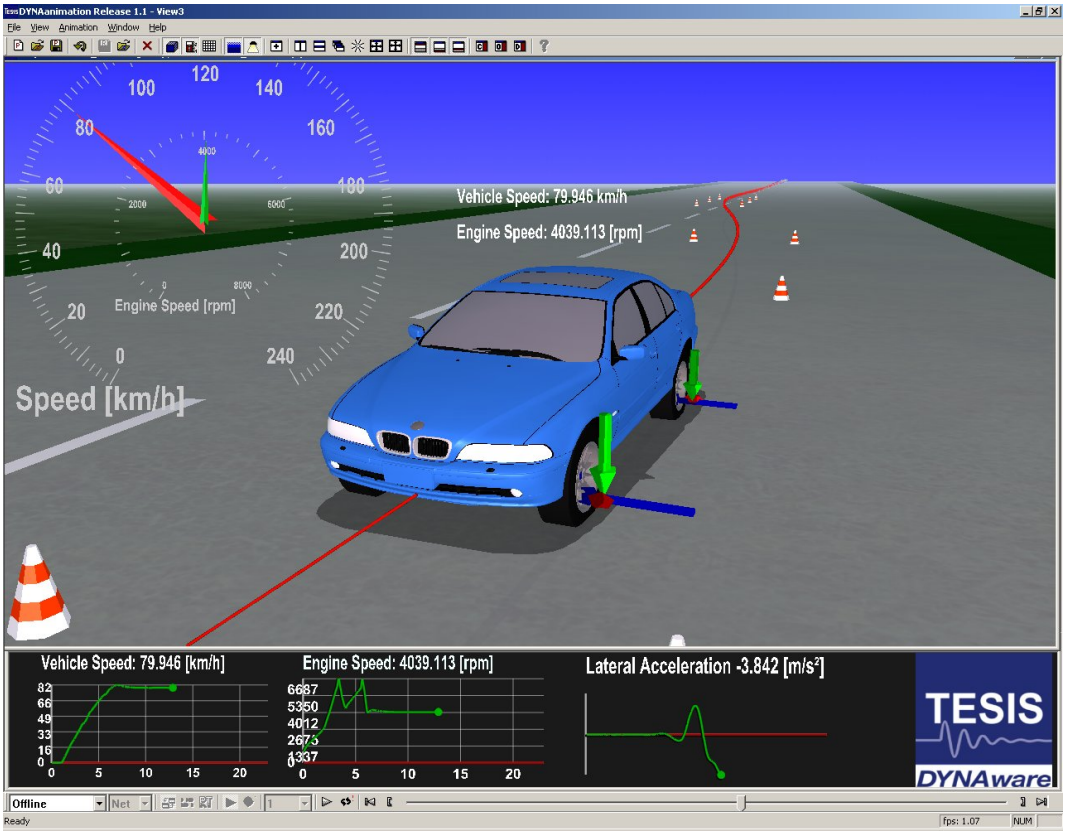
TESIS DYNAware

DYNAanimation 1.2

User Manual



User Manual

Modification: June 27, 2006

For Software Version: DYNAanimation 1.2

TESIS DYNAware
Technische Simulation Dynamischer Systeme GmbH

Baierbrunner Straße 15

D-81379 Munich

Germany

Telephone: +49 89 74 73 77-0

Telefax: +49 89 74 73 77-99

E-mail: info@tesis.de

Web: www.tesis.de/dynaware

Contents

1 Introduction

The animated display of simulation results for dynamically moving objects is a valuable means to get an immediate understanding of the system's dynamic behaviour

DYNAanimation enables the straightforward animation of result data from your Simulink model synchronous with the running simulation. In principle, any moving mechanical system can be animated. Furthermore, selected results can be highlighted by means of various graphical displays, e.g. bar graphs, gauges, plots, and many others.

Whether you just wish to check what is happening during your simulation or you are in need to create eye-catching virtual environments for presentations - our tools provide many options for the individual design of the display as well as standard settings for a quick start.

The installation comes with three examples of a simulation model including a DYNAanimation block:

`demo_vehicle` a single track vehicle model

`demo_v4_engine` a four-cylinder engine model

`demo_translation` a MATLAB script defining a remote animation

DYNAanimation is also an integral part of the vehicle dynamics simulation program veDYNA.

In this manual the following topics are covered:

- Incorporating DYNAanimation into your Simulink model
- The DYNAanimation User Interface
- Generation of a Custom Animation
- Remote Control via ActiveX

1.1 What's New in DYNAanimation Version 1.2

1.1.1 Extension of Product Line

The product line has been extended to provide the right tool for different types of users.

The **DYNAanimation Viewer** can now be used without license on any computer.

For test purposes the new **DYNAanimation Demo** is provided. This version provides full functionality of the DYNAanimation Editor for 30 days.

Another new product is **DYNAanimation Edu**, the free version of the **DYNAanimation Editor** for education institutions and students.

1.1.2 Import / Export of Complete Animation Environments

For easier transfer of animations for presentations on different computers, the entire animation (including objects, geometries, simulation results) can be exported and imported.

1.1.3 Multi-Display Visualisation

The different views of the animation can be shown on different monitors.

1.1.4 Setting of Animation Speed

Recorded animations can now be played at a selected speed, either slow motion or fast motion. For a smooth visualisation the animation data is appropriately interpolated.

1.1.5 Welcome Screen

To ease the first steps with DYNAanimation, a ”Welcome Screen” provides help on selected topics as well as ready-to go simulation examples (Simulink models). Moreover, the operation of DYNAanimation is shown in video clips.

1.1.6 Merging of Different Simulation Projects

The merging functionality was extended, so that it is now possible to compare different simulation types. That includes simulations of varying simulation time.

1.1.7 Improved usage of Remote Control

The connection to the ActiveX interface was simplified.

1.2 The DYNAanimation Product Family

DYNAanimation is available in four different editions:

DYNAanimation Viewer is a tool to visualise the movement of mechanical objects in a virtual 3D world. This module is included in veDYNA Light and veDYNA Entry. The DYNAanimation Viewer can be downloaded free of charge at www.tesis.de/dynaanimation and does not require licensing

DYNAanimation Editor includes all the functionality of DYNAanimation Viewer. Additionally, an extended graphical user interface provides full control over the settings of the virtual world, including geometric objects, display controls, and cameras. Animations can be recorded and exported (avi, images, animation environment)
ActiveX controls enable remote control via MATLAB. A game control interface provides the option to move in the virtual world by means of a hardware device such as a joystick or, in case of a vehicle, a steering wheel.
This module is included in veDYNA Standard.

DYNAanimation Edu is a full version of the DYNAanimation Editor for education and students. The software can be downloaded free of charge at www.tesis.de/dynaanimation. A license can be obtained at this site, too.

DYNAanimation Demo is a full version of the DYNAanimation Editor for test purposes. The software can be downloaded free of charge at www.tesis.de/dynaanimation. A license which is restricted to 30 days can be obtained at this site, too.

1.3 DYNAanimation Features Overview

	Editor	Viewer
Load virtual worlds	**X**	**X**
Generate and edit virtual worlds	**X**	
Save and merge simulation results	**X**	
Load results	**X**	**X**
Save images / videos (avi)	**X**	
Slow Motion and Fast Motion replay	**X**	
Export simulation environment	**X**	
Edit camera settings	**X**	
Edit object settings	**X**	
Edit display settings	**X**	
Support of several views	**X**	
Change OpenGL settings	**X**	**X**
Full screen mode	**X**	**X**
Connect to simulation data (presently via Simulink$\hat{a}$g	**X**	**X**
Record simulation	**X**	**X**
Synchronisation with simulation	**X**	**X**
Step mode	**X**	**X**
Real-time mode	**X**	**X**
Game control input	**X**	
Control result animation	**X**	**X**
Online / Offline mode	**X**	**X**
External real-time boards	**X**	
Connection type Net / Bus	**X**	
Remote Control via ActiveX	**X**	

For further information on our product portfolio please see our internet page http://www.tesis.de.

1.4 DYNAanimation Hardware and Software Requirements

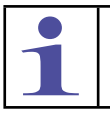
To run DYNAanimation you need a Windows 2000/XP PC. If using Windows 2000, please make sure that DirectX 9 or higher is installed on your computer.

We suggest using a 1,6GHz CPU or better. Another important prerequisite for satisfactory performance of DYNAanimation is a graphics card with OpenGL support. An overview is available at the official OpenGL website at http://www.opengl.org.

For best graphic performance it is desirable to install an actual video card driver. You can get information about recent drivers at the home page of your video card manufacturer.

Please note that OpenGL drivers are sometimes provided by the manufacturers of the graphics chip rather than by the makers of the board. DYNAanimation runs smoothly with graphics cards based on a NVIDIA chip (Geforce) or ATI (Radeon).

DYNAanimation supports MATLAB version 6.5 and higher. However, we recommend using MATLAB R14 SP1 and higher.



Please note that you need to be logged in as an administrator or a main user for the installation of DYNAanimation, as the installer attempts to write on the system registry.

1.5 DYNAanimation Directory Structure

The following figure shows the DYNAanimation directory structure after installation and configuration.

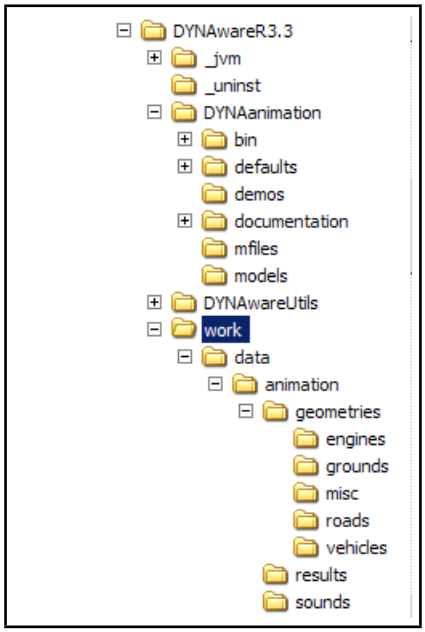


FIGURE 1.1: DYNAanimation directory structure for standalone installation (left)

The work – directory is meant to hold user data, such as animation projects, geometries, and results. A number of defaults is supplied.

2 Getting Started

2.1 Installation and Configuration

Execute the file `setupwin32.exe(` or `setupwin32_nojvm.exe`, if you have the Java Virtual Machine installed). The Installation assistant will guide you through the installation and copy the software to the indicated location on your hard disk.

After installation, the software needs to be configured. For this purpose start DYNAanimation from the Windows Start Menu. The installation has created the following entry

Programs | **TESIS DYNAware R3.3** | **Start DYNAanimation 1.2**

On activating this item, a configuration dialog will ask for the location and name of the license file and your MATLAB installation. Select the license file and your MATLAB installation directory in the respective file dialogs. Then the work directory has to be set. Select an existing directory or create a new directory in the file dialog, set a name and double click the OK button. A work directory containing supplied default projects and geometries will be created and DYNAanimation starts.

