Department of Mechanical Engineering
FACULTY OF ENGINEERING AND DESIGN

FINAL YEAR MEng PROJECT REPORT

Eco Academy: An educational video game

Tom Hunt

13.05.2020

Supervisor: Dr Jos Darling

Assessor: Dr Hui Tang

Summary

This project aimed to create an educational video game focused on minimising the fuel consumption of a vehicle in realistic driving conditions. From a literature review, mathematical models of sufficient fidelity were identified for key subsystems. The engine was modelled with static engine maps. A state based clutch model was used to provide distinct clutch behaviours when slipping and sticking. The classical Magic Formula was used to determine the longitudinal tyre force. A one-dimensional vehicle dynamics model was deemed sufficient for the purposes of the learning outcomes of the game. These subsystem models were combined into a full vehicle model described in a block diagram format. A block diagram simulation framework was created in the C++ programming language and the vehicle model was implemented in it. The framework featured a topological sorting algorithm to determine an appropriate block update order. The vehicle simulation was wrapped in a plugin for the game engine Unreal Engine 4. This was then used to create a prototype for the Eco Academy game. Validation tests were performed on the clutch and vehicle dynamics subsystems to ensure they were correctly implemented. Results from these tests matched predictions. Two tests were then executed with the full vehicle model to ensure it qualitatively behaved as expected. The first test showed that the transmission behaved correctly: lower gear ratios enabled a higher terminal velocity but at a lower rate of acceleration. Additionally, engine braking behaviour was also observed. The second test showed that the fuel efficiency of the vehicle responded qualitatively as expected: for a given speed, higher gears increased fuel efficiency and fuel efficiency in a given gear increased as the velocity reduced.

Acknowledgements

I would like to thank the following people from the department. Dr Jos Darling for his guidance and support throughout this project. Dr Hui Tang for providing helpful feedback. Dr Richard Burke for providing me with engine data. Dr Andrew Cookson for his support with simulation techniques.

I also wish to thank my housemates for their suggestions and feedback during the development of my project. Finally, I want to thank my parents for always supporting me throughout my education.

Contents

1 List of Abbreviations

Abbreviation	Definition
DAE	Differential-algebraic system of equations
ICE	Internal combustion engine
UE4	Unreal Engine 4

2 List of Symbols

Symbol	Definition
A	Frontal area
b_e	Engine crankshaft coefficient of viscous friction
b_w	Wheel coefficient of viscous friction
C_d	Coefficient of drag
C_{RR}	Coefficient of rolling resistance
f	Cumulative fuel usage
$\dot{f}$	Fuel flow rate
F_w	Wheel force
g	Acceleration due to gravity
G	Gear ratio
i_G	Gear index
I_e	Engine crankshaft moment of inertia
I_w	Wheel moment of inertia
m	Vehicle mass
r	Wheel radius
T_{cl}	Clutch torque capacity
T_e	Engine torque
T_w	Wheel torque
x	Vehicle position
$\dot{x}$	Vehicle velocity
α	Throttle pedal position
α_{in}	Demand throttle pedal position
β	Brake pedal position
γ	Clutch pedal position
η	Instantaneous fuel efficiency
θ	Road gradient
ρ	Air density
ω_e	Engine crankshaft rotational speed
ω_t	Wheel side clutch plate rotational speed
ω_w	Wheel rotational speed

3 Introduction

3.1 Background

Racing Academy was an educational video game targeted towards students at secondary school level through to undergraduate level [1]. The game was developed in collaboration with the University of Bath Department of Mechanical Engineering. Players competed against a virtual opponent in a drag race format. With each victory the player unlocked a new level of opponent difficulty. This pushed the user into adjusting their vehicle set-up to overcome the new challenge. Parameters that could be changed included gear ratios, engine selection and tyre selection. The process of improving their vehicle set-up engaged the user in engineering and taught them through experimentation about key vehicle engineering principles. The game relied on a physically accurate vehicle model to deliver the experience.

The Racing Academy application is no longer maintained and as a result is not usable on modern hardware. Furthermore, real time computer graphics has advanced significantly since the creation of Racing Academy and the now game looks outdated. A picture from the game is shown in figure 1 for reference. Lastly, public opinion towards climate change and the contribution to this by individuals has shifted since the game was made. Therefore, it was decided that a game based on reducing fuel consumption is more appropriate for today's audience. This project addresses these issues by creating a new educational video game focused on minimising fuel consumption in realistic driving scenarios by adjusting vehicle set-up and driving style.

Figure 1: Picture taken from Racing Academy [2].

3.2 Aims and Objectives

The aim of this project was to build a new educational video game to replace the outdated Racing Academy. The new game was to focus on reducing fuel consumption in realistic driving scenarios by requiring that the user experiment with different driving styles and vehicle set-up. The project objectives, listed in chronological order of completion, were:

1. Create a block diagram simulation framework in C++.
2. Build a vehicle model in the developed framework.
3. Integrate the vehicle model into a plugin for the game engine Unreal Engine 4.
4. Develop a game in Unreal Engine 4 using this plugin.

4 Literature Review

4.1 Overview

Underpinning all physically accurate simulations are mathematical models. The fidelity of a model is often closely tied to the computational effort required to simulate it. Higher fidelity models place a greater burden on the computer solving them. This project required the full vehicle model to be simulated at a rate faster than real-time in order to allow time for the results to be rendered to the screen. Models of key vehicle components were chosen to be of sufficient fidelity to support the learning outcomes of the game, whereas simple models were chosen for components deemed unessential from a learning perspective. A background review of the literature for the modelling of key vehicle components is presented in this section.

4.2 Internal Combustion Engine

The game centres around minimising fuel usage in internal combustion engine (ICE) powered vehicles. In physical ICEs, complex transient behaviour is observed which depends on many factors including: intake air conditions, current turbocharger or supercharger state and the engine operating temperature. As a result engine models span orders of magnitude of complexity and fidelity.

The simplest physically based ICE models use empirically found maps to find torque and other output parameters given engine speed and other input parameters. Maps are created experimentally by applying different loads to a physical engine in a dynamometer and measuring the steady state performance. This approach is computationally efficient and gives reasonable accuracy for a wide range of realistic driving scenarios [3, 4]. Modifications can be made to these to approximate transient behaviour. Applying a first order lag to the torque output of the engine map can be used to simulate turbo lag [5]. A similar methodology has been shown to provide good results for modelling cold starts and their implications on engine emissions [6]. An alternative approach uses a probabilistic method to modify steady state engine maps, delivering accurate engine emissions predictions in transient conditions [7].

In recent years machine learning techniques such as artificial neural networks have been successfully applied to emissions modelling for transient engine behaviour [8, 9, 10]. These models are efficient to evaluate but require a significant amount of computation and data upfront.

For this project it was decided that transient ICE behaviour would not benefit the learning outcomes of the game. Instead, the static engine map approach was chosen for its simplicity.

4.3 Clutch

ICEs can only operate stably above a minimum crankshaft rotational speed. Clutches are used to decouple the speeds of the engine and transmission shafts, allowing the vehicle to come to a stop whilst the engine remains rotating [11].

A clutch consists of two plates that are forced together mechanically. Torque is transferred between the plates by friction. The simplest model of friction commonly used in clutch modelling is Coulomb friction. In this model, torque is proportional to the normal force pushing the plates together and acts in the opposite direction to the relative speed of the plates. This approach introduces a discontinuity when the relative speed is zero so instead can be approximated using a smooth hyperbolic tangent function to improve numerical stability [12]. Non-linear friction effects such as Stribeck friction, which increases the friction provided at higher relative speeds, are sometimes modelled, however, these were deemed too complicated for this project [13]. The highest fidelity models use Finite Element techniques and account for heating effects on the ability of clutch plates to provide friction [14]. This approach is very computationally intensive and so was not considered for this project.

Clutches have two distinct behavioural states: an unlocked (slipping) state and a locked (sticking) state. In the unlocked state the torque applied to the clutch is greater than the friction capacity of the clutch and the two plates rotate independently. In the locked state the torque applied is less than the friction capacity of the clutch and the two plates rotate at the same rate, acting as a single shaft. This has the effect of removing a degree of freedom from the system. A common method of simulating this is to have separate dynamic systems representing each state and switch between them based on the operating conditions [15, 16]. This works well for single clutch systems but becomes increasingly complex as clutches are added as each permutation of clutch states must be modelled and switched between. The same effect can also be achieved using a single matrix

system by using matrix manipulation techniques to cancel out certain rows and hence degrees of freedom [17]. This is particularly useful for multi clutch systems. A third approach to approximating locking behaviour is to artificially add a stiff torsional spring and damper between the two clutch plates when in the lock state [18]. This has the benefit of maintaining the same number of degrees of freedom and so reduces the complexity of the simulation. However, the torsional spring has to be sufficiently stiff to mimic locking which greatly reduces the numerical stability of the system. As a result, integration schemes designed for stiff systems must be used in order for the simulation to remain stable with a reasonable timestep. This project concerns passenger vehicles which typically only use one clutch. Therefore, the two dynamic systems approach was chosen for its simplicity.

4.4 Tyre

A key learning outcome of the game is the impact of driving style on fuel economy. It was decided that just longitudinal motion was sufficient to investigate different driving styles and hence only longitudinal tyre dynamics need be modelled.

A widely used tyre friction model is the empirically derived Magic Formula model by Pacejka created in the 1980's [19, 20]. A computationally simple function with few parameters is fitted to experimental data to relate slip ratio and tyre normal force to longitudinal force. The accuracy provided by this method given its simplicity makes it ideal for real time simulation. Extensions to this model have been made to predict other phenomena such as the aligning moment due to pneumatic trail, however, these tend to concern three-dimensional motion and hence are unnecessary for this project [21].

Tyres are dynamic systems with complex properties dependant on the visco-elastic behaviour of the tyre material [22, 23]. This can be modelled to determine the tyre normal force. Figure 2 shows different models for this behaviour in order of increasing fidelity and complexity. To use these models the pitching behaviour of the vehicle must be simulated. This project treated vehicle motion as one-dimensional so this information was not available. As a result, the normal force was assumed constant and only a function of vehicle weight and weight distribution.

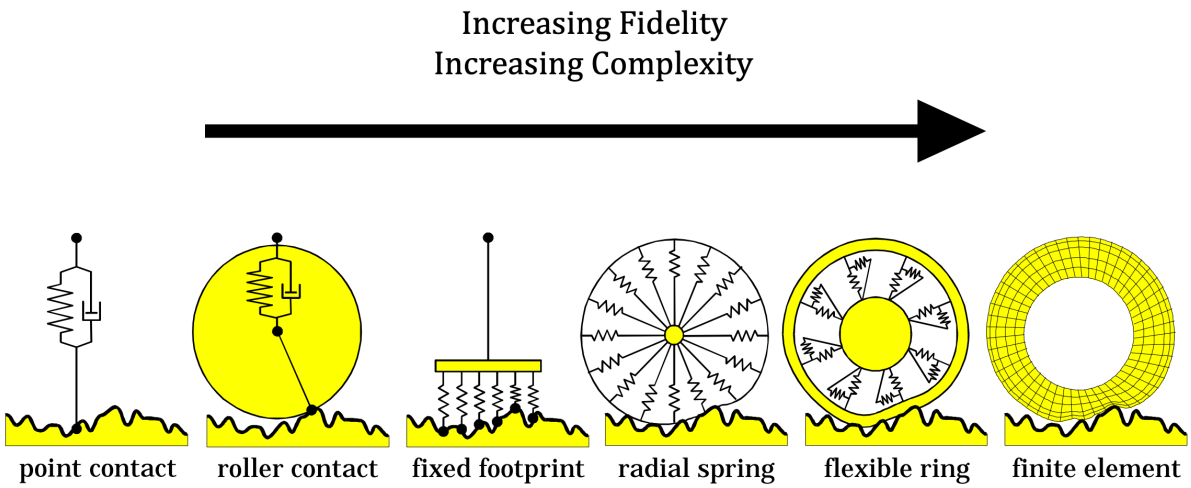


Figure 2: An overview of different models for tyre visco-elasticity [23].

5 Vehicle Model

5.1 Overview

In this section a description of the complete vehicle model is presented. The model is comprised of distinct subsystems which represent different systems on a real vehicle. A diagram of the significant systems which make up the drivetrain of a typical passenger vehicle is given in figure 3 as reference. The inertia of the engine crankshaft and flywheel and the inertia of the transmission, propeller shaft, differential, rear axle and wheels are modelled in the clutch and transmission subsystem. Torques are exerted on these inertias by the engine and wheel subsystems. The longitudinal inertia of the vehicle is modelled in the vehicle dynamics subsystem and is forced by the wheel subsystem. The road subsystem provides information on the gradient of the road, given the vehicle position. The vehicle controller subsystem mimics the behaviour of driver clutch and throttle pedal inputs. The block diagram for the complete system showing each subsystem interaction is given in figure 4.

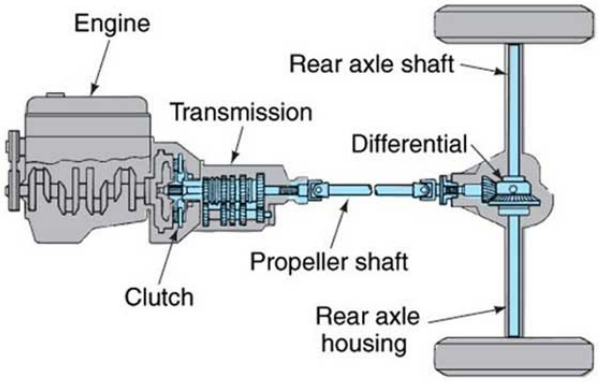


Figure 3: Labelled drivetrain of a typical passenger vehicle [24].

