ViruSim: An Interactive Epidemic Simulator

Alexander D. Christoforides
Rensselaer Polytechnic Institute
chrisa4@rpi.edu

Chris Dicovskiy
Rensselaer Polytechnic Institute
dicovc@rpi.edu

Figure 1: A screenshot from the developed ViruSim application which exhibits the live-updating graph in the top right which shows the proportions of the susceptible, infected, and removed populations, the main visualization space color coded by agent status, and the parameters panel in the bottom-left. Also notice how live counts for each population are displayed below the main visualization area in the statistics panel.

ABSTRACT

In an age like today where people across the globe have unprecedented access to information and data online, it is inevitable that misinformation can spread and affect the overall public's knowledge regarding certain situations. More specifically, and undoubtedly related to our project topic, when global pandemics or events occur, an immense amount of material is published online in attempt to educate the public, either in both good or bad faith ways. Through the information dissemination mechanism most prevalent today, the internet, people can find a variety of sources which recommend a swath of differing opinions regarding disease prevention and methods for mitigating further the disease's virulent spread. Additionally, it has become apparent through news stories reported online and on television that lay people sometimes do not have a well-founded intuition with respect to how viruses spread and various actions people can take to mitigate it. We put forth an implementation of an interactive visualization (ViruSim) with the purpose of educating lay people on how viruses spread and how various factors pertaining to the virus and human behavior can either help or harm the spread of the disease. Additionally, we put forth an extension of cellular automaton simulations which takes inspiration from previous agent-based simulation research to create an easily digestible visualization accompanied by a live-updating graph which visually shows the proportions of the population who are either susceptible to the disease, infected with the disease, or removed (recovered or deceased).

KEYWORDS

epidemic simulation, agent-based simulation, cellular automata, interactive, visualization

1 INTRODUCTION

During a pandemic situation like the one currently occurring (COVID-19), having resources to educate the general population on how viruses spread and how human behavior affects its overall spread is quintessential in mitigating the devastating consequences of the disease. This interactive visualization project aims to give a well-founded intuition to users and is an implementation of an epidemic simulation by way of agent-based, cellular automata. After reading a

collection of relevant literature on mathematical epidemic models, such as the SIS and SIR models as discussed in section 4, we felt we were capable of designing our own model which in its own capacity is able to visualize the spread of an epidemic in such a way that is not only based on fact but also can educate lay people how certain actions and simplified parameterizations of viruses can affect the overall virulence and spread of a disease. We wanted our simulation to be accessible and usable by anyone, so we felt a strong, minimalist, and straightforward user interface (UI) was important to the application.

Through interaction with the easy-to-navigate UI, users are able to select various parameterizations of the virus and human behavior (only density allowed to be tweaked with at this time) and visually see the resulting spread and consequences of the hypothetical disease. Various situations can be created with the limited number of parameters already implemented and further extensions to this application can make it ever-more complex and allow for more intricate and nuanced scenarios which more accurately reflect the spread and consequences of viruses. In the following sections, we talk in more detail regarding our motivations, goals, implementation details, and general results.

2 PROJECT MOTIVATIONS AND GOALS

In this section we discuss the various motivations and goals for the project and how they relate to the final, developed product. Section 2.1 discusses our primary motivations for this project and section 2.2 touches upon other pieces of work which we found online to help inform the idea structure and form of our project.

2.1 Motivations

As previously stated in section 1, our motivations lie solely in the necessity for easily accessible and true information regarding how viruses spread at a surface level. The current COVID-19 situation that plagues the world highlights many holes in peoples' intuitions regarding how viruses spread and how various precautions and advice given by government organizations and global watchdogs go undigested. As an example, the state of Georgia in the United States is going against all advisories given by the Centers for Disease Control (CDC) and World Health Organization (WHO). The state is in no better state than it was previously and the leadership there sees it fit to reopen the state to help mitigate the economic impacts that the virus has had on its citizens [5].

This is not an isolated incident by any stretch of the imagination. Vocal minorities specifically across the United States are holding rallies urging government officials to lift the state-wide shutdowns currently enacted in the majority of states in the United States. What motivates us the most is that while these groups are minorities with respect to their size, they appear to have a significant amount of influence as we see more and more states flirt with the ideas espoused by the protesters and intentions of the president [2].

It is because of these instances and a demonstrated lack of general knowledge online and in media regarding how viruses spread that we decided to create this project.

2.2 Miscellaneous Project Inspirations

When designing our project, we took inspiration from two different sources: one linked to us by our professor from The Washington Post [7] and another from a YouTube video [1].

