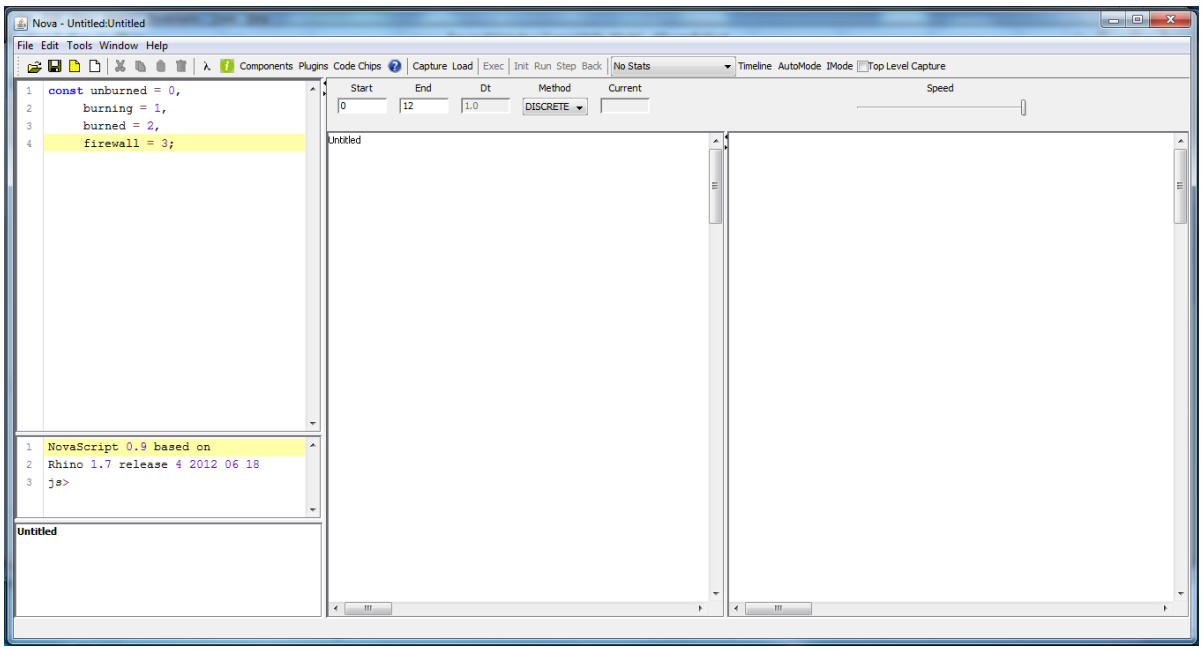
NOVA Fire model tutorial

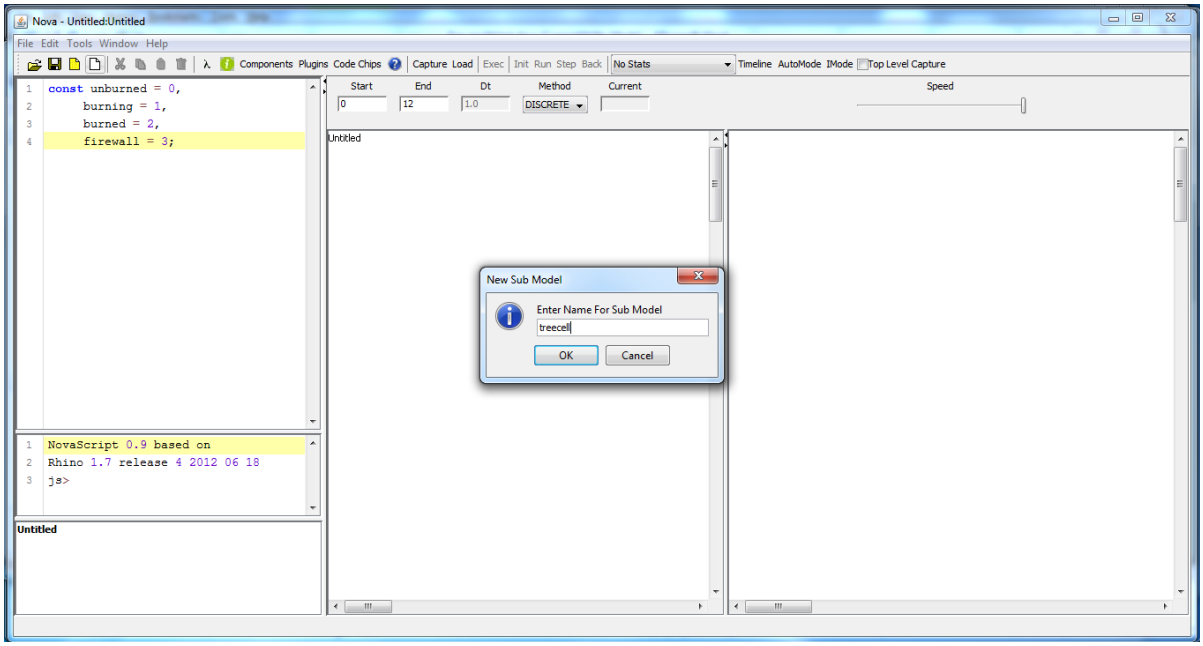
Notes taken by C. S. Price during R. Salter's tutorial, July 2014

This model is stochastic – i.e. each time you run it, you should get a different result.

First define the constants in the Programming Window (top left hand side): `unburned, burning, burning, firewall` (i.e. fire break). These are the states of the cells (patches).

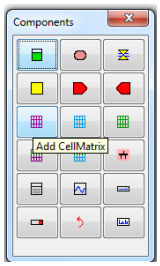


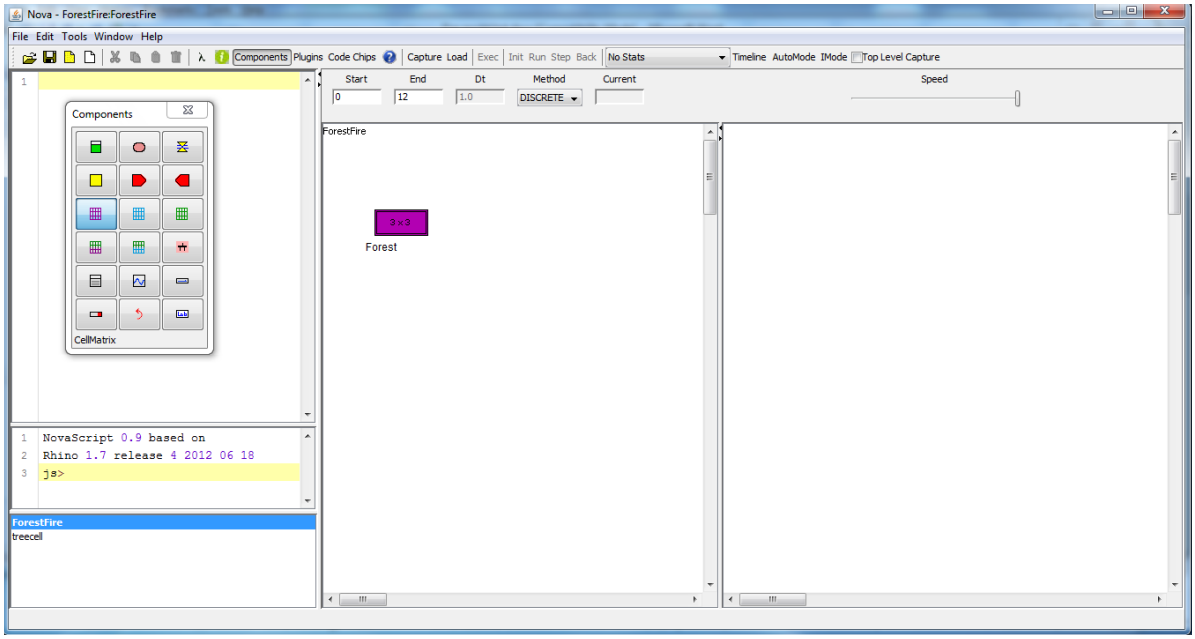
Create a submodel by clicking on the white "page" icon:



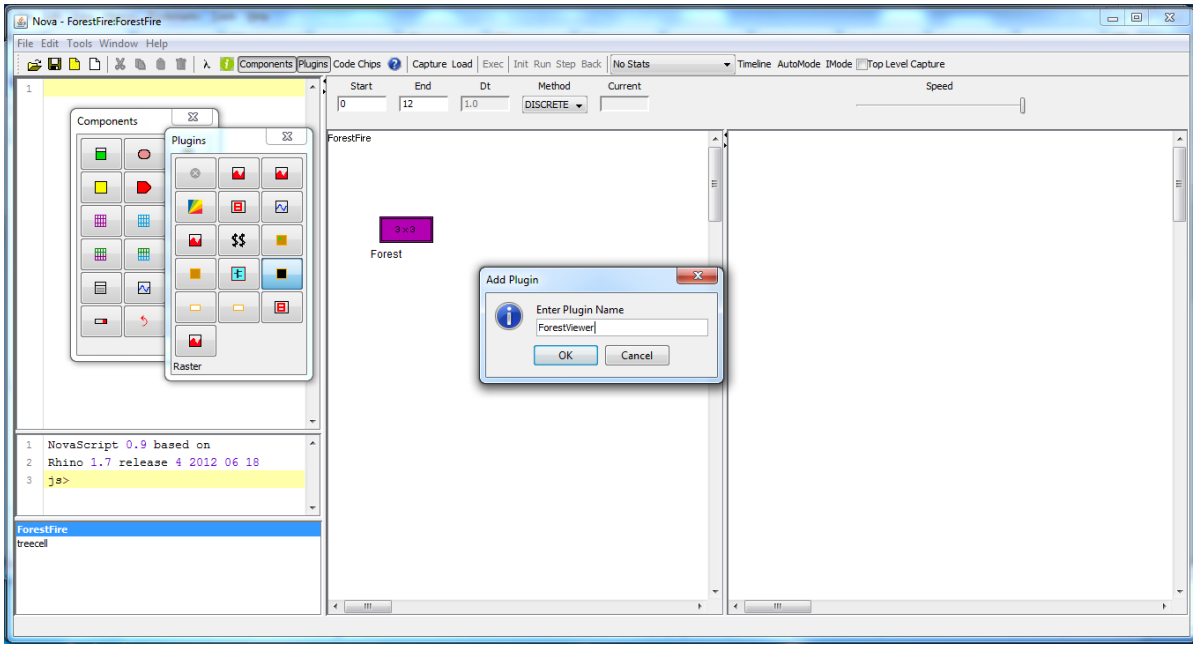
Save the model: call it ForestFire

Add a cellmatrix; call it Forest:

