

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

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

As Coronavirus continues to grow, so does misinformation about it on social media. While are 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 scientifically accurate figures to visualise transmission and the effectiveness of various countermeasures.

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 continuous support and guidance throughout this project, and my friends for keeping me motivated and on track this year with our countless online study sessions. I would also like to thank my family for always keeping me grounded, and finally Danniella for her constant love and support.

Table of Contents

[Table of Figures 7](#_Toc59054279)

[1. Introduction 8](#_Toc59054280)

[1.1 Project Background 8](#_Toc59054281)

[1.2. Project Description 8](#_Toc59054282)

[1.3. Project Aims and Objectives 9](#_Toc59054283)

[1.4. Project Scope 9](#_Toc59054284)

[1.5. Thesis Roadmap 9](#_Toc59054285)

[Project Background 9](#_Toc59054286)

[Prototype Design 9](#_Toc59054287)

[Project Design 10](#_Toc59054288)

[Future Work 10](#_Toc59054289)

[2. Project Background 10](#_Toc59054290)

[2.1 Introduction 10](#_Toc59054291)

[2.2. Alternative Existing Solutions 10](#_Toc59054292)

[Exploring new ways to simulate the coronavirus spread (1) 10](#_Toc59054293)

[Software Features: 11](#_Toc59054294)

[Software Features: 13](#_Toc59054295)

[2.3. Technologies Researched 14](#_Toc59054296)

[Godot (3) 14](#_Toc59054297)

[Unity (4) 14](#_Toc59054298)

[Unity Render Pipelines (5) 15](#_Toc59054299)

[Unity Navigation Mesh (6) 15](#_Toc59054300)

[ParticleSystem (7) 15](#_Toc59054301)

[Entity Component System (8) 15](#_Toc59054302)

[C# (9) 15](#_Toc59054303)

[C# job system (10) 16](#_Toc59054304)

[2.4. Other Research Done 16](#_Toc59054305)

[COVID-19 Transmission Research 16](#_Toc59054306)

[Data Visualisation 16](#_Toc59054307)

[2.5. Existing Final Year Projects 17](#_Toc59054308)

[Traffic Simulation System for Driverless Vehicles by Fionn McGuire. 17](#_Toc59054309)

[Irish Crime Data Visualisation by Max Curtis. 17](#_Toc59054310)

[2.6. Conclusions 17](#_Toc59054311)

[3. Prototype Design 18](#_Toc59054312)

[3.1 Introduction 18](#_Toc59054313)

[3.2 Prototype Development 18](#_Toc59054314)

[3.3 Issues 21](#_Toc59054315)

[3.4 Conclusion 21](#_Toc59054316)

[4. Project Design 21](#_Toc59054317)

[4.1 Introduction 21](#_Toc59054318)

[4.2. Software Methodology 21](#_Toc59054319)

[4.3. Overview of System 22](#_Toc59054320)

[4.3.1. Technical Architecture 22](#_Toc59054321)

[4.3.2. System Diagram 22](#_Toc59054322)

[4.3.3. Requirements Table 22](#_Toc59054323)

[4.4. Front-End 23](#_Toc59054324)

[4.4.1. Key Screens 23](#_Toc59054325)

[4.4.2 Use Cases 25](#_Toc59054326)

[4.6. Back-End 25](#_Toc59054327)

[4.6.1 Class Diagrams 26](#_Toc59054328)

[4.7. Conclusions 27](#_Toc59054329)

[5. Future Work 27](#_Toc59054330)

[5.1 Introduction 27](#_Toc59054331)

[5.2. System Development 27](#_Toc59054332)

[5.2.1. Prototype Development 27](#_Toc59054333)

[Features implemented in prototype: 28](#_Toc59054334)

[5.2.2. Production Development 28](#_Toc59054335)

[5.3. System Testing 28](#_Toc59054336)

[5.3.1. Prototype Testing 28](#_Toc59054337)

[5.3.2. Production Testing 29](#_Toc59054338)

[Test 29](#_Toc59054339)

[Expected Result 29](#_Toc59054340)

[Unity Test Framework (20) 29](#_Toc59054341)

[5.4. Project Plan 29](#_Toc59054342)

[5.5. Conclusion 31](#_Toc59054343)

[Bibliography 32](#_Toc59054344)

# 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

## 1.1 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 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.

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.

## 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, 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 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.

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.

## 1.3. 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

## 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.

## 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 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.

## Prototype 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.

## Project Design

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.