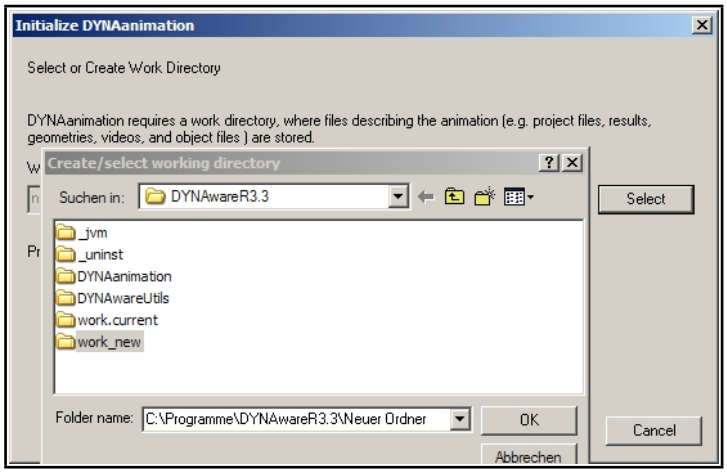


FIGURE 2.1: **File dialog for setting the work directory**

	Please note that now you have started DYNAanimation alone, without MATLAB. To run a simulation and generate simulation results to be animated you need a MATLAB environment. To start Matlab, activate the option **Simulation**	**Start Matlab . . .** from the DYNAanimation GUI.

For further details please refer to the Installation Manual.

2.2 Starting the software

Select **Programs** | **TESIS DYNAware R3.3** | **Start DYNAanimation 1.2** in the Windows start menu. DYNAanimation will start up (without Matlab). To run a simulation you need to start Matlab. This can be done by activating the option **Simulation** | **Start Matlab** in the DYNAanimation window.

If using DYNAanimation with veDYNA, first start veDYNA. In the veDYNA menu select **Extras** | **Animation** | **Select Animation Type** | **local** to activate the animation and then start DYNAanimation by means of the menu option **Extras** | **Animation** | **DYNAanimation**.

DYNAanimation starts with a default project (`default.dap`).

2.3 Deinstallation

If you wish to remove DYNAanimation from your computer, start the program `uninstaller.exe`, which has been installed in the directory `_uninst` under your installation directory.

3 The DYNAanimation Library

The following explanations are needed when using the DYNAanimation library blocks. As an Add-On to veDYNA DYNAanimation is part of the veDYNA Simulink model.

3.1 The DYNAanimation library blocks

Simulink library provides function blocks to be included into your Simulink model. The DYNAanimation library blocks are accessibe in the Simulink library browser under DYNAanimation blockset. The following blocks are included in the library:

- DYNAanimation Displays:

  This block is used to define displays such as gauges, plots, bar graphs

- DYNAanimation Objects

  This block is used to define animated objects

- DYNAanimation Game Controller Interface

  This block is used to configure the game controller interface

- DYNAanimation Realtime Control

  This block is used to synchronise your animation with the running simulation and to switch the real-time and step mode on/off.

3.2 Incorporating DYNAanimation into your Simulink model

Figure 3.1 shows the DYNAanimation subsystem connected to a vehicle single track model. The DYNAanimation subsystem contains the blocks DYNAanimation Display, DYNAanimation Objects, and DYNAanimation Realtime Control. The first two blocks allow the configuration of the data shown in displays and the data for moving objects, respectively.

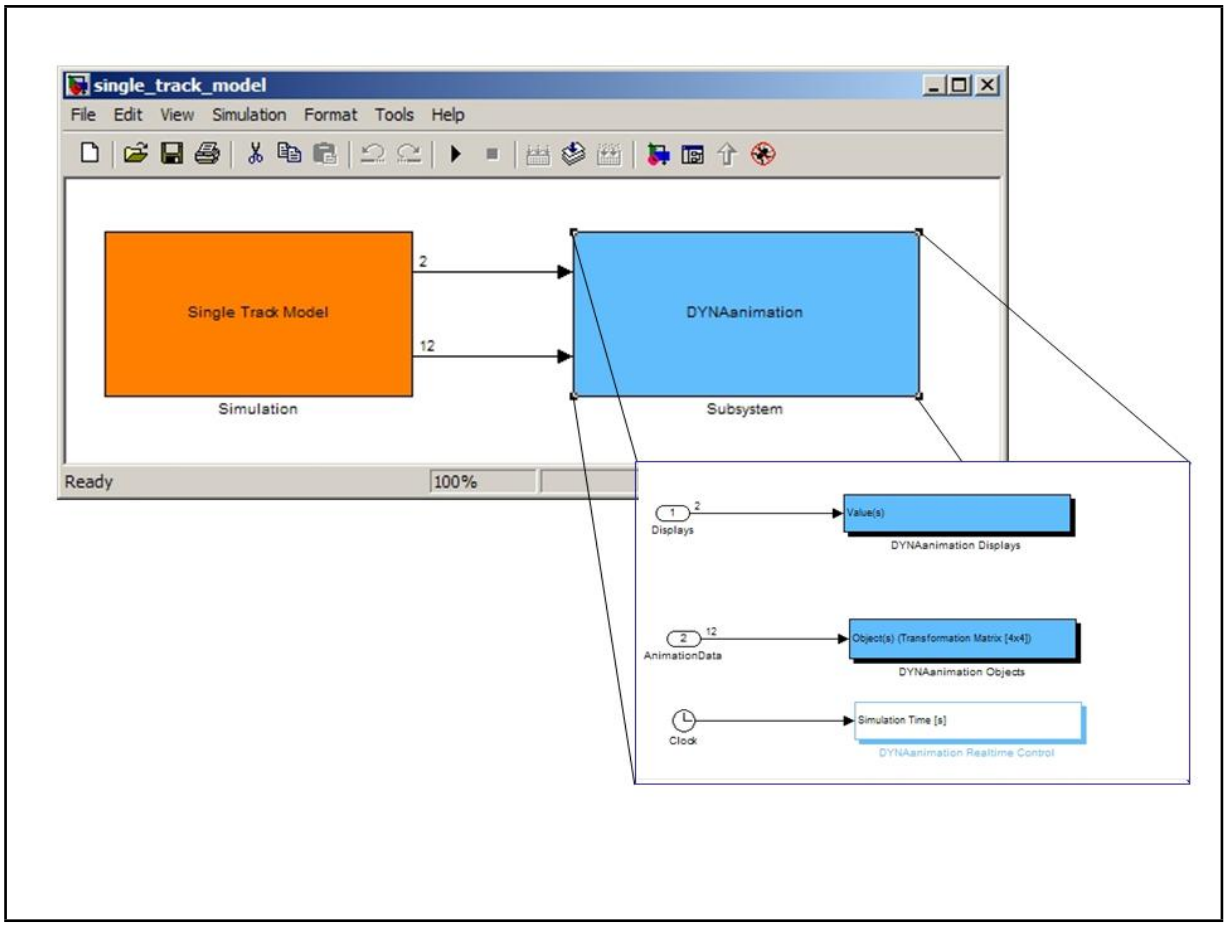


FIGURE 3.1: **Simulink Simulation Model Connected to DYNAanimation**

The DYNAanimation block requires data input from your model. Two types of data can be input into the DYNAanimation subsystem:

- For the displays, a vector of values to be shown in the displays like bar graph, plot, etc. (one value for each display)
- For the moving objects, a vector of animation matrices for the animated objects (one matrix for each object). Each animation matrix is stored in a vector of width 12.

3.2.1 Calculation of Motion Data for Input to DYNAanimation

For the description of the spatial position and orientation of a body three coordinates and three angles are required. This is what the simulation program usually produces as a result.

The choice of angles depends on the user and on the application. Commonly used sets of angles are cardan angles or Euler angles. The commonly used yaw, pitch and roll angles for describing vehicle motion (in this order) are cardan angles.

In order to have a uniform input for DYNAanimation, the transformation matrix that can be set up from these angles is required rather than the rotation angles. The transformation matrix describes the relation between the coordinates of a vector given in a coordinate system that is fixed to the object and the coordinates of that vector in the inertial system.

Animation data is arranged in the vector in the following order:

Elements 1-3 : coordinates (x,y,z) of object position

Elements 4-12: 3x3 rotation matrix, the matrix elements being arranged row-wise

For the specification of the object position select one reference point. This should be the point where the local

coordinate system originates and around which the rotation of the object is defined. Supply the inertial x,y,z position of this reference point.

In case of cardan angles the rotation of the object is described by subsequent rotations α, β, and γ around the x-axis, the y-axis (after the rotation), and the z-axis (after both rotations), respectively. Each rotation is described by an elementary transformation matrix $A_x(\alpha)$, $A_y(\beta$, $A_z(\gamma$.

$$A_X(a) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{pmatrix} \quad A_Y(\beta) = \begin{pmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{pmatrix}$$

$$A_Z(\gamma) = \begin{pmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$A = A_z \cdot A_y \cdot A_x$

The overall tranformation matrix (rotation matrix) is calculated by means of the multiplication

$A = A_z \cdot A_y \cdot A_x$

3.2.2 DYNAanimation Display Configuration

On opening the DYNAanimation Display block, the following configuration dialogue is invoked:

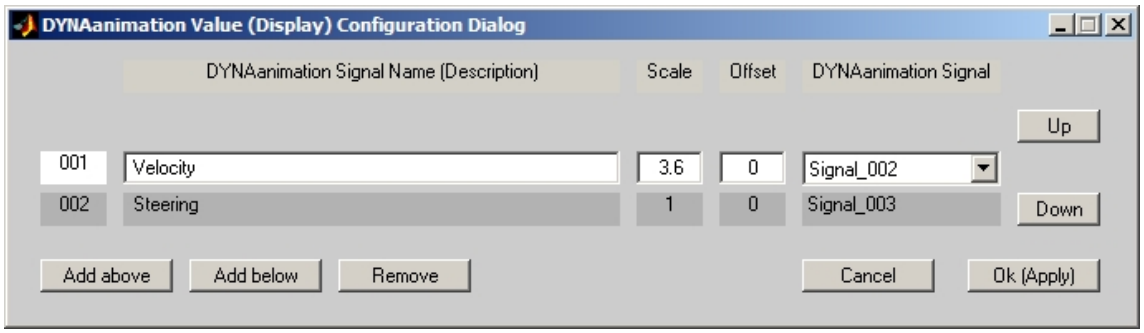


FIGURE 3.2: **DYNAanimation Display Configuration Dialogue**

In this dialogue, the values to be displayed in the animation are shown in one row each ordered according to their position in the input vector. It is possible to navigate through the data by means of the up and down buttons at the right hand side. The currently active data element is highlighted.

Each display value has to be assigned to the corresponding signal list in DYNAanimation:

1. Assign a signal name under which the current value can be addressed in the DYNAanimation signal list.
2. Set the position of the current data entry in the DYNAanimation signal list. Per default, the signals will appear as they are arranged in the data vector from the simulation model. The first position in the DYNAanimation is reserved for the simulation time.

It is also possible to prescribe scaling factors and offsets. This is useful for using different units in the simulation and the animation, e.g. displaying a velocity which is calculated in m/s in km/h.

Having configured the displays and the objects, the simulation should be started for a short time to initialise the connection.

Now the signals are available in the DYNAanimation user interface (see next chapter). Open the Display Dialog to include the required objects or displays and connect to the corresponding signal.