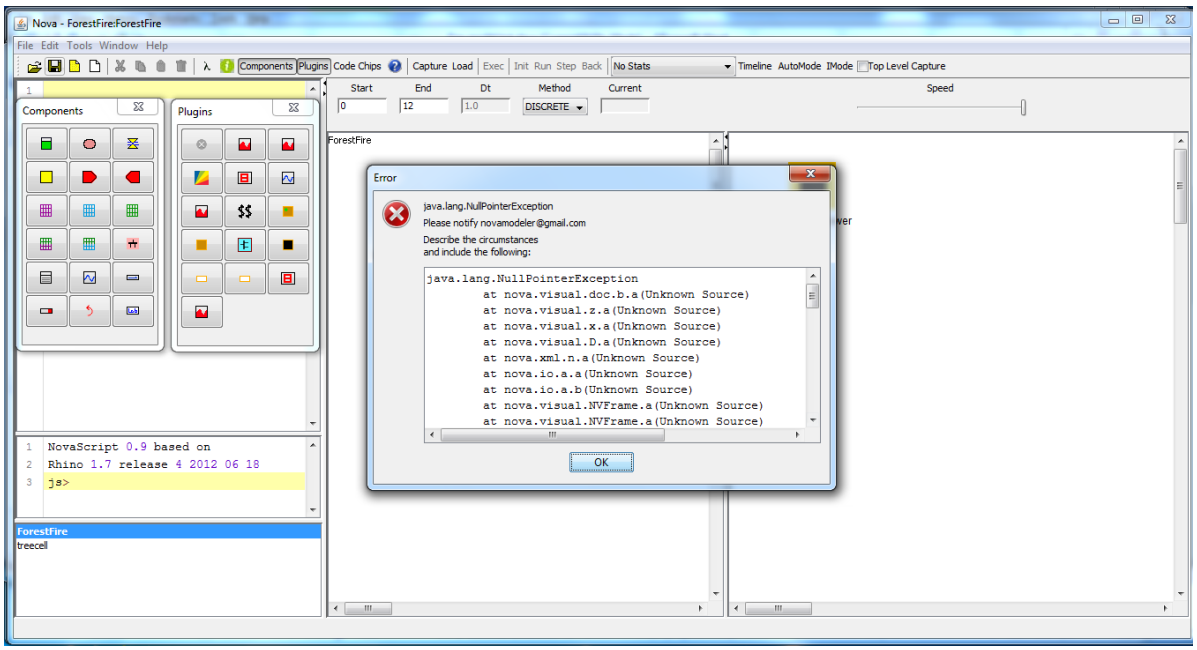




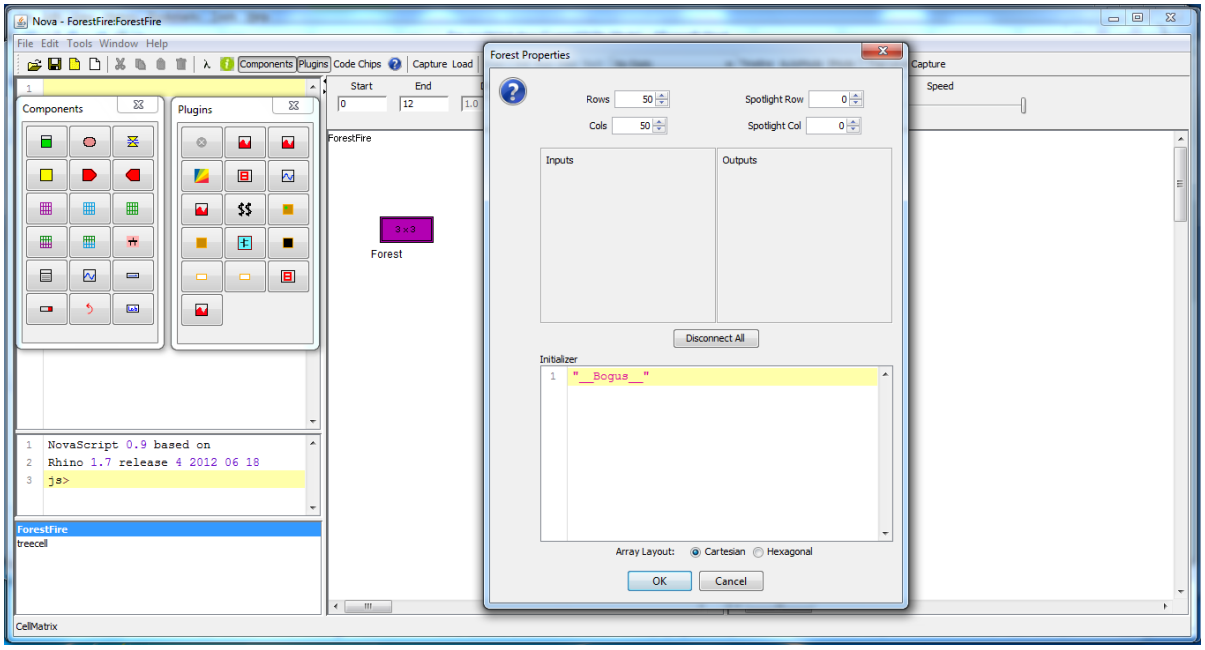
Add a raster viewer from Plugins; call it ForestViewer:



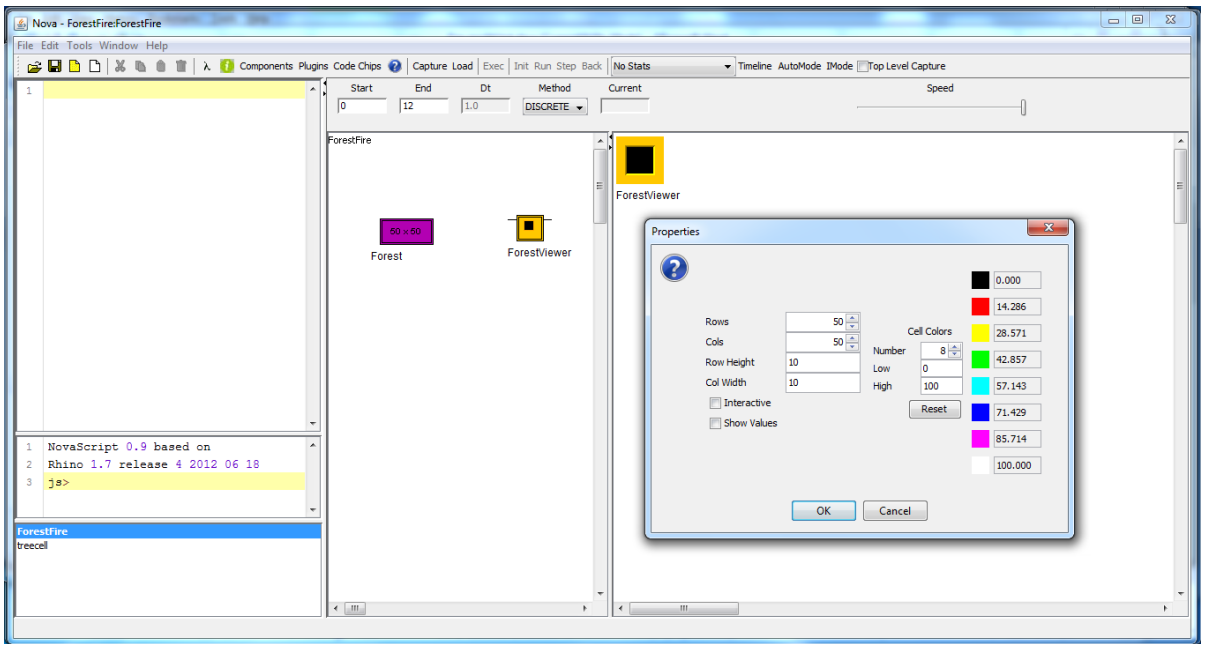
Saving at this point gives an error ☹



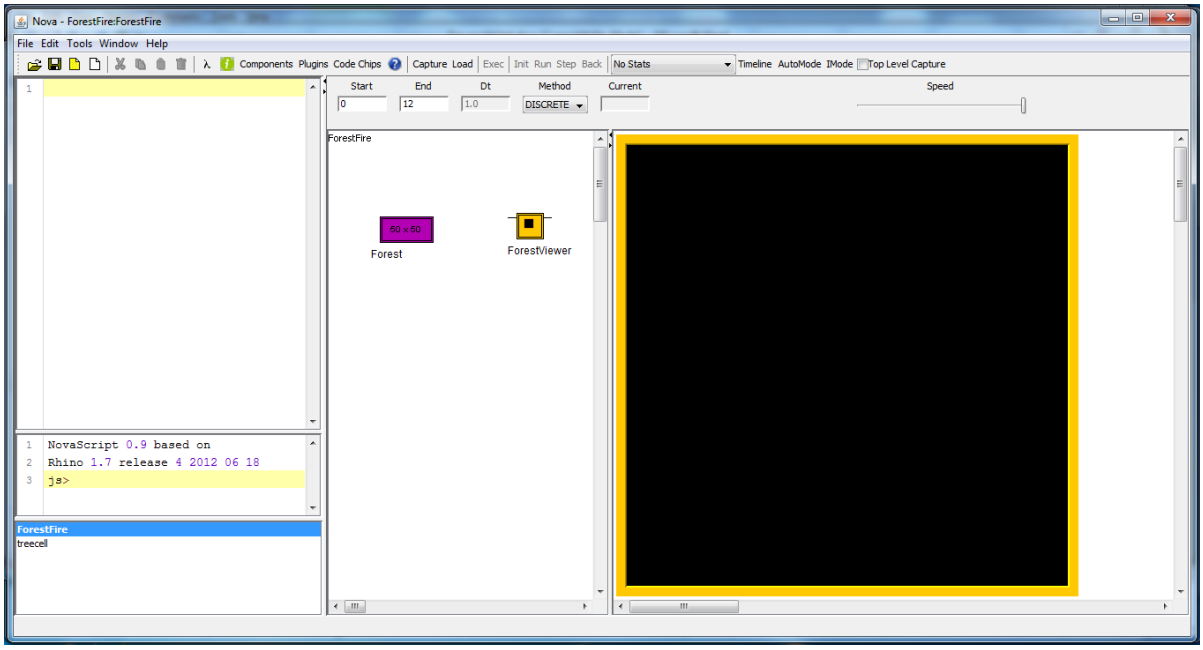
Need to dimension the viewer – in 2 places: first in the Forest (right-click on Forest to get its Properties):



Then in the ForestViewer on the right hand side – right click on the yellow coloured boundary:

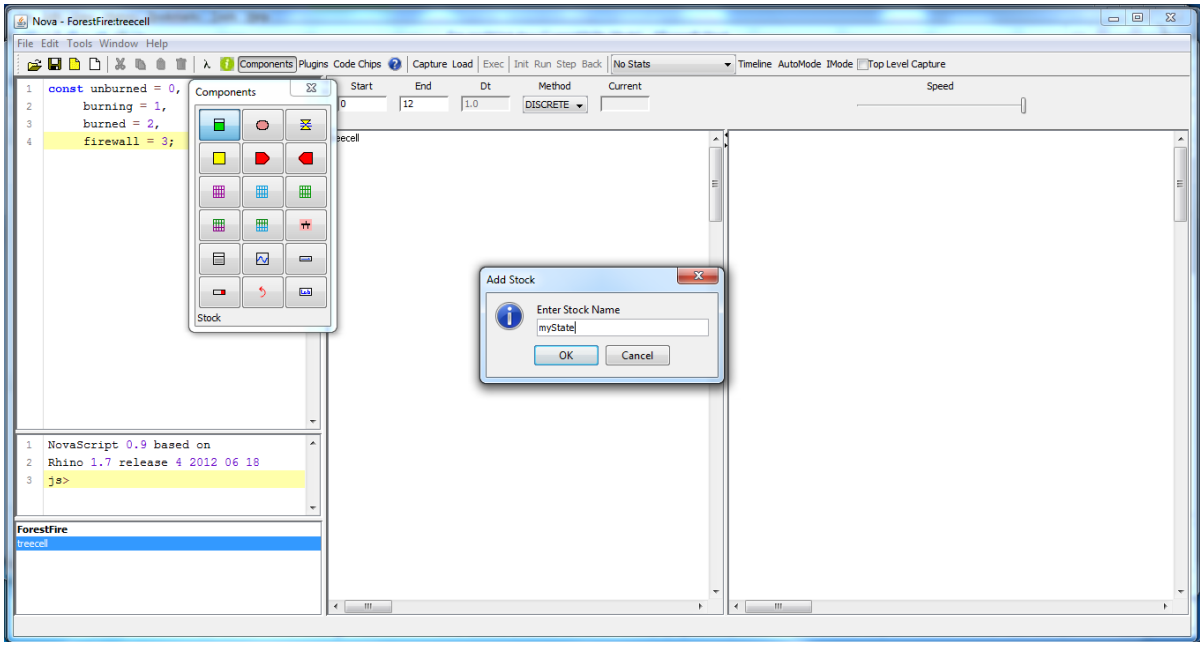


Which gives the right hand side:

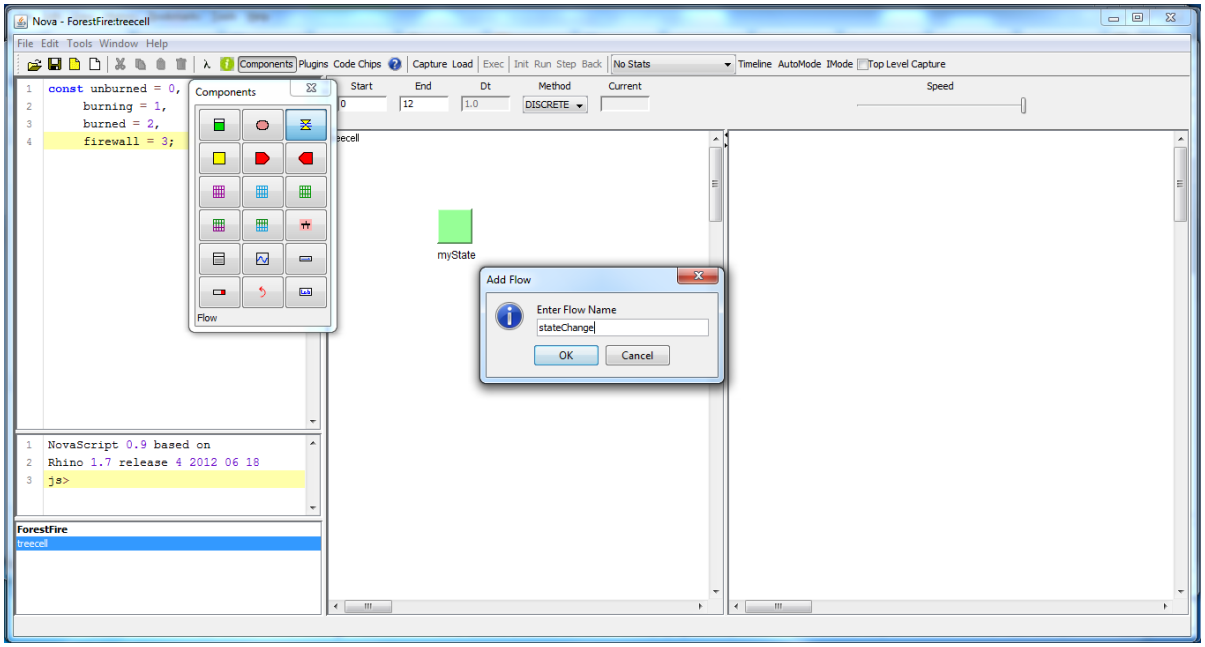


From the top level, drag the treecell model into the Forest. This fixes the bug ☺ and one can save the model.

