Creating "Contagion" with AgentCubes Online

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

Created by: Catharine Brand and Susan Miller, University of Colorado

This tutorial has been designed as part of the Scalable Simulations Design project. It draws inspiration and content from simulations designed by Fred Gluck and an AgentSheets tutorial written by Susan Miller.

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.

3D-Contagion

Vocabulary/Definitions

Agent Attribute.....**a named value-holder or local variable belonging to an agent, which can be assigned different values. Scent and sick_clock are agent attributes.**

Algorithm .....**a set of instructions designed to perform a specific task.**

Broadcast .....**A way for agents to communicate with other agents that are not adjacent to them - agents broadcast (or send out) a method name, telling other agents to check the rules in that method. The method referenced in the broadcast must be defined in the receiving agents' set of methods.**

Diffusion .....**the process in which an attribute like scent that belongs to a group of agents such as ground agents changes its value, being larger near its source and smaller farther way from its source**

Immunity.....**to keep yourself from being affected by a disease**

Increment .....**to increase by one**

Hill Climbing .....**a specific form of searching/seeking algorithm, by which the seeking/searching agent checks the values of an agent attribute belonging to another agent.**

Method.....**a set of rules to follow when an agent must make a decision.**

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.**

Student Handout 1A: Part 1 – Very Basic Contagion

You will be modeling or simulating the spread of a disease. You will begin with a simple model and then make it more realistic.

In this first model, we have a very basic world. Healthy people and sick people walk around. If you are healthy and next to a sick person, you might get sick. If you are sick, you might recover and become healthy again.

Talk with a partner, and answer these questions:

1. What agents are needed? Think about these agents. Do you need different agents for healthy/sick people, or one agent (people) with two shapes (healthy/sick)?
2. What actions will they have?

Once you have a plan, begin to make the simulation.

Once your simulation is programmed correctly, use the single step control  above the world to run it one step at a time and see what happens.

- What happens when you increase the percent chance of getting sick to 100%? What would this mean in real life?
- What happens when you decrease the percent chance of recovering to 0%? What would this mean in real life?
- What probabilities for getting sick and recovering seemed most realistic?
- Does the simulation end?
- How can you tell how many agents get sick? Die? Recover? Are there ways to use programming to help you do this?
- Do sick people move? Should they?
- In what ways is this a realistic simulation? In what ways does this simulation **not** match what really happens?

Student Handout: Using Simulation Properties

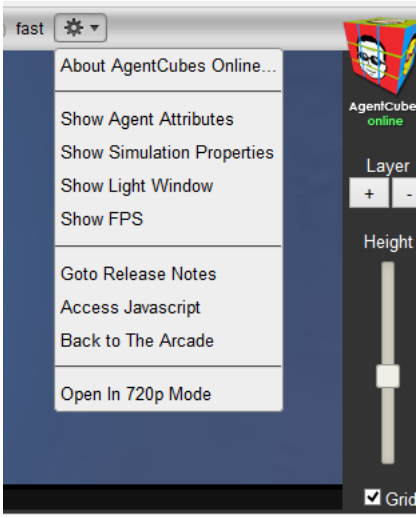
What if we want to look at different percentages to see how the model changed? For example, some diseases like colds, which are spread by coughing and sneezing, pass easily from person to person so there is a big chance that an exposed person will get sick. Other diseases are not transmitted as easily so there is a smaller percent chance that an exposed person will get sick. That's difficult to model when the percentage is built into the code. We can fix that by using simulation properties. Simulation properties (or global variables) are variables that are accessed by all the agents in the simulation. We will use these variables in the percent-chance conditions because it is easy to change the value and different diseases can be modeled.

