IPG AUTOMOTIVE

Quick Start Guide Version 6.0

CarMaker®

SOLUTIONS FOR VIRTUAL TEST DRIVING

Contents

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 to be logged into an account with administrative privileges and extract the downloaded installation package (.zip archive). The executable file *ipg-install.exe* can be opened by double-clicking the mouse.

In case the user has received an installation CD, it needs to be inserted to the CD-ROM drive on 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 used needs to contact the *IPG License Team* in order to receive a valid license. Then the following steps are carried out:

- Within the Windows Explorer, the folder ./tools-win32 of the installation package needs to be selected.
- In this folder, a tool called *wlicinfo.exe.* can be started:

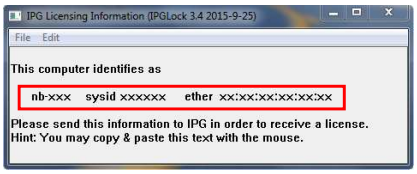


Figure 2.1: Executing the wlicinfo.exe and required output

- To apply for a license, a license key request form can be found in the support area of the IPG-homepage (http://www.ipg-automotive.com/). The form needs to be filled out and the output provided by wlicinfo.exe must be copied to the “PC identification“ field.
- Subsequently, the user will receive a valid license file from the IPG License Team.
- The license file is saved under <*InstallationDirectory*>*etc*.
- Now, the software is activated and ready to use.

Chapter 3

Starting CarMaker

Starting CarMaker in Windows: *“Start“ button > 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 menu has been selected, the CarMaker software opens.

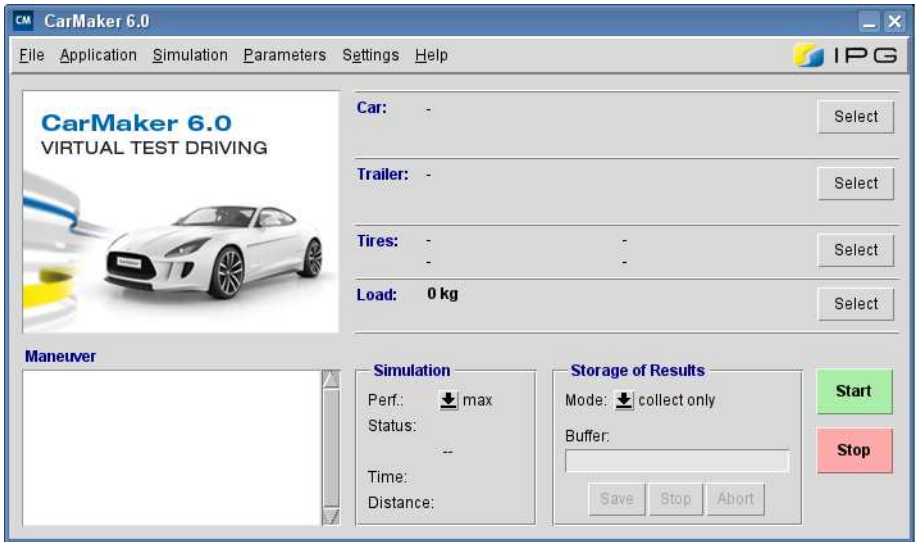


Figure 3.2: CarMaker main GUI

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

Now, CarMaker is ready to work and can be used right away, with no further preparation necessary. To find out how to continue, please refer to the entire section 'Performing your first Simulation within five Minutes' on page 11.

Chapter 4

Performing your first Simulation within five Minutes

4.1 Loading a predefined environment parameterization

Performing a simulation in CarMaker requires similar defintions 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 a predefined model for each of these requirements. A combination of these models and settings form what is called a *TestRun*.

The standard installation of CarMaker includes a variety of 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.

When starting CarMaker for the first time, a project folder, that will contain all the TestRun data, needs to be created. This is done ny selecting *File* > *Project Folder* > *Create Project* in the CarMaker main GUI.

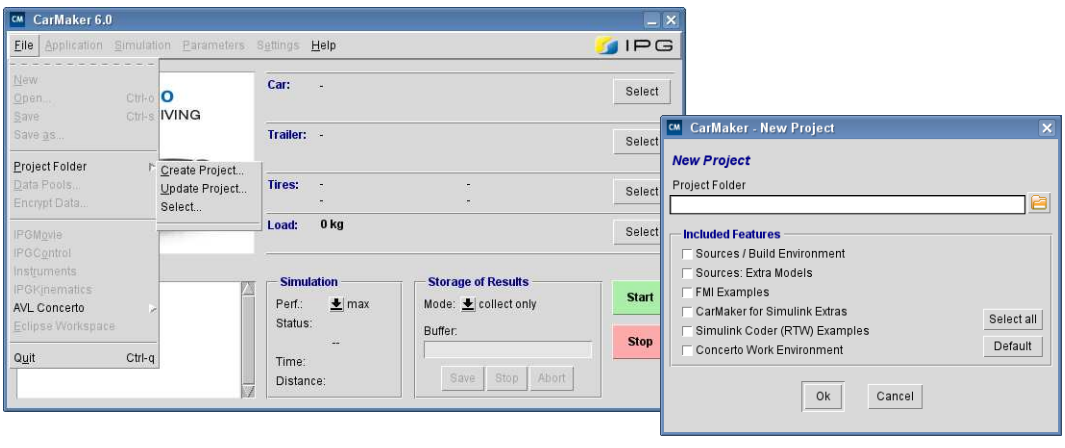


Figure 4.1: Creating a new project folder

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. In this tutorial, none of the available features need to be selected

Now, the first TestRun is ready to be loaded.

Loading TestRun “SegmentBasedClosedTrack”:
In the CarMaker GUI, click on *File > Open,* select the product data pool, then *Examples > BasicFunctions > Road > ArtificialRoads > SegmentBasedClosedTrack*

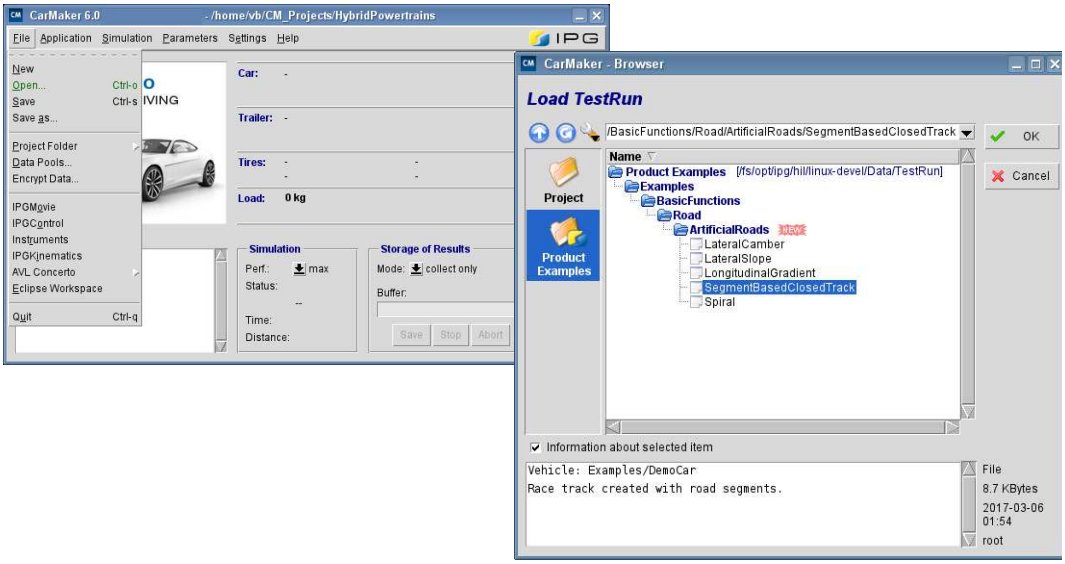


Figure 4.2: Loading the TestRun SegmentBasedClosedTrack

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

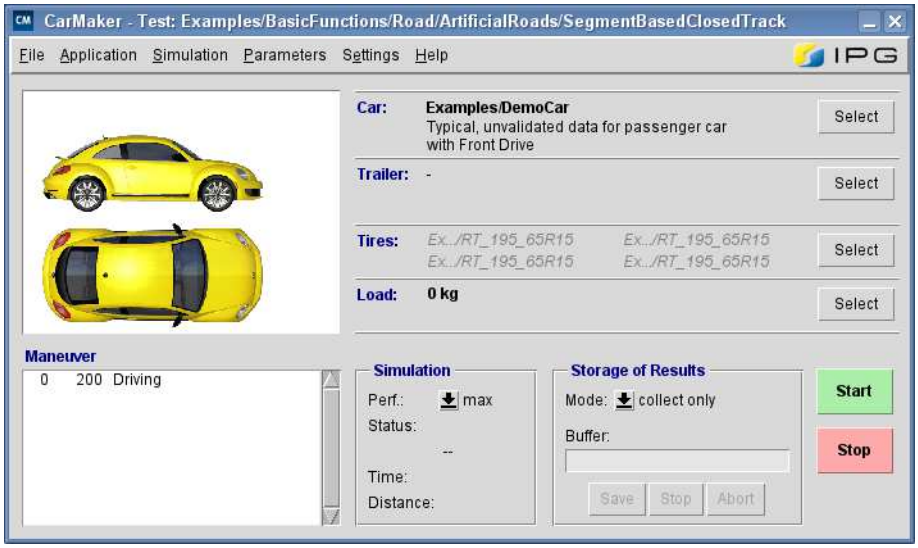


Figure 4.3: The CarMaker GUI containing all data sets for the TestRun

What is a TestRun?

CarMaker is based on fixed models (vehicle, suspension, tire, 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 between them are already defined and the user doesn't need to do so on their own. If the model is desired to be extended, CM4SL is recommended. For further information on this topic, please refer to section 'CarMaker for Simulink' on page 90.

The models may already be defined but they still need to be parameterized according to their environment. This means that a so called *data set* must be manually implemented or loaded for each model. Parameterizing the models includes selecting a vehicle, selecting or designing a road, defining a type of driver and defining the maneuver. Only after all these components have been set, does the used have all all information necessary to control the *virtual vehicle environment (VVE)*.

These settings are stored in a file used by the VVE during simulation. Said file, which can be saved, loaded or edited is called a *TestRun definition*. Loading and executing this file results in the simulation of that specific test.

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

4.2 Starting and Stopping the Simulation

In section 4.1 on page 11, a TestRun is already loaded. This means that CarMaker has already been provided all the data necessary for successfully performing a simulation.

To start the simulation: Click on the green "Start" button in the CarMaker GUI.

The simulation has begun. This is recognizable for various reasons:

- In IPGMovie: The animation movie of the current simulation is running and can be viewed by the user. After the simulation replay in different speeds is possible.
- In the CarMaker GUI: In the box labelled *Simulation* a timer and indicator showing the distance covered by the car, as well as the current status of the simulation are displayed
- In the 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 ba clicking "File > IPGMovie" and "File > Instruments" in the main GUI.

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

To 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.

Loading a TestRun and clicking “Start“ and “Stop“ are the basics of CarMaker.

Now, the following chapters will describe what happens while the simulation is running in more detail.

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

Now, the user can observe the CarMaker GUI, IPGMovie, the Instruments window, as well as IPGControl, while the simulation is running.

4.3 Interactive View of the Current Simulation

IPGMovie

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

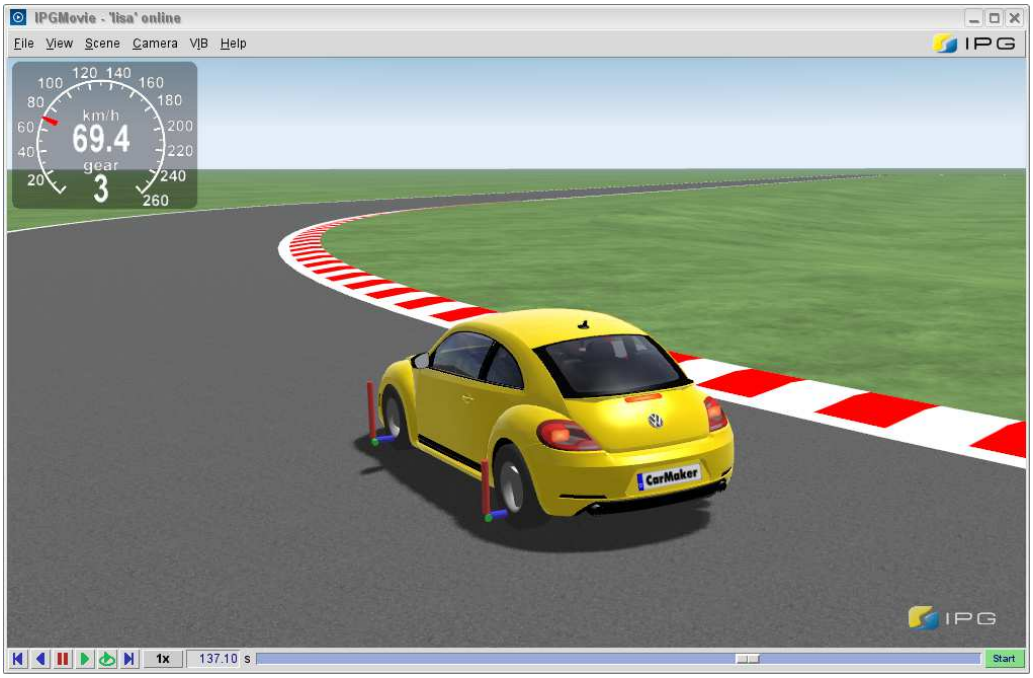


Figure 4.5: Online View of a simulation using IPGMovie

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 expoirted to a file for further use in presentations or advertising.

A more detailed explanation on using IPGMovie is given insection 4.4 ’IPGMovie’ on page 19.

CarMaker GUI

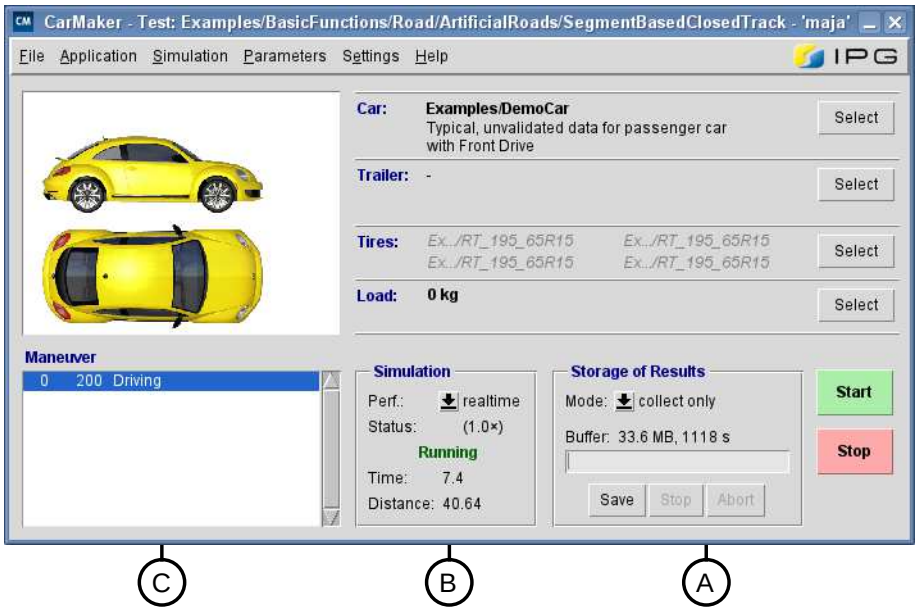


Figure 4.6: The CarMaker GUI

- Box (A): Storage of Results

The results generated by CarMaker are usually saved to a buffer in the computer memory. They can, however, be saved to a file as well, for further analyzation. This box helps control and define the saving strategy.

- Box (B): Simulation

In this box the user can define the speed of the simulation: real time, slower or faster than real time. The “max“ option enables simulation speeds that are as fast as the current CPU will allow. This can be up to 40 times faster than real time.

Speeding up the simulation saves time, especially when working with automated TestRuns where often a very large number is carried out in sequence. another feautre 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 (I*dle* 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 5.2 ’Defining the Maneuver’ on page 42, several maneuver steps can be defined. They are all shown in this box. 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. The content of Instruments is very similar to what can be seen in the cockpit of a real vehicle!

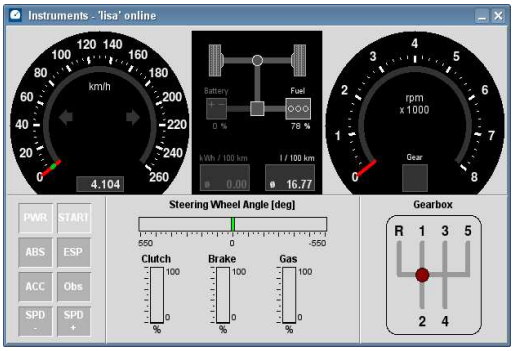


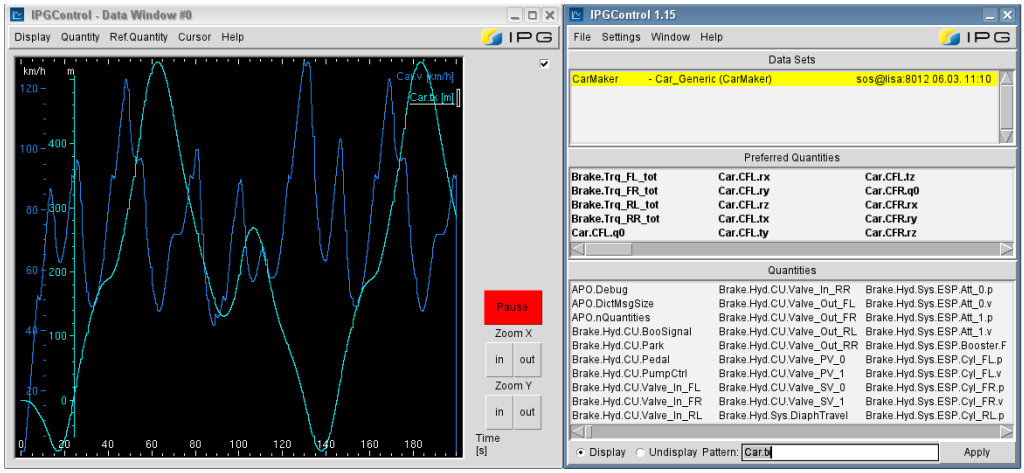
Figure 4.7: The Instruments display

The content of Instruments can optionally be 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.

IPGControl

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

Open IPGControl: *File > IPGControl*



IPGControl Data Window IPGControl Selection Window

Figure 4.8: IPGControl Data and Selection Window

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

In the CarMaker GUI, start the simulation.

After a while, click “Pause“ in the IPGControl Data Window, right-click in the middle of the window and select “Total fit“ in order to fit the diagram within the available space.

IPGControl usage is explained in further detail in section 4.5 ’Result Management’ on page 21.

4.4 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 already been carried out can be shwon 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.

Changing 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 during the simulation

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 selects 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 simulatneously in IPGMovie, allowing quick, visual analyzation of the results. This feature is explained in further detail in section 5.3.3 ’Comparing two Simulations in IPGMovie’ on page 53.
- A variety of options that are 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, a personalized camera view 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 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.

All modifications that are made to the road definition will be displayed in IPGMovie. However, the animation needs to be updated first. Therefor, the simulation needs to be started and stopped once. In section 5.1 ’Exploring the IPGRoad Features’ on page 27 the user will learn how define a road.

4.5 Result Management

4.5.1 IPGControl

This section will explain how to analyze and plot the 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 a previous TestRun. This is called *offline result management*. This new source of results is displayed in the Data Sets list in the selection window. The data saved in this file 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 axis so that the results can be analyzed more efficiently and effectively.

Opening IPGControl

IPGControl is opened via the *File* menu in the CarMaker GUI. Both the IPGControl selection and IPGControl Data Window appear.

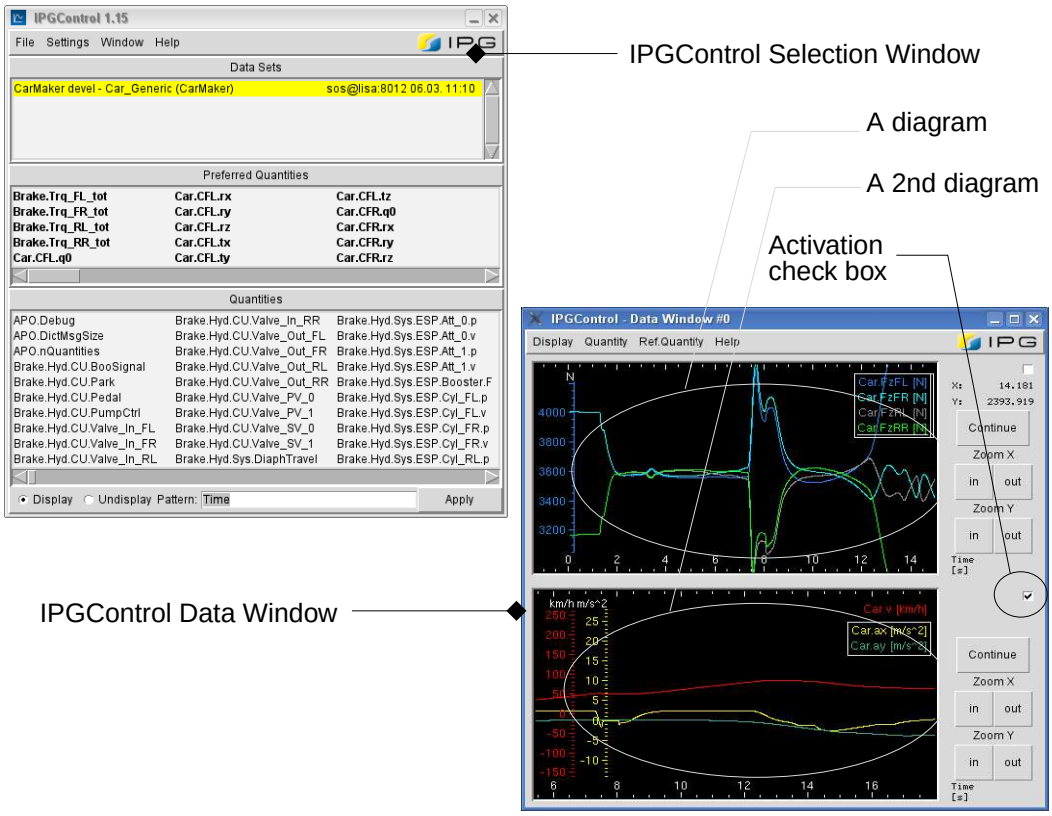


Figure 4.10: IPGControl Selection IPGControl Data Window

Using the Selection Window, the result files can be selected that are to be displayed in the Data Window.

Selecting the Quantities to be plotted

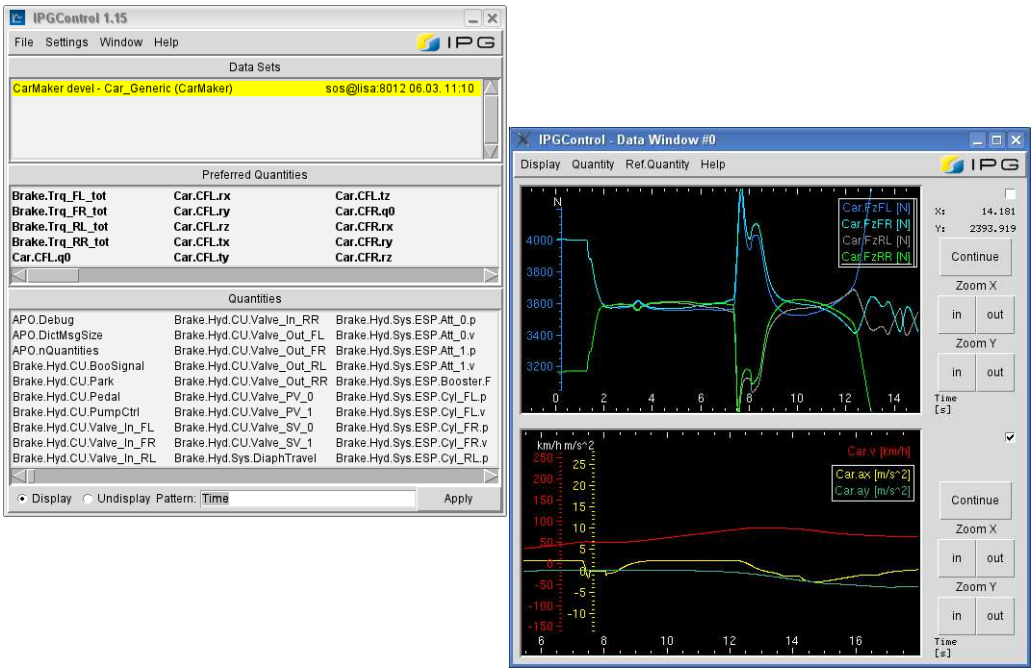


Figure 4.11: Quantities to be plotted

First and foremost, if more than one diagram are present in the Data Window, one has to be activated in order to define quantities to be plotted. Therefor, the desired diagram is left-clicked once. The activation check box informs the user about which diagram is currently active.