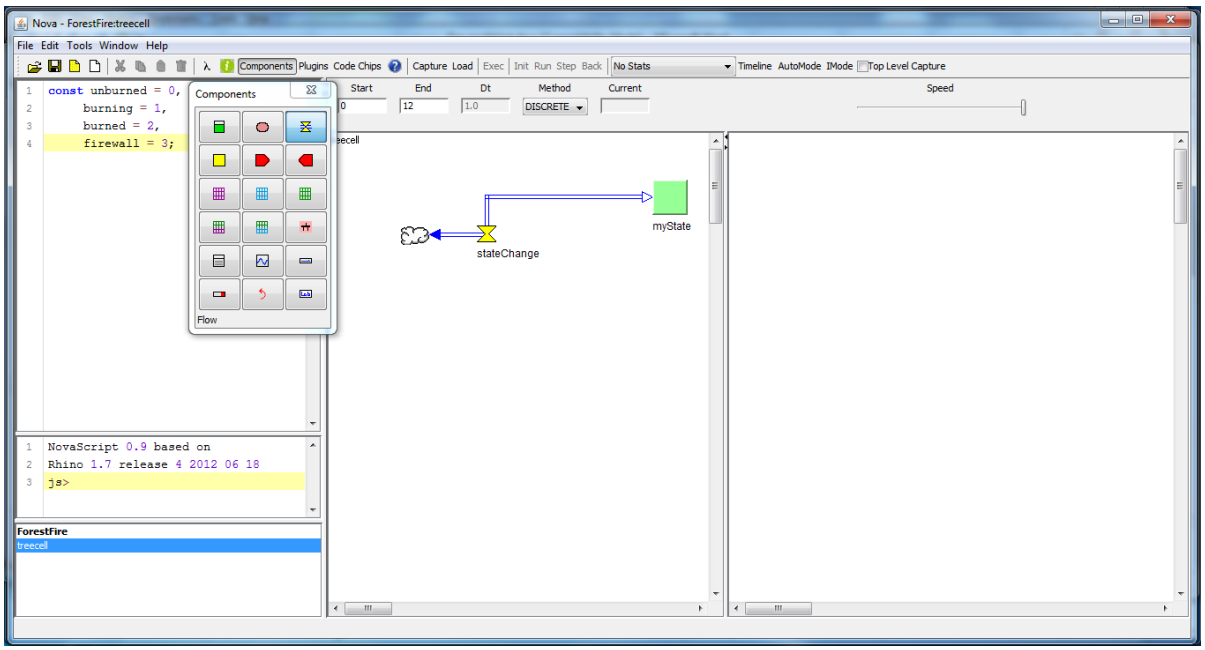
Go to the treecell model by clicking on it on the left hand side, and add a state:



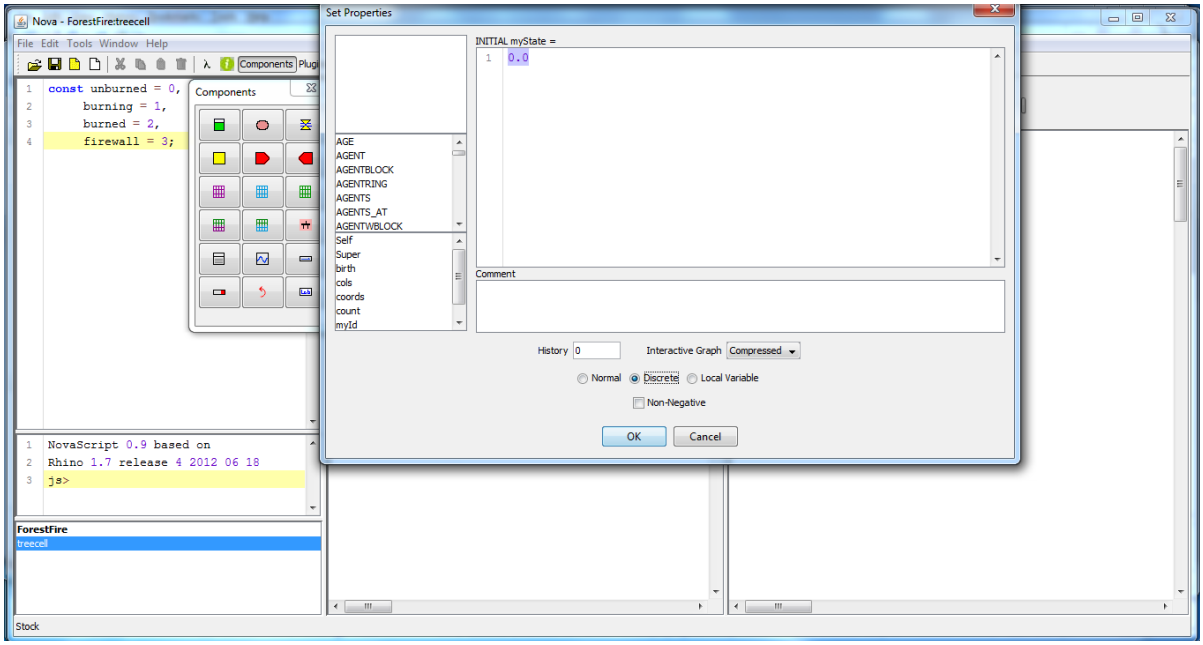
Add a flow:



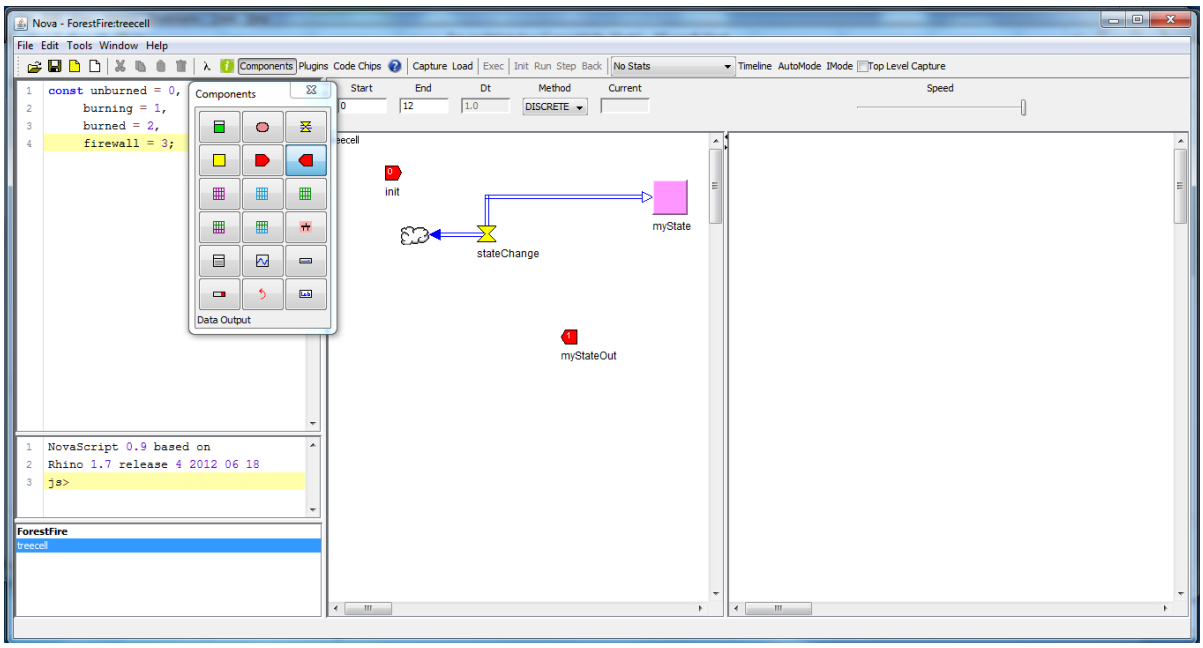
Link the state with the flow by dragging the point of the right hand flow arrow and dropping it onto the state.



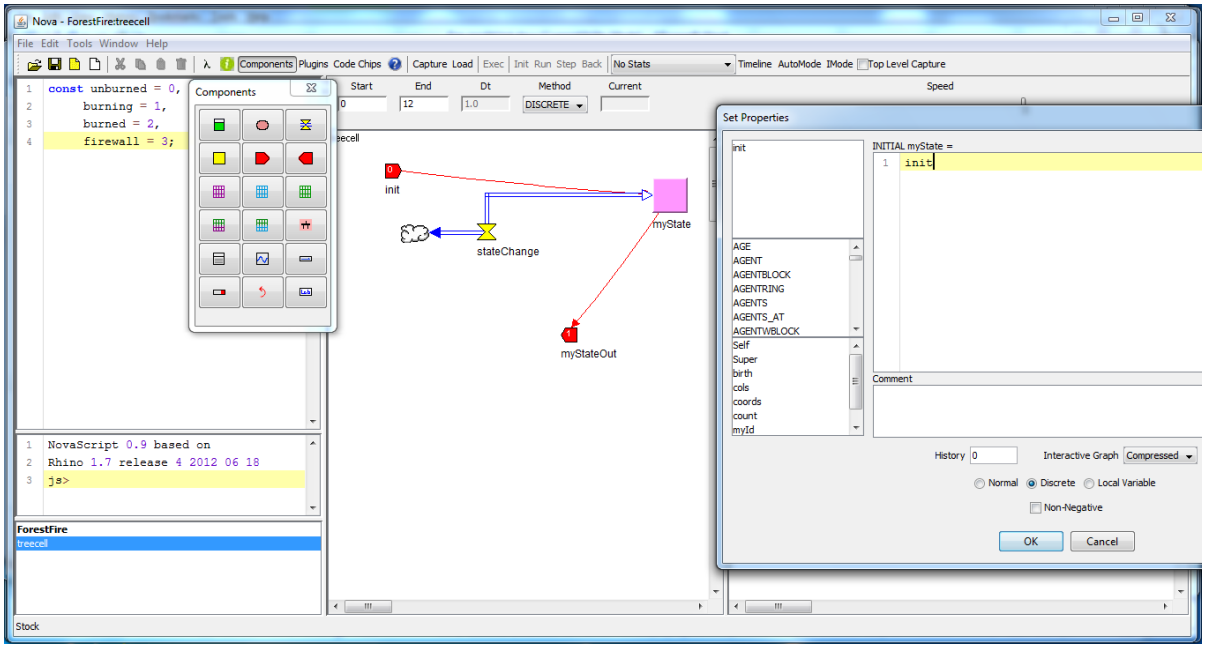
Make the state discrete:



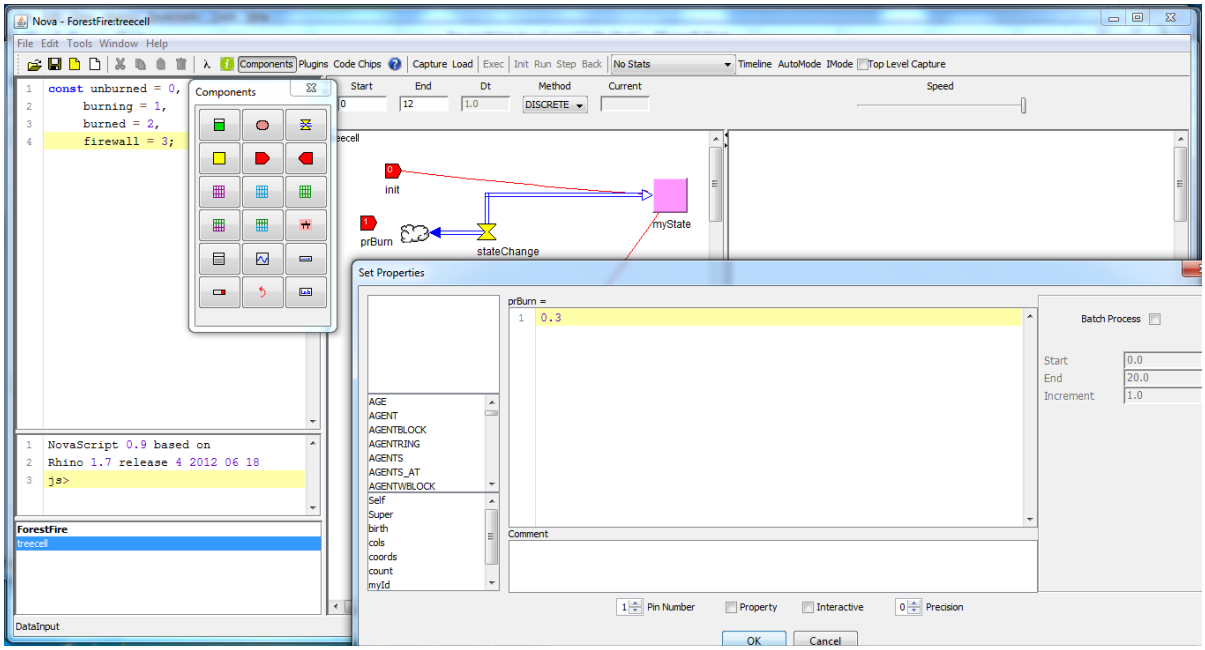
Add an input pin and an output pin (init) and an output pin (myStateOut):



Link them up with arrows (from Components menu, click on arrow, then click on the start and end component in turn). Make sure that the myState has an initial value of init.

