755 Phoenix Drive, Ann Arbor MI, 48108, USA
Phone: 734 668-2930 • Fax: 734 668-2877 • Email: info@carsim.com

Introduction to CarSim

This document describes the main features of the CarSim software and provides references to more detailed information within the extensive CarSim documentation.

Unless otherwise noted, all features that are described in this document for CarSim also exist in TruckSim and BikeSim. (Exceptions for TruckSim and BikeSim are noted in parentheses.)

Overview of CarSim

What Is CarSim?

CarSim is a software tool for simulating the dynamic behavior of passenger vehicles and light-duty trucks. It uses 3D multibody dynamics models to accurately reproduce the physics of the vehicle in response to controls from the driver and/or automation: steering, throttle, braking, and gear shifting. Environmental conditions include a 3D ground/road surface as well as aerodynamic and wind effects.

As a tool, CarSim is extensively validated and correlated to real-world results as measured and observed by many automotive OEMs around the world. The foundational technology upon which CarSim is based is called VehicleSim, abbreviated as “VS” when referencing other content in the product, e.g., the VS Visualizer (video and plotting) and VS Commands (scripting language).

High-Fidelity System-Level Vehicle Models: CarSim math models are represented at the system level, meaning the vehicle data is intended to be either measured or calculated and does not depend on detailed knowledge of component materials, suspension linkage intricacies, etc. The mathematics representing the vehicle are sufficiently detailed such that the simulation can replicate physically measured responses within the limits of testing repeatability. To accomplish this, data describing the vehicle such as kinematics and compliance of the suspensions, tire force and moment properties, and environmental conditions are required. In the case of limited reference data, the math models still provide representative results well-suited for evaluating alternative designs and control strategies.

Parametric Vehicle Definition: CarSim uses a combination of parameters and variables to represent the vehicle. As shown in Figure 1, parameters represent measurable properties such as dimensions and inertia properties. Configurable Functions relate variables in the model with linear coefficients or tables that use a variety of interpolation and extrapolation methods. Many of the tables support data obtained from suspension and tire test rigs. The tabular form is also convenient for other potentially nonlinear relationships that can be imported from spreadsheets, such as road geometry and friction.

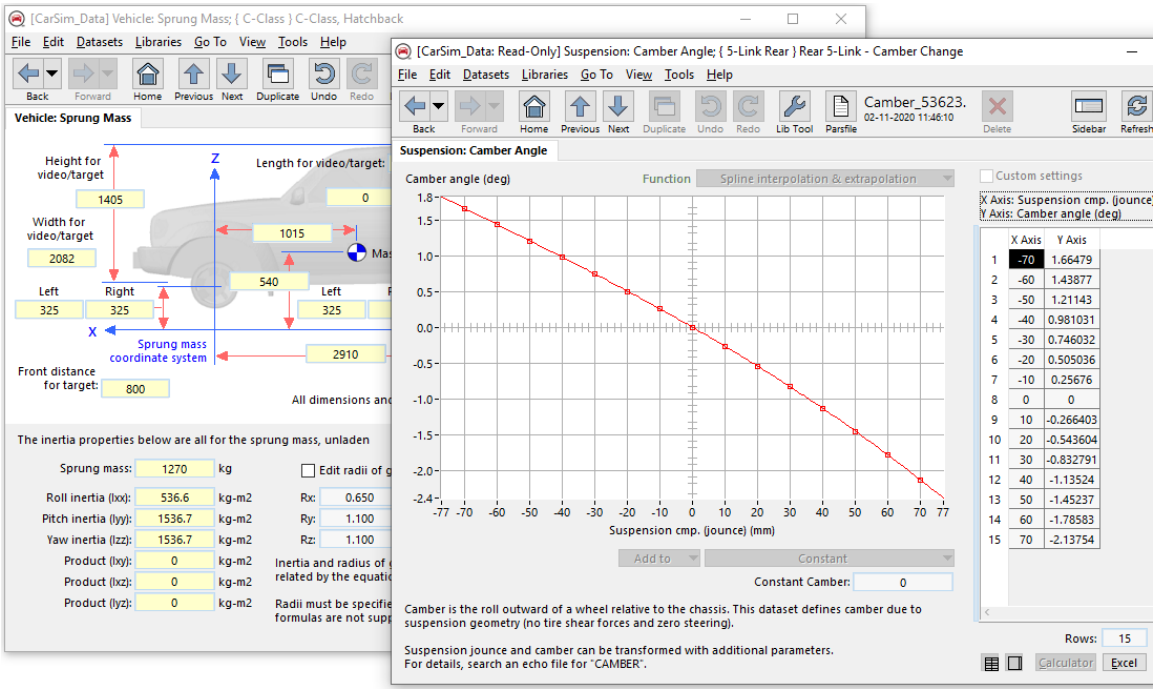

Figure 1. Parametric and tabular data used in CarSim.

ADAS and AV Support: CarSim includes moving "target" objects that are used to represent traffic vehicles, pedestrians, bicycles, etc. to simulate scenarios involving Advanced Driving Assistance Systems (ADAS) and/or autonomous vehicles (AVs). These target objects are detected by virtual sensors, with each sensor providing 24 calculated output variables for each possible sensor / target detection pair.

Database, User Interface, and Documentation: CarSim comes with over 600 example simulations, each consisting of one or more vehicles and a set of test conditions. These examples include over 40 example vehicles representing at least 10 unique vehicle configurations.

The Windows version includes a Graphical User Interface (GUI) called the VS Browser. Use it to run CarSim, set up parameters and tables, and view results. The different screens within the Browser correspond to different vehicle, environment, and procedure data stored within a database. Each screen has a **Help** button; clicking it will bring up a document describing the screen's function and various options. Technical memos, reference materials, and tutorials are available from the **Help** drop-down menu. All of the documents are PDF files contained in a Help folder that is indexed for rapid searching with Adobe Reader.

In the following material, references to documentation are made using the item on the **Help** menu. For example, a reference to the CarSim Quick Start Guide is: **Help** menu item: *Guides and Tutorials > CarSim Quick Start Guide.*

Plots and Visualization: CarSim includes VS Visualizer, a tool for providing high-quality animation of simulation results corresponding to engineering plots of hundreds (or sometimes thousands) of variables from the math model (Figure 2). Time-synchronization of the plots and video facilitates the viewing of both qualitative and quantitative data, and the option to overlay multiple runs (up to six total) allows for comparisons of various vehicle and control combinations.

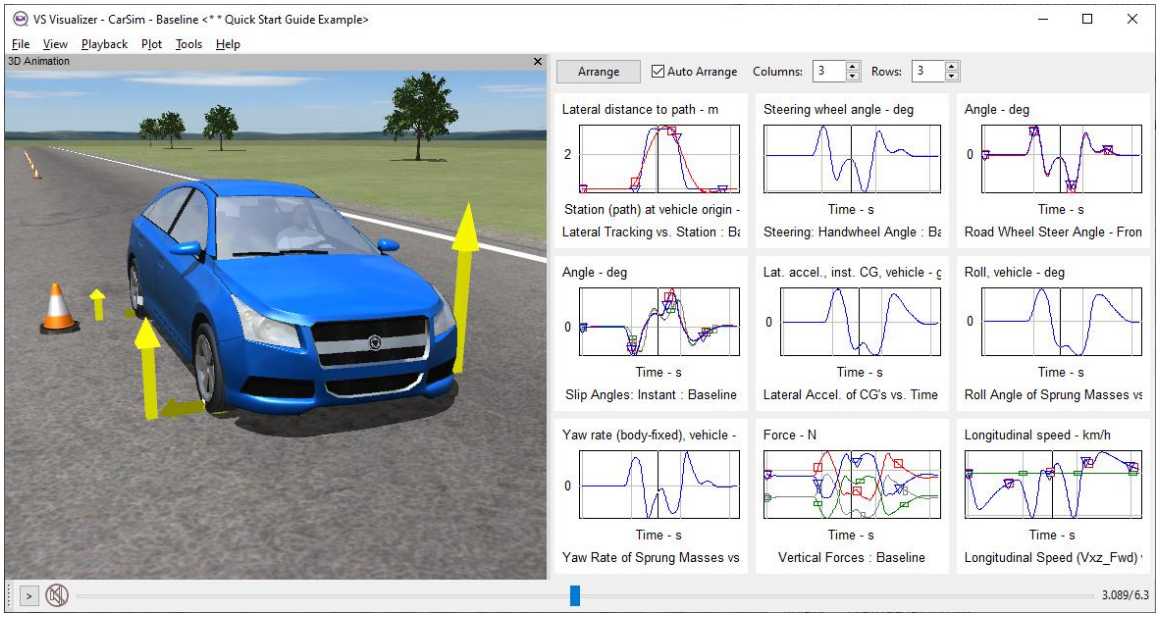

Figure 2. Viewing synchronized video and plot of CarSim vehicle response with VS Visualizer.

