Stage 1:

Problem specification:
There are persons in a City. Persons move randomly in the City.

Agents:
- Person
- City

Construction:
- Create AgentCubes world: 'world 1'.
- Populate 'world 1' completely with City agents.
- Place one Person agent in 'world 1'.
- Create 'world 2' world.
- Populate 'world 2' completely with City agents.
- Place an array of several rows and columns of Person agents in 'world 2'.

Behavior:
- Rule in Person: 'move randomly on City'.

Run the simulation:
- Run the simulation in World 1; verify the person agent moves randomly.
- Run the simulation in World 2; verify the person agents move randomly.

Stage 2:

Problem specification:
There are persons in a City. Some persons are healthy and some are sick. Healthy persons move randomly in the City. If a healthy person is next to a sick person, there is a 50% probability that the healthy person will become sick. A sick person has a 50% probability of recovering, becoming a healthy person.

Agents:
- Person agent: create new shapes Healthy_Person and Sick_Person.
- City

Construction:
- Create ‘world 3’ world.
- Populate ‘world 3’ completely with City agents.
• Place several rows of Healthy_Person agents in ‘world 3’.
• Place one Sick_Person agent in the center of the Healthy_Person agents.

Behavior:
• Duplicate the rule in the Person agent.
• In the first rule of the Person agent, add the condition “see [me] as a Healthy_Person”; add the condition “next to ≥ 1 Sick_Person”; add the condition “% chance (50)”. Remove the ‘move randomly on City’ action; add the action “change to (Sick_Person)”.
• In the second rule of the Person agent, add the condition “see [me] as a Healthy_Person”.
• Add a third rule to the Person agent with the conditions “see [me] as a Sick_Person” and “% chance (50)”. Add the action “change to (Healthy_Person)”.

Run the simulation:
• Run the simulation in World 3.
• Change the probability of recovering to be 0. Then single-step the simulation.

Observations:
• Healthy_Person agents move randomly; Sick_Person agents do not.
• Usually with the 50% probability of recovering, eventually, there are no longer Sick_Person agents remaining.
• When the probability of recovering is set to 0, eventually there are no longer Healthy_Person agents remaining.
• When the probability of recovering is set to 0 and the simulation is single-stepped, Healthy_Person agents that are not next to a Sick_Person agent become sick in the same cycle. Why?
• It is difficult to keep track of the number of Healthy_Person and Sick_Person agents.

Conclusions:
• It would be helpful to make variables (simulation properties) the probabilities to make it easier to experiment without actually changing rules (i.e., the programming).
• It would be helpful to maintain (and even plot) the various healthy and sick person populations to more easily visualize what is happening.
• It would be helpful if the simulation would terminate when there is no remaining opportunity for agent state to change; i.e., no more Sick_Person agents remaining.
• Single-stepping the simulation indicates that the model is not accurately representing the specification; i.e., Healthy_Person agents that are not next to a Sick_Person agent get sick in the same simulation cycle.
• A programming change is necessary to make the programming model represent the specification.

Discussion: The last conclusion indicates the need for a different kind of computational thinking pattern than simple collision to determine when Person agents change characteristics. A new computational thinking concept is introduced, called “Perceive-Act”. This requires a separate agent to serve as a monitor, or controller, to direct – script – the behavior of the Person agents. The new behavior will be divided into two separate groups of rules: one group to evaluate the
current state of the agent and determine whether a change is appropriate (“Perceive”), and the other group to execute that change (“Act”).

Stage 3:

Problem specification:
There are persons in a City. Some persons are healthy and some are sick. Healthy persons move randomly in the City. If a healthy person is next to a sick person, there is a probability that the healthy person will become sick. A sick person has a probability of recovering, becoming a healthy person. The probabilities are variables (simulation properties). There is a monitor agent that keeps track of the current number of Healthy_Person and Sick_Person agents, as well as the number of simulation cycles (which can be thought of as days). It detects the end of simulation when there are no remaining Sick_Person agents. In addition, the Monitor controls the behavior of the Person agents via scripting of Perceive and Act methods in the Person agent, which replace the original While Running method.

Agents:
- Person agent: create new shapes Healthy_Person and Sick_Person.
- City
- Monitor

Simulation properties:
- @Get_Sick (0-100): probability (percent) of a healthy person becoming sick when next to a sick person. Initial setting: 50.
- @Recover (0-100): probability (percent) that a sick person will recover. Initial setting: 50.
- @Cycles: the number of simulation cycles completed.
- @Healthy_Persons: current number of Healthy_Person agents.
- @Sick_Persons: current number of Sick_Person agents.
- @Total_Persons: number of all persons.