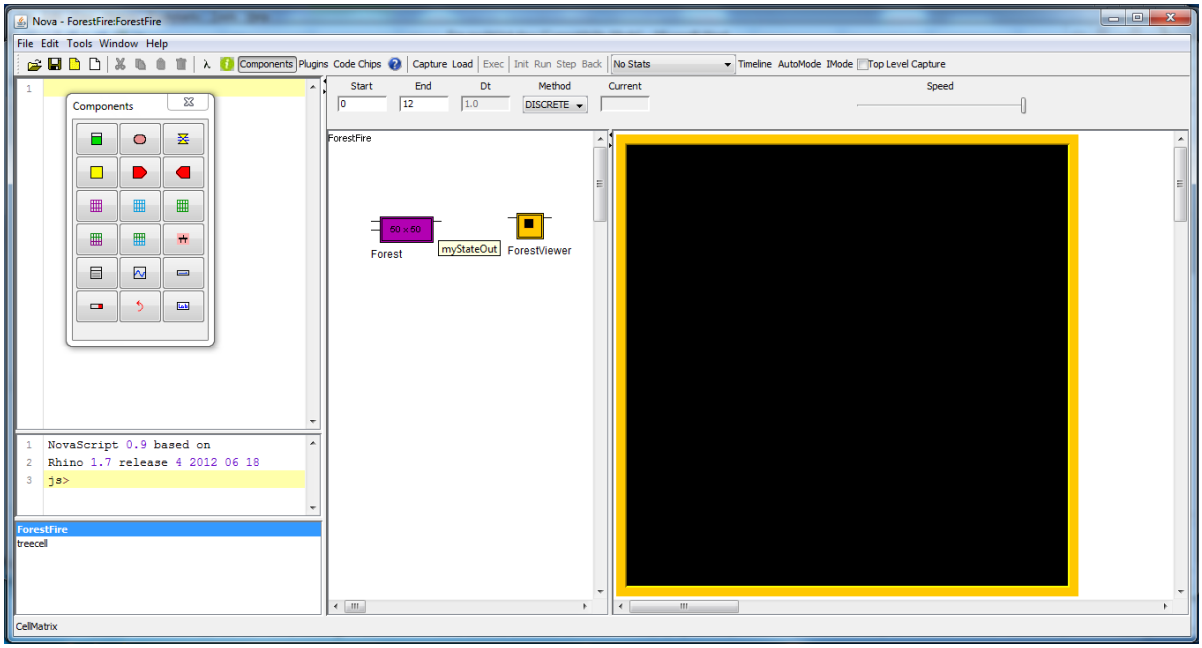


We need to add another input pin to denote the probability of burning, and set its initial probability to 0.3:

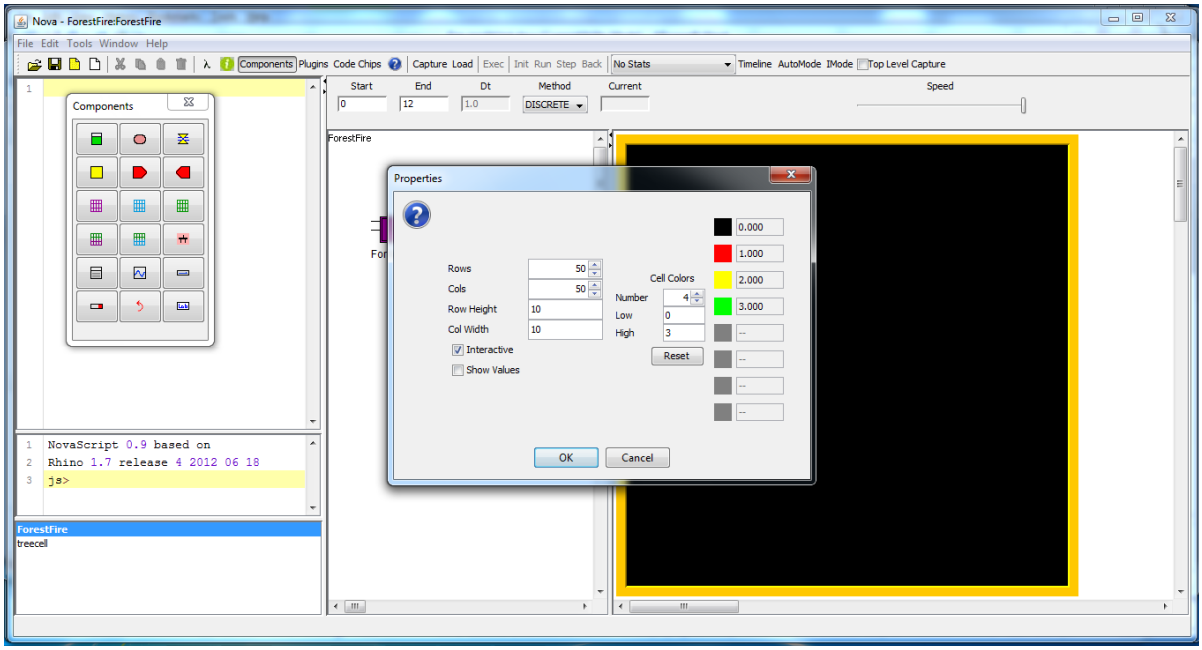


Go to the top level:

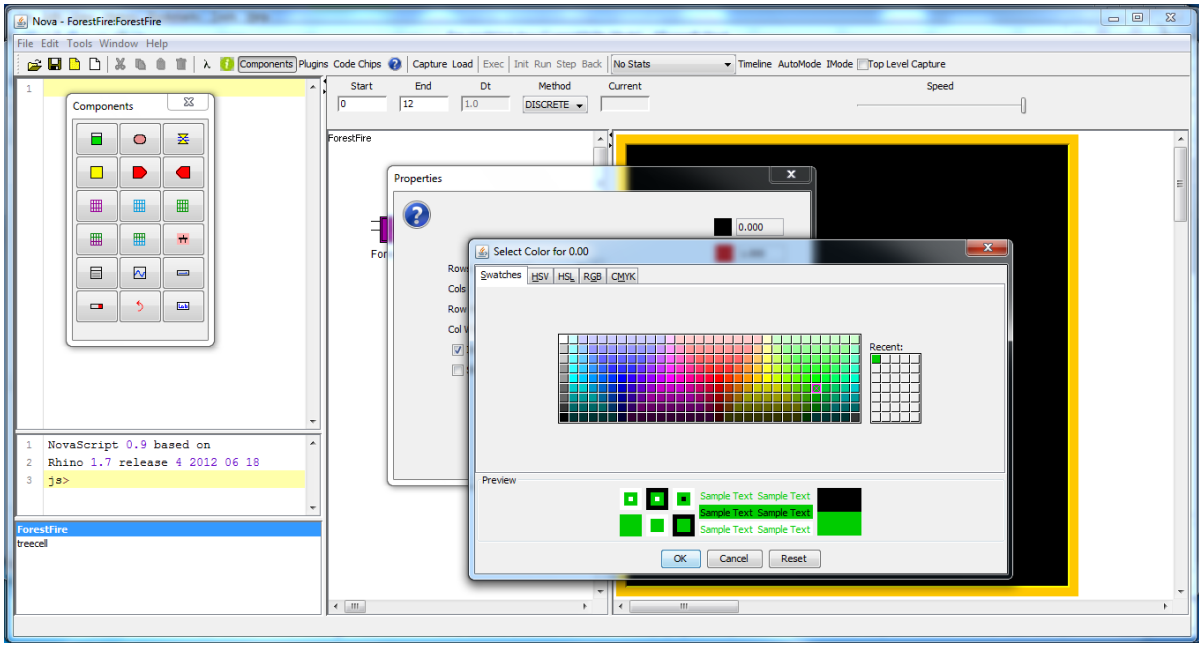
The cellmatrix Forest has grown 2 pins, according to the pins in the submodel; the screenshot shows the mouse hovering over the output pin:



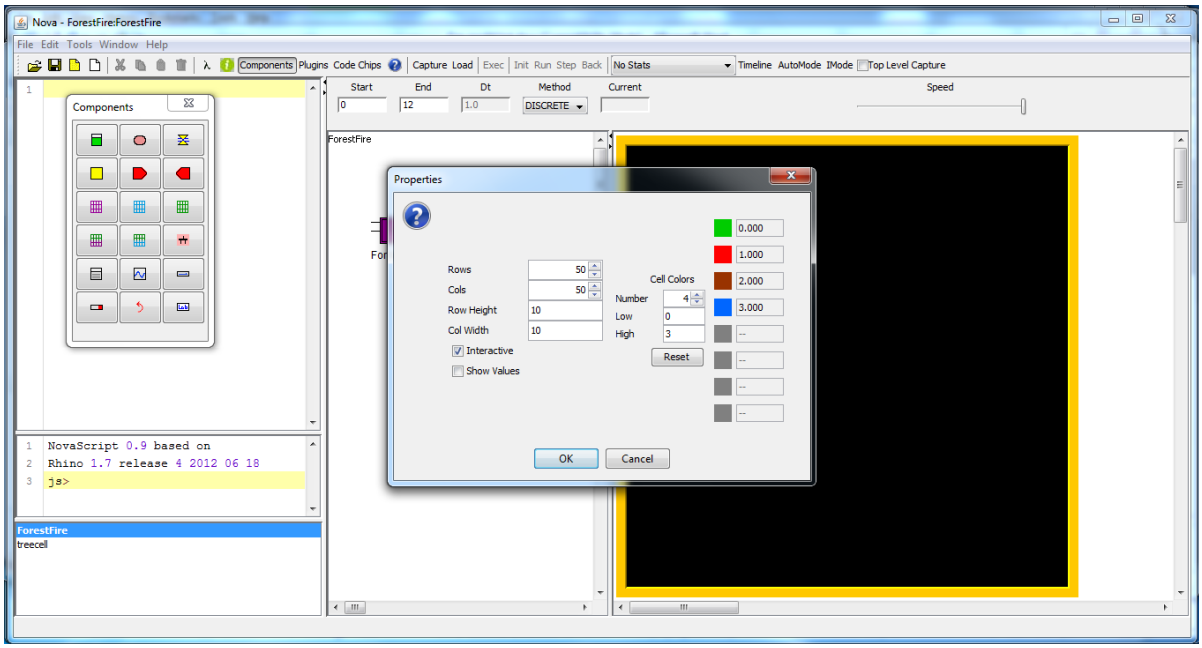
On the edge of the viewer on the right hand side, click on Interactive to allow the user to set trees, burning trees, burned trees and firewalls. Choose 4 colours: and let them run from 0 to 3:



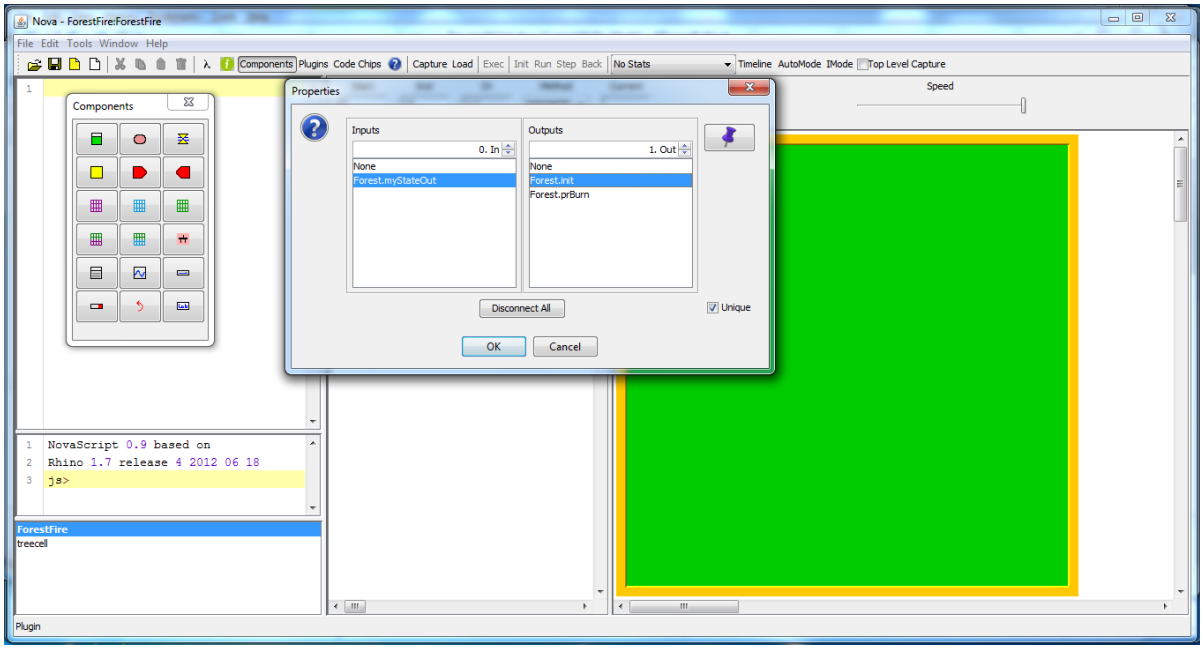
Need to change the colours: 0 (unburned) should be green – forest. Burned (value 2) should be charred brown. Firewall (value 3) should be a different colour.



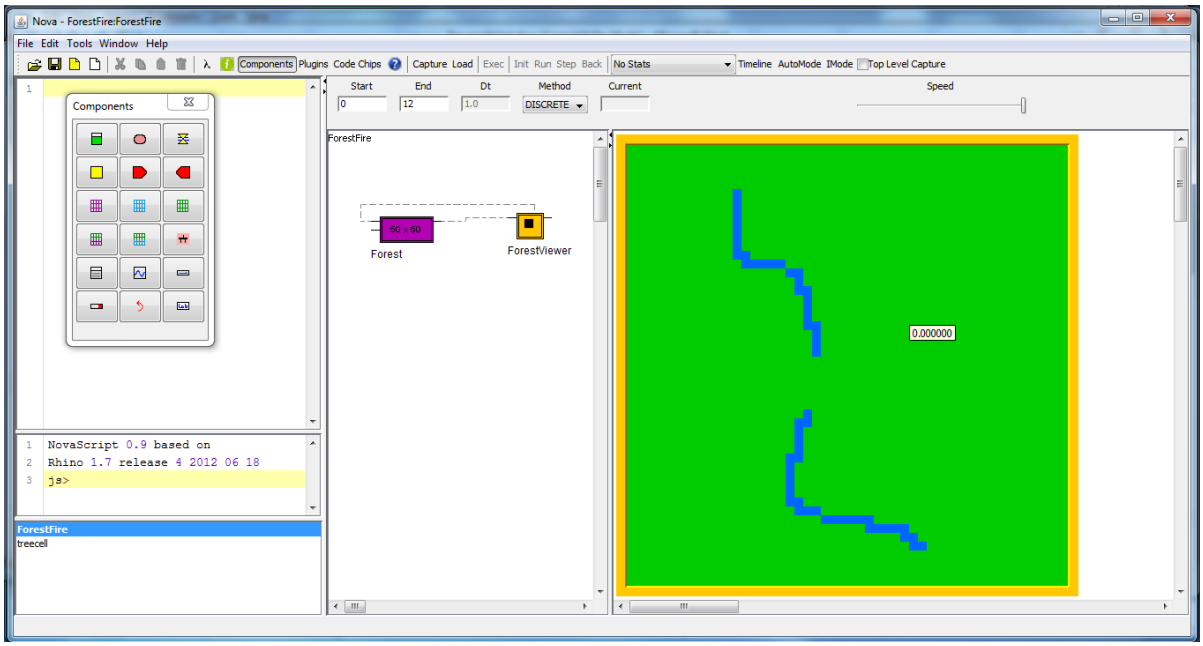
This results in the following screenshot:



Now right-click on ForestViewer: change the input to Forest.myStateOut and the output to Forest.init. Also tick Unique.

