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

Animator: Wheel Arrows and Other Indicators

The **Animator: Wheel Arrows and Other Indicators** screen is used to define a set of visual indicators for tire forces or other variables of interest that are repeated for each wheel, tire, or axle in the simulated vehicle.

All VehicleSim (VS) products include VS Visualizer, a program for viewing animations from a virtual camcorder that operates in a simulated 3D world. The typical virtual 3D world includes fixed objects such as the road, trees, and sky, and moving objects such as the body and wheels of a simulated vehicle. In addition to these obvious objects, the animator can be used to visualize other kinds of information by adjusting size or visibility properties of 3D objects based on variables of interest.

For example, Figure 1 shows an animation where yellow arrows are sized to indicate magnitudes of tire forces in the lateral and vertical directions. The figure also shows skid marks based on large tire slip generated as the vehicle engages in an extreme maneuver.

This screen is one of a few that provide data largely for VS Visualizer, whose only connection with the VS Math Model is through access to output variables written to file. This screen provides information to VS Visualizer for using output variables so show indicators such as force arrows. It also provides information to the VS Math Model to activate the output variables for being written to output files and to stream them for live animation. It may also include VS Commands instructing the VS Math Model to define new variables and assign values to them with custom equations.

Basic Controls for Multiple Wheel Indicators

The **Animator: Wheel Arrows and Other Indicators** screen has information that may be repeated for each wheel, tire, or axle in the simulated vehicle. Each dataset made with this screen involves a 3D shape (or group of shapes) that is repeated as needed and modified in terms of size or translucency based on dynamic variables. For example, Figure 2 (page 3) shows a dataset that causes the animator to show the arrows with lengths proportional to vertical tire force in Figure 1.

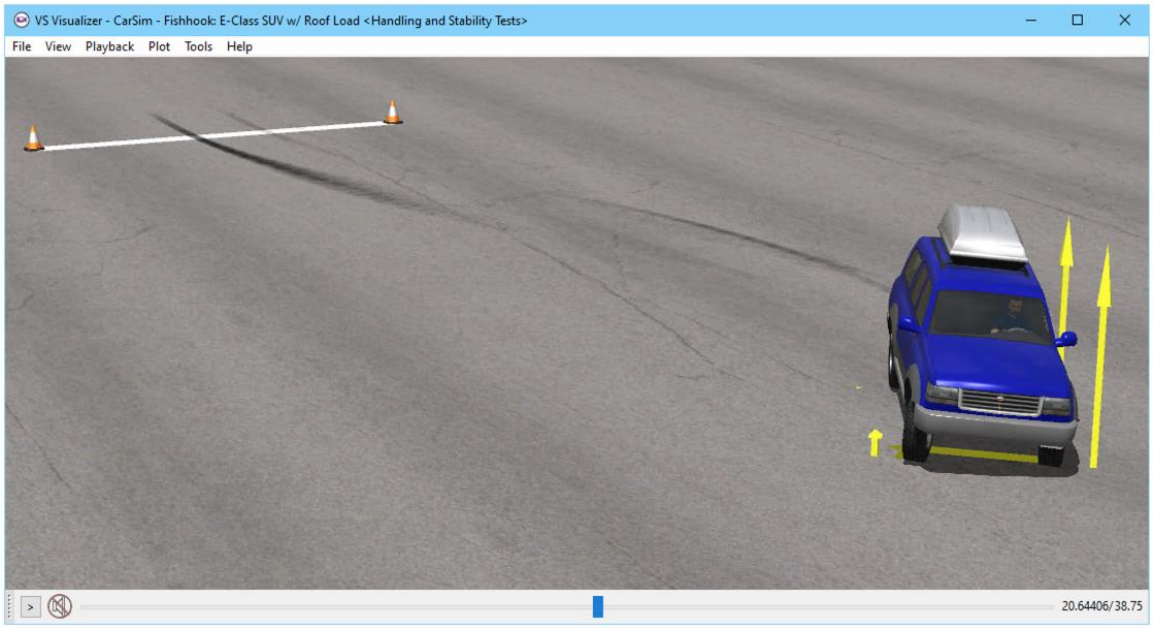

Figure 1. Animation showing tire force arrows and skid marks.

Placeholder Character '?'

This screen simplifies the description of multiple animator objects by repeating similar descriptions for each wheel or axle, changing only the names of variables that indicate the position in the vehicle model. For each wheel, we want an arrow to be shown that points in the direction of the vertical tire force, with the tail of each arrow located 0.5m to the side of the center of tire contact (CTC). The same shape will be shown four times, with a dimension in the Z direction adjusted in proportion to the force.

When the Browser repeats information for multiple axles or wheels on this screen, it treats the character `?` as a placeholder in any variable names, and will replace it with an ID for the axle or wheel in the written dataset Parsfile. For axles, the ID is simply the number of the axle (1, 2, …). For wheels and tires, it is two characters indicating the axle number and side (`L1`, `L2`, `R1`, `R2` for a four-wheeled vehicle). In the example, the name "`Fz_?`" is used four times to generate the desired names `Fz_L1`, `Fz_L2`, `Fz_R1`, and `Fz_R2`.

Basic User Controls

Figure 2 shows the screen with only the basic controls displayed, as used to indicate tire forces with arrows as shown in Figure 1. Advanced controls on the screen are shown and described in the next section.

The controls at the top of the screen (① - ⑧) define the conditions for repeating (number of axles, which side, point used to locate the 3D object, etc.).

