Yuvraj Kumpavat

Supervisor: Benjamin williams

Second supervisor: tom oliver

Physics-Based Vehicle Simulation: Understanding the Mathematical and Mechanical Foundations of Racing Games

GDEV60001 Games development project

Contents

[Abstract 2](#_Toc191371794)

[Introduction 3](#_Toc191371795)

[Background / Related Work 5](#_Toc191371796)

[Literature Review 6](#_Toc191371797)

[Basic Forces acting on a car when its racing 7](#_Toc191371798)

[Tyre Physics 10](#_Toc191371799)

[Suspension Dynamics 16](#_Toc191371800)

[**Geometry:** 16](#_Toc191371801)

[**Different types of suspension used in motorsport/sim racing:** 19](#_Toc191371802)

[Engine and Powertrain 23](#_Toc191371803)

[Collisions 31](#_Toc191371804)

[Research Methodologies: 33](#_Toc191371805)

[Results and Analysis: 38](#_Toc191371806)

[Conclusion 42](#_Toc191371807)

[Recommendations 43](#_Toc191371808)

[References 45](#_Toc191371809)

# Abstract

This study compares the performance of Chaos Vehicle Physics, Unreal Engine’s built-in physics system, with a Custom Vehicle Physics developed from scratch to simulate advanced vehicle dynamics, including engine torque distribution, Pacejka’s friction model, different drivetrain model and raycast suspensions. Using Unreal Engine’s profiling tools, both systems were evaluated in terms of total inclusive time and total exclusive time across key metrics such as frames processing, task management, physics calculations on every tick/frame, rendering and memory usage. The results indicates that Chaos outperforms the custom system in each tasks. These advantages are attributed to Chaos’ multi-threaded architecture, optimized memory management, and tight integration with Unreal Engine’s rendering pipeline. While the custom system demonstrates superior accuracy in specific physics calculations, the reliance on sequential processing and inefficient memory allocation leads to higher CPU and GPU overhead. This research provides actionable insights for developers aiming to balance realism and performance in vehicle physics simulations.

# Introduction

Vehicle Physics are a fundamental part of a racing games. Games such as BeamNG.Drive (BeamNG.drive, 2013), Forza Horizon 5 (Forza Horizon, 2021) and Gran Turismo 7 (Polyphony Digital, 2022) are just a few examples which rely on accurate real-time physics models to replicate the motions of virtual vehicles.

In the past, most racing games adopted simplified physics models, favouring accessibility over realism. Classic arcade racing games like Outrun (Horowitz, 2004) and Ridge Racer (Shea, 2017) focused on high-speed thrills and exaggerated drifting, rather than simulating real-world vehicle dynamics. However as gaming hardware improved, developers began implementing more complex physics engine, leading to rise of realistic simulation racing games such as Gran Turismo, iRacing and Assetto Corsa. Looking at the current state of racing games in terms of their distinctive vehicle physics model, BeamNG has created the most sophisticated physics model called Soft-body physics (BeamNG.drive, 2013) which simulates each and every part of the car and giving an incredibly accurate damage model.

Developing a realistic vehicle physics given the constraint of real-time and seamless simulation is inherently difficult problem to solve. Game developers are tasked with simulating tyre dynamics, suspensions and downforce in real time.

Vehicle physics can be divided into several key components:

* Tyre Physics and Grip Mechanics: Friction coefficient, slip angle and slip ratio, load sensitivity, tyre wear and temperature changes.
* Suspension and weight transfer: Body roll and pitch, shock absorption and damping, center of mass and weight distribution.
* Aerodynamics and speed dynamics: Torque, horsepower, drag force, downforce, slipstreaming and drafting techniques.
* Collision and damage models: Scripted collisions, rigid-body physics-based damage and soft-body physics simulation.

It is clear that real-time vehicle physics are pivotal in providing a realistic simulation of vehicle dynamics. As such, many games development studios have created their own vehicle physics engines which are used across several series of games. One example of this is Gran Turismo Concept engine and they have their own dedicated car simulation engineers to fit the company’s need for it. (Tan, 2015)

With highly sophistication of vehicle physics models being made my developers using their in-house engines to solve their problems related to vehicle dynamics or soft-body physics simulation, this dissertation focuses on two research questions:

RQ1: What is the current state of literature regarding vehicle physics simulation, with specific reference to a video games?

RQ2: Can a frontrunning game engine, such as Unreal Engine, be leveraged to provide a realistic real-time vehicle system toolset to aid in the development of racing games?

This is inherently difficult problem to solve. As such, a boilerplate framework which provides a majority of realistic vehicle simulation would find high utility with games developers. The benefits of this area that it would significantly reduce the development time required for implementing vehicle physics, allowing game developers to focus on gameplay mechanics, AI and world design and it is with this motivation in mind that this thesis aims to provide a general understanding of vehicle physics simulation and a toolset to aid suspension dynamics, tyre friction model and vehicle dynamics.

# Background / Related Work

Car physics engine is a big topic especially in the field of video games. There are multiple car physics simulation software available. However, most of the high quality racing simulations are either built in-house by the companies or not free or free, but not for commercial use.

Vehicle Physics Pro (2010 – present) is an advanced vehicle simulation kit for Unity. One must have good knowledge of vehicle mechanics, car tuning and real world set up techniques. This kit supports aerodynamics drag and downforce, driving assists like ABS (Anti-lock Braking System) and TCS (Traction Control System), multiple steering axles and Ackermann geometry, differentials like Open, Locked, LSD (Limited Slip Differentials) and diagnostics like telemetry, graphic telemetry, suspension charts and up to 10 axles (20 wheels) support. (Vehicle Physics Pro, 2010)

Project Chrono (1998 – present) is a multi-physics modelling and simulation infrastructure based on a platform-independent, open-source design. It is a C++ object-oriented library which can be used to perform multi-physics simulations, including multibody and finite element. It was made by professors from University of Parma, Italy and University of Wisconsin-Madison, USA. It is used by US Army Research Office and various robotics companies. Its applications are robotics, wheeled vehicle dynamics, tracked vehicle dynamics, seismic engineering, terramechanics, and much more. (Project Chrono, 1998)

FGear is a paid Unity and Unreal plugin that helps to build arcade or semi-arcade racing physics. The core features are configurable transmissions like auto/sequential/manual (H – pattern), 3D wheel support (multiple raycast/sphere/convex cast), simplified pacejka and MF6.1 tyre model options, detailed telemetry UI and aerodynamics components for drag and downforce effects. (FGear, 2019)

Video games which have realistic physics simulations is: BeamNG.Drive. (Nesvadba, 2023)

BeamNG.drive (2013) is a vehicle simulation vehicle game which features a realistically accurate crash physics which replicates the real-life crashes due to their soft-body physics. The soft-body physics model allows for modelling of stress and deformation on vehicle parts. In BeamNG, springs are called *beams* and mass points are called *nodes*. Nodes have a position in space that is updated dependent on the net forces acting on them and influences the state of the beams. Their engine BeamNG.tech is highly used for ADAS (advanced driver-assistance system), autonomous driving and driving simulation (BeamNG.Drive) (Maul, 2021)

# Literature Review

Physics provides with number of helpful definitions and relations, such as the concept of displacement, velocity and acceleration which allows to describe numerically the motion of any mass.

In Newton’s Second Law of Motion, Newton defined a force as a ‘change in momentum’ of a system and derived an equation: (The Physics Classroom, n.d.)

F = ma

(Force = mass x acceleration)

**Centripetal force**: a force that holds a mass in circular motion with constant radius and tangential speed.

A black and white image of a symbol

AI-generated content may be incorrect.

Figure m (mass), vt(tangential speed), r radius

**Friction:** a force that resists motion between two surfaces, proportional to the normal force between the surfaces and the coefficient of friction (depends on the material).



Figure Fn (Normal force) and mhu (coefficient of friction)

**Work:** a force exerted over a distance.

**Power**: the rate at which work is done.

**Torque**: A twisting equivalent to force. A force applied a given distance *r* from an axis of rotation creates a torque on the system. Think of a tyre iron loosening up a lug nut. The force is applied at a distance away from the nut, creating a torque that loosens it. *T = r x F*

**Aerodynamic Forces**: Drag and lift are the epitome of non-linear phenomena. For the classis problem of a fluid (air or water, for example) flowing over a surface (like a car or an airplane wing) an agreed-upon equation for the force is:

A black and white math equation

AI-generated content may be incorrect.

Figure C (Coefficient of drag), p (density of fluid), v (velocity of the air or car)

## Basic Forces acting on a car when its racing

(Smith, 1999) says that vehicle dynamics deals with how the car responses to driver inputs (steering, throttle, brakes) and external forces (gravity, friction, inertia). Inertia refers to the car’s resistance to changes in its rotational motion. Cars with their mass concentrated near the center (low moment of inertia) are easier to turn and handle better.

