

CoviSim: Research on the Use of a Simulated Environment to Demonstrate the Transmission of COVID-19

Final Project Report

DT211c

BSc in Computer Science (Infrastructure)

**Kyle Heffernan**

**C17444434**

**Bryan Duggan**

School of Computer Science

Technological University, Dublin

**31/03/2021**

Abstract

Computer simulation has always been an invaluable tool when it comes to researching infectious diseases, as real-life experiments have many potential risks. Over the course of the past year, countless scientists and doctors all over the world have been continuously researching Coronavirus in a global effort to overcome the pandemic and get back to normal everyday life. There have been numerous Coronavirus related simulations made over the past year focusing on a wide variety of aspects of the virus.

Many simulations offer a high-level overview of the pandemic on a large scale, having only a few variables affecting the results. These simulations tend to focus on the spread throughout a city, and the virus is transmitted when agents come within a certain range of an infected agent. While this serves as a good visualisation of spread throughout a population, it is a drastic oversimplification of how transmission can occur and does not show how the virus actually transmits between people.

This project is focused on transmission in a closed environment, highlighting the actual methods of transmission and allowing the user to truly understand how certain countermeasures affect the results. There is a surplus of medical papers and scientific studies from around the world which provide statistics on transmission rates and the effects of various countermeasures. Some of these statistics have been utilised in the simulation as parameters to give a more accurate result.

As Coronavirus continues to grow, so does misinformation about it on social media. While a small amount of information is given to the public about countermeasures that they can take to prevent transmission, the results of these countermeasures are not easy to identify. This simulation is a practical solution to this, using real figures to visualise transmission and the effectiveness of various countermeasures in a real-time closed environment.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Kyle Heffernan

Date: 31/03/2021

Acknowledgements

I would like to thank my supervisor Bryan Duggan for his continuous support, guidance and motivating words throughout the course of this year. I would also like to thank my friends for keeping me motivated and on track this year with our countless online study sessions.

A special thanks to my family for always keeping me grounded and helping me through the more stressful times, and finally Danniella for her constant love and support which I could not have done this project without.

Table of Contents

[1. Introduction 7](#_Toc21459636)

[1.1. Project Background 7](#_Toc21459637)

[1.2. Project Description 7](#_Toc21459638)

[1.3. Project Aims and Objectives 7](#_Toc21459639)

[1.4. Project Scope 7](#_Toc21459640)

[1.5. Thesis Roadmap 7](#_Toc21459641)

[2. Literature Review 8](#_Toc21459642)

[2.1. Introduction 8](#_Toc21459643)

[2.2. Research Topic 1 8](#_Toc21459644)

[2.3. Research Topic 2 8](#_Toc21459645)

[2.4. Existing Final Year Projects 8](#_Toc21459646)

[2.5. Conclusions 8](#_Toc21459647)

[3. Experiment Design 9](#_Toc21459648)

[3.1 Introduction 9](#_Toc21459649)

[3.2. Software Design 9](#_Toc21459650)

[3.3. Software Test plan 9](#_Toc21459651)

[3.4. Front-End 9](#_Toc21459652)

[3.5. Middle-Tier 9](#_Toc21459653)

[3.6. Back-End 9](#_Toc21459654)

[3.7. Conclusions 9](#_Toc21459655)

[4. Experiment Development 10](#_Toc21459656)

[4.1. Introduction 10](#_Toc21459657)

[4.2. Software Development 10](#_Toc21459658)

[4.3. Front-End 10](#_Toc21459659)

[4.4. Middle-Tier 10](#_Toc21459660)

[4.5. Back-End 10](#_Toc21459661)

[4.6. Conclusions 10](#_Toc21459662)

[5. Evaluation 11](#_Toc21459663)

[5.1. Introduction 11](#_Toc21459664)

[5.2. Software Evaluation 11](#_Toc21459665)

[5.3. Specific Evaluation 11](#_Toc21459666)

[5.4. Questionnaires and Interviews Evaluation 11](#_Toc21459667)

[5.5. Conclusions 11](#_Toc21459668)

[6. Conclusions and Future Work 12](#_Toc21459669)

[6.1. Introduction 12](#_Toc21459670)

[6.2. Conclusions 12](#_Toc21459671)

[6.3. Future Work 12](#_Toc21459672)

[Bibliography 13](#_Toc21459673)

# Table of Figures

[Figure 1 – Grocery Store simulation 11](#_Toc59054345)

[Figure 2 – Spread simulation 12](#_Toc59054346)

[Figure 3 – Live output 13](#_Toc59054347)