## 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. Project Background

## 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

### 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 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 (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

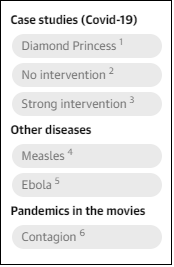


Figure 4 – Case studies

### 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 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 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.

### Entity Component System (8)

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# (9)

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 (10)

The Unity C# Job System allows users to have multithreaded code in their projects. 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 Done

### COVID-19 Transmission Research

There has been a great amount of research done in the last year regarding the transmission of COVID-19, and numerous factors have been found to influence the probability of transmission. Physical distancing has been shown to reduce transmission rates (11) as the infected particles can only be expelled a certain amount. (12) Masks have also been shown to reduce the particles expelled from an infectious person. Factors such as vitamin D levels (13) or age can determine a person’s susceptibility to the virus due to the strength of their immune system.

Closed environments have also been found to be a contributor to secondary transmission and can lead to superspreading events. (14)

This simulation has a vast number of parameters that the user can change and see how they alter results. The parameters this system uses have been chosen as they have been shown to have an effect on transmission.

### Data Visualisation

The use of images and simulations to visualise data has been shown to help develop a greater understanding and comprehension of data than ever before. (15) Many people struggle to truly grasp the implications of raw data without some useful kind of visualisation. Some methods of visualisation do a much better job than others though.

Game techniques and mechanisms such as real time simulations have been shown to aid in the understanding of certain topics as they are a more engaging form of learning. (16)

## 2.5. Existing Final Year Projects

### Traffic Simulation System for Driverless Vehicles by Fionn McGuire.

A traffic simulation system for the deployment of driverless vehicles in modern day society by using Unity3D. The platform utilizes an interactive OSM map of Manhattan populated with both drivers and driverless vehicles. The vehicles generate a route to follow while perpetually responding to changes in the environment.

This traffic simulation is also made in Unity and contains many agents which populate an environment and have a set of behaviours to follow. This project has a lot of similar mechanics, as it is also simulating multiple agents in an environment which have a set of behaviours and have interactions.

### Irish Crime Data Visualisation by Max Curtis.

A system to allow for the visualization of Ireland’s crime statistics. This data is an untapped resource in its current state. This project is an application that helps users understand a mountain of data using data visualisation techniques.

This data visualisation application has some similar concepts to this project, the main one being that the application helps users develop a greater understand of the available data. There is a vast amount of data available online about Coronavirus and its transmission but having a good visualisation can help users truly comprehend what the data implies.

## 2.6. Conclusions

In this chapter, the research done for the project was shown and presented. This research included similar existing systems, technologies related to the simulation, Coronavirus transmission research, and data visualisation research. Finally, two similar existing fourth year projects were discussed. When the project was being planned, these technologies were researched and reviewed thoroughly to determine what would be used.

# 3. Prototype Design

## 3.1 Introduction

Following on from the existing software and research papers that were reviewed previously, in this chapter the design of the prototype of the simulation is presented and discussed, as well as some issues that were encountered.

## 3.2 Prototype Development

The prototype for this project is made in Unity 2019 and C#, using the built-in render pipeline. The Unity project which the prototype is being developed in is saved in a GitHub repository, so important files are backed up in case of system failures, progress is easily tracked, and version control is kept in check.

The environment of the prototype takes advantage of the Snaps asset pack prototype environment from the Unity asset store. (17) Snaps prototype asset packs are collections of prefabs made to real-world scale to speed up the process of environment design. The packs contain a vast collection of themed assets that are modular and customizable. The assets snap together on all three axes making scene development much faster.



Figure 5 – Sample assets

Each prototype pack available follows a specific theme, and the theme utilised by this project prototype is office, as it is a common location for virus transmission occurrences.



Figure – Office environment

With this sample office set up, the navigation mesh is implemented. To do this, objects that agents should be able to walk on (E.g. floor, stairs) are set to “navigation static”. They are then all used in a process called “baking” which returns all of the walkable surfaces in the environment. This walkable area is highlighted blue in the scene, providing the correct settings are selected.



Figure 7 - NavMesh

With the navigation mesh set, agents can traverse it easily. A prototype agent has been put in the scene that can travel throughout the office environment using the navigation mesh.