For example, rather than include a number percent in the code, use **@Get_sick** in the percent condition 

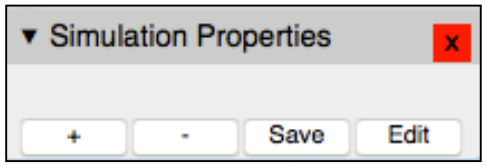
to represent the probability of a healthy person becoming sick when next to a sick person**. Remember that the @ symbol must be used in front of the simulation property name in order to access the its value.**

Follow these steps to create and save the Get_sick Simulation Property:
Step 1: Create a global variable.

Open the simulation properties. Click on the Gear button  in the top bar of the AgentCubes Online window, then choose Show Simulation Properties.



Step 2:
Click on the + button in the Simulation Properties window.

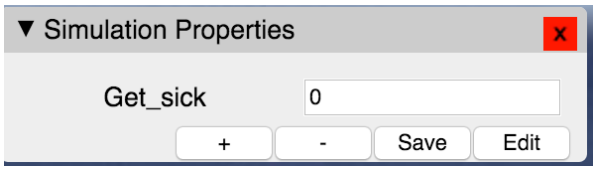


Step 3:
Name the property Get_sick.

3D-Contagion (Continued)

Step 4:

Select the Get_sick property and click on the Edit button.

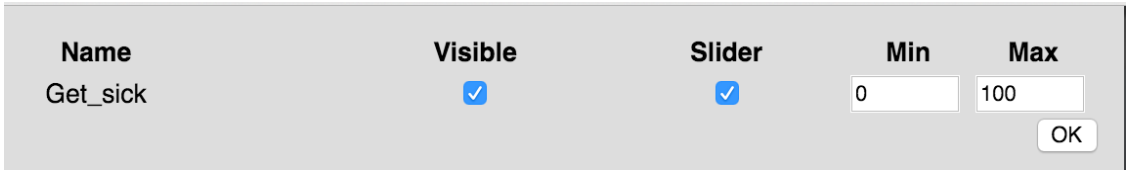


Simulation Properties:

Simulation properties are simply variables. In math, we often use x and y as our variables. Sometimes, if we are talking about girls and boys, we might use g and b as our variables to help us keep track of which item each variable represents. In programming, rather than using a letter, you can use a word, like Get_Sick. This makes it very easy to track each variable.

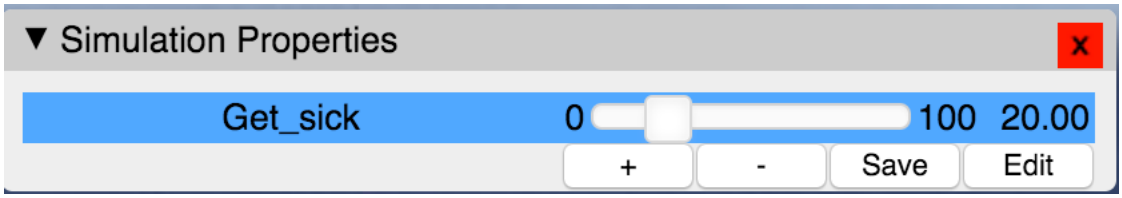
Step 5:

Click the slider button and set the max value to 100.



Step 6:

Now the value of Get_sick can be changed to any number from 0 to 100 by using the slider.



Save your Simulation Property now
or it will not be there next time you open the project!

Follow these steps to create the other simulation property that is useful in the contagion simulation:

Recover = probability that a sick person will recover.
The values of this simulation property must be between 0 and 100.

Student Handout 2a: Adding a Monitor Agent

Step through your simulation again. Pay close attention to who gets sick on each step. You might be seeing people get sick who aren't next to a sick person. This is because the computer does two things you might not know…first, it runs faster than your eye can track. This means that an agent might move without you seeing it. Second, when it checks each agent on the screen, it starts at the top left corner, works through the row to the right, and then moves down to the next row. This doesn't do a good job of modeling what happens in the real world because even though someone may be next to a sick person, they might move before they check to see if they get sick. So we are going to change the programming so that all people agents check to see if they are next to a sick person at the same time!

