UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Development of Virus Dispersion Simulation

Diploma Thesis

Ziogas Dimitrios

Supervisor: Tsalapata Hariklia

Volos 2023

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Development of Virus Dispersion Simulation

Diploma Thesis

Ziogas Dimitrios

Supervisor: Tsalapata Hariklia

Volos 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και Ανάπτυξη Προσομοίωσης Διασποράς Ιών

Διπλωματική Εργασία

Ζιώγας Δημήτριος

Επιβλέπων/πουσα: Τσαλαπάτα Χαρίκλεια

Βόλος 2023

Approved by the Examination Committee:

Supervisor **Tsalapata Hariklia**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Member **Stamoulis Georgios**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Daskalopoulou Aspasia**
Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

I would like to thank my supervisor, Dr. Hariklia Tsalapata for her help and valuable assistance in implementing this thesis. Moreover, I would like to thank Dr. Aspassia Daskalopulu and Dr. Georgios Stamoulis for their useful comments and for participating in the examination committee. Last but not least, I would like to thank my family and my friends for always being there for me throughout my studies. Their support played a vital role in composing this thesis.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Ziogas Dimitrios

Diploma Thesis

Design and Development of Virus Dispersion Simulation

Ziogas Dimitrios

Abstract

Nowadays, given the impact of virus outbreaks like COVID-19, there is a growing need for education and understanding of virus dispersion mechanics. The simulation developed in this thesis, using the Unreal Engine, provides valuable insights into the spread of viruses as it allows for customization of parameters and testing of different scenarios, including the impact of vaccines, quarantine measures, and use of masks. The agents in the simulation being AI-controlled and having their own social life and routines adds to its realism. The simulation demonstrates the versatility of agent-based modeling as a tool for simulating complex systems, such as virus spread. With its 3D modeling and game-like appearance, it serves as a unique and engaging educational tool for teaching about the dangers of outbreaks and the importance of preventative measures.

Διπλωματική Εργασία

Σχεδιασμός και Ανάπτυξη Προσομοίωσης Διασποράς Ιών

Ζιώγας Δημήτριος

Περίληψη

Στις μέρες μας, δεδομένου του αντίκτυπου των πανδημιών όπως ο COVID-19, υπάρχει μια αυξανόμενη ανάγκη για εκπαίδευση και κατανόηση των μηχανισμων διασπορας των ιων. Η προσομοίωση που αναπτύχθηκε σε αυτή τη διπλωματική, χρησιμοποιώντας την Unreal Engine, παρέχει πολύτιμες πληροφορίες για την εξάπλωση των ιών καθώς επιτρέπει την προσαρμογή των παραμέτρων και τον έλεγχο διαφορετικών σεναρίων, συμπεριλαμβανομένων των επιπτώσεων των εμβολίων, των μέτρων καραντίνας και την χρήση μασκών. Οι πράκτορες στην προσομοίωση ελεγχόμενοι από τεχνητή νοημοσύνη και έχοντας τη δική τους κοινωνική ζωή και ρουτίνες, προσθέτουν στον ρεαλισμό της. Η προσομοίωση καταδεικνύει την ευελιξία της μοντελοποίησης που βασίζεται σε πράκτορες ως εργαλείο για την προσομοίωση πολύπλοκων συστημάτων, όπως η εξάπλωση ιών. Με την τρισδιάστατη μοντελοποίηση και την εμφάνιση που μοιάζει με παιχνίδι, χρησιμεύει ως ένα μοναδικό και ελκυστικό εκπαιδευτικό εργαλείο για τη διδασκαλία σχετικά με τους κινδύνους των πανδημιών και τη σημασία των προληπτικών μέτρων.

Table of contents

Abbreviations

AI	Artificial Intelligence
UE4	Unreal Engine 4
BT	Behavior Tree
EQS	Environment Query System
GPU	Graphics Processing Unit
CPU	Central Processing Unit
ABMs	Agent Based Models
HUD	Heads-Up Display
SIR	Susceptible-Infected-Recovered
SIS	Susceptible-Infected-Susceptible
SEIR	Susceptible-Exposed-Infected-Recovered

List of figures

Chapter 1

Introduction

The recent outbreak of COVID-19, caused by the novel coronavirus, has had a significant impact on society around the world. The virus has caused widespread illness and death, and efforts to contain it have led to significant disruptions in daily life. In this thesis, we will explore how computer simulations and serious games can be used to better understand and mitigate the impact of the pandemic.

Simulations can be used to create models that can predict the spread of the virus and the effectiveness of different containment measures. These models can provide valuable insights into the potential impact of different scenarios and help decision-makers make informed choices about how to respond to the crisis. Additionally, simulations can be used to test and evaluate the effectiveness of potential treatments and vaccines, which can help accelerate the development and distribution of these important tools in the fight against the virus.

Serious games, also known as educational or learning games, are designed to teach specific skills or knowledge to players while also being entertaining. They have been used in a variety of fields, including education, healthcare, and corporate training, and have been shown to be effective at improving learning outcomes, increasing retention, and increasing engagement. In the context of the COVID-19 pandemic, serious games can be used to train healthcare professionals and others on how to effectively respond to the crisis and protect themselves and others from infection.

1.1 Aim of the thesis

In this thesis, I will not only explore the role that simulations and serious games can play in understanding and responding to the COVID-19 pandemic, but I will also take a hands-on approach by creating my own simulation in Unreal Engine. My simulation will focus on the design and development of a virus dispersion simulation, which can help everyone visualise and better understand the potential impact of different scenarios on the spread of the virus and inform their responses to the crisis.

Using Unreal Engine, I will design and develop a simulation that can be used to explore the dynamics of virus dispersion under different conditions. I will also consider the potential applications of my simulation, including its use in teaching adults and children on how to effectively respond to pandemics and protect themselves and others from infection.It may also be used and improved from epidemiologists in order to become more accurate. At a minimum,it can be considered as a tool to raise awareness.

1.1.1 Contribution

Through this project, I hope to contribute to the ongoing efforts to understand and mitigate the impact of the COVID-19 pandemic and demonstrate the potential of simulations and serious games as tools for improving public health and safety. In addition, the simulation could help individuals to get a better understanding of the potential impact of their own actions on the spread of a virus and the importance of taking personal responsibility in preventing the transmission of diseases. By providing a clear and intuitive visualization of the spread of a virus and the impact of different interventions, the simulation could serve as an effective educational tool for helping individuals to better understand the role they play in preventing the spread of diseases and the importance of taking preventative measures to protect themselves and others.

Chapter 2

Simulation models and applications

2.1 Introduction

There are several applications that simulate the spread of diseases, also known as disease transmission or disease dispersion. These applications are often used by public health officials, epidemiologists, and other professionals to understand how diseases spread and to identify potential control measures.

Among the different types of models used to study infectious diseases, compartmental models are the most widely used. Compartmental models divide the population into different compartments such as susceptible, exposed, infected, and recovered individuals, and use mathematical equations to describe the flow of individuals between compartments.These models often assume that every person has an equal chance of coming into contact with those who are infected. However, this assumption does not take into account that people typically interact mostly within smaller groups. Network models, on the other hand, provide a route into analyzing epidemics in a way that takes these patterns of interaction into account. Recently, however, in the field of epidemiology and disease modeling, there has been a shift towards using agent-based models (ABMs) instead of simpler compartmental models. Agent-based models are a type of computational tool that simulates the behavior of individual agents in a population. These models are used to analyze complex systems, such as the spread of viruses, by representing each agent as an autonomous entity that interacts with other agents and its environment.. [3]

2.2 Various popular models

2.2.1 Compartmental models

Compartmental models that use equations to represent the spread of diseases are often used to understand and predict the dynamics of infectious diseases. These models can take various forms, such as differential equation models.

An example of a differential equation model is the Susceptible-Infected-Recovered (SIR) model, which is commonly used to study the spread of infectious diseases. The SIR model is a generic compartmental model used to describe the spread of infectious diseases in a population and is often run with ordinary differential equations. It stands for Susceptible, Infected, and Recovered or Removed. The model is based on the premise that a population can be divided into three categories:

- Susceptible (S): This group represents individuals who are capable of getting infected with the disease, but are not currently infected.
- Infected (I): This group represents individuals who are currently infected with the disease and are capable of spreading it to others.
- Recovered or Removed (R): This group represents individuals who have either recovered from the disease and gained immunity, or have been removed from the population (e.g. through death).

The SIR model is based on the following assumptions:

- The population is homogeneously mixed, meaning that all individuals have an equal probability of coming into contact with infected individuals.
- The disease is transmitted through direct contact between individuals.
- The rate at which individuals move from the susceptible to the infected group (i.e. the transmission rate) is constant.
- The rate at which individuals recover from the disease or are removed from the population is constant.

The SIR model uses three differential equations to describe the changes in the number of individuals in each group over time. These equations are:

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

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

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

where S, I, and R represent the number of individuals in the susceptible, infected, and recovered/removed groups, respectively and N is the sum of these three; β represents the transmission rate; and γ represents the recovery rate. To use the SIR model to model the spread of a disease, you need to know the initial number of individuals in each group (S_0, I_0, and R_0), as well as the values of β and γ. You can then use these values to solve the differential equations using numerical methods. [4]

One of the key outputs of the SIR model is the reproduction number (R_0), which represents the average number of new cases that one infected individual is expected to generate in a population where everyone is susceptible to the disease. R_0 can be calculated as follows:

$$R_0 = \frac{\beta}{\gamma}$$

If R_0 is greater than 1, the disease will spread; if it is less than 1, the disease will die out. The value of R_0 can be used to predict the potential severity of an outbreak and to guide the implementation of control measures. [4]

The SIR model is useful for understanding the dynamics of infectious diseases and can be used to make predictions about the spread of a disease and the effectiveness of different control measures.This model is suitable for infectious diseases that lead to lifelong immunity, such as measles.There is also the Susceptible-Infected-Susceptible (SIS) model that is used for diseases where repeated infections can happen such as chlamydia. Another example is the SEIR model, which is similar to the SIR model but includes an additional category for individuals who are exposed to the disease but have not yet become infected. This model can be used to study the spread of diseases that have an incubation period, during which individuals are contagious but not yet showing symptoms. It is important to note that the SIR model is a simplified representation of disease transmission and does not consider various factors that can affect the spread of the disease, such as demographics, population density, and environmental conditions. [5]

2.2.2 Network-based models

In many real-world situations, the number of contacts that an individual has is considerably smaller than the population size, and random mixing does not occur. This is known as ”assortative mixing,” and it can have a significant impact on the spread of infectious diseases. When assortative mixing occurs, individuals are more likely to come into contact with others who are similar to them in some way, such as having the same age, occupation, or lifestyle. This can lead to the spread of infectious diseases within certain subgroups of the population, rather than evenly across the whole population.

To account for the impact of assortative mixing on the spread of infectious diseases, more complex models, such as network-based models, may be used. These models represent the interactions between individuals in a population as a network, rather than assuming random mixing. Network-based models can be more accurate in predicting the spread of infectious diseases, but they can also be more difficult to analyze and interpret. They represent the spread of diseases through social or other types of networks. These models can be used to study how infectious diseases spread through networks of contacts, such as social networks or networks of transportation. [5] [3]

One example of a network model for studying the spread of diseases is a model that represents the spread of a disease through a social network, such as a network of friends or colleagues. The model might consider the connections between individuals and the frequency of their interactions, as well as the likelihood that an infection will spread from one individual to another based on these interactions.

Another example of a network model for studying the spread of diseases is a model that represents the spread of a disease through a transportation network, such as an airline network. The model might consider the connections between different cities or locations and the frequency of travel between these locations, as well as the likelihood that an infection will spread from one location to another based on these connections.

We can create a simple model using an ‘adjacency matrix’, A, where $A_{ij}=1$ if there is a possible contact between individual i and individual j and $A_{ij}=0$ if there is not. In this case, A^m provides information for the networks of length m in the matrix. For example, the element $A^m{}_{ij}$ shows how many paths of length m exist between i and j. If we add matrices A, A^2,..., A^m we get a matrix B, where B_{ij} equals the number of possible ways to connect individual i with individual j using paths where the maximum length equals to m. This means that, when $B_{ij}=0$,

there is no possible way for individual j to get infected from individual i. Through network theory we can come to many useful conclusions, however there are some disadvantages using this approach such as:

- Complexity: Network-based models can be complex to build and require a lot of data and computing power. This can make it difficult to use them in real-time or in resource-limited settings.

- Limited data: Network-based models rely on data about the connections between individuals in a population, which may be difficult to obtain or incomplete. This can lead to inaccurate or incomplete predictions.

- Assumptions: Network-based models often make assumptions about the behavior of individuals in the population, such as how likely they are to transmit the virus to others. These assumptions may not always be accurate, which can affect the accuracy of the model.

Knowledge of every individual in a population and every relationship between individuals is required, in order to determine a complete mixing network. Furthermore, this model is not well suited for high uncertainty between contact and actual transmission of the disease. For instance, network-based models are well-suited for analyzing the spread of sexually transmitted diseases (STDs) because they can take into account the specific patterns of sexual contact between individuals in a population. This type of model can simulate the transmission of the disease from one person to another through sexual contact, taking into account factors such as the number and type of sexual partners an individual has, and the frequency of sexual encounters. [5]

On the other hand, network-based models may not be as well-suited for analyzing the spread of airborne diseases, such as influenza or COVID-19. These types of diseases can be transmitted through the air by respiratory droplets, and can be transmitted by individuals who are not in close contact with one another.

2.2.3 Agent-based models

An agent-based model (ABM) is a type of computer simulation that represents the behavior and interactions of autonomous agents within a system. Agents are individual entities

within the model that can make decisions, take actions, and interact with each other and their environment. ABMs are commonly used in a wide range of fields, including economics, biology, sociology, and engineering, to study complex systems and to make predictions about their behavior. [6]

One of the key advantages of ABMs is that they allow researchers to model and analyze the behavior of individual agents within a system, rather than just considering the system as a whole. This can provide a more detailed and nuanced understanding of how a system operates and how it may change over time. ABMs are typically built using a set of rules that specify how the agents in the model behave and interact. These rules can be based on real-world data or be the result of assumptions made by the modeler. The model is then run using a computer to simulate the behavior of the agents over time. By analyzing the output of the simulation, researchers can gain insights into the behavior of the system and make predictions about how it may change in the future. [6]

ABMs have some limitations, however. One major limitation is that they are often based on simplifying assumptions and may not accurately capture the complexity of real-world systems. Additionally, ABMs require a significant amount of computational power and can be time-consuming to run, which can limit their scalability. Finally, ABMs are only as accurate as the data and assumptions used to build them, so the quality of the model's output depends on the accuracy of these inputs. [7]

One example of an agent-based model for studying the spread of diseases is a model that represents the behavior of individuals in a population as they make decisions about whether to get vaccinated, wear masks, or seek medical care. The model might simulate the interactions between individuals, such as how they decide whether to follow public health recommendations, and how these interactions affect the spread of the disease. Another example of an agent-based model for studying the spread of diseases is a model that represents the behavior of individuals as they travel between different locations and interact with other people. This type of model might be used to study the spread of diseases through transportation networks, such as airline flights or shipping routes.

2.3 Conclusion

While SIR models may be sufficient for large populations with homogeneous transition probabilities, the increasing computational power of hardware has made it more practical to perform large-scale agent-based simulations that take into account individual heterogeneity and random events. Network-based models, which represent the spread of the disease through a network of interconnected individuals, can also be used to model the spread of infectious diseases. However, these models may not be as well-suited to capturing the complexity of disease transmission in certain situations, such as in the early stages of an emerging infectious disease.