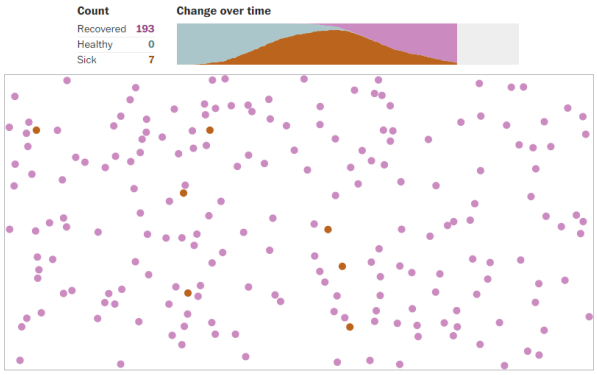

Figure 2: Disease spread simulation from Harry Stephens at the Washington Post [7]. In the non-static form of the visualization, the agents (circles) move around and collide with each other; this is how the disease is spread. Additionally, the graph at the top updates as the simulation continues and shows the portions of the population who are susceptible, infected, or removed.

The first, as seen in figure 2, highlights moving agents in a simulation and uses contact between them to determine whether or not a susceptible agent becomes infected; it also shows a stream-like graph which communicates the proportions of each population in the simulation colored by their infected status. While this implementation was accomplished in D3 (https://d3js.org/), we thought it would be possible to implement something similar in a native desktop, python application instead. Additionally, we saw their version of the stream-like graph and sought to make something similar on a larger scale so that it was easier to read. Furthermore, the author's explanation of various situations that they setup in the simulation inspired us to think about possibly having a mechanism for allowing custom creation of scenarios with hand-drawn boundaries, but the time we had for the project

did not permit this and will be explored further in section 8 where we discuss possible future work extensions to our existing project.

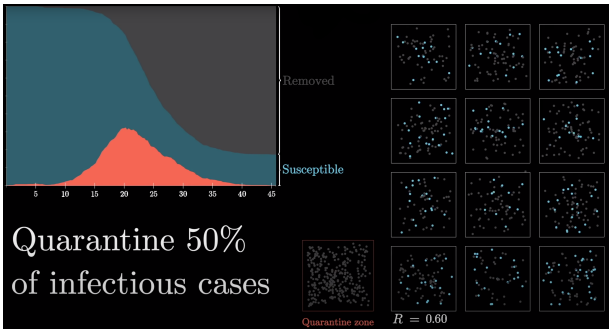

Figure 3: Disease spread simulation from 3Blue1Brown, a YouTuber [1]. This visualization similarly utilizes moving agents (as seen in figure 2) and throughout the video demonstrates various scenarios and analyzes their impacts on the overall spread of the disease. Notice how this visualization also has a stream-like graph; inspiration for ours came from here as well. Notably, the shown scenario of quarantining 50% of the infectious cases can be seen and acted as a catalyst for us to think of other types of situations that we could implement and control in the future.

The second miscellaneous project inspiration came from a YouTube video titled "Simulating an epidemic" by YouTuber 3Blue1Brown [1] (see figure 3). This simulation can be characterized as very similar to the one created by Harry Stephens from The Washington Post as the both feature moving agents and a live-updating stream-like graph to communicate the number of agents who are susceptible, infected, and removed over time. This resource in general proved to be insightful however since the video covered many more possible scenarios that you can construct with a simulation like theirs and allowed us to ponder many new features and possible extensions (discussed in section 8) that we could integrate into our project.

2.3 Goals

Goals for this project can be categorized as both abstract and technical. To understand what these are, please read sections 2.3.1 and 2.3.2 respectively.

2.3.1 Abstract Goals . Our goals with this project can generally be understood as providing an easily accessible piece of software which allows people who do not understand viruses and the general mechanics of disease spread to come to develop a better, fact-based intuition regarding how viruses actually spread given a set of simplistic parameters which govern how the simulation is ran. When planning the project, we knew that we did not want to target experts or provide a piece of software to them which can help them develop models and legislative policy since the existence of this type of software for them already exists and is magnitudes more complex than the type of software we initially set out to write. From a survey of popular media sites and reactions by people online, it was clear that many people did not fully understand why stay at home orders and social distancing policies are so essential in the fight against a disease like COVID-19; this led us to generate our goal of making this type of software. Substantively, the simulation should be easily understood and the parameters should be self explanatory so that the barrier to entry regarding the use of our software would be as low as possible.

2.3.2 Technical Goals . From a technical perspective, our goals primarily pertain themselves with being able to not only create an application capable of simulating the spread of a disease under various parameterizations but also easily communicating those results to viewers through a graph-like visualization. While giving the user an area where they can see the raw numbers of agents in the simulation who are susceptible, infected, or removed, is nice, we believe that this is not good enough in allowing people to develop the intuition we have mentioned in prior sections. To alleviate this shortcoming, another technical goal we have is creating and implementing a type of live-updating graph which is able to clearly communicate the proportions of the population which fall into the three categories which label every agent in the simulation (susceptible, infected, removed). Additionally, to better digest the results of the simulation and to give users a view of how changes in the simulation parameters give rise to varying consequences in the effect of the virus spread, a pause and resume and restart simulation mechanism is necessary, and thus another goal for our project.

3 AUDIENCE, RESEARCH QUESTION, AND HYPOTHESIS

Below is a review of our target audience, proposed research question, and following hypothesis.

