ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

KIV/VSS

Coronavirus Outbreak Simulator

Jakub Šilhavý
A21N0072P
silhavyj@students.zcu.cz

15. 12. 2022

Contents

1 Introduction

Within this assignment, I decided to implement a Coronavirus outbreak simulation. The goal of the simulation is not to make any definitive conclusions about the disease as much as it is about comparing the results shown in other simulations to my simulation. In particular, I was curious to find out how frequent visit to popular places, such as pubs or restaurants, affects the spread of the virus.

The spread of the virus was visualized through a `GUI`[1], which allows the user to tweak different kinds of parameters of the simulation. Please, also read the `initial_design.pdf` as this document represents a follow-up.

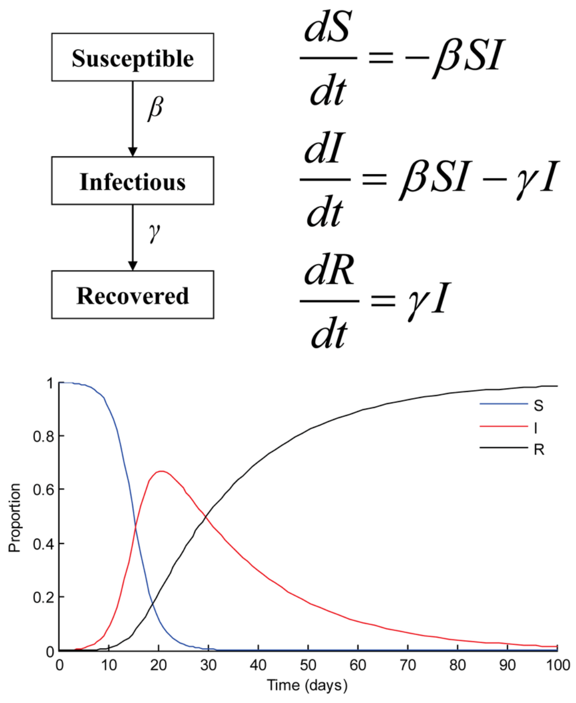
[1]`GUI` stands for Graphical User Interface

2 Model description

The basic model of the problem consists of a population made up of a number of people systematically moving around on a `2D` grid.

2.1 SIR model

Each person in the population can be either *susceptible*, *infectious*, *recovered* (*immune*), or *deceased*. These kinds of models, `SIR` models [1], can be solved using differential equations. A simple `SIR` model takes as an input two parameters - the transmission rate and the recovery rate. They are also referred to as β and γ.



$$\frac{dS}{dt} = -\beta SI$$

$$\frac{dI}{dt} = \beta SI - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Figure 2.1: Example of an `SIR` model

The results of the equations give us the progress of the disease among the people over time - how many people are currently infected, how many have recovered, etc. Since my model is based off of an `SIR` model, the final graphs should be similar to the one shown in figure 2.1.

[1]`SIR` is an abbreviation for *Susceptible-Infectious-Recovered*

2.2 Infection states

In addition to the `SIR` model, I have also added state *deceased.* The following diagram represents the transition between individual states.

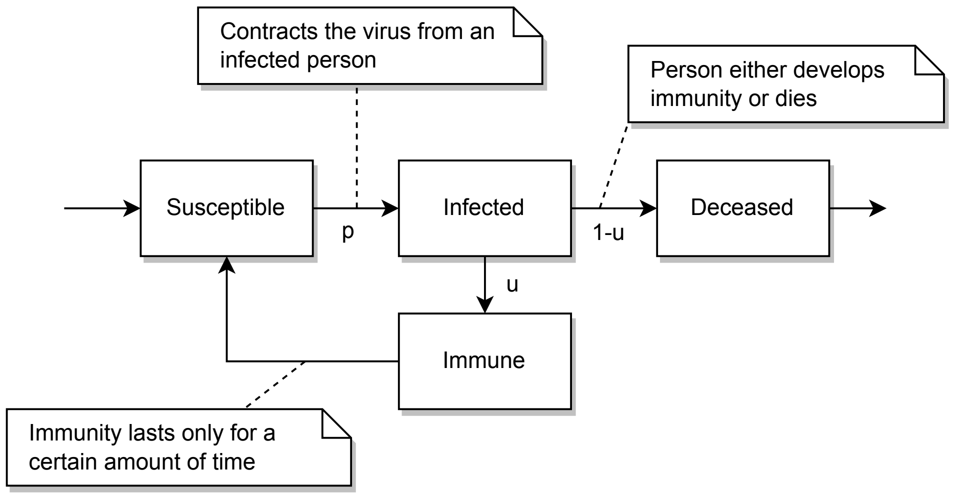

Contracts the virus from an infected person
Person either develops immunity or dies
Susceptible
p
Infected
1-u
Deceased
u
Immune
Immunity lasts only for a certain amount of time

Figure 2.2: Transition between different infection states

The simulation begins with a few unfortunate individuals who start off as infected and terminates when there are no infected people left.

