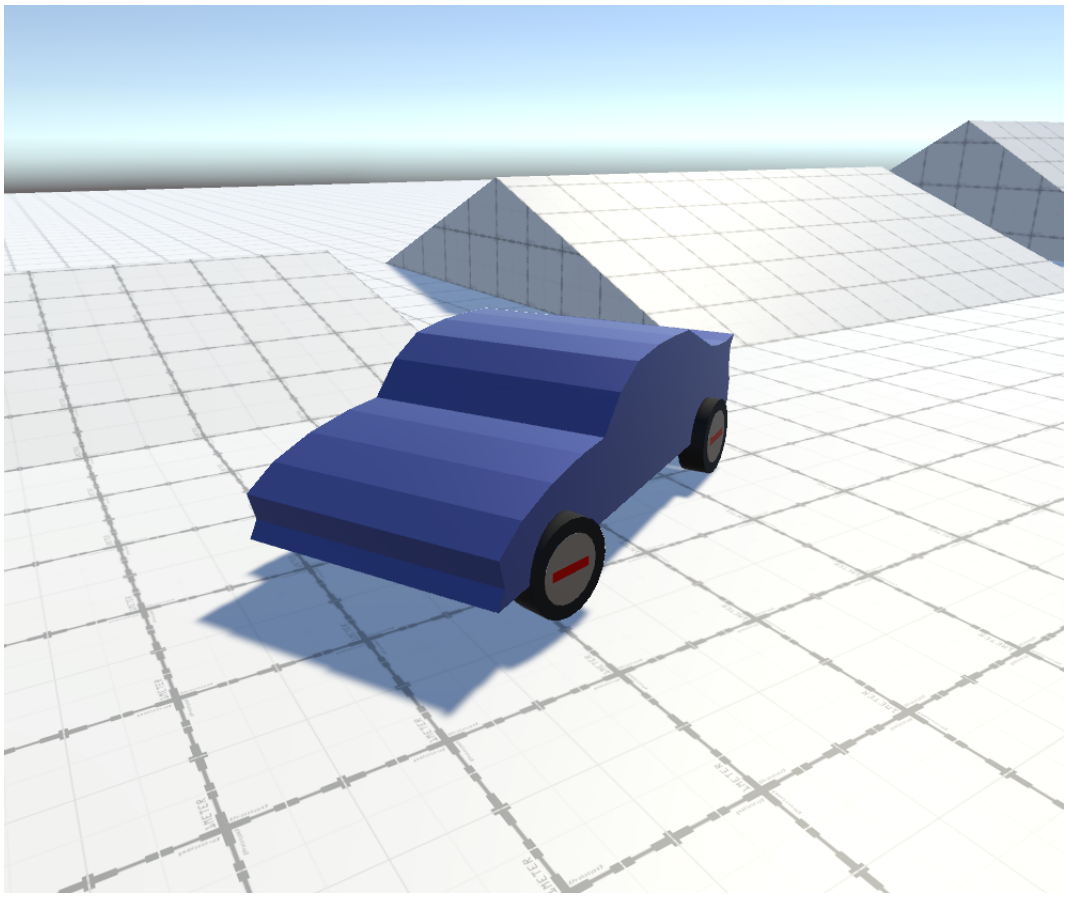

Bit Shox
Vehicle Simulation in Unity

Version 1.0

Lewis Cabal

December 7, 2025

Contents

List of Code Listings

1 Introduction

My main goal with Bit Shox is to create a deeply customizable and realistic vehicle simulation system. I wish to be able to represent a wide variety of real-life vehicle types, from everyday family haulers, 2-seater sports cars, and rock-crawling trucks. I've always been a fan of motorsport, especially Rally Racing and the World Rally Championship. Rally cars are not only fast, but they must also weather harsh and volatile environments and terrain. Stages can range from smooth tarmac, gravel trails, and snowy mountain roads, meaning cars must be precisely tuned.

