CHAPTER 8

# Cars and Motorcycles

Car racing games are a lot of fun. You can drive high-powered vehicles that you couldn't possibly afford in real life, and you can drive them into trees. Between the speed, the power, and the spectacular fiery crashes, you can create a lot of exciting game scenarios using cars, motorcycles, and other motorized vehicles. But it's important to get the physics right in a car or motorcycle simulation. Your game should properly model, for example, whether a Nissan 350Z can outrun a police cruiser.

In this chapter, we will explore the physics of cars and motorcycles, but the basic principles we will explore are equally applicable to other types of motorized vehicles. The focus will be on the external physics of the vehicles—how they accelerate, how they brake, how they travel around curves. We won't go into the physics of the internal workings of motor vehicles. As a game programmer, you don't really need to know the physics of the internal combustion engine or how disk brakes work.

Some of the topics we will cover in this chapter include the following:

* A brief history of the automobile
* The basic force diagram of a car
* Engine torque and power: how to compute them from the engine turnover rate
* Gears and gear shifting: how gears are used to increase the torque applied to the wheels
* Rolling friction of car tires: what it is and how to calculate it
* Computing the acceleration and velocity of a car
* Braking: how brakes work and how to model braking in game programs
* Wheel traction and how it can limit the acceleration of a car
* Turning: how to compute turn radius and turn rate and how to model the effects of high-speed turns
* Motorcycles and how they turn

We'll also develop a car simulator that will model the performance and operation of a sports car.

## Cars

You can create a lot of exciting game simulations involving cars, whether it's car races, car chases, or just simulating the life of a taxi driver. Putting realistic physics into a car simulation really just involves applying some basic concepts from Newtonian mechanics and kinematics with a little knowledge about how power is transferred from the engine to the wheels.

In this section, we'll start with the basics of straight-line driving and explore topics such as the forces that act upon a moving car, engine torque, and how gears and a transmission transfers the engine power to the wheels. We'll revisit the subjects of aerodynamics and rolling friction as they apply to the motion of a car. With these topics in hand, we'll explore how to model the acceleration and velocity of a car. Later, we'll investigate such topics as what happens when a car drives around a curve and wheel traction, and we'll also develop a simple car simulator.

Keep in mind when reading this section that it provides the basic theory of car physics. It will give you the information you need to create a fairly realistic car simulation. Advanced topics such as weight transfer during braking and cornering are not included in the model we will develop, but if you want to get "fancy" with your car simulation, it would be pretty straightforward to add advanced effects to the basic model. We'll discuss briefly how to go about adding advanced effects at the end of the chapter.

## A Brief History of the Automobile

People have been thinking about, designing, and building motorized vehicles for a very long time. The first working motorized vehicle was a steam-powered tractor built by a French engineer named Nicolas Cugnot in 1769. It was intended to pull cannons and had a top speed of 4 *km/hr*. Mr. Cugnot was also the first person to have a car accident when he drove his vehicle into a stone wall in 1771.

The development of automobiles took a big leap forward towards the end of the nineteenth century when a German engineer named Karl Benz designed and built the first vehicle powered by an internal combustion gasoline-powered engine. While cars gained in popularity over the next 20 years, they were quite expensive and considered mostly toys for the rich. In 1913, the American Henry Ford perfected the assembly line, which could assemble a Model T car in only 93 minutes. Such improved productivity greatly reduced the cost of buying a car. By 1927, over 15 million Model T's had been built and sold.

Obviously, cars have never looked back, and they are a crucial element of the transportation systems of almost every country on earth. Cars have also become a form of entertainment for people who like to drive fast, look good, or just generally have a good time.

## Basic Force Diagram

A schematic of the forces acting on a car driving in a straight line on an inclined surface is shown in Figure 8-1. The angle of the slope is equal to $\theta$. The car is influenced by the forces of gravity, static friction, rolling friction, and aerodynamic drag—all subjects you have learned about earlier in the book. The engine applies a torque, $T_e$, to the car wheels that generates the force that moves the car forward. In Figure 8-1, the torque is applied to the front wheels of the car.

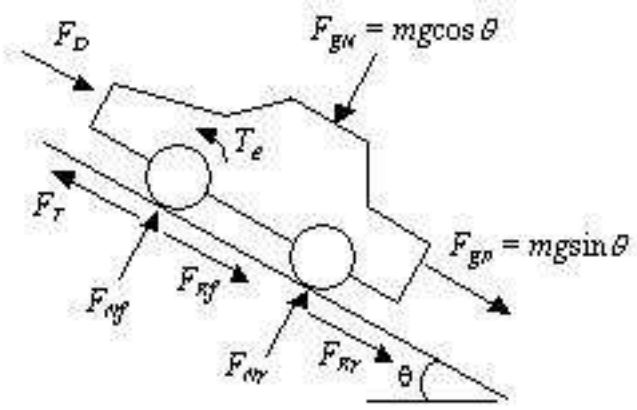


*Figure 8-1. Force balance on a car driving in a straight line on a horizontal surface*

As you can see from Figure 8-1, there are quite a few forces acting on the car, so let's go over them one by one. The force of gravity pulls the car towards the earth. It acts both normal to the slope with a force, $F_{gN} = mg\cos\theta$, as well as parallel to the slope with a force, $F_{gP} = mg\sin\theta$. Depending on whether the car is pointing uphill or downhill, the parallel component of gravitational force can either pull the car forwards or backwards.

This force of gravity in the normal direction, $F_{gN}$, is balanced by normal forces, $F_{Nf}$ and $F_{Nr}$, that act along the surfaces of the front and rear tires that are in contact with the ground. The total normal force, $F_N$, is the sum of the forces on the front and rear tires and is equal to the mass of the car multiplied by the acceleration due to gravity and the cosine of the slope angle, $\theta$.

$$F_N = F_{Nf} + F_{Nr} = mg\cos\theta \qquad (8.1)$$

The engine generates torque, which when applied to the wheels causes them to rotate. Friction between the tires and the ground resists this motion, resulting in a force applied to the tires in the direction opposite to the rotation of the tires. The force applied to the tires, $F_T$, is equal to the torque applied to the wheels, $T_w$, divided by the wheel radius, $r_w$.

$$F_T = \frac{T_w}{r_w} \qquad (8.2)$$

As we shall see in the "Gears and Wheel Torque" section a little later in this chapter, the torque applied to the wheels is generally not equal to the torque generated by the engine.

When the car is in motion, an aerodynamic drag force will develop that will resist the motion of the car. As we saw in Chapter 5, drag force can be modeled as a function of the air density, $\rho$, frontal area, $A$, the square of the velocity magnitude, $v$, and a drag coefficient, $C_D$.

$$F_D = \frac{1}{2} C_D \rho v^2 A \qquad (8.3)$$

As was the case with projectiles, aerodynamic drag acts in the opposite direction to the velocity of the vehicle. We'll discuss aerodynamic drag in more detail a little later in this chapter. The final force in the basic force diagram is due to rolling friction, which was introduced in Chapter 7. This force acts on all four wheels and resists the rolling motion of the car. The total rolling friction force, $F_R$, is equal to the total normal force, $F_N$, multiplied by the coefficient of rolling friction for the vehicle, $\mu_r$.

$$F_R = \mu_r F_N = \mu_r mg\cos\theta \qquad (8.4)$$

The net force on the car parallel to the direction the car is driving, $F_{Total}$, is equal to the sum of the forces due to engine torque, gravity, aerodynamic drag, and rolling friction.

$$F_{Total} = \frac{T_w}{r_w} - \mu_r mg\cos\theta - mg\sin\theta - \frac{1}{2}C_D\rho v^2 A \qquad (8.5)$$

For the car shown in Figure 8-1, the sign on the parallel gravity force term, $mg\sin\theta$, in Equation (8.5) is negative to indicate that it is pulling the car backwards. The acceleration of the car at any given time is equal to the net force on the vehicle divided by the mass of the vehicle, $m$.

$$a = \frac{T_w}{r_w m} - \mu_r g\cos\theta - g\sin\theta - \frac{1}{2}\frac{C_D\rho v^2 A}{m} \qquad (8.6)$$

The velocity of the vehicle at any given time can be found by integrating Equation (8.6). Before we can integrate Equation (8.6), however, we need to evaluate the torque that is applied to the wheels.

## Engine Torque and Power

When the engine runs, it generates a torque that is used to drive the car forward or backward. As we shall see later in this chapter, the torque generated by the engine is typically not the same as the torque applied to the wheels. The engine torque is a function of the rate at which the engine is turning over.

$$T_e = T_e(\Omega_e) \qquad (8.7)$$

The engine turnover rate, $\Omega_e$, in Equation (8.7) is usually expressed in terms of revolutions per minute, or *rpm*. If the engine torque is plotted as a function of engine turnover rate, the result is what is known as a **torque curve**. These curves are usually available for a given car from the manufacturer or from other sources. A typical torque curve is shown in Figure 8-2. One characteristic of engine torque is that it does not always increase with increasing engine turnover rate. The torque in the curve shown in Figure 8-2 increases with increasing *rpm* until it reaches a peak value of 309 *N-m* at about 4600 *rpm*. As the engine turnover rate increases beyond 4600 *rpm*, the torque delivered by the engine decreases.

