Programming project: Disease

Anni Penttilä, 1020539 Bioinformation technology, 2nd year

CS-A1121 Spring 2023 22.2.2023

General description & difficulty level

The program will describe how a disease spreads through a population in an abstract space. In the population the individuals are either susceptible to the disease, spreading it, recovered from it and thus gained immunity or deceased and consequently removed from the simulation. Spreaders will infect susceptible people with a certain probability if they have been near by each other. After catching the disease, a susceptible individual turns into a spreader. The spreader either recovers or will be deceased after a certain time period. The simulation ends when everyone has either catch the disease or when there are no more spreaders left.

The simulation is set up with certain parameters, part of them determined by the user. A user interface will be applied for this purpose. Parameters given by the user such as a certain number of characters and a certain number of spreaders, and other predetermined parameters such as the probability of catching the disease, and the time spent being a spreader will affect how the simulation will proceed.

Giving each character also randomly generated age group or letting the user place a certain number of characters in each age group would increase the uniqueness of the project. Another feature to consider could be an additional character type, e.g., a doctor or a nurse, that could alleviate suffering and shorten the recovery time with appropriate treatment.

The initial idea is to implement the project at the medium difficulty level. If the project seems to make progress and proceeds smoothly, I might also consider the more demanding difficulty level.

Use case description and draft of the user interface

The program communicates with the user via console user interface since the wanted parameters are numerical and the interaction with the user is simple to implement this way. The user may set several parameters to preferred values. A GUI is also a possible alternative. If it was the chosen method, there would be three or more windows to plug in the wanted parameter values. The user sets the values for

- the number of characters
- the initial number of spreaders
- the surface area where the characters can move.

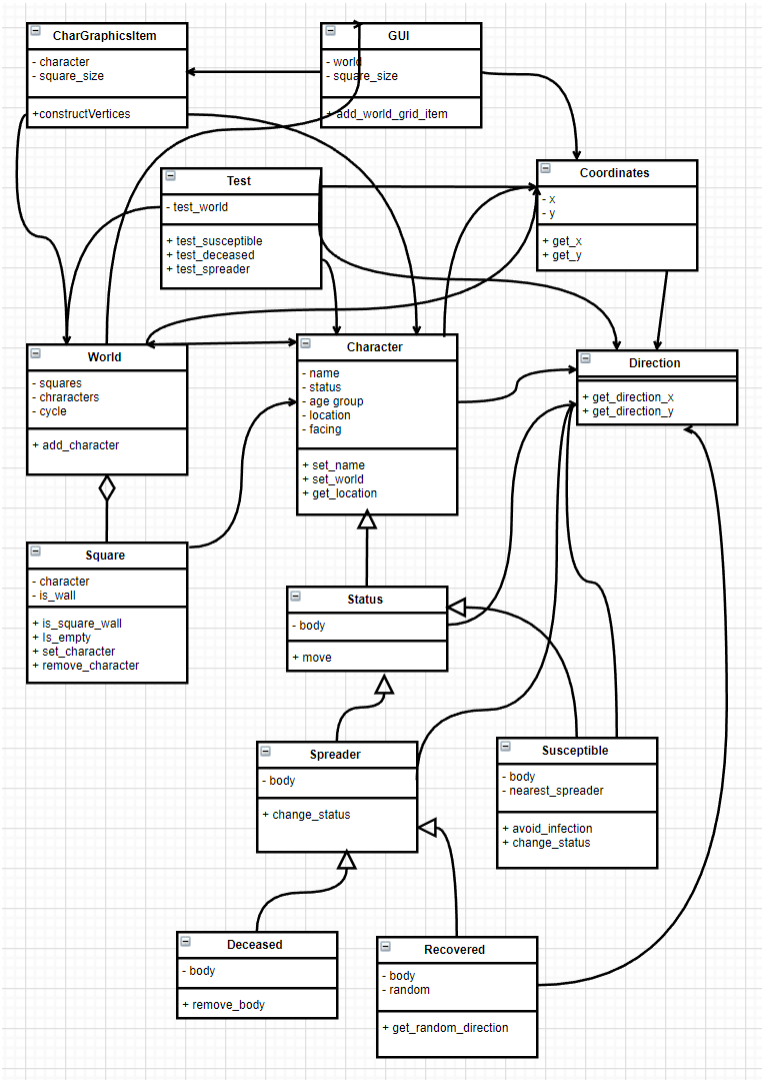
Some features of the simulation world and character AIs are initialized based on the user input. After the parameter set up the simulation begins and runs until one of the conditions for termination is met. Finally, the program prints data about the simulation; how many caught the disease / died from it, number of cases and so on.

Program's structure plan

A rough class diagram for the program is shown below. Some of the relations between classes might not be accurate but it describes the structure and the main idea behind the program. The different character types (4 to 5 classes) inherit the character AI class (Status). Each character has its own AI which builds on the methods of the Status class. Each character moves in its own specific way.

The simulation world consists of a two-dimensional grid, in which every square is either empty, a wall or contains a character. The characters can only move into empty squares. The location of each character is a feature of the Coordinates class. The Direction class aids with determining the direction of the movement during one turn or cycle in the simulation.

