Agent-Based Modeling of Urban Society and Interactions

Creating a Realistic City Simulation in Order to Model Infections

Andrew Imm

TJHSST Computer Systems Research Lab

2009-2010

Abstract

Current systems used to model the spread of disease treat popu- lations as single entities, and neglect the actions of individuals. By developing an agent-based simulation focused upon the accurate mod- eling of social interactions seen in an urban environment, a testing bed that resembles a modern city arises. This testing environment — with its accurate modeling of day-to-day interactions within a city — provides a far better system to use when developing epidemiology simulations. Using an implementation of goal-oriented agents who are guided by a number of variables that make up their unique and indi- vidualistic "personalities," this program attempts to create this type of urban model and use the system to run a number of epidemiological studies. Once the virtual society is established within its routines, and a network of social relationships has developed within the city's inhab- itants, the simulation will reach an autonomous running state where it can develop and grow on its own. At this point, various scenarios can be implemented to draw conclusions about human populations in urban environments. In one such scenario, the introduction of a sim- ulated influenza virus will help determine how an urban population reacts to such an infection, and how the disease is likely to spread through the city.

Background

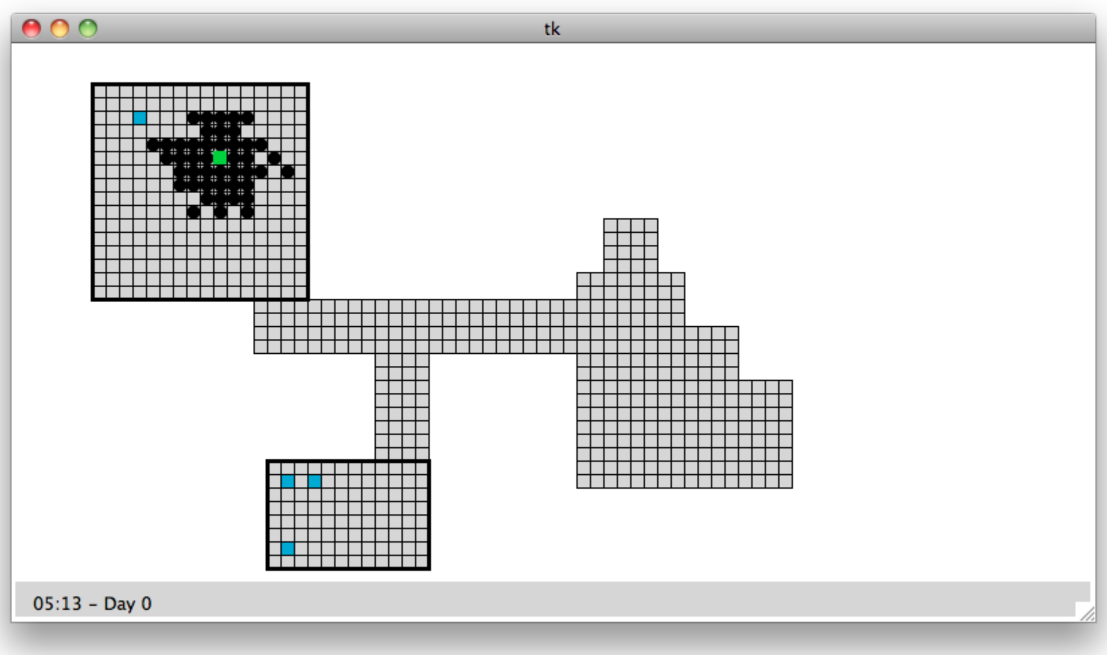
In the field of epidemiology, most models used to predict the outcomes of plagues and epidemics are math-based. They treat the entire population of a region or nation as a single entity. This take on the problem of studying the spread of disease has one major downfall — it assumes that all members of the population have similar behaviors. If any stratification is done to divide the population into subgroups, these are generally only related to susceptibility to the disease in the study. In other words, the unique characteristics of individuals are lost. An agent-based model, while more processor-intensive than a strict mathematical model, brings into play this individuality. How- ever, past models that took an agent-based approach were very simplistic. For instance, viral modeling has been popular in the TJHSST Computer Systems Research Lab for years, but nearly every project has involved agents moving randomly within a closed, featureless environment. Effectively, these simulations resembled nothing more than an experiment of specialized bacteria moving around in a petri dish — hardly an experiment that can be used to make generalizations or conclusions about a human population. For such conclusions, the agents in the model must act as humans do; this necessity provides the reason for developing an accurate simulation of an urban society.