Agent-based models are often preferred over deterministic models, due to their ability to capture individual heterogeneity and random events in the spread of an infectious disease. While deterministic models may be sufficient for large populations with homogeneous transition probabilities, agent-based models can provide a more realistic representation of disease transmission in situations where individual behavior and interactions are important.Therefore, in this thesis, I chose to develop an agent-based simulation that is presented as a serious game. In that way, it can serve as a tool for a broader audience to play and test different scenarios, gaining useful knowledge about the ways a virus can spread and what actions can protect them from getting infected.

Chapter 3

Existing agent-based applications

3.1 Introduction

Given the usefulness of agent-based models for understanding the spread of infectious diseases, there are several existing applications that utilize this approach. Disease spread simulation applications can vary in their features, capabilities, and intended uses.The majority of them follow a variance of an SIR model. In this chapter, we will take a closer look at some of these existing agent-based applications and explore their capabilities and limitations.

3.2 Netlogo Virus model

NetLogo is a multi-agent, agent-based modeling (ABM) platform. It allows simulating the behavior of a population of agents and how their interactions lead to the emergence of aggregate-level phenomena. NetLogo is a user-friendly, graphical programming environment that allows users to create and run agent-based models. It provides a simple programming language based on the Logo programming language and a wide range of built-in tools for visualizing and analyzing the results of simulations. [8]

The model shown is the Virus model from the NetLogo library. It simulates the spread of a virus in a population. It begins with 150 people, 10 of whom are infected. The individuals can be in one of three states: healthy and susceptible to infection, sick and infectious, or healthy and immune. The population density, population turnover, degree of immunity, infectiousness, and duration of infectiousness are all factors that affect the spread of the virus in the population. The model allows users to adjust various parameters, such as population size

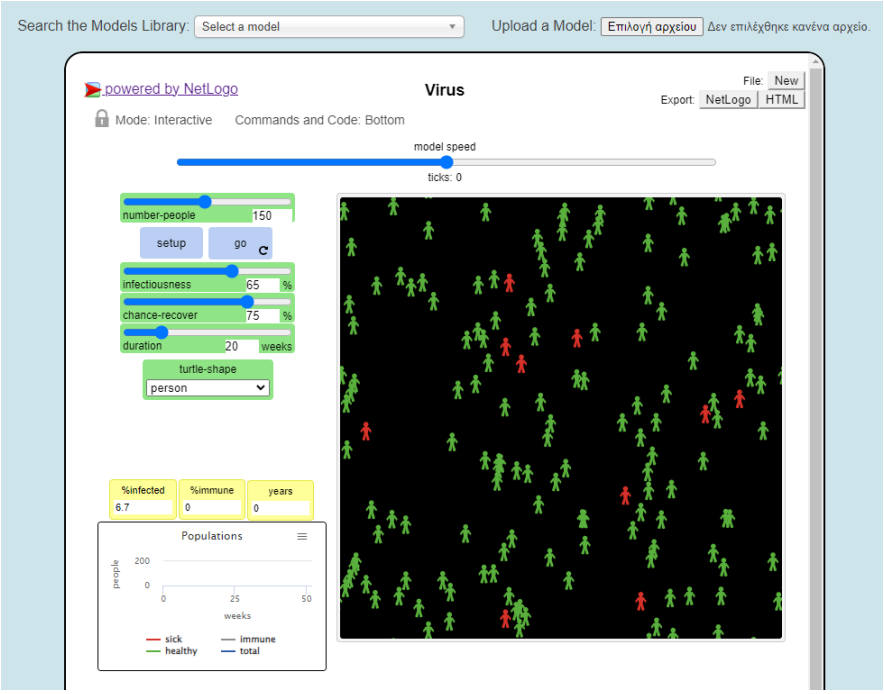


Figure 3.1: NetLogo web application

and the chance of recovery, to see how these factors affect the spread of the virus over time. The model also includes a plot that shows the number of susceptible, infected, and immune individuals in the population over time.

In this model, the population density is determined by the NUMBER-PEOPLE slider, which allows you to change the size of the initial population. Population turnover is also an important factor, as individuals die and are replaced by new, susceptible individuals. People may die from the virus or from old age, with the chance of dying from the virus determined by the CHANCE-RECOVER slider. Additionally, the model includes a carrying capacity of 300 individuals, and when the population dips below this number, healthy individuals may produce healthy offspring. The degree of immunity, infectiousness, and duration of infectiousness are also factors that affect the spread of the virus in the population, and can be adjusted using sliders in the model.

The SETUP button in the model allows users to reset the graphics and plots and randomly distribute the number of people in the simulation. The majority of individuals will be set to be healthy and susceptible to infection, while a small number will be set as infected individuals of randomly distributed ages. The GO button starts the simulation and the plotting function. The TURTLE-SHAPE chooser controls whether the people are visualized as person shapes or as circles. Three output monitors show the percentage of the population that is infected,

the percentage that is immune, and the number of years that have passed. The plot shows the number of susceptible, infected, and immune individuals in the population over time, as well as the total population in blue.

The factors controlled by the sliders interact to influence how likely the virus is to spread in the population. It is important to notice that these factors must create a balance in which an adequate number of potential hosts remain available to the virus and in which the virus can adequately access those hosts. Initially, there may be an explosion of infection since no one in the population is immune, similar to an outbreak of a viral infection in a population. However, as the population dynamics change, the virus may become less common. The ultimate outcome of the virus is determined by the factors controlled by the sliders. Additionally, viruses that are too successful at first may not survive in the long term since everyone infected generally dies or becomes immune, limiting the potential number of hosts. However, if the duration of infection is set to be high, population turnover (reproduction) can provide new hosts. [8]

3.3 Epidemic Simulation

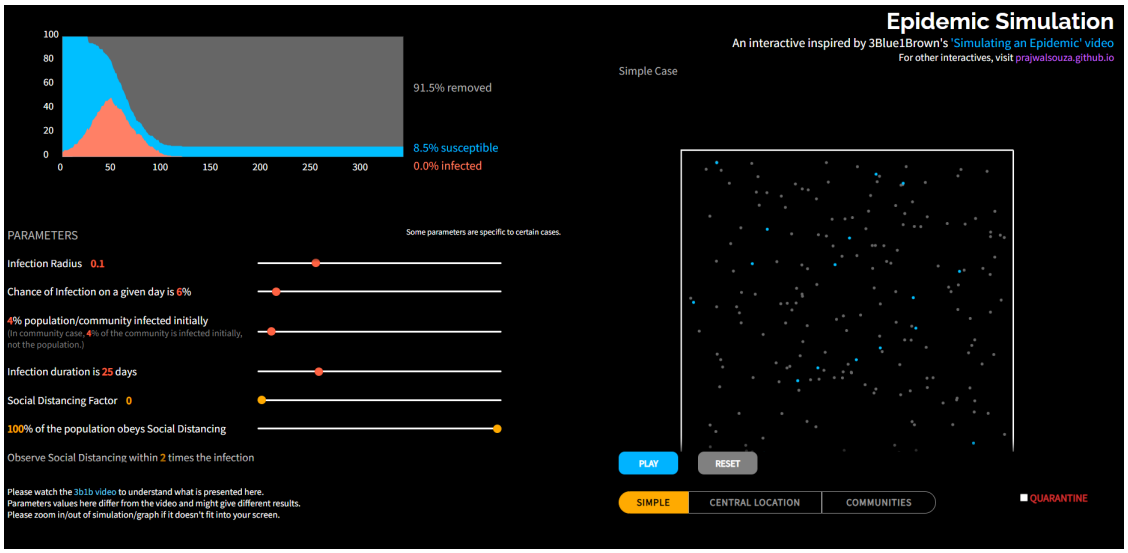


Figure 3.2: Epidemic Simulation

Epidemic Simulation is an interactive web application ,inspired by 3Blue1Brown's 'Simulating an Epidemic' video, that simulates the spread of a disease. It uses various parameters to model the spread, including an infection radius, chance of infection on a given day, and initial number of infected individuals in a community. The simulation also incorporates social

distancing measures, quarantine protocols, and the ability for individuals to travel between communities. Additionally, the simulation allows for the visualization of the spread of the disease using particles, with options to adjust the number of particles, particle size, and frame rate. The simulation can be used to study the effectiveness of different interventions and to better understand the dynamics of disease spread. [9]

The simulation also includes a parameter for the percentage of the population that is removed, which includes those who have recovered or died from the disease, and a percentage of the population that is susceptible, which includes those who have not yet been infected.Other important parameters are a social distancing factor, which is set to 0 in this case, indicating that 100% of the population is obeying social distancing measures, a parameter for the distance at which individuals will observe social distancing, which is set to 2 times the infection radius, the quarantine protocol, which begins 10 days after an individual is infected and starts 2 days after the beginning of the epidemic and the percentage of infected individuals who do not show symptoms and will not be quarantined.The simulation also takes into account the movement of individuals between communities, with a parameter for the chance of an individual visiting a central location in any given hour and the chance of an individual traveling to another community on a given day. The simulation is set to initially infect only 2 communities, with 24 individuals per community.

Overall, the Epidemic Simulation web application is a powerful tool for studying the spread of diseases and understanding the impact of different interventions and behaviors on disease transmission. The various parameters allow for the simulation of different scenarios and the visualization of the spread of the disease helps to better understand the dynamics of disease transmission. [9]

3.4 Overall analysis

The NetLogo Virus application provides less flexibility in comparison to the more advanced epidemic simulation model. This is because NetLogo is a general-purpose modeling environment, and while it can be used to create simulations of agents interacting with one another and their environment, it may not have all the parameters and features that are specific to the spread of a disease.

The Epidemic simulation, on the other hand, is typically more detailed and has more

parameters that allow for a more accurate representation of the spread of a disease. This model includes detailed mathematical equations that take into account the biology of the disease, the characteristics of the population, and the effectiveness of different interventions. Epidemic Simulation, in comparison with the Netlogo Virus model, offers some additional features that take into account the movement of individuals between communities, as well as other parameters that affect the spread of the disease. These include the likelihood of an individual visiting a central location within a specific time period, the likelihood of an individual traveling to a different community within a given time frame, the degree of social distancing implemented by individuals, and the likelihood of individuals being placed in quarantine when they are infectious.

However, it's important to note that even with more parameters and features, epidemic simulation models may not be able to accurately mimic the interaction between agents. This is because these models rely on a number of assumptions and simplifications, which can lead to inaccuracies in the results. Additionally, the complexity of the interaction between agents in a real-world population makes it difficult to fully capture in a simulation. In reality, citizens have their own daily routines and social groups which vary vastly from the usual random movement that these simulations are based on. For example, public transport, crowded events like festivals and public centers like malls can play an important role to the spread of an infection, each having their one unique parameters that affect their contribution to the spread of the infection. Therefore, in this thesis, I have created a simulation that closely resembles a small community, that includes houses where families live,a hospital, a mall,a park,a school, public transport and some workplaces and citizens have their own daily life. Furthermore, many variables have been taken into account and many of them are editable by the user. Further analysis will be conducted in the next chapters.

Chapter 4

Unreal Engine

4.1 Introduction to Unreal Engine

Unreal Engine is a game development engine developed by Epic Games. It is a powerful tool that can be used to create a wide variety of interactive experiences, from video games to virtual reality applications and architectural visualizations. Unreal Engine is built on a robust and well-documented set of features and tools that allow developers to create highly detailed and realistic worlds, with advanced lighting, physics, and animation systems. It also includes a wide range of built-in tools for asset creation and management, scripting, and debugging. In addition, Unreal Engine features a powerful scripting language called Blueprints, which allows developers to create logic and controls without the need for traditional programming. C++ can be used to create custom game logic, extend the engine's functionality, and create custom game mechanics. It can also be used to create plugins, new game types, and other custom features that are not available in the engine's built-in scripting language. Additionally, C++ can be used to optimize the performance of a game by allowing developers to write performance-critical code that runs faster than code written in the engine's scripting language. In this chapter, I will focus on the tools that was used in order to create and optimize the virus dispersion simulation. [1]

4.2 User interface

The editor layout of Unreal Engine 4 (UE4) is designed to be customisable and efficient for game development. The main layout includes several important panels and tools that are

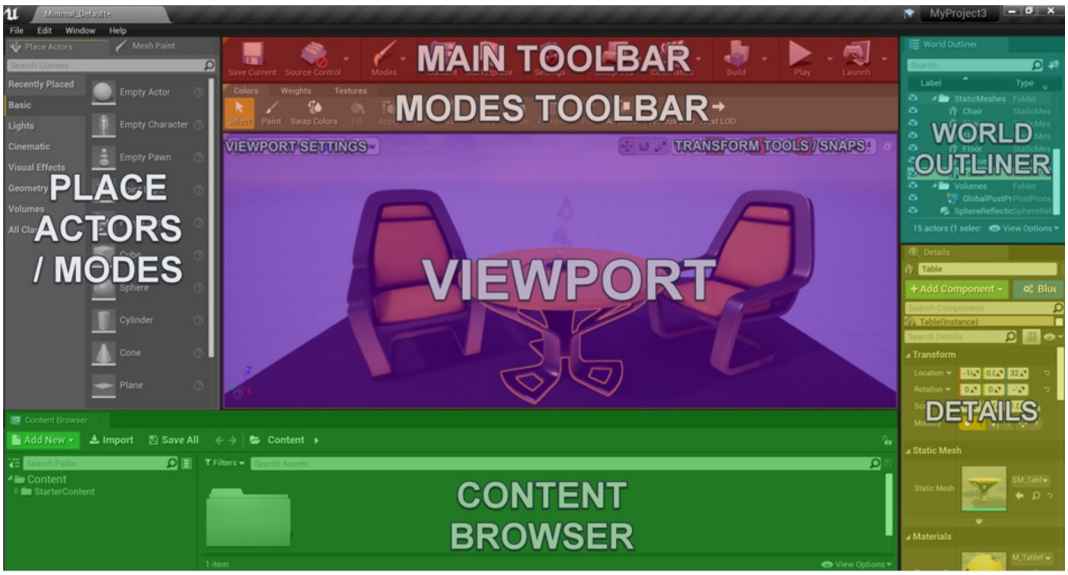


Figure 4.1: Unreal Editor's layout [1]

essential for game development.

- **Unreal Editor viewport**: This is the main area where users can see their level design and interact with their assets in real-time. The viewport allows users to move, rotate and scale objects, place actors, and create and edit terrain. It also supports multiple view modes such as wireframe, solid, and lit.

- **Content Browser**: This panel is used to import, manage, and organize assets. Users can preview and search for assets, create folders to organize their assets, and drag and drop assets into the level.

- **Place Actors/Modes**:It is a convenient tool for quickly placing actors in the scene. The panel allows users to search for actors by name and preview them before placing them in the scene. Users can also filter the actors by type, such as static meshes, lights, or character blueprints, and can set properties such as the actor's location and rotation.

- **Actor Details panel**: This panel displays and allows for editing of the properties of selected actors in the scene. It provides a wide range of options for fine-tuning the behavior of actors such as physics properties, collision settings, and spawning parameters.

- **Toolbar**:This panel is at the top of the screen that provides quick access to frequently used tools and commands. The toolbar contains buttons for common actions such as

saving the level, playing and pausing the game, and switching between different view modes. It also includes buttons for various tools such as the transform tool for moving and rotating actors, the brush tool for sculpting terrain, and the paint tool for painting materials on surfaces.

