TJHSST Senior Research Project
Sugarscape: An Application of Agent Based Modeling
2007-2008

Andy Menke

June 11, 2008

Abstract

Sugarscape was a complex program created to simulate human culture and society through the interactions of agents that travel around the world (known as a scape) collecting sugar, which they need to survive. Sugarscape allows social scientists to actually set up experiments and test hypotheses, which is normally extremely difficult. Sugarscape is one implementation of agent based modeling, which is the idea that many complex phenomena can be explained by the interactions between rather simpler agents. I am in the process of recreating Sugarscape entirely in the programming language Java and extending it, while allowing for relatively simple modification by others. I have implemented simple agent functionality and have begun working on making the agents live and die, which will help them act much more like actual living things.

1 Background

1.1 Agent Based Modeling

Agent based modeling grew out the idea of *cellular automata*, and may be considered a more complex form of these. A cellular automaton is a grid of cells, each of which can be in one of several states. Which state each cell is in

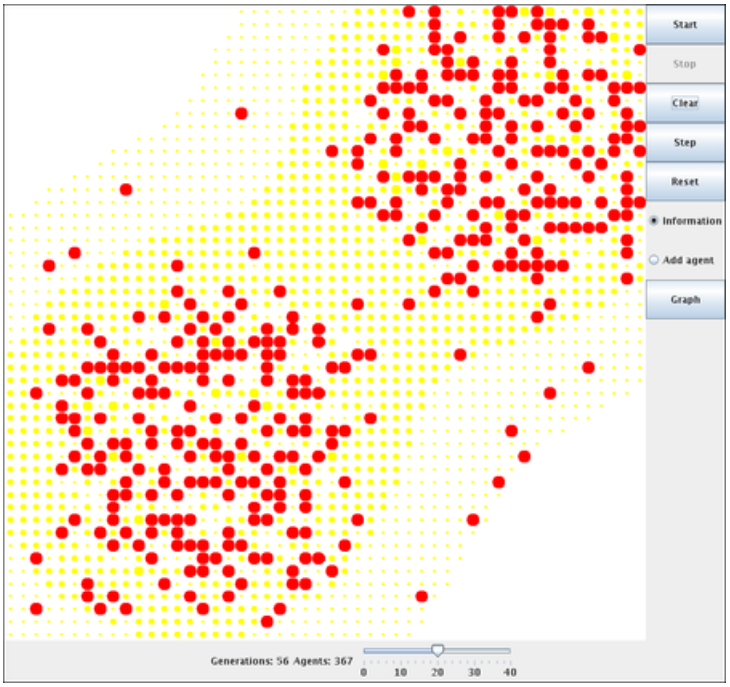
is determined by local rules, which usually rely on what states neighboring cells are in to determine a given cell's new state. One well known cellular automaton is Conway's Game of Life, where cells have only two states. In agent based models, there are relatively simple rules that govern the behavior of agents; from the interactions of these agents, which may or may not be based on a grid, complex behaviors emerge. One of the earlier agent based models was *Boids*, which simulated the behavior of flocking birds. In general, each agent tries to avoid crowding the other agents, while staying near them and flying in the average direction of nearby agents. From these simple rules, flocks of agents that travel across the "map" arise.

1.2 Potential Problems

One problem that agent based modelling in the social sciences suffers from is its opacity. It is extremely difficult to tell from simply observing an agent based model what its behavior is or will be. Also, it is sometimes difficult to tell whether or not results are really meaningful. The results may be simply caused by the specific input parameters and not be meaningful. Another large problem with agent based societies is that it is very difficult to match results to the real world. A pattern observed in the model may be completely hidden by other factors in the real world. However, one can still learn about rules that apply to societies generally, and not just human societies (Srbljinovic).

1.3 Sugarscape

Sugarscape is one of the most complex agent based models that has ever been created. It is meant to model simple human societies and make sociology more of a "hard" science, like chemistry or physics. The basic underlying structure of Sugarscape is the scape, which consists in its simplest form of a grid of cells, each of which contains a certain amount of sugar. Agents travel around the map, collecting the sugar that they need to survive and interacting with each other. More complex versions of Sugarscape also include spice, which the agents can trade amont themselves. Agents can have children together, fight with each other, and transmit their cultural attributes.



Sugarscape can be extended to study many different aspects of sociology. Flentge, Polani, and Uthmann extended Sugarscae to allow agents to claim plots of land as their own and transmit memes that determine how agents act about the land claims of others. I hope to create a version of Sugarscape that will be easily extensible.

