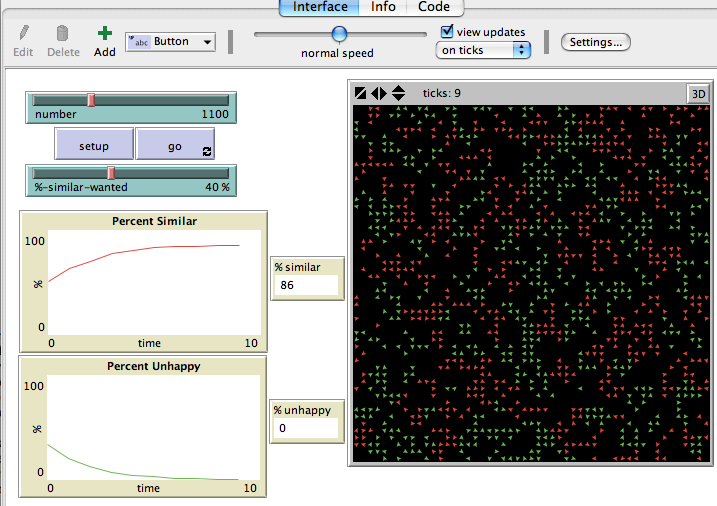
Netlogo’s Implementation of the the Thomas Schelling Segregation Model

The example Netlogo script for the Thomas Schelling Segregation Model describes how individual preferences for neighbors change based on a percentage of desired segregation. This simple model is based on Thomas Schelling’s social system model. In the example below I held the number of entities constant to ~1100. The intent is to show how the %SIMILAR-WANTED variable describing entity preference for other similar entities, affects the choice of turtle locations on the grid. Note in our example we have two entity types that are both equally affected by the %SIMILAR-WANTED variable.

|  |  |
| --- | --- |
| Setting | %SIMILAR-WANTED value |
|  | 10% : Entities quickly conform to a stabilized acceptable amount of segregation. 10% allows for a significant amount of dissimilar entities to be stationed near each other on the grid. Still the value of % similar remains high at around 60% (see suggested reasoning below) but stabilizes in that range. As happy as defined by  *happy = similar-nearby >= ( %-similar-wanted \* total-nearby / 100 )*  The variable *%-similar-wanted* keeps the value to the right of the equation sign low creating more happy entities. Time to 0% unhappy is fast compared to other markers. |
|  | 46%: Segregation is more prevalent, clusters are more defined and % similar values are higher as more ‘like’ entities exist next to one another due to the increase of % similar wanted. The system takes several seconds longer to stabilize at *0% unhappy* due to the algorithm having to reposition entities often that were not happy with their location.  *to move-unhappy-turtles*  *ask turtles with [ not happy? ]*  *[ find-new-spot ]*  *end* |
|  | 77%: Time to generate 0% unhappy population goes up considerably (364) as does % of similar entities. The graphs that illustrate *% unhappy* are not as smooth which from my calculations (python code) appear to be due to random placement of entities (*move-unhappy-turtles)* that cause not only that entity’s ‘unhappiness’ but that of his neighbors. This was not as prevalent in earlier %-similar-wanted values because the variable allowed for more acceptance of neighbors. |
|  | 85%: At this setting the *% unhappy* never stabilizes to 0%. The desired number amount of %-similar-wanted reaches a threshold at around 80% where entities can not find a cluster of acceptable neighbors. The image to the left shows that even at even with a desired value of %-similar-wanted, the model stabilizes at around 54% similar. Again this is likely due to random placement of entities at locations that produce unhappy neighbors. The cycle forces entities to relocate in a manner that infrequently generates an entity with at least 85% neighbor similarities. |

### If each turtle wants at least 40% same-color neighbors, what percentage (on average) do they end up with? :



Answer: Roughly 80%

The key to understanding this is the NetLogo Script below:

*total-neighbors = sum [total-nearby] of turtles*

*percent-similar = (similar-neighbors / total-neighbors) \* 100*

I may want 40% of my neighbors to be similar, but so does my neighbor, which in effect creates a clustering of similar (in this case) turtles. The more clustering that takes place, the more likely the value of *% similar* will increase. In the simple example below when x0 lands on grid point (1,1), the x values on grid points (0,0), (1,0), and (2,0) now have **x0** (1,1) as a closest neighbor.

|  |  |  |
| --- | --- | --- |
| Grid Points | 0 | 1 |
| 0 | x |  |
| 1 | x | x0 |
| 2 | x |  |

**Additional Work**

I was interested in getting a bit more into the heart of the Shelling model, so I rewrote it in Python. It is a reasonably unoptomized version of the netlogo code but it does in effect allow me to look at variables as they grow and shrink within a debugging environment. I also added a little bit of a slant on the model. Instead of a random placement of dissimilar turtles when they were unhappy with their location, I created a ‘roaming’ model where turtles would need to choose a direction and move until they were happy with their location. There was an interesting consequence of this approach. As the turtle moved to a location of preference, he would sometimes dislocate satisfied turtles, especially if the *% similar value* was high. One might see this as a simple example of how refugees flow into surrounding territories and displace the existing population.