The quantity for the x-axis is selected in the selection window (field “Quantities” or “Preferred Quantities“) using the middle mouse button, the quantities on the y-axis are selected using the left one. Alternatively, the menus “Ref.Quantitiy“ (quantity for the x-axis) and “Quantity“ (quantities for the y-axis) of the data window can be used.

Quantities can be removed from the diagram by reselecting them in the previously mentioned lists (Quantity and Ref. Quantity), by *undisplaying* them in the menu that opens after a right-click in the graphical window or by double-clicking the quantity in the list on the right in the graphical window.

A Quantity in IPGControl can also be referred to as a *User Accessible Quantity (UAQ)*. For an explanation of each of the UAQ available for plotting in IPGControl, please refer to the Reference Manual, section “User Accessible Quantities”.

Functionalities of the Data Windows

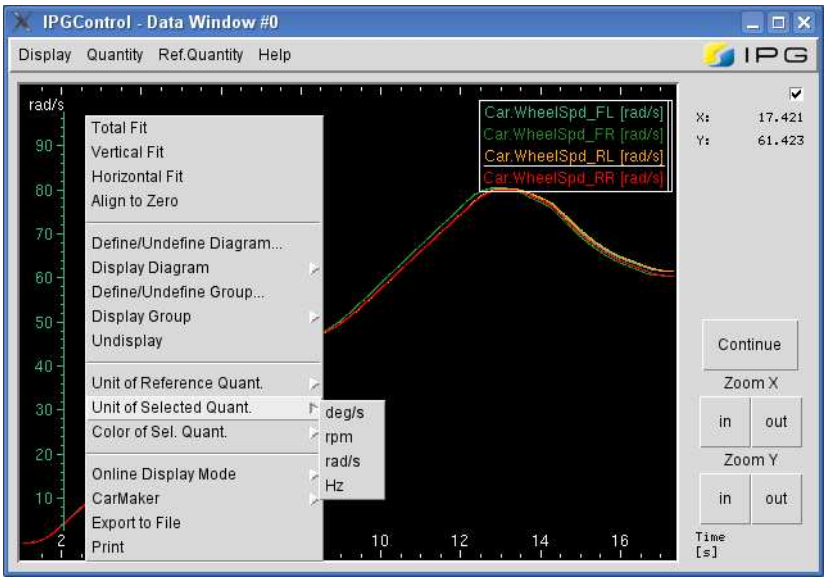


Figure 4.12: Data Window functionalities

- 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 x and Shift-x: Changes the scale of the x-axis.
  - Pressing y and Shift-y: Changes the scale of all y-axes. To scale only one y-axis, the desired curve needs to be selected first.
  - Pressing the up and down keys: Changes the limits of all y-axes. To limit only one y-axis, the desired curve needs to be selected first.
  - Pressing Shift and the up or down keys: Changes the limits of all y-axes with a smaller step size. To limit only one y-axis, the desired curve needs to be selected first.
  - Pressing PageUp and PageDown: Changes the limits of the x-axis.
  - Pressing Home or End: Changes the limits of the x-axis so that the entire range of data is displayed.
  - Pressing Delete or Backspace: Removes the selected quantities from the diagram.
  - 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: Adds to orremoves it from the list of selected quantities.
  - 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 > Color / Unit of Selected Quantity: changes the color or the unit of the selected quantity.
- Rightclicking on a selected quantity > Unit of Ref.Quantity: changes the unit of the reference quantity.
- Right-clicking in the middle of the diagram > Define / Undefine Diagram: Remembers the quantities currently displayed in order to use the same set of quantities in future work (option “Display Diagram“). These settings must be saved (selection window > Settings > Save Settings).
- Right-clicking in the middle of the diagram > Define / Undefine Group: Remembers the quantities currently displayed in order to use the same set of quantities in future work (option “Display Diagram“). These settings must be saved (selection *window* > *Settings* > *Save Settings*). The difference to the option “Save Diagram“ is that several groups of quantities can be displayed in the same diagram. Using “Display Diagram“, the selected quantities will be removed.
- Selecting the top menu *Display* > *Add/Remove Diagram*: adds or removes a diagram in the data window.

Using the Selection Window

In the Selection Window the user can choose the source from which the data is displayed (field "Data Sets"). This can be either the current simulation or loaded result files from previous simulations.

Furthermore, the sample rate of the data can be defined under *Settings* > *Online Sample Rate*.

With the option *Window* > *New Data Window*, more Data Windows can be added to the display.

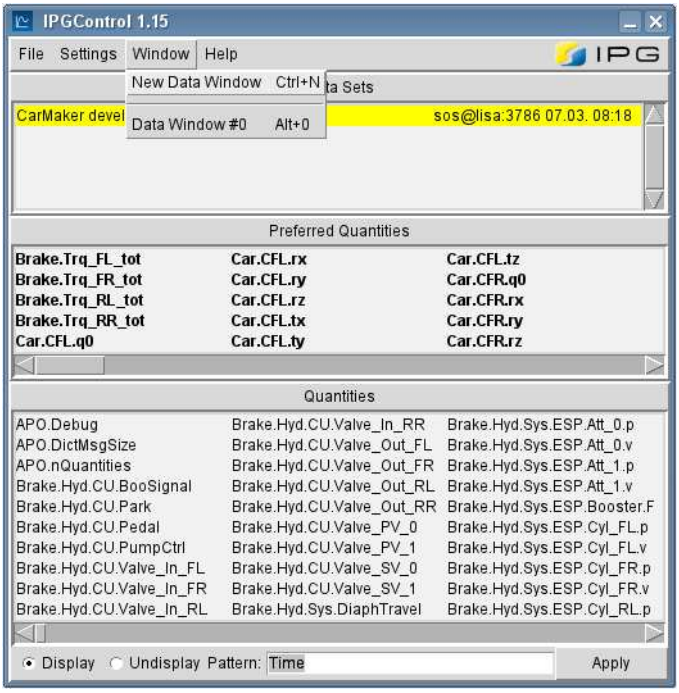


Figure 4.13: Adding Additional Data Windows

The field "Preferred Quantities" contains only the most frequently used UAQs, while the field “Quantities“ contains over 600 other available UAQs.

- Key shortcuts available in the “Quantities“ and “Preferred Quantities“ fields:
  - Left-click: Displays/Undisplays the quantity in the active diagram.
  - Shift and left-click: Displays/Undisplays the quantity in the active diagram, giving it its own y-axis.
  - Middle-click: Defined a quantity as the reference quantity of the active diagram.
  - Ctrl and left-click: Adds/Removes the quantity to/from the list of preferred quantities.
  - Bold font: The quantity data is monotone (increasing values).
  - Red font: The Quantity is the reference quantity of the active diagram, for e.g. the quantity represented on the x-axis.
  - Yellow background: The quantity is displayed in the active diagram.

Exercise: Steps below.

Open the CarMaker GUI and load the TestRun SteadyStateCircular42m:

- *CarMaker GUI > File > Open > Product Examples > Examples > VehicleDynamics > Handling > SteadyStateCircular42m*

Perform the first simulation:

- *CarMaker GUI > Start*

Open IPGControl. In the Data Window, add a diagram:

- *Data Window > Display > Add diagram*

Select the upper diagram:

- Click once in the middle of the diagram.

In the selected diagram, select *time* for the x-axis and v*ehicle speed* and *wheel speeds* for the y-axis:

- *Data Window > Ref. Quantity > Time*
- *Data Window > Quantity > Car[S..Y] > Car.v*
- *Selection Window > Pattern (field (A) at the bottom of the window) > write *WheelSpd* > Select “Display“ > click “Apply*”

Select the lower diagram:

- click once in the middle of the diagram.

Select lateral acceleration for the x-axis and steering angle for the y-axis:

- *Data Window > Ref. Quantity > Car [A..G] > Car.ay*
- *Data Window > Quantity > Steer > Steer.WhlAng*

Start the simulation again:

- *CarMaker GUI > Start*

Now the curves are being plotted online. That 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.

4.5.2 Other Capabilities

In addition to IPGControl, CarMaker features an interface for two of the most important result management tools, Excel and Matlab.

Microsoft Excel

When saving simulation results to a file, CarMaker uses binary format. Afterwards, a tool called *resutil* can be used to convert the result file to ASCII format (text file), which can then be loaded directly into Excel.

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.

The Mathworks MATLAB

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

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

Next!

Now, the first simulation using a predefined TestRun has been executed and the online capabilities of the main tools have been tested.

The section 'Extending the Test Scenario Definition' on page 27 will explain how a predefined TestRun can be individually extended. This way the main part of the CarMaker virtual environment can be modified.

Chapter 5

Extending the Test Scenario Definition

5.1 Exploring the IPGRoad Features

Load a simple TestRun: In the CarMaker GUI, click on *File > Open,* select the product data pool, then click on *Product Examples > Examples > VehicleDynamics > _QuickStartGuide > Step1_DefineFirstScenario*.

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 "My_TestRuns". Double-click on the new folder and enter any name for the TestRun, for e.g.*"My_Step1" > OK.*

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

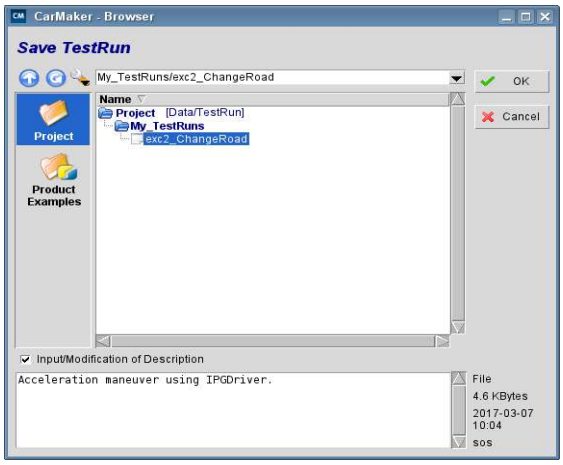


Figure 5.1: Saving the TestRun under a different name

This chapter will describe how to extend the TestRun Step1_DefineFirstScenario in order to build a closed circuit.

If CarMaker is closed before the end of this chapter, the work can be saved by clicking *File > Save* in the CarMaker main GUI or by using the key combination *Ctrl + S*.

5.1.1 Opening the Scenario Editor

The Scenario Editor is a Graphical User Interface used to parameterize the road and build road networks.

The IPG road model, called IPGRoad is a 3D model. The user can define the trajectory (middle line) of a track, as well as its width and the friction coefficient.

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

- Scenario Editor: Lets the user design individual road networks using IPG's intuitive tool.
- ASCII files: Implements roads into CarMaker by loading road descriptions written in ASCII format (text file). These need to contain cartesian (x, y, z) or geographic (long, lat) coordinates.
- KML-files (Keyhole Markup Language): These files are created by Google Earth or Google Maps and can be uploaded in CarMaker to produce specific road networks.

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

Open the Scenario Editor: In the CarMaker GUI, click on *Extras > Scenario Editor*.

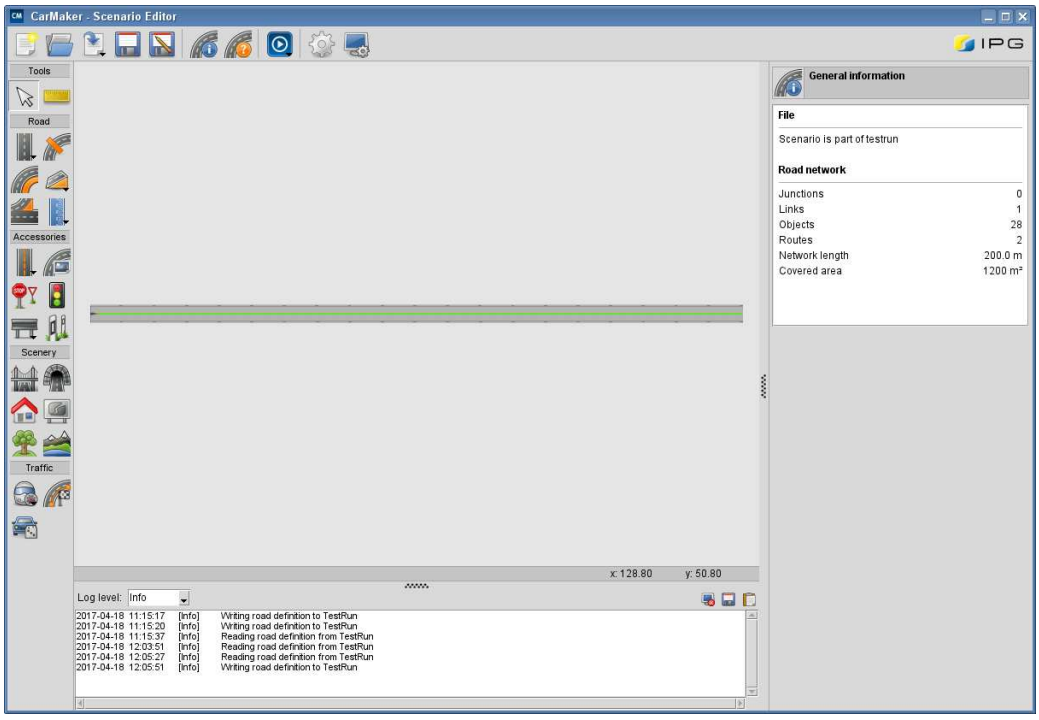


Figure 5.2: Scenario Editor

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

- 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.

  - Road: This tab is used to implement Roads (straight, curve, clothoid, etc.), Lane sections, Lanes, elevation profiles and road Bumps. 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 object can be stochastically 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 e.g.: 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).

5.1.2 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.

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".

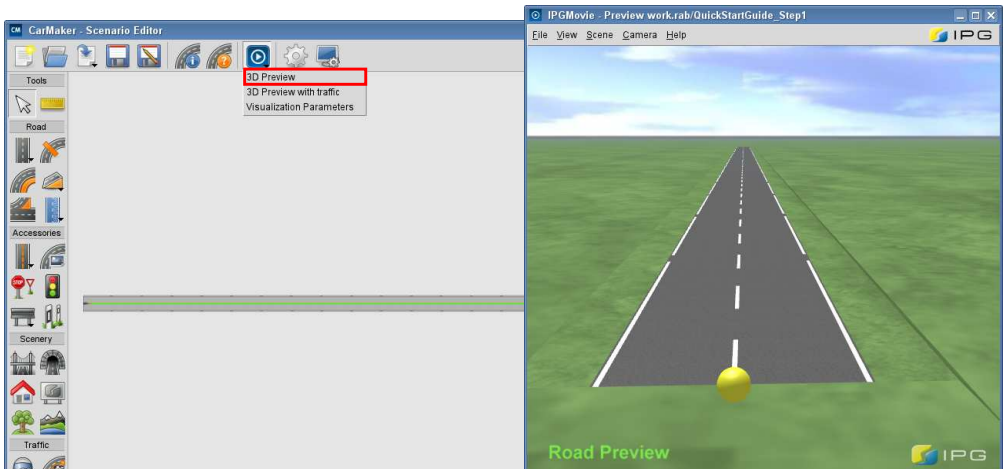


Figure 5.3: 2D-Preview in Scenario Editor and 3-D-Preview in IPGMovie

The display in the Sceanrio 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 the roads during simulation preparation.

IPGMovie shows the road definition in 3D and with 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.

At this point, the road in the loaded TestRun 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 and ready for viewing.

5.1.3 IPGRoad and Maneuver Definition Interaction

Now, two additional segments will be added to the previously loaded and saved TestRun.

In the "Road" tab on the left side of the Scenario Editor, click and hold the first icon (Road segment).

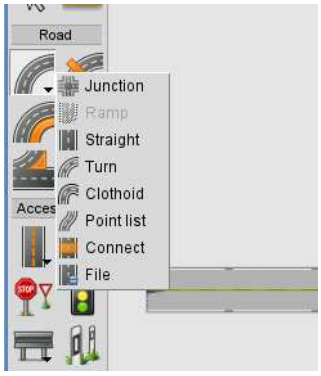


Figure 5.4: Inserting new road segments

Now a type of segment can be selected.

1. Release the mouse button over the "Turn" option, click once on the end of the straight road to define the beginning of the new section, form a left curve and click again to set the end of the segment. Now, set the curve radius to 50m and the angle to 90 degrees in the tag on the right.

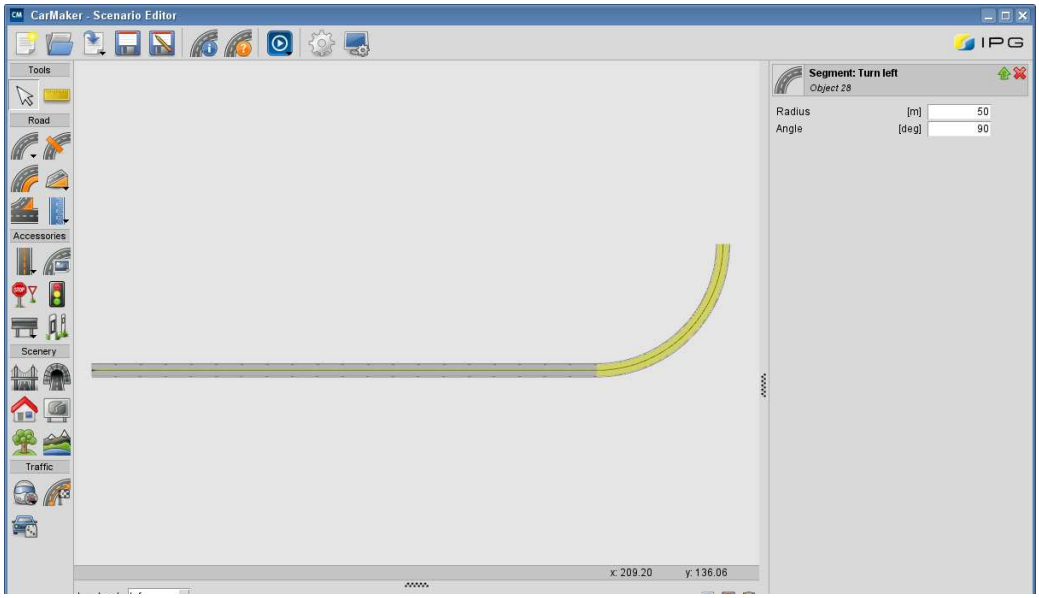


Figure 5.5: Defining the left curve

2. Repeat the previous actions, but this time form a right turn and edit the parameters so that the curve has a radius of 100m and angle of 180 degrees.

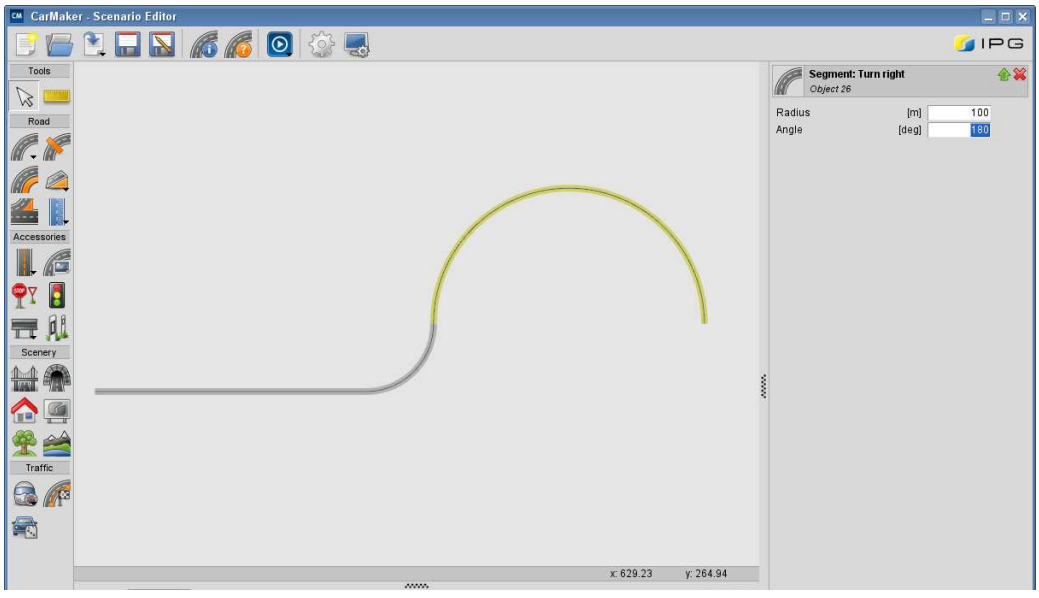


Figure 5.6: Defining the right curve

Check the road geometry in 3D-Preview using IPGMovie by pressing the button marked with the play-symbol again.

Figure 5.7: 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 newly added segments 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 doesn’t 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.

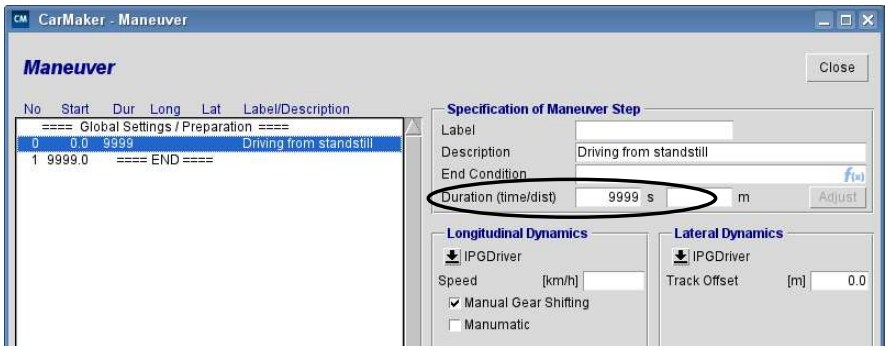


Figure 5.8: Changing the duration of the maneuver

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

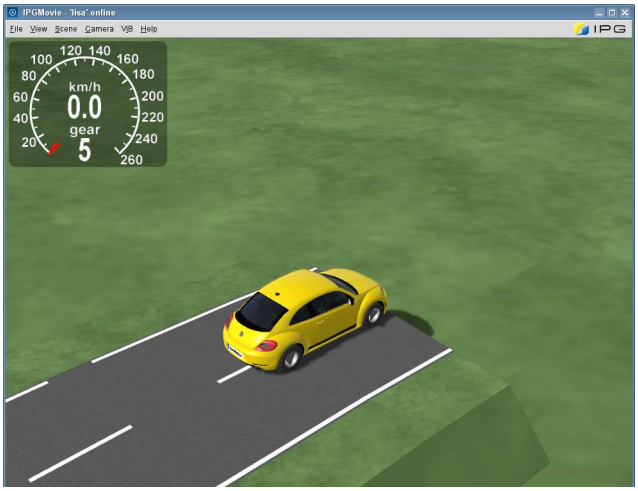


Figure 5.9: Vehicle at the end of the road

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 5.2 ’Defining the Maneuver’ on page 42.

5.1.4 Creating a Closed 3D Road

The previously built road will now be extended in order to build a closed 3D track. To load the previous TestRun *My_Step1*, click on *File* > *Open* > *My_Step1* > *OK* in the CarMaker main GUI. For better visualisation, IPGMovie should be opened, too.

For this example, seven new segments with the following geometry will be added to the previously built track:

- Segment number 3 (the number of the segment is displayed when the segment is selected in selection mode):

  - straight line
  - length: 100m
  - gradient: elevation to an end value of 10m (gradient = longitudinal inclination)
- Segment number 4:
  - straight line
  - length: 300m
- Segment number 5:
  - straight line
  - length: 100m
  - gradient: elevation to an end value of 0m
- Segment number 6:
  - right curve with radius: 100m
  - angle: 180 degrees
  - slope (lateral slope): 15%
- Segment number 7:
  - left curve with radius: 50m
  - angle: 90 degrees
- Segment number 8:
  - straight line
  - length: 200m
- Segment number 9:
  - right curve with radius: 200m
  - angle: 180 degrees

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

Segment 3: Click and hold the "Road segment" icon in the "Road" tab and select "Straight". Click once at the end of the current Track and again somewhere in the free space, in order to create a straight road segment. Adjust the parameter "length" in the right tab to 100m.

Segment 4 and segment 5: Repeat the steps given for segment 3 but note that the lengths of these segments are 300m and 100m.

Segment 6: Click and hold the "Road segment" icon in the "Road" tab and select "Turn". Click once at the end of the current Track and again somewhere in the free space, in order to create a right turn (in this case, a right turn is slightly left of the track because the driving distance always needs to be considered). Adjust the parameters "Radius" and "Angle" to 100m and 180 degrees.