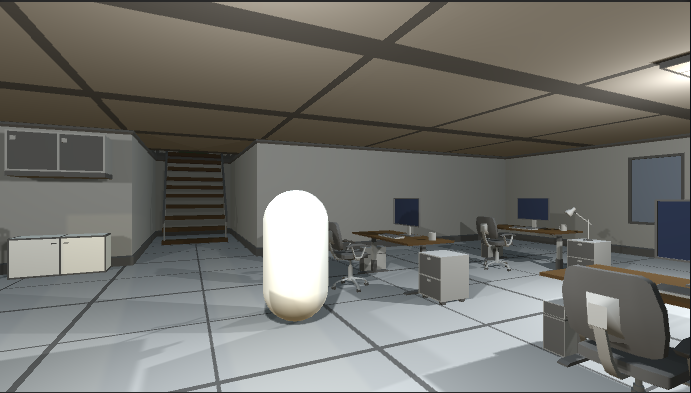


Figure 8 – Prototype Agent

The user can move the agent throughout the scene by left clicking any location on the screen, and the agent will get as close to the clicked location as possible via the navigation mesh. This is implemented using a simple ai script. The script makes a raycast from the mouse click location and sets that location as the agent’s destination. This is update method, so it checks every frame for a new mouse click.



Figure – Movement Code

There user also spawns as an agent themselves and can move throughout the office using keyboard controls. This was implemented with a simple player movement script, and by attaching the main camera to the player. This feature was added to allow quick and easy movement throughout the office to assist with testing certain future features.

## 3.3 Issues

At the beginning of prototype development, the prototype was planned to be developed using the Entity Component System (ECS) approach. After a substantial amount of research into ECS done, and many tutorials followed, development was started but numerous issues arose. Due to ECS still being in beta, a lot of online documentation was out of date and contradicted other available resources.

Eventually, a decision was made to make ECS integration a lower priority for the system, focusing on getting other more important features working first, and then attempting to integrate the system with ECS having a greater understanding of the general concepts involved.

## 3.4 Conclusion

In this chapter the design of the prototype of the simulation was presented, and the underlying concepts and features implemented were discussed as well as issues that came up during development.

# 4. Project Design

## 4.1 Introduction

In this chapter the design of the project and simulation are discussed. First the methodology used in this project will be outlined and following this a discussion of the technical architecture will be presented. The front-end design of the system will be presented next, showing the 2 key screens of the simulation. The back-end is also discussed with Class Diagrams to show the design of the objects.

## 4.2. Software Methodology

Agile software methodologies focus on continuous delivery of valuable software, and the primary measure of progress is working software. (18)

The software methodology this project uses is agile scrum. Agile scrum focuses on dividing the project into sprints, which are short timed periods in which an amount of work is set to complete, generally focusing on a specific feature of the project. Before each sprint, it is planned what work will be delivered from the sprint, and how that work will be achieved. One feature of the scrum methodology is regularly reflecting on work done and learning from it, in turn becoming more efficient as behaviour is adjusted accordingly. (19)

Scrum works well for projects with many important features, so it is perfect for this project. This project contains numerous important features which can be developed and implemented during these sprints, and then further improved with future iterations. This approach allows for many core features to be implemented successfully with some complexity, and then they can be improved and fleshed out later with less important features.

The waterfall model is a good example of a software methodology that would not work with this project. The waterfall model entails running through the entire project in a single iteration, never going back and making modifications or changes. This would not work as features of this project need to be iteratively designed, with a simple design working first, and then after some testing, reflecting, and further development, they can be revisited with a greater understanding of requirements and design.

## 4.3. Overview of System

### 4.3.1. Technical Architecture

As this project is entirely in Unity and C#, it’s technical architecture is a standalone system.