- **World Outliner**: This panel shows the hierarchy of actors in the scene, it's a convenient way to navigate the level and select objects.

Other panels such as the Material Editor, the Lighting and Animation tools, provide additional functionality for game development such as creating and editing lights, creating and editing animations and realistic and custom materials.Users can arrange and customize the layout of these panels to suit their workflow, it's possible to dock panels, create new tabs, or even create new work spaces with different layouts to optimize the workflow. [1]

4.3 Blueprint system

Unreal Engine's Blueprint system is a visual scripting tool that allows users to create and manipulate game logic without the need to write code. It allows designers and artists to create gameplay mechanics, interactive objects, and entire levels without the need for programming. The Blueprint system is based on a node-based visual scripting interface, which allows users to create scripts by connecting pre-built nodes together in a flowchart-like structure. This makes it easy to create and understand the logic of a script, even for those without any programming experience. Additionally, the Blueprint system is fully integrated with the rest of the Unreal Engine, allowing users to easily access and manipulate assets and objects in the game world.The most common Blueprint types you will be working with are Level Blueprints and Blueprint Classes. [1]

4.3.1 Level blueprint

The Level Blueprint fills the same role that Kismet did in Unreal Engine 3, and has the same capabilities. Each level has its own Level Blueprint, and this can reference and manipulate actors within the level, control cinematics using Matinee Actors, and manage things like level streaming, checkpoints, and other level-related systems. The level blueprint can also interact with blueprint Classes placed in the level, such as reading/setting any variables or

triggering custom events they might contain. [1]

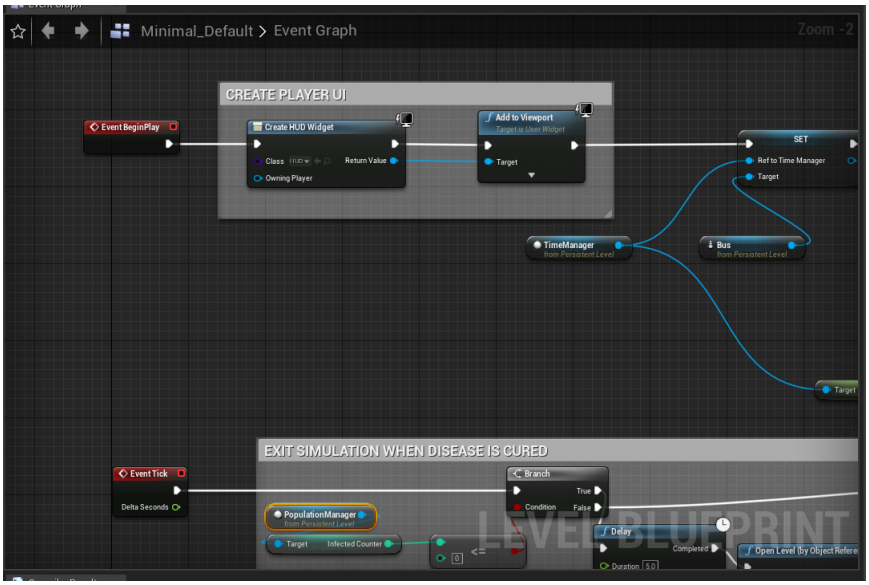


Figure 4.2: Level Blueprint(Player UI creation and level ending)

4.3.2 Blueprint class

Blueprint classes are ideal for making interactive assets such as doors, switches, collectible items, and destructible scenery. They contain the necessary script to respond to player overlap events, make them animate, play sound effects, and change their materials. Because of the self-contained nature of Blueprints, they can be constructed in such a way that you can drop them into a level and they will simply work, with minimal setup required. This also means that editing a blueprint that is in use throughout a project will update every instance of it. Some of its key functionalities are: [1]

- **Event-Driven Logic:** Blueprints can respond to in-game events, such as player input or collisions, to trigger custom behavior.
- **Variables:** Blueprints can store data in variables, which can be used to control game behavior and store information.
- **Functions:** Blueprints can define custom functions that can be reused throughout the game.
- **Components:** Blueprints can be used to create new game objects and components that can be added to actors.

- **Animation:** Blueprints can control animation sequences and blend animations based on game events.
- **User Interface:** Blueprints can be used to create custom user interfaces and menus.
- **Multiplayer:** Blueprints can be used to create multiplayer games, including client-server communication and replication.

In short, Blueprints in UE4 provide a powerful and flexible way to create game mechanics and interactions without writing code.

Examples of what you can create using Blueprints include:

- A playable game character
- HUD
- Interactive objects such as doors, switches, and collectibles
- Destructible scenery and other

Whether you are building a Level Blueprint or a Blueprint Class, you will be using Blueprint Elements assembled in a Blueprint Editor.

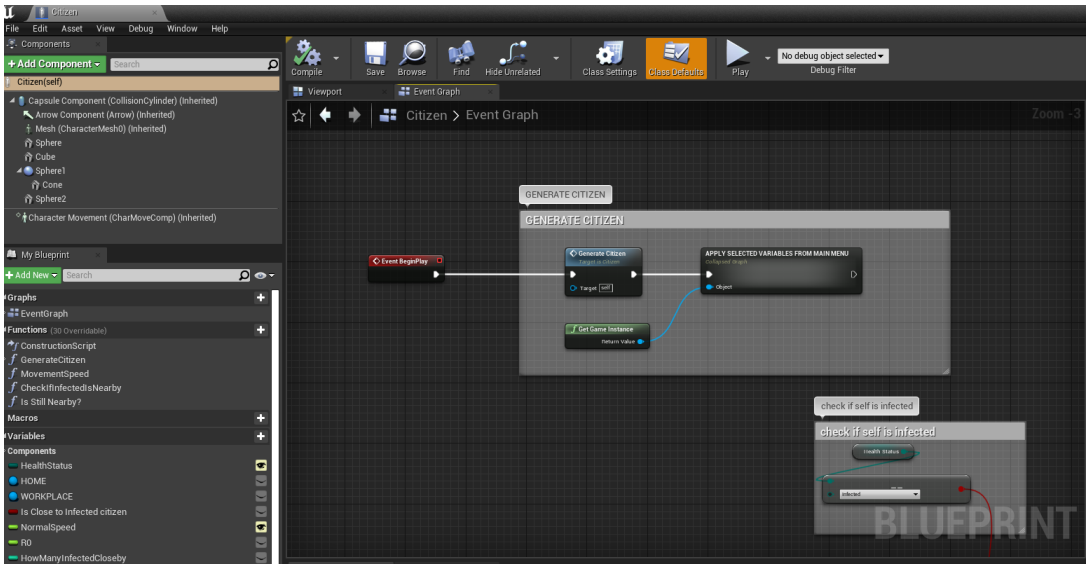


Figure 4.3: Blueprint Editor

4.4 AI system

Creating believable and responsive Artificial Intelligence (AI) for characters or entities in Unreal Engine 4 (UE4) projects involves utilizing multiple systems that work together seamlessly. The behavior tree system is the foundation for creating decision-making processes for AI characters. It allows developers to create complex, hierarchical AI behaviors using a simple, visual interface. The Environment Query System (EQS) is used to gather information about the environment, such as the location of objects or obstacles, and to evaluate conditions that influence the AI's behavior. The AI Perception system is responsible for processing sensory information, such as sight, sound, or damage, and passing it on to the behavior trees for decision-making. [1]

In UE4, all of these systems are integrated and can be easily debugged using the AI Debugging tools, which provide insight into the AI's thought process and actions at any given moment. This allows developers to fine-tune their AI, ensuring that it behaves in a believable and responsive manner.

4.4.1 Behavior trees

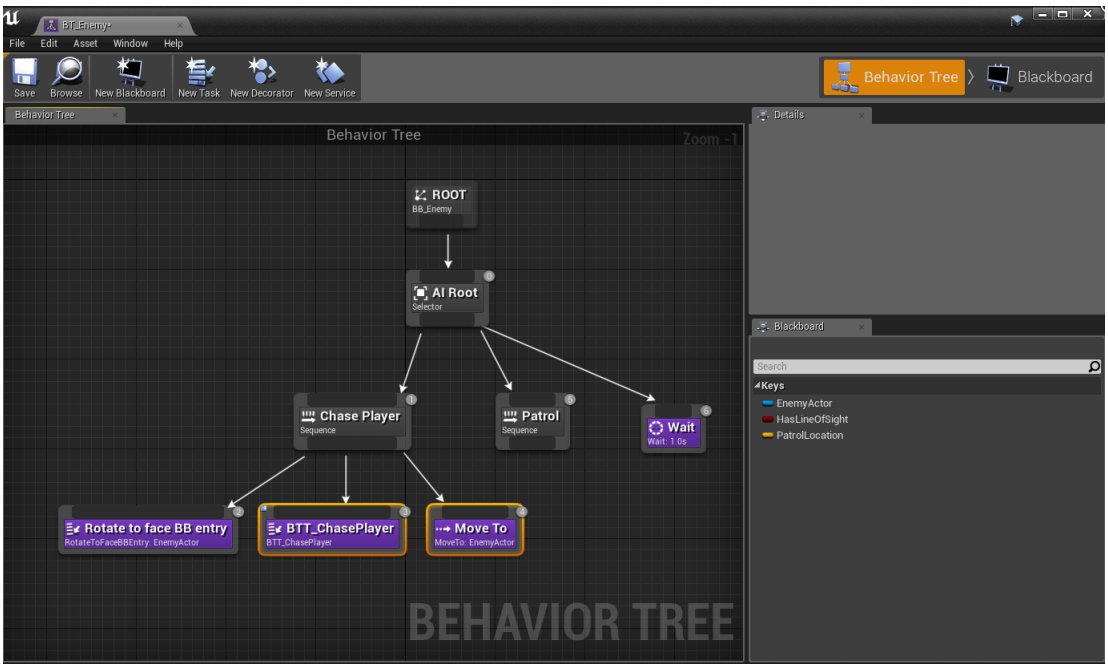


Figure 4.4: Behavior tree [1]

Behavior trees assets in Unreal Engine 4 (UE4) can be used to create artificial intelligence (AI) for non-player characters in your projects. While the behavior tree asset is used to execute

branches containing logic, to determine which branches should be executed, the behavior tree relies on another asset called a Blackboard which serves as the ”brain” for a behavior tree.The blackboard contains several user defined Keys that hold information used by the behavior tree to make decisions. For example, you could have a Boolean Key called Is Light On which the behavior tree can reference to see if the value has changed. If the value is true, it could execute a branch that causes a roach to flee. If it is false, if could execute a different branch where the roach maybe moves randomly around the environment. Behavior trees can be as simplistic as the roach example given, or as complex as simulating another human player in a multiplayer game that finds cover, shoots at players, and looks for item pickups. [1]

4.4.2 Navigation system

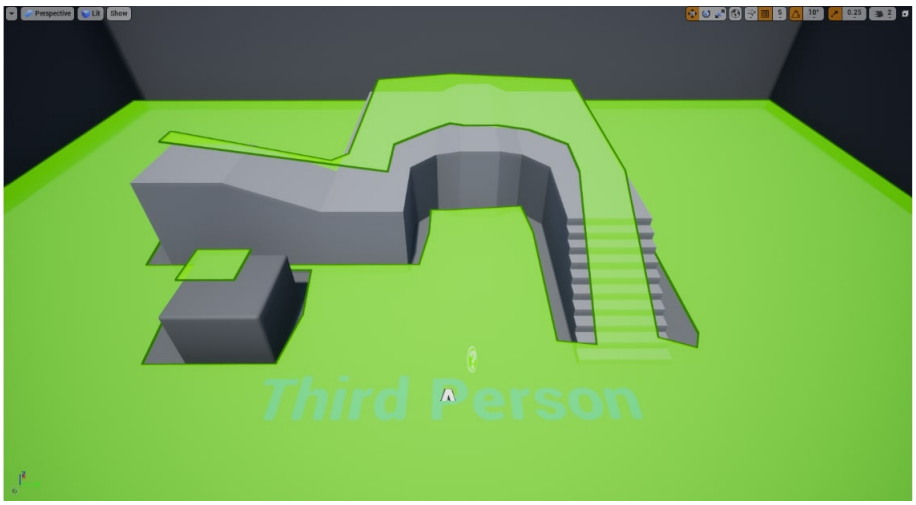


Figure 4.5: Navigation mesh [1]

The Unreal Engine navigation system enables game AI characters to move around the game environment by using path-finding. The system generates a Navigation Mesh, which is a map of the game environment divided into tiles and polygons. These polygons form a graph that the AI characters use to find the most optimal path to their destination. The navigation mesh is generated using the collision geometry of the game environment and each polygon is assigned a cost for the AI characters to consider when determining their path. The system also allows for modification of the Navigation Mesh generation process, such as connecting non-contiguous areas and adjusting the cost assigned to polygons. The navigation system has three generation modes: static, dynamic, and dynamic modifiers only, providing options to suit the needs of the project. [1]

4.4.3 EQS system

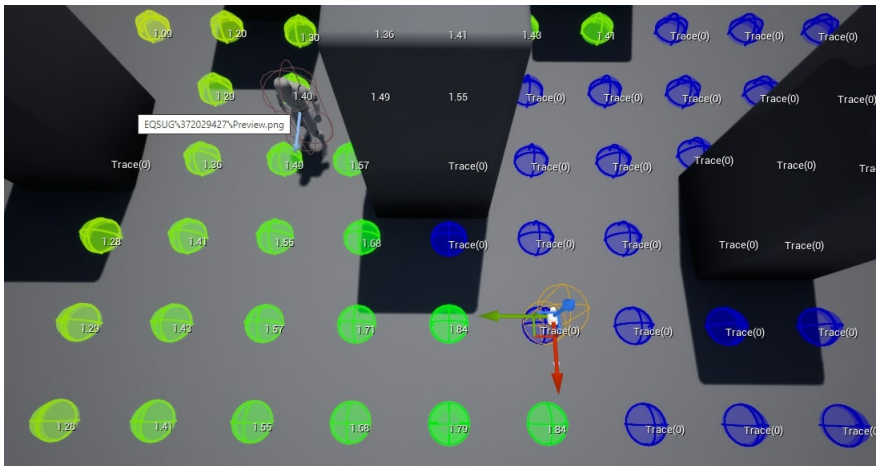


Figure 4.6: EQS system [1]

The Environment Query System (EQS) in Unreal Engine 4 is a tool for Artificial Intelligence (AI) that allows for the collection of data from the game environment. EQS allows for asking questions about the collected data through a variety of different tests, which produces the best fitting item for the type of question asked. EQS queries can be used in a behavior tree to make decisions on how the AI should proceed, based on the results of the tests. EQS queries can be used to direct AI characters to find the best location for attacking a player by having a line of sight, finding the nearest health or ammo pickup, or locating the closest cover point among other possibilities. [1]

4.4.4 AI perception system

The AI perception system is another tool within the AI framework in Unreal Engine 4 that provides sensory information for AI characters. This system allows pawns (AI characters) to gather information about their environment, such as the location of noises, if they have been damaged, or if they can see something. This is achieved through the AI perception component, which acts as a listener for stimuli and collects information from registered Stimuli Sources. The AI perception system allows for more realistic and dynamic interactions between AI characters and their environment, as they can react to stimuli in real-time. [1]

Figure 4.7: AI perception (sight and hearing)