All dynamic systems were transformed into state space form, stated in equation 1 for reference. The top equation expresses how the state vector, $\mathbf{x}$, changes in time in response to the input vector, $\mathbf{u}$. This is integrated in time to give the trajectory of the state vector. The bottom equation is used to augment the state and input vector to give an output vector, $\mathbf{y}$, of arbitrary length. To transform a dynamic system into this format, the equations of motion are decomposed into a system of first order ODEs.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned} \tag{1}$$

A block common to multiple subsystems is the Coulomb friction block. This block has two inputs: the relative sliding velocity of the two friction surfaces, $\dot{x}$, and the normal force pushing the friction surfaces together, F_N. The equation for Coulomb friction is presented in equation 2. Here μ is the coefficient of friction and is held constant.

$$F = \text{sign}(\dot{x})\mu F_N \tag{2}$$

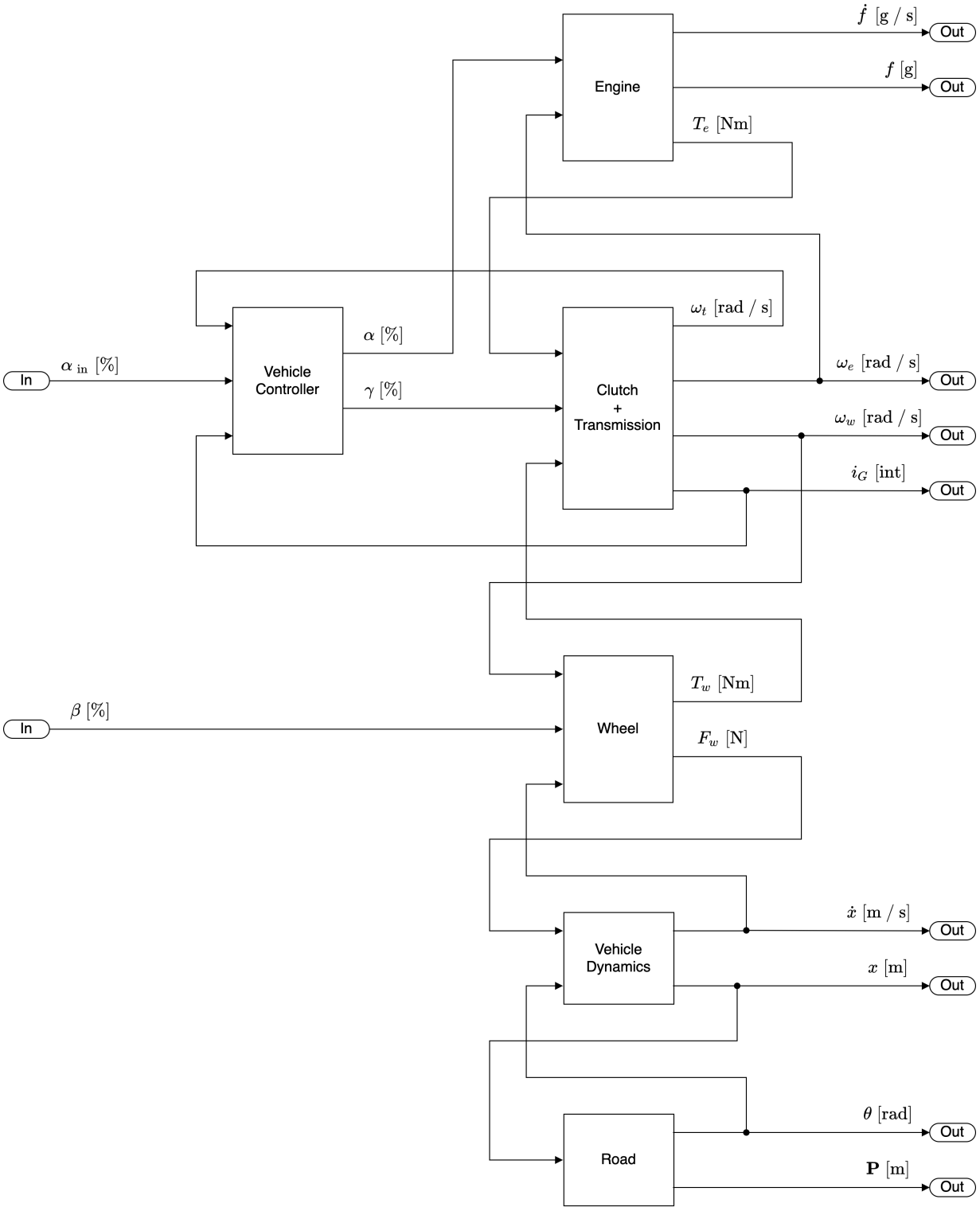


Figure 4: Vehicle model block diagram showing the interaction of subsystems.

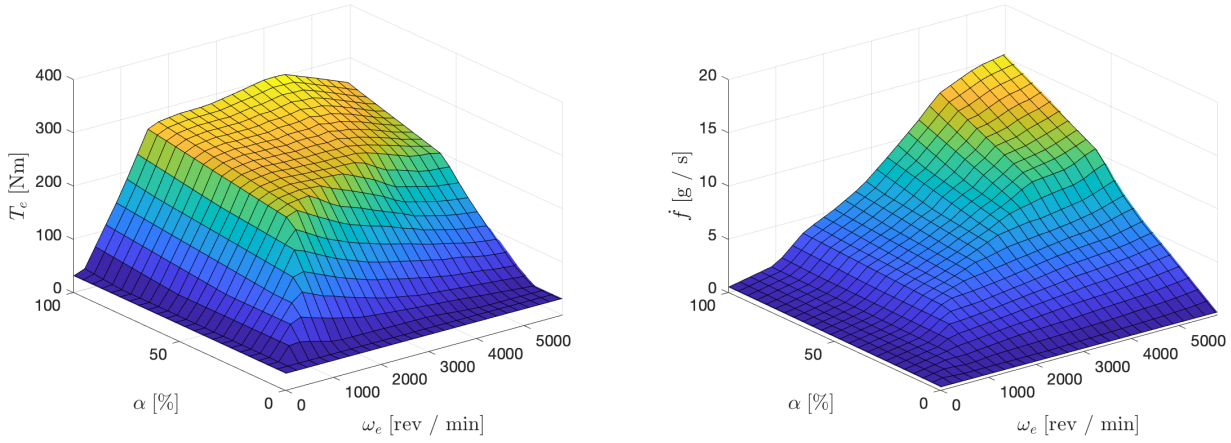
5.2 Engine

The engine subsystem outputs torque, T_e, cumulative fuel usage, f, and fuel flow rate $\dot{f}$. Static engine maps are used to approximate torque and fuel flow rate given crankshaft rotational speed, ω_e, and throttle pedal position, α. The engine maps used are shown in figure 5.

The maps were created by modifying measured data from an engine dynamometer test. The speed range was extended to give an output torque at speed values below the stable speed of the engine. This simplified the model as it removed the need to model engine stalling and ignition which are very transient phenomena. Instead the engine outputs a small amount of torque at 0 rpm, to allow the user to accelerate the engine back to normal operating speeds. This change was replicated in the fuel map by adding in small values of fuel rate even at 0 rpm.

Complex behaviours can be encoded by these maps in order to provide a clearer distinction between engine types. In the torque map presented there is only a small increase in torque when the throttle is in the range 50% and 100%, but a considerable increase in fuel flow rate. This requires users to perfect their driving style to use as little throttle as possible to maintain speed.

Figure 6 shows the block diagram for the engine subsystem. The map blocks linearly interpolate within the engine maps. The output of the fuel flow rate map is integrated to give the cumulative fuel usage.



(a) Torque map.

(b) Fuel flow rate map.

Figure 5: Engine maps used.

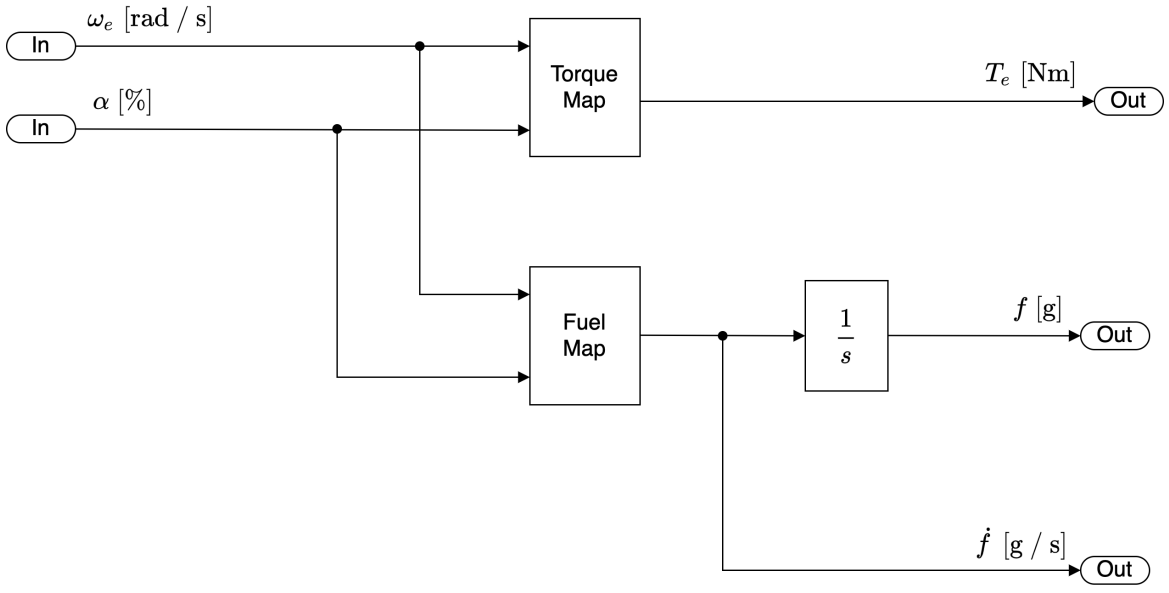


Figure 6: Engine subsystem block diagram.

5.3 Clutch and Transmission

The clutch and transmission subsystem models the inertia of the drivetrain. There are two distinct states of behaviour for the drivetrain. When the torque exerted on the clutch is greater than the ability of the clutch to transmit the torque, for example when the clutch pedal is down and the clutch plates are pulled apart, the engine crankshaft rotates independently of the other drivetrain elements. This is the unlocked state. When the torque capacity of the clutch is greater than the torque exerted on the clutch and speeds of the clutch plates match, the clutch enters the locked state and the entire drivetrain is modelled as rigidly connected. The two states and the conditions for transitioning between them is shown in the state diagram in figure 7. In the unlocked state the rotational velocity of engine crankshaft is labelled ω_e and the rotational velocity of the other drivetrain components, referred to as simply the wheel from this point, is labelled ω_w. The rotational velocity of the wheel side clutch plate, ω_t, is related to the wheel velocity by the transmission speed ratio G. The torque capacity of the clutch is T_{cl} and the torque exerted on the clutch is T_{ex}.

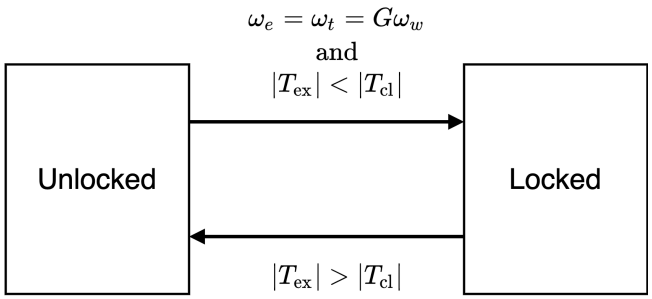


Figure 7: State diagram of clutch behaviour.

Figure 8 shows a free body diagram of the clutch and transmission in the unlocked state. The input torque from the engine is T_e and the load torque from the wheel is T_w. Parameters I_e and b_e represent the moment of inertia and coefficient of viscous friction for the engine and I_w and b_w represent the moment of inertia and coefficient of viscous friction for the wheel. In the unlocked state the friction torque generated by the plates is equal to the torque capacity of the clutch, modelled as a simple Coulomb friction. The coefficient of friction is set to the peak torque that can be transferred by the clutch when the clutch pedal is fully up. The normal force input is the clutch pedal position, γ, with a value of 0 representing the clutch pedal fully down and a value of 1 representing the clutch pedal fully up. Directionality input is given by $\omega_e - \omega_t$.

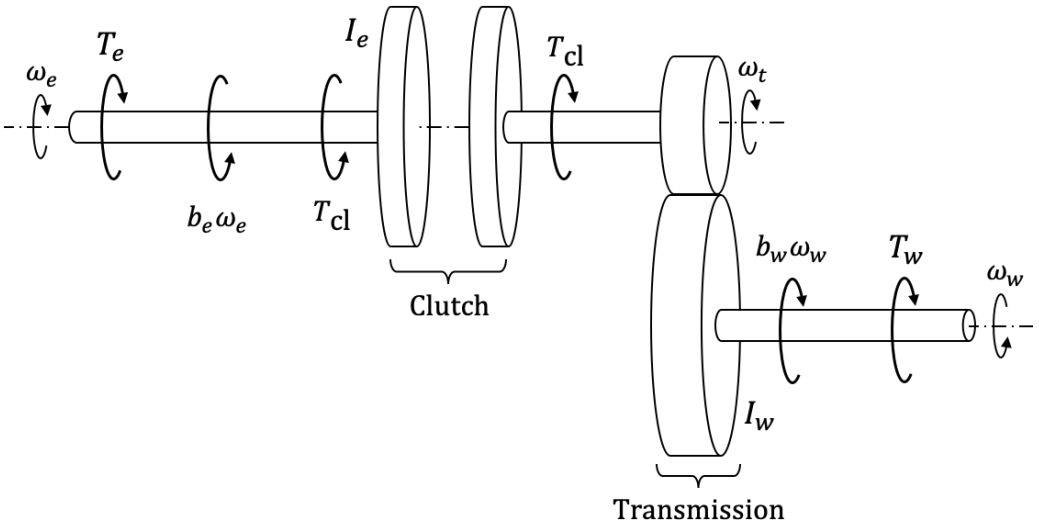


Figure 8: Clutch and transmission free body diagram in the unlocked state.

Equations of motion for ω_e and ω_w in the unlocked state are given in equations 3a and 3b respectively.

$$T_e - b_e\omega_e - T_{\text{cl}} = I_e\dot{\omega_e} \tag{3a}$$
$$GT_{\text{cl}} - b_w\omega_w - T_w = I_w\dot{\omega_w} \tag{3b}$$

This is written in state space form with vectors and matrices as in equation 4.

$$\mathbf{x} = \begin{bmatrix}\omega_e \\ \omega_w\end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix}T_{\text{cl}} \\ T_e \\ T_w\end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix}\omega_t \\ \omega_e \\ \omega_w\end{bmatrix} \tag{4}$$

$$\mathbf{A} = \begin{bmatrix}-\frac{b_e}{I_e} & 0 \\ 0 & -\frac{b_w}{I_w}\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}-\frac{1}{I_e} & \frac{1}{I_e} & 0 \\ \frac{G}{I_w} & 0 & -\frac{1}{I_w}\end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix}0 & G \\ 1 & 0 \\ 0 & 1\end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix}0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0\end{bmatrix}$$

When the clutch is in the locked state, velocity continuity requires $\omega_e = \omega_t = G\omega_w = \omega$ and hence $\dot{\omega_e} = \dot{\omega_t} = G\dot{\omega_w} = \dot{\omega}$. Rearranging equations 3a and 3b for T_{cl}, equating and manipulating gives the equation of motion for the locked system, equation 5.

$$\dot{\omega} = \frac{1}{(I_w + G^2I_e)}\left[G^2T_e - (G^2b_e + b_w)\,\omega - GT_w\right] \tag{5}$$

Defining the combined effective inertia, $I_{\text{eff}} = I_w + G^2I_e$, the equation of motion is written in in state space form with vectors and matrices as in equation 6.

$$\mathbf{x} = \begin{bmatrix}\omega\end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix}T_e \\ T_w\end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix}\omega_t \\ \omega_e \\ \omega_w\end{bmatrix} \tag{6}$$

$$\mathbf{A} = \begin{bmatrix}\frac{-\left(G^2b_e + b_w\right)}{I_{\text{eff}}}\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}\frac{G^2}{I_{\text{eff}}} & -\frac{G^2}{I_{\text{eff}}}\end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix}1 \\ 1 \\ \frac{1}{G}\end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix}0 & 0 \\ 0 & 0 \\ 0 & 0\end{bmatrix}$$

The exerted torque, T_{ex}, is found using the same velocity continuity condition, rearranging equations 3a and 3b for $\dot{\omega}$, equating an rearranging for T_{cl}. In this state this is equivalent to the exerted torque on the clutch. The result is presented in equation 7.

$$T_{\text{tr}} = \frac{1}{(I_w + G^2I_e)}\left[I_wT_e + I_eGT_w + (I_eb_w - I_wb_e)\,\omega\right] \tag{7}$$

The block diagram of the clutch and transmission subsystem is presented in figure 9. The two states are implemented as separate state space system blocks. A switch is used to change which state space system block is read by downstream blocks. The crossing detection block outputs a value of true when the value of $\omega_e - \omega_t$ crosses zero and false at every other time. This is used to determine when the criteria for transitioning to the locked state are met. The lockup state controller block evaluates the exerted torque and throws the switch block if the criteria for transitioning to the locked state are met.