Agent attributes:
- Person agent: infected: prepares a Healthy_Person to become sick.
- Person agent: get_well: prepares a Sick_Person to recover

Behavior:
- Person agent:
  - Split the rules in the While Running method between two new methods: Perceive and Act.
  - In the Perceive method, test whether a Healthy_Person should become sick and set its ‘infected’ attribute to 1.
In the Perceive method, test whether a Sick_Person should recover and set its ‘get_well’ attribute to 1.

In the Act method, change a Healthy_Person to a Sick_Person if its ‘infected’ attribute = 1, and reset the ‘infected’ attribute to 0.

In the Act method, change a Sick_Person to a Healthy_Person if its ‘get_well’ attribute = 1, and reset the ‘get_well’ attribute to 0.

In the Act method, a Healthy_Person that is not infected moves randomly on the City.

Monitor agent:
- Broadcast to Person agents ‘Perceive’ and ‘Act’.
- Update statistics: @Healthy_Persons, @Sick_Persons, @Cycles.
- Plot @Healthy_Persons and @Sick_Persons.
- Test simulation complete if either @Healthy_Persons or @Sick_Persons = 0.

Example project: [https://www.agentcubesonline.com/project/65890](https://www.agentcubesonline.com/project/65890)

Run the simulation:
- Run the simulation with the default values of 50% for getting sick and recovering.
- Run the simulation with @Recover = 0 and different values for @Get_Sick.

Possible Extensions

**Length of individual sickness:** Add capability to specify minimum length of time a sick person remains sick.

In this extension, as a Healthy_Person becomes infected, a “clock” would be set to indicate how long the sickness lasts before the possibility of recovery. One way to do this is to add a simulation property that specifies this time (in simulation cycles). Then, when an infection begins, a ‘sick_clock’ agent attribute would be set. The clock could be either a count-down or count-up clock. There are advantages to each. A count-up clock allows the user to view an agent’s attribute to see how long the sickness has lasted. Note that after the clock has reached the specified time, the probability of recovery would take effect; thus, the person could remain sick for many cycles longer. A count-down clock would simply show how many cycles remain before the person has a chance to recover. In each simulation cycle, the clock would be incremented or decremented until the specified limit has been reached, after which the rule containing the probability to recover would be examined.

Example project: [https://www.agentcubesonline.com/project/65892](https://www.agentcubesonline.com/project/65892)

**Fatality:** Add the possibility that a patient dies instead of recovering.
In this extension, a Sick_Person might die instead of recovering. There are multiple possible implementations. One possibility includes the implementation of a specified minimum length of sickness, after which a test for recovery is made; if recovery is successful, the person becomes a Healthy_Person. Otherwise, the person dies. There are multiple possible implementations of the latter occurrence. One possibility is to create an additional Dead_Person shape and change the Sick_Person into that shape. Another possibility is to erase the Sick_Person agent from the world. In the former case, it is possible for the Monitor to maintain a statistic for the number of Dead_Person agents; in the latter case, the Person agent must maintain this statistic, because the agent will be erased. In either case, the statistic is a new simulation property.

Example project: https://www.agentcubesonline.com/project/65891

**Immunity**: Add the capability for temporary (short-term) or permanent immunity following recovery from infection, or the capability for “natural” immunity.

Immunity is an individual agent characteristic, which means that it would be managed as an agent attribute. However, it would be beneficial to have a simulation property – a “switch” – that allows immunity to be operative. This enables comparing simulation runs with all other simulation properties constant except for immunity being operative or not. Additional nuances of immunity can be considered, such as whether recovery from sickness automatically confers immunity, as well as the length of immunity – that is, a specified number of simulation cycles or permanent.

If immunity is implemented, one natural question arises as to the degree in which re-infection occurs when immunity is not operative. This implies the need for a simulation property that tracks the number of agents that experience repeated infections. In addition, the latter implication is that the Person agent requires an agent attribute to document whether that specific agent has been infected more than once.

