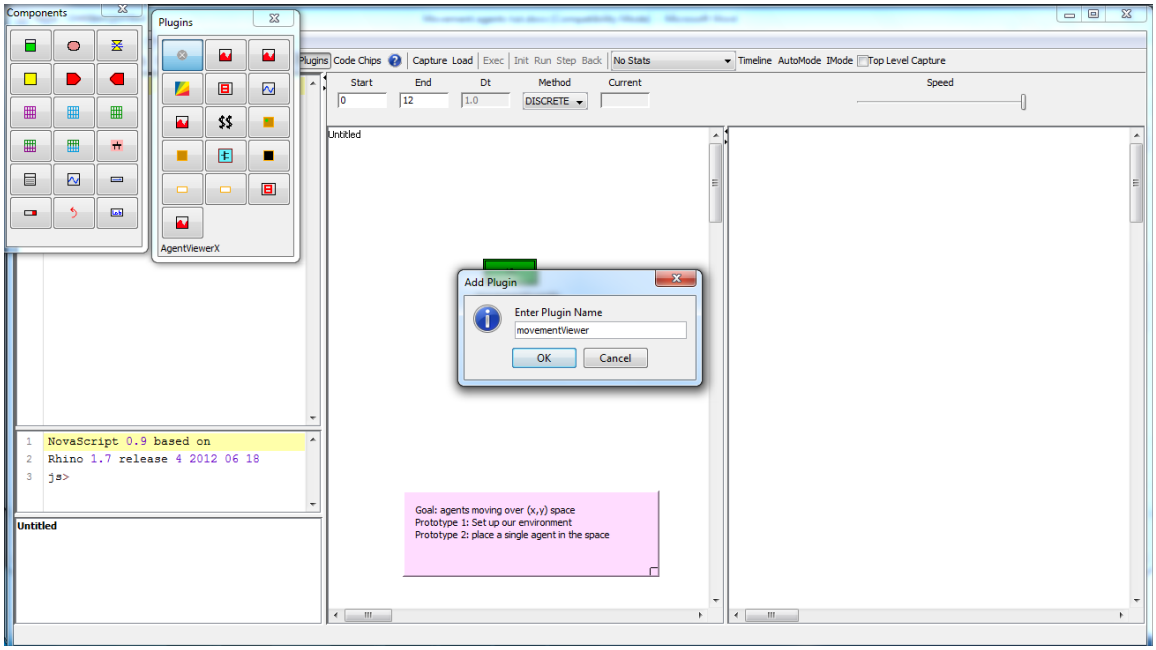
Prototype 1: create an agent environment

Using incremental development, the first prototype is to create an agent environment.

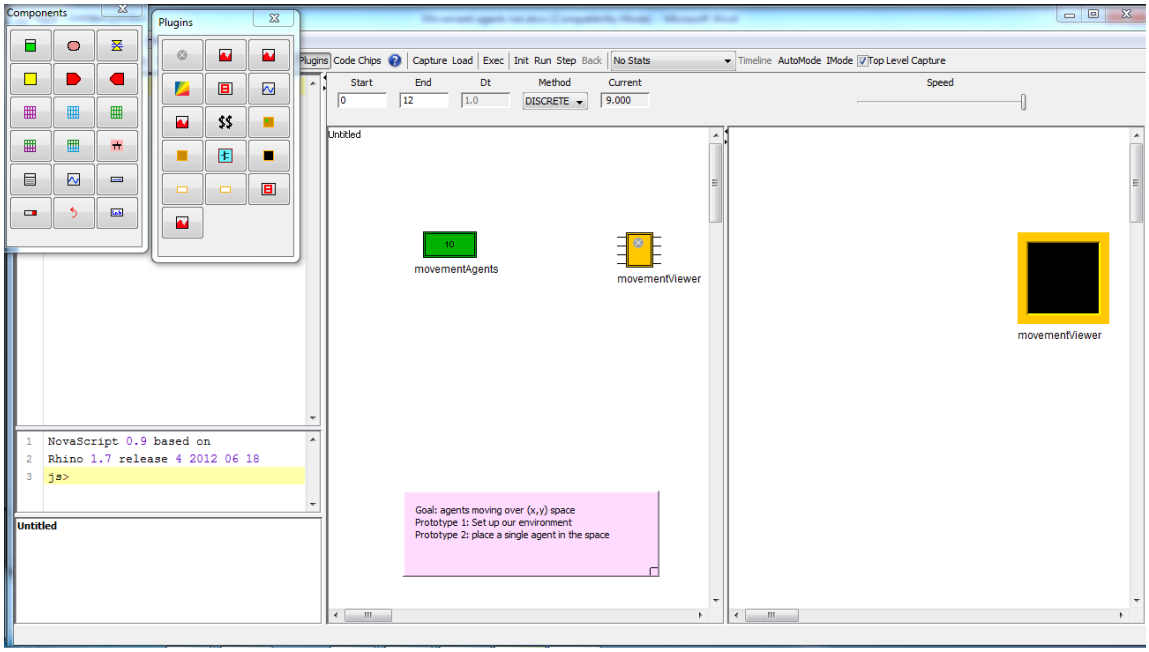
In NOVA, add an agent vector from the Components list:



In plugIns choose AgentViewer:



This completes the first prototype:

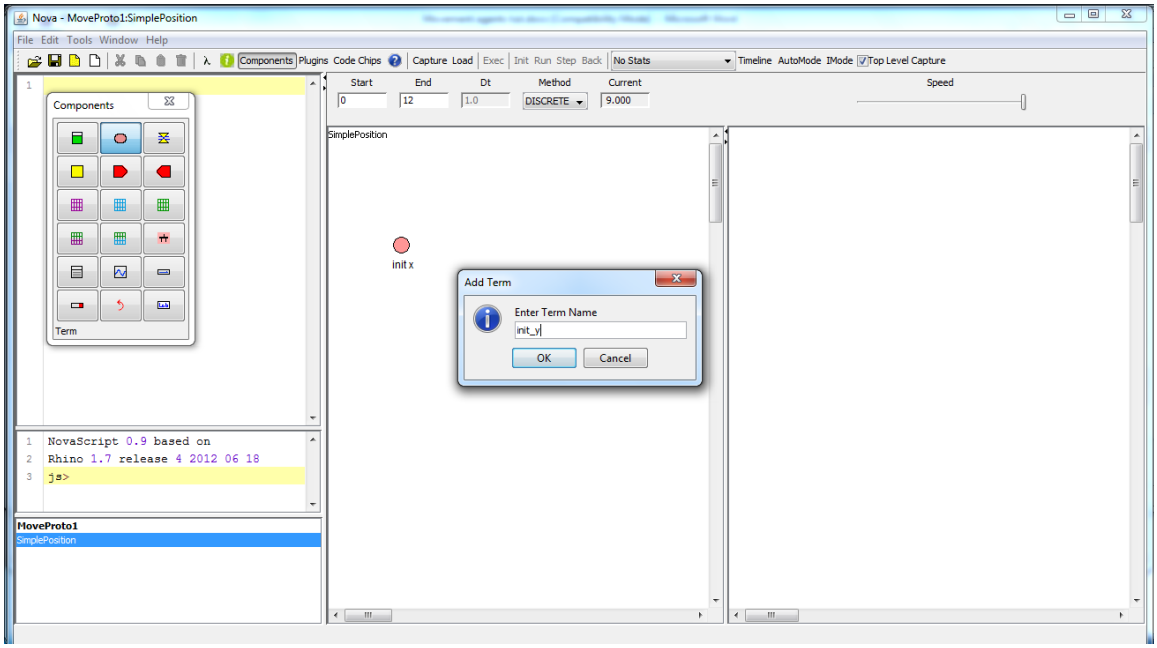


SaveAs MoveProto1.

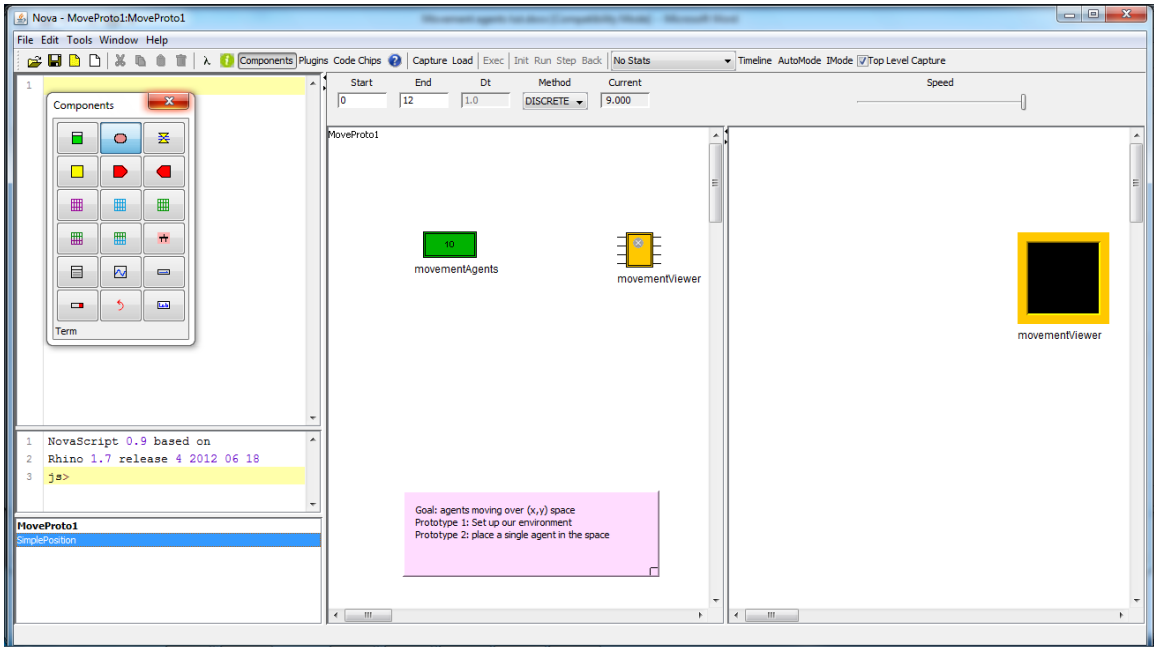
Prototype 2: place a single agent in the space

Now start the next prototype which is to place a single agent in the space.

Add a new submodel: SimplePosition. In the submodel, using the Components menu, add a new term: init_x and another term init_y:



Go back to top model (MoveProto1). Drag an instance of the SimplePosition model and drop over the movementAgents agent vector:



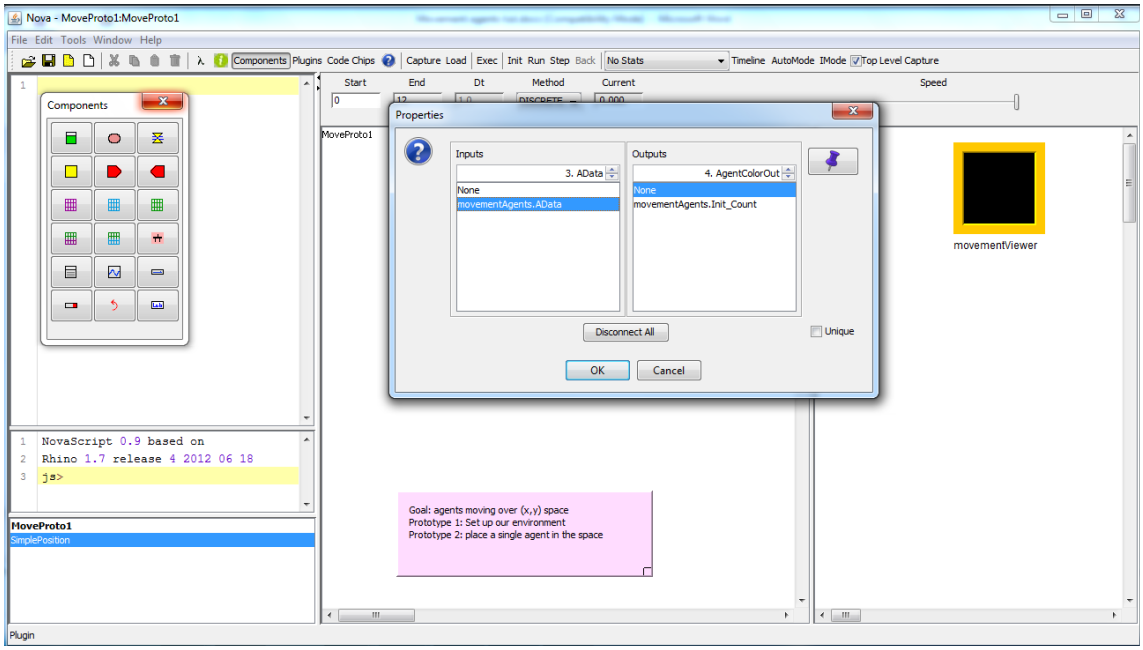
The agent vector should now have 2 pins:

and the agent is now in the space (grid). This is the end of the second prototype (save the model).

Prototype 3: place the agent in the middle of the space

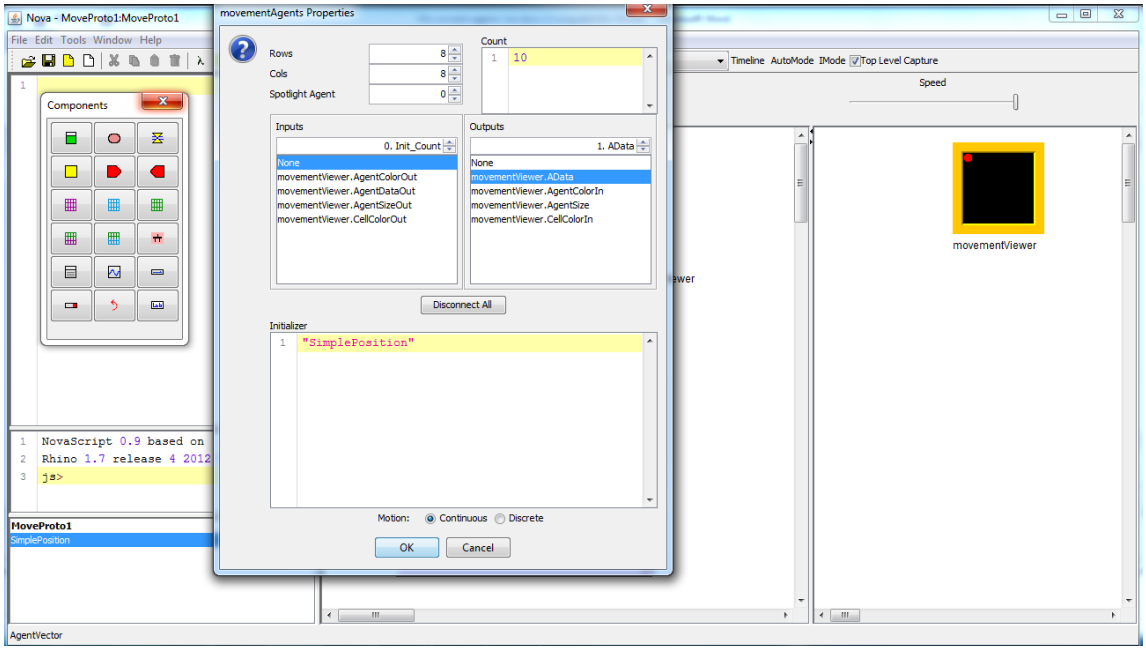
For prototype 3, we want to put the agent in the middle of the space.

Connect the movementAgents to the movementViewer. Then in movementViewer, connect the input of the viewer to the output of the agent:

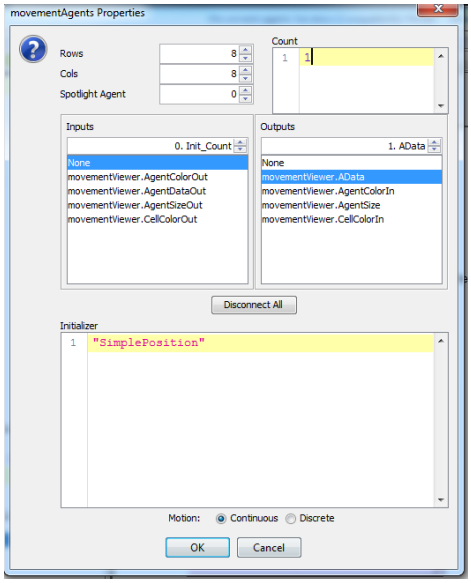


The result should look like this:

By default, the agent vector holds 10 agents (see Count box on the top right hand side).