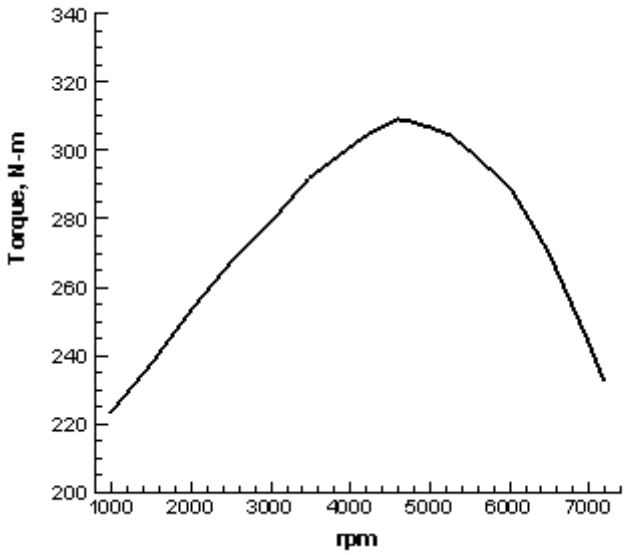


*Figure 8-2. A typical torque curve*

## The 2004 Porsche Boxster S

To help you understand how to apply the equations presented in this chapter, we will use as a test case the 2004 Porsche Boxster S sports car, a picture of which is shown in Figure 8-3. The Boxster S was chosen because it is a fast, sporty-looking car and because the author has always wanted to own one.

*Figure 8-3. The Porsche Boxster S (Photo courtesy of Tony Straughn, www.pictures-of-cars.com)*

The torque curve for a 2004 Porsche Boxster S is the one shown in Figure 8-2 and is based on data obtained from the Porsche website at `www.porsche.com`. The peak engine torque value of 309.2 *N-m* occurs when the engine is turning over at 4600 *rpm*. The torque curve

will be used a little later on to develop a model that will compute the acceleration of the Boxster S at any point in time. But first let's discuss another important topic concerning engine performance, namely engine power.

### Power and Torque

Generally, when people talk about the performance of an engine, they refer to its power rather than its torque. Recall from Chapter 3 that power is an amount of work done in a unit of time. When applied to the output of a car engine, power, $P_e$, is equal to the engine torque multiplied by the angular velocity of the engine.

$$P_e = T_e \omega_e \qquad (8.8)$$

The angular velocity of the engine, $\omega_e$, in *rad/s* can be obtained by multiplying the engine turnover rate by $2\pi$ and dividing by 60.

$$\omega_e = \frac{2\pi\Omega_e}{60} \qquad (8.9)$$

A plot of the power generated by the Boxster S as a function of engine turnover rate, known as a **power curve**, is shown in Figure 8-4. Similar to the torque curve, the peak power occurs at a certain *rpm* level and decreases after that. The peak power output of the Boxster engine is 1.92e + 5 *W* and occurs at about 6200 *rpm*. Comparing the power and torque curves in Figure 8-2 and 8-4, the peak power for the engine occurs at a higher *rpm* level than does the peak torque.

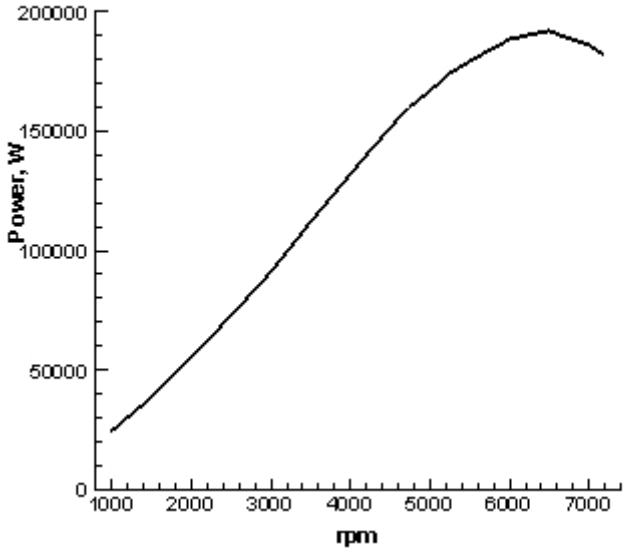


*Figure 8-4. The power curve for the 2004 Boxster S*

## Gears and Wheel Torque

The torque applied to the wheels of a car determines its acceleration. In general, the torque applied to the wheels is not the same as the engine torque because before the engine torque is applied to the wheels it passes through a **transmission**. A typical transmission cross-section

is shown in Figure 8-5. You can see that a modern transmission is quite complicated, with a lot of gears, shafts, and other strange-looking things.

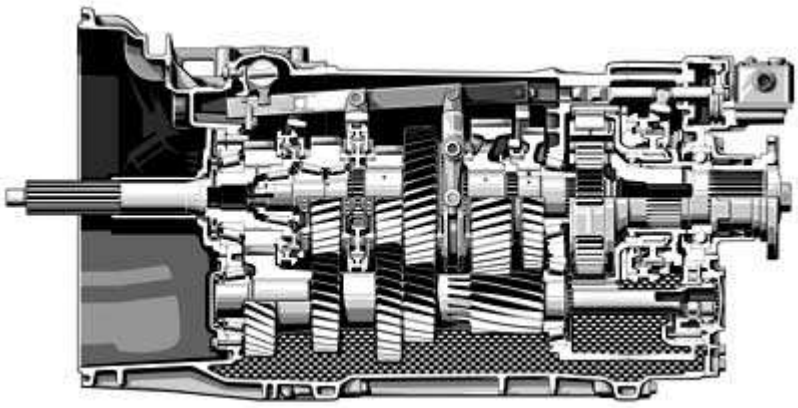

*Figure 8-5. A cross-section of a transmission (Photo courtesy of Daimler-Chrysler)*

You might ask yourself, "Why bother with a transmission? Why not connect the torque from the engine directly to the wheels?" A big reason for the existence of transmissions is performance. In looking at Figure 8-2, when the engine turnover rate is low, the torque and therefore acceleration is relatively low as well. In fact, if the engine was connected directly to the wheels, it would take the Boxster S over 16 seconds to accelerate to 100 *km/hr*. That would be pretty boring performance for a sports car.

Fortunately, the acceleration of a car can be greatly increased by using a transmission. What the gears inside the transmission do is to change the angular velocity and torque transferred from the engine. To see how this is done, consider the two gears shown in Figure 8-6. The second gear has twice the diameter of the first gear, so for every revolution the second gear makes, the first gear will make two. The second gear has half the angular velocity of the first gear. However, the torque that the second gear can exert is twice that of the first gear.

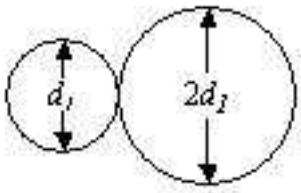


*Figure 8-6. Gears are used to change angular velocity and torque.*

The **gear ratio** between two gears is the ratio of the gear diameters. In Figure 8-6, the gear ratio between the second and first gears would be 2:1. Car transmissions will typically have between three and six forward gears and one reverse gear. There is also an additional set of gears between the transmission and the wheels. In many cars, this gearset is known as the **differential**. The gear ratio of this final gearset is known as the **final drive ratio**.

So what the transmission does is to (generally) increase the torque that comes out of the engine at the cost of reducing the gear turnover rate. To determine the acceleration of the car,

we need the torque applied to the wheels. The wheel torque, $T_w$, is equal to the engine torque, $T_e$, multiplied by the gear ratio, $g_k$, of whatever gear the car is in and the final drive ratio, $G$, of the car.

$$T_w = T_e g_k G \qquad (8.10)$$

Using Equation (8.10), the equation for the acceleration for the car shown in Figure 8-1 can be modified in terms of the engine torque and gear ratios.

$$a = \frac{T_e g_k G}{r_w m} - \mu_r g \cos\theta - g \sin\theta - \frac{1}{2}\frac{C_D \rho v^2 A}{m} \qquad (8.11)$$

Another effect of the transmission gears is to change the angular velocity of the wheel relative to the turnover rate of the engine. The relationship between the engine turnover rate, $\Omega_e$, and wheel angular velocity, $\omega_w$, becomes the following:

$$\omega_w = \frac{2\pi\Omega_e}{60 g_k G} \qquad (8.12)$$

The "60" term in Equation (8.12) is to convert the minutes in *rpm* to seconds. If the tires roll on the ground without slipping, the translational velocity of the car, $v$, can be related to the angular velocity of the wheel, and therefore to the engine turnover rate.

$$v = r_w \omega_w = \frac{r_w 2\pi\Omega_e}{60 g_k G} \qquad (8.13)$$

In looking at Equations (8.11) and (8.13), we can make the following observations about gear and final drive ratios:

* The higher the gear ratio, the higher the acceleration and the lower the car velocity for a given *rpm*.
* Increasing the final drive ratio increases the acceleration for all gears but likewise decreases the car velocity for a given *rpm* for all gears.

As an example of the gear ratios for a typical sports car, Table 8-1 shows the gear ratios for the six forward gears of the 2004 Porsche Boxster S. The final drive ratio for the car is 3.44.

*Table 8-1. Porsche Boxster S Gear Ratios*

| Gear | Gear Ratio |
|---|---|
| First | 3.82 |
| Second | 2.20 |
| Third | 1.52 |
| Fourth | 1.22 |
| Fifth | 1.02 |
| Sixth | 0.84 |

### Determining Wheel Radius

The acceleration and velocity expressions shown in Equations (8.11) and (8.13) are functions of the wheel radius, but how can this quantity be determined? Fortunately, the wheel radius can be calculated from information on the tire itself. Every tire will have a series of letters and numbers that identify the tire. For example, the front tires of the Porsche Boxster S have the identification 225/40ZR-18. The first number, 225 for the Boxster, indicates the width of the tire in millimeters. The number after the slash symbol, /, is the ratio of the tire thickness to the tire width expressed as a percentage. The letters indicate the conditions for which the tire is designed. The last two numbers represent the diameter of the wheel that fits the tire in inches. The tire radius is equal to the wheel radius added to the tire thickness. Based on the tire designation, 225/40ZR-18, the radius of the Boxster S front tire is 0.3186 *m*.