Segment 7: Repet the steps given for segment 6 but note that the turn must go to the left, the radius is 50m and the angle is 90 degrees.

Segment 8 and segment 9: build these road segments according to the steps given for the previous ones.

The track should now have the course displayed in the figure below.

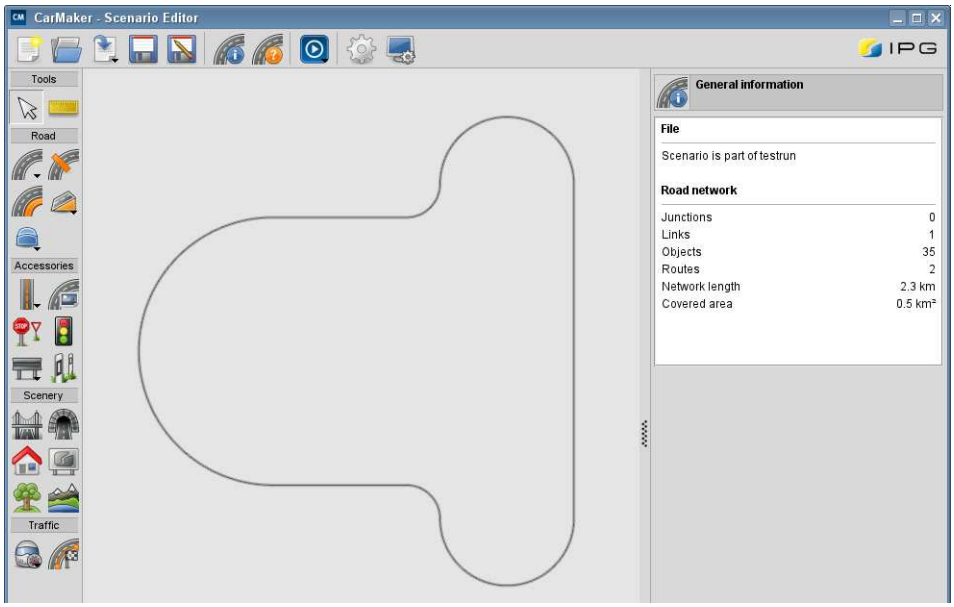


Figure 5.10: Course without elevation profile

Now, the different elevation profiles for specific lane sections are implemented. It is important to know that an entire road network can have only one elevation profile of each type (elevation, slope and camber). When the user has initiated a list for either of these parameters, trying to insert a new one would delete the current values. Therefore, when adding further elevations to the road, the user should expand the existing list instead of trying to add a new one.

Elevations are defined relative to the s-axis 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-axis coordinates: Beginning and end can be set exactly in the elevation profile, relative to the beginning of the correspionding Link.
- Intuition: Using the cursor the user can select points in the disply 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 are set intuitively using the cursor in the display window.
Inserting the elevation profile for sections 3 and 5: since we already know that segments 3 and 5 are to have an elevation profile, both can be inserted directly in one step.

Click and hold the "3D Surface" button in the "Road" tab. Choose "Elevation profile". Click on the relevant link and define the first point (aroung the beginning of section 3, after the curve). Select the second (at the beginning of section 4, after about 100m), the third (at the beginning of section 5, after about 300m) and the fourth (at the end of section 5, after about another 100m) point. The selection is ended with a double-click on the last point.
Now a list representing the elevation profile is visible in the right tab. Insert 0m for the height of the first point, 10m for the second and third and 0m again for the last point.
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:

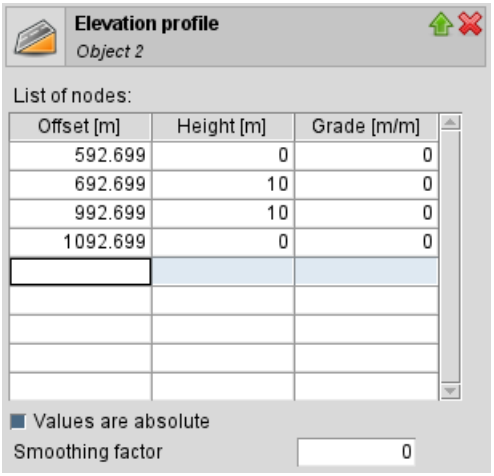


Figure 5.11: Elevation profile for the longitudinal slope

It doesn't matter if the points differ slightly since they were selected intuitively.

Now the lateral slope in section 6 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 and define four points around segment 6.
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 slope, shortly after point 3.
Now a list representing the slope profile is visible in the right tab. Insert 0.15 (for 15%) for the slope of the second and third point, and 0 for the rest. Also, set 0 for the gradient of all four points.

This is what the slope profile should look similar to:

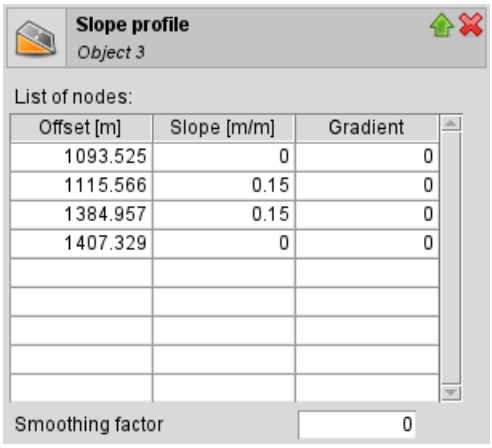


Figure 5.12: Slope profile for the longitudinal slope

Save the TestRun and start the simulation again.

In IPGMovie the road characteristics that were just defined are shown in detail, especially the longitudinal and lateral slopes.

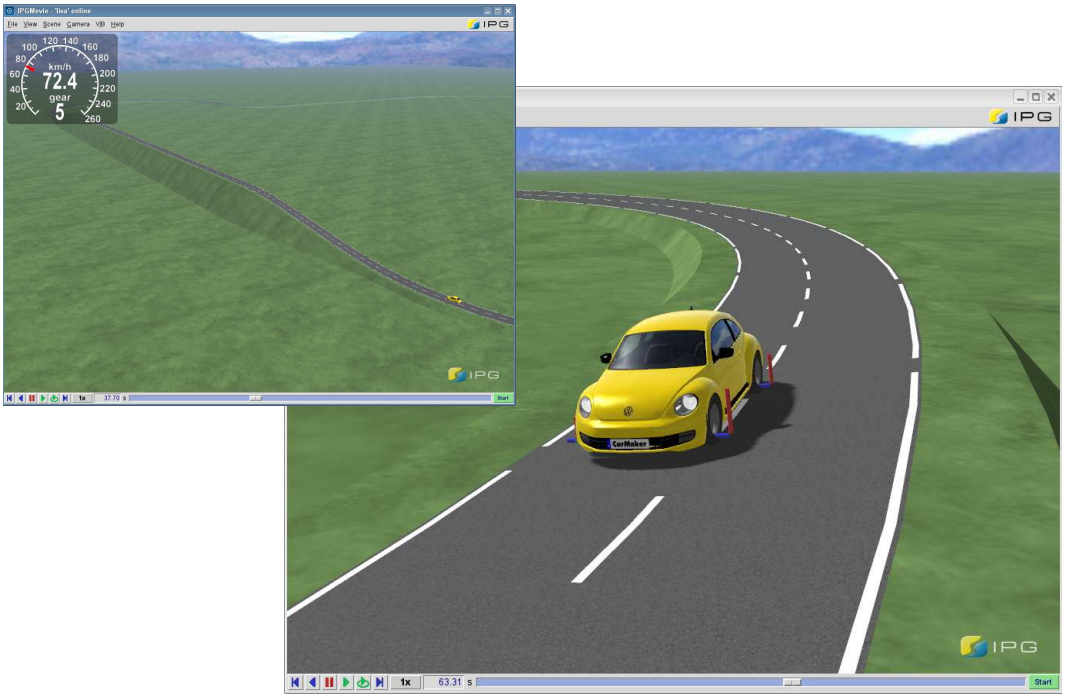


Figure 5.13: Longitudinal and lateral slopes

5.1.5 Modifying the Width and Coefficient of Friction

Up until now the designed roads have all had the coefficient of friction and road width that were predefined in the TestRun *Step1_DefineFirstScenario.* If desired, the user can easily adapt both parameters in the Scenario Editor. How the user must proceed to do so will be described in this chapter.

In this example, again, the saved TestRun *My_Step1* is used.

**Change the lane width: Click on the "Lane" icon in the "Road" tab and select the left lane (be aware which lane is left, depending on the driving direction). Set the lane width to 3m in both fields ("Width at start" and "Width at end") on the right.
Do the same with the right lane, but change the width to 4m.**

Start the simulation and watch how the track has been changed.

When the width of a lane is set, the value 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 set in the display window by clicking on the road at the desired point. Now, the lane width within this section can be varied independent of the other Lane Section lane widths in the known manner.

**Add a vehicle dynamics plattform by changing the lane width around segment 4 on the inclined straight:
First, a new Lane Section needs to be defined. As previously described, select the "Lane section" tool and define the beginning and end point so that the incline is limited within the Lane Section.
Now select the left lane in the new Lane Section and set the width for both "Width at start" and "Width at end" to 30m. Repeat this step for the right lane.**

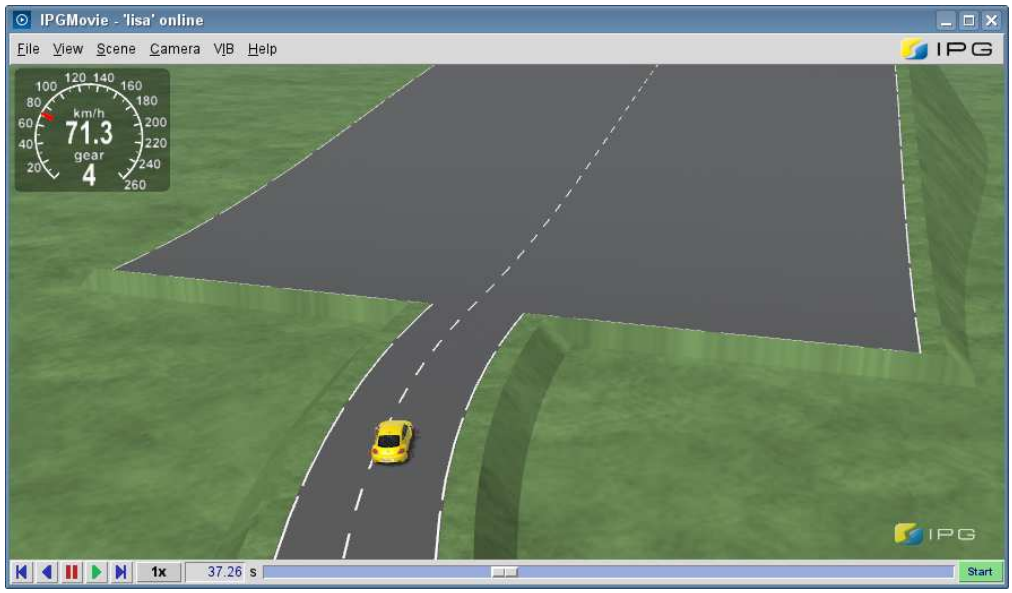


Figure 5.14: “Vehicle dynamics plattform

Values within a newly set Lane Section that don’t receive new values, keep the default values that are valid for the rest of the road. For instance, in the picture above, the width of the track on the left side of the new Lane Section was changed. However, the friction coefficient and margin width were left as is.

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 varibale 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 track: as previously described, select the "Bumps" tool and then the pick "Friction" option. Click on the Link and place the beginning point at s = 0m and the end point at s = 200m, so that segement 0 is now covered in a friction stripe.
In the tab on the right set the Lateral offset to 0.5m (positive value is the left side of the road), the Width to 1m and the Friction to 0.8.

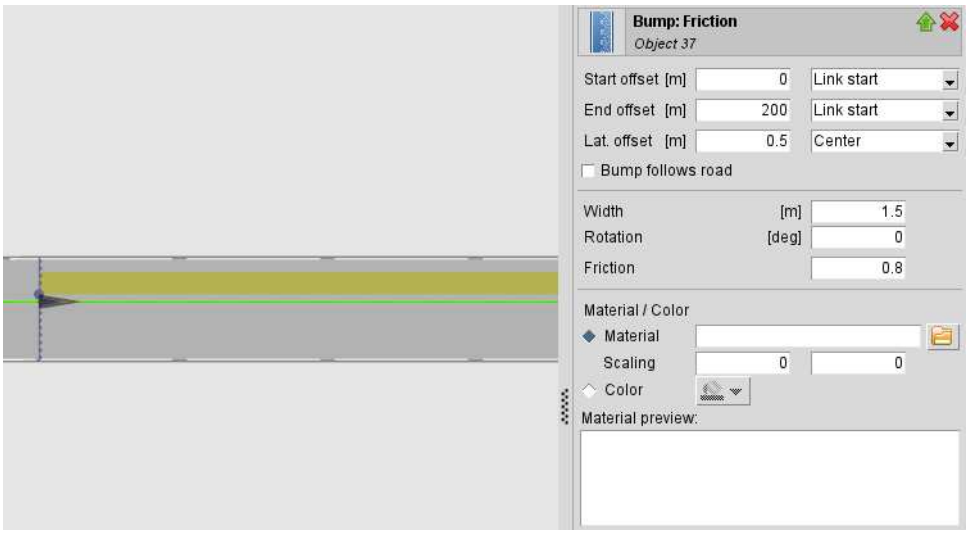


Figure 5.15: Friction stripe

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

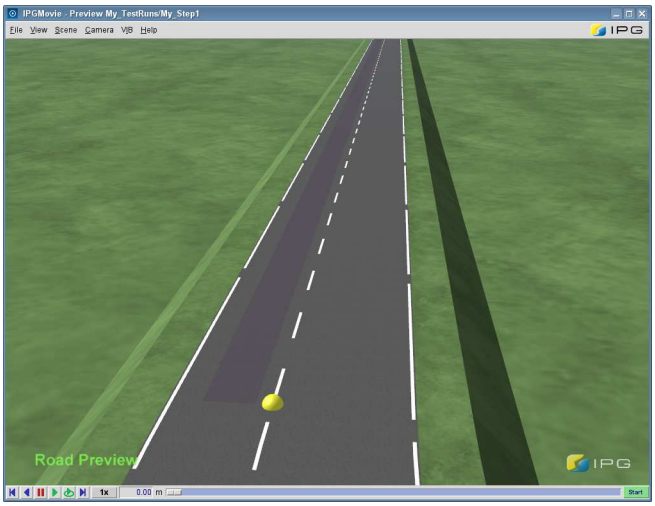


Figure 5.16: Defined Friction Stripe

5.1.6 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 5.17: Special items available in Senario 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 appearence 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 aner vehicle behaviour, for e.g. the driver sill stop at a stop sign.

- 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 5.18: A Road with activated Markers and Movie Objects

Next!

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_Step1* with the predefined TestRun *Step2_ChangeRoad* (located in *Product Examples > Examples > VehicleDynamics > _QuickStartGuide*).

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

5.2 Defining the Maneuver

For the examples in this chapter the previously built TestRun *My_Step1* is required.

Save the TestRun as *My_Step2_ChangeRoad*. Alternatively, use the predefined TestRun *Step_2*.

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

5.2.1 Opening the Maneuver GUI

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

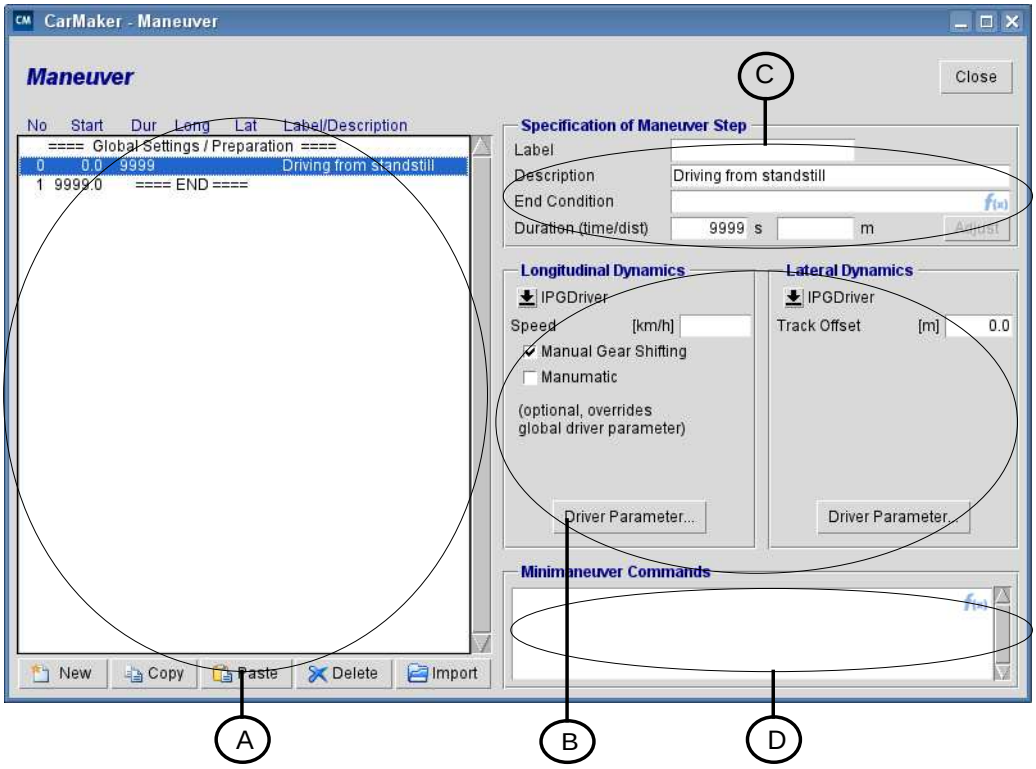


Figure 5.19: The Maneuver GUI in CarMaker

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.
- 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.

5.2.2 Limiting a Maneuver Step by Distance

Up until now, the maneuver step in the TestRun has always been the same: a single maneuver step, driving from standstill, 9999s long (see section 5.1.3 page 33).

In this chapter the maneuver step duration is to be limited not by time, but rather by distance. This means that as soon as the vehicle has covered a specific distance, the simulation is terminated.

In the following example, the maneuver is to end as soon as the vehicle reaches the beginning of the dynamic test area (segment 4, around the top of the slope).

Find out where the dynamic test area begins: *Scenario Editor* > *3D-Preview*. Move the slider, so that the yellow marker is at the desired point and read the distance in the field at the bottom left of the IPGMovie window (around s = 707m, remember this value).

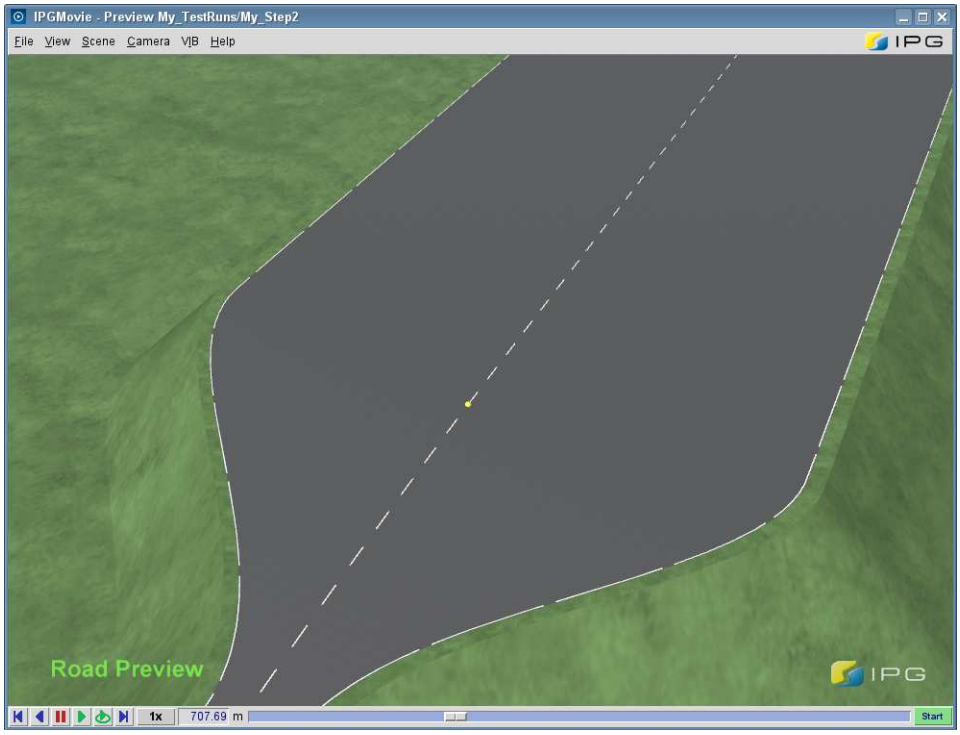


Figure 5.20: The distance to stop the maneuver

In this example the estimated value is 707m. This value will be further used to define the end condition.

Set the end condition: In the CarMaker main GUI, open *Parameters* > *Maneuver* and select the single maneuver step "Driving from standstill". On the right, in "Duration (time/dist.)" enter 707m for the "dist"-value.

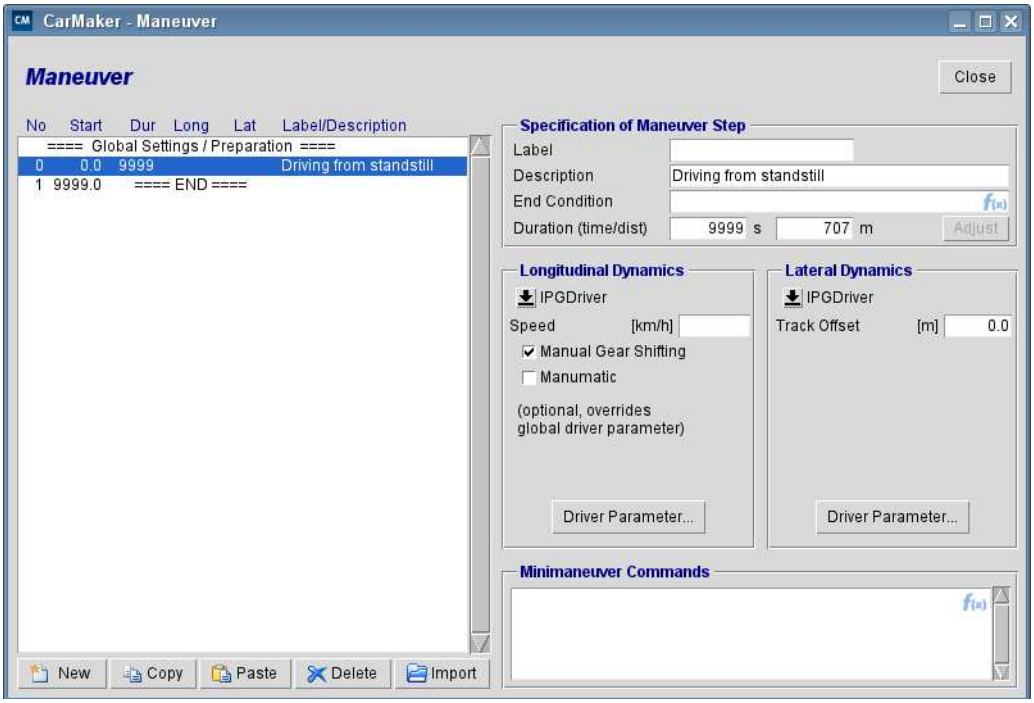


Figure 5.21: Defining the distance as an end condition for the maneuver

Save the TestRun and start the simulation.

Figure 5.22: The new maneuver step displayed in IPGMovie

Now the vehicle stops at the beginning of the dynamic test platform. This happens although an end condition time of 9999s is still set in the maneuver. The vehicle stops at 707m, and not after 9999s, because the end condition that is decisive, is the one that comes into effect first. In this case, the vehicle reaches the 707m before the 9999s, therefor, the distance is the end condition that is considered.