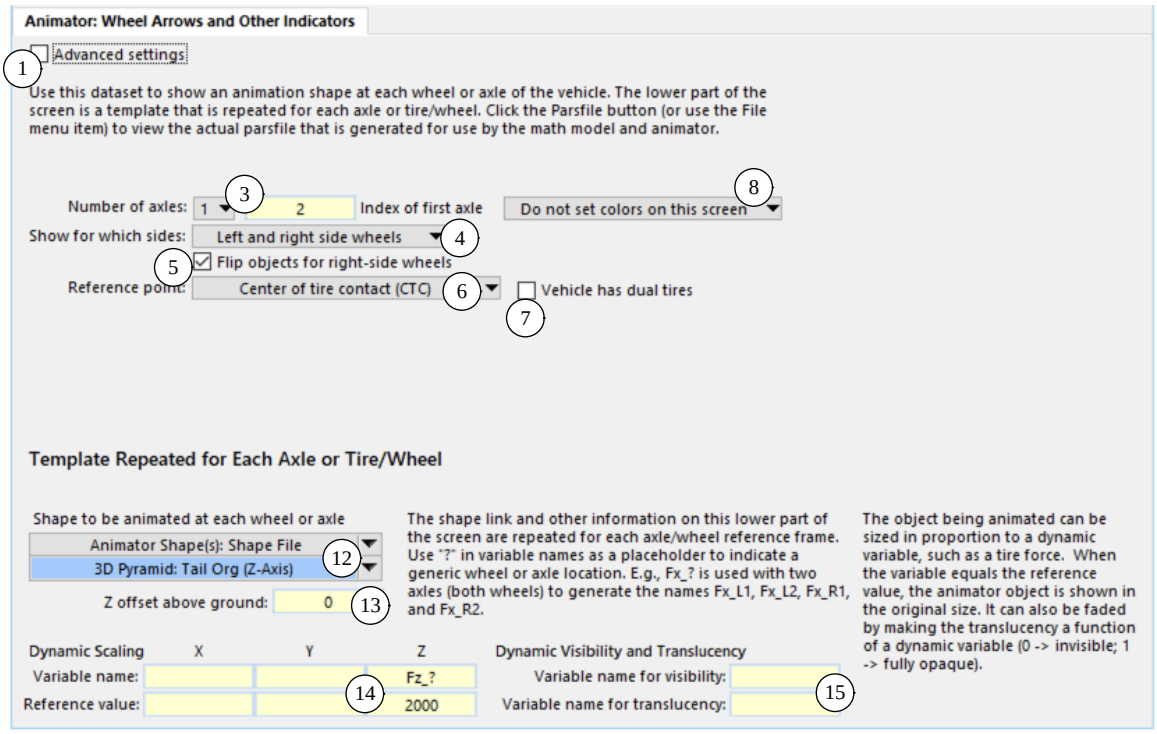

Figure 2. The Animator: Wheel Arrows and Other Indicators screen used to show tire forces.

The controls at the bottom of the screen provide a template for the information that will be repeated. The VS Browser automatically defines a reference frame for the animator for each copy of the indicator and sets up the reference frame based on settings provided on this screen. The reference frame locates the object and orients it based on the axle, wheel center, or center of tire contact, depending on the selected options.

(1) **Advanced settings** checkbox. Check this box to view two extra data fields for VS commands or equations for more advanced displays that are described in the next section (page 7).

(3) Drop-down list to specify the number of axles that will have animated indicators. An adjacent yellow field is used to specify the index for the first axle. If including the lead unit, this should be 1, as shown. For a trailer it should be the number of the first axle on the trailer. If blank, the default 1 is used.

In CarSim, the maximum number of axles is 5; TruckSim has a maximum of 20; and BikeSim has a maximum of 3.

(4) Drop-down list of options for determining whether the indicators are located at the center of the axle, at the left wheel, the right wheel, or both wheels (Figure 3).

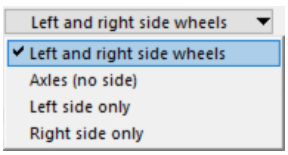

Figure 3. Drop-down control to select which side(s) of the axle have indicators.

This control is not present in BikeSim; there is only one wheel per axle.

(5) **Flip objects for right-side wheels** checkbox. If selected, the reference frames for wheels on the right-hand side of the axle will be flipped 180° in yaw and with the roll and pitch angles reversed. This is done for asymmetric indicators that have an inside/outside design. For example, the vertical force arrows in Figure 1 are located 0.5m outside of the wheels; the arrows on the left side are located 0.5m to the left of the tire contact center (the direction of positive Y) and the arrows on the right side are 0.5m to the right (negative Y direction). If this box were not checked, all four arrows would be located to the left of the wheels, with the arrows on the right-hand side of the vehicle hidden inside the vehicle body shape.

This checkbox is visible only if indicators are used on the right-hand side wheels (4). If indicators are shown only for the left-hand wheels, or only for the axles, then this checkbox is not displayed in CarSim and TruckSim.

In BikeSim, it is available to flip objects for the rear wheel, if the indicator is offset in the X direction.

(6) Drop-down list of options for determining the location of a reference point for the indicator object (Figure 4). This list is only visible in CarSim and TruckSim if the indicators are located at wheels; if indicators are shown only for the axles, then this control is not displayed. It is always visible in BikeSim.

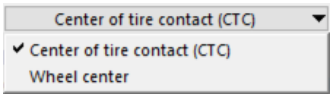

Figure 4. Location of reference point for indicator object.

