

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

Interim Report

DT211C

BSc in Computer Science (Infrastructure)

**Kyle Heffernan**

C17444434

Bryan Duggan

School of Computer Science

Technological University, Dublin

**14/12/2020**

Abstract

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 real life experiments being too risky to carry out, computer simulation has been an invaluable tool for developing further understanding.

Simultaneously, most peoples everyday lives.

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: 14/12/2020

Acknowledgements

I would like to thank my supervisor Bryan Duggan for his constant support and guidance throughout this project.

Table of Contents

[1. Introduction 7](#_Toc58250492)

[1.1 Project Background 7](#_Toc58250493)

[1.2. Project Description 7](#_Toc58250494)

[1.3. Project Aims and Objectives 7](#_Toc58250495)

[1.4. Project Scope 7](#_Toc58250496)

[1.5. Thesis Roadmap 8](#_Toc58250497)

[2. Project Background 9](#_Toc58250498)

[2.1 Introduction 9](#_Toc58250499)

[2.2. Alternative Existing Solutions 9](#_Toc58250500)

[2.3. Technologies You’ve Researched 9](#_Toc58250501)

[2.4. Other Research you’ve done 10](#_Toc58250502)

[2.5. Existing Final Year Projects 10](#_Toc58250503)

[2.6. Conclusions 10](#_Toc58250504)

[3. Prototype Design 11](#_Toc58250505)

[3. Project Design 11](#_Toc58250506)

[3.1 Introduction 11](#_Toc58250507)

[3.2. Software Methodology 11](#_Toc58250508)

[3.3. Overview of System 12](#_Toc58250509)

[3.3.1. Technical Architecture 12](#_Toc58250510)

[3.3.2. System Diagram 13](#_Toc58250511)

[3.3.3. Requirements Table 14](#_Toc58250512)

[3.4. Front-End 16](#_Toc58250513)

[3.4.1. Key Screens 16](#_Toc58250514)

[3.4.2 Use Cases 19](#_Toc58250515)

[3.5. Middle-Tier 22](#_Toc58250516)

[3.6. Back-End 23](#_Toc58250517)

[3.6.1 Entity Relationship Diagrams 23](#_Toc58250518)

[3.6.2 Interaction Sequence Diagrams 23](#_Toc58250519)

[3.6.3 Class Diagrams 23](#_Toc58250520)

[3.7. Conclusions 25](#_Toc58250521)

[4. Future Work 26](#_Toc58250522)

[4.1 Introduction 26](#_Toc58250523)

[4.2. System Development 26](#_Toc58250524)

[4.2.1. Prototype Development 26](#_Toc58250525)

[4.2.1. Production Development 27](#_Toc58250526)

[4.3. System Testing 27](#_Toc58250527)

[4.3.1. Prototype Testing 27](#_Toc58250528)

[4.3.1. Production Testing 28](#_Toc58250529)

[4.4. Project Plan 28](#_Toc58250530)

[4.5. Conclusions 30](#_Toc58250531)

[Bibliography 31](#_Toc58250532)

# 1. Introduction

## 1.1 Project Background

The purpose of this project was to design a simulation of the transmission of COVID-19 between people in a populated environment. As Coronavirus became a sizeable factor of everyday life for most people[x], I wanted to make a simulation to assist in visualizing how easily it can be spread in a populated environment.[x] I believe it will be beneficial for greater understanding of the transmission of COVID-19, as observing it spreading in real time will highlight the severity of the virus in everyday conditions.

## 1.2. Project Description

## The Project includes simulated environments of populated buildings, such as a restaurant or office. There are autonomous agents walking around the buildings representing people going about their daily lives. The building has a navigation mesh to aid the agents[x], and they path find through this mesh, doing random tasks and having various interactions based on their role/job. Some agents are infected with coronavirus, so as they walk around, they breath/cough/sneeze intermittently creating a particle system from their mouths dispersing outwards.

## The particles in this particle system[x] have a chance of being infected, and if they are, they expose whatever they come in contact with to the virus. If they encounter a surface, it becomes contaminated with the chance of contaminating other agents. If the particles come in contact with another agent, there is a chance they will be exposed to the virus as well.

## The maths for the chance of particles being infected, how long they stay on surfaces and the chance of another agent getting exposed to the virus are all taken from medical papers[x]

There is a graphical user interface, allowing the user to alter numerous factors that affect the results of the simulation.

Factors the user can change:

## **Viral Load:** The number of particles coming from the particle system that carry the infection.

## **Masks:** Reduce the number of particles coming from each agent.

## **Number of Infected Agents**: The number of agents infected at the start of the simulation.

* **Susceptibility Factors:** Factors such as vaccinations, age, vitamin deficiencies, etc.
* **Time scale:** The rate at which time passes in the simulation.

