

# Building a Simulation of the Spread of a Virus

# Using the AgentSheets simulation-authoring tool

Alexander Repenning, Andri Ioannidou, and Jonathan Phillips AgentSheets Inc. and the Center of LifeLong Learning & Design, University of Colorado at Boulder

This Sim Corner column is a hands-on description of how to use the AgentSheets<sup>®</sup> simulation-authoring tool to 1) create a simulation of a virus attack spreading through a community of people, 2) publish this simulation as a Java applet on the web, and 3) connect the simulation with other educational components. A simple but detailed example was chosen here to illustrate that simulations relevant to school curriculum (for example, math, and in particular, the Connected Mathematics Project curriculum [Connected Math Curriculum]) can be built quickly and do not require the builders to have a programming background. A more general discussion of AgentSheets, including more complex simulations, can be found at the end of this article in the "About AgentSheets" section.

# 1. Create a simulation of a virus attack

To illustrate how an AgentSheets simulation is created from scratch, a stepby-step description of how to build Virus Attack, a simulation of a virus spreading through a community, is presented. The point of this simulation is to understand the basic virus mechanism. Does the number of people infected by the virus go up linearly, or is there some other function behind the rate of growth? How fast does the virus spread? What can be done to contain the virus? The principles behind this kind of simulation are not limited to virus spreading. The same laws apply even to the spreading of rumors, fads, and other types of information relevant to society.

In Virus Attack, simulated people (called "agents" in AgentSheets) move around randomly, representing real people moving around by going to work, shopping, and traveling. A healthy person standing next to an infected person has a 5 percent chance of getting the virus. The goal here is to explore some of the questions about the spread of a virus by creating, running, and studying a simulation. While we believe that actually building a simulation is the best educational approach, we realize that some may wish to get started studying the spread of a virus even quicker. At the end of this article are URLs allowing you to run the finished simulation as an applet, to download the completed simulation so that you can both run it and continue with its development, and to watch a QuickTime video showing all the details of how the simulation was built.

Building a Simulation of the Spread of a Virus

**Creating a project:** In AgentSheets, a project is a set of agents and a set of simulations (called worksheets) that use those simulations. Create a new project (menu File  $\rightarrow$  New Project...) and name it. Choose the desirable size of agents (custom size 24 x 24, in our case).

**Creating agents' looks:** The system will provide a Gallery, which is the place where all the agents of a simulation get defined. For the Virus Attack simulation, you need a number of agents to represent healthy people, infected people, doctors, and background tiles for the people to move around on. Once you create your new agents (menu Gallery  $\rightarrow$  New Agent...) and name them, use the icon editor to create their depictions—that is, the agents' look—by double-clicking each agent in the Gallery. Figure 1 shows the People Gallery with all the agents created: a Person, a Sick Person, a Doctor, and the Background. Each agent can have any number of depictions representing different states the agent is in. For instance, to indicate if a person is sick or not, the person agent can have a normal and a sick depiction (Figure 1). The "Person-Sick" depiction was created by selecting the "Person" agent and creating a new depiction for it (menu Gallery  $\rightarrow$  New Depiction...).



Figure 1. The People Gallery with Person, Sick Person, Doctor, and Background agents.

Instead of using the basic icon editor provided by AgentSheets, you can also use prebuilt icons [Icon Factory], or draw your own icons with graphics tools such as Adobe Photoshop. AgentSheets' grab screen tools (in the Gallery menu) allow you to import icons from anywhere on the screen, including web browsers.

Creating a simulation world: Opening a worksheet (menu File  $\rightarrow$  New Worksheet) creates a simulation world, where the Virus Attack simulation will take place. Populate your new simulation world with "Background" and "Person" agents from the Gallery by first selecting the agents in the gallery,

# Worksheet Definition: Agent Size: Custom... V pixels Width: 24 Height: 24 Cancel OK









Sick Person







Background

Building a Simulation of the Spread of a Virus



selecting the **Pencil tool** in the worksheet, and then drawing any number of agents into the worksheet. Let's make a community of healthy people, as shown in Figure 2.



Figure 2. The worksheet is the simulation world.