5.2.3 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:
Duration: 3s
Longitudinal Dynamics: IPGDriver
Lateral Dynamics: Sinus steering, Amplitude = 50 deg, Period = 1s, # Periods = 3

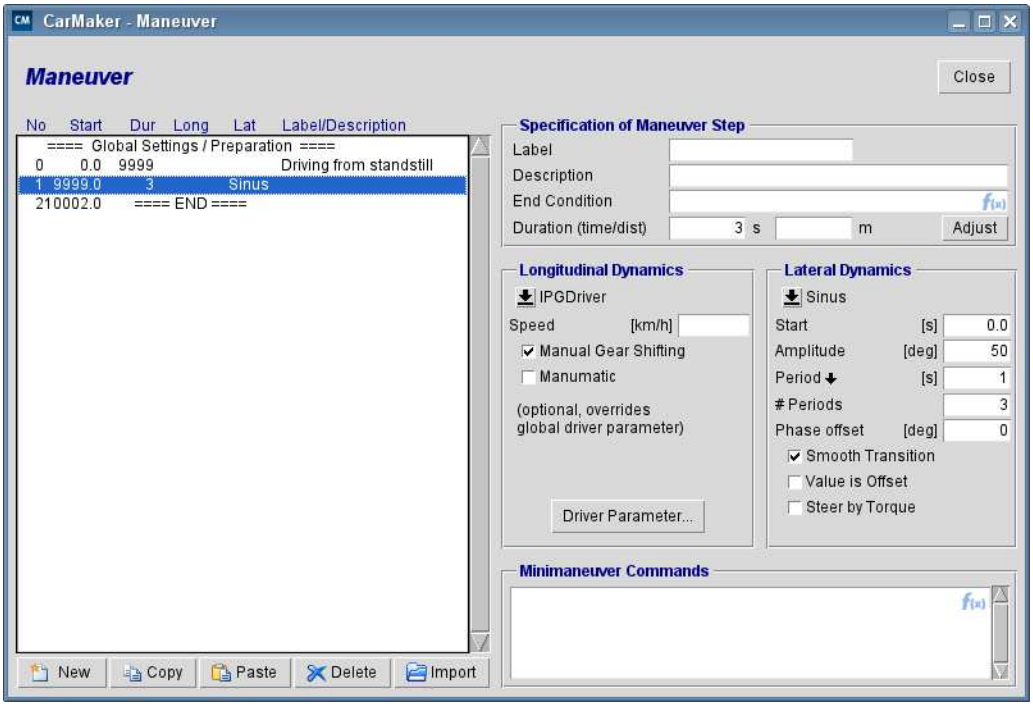


Figure 5.23: Open loop maneuver step

Save the TestRun.

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

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, both maneuver steps are displayed. During the first part of the overall maneuver, which lasts up to the end of the inclination, the first maneuver step is highlighted. As soon as the sinus steering phase starts, the second 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.

5.2.4 Adding a Closed Loop Maneuver Step with IPGDriver

After the sinus steering maneuver step, the maneuver will be extended by a closed loop maneuver performed by IPGDriver.

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 Maneuver GUI, click “END” and then “New“ to insert a new maneuver step.

Define the new step as follows:
Duration: 9999s
Longitudinal Dynamics: IPGDriver, Speed (100 km/h)
Lateral Dynamics: IPGDriver.

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

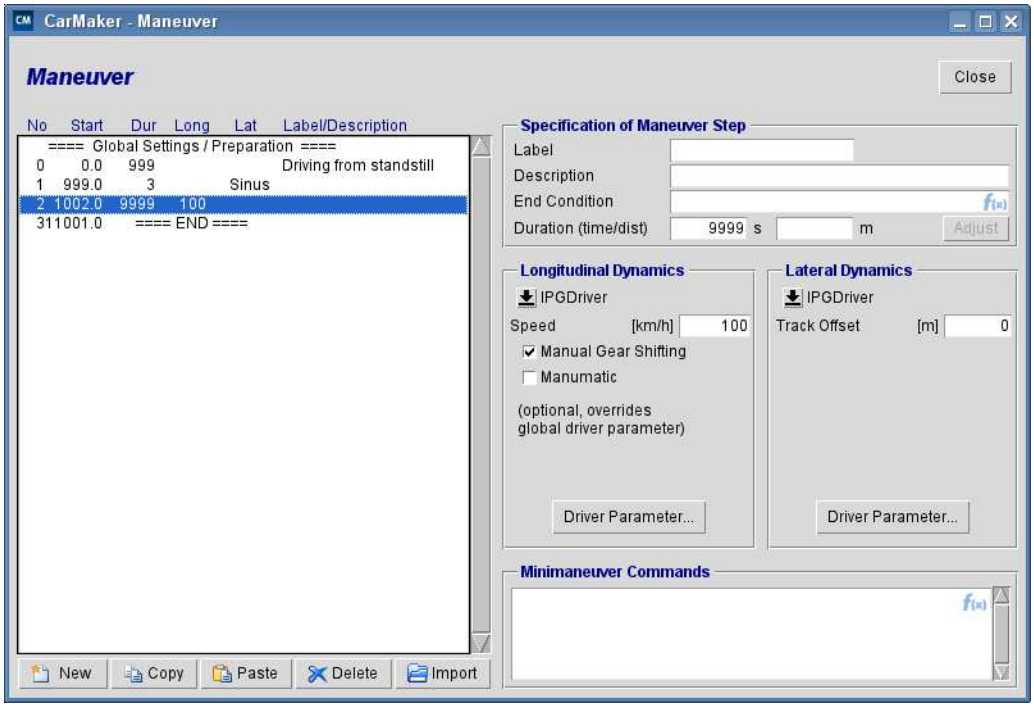


Figure 5.24: Closed loop maneuver step

The vehicle now continues driving after the sinus steering, since another maneuver step with a duration of 9999s has been added.

Observe the speed using Instruments: during the third maneuver step, the driver accelerates up to 100 km/h. These values correspond to what is defined in the corresponding maneuver.

This example shows two things:

- The maximum speed that the driver is to aim for can be defined seperately for each maneuver step.
- The driver stays on the track and identifies the ideal trajectory automatically. The driver’s parameters will be explained in section 5.3.2 ’Parametrizing IPGDriver’ on page 52.

5.2.5 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 5.25: Defining the duration of a Maneuver Step

In section 5.2.2 'Limiting a Maneuver Step by Distance' it was discussed how a maneuver step can be restricted by a certain time or distance. These are by far not the only possibilities 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.

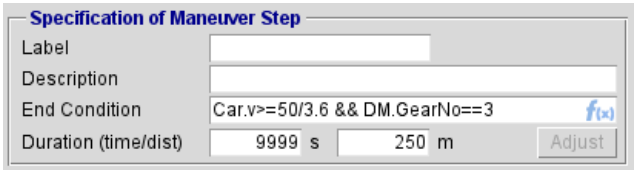


Figure 5.26: Defining the "End Condition" Parameter

Longitudinal Dynamics

Figure 5.27: Defining the longitudinal dynamics

To control the longitudinal dynamics, for 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. See section 5.3.2 'Parametrizing IPGDriver' on page 52.
- 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

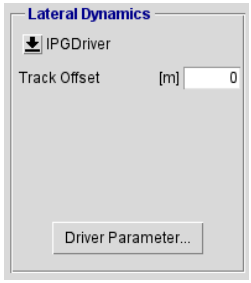


Figure 5.28: Defining the lateral dynamics

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

- IPGDriver: Fully automated control of the steering wheel. See section 5.3.2 'Parametrizing IPGDriver' on page 52.
- Sinus: Sinus input on the steering wheel.
- Sinus Sweep: Alternating sinus input on the steering wheel.
- Steer Step: Steer step input on the steering wheel.
- Follow course: This option is a steering wheel controller like IPGDriver, but simplified.

Next!

The TestRun altered in this chapter *My_Step2* can now be compared to the predefined TestRun *Step3_ChangeManeuver* (located under Product Examples > Examples > VehicleDynamics > _QuickStartGuide).

The next chapter will contain further information regarding IPGRoad and IPGDriver.

5.3 Advanced Features

5.3.1 Using a Digitized Track

In the CarMaker main GUI, load the TestRun “Nurnurgring_RoadFeatures“: *Examples > BasicFunctions > Road > SampledRoads > Nurburgring_RoadFeatures*

Set simulation speed to “Realtime“, open IPGMovie and start the simulation.

Figure 5.29: The TestRun “Nurburgring_RoadFeatures” in IPGMovie

The greatest difference between this road and the one built in the previous chapters is that the road used here has a much higher resolution. To achieve this kind of result, a digitized road was used.

This means that the x-, y- and z-coordinates of the middle line of the track have been written to a file. In this case, there is no need to define the segments since the trajectory is already given by the data in the file.

Save this TestRun as *My_Step3_ChangeManeuver* and open the Scenario Editor.

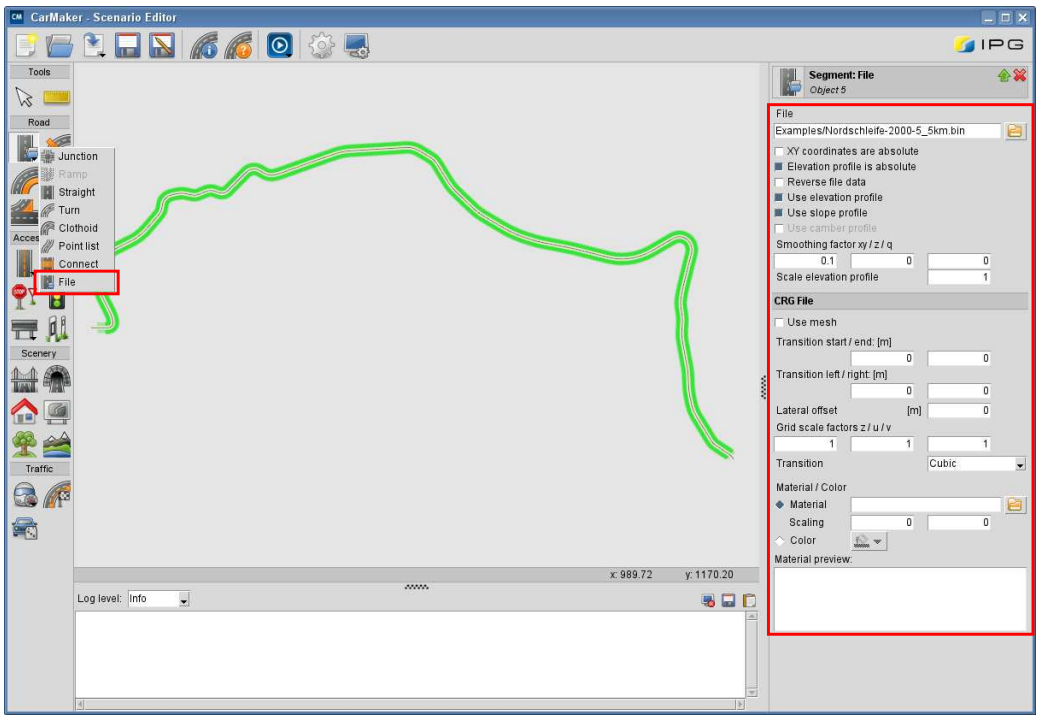


Figure 5.30: A digitized road loaded in the Scenario Editor

In the side bar of the Scenario Editor under the *Road* icon *Road Segment*, there is an option called *File* to load a digitized track file, usually this is a simple ASCII file (text file). Optionally, the ASCII file can be converted into a binary file, enabling the simulation to run faster, as is the case with the TestRun at hand.

In the parameterization menu at the right side, the digitized road profile can be parameterized. E.g. different profile for elevation and slope can be activated/deactived and filter patameters set. However, using a digitized road, the general settings of the road such as track width and friction still need to be parameterized. The same means as for a user built road are available.

Open the Scenario Editor, click the "Bump" icon in the "Road" tab and select the "friction" option. Click on the desired Link and define the beginning and end points for the stripe. For e.g.: Point 1 = 40 and point 2 = 100.
Then, edit the parameters in the tab on the right according to the figure below.
"Bump follows road" makes sure that the stripe doesn't continue straight on when the road makes a turn.

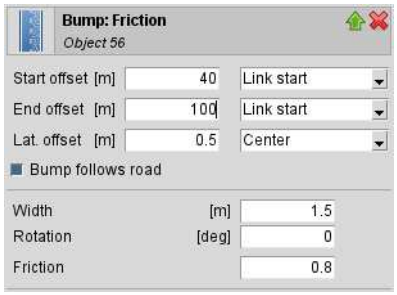


Figure 5.31: Friction stripe parameters

Start the simulation.

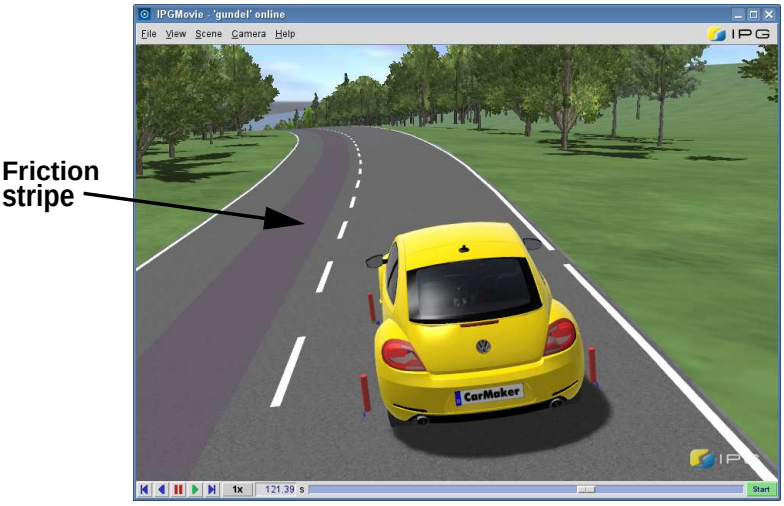


Figure 5.32: The visualization of the friction stripe in IPGMovie

So even when using digitized, predefined roads, friction stripes can be added to the road definition.

5.3.2 Parametrizing IPGDriver

The following chapter will provide a little insight regarding the parameterization of IPGDriver.

Open the IPGDriver GUI in the CarMaker main GUI: *Parameters > Driver*.

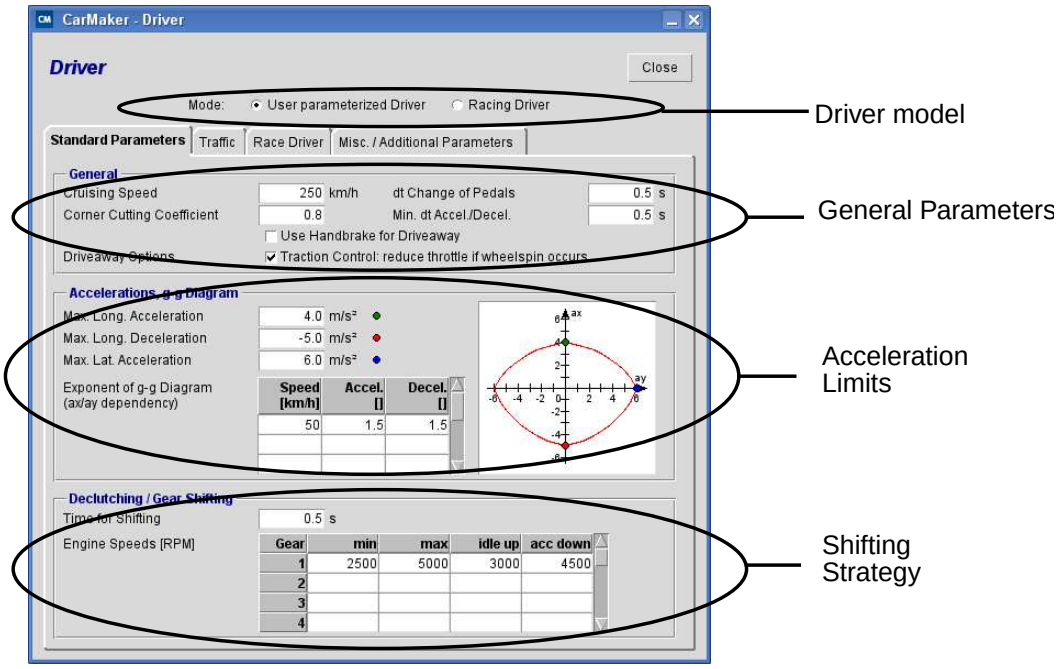


Figure 5.33: Parameterization of IPGDriver

IPGDriver provides two models:

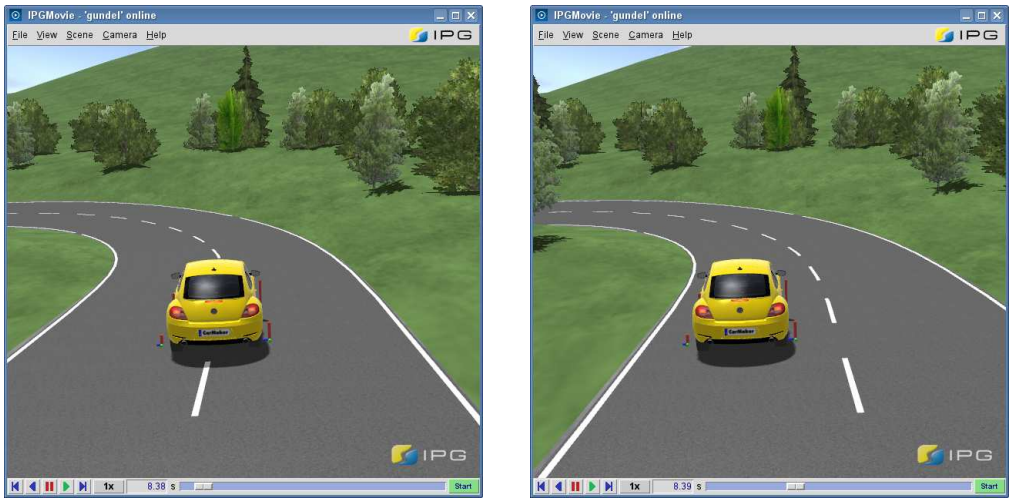
- *User parameterized Driver*: Here, the user can define every parameter for the IPGDriver individually. This driver model is used in the following examples.
- *Racing Driver*: After a neccessary learning phase, this model calculates the driver's physical limits so that these do not need to be defined by the user.

The IPGDriver GUI grants access to several parameters:

- *General parameters*. The most important ones are:
  - Cruising Speed: Velocity that the driver will try to reach while keeping the vehicle on track. It can be redefined for each maneuver, as has been tested in section 5.2.4 'Adding a Closed Loop Maneuver Step with IPGDriver' on page 46.
  - Corner Cutting Coefficient: Controls how badly the driver cuts curves. Values range from 0 to 1 with 0 meaning that the driver does not cut curves at all.
- *Acceleration limits*: These are defined via the G-G diagram and are used to set:
  - the driver's maximal longitudinal and lateral acceleration (green, blue and red points in the diagram).
  - the exponents: help define the shape of the diagram (red line). They give the driver the information regarding his maximal acceleration when longitudinal and lateral acceleration are linked. The exponent can be defined for various speeds.
- *Shifting strategy*: Defines the engine speed at which the driver shifts up or down.

The following example can be used to test the corner cutting coefficient.

Set the corner cutting coefficient to 0 and the simulation speed to “realtime” and start the simulation.
Alternatively, set the coefficient to 0.8 and observe the different behaviour.



Corner Cutting Coefficient 0.0 **Corner Cutting Coefficient 0.8**

Figure 5.34: Examples for different Corner Cutting Coefficients

This small example shows how the driver’s trajectory can be modified. The pictures above show a clear difference.

The imagines were generated by taking screen shots. This a rather inconvenient way of comparing vehicle behaviour since it is hard to catch the exact same moment twice. The next section describes how the animations of two different simulations can be properlycompared.

Do not close IPGMovie.

5.3.3 Comparing two Simulations in IPGMovie

In the IPGMovie window:
1) Click *Scene > Reference Vehicle > Copy current Motion Data*
2) Activate the option *Scene > Primary and Reference Vehicle*

With the first action, the animation of the last simulation is saved in a buffer memory.

With the second action, IPGMovie displays the animation of the current simulation (“Primary Vehicle“), but also the one that is in the buffer memory (“Reference Vehicle“). The Reference Vehicle will be displayed as a shadow.

**Open the IPGDriver GUI and set the corner cutting coefficient to 0.
Set simulation speed to "realtime" and start the simulation again.**

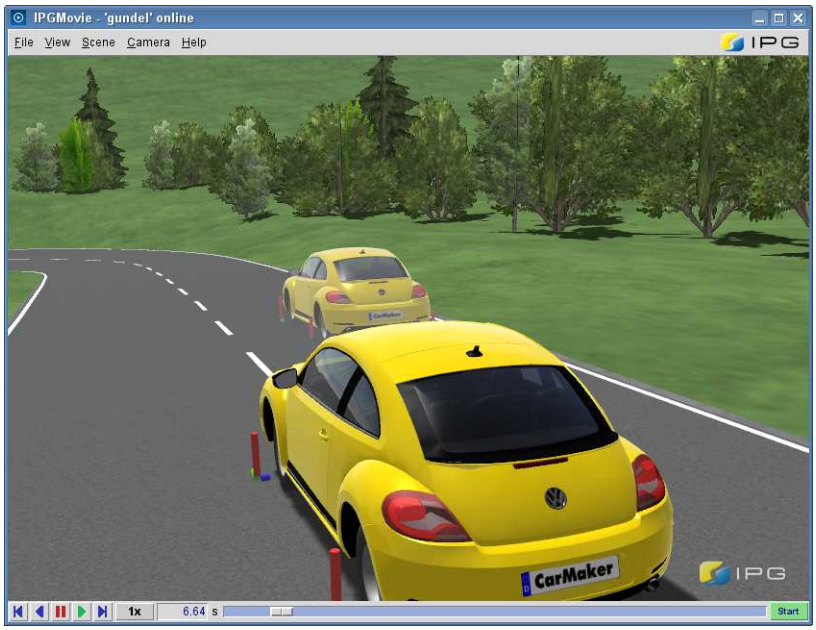


Figure 5.35: Primary and Reference Vehicle displayed in IPGMovie

Now, the animation displays two vehicles: the shadowed vehicle is the one from the previous simulation, the other is from the current simulation.

Do not save the last change made to the TestRun (corner cutting coefficient).

Next!

The following chapter explains how to use other vehicles, trailers and tires in CarMaker.

Chapter 6

Changing the Vehicle and Tire Data

6.1 Using another Vehicle Parameterization

In the previous chapters throughout all the TestRuns the same vehicle was used, the so called *DemoCar*. In this chapter the user will lear to use another vehicle parameterization. For the folllowing examples, *Step4_ChangeDriver* needs to be loaded from *Product Examples > Examples > VehicleDynamics > _QuickStartGuide/*.

Save the TestRun as “My_Step4“.
In the CarMaker main GUI, click “Select“ in the box “Car“, choose the “Demo_Medium“ car and click on OK.

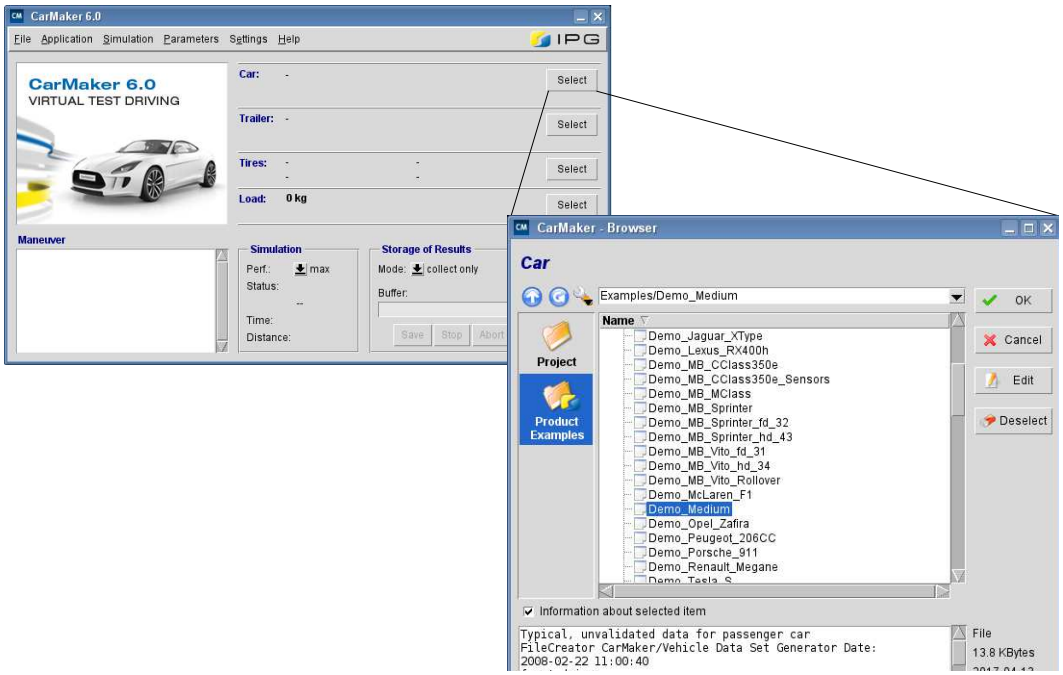


Figure 6.1: Changing the car

Now the “Medium“ car is displayed in the CarMaker GUI and will be used for further simulations.

Save the TestRun and open IPGMovie.
Start the simulation.