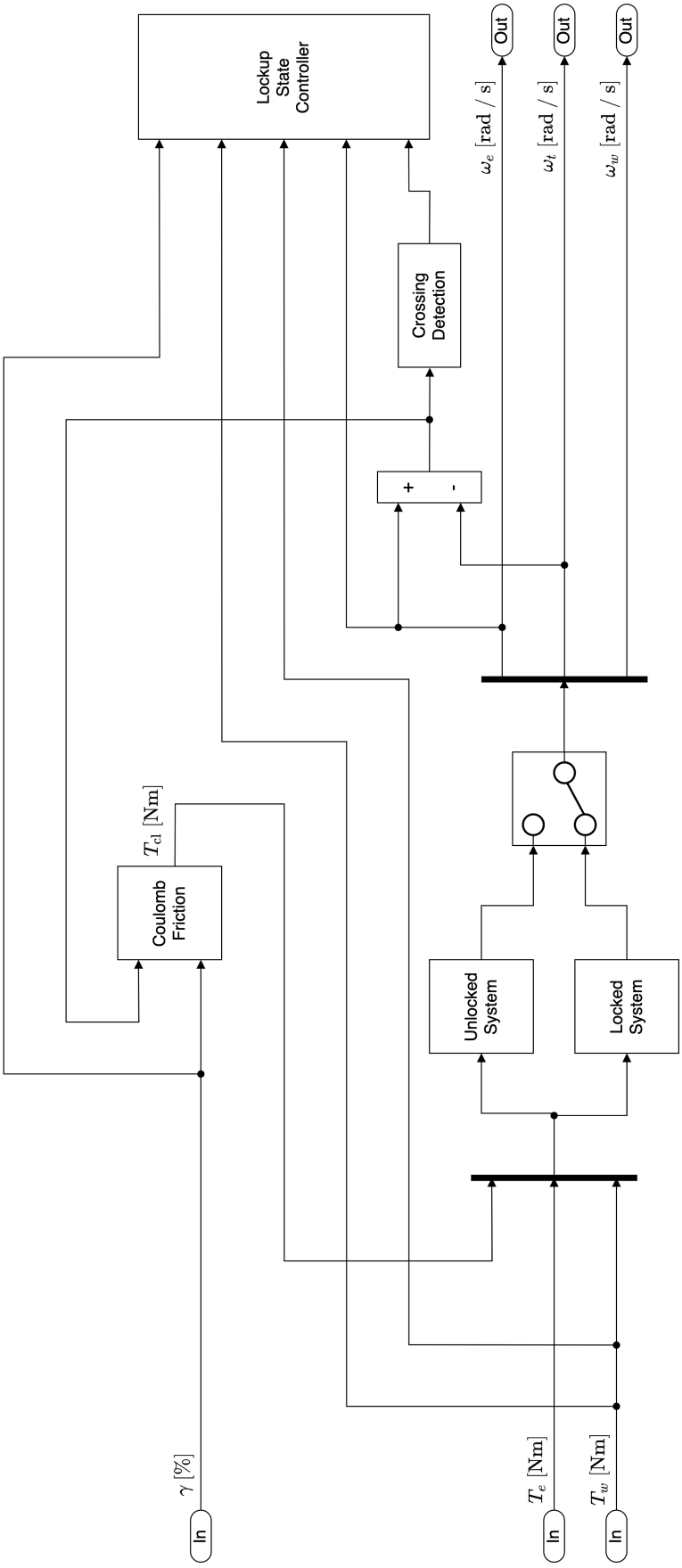


Figure 9: Clutch and transmission subsystem block diagram.

5.4 Wheel

The wheel subsystem outputs a torque, T_w, to act on the clutch and transmission subsystem and a force, F_w, to act on the vehicle dynamics subsystem. The inputs to the subsystem are the rotational speed of the wheel, ω_w, the linear velocity of the vehicle, $\dot{x}$, and the brake pedal position, β.

The longitudinal force provided by the tyre is modelled using the so called Magic Formula given in equation 8. The vertical force acting on the tyre, F_z, was modelled as the static weight of the vehicle. The values used for the coefficients were: $B = 10, C = 1.9, D = 1, E = 0.97$. This is multiplied by the tyre radius, r, to give the tyre torque, T_{tyre}.

$$F_w = F_z D \sin(C \arctan(Bk - E(Bk - \arctan(Bk)))) \tag{8}$$

The wheel slip ratio, k, is a non-dimensionalised measure of the amount of wheel spin. When $k = 0$, the rate of rotation of the wheel is equivalent to the rate of rotation of a wheel of the same radius rotating freely without any sliding. When $k > 0$, the rate of rotation is greater such that the tyre is sliding. To avoid a singularity when the vehicle is stationary, the slip ratio is approximated for speeds below a threshold $\dot{x}_{\text{th}}$.

$$k = \begin{cases} \dfrac{r\omega_w - \dot{x}}{|\dot{x}|}, & |\dot{x}| > \dot{x}_{\text{th}} \\ \dfrac{2(r\omega_w - \dot{x})}{\left(\dot{x}_{\text{th}} + \dfrac{\dot{x}^2}{\dot{x}_{\text{th}}}\right)}, & |\dot{x}| <= \dot{x}_{\text{th}} \end{cases} \tag{9}$$

Braking is modelled with a simple Coulomb friction model with the peak brake torque set as the coefficient of friction, the normal force set as the brake pedal position, β, and the directionality given by the wheel rotational speed, ω_w. The output of this is summed with the tyre torque to give the resultant torque acting on the clutch and transmission subsystem. The block diagram for this subsystem is presented in figure 10.

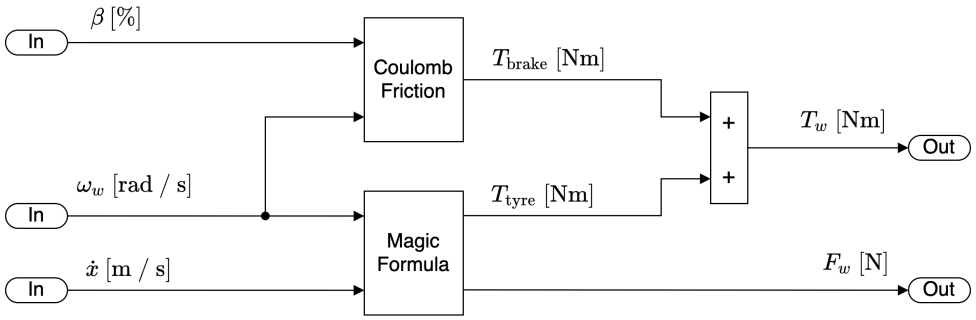


Figure 10: Wheel subsystem block diagram.

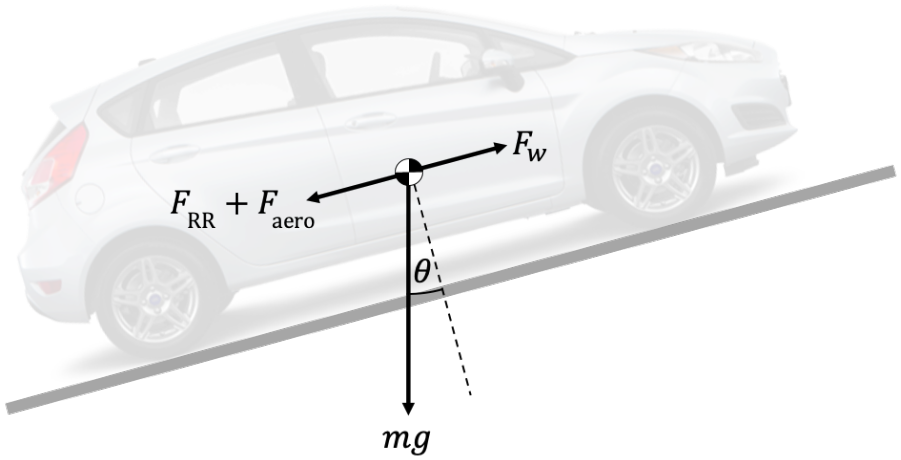


Figure 11: Free body diagram of a vehicle.

5.5 Vehicle Dynamics

The vehicle dynamics subsystem outputs vehicle displacement, x, and velocity, $\dot{x}$. The only input is the propulsive force from the driving wheels, F_w. The vehicle is modelled as a point mass constrained to a curve on the forward-vertical plane. A free body diagram showing forces acting on the vehicle in presented in figure 11. Labelled here is: the force due to rolling resistance, F_{RR}, the force due to aerodynamic drag, F_{aero}, the weight of the vehicle, mg, and the gradient of the road, θ. The vehicle equation of motion in the tangential direction is given in equation 10. Here: ρ is the ambient air density, A is the frontal area of the vehicle, C_d is the coefficient of drag and C_{RR} is the coefficient of rolling resistance.

$$F_w - F_{\text{grav}} - F_{\text{aero}} - F_{\text{RR}} = m\ddot{x} \tag{10}$$

$$\begin{aligned} F_{\text{grav}} &= mg\sin(\theta) \\ F_{\text{aero}} &= \text{sign}(\dot{x})\,\frac{1}{2}\rho A C_d \dot{x}^2 \\ F_{\text{RR}} &= \text{sign}(\dot{x})\,mg\,C_{\text{RR}} \end{aligned}$$

The equation of motion was implemented as a state space model with the input and state vectors and state space matrices as presented in equation 11.

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} F_w & F_{\text{grav}} & F_{\text{aero}} & F_{\text{RR}} \end{bmatrix}^T, \quad \mathbf{y} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{11}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} & -\frac{1}{m} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The block diagram of the vehicle dynamics subsystem is shown in figure 12. Rolling resistance was implemented as a Coulomb friction block. Equations for aerodynamic drag and gravity force were implemented as unique blocks.

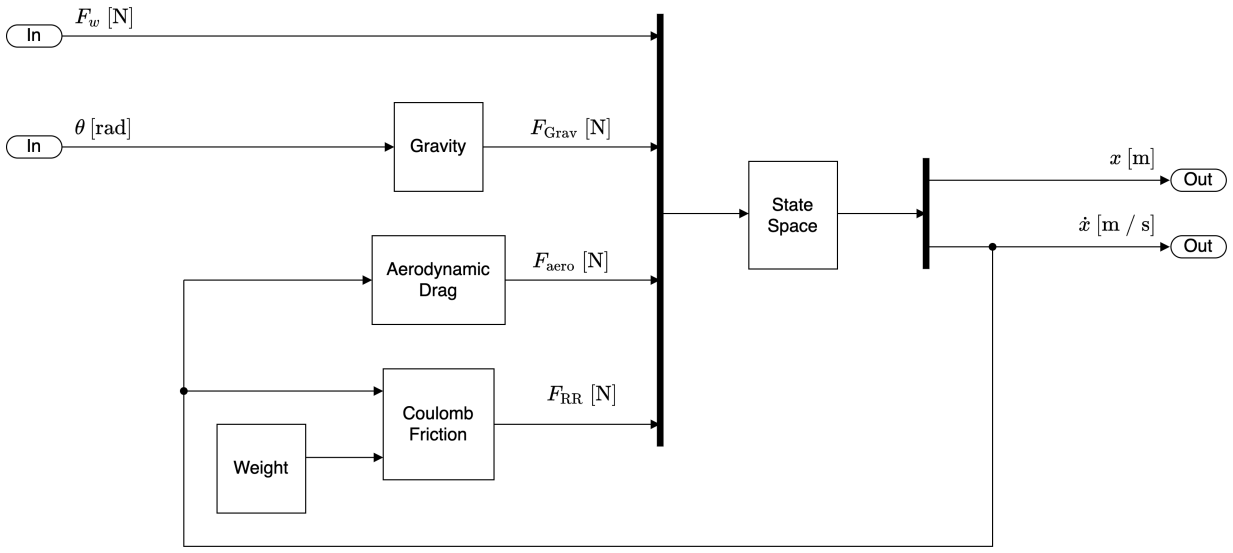


Figure 12: Vehicle dynamics subsystem block diagram.

5.6 Vehicle Controller

The vehicle controller subsystem was created to: simulate clutch pedal input when accelerating from a standstill, simulate clutch and throttle pedal input when changing gear and approximate neutral gear. Inputs to the subsystem are: the rotational speed of the clutch plate on the wheel side, ω_t, the index of the current gear, i_G, and the demand throttle pedal position, α_{in}. Outputs are: an augmented throttle pedal position, α, and the clutch pedal position, γ.

The clutch controller block contains the logic for approximating neutral gear and controlling the clutch pedal position when moving off from a standstill. Neutral gear is approximated by setting the clutch pedal to be in the fully down position, $\gamma = 0$, when the gear index, i_G, is zero. When the vehicle is in gear and the input throttle position is non-zero, the vehicle controller will output a non-zero value of clutch pedal position. When moving off, the clutch pedal position is increased linearly from a starting value, γ_0, to $\gamma = 1$ at a threshold speed, ω_{thr}, as shown in figure 13. Above the threshold speed the clutch pedal position is always fully up.

From the moment the vehicle changes gear, the throttle and clutch pedal positions are assumed to vary linearly in time from 0 back to 1. A trigger block was used to simulate this. A trigger block contains its own clock. When triggered the clock resets and the output of the block is set to 1. This then drops linearly back to 0 over a period of τ_{trig} seconds, as show in figure 14. The output of the trigger block was set as the parameter λ input of the linear blend blocks. The linear blend blocks linearly interpolates between two signals, $\mathbf{x_0}$ and $\mathbf{x_1}$, according to equation 12. A block diagram of the vehicle controller subsystem is shown in figure 15.

$$\mathbf{g}(\mathbf{x_0}, \mathbf{x_1}, \lambda) = (1 - \lambda)\mathbf{x_0} + \lambda\mathbf{x_1} \tag{12}$$

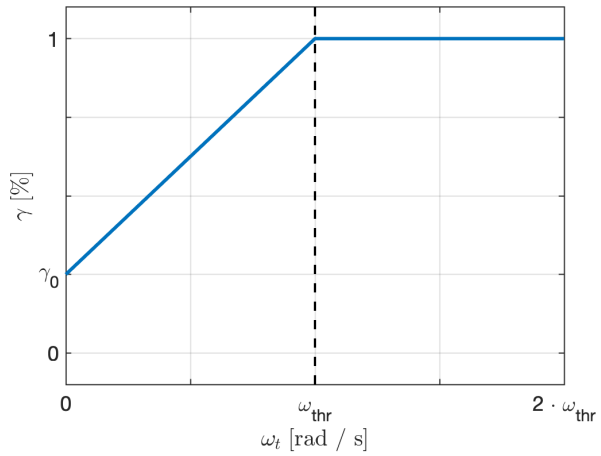


Figure 13: Clutch pedal position with transmission speed.