2.2.1 Virus transmission

When a vulnerable person happens to be within distance l of an infected person, there is a certain probability p that they will contract the virus themselves. Assuming the person does catch the virus, they will go home with probability q and follow the same rules as if they had been self-isolating since the very beginning of the simulation. The probability $1-q$ could be interpreted as if the person has not developed any symptoms after contracting the virus or their test came back negative. Nevertheless, they can still spread the virus as they continue to move around unaware of the fact that they have become contagious.

The infection lasts for $x \sim Po(\lambda_x)$ number of days, after which the person either develops immunity for $y \sim Po(\lambda_y)$ days, with probability u, or dies with the probability of $1-u$. Additionally, if the number of infected people raises above threshold s, the probability $1-u$ doubles (or generally multiplies by k) as the healthcare system becomes saturated.

The Poisson distribution was chosen over the Normal distribution for the following reasons:

- The Poisson distribution can be used as a substitute for the Normal distribution for reasonable values of λ (e.g. $\lambda > 10$).
- The number of days is an integer (using the Normal distribution would require additional converting/casting).
- The Poisson distribution does not generate negative numbers (using the Normal distribution would require additional filtering).

2.3 Mobility model

2.3.1 Picking the next location

At the beginning of the simulation, each person is located at their assigned home location. Each person is allowed to stay at any given location for up to z hours (z updates of the simulation). When the time is up, they have to pick what location they will travel to next. There are three actions in total the user can take. They can either stay at home (which is also considered traveling to a new location), go to an arbitrary location on the map, or go to one of the popular locations. These actions are represented by probabilities a, b, and c which must add up to 1.

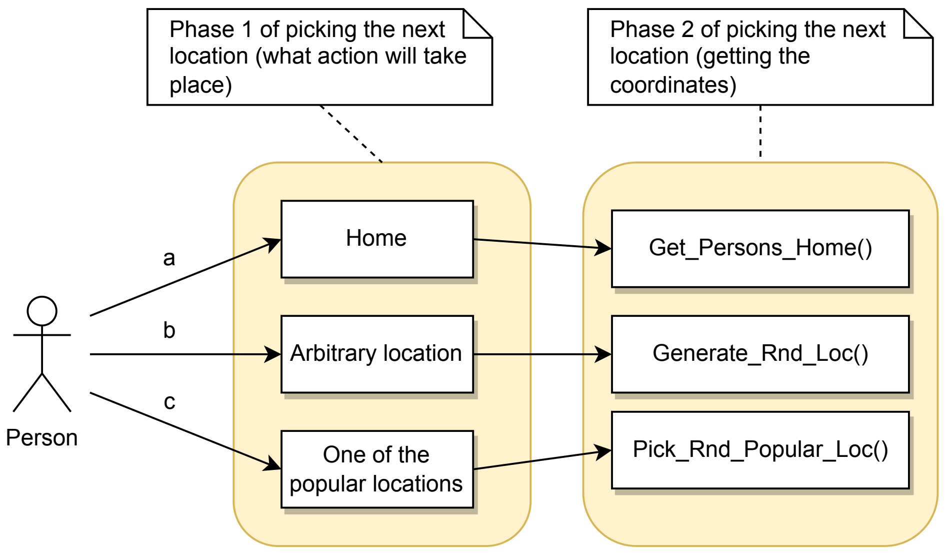

Phase 1 of picking the next location (what action will take place)
Phase 2 of picking the next location (getting the coordinates)
a
b
c
Person
Home
Arbitrary location
One of the popular locations
Get_Persons_Home()
Generate_Rnd_Loc()
Pick_Rnd_Popular_Loc()

Figure 2.3: Process of picking the next destination

Generating a random location as well as picking one of the popular locations is done using the Uniform distribution.

2.3.2 Self-isolation

At the beginning of the simulation, there is a percentage of people who decide to practice social distancing, meaning they prefer staying at home most of the time. However, they can still make occasional trips, such as going to a grocery store.

