DEVELOPMENT OF A GAME ENGINE-BASED TOOL FOR TRAFFIC MICROSIMULATION USING MOBILITY BEHAVIOURS

by

Sri Krishna Chaitanya Avala

Submitted in partial fulfilment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
April 2021

Dedicated to

My Dad A. Pandu Ranga Rao

My Mom A. Bala Tripura Sundari

My Brother A. Siva Abhishek

&

My Dog Scooby

Table of Contents

List of Tables

List of Figures

Abstract

Gaming tools gained significant interest among transportation modellers for mobility analysis, visualization and scenario testing. Commercial simulation software programs were used for running traffic microsimulations for years. They serve as traditional tools for both macroscopic and microscopic analyses. However, they lack flexibility when testing new methods or scenarios. This thesis presents a framework to create an open-ended 3D traffic microsimulation tool. It is achieved by incorporating a virtual 3D environment and integrating mobility behaviour models of vehicles and pedestrians into a game engine, thus creating a robust simulation tool. The tool is tested by simulating pedestrian scenarios before and after pandemic situations to analyze the percentage of violations with the help of a social force model. The tool's flexibility is then tested by using it to evaluate performance differences in various kinds of traffic consisting of autonomous and human-driven vehicles.

List of Abbreviations and Symbols Used

AV	Autonomous vehicle
HV	Human driven vehicle
ATS	Artificial transportation system
BAU	Business-as-usual
CMR	COVID-19 scenario with social distancing and mobility restrictions
d	Distance between pedestrians
A	Parameter for repulsion forces
n	Noise
n’	Number of pedestrians to react
λ	Anisotropy of forces
fv	Deceleration force
f_θ	Directional force
θ	Sign of angle
a	Acceleration
u	Current speed of vehicle
t_m	Headway time in seconds
d_s	Safety distance
ax	Standstill distance
bx	Time adjustment requirement value
ad	Following variation

maxspeed	Desired vehicle speed
msv	Maximum speed allowed
Sov	Speed dependency of oscillation
refval	Vehicle's distance from preceding vehicle
Val	Maximum value for speed variation
s	Saturation flow rate
h_s	Saturation headway
l	Last queued vehicle position
h_j	Headway of j^{th} queued vehicle
n_q	Position of queued vehicle

Acknowledgments

I would first like to send my deepest gratitude to my supervisor, Dr. Ahsan Habib, for taking me on and providing support, especially during the COVID-19 pandemic. I want to thank him for his patience, wisdom, and guidance he provided for this thesis.

I want to acknowledge the help I received from my lab mates to complete this thesis during the pandemic and thank them for their guidance and the knowledge they shared with me during my study.

I would also like to send my love and thanks to my family and friends who provided emotional support and company while writing this thesis.

Finally, I would like to recognize the efforts of frontline workers who sacrificed and helped millions of people in need during the difficult times caused due to COVID-19.

Chapter 1: Introduction

Since the advent of video games in the early 1950s, they have evolved into a cultural phenomenon (History.com, 2019). Today, nearly two-thirds of American homes have household members who frequently play video games, making up a 100-billion-dollar global industry.

Virtual environments/games are also being used for scientific and analytical purposes. There are many studies showing games as a learning tool to help people get educated in various disciplines. Such games are referred to as serious games, and their primary purpose is to create an impact on the audience beyond entertainment (Francesco Bellotti et al., 2013). Research shows that serious games could be helpful in cognitive training applications (Greitzer et al., 2007). Such platforms can also train adolescents for adherence to treatments for various diseases (Kato et al., 2008). It is also important to point out how games can improve learning efficiency by motivating players to master the game compared to conventional classrooms (Gee, 2003).

Games/virtual environments can be developed using a game engine; a set of software tools that helps in optimizing or creating a video game (Gamescrye, 2016). Game engines contain unique physics engines to help simulate physics in virtual environments. To facilitate the proper functioning of the virtual environments, game engines use various scripts designed to establish the mechanics of the virtual environment. This allows game engines to be used flexibly in scientific research to visualize a user's research or project

(Lewis & Jacobson, 2002). Game engines enable us to create virtual environments, buildings, streets, and cities. They are highly versatile and could be used to integrate with other applications such as CAD (Shiratuddin & Thabet, 2002). This allows researchers to create scripted tasks or even better, let users directly interact with the virtual environments to observe and evaluate various tasks and performances.

Another utilization of these platforms is in the field of planning and design. As we can create virtual environments, users can create unique models to put their ideas to the test. Urban planning, evacuation planning, and construction management are some of such examples which benefit from developing virtual environments and analytical platforms (Jorge et al., 2007; Li et al., 2015; Lopes & Lindström, 2012; Mól et al., 2008).

A study by Kourounioti et al. (2018) used simulation gaming to understand the stakeholders' behaviour and decision making when asked to tackle a freight transport innovation called Synchromodality. The authors developed five games to help the stakeholders familiarize themselves with the problems and opportunities of synchromodal transport. The results stated that simulation games involving both digital and board games proved helpful in educating the stakeholders about the innovation and helped them gain a positive attitude towards synchromodality.

This thesis develops a framework in which an open-ended 3D traffic microsimulation tool is created by incorporating a 3D model built in a 3D modelling computer program and integrating mobility behaviour models for vehicles & pedestrians

into a game engine, creating a robust virtual environment for analysis. Using a game engine allows a new way of analyzing traffic operations without the need for bridging multiple software to evaluate new hypothetical scenarios. This benefits transportation researchers and also planners by utilizing this tool to create and present immersive environments for planning purposes. The research extends with further usage of this tool by testing human and vehicular traffic operations. The tool contributes to the analysis of COVID-19 scenarios by testing mobility restrictions on pedestrians with the help of a social force model. In a later chapter, the tool is used to test the benefits and characteristics of autonomous vehicles in conjunction with human-driven vehicles in roadways.

1.1 Research Objective

This research aims to develop a framework for using a game engine in the fields of traffic engineering and transportation planning. The secondary goal of this study is to build a 3D microsimulation tool that can be later expanded/modified to encompass the city of Halifax. The technical objectives of the thesis include:

1. Develop a 3D microsimulation tool that incorporates realistic city infrastructure with vehicle and pedestrian behaviours in the network.
2. To assess the implications pedestrian walking behaviours reflecting social distancing measures due to COVID-19 pandemic.
3. To analyze the performance of autonomous and human-driven vehicles on the network within the game engine for traffic microsimulation.

1.2 Outline of Thesis

This thesis comprises of six chapters. The second chapter reviews the existing literature on gaming and visualizations used in the fields of planning and transportation. The third chapter details the framework of the 3D microsimulation tool. Chapter 4 shows the application of the developed game engine tool to analyze the impact of social distancing regulations for pedestrians on social distancing violations with the help of a social force model. Chapter 5 showcases the application of the developed tool in evaluating performance of autonomous and human-driven vehicles in varying traffic conditions. Chapter 6 concludes the thesis by summarizing the analyses and discussing the flexibility of using game engines for transportation engineering and planning.

Chapter 2: Literature Review

In this chapter, existing literature on the use of visualizations and game engines in planning and transportation is discussed. The flexibility of using game engines can be pronounced in numerous aspects. One such aspect is the scale of the project. A study used a game engine to create an interactive planning system by incorporating a virtual environment with a problem-solving mechanism (Calderon and Cavazza, 2001). The created application allows a user to solve a constraint-driven task which is to allocate a machine in a bank hall accommodated with spatial/resource limitations. The authors state that game engines are suitable to be used as an interface for planning systems as the tasks are to be completed in a 3-dimensional environment.

In urban design, people use computer-aided design applications to design infrastructure. These designs serve as an initial proposal for future projects. Some authors improved the interaction capability with such software by creating a system called ArchSplit, which allows the automatic generation of exploded views for any architectural design (Mike Houston, Chris Niederauer, and Maneesh Agrawala, 2004). To make the visuals more appealing and immersive, people have started using game engines to create virtual environments. One such research used a game engine to evaluate the usefulness of a game engine in urban design (Indraprastha and Shinozaki, 2009). The author tested the compatibility of Unity3D (a game engine) with software related to architecture such as ArcGIS and CAD and found that Unity3D was capable of simulating scenarios involving such software.

Similar virtual environments were created and tested to see the cognitive differences in spatial knowledge between two groups: one with a map view/bird's eye view and the other with a navigational computer simulation (Michael Tlauka et al., 1996). Mean pointing errors were used as data for the research. The results indicated that similar kinds of spatial knowledge existed in both groups. One study designed web games for teaching transportation engineering to students (Wang and Abbas, 2018). Five games were developed to educate students about the key concepts in the course. Games included traffic planning, signal control, traffic safety, highway design, and pavement design. For example, the game used for highway design had control variables like curve design and budget, which the students need to get familiarized. Results were achieved by comparing before and after quizzes scores which showed an improvement in scores after playing the games.

Virtual environments were also used for testing different scenarios. Rossetti et al. (2013) suggest a conceptual framework for using serious games in artificial transportation systems (ATS). The authors achieve this by incorporating behaviour elicitation with peer-designed agents, which would allow users to project their behaviour into the system. The authors propose tools related to pedestrians where walking patterns and interactions can be obtained for planners and vehicle driving where computer interactions and safety aspects can be examined. Some studies tested how virtual environments could be used for evacuation planning. One study tested the capability of a virtual environment to simulate a real-world evacuation experience (Jorge et al., 2007). The authors created a 3D model of a building considering its physical dimensions and

timed the participant's exit from the building. The times were later compared to a real-world evacuation, and it was found that the game engine was able to simulate the real-world environment adequately. Another study used Unity3D to create an evacuation scenario where participants were asked to play the simulation in which they had to evacuate an island due to a tsunami (Doirado et al., 2012). After all the participants have played the simulation, they were asked to give feedback on their level of engagement with the simulation. The authors found that they met their design goals, but there was still room for improvement. A similar study used a serious game as an evacuation simulator to improve fire drills and examine evacuation dynamics (Ribeiro et al., 2009). The study used 3D models built in Blender and imported into Unity 3D to build a virtual environment for participants to play the evacuation scenario. The research found that most of the participants missed the emergency exit and followed a longer path to exit the building.

Many pedestrian simulators have been built in the past to study pedestrian behaviours during road interactions. A study was conducted in Versailles, France, to study the short-term and long-term effects of simulator-based training on older pedestrians (Dommes and Cavallo, 2009). This helped them carry out gap-acceptance studies and test navigational aids for older people. Two groups of older pedestrians were used in which one group was trained with the help of a crossing simulation, and the other was trained with the help of the internet. Both groups were tested before, immediately and six months after the training. Results indicated a significant improvement in the overall safety of the older pedestrians while street crossing due to the simulation-based

training. A similar study was conducted in which the authors analyzed the impact of simulation-based training on older pedestrians (Dommes and Cavallo, 2011). The authors observed that behavioural improvements noticed in older pedestrians could be attributed to two reasons. The first reason is that the participants in the simulation-based training group took advantage of the feedback and education given to them during training. The other reason was that these people had undergone the repetition of crossing trials.

Moloney and Harvey (2004) developed a collaborative virtual environment to serve as an educational tool in which people could collaborate in real-time with the help of a server that lets them draw and speak. Virtual environments are also being used in real projects. One such project is the Personal Rapid Transit system in the city of Uppsala, Sweden (Lopes and Lindström, 2012). It involved developing a virtual reality simulation of parts of the city that would later turn out to be a critical factor in increasing the engagement with the project.

Microsimulations can also be simulated in game engines. One study has integrated traffic microsimulation and procedural 3D modelling with a game engine to showcase the potential use of virtual environments and virtual reality (Erath et al., 2017). The developed framework uses an exported file from Vissim (a traffic microsimulation software) to render the visuals using a game engine. This means that the flow of data is one-directional. Their work aims to establish a loop that would exchange data back and forth conveniently. However, this could pose many complications as researchers would be forced to rely on both parties (Vissim and the game engine) to support their project.

Miao et al. (2011) introduced a game-engine-based computing platform for ATSs. The authors used Delta3D for the game engine and created a virtual 3D environment. Pedestrians and vehicles were generated using data of the Chinese population at both macroscopic and microscopic levels. At the macroscopic level, the population travelling direction flows are defined, and at the microscopic level, the population's behaviour attributes like walking and relaxing were defined. The simulation platform was tested by analyzing the vehicular traffic average speed and noted that the average speed of vehicles dropped with the increase in population. This study serves as a starting point for the usage of game engines in traffic simulations but lacks the characterization of pedestrian and vehicular movement behaviour.

This thesis provides a framework that acts as a solution to show how a game engine can be used to perform traffic microsimulations with the help of mobility behaviour models for pedestrians and vehicles. A framework is proposed to build a virtual 3D environment that can simulate new situations of traffic microsimulations within the game engine with the help of object-oriented programming. Such framework allows the ability to modify and analyze theoretical transportation scenarios and at the same time allows users to visualize their ideas. This thesis advocates a game engine-based platform as a decision support tool that can be used flexibly to visualize and analyze scenarios for transportation and planning purposes.

Chapter 3: Conceptual Framework

3.1 Study Area

This research is focused on Spring Garden Road in Halifax, Canada. It is located in the heart of the city's downtown area, known for its eateries and shopping stores. Therefore, the location attracts many people and therefore contributes to more pedestrian and vehicular traffic and therefore is considered appropriate for this research.

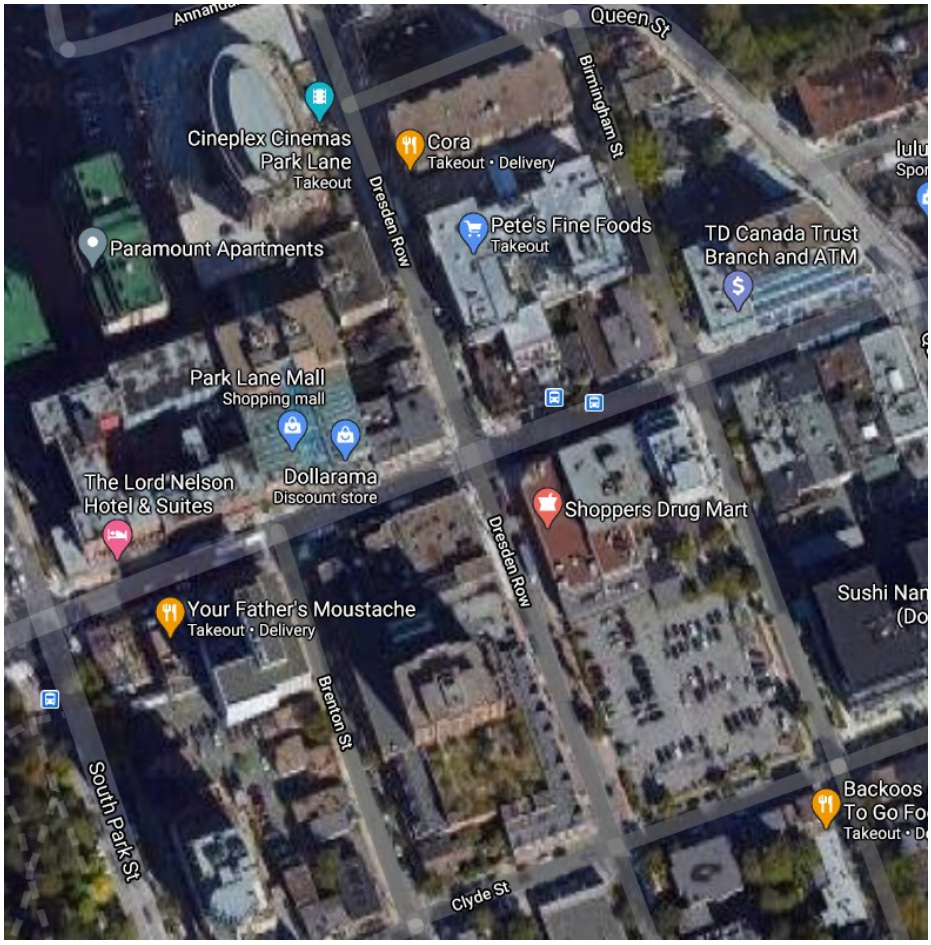

Figure 0-1 Aerial View of Spring Garden Rd

3.2 Methodology

The methodology of this thesis is based upon a systematic approach to developing a dynamic virtual environment. First, the 3D model of the street is created in a modelling program which is then imported to a game engine. Later the dynamics of the environment are designed with the help of coding to achieve the required functionality. The dynamics include pedestrian and vehicle behaviours, pedestrian and vehicle movements, and traffic signal systems. The framework of implementation is represented in the flow chart below.

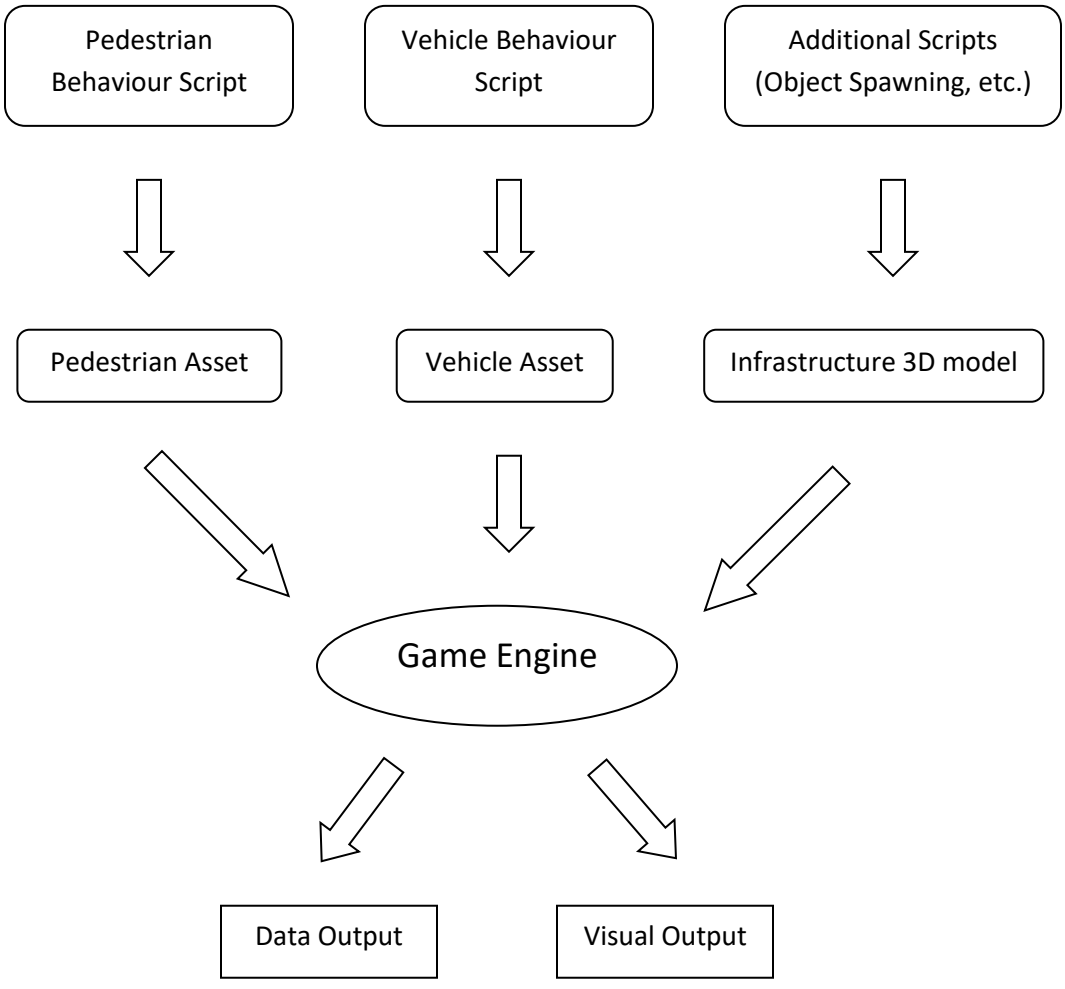

Figure 0-2 Framework of the Models Developed in this Study

To begin integrating the 3D virtual environment in Unity, a 3D model needs to be created first. For this task, Sketchup (3D modelling computer program) was chosen as it is a free software which facilitates the creation of a highly detailed environment. It also outputs compatible files that can be used in a game engine. The 3D street network was developed based map data retrieved from CADmapper (an application used for accessing street map data). After retrieving the map data, the 3D model was designed in a virtual space using height, length, and width measurements, and Google street view for colours and textures. The same 3D model could be created in a game engine, but it would not be as flexible as Sketchup in creating a static environment (Habib and Holmes, 2019).