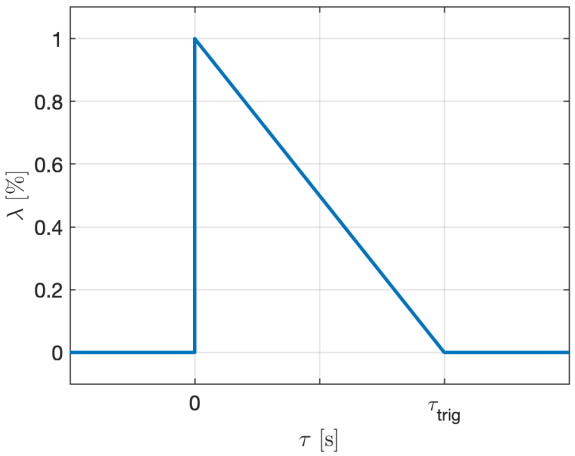


Figure 14: Trigger block output.

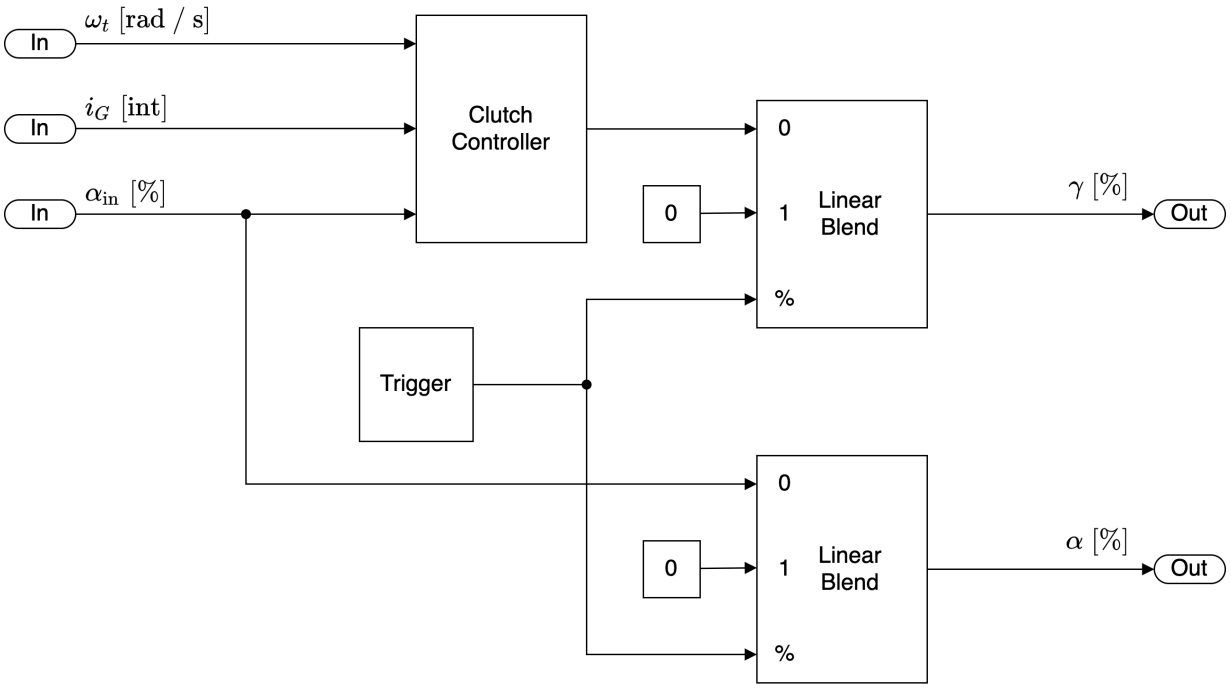


Figure 15: Vehicle controller subsystem block diagram.

5.7 Road

The road subsystem contains all information about the geometry of the road. The subsystems returns the gradient of the road, θ, and the three dimensional coordinate of the road, $\mathbf{P}$, at the input vehicle displacement, x. The road is modelled as a piecewise linear curve. For each road element, equation 13 is computed, where L_i is the arclength of road element i. This represents the total arclength up to the start of each element. The index of the road element in which x falls is found from this. The three dimensional coordinate is then interpolated from the coordinates of the two ends of the element. The gradient is constant within each road element and is computed before simulation. All functionality of the subsystem is implemented in one block, as seen in figure 16.

$$L_i = \sum_{j=1}^{i-1} L_j \tag{13}$$

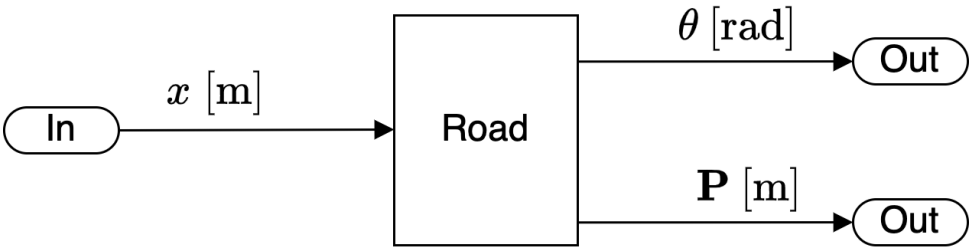


Figure 16: Road subsystem block diagram.

6 Block Diagram Simulation

6.1 Overview

The vehicle model presented was described in a block diagram representation. This representation conveniently conveys what is in fact a large differential-algebraic system of equations (DAE) in a readable, causal format. Numerical algorithms to solve generalised DAEs are complex. However, as the causality of the block diagram is explicit by its structure, the simple algorithm presented in this section could be used. Another benefit of block diagram representation is the modularity that it provides; each individual subsystem can be implemented and tested independently from the others and models can easily be replaced by alternative models if required.

6.2 Block Types

6.2.1 Overview

A block diagram is made up of blocks and signals which connect them. Blocks can perform mathematical operations using the values of their input signals and can write the result to output signals. Four different fundamental block types were implemented in the simulation framework. Block inputs are labelled $\mathbf{u}$. Some blocks have an internal state, labelled $\mathbf{x}$. Block outputs are labelled $\mathbf{y}$.

6.2.2 Source

The source block type has no input signals. Source blocks are of the form given in equation 14. As implemented in the numerical solver, the output of a source block at the next timestep is computed based on the time at the next timestep and the internal state at the next timestep as expressed in equation 15. Two commonly used source blocks used in this project are input and constant blocks. The input block writes out the value of its internal state which can be set from outside the solver. The constant block writes out a single value for the duration of the simulation.

$$\mathbf{y} = \mathbf{y}(t, \mathbf{x}) \tag{14}$$

$$\mathbf{y_{n+1}} = \mathbf{y}(t_{n+1}, \mathbf{x_{n+1}}) \tag{15}$$

6.2.3 Dynamic System

A dynamic system is a system which can evolve over time in response to input and an internal state. They are described by an equation of state in the form given in equation 16. This is integrated in time to give the trajectory of the state vector. The output of a dynamic system block is a function in the form given by equation 17.

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}(t, \mathbf{x}, \mathbf{u}) \tag{16}$$

$$\mathbf{y} = \mathbf{y}(t, \mathbf{x}, \mathbf{u}) \tag{17}$$

This project used classical explicit Runge-Kutta numerical integration to determine the value of the state vector at each subsequent time step. A zero order hold approximation was applied to the input vector for simplicity. As implemented, the values of the state vector and output vector at the next timestep take the form expressed in 18 and 19 respectively. The state space block is the most significant example of a dynamic system block. The equation of state and output equation of a state space system are given in equation 1.

$$\mathbf{x_{n+1}} = \mathbf{x}(t_n, \mathbf{x_n}, \mathbf{u_n}) \tag{18}$$

$$\mathbf{y_{n+1}} = \mathbf{y}(t_{n+1}, \mathbf{x_{n+1}}, \mathbf{u_n}) \tag{19}$$

6.2.4 Function

The output of a function block is a pure function of its inputs. It contains no states and is not transient. Functions take the form presented in equation 20. The output of a function block at the next timestep is the output of its function applied to the input value at the next timestep as shown in equation 21 . Function blocks were the most common type used in this project. Examples include: gains, summing junctions and switches.

$$\mathbf{y} = \mathbf{y}(\mathbf{u}) \tag{20}$$

$$\mathbf{y_{n+1}} = \mathbf{y}(\mathbf{u_{n+1}}) \tag{21}$$

6.2.5 Sink

A sink block takes in input but gives no output. It has an internal state which is updated according to the relationship given in equation 22. In the numerical solution the value of the internal state at the next timestep is a function of the time and input vector at the next timestep, as expressed in equation 23. The output block is a sink which copies the input value to its state to be read from outside the solver.

$$\mathbf{x} = \mathbf{x}(t, \mathbf{u}) \tag{22}$$

$$\mathbf{x_{n+1}} = \mathbf{x}(t_{n+1}, \mathbf{u_{n+1}}) \tag{23}$$

6.3 Simulation Initialisation

The initialisation algorithm sets the internal states of source and dynamic system blocks to specified initial conditions, $\mathbf{x_0}$, and updates their outputs to initial values, $\mathbf{y_0}$. The output of function blocks is then calculated based on the initialised outputs of source and dynamic system blocks. Finally, the state of sink blocks is updated based on the initialised outputs of all other blocks. The initialisation algorithm is shown as a flow chart in figure 17. This algorithm is approximate as dynamic system blocks write initial output based only on initial conditions and not also input as they would normally. This approximation greatly simplified the initialisation algorithm and was not found to be an issue for this project as the output equations for all dynamic system blocks used did not use the block input; all $\mathbf{D}$ matrices in state space blocks were zero valued.

6.4 Simulation Update

The update algorithm updates the block diagram from its current time value, t_n, to the next time value, t_{n+1}. Initially, the block diagram is everywhere at time t_n, internal states are $\mathbf{x_n}$, input signals are $\mathbf{u_n}$ and output signals are $\mathbf{y_n}$. The algorithm is presented as a flow chart in figure 18.

First all dynamic system blocks read their inputs, $\mathbf{u_n}$. Next each dynamic system block integrates its equation of state from time t_n to time t_{n+1} to give $\mathbf{x_{n+1}}$. The zero order hold approximation applied to the input avoids the need to determine input values between timesteps in a multi-stage integration scheme. The output of the dynamic system blocks, $\mathbf{y_{n+1}}$, is then computed and written to the output signals. At this stage the output from all dynamic system blocks is at time t_{n+1}, however, source, function and sink blocks are still at time t_n. Next all source blocks update their state to time t_{n+1} and write output, $\mathbf{y_{n+1}}$. As sink blocks give no output, all inputs to function blocks are now at time t_{n+1}. In sorted order, discussed in the following subsection, each function block reads input, $\mathbf{u_{n+1}}$, then computes and writes output, $\mathbf{y_{n+1}}$. Finally, all sink blocks read input, now everywhere at time t_{n+1}, and update internal states.

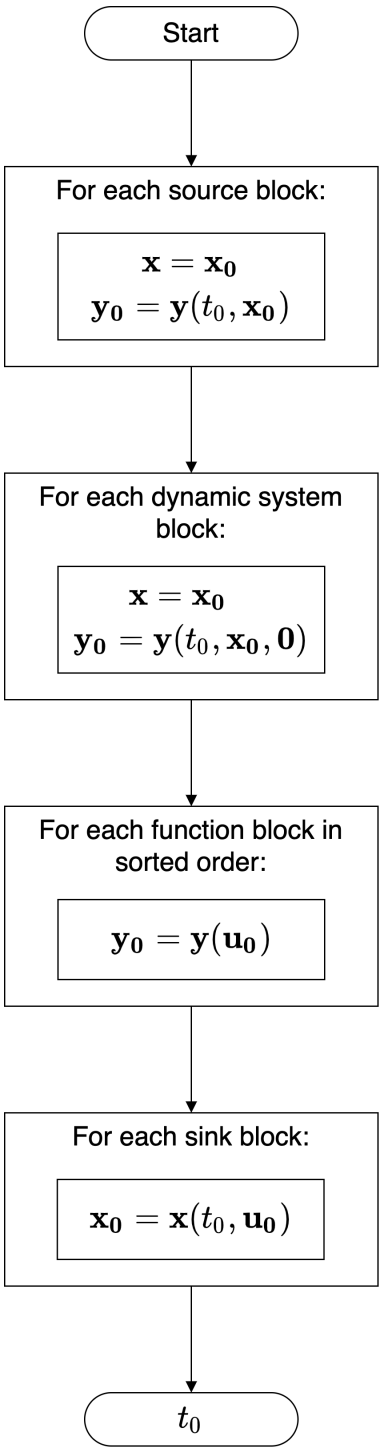


Figure 17: Flow chart of the initialisation algorithm used to update the block diagram to t_0.

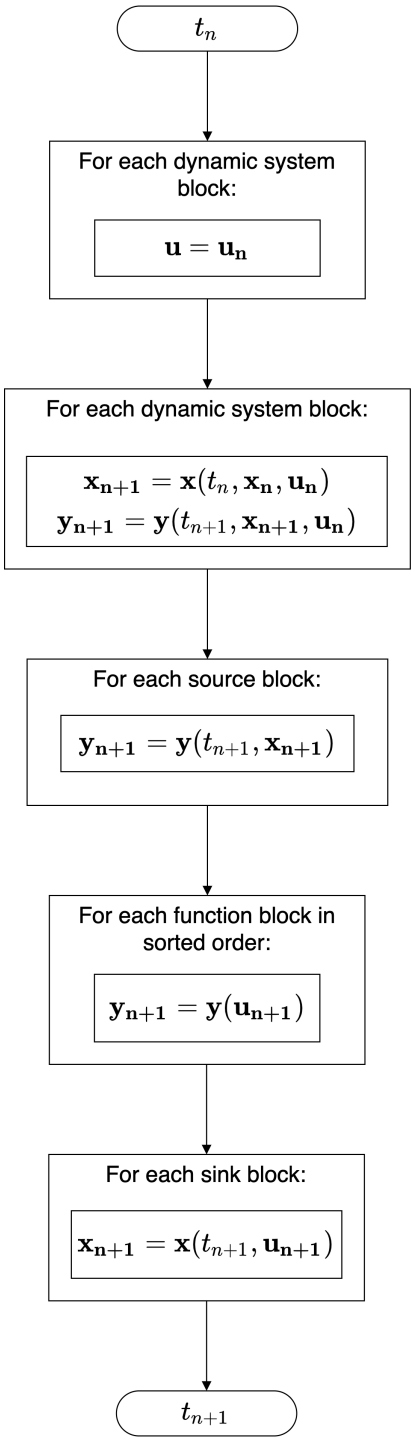


Figure 18: Flow chart of the update algorithm used to advance the simulation in time from t_n to t_{n+1}.

