Quick Start Guide Version 7.1.1

CarMaker®

SOLUTIONS FOR VIRTUAL TEST DRIVING

The information in this document is furnished for informational use only, may be revised from time to time, and should not be construed as a commitment by the IPG Automotive Group. IPG Automotive Group assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

This document contains proprietary and copyrighted information and may not be copied, reproduced, translated, or reduced to any electronic medium without prior consent, in writing, from IPG Automotive Group.

© 1999 - 2018 by IPG Automotive Group – www.ipg-automotive.com
All rights reserved.

FailSafeTester, IPGCar, IPGControl, IPGDriver, IPGEngine, IPGGraph, IPGKinematics, IPGLock, IPGMotorcycle, IPGMovie, IPGRoad, IPGRoaddata, IPGTire, IPGTrailer, IPGTruck, RealtimeMaker, Xpack4 are trademarks of the IPG Automotive Group.

CarMaker, TruckMaker, MotorcycleMaker, MESA VERDE are
registered trademarks of IPG Automotive Group.

All other product names are trademarks of their respective companies.

IPG Automotive Group means any corporate body of which IPG Automotive GmbH has a majority stake.

Contents

1 About This Quick Start Guide 6

2 CarMaker Installation 7

3 Starting CarMaker 8

4 Running Example TestRuns 10

4.1 Loading a Predefined Virtual Vehicle Environment 11

4.2 Running the Simulation 12

4.2.1 Interactive View of the Current Simulation 13

4.3 Animation with IPGMovie 16

4.4 Data Analysis 18

4.4.1 IPGControl 18

4.4.2 Alternative Methods 24

5 Creating Your First TestRun 25

5.1 TestRun with Main Focus: Vehicle Dynamics 25

5.1.1 Building a Road Network Using the Scenario Editor 25

5.1.2 Defining the Maneuver 28

5.1.3 Selecting a Vehicle and Simulating 29

5.1.4 Saving Your TestRun 31

5.2 TestRun with Main Focus: ADAS 32

5.2.1 Building a Road Network Using the Scenario Editor 32

5.2.2 Defining a Maneuver 36

5.2.3 Adding Traffic 38

5.2.4 Collision Detection 40

5.2.5 Setting up an AEB System 41
5.2.6 Using NamedValues 43
5.2.7 Executing Multiple TestRun Variations using the TestManager . 45
5.3 TestRun with Main Focus: Powertrain 46
5.3.1 Creating the 48V Hybrid P1 46
5.3.2 Importing the Road 49
5.3.3 Maneuver and Driver Parameters 51
5.3.4 Implementing Speed Limits 53
5.3.5 Creating the 48V P2 Hybrid 56
5.3.6 Test Automation using the Test Manager 58
5.4 TestRun with Main Focus: Scenario Editor 62
5.4.1 Building a Road Network using the Scenario Editor 62
5.4.2 Defining the Maneuver 79

6 Vehicle Model Parameterization 87
6.1 What is a Data Set? 87
6.2 Creating a New Vehicle Data Set 87
6.2.1 Using the Vehicle Data Set Generator 88
6.2.2 Using the Import Feature 89
6.3 Vehicle Submodels 89
6.3.1 Vehicle Body 90
6.3.2 Bodies 91
6.3.3 Engine Mount 92
6.3.4 Suspensions 93
6.3.5 Steering System 100
6.3.6 Tires 101
6.3.7 Brake System 101
6.3.8 Powertrain 102
6.3.9 Aerodynamics 105
6.3.10 Sensors 106
6.3.11 Vehicle Control 107
6.3.12 Misc. 108
6.4 Save as a New Data Set 108
6.5 Creating a New Trailer Data Set 109
6.6 Creating a New Tire Data Set 110
6.6.1 Overview of Tire Models 110
6.6.2 Tire Model Exercises 111

7 CarMaker for Simulink 112
7.1 Starting CarMaker for Simulink 112
7.2 First simulation with CarMaker for Simulink 114
7.3 Description of the CarMaker Environment in Simulink 116
7.4 Advantages and Disadvantages of using CarMaker for Simulink 117

8 Features of the CarMaker for Simulink blockset 118
8.1 General Information 119
8.1.1 Block Properties 119
8.1.2 Modeling Principles 119
8.1.3 Purpose of the Sync_In and Sync_Out Ports 120
8.2 Utility Blocks 120
8.2.1 CarMaker GUI 120
8.3 CarMaker Dictionary Blocks 121
8.3.1 Read CM Dict 121
8.3.2 Write CM Dict 121
8.3.3 Define CM Dict 122
8.3.4 Dictionary Initialization 123

9 How to Extend the Simulink Model 124
9.1 Overview of the CarMaker Simulink Structure 124
9.2 Example 126
9.2.1 Contents 126
9.2.2 Create a new Simulink Model 126
9.2.3 Prepare the extension 126
9.2.4 Connect Inputs 127
9.2.5 Demonstration Examples 130

10 Additional Information About CarMaker 131
10.1 What are the Differences between each CarMaker Version? 131
10.2 Typical Tests 131
10.3 Feature List 133

Chapter 1

About This Quick Start Guide

This document is intended to give a first impression of CarMaker and how it can be used to perform simulations, either in form of a stand-alone application (for e.g. when being run "as is", without further software), or as an embedded environment in Simulink.

The Quick Start Guide can be seen as a tutorial, providing shallow insight into the powerful world of virtual test driving with CarMaker. That the software is much more complex than this document may lead to believe, can be seen in the many other documents, that can be found in the CarMaker Help section. Other CarMaker manuals and documentations describe the software in much greater detail, seeing as they contain all aspects of using, adapting and programming the CarMaker software package.

All paragraphs in this format contain instructions on how to proceed.

Chapter 2

CarMaker Installation

To install CarMaker, the user needs administrative privileges and extract the downloaded installation package (.zip archive). The IPG Installer can be started via double clicking on the executable *ipg-install.exe*.

In case the user has received an installation CD, this needs to be inserted into the CD-ROM drive of the computer. If AutoRun is activated, *IPGInstall* should start automatically. Otherwise, the executable file *ipg-install-exe* must be opened manually.

Once CarMaker has been installed, the user must contact the *IPG License Team* to apply for a valid license. To apply for a license, a license key request form can be found in the support area of the IPG website *http://www.ipg-automotive.com/*. The form needs to be filled out completely. Subsequently, the user receives a valid license file from the IPG License Team. The license file must be saved in the <*InstallationDirectory*>*etc* folder. Now the software is activated and ready to use.

Please find more detailed information about the IPG Installer or about e.g. using a floating license or installing a HIL system in the Installation Guide, available in the *Help* menu of the CarMaker main GUI.

Chapter 3

Starting CarMaker

Start CarMaker in Windows: ***Start > Programs > IPG > CarMaker Version No. > CarMaker.***

Starting CarMaker in Linux: ***Open console > enter "CM".***

Figure 3.1: Opening CarMaker using Windows

Once the final CarMaker executable has been selected, the CarMaker software opens.

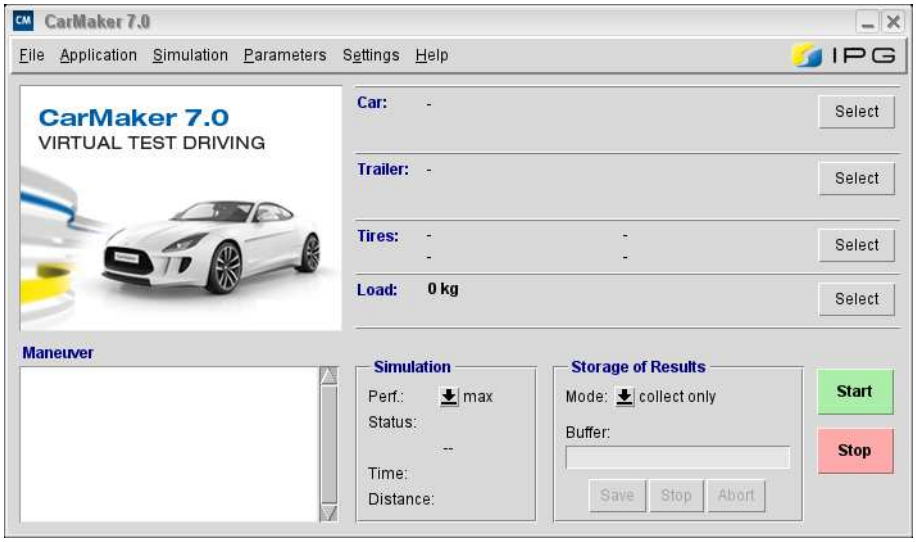


Figure 3.2: CarMaker main GUI

The first thing the user sees is the *CarMaker main GUI* (main Graphical User Interface), as displayed in the figure above. This is CarMaker's control center, with which all functionalities and sub windows are accessed.

Now, CarMaker is ready to work and can be used right away.

Chapter 4

Running Example TestRuns

Performing a simulation in CarMaker requires similar definitions to when carrying out a real test drive. The type of vehicle, tires, driver, test track and a maneuver, that the driver is to perform, need to be defined. CarMaker provides several predefined models for each of these requirements. A combination of these models and settings form what is called a *TestRun*.

What exactly is a TestRun?

CarMaker is based on fixed models (vehicle, suspension, tires, etc.) whose properties (e.g. values for the mass of each body or spring stiffness) can be varied.

This means that the number of bodies and the DOF (degrees of freedom) between them are already defined and the user doesn't need to model these on his own. If the user wishes to extend a model, CM4SL is recommended. For further information on this topic, please refer to section 'CarMaker for Simulink'.

The models themselves are already defined, but they still need to be parameterized according to their environment. For this, a so called *data set* is manually implemented or loaded for each of the models. Parameterizing includes selecting a vehicle, selecting or designing a road, defining a type of driver and defining a maneuver. After all of these components are set, CarMaker has the information necessary to control the *virtual vehicle environment (VVE)* and simulate.

All of these settings are stored in a file used by the VVE during simulation. Said file, which can be saved, loaded or edited, is what we call the *TestRun definition*. Loading and executing this file results in the simulation of that specific test.The TestRun files have the format of a regular text file and can be viewed with an editor.

In summary: a TestRun represents a test scenario in which all parameters of the virtual environment (vehicle, driver, tires, etc.) are sufficiently defined.

The standard CarMaker installation includes a variety of example TestRuns containing all the data required to describe each of these models. This makes it possible for a new user to work with CarMaker and get to know the other main features of the program right away.

4.1 Loading a Predefined Virtual Vehicle Environment

When starting CarMaker for the first time, a project folder that will contain all the TestRun data, needs to be created. This is done by selecting *File > Project Folder > Create Project* in the CarMaker main GUI. When you start CarMaker again in the future, you can then either keep using the same project folder or create a new one. Either way, a project directory must be defined.

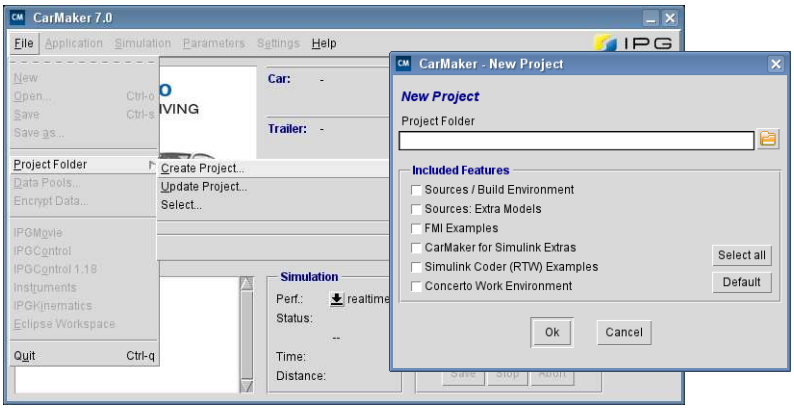


Figure 4.1: Creating a new project folder

In the field labelled *Project Folder* a path is selected where the project directory will be placed. The additional features that are available can be activated optionally be ticking the appending boxes. For this tutorial, none of the available features needs to be selected.

Now, after a project directory has been defined, a TestRun can be loaded.

Load the TestRun "SegmentBasedClosedTrack":
In the CarMaker GUI, click on *File > Open,* select the *Product Examples*, then *Examples > BasicFunctions > Driver > HandlingCourse.*

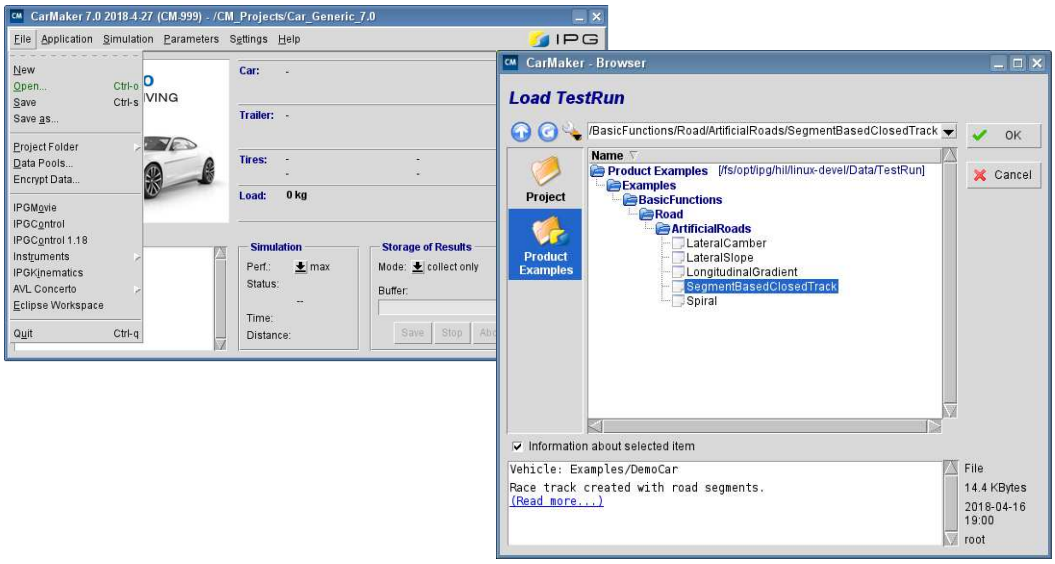


Figure 4.2: Loading "SegmentBasedClosedTrack"

Looking into the CarMaker main GUI, it is now filled with all the data sets that are required to execute a simulation. The TestRun is now ready for simulation.

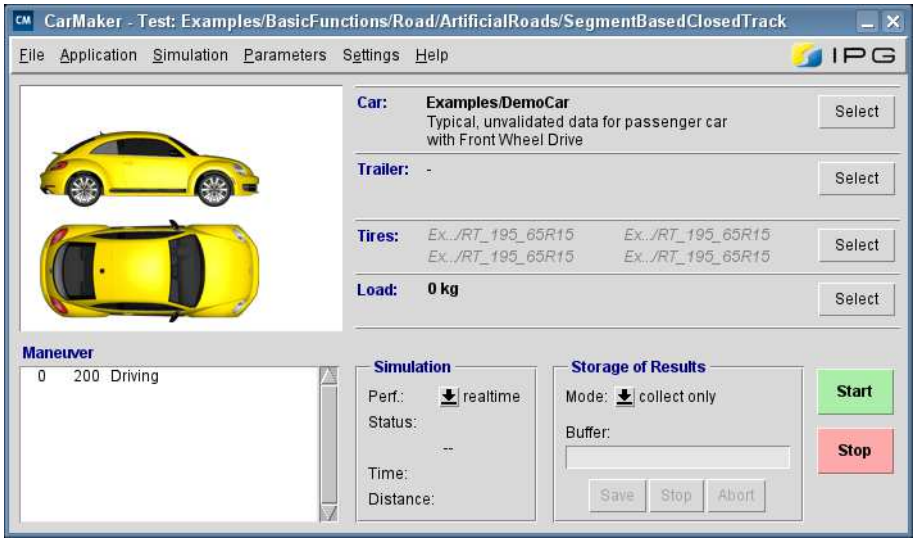


Figure 4.3: CarMaker GUI containing relevant data sets

4.2 Running the Simulation

Now, since there is a TestRun loaded in CarMaker, all the data necessary to successfully perform a simulation is already at hand.

Start the simulation: Click on the green “Start” button in the CarMaker GUI.

The simulation begins. This can be recognized by noticing the following points:

- In *IPGMovie*: The animation of the current simulation is running and can be viewed by the user. After the simulation has ended, it is possible to replay the simulation in various speeds.
- In the CarMaker GUI: In the box labelled *Simulation* a timer, an indicator for the distance covered by the car, as well as the current status of the simulation are displayed.
- In *Instruments*: The operating displays, especially the tachometer, can be viewed in real time.

Note: If IPGMovie and Instruments do not automatically pop up when CarMaker is started, these can be opened by clicking *File* > *IPGMovie* and *File* > *Instruments* in the main GUI.

Figure 4.4: Running simulation displayed in the main GUI, Instruments and IPGMovie

Stop the simulation: Click on the red “Stop“ button in the CarMaker GUI.

The IPGMovie animation ends, the Instruments slowly come to a stand still and all other displays in the CarMaker GUI are stopped.

The *Stop* button is used when it is desired to abort the TestRun before completion. Otherwise, when the TestRun comes to an end on it’s own, the simulation also stops and returns to idle state, the user does not need to manually end it.

The following chapters will describe what happens during the simulation in further detail.

Click the green “Start“ button in order to start the TestRun again.

Now you can observe the CarMaker GUI, IPGMovie, the Instruments window and IPGControl to find out what these display during an active simulation.

4.2.1 Interactive View of the Current Simulation

CarMaker offers various methods to observe a simulation. A few of these will be described briefly in this chapter. The most important ones will be picked up again in later chapters e.g. section 4.4.1 ’IPGControl’.

IPGMovie

IPGMovie enables the user to watch an animation of the current simulation.

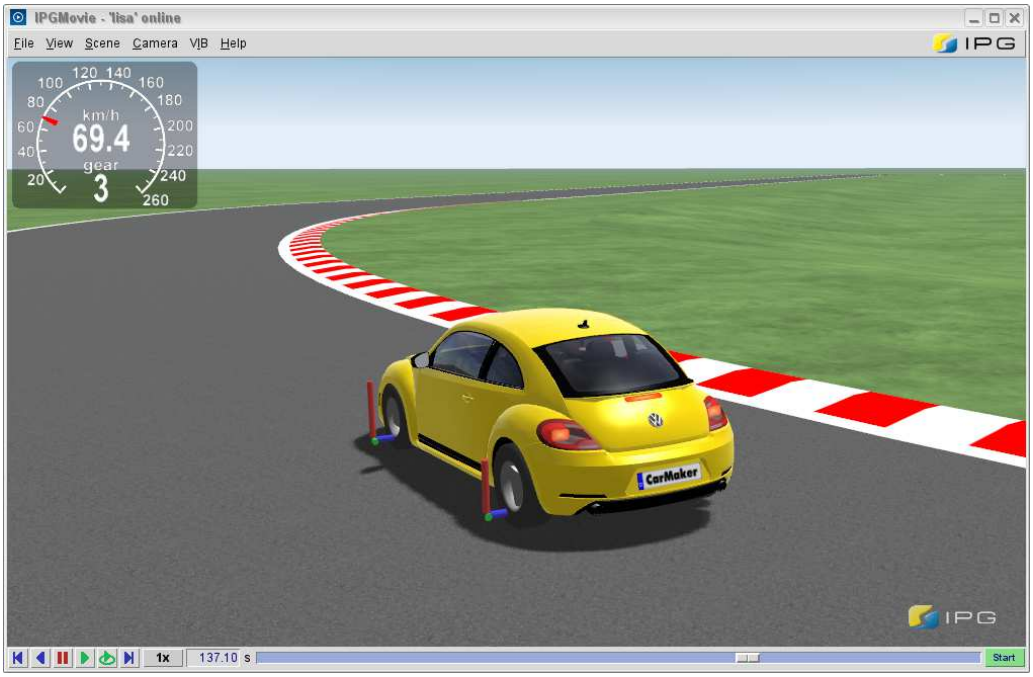


Figure 4.5: Online view of a simulation in IPGMovie

During the animation, different points of view and background scenery can be selected. Despite the view and design settings available, IPGMovie is much more than just an animation tool. For instance, current animations can be exported to a file for further use in presentations or advertising.

A more detailed explanation on using IPGMovie is given in section 4.3 'Animation with IPG-Movie'.

CarMaker GUI

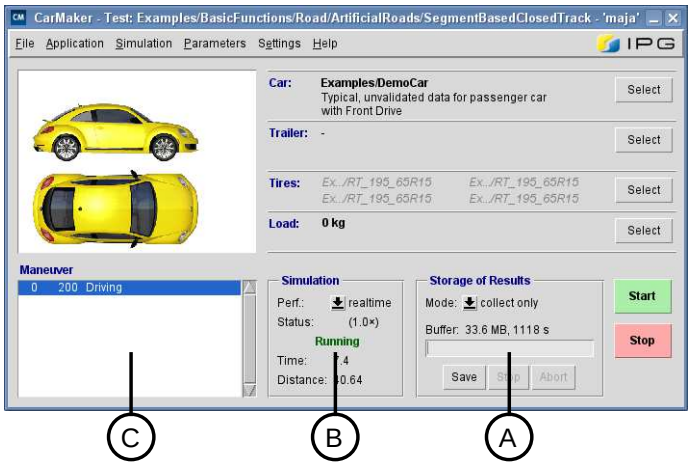


Figure 4.6: The CarMaker GUI

- Box (A): *Storage of Results*

The results generated by CarMaker are usually saved to a buffer in the computer RAM. They can, however, also be saved to a file for further analysis by selecting the *mode* option *save all* instead of *collect only*. This box helps control and define the saving strategy. For further information, please refer to the User's Guide which can be found under *Help* in the CarMaker main GUI.

- Box (B): *Simulation*

In this box the user can define the speed of the simulation: real time or slower/faster than real time. The *max* option enables simulation speeds that are as fast as the current PC hardware and active software will allow. This can be up to 40 times real time and remember that opening IPGMovie will have an effect on the PC performance, just like all other applications.

Speeding up the simulation saves time, especially when working with automated TestRuns where often a very large number is carried out in sequence. Another feature is that the simulation speed can be altered during the simulation.

Additionally, in this box the time and distance of the current simulation are displayed. The user also receives information regarding the status of the simulation: *Idle* when no simulation is running, *Preparing* during the start phase or *Running* while the simulation is being carried out.

Observe changes in IPGMovie when the Simulation Speed is *2x*, *Max* and *Realtime*.

- Box (C) *Maneuver*

As will be further described in section 'Creating Your First TestRun', several maneuver steps can be defined that are all displayed in this section. The current maneuver step is highlighted in blue so that it is clear which step is currently being simulated.

Instruments

Instruments is an additional display that is used to quickly, visually check the most important data and gives an overview of the driver's actions. The content of Instruments is very similar to what can be seen in the dashboard of a real vehicle!

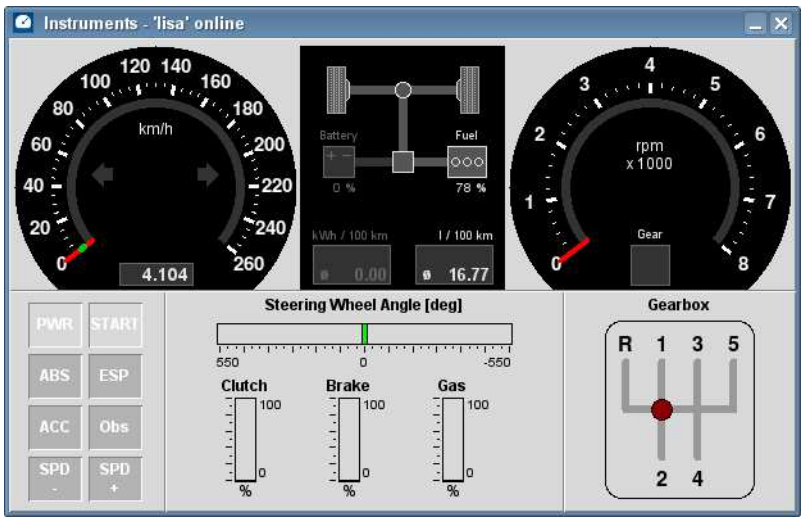


Figure 4.7: The Instruments GUI

What is displayed in Instruments can optionally be individualized and extended by the user. In order to do so, CarMaker features an interface using tcl/tk script language that enables each user to build an individual GUI. Further information regarding this topic can be found in the Programmer's Guide.

IPGControl

IPGControl is an embedded tool that is used to plot various diagrams of the simulation results online.

Open IPGControl: *File > IPGControl*

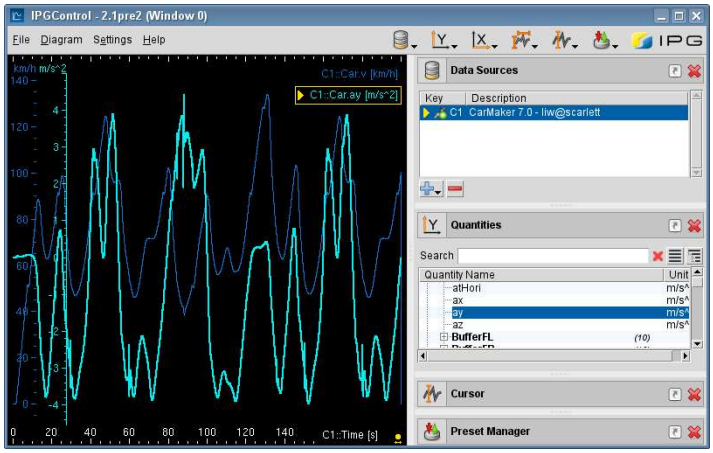


Figure 4.8: IPGControl Window

Plot the speed and lateral acceleration of the vehicle: *IPGControl Sidebar> "Quantities" field > left-click on the variables "Car.v" and "Car.ay".*

In the CarMaker GUI, start the simulation.

After a while, select *Fit totally* in order to fit the diagram within the available diagram area.

IPGControl usage is explained in further detail in section 4.4.1 'IPGControl'.

4.3 Animation with IPGMovie

Click the green *Start* button to simulate the TestRun *SegmentBasedClosedTrack* again.

IPGMovie offers the functionality of an *online animation*. This means that the current simulation data is provided without delay - the virtual world is pictured directly during the simulation. Additionally, by loading external result files, the animations of TestRuns that have previously been carried out can still be shown in IPGMovie. This is called *offline animation*.

The first IPGMovie feature to be explained here is that using the mouse, the user can change the point of view and zoom in and out during the simulation.

Change the point of view: Push the left mouse button and move the cursor in the desired direction.

Zoom: Keep the middle mouse button pressed and move the cursor up and down or use the mouse wheel.

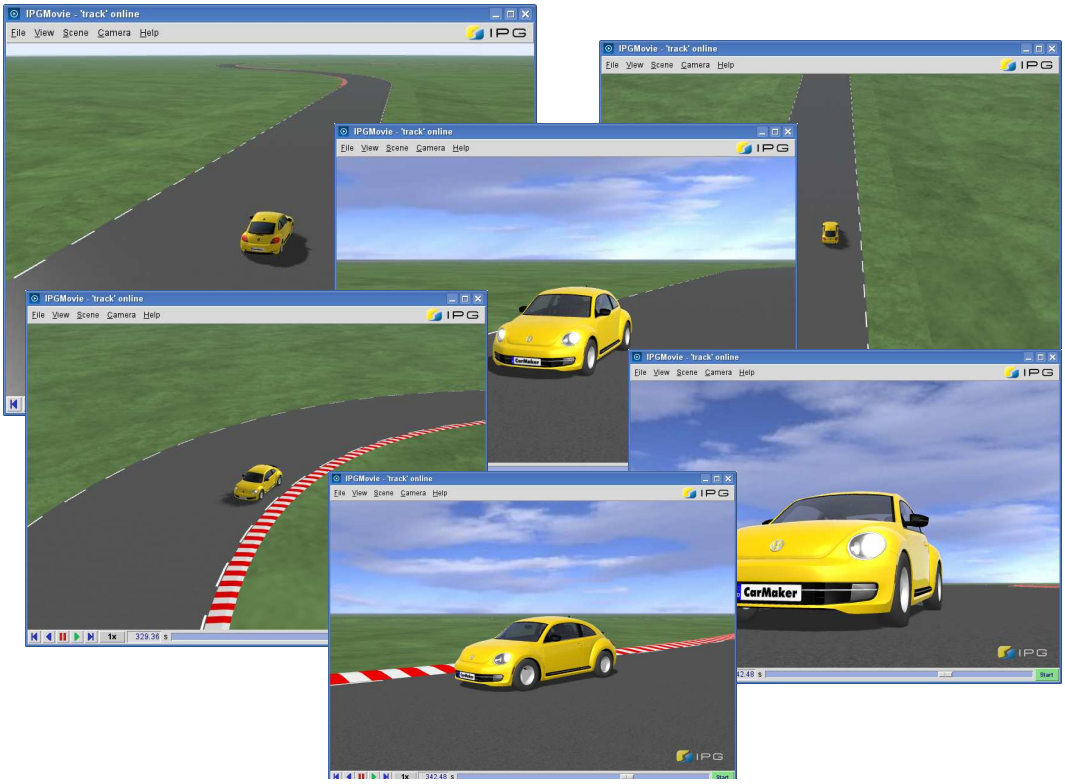


Figure 4.9: Different points of view in IPGMovie

Abort the Testrun: Click the red *Stop* button in the CarMaker main GUI.

After a simulation has ended (when the TestRun is over or after the user clicks the *Stop* button in the CarMaker main GUI), the animation can be replayed in IPGMovie using the cursor in the Control Bar. The camera's point of view can still be modified at all times.

Watch the movie from the last simulation in IPGMovie again by using the cursor in the Control Bar. Change the point of view.

Furthermore, IPGMovie offers a whole variety of useful functionalities:

- Comparing the animations of two different simulations: two simulations with different settings are carried out and can then be played simultaneously in IPGMovie, allowing quick, visual analysis of the results.
- A variety of options accessible via right-click within the IPGMovie window.
- Different Camera options that can be selected in the IPGMovie window by clicking *Camera* in the top menu. Additionally, multiple personalized camera views can be defined.
- Various options that can be selected via the *Scene* option in the top menu.
- Exporting videos or pictures from the animation: *File* > *Export* (DivX is one possibility that is only possible if the codec is installed on the current system).
- Viewing tire forces: *View* > *Show* > *Forces*. The colored bars at the contact points of each tire represent the forces in each of the three directions or the coordinate system.

All modifications that are made to the road definition will be displayed in IPGMovie. However, the animation needs to be updated first. That means the simulation needs to be started and stopped at least once. Further information regarding IPGMovie can be found in the Movie manual.

4.4 Data Analysis

There are several ways to analyze the data that is generated during a CarMaker simulation. IPG offers it's own tool called IPGControl that was developed for viewing and analyzing purposes and can also be used standalone. Apart from this, a wide variety of third party tools like Excel or Matlab can be used for data analysis.

4.4.1 IPGControl

This chapter will explain how to analyze and plot simulation results using IPGControl.

IPGControl offers the functionalities of an *online result management application*. This means that the current simulation data is provided without delay. Diagrams can be displayed directly during the simulation. By loading external result files, the user also has the possibility to display the results of previous TestRuns. This is called *offline result management*. Each new source of results is displayed in the *Data Sets* list in the selection window. The data saved in these files can be plotted in another diagram.

The figure below makes clear that there is a very high number of quantities (simulation variables) that can be plotted. IPGControl allows the user to manipulate the plotted quantities and change scaling and the axis so that the results can be analyzed more efficiently and effectively.

Opening IPGControl

IPGControl is opened via the *File* menu in the CarMaker main GUI.

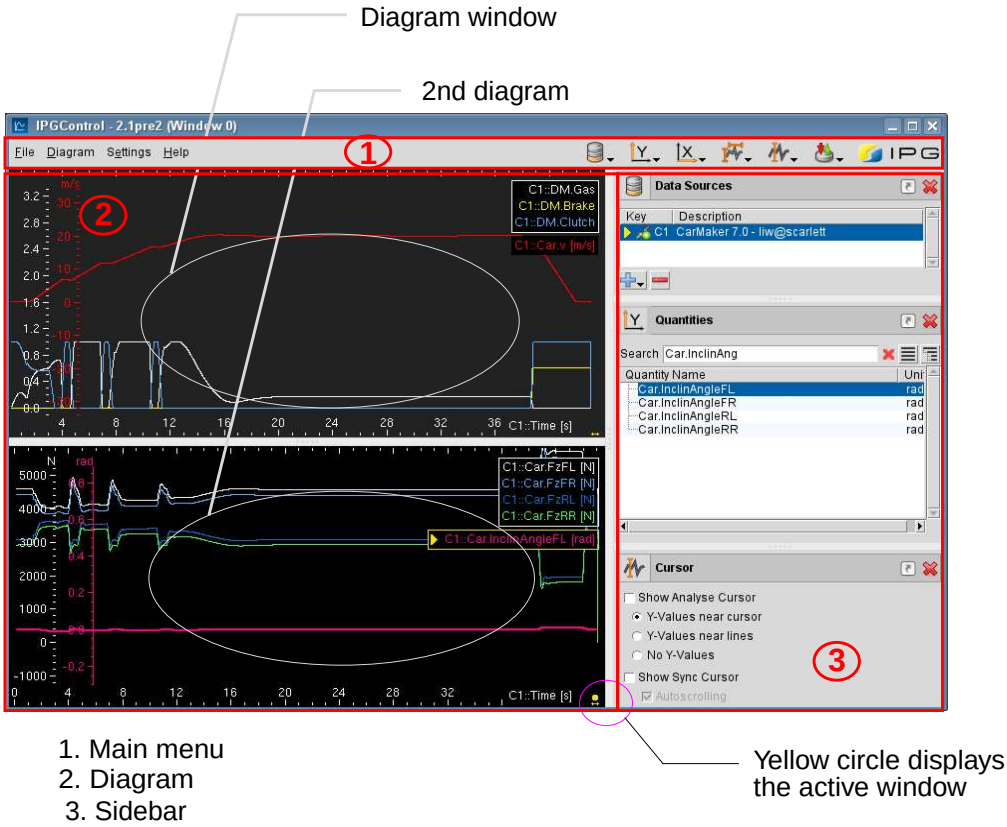


Figure 4.10: IPGControl Window

Using the *Data Sources* in the Sidebar, the result files can be selected that are to be displayed in the Diagram.

Selecting the Quantities to be plotted

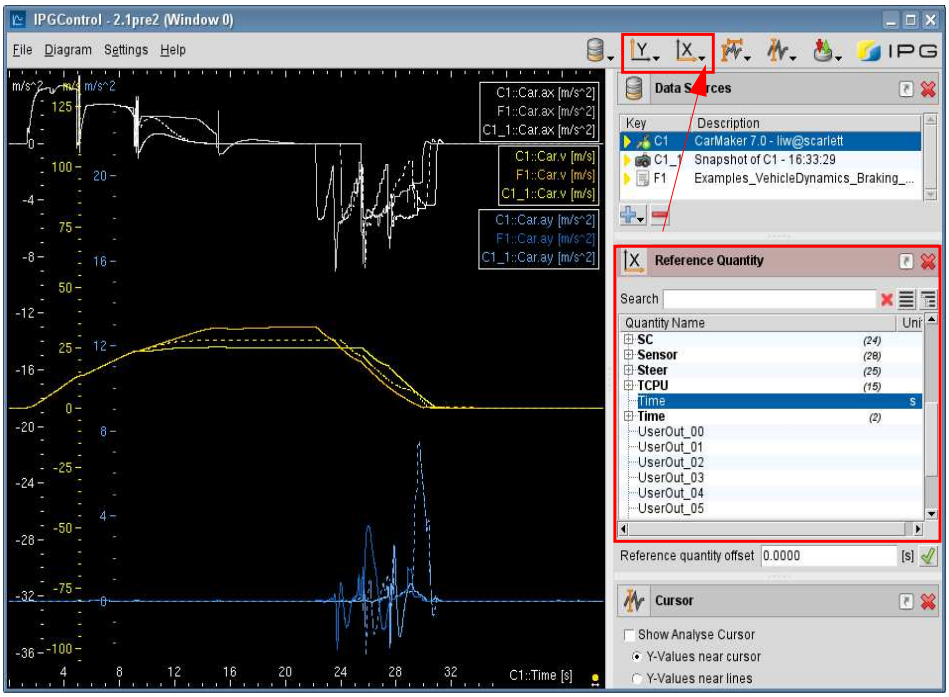


Figure 4.11: Choosing quantities to plot

First and foremost, if more than one diagram is present in the Data window, one has to be activated for editing. Only then quantities can be selected for plotting. Therefor, the desired diagram is left-clicked once. The small, yellow symbol informs the user which diagram is currently active.

The quantities and reference quantity can either be chosen from a quantity list in the sidebar or by using the Quantity browser. The Quantity browser can be opened by clicking and holding the X or Y buttons in the main menu. By clicking just once on the X/Y buttons in the top menu, the quantity list on the sidebar swaps between usage for the vertical or horizontal axis (see figure 4.11). The *Search* field applies a filter to the quantity list in the sidebar. The filter can be unapplied by clicking on the red cross next to the entry field.

Quantities can be removed from the diagram by reselecting them in the previously mentioned lists (Quantity and Ref. Quantity), by *undisplaying* them in the Quantity browser using the red cross sign or by double-clicking the quantity itself in the list on the right side of the diagram.

The quantities displayed in IPGControl can also be referred to as *User Accessible Quantities (UAQ)*. For an explanation of each UAQ available for plotting in IPGControl, please refer to the Reference Manual, section *User Accessible Quantities*.

Using the main menu

Using the drop-down menu under *Data Sources,* the user can choose the source from which the data should be displayed. This can be either the current simulation or a loaded result file from previous simulations.

Furthermore, the sample rate of the data can be defined via *Settings* > *Sampling rate*.

With the option *Diagram > Add window*, more Diagram windows can be added to the display.

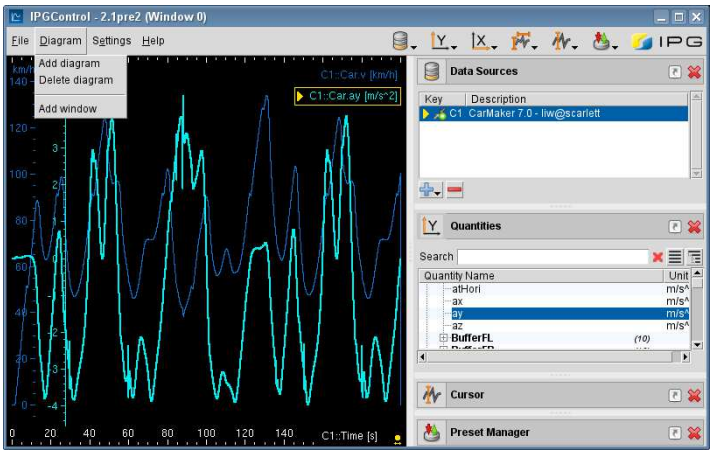


Figure 4.12: Adding additional Diagram windows

The separate windows can each have the same or independent view settings and data sources compared to the main window. The sidebar configuration, data sources and signal selection is managed entirely independent of the main view.

With the option *Diagram > Add/Delete Diagram* you can add or remove a diagram in the data window

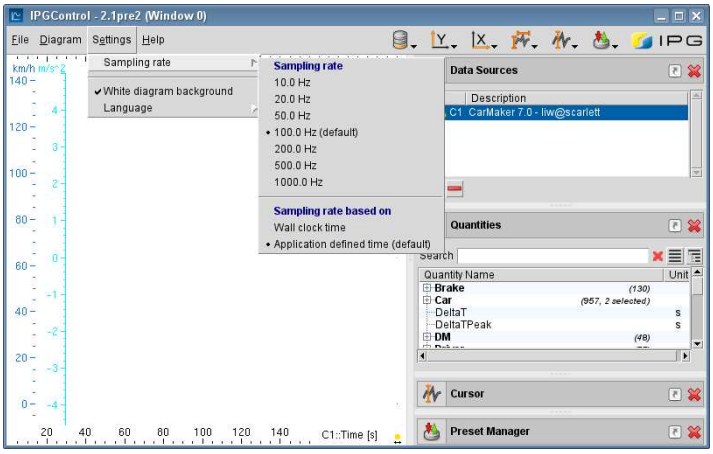


Figure 4.13: Changing the sampling rate on a diagram with white diagram background

The background color of the diagrams can be changed from black (default) to white via *Settings > White diagram background*. Please note that IPGControl has to be restarted for the background color change to become active.

Functionalities of the Diagram Window

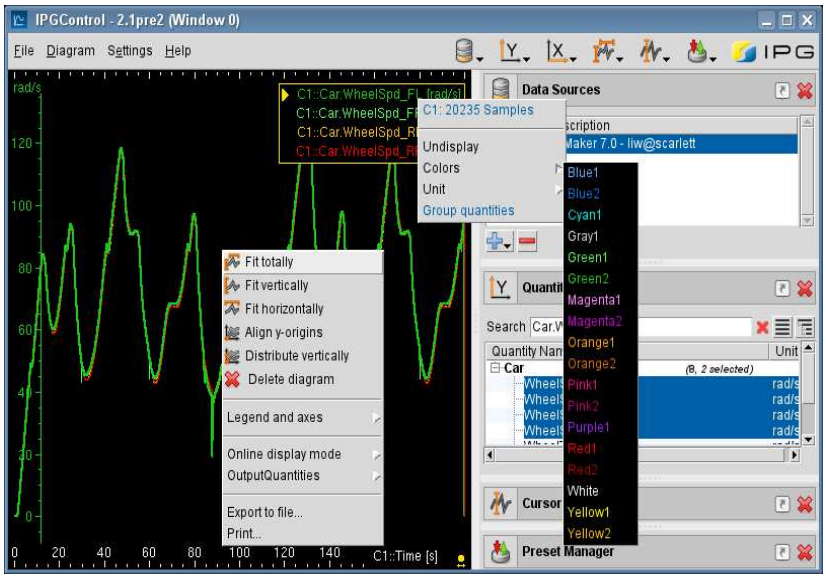


Figure 4.14: Diagram Window functionalities

Functionalities for adapting the Diagram:

- Right-clicking in the middle of the diagram > *Fit* options: Fits the selected curves (or all of them, if no specific curve is selected) to the available space in the diagram in the desired manner.
- Pressing Shift + x or only x on the keyboard: Changes the scale of the x-axis.
- Pressing Shift + y or only y: Changes the scale of all y-axes. To scale only one y-axis, the desired curve needs to be selected first.
- Pressing Ctrl + y or Ctrl + z: Zooms all y-axes
- Left-clicking in the diagram + move the mouse: Moves the diagram along all diagram axes
- Pressing Shift + Left-clicking in the diagram + move the mouse: Moves the diagram along either vertical or horizontal axes depending on mouse movement
- Pressing Shift + Mouse wheel: Zooms active y-axes (if no active axis: zoom all)
- Pressing Ctrl + Mouse wheel: Zooms x-axis
- Pressing Shift + Ctrl + Mouse wheel: Zoom all y-axes
- Pressing the up and down keys: Shifts all y-axes simultaneously. To move only one y-axis, the desired curve needs to be selected first.
- Pressing shift and the up or down keys: Shifts all y-axes at a smaller step size. To move only one y-axis, the desired curve needs to be selected first.
- Pressing PageUp and PageDown: Shifts the x-axis.
- Pressing Home or End: Changes the limits of the x-axis so that the entire range of data is displayed.
- Clicking left on a diagram: Activates the diagram.
- Clicking left on a quantity name: Selects the quantity.
- Clicking left on a y-axis: Selects the y-axis and all quantities attributed to it.
- Pressing Ctrl and left-clicking on a y-axis or quantity name: Allows you to select multiple quantities at the same time.
- Pressing Ctrl + a: Selects all graphs
- Pressing Ctrl + o/l:+/- Offsets the reference quantity

- Pressing Ctrl + r: Reconnects/Reloads/Updates the selected data source (depending on it's type)
- Clicking right: Opens a diagram/quantity related context menu.
- Clicking left on a quantity name, holding the mouse button, dragging the quantity across the diagram and dropping it on another quantity of the same unit kind (mouse cursor: |<-): Displays the quantity on the same y-axis as the quantity targeted on a y-axis of the same unit kind.
- Clicking left on a quantity name that uses the same axis as another one, holding the mouse button, dragging the quantity across the diagram and dropping it on a free area in the diagram (mouse cursor: _|_): Displays the quantity on its own (new) y-axis.
- Double-left-clicking on a y-axis: Opens an axis parameters dialog window.
- Double-left-clicking on a quantity name: Undisplays the selected quantity.
- Right-clicking on a selected quantity > *Colors* / *Unit*: changes the color or the unit of the selected quantity.