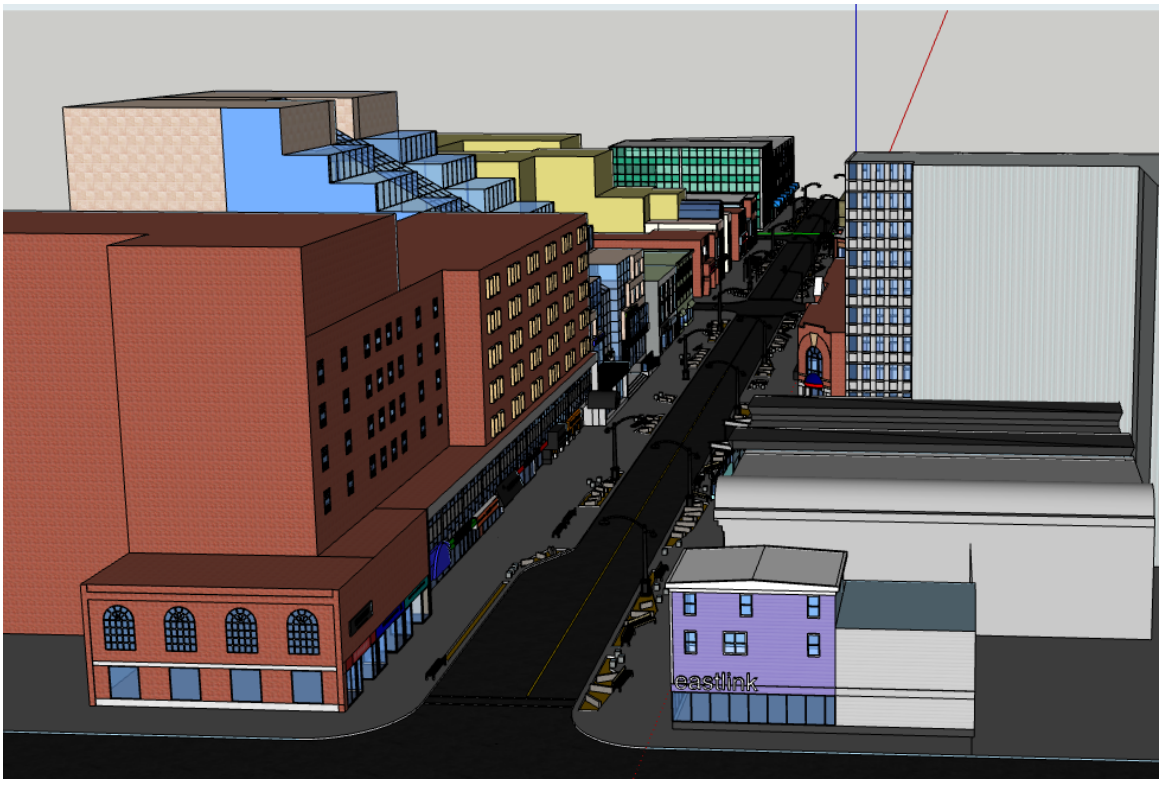

Figure 0-3 Sketchup Model used in the Study

The created model was then placed in Unity, a cross-platform game engine used for developing real-time 3D content. A simple drag and drop into Unity imports the 3D model. Problems were faced while importing the 3D model, which resulted in a loss of model materials. However, most of the model was usable, and Unity lets users apply materials to the imported models. Figures 3-4 and 3-5 show the before and after visualization of the virtual 3D environment.

Figure 0-4 Initial Stage of the Imported 3D Model

Figure 0-5 Imported 3D model after Applying Materials and Textures

After importing the static model into Unity, the next step is to create dynamic pedestrian and vehicular traffic. This requires 3D models for vehicles and pedestrians and needs coding of scripts that enable these assets to move accordingly. Assets are any file or item that can be used in Unity. Examples of such assets are audio files, 3D models and basically any item that Unity supports. Finding 3D models for vehicles and pedestrians is easy. Unity has a built-in asset store where anyone can download packages and assets.

Figure 0-6 Car Asset in Unity Store

Most of the assets downloaded from the store do not have embedded animations that animates the objects according to their movement. Designing these animations also requires scripts. Unity supports many programming languages, but the most commonly used language is C#. Variables and properties like box colliders and mesh renderers must be added/modified according to our usage for each asset.

Every animation that is supposed to happen in the simulation must be scripted—for example, rotation and turning of vehicle wheels and walking movement of pedestrians. Apart from the moving models, static models like signal poles are also scripted according to the requirement. Other visuals like trees and roadblocks are also used to make the virtual environment look more appealing, making it feel closer to the real world.

GameObjects and their Properties

All objects or items present inside the simulation/game in Unity are called as *GameObjects*. Any *GameObject* imported into the scene can have various properties. These properties can be added or modified using the inspector window.

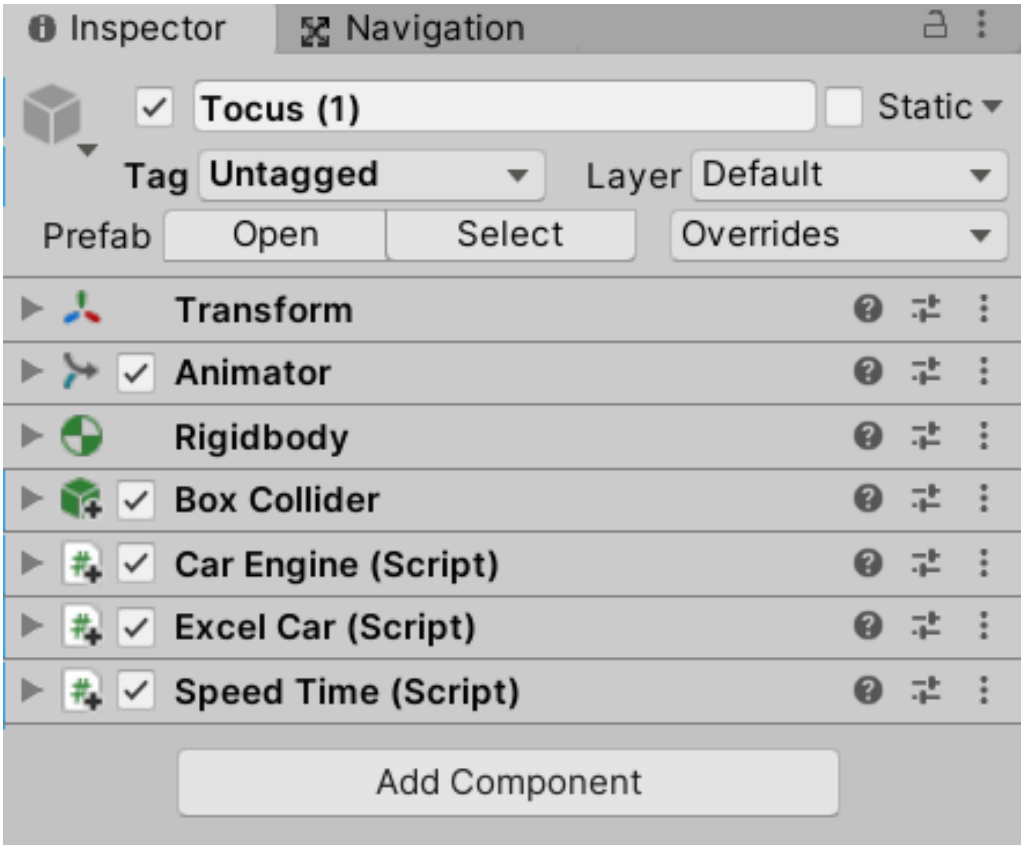

Figure 0-7 Inspector Window Showing the Properties of a GameObject

There are three essential components that are commonly used for each *GameObject in* this thesis:

Transform – Every *GameObject* has the *Transform* component attached to it. It is not possible to have a *GameObject* without this component. The *Transform* component is

used to store a *GameObject's* position, rotation, and the size of the object in all three axes.

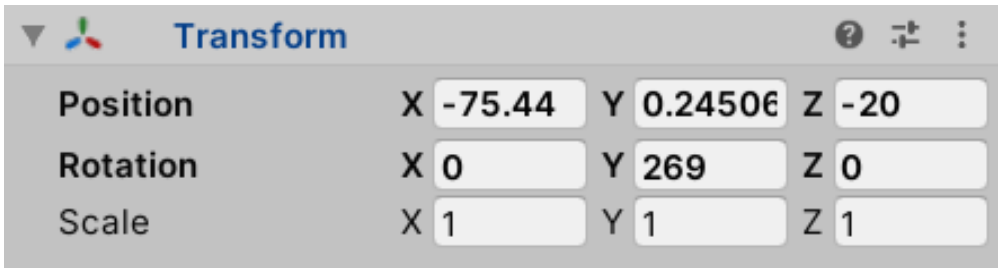

Figure 0-8 Transform Component of a GameObject

Rigidbody – The *Rigidbody* component can be attached to any *GameObject* which needs to be under the control of Unity's physics. This component lets us access properties like mass and, gravity which is essential for *GameObjects* like vehicles and pedestrians. For example, the mass of a car is defined as 1000 kilograms, and the mass of a pedestrian is defined as 60 kilograms. All pedestrians and vehicles also use gravity.

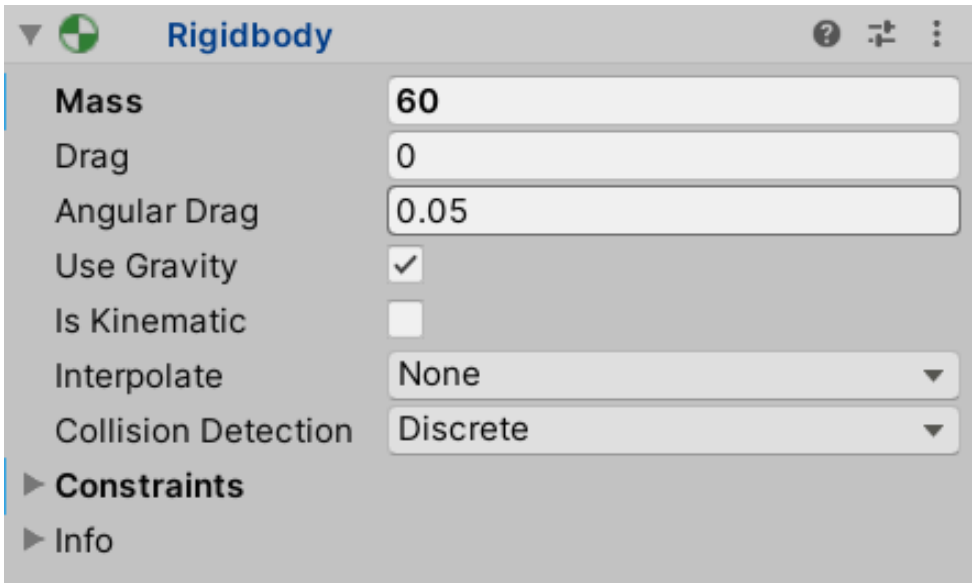

Figure 0-9 Rigidbody Component of a Pedestrian GameObject

Box Collider – The box collider component is used for detecting collisions. It is basically a cuboid that can be manipulated to define the perimeter of the *GameObjects.* One of the critical properties of this component is *isTrigger.* This property allows the execution of user-defined tasks when the *GameObject* comes in contact with another *GameObject.*

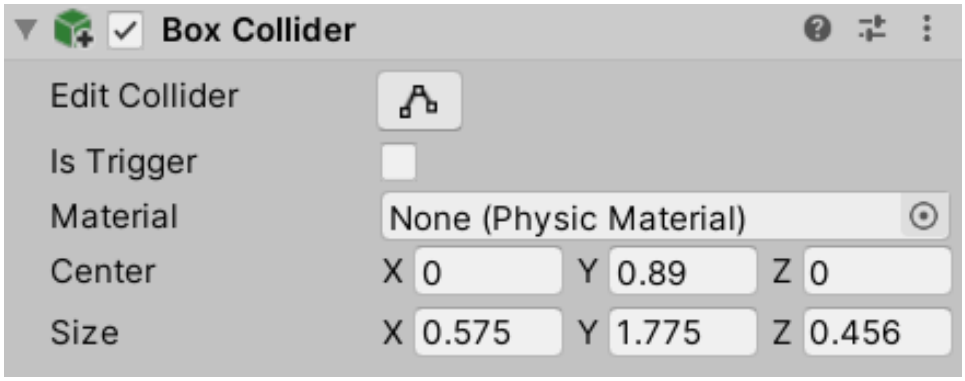

Figure 0-10 Box Collider Component of a GameObject

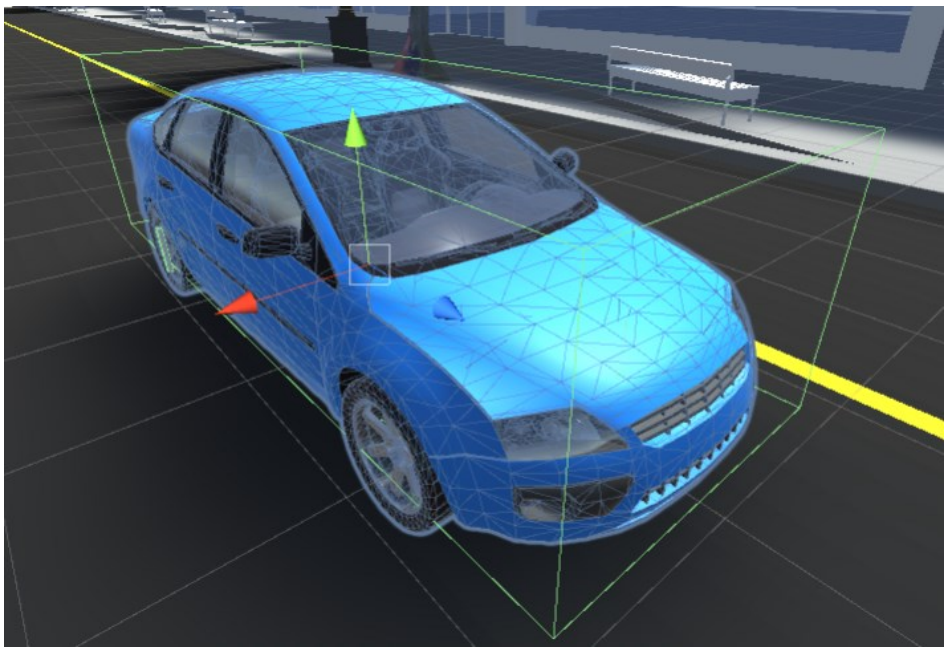

Figure 0-11 Box Collider of a Car

Apart from these three components, there are many more components that can be used to define a *GameObject.* One such component is called *Animator,* which helps with the animation of a *GameObject.* This is mainly used for all human *GameObjects* in the simulations. Scripts coded by users can also be added as a component to a *GameObject*, providing additional functionality.

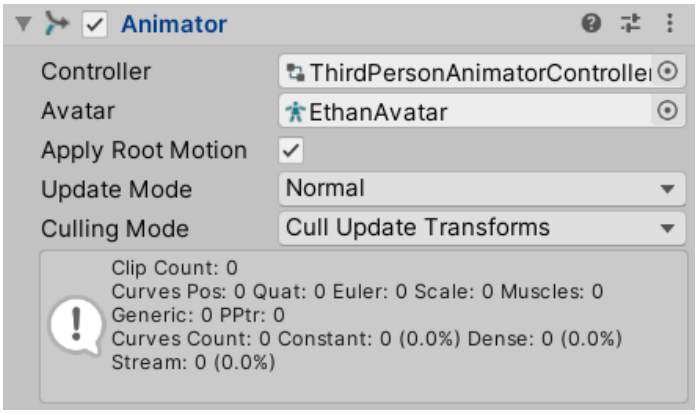

Figure 0-12 Animator Component of a Pedestrian GameObject

Pedestrian Representation

Pedestrians in this thesis are represented by a 3D model imported from the Unity asset store. The human asset is integrated with a prebuilt animator component that adds the animation of walking and standing still for pedestrians. Every pedestrian has an origin and a destination, which are predefined by the user with the help of scripts.

The spawning of pedestrians is facilitated by an empty *GameObject* embedded with a spawning script. This spawning *GameObject* can spawn multiple *GameObjects* at random times or by the time defined by the user. To add randomness and realism to the

simulation, some pedestrians are randomly assigned their destination at the beginning of their spawn.

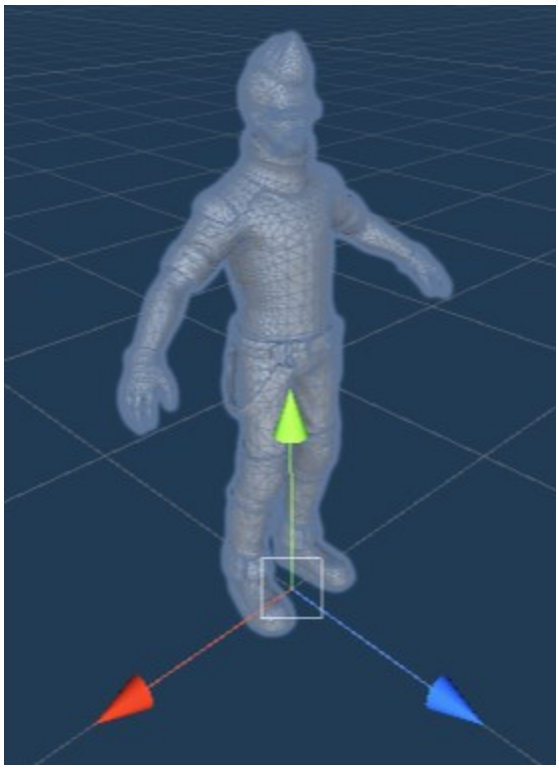

Figure 0-13 Pedestrian GameObject

An essential component of pedestrian *GameObjects* is *NavMeshAgent*. *NavMeshAgent* component lets a *GameObject* recognize another *GameObject* on the *NavMesh.* This helps with the avoidance of collisions while the pedestrians move towards their goal. *NavMesh* is a navigation mesh that defines the walkable/movable areas of the game/simulation. *NavMesh* lets users *bake* a user-defined area on the map, allowing *GameObjects* to avoid unwanted areas that are not defined for movement. Such unwanted areas include rooftops, chairs, or any other *GameObject* which is not designed as a walkable area. The *NavMesh* can be edited according to the user's preference. The

NavMesh baking properties can be accessed in the Navigation window. The user can define properties like *Agent*'s radius and height, maximum slope and step height an *Agent* can access.

Pedestrain *GameObjects* are also integrated with various scripts to add more behaviour. AiCharacterControl and ThirdPersonCharacter are two scripts that are embedded into the imported human 3D model. Apart from these, other scripts for data collection and social force model are added to the pedestrian *GameObject,* which is discussed in detail in Chapter 4.