Fast Calculations: The CarSim math model is highly optimized for calculation efficiency and typically runs 15 to 20 times faster than Windows clock time, e.g., a simulation of a 60s test will finish in 3 or 4s on a modern Windows computer. The simulation execution speed may be reduced under high load cases such when many moving objects and sensors are added, generating thousands of outputs to be saved to file. Even so, the simulations still run faster than real time on Windows.

Installation: Installation of CarSim on Windows is done via a program called `Setup_CarSim_<version>_<revision>.exe`. Once installed, access to CarSim is typically via a shortcut on the desktop or the Start menu.

The installation includes over 60 archived and compressed "mini databases" that are used to build multiple custom databases as needed. (CarSim allows multiple database folders, in support of multiple projects.) Pre-defined combinations include a minimal database, basic or standard ADAS examples, basic or standard vehicle dynamics examples, and a "full" database with most examples that are not specialized. In additions, the specialized archives (e.g., trailers, third-party tire models, real-time platforms, software development kit (SDK), desktop driving simulator, etc.) may be used to build separate databases, or added tin any manner to make custom databases.

Basic Use of CarSim

There are certain basic features in CarSim that will be used for almost any application if you are running with the Browser and VS Visualizer on Windows. You will use CarSim to view existing simulation results and the data used to generate the simulations. You can also make new simulations using existing vehicle examples or create new examples by copying existing ones and modifying them.

Browser: Graphical User Interface (GUI) and database manager

The Browser uses GUI controls such as buttons, check boxes, drop-down lists, and text fields.

Run Control: The Home screen (Figure 3) shows the main screen of the CarSim Browser. The CarSim GUI screens are organized into libraries in the database folder. The current library name is shown in the Windows title bar (1), along with the title of the dataset in view (2).

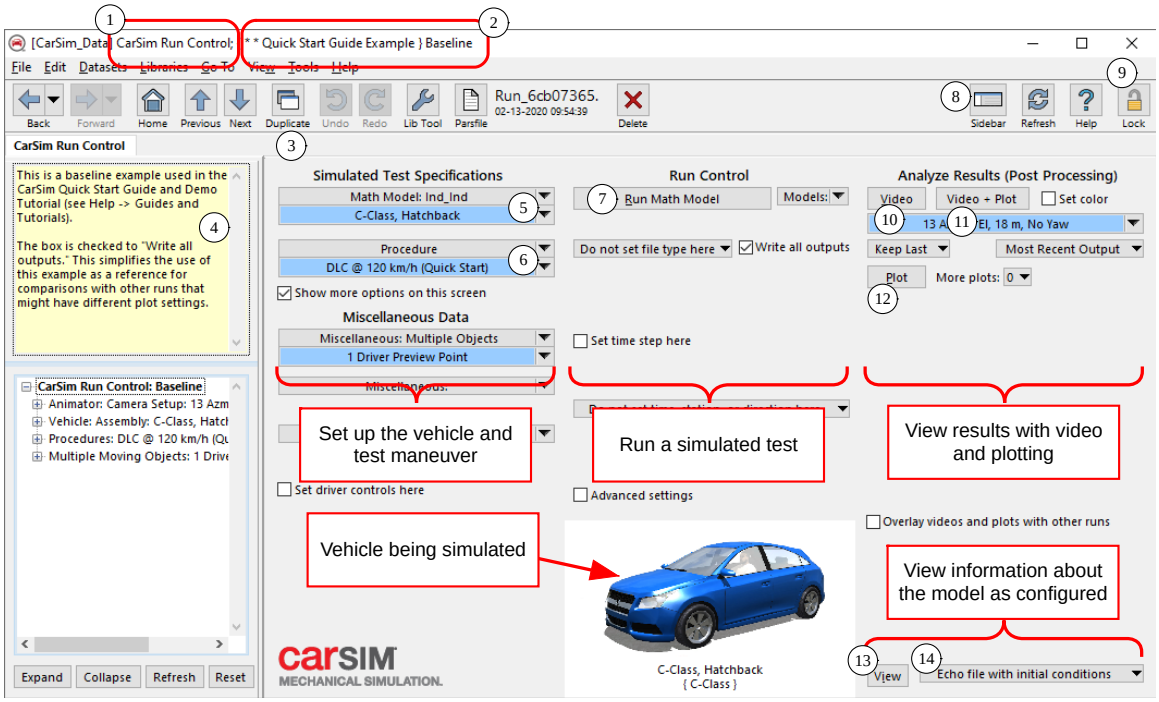

Figure 3. The Run Control screen in CarSim (Windows).

There is a row of buttons below the menu bar. The **Duplicate** button (3) is used to copy the current dataset before making changes to accommodate a new set of data. Locations for editable text are identified with a yellow background (4) if the dataset is unlocked (9). The **Sidebar** button (8) is used to show or hide the region on the left that includes optional notes (4) about the current dataset and a tree view of the current simulation.

The drop-down control in the lower right corner (14) contains information about the current simulation, viewable (13) using a text editor or spreadsheet (Figure 4).

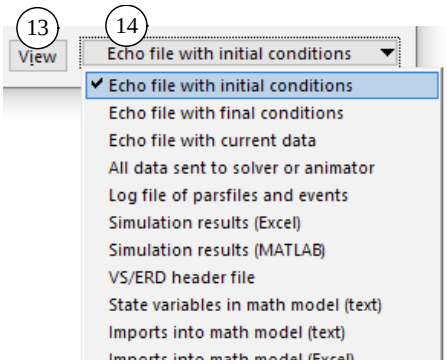

Figure 4. Drop-down control for View options.

The first item on the list (Echo file with initial conditions) shows a full description of the model when the simulation begins (Figure 5).

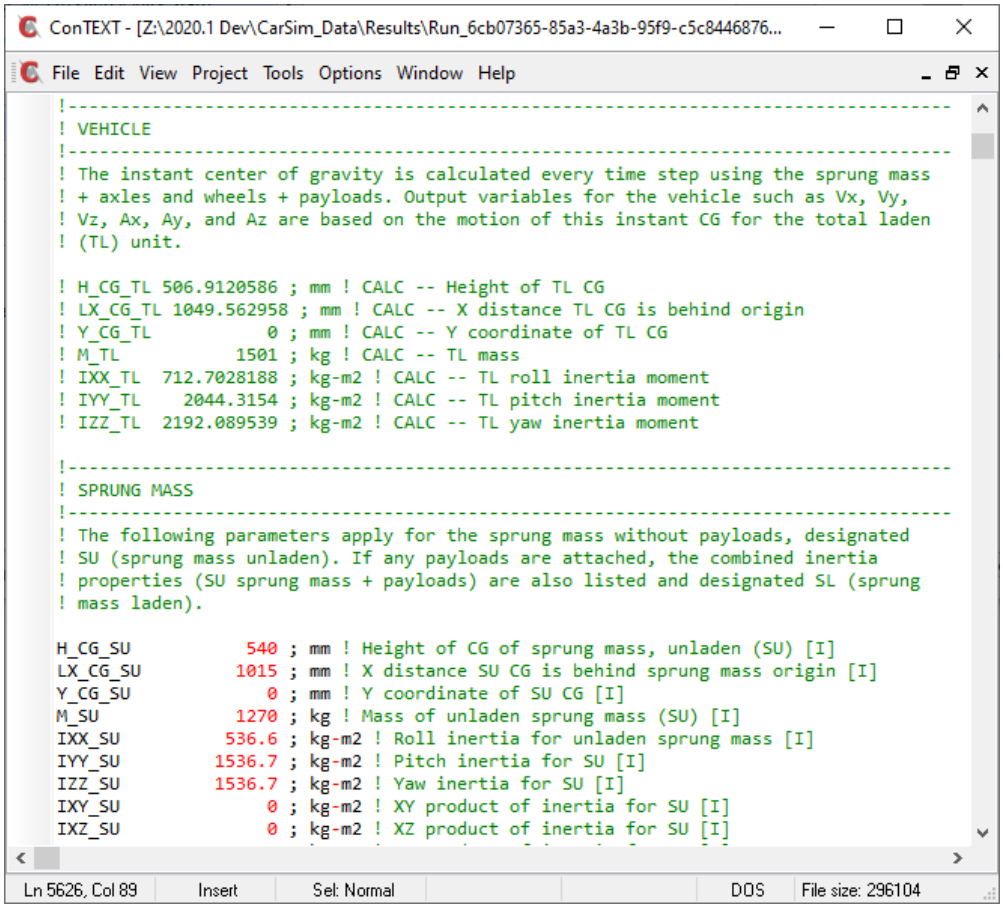

Figure 5. Portion of Echo file listing all simulation data.

The Echo file is organized into sections describing different parts of the model such as the Sprung Mass, Suspension, Powertrain, Tires, Roads, Paths, and more.

The GUI makes extensive use of blue configurable hyperlinks, similar in function to hyperlinks seen with Web browsers (Figure 6). In the same manner as a web link, hovering the mouse cursor over a blue link (5) will underline the text. Here, clicking on the blue link from the **Run Control** screen will show the linked **Vehicle: Assembly** dataset (Figure 7).

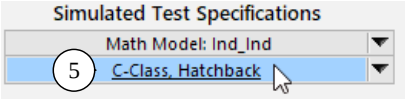

Figure 6. Blue link to another dataset.

