Virtual Cycling Environment

Zack Denieffe

Final-Year Project– BSc in Computer Science

Supervisor: Dr Aisling O'Driscoll

Department of Computer Science,

University College Cork

April 2024

ABSTRACT

The primary objective of the project was to build a virtual cycling environment using open source components (Unity, SUMO, sensors, V2X comms) to learn more about vulnerable road user (VRU) behaviour. The secondary goal being to explore the use of haptic signals to deliver safety related warnings to VRUs if time allowed. Simulating human and traffic interactions in a virtual environment allows the human response to mixed traffic conditions to be assessed. The work of the project succeeded in integrating multiple sections of the virtual environment, but efforts to synchronise these are ongoing. This project revolves around a few systems that you need to learn well to use effectively. Knowing how to work efficiently in Unity, SUMO and CityEngine as well as good knowledge of how to code in C# and the sensors used are vital. This was achieved in the current project.

Declaration of Originality

In signing this declaration, you are conforming, in writing, that the submitted work is entirely your own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award.

I hereby declare that:

- this is all my own work, unless clearly indicated otherwise, with full and proper accreditation;
- with respect to my own work: none of it has been submitted at any educational institution contributing in any way to an educational award;
- with respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

Signed:. . . . . Zack Denieffe . . . . . .

Date: . . . . . 24/04/24. . . . . .

ACKNOWLEDGEMENTS

Despite the profound struggle, I've gotten over the line. Thank you to everyone that made these last four years any bit easier.

and to my biggest supporter

Thanks Mammy

Table of Contents

Abstract 2
Declaration of Originality 3
Acknowledgements 4
Table of Contents 5
1.0 Introduction 6
2.0 Analysis 7
    Bike Setup 8
    Sensors 9
    Unity 10
    SUMO 11
    OSM 12
3.0 Design & Implementation 13
    Modelling the Urban Environment 13
        SUMO 14
        TRACI 15
        ArcGIS CityEngine 15
    Bike Model and Data Streams 16
        Speed Data Input 17
        Steering 18
        Syncing all Data Input 18
4.0 Evaluation 19
5.0 Conclusions 21
6.0 References 22

1.0 Introduction

Vulnerable road users (VRUs) represent a growing number of individuals on Irish road-ways. As this contingent grows so too does the need to protect them, research published by the Road Safety Authority of Ireland (RSA) shows a worrying increase in people being injured while cycling. The research shows that between 2016-2021 there were 1,435 serious injuries recorded of people cycling. For each cycling fatality there were 25 serious injuries. Failure to observe by other drivers was the most frequently noted cause for collisions involving people who cycle.[1] While the global share of fatalities has fallen 1% in 4-wheel vehicles and 2% among two and three-wheeled vehicles since 2010, it has risen from 5% to 6% among cyclists. A rise of 20%. Road traffic injuries remain the leading killer of children and young people aged 5-29. More than half of fatalities occur among pedestrians, cyclists and motorcyclists, all vulnerable road users[2]

Most of the public are aware of campaigns to increase safety on our roadways with specific cycling safety campaigns, yet driving and traffic simulation focus tends towards vehicle dynamics and especially cars. Simulating human and traffic interactions in a virtual environment allows the human response to mixed traffic conditions to be assessed. The goal of the current work was to build a virtual cycling environment using open source components (Unity, SUMO, sensors, V2X comms) to learn more about vulnerable road user (VRU) behaviour. The project was conducted by creating a virtual environment connected to traffic simulation software, input was fed via a real world bike setup to a bike model within the 3D environment. Speed and gyro sensors were used to gather data for the simulation.

2.0 Analysis

The primary objective of the current work was to build a virtual cycling environment using open source components (Unity, SUMO, sensors, V2X comms) to learn more about vulnerable road user (VRU) behaviour. The secondary goal being to explore the use of haptic signals to deliver safety related warnings to VRUs if time allowed.

It was necessary to create a 3D environment which would allow a simulated bike, taking input from sensors placed on the real world bike, to move around a simulated urban environment. The bike model should have physics and mechanics akin to real world cycling, with the urban environment connected to a traffic simulator.