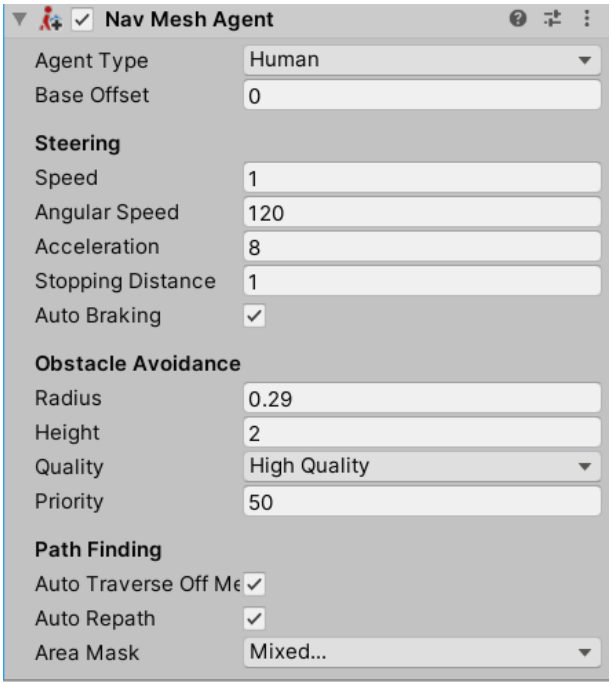

Figure 0-14 Example NavMesh Component Properties

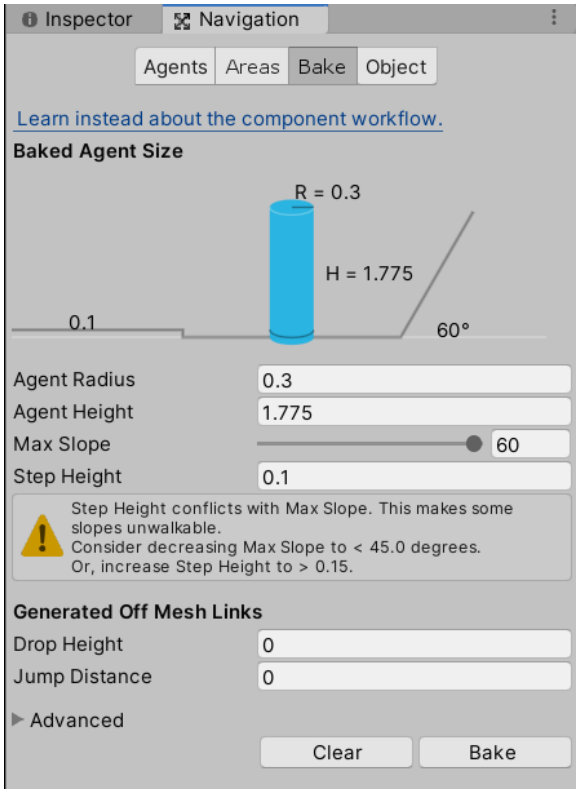

Figure 0-15 NavMesh Baking Settings Example

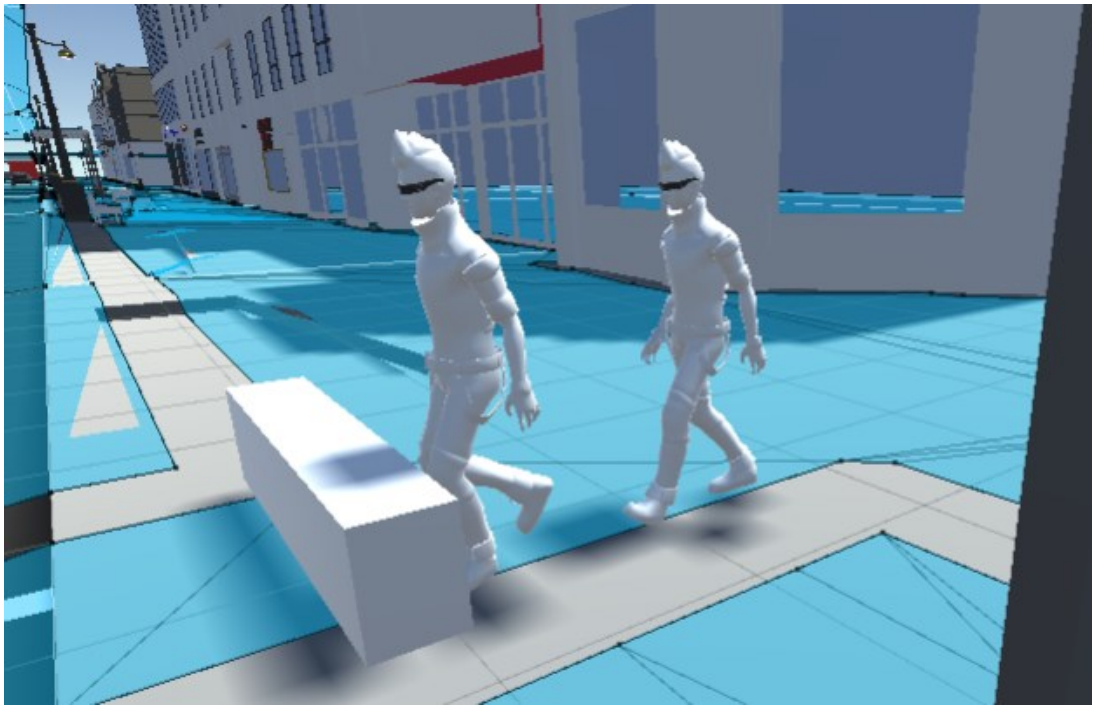

Figure 0-16 Baked NavMesh Indicated by the Highlighted Blue Area

Vehicle Representation

Vehicles are more complicated than human assets. After importing 3D models from the Unity store, properties of the wheels should be defined as the movement of a vehicle relies on its wheels. This is done using the *Wheel Collider* component. Defining the *Wheel Collider* properties is done by a trial-and-error method. Figure 3-17 shows the settings used for this thesis. Like the pedestrian *GameObjects,* vehicles also have their origin and destination points.

Vehicles were not integrated with *NavMesh,* unlike their pedestrian counterparts but were designed to rely on the physical properties of the environment. They are also required to start and stop according to the object in front of them. This is achieved using the *Raycast* system in Unity.

In Unity, the *Raycast* system throws a ray at a given distance in a given direction for a given length. If the ray hits a *collider* of a *GameObject*, it retrieves the information of the *GameObject* and can be scripted to do the required processes. For a vehicle, two sets of *Raycast* systems are used: one for detecting pedestrians and the other for detecting vehicles in front. When a pedestrian is detected, the vehicle is designed to come to a stop and let the pedestrians pass through. When a vehicle is detected, the asset has two options. If the vehicle in front is standing still, then it comes to a stop based on the standstill distance assigned to it. If a moving vehicle is detected, it tries to achieve the speed according to the defined circumstance.

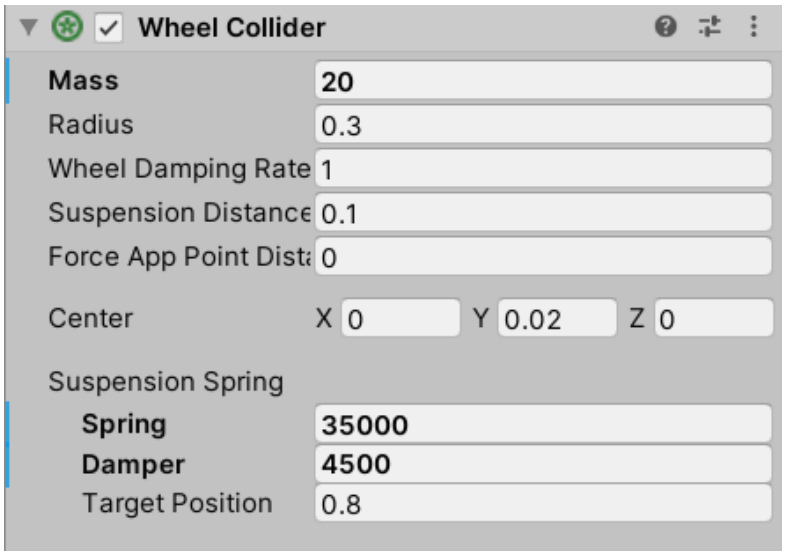

Figure 0-17 Wheel Collider Properties of a Wheel

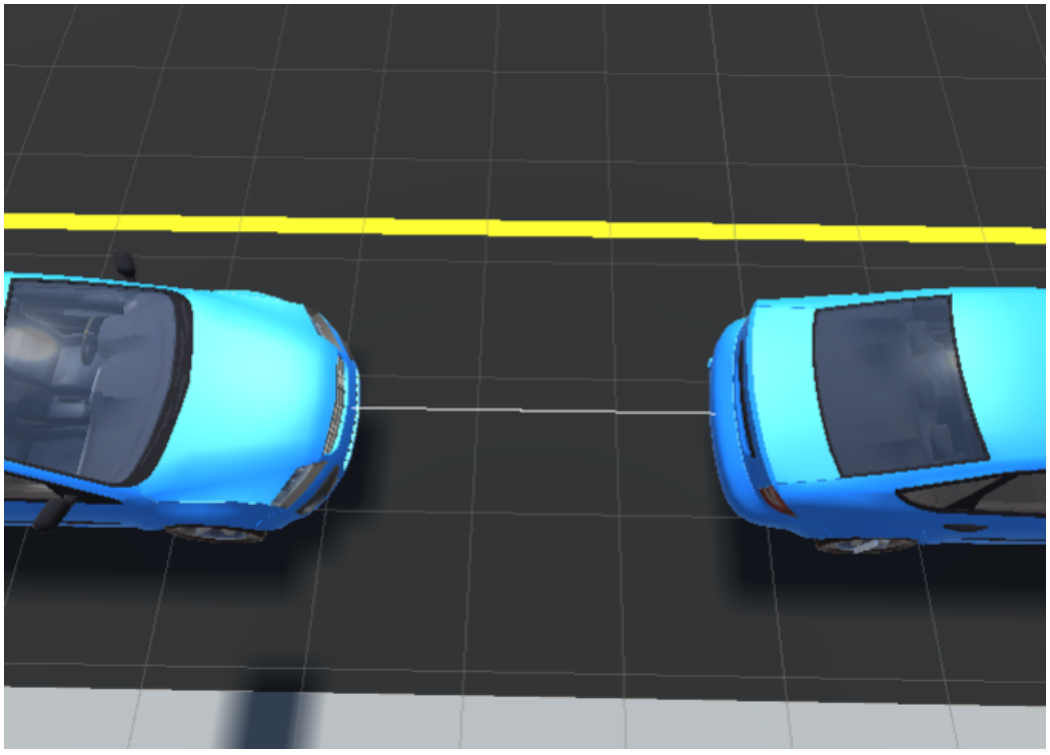

Figure 0-18 A Ray Thrown at the Vehicle in Front

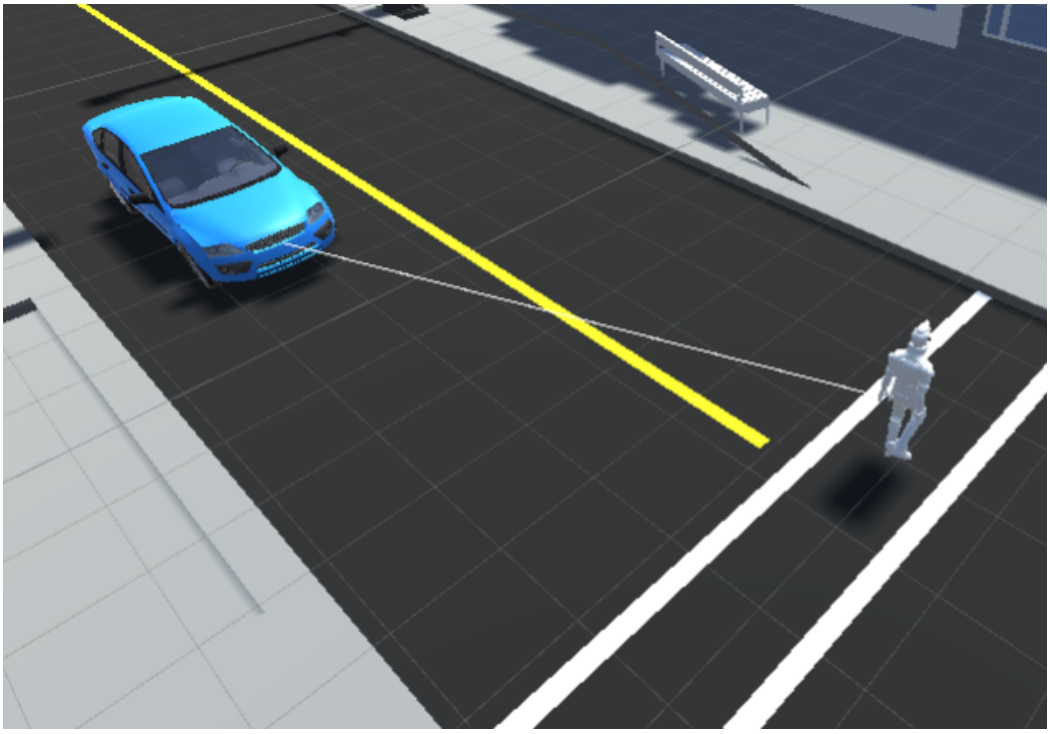

Figure 0-19 Vehicle Detecting Pedestrian with the help of Raycast System

The basic moving behaviour of a vehicle in this thesis is based on the desired safety distance of a vehicle which is calculated by the below formula (PTV, 2018). This formula is embedded into the *Raycast* system to detect the *GameObjects* in front:

$$d_s = ax + bx \tag{1}$$

Where,

d_s = safety distance;

ax = standstill distance;

bx = time adjusting requirement values; bx is calculated as follows:

$$bx = (bx_{add} + bx_{mult} * z) * \sqrt{v} \quad (2)$$

Where,

v is vehicle speed in m/s;

z is a value of range [0, 1], which is normally distributed around 0.5 with a standard deviation of 0.15.

The length of the ray thrown is equal to the safety distance calculated, which depends upon the speed of the vehicle and the given standstill distance. A single is thrown from the center of each vehicle to detect a vehicle in front. To detect pedestrians, rays are projected for each degree to a total coverage of thirty degrees on both left and right sides of the vehicles. A pictorial representation of this system is shown in Figure 3-20. Therefore, if the vehicle is moving at a high velocity, it will have a higher length of ray, and if it is moving slowly, it will have a smaller length of ray. This kind of setup is designed to discard unwanted detections. This model is used for detecting both pedestrians and vehicles; however, the system for vehicles can be altered to change the behaviour of the vehicles, which is discussed later in Chapter 5.

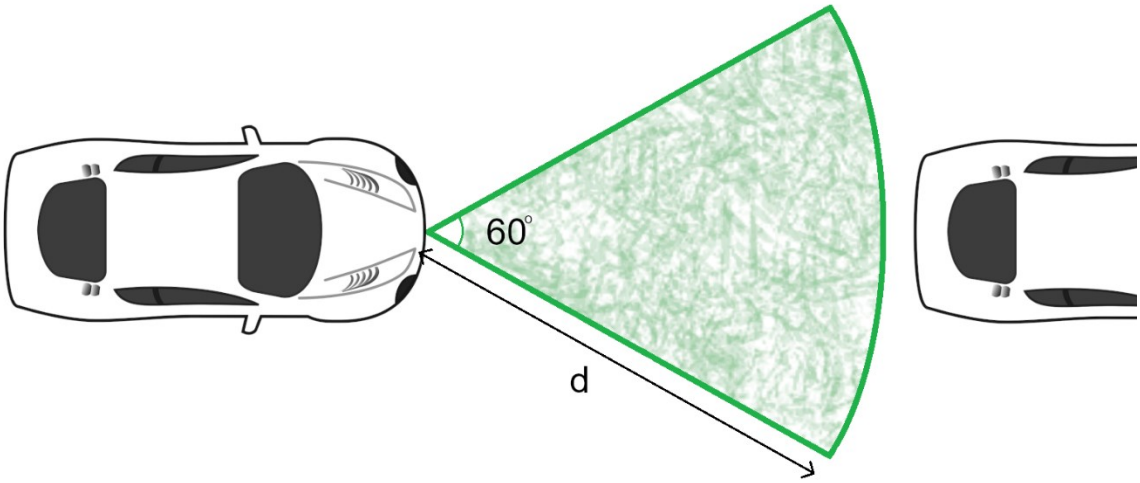

Figure 0-20 Pictorial Representation for Pedestrian Sensor System

Intersections

The virtual environment is modelled for two main intersections to observe different aspects of the traffic simulation model. The first intersection represents the intersection at Spring Garden Rd and Dresden Row and is represented as intersection 'A'. This virtual intersection is designed with traffic signals that lets vehicles and pedestrians cross appropriately. This is achieved using a script, which is embedded into the 3D model. The vehicles and pedestrians are therefore controlled by using different *GameObjects.* For example, a transparent *GameObject* was used to communicate with vehicles whether they must stop or proceed. This transparent *GameObject* is activated when the traffic signal is red or yellow and deactivated when the signal is green. When a vehicle comes in contact with this *GameObject,* the vehicle is asked to slow down and then start moving again

once this transparent *GameObject* is deactivated. This kind of mechanism helps collect data related to variables like saturation headway.

Pedestrian *GameObjects* are also controlled using other *GameObjects.* When the signal is green for the vehicles, pedestrians are blocked to let vehicles pass through the intersection and vice versa.

Figure 0-21 Signal Pole Showing Green Signal

Figure 3-22 represents the designed signal cycle for this thesis. The cycle can be modified according to necessity. Pedestrians and vehicles are designed to stop in coherence with the traffic signal cycle.

Figure 0-22 Reference Signal Cycle in Seconds

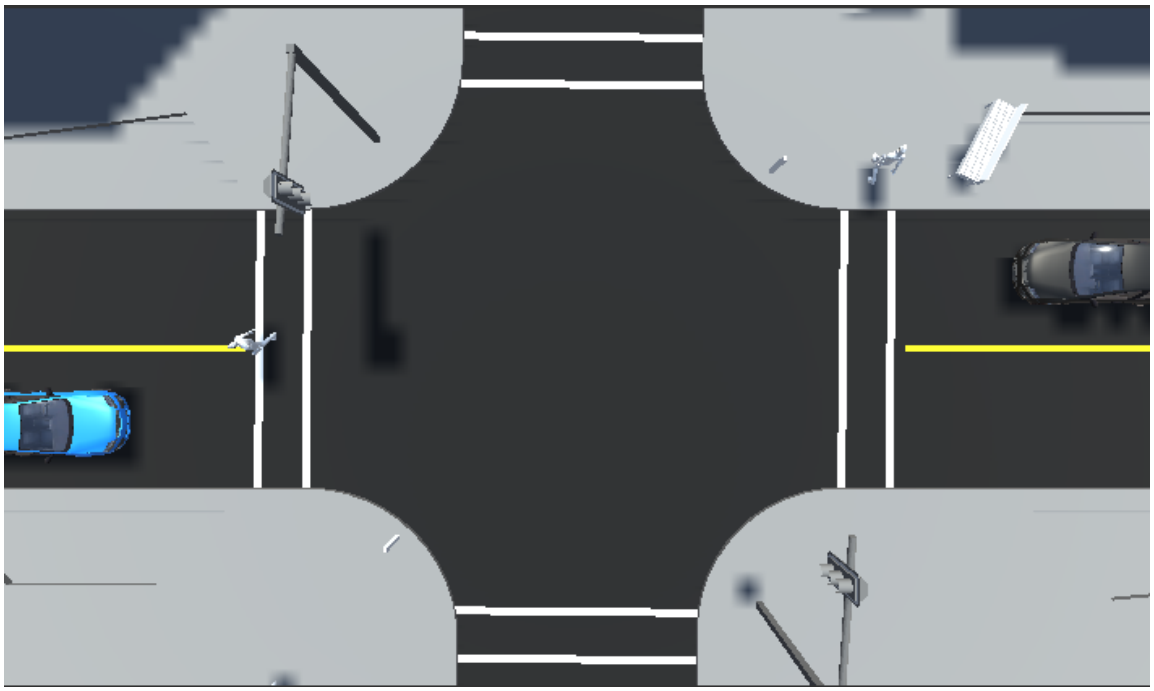

Figure 0-23 Intersection 'A' with Traffic Signals

Figure 0-24 Pedestrians Crossing the Road

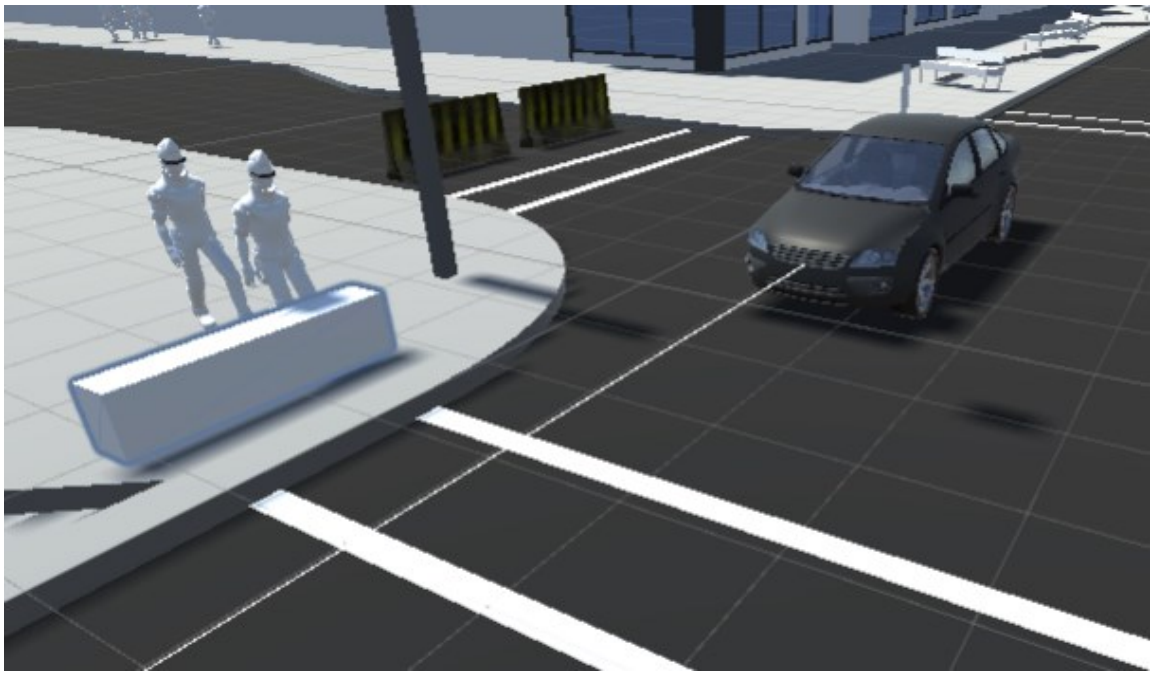

Figure 0-25 Pedestrians Blocked to Allow Vehicles to Pass

The second intersection is modelled after the intersection at Spring Garden Rd and Birmingham St and is represented as intersection ‘B’. The virtual model is designed without a traffic signal which allows the pedestrians to cross freely. This forces the vehicles to stop and move in relation to the presence of the pedestrians nearby. The stoppage of vehicles is built using the *Raycast* system available in Unity.