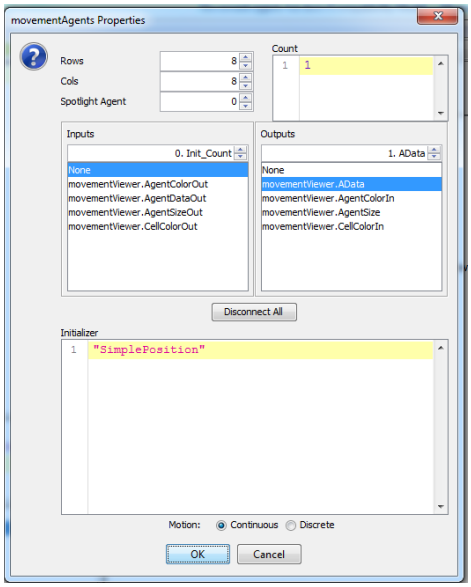


Change this to 1:



The number of agents in the agent vector changes to 1:

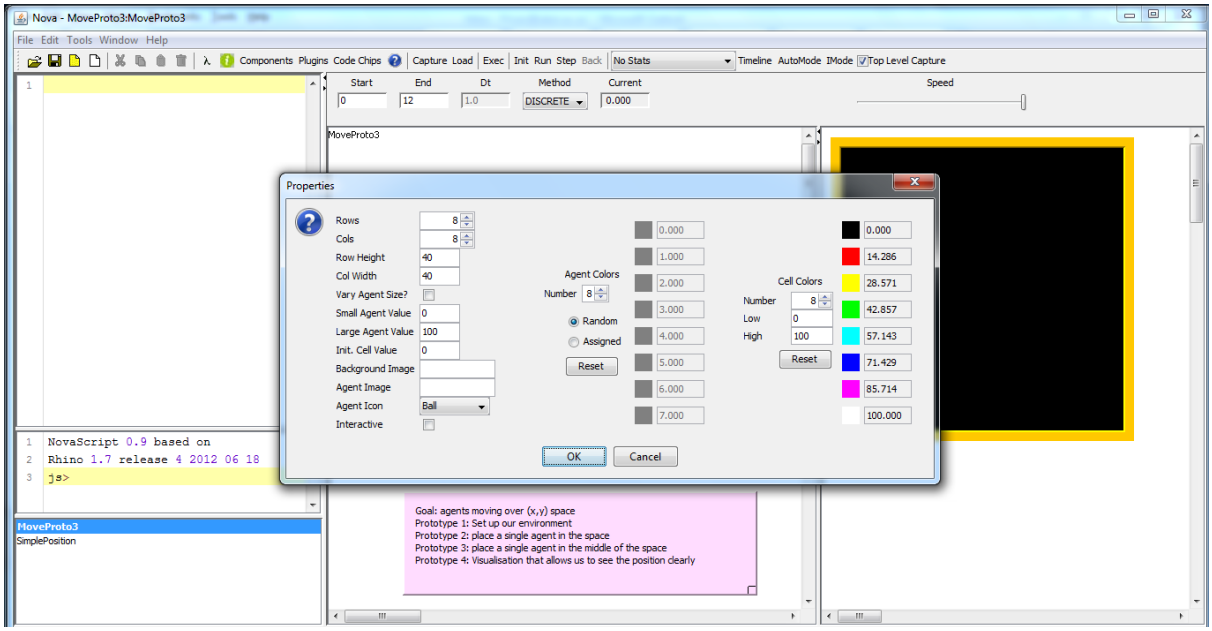
The default agent space is 8x8, as can be seen on the left hand side:



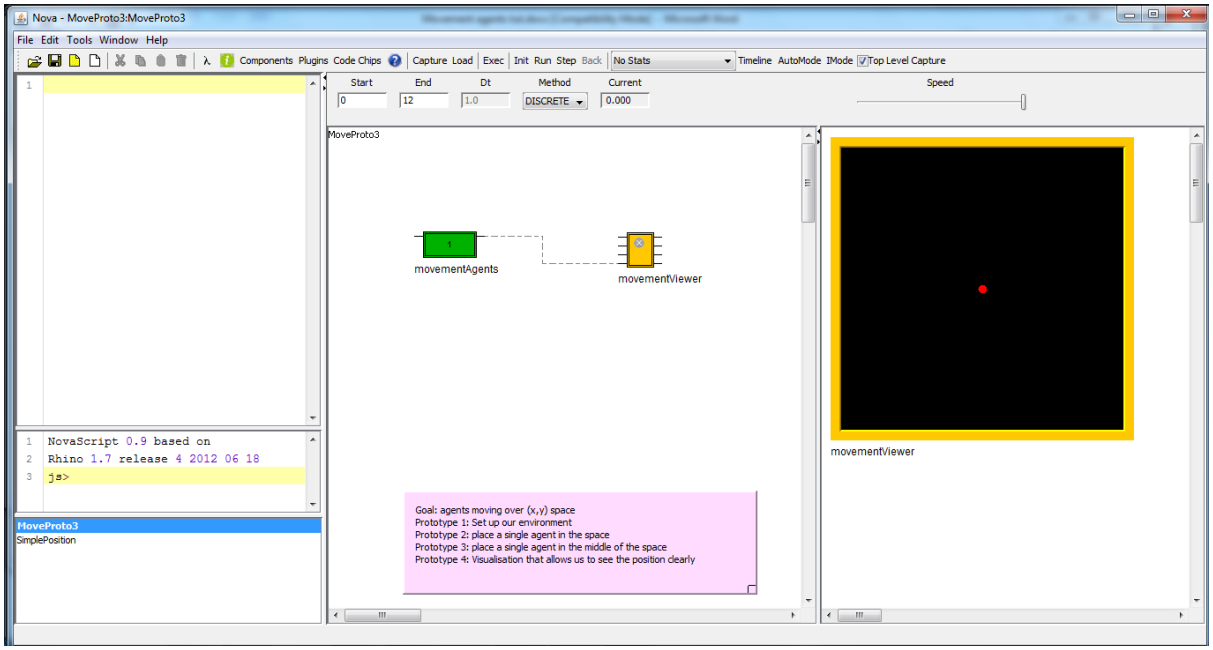
Go to the sub model. Set the initial x and y values to 4 and 4. (Actually, to get the agent right in the middle of the space, these values should be 3.5 and 3.5 – updated later!)

Prototype 4: place the agent in the middle of the space

To see the model clearly enough, right-click on the space (grid)'s yellow boundary:



To get the spot representing the agent right in the middle of the grid, change the init_x and init_y values to 3.5 and 3.5 respectively. The result should look like this:

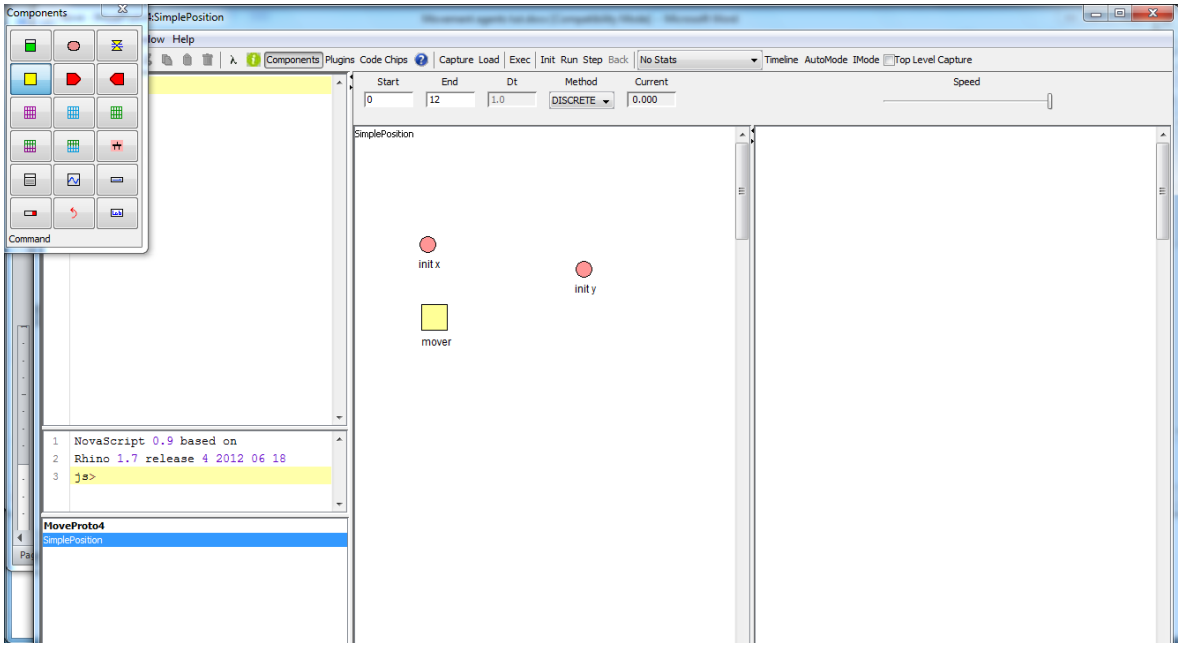


The fourth prototype has been achieved (save the model).

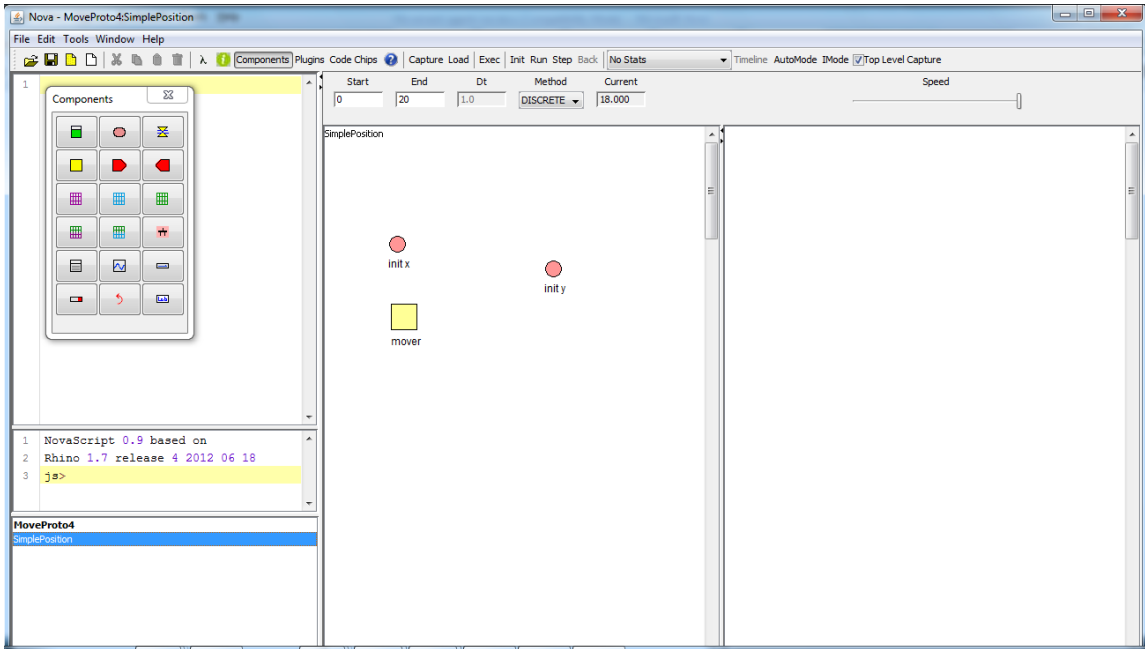
Prototype 5: move the single agent randomly

The fifth prototype is to move the single agent randomly.

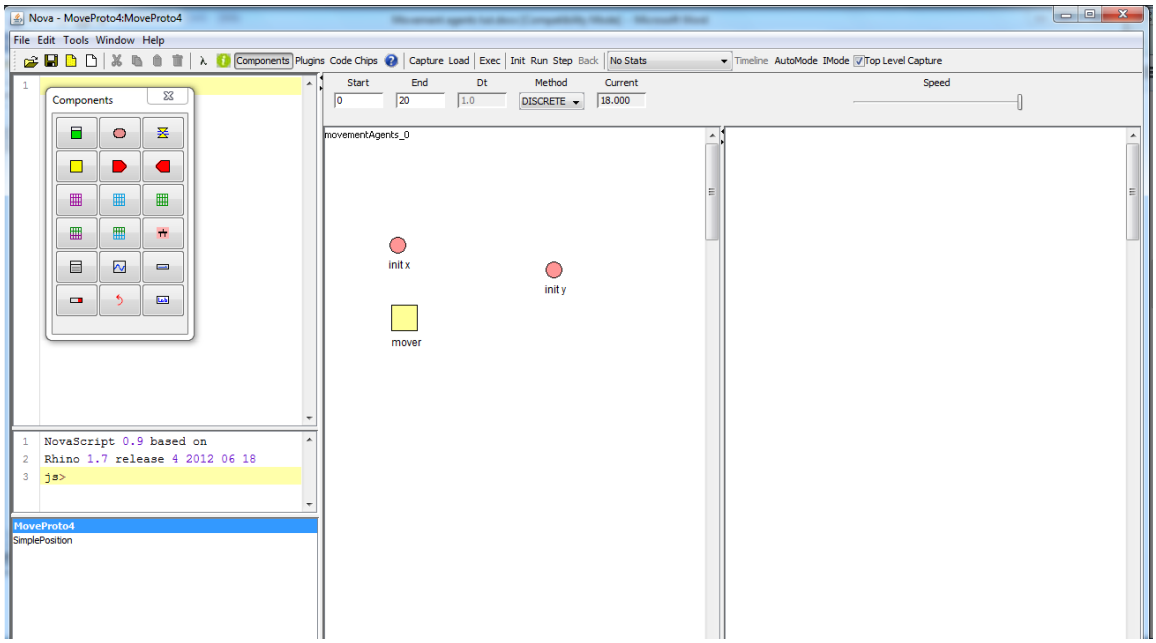
To move the agent, add a command from the Components menu:



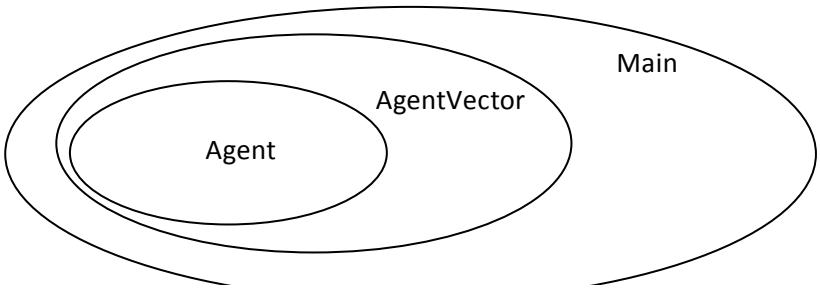
Change to the lower level (SinglePosition):



Note that the SinglePosition code isn't the agent. It's the design for the agent. The agent only is created when you drag the agent into the holder. The instance of the agent is obtained from double clicking on the container:



NOVA evaluates the instructions so that the last thing that is evaluated is a stock. To update a stock in each step, several terms have to be evaluated.



In Console part of the screen (i.e. in the middle of the left hand side of the screen), drill down:

First Capture and Load. Then to find where the agent thinks it is, in the Console, type

`main` <Enter>

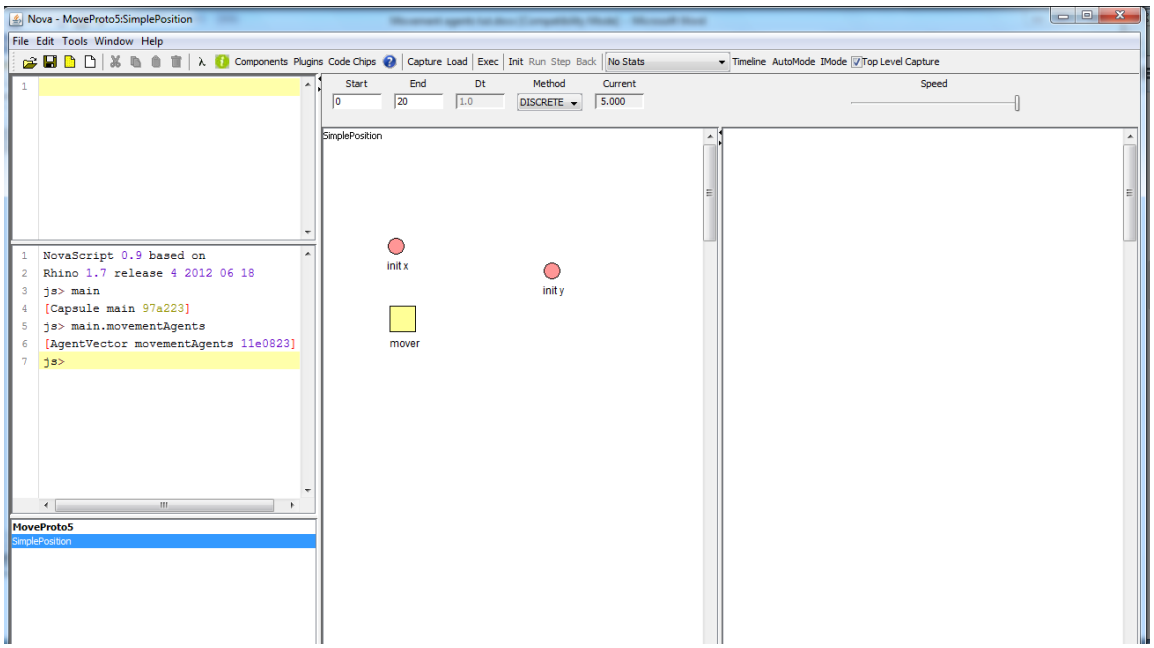
It should return the address of the main.