Before discussing the components which could be utilised in this project, a brief overview of previous relevant work is provided:

The current work builds on investigations done over the last few years into the viability of virtual bike environments and hardware in the loop approaches to human traffic interaction analysis. It was also possible through the work of former students of UCC who did work on SUMO-Unity connection using TRACI.

Some of the foremost research has been done by the University of Paderborn in conjunction with TU Berlin to test the viability of and then build a Virtual Cycling environment with haptic and visual feedback. In their project they utilised a phone app for gyro data/steering and an IR sensor for speed[3]

A past student of UCC who will be referred to purely by their GitHub handle, DarraghMac, completed work involving Unity, SUMO and connecting the two with TRACI. All elements which I will discuss further.[4]

The components to be discussed are;

- Bike setup
- Sensors
- Unity
- Sumo

Bike Setup

To adopt the idea of a Hardware-in-the-loop (HIL) or Human-in-the-loop (HITL) solution a bike setup was needed that would be used to interact with the virtual environment. HIL is when a physical system, or hardware, is connected to a simulated environment, so in our case the bike setup and all associated sensors mounted on it are connected directly to our virtual environment within unity and influencing the control of the bike model. HITL is somewhat similar in that a user actively participates in the control or decision-making process alongside automated components, in this case traffic simulation.

The bike setup consisted of an everyday city bike mounted on a bicycle trainer and a front wheel riser block.

A bike trainer is a piece of equipment that allows a bike to be ridden indoors while it remains stationary i.e. allows the back wheel of the bike to turn freely. For the purposes of this project a basic model trainer, the Halfords Turbo Trainer was used. It has simple capabilities, allowing for the user to adapt the resistance on the back tyre, not especially useful in our case. Apart from not offering a wide fit and limiting the range of bikes usable, it was perfect. This model was chosen over more high-end bike trainers due to the obvious budget discrepancy and its simplicity. Often higher end trainers come ready to use with designated software and the goal of this project was to implement a virtual cycling environment using open source components.

A front wheel riser block is an essential piece of equipment for indoor cycling. It is typically a simple mount for the front wheel of the bike while the back wheel is contained within the bike trainer of choice. For the purposes of our project we needed a riser block that was capable of swiveling horizontally. The block chosen was the LifeLine Indoor Cycling Swivel Riser Block that as the name might imply swiveled allowing for unimpeded motion of the front column.

Sensors

In order to allow for movement and steering of the bike in the 3D environment, data needed to be transferred from the real world bike to the bike model in the virtual environment. Speed data and steering input was needed.

The sensors used in this project were a Wahoo RPM Speed Sensor and an iPhone 11's internal gyroscope for steering.

Speed Sensor

Figure 1: Wahoo RPM Speed Sensor

The Wahoo RPM Speed Sensor has dual band capabilities, able to connect via ANT+ or Bluetooth Smart technologies. Bluetooth devices have names they are identifiable as, services available on them which are the default services e.g. battery and so on as well as the services unique to the device, in our case the speed and cadence service. Each service has a number of characteristics available to it. Characteristics in this case like total revolutions, last wheel event, crank revolutions etc.

Services, Characteristics and Descriptors are all types of attributes associated with Bluetooth Low Energy(BLE) devices. Universally unique identifiers (UUIDs) are 128-bit numbers used to uniquely identify services, characteristics, and descriptors in Bluetooth Low Energy (BLE) devices. They are typically represented in hexadecimal format. Services represent different functionalities or features offered by a Bluetooth device. Each service has a unique UUID. Examples of services include the Generic Access Profile (GAP), Generic Attribute Profile (GATT), Battery Service and Speed and Cadence Service. Characteristics are data points within a service that represent specific attributes or data. Each characteristic has a unique UUID within the scope of its service. Characteristics can be read, written to, or subscribed to for notifications. Descriptors are characteristic metadata, usually providing some extra info on individual characteristics, all characteristics are well documented online.

Steering Sensor

All modern IOS and android devices tend to have gyroscopes and accelerometers within them. The main concern when utilising a phone as an input device for Unity is the lack of support for such. You can build directly for IOS, but many solutions researched tended towards establishing a client-server relationship of sorts, we will discuss this further later in the report.