3.3 Discussion and Conclusion

The virtual environment was initially established by importing 3D models into Unity. This presented a static 3D virtual environment. Scripts were introduced to add movement to vehicle and pedestrian *GameObjects* making the environment dynamic. Additional 3D visual props like trees and roadblocks were also used to add realism to the virtual environment. The virtual environment was finally tested to see the simulation in action. The pilot simulation was let run for more than two hours to notice bugs and other errors.

The animation of pedestrian *GameObjects* was smooth. The animation of vehicles was also fluent, but the lack of character for each car made the simulation look robotic. This is rectified in Chapter 5, where more behavioural elements for vehicles are added. The intersection ‘A’ with signals was able to perform its task adequately by stopping and letting vehicles and pedestrians pass through appropriately. The intersections ‘B’ without the traffic signals was also observed to have no failures.

Vehicles were successfully able to recognize the *GameObjects* in front of them with the help of the *Raycast* system. They were also able to behave according to

the *GameObjects* in front. No vehicle was observed to collide with another vehicle or pedestrian *GameObjects*. The initial stage of building a dynamic virtual environment is therefore completed. This environment is further used in Chapters 4 and 5 to test the flexibility of a game engine to perform various simulations.

Chapter 4: Application of Game Engines in COVID-19 Simulations

4.1 Introduction

In December 2019, there was an outbreak of a deadly virus, which is now known as COVID-19 (Warren & Skillman, 2020). The virus soon started penetrating other countries across the globe. As of 22nd February 2021, the number of confirmed cases of COVID-19 infected people surpassed 111 million people globally, taking the lives of more than two million people (WHO APA, 2021).

To contain this virus, many countries across the world started implementing drastic measures in the hopes of decreasing the transmission rates (Ghader et al., 2020). Among those measures, social distancing measures are the most effective non-medical initiatives taken up by countries across the globe. Despite these measures, many people are choosing not to socially distance themselves from others. A Stanford-led study revealed that 39.8 percent of the total respondents stated that they were not complying with the social distancing measures (Stanford, 2020). Regardless of the various reasons prompted by the respondents, people are being encouraged to get fully vaccinated. However, with the constant evolution of COVID-19 variants, it is also possible that these vaccines might not work against certain variants (CDC, 2021).

Social distancing is considered an important measure to limit the spread of COVID-19 (WHO, 2021). One article established pedestrian density limits for urban and architectonic spaces with the help of laboratory experiments. (Echeverría-Huarte et al.,

2021). The impacts of variables like pedestrian density, pedestrian speed, and recommended safety distance on interpersonal distances between people in dense crowds were studied.

This research focuses on showing the application of the developed game engine tool in simulating and testing for pandemic scenarios. As game engines offer great flexibility in modifying or creating any scenario, it is used as a tool to analyze pedestrian traffic operations. With the help of a pedestrian social force model, pedestrians in this study replicate three different scenarios. The first scenario represents a standard day-to-day pedestrian walking behaviour, and the second scenario represents socially distanced pedestrian walking behaviour with mobility restrictions. The third scenario represents the reopening stage once the lockdowns are ended. The results for all three models is discussed, and conclusions were drawn to assess their impact on social distancing violations. Additional measures are proposed to tackle the issue of violations.

4.2 Literature Review

Literature on the interactions of COVID-19 and public spaces, specifically urban streets is limited. The following literature review discusses various studies about pandemic scenarios and social distancing.

Min W. Fong et al. (2020) conducted a systematic review on the effectiveness of various measures taken during pandemics. Isolating the ill, quarantining exposed persons, dismissing schools, contact tracing, workplace measures, and avoiding crowding were the

six measures taken into consideration. The evidence of the effectiveness was primarily taken from observational studies and simulation studies. The review concluded that these measures showed that such measures help reduce the transmission and alleviate the impact of influenza pandemics. Bouchnita & Jebrane (2020) used a hybrid multiscale model to assess the potential of non-pharmaceutical measures using COVID-19 transmission dynamics. It uses a social force model for the individuals in which everyone is either susceptible, infected, quarantined, immunized or deceased. The study reveals that pre-symptomatic transmission accelerates the inception of the exponential growth of COVID-19 cases. The research shows that restricting the movement of individuals flattens the epidemic curve of the number of cases.

From a slightly different perspective, a study shows the effects of social distancing on the economy, public health, and environment. The integrated epidemiological-economic model built by the authors (Newbold et al., 2020) suggests that deployment of physical distancing policy with optimum timing and intensity could save millions of lives along with generating significant net benefits in terms of the economy when compared to a scenario without the existence of a physical distancing policy. The authors also created a model that could estimate the number of deaths averted by the decrease in air pollution caused due to social distancing. The authors pointed that air pollution could cause respiratory diseases that could worsen the condition of people affected with COVID-19. Another research driven by economics analyses states that the optimal control during a pandemic could be achieved by one of the two extremities: to let the epidemic run its course or exercising a highly cautious control (Maharaj & Kleczkowski, 2012). The

authors showed that the worst possible economic outcome could emerge when control is attempted but not cautiously enough.

There is also research demonstrating the effect of social distancing indoors. A systematic review of existing literature suggests that following physical distancing in non-healthcare workplaces would produce a notable reduction in an influenza attack rate (Ahmed et al., 2018). Another indoor modelling study using a pedestrian-based epidemic spreading model found that restricting peoples travelling frequency inherently increases the time they spend in grocery stores (Xiao et al., 2020). The study involves a before-and-after travel demand analysis which suggests entry limitations to such stores that would decrease the risky contacts, meanwhile mentioning the likely increase of waiting queues outside stores. Another study focuses on the design of pedestrian queues to reduce infectious disease transmission (Derjany et al., 2020). With the help of a multiscale model which combines pedestrian dynamics with stochastic infection spread models, the authors show that using wall separators instead of rope separators helps in reducing the number of unwanted contacts and disease spread.

A study was conducted using a simulation-based estimation to predict the actual cases of COVID-19 in Iran, which were later compared to the official confirmed cases of COVID-19. The integration of data from various sources suggests that the actual numbers are at the least an order of magnitude higher than the official numbers (Ghaffarzadegan & Rahmandad, 2020). Another large-scale study involving the integration of aggregated smartphone location big data with geographic information systems (GIS) looked at how

people reacted in different states and counties of the United States to social distancing guidelines. The authors observed an excellent adherence to the guidelines when they were first announced and how the mobility patterns were affected following certain significant events, which then later was associated with increasing COVID-19 cases (Gao et al., 2020).

A recent study involving a microsimulation model which aimed to understand the impacts of social distancing on pedestrian environments shows the impacts by comparing a number of social distancing violation in a pandemic scenario which comprises of widening of sidewalks, revised walking behaviours, and reduced pedestrian traffic with a business-as-usual (pre-pandemic) scenario (Alam et al., 2020). The results showed a significant decrease in social distancing violations in the pandemic scenario against the business-as-usual case for Spring Garden Road.

This chapter shows how walking behaviours and social isolation impact the number of social distancing violations. It essentially showcases the application of a game engine in COVID-19/pandemic simulations. To do so, pedestrians are generated accordingly to replicate pre-pandemic, pandemic, and post-pandemic scenarios. The pedestrian movements are produced with the help of a social force model that helps reproduce scenario-based walking behaviours. Later, the results from different scenarios are compared to and additional measures be proposed.

4.3 Methodology

The 3D virtual game engine-based tool developed in Chapter 3 is used to test multiple scenarios of pedestrian simulation during COVID-19. Pedestrian movement is an essential aspect of social distancing. Therefore, a social force model is used to generate pedestrian movements. The simulation consists of fundamental pedestrian flow concepts where the human assets travel in opposite directions to reach their destinations.

This study consists of three scenarios. The first scenario depicts the motions of pedestrians before COVID-19, where pedestrians would not mind the gap between each other. The second scenario depicts the motions of pedestrians but with social distancing measures in place along with mobility restrictions representing lockdowns. The third scenario would have the same pedestrian behaviour as the second scenario, but the pedestrian traffic intensity would be the same as the first scenario.

To assess the performances of each scenario, the number of social distancing violations are compared. In this study, social distancing violations are defined as when the distance between two pedestrians is less than or equal to two meters. They are also visually represented during the simulation with green lines. Other data is also considered. The amount of time travelled by each pedestrian to reach their destination and the pedestrian density of the testing area are considered. Equation 3 is used to calculate pedestrian density. This kind of data is retrieved using various scripts. Figure 4-1 shows a screenshot of the simulation. For this thesis, pedestrian density is calculated by the below equation.

$$Pedestrian\ Density = \frac{Number\ of\ Pedestrians}{Area} \quad (3)$$

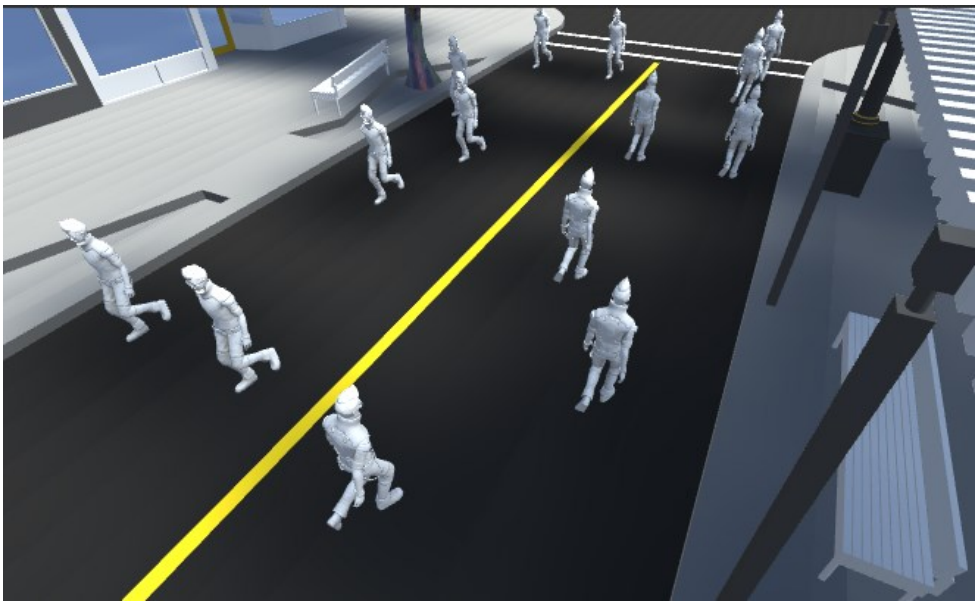

Figure 0-1 Business-As-Usual Scenario

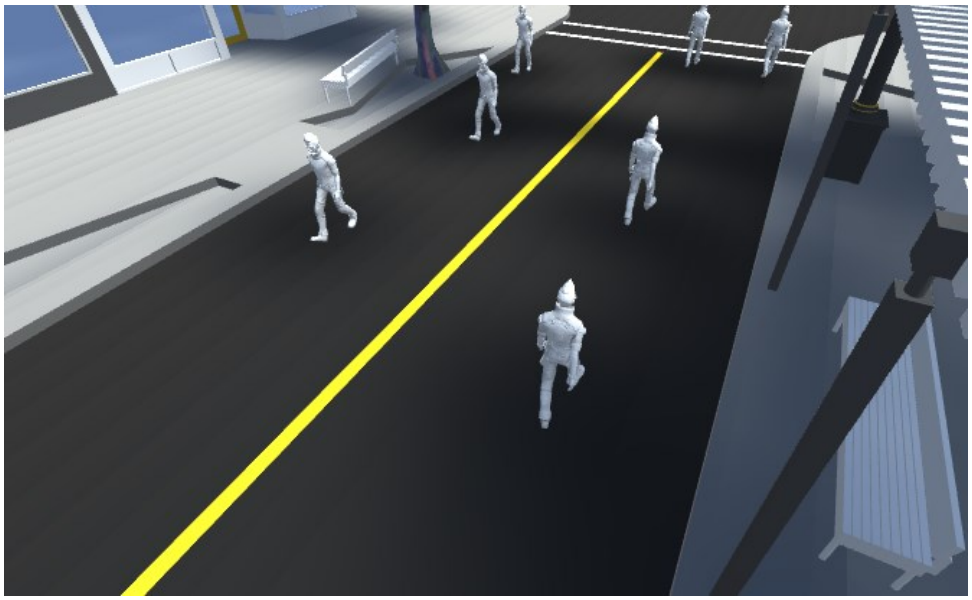

Figure 0-2 After Pandemic with fewer pedestrians

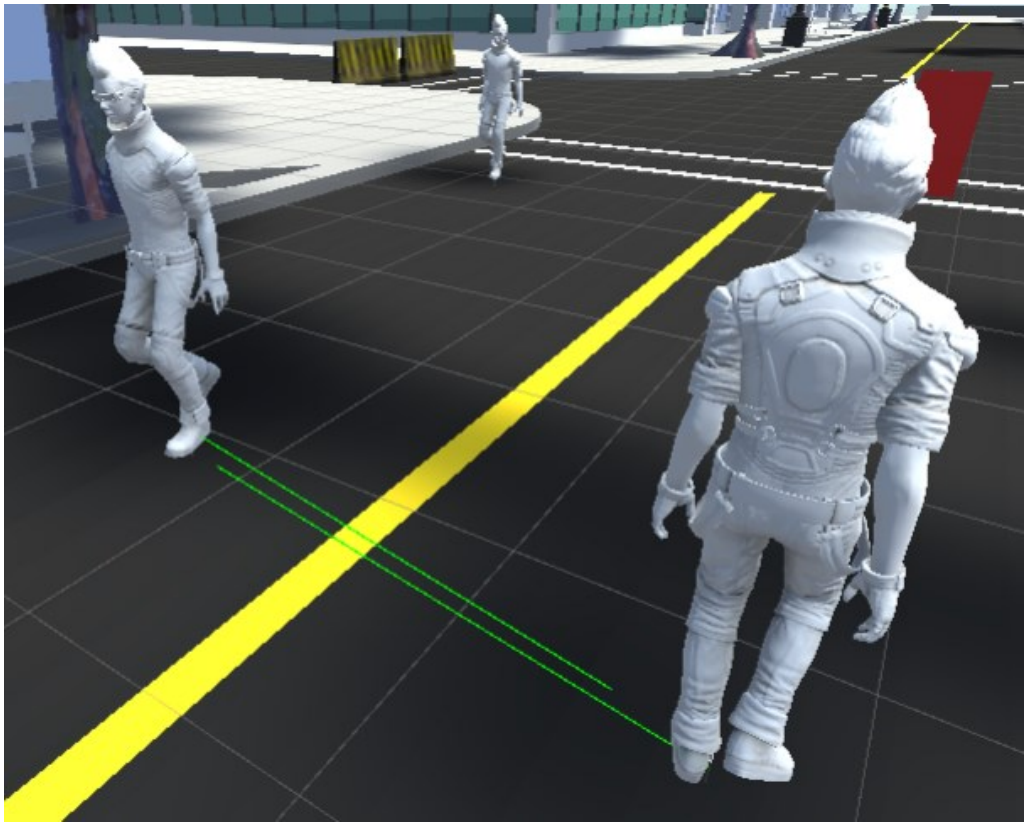

Figure 0-3 Social Distancing Violation Indicated by Green Lines

Social Force Model

To replicate the natural movement of pedestrians, Helbing introduced an empirical model called the social force model (Helbing and Molnár, 1995). This study uses the improvised version of Helbing's model put forth by Mehdi Moussaïd. The social force model can be represented as follows (Moussaïd et al., 2009):

$$f_v(d,\theta) = -Aexp(-\frac{d}{B} - (n'B\theta)^2) \tag{4}$$

$$f_\theta(d,\theta) = -AKexp(-\frac{d}{B} - (nB\theta)^2) \tag{5}$$

Where, f_θ and f_v are directional change and deceleration forces respectively; *A, B, n*, and *n'* are model parameters; $K = \theta/|\theta|$ is the sign of angle θ; d is the distance between the pedestrians.

A study to optimize these parameters found that increasing the values of *B* and decreasing the value of *A* would diminish the social force repulsion between pedestrians (Sticco et al., 2020). As this chapter focuses on distancing the pedestrians, multiple sets of parameter assumptions were made and observed to see the optimum parameter values. To find the optimal value, a simulation-based observation was conducted to note the number of violations caused by a change in parameter *A* while keeping the other model parameters *B, n,* and *n'* constant. A bidirectional high-intensity pedestrian flow was established in the simulations to note these violations.

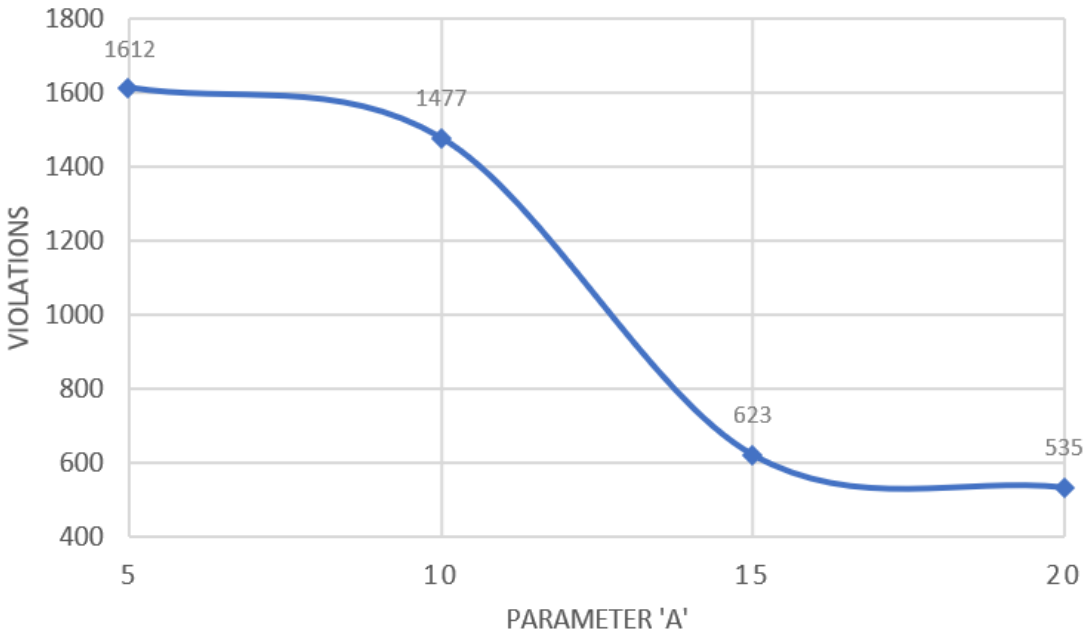

Figure 0-4 Change in violations with change in parameter 'A'

From Figure 4-4, we can see that as the force/value of *A* increases, the violations decrease. However, the curve for the number of violations flattens as the value is increased from 15 to 20. Therefore, for parameter *A*, a value of 15 is considered optimal to reproduce the scenario with social distancing measures.

4.4 Discussion of Results

Three scenarios were tested. The first scenario is the business-as-usual (BAU) scenario in which pedestrians are represented without social distancing measures. This scenario is produced by changing the value of parameter *A* to 4.5. This change in parameter *A* helps reduce the repulsion forces between the pedestrians, therefore, simulating pedestrian movements that indicate pedestrian movements before the deployment of social distancing measures. The second scenario was produced considering mobility restrictions in place along with social distancing measures and is referred to as CMR (COVID-19 scenario with Mobility Restrictions). The value of parameter *A* is now changed to 15 to increase the repulsive forces between pedestrians replicating social distancing by pedestrians and the pedestrian traffic is decreased by half of the first scenario. The last scenario represents the reopening stage when the mobility restrictions are lifted but with social distancing measures still in place. This scenario is referred to as Reopening scenario. The value of parameter *A* is same as the second scenario, but the pedestrian traffic is increased to that of BAU. One-hour simulations were run for all three scenarios.

Figure 4-5 represents an average pedestrian density graph. It can be seen that for BAU and Reopening scenarios, the pedestrian density is higher, which translates to the

pedestrian traffic being unchanged. For 'CMR', the pedestrian density is significantly decreased. It should also be noted that the pedestrian density for Reopening is slightly greater than BAU, which is expected as pedestrians take more time while travelling trying to avoid other pedestrians, which inherently increases the pedestrian density as the pedestrians stay longer in a given area.

