Lab 3

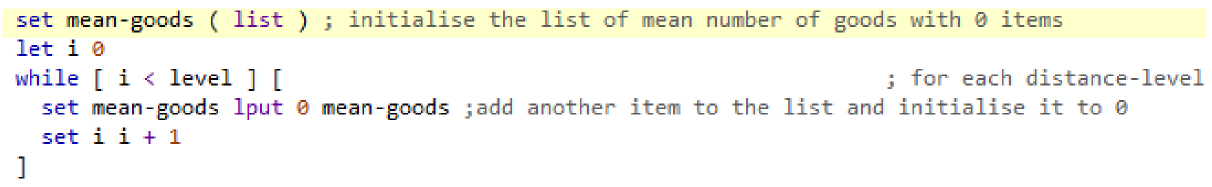
Brian Burke - 18322240

Introduction

This laboratory explored the setting up of a trade model in Netlogo using breeds and user-defined variables. A production centre is created in the middle of the map and vendors are placed around the producer. Multiple sliders were created to customise a specific trade model. Some of the sliders created include the production level, storage and storage threshold of the producers and vendors. Trading occurs between partners if one has surplus goods and the other has space for goods left. While loops and lists in Netlogo were also explored in this lab. The uptake of goods by vendors at each distance level was monitored with a plot. Various values for parameters related to the trading model were implemented to examine their effect on the model.

Body

When editing the slider to be greater than 6 and running the program an error was created as the list length created in the setup was initialised manually to have 6 items. As such, when the function to update the number of goods at a vendor greater than item 6 in the list was ran, an error was thrown as this item in the list did not exist. I adapted the code to handle any level size as shown below.

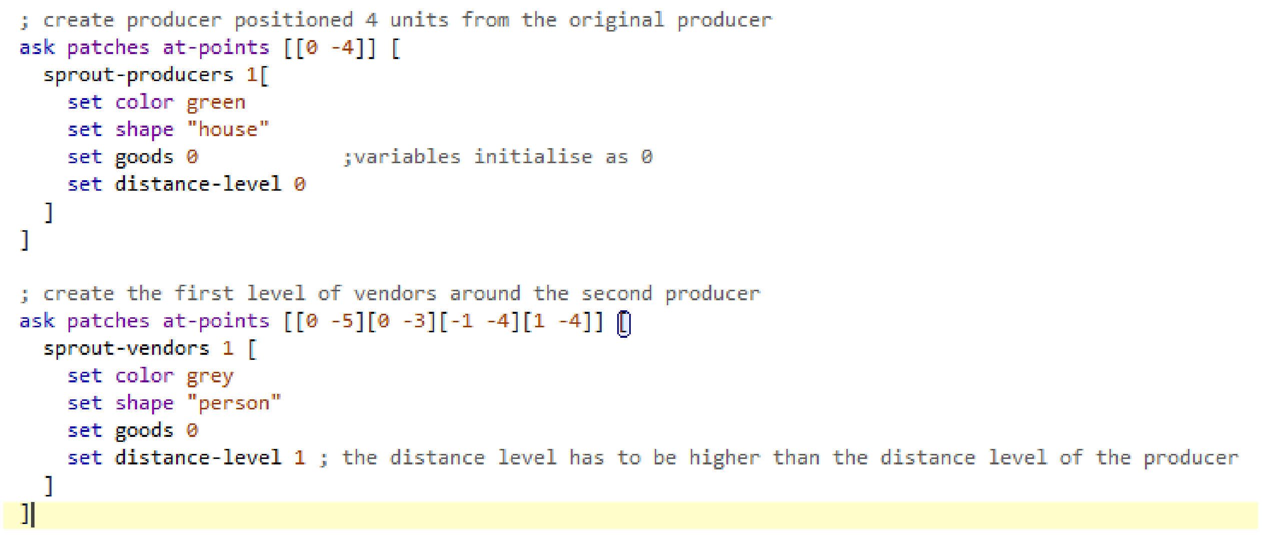


```
set mean-goods ( list ) ; initialise the list of mean number of goods with 0 items
let i 0
while [ i < level ] [                                           ; for each distance-level
  set mean-goods lput 0 mean-goods ;add another item to the list and initialise it to 0
  set i i + 1
]
```

Figure 1 – Code showing the creation of a list to handle any level size

The code above operates by initialising a list with no items and then running a while loop to continually add items to the list and initialise their values to 0 until the number of items as specified by the level variable is reached.