Unity

Figure 2: Unity Technologies Logo

Unity is a cross-platform game engine developed by Unity technologies and it can be used to make video games, simulations and many other kinds of interactive experiences. Unity provides a user friendly interface that allows users to interact with its extensive set of features and tools to create. Tools such as graphics rendering, physics simulation, scripting and more. Scripting in Unity utilities C#. Another very useful aspect to unity is the existence of the unity asset store that makes available a wide range of premade models, scripts, tools and plugins. Unity is a platform with a lot of usage world wide which has resulted in the existence of a large bank of material in the form of tutorials and answers to queries concerning many aspects of Unity development and usage of the platform in general.

Within Unity, the developer can create scenes which contain all of the objects and components that make up any one level of a game or environment associated with a simulation. In the case of this project, only one scene is needed containing the primary simulation. Further scenes could be considered upon completion such as menus, different traffic scenarios and more.

Cameras in unity capture and render scenes from a set point of view. Cameras can be set at a specific point in space or attached to objects such as the bike model for this project

Materials in Unity are assets that determine how an object is rendered visually. It utilises properties such as color, texture, transparency and reflectivity.

Shaders run on the system's GPU and determine how the pixels of an object are rendered. Shaders control the shading, lighting, and surface properties of objects.

Objects serve as containers for components such as scripts, transforms, colliders and shaders. They represent entities within your game world. Within this project objects exist in the for of the bike model, urban environment and autonomous cars moving throughout.

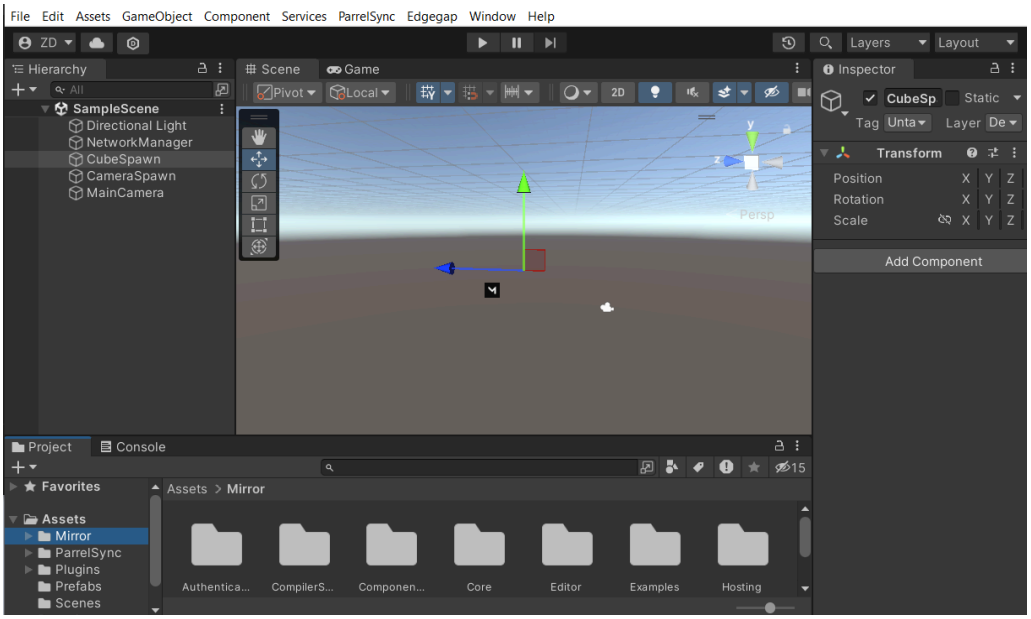

Figure 3: Unity Editor Sample

SUMO

"Simulation of Urban MObility" (SUMO) is an open source, highly portable, microscopic and continuous traffic simulation package which was used in this project to replicate traffic in an urban environment. SUMO is developed by the German Aerospace Centre(DLR) for use in primarily simulation of traffic in u urban environments. However it can also be used to simulate traffic in highways and other road networks. SUMO was intended to be used by academics/researchers, urban planners and transport engineers to understand traffic flow, test designs and evaluate traffic soothing and management techniques.