Newton’s laws of motion:

These are the fundamental laws that apply to all large things in the universe:

1. Newton’s First Law – A car in straight-line motion at a constant speed will keep such motion until acted on by an external force (like friction, drag, or turning forces). The only reason a car in neutral will not coast forever is friction, an external force, gradually slows the car down. Friction comes from the tyres on the ground and the air flowing over the car. The tendency of a car to keep moving they way it is moving is inertia of the car, and this tendency is concentrated at the centre of gravity point. (Berg, 2022)
2. Newton’s Second Law – When a force is applied to a car, the change in motion is proportional to the force divided by the mass of the car. (Berg, 2022)
3. Newton’s Third law – Every force on a car by another object, such as the ground, is matched by an equal and opposite force on the object by the car. When the driver apply the brakes, he cause the tyres to push forward against the ground, and the ground pushes back. As long as the tyres stay on the car the ground pushing on them slows the car down. (Berg, 2022)

An important effect when accelerating or braking is the effect of dynamic weight transfer. When braking hard the car will nosedive. During accelerating, the car leans back. This is because just like the driver is pushed back in his seat when the pedal hits the metal, so is the car’s center of mass. The effect of this is that the weight on the rear wheels increases during acceleration and the front wheel conversely have less weight to bear. (Monster, 2003)

The effect of weight transfer is important for driving games for two reasons. First of all the visual effect of car pitching in response to driver actions adds a lot of realism to the game. Suddenly, the simulation becomes a lot more lifelike in the user’s experience. Second of all, the weight distribution dramatically affects the maximum traction force per wheel. (Monster, 2003)

A diagram of a car braking

AI-generated content may be incorrect.

Figure G - force of gravity that pulls the car towards the Earth. B - Braking, L - Lift, f- front, r - rear.

Lf is the lift force exerted by the ground on the front tyre, and Lr is the lift force on the rear tyre. These lift forces are as real as the ones that keep an airplane in the air, and they keep the car from falling through the ground to the center of the Earth. It is really hard to notice the forces that the ground exerts on objects because they are so ordinary. The reason is that the magnitude of these forces determine the ability of a tyre to stick, and imbalances between the front and rear lift forces account for understeer and oversteer. Newton’s third law requires that these equal and opposite forces exist, but the main thing is how the ground and the Earth’s gravity affect the car. (Beckman, 1991)

If the rear tyres approach their traction limit more rapidly than the front, then the effect is for the rear of the car to steer a wider path than the front wheels. This rotates the car more than the driver intended and, if nothing is done, leads to the car turning a smaller radius corner. When this occurs the car is said to *oversteer.* (Bower, 1999)

If the front tyre approach the traction limit more rapidly, the effect is that the front of the car takes a wider radius curve than the driver intended. The car is said to *understeer*. (Bower, 1999)

Here’s another figure for reference when dealing with weight transfer in simulation:

A car with a diagram

AI-generated content may be incorrect.

Figure W - (weight on axles), h is the height, c and b is the distance between the rear and front axle to the CM (Centre of Mass)

For a stationary vehicle the total weight of the car (W, which equals M \* g) is distributed over the front and rear wheels according to the distance of the rear and front axle.

Wf = (c/L)\*W Wr = (b/L)\*W

where b is the distance from CG to front axle, c the distance from CG to rear axle and L is the wheelbase. (Monster, 2003)

When a car is cornering at speed, the car’s weight transfer from inside wheel to the outside wheel. There is weight transfer due to lateral acceleration as well as weight transfer introduced by body roll. The weight transfer due to lateral acceleration is determined as follows:

Weight transfer = (Lateral acceleration / g) \* weight \* height of CG / Track width (Srisuchat, 2012)

The amount of weight transfer due to body roll is determined as follows:

Weight transfer = (Height of CG \* sin(body roll angle) / Track width) \* weight (Srisuchat, 2012)

Aerodynamic drag or air resistance acts upon a vehicle by resisting its motion. The more aerodynamic a vehicle is, the less drag/resistance it experiences. This parameter was implemented because, just as in real life, the engine can never reach its full potential due to resistive forces acting upon it. In games, aerodynamic drag was modelled by a force in front of the vehicle pushing back on it.

A wireframe of a race car

AI-generated content may be incorrect.

Figure more external forces acting on a car.

Drag is the result of aerodynamic forces that acts in the longitudinal axis of the car, opposing its movement. This is a crucial element of aerodynamics study, and it is of primary concern in road cars aerodynamic design. It must be overcome by the tractive force generated by the engine. (Santos, 2014)

Lift is the resultant of aerodynamic forces that acts upward. Lift reduces the vertical forces of the car, and its reduction is of primary concern in racing car aerodynamics study. The opposite of lift is downforce, which is the resultant of aerodynamic forces that pushes the car against the ground. (Santos, 2014)

Side Force is the resultant of aerodynamic forces that pushes the car sideways. This force is generated by lateral winds acting upon the vehicle. It is important for stability studies on road cars, but since lateral wind components are relatively small in racing vehicles, its importance is reduced in motorsport environment. (Santos, 2014)

A math equations and formulas

AI-generated content may be incorrect.

Figure classic formula of aerodynamic drag (Bower, 1999)

## Tyre Physics

Tyres are the only point of contact between the car and the road, so understanding their behaviour is critical for optimizing traction, braking and cornering. A tyre isn’t round when it’s supporting the weight of the car: The flat spot where the tyre touches the ground is called contact patch.

(Mazhar, 2008) Wheel friction was modelled as a step function. The friction was constant until the vehicle experienced a lateral acceleration greater than a certain amount. Once the lateral acceleration was greater than the bounding value, the friction coefficient decreased to a lower value. If the lateral acceleration decreases below the bounding value, the friction coefficient increased to its original value. The speed at which the friction oscillated was dependent on the lateral velocity of the vehicle; higher lateral velocities resulted in the vehicle spending more time sliding while lower velocities caused the vehicle to spend more time in traction. (Mazhar, 2008) Consequently, this allowed the vehicle to drift around corners if the friction coefficient on the rear wheels is lower than on the front. The rear of the vehicle would swing out as the car slid, allowing a drift to be completed. The process by which the vehicle enters and exits the drift is as follows: the vehicle enters a turn at high velocity causing slipping to occur. At the start of the turn the fast oscillations of the friction coefficient causes the vehicle to slide, at this point more time is spent sliding than not. Once the vehicle is sliding it beings to lose velocity, after a certain point more time is spent in traction than slipping. Once this occurs the vehicle regains control, allowing the player to exit the drift in a controlled manner. (Mazhar, 2008)

If there were no such thing as grip, cars just would not be able to move at all. The wheels would spin and the driver would not be able to budge the vehicle. Even on a straight road and at steady speed, there is no alternative to grip. This is because a moving vehicle has to deal with natural forces, such as the banking, the slope or the unevenness of the road, or rolling resistance, which are constantly trying to slow the vehicle down or push it off its path. (Michelin, 2001)

Calculating the wheel angular velocity from the car speed is only allowed if the wheel is rolling, in other words if there is no lateral slip between the tyre surface and the road. This is true for the front wheels, but for drive wheels this is typically not true. (Monster, 2003)

Tyres generate forces by sticking and sliding and everything in between. They transmit these forces to the wheels by elastic deformation. The elastic deformation is extremely complex and theoretical computation requires numerical solution of finite-element equations. (Beckman, 1991) However, despite fierce trade secrecy, industry and academia have reached apparent consensus in recent years on a formula that summarizes experimental and theoretical data. It is the Pacejka’s Magic formula which is not a solution to equations but more of a convenient fitting of commonplace mathematical functions to data. It allows one to compute forces at a higher precision but without integrating equations. Therefore, forces can be computed within a reasonable time, say in a real-time simulation program. (Beckman, 1991)

To understand the magic formula, first the definition of its inputs, which include *slip*. Slip is an indirect measure of a fraction of contact patch that is sticking. It is frequently asserted in the literature that a tyre with no slip at all cannot create forces. If there is any friction between the tyre and the surface, there must be slip. (Beckman, 1991)

Pacejka’s curves are a big part of the tyre models (and tyre modelling is about 50% of a car simulator) (Racer, 2000)

They represent the forces that are generated by the tyre as a result of the tyre not following the road properly. Steer the tyre a little, and get slip angle, and this is input into the Pacejka’s Fy formula, giving a sideways force. Press the throttle, and the wheel starts spinning a bit; this gives a different ratio of wheel spin speed vs ground speed, and this gives a forward (longitudinal) force. (Racer, 2000)

#### Inputs and outputs (Racer, 2000)

The curves output tyre forces (lateral/longitudinal) and moments (Mz for aligning moment for example) based on just a few inputs. These inputs are:

1. Slip angle; the difference between the direction the tyre is facing, and its velocity. 0 means the tyre is going straight ahead (no slip). Typical peak slip angle for tyre are around 3-6 degrees.
2. Slip ratio; the spin velocity divided by its actual world velocity. A slip ratio of -1 means full braking lock; a ratio of 0 means the tyre is spinning at the exact same rate as the road is disappearing below it. A slip ratio of 1 means it is spinning.
3. Camber; the angle of the tyre with respect to the surface.
4. Load; the amount of force pressing down on the tyre. Typically each wheel carries around 1/4th of the car’s weight.

A graph of a graph of a graph

AI-generated content may be incorrect.

Figure Pacejka curves typically look like this. The black line is Fx (longitudinal, a result from slip ratio), the red is Fy (lateral) and green line is Mz (aligning moment) (Racer, 2000)

Most of the sim racing games use Pacejka magic formula for the tyre forces. Pros are it is a standard model which a lot of people already understand and common in the high-end racing simulation world. Cons are not too much non-proprietary data available and the standard coefficients don’t all have a direct physically explanation. (Racer, 2000)

Here’s another simple graph to understand slip ratio with respect to longitudinal force (N - Newton)

A graph with a line going down

AI-generated content may be incorrect.

Figure the relationship between longitudinal (forward) force and slip ratio can be described by a curve such as the following. (Monster, 2003)

Note how the force is zero if the wheel rolls, that is slip ratio = 0, and the force is at a peak for a slip ratio of approximately 6% where the longitudinal force slightly exceeds the wheel load. The exact curve shape may vary per tyre, per road surface, per temperature, etc.

The means a wheel grips best with a little bit of slip. Beyond the optimum, the grip decreases. That’s why a wheel spin, impressive as it may seem, doesn’t actually give the best possible acceleration. (Monster, 2003) There is so much slip that the longitudinal force is below its peak value. Decreasing the slip would give more traction and better acceleration.