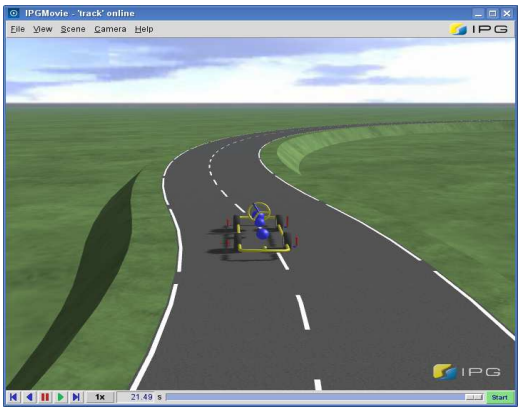


Figure 6.2: Simulation with the Demo_Medium car

The animation in IPGMovie no longer displays a vheicle body. This is because no body has been defined for the vehicle data set “Demo_Medium“ to be displayed in IPGMovie.

This example shows the difference between the data in the parameterization of the vehicle model and what can be seen in IPGMovie. The body used in IPGMovie is just a file that can be varied randomly. Fior e.g. even when simulating with the medium car, it would be possible to display the beetle body in IPGMovie.

By the way, this example shows you the difference between the data in the parameterization of the vehicle model, and what you see in IPGMovie: the body used in IPGMovie is just a file that can be changed.
E.g. we could display in IPGMovie the New Beetle used in the previous TestRun even if we simulate the Medium car. We will do that in the next chapter.

section ’Parameterizing the Vehicle and Tires’ on page 60 will shwo how the user can parameterize the vehicle to suit his individual needs.

6.2 Using another Trailer Parameterization

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.

6.3 Using another Tire Parameterization

In the CarMaker main GUI, click on “Select“ in the box “Tire“, choose the tire “MF_205_60R15_V91“ and then click on OK.

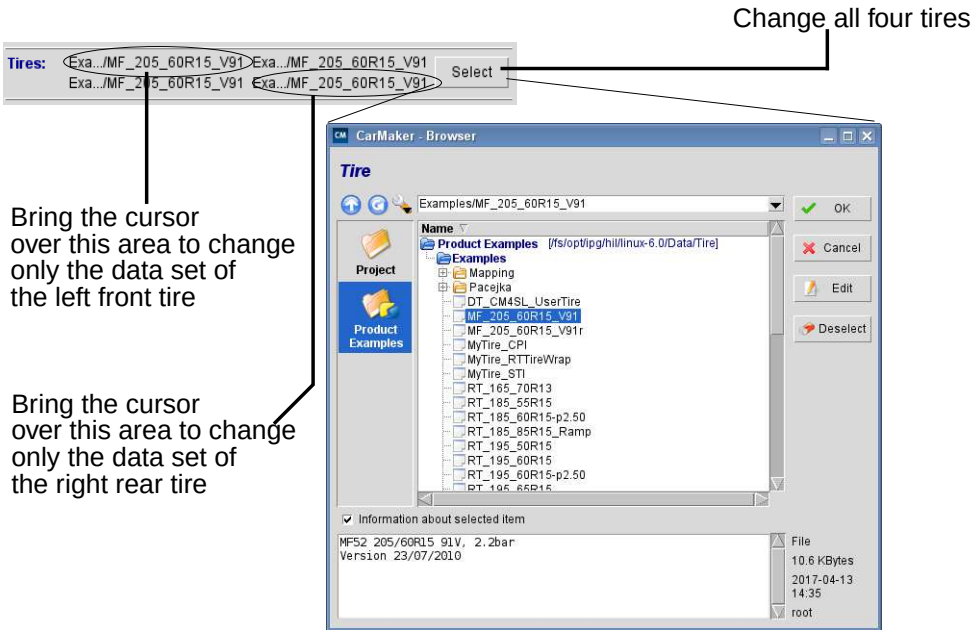


Figure 6.3: Changing the tires

Now, the new tire data set is displayed for all four wheels. Optionally, each tire data set can be changed separately in order to have different tires in the front and rear.

No usable file will can be found in the folder “Mappings”. It is best not to choose any file here.

Click the “Select” button shown in the picture above and select the data set “RT_195_60R15“.

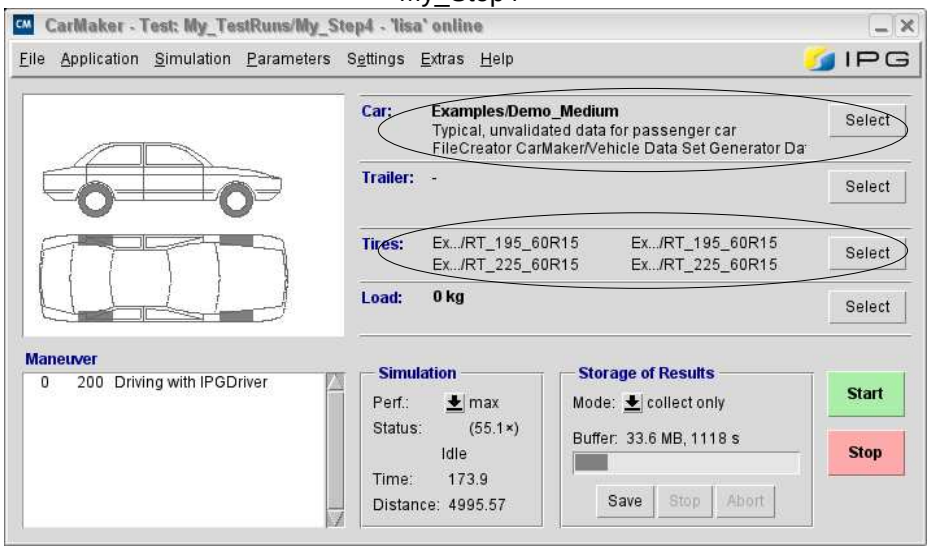
Click the area correspondant to the left rear tire and select the data set “RT_225_60R15“. Repeat for the right rear tire.

In section 7.3 ’Creating a new Tire Data Set’ on page 83, it is explained how a new tire data set is created that corresponds to the user’s needs. Alos, a few in CarMaker available tire models will be presented.

6.4 Saving the new Data Sets within the TestRun

Save the TestRun *My_Step4*.
Now open the TestRun *My_Step2* and look at the differences.

My_Step4



My_Step2

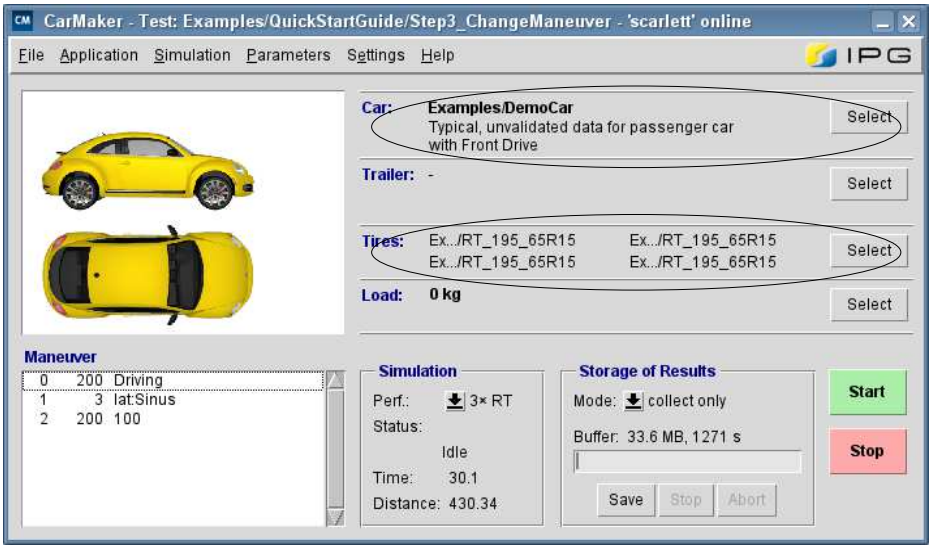


Figure 6.4: Comparison between the two vehicles

This action shows that the user can implement all sorts of combinations in a TestRun. All TestRuns including their complete parameterizations can be saved and loaded as pleased.

6.5 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 onle 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.

Next!

The next chapter will explain how the mentioned Data Sets can be created.

Chapter 7

Parameterizing the Vehicle and Tires

7.1 Creating a new Vehicle Data Set

For the following examples, the TestRun *Step5_ChangeVehicle* needs to be open. It can be found under *Product Examples* > *Examples* > *VehicleDynamics* > *_QuickStartGuide*.

Save the TestRun as *My_Step5*.

7.1.1 Save as a New Data Set

In the CarMaker main GUI, click *Parameters* > *Car* to open the Vehicle editor.

In the Vehicle editor (also called Vehicle Data Set), click on *File* > *New*

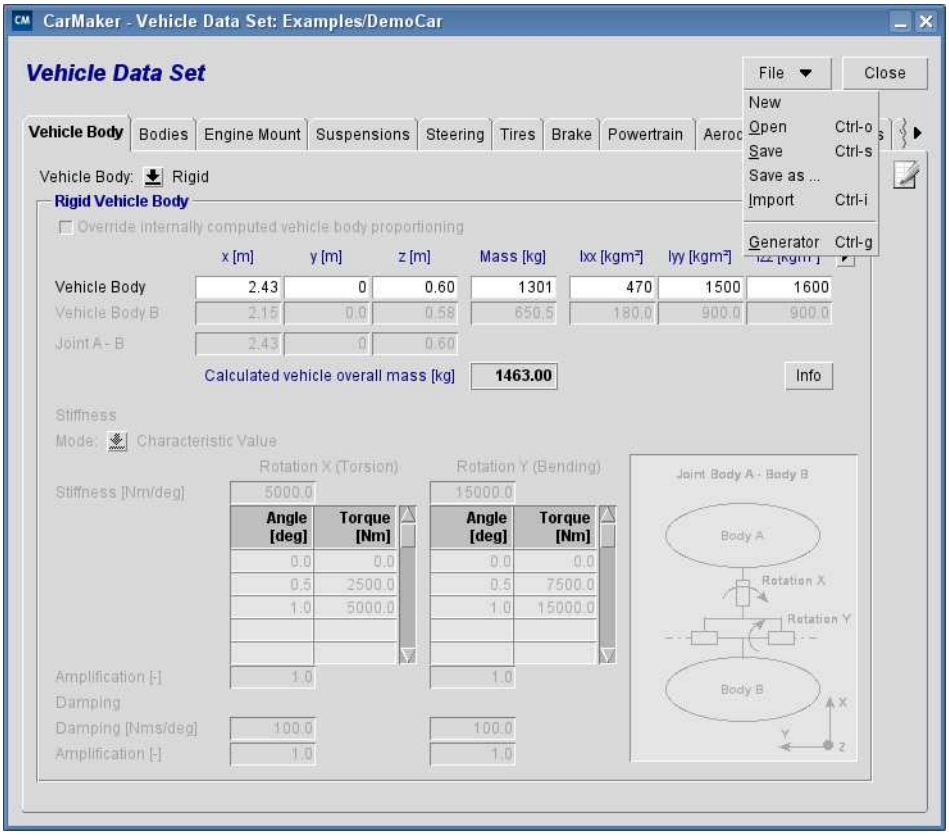


Figure 7.1: Vehicle Data Set

With this action, a default vehicle data set has been generated.

Save the Vehicle Data Set: In the Vehicle editor, click *File* > *Save As* > type the name of the new vehicle data set you are generating, e.g. “my_Car” > *OK*.

The new data set is saved. Now it needs to be parameterized correctly, since there are only default values filled in at the moment.

7.1.2 Using the Vehicle Generator

In the Vehicle editor, click on *File > Generator*.

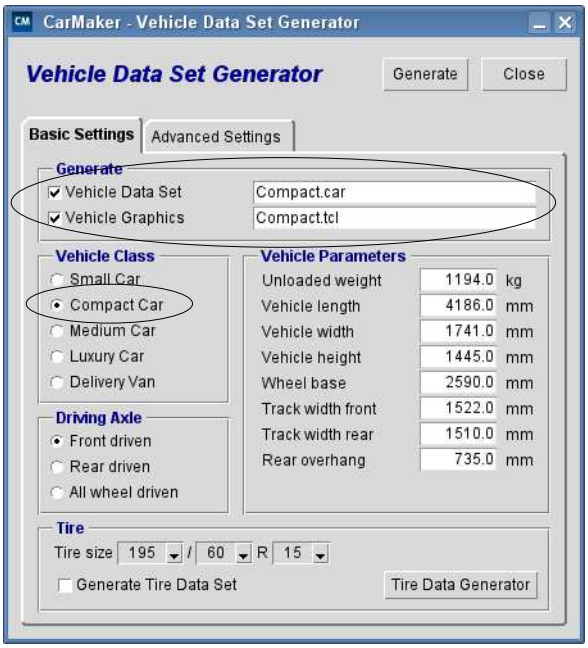


Figure 7.2: Vehicle Data Set Generator

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*.

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

7.2 Parameterizing the Vehicle Model

This chapter describes the parameters of a vehicle data set that need to be specified. The following sub-sections (Main Body, Masses, etc.) refer to the various tabs of the Vehicle editor.

7.2.1 Vehicle Body

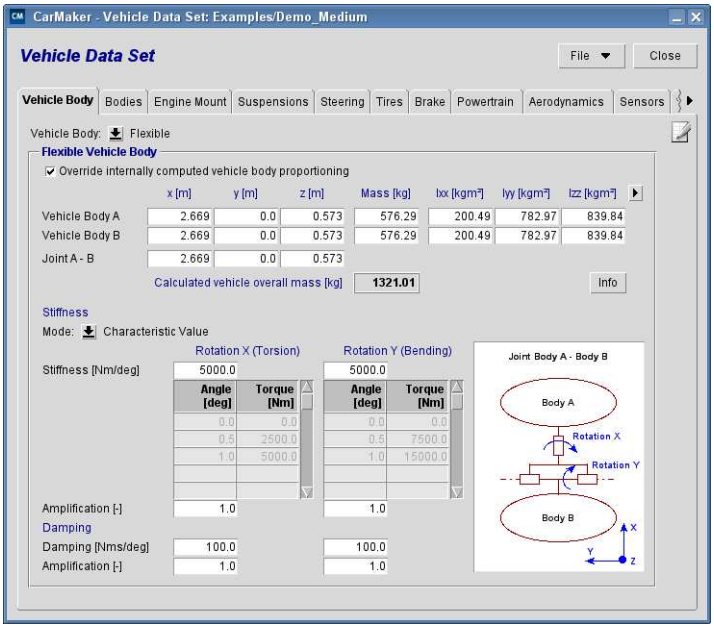


Figure 7.3: Selecting a vehicle body

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

- Flexible 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.

7.2.2 Bodies

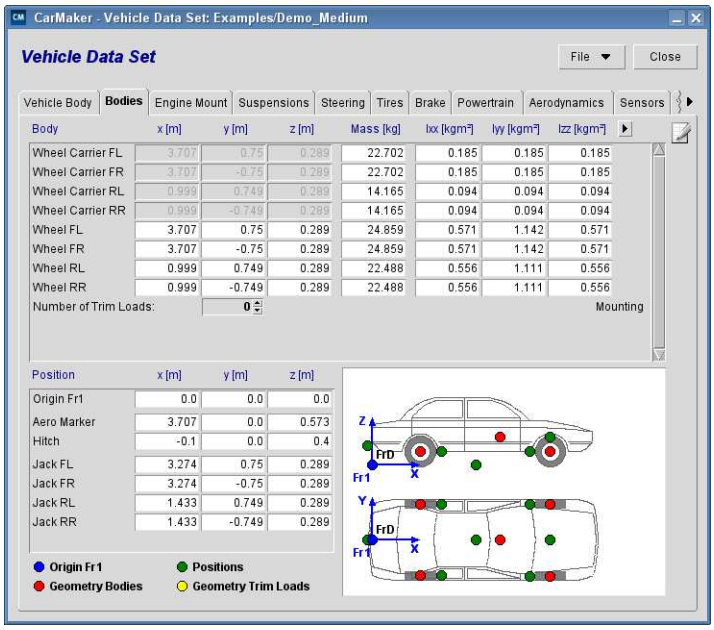


Figure 7.4: Body definition

In this tab, all other masses are defined. Again, the definition takes place in the design position, according to the vehicle model chosen in the “Main 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:

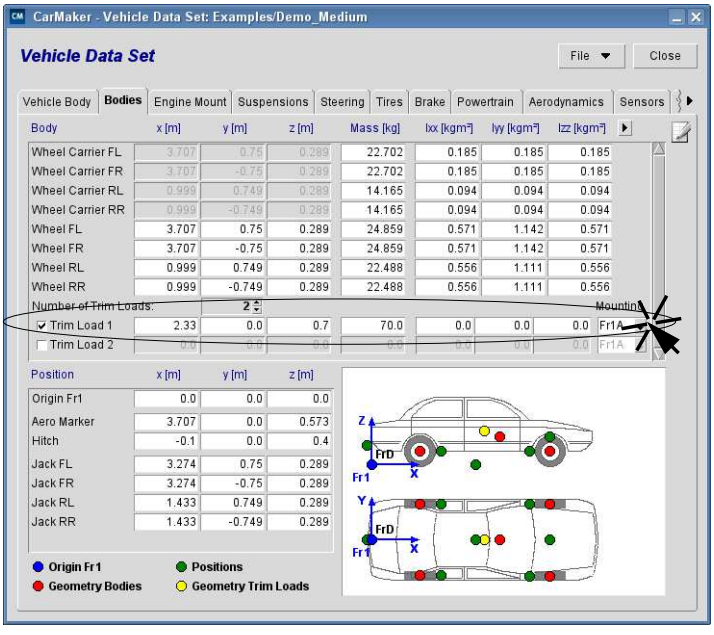


Figure 7.5: Trim Load definition

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

7.2.3 Engine

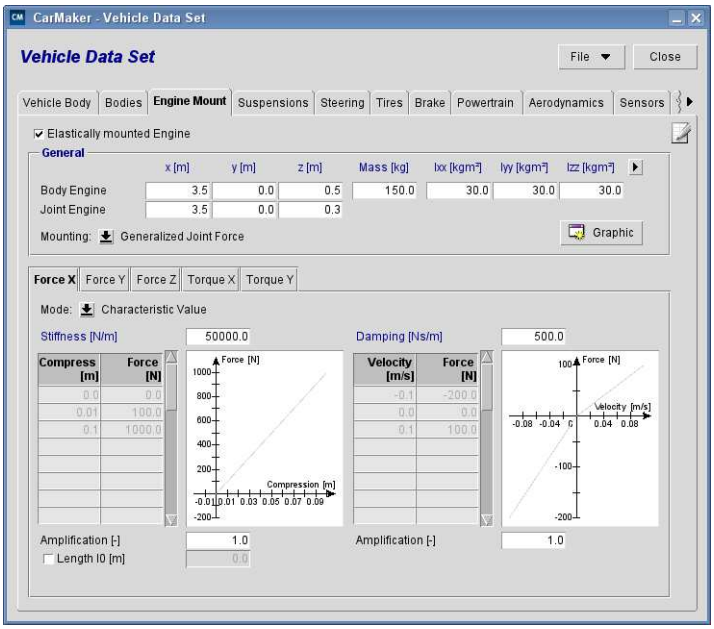


Figure 7.6: Engine definition

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 between engine and main body. The joint consists of a spring damper element which can be characterized in the lower part of the dialog.

7.2.4 Suspensions

Spring

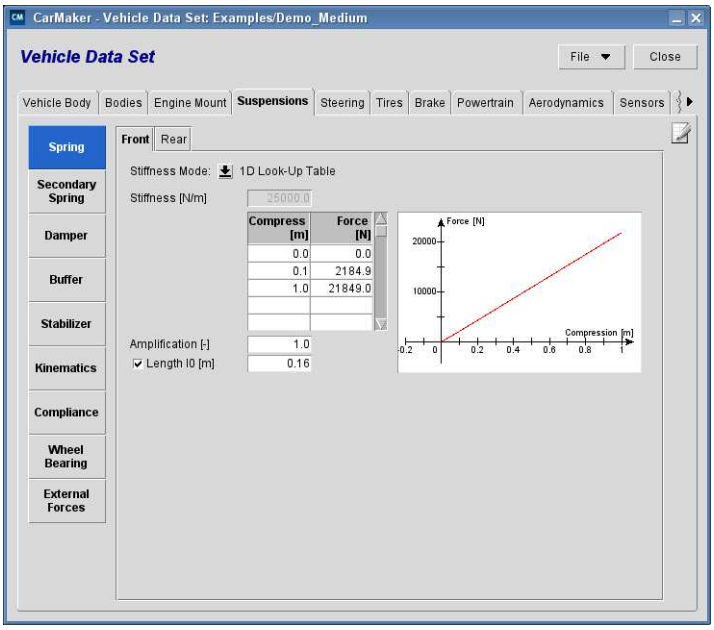


Figure 7.7: Spring definition

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

Secondary Spring

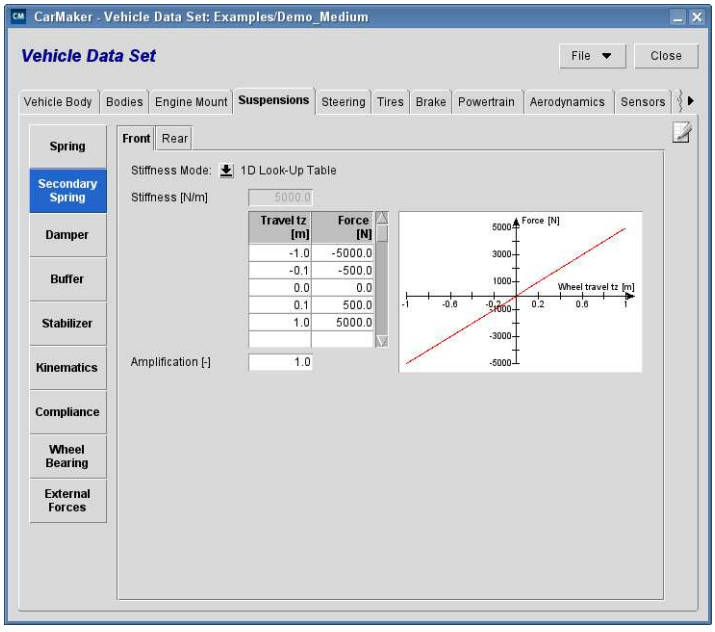


Figure 7.8: Secondary Spring definition

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

Damper

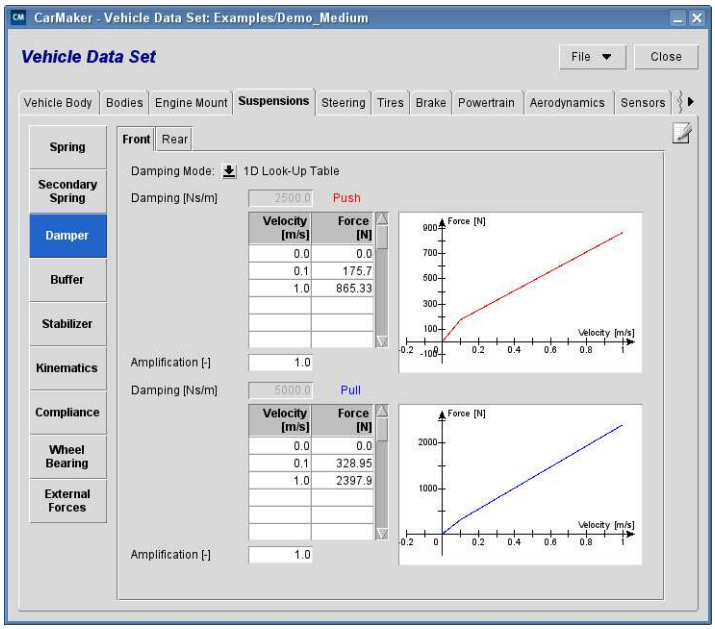


Figure 7.9: Damper definition

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

Buffer

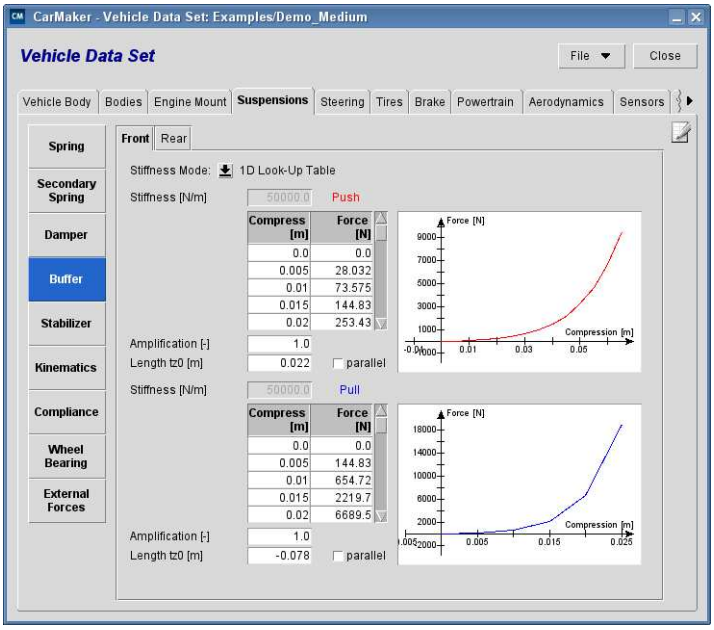


Figure 7.10: Buffer definition

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.