Exercise:

1.) Open the CarMaker GUI and load the TestRun *SteadyStateCircular42m*:
CarMaker GUI > File > Open > Product Examples > Examples > VehicleDynamics > Handling > SteadyStateCircular42m

2.)* Perform the first simulation*:
CarMaker GUI > Start

***3.)* Open IPGControl. Add a diagram:**
Main menu > Diagram > Add diagram

***4.)* Select the upper diagram:**
Click once in the middle of the diagram

***5.)* In the selected diagram, select *time* for the x-axis and *vehicle speed* and *wheel speeds* for the y-axis:**
- Main menu > X Ref. quantity browser > Time
- Main menu > Y quantity browser > Car[S..Y] > Car.v
- Sidebar > Y Quantities> Search > write "WheelSpd" > pressing "Enter"

***6.)* Select the lower diagram:**
Click once in the middle of the diagram

***7.)* Select lateral acceleration for the x-axis and steering angle for the y-axis:**
- Sidebar > X Ref. Quantity > Car[A..G] > Car.ay
- Sidebar > Y Quantity > Steer > Steer.WhlAng

***8.)* Start the simulation again:**
CarMaker GUI > Start

Now the curves are being plotted online, which means simultaneously to the simulation. After the TestRun is complete, the diagrams can be scaled individually and the curves can be moved around within the window. Try using two different axes for the wheel speeds.

4.4.2 Alternative Methods

In addition to IPGControl, CarMaker features an interface to other result management tools: e.g. Excel and Matlab.

Microsoft Excel

Result files can be saved directly as ASCII files. The data, separated by a tabulator, can be easily imported to an Excel sheet.

Another possibility is to write the data plotted in a diagram directly to an Excel file using the right mouse button in the diagram window and clicking *Export to File*. Excel file formats, as well as ASCIIs are available.

MathWorks MATLAB

After running a MATLAB script for CarMaker, a special command in Matlab (*cmread*) can be used to load the content of the result file into a matrix.

Thus, the content of a result file can be directly read and plotted in Matlab.

Other formats

Besides the above mentioned output file formats ERG and ASCII, CarMaker can store the simulation results directly in MDF format. This popular data exchange format is accepted by many postprocessing tools.

Further information on post-processing CarMaker simulation results can be found in the User's Guide.

Chapter 5

Creating Your First TestRun

This chapter will describe how to build an individual TestRun and have the possibility to choose a preferred example, depending on the field of interest.

5.1 TestRun with Main Focus: Vehicle Dynamics

In this chapter an example of how to build up a mu-split braking test will be described.

5.1.1 Building a Road Network Using the Scenario Editor

In this chapter the user will learn how to build a road with a friction stripe.

Open the Scenario Editor: In top bar of the CarMaker main GUI, click on *Parameters > Scenario / Road*.

Using the mouse, navigate to the *Road* section of the tab on the left side of the Scenario Editor window. Click and hold the first icon called *Road segment* until a drop-down menu opens. Select the Road segment type *Straight*.

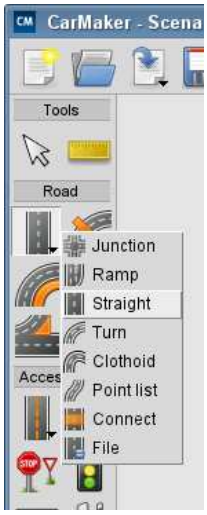


Figure 5.1: Choosing the Straight segment option

Use the cursor to navigate into the design area of the Editor (the middle window) and click once to define the starting point of the straight segment. The first point that is set is also the origin of the global coordinate system. Hold shift and click again to create a horizontal, straight road segment that is 1500m long.

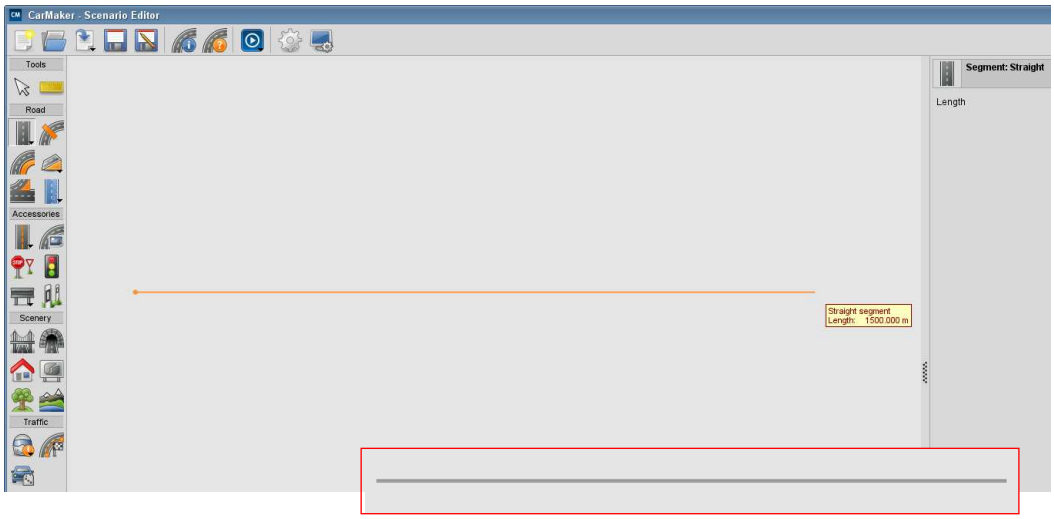


Figure 5.2: Implementing the Straight segment

Make the right lane wider: In the *Road* tab, select the *Lane* option and click the right lane of the road that you just implemented. In the tab on the right side, change both parameters *Width at start* and *Width at end* to *30m*.

Delete the left lane by clicking on it while the Lane mode is still active and then clicking the red "X" in the top right corner of the tab on the right-hand side.

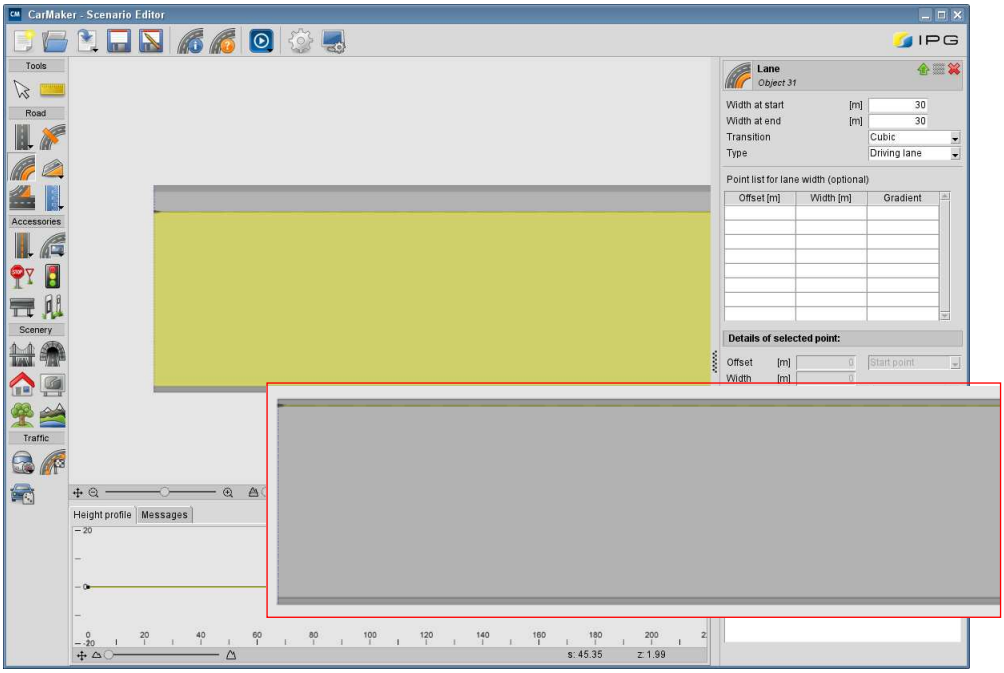


Figure 5.3: Changing the lane width and deleting lanes

Add a friction stripe: In the *Road tab*, left-click and hold the *Bumps* icon. In the drop-down menu, select the option *Friction*.

Click once on the Link (Straight road segment) to select it and then place two points anywhere on the Link to define beginning and end of the friction stripe. Alter these points in the tab on the right hand side so that the stripe starts *800m from Link start* and ends *1300m from Link start*.
Set *Lat. offset* to *-15m from Center* of the road and make the stripe *10m wide*. The *friction coefficient* should be *0.5*.

See the settings and what the stripe looks like in the figure below. Edit the Road road marking, if you wish.

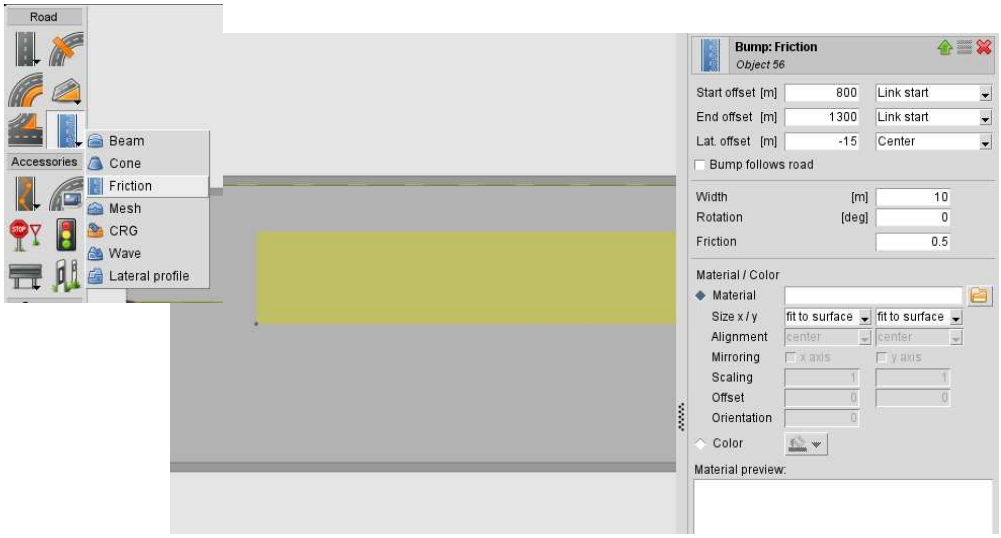


Figure 5.4: Adding a friction stripe

Now, in order for the vehicle to be able to drive on the road, a route is necessary.

Since this is a single lane on the right side of the road, the vehicle will drive along the center, as long as the *Driving side* is *Right-hand traffic*. You can check the active driving side by clicking on the gear icon in the top bar of the Scenario Editor and having a look at the specification. *Left-hand traffic* will work just fine, too. Just remember that the driving side can only be changed prior to the road being built.

Add a Route to the road: In the toolbar on the left side of the Editor, in the *Traffic* tab, select the *Route* icon. Using the cursor, click once on the predefined Path that's marked in green to select it and click again to confirm that you want to start generating the Route on this Path. It will become highlighted in orange. Since we don't want to select further Paths, double-click anywhere outside of the road to end the Route definition.
Hint: To simplify the selection of the predefined Path, it is useful to zoom up the road.

Now the road is complete. If you need to, open the solution in the product examples to be able to move on. The TestRun including the solution *Step1_Scenario* can be found in *Product Examples > Examples > VehicleDynamics > _Quickstartguide.*

5.1.2 Defining the Maneuver

For this type of driving experiment, the vehicle first needs to accelerate to a specific speed and then brake as soon as it reaches the friction stripe. Then, while braking, the two left wheels will be on the more slippery side of the road and the two right wheels will be on the side of the road with more grip.

This means that two maneuver steps are required which both also need an end condition.

Define the first maneuver step *Accelerating*: In the CarMaker main GUI, open *Parameters > Maneuver* and highlight the last line in the list (*END*) by clicking on it. Now select *New* to insert a maneuver step.
Write "Accelerating" in the *Description*, change the *Duration* to *999s* (to make sure the maneuver runs long enough) and give it a proper *End condition*:

Car.Road.sRoad >= 800

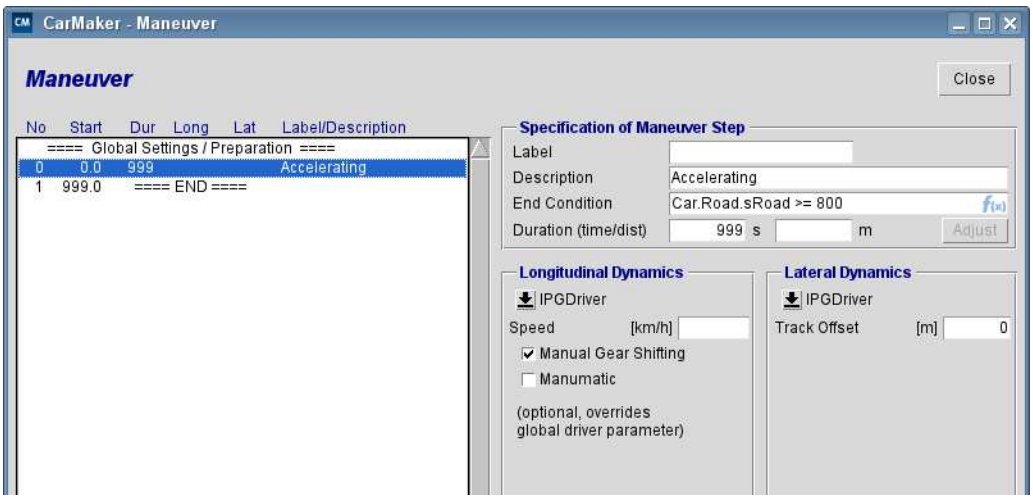


Figure 5.5: Defining the accelerating maneuver step

The end condition is in what is called *Realtime Expressions*. The end condition is dependent of the value of a specific quantity. In this case the quantity is *Car.Road.sRoad* which represents the vehicle's road coordinate. That means the s-coordinate of the vehicle's current position.

Since the friction stripe on the previously built road begins at 800m from Link start, the driver should stop accelerating at this point and start braking. That's why the first maneuver step ends here.

Define the second maneuver step *Braking*: highlight the last line in the list (*END*) again by clicking on it. Now select *New* to insert a second maneuver step below the first one.

Note that new maneuver steps are always inserted above the highlighted row in the list of maneuver steps.

Call this maneuver step "Braking", change the *Duration* to *999s* and the *End condition* to:

Car.v <= 0.001

Furthermore: Select the *Longitudinal Dynamics* option *Manual* and set the values of *Brake* and *Clutch* to *1*.

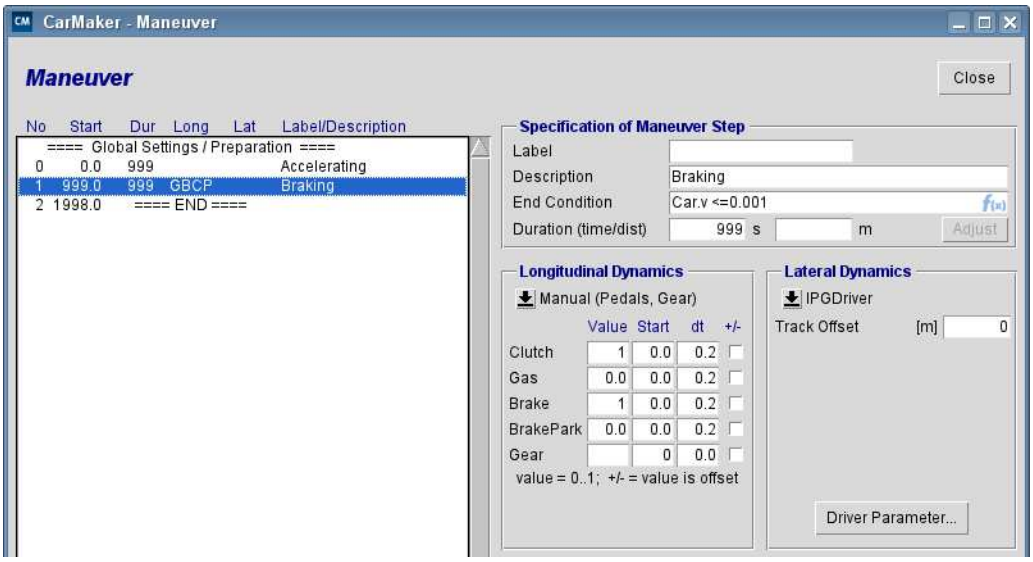


Figure 5.6: Defining the braking maneuver step

The end condition that is implemented here means that when the vehicle speed becomes lower than 0.001 m/s, the maneuver step is over and since this is the last step, the simulation will accordingly come to an end.

For the solution to the exercises up until now, open the TestRun *Step2_Maneuver* that can be found via *Product Examples* > *Examples* > *VehicleDynamics* > *_Quickstartguide.*

5.1.3 Selecting a Vehicle and Simulating

The only relevant component that is still missing for the Testrun to be sufficiently parameterized is a vehicle. CarMaker offers a wide variety of example vehicles that can be selected in the product examples.

Select a vehicle: In the Carmaker main GUI, left-click on *Select* in the field labelled *Car*.

In the browser that opens, navigate to the Product Examples and choose the *DemoCar*.

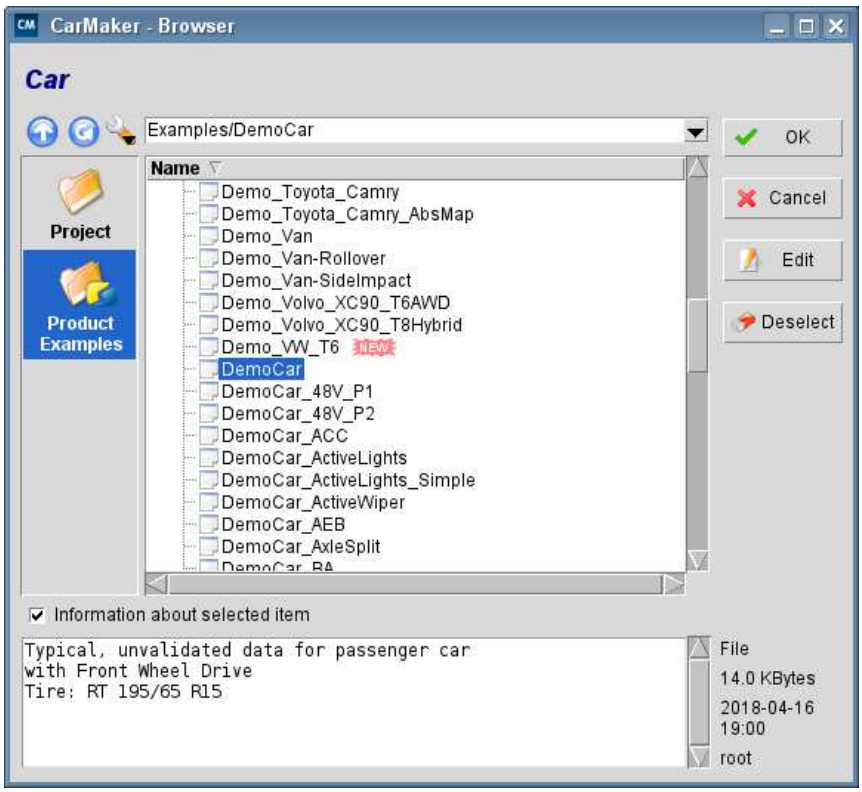


Figure 5.7: Selecting the CarMaker DemoCar

Now that the road is defined, a maneuver has been implemented that instructs the driver to accelerate until he reaches the friction stripe and then brake and a vehicle has been selected, the TestRun is complete and the simulation can be started.

Start the simulation: Click on the green *Start* button in the CarMaker GUI.

To watch an animation of the simulation, open *IPGMovie* via the *CarMaker main GUI > File > IPGMovie.*

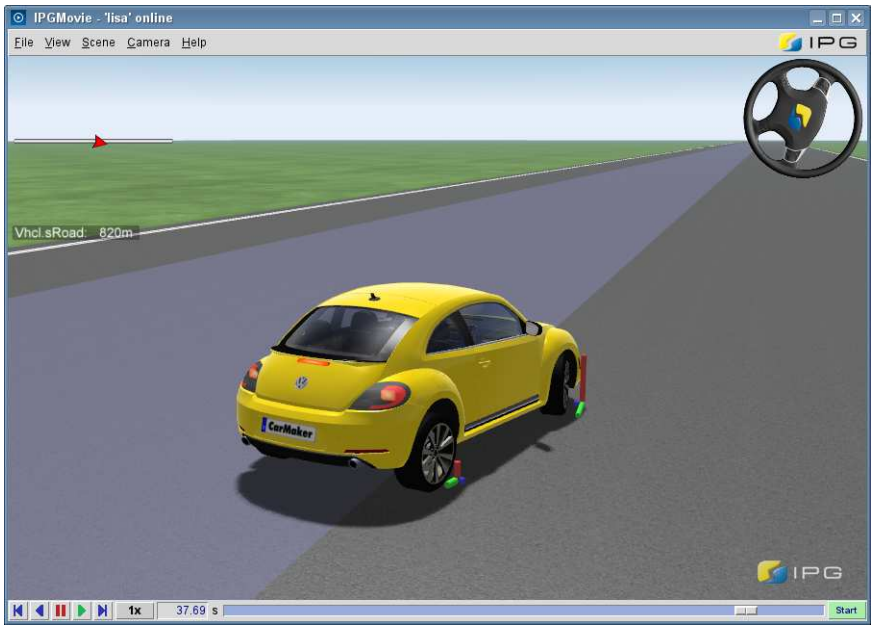


Figure 5.8: Animation of the mu-split braking test in IPGMovie

For the solution to the exercises up until now, open the TestRun *Step3_Vehicle* that can be found via *Product Examples* > *Examples* > *VehicleDynamics* > *_Quickstartguide.* Alternatively, you can also have a look at the TestRun *Final_VehicleDynamics* which is basically the same as *Step3_Vehicle*, but contains a few 3D objects for beautification of the scenario.

5.1.4 Saving Your TestRun

Save the TestRun, so that you can retrieve your work: In the CarMaker GUI, click on *File* > *Save as* and choose Project on the left bar in the dialog. With right-click, create a *new folder* "My_TestRuns". Double-click on the new folder and enter any name for the TestRun, e.g. *My_TestRun_VehicleDynamics* > *OK*.

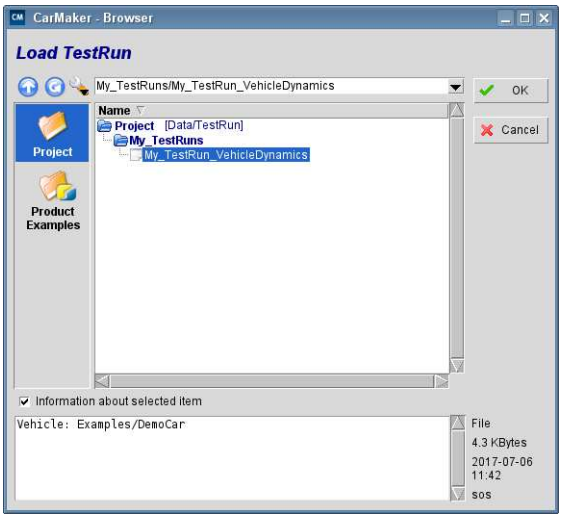


Figure 5.9: Saving the TestRun

5.2 TestRun with Main Focus: ADAS

The following example will demonstrate how to implement an autonomous emergency braking (AEB) TestRun. This example may be more interesting for users with an interest in ADAS applications.

5.2.1 Building a Road Network Using the Scenario Editor

In this chapter you will build a T-junction using the Scenario Editor.

Open the Scenario Editor via the CarMaker main GUI: Left-click *Parameters > Scenario / Road* or use the shortcut *crtl + r.*

Add road segments: In the *Road* tab on the left hand side, left-click once on the first icon *Road segment* in the *Road* tab and then right-click in the drawing area of the Scenario Editor to open a small drop-down menu. Left-click once on *Road segment* to open up another drop-down menu with all the options for road segments.

Left-click to choose *Junction.*

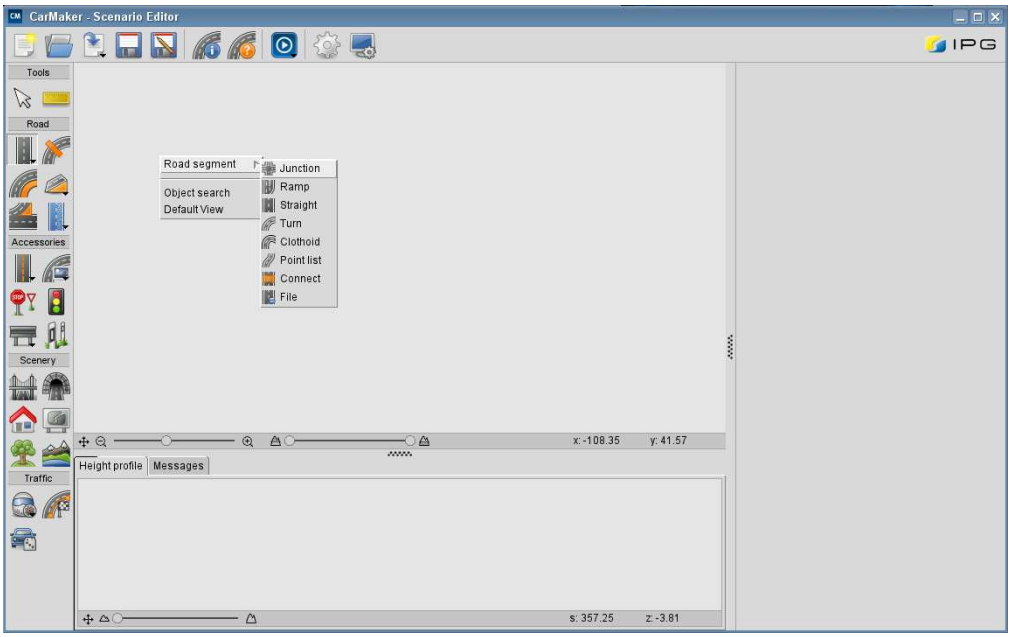


Figure 5.10: Selecting the road type Junction

Implement the Junction: Using the cursor, left-click once in the drawing area to set the middle of the junction.
Note that the first point you define in the Scenario Editor also initializes the *x,y,z - coordinate system* and is set with the values (0,0,0).

Move the cursor to the left and holding *shift* (shift makes the cursor lock on to specific values) click again to create the first junction arm at an angle of 180° with a length of 10m.

Now move the cursor to the right and, holding shift, click to implement the second junction arm with a length of 10m, at an angle of 0°.

Finally, move the cursor to the bottom and hold shift. When the cursor is at the right spot to define the last junction arm with a length of 10m, at an angle of 270°, *double-click* to finish building a junction with the last arm.

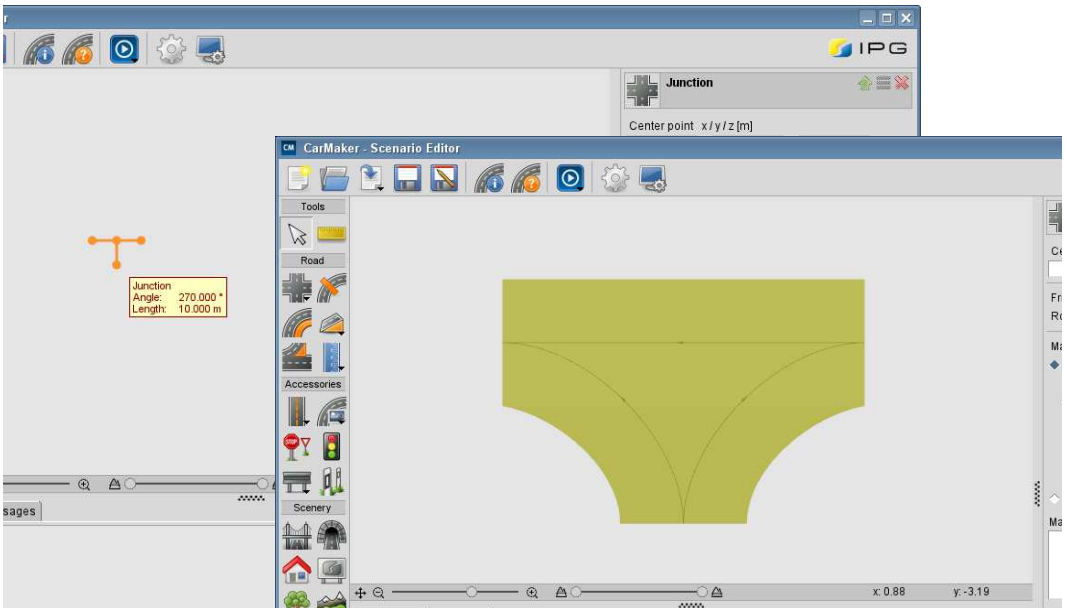


Figure 5.11: Implementing the Junction

Add a road to the first Junction arm: Click and hold the *Road segment* icon in the *Road* tab on the left hand side. In the drop-down menu, select the Road segment type *Straight*.

Move the cursor to the end of the first junction arm (Junction arm 0) and hover over the middle until an orange circle appears and the cursor locks onto the reference line. Left-click once to start drawing the straight segment here, move the cursor to the left and click again to implement the straight. In the tab on the right hand side you can edit the *Length* of this straight segment.

Set the length of this straight segment to 800m.

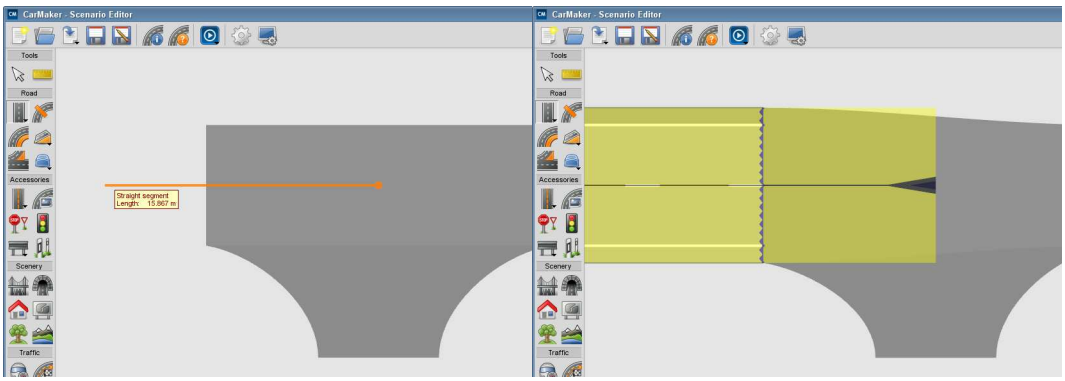


Figure 5.12: Adding a Straight to Junction arm 0

Analogously, add the other two Junction arms: Both on the right side and to the bottom, add a straight segment with a length of 400m.

The road network should now look like in the figure below.

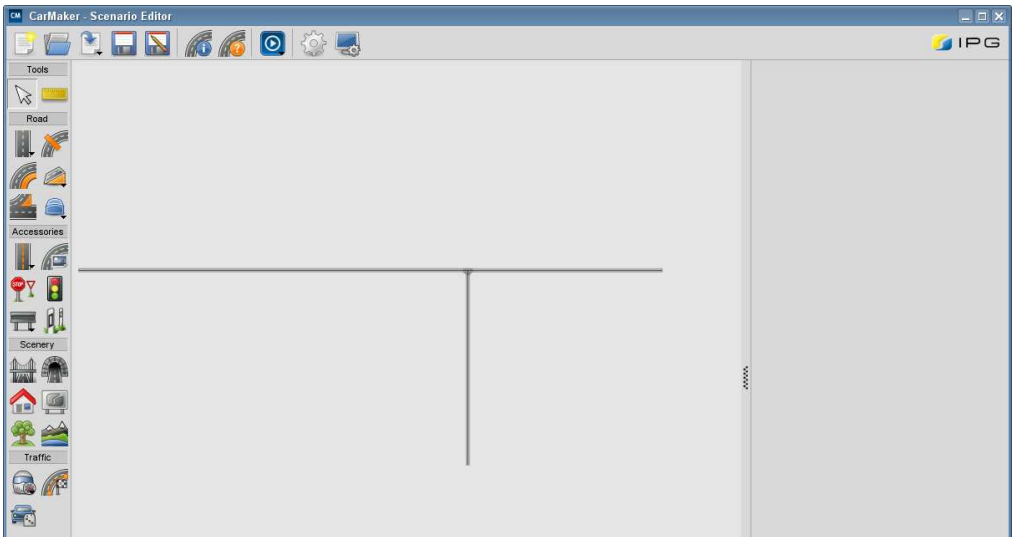


Figure 5.13: Finished T-junction

In order for the vehicle to know in which order he is to drive on the links, a route must be defined.

Click on the *Route* icon on the bottom left hand side of the Scenario editor.

Route_0: Click on the predefined path on the right lane of Link 1 (as labelled in the figure below) to select this path. As soon as a broken, orange line appears, click again to confirm that this is the correct path on which to begin the route definition. Now, click on the path that goes directly across the junction and then the one going straight forward on Link 2 to extend the Route by these two paths. To finish implementing the Route, double click somewhere on the Link.

Route_1: First, select the path on the right lane of Link 3 and confirm as previously described, then select the path that takes a left turn over the junction to extend the Route. Lastly add the path segment on the left lane of Link 1. Double click to finish. Now the Route comes from below and makes a left turn at the junction.

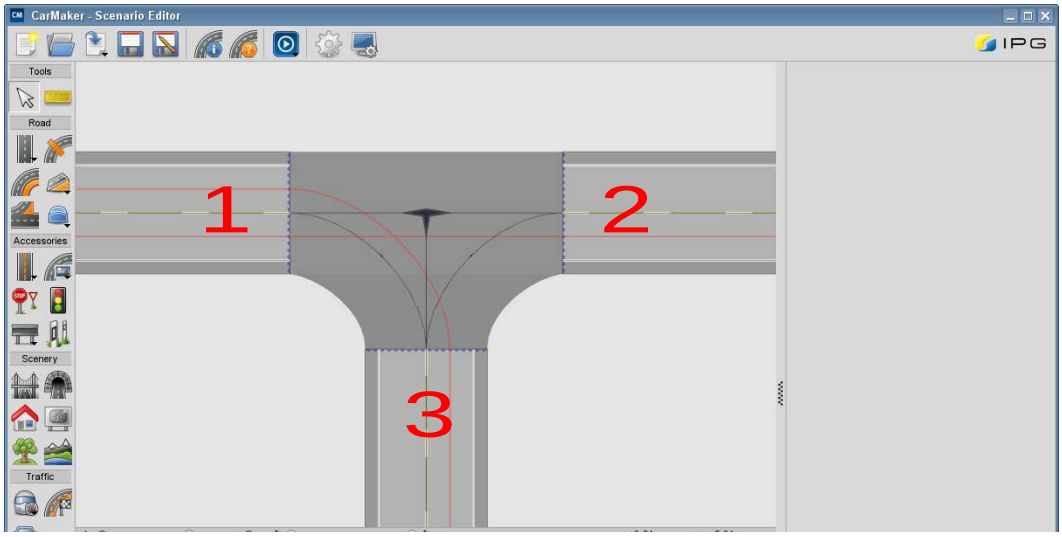


Figure 5.14: Finished Route

Within the Scenario Editor the user has the possibility to generate *Terrain* for his road network. The terrain's height profile is generated randomly by CarMaker and cannot be individualized by the user, only a minimum and maximum height value can be defined. Furthermore, the user can select a geometry file (.obj file) for the terrain's surface that is chosen after generation in the tab on the right hand side. If no file is selected in this field, the CarMaker default image is used. Terrain is added using the *Terrain Generator*.

Generate terrain: Click on the icon labelled *Terrain generation* in the *Scenery* tab on the left hand side of the Scenario Editor and click *Generate* in the tab on the right side.

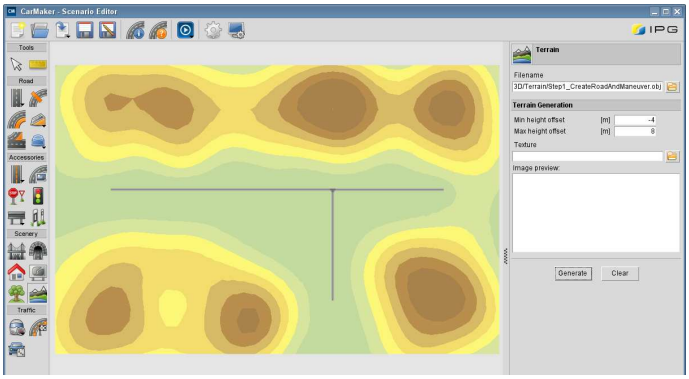


Figure 5.15: Terrain Generator of the Scenario Editor

5.2.2 Defining a Maneuver

After configuration of the road, a test car can be selected.

Select a vehicle: In the Carmaker main GUI, left-click on *Select* in the field labelled *Car*.

In the browser that opens, navigate to the *Product Examples* and choose the *DemoCar.*

Now a maneuver needs to be defined to tell the driver how to handle the vehicle.

Open the *Maneuver* Dialog via the *CarMaker main GUI* > *Parameters* > *Maneuver* or by using the shortcut *crtl* + *m.*

Add a new maneuver step to the list of maneuvers (that is still empty at the moment) and change the *Duration* to 60s. The duration of the maneuver step can be altered on the right side by editing the entry in the field labelled *Duration.*

In the field labelled *Longitudinal dynamics*, make sure that *IPGDriver* is the active option and change the *Speed* to 70km/h.

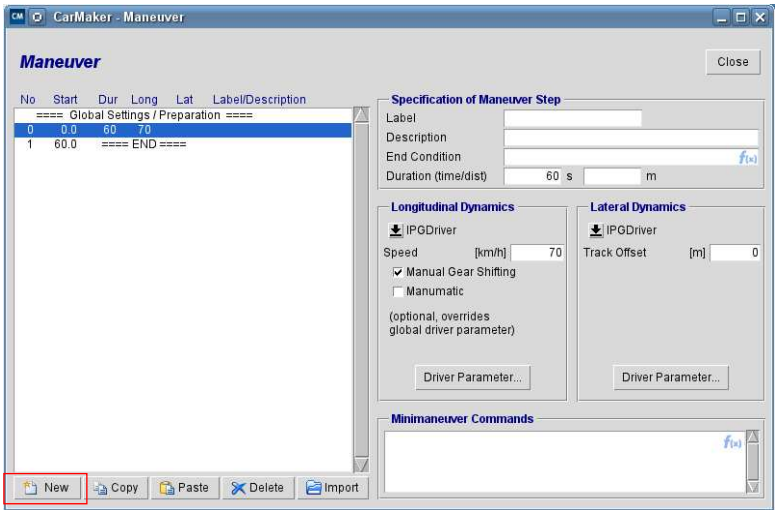


Figure 5.16: New maneuver step

Additionally, by clicking the item at the very top of the maneuver list *Global Settings / Preparation* the user can define various start values and additional commands.

Click on *Global Settings / Preparation*, set the *Start value Velocity* to 70 km/h and the *gear* that is to be active at the beginning of the simulation to the 5th gear.

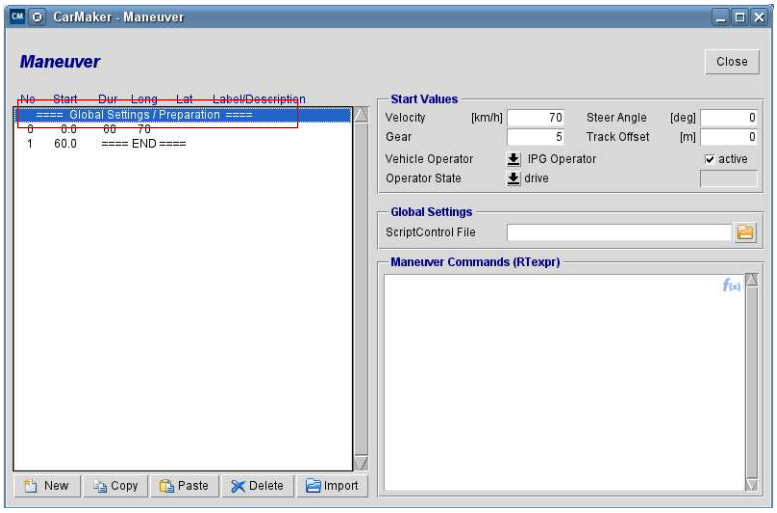


Figure 5.17: Maneuver Global Settings / Preparation

Now that the road, a vehicle and maneuver are defined, the TestRun can be started.

Start the simulation: Select *Route_0* as *Active Route* (Route on which the vehicle will drive) by clicking on *Scenario Settings* in top menu, then *Vehicle start position* > *Active route*. In the CarMaker main GUI, press the green *Start* button.

Open IPGMovie to observe the simulation.

5.2.3 Adding Traffic

Now a traffic object will be included in the simulation to show how the object detection in CarMaker works. This is necessary for the AEB system to become active. There is a very wide variety of available traffic objects and possible functions and maneuvers which are described in further detail in the User's Guide in the *Traffic* chapter. The following example will demonstrate only one possibility to integrate a traffic object.

Open the *Traffic* dialog: *CarMaker main GUI* > *Parameters* > *Traffic* and click on *New* to insert a new traffic object. Per default, this object is called *T00.*

Set the *Start position (s and t coordinates)* to 285m and 0m on Route 1 (the Route that takes the left turn).

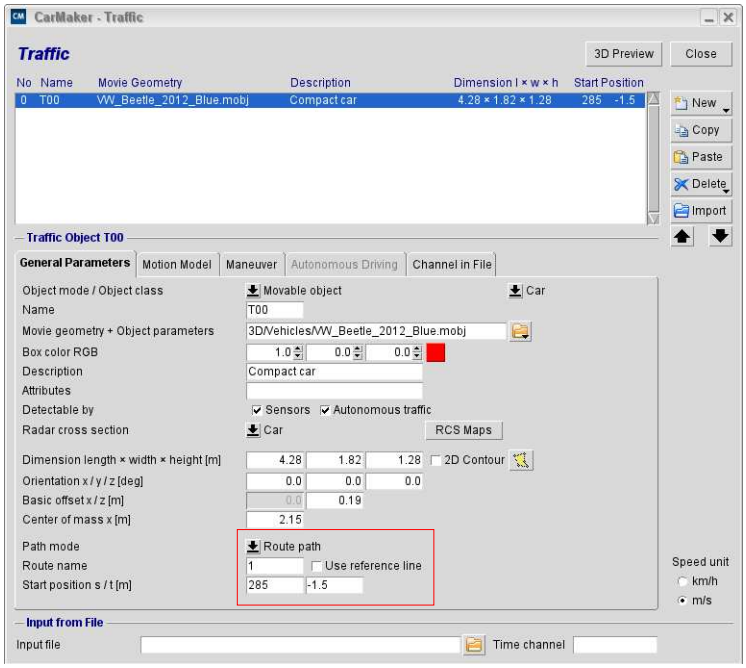


Figure 5.18: Traffic object configuration (1)

Add two maneuvers to the traffic object:

1. *Duration -> Time* 300 s and *long. Motion -> Velocity* 0 km/h, *End Condition* Car.Road.s2nextJunc < 55 (See maneuver step 1 definition in the figure on the top).

2. *Duration -> Time* 5 s and *long. Motion -> Acceleration* 3 m/s^2 (See maneuver step 2 definition in the figure on the bottom).