[Figure 4 – Case studies 13](#_Toc59054348)

[Figure 5 – Sample assets 18](#_Toc59054349)

[Figure 6 – Office environment 19](#_Toc59054350)

[Figure 7 - NavMesh 19](#_Toc59054351)

[Figure 8 – Prototype Agent 20](#_Toc59054352)

[Figure 9 – Movement Code 20](#_Toc59054353)

[Figure 10 - Architecture 22](#_Toc59054354)

[Figure 11 – System Diagram 22](#_Toc59054355)

[Figure 12 – View of simulation 24](#_Toc59054356)

[Figure 13 - GUI 24](#_Toc59054357)

[Figure 14 – Use Case 25](#_Toc59054358)

[Figure 15 – Class Diagram 26](#_Toc59054359)

# 1. Introduction

## Project Background

As the number of Coronavirus cases continue to grow worldwide, scientists and medical professionals from all over the world have been researching and studying the virus and its transmission to better understand and subsequently overcome it. Due to many real-life experiments being too risky to carry out, computer simulation has been an invaluable tool for developing further understanding of the virus and its transmission. This project involves creating a simulation of an environment in which transmission commonly occurs, an office.

There is a vast number of platforms available for developing in this field, but Unity stands out with its countless invaluable features and tools that enable swift and efficient development of real time simulations. The use of Unity also allows the use of various complex technologies, such as navigation mesh which creates a map of traversable areas in a scene and grants agents the ability to find the shortest path to their destination. A behaviour tree is another technology available in Unity which is a mathematical model of plan execution, meaning an artificially intelligent agent can switch between a set of tasks in a modular fashion. Unity also grants the ability to implement particle systems which can be used to simulate particles being expelled during breathing. Finally, Unity also has the entity component system, which is a new data-orientated design system which significantly boosts performance of the simulation if implemented correctly.

## Project Description

The purpose of this project is to simulate the environment of a populated office. The program starts off on a screen which allows the user to adjust certain variables which will affect the result.

Once the simulation is started, autonomous agents enter the building representing workers going about their daily work shift. The building has a navigation mesh which is utilised by the agents so they can path find through the office. At the start of the simulation, each agent goes to their respective desk, which is chosen at random at the start, and begins working. Agents will intermittently do various tasks such as retrieving a file or printing something off, and then return to their desks. This is due to each of the agents having a behaviour tree, so they have a set of tasks which they switch between.

One of the agents is infected with COVID-19 and is continuously spreading it throughout the office as the day goes on via a particle system which emulates breathing. The virus is spread by the particles emit, which can contaminate a surface or expose an agent if they collide with it. Agents also have a chance of becoming exposed if they touch another exposed agent or if they touch a contaminated surface. The chances of an agent becoming exposed when they come in contact with an infectious particle is affected by the what the user selected at the start of the simulation, and the figures used in these calculations are taken from a WHO backed study. (x)

As the simulation runs, the user can walk around the office to get a better view of the virus spreading, or they can look through the office security cameras. The user is also able to change the rate at which time passes, so they could have the simulation run at times ten speed to see the results faster.

Once the working hours set by the user have ended, the agents begin to leave the office. Once they all leave, a screen is displayed to the user with statistics from the simulation that was just run, and the user has the option to run the simulation again with different options.

### Factors the user can change:

**Working Hours:** The amount of time the users stay in the office.

**If healthy agents are vaccinated**: This alters the chances of a healthy agent becoming exposed.

**If healthy agents wear masks**: This alters the chances of a healthy agent becoming exposed.

**If infected agents wear masks**: This alters the number of particles emit by the infectious agent.

**Time scale:** The rate at which time passes in the simulation.

## Project Aims and Objectives

1. Identify and review suitable literature and other references relevant to this project
2. Describe some other software systems that are like this project
3. Undertake a thorough design process, including a methodology and detailed design
4. Develop a working software system using suitable technologies
5. Test and Evaluate the developed system
6. Critically reflect on the outcomes of this entire process

## Project Scope

This project allows users to view a COVID-19 simulation in real time and alter certain variables to see how they affect the transmission results. The simulation is made using Unity, and the environment in which the simulation takes place in is a populated building with autonomous agents using behaviour trees. Navigation Mesh is used to map out the walkable paths for the agents throughout this environment. The agents have human models and custom AI allowing them to go to their assigned desk and work, intermittently going to do various tasks around the office. Then once their working hours end, they leave the office.