The hyperlink control is extended in two ways using attached drop-down controls. Consider the blue link in the upper left corner of the **Vehicle: Assembly** screen (Figure 8), used to specify the Sprung Mass data for the vehicle.

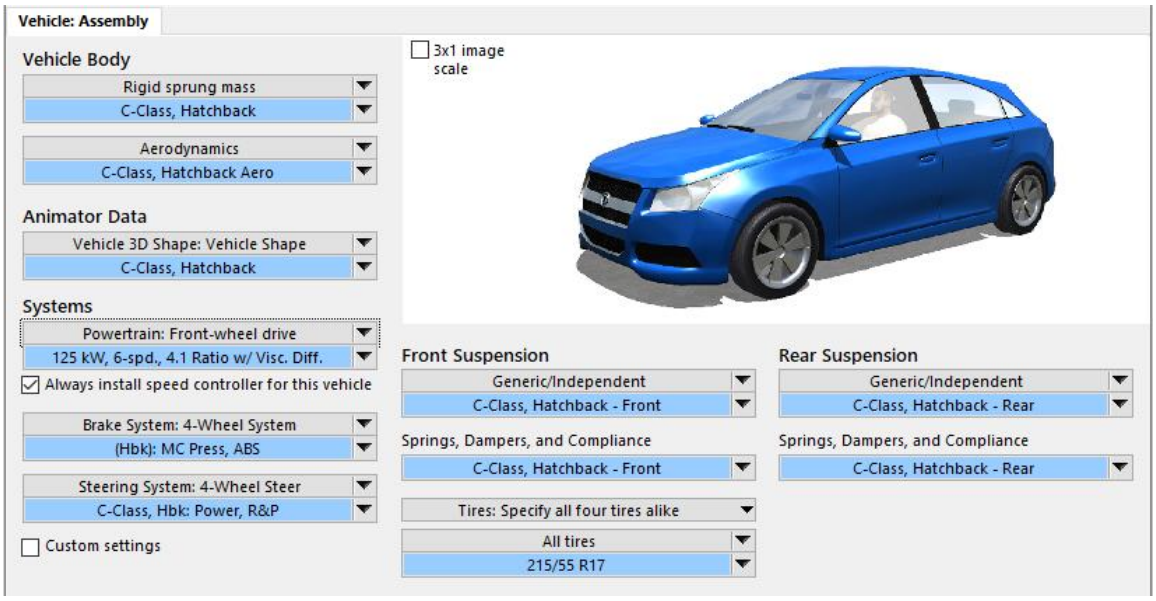

Figure 7. A Vehicle: Assembly dataset used in the example simulation setup.

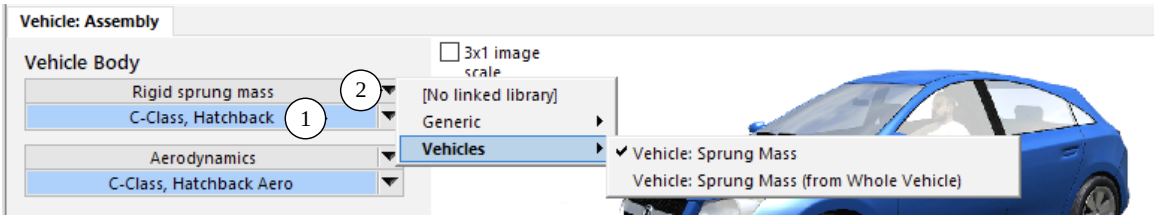

Figure 8. Libraries available for a blue link.

The **Sprung Mass** blue link (1) serves as a link to the underlying dataset; clicking it will switch to the linked **Sprung Mass** screen. The top control (2) is a drop-down menu used to select a library in the database. In this example, CarSim is shown to have two ways of representing the Sprung Mass data: **Sprung Mass** and **Sprung Mass (from Whole Vehicle)**. The top control is used to choose between these two libraries.

The drop-down control to the right of the blue link (3) (Figure 9) shows all datasets in the selected library when pressed. Choose any dataset name from the drop-down control to configure the blue link to use that dataset or copy an existing dataset to make a new dataset that can be modified as needed.

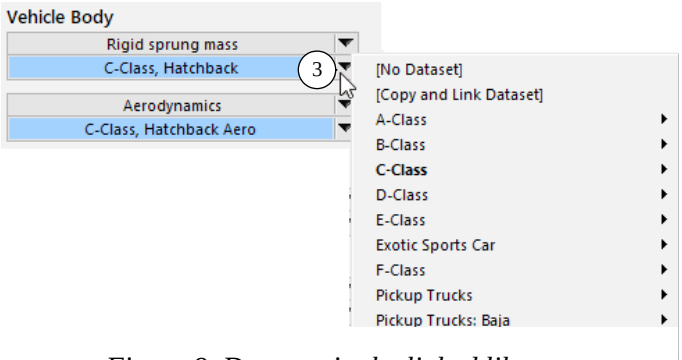

Figure 9. Datasets in the linked library.

More instructions about the GUI controls are provided in the **Help** menu item: *Guides and Tutorials > CarSim Quick Start Guide*. The full reference for the GUI is in the **Help** menu item: *Reference Manuals > VS Browser (GUI and Database)*.

Parsfiles

The data for each screen is contained in a text file called a *Parsfile*. Each screen has a button (1) (Figure 10) that you can click to view the Parsfile. Text in a yellow field in the GUI will appear in the Parsfile preceded by a keyword that identifies a corresponding parameter in the model.

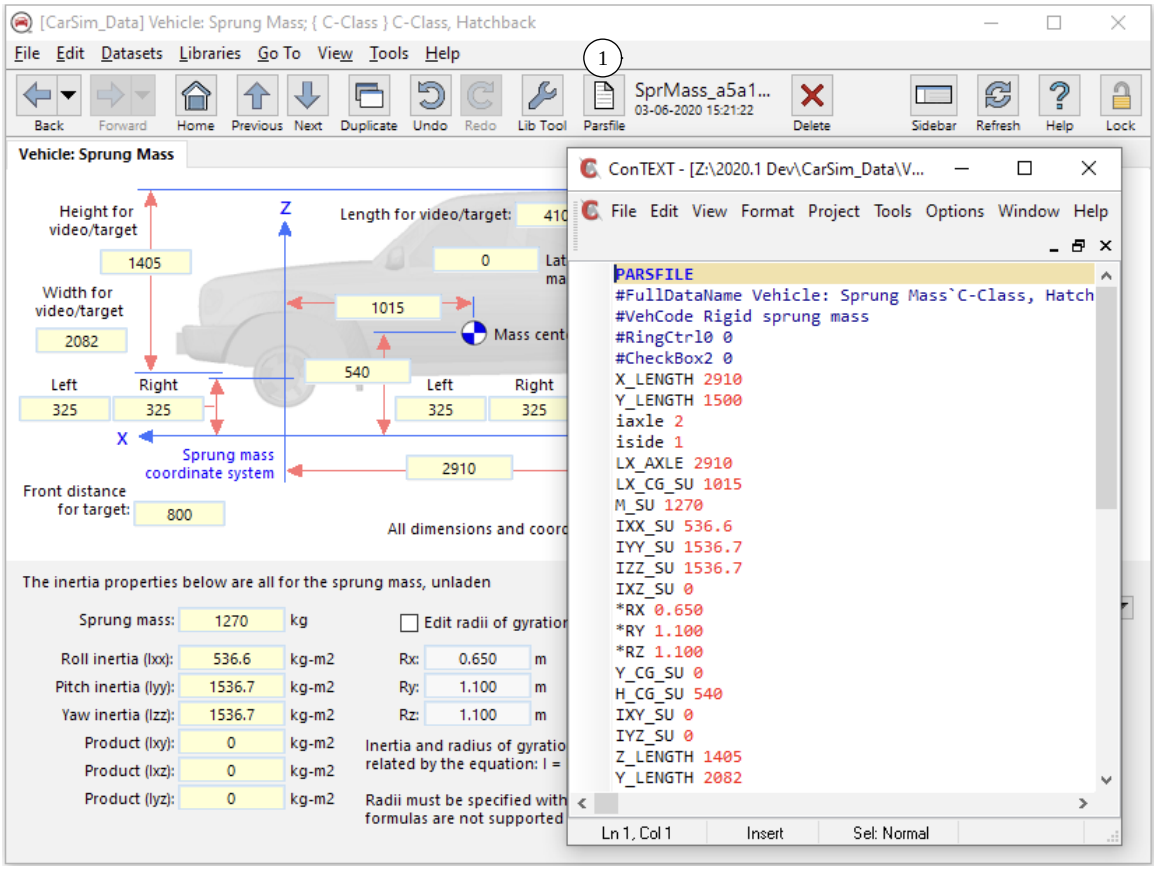

Figure 10. The dataset for each screen is contained in a Parsfile.

The Parsfile format is also used for Echo files. This is done to support applications where simulations are continued from previous runs using the Echo files as an input.