If the choice is **Center of tire contact (CTC)**, then the indicators are shown where the tire contacts the ground (Figure 1). If the choice is the wheel center, then the indicators are shown at the center of the wheel. In BikeSim, another choice is the crown ground contact point.

(7) Checkbox for vehicles with dual tires in CarSim. If the vehicle has dual tires, then this checkbox must be used for the wheel indicator animation to work properly. If selected, then the choices for the reference point location (6) are different. You can now select which tire to be the location of the reference point (Figure 5).

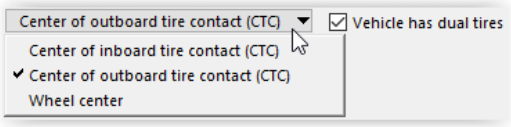

Figure 5. Revised list of possible locations for reference point if there are dual tires.

If the vehicle with duals has an axle without dual tires then the inboard/outboard selection will simply select the CTC of the single tire contact.

This checkbox does not exist in TruckSim, which always has dual tires available. Nor does it exist in BikeSim, which never has dual tires.

(8) Drop-down list of options for setting colors.

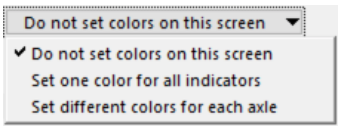

Figure 6. Options for setting colors for indicators on each axle.

If the selected option is to not set colors here, then the color of the indicator is set normally, using the properties of the linked object dataset (12) or possibly the current run color that might be set from the **Run Control** screen. More information about the additional settings are provided in the next section (page 9).

(12) Link to the animator shape. This is typically a simple shape such as an arrow or square that will be resized to indicate the state of a variable of interest. (The arrow shape used in this example is described in the following subsection.) A link can also be made to an assembly or group of shapes.

The object that is linked here should be set up to be located at the point specified with the controls in the top part of the screen ((4), (5), and (6)). For example, the vertical arrows shown in Figure 1 are defined such that the center of the tail of the arrow goes 0.5m to the side of the specified reference point.

(13) Z offset above CTC (center of tire contact). This optional value is used to raise the indicator object by a specified amount if the reference point is a CTC. If the reference point (defined with options (4) and (6)) is not the CTC, then this data field is not displayed.

(14) Dynamic scaling information. The linked object (12) can be stretched in the X, Y, and/or Z directions using the animator keywords `SET_FRAME_SCALE_X_NAME`, `SET_FRAME_SCALE_Y_NAME`, and `SET_FRAME_SCALE_Z_NAME`. The scaling is defined as [*current value of variable*]/[*reference value*]. For example, in Figure 2, the variable name for the Z direction is `Fz_?` and the reference value is `2000`. That means that if the vertical force is 2000N, the arrow is shown at the original (unscaled) height.

Any occurrences of the `?` character in a variable name will be replaced with a suffix for the object. As described earlier, the specified name `Fz_?` will be used to generate inputs for the animator for four wheels, replacing `?` with wheel IDs to obtain the names `Fz_L1`, `Fz_L2`, `Fz_R1`, and `Fz_R2`.

If the fields for a direction are blank, then no scaling is performed in that direction. For the example dataset, the arrow shape is shown with the original dimensions in the X and Y directions.

(15) Dynamic visibility information. The linked object (12) can also be dynamically modified in terms of visibility and translucency (transparency). Visibility is either on or off when the dynamic visibility is linked to a variable. If the linked variable is zero, the object is invisible; if the linked variable is not zero, then the object is visible. The animator keyword for this setting is `SET_FRAME_VISIBLE_NAME`.

Another option is to link a variable to the object to control translucency. A value of 0 for the linked variable means the object is invisible, a value of 1.0 or higher means the object is fully visible and opaque, and a value between 0 and 1 means the object is translucent. This option

works if the object (12) is defined with an OBJ or OSG file, and is set to display the front surface only. The animator keyword for this option is `REFERENCE_FRAME_ALPHA_NAME`.

The translucency feature is used in the example tire skid marks (Figure 1) and will described in more detail later (page 12).

Force Arrow Example

The original purpose of this library was to help set up the visual display of force vectors as shown in Figure 1. (This is why the folder containing the library files is named `Arrows`.) The setup for the vertical arrows was shown in Figure 2. In this section, we will look at the linked **Animator: Shape File Link** dataset (Figure 7).