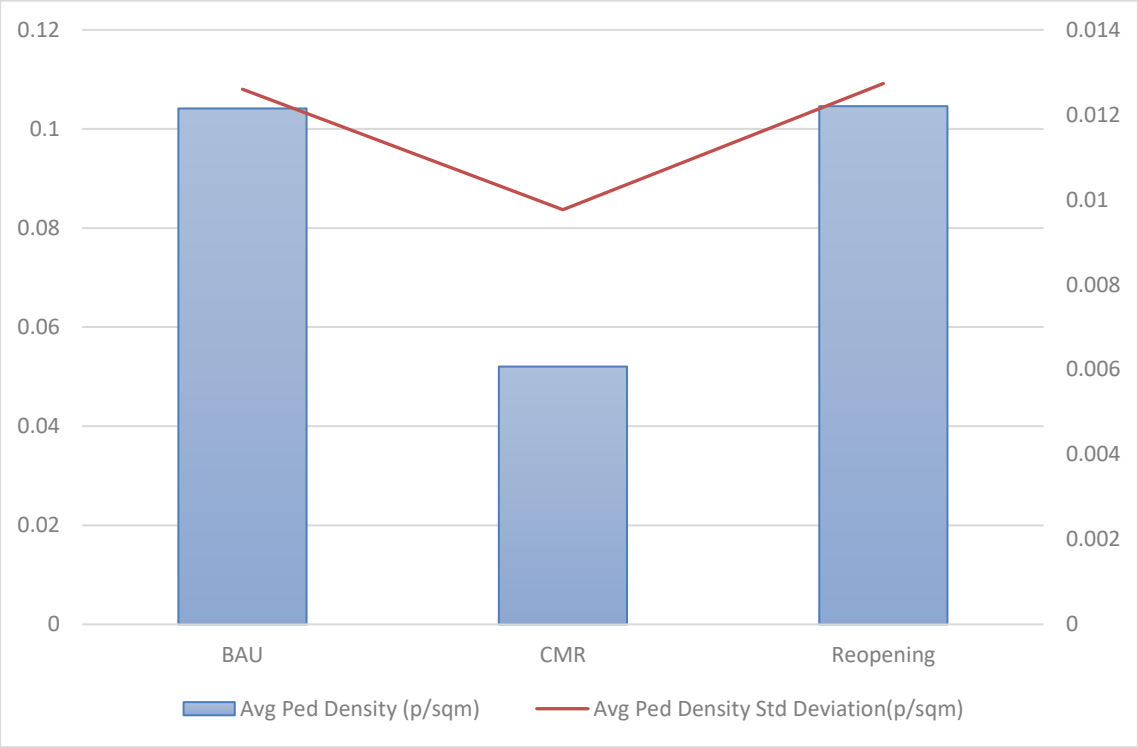

Figure 0-5 Average Pedestrian Density and its Standard Deviation

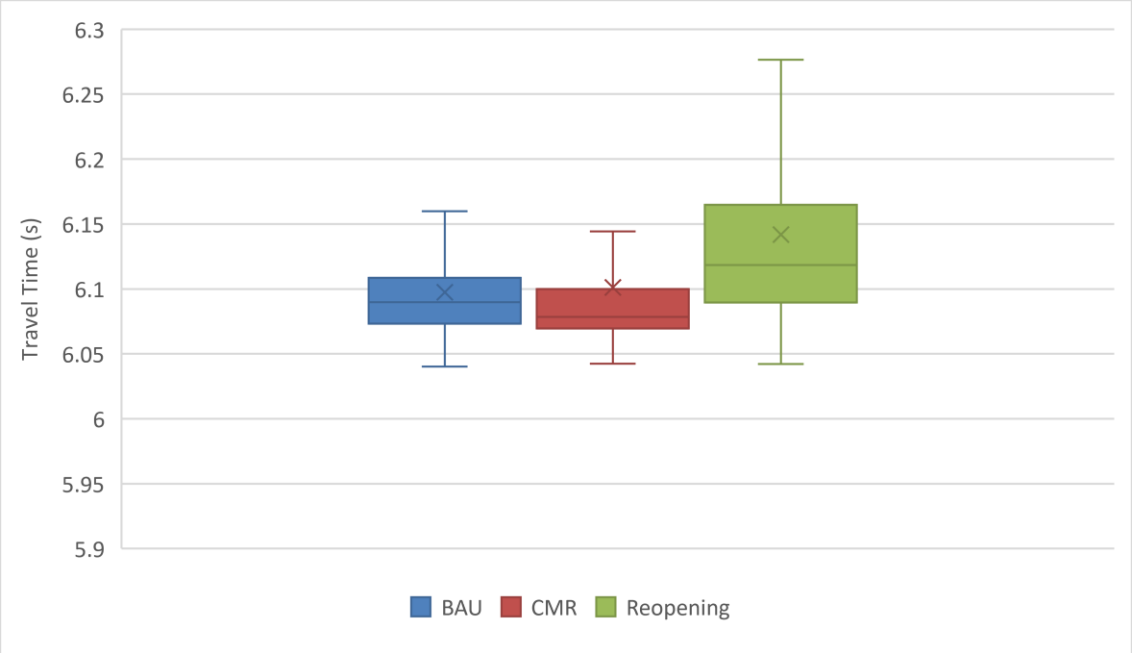

Figure 0-6 Box plot of average pedestrian travel time (s)

Figure 4.6 represents a box plot graph for average pedestrian travel time in which the minimum, maximum, and the mean of the travel times are indicated. The minimum travel times for all three scenarios are equal, which indicates that these pedestrians did not have to avoid other pedestrians. The Reopening scenario shows the highest average and maximum travel times of all three scenarios. This indicates that pedestrians in the Reopening scenario spent more time trying to avoid other pedestrians, thus taking longer than usual to reach their destinations. The CMR scenario also has higher average and max travel times than BAU despite having half the number of pedestrians in the same area.

Scenarios	*Number of Violations*	*% Reduction in Violations Compared to BAU*
BAU	914	n/a
CMR	205	77.57
Reopening	804	12.04

Table 0-1 Average time travelled and number of Violations

Finally, the number of violations in the sidewalk is considered, as seen in Table 4-1. BAU scenario is observed to have the greatest number of violations as no forces were driving the pedestrians apart. With a 77.57% reduction in violations than BAU, the highest decrease in the number of violations was noted in CMR scenario. Although the percentage reduction in violations is significant in the Reopening scenario, it pales in comparison to the CMR scenario. This could be caused due to the limited availability of space for pedestrians to move in the sidewalk. The Reopening scenario has the second-highest number of violations. The decrease in the number of violations in the Reopening scenario compared to BAU is noted to be 12.04%. This can be attributed to the increased repulsive forces between pedestrians trying to follow the social distancing measures. This shows how pedestrians contribute to decreased violations with mobility restrictions in place and could contribute to increased violations after the end of lockdowns. Measures should be taken not only to monitor these violations but also decrease these violations. Possible options for such measures is discussed in the next section.

4.5 Conclusion

This chapter examines the application of a game engine to simulate pedestrian traffic interactions in pandemic scenarios. Three scenarios were established to show and evaluate the impact of social distancing measures on pedestrian interactions. BAU scenario depicts a regular pedestrian flow which represents the case before COVID-19. CMR scenario illustrates social distancing in place with reduced pedestrian traffic representing mobility restrictions on pedestrians. The Reopening scenario represents a stage after the end of lockdowns which involves physical social distancing.

Various attributes were considered while evaluating the performance of all three scenarios. The Reopening scenario exhibited the highest pedestrian density indicating that pedestrian density increases as people stay longer on the sidewalk trying to avoid other pedestrians. The average time travelled is also considered. Maximum travel times were noted in the Reopening scenario as pedestrians spend more time trying to avoid other pedestrians. Although the CMR scenario has reduced pedestrian traffic, it showed higher travel times than BAU as pedestrians try to maintain physical social distancing. Finally, the number of violations is observed to reduce with social distancing measures in place. CMR scenario showed the greatest impact on the percentage in reducing social distancing violations which were promoted by physical social distancing combined with reduced pedestrian traffic caused by the existence of mobility restrictions.

When the pandemic comes to an end, there is a great chance of increased social distancing violations. This is a great cause of concern as reports state there is still a

chance for people to get infected even after taking vaccines. Therefore, measures should be taken to direct and monitor the pedestrian flows. For example, crosswalks wide enough to accommodate social distancing could be incorporated with visual cues to help pedestrians organize and redirect their walking behaviour. Such visual can include markings on pavements and crosswalks. Monitoring sensors/devices can also be placed to alert pedestrians when they violate social distancing.

This study shows how a game engine can be used to depict and simulate pedestrian-related simulations. This kind of methodology can be used to simulate other kinds of simulations such as evacuation planning in buildings/airports and virtual tours for new infrastructures. This proves that game engines provide great flexibility in designing the desired virtual environments for various projects/research. The developed game tool is also superior to other commercially available software as this tool provides excellent flexibility and can be enhanced further by integrating infectious disease spread models to analyze and visualize various solutions.

Chapter 5: Application of Game Engines in Analyses of Autonomous Vehicular Operations

5.1 Introduction

Self-driving autonomous technology in cars is on the horizon to begin new mobility possibilities of the future. Although the first attempt on AVs was attempted back in the early 1920s, it was in the year 2009 when Google launched its self-driving car project called Waymo (Ronan Glon & Stephen Edelstein, 2020). By the 2010s, major car manufacturing companies like BMW and Ford started working on their self-driving technology. Although the future is promising, autonomous vehicles are generally interpreted as vehicles that drive themselves or have complete control. To mitigate the confusion, AVs were defined/classified by the Society of Automotive Engineers (SAE) International, a U.S.-based professional association, in the following ways as described in Table 5.1.

Significant benefits emerge from using autonomous self-driving technology. These include reduced parking needs, increased vehicle sharing, increased road capacity, alleviating stress from drivers, increased mobility for non-drivers, and most importantly, increased safety (Lee, 2020). Apart from the potential to improve safety, a study states that autonomous technology will have tangible and quantifiable benefits (Fagnant & Kockelman, 2015). An economic analysis conducted by the authors states that AVs have the potential to save roughly 27 billion and 450 billion USD annually at 10 percent and

90 percent penetration rate, respectively, in the U.S. economy alone. Benefits like safety impacts, congestion benefits, and parking costs were considered for the analysis.

Level 1	*Level 2*	*Level 3*	*Level 4*	*Level 5*
Driver Assistance	Partial Automation	Conditional Automation	High Automation	Full Automation
Controlled by the driver with some assist features included in vehicle design	Vehicles has combined automated functions, like steering and acceleration, but driver must be engaged and monitor the environment.	Driver must be ready to take control but not necessarily be engaged.	Driver has the option to control but usually the vehicle is in control under limited conditions.	Vehicle is in full control under all conditions but driver has option to take control.

Table 0-1 Levels of Driving Automation (SAE J3016 2018)

In recent years, fully autonomous vehicles are being tested for their implementation in the real world. In 2019, Waymo started testing their autonomous self-driving cars on public roads without a driver in the driver's seat (Waymo, 2019). Experts believe that fifteen percent of the new cars sold in 2030 could be fully autonomous (Automotive revolution, 2016). While the transitioning to fully autonomous vehicles seem to have a positive impact, the transition itself could take years or possibly decades.

During the transition, the traffic will include various levels of vehicle automation. Testing these various vehicle combinations in the real world will take an enormous

amount of time. In such a situation, game engine-based tools help run virtual simulations necessary to test these theoretical traffic scenarios. Many studies were conducted to understand the impact of these traffic scenarios. However, this chapter observes and evaluates the differences between each of these scenarios by using a game engine for the first time. This provides superior flexibility in designing both the environment and the behaviour of pedestrians and vehicles. It also lets us examine the performance differences of networks containing human-driven and autonomous vehicles.

5.2 Literature Review

AV technology is gradually penetrating the global markets, but it might take decades for AVs to overtake existing vehicles and new vehicle purchases. However, studies show the impact of autonomous vehicles at different penetration rates. One such study analyzed the impact of connected autonomous vehicles (CAVs) on traffic safety under varying penetration rates (Ye and Yamamoto, 2019). Values like time to collision and frequency of dangerous situations were considered while assessing the safety impact. Results show that traffic safety is significantly increased with the increase in CAV penetration rates. The study also showed other positive impacts of increased CAVs on traffic flow, like increased smooth driving and decreased velocity differences between vehicles.

According to a survey based in the United States, human errors are responsible for ninety-four percent of the vehicle (National Highway Traffic Safety Administration, 2015). Such human errors could be avoided by introducing autonomous technologies. A study was conducted by Detwiller and Gabler (2017) on vehicle-pedestrian crashes,

which could have been avoided with the use of AVs. A total of 523 real crashes obtained from the National Highway Traffic Safety Administration (NHTSA) were studied. Two sets of code containing rule sets were coded by which the AVs would abide. The first set makes the AV abide by the traffic rules, and the second set asks the AV to drive cautiously whenever there is a possibility of a pedestrian conflict. The study concluded that upon using the first set of code, forty percent of crashes would have been avoided, and all but 27 out of 523 crashes would have been avoided by using the second set of code.

A 2013 study investigated near-miss situations with the help of drive recorders installed in passenger cars (Matsui, Hitosugi, Takahashi, et al., 2013). It was found that the average time to collision (TTC) for pedestrians without using a crosswalk was less than the ones who were using a crosswalk. The research showed that the average TTC for pedestrians emerging from obstructions was lesser than those who were not obstructed by objects such as buildings. Further research showed that when pedestrians emerge from obstructions, the TTC for vehicles and pedestrians was not significantly different (Matsui, Hitosugi, Doi, et al., 2013). As a result, it indicated a high risk of collision if the driver and the pedestrian were not paying attention.

A study by Combs et al. (2019) indicated that AVs with varying technologies could detect pedestrians in advance of fatal collisions. The study was conducted by involving three activities. First, the study established the functional range of advanced, up-to-date pedestrian sensor technologies. Second, they collected data from the Fatality Analysis

Reporting System for each state in the U.S. and the district of Columbia. Lastly, they assessed the maximum number of fatalities involving pedestrians that could have been avoided had the vehicles been replaced with the autonomous versions.

Will there be any difference in the pedestrian's behaviour given that the future technology will be self-aware and be better at risk assessing? Millard-Ball, A. (2018) used a game theory to analyze the interactions between pedestrians and autonomous vehicles. Game theory is used to analyze situations involved with people interactions in which the actors participating in the analysis are influenced by the result of their interactions. The author interpolates the pedestrians crossing at a crosswalk to a game of chicken. The author introduces a new model that formalizes the game "crosswalk chicken" between the pedestrians and the autonomous vehicles. The author suggests that as the autonomous vehicles are more risk-aware, the pedestrians cross with impunity giving them an edge over autonomous technology. This in turn, decreases the efficiency of autonomous traffic flow, making human driving more desirable as they tend to have the upper hand in the game and can approach their destination much faster. The author finally emphasizes that besides the technological progress, legal and urban planning response would have a massive impact on autonomous vehicles.

Nevertheless, does implementing autonomous technology improve the overall network performance? A micro perspective-based study on mixed traffic systems indicates that even having a low ratio of HVs in the network would cause a considerable negative impact on the network performance (Chen et al., 2020). A 2016 study states that

there is a potential for reducing network performance considering user preference (Atkins, 2016). However, the research also states that AVs would be beneficial in congested networks as the vehicle driving behaviour influences the network density. Another study shows that by implementing simulation-based surrogate safety measures and high penetration rates, there was a significant reduction in the number of conflicts between vehicles despite the modelling for shorter headways (Morando et al., 2018). One study suggests an infrastructure for the coexistence of HVs and AVs where AVs will be given exclusive lanes (Santana et al., 2021). The resultant was reduced average travel time and space. The research uses mesoscopic traffic simulation to assess the system in realistic scenarios.

Research on autonomous vehicles is expected to increase drastically. This can only be done by simulating autonomous vehicles because testing autonomous vehicles in the real world will take an enormous amount of time and resources. One study challenges the practicality of testing the safety aspect of autonomous vehicles in the real world (Kalra & Paddock, 2016). The authors calculated the number of years it would take to test autonomous vehicles and found that it would take hundreds of millions of miles to billions of miles. This is certainly not possible, and therefore simulations should be carried out for testing various aspects of AVs. Although there are commercially available software programs that can fulfil this requirement, they do not match the flexibility of a game engine. For example, users can use a newly developed vehicle model embedded with virtual sensors in a simulation that take advantage of a virtual environment. This chapter demonstrates the flexibility of using a game engine in autonomous vehicles

research by simulating scenarios with different traffic conditions involving AVs, HVs, and mixed traffic and evaluates their performance and differences.

5.3 Modelling Approach

The same virtual environment used in the previous chapters is used in this chapter. However, different vehicular and pedestrian inputs are used in different scenarios. Additional behaviour characteristics are added to vehicles to differentiate between AVs and HVs. Pedestrians are designed with a simple origin-to-destination system. As the chapter focuses on traffic safety, simulations were run to capture pedestrians crossing the road as it is the most vulnerable time during pedestrian and vehicle interactions. Therefore, pedestrian forces between the pedestrians were removed during road crossings. Three separate *scenes* were created in Unity for three scenarios. The first scenario evaluates the performance of the network containing only AV traffic. The next two scenarios contain HV traffic and mixed traffic, respectively. Different scripts were also used to retrieve results.

Vehicle Behaviour Parameters

Vehicle behaviour parameters let users characterize the movement of vehicles. By changing vehicle behaviour parameters, we can model the driving behaviour of vehicles to AVs or HVs. The longitudinal behaviour of the vehicles can be modelled by using the parameters shown in5-2 (Atkins, 2016).

Parameters	*Description*
Standstill distance	The desired distance between stopped vehicles
Headway time	The gap that a vehicle keeps in seconds
Following variation	The additional distance to the allowed safety distance that is permissible before the vehicle comes closer to the preceding vehicle
Negative following threshold and Positive following threshold	Control speed differences during car following
Speed dependency of oscillation	Influence of distance on speed oscillation
Oscillation acceleration	Influence of vehicle acceleration during car following oscillation
Standstill acceleration	Desired acceleration when starting from standstill
Acceleration at 80km/h	Desired acceleration from a speed of 80km/h
Smooth closeup behaviour	Vehicles slow down more evenly when approaching a standing obstacle

Table 0-2 Vehicle Behaviour Parameters

For this study, the below four parameters were used. They are:

- Standstill Distance
- Headway Time
- Following Variation
- Speed dependency of oscillation

Standstill Distance:

This is the desired distance maintained between two vehicles when they are in a queue, for example, at a signal intersection during the red light. Given that the AVs are much more responsive and sensitive, they can have a smaller standstill distance than human-driven vehicles. For this simulation, we considered a standstill distance of 1m and 1.5m for AVs and HVs, respectively. To produce this difference in the vehicle simulation, the value for standstill distance in equation (1) was changed respectively for AVs and HVs.

Headway Time:

Headway time is the distance in seconds a vehicle maintains with the vehicle in front while moving to avoid a rear-end collision. Again, as AVs are more responsive their headway can be shortened compared to HVs. For our simulation, we assigned a headway time of 0.9s for HVs and 0.5s for AVs.

For this thesis, the below equation is created to calculate the acceleration of each vehicle. The equation uses the values of established headway times to determine the

acceleration of the wheels of the vehicle which helps to propel the vehicles forward. The formula is as follows:

$$a = \frac{(maxspeed - u)}{t_m} \quad (3)$$

Where,

a = acceleration in m/s^2

maxspeed = desired vehicle speed in m/s;

u = the vehicle's current speed in m/s;

t_m = headway time in seconds.

Following Variation:

Following variation is the distance that is maintained in addition to the safety distance. This value is modelled after human-driven vehicles. For AVs, this value is equal to zero as they are more efficient compared to HVs. HVs were given a value of 4m. In the simulation, a random value between 0m to 4m is assigned to every vehicle. To incorporate this parameter into the *Raycast* system, a variable *ad* is added to equation (1) which produces the below formula:

$$d_s = ax + bx + ad \quad (4)$$

Where,

d_s = safety distance;

ax = standstill distance;

bx = time adjusting requirement values;

ad = following variation value in m.

Speed dependency of oscillation:

Speed dependency of oscillation is the influence of distance on speed. It is the variation of speed for the vehicle around the desired speed. This value oscillates between 0 to the maximum given value depending upon the distance of the vehicle with the vehicle in front. The closer the vehicle to the preceding vehicles, the lesser the variation and vice versa. For this simulation, we assigned a variation of 1m/s for HVs. As AVs are more precise in assessing the distance between vehicles, the speed dependency of oscillation would be zero. This parameter is incorporated by adding it to the maximum allowed speed limit of the given network which produces the below equation:

$$maxspeed = msv + sov \tag{5}$$

Where,

maxspeed = desired vehicle speed in m/s;

msv = maximum speed allowed in m/s;

sov = speed dependency of oscillation in m/s. The formula of *sov* is as follows:

$$sov = \frac{Val * \frac{(refval - d)}{(16 - refval)}}{\frac{(refval - d)}{(16 - refval)} + 1} \tag{6}$$

Where,

sov = speed dependency of oscillation in m/s ;

d = safety distance in m;

refval = The vehicle's distance from the preceding vehicle in m;

Val = The maximum value for speed variation in m/s.

5.4 Saturation Flow

Saturation flow is an essential indicator for the performance of a network. Saturation flow is the number of vehicles that pass through the intersection during the green time of the intersection per hour. To evaluate the performance of our network, we used virtual saturation flow sensors that record the time the rare axle of a vehicle crosses the stop line. This is achieved using an invisible *GameObject* to represent the centre of a vehicle's rear axle, as shown in Figure 5-2. From the retrieved data, the saturation flow rate of the network is calculated. There are two sensors for each lane that would note the vehicle and the time when the vehicle passes by the sensor.