4.5 Tools

4.5.1 AI debugging tools

The AI debugging tools allow you to gain a better understanding of the behavior of AI characters in a game or application. The information is split into behavior tree information, blackboard information, EQS information and perception system information. The behavior tree information displays the class of behavior tree being used, the currently executed branch of the tree, and the nodes within that branch. This information can be useful for understanding how the AI character is making decisions. The blackboard information displays the Blackboard asset being used, along with any Blackboard keys and their current values. This information can be useful for determining why the AI character is or is not performing an action based on the value of a key. [1]

The Environmental Query System (EQS) debug information display the current Environmental Query that is being run and the Generator used. In this way, you can understand the process of how the AI is determining the best location to move based on the grid. The debug visualization shows the points on the grid represented by spheres, with green spheres representing locations that pass the test (has line of sight to the player) and blue spheres representing locations that fail the test (does not have line of sight to the player). The highest weighted value is chosen as the location that the AI character chooses to move to. At runtime,the perception system information shows the position were the player was last perceived by the AI character.These tools can be accessed through the Unreal Editor and can be useful

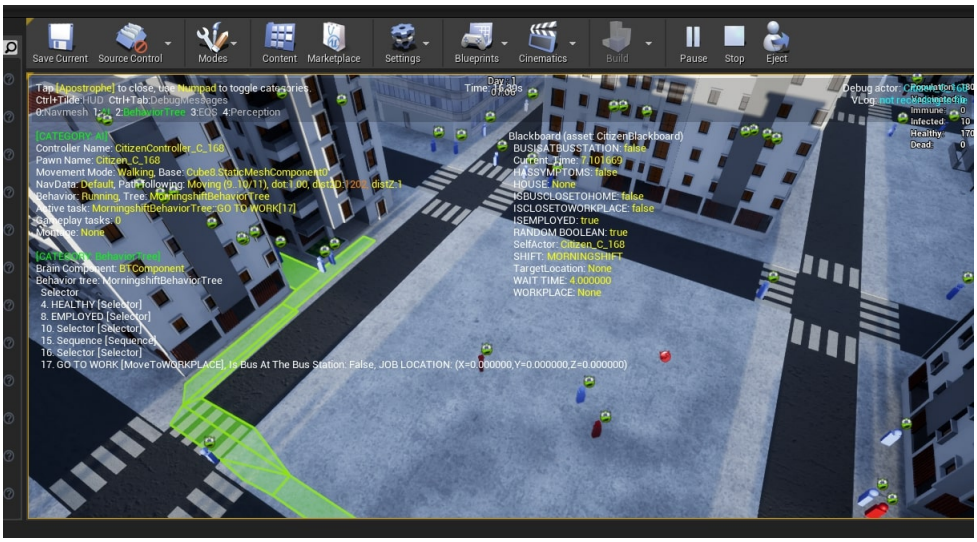


Figure 4.8: AI debugging

for debugging and optimizing AI systems and ensuring that they are behaving as expected. [1]

4.5.2 Performance optimisation tools

Optimizing performance for both the GPU and CPU is crucial in the development of games and applications that require high-quality graphics and smooth performance across different hardware configurations. Unreal Engine provides a comprehensive suite of tools and techniques for optimizing both the GPU and CPU performance.

- For GPU optimization, developers can use techniques such as anti-aliasing, level of detail, asynchronous compute, dynamic resolution scaling, shader optimization, and object culling to strike a balance between visual quality and performance.

- For CPU optimization, Unreal Engine offers techniques such as multithreading, streaming, profiling, optimizing Blueprints, and optimizing code written in C++. These techniques allow developers to minimize the load on the CPU and improve performance, ensuring that their projects run efficiently on a wide range of hardware configurations.

By using these techniques, developers can create games and applications with both visually stunning graphics and smooth performance on a variety of hardware.In this section, 2 main optimisation tools that were used in order to improve the performance, are briefly analyzed.

Shader Complexity Viewmode: Shader complexity is a metric that is used to measure the performance of the shaders in a game or application. It is an indicator of how much work

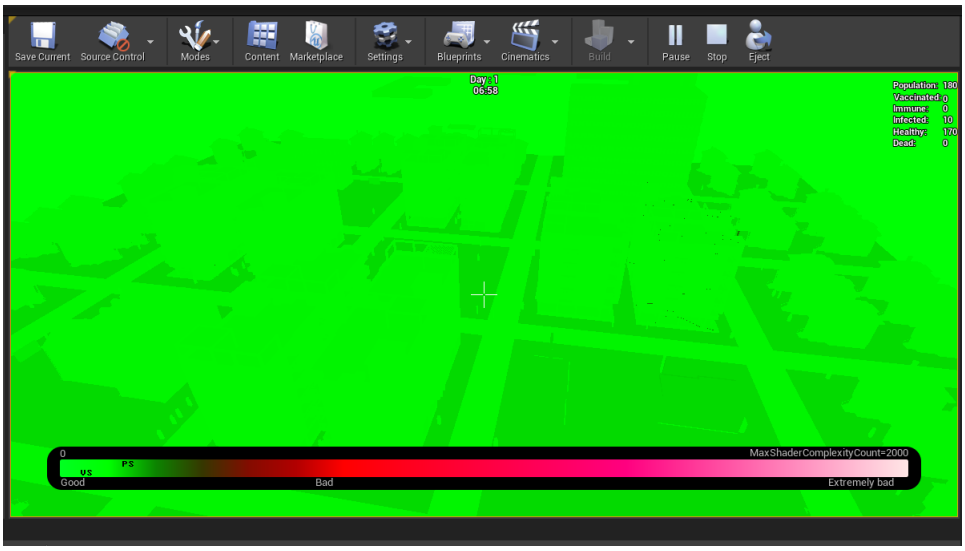


Figure 4.9: Shader Complexity Viewmode

the GPU has to perform to render the scene, and is affected by factors such as the number of texture samples, the number of instructions in the shader, and the number of lights that are affecting a surface.The Shader Complexity Viewmode is a visualisation mode that allows you to see the relative cost of different materials in the scene, this can help you identify which materials are the most expensive to render and make optimization decisions. [1]

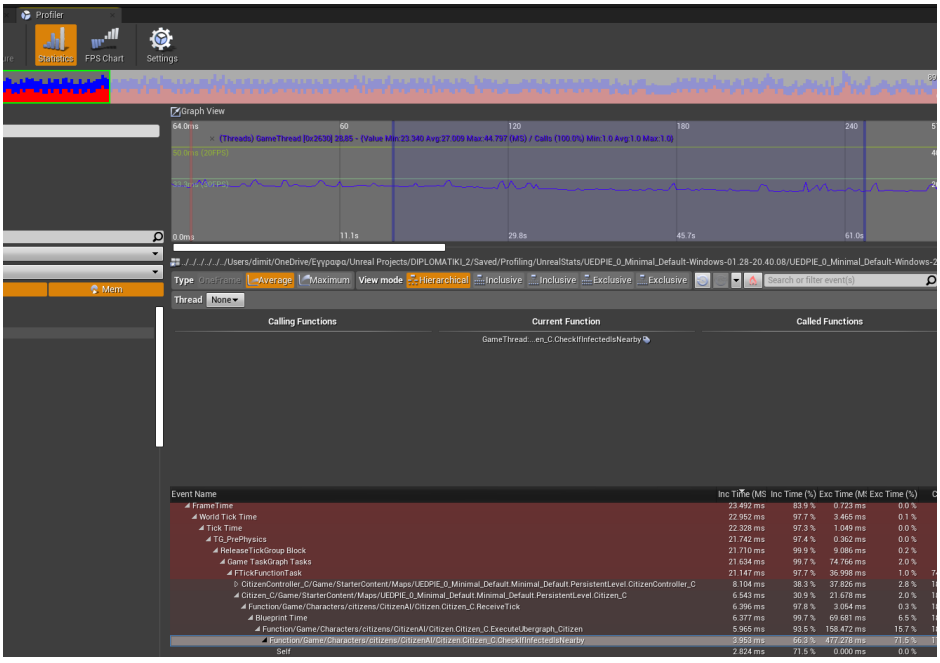


Figure 4.10: Profiler

Profiler: It is a tool that can be used to measure the performance of a game or application. It can track various aspects of the engine's performance, including CPU usage, GPU usage,

memory usage, and more. The profiler can also be used to track specific areas of the code, such as the performance of individual actors or components. The information gathered by the profiler can be used to identify bottlenecks in the performance of the game or application, which can then be optimized to improve overall performance. The profiler can be accessed through the Unreal Editor, and the data can be viewed in various formats, including graphs, tables, and call stacks. [1]

4.6 Why Unreal Engine

Advantages of Using Unreal Engine:

- **Large Community:** The Unreal Engine has a large and active community of developers, which makes it easy to find help and resources for creating simulations. This community also creates and shares free assets, such as 3D models and textures, which can be used in a virus dispersion simulation.
- **Free assets:** The Unreal Engine has a large library of free assets that can be used in a simulation, such as 3D models of buildings and landscapes, textures, and animations. These assets can be used to create a realistic environment for the simulation, which can help the user better visualize the spread of the disease.
- **Scripting with blueprints:** The Unreal Engine uses a visual scripting language called Blueprints, which makes it easy to create and edit scripts for the simulation. This can be beneficial in a virus dispersion simulation as it allows the user to quickly test different interventions and scenarios.
- **AI system:** The Unreal Engine has a built-in AI system that can be used to create agents that can move and interact with each other. In a virus dispersion simulation, this can be used to model the movement and interactions of individuals, which can affect the spread of the disease.

In conclusion, the Unreal Engine is a powerful tool for creating 3D agent-based virus dispersion simulations. Its large community, free assets, scripting capabilities with Blueprints, and built-in AI system provide many advantages for creating realistic and efficient simulations. It should be also stated that the version used to create the simulation is 4.27 [1]

Chapter 5

Virus Dispersion Simulation

5.1 Simulation overview

In this thesis I chose to develop an agent-based simulation that simulates the spread of an infectious disease in a small town. The purpose of this simulation is to explore and understand the factors that contribute to the spread of the disease and the impact of various preventive measures. By examining different parameters of the disease and observing how they affect the spread, I hope that it will provide an insight into the dynamics of infectious disease outbreaks. Citizens have a daily routine which can include going to work or school, shopping at the mall, hanging out with friends, visit other houses and other things that will be analysed further in later sections. Through simulating a town, as close to reality as possible, we can better study the spread of a virus like Covid-19 and test the effects that different parameters have. In order to create the simulation I used the engine's scripting language called Blueprints. In order to create the necessary AI to control the citizens I used the engine's build in BehaviorTree system while creating my own executable nodes. Futher analysis will be conducted in this chapter.

5.1.1 Initialization

When the application start running the main menu appears to the user. In that menu, the user can choose the value of some parameters of the simulation that will remain constant throughout the time that the simulation is running. These values include:

- **the population of the town**
- **the number of people that are infected from the beginning**

and some parameters of the disease that usually remain constant throughout the outbreak which are:

- **Chance of infection per hour** of close contact, where protective measures such as masks are not applied.
- **Average period of infection**, in days, meaning the average number of days it takes for an infected person to become non infectious to others and immune.
- **Average asymptomatic period**, in days, which is the time period from when a person gets infected to when the symptoms show up, if a person is not asymptomatic.
- **Chance of not showing symptoms**, which is the chance of an infected person being asymptomatic.
- **Death rate**, which is the percentage of infected people dying from the infection.
- **Chance of immune getting infected**, which is the chance of someone vaccinated or cured from the disease, contracting the disease again relative to the chance of an susceptible person getting infected. A value of 0,1 means there is 10 times less chance for a Recovered individual to get infected than a Susceptible individual.

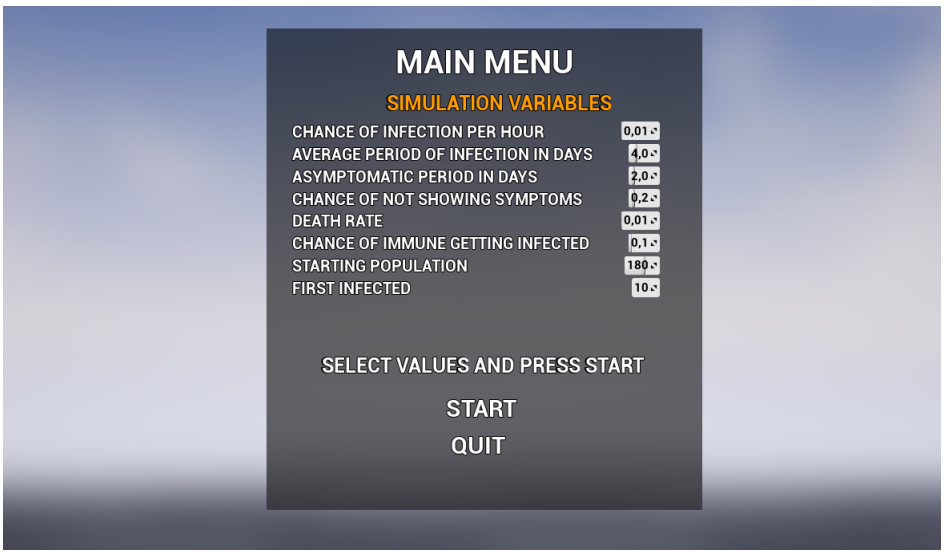


Figure 5.1: Main Menu

For each option there is a slider which can be used in order to adjust its value in a range or else, the user can enter the value of each parameter typing it directly into the box. When

all values are set the user can press start in order for the Simulation to begin and the main level will be loaded.

The simulation starts paused and the pause menu appears. In the menu you can activate or deactivate safety measures in order to test how these measures affect the spread of the disease.The difference is that the values on the pause menu can be altered at any point of the simulation. In that way, a more flexible approach is taken giving the user the freedom to study how certain parameters or measures affect the outcome when are being applied early or later in the progression of the infection.

Figure 5.2: Pause Menu

In particular, those parameters include:

- Vaccinate a certain percentage of the susceptible population.
- Usage of masks inside the bus(either true or false).
- Wearing masks at the mall.
- Wearing masks at the workplace.
- Wearing masks at free time(when meeting friends or go shopping).
- Wearing masks at school.
- Enabling or disabling Public transport.
- Working from home.

- Schools are open or closed.
- Shops are open or closed.
- Social gatherings are allowed or not.

Most of these measures were actually implemented during Covid-19 in order to reduce the spread of the virus. Wearing a mask as well as social distancing can reduce the transmission rate. However some measures have greater effect than others and sometimes it depends on the existing conditions. It should be stated that in this simulation some of these measures are immediately applied and some others take effect after midnight. In addition, some logical exception have been made, for example, workers at the hospital can't work remotely.

The user can alter the default options and then start the simulation by pressing "P" or "Escape". While the simulation is running, pressing those buttons will pause the simulation and show the pause menu again. In this way, the user can change the values at any point of the simulation. There is also the option to quit the simulation. Alternatively the simulation will end 5 seconds after the last infected citizen is cured or deceased.

5.1.2 Runtime basics

As the simulation starts running citizens spawn at their houses which are randomly selected. This results in some houses having large families and others less people and there is a chance for some houses being empty. Some of them start as infected but are still not infectious until they show symptoms. Everyone can get infected if they remain a certain time in close proximity to an infected person. This possibility increases with every second spend close to an infected, if they are indoors, if they are not wearing a mask etc.