3.2.3 DYNAanimation Object Configuration

On clicking the *DYNAanimation Object* block, the following configuration dialogue is invoked:

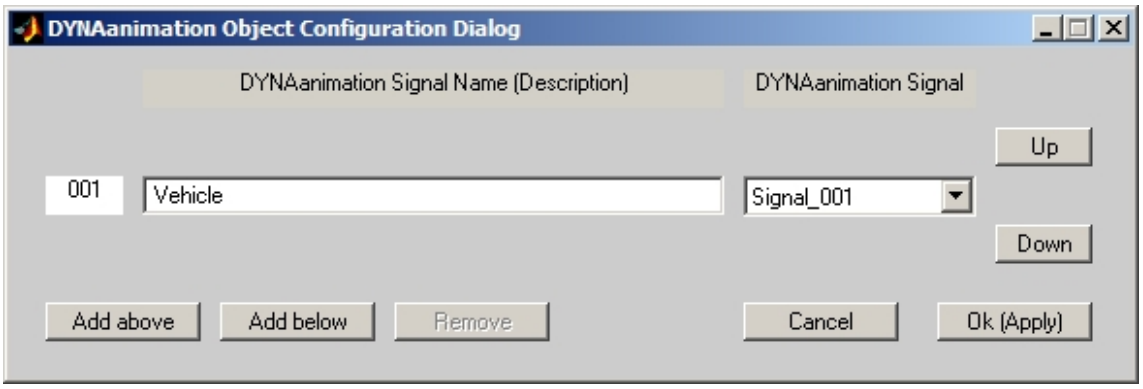


FIGURE 3.3: **DYNAanimation Object Configuration Dialogue**

In case of animated objects, the signals from the simulation model are shown in one row each according to their order in the block input. Please note that the signal for each object is the animation matrix for the object.

It is possible to navigate through the data by means of the up and down buttons at the right hand side. The currently active data element is highlighted.

Assign the signals to the corresponding signal in in the DYNAanimation list as follows:

1. Prescribe a signal name under which the current value can be addressed in the DYNAanimation signal list.
2. Set the position of the current data entry in the DYNAanimation signal list. Per default, the signals will appear as they are arranged in the data vector from the simulation model.

Having configured the objects, the simulation should be started for a short time to initialise the connection.

Now the signals are available in the DYNAanimation user interface (see next chapter). Open the Object Dialog to include the required objects and connect them to the corresponding signals.

3.3 The Game Controller Interface

The *Game Controller Interface* block in the DYNAanimation Simulink library enables the manipulation of the simulation model via external steering wheel or game controller. In addition, forces can be transferred to the connected device (force feedback). The data flow from and to the external device is handled by this block.

To use this feature, open the DYNAanimation Simulink library and copy the *Game Controller Interface* block into your model. The block input is the steering wheel torque for the force feedback, which has to be computed by the simulation model, and the gear range. The block output is a vector *Driver Input* with the components

- Steering Wheel Angle
- Throttle Position
- Brake Position
- Gear Position

These values have to be forwarded to your model.

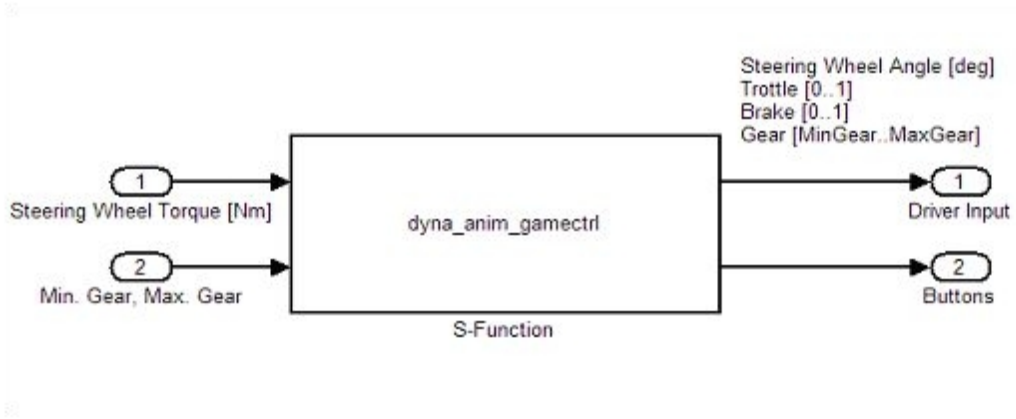


FIGURE 3.4: **Game Controller Interface block**

To use an external steering wheel or joystick to drive a veDYNA model, proceed as follows:

1. Connect and install the steering wheel.
2. Include the *Game Controller Interface* into the veDYNA Simulink model.
3. Connect the Driver Inputs with the Maneuver Controller Inputs Block.
4. Select the veDYNA simulation procedure go_gamecontroller.
   This procedure contains the following maneuver settings:
   longitudinal: no internal vehicle motion, external source for accelerator pedal position
   lateral : no internal vehicle steering, external source for steering wheel angle
   constraints: external source for brake pedal position, normal gear range,
   tyre friction scaling factor 1
5. Set the simulation time to about 60 seconds or more, that you can drive for some time.
6. Start DYNAanimation and enable the Game Controller and force feedback.

4 The DYNAanimation User Interface

4.1 Main Window

On starting DYNAanimation (or on activating **Extras** | **Animation** | **DYNAanimation** in the veDYNA main GUI) the animation window opens, which may look as shown in the following figure

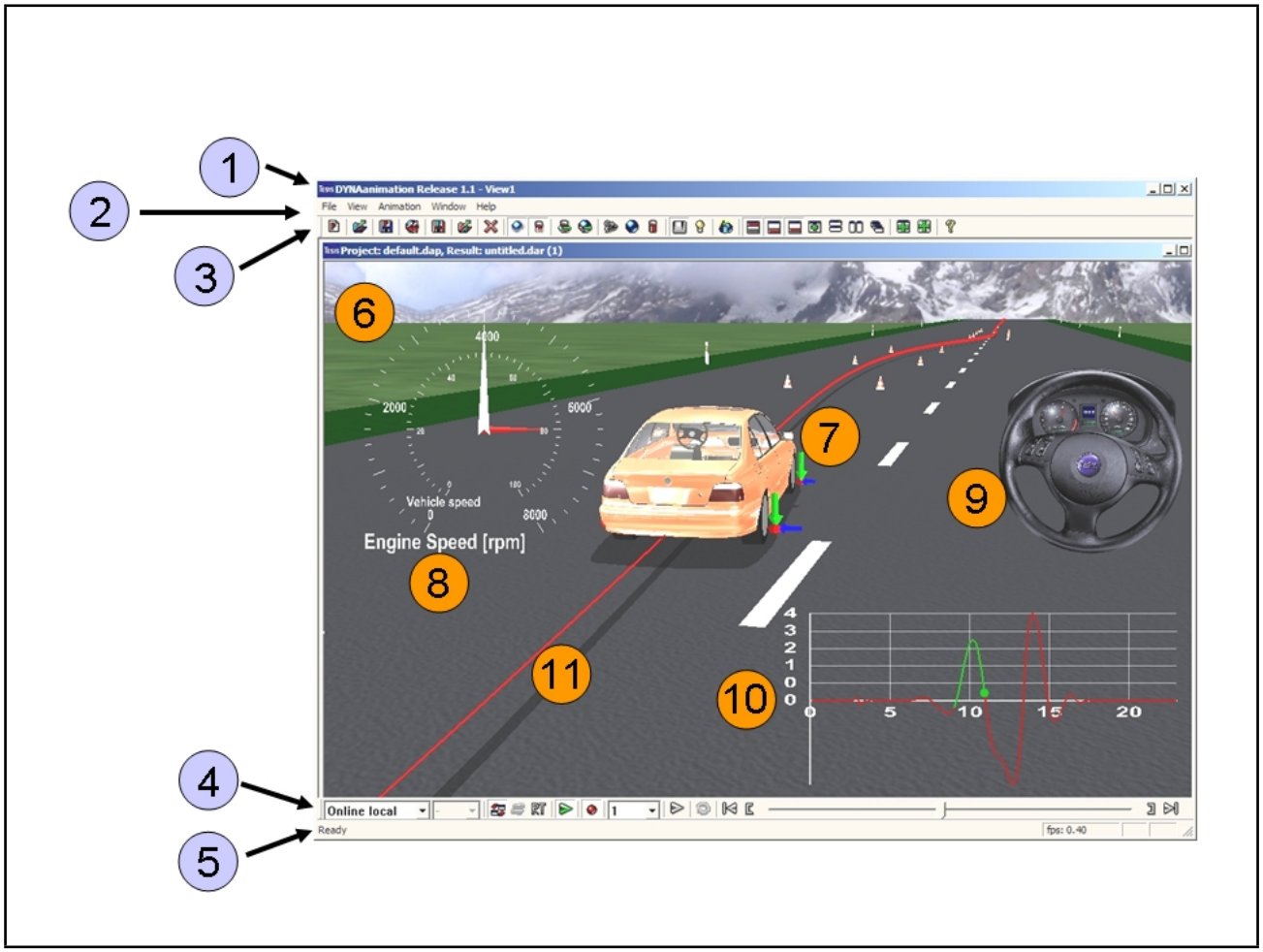


FIGURE 4.1: **DYNAanimation Main Window with Landscape, Vehicle and a Selection of Displays**

The animation window is composed of the following parts:

1. **Title Line:** Shows the loaded project file defining the loaded virtual world with all contained 3D objects, displays and settings and the loaded result file containing the simulation data.
2. **Main Menu:** The main menu controls file access, view control, animation control and window control.
3. **Main Toolbar:** The context sensitive main toolbar gives direct access to important features of DYNAanimation via function buttons.
4. **Control Toolbar:** The context sensitive control bar provides control over the animation.
5. **Status Bar:** Gives status information at runtime and displays information on the functions to the main

toolbar. It can be switched on and off by the option **View** | **Toolbars, Settings & Dialogs** | **Show Status Bar** or by pressing the shortcut key F3.

(6) **Animation Window:** This window shows the three-dimensional view of the animation. DYNAanimation allows opening several windows. We recommend to use a mouse with three buttons and a wheel to navigate in the three dimensional scene. The position of the camera can be changed via the mouse as follows:

Left button: Sets the X- and Y-position of the camera.

Centre button: Sets the Z-position of the camera.

Left and right button together: Sets the orbital rotation of the camera about a selected object.

Right button: Sets the Y- and Z-rotation of the camera.

Mouse wheel: Zooms in and out.

Left double click: Moves to a standard camera position (behind the vehicle)

(7) **Simulated Objects:** This is an example of an animated mechanical system showing the simulated motion.

(8) **Gauge Display:** This display can be used to visualise a tachometer for measuring speeds

(9) **Cockpit Display:** One of the more complex displays supported by DYNAanimation showing engine speed, vehicle speed and position of the steering wheel like in the cockpit of a car.

(10) **Plot Display:** Shows curve plots of selected simulation results over time

(11) **Trace Display:** Shows the trajectory of the moving object. This display can only be used off-line, i.e. for a simulation that has been successfully performed before.