**Conditions**, **actions**, **and rules**: AgentSheets features a rule-based language, called Visual AgentTalk<sup>™</sup> (VAT), for programming agents' behaviors. In VAT, rules are composed of two parts: conditions and actions. When all the conditions in a rule are true (for example, the actor's health < 5 AND there is no doctor agent nearby), then the actions are executed (the actor becomes sick). An extendible set of Condition and Action commands resides in palettes, shown in Figure 3. These include conditions that check the appearance of agents; read web pages and retrieve data from them; and test the values of an agent's variables; and actions that set these variables to formulae, make sounds, open web pages, change the appearance of neighboring agents, destroy agents, and create new agents.



Figure 3. The Actions Palette (left) and the Conditions Palette (right).

Building a Simulation of the Spread of a Virus

Being a Macintosh-first application, AgentSheets takes advantage of certain platform capabilities by providing actions to play MIDI music using the various QuickTime musical instruments available, to play QuickTime Video, and to speak text using the speech synthesizer. AgentSheets can even include speech recognition commands that allow users to talk to their simulations to interact with them. We believe that speech recognition will be an important future direction for the Mac, and we are looking forward to taking full advantage of it in a future version of AgentSheets.

VAT commands are not just language pieces with enhanced visual representations to help program readability, but they also have interactive interfaces to assist with language comprehension and program writability. Conditions and Actions are elevated to the level of manipulable objects. Before even starting to program using the VAT Conditions and Actions, users can explore the functionality of any command by simply dragging and dropping it onto any agent on the worksheet. When a Condition is dragged and dropped onto an agent, audiovisual feedback informs the user whether that condition is true in the current context. For example, dragging the **Next to** condition onto the Person agent in the bottom left corner of Figure 2 would give negative feedback, as the condition is not true at the given situation (that is, the agent is not next to more than one sick person).

When an action command is dragged onto an agent, that action is immediately performed. For example, dragging the **Move** action onto any of the Person agents in Figure 2 would cause it to move right one square. This novel approach to programming is called Tactile Programming. Tactile Programming facilitates an exploratory style of programming by allowing users to experience the function of language elements. Every condition, action, rule, or method can be tested at any time with any agent to see how things work. This is a big advantage over traditional programming approaches that require users to build a complete program first before they can test it.

**Creating agents' behaviors:** Agents' behaviors are created by combining Condition and Action commands into rules and by grouping rules into methods. Open the behavior editor for your Person agents by doubleclicking any of them in the worksheet. An empty behavior is brought up, as shown in Figure 4. Make the Person agent move randomly on the gray background by dragging the **Move random on** action into the Then box on the right side of the empty rule.



Next to



Move



Move random on

Behavior: Person

Figure 4. The Behavior editor for the Person agent.

Each command is like a little smart template, keeping users from making syntactic mistakes. For instance, instead of specifying depictions by typing in names, depictions are pop-up dialogs providing only valid choices to users.

At any point during the creation of a simulation, rules can be tested by running the simulation. Also, entire rules can be dragged onto an agent to test them in specific contexts. If the rule can fire—meaning that all its conditions are true—it will execute all its actions. If the rule cannot fire, it will indicate why by making the unsatisfied condition blink.

Now that you have made your people move around, let's spread the virus. Create a new rule in the Person agent's behavior by clicking on the New Rule button. Make the Person agent get the virus with a 5 percent chance if it is next to at least one infected person by dragging the **Next to** condition and the % **chance** condition into the If part of the new rule, and by dragging the **Change** action into the Then part of the rule (Figure 5). Also, add a **Move random on** action command so that the sick person does not get stuck.

# 👂 Run 🛛

% chance

% chance 5



Change

Building a Simulation of the Spread of a Virus



Figure 5. The Person agent's behavior.

Apply your newly created rules by clicking the Apply button. To start the virus attack, add a Sick Person agent in your world and run the simulation to observe how the virus spreads. Eventually your entire community gets infected with the virus.

To program doctors to heal the sick people, open the doctor's behavior editor and add a rule specifying that if the doctor sees a sick person to its left by using the **See** condition, then the doctor heals the person by changing it back to the healthy-looking depiction with the **Change** action.