To get from normalized longitudinal force to actual longitudinal force, multiply by the load on the wheel.

F(long) = f(N, long) \* Fz

Where f(N,long) is the normalized longitudinal force for a given slip ratio and Fz is the load on the tyre. (Monster, 2003)

“Looking beyond graphics; tires is half a racing simulator’s work. The physics model is the other half. The graphics are the other, eh…” (Racer, 2000)

Extrinsic Factors which affects the rubber compound:

1. Compound Temperature: The temperature of the tyre compound affects adhesion by increasing both the conformance and the penetration of peaks and valleys in the road into the contact patch. This also increases the rate of chemical reaction between the tyre rubber and asphalt, but only up to a point, after which the tyre will ‘go-off’ and grip levels reduce.

(Campbell-Brennan, 2020)

1. Inflation Pressure: Due to the flexible nature of tyre rubber, inflation pressure introduces deformation at the contact surface, ranging from a concave profile (low pressure) to a convex profile (high pressure). This affects the surface area of the contact patch. (Campbell-Brennan, 2020) Somewhere in between the two is a flat profile which provides maximum contact area and the optimal adhesion. This is the goal of the dynamicist. Interestingly, sometimes if the teams are struggling for tyre temperature, they boost the tyre pressures which results in a convex profile and a very narrow contact patch. This contact patch heats up quicker, which then radiates throughout the rest of the tyre, increasing its overall temperature. (Campbell-Brennan, 2020) However, it is important to understand the properties of gas in that for a given volume, if the temperature increases so does the pressure – leading to a compounding effect of both reduced adhesion and reduced contact patch area.
2. Track Conditions:

A diagram of smooth surface and smooth surface

AI-generated content may be incorrect.

Figure Surface roughness can be described by micro and macro roughness.

(SkyFall, 2023) this video explains how beamNG created the soft-body physics which has nodes with their own mass and beams to connect those nodes. Every part of the car, especially the tyres also use nodes and beams to simulate the soft-body physics and create dynamic tyre deformation.

The way a tyre receives and exchanged heat during its rolling motion is a very complicated aspect to be modelled. The sources of heat generation in a tyre are several and it is quite difficult to predict the right percentage of relevance of each component. (Preiti, 2018)

The tyre in LFS has a cycle. It starts cold and starts to heat up and will reach peak temperature, and as it starts to wear down the tyre will release dome of the heat and stat to go down in temperature. This cool off effect is greater than in real life and on some tracks the driver must balance the peak and the eventual puncture temperature. Having higher air pressure in the tyres will make the tyre peak temperature lower. Having perfect camber and minimizing slides is also things that can make the tyre last longer and also bring down the peak temperature slightly. (Cyber Racing teams, 2024)

Unity uses a feature called “Wheel Colliders” to model vehicle tyres and it is a robust slip-based tyre model combined with collision detection and wheel physics (Unity 6, 2025) A tyre friction coefficient is determined from a spline fit relating friction coefficient to slip, both laterally and longitudinally. The spline is define by two developer-set points, one for what is referred to extremum, the other as asymptote point. Extremum is the point of maximum friction coefficient and asymptote is the point where the decline in friction coefficient at higher slip values begins to level off. Both points are defined by their corresponding slip and friction values. (Unity 6, 2025)

(GDC, 2021) The developers of Just Cause 4 created their own version of Tyre dynamics for Just Cause 4 to suit the needs of driving in the game. The tyre dynamics in here also follows the same magic formula (Pacejka’s) to measure forces in a machine with varying input parameters and parameterize so mathematical formulae curve -fit forces. They took the values of the real tyre to create its own behaviour since real tyres have undesirable properties for an open world game.

A screenshot of a computer

AI-generated content may be incorrect.

Float longitudinal\_wheel\_speed\_ms = wheel\_contact\_velocity\_relative\_to\_ground.dot(wheel\_forward\_dir); (GDC, 2021)

Float wheel\_slip\_ratio\_SAE = ((wheel\_angular\_velocity \* wheel\_radius) / longitudinal\_wheel\_speed\_ms) – 1.0f; (GDC, 2021)

In a typical situation where the car is cruising at constant speed, the rear wheels will be rotating faster than the front wheels. The front wheels are rolling and therefore have zero slip. One has to calculate their angular velocity by just dividing the car speed by 2 pi times the wheel radius. The rear wheels however are rotating faster and that means the surface of the tyre is slipping with regard to the road surface. This slip cause a friction force in the direction opposing the slip. (Monster, 2003) The friction force will therefore be pointing to the front of the car. In fact, this friction force, this reaction to the wheel spring, is what pushes the car forwards. This friction force is known as the traction or as the longitudinal force. The traction depends on the amount of slip. And the standardised way to express amount of slip is called Slip Ratio. (Monster, 2003)

Friction Circle:

The last fundamental of tyre dynamics that is necessary to understand is that of the friction circle or g-g diagram. The friction circle graphically illustrates the limits of a tyre generating both longitudinal and lateral acceleration simultaneously, and allows understanding of how the vehicle is being driven relative to this. (Campbell-Brennan, 2020)

Here’s a good example of g-g diagram. The small blue dot shows that the car is in the process of braking and taking a left turn hence both the driver and the tyre is experience g-forces along the line.

A circle with a red line and blue dots

Description automatically generated

The g-g diagram resembles an ellipse rather than a perfect circle, a driver cannot expect the level of lateral acceleration generated in pure cornering whilst demanding acceleration/braking and vice versa. (Campbell-Brennan, 2020)

Techniques such as ‘trail=braking’ emerged exactly to maximise the frictional effort of the tyre and ensure the race car is at the ‘edge’ of the friction circle as much as possible.

## Suspension Dynamics

Suspension is a system of springs, shock absorber, and linkages that connects a car to its wheels and allows relative motion between the two. Implementing the suspension system adds a lot of realism to the dynamics of a car. The suspension system adds a lot of realism to the dynamics of the car. The suspension system of a car is very complicated. (Srisuchat, 2012)

The springs apply equal and opposite force to each object. It force follows Hooke’s Law and is a function of the stretched or compressed length of the spring relative to the rest length of the spring and the spring constant of the spring. The spring constant is a quantity that relates the force exerted by the spring to its deflection. The spring equation of Hooke’s Law is:

Fs = Ks (L – r) where Fs – spring force, Ks – spring constant, L – length (stretch or compress) and r – rest length. (Srisuchat, 2012)

The geometry of any wheel suspension system determines the linear and angular paths that the wheel and tyre will follow when it is displaced from its static position – either by the effect of road irregularities on the unsprung mass or by movement of the sprung mass in response to the load transfers produced by accelerations in the various plane. (Smith, 1999)

(Smith, 1999) Early racing cars, like carts and carriages, were built with beam axles at each end. Surprisingly, with some notable but not very successful exceptions, this situation continued until the late 1920s or early 1930s. Very early on it became apparent that the beam axle had inherent limitations which placed very definite limits on vehicular performance. This is not good at all, especially if the road surface should be less than perfect. The beam axle is also very heavy – all unsprung – requires a lot of space, if we are going to have provision for a reasonable amount of vertical wheel travel, calls for some heavy point loadings to be fed into the chassis and has a high roll center – which is why the early race cars didn’t roll much. And that’s how various design and configurations of suspension made so that the team could stay ahead of their competitors or to fit certain racing regulations. (Smith, 1999)

Sim racing comes packed with multiple options of adjustment when it comes to vehicle set up. The ones which is widely used are: Double Wishbone suspension, MacPherson Strut, Pushrod Suspension, Pullrod Suspension, Torsion Bar suspension.

The suspension is responsible for presenting the wheel to the road surface in the desired orientation – the intricacies of which are highly important in the sense of tyre dynamics.

### **Geometry:**

1. Camber – Camber is the angle of wheel when viewed from the front of the car. If the top of the wheel leans in towards the centre of the car, this is called negative (-) camber. If the top of the wheel leans out towards the outside of the car it is called positive (+) camber. If the wheel is stood up perfectly vertically then the wheel has zero camber. Negative camber is the most common setting on a race or track car. (Suspension Secrets, 2020)

With zero camber, the force on the tyres are equally distributed along the contact patch when you’re standing still or driving in a straight line. This increases the available grip under straight line braking and acceleration. Cornering with zero camber causes one side of the tyre to unload, while the other side of the tyres takes more load. This causes unequal load distribution and lowers the overall available grip on the tyre when you need it the most: **while cornering**.

With negative camber, the force distribution along the contact patch is somewhat unequal and while driving in a straight line. However, when cornering forces and carcass deflection come into play, they can negate the effect of negative camber, equalising load distribution along the contact patch. This maximises the available grip on the outside tyres.

“The magnitude of the camber angle depends on the magnitude of the car’s acceleration. Under constant velocity, the wheels tend toward their static position.” (Srisuchat, 2012) For game programming purposes, one could defined a maximum camber angle and compute the current camber from the percentage of the longitudinal weight transfer times the maximum car angle.

A diagram of a diagram of a wheel

AI-generated content may be incorrect.

Figure Camber angle and chassis dynamics

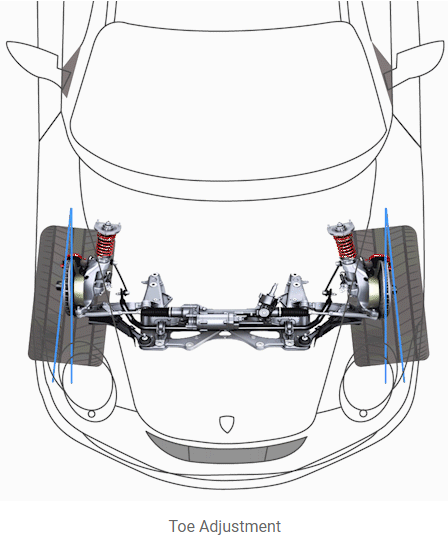
A drawing of a car

AI-generated content may be incorrect.