Each individual is given a specific job and shift or they are considered unemployed. Some jobs can have evening-shifts and night-shifts. For example, hospital staff is split into 3 shifts from 7:00 to 15:00, 15:00 to 23:00 and 23:00 to 7:00. Schools are open only at morning from 7:00 to 15:00. People who live close to a bus station but far from their workplace will chose to take the bus if the option is enabled in the pause menu. The mall and office have morning shifts and evening shifts. In their free time individuals can randomly chose to stay at home, visit someone's house, visit the mall, or hang out at the central square. At night, everyone returns home and sleeps until the next day, except from those working night-sift.As

a result of this design, every individual has a circle of people that interacts with most often and interacts with random people less frequently.

There is a day-night cycle implemented. The daily routines of the citizens are closely tied to it. Every hour has 100 seconds in real-time, however every parameter is adjusted accordingly. There are 5 speed options which allow time to flow faster or slower than real time. The main camera used remains unaffected however everything else speeds up accordingly including animations and walking speed. This is very useful for getting faster results with minimal disadvantages.The speed can be set to 1x, 5x, 50x, 100x, 0,5x pressing ’1’, ’2’, ’3’, ’4’ and ’0’ accordingly. The user observes the simulation through a top-down camera controlled by the mouse and the ”WASD” keys. The user can move to different areas, zoom in, and zoom out and tilt up or down within limits. The interior of certain importan buildings can be accesed through keys ’5’, ’6’, ’7’, ’8’. At last, during the simulation there are some numbers visible at all times, which include the day, the time and the SIR numbers.

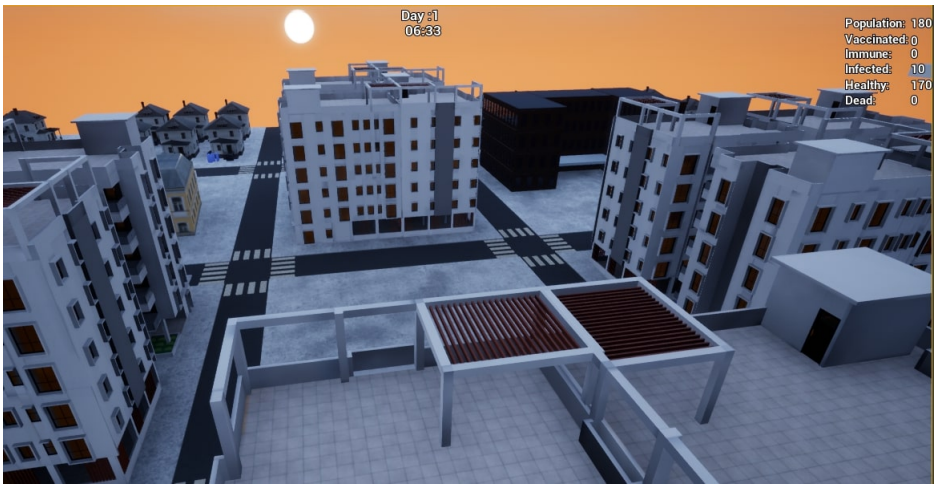


Figure 5.3: Day-Night cycle

5.2 Simulation world

Building the world is an important task. It should include a big enough area with a variety of buildings that are important when simulating a residential area. In addition, there should be areas with less population density and other areas with apartment buildings with many residents. Also it is important to include open public areas where many people can go like a central square and a mall, and also include public transport where many people are gathered together in tight close spaces for a short amount of time.

In this simulation the navigable area is about 25.000 m^2 or 2,5% of 1km^2. Given a normal greek city population density, there should be 100 to 200 residents in that area. For example, the urban population density of Thessaloniki is approximately 7100 residents per km^2 meaning 177 residents per 25.000 m^2. [10] Of course, the user has the ability to test different population densities and compare the results but the default value is set to 180.

Figure 5.4: World Overview

The area is divided in 25 city blocks. In the center there is the central square. The mall, as well as the hospital, the school, the office building and some apartment buildings can be found around the central square. There are single-family houses that are located on the outer regions of the town with a smaller population density. Every block has sidewalks and is connected to other blocks through pedestrian crossings that the residents can use to walk from one area to another.This creates some finite paths that people can take and differs from other agent-based simulations where the agents are moving randomly in a specific area. In this section will take a more detailed look to the areas of this world. Every building model was imported from Unreal Engine free 3d assets store. Citizens and the rest assets where created inside Unreal Engine.

5.2.1 Central square

The central square is located in the middle of the map, and plays an important role for the spread of the disease as high traffic area. This is the place where many residents hang out and others pass through in order to get to their destination. The high traffic increases the risk of virus spread because it leads to increased human interaction and close contact. Crowded places make it easier for the virus to spread from person to person through droplets expelled while speaking, coughing, or sneezing. As a result, the virus can spread quickly in high-traffic areas, especially if preventive measures such as wearing masks, practicing physical distancing, and frequent hand washing are not followed.

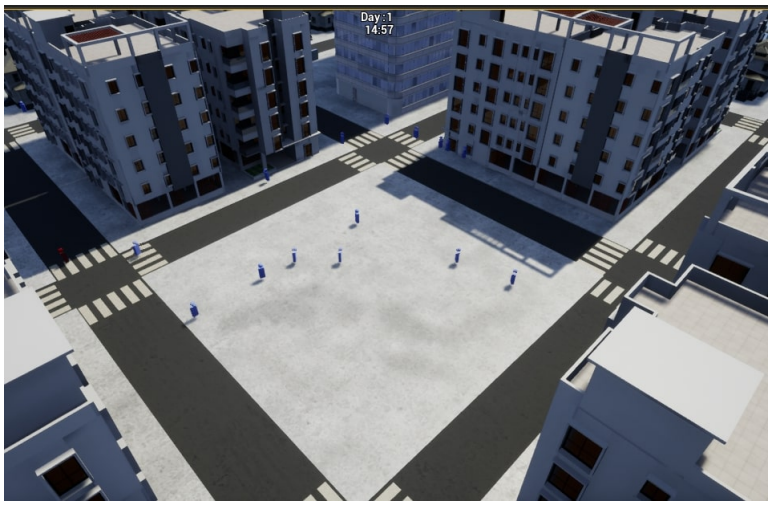


Figure 5.5: Central Square

5.2.2 Hospital

The hospital is a very important place because it acts as a quarantine zone for the infected people that show symptoms. These people will be put in a separate area of the building where they can not infect other people. This changes the dynamic of the disease because the spread of the disease counts on people infecting others before having serious symptoms of the disease. There is also medical staff in the building and they have to work close to infected people increasing the risk of getting infected. For better monitoring the interior of the building a seperate model has been created. There is a camera that observes the interior of the hospital and can be accessed during the simulation by pressing ’6’ on the keyboard.

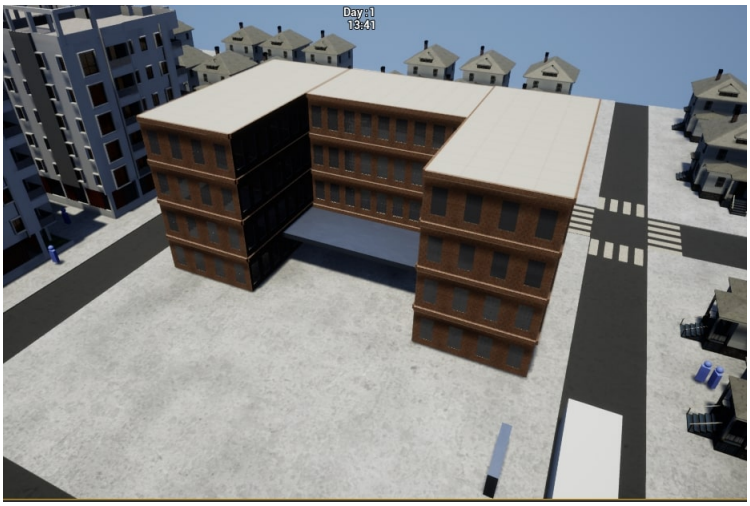


Figure 5.6: Hospital

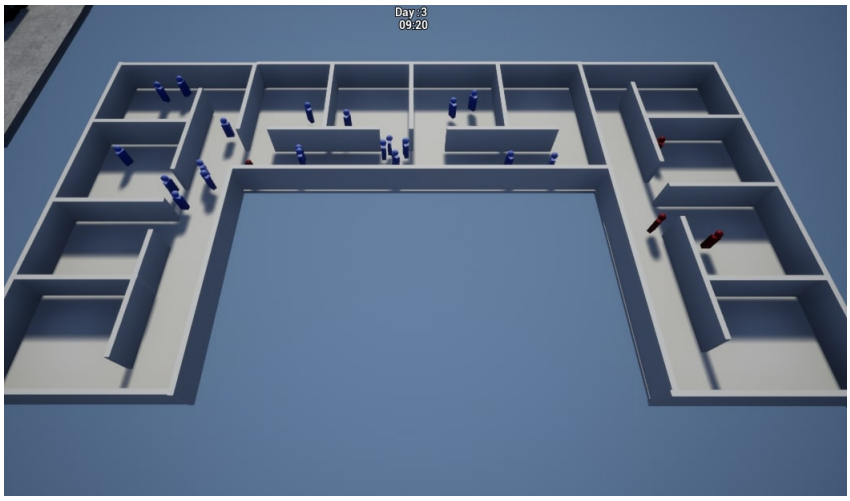


Figure 5.7: Hospital's interior

5.2.3 School

The school in this world is also at higher risk of virus spread due to the close contact and frequent interaction among students and staff. Schools are ideal environments for the virus to spread because they involve large groups of people who are in close proximity to one another for extended periods of time. Crowded classrooms, shared spaces, and communal areas such as bathrooms, hallways, and cafeterias create opportunities for the virus to spread. Some diseases have different infection rates between younger people compared to adults, therefore it would be helpful to study the spread of the disease in a school. Like the hospital, there is a separate model for the interior which can be viewed by pressing '7' on the keyboard.

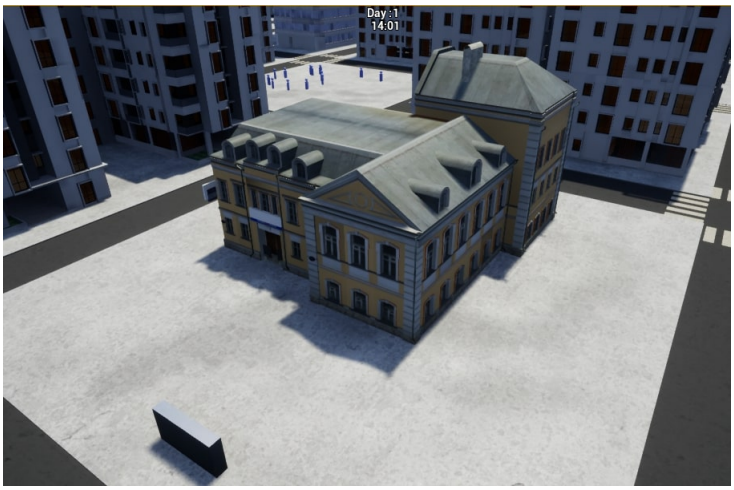


Figure 5.8: School

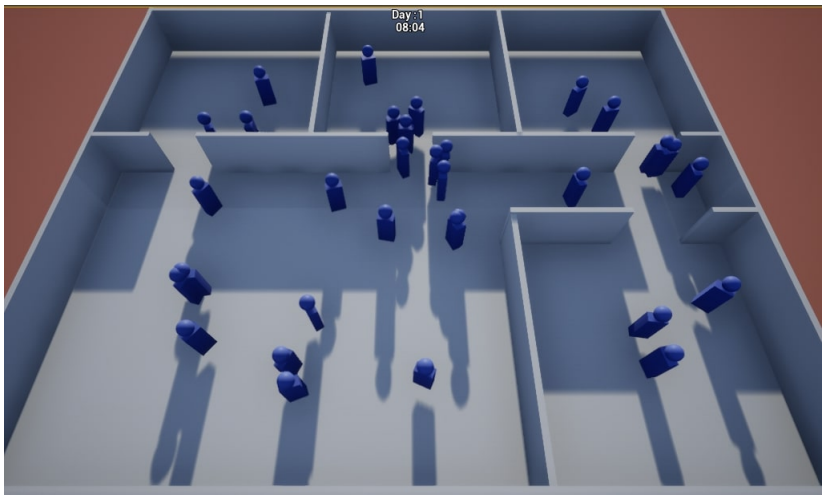


Figure 5.9: School's interior

5.2.4 Office building

An office building in this simulation would also be at risk of virus spread due to the close contact and frequent interaction among employees. Offices often have crowded common areas, shared spaces, and break rooms, which can create opportunities for the virus to spread. Additionally, employees may travel on crowded elevators, use shared equipment, and attend in-person meetings, increasing their risk of exposure to the virus. This building follows the same idea as the hospital and school having a separate interior model which can be viewed by pressing '8' on the keyboard.

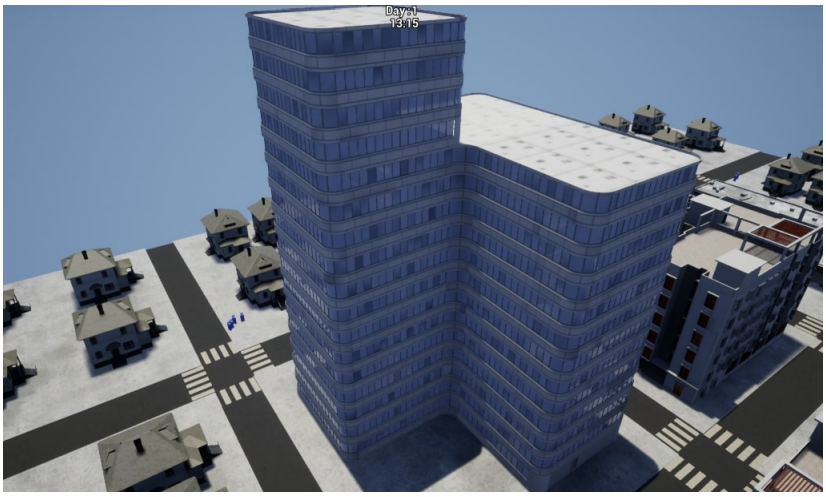


Figure 5.10: Office building

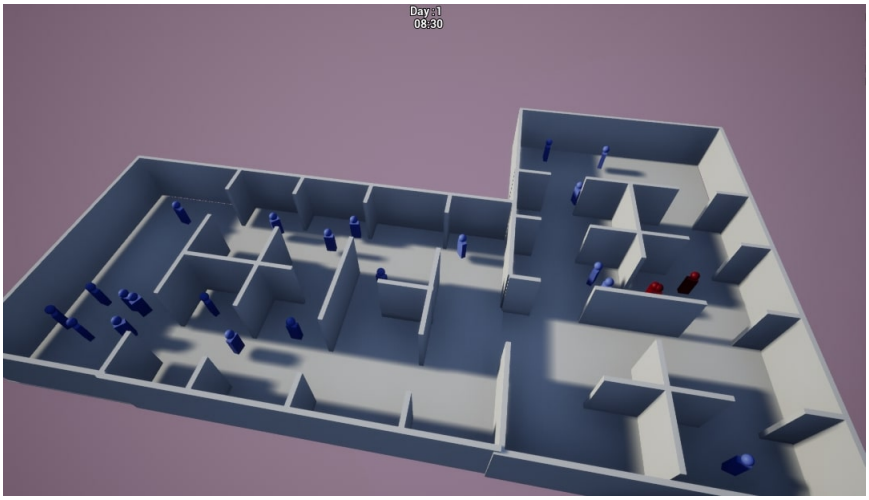


Figure 5.11: Office interior

5.2.5 The mall

A shopping mall in the simulated town could be a source of virus transmission due to the large crowds it attracts and the close proximity of individuals in shared spaces. Malls are open-plan structures that include a variety of stores, restaurants, and common areas, leading to high levels of human interaction and potential for overcrowding. In addition, unlike offices and schools where there are certain people around you, at the mall there is interaction between random people in shared spaces. That can play an important role to the spread of the infection outside of social circles.