`main.movementAgents` <Enter>

(where the movementAgents is the name of the agent vector)

Since we have only one agent, we can ask for the first agent

`main.movementAgents.AGENTS[0]`



To get the x co-ordinate, type in

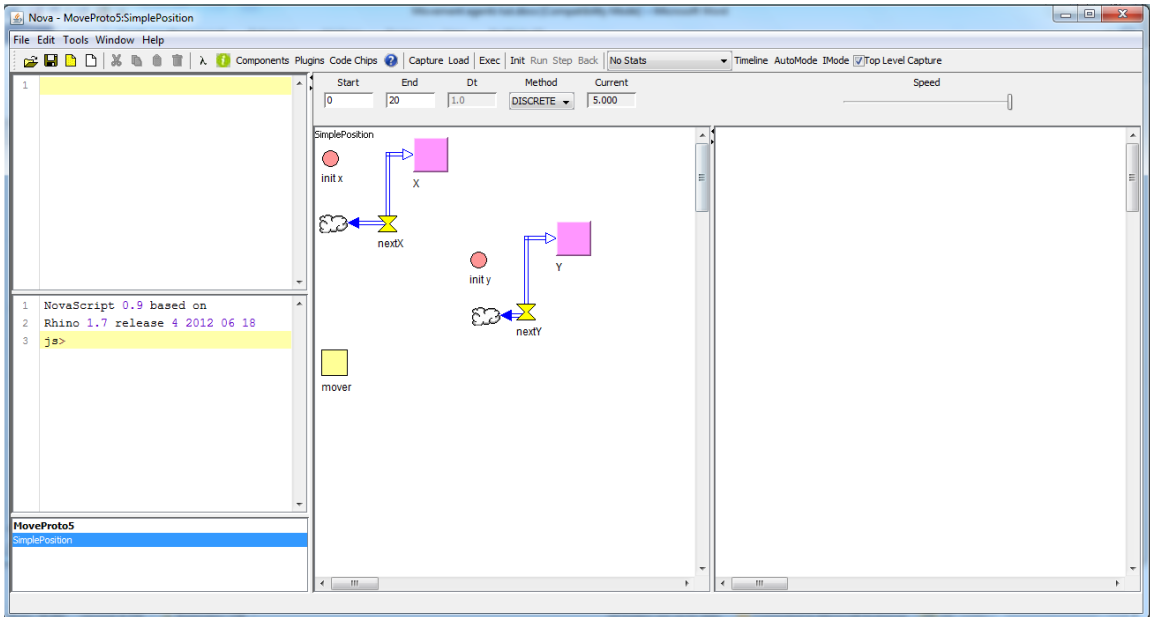
`main.movementAgents.AGENTS[0].LOCATION().x`

Similarly, to get the y co-ord, type in

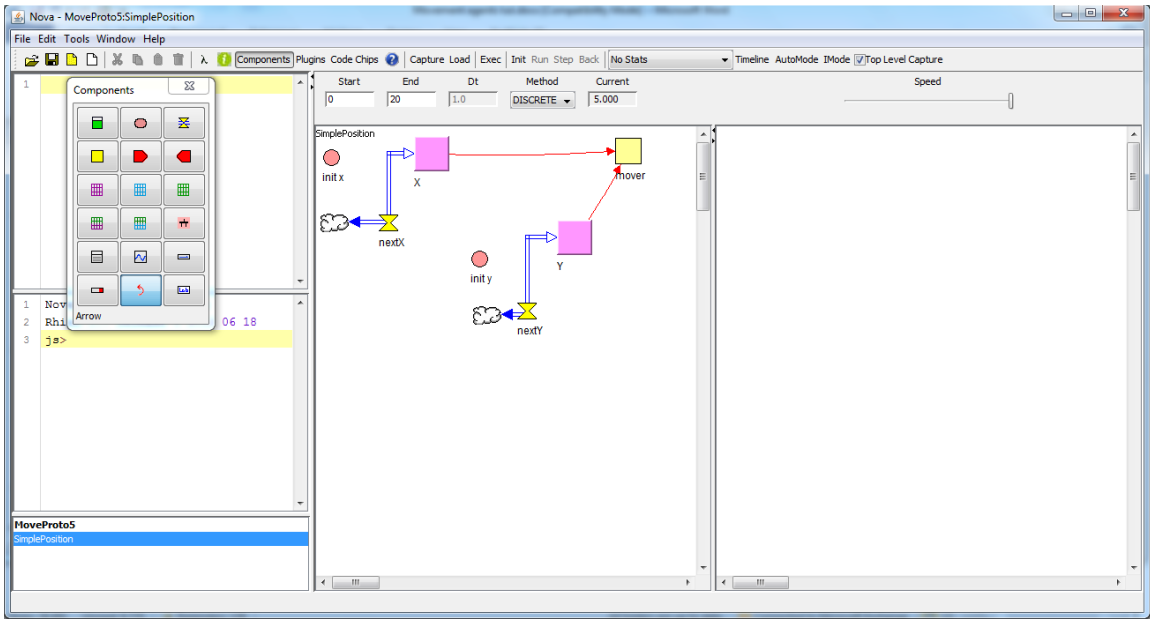
`main.movementAgents.AGENTS[0].LOCATION().y`

Prototype 6: use a stock and a flow to control the agent's position

To keep control the position of the agent, add stocks and flows to each part of the co-ordinate (x and y). The State must be discrete.



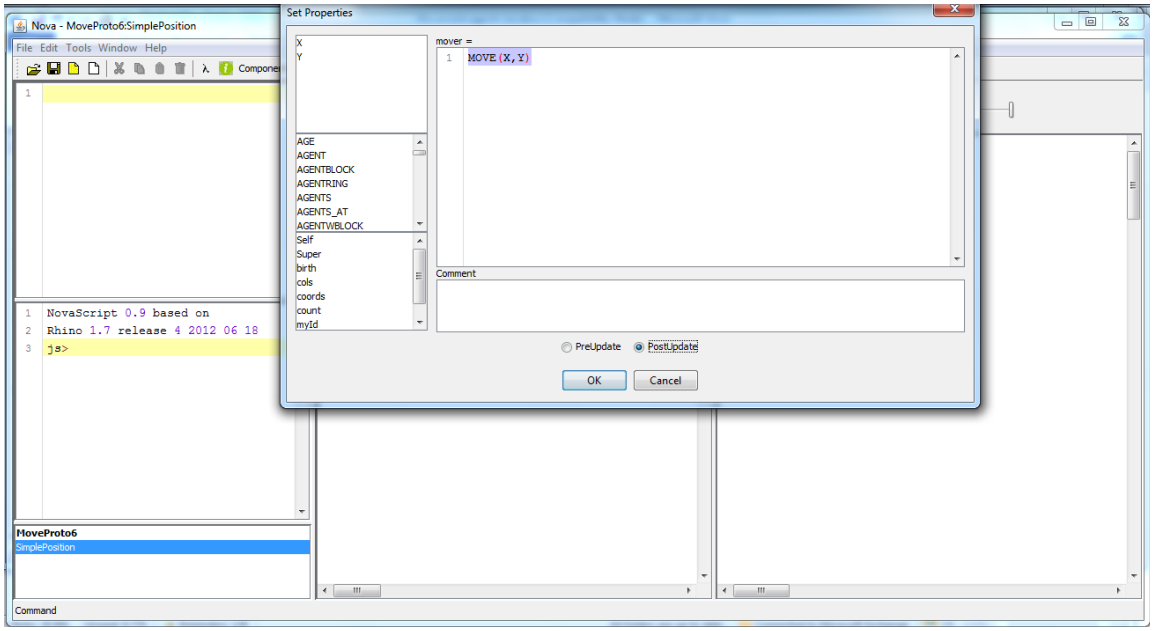
Connect the Stocks to the mover command:



In the command, add the code:

```
MOVE(x,y)
```

and make sure that the command is PostUpdate:



Prototype 7: the agent should move randomly in the 8x8 space

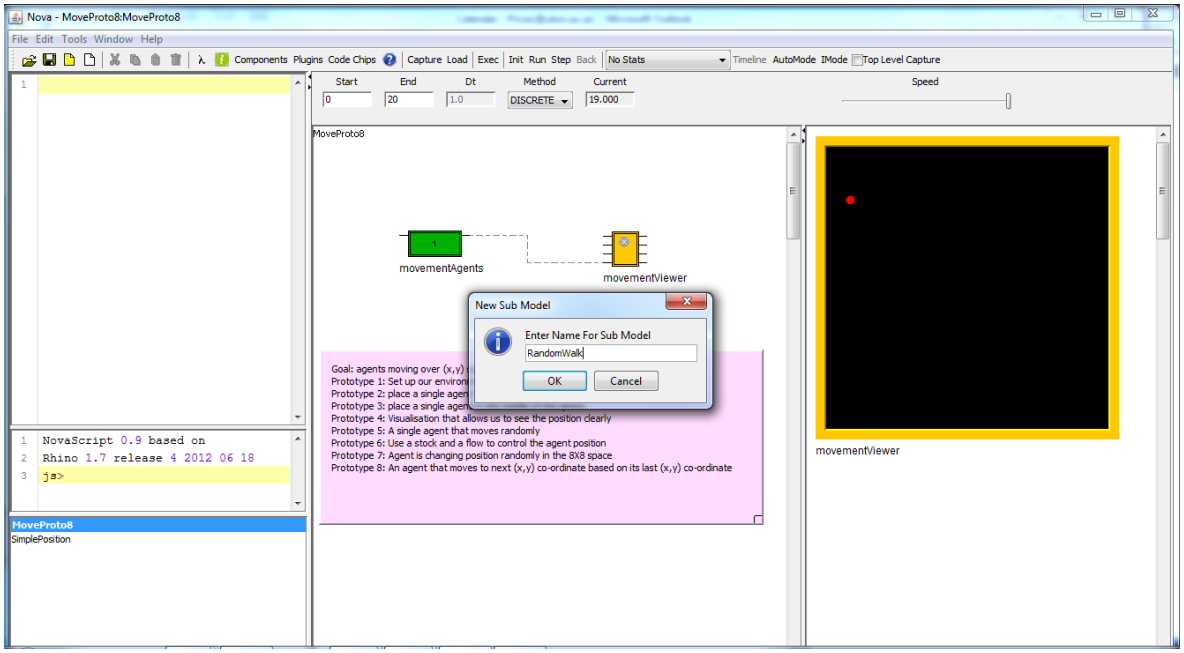
Change the flows for X and for Y to

```
Math.random()*7
```

respectively. This will cause the agent to “jump” around randomly.

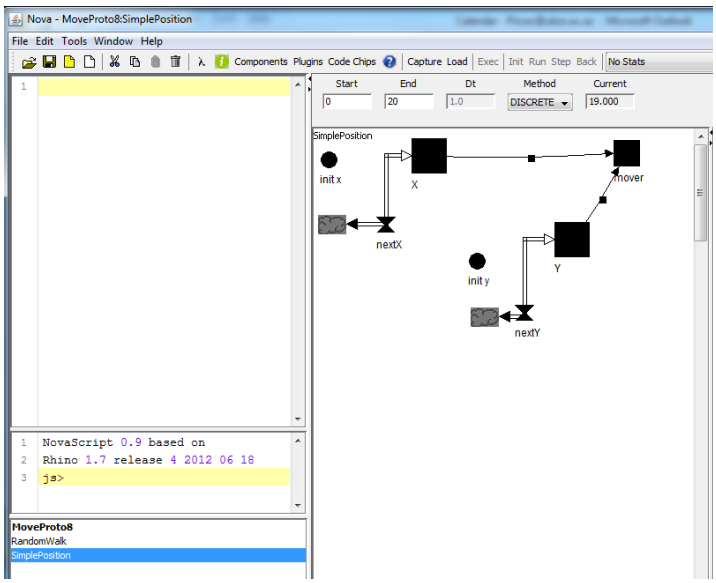
Prototype 8: the agent moves to the next position based on its previous position

To create an agent with a random walk, create a new sub-model:

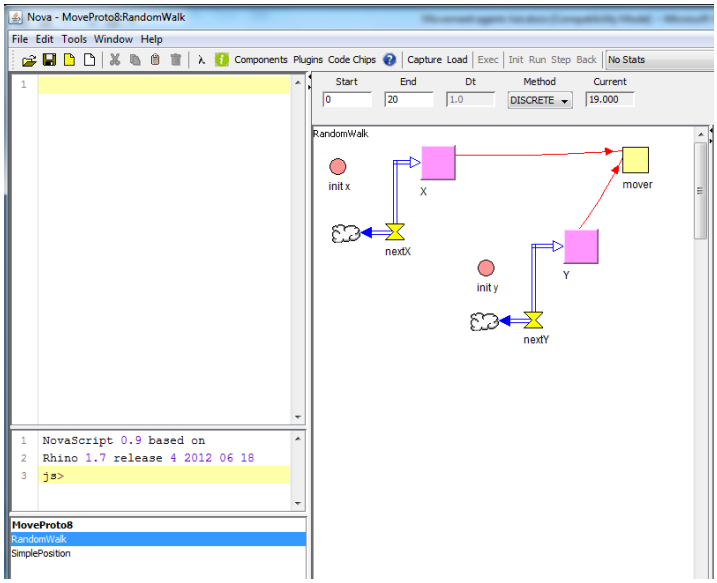


Go to SimplePosition submodel, highlight all, copy, go to RandomWalk and paste:

Copy:

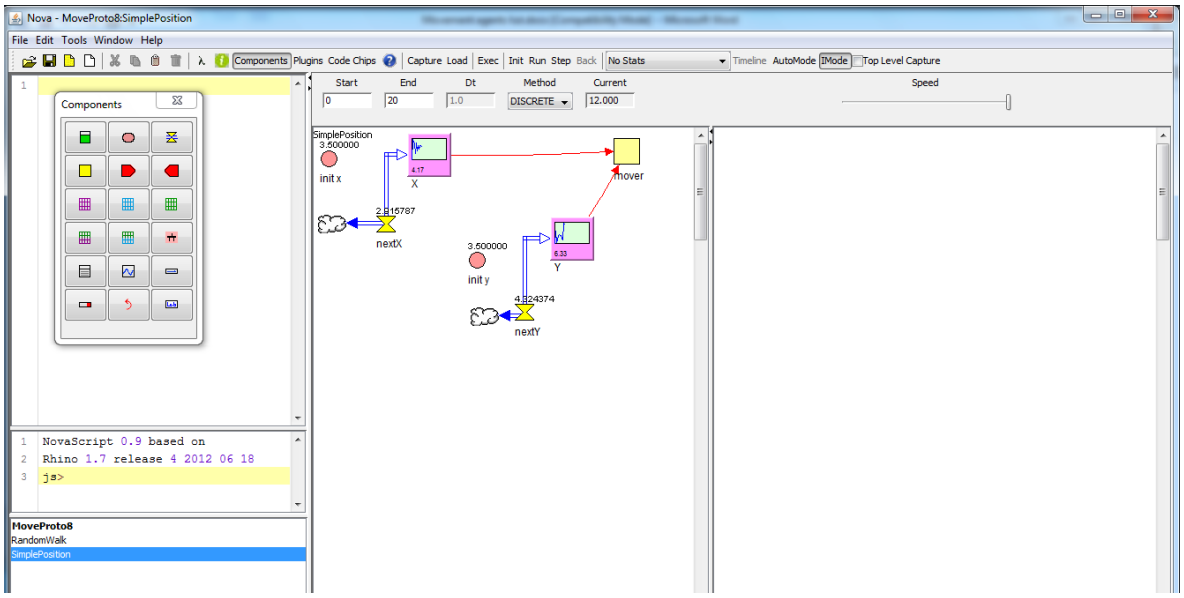


Paste:

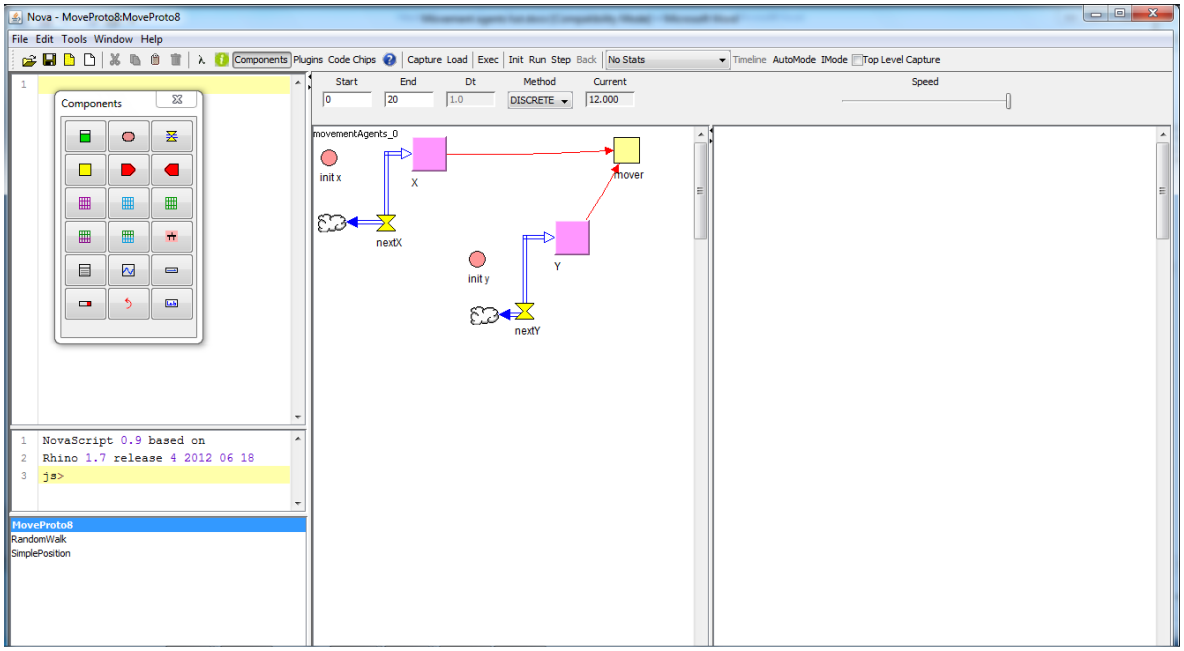


My SimplePosition model's arrows were disconnected after I copied the model ☹ ☹ ☹ ! Needed to reconnect them.