It is also possible to press the SPACE bar instead of a mouse button to set the camera position. Pressing the SPACE bar again deactivates this modus. In addition, it is possible to navigate by means of the arrow keys of the keyboard.

Objects may also be moved, rotated and scaled using the mouse, for further details see Object Dialog

4.2 File Access

The **File** menu provides control over projects and the animation results. The projects (dap-files) contain the information on the appearance of the animation, e.g. the view settings, the used objects, displays, camera perspectives, etc. Different projects can be generated, saved for later use and reloaded when needed. It is also possible to set a default project which is automatically loaded on starting DYNAanimation.

The installation comes with default projects which are suitable for animations of a veDYNA passenger vehicle (default.dap) and a veDYNA truck (default_truck.dap). In case of changes of the vehicle parameters in veDYNA, e.g. when activating an additional rear axle, the project is automatically updated and stored under the name of the current vehicle data set (e.g. `bmw_325i_88.dap` or `limousine.dap`).

An animation can be exported as a video clip (avi), or the current frame can be exported as an image by means of the options **File** | **Save avi** or **File** | **Save Image**, respectively (see also Generate Video / Image). Proceed as follows:

1. Load a DYNAanimation project and perform a simulation. You need to record the animation.
   or
   Load a project file containing a scene and a stored result file
2. Activate the option **File** | **Save Avi** in the main menu
3. Enter a file name in the file dialogue

4. Another dialogue requests the selection of a video codec (e.g. XviD Mpeg4-Codec or any codec present on your computer). If you wish to play the video on another computer, you should choose a codec which is commonly used. It is also possible to configure the settings of your codec to influence the video quality and the resulting size of the avi file.
5. Press Ok to start the generation of the video file. You can watch the video during the export process. It is also possible to stop the export at any time.

A free codec which is appropriate for the video generation is the XviD MPEG 4 codec. To install this codec on your computer, download XviD from the URL http://www.xvidmovies.com/codec.

After installing the software select *Configure* and then *Advanced Options / Debug* from the configuration dialog. Deactivate the option *Display according status*.

The option **File** | **Save Results as . . .** allows saving recorded simulation results (*.dar). Saved results can be reloaded with the option **File** | **Load Results.**

A special option is **File** | **Merge Results** to enable the simultaneous display of several simulation runs for comparison or for animating traffic, see also Show Different Simulation Runs in one Animation for further details. Please note that the animation project has to be adjusted to show the additional moving objects (vehicles). There has to be one additional object for each simulation run to be merged.

The options **File** | **Import** and **File** | **Export** allow importing and exporting entire animations, including the scenery, geometries, simulation results, and view settings. The animation information is stored in one file with the extension `die` (dynaanimation import environment).

The option **File** | **Settings** allows setting and changing the license file and the work directory settings. On activating the Change button, a file dialog will ask for location and name of license file and work directory.

The main toolbar also allows loading and saving projects and result files.

4.3 View Control

The **View** menu provides control over the look of the animation as well as the user interface. Most of the view control functions are also accessible via the main toolbar.

The option **View** | **Toolbars, Settings and Dialogs** allows showing or hiding toolbars as well as dialogs for camera settings, object settings and display settings.

The option **View** | **Semi-Fullscreen Mode** (shortcut key F9) hides all toolbars and dialogs except the main menu.

The option **View** | **Fullscreen Mode** (shortcut key F10) maximises the animation window. To end the full-screen mode, hit ESC or F10 again.

The **View** menu also provides functions to toggle between camera settings (shortcut keys Page Down for **Next Camera**, Page Up for **Previous Camera**) as well as to show or hide objects and head-up displays.

The **View** | **Edit Display** option allows positioning and sizing the displays by means of the mouse. This mode can be also switched on and off by the shortcut key Shift+F3.

The appearance of the animation can be influenced by a number of visual settings available with the option **Open GL Settings:**

- Use Wire Frame
- Use User Defined Background
- Use Lighting
- Use Smooth Shading
- Use Anti-aliasing
- Use Textures
- Use Fog

- Use Shadows
- Use Complex Background
- Use Motion Blur
- Use Soft Shadows
- Use Glow

These features may be turned on and off either separately by activating the respective item or by selecting one of the options **View** | **All Features On** / **All Features Off**.

4.3.1 Use Wire Frame

Instead of filled areas a wire frame display is generated for all objects.

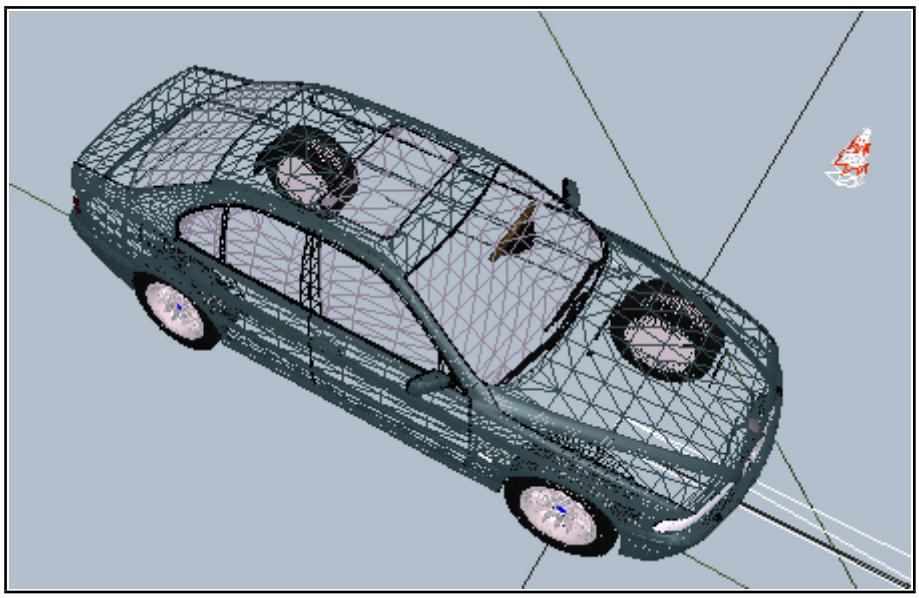


FIGURE 4.2: **Wire frame view of the vehicle**

4.3.2 User User Defined Background

An image can be loaded as background.

4.3.3 Use Lighting

The lighting option will activate the calculation of colour gradients. The display of an object is influenced as shown in below in Figure 4.3.

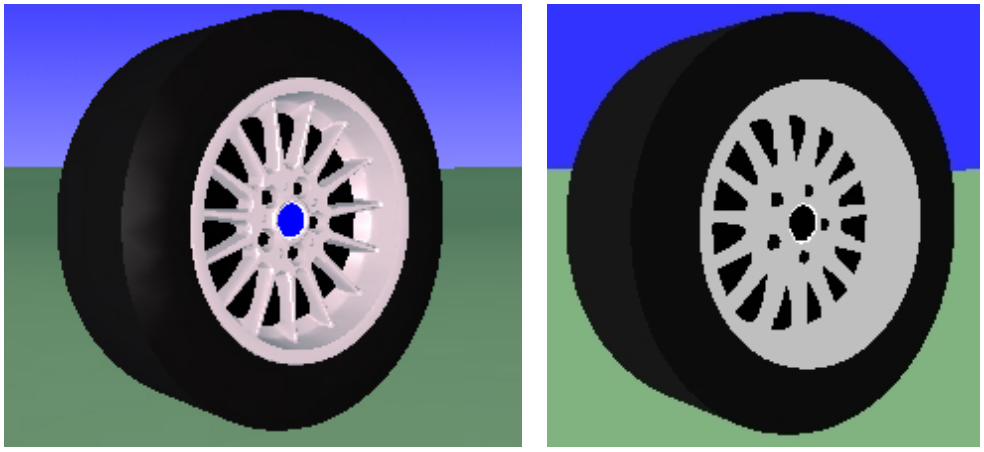


FIGURE 4.3: **Influence of the Lighting option: Switched on (left) and off (right)**

4.3.4 Use Smooth Shading

This option influences the smoothness of the surface display as shown in the following figure.

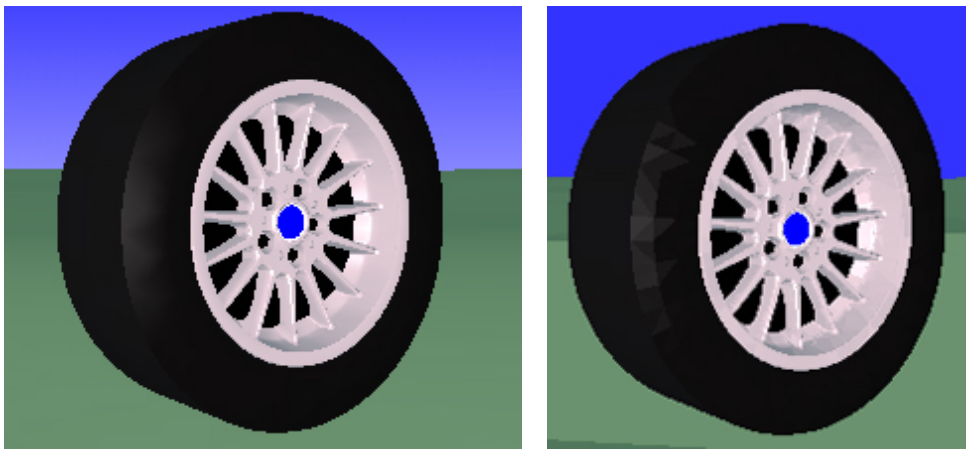


FIGURE 4.4: **Smooth Shading option: Switched on (left) and off (right)**

4.3.5 Use Anti-aliasing

This option will lead to an improved display of lines, especially for video export.

4.3.6 Use Textures

This option enables the usage of textures to enhance the display quality of objects like road and background. An example rendering is shown below in Figure 4.5.

FIGURE 4.5: **Influence of the Use Textures option: Switched off (left) and on (right)**

4.3.7 Use Fog

This option adds fog to the scene as shown below in Figure 4.6.

FIGURE 4.6: **Display of Fog**

4.3.8 Use Shadows

Object shadows are shown.

4.3.9 Use Complex Background

This option allows including a complex background. Presently only one background is provided. See Figure 4.7.

FIGURE 4.7: **Display of complex background**

4.3.10 Use Motion Blur

This option blurs the object motion.

4.4 Animation Control

4.4.1 Run and Stop Animation, Reset to Start

Animation control includes running and stopping the animation, as well as resetting simulation results to the start. These options are also available in the control bar.

4.4.2 Animation Speed

The animation speed can be varied from slow motion (1/20 of original speed) to fast motion (10x original speed). To guarantee smooth motion, the animation data is interpolated if a different speed than the original is selected.

4.4.3 Animation Mode

The animation mode for online simulation (step, real-time or synchronous with the simulation) can be set. Recording and the connection of signal data can be enabled.

In case of offline animation it is possible to enable the loop mode (animation will automatically restart after having reached the end of the simulation time).

4.4.4 Enable Game Controller

This option is only active, if a joystick is connected to the computer. If activated, the joystick signals will be forwarded to your model. To use the joystick inputs in your model, you need to include the Simulink block *DYNAanimation Game Controller Interface* to your simulation model (see also Game Controller Interface). This block generates appropriate inputs to your model. Presently the *DYNAanimation Game Controller Interface* block is intended to be used for operating a vehicle via the game control input.