First each person agent looks around and decides what it should do. When all the person agents have had an opportunity to look and decide what to do, each person agent in turn then does its actions. The monitor agent **scripts** the agents' behavior, telling them first to **PERCEIVE**, then to **ACT**.

We will also need to create a 'switch,' which is a local variable that keeps track of whether each agent should 'switch' from healthy to sick, or sick to healthy. We will call these INFECTED, and GET_WELL.

Agent: Monitor (Be sure to place the monitor on the worksheet!)

Rules:

- The Monitor tells your person agents to PERCEIVE who is around them, and determine if they need to switch. Then, the Monitor tells them to ACT, based on that knowledge.
    1. PERCEIVE
        - If you are healthy and next to at least one sick person, identify yourself as infected (INFECTED = 1), based on some probability
            - Consider what percent of the time that should be.
        - If you are sick, identify yourself as getting better (GET_WELL = 1), based on some probability.
            - Consider what percent of the time that should be.
    2. ACT
        - If you are healthy and now infected, change yourself to a sick person, reset your INFECTED switch back to zero.

3D-Contagion (Continued)

- If you are sick, and no longer infected, change yourself to a healthy person, reset your GET_WELL switch back to zero.
- Otherwise, move randomly around the world.

PERCEIVE and ACT are methods. Using a METHOD is like having a special set of procedures that get done only when asked. A fire drill in your school is a real-world example of a method. No matter what else is going on, when you hear the fire alarm, you stop what you are doing, and follow the directions of your teacher (usually to find the nearest exit and leave the building!). This method (fire alarm) is 'called' by the sound of the alarm.

In AgentCubes Online, we 'call' a method by 'broadcasting' to the agents. We then put the rules in a method box with the appropriate agent, which is associated with the method name.

Let's get started:

Some code is provided. Some is not. Work with a partner if you get stuck!

Step 1: Create the monitor agent and place it on the worksheet.

Step 2: Create a single rule for the Monitor: Have the monitor tell (broadcast) the person agents to PERCEIVE and then tell (broadcast) the person to ACT

Create the METHODs

Step 3.

In the person agent.

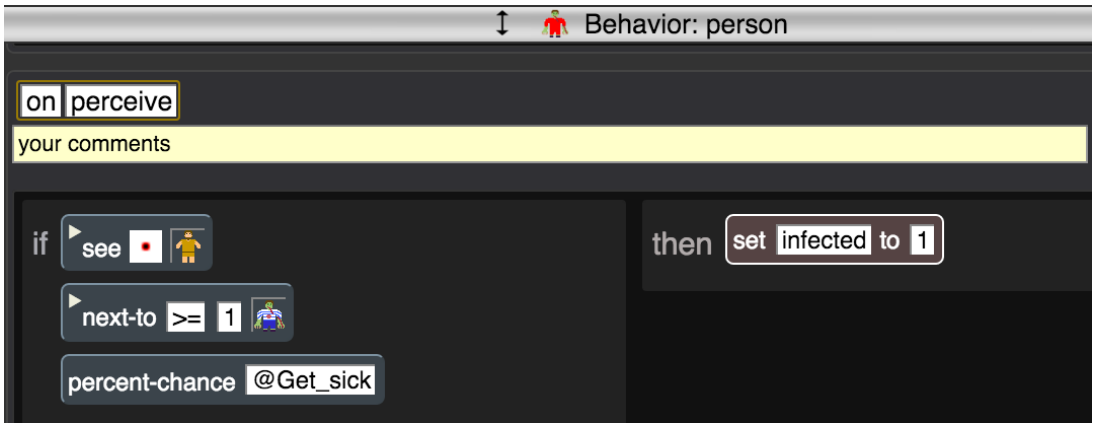
1. Use the +Method button to add 2 new methods to the person agent.
2. Click on the word "unknown" in the upper left corner of each new method.
3. Name one method "perceive" and the second method "act".
4. Click on the header for the person's while running method and then click on the minus button  on the bottom of the AgentCubes Online window to delete the while running method.

