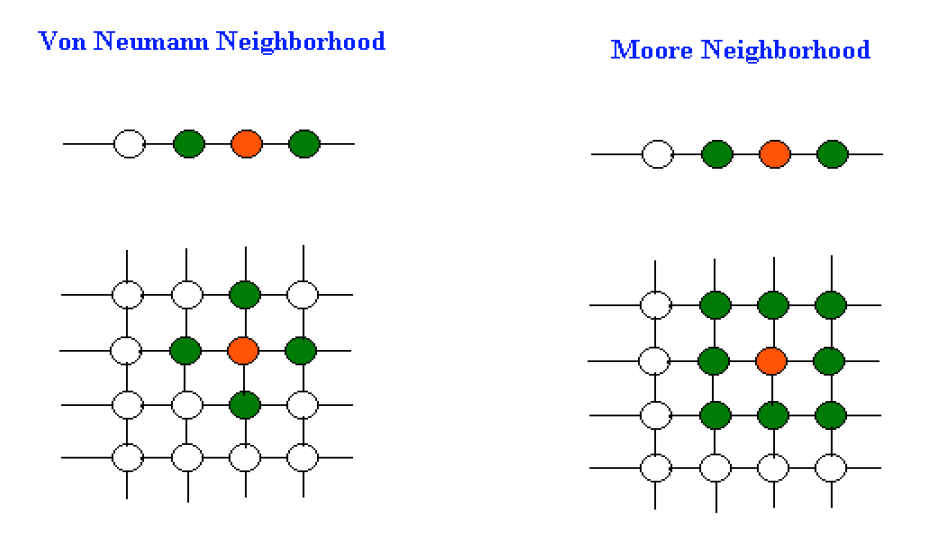
**Description of cellular automata**

“A cellular automaton is an array of identically programmed automata, or cells, which interact with one another in a neighborhood and have definite state.” This array can be one-dimensional or two-dimensional, which in a programming language like Python would simply be a list or list of lists. Each index would represent or contain a cell that can only affect other automata within its neighborhood based on its state. Each cell must operate exactly the same when placed in identical contexts; cell behavior must be orthogonal. A cell’s neighborhood – the surrounding cells allowed to affect the state of it – can either be from the Von Neumann or Moore strategy, where in the former the allowed cells are on the north, south, east, and west, and in the latter, all surrounding cells are allowed. Lastly, each cell can only possess a definite number of states.



The SLIR (**S**usceptible, **L**atent, **I**nfectious, **R**ecovered) disease model uses four differential equations to model the change in population state and number of individuals present in each state throughout the spread of any disease. In brief, the four equations are

where each equation represents the change in state population per unit time. The SLIR model works well for diseases that do not immediately spread from an infectious person once they become infected, hence the **L**atent/incubation state. Using this model, **S**usceptible individuals can only be infected by the **I**nfectious population. Once a **S**usceptible individual has been infected, they enter the **L**atent state for the duration of the diseases latent period. Once they have met that requirement, they automatically transition into the **I**nfectious stage for the duration of the infectious period. Lastly, they automatically transition to the **R**emoved stage when they meet the infectious period limit. (Note: there is no distinction between death and recovery in the **R**emoved stage.)

This simulation uses the SLIR model of a disease with the cellular automata data structure to model disease spread throughout a student population. It provides a visual to track the progress of a disease in this population by appending all end-of-day information to form a .gif image. For the duration of this discussion, the term ‘Individual’ is a moveable object that can occupy a cell in the cellular automaton. Also, ‘Individual’ refers to the class object in the simulation; ‘individual’ is the general term for a unit in disease spread discussion. The cells do not possess a state; the Individual does. Secondly, the term ‘grid’ will replace ‘cellular automaton’ when discussing the entire simulation automaton (not a cell).