4.4.5 Start / End Frame

It is possible to define a time interval (set an upper and lower time limit) for replay. In case of video export, only the set time range will be exported. The animation can be reset to the start time, which will be the first

frame of the selected interval. Most of these functions are conveniently accessible via the control bar.

4.5 Simulation

The option **Simulation** | **Start MATLAB** opens the simulation model environment.

4.6 Window Control (for Multi-Window Setup)

It is possible to use several windows, e.g. to view the animation from different perspectives at the same time. An additional window can be created in three ways:

- Select **Window** | **Additional Animation Window** in the main menu
- Press the button in the main toolbar to create an additional animation window.
- Use the shortcut Alt+V

The mouse can be used to position and resize the new window. The windows can also be arranged by means of the options **Window** | **Cascade**, **Window** | **Tile Horizontal** and **Window** | **Tile Vertical** in the main menu. The size of the active window can be set by means of the option **Window** | **Set Window Size.**

For the active window, dialogs to set the camera perspective and to add objects or displays are available.

The active window can be undocked from the main DYNAanimation window by means of the toolbar function ”Floating Window”. Double clicking the title line displays this window on the full screen. Another double click brings the window back to the original size. Clicking the (**Floating Window**) button again and double clicking the title line docks the window again.

The floating window can be used for a screen in screen display of multiple views, e.g. the rearview mirror. Different windows and views can be shown on different monitors, if required.

4.7 Camera Dialog

To open the camera dialog press the button of the main toolbar, or select the option **View** | **Toolbars, Settings & Dialogs** | **Show Camera Settings** or use the shortcut key F4. The camera dialog can be undocked from the DYNAanimation window, it is then possible to move the Camera Settings window freely on your monitor. The camera dialog is shown in the following figure.

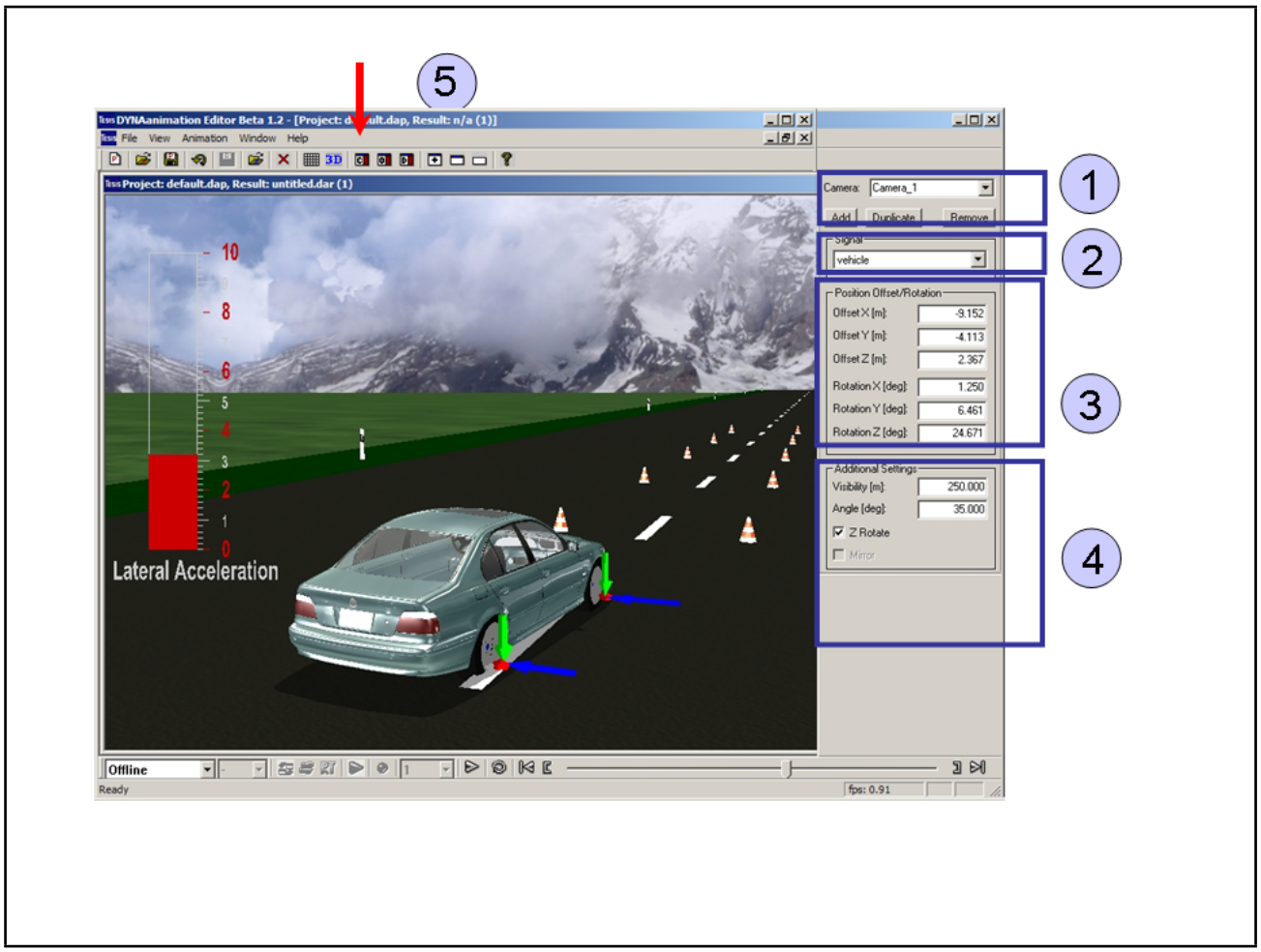


FIGURE 4.8: **Camera Dialog to set View Perspective of Animated Objects**

1. **Main Camera Settings:** In the camera dialog, a camera (view perspective) can be added, duplicated or removed. The name of the selected camera setting in the drop-down menu can be changed.

2. **Signal:** The selected camera will be connected to the selected signal.

3. **Position Offset** / **Rotation:** For each camera the position and angular offset can be set to define the perspective.

4. **Additional Settings:** The visibility can be set by means of distance and apex angle.

   The option **Z Rotate** makes the camera follow the yaw motion of the connected object.

4.8 Object Dialog

3D objects are necessary to build a virtual world. DYNAanimation allows the import of VRML 2.0 objects. Some objects, e.g. a road, are static and therefore not connected to any simulation signals. An object representing a dynamical mechanical system, such as a vehicle, must be connected with a signal from the signal list to animate the motion. The signals appearing in the signal list are the ones that have been assigned in the object configuration dialog (see Incorporating DYNAanimation into your Simulink model). In case of using DYNAanimation together with veDYNA, objects and signals are preconfigured. For special applications it is possible to extend the object list and the corresponding signals.

Objects may be assigned further properties to adjust their appearance in the animation.

To set 3D objects and object properties, open the object dialog by pressing the button of the main toolbar or select the option **View** | **Toolbars, Settings & Dialogs** | **Show Object Settings** or use the shortcut key F5.

A dialog window as shown in the following figure opens. The object dialog window can be undocked from the DYNAanimation window, it is then possible to move the Object Settings window freely on your monitor.

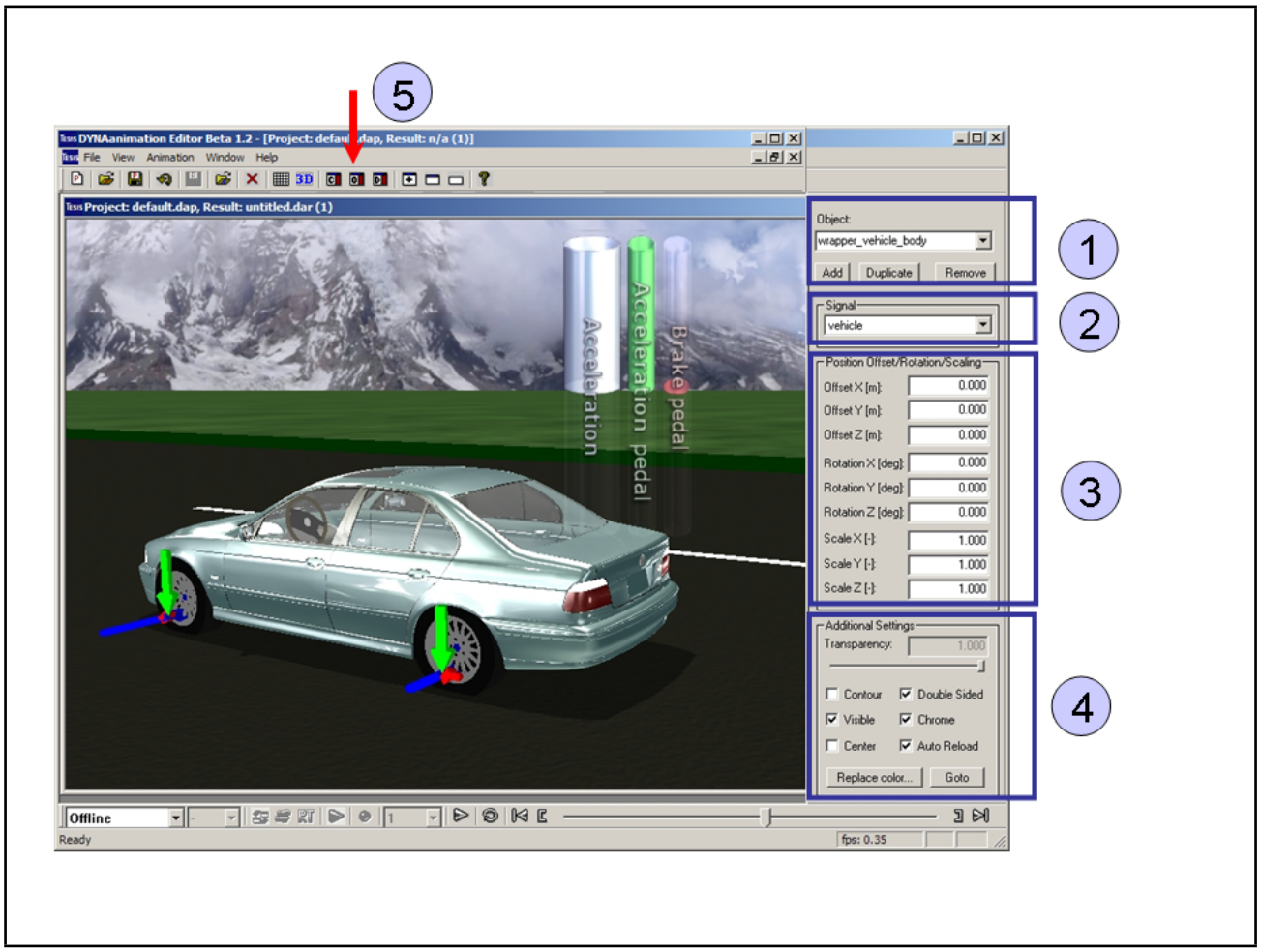


FIGURE 4.9: **Object dialog**

1. **Main Object Settings:** In the object dialog, 3D objects can be added, duplicated or removed. The name of the selected object in the drop-down menu can be changed. All object files must be loaded as VRML-object. DYNAanimation can import all objects which support the VRML 2.0 format.