Figure Camber adjustment (currently the Porsche has negative camber)

1. Toe – Toe is the angle of the wheel when viewed from above. The best way to envisage toe is to think of the wheels steering inboard or outboard of the car. This is toe again. Toe in (or positive toe (+)) is where the front of the wheel points in towards the centre of the car from a top view. Toe out (or negative toe (-)) is when the front of the wheel point out towards the outside of the car from a top view. There are many different ways to set toe on the front and rear wheels and not one rule applies to all cars. (Suspension Secrets, 2020)

Effects of Toe-in and toe-out has more than one effect. If a person rolls a free tyre at an angle, it would want to follow an elliptical trajectory instead of a straight line. In other words: an angled tyre wants to turn. The force that causes this effect is called camber thrust. This results in a bit more friction, heat and water, which can be offset by a toe-out adjustment. He can also use a toe-out adjustments to get the slip angles of the front tyres in a more optimal spot.



1. Caster – Caster angle is only present at the front wheels and is a little more complex than toe or camber. For a MacPherson strut vehicle a line can be drawn down the length of the damper from a side view down to the ground. The angle at which this line leans away from perfectly vertical line is the caster angle. (Suspension Secrets, 2020) For a double wishbone set up a line is drawn from a side view through the upper and lower wishbone points where the upright mounts to them. Again this is drawn to the ground and the angle of this line relative to a perfectly vertical line through the centre of the wheel is the caster angle. If the line leans towards the back of the car it is called positive (+) caster. If the line leans towards the front of the car it is called negative (-) caster. The vast majority of the cars use positive caster. (Suspension Secrets, 2020)

A drawing of a car wheel

AI-generated content may be incorrect.

Figure Caster adjustment

### **Different types of suspension used in motorsport/sim racing:**

1. Double wishbone or four bar link independent suspension system – This is where eighty years of motor racing development has led them. For the past fifteen years at the rear and a lot longer than that at the front, virtually every serious racing car has employed one form or another of the four bar link independent suspension system. (Smith, 1999)

It offers a vastly different geometry to a MacPherson Setup, which features a control arm and radius rod joining the hub at the bottom, with the shock absorber protruding form the top. (Robinson, 2024)

As a result of its design, the more it is compressed, the more negative camber is introduced thanks to the use of unequal-length control arms. As the driver is cornering hard and the car starts to roll, a double wishbone system will maintain a better tyre contact patch on the road. (Robinson, 2024)

A car chassis with springs and springs

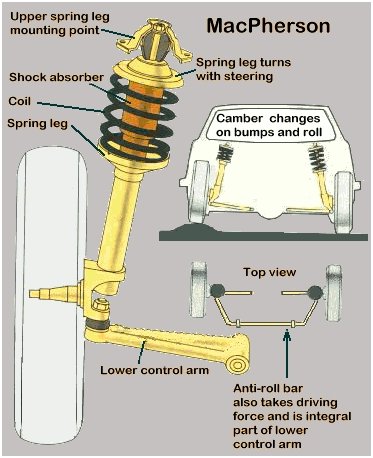
AI-generated content may be incorrect.

This can be achieved by stiffening the MacPherson setup and the biggest advantage is a softer damping without compromising the contact patch. The result is a car that’s typically more comfortable and less nervous-feeling than on bumpier roads. (Robinson, 2024) As a more complex design, the double wishbone allows for greater control over camber, caster and roll centre. There’s also much more freedom with the placement of the dampers, leading to the trick inboard setups on many racing cars and even some road-going supercars.

But the main con of a double wishbone suspension system is *cost*. Where MacPherson is simple and cheap, double wishbone suspension is more expensive and more complex. The design and manufacture of double wishbone setup is generally going to increase costs, which is why it is usually relegated to pricier machinery where ultimate handling is a bigger priority. Plus with more joints to worry about, the owner is potentially going to be burdened with higher service costs relative to MacPherson setup. (Robinson, 2024)

1. MacPherson setup – It is used, with some variations, at the front of most small passenger cars – and a large number of sports and GT cars. It is therefore very common in Touring and Grand Touring Race Cars. Its popularity has come about because it is very cheap to produce and offers pretty good camber control. (Smith, 1999) Unfortunately the camber control isn’t that good. It is difficult to arrange sufficient component stiffness to avoid compliance – particularly when race tyres are used – and it is virtually impossible to hide the strut inside a wide wheel – so that the steering offset on a production racer is going to become extreme when bolted on the wide wheels. (Smith, 1999)

(Smith, 1999) Years ago Colin Chapman adopted the MacPherson strut principle to the rear of several early Lotus racing cars and to the road going Lotus Elise. It worked just fine with the tyres available then but would not be suitable for use on a racing car today.



The system uses linkages that keeps the vehicle occupant comfortable and well isolated from road noise, bump and variations, etc. To simulate the dynamics of the linkages, first determine the surface of the terrain with the road noise removed. After that, determine the height of the road noise affecting each wheel. Finally, the links to these heights. But they will not be moved upward more than the specified value. In this way, the orientation of the car’s body is changing minimally. (Srisuchat, 2012)

The control arm locates the wheel laterally and the radius rod stops it moving fore and aft in the wheel arch. The control arm was usually the chunkier and stronger of the two and was the one that attached directly to the lower part of the wheel carrier, otherwise known as the hub. These days, the two have been streamlined into one, much larger, control arm in conjunction with a stabiliser bar that links the chassis and suspension unit.

The design really came into its own as cars began to be produced with ‘unibody’ chassis, also called monocoques. Monocoques have high relative rigidity between the MacPherson strut’s mounting point areas, giving it the kind of support and control it needs in order to work properly.

Brake bias – The brake bias can be adjusted towards the front or rear wheels to change how the car enters a corner and to control wheel lock up. If the front or rear tyres are locking when braking hard into a corner then shift the bias towards the non-locking wheels. This will even out the braking forces and prevent wheel lock up and loss of grip. (Suspension Secrets, 2020)

For tighter twisty circuits with smaller straight sections one can run a much more aggressive setup to increase turn in speed and maximise cornering performance. First of all increase the downforce at front and rear as this will generate more cornering speed. With that complete, increase negative camber at the front and rear. (Suspension Secrets, 2020) This will help increase contact patch as it can carry more speed through the corner and roll onto the tyre. Next increase toe out at the front wheels. This will help for initial turn in and will help to reduce understeer. If it’s a FWD (Front Wheel Drive) then some tow out at the rear wheels will also help to remove understeer and improve corner entry handling. For a RWD (Rear wheel drive) car slight toe in will still be ideal to keep the rear in control and to prevent oversteer. (Suspension Secrets, 2020)

Steering Geometry – The steering wheel in (Racer, 2014) is simple. It has lock (how many degrees it can turn), can deliver ackerman effects (one wheel turning more than the other) and has linearity added to modify steering responsiveness per car.

The maximum graphical angle of the steering wheel is specified in steer.lock (in degrees, not radians). It is mostly a graphical option: the relationship to the real controller’s lock may be preserved. The steering ratio (ratio of steering wheel rotation divided by wheel lock) is calculated based on this value, and the value of wheel 0’s lock. So: steeeringRatio = steer.lock/wheel0.lock. (Racer, 2014)

The Ackerman effect is that of the inner turning more than the outer wheel in a tight turn. This improves turn-in for low-speed big-angle corner.

wheel<x>.ackerman gives the factor at which the inner wheel will turn more than usual . (Racer, 2014) So if ‘ackerman’ = 2, then for the right turn the right wheel will turn twice as much.

The different curvature radii mean that to avoid sliding, the steering geometry must steer the inside front tyre at a larger angle than the outside front. Ackerman Steering refers to geometric configuration that allows both front wheels to be steered at the appropriate angle to avoid tyre sliding.

A math equation with a number of numbers

Description automatically generated with medium confidence

This a complicated formula to solve for the Ackerman geometry, for a given turn radius R, wheelbase L, and track width T, engineers calculate the required front steering angles (f,in) and (f,out).

A drawing of a square with a shadow

Description automatically generated with medium confidence

Different types of Terrain:

A racing game usually comprises more than one type of terrain. To simulate types of terrain in the car physics engine, one can specify the region for each type of terrain. A type of terrain has two attributes, which are rolling resistance coefficient and maximum height variation. (Srisuchat, 2012) For road, the maximum height variation is set as zero and the rolling resistance coefficient is set to a low value, whereas for tracks made of mud, the maximum height variation is set to a low value and the rolling resistance coefficient is set to a high value. The car moves slower in terrain where the rolling resistance coefficient is set to a higher value. The maximum height variation controls the roughness of the road surface. The car physics engine generates a random number between zero and the maximum height variation and add this number to the height that a wheel lies on. It then repeats the process for the three other wheels. In this way, it simulates the roughness of the terrain that the car moves on. (Srisuchat, 2012)

## Engine and Powertrain

When the engine runs, it generates a certain amount of torque. The torque that an engine delivers depend on the speed at which the engine is turning, usually expressed in terms of revolutions per minute, or *rpm*. The relationship torque versus rom is usually provided as a curve known as a torque curve. A torque curve of the 2014 Ferrari 458 Italia (Automobile Catalog, 2014):

A graph on a screen

AI-generated content may be incorrect.