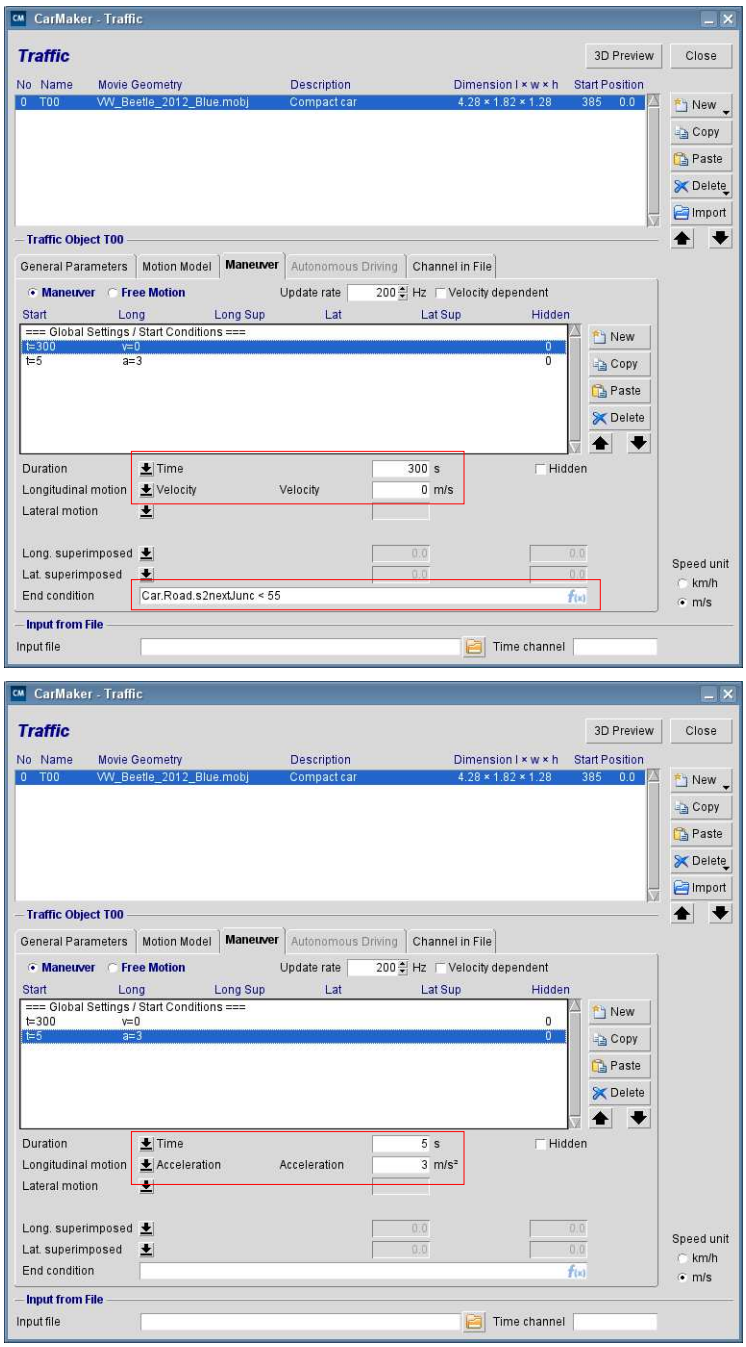


Figure 5.19: Traffic object configuration (2)

The driver will consider the traffic object by default. For the AEB test, IPGDriver should not consider any traffic objects.

Turn off the traffic consideration: deselect the *Consider Traffic* checkbox in the *MainGUI > Parameters > Driver > Traffic interface*

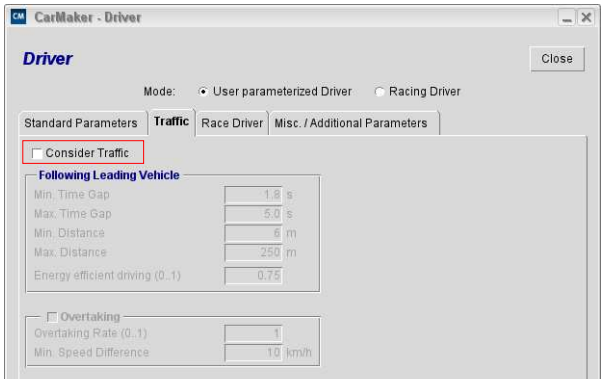


Figure 5.20: Driver parameter *Consider Traffic*

Start the TestRun and observe the animation in IPGMovie.

5.2.4 Collision Detection

In this step, an automatic collision detection system is added to the simulation. For this, modifications need to be made to the vehicle, as well as the maneuver.

Activate the option *General* in *MainGUI > Parameters > Car > Sensors > Collision.*

By default the collision detection uses the *Vehicle Outer Skin*, defined in *MainGUI > Parameters > Car > Misc.* If a outer skin with more detail is necessary, *2D Contour* can be selected.

Furthermore, the collision detection also needs to be defined within the Maneuver.

In the *Maneuver* dialog, click on the defined maneuver *Maneuver Step 0*. On the right side, in the field labelled *End Condition*, type: *Sensor.Collision.Vhcl.Fr1.Count*>=1

For notification that a collision has taken place, a log message can be delivered. If the detection counter is greater or equal to 1, the maneuver control will continue to produce the error log. In order to do so, the following Real Time Expression needs to be added to the Minimaneuver Commands dialog:

```
Eval (Sensor.Collision.Vhcl.Fr1.Count>=1) ? LogErr ("Collision detected!")
```

The function *LogErr* will stop the simulation and print "Collision detected!" to the session log.

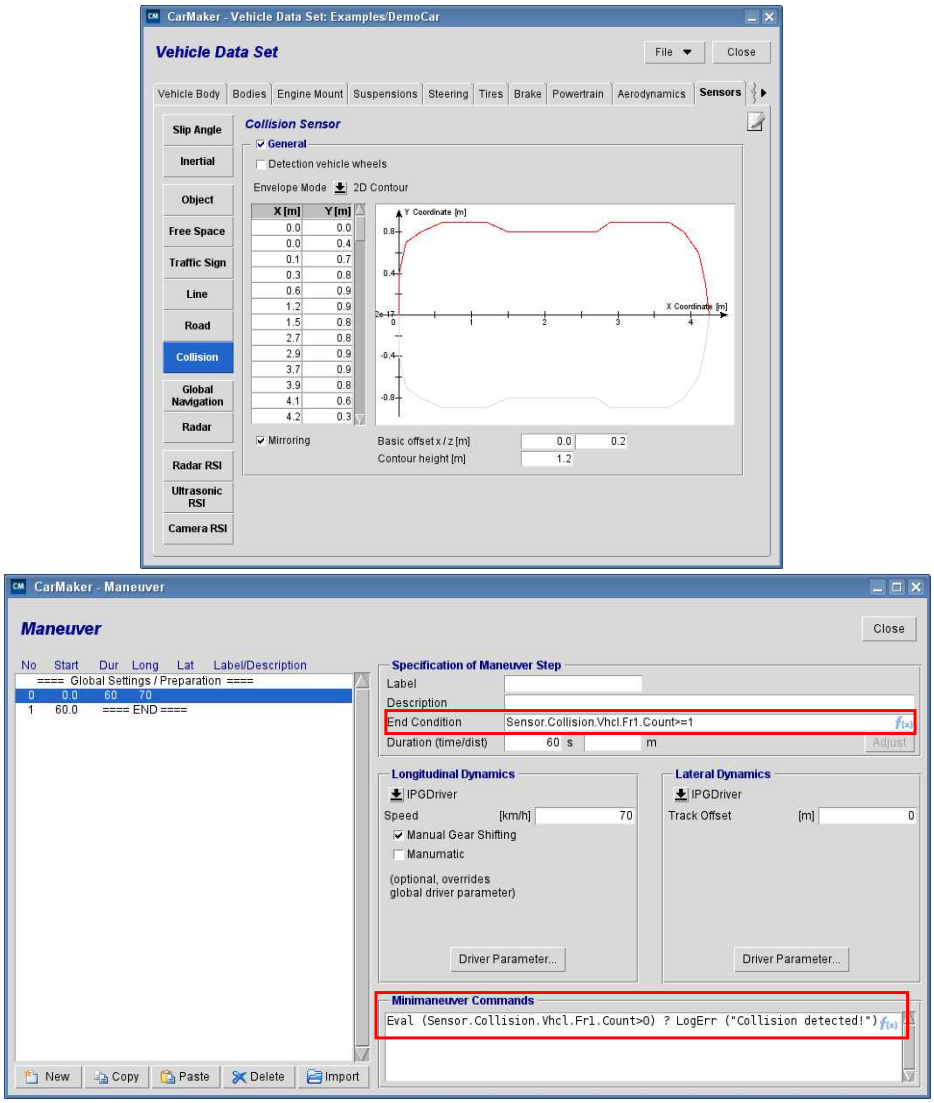


Figure 5.21: Implementing the collision detection

5.2.5 Setting up an AEB System

Currently you are using the CarMaker *DemoCar*. This vehicle is often chosen for simple examples when no extended features are required. For specific testing scenarios a different vehicle can be used or the DemoCar needs to be adjusted.

For the AEB experiment an *AEB system* is required. Since the DemoCar isn't equipped with one of these, the following exercises will demonstrate how to add a *sensor* and *controller* to the vehicle to emulate an AEB system.

Define a sensor for the longitudinal control of the vehicle: *CarMaker main GUI > Parameters > Car > Sensors > Object*.

Click on *Add* to add an object-sensor and name it *Radar*. Edit the rest of the sensor parameters so they match those in the figure below.

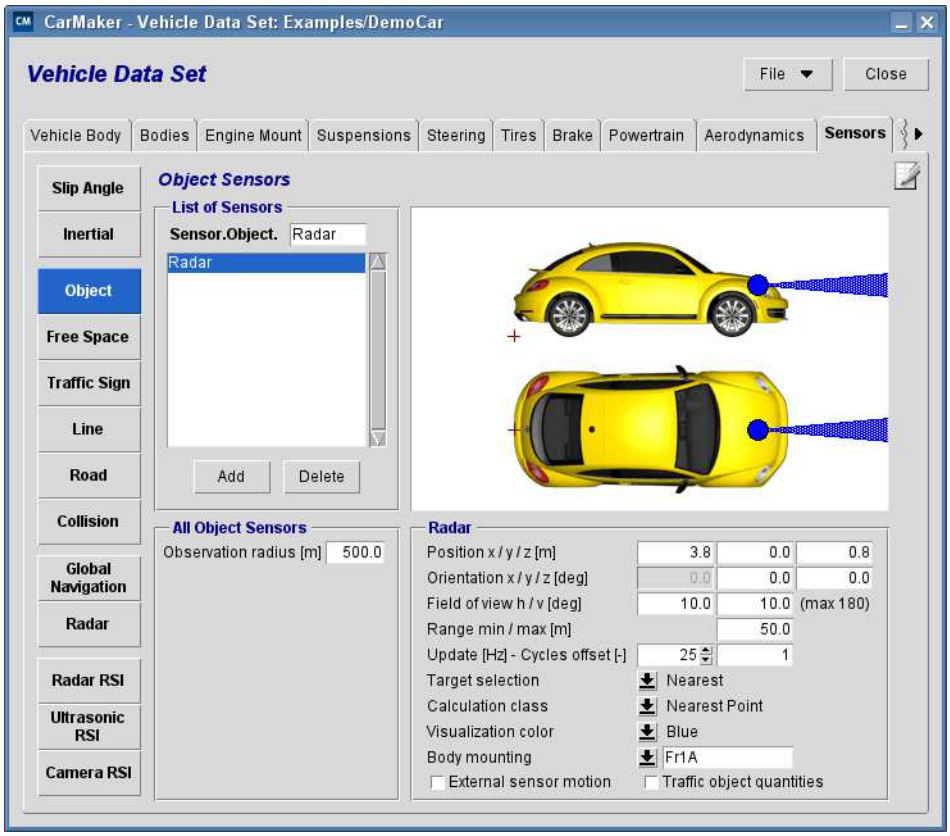


Figure 5.22: Definition of the Object sensor named "Radar"

Implement a controller: *CarMaker main GUI > Parameters > Car > Vehicle Control*.

For *Control Model 0*, select *Generic Longitudinal Control* and set the *Referenced ObjectSensor* to *Radar* (the object sensor created before).

Fill in the other parameters according to the figure below.

Figure 5.23: Controller for longitudinal vehicle control

The last step is to extend the maneuver end condition by another option. The aim is to stop the simulation if a collision is detected or the car brakes to stand still.

The extension for the end condition is the command:

```
Sensor.Collision.Vhcl.Fr1.Count>=1 || Car.v < 0.01
```

Start the simulation and observe the function of the AEB system at the intersection.

5.2.6 Using NamedValues

In this step, *NamedValues* are introduced. These allow the user to carry out variations of the TestRun using different parameter sets.

The first NamedValue will be set in the IPGDriver maneuver control. NamedValues can be used for parameter variations, but to begin with, they need to be given a default value. In this case the IPG Driver Target Velocity will be defined by a NamedValue.

Open the *Maneuver* dialog and click on *maneuver Step 0*. In the *Longitudinal Dynamics* field, change the entry for the *Speed* to $*Speed=70.*

Now the Quantity for the IPGDriver's Target Speed has been named "Speed" and has been preset to 70 km/h. Now, also the traffic object's end condition will be adapted.

In the *Traffic* GUI, change the *end condition* of the traffic object's first maneuver to

```
Car.Road.sRoad>=(800-($Trigger=150))
```

$Trigger is a NamedValue with a default value of 150. The test car reaches the intersection at Car.Road.sRoad=800. Thus, the value of $Trigger determines the distance of the test car to the intersection at which the traffic object begins driving onto the intersection.

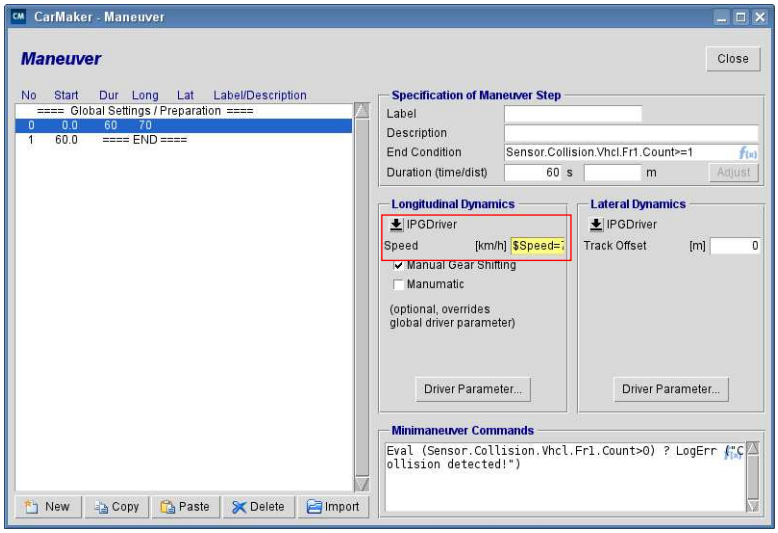


Figure 5.24: TestRun modification using NamedValues (1)

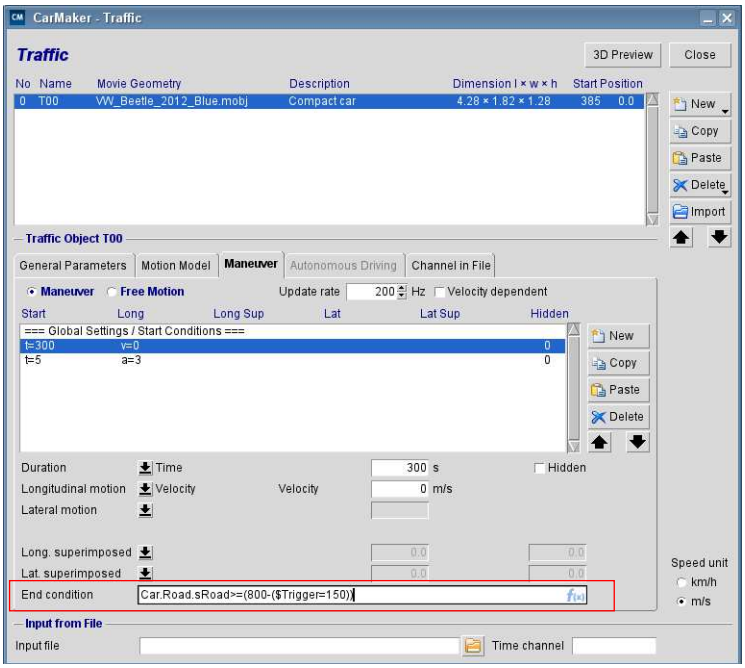


Figure 5.25: TestRun modification using NamedValues (2)

Additionally, the *horizontal aperture* of the object sensor is set to another NamedValue *$RadarHoriz=10*, as shown in the figure below. This will allow the user to vary the detection area.

Save the vehicle via *File > Save* in the Vehicle Data Set Editor and then save the TestRun via *File > Save* in the CarMaker main window.

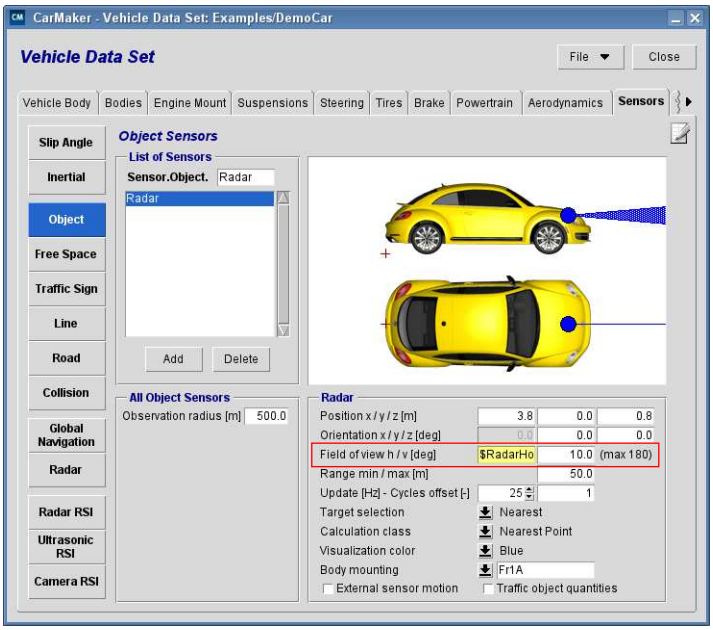


Figure 5.26: TestRun modification using NamedValues (3)

5.2.7 Executing Multiple TestRun Variations using the TestManager

In this last step, the user will create multiple variations of the driving experiment using the NamedValues that were defined in the previous chapter.

Open the Test Manager: *CarMaker main GUI* > *Simulation* > *Test Manager* and add a TestRun to the TestSeries by left-clicking *Add* > *TestRun*. Select the TestRun that you've built up until now by clicking on the folder icon right of the file labelled *TestRun*. This opens a browser where you can navigate to the according TestRun file.

The figure below shows how to add parameters (Step 1) to a variation using the type: NamedValue (*NValue*) and how to create the variation itself (Step 2) by clicking on *Add* > *Variation*.

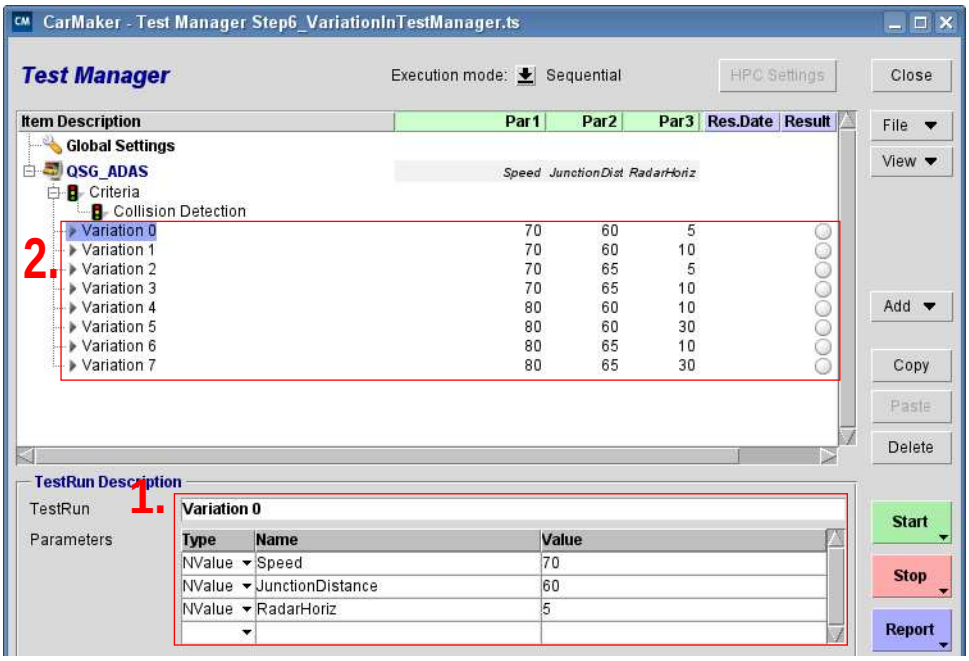


Figure 5.27: Variation with TestManager (1)

Furthermore, a condition for successful completion of the TestRun is added. For this, select the TestRun and then go to *Add* > *Criteria*. A successful completion *good* is defined on the tab *Evaluation* by the condition stated below.

```
[get Sensor.Collision.Vhcl.Fr1.Count] < 1
```

In this statement, the function *get* is used to access a CarMaker quantitiy.

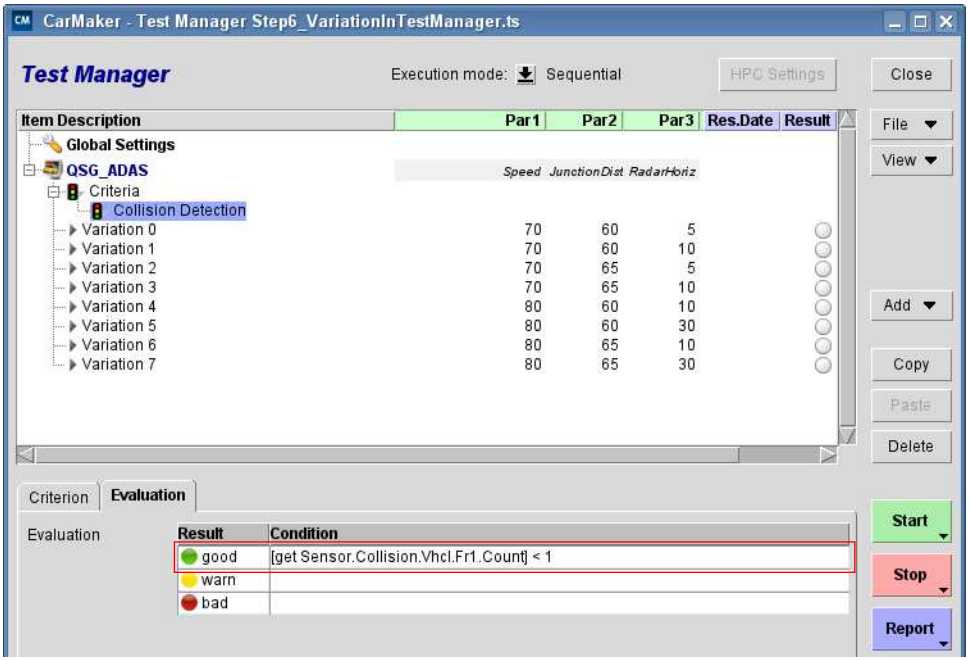


Figure 5.28: Variation with TestManager (2)

In the variations created, the NamedValues previously defined in the TestRun can be modified. Here, the test vehicle's speed, the triggering condition for the traffic object to enter the intersection and the horizontal aperture of the sensor are varied.

5.3 TestRun with Main Focus: Powertrain

This chapter explains how to build a 48V P1 and P2 hybrid and how to compare their fuel consumption on an RDE track.

5.3.1 Creating the 48V Hybrid P1

This chapter shows, how to create a 48V P1 hybrid out of a conventional vehicle, in this case: the DemoCar.

Load the DemoCar: *Main GUI > Car > Select,* Select *ProductExamples > Examples > DemoCar.*

Change the powertrain layout from conventional to P1 hybrid: *Parameters > Car > Powertrain*. Right-click anywhere in the window and select *Parallel Hybrid -> P1*.

We want to keep the conventional engine and just add a stronger starter generator.

Go to *Drive Source* and select *Starter Motor*. The *Ratio* is set in the *General* tab, change it to 1 (directly on the crank shaft).

Change the *Mechanical power* to 12 kW in the tab labelled *Torque,* the *Maximum torque* to 130 Nm and the *Maximum rot. speed* to 8000 rpm.

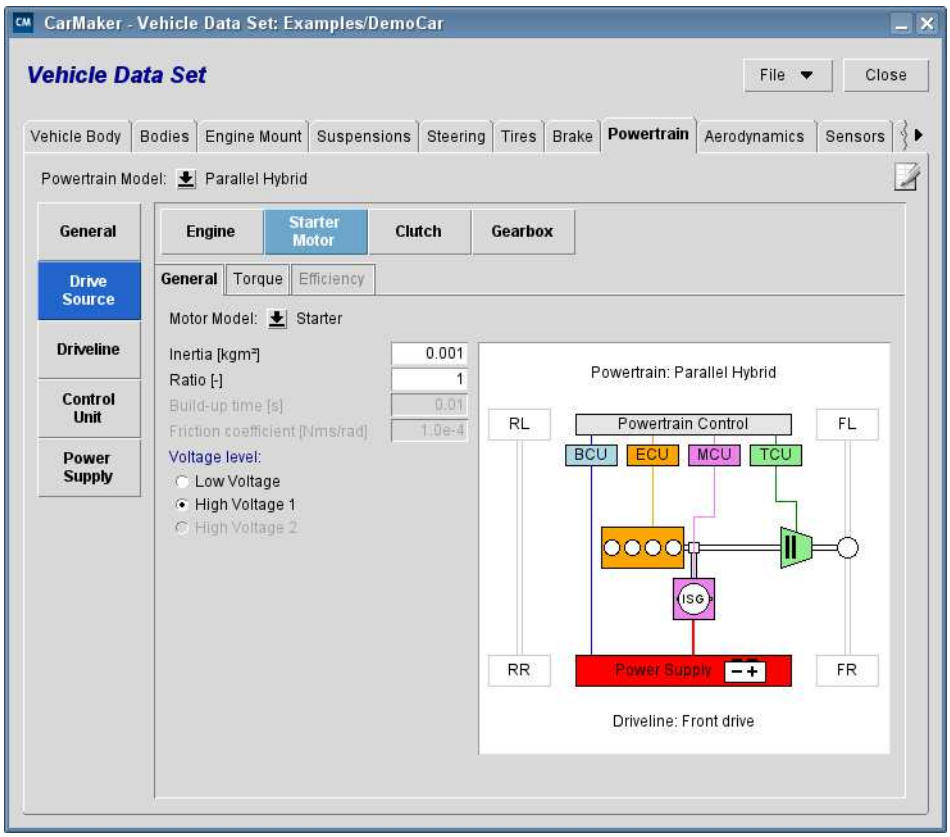


Figure 5.29: Parameterizing the Starter Motor

After defining the hardware of the drive sources, we adapt the control strategy. Since we have higher torque using the electric motor, we have to change the gas pedal-to-torque converter. The maximum torque at full gas should now be 315 Nm (185 Nm from the combustion engine and 130 Nm from the electric motor).

Click on *Control Unit*, then *PT Control* and in the tab labelled *Gas*, change the *Max torque at full gas* to 315 Nm.

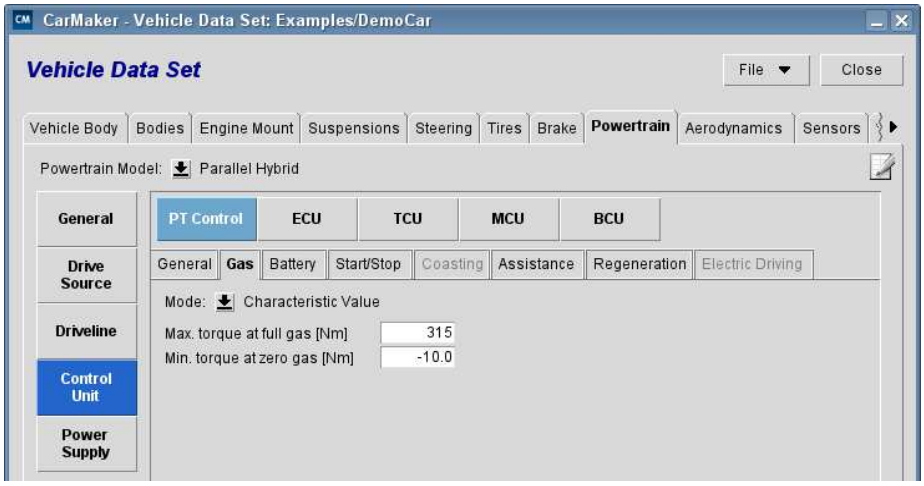


Figure 5.30: Parameterizing PTControl

The created vehicle has an automatic transmission. For realistic shifting behavior, we must adapt a few values.

Go to the *TCU* tab and then to *Shifting Limit*s. Change the *Shifting Limits* from *1D Look-Up Table* to *2D Lookup Table* so that the gas pedal position influences the shifting strategy.

Now the minimum rotation speed for gears 4 and 5 at 0% gas pedal position are too high, resulting in oscillating shifting at some velocities so we must change the values for gears 4 and 5 at 0% gas pedal position.

Change the *Rot. Speed Min* values for gears *4* and *5* at *0.0 Throttle* to 1250 rpm.

For a 48 V hybrid vehicle we must change the size and voltage of the high voltage battery. The electric converter of the vehicle should have 3 kW.

Click on the *Power Supply* tab and under *General*, click on *Converter HV1 LV.* Change the *Max. converter power* to 3 kW.

The battery itself is parameterized under the *HV Battery* tab. Change the *Capacity* to 10 Ah and the *Idle voltage* to 48 V. The *Maximum power* should be 14 kW.

The last step is allowing the brakes to regenerate. This is done in the *Brake* tab. Click on the *Brake* tab, go to *Control* and set the *Regenerative Brake Mode* to *Serial.*

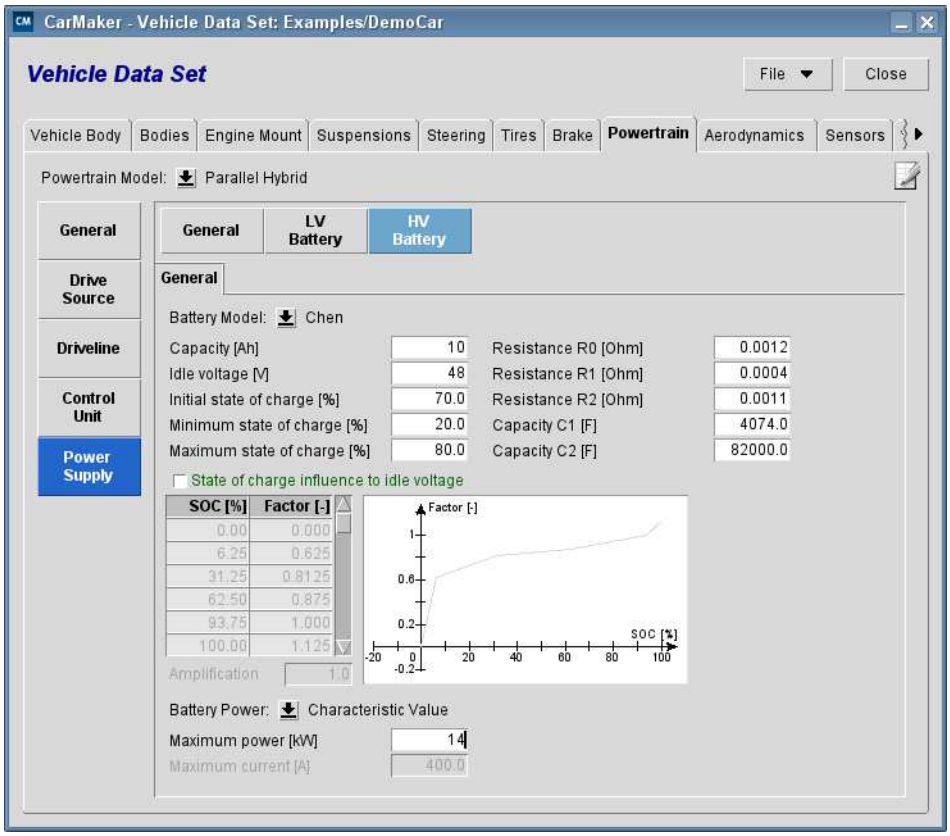


Figure 5.31: Parameterizing the HV Battery

Now a P1 48V hybrid has been created out of a conventional vehicle. Save it via *File > Save As > DemoCar_48V_P1.*

5.3.2 Importing the Road

As we have already created our vehicle, we now need a road. For this, the RDE track which is already included in CarMaker will be used. Alternatively, the user can import his own roads, e.g. by using *.kml data or other sources. Nevertheless, to keep it simple, one of the Product Examples will be used.

Open the Scenario Editor via *Parameters > Scenario / Road* in the CarMaker main GUI.

Insert a *Road segment* Type *File* by clicking and holding *Road segment* in the *Road* tab and choosing *File*. Now, click anywhere in the drawing area to set the first point. This point also becomes the origin for the x, y, y coordinate system.

After confirming the direction vector with a second click (the direction doesn't play a role in this example) a browser opens where you can select the file *Product Examples/Examples/"IPG_RDE_KA_Route.bin".*

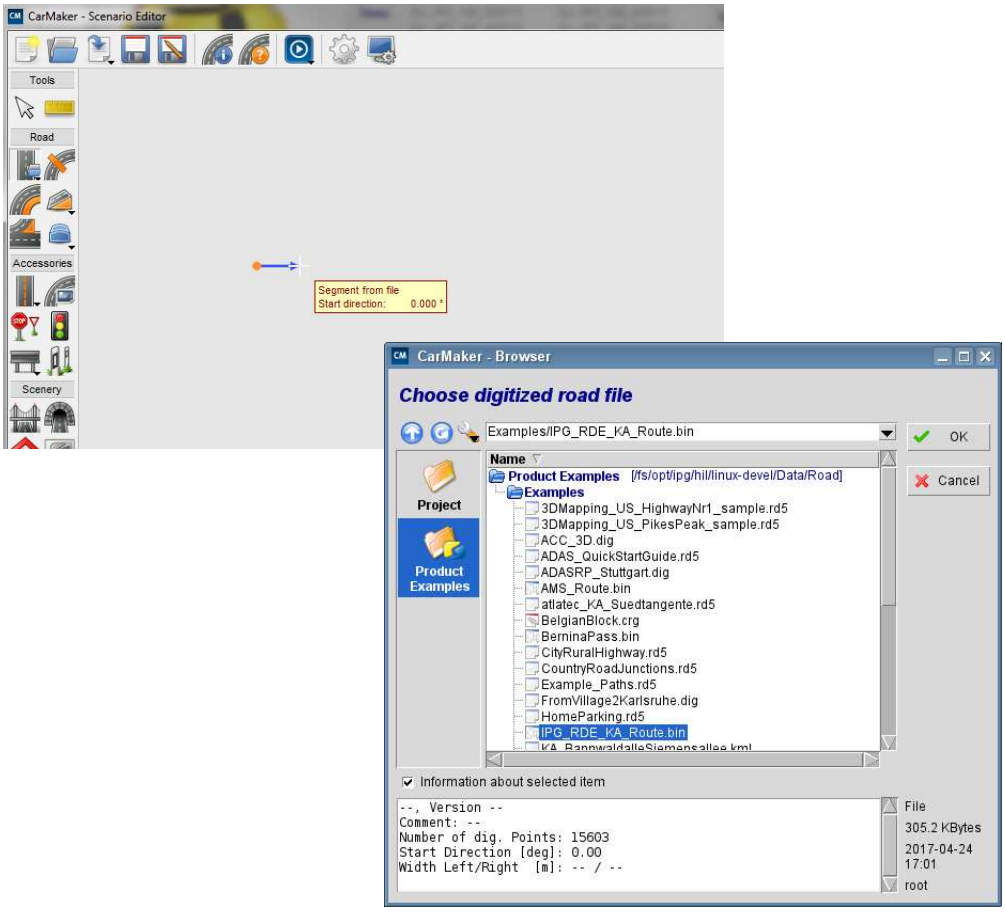


Figure 5.32: Adding a Road segment type File

By pressing *OK* the route is imported. We have the road now, but we have to define a route where the vehicle should drive, too. A further (optional) step is connecting the road's ends to turn it into a closed loop circuit. This way the vehicle will continue driving in the loop until the maneuver comes to an end. Otherwise, the simulation is aborted as soon as the vehicle reaches the end of the road. In this case the user may receive an error notifying that the vehicle has left the road.

Optional: Select the *Connect* Road segment type and connect the open ends of the road to generate a closed loop circuit.

Select the *Route* symbol in the *Traffic* tab: Click on the predefined Path on the right lane of the Link. When a broken, orange line appears, click again to confirm that this is the Path we want to drive on. To finish implementing the Route double-click anywhere close to the Link.

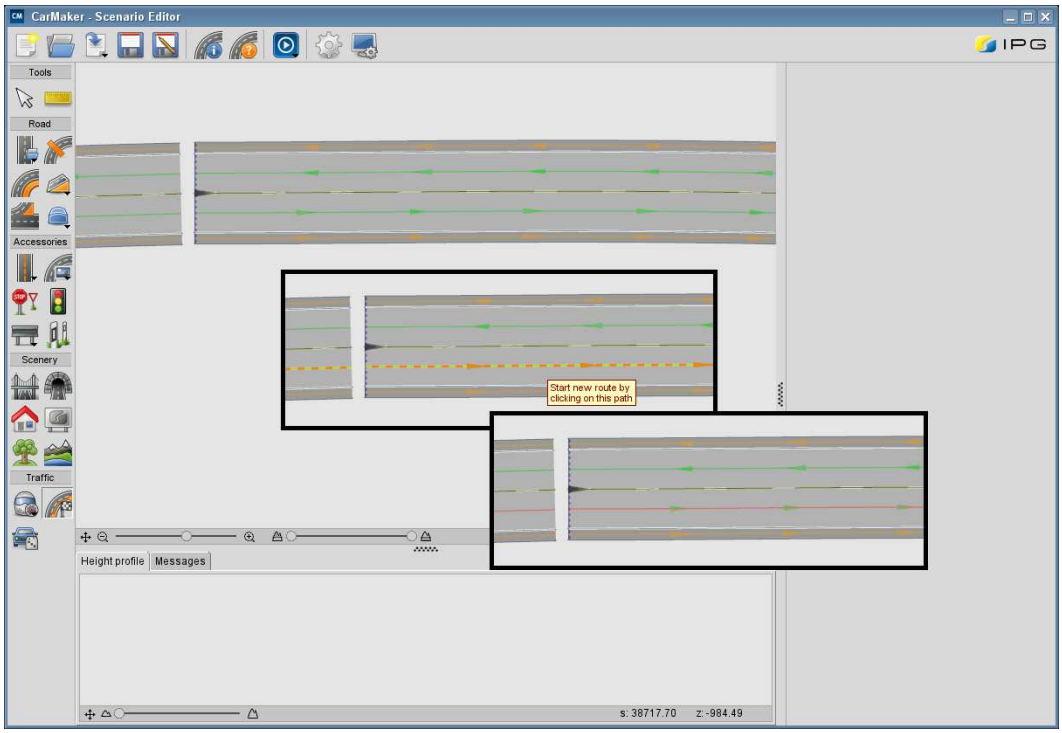


Figure 5.33: Adding a Route

Now the vehicle, road and route are finished. In order for the TestRun to be complete, a maneuver, that tells the driver which actions to proceed, is still missing.

Save the TestRun as *PT_QuickStartGuide_BasicRoad.*

5.3.3 Maneuver and Driver Parameters

In this example, a simple maneuver will be created where the driver model - *IPGDriver* - controls the vehicle in both longitudinal and lateral direction.

Open the *Maneuver* GUI via *Parameters* > *Maneuver* and click on *New* to add a new Maneuver Step. The default maneuver step that is added is sufficient for our example.

Give the driver a defensive characteristic: In the Maneuver GUI, click on *Driver Parameter.* Alternatively you can click on *Parameters* > *Driver* in the main GUI. Right-click anywhere in the Driver GUI and choose *Parameter Set "defensive"* to select the preset for a defensive driver.

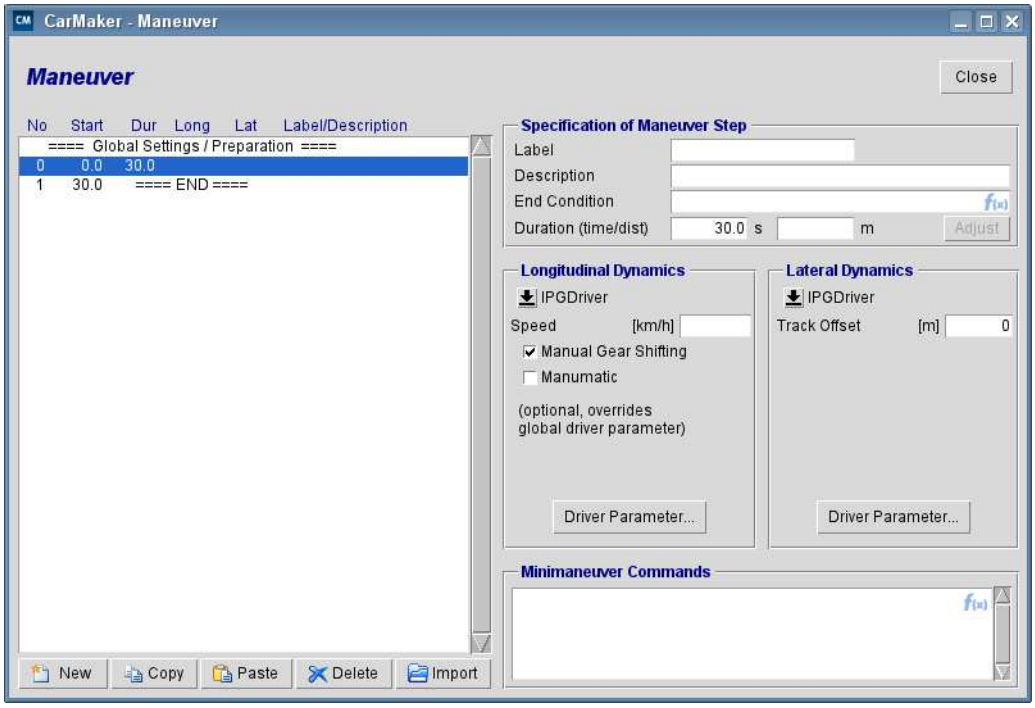


Figure 5.34: Adding a default maneuver step

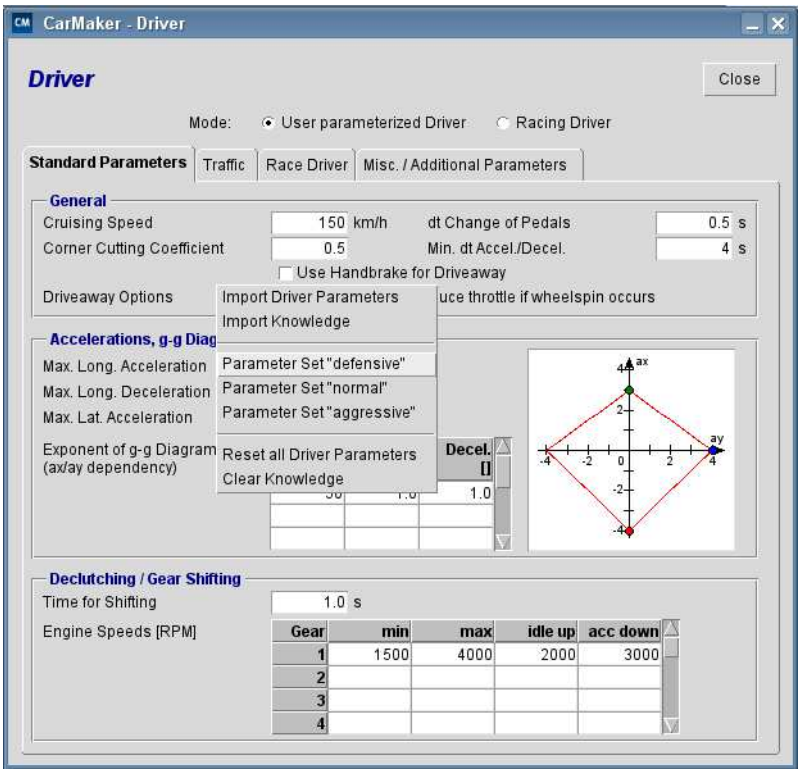


Figure 5.35: Choosing the defensive driver preset

Now all necessary components have been implemented and the TestRun can be simulated for the first time. First, save the TestRun.

Save the TestRun as *PT_QuickStartGuide_Maneuver*.

Start the simulation: In the main GUI, set the *Simulation Perf.* to *realtime* and then press the green *Start* button. Watch the simulation in IPGMovie.

If you choose *Speed and Powertrain* in the menu *View* > *Overlay Left*, you can observe the different operating strategies.

5.3.4 Implementing Speed Limits

watching the simulation, it can be seen that there is no speed limit on the road. This means that the driver would usually try to reach the desired speed he is given by the user in the Maneuver step. In this case, 100 km/h are already preset by the defensive driver model.

The next step is to set up a speed limit. First, the TestRun should be saved again as *PT_QuickStartGuide_SingleSpeedLimit*.