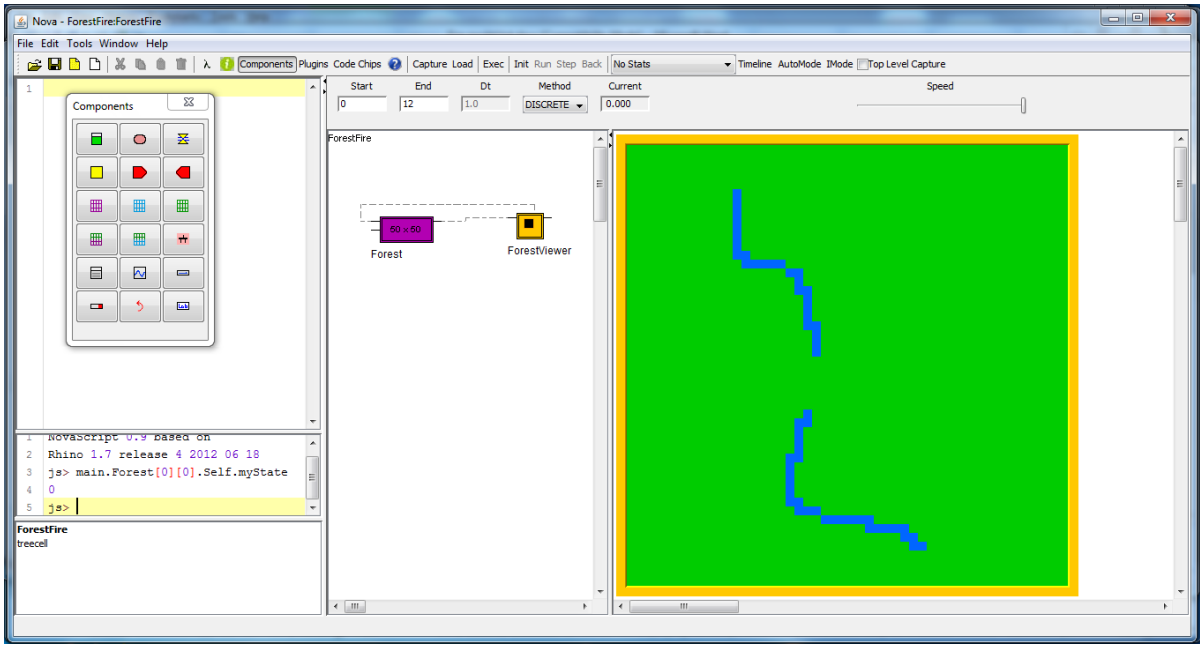


On the dashboard (right hand side of the screen), you can add a firewall and some burned trees etc. If you click on a patch, it will have value 0 (unburned). Clicking on the same patch will make it 1 (burning). Clicking on the same patch will make it 2 (burned), etc. Whatever the last colour was, if you drag your mouse over the patches, they will change into the colour of the last change.

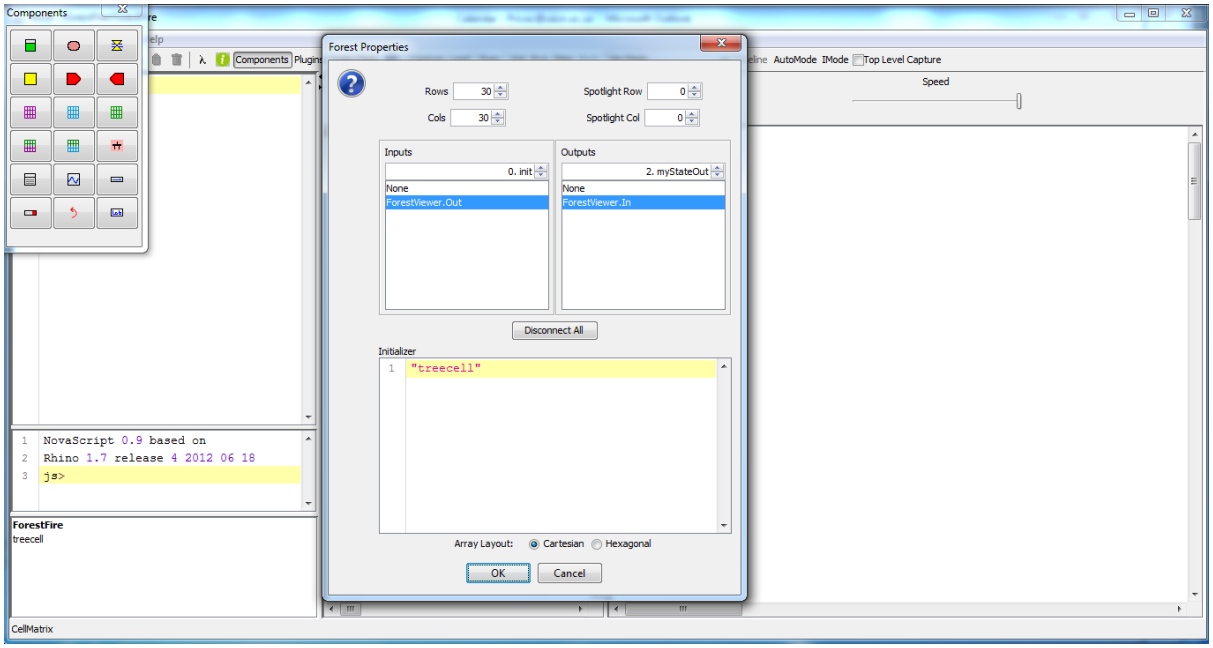


You can check the values of the patches in the console (left hand side, middle section of the screen):

```
Main.Forest[0][0].Self.myState
```

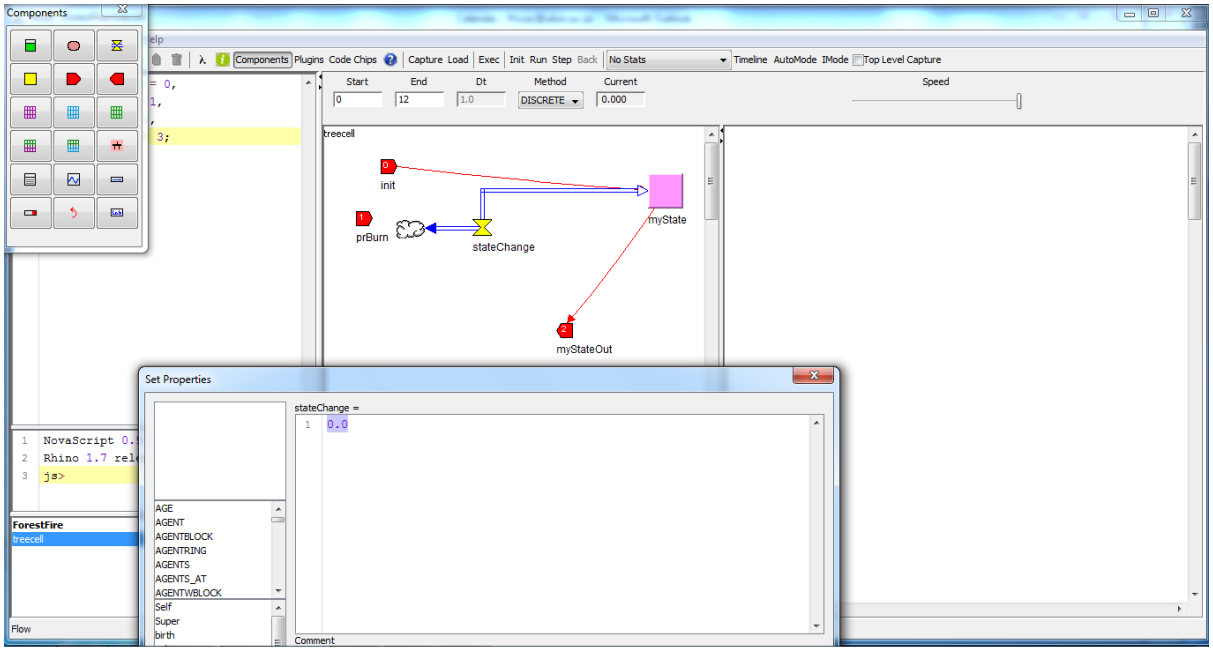


To make sure that the model initialises fast enough, change the size of the Forest CellMatrix to 30 x 30:

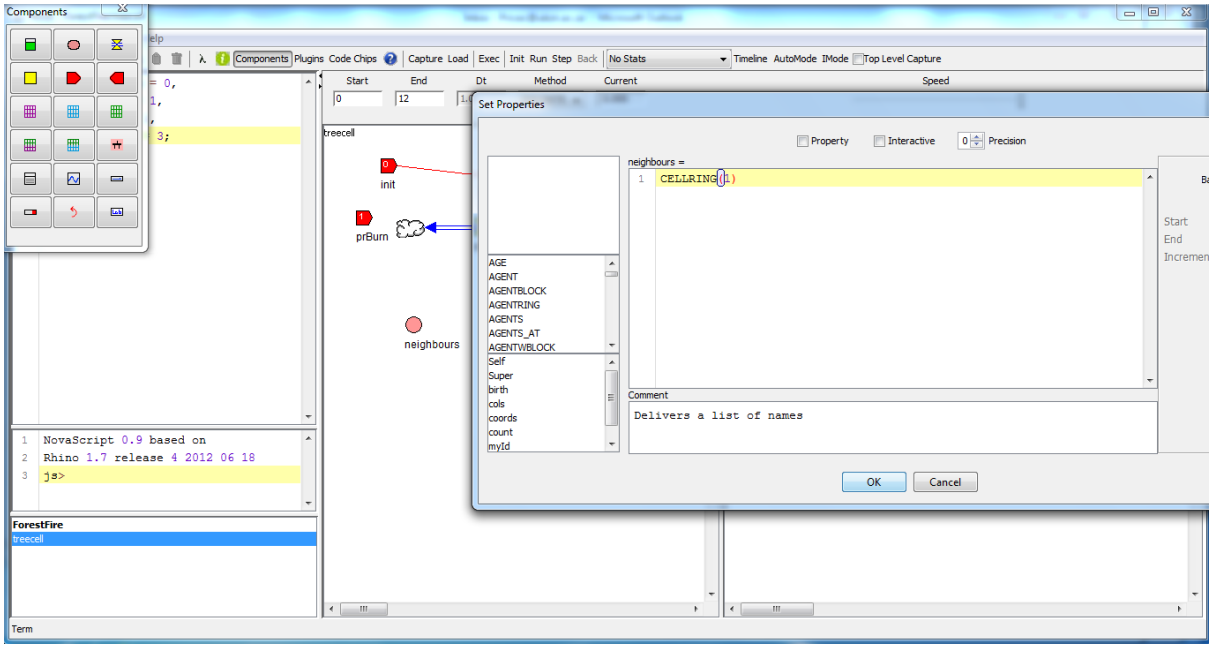


Click on Capture Load and Init to resize the grid on the dashboard (right hand side of the screen).

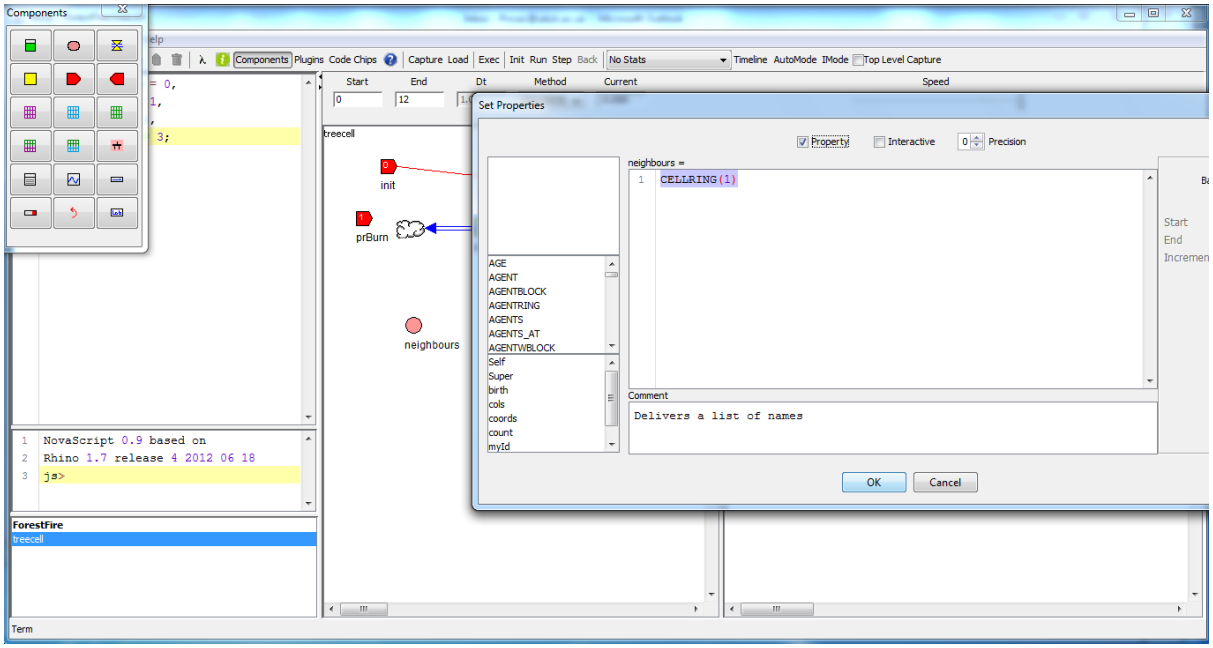
In the treecell submodel, add a term called neighbours.