Figure torque Curve of Ferrari 458 Italia. Look how the white line (horsepower) just shooting up and the torque (orange line) just there.

The angular velocity of the engine in rad/s is obtained by multiplying the engine turnover rate by 2 pi and dividing by 60.

A number with numbers and letters

AI-generated content may be incorrect.

“Torque is twisting ability, like turning a bolt of crankshafts. The torque is the force one apply time the perpendicular distance between where he apply the force and the center of rotation. If someone’s having a hard time loosening a lug nut, try using a longer wrench. If a person exert 30 pounds of force on a foot-long wrench, he has exerted 30 foot-pounds of torque.” (Leslie-Pelecky, 2009, p.72)

Torque determines how fast the cars can accelerate, and the amount of torque is determined by the force with which the pistons rotate the crankshaft. An engine’s torque is always measured at a specific engine speed. The engine speed is how many revolutions per minute (rpm) the crankshaft makes, and the engine produces different amounts of torque at different speeds. (Leslie-Pelecky, 2009) The dynamometer provides varying amounts of resistance to rotation, which changes the engine speed. The amount of torque the engine produces at each speed is recorded and also looks like the solid line on the graph.

“The engine’s horsepower is calculated by multiplying the torque (in foot-pounds) by the rotational speed of the engine (in rpm) and dividing by 5252.” (Leslie-Pelecky,2009,p.73)

The unit of horsepower was introduced by James Watt, the inventory of steam engine, and appears to have been more of a marketing tool than a true measure of a horse’s power output. The metric unity for power is the watt, so engine power may be given in kilowatts, which are thousands of watts.

Here’s another example for the 5.7 litre V8 engine found in Corvettes from 1997 to 2000; the LS1:

A graph with a line and numbers

AI-generated content may be incorrect.

Note that the torque curve peaks at about 4400 rpm with a torque of 350 lb-ft (475 Nm) and horsepower peaks at 5600 rpm at 345 hp (257 kW). The curves are only defined in the range from about 1000 to 6000 rpm, because that is the operating range of the engine. Any lower, and the engine will stall. Any higher (redline), and it’ll damage. (Monster, 2003)

A little history about horsepower: (Engineering Explained, 2012)

Since,

Power = force x velocity

Torque = force x radius => force = torque / radius

Velocity = distance / time

Distance / time = circumference (w.r.t. a wheel) \* RPM

= 2 pi r \* RPM (which is basically velocity)

And now if plug those values in here:

Power = (torque \* 2 pi r \* RPM) / r => radius cancels out leaving,

Power = torque \* 2 pi \* RPM

James Watts, a Scottish engineer, did an experiment on his horse about how much power a horse makes. In his little experimentation, he found that:

1 horse power = 33,000 lb-ft / min

And,

Power = torque \* 2 pi \* RPM / 33,000 it boils down to.

Power = torque \* RPM / 5252 = Imperial unit of horsepower or

kW = Nm \* RPM / 9549 = Metric unit of horsepower.

Now, the torque from the engine (that is at the crankshaft) is converted via the gear and differential before it’s applied to the rear wheels. The gearing multiplies the torque from the engine by a factor depending on the gear ratios. (Monster, 2003) Unfortunately, quite some energy is lost in the process. As much as 30% of the energy could be lost in the form of heat. This gives a so-called transmission efficiency of 70%.

The torque applied to the wheels is not the same as the engine torque because the engine torque passes through a transmission before it is applied to the wheels. The gear ratio between the two gear is the ratio of the gear diameters. (Srisuchat, 2012) Car transmission will typically have between three and six forward gears and one reverse gear. There is also an additional set of gear between the transmission and the wheel known as the differential and the gear ratio of this gearset is called the final drive ratio. The wheel torque can be obtained using the following equation:

Tw = Te \* gk \* G

Where Te is the engine torque, gk is the gear ratio of whatever gear the car is in and G is the final drive ratio.

The process to calculate rpm is:

Rpm = wheel rotation rate \* gear ratio \* differential ratio \* 60 / 2 pi

The 60 / 2 pi is a conversion factor to get from rad/s to revolutions per minute. There are 60 seconds in a minute and 2 pi radians per revolution. According to this equation, the crankshaft rotates faster than the drive wheels.

The relationship between the engine turnover rate and the wheel angular velocity is as follows:



If assumed that the tyres roll on the ground without slipping, the translational velocity of the car can be related to the engine turnover rate as follows:



The engine simulated in Racer is relatively simple. It can stall and restart it. The engine generates a specific amount of torque depending only on engine RPM. (Racer, 2014) The (normalized) torque is given through a curve file, which can be edited by Curved in Racer’s windows editor and determines the characteristics of output torque vs engine RPM. Through engine.max\_torque one can scale the curve to generate the actual torque (in Nm – Newton metre).

At low gears, the engine may produce a lot of torque, making controlling the car quite hard. Adding engine mapping parameters, per gear, defines the linearity of the throttle for each gear. This is used in Formula 1 for example for some more control in lower gears; it is also useful for other high-power cars. (Racer, 2014) Perhaps inverted linearity is also useful for low-power cars, to give them a bit more responsiveness.

Lower gear results in a greater acceleration but gear shifting is necessary as the velocity of the car is a function of the engine turn over rate and gear ratios, and every car has a limit to how fast the engine can turn over. There are two types of transmission – manual transmission and automatic transmission. To simulate automatic transmission in the game, one could just specify the engine turnover rate in which the transmission would shift. (Srisuchat, 2012) Brakes can also be placed in the powertrain module, mapped as a percentage of maximum brake torque (a fixed parameter) to brake pedal position. Brake torque acts as a torque on all tyres in the opposite direction as drive torque.

##### Gear Ratios

The following gear ratios apply to Corvette C5

First gear g1 2.66

Second gear g2 1.78

Third gear g3 1.30

Fourth gear g4 1.0

Fifth gear g5 0.74

Sixth gear g6 0.50

Reverse gR 2.90

Differential Ratio xd 3.42

Max torque 475 Nm (350 lb-ft) at 4400 rpm, mass = 1439 kg. In first gear at max torque this gives a whopping 475 \* 2.66 \* 3.42 \* 0.7 / 0.33 = 9166 N of force. This will accelerate a mass of 1439 kg with 6.4 metre per second square which equals 0.65 g. (Monster, 2003)

“The combination of gear and differential acts as a multiplier from the torque on the crankshaft to the torque on the rear wheels. For example, the Corvette in first gear has a multiplier of 2.66 \* 3.42 = 9.1. This means each Newton meter of torque on the crankshaft results in 9.1 Nm of torque on the rear axle. (Monster, 2003) Accounting for 30% loss of energy, this leaves 6.4 Nm. Divide this by the wheel radius to get the force exerted by the wheels on the rad (and conversely by the road back to the wheels). Let’s take a 34 cm wheel radius, that gives 64 Nm/0.34 m = 2.2N of force per Nm of engine torque. What the drivers gain in torque, have to pay it back in angular velocity. It is a trade-off strength for speed. For each rpm of the wheel, the engine has to do 9.1 rpm. The rotational speed of the wheel is directly related to the speed of the car. One rpm (revolutions per minute) is 1/60th of a revolution per second. Each revolution takes the wheel 2 pi R further that is 2 \* 3.14 \* 0.34 = 2.14m. So when the engine is doing 4400 rpm in first gear, that’s 483 rm of the wheels, is 8.05 rotations per second is 17.2 m/s about 62 km/h.” (Monster, 2003)

In low gears the gear ratio is high, the driver gets a lot of torque but no so much speed. In high gears, more speed but less torque. This can be represented in a graph as a set of curves, one for each gear:

Graph of a graph with colored lines

AI-generated content may be incorrect.

Note that these curves assume a 100% efficient gearing. The engine’s torque curve is shown as well for reference, it’s the bottom one is black. The other curves show the torque on the rear axle for a given rpm of the axle rather than the engine

Drive Wheel Acceleration:

The torque example in the torque curves for a given rpm, is the maximum torque at that rpm. How much torque is actually applied to the drive wheels depends on the throttle position. This position is determined by user input and varies from 0 to 100%. So, in pseudo-code, this looks like:

max\_torque = Lookup\_torqueCuve (rpm);

engine\_torque = throttle\_position \* max\_torque

The function Lookup\_torqueCurve by using an array of torque/rpm value pairs and doing linear interpolation between the closest two points.

This torque is delivered to the drive wheels via the gearbox and results in what is called Drive Torque:

drive\_torque = engine\_torque \* gear\_ratio \* differential\_ratio \* transmission\_efficiency

#### Here’s another lookup table for difference between Power and Torque

The performance on an engine is usually quoted by quoting the maximum power and torque and the engine rotation speeds at which they occur. (Bower, 1999)

A screenshot of a computer

AI-generated content may be incorrect.

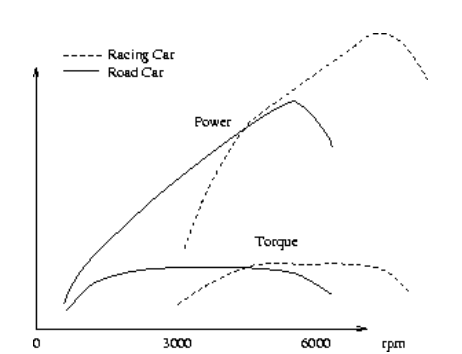


Figure power curve of an idealised petrol engine. solid line is road car, dotted line is race car.

These are maximum values. The power curve of a typical petrol engine is shown in figure 14. The power grows roughly proportionally to road speed while the torque is roughly constant. This is because they are both related to the force applied to the road to accelerate the vehicle. (Bower, 1999)

Hence,

Torque is inversely proportional to power divided by engine rpm.