SUMO allows an individual to plot routes on a map of real world data. Using a .OSM file

OSM

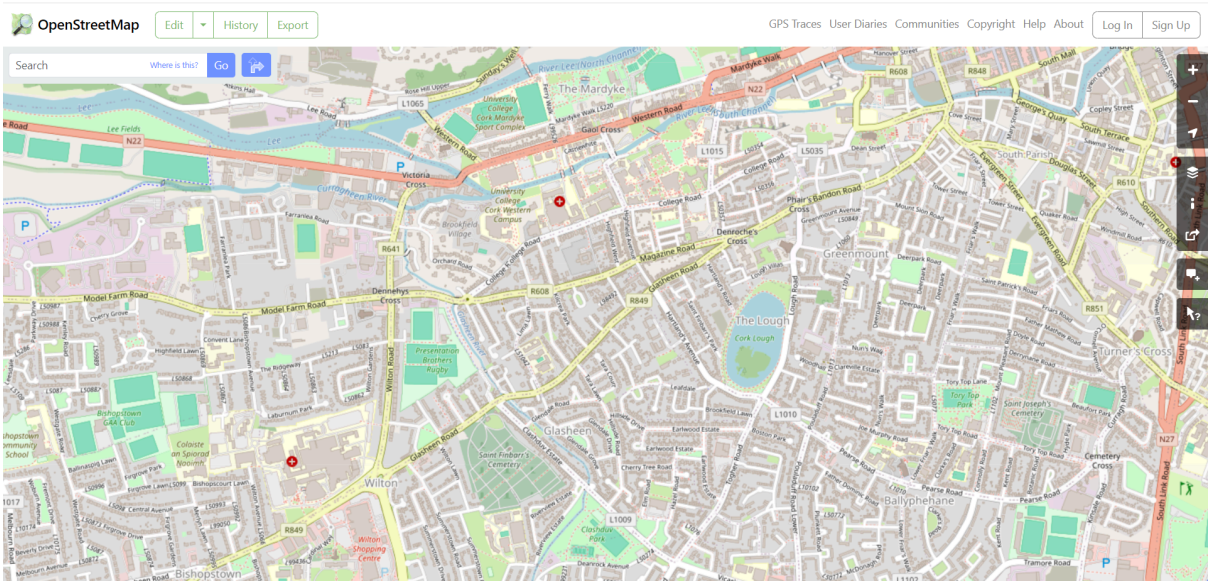

Figure 4: OpenStreetMap.org view of Cork

Utilising OpenStreetMap(OSM) files within SUMO you can create networks with cars travelling routes planned out on real world map data. "OpenStreetMap is built by a community of mappers that contribute and maintain data about roads, trails, cafés, railway stations, and much more, all over the world."[5] One can simply access the OSM website and export an area as a **.**OSM file, The OSM file contains all relevant geographical data, such as location, usage, elevation etc.

There are a range of commands and edit options available to clean up OSM files for use in traffic simulation. Java OpenStreetMap editor (JOSM) is the most common option for editing. A user can use the clear UI to interact with and edit components whether it is for contribution to the OSM community or for specific uses like this project.

ArcGIS CityEngine

"ArcGIS CityEngine is a commercial three-dimensional modeling program developed by Esri R&D Center Zurich and specialises in the generation of 3D urban environments"[6]. Using CityEngine, one is able to import a .OSM file and generate a model of the map with the click of a button. It outputs as a FilmBox (.FBX) file which can be transferred to Unity. The .FBX retains the geographical data present in the original .OSM file. When used in Unity the buildings now exist as objects, the

associated materials have been drawn from satellite imagery and are all available as well. Meaning your virtual environment should look similar to its real world inspiration.