In summary, a person can pick up a virus from a sick person with a 5 percent chance and will be healed by a doctor with a 100 percent chance. Try to predict the effectiveness of this health system. Introduce a few doctors in your simulation world and observe how the virus spreading and healing unfolds.

#### 2. Publish the simulation as a Java applet

Once the programming is done, with the press of the Ristretto<sup>™</sup> button, you turn the simulation into a Java applet embedded in a web page (Figure 6). There is no need to learn Java programming.









Building a Simulation of the Spread of a Virus



Figure 6. The Virus Attack simulation running as a Java applet in a web page.

Your applet is ready to run. Upload it for other users onto a web server, launch it on your Mac, or try the one we made for you.

Adding simulation properties: From the running Virus Attack simulation (either as an AgentSheets simulation or as a Java applet), you can observe the visual representation of the spreading virus in the community. However, it is difficult from the simulation alone to infer at what rate the virus spreads. Is the number of infected people over time a linear curve, an exponential one, or other? Since our goal was to create a Virus Attack simulation and experiment with the spreading rate of the virus, it would be desirable to have a means of graphing the number of infected people over time. AgentSheets enables you to turn your simulation into a JavaBean that can be connected to other components, such as a SimCalc component to graph the total number of sick people over time [ESCOT].

While the JavaBean generation is taken care of by Ristretto, the simulation needs to export the number of infected people so that it can be graphed over time. To do that, you need to create a Simulation Property, which is a global property accessible to all the agents in a simulation. Create a new Simulation Property in the editor (menu Tools  $\rightarrow$  Simulation Properties) called "Total." This property needs to be incremented every time a person gets sick, to reflect the correct number of sick people in the simulation at any given time. To do this, you need to edit the rules of the Person agent by adding the **Set @Total** action at the point where the



Simulation Properties

Set	@Total	to
@T	otal + 1	

Set @Total

Building a Simulation of the Spread of a Virus

person gets sick (Figure 7). And to be complete, you should edit the doctor's behavior, to decrement the Total property by one every time the doctor heals somebody. Please note that the "@" before the property name is just a syntactic way of differentiating between agents' variables and global simulation variables. Also note that a See command was added to the "Become Infected" rule, so that it applies only to healthy people.



Figure 7. The Person's behavior after adding the "Total" property.

#### 3. Connect simulation to other components

The Virus Attack simulation could be turned into a richer learning activity if it could be connected to other educational components such as plotters, databases, and spreadsheets. To address these kinds of issues, the National Science Foundation is supporting the Educational Software Components of Tomorrow [ESCOT] project with the goal of exploring the use of Javabased component technology in education. To this end, ESCOT brings together researchers, practitioners, developers, curriculum designers, publishers, and content experts.<sup>1</sup> Along with SimCalc and Geometer's Sketchpad<sup>®</sup>, AgentSheets is one of the cornerstone ESCOT tools to generate educational components and build math activities.

In Figure 8 the Virus Attack simulation is connected to the SimCalc graphing component to plot the number of infected people while the simulation is running. This allows users to track the spread of the virus through graphical and numerical data, as well as simulated visual data. The ESCOT building tool is not yet publicly available. Therefore we have left out the steps to connect the components and show only the finished result here.



Figure 8. ESCOT Connected Math activity combining AgentSheets Virus Attack simulation component (left) with SimCalc graphing component (right).

The Virus Attack example activity will be appearing as a Math Forum Problem of the Week. [Math Forum] Such activities are linked to national curricula; in this case, mathematics.

#### Explorations

Most educational simulations are open-ended in nature. There is always some additional factor that could be added, new avenues to be explored, and some additional questions to be asked. Here are some suggestions that you and your students can consider, though we suspect that you will probably think of ones we never have.

- 1. What is the optimal ratio (if there is one) of doctors to people to keep the disease under control?
- 2. What if there was a medication that would preserve immunity for some finite amount of time?
- 3. What if people died when they are not healed within a certain time frame?
- 4. What if there was a gestation period for the virus? In other words, there could be a delay between getting the virus, showing symptoms, and being able to spread the virus any further.