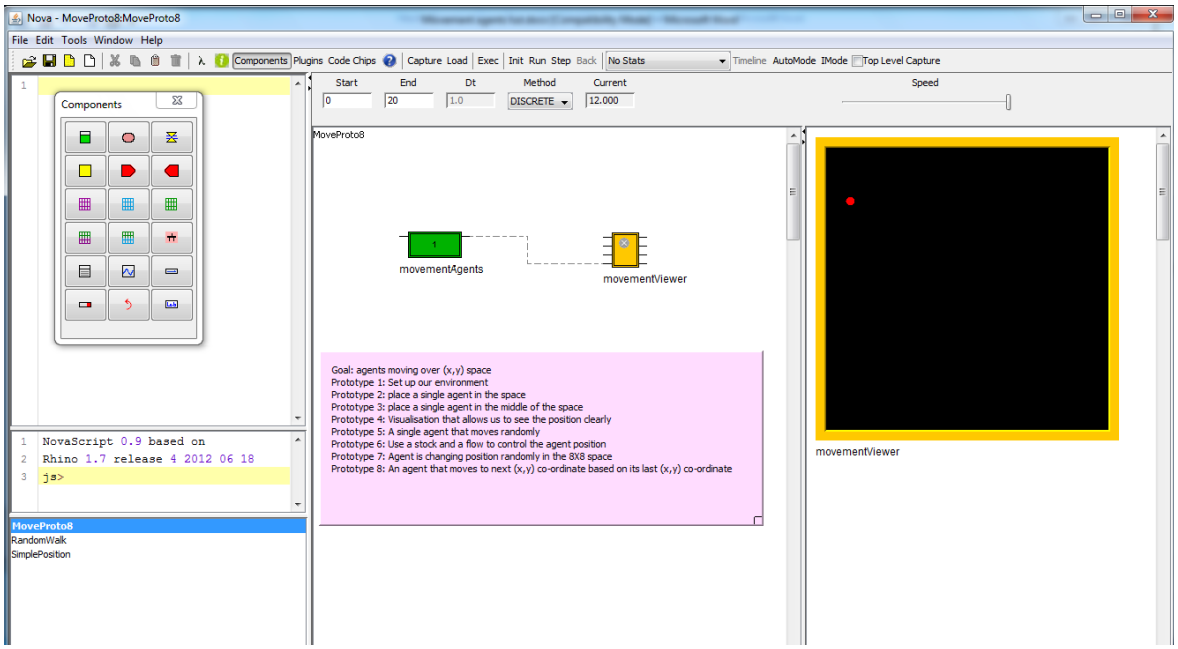
You can check the SimplePosition model by Capture Load Init IMode:



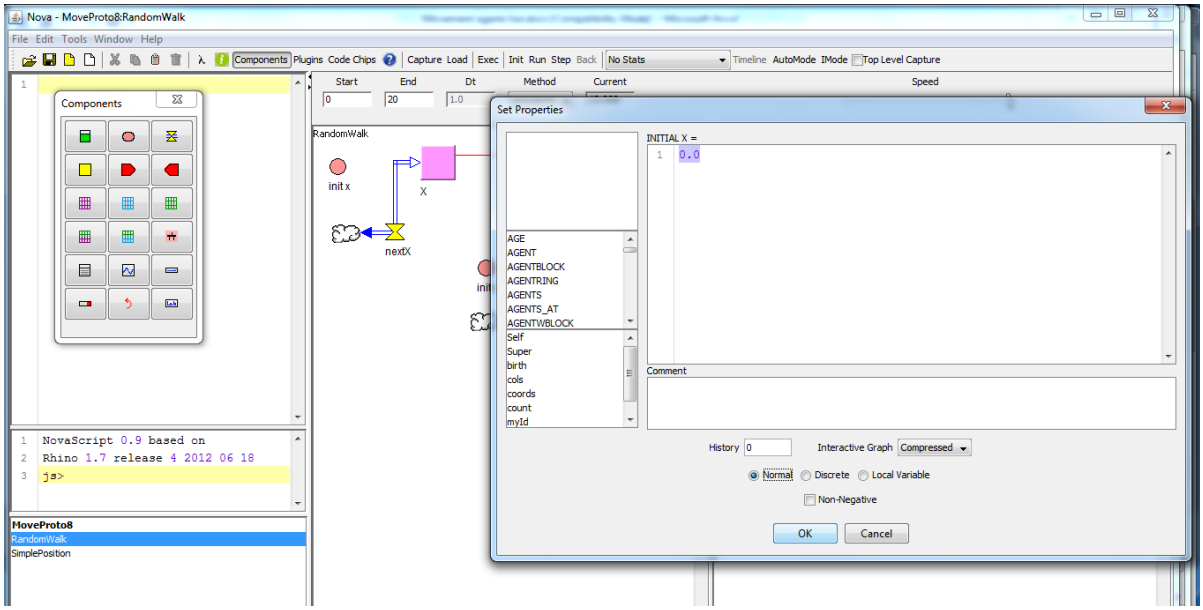
Note that this screen shows the SimplePosition instructions, whereas the following screen shows an instance of the agent. The difference is the name in the top left hand side of the canvas.



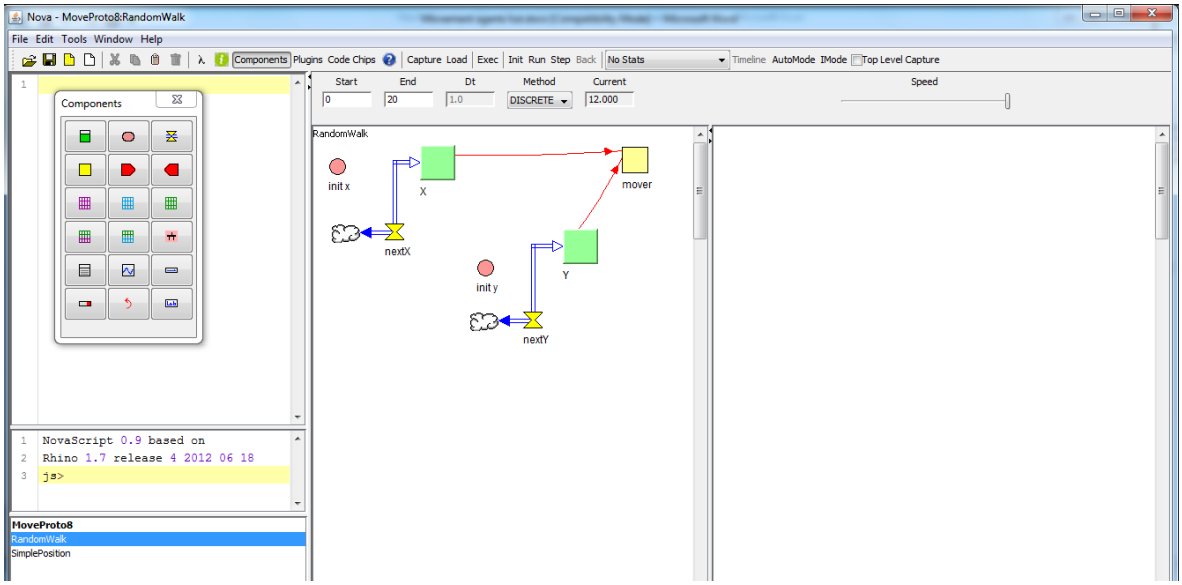
I got to the following screen by double clicking on the agent vector movementAgents:



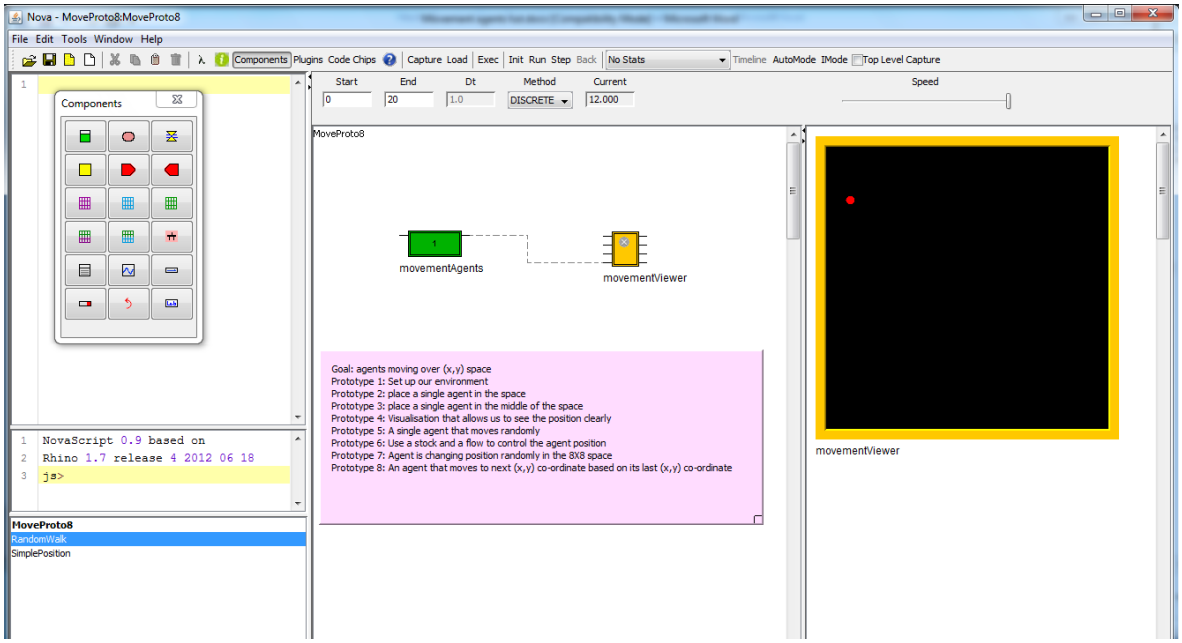
In the RandomWalk model, change the stock for X to be Normal:



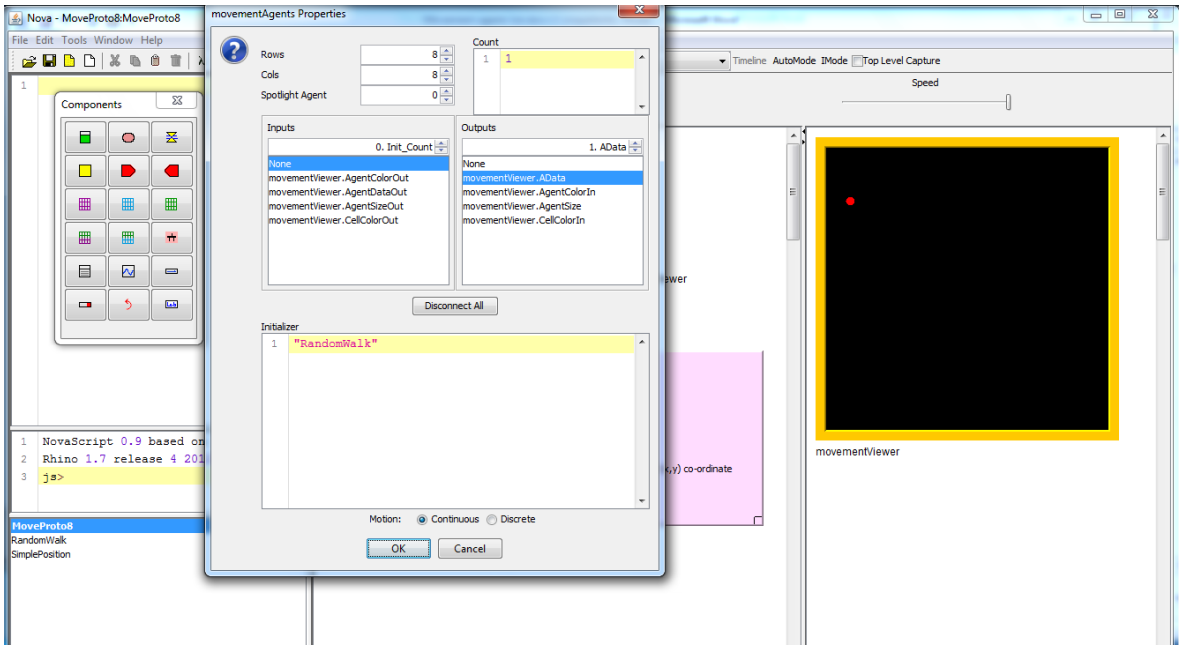
This will add the new value of the stock to the previous value.



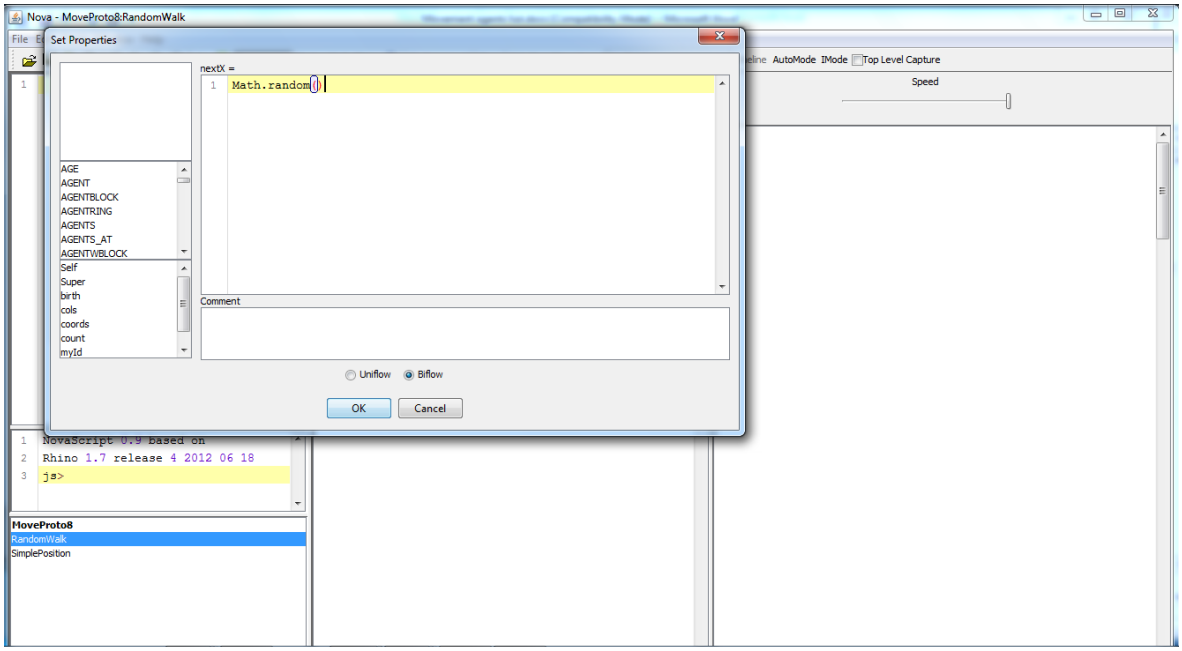
We want the RandomWalk rules to be the ones that are run in the main agent vector. We need to drag and drop the RandomWalk rules into the agent vector (thus replacing the SimplePosition rules):



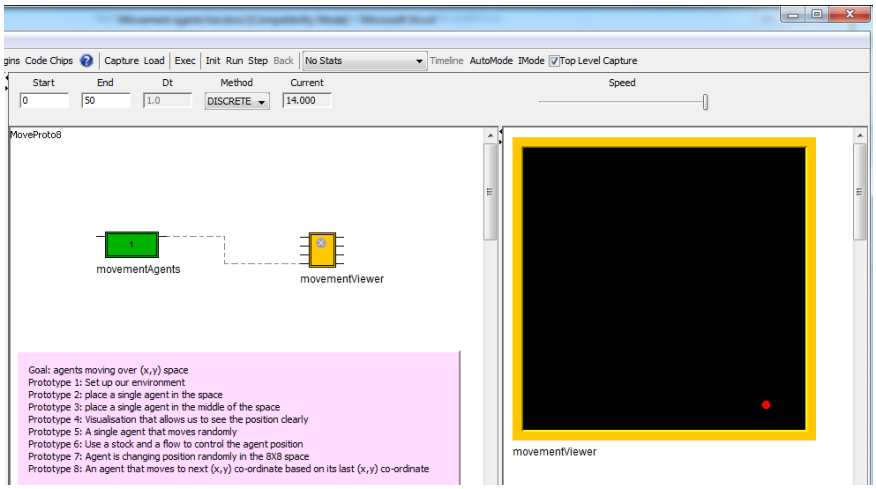
Check which code is in the agent vector by right-clicking on movementAgents: the model "RandomWalk" will run (see the Initilizer window).