Set a speed limit: In the Scenario Editor, navigate to the beginning of the track. Look for the x- and y-values (0,0) or right-click in the drawing area and select *Default View*.

Click and hold the *Markers* icon and in the dropdown-menu, select *Driver speed*. Click on the road to select the Link and then click once to define the starting and once to define the end point of the road section with the speed limit.

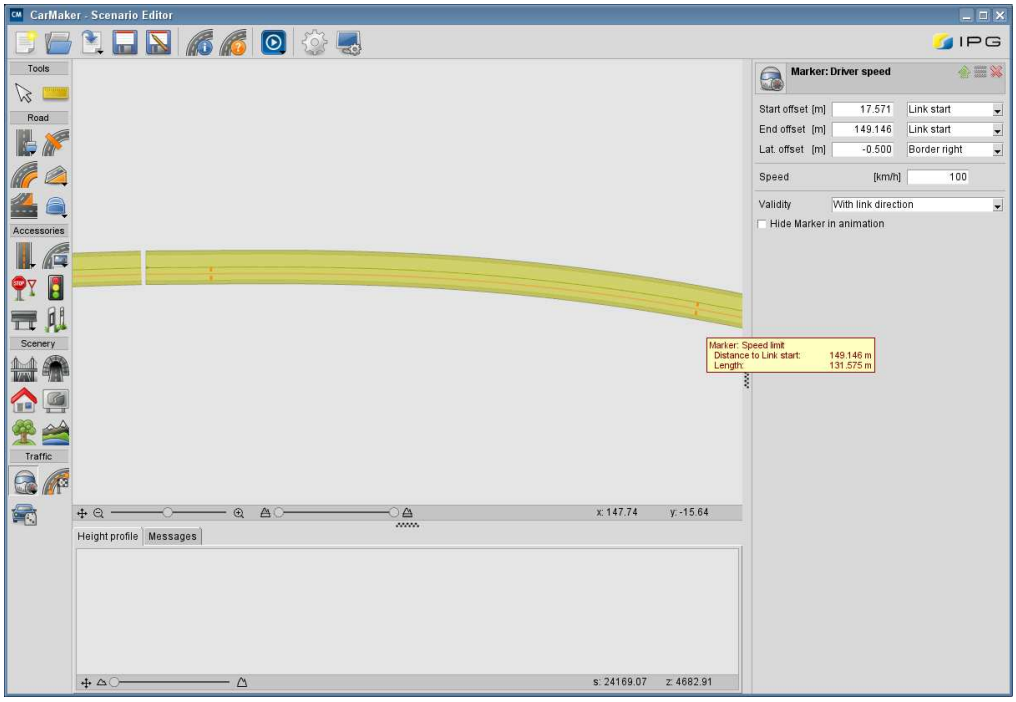


Figure 5.36: Setting a Marker, type: Driver speed

After setting the start and end points for the speed limit, the user has the possibility to change various settings afterwards as well. This can be done in the small tab that opens up on the right-hand side of the Scenario Editor after the speed limit has been selected by left-click in Selection mode. The figure below shows which parameters can be set.

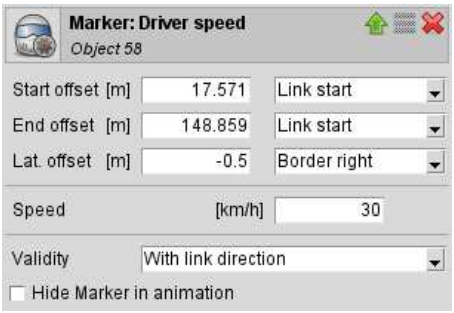


Figure 5.37: Driver speed parameters

Please note that *Traffic signs* allow the user to add traffic signs to the road network ONLY for beautification purposes. The driver will not react to these, as they can only be detected by Traffic Sign Sensors. For signs that have an influence on the driver's behavior, markers must be chosen.

Set the *Speed* for the speed limit to 30 km/h.

Save the TestRun by clicking the Save symbol in the Scenario editor and start the simulation.

The driver will now only drive at a maximum of 30 km/h because he always tries to follow the guidelines.

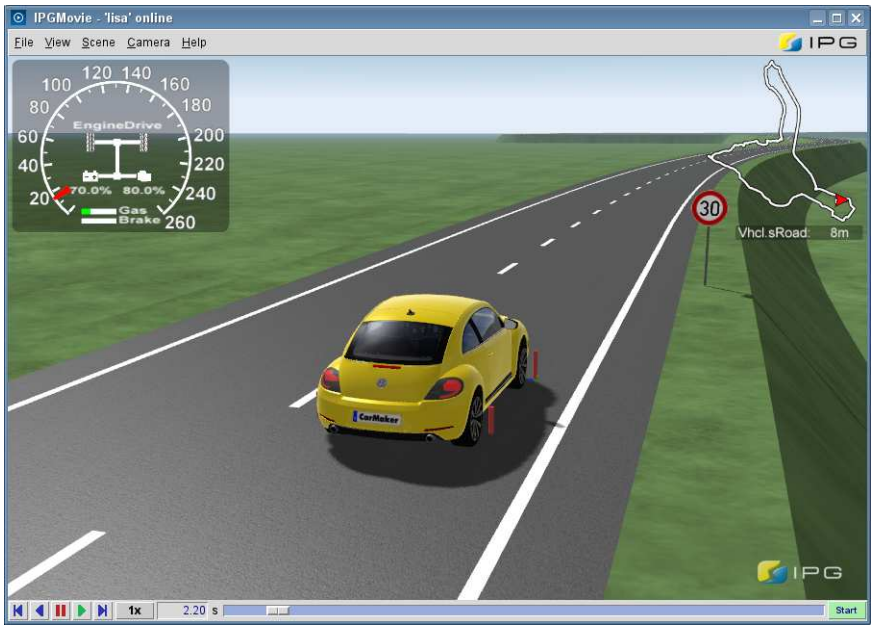


Figure 5.38: Driver passing the starting point of the speed limit

Since the considered TestRun is about 90 km long, it isn't very efficient to set all the speed limits manually. Here, we have the option of importing speed limits from other TestRuns. For this case, the speed limits will be exported from an example TestRun.

First, the TestRun should be saved as *PT_QuickStartGuide_RDE_Speedlimits*.

Open the Scenario Editor, click and hold the *Import road definition* icon in the top bar and choose I*mport from TestRun*. Select the TestRun *Product Examples > Examples/ Powertrain/DrivingScenarios/Karlsruhe_RDE_IPG_FreeDriving*.

Now all the traffic signs and traffic lights from the RDE TestRun should have been imported. Save the TestRun in the Scenario Editor.

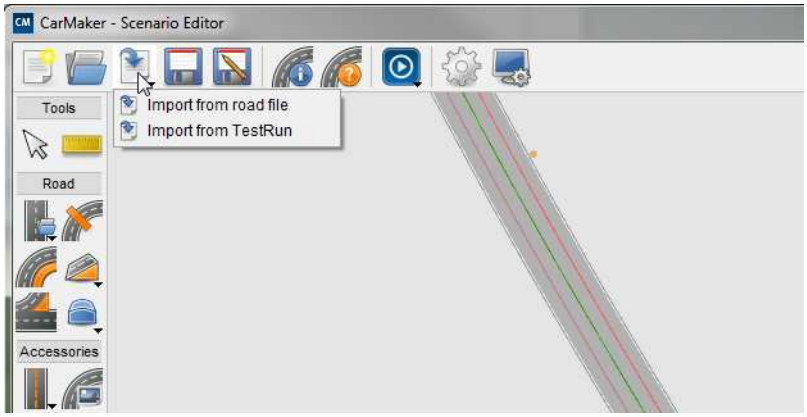


Figure 5.39: Importing components from another TestRun

Open the *Maneuver* GUI and change the *Duration* of the Maneuver step: Delete the 30s time condition and instead, define a restriction of 5000 m regarding the distance.

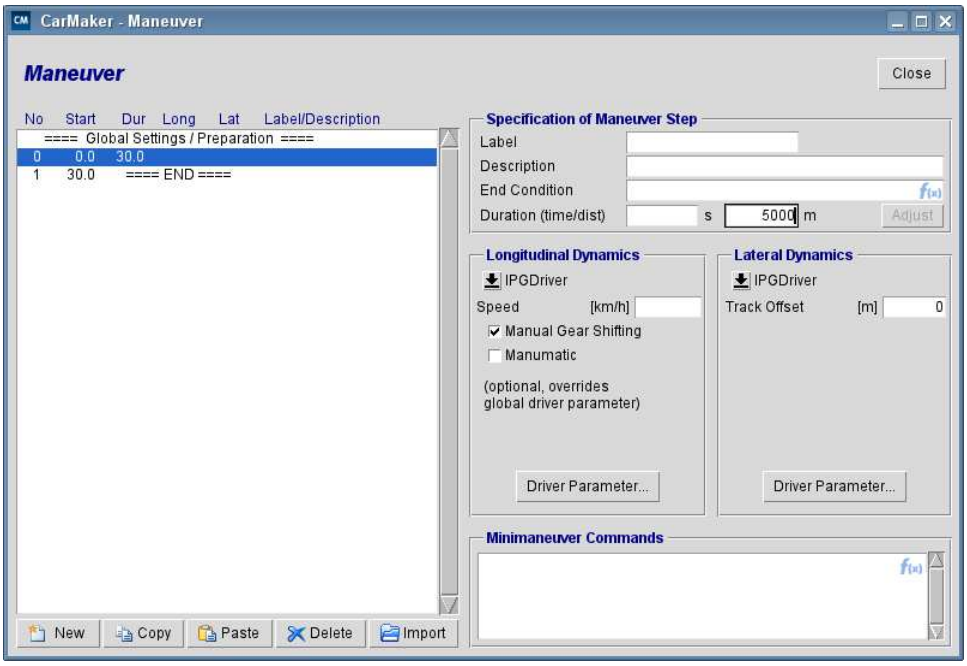


Figure 5.40: Changing the duration of the maneuver step

Additionally, the fuel consumption is to be logged. In order to do so, a maneuver step that must have a duration of 0 s needs to be added. These types of maneuver steps (Duration = 0s) are a special kind of step where the actions defined here are always carried out at the end of a TestRun, no matter at which point the simulation ended. Mostly, this kind of step is used to initiate post-processing actions.

Create a new Maneuver step by highlighting the last line in the Maneuver step list (END) and clicking New.

Set the *Duration* to 0 s and add the following command to the *Minimaneuver Commands* field:

Eval first() ? Log("Average fuel consumption: %.1f l/100km",PT.Control.Consump.Fuel.Avg)

This command will show the average fuel consumption at the end of each simulation in the *Session Log* (Main GUI > Simulation > Session Log).

Save the TestRun and start the simulation.

The vehicle drives through the RDE scenario with different speed limits and traffic lights on it's way. At the end of the TestRun, the session log shows the average fuel consumption. You can speed up the simulation in the Main GUI by selecting the desired speed in the CarMaker main GUI.

5.3.5 Creating the 48V P2 Hybrid

Now the 48V P2 hybrid, that can drive purely electrical, can be created. The 48V P1 hybrid that was created in the first chapter can be used as a base to make parameterizing easier. First of all, the powertrain architecture needs to be adapted:

Open the *Vehicle Data Set* and go to the *Powertrain* > *General* tab. Right-click anywhere in the dialog window and select *Parallel hybrid -> P2.*

As the P2 hybrid has a conventional starter motor, we'll have to change power and torque from the P1 hybrid. Furthermore, the electric motor has to be defined.

Under *Drive Source* > *Starter Motor*, go to the *General* tab and set the *Ratio* to 4. Under the *Torque* tab, set the *Mechanical power* to 3 kW and the *Maximum torque* to 50 Nm.

Under *Electric Motor* > *Torque*, set the following performance data for the electric *Motor Model*: *Mechanical power* = 12 kW, *Maximum torque* = 130 Nm. For simplicity reasons, apply the same parameters for the *Generator Model* beneath.

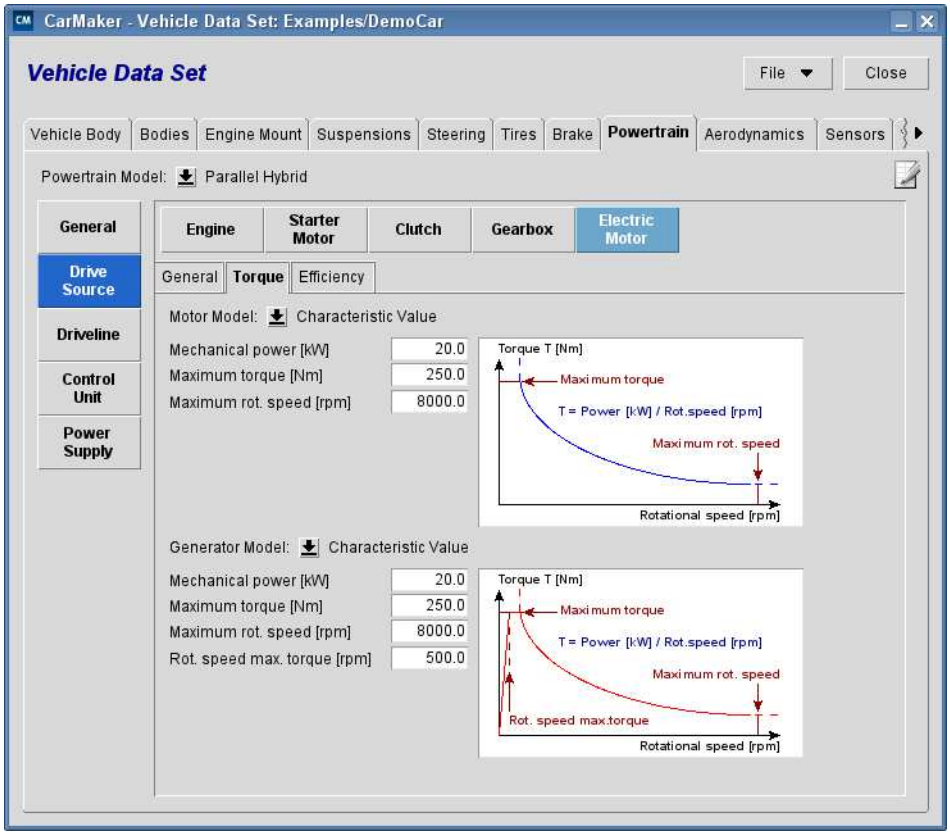


Figure 5.41: Parameterizing the electric motor

The P2 hybrid has a clutch between the engine and the electric motor. With this clutch, it is possible to activate coasting while the engine is turned off.

Go to *Control Unit > PT Control > Coasting* and activate the option *Expert parameters*. You can leave the default values as they are.

In the *Electric Driving* tab: At the moment, electric driving is only allowed to a limit of 50% SOC and up to 50 km/h. Please change the SOC limit to 25%, the range to 10% and the speed limit to 75 km/h (for "Active cruising" with engine off).

Save the vehicle as *DemoCar_48V_P2* and the TestRun as *PT_QuickStartGuide_RDE_Speedlimits_P2.*

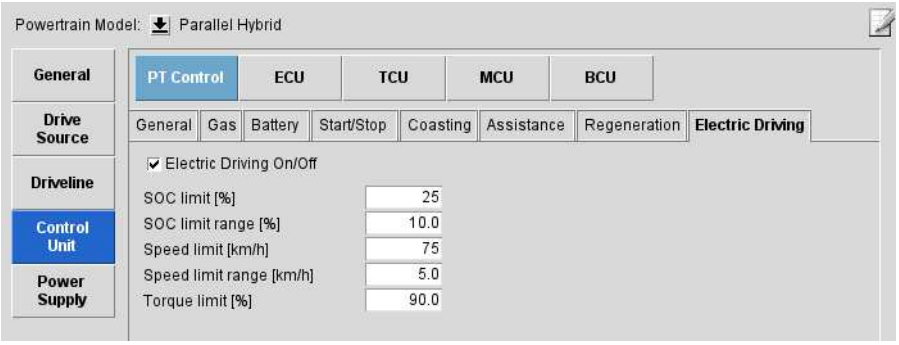


Figure 5.42: Parameterizing PT Control

5.3.6 Test Automation using the Test Manager

A powerful tool in the CarMaker program is the Test Manager. In this chapter it will be described how to create variations of a given TestRun by changing the vehicle or increasing the payload.

Open the *Test Manager*: CarMaker main GUI > *Simulation* > *Test Manager*.

Add a TestRun: Left-click on *Add* and select *TestRun*. Add the created TestRun *PT_QuickStartGuide_RDE_Speedlimits_P2.* from the previous chapter by clicking on the folder icon.

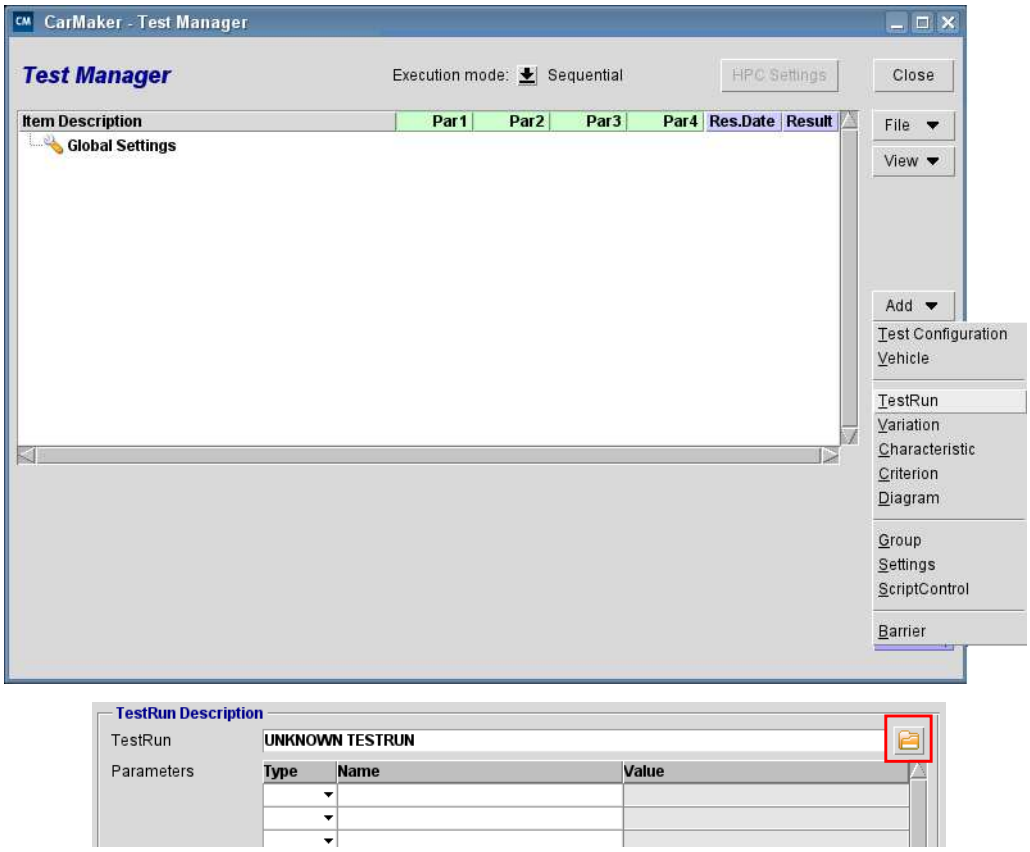


Figure 5.43: Adding a TestRun to the Test Manager

Now, the goal is to execute two TestRuns, first with the P1 hybrid, then with the P2. Therefore, the user must add a *Variation*. KValues can be used when the user wants to change an entire parameter set, like the vehicle or tires. You can find more information regarding KValues (and NValues) in the CarMaker User's Guide in chapter 7.3 "Variable Types".

Click on *Add* > *Variation* and for the *Type*, select *KValue.* The *Name* must be *TestRun:Vehicle* which means that you want to change the vehicle file.

For the *Value* open the vehicle dialog (Main GUI > *Car* > *Select*) and choose the vehicle you want to use in this variation: DemoCar_48V_P1. Mark the name and copy it. Close the browser and insert the path and vehicle name in the *Value* field in the *Test Manager*.

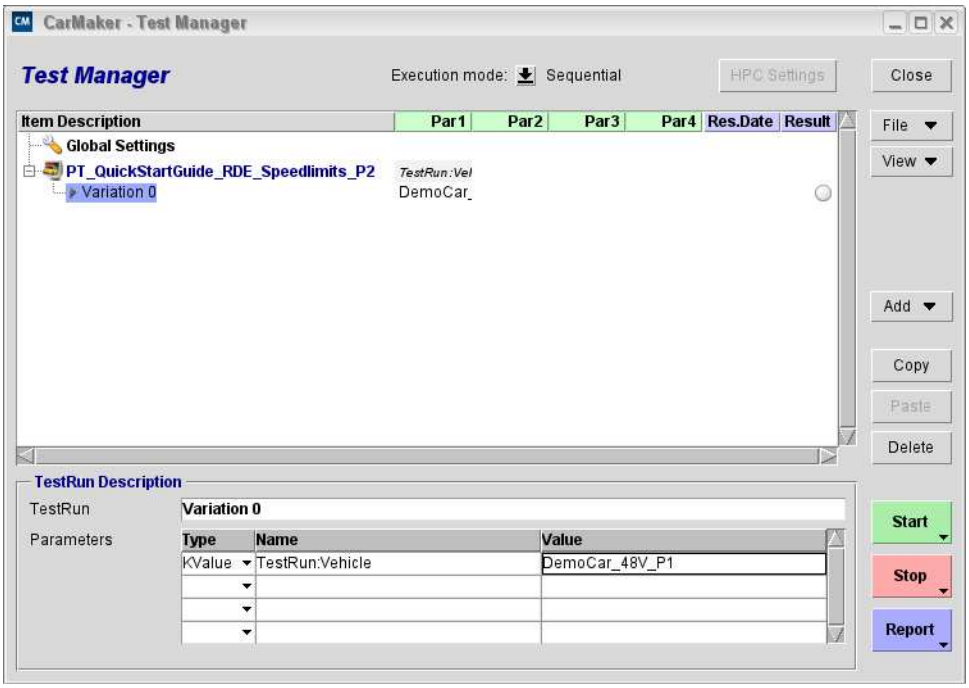


Figure 5.44: Vehicle variation in the Test Manager

Choose a *Description* for this specific variation (e.g. *P1 base*) and then copy and paste the variation once.

The second variation should run with the P2 hybrid, so change the *Value* of the *KValue* to *DemoCar_48V_P2* and change the *Description* of this variation to *P2 base*.

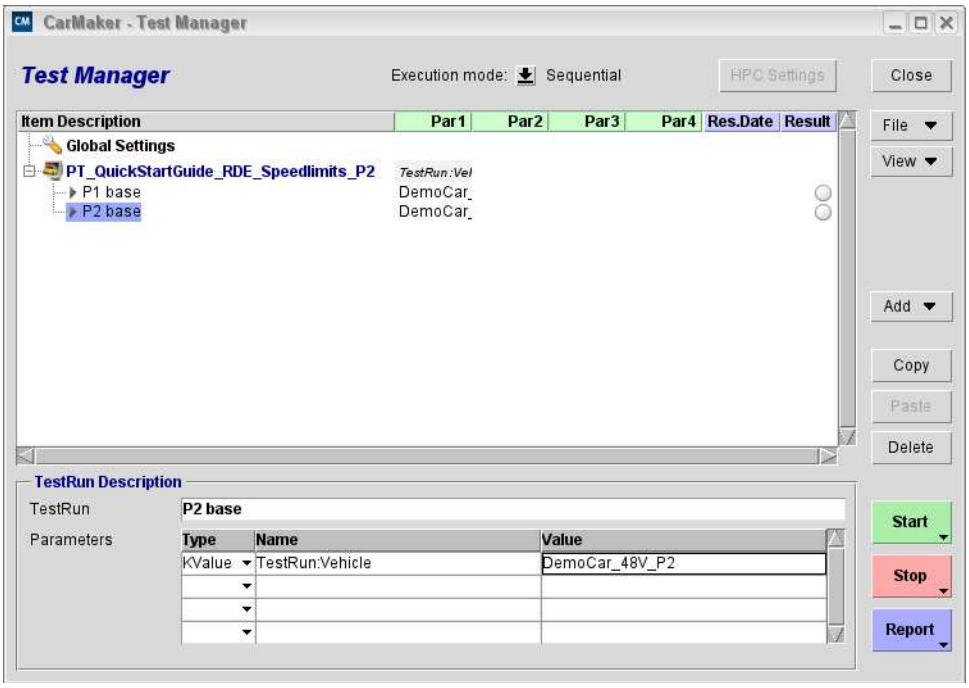


Figure 5.45: Second vehicle variation in the Test Manager

Click on *Start > Run* in the Test Manager to initiate the test series.

Now a series of two TestRuns has been started. By opening the Session log, the user can compare the fuel consumption of the two different cars.

At last, we want to examine the influence of the load. To do so, we have to define a *NamedValue* (*NValue*).

In the CarMaker main GUI, open the example TestRun "*Step4_RDE_SpeedLimits*" that you can find in *Examples/Powertrain/_QuickStartGuide/*. Make sure that the *DemoCar_48V_P1* vehicle is being used.

In the main GUI, click on *Select* in the *Load* field. The *Loads* dialog opens where you can define different loads at various positions.

Set the *NValue*: In the first field where the mass of a load is defined, type *$load=0*. This is how *NValues* are initiated. They begin with a *$* sign, followed by a user defined name that has been chosen for the parameter of this field and must always end with a *default value*.

In this case. The default value is 0 kg. This means that if nothing else is defined, an extra load of 0 kg, meaning no load, is set. The load can now be varied in the Test Manager.

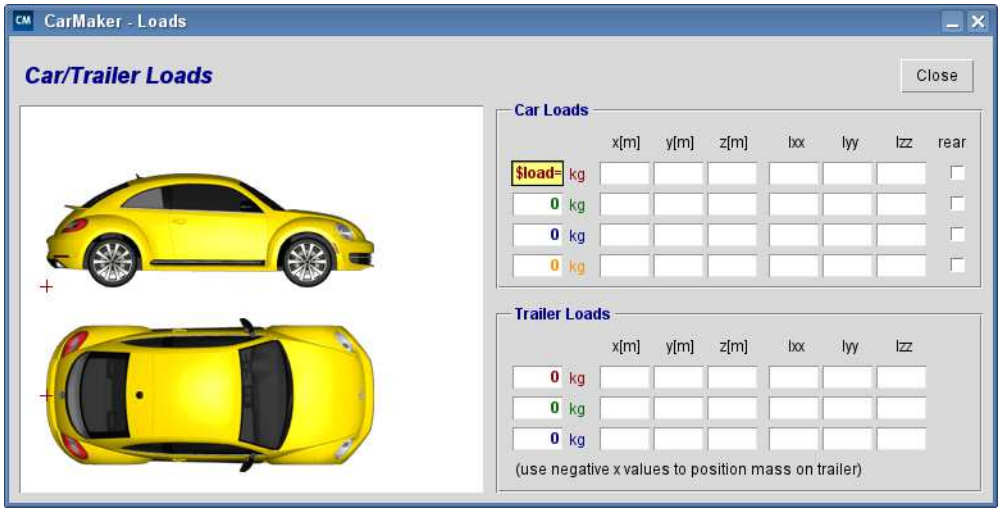


Figure 5.46: Setting the Named Value

Save the current TestRun as *PT_QuickStartGuide_RDE_Speedlimits_P1.*

Open the Test Manager and replace the defined TestRun with the one you just saved.

Add a new variation (*Description: P2 base payload 500*) by copying and pasting the *P2 base* variation. Add a *Parameter*, for the *Type*, choose *NValue*, the *Name* should be *load* and for the *Value*, set *500.*

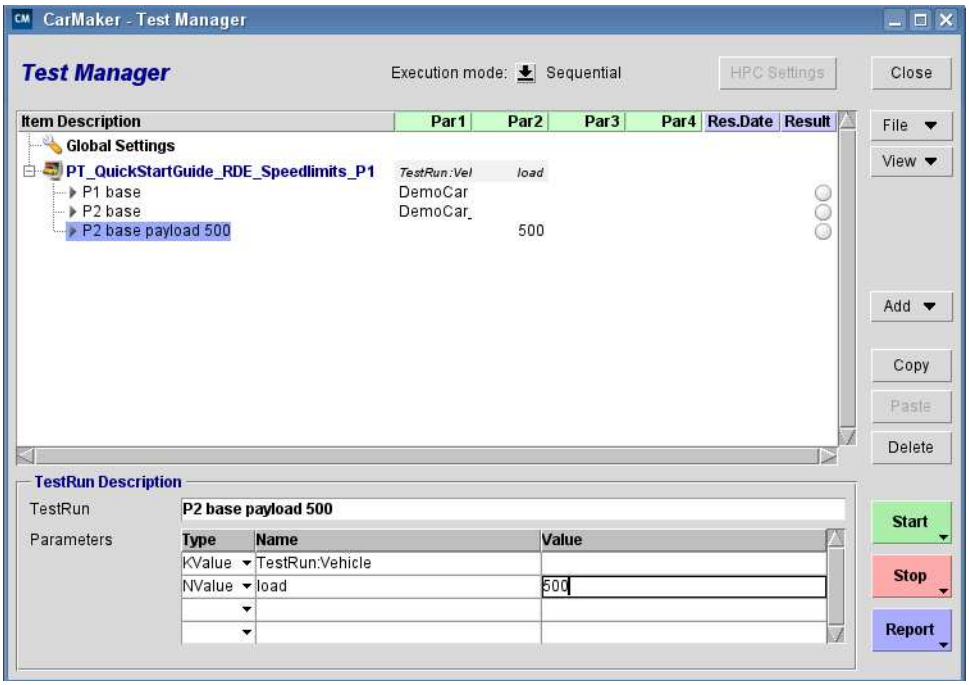


Figure 5.47: Adding a variation with a Named Value

Save the TestSeries by clicking *File* > *Save as* and name it *Powertrain_TestSeries*.

Run all variations and compare the different fuel consumption levels.

5.4 TestRun with Main Focus: Scenario Editor

In this chapter the user will be given instructions to create a fully parameterized TestRun with a main focus on the Scenario Editor. For a TestRun to be sufficiently parameterized, models for the vehicle, driver, road and maneuver must be defined. Further components are optional.

5.4.1 Building a Road Network using the Scenario Editor

Load the prepared TestRun *Step1_TestRunWithoutScenario*:
In the CarMaker GUI, click on *File* > *Open,* select the product data pool, then click on *Product Examples* > *Examples* > *BasicFunctions* > *Road* > *_QuickStartGuide* > *Step1_TestRunWithoutScenario*.

Save the TestRun under a different name, so that you can retrieve your work:
In the CarMaker GUI, click on *File* > *Save as* and choose *Project* on the left bar in the dialog. With right-click, create a new folder (if you haven't already) "My_TestRuns". Double-click on the new folder and enter any name for the TestRun, e.g."My_TestRun_ScenarioEditor" > *OK.*

Close IPGControl and Instruments for the moment, but keep IPGMovie open.

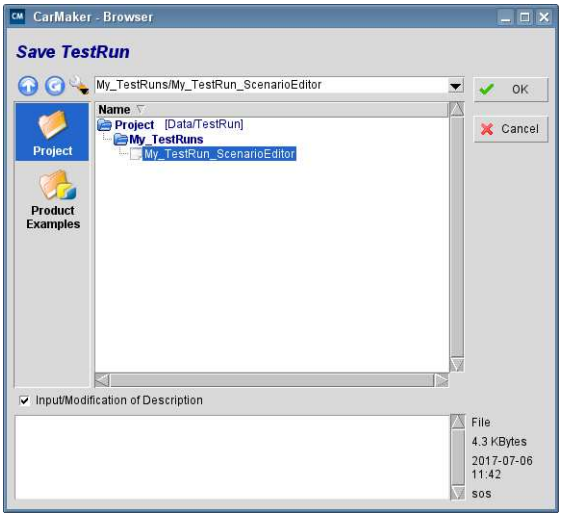


Figure 5.48: Saving the TestRun under a different name

The following chapters will now show how to extend this TestRun, for which only a vehicle and simple maneuver have been defined for up until now. Work can be saved every now and then via *File* > *Save* in the CarMaker main GUI or by using the shortcut *Ctlr* + *S.*

Scenario Editor Basics

The Scenario Editor is a graphical user interface used to parameterize the road and build road networks. The road model, called IPGRoad is a 3D model. A road consists of one or several *Links* that are connected by *Junctions*. The user can define the shape of these Links and Junctions, add a 3D profile (e.g. elevation), define the number of lanes and their widths, change the road surface's friction coefficient, add other elements such as road markings and traffic signs to the scenario and specify the route that the vehicle will drive on.

Generally, there is more than one way to define the course of a track:

- Based on segments (e.g. straights, turns) joint to each other in the Scenario Editor.
- (Measured) road data saved in an ASCII file (text file) that can be loaded into CarMaker. These files need to contain cartesian (x, y, z) or geographic (long, lat) coordinates.
- Road data saved in KML-files (Keyhole Markup Language). These files are created by Google Earth or Google Maps and can be uploaded in CarMaker.
- Combination of segments and road data from ASCII- or KML-files.

All of these options are further described in the User's Guide.

The following examples will explain how to use the Scenario Editor.

Open the Scenario Editor: In top bar of the CarMaker main GUI, click on *Parameters > Scenario / Road*.

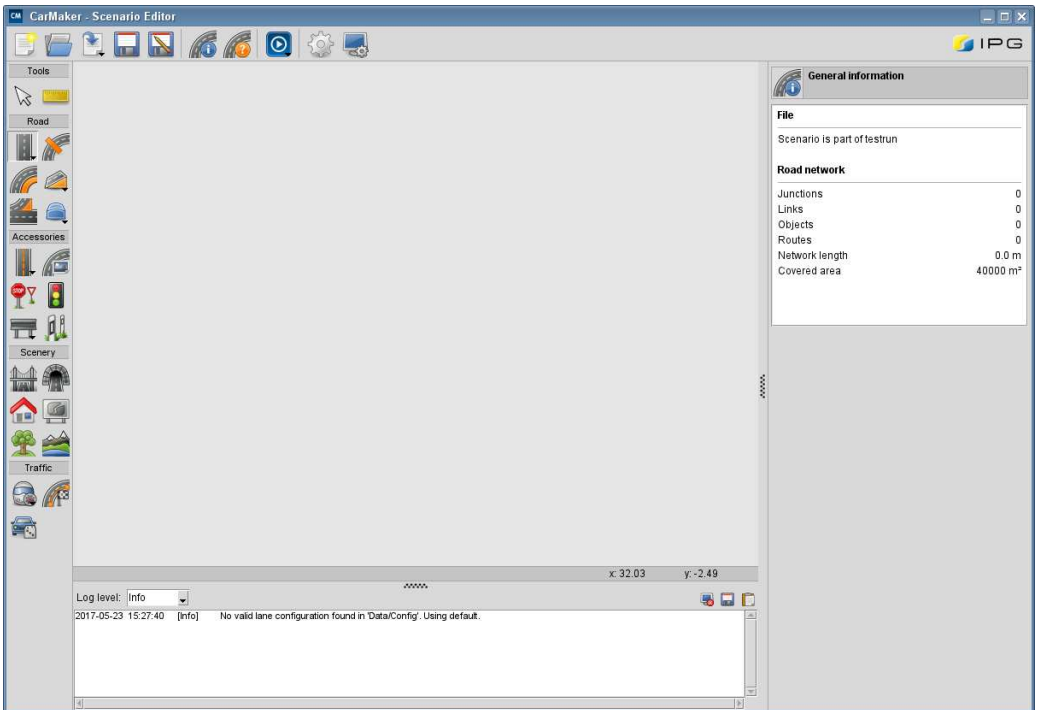


Figure 5.49: Scenario Editor GUI

In this GUI, the road network and road properties are defined. A birds eye view visualization helps the user see what is currently being modeled.

- Tabs on the left side of the GUI:
  - **Tools:** In this tab the user can switch to *Selection mode* for a standard cursor and measure random distances on the road by clicking the ruler (*Measure distance*).
  - **Road:** This tab is used to create *Road segments* (straight, curve, clothoid, etc.), *Lane sections, Lanes, elevation profiles* and *Bumps* (local road surface deformations). Detailed information regarding all of these components can be found in the corresponding section of the User's Guide.
  - **Accessories:** Here, road accessories can be added to the existing roads, such as *road markings, road paintings, traffic signs, traffic barriers* and *guide posts*. Again, further information can be found in the corresponding section of the User's Guide.
  - **Scenery:** *Bridges, tunnels, signs, geometric objects, trees* and *terrain* can be added to the simulation to beautify the environment. These components are selected via this tab and can be added to existing roads.
  - **Traffic:** Here, *markers* for special maneuvers or measurements and route definitions for the driver can be set. Further, *traffic objects* can be randomly distributed on the road.

- Display Window in the center of the GUI:
  - This is the space where the user designs the road networks.
  - The 2D preview from a bird's eye view gives an immediate overview of the road network.
  - In selection mode, certain components of the road network are selected in this window. Selected components are highlighted to show which part of the road is currently activated.

- Tab on the right side of the GUI: In this field, information regarding currently selected components is listed and corresponding parameters can be edited. For example when a link or lane is selected, this is where width and coefficient of friction are defined.

- Top menu: In addition to other features, that are further described in the User's Guide, the designed road network can be viewed using *3D-Preview* (the button with the Play-symbol).

Implementing and Checking the Road Geometry

There are two options for checking the road definition: the bird's eye view of the road in the Scenario Editor display window and 3D-preview using IPGMovie. The following exercises will describe how to implement a few road segments that can then be viewed either in the Editor or in IPGMovie.

Using the mouse, navigate to the *Road* section of the tab on the left side of the Scenario Editor window. Click and hold the first icon called *Road segment* until a drop-down-menu opens. Select the Road segment type *Straight*.

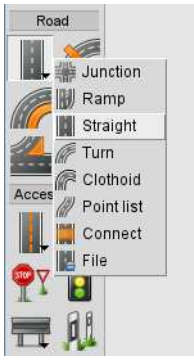


Figure 5.50: Inserting new road segments

Use the cursor to navigate into the design area of the Editor (the middle window) and click once to define the starting point of the straight segment. The first point that is set is also the origin of the global coordinate system. Hold shift and click again to create a horizontal, straight road segment that is 100m long.

The segment's length can be altered by entering the desired value in the field labelled *Length* in the tab on the right hand side while the corresponding segment is selected.

To select specific elements in the Scenario Editor, *Selection mode* is used.

Save the TestRun and open the 3D-Preview in IPGMovie: Click on the button with a play-symbol in the top panel in the Scenario Editor and select *3D Preview*.

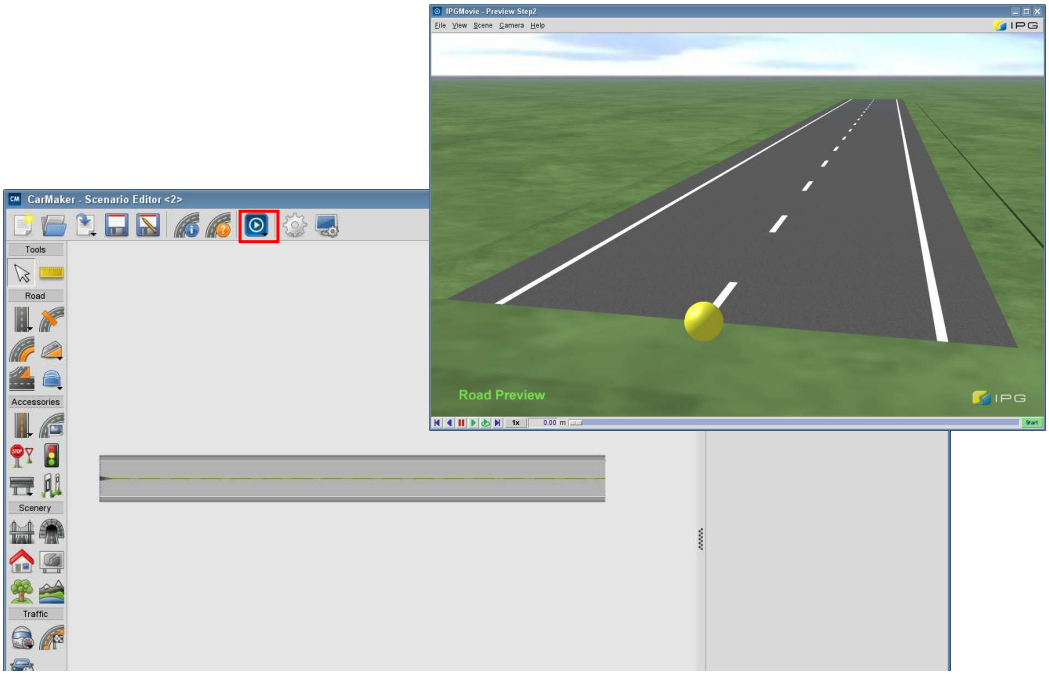


Figure 5.51: 2D-Preview in Scenario Editor and 3D-Preview in IPGMovie

The display in the Scenario Editor shows the geometry of the road in two dimensions. This viewing method may not be as detailed as the preview in IPGMovie, but it is very convenient for implementation and quick-checking during simulation preparation.

IPGMovie shows the road definition in a 3D animation with far more detail. By moving the slider in the control bar or by pressing the play button, the yellow marker in the road preview moves along the track, displaying the road and all its features. This feature is only available after a route has been defined. This will be explained in the course of this example.

At this point, the road is just a straight line with a constant coefficient of friction. The next section of this document will show that when changes are made in the road definition, they are immediately updated in the 3D-preview.

Building a Road Network

Now, two junctions and a few additional links will be added to the previously saved TestRun.

In the Road tab on the left side of the Scenario Editor, click and hold the first icon *Road segment* again.

1. This time, release the mouse button over the *Junction* option. Navigate to the end of the first Segment (Link 0, Segment 0, the straight) using the cursor and look for a small, orange circle that appears when the end of the reference line has been captured. Click once to define the straight as the first junction arm.

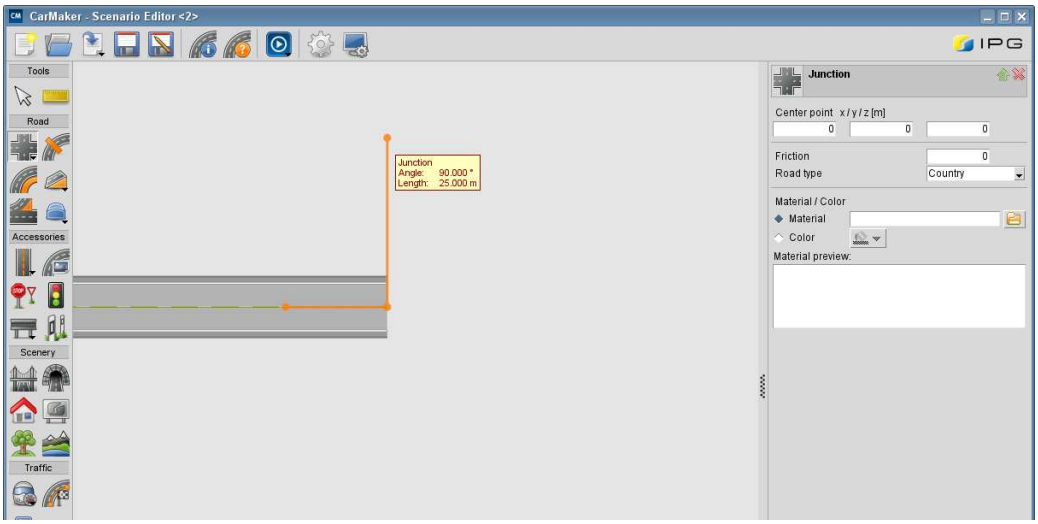


Figure 5.52: Defining the first Junction arm

2. Hold *Shift* and add three 25m long arms, each 90 degrees to another, to the junction and confirm by either using a double-click on the last junction arm or by clicking exactly in the middle of the junction.