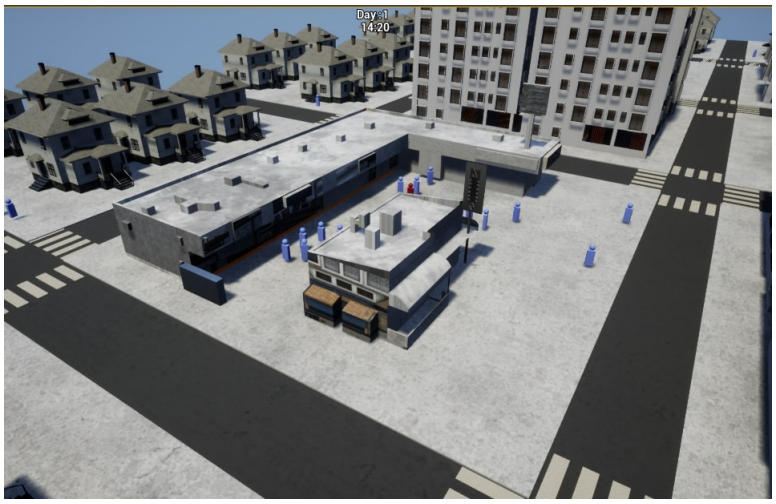


Figure 5.12: Mall

5.2.6 Public Transport

Public transportation and in particular in this simulation, the bus, can contribute to the spread of the virus due to close proximity of passengers and shared surfaces. Buses are often crowded, and passengers may come into close contact with each other and with high-touch surfaces such as handrails and seat belts. This is also a place where many random people come to close proximity and the chances of spreading the infection is high. In addition, waiting at bus stations can also contribute to the spread of the infection. The user can view the interior of the bus by pressing ’5’ on the keyboard.

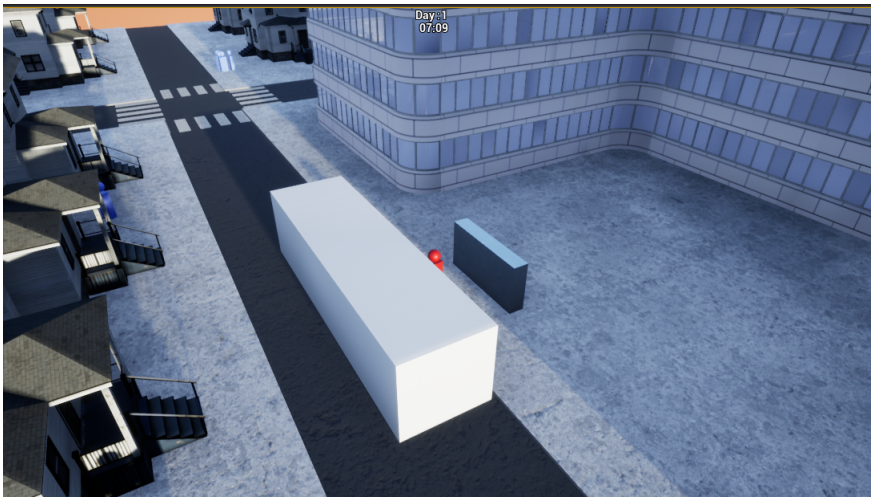


Figure 5.13: Bus station

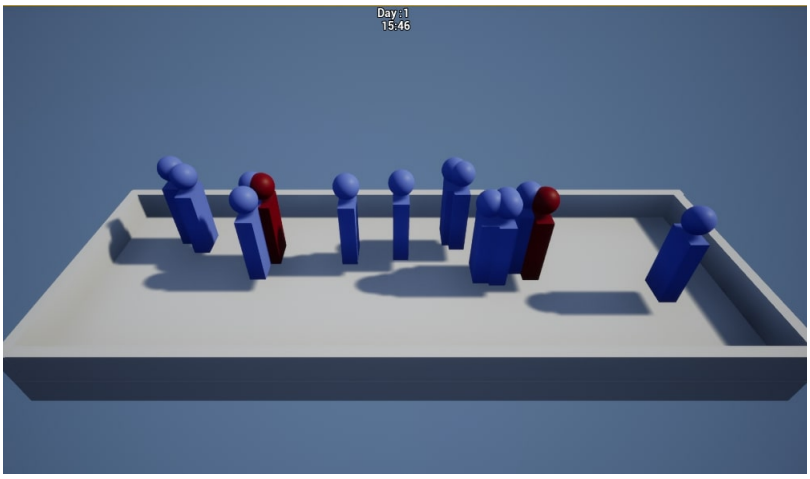


Figure 5.14: Crowded bus

5.2.7 Apartment buildings

Apartment buildings in the simulated town can play a role in the spread of the infection due to close proximity of residents and shared spaces. In apartment buildings, residents may have close contact with each other in common areas such as elevators, laundry rooms, and hallways, increasing the risk of transmission. Additionally, shared ventilation systems can also spread the virus through the air.

Figure 5.15: Appartment Buildings

5.2.8 Detached houses

Detached houses in the simulated town can have a more mild effect to the spread of the virus due to their private nature and lower risk of close contact with others. In a detached

house, residents have their own separate living spaces, reducing the likelihood of close contact with others and reducing the risk of transmission. The private nature of detached houses also allows for more control over one's environment, including the ability to limit the number of visitors, maintain good hygiene practices, and reduce exposure to others who may be carrying the virus. This can help to reduce the risk of transmission and keep residents as safe as possible. However, an infection can be spread between family members and then to another family through social networks.

Figure 5.16: Neighborhood

5.3 Citizens

5.3.1 Citizen initialization

As previously stated, every individual has its own routine and social network, meaning:

- a specific job
- a specific house
- family members or roommates
- but random habits

Their workplace, their shift and their house are randomly selected when the simulation start running and stay constant until the end of the simulation in contrast with their free time activities which change every day.

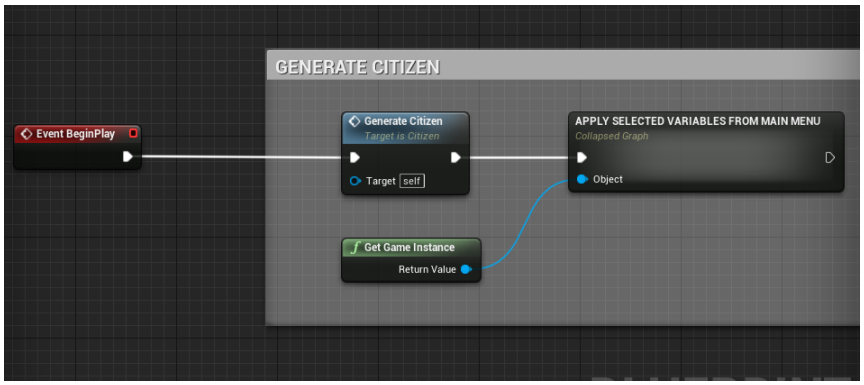


Figure 5.17: Citizen Generation Graph (blueprints)

Each individual has 4 states:

- Healthy but Susceptible
- Infected
- Immune or Vaccinated
- Dead

As long as people remain healthy or become infected but do not show symptoms they follow their daily routine. As soon as symptoms appear, the infected person goes to hospital quarantine or stays at home and waits there until recovery. However, there is a self isolation percentage setting that allows for some infectious individuals to deny isolation. Some people remain asymptomatic after the incubation period and can transmit the disease throughout their infection although they are less infectious . In the case of covid-19, after review of many studies that percentage of infectiousness compared to a symptomatic individual is 42% . [11]

5.3.2 AI Controller

Every citizen is a pawn that can be controlled through an AIcontroller class. It has the ability to move from a certain point to another. However, the brain that moves this pawn is the AIcontroller which I named CitizenController. CitizenController is in charge of feeding the necessary variables and running the suitable behavior tree. This variables are stored in the Blackboard. Blackboard is a component that stores values of certain variables which are used from the behavior tree in order to execute the right node. In this case this variables include

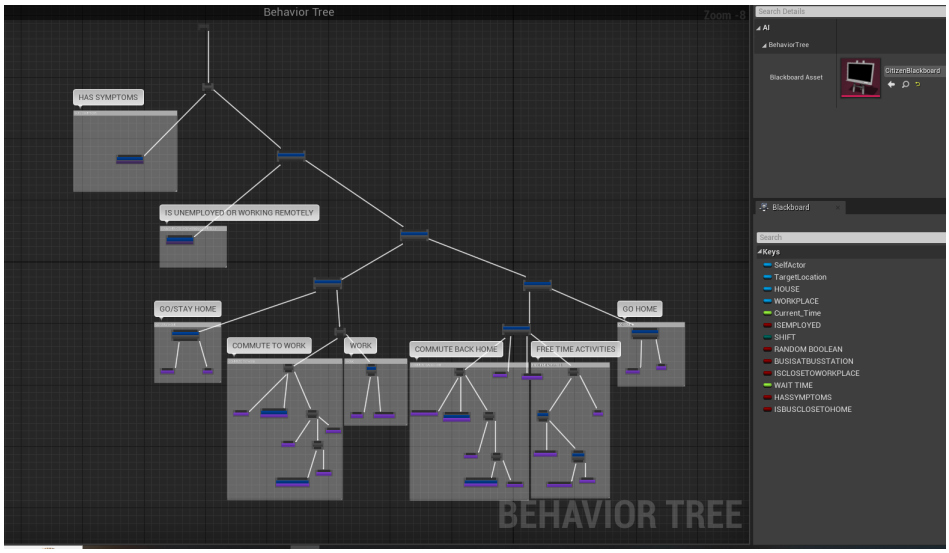


Figure 5.18: BehaviorTree for morning shift

personal data of the citizen such as his house, workplace, shift, as well as information that is considered in order to make a decision like time, health status and more.

As shown in the picture behavior trees in Unreal Engine 4 allow for complex, hierarchical decision-making structures, where actions and conditions are combined to create a behavior for a game character.

In UE4, a behavior tree consists of individual nodes that are connected to form a tree-like structure. The nodes represent actions, conditions, and control structures. The tree is processed from top to bottom, and the behavior of the game character is determined by the execution of the nodes.

Actions are nodes that perform some specific behavior, such as moving to a certain location or attacking the player. Conditions are nodes that test whether a certain condition is true or false, such as checking if it is time to go to work or if the character's health status has changed. Control structures are nodes that dictate the flow of execution, such as loops, sequences, and parallel tasks.

For simplicity reasons, five different behavior trees were created. Three of them for each working shift, one for the unemployed and one for the infected citizens that show symptoms. The AI code is executed starting from the root of the tree. Every tree works based on the same logic and variables. Top priority is given to the health status node. If the citizen shows symptoms then the ”Hospital ” node is executed. When HasSymptoms value equals to false , the second priority node ”IsEmployed” is examined. If its value is false then the ”Unemployed”

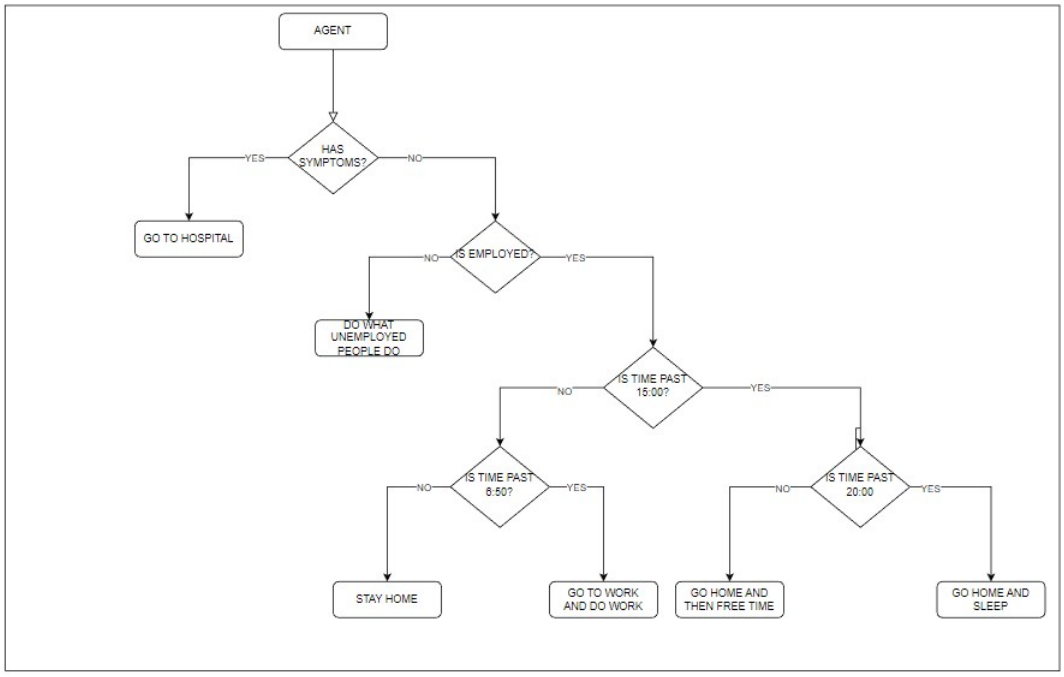


Figure 5.19: Simplified BT diagram

node is executed. Even when a citizen is employed the state of this node can change when the user disables the option to work on site. This makes the agents follow the unemployed behavior tree. The next priority is a group of nodes that check the time of the day. This value is changing every frame and is responsible for creating the daily routine of an agent. This is the main part of the tree that differs between the 3 working shifts. Depending on the current time different subroutines are executed .

5.4 Infection spread mechanics

5.4.1 Basic mechanics

In this simulation, the spread of an infection is modeled only through air transmission, making the model simpler and less demanding in terms of computational resources. Each agent has a sphere with a radius of 2 meters and a cone that extends from their mouth to a distance of 2 meters in front of them.