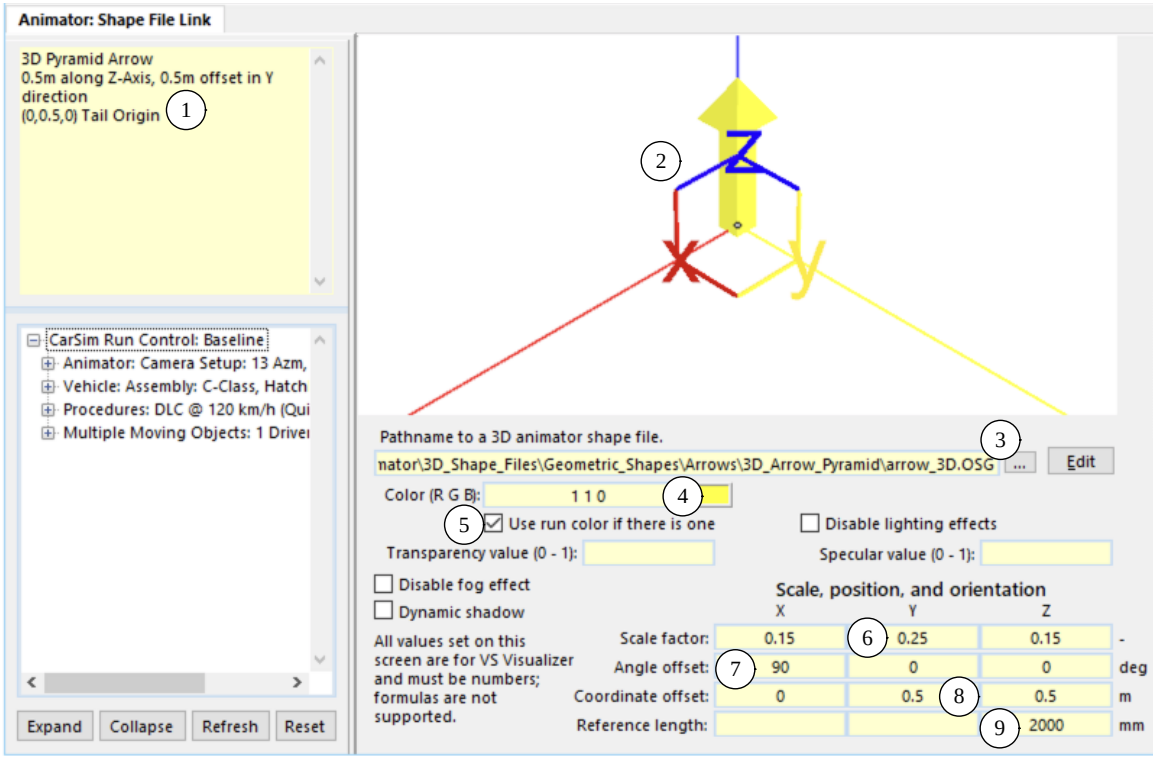

Figure 7. Shape File Link dataset for the vertical arrow.

The Notes field (1) indicates that the 3D object is 0.5m long along the Z axis, and that it is located by the tail, with an offset of 0.5m in the Y direction. The image shows the approximate appearance and orientation (2).

A single 3D shape file (3) is used for all three pyramid force arrows (X, Y, and Z directions). The arrow is resized to be smaller in all dimensions, with more reduction in the X and Y directions (6); the original length is 2m, and the 0.25 scaling reduces it to 0.5m. The original has the point at the origin, and is oriented to be parallel with the Y axis.

For this dataset, an angle offset of 90° about the X axis (7) points it in the direction of the Z axis. (Different rotations are specified to point the arrow in the X and Y directions.) There are two offsets applied (8): the shape is moved up by 0.5m to put the tail at the origin rather than the point, and it is also moved 0.5m in the Y direction, so the arrows will be 0.5m outside the tire CTC in the

animations, as shown in Figure 1. The reference length is set here to 2000 (9). (This is the same value specified on the **Animator: Wheel Arrows and Other Indicators** screen in Figure 1, so the value set here is not significant because it would be overridden by the value on the other screen.)

Critical settings here are:

1. The arrow is pointing parallel to the Z axis.
2. The tail is at the origin, so Z scaling will move the point up and down, with the back of the tail remaining at the origin of the reference frame. Dynamic scaling was set in Figure 2 to use the variable `FZ_?`, where ? is replaced for four time for suffixes `L1`, `L2`, `R1`, and `R2`.
3. The origin is located 0.5m in the Y direction of the reference frame.
4. The box **Use run color if there is one** is checked (5), so the force arrows will match the run color if one is specified.

Advanced Controls

The controls shown earlier are suitable for setting up visual indicators for variables that already exist in the VS math models, such as tire forces, slip angles, other forces, etc. The basic controls cover the main animator reference frames that have been used for most past examples.

Depending on the settings made in the top portion of the screen, the reference frames that are defined automatically are oriented using the axle X-Y-Z directions, the wheel-spindle X-Y-Z directions, or the tire/ground contact X-Y-Z directions.

Advanced users can define new output variables with VS commands, or override the definitions of reference frame to change the coordinates or orientation variables. As an example, Figure 8 shows the paths taken by the tires on the four axles of a combination vehicle in TruckSim. The variables used to show the paths are not built into the VS Math Model, but are easy to create using VS Commands.

Figure 9 shows an **Animator: Wheel Arrows and Other Indicators** dataset used to set up this animation. The tire paths made by utilizing the capability of VS Visualizer to show "ghosts" of some objects from previous time steps.

To see two large miscellaneous fields ((2) and (11)), check the box (1) **Advanced settings**.

(2) This field is provided for advanced users to set animator parameters and to define new variables with VS commands.

The contents of this field are written before any reference frames are defined, so it can be used to define variables that might be used in equations that are supplied for individual wheels or axles in another field in the lower portion of the screen (11). For example, the dataset shown in Figure 14 defines a new parameter that is used in the other miscellaneous field (11). The contents of this field are described in a following subsection.

This field is not visible unless the **Advanced settings** checkbox is checked (1).

(11) This field is provided for advanced users to provide commands or settings that are repeated for every axle or wheel. The contents of this field are described in a following subsection.