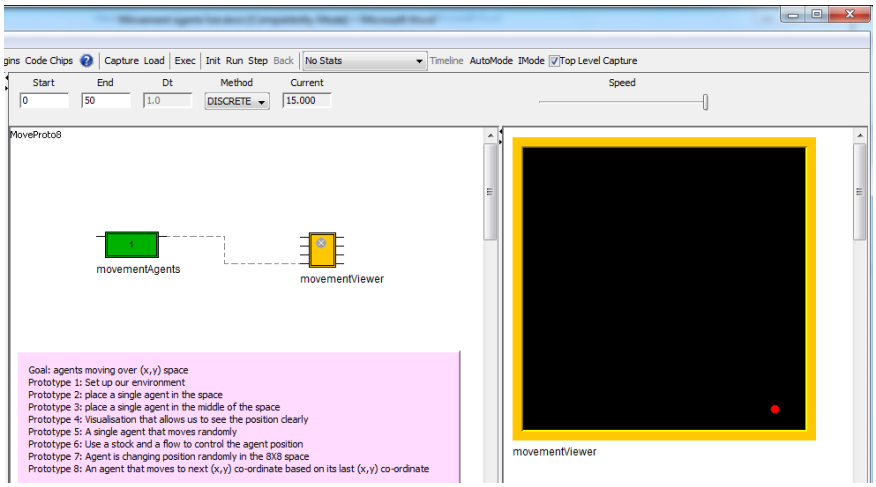


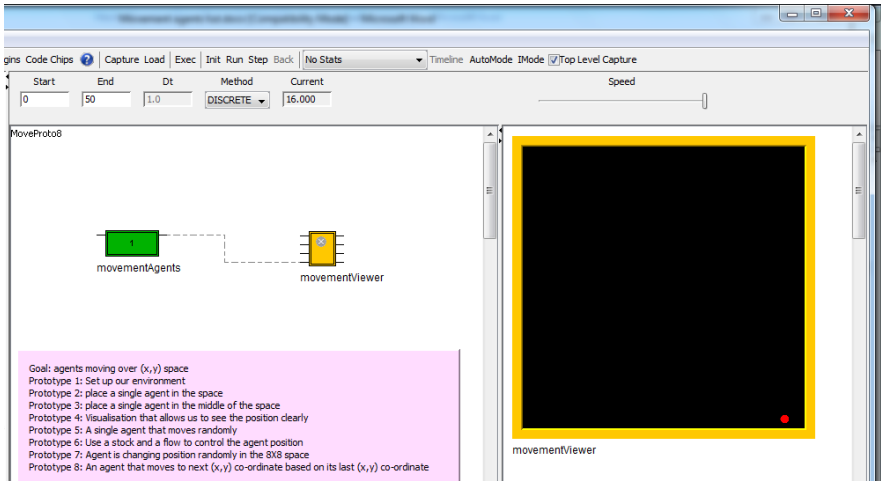
The code causes the agent to go off the screen sometimes. So need to remove the *7 in the flow of X and Y:

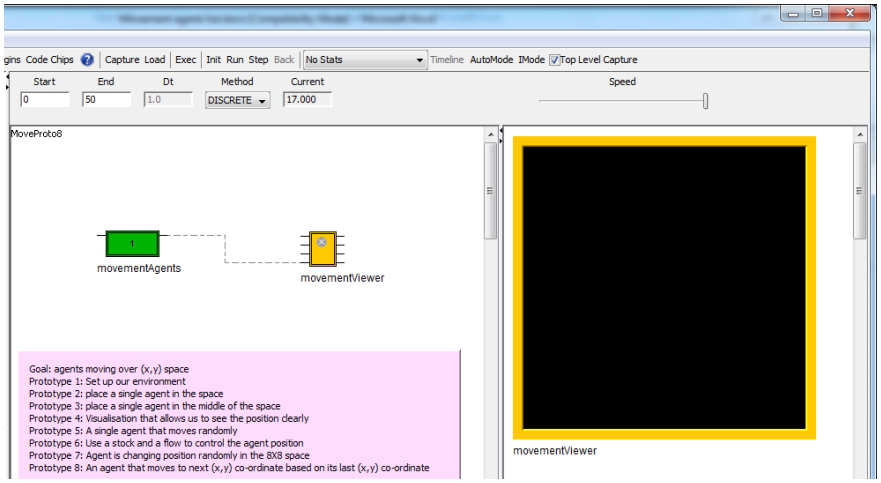


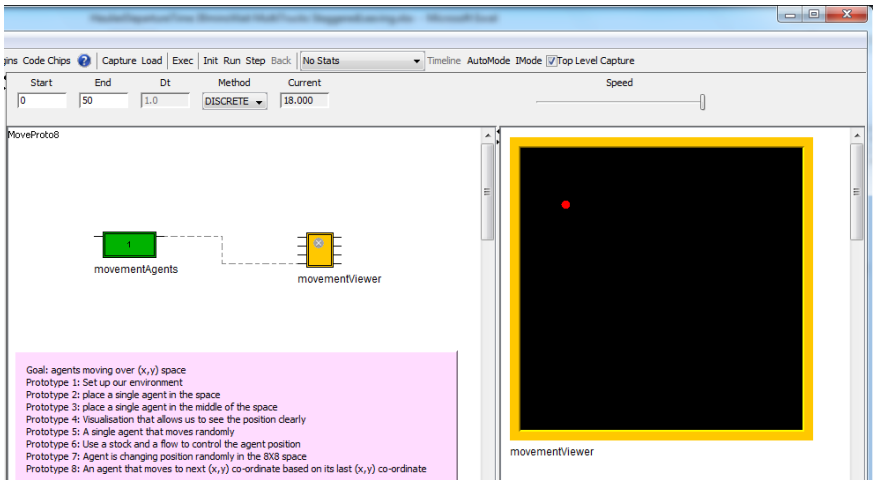
The agent now moves as a torus at the moment (note the current timestep, and how the agent moves in the following screenshots):

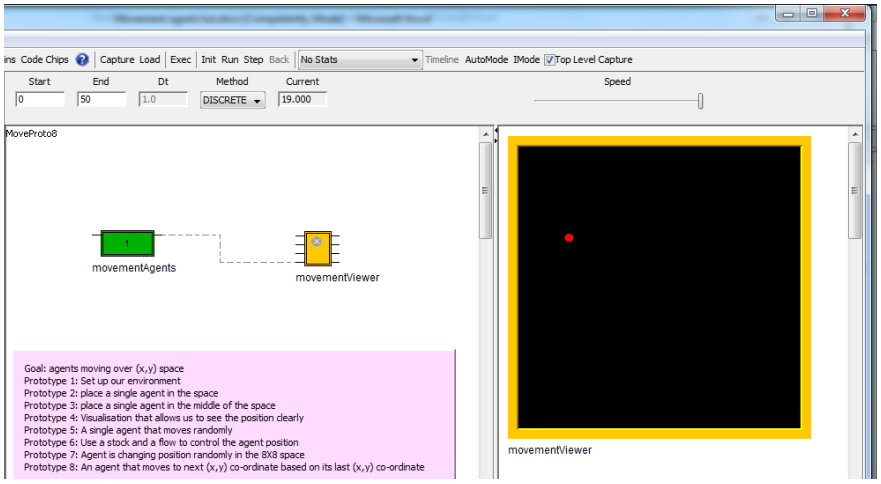




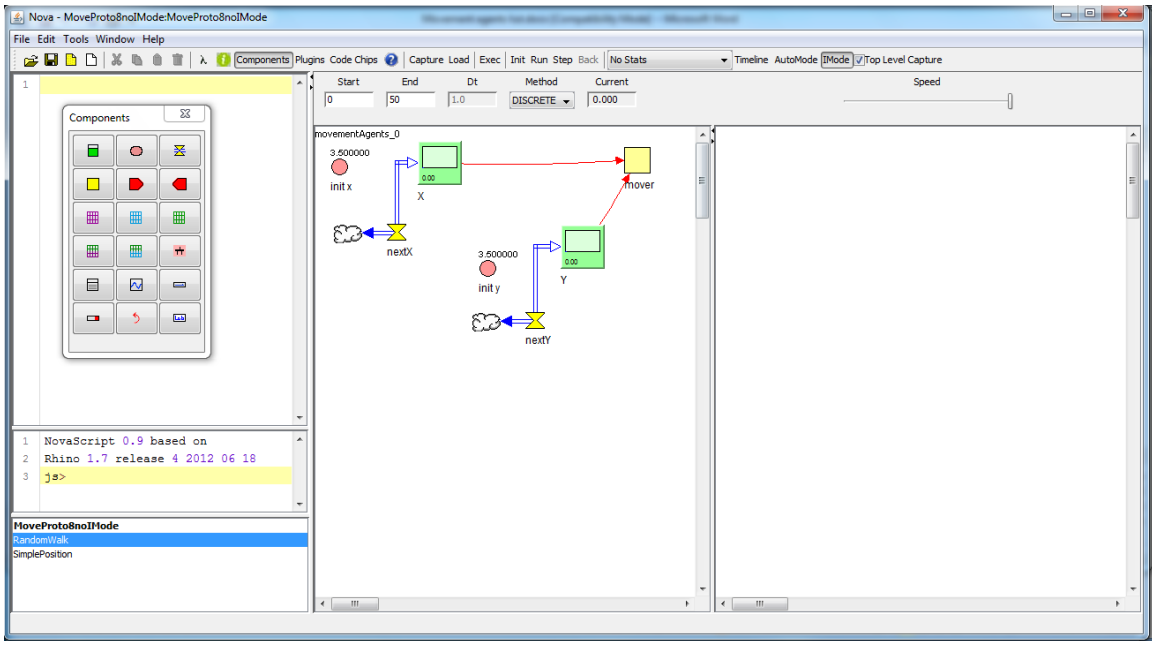

Code Chips
Capture Load
Exec
Init Run Step Back
No Stats
Timeline AutoMode IMode Top Level Capture
Start End Dt Method Current
0 50 1.0 DISCRETE 16.000
Speed
MoveProto8
1
movementAgents
movementViewer
Goal: agents moving over (x,y) space
Prototype 1: Set up our environment
Prototype 2: place a single agent in the space
Prototype 3: place a single agent in the middle of the space
Prototype 4: Visualisation that allows us to see the position clearly
Prototype 5: A single agent that moves randomly
Prototype 6: Use a stock and a flow to control the agent position
Prototype 7: Agent is changing position randomly in the 8X8 space
Prototype 8: An agent that moves to next (x,y) co-ordinate based on its last (x,y) co-ordinate
movementViewer


Code Chips
Capture Load
Exec
Init Run Step Back
No Stats
Timeline AutoMode IMode Top Level Capture
Start End Dt Method Current
0 50 1.0 DISCRETE 17.000
Speed
MoveProto8
1
movementAgents
movementViewer
Goal: agents moving over (x,y) space
Prototype 1: Set up our environment
Prototype 2: place a single agent in the space
Prototype 3: place a single agent in the middle of the space
Prototype 4: Visualisation that allows us to see the position clearly
Prototype 5: A single agent that moves randomly
Prototype 6: Use a stock and a flow to control the agent position
Prototype 7: Agent is changing position randomly in the 8X8 space
Prototype 8: An agent that moves to next (x,y) co-ordinate based on its last (x,y) co-ordinate
movementViewer

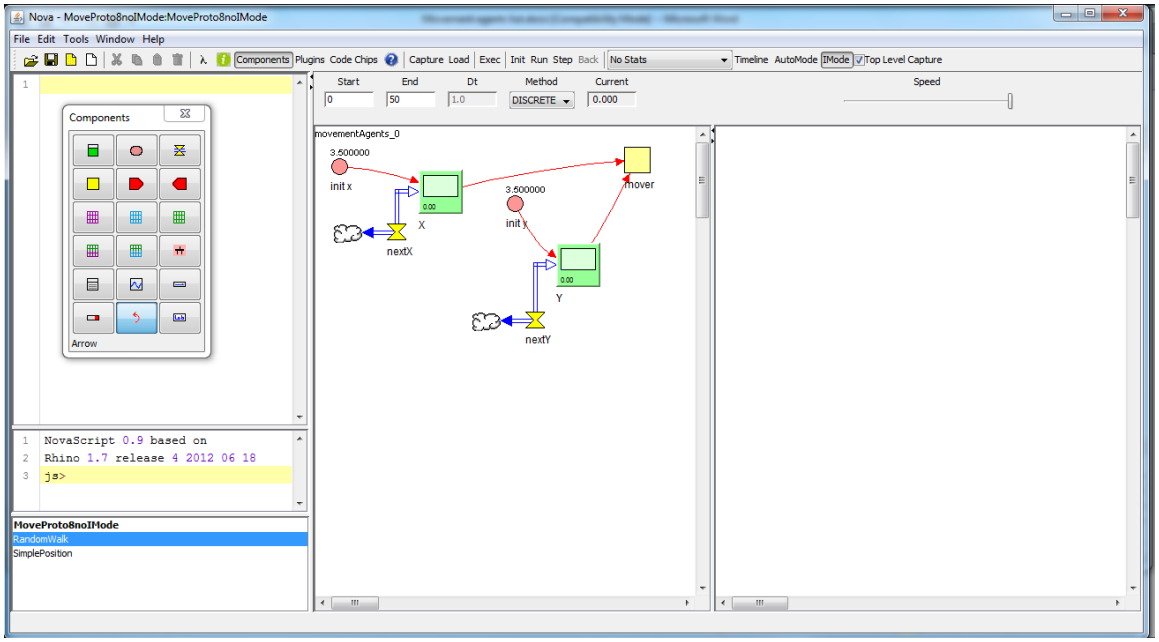

Code Chips
Capture Load
Exec
Init Run Step Back
No Stats
Timeline AutoMode IMode Top Level Capture
Start End Dt Method Current
0 50 1.0 DISCRETE 18.000
Speed
MoveProto8
1
movementAgents
movementViewer
Goal: agents moving over (x,y) space
Prototype 1: Set up our environment
Prototype 2: place a single agent in the space
Prototype 3: place a single agent in the middle of the space
Prototype 4: Visualisation that allows us to see the position clearly
Prototype 5: A single agent that moves randomly
Prototype 6: Use a stock and a flow to control the agent position
Prototype 7: Agent is changing position randomly in the 8X8 space
Prototype 8: An agent that moves to next (x,y) co-ordinate based on its last (x,y) co-ordinate
movementViewer



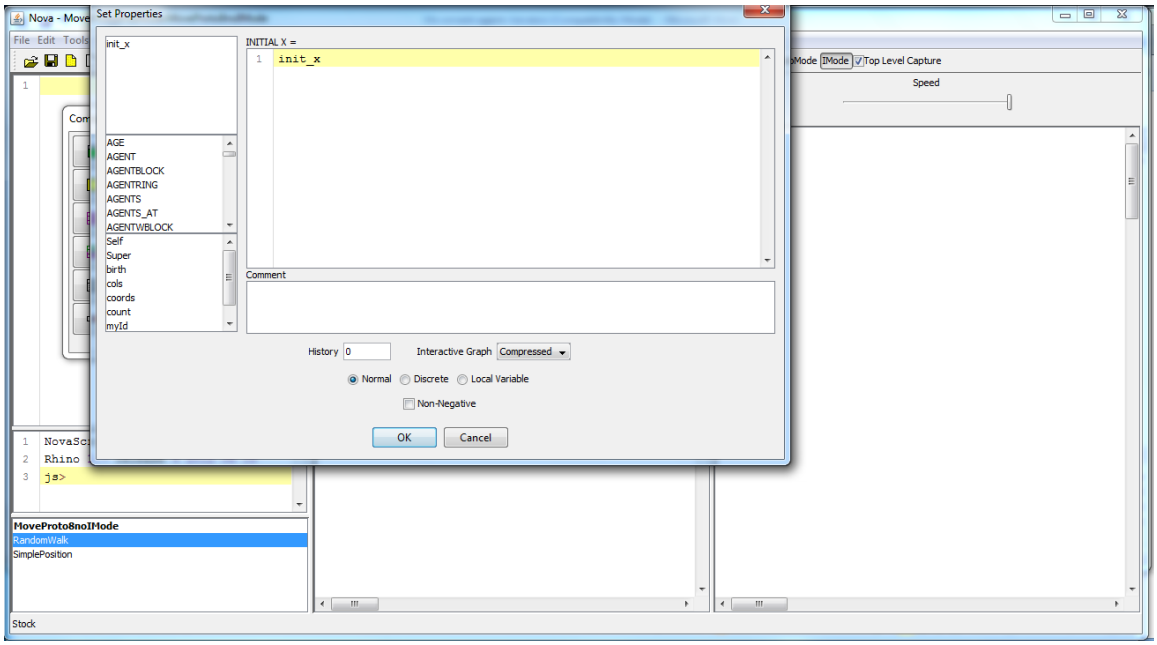
On running the model, the initial value of X is 3.5 (same for Y), but when moving, it starts from (0,0). This is not what was intended.

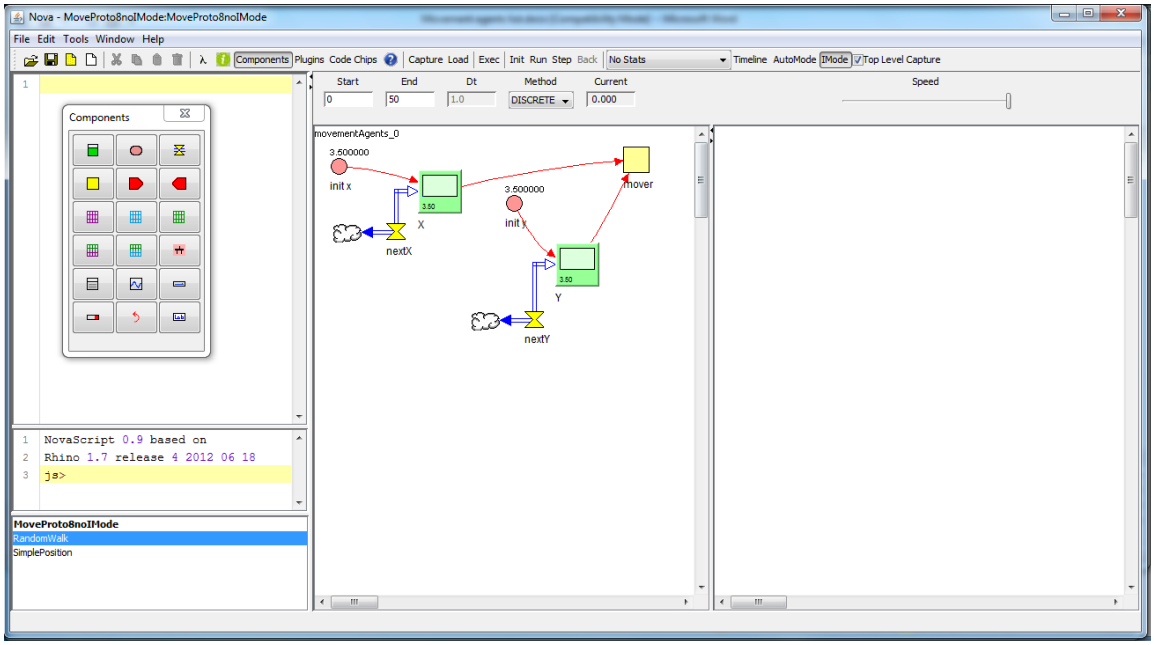


We need to fix this by connecting the term to the stock for X and Y respectively.