Running the Simulation

You run a new simulation by clicking on the **Run Math Model** button on the **Run Control** screen (7, Figure 3, page 4). When you do this, CarSim performs several steps to run the simulation:

1. The Browser creates a large Parsfile starting with the **Run Control** Parsfile, plus all Parsfiles linked to the **Run Control** dataset (vehicle, procedure, etc.), plus all Parsfiles linked to those, and so on. This Parsfile is called the *All Parsfile*.

2. The Browser writes a simulation control file called a *Simfile* which lists the full pathname for the All Parsfile, plus pathnames for requested output files.

3. The Browser loads `carsim_32.dll` (a VS Solver library) and uses VS Modularity functions to read the Simfile and All Parsfile, and construct a VS Math Model for the simulation.

4. The VS Math Model write an Echo file (as seen in Figure 5 on page 5) after reading all input data, then calculate svariables over the duration of the simulation while writing to output files, and then generates another Echo file when the run ends.
5. The Browser unloads the DLL.

The outputs from the VS Math Model are written into a file with a specified type (binary or text), sample interval, and selection of outputs. By default, only outputs needed for animation or specified for plotting are written to file. Other options are to write all outputs, or to write a set of specifically selected outputs.

Comprehensive information about the VS Solver library is provided in the **Help** menu item: *Reference Manuals* > *VS Solvers*, including file handling, syntax, numerical methods, and more.

VS Visualizer

VS Visualizer is launched by clicking one of three buttons on the **Run Control screen** (Figure 3, page 4): **Video** (10), **Video + Plot** (11), or **Plot** (12). If video is to be shown, the camera settings are obtained from an **Animator: Camera** dataset located below the **Video + Plot** button (11). Plot datasets are typically linked on the **Procedures** screen (6) and can represent either the time history of an output variable or a cross-plot between any two output variables.

Clicking the **Video + Plot** button (11) to launch VS Visualizer will bring up a window showing a camera view and a grid of plots (Figure 2, page 3).

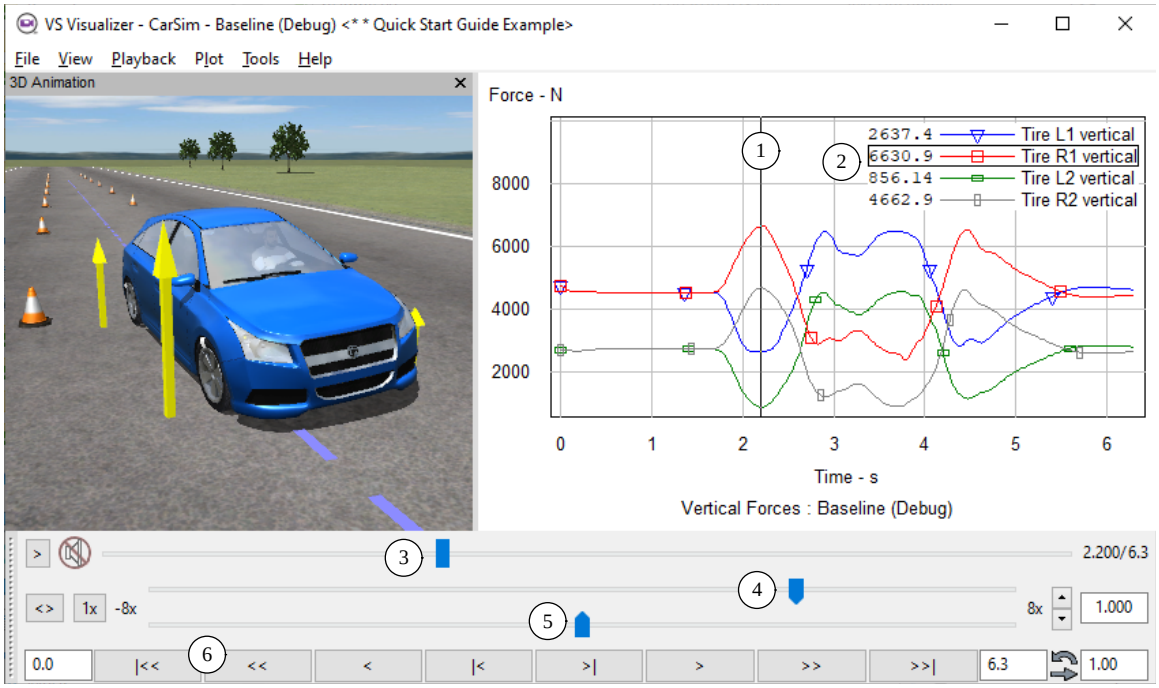

Figure 11. VS Visualizer interactive controls.

Any plot can be quickly expanded to view details (Figure 11). For example, a vertical line indicates the current time stamp (1) with corresponding animation, and digital values are shown in the plot

legend for that time (2). If the animation is paused, a slider control at the bottom of the window can be used to move the time forward and backward (3).

VS Visualizer supports the overlay of both animation and plots (3), (4) (Figure 12), effectively demonstrating the difference in results. A virtual heads-up display (HUD) option (2) is available for emulating driver alerts and showing calculated math model output. Visual effects such as skid marks are also supported (1).

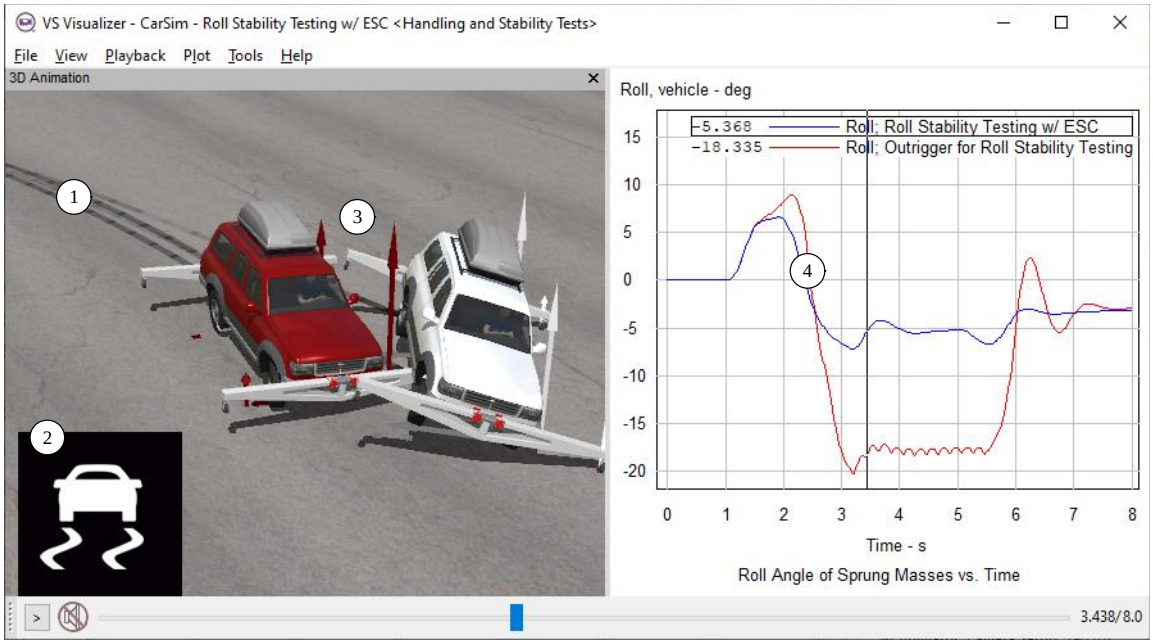

Figure 12. VS Visualizer shows overlays of multiple runs, heads-up displays, and skid marks.

Help menu items: *Guides and Tutorials* > *CarSim Demo Tutorial* and the *CarSim Quick Start Guide*. Please also see *Reference Manuals* > *VS Visualizer Reference Manual.*

CarSim Advanced Features

CarSim is used in a wide variety of simulation applications, from classical vehicle dynamics through ADAS and autonomous vehicle development. There are many features which might be considered advanced by some users, but which are absolutely essential for other users. This section summarizes some of these features.

Vehicle Options and Controls

CarSim is a self-contained simulation tool that does not require any other software to run. The vehicle models include the fundamental multibody physics and vehicle component equations (tires, suspensions, powertrain, aerodynamics), as well as open-loop and closed loop control options. The built-in options for extending the model are described below.

Vehicle Math Model Options

The VS Math Model constructed by CarSim always includes vehicle dynamics for the selected type of vehicle. Options listed here are added to the model based on commands written into the screen parsfiles (either by linking to a screen or selecting an option on a screen).

Trailers: Trailers with one, two, or three suspensions can be added to the vehicle model and are supported by dedicated GUI screens. An extra license is needed to run a simulation with trailers. **Help** menu item: *Vehicles*. (TruckSim includes trailers in the basic license; BikeSim does not support trailers.)

Payloads: Up to 99 payloads may be added to the model, attached to the sprung mass of the motor vehicle and/or trailer. **Help** menu item: *Payloads*.