3.1 Audience

The audience for this project was rather important to define. This is primarily a tool meant for lay people not necessarily knowledgeable about the spread of diseases; the audience can be loosely defined as anyone interested at seeing the occasional cataclysmic sequence of chain reactions that constitute an epidemic, from a third-person, almost bird's eye view. If we could magically make people use our application, we would target everyday people who currently do not have

a grounded intuition regarding how viruses spread and how certain aspects of viruses and human behaviors directly contribute to the overall success of a virus. A specific audience as an example would be the people previously mentioned who are protesting the government to lift stay at home orders; if they understood the ramifications of doing this, not only in an economical sense but also in terms of the resulting virus spread, then we believe the intuition they would gain would lead them to change their general position regarding policies enacted by the government to stymie the spread of the virus.

3.2 Research Question

Can an application made for lay viewers communicate the effectiveness of the recommendations given by government bodies and health officials in an epidemic or pandemic scenario? Our primary target audience helped us to come up with this research question for our project. In a time where it appears that many do not understand the fundamentals regarding how viruses spread, we saw it fit to create an application which is easily accessible and understandable by most lay users to help them gain an intuition regarding how viruses spread and how various things like social distancing help to mitigate the destructive nature of the virus currently spreading. Even though in-depth answering of this question would most likely require a user study (which is unfeasible for this project), we can make general assumptions and implement best practices which have already been proven to be useful in helping to communicate results to the viewer in an effective manner; one such finding we draw upon is using an attractive, slightly-modified, color-blind aware color scheme from ColorBrewer (https://colorbrewer2.org/) to color our agents and detail our live-updating graph.

3.3 Hypothesis

We hypothesize that after end users toy around with the parameters of our simulation, they will acknowledge the importance of the recommendations given by the aforementioned government bodies and health officials, the most notable being social distancing, or at least develop an intuition which would eventually lead them to understand the recommendations by government bodies and health officials. While we cannot guarantee that a user of the application will immediately understand virus spread at a surface level, we can confidently assert that they should be able to develop an intuition which would lead them to viewing the virus in a way which is more in line with the description from health officials. Even though this would be hard to test given the lack of a formal user study (since this is a school project), following best practices to create visualizations like this help us to rationalize the possibility for users to develop the intuition we seek to give them through this application.

4 PRIOR WORK

Below you can find a small survey of existing academic literature which we used and expanded upon in the creation of this project. To create our model, we took inspiration from the references below and integrated those aspects into our project to create something based upon prior areas of research.

4.1 *Three Basic Epidemiological Models*, Hethcote, 1989

Hethcote, in his work titled Three Basic Epidemiological Models, reviews three models for infectious disease spread and lays out the general math formulations and equations necessary to not only understand the various models, but to also simulate them. While both the SIS and SIR models are reviewed, the third is more or less a description and conceptualization of a model which takes into account vaccines and herd immunity and their subsequent effect on the spread of the disease. Hethcote goes into precise detail regarding the notations used to formulate the various models and explains how subtle variations in each can have a dramatic effect on the results of the disease spread being modeled [4].

4.2 *Agent-Based Simulation Tools in Computational Epidemiology*, Patlolla et al., 2004

Patlolla et. al put forth an analysis of the state-of-the-art modeling methods used for simulating the spread of diseases in their paper titled *Agent-Based Simulation Tools in Computation Epidemiology*. The authors mention the lack of computational models relating to epidemiology since the access to data in the past was more bleak than it is today; they motivate new methods of computational epidemiology through the existence of an abundance of data today and discuss the benefits of technologies which utilize agent-based models. Patlolla explains the various usage scenarios of agent-based and cellular automata models and how each one is better at modeling various situations like local and global outbreaks. Throughout the piece, they explain how some agent-based models are created and the various factors that can be explored in attempt to more accurately simulate the locomotion and interactions between humans in a society. While they implemented a model based on collected data, they outline the underlying simulation required to generate "believable" actions taken by the agents; the theory behind modeling the "thresholds" as demonstrated by Patlolla et. al was expanded upon in our implementation of ViruSim [6].

4.3 *Modelling disease outbreaks in realistic urban social networks*, Eubank et al., 2004

Eubank et al., in their article *Modelling disease outbreaks in realistic urban social networks*, describe the processes involved in their model for simulating disease outbreaks. The model is based off bipartite graphs which reflect the real-life patterns of physical contact between people. These bipartite graphs are meant to represent two elements of the simulation, notably the people and locations; the people here are edge-connected to locations. These edges are noted by arrival and departure times, indicating when the person came and left the location. Should two or more people coexist at a location simultaneously, a disease-harboring agent would spread the disease to a non-disease-harboring agent. Using bipartite graphs in this manner is effective in showing the potential for disease to spread solely based on the presence of agents in a hypothetical, non-spatial location in a simulation. While this is somewhat contrary to the idea of our physical 2D simulation, the article provides much insight on how we could potentially implement graph theory to model relationships between agents and their physical locations based on proximity [3].