This feature is implemented using two sets of values of probabilities a, b, and c. For instance, when a person is self-isolating, their movement probabilities are set, by default, to 0.998, 0.001, and 0.001 accordingly 2.4.3.

2.3.3 Traversing to the next location

At each update of the simulation, the user is moved toward their final destination by a certain distance, which can be interpreted as the person's speed $\sim \mathcal{N}(\mu, \sigma^2)$[2].

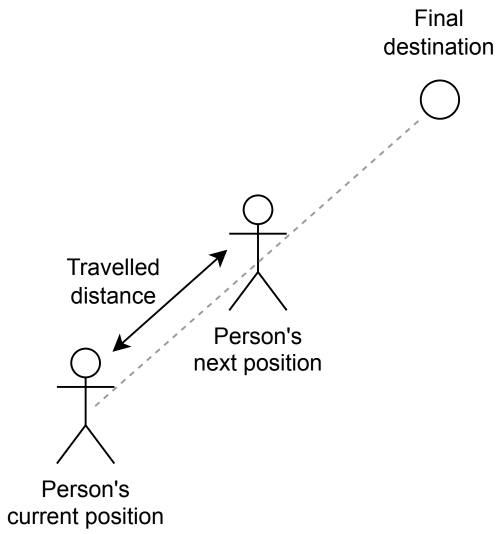


Figure 2.4: Single step of moving toward the next locations

Upon the person's arrival at the destination, they will stay there for $z \sim U(0, z\_max)$ hours, after which the same process of choosing the next destination starts again 2.3.1.

[2]Negative numbers get filtered out (person has to move towards the final destination, not away from it).

2.4 Model parameters

The model is driven by several parameters, all of which can be found in the `src/simulation/config.h` file. There are some default values assigned to each of the parameters. However, the user is more than welcome to change them as they wish. There are three groups of settings.

2.4.1 General

The first category, referred to as general, consists of the following parameters.

- Number of people in the population
- Number of initially infected people
- Percentage of how many people will be self-isolating
- Saturation level s of the healthcare system (e.g. if 20% of the population is sick at once, the system becomes saturated)
- World size (the world is represented as a square, which is another simplification)

The default values of the general settings.

```
struct TGeneral
{
    std::size_t number_of_people{500};
    std::size_t number_of_initially_infected_people{5};
    float ratio_of_people_in_self_isolation{0.5};
    float saturation_level{0.2};
    float world_size{2000};
};
```

2.4.2 Disease

The disease section of the settings holds information related to the virus and the way it spreads among people.

- Distance l, within which the virus can be transmitted
- Probability of the transmission $p_{outside}$ (when the person is moving toward their next location - being outside)
- Probability of the transmission p_{home} (when the person is at home)
- Probability of not recovering from the disease $1 - u$ (the healthcare system is not saturated)
- Probability of not recovering from the disease $1 - u_{sat}$ (the healthcare system is saturated)
- Probability of the person going home to quarantine once they catch the virus q
- Average period, in days, for which a person remains infectious λ_x
- Average period, in days, for which immunity will last λ_y

The default values of the disease settings.

```
struct TDisease
{
    float transmission_distance{5};
    float transmission_prob_on_move{0.5};
    float transmission_prob_at_home{0.5};
    float death_prob{0.15};
    float death_saturated_prob{0.3};
    float self_isolating_when_infected{0.5};
    std::size_t average_infection_period{14};
    std::size_t average_immunity_period{60};
};
```

2.4.3 Mobility

The mobility section represents the last section of the settings of the simulation. It defines parameters related to people's mobility.

- Average distance by which a person can move with each update of the simulation
- Variance of a person's speed
- Up to how many hours a person is allowed to stay at a location z_{max}
- Transition probabilities when a person is in isolation a, b, and c 2.3.1
- Transition probabilities when a person is not in isolation a', b', and c' 2.3.1

The default values of the mobility settings.

```
struct TMobility
{
    float average_person_speed = 10;
    float variance_person_speed = 1;
    std::size_t max_hours_spent_at_location = 8;

    struct TTransition
    {
        float go_to_popular_location_prob;
        float go_to_random_location_prob;
        float go_home_prob;
    };

    TTransition isolation     { 0.001, 0.001, 0.998 };
    TTransition non_isolation { 0.4,   0.2,   0.4   };
};
```

2.5 Measured statistics

Within each 24th update of the simulation, the following statistical values are collected, displayed, and plotted out.