Building a Simulation of the Spread of a Virus

#### **About AgentSheets**

The AgentSheets research started in 1989 with the goal of creating a versatile simulation-authoring tool allowing a wide range of people to explore, comprehend, and communicate complex ideas through interactive simulations. Completely new programming approaches were required to empower end users such as children to build their own simulations. AgentSheets innovations, including the combination of graphical rewrite rules and programming by example, dramatically lowered the threshold of programming. Eight-year-old students could create simulations by simply manipulating objects on the screen instead of typing in obscure programming text. After collaboration with Apple Computer started in 1993, graphical rewrite rules also got adopted into the prototype simulation building kits done by Apple.

Ristretto, a unique Java generator included in AgentSheets, turns every simulation at the press of a button into a complete interactive web page. Simulations in web pages run as Java applets or JavaBeans. Users can modify and interact with the running simulations in ways similar to interacting with commercial simulations such as SimCity<sup>®</sup>. The ease with which nonprogrammers can build applets is probably best described by a group of users in Faridabad, India, who in the summer of 1998, despite several power outages, extreme heat, and with just a handful of Macintosh computers, created exciting Java content:

"When the Jiva-Java Project started, its 20 secondary students had never used a mouse or seen the web. Three weeks later they had created applets (using the AgentSheets authorware) and linked them to their own web pages."

AgentSheets has been designed to serve the needs of lifelong learning. To be able to create more realistic simulations, the initial programming-by-example model got replaced with the more powerful Visual AgenTalk language described in this article. Visual AgenTalk can be tailored toward specific application areas in lifelong learning, not just K–12 education, to enable a wider range of users to build useful simulations. Thousands of simulations have been built and published around the world. The following pages show some examples.

Building a Simulation of the Spread of a Virus

#### K-12 Education: Elementary School



**Collaborative Learning:** Students learn about life science topics such as food webs and ecosystems by designing their own animals. The AgentSheets Behavior Exchange is used to facilitate collaborative animal design. Groups of students put their animals into shared worlds to study the fragility of their ecosystems.

#### K-12 Education: High School



**Interactive Storytelling:** History students create interactive stories of historical events such as the Montgomery bus boycott.

#### Training



Distance Learning: With SimProzac, patients can explore the relationships among Prozac, the neurotransmitter serotonin, and neurons. By playing with this simulation in their browsers, patients get a better sense of what Prozac does than by reading the cryptic description included with the drug.

#### Scientific Modeling



Learning by Visualization and Modeling: The effects of microgravity on E. coli bacteria are modeled by NASA. This is a simulation of an experiment that was aboard the Space Shuttle with John Glenn. This simulaton requires several thousand agents.

Building a Simulation of the Spread of a Virus

#### **Educational Games**



Learning Through Simulation Use: This simple voting simulation explains concepts such as clustering, migration, and stability of two-party political systems. Can it predict the outcome of the election in 2000?

#### **Noneducational Games**

00	• •	00	000	• •	• •	• •	0	्]
ĕ (	0 0	 	000	6	• •	0 0	P	33
°	0 0	1	00	1 0	0 0	 		0 0
0 0 0 1	0 0	00	) 0 0   0 0	10	0 0	00		0 0
0 0	•   • •	00	0	-	<u>.</u>	00	) 0 	0 0
0 U 0 0	0 =	( ) c	000		0 × 0 0	00		0 0
100		2015/2015		9119			1921218	

Learning Through Design: Even if the finished simulation or game is not directly related to educational goals, the process of building the simulation may be very educational. The Ultimate Pacman is a complete game based on complex artificial intelligence algorithms and the nontrivial math of diffusion processes.

#### Interactive Illustrations



How Does a TV Work? This simulation illustrates how a picture is scanned in by a camera (left), transmitted to a TV set, and converted back into a picture (right). Users can paint their own pictures and play with TV signal processing parameters.

#### **Deconstruction Kits**



Learning by Taking Apart: What makes a bridge stable? The goal presented to the users of this simulation is to remove as many elements of the bridge as possible without making the bridge collapse. A number of connected issues are revealed, including forces, architecture, and geometric perspective. This simulation was featured on the PBS show *Mathline*.

