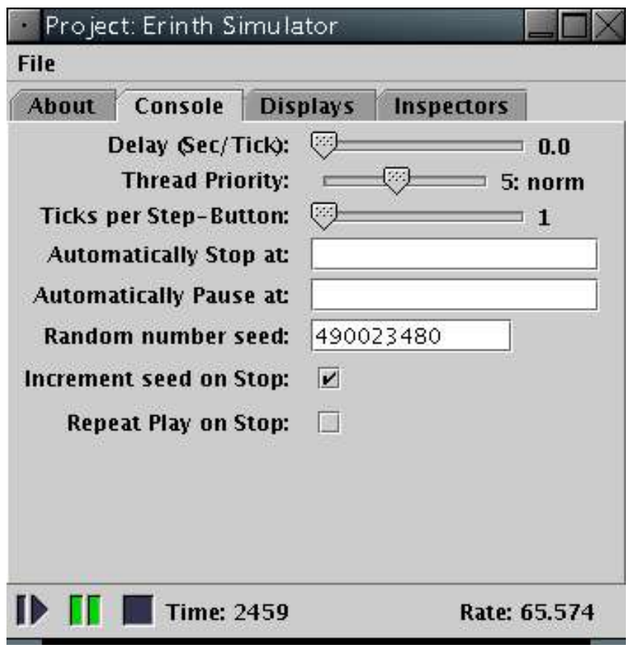
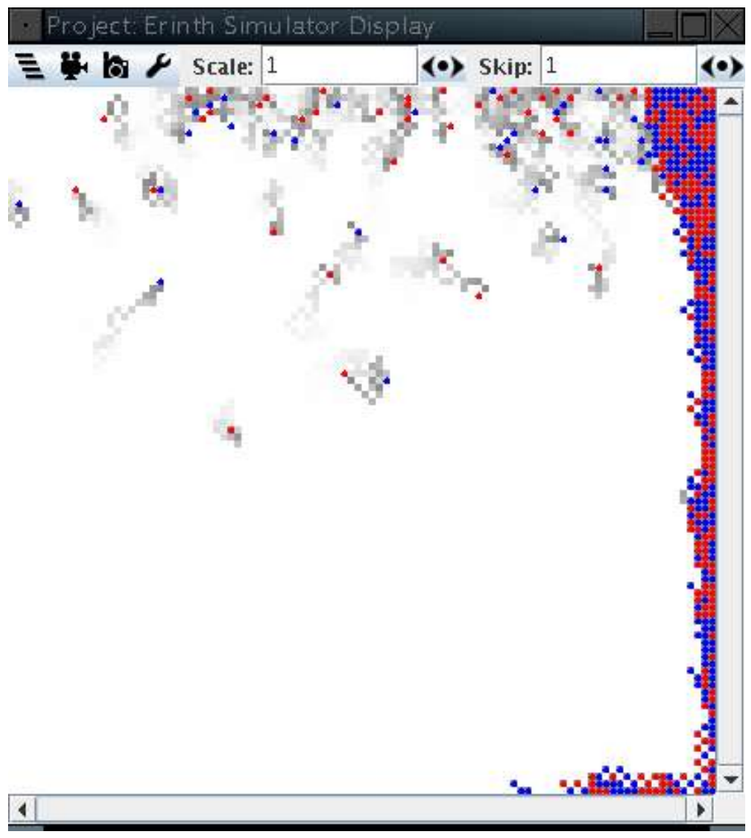
Modeling Evolving Social Behavior

by Stephen Hilber

Abstract

With the creation of Epstein and Axtell's Sugarscape environment, increasing emphasis has been placed on the creation of "root" agents - agents that can each independently act and interact to establish patterns identifiable in our everyday world. Models created for traffic patterns and flocking patterns confirm that these conditions are caused by each participating agent trying to achieve the best possible outcome for itself. The purpose of this project is to attempt to model evolutionary behavior in agents in an environment by introducing traits and characteristics that change with the different generations of agents. Using the modeling package MASON programmed in Java, I will be able to create an environment where agents will pass down their genetic traits through different generations. By adding certain behavioral traits and a common resource to the agents, I hope to create an environment where certain agents will prosper and reproduce while others will have traits that negatively affect their performance. In the end, a single basic agent will evolve into numerous subspecies of the original agent and demonstrate evolutionary behavior. This project will show that agents which possess the capability to change will change to better fit their environment.