- How many people are currently infected, immune, vulnerable, and deceased
- Whether or not the healthcare system is saturated
- How many times each person has been infected on average

2.6 Simplifications

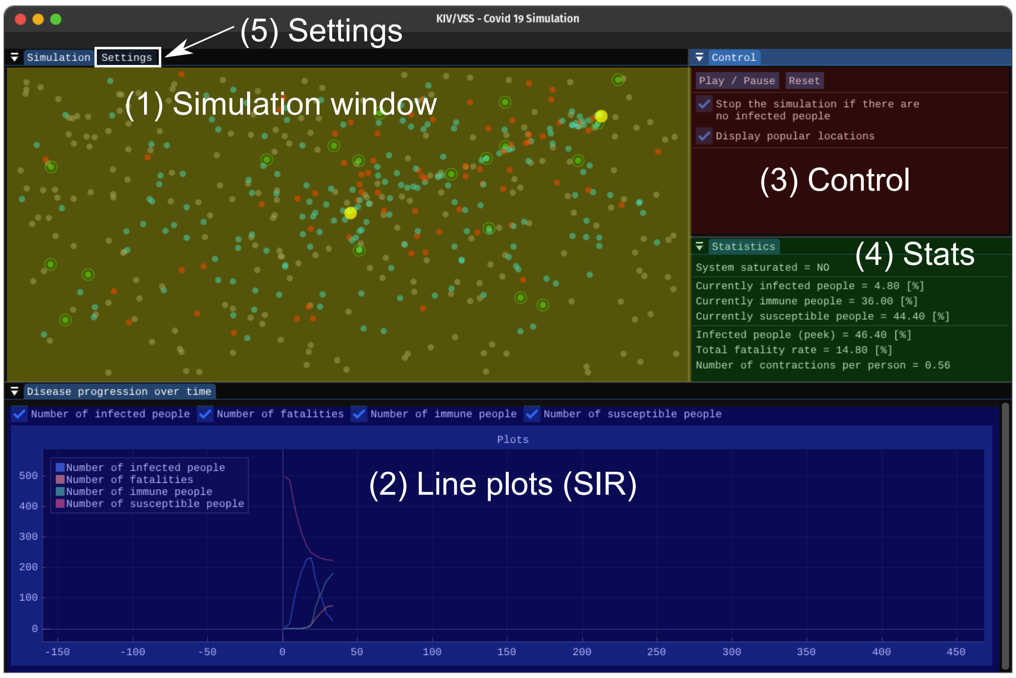
The simulation was implemented as a discrete simulation, meaning there is an error in interpreting continuous, events such as when a person is traveling to a certain location. Also, the people are moving on a 2D square grid, which may not be the best representation of a real-world scenario. Another simplification is the lack of traveling between different populations (e.g. international flights). It is limited to only one isolated population where people do not reproduce as the disease progresses. One attempt to improve other online simulations was to implement a mobility model that would better capture how people move around on a daily basis. However, the proposed model still has some flaws. For example, it does not reflect scenarios such as public transportation, which could potentially have a further impact on how the virus spreads.

3 GUI description

The GUI was written in `C++` with the help of `ImGUI`, which is an immediate mode library for creating `GUIs`.

The entire `GUI` consists of a couple of windows that can be arranged in any preference as the entire application is fully dockable. `ImGUI` creates an `imgui.ini` config file, which describes the layout of the windows making up the application. If the file does not exist, it will be created upon the start of the application. However, a default `imgui.ini` file was provided with the application. The layout it defines can be seen further below.

3.0.1 Window layout


KIV/VSS - Covid 19 Simulation
(5) Settings
Simulation
Settings
(1) Simulation window
Control
Play / Pause
Reset
✓ Stop the simulation if there are no infected people
✓ Display popular locations
(3) Control
Statistics
(4) Stats
Disease progression over time
✓ Number of infected people
✓ Number of fatalities
✓ Number of immune people
✓ Number of susceptible people
Plots
(2) Line plots (SIR)
Number of infected people
Number of fatalities
Number of immune people
Number of susceptible people
500
400
300
200
100
0
-150
-100
-50
0
50
100
150
200
250
300
350
400
450

```
System saturated = NO
Currently infected people = 4.80 [%]
Currently immune people = 36.00 [%]
Currently susceptible people = 44.40 [%]
Infected people (peek) = 46.40 [%]
Total fatality rate = 14.80 [%]
Number of contractions per person = 0.56
```

Figure 3.1: Layout of the windows of the application

Simulation window

The user can add different popular locations by clicking on the map. The point will also appear in a table located in the control window, where the user can further adjust its coordinates.