I added a second production centre 4 units below the existing production centre. I also initialised 4 vendors around this second production centre, as shown below.



```
; create producer positioned 4 units from the original producer
ask patches at-points [[0 -4]] [
  sprout-producers 1[
    set color green
    set shape "house"
    set goods 0             ;variables initialise as 0
    set distance-level 0
  ]
]

; create the first level of vendors around the second producer
ask patches at-points [[0 -5][0 -3][-1 -4][1 -4]] [
  sprout-vendors 1 [
    set color grey
    set shape "person"
    set goods 0
    set distance-level 1 ; the distance level has to be higher than the distance level of the producer
  ]
]
```

Figure 2 – Code showing the creation of a second production centre with 4 vendors surrounding it

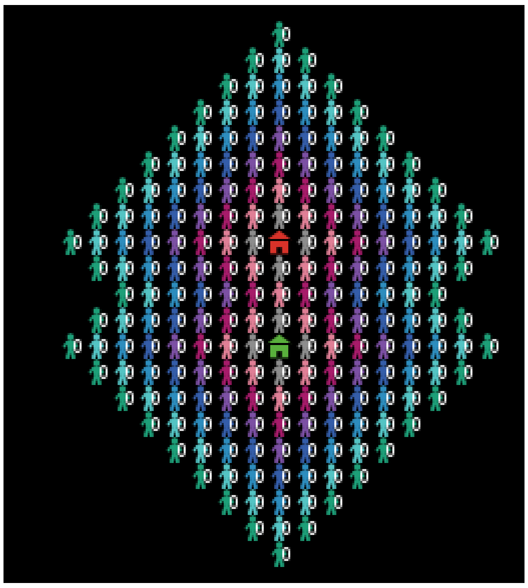
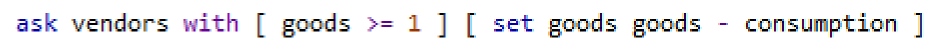


Figure 3 – Figure showing 2 production centres with level 8

The changing of consumption into a user defined value was completed by creating a slider named consumption and updating the following line of code in the to go function. Here consumption has replaced the number 1.

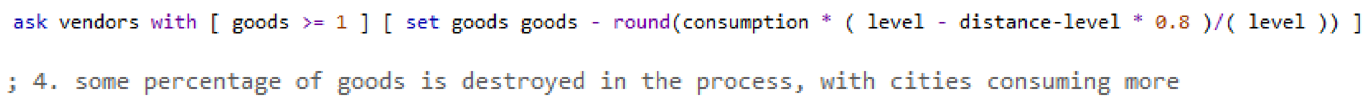


```
ask vendors with [ goods >= 1 ] [ set goods goods - consumption ]
```

Figure 4 – Figure showing updated code which allows user defined consumption

By increasing the consumption value, the number of goods at each level was significantly reduced. By increasing it further, many vendors were storing a negative amount of goods which made no sense. By decreasing its value to 0, every single vendor quickly reached maximum occupancy.