Research & Background

Conway's Game of Life was the first prominent agent-based model. Each cell was an "agent" that contained either a 0 or a 1 (alive or dead) depending on how many neighbors it had, and acted independently of the environment. Conway's Game of Life didn't lead to any profound insights, but it did pave the way for future agent-based modeling. The advantage of agent-based modeling, as many found out, is that it did not assume prior conditions. It was a method of building worlds "form the bottom up", where independent agents were able to create complex worlds without any overseers. One popular psychological game, Prisoner's Dilemma, spawned a series of games where agents tried t maximize their outcome, often at the expense of other agents. Eventually, these agent-based models were incorporated in studies of flocking. The models created to show flocking behavior in birds did not incorporate flock leaders, as many presumed. Instead, the birds all acted for their own best interests, and directions and resting points were chosen as compromises of sorts.

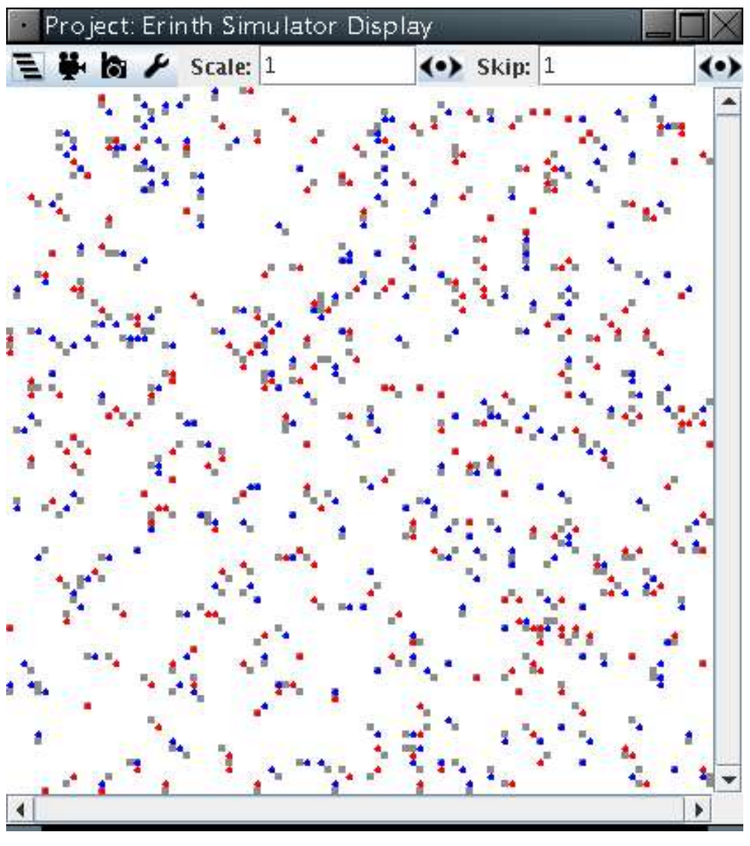
Using the theory that independent agents can create organized structures such as flocks, Epstein and Axtell created the Sugarscape world in an effort to discover if social behaviors and human characteristics could emerge through independent actions. The Sugarscape model had agents able to breed, fight, trade, and die, and the core of the model was the resource sugar. Each agent had a metabolism rate which burned off its sugar; if it ran out of sugar, the agent died. Instead of isolated behavior, however, the agents soon used their resources to work together. Agents shared sugar, sent "scouts" to gather sugar for the benefit of all, engaged in wars, and in general performed a startlingly large amount of human behavioral characteristics. When spice, a second resource with its own metabolism rate, was introduced into the world of Sugarscape, trade emerges as agents tried to meet their needs as best they could - and tried to get the best deal as a result. These behaviors are surprising, but ultimately show the value of agent-based modeling and the useful insight it can provide.