2. **Select Signal:** The selected object will be connected to the selected signal from the signal list.

3. **Position Offset / Rotation / Scaling:** For each 3D object offset, rotation, and scale can be set.

4. **Additional Settings:** Here, it is possible to set the transparency degree by means of a slider. Moreover the following object properties can be set:

   - Contour mode (no filled areas, only contours are displayed)
   - Visibility
   - Center (correction for objects with origin not in the object centre)
   - Double Sided (hidden edges and surfaces are calculated)
   - Chrome (gives a metallic shine to the object surfaces)
   - Auto Reload (when geometry is changed, e.g. during vehicle update)

   The button Replace Colour opens a dialog which shows all colours used for the current objects. Clicking one of the colour squares invokes a slider window which shows the RGB components of the selected colour. By moving the sliders, the colour can be adjusted.

(5) The object dialog can be opened via the main toolbar.

To move and rotate an object, move the mouse pointer to that object. Then press the STRG key and click the left mouse button. The selected object is highlighted, and a coordinate system is displayed. For moving the object, press the left mouse button and move the mouse along the axes of the coordinate system. To rotate the object, press the right mouse button and move the mouse along the axes of the coordinate system. Press left and right button together to scale the object.

4.9 Display Dialog

Displays can be used to add special graphical elements to the animation and to insert displays of selected simulation results. Presently the following display types are available:

Display Name	Description
Text	Textual display of simulated value, including caption
Gauge	Tachometer-like display, including caption
Vertical Bar	Bar graph, including caption
Horizontal Bar	Bar graph, including caption
Acceleration	Display of longitudinal acceleration, position of gas pedal and brake pedal in columns (3 signals are expected for this animation)
Gears	Photorealistic display of gearshift lever
Rear View	Shows the rear view of a vehicle, should be connected with the roll angle
Side View	Shows the side view of a vehicle, should be connected with the pitch angle
Cockpit	Photorealistic display of a steering wheel with view on engine speed and vehicle speed (3 signals are expected for this animation)
Logo	A company logo can be displayed somewhere in your animation (static)
Plot	The selected signal is drawn as a curve plot. It is possible to load a signal from a previous simulation that will be displayed statically.
Trace	The trajectory of the moving object will be shown as a line. The object to be traced has to be selected in the signal list. This display does not work on-line, i.e. you need to calculate and store the simulated trajectory before you can load it into your animation.
Ruler	a configurable sign-posting to show a distance on a road, e.g. a braking distance

To include displays and set display properties, open the display dialog by pressing the button  of the main toolbar or select the option **View** | **Toolbars, Settings & Dialogs** | **Show Object 2D Settings** or use the shortcut key F6. A dialog window as shown in the following figure opens. The display dialog can be undocked from the DYNAanimation window, it is then possible to move the Display Settings window freely on your monitor.

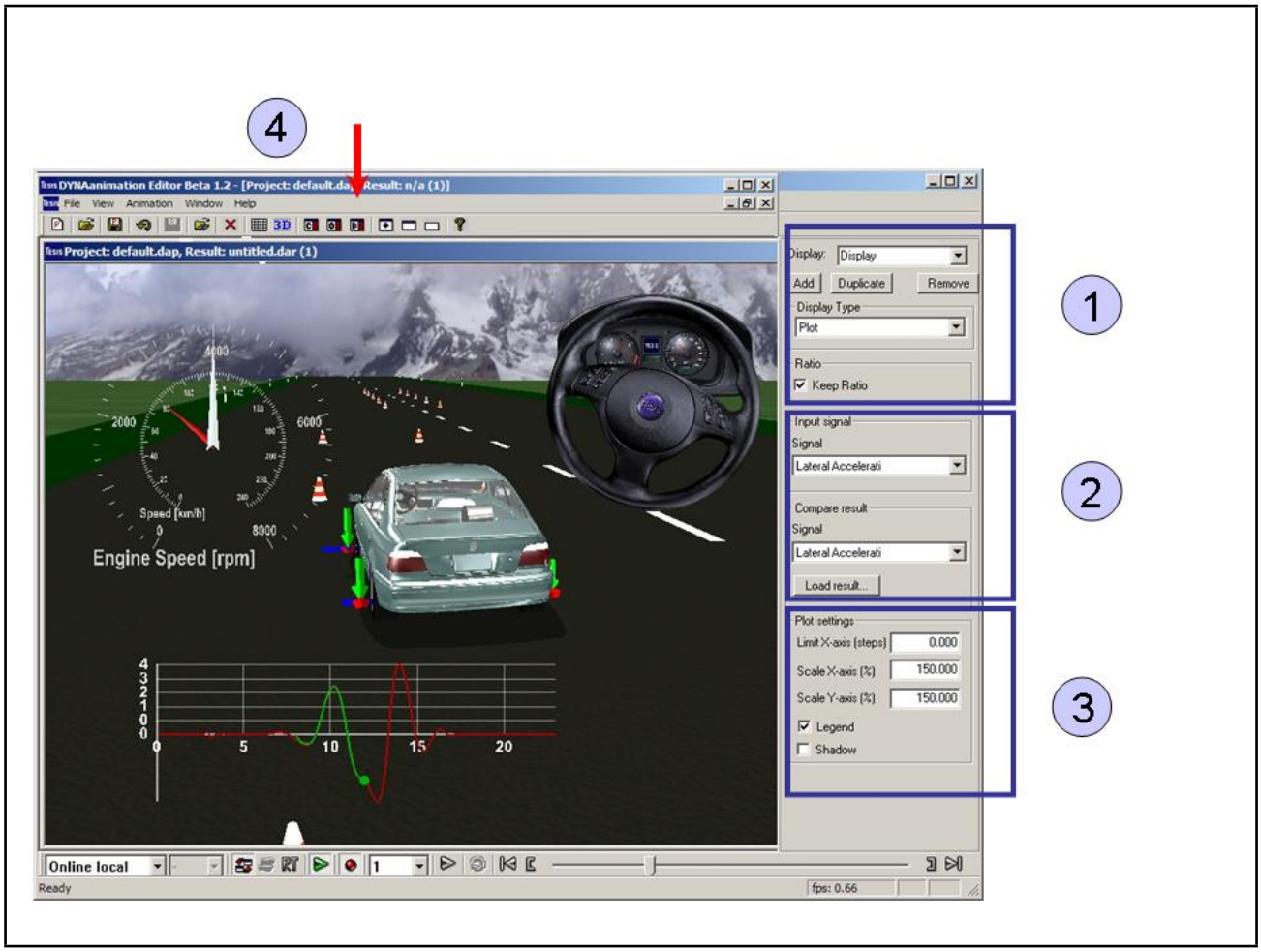


FIGURE 4.10: **Display dialog**

1. **Main Display Settings:** In the display dialog, displays for each open window can be added, duplicated or removed. The name of the selected display in the drop-down menu can be changed. In the pull-down menu **Display Type** one of the available displays can be selected.

2. **Input Signal:** Each display can be connected to one or more signals of the signal list, depending on the display type.

3. This context sensitive area allows the specification of the selected display. Moreover, the display's appearance, e.g. colour, transparency, minimum and maximum values, captions, etc. can be set

4. **Edit Displays (grid on/off):**If the grid is turned on, a display can be moved within the animation window by pressing the left mouse button on the display and moving the mouse pointer to the required position. The size of the display can be changed by pressing the right button on the display and moving the mouse pointer towards the upper left or lower right corner.

4.10 Main Toolbar

The main toolbar provides easy access to common DYNAanimation functions. The toolbar can be switched on and off by means of the toolbar button or by activating the option **View** | **Toolbars, Settings & Dialogs** | **Show Main Toolbar** of the main menu or by pressing the shortcut key F1. On moving the mouse pointer over one of the buttons a short explanation of the respective button is shown in the status bar.

FIGURE 4.11: **Main toolbar**

The following functions are accessible via the main toolbar (from left to right)

Function Class	Icon	Description
File Settings:		Open default project (and close current project)
		Load existing project (and close current project)
		Save current project
		Reload geometries
		Load existing results
		Close project
Object & Display Settings		Move Displays, (allows moving and resizing displays with the mouse)
		Selection mode for objects on / off (for moving, rotating, and scaling)
Dialogs		Open camera dialog
		Open object dialog
		Open display dialog
Window Settings		Create additional window
		Floating window: undocks the dynaanimation window and allows moving this window independently from the tool bar and menu bar
		Window Frame: removes the title line from the floating window.
		About: information on DYNAanimation

4.11 Control Bar

The control bar can be switched on and off by activating the option **View** | **Toolbars, Settings & Dialogs** | **Show Control Bar** or by pressing the shortcut key F2. On moving the mouse pointer over one of the buttons a short explanation of the respective button is shown in the status bar.

FIGURE 4.12: **Control bar**

The source from which DYNAanimation receives simulation data can be switched between **Offline**, **Online local** and **External** real-time boards by means of the leftmost drop-down box.

The **Online local** mode connects DYNAanimation with the current Simulink model. When a simulation is started, DYNAanimation receives the results.

Offline mode indicates that a recorded animation or a loaded result file is to be replayed.

On selecting a connection to a real-time board, the drop-down box with the supported connection types (**Net** or **Bus**) is activated. If a supported real-time board is connected, DYNAanimation can receive simulation data from the selected real-time board.

In case of **Online local** mode, the following options are available:

- The button switches the synchronisation between DYNAanimation and the simulation with the Simulink $\hat{a}$ model on or off.
- The step mode button enables the animation of all simulation steps.
- The button real-time mode makes the animation run in real time. If disabled, the animation runs synchronous to the actual simulation speed of the computer, that means faster or slower than real time.
- The button initiates and stops recording of the animation data during simulation.
- The drop-down menu for record downsampling controls the frequency of data storage during simulation.

After recording an animation the buttons to control the replay of animation results are enabled.

FIGURE 4.13: **Subset of control bar for animation control**

These buttons allow the following actions (from left to right):

	starts and stops the replay of the animation results
	automatically restarts the replay of the animation results when finished
	moves the replay of the animation results to the first frame
	changes the first frame to a later position
The slider	allows changing the current frame position
	changes the last frame to an earlier position
	moves the replay of the animation results to the last frame

4.12 Moving and Rotating Objects via Mouse Control

Objects may be comfortably moved, rotated and scaled using the mouse.

To move or rotate an object, move the mouse pointer to that object. Then press the STRG key and click the left mouse button. The selected object is highlighted as shown in the figure below, and a coordinate system is displayed. For moving the object, press the left mouse button and move the mouse along the axes of the coordinate system. To rotate the object, press the right mouse button and move the mouse along the axes of the coordinate system. Press left and right button together to scale the object along the respective axis.