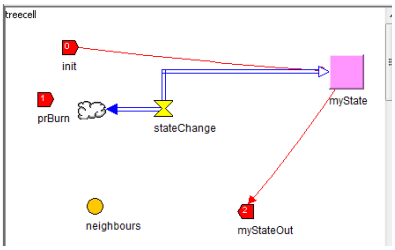
In the term neighbours, add `CELLRING(1)` which returns a list of cells surrounding the current cell.



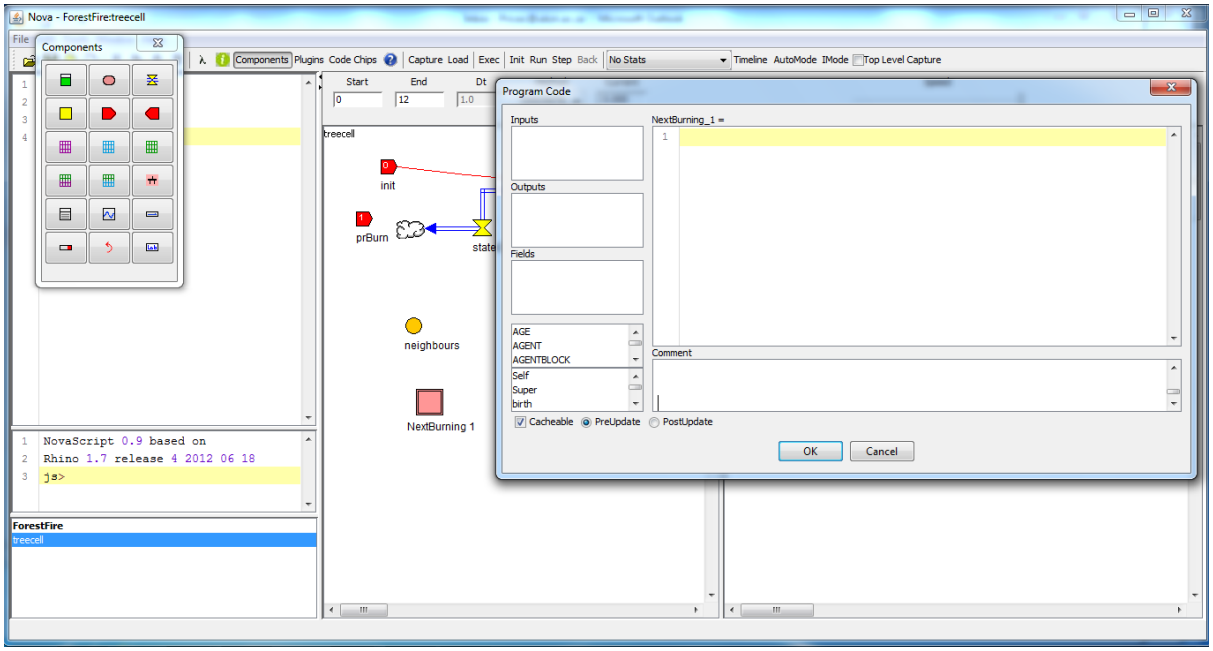
Since each cell in the matrix has the same number of neighbours for each time step in the simulation, and the neighbours don't change over time, it would be sensible to calculate this number only once (to save computing time). Do this by ticking the Property field at the top:



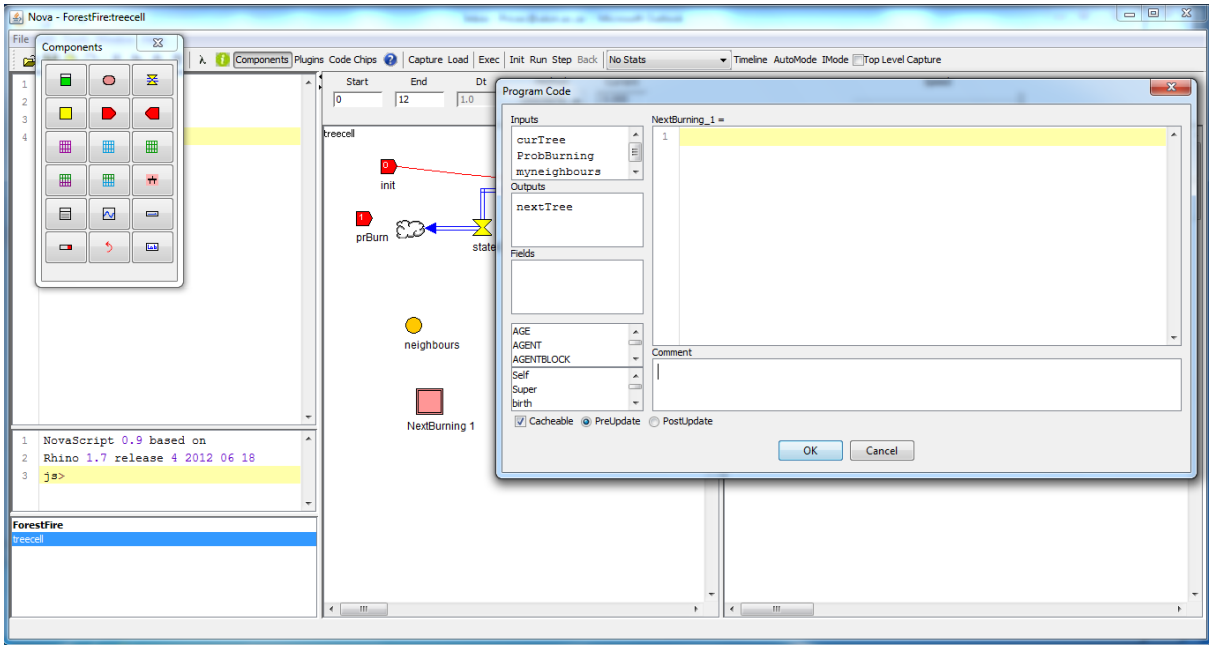
This changes the colour of the term neighbours to yellow.



Add a new CodeChip called NextBurning:



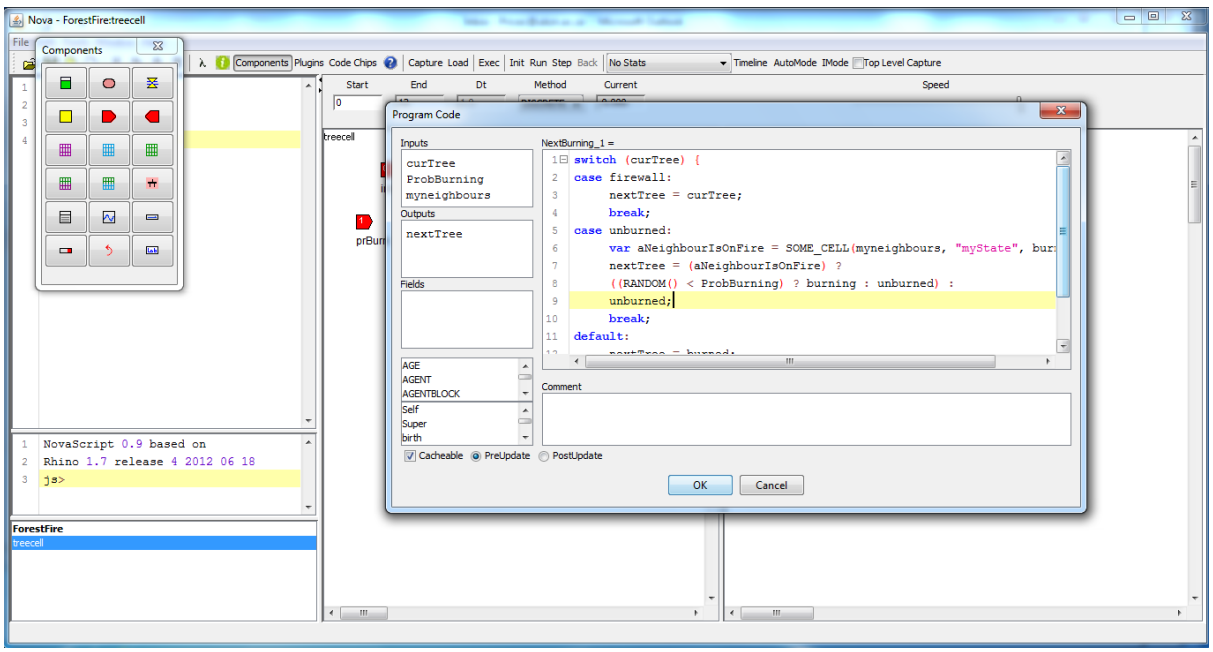
Adding the inputs (`ourTree, ProbBurning, myneighbours`) and the output (`nextTree`) adds pins to the chips:



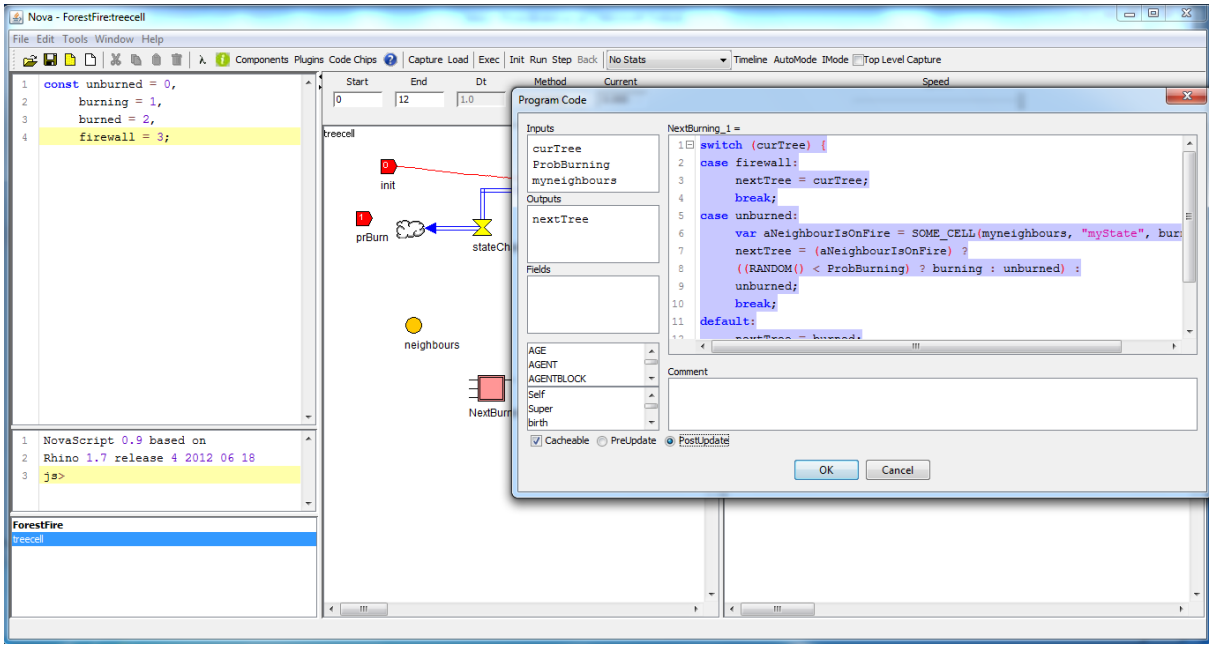
Add the code in the right hand pane:

```
switch (curTree) {
case firewall:
	nextTree = curTree;
	break;
case unburned:
	var aNeighbourIsOnFire = SOME_CELL(myneighbours, "myState", burning);
	nextTree = (aNeighbourIsOnFire) ?
	((RANDOM() < ProbBurning) ? burning : unburned) :
	unburned;
	break;
default:
	nextTree = burned;
	break;
}
```

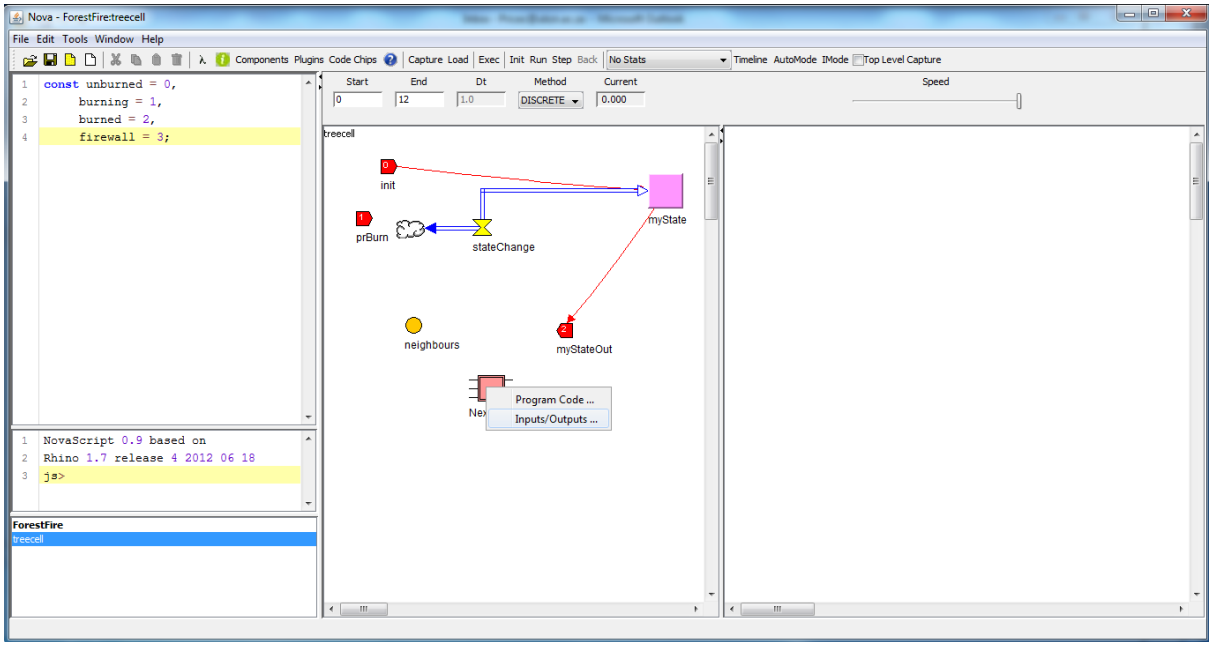
Note that “myState” must match the name of the stock in the treecell model. The inputs and outputs must also be spelled in the code in the same way as they are declared in the left hand boxes.