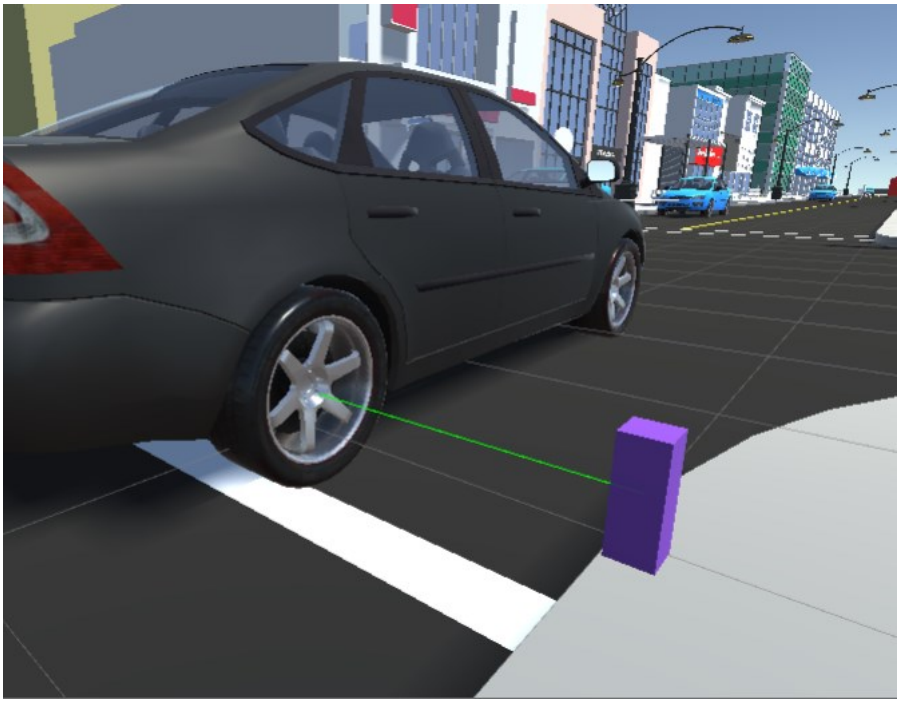

Figure 0-1 Saturation Flow Sensor Recording a Car Crossing the Intersection 'A'

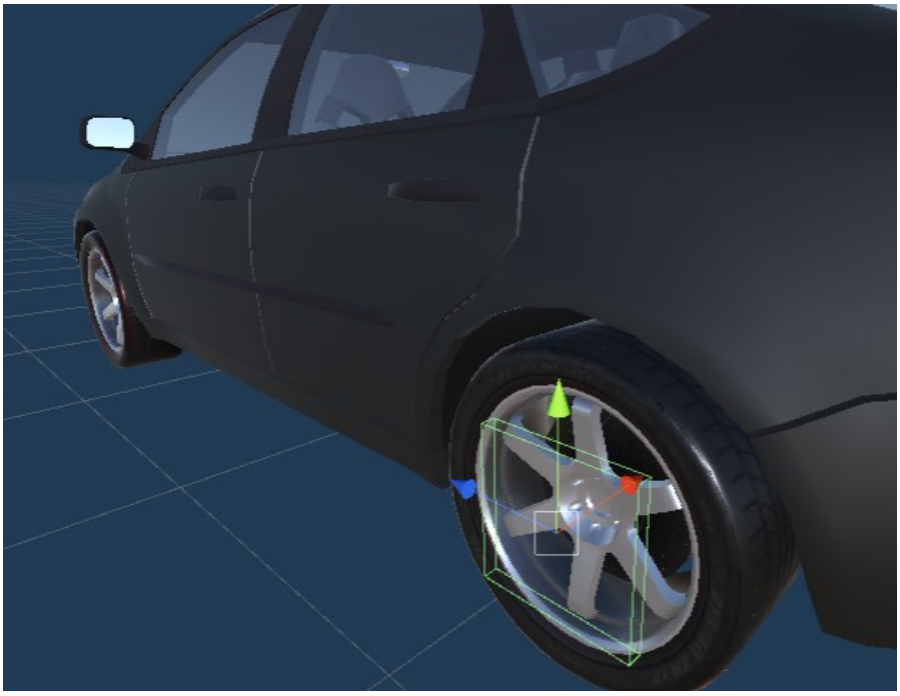

Figure 0-2 Invisible GameObject Representing the Location of Vehicle's Rear Axle

The formula for saturation flow is as follows (Bester and Meyers, 2007):

$$s = \frac{3600}{h_s} \tag{7}$$

Where,

s = saturation flow rate in vphgl;

3600 = number of seconds per hour;

h_s = saturation headway in s.

The formula for saturation headway is as follows:

$$h_s = \frac{\sum_{j=n_q}^{l} h_j}{(l + 1 - n_q)} \tag{8}$$

Where,

h_s = saturation headway in s;

l = last queued vehicle position;

h_j = headway of j^{th} queued vehicle in s;

n_q = position of queued vehicle from where saturation flow region started.

5.5 Near Misses

To evaluate and compare the safety aspect of HVs and AVs, we simulated scenarios where the vehicles interact with the pedestrians crossing the road. The vehicles avoid these humans with the help of their *Raycast* sensor system. However, there might be cases when the cars come too close to the pedestrians. To detect these situations, an

additional sensor system is created. When a car comes one meter or closer to a pedestrian, the incident is detected, and the distance at which the near-miss incident is detected is recorded. This incident is called a near-miss incident. Figure 5-3 shows the detection of a pedestrian during a near-miss incident.

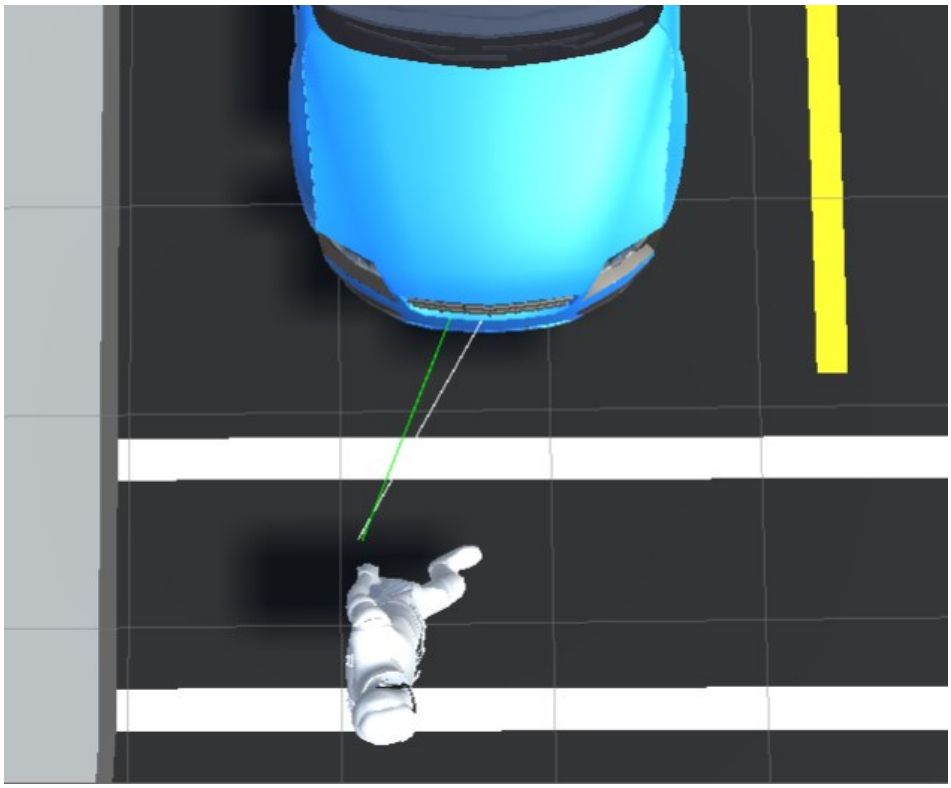

Figure 0-3 Vehicle Detecting a Near Miss Incident at Intersection 'B'

These incidents involving near misses are then identified and stored in an excel sheet for later evaluation. To identify these incidents, the vehicles are modelled to have sensors that detect the humans in front. For this study, the sensors detect any humans at the intersection 'B' that come close to 1m of the front of the car. A representation of this can be seen in Figure 5-4.

Following the detection, data is entered into an excel sheet with the below information:

- Time of incident in seconds
- Name of the pedestrian
- Name/type of the vehicle

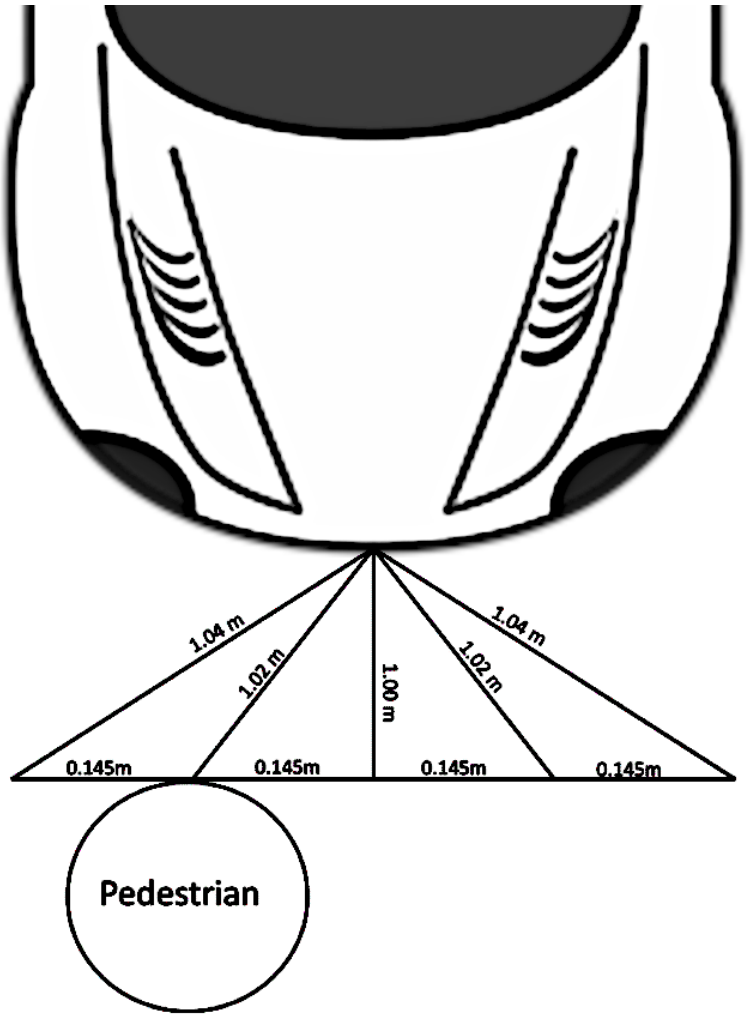

Figure 0-4 Representation of vehicles' Virtual Near Miss Detection Sensor System

5.6 Level of Service

Lane capacity can be assessed in terms of level of service (LOS) at signalized intersections. LOS is a qualitative measure used to describe operational conditions within a traffic stream (Council & TRB, 2000). Although parameters like speed and travel time can be used to calculate LOS, average delay of a vehicle at intersections can also be used. For this thesis, control delay is used to measure LOS for a lane at intersection A (intersection at Spring Garden Road and Dresden Row). The control delay *d* is divided into three parts: uniform delay d_1, incremental delay d_2, and initial queue delay d_3 (Council & TRB, 2000). Below are the formulae for calculating control delay:

$$d = d_1(PF) + d_2 + d_3 \tag{9}$$

$$d_1 = \frac{0.5C\left(1 - \frac{g}{C}\right)^2}{1 - \left[\min(1, \mathrm{X})\frac{g}{C}\right]} \tag{10}$$

$$d_2 = 900T\left[(X-1) + \sqrt{(X-1)^2 + \frac{8kIX}{cT}}\right] \tag{11}$$

Where,

d_1 = uniform delay d1 in s;

d_2 = incremental delay accounting for effect of random and oversaturation in s;

d_3 = initial queue delay in s/veh;

X = volume to capacity ratio (v/c);

c = capacity in veh/h;

g = green time in s;

C = cycle length in s;

T = duration of the analysis period in h;

I = upstream filtering adjustment factor;

k = incremental delay factor varying in value from 0.04 to 0.50, For fix-timed signals k = 0.5 for M/D/1 queuing system as an approximation of queuing system at signalized intersections.

PF = progression adjustment factor which is calculated by the below formula:

$$PF = \frac{1 - P}{1 - \frac{g}{C}} \tag{12}$$

Where,

P = proportion of vehicles arriving on green volume-to-capacity ratio;

g = green time in s;

C = cycle length in s;

When a queue from a previous signal cycle is present at the start of the analysis, newly arriving vehicles will experience a delay which is called as the initial queue delay and is represented as d_3. If there is no initial queue d_3 will be equal to zero. Once the average control delay is achieved it can be compared with the table from Table 5-3 (Highway Capacity Manual, 2010).

LOS	Control Delay per vehicle (seconds per vehicle)
A	≤ 10
B	> 10-20
C	> 20-35
D	> 35-55
E	> 55-80
F	> 80

Table 0-3 Level of Service Standards for Signalized intersections

5.7 Discussion of Results

Two sets of scenarios are tested. For the first set, real-world traffic counts were retrieved from Halifax Regional Municipality and DalTRAC to test traffic performance of various kinds of traffic in present conditions. The data includes pedestrian and vehicle counts for morning and evening peak times. The selected peak times denote the busiest times of the day during the morning and evening hours at Spring Garden Road. These busiest times are the result of the location of schools, offices, restaurants, and other attractions at Spring Garden Road. The selected times include 8 am to 9 am for the morning hours and 5:30 pm to 6:30 pm during the evening hours. This kind of testing would bring insight into the present traffic conditions when the traffic is replaced entirely or partially by autonomous vehicles. The types of traffic used for these scenarios are AV, HV, and Mixed (equal mix of HVs and AVs). HV traffic resembles the current situation where the traffic does not consist of fully autonomous vehicles, and AV traffic resembles traffic

consisting of only fully automated vehicles. For mixed traffic, the traffic consists of half AVs and half HVs. This traffic resembles the traffic condition when half of the vehicular fleet is occupied by AVs which is believed to happen by the year 2060 (Lee, 2020).

One-hour simulations were run to assess the performance of the network. Basic data like max allowed speed for the network, vehicle and pedestrian counts, and signal cycle timing were set according to the scenarios to achieve and observe results like average speeds of the vehicles, vehicle travel times, saturation flows, and near misses. Figure 5-5 shows the average speed and their standard deviation of vehicles recorded during each simulation. The data was retrieved by recording the average speed of each vehicle during its time in the simulation. The time of the vehicle spent in the simulation is defined as the time taken by a vehicle to reach its destination from the second it was spawned. The average speed is calculated by dividing the total distance travelled by the vehicle by the amount of time taken to travel the distance.

It can be seen that during the morning peak time, AVs have the lowest average speed and HVs have the highest average speed. This is because humans do not precisely calculate the required acceleration; they tend to cross the maximum speed limit allowed in the given area. The speed dependency of oscillation helps show this difference. The value *sov* helps mimic the behaviour of human drivers, which therefore resulted in a higher average speed. The average speed standard deviation is also higher for HVs due to the difference in maximum achieved speed for each vehicle. For mixed traffic, the average speed and its standard deviation are lower than AVs but higher than HVs. This is

caused due to the combined traffic of HVs and AVs. The results recorded for evening peak time are similar to that of the morning peak time but with lower average speeds contributed by a higher traffic volume. Traffic with only HVs and mixed traffic scenarios were observed to have a considerable decrease in average speeds compared to traffic with only AVs, which is contributed by the following variation character present in HVs.

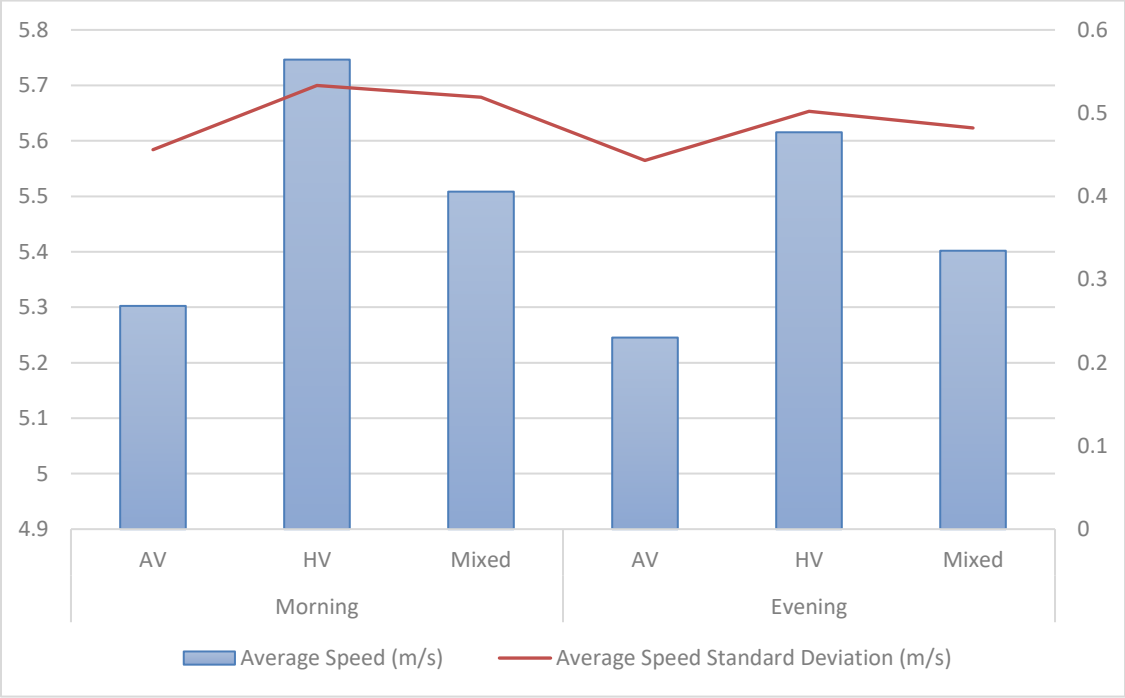

Figure 0-5 Averages of Vehicle Speed and their Standard Deviation Classified by Vehicle Types and Time of the Day

Figure 5-6 shows the travel time averages of AVs, HVs, and mixed traffic scenarios classified by morning and evening peak times. The graph indicates that AVs take longer to complete their trip compared to HVs and mixed traffic. This is caused by the strong abidance of AVs' speed to the maximum allowed speed limit of the network, resulting in more extended trips. HVs show the least amount of travel time caused by

higher velocities achieved when compared to AVs. Although this is beneficial in completing the trip faster, it is not suggested to surpass the speed limits of a given area. Mixed traffic shows the highest travel time standard deviation for both peak times, which could be attributed to the difference in speeds achieved by AVs and HVs.

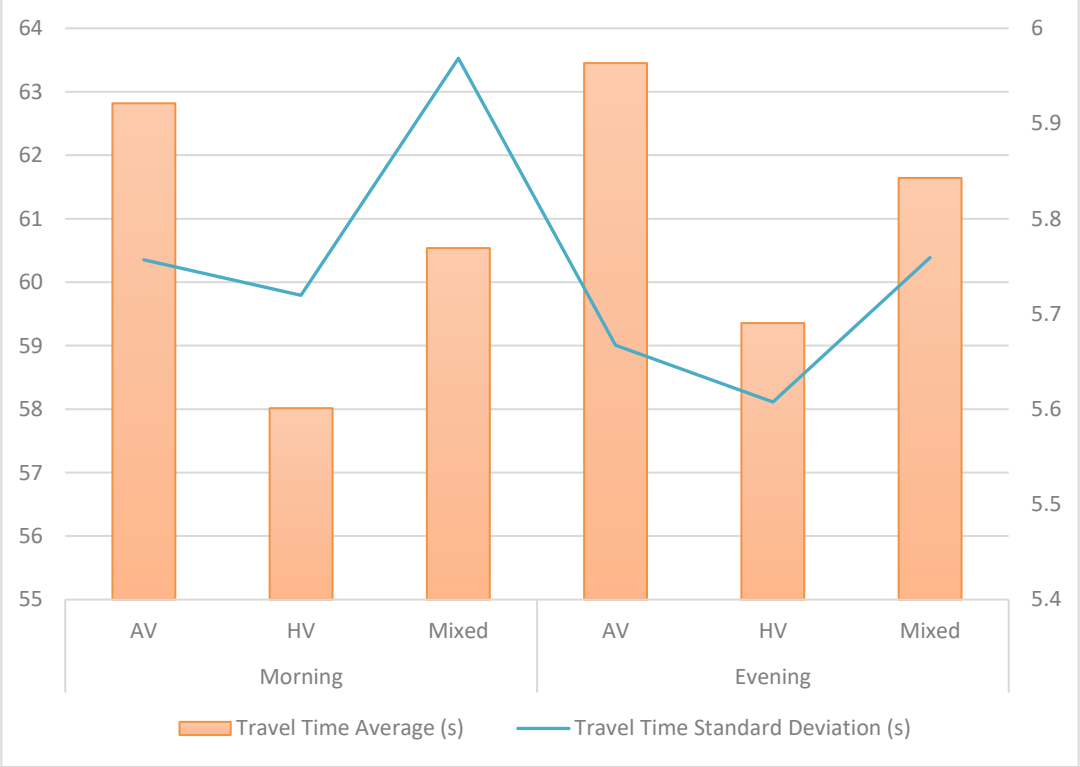

Figure 0-6 Averages of Travel Times and their Standard Deviation Classified by Vehicle Types and Time of the Day

Table 5-3 shows the average distance of the pedestrian from a vehicle when a pedestrian comes close to a car within one meter or less. The closest distance of this interaction is noted and exported. During morning peak time, HVs are observed to have a higher average distance at which a near miss is detected. This is because HVs have a higher velocity on which the length of the *Ray* is calculated. This helps in detecting the

pedestrians sooner and helps the vehicle to stop earlier when compared to AVs. For evening peak times, the traffic flow is noted to increase. Under these conditions, at lower speed, the *Raycast* system built to detect pedestrians are blocked by the vehicles in front. These allow lower velocity vehicles (AVs) to stop faster and higher velocity vehicles (HVs) to stop slower, therefore, decreasing the distance between the vehicle and the pedestrian. Mixed traffic vehicles have the lowest average distance in both morning and evening peak times. It should also be noted that these distances are perpendicular distances between vehicles and pedestrians.