Stabilizer

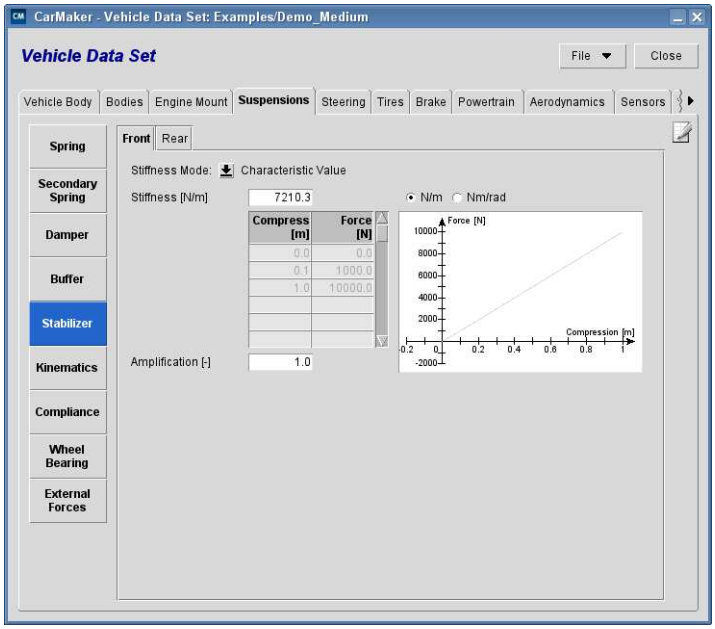


Figure 7.11: Stabilizer definition

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

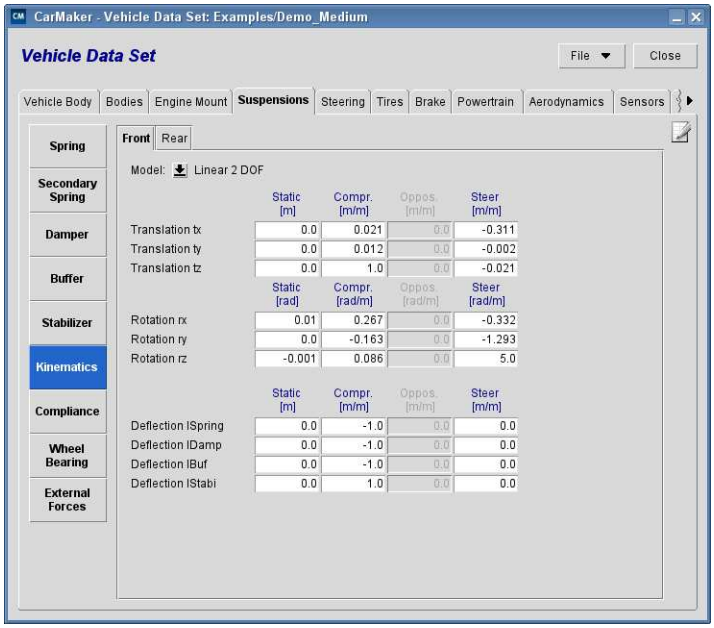


Figure 7.12: Kinematics definition

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
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:

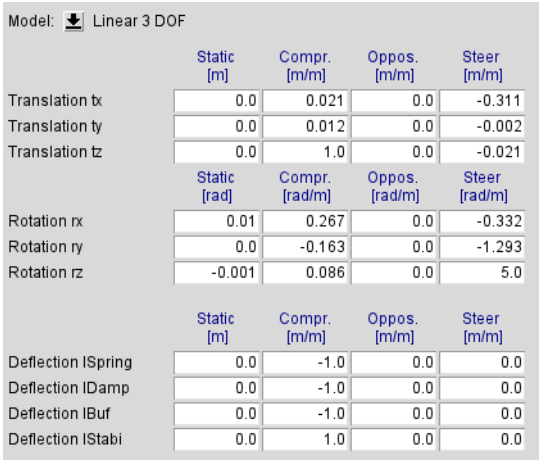


Figure 7.13: Translations and Rotations

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

The columns represnt 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

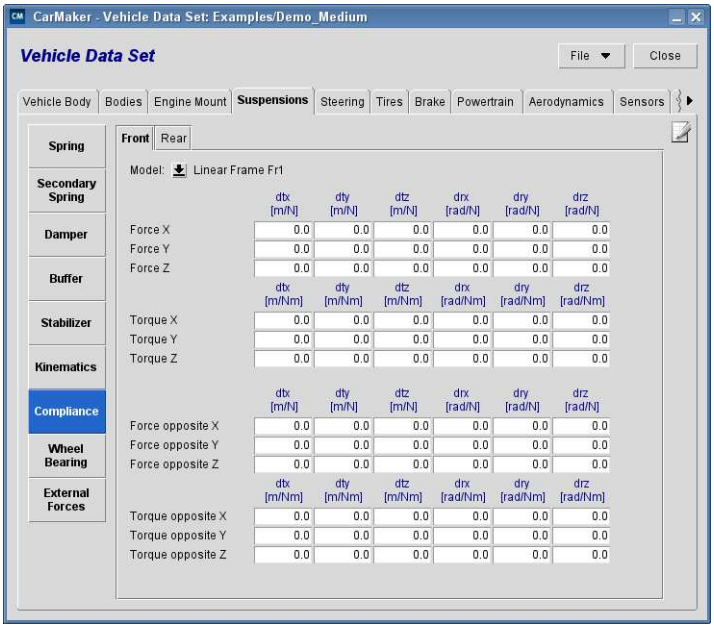


Figure 7.14: Compliance definition

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

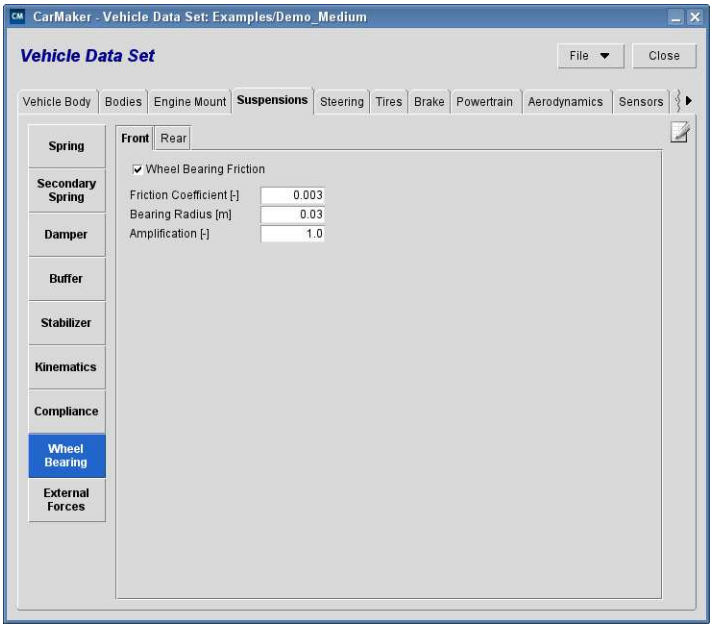


Figure 7.15: Wheel Bearing definition

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

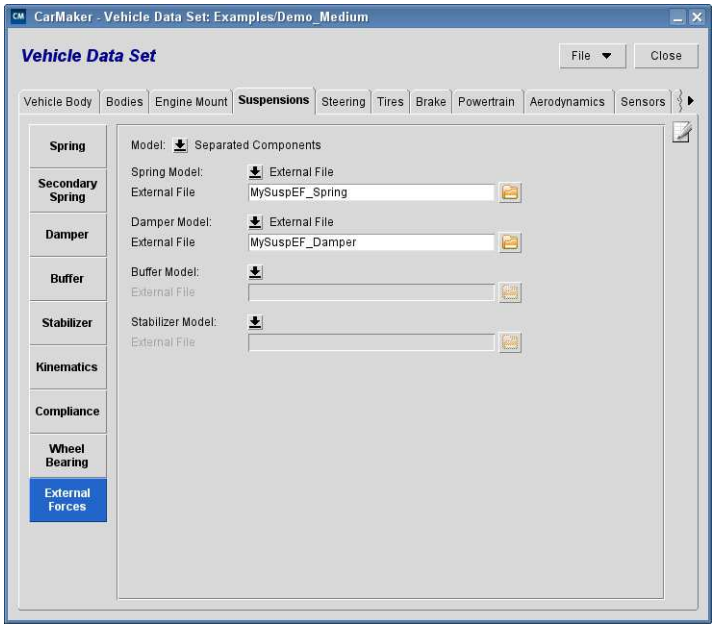


Figure 7.16: External Forces definition

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

7.2.5 Steering System

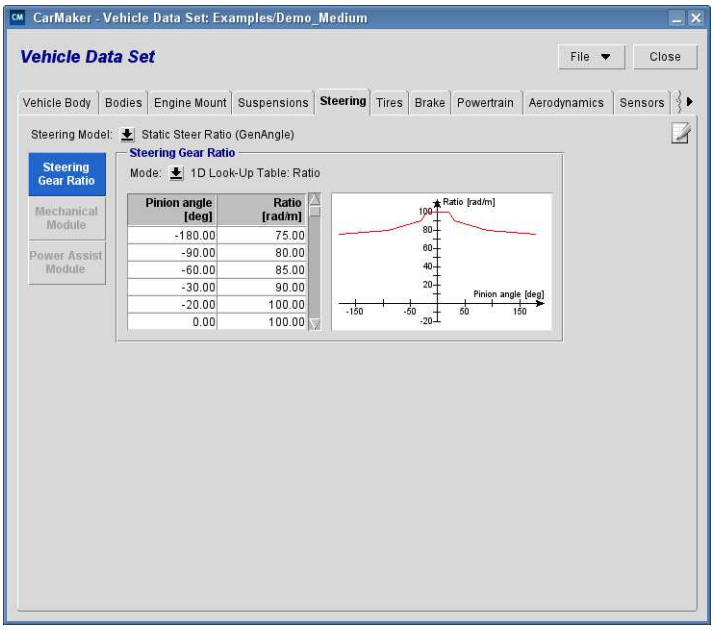


Figure 7.17: Steering System definition

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, for 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 frictions effects. The power steering can be either hydraulic or electric.

7.2.6 Brake System

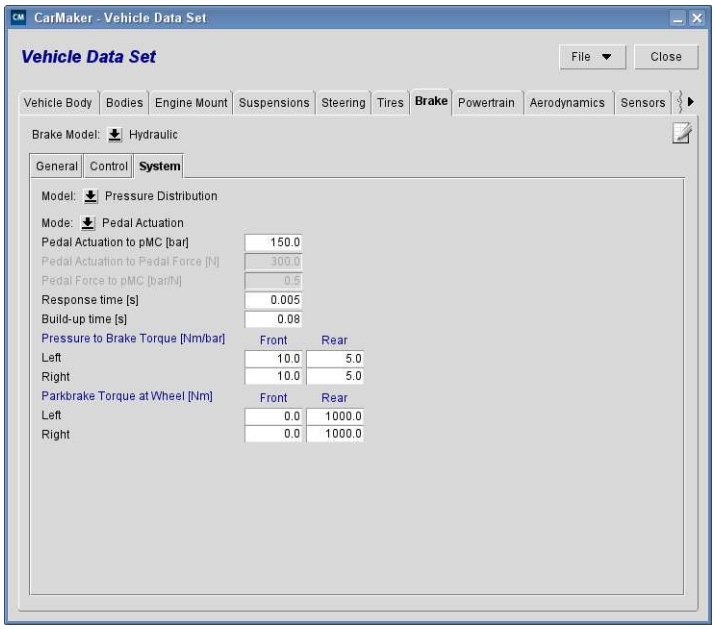


Figure 7.18: Brake System definition

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.

7.2.7 Powertrain

Model

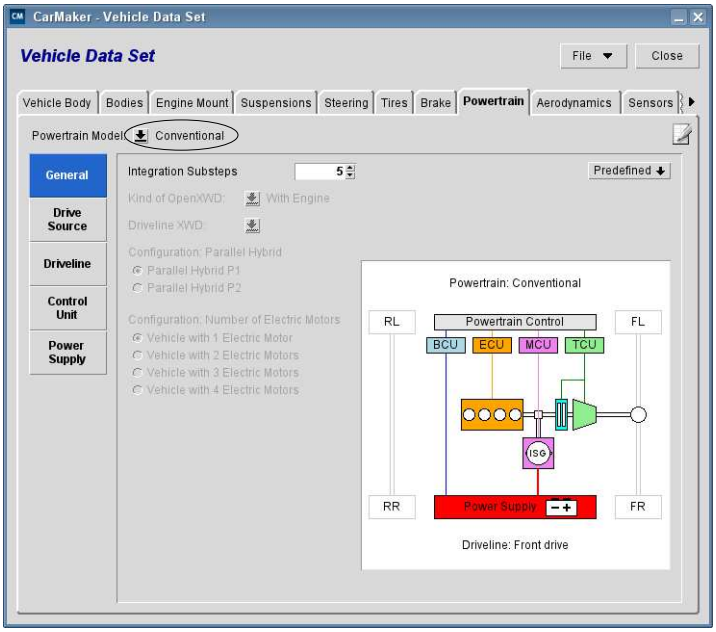


Figure 7.19: Powertrain Model definition

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

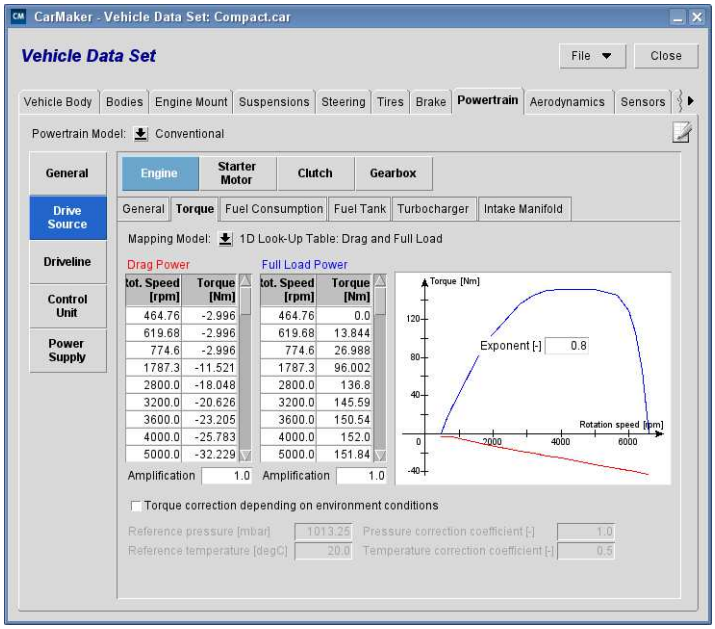


Figure 7.20: Engine definition

The user can define the torque characteristics of the engine. In the menu *Engine Model,* the kind *Look-Up Table* and in the menu *Engine Mapping* the kind *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

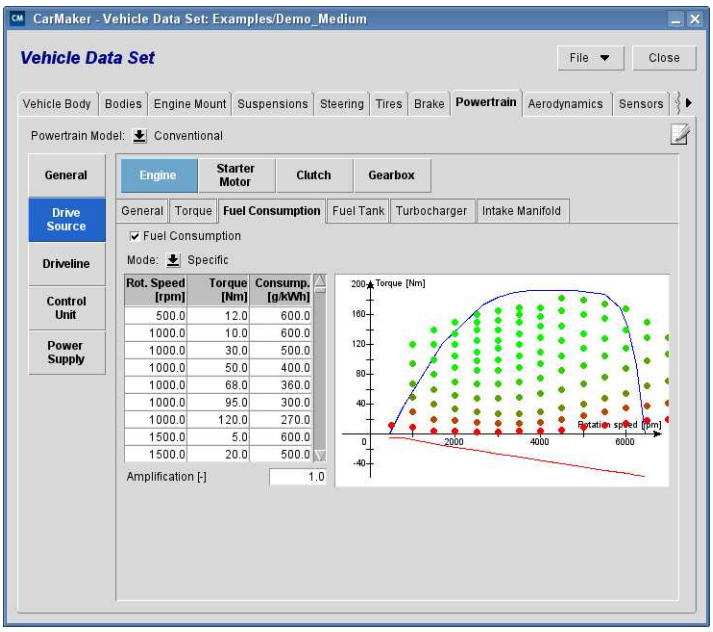


Figure 7.21: Fuel Consumption definition

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

Clutch

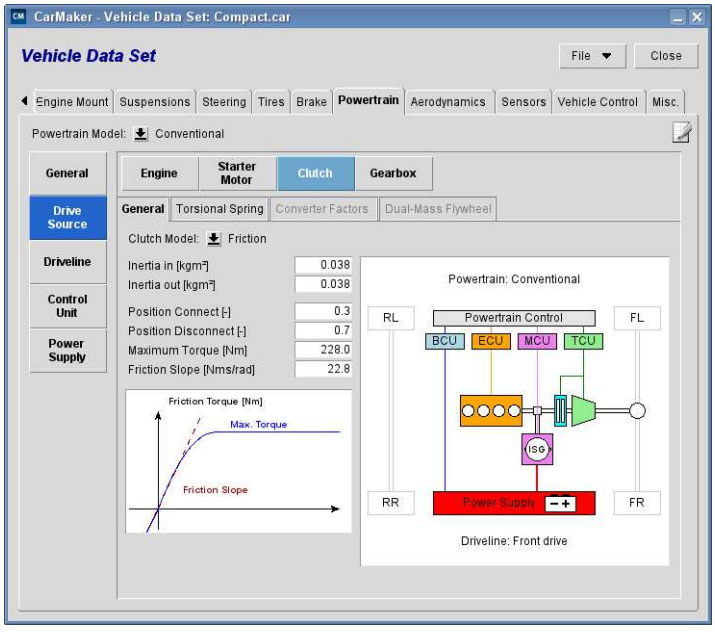


Figure 7.22: Clutch definition

Gear Box

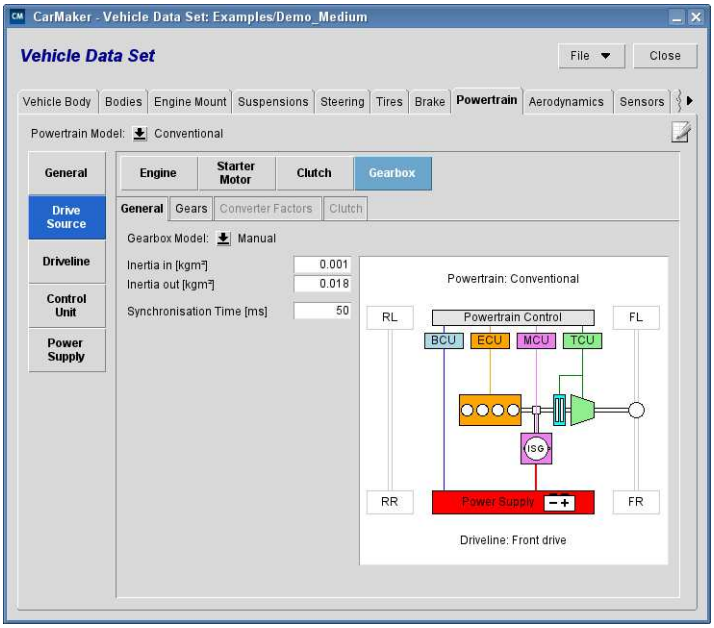


Figure 0.1: Gear Box definition

Here you can parameterize the gear ratios. Two models are available: a manual gearbox as well as an automatic one.

Driveline

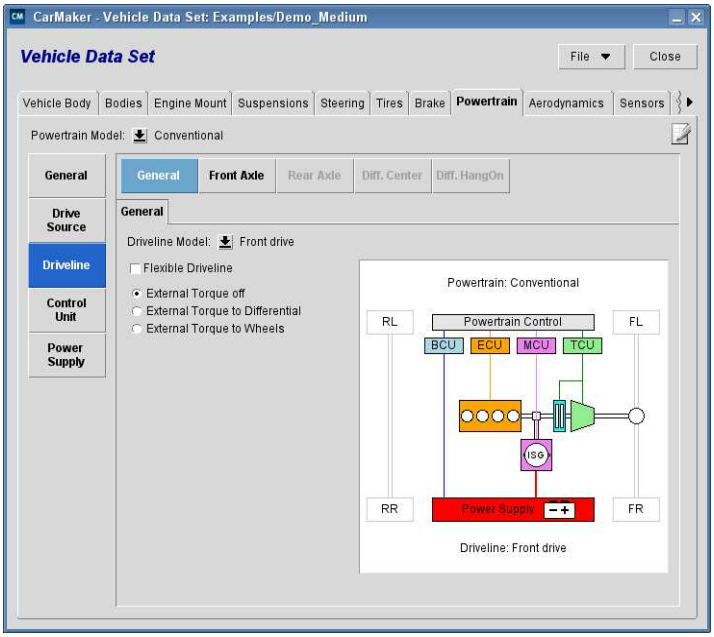


Figure 0.2: Driveline definition

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

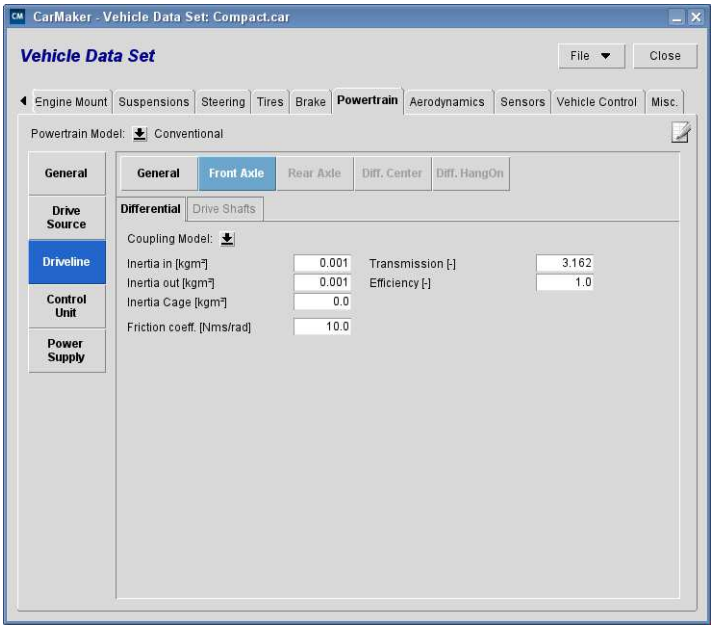


Figure 7.25: Differentials definition

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 (refer to section 'Additional Information about CarMaker' on page 109).

7.2.8 Aerodynamics

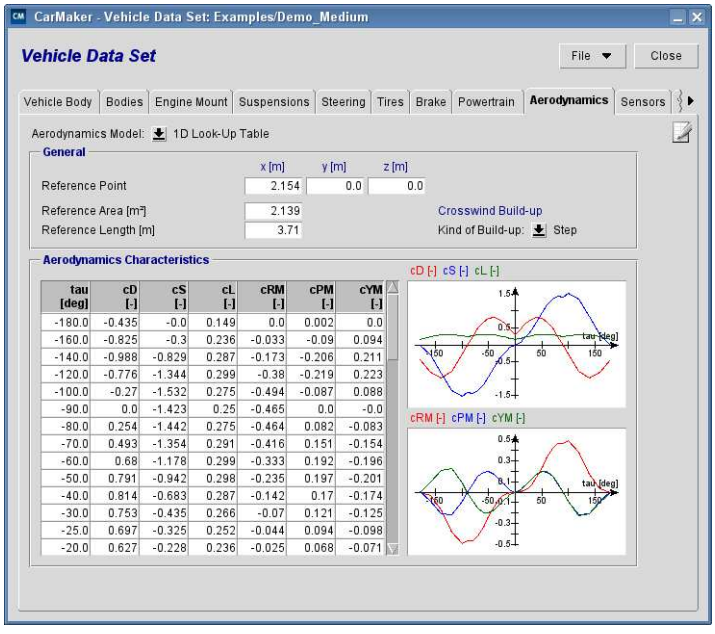


Figure 7.26: Aerodynamics definition

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.