[photo of GUI design]

## 1.3. Project Aims and Objectives

* **Real Time Simulation**

The user can watch the simulation run in real time and observe the virus spreading between agents.

* **Change Variables**

The user can adjust certain variables that alter the results of the simulation.

* **Utilize Unity’s Navigation Mesh System**

A navigation mesh is a data structure which helps agents path find through complicated spaces by returning a collection of walkable nodes. The autonomous agents in this simulation will be able to path find through the building with the help of the navigation mesh.

* **Autonomous Agents**

The agents follow certain behaviours as they path find throughout the building following various feasible paths.

* **Use Data from Medical Papers**

The methods of transmission and exposure will make use of statistics taken from medical papers/journals

* **Utilize Unity’s Entity Component System**

The entity component system is a new approach to designing Unity projects which entails using data orientate design rather than the usual object orientated approach. It makes use of the C# Job System and multithreading to optimize performance.

* **Display Results**

The user can see statistics about the number of agents exposed to the virus as the simulation runs.

* **Utilize Unity’s Particle System**

The infected agents will be emitting infected particles from their mouths with the use of Unity’s particle system.

## 1.4. 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 walking around. Navigation Mesh is used to map out the walkable paths for the agents throughout this environment. The agents have simple designs and basic AI allowing them to walk through the building performing appropriate tasks. Infected agents emit particles using Unity’s Particle System that leave a surface infected, or they expose other agents to the virus based on their susceptibility. 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 such as susceptibility factors.

[photo of design]

## 1.5. Thesis Roadmap

## Project Background

In this chapter, a description of the main technologies and resources researched is presented, including academic papers, tutorials, books, and websites. The main technologies discussed are Game Engines, Unity Render Pipelines, Unity Navigation Mesh, Unity Entity Component System, Unity Particle System, Pathfinding AI, the C# Job System, and finally COVID-19 medical papers. It also looks at existing virus simulations made in Unity and previous final year projects with similarities to this project.

## Prototype Design

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

## Project Design

it presents a description of the methodology used in this project, as well as a system overview. It also describes in detail the design of both the front-end of the system, and the back-end design of the system.

## Future Work

it 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. Project Background

## 2.1 Introduction

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 non-stop 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 affects of various countermeasures. These statistics can be utilised in the simulation as parameters to give a scientifically accurate result.

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 some complex technologies such as the entity component system, which is a new data-orientated design system which significantly boosts performance of the system if implanted correctly.

As Coronavirus continues to grow, so does misinformation about it on social media. While some basic guidelines are given to the public about countermeasures they can take to prevent transmission, the results of these countermeasures are not easy to identify and can lead to people not trusting their effectiveness. This simulation is a practical solution to this, using scientifically accurate figures to visualise transmission and the effectiveness of various countermeasures.

## 2.2. Alternative Existing Solutions

### Exploring new ways to simulate the coronavirus spread

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.



### 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 on screen with configurable parameters is especially close to the GUI that this project has. A lot of the other features are rather similar too, such as having agents spawn and walk through random yet traversable paths in the environment. The logic of having infectious agents exposing healthy agents to the virus is the same, although this project is much more in depth, accounting for infected particles and many more adjustable parameters. The grocery store itself is also similar to the simulated environment in which this project takes place in.

Both the grocery store project and this project are made completely in Unity and C#, so the technologies used are very closely related.

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

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.



### 

### 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 statistics from real life cases and use them as parameters for the simulation, but also allow the user to adjust them and see the results. They also both focus on visualising the spread of the virus, although the population spread simulation is a lot less detailed than this project 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

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

Unity is cross platform game engine developed by Unity Technologies 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

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

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

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.

### Entity Component System

Entity Component System (ECS) is a new way to develop in Unity that focuses on data-oriented design rather than object-oriented design. It breaks the project into 3 sections:

**Entities** – The actual things in your simulation

**Components** – The data associated with these entities but organised by the data rather than by entity.

**Systems** – The behaviours that update the component data. For example, A movement system would update positions of moving entities by their velocity and time passed.

Projects using ECS have greatly improved performance, making it an extremely useful instrument for simulations with a lot going on.

### C#

C# is a modern object oriented, component orientated programming language. It was developed by Microsoft in 2000 as part of its .NET initiative, and approved as an international standard in 2002. Like Unity, due to its widespread use, there is a vast amount of resources available online to assist in understanding the underlying concepts. Applications made with C# are generally quite robust due to its many supportive features. Exception handing is a feature of C# which allows the detection and recovery of errors. Garbage collection is another useful feature which automatically reclaims unused memory.

It is the language that Unity scripts are mainly written in, so the coding in this project is mostly done in C#.

### C# job system

The Unity C# Job System allows users to write multithreaded code which interacts well with Unity. It integrates with Unity’s native job system, so user-written code and Unity share worker threads. This ensures that there are not more threads than CPU cores. This multithreaded code can greatly improve performance of the project. The C# job System works well with the Unity Entity Component System due to its efficient way of writing code.