The simulation provides the user with plenty of disease and population-defining parameters like defining the age distribution of the population or whether infected individuals display symptoms or take measures to prevent the spread from themselves to others (e.g. quarantining or wearing a mask). The customization parameters offered are the size of the cellular-automaton/grid (an N-by-N-sized 2D list of lists), the number of initially infected individuals, the population size, the age distribution of the population, the disease mortality rates of each age group, the minimum and maximum number of days an infected individual could remain in the latent period, the minimum and maximum number days an infectious individual could remain in the infectious period, the maximum number of exposure points an individual must obtain to become infected, the ratio of mask wearers, quarantining individuals, and symptomatic individuals in the population, and the simulation neighborhood strategy (von Neumann or Moore).

Each Individual has the following characteristics in the simulation: a state of health, a unique identifier, an age, their age’s mortality rate, their row and column position in the grid, the row and column position they will move towards, a counter tracking the number of days they’ve spent in a certain state, the duration in days of their latent and infectious periods, the exposure points count, and the fact that they wear a mask, quarantine, or are symptomatic when infectious. Now we will explain a certain number of these for further clarification.

According to the SLIR model, an Individual’s state of health can either be healthy and susceptible, infected and in the latent stage, infected and infectious, or recovered, dead or alive. Therefore, Individuals can possess only those four states, represented as an integer (0 for susceptible, 1 for latent, 2 for infectious, and 3 for recovered). Although not strictly a state, an individual can be immune to a disease, either before or after an infection. This feature will be implemented in this program shortly.

Age has a significant influence on an individual’s recovery. Generally, the older someone is, the more likely they will exit the **I**nfectious stage by death. The simulation includes this by associating the age of an Individual with their mortality ratio. These can be taken from real data if a real disease is being considered.

An Individual has the ability to roam the simulation grid until they reach a targeted location, determined at instantiation. Every day, the Individual will move one cell towards their target location until they reach it, at which point, they are assigned a new target location and the process repeats. Multiple Individuals can occupy a single cell, so no collisions are possible nor evasion maneuvers necessary between two Individuals who cross paths. This mimics real-life behavior, as when two people cross paths, the steps they take to avoid colliding are small enough to be considered negligible, yet not nearly enough to eliminate any change of catching a disease from them. According to the CDC, for COVID-19, the recommended distance to remain away from other individuals is around six feet. Therefore, Individuals traveling to their target position, unless they are quarantined, do not respect that six-foot distance when crossing paths.

In the real world, the duration of an individual’s latent and infectious period follows a normal distribution. However, this simulation employs a uniform distribution between values determined by the user. Other distribution alternatives can be added later.

During the SARS-CoV-2 pandemic, the CDC defined close contact as maintaining less than a six-foot distance for 15 minutes or more with an infectious individual. The predominant unit of time in this simulation is days, so to account for the mere minutes it takes to become likely to become infected, exposure points track the number of close contacts an Individual has encountered. The number of exposure points an Individual accumulates is directly proportional to the number of infectious Individuals who are in their neighborhood (defined as either Von Neumann or Moore). Once an Individual has reached the user-defined threshold of exposure points, they transition to the latent stage. This method is similar to real world infections as a person has a higher likelihood of getting sick the as they encounter more sick people at once. Exposure points carry over to subsequent days (and this particular feature might be removed as it doesn’t resemble real life behavior).

Individuals who are symptomatic display symptoms that they can notice, as opposed to individuals who are asymptomatic. In the former case, if diligence prevails, they can isolate themselves and take other preventative measures to keep from spreading the disease. In the latter, the person will likely continue their lives unknowingly spreading the disease they don’t know they have. All three of these kinds of individuals are represented in this simulation. Users can gather real world data to input for these parameters by taking a survey, counting the number of people who wear masks or isolate when sick and finding the ratio of symptomatic cases in all known cases of a disease. For example, the CDC reported 40% of COVID-19 cases are asymptomatic; during one survey taken at a Walmart, ~80% of shoppers wore a mask properly at the time they walked into the store; and 50% of people who get sick will stay at home if it’s not severe (random number I made up for this example, but I’m betting it’s lower than that).