## Gear Shifting

We learned in the last section that a lower gear (with a higher gear ratio) results in a greater acceleration. So why not just stay in first gear all the time? Wouldn't that optimize the acceleration of the car? This is the flip side to the question, "Why not connect the engine directly to the wheels?" The answer is, "No, you shouldn't stay in first gear all the time." The reason is that the velocity of the car is a function of engine turnover rate and gear ratios, and there is a limit to how fast the engine can turn over.

Every car engine has a characteristic known as a **redline** *rpm* value. The engine cannot exceed this turnover rate for more than a brief period of time without causing damage to the engine. On the Porsche Boxster S, the redline value is 7200 *rpm*. Using Equation (8.13) and the data in Table 8-1, the theoretical maximum velocity for each gear at 7200 *rpm* can be computed, and the results are shown in Table 8-2.

*Table 8-2. Theoretical Maximum Velocity for Each Gear for the Boxster S*

| Gear | Maximum Velocity (*m/s*) | Maximum Velocity, (*km/hr*) |
|---|---|---|
| 1st | 18.3 | 65.8 |
| 2nd | 31.7 | 114.3 |
| 3rd | 45.9 | 165.4 |
| 4th | 57.2 | 206.0 |
| 5th | 68.5 | 246.4 |
| 6th | 83.1 | 299.3 |

We can see from Table 8-2 that although the maximum acceleration occurs in first gear, the maximum velocity the car can attain in first gear is 65.8 *km/hr*. At this point, you would reach the redline *rpm* value and have to shift into second gear, which would provide the optimum acceleration between 65.8 and 114.3 *km/hr*. At this point, the *rpm* level would reach the redline value again, and the car would have to be shifted into third gear. Most transmissions are designed so the shift point for optimum acceleration is at the redline value of the car.

Keep in mind that the values in Table 8-2 are theoretical maximum velocities. The Boxster S can't really reach 299.3 *km/hr* in sixth gear. According to the manufacturer's

specifications, the top speed of the car is "only" 266 *km/hr*. The reason the car can't reach the theoretical maximum velocity in sixth gear is because the car is also subject to the decelerating forces of aerodynamic drag and rolling friction.

Equation (8.13) can also be used to calculate what the engine *rpm* value will be after a gear shift. If the car is shifted into a higher gear, the gear ratio is reduced. If the velocity of the car is assumed to be constant before and after the gear shift, the engine *rpm* level will decline because of the lower gear ratio. The new engine turnover rate, $\Omega_e(new)$, will be equal to the engine turnover rate before the gear shift, $\Omega_e(old)$, multiplied by the ratio of the new gear ratio to the previous gear ratio.

$$\Omega_e(new) = \Omega_e(old)\frac{g_k(new)}{g_k(old)} \qquad (8.14)$$

For example, if the Boxster S shifts from first gear to second gear at 7200 *rpm*, the new *rpm* level of the engine after the gear shift will be the following:

$$\Omega_e = 7200\frac{2.2}{3.82} = 4147\ rpm \qquad (8.15)$$

This is an effect you've probably seen quite a bit. If you are driving a car and shift from a lower gear to a higher gear, the *rpm* level of the engine falls. The opposite is also true; if you shift from a higher gear into a lower gear, the *rpm* level of the engine will surge.

## Manual and Automatic Transmissions

There are two general types of transmissions. With a **manual transmission** the driver must make all of the gear shifts manually. An **automatic transmission** is one where the transmission shifts automatically. When the automatic transmission will shift varies from transmission to transmission but usually is dependent on the car velocity, the engine turnover rate, and the load being put on the engine. If you wanted to use an automatic transmission in your game programs, you could just specify when the transmission would shift. The automatic transmission in the author's 1997 Toyota Camry shifts when the engine turnover rate is between 3000 and 4000 *rpm*.

## Aerodynamic Drag

A car in motion is subject to the force of aerodynamic drag. As we know, drag acts in the opposite direction to the velocity vector of an object, so drag force will cause the car to slow down. As shown in Equation (8.3), the drag force on a car is expressed as a function of the air density, velocity of the car squared, the frontal area of the car, and a drag coefficient.

To evaluate Equation (8.3), we need to determine the drag coefficient and frontal area of the car.

### Drag Coefficients for Motor Vehicles

The drag coefficient for a car or other motor vehicle will depend on the shape of the vehicle. A sports car will have a lower drag coefficient than will a garbage truck. Typical drag coefficient ranges for several vehicle types[1] are shown in Table 8-3.

*Table 8-3. Drag Coefficients for Some Typical Vehicle Types*

| Vehicle Type | Drag Coefficient |
|---|---|
| Sports car | 0.27 – 0.38 |
| '60s muscle car | 0.38 – 0.5 |
| Sedan | 0.34 – 0.5 |
| Truck | 0.6 – 1.0 |
| Tractor-Trailer | 0.6 – 1.2 |
| Motorcycle | 0.5 – 1.0 |

While there may be some variation in vehicle drag coefficient due to its Reynolds number and other effects, for game programming purposes, you can assume a constant drag coefficient for the vehicles you are simulating. The drag coefficient for the 2004 Porsche Boxster S that we have been using as an example in this chapter is 0.31.

### Frontal Area

The drag force, as expressed by Equation (8.3), is a function of the frontal area of the car. The simplest estimate of the frontal area of a vehicle is the product of the width and height of the vehicle. This method assumes that the frontal cross-section of the vehicle is rectangular. In reality, sides of most cars are sloped so the true frontal area is less than the product of the width and height. One way to account for frontal area slope is to multiply the width and height of the vehicle by a factor between 0 and 1. For the car simulator we will develop later in this chapter, a factor of 0.85 is used to compute the frontal area of the Boxster S.

$$A = 0.85 * width * height \qquad (8.16)$$

The Boxster S has a width of 1.78 *m* and a height of 1.28 *m*. Using Equation (8.16), the frontal area of the vehicle, for purposes of computing the drag force, would be 1.94 $m^2$.

## Rolling Friction

As you learned in Chapter 7, rolling friction is a force that resists the rolling motion of an object. While it is referred to as friction, it really is a contact force caused by the deformation of the object and the surface it is rolling over. As shown in Equation (8.4), the force due to rolling friction, $F_r$, between the tires and the ground is equal to the normal force exerted on the object, $F_N$, multiplied by a coefficient of rolling friction, $\mu_r$. If the car is traveling over flat ground, the normal force will be equal to *mg*.

The value of the coefficient of rolling friction tends to be significantly lower than the coefficients of static or kinetic friction for the same object. For car tires, the coefficient of rolling friction ranges from 0.01 to 0.02.[2]

## Computing Acceleration and Velocity

In order to create a car simulation, it is necessary to determine the acceleration and velocity of the car at any point in time. The starting point for this analysis is Equation (8.11). In the preceding sections, values were presented for the coefficient of rolling friction and drag

coefficient of a car. If the slope angle, frontal area, and air density are known, the only unknown quantity in Equation (8.11) is the wheel torque, $T_w$.

According to Equation (8.10), the wheel torque is the product of the engine torque, $T_e$, the current gear ratio, $g_k$, and the final drive ratio, $G$. The engine torque, $T_e$, can be obtained from the torque curve of the engine. Torque curve data is usually presented either as a plot or as a table of numbers. For game programming purposes, the torque curve needs to be expressed as a mathematical expression. The easiest way to mathematically model a torque curve is with a series of straight lines.

For example, the torque curve for the Boxster S as shown in Figure 8-2 can be approximated by three straight lines. The first line extends from a torque value of 220 *N-m* at 1000 *rpm* up to 4600 *rpm* where the torque peaks at 309.2 *N-m*. A second line is drawn from 4600 *rpm* to the redline value of 7200 *rpm* where the torque is approximately 227 *N-m*. Below 1000 *rpm*, the idle speed of the engine, the torque will be assumed to be 220 *N-m*. The simplified torque curve is shown in Figure 8-7.

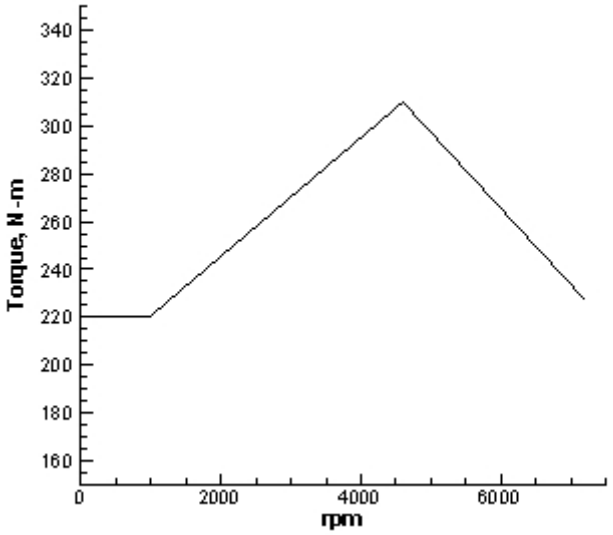


*Figure 8-7. Simplified torque curve for the Porsche Boxster S*

Using the simplified torque curve, the torque for the Boxster S can be modeled by three equations. The units for engine torque in all three equations are in *N-m*.

$$T_e = 220 \qquad \Omega_e \le 1000 \qquad (8.17a)$$

$$T_e = 0.025\Omega_e + 195 \qquad 1000 < \Omega_e < 4600 \qquad (8.17b)$$

$$T_e = -0.032\Omega_e + 457.2 \qquad \Omega_e \ge 4600 \qquad (8.17c)$$

All three of the lines described by Equations (8.17a) through (8.17c) are specific cases of the general equation for a straight line.

$$T_e = b\Omega_e + d \qquad (8.18)$$

The *b* parameter in Equation (8.18) is the slope of the line. Of course, straight lines aren't the only way to mathematically model a torque curve. Depending on the shape of the curve, a parabolic or exponential function could also be used to approximate a torque curve.