3.0 Design & Implementation

Figure 5 shows a high-level thumbnail sketch of overall design

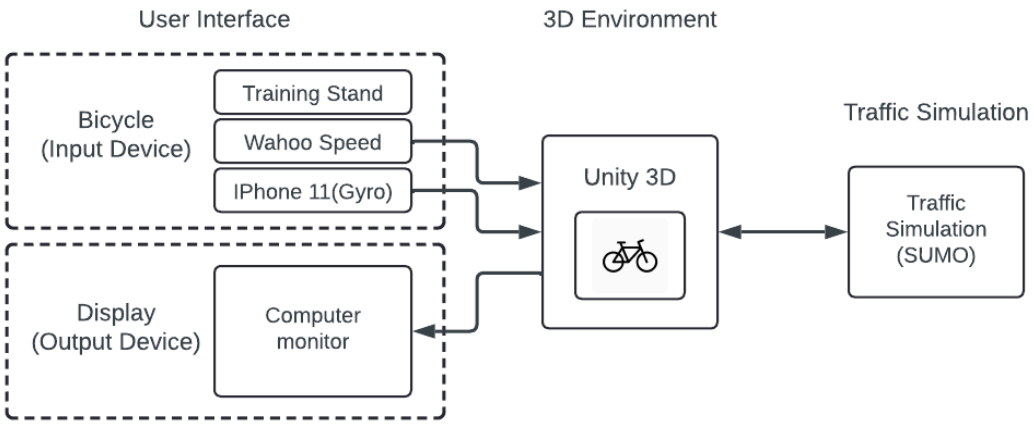

Figure 5: Overall Design

Modelling the Urban Environment

Design: A simulation of Cork city or more specifically a subsection of Cork city surrounding The Lough. Model a simple traffic network on top of it. Turn the same OSM file into an object via CityEngine for use in Unity. Finally, link SUMO and Unity using TRACI.

Implementation:

While obtaining Cork city map data from openstreetmap.org it was clear that simulating and modeling Cork would be more of a challenge than was initially thought. OSM being community focused and maintained is great for many reasons, however Cork is not well defined in a few aspects.

SUMO

First step was creating a simulation using Cork map data within SUMO. Using a simple editor (JOSM) you are able to remove difficult features and configure junctions of different kinds for ease of simulation. A low number of routes were placed in the simulation to keep the project realistic, the project being more about the behaviour of the user around the vehicles rather than a pure traffic analysis. The “messiness”/disorder of Corks map data crops up here, in figure 7 you can see how each and every driveway gets configured as a road, an issue that will feature in a future section.