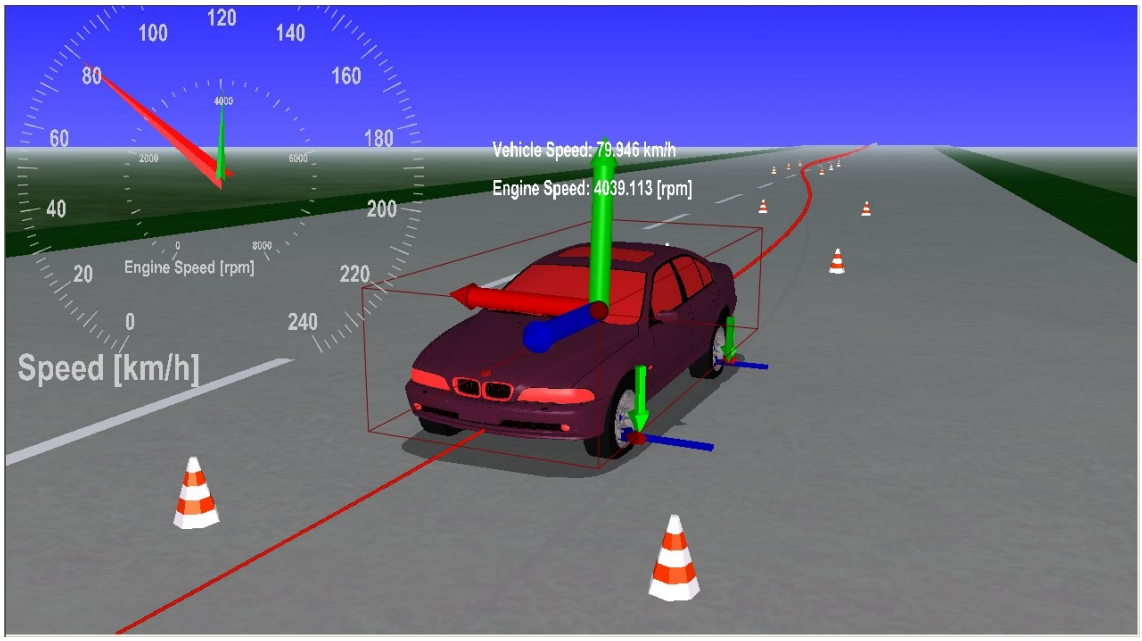


FIGURE 4.14: **Moving or rotating a selected object**

Another option to select an object is via the button (Select Object mode) of the tool bar. Clicking the selected object highlights this object and shows the coordinate system, double-click zooms to that object. Then the positioning can be performed via mouse movements along the axes as described above.

4.13 Camera Settings

Camera settings can be controlled directly via the mouse, too.

Left button: Sets the X- and Y-position of the camera.

Centre button: Sets the Z-position of the camera.

Left and right button together: Sets the orbital rotation of the camera about a selected object.

Right button: Sets the Y- and Z-rotation of the camera.

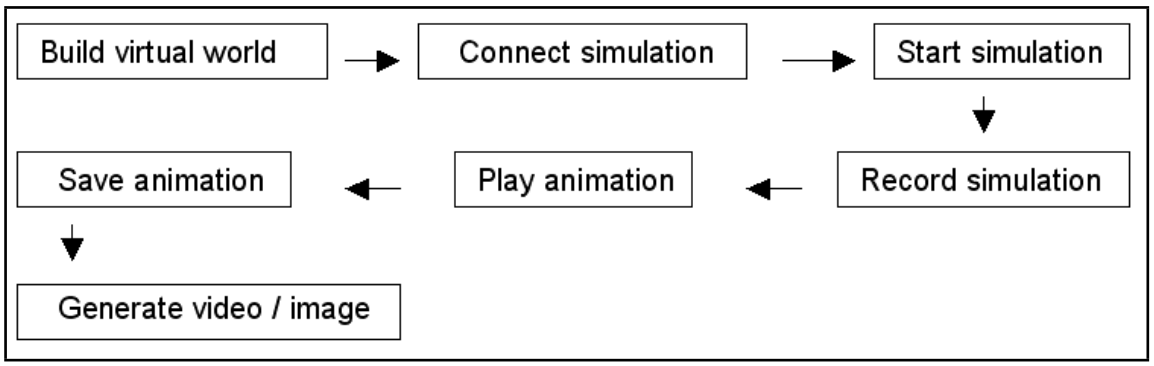
Mouse wheel: Zooms in and out.

Left double click: Moves to a standard camera position (behind the vehicle)

It is also possible to press the SPACE bar instead of a mouse button to set the camera position. Pressing the SPACE bar again deactivates this modus.

5 Generation of a Custom Animation

In the next few sections the procedure to generate a custom animation from a simulation will be introduced step by step. Each step is explained by means of an example.



5.1 Build a Virtual World

A virtual world (scene) is made up of various 3D objects. The following steps describe how to create this world with DYNAanimation. You will use static and movable objects for the scene.

5.1.1 Step 1: Start DYNAanimation

A stand-alone installation of DYNAanimation is started via the Windows start menu.
Select **Programs** | **TESIS DYNAware R3.3** | **Start DYNAanimation 1.2**. DYNAanimation will start up without Matlab. To start Matlab, either select the option **Simulation** | **Start Matlab** in the DYNAanimation window or execute one of the example simulation models from the Welcome Screen. If using DYNAanimation with veDYNA, first start veDYNA. In the veDYNA menu select **Extras** | **Animation** | **DYNAanimation**

DYNAanimation starts with a default project. Click **File** | **New Project** in the DYNAanimation menu to initialise an empty project.

5.1.2 Step 2: Import Static 3D Objects

An empty project shows only the background without any objects as can be seen in Figure 5.1. DYNAanimation can import 3D objects which support the VRML 2.0 format.

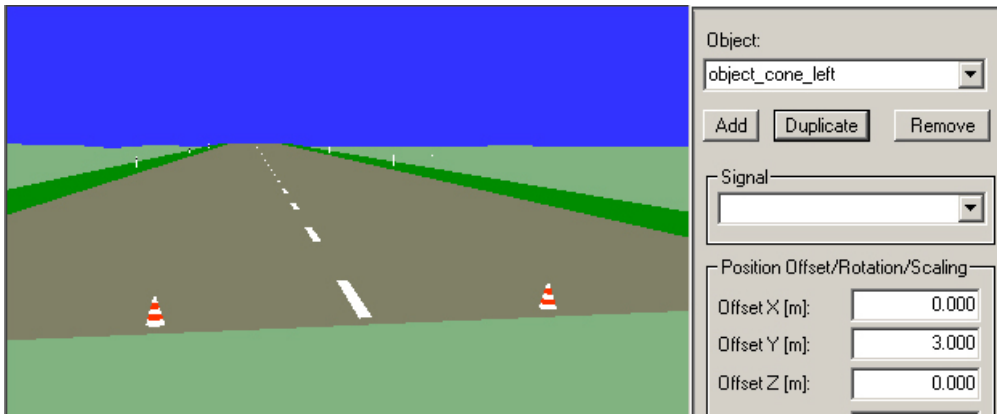


FIGURE 5.1: **Scene resulting from empty project**

To import an object, open the Object Dialog and click the Add button. A file dialog is invoked. Change to the directory of the object database `~work.current\data\animation\geometries`. To import a static object, e.g. a hilly ground, open the subdirectory `grounds` and select `ground.wrl` .

Another object describing a scene is a road. Click Add and load the file `road_straight_flat.wrl` from the subdirectory `roads`.

To add further objects, e.g. a cone, click Add and import `object_cone.wrl` from the subdirectory `misc`. If you wish to import more than one instance of an object, mark the name of this object in the drop-down-list and change it. Here the name of the cone is changed to *object_cone_left* and an offset is defined (Offset Y = 3). To add another cone at the right hand side, click Duplicate and change the Offset Y to –3, then rename this object to *object_cone_right*.

5.1.3 Step 3: Import Dynamic 3D Objects

The animated object in this case consists of a vehicle body, a steering wheel and tyres, which all can move. Therefore you need to import these objects separately.

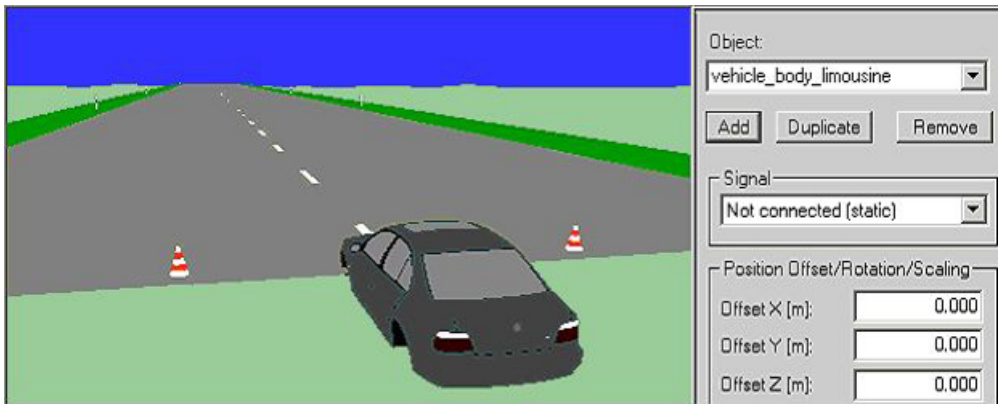


FIGURE 5.2: **Including a dynamic object**

Click on the Add button and change to the subdirectory `vehicles` of the object database. Import the three VRML files `vehicle_body_limousine.wrl`, `steeringwheel_limousine.wrl` and `tire_limousine.wrl`, which are supplied in this directory. Use the button Duplicate to duplicate the object **tire_limousine** until four tyre geometries are available in the **Object** list. Afterwards change the names of the four tyres to tire_limousine_fl, tire_limousine_fr, tire_limousine_rl and tire_limousine_rr.

5.1.4 Step 4: Adjust Object Properties : Colour and Position

To adjust the object colours, open the object dialogue. Click the Replace Colour button. A dialog window opens, which shows all colours used for the current object. Click the colour spot you wish to change. A slider window shows the RGB components of the selected colour. By moving the sliders, adjust the colour according to your preference.

To move and rotate an object, move the mouse pointer to that object. Then press the Strg key and click the left mouse button. The selected object is highlighted, and a coordinate system is displayed. For moving the object, press the left mouse button and move the mouse along the axes of the coordinate system. To rotate the object, press the right mouse button and move the mouse along the axes of the coordinate system. Press left and right button together to scale the object.

5.2 Connect Simulation Data

At the moment all 3D objects that form the vehicle are located in the centre of the display area because they are not connected with the simulation data.

5.2.1 Step 1: Initialise Connection

For receiving data from your simulation model set the animation mode of DYNAanimation to **Online local** and activate the button  in the control bar.

5.2.2 Step 2: Refresh Signal List

If the signal names in the signal list of the object dialog are Signal 0 . . . Signal 100, the connection is not up to date. Execute your simulation for a short while to refresh the list. Then the signal list should show the signal names has they have been defined in the Simulink block **DYNAanimation Objects**, see also Object Configuration.

5.2.3 Step 3: Connect Signals

Each dynamic object needs to be connected with a signal of the simulation to receive motion data. To connect the vehicle components with the corresponding signals, you need to select first the object from the **Object** drop-down menu and then the respective signal from the **Signal** drop-down menu. The object names and the respective signal names of each vehicle part (as defined in veDYNA) are listed in the following table.