Figure 10 - Architecture

### 4.3.2. System Diagram



Figure 11 – System Diagram

### 4.3.3. Requirements Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Priority** | **Version** |
| 1 | Real Time Simulation | The user can watch the simulation run in real time and observe the virus spreading between agents. | High | 1.0 |
| 2 | Change Variables | The user can adjust certain variables that alter the results of the simulation via the GUI. | High | 1.1 |
| 3 | Utilize Unity’s Navigation Mesh System | The autonomous agents in this simulation will be able to path find through the building with the help of the navigation mesh. | High | 1.0 |
| 4 | Autonomous Agents | The agents follow certain behaviours as they path find throughout the building following various feasible paths. | High | 1.0 |
| 5 | Use Data from Medical Papers | The methods of transmission and exposure will make use of statistics taken from medical papers/journals | High | 1.2 |
| 6 | Display Results | The user can see statistics about the number of agents exposed to the virus as the simulation runs. | High | 1.1 |
| 7 | Utilize Unity’s Particle System | The infected agents will be emitting infected particles from their mouths with the use of Unity’s particle system. | High | 1.3 |
| 8 | GUI with sliders | A GUI will be displayed to the user containing sliders to alter variables and display some real time results. | Low | 1.1 |
| 9 | Time Scale | The user can alter the time scale via the GUI to have the simulation run faster or slower depending on their preferences. | High | 1.4 |
| 10 | Infectious agents | Agents will either be healthy, infectious or exposed. Most spawn as healthy and can become exposed if they come in contact with the virus. | High | 1.1 |
| 11 | Simulated environment | The project takes place in a simulated environment of a location in which transmission would occur. | High | 1.0 |
| 12 | Utilize Entity Component System | The entity component system is a data-oriented way of programming which significantly increase performance. | Low | 1.5 |
| 13 | Utilize C# Job System | The C# Job System would allow for Scripts and certain processes to be multithreaded. | Low | 1.5 |

## 4.4. Front-End

### 4.4.1. Key Screens

The project contains 2 main key screens. One of which being the actual game view of the 3D simulation. With this, the user will able to watch in real time as the agents walk around the environment, possibly being exposed to transmission of the virus.



Figure 12 – View of simulation

The second key screen of the project is the interactive GUI in which the user can adjust various parameters of the simulation. This screen is filled with buttons and sliders which adjust the actual parameters used in the simulation.



Figure 13 - GUI

### 4.4.2 Use Cases

Below, a simple use case for this system is shown. The users primary goal would be to run the simulation and observe the results, altering parameters and noticing how they affect the output.



Figure 14 – Use Case

## 

## 4.6. Back-End

### 4.6.1 Class Diagrams

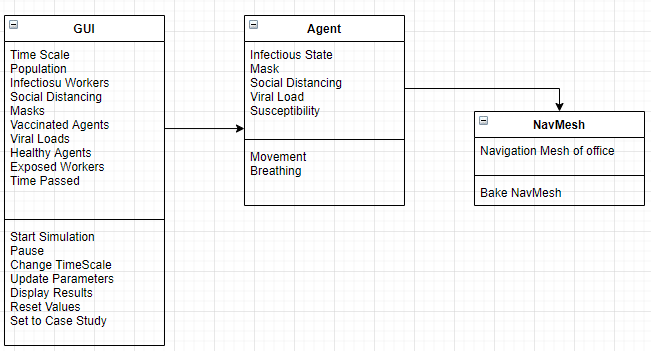


Figure 15 – Class Diagram

## 4.7. Conclusions

In this chapter, the design of the simulation system was presented. First, the agile scrum methodology was discussed as the approach to be used in this project. Following this, the technical architecture of the system was presented.

The front-end design of the system was presented next, showing the 2 key screens of the simulation. The back-end was also discussed with a class diagram to show the design of the objects.

# 5. Future Work

## 5.1 Introduction

This chapter presents the development and testing process that will be undertaken as part of this project as well as a project plan that highlights the key challenges ahead. The development section comprises of two main sub-sections, the first covers work that has already been done in terms of the development process, describing the extent of the prototype system that has built so far, and what challenges were encountered. The second sub-section explains future development plans. The testing section also comprises of two main sub-sections, the first covers the testing work done to date. The second section will outline any potential software testing libraries that can be used in the development environment, and what functionality it would be important to test. Finally, a project plan is presented.

## 5.2. System Development

This section outlines plans for the development of the simulation system, as well as all the work done so far. The first section describes the prototype development process that has been undertaken so far, and the second section discusses future development plans.

### 5.2.1. Prototype Development

As seen in chapter 3, the current system prototype has a small amount of the base system features implemented. The Unity project has been created and is saved to a GitHub repository for version control and to prevent losing work in the case of any technical issues. In this Unity project, a prototype office environment has been implemented with the use of the Snaps asset pack, which is a package from the Unity asset store containing some sample assets to help with scene building. A navigation mesh has been made from all of the walkable areas of the environment. An agent has been placed in the scene, and travels to wherever the user clicks using a simple movement script that allows it to traverse the navigation mesh. The user spawns in as an agent themselves and can also travel throughout the scene using keyboard controls.