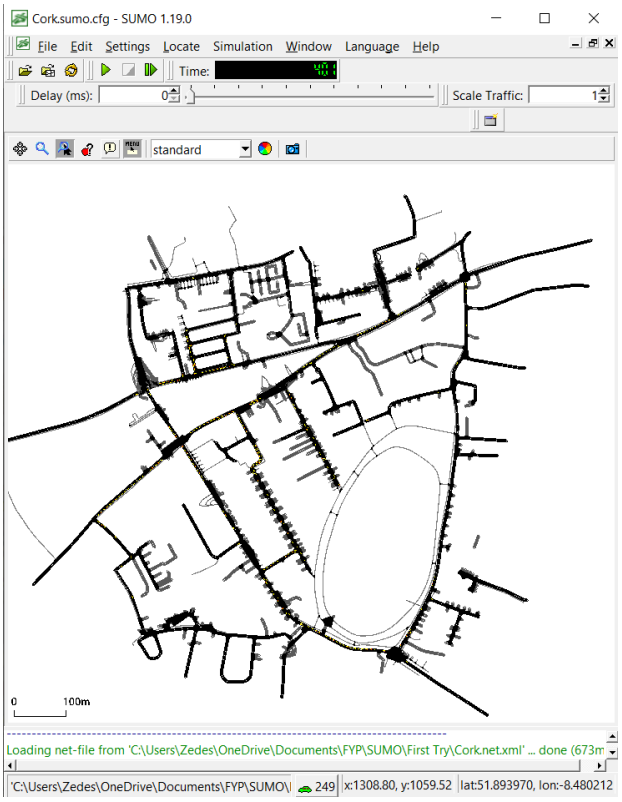

Figure 6: SUMO Editor view of Cork Simulation

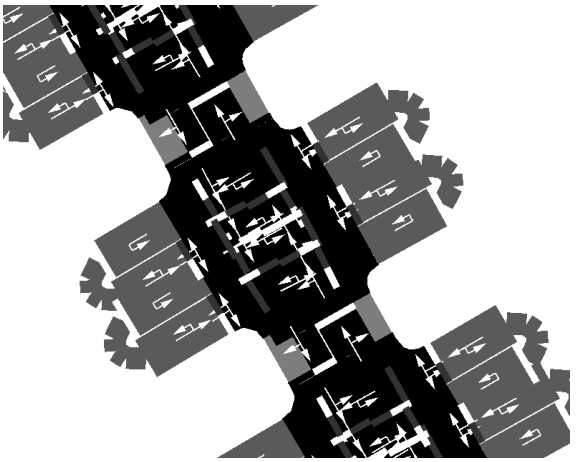

Figure 7: Cork Simulation driveway disorder

TRACI

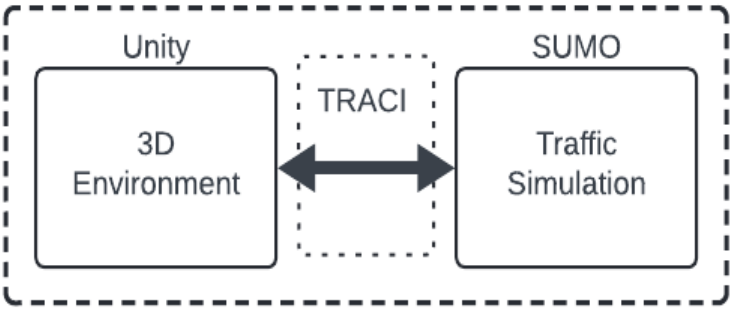

Figure 8: TRACI connection outline

TRACI (Traffic Control Interface) facilitates communication between Unity and SUMO. TRACI allows integration of traffic simulations generated in SUMO into Unity-based virtual environments. TRACI allows for the flow of data in both directions, meaning the bike model is tracked in the SUMO simulation and the movement of the traffic is replicated in both components.

ArcGIS CityEngine

When importing the map and generating the Unity usable model in CityEngine the issues with Cork arose again. The problem was multifaceted, issues with realism and the actual modelling presenting. Cork is known for its hills and tight streets and that wasn't being properly reflected in the model. Junctions and service roads were a big problem, when generated in CityEngine we ended up with many out of place and/or completely missing blocks of road/path/infrastructure. Curbs on curbs in places and holes in the object in others, generally unusable.

CityEngine was part of the reason that I ultimately opted for the Manhattan model created by a past student (Darraghmac). For the sake of momentum and further development of the project, his project was adapted for use with a bike.

Bike Model and Data Streams

Design: The bike model needed to simulate the real world mechanics and physics of a bike. This included ideas like the angle of steering possible, the lean, the look and appeal, the rotation of the wheels and the point of view of the user as a would-be cyclist.

Implementation: The following sections outline the implementation approach

Bike Model

After contemplating the design of a bike model, I found a model freely available from the Unity asset store. The bike model differed from others available, many tend towards a more game friendly feel and some towards cycling race simulators like zwift. The mechanics of both didn't fit the requirements. The bike by Razyn[7] happened to be very simply coded and easily adaptable. Changes could be made to the bike model aesthetic by altering the colliders and shaders used. Sliders and simple input fields allowed developers to change components like motor force and steering angles, leaning too. For the purposes of the project I limited the leaning and steering angles to not only hopefully resemble real world physics but also make the user experience not as jarring when the player camera is positioned atop the bike as opposed to a wider third person point of view.

Figure 9: First Person View of handlebars

Figure 10: Razyn bike model side view

Speed Data Input

The speed of the bike within the virtual environment was the next step. The bike model needed to move at a reasonable speed in relation to the model of the urban environment as well as in accordance with the rate at which the user was turning the pedals of the real world bike. Our sensor, the Wahoo RPM Speed sensor works using Bluetooth Smart or ANT+. The design centered around using bluetooth smart technology. It was mounted directly on the axle of the back wheel, as close to the centre of the wheel as possible.

Bluetooth Connection Program

Utilising a range of tutorials and examples available, a project was put together capable of connecting to the sensor and pulling data off of it written in C#.

By scanning, subscribing to a specified service and characteristic one can read from a bluetooth sensor, getting data in byte format, with each index corresponding to a characteristic. In this case, the program searches for the specified sensor, matches service and characteristic UUID and then subscribes for continued data transfer. The Wahoo Speed Sensor's most useful field was the total revolutions. The speed data was calculated using a script with the total revolution change in accordance with the time.