3D-Contagion (Continued)

PERCEIVE METHOD

Step 4.

Add a rule to the perceive method: if they are healthy and next to a sick person, with some probability, then set the switch for labeling them as INFECTED.



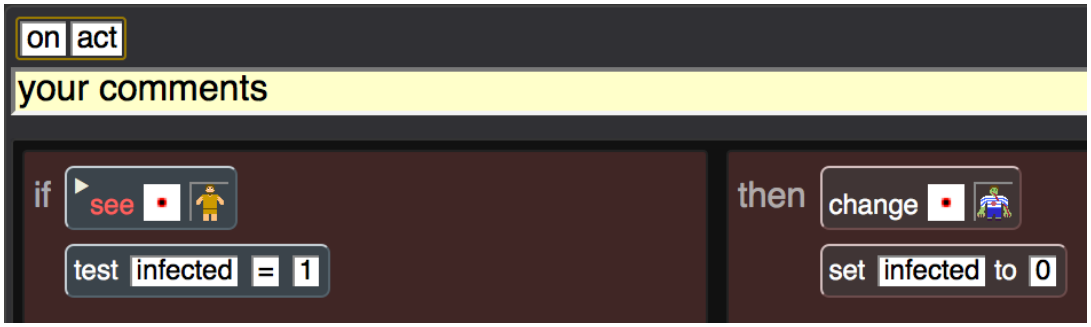
Set the agent attribute "infected" to 1 if the person will get sick. Infected will stay equal to 0 if the person does not get sick this turn.

Step 5.

Add the rule to the perceive method that makes the person get well. Set the local variable "get_well" to 1 if the person will recover. Remember that the actual change to a healthy person will happen in the act method.

ACT METHOD

Step 6. ***Add a rule*** to the act method that makes a healthy person who is infected get sick. If a person is infected, change the person to look sick and reset the infected agent attribute back to 0.



3D-Contagion (Continued)

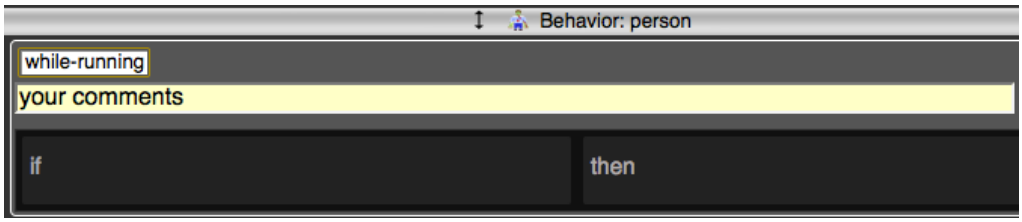
Step 7.

Add a rule If you are a healthy person, and not infected, just make move randomly on the world.

Step 8.

Add a rule If you are a sick person, and should get well, change to a healthy person and reset the get_well attribute to zero.

HINT: Double check to ensure that the person agent's 'while running' method contains no rules (it will look like it has one empty rule) so that the person agent will only do the perceive and act methods when the controller broadcasts the perceive and act messages!



In our version, we chose to have only the healthy people walk around. Is that realistic? What changes might make this more realistic?