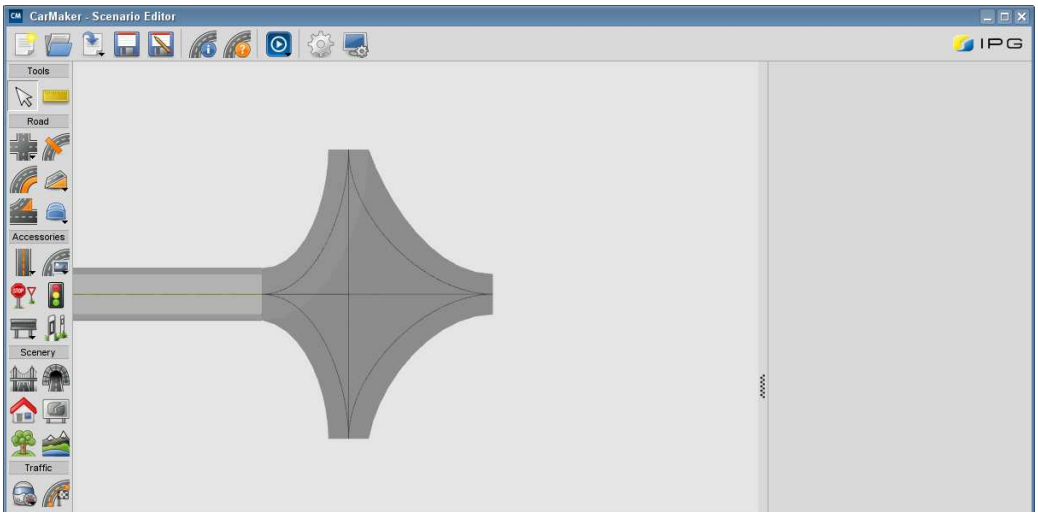


Figure 5.53: Finished Junction

3. Add a straight segment to each Junction arm. Be sure to capture the right point of the reference line when adding the new segment.

The segments that go upwards and to the right should be 200m long and the segment that goes downwards should be only 50m long.

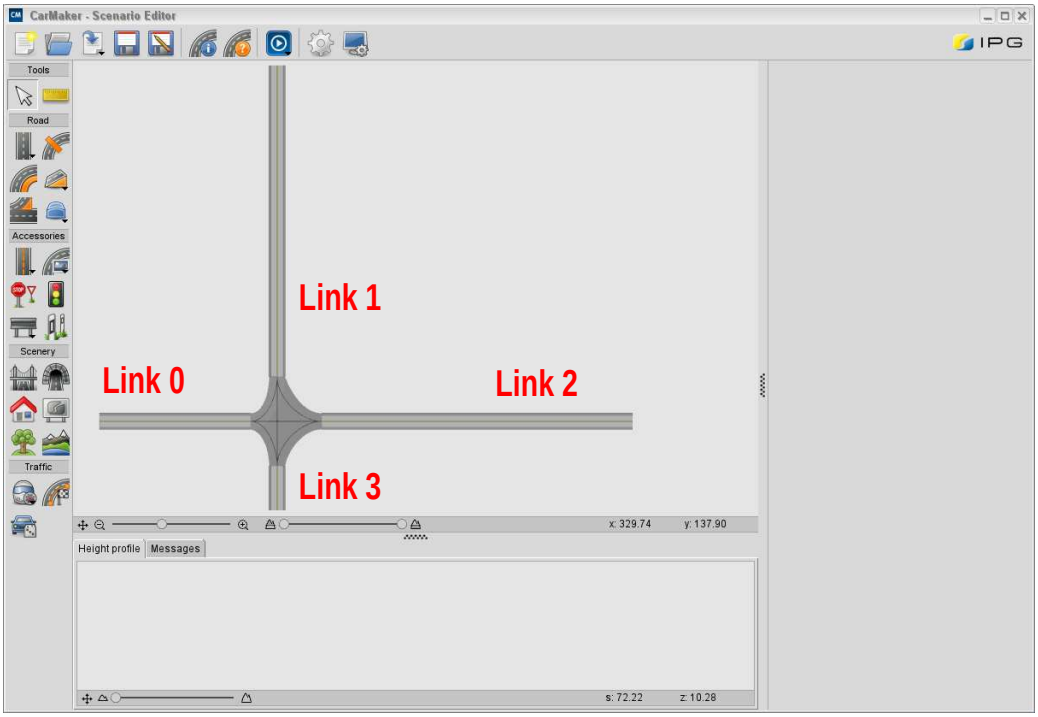


Figure 5.54: Junction with four Links

4. Add a second Junction at the right end of the road network: Select the road segment type Junction again in the Road tab on the left side and click once to capture the correct point at the end of Link 2.

Holding Shift, add one 25m long Junction arm that points to the top right at 40 degrees and one that points downwards at 270 degrees.

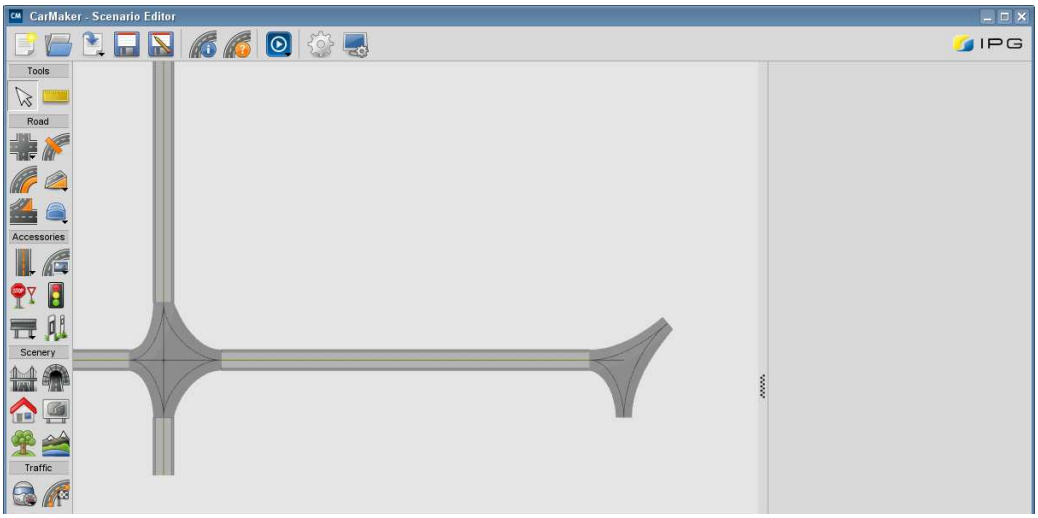


Figure 5.55: Second Junction with three arms

5. Add a long, curvy road to the top right-hand side of the road network using *Point list*: Click and hold the road segment icon again and select *Point list* in the dropdown-menu.

Make sure to capture the right point of the reference line at the Junction arm on the top right-hand side and place as many points as you like to create a Link broadly similar to the one in the figure below. Double click on the last point to end this mode.

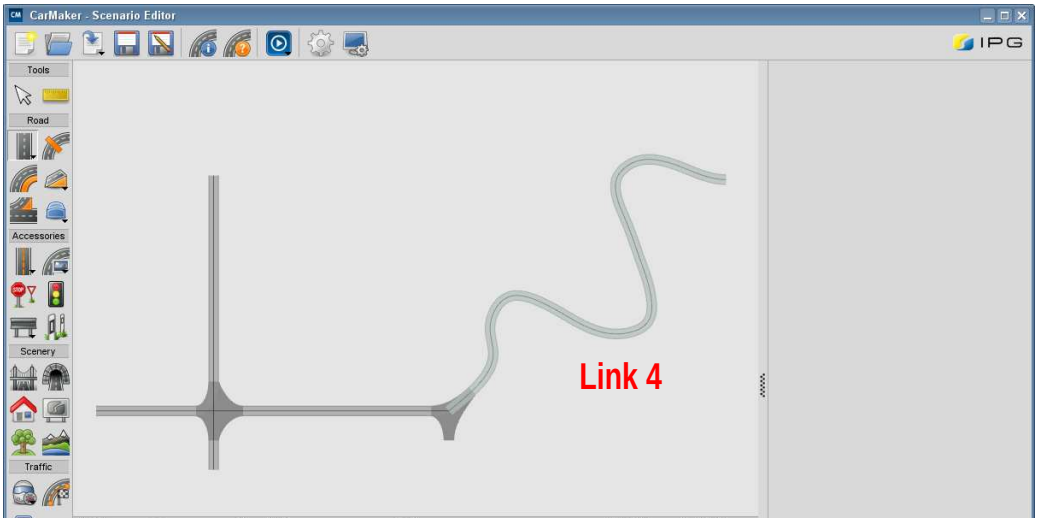


Figure 5.56: Point list at the right end of the road network

6. Add a Route to the road: In the toolbar on the left side of the Editor, in the "Traffic" tab, select the *Route* icon. Using the cursor, select the predefined Path on the right lane of the Link on which the vehicle will start. In this case, that means clicking the straight, horizontal Path on the bottom lane of Link 0 on the left side.

Now, hove the cursor over this Path again and left-click once again as soon as the line is marked with an orange, broken line.

Now, all of the following Path segments that are required for our Route need to be selected one after the other. First, the Path that goes straight across the first junction, then the right lane Path on Link 2, the Path that goes upwards over the junction and lastly the right lane Path on Link 4. This sequence of Paths defines our Route.

To set and complete the Route definition, double click onto the Route itself (or anywhere besides the road) until it changes color. In the tab on the right-hand side, name the Route "Woods".

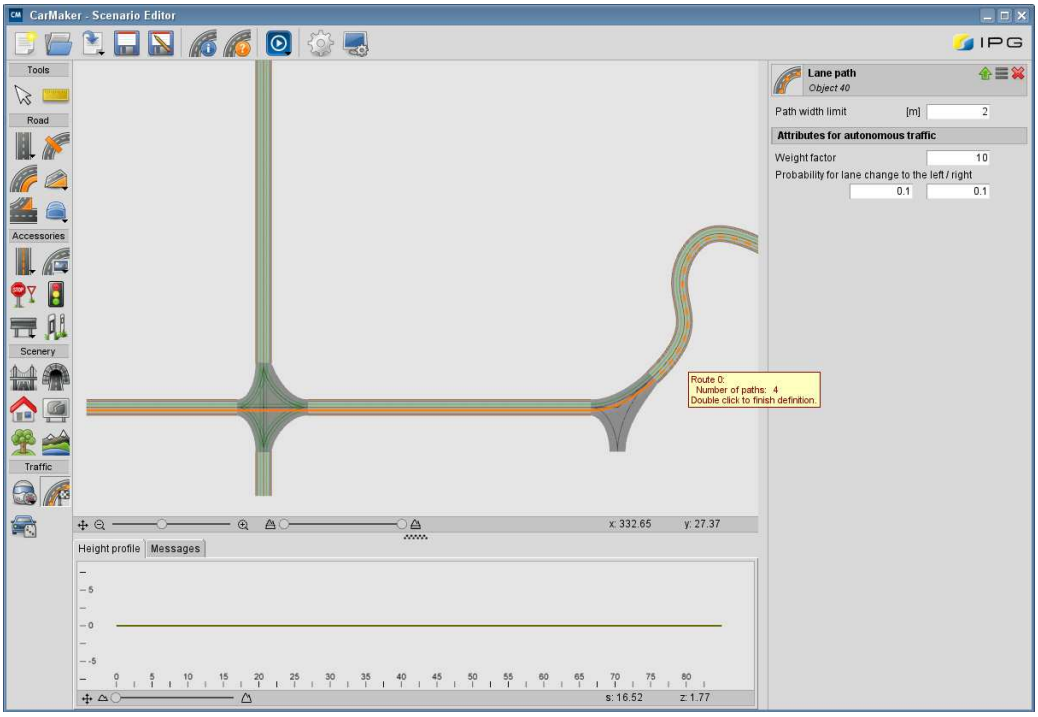


Figure 5.57: Defining the Route

Check the road geometry in 3D-Preview using IPGMovie by pressing the button marked with the play-symbol again.
Now the slider can be used to move along the Route in the preview.

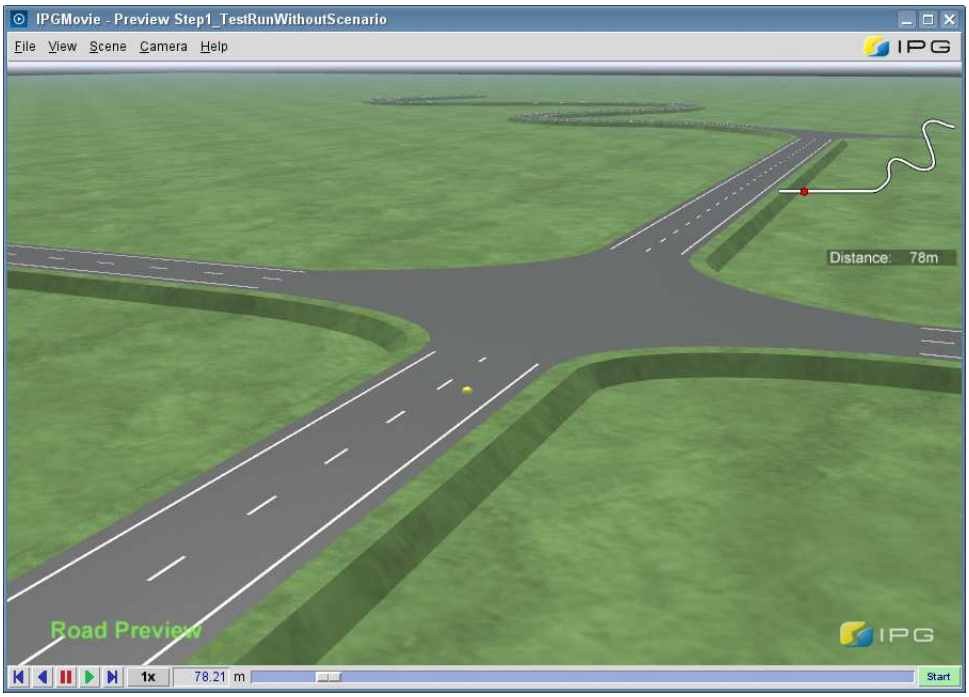


Figure 5.58: Checking the geometry using 3D-Preview

Start the TestRun: Click on the green “Start” button in the CarMaker main GUI.

Watching the simulation in IPGMovie, it is visible that the vehicle stops before it even reaches the end of the track. This example shows the dependency between the road and maneuver definition.

To make the vehicle drive to the end of the track, a maneuver needs to be defined that does not end before the track does. This means that the simulation is over as soon as either the road or maneuver definition has come to an end.

Edit the maneuver so that the vehicle reaches the end of the track: In the CarMaker GUI, click on *Parameters > Maneuver.*

In the field *Duration*, insert the following value: 9999.

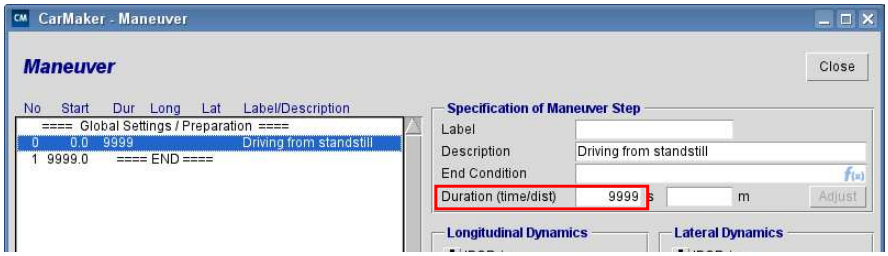


Figure 5.59: Changing the duration of the maneuver

Save the TestRun by clicking *File* > *Save* in the CarMaker main GUI and start the simulation again.

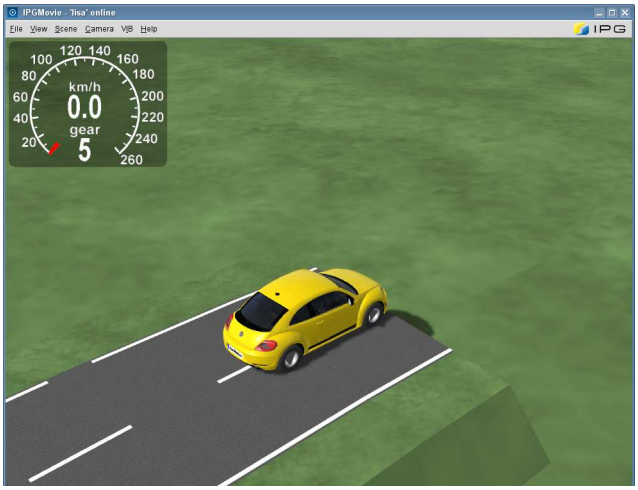


Figure 5.60: Vehicle at the end of the road

The vehicle now drives to the end of the road and the simulation is only terminated once the road has ended. The maneuver definition is explained in further detail in section 5.4.2 'Defining the Maneuver'.

Create a Closed Track with 3D Surface

The previously built road will now be extended in order to build a closed track. To load the previous TestRun *My_TestRun_ScenarioEditor*, click on *File* > *Open* > *My_TestRun_ScenarioEditor* > *OK* in the CarMaker main GUI. For better visualization, IPGMovie should be opened, too. In case needed, a solution to the steps up until now is available in the example TestRuns under *Product Examples* > *Examples* > *BasicFunctions* > *Road* > *_QuickStartGuide* > *Step2_Woods.* In case you are using the solution, be sure to save it as "My_TestRun_ScenarioEditor" for further use.

For this example, two new segments and an elevation and slope profile with the following geometries will be added to the previously built track. Following the instructions in the list below, the closed track that can be seen in figure section 5.61 'Course without elevation profile' will be built

- Link 3, Segment 1:
  - Subsequent to segment 0
  - Segment type: Turn
  - Radius: 100m
  - Angle: 180 deg
  - Slope profile with a height of 0.5 m on the outer corner
- Link 3, Segment 2:
  - Subsequent to segment 1
  - Joins the previous turn with the second junction
  - Segment type: Connect
- Elevation profile on Link 2 (Link between the two junctions)
  - Short section with a height of 8 m at the top

First, all segments are implemented in the plane with the correct length and curvature. The elevation profile is added to the track in the end.

Link 3, Segment 1: Click and hold the *Road Segment* icon in the *Road* tab again and select *Turn*. Click once at the end of the straight and holding shift, click again to create a left turn with a 100 m radius and 180 degree angle. If necessary, adjust the parameters *radius* and *angle* in the tab on the right side.
Note that when implementing a turn, the second click defines whether the turn goes to the right or to the left.

Link 3, Segment 2: Click and hold the *Road segment* icon and select *Connect*. Click once at the end of the turn and again at the bottom Junction arm of the second Junction. Be sure to capture the orange point in order to create correct transitions.

The track should now look like displayed in the figure below.

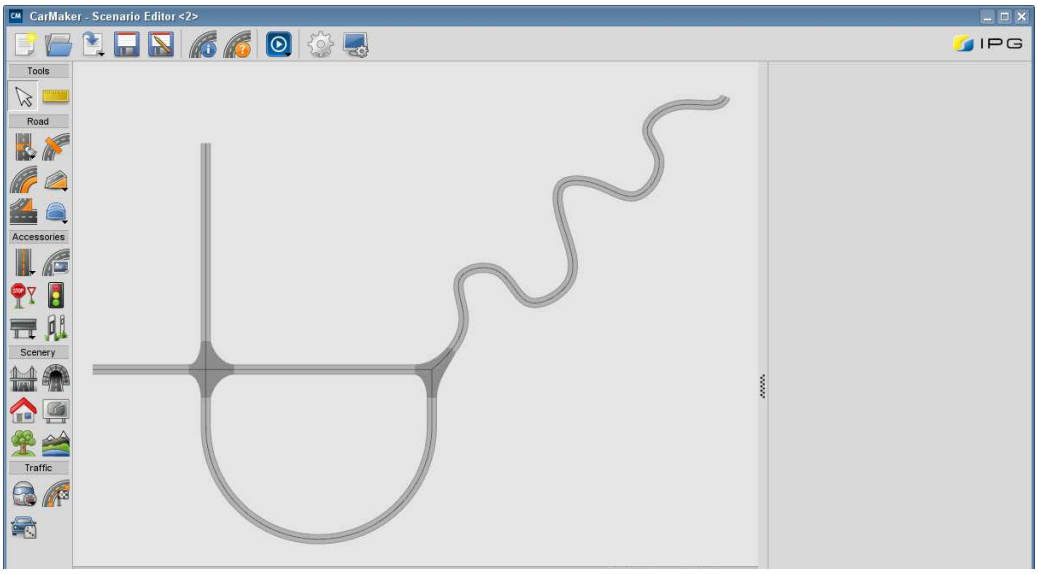


Figure 5.61: Course without elevation profile

Now, the different height profiles for specific links are implemented. It is important to know that each link can have only one height profile of each type (elevation, slope and camber). When adding further height node points to the same link, the user should expand the existing list instead of trying to add a new one.

Height profiles are defined relative to the s-coordinate which runs along the center of the road (reference line). There are two ways to define the beginning and end points (limits) for certain elevations:

- Precise s-coordinates: Beginning and end can be set exactly in the elevation profile, relative to the beginning of the corresponding Link.
- Intuition: Using the cursor the user can select points in the display window of the Scenario Editor to define plausible beginning and end points for an elevation profile.

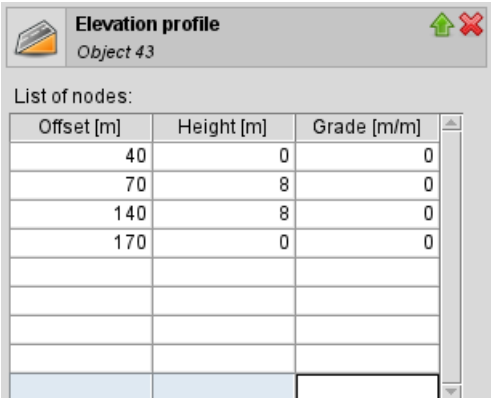
In this example the beginning and end points for the elevation profile can be set intuitively using the cursor in the display window. If desired, the values can be adjusted in the tab on the right-hand side to fit those in the solution later.

Add the elevation profile to Link 2: Click and hold the 3*D Surface* button in the *Road* tab. Choose *Elevation profile*. Click on the link and define the first point shortly after the beginning of Link 2. Select the second point shortly after the first one, the third closer to the end of the Link and the fourth right at the end. The selection is ended with a double-click on the last point.

Now a list representing the elevation profile values is visible in the right-hand tab. In case too many points were placed or some points are at the wrong place, this table can be used to delete or alter the values for each point.

Insert a height of 8m for the two middle points, 0m for the others. So that the spline for the elevation profile is calculated correctly and the rest of the course isn't deformed, select 0 for the grade in each point.

This is what the elevation profile should look similar to



Offset [m]	Height [m]	Grade [m/m]
40	0	0
70	8	0
140	8	0
170	0	0

Figure 5.62: Elevation profile for the longitudinal slope

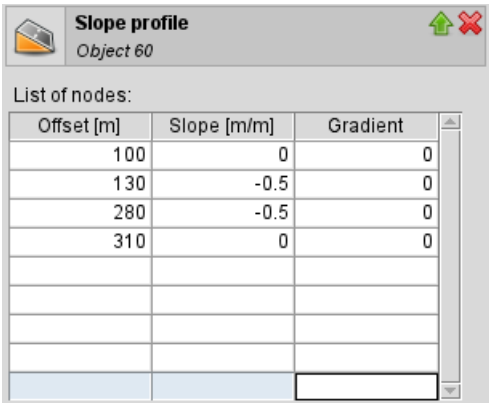
It doesn't matter if the points differ slightly since they were selected intuitively. Clicking on *Preview* shows what the road now looks like.

Now the lateral slope for Segment 1 on Link 3 can be implemented.

Similar to when defining the elevation profile for the longitudinal slope, click and hold the *3D Surface* button in the *Road* tab. Choose *Slope profile*, select the relevant Link (the long turn at the bottom) and define four points on segment 1 (the turn).
Point 1: Around the beginning of the curve. As off this point the slope will begin to gradually reach the desired value.
Point 2: A few meters behind the first point. As off this point the slope will be constant at the set value.
Point 3: Shortly before the end of the curve. As off this point the slope will begin to decline.
Point 4: End of the curve, shortly after point 3.

Now a list representing the slope profile is visible in the right-hand tab. Insert -0.5 for the slope of the second and third point, and 0 for the rest. Also, set 0 for the gradient of all four points. Whether the slope value needs to be positive or negative depends on which direction the slope should go in.

This is what the slope profile should look similar to:



Offset [m]	Slope [m/m]	Gradient
100	0	0
130	-0.5	0
280	-0.5	0
310	0	0

Figure 5.63: Slope profile for the lateral slope

Insert a second Route: Click the *Route* icon again and first select the right lane Path on Link 0 (the first one on the left) and confirm this Path after the orange, broken line appears. Accordingly, select the Path that goes across the Junction, the right lane Path on Link 2, the Path that goes downwards over the junction, the right lane Path on the long turn and then the Paths upwards to Link 1.

Double-click to end the Route definition and name this Route "TestTrack".

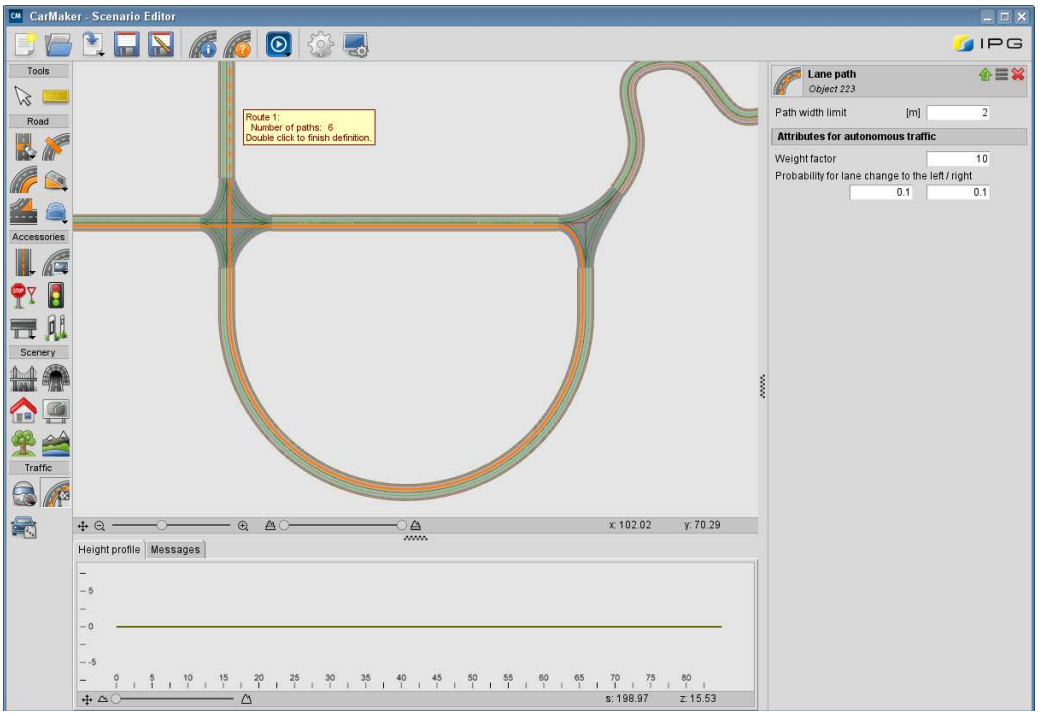


Figure 5.64: Defining route "TestTrack"

To select which route the driver should drive on is defined in the global settings. These can be accessed using the gear icon in the top bar of the Editor. A field labelled *Active Route* on the right side opens a dropdown-menu containing all available routes.

Select "TestTrack" as the active route, save the TestRun and start the simulation again. You can compare your TestRun to the example TestRun *Step3_TestTrack.*

In IPGMovie the road characteristics that were just defined are now more noticeable, especially the longitudinal and lateral slopes.

Figure 5.65: Longitudinal and lateral slopes

Modifying the Width and Coefficient of Friction

Up until now the designed roads have all had the coefficient of friction and road width that are predefined in the Scenario Editor. If desired, the user can easily adapt both parameters individually. How the user must proceed to do so will be described in this chapter.

In this example, again, the saved TestRun *My_TestRun_ScenarioEditor* is used. Alternatively, the solution to the previous steps can be found under *Product Examples* > *Examples* > *BasicFunctions* > *Road* > *_QuickStartGuide* > *Step3_TestTrack.* Remember to save as *My_TestRun_ScenarioEditor.*

When the width of a lane is set, this applies to the entire lane.

In order to define areas with individual lane widths, a new *Lane Section* must be defined (Lane section = section of the road with a constant number of lanes). To do so, *Lane section* in the *Road* tab needs to be selected and using the cursor, the limits for the new Lane Section can be defined in the display window by clicking on the road at the desired points. Now, the lane width within this section can be varied independent of the other Lane Section lane widths in the known manner. This isn't required in the following example.

Add an area to perform a braking test by changing the lane settings on Link 1 (the link going upwards from the first Junction):

First, delete the left lane of Link 1. Since we want to drive in the middle of the road for our braking test, the right lane is sufficient: Click on the Lane icon in the Road tab and select the left lane. Delete it by clicking the red "X" in the tab on the right side.

Change the lane width of Lane section 1: Click on the *Lane* icon in the *Road* tab and select the single lane of section 1. Set the lane width to 10m in both fields (*Width at start* and *Width at end*) in the tab on the right-hand side.

Adapt the Road Markings on the left hand side by selecting the *Road Marking* tool in the *Accessories* tab, clicking on the currently broken line and changing the *Line type* to *Single line*.

Start the simulation and watch how the track has been changed to be wider.

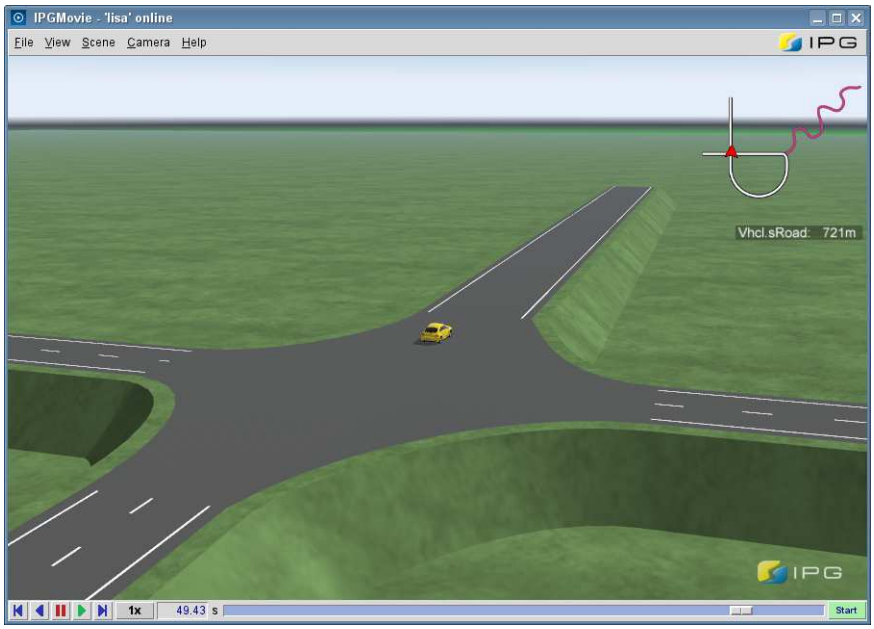


Figure 5.66: Vehicle dynamics platform

In addition to the lane width, the coefficient of friction can be user-defined using two methods:

- Change the coefficient of friction for the entire Link (globally): In selection mode, click on the road and select the desired Link. In the tab on the right side a value can be assigned to the parameter "Friction". This changes the coefficient of friction for the entire track.
- Change the coefficient of friction in sections: Click and hold the "Bumps" icon in the "Road" tab and select "Friction". This tool allows the user to place variable friction stripes. Click on the desired Link and define the two points that will limit the friction stripe. In the tab on the right you can now define the lateral offset, width and coefficient of friction for the friction stripe.

Add a friction stripe to the vehicle dynamics platform: as previously described, select the *Bumps* tool and then pick the *Friction* option. Click on Link 1 (the Link we just adapted) and place two points to define beginning and end of the friction stripe. Alter these points in the tab on the right hand side so that the stripe starts at 120m and ends at 200m.

Also in the tab on the right, set the Lateral offset to -10m from the Reference Line (negative value is the right side of the road), the Width to 10m and the Friction to 0.5.

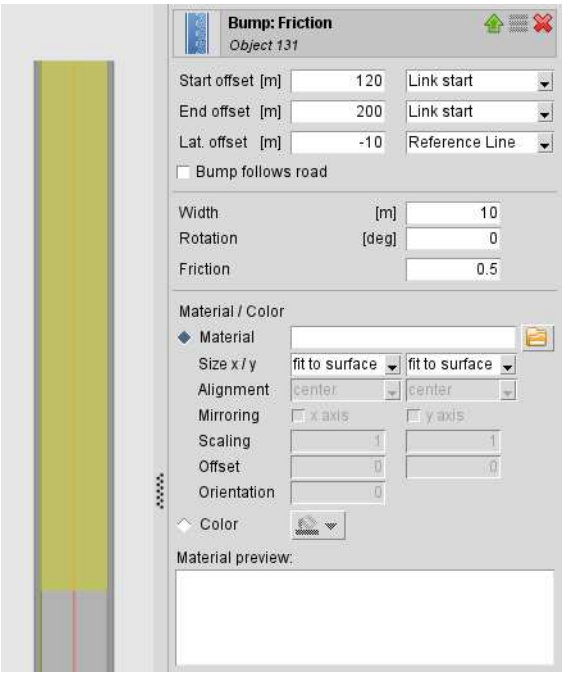


Figure 5.67: Friction stripe parameters

The friction stripe as it is defined in the figure above creates the following visualization in IPGMovie:

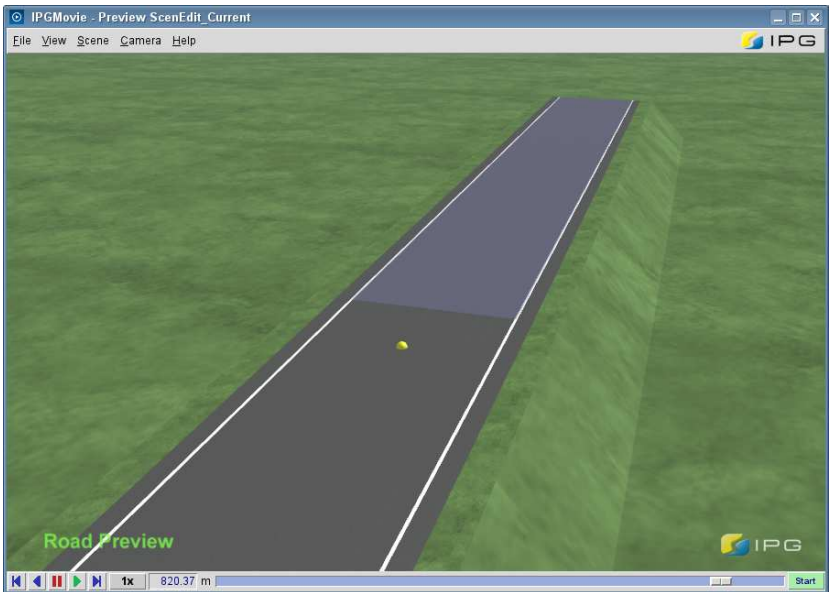


Figure 5.68: Defined friction stripe

Inserting Bumps, Markers and Movie Objects

Changing the road's track width and friction are only two of the numerous possibilities that IPGRoad offers. In the tab on the left, there are many options that allow the user to add special items to the road and surrounding scenery.

Figure 5.69: Special items available in Scenario Editor

The selectable features are grouped in *Road*, *Accessories* and *Scenery*:

- Road - Bumps: With this tool the user can choose from beams, cones, friction stripes, mesh, waves and lateral profiles. These change the road's 3D appearance and alter the surface.
- Accessories: Road markings, road paintings, traffic signs, traffic lights, traffic barriers and guide posts can be implemented in this tab. Some of these components may have an influence on the driver and vehicle behavior, e.g. the vehicle can be defined to stop at a stop sign using traffic sign sensors.
- Scenery: These objects do not directly influence the simulation, as they are design elements. The user can implement bridges, tunnels, geometry objects, sign plates, tree strips and terrain.

For more information regarding the individual parameters and their elements, please refer to the Scenario Editor section in the User's Guide.

Figure 5.70: A Road with activated Markers and Movie Objects

Save the TestRun and the road file. Save the road file in the Scenario editor. This will save a file with a .rd5 ending. Call it *My_Road.rd5*.

Compare TestRuns:

Compare your TestRun *My_TestRun_ScenarioEditor* with the predefined TestRun *Step4_FrictionStripe* (located in *Product Examples* > *Examples* > *BasicFunctions* > *Road* > *_QuickStartGuide*) or have a look at the TestRun *Final_Road* in the same folder to see how various geometry objects and environment settings can change the road network and surroundings.

After having built a road, it is now time to look at the Maneuver definition in further detail.

5.4.2 Defining the Maneuver

For the examples in this chapter the previously built TestRun *My_TestRun_ScenarioEditor* or one of the solutions *Step4_FrictionStripe* or *Final_Road* (including environment changes) is required.

Save the TestRun *My_TestRun_ScenarioEditor*. Alternatively, use one of the predefined TestRuns *Step4_FrictionStripe* or *Final_Road* and save it as *My_TestRun_ScenarioEditor* so that your TestRun is up to the current status.

Based on this TestRun, the maneuver definition will be explained and extended.

The Maneuver GUI

In the CarMaker main GUI, click on *Parameters* > *Maneuvers*.

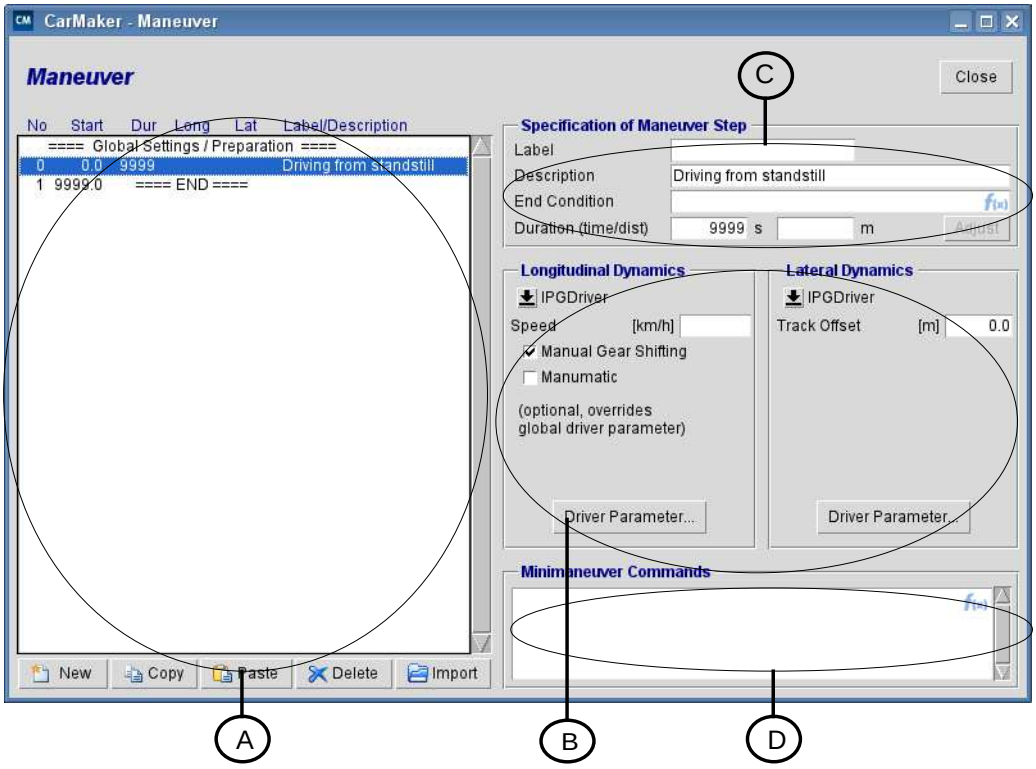


Figure 5.71: The Maneuver GUI in CarMaker

The figure above shows the Maneuver GUI:

- Field (A): In this section a list of all the maneuver steps, that will be carried out consecutively, is shown. When one step is selected, more information regarding it's parameters becomes visible on the right-hand side.
- Field (B): In this section the longitudinal and lateral dynamics are specified. These define how the vehicle is controlled. The longitudinal and lateral control of the vehicle are defined completely independent of each other.
- Field (C): Here the user can set a description for each maneuver step and define end conditions. The end condition can be a time or distance that, when reached, terminates the simulation.
- Field (D): The "Minimaneuver Commands" field is a feature for experienced CarMaker users. Advanced maneuver definitions can be individually programmed and integrated ion this field. As this field is optional, it will remain empty in this example.

Limiting a Maneuver Step with DrivMan Jump

Up until now, the maneuver step in the TestRun has always been the same: a single maneuver step, driving from standstill, 9999s long.

In this chapter the maneuver step duration is to be limited not by time, but rather by a marker that will redirect the driver to another maneuver. This means that as soon as the vehicle passes said marker, a new maneuver step is initiated and carried out until it's end-condition is fulfilled.

In the following example, a marker is placed shortly before the braking area. When the vehicle reaches this marker, a braking maneuver will be initiated so that the driver begins to brake while coming onto the slippery friction stripe. The field called "Target label" defines the maneuver step to which the driver will jump after passing the marker.

Set a Marker on the braking area: On the left-hand side of the Editor, under the tab labelled *Traffic*, click and hold the *Marker* icon. In the dropdown-menu, select the option called *DrivMan Jump*, click on the vehicle dynamics platform (Link 1) and click again to place the marker.
In the right-hand tab, edit the offset to 110m and the target label to *Braking* (this refers to the second step in the list of maneuver steps).

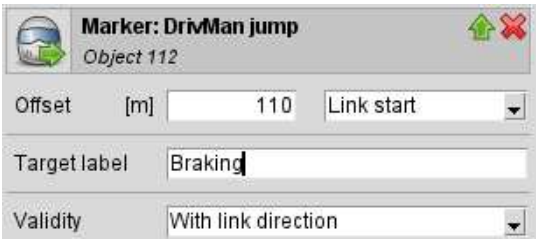


Figure 5.72: Marker settings

Now a marker has been set 10m in front of the friction stripe on the vehicle dynamics platform that refers to the maneuver step 1. Currently only one maneuver step (step 0) is defined. So if the simulation is started now, it will be aborted as soon as the driver reaches the marker because he will be referred to a non-existing maneuver.

The second maneuver step "Braking" needs to be defined.

Define the second maneuver: In the CarMaker main GUI, open *Parameters > Maneuver* and highlight the last line in the list (*END*) by clicking on it. Now select *New* to insert a second maneuver step under the first one.

Write "Braking" in the *Description*, as well as the *Label* and change the longitudinal dynamics to *manual* by clicking on the small arrow in the corresponding field.

In order for the driver to suddenly brake, change the values of both the clutch and brake to 1. This means that after the marker, the driver will quickly press and hold both pedals until standstill.

Change the *Duration* to 999s to make sure the driver reaches a standstill before the maneuver ends.

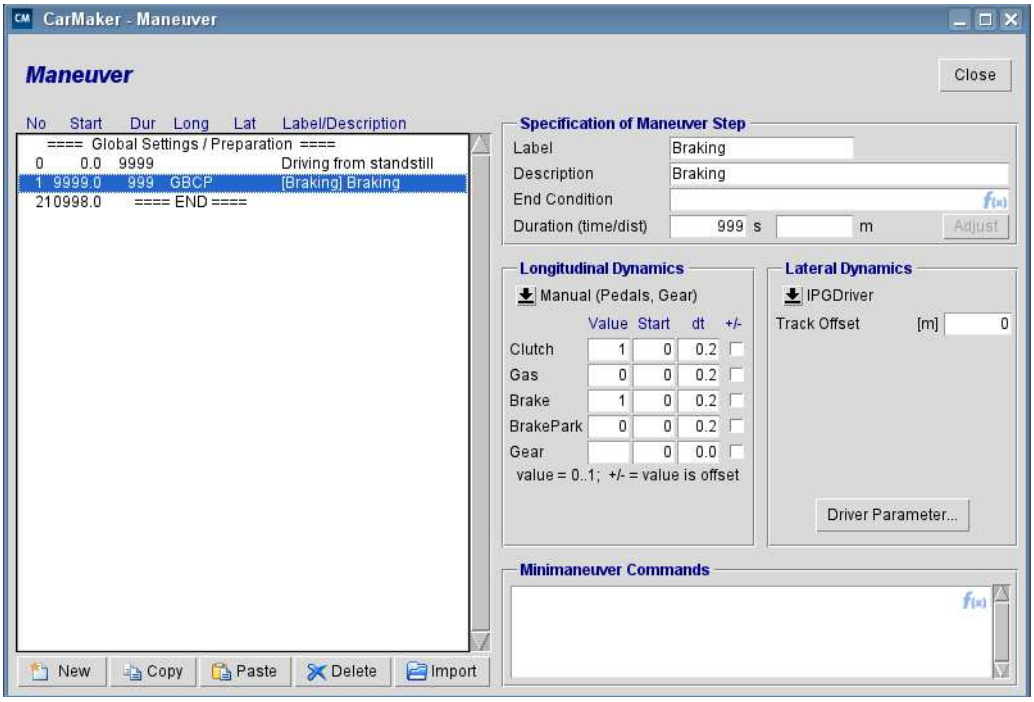


Figure 5.73: Defining the braking maneuver

Make sure that "TestTrack" is the active Route, save the TestRun and simulate.

