A virus is infecting your world. As people move randomly and come face to face with a sick person, they too become sick. Learn more about how illnesses can spread through this simulation.

Created by: Susan Miller, University of Colorado, School of Education

This curricula has been designed as part of the Scalable Simulations Design project. It was created using ideas from and portions of prior work completed by Fred Gluck.

This material is based upon work supported by the National Science Foundation under Grant No. DRL-1312129 and CNS-1138526. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Lesson Objective:
- To create a simulation of an epidemic
- To explain and use the Computational Thinking Patterns listed below

Prerequisite Skills:
- Students are presumed to know the following skills. Return to the Frogger Lesson Plans for detailed instructions on these skills.
- Create agents
- Basic agent behavior including:
  - Key control
  - Random movement
  - Ending the simulation

Computational Thinking Patterns:
- Cursor Control
- Collision
- Simulation Properties

Challenge projects:
- Diffusion
- Hill-Climbing
- Polling

Length of Activity:
- Five to Eight 30-45 minute lessons, although some students may advance more quickly

Activity Description:
- Part 1: Create basic world (worksheet) and the agents
- Part 2: Program the illness to spread from person to person
- Part 3: Enable sick people to visit the hospital to recuperate
- Part 4: Polling and ending the simulation

Challenges:
- Deaths
- Quarantine
## Table of Contents

**Teacher Instructions**: Part 1 – Create Worksheet and Agents  
**Student Handout 1A**: Part 1 – Create Worksheet and Agents  
**Student Handout 1B**: Part 1a - Create a Simulation  
  Part 1b – Create Agents  
  Part 1c – Create a Worksheet  
**Student Handout 1C**: Agent Creation Models  

**Teacher Instructions**: Part 2 – Understand Agents Behaviors  
**Student Handout 2**: Part 2 – Understand Agents Behaviors  

**Teacher Instructions**: Part 3 – Program the illness to spread  
**Student Handout 3**: Part 3 – Program the illness to spread  

**Teacher Instructions**: Part 4 – Program the hospital  
**Student Handout 4**: Part 4 – Program the hospital  

**Teacher Instructions**: Part 4 – Polling and Ending the Simulation  
**Student Handout 4**: Part 4 – Polling and Ending the Simulation
Vocabulary/Definitions

Agent Attribute..........................an assigned feature of an agent (such as scent)
Algorithm ....................................a set of instructions designed to perform a specific task.
Brackets ................................method of setting information apart using “[“ and “]”
Broadcast ................................A way for agents to communicate with other agents that are not adjacent to them - agents broadcast (or send out) a signal
Collision ...........................................an event wherein two agents interact when next to each other.
Decrement ......................................to decrease by one
Diffusion ...........................................the process in which an attribute moves from areas of strong concentration to weak concentration
Increment ......................................to increase by one
Hill Climbing ...............................a specific form of searching/seeking technique, or algorithm, by which the seeking/searching agent uses information (agent attribute) embedded in the floor.
Method ..........................................a set of rules to follow in a specific situation
Parentheses .................................method of setting information apart using “(“ and “)”
Polling ..........................................the process of contacting and communicating with each agent
Propagated .................................the spreading of an agent attribute
Randomly ................................to occur in non-systematic ways
Rule Order .................................the order in which rules are placed for each agent
Simulation Property .................an attribute (value) accessible by all agents.
General Teaching Strategies

Basic Philosophy

- The educational goal of these lessons is to learn and apply Computational Thinking Patterns in the context of a familiar game. Emphasis on these Computational Thinking Patterns is essential for student understanding.

- Every effort has been made to create instructions with an eye toward guided discovery. Direct instruction has been used for those aspects where students are learning the code for the first time; however, materials have been provided to ensure that students understand the programming concepts, as opposed to simply copying code. Note that special materials have been designed for students who are new to this program.

- Student materials are available for each portion of the game design. These materials are intended to be used in addition to teacher materials, which provide prompts and discussion points. Students may become frustrated with too little teacher support. Students may lose out on conceptual understanding with too much teacher support.