Having a mathematical expression for the torque curve is all well and good, but to solve for the acceleration of the car what we really need is an equation for the wheel torque as a function of the current velocity of the car. An equation that relates wheel torque to car velocity can be derived if the assumption is made that the tires roll without slipping. Under this condition, the velocity of the car, $v$, is equal to the wheel radius, $r_w$, multiplied by the angular velocity of the wheel, $\omega_w$. As seen in Equation (8.12), the angular velocity is a function of the engine turnover rate and the gear and final drive ratios.

$$v = r_w \omega_w = \frac{2\pi r_w \Omega_e}{60 g_k G} \qquad (8.19)$$

As a reminder, the "60" term in Equation (8.19) converts the engine turnover rate from *rpm* to *rev/s*. Plugging Equations (8.18) and (8.19) into Equation (8.11) results in an expression for the acceleration of the car as a function of the current velocity of the car.

$$a = \frac{60 g_k^2 G^2 b v}{2\pi m r_w^2} + \frac{g_k G d}{m r_w} - \frac{1}{2}\frac{C_D \rho v^2 A}{m} - \mu_r g \cos\theta - g \sin\theta \qquad (8.20)$$

Equation (8.20) looks really messy, but it's really just an algebraic equation. The constants can be grouped together to form a simpler equation in which the acceleration of the car is a function of the current velocity of the car.

$$a = \frac{dv}{dt} = c_1 v^2 + c_2 v + c_3 \qquad (8.21)$$

The constants, $c_1$, $c_2$, and $c_3$, in Equation (8.21) are the following:

$$c_1 = -\frac{1}{2}\frac{C_D \rho A}{m} \qquad (8.22a)$$

$$c_2 = \frac{60 g_k^2 G^2 b}{2\pi m r_w^2} \qquad (8.22b)$$

$$c_3 = \frac{g_k G d}{m r_w} - \mu_r g \cos\theta - g \sin\theta \qquad (8.22c)$$

In looking at Equation (8.20), we can observe what parameters influence the acceleration value of the car. Some conclusions are pretty obvious—the heavier the car, the lower the acceleration. If the gear and final drive ratios are increased, the acceleration is increased. Reducing the rolling friction of the wheels increases the acceleration.

Keep in mind that Equation (8.20) represents the maximum acceleration available at a given velocity. It's based on the wheel torque that would result if you pushed the gas pedal all the way to the floor. In real life, using the maximum acceleration all the time would be a pretty extreme way of driving. If the gas pedal was pushed only part way down, the actual torque applied to the wheels would be some fraction of the maximum possible torque. For game programming purposes, you might apply the maximum possible wheel torque if the gas pedal were pushed all the way down, half the torque if the pedal were pushed half way down, and so on.

The acceleration shown in Equation (8.20) assumes that the tires roll without slipping on the ground. In many cases, the maximum available torque will generate a force that is greater than the maximum frictional force between the tires and the ground. When this happens, the wheels won't roll without slipping; instead, the wheels will spin across the road surface in the classic "burning rubber" effect. We'll discuss tire slippage in the "Wheel Traction" section a little later in this chapter.

Another thing to remember is that the acceleration equation shown in Equation (8.20) is really an idealized case. It assumes there is no loss in engine torque as it goes through the transmission and differential. In reality, there is some loss due to friction between the mechanical parts. On the other hand, using straight lines to model the torque curve tends to underpredict the engine torque. The two assumptions would somewhat cancel each other out, and for game programming purposes, Equation (8.20) is probably sufficient.

Equation (8.20) or the alternative form shown in Equation (8.21) can be used to solve for the velocity of the car over time. It turns out that there is a closed-form solution to Equation (8.21), but it is quite messy and has different forms depending on the relative values of the $c_1$, $c_2$, and $c_3$ constants. It's easier if slightly slower to solve Equation (8.21) using our ODE solver, and that's exactly what we'll do when we develop a car simulator later in this section.

### Maximum Velocity

Equation (8.21) can be used to compute the theoretical maximum velocity that a car can achieve. The maximum velocity will be the point where the net acceleration on the car is zero. But there is a catch, because at lower gears the redline *rpm* will be reached before the net acceleration on the car reaches zero. In this case, the maximum velocity, $v_{max}$, of a car is limited by the redline *rpm* value of the engine.

$$v_{\max} = \frac{2\pi r_w \Omega_{redline}}{60 g_k G} \qquad (8.23)$$

At higher gears (with lower gear ratios) the maximum velocity of a car is drag-limited. Drag will stop the car from accelerating any further before the redline *rpm* value is reached. If you recall, the drag force is proportional to the square of the velocity of the car. As velocity increases, the aerodynamic drag increases until a velocity is reached where the torque applied to the wheels is exactly balanced by the aerodynamic drag and rolling friction experienced by the vehicle. At this point the acceleration of the car, $a$, is zero.

$$a = c_1 v^2 + c_2 v + c_3 = 0 \qquad (8.24)$$

The maximum velocity can be found using the standard equation for finding the roots of a quadratic equation.

$$v_{\max} = \frac{-c_2 \pm \sqrt{c_2^2 - 4c_1 c_3}}{2c_1} \qquad (8.25)$$

Let's use Equation (8.25) to compute the maximum velocity of the Porsche Boxster S when the car is in sixth gear. The drag coefficient of the car is 0.31, the empty mass is 1323 *kg*, and the radius of the front wheels is 0.3186 *m*. We'll assume that the coefficient of rolling friction is 0.015, the frontal area of the car is 1.94 $m^2$, and the weight of the driver is 70 *kg*. The air density value will be taken to be 1.2 $kg/m^3$. The car is assumed to be driving over flat, level ground, so the slope angle, $\theta$, is equal to zero.

The maximum velocity will occur in the higher *rpm* range, so the engine torque will be approximated by Equation (8.17c). This means that the *b* coefficient is equal to –0.032 and the *d* coefficient has a value of 457.2. In sixth gear, the maximum velocity of the car is likely to be drag-limited, so we'll use Equation (8.25) to compute the maximum velocity.

$$c_1 = -\frac{1}{2}\frac{C_D \rho A}{m} = -\frac{1}{2}\frac{0.31*1.2*1.94}{1393} = -2.59e-4 \qquad (8.26a)$$

$$c_2 = \frac{60 g_k^2 G^2 b}{2\pi m r_w^2} = -\frac{60*0.84^2*3.44^2*0.032}{2\pi*1393*0.3186^2} = -0.018 \qquad (8.26b)$$

$$c_3 = \frac{g_k G d}{m r_w} - \mu_r g = \frac{0.84*3.44*457.2}{1393*0.3186} - 0.015*9.8 = 2.83 \qquad (8.26c)$$

$$v_{\max} = \frac{0.018 \pm \sqrt{0.018^2 + 4*2.59e-4*2.83}}{-2*2.59e-4} = 75.4\ \frac{m}{s} = 271.5\ \frac{km}{hr} \qquad (8.26d)$$

The other solution to the $v_{max}$ equation gives a negative value and can be ignored. The manufacturer's published value for the top speed of the Boxster S is 266 *km/hr*. Considering the assumptions and simplifications that went into our model, it did a pretty good job of computing the top speed of the vehicle.

The value of 271.5 *km/hr* is the drag-limited maximum velocity of the Boxster S. Let's compare it to the redline-limited value calculated using Equation (8.23).

$$v = \frac{2\pi r_w \Omega_{redline}}{60 g_k G} = \frac{2\pi*0.3186*7200}{60*0.84*3.44} = 83.1\ \frac{m}{s} = 299.3\ \frac{km}{hr} \qquad (8.27)$$

So the Boxster S could go 299 *km/hr* in sixth gear if there were no aerodynamic drag and no rolling friction. To get a feeling for the relative magnitudes for some of the force terms, let's compare the magnitude of aerodynamic drag and rolling friction on the Boxster S when it has reached a speed of 271.5 *km/hr*. The drag force can be calculated from Equation (8.3).

$$F_D = \frac{1}{2}0.31*1.2*75.4^2*1.94 = 2051 N \qquad (8.28)$$

The force of rolling friction can be computed from Equation (8.4).

$$F_R = 0.015*1393*9.81 = 205\ N \qquad (8.29)$$

In comparing the results from Equations (8.28) and (8.29), the aerodynamic drag force is 10 times as large as the rolling friction force when the car is traveling at 271.5 *km/hr*. The rolling friction force is not a function of velocity. If the car were going 10 *km/hr*, the rolling friction force would still be 205 *N*, whereas the drag force would only be 2.8 *N*.

## Braking

Driving a car is not all acceleration; sometimes you need to slow down, too. In this section, we will discuss two general ways a car can slow down (and no, one of them is not running into a tree). It turns out that an engine will slow itself down just by the nature of how the cylinders move up and down inside the engine. This effect is known as **engine braking**. The

torque due to engine braking, $T_{eb}$, is modeled mathematically by a constant known as the **engine braking coefficient**, $\mu_{eb}$, multiplied by the turnover rate of the engine in *rev/s*.

$$T_{eb} = \mu_{eb} \frac{\Omega_e}{60} \qquad (8.30)$$

It can be difficult to obtain the value of the engine braking coefficient for a given car. For an F1 race car, the coefficient has a value of 0.74.[3] If this value is applied to the Boxster S, the torque due to engine braking at 6000 *rpm* is equal to 74 *N-m* corresponding to an acceleration of –0.17 $m/s^2$. If you want to include the effects of engine braking in your car simulation and don't have the precise value of engine braking coefficient for the car you are modeling, assuming a value of 0.74 is probably a reasonable estimate.