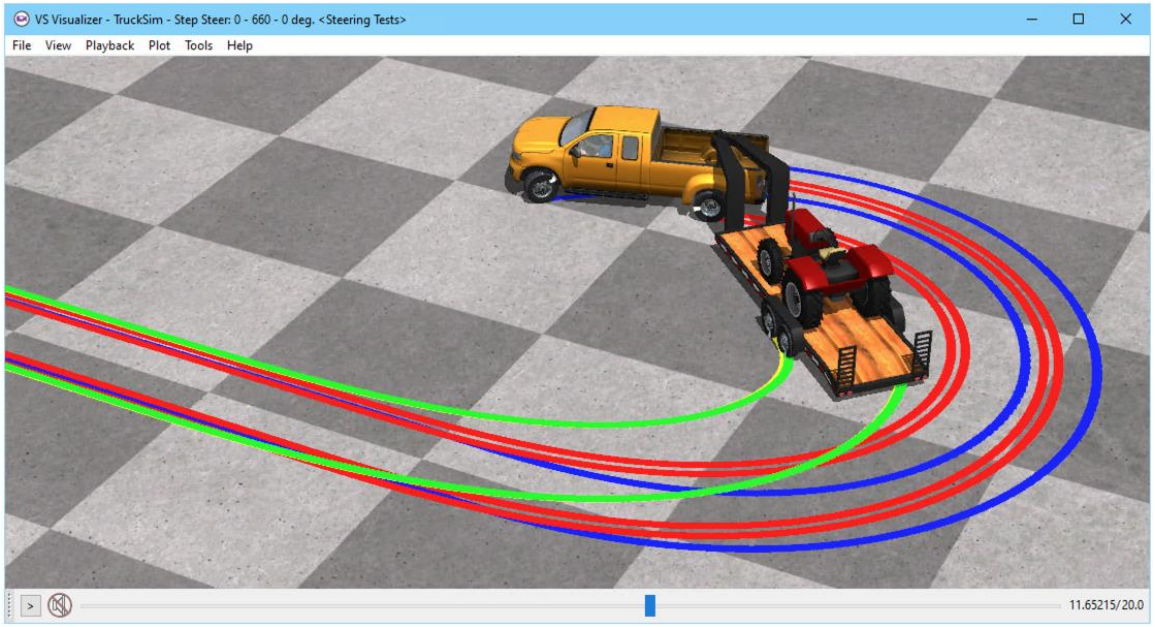

Figure 8. Animation showing tire paths to visualize tracking.

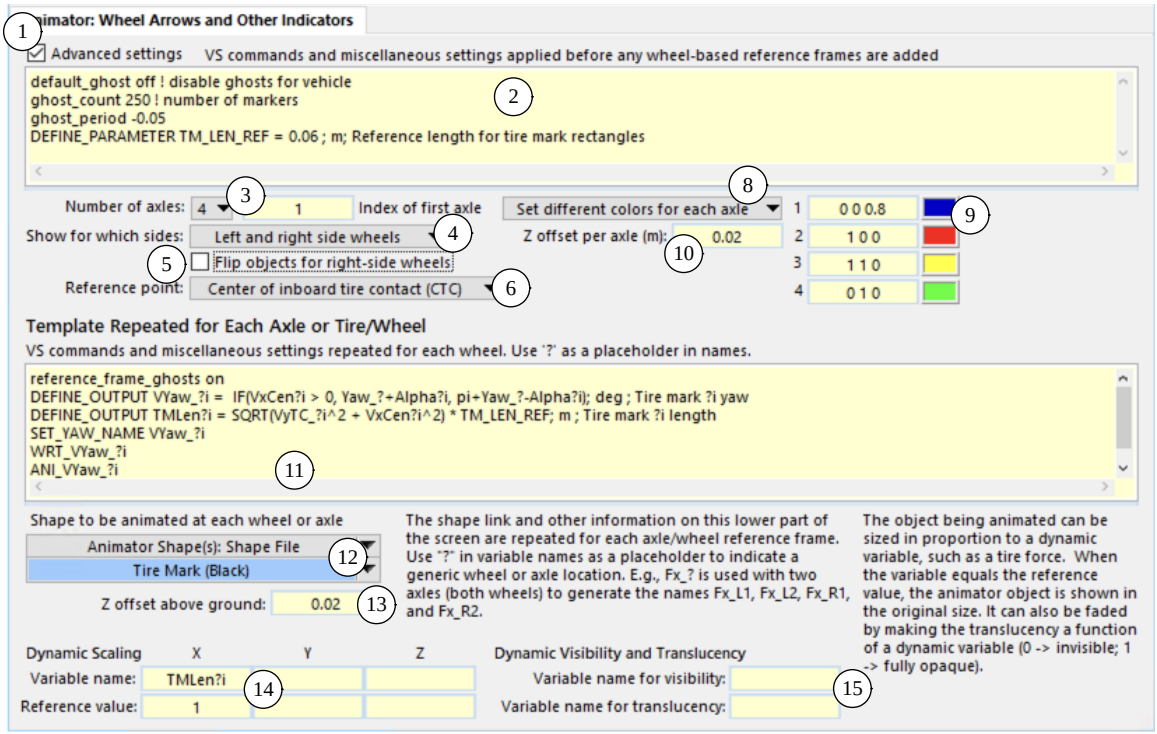

Figure 9. Dataset used to show tire paths.

This field is not visible unless the **Advanced settings** checkbox is checked (1).

Setting Colors

(8) Drop-down list of options for setting colors. If the selected option is to not set colors here, then the color of the indicator is not set from this screen.

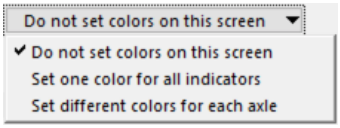

Figure 10. Options for setting colors for indicators on each axle.

On the other hand, if one of the options is selected for setting color here, then additional data fields are shown ((9) and (10)).