Vehicle Object	Name of Object in Object List	Name of Signal in Signal List
Vehicle body	*vehicle_body_limousine*	*vehicle*
Steering wheel	*steeringwheel_limousine*	*steeringwheel*
Front left tyre	*tire_limousine_fl*	*tire_fl*
Front right tyre	*tire_limousine_fr*	*tire_fr*
Rear left tyre	*tire_limousine_rl*	*tire_rl*
Rear right tyre	*tire_limousine_rr*	*tire_rr*

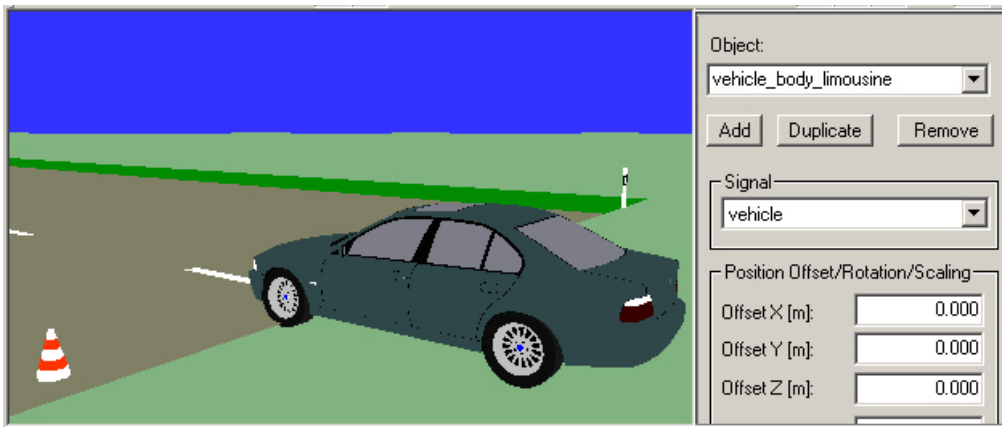


FIGURE 5.3: **Connecting Vehicle Objects with Corresponding Signals**

5.2.4 Step 4: Save Project or Save Animation

The virtual scene (without results) can be saved in a project file. Use the menu option **File** | **Save Project As ...** and provide a new project name (extension .dap). Project files can be re-used.

The complete animation environment (scene and results) can also be saved. Use the menu option **File** | **Save Project As ...** and provide a new file name (extension .die).

5.3 Add Displays

Displays highlight one or more simulation results by means of graphical controls during a running animation. DYNAanimation supports various different display types.

In this example a text field showing the simulation time and a tachometer showing the vehicle speed are added. To add displays to your scene, open the Display Dialog and click the Add button. A grid becomes visible and a display is added.

To add the text field change the **Display Type** to **Text** and type the description you wish to see in the animation into the **Caption** field (here: Time:). Type the unit into the **Caption [unit]** field (here: [sec]). Move the display to the desired position using the mouse. Now connect the display with the required simulation data item which is selected from the signal list. In this example you would select ”Simulation Time” from the **Signal** drop-down menu.

In the same way, add another display of type **Gauge** to display the vehicle speed. Write in the **Caption** field the description of the displayed value, e.g. “Speed [km/h]” and set the minimum and maximum values, e.g. to “0” and “200”, respectively. Select the signal to be diplayed from the signal list. When you have finished the display setting turn off the grid.

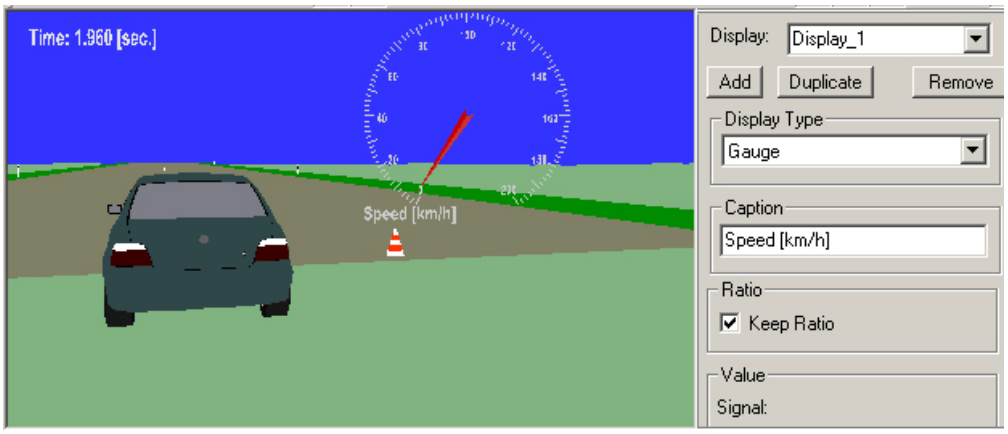


FIGURE 5.4: **Inserting Displays**

5.4 Start Simulation

After creating a virtual world with static and dynamic objects and connecting the dynamic objects with the correct simulation signals, the simulation can be executed.

Execute your simulation model or run a simulation with the current settings in veDYNA. As soon as the simulation is running, DYNAanimation will receive the calculated simulation data and the 3D animation will be started.

5.5 Record and Play Animation

If you wish to store you animation replay, for later use, for comparison of variants, or if you wish to use advanced display types such as **Trace**, it is necessary to record and store the calculated animation data.

5.5.1 Step 1: Record Animation

Before you start the simulation, activate the button of the control bar to record the animation during simulation. This should be your default setting, as it is the precondition to view the animation again or observe certain time intervals more thoroughly after the simulation has finished.

5.5.2 Step 2: Set Record Downsampling

Set the downsampling frequency of data storage during simulation via the drop-down menu of the control bar.

5.5.3 Step 3: Save Animation

To save the animation data in a result file after the simulation has finished, select the menu option **File** | **Save Results As...** or the button and provide a new result file name.

5.5.4 Step 4: Replay Animation

After recording the animation can be played using the right-hand part of the control bar. The animation can be stopped at any time. It is possible to select only a part of the animation to be shown. The animation can also be shown step-wise. The slider allows moving forward and backward in the animation.

5.6 Show Different Simulation Runs in one Animation

For comparing model variants, it is very helpful to visualise the differences in the vehicle motion in one animation. To generate an animation showing the results of two different simulation runs, proceed as follows:

1. Define your model variants.
2. Set up your virtual world in DYNAanimation for a normal simulation. Connect the signals appropriately (see Build a Virtual World).
3. Simulate the first model variant, record the animation data and save the results under a user-defined name, e.g. model_var1.dar
4. Simulate the second model variant, record the animation data and save the results under a user-defined name, e.g. model_var2.dar
5. Activate the menu option **File** | **Merge Results**. A file dialog opens, requesting the selection of the first result file. Then another file dialog opens, requesting the selection of the second result file. The two selected result files are then merged and need to be saved.
6. Confirm to load the merged result file.
7. Add another object to show the second set of simulation results to your scene. The easiest way to do that is to duplicate the object whose motion you want to overlay (in case of a veDYNA simulation that would be the vehicle).
8. Now you have to sets of simulation results, which are distinguished in the signal list by means of numbers in brackets. You have also two objects, but the second one is not yet connected to the simulation results. Connect the second set of results with the second object.
9. On running the animation, the displayed objects will move according to the different simulation results they are connected with.
10. For more than two variants, just add another simulation result to the merged result file and another object to the animation project and proceed as described in steps 4-8.

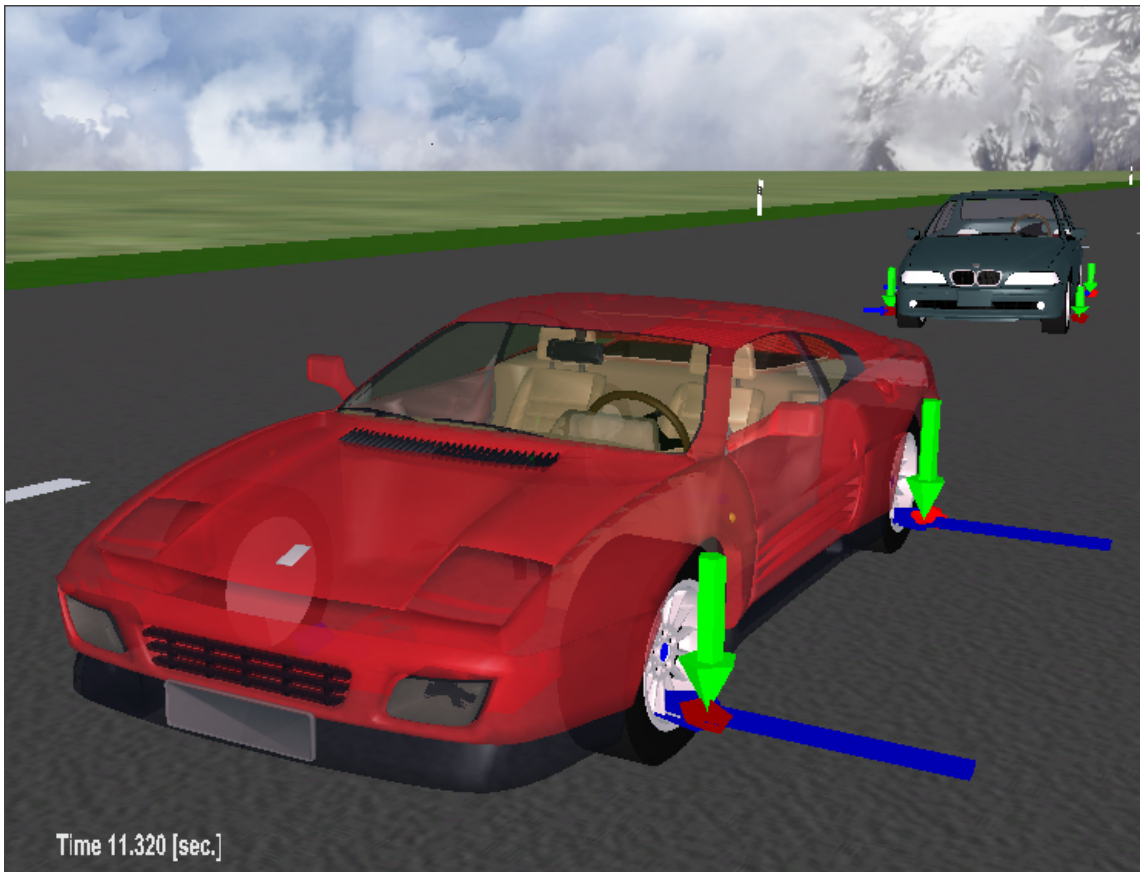


FIGURE 5.5: **Comparison of different simulation runs with model variants**

5.7 Generate Video / Image

In order to present simulation results without running the simulation software, video clips or images can be generated.

5.7.1 Option 1: Generate Video

1. Load a DYNAanimation project and perform a simulation. You need to record the animation.
   or
   Load a project file containing a scene and a stored result file.
2. Activate the menu option **File** | **Save Avi**.
3. Enter a file name in the file dialogue.
4. Another dialogue requests the selection of a video codec (e.g. XviD Mpeg4-Codec or any codec present on your computer). If you wish to play the video on another computer, you should choose a codec which is commonly used. It