Simulation progress is visualized once per simulated day. Each individual’s state of health is mapped to a specifically colored square .png image which is pasted to a large, black NxN-sized .png canvas – where N is the number of rows/columns in the simulation grid - in their location in the simulation grid. If multiple individuals are located in the same spot, their images are shrunk to fit all of them in an MxM grid – where M is its number of rows/columns so that no more than M2 individuals can fit. M is calculated by taking the ceiling of the square root of the number of individuals possessing a spot. The visualization processing takes a super-majority of the simulation execution time. (RUN A TEST WITH A LOT OF PEOPLE TO PROVE YOUR POINT).

Most students and faculty at MSU were mandated to wear masks when a six-foot distance between each other wasn’t possible and while being indoors. Those who do get sick are told to stay home. For this simulation, we assume 99% of Individuals are mask wearers and 80% of those who are sick will isolate until they feel better. Individuals who isolate are still left in the simulation grid, however, in `checkNeighbors` they can’t affect any susceptible Individuals in their neighborhood. As for mask wearers, all exposure points infectious Individuals’ can deal to their susceptible neighbors are cut in half.

The project was developed with an agile software engineering methodology. To add, the cellular automata project was a continuation of a project given in the Computational Epidemiology class held at Midwestern State University. Therefore, the project had already met the requirements from the course and continued starting with code refactoring. Afterwards, new features were added incrementally and tested as they went along.

**The Individual class**

Every person/being that has the potential of infection is represented as an Individual object. Each Individual possesses attributes such as their state of health, the length of time spent in their current state of health, and factors pertaining to the disease like duration of immunity retention, infectious period length, and whether they display symptoms when infectious. The Individual occupies a spot in the simulation grid and stores that as an attribute as a row-column tuple. The Individual will also travel around the simulation grid toward a desired location (which right now is determined at random). (In the future, certain classes of Individuals will travel toward the same locations to simulate a group of people all traveling the same locale.) Of course, once they reach their location, a new one is generated and they change course for it. Individuals continue this cycle of traveling without outside interference barring those tied to the disease. An Individual, once they have entered the recovered stage, will either continue to move about the simulation grid – meaning they’re alive – or not – meaning, they have died. **Should I remove them from the grid so they don’t clog up the visualization? Should I add a counter to the main program that tracks number of dead?**

**The Visualizer class**

If the parameter is set in `params.json`, the simulation will generate a .gif compilation of each day’s results, comprised of all Individuals represented in their respective positions in the simulation grid with a colored tile reflecting their state of health. Each Individual’s state of health is mapped to a specifically colored square .png image which is pasted to a large, black NxN-sized .png canvas – where N is the number of rows/columns in the simulation grid - in their location in the simulation grid. If multiple individuals are located in the same spot, their images are shrunk to fit all of them in an MxM grid – where M is its number of rows/columns so that no more than M2 individuals can fit. M is calculated by taking the ceiling of the square root of the number of individuals possessing a spot. The visualization processing takes a super-majority of the simulation execution time. (RUN A TEST WITH A LOT OF PEOPLE TO PROVE YOUR POINT). **Should I remove the dead from the visualization? Should I add counters to the bottom right of the .gif that should number of sus, latent, infectious, and recovered?**

Incorporated disease models

SLIRS

Similar to the SLIR model, the SLIRS model accounts for the possibility of reinfection in recovered individuals. This model fits diseases like the common cold or influenza, which produce different viral strains so frequently reinfection is common (baseless claim until I put some references here…). To be more contemporary, the recently discovered SARS-CoV-2 virus has the possibility to reinfect previously sick individuals in around 3 months (so says the lady at the blood institute I donated at. She said the antibodies last 3 months, but I’m sure I can find a more ‘reasonable’ source on the CDC website soon.) This simulation accounts for reinfection if the user desires by choosing a number at random between two user-defined limits for the number of days an Individual will remain “immune,” - the duration of the recovered stage. **Are most diseases modeled by the SLIRS model endemic?** **What diseases these days are SLIRS?**