Another way a car can be slowed down is if its brakes are applied. When the driver steps on the brake pedal, a brake pad is pressed up against a flat metal disk attached to the wheel. Friction between the brake pad and disk generates a torque that slows the wheels down. The torque caused by the brake pad acts in the opposite direction that the wheel is rotating.

It can be difficult to find information about brake torque for a given car. Information on braking is usually presented as the distance it takes to brake a car from an initial velocity to a full stop. For example, the Boxster S requires 34 *m* to brake from a velocity of 26.8 *m/s* (60 *mi/hr*) to a full stop. If braking distance data is available, the braking acceleration, $a_b$, can be obtained from the Newtonian mechanics as a function of the initial velocity, $v_0$, and braking distance, $x$.

$$a_b = -\frac{v_0^2}{2x} = -10.4 \frac{m}{s^2} \qquad (8.31)$$

Keep in mind when looking at the results in Equation (8.31), the driver who performed this test was trying to get the smallest braking distance possible, so he or she probably slammed on the brakes. The –10.4 value therefore can be considered the maximum braking deceleration for the Boxster S.

For game programming purposes, if the value for the brake torque for a given car isn't available, you can calculate the braking acceleration from braking distance data and apply that in your code. Keep in mind that the braking distance is based on the maximum braking force—if you slammed on the brakes. To simulate a more gentle braking action, you could simply take some fraction of the maximum braking acceleration.

## A Car Simulator

Let's take what we've learned about modeling the physics of cars and develop a car simulator. The GUI display for the car simulator is shown in Figure 8-8. At the top of the simulator is a display area that shows a picture of the car. Above the car are two rectangular markers. When the simulator runs, the car location remains fixed on the screen, and the markers move from left to right to simulate the forward motion of the car. The speed at which the markers move is proportional to the speed of the car.

On the left-hand side of the GUI are three radio buttons that determine whether the car is accelerating, cruising at a constant velocity, or slowing down by braking. The radio buttons are mutually exclusive, so only one may be selected. Below the radio buttons are five buttons. The Start button starts or resumes the simulation. The Shift Up and Shift Down buttons cause the car to shift gears up or down. The Stop button stops the car, shifts the car into first gear, and lowers the engine turnover rate to 1000 *rpm*. The Reset button does the same things as the Stop button, but it also resets the distance and time values to zero.

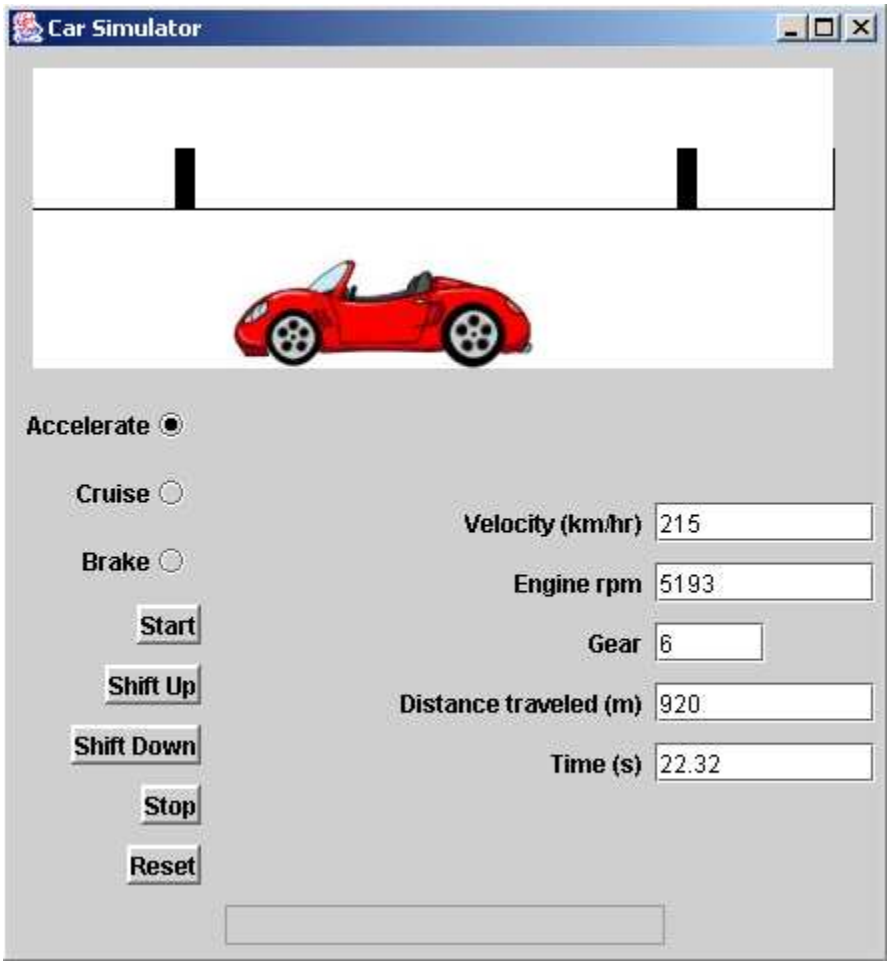


*Figure 8-8. Car Simulator screen shot*

On the right-hand side of the display are text fields that show the current velocity and engine turnover rate of the car as well as what gear the car is currently in. Also included are text fields that display the distance the car has traveled and the total elapsed time of the simulation. At the bottom of the display is a text field used to present warning messages. If the engine turnover rate exceeds the redline *rpm* value, a message to that effect is shown in the text field. If the engine turnover rate exceeds 8000 *rpm*, you've "blown" the engine and the simulation will stop, as shown in Figure 8-9.

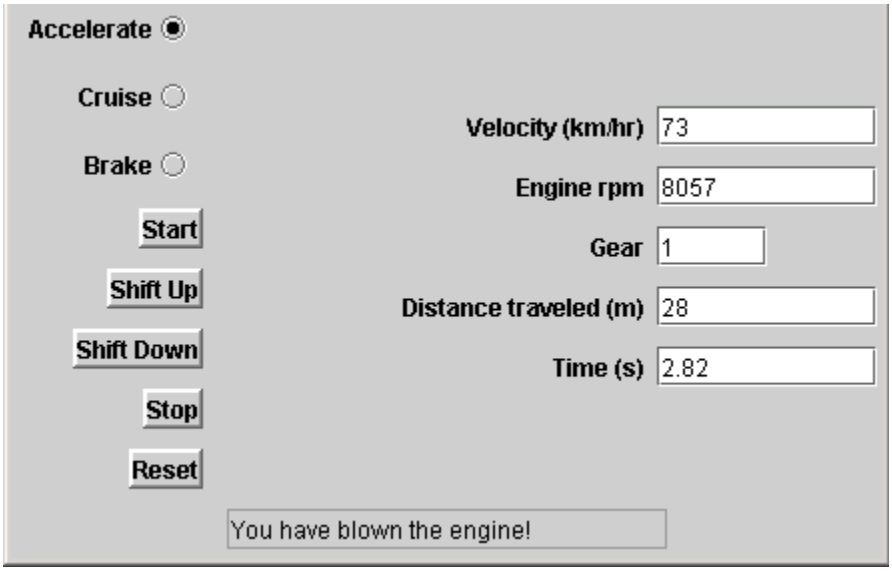


*Figure 8-9. If the rpm exceeds 8000, you've blown the engine.*

The GUI for this car simulation is pretty primitive, but the physics inside it are real and are based on the equations developed in this chapter. The effects of aerodynamic drag and rolling friction are included in the model. Some simplifications were made, however. The car is assumed to be driving in a straight line on flat ground. When the car is braking, the acceleration due to braking is assumed to be a constant –5.0 $m/s^2$ at all times. We are also not modeling the reverse gear in this simulation, so the car is either stopped or is moving forward.

The tires of the car in the Car Simulator are assumed to roll without slipping at all times. In real life, if too much torque is applied to the wheels, the maximum frictional force between the tires and the ground will be exceeded, and the tires will slip along the ground—the "burning rubber" effect. We'll explore how to model tire slippage in more detail in the "Wheel Traction" section a little later in this chapter.

Creating the car simulation requires two general types of classes—one representing the car and the other defining the GUI. We'll start by discussing the classes that represent the cars. Two car classes will be written. The first class is called the `Car` class and represents a generic car. The `Car` class will declare the fields and methods that are common to all cars. The `Car` class will be written as a subclass of the `DragProjectile` class so it can reuse the code declared in the `DragProjectile`, `SimpleProjectile`, and `ODE` classes.

```
public class Car extends DragProjectile
{
  private double muR;
  private double omegaE;
  private double redline;
  private double finalDriveRatio;
  private double wheelRadius;
  private int gearNumber;     //  Gear the car is in
  private int numberOfGears;  //  Total number of gears
  private String mode;
  private double[] gearRatio;  //  Gear ratios
```

The `muR` field represents the coefficient of rolling friction. The `gearNumber` field is the gear the car is currently in. The `numberOfGears` field contains the total number of forward gears in the transmission. The `mode` field defines whether the car is accelerating, cruising at constant velocity, or braking. The `omegaE` field is the engine turnover rate in *rpm*. The names of the other `Car` class fields are self-explanatory.