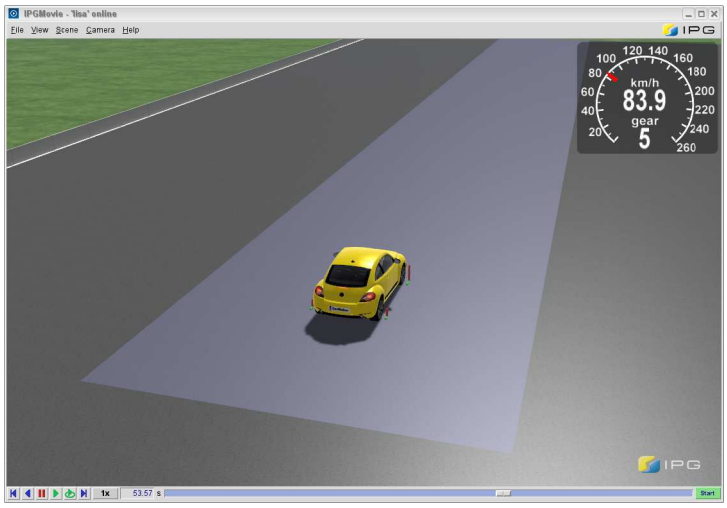


Figure 5.74: The new maneuver step displayed in IPGMovie

Now the vehicle stops after it has reached the Marker and the driver started braking. Due to the slippery friction stripe, the braking distance is higher than it would usually be. Try varying the coefficient of friction on the friction stripe to see changes in the simulation.

Adding an Open Loop Maneuver Step

By inserting an open loop maneuver, several maneuver steps can be used during the same simulation.

In this example, the dynamic test platform will be used to perform a sinus steering maneuver.

Insert an new maneuver step: In the Maneuver GUI, click on the last maneuver step (“END”) and then click on “New“ at the bottom of the list.

Select and define the new maneuver step:
Label: Sinus steering
Description: Sinus steering
Duration: 3s
Longitudinal Dynamics: IPGDriver
Lateral Dynamics: Sinus, Amplitude = 50 deg, Period = 1s, # Periods = 3

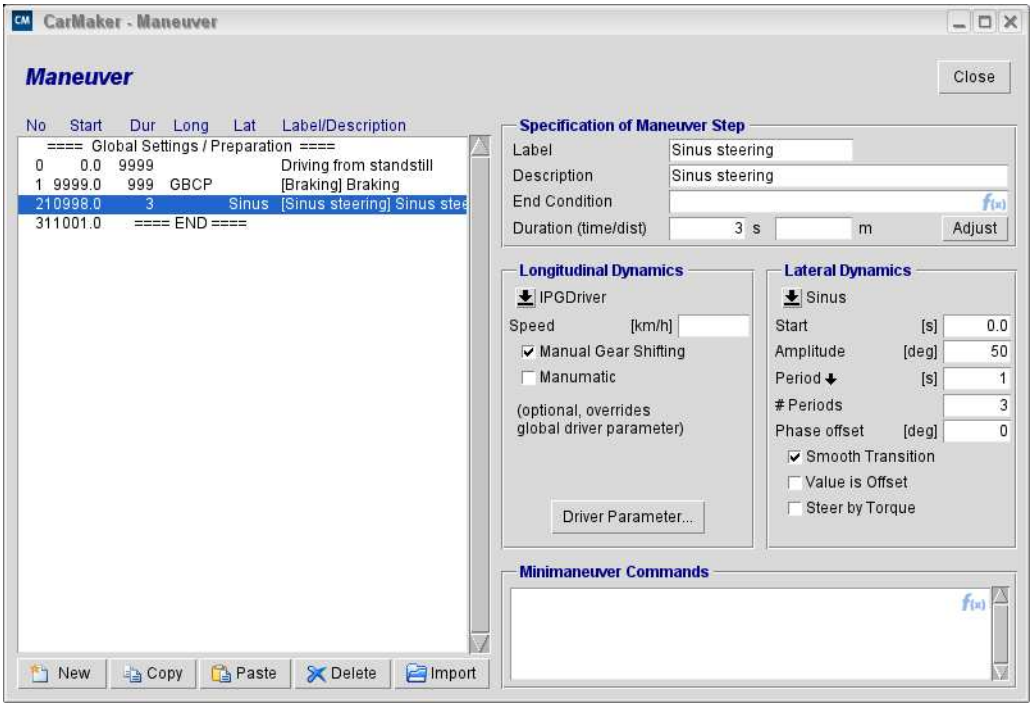


Figure 5.75: Open loop maneuver step

In the Scenario Editor, select the Marker and insert "Sinus steering" for the *Target label* so the driver jumps to maneuver step 2, instead of 1.

Save the TestRun.

In the CarMaker main GUI, set simulation speed to *realtime*.

Save and start the simulation, watch in IPGMovie and pay attention to the box *Maneuver* in the CarMaker main GUI.

In the *Maneuver* box of the CarMaker GUI, all maneuver steps are displayed. During the first part of the overall maneuver, which lasts up to the marker, the first maneuver step is highlighted. As soon as the sinus steering phase starts, the third maneuver step is highlighted.

This very simple example shows how to insert an open loop maneuver and then check which maneuver step is currently running.

Adding a Closed Loop Maneuver Step with IPGDriver

The previous simulations were all open loop maneuvers since at some point the vehicle came to a standstill or reached the end of the road. During a closed loop maneuver IPGDriver continues driving an unlimited amount of laps of a specific Route. Note that the Route needs to be a circuit so that a closed loop maneuver is even possible.

IPGDriver is a model developed by IPG that acts and reacts exactly as a real driver would. The virtual driver is composed of a controller that urges him to follow the course and a speed controller that regulates the velocity.

In the Scenario Editor, define a new Route by clicking the "Route" icon. Select the predefined Path on the right lane on the first Link (Link 0) and confirm by clicking again. Now select the Path that goes across the junction and accordingly, select all Paths necessary to drive around the long turn. The last Path segment to be added to the Route should be the Path that takes a right turn over the first junction. Double click to confirm the Route. Name it "ClosedLoop".

IPGDriver will now recognize that the last point of the Route lays on another point of the same Route whereas he will continue to drive this Route endlessly.

Save the TestRun, open the Instruments and start the simulation.

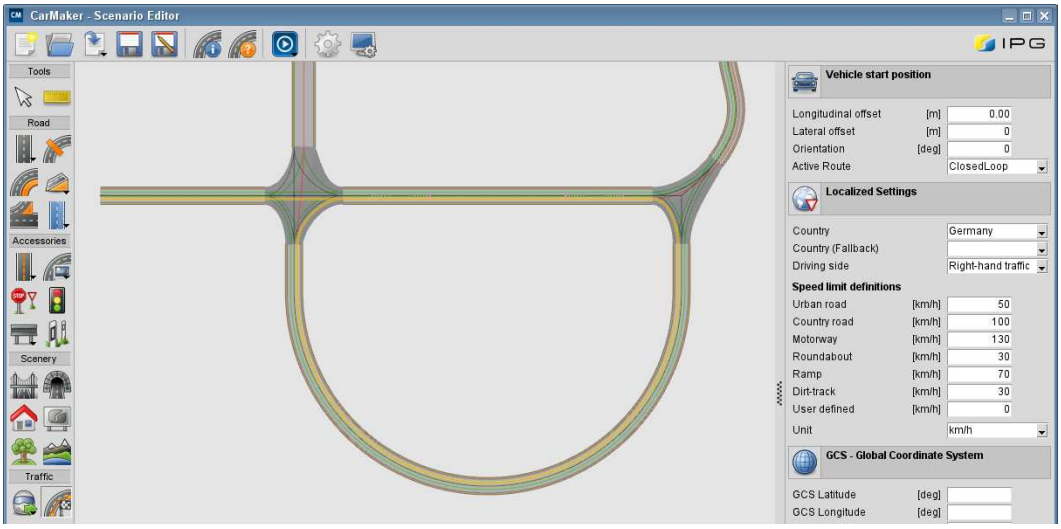


Figure 5.76: Closed Loop Route

The vehicle now continues driving. Observe the speed using Instruments. In the maneuver GUI a target speed can be set for each maneuver step individually. The driver will try to reach and keep this speed, all while considering the vehicles physical limits and aiming to stay on track. Try entering a target speed and watch how the driver handles these guidelines using *Instruments*.

This shows two things:

- The maximum speed that the driver is to aim for can be defined separately for each maneuver step.
- The driver stays on the track and identifies the ideal trajectory automatically.

Other Options for the Definition of a Maneuver Step

So far, only IPGDriver and a very simple sinus steering have been used in the maneuver steps. These are among the most important options used in CarMaker. However, there are various other options available.

Duration of the Maneuver

Figure 5.77: Defining the duration of a Maneuver Step

Previously it was discussed how a maneuver step can be restricted by a marker that forced the initiation of a new maneuver. This is by far not the only possibility for implementing end conditions.

In the field labelled *End Condition*, other parameters can be defined to abort the maneuver step. By right-clicking in the entry field, a list of all User Accessible Quantities in CarMaker is presented. Any of these can be chosen to be combined with a logical operator in order to become an end condition. Logical operators can be *smaller as (<=)*, *larger than (>=)* or *same as (==)*. Furthermore, it is possible to define more than one condition at the same time. Quantities can be combined using logical *AND (&&)* or *OR (||)* operators.

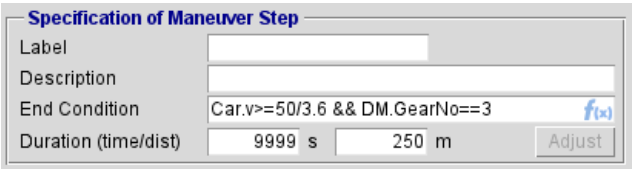


Figure 5.78: Defining the “End Condition” Parameter

Longitudinal Dynamics

Figure 5.79: Defining the longitudinal dynamics

To control the longitudinal dynamics, e.g. the speed and/or longitudinal acceleration of the vehicle, several options are available:

- IPGDriver: Fully automated control of the vehicle speed and acceleration by the IPG driver model.
- Speed Profile: This option enables the use of speed measurements.
- Speed Control: This option represents a simple speed controller.
- Manual: Enables direct control of the pedal positions.
- Stop Vehicle: Stops the vehicle. This option is especially designed for engineers working on active park assistance systems.

- Drive Backwards: Same option as IPGDriver, but backwards.
- IPGDriver + User Driver: Activates a user-defined driver model.

Lateral Dynamics

Figure 5.80: Defining the lateral dynamics

The vehicle's direction can be defined using several options:

- IPGDriver: Fully automated control of the steering wheel.
- Sinus: Sinus input on the steering wheel.
- Sinus Sweep: Sinus input on the steering wheel with different amplitudes (specifically a start and end amplitude that can have a linear or non-linear course).
- Steer Step: Steer step input on the steering wheel.
- Follow course: This option is a steering wheel controller like IPGDriver, but simplified.

The TestRun altered in this chapter can now be compared to the predefined TestRun *Final_Maneuver* (located under *Product Examples* > *Examples* > *BasicFunctions* > *Road* > *_QuickStartGuide*).

Chapter 6

Vehicle Model Parameterization

6.1 What is a Data Set?

Instead of using the term Vehicle Model, CarMaker tends to refer to *Data Sets*. This is because technically, CarMaker possesses only ONE vehicle model. This model has defined and fixed masses, DOF, spring/damper elements, etc. What happens then in CarMaker is that the model is parameterized with various Data Sets according to the user's needs. The Data Set contains the values necessary to parameterize the model, for e.g. the weight of the masses, the stiffness of the springs, etc.

In CarMaker, the Vehicle Data Set can be accessed via the CarMaker main GUI by either clicking directly on the field labelled *Car* in the main GUI, by clicking on *Parameters* > *Car* or by using the shortcut *Ctrl* + *f.*

6.2 Creating a New Vehicle Data Set

Apart from the large variety of predefined vehicle models available for use in CarMaker, the user often needs his own, individual model. There are several ways to generate a Vehicle Data Set from scratch that fulfills the user's exact needs. CarMaker offers a few tools that support the user in building his Vehicle Data Set that can also be used in combination.

In general, there are three approaches to parameterizing a Vehicle Data set from scratch:

- Using the *Vehicle Data Set Generator* to generate a Data Set based on a few basic specifications like vehicle class.
- Using the Vehicle Data Set Generator to create a basic Data Set and then specifying relevant components in further detail manually.
- Configuring an entire Data Set manually with the help of a *request form*.

Furthermore, CarMaker offers the option to *import* existing vehicle data for single vehicle components.

6.2.1 Using the Vehicle Data Set Generator

Open a new Testrun and open the Vehicle Data Set. Click on *File > Generator*.

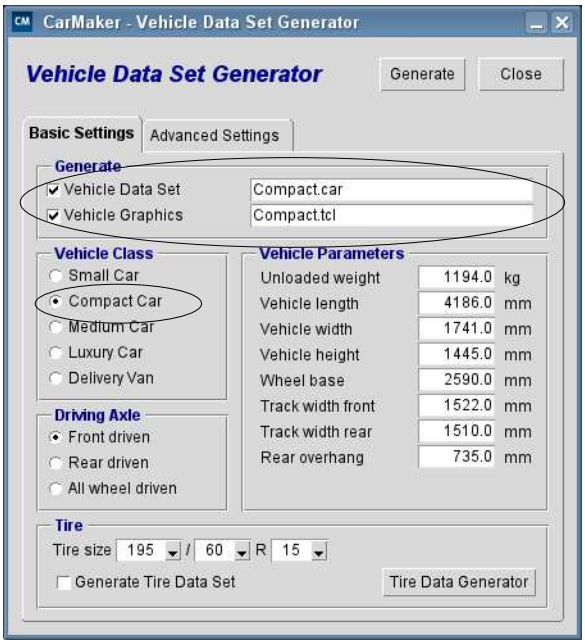


Figure 6.1: Vehicle Data Set Generator

The Vehicle Data Set Generator is a tool that automatically generates a plausible vehicle data set, based on the most important data necessary to characterize a vehicle.

Furthermore, the user can directly select a vehicle class, for e.g. *Compact Car* in order to assign the typical values for this kind of car.

Select the Vehicle Class "Compact Car".
In "Vehicle Data Set", write the name of the vehicle: "My_Car" and click on "Generate".
Close the Generator.

Now, the values written in the Vehicle editor have changed. Using the Vehicle Generator has implemented a new vehicle data set which can now be worked with.

In the Vehicle editor, click *File > Save* to save your vehicle.

From now on, the new values are not only displayed in the editor, but also saved to the corresponding file.

6.2.2 Using the Import Feature

The import function allows the user to replace certain vehicle components with those of another data set. In the Vehicle Data Set via *File* > *Import* the dialog displayed in the figure below opens up.

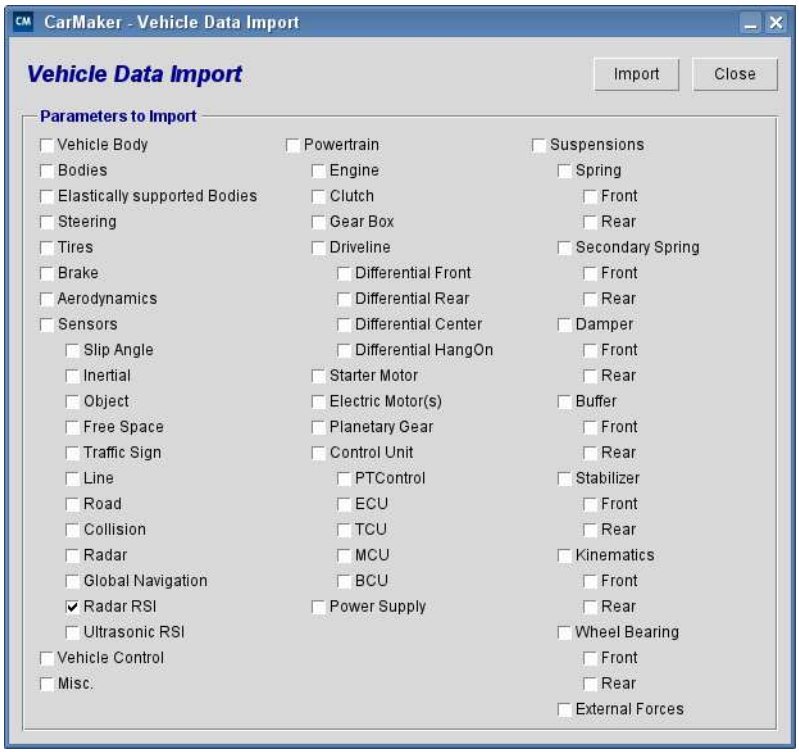


Figure 6.2: Vehicle Data Import

By activating the separate boxes the user can choose which specific components of the current vehicle model should be replaced by those of another vehicle. After clicking *Import* a browser opens where the vehicle from which the information is to be imported can be selected.

6.3 Vehicle Submodels

This chapter describes the parameters of a vehicle data set that need to be specified. The following sub-sections refer to the various tabs of the Vehicle editor.

6.3.1 Vehicle Body

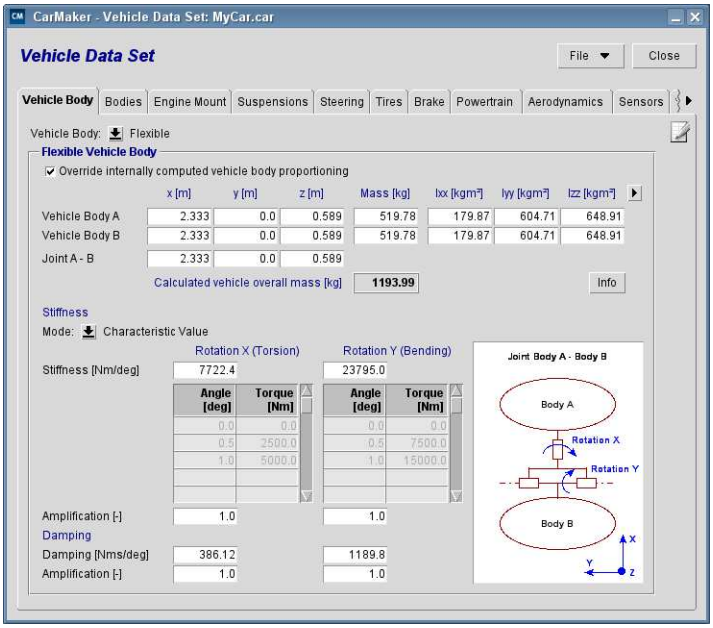


Figure 6.3: Vehicle body

In this tab the user can choose which vehicle model is adequate depending on the *Vehicle Body*:

- Flexible body
- Rigid body: Simple vehicle model without flexibility

According to the model that has been chosen, various parameters must be defined for each body (A: front; B: rear):

- Position of the mass(es)
- Weight and moments of inertia of the mass(es)
- Optionally: Stiffness of the vehicle body joint, linear or non-linear
- Optionally: Damping of the vehicle body joint

The masses are defined in the so called *design position*. This is a fictitious state where no forces and weights are applied.

6.3.2 Bodies

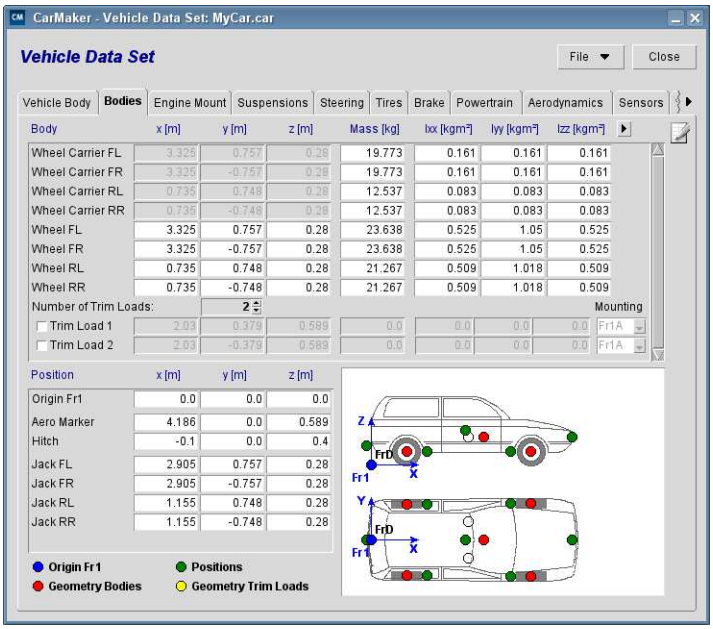


Figure 6.4: Body definition

In this tab, all other masses including their center of gravity and inertias are defined. Again, the definition takes place in the design position, according to the vehicle model chosen in the *Vehicle Body* tab.

- Engine: Can be placed either on body A, body B, or both bodies (engine masses are optional since the engine mass can be merged with the body mass).
- Wheels: Unsprung spinning masses.
- Wheel carriers: Unsprung masses that do not rotate.
- Trim Loads: Additional loads that can be fixed to body A or B. These are available by clicking on the arrow as shown on the following picture:

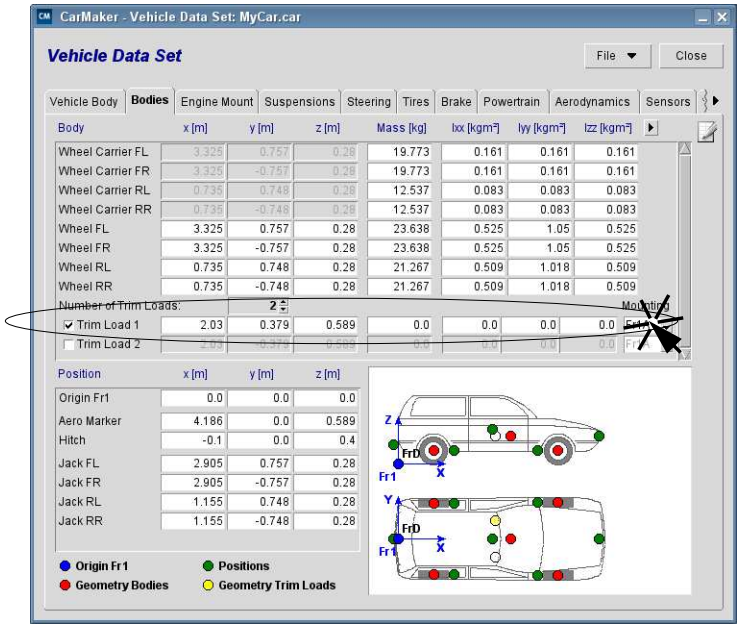


Figure 6.5: Trim Load definition

- Other fields:
  - Origin F1: Allows the user to move the origin of the default coordinate system of the vehicle.
  - Aero Marker: Point on the vehicle where aerodynamic forces are applied. These can come from all directions.
  - Hitch: Position of the hitch if a trailer is being used.

The exact description of these fields, as well as the *Jack* definition, can be found in the User's Guide.

6.3.3 Engine Mount

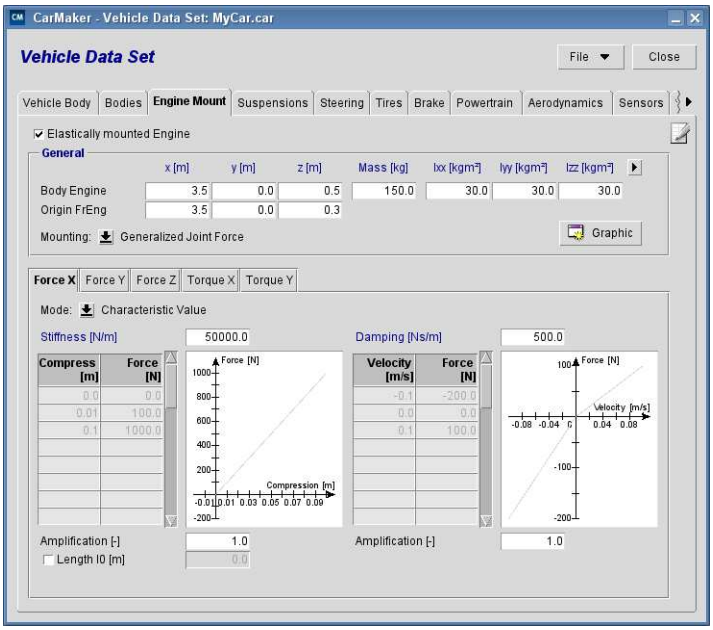


Figure 6.6: Engine definition

The CarMaker multi-body model also includes a separate engine body that is elastically mounted on the main body. Once activated, the position and engine mass need to be defined, as well as the location of the joint (or joints) between engine and main body. The joint consists of a spring damper element which can be characterized in the lower part of the dialog.

6.3.4 Suspensions

Spring

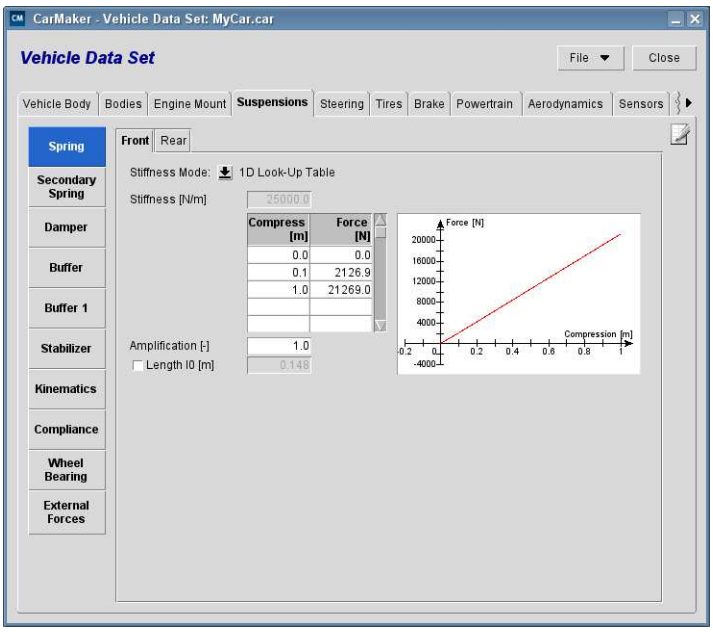


Figure 6.7: Spring definition

In this tab the user can independently define the spring stiffness and free length of the front and rear axle.

Secondary Spring

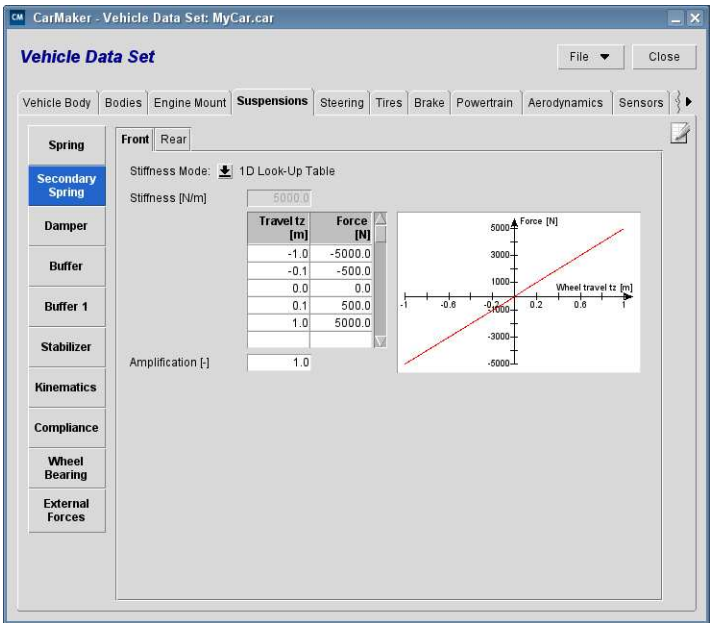


Figure 6.8: Secondary Spring definition

The Secondary Spring tab allows the user to define a spring force that describes the force generated by the suspension bushing torsion.

Damper

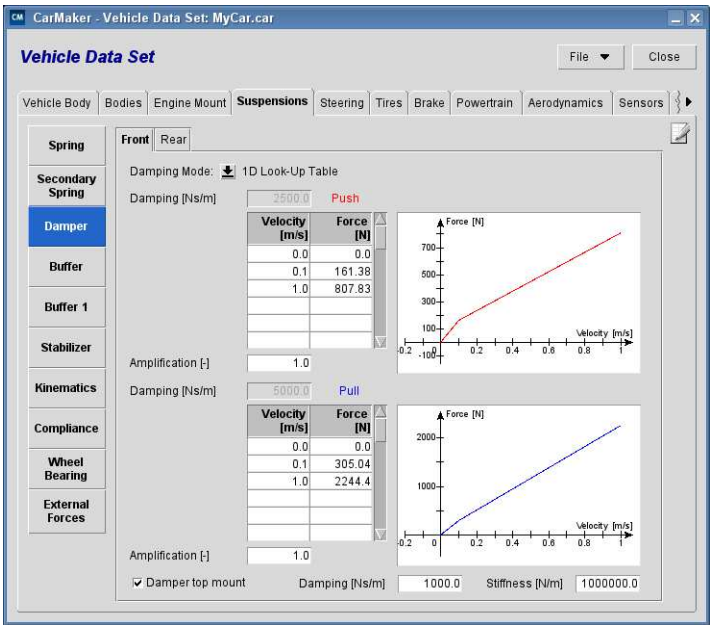


Figure 6.9: Damper definition

Here the damper characteristics can be defined independently for the front and rear axle. Push and pull domain are separately parameterized.

Buffer

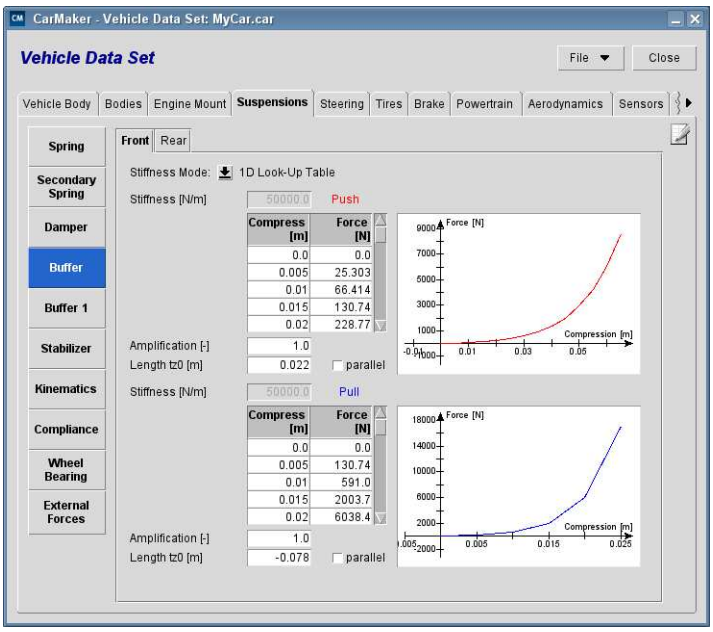


Figure 6.10: Buffer definition

The buffer limits the wheel travel and is defined using the parameter *length tz0*.

If the wheel travel reaches the given distance (tz0), the buffer acts like an additional spring with the parameters set in the tables.

Buffer 1

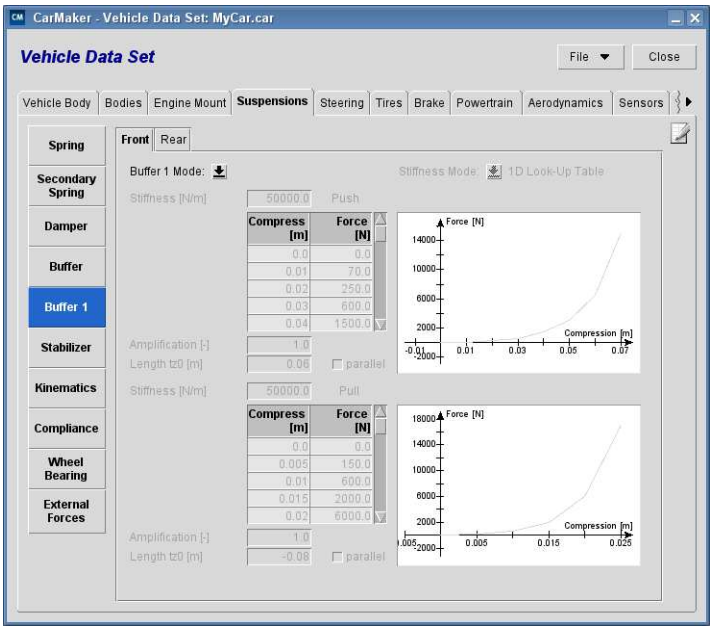


Figure 6.11: Buffer 1 definition

Optionally, an additional buffer can be added within either a parallel or series connection. To parameterize this second buffer, the same parameters are used as for the primary buffer. Internally, both buffers are merged to one with the same characteristics.

Stabilizer

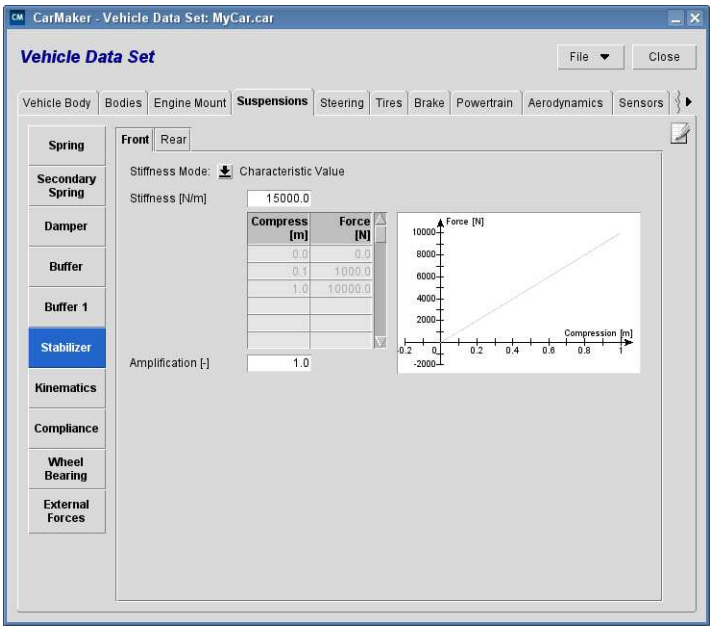


Figure 6.12: Stabilizer definition

The stiffness of the stabilizer is entered in N/m (approximation of virtual spring on small displacements). Optionally, you can parameterize it in Nm/rad.

Kinematics

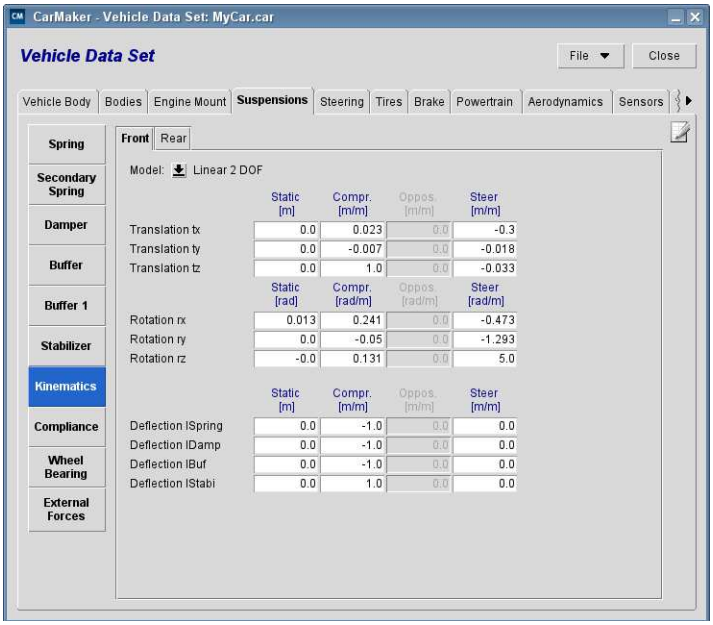


Figure 6.13: Kinematics definition

CarMaker offers a kinematics model where it is not necessary to parameterize the geometry of the vehicle axle. For CarMaker, the only important information is the positions of the wheels according to the wheel travel, steering rack displacement (for a front axle only) and the movement of the opposite wheel (in case of a rigid rear axle).

In the *Model* option of the kinematics tab, following kinematic characteristics can be chosen:

Mode	Description	Axle
Linear 1 DOF	Linear uni-dimensional kinematics. The kinematics only depend on the wheel travel.	rear
Linear 2 DOF	Linear two-dimensional kinematics. The kinematics depend on the wheel travel and the steering rack displacement. This mode can be used to parameterize a steered front axle for an independent wheel suspension.	front
	Linear two-dimensional kinematics. The kinematics depend on the wheel travel and the movement of the opposite wheel. This mode can be used to parameterize a rigid or semi rigid rear axle.	rear
Linear 3 DOF	Linear three-dimensional kinematics. The kinematics depend on the wheel travel, the steering rack displacement and the movement of the opposite wheel. This mode can be used to parameterize a steered rigid axle or semi rigid front axle.	front or rear

Mode	Description	Axle
External SKC file	Non-linear kinematics. The kinematics are parameterized by the kinematics tables written to an external file. In this case the values of each variable are defined for numerous steps of the wheel travel (e.g. from max. to min. bouncing, at step size 5mm) and optionally by the steering rack displacement or the wheel travel of the opposite wheel.	front or rear
External MBS File	Non-linear kinematics. The kinematics are modeled by a multi-body system.	front or rear

The values *tx/ty/tz* and *rx/ry/rz* define the translations and rotations of the wheel in the three directions:

Model: Linear 3 DOF

	Static [m]	Compr. [m/m]	Oppos. [m/m]	Steer [m/m]
Translation tx	0.0	0.023	0.0	-0.3
Translation ty	0.0	-0.007	0.0	-0.018
Translation tz	0.0	1.0	0.0	-0.033
	Static [rad]	**Compr. [rad/m]**	**Oppos. [rad/m]**	**Steer [rad/m]**
Rotation rx	0.013	0.241	0.0	-0.473
Rotation ry	0.0	-0.05	0.0	-1.293
Rotation rz	-0.0	0.131	0.0	5.0
	Static [m]	**Compr. [m/m]**	**Oppos. [m/m]**	**Steer [m/m]**
Deflection lSpring	0.0	-1.0	0.0	0.0
Deflection lDamp	0.0	-1.0	0.0	0.0
Deflection lBuf	0.0	-1.0	0.0	0.0
Deflection lStabi	0.0	1.0	0.0	0.0

Figure 6.14: Translations and Rotations

Parameter	Description	Unit (Compress)	Unit (Steering)
tx	wheel base variation	$m/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}}$
ty	track variation	$m/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}}$
tz	wheel travel variation	$m/m_{\text{wheel travel}}$ (it is of course always 1!)	$m/m_{\text{steering rack travel}}$
rx	camber variation	$rad/m_{\text{wheel travel}}$	$rad/m_{\text{steering rack travel}}$
ry	spin angle variation	$rad/m_{\text{wheel travel}}$	$rad/m_{\text{steering rack travel}}$
rz	toe variation	$rad/m_{\text{wheel travel}}$	$rad/m_{\text{steering rack travel}}$
lSpring	Spring length variation	$m/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}}$
lDamp	Damper length variation	$m/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}}$
lBuff	Buffer length variation when activated	$m/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}}$
lStabi	Stabilizer length or angle variation (according to the unit you have written the stabilizer stiffness)	$m/m_{\text{wheel travel}}$ or $rad/m_{\text{wheel travel}}$	$m/m_{\text{steering rack travel}}$ or $rad/m_{\text{steering rack travel}}$

The columns represent the following options:

Column	Description
Static	The values in this column indicate the design static values, i.e when the wheel travel is null.
Compress	The values in this column give the variation of a parameter (e.g the camber angle rx, in rad), per meter of wheel travel.
Steering	The values in this column give the variation of a parameter (e.g the camber angle rx in rad), per meter of steering rack displacement. This column is used only for a front axle (Mode Linear2D).
Opposite	The values in this column give the variation of a parameter (e.g the camber angle rx in rad), per meter of wheel travel of the opposite wheel.

The axis are oriented as follows:

- X: Forward
- Y: Left
- Z: Upwards

To parameterize an axle, only the left side needs to be implemented, since CarMaker assumes symmetry and automatically mirrors the values for the other side.

A non-linear kinematics description is more accurate, but according to the users needs, a simple, linear 2D-model is often be sufficient. A detailed explanation of the non-linear kinematics can be found in the User's Guide, as well as in the Reference Manual.

Compliance

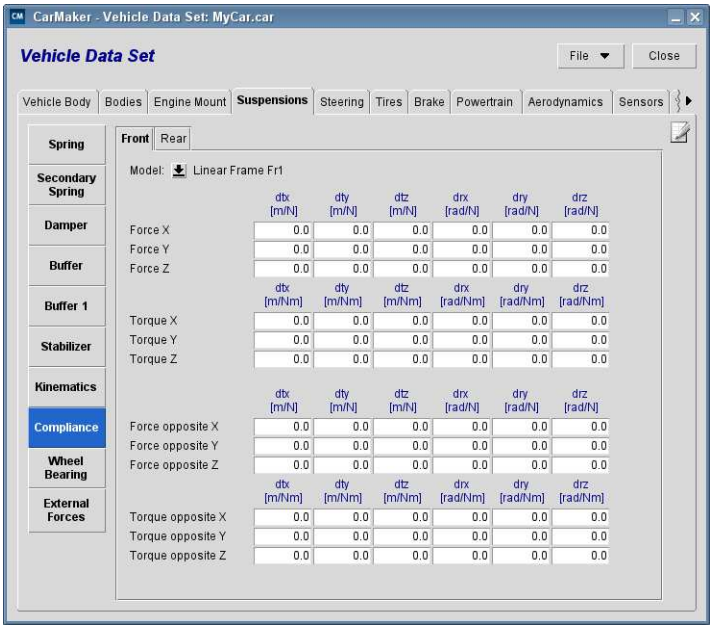


Figure 6.15: Compliance definition

The compliance description is optional. Following models can be activated:

- Linear description of the compliance model: To activate this model, either *CoeffConstrFr1* (coefficient defined in the vehicle frame) mode or *CoeffConstrFr2* (coefficient defined in the wheel carrier frame) mode must be selected.
- Non-linear compliance model: To activate this model, the user has to write the compliance table in the same file as the one used for the linear kinematics. An explanation for the non-linear compliance model is provided in the User's Guide and Reference Manual. The compliance triggers additional movements of the wheel, depending on the forces applied at the wheel center.

The parameters in the tables of the editor are coefficients that define the variation of the wheel position in meters, rad per Newton (when a force is applied) or Newton-meters (when a torque is applied) at the wheel center and possibly at the wheel center of the opposite wheel.

Wheel Bearing

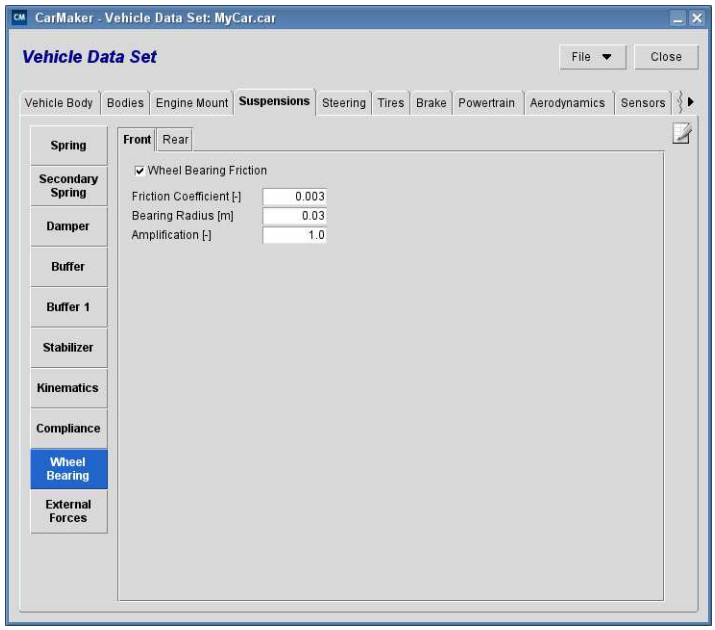


Figure 6.16: Wheel Bearing definition

This option activates/deactivates the consideration of the friction in the wheel bearings. Upon activation, further parameters can be visualized.