So why is it useful to quote both? It is simplest to concentrate on the torque curve. This is what actually accelerates the car. The whole of the torque plateau is usually only 20% below the max torque, so this figure gives the level of plateau. In a racing car, engineers are not to much worried about where the plateau starts, but the lower that it begins, the easier it is to drive the car without continually changing gear to keep the engine spinning fast.

The maximum power is useful for calculating maximum speeds without knowing the gear ratios, but it also serves to tell where the power and torque curves drop. There is no point revving the engine much beyond the max\_power rpm. Once the peak power is reached, the driver soon need to change gear; this will reduce the engine rpm so that he can accelerate again, but the force applied to the road will be smaller because of the smaller xg (gear-box). (Bower, 1999)

(Mazhar, 2008) The gearbox was modelled with the following conditional statements in a video game for reference:

* If RPM is greater than max\_rpm (say around 8000) of a car, shift up.
* If RPM is less than 1000 times the current gear ratio, shift down.
* If in sixth gear, RPM cannot exceed 8250 RPM
* RPM increases by a fixed amount while accelerating depending on the gear.
* RPM slowly decrease if the vehicle is not accelerating
* Engine force is zero when gear is changing.

Torque on drive wheels:

To recap, the traction force is the friction force that the road surface applies on the wheel surface. This force will cause a torque on the axis of each drive wheel:

traction\_torque = traction\_force \* wheel\_radius

This torque will oppose the torque delivered by the engine to that wheel (called drive torque). If the brake is applied this will cause a torque as well (back torque or brake torque).

Diagram of a machine with words

AI-generated content may be incorrect.

Figure The diagram illustrates traction torque for an accelerating car.

The engine torque is magnified by the gear ratio and the differential ratio and provides the drive torque on the rear wheels. The angular velocity of the wheel is high enough that it causes slip between the tyre surface and the road, which can be expressed as a positive sip ratio. This results in a reactive friction force, known as the traction force, which is what pushed the car forward. The traction force also results in a traction torque on the rear wheels which opposes the drive torque. In this case net torque is still positive and will result in a acceleration of the rear wheel rotation rate. This will increase the rpm and increases the slip ratio. (Monster, 2003)

The net torque on the rear axle is the sum of the following torques:

total\_torque = drive\_torque + traction\_torques from both wheels + brake\_torque

Remember that torques are signed quantities, the drive torque till generally have a different sign than the traction and brake torque. If the driver is not braking, the brake torque is zero.

This torque will cause an angular acceleration of the drive wheels, just like a force applied on a mass will cause it to accelerate:

angular\_acceleration = total\_torque / rear\_wheel\_inertia

“A positive angular acceleration will increase the angular velocity of the rear wheels over time. As for the car velocity which depends on the linear acceleration, we simulate this by doing one step of numerical integration of each time we go through the physics calculation:” (Monster, 2003)

rear\_wheel\_angular\_velocity += rear\_wheel\_angular\_acceleration \* time\_step

## Collisions

Unlike most games, which use ‘Rigid-body’ physics simulation, BeamNG is a ‘soft-body’ physics simulator. (BeamNG.drive, 2013) In short, this means that physics objects (such as cars) are deformable. This is achieved through “Node and Beams” structures.

A close-up of a model of a pyramid

AI-generated content may be incorrect.

Nodes (The Chrome Balls) – can be thought of as particles, each node has mass, and can move freely in space.

Beams (The blue sticks) – hold nodes together in a structure. Beams have no mass, and always have a node at each end. They behave as springs. They are allowed to rotate freely around nodes, and only serve to keep two nodes at a set distance from each other.

“When building a node-beam structure, it is important to keep in mind that a box with no cross braces will fall over a the beams can pivot around the nodes. Adding diagonal braces between the edges will make the box into a rigid structure.” (BeamNG.drive, 2013)

With enough nodes and beams, it is possible to build complex structures like cars, with node-beam structures simulating the chassis, suspension, wheels, and many other components.

A drawing of a network

AI-generated content may be incorrect.

Figure This is how the skeletal version of a car looks like in BeamNG

“For game applications, speed is a major issue, and very accurate collision detection can be slow. For the sake of speed and simplicity, a bounding sphere scheme along with bounding box and vertex edge and vertex face collision detection schemes are used in this masters project.” (Srisuchat, 2012) Bodies that are colliding are treated as rigid bodies irrespective of their construction and material. There is a method to deal with collision response known as penalty method. In penalty method, the force at impact is represented by a temporary spring that gets compressed between the objects at the point of impact. This spring compresses over a very short time and applies equal and opposite forces to the colliding bodies to simulate collision response. (Srisuchat, 2012)

# Research Methodologies:

This study focuses on developing a custom vehicle physics system within Unreal Engine 5. The objective is to create a high-fidelity simulation that provides detailed control over vehicle dynamics, including suspension, drivetrain and tyre friction. The custom physics artefact is designed to incorporate advanced computations and features not present in Unreal’s built-in Chaos Physics system. While Chaos Physics is highly optimized for general real-time applications, it simplifies certain physics interactions to enhance performance. The custom artefact, in contrast, prioritizes realism and configurability, making it suitable for applications such as vehicle prototyping or simulation-based testing.

This chapter outlines the research design, implementation strategy, benchmarking approach and evaluation methodology used to assess the performance and accuracy of the custom physics artefact in comparison to Chaos Physics.

Research Design:

The research follows a comparative experimental design paradigm, involving the parallel implementation of:

1. The built-in Chaos Physics system, serving as a performance benchmark.
2. A custom vehicle physics system, designed to provide greater control over drivetrain mechanics, tyre friction and suspension behaviour.

The evaluation is structured around three core dimensions:

1. Performance Benchmarking – Measuring computational performance using Unreal Profiler.
2. Simulation Accuracy Testing – Assessing the fidelity of physical calculations, including suspension response, drivetrain dynamics, and tyre friction models.
3. Stability and Stress Testing – Evaluating system performance under various controlled conditions, such as high-speed manoeuvres, and long duration simulations.

This methodology allows for a direct comparison between Chaos Physics and the custom artefact, highlighting differences in efficiency, computational complexity and accuracy.

The research will be conducted in four phases:

1. System Development and Configuration
2. Performance Benchmarking
3. Simulation Accuracy and Stability Assessment

Development Process:

The custom vehicle physics system is developed using Blueprints within Unreal Engine 5. The system consists of multiple components, each designed to improve simulation accuracy and allow for a modular approach to vehicle dynamics.

1. Suspension system : A single-line trace (raycast) will be used for suspension simulation, similar to Chaos Physics. Suspension forces are computed using spring stiffness, damping coefficients and travel limits to simulate realistic vertical displacement. Initialization parameters including current length and last length to track suspension travel over a given delta time.
2. Drivetrain and Wheel Dynamics : The system supports various drivetrain configurations, including front-wheel drive (FWD), rear-wheel drive (RWD) and all-wheel drive (AWD). A manual transmission model will be implemented, allowing for gear shifting, reverse gear and unique gear ratios per vehicle. Wheel speed and torque calculations will include RPM-to-rad/s conversions, per-wheel acceleration computations, and angular velocity adjustments based on gear ratios. Brake bias will be incorporated to allow realistic braking force distribution across front and rear axles.
3. Tyre Friction and Force Application with Debug : The Pacejka’s “Magic Formula” will be used to calculate tyre forces dynamically with the help of Pacejka’s tyre slip curve, ensuring accurate grip levels based on slip ratios and velocity. Friction torque with respect to different surfaces and longitudinal forces are computed per wheel, adjusting real-time vehicle behaviour.
4. Wheel Position and Rotation Updates : Unlike Chaos Physics which utilizes skeletal mesh and body rigged animation for wheel movement, this system will apply direct transformations based on computed drivetrain outputs. Each wheel’s position and velocity are continuously updated based on suspension travel, drivetrain input and terrain interaction.

For comparison, the built-in Chaos Physics system is configured with default vehicle parameters. Chaos Physics utilizes:

1. Pre-optimized suspension and tyre models, providing an efficient approximation of real-world vehicle dynamics.
2. A simplified drivetrain model, optimized for real-time applications.
3. Multi-threaded physics calculations, allowing for improved performance at the cost of reduced simulation detail.

This setup ensures that both systems are tested under identical conditions, enabling an accurate performance and accuracy comparison.

Evaluation Methodology:

The performance of both system is analysed using Unreal Profiler, focusing on:

* Frame Time (ms): The time required to process each frame.
* CPU and GPU Utilization: The percentage of processing power used by physics calculations.
* Memory Allocation: The total memory consumption of each system.
* Physics Thread Efficiency: The level of parallelization achieved within the physics engine.

Stress tests are performed to examine system behaviour under conditions such as:

* High-speed manoeuvres to assess vehicle stability.
* Prolonged simulation runs to measure consistency over time.
* Variable terrain interactions to evaluate adaptability under different surface conditions.

Each test will be conducted under identical circumstances for both systems to ensure fair comparisons.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure The figure above shows the class diagram of the physics model describing the system's classes, their attributes and operations. (Only some of the attributes and operation are listed)

Get\_Unit\_Convert() is to convert the values from RPM to radian/second for engine angular velocity and vice versa for calculating horsepower.

Ray\_CastINI() is to initialize raycast on each toplink give the suspension properties.

SuspensionINI() is to initialize suspension with the values of stiffness, damper, length of a suspension and how much the suspension should travel with last\_length as well.

TransmissionINI() is to initialize the drive train to give accessibility for the player to switch between FWD (Front wheel drive), RWD (Rear Wheel Drive) and AWD (All Wheel Drive).