Implementing Speed Data Transfer in Unity Project(explain.dll)

To integrate the bluetooth connection into Unity I had to utilise the .dll file that was built from a solution within Visual Studio. Using the .dll and wrapping lines of my script in Unity in .dll calls, a simple bluetooth API was written. A script controlling the bike model would then be able to interact with this API to access current speed data.

The project did not progress to the stage of full functionality in this regard. Utilising the speed data in Unity is ongoing. Further details are provided in the section: Evaluation.

Steering

The steering of the bike model in Unity relied upon an iPhones' gyroscope input. The final design resulted in the iPhone being placed in a phone mount on the handlebars and connected to the system hosting the project via a simple USB cable.

The initial idea of building a steering app was the ultimate goal but utilising Unity remote during development proved helpful. Unity Remote is an app usable on mobile devices, it allows quick deployment of a Unity project to a phone to test. However while choosing to develop the project in this fashion, the issue of syncing multiple streams of data arose. How would one go about getting speed data to a Unity instance that is essentially running from a phone.

Syncing all Data Input

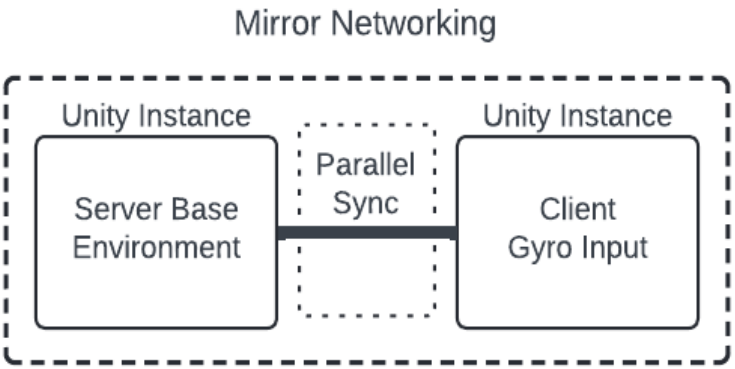

Figure 11: Mirror Networking Outline

As development continued, the challenge of syncing speed and steering data in one Unity instance became apparent. Unity does not offer many solutions for using a phone as a controller for a desktop instance. This led to Mirror Networking[8] and Parallel Sync design. Both being freely available from the asset store. Mirror Networking is an intended solution to adding multiplayer to games built in Unity. It adds networking infrastructure and simplifies the process of adding multiplayer functionality to Unity projects by providing an API built on top of pre-existing high level Unity functionality with regard to networking.

The solution involves cloning an instance of the project, with changes maintained across the two using parallel sync. Mirror then provides a library with a wealth of pre-written scripts and assets to avail of. Network manager scripts can prove useful, but for projects like this which tend to deviate from the default usage of the software, a custom network manager was

developed. Assigning the network manager script to an object within the unity scene allows you to utilise it. Ready made scripts for network transforms and player objects are also available, these allow the framework to sync object movement across the instances.

The attempt at the synchronization of all the input streams used the first instance of the Unity project as the server and the second as the client. The server would connect to the environment and control the speed input, the client would run via Unity Remote on the IOS device and connect to an obsolete camera or empty object within the scene that either wouldn't render or would not be seen. A script on the bike model that controls all movement then allows for a call to the client object to make use of the steering data.

4.0 Evaluation

Ultimately the completed work can be surmised as

- The BLE connection program and work to integrate within Unity
- Steering in Unity using a Bike model
- Cork Unity-SUMO integration, Adapted Manhattan Model for use
- Real world bike setup

The connection established with the bluetooth sensor makes use of extensive work done on learning how BLE devices work, how the specific sensor data is stored and how to get that consistently across to the system running the program. A weakness of the sensor used is the update rate, the Wahoo RPM Speed and Cadence Sensor delivers updates once a second (1Hz). This is just not fast enough for use in a system that attempts to model real world conditions. During testing the program consistently reads 0 kmph, although the wheel continues to rotate the sensor is delayed in conveying that. The reading can fluctuate from 0 kmph to unrealistically high and inaccurate speeds as the cumulative change in total rotations has been pushed to the next update.