The `Car` constructor is used to initialize the fields in the `Car`, `DragProjectile`, `SimpleProjectile`, and `ODE` classes. The first thing the constructor does is to call the `DragProjectile` class constructor.

```
  //  The Car constructor calls DragProjectile constructor and
  //  then initializes the car-specific variables.
  public Car(double x, double y, double z,
             double vx, double vy, double vz,
             double time, double mass, double area,
             double density, double Cd, double redline,
             double finalDriveRatio, double wheelRadius,
             int numberOfGears) {

    super(x, y, z, vx, vy, vz, time, mass, area,
          density, Cd);
```

The `Car` constructor then initializes the fields declared in the `Car` class with values passed to the constructor. The size of the `gearRatio` array is set according to the value of the `numberOfGears` field. The gear ratios are initially given dummy values of 1.

```
    //  Initialize some fields based on values passed
    //  to the constructor.
    this.redline = redline;           //  Redline rpm
    this.finalDriveRatio = finalDriveRatio;  //  Final drive ratio
    this.wheelRadius = wheelRadius;   //  Wheel radius
    this.numberOfGears = numberOfGears;   //  Number of gears

    //  Initialize the array that stores the gear ratios.
    //  The array is shifted so the first index in the
    //  array correpsonds to first gear and so on.
    //  Give all gear ratios the dummy value of 1.0
    gearRatio = new double[numberOfGears + 1];
    gearRatio[0] = 0.0;
    for(int i=1; i<numberOfGears+1; ++i) {
      gearRatio[i] = 1.0;
    }
```

Some of the `Car` field values will be set to the same value for all car classes including the fields that represent the coefficient of rolling friction, initial engine *rpm*, starting gear number, and mode.

```
    //  Set some fields the same for all cars.
    muR = 0.015;             //  Coefficient of rolling friction
    omegaE = 1000.0;         //  Engine rpm
    gearNumber = 1;          //  Gear the car is in
    mode = "accelerating";   //  Accelerating, cruising, or
                             //  braking
  }
```

After the constructor, the `Car` class declares a series of get/set methods to access or change the value of the fields declared in the class. Only some of the get/set methods are shown here. Download the complete code listing from the Apress website to see all of the get/set methods.

```
//  These methods return the value of the fields
//  declared in this class.
public double getMuR() {
  return muR;
}

public double getFinalDriveRatio() {
  return finalDriveRatio;
}

//  Other get methods not shown ...

public void setOmegaE(double value) {
  omegaE = value;
}

//  Other set methods not shown ...
```

One of the features of this car simulation is that you can shift gears. This functionality is implemented in the `shiftGear` method. The first thing the method does is to determine whether the desired shift is outside the possible range of gear numbers, in which case the method returns. If the shift is possible, the value of the `gearNumber` field is changed, and the new engine turnover rate is computed by multiplying the old turnover rate by the ratio of the new gear ratio to the old gear ratio.

```
//  This method simulates a gear shift.
public void shiftGear(int shift) {
  //  If the car will shift beyond highest gear, return.
  if ( shift + getGearNumber() > getNumberOfGears() ) {
    return;
  }
  //  If the car will shift below 1st gear, return.
  else if ( shift + getGearNumber() < 1 ) {
    return;
  }
  //  Otherwise, change the gear and recompute
  //  the engine rpm value.
  else {
    double oldGearRatio = getGearRatio();
    setGearNumber(getGearNumber() + shift);
    double newGearRatio = getGearRatio();
    setOmegaE(getOmegaE()*newGearRatio/oldGearRatio);
  }

  return;
}
```

Since the ODE solver will be used to update the position and velocity of the car, the `Car` class has to declare a `getRightHandSide` method to define the right-hand sides of the equations

to be solved. The first part of the `getRightHandSide` method is similar to that found in many of the classes we've written previously. The intermediate values of location and velocity for the car are computed. In this simulation, we only are concerned with the x-components of location and velocity, but the y- and z-components are included in the method to make the class easily extendable to a car traveling in all three directions.

```
public double[] getRightHandSide(double s, double q[],
                                 double deltaQ[], double ds,
                                 double qScale) {
  double dQ[] = new double[6];
  double newQ[] = new double[6];

  //  Compute the intermediate values of the
  //  dependent variables.
  for(int i=0; i<6; ++i) {
    newQ[i] = q[i] + qScale*deltaQ[i];
  } getRightHandSide
```

The next thing the method does is to define the torque curve. We're going to use the simplified torque curve shown in Figure 8-7 where three straight lines approximate the torque curve. The three lines are defined in Equations (8.17a) through (8.17c). Which line to use depends on the engine turnover rate.

```
  //  Compute the constants that define the
  //  torque curve line.
  double b, d;
  if ( getOmegaE() <= 1000.0 ) {
    b = 0.0;
    d = 220.0;
  }
  else if ( getOmegaE() < 4600.0 ) {
    b = 0.025;
    d = 195.0;
  }
  else {
    b = -0.032;
    d = 457.2;
  }
```

The `getRightHandSide` method computes the total drag and rolling friction forces from Equations (8.3) and (8.4). Because the field that represents the gravitational acceleration, `G`, was given a value of –9.81 in the `SimpleProjectile` class, the value of the rolling friction force will be negative.

```
  //  Compute velocity magnitude.
  double vx = newQ[0];
  double vy = newQ[2];
  double vz = newQ[4];
  double v = Math.sqrt(vx*vx + vy*vy + vz*vz) + 1.0e-8;

  //  Compute the total drag force.
  double Fd = 0.5*getDensity()*getArea()*getCd()*v*v;

  //  Compute the force of rolling friction. Because
```

```
    //  the G constant defined in the SimpleProjectile
    //  class has a negative sign, the value computed here
    //  will be negative.
    double Fr = getMuR()*getMass()*G;
```

The final part of the `getRightHandSide` method defines the right-hand side of the ODEs that describe the motion of the car. If the car is accelerating, the acceleration of the car is computed from Equation (8.20). If the car is braking and the velocity is positive, the acceleration of the car is set to $-5.0\ m/s^2$. If the car is cruising at constant velocity, the acceleration is set to zero. The equations for the y- and z-components of velocity are all set to zero.

```
    //  Compute the right-hand sides of the six ODEs
    //  newQ[0] is the intermediate value of velocity.
    //  The acceleration of the car is determined by
    //  whether the car is accelerating, cruising, or
    //  braking. The braking acceleration is assumed to
    //  be a constant -5.0 m/s^2.
    if ( mode.equals("accelerating") ) {
      double c1 = -Fd/getMass();
      double tmp = getGearRatio()*getFinalDriveRatio()/
                   getWheelRadius();
      double c2 = 60.0*tmp*tmp*b*v/(2.0*Math.PI*getMass());
      double c3 = (tmp*d + Fr)/getMass();
      dQ[0] = ds*(c1 + c2 + c3);
    }
    else if ( mode.equals("braking") ) {
      //  Only brake if the velocity is positive.
      if ( newQ[0] > 0.1 ) {
        dQ[0] = ds*(-5.0);
      }
      else {
        dQ[0] = 0.0;
      }
    }
    else {
      dQ[0] = 0.0;
    }

    dQ[1] = ds*newQ[0];
    dQ[2] = 0.0;
    dQ[3] = 0.0;
    dQ[4] = 0.0;
    dQ[5] = 0.0;

    return dQ;
  }
}
```

The `Car` class represents a generic car and declares the fields and methods common to all cars. Classes for specific car types can be written as subclasses of the `Car` class and can define the field values for a specific car. For example, the Car Simulator will simulate the Boxster S, so we will write a `BoxsterS` class to represent that specific type of car.

Because Java is an object-oriented programming language, writing the `BoxsterS` class is really easy because almost all of the functionality the `BoxsterS` class needs has already been defined in earlier classes. All the `BoxsterS` class needs to do is to define fields that contain the specs for the Boxster S. The `BoxsterS` constructor simply calls the `Car` constructor with the appropriate Boxster S values and then sets the gear ratio value by calling the `setGearRatio` method.

```
public class BoxsterS extends Car
{
  //  The BoxsterS constructor calls the Car constructor
  //  and then sets the gear ratios for the BoxsterS.
  //  Here are some specs for the BoxsterS
  //  mass = 1393.0 kg (with 70 kg driver)
  //  area = 1.94 m^2
  //  Cd = 0.31
  //  redline = 7200 rpm
  //  finalDriveRatio = 3.44
  //  wheelRadius = 0.3186
  //  numberOfGears = 6;

  public BoxsterS(double x, double y, double z, double vx,
             double vy, double vz, double time, double density) {

    super(x, y, z, vx, vy, vz, time, 1393.0, 1.94,
          density, 0.31, 7200.0, 3.44, 0.3186, 6);

    //  Set the gear ratios.
    setGearRatio(1, 3.82);
    setGearRatio(2, 2.20);
    setGearRatio(3, 1.52);
    setGearRatio(4, 1.22);
    setGearRatio(5, 1.02);
    setGearRatio(6, 0.84);
  }
}
```

Now that the `Car` and `BoxsterS` classes are defined, they can be incorporated into the Car Simulator GUI. The class that implements the GUI is named `CarSimulator`. As with the other GUIs in this book, we will not go over every detail of the `CarSimulator` class, but you are encouraged to download the source code from the Apress website. As with most of the other sample games in this book, the `CarSimulator` class makes use of a `Timer` object to control the execution speed of the game. Among the fields declared in the `CarSimulator` class is a `BoxsterS` object that represents the car being modeled in the simulation.

```
import javax.swing.*;
import java.awt.*;
import javax.swing.border.BevelBorder;
import java.awt.event.*;
import javax.swing.Timer;

public class CarSimulator extends JFrame implements ActionListener
{
  //  Other field declarations not shown ...
```

```
private BoxsterS car;
```