WheelINI() is to initialize the wheel with properties like radius of a wheel and mass (weight) of a wheel.

GetDriveTorque() is to get how much torque each wheel is getting when the car accelerates or brakes.

WheelAcceleration() is to get angular\_acceleration and angular\_velocity and clamp the values with the total gear ratio and set the wheel\_angular\_velocity.

Get\_Suspension() is to get the toplink component of each wheel, set the ray length and create a raycast and mimic the behaviour of a suspension with length and travel. Also getting the length and applying springForce and damperForce to the suspension to minimise the jump nature.

Get\_Tyre\_Force\_Combined() is a tyre friction model which replicates the Pacejka’s tyre friction model by using a Curve float and check for traction. If carSpeed \* slipSpeed > 0 traction, else friction. This function is useful to check for longitudinal slip and lateral slip, normalizing it and multiply it by with the Curve graph and set the Fx and Fy values on the wheels.

GetTotalDriveVelocity() is to set the wheel angular velocity with respect to drivetrain.

DrawDebugForce() is to showcase how the forces are behaving in the game world.

WheelRotation() is to rotate the wheel.

And below the parent class are the child classes:

1. Subaru – All Wheel drive, max rpm is 6700 and number of gears is 6 with its own gear ratio value.
2. Jaguar – Rear Wheel drive, max rpm is 7000 and number of gears is 7.
3. Volkswagen – Front Wheel drive, max rpm is 6500 and number of gears is 5.

# Results and Analysis:

Driving the car gently and vigorously, the engine is able to simulate the physics of cornering at low speed, cornering at high speed, oversteering, understeering and drifting accurately. The suspension seems to look quite realistic and the tyre friction model seems to do its job. Engine RPM and torque also able to do calculations in real-time to generate horsepower of the car.

Before jumping into the results, let’s understand what total inclusive time and total exclusive time means in Unreal Engine’s Profiler.

Total Inclusive Time – is the total time a function takes, including the time spent in any other functions it call. For example,

function A() that calls B() inside it:

* A() takes 10ms total, but B() takes 6ms inside A()
* Total Inclusive time for A() = 10ms (since it includes B’s time too)
* Total Inclusive time for B() = 6ms (since it does not call anything else)

Total Exclusive Time – is the time spent only in that function itself, excluding the time taken by any other functions it calls. For example, (continuing the same function A() and B() time)

* Total Exclusive time for A() = 4ms (since 6ms was spent in B())
* Total Exclusive time for B() = 6ms (since it does all its work itself)

Why does this matter? – High inclusive time, low exclusive time means the function is mostly waiting on other functions it calls and high exclusive time means the function itself is slow and needs optimization.

Keywords from the graph below:

**1** – WaitForTasks – Time spent waiting for tasks to complete. This includes synchronization points where the engine waits for asynchronous tasks (for example, physics, rendering) to finish.

**2** – FEngineLoop::Tick – Time spent in the main engine loop’s tick function, which handles the core update logic for the engine (for example, game logic, physics, rendering).

**3** – Frame – Total time spent processing a single frame, including all subsystems (for example, game logic, physics, rendering).

**4** – RenderingFrame – Time spent on rendering tasks for a single frame, including drawing objects, applying shaders, and managing the GPU.

**5** – WaitUntilTasksComplete – Time spent waiting for all tasks to complete before proceeding to the next frame.

**6** – UWorld\_Tick – Time spent updating the game world, including actors, components, and physics objects.

**7** – ExecuteCommandList – Time spent executing rendering commands on the GPU

**8** – Submission\_Wait – Time spent waiting for the GPU to finish submitting rendering components.

**9** – ReceiveTick – Time spent processing networking ticks, which are updates received from the network.

**10** – ReceiveBeginPlay – Time spent processing the BeginPlay event, which is triggered when an actor is spawned or the game starts.

**11** – Allocate – Time spent on memory allocation operations.

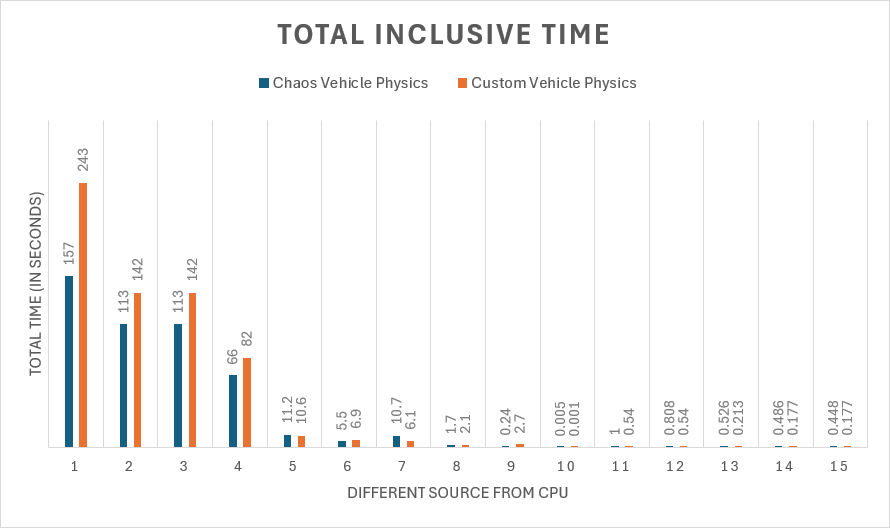
**12** – Tasks\_Wait – Time spent waiting for tasks to start or complete.

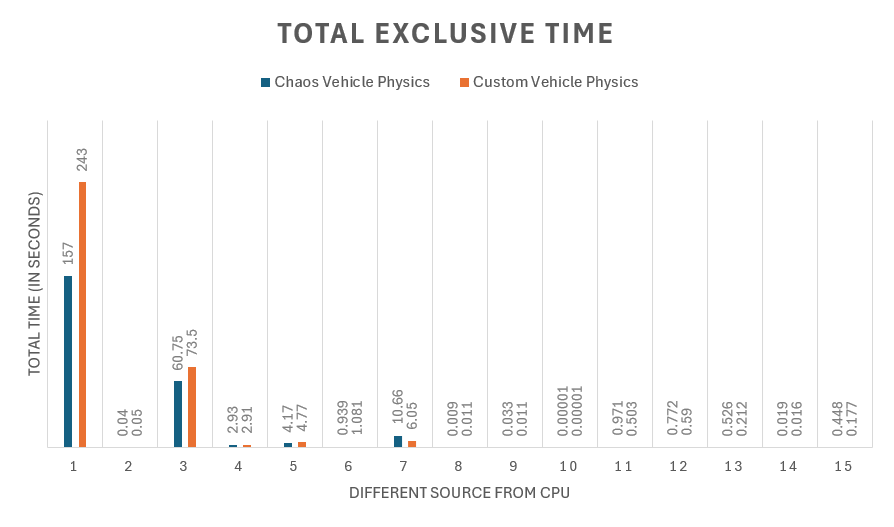
**13** – Buffers – Time spend managing rendering buffers

**14** – Chaos\_PhysicsParallelFor – Time spent executing parallel physics calculations

**15** – GameThreadWaitForWork – Time spent by the Game Thread waiting for work from other threads (for example Physics Thread, Render Thread)

So after a thorough analysis with identical frame rate (60 fps) and identical racetrack with props, here are the results of the results for the performance between Chaos Vehicle Physics system and the custom vehicle physics artefact:





After comparing both the Chaos and Custom Vehicle artefact, the results are somewhat similar. Where Chaos was flourishing on frames, thanks to being Unreal’s built-in system, the custom artefact was putting up a good fight in terms of memory allocation, buffers, tasks\_wait, executing command list, and waiting until the tasks are complete.

Why the custom vehicle physics is almost same or below average than Chaos physics system? Because Chaos has a C++ vehicle physics component system which does not require to calculate basic wheel or suspension model in runtime. The reason why custom artefact was struggling with rendering frames, frames and FEngineLoop::Tick because each variable whether it is the rpms, how much torque is going on each wheel, producing torque on each wheel with respect to the gear ratio, number of raycast lines being created as the car goes forward and other advanced physics models were being accurate caused bottlenecks by serial execution.

Chaos wins overall due to parallelization across threads, whether it is physics, rendering and tasks are spread across the threads. Memory efficiency – even though the custom artefact allocating memory faster than chaos, chaos uses memory pools for physics entities to minimize runtime allocation.

Chaos spends marginally more time on parallel physics using parallel-for loops distributing physics calculations across threads where as the custom artefact was going through each wheel using foreach loop on each function to calculate the data.

Below another table to showcase how Chaos is 2-15% better than Custom Physics System

Game thread is like the brain of the game. It handles all the logic, like how the car moves, when it collides with something, and other game rules.

Render thread is it takes all the information from the Game Thread and draws it on the screen to see what is happening.

|  |  |  |
| --- | --- | --- |
| Instances | Chaos Vehicle System | Custom Vehicle System |
| When Game Start | Game Thread = 2.1s to reach 2fps when loading the game | Game Thread = 168.5ms (0.168s) to reach 2fps when loading the game. |
| When running around the track | Game Thread = 17.39ms to reach 57.5fps  Render Thread = 8.3ms to reach 200fps | Game Thread = 18.19ms to reach 55fps  Render Thread = 11.46ms to reach 87.3fps |
| When colliding with the blue cylinders laid on the track | Game Thread = 16.65ms to reach 60.1fps  Render Thread = 6.32ms to reach 158.3fps | Game Thread = 16.24ms to reach 61.6fps  Render Thread = 8.28ms to reach 120.8fps |
| When exiting the game | Game Thread = 147.58ms to reach 6.8fps  Render Thread = 137.15ms to reach 7.3fps | Game Thread = 156.22ms to reach 6.4fps  Render Thread = 147.57ms to reach 6.8fps |

# Conclusion

A commercial grade car physics system involves many areas of physics to achieve realistic values. This thesis only shows some techniques to simulate the dynamics and compare the CPU/GPU performances of a racing sim. While some calculations in the engine are only approximation and some do not rely on real physics, it produces decent results and supports many cars with different drivetrain.