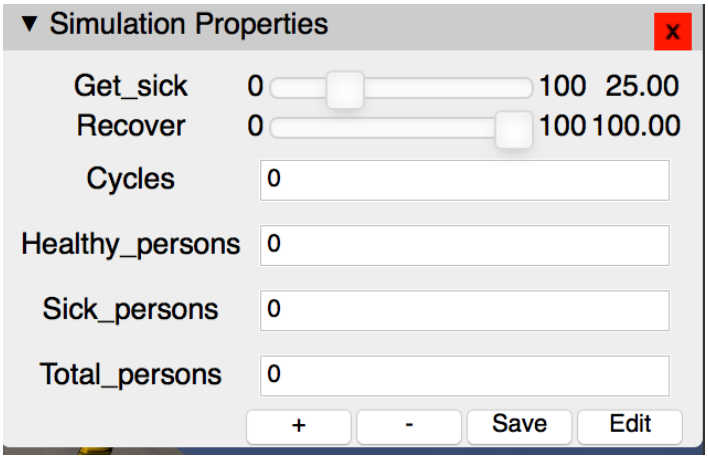
3D-Contagion

Part 2b: Student Instructions – Counting/Graphing All the Sick and Healthy Persons in the World

The World Health Organization tracks illnesses throughout the world. Graphs help us to visualize whether the disease is spreading, and whether people are recovering. In order to make a graph with the number of sick and healthy people, the monitor agent must keep track of the numbers of sick and healthy persons over time.

COUNT THE PEOPLE
Step. 1

Make new simulation properties that track the number of healthy persons, sick persons and total persons. I've chosen to name mine like this:



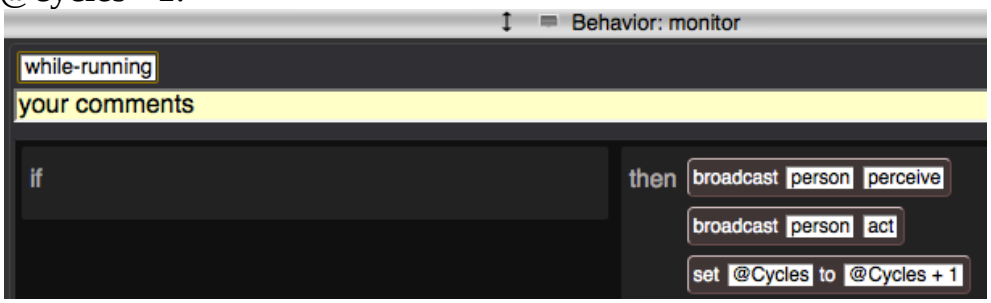
This time I didn't set them up as sliders. Why did I make this choice? Talk to a partner about why sliders are not helpful in this situation.

Make sure to click on the Save button!

Step 2.

Add the **set** actions that update the simulation properties to the rule in the monitor agent's while running method.

I want the time to increase by one each time we count. Therefore, I set the @cycles to @cycles +1.



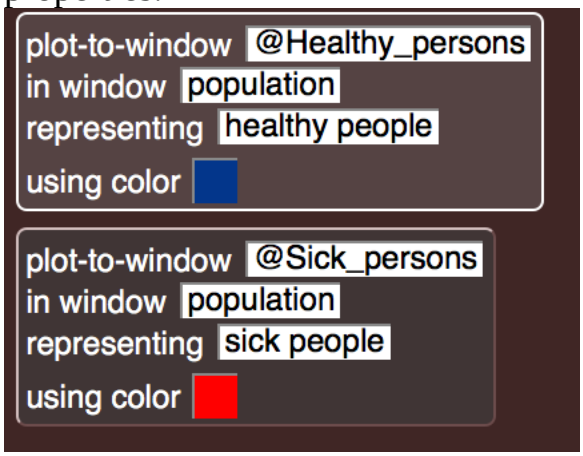
3D-Contagion (Continued)

Add some additional statements to this same rule:

1. Set @Healthy_persons to agents_with_shape(“healthy_person”)
2. Set @Sick_persons to agents_with_shape(“sick_person”)
3. Set @Total_persons to agents_of_type(“person”)

GRAPH THE PEOPLE

Add these two actions to the main rule of the monitor agent to graph the simulation properties.



Run your simulation and look at the graph. Does it look like mine? What might cause it to look different?