Peak Times	***Vehicle Type***	***Near Misses Average Distance (m)***	***Near Misses Distance Standard Deviation (m)***
Morning	AV	0.955354	0.032196837
	HV	0.96762	0.010890927
	Mixed	0.938191	0.035260388
Evening	AV	0.951437	0.037216654
	HV	0.944928	0.035673447
	Mixed	0.941832	0.039118874

Table 0-4 Near Miss Incidents Detection Distance for Peak Times

The second set of scenarios were tested to evaluate the performance of the designed network at various intensities of pedestrian traffic flow. Three different pedestrian flow intensities were tested, which are referred to as mild, moderate, and intense. This would test the capabilities of modelled vehicles in mild to intense crossing scenarios. For mild, moderate, and intense scenarios, a pedestrian flow of 300, 500, and

700 pedestrians per hour are chosen respectively to cross the intersection 'B'. Figure 5-7 shows the average speeds of vehicles at different intensities of pedestrian crossings. It is observed that the average speed of vehicles decreases with the increase in pedestrian flow. This is because the *Raycast* system used for pedestrian detection can detect more than one pedestrian. As the pedestrian intensity increases, there is a greater chance for the vehicles to let multiple pedestrians cross the road at once. It can be seen that the mild and moderate scenarios are similar but with overall reduced average speeds. Both scenarios state that HVs have the highest average speed and AVs have the lowest, which is contributed by the vehicle behaviour parameters like speed dependency of oscillation. For the intense scenario, AVs are observed to have a slightly higher average speed. This is due to the decreased length of *ray* cast for pedestrian detection, which would let the vehicle move swiftly between pedestrian crossings.

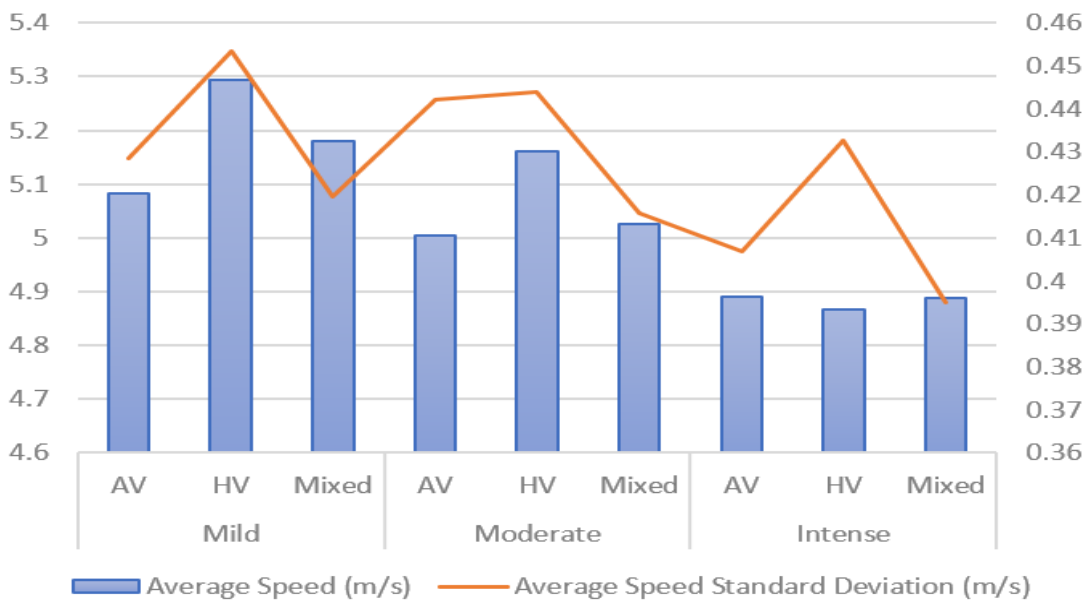

Figure 0-7 Averages of Speeds and their Standard Deviation Classified by Vehicle Type and Pedestrian Flow

Figure 5-8 shows the average travel times of vehicles at different intensities of pedestrian crossings. Again, the mild and moderate scenarios are similar but with an increased average travel time for the moderate scenario. AVs are noted to have the highest travel times for both scenarios, and HVs have the lowest travel time. This is caused by the lower speeds for AVs and higher speeds for HVs. For the third scenario, travel times are noted to increase significantly when compared to the mild scenario. This can be explained by the increased waiting times at the intersection 'B'. As stated before, AVs have lower *ray* length that would help move the vehicle swiftly between crossings. Mixed traffic performance takes second place in all three scenarios.

Figure 0-8 Averages of Travel Times and their Standard Deviation Classified by Vehicle Type and Pedestrian Flow

Table 5-4 shows the saturation flow for the network at intersection 'A'. The results are calculated by assuming the lane is green for the hour. The saturation flow for AVs is

higher than HVs, which indicates that a network having only AVs is more efficient than having HVs. This efficiency is caused by the reduced following distance and standstill distance of AVs. The saturation flow of mixed traffic is higher than HVs and lower than AVs, which can be interpreted as the result of having both HVs and AVs in the network.

Vehicle Type	***Hs (s)***	***Saturation Flow Rate (vphgl)***
AV	2.12	1698.11
HV	2.45	1469.39
Mixed	2.24	1607.14

Table 0-5 Saturation Flow Calculated for the Network for Each Vehicle Type

Table 5-5 shows the average distances of pedestrian detection from a vehicle at various intensities of pedestrian crossings. AV and HV traffic showed similar average distances, while mixed traffic showed the greatest average distance in the mild scenario. This is attributed by the change in vehicle compositions where the randomly assigned *sov* values could be higher for HVs. For the moderate scenario, HV traffic has a higher average distance which could be possible due to an increase in sensor length caused by the increased values of *ad*. For the intense scenario, HV traffic scored the lowest average distance by the higher number of vehicles queueing at intersection 'B', causing their *Raycast* system sensors to be blocked by the vehicles in front.

Pedestrian Intensity Scenario	*Vehicle Type*	*Near Misses Average (m)*	*Near Misses Standard Deviation (m)*
Mild	AV	0.894241	0.119371
	HV	0.89325	0.122478
	Mixed	0.921881	0.088999
Moderate	AV	0.875034	0.168599
	HV	0.914163	0.099726
	Mixed	0.940626	0.059992
Intense	AV	0.913232	0.126601
	HV	0.854776	0.161284
	Mixed	0.931037	0.090967

Table 0-6 Near Miss Incidents Detection Distance for various Pedestrian flows

Table 5-5 shows the average distances of pedestrian detection from a vehicle at various intensities of pedestrian crossings. AV and HV traffic showed similar average distances, while mixed traffic showed the greatest average distance in the mild scenario. This is attributed by the change in vehicle compositions where the randomly assigned *sov* values could be higher for HVs. For the moderate scenario, HV traffic has a higher average distance which could be possible due to an increase in sensor length caused by the increased values of *ad*. For the intense scenario, HV traffic scored the lowest average distance by the higher number of vehicles queueing at intersection 'B', causing their *Raycast* system sensors to be blocked by the vehicles in front.

Table 5-6 shows the delays calculated to estimate LOS at intersection A. For all three traffic types, the initial queue delay d_3 is noted to be zero as they were no vehicles queueing in the beginning of the analysis. For AV traffic the uniform delay is noted to be the lowest given their shortened sensor length produced by the vehicles' lower speed. HV traffic has the highest uniform delay due to its higher speed. The incremental delay is highest for the HV traffic followed by mixed traffic. This is caused due to HV traffic's lower saturation flow rate as their increased *ray* length would cause human driven vehicles to arrive slowly at the stop lines compared to autonomous vehicles despite their higher speed. Finally, the control delay is noted to be the highest for HV traffic which would score an LOS level of 'C' after comparing the results with Table 5-3. For AV and mixed traffic, the LOS is noted to achieve a level of 'B'.

Traffic Type	Uniform Delay d_1 (s)	Incremental Delay d_2 (s)	Initial Queue Delay d_3 (s)	Control Delay *d* (s)	LOS
AV	10.63	4.95	0	15.58	B
HV	11.39	10.42	0	21.81	C
Mixed	10.89	6.6	0	17.49	B

Table 0-7 Delays Calculated to Estimate Level of Service

5.8 Conclusion

This chapter examines the performance of the network under various kinds of traffic using a game engine. The kinds of traffic include AVs, HVs, and mixed traffic containing equal units of both HVs and AVs. Two sets of scenarios were tested. The first scenario tests network traffic performance during morning and evening peak times, and the second

set of scenarios tests the performance of the network traffic under mild to intense pedestrian flow conditions.

In the first set of scenarios, AVs were observed to have the lowest average speeds and highest average travel times. HVs were observed to have the highest average speeds and lowest average travel times. These results were contributed by the designed vehicle behaviour parameters for AVs and HVs. The vehicle parameters included headway time, standstill distance, following variation, and speed dependency of oscillation. The average distances at which the near misses were detected were also noted. The length and angle of the sensor ray played a vital role in the results as HVs were observed to have higher average distance during low vehicular traffic and lower average distance during higher vehicular traffic.

In the second set of scenarios, the average speeds of the vehicle for mild and moderate pedestrian flow scenarios show that AVs were able to maintain a lower speed than HVs, indicating that as humans do not precisely calculate the required acceleration, they have the tendency to cross the maximum speed limit allowed in the given area. The speed dependency of oscillation helped show this difference. In all scenarios, the average speed of vehicles in mixed traffic always stayed in between the range of traffic with only HVs and only AVs. For the intense pedestrian flow scenario, the average speeds of all three traffic types were noted to be less than the former two scenarios. This could be attributed to the fact that as more pedestrians were crossing the road, the vehicles had to stop for a more extended amount of time. AVs showed a greater average travel time than

HVs and mixed traffic for mild and moderate pedestrian flows. As AVs continuously operated at speed below the maximum allowed speed of the given network, they took more time to complete their journey. Saturation flow results show that networks with only AVs are more efficient as they help produce higher saturation flow rates than HVs and mixed traffic. The near-miss incident results showed lower values for HV traffic in an intense scenario, which was caused by a higher number of queued vehicles at the unsignalized intersection. The mixed traffic showed the highest average distances of near-miss incidents for all three scenarios, which is caused due to the varied speed differences between vehicles. LOS was calculated at signalized intersection 'A' using the saturation flow rates and control delay times for all three traffic types. HV traffic scored a worse LOS with a level of C. This can be explained by its greater incremental delay compared to AV and mixed traffic. Mixed traffic achieved an LOS of 'B' which is the same as AV traffic, but it is worth noting that the control delay time was higher for mixed traffic indicating that more time was lost at the intersection due to the presence of HVs in the composition.

Therefore, the application of a game engine in evaluating the performance of networks with different types of traffic is proved to be valid. The virtual 3D network can be expanded to multiple intersections and lanes to test and solve complex traffic microsimulation problems. Different types of data required can also be retrieved by using various scripts. The virtual models of AVs can be further improved by designing virtual technology in the 3D environment. Virtual infrastructure like charging stations or battery swap stations can also be incorporated to support the penetration of AVs into global

markets. The scope of expansion for such improvisations is unlimited, and it is only possible with the help of game engines.

Chapter 6: Conclusion

This thesis presents a framework to develop a game engine-based tool that uses a 3D virtual environment for traffic analysis purposes. The developed game engine-based tool is used as used for analyzing traffic microsimulations. To achieve this, first, a 3D model is built using Sketchup, a 3D modelling computer program. The 3D model built representing the Spring Garden Road in Downtown, Halifax, is then imported into Unity, and therefore the 3D environment is established. The established static 3D environment then needs to be enhanced to a dynamic animated environment. This was achieved by adding scripts to *GameObjects*. The scripts add various functionality to each *GameObject*. The *GameObjects* include 3D models like cars, trees, and humans imported from the Unity asset store. The 3D models were modified accordingly by attaching various components that aid the *GameObjects* by providing visual characteristics and providing them with the ability to interact with the virtual environment. These components include *Rigidbody, Mesh Renderer, Box Collider, Wheel Collider, NavMesh Agent,* and *Animator*. Various scripts are then added to the required *GameObjects* as components. Example functionalities of the scripts include signal systems, vehicle behaviours, pedestrian behaviours, spawning, destinations, destroying *GameObjects*, wheel rotation, and data retrieval. The movement of cars is designed with the help of Unity physics, and the movement of pedestrians is designed with the help of *NavMesh*, a mesh generated in Unity to build walkable areas in the virtual environment. The 3D virtual environment is then tested by running the simulation

and then checking standstill distances between vehicles, signal operations, vehicle and pedestrian movements, and exported data.

After testing the built environment, the 3D platform is then used for various applications. The capability of using a game engine in pandemic-related pedestrian studies is tested. This is done by simulating three scenarios in which the movements of pedestrians are produced with the help of a social force model. The first scenario, BAU, depicts the movement of pedestrians on a sidewalk before COVID-19. The second scenario depicts the pedestrian movements trying to socially distance between each other by maintaining a two-meter gap with other pedestrians while waking. The second scenario also has mobility restrictions which is achieved in the simulation by decreasing pedestrian traffic. The third scenario depicts the same pedestrian movements as the second scenario, but the pedestrian traffic is again increased to the same as BAU. This scenario represents a reopening phase where mobility restrictions are lifted. All three scenarios are analyzed to observe the number of violations in social distancing. Data like average travel time of pedestrians and pedestrian density were analyzed to understand the outcome. It is noticed that with the increase in social forces like repulsion between pedestrians increases travel time. This study shows the possible violations in pre-pandemic, pandemic, and post-pandemic scenarios. It was noticed that post-pandemic, the number of violations are still significant even with social distancing forces. To control these violations measures should be taken to help people organize their walking behaviour. Such measures include visual cues and violation monitoring devices. The

game engine tool was therefore proved useful to study pedestrian interactions and hence can be used to perform pedestrian microsimulations.

Another significant contribution from this thesis is evaluating the performance of AV, HV, and mixed traffic. Vehicle behaviour characteristics were added to the vehicles by modifying scripts. The modified scripts included additional vehicle behaviour parameters like standstill distance, headway time, following variation, and speed dependency of oscillation. These parameters helped in differentiating HVs from AVs. The performance evaluation for all three traffic types is analyzed with the help of two sets of scenarios. The first set of scenarios were modelled after the morning and evening peak time traffic counts of Spring Garden Road. It was noticed that AV traffic had the lowest average speeds and HV traffic had the highest average speeds, which were contributed by the speed dependency of oscillation parameter.

As a result, AV traffic took longer to complete their trips, and HV traffic took the lowest time. Average near-miss distances were also calculated, and it was observed that AV traffic recorded the lowest average near miss distance during the morning peak time and the highest average near miss distance during the evening peak time. It was noticed that the change in vehicular traffic intensity has contributed to these results. The second set of scenarios constituted three scenarios; mild, moderate, and intense, which depicted low to high-intensity pedestrian flow scenarios. These scenarios were tested to evaluate the three types of vehicular traffic in various pedestrian flow situations. For all vehicular traffic types, mild and moderate scenarios showed similar results but lower average

speeds and higher travel times with increased pedestrian traffic. AV traffic had the lowest average speeds and highest travel times, while HV had the highest average speeds and lowest travel times. Mixed traffic always produced result values between AV and HV traffic. For the intense scenario, AV recorded the highest average speeds and lowest travel times due to their shortened sensor *ray* lengths which were contributed by the null values of the following variation parameter. The network's saturation flows for different vehicular traffic are calculated and found that AVs have the highest saturation flow rate, and HVs have the lowest saturation flow rate. Finally, LOS was calculated for all three traffic types. HV traffic received a lower LOS when compared to AV and mixed traffic which was caused due to lower saturation flow rates.

Although this thesis proves that game engines are helpful in developing new virtual tools for research, they do have certain limitations. Computer hardware that constitutes the graphical power or processing power is one of the critical limitations for using this kind of framework. This thesis is enclosed to the simulation of one street, but large-scale projects involving multiple streets or cities might require an initial investment in powerful computer hardware. The second limitation of using a game engine-based tool is the reliability of its physics engine. Establishing wheel physics for this thesis took multiple trial and error sets. However, this can be fixed with future software updates.

This thesis proves that game engines can be used in transportation research. It can also be used in transportation planning, where planners can visualize their models to put forth their ideas. Unlike commercially available microsimulation software programs, a

game engine can provide great flexibility in terms of visualization and virtual environment interactions. This thesis could be further enhanced exponentially. For example, infectious disease models can be integrated with pedestrian microsimulations which therefore would help in visualizing the accurate outspread of diseases. More lanes and roads can be added to build 3D models of cities. Other kinds of traffic like buses, trains, and airplanes can be added to represent a diverse transport network. Simulators involving user participation like car driving simulators, cycling simulators, and walking simulators can be added to the current network to study the behavioural patterns of participants. The scope of enhancement for the framework this thesis presents is exponentially more when compared to commercially available software programs where users must rely on the development of the program or the addition of new modules. Such software programs can improve their flexibility by integrating a game engine style module which would allow users to manipulate various aspects of microsimulations such as the surrounding virtual environments or unique behavioural and visual characteristics of vehicles. These kinds of changes might also require the addition of physics engines that would bring the simulations much closer to the real world.

References

Ahmed, F., Zviedrite, N., & Uzicanin, A. (2018). Effectiveness of workplace social distancing measures in reducing influenza transmission: A systematic review. *BMC Public Health*, *18*(1), 1–13. https://doi.org/10.1186/s12889-018-5446-1

Atkins. (2016). *Research on the Impacts of Connected and Autonomouse Vehicles (CAVs) on Traffic Flow*. *March*, 58. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/530091/impacts-of-connected-and-autonomous-vehicles-on-traffic-flow-summary-report.pdf

Automotive revolution. (2016). *Automotive revolution – perspective towards 2030*. https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/disruptive-trends-that-will-transform-the-auto-industry/de-de#:~:text=Once technological and regulatory issues,be commercially available before 2020.

Bester, C. J., & Meyers, W. L. (2007). Saturation flow rates. *SATC 2007 - 26th Annual Southern African Transport Conference: The Challenges of Implementing Policy*, *July*, 560–568. https://doi.org/10.1061/9780784404645.ch07

Bouchnita, A., & Jebrane, A. (2020). A hybrid multi-scale model of COVID-19 transmission dynamics to assess the potential of non-pharmaceutical interventions. *Chaos, Solitons and Fractals*, *138*, 109941. https://doi.org/10.1016/j.chaos.2020.109941

Calderon, C., & Cavazza, M. (2001). Using Games Engines to implement Intelligent Virtual Environments. *Game-On*, *January*, 71.

CDC. (2021). *When You've Been Fully Vaccinated*. https://www.cdc.gov/coronavirus/2019-ncov/vaccines/fully-vaccinated.html

Chen, B., Sun, D., Zhou, J., Wong, W., & Ding, Z. (2020). A future intelligent traffic system with mixed autonomous vehicles and human-driven vehicles. *Information Sciences*, *529*, 59–72. https://doi.org/10.1016/j.ins.2020.02.009

Combs, T. S., Sandt, L. S., Clamann, M. P., & McDonald, N. C. (2019). Automated Vehicles and Pedestrian Safety: Exploring the Promise and Limits of Pedestrian Detection. *American Journal of Preventive Medicine*, *56*(1), 1–7. https://doi.org/10.1016/j.amepre.2018.06.024

Council, N. R., & TRB. (2000). Highway capacity manual. In *National Research Council, Washington, DC*.

