#### **Proposed Project Extension upon the Schelling Model (and other CA models)**

by John Hellrung and Shadia Prater

**MOTIVATION:** Our final project video will explore the Schelling model in depth and possibly compare it with other CA models. We will also show what the Schelling model has taught us about recurrences in nature and its limitations.

**PURPOSE:** The Schelling model can able applied to many occurrences and life and should be shown as much. Our software will take in a set of values for the board, red to blue color ratio, and show how many iterations it will be before everyone is happy with their placement. It will then show the average happiness of each dot.

**RESOURCES:** This program is based on the Lab 1 assignment of the schelling model and various sources given to us throughout the class, below is just a few.

● <http://cs.berea.edu/courses/complexity/tasks/schelling.html>

● <http://cs.berea.edu/courses/complexity/tasks/schelling-implement.html>

We will use:

● <http://camstudio.org/> for screen capture

●  [sony vegas pro 10](http://www.sonycreativesoftware.com/download/vegaspro10) for traditional video editing

**INITIAL DESIGN PLAN:**

● The original implementation of L1: Schelling included the following functions:

|  |  |
| --- | --- |
| Class name: Schelling Model |  |
| Class Methods: | Class Collaborations (other classes): |
| ● initializer method (aka constructor) called \_\_init\_\_(). Creates an new board (matrix) with dimensions set by user  # ------- Creating Agents  ● create\_pos: creates a matrix of location and places random agents in the matrix using the inputs given by the user. This matrix contin [ = H [ = W [color,similar] = O ] ]  # ------ Movement  ● similar\_cal (int): Calculates the similarity for an agent.  ● check\_move (true or false): check if an agent can move based on color and threshold  ● turn: maintain the order in which the agent should be moved and calls all move related methods.  ● move\_distance\_cal  ● search\_new\_pos: searches all the possible new positions and works with check\_move. And outputs a list containing the positions of the possible moves. ### it was hard to implement our sonar search idea, so we opted for a linear search.  ● new\_pos: uses distance\_cal to output the new position to swap with.s ### if distance is same random was not possible, it would return the last possible distance  # ------ Testing  ● avg\_similarity: this is the segregation level after simulation, uses similarity\_cal for each agent.  # ----- Virtual  ● update\_pos: swaps the agents(color) in the matrix. | ● Board |
| Class Data: | Class Collaborations (other classes): |
| ● size of the board: width and height  ● colors: blue / red ; None = white  ● discrimination: .60  ● occupation: .70  ● contentedness: .50  ● matrix of agents: [ = H [ = W [color,similar] = O ] ]    ############################  def \_\_init\_\_(self, width=0, height=0, occupation=0,discrimination=0,contentedness=0 ): *# Each Point object has its own x and y coordinates and possibly a turtle*  *'''initializer method aka Constructor:*  *Creates a new point at x, y. If no x, y are given, the point is created at 0, 0 '''*  self.width = width  self.height = height  self.occupation = occupation  self.discrimination = discrimination  self.contentedness = contentedness  self.white = 0  self.blue = 0  self.red = 0 | ● |

|  |  |
| --- | --- |
| Class name: | Board |
| Class Methods: | Class Collaborations (other classes): |
| ● Dots: Collaborates with Turtle Library to plot each of the agents on the board.  ● Create Board: Collaborates with turtle library to create a visual representation of board  ●  ● | ● Turtle |
| Class Data: | Class Collaborations (other classes): |
| ● x: represents the height of the board  ● y:  ● size  ● x\_length  ● y\_length  ● self.agent  ● startingposition\_x  ● startingposition\_y  ● Agents list |  |

IMPLEMENTATIONS: A list in bullet form of specifically what was accomplished in code including any challenges overcome and innovations that were not specifically required by the assignment.

* At first we had a few bugs that were left over. One of those was that iteration where it would error out saying “Divided against zero”, this I fixed this with simple try and expect statements in the similar\_cal method in the Shellings\_Model class.
* Next we try to fix the counter because it would try to count moves that didn’t happen. What we did is first add the movement counter out of the Turn method and turn the variable global that way we could call that data any time during the process. We found that I was work but one cannot see every agent move based on the simple rules. After testing and point out some of the problems. we solve some of the errors by putting in more checks. In Order to check if that was being done, we made a list of moved agents. This list included the first two location selected and the colors for each location, example ['(1,2),(2,2),(red),(white)']. This showed us the results of the model and allow us to track it moved movement more precisely. Now, the model is model is more accurate and gives a more accurate movement counter.
* We also fix the broken test suit by adding the board class and rearing method. Now the test suit fully and print are new movement list.
* We tried fixing the scaling issue with the board class with no avail. The problem arises when you input a board size the in same the same length and height. What happens is the virtual agents will split outside the grid. We decided to just give the user one option of a number that will detriment the length and height.

FILES: A list in bullet form of the names of all files submitted (source code and input, etc.)

* main.py
* Testing.py
* shelling\_model.py
* borad\_class.py
* Final Project Design Document

VIDEO: A publicly available link to a YouTube video on your project topic.

URL: <https://youtu.be/F-1hpaj0GWs>

REFERENCES:

* Ishwar Agarwal, William Mosier, Haleigh George, Ben Quesada, Jan Pierce
* Think Complexity Chapters 6 and 10
* <http://ncase.me/polygons/>
* <http://boingboing.net/2014/12/08/parable-of-the-polygons-segre.html>
* <http://www.sebastian-grauwin.com/wp-content/uploads/2011/04/rapportM2.pdf>
* <https://sites.google.com/site/agilleyschellingmodel/aonrc-uor>
* <https://en.wikipedia.org/wiki/Tipping_point_%28sociology%29>
* <https://sites.google.com/site/agilleyschellingmodel/>

COMMENTS:

We liked the concept of this idea and I think it should implemented more often but next time give us more time, so we can make an even better video. The approach towards going in depth on something we have learned throughout the semester. I think this was a wonderful final project. Plus, go through concept gave us some interesting insight on how global environments work. Please do this in the future and this is a cool idea and we wish that other classes did the same thing.