5 VISUALIZATION DESIGN EVOLUTION

In the following sections, we discuss the design of our visualization and how it has evolved over time.

5.1 Initial Vision and Storyboard

Our initial design for this project revolved around creating a simulation for easy consumption in addition to helping people develop an intuition regarding how viruses spread given a set of easy-to-understand parameters and a clear visualization method. The initial storyboard used to design and construct this project can be seen in figure 4.

The initial plan was to create a multi-agent-based simulation which utilizes agent movement and collisions in order to transmit the disease from an infected agent to a susceptible agent. Viewers would have been able to see the agents moving, in real-time, in the main visualization space in the application and they would be allowed to tweak parameters pertaining to the simulation in addition the epidemiological model which the simulation would use when trying to simulate the phenomena. While the simulation is running, a live-updating chart would begin to fill out where each time slice represents a given time step in the simulation; the time slices show the proportions of the population who are susceptible to the disease, infected with the disease, and removed (killed by or recovered from the disease). These were the initial design goals we conceived before creating the implementation, but as explained in section 5.2, not all of

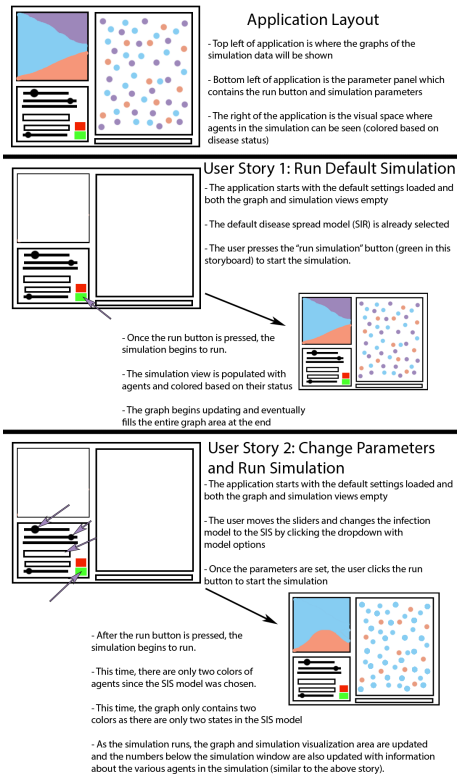

Figure 4: The above is a picture of our storyboard which we used to plan and design our application. Take note of the two user stories we included and see how they are both different from the visuals and functionality of our end result project in figure 1.

these made sense and were later revised to better accomplish the overarching goals we set out to achieve with this project.

5.2 Design Revisions

Detailed below are three major design revisions we went through in attempt to make the project useful and functionally make sense with respect to the goals we outlined in sections 2.3.1 and 2.3.2.

5.2.1 Revising Usage of the SIS and SIR Models. When initially designing the project, we thought it useful to delve into the details of both the SIS and SIR models described in Hethcote's work [4] to devise a simulation which embodies the key characteristics and parameters which exist in these models. Upon further investigation, however, we noted that these were the definitive mathematical models which describe how a virus ideally spreads within a given population and does not lend itself towards the abstract goals (see section 2.3.1) we were trying to accomplish. Simulating the

SIS and SIR models given various parameterizations would yield the same results every time since the models describe idealized disease spread in contrast to being a framework which generalizes disease spread and the various factors which contribute to it; as previously stated, simulating them would result in the same simulation outputs to occur each and every time.

Based on these findings, we made the decision to move away from these models as the underlying basis for our simulation and move towards a more agent-based system as discussed in both [6] and [3]. The prior work mentioned describes ways to devise meaningful parameters which govern how simulations, most notably disease simulations, can operate. This proved more fruitful as it gave us the ability to create agents which react to these parameters in a non-deterministic way, thus giving rise to our simulation results which can be seen through the operation of our application; instead of receiving the same results every time, users can change the carefully-constructed parameters and see nuanced changes with respect to the findings presented in the simulation.

As one can see in figure 4, we moved away from the drop-down item which allows users to select the model they wish to use and instead kept the simulation parameters box and additionally added more entries to this list so that more complex and comprehensive simulations could be ran.

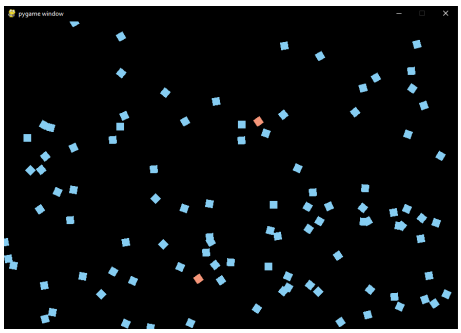

Figure 5: This shows our attempt at trying to create free-moving, circular agents which can collide with each other with pygame. Unfortunately, as seen in the picture above, we were not able to generate circular agents which can collide. Additionally, when the simulation is ran, the squares were not even colliding with each other as well. In the interest of developing a working application which still adhered to our goals described in sections 2.3.1 and 2.3.2, we opted to switch to an agent-based, cellular automata based simulation.