Guided Discovery Process

- **Model the process** rather than just giving students the answer. Building the game on your own, before trying it with your students will enable you to see possible struggling points.

- Have students work through problems on their own. Ask directing questions or give helpful suggestions, but **provide only minimal assistance** and only when needed to overcome obstacles.

- Don’t fear **group work**! It is common for computer programmers to talk through problems with one another, and to use code snippets found from other programs, and other programmers. Talking through coding problems enables students to think more critically about Computational Thinking Patterns, as well as the steps needed to solve a problem. Additionally, seeing how others solved an

---

1 This information is supported by research found in the following documents:

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010, June). Using scalable game design to teach computer science from middle school to graduate school. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 224-228). ACM.


issue with code helps students realize that problems often have multiple solution strategies, and some that might be more effective than others

- Recognize that programming is largely a process of trial and error, particularly when first learning. It is helpful to encourage this mindset with your students.

**Building Blocks**

- Each project is designed to build on the prior one. Very little student support is provided where expertise has already been created. Conversely, material that is new has more support.

- Be sure to talk through the building blocks (especially for PacMan in the area of diffusion and hill climbing) as these Computational Thinking Patterns will appear often in future games and simulations.

- Remember that conceptual understanding takes time, and it may be necessary to explain these concepts multiple times, using different examples, so that all students can be successful.

**Support Learning**

- Research shows that game design is associated with engaged students, and engaged students show higher levels on conceptual understanding. Allowing students to personalize their games aids in this engagement and motivation.

- Coding may be difficult for some students, and all students are likely to be frustrated at times when the code does not produce the expected results. Praise students for sticking with the troubleshooting process and encourage them to share what they learned with others.

- Be sure to communicate that the process is more important than the answer, and that coding of a project often takes time. Do not place pressure on your students to ‘hurry up’ and resort to giving them the code. The process of figuring it out on his/her own will result in much stronger conceptual understanding.

**Differentiated Instruction**

Note that there are many vocabulary words in this lesson that may be new for your students. Take time to define those words. Using the words in context often will reinforce their meaning for the students.

- **Students who need a challenge:** Some students with more fluency in programming may finish this very quickly – be prepared for them to move on earlier than other students by having student materials ready in advance.

- **Students who need more assistance:** Other students (especially those with no Frogger experience) may struggle a bit more. There are two options for differentiated instruction. Consider the needs of the student and the class as you decide which will work best.
Contagion (Continued)

- **Lesson 1:**
  - Option 1: Pair a struggling student with an experienced student
  - Option 2: Provide struggling students with Handout 1A, which provides more directed instruction steps
  - Vocabulary for ELL Students: Contagion, Sick, Healthy, Randomly, Rule Order, Depiction
  - Time management issues: While students can be more engaged when they design their own agents, some students can spend too much time on this design or find it frustrating. Suggest students use a pink box for a healthy person and a green box for a sick person.

- **Lesson 2:**
  - Pairing the student with an experienced student should alleviate many problems.
  - Vocabulary for ELL Students: Algorithm, propagated, attribute, local variable, diffusion, method, parentheses, brackets

- **Lesson 3:**
  - Note that this is a challenging lesson for students – plan extra time and additional help from more experienced students.
  - Pairing the student with an experienced student should alleviate many problems.
  - Vocabulary for ELL Students: polling, broadcast, increment

Note that there are two versions of the student pages. The STANDARD version is designed for most students. It presumes basic knowledge of AgentSheets. The ALTERNATIVE version is designed for those students who may need more support. It provides explicit directions for the first part of the project, and step-by-step instructions for the challenges.
Teacher Instructions:

Part 1 – Basic Contagion

Introduce this project to the students by asking them how diseases are spread.

- Ask students to explain how colds are spread, and determine the ‘rules’ of an epidemic
  - Healthy people and sick people walk around the world
  - When they come into contact with one another, they sometimes get sick
  - When sick people rest (or go to the doctor or hospital) they usually (but not always) get better
- You may choose to show this video by Emery University that uses images from Contagion, the movie to describe how diseases spread.
  - [https://www.youtube.com/watch?v=OH9_hZ9uomk](https://www.youtube.com/watch?v=OH9_hZ9uomk)

If you show these videos, ask your students…

- How does the disease spread?
- What happens as more and more people get sick?
- Which parts can we model? Which parts may be difficult to model?
- Are there variables? For example, does it matter how contagious it is? Does it matter how long the illness usually lasts? Does it matter if you go see a doctor?