Every frame the infected agents check if there is some other agent inside there effective radius. If there are, there is a user defined chance to transmit the disease:1

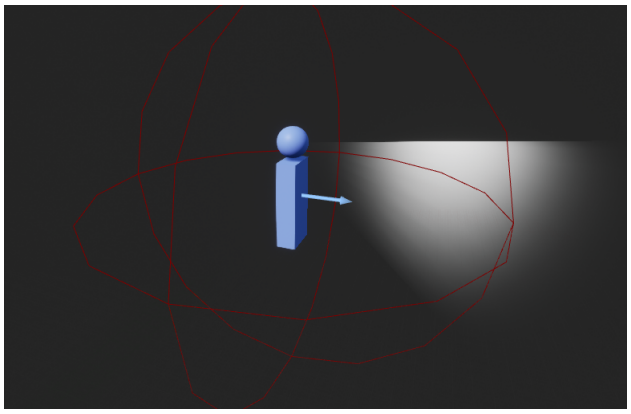


Figure 5.20: Healthy Citizen

Algorithm 1 Spreading the disease

Require: $random \geq 0$ **and** $random \leq 1$

for $AgentsinRadius$ **do**

if $Agent.Healthstatus \neq Infected$ **then**

Calculate $ChanceOfGettingInfected$

$random \leftarrow RandomFloat()$

if $random < Agent.ChanceOfGettingInfected$ **then**

$Agent.Healthstatus \leftarrow Infected$

else

// Do not transmit the virus//

end if

end if

end for

5.4.2 Dynamic factors

The term dynamic factor is used to describe the factors of an infection that can be different to every individual and different conditions. Of course the **ChanceOfGettingInfected** variable is initialised by the user on the main menu, however, further adjustments are made to determine the actual probability:

- **Mask protection:** when the agent wears a mask the probability of spreading the disease is reduced to 30% as shown in figure 5.21 by the ratio between $P_{inf,pop,mask}$ and $P_{inf,pop}$ in the darker area. If the other agent wears a mask the probability is reduced to 30% leading to an overall 9% of the initial probability with universal masking . [2]

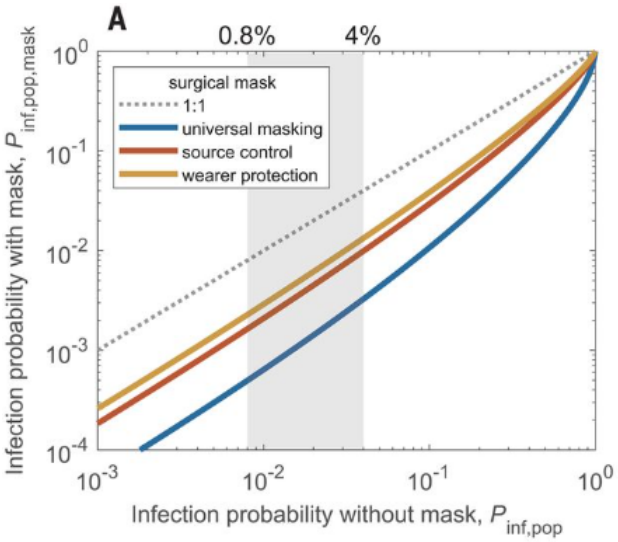


Figure 5.21: Mask effectiveness ratio [2]

- **Immunity:** When the other agent is considered immune the probability gets further reduced to the value that is determined by the user on the main menu. If the value equals to zero then vaccinated or recovered individuals cannot get infected again.
- **Asymptomatic individuals:** asymptomatic individuals have a reduced chance of spreading the disease. In this simulation, it is set to 42% according to this study. [12]
- **Environment**: contact with an infected person indoors has a higher risk of infection than outdoors. [13]
- **Distance and direction:** when a healthy agent is overlapping the cone component in front of the agent the probability increases.

When an agent gets infected, its color becomes red. There is an incubation period, defined by the user, where the agent is 42 % as infectious as in later stages of the infection. In addition, there is a user predetermined chance of not showing symptoms during the infection. This is important because these individuals will not get quarantined and continue to spread the virus until they recover. After recovery, they become immune , to a degree, and change color to green. This creates a dynamic environment where when an aggressive virus can actually have a reduced spread compared to a more mild infection.

Another dynamic factor to the spread of the infection is the structure of the society. Every individual has their own unique routine and social network which includes factors such as their job, place of residence, family members or roommates, and personal habits. This creates

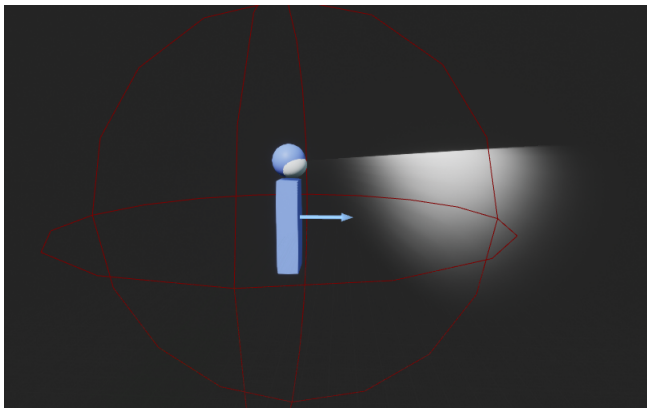


Figure 5.22: Citizen wearing mask

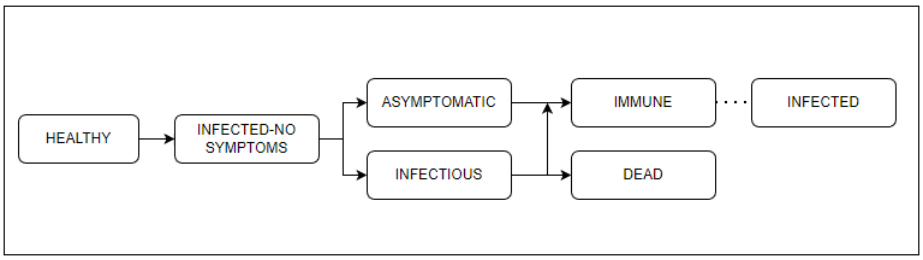


Figure 5.23: Infection states diagram

uneven odds for infection to spread between random people in contrast with other simulation models. In this model there is a dynamic network which works in combination with the agent-based model. Every agent has some individuals that is more likely to come into contact than other agents. Some networks are obvious like family member and colleagues but there are some less obvious. For instance, if a resident uses public transport to commute to work and back, we conclude that he often shares the bus with the same people.

In conclusion, this simulation model gives a bigger depth to the infection mechanics enabling the user to test more measures and parameters and extract new conclusions. It is a tool that can be used to teach students how a virus spreads in our society.

5.5 Performance optimization

The primary focus when modeling the spread of an infection is on accuracy and efficiency, not the visual representation. Simple 3D models can be used, as the emphasis is on the simulation, not the graphics. However, when multiple AI-controlled agents are involved, optimizing the performance of the CPU becomes increasingly important. UE4 provides a range of tools and techniques for a developer to optimize both the GPU and CPU performance which have

been mention in chapter 4.

5.5.1 GPU optimization

In this simulation, simple 3d models where used preventing the simulation for being GPU heavy. This can be seen through the shader complexity viewmode and by using the game stat command we see that the vast majority of the frame time is spend on game logic. In the figure 5.24 we see that the GPU spends 1.4 ms to render the frame. Figure 5.25 shows that this simulation world has simple geometry.

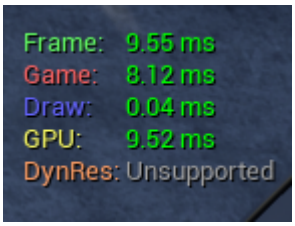


Figure 5.24: Unit Stat

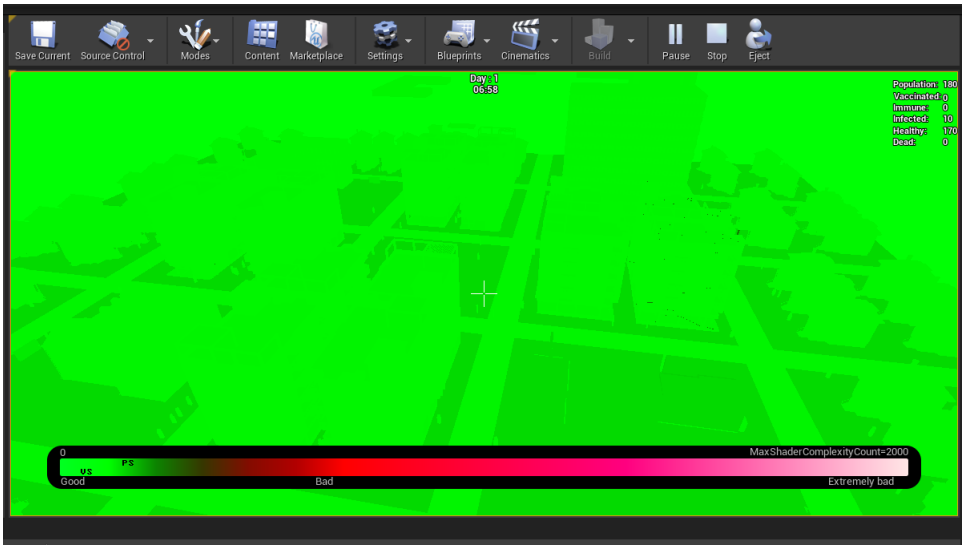


Figure 5.25: Shader Complexity Viewmode

5.5.2 CPU optimization

As stated in Chapter 2, agent-based simulations are computationally expensive to run and scale. This is due to the fact that each agent in the simulation must be tracked and updated individually, which can result in a large amount of computational resources being required.

This can be especially true for large-scale simulations with many agents and complex interactions. Additionally, agent-based simulations often require significant storage space to store the data generated by the simulations. To mitigate these issues, various optimization techniques can be used to improve the efficiency and scalability of agent-based simulations.

Unreal Engine's multithreading capabilities can help to mitigate the computational expense and improve the efficiency and scalability of the simulations. However, UE4 uses 1 main game thread to compute all the game logic, AI decisions, collision check etc. There are some basic things that can be done in order to improve performance that can be found in the UE4 documentation including: [1]

- Check expensive calculations such as collision etc. to ensure the settings you are using are optimal, reduce collisions or the amount of particles/meshes colliding.
- Remove expensive calculations such as collision, dynamic parameters, etc. if necessary.
- Replace Game Thread calculated particle effects with Static Mesh effects if possible.
- Avoid using expensive functions and big loops in part of the code that is running every frame.
- Set the actors to tick only when necessary.

In this simulation, due to the large number of agents, most of the computational power goes to calculating collisions and overlaps with other agents which must be updated every frame, as well as running behavior trees . The simulation requires that an infected agent tracks every other agent who is inside its effective radius and calculates if it should transmit the virus. Many agents gathered together leads to an increased cost, because each infected agent has to track many agents in close proximity. Furthermore, running hundreds of instances of AI code , especially when the agents are not idle, causes a big hit to the simulations performance. An AIController is used for every agent meaning that up to 250 AI related threads can be active simultaneously. Using the Profiler as shown in figure 5.27 we confirm that CheckIfInfectedNearby function is quite demanding. In order to solve these problems, 5 important steps were made:

- Tracking of other agents in close proximity only when the agent is infected. In that way, expensive calculations are made less frequently while still remaining accurate.

- The AIController tick was set to 50 milliseconds, as making a decision every frame was not necessary in this model.
- Behavior trees were designed in such a way where different groups of agents were idling and being active at a different time of the day spreading the load more evenly and as a result, fps drops were less severe.
- Blueprint code was written with performance in mind, making sure to use expensive functions only in rarely used parts of the code ,avoid loops and accessing big data-sets frequently.
- C++ was used in complex calculations that had to be calculated every frame for a large number of agents.

```cpp
#include <stdlib.h>
#include "CalcProb.h"


bool UCalcProb::MultiplyFloats(float ChanceOfImmuneGettingInfected, float ChanceofInfectionPerHour, float DeltaSeconds,
    bool IsImmune, bool IsOutside, bool IsWearingMask, bool HasSymptoms, bool OtherMask)
{
    double result=1.;

    if (IsImmune) { result = ChanceOfImmuneGettingInfected; }

    if (IsOutside) { result *= 0.5f;  }

    if (IsWearingMask) { result *= 0.3f; }

    if (OtherMask) { result *= 0.3f; }

    if (!HasSymptoms) { result *= 0.42f;  }

    result *= ChanceofInfectionPerHour ;
    result *= DeltaSeconds;
    float random = static_cast <float>( rand()) / static_cast <float>(RAND_MAX);
    //UE_LOG(LogTemp, Warning, TEXT("Result =  % f"), result);
    return (result > random*100 );
}


void UCalcProb::UpdateCounters(int HealthyCounter, int VaccinatedCounter, int ImmuneCounter,
    int VaccunatedSum, int &HCounter, int &VacCounter, int &ICounter, int &VacSum ) {

    HCounter= HealthyCounter - VaccinatedCounter;
    ICounter = ImmuneCounter + VaccinatedCounter;
    VacSum = VaccunatedSum + VaccinatedCounter;


}
```

Figure 5.26: C++ functions for probability calculation and counter updating

After all those optimisations, a mid-range 5 year old quad core CPU is able to handle a simulation with 200 agents, while keeping frame rates over 30 fps. In conclusion, performance optimization was a critical aspect of the development process of the UE4 simulation of the spread of an infection in an agent-based model. The use of optimization techniques, such as profiling and bottleneck analysis, allowed us to improve the simulation's speed and reliability, resulting in a more seamless and immersive user experience. Moreover, incorporating optimization techniques in the blueprint scripting also highlighted the potential benefits of using C++ in UE4 development. Utilizing C++ in future developments of this simulation can

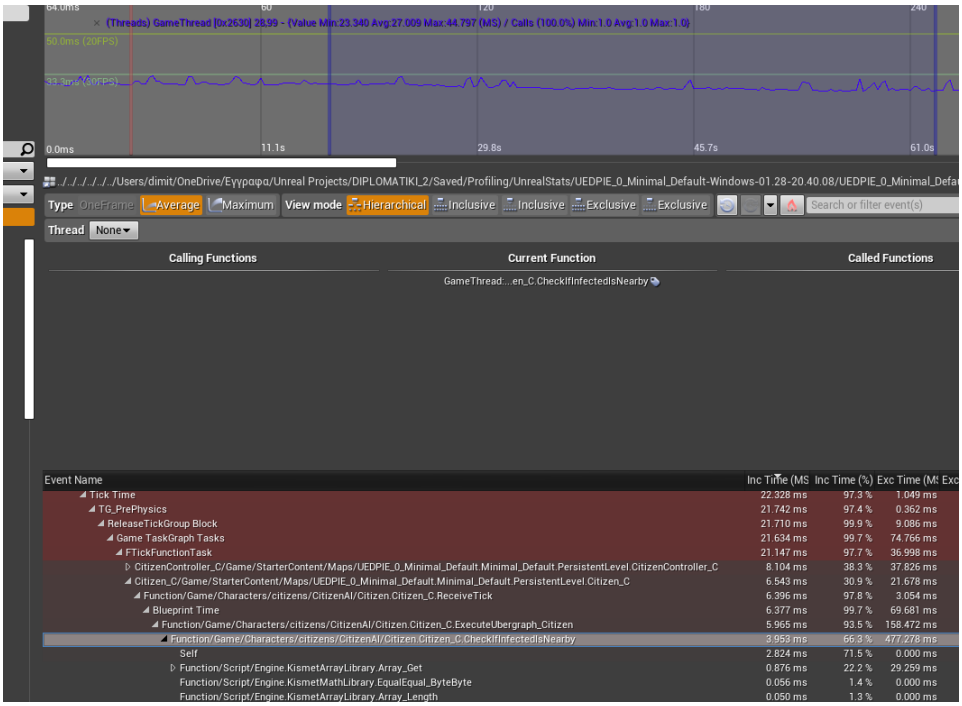


Figure 5.27: UE4 Profiler. Checking the run time of the CheckIfInfectedNearby function

further optimize the performance, making it more scalable and efficient. In summary, performance optimization was a crucial component in the development of this simulation and will continue to play a vital role in ensuring the simulation delivers the best possible user experience.

5.6 Testing

Before running the simulation, it is important to take a look at the default values of the variables involved. Most of them are based on various studies and are set based on Covid-19 studies, as this is the most recent outbreak. Some of these variables can be set by the user, however for simplicity reasons some are predetermined with Covid-19 in mind. There are many variables included in Citizen class , showing the complexity of the developed model. However only the last 7 are important for testing.