5.2.2 Agent-Based Cellular Automata instead of Free-Moving Agents. Originally, the plan for the main visualization area of the project was to have circular agents move around freely and collide with other agents to simulate them passing along the disease from themselves to another susceptible agent. This proved difficult however, so we opted to switch to a hybrid simulation which takes inspiration from both agent-based simulations and cellular automata based simulations.

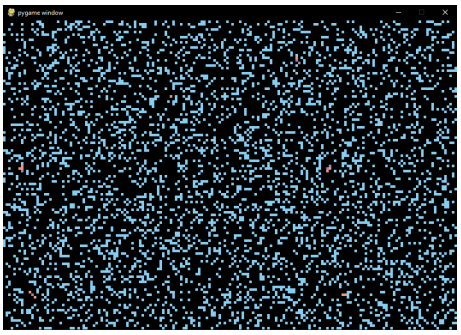

Figure 6: The image above shows our initial attempt at visualizing the agents in our simulation. As seen in the image, it is hard to discern certain agent statuses when they are displayed so small on the screen. Additionally, it is hard to discern certain geographic distributions and identify how their position and size affect the overall ability of the disease to spread.

The results we received from running the simulation lined up with various other visualizations on this topic which did not solely simulate the SIS and SIR models. Despite the difficulties, we were able to generate free moving agents but they were not circular and did not collide with each other (see figure 5). While it was not what we initially envisioned, the new method of visualization still proved useful and faithful to our goals. In some ways, it may be more effective since it does not clutter the visualization with many moving agents and instead discretized the agents which emphasizes their status and area of effect with respect to the potential agents they could affect in their immediate vicinity. As future work, we would like to change this model to use the freely-moving agent-based method for simulating these phenomena since we would be able to introduce a level of complexity pertaining to the agents which determine how they move and behave; this would give rise to more complex simulation scenarios and results that could prove more intuitive.

Along with this change, we originally thought that it would be better to have more cells visible at once during the simulation. After seeing this for the first time however, we saw how tiny each of the agents were and we came to the conclusion that it would be difficult to discern certain agent statuses in addition to various geographic distributions in the data which help prevent further spread of the disease (see figure 6).

5.2.3 UI Color Scheme. The initial color scheme at the beginning of the project was undecided, so we both came to the conclusion that we should employ a color scheme which captures the eye's attention in addition to highlight the various regions of the application to the user for easy use. When the initial colors were chosen, we made a conscious decision to use colors which would really stand out and allow viewers, especially those with some troubles seeing, to be able to discern the various parts of the application. In figure 7 you are able to see the initial coloring we decided to use.

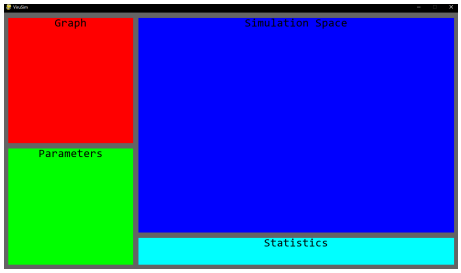

Figure 7: This is an image which shows the old user interface coloring that we initially had for our project. While it was mainly used for debugging purposes, we chose colors that were different in order to help some viewers identify which panel they were looking at more clearly. The colors selected here were too saturated and had no coherent theme, but the underlying principle behind selecting them remained present. The final version of our application utilizes one main background color since it is more aesthetically pleasing and panels could still be distinguished easily due to the difference in brightness between panel backgrounds and the application's background.

After merging both parts of the code together (UI and simulation), we took the time to analyze what we had and switched the color theme so that there was one dominant color which represented the four panels in the simulation rather than use four different colors to accomplish the same thing. Looking back, the original design, while it probably would never have been used as the final product, looked like a child made it. The main ideas were decent in that we were thinking of using different colors to help some users of our application, but the initial selections were too saturated and eccentric. While they were mainly chosen for debugging purposes, they were also selected in the same vain in which we opted to use larger agent sizes; for some users, one main color may not be as easy to see and this would possibly have the ability to confuse viewers with respect to what panel they were looking at. We eventually switched to using one main background color and a darker shade of the same color to distinguish the panel backgrounds from the main background. We believe this to be just as effective; the equidistant spacing between all the panels in the visualization also contribute to a users ability to separate the panels at which they are looking at and generally creates a more coherent presentation for viewers to look at and analyze results of the simulation.

6 VISUALIZATION DESIGN FEEDBACK AND CRITIQUES

When creating this project, we were initially given some feedback regarding our scope and possible extensions to the proposed idea to make it better.

One comment revolved around making a Plague Inc. type of game/visualization and to allow users to paint virus areas in the visualization and see the corresponding results. While this was never implemented, we saw it fit, given the current functionality of the project, to create infected agents in a random manner for now. Due to time constraints of adding this extra level of interactivity, we believed it not feasible within the given time frame. This is not to say, however, that it was a bad suggestion by any stretch of the imagination. As discussed in section 8, you can see our discussion regarding more interactive painting features and can clearly see how this influenced our thinking in this area.