Part 2c: Student Instructions – Ending the Simulation

Unlike a game that can be won, simulations run until an endpoint decided by you. You might choose the end the simulation after 100 cycles. That would help you answer the question, 'What happens after 100 days of an illness.' Or you might want to run the simulation until either the disease has been eradicated (meaning that only healthy people remain) or the disease has wiped out civilization (only sick people remain).

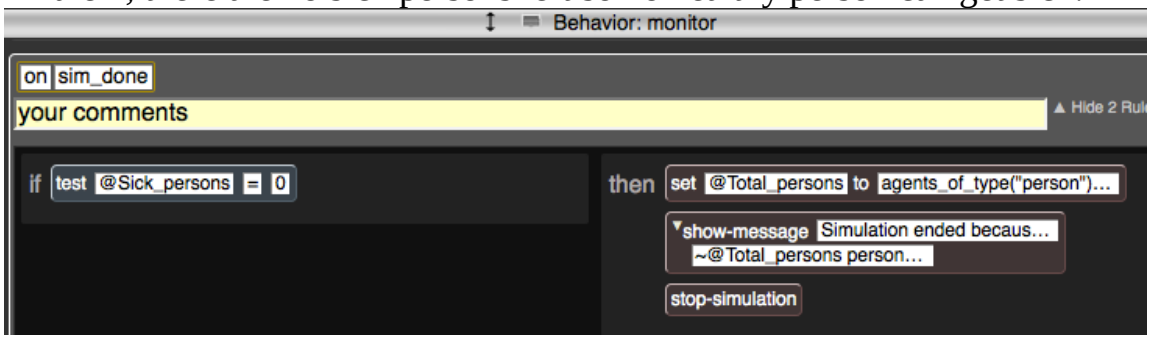
We will show you how to end the simulation once everyone is healthy. You can decide if you want to include more possibilities.

Step 1: Make the monitor check whether the simulation is finished.

Add a final action to the rule in the monitor's while running method.
This action tells the monitor to send a message to itself to check its 'sim_done' method.

Step 2: Program the sim_done method.

For each rule, show a message to the observer that explains what happened to end the simulation.
In rule 1, there are no sick persons left so no healthy person can get sick.



If you would like to include how many cycles the simulation ran, write ~@Cycles cycles and AgentCubes Online will produce a message that says 100 cycles, replace "~@Cycles" with the value of the simulation property Cycles. Make sure that there is a space immediately after "~@Cycles" so that AgentCubes Online does not get confused by an extra character.

Run the game and vary the percentages of time people get sick and recover. What numbers are most realistic for a common cold? For Ebola? Why?

Student Handout 3 – Getting Better: Adding in the length of the illness

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 that recovery takes a number of days typical of whatever illness we are modeling. For example, colds usually last about 7 to 10 days.

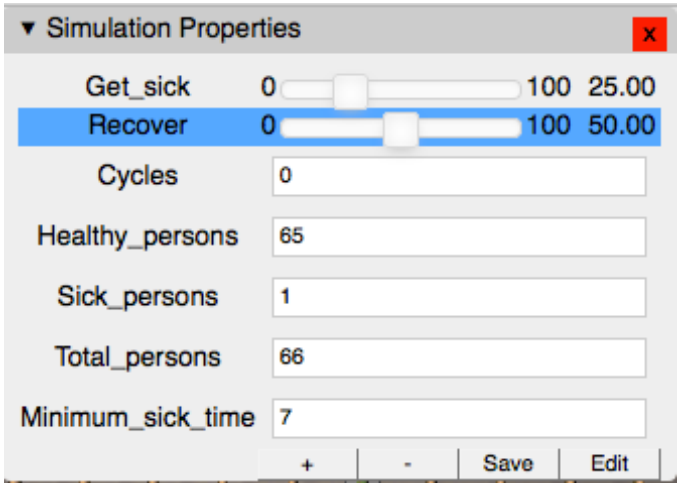
We will use a simulation property, Minimum_sick_days to designate the length of time that our person agents will be sick. We will use a variable called as SICK_CLOCK to count up how long someone has been sick. Confused? Don't be – we'll take this step by step.