To fill this gap in epidemiology research, one of the main goals of this project was to develop a testing environment where urban scenarios can be implemented, and virtual populations can be experimented upon. The simulation environment provides a series of tools which can be used by scientists and researchers to customize and create their own simulations; the simulation system can be modified and manipulated as the user sees fit, from the drawing of maps to the creation of agent schedules and custom interaction methods. An epidemiologist studying viral spread can write interaction methods that cause a virus to spread from one agent to another, while a sociologist studying the proliferation of rumors can explore them through methods that transmit knowledge between agents. Items as intangible as thoughts, and as tangible as money can be transferred from one agent to another with the ability to customize the simulation at will.

Beyond the ability for customization, the in-depth qualities of realism in this simulation tool will provide an effective testing environment for epidemiology studies. Because the simulation is designed with a focus on individual interactions, the program works well for simulating the spread of disease from one individual to another. Using agent-based models to analyze the spread of disease is something that has been explored before by a few scientists, but the fact is that it is not a mainstream method of epidemiology modeling. Dr. Stephen Eubank from Virginia Polytechnic Institute is one of the leaders in the field of agent-based epidemiology modeling, and his projects explore the spread of many diseases in a variety of environments. For instance, his "Modelling Disease Outbreaks in Realistic Urban Social Networks" takes on a similar problem as my project does — exploring the spread of disease through a social network. However, his program does not take into account all of the environmental aspects of simulating a city that my project does. With the extra features found in my simulation model, I hope to establish a platform that can accurately assess the quality of various quarantine methods when dealing with infectious diseases.

Simulation Program

The simulation makes up the majority of the code written for this project. At run time, it is provided with the location of a simulation file, which tells the program where to look for each component of the simulation. The data needed to run the model is divided into various files: a file that defines the world, a file that defines agents and their schedules, and a file which contains the code for custom interaction methods. Once all of these program files have been located, the simulation loads the map file in order to properly construct the city environment. With the empty world now loaded into memory, the program then loads an agent file which tells the computer how to configure the virtual city's populous. Each agent is assigned a name, a schedule, and a "personality" — a set of preferences that dictate how likely the agent is to perform various actions. Once the world and its inhabitants have been built, the program initializes its internal clock to 12:00 midnight on day 0. As the model runs, the virtual clock updates, and eventually agents wake up. As time progresses in the simulation, the agents go about the daily routines dictated in their schedules, navigating the city using the simulation's path- finding algorithm. Using built-in methods, they can be ordered to travel to different buildings or areas of the map, and are able to find their own space to inhabit in each building they visit. Inherently, the agents encounter others throughout the day, and begin to remember other agents whom they often see. These memories of acquaintances are the beginning of the agent's social network: a stored list of friends and colleagues that allows the agent to keep track of people it has already met. The agent's list of acquaintances also keeps track of how well the agent knows others; this data is used by the agent to decide whom to interact with. As the simulation ages, the virtual city begins to resemble its real counterpart. Agents become established in their routines, and have dependable networks of friends that keep them socially active. At this point, a range of tests can begin in the simulation. Manipulation or addition of variables — such as a virus — at this stage ensures that the results resemble a real-world reaction as best as possible.