Derjany, P., Namilae, S., Liu, D., & Srinivasan, A. (2020). Multiscale model for the optimal design of pedestrian queues to mitigate infectious disease spread. *PLoS ONE*, *15*(7 July), 1–21. https://doi.org/10.1371/journal.pone.0235891

Detwiller, M., & Gabler, H. C. (2017). Potential Reduction in Pedestrian Collisions With an Autonomous Vehicle. *Esv*, 1–8.

Doirado, E., Lint, H. Van, Prendinger, H., & Hoogendoorn, S. (2012). *Everscape : The Making of a Disaster Evacuation Experience*. 2285–2290.

Dommes, A., & Cavallo, V. (2009). *A Simulator-Based Street-Crossing Training for Older Pedestrians: Short and Long Term Effects*. 83–89. https://doi.org/10.17077/drivingassessment.1306

Dommes, A., & Cavallo, V. (2011). Can simulator-based training improve street-crossing safety for elderly pedestrians? *Transportation Research Part F: Traffic Psychology and Behaviour*, *15*(2), 206–218. https://doi.org/10.1016/j.trf.2011.12.004

Echeverría-Huarte, I., Garcimartín, A., Hidalgo, R. C., Martín-Gómez, C., & Zuriguel, I. (2021). Estimating density limits for walking pedestrians keeping a safe interpersonal distancing. *Scientific Reports*, *11*(1), 1–9. https://doi.org/10.1038/s41598-020-79454-0

Erath, A., Maheshwari, T., Joos, M., Kupferschmid, J., & van Eggermond, M. (2017). *Visualizing transport futures: The potential of integrating procedural 3d modelling and trafficmicro-simulation in virtual reality applicatio*. 12–19.

Fagnant, D. J., & Kockelman, K. (2015). Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, *77*, 167–181. https://doi.org/10.1016/j.tra.2015.04.003

Gamescrye. (2016). *What is a Game Engine?* https://gamescrye.com/blog/what-is-a-game-engine/

Gao, S., Rao, J., Kang, Y., Liang, Y., & Kruse, J. (2020). Mapping county-level mobility pattern changes in the United States in response to COVID-19. *ArXiv*, 16–26. https://doi.org/10.1145/3404820.3404824

Ghader, S., Zhao, J., Lee, M., Zhou, W., Zhao, G., & Zhang, L. (2020). Observed mobility behavior data reveal “social distancing inertia.” *ArXiv*, 1–12.

Ghaffarzadegan, N., & Rahmandad, H. (2020). Simulation-based estimation of the early spread of COVID-19 in Iran: actual versus confirmed cases. *System Dynamics Review*, *36*(1), 101–129. https://doi.org/10.1002/sdr.1655

Helbing, D., & Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, *51*(5), 4282–4286. https://doi.org/10.1103/PhysRevE.51.4282

Indraprastha, A., & Shinozaki, M. (2009). The Investigation on Using Unity3D Game Engine in Urban Design Study. *ITB Journal of Information and Communication Technology*, *3*(1), 1–18. https://doi.org/10.5614/itbj.ict.2009.3.1.1

Jorge, C. A. F., Sales, D. S., Couto, P. M., Botelho, F. M., & Felipe, R. (2007). *Virtual Environment Simulation As a Tool To Support*.

Kalra, N., & Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, *94*, 182–193. https://doi.org/10.1016/j.tra.2016.09.010

Lee, D. (2020). Autonomous Vehicle Implementation Predictions: Implications for Transport Planning. *Transportation Research Board Annual Meeting*, *42*(5 June 2020), 1–39.

Lopes, C. V., & Lindström, C. (2012). Virtual cities in Urban planning: The Uppsala case study. *Journal of Theoretical and Applied Electronic Commerce Research*, *7*(3), 88–100. https://doi.org/10.4067/S0718-18762012000300009

Maharaj, S., & Kleczkowski, A. (2012). Controlling epidemic spread by social distancing: do it well or not at all. *BMC Public Health*, *12*, 679. https://doi.org/10.1186/1471-2458-12-679

Matsui, Y., Hitosugi, M., Doi, T., Oikawa, S., Takahashi, K., & Ando, K. (2013). Features of Pedestrian Behavior in Car-to-Pedestrian Contact Situations in Near-Miss Incidents in Japan. *Traffic Injury Prevention*, *14*(SUPPL1). https://doi.org/10.1080/15389588.2013.796372

Matsui, Y., Hitosugi, M., Takahashi, K., & Doi, T. (2013). Situations of Car-to-Pedestrian Contact. *Traffic Injury Prevention*, *14*(1), 73–77. https://doi.org/10.1080/15389588.2012.678511

Miao, Q., Zhu, F., Lv, Y., Cheng, C., Chen, C., & Qiu, X. (2011). A game-engine-based platform for modeling and computing artificial transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, *12*(2), 343–353. https://doi.org/10.1109/TITS.2010.2103400

Michael Tlauka, P. N. W. (1996). *Orientation-Free Representations from Navigation through a Computer-Simulated Environment*.

Mike Houston, Chris Niederauer, Maneesh Agrawala, G. H. (2004). *VISUALIZING ARCHITECTURAL ENVIRONMENTS*.

Min W. Fong, Huizhi Gao, Jessica Y. Wong, Jingyi Xiao, Eunice Y.C. Shiu, Sukhyun Ryu, B. J. C. (2020). Nonpharmaceutical measures for pandemic influenza in nonhealthcare settings-international travel-related measures. *Emerging Infectious Diseases*, *26*(9), 2298–2299. https://doi.org/10.3201/eid2609.201990

Morando, M. M., Tian, Q., Truong, L. T., & Vu, H. L. (2018). Studying the Safety Impact of Autonomous Vehicles Using Simulation-Based Surrogate Safety Measures. *Journal of Advanced Transportation*, *2018*. https://doi.org/10.1155/2018/6135183

Moussaïd, M., Helbing, D., Garnier, S., Johansson, A., Combe, M., & Theraulaz, G. (2009). Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proceedings of the Royal Society B: Biological Sciences*, *276*(1668), 2755–2762. https://doi.org/10.1098/rspb.2009.0405

Muhammad Ahsanul Habib, Jahedul Alam, D. H. (n.d.). *TRB Annual Meeting Pedestrian Micro-simulation for Evaluating the Impacts of Social Distancing Regulations on a Dense Urban Street in Halifax , Canada MD Jahedul Alam , PhD Candidate*.

Muhammad Ahsanul Habib, D. H. (2019). *Virtual Reality For Evaluating Active Transportation Improvements For Roadways*. *264*, 1–18.

National Highway Traffic Safety Administration. (2015). *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*.

Newbold, S. C., Finnoff, D., Thunström, L., Ashworth, M., & Shogren, J. F. (2020). Effects of Physical Distancing to Control COVID-19 on Public Health, the Economy, and the Environment. *Environmental and Resource Economics*, *76*(4), 705–729. https://doi.org/10.1007/s10640-020-00440-1

PTV. (2018). PTV Vissim 11 User Manual. *Ptv Ag*, 1278.

Ribeiro, J., Almeida, J. E., Rossetti, R. J. F., Coelho, A., & Coelho, A. L. (2009). *Towards a serious games evacuation simulator*.

Ronan Glon, & Stephen Edelstein. (2020). History of Self-Driving Cars. *Digital Trends*. https://www.digitaltrends.com/cars/history-of-self-driving-cars-milestones/

Rossetti, R. J. F., Almeida, J. E., Kokkinogenis, Z., & Goncalves, J. (2013). Playing transportation seriously: Applications of serious games to artificial transportation systems. *IEEE Intelligent Systems*, *28*(4), 107–113. https://doi.org/10.1109/MIS.2013.113

Santana, E. F. Z., Covas, G., Duarte, F., Santi, P., Ratti, C., & Kon, F. (2021). Transitioning to a driverless city: Evaluating a hybrid system for autonomous and non-autonomous vehicles. *Simulation Modelling Practice and Theory*, *107*(May 2020), 102210. https://doi.org/10.1016/j.simpat.2020.102210

Stanford. (2020). *Work and a desire to exercise, socialize are why people didn't social distance, Stanford researchers find*. https://news.stanford.edu/2020/04/14/people-didnt-social-distance/

Sticco, I. M., Frank, G. A., & Dorso, C. O. (2020). Social Force Model parameter testing and optimization using a high stress real-life situation. *ArXiv*.

Wang, Q., & Abbas, M. (2018). Designing web-games for transportation engineering education. *Computer Applications in Engineering Education*, *26*(5), 1699–1710. https://doi.org/10.1002/cae.22031

Warren, M. S., & Skillman, S. W. (2020). Mobility Changes in Response to COVID-19. *ArXiv*.

Waymo. (2019). *Seeing the Road Ahead*. https://waymo.com/journey/

WHO. (2021). *COVID-19: Physical Distancing*. https://www.who.int/westernpacific/emergencies/covid-19/information/physical-distancing

WHO APA. (2021). *Weekly Operational Update on COVID-19*. *November*, 1–10. https://www.who.int/publications/m/item/weekly-update-on-covid-19---16-october-2020

Xiao, Y., Yang, M., Zhu, Z., Yang, H., Zhang, L., & Ghader, S. (2020). Modeling indoor-level non-pharmaceutical interventions during the COVID-19 pandemic: A pedestrian dynamics-based microscopic simulation approach. *ArXiv*, 1–24.

Ye, L., & Yamamoto, T. (2019). Evaluating the impact of connected and autonomous vehicles on traffic safety. *Physica A: Statistical Mechanics and Its Applications*, *526*, 121009. https://doi.org/10.1016/j.physa.2019.04.245

Highway Capacity Manual 2010. (2010). Transportation Research Board of the National Academies.

Appendix

A: Sample Scripts

1. Sample Code for Spawning Pedestrians

```
public class PedestrianSpawner : MonoBehaviour
{
   public GameObject pedesstrianPrefab;
   public int pedestriansToSpawn;
   public List<GameObject> pedestrians;

   void Start()
   {
      StartCoroutine(Spawn());
   }

   IEnumerator Spawn()
   {
      int count = 0;
      while(count < pedestriansToSpawn)
      {
         GameObject obj = Instantiate(pedesstrianPrefab);
               Transform child = transform.GetChild(UnityEngine.Random.Range(0,
               transform.childCount - 1));
         obj.GetComponent<WaypointNavigator>().currentWaypoint =
               child.GetComponent<Waypoint>();
         obj.transform.position = child.position;
         yield return new WaitForEndOfFrame();
         count++;
      }
   }

}
```

2. Sample Code for Recording Saturation Flow Rate

```
currentTime = Time.time.ToString("f6");
RaycastHit hit;
Vector3 stpt1 = transform.position;
stpt1 += transform.forward * fsrp1.z;
stpt1 += transform.up * fsrp1.y;
stpt1 += transform.right * fssrp1;

if (Physics.Raycast(stpt1, Quaternion.AngleAxis(180, transform.up)
* transform.forward, out hit, 4f))
        {
          if (hit.collider.CompareTag("CB1"))
          {
            if (hit.transform.name != nm)
            {
              Debug.DrawLine(stpt1, hit.point, Color.green);
              AddRecord(currentTime, hit.transform.name, "Sensor1",
              "Saturation_Flow.csv");
              nm = hit.transform.name;
            }
          }
        }
    }

    public static void AddRecord(string tt, string name, string cat, string
        filepath)
    {
      try
      {
              using (System.IO.StreamWriter file = new
              System.IO.StreamWriter(@filepath, true))
               {
                      file.WriteLine(tt + "," + name + "," + cat);
               }
      }
      catch (Exception ex)
      {
        throw new ApplicationException("Something went wrong :", ex);
      }
    }
```

3. Sample Code for Signal System

```
if (t == 8)
        {
            Sig1.gameObject.SetActive(true);
            Sig2.gameObject.SetActive(true);
        }
if (t == 12) //13s
        {
            R1.SetActive(false);
            R2.SetActive(false);
            G1.SetActive(true);
            G2.SetActive(true);
            cSig1.gameObject.SetActive(false);
            cSig2.gameObject.SetActive(false);

        }
if (t == 25) //3s
        {
            G1.SetActive(false);
            G2.SetActive(false);
            Y1.SetActive(true);
            Y2.SetActive(true);
            cSig1.gameObject.SetActive(true);
            cSig2.gameObject.SetActive(true);
        }
if (t == 28) //2s
        {

            Y1.SetActive(false);
            Y2.SetActive(false);
            R1.SetActive(true);
            R2.SetActive(true);
        }
if (t == 30)
        {
            rectime += n;
        }
if (t == 0)
        {
            Sig1.gameObject.SetActive(false);
            Sig2.gameObject.SetActive(false);
        }
    }
}
```

4. Sample Code for Recording Average Speed and Time Travelled by a Vehicle

```
private void Awake()
    {
        a = (float)Time.time;
    }

void Update()
    {
        speed = (float)rb.velocity.magnitude;
        speeds.Add(speed);
        t = (float)Time.time - a;
    }

void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Finish"))
        {
            float averageTotal;
            for (int i = 0; i < speeds.Count; i++)
            {
                finalTotal += speeds[i];
            }
            averageTotal = finalTotal / speeds.Count;
            AddRecord(this.name, t, averageTotal, "SpeedTime.csv");
        }
    }
public static void AddRecord(string cat, float tt, float avgsp, string   filepath)
    {
        try
        {
using (System.IO.StreamWriter file = new System.IO.StreamWriter(@filepath, true))
            {
                file.WriteLine(cat + "," + tt + "," + avgsp);
            }
        }
        catch (Exception ex)
        {
            throw new ApplicationException("Something went wrong :", ex);
        }
    }
}
```

5. Sample Code for Vehicle Movement

```
private void Drive()
   {
      currentSpeed = (float)rb.velocity.magnitude;
      bx = (bxadd + (bxmul * z)) * (Mathf.Sqrt(currentSpeed));
      D = ax + bx + ad;
      if (refval > D)
      {
               sov = (val * ((refval - D) / (16 - refval))) / (((refval - D) /
               (16 - refval)) + 1 );
      }
      else
      {
         sov = 0;
      }
      maxSpeed = msv + sov;
      if (!avoiding)
      {
         if (fol != 0f)
         {
            if (D <= fol)
            {
               u = (float)rb.velocity.magnitude;
               a = (maxSpeed - u) / tm;
               Force = 1000f * a;
               Torque = 0.3f * Force;
               wheelFL.motorTorque = Torque;
               wheelFR.motorTorque = Torque;
               wheelFL.brakeTorque = 0;
               wheelFR.brakeTorque = 0;
            }
            else
            {
               a = 0;
               Force = 1000f * a;
               Torque = 0.3f * Force;
               wheelFL.motorTorque = Torque;
               wheelFR.motorTorque = Torque;
               wheelFL.brakeTorque = maxBrakeTorque;
               wheelFR.brakeTorque = maxBrakeTorque;
            }
         }
```

6. Sample Code for Vehicle Path

```
void OnDrawGizmosSelected()
  {
    Gizmos.color = Color.yellow;

    Transform[] path = GetComponentsInChildren<Transform>();
    nd = new List<Transform>();

    for(int i = 0; i < path.Length; i++)
          {
      if(path[i] != transform)
                {
        nd.Add(path[i]);
        }
      }

    for(int i = 0; i < nd.Count; i++) {
      Vector3 cn = nd[i].position;
      Vector3 pn = Vector3.zero;

      if (i > 0)
                {
        pn = nd[i - 1].position;
        }
         else if(i == 0 && nd.Count > 1)
            {
        pn = nd[nd.Count - 1].position;
        }

      Gizmos.DrawLine(pn, cn);
      Gizmos.DrawWireSphere(cn, 0.3f);
    }
  }
```

B: Sample Data for COVID-19 Application

1. Travel Time Data for BAU scenario

Pedestrian Name	Travel Time (s)
SFC2_1	5.618999
SFC1_1	5.607123
SFC1_2	5.618898
SFC2_2	5.638484
SFC1_3	5.63641
SFC2_3	5.638397
SFC1_4	5.612287
SFC2_4	5.620705
SFC1_5	5.612818
SFC1_6	5.58828
SFC2_5	5.659752
SFC1_7	5.617393
SFC2_6	5.626469
SFC1_8	5.609224
SFC1_9	5.61919
SFC2_7	5.627308
SFC1_10	5.615734
SFC2_8	5.616875
SFC1_11	5.570717
SFC2_9	5.637741
SFC1_12	5.616055
SFC1_13	5.591923
SFC2_10	5.616653
SFC1_14	5.636787
SFC2_11	5.635143
SFC1_15	5.586124
SFC1_16	5.614525
SFC2_12	5.630562
SFC1_17	5.59761
SFC2_13	5.623306
SFC1_18	5.593079
SFC2_14	5.627975
SFC1_19	5.598358
SFC1_20	5.618698
SFC2_15	5.609322

2. Pedestrian Density Data for BAU Scenario

Pedestrian Density (p/sqm)	Time (s)
0.0639713	0
0.0639713	1
0.0639713	2
0.095957	3
0.095957	4
0.1279427	5
0.1279427	6
0.1599284	7
0.1599284	8
0.1599284	9
0.2238997	10
0.191914	11
0.1599284	12
0.1599284	13
0.191914	14
0.191914	15
0.1599284	16
0.1599284	17
0.1599284	18
0.1599284	19
0.191914	20
0.191914	21
0.1599284	22
0.1599284	23
0.1599284	24
0.191914	25
0.1599284	26
0.1599284	27
0.191914	28
0.1599284	29
0.191914	30
0.1599284	31
0.191914	32
0.1599284	33
0.1599284	34
0.191914	35
0.1599284	36

3. Violation Notes for BAU Scenario

Time of Violation (s)	Pedestrian Name 1	Pedestrian Name 2
5.54	SFC2_1	SFC1_1
5.56	SFC1_1	SFC2_1
7.28	SFC2_1	SFC1_2
7.3	SFC1_2	SFC2_1
8	SFC1_1	SFC2_2
8.02	SFC2_2	SFC1_1
9.08	SFC2_1	SFC1_3
9.12	SFC1_3	SFC2_1
9.7	SFC1_2	SFC2_2
9.719999	SFC2_2	SFC1_2
10.52	SFC1_1	SFC2_3
10.56	SFC2_3	SFC1_1
10.84	SFC2_1	SFC1_4
10.88	SFC1_4	SFC2_1
11.5	SFC1_3	SFC2_2
11.5	SFC2_2	SFC1_3
12.56	SFC2_3	SFC1_2
13.32	SFC2_2	SFC1_4
13.34	SFC1_4	SFC2_2
14	SFC1_3	SFC2_3
14.02	SFC2_3	SFC1_3
15.16	SFC2_2	SFC1_5
15.2	SFC1_5	SFC2_2
15.8	SFC1_4	SFC2_3
15.82	SFC2_3	SFC1_4
16.54	SFC1_3	SFC2_4
16.58	SFC2_4	SFC1_3
17.84	SFC1_5	SFC2_3
17.86	SFC2_3	SFC1_5
18.32	SFC1_4	SFC2_4
18.34	SFC2_4	SFC1_4
19.44	SFC2_3	SFC1_6
19.48	SFC1_6	SFC2_3
20.12	SFC1_5	SFC2_4
20.14	SFC2_4	SFC1_5
20.86	SFC1_4	SFC2_5
20.9	SFC2_5	SFC1_4

4. Saturation Flow Rate Sensor

Time of Detection (s)	Vehicle Name	Sensor Name
44.08	Tocus (2)_1	Sensor2
44.1	Tocus (1)_1	Sensor1
46.3	Tocus (2)_2	Sensor2
46.44	Tocus (1)_2	Sensor1
48.74	Tocus (2)_3	Sensor2
48.84	Tocus (1)_3	Sensor1
51.06	Tocus (1)_4	Sensor1
51.1	Tocus (2)_4	Sensor2
73.86	Tocus (2)_5	Sensor2
74.06	Tocus (1)_5	Sensor1
76.2	Tocus (2)_6	Sensor2
76.38	Tocus (1)_6	Sensor1
78.6	Tocus (2)_7	Sensor2
78.66	Tocus (1)_7	Sensor1
81.02	Tocus (2)_8	Sensor2
81.1	Tocus (1)_8	Sensor1
85.84	Tocus (2)_9	Sensor2
104.04	Tocus (1)_9	Sensor1
104.18	Tocus (2)_10	Sensor2
106.44	Tocus (1)_10	Sensor1
106.6	Tocus (2)_11	Sensor2
108.68	Tocus (1)_11	Sensor1
108.94	Tocus (2)_12	Sensor2
111.04	Tocus (1)_12	Sensor1
113.76	Tocus (2)_13	Sensor2
133.74	Tocus (1)_13	Sensor1
134.16	Tocus (2)_14	Sensor2
136.06	Tocus (1)_14	Sensor1
136.36	Tocus (2)_15	Sensor2
138.52	Tocus (1)_15	Sensor1
138.7	Tocus (2)_16	Sensor2
140.92	Tocus (1)_16	Sensor1
144.72	Tocus (2)_17	Sensor2
145.08	Tocus (1)_17	Sensor1
164.02	Tocus (1)_18	Sensor1
164.14	Tocus (2)_18	Sensor2