Code for Modeling Length of Sickness

Step 1. Create two new simulation properties: Minimum_sick_time and Recover.

To model the common cold, set Minimum_sick_time to 7.

Edit the Recover property to be a slider with values from 0 to 100 because it stands for the probability that the person agent will recover **on any given cycle**. In this case, set Recover to 50 percent. The result will be that some persons recover after 7 days, some after 8, 9 or 10 days and a few take even longer to get well.



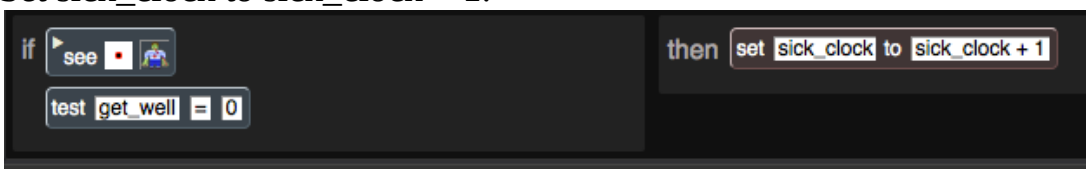
Remember to save your simulation properties!

3D-Contagion (Continued)

Step 2. The person agent's sick_clock is automatically set to 0 by AgentCubes Online.

Step 3. If the sick person is still sick, their sick_clock counts up. <u>Add this rule to the Act method.</u>

Set sick_clock to sick_clock + 1.



Step 4. <u>Edit the rule in the Perceive method</u> so that if person agent's sick_clock is greater than or equal to the <u>Minimum_sick_time</u>, the person has some chance of recovering. The simulation property <u>Recover</u> controls how likely the person is to recover on any given cycle.

Use a test condition to check whether sick_clock >= @Minimum_sick_time.

Note the "@" before the simulation property names. We need it to get the value of each simulation property.

For example, if 10 person agents have sick_clocks = Minimum_sick_time and **<u>Recover = 50</u>**, on the first tick of the simulation clock, on average, 5 will recover. So those 5 would be sick for just the minimum time of 7 days.

Of the 5 remaining sick people who could recover, on the second tick of the simulation clock, 3 might recover. Those 3 would be sick for 8 days.

Of the 2 remaining sick people, on the next click of the simulation clock, 1 might recover. So that one person would be sick for 9 days.

On the next tick of the simulation clock, the last sick person has a 50 percent chance of recovery. So if that person recovers, it would have been sick for 10 days.

3D-Contagion (Continued)

If the last person does not recover until the next tick of the simulation clock, the last person would be sick for 11 days.

This method lets us have a more realistic simulation in which the majority of sick people recover in the minimum time but a few take longer. And we can not predict exactly which people will be sick longer, just like in real life.

Run your revised simulation. What happens in your simulation? Is it what you expected?

- In what ways does this simulation reflect the real world? In what ways does the simulation not reflect what really happens?

3D-Contagion

Student Handout: Challenge 1.0
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:

When serious diseases like Ebola are modeled, not everyone gets better. Some people will die. Change your simulations to have some small percentage of people die.

First, change the PERCEIVE method by adding a rule that makes person agents die. When the sick_clock reaches the Minimum_sick_time… I have a chance to get better, I may stay sick, or I may have a chance to die. So, create one rule that checks to see if I recover with a Recover percent chance and a second rule that checks whether I die with a percent chance from a new simulation property named "Fatality". Don't forget the switches - The action for the die rule in the PERCEIVE method sets an agent attribute "die" equal to 1.

Then add a rule in the ACT method that checks for agents with a "die" attribute equal to 1. Add some actions to help the viewer see that people have died. Perhaps they will change shape? Make sure to include a short wait so that the change in shape is visible. Maybe there will be a message or a sound?