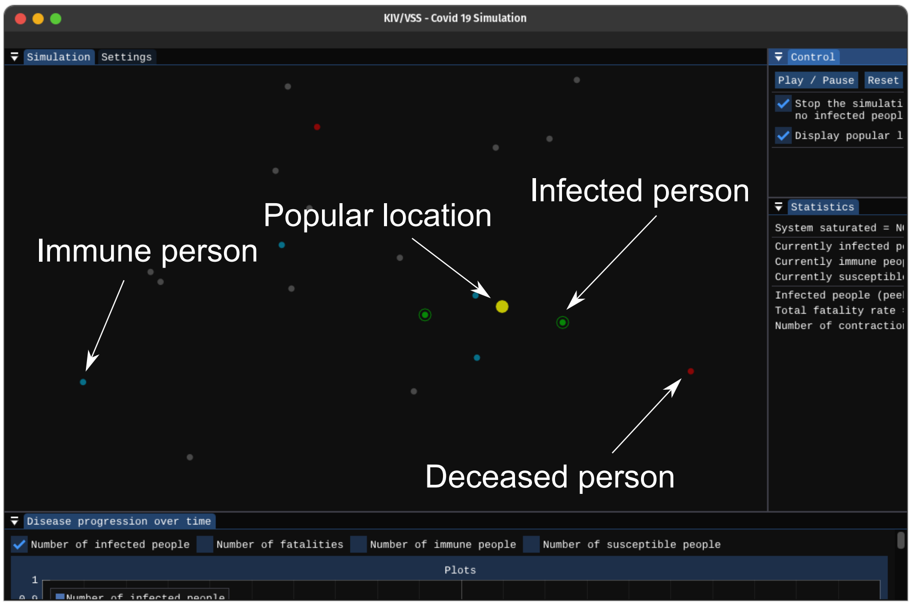


Figure 3.2: Simulation window

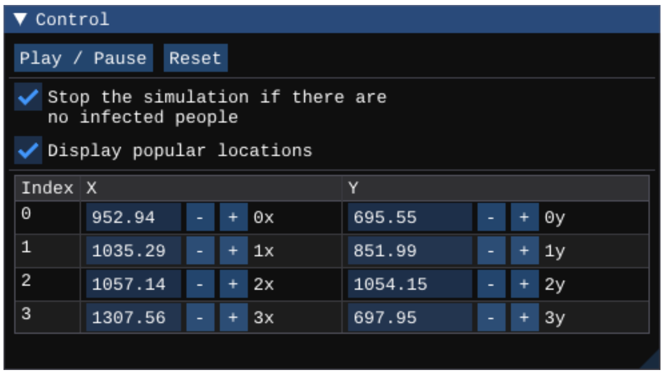
- Yellow circle represents a popular location
- Blue dot represents a person who has developed immunity
- Red dot represents a deceased person
- Green dot represents an infected person
- Gray dot represents a susceptible person

Plots

There are four different kinds of data being plotted out in real time. These plots should resemble the `SIR` model. The user can hide each and every line by checking its corresponding `checkbox` located in the top area.

Control

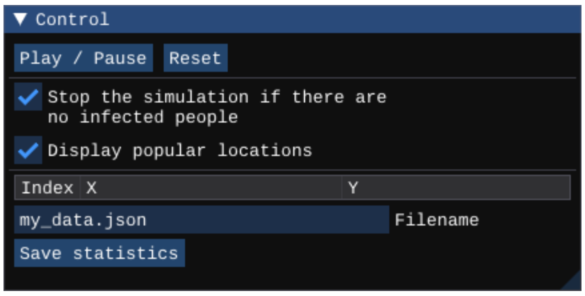
The control window consists of a `Play/Pause` button that allows the user to play or pause the run of the simulation, a `Reset` button that resets the whole simulation, a `checkbox` that terminates the simulation when there are no infected people, a `checkbox` that hides the popular locations on the map, and a table that allows the user to adjust the coordinates of individual popular locations.


▼ Control
Play / Pause
Reset
✓ Stop the simulation if there are no infected people
✓ Display popular locations

Index	X				Y			
0	952.94	-	+	0x	695.55	-	+	0y
1	1035.29	-	+	1x	851.99	-	+	1y
2	1057.14	-	+	2x	1054.15	-	+	2y
3	1307.56	-	+	3x	697.95	-	+	3y

Figure 3.3: Control window

Once the simulation is over, the user can save the measured data into a `JSON` file for potential further analysis.