Agents swarm a green square as the simulation runs

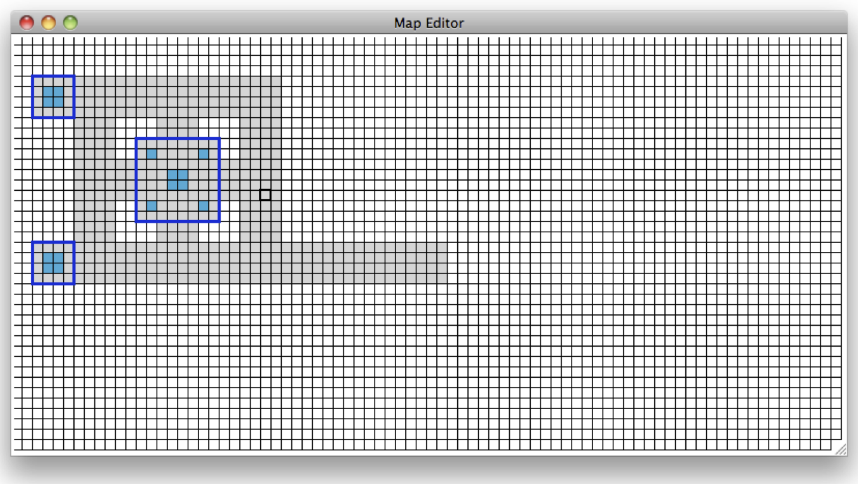
Agents

Agents navigate the map according to their schedule, stepping through it to check whether they should be moving to a new location with each time increment. Their navigation method is a standard A* search based on the grid of the map, where horizontal and vertical movements of one square constitute a cost of 10, and diagonal movements of one square constitute a cost of 14 (10 times the square root of 2). Since agents are constantly moving, they are not treated as obstacles by the navigation method. However, multiple agents cannot occupy the same square on the map, so agents who are directly neighboring the currently-navigating agents are temporarily seen as obstacles. This behavior allows many agents to attempt to reach a single point in an effective realistic way: as the crowd gathers around the goal point, agents fill in the gaps in the crowd to form an approximately circle-shaped mass of agents around the point.

When agents find themselves neighboring each other, they always take notice of each other. They also record how often they have encountered certain individuals, so as to remember acquaintances and how well they know each other. If they see another agent whom they know well, they are given the opportunity to interact with that agent. Such interaction can involve a variety of actions, depending on the structure and application of the simulation. Agents might wish to share specialized knowledge with each other, or they might unknowingly spread a virus through interaction.

Additional Programs

This project requires the creation of other programs that speed up the pro- cess of development. For instance, the simulation uses complex files to store maps, and the easiest way to create these maps is with a secondary program. The map builder allows the user to create maps with a graphical interface that displays the map as it will appear when the simulation is run. The program can also be used to create buildings on the map, and such build- ings are used by the simulation in order to determine where to send agents. The map builder also features a variety of other features that can be useful for development, including distance calculations and map-printing abilities. These programs are external from the final simulation program, but they are necessary for producing the components of any simulation, and therefore are a part of the suite of tools which have been produced to develop urban simulation models.



The map editor, displaying various terrain types and buildings.

Discussions

The simulation, now completed, is able to load a simulation file which dic- tates where all of the variable pieces of the simulation may be found. The specifications laid out in the map file give a variety of details as to world terrain and the placement of named "buildings," which are used to refer to contiguous areas of the map. The program creates the agents as they are specified within the agent file, each with its own name, schedule, and per- sonality attributes. When the simulation begins, the program keeps track of virtual time, and uses this clock to time and control the actions of agents. As it is, the agents within the simulation can continue to navigate the map indefinitely, moving to the various destinations indicated in their schedule. At they follow their daily schedules, they have the opportunity to interact with neighboring agents, potentially transmitting items through this inter- action. Tests have been performed which involved the transfer of knowledge through this medium, and it has been used in larger experiments to examine the spread of disease. The other large piece of code is the map builder, which creates maps with a variety of useful features that are used in the simulation. The map builder features a graphical user interface that makes creation of the map much easier than editing a text file by hand. Dialogs for creating buildings on the map, as well as defining new, custom types of terrain allow for a feature-filled map creation environment.

Conclusions

This project initially began as an examination into the spread of a virus through the implementation of a realistic urban simulation environment. As it progressed though, the focus changed to the creation of such a system that was dynamic and modular enough to be used for a variety of diverse modeling purposes. It was recognized that a system that can be easily changed and manipulated holds far more value in the field of agent-based modeling and simulation than a single-purpose program ever would. Through the year, the program was developed and expanded, making improvements in efficiency, realism, and the ability to customize simulations in a variety of ways. The final project was put to the test, creating two different models and exploring how they behaved; in the end, it was found that the simulation environment worked well for its newfound purpose, and was an effective way to implement models with widely differing properties.