### Features implemented in prototype:

* Office Environment in which the simulation takes place
* Navigation Mesh of environment implemented
* Agent which traverses navigation mesh
* Player can control an agent to move throughout the scene

### 5.2.2. Production Development

The final system will differ drastically from the prototype, with many more key features integrated. The click to move agent feature will be removed and replaced by ai with random but coherent paths throughout the office.

The exposure mechanic will also be integrated, with infectious agents spreading infected particles and exposing healthy agents to the virus.

This will be implemented with the help of Unity’s particle system and be in line with data collected from real life case studies and medical papers.

The agents will have various parameters which affect the probability of transmission.

The GUI will be implemented, displaying results as the simulation runs, and allowing the user to alter numerous parameters of the simulation.

The time scale feature will be implemented, allowing the user to change the speed that time passes in the simulation.

The system will be integrated to ECS, making the simulation much mor efficient and improving performance.

The C# Job System will be utilised to take advantage of it’s multithreading capabilities, improving overall performance.

## 5.3. System Testing

This section outlines plans for the testing of the simulation system. The first section describes the prototype testing process that has been undertaken so far, and the second section discusses future testing plans, including software testing tools.

### 5.3.1. Prototype Testing

Due to the system prototype being done made mainly in Unity, there were not many options for conventional testing methods.

One of the methods available is for testing scripts is unit testing, which involves ensuring that methods produce their expected outputs by running unit tests on them with an input and expected output. If the output matches the expected output, then the method has passed the test. While this works for some scripts, it does not cover the rest of the Unity project. Fortunately, with the way Unity is designed with a scene view, game view, debugger, and console, it is easy to run a project with expectations of how things should run, and then recognise any bugs or problems and determine what is causing them. This can be done with a third party, to get a different perspective on how system components should behave.

This was the testing method used during prototype development.

### 5.3.2. Production Testing

As production continues, the methods for testing used during prototype development will continue to be used, testing iterations of every new feature being developed and implemented into the system.

An example of some simple tests is shown below:

|  |  |
| --- | --- |
| Test | Expected Result |
| The user clicks start simulation | The simulation begins to run using parameters displayed |
| The user runs the simulation | The simulation continues running, and the outputs are updated live |
| The user changes the time scale | The simulation runs at the chosen speed |
| The user changes a parameter | The simulation results change |

### Unity Test Framework (20)

The Unity Test Framework is a Unity package which provides a standard test framework and allows you to test code both in edit mode and play mode. It uses an integration of NUnit library, which is an open source library for unit testing. This could be used be used for testing as development proceeds and as features become more complex.

## 5.4. Project Plan

## 

In this section the project plan is presented below, which includes all the key stages in this project as well as an approximate duration for each stage.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **September** | | | | **October** | | | | **November** | | | | **December** | | | | **January** | | | | **February** | | | | **March** | | | | **April** | | | | **May** | | | |
| **Initial**  **Proposal** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Requirements**  **Gathering** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Initial**  **Design** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Vertical**  **Prototype** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Interim**  **Report** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Interim**  **Demonstration** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Final**  **Design** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Implementation**  **Process** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Software**  **Testing** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Project**  **Evaluation** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Dissertation**  **Document** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Final**  **Demonstration** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## 5.5. Conclusion

This chapter presented the direction in which the project is heading. It focused on two aspects of the future work, first looking at the development process, and presented what has already been done so far, and the plans going forward. The second aspect that the project presented was the testing aspect of the project, again considering what has been done so far in terms of designing tests, and what will be done subsequently. Finally, a project plan is presented, and discussed.

## Bibliography

1.

Fort J. Exploring new ways to simulate the coronavirus spreadUnity Simulation을 이용한 코로나바이러스 확산 시뮬레이션コロナウイルスの拡散をシミュレーションする新しい方法を探る - Unity Technologies Blog [Internet]. Unity Technologies Blog. 2020 [cited 2020 Dec 13]. Available from: <https://blogs.unity3d.com/2020/05/08/exploring-new-ways-to-simulate-the-coronavirus-spread/>

2.

Evershed N, Ball A. See how coronavirus can spread through a population, and how countries flatten the curve [Internet]. the Guardian. The Guardian; 2018 [cited 2020 Dec 14]. Available from: <https://www.theguardian.com/world/datablog/ng-interactive/2020/apr/22/see-how-coronavirus-can-spread-through-a-population-and-how-countries-flatten-the-curve>

3.