7.2.9 Sensors

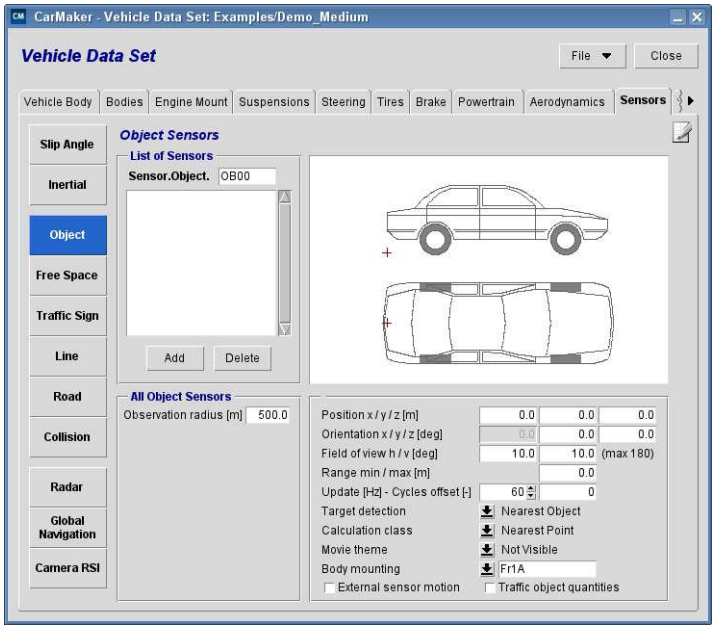


Figure 7.27: Sensor definition

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 (GPU based)
- 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
- Radar Sensor: physical sensor model for obstacle detection
- Global Navigation Sensor: simulates the positions of satellites and a receiver build in the vehicle
- Camera RSI: raw signal interface to image data provided by IPGMovie

7.2.10 Vehicle Control

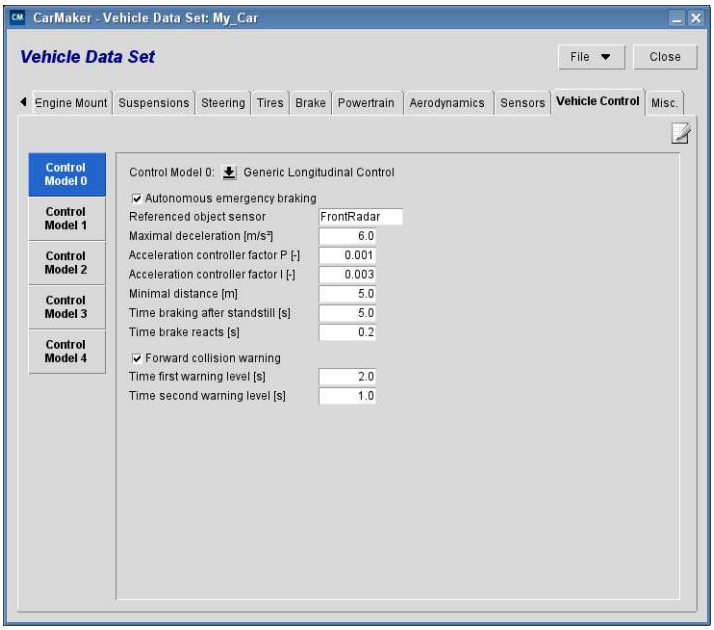


Figure 7.28: Vehicle Control selection

In the Vehicle Control tab you can select controller models like driver assistance systems. A maximum of five 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.

7.2.11 Misc.

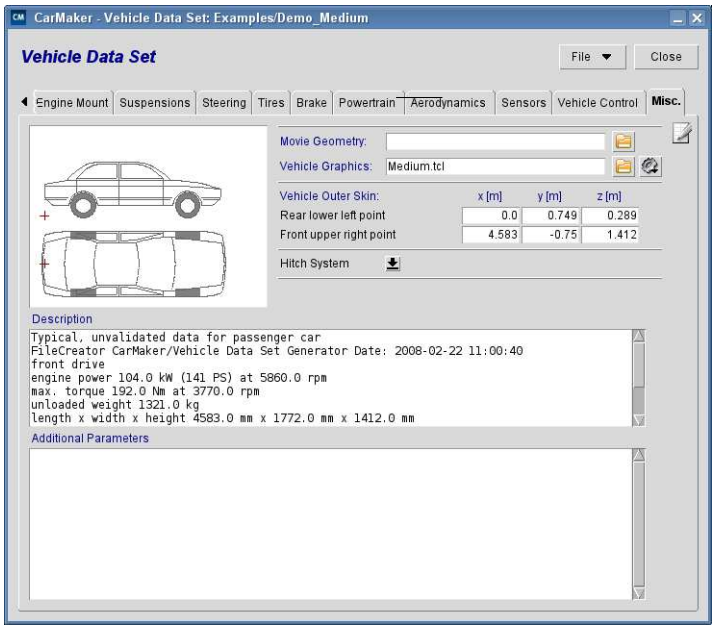


Figure 7.29: Misc. definitions

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 nods.
- 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.

7.2.12 Using the Import Feature

Within the Vehicle editor, CarMaker provides another unique feature. When building vehicle data sets, some vehicles may need to be parameterized that have partially common parameters to other vehicles. CarMaker makes it possible to import only specific, seperate parts of a data set from other vehicles.

Open the "My_Car" vehicle data set and import the front axle from the DemoCar:

In the Vehicle editor, click *File > Import*, select the modules "Kinematics front" and "Misc." and click "Import".
In the dialog "Import Vehicle Data Parameters from..." choose "DemoCar".
Save the vehicle data set ("My_Car") via *Vehicle Editor > File > Save.*

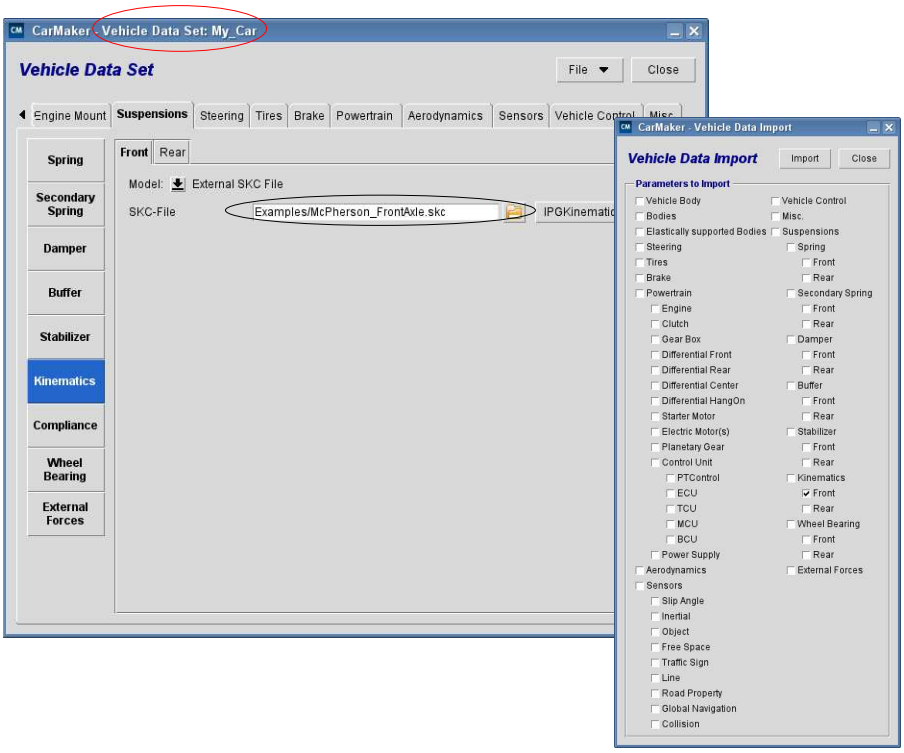


Figure 7.30: Vehicle Data Import for “my car”

The vehicle My_Car now uses the same non-linear kinematics description, picture and object file as the DemoCar. When the simulation is started again, the yellow “New Beetle” will be visible.

Save both the vehicle and TestRun.

7.3 Creating a new Tire Data Set

CarMaker provides the following 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
- Michelin TameTire
- IPGTire, for MotorcycleMaker only

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

These tire models (formerly known as TASS Delft Tire) are supported by CarMaker, but the models are not directly integrated in CarMaker. Thus, the tire model library is not part of the CarMaker installation, but can be linked dynamically. Input are the Magic Parameters similar to the standard Pacejka model (ADAMS property files are accepted, too).

Michelin TameTire

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).

In the CarMaker GUI, click on “Select“ in the tire box, 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 box, 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 > Pacejka > 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.
Load My_Step5 again.

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.

Next!

After having made some changes to the original configuration (*Step1*), the TestRun has been extended (*My_Step5,* which should be the same as *Step6_GenerateVehicle* in the examples). The next chapter describes how a completely new TestRun can be generated from scratch within five minutes!

Chapter 8

Building a TestRun from Scratch

This chapter will explain how to build the same TestRun that was created in section ’Extending the Test Scenario Definition’ from scratch.

8.1 Step 1: Opening CarMaker, Creating a TestRun

Open CarMaker.

Windows Start button > Programs > IPG > CarMaker

Create a new TestRun.

In the CarMaker main GUI, click on *File > New*.

Save under “My_TestRun“.

In the CarMaker main GUI, click on *File > Save As > My_TestRun > OK*.

8.2 Step 2: Loading a Vehicle

Load the vehicle data set “My_Car“ that was created in section 7.1.2 ’Using the Vehicle Generator’ on page 62.

In the CarMaker main GUI, click in the box *Car* on the button *Select“ > my_Car > OK*.

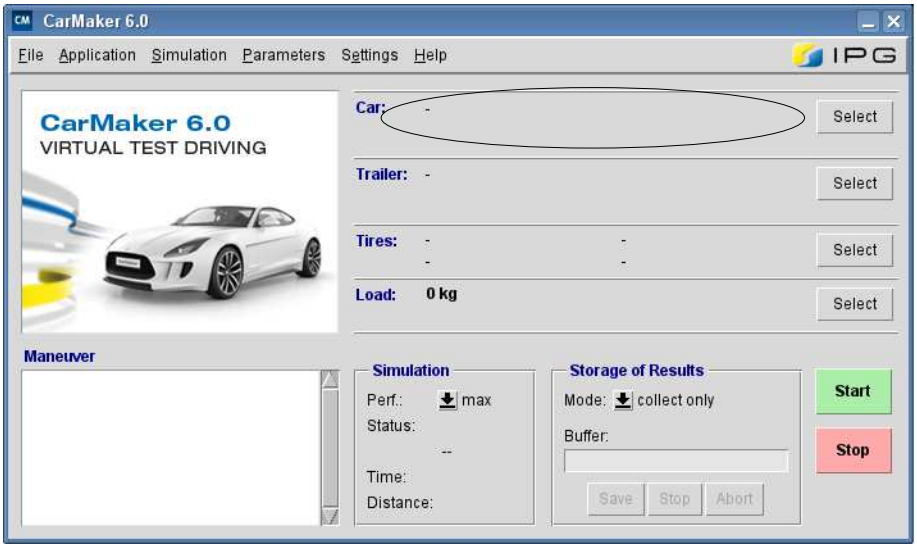


Figure 8.1: Selecting a car via the CarMaker GUI

8.3 Step 3: Putting the Vehicle on a Road

Define a road in the Sceanrio Editor as follows:

Open the Scenario Editor and click on the third icon in the top tab to import a road file. *Import road definition > Project > My_Road.rd5*.

To open the Scenario Editor: in the CarMaker GUI, click on *Extras* > *Sceanrio Editor*.

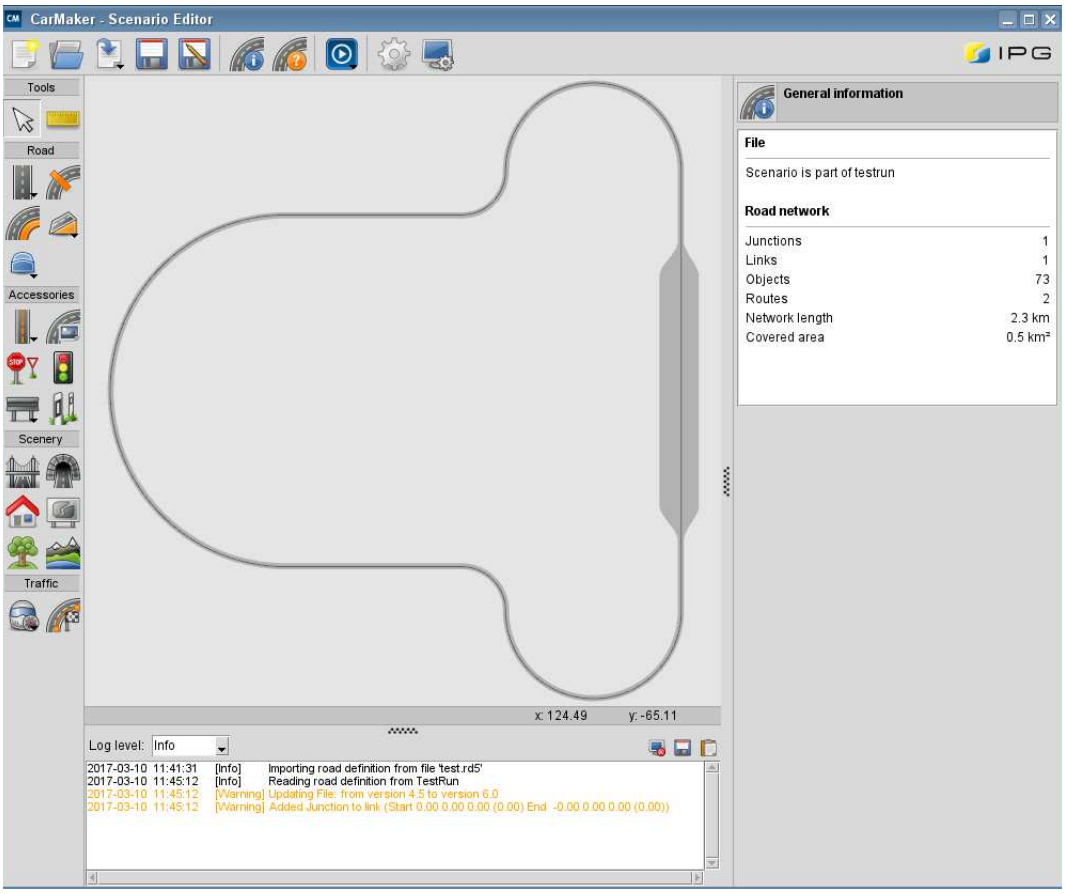


Figure 8.2: Uploaded road in the Sceanrio Editor

8.4 Step 4: Controlling the Vehicle (Maneuvers)

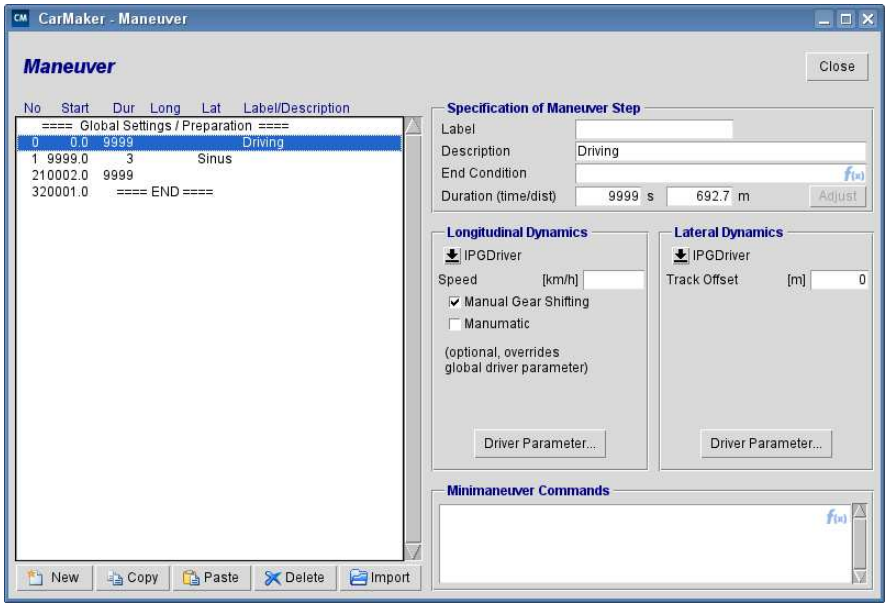


Figure 8.3: Maneuver window

Define the maneuver as follows:
In Maneuver GUI, click “Import“ at the bottom of the window > *My_Step2* > *OK*.

To open the Maneuver GUI, click on *Parameters* > *Maneuvers* in the CarMaker main GUI

8.5 Step 5: Saving the TestRun and Simulating

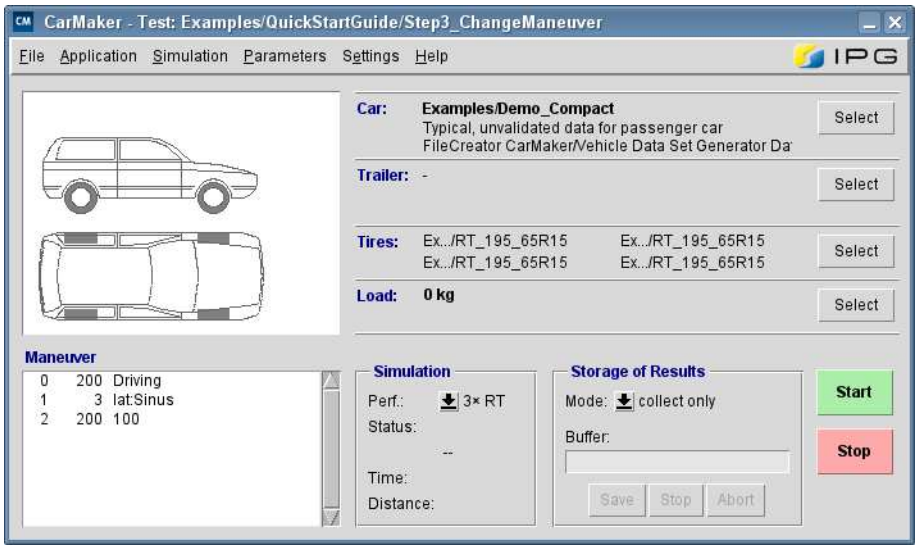


Figure 8.4: Saving the TestRun

Save the TestRun.

In the CarMaker main GUI, click on *File* > *Save*.

In the CarMaker main GUI, click on *Application* > *Start & Connect*.

This action initializes the simulation environment (useful for IPGControl).

Open IPGMovie, Instruments and IPGControl.

All these tools can be found by clicking *File* > *[name of the tool]* in the CarMaker main GUI.

In IPGControl, plot the vehicle speed during the simulation (*Quantities > Car[P..Y] > Car.v*).

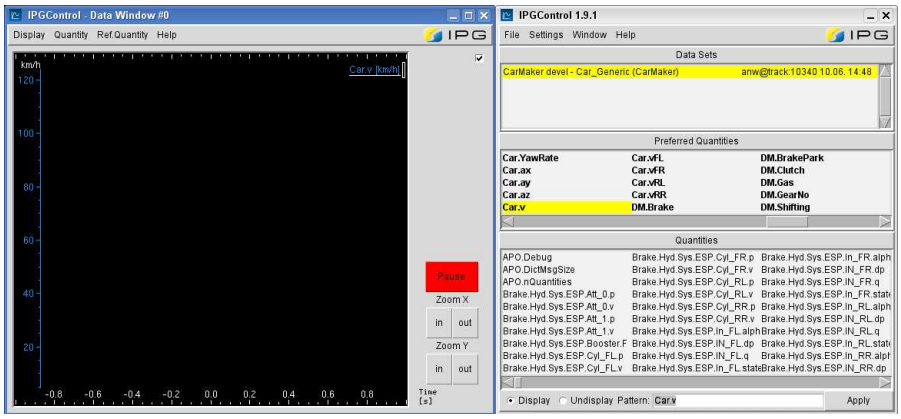


Figure 8.5: IPGControl Data and Selection window

Start the simulation.

8.6 Final Results

Loading a TestRun, parameterizing the environment, changing the tires, defining the maneuvers, creating a new vehicle data set, analyzing the results and watching the animation of the simulation. All these tasks took only a few clicks, a few minutes and the TestRun from the previous chapters could be rebuilt.

Next!

If the correct software is installed, CarMaker can be used in a MATLAB/Simulink environment. The next chapter will explain the most important features regarding CarMaker with Simulink.

Chapter 9

CarMaker for Simulink

9.1 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”.

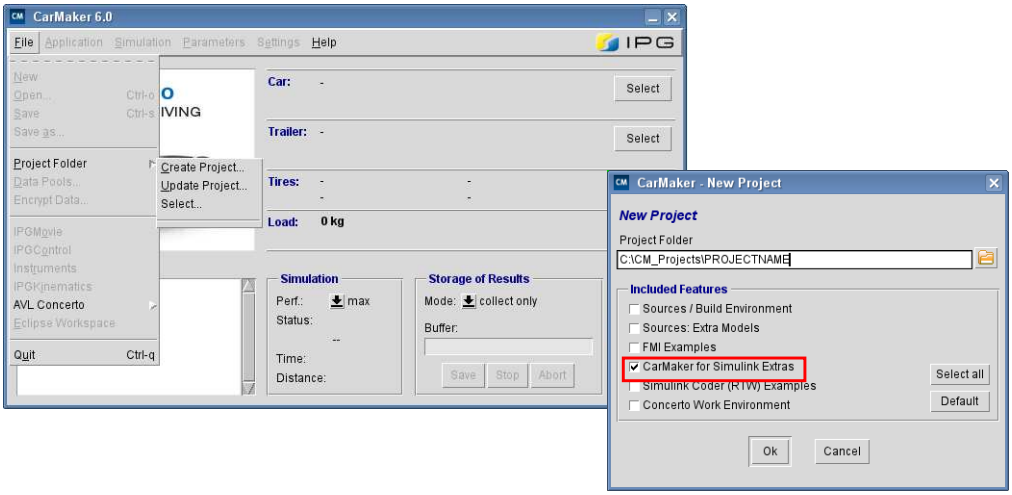


Figure 9.1: Creating a new project with “Carmaker for Simulink Extras”

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.