6.5 Block Sorting

A block diagram can be represented as mathematical graph with blocks as nodes and signals as edges. Each signal can only be written to by one block, making the graph directed. The directedness of a block diagram graph ensures causality in the model has been determined. The output of a function block directly depends on the input to the block. As a result, all input signals must be updated before the block itself updates. As function blocks can feed into other function blocks, every upstream function block must be updated before a function block can be updated. To sort blocks manually becomes an error prone task as models become larger, so an automated block sorting algorithm was implemented. To simplify the solution, block diagrams were required to contain no algebraic loops. This ensures that subgraphs made up of function blocks are acyclic. This property enabled the use of a simple topological sorting algorithm to determine the update order for function block subgraphs. Figure 19 shows an example block diagram made up solely of function blocks. It can be seen from inspection that blocks must be updated in the order: A, B then C or C then B, D, E, F then G or G then F.

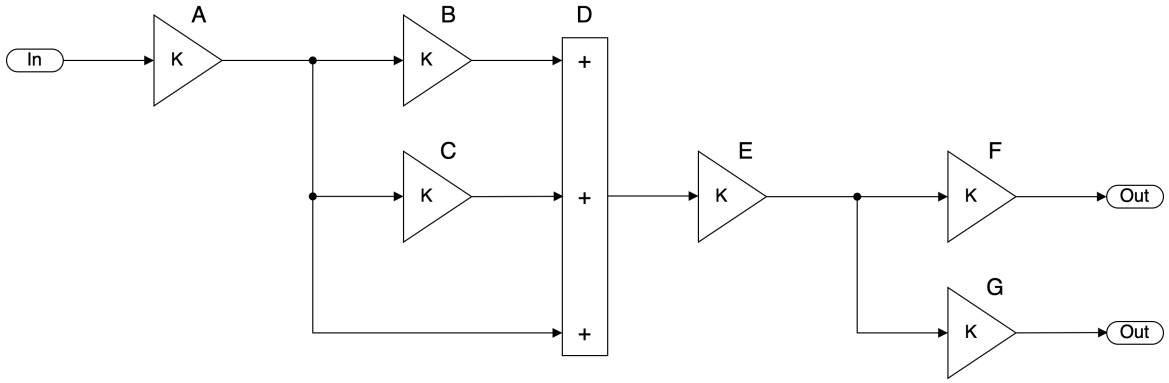


Figure 19: Example block diagram consisting solely of function blocks.

The equivalent graph for this block diagram is shown in figure 20. The adjacency list for the graph is also shown. This is a list of edges leaving each node. The sorting algorithm takes a list of nodes in arbitrary order and the corresponding adjacency list and returns the list of nodes sorted into a correct update order. Two data structures are maintained during the course of the algorithm: the visited list and the node stack. The visited list tracks whether or not the algorithm has visited a node. The node stack is an initially empty stack that is populated by the algorithm as it runs. The algorithm is made up of an outer function and a recursive inner function. The pseudo code for these functions is given in figure 21. An example of the algorithm working for the graph in figure 20 is shown in figure 22. Here the node list is initially in the order: F, E, D, A, B, C, G. The algorithm corrects sorts the node list into the order: A, C, B, D, E, G, F.

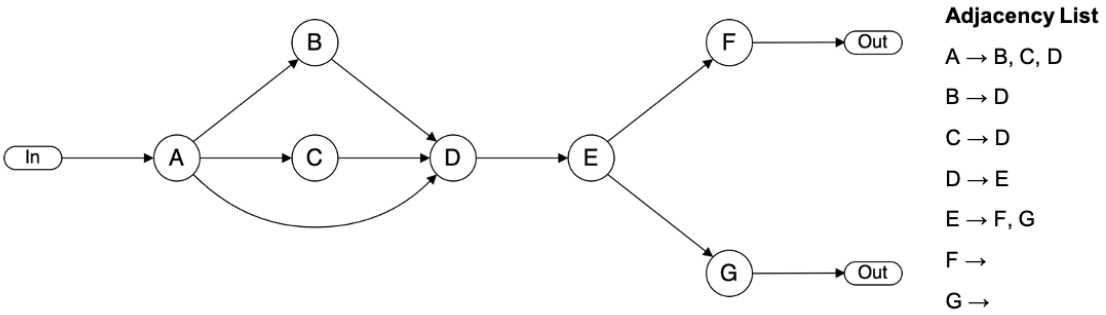


Figure 20: Equivalent graph for the block diagram presented in figure 19.

```
SortBlocks(node list, adjacency list)
{
    set all nodes unvisited

    for each node i in list
    {
        call RecursiveFunction(node i)
    }

    reverse stack

}
```

```
RecursiveFunction(node i)
{

    if node i unvisited
    {
        for each adjacent node j
        {
            call RecursiveFunction(node j)
        }

        add node i to the stack
        mark node i as visited
    }
}
```

Figure 21: Pseudo code for the topological sort algorithm.

1. Initialise stack and visited list.

Node	A	B	C	D	E	F	G
Visited	0	0	0	0	0	0	0

Stack							

2. RecursiveFunction(Node F)

Node F unvisited:
No adjacent nodes.
Add F to stack.
Mark F visited.

Node	A	B	C	D	E	F	G
Visited	0	0	0	0	0	1	0

Stack	F						

3. RecursiveFunction(Node E)

Node E unvisited:
Two adjacent nodes:
Node F visited.
Node G unvisited:
No adjacent nodes:
Add G to stack.
Mark G visited.
Add E to stack.
Mark E visited.

Node	A	B	C	D	E	F	G
Visited	0	0	0	0	1	1	1

Stack	F	G	E				

4. RecursiveFunction(Node D)

Node D unvisited:
One adjacent node:
Node E visited.
Add D to stack.
Mark D visited.

Node	A	B	C	D	E	F	G
Visited	0	0	0	1	1	1	1

Stack	F	G	E	D			

5. RecursiveFunction(Node A)

Node A unvisited:
Two adjacent nodes:
Node B unvisited:
One adjacent node:
Node D visited.
Add B to stack.
Mark B visited.
Node C unvisited:
One adjacent node:
Node D visited.
Add C to stack.
Mark C visited.
Add A to stack.
Mark A visited.

Node	A	B	C	D	E	F	G
Visited	1	1	1	1	1	1	1

Stack	F	G	E	D	B	C	A

6. RecursiveFunction(Node B)

Node B visited.

Node	A	B	C	D	E	F	G
Visited	1	1	1	1	1	1	1

Stack	F	G	E	D	B	C	A

7. RecursiveFunction(Node C)

Node C visited.

Node	A	B	C	D	E	F	G
Visited	1	1	1	1	1	1	1

Stack	F	G	E	D	B	C	A

8. RecursiveFunction(Node G)

Node G visited.

Node	A	B	C	D	E	F	G
Visited	1	1	1	1	1	1	1

Stack	F	G	E	D	B	C	A

9. All nodes sorted.
Reverse stack.

Stack	A	C	B	D	E	G	F

Figure 22: Step by step example of the topological sort algorithm applied to the graph shown in figure 20.

7 Implementation

7.1 Overview

This project is both a vehicle simulation and a visual and interactive game. The simulation was implemented in the programming language C++. C++ was chosen for its speed and good object orientated features, something that this project made extensive use of. Additionally, C++ is the native language for the game engine Unreal Engine 4 (UE4), which was chosen to build the game elements. UE4 is a powerful game engine widely used in the games industry. Game engines greatly reduce the amount of work required to create games by taking care of many tasks that are common between games. Game logic was implemented in UE4 using the inbuilt node based visual scripting language, blueprints. This is a powerful yet simple means to program complicated behaviour.

7.2 Project Structure

The project was built in four separate software subprojects. The dependencies of these subprojects are shown in figure 23. The core block diagram simulation framework and common component blocks were implemented as a static C++ library named SimFramework. The open source C++ library Eigen 3 [25] was used for matrix vector calculations. The vehicle model was then implemented in a second static C++ library named SimModels, built using SimFramework. The third party C++ library nlohmann/json [26] was used to parse json files, used for the road and engine blocks. The SimModels library was wrapped into a UE4 plugin named SimInterface to provide an interface for the UE4 blueprint language. Finally, the Eco Academy game was built in UE4 using the SimInterface plugin. This subproject approach was taken to follow the separation of concerns principle in software development; each subproject handles a specific task and so can be tested for that specific task only. All code for the project was developed using the version control system Git and is hosted on the platform GitHub. Table 1 lists the URLs for each subproject repository. SimModels was developed in the same repository as SimFramework for convenience.

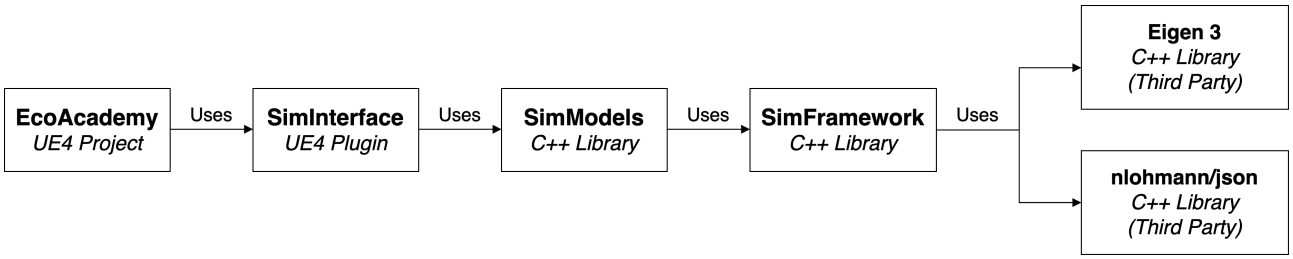


Figure 23: Diagram of subprojects subsystem block diagram.

Project	GitHub URL
SimFramework	`https://github.com/tomjhunt13/SimFramework`
SimModels	`https://github.com/tomjhunt13/SimFramework`
SimInterface	`https://github.com/tomjhunt13/SimInterface`
EcoAcademy	`https://github.com/tomjhunt13/EcoAcademy`

Table 1: Location of code and files for each subproject.

Figure 24 is a high level flow diagram of the main game loop showing the responsibilities of each of the project components. The outermost game component presents the user with a visual and interactive interface. It continually runs the game loop until the user exits or the game ends. Each iteration of the game loop takes dt seconds to run. The game component polls user input and passes it to the plugin component. The plugin directly writes to the input blocks in the simulation and reads from the output blocks. The simulation then advances forward in time by dt seconds. To ensure numerical stability, the simulation can take a number of smaller sub-steps to reach dt seconds.

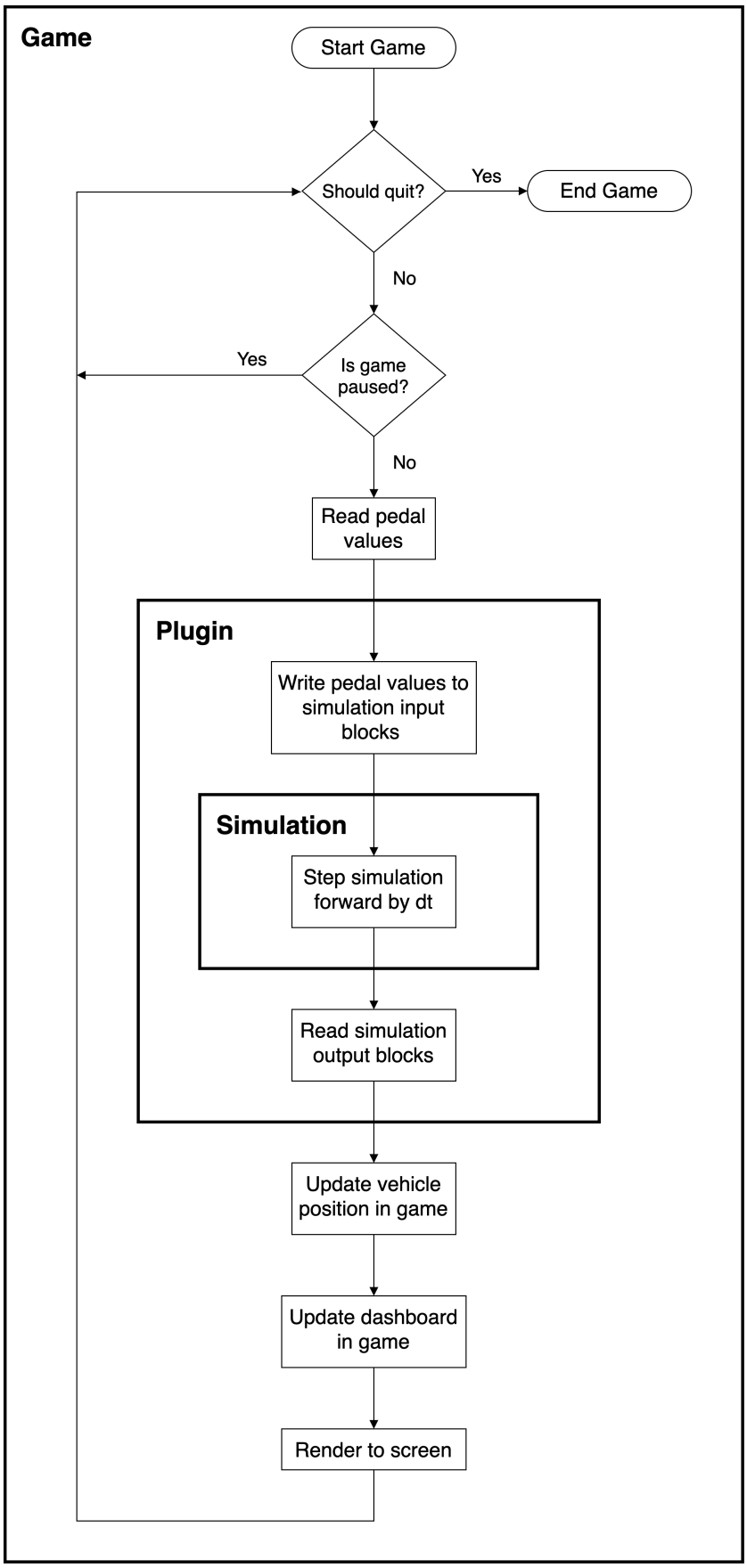


Figure 24: High level overview of how the simulation was integrated into a game.