Explain that these are all design features that must be considered when planning a simulation. Now, tell them that they will be designing their own contagion simulation.

As a class, briefly create a description of the Contagion simulation similar to the one below.

- identify simulation objects, called agents, by locating nouns in the simulation description
  - When the nouns are the same, with different descriptive adjectives, we may want to use different depictions.
- identify agent interaction by locating verbs in the simulation description
It might look something like this:

Healthy people and sick people walk around the world. When they come into contact with one another, they sometimes get sick. When sick people rest (or go to the doctor or hospital) they usually (but not always) get better.

Agents:
- people (two depictions: healthy and sick)
- ground

Actions:
- Next to, at some probability, change to sick people
- After some time, get better

Give the students 10 minutes or so to work with a partner to discuss what steps will be needed to create this simulation. If students have already completed Frogger and Journey or PacMan, they should have a pretty good idea of how to do this. Add new students to a pair of experienced students who will be willing to talk through their thinking. Stress the need to think through the programming process. At this point, there should not be hands on the keyboards (even though some will want to jump right in to programming!).

This lesson is intended to be taught in a guided discovery manner. Be sure to give students time to work on their own and figure things out using the program. Encourage students to work together and talk through problems with one another. Emphasize that troubleshooting is a normal and important part of programming.

Solicit and discuss possible ideas, without providing any evaluative feedback (do not tell students if their ideas are good/bad, right/wrong). Once there has been some class discussion, provide students with the appropriate handout.

- **STUDENT HANDOUT 1A** simply provides the details needed for experienced students to get started on their own. (Page 3 of the Student Only Pages)
- **STUDENT HANDOUT 1B** provides added details needed for students to get started on their own WITH the code provided. (Page 4 of the Student Only Pages)
- **STUDENT HANDOUT 1C** Provides step-by-step instructions for creating a worksheet and agents for students who are new to AgentSheets. (Page 3 of the Student Only – No Agent sheet experiences pages)
Part 1 – Basic Contagion

You will be modeling or simulating an epidemic. We will begin with a very simple model and adjust it to make it more accurate.

Tasks:

1. Create a new project called Contagion
2. Create agents for the simulation. You will need the following agents:
   - Ground
   - Person (two depictions, healthy and sick)
3. Create a basic worksheet that is covered in ground, with many healthy people and one sick person.
4. Program the agents using a method to make decisions:
   - Every 0.1 seconds DECIDE what happens next
     - If you are healthy and next to a sick person, determine if you get ill using the method TEST_GET_SICK.
     - Create a rule so that when a healthy person is next to a sick person, he SOMETIMES gets sick. Consider what percent of the time that should be.
     - If you are healthy and not next to a sick person, you should just move randomly on the ground.
     - If you are sick, you should move randomly on the ground.
5. Once your simulation is programmed, run it to see what happens. Work with a partner to answer these questions:
   - What happens when the percent is changed to zero? What would that mean in real life?
   - What happens when the percent is changed to 100. What would that mean in real life.
   - What percent is realistic for this simulation if it were the common cold? What percent is realistic for this simulation if it were Ebola?
   - What causes the simulation to stop? Why?
Part 1 – Basic Contagion

You will be modeling or simulating an epidemic. We will begin with a very simple model and adjust it to make it more accurate.