About — Godot Engine (stable) documentation in English [Internet]. Godotengine.org. 2020 [cited 2020 Dec 14]. Available from: <https://docs.godotengine.org/en/stable/about/index.html>

4.

Unity Essentials - Unity Learn [Internet]. Unity Learn. Unity Learn; 2020 [cited 2020 Dec 15]. Available from: <https://learn.unity.com/pathway/unity-essentials>

5.

Unity Technologies. Unity - Manual:  Render pipelines [Internet]. Unity3d.com. 2019 [cited 2020 Dec 15]. Available from: <https://docs.unity3d.com/Manual/render-pipelines.html>

6.

Unity Technologies. Unity - Manual: Building a NavMesh [Internet]. Unity3d.com. 2019 [cited 2020 Dec 15]. Available from: <https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html>

7.

Unity Technologies. Unity - Scripting API: ParticleSystem [Internet]. Unity3d.com. 2019 [cited 2020 Dec 15]. Available from: <https://docs.unity3d.com/ScriptReference/ParticleSystem.html>

8.

Entity Component System | Entities | 0.16.0-preview.21 [Internet]. Unity3d.com. 2020 [cited 2020 Dec 15]. Available from: <https://docs.unity3d.com/Packages/com.unity.entities@0.16/manual/index.html>

9.

BillWagner. A Tour of C# - C# Guide [Internet]. Microsoft.com. 2020 [cited 2020 Dec 15]. Available from: <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>

10.

Unity Technologies. Unity - Manual:  C# Job System Overview [Internet]. Unity3d.com. 2018 [cited 2020 Dec 15]. Available from: <https://docs.unity3d.com/2018.4/Documentation/Manual/JobSystemOverview.html>

11.

Tuite AR, Fisman DN, Greer AL. Mathematical modelling of COVID-19 transmission and mitigation strategies in the population of Ontario, Canada. Canadian Medical Association Journal [Internet]. 2020 Apr 8 [cited 2020 Dec 15];192(19):E497–505. Available from: <https://www.cmaj.ca/content/192/19/E497.full>

12.

Bourouiba L. A Sneeze. New England Journal of Medicine [Internet]. 2016 Aug 25 [cited 2020 Dec 15];375(8):e15. Available from: <https://www.nejm.org/doi/full/10.1056/NEJMicm1501197>

13.

Horne M. Coronavirus in Scotland: Vulnerable will receive vitamin D supplements [Internet]. Thetimes.co.uk. The Times; 2020 [cited 2020 Dec 15]. Available from: <https://www.thetimes.co.uk/article/coronavirus-in-scotland-vulnerable-will-receive-vitamin-d-supplements-zc8stdpkh>

14.

Nishiura H, Oshitani H, Kobayashi T, Saito T, Sunagawa T, Matsui T, et al. Closed environments facilitate secondary transmission of coronavirus disease 2019 (COVID-19). 2020 Mar 3 [cited 2020 Dec 15]; Available from: <https://www.medrxiv.org/content/10.1101/2020.02.28.20029272v2>

15.

Few S. Data Visualization - Past, Present, and Future [Internet]. 2007. Available from: <https://www.perceptualedge.com/articles/Whitepapers/Data_Visualization.pdf>

16.

Kiryakova G, Angelova N, Yordanova L. Gamification in education. Proceedings of 9th International Balkan Education and Science Conference.

17.

Snaps Prototype | Office [Internet]. @UnityAssetStore. Unity Asset Store; 2019 [cited 2020 Dec 15]. Available from: <https://assetstore.unity.com/packages/3d/environments/snaps-prototype-office-137490>

18.

Principles behind the Agile Manifesto [Internet]. Agilemanifesto.org. 2020 [cited 2020 Dec 16]. Available from: <http://agilemanifesto.org/principles.html>

19.

Atlassian. What is Scrum? [Internet]. Atlassian. 2018 [cited 2020 Dec 16]. Available from: <https://www.atlassian.com/agile/scrum>

20.

About Unity Test Framework | Test Framework | 1.1.20 [Internet]. Unity3d.com. 2019 [cited 2020 Dec 16]. Available from: <https://docs.unity3d.com/Packages/com.unity.test-framework@1.1/manual/index.html>

x.

Unity 2017 Game AI programming - Third Edition | Packt [Internet]. Packt. 2017 [cited 2021 Apr 4]. Available from: <https://www.packtpub.com/product/unity-2017-game-ai-programming-third-edition/9781788477901>