▼ Control
Play / Pause
Reset
✓ Stop the simulation if there are no infected people
✓ Display popular locations
Index X Y
Save statistics

Filename: my_data.json

Figure 3.4: Saving statistical values into a `JSON` file

Statistics

This window contains the information on whether or not the healthcare system is saturated, current `SIR` values relative to the size of the population, the peak in the number of infected people, total fatality rate, and the average of how many times each person has contracted the virus.

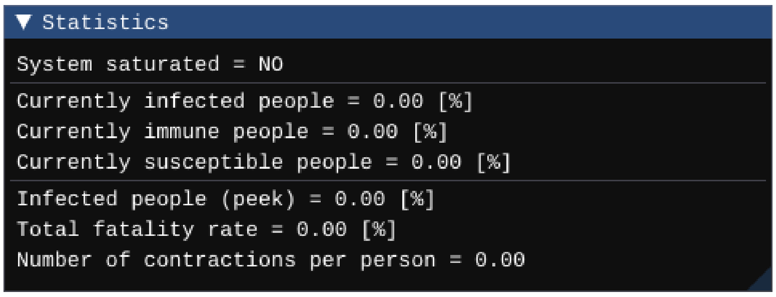
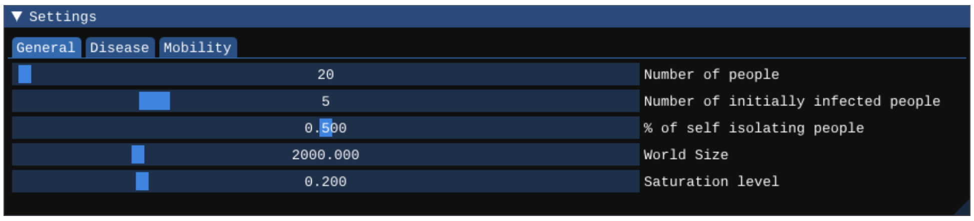

```
▼ Statistics
System saturated = NO
Currently infected people = 0.00 [%]
Currently immune people = 0.00 [%]
Currently susceptible people = 0.00 [%]
Infected people (peek) = 0.00 [%]
Total fatality rate = 0.00 [%]
Number of contractions per person = 0.00
```

Figure 3.5: Statistics

Settings

The settings windows allow the user to tweak different parameters of the simulation. Some changes, such as changing the size of the population, may require pressing the `Reset` button. There cannot be any changes made to the settings while the simulation is running.



```
20          Number of people
5           Number of initially infected people
0.500       % of self isolating people
2000.000    World Size
0.200       Saturation level
```

Figure 3.6: Settings

3.0.2 Build and Run

As to how to build and run the application, please see the `README.md` file located in the root folder of the project structure.

4 Experiments

For each of the following experiments, the simulation was run five times. The final value was then calculated as the average of all five runs. In order to generate random numbers, I used `std::random_device`.

4.1 Self-isolation

The first experiment was to find out how self-isolation affects the spread of the virus. All parameters were left at their default values 2.4, except for the percentage of people voluntarily staying at home most of the time.

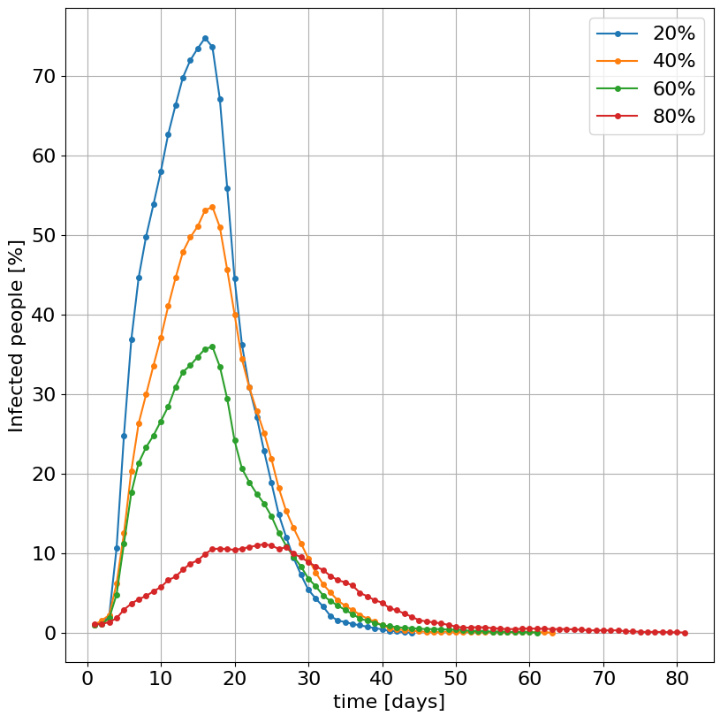


Figure 4.1: Growth of the number of infected people based on the percentage of people practicing self-isolation