7.3 Game Design

A simple graphical user interface was made as a starting point for a more complex game. When the user starts the game they are presented with the screen shown in figure 25. Clicking on the screen gives the user control over the input pedal position and begins the simulation. A single input pedal was chosen for simplicity, consisting of a brake section in red, dead-zone in grey and throttle section in green. Moving the mouse forward increases the position of the pedal and moving the mouse backwards decreases the position of the pedal. A picture of the game during play is shown in figure 25. Here the dashboard, which provides numerical feedback to the user, can be seen. Presented on the dashboard is: engine speed, vehicle speed, current gear, input pedal position, instantaneous fuel efficiency and average fuel efficiency.

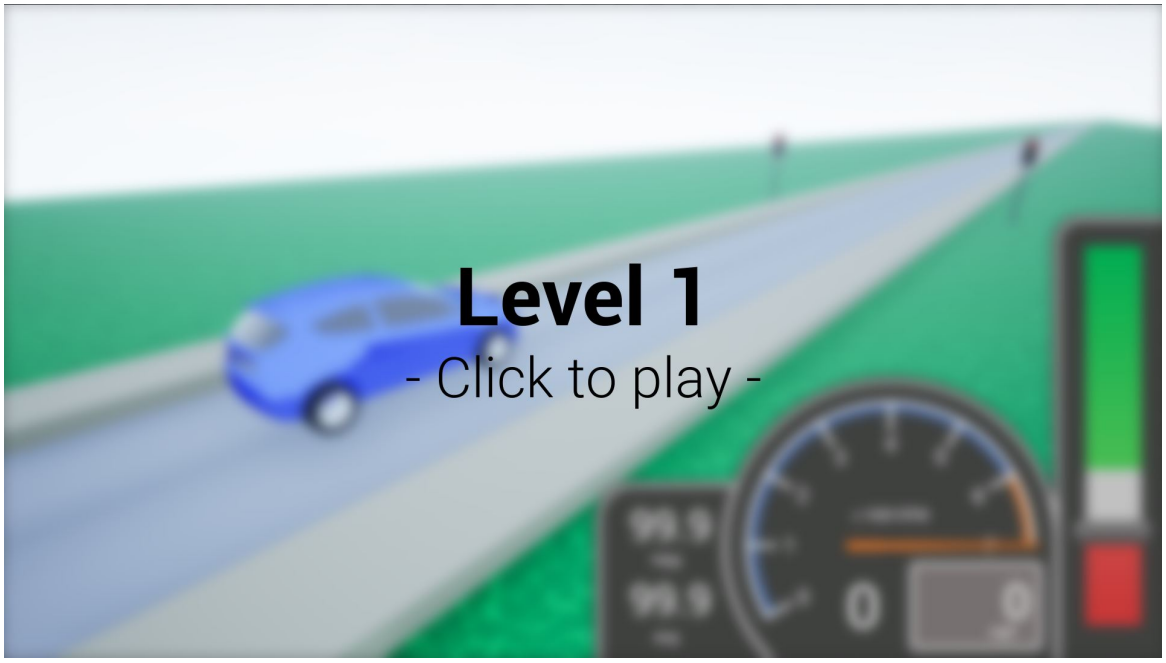


Figure 25: Picture of the game when it is first started.

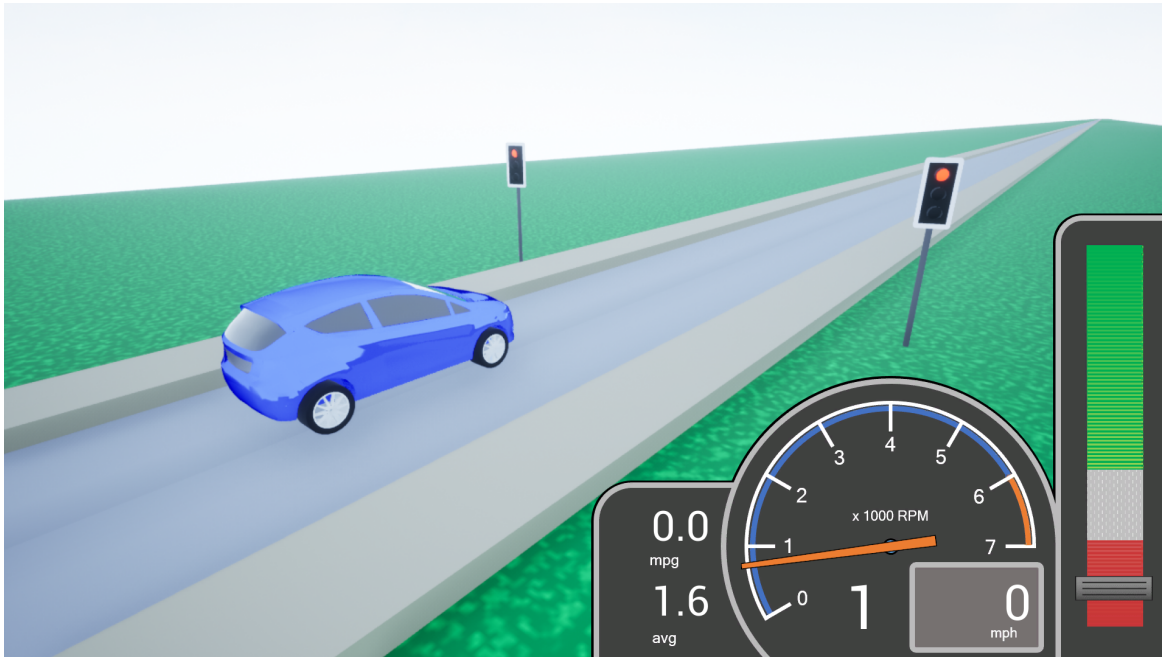


Figure 26: Picture of the game during play.

8 Model Validation

8.1 Overview

It is important to perform validation tests with any simulation to assess whether or not it is performing as anticipated. Validation tests were performed first for the clutch and transmission and vehicle dynamics subsystems by themselves and then for the full model.

8.2 Clutch and Transmission Subsystem

The clutch and transmission subsystem is the most complicated in the vehicle model due to the two state behaviour. The subsystem was tested in isolation with parameters: $I_e = I_w = 1$, $b_e = b_w = 0.05$ and $G = 2$. Initially the engine speed was 100 rad / s and the wheel speed was 300 rad / s. No engine torque was applied to model for $0 \leq t \leq 20$. For $t > 20$, an engine torque of 10 Nm was applied. The clutch pedal was set to be fully down, $\gamma = 0$, for $0 \leq t \leq 5$ and $t > 30$ and fully up, $\gamma = 1$, for the rest of the simulation. Figure 27 shows the engine speed, ω_e, wheel side clutch plate speed, ω_t and wheel speed, ω_w, during the simulation. At all times $\omega_w = \frac{1}{2}\omega_t = \frac{1}{G}\omega_t$, validating the transmission aspect of the subsystem. Initially all speeds fall due to viscous friction. When the clutch pedal is first set to fully up, the clutch is still in the unlocked state as the plate speeds do not match. The wheel side clutch plate exerts a torque on the engine side clutch plate causing ω_e to increase and ω_t and ω_w to decrease. As soon as the speeds match the condition for transitioning to the locked state is met and the two plates began to move as one. The two plates accelerated at the same rate when the engine torque was applied. When the clutch pedal is next set to fully down, the torque applied is greater than the torque capacity of the clutch and the subsystem returns to the unlocked state. The engine side plate continues to accelerate under the applied torque and the wheel starts decelerating under viscous friction.

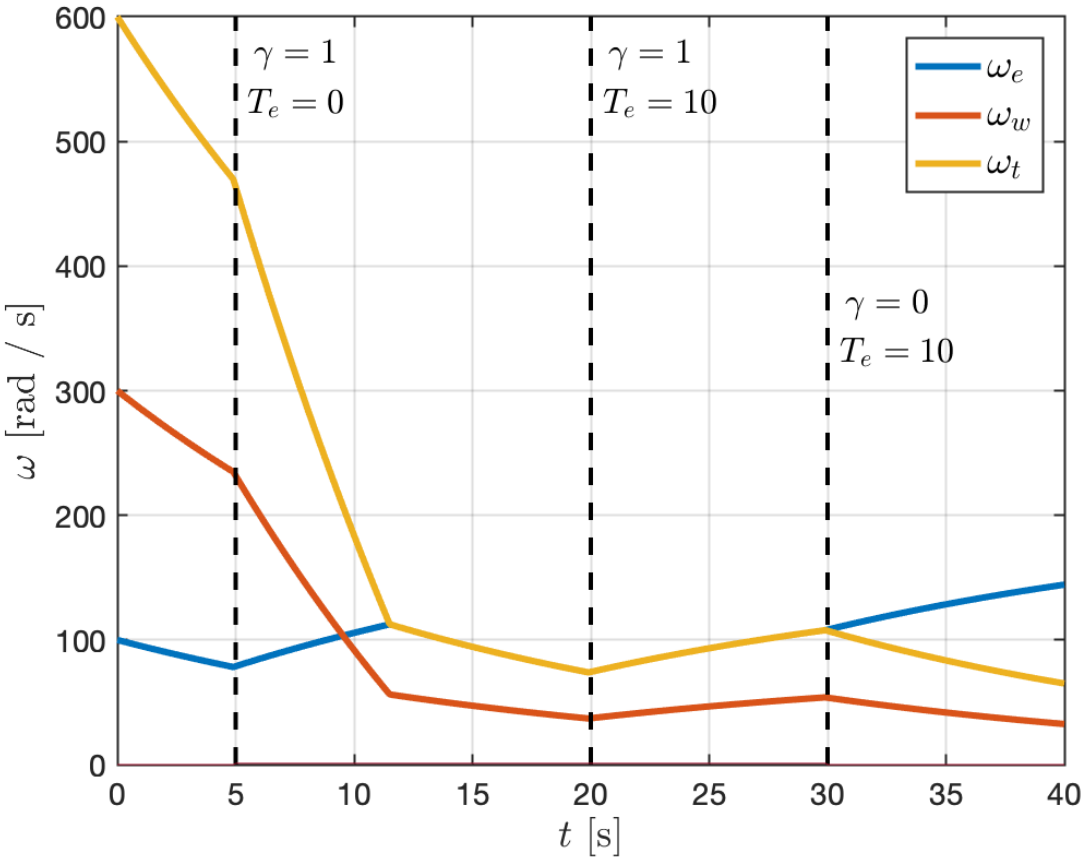


Figure 27: Engine speed, ω_e, wheel side clutch plate speed, ω_t and wheel speed, ω_w, during clutch and transmission validation test.

8.3 Vehicle Dynamics

Validation tests were performed on the vehicle dynamics subsystem to ensure correct implementation. In the first test, a constant tyre force of 10000 N was applied for 200 seconds followed by 200 seconds with zero tyre force. The test was performed on a flat road for five different masses of vehicle: 500 kg, 1000 kg, 1500 kg, 2000 kg and 2500 kg. The other vehicle parameters for this test were: $A = 2.5$ m^2, $C_d = 0.3$ and $C_{\text{RR}} = 0.015$. Figure 28 shows the velocity trace for each vehicle mass. As expected from Newtons second law, decreasing the vehicle mass increased the initial acceleration rate and reduced the time taken to reach terminal velocity. The terminal velocity increased as mass decreased due to the reduced rolling resistance. When the force was removed the high aerodynamic drag quickly slowed the vehicles. Again, the lower the mass, the faster the change in velocity.

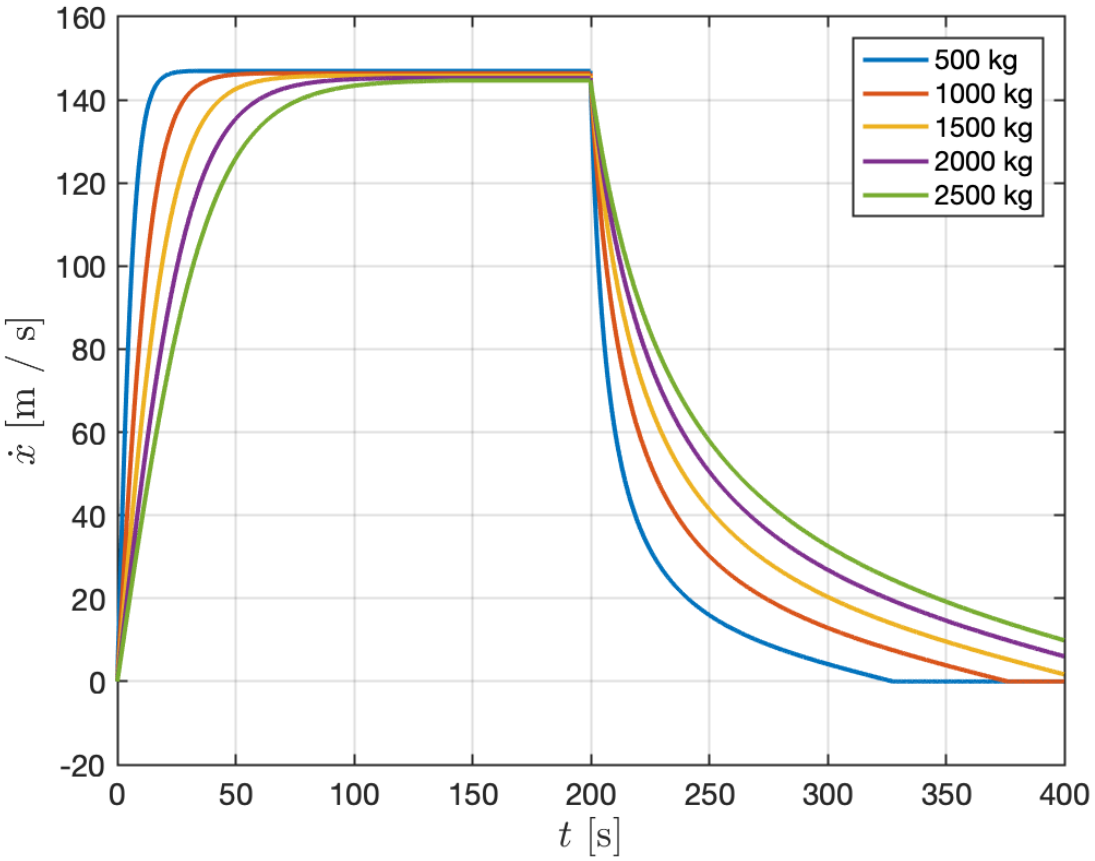


Figure 28: Velocity traces for the inertial validation test.