When the Start button is pressed, the `start` method is called on the `Timer` object to start the simulation. The `Timer` object is set up to call the `actionPerformed` method every 0.05 seconds. The first thing the `actionPerformed` method does is to determine whether the car is accelerating, cruising at a constant speed, or braking, and sets the value of the `mode` field accordingly.

```
//  This ActionListener is called by the Timer
class GameUpdater implements ActionListener {
  public void actionPerformed(ActionEvent event) {
    //  Figure out if the car is accelerating,
    //  cruising, or braking, and set the mode of
    //  the car accordingly.
    if ( accelButton.isSelected() == true ) {
      car.setMode("accelerating");
    }
    else if ( cruiseButton.isSelected() == true ) {
      car.setMode("cruising");
    }
    else {
      car.setMode("braking");
    }
```

The `updateLocationAndVelocity` method is called to update the location and velocity of the car. This method is implemented in the `DragProjectile` class, but since the `BoxsterS` class is a subclass of `DragProjectile`, the method can be accessed inside the `BoxsterS` class.

```
    //  Update the car velocity and position at the next
    //  time increment.
    double timeIncrement = 0.06;
    car.updateLocationAndVelocity(timeIncrement);
```

The new *rpm* value of the engine is computed using Equation (8.13). If the value exceeds the redline value for the car, a warning message is displayed. If the *rpm* value exceeds 8000, the engine is blown and the simulation stops.

```
    //  Compute the new engine rpm value.
    double rpm = car.getVx()*60.0*car.getGearRatio()*
        car.getFinalDriveRatio()/(2.0*Math.PI*car.getWheelRadius());
    car.setOmegaE(rpm);

    //  If the rpm exceeds the redline value, put a
    //  warning message on the screen. First, clear the
    //  message text field of any existing messages.
    messageTextField.setText("");
    if ( car.getOmegaE() > car.getRedline() ) {
      messageTextField.setText("Warning: Exceeding redline rpm");
    }
    if ( car.getOmegaE() > 8000.0 ) {
      messageTextField.setText("You have blown the engine!");
      gameTimer.stop();
    }
```

Two rectangular markers are used to simulate the motion of the car. The car stays in a set location in the GUI display, and the rectangular markers move from right to left. The location of the markers is updated based on the velocity of the car. The factor of 10 is a scaling factor that relates the size of the car image to the actual length of the car. After the new marker locations have been determined, the GUI display is updated.

```
      //  Update the location of the rectangular markers.
      rectangleOneX = rectangleOneX + 10.0*car.getVx()*timeIncrement;
      rectangleTwoX = rectangleTwoX + 10.0*car.getVx()*timeIncrement;

      //  If the markers have gone off the display, move them
      //  back to zero.
      if ( rectangleOneX > 401.0 ) {
        rectangleOneX = 0.0;
      }
      if ( rectangleTwoX > 401.0 ) {
        rectangleTwoX = 0.0;
      }

      //  Update the display.
      resetDisplay();
    }
  }
}
```

Play around with the Car Simulator. Switch the mode to "accelerate" and push the Start button to start the simulation. Be sure to watch the *rpm* value and shift up before it hits the redline value. When you get to sixth gear, let the car run and see what the maximum velocity of the car is. Then select the "brake" mode and watch the car slow down. If you want to play with shifting gears up and down, you can set the mode to "cruise", which will hold the velocity constant. If for some reason the entire GUI is not rendered, press the Reset button to redraw the GUI.

Keep in mind that the Car Simulator uses the maximum possible acceleration of the car according to Equation (8.20). In the Car Simulator, you're driving "pedal to the metal."

## Wheel Traction

Up to this point, we have been modeling the car tires under the assumption they roll without sliding along the ground. In certain situations, however, this may not be the case. If a car is stopped and the accelerator is pushed to the floor, the tires may spin in place for a moment before the car starts to move forward. If a car tries to drive around a corner at too high a speed, it may slide outward.

If you recall from the beginning of this section, the tires move the car forward because of the friction that exists between the tire and the surface of the ground. This force, known as the **traction force**, is equal to the normal force exerted on the tire multiplied by the coefficient of friction between the tire and road.

$$F_T = \mu_k F_N = \mu_k mg\cos\theta \qquad (8.32)$$

The traction force, shown in Equation (8.32), is the maximum force that can be applied to the tire for it to roll without sliding on the ground. The traction force for a given car is typically determined by putting the car through what is known as a **skidpad** test. The car is

driven around a level, circular track. The velocity of the car is increased until centripetal acceleration of the car is equal to the traction force of the tires.

$$\mu_k mg = m\frac{v^2}{r} \qquad (8.33)$$

The *r* parameter in Equation (8.33) is the radius of the track. If the velocity of the car increases beyond this point, the centripetal force is greater than the traction force, and the car begins to slide outward on the track. The results of the skidpad test are usually expressed in terms of the maximum acceleration the tires can be subject to without sliding. On dry pavement, the 2004 Porsche Boxster S has a maximum tire acceleration of 0.91*g*, where *g* is the gravitational acceleration.

This limiting acceleration applies to straight-line motion as well. If the torque applied to the wheels results in an acceleration that is greater than the maximum tire acceleration, the tires will spin against the ground. The maximum tire acceleration is therefore a limiting value on the acceleration of the car. No matter how much torque the engine is applying to the wheels, the acceleration of the car won't be greater than the maximum tire acceleration.

For game programming purposes, the implementation of wheel traction effects is fairly straightforward. The first thing to do is to compute the force applied to the wheels from the engine, which is a function of the engine torque, gear ratios, and wheel radius.

$$F_T = \frac{T_w}{r_w} = \frac{T_e g_k G}{r_w} \qquad (8.34)$$

Next we compute the maximum frictional force from Equation (8.32). If the engine torque force is less than the maximum frictional force, the engine torque force is used in the equations of motion. If the engine torque force is greater than the maximum frictional force, then the wheels are sliding, and the maximum frictional force should be used in the equations of motion.

The value of the coefficient of friction in Equation (8.32) depends on the condition of the tires and the surface on which the car is driving. A bald tire will have a lower coefficient of friction than will a tire with a normal tread. A tire will have a lower coefficient of friction on ice than it will on dry pavement.

One final note about maximum tire acceleration is that it applies to the total acceleration of the car. A car going around a curve will experience a centripetal acceleration. The total acceleration of the car is equal to the square root of the sum of the squares of the centripetal and straight-line acceleration of the car.

$$a_{total} = \sqrt{a_{straightline}^2 + \left(\frac{v^2}{r}\right)^2} \qquad (8.35)$$

One conclusion from Equation (8.35) is that a car that is accelerating into a curve is more likely to skid than a car traveling at a constant velocity around the same curve.

## Driving Around Curves

Up to this point the discussion on the physics of cars has focused on the straight-line motion of a car. Of course, cars can't drive in a straight line forever. At some point they need to turn or drive around a curve. Modeling a car driving around a curve can be separated into two areas depending on whether the car is performing a high-speed or low-speed turn.

We'll start with the subject of a car driving around a curve at low speeds. As you would expect, modeling low-speed curves is easier than modeling high-speed curves because some factors, such as centripetal acceleration, can be ignored. The wheels can be assumed to be rolling without slipping. Consider the car shown in Figure 8-10. The front wheels of the car are turned at an angle, $\delta$, such that the car is making a right turn. If the car is traveling at a constant speed, it will drive in a circle of radius, $r_c$.

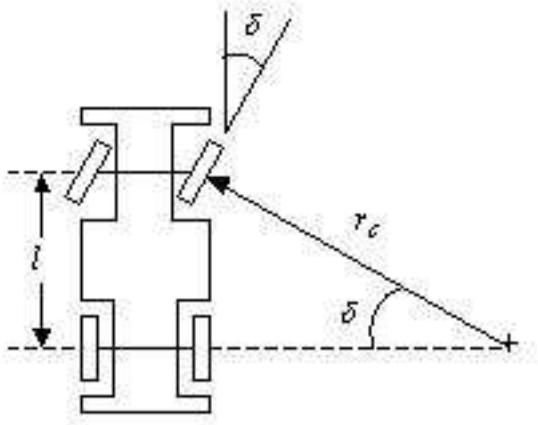


*Figure 8-10. A car making a turn at low speeds*

The center of the circle that the car is traveling on is located at the intersection of lines drawn perpendicular to the front and rear right wheels. The radius of the circle can be found from trigonometric relations. The distance from the centers of the front and back wheels, $l$, is known as the **wheelbase**. The ratio of the wheelbase to the circle radius is equal to the sine of the wheel angle, $\delta$. Rearranging this relation results in an equation for the circle radius.

$$r_c = \frac{l}{\sin\delta} \qquad (8.36)$$

Another important quantity to determine is the rate that a car will make the turn—that is, the angular velocity of the car during its turn. If the wheels are rolling without friction, the angular turn velocity, $\omega_t$, is equal to the translational velocity magnitude, $v$, of the car divided by the turn radius.

$$\omega_t = \frac{v}{r_c} \qquad (8.37)$$

Using Equation (8.36), the angular turn velocity can be expressed in terms of the wheelbase and wheel angle.

$$\omega_t = \frac{v\sin\delta}{l} \qquad (8.38)$$

Equations (8.36) and (8.38) provide all the information needed to model a low-speed turn. The turn radius can be determined from the wheelbase and wheel angle. The car then travels along this circle at an angular velocity determined from Equation (8.38). Let's look at an example to see how it all fits together. Let's say the driver of a Boxster S car wants to perform a low-speed 90-degree turn at a translational velocity of 10 *m/s* (36 *km/hr*). To make

this turn, the wheels are turned at an angle of 10 degrees. The wheelbase of the Boxster S is 2.41 *m*. What is the radius of the turn, and how long will it take the car to make the turn?