2 System Specifications

Bit Shox is a realistic vehicle simulation system in Unity that gives users the tools to create a wide variety of cars easily and quickly. Car specifications from wheelbase, car weight, drive-train, and suspension characteristics are all modifiable within this system.

Bit Shox was developed in Unity 2022.3, and has not been tested in more recent versions of Unity (ex. 2023.x, 6000.x). To control the car, it is ideal to use a gamepad. If more than one car exists in the scene, they are all controlled at once by the same input. Table 1 contains the input control scheme.

	Throttle	**Brakes**	**Steering**
Gamepad	Right Trigger	Left Trigger	Left Joystick
Keyboard	W	S	A & D

Table 1: Control Scheme

2.1 Features

Regarding car customization, the following parameters are openly available to developers: Customizable Visuals; Wheelbase and Track (distance between wheels); Car Weight; Drive Type (FWD, RWD, AWD); Steering Angle; Suspension Depth, Angle, and Rest Length; Suspension Spring and Damping Coefficients; Tire Friction, Width, and Diameter; and more ...

2.2 System Structure and Classes

For most users, the *BSCar* script and class will be all that you need to get a car moving in your scene. Figure 1 depicts the key classes included in the package. The *BSWheel* class encapsulates the suspension, power, and traction physics for each individual wheel. For each *BSCar*, there are four *BSWheel* children. The list of methods and attributes are not exhaustive, but instead outline the core.

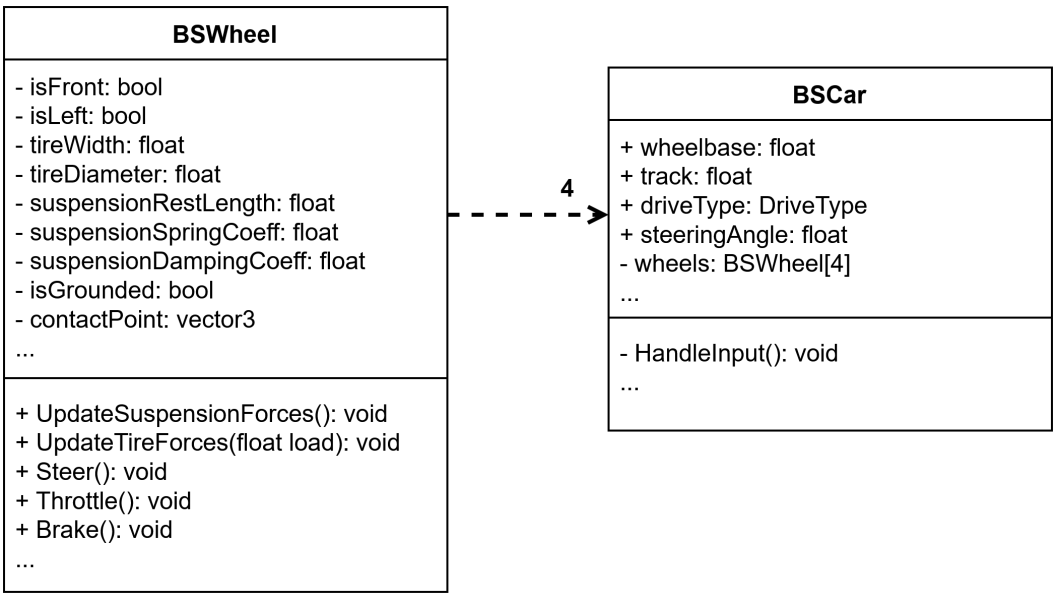

Figure 1: System Architecture

2.2.1 BSCar Class

The *BSCar* class is the main entry point for the system. It's main purpose is managing and high-level control and physics that cannot be self-contained within an individual wheel. It primarily handles any player inputs, and facilitates the simulation loop regarding suspension and wheels. Below is a high-level overview of a single simulation loop iteration:

2.2.2 BSWheel Class

The *BSWheel* suspension system uses the popular Raycasting method to determine suspension forces. It executes this by casting a ray from the

```
void FixedUpdate()
    HandleInput();
    foreach (BSWheel w in wheels) w.UpdateSuspensionForces();
    foreach (BSWheel w in wheels) w.UpdateTireForces();
```

Listing 1: BSCar Simulation Loop

suspension base out towards the wheel position. On contact, the point and normal can be determined allowing for spring and damping calculations to be performed, and forces to be enacted appropriately. The physical simulation methods are defined in *BSWheel*, but are called in *BSCar*. The only updates it performs itself relate to the visual representation of the wheel, repositioning itself to track the wheel-end of the suspension system.

Regarding the physical simulation of *BSWheel*, there are three main components: Throttle forces, Lateral forces, and Suspension forces. Throttle forces are enacted on the center of powered wheels when the Throttle input is given. Lateral forces give wheels traction, enabling steering through the front wheels. Suspension forces are a result of compression with the driving surface, applying force to the base of the suspension strut.

3 System Usage

Setting up a car is as easy as dragging and dropping a setup from *Assets/Prefabs/* and adjusting the parameters from there. These prefabs include a Camera setup which is controlled by the Right Joystick. Other controls can be seen in Table 1.

To create a car from scratch and use your own Camera setup, do the following:

1. Create an empty ***Game Object*** in your scene and name it.
2. Click ***Add Component*** and add the ***BSCar*** script.
3. With the ***Game Object*** selected, adjust the ***BSCar*** script values to your desired vehicle specifications in Unity's Inspector tool.

4 Physics Framework

Bit Shox's core physical features revolve around three forces: Throttle forces that propel the car through its powered wheels; Lateral friction forces that push the car to a desired direction through steering; and Suspension forces that push the cars body away from the ground and ensure contact between tires and surface is maximized. These three forces are necessary in basic car functions. However, they can vary in complexity and depth of their implementation, determining the physical accuracy of the system. The implementations these systems can be found in the *BSWheel* class.