(9) Color of object (keyword = `SET_RUN_COLOR`). If the linked shape dataset (12) was set up with a checkbox **Use run color if there is one** ((5) Figure 7), then the color set here will be used. However, if the box is not checked, then the original color for the shape is used and any setting on this screen will be ignored.

The color can be specified with RGB numbers separated by space or with the adjacent color control that displays the current color.

If the drop-down list (8) option is to not set colors (e.g., Figure 2, page 3), then these fields are not visible.

(10) Z offset per axle. If multiple shapes with different colors are located in exactly the same plane, there can be flicker problems in the animation. To avoid the flickering, an offset can be specified here to raise the reference frames for each axle reference frame, putting the different paths in different planes. For example, the animation in Figure 8 shows the green paths (axle 4) on top, and the blue paths (axle 1) on the bottom.

This field is visible only if the option is chosen to set different colors for each axle.

Commands in the Miscellaneous Fields

Figure 11 shows part of the Parsfile for the dataset shown in Figure 9. It contains the information written in the top miscellaneous field (2) exactly as written (lines 22 – 25). As mentioned earlier, this example makes use of the capability of VS Visualizer to show objects from previous time steps as "ghosts." The parameter `default_ghost` is set to `off`, to so only specified reference frames will show ghost images. The parameter `ghost_count` is set to `250`, the number of ghost images that will be shown for each reference frame that has ghosts enabled. The third parameter, `ghost_period`, is the time interval between ghosts, and is set to `-0.05s`.

In addition to setting the three parameters in VS Visualizer, the first miscellaneous field has a VS Command to add a parameter `TM_LEN_REF` for a reference length of tire marks.

The contents of the bottom miscellaneous field (11) are then written with a # prefix, as they appear in the field, using ? as a placeholder for a suffix (lines 29 - 34).

Figure 12 shows another part of the same Parsfile.

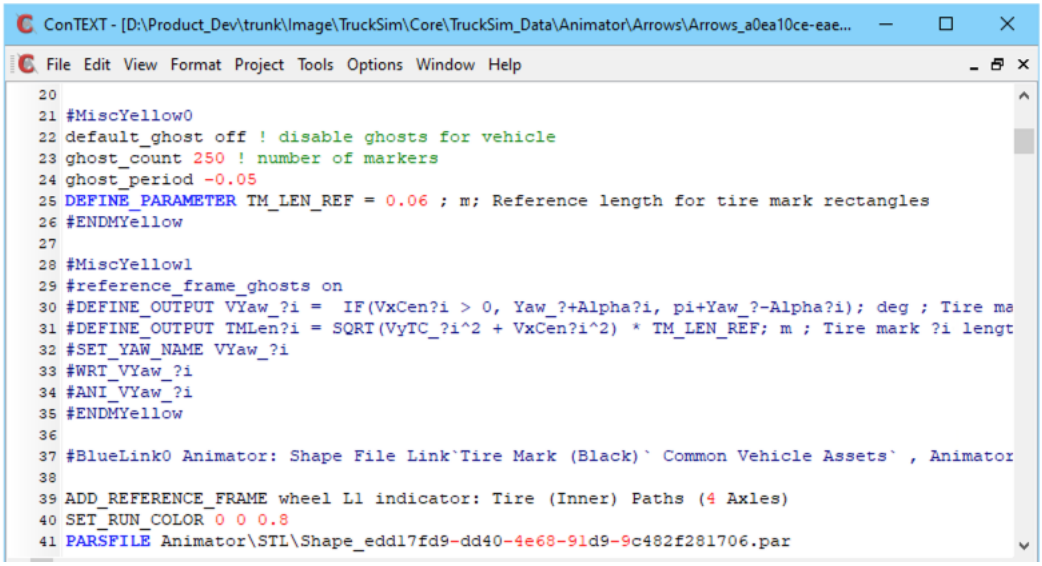

Figure 11. Top part of Parsfile generated for dataset of Figure 9.