Procedure & Theory

In the first stages of the project, I created an environment based off of Mason's Particle tutorial. My first step was to create an environment where agents used random movement to interact with each other. To this end, I created a "move" method which provided the foundations for many aspects of my project. The move method took into account the boundaries of the environment, and using Java's standard Math.random class generated sequences of random numbers by which the agents could move in the environment. This proved to be only somewhat effective, so I looked through various sites to try and find better generators. I eventually found the Mersenne Twister, a reliable and popular random number generator. By incorporating the Mersenne Twister into my project, I was able to create a more reliable random movement scheme. At the end of the first step of the project, I had an environment capable of supporting thousands of agents moving randomly.

The second step of my project was to create introverted and extroverted agents. To do this, I at first created a scale from one to five. Agents were then given numbers on a gradient from one to five. Agents who scored high were considered to be extroverted, wanting to be around as many other agents as possible. Agents who scored lower on the scale were considered to be introverted, wanting to avoid contact with agents as much as possible. At this time, the input of the introversion or extraversion was the same for all agents involved. When I ran the model with inrtoverted agents, the agents would be spread throughout the environment like a fine mist. When the extroverted agents were inputted into the model, however, i was surprised that there seemed to be fewer agents than the 500 standard I had included into my program. It turns out that although my extraverted agents were grouping together correctly, the module supported multiple agents sharing the same space in the environment. Because of this, the extraverted agents were sharing the same space as other extraverted agents, causing the total amount of agents onscreen to be reduced.

In the later stages of development, I decided to change the scaling system of introverts and extraverts to a static class system. The reason for this was twofold: first, specific introverts and extraverts allowed the user to more easily understand the mechanics of the model; secondly, using specific classes for each agent that extended the basic Agent class allowed for different agents to have different coloring schemes, a necessity to the project that was hindered by the restrictions of the MASON package. Instead of completely recreating my move method, I decided to extend the original Agent class and create two sub-classes of Agent. AgentE represented extraverts, and AgentI represented the introverts. By this time, most of the work on the model had been completed. The interactions between the two types of agents on a social level were working flawlessly. Although I didn't implement any evolutionary behavior as I originally intended to do, I did manage to create a social simulation that can set the groundwork for a more detailed project.



Conclusion

When I originally started to build this model, the results I envisioned for my project were simple. Introverts would stay away from communities of agents in general, and extroverts would group together mainly by themselves. Occasionally an extrovert might chase after an introvert, but for the most part extroverts would be in groups and introverts would be drifters. The actual results proved differently. Introverts would group together with other introverts, and extroverts would group with other extroverts This occurred regardless of the environment, and this separation of introverts and extroverts was surprising. It seems that although the introverts are normally adverse to being too close to other agents, they prefer to interact with like kinds instead of being trapped in a sphere of different kinds of agents.

The fact that my findings are different from my initial expectations only show the advantages of model-based experimentation. Using a computer system with preexisting equations, I was able to find that although introverts tends to stay away from other agents more than extroverts, the net effect creates a world where the introverts and the extroverts tend to group together. Computer modeling is an incredible tool for experimentation. Over time, this model can and should be extended to show more detailed and meaningful results while comparing introverts and extroverts.

Credit goes to Conway for The Game of Life, Epstein and Axtell for Sugarscape, the MASON team for developing MASON, the Myers-Briggs Type Indicator, NetLogo, Swarm, and Dr. John A. Johnson's IPIP-NEO.