Example project: https://www.agentcubesonline.com/project/65891

**Allow sick people to move**: Add the capability to specify whether a Sick_Person can move.

This extension would provide the option (using a simulation property) to allow Sick_Person agents to move. The original implementation only allows Healthy_Person agents to move. Movement of sick people has a significant influence on the spread of contagion in the world today, so this could be an interesting comparison for students to examine. A further refinement of this feature would be to allow a different rate of movement for Sick_Person agents, such as every other simulation cycle, to show that a sick person is less mobile.

Example project: https://www.agentcubesonline.com/project/66006
**Add a hospital:** Add the capability for Sick_Person agents to seek medical treatment after a specified number of simulation cycles have elapsed for that agent’s sickness.

As with other enhancements, a simulation property could be added to specify the nominal number of simulation cycles of a sickness, after which a Sick_Person would seek treatment. A Hospital agent would be added, and diffusion and hill-climbing would be implemented in the City and Person agents, respectively. Of course, the Sick_Person could recover while en-route to the Hospital, in which case, the hill-climbing would terminate.

**Add a doctor:** Add a Doctor agent that would seek Sick_Person agents and treat them.

This feature would require adding an attribute to the Sick_Person agent (upon becoming infected) that would permit diffusion by the City, so that the Doctor agent could then seek Sick_Person agents via hill-climbing. If a Sick_Person is next to a Doctor agent, then a recovery would occur, resetting the attribute so that the Doctor agent would seek another Sick_Person.

**“Real life” comparison:** Students can do research to learn about major factors that contributed to actual epidemic or pandemic outbreaks to determine what capabilities to add to the model. For example, age is a major factor in some diseases (e.g., childhood diseases). Sanitation is a major factor in many outbreaks. Some diseases are spread by animal contact, such as mosquitoes, fleas, and ticks. How would these factors be implemented in a model?

**Generalization of the model:** The contagion model can be likened to other physical models, such as crystallization or accretion. This can be visualized in the Stage 3 model by setting the probability of recovery to zero and experimenting with different values for the probability of getting sick. The latter variable then determines the overall shape of a crystal forming from a solution, for example. The contagion model also is similar in concept to the spread of a rumor. In this case, rather than a person being healthy or sick, the person is either “informed” or not with respect to a rumor, and is either a gossip or not in terms of spreading the rumor.

**Example project:** [https://www.agentcubesonline.com/project/66004](https://www.agentcubesonline.com/project/66004)
Appendix

**Length of individual sickness:** Add capability to specify minimum length of time a sick person remains sick.

In this extension, as a Healthy_Person becomes infected, a “clock” would be set to indicate how long the sickness lasts before the possibility of recovery. One way to do this is to add a simulation property that specifies this time (in simulation cycles). Then, when an infection begins, a ‘sick_clock’ agent attribute would be set. The clock could be either a count-down or count-up clock. There are advantages to each. A count-up clock allows the user to view an agent’s attribute to see how long the sickness has lasted. Note that after the clock has reached the specified time, the probability of recovery would take effect; thus, the person could remain sick for many cycles longer. A count-down clock would simply show how many cycles remain before the person has a chance to recover. In each simulation cycle, the clock would be incremented or decremented until the specified limit has been reached, after which the rule containing the probability to recover would be examined.

**Example project:** [https://www.agentcubesonline.com/project/65892](https://www.agentcubesonline.com/project/65892)

**Details:**

- Add simulation property @Min_Sick: minimum number of simulation cycles a sick person remains sick before a decision to recover is made. Initially, leave its initial value to 2 and save the simulation properties. Later, explore the effect of increasing the minimum sickness time.
- Add condition to rule in “Perceive” method that tests for recovery: test sick_clock ≥ @Min_Sick. Thus, only when the sick_clock has reached the minimum number of simulation cycles will there be a possibility of recovery.
- Add action to Person agent in “Act” method to set agent attribute ‘sick_clock’ to 0 when a Healthy_Person changes to a Sick_Person.
- Add rule to Person agent in “Act” method for a Sick_Person that if NOT get_well=1, increment sick_clock.

**Fatality:** Add the possibility that a patient dies instead of recovering.

In this extension, a Sick_Person might die instead of recovering. There are multiple possible implementations. One possibility includes the implementation of a specified minimum length of sickness, after which a test for recovery is made; if recovery is successful, the person becomes a Healthy_Person. Otherwise, the person dies. There are multiple possible implementations of the latter occurrence. One possibility is to create an additional Dead_Person shape and change the Sick_Person into that shape. Another possibility is to erase the Sick_Person agent from the world. In the former case, it is possible for the Monitor to maintain a statistic for the number of Dead_Person agents; in the latter case, the Person agent must maintain this statistic, because the agent will be erased. In either case, the statistic is a new simulation property.