Infected agents emit particles using Unity’s Particle System that leave surfaces contaminated and they can expose other agents to the virus based on their susceptibility. Agents can also become exposed from a contaminated surface or from getting too close to a different exposed agent. As the simulation is running, the user can walk around or look through the office security cameras. The user can alter the time scale to speed up the simulation, and they can also adjust variables that affect the result of the simulation. When the agents all leave the office, a screen is displayed with some statistics from the simulation that was just run, and the user can then restart the simulation with different variables.

## Thesis Roadmap

### Literature Review

In this chapter, a description of the main technologies and resources researched is presented, including academic papers, tutorials, books, and websites. The main technologies involved with the system are discussed, along with some other related research. It also looks at existing virus simulations made in Unity and previous final year projects with similarities to this project.

### Experiment Design

In this chapter, a prototype of the project is presented and discussed. It is developed in Unity and C# and makes use of the Unity Navigation Mesh. Challenges faces are also discussed.

### Experiment Development

In this chapter, a description of the methodology used in this project is presented, as well as a system overview. It also describes the design of both the front-end of the system, and the back-end design of the system.

### Evaluation

### Conclusions and Future Work

This chapter describes the development process that has been undertaken so far as well as the plans for future development. It also describes the Software Test planning that has already been undertaken, as well as the Testing that is planned.

# 2. Literature Review

## 2.1. Introduction

In this chapter a review of relevant research and other software is presented as it relates to the simulation system. First existing software that performs similar functions to this project are presented, and following that, the technologies be used in this system Other research including academic papers and web information are presented. Finally, two existing final year projects are discussed.

## 2.2. Alternative Existing Solutions to Your Problem

### Exploring new ways to simulate the coronavirus spread (1)

Released in May 2020, this Unity Blog is about a Coronavirus spread simulation which is developed in Unity and C#. The project contains a simulation of a grocery store, with customers coming and going to and from the store. Some customers are infected and can expose other customers to the virus if they are within a certain range for long enough. The project has a GUI at the side of the screen which allows the user to alter various parameters, apply the changes, and see how they affect the results which are also displayed on the GUI.



Figure 1 – Grocery Store simulation

### Software Features:

**Grocery Store Environment:** The project contains a simulated grocery store, with aisles, registers, entrances etc. The shoppers travel around this simulated store.

**Shoppers:** There are agents in the shape of capsules which represent shoppers. They follow certain routes throughout the store.

**Configurable parameters:** Parameters like exposure distance and transmission probability are adjustable using the sliders in the GUI on the right of the screen. Once the “Apply and Reset” button is pressed, the actual variables which are used in the simulation are updated accordingly, and the effects will be visible.

**Time scale:** The scale of the simulation can be adjusted using the GUI, allowing the user to choose how fast they would like time to go by in the simulation.

**Mapping:** The traversable routes are determined procedurally based on criteria including entrances and exits, whether certain sections are one way only, and making sure there are no collisions.

**Movement:** When shoppers spawn, they pick random traversable paths throughout the store. These paths start at the entrance, have random amounts of intermediate goals, and end at the exit.

**Exposure:** Shoppers spawn as either healthy or infectious. When infectious shoppers come close with other shoppers, they can expose them to the virus based on some set parameters. These shoppers are then set to exposed.

**Queuing:** Before each shopper approaches the registers, they check if there are any open registers, and then get queued accordingly based on the store policy parameters.

This grocery store simulation has many similar features to this project. The concept of having a GUI screen with configurable parameters is close to the GUI that this project has, although this project has project has the GUI screen only at the start. A lot of the other features are rather similar too, such as having agents walk throughout the simulated environment with a chance of becoming exposed. The logic of having infectious agents exposing healthy agents to the virus is the same, although this project is much more in depth, having the actual particles emit from breathing being the carrier of the virus rather than just a simple collision behaviour. The grocery store itself is also similar to the simulated office in which this project takes place in, although this project offers a 3d space in which the player can move around in rather than just a top-down view.

Both the grocery store project and this project are made completely in Unity and C#, so the technologies used are closely related, although this project makes use of some more complicated technologies such as behaviour trees for AI and particle systems to emulate breathing.

How coronavirus spreads through a population and how we can beat it (2)

Published in early 2020, this article presents a simulation of the spread of certain viruses throughout a population of people. It allows the user to adjust some parameters using the sliders at the top, and then shows how the virus would spread over a period of time. As well as allowing the user to adjust these parameters, they can also select one of the case studies and see a visualisation of the spread using statistics from the actual case study.



Figure 2 – Spread simulation



Figure 3 – Live output

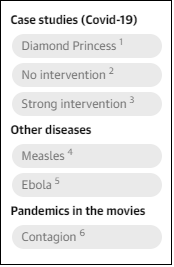


Figure 4 – Case studies

### Software Features:

**Infectious indicators:** Members of the population start off as yellow which indicates a healthy person. Red indicates they are infected with the virus, and purple represents people who have died from the virus.