The initial value of the stocks also needs to be changed:



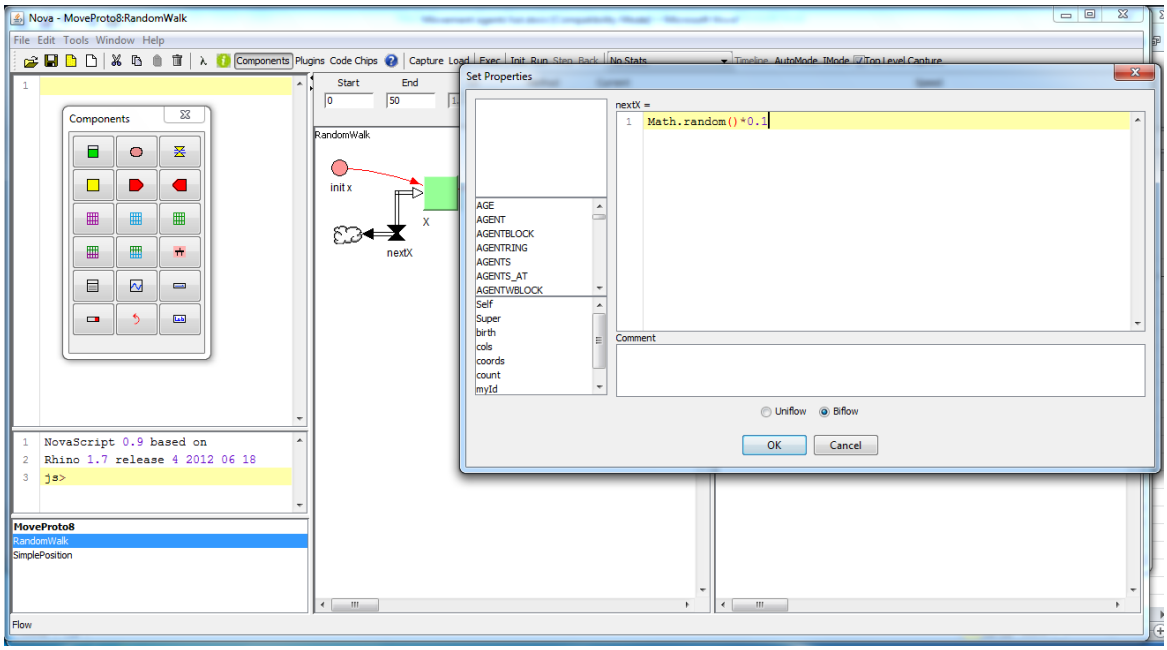


Save as a new version.

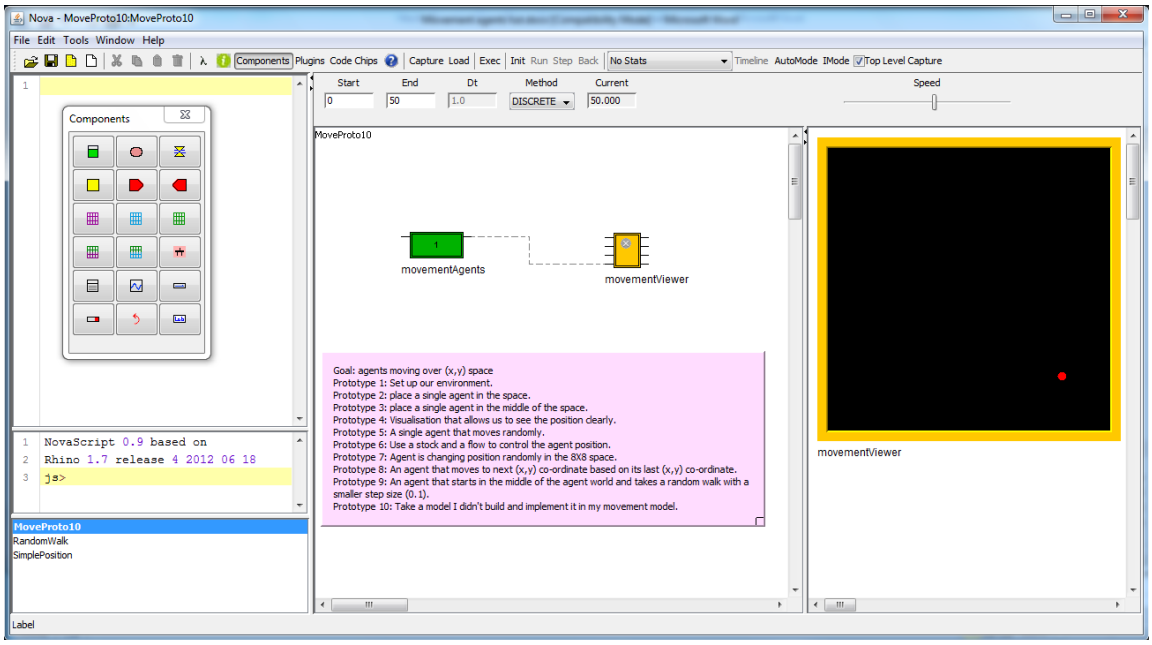
In this version, the x and y positions always increase.

Prototype 9: the agent starts in the middle of the space and moves to the next position based on its previous position with a smaller step size

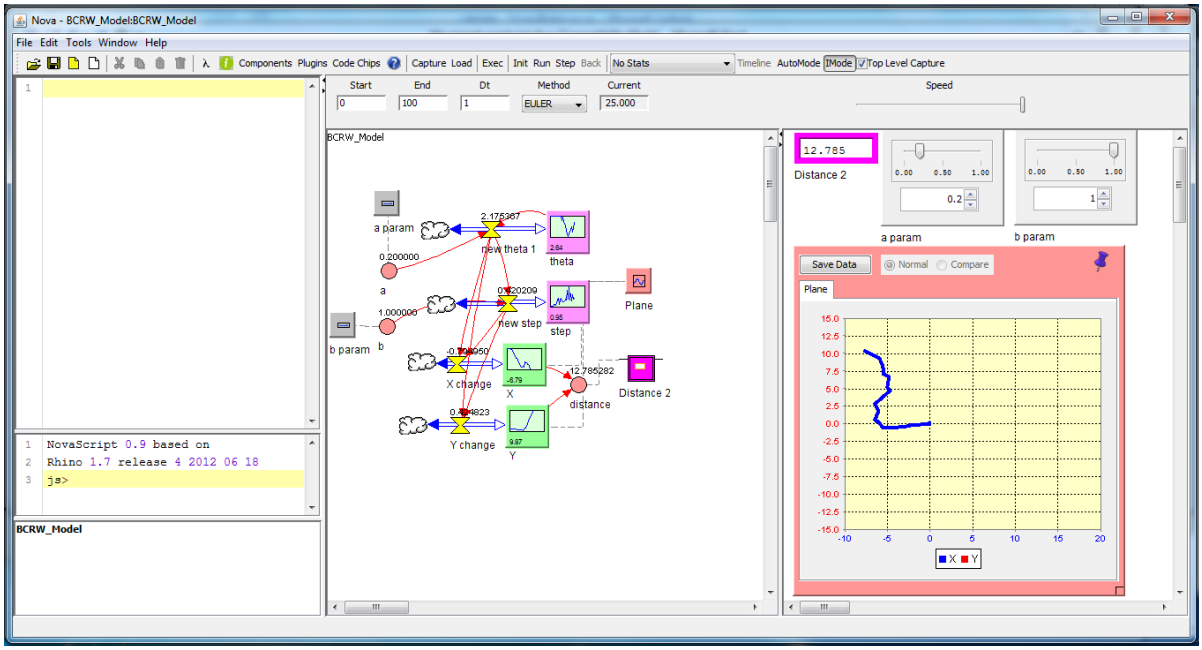
To make the step size smaller, go to the RandomWalk model and update the X and Y flows:



Prototype 10: use a model I didn't build and implement it in my movement model



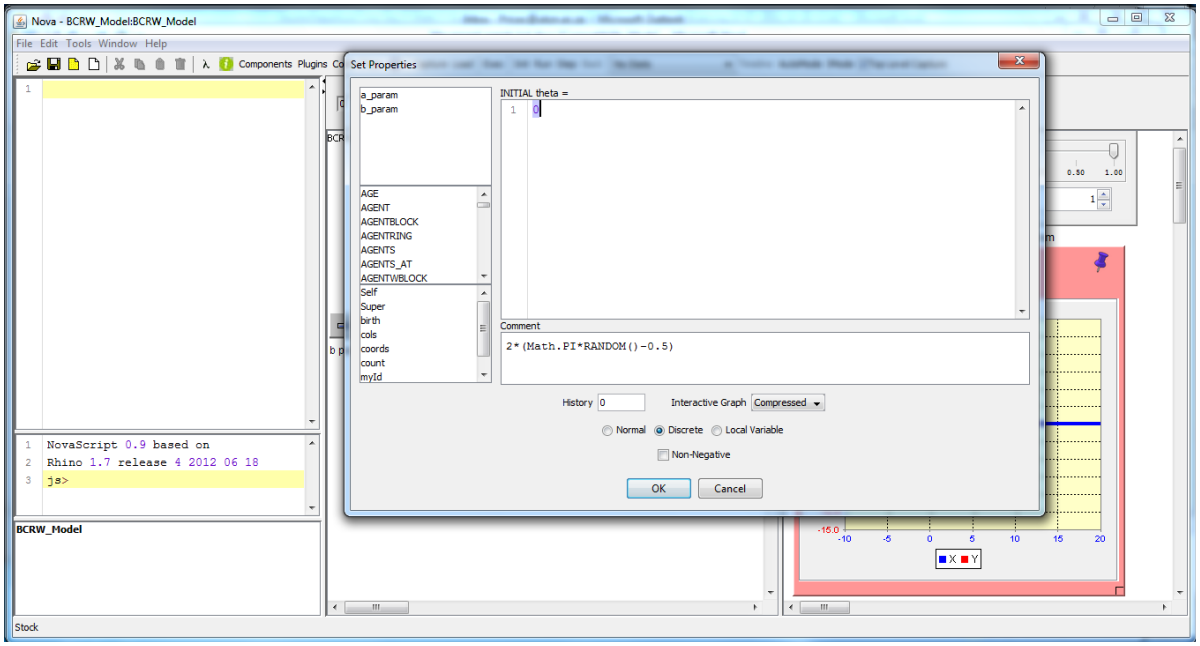
Looking at someone else's model for a Biased Correlated random walk: look at model BCRWSim.nva using the IMode (one can change the sliders for a param and for b param to experiment):



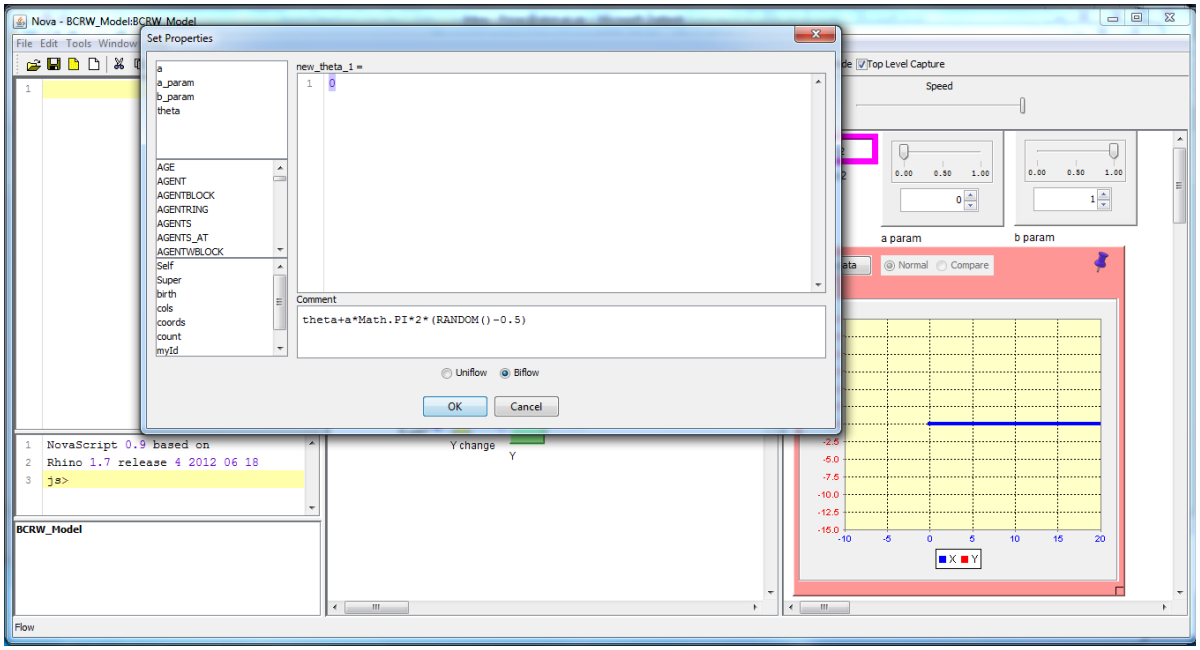
If the value of a = 0, the walk is in a straight line.

To deconstruct the model (to see what the model actually does), you can try changing the values of theta, e.g.

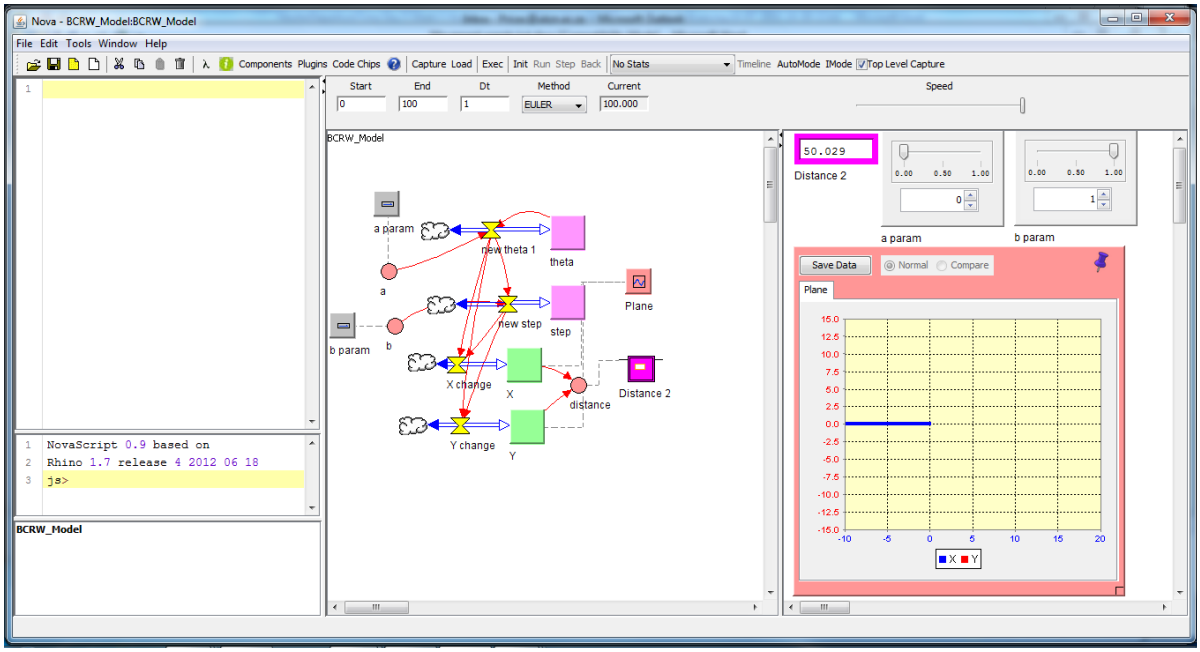
Changing the stock's initial value:



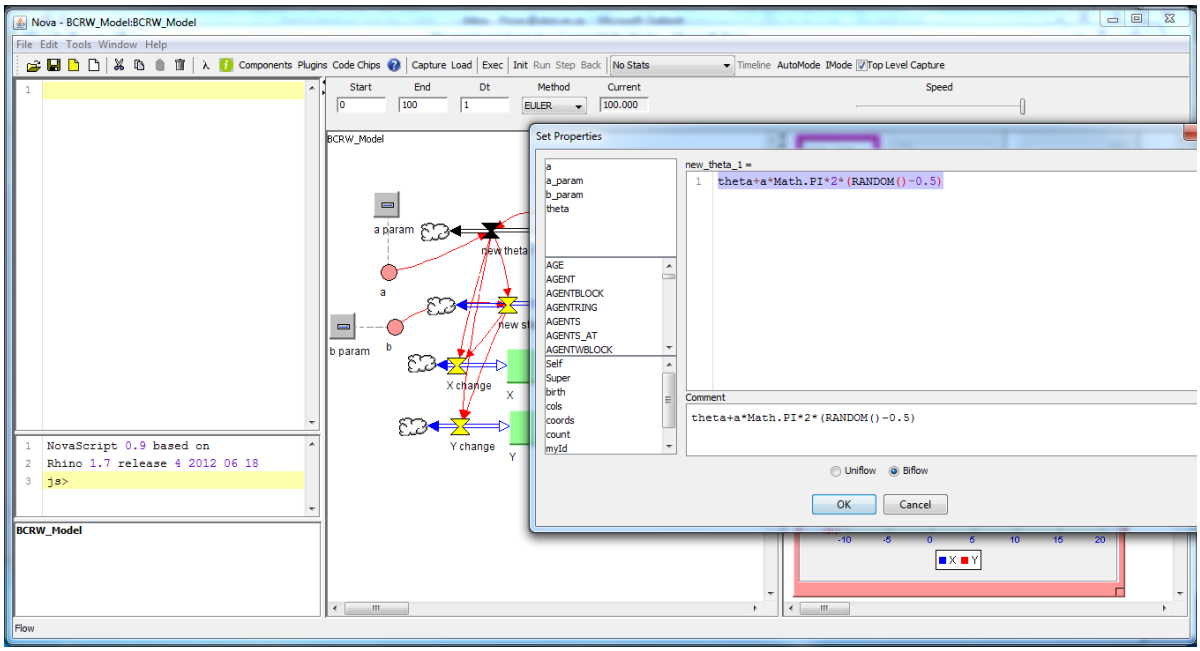
Changing the flow:

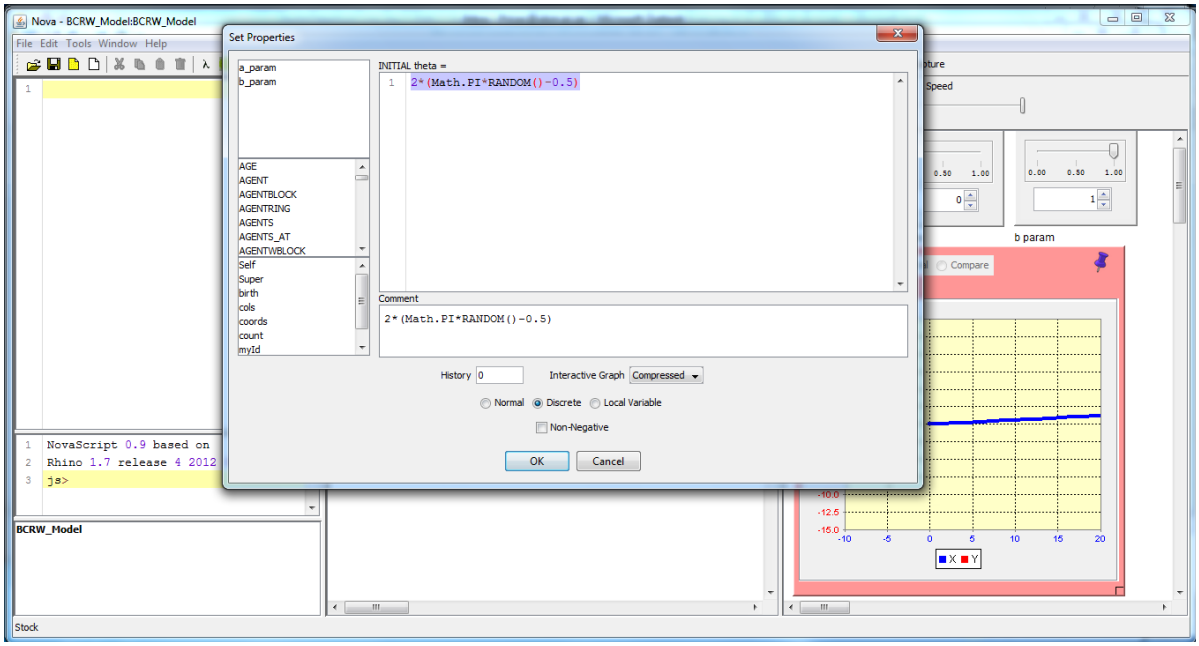


Changing the flow to 3.14 (pi):

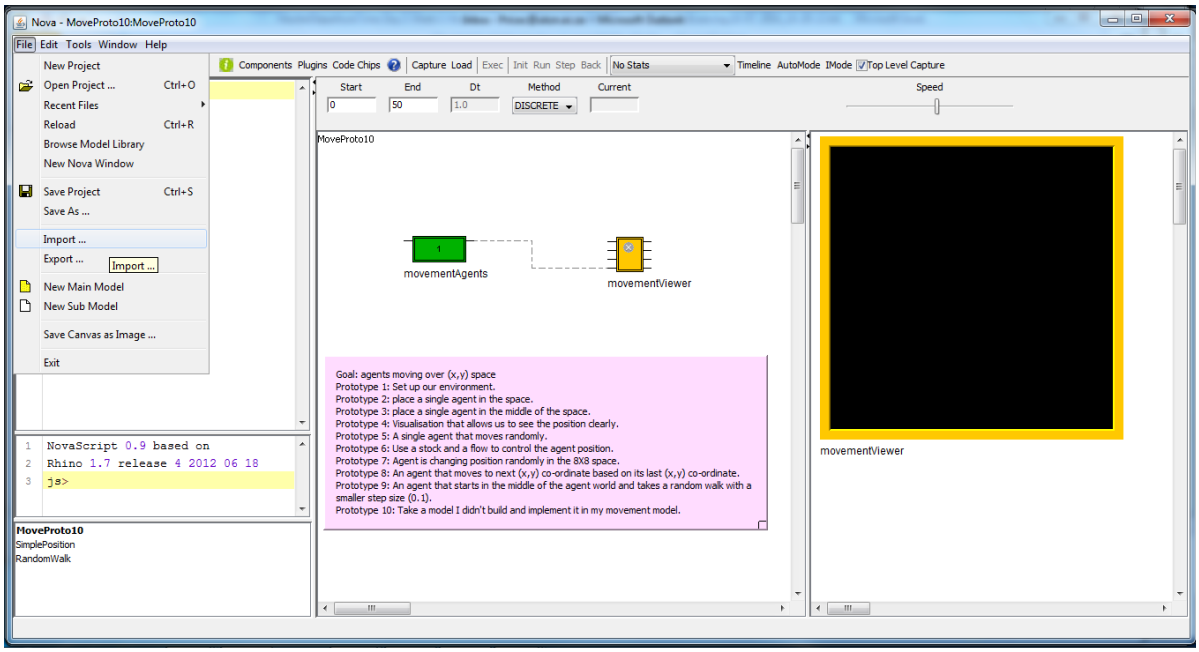


Adjust the model back to how it was (in the theta stock and flow).

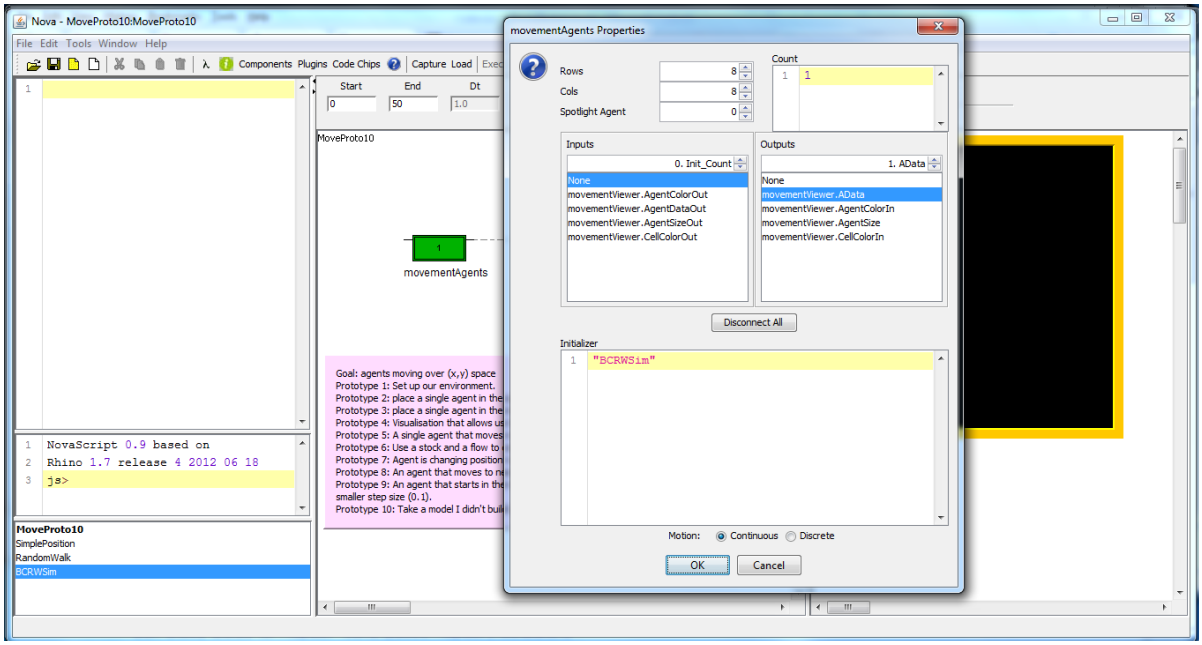




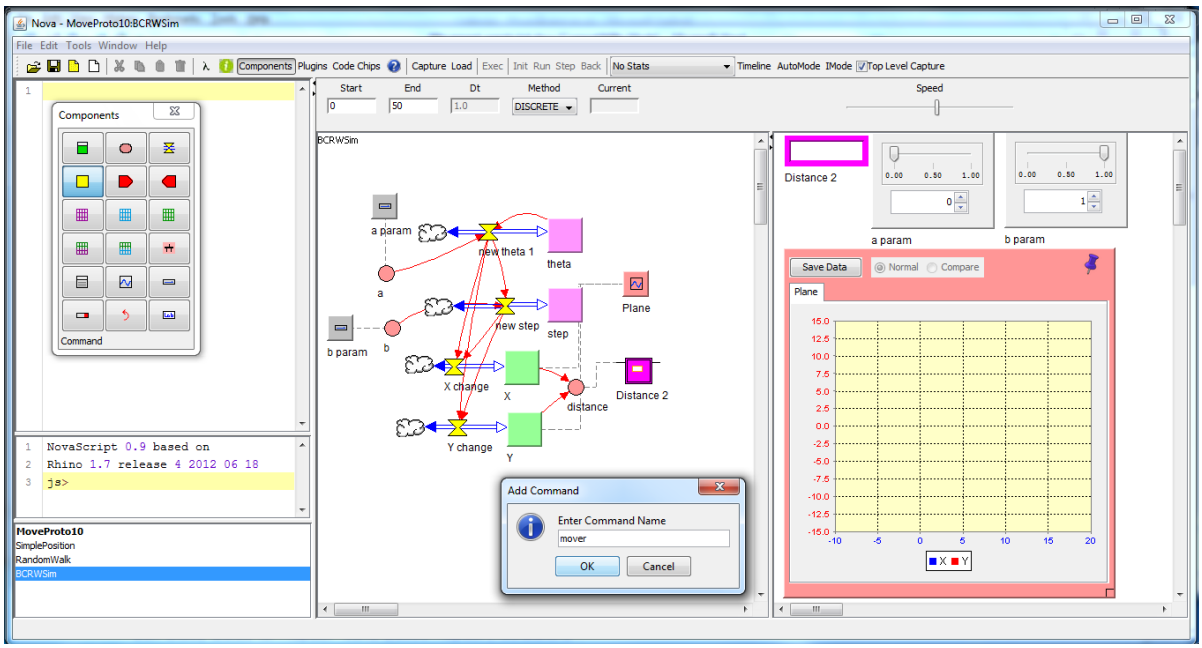
To import the Biased Correlated Random Walk model into our prototype agent model, reload the previous prototype model. Then import the BCRWmodel.

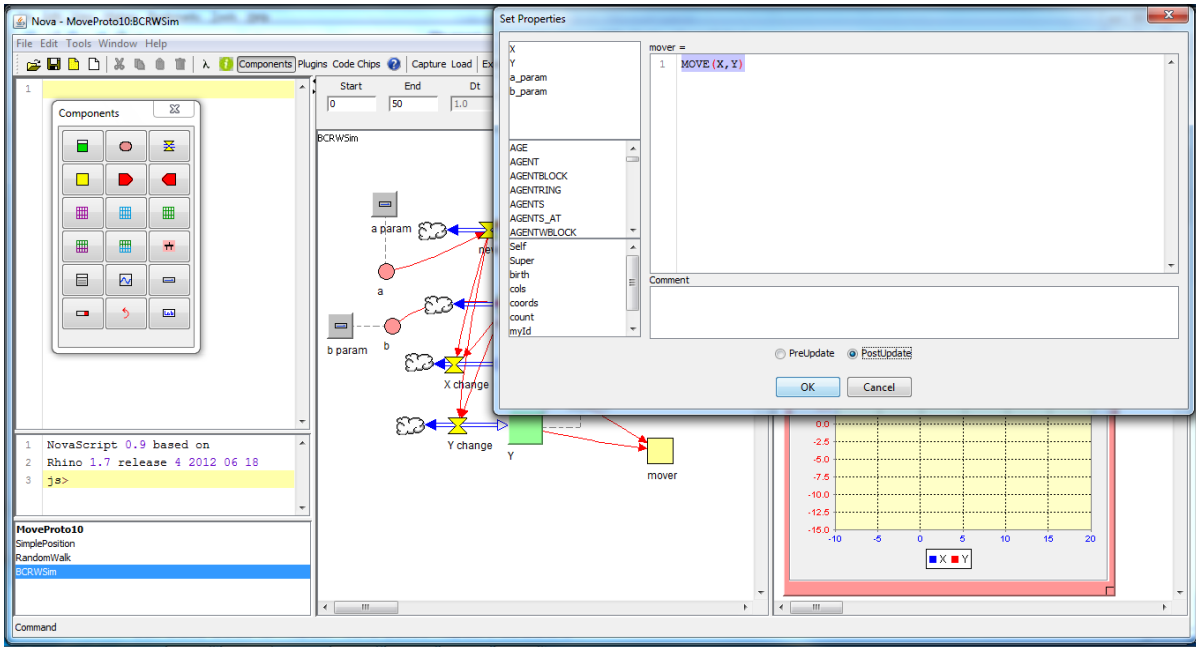


Move the BCRWSim into the agent vector movementAgents (and check that it's there by right-clicking on it):



In the BCRWSim model, add a mover command:

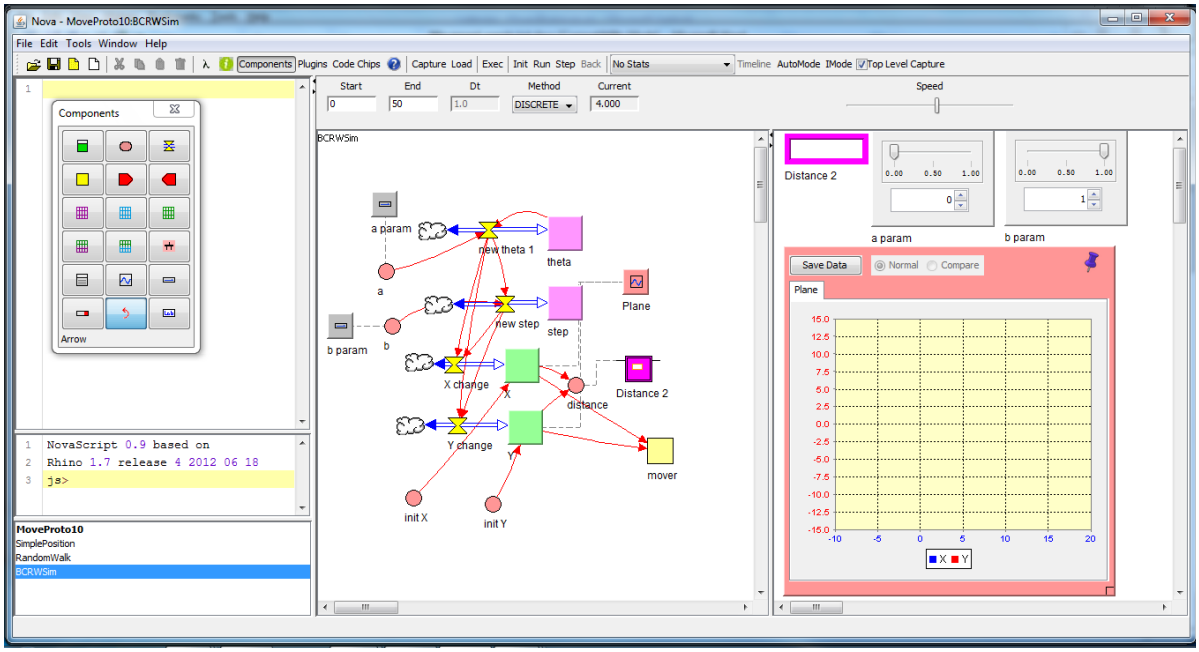


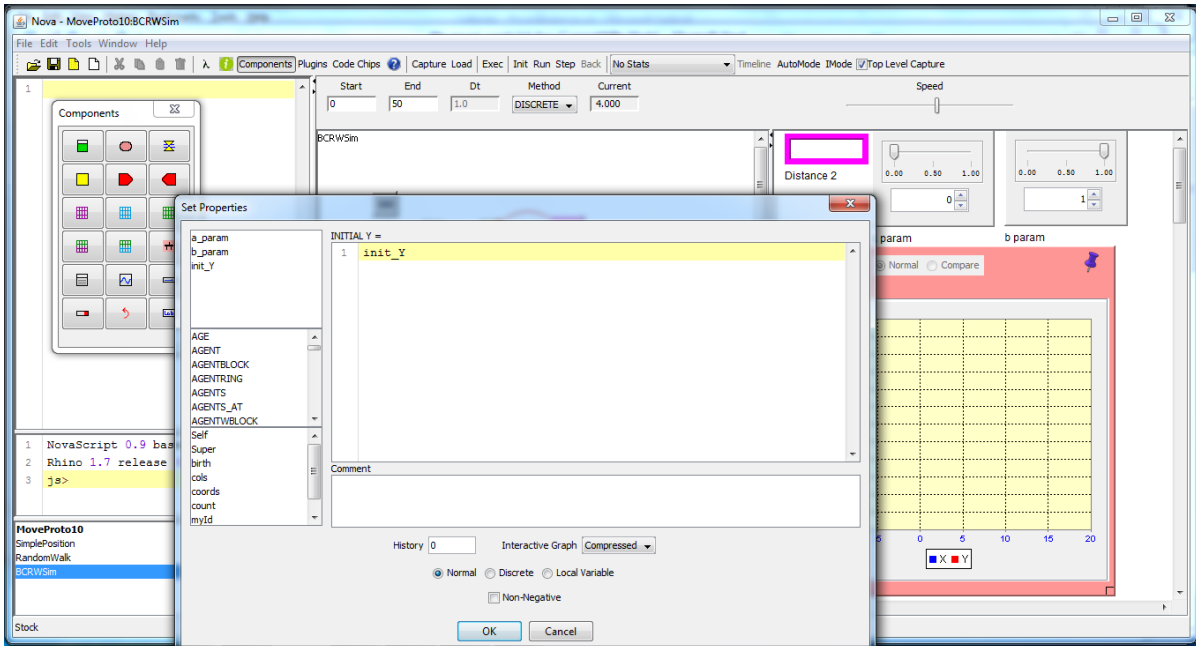


Remember to change the command to evaluate PostUpdate. You need this, because you need to compute the x position and the y position and then show the outcome on the graphic (i.e. after the calculation in the stock has taken place). (If you kept the command PreUpdate, your visuals would always be one step behind.) Tip: the MOVE command always should be used with PostUpdate.