As you can see in the figure above, with the majority of the people staying at home, the spike in the number of infected people is considerably decreased. On the contrary, if only 20% of all people stay at home, the healthcare system becomes saturated very early, resulting in a higher fatality rate as the people will not be able to receive the help they may need.

This kind of approach is also commonly known as "flatting the curve".

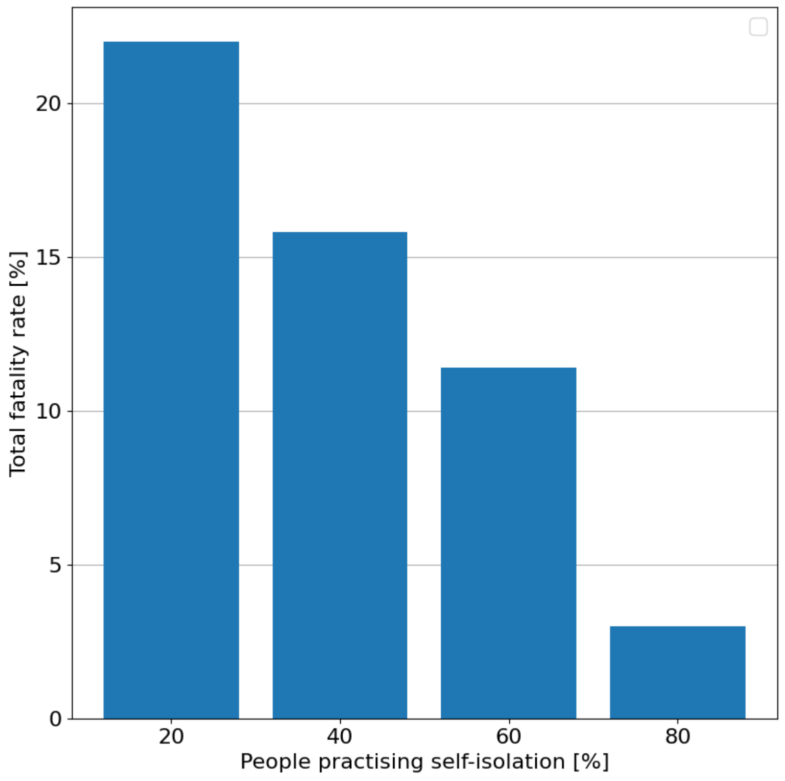


Figure 4.2: Fatality rate based on the percentage of people practicing self-isolation

4.2 Immunity

In this experiment, I wanted to test out the importance of people's immunity to the virus.

As you can see in the figure below, the longer the immunity lasts, the more it is able to fight off the disease. With immunity lasting for approximately 42 days, the virus disappears after the first wave. On the other hand, the fewer days immunity lasts, the more oscillations can be observed in the number of infected people.

As in the previous experiment, all parameters remained default, except for the immunity period.