Building a Simulation of the Spread of a Virus

More information, free demo versions, papers, simulation book references, and additional example simulations can be found at the AgentSheets web site [AgentSheets].

Here is just a short list of some of the unique features of AgentSheets:

- Fastest: AgentSheets simulations run the fastest. No other tool can produce this kind of simulation running this fast natively on Macintosh computers or in Java on Mac, Windows, and UNIX systems. Even in simple simulations, speed quickly becomes an issue.
- No players, no plug-ins: Ristretto-generated Java applets run in web browsers without plug-ins. Other simulation-authoring tools require player software that viewers of your content need to download before they can use your simulation.
- School hardware/software friendly: The simulation engine and the authoring tool do not require the latest and greatest computers. They are highly compact and can run on older hardware, including Macintosh II systems. This capability allows AgentSheets to run on hardware typically found in schools with older CPUs, less RAM, and old versions of the operating system, and without the need to have Java installed.
- Most interactive: AgentSheets simulations can be changed at any time. While the simulation is running, users can change the scene, add or remove agents, and even modify the behavior of agents.
- Multimedia: AgentSheets provides access to Apple technologies including QuickTime musical instruments (MIDI), QuickTime video, speech, sound input, and sound output.
- Collaboration support: The AgentSheets Behavior Exchange [Behavior Exchange] allows users to collaborate on simulations by gathering, annotating, and exchanging agents through a web-based repository of agents.

#### Acknowledgements

AgentSheets research and AgentSheets Inc. are supported by the National Science Foundation (DMI 9761360, REC 9804930, REC 9631396, RED 9253425, CDA-940860). ESCOT is supported by NSF grant REC 9804930. Many thanks to Michele Payton, who worked on the Virus Attack video.

Building a Simulation of the Spread of a Virus

#### URLs

# General AgentSheets Resources

- [AgentSheets] Information, free demo versions, papers, simulation book references, and additional example simulations at www.agentsheets.com
- [Behavior Exchange] A place to exchange agents: www.agentsheets.com/behavior-exchange.html

#### Resources to help build the Virus Attack simulation

- Run the project as an applet running in a web page at www.agentsheets.com/applets/virusattack
- Watch a QuickTime movie on creating the Virus Attack simulation at www.agentsheets.com/videos/virusattack.mov
- Download the Virus Attack project at www.agentsheets.com/projects/virusattack.hqx
  To run the project with simulation properties, you need to have AgentSheets 1.4b3 or later; if you are using an older version, such as 1.4b2, you can download a version of the project without the properties at www.agentsheets.com/projects/virusattack-no-properties.hqx
- [Icon Factory]: The Icon Factory features a large collection of high-quality icons: www.iconfactory.com

#### Components and Curriculum

- [ESCOT] Educational Software Components of Tomorrow: www.escot.org
- [Connected Math Curriculum] www.showmecenter.missouri.edu/showme/cmp.shtml
- [Math Forum] www.forum.swarthmore.edu
- [PBS Mathline] www.pbs.org/teachersource/math/concepts/nov98/activity3.html
- [Simulations] www.apple.com/education/LTReview/current/simulation/

Building a Simulation of the Spread of a Virus

#### About the authors

Alexander Repenning is the director and a founder of AgentSheets Inc. He is also a computer science professor at the Center of LifeLong Learning & Design at the University of Colorado, Boulder. His research interests include end-user programming, computers in education, agent-based simulations, component software, and visual programming. He received a Ph.D. in computer science from the University of Colorado. He is a member of the IEEE, the ACM, and the Association for the Advancement of Computing in Education. Contact him at alexandr@agentsheets.com.

Andri Ioannidou is a senior programmer for AgentSheets Inc. and a doctoral student at the University of Colorado, Boulder. Her research interests include educational uses of technology, end-user programming, and interactive simulations as storytelling. She received her B.S. and M.S. in computer science from the University of Colorado. Contact her at andri@agentsheets.com.

Jonathan Phillips is a senior programmer for AgentSheets Inc. and a doctoral student at the University of Colorado, Boulder. His interests are end-user programming and communities of learning. He received a B.S. in computer science from the University of Colorado. Contact him at jonathan@agentsheets.com.