2 Testing

As I recreate piece by piece the work that was done in *Growing Artificial Societies*, my program should have similar results to those in the book. The results will not be exactly the same, because Sugarscape is not entirely deterministic (the order in which the agents move is random), but results will be similar no matter what random movement order is selected.

3 Procedures

3.1 Software

My program is completely written in Java. Agent based modeling is best implemented in an object oriented programming language, and my program will take full advantage of Java for easy extensibility.

3.2 Rules

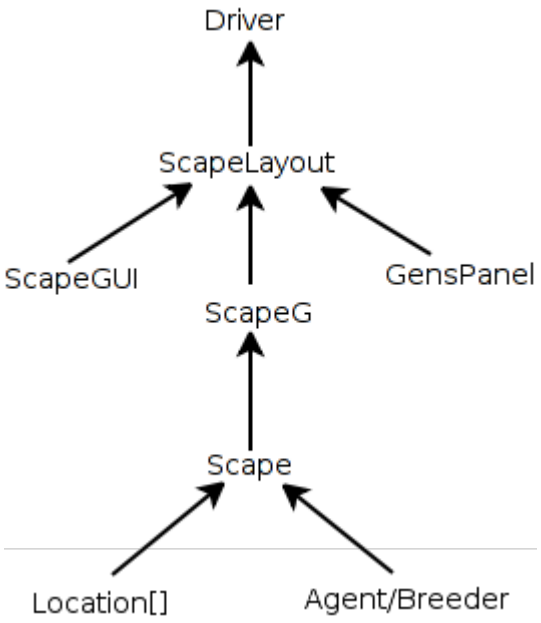
These are some of the basic rules that agents and the environment follow.

1. Sugarscape Growback Rule G(A): At each location on the scape, sugar grows back at the rate of A units every unit of time until the amount of sugar is the maximum allowed at that location.

2. Agent Movement Rule M: Each agent looks as far as it can in the four permissible directions: north, south, east and west. The agent then moves to the closest unoccupied location with a maximal amount of sugar and collects the sugar at that location.

3. Breeding Rule S: Agents have ages and genders. When two agents of opposite gender that are old enough are next to each other a new agent (their "child") is created. The child inherits both metabolism and vision rom one of the parents, and gets half of each parent's sugar supply to start out. The child is placed in a randomly selected empty location that is adjacent to one of the parents.

4 Structure



q3.png

My program is made up of many classes that contain instances of one another. The two classes that do the most work are probably ScapeG and Scape, which handle graphics and scape updating respectively.

1. Driver: This class is what the user runs. It creates an instance of the main GUI class, ScapeLayout.

2. ScapeLayout: This class does very little except hold instances of the three main panels: ScapeGUI, GensPanel, and ScapeG.

3. ScapeGUI: This class has most of the buttons that control the operations of the scape, such as start and stop.

4. GensPanel: This class has some more information and controls, such as a label displaying the number of generations that have passed and a slider bar for controlling the speed.

5. ScapeG: This class handles all of the graphics work that my project does. It also has the main timer that tells the Scape when to do things (like update).

6. Scape: This class is the main class the controls the operations of the scape. It mainly consists of a two-dimensional array of Locations and an ArrayList of Agents (or Breeders).

7. Agent: This class represents a single agent and contains methods for things like collecting and eating food and determining the best place to move to next.

8. Breeder: This class is and extension of the Agent class and includes methods for agent death and reproduction.

9. Location: This class represents a single square of the grid, and stores the amount of sugar and an agent (if there is one).

5 Results

My program does not have a certain result that should come out of it. I hope to create a program that will be useful to other students and researchers working on problems related to sociology. I have been working to recreate the results that were obtained by Epstein and Axtell in *Growing Artificial Societies*. After that, there are no real precedents for what I will be doing; I will have to see what happens.

I am considering releasing my project under the GNU General Public License, which allows others to use and make changes to the code. However, at the moment, I do not feel that my project is complete enough to be released for others to use.

References

[1] Epstein, Joshua M. and Axtell, Robert. Growing Artificial Societies. Brookings Institution: Washington, D.C., 1996.

[2] Flentge, Felix, Daniel Polani, and Thomas Uthmann. "Modelling the Emergence of Possession Norms using Memes." Journal of Artificial Societies and Social Simulation 4.4 (2001) http://www.soc.surrey.ac.uk/JASSS/4/4/3.html

[3] Srbljinovic, Armano and Ognjen Skunca. "An Introduction to Agent Based Modelling and Simulation of Social Processes." Interdisciplinary Description of Complex Systems. 1.1 (2003): 1-8.