Dual tires: The default wheel type in CarSim is for single tires. However, dual tires also available. Documentation about dual tires is available throughout the Help system, including the **Help** menu item: *Tire Models*. (TruckSim always uses dual tires; BikeSim does not support dual tires.)

Custom forces and motion sensors: Up to 99 motion sensors may be added to a simulation and attached to sprung mass or unsprung mass bodies. Up to 27 output variables per sensor are available, related to the motion of a point of interest in the vehicle (position, velocity, acceleration, jerk, curvature).

There is also an option to add up to 99 points where user-defined forces may be applied. Custom moments may also be specified for the sprung masses and all non-spinning wheel bodies (hubs). The force and moment magnitudes may be defined using the built-in scripting language VS Commands (see page 11), or imported from external software (see page 18). **Help** menu item: *Model Extensions and RT > Custom Forces and Motion Sensors*.

Driver Controls

CarSim has closed-loop controllers for steering and speed, allowing the vehicle to follow a target path at an appropriate speed. CarSim supports up to 500 paths and can switch between them as needed, so it is easy to simulate most scenarios that involve driver control.

Steering controller: The CarSim closed-loop steering control steers the vehicle to follow a prescribed path. Several modes of operation are available: steer by angle, steer by torque, steer with a single preview point, or steer with an optimal control method (10 preview points). The single point controller works while driving forward or in reverse. Switching between modes, including open-loop control (angle or torque), is available at any time during the simulation. (BikeSim has a rider controller that both steers and leans to follow a path.)

Speed controller: The CarSim closed-loop speed controller applies throttle and braking to achieve a target speed. Target speed can be a constant or a function of time and station (distance along a path). The speed controller can also determine speed based on a path preview, considering limits in lateral and longitudinal acceleration, as well as lateral and vertical curvature. Options exist for three driver skill levels. **Help** menu item: *Controls > Driver Control Screens*.

Electronic Brake Systems (EBS)

ABS: CarSim includes a basic antilock brake system (ABS) control that is supported by the GUI. If selected, a command is automatically applied to add the system. **Help** menu item: *Brake System*.

ESC: CarSim includes a basic electronic stability control (ESC) control that is supported by the GUI. If selected, a command is automatically applied to add the controller. **Help** menu item: *Controls > Electronic Stability Control*. (BikeSim does not have built-in ESC.)

Model Extension with VS Commands and Python

VS Commands

The VS Math Model may be extending using a scripting language called VS Commands.

Formulas: Numerical values for parameters in the VS Solver are normally provided as numbers. VS Solvers also work with formulas involving numbers (e.g. 1/16) or names of parameters and variables. The formulas may use common math functions, and arithmetic and Boolean operators.

New variables: New variables may be added using VS Commands such as `define_parameter`, `define_output`, `define_import`, and others.

New equations: New equations can be added that involve existing variables and new variables added with `define_` commands. Commands are available to insert the equations at various points in the calculation sequence, from initialization (`eq_init`) to various places in the time step (`eq_in`, `eq_dyn`, `eq_out`, `eq_full_step`, etc.). New degrees of freedom can be added with ordinary differential equations using the `eq_differential` command.

New functions: New functions can be created for use in the formulas applied in VS Command equations.

Events: The `define_event` command is used to define a condition using a formula; when the condition is non-zero (Boolean true), a new Parsfile is read that changes the simulation. Figure 13 shows a run where `define_event` and other VS Commands are used to set controls for the vehicle in response to detections of road signs, bicycles, and pedestrians.

Manage units: Units can be changed for parameters and variables, and new units can be introduced with the `define_units` command.

Linearize: The VS Command `linearize` will numerically perturb the state variables calculated with differential equations to derive linear A, B, C, and D matrices, and print a MATLAB text file. MATLAB can then be used to determine properties of the linearized system, such as Eigenvalues and plots such as root locus and frequency response.

Save and Restore states: The VS Command `save_state` will take a snapshot of the state of the entire math model and save it in memory. The command `restore_state` will later completely restore the state. The commands can be applied in Events to support highly efficient optimizations in which parameters or other model properties are changed, and the simulation jumps back in time to try again.

Help menu items in the *Reference Manuals submenu: VS Commands* and *VS Commands Summary*.

Embedded Python

CarSim on Windows and Linux (non-RT systems) support embedded Python. A Python file may be loaded and enabled with the command `run_python_string`. Python functions that have been loaded are accessed with the command `Python`().

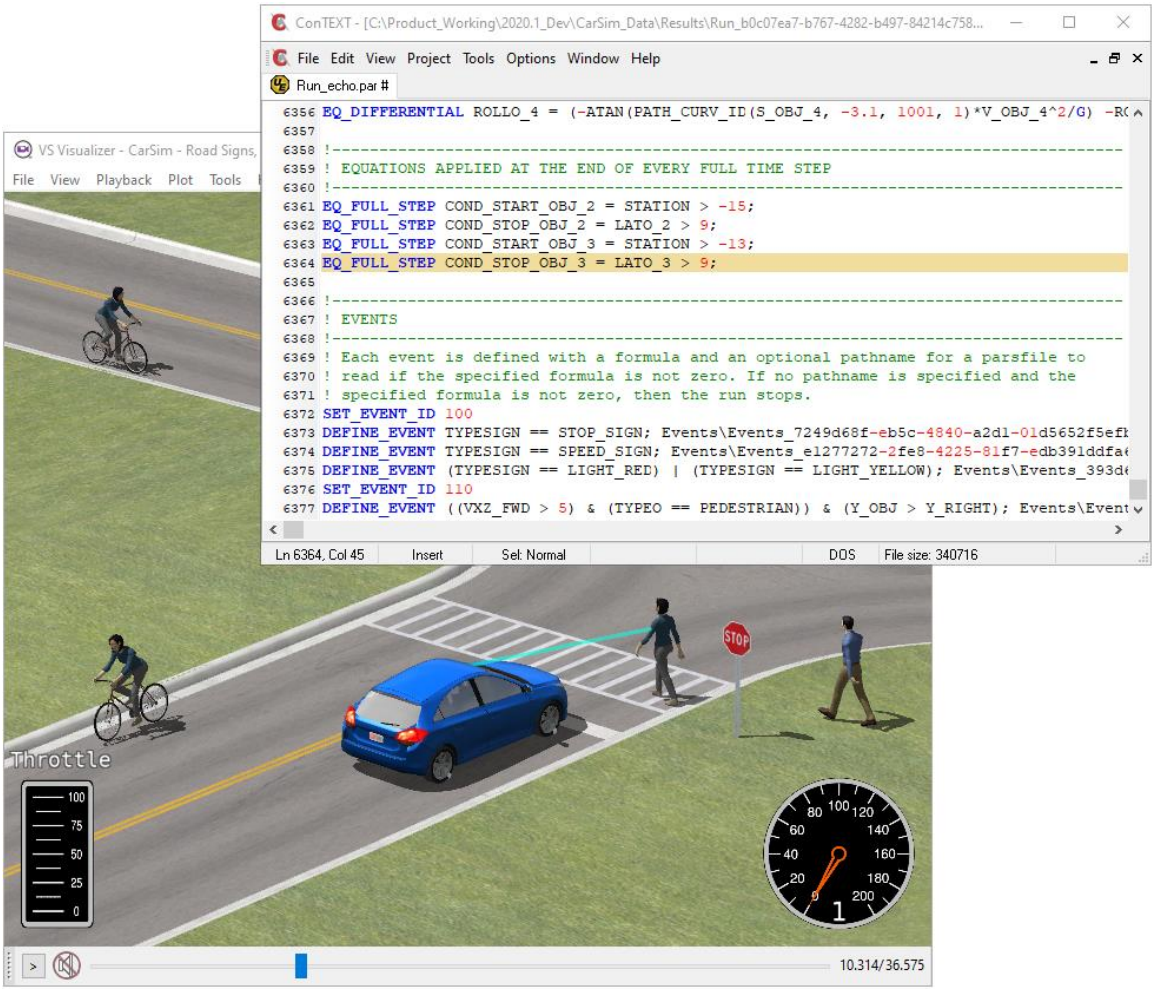

Figure 13. VS Commands are used to manage detection of signs, bicycles, pedestrians.

Help menu items: *Reference Manuals* > *VS Commands* and *Technical Memos* > *Example: Extending a Model with Embedded Python Utility*.

ADAS and Automated Vehicles

The development of Advanced Driver Assistance Systems (ADAS) and Automated Vehicles (AVs) relies heavily on simulation due to the number of variations in control strategies and environment variables.

Moving Objects (Targets)

CarSim supports up to 200 detectable objects whose location or speed is of interest; these include traffic vehicles, buildings, signs, pedestrians, bicyclists, etc. (Figure 13, Figure 14). These objects may be positioned anywhere and support multiple options for motion control: position, velocity control, acceleration control, direct attachment to a vehicle sprung mass.

Visibility and color controls are useful when simulating traffic signals. Other properties that factor into an object's detection include its shape, size, material type, and reflectiveness.