Be sure to create and graph a simulation property that tracks the number of dead people. (Something to consider: once the dead people disappear off the worksheet, they are no longer countable – how will you keep track of each death?)

Student Handout: Challenge 2.0
Modeling Immunity

Design Challenge:

In the current version of the Contagion simulation, person agents get sick over and over. In reality, people do not get the exact same virus more than once. Humans do get many illnesses with similar symptoms like colds but the viruses that cause them are different in small ways.

Instead, people develop an immunity to a virus. Immunity means that a person's immune system can identify and destroy a virus that has infected that person at some earlier time.

How could you represent immunity to an illness in this simulation of contagion which takes place over a relative short time?

You could use an agent attribute to represent immunity.

- If "immune = 1" then the person agent cannot get sick but if "immune = 0" then the person agent has a chance of becoming ill.

Think about when the person agents should not have immunity and when they should be immune to a sickness.

- When should the person agent's attribute "immune" be set? Before the person gets sick? When the person recovers?
- What is the effect of being immune? Can a person agent who is immune become sick?
- When an agent is created, should it be immune to the disease in the simulation?

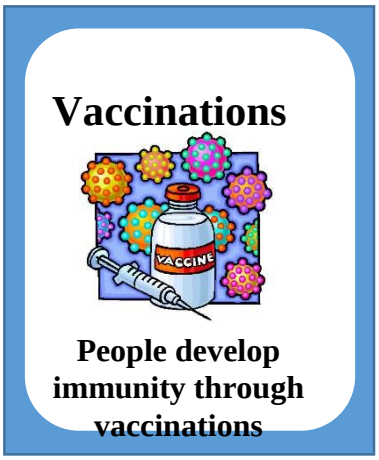
Additional Design Challenge

Humans do not develop a complete immunity to infections caused by microorganisms like the single-celled protozoans which cause malaria. In the case of malaria, humans who survive the first infection will develop a partial immunity and are less likely to die from a second infection although this partial immunity weakens over time. Malaria is treated with drugs which kill the parasite. Vaccines are in development but not available yet.

How could an illness with repeated reinfections be modeled?

Student Handout: Challenge 3.0

Vaccination



Design Challenge: Show the effect of Immunity from Vaccinations

Another way to give people immunity is to vaccinate them before they get sick.

How would you add vaccination to the simulation? Here are some ideas…

- Add a nurse or a doctor who uses a hill climbing search to find healthy person agents. Healthy person agents must diffuse their scent through the City tiles. When the nurse is next to a healthy person, then that person could become immune to the illness being modeled.
- Add a hospital and make healthy people search for it and get vaccinated when they reach it.

What happens if most people are immunized? Does the sickness spread?

"Herd immunity" means that the majority of vaccinated people become immune and even if a few people get sick, the illness cannot spread easily. If most people are immune to an illness, those who are too young or too ill to be vaccinated and those whose vaccination failed to make them immune will be protected too. You can model herd immunity by making vaccinations work most but not all of the time. Do this by adding a percent-chance condition to the vaccination rule.

How well does a vaccination need to work to provide herd immunity and stop the spread of an illness?

Student Handout: Challenge 4.0
When should I stay home?

Design Challenge: Show the effect of going out when sick

Sometimes, even when people are sick, they still go to work and to school. Many say they do that because they don't want to have to make up all that missed work. How does that affect others when sick people can move around?

Change the simulation so that some percentage of sick people walk randomly through the world. Create graphs for different values of sick people who roam. Does it make a difference? Is it okay if a small number of people do this?

What about the level of contagion? Does it make a difference if the disease is very contagious compared to ones that are not very contagious?

Additional Design Challenge

What if we sent all the sick people to the hospital? How would that affect the model?

To do this, create a hospital and have sick people use a hill climbing search to find the hospital. The hospital agent must diffuse its scent through the City tiles. Once the sick people are healthy, they can leave the hospital.