The turn radius can be computed from Equation (8.36).

$$r_c = \frac{2.41}{\sin\left(\frac{10\pi}{180}\right)} = 13.9\,m \qquad (8.39)$$

The time required to make a 90-degree turn is equal to the number of radians in the turn, $\pi/2$, divided by the angular turn velocity.

$$t = \frac{\pi}{2\omega_t} = \frac{\pi l}{2v\sin\delta} = \frac{\pi * 2.41}{2 * 10 * \sin\left(\frac{10\pi}{180}\right)} = 2.2\,s \qquad (8.40)$$

### High-Speed Turns

A general model for describing high-speed car turns is complicated by several factors. For one thing, as the car goes around the curve, the centripetal force experienced by the car may cause the tires to slide outward. In other words, the tires will have a velocity component normal to the direction in which they are rotating. The normal force component can also generate a torque about the center of mass of the car, causing the entire vehicle to rotate. You have probably seen this effect in watching a car take a high-speed turn where the back end of the car slides outward or "fishtails."

The simplest way to model high-speed turns is to compute the lateral force, $F_{lateral}$, on the car as being equal to the difference between the centripetal force on the car and the frictional force acting on the tires.

$$F_{lateral} = \frac{mv^2}{r_c} - \mu_k mg\cos\theta \qquad (8.41)$$

The angle $\theta$ is the angle of any slope that the car might be driving on. In Equation (8.41), a positive lateral force is one acting outwards. Because the frictional force will never exceed the centripetal force (the car won't be sucked into the center of the turn circle), the lateral force term will always be greater than or equal to zero.

Equation (8.41) provides a rough approximation to lateral force, but it doesn't model effects such as fishtailing or spinouts when a car tries to take a curve too fast. To get the more sophisticated high-speed turning effects requires that the lateral force be evaluated for each tire as it goes around the curve. This analysis is pretty complicated, involving concepts such as wheel slip angles, and is beyond the scope of this book.

## Modeling Car Crashes

As we all know, cars sometimes run into things. In real life, hitting something with your car is generally a bad thing to have happen to you. In car simulations, sometimes it seems like half the fun is running into or bouncing off of other objects. We learned about the basics of collision modeling in Chapter 6, and many of the same concepts can be applied to cars. Cars are not solid blocks of metal. When they hit something, unless it is at very low speeds, the body of the car will crumple as a result of the collision. The collision is inelastic because

some of the kinetic energy of the car and whatever it hits will be converted into work that is performed in damaging the car.

In Chapter 6, equations were presented to compute the post-collision velocities of two objects in the direction of the line of action of the collision. Those expressions are repeated here in Equations (8.42a) and (8.42b). The post-collision velocities, $v'_1$ and $v'_2$, are functions of the masses of the two objects, $m_1$ and $m_2$, the pre-collision velocities, $v_1$ and $v_2$, and the coefficient of restitution, $e$. One of the objects will be the car. The other object could be almost anything—another car, a tree, a fast food restaurant, and so on.

$$v'_1 = \frac{m_1 - em_2}{m_1 + m_2} v_1 + \frac{(1+e)m_2}{m_1 + m_2} v_2 \qquad (8.42a)$$

$$v'_2 = \frac{(1+e)m_1}{m_1 + m_2} v_1 + \frac{m_2 - em_1}{m_1 + m_2} v_2 \qquad (8.42b)$$

If any part of the car is deformed during the collision, then the collision is inelastic, and the coefficient of restitution will be less than one. As a reminder, a more extensive discussion of elastic and inelastic collision can be found in Chapter 6. An extreme case for the car collision would be if the collision were completely inelastic, meaning that the coefficient of restitution is equal to zero. In this case, the car and the object it collided with would stick together, and they would have the same post-collision velocity shown in Equation (8.43).

$$v'_1 = v'_2 = \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2} \qquad (8.43)$$

In most cases, the collision won't be completely inelastic, and the coefficient of restitution will have some nonzero value. The car will suffer a certain amount of damage, and it will bounce off the object it hits. The F1-Spirit Formula One racing game uses a value of 0.25 for the coefficient of restitution.

# Motorcycles

Motorcycles are another type of motor vehicle that can be used to create exciting game scenarios. Generally speaking, motorcycles are lighter, more agile, and have greater acceleration potential than cars, but much of the general physics to describe the acceleration and braking is the same between motorcycles and cars. Motorcycle engines generate torque that is applied to the back wheel, and friction between the wheel and the ground propels the motorcycle forward. Motorcycles are subject to the forces of aerodynamic drag, rolling friction, and traction, just as cars are.

Table 8-4 compares some physical and performance characteristics between a sports car and a performance motorcycle. The sports car is the 2004 Porsche Boxster S we used in the "Cars" section of this chapter. The motorcycle is the 2004 Honda CBR1000RR. As you would expect, the mass of the motorcycle is considerably less than that of the car. The Boxster S has a higher top speed, but the motorcycle has greater acceleration. It can reach 100 *km/hr* in a little over half the time it takes the car. The car engine is more powerful in terms of peak torque, but the motorcycle engine has a much higher redline *rpm* value (which is one reason why the acceleration potential is higher).

*Table 8-4. A Comparison of Motorcycle and Car Characteristics*

| Quantity | Honda CBR1000RR | Porsche Boxster S |
|---|---|---|
| Vehicle mass (*kg*) | 180 | 1323 |
| 0–100 *km/hr* (*sec*) | 2.95 | 5.5 |
| Top speed (*km/hr*) | 225 | 266 |
| Redline *rpm* | 11650 | 7200 |
| Peak engine torque (*N-m*) | 106 @ 8500 *rpm* | 310 @ 4600 *rpm* |
| Peak engine horsepower | 153.5 @ 11000 *rpm* | 258 @ 6400 *rpm* |

As we said before, many similarities exist in the physics that describe the motion of a motorcycle and car. There is one important area of difference that we will explore in a little more detail—how a motorcycle turns.

## Turning a Motorcycle

Turning a car is pretty straightforward. The wheels are turned in the direction of the turn. If you try to turn in this manner on a motorcycle, except at very low speeds, you will crash the bike. The reason is an effect called **gyroscopic precession**. When the wheel of the bike is turned in one direction, a torque is applied to the wheel in the opposite direction, causing the bike to lean. In other words, if you turn the front wheel of a bike to the left, the bike will lean to the right and vice versa. Because of this effect, if you try to turn the front wheel of a motorcycle in the direction of the turn, you will fall forward off the bike.

---

Tidbit It's easy to demonstrate gyroscopic precession, and the bike doesn't have to be moving to do it. Stand your motorcycle or bicycle straight up and turn the front wheel 90 degrees in either direction. The bike will lean in the opposite direction.

---

The secret to successfully turning a motorcycle at higher speeds is to lean *into* the turn as shown in Figure 8-11. This type of leaning stabilizes the motion of the motorcycle during the turn. There are several ways to get a bike to lean into a turn. The first technique makes use of gyroscopic precession and is known as **countersteering**. To initiate a turn, the driver must turn the front wheel in the *opposite* direction of the turn. This maneuver may seem strange, but remember that turning the front wheel in the opposite direction of the turn causes the bike to lean into the turn. The proper lean can also be created or augmented by having the driver lean his shoulders into the direction of the turn.


*Figure 8-11. To successfully turn, a motorcycle must lean into the turn. (Photo courtesy of Brett McLeod)*

The mathematical equations that describe the forces and moments that exist during a countersteered motorcycle turn are quite complicated, with various moments of inertia terms and angular velocities. Unless you are building a very detailed motorcycle simulation, there is probably no reason to try to include that level of complexity in your game. The way to include countersteering is probably more as a visual effect than anything else. When the motorcycle riders in your games make a turn, have the motorcycle lean into the turn.

# Adding Sophisticated Effects to the Car or Motorcycle Models

This chapter has covered the basics of modeling the motion of cars and motorcycles. As you might imagine, certain sophisticated physical effects that govern the motion of cars and motorcycles were not covered in detail. This chapter already mentioned that a true representation of the lateral force experienced by a car during a turn involves analyzing the lateral force experienced by each of the four wheels. Another effect that happens during a turn is that weight is shifted to the outside wheels.

To add more sophisticated effects to your car or motorcycle model, use the same sequence of steps that have been used to come up with the basic model. First, create a force diagram to determine what forces and torques act on the vehicle and the direction in which they act. Second, come up with equations that describe the motion of the vehicle based on the force diagram. Finally, code up and solve the equations of motion using an ODE solver.

# Summary

In this chapter, once again armed with a basic knowledge of Newtonian mechanics and kinematics, you learned the basic physics behind the forces and accelerations that act upon cars and motorcycles. Many of the concepts covered in this chapter are equally applicable to other types of motor vehicles such as snowmobiles or tanks. You should now be able to create fun and realistic motor vehicle simulations.

Some of the specific things you learned in this chapter include the following:

* How transmission gears are used to increase engine torque
* How to model aerodynamic drag and rolling friction for a car
* What the redline *rpm* value is and its effect on car performance
* How the wheel torque is function of the engine torque, the gear ratio, and the final drive ratio
* How wheel traction limits the maximum acceleration of a car
* How to calculate the turn radius and turn rate for a car driving around a curve
* How motorcycles turn by leaning into the turn

# References


1. B. Bowling, "Air Drag Coefficients and Frontal Area Calculation," www.bgsoflex.com/airdragchart.html.
2. C.E. Mungan, "Rolling Friction of a Free Wheel," http://usna.edu/Users/physics/mungan/Scholarship/RollingFriction.pdf.
3. R. van Gaal, "Car Physics Basics," www.racer.nl/reference/carphys.htm#enginebraking.