Another pertained to allowing the user to specify the more biologically-relevant factors of a virus to more accurately simulate its spread. While this is an enticing extension to the project, we again felt that it would not only be unfeasible during the time allotted, but also way too complex and would hinder the ability for our target audience to gain a surface level intuition regarding virus spread and the ability for modified human behavior to help curb its spread.

The SIR and SIS models were also mentioned as a point of feedback towards us; while they were great to read about and helped us develop the various parameters currently present in our simulation, they are not useful with respect to simulation since simulating them would yield the same results over and over again unless a new parameterization was chosen for the factors which are considered in those models. While this is true, it led us to consider the potential use of the SIS and SIR models in a comparison type of view which would allow users playing with the simulation tool to understand how the results they are seeing line up with an idealized form of the virus spread. While not implemented, we discuss this further in section 8.

One important recommendation we received was regarding the overall scope of the application. When initially pitched as an idea, all of the existing material was present, but also other features such as multiple disease spread models and

more complex agent models were part of the proposed design. When reading this feedback, it helped us to identify which parts of the application were most critical given the time constraint in addition to allowing us to hone in on our audience and create an application which best targets them and still achieves our goals outlined in sections 2.3.1 and 2.3.2.

Last, but certainly not least, while we presented our project during class, one person mentioned possibly adding a healthcare utilization parameter which would simulate consumption and over-exhaustion of real-world resources which are necessary to deal with diseases. For further explanation of this, please see section 8 which describes the potential areas for future work on top of our existing application.

7 IMPLEMENTATION DETAILS AND CORE FEATURES

Below is a survey of various existing libraries, custom data structures, and algorithms we used in the creation of this project. For quick reference, section 7.1 discusses the third-party libraries we chose to use, section 7.2 discusses the main features of the application, section 7.3 describes the custom data structures we opted to use, and section 7.4 chronicles the details of the algorithm that we employed to run the simulation. Furthermore, section 7.5 explains the various challenges we encountered along the way while developing this project.

7.1 Usage of Existing Libraries

The visualization of our simulation model came about from usage of the PyGame library. This extensive library was mostly used to draw two-dimensional shapes onto the screen, shapes which are representative of our agents. The library also helped in creating the four panels present in the simulation in addition to allowing us to color our agents with convenient RGB combinations. All of these necessary tasks were accomplished via the `pygame.draw.rect()` function, which would specify the coordinate location, size, and color of a rectangle. On top of this, the library enabled us to create the window, set the desired window properties, and allowed us to specify that our application should be double-buffered to avoid screen tearing.

ThorPy (http://www.thorpy.org/), a small library developed by Yann Thorimbert, was used for the creation of the user interface employed in our application. ThorPy enabled us to easily create interactive user interface elements which can be fiddled with in order to tweak the settings currently employed in the simulation. This library, while most features were undocumented, gave us many freedoms with respect to stylizing the elements in our desired way. Most importantly, the ability to group these elements in menus and position them quickly and easily in the application made our development process all the more quick. As previously mentioned, most features in this library are undocumented, one being how to retrieve data from the element, but after looking through one of a handful of examples, we were able to discern how to accomplish this and to our surprise, it was trivial.

7.2 Core Features

The ViruSim application window is segmented into four distinct sections that make up the features of the project. First, the simulation section that takes up the majority of the space in the application is where the visual aspect of the simulation takes place. This naturally takes up the most space in the application because it is the main feature. The agents in the simulation are color-coded with visually qualitative colors to distinguish between different states. This simulation is always updating with agents changing color in a manner that reflects the rules of the model. We hope that being able to actually see the epidemic spread around this fixed area is of enough interest to encourage users of the application to toy around with the next feature, the parameters. The parameters encompass the interactive aspect of this interactive visualization. This feature is meant to keep the user engaged with designing the myriad of scenarios possible for an epidemic to spread. There are sliders and text boxes that allow a user to change the infection chance, the infection radius, the population density, the removal chance, the initial amount of infected agents, and the random seed. Along with this, there are buttons to run, pause and restart the simulation. Next, is the graph which is nestled in the upper left corner of the application. This feature informs the user, in a more straightforward manner compared to the visual simulation, the proportion of susceptible, infected and removed agents. This is updated simultaneously with the visual simulation. Finally, below the visual simulation is the statistics section. This small feature is updated with every tick and keeps track of the actual numerical composition of the agents' statuses.

7.3 Data Structures

The ViruSim application requires a number of data structures to organize and encapsulate distinct portions of code.

7.3.1 `Agent`. An `Agent` object is contains all the code that is relevant for agents being updated at every tick of the simulation, drawn to the screen, check for infection, and check for recovery. These last two processes are dependent upon the `Disease` object that the `Agent` class houses as a member variable.