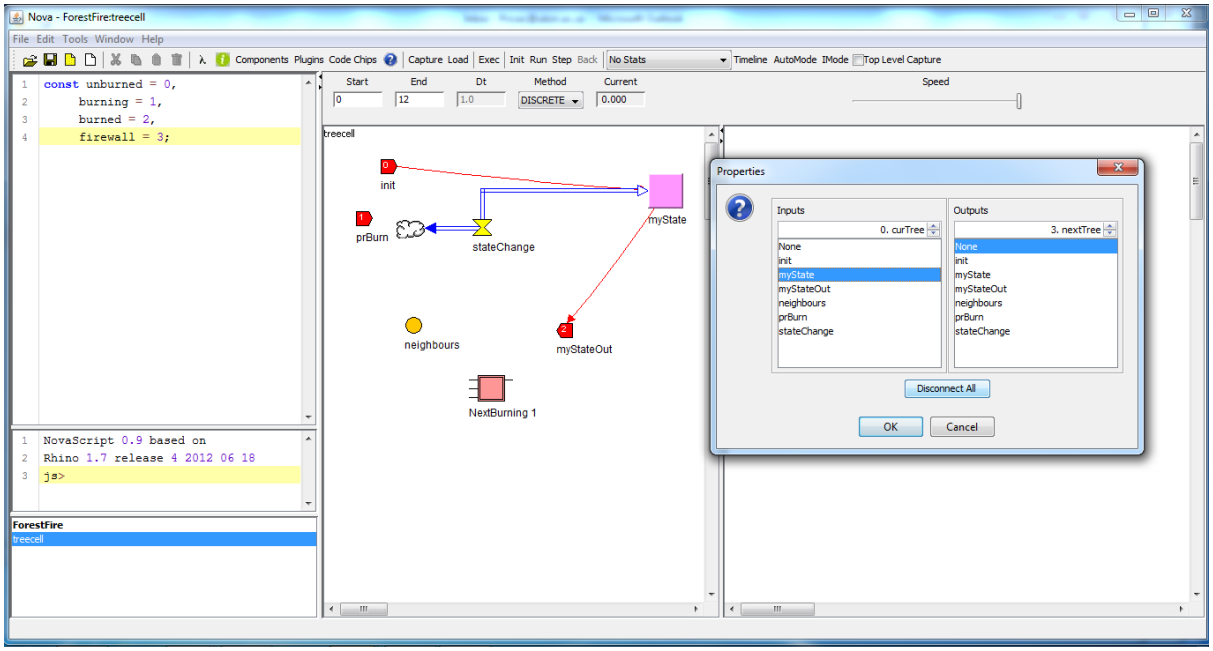
Remember that the code in the codechip must be evaluated PostUpdate.



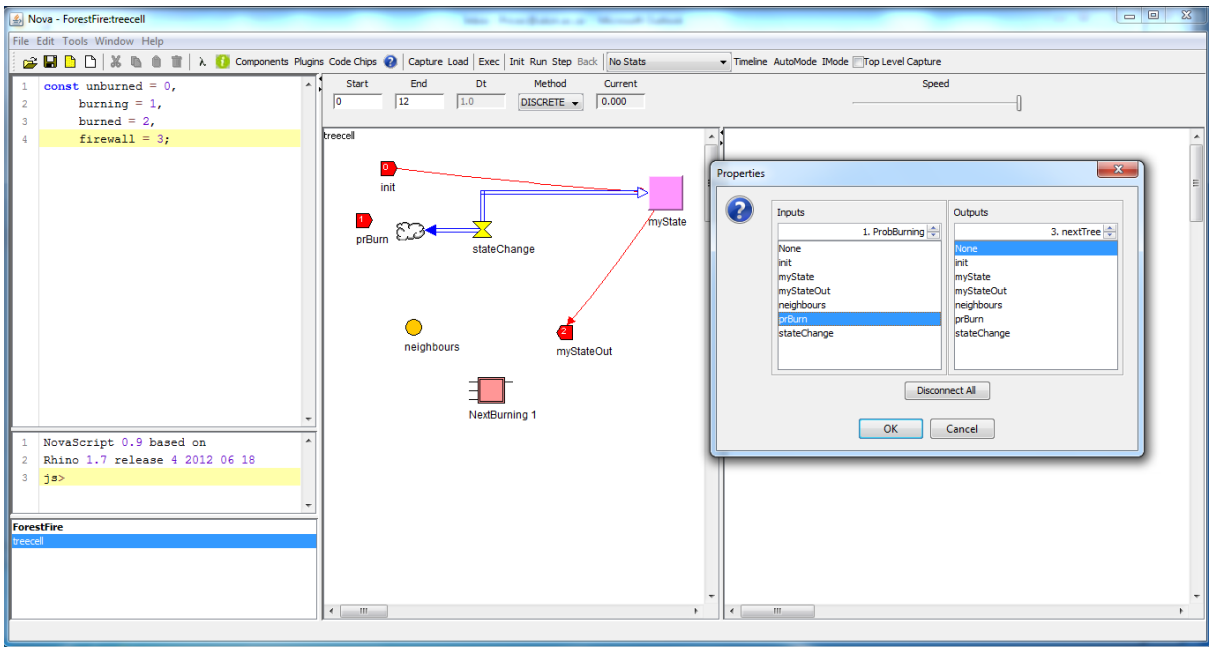
Now we need to attach the input and output pins of the codechip to the rest of the model. Right-click on the codechip and choose Inputs/Outputs...:



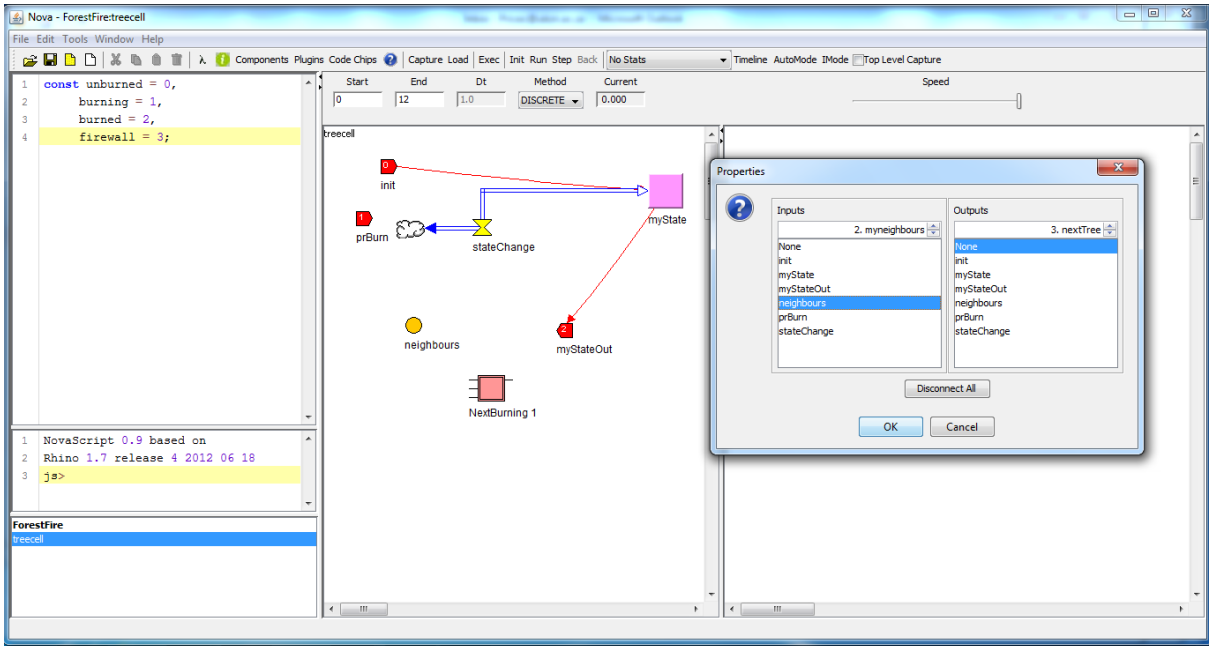
Linking input `curTree`:



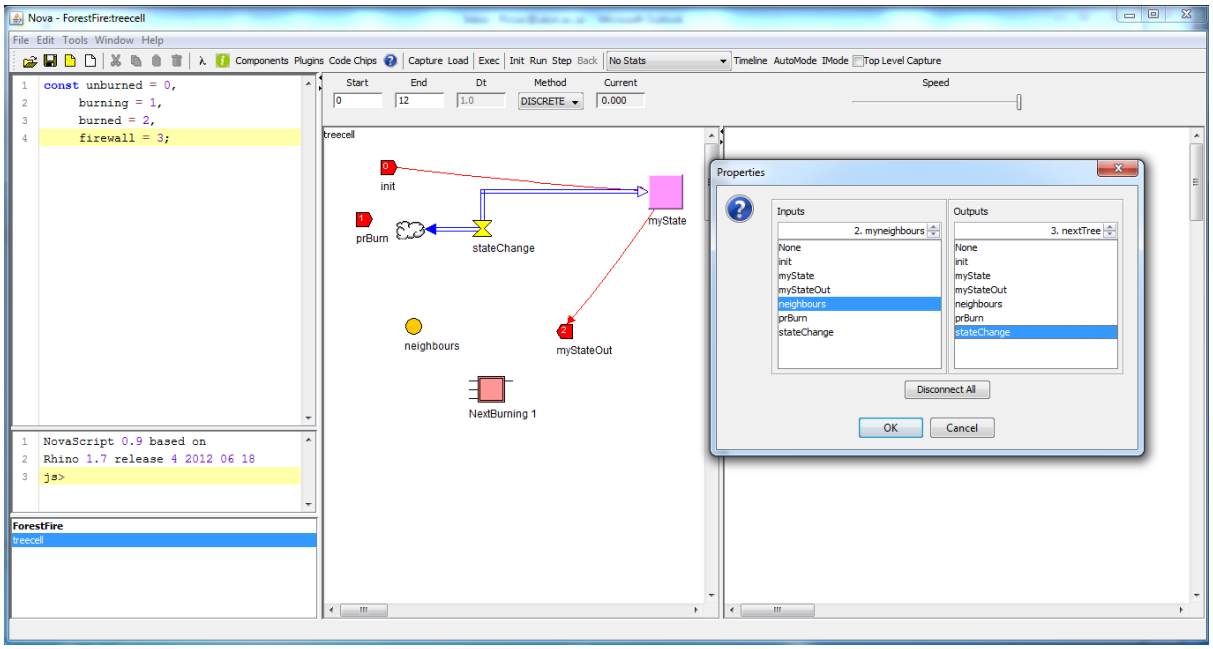
Linking input `ProbBurning`:



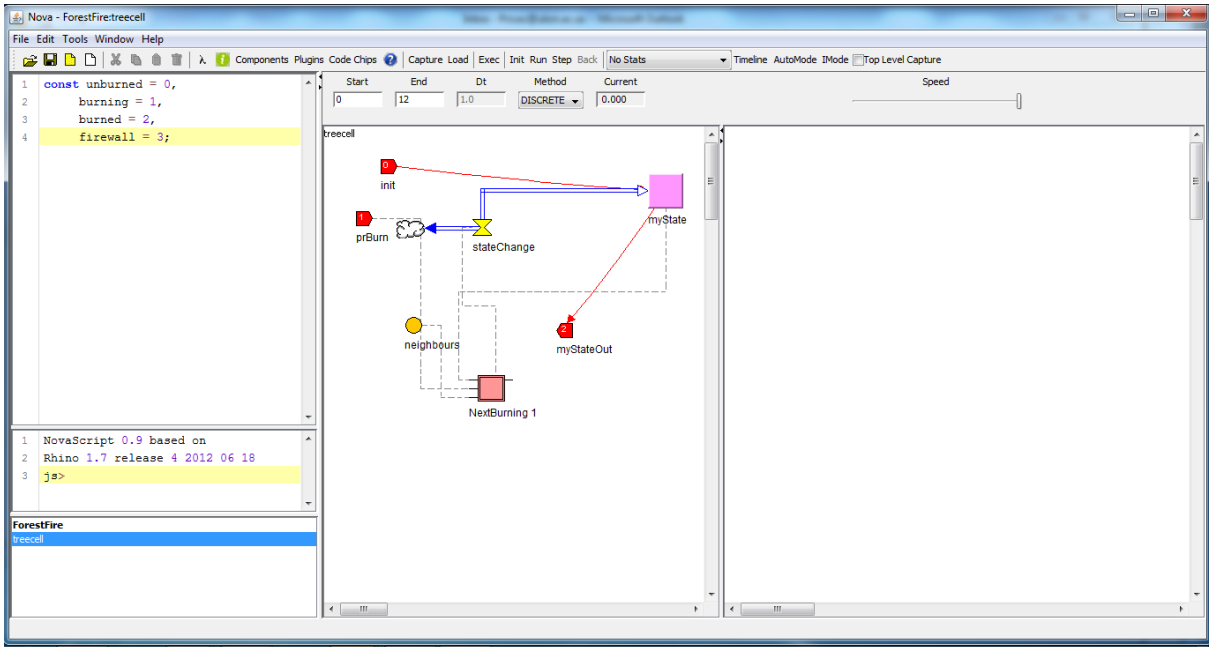
Linking input `myneighbours`:



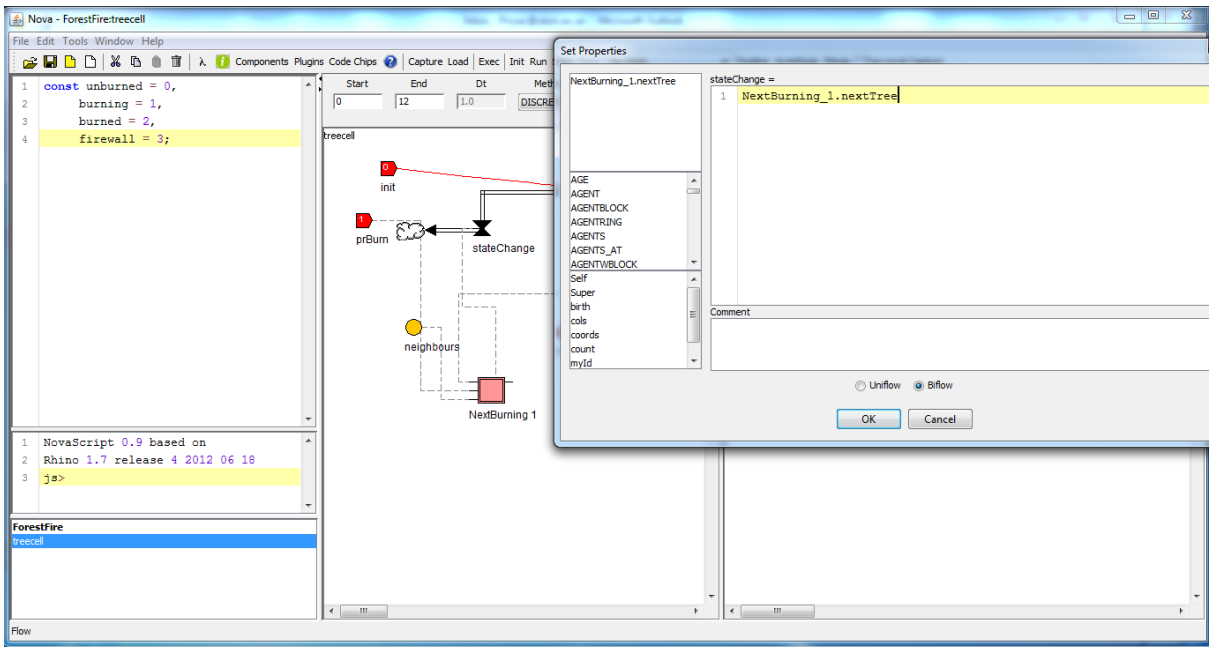
Linking output `nextTree`:



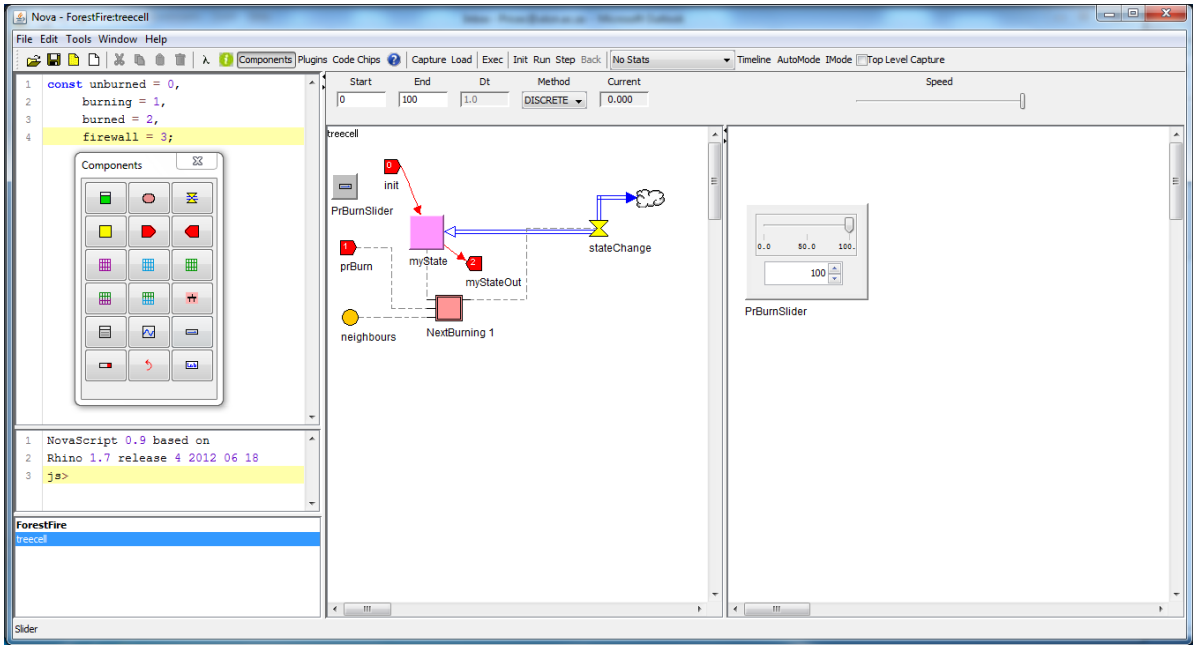
You should then get a model with the codechip wired up:



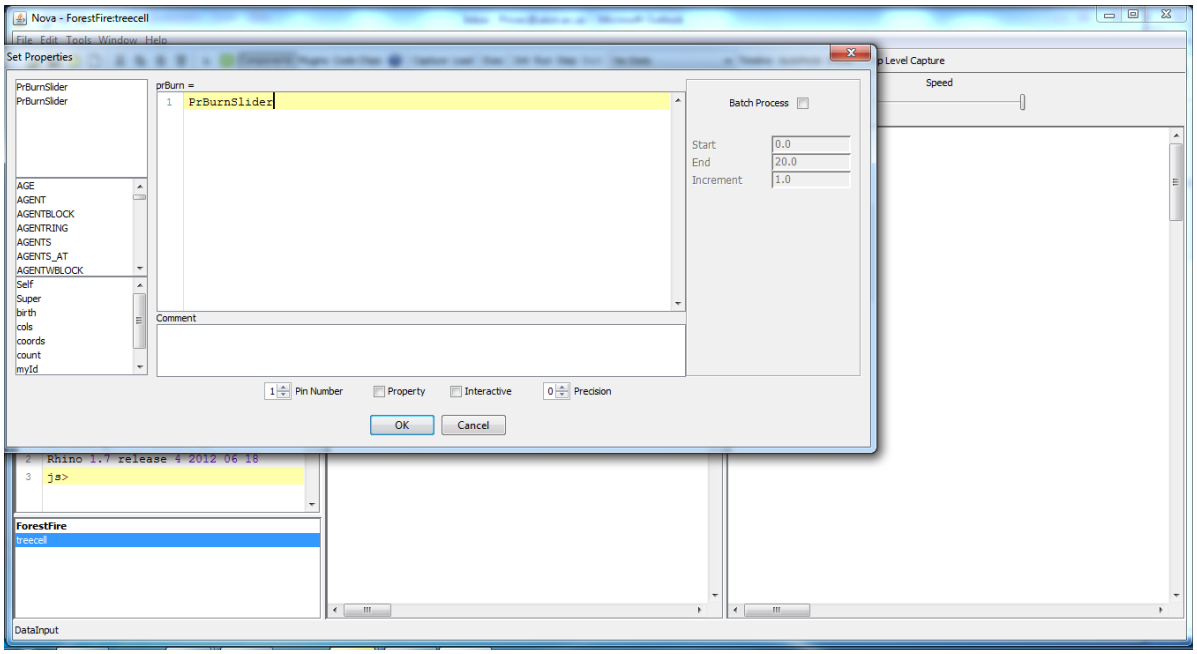
In the flow, change the flow's initial value to `NextBurning_1.nextTree` by double clicking on `NextBurning_1.nextTree` in the left hand pane:



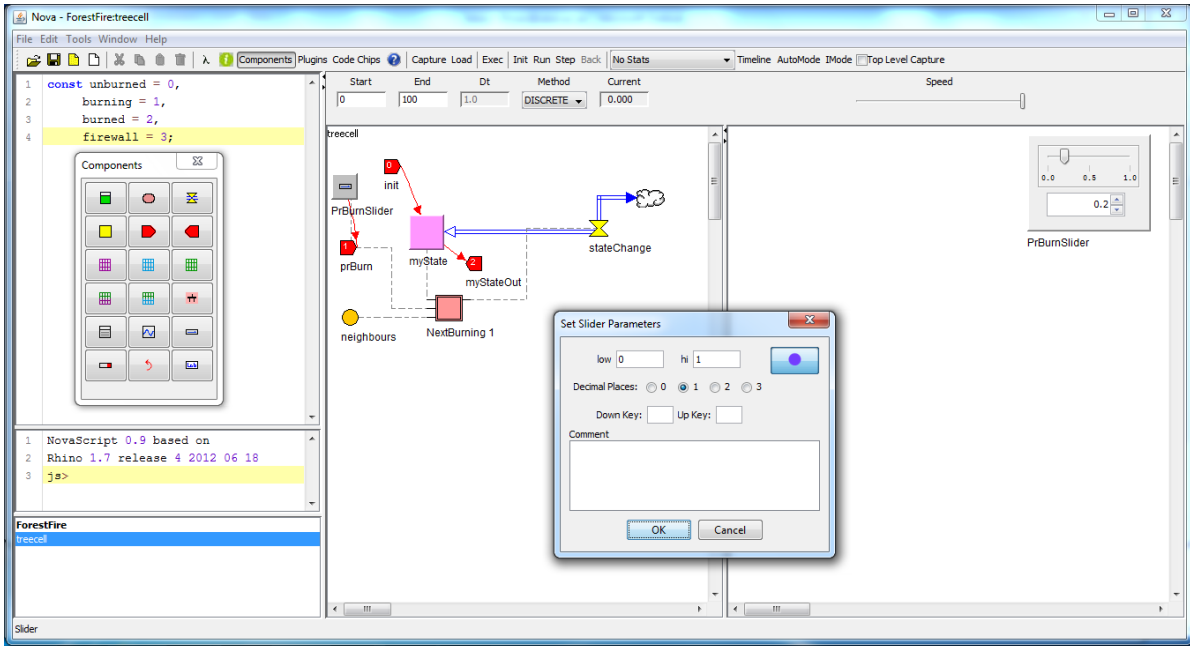
To be able to change the probability of burning, add a slider to the treecell model.



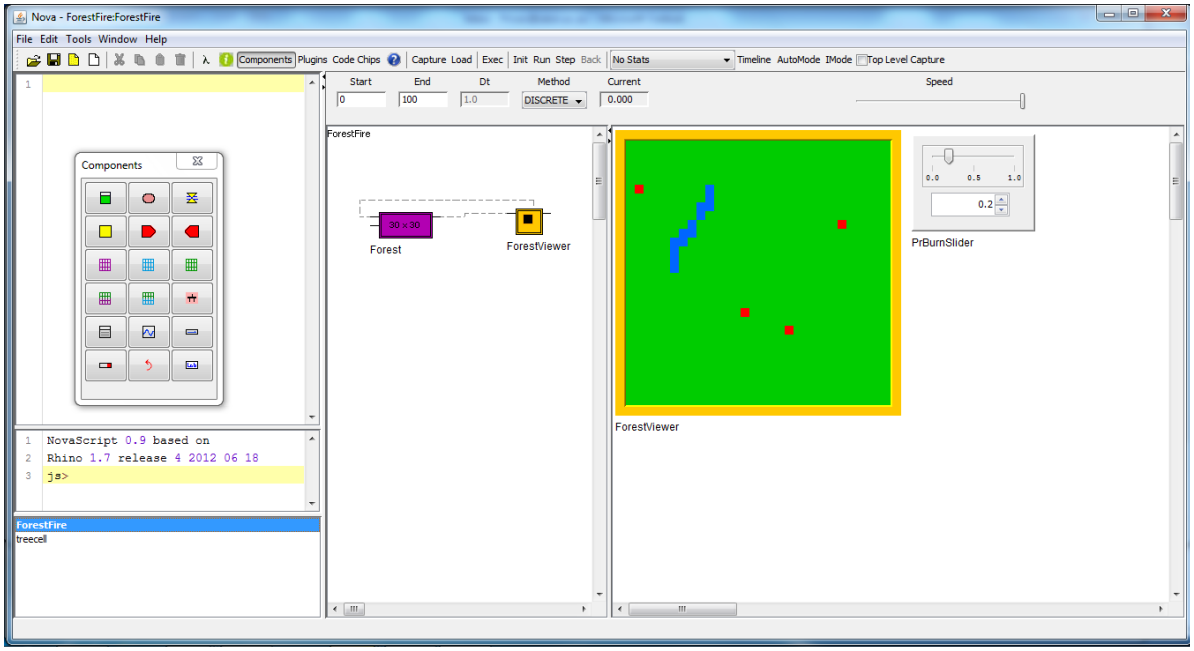
Link the slider to the input pin prBurn.



Then click the pin on the top right hand side of the Set Slider Parameters window to make the slider available in the top level model.



This makes the slider available in the top level model:



Save the model.

Click on Capture Load Init and Run to watch the model run.