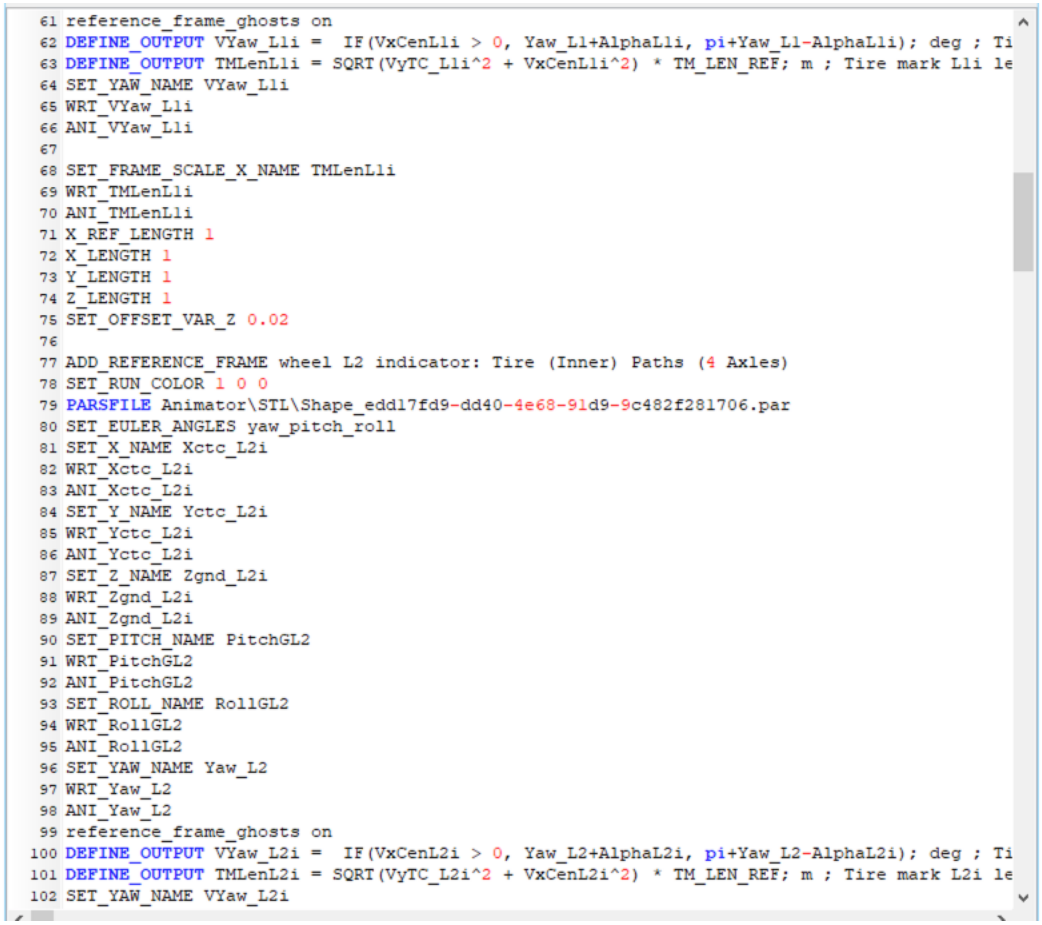

Figure 12. Part of Parsfile with section repeated for each axle.

Lines 61 - 66 show the contents of the second miscellaneous field again, but without the # prefix on each line, and replacing the `?` character with the text `L1`, e.g., `VYaw_L1i` in line 64.

Notice how the VS command `DEFINE_OUTPUT` is used to define two new output variables for the L1 wheel. One is a new yaw angle to orient the reference frame (e.g., `VYaw_L1i`, defined in line 62 and used in lines 65 - 67). The other is a dynamic length for the tire mark (e.g., `TmLen_L1i`, defined in line 63 and used in lines 69-71, based on the yellow field (14) in Figure 9).

All lines in the figure going from 62 to 70 reference variables that end with the suffix `L1i`. Starting with line 81, the same content is repeated, but with variables ending with the suffix `L2i` (tire) or `L2` (wheel). This dataset specified that the indicators are used for two axles, on both sides, showing skid marks for four tires.

Going back to the screen setup (Figure 9, page 8), notice that the scaling of the rectangular tire mark (14) is proportional to the new output variable `TMLen?i`; the rectangular ghost images are longer if the wheel is moving fast.