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:

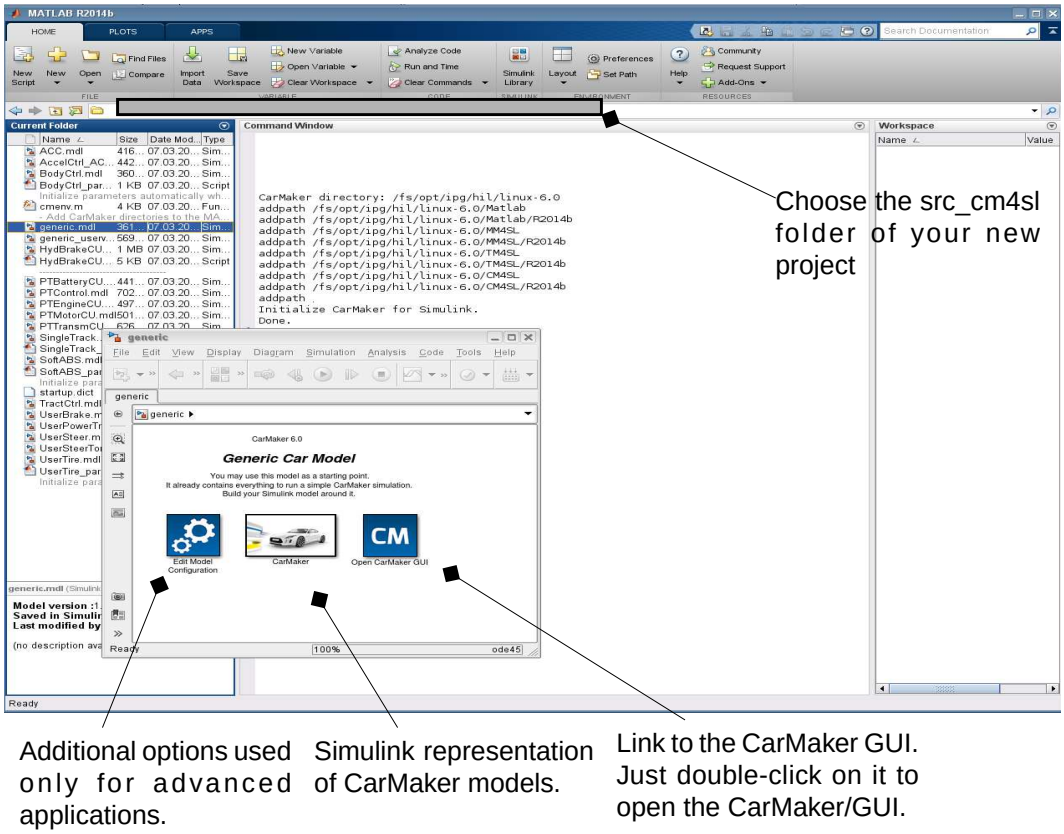


Figure 9.2: MATLAB window with Simulink model “generic.mdl”

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:

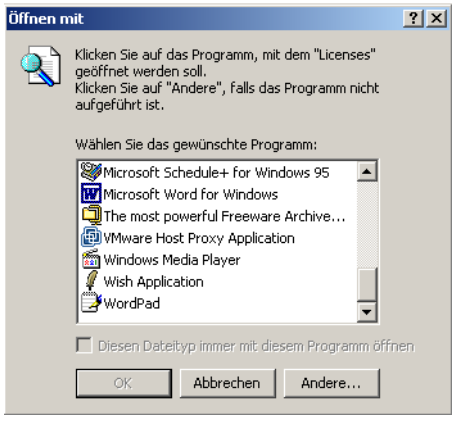


Figure 9.3: “Open with...” dialog window

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.
```

9.2 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 of this system will follow in the next section.

Double-click on the “Open GUI“ block and the CarMaker windows pop up:

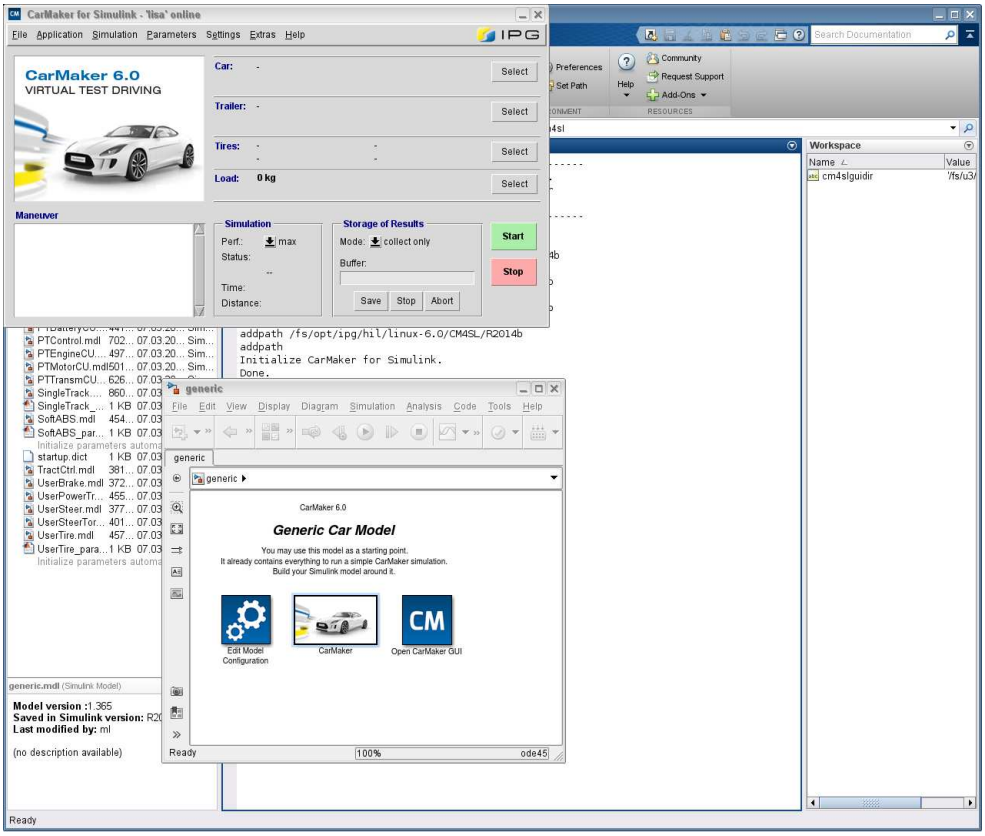


Figure 9.4: CarMaker GUI called up from MATLAB

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 > Road > ArtificialRoads > SegmentBasedClosedTrack.*

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.

Next!

CarMaker for Simulink is ready to work!

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

9.3 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.

9.4 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 10

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.

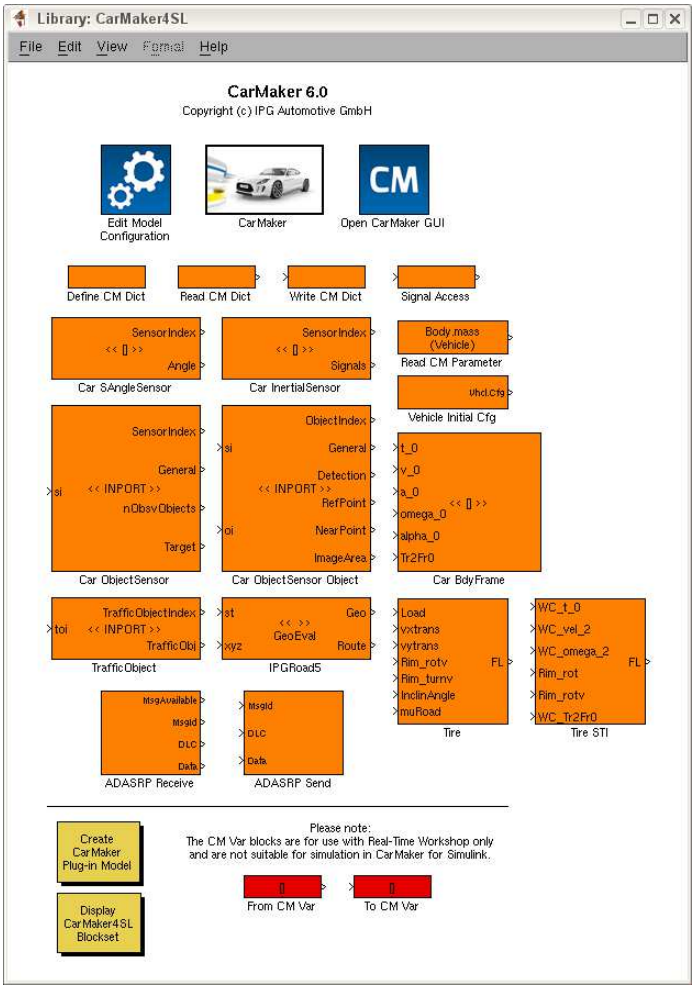


Figure 10.1: Simulink library

10.1 General Information

10.1.1 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 ot Simulink.

10.1.2 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 possible to:

- Replace, remove or reorder CarMaker blocks

For further detail, please refer to the demonstration examples and their description in this manual.

10.1.3 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. For 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.

10.2 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.

10.2.1 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, for 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.

10.3 CarMaker Dictionary Blocks

10.3.1 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.

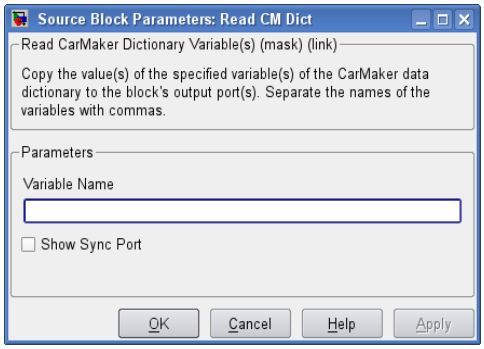


Figure 10.2: Read CM Dict block parameters dialog

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.
```

10.3.2 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.

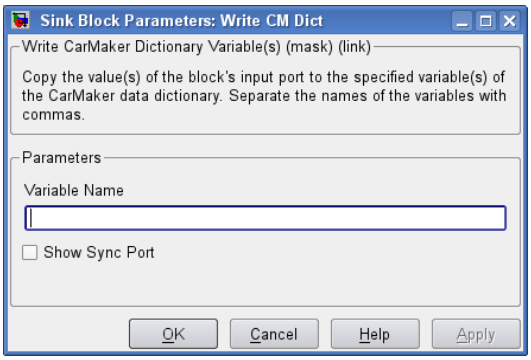


Figure 10.3: Write CM Dict block parameters dialog

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.
```

10.3.3 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.

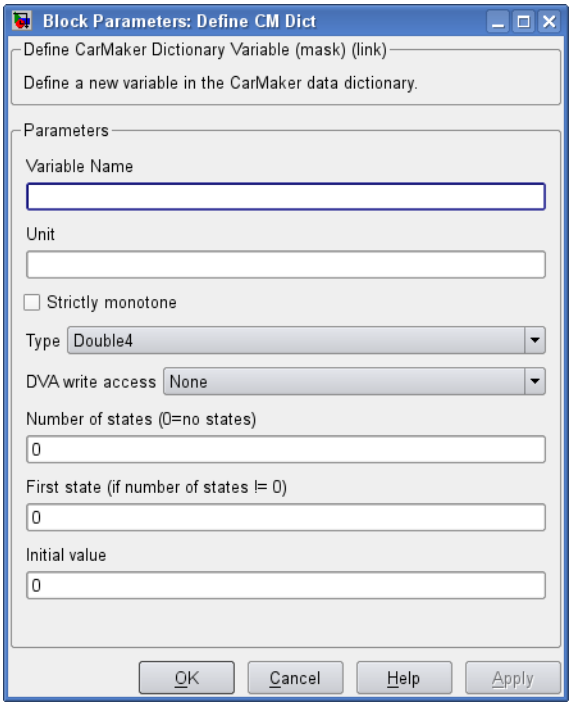


Figure 10.4: Define CM Dict block parameters dialog

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 *Stricly 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 six integer types. When in doubt, it is adviosed 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.

10.3.4 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, 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 11

How to extend the Simulink Model

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

11.1 Overview of the CarMaker Simulink Structure

Double-click the CarMaker block.

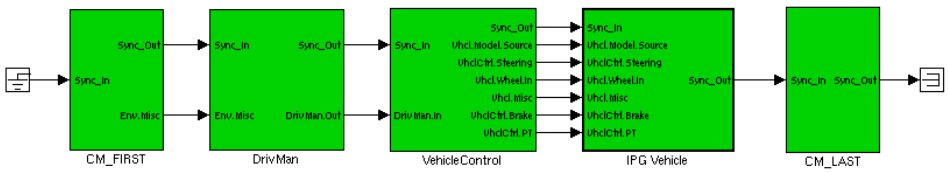


Figure 11.1: General structure of CarMaker in Simulink

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.

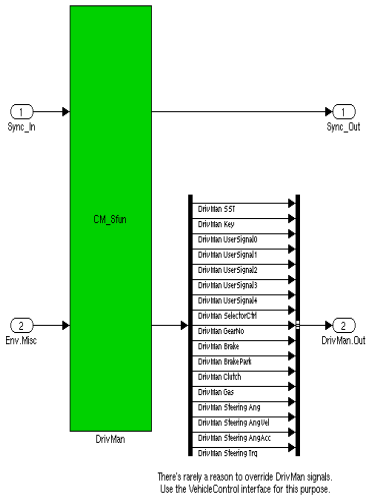


Figure 11.2: DrivMan block in Simulink

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.

Double-click the Vehicle block.

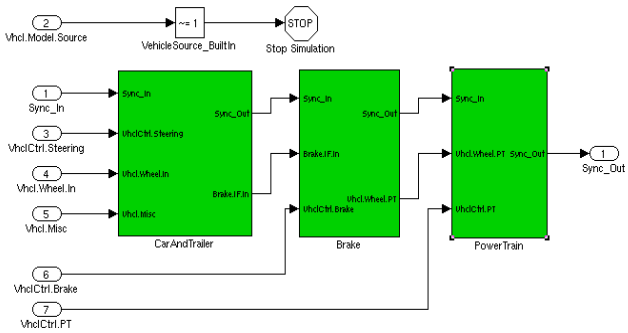


Figure 11.3: Vehicle block in Simulink

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

11.2 Example

11.2.1 Contents

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

11.2.2 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*.

11.2.3 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 neccessary 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 does'nt 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/Vehicle/PowerTrain subsystem.

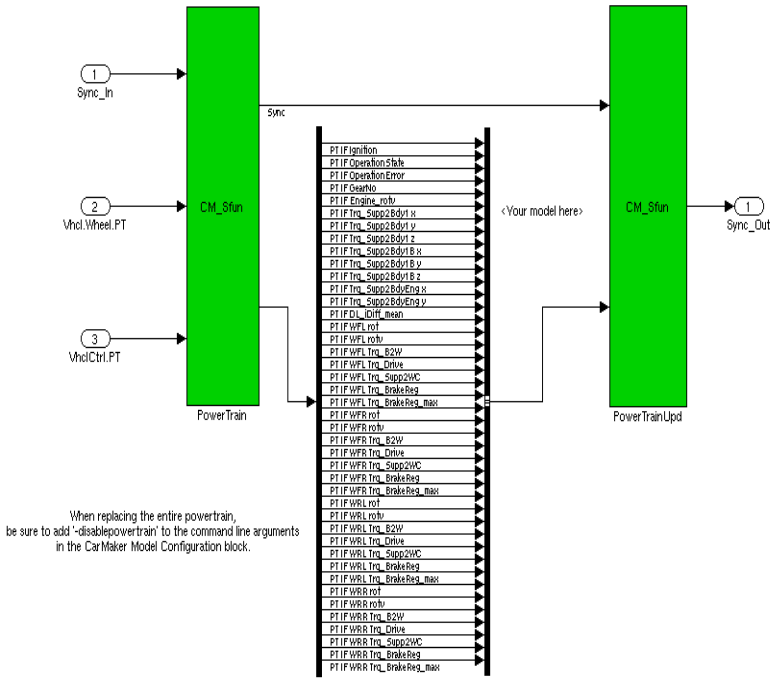


Figure 11.4: Powertrain Subsystem

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

11.2.4 Connect Inputs

The most important values of the simulation are the signals and busses between sub-models. If possible, these signalsshould 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 96.

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

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

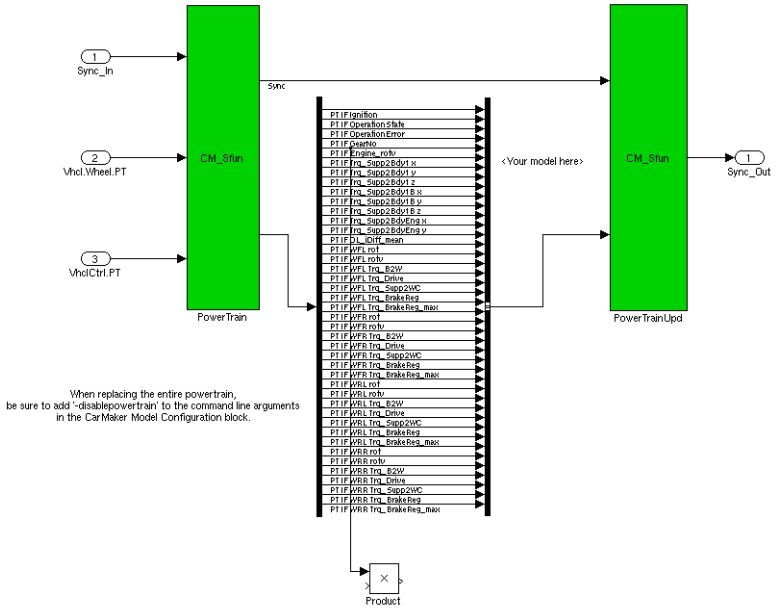


Figure 11.5: Multiplicator Block

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 96).

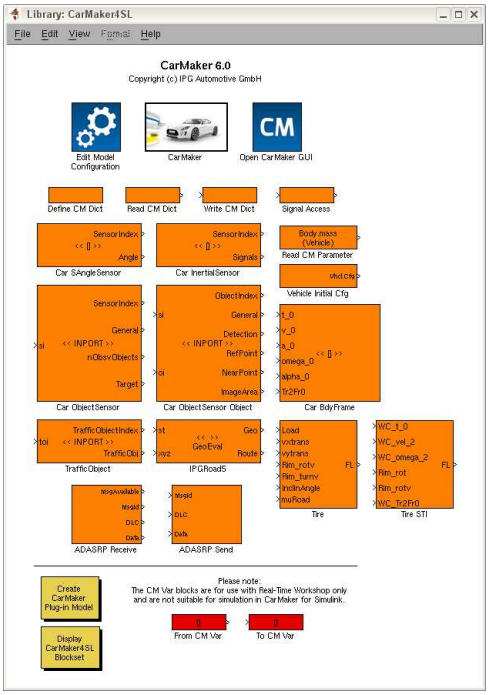


Figure 11.6: CarMaker for Simulink Library

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.

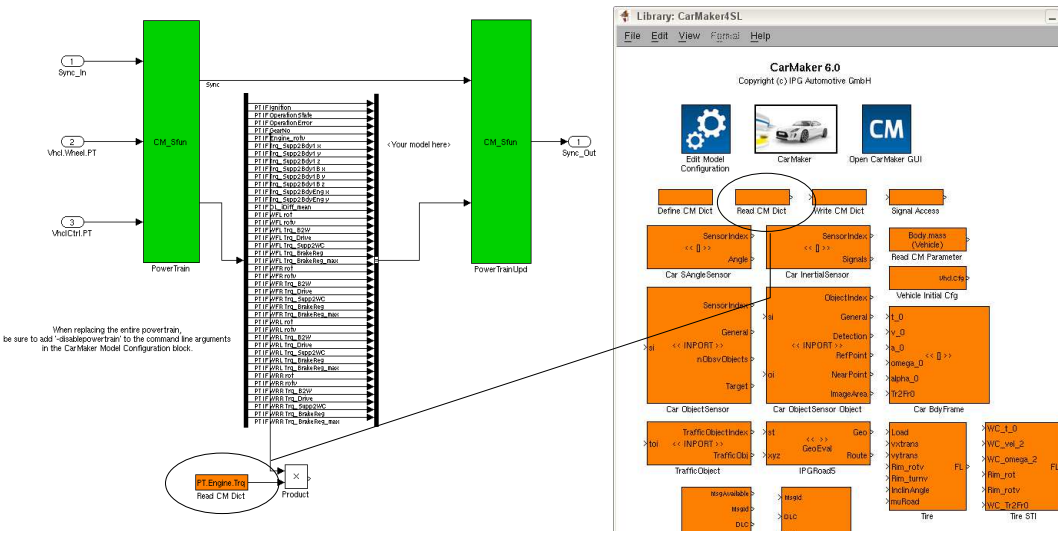


Figure 11.7: Read CM Dict

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“:

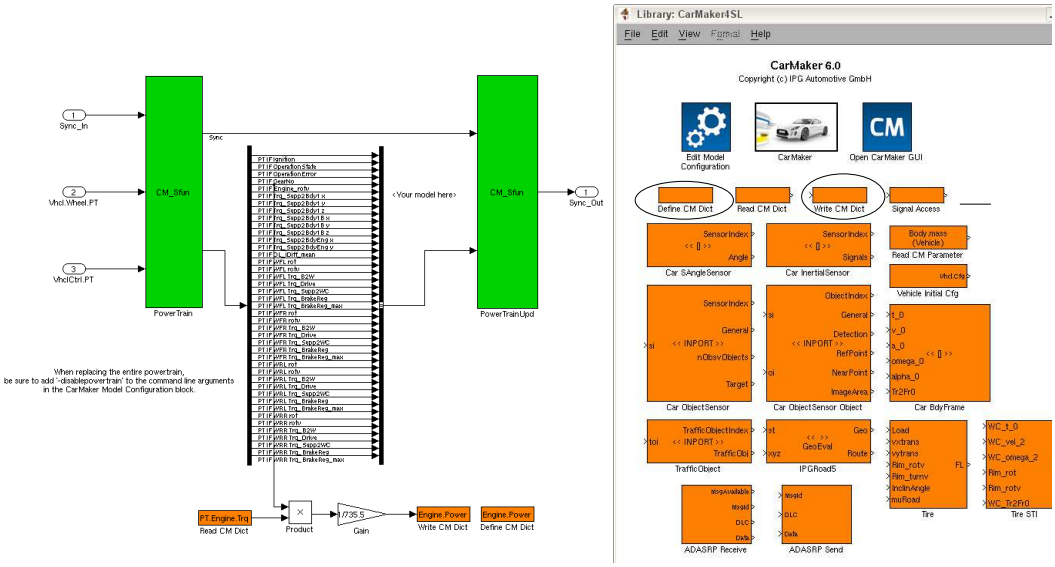


Figure 11.8: Engine Power

Now switch to the CarMaker GUI and load the TestRun *Product Examples* > *Examples* > *VehicleDynamics* > *_QuickStartGuide* > *Step1*. Open IPGControl and start the simulation.

The engine varibale that was just created can now be observed in IPGMovie.

11.2.5 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, for e.g. TestRuns, vehicle data, tire data, etc.

A detailled 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 12

Additional Information about CarMaker

12.1 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.

12.2 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*.

12.3 Feature List

DVA: How to Manipulate the Variables Online

CarMaker allows the user to check the values of specific variables online, for 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, for 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.*

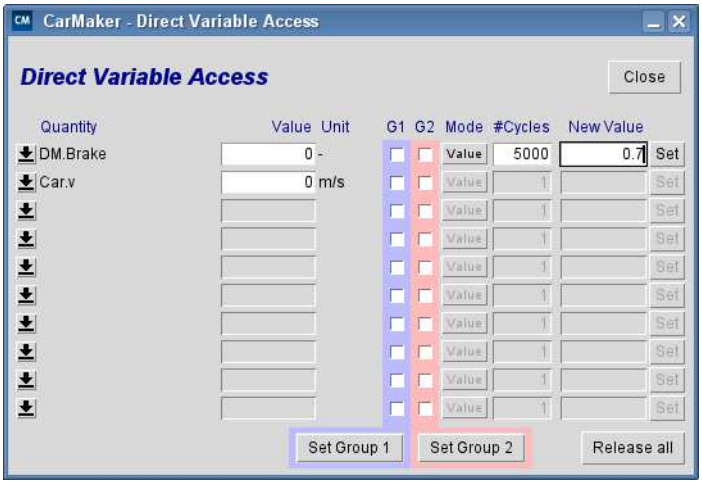


Figure 12.1: DVA window

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*.

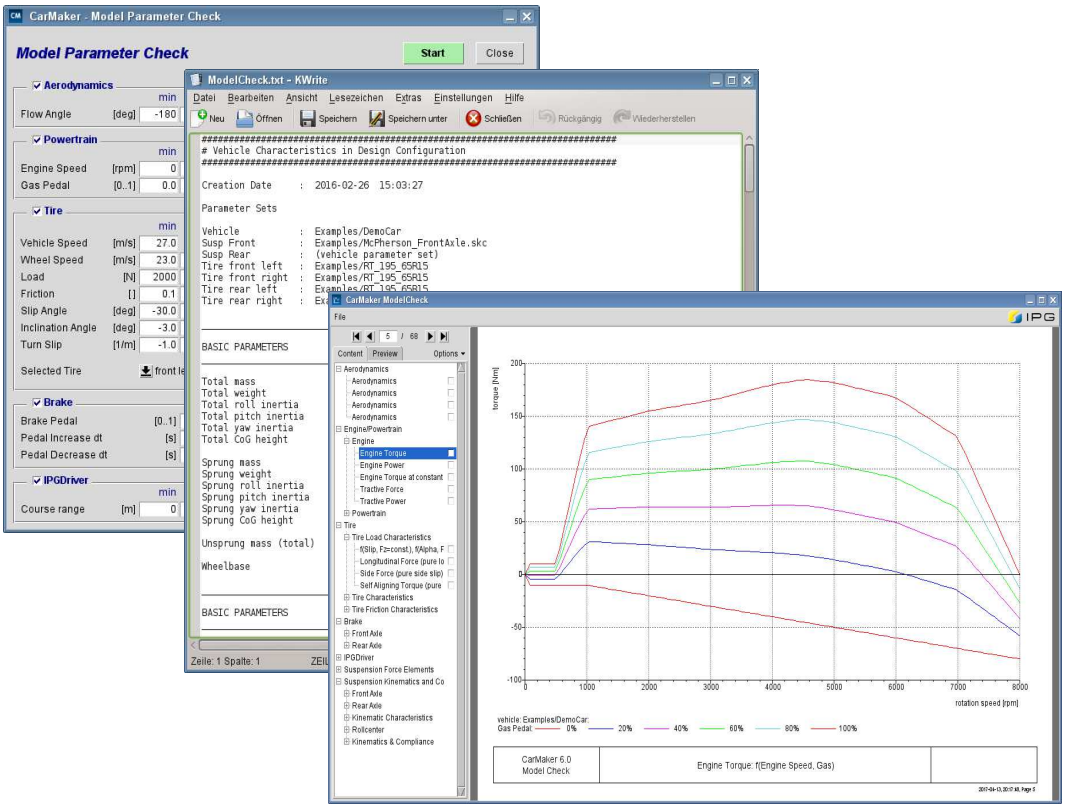


Figure 12.2: ModelCheck GUI

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.

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:

- Cylinders
- Beams
- Waves
- Cones

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* > *Road* > *Infrastructure* >*SideWind*.

Traffic Simulation

It is possibile 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 real 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.

For 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 ragarding 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

- 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 in ASCII files to be read with EXCEL.

Documentation of CarMaker

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