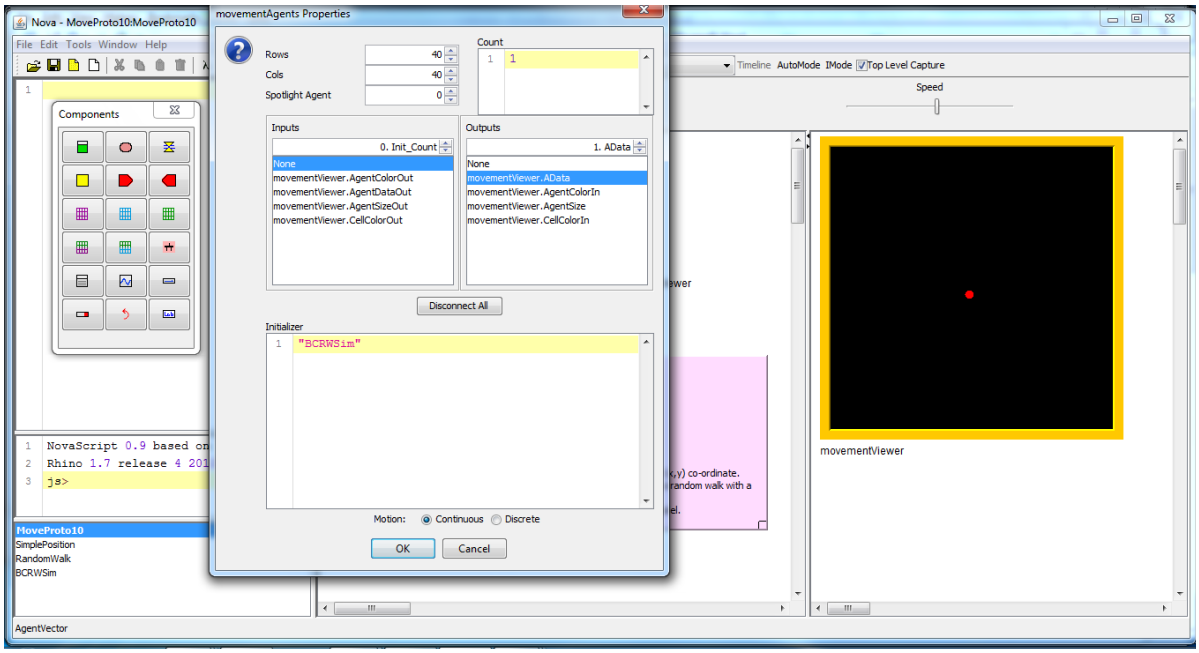
Tip: the commands should always be in CAPS.

Go back to the BCRWSim model: initialise the X stock and Y stock to initial values (terms):

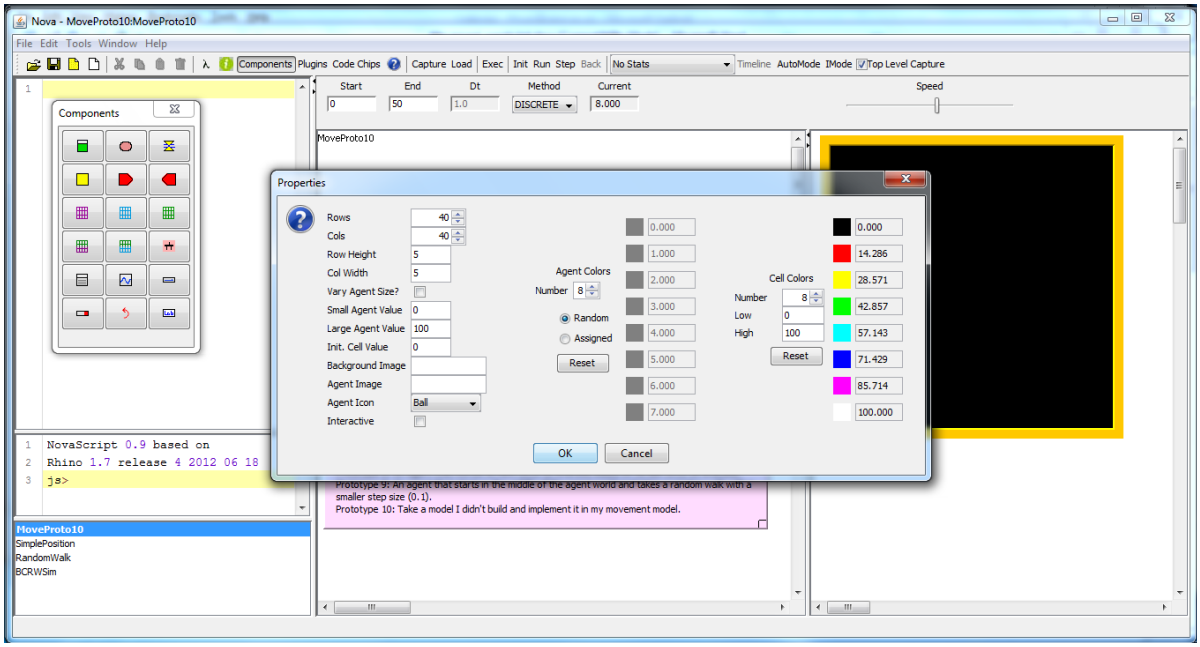




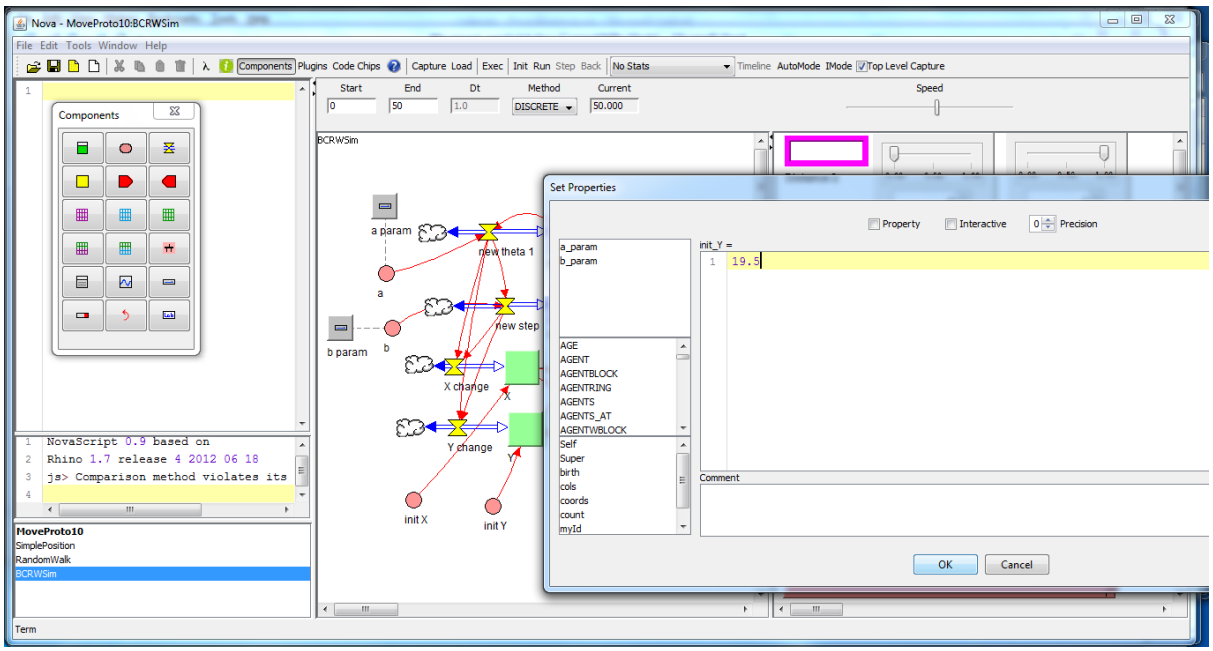
What about the size of the viewer? It depends on the values of a and b. It seems that 40x40 would be suitable values.



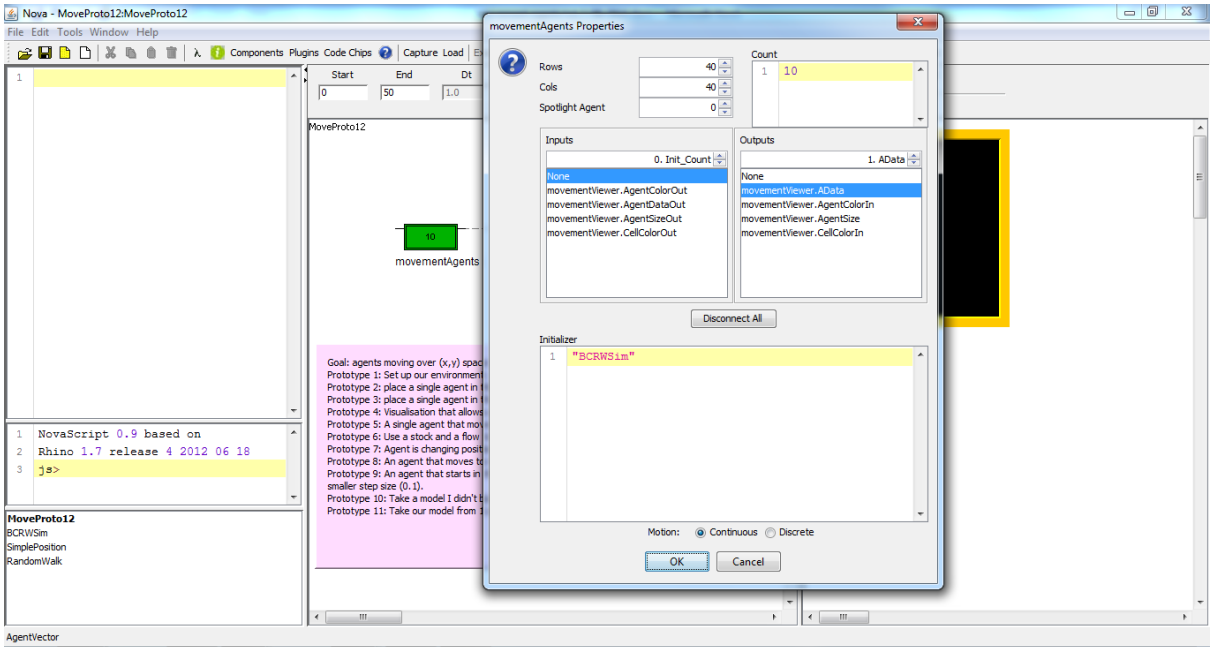
We need to also change the movementViewer properties by right-clicking on the outside border:



To make the agent start at the centre of the grid, change the init values for x and y to 19.5 each:

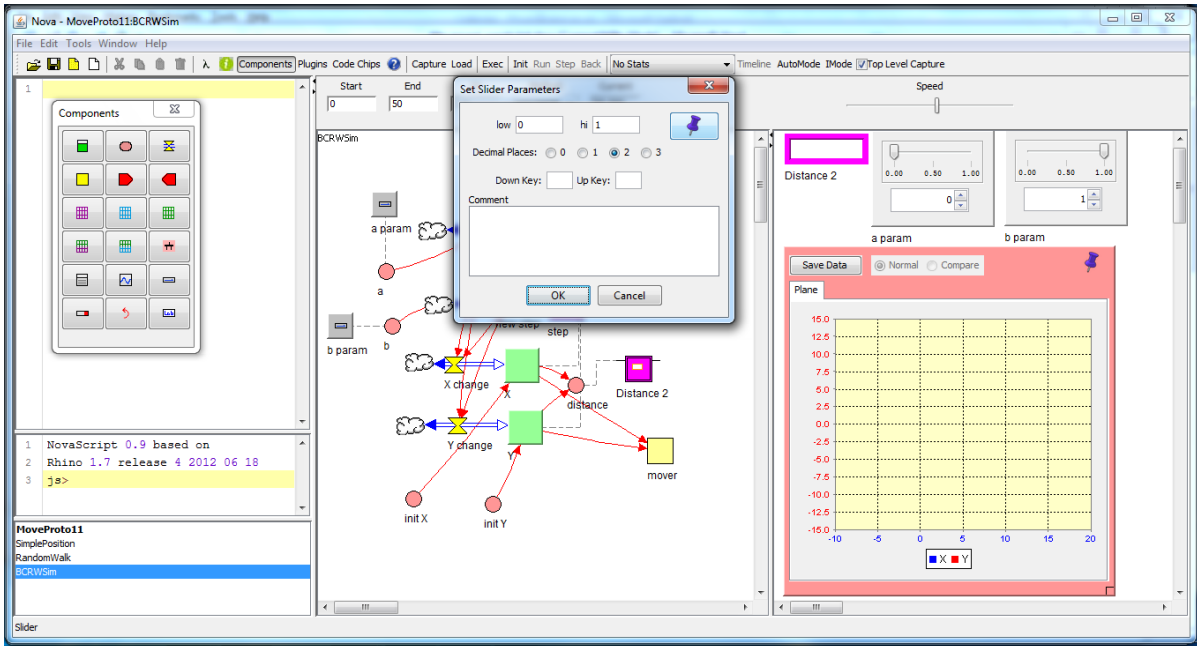


Prototype 11: expand the model to work for 10 agents (not just 1)

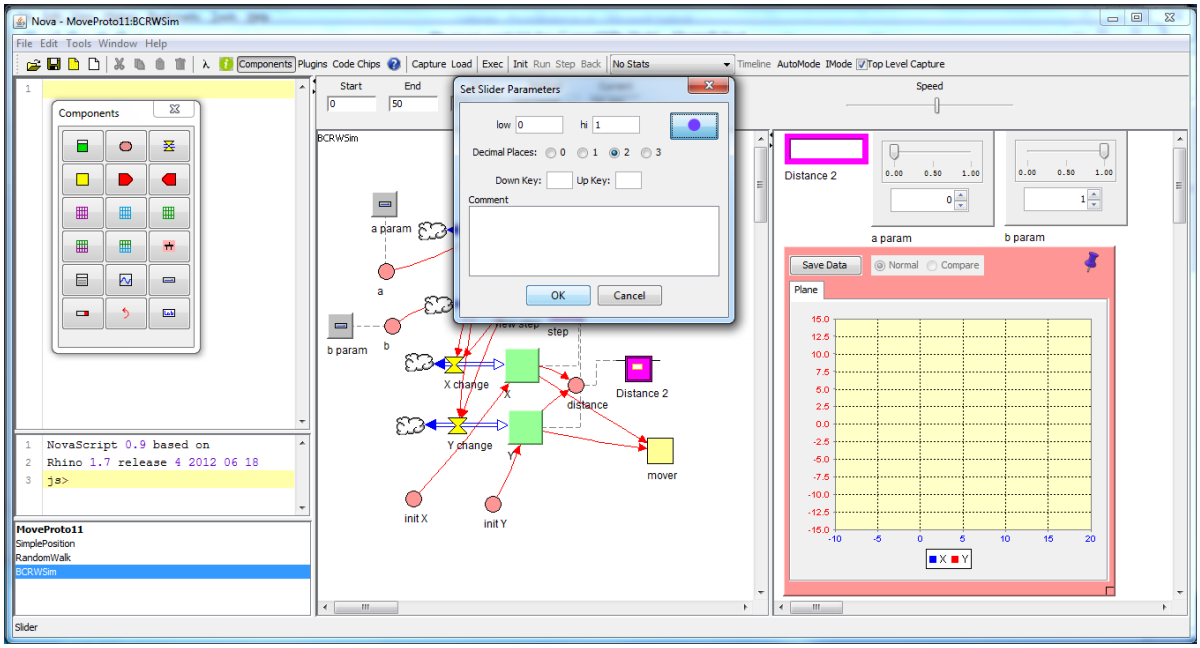


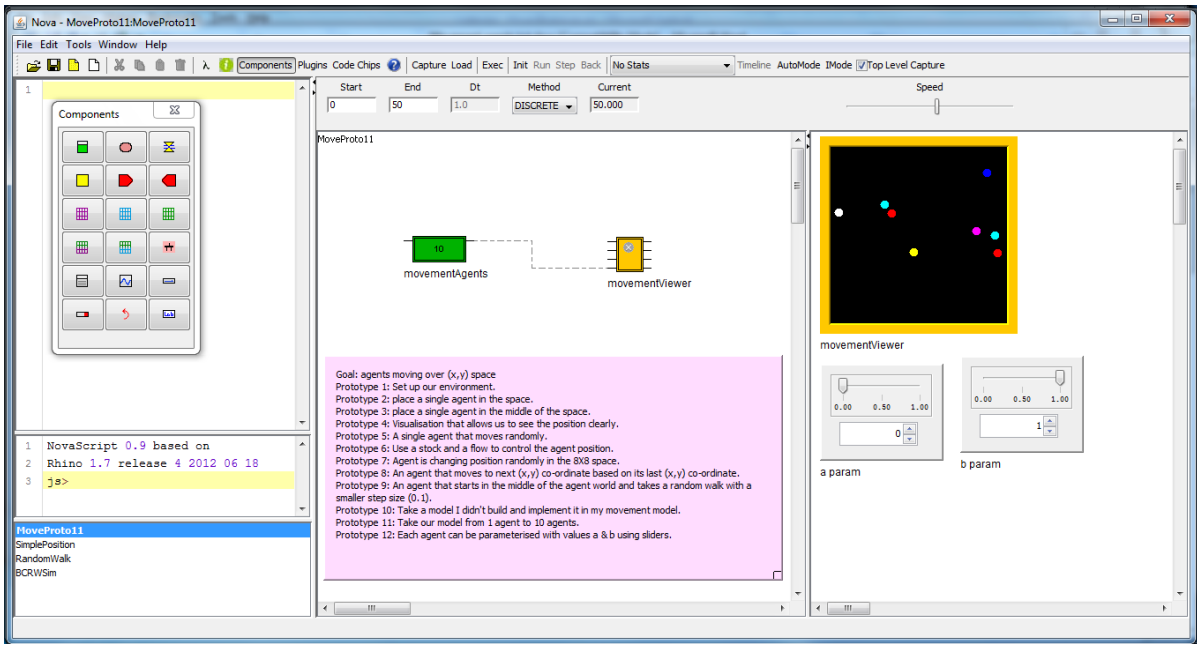
Prototype 12: each agent can be parameterised with values a and b using sliders

To create sliders for values a and b that are accessible to the top model:



Click on the pin button in the top right hand side of the Set Slider Parameters window:





NB Make sure that the integrate method for the model is DISCRETE.