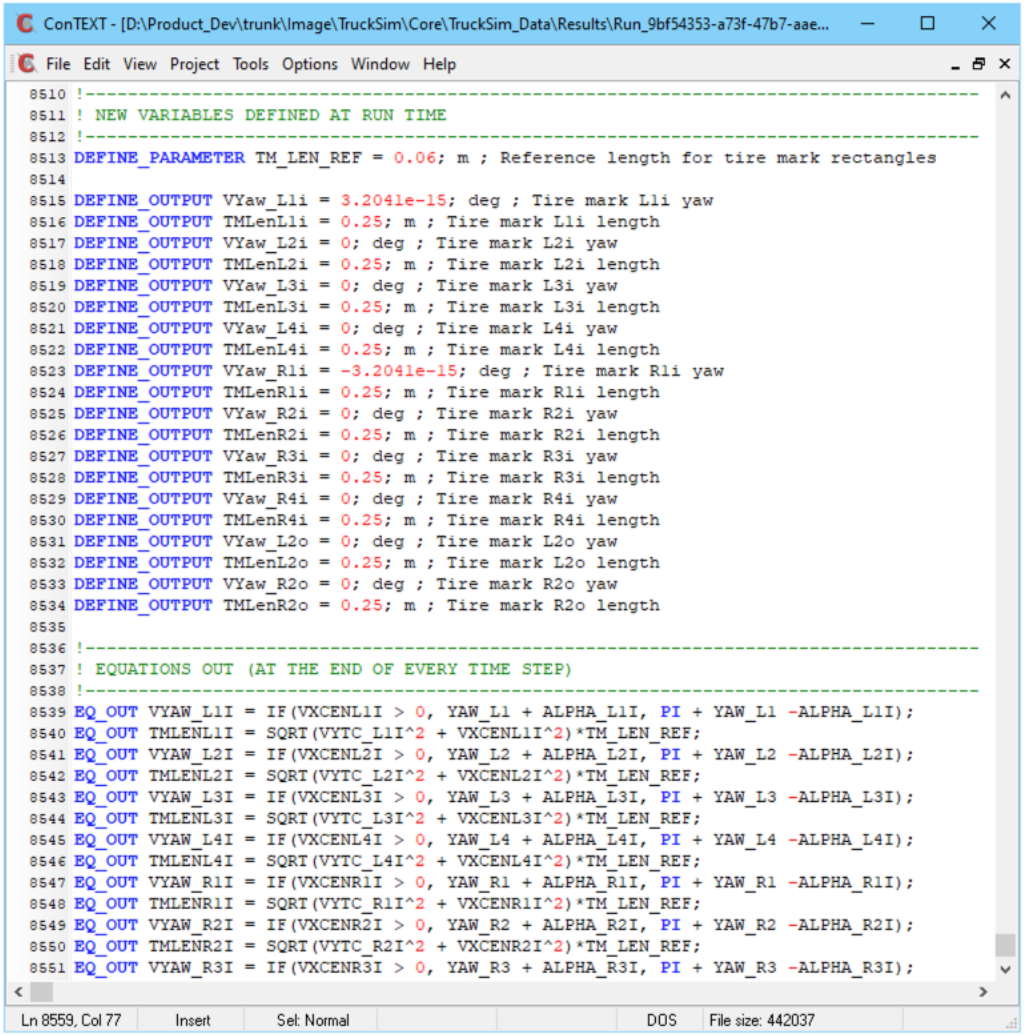

Figure 13. Part of echo files generated for TruckSim example with tire paths.

Echo File

Figure 13 shows part of the Echo file generated by TruckSim for the tire path example.

It has the single `DEFINE_PARAMETER` command for a reference tire mark length, then 20 `DEFINE_OUTPUT` commands for new output variables for the 20 tires in the model (one axle with duals, the other three axles with single tires). The four outboard tires on axle 2 were added with a different dataset that defined the outputs in lines 8531 – 8534.

Tire Skid Marks

The first example simulation (Figure 1, page 2) shows force arrows and skid marks. Figure 14 shows the dataset used to display tire skid marks. This CarSim setup is like that used for the TruckSim colored paths, but this version makes use of the dynamic translucency available with VS Visualizer.

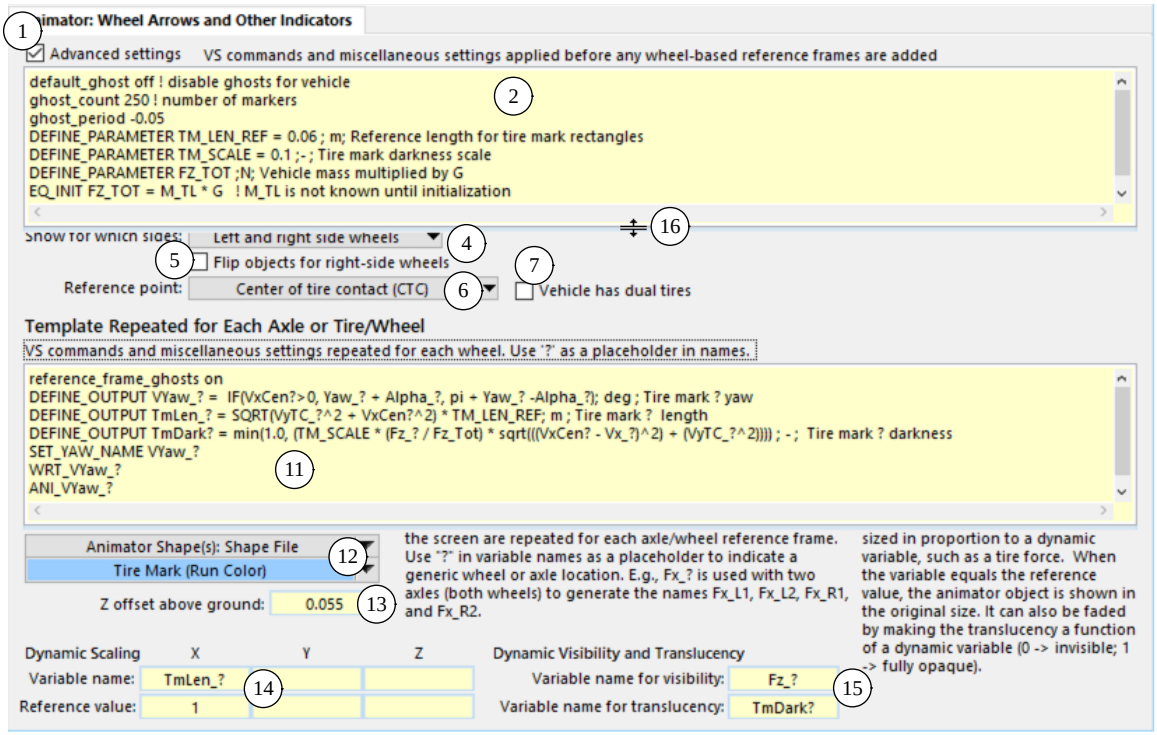

Figure 14. The Animator: Wheel Arrows and Other Indicators screen with advanced settings.

Some differences between this setup and the one used for colored paths are:

1. The top miscellaneous field defines three new parameters instead of just one. One of the new parameters is `TM_SCALE`, a darkness scale used to set darkness of the path based on slipping.
2. The other new parameter, `FZ_TOT`, is based the calculated total mass of the vehicle, which is not known until the initialization is complete. Therefore, the value is assigned with an `EQ_INIT` command, rather than a simple assignment.

3. Both fields have more commands. The splitter control (16) (shown at the bottom of the top field) was used to increase the heights of both fields to show their complete contents.
4. The color control (hidden under the top field) was set to black for all wheels.
5. The bottom miscellaneous field defines an output `TmDark?` for darkness of the tire mark. An equation is if sets the darkness using the new parameter `TM_SCALE`, the vertical load, and slipping of the tire.
6. The visibility of the tire mark depends on vertical load, `Fz_?` (15). If `Fz_?` is zero, the mark is invisible.
7. The translucence of the tire mark is proportional to the new output `TmDark?`.