The C# Job System improves performance, which is important when the simulation contains some performance heavy behaviours.

## 2.4. Other Research you’ve done

COVID-19 Transmission Research

COVID-19 Closed environments

Data Visualisation

## 2.5. Existing Final Year Projects

Traffic Simulation

Data Visualisation

## 2.6. Conclusions

# 3. Prototype Design

**As least 6 pages, but as many as you like (but lots of diagrams, which count towards the page total).**

SNAPS asset pack

# 4. Project Design

## 4.1 Introduction

## 4.2. Software Methodology

## 4.3. Overview of System

### 4.3.1. Technical Architecture

* **Include a diagram – One of these???**

### 4.3.2. System Diagram

**Another Diagram of system, maybe?**

### 4.3.3. Requirements Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Priority** | **Version** |
| 1 | Database | Set up database for all stored data | High | 1.0 |
| 2 | Registration | Ability to register user through app | High | 1.0 |
| 3 | Login | Ability for registered user to log in. | High | 1.0 |
| 4 | Location Tracking | App uses user location. | High | 1.1 |
| 5 | Message posting | Users ability to post messages | High | 1.1 |
| 6 | Post visibility by area | To limit posts to be visible to the area they are relevant to. | High | 1.1 |
| 7 | Post access limited by ownership and area | User’s own post should be visible, but other posts access is limited by area. | High | 1.1 |
| 8 | User voting | User’s should be able to promote other posts that they think is important or high quality. | Low | 1.5 |
| 9 | User preference | Content should be limited by the user’s own preference settings. | High | 1.3 |
| 10 | Posts Sorting | Ability to sort posts by different factors, i. e. most popular, most interactions, most important | Low | 1.5 |
| 11 | Post access relevance to area scaling | User’s should be able to see content relevant to for other locations depending on it’s importance e.g. emergencies | Medium | 1.2 |
| 12 | Register as a club or organization | Users should be able to create accounts for clubs or organizations | Low | 1.5 |
| 13 | Users preferences hobbies | Preferences should include hobbies | Low | 1.6+ |
| 14 | Register for clubs or organizations | Users should be able register membership for clubs or organisations | Low | 1.6+ |
| 15 | Editing posts | Users should be able to edit their posts after they have been made. | Medium | 1.3 |
| 16 | Users should be able to see the original post prior to editing | Post history should be available to be viewed. | Medium | 1.2 |
| 16 | User downvoting | Users should be able to downvote content they think isn’t relevant. | Low | 1.6 |
| 17 | Report content | Users should be able to report content they think is inappropriate | Low | 1.6+ |
| 18 | Admin UI | Admin accounts should have it’s own UI | Low | 1.5 |
| 19 | Admin remove | Admins should be able to remove posts | Low | 1.5 |
| 20 | Quick Glance | Users should be able to see the immediate most relevant post in an area without logging in | Medium |  |
| 21 | Home View | Users should see content relevant to their home without being in that location. | Medium |  |
| 22 | Work view | Users should see posts from their work area without being at that location | Medium |  |
| 23 | Private zone | A specific building or location should have it’s own private thread so only people with access can view it | Low |  |
| 24 | Draggable menu | By having a side swap menu for post making users can avoid making accidental posts. | Low |  |
| 25 | Password Reset | Users can reset password | High |  |
| 26 | Menu buttons | Menu buttons bring user to correct pages | High |  |
| 27 | Firebase authentication | Firebase authenticates logins correctly | High |  |
| 28 | Push notification feature | App can send push notifications to user. | Low |  |
| 29 | User preferences simple to manage | Users can set their preferences without a lot of effort | High |  |
| 30 | Users can filter out posts that contain certain words | Users can blacklist words in their preferences, they won’t see posts that contain those words | Medium |  |
| 31 | Scale access | Users should be able to increase and decrease the range of the posts they see | Medium |  |
| 32 | User profiles private | A user’s profile should only be visible by that user | High |  |
| 33 | User post score | Users should be able to see how popular their posts are on their private profile | Low |  |
| 34 |  |  |  |  |

## 4.4. Front-End

### 4.4.1. Key Screens

Including

### 4.4.2 Use Cases

## 4.5. Middle-Tier

## 4.6. Back-End

### 4.6.1 Entity Relationship Diagrams

### 4.6.2 Interaction Sequence Diagrams

### 4.6.3 Class Diagrams

## 4.7. Conclusions

# 5. Future Work

## 5.1 Introduction

## 5.2. System Development

### 5.2.1. Prototype Development

### 5.2.2. Production Development

## 5.3. System Testing

### 5.3.1. Prototype Testing

### 5.3.2. Production Testing

## 5.4. Project Plan

## 

## 5.5. Conclusions

# 

# Bibliography