The second vehicle dynamics validation case tested the aerodynamic drag implementation. Again, a constant tyre force of 10000 N was applied for 200 seconds followed by 200 seconds with zero tyre force. The road was flat throughout. Five different aerodynamic coefficients, $\rho A C_d$, were tested: 0.25, 0.5, 1, 2, 4. The coefficient of rolling resistance, C_{RR} was set to zero so that aerodynamic drag was the only resistance. The mass was set as 1500 kg. In the steady state with no rolling resistance or gravity force, the vehicle dynamics equation of motion, equation 10, can be rearranged to give the steady state, terminal, velocity of the vehicle. This is given in equation 24. Figure 29 shows the velocity trace for each drag level. When the vehicle is at rest there is no aerodynamic drag and hence the acceleration is same for each trace. It can be seen that the terminal speeds for each trace match their expected values calculated using equation 24. The deceleration at the moment the tyre force was removed was the same for each drag level; equal to the tyre force that was removed.

$$\dot{x} = \sqrt{\frac{2F_w}{\rho A C_d}} \tag{24}$$

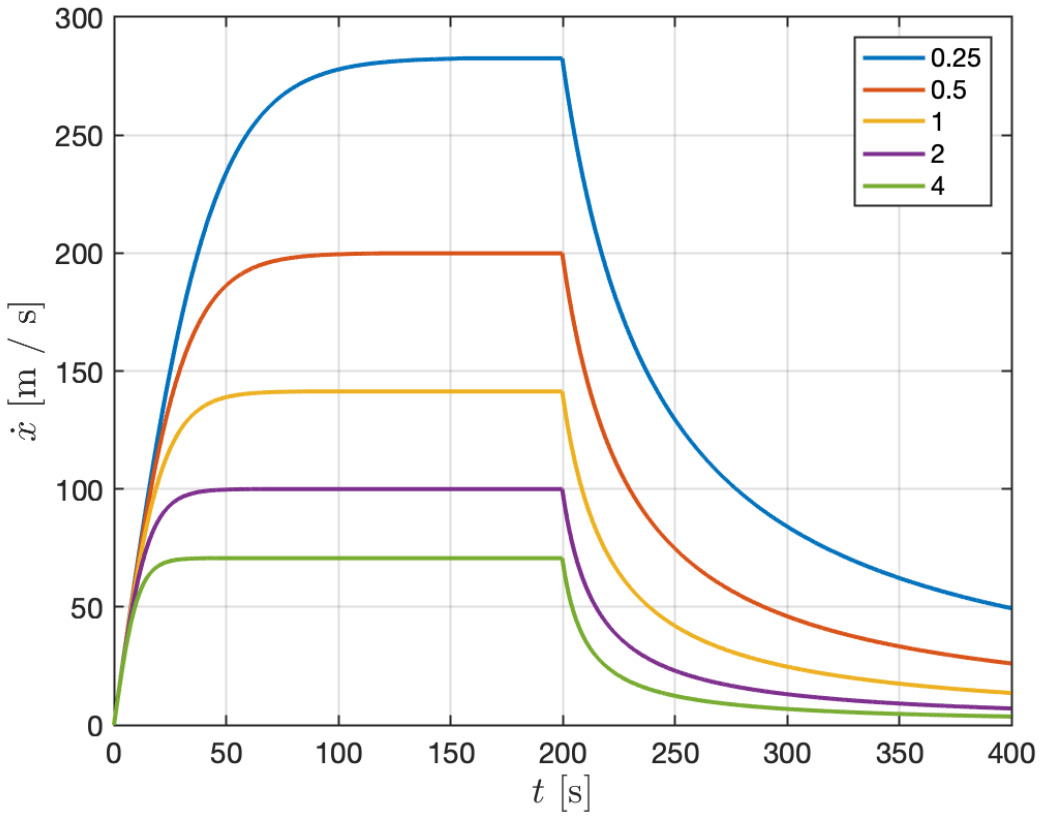


Figure 29: Velocity traces for the aerodynamic drag validation test.

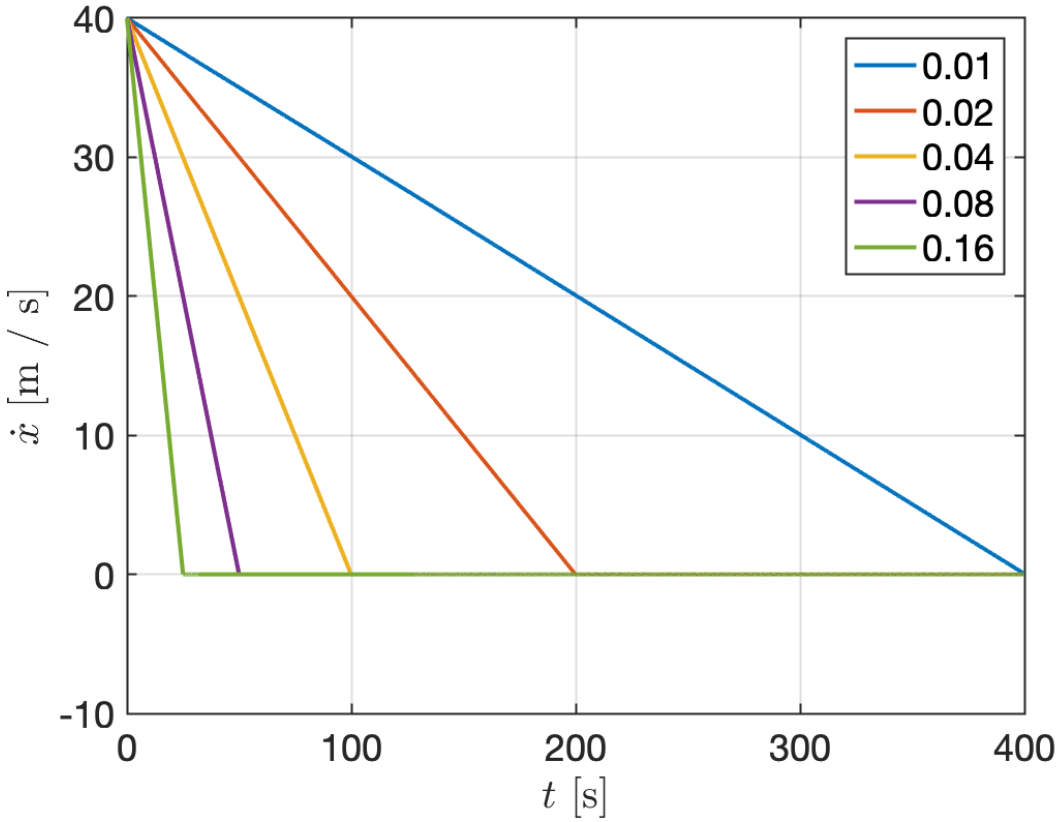


Figure 30: Velocity traces for the rolling resistance validation test.

Next rolling resistance in the vehicle dynamics subsystem was tested. In this test the vehicle had an initial speed of 40 m / s and was left to decelerate on a flat road under only rolling resistance with no tyre force applied. Five different coefficients of rolling resistance were tested: 0.01, 0.02, 0.04, 0.08, 0.16. For this test the constant g was set to 10 m / s^2. The other vehicle parameters were: $m = 1000$ kg and $\rho A C_d = 0$. Figure 30 shows velocity traces for each simulation. Rolling resistance provides a constant force meaning deceleration should be constant. This is seen in the traces. Doubling the coefficient should double the force and so half the acceleration. Again, this is seen in the traces. The vehicle does not continue decelerating past 0 m / s, showing that rolling resistance is correctly signed as implemented.

The final validation test case for the vehicle dynamics subsystem tested the implementation of gravity. The parameters for the test were: $\rho A C_d = C_{\text{RR}} = 0$, $m = 1000$ kg and $g = 10$ m / s^2. Initially no tyre force was applied and the road gradient was set to -0.1 radians for 20 seconds. The gradient was then set to 0 radians for a further 20 seconds. Next, the gradient was set to 0.1 radians for the remainder of the test. For the final 20 seconds a tyre force was applied: $F_w = mg\sin(0.1)$. Figure 31 shows the velocity trace for the test. When the road gradient is constant the force from gravity acting tangentially is constant and so the acceleration is constant. This is seen for the first 20 seconds and from 40 seconds to 80 seconds. When the gradient is zero there is no force on the vehicle and so no acceleration, seen between 20 and 40 seconds. At 60 seconds the velocity of the vehicle is zero. This shows the simulation has conserved energy. When the tyre force is applied at 80 seconds, the net tangential force on the vehicle is zero. This is seen in the trace as there is no acceleration.

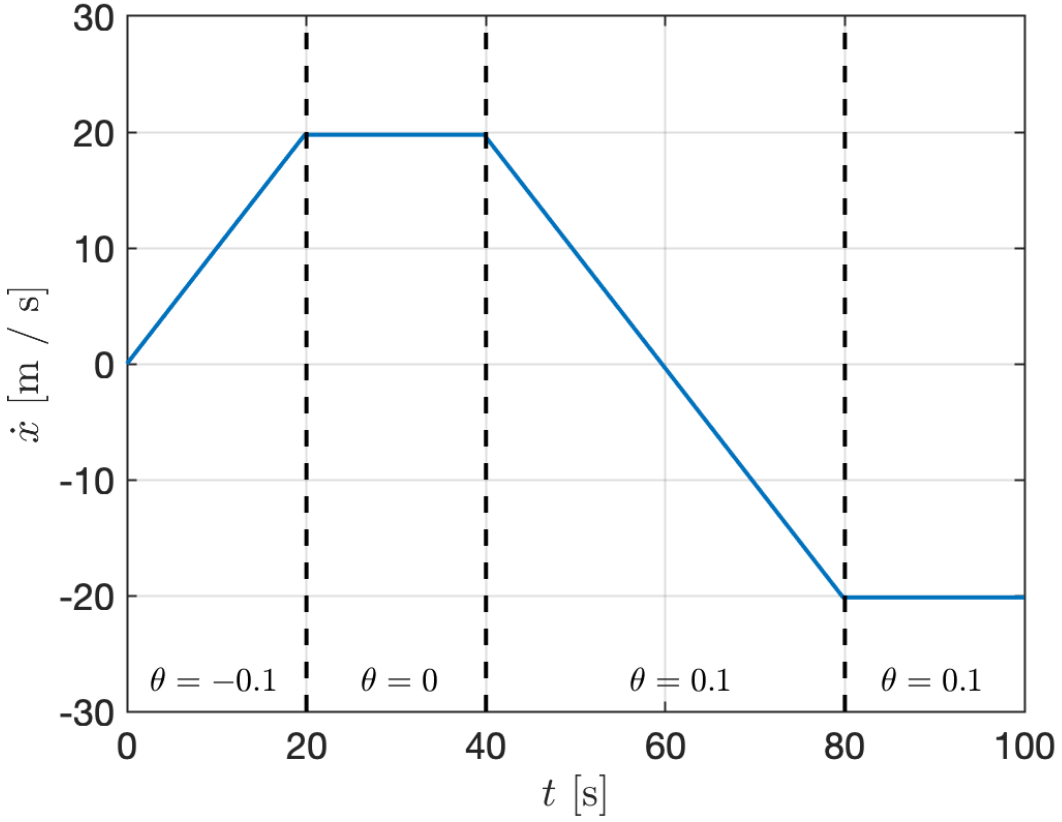


Figure 31: Velocity trace for the gravity validation test.

8.4 Full Vehicle

Validation tests were performed for the full vehicle model. In the first test the vehicle was accelerated at full throttle from a standstill to terminal velocity in the highest gear and then allowed to slow down at 25 % throttle to first gear. This was performed for three different sets of gear ratios, given in table 2. The short set ratios are 20 % higher than the base set ratios which are 20 % higher than the long set ratios. Each simulation used the same vehicle parameters, given in table 3. Figure 33 shows traces of velocity, $\dot{x}$, engine speed, ω_e, and gear ratio i_G for each simulation.

The velocity plot shows that reducing the gear ratios increased the terminal velocity achieved. The torque delivered by the engine map drops significantly for speeds above approximately 6000 rev / min. A lower gear ratio reduces the engine speed for a given wheel speed, allowing it to remain below the 6000 rev / min threshold at a higher vehicle speed. However, reducing the ratio reduces the torque exerted on the wheel, limiting the ability of the vehicle to achieve a higher speed. This is not an issue for low speeds as the drag load is small and the engine capably accelerated to the 6000 rev / min threshold for each ratio set. The engine speed plot shows that the engine could not accelerate up to 6000 rev / min in gears 4 and 5, with the long gear set reaching a lower steady state engine speed. Despite the lower engine speed, the gear ratio was small enough such that the terminal vehicle speed was higher.

The engine viscous friction modelled in the clutch and transmission subsystem exerts a resistive torque proportional to the engine speed. At high engine speeds with the clutch in the locked state, this acts to slow the vehicle. This engine braking phenomena is seen in this validation test when the vehicle shifts from gear 5 to gear 4 at terminal speed. The engine rapidly accelerated up to a speed higher than the 6000 rev / min threshold and the vehicle speed sharply fell.

Ratio	**Short**	**Base**	**Long**
1	14.4	12	10
2	9.6	8	6.7
3	6.4	5.3	4.4
4	4.3	3.6	3.0
5	2.8	2.4	2.0

Table 2: Sets of gear ratios used for the first full vehicle model validation test.

Parameter	**Value**	**Units**
m	1500	kg
C_d	0.3	-
A	3	m^2
C_{RR}	0.015	-
I_e	1	kg m^2
I_w	2	kg m^2
b_e	0.4	Nm sec / rad
b_w	0.1	Nm sec / rad

Table 3: Vehicle parameters used for the first full vehicle model validation test.

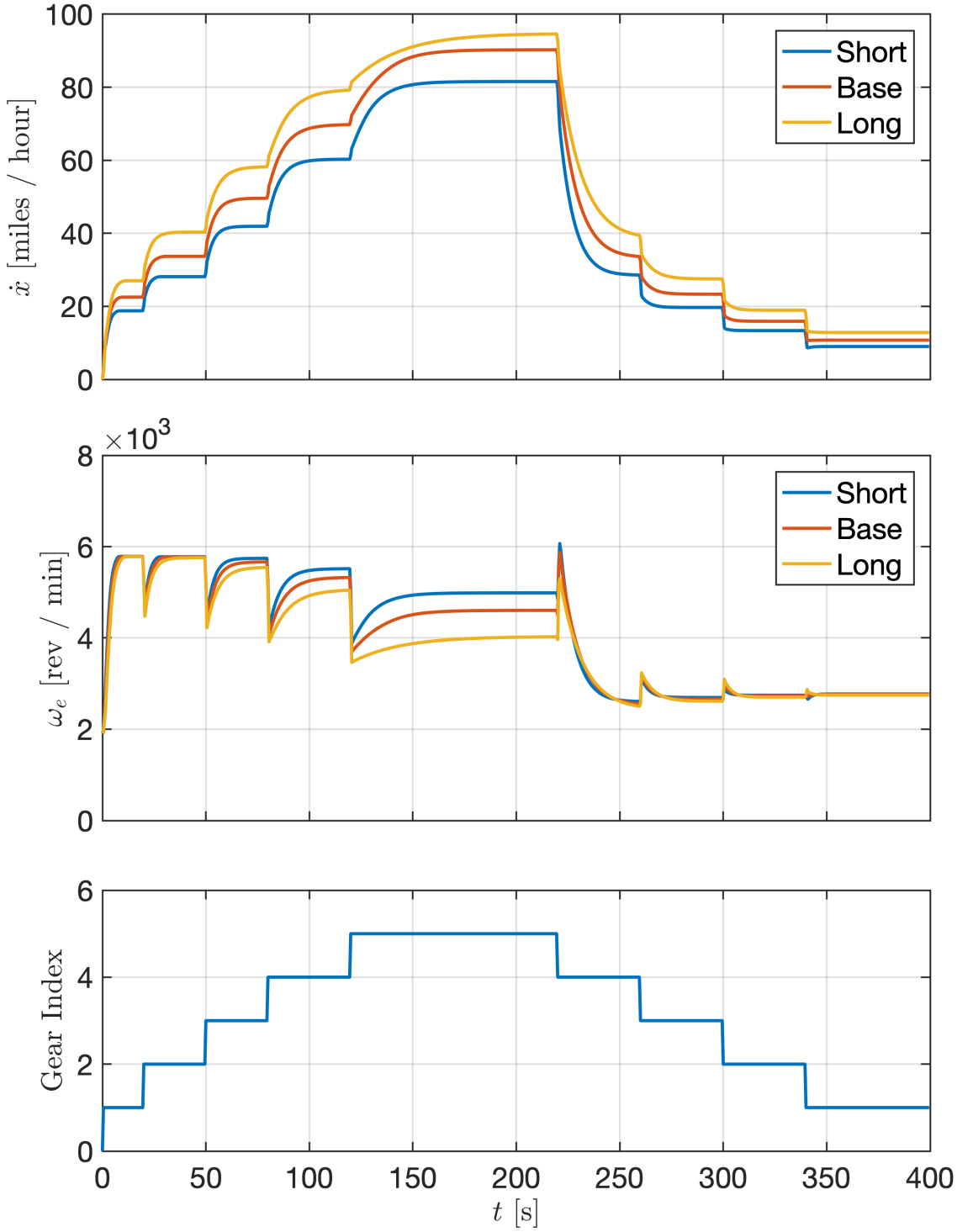


Figure 32: Velocity, $\dot{x}$, engine speed, ω_e, and gear ratio i_G for the first full model validation case.