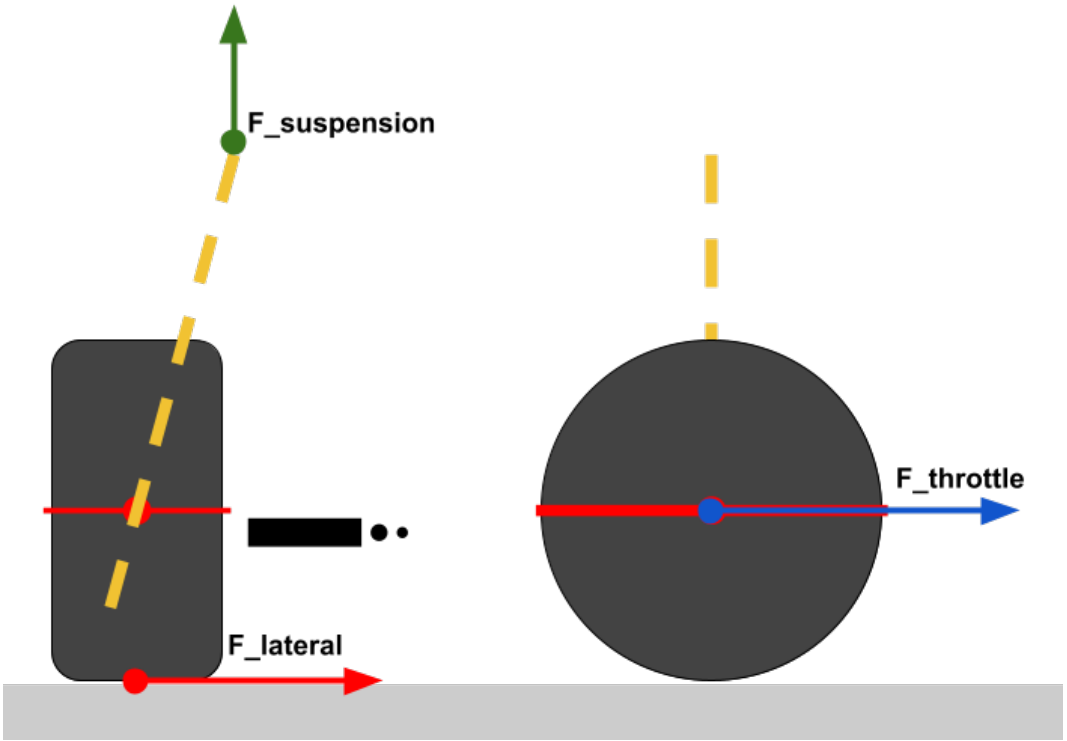

Figure 2: Forces Summary

4.1 Throttle Forces

Throttle forces are the primary force that propel the vehicle in your desired direction. In real cars, the engine creates torque which is applied to the wheels. The wheels then grip the surface below it, propelling the car forward.

In Bit Shox, the throttle force is determined using the Trigger input ($x \in [0, 1]$), some maximum power constant (k), and the normalized forward direction parallel to the contact surface ($\vec{p}$). This force is applied to the center of the wheel.

This gives us the equation: $F_{throttle} = \vec{p} \cdot xk$

This is a fairly simple implementation, disregarding car components such as differentials and accurate power curves.

4.2 Suspension Forces

Suspension forces are a key force in vehicle dynamics, maximizing the contact between each wheel with the ground surface. In real cars, a combination of spring(s), dampers, and complex geometry manage the wheel's range of movement.

In Bit Shox, the suspension system utilize a spring and damping force components to calculate the final force applied to the body. The following are the key components for calculating the net force: Spring's compression, $C = \text{clamp}(L_{rest} - L_t, 0, L_{rest})$; Spring's speed, $v_s = \frac{L_{t-1} - L_t}{\Delta t}$; Spring constant, s_k; Damping constant, s_d; Contacted surface's normal, $\vec{n}$. Using these components, we get the following and apply it to the base of the suspension strut.

Spring Force: $F_{spring} = \vec{n} \cdot s_k \cdot C$

Damping Force: $F_{damping} = \vec{n} \cdot s_d \cdot v_s$

Net Force: $F_{suspension} = F_{spring} + F_{damping}$

4.3 Lateral Forces

Lateral forces are key in enabling complex car control and driving feel. In real cars, wheels rotate about a single rolling axis, while opposing movement laterally. The force generated by this enables steering, drifting, and other complex tire dynamics.

In Bit Shox, lateral forces include slightly more complexity, as the loaded weight on a given tire has drastic effects on its physical properties. In a naive implementation, the weight is simply divided by the number of wheels,

assuming a uniform distribution of weight. Although this is true in a stationary vehicle, it quickly fails under acceleration and other external forces. The Lateral force is determined using the following components: Normalized lateral vector ($\vec{z}$), the lateral speed of the wheel ($v_{lateral}$), a friction coefficient (μ), and the current loaded weight on the wheel, in which we can re-use $F_{suspension}$.

This gives us the equation: $F_{lateral} = -\vec{z} \cdot v_{lateral} \cdot \mu \cdot |F_{suspension}|$

This is a more detailed implementation relative to the Throttle forces. One shortcoming is the constant friction coefficient. In reality, wheels can lock up, changing the tire dynamics. Another shortcoming is the maximum lateral force a given tire can output. In reality, this maximum force is a function of the tire's slip angle.

5 Future Work

While Bit Shox is in a decent state currently, there are still some significant shortcomings regarding the physical models and representation. First, the wheel contact is represented by a single point (Ray Cast). While not an issue on smooth terrain, the visual wheel often clips unnaturally on bumps and uneven surfaces. Second, proper throttle and engine power simulation would be extremely nice to have, and elevate the entire system with it. Third, utilization of slip angle and other parameters to dynamically calculate a maximum lateral force for a tire. Fourth, implementation of Anti-Roll Bar physics would create more stable car configurations.