The wheel bearing friction is characterized by a single constant coefficient that defines the friction force on the bearing. The bearing radius enables evaluation of the friction torque on the bearing based on the friction force.

External Forces

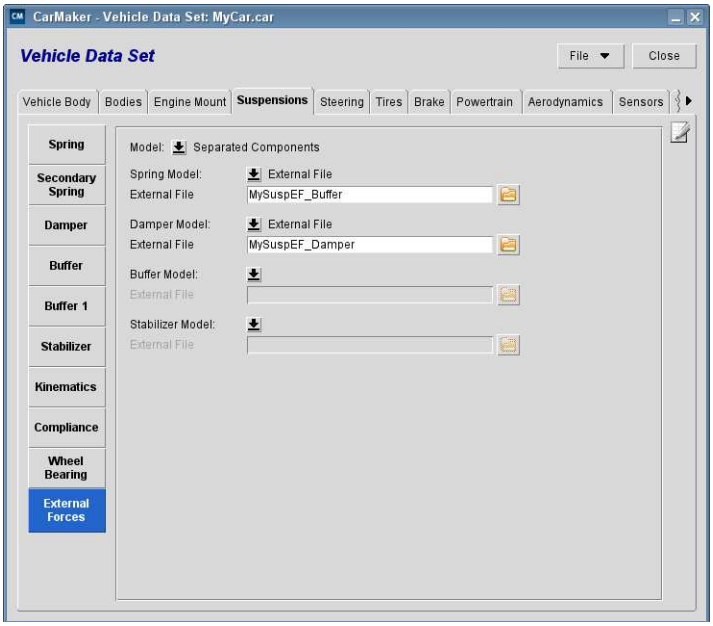


Figure 6.17: External Forces definition

These options are only useful when the CarMaker kinematics model is extended, e.g. when modeling an active stabilizer.

6.3.5 Steering System

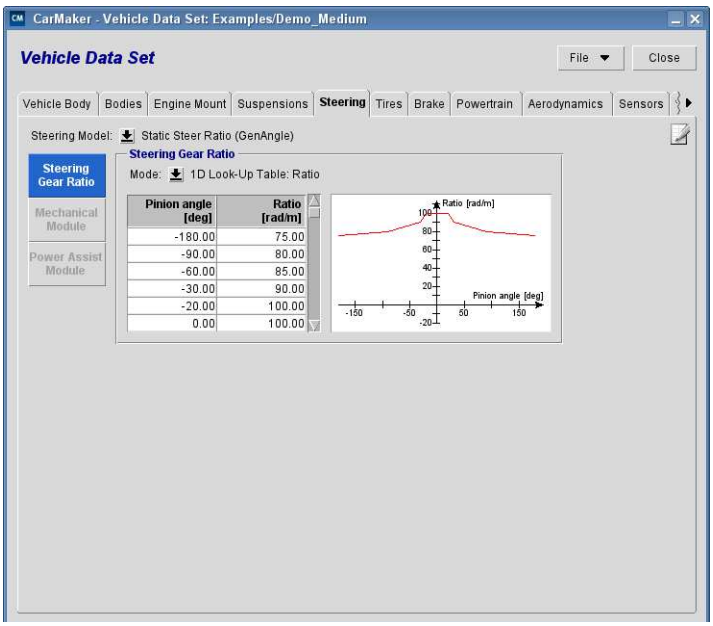


Figure 6.18: Steering system definition

In this window, the relation between the steering wheel angle or steering wheel torque and the steering rack displacement is defined. This relation can be linear (Mode *Static*) or non-linear (Mode *Dynamic*).

The overall steering ratio, e.g. the ratio between the steering wheel angle and the wheel angle is calculated in CarMaker by multiplying the steering ratio *Rack to steering wheel* with the kinematics parameter *rz* (angle variation of the wheel at the z axis) in the *Steering* column of the kinematics description.

Additionally, an extended steering model called *Pfeffer with Power Steering* is available. This model provides a highly detailed mechanical model of the steering column, intermediate shaft, torsion bar and steering rack, including damping and friction effects. The power steering can be either hydraulic or electric.

6.3.6 Tires

In this tab, the default tire set used in combination with the vehicle is selected. Please refer to section 6.6 'Creating a New Tire Data Set' for more details on supported tire models and their parameterization.

6.3.7 Brake System

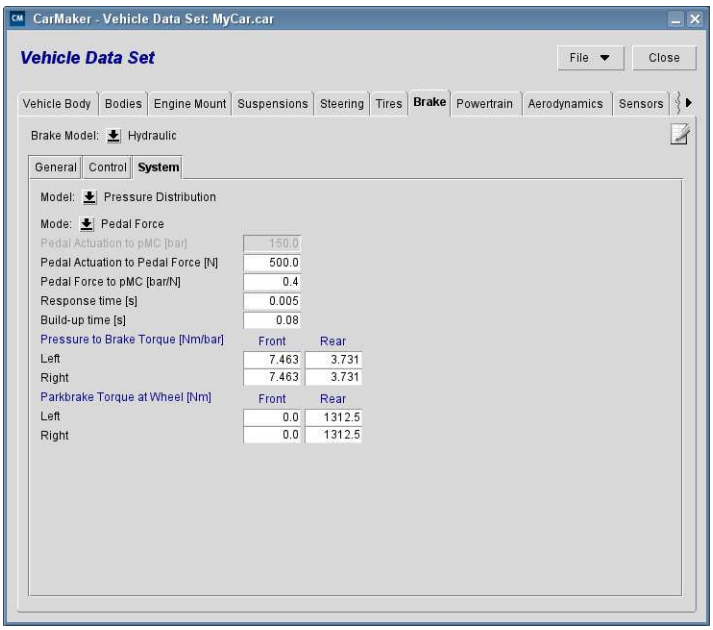


Figure 6.19: Brake system definition

The user can decide between two hydraulic models:

- *PresDistrib*: This is a simple model that converts the force at the brake pedal to a pressure in the master cylinder and then to a braking torque at each of the four wheels. The parameter *Pedal Actuation to Pedal Force* defines the force required by the driver to push the brake pedal to the maximum.
- *External File > HydESP*: This model can be used if an ESP controller model is to be simulated. With this hydraulic model, the hydraulic part of the brake system can be parameterized as well. This is specific to the ESP assembly.

6.3.8 Powertrain

Model

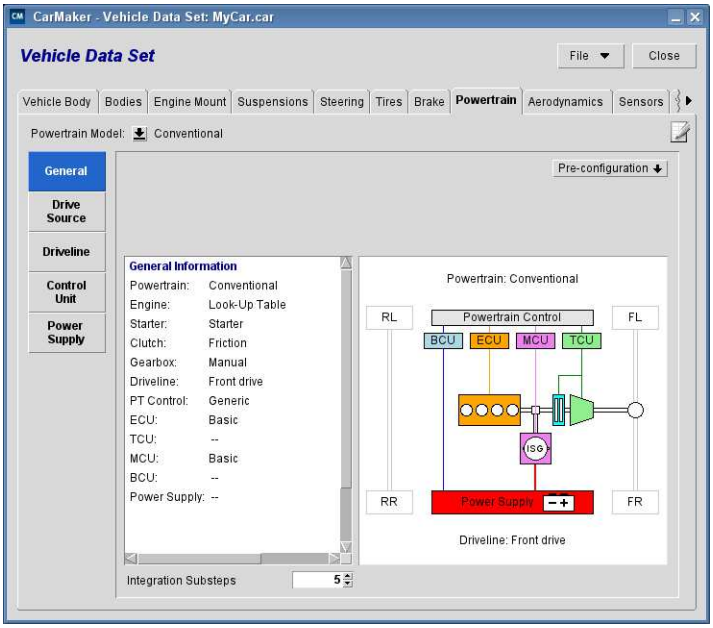


Figure 6.20: Powertrain Model definition

The option *Powertrain - Model* selects the kind of torque generation. *Conventional* would be a common combustion engine, whereas *OpenXWD* disconnects the gearbox output from the wheel in order to insert for e.g. a generator. This model can even be used to replace the whole engine and drivetrain. The options *GT_SUITE* and *AVL_CRUISE* offer an interface to co-simulation with an external drivetrain modeler.

The option *Driveline - Model* selects the kind of transmission: front, rear, or any kind of 4WD.

Engine

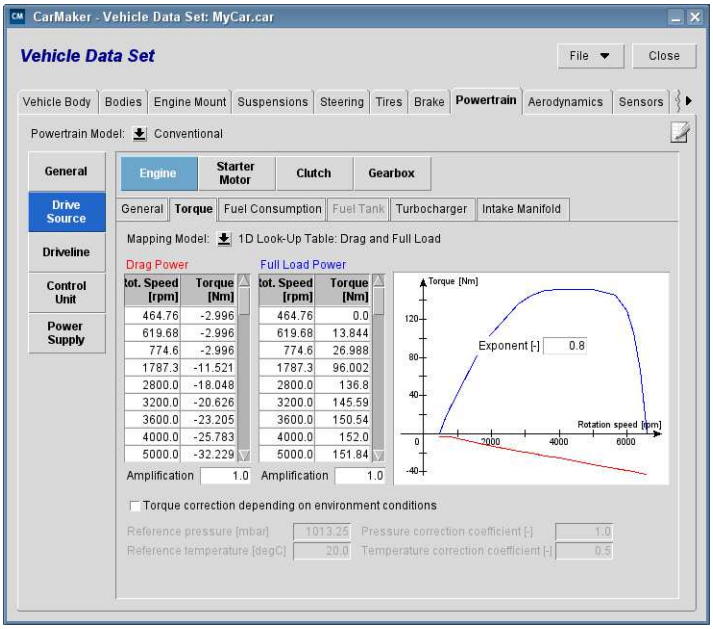


Figure 6.21: Engine definition

The user can define the torque characteristics of the engine. In the menu *Engine Model,* the type *Look-Up Table* and in the menu *Engine Mapping* the type *1D Look-Up Table: Drag and Full Load* can be chosen.

This option lets the user parameterize the full load power characteristic, as well as the negative torque of the engine when slowing down.

Fuel Consumption

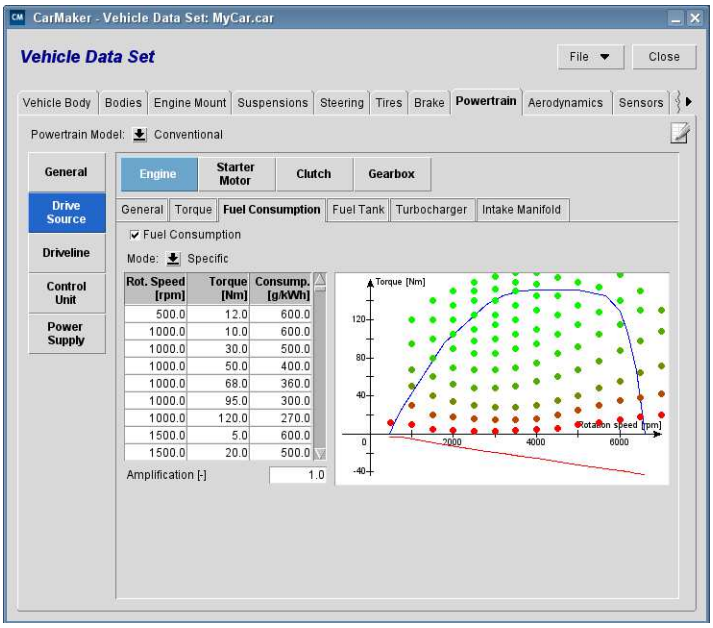


Figure 6.22: Fuel Consumption definition

Optionally, the user can also determine the consumption characteristics of the drivetrain.

Clutch

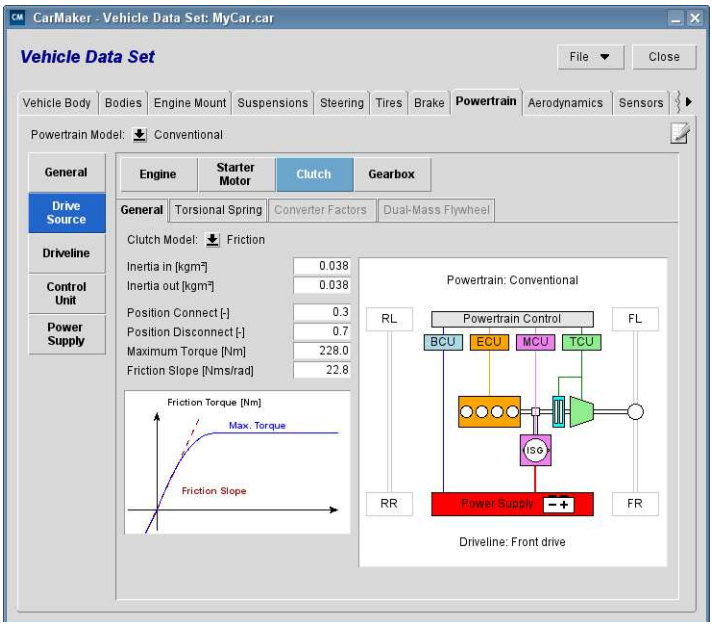


Figure 6.23: Clutch definition

Gear Box

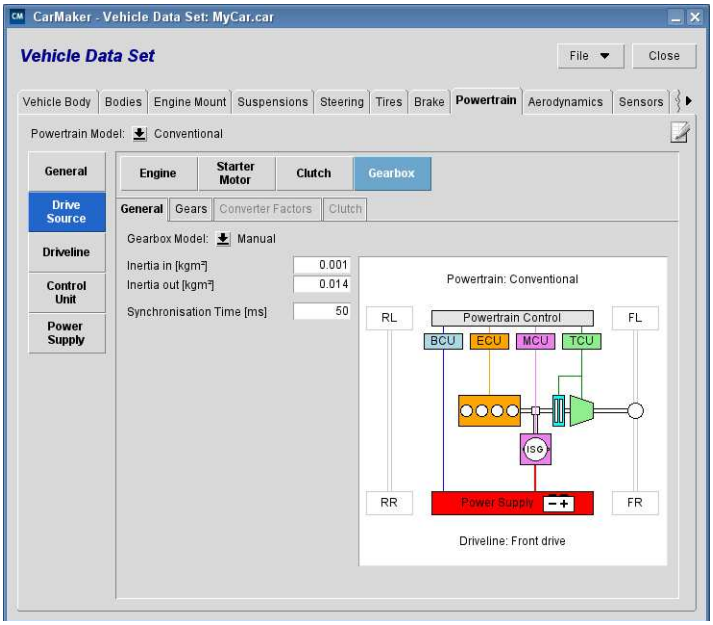


Figure 6.24: Gear box definition

Here you can parameterize the gear ratios. Nine different gearbox models are available.

Driveline

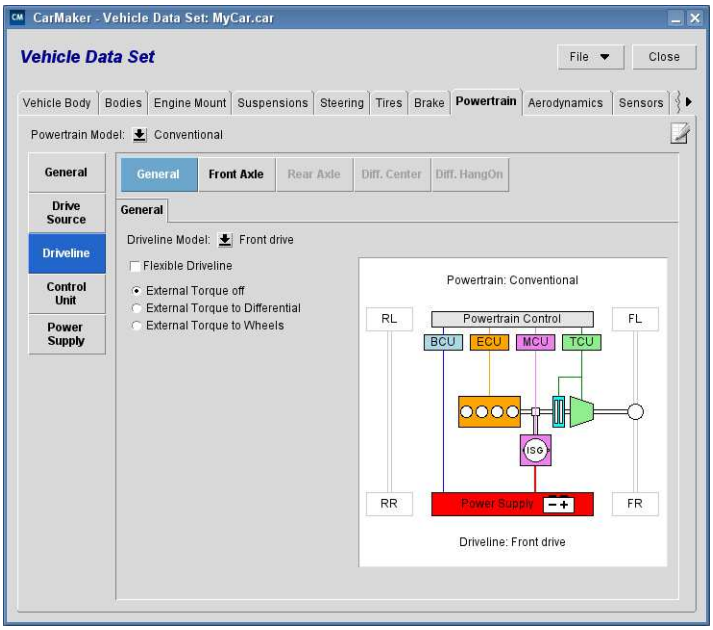


Figure 6.25: Driveline definition

In CarMaker you have the possibility to choose, if your car is front, rear or all wheel driven. You also have the option to select a shiftable all wheel drive and to determine if you want your driveline to be considered flexible or stiff.

Differentials

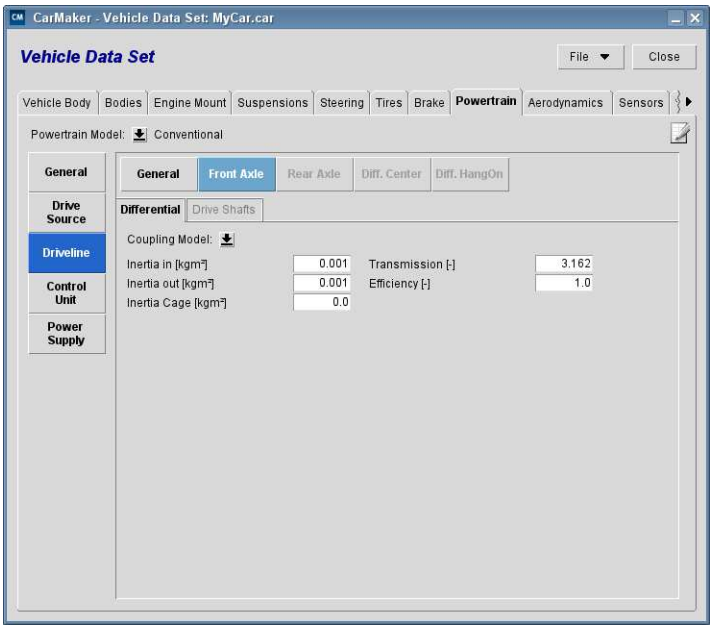


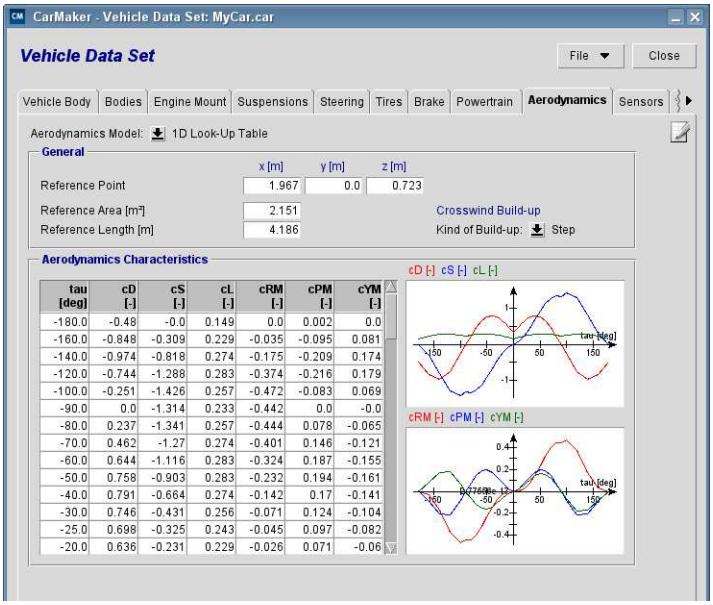
Figure 6.26: Differentials definition

According to the selected transmission mode, various differentials need to be parameterized. The differential models are the same. Default *Mounting* is *left to right*, other options are for very specific use.

If the user requires a very simple differential coupling model, as is common on most European cars, *Coupling Model* = *not specified* can be defined.

Modeling torque vectoring is possible by choosing the option *Coupling Model* = *Torque Vectoring*. Then, the torque distribution can be manipulated by DVA (direct variable access).

6.3.9 Aerodynamics



tau [deg]	cD [-]	cS [-]	cL [-]	cRM [-]	cPM [-]	cYM [-]
-180.0	-0.48	-0.0	0.149	0.0	0.002	0.0
-160.0	-0.848	-0.309	0.229	-0.035	-0.095	0.081
-140.0	-0.974	-0.818	0.274	-0.175	-0.209	0.174
-120.0	-0.744	-1.288	0.283	-0.374	-0.216	0.179
-100.0	-0.251	-1.426	0.257	-0.472	-0.083	0.069
-90.0	0.0	-1.314	0.233	-0.442	0.0	-0.0
-80.0	0.237	-1.341	0.257	-0.444	0.078	-0.065
-70.0	0.462	-1.27	0.274	-0.401	0.146	-0.121
-60.0	0.644	-1.116	0.283	-0.324	0.187	-0.155
-50.0	0.758	-0.903	0.283	-0.232	0.194	-0.161
-40.0	0.791	-0.664	0.274	-0.142	0.17	-0.141
-30.0	0.746	-0.431	0.256	-0.071	0.124	-0.104
-25.0	0.698	-0.325	0.243	-0.045	0.097	-0.082
-20.0	0.636	-0.231	0.229	-0.026	0.071	-0.06

Figure 6.27: Aerodynamics definition

The aerodynamic model applies additional forces and torques depending on the vehicle speed and wind direction at the Aero Marker. This point is defined in the Bodies tab.

An explanation for the various coefficients can be found in User's Guide and Reference Manual.

6.3.10 Sensors

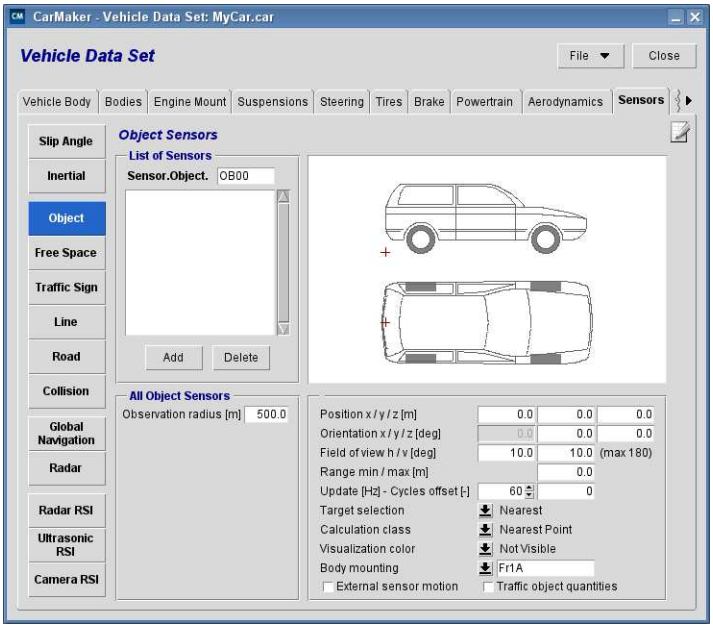


Figure 6.28: Sensor definition

The sensor model can place different sensor types on the car:

- Slip Angle Sensor: as reference where the side slip angle should be measured
- Inertial Sensor: sensor to measure accelerations and translations at certain points on the car
- Object Sensor: sensor to detect obstacles in the environment
- Free Space Sensor: sensor to detect occupied and free space in the surroundings, considers traffic objects only
- Free Space Sensor Plus: sensor to detect occupied and free space in the surroundings, considers whole 3d geometry (works on GPU)
- Traffic Sign Sensor: for traffic sign recognition
- Line Sensor: sensor which detects road markings
- Road Sensor: sensor to determine at a predefined preview distance important road specific attributes
- Collision Sensor: recognizes collisions with traffic objects
- Global Navigation Sensor: simulates the positions of satellites and a receiver build in the vehicle
- Radar Sensor: physical sensor model for obstacle detection
- Radar RSI Sensor: used for sensor component development
- Ultrasonic RSI Sensor: used for sensor component development
- Camera RSI Sensor: raw signal interface to image data provided by IPGMovie

6.3.11 Vehicle Control

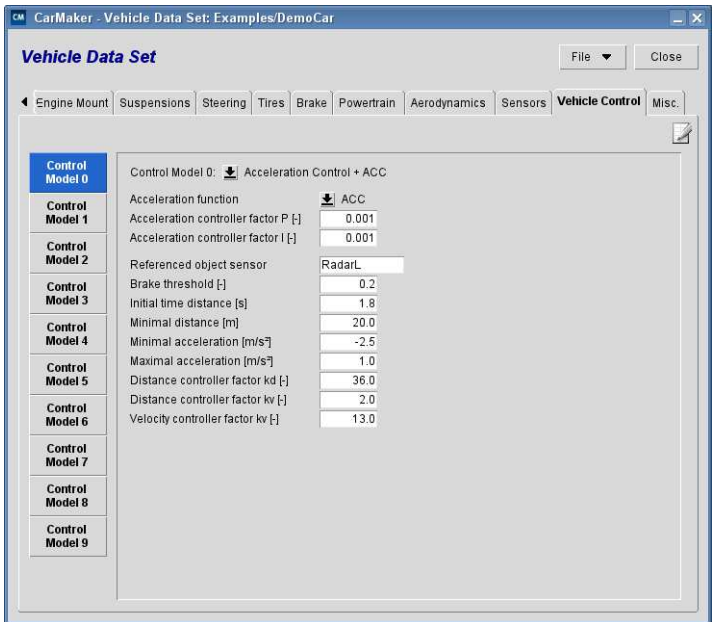


Figure 6.29: Vehicle Control selection

In the Vehicle Control tab you can select controller models like driver assistance systems. A maximum of ten Vehicle Control models can be used in one vehicle. Available example controllers are the following:

- "Acceleration Control + ACC"
  Example for an active cruise control including an acceleration controller to transfer the target acceleration to a gas / brake pedal position.
- "Generic Longitudinal Control"
  The example longitudinal controller comes with an autonomous emergency braking assistant and a forward collision warning algorithm.
- "Generic Lateral Control"
  The example lateral controller consists of a lane keeping assistance system and a lane departure warning.

6.3.12 Misc.

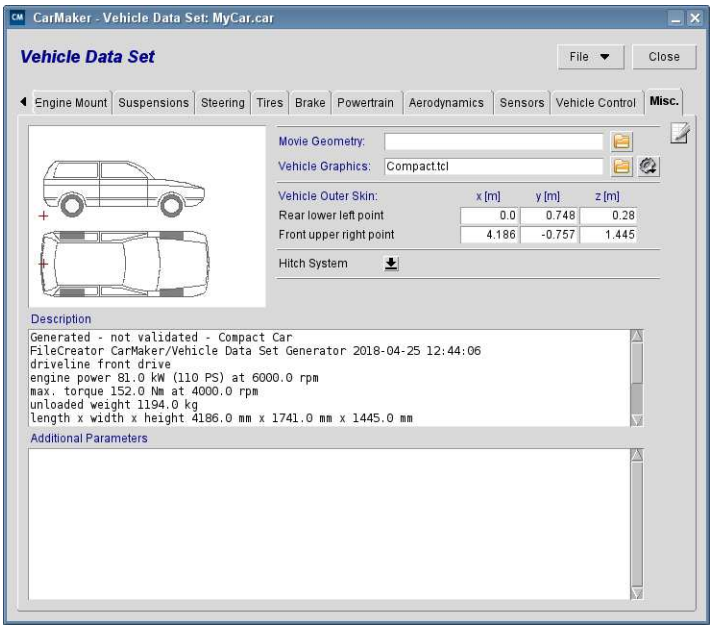


Figure 6.30: Misc. definitions

The object file used by IPGMovie can be selected under *Movie Geometry*. This file can be created by the user himself with the help of a CAD software, or one can be bought online. The file must be in .obj, .kmz, .dae or .3ds format.

File definition guidelines:

- ONLY polygons and triangles, NO splines or free forms.
- No more than 50,000 nodes.
- If this option is available while generating the file: generate the normal for all points
- Colors: Generate the corresponding material file.

Under *Vehicle Graphics*, a picture in the PNG format can be chosen to be displayed in the CarMaker main GUI and in the vehicle editor as preview to the 3D object. The user can also create an own picture by taking snapshots of the 3D-object by the help of the button next to the file browser.

The field *Additional Parameters* gives room for additional, optional parameters.

6.4 Save as a New Data Set

After you have finished parameterizing your Vehicle Data Set, you can save it independent of the TestRun. This gives you the possibility to use this Vehicle in other TestRuns.

To save a Vehicle Data Set: In the CarMaker main GUI, click *Parameters > Car* to open the Vehicle editor.

In the Vehicle editor, click on *File > Save*

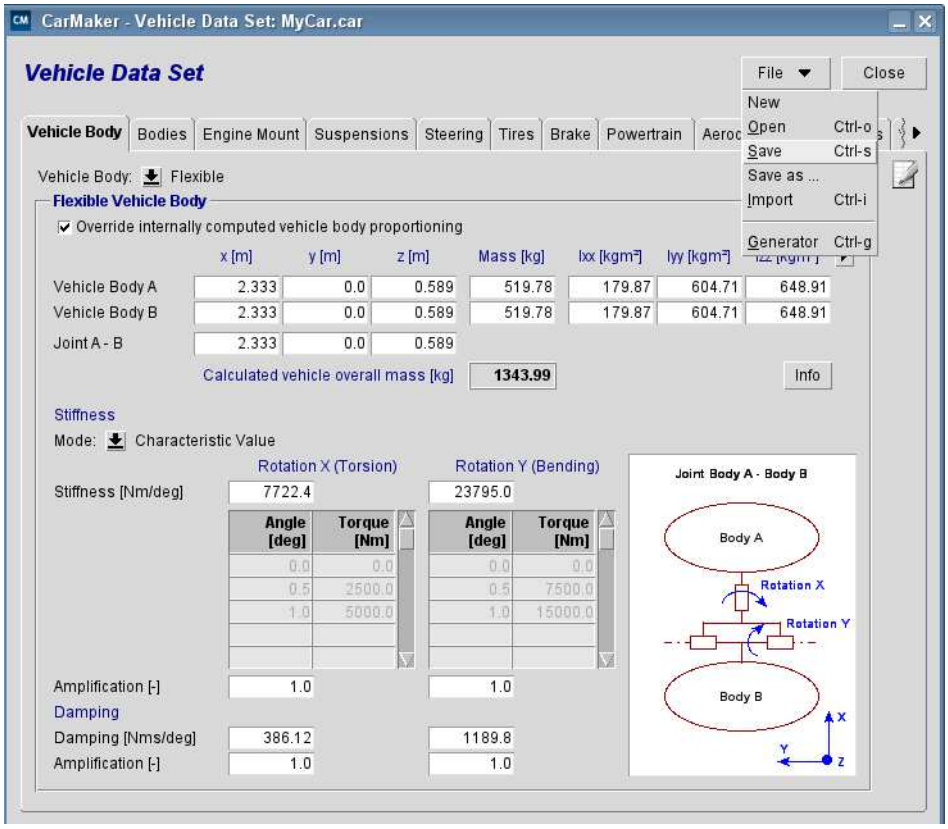


Figure 6.31: Vehicle Data Set

Type in the name of the new Data Set you are saving, e.g. “My_Car” > *OK*.

The new data set is now saved and can be integrated in other TestRuns.

6.5 Creating a New Trailer Data Set

CarMaker also provides a trailer model that can be parameterized in the same manner as the vehicle model.

For further information regarding the trailer parameterization, please refer to the User’s Guide or the Reference Manual.

6.6 Creating a New Tire Data Set

Analogously to creating an individual Vehicle Data Set, the user can also create his own Tire Data Set. CarMaker provides a variety of tire models:

- Real Time Tire (RTTire), based on the TYDEX format
- Pacejka (MF-Tire) versions 5.2 and 6.1
- MF-Tyre / MF-Swift versions 6.2 and 7.2
- Michelin TameTire 4.0
- FTyre provided by cosin, versions 2017-1 up to 2018-1
- IPGTire, for MotorcycleMaker only

6.6.1 Overview of Tire Models

RTTire

TYDEX is a common format used to save measured tire characteristics. With the help of a tool, the TYDEX file is converted into binary and, at the same time, an Infofile that points to the binary file is generated.

Finally, the Infofile is uploaded in the CarMaker GUI.

As soon as the characteristic curves of the tire (e.g. from measurements or the tire manufacturer) are present, the user can generate the corresponding RTTire file for CarMaker on their own.

Pacejka versions 5.2 and 6.1

CarMaker supports the standardized Pacejka format, as well as the ADAMS Property file.

In the first case, if the user has the Pacejka parameters, they merely need to be written in the corresponding parameter fields of the Tire dialog. Via this dialog, the parameters can also be imported from a .tir file.

MF-Tyre / MF-Swift versions 6.2 and 7.2

These tire models (formerly known as TASS Delft Tire) are supported by CarMaker, the libraries are directly integrated in CarMaker, a separate license from TASS International is needed for the version 7.2 and for some advanced functionality. Input are the Magic Parameters similar to the standard Pacejka model (ADAMS property files are accepted, too).

Michelin TameTire 4.0

CarMaker provides an interface to the thermo-mechanical tire model by Michelin. This model is not part of the CarMaker evaluation package. Please contact the CarMaker Service Team for further information (CarMaker-Service@ipg-automotive.com).

FTyre provided by cosin, versions 2017-1 up to 2018-1

The FTyre model from cosin can be co-simulated with CarMaker. For this, the FTyre library is dynamically linked to the CarMaker application. A full FTyre installation including a license file is required besides the CarMaker installation and license.

6.6.2 Tire Model Exercises

Open any random example TestRun (since this makes no difference for the aim of this exercise) and select any car, e.g. the *DemoCar*. Click on *Select* in the *Tire* field in the CarMaker main GUI. Click on the tire *MF_205_60R15_V91* and then *Edit*.

This is an example of a Pacejka file, in which the user can directly write the values of his parameters.

In the CarMaker GUI, click on *Select* in the *Tire* field again, but this time click on the tire *MF_205_60R15_V91r* and then *Edit*.

This is an Infofile that points to an ADAMS property file (see tab *Model Parameters* > *Import Adams Property File*).

Open the *ADAMS* property file: Use a text editor to open the file from your CarMaker project folder under *Data* > *Tire* > *Examples* > *TirePropertyFile* > *MF_205_60R15_V91.tir*.

This is the same Pacejka data set as previously opened, but this time it is written in the ADAMS format.

Please note that an internal mechanism can erase the instability at standstill caused by the Pacejka model.

The changes made in this section must not be saved.

Each CarMaker example vehicle contains a suitable tire data set. Tires only need to be selected in the main GUI, if the user wishes to change the tire configuration for a TestRun.

Chapter 7

CarMaker for Simulink

7.1 Starting CarMaker for Simulink

Open CarMaker using the Windows Start Button and create a new project: *CarMaker GUI > File > Project Folder > Select / update project* and toggle the option *CarMaker for Simulink Extras*. Click on OK to finish creating the project folder.

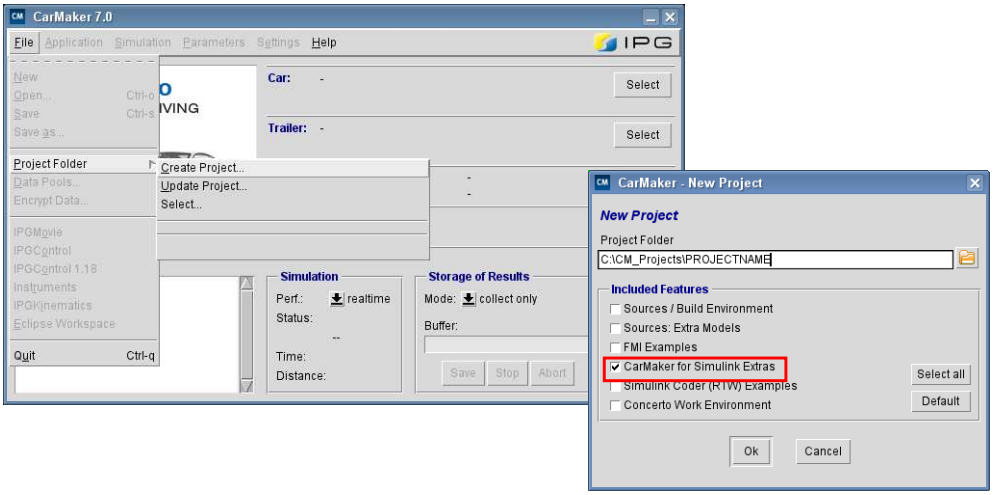


Figure 7.1: Creating a new project with *CarMaker for Simulink Extras*

Now the project folder contains an additional folder called *src_cm4sl* which provides the interface to Simulink.

Close the CarMaker GUI and open Matlab. As working directory, select the folder *src_cm4sl* of the project.

In the tab on the left that displays the contents of the working directory, right-click on the file called *cmenv.m* and run the script. This option ensures the connection between CarMaker and Matlab.

Open the *generic.mdl* model.

Now, Matlab should have opened, followed by the Simulink environment and the Simulink model *generic.mdl*. A successful start-up procedure leads to the following window:

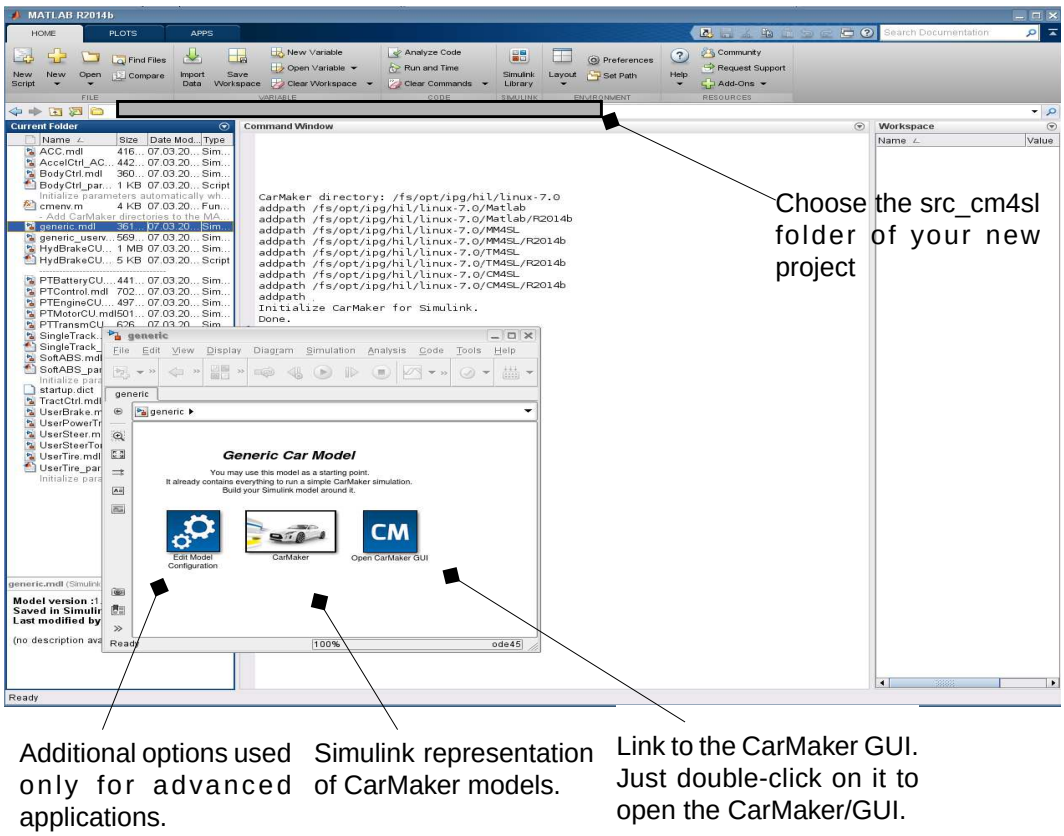


Figure 7.2: MATLAB window with Simulink model *generic.mdl*

Troubleshooting

Sometimes, the file type .mdl is not associated with Matlab. A new dialog appears which asks to open the file with one of the following programs:

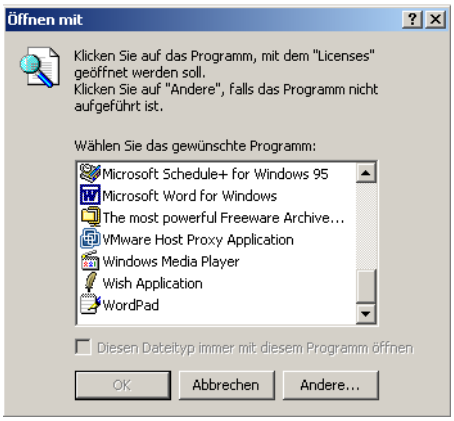


Figure 7.3: *Open with...* dialog window

The Matlab executable must be found and then the execution can be confirmed.

Start-up confirmation

Matlab and CarMaker for Simulink work together by extending Matlab's search path, so it knows where to find the CarMaker for Simulink blockset and auxiliary commands. The Matlab search path is extended by the execution of a small script called *cmenv.m*, which is located in the mdl subdirectory - *src_cm4sl* - of the CarMaker project directory. The script may be executed manually, but there is also a way to invoke it automatically each time Matlab is started or a model is loaded.

The most important rule is: Always keep the cmenv.m script in the same directory as the Simulink model. Whenever a Simulink model which contains the CarMaker subsystem block from the CarMaker for Simulink blockset is loaded, cmenv.m will be executed automatically. This default behavior should be sufficient for most uses of CarMaker for Simulink.

The cmenv.m script should generate a message similar to the one below in the Matlab console window:

```
CarMaker directory: c:/ipg/hil/win32-x.x
addpath c:/ipg/hil/win32-x.x/Matlab
addpath c:/ipg/hil/win32-x.x/Matlab/vx.x-r20xxx
addpath c:/ipg/hil/win32-x.x/CM4SL
addpath c:/ipg/hil/win32-x.x/CM4SL/vx.x-r20xxx
addpath <x:/path/to/your/project_dir>/src_cm4sl
Initialize CarMaker for Simulink.
Done.
```

7.2 First simulation with CarMaker for Simulink

The generic.mdl example is intended as a start-up example model and provides a basic CarMaker subsystem (similar to the stand-alone version) without any additional control blocks. More details on this system will follow in the next section.

Make sure any previously open CarMaker GUIs are closed.

Double-click on the “Open GUI“ block in Simulink and the CarMaker for Simulink window pops up:

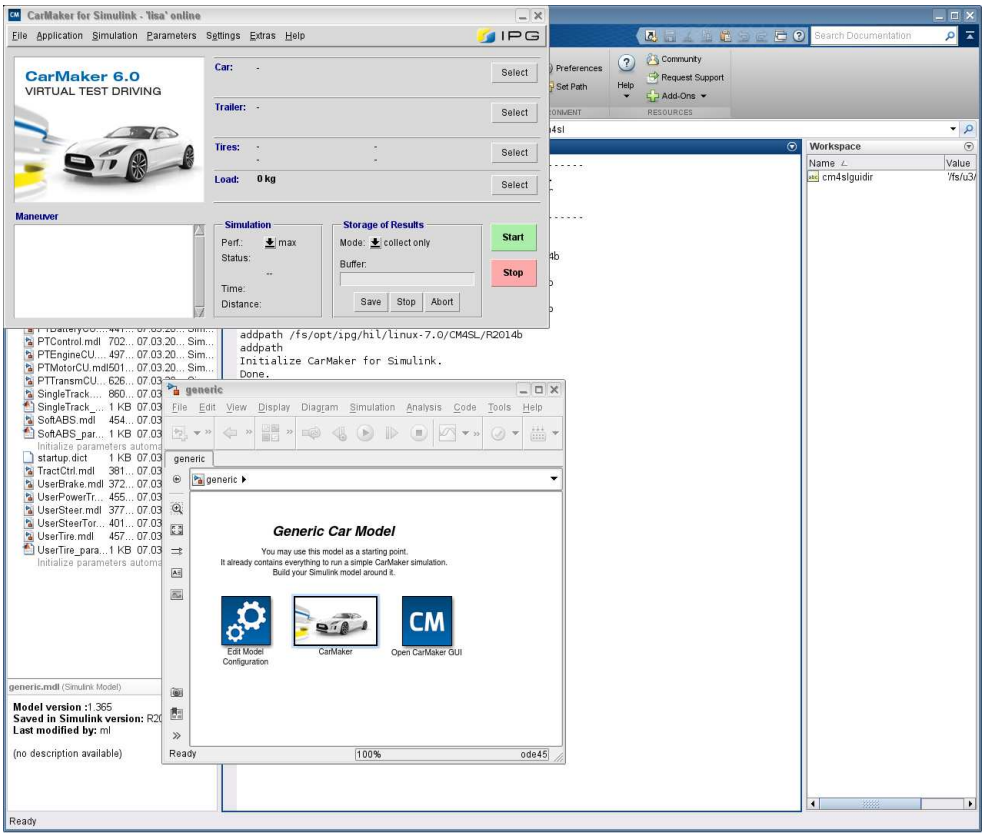


Figure 7.4: CarMaker GUI called up from MATLAB

The same GUIs as in the previous chapters, containing information on loading a TestRun, changing a vehicle, etc., will open.