The focus of the Eco Academy game is to optimise vehicle parameters and driving style in order to minimise fuel consumption. Therefore, it is key that the vehicle model can produce a reasonable approximation of fuel consumption. A simple validation test was performed to verify that the model qualitatively does this. The vehicle was driven to a series of steady state velocities in different gears and the instantaneous efficiency, η, was computed using equation 25. Here ρ is the fuel density. The value of this was artificially set in order to give a fuel efficiency similar to that that a modern passenger vehicle could provide. This was deemed justified as the learning outcome of the game is unaffected.

$$\eta = \frac{\dot{x}\rho}{\dot{f}} \tag{25}$$

The gear ratios and vehicle parameters used for this test are shown in tables 4 and 5 respectively. Figure 33 is a plot of the validation test results. The fuel flow rate increases with engine speed and throttle position. Hence, reducing these will reduce the fuel consumption. The engine speed can be reduced by driving at a lower speed, or by reducing the gear ratio as previously discussed. Therefore, fuel consumption is reduced in higher gears for the same vehicle speed. This is seen for every speed in the test. As vehicle speed increases, the viscous and aerodynamic drag forces increase, requiring more power to maintain speed. In the validation test, increasing speed increased fuel usage in every gear as predicted. Low vehicle speeds cannot be achieved in higher gears as the corresponding engine speed would be below the stable operating speed of the engine. Additionally, the torque output of the engine at low engine speeds is reduced. This challenges the user of the game to compromise between an efficient gear ratio choice and safe driving technique.

Ratio	Value
1	12
2	7.5
3	5
4	4
5	3
6	2

Table 4: Sets of gear ratios used for the second full vehicle model validation test.

Parameter	Value	Units
m	1200	kg
C_d	0.3	-
A	2	m^2
C_{RR}	0.015	-
I_e	0.1	kg m^2
I_w	3	kg m^2
b_e	0.4	Nm sec / rad
b_w	0.05	Nm sec / rad

Table 5: Vehicle parameters used for the second full vehicle model validation test.

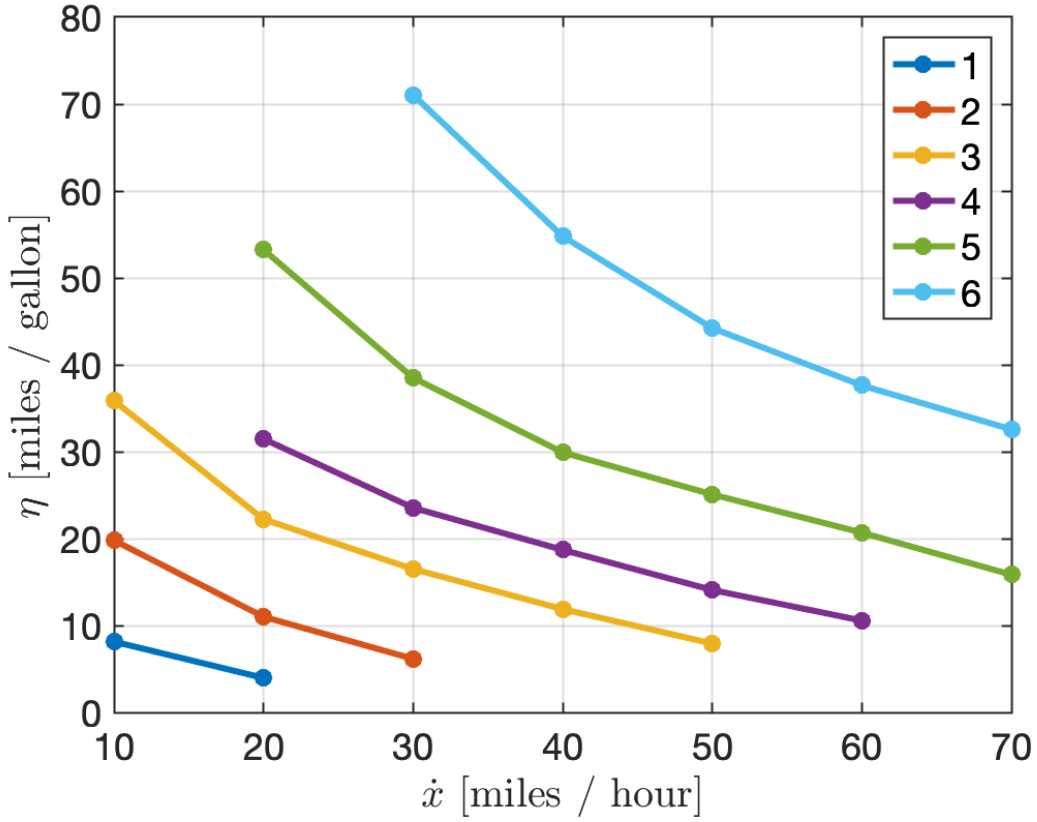


Figure 33: Fuel efficiency with speed and gear ratio.

9 Conclusion

This report documented the process of creating an educational video game based on minimising the fuel consumption of a vehicle in realistic driving conditions. A literature review was performed to determine which models of important vehicle subsystems were most appropriate for this application. It was decided that one-dimensional vehicle behaviour was adequate for conveying the learning objectives of the game. Static engine maps were used to approximate engine torque and fuel flow rate. A state based clutch model allowed real-time simulation of clutch behaviour. The simple Magic Formula tyre model was chosen to determine longitudinal tyre force. A one-dimensional point mass model for vehicle dynamics was used for simplicity. To simulate the vehicle model, a block diagram simulation framework was created in the C++ programming language. A key feature of the framework was the topological sorting algorithm used to determine the block update order. The vehicle model was implemented in this framework and then integrated into a plugin for the Unreal Engine 4 game engine. This allowed interactions between the simulation and the game. A simple graphical user interface was created for the game with intuitive control input. The clutch and vehicle dynamics subsystems were validated in isolation to ensure their implementation was correct. In all tests the subsystems delivered the expected results. Two tests were then performed on the full vehicle model. In the first test, the vehicle was accelerated to terminal velocity in the highest gear, down to terminal velocity in the lowest gear. This was performed for three different sets of gear ratios. As predicted, the lower ratios enabled a higher terminal velocity but at a lower rate of acceleration. Engine braking behaviour was also observed. The second test accelerated the vehicle to steady state in a range of speeds and in different gears and recorded the fuel efficiency. It was seen that for a given speed, higher gears increased fuel efficiency. Additionally, fuel efficiency in a given gear increased as the velocity reduced. The absolute efficiency numbers were artificially scaled to fall in a reasonable range, but as this does not affect the learning outcomes of the game this was considered justified. This project met all objectives and created a useful prototype for the Eco Academy game.

10 Future Work

The outcome of this project was a working vehicle simulation integrated into a simple, graphical game. This serves as prototype for the Eco Academy game. However, in this state, the application is not structured in a way to create an engaging educational experience. The premise of the game was set in realistic driving scenarios. Hence, items such as traffic lights, other road users and speed limits should be replicated in the game in order to improve the realism and challenge. Additionally, the visual representation of the driving environment is currently very plain. Items such as buildings and trees in the background would significantly improve the immersion of the experience. Finally, a menu system must be made to present an interface for changing vehicle parameters, unlocking new levels and viewing results.

References

[1] R. Sandford and B. Williamson, "Racing academy: A futurelab prototype research report," *National Foundation for Educational Research*, Dec 2004, accessed on: April 30, 2020. [Online]. Available: https://www.nfer.ac.uk/racing-academy-a-futurelab-prototype-research-report/

[2] I. Mireles, "Racing academy 1.05," *software.informer*, 2019, accessed on: April 30, 2020. [Online]. Available: https://racing-academy.software.informer.com

[3] P. Dekraker, D. Barba, A. Moskalik, and K. Butters, "Constructing engine maps for full vehicle simulation modeling," in *WCX World Congress Experience*. SAE International, apr 2018. [Online]. Available: https://doi.org/10.4271/2018-01-1412

[4] G. Salemme, E. Dykes, D. Kieffer, M. Howenstein, M. Hunkler, and M. Narula, "An engine and powertrain mapping approach for simulation of vehicle co2 emissions," sep 2015. [Online]. Available: https://doi.org/10.4271/2015-01-2777

[5] Prepared for EPA by Ricardo Inc. and Systems Research and Applications Corporation (SRA), "Computer simulation of light-duty vehicle technologies for greenhouse gas emission reduction in the 2020–2025 timeframe," *United States Environmental Protection Agency*, 12 2011.

[6] Z. Gao, J. C. Conklin, C. S. Daw, and V. K. Chakravarthy, "A proposed methodology for estimating transient engine-out temperature and emissions from steady-state maps," *International Journal of Engine Research*, vol. 11, no. 2, pp. 137–151, 2010. [Online]. Available: https://doi.org/10.1243/14680874JER05609

[7] J. D. Bishop, M. E. Stettler, N. Molden, and A. M. Boies, "Engine maps of fuel use and emissions from transient driving cycles," *Applied Energy*, vol. 183, pp. 202 – 217, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261916312843

[8] S. M. Aithal and P. Balaprakash, "Maltese: Large-scale simulation-driven machine learning for transient driving cycles," in *High Performance Computing*, M. Weiland, G. Juckeland, C. Trinitis, and P. Sadayappan, Eds. Cham: Springer International Publishing, 2019, pp. 186–205.

[9] C. Yu, M. Seslija, G. Brownbridge, S. Mosbach, M. Kraft, M. Parsi, M. Davis, V. Page, and A. Bhave, "Deep kernel learning approach to engine emissions modelling," *Cambridge Centre for Computational Chemical Engineering*, 2019. [Online]. Available: https://como.ceb.cam.ac.uk/media/preprints/c4e-preprint-243.pdf

[10] J. Martínez-Morales, E. Palacios, and G. V. Carrillo, "Modeling of internal combustion engine emissions by lolimot algorithm," *Procedia Technology*, vol. 3, pp. 251 – 258, 2012, the 2012 Iberoamerican Conference on Electronics Engineering and Computer Science. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212017312002563

[11] Y. Zhang, *Automotive power transmission systems*. Hoboken, NJ: Wiley, 2018.

[12] M. Băţăuş, A. Maciac, M. Oprean, and N. Vasiliu, "Automotive clutch models for real time simulation," *Proceedings of the Romanian Academy Series A - Mathematics Physics Technical Sciences Information Science*, vol. 12, 04 2011.

[13] E. L. Duque, M. A. Barreto, and A. d. T. Fleury, "Use of different friction models on the automotive clutch energy simulation during vehicle launch," *IOP Conference Series: Materials Science and Engineering*, vol. 5, pp. 1376–1389, 2012.

[14] O. I. Abdullah and J. Schlattmann, "Contact analysis of a dry friction clutch system," *ISRN Mechanical Engineering*, 2013.

[15] H. N. Quang, "Dynamic modelling of friction clutches and application of this model in simulation of drive systems," *Periodica Polytechnica Mechanical Engineering*, 1998.

[16] A. Serrarens, M. Dassen, and M. Steinbuch, "Simulation and control of an automotive dry clutch," in *Proceedings of the American Control Conference*, 2004.

[17] M. Dassen, "Modelling and control of automotive clutch systems," *Technische Universiteit Eindhoven*, Jul 2003.

[18] M. Bachinger, M. Stolz, and M. Horn, "Fixed step clutch modeling and simulation for automotive real-time applications," in *Proceedings of the American Control Conference*, 2014.

[19] E. Bakker, L. Nyborg, and H. B. Pacejka, "Tyre modelling for use in vehicle dynamics studies," in *SAE Technical Paper*. SAE International, 02 1987. [Online]. Available: https://doi.org/10.4271/870421

[20] H. B. Pacejka, "Chapter 4 - semi-empirical tire models," in *Tire and Vehicle Dynamics (Third Edition)*, third edition ed., H. B. Pacejka, Ed. Oxford: Butterworth-Heinemann, 2012, pp. 149 – 209. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080970165000048

[21] E. Kuiper and J. J. M. V. Oosten, "The pac2002 advanced handling tire model," *Vehicle System Dynamics*, vol. 45, no. sup1, pp. 153–167, 2007. [Online]. Available: https://doi.org/10.1080/00423110701773893

[22] G. Gim, Y. Choi, and S. Kim, "A semiphysical tyre model for vehicle dynamics analysis of handling and braking," *Vehicle System Dynamics*, vol. 43, no. sup1, pp. 267–280, 2005. [Online]. Available: https://doi.org/10.1080/00423110500140849

[23] P. Zegelaar, "The dynamic response of tyres to brake torque variations and road unevennesses," Ph.D. dissertation, Delft University of Technology, Netherlands, 3 1998.

[24] N. Hayato, "Rwd vs awd in drag racing," *RacingJunk News*, Sep 2017, accessed on: April 20, 2020. [Online]. Available: https://www.racingjunk.com/news/2017/09/12/rwd-vs-awd-in-drag-racing/

[25] Eigen 3, "Eigen is a c++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms." [Online]. Available: https://gitlab.com/libeigen/eigen

[26] N. Lohmann, "Json for modern c++." [Online]. Available: https://github.com/nlohmann/json