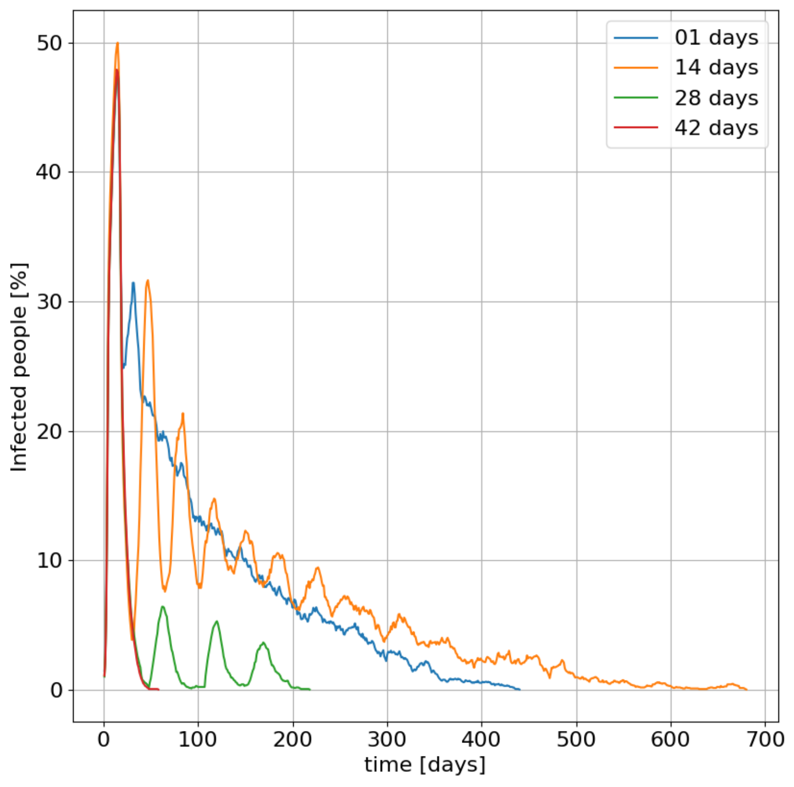


Figure 4.3: Disease progression with different immunity periods

4.3 Popular locations

The goal of this experiment was to test out how the geographical spread of frequently visited places affects the growth in the number of infected people. I chose to use 4 different popular locations arranged in a square pattern located in the center of the map (downtown area). The distance between two adjacent places was then varied with each run of the simulation.

As you can see in the figure below, the farther the locations are, the smaller the peak in the number of infected people is. However, it can be also observed that the pandemic lasts longer with each adjustment of the location of the popular places.

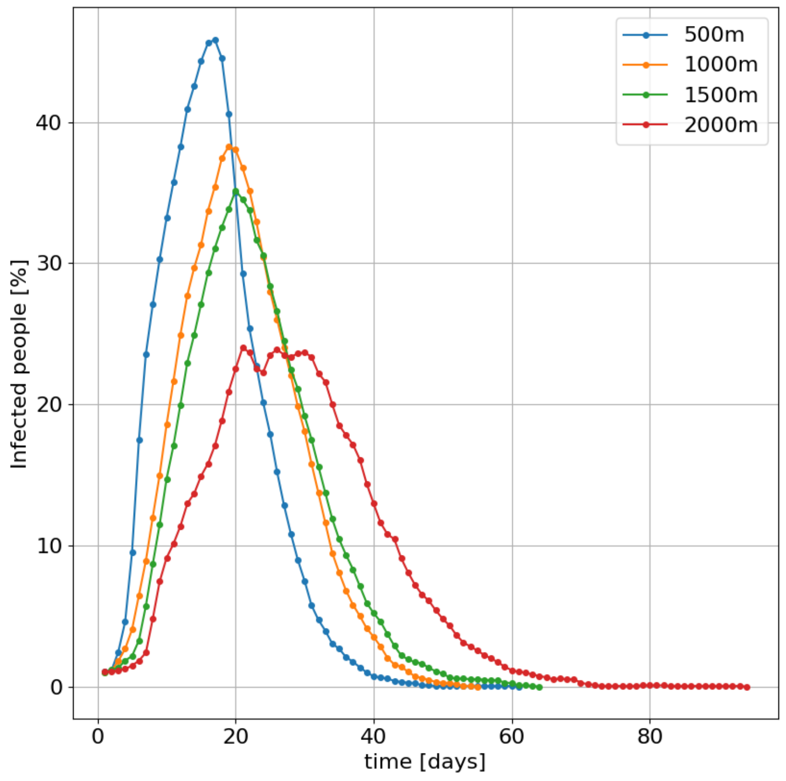


Figure 4.4: Progression of the number of infected people with different distances between 4 popular locations

5 Conclusion

The simulation model was found to be very unstable, meaning even a slight change in the settings can have a dramatic impact on the overall progress of the disease. For example, in some runs of the simulation, it took only a single infected person to make the number of infected people spike again. The most significant factors of the simulation were found to be the number of popular locations, people's willingness to self-isolate, and the ability of immunity to fight off the virus.

The application is meant to be interactive and configurable, so the user can run their own experiments and test out how different settings affect the spread of the virus. Despite some simplifications made to the simulation, the resulting curves still turn out as expected (`SIR`).

There are also some things in the `GUI` that could be potentially improved upon. For example, popular locations cannot be deleted once they are created (the user has to click on the `Reset` button).