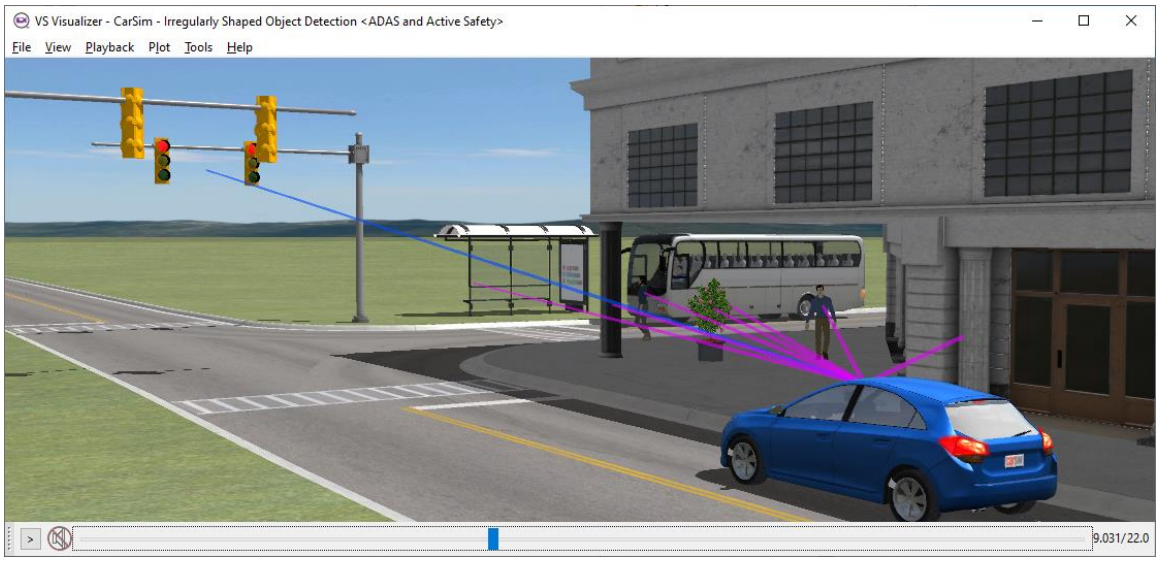

Figure 14. Scenario with a building, traffic lights, a bus, and pedestrians.

ADAS Sensors

CarSim supports up to 99 Advanced Driver Assistance Systems (ADAS) range and tracking sensors (camera, radar, ultrasound, etc.) that detect moving objects as targets. These are "true detections" based on 3D vectors that connect a sensor location to points on the moving objects. (An optional license is needed to use the built-in ADAS sensors.)

In addition to the location and motion properties and variables for target objects, the objects also have properties that affect how they appear to sensors, such as shape and reflectiveness.

A set of 24 output variables is calculated for each combination of sensor and target object. These include three bearing angles, distances to points on the object, relative velocities, local coordinates, and more. Two of the variables are based on user-defined properties — type and message — that are useful when the target represents a road sign such as a Speed Limit or Stop Sign.

The detection calculations take occlusion into account. Figure 15 shows blue lines connecting a sensor in the blue car to target objects. The motorcycle occludes the detection of the wall, and the wall and motorcycle both occlude the detection of the truck on the ramp. **Help** menu item: *ADAS Sensors and Target Objects*.

For those interested in image-based detection — color, surface normal, pixel depth, etc. — CarSim also supports a shared memory buffer-based sensor option. **Help** menu item: *Reference Manuals > VS Visualizer*.

Roads and Environment

The CarSim VS Math Model includes the vehicle, plus environmental conditions of interest. This is particularly valuable when using simulation in the development of ADAS and autonomous vehicle control systems. The ability to control and simulate scenarios involving wind, detailed 3D road surfaces, and other actors such as traffic vehicles, pedestrians, bicycles, buildings and infrastructure allows you to examine the control response to a variety of external stimuli, sensor faults, and other inputs.

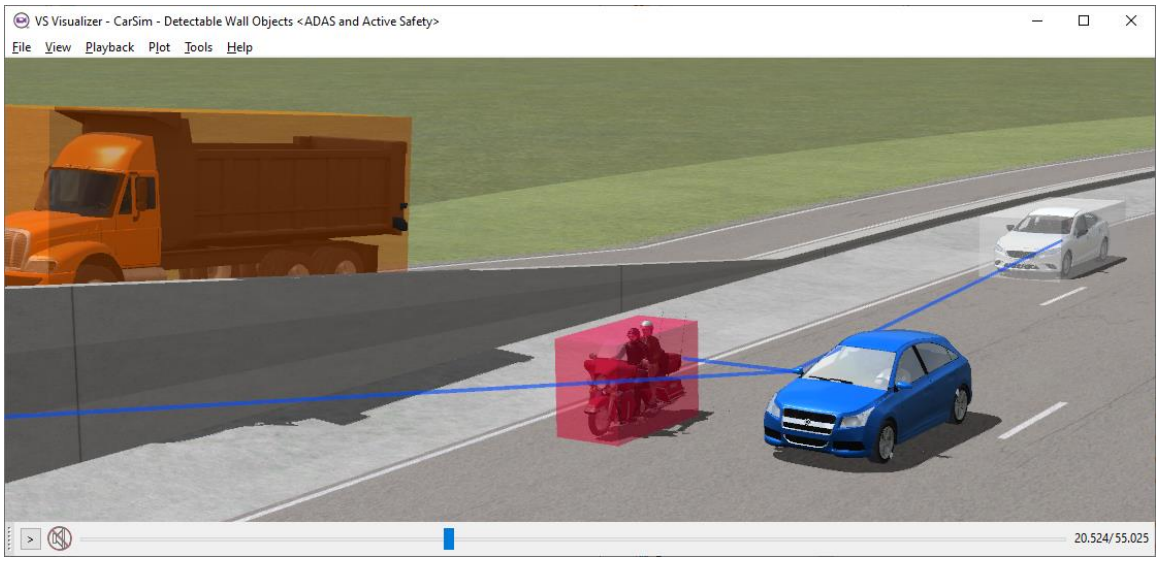

Figure 15. Objects can block (occlude) detection of other objects.

Road Surfaces

In order to simulate ADAS scenarios, it is sometimes essential to include realistic 3D models of intersections and complicated road designs (Figure 16). To do this without external software tools, CarSim supports ground surfaces made from up to 200 road surfaces that use S-L path-based coordinate systems (S is station, the longitudinal distance along a path, and L is a lateral distance perpendicular to the path). Connections between surfaces are made automatically for each point of contact for a vehicle tire or moving object.

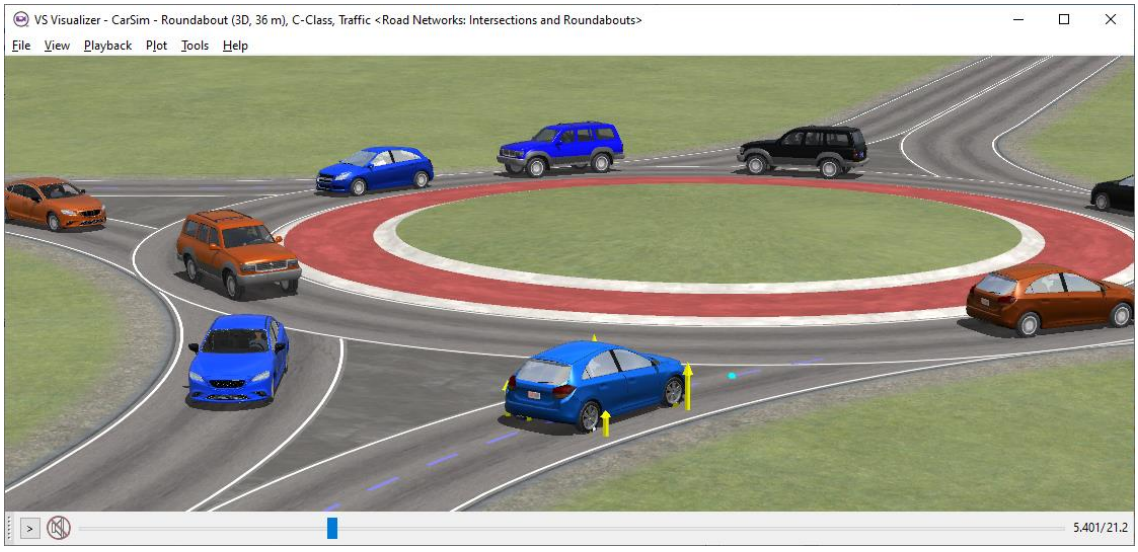

Figure 16. Ego vehicle and traffic on a roundabout made from five VS Road surfaces.

Help menu item: *Paths, Road Surfaces, and Scenes > Path and Road Surfaces*.

VS Terrain

Information about the 3D ground surface is sometimes available from 3D files made with third-party software. These include the FBX and OBJ formats supported by 3ds Max (3D modeling software), OpenDRIVE, and the Unreal Engine. These 3D assets use a mesh of points to represent a surface. The VS Terrain format provides a high-efficiency mesh representation well-suited for providing ground input to tire models. Along with geometry at the point of contact, VS Terrain provides friction and a rolling resistance coefficient needed by tire models. Figure 17 shows pavement and curb geometry in the Mcity test facility, provided by a VS Terrain file.