Tasks:

1. Create a new project called Contagion
2. Create agents for the simulation. You will need the following agents:
   - Ground
   - Person (two depictions, healthy and sick)
3. Create a basic worksheet that is covered in ground, with many healthy people and one sick person.
4. Program the agents using a method to make decisions:
   - Every 0.1 seconds DECIDE what happens next
     - If you are healthy and next to a sick person, determine if you get ill using the method TEST_GET_SICK.
     - Create a rule so that when a healthy person is next to a sick person, he SOMETIMES gets sick. Consider what percent of the time that should be.
     - If you are healthy and not next to a sick person, you should just move randomly on the ground.
     - If you are sick, you should move randomly on the ground.
5. Once your simulation is programmed, run it to see what happens. Work with a partner to answer these questions:
   - What happens when the percent is changed to zero? What would that mean in real life?
   - What happens when the percent is changed to 100. What would that mean in real life.
   - What percent is realistic for this simulation if it were the common cold? What percent is realistic for this simulation if it were Ebola?
   - What causes the simulation to stop? Why?
### Part 1: Create Agents

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Create Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click on the new simulation icon (far left)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Name the Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name it Contagion and click OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3</th>
<th>Define Agent Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Do not change - Click OK</td>
</tr>
</tbody>
</table>

![Define Agent Size](image-url)
<table>
<thead>
<tr>
<th>Step 4</th>
<th>Create Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click on New Agent</td>
</tr>
<tr>
<td></td>
<td>Name it Contagion</td>
</tr>
<tr>
<td></td>
<td>Click ok</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 5</th>
<th>Edit Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click Edit Depiction</td>
</tr>
<tr>
<td></td>
<td>Click Clear to erase the current image.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 6</th>
<th>Draw the ground as a solid color.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click Done</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 7</th>
<th>Draw the person with two depictions.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add a second depiction by clicking on New Depiction and naming it sickpeople. Then, edit the depiction to make the person look sickly. Click done.</td>
</tr>
</tbody>
</table>
Part 2: Create the Worksheet

Select Tool

Pencil Tool – places a single agent on the worksheet
Eraser – erases agents from the worksheet
Will be defined later
Will be defined later

Draw Rectangle – places agents in an array (rectangle)
Erase Rectangle – erases agents in an array
Will be defined later

The worksheet is the simulation space – it is where the agents will perform their actions.

<table>
<thead>
<tr>
<th>Step 8</th>
<th>Make the worksheet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Click File&gt;&gt;New Worksheet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 9</th>
<th>Make the worksheet bigger</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Notice it is big, but not so big that it fills up the whole space.</td>
</tr>
<tr>
<td>Step 10</td>
<td>Use the tools to place the floor on the worksheet. Then place many healthy people and one sick person on the floor.</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Pencil: places agents one at a time (Works well for people)</td>
</tr>
<tr>
<td></td>
<td>Filled in Rectangle: Places agents in an array. (Works well for the floor)</td>
</tr>
<tr>
<td></td>
<td>It is important that you do not draw over the agents. Agents can stack on top of one another which is not visible from a 2-dimensional perspective. Take care to create a single layer of agents.</td>
</tr>
</tbody>
</table>

This is a good time to save the worksheet
Contagion (Continued)

Part 3: Program the Agents

Click on the agent and then click EDIT BEHAVIOR to add behaviors to that agent.

Three boxes will appear: conditions, behaviors, and actions. It helps to arrange your page so they stay in this order. You can click on the blue or red coloring to drag in conditions and actions. If you make a mistake, drag it to the trash.

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Create behavior for the person</th>
<th>Click on the person agent. Then, click EDIT BEHAVIOR.</th>
</tr>
</thead>
</table>

Now it’s time to program…

Every 0.1 seconds DECIDE what happens next
If you are healthy and next to a sick person, determine if you get ill, using the method TEST_GET_SICK.

6. Create a rule so that when a healthy person is next to a sick person, he SOMETIMES gets sick. Consider what percent of the time that should be.
   - If you are healthy and not next to a sick person, you should just move randomly on the ground.
   - If you are sick, you should move randomly on the ground.
### Contagion (Continued)

<table>
<thead>
<tr>
<th>Step 2: Create rules:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy people become sick when they are next to sick people (at some percent of the time!)</td>
</tr>
<tr>
<td>Click NEW RULE:</td>
</tr>
<tr>
<td>If they are NOT next to the sick person, the second rule is true.</td>
</tr>
</tbody>
</table>

Click on EXPLAIN to understand the rule. It says:

IF…
I see that I am healthy
AND I am next to at least One sick person
THEN check to see if I get sick

When I check to see if I get sick, my rule says,
IF the % chance is 100%
THEN change me to a sick person.