Due to lack of user participation this artefact didn’t have accurate results and analysis in terms of how the framework/tool is helpful for others.

After a thorough analysis comparing the performances between Chaos and the custom vehicle physics it is safe to say that Unreal can be a frontrunning engine to provide a realistic real-time vehicle system. The artefact will be a toolset for the development of racing game but there is some compromise in terms of CPU/GPU profiling to achieve total realism. Even though it is inherently difficult to balance performance with realism, it is not entirely impossible. The custom vehicle physics artefact had functions after functions to calculate forces in real-time which ended up axing most part of frames for rendering and runtime speed. Due to the lack of user participation there was no proper feedback on the framework and how to improve it.

# Recommendations

###### The limitations of the custom vehicle artefact are –

Tyre temperatures like Live For Speed implemented a life cycle to deal with tyre temperatures. 12

Telemetry UI to help visualize any forces whether it is the engine torque, horsepower or even the Pacejka’s friction model. Figure 14

An overall better collision body because the custom artefact is using convex collision which is slightly daunting on the CPU and something like a rigid-body would be a massive burden taken off the shoulders of the custom artefact. Figure 17

Sound cues for when the tyre is losing grip while oversteering or understeering, when the engine is in idle or revving or when it is shifting gears.

Accurate suspension models since the framework is using a single-line raycast suspension, every bump on the road is inconsistent. Perhaps a multi-line raycast would work or something like nodes and beams like BeamNG did to create soft-body physics for the wheels.

Converting the whole project into C++ so it could be optimized on how the calculations are executed, not just what is being calculated. There has to be a way to do this more efficiently.

There are no driving assists like ABS (Anti-lock Braking System) or TCS (Traction Control System) for accessibility for casual players.

No aerodynamic drag and downforce acting on the car.

No camber or toe option in the suspension system to create more grip while cornering.

Better camera control on the car.

There is not a single toggle UI to switch between different drivetrains (FWD, AWD, RWD) in real-time, to change the suspension spring length, dampers and travel or other accessibilities options with respect to car customization.

##### Future Work –

There are plenty of basic things which needs to be done. Like:

Working on the clutch model and automatic transmission. Basically how clutch works in real life, when the driver wants to shift gears up or down, he needs to first press the clutch and then proceed to gear up or down. Then automatic transmission feature which will be quite easy to implement because all focus will be going into producing maximum torque with minimum to none gear ratios.

Wheels particles effect. So when the car drifts or accelerates suddenly there are two thing happening: tyre skid marks and smokes coming out of the tyre.

Better scalability, system integration, function and class handling with the help of C++. Most of the physics calculations will go in C++ rather on blueprints to avoid CPU overhead.

Different surfaces like gravel, tarmac, snow and etc. There are different surfaces in the framework but there is no proper use of it at the moment.

A basic UI like rev counter and gear shifting. Back in the 60s racing cars used to have only rev counters, no indication of how fast the driver is going. So to stay true to the tradition of racing in the 60s there will only be rev counter and gear number.

Using Unreal’s Enhanced Input System for better input handling than axis values which was causing performance issues.

Even though already hinted, shifting the project into C++.

Making a simple racing game using the custom vehicle physics to test the accuracy of real-life counterpart of each car and the CPU/GPU performance of physics calculations in real-time.

# References

Alex, 2023. *The Decline of Vehicle Physics in Arcade Racers.* [Online]   
Available at: https://youtu.be/e4uhTKH8\_xQ?feature=shared

Automobile Catalog, 2014. *Torque Curve / Horsepower of Ferrari 458 Italia.* [Online]   
Available at: https://www.automobile-catalog.com/curve/2014/1227470/ferrari\_458\_italia.html#gsc.tab=0

BeamNG.drive, 2013. [Online]   
Available at: https://www.beamng.com/game/about/physics/

BeamNG.drive, 2013. *Soft-body Physics.* [Online]   
Available at: https://www.beamng.com/game/about/physics/

Beckman, B., 1991. *The physics of Racing.* [Online]   
Available at: http://autoxer.skiblack.com/phys\_racing/phors01.htm

Berg, A., 2022. *The Physics of Racing Part 1 - Weight Transfer.* [Online]   
Available at: https://www.allenbergracingschools.com/expert-advice/physics-racing-part-1-weight-transfer/

Bower, R., 1999. *The Physics of Motorsport.* [Online]   
Available at: https://rgbower.webspace.durham.ac.uk/the-physics-of-motorsport-2/

Campbell-Brennan, J., 2020. *Tyre Dynamics.* [Online]   
Available at: https://www.racecar-engineering.com/tech-explained/tyre-dynamics/

Campbell-Brennan, J., 2020. *Tyre Dynamics.* [Online]   
Available at: https://www.racecar-engineering.com/tech-explained/tyre-dynamics/

Cyber Racing teams, 2024. *Basic sim racing setup.* [Online]   
Available at: https://en.lfsmanual.net/wiki/Basic\_Setup\_Guide#Tires

Engineering Explained, 2012. *Engines.* [Online]   
Available at: http://www.howdoesacarwork.com/p/1-engines\_10.html

FGear, 2019. [Online]   
Available at: https://fgearvp.wordpress.com

Forza Horizon, 2021. [Online]   
Available at: https://forza.net/horizon

GDC, 2021. *Vehicle Physics and Tyre Dynamics in Just Cause 4.* [Online]   
Available at: https://www.youtube.com/watch?v=0jsENVOmkxc

Horowitz, K., 2004. *OUTRUN.* [Online]   
Available at: https://www.sega-16.com/2004/07/outrun/

Leslie-Pelecky, D., 2009. *The physics of NASCAR.* [Online]   
Available at: https://drdiandra.com/motorsport/physics-nascar/

Maul, P., 2021. *BeamNG.tech Technical Paper..* [Online]   
Available at: https://beamng.tech/blog/2021-06-21-beamng-tech-whitepaper/bng\_technical\_paper.pdf

Mazhar, H., 2008. *Real-time Vehicle Simulation For Video Games using Bullet Physics Library.* [Online]   
Available at: https://sbel.wisc.edu/wp-content/uploads/sites/569/2018/05/Real-time-Vehicle-Simulation-for-Video-Games-Using-the-Bullet-Physics-Library.pdf

Michelin, 2001. *The Tyre..* [Online]   
Available at: http://docenti.ing.unipi.it/guiggiani-m/Michelin\_Tire\_Grip.pdf

Monster, M., 2003. [Online]   
Available at: https://www.asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html

Nesvadba, M., 2023. *Transforming Real-world race course into racing simulator level.* [Online]   
Available at: https://is.muni.cz/th/y7kue/thesis.pdf

Polyphony Digital, 2022. [Online]   
Available at: https://www.gran-turismo.com/gb/

Preiti, D., 2018. *Optimisation of the tyre model in rFactor2.* [Online]   
Available at: https://www.politesi.polimi.it/bitstream/10589/153184/1/Optimisation%20of%20the%20tyre%20model%20in%20rFactor2%20environment%20for%20AVEHIL%20professional%20simulator.pdf

Project Chrono, 1998. *Simulation Engine..* [Online]   
Available at: https://projectchrono.org

Racer, 2000. *Pacejka's Magic Formula.* [Online]   
Available at: http://www.racer.nl/reference/pacejka.htm

Racer, 2014. *basic car physics..* [Online]   
Available at: http://www.racer.nl/reference/carphys.htm#steerlock

Robinson, M., 2024. *What is Double Wishbone suspension?.* [Online]   
Available at: https://www.carthrottle.com/news/what-actually-double-wishbone-suspension

Santos, R. d. O., 2014. *Race Car Aerodynamics.* [Online]   
Available at: https://racingcardynamics.com/race-car-aerodynamics-air-lap-times/

Shea, C., 2017. *The original ridge racer: A playstation launch classic.* [Online]   
Available at: https://www.ign.com/articles/2015/06/23/the-original-ridge-racer-a-playstation-launch-classic

SkyFall, 2023. *The Physics of BeamNG.* [Online]   
Available at: https://www.youtube.com/watch?v=-6OiHfPDjmI

Smith, C., 1999. [Online]   
Available at: https://www.carrollsmith.com/books/tune2win.html

Suspension Secrets, 2020. *Sim Racing Suspension Set up guide..* [Online]   
Available at: https://suspensionsecrets.co.uk/sim-racing-suspension-set-up-guide/

Tan, A., 2015. *Car Simulation Development Engineer.* [Online]   
Available at: https://www.polyphony.co.jp/people/1001/

The Physics Classroom, n.d. [Online]   
Available at: https://www.physicsclassroom.com/class/newtlaws/lesson-3/newton-s-second-law  
[Accessed 2025].

Unity 6, 2025. *Wheel Colliders.* [Online]   
Available at: https://docs.unity3d.com/Manual/wheel-colliders.html

University of Bournemouth, 2012. *Development of Car Physics Engine for games.* [Online]   
Available at: https://nccastaff.bournemouth.ac.uk/jmacey/MastersProject/MSc12/Srisuchat/Thesis.pdf

Vehicle Physics Pro, 2010. *Features of Vehicle Physics Pro.* [Online]   
Available at: https://vehiclephysics.com/about/features/