**Example project:** [https://www.agentcubesonline.com/project/65891](https://www.agentcubesonline.com/project/65891)
Details:
- This project builds on the project implementing minimum sick time.
- Add simulation property “@Fatality” as a switch. When set to 1, fatality is implemented, so that Sick_Persons die if they do not recover; when set to 0, fatality is not implemented, so the Sick_Person does not die, but eventually recovers.
- Add simulation property “@Deaths” to maintain a count of the number of Sick_Person agents that do not recover.
- Add a condition in the Person agent “Perceive” method that tests for the sick_clock reaching the minimum sick time (@Min_Sick) to test for @Fatality = 0. Thus, if the fatality switch is not on, recovery occurs based on the probability for recovery (@recover).
- Add a rule in the Person agent “Perceive” method for a Sick_Person that, when the sick_clock reaches the minimum sick time and if the fatality switch is on (@Fatality=1), a message is sent to a new method (“Test_Fatality”) to determine whether the Sick_Person will recover or die. Note that this rule does not have the %-chance condition; rather, once the minimum sick time is reached, then the person will either recover or die.
- Add the new “Test_Fatality” method to the Person agent with two rules. The first rule has the %-chance condition for recovery based on the @Recover simulation property. If that condition is true, then the Get_Well attribute is set to 1. Otherwise, unconditionally, a new attribute “Die” is set to 1.
- Add a condition to the rule in the “Act” method in the Person agent which tests “NOT get well = 1” to test “NOT die = 1”, meaning that if the Person agent is not ready to get well, nor is it set to die, then the sick_clock will continue to be incremented.
- Add a new rule in Person agent “Act” method that tests if a Sick_Person has the attribute “die = 1”, to increment the @Deaths simulation property and erase the agent. NOTE: this implementation removes the agent from the world. An alternative implementation would be to create an additional Dead_Person shape in the Person agent and change the Sick_Person to the Dead_Person shape. This has significant consequences on the simulation, as the Dead_Person agents act like a barrier to mobility of other agents, which affects infection rates.
- Modify “Show-message” actions in the Monitor agent to report the number of deaths that occur when the simulation ends.
- Add a Plot-to-window action in the Monitor’s Test_Complete method to plot the number of deaths (@deaths simulation property).
- Optional: The following additions provide a direct capability of testing the implementation of fatality code By setting the @Recover value to zero, all Sick_persons will die. This means that the @Deaths simulation property should equal the @Total_Infections simulation property described below.
  - Add a @Total_Infections simulation property to track the absolute total number of infections. Set the initial value to 1 to reflect the initial Sick_Person in the world, and save the simulation properties.
  - In the Act method of the Person agent, add a Set action to increment @Total_Infections to the rule in which the agent attribute “Infections” is incremented.
In the Show Message dialog of the “Test_Complete” method in the Monitor agent, add text to report the total number of infections when the simulation is complete. Also add text to report the total number of infections.

Immunity: Add the capability for temporary (short-term) or permanent immunity following recovery from infection, or the capability for “natural” immunity.

Immunity is an individual agent characteristic, which means that it would be managed as an agent attribute. However, it would be beneficial to have a simulation property – a “switch” – that allows immunity to be operative. This enables comparing simulation runs with all other simulation properties constant except for immunity being operative or not. Additional nuances of immunity can be considered, such as whether recovery from sickness automatically confers immunity, as well as the length of immunity – that is, a specified number of simulation cycles or permanent.

If immunity is implemented, one natural question arises as to the degree in which re-infection occurs when immunity is not operative. This implies the need for a simulation property that tracks the number of agents that experience repeated infections. In addition, the latter implication is that the Person agent requires an agent attribute to document whether that specific agent has been infected more than once.