Further work into the use of speed sensors for related projects should include higher specification sensors. IR sensors have been used as well as projects developed using VR components and OpenXR within Unity.

Steering was initially considered to be straightforward, Unity makes accessing the gyroscope data and isolating the various planes of movement within that simple for the developer, a line or two of code in total. However this is generally for use in a project built for IOS or Android. There is very little support for utilising a phone as a controller or input device for a desktop Unity project. Utilising Unity Remote along with mirror networking as the solution came with its own challenges. Unity Remote and Windows OS struggles with permissions and establishing a connection with the phone. Small fixes like the installation of iTunes were needed to prompt the user whether the system was trusted or not and even then was inconsistent. With regard to mirror networking, multiplayer experience was not the target, instead the attempt was to access both streams of data (speed and steering) while dealing with the lack of support for a phone as an input device. Creating a custom network manager and user scripts (handling input) that are capable of the synchronization of a bike model across the network proved difficult. However the solution would be found with continued work.

Similarly to speed data further work into steering might include switching the sensor. The VR controllers and trackers available on the market are very efficient at modelling movement like handlebar steering. Computer vision usage is also a likely path forward.

The Cork model and its disorder was usable within SUMO but posed a challenge modelling in CityEngine. Primary roads function as expected but road boundaries become unsightly due to the labeling of driveways as service roads. CityEngine itself is straightforward in that generating the model is as simple as the click of a button, however managing the resulting model can be difficult and memory consuming. CityEngine is an incredible piece of software but when the OSM file provided is not ordered and cluttered it can struggle to create tidy models. It also comes with the caveat of a very short usage window, the lack of screenshots of the program comes as a result of this.

To compare what was achieved to the initial goal, it is not a complete virtual environment but there is a design in place that would culminate in one, with progress ongoing at the time of report writing.

5.0 Conclusions

As outlined throughout the report the primary objective of the project was to build a virtual cycling environment using open source components (Unity, SUMO, sensors, V2X comms) to learn more about vulnerable road user (VRU) behaviour. The secondary goal being to explore the use of haptic signals to deliver safety related warnings to VRUs if time allowed. The key achievements were:

- Successful BLE connection program and work to integrate within Unity
- Steering integrated in Unity using a Bike model
- Exploration of Cork Unity-SUMO integration, Adapted Manhattan Model demo use
- Real world bike setup

Ultimately the work of the project succeeded in integrating multiple sections of the virtual environment, but efforts to synchronise these are ongoing. This project revolves around a few systems that you need to learn well to use effectively. Knowing how to work efficiently in Unity, SUMO and CityEngine as well as good knowledge of how to code in C# and the sensors used are vital. This was achieved in the current project.

Recommendations for future work would be to investigate the use of higher specification hardware, specifically sensors. The secondary goal to explore the use of haptic signals to deliver safety related warnings to VRUs was not achieved but allow even better analysis of VRU behaviour and could be pursued in future studies.

6.0 References

[1] Cycling Safety Campaign, RSA Ireland (Available Online: https://www.rsa.ie/road-safety/campaigns/cycle-safety-campaign)

[2] The Global status report on road safety 2023 (Available Online: Global status report on road safety 2023 (who.int))

[3] M. Oczko, L. Stratmann et al Integrating Haptic Signals with V2X-based Safety Systems for Vulnerable Road Users(Available Online: https://www2 .tkn. tu-be rlin.de/b ib/oczko2020integrating/o czko2020integrating.pdf)

[4] Real-time-Traffic-Simulation-with-3D-Visualisation (Available Online: https://git hub .com/DarraghMac97/Real-time-Traffic-Simulation-with-3D-Visualisation)

[5] OpenStreetMap Description (Available Online: https://www.openstreetmap.org/about)

[6]ArcGIS CityEngine Overview(Available Online: https://www.esri .com/en-us /arcgis/products/arcgis-cityengine/overview)

[7] Razyn Bike Model (GitHub: https://github.com/RayznGames/BicycleSystem/releases)

[8] Mirror Networking (Documentation: https://mirror-networking.gitbook.io/docs)