7.3.2 `Disease`. The `Disease` class behaves more as a struct as it has no member functions, instead serving more as an

organizational body for a disease's traits. It contains a radius, an infection chance and a removal chance, all three of which are parameterized by the user. There is also a boolean member variable that indicates whether the `Agent` with this `Disease` object is recovering or not.

7.3.3 `grid`. Finally the data structure that contains all of these agents is a two-dimensional Python list, `grid`. This list has matrix-like indices that take into account the spatial location in the simulation. For example an `Agent` at position `grid[0][0]` is to the left of an `Agent` at position `grid[0][1]` in the actual simulation.

7.3.4 `UIPanel`. Every user interface panel in the application was modularized and created with a helper class in order to create easily customizable panels for the content to be placed within. Each `UIPanel` object was represented by a position (x, y) and a width and height (w, h). Additionally, the panels contains optional header text which can be rendered directly in the top-center of the panel using a member function `render_panel_title_text()`. Through this and other helper functions like `get_title_height()`, positioning other UI elements in those panels was made trivial and allowed us to focus on other parts of the application's development.

7.3.5 `Graph Data`. To render the live-updating graph, a list structure was used to contain all of the previous line segments for all previous time steps. PyGame requires the program to redraw all elements on the screen every frame so caching the red, blue, and purple line segments for each frame was necessary. To accomplish this, the list was formatted as follows: $segments = [s_1, s_2, s_3, ..., s_n] : s_i = ((start_x, start_y), (end_x, end_y), color)$. The various x and y positions in the list correspond to the start and end locations of a given line segment; while drawing the segment, PyGame utilizes the color argument to color the actual line. This means that for each line drawn to the screen, three sub-lines are drawn. Every frame, the application iterates over the entire list and redraws all of the lines so that prior time slices are represented and new ones are continually added to the list for visualization.

7.4 Simulation Algorithm Discussion

Before the simulation can begin, a set amount of agents must be spawned into `grid` depending upon the population density parameter. This random distribution is achieved by iterating through each slot in `grid`, producing a pseudo-random number between 0.0 and 1, and initializing an `Agent` in that slot if the generated number is below the population density parameter. Simultaneously, initially infected agents are also populating `grid` in random locations.

After `grid` is populated, the simulation can commence. The simulation enters a loop that iteratively updates the agents. Because only infected agents can alter the state of the simulation, we only really have to check whether an infected agent infects its neighbors, or removes itself from the simulation.

For an `Agent` to infect another, it must itself be infected. When this occurs, either by being an initial infected, or being infected by another, it must, at every tick of the simulation check its neighbors, generate pseudo-random numbers between 0.0 and 1 for each neighbor, and infect those with numbers lower than the infection chance parameter determined by the user. The number of neighbors for each agent is determined by the radius parameter, and can be explained simply as the number of "rings" around an `Agent`. For example, a radius of 1 would check the maximum 8 agents that surround any `Agent` in the `grid`, besides those on the boundaries. A radius of 2 is checking the neighbors of radius 1 neighbors along with the "ring" of agents that envelop the radius 1 neighbors. This pattern is repeated for higher radii. An `Agent` that has been infected by this process must partake in this infection checking of its neighbors, meaning the speed of the simulation is often significantly slowed with a significantly large amount of infected agents.

For an infected `Agent` to remove itself from the simulation, it must also perform a pseudo-random number check. This check is, like an infection check, done at every tick of the simulation. If the 0.0 to 1 number generated is below the user parameterized removal threshold, the infected `Agent` is now considered removed and unable to infect or be infected.

7.5 General Challenges

An impediment faced during the programming of the model forced us to change the model itself. We initially outlined a model that would involve moving agents that would randomly bump into each other in order to spread an infection. But while almost everything was in place for this version of the simulation to materialize, complications with PyGame's collision logic forced us to take drastic measures and build current model. It was a shame really that we were unable to get our initial model up and running, especially since it meant we had to scrap a significant amount of code that did work. For example, it took us quite a while to write up code that had agents moving in an organic-looking manner. But after struggling with PyGame's implementation of collision checking for almost two days, and panic setting in regarding the deadline of the project, we made the difficult decision to cut our losses with the progress we had made with our original model.

8 FUTURE WORK

There are many directions that ViruSim can be taken, but most work will seem to center around adding more features in the way of parameters, a tool that allows for the comparison of different models, a tool that allows for custom scenario design, and a conversion from our current cellular automata model to our original free-roaming agent simulation.

Firstly, adding more parameters could be have beneficial use to the user in terms of interactivity, and an improvement in general to the simulation as it promotes better fine-tuning of an epidemic scenario. These parameters could include, but are not limited to, a healthcare system, a governmental body system, and an `Agent` age system. The healthcare system may take the form as a limited number of hospital beds, which would have the benefit of improving the speed at which agents are removed from the simulation, simultaneously decreasing the rate at which new agents are infected. This could be interacted with a slider that determines the maximum amount of hospital beds usable by agents. The governmental body system could be implemented as an alteration of behavior for the agents. Imagine this governmental body enforcing social distancing, or quarantine. If this were the case agents in the simulation would have to spatially distance themselves from other agents in the case of social distancing, or move into clusters in case of quarantine. This behavior is not implementable with the current limitations of `Agent` behavior, but could be given future improvements to the model. Finally, an age system for the agents in the model could allow for some interesting dynamics. Very young or very old agents could be removed from the simulation at faster rates simulating the weakened immune systems of these two groups of people.