Talk with the person next to you – is 100% logical? If not, change it to a more reasonable number.

The other rules (which only activates when the healthy person is NOT next to a sick person) tell the people (both sick and healthy) to move randomly on the floor.

Once your simulation is programmed, run it to see what happens. Work with a partner to answer these questions:

- What happens when the percent is changed to zero? What would that mean in real life?
- What happens when the percent is changed to 100. What would that mean in real life?
- What percent is realistic for this simulation if it were the common cold? What percent is realistic for this simulation if it were Ebola?
- What causes the simulation to stop? Why?
Teacher Instructions: Using simulation properties

Students will find that they want to quickly be able to change the percentage of time that the healthy people become sick. To do this, students can create a simulation property (referred to as a simulation property). If students have not worked with simulation properties before (in Journey or PacMan, for example) this is best taught as a whole class session once the issue is raised. If the students have worked with simulation properties in the past, challenge them to use them now. Give them time to figure it out on their own before you offer help.

Rather than include a percent in the code, list the % chance as @percent. Remember that the @ symbol makes it a simulation property, for all agents.

Then open the simulation properties (TOOLS>>SIMULATION PROPERTIES).

Click on EDIT to edit the properties. Make the changes shown below (click box, add min/max)

Before changes: After changes:

![Simulation Properties](image)

Note that the MAX value has changed to 100. This allows the value to act as a percentage, from 0 – 100%.

These changes enable the user to change the percentage from 0% - 100% with a slider. Click OK - It now looks like this:
Teacher Instructions: Part 2 – Getting Better

Making the simulation more realistic – Adding in the length of the illness

In this next lesson, students will learn to improve their simulation by making it more realistic. Take a moment to talk with the students about getting sick. Consider these prompts:

- What happens when you get sick?
- Do you stay sick forever?
- How do you get better?
- Do you have to go to the doctor to get better?
- Why/Why not?
- How long are you sick with a cold?

You may choose to watch this video with your students, describing the common cold: http://www.youtube.com/watch?v=UWgiyQV3nYc

Near the end of the video, the narrator states that the common cold lasts 7-10 days if you have ‘average health.’ This brings up some more topics for discussion...

- What is ‘average health’?
- What makes colds last longer?
- What might shorten the length of a cold?
- How can we program 7-10 days, when it’s not a fixed value?

Students will now build in a ‘timer’ to their simulation, to model the idea that people get better after 7-10 days. To do this, we will use a random number generator with the code Random(variable_name). Random creates a random number from 0 to the value of the variable. So, if we say that someone is sick from 7 – 10 days, that means that we can represent this as:

- 7 + (a random number between 0 and 3)
  - If the random number is 0, the person will be sick 7 days
  - If the random number is 1, the person will be sick 8 days
  - If the random number is 2, the person will be sick 9 days
  - If the random number is 3, the person will be sick 10 days

You can model this with students using numbers in a hat from 0 – 3, and allowing them to draw a number randomly. (Be sure to have many repeated numbers in the hat, so that the draw is random.) If they draw out a zero, they were sick for 7 days. If they draw out a 3, they were sick for (7+3) 10 days. Once students understand the concepts, pass out the Student Handout, which can be found on page 5 of the Student Only Pages, and page 9 of the No AS Experience student pages.
Part 2 – Getting better!

As you discussed with your class, you don’t stay sick forever – even though you might feel miserable with a cold, you will get better, generally in a week or so. We are going to change the simulation to reflect that people do get better, and it takes 7 – 10 days.

We are going to designate that all sick people will take at least 7 days to get better (using a simulation property MINIMUM_SICK_TIME), and some might take a bit longer (as many as 10 days) using a randomly generated value of the variable (EXTRA_SICK_TIME). We will use a variable called as SICK_CLOCK to count down how long someone has been sick. We will use the TEST_RECOVERED method to check how many days are left of being sick.

Confused? Don’t be – we’ll take this step by step.