**Adjustable parameters:** As seen in the top of the screenshot, the user can move the sliders to change the parameters of the simulation. They can then see of visualisation of how the chosen values would affect the results.

**Case studies:** The user can select from a short list of case studies to see a visualisation of the spread that took place during these case studies.

**Utilising real statistics:** If a case study is chosen by the user, the simulation will run using parameters taken from real life statistics.

**Displaying results:** As the simulation runs through the phases, it updates the visualisation of the population with the corresponding colours. It also displays the numbers after each phase and displays the stage on a chart as it updates.

This population spread simulation also has numerous similarities to this project. Both projects take some statistics from real life and use them as parameters for the simulation, and also allow the user to adjust variables and see the results. They also both focus on visualising the spread of the virus, although the population spread simulation went in a completely different direction, focusing on spread over a long amount of time, and as a result it is much less detailed than this project is and does not touch on the transmission methods of the virus, in turn making it a somewhat simple system.

## 2.3. Technologies you’ve researched

### Godot (3)

Godot is an open-source game engine that is known for its node-based architecture and object-oriented API. It was released under the MIT license and runs on most operating systems. It has many useful tools for game development, such as the scene tree editor, the script editor, a script debugger, etc. It also has an asset store from which numerous plugins can be downloaded to extend functionality. Godot contains engines for physics and lighting and many other mechanics that make game development swift and efficient.

Godot is a useful tool for developing projects such as simulations due to its long list of features, although it is nowhere near as widespread or as popular as Unity, therefore there is much less documentation and tutorials available online for it.

### Unity (4)

Unity is cross platform game engine that is widely used for a variety of applications. It was developed by Unity Technologies and released in 2005. The Unity asset store has an ever-growing catalogue of assets and tools which make project development with Unity considerably faster than many alternatives. Unity is also full of useful tools such as a debugger, a script editor, a scene editor etc.

It is extremely accessible and used globally, so there is a surplus of tutorials and online resources to learn from. These resources include plenty of sample projects full of detailed documentation which allows users to develop a detailed understanding of the underlying concepts in these projects. It also excels in real-time simulation, which is perfect for this project.

### Unity Render Pipelines (5)

In Unity, a project can use one of various render pipelines. The render pipeline performs a set of operations which entail taking the contents of a scene and displaying them on the screen. Different render pipelines have different capabilities and performance, so it depends on the nature of the project. The built-in render pipeline is the default render pipeline for Unity. It has limited customisation, for general purposes. There are other render pipelines available which focus more on graphics, but this project does not centre on graphics, so it is using the built-in render pipeline.

### Unity Navigation Mesh (6)

NavMesh (Navigation Mesh) is a tool for mapping out the traversable areas of an environment and the paths that agents can take through this environment. The process entails rendering a mesh of the walkable areas, allowing agents to determine the shortest possible paths between locations. This helps AI look more natural as it travels through an environment. This project has autonomous agents following random paths through the course of the simulation, so navigation mesh was an obvious choice to assist in the pathfinding.

### ParticleSystem (7)

ParticleSystem is Unity’s in-built implementation of a particle system, containing a vast amount of properties and methods which can be altered to get different effects. When properties are set, they are passed immediately into native code to give the best performance. ParticleSystem is used to display a wide array of items such as fire, liquids, explosions, gasses etc.

This simulation uses ParticleSystem to implement the actual virus particles being expelled from infectious agents which is the method of virus transmission.

## 2.4. Other Research you’ve done

Domain specific research

## 2.5. Existing Final Year Projects

## 2.6. Conclusions

# 3. Experiment Design

## 3.1 Introduction

## 3.2. Software Methodology

## 3.3. Overview of System

Include a diagram

## 3.4. Front-End

Including screen prototypes and Use Cases

## 3.5. Middle-Tier

## 3.6. Back-End

Including ERDs, and maybe ISDs

## 3.7. Conclusions

# 4. Experiment Development

## 4.1. Introduction

## 4.2. Software Development

## 4.3. Front-End

## 4.4. Middle-Tier

## 4.5. Back-End

## 4.6. Conclusions

# 5. Testing and Evaluation

## 5.1. Introduction

## 5.2. System Testing

## 5.3. System Evaluation

## 5.4. Conclusions

# 6. Conclusions and Future Work

## 6.1. Introduction

## 6.2. Conclusions

## 6.3. Future Work

# Bibliography