Load the TestRun *Product Examples > Examples > BasicFunctions > Driver > HandlingCourse.*

The environment is very similar to usual. Even within Simulink, no special Simulink model needs to be prepared.

When the model “Generic“ is loaded, the simplest vehicle model (this time it’s a model, not a data set!) is selected, which is in fact the counterpart of the model available in the stand alone version of CarMaker.

Now CarMaker for Simulink is ready to work!

In the following section, the basic differences between CarMaker stand-alone and CarMaker for Simulink are presented.

7.3 Description of the CarMaker Environment in Simulink

CarMaker for Simulink is a complete integration of IPG's vehicle dynamics simulation software into the MathWorks modeling and simulation environment Matlab/Simulink.

Thus, in CarMaker for Simulink, the user is able to extend the *vehicle model* itself.

The highly optimized and robust features of CarMaker were added to the Simulink environment using an S-function implementation and the API functions that are provided by Matlab/Simulink.

CarMaker for Simulink is not a loosely coupled co-simulation but a closely linked combination of the two best-in-class applications, resulting in a simulation environment that assures both good performance and stability.

Because of this integration, it is now possible to use the power and functionality of CarMaker in the intuitive and full-featured Simulink environment. Furthermore, using CarMaker in Simulink is no different than using standard S-function blocks or built in Simulink blocks. The CarMaker blocks are connected the same way other Simulink blocks are connected and existing Simulink models can now easily be added to the CarMaker vehicle model with literally a few clicks.

Integration does not, however, mean a loss of functionality, seeing as all the features that make CarMaker the premier software in its domain have been included, and can now be used together with Simulink's rich tool set. The CarMaker GUI can still be used for simulation control and parameter adjustments, as well as defining maneuvers and road configurations. IPGControl can still be used for data analysis and diagram plotting. IPGMovie can still be used to bring the vehicle model to life with realistic animation and rendering of the multi-body vehicle model in three-dimensional space.

In short, CarMaker for Simulink is not a stripped down version of the original, but a complete system that can become a part of any Simulink simulation quickly and easily. The next steps will show just how easy it can be.

7.4 Advantages and Disadvantages of using CarMaker for Simulink

Feature	CarMaker - standalone	CarMaker running in Simulink	Comment
Simulation Speed	Very fast.	Slow, especially with additional models.	The simulink environment needs a lot of processing power. The best performance is reached when the simulink model "generic.mdl" is used. Even then, there is a big difference in simulation speed. Even little add-ons to the model can lead to performance loss.
Flexibility	Full flexibility.	Limited flexibility, every implemented model will be simulated by Simulink even when it is not needed.	The c-code offers the opportunity to use parts of the simulation only when they are needed (e.g. additional submodels, dynamic traffic objects).
Exchange of models	Easy, all global variables accessible in the c-library are available.	Very easy due to the graphical interface, but only the interface variables and User Accessible Quantities are available.	To use the interfaces of the stand alone version, c-code knowledge is required.
Manageability	Very good. C-Code is ideal to be structured. Creating libs and objects, you can also share models easily.	Very good up to a middle diversity level. From there on models become harder and harder to manage.	The c-code is mostly well arranged and is better to understand than the complex simulink models.
Graphical modeling	Less comfortable, only using RealTime-Workshop.	Very easy.	Especially for beginners and people without c-code knowledge, it is a lot easier to extend CarMaker in Simulink.
Value for Money	Very good.	Same as stand alone + additional costs coming with Matlab/Simulink.	Both variants are available in the CarMaker/Office package. For the Simulink version you need Matlab and Simulink which cost a few thousand Euros in addition.

Chapter 8

Features of the CarMaker for Simulink blockset

The CarMaker for Simulink blockset can be found in the Simulink library browser, under *Blocksets and Toolboxes*. It is possible to double click *CarMaker for Simulink* or simply type

```
>> CarMaker4SL
```

in the Matlab command window.

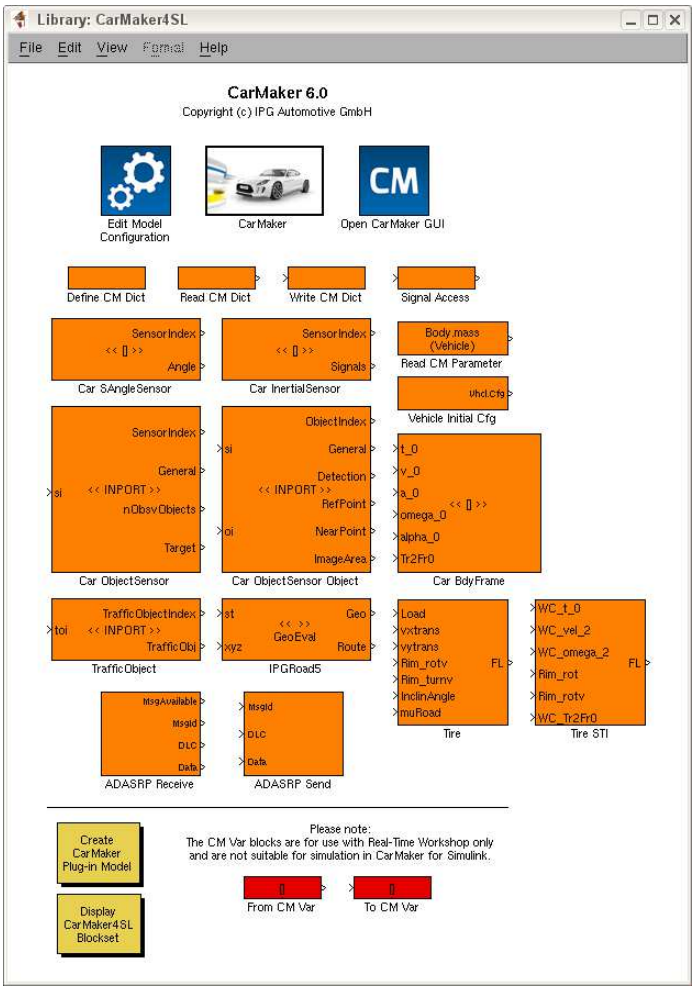


Figure 8.1: Simulink library

8.1 General Information

8.1.1 Block Properties

The CarMaker for Simulink blocks of the CarMaker subsystem are directly fed through and run with a fixed step size of 1ms, independent of the rest of the model.

CarMaker for Simulink blocks do not have continuous states in the Simulink sense of the word. They have internal state variables but these are integrated using CarMaker internal solvers, independent of Simulink.

8.1.2 Modeling Principles

The CarMaker for Simulink simulation model consists of a subsystem containing a chain of individual blocks. When Simulink executes CarMaker, all blocks of the CarMaker block chain must be executed exactly once and in order. Algebraic loops involving individual blocks of the CarMaker block chain are not permitted.

There are two possibilities for replacing existing CarMaker functionalities by Simulink blocks:

- Overwriting signals
- Disabling unneeded internal CarMaker functionalities using special parameters

It is not recommended to:

- Replace, remove or reorder CarMaker blocks

For further detail, please refer to the following demonstration examples and their descriptions.

8.1.3 Purpose of the Sync_In and Sync_Out Ports

The *Sync_In* and *Sync_Out* ports are an important concept for CarMaker for Simulink.

- They guarantee the proper order of execution for the CarMaker blocks.
- They let the user choose exactly when a CarMaker dictionary variable is accessed with a *Read CM Dict* or *Write CM Dict* block. E.g., this way it is possible to read the most up to date value of a variable (and not the value calculated in the previous cycle) or override the value of a variable only after it has been calculated internally by CarMaker.

8.2 Utility Blocks

In the following sections the most important elements of the CarMaker for Simulink library are introduced. A complete description of all CarMaker for Simulink blocksets can be found in the Programmer's Guide.

8.2.1 CarMaker GUI

The CarMaker GUI block is used to connect the Simulink model to a running CarMaker GUI process. If no running process is detected, it will prompt the user to open a new CarMaker GUI and manually reconnect the undetected CarMaker GUI or cancel the connection attempt.

Double clicking on the CarMaker GUI block can also be used to switch between models, e.g. to activate a certain model in case several models have been loaded in Matlab. The active model is the one that will be simulated when the *Start* button in the CarMaker GUI is clicked. Please also note that the simulation must be started this way, via the GUI.

8.3 CarMaker Dictionary Blocks

8.3.1 Read CM Dict

The *Read CM Dict* block reads a variable in the CarMaker dictionary and provides its current value on the block's output port. The variable doesn't need to be defined with a *Define CM Dict* block in the model; any existing dictionary variable can be read.

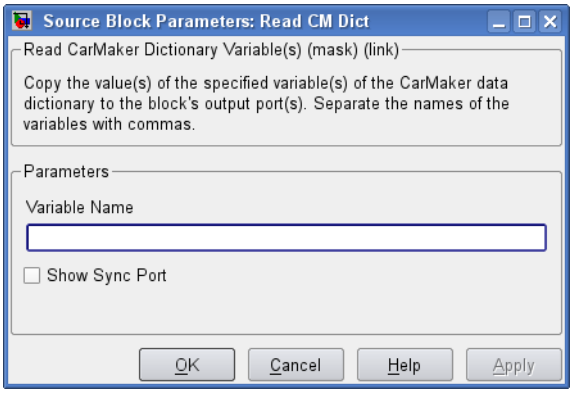


Figure 8.2: Read CM Dict block parameters dialog

Enter the name of the variable in the block's parameters dialog. In the model, the name will be displayed inside the block's symbol.

When a non-existing dictionary variable is defined as a parameter of a *Read CM Dict* block, Simulink will report the following error at the start of the simulation:

```
Variable 'abcde' not in dictionary.
```

8.3.2 Write CM Dict

The *Write CM Dict* block writes the current value at the block's input port to a variable in the CarMaker dictionary. The variable doesn't need to be defined with a *Define CM Dict* block in the model; any existing dictionary variable can be overwritten.

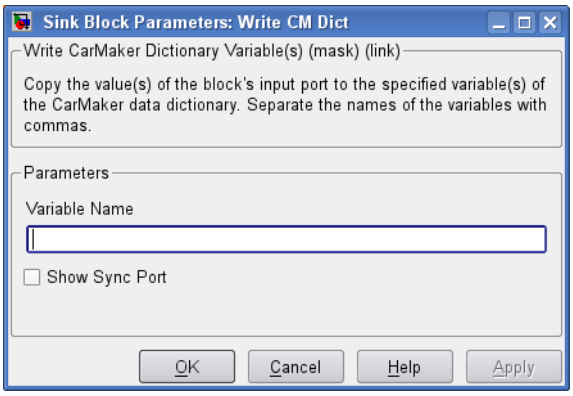


Figure 8.3: Write CM Dict block parameters dialog

Enter the name of the variable in the block's parameters dialog. In the model, the name will be displayed inside the block's symbol.

When a non-existing dictionary variable is defined as a parameter of a *Write CM Dict* block, Simulink will report the following error at the start of the simulation:

```
Variable 'abcde' not in dictionary.
```

8.3.3 Define CM Dict

A Simulink model can define its own variables in the CarMaker dictionary, because there might be signals that are to be monitored with IPGControl or accessed using Direct Variable Access. Furthermore, only signals defined in the CarMaker dictionary can be saved as part of the CarMaker simulation result file.

The *Def CM Dict* block defines a variable in the CarMaker dictionary.

When a dictionary variable is defined in a Simulink model, it is recommended to prefix its name with the model's name or other convenient abbreviation. This makes it easier to identify the model's variables in the dictionary with tools like the CarMaker GUI or IPGControl.

Example: A dictionary variable `xyz` defined in a Simulink model called `MyModel` should be given the name `MyModel.xyz` or `MM.xyz`.

It is important not to define a dictionary variable with a *Define CM Dict* block that is already defined somewhere else. Currently no error message will be issued in this case.

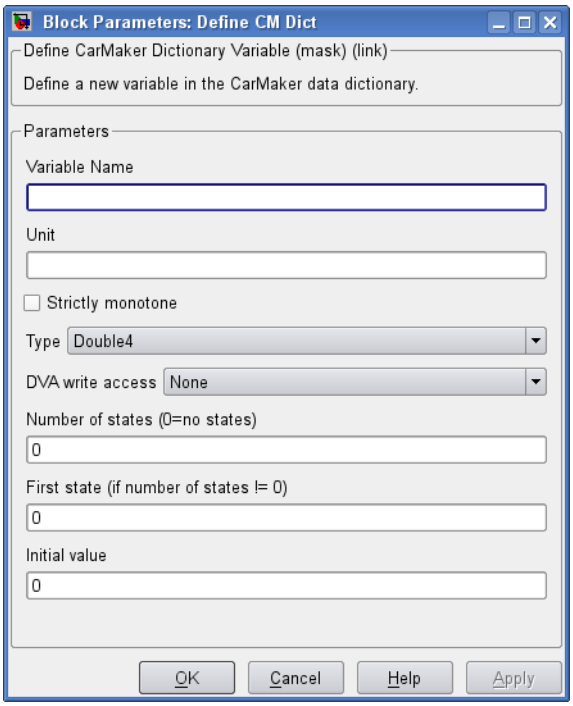


Figure 8.4: Define CM Dict block parameters dialog

Enter the name of the variable in the parameters dialog. In the model, the name will be displayed inside the block's symbol.

The variable may also be given an unit. This is recommended, but optional. IPGControl provides a list of the units used in CarMaker. Specifying a unit serves as a kind of documentation for the variable and allows IPGControl to display it on one axis with other dictionary variables of the same unit.

If the variable's values are strictly monotonic, increasing over time, the *Strictly monotone* checkbox should be selected. Again, this is an information that tools like IPGControl can use for proper display of the values.

The variable's type can be chosen according to the user's needs and the range of values of the variable. Possible types are floating point types and integer types. When in doubt, it is advised to use *Double*.

For discrete variables (any of the integer types) with values starting at 0 and a reasonably low upper limit (e.g. an indicator light that is either on or off), it may make sense to specify the number of discrete states of the variable. For the indicator light there would be two discrete values (0=off, 1=on). Again, this information serves mainly IPGControl, which displays variables with a limited number of states in a special, space saving way. Specifying a value of 0 in this field means that no special state info is available. For the *Double* and *Float* type, the value of this field is ignored.

8.3.4 Dictionary Initialization

When CarMaker for Simulink has been started, but before the first simulation is run, the CarMaker dictionary does not yet know of any additional dictionary variables that will be defined in the model. To make these variables known to CarMaker for Simulink at start-up, a file called *startup.dict* can be created in the model directory (*src_cm4sl* folder), that describes the variables' properties.

The file is in ASCII format and may contain empty lines, comment lines starting with a hashtag and lines defining dictionary variables. Each line containing a variable definition consists of five columns, separated by tabs and spaces:

- Column 1: Name of the variable.
- Column 2: Unit of the variable. A minus sign (-) means that the variable has no unit. The unit is used only for display purposes, in tools like IPGControl. CarMaker for Simulink uses SI units internally and doesn't convert units automatically.
- Column 3: Variable type. Following types (and their corresponding C-type) are allowed:

```
double              double
float               float
ulongunsigned       long
longsigned          long
ushortunsigned      short
shortsigned         short
ucharunsigned       char
charsigned          char
```

- Column 4: Number of states.
- Column 5: Special properties. Again, these are only for display purposes. If the variable is monotone and increasing, the user must specify "M", otherwise "-".

Examples:

```
PT.GearBox.GearNo   -      long      0      -
PT.Engine.rotv      rad/s  float     0      -
Car.Distance        m      float     0      M
```

For more information refer to the documentation of the *Define CM Dict* block.

Chapter 9

How to Extend the Simulink Model

This chapter will give a short yet intensive insight into the Simulink implementation of CarMaker. Basic Matlab and Simulink knowledge is a requirement.

9.1 Overview of the CarMaker Simulink Structure

Double-click the CarMaker block.

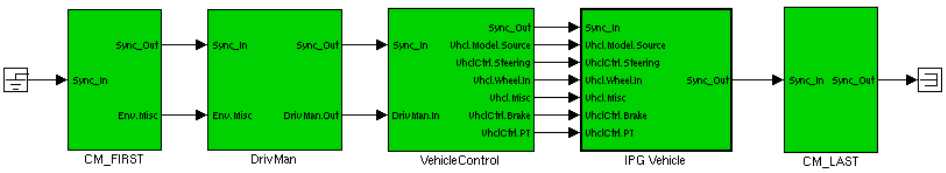


Figure 9.1: General structure of CarMaker in Simulink

This is the general structure of CarMaker in Simulink. CarMaker's Simulink representation consists mainly of a Driver/Driving Maneuver, Vehicle Control and Vehicle model.

The rest (IPGRoad, etc.) is contained in the CarMaker library for Simulink.

Double-click the DrivMan block.

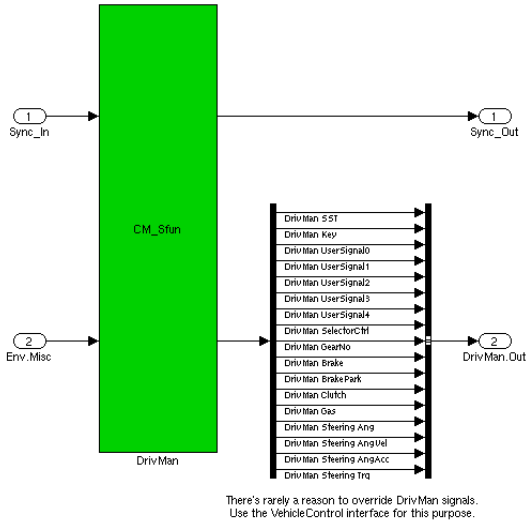


Figure 9.2: DrivMan block in Simulink

Two details can be recognized:

- The green block on the left is an S-function. This means that the full functionality is masked behind this level.
- Output of the S-function is a multiplexed signal that contains all information regarding the interface of the S-function.

Return and double-click the Vehicle block.

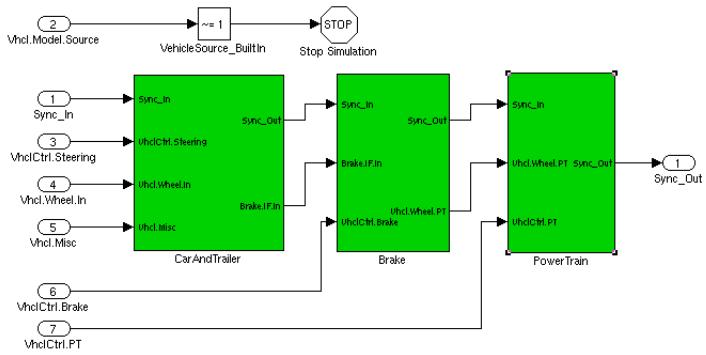


Figure 9.3: Vehicle block in Simulink

Here, a more detailed segmentation can be found. The Simulink block representation ends with an S-function each time.

9.2 Example

9.2.1 Contents

- Calculate a new variable.
- Link the variable to CarMaker's User Accessible Quantities.
- Observe the changes using IPGControl.

9.2.2 Create a new Simulink Model

Open the Simulink Model "Generic.mdl".
Save it as "My_Model.mdl".

Open the *Generic.mdl* model and then from this model, open the model *My_Model.mdl*.

9.2.3 Prepare the extension

Sometimes the calculation of additional variables takes place independent of the vehicle behavior. But most times, some information regarding one or more state variables of the vehicle is required to generate the necessary information.

For example, if the user wants to calculate the current engine power, information regarding the engine speed and torque is required.

Multiplying both and using the factor Watt to Horsepower will deliver the current engine power in the unit Horsepower.

Now navigate to the mdl folder of the project directory, open "My_Model.mdl" and double-click the CarMaker block.

It doesn't matter in which subsystem the change takes place. Simulink offers subsystems only for graphical purposes. At simulation time, sub-systems do not exist.

Switch to the CarMaker > IPG Vehicle > PowerTrain subsystem.

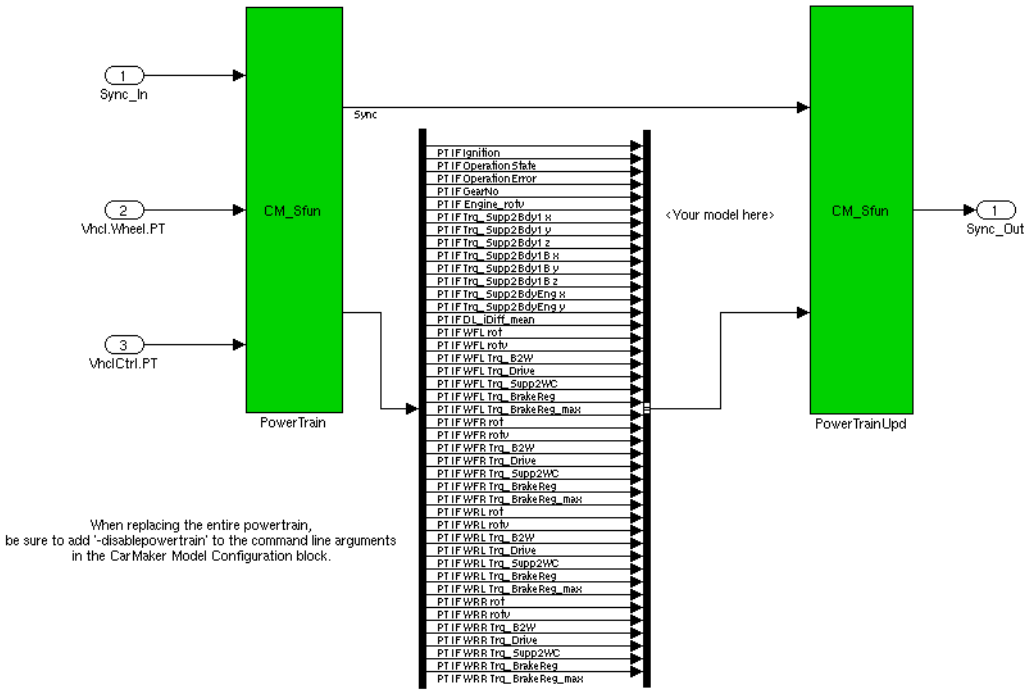


Figure 9.4: Powertrain Subsystem

Now, information from the powertrain component “Engine” will be accessed.

9.2.4 Connect Inputs

The most important values of the simulation are the signals and busses between sub-models. If possible, these signals should always be used to get information regarding the simulation.

Sometimes additional information is required, that is displayed in IPGControl as a User Accessible Quantity, but may not be available as a Simulink signal.

In this case the information can be accessed using a CarMaker4SL block *Read CM Dict*. More information on the CarMaker’s Simulink blockset can be found in section ’Features of the CarMaker for Simulink blockset’ on page 118.

With this block the user can access all User Accessible Quantities offered by CarMaker.

Take a multiplication block and link the engine speed signal “PT.Engine.rotv“ to one entry.

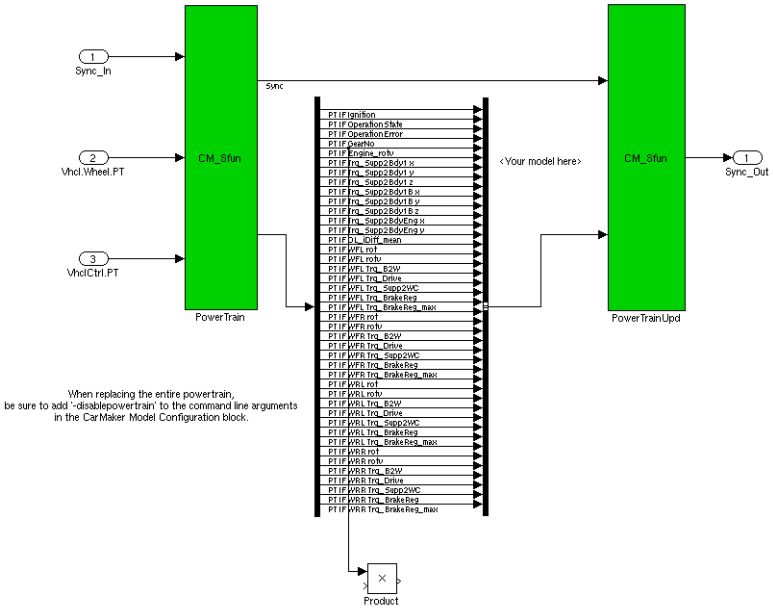


Figure 9.5: Multiplication Block

Now, a signal can be read from CarMaker in order to receive the engine torque. Since the engine torque is not directly available as a Simulink signal, the same values have to be gained from CarMaker, as are available in IPGControl.

Open the CarMaker for Simulink library (see section 'Features of the CarMaker for Simulink blockset' on page 118).

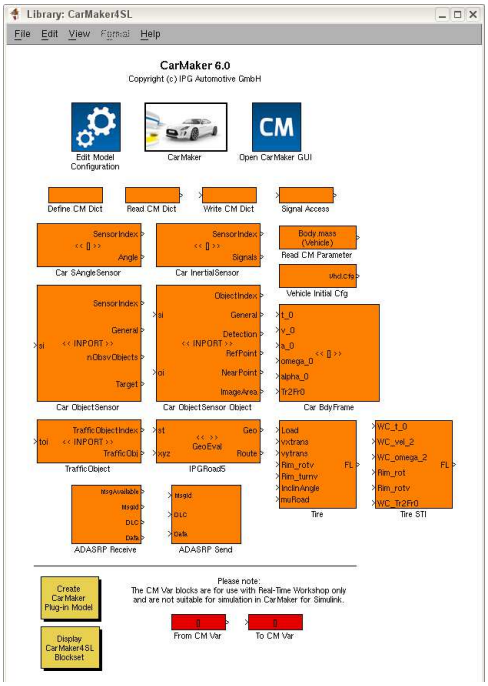


Figure 9.6: CarMaker for Simulink Library

Import a "Read CM Dict" block to the model, and define its property as "PT.Engine.Trq" in order to read the torque from CarMaker.

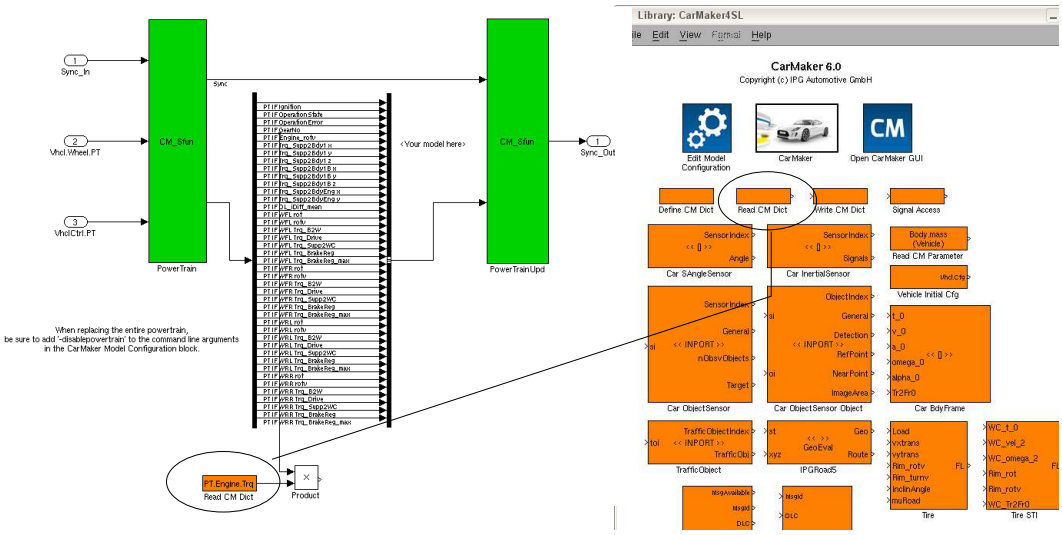


Figure 9.7: Read CM Dict

In CarMaker, the power is given in Watt. To monitor it in Hp, a gain block must be inserted for the conversion. The gain block is to have a value of 1/735,5. Now, define a new quantity to calculate the engine power and plot its value in IPGControl.

Add a “Define CM Dict” block to the model and define its property as “Engine.Power”.
Add a “Write CM Dict“ block to the model and define its property as “Engine.Power“:

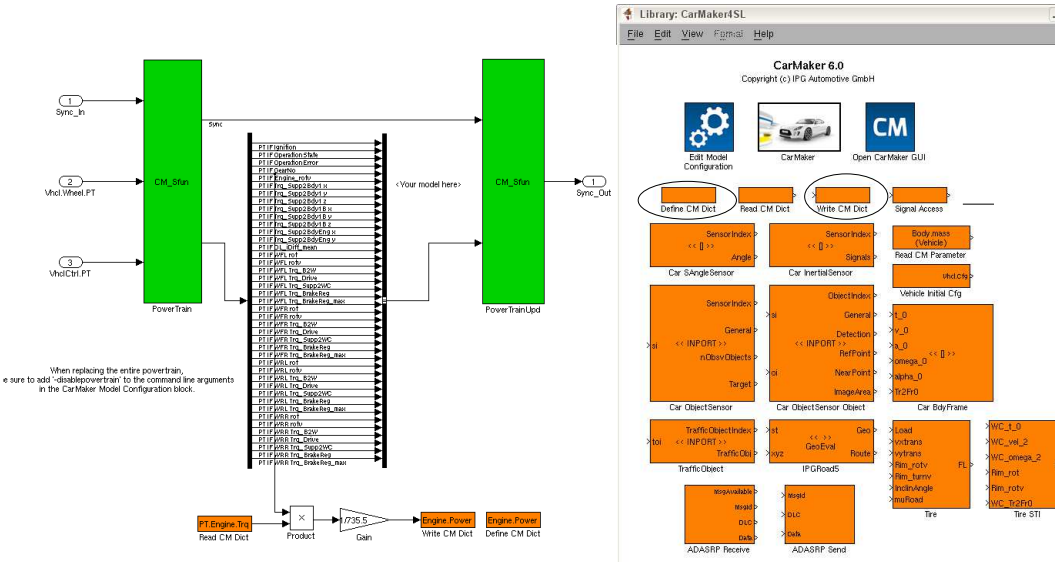


Figure 9.8: Engine Power

Now switch to the CarMaker GUI and load the TestRun *Product Examples* > *Examples* > *VehicleDynamics* > *_QuickStartGuide* > *Final_VehicleDynamics*. Open IPGControl and start the simulation.

The engine variable that was just created can now be observed in IPGControl.

9.2.5 Demonstration Examples

CarMaker comes with a number of example models demonstrating various applications, features and modeling techniques. All examples can be found in the src_cm4sl folder of the CarMaker project directory.

These examples include:

- The Simulink model
- Matlab parameter files and scripts
- Configuration files, e.g. TestRuns, vehicle data, tire data, etc.

A detailed description of all models and the special blocksets used can be found in the Programmer’s Guide, section "CarMaker for Simulink".

Please note: The examples intend to demonstrate how to make use of a particular interface of CarMaker for Simulink, but not to provide an elaborate application or a full fledged replacement for a particular CarMaker subsystem.

All examples can be used with the Matlab versions supported by CarMaker. An overview of these versions can be found in the CarMaker Release Notes (*CarMaker GUI* > *Help* > *Release Notes*).

Chapter 10

Additional Information About CarMaker

10.1 What are the Differences between each CarMaker Version?

First, a distinction must be made between the two general types of CarMaker versions:

- CarMaker/Office: Purely software simulations.
- CarMaker/HIL: CarMaker software adapted for Hardware In the Loop simulations.

CarMaker is a virtual vehicle environment. It can be used “as is“, without any additional software. This version of CarMaker applies to CarMaker Office, as well as CarMaker/HIL.

CarMaker can also be used in connection with Simulink: In this case, the CarMaker models are integrated in the Simulink environment. When the simulation is started, Simulink is the simulation software and uses the CarMaker library to perform the calculations. This version applies only to CarMaker Office.

Alternatively, extra models can be built using Simulink (in the CarMaker environment), by generating C-code correspondent to the model with the Real Time Workshop, and integrating the resulting library in CarMaker in the C level. This version applies to CarMaker Office and CarMaker/HIL.

10.2 Typical Tests

The following table is a list of the most important pre-defined TestRuns that display other capabilities of CarMaker. These can be found in the Product Examples.

It is advised to load TestRuns that seem interesting and helpful to see and understand how they are built. The column labelled *path* indicates in which sub-directory a TestRun can be found:

Topic	Path	Comments
Real Tests	./Examples/VehicleDynamics/	These TestRuns show you how to build common vehicle dynamic tests including slalom, lane change and steady circle tests. Pay attention to the definition of IPGDriver, especially to the maximum values of longitudinal and lateral acceleration and to the road definition.
Real Tests	./Examples/VehicleDynamics/StabilityControl/	These tests show the influence of control systems such as ABS or ESP. Pay attention to the road definition "*Override Attributes* for each road segment), and the maneuver definition (manual control of the pedals).
Using real measurements	./Examples/BasicFunctions/Maneuvers/RecordAndReplay/	Usage of real measurements. Pay attention to the following menu: *CarMaker GUI* > *Parameters* > *Input from File*. You can check the content of the input file by clicking on *Content* (Box Input File).
Drive Cycles	./Examples/Powertrain/DrivingCycles/	These tests include popular drive cycles, such as FTP, NEDC short and NEDC long. Pay attention to the maneuver definition (Input From File) and the vehicle's powertrain configuration.
Traffic Environment	./Examples/BasicFunctions/Traffic/	Show how to simulate traffic objects and define an overtaking maneuver. Pay attention to the traffic (*Parameters* > *Traffic*), and maneuver definition.
Driver Assistance Systems	./Examples/DriverAssistance/	These tests show special assistance systems: ACC, lane keeping and parking assistance systems. Pay attention to the sensor definition (*Parameters* > *Vehicle* > *Sensors*).
Vehicle features	./Examples/BasicFunctions/DVA/InteractWithModelInterfaces/ ./Examples/BasicFunctions/VehicleModel/Steering/ ./Examples/BasicFunctions/Sensors/	These tests present some of the special features of CarMaker's vehicle model, like flexible body, Pfeffer steering model and sensor definitions. Pay attention to the vehicle parameterization (*Parameters* > *Vehicle*).
Driving as usual	./Examples/BasicFunctions/Driver/	Show how to simulate reversing, turning on intersections, race conditions, etc. Pay attention to the settings in IPGDriver.
Road features	./Examples/BasicFunctions/Road/	Show how to realize comprehensive road configurations including real measurements, 3D tracks and special bumps and markers.
Simulation specific functions	./Examples/BasicFunctions/Maneuvers/MinimaneuverCommands/ ./Examples/BasicFunctions/Maneuvers/SpecialManeuvers/ ./Examples/BasicFunctions/DVA/ ./Examples/BasicFunctions/RTExpressions/	Special maneuver definitions including the minimaneuver command language, DVA commands and RTExpressions. Pay attention to the Minimaneuver Commands in the maneuver definition.
Testautomation	./Examples/BasicFunctions/TestAutomation/TestManager/	Using the TestManager functionality to create a test series including parameter variations, criteria evaluation, etc. See *CarMaker GUI* > *Simulation* > *TestManager*.

10.3 Feature List

DVA: How to Manipulate the Variables Online

CarMaker allows the user to check the values of specific variables online, e.g. to plot a diagram directly during the simulation. These variables are called *User Accessible Quantities*.

The UAQs are useful, not only to be *read*, but also to be *written* during the simulation: This is called *Direct Variable Access*.

It's a unique and powerful feature of CarMaker that allows influencing the simulation online, e.g. to trigger exceptional actions, such as an accident or a failure in the vehicle that blocks the steering system.

The DVA functionality is also programmable.

The DVA GUI is accessed via *CarMaker GUI > Application > Direct Variable Access.*

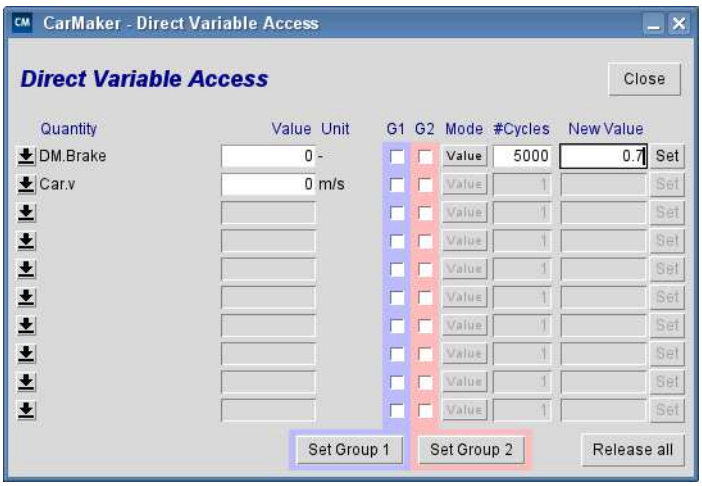


Figure 10.1: DVA window

Model Check: How to Check the Vehicle Parameterization

ModelCheck is a tool used to plot diagrams from the CarMaker TestRun definition that are standard in the automotive domain.

The ModelCheck GUI is accessed through the *CarMaker GUI > Simulation > ModelCheck*.

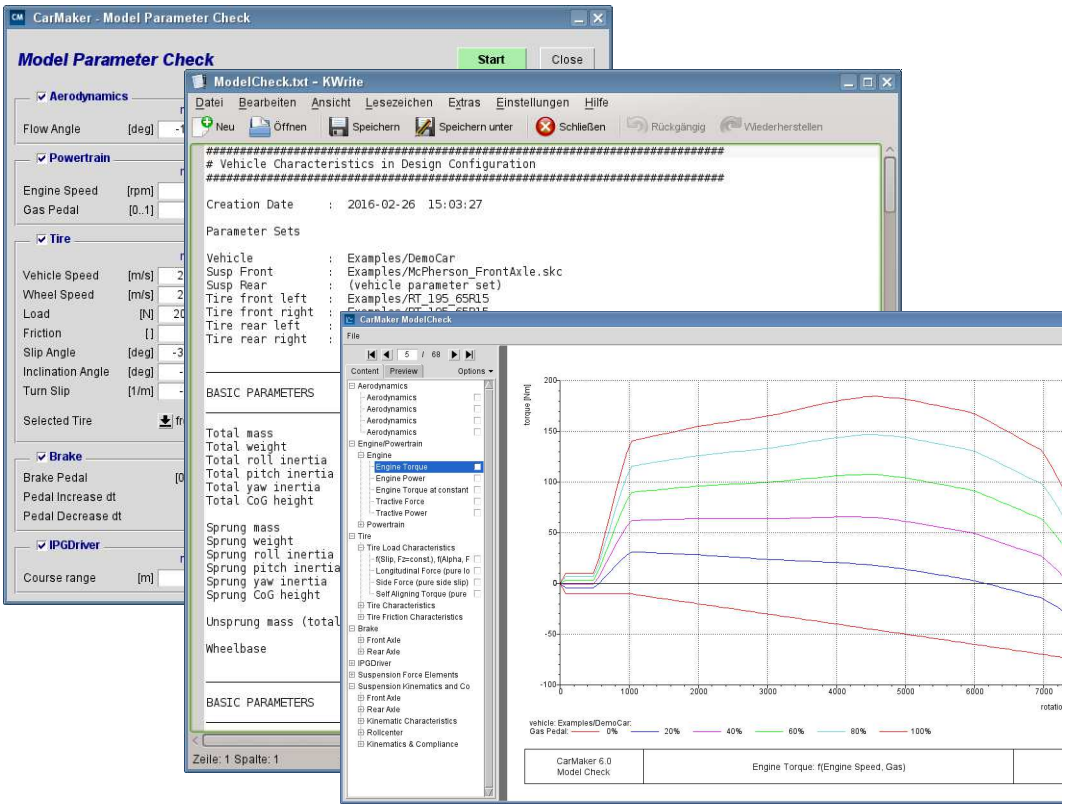


Figure 10.2: ModelCheck GUI

Test Automation

CarMaker offers many solutions for simulation test automation. The descriptions of these tools are documented in the User's Guide and Reference Manual.

- *Test Manager*

The Test Manager functionality enables the user to define a series of TestRuns, where CarMaker automatically starts the next TestRun when one is finished. If the *Storage of Results* option is set to *Save all,* a series of tests can be started and left alone, so that the user doesn't need to do anything and can later come back to the finished simulations and analyze the results right away.

Variations of quantities and parameters in a TestRun are easily conductible and it is possible for the user to integrate individual scripts for controlling the execution of the TestRuns.

- *Script Control*

ScriptControl, next to other functionalities, mainly enables automation of all actions that are usually manually conducted via the GUI. As it is based on a script language (tcl/tk), closed loops can be defined to load tests, save results or define counters.

- *DDE*

ScriptControl can also be controlled by DDE channels.

- *Matlab*

Last but not least, SciptControl can be controlled via Matlab.

IPGDriver: Driver Model

- *Standard Driver*

The standard driver model is the one that has been used during the previous chapters. It is much more than a simple speed and steering wheel controller.

The driver can shift gears, decide when it is reasonable to shift sooner in order to have more power at the end of a curve and looks ahead to predict speed and course.

- *Race Driver*

The race driver teaches himself the limits of the vehicle in order to optimize its trajectory by driving multiple courses of a specific Testrun with a specific vehicle. Each lap he increases his maximum lateral and longitudinal accelerations and saves them as *Driver Knowledge*.

Input From File: Integration of Measurements to the Simulation

This feature enables the definition of the maneuver based on measurements that the user has performed.

This can be very useful for comparing simulation results with measurements. The input for the simulation can be directly transferred from that of the real measurement, this way the boundary conditions are similar, and the results can be easily compared.

As an example, see the TestRun *Product Examples* > *Examples* > *BasicFunctions* > *Maneuvers* > *RecordAndReplay* > *MeasurementReplay* where a braking maneuver is analyzed. The measured positions of the pedals during the maneuver are used as input for the simulation and then braking distance, longitudinal acceleration and a few more quantities of the simulation results can be compared with the data from the measurement.

Obstacles on the Road Surface

The CarMaker road model features a set of obstacles that help alter its surface:

- Beams
- Waves
- Cones
- Friction stripes
- Mesh
- CRG
- Lateral profiles

Examples to these obstacles can be found under *Product Examples* > *Examples* > *BasicFunctions* > *Road.*

Simulation of Wind Effects

It is also possible to simulate wind effects. Find the appending example under *Product Examples* > *Examples* > *BasicFunctions* > *Driver* >*SteeringByTorque*.

Traffic Simulation

It is possible to simulate the presence of additional vehicles on the road, in both directions.

Furthermore the driver can be influenced to slow down due to velocity signs that are also displayed in IPGMovie.

See Examples to the topic traffic environment in *Product Examples* > *Examples* > *BasicFunctions* >*Traffic.*

Secured Files: How to Exchange Data Sets Safely

Using CarMaker, data sets can be converted into binary files. Other users will not be able to read the values defined in the file, but use them for simulations with CarMaker.

IPGControl

IPGControl is a tool used to plot diagrams of any of the UAQ (see definition in *Product Example* > *Examples* > *BasicFunctions* > *DVA*) online.

IPGControl also works offline.

IPGMovie

IPGMovie is CarMaker's online animation tool.

IPGMovie also works offline.

Online Capabilities

It is always possible to online check how the simulation is running.

E.g. if the animation in IPGMovie reveals that a mistake has been made while choosing the vehicle, the simulation can be immediately aborted, the correct vehicle or TestRun can be loaded and the simulation is ready to start again. The user does not need to wait for the end of the simulation in order to make changes in case there has been an error.

3D road

The CarMaker road model is three dimensional.

The user can create own roads without having data prior to simulation, thanks to the Scenario Editor that allows free design of road networks and surroundings. On the other hand, if the user does have data regarding the road, it can be used to model the course in CarMaker. This allows the user to perform simulations in reality and the virtual CarMaker environment simultaneously for comparison.

Integration of External Models

CarMaker delivers several interfaces that help integrate external models:

- Simulink: In Simulink, CarMaker models can be modified and it can be used to read and write values.
- C-code: The user can program individual codes, or interfaces with his own tools.
- AVL Cruise
- Sherpa Engineering thermodynamical engine model

Post processing

- Besides the IPG specific ERG result file format, CarMaker can generate output data files in MDF and ASCII format, too.
- MATLAB: CarMaker has an interface to MATLAB which helps analyze the results. You can load the CarMaker results in Matlab.
- EXCEL: Results can be exported directly from IPGControl to an Excel sheet.

Documentation of CarMaker

Further information on CarMaker and it's functions, please refer to the other documentations User's Guide, Reference Manual and Programmer's Guide.