First, let’s create some variables. Open the simulation properties (Tools>>Simulation Properties)

Click on NEW, and title it minimum_sick_time. Click OK. You should now see this:

Set this to 7 (to represent the 7 days) and click on SAVE.

Now, repeat this process and create the variable extra_sick_time. Set it to 3 (to represent the three extra days someone could be sick) and click SAVE.

Your simulation properties should now look like this:

To do the remaining programming, we will do it in steps...

Step 1: When a person gets sick, start the sick clock countdown

Step 2: If I am sick, check to see if I’m still sick each ‘day’
Step 3: Check to see if the sick clock is down to zero – if it is, change me back to a healthy person. (If not, count down another day - I stay sick).

Step 4: Create a sick clock starting time for your first sick person (already on your worksheet).

Let’s get started!

**Step 1:** When I become sick, set the sick_clock to the minimum_sick_time plus a random amount of extra_sick_time.

\[
\text{Set } \text{sick\_clock to } @\text{minimum\_sick\_time} + \text{random}(@\text{extra\_sick\_time})
\]

**Step 2:** Once I’m sick, check to see if I am still sick.
Step 3: Add the method test_recovered to count down the sick clock and then change me back to healthy if the clock is zero.

Step 4: Before you test this, you need to set the sick clock for your very first sick person. To do this, add this code to your person:

If I am sick, Then, Set sick_clock to @minimum_sick_time + random(@extra_sick_time)

Delete your original sick person from your worksheet and add a new sick person. SAVE YOUR WORKSHEET. You can check to see the sick_clock value for your sick person by clicking on the arrow on the worksheet, and then clicking on the sick person (highlighting him). Then select (Tools>>Agent Attributes). It should show the sick clock for your agent.
Run your simulation several times to see what happens. (Remember that it will operate as fast as your computer processor – you can use the slider at the bottom of the worksheet to slow it down, or you can add a ‘once every’ command to the first line of your code to slow it down.

What happens in your simulation? Is it what you expected?

Open the simulation properties (Tools>>Simulation Properties). Try changing the variables to determine how they affect the simulation.

- What happens if you make the percent who get sick higher? Lower?
- What happens if the cold lasts longer?
- What happens if some people already have a weakened immune system and stay sick longer?
- What happens if there are more sick people in the simulation?
- What happens if there are more healthy people in the simulation?
- In what ways does this simulation reflect the real world? In what ways does the simulation not reflect what really happens?
Part 3: Who’s sick – Who’s not? Gathering and Plotting the data

Teacher Instructions: Using simulation properties to plot the number of healthy and sick people

In this next lesson, students will learn to use simulation properties to track the number of healthy and sick people in their 'world' and then how to plot those properties, which can be found on page 9 of the Student Only Pages, and page 13 of the No AS Experience student pages.

We will introduce the World Health Organization. This organization is responsible for the eradication of smallpox, and is working on eradicating several communicable diseases including HIV, malaria and tuberculosis. This organization is also very involved in trying to mitigate the current Ebola outbreaks. As part of its responsibilities, this organization surveys huge numbers of people every year to determine who has what illness. In our simulation, the WHO will become like the controller in Journey or PacMan and will survey all of the people on the worksheet to determine who is healthy and who is sick.

Step 1: The WHO will set the number of healthy people and sick people to zero. The WHO will then survey the people by broadcasting a request for them to count themselves.

When the sick people are called by the WHO, they count up.
Contagion (Continued)

Students will first watch the data change using the simulation properties window, and then graph the data using the plot function.

To watch the data, students will click on Tools>>Simulation properties. They will see the numbers of sick and healthy people change.

To plot the data, they will use the option to plot them in the Simulation Properties box.

Highlight the property you wish to plot. Click on plot.

Create a window name (I chose Total_people) and change the min/max values to reasonable values for your worksheet.

Do the same for the second variable. Again, be sure to rename the window to match the first same so that they are plotted on the same graph. Change the min/max values to reasonable values for your worksheet. Change the color to a different color.