Figure 17. Complicated 3D ground geometry in Mcity is provided by VS Terrain.

3D files in FBX and OBJ format may be converted to VS Terrain files using the VS Terrain Utility, accessed from the **Tools** menu and described in the **Help** menu item *Paths, Road Surfaces, and Scenes > VS Terrain*.

VS Scene Builder

CarSim includes the VS Scene Builder, a tool for assembling a scenario from a library of tiles and 3D assets such as animated pedestrians, barriers, and signs (1) (Figure 18). The tiles are connected to form a scenario region (2), and a path or series of paths are created by clicking on the arrows presented on the tiles (3). These paths, along with the terrain and animation assets that make up the tiles, are imported into CarSim (4) to define the driving environment. Figure 19 shows the urban scene in VS Visualizer after it has been imported into CarSim.

The VS Scene Builder includes an option in the **File** menu to import data from OpenDRIVE. This will create a new tile, after which paths can be specified and subsequently imported into CarSim.

Help menu item: *Paths, Road Surfaces, and Scenes > VS Scene Builder*.

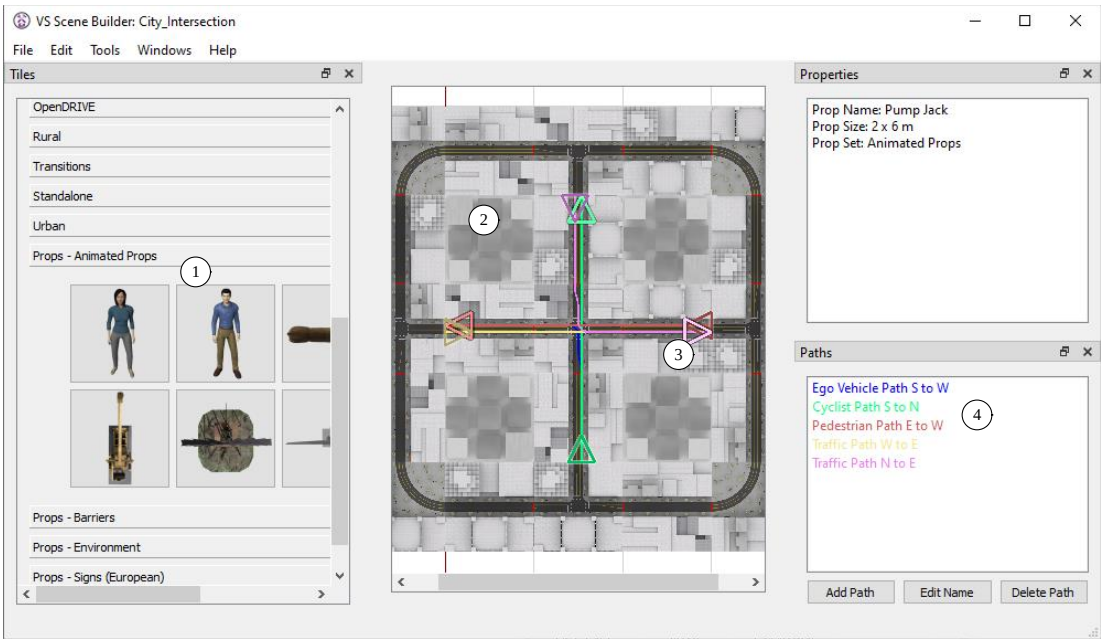

Figure 18. VS Scene Builder tool from CarSim.

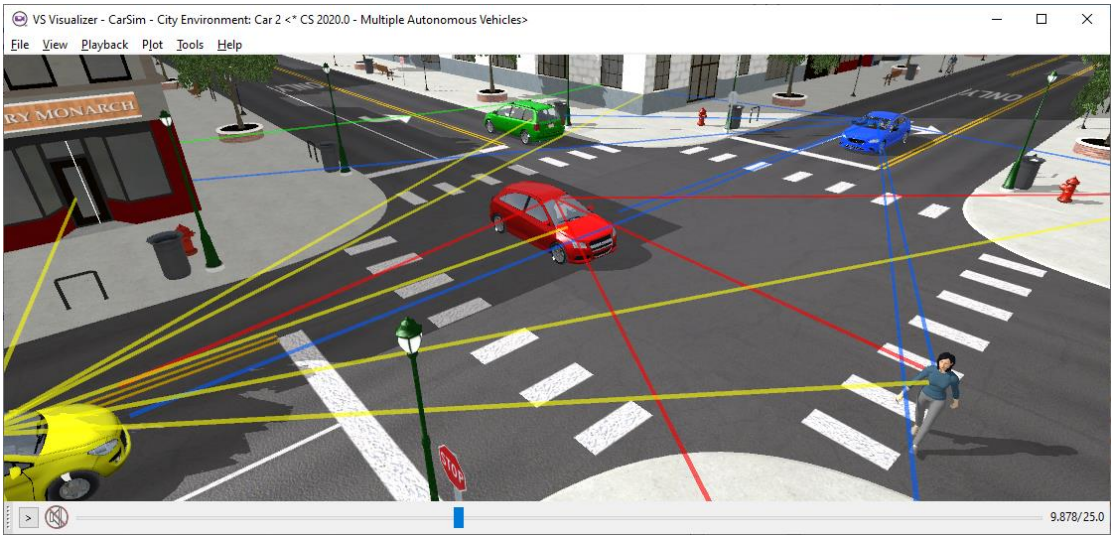

Figure 19. Urban scene with buildings and roads from VS Scene Builder.

Atlas Road and Path Import

There is often a need to simulate vehicle behavior on existing roads. One way to specify the geometry for road surfaces is by downloading GPS data from a mapping service, such as Google maps, using the CarSim Atlas web tool. The appeal of this approach is that mapping services cover most of the developed world and data acquired using Atlas is free. The amount of online data is increasing rapidly in support of navigation services for human drivers and autonomous driving software.

CarSim users have access to the Atlas tools at atlas.carsim.com. Atlas provides an interface to data from mapping services by Google and Here, shown in the center panel in Figure 20. The left-side

panel shows a highway off-ramp as viewed in Google, and the right-side panel shows the same highway off-ramp imported into CarSim after some clean-up effort.

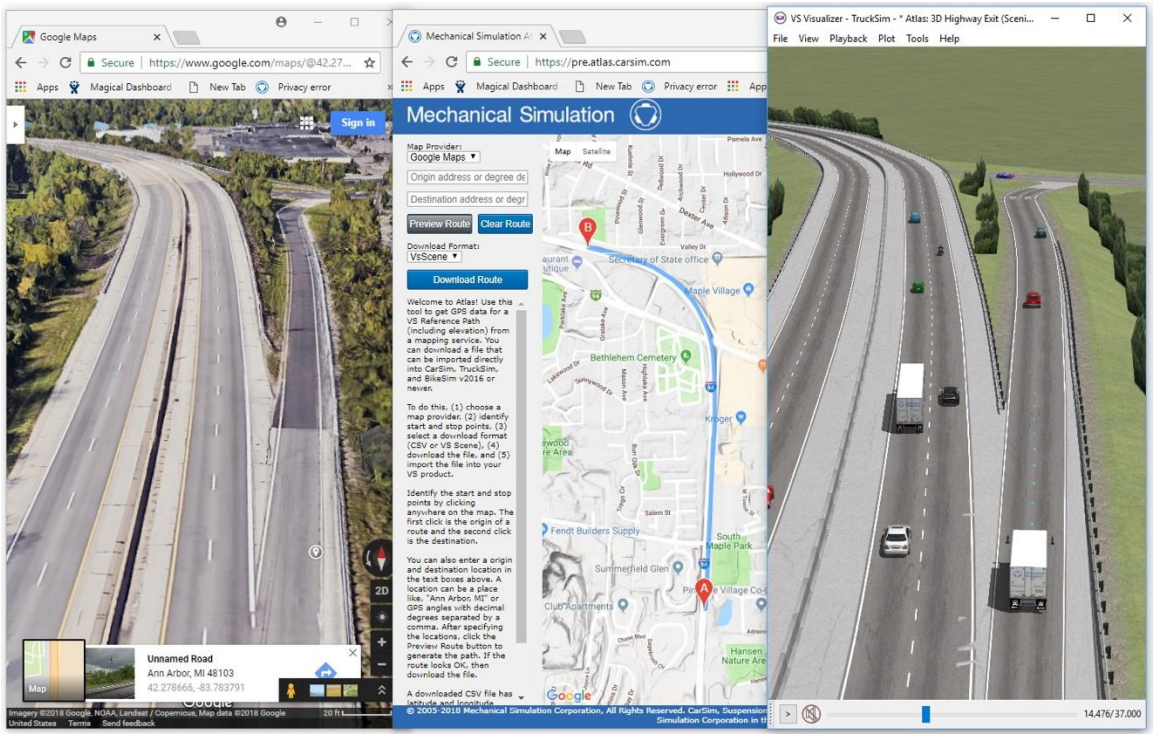

Figure 20. The Atlas web tool used to reproduce the 3D geometry of a highway exit.