- Chance of infection per hour, meaning the probability of an individual contracting the virus in an hour of exposure. It cannot be directly measured with accuracy so it is calculated using probabilities.

- Infection period is set to 1200 time units which translates to 5 days, and can be altered

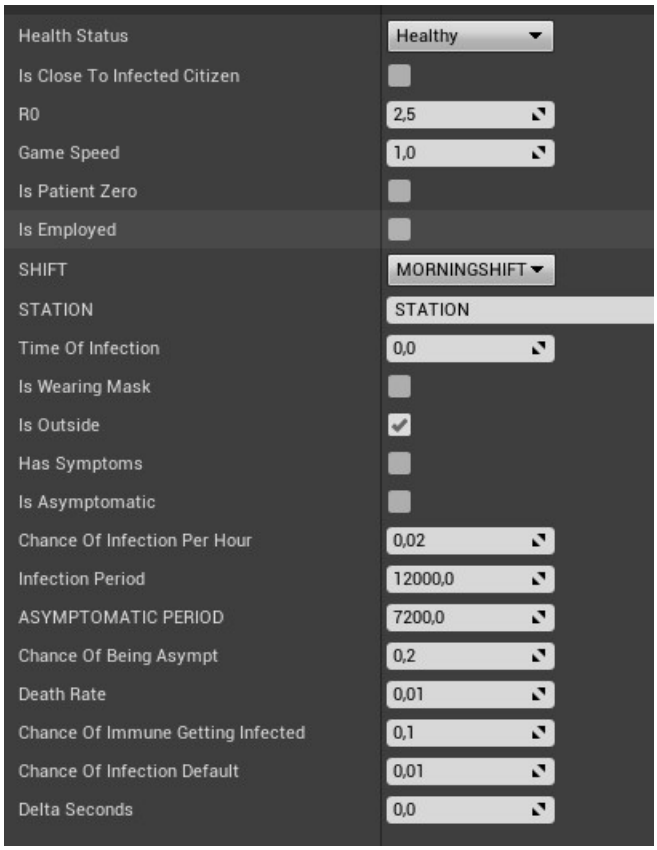


Figure 5.28: Variables of Citizen class

by the user. Usually the infectious period of Covid-19 is higher.

- Asymptomatic period is the incubation period. It is set to 3 days and can be altered by the user.
- The probability of showing no symptoms while being infected. It is set to 0,3 and that value is derived as an average from Covid-19 studies that estimate the average range from 0,2 to 0,4. Default value is 0,4 in order to include very mild symptomatic. [12] [14]
- Death rate is the case fatality risk and is set to 1% after examining various Covid-19 data. The world average is closer to 2% but in developed countries is lower. [15] [16]
- Chance of immune getting infected correlates with the effectiveness of vaccines or natural immunity after exposure. Studies show a 95% to 70% effectiveness depending on the variant. Reduction in effectiveness is unrelated in small time periods. [17]

Research data estimate the R_0 of Covid-19 between 2.0 and 5.0, meaning each initial infection leads to 2 or 5 secondary infections. This is a very wide range which makes it difficult

to get a good estimate for the infectiousness of the virus. Therefore, a different approach is taken.

According to a study, family members have 39% chance of getting infected and coworkers 30%. The average infectious period of Covid-19 is estimated to be 6.5 to 9.5 days and the incubation period 1 to 4 days. During quarantine we can assume that family members spend 40 to 60 hours together in a similar time window. There is big uncertainty in these data ,as many factors can affect the results , however we can try and calculate the ChanceofInfectionPerHour variable by comparing data. According to another study, that took place in a prison, which is a fairly controlled environment, they concluded that in 2.3 days of exposure there is a 36% chance of a non-vaccinated cellmate contracting the virus. We can estimate the average risk in an hour of exposure by dividing the estimated hours spend together in 2.3 days into different probability tests. [18] [19] [20]

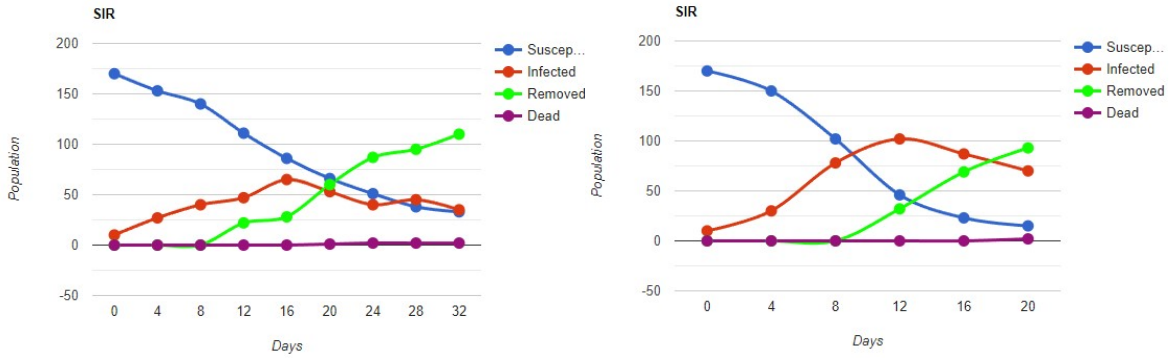
Considering $P(A)$ the chance of getting infected in an hour, and assuming 44 hours were spend together we can express the probability of A occurring at least once in 44 experiments as follows:

$$1 - (1 - P(A))^{44} = 0.36$$

Using this formula we can calculate $P(A) \approx 0.01$. This value can vary from 0.008 to 0.012 depending on the actual time the cellmates spend together. Using the same formula for family members, we reach a similar range. However, the rate of infectiousness vary during the infection period. During the early and late stages the patient is less infectious. Of course, the conclusions derived are not intended to be used for scientific purposes, therefore some uncertainty in the data is accepted. In the end, the user is given the freedom to gather data from any infectious disease and apply the settings accordingly.

Starting the simulation using Covid-19 data and setting ChanceofInfectionPerHour to 0,01, infection period to 8 days and incubation period to 3 days, we get the graph shown in figure5.29α′. Once the test simulation is over we run it again but this time only 50% of symptomatic stays isolated, as shown in figure 5.29β′. The results show that staying in isolation when symptoms appear, can play a significant role, keeping R in control during the outbreak. $R_0 \approx 3$ and $R_0 \approx 7$ respectively.

In test 3 the population is reduced from 180 to 120, modeling the population density of a smaller town. The rate is similar, $R \approx 3,2$, but the overall percentage of population infected



(α′) Self isolation 90% (β′) Self isolation 50%

Figure 5.29: Testing the importance of self isolation, $P(A) = 0,01$

(α′) Population =120 (β′) Infection period=4 days

Figure 5.30: Lower population density and half duration of symptoms,$P(A) = 0,01$

is reduced, as shown in figure 5.30α′. In test 4, figure 5.30β′, the incubation period is set to 2 days and the infection period to 4 days. This shows that aggressive infections usually have a low infection rate and in order for a virus to be spread widely it has to be either more infectious or have more mild symptoms,especially when self isolation percentage is high(80%).

In the end, there is too much uncertainty in the variables to be able to accurate simulate a real disease, however, the user can compare different scenarios and use his own data. This is a learning tool and its biggest advantage is the versatility and the engaging approach that it is implemented with. After all, the model can only be as accurate as the data used.

Chapter 6

Conclusion

This section is dedicated to the conclusion of the present thesis. In it, a summary of the entire work is made, as well as the results that arose. Furthermore, the future extensions and plans for this simulation are mentioned.

6.1 Summary

There is a growing interest in virus dispersion simulations in the current climate, given the impact that outbreaks such as COVID-19 have had on the world. These simulations provide valuable tools for understanding the complex dynamics of virus spread and for testing different measures for controlling its spread. By simulating real-world scenarios, these simulations allow us to assess the impact of various measures and to identify the most effective approaches for controlling the spread of a virus.

The UE4 agent-based simulation developed in this thesis is a prime example of the type of valuable tool that can be developed to study virus dispersion. The ability to customize various parameters and test different scenarios makes this simulation a powerful tool for exploring the complex dynamics of virus spread and for assessing the impact of various measures , including vaccines, quarantine measures, and the effectiveness of masks for controlling its spread. The agents in the simulation being AI controlled and having their own social life and daily routines adds to its realism and provides a dynamic environment for the infection to spread into, enhancing its effectiveness as a tool for studying virus dispersion.

In addition,the simulation developed in this thesis makes a significant contribution to the field of computer engineering. The simulation demonstrates the power and versatility of

agent-based modeling as a tool for simulating and analyzing complex systems, such as the spread of viruses in a population. By using 3D modeling and game-like appearance, the simulation provides a unique and engaging approach to the subject matter, making it an effective educational tool for teaching children and adults about the dangers of an outbreak and the importance of preventative measures.

This work also highlights the importance of interdisciplinary collaboration between computer engineering and fields such as epidemiology and public health, in developing effective solutions for addressing real-world challenges.

Github link: `https://github.com/diziogas/thesis`

6.2 Future work

This simulation provides a solid foundation for further work in the field of virus dispersion and control. Here are some potential areas for future work based on this simulation: The simulation that was created as part of this study has shown promise in its ability to model the spread of viruses. However, there are several areas in which the simulation could be improved to better reflect real-world scenarios and to provide more comprehensive insights into the spread of viruses. In this section, we will discuss some future work ideas that could enhance the value of the simulation.

One of the key areas of improvement is to expand the simulation to include more realistic and complex scenarios. Currently, the simulation models the spread of viruses in a limited population range and does not account for different age groups or social structures. Expanding the simulation to include these factors would provide a more accurate representation of how viruses spread in the real world and could provide additional insights into the factors that influence the spread of viruses.

Another area of improvement is the development of new visualization and analysis tools to better understand the results of the simulation. By incorporating more advanced visualization tools, we can gain a deeper understanding of the patterns and trends in the spread of viruses, which can help us to identify the key factors that influence their spread. Additionally, developing new analysis tools will enable us to quantify the impact of different factors on the spread of viruses and to make more informed decisions about how to control their spread.

A third area of improvement is the incorporation of additional measures for controlling

the spread of viruses. Currently, the simulation models only airborne transmission of viruses, but there are many other factors that can influence the spread of viruses. By incorporating these additional factors into the simulation, we can evaluate their impact on the spread of viruses and determine the most effective strategies for controlling their spread.

Another important aspect of improving the simulation is to compare its results with real-world data. This will allow us to validate the accuracy of the simulation and to identify any areas where it may need to be improved. By comparing the results of the simulation with real-world data, we can ensure that it is providing accurate and meaningful insights into the spread of viruses.

Finally, we could make the simulation even more engaging by integrating challenges and a score system. By making the simulation more fun and interactive, we can attract more people to participate and learn about the spread of viruses, which can help to raise awareness about the importance of controlling their spread.

Bibliography

[1] Unreal engine 5 documentation. https://docs.unrealengine.com/4.27/en-US/. Accessed: 2023-01-25.

[2] Yafang Cheng, Nan Ma, Christian Witt, Steffen Rapp, Philipp S. Wild, Meinrat O. Andreae, Ulrich PI¶schl, and Hang Su. Face masks effectively limit the probability of sars-cov-2 transmission. *Science*, 372(6549):1439–1443, 2021.

[3] Craig, Ben R., Tom Phelan, Jan-Peter Siedlarek, and Jared Steinberg. Improving epidemic modeling with networks. https://doi.org/10.26509/frbc-ec-202023. Federal Reserve Bank of Cleveland, Economic Commentary 2020-23.

[4] Abolmaali S. and Shirzaei S. A comparative study of sir model, linear regression, logistic function and arima model for forecasting covid-19 cases. *AIMS Public Health*, 8(4):598–613, Sept. 2021.

[5] M. J. Keeling and K. T. . Eames. Networks and epidemic models. *Journal of The Royal Society Interface*, 2(4):295–307, Sept. 2005.

[6] Elizabeth Hunter, Brian Mac Namee, and John D Kelleher. A comparison of agent-based models and equation based models for infectious disease epidemiology. In *AICS*, pages 33–44, 2018.

[7] L. Willem. *Agent-Based Models For Infectious Disease Transmission*. PhD thesis, University of Antwerp, Dec. 2015.

[8] U Wilensky. Netlogo virus model. http://ccl.northwestern.edu/netlogo/models/Virus, 1998. Accessed: 2023-01-12.

[9] Epidemic simulation. https://prajwalsouza.github.io/Experiments/Epidemic-Simulation.html. Accessed: 2023-01-11.

[10] Thessaloniki, population. `https://worldpopulationreview.com/world-cities/thessaloniki-population`. Accessed: 2023-01-10.

[11] Oyungerel Byambasuren, Magnolia Cardona, Katy Bell, Justin Clark, Mary-Louise McLaws, and Paul Glasziou. Estimating the extent of asymptomatic covid-19 and its potential for community transmission: Systematic review and meta-analysis. *Journal of the Association of Medical Microbiology and Infectious Disease Canada*, 5(4):223–234, 2020.

[12] Diana Buitrago-Garcia, Dianne Egli-Gany, Michel J. Counotte, Stefanie Hossmann, Hira Imeri, Aziz Mert Ipekci, Georgia Salanti, and Nicola Low. Occurrence and transmission potential of asymptomatic and presymptomatic sars-cov-2 infections: A living systematic review and meta-analysis. *PLOS Medicine*, 17:1–25, 09 2020.

[13] B.R. Rowe, A. Canosa, J.M. Drouffe, and J.B.A. Mitchell. Simple quantitative assessment of the outdoor versus indoor airborne transmission of viruses and covid-19. *Environmental Research*, 198:111189, 2021.

[14] Qiuyue Ma, Jue Liu, Qiao Liu, Liangyu Kang, Runqing Liu, Wenzhan Jing, Yu Wu, and Min Liu. Global Percentage of Asymptomatic SARS-CoV-2 Infections Among the Tested Population and Individuals With Confirmed COVID-19 Diagnosis: A Systematic Review and Meta-analysis. *JAMA Network Open*, 4(12):e2137257–e2137257, 12 2021.

[15] Elisabeth Mahase. Coronavirus: covid-19 has killed more people than sars and mers combined, despite lower case fatality rate. *BMJ*, 368, 2020.

[16] Rachel E Jordan, Peymane Adab, and K K Cheng. Covid-19: risk factors for severe disease and death. *BMJ*, 368, 2020.

[17] Dan H. Barouch. Covid-19 vaccines immunity, variants, boosters. *New England Journal of Medicine*, 387(11):1011–1020, 2022.

[18] Tsang TK, Fang LQ, Zhang, Jiang FC, Ruan SM, Liu LZ, Cowling BJand Liu W, and Yang Y. Variability in transmission risk of sars-cov-2 in close contact settings: A contact tracing study in shandong province, china. *Epidemics*, 39(100):553, 2022.

[19] Andrew William Byrne, David McEvoy, Aine B Collins, Kevin Hunt, Miriam Casey, Ann Barber, Francis Butler, John Griffin, Elizabeth A Lane, Conor McAloon, Kirsty

O'Brien, Patrick Wall, Kieran A Walsh, and Simon J More. Inferred duration of infectious period of sars-cov-2: rapid scoping review and analysis of available evidence for asymptomatic and symptomatic covid-19 cases. 10(8), 2020.

[20] S.T. Tan, A.T. Kwan, and I. Rodríguez-Barraquer. Infectiousness of sars-cov-2 breakthrough infections and reinfections during the omicron wave. *Nature Medicine*, 2023.