If the graphical user interface is applied to the program, the GUI and CharGraphicsItem classes will be implemented. The program will be tested with the Test class. Its methods test the functionality of each character type.



Data structures

Most of the data in this program will be saved just as class parameters which can be used between various classes based on the connections shown in the class diagram. An important data structure for handling the existing data will be lists that can be read and modified between the classes. For example, in the simulation world, the characters / individuals will be stored in an object list. An access to a certain class parameter will be gained calling the specific object methods. Lists are a handy way of storing data in this program, since they are a useful yet simple tool in OOP. When dealing with the locations of the characters, coordinates could be stored in a two-dimensional list, in which each element would consist of two integers (coordinate pairs).

Files and file formats

This program doesn't require loading data from files, but storing the recorded data would be sensible to save into a text file. One requirement for the harder difficulty level was that the program would write an external file where various data about simulations would be stored. The data would be presented in a table format so that the data from several simulations could be comparable and easy to understand.

Algorithms

In this program the mathematical computation regards the movement of the characters and the probabilities of catching the disease and recovering from it. The characters have their own AI, which doesn't base its decisions on randomness unless the individual has recovered from the disease.

The world where the characters move is a two-dimensional area, a grid formed by squares. During one turn in the simulation, the characters will take one step towards the preferred direction. If the square is occupied, the individual tries to find the nearest empty square. If all possible squares are occupied, the character stays in its current position. One step can be taken into the neighboring squares, also diagonally. Thus, in one turn, if all the surrounding squares are empty, and individual has eight possible directions to choose.

Below is described some algorithms behind the character AI's.

Susceptible: Susceptible individuals will try to move away from the nearest spreader.

Spreader: The area around the spreader is divided into two infectious zones. All the squares that are one step away from the spreader form the high infectious zone. And all the surrounding 16 squares around the high infectious zone (two steps away from the location of the spreader) form the low infectious zone. The probabilities of catching the disease vary when the infectious zone changes.

Spreaders will try to take a step towards the nearest nurse, whose purpose is to cure the infected individuals. After a certain time period has passed a spreader will either recover or is deceased. The chance of survival increases if the spreader encounters a nurse (enters a square one step away from the nurse).

Nurse: Nurses try to approach infected individuals with a certain logic. A nurse chooses the individual based on its location and the time period the spreader has been infected. The longer the spreader has been infected, the closer is the possible death. Thus, the nurse prefers individuals that have been infected the longest time because this way the total death rate could probably be minimized. Consequently, the direction the nurse takes is chosen by the following formula

individual_to_cure = distance - time_being_infected.

Of all the spreaders, the individual with the lowest score will be chosen and the nurse starts to approach the concerned individual. If the nurse were to prioritize the most recently infected individuals, the helping operation could more likely be terminated because new infections are likely to happen near the nurse. This could lead to an unintended zigzag movement.

Recovered: Since recovered individuals are immune to the disease, they don't need to avoid any individuals. The directions of their steps are just based on randomness.

Deceased: Deceased individuals will be removed from the simulation.

The distance between two characters will be calculated with the following formula for distance between two points. The formula for squared distance could also be implemented for this purpose. The coordinates of each character are known and thus we get

D = sqrt ((x2 - x1)^2 + (y2 – y1)^2).

Testing plan

There are many occasions that could terminate the simulation if the program is malfunctioning. Some essentials things this program should be able to do are storing the data correctly, writing files and making sure that the UI works as intended and displays the right information to the user so that the user experience is convenient. These functions could be tested by comparing expected and actual values and asserting equality. Also testing the validity of the input data is important when it comes to testing the functionality of the UI.

Since the fundamental purpose of the program is to simulate the spread of a virus in a population, the Coordinates and Direction classes should work properly. It is important to test whether all the different character types move according to the wanted logic. The probability of many characters trying to move into the same square is high. Thus, the functionality of the movement methods and mathematical calculations should be tested. Another essential thing is that the change of a character's status works problem-free. This could also be tested by comparing the expected and actual statuses of concerned characters in different situations.

Libraries and other tools

Here are some of the external libraries I have considered or thought of using in my project. For this part of the plan, I am willing to do some more research and I am happy to discuss with the course assistant which external libraries would be suitable alternatives to use in this project.

- PyQt (graphical user interface)
- Matplotlib (data visualization)
- Numpy (mathemathical fuctions)
- AgentPy (development and analysis of agent-based models, an open-source library)

10. Schedule

I am planning to do a more detailed schedule for my project in the upcoming weeks. Although the deadline for the project documentation and program codes is in June, I am willing to start working systematically with the project as soon as possible. My goal or estimate for the weekly working hours is from two to three hours. The guideline for the dates for checkpoints mentioned in A+ sound suitable for me. For the first checkpoint, my goal is to have all the different classes and their methods implemented. For the second checkpoint, the program is going to include also some testing and possibly even a rough version of the visual simulation.

11. Literature references and links

Here are some web sites I visited during this planning process.

AgentPy – Agent based modeling in Python - https://agentpy.readthedocs.io/en/latest/

Building simulations in Python - https://towardsdatascience.com/building-simulations-in-python-a-complete-walkthrough-3965b2d3ede0

Virus spread simulation - https://agentpy.readthedocs.io/en/latest/agentpy_virus_spread.html