Note: GPS data suitable for vehicle navigation is not guaranteed to be suitable to use "as-is" as a road geometry definition in CarSim; the expectation is that some clean-up effort will be required.

Help menu item: *Paths, Road Surfaces, and Scenes* > *Connecting 3D Roads from Atlas*.

Co-Simulation and Third-Party Software

All features described up to this point are part of any CarSim installation, with the exception that CarSim for non-RT Linux does not include a VS Browser.

CarSim users often run CarSim together with other (third-party) software. The architecture of CarSim is intended to provide support for easily building stacks with CarSim and other tools like MATLAB\Simulink, Unreal Engine, Python, and others. The System Requirements and Compatibility documentation details more about what types of other software can be used with CarSim. **Help** menu item: *Release Notes* > *System Requirements*.

The Windows version has two VS Solver library files, `carsim_32.dll` and `carsim_64.dll`, to be loaded in 32-bit and 64-bit applications, respectively. For Linux, the single file `libcarsim.so.2020.1` is a 64-bit library. The VS Solvers are provided in this form so they can be loaded into a variety of programs.

Import and Export Arrays

Running a CarSim math model from other software can be as simple as loading the VS Solver library and applying a Run command. However, usually the CarSim model will be included as part of a larger simulation and will need to communicate information to and from the other software. This is handled with arrays of Import and Export variables.

Lists of available Import and Export variables are configured based on the options selected for a given simulation. Many features, such as Motion Sensors, ADAS Sensors, or trailers are modular in nature and are not included if they are not used. In addition to the built-in options, new Import and Export variables may be added with VS Commands. A CarSim simulation is likely to contain several hundred available Import Variables and hundreds if not thousands of calculated output variables.

The Import and Export arrays are configured by activating existing variables. A typical setup for running under external software will have a subset of the available variables activated for import and export, matching the requirements of the external model. For example, if the external model has a braking intervention system, it should provide throttle and brake controls and receive sensor outputs from CarSim.

Help menu item: *Reference Manuals* > *VS Solvers.*

Simulink and LabVIEW

CarSim includes a set of Simulink S-Functions that connect the CarSim solver library to Simulink models. Figure 21 shows how the S-Function `vs_sf` appears in a Simulink model. The block is connected to the rest of the model with Import and Export variables that were activated in CarSim to match the requirements of the Simulink model.

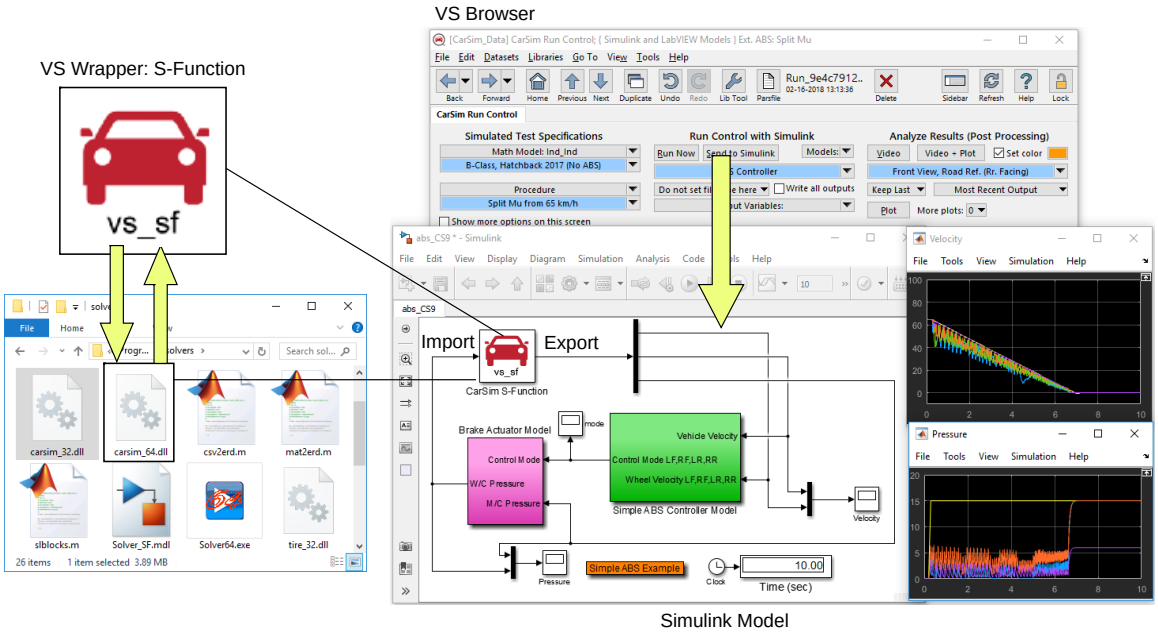

Figure 21. Running CarSim under Simulink.

When the Simulink model is run, the S-Function loads the CarSim VS Solver library, uses VS Modularity functions to construct a VS Math Model, and exchanges I/O data each time-step with Simulink. Plots in Simulink can show the results as the simulation proceeds. Once the simulation has finished, VS Visualizer may be used to further analyze data and view the animated video.

Help menu items in the *Guides and Tutorials* submenu: *CarSim Demo Tutorial* and *Running a VS Math Model in Simulink*. Similar capabilities are provided for LabVIEW.

FMI/FMU

Functional Mock-up Interface (FMI) is a standardized interface used in computer simulations to support both model exchange and co-simulation of dynamic models from different simulation tools. (Check the web site https://www.fmi-standard.org for definitive information about the FMI standard.)

One means for combining simulation tools with FMI is through co-simulation in which each tool is represented in the form of a Functional Mock-up Unit (FMU). CarSim supports FMI co-simulation by including the capability to automatically generate a slave VS FMU.

Help menu item: *Guides and Tutorials* > *Running a VS FMU in Simulink*.

Real-Time Platforms

While many CarSim simulations can run purely on Windows or Linux, a vast number of toolchains involving CarSim require physical hardware to be included, such as a hardware ECU, an entire brake system, a powertrain dynamometer, a camera for ADAS applications, or other types of physical hardware. Hardware can be included in a CarSim simulation using a Real Time platform, such as those provided by dSPACE, Opal-RT, ETAS, National Instruments, Concurrent, and A&D.

In all cases, a Windows Host machine runs the CarSim Browser and VS Visualizer. When making a simulation, the files are copied automatically to the RT Target machine. Depending on the system, the CarSim solver library is connected to either a Simulink RT S-Function or an RT FMU.

When configuring CarSim for use with RT hardware, operation is very similar to working with Simulink or FMI systems on Windows. However, it is necessary for users to also be familiar with the RT System software and tools to execute the simulation.

Help menu items: *Model Extensions and RT* > *External Models and RT Systems*. Documents specific to each RT platform are found in **Help** > *Real-Time and DS Systems*.

Unreal Engine

The Unreal Engine is a game engine developed by Epic Games. Originally created for first-person shooter games, it is now used in a variety of other genres including simulation environments for driving simulators.

The VehicleSim Dynamics plugin for Unreal Engine allows CarSim math models to run in an environment created with Unreal Engine (Figure 22). The plugin is available from the Unreal Marketplace and the User Section of the CarSim web site. (There is not a plugin for BikeSim.)

Figure 22. An Unreal Engine scene with vehicle physics provided by the CarSim plug-in.

VehicleSim Software Development Kit (VS SDK)

The CarSim solver libraries can be used from custom software created by advanced users and programmers working at companies that use CarSim. The SDK may be obtained from the carsim.com website, and includes all the tools, libraries, documentation, and example projects necessary to get working on a project with as little configuration as possible.

Running Multiple Vehicles

CarSim ADAS scenarios are typically simulated using moving objects as sensor targets, where these targets represent vehicles following simple motions such as a specified speed or acceleration along a path. CarSim also has two options to support scenarios with multiple full vehicle models.

Multiple vehicles in a single solver instance. Starting with version 2020.1, the VS Math Model can be constructed to include up to four independent vehicles. Each vehicle has its own set of controls (open-loop and/or closed-loop, powertrain, brake system, etc.). All vehicles share the same environment with the same ground surface, paths available for controllers, and other "actors" such as pedestrians. Each vehicle can be detected by other vehicles. Figure 19 (page 16) shows four complete vehicles all running in the same environment. See **Help** *Tech Memos > Simulations with Multiple Vehicles*. (BikeSim does not support this option.)

Co-Simulation with Parallel CarSim Solvers. Up to 20 VS Solver libraries can be used in parallel under MATLAB/Simulink, dSPACE SCALEXIO, Concurrent SimWB, NI Linux RT, or ETAS RT. These are set up using a dedicated GUI screen (**Tools > Parallel Solvers**) in which a CarSim VS Solver library is duplicated and given a unique name. These VS Solver libraries — along with their associated data — will all be loaded into a single simulation. The example simulation shown in Figure 19 (page 16) can also be run under the control of Simulink. See **Help** *>Tools > Running with Parallel Vehicles*.