Example project: https://www.agentcubesonline.com/project/65891

Details:

• This project builds on the project implementing minimum sick time.
• Add new simulation property: @Immunity. This is a switch, like the @fatality switch. If @Immunity=1, then immunity occurs when a Sick_Person recovers. If @Immunity=0, then there is no immunity as a result of recovery from sickness.
• In the “Perceive” method of Person agent, in the rule with the %-chance for getting sick, add the additional condition “test @Immunity = 0” in order for a Healthy_Person to become sick (assuming the %-chance condition is satisfied). This means that if immunity is not in effect, the simulation operates as if the feature is not implemented.
• In the “Perceive” method of Person agent, duplicate the previously described rule. Change the test regarding immunity to “test @Immunity = 1”, meaning that immunity is in effect for the simulation. Add an additional condition: “test Infections = 0”. These combined conditions mean that if immunity is in effect for the simulation then a Healthy_Person can only become sick once; if the Infections attribute is not zero then the agent has already been sick at least once and cannot become re-infected.
• In the “Act” method, in the rule in which a Healthy_Person changes to a Sick_Person, add the additional action “set Infections = Infections + 1”. The ‘Infections’ agent attribute records the number of infections that a particular agent has undergone. It is examined in the “Perceive” method to determine whether a Healthy_Person can become sick if immunity is in effect.
• In the world (World 3), open the agent attributes window by first selecting the single Sick_Person agent in the world. Using the “+” button, create the “Infections” attribute,
then edit it and set it to 1. Finally, save the world. Note that it is necessary to save the world after adding/editing the agent attribute because agent attributes are stored with the agent information in the world. This shows that the sick person has been infected once; if immunity is in effect, the person will not be re-infected.

**Allow sick people to move:** Add the capability to specify whether a Sick_Person can move.

This extension would provide the option (using a simulation property) to allow Sick_Person agents to move. The original implementation only allows Healthy_Person agents to move. Movement of sick people has a significant influence on the spread of contagion in the world today, so this could be an interesting comparison for students to examine. A further refinement of this feature would be to allow a different rate of movement for Sick_Person agents, such as every other simulation cycle, to show that a sick person is less mobile.

**Example project:** [https://www.agentcubesonline.com/project/66006](https://www.agentcubesonline.com/project/66006)

**Details:**
- This project builds on the project implementing minimum sick time.
- Add new simulation property: @Sick_Move. This is a switch to enable/disable Sick_Person agents from moving. When @Sick_Move = 1, Sick_Person agents can move. Otherwise, they cannot.
- Add a message action to the rule in the Person agent “Act” method that tests for “NOT get_well = 1” (i.e., the person agent is not ready to recover). The additional action is “message (. ) Test_Sick_Move”, where “Test_Sick_Move” is a new method.
- Add the “Test_Sick_Move” method to the Person agent.
- Add the rule to the new “Test_Sick_Move” method: If ‘@Sick_Move = 1’ then ‘move random on City’. Thus, if the @Sick_Move switch is set (i.e., 1), a Sick_Person agent will move randomly in the same way as a Healthy_Person agent.

**Add a hospital:** Add the capability for Sick_Person agents to seek medical treatment after a specified number of simulation cycles have elapsed for that agent’s sickness.

As with other enhancements, a simulation property could be added to specify the nominal number of simulation cycles of a sickness, after which a Sick_Person would seek treatment. A Hospital agent would be added, and diffusion and hill-climbing would be implemented in the City and Person agents, respectively. Of course, the Sick_Person could recover while en-route to the Hospital, in which case, the hill-climbing would terminate.

**Add a doctor:** Add a Doctor agent that would seek Sick_Person agents and treat them.

This feature would require adding an attribute to the Sick_Person agent (upon becoming infected) that would permit diffusion by the City, so that the Doctor agent could then seek
Sick_Person agents via hill-climbing. If a Sick_Person is next to a Doctor agent, then a recovery would occur, resetting the attribute so that the Doctor agent would seek another Sick_Person.

**“Real life” comparison:** Students can do research to learn about major factors that contributed to actual epidemic or pandemic outbreaks to determine what capabilities to add to the model. For example, age is a major factor in some diseases (e.g., childhood diseases). Sanitation is a major factor in many outbreaks. Some diseases are spread by animal contact, such as mosquitoes, fleas, and ticks. How would these factors be implemented in a model?

**Generalization of the model:** The contagion model can be likened to other physical models, such as crystallization or accretion. This can be visualized in the Stage 3 model by setting the probability of recovery to zero and experimenting with different values for the probability of getting sick. The latter variable then determines the overall shape of a crystal forming from a solution, for example. The contagion model also is similar in concept to the spread of a rumor. In this case, rather than a person being healthy or sick, the person is either “informed” or not with respect to a rumor, and is either a gossip or not in terms of spreading the rumor.

**Example project:** [https://www.agentcubesonline.com/project/66004](https://www.agentcubesonline.com/project/66004)