To make the consumption market specific I assumed that the closer you are to a production centre in this model, the closer you are to a market representing a city and thus the higher your consumption. I implemented a higher city consumption using the following line of code.



```
ask vendors with [ goods >= 1 ] [ set goods goods - round(consumption * ( level - distance-level * 0.8 )/( level )) ]

; 4. some percentage of goods is destroyed in the process, with cities consuming more
```

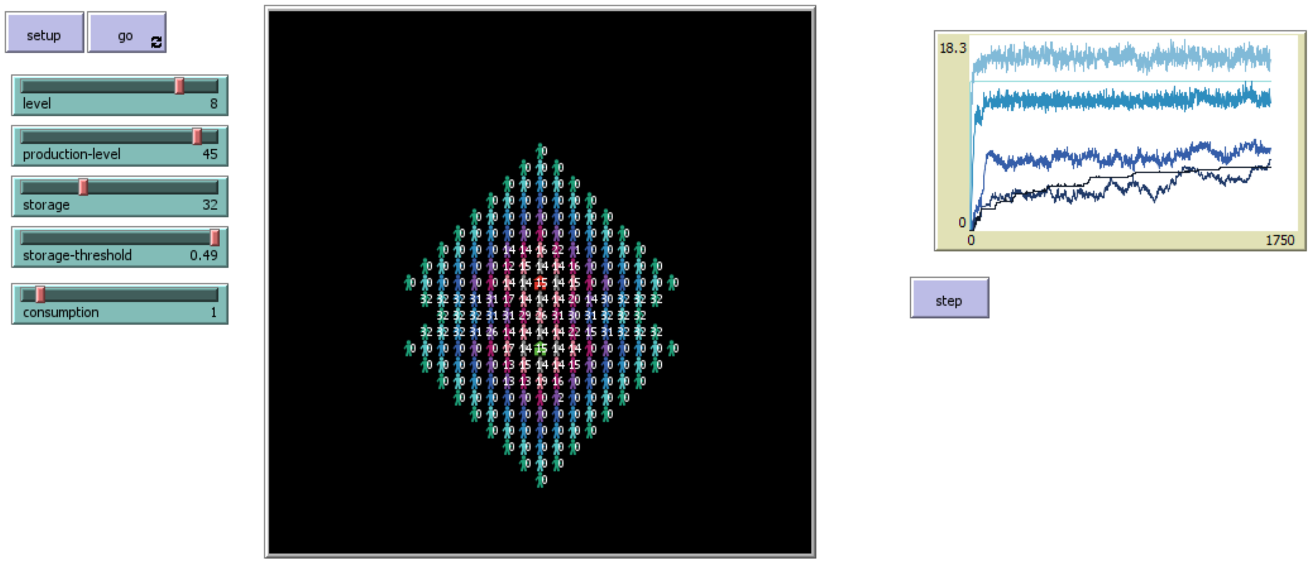
Figure 5 – Figure showing code for higher city consumption

The above code works by obtaining a fraction which is greater the closer you are to a production centre. A scaling of 0.8 is added to ensure vendors at the furthest distance level do not consume 0 food. To ensure the value generated is not a fraction which would complicate the trade model unnecessarily I ensured that the value generated was a whole number using the round function.

Another possible method of measuring the economic activity would be to measure the total number of goods traded during a time period, or the number of ticks in Netlogo. This could be implemented by creating a global variable which would increment each time the line $set\ goods\ goods\ +\ 1$ was run. After a certain number of ticks the value of the variable would be noted and the value itself reset for the next time period.

Results

The output of the trade model implemented in this lab is displayed below.



level 8

production-level 45

storage 32

storage-threshold 0.49

consumption 1

Figure 6 – Figure showing the output of the trade model

One thing of note about the trade model is that the vendors roughly halfway between the two production centres had a much greater number of goods stored than those who were who were situated at the same distance level of a production centre but not in between the two centres. The difference was almost double.

Conclusion

The Netlogo software has many useful data tracking techniques which I had not fully realised the scope of until after this lab. The 4 end-of-chapter exercises were successfully completed with no problems. Software tools such as lists and while loops which can be tricky in other software languages are very simple in Netlogo, especially lists. I am very much interested in the idea of trade models and their applications and as such I would like to explore the idea of adding increasing complexity and intricacy to the trade model to generate one which is more reflective of trade patterns in real life.