Having the ability to compare between different epidemic models would be very interesting to see. As rudimentary as the simulation is in its current state, it would still be important to see how the model stacks up against the more tested mathematical models of SIS and SIR. This tool would be invaluable for future development of ViruSim as it is crucial that we produce simulation results that are at the very least comparable to these older models.

A custom scenario design tool would also be intriguing to have for promoting increased interactivity and subsequently user retention. This tool would allow users to paint boundaries or walls in the simulation space. These boundaries would block infections from passing between two agents on both sides of the boundary, leading to an overall change in the spread of an epidemic. What's interesting about this tool is that it could be implemented with the current cellular automata model of the simulation.

Finally, the free-roaming implementation we wanted for our original simulation is of key importance for future work. Not only does it more closely mimic the movements of humans, but it allows for the development of the previously mentioned future work. Having free-roaming agents could also contribute to the overall user experience as seeing agents bounce around is a much more interesting display when compared to a grid of position-bound agents.

9 PROJECT ROLES

The following section describes what each member of the group contributed to the project.

9.1 Alexander D. Christoforides

Alexander was primarily responsible for developing the user interface and extracting relevant data from user interactions in order to setup the simulation with the appropriate parameters which have been selected. Additionally, Alex worked on modularizing the user interface code to allow for easy creation of UI panels with header text. Moreover, he was tasked with integrating ThorPy into the application so that the various sliders, text boxes, and buttons were functional and synced with the locally-stored parameters for the simulation. Lastly, Alex primarily merged both halves of the code (from Alex and Chris) and put the finishing touches on the application so that the debug color scheme used for the UI was no longer present and a new, attractive color scheme was instead employed.

9.2 Chris Dicovskiy

Chris was responsible for the simulation side of programming, the overall implementation of the simulation model using PyGame. To this extent, he was tasked with learning enough PyGame to get a working simulation up and running. When working on the original free-roaming model, he wrote a significant amount of code that dictated the movement patterns of agents, only to have such work wasted when he was faced with difficulties regarding PyGame collision detection. After this hiccup, he promptly set out to build the humbler cellular automata model, that ViruSim currently has. This includes all data structures and simulation logic that are relevant to the running of the model.

9.3 Both

Both Alex and Chris were responsible for coming up with the initial design for the application, creating the in-class presentation, and writing this paper. Additionally, they both worked together to debug frustrating bugs and helped each other develop easily maintainable and extendable code so that if future work were to happen on the project, it would not be too difficult to make the model more complex or update the UI to include more views and interactive features.

ACKNOWLEDGMENTS

To Barb Cutler and everyone else who provided invaluable feedback and critiques to help make our project more focused and be the success that it is. Also to Yann Thorimbert for creating the ThorPy (http://thorpy.org) GUI library with easy integration with PyGame.

REFERENCES

[1] 3Blue1Brown. 2020. Simulating an Epidemic. https://www.youtube.com/watch?v=gxAaO2rsdIs

[2] Julie Bosman, Sabrina Tavernise, and Mike Baker. 2020. Most People Back Stay-at-Home Orders. Here's Why Some Are Protesting Them. https://www.nytimes.com/2020/04/23/us/coronavirus-protesters.html

[3] Stephen Eubank, Hasan Guclu, V. S. Anil Kumar, Madhav V. Marathe, Aravind Srinivasan, Zoltán Toroczkai, and Nan Wang. 2004. Modelling disease outbreaks in realistic urban social networks. *Nature* 429, 6988 (2004), 180–184. https://doi.org/10.1038/nature02541

[4] Herbert W. Hethcote. 1989. *Three Basic Epidemiological Models.* Springer Berlin Heidelberg, Berlin, Heidelberg, 119–144. https://doi.org/10.1007/978-3-642-61317-3_5

[5] Nathaniel Lash and Gus Wezerek. 2020. Why Georgia Isn't Ready to Reopen, in Charts. https://www.nytimes.com/interactive/2020/04/24/opinion/coronavirus-covid-19-georgia-reopen.html

[6] Padmavathi Patlolla, Vandana Gunupudi, Armin R. Mikler, and Roy T. Jacob. 2006. Agent-Based Simulation Tools in Computational Epidemiology. In *Innovative Internet Community Systems*, Thomas Böhme, Victor M. Larios Rosillo, Helena Unger, and Herwig Unger (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 212–223.

[7] Harry Stephens. 2020. These simulations show how to flatten the coronavirus growth curve. https://www.washingtonpost.com/graphics/2020/world/corona-simulator/