Run the simulation. Hint: the plot is sometimes hiding behind the worksheet. Students may have to do some rearranging.
Part 3 – Who’s sick – Who’s not?  Gathering and Plotting the data

In this next lesson, you will learn to use simulation properties to track the number of healthy and sick people in their ‘world’ and then how to plot those properties.

Because epidemics can affect huge numbers of people all over the world, the United Nations created the World Health Organization to provide leadership on global health issues. This organization is responsible for the eradication of smallpox, and is working on eradicating several communicable diseases including HIV, malaria and tuberculosis. This organization is also very involved in trying to mitigate the current Ebola outbreaks. As part of its responsibilities, this organization surveys huge numbers of people every year to determine who has what illness. In our simulation, the WHO will become the controller (like the controller in Journey or PacMan) and will survey all of the people on the worksheet to determine who is healthy and who is sick.

Step 1: The WHO will set the number of healthy people and sick people to zero. The WHO will then survey the people by broadcasting a request for them to count themselves.
When the sick people are called by the WHO, they count up.

To watch the data, click on Tools >> Simulation properties. You will see the numbers of sick and healthy people change.

To plot the data, use the option to plot them in the Simulation Properties box.

Highlight the property you wish to plot. Click on plot.

Create a window name (I chose Total_people) and change the min/max values to reasonable values for your worksheet.
Do the same for the second variable. Again, be sure to rename the window to match the first same so that they are plotted on the same graph. Change the min/max values to reasonable values for your worksheet. Change the color to a different color.

Run the simulation. Hint: the plot is sometimes hiding behind the worksheet. You may have to do some rearranging.
Student Handout:

Challenge 1.0: WHO Reporting

Before your start this challenge:

- You must have a complete basic Contagion simulation that enables people to get better after a certain amount of time.
- You have a World Health Organization (WHO) that polls people, asking them to count who is sick and who is healthy.

Design Challenge:
For serious diseases (like Ebola), the World Health organization is responsible for tracking the illness and reporting on the spread of that illness.

Add a method to notify the player when the simulation must end, either because:

- There are no sick people left
- There are no healthy people left.
Student Handout:

**Challenge 2.0: Deaths**

**Before your start this challenge:**

You must have a complete basic Contagion simulation that enables people to get better after a certain amount of time.

**Design Challenge:**

For serious diseases (like Ebola), not everyone gets better. Some people will die. Change your game to have some small percentage of people die. Be sure your percentage of people who die can be changed using a slider.

Hint, to do this, change the second rule in this code:

![Code Snippet]

At the end of the sick time...**EITHER**, I get better, OR I die. So, create a method that checks to see if I live...At some percent of mortality, I **change** to a **dead person**, otherwise I **change** to a **healthy person**.

You should be seeing some new actions and a new depiction.

Something to consider...do dead people continue to move randomly?

Be sure to plot the deaths.
**Student Handout:**

**Challenge 3.0: Quarantine**

Before your start this challenge:

You must have a complete basic Contagion simulation that enables people to get better after a certain amount of time.

**Design Challenge:**

For serious diseases (like Ebola), not everyone gets better. To prevent a spread of the disease (so that it doesn’t become an epidemic), sometimes people are quarantined.

Create a **hospital area** where **sick people go**. They **remain in quarantine** until they are **healthy**.

Again, you should see a new agent and new actions.

Hint: You will need to use diffusion and hill climbing to get the sick people to search out the quarantine area. Look back at Journey if you need a reminder of how to do that coding.

Questions to answer once you are done...

- Does a quarantine area prevent diseases from spreading?
- How long must someone remain quarantined for it to be effective?
- When is quarantining necessary? Should we quarantine people with the flu? The common cold? Cancer?
Contagion

ISTE Standards\(^2\) specific to the implementation of Contagion (Denoted with (\(\star\)))

Creativity and Innovation

*Students demonstrate creative thinking, construct knowledge, and develop innovative products and processes using technology. Students:*

Apply existing knowledge to generate new ideas, products, or processes:

- Design and develop simulations
- Design and develop computational science models

Create original works as a means of personal or group expression.

- Design original simulations
- Model your local environment, e.g., ecology, economy

Use models and simulations to explore complete systems and issues.

- Model scientific phenomena, e.g., predator/prey models
- Create visualizations

Identify trends and forecast possibilities.

- Build predictive computational science models, e.g., how the pine beetle destroys the Colorado pine forest
- Build live feeds to scientific web pages (e.g., weather information), process and visualize changing information

Communication and Collaboration

*Students use digital media and environments to communicate and work collaboratively, including at a distance, to support individual learning and contribute to the learning of others. Students:*

Interact, collaborate, and publish with peers, experts, or others employing a variety of digital environments and media:

- Students work in teams to build and publish their simulations as web pages containing java applets.

Communicate information and ideas effectively to multiple audiences using a variety of media and formats.

- Effectively combine interactive simulations, text, images in web pages

Develop cultural understanding and global awareness by engaging with learners of other cultures.

- Students and teachers from the four culturally diverse regions interact with each other

Contribute to project teams to produce original works or solve problems.

- Define project roles and work collaboratively to produce simulations and simulations

Research and Information Fluency

\(^2\) ISTE Standards for Students (ISTE Standards•S) are the “standards for evaluating the skills and knowledge students need to learn effectively and live productively in an increasingly global and digital world.” http://www.iste.org/standards/standards-for-students

Contagion Curriculum v1.0 Page 32 of 34 Scalable Game Design
Students apply digital tools to gather, evaluate, and use information. Students:

**Plan strategies to guide inquiry.**
- Explore web sites and identify interesting connections

**Locate, organize, analyze, evaluate, synthesize, and ethically use information from a variety of sources and media.**
- Find relevant related web-based information, compute derivate information

**Evaluate and select information sources and digital tools based on the appropriateness to specific tasks.**
- Understand validity of information, e.g. Scientific journal information vs. Personal blogs

**Process data and report results.**
- Write programs to access numerical information; define functions to process data and create output based on voice or plotting to represent data.

**Critical Thinking, Problem Solving, and Decision Making**

Students use critical thinking skills to plan and conduct research, manage projects, solve problems, and make informed decisions using appropriate digital tools and resources. Students:

**Identify and define authentic problems and significant questions for investigation.**
- Define research questions and explore approach of exploration

**Plan and manage activities to develop a solution or complete a project.**
- Outline sequence of exploratory steps
- Experience complete bottom-up and top-down design processes
- Employ algorithmic thinking for creating programs to solve problems

**Collect and analyze data to identify solutions and/or make informed decisions.**
- Collect data as time series, e.g., collect group size of predator and prey, export time series to excel, explore various types of graph representations, e.g., x(t), y(t) or scatter y=f(x)

**Use multiple processes and diverse perspectives to explore alternative solutions.**
- Experience and understand design trade-offs, e.g. Bottom-up vs. Top-down

**Digital Citizenship**

Students understand human, cultural, and societal issues related to technology and practice legal and ethical behavior. Students:

**Advocate and practice safe, legal, and responsible use of information and technology.**
- Learn how to use tools to locate resources, e.g., images with google image search, but understand copyright issues

**Exhibit a positive attitude toward using technology that supports collaboration, learning, and productivity.**
- Stay in the flow, where design challenges match design skills
- Experience success through scaffolded simulation design activities
- Mentor other students

**Demonstrate personal responsibility for lifelong learning.**
Contagion (Continued)

- Explore options of going beyond expected learning goals
- Exhibit leadership for digital citizenship.
  - In a collaborative setting become a responsible producer of content for diverse audiences

Technology Operations and Concepts

Students demonstrate a sound understanding of technology concepts, systems, and operations. Students:

Understand and use technology systems.

- Know how to organize files and folders, launch and use applications on various platforms

Select and use applications effectively and productively.

- Know how to orchestrate a set of applications to achieve goals, e.g., make simulation and simulations using Photoshop (art), AgentSheets (programming), and Excel (data analysis).

Troubleshoot systems and applications.

- Debug simulations and simulations that are not working

Transfer current knowledge to learning of new technologies.

- Reflect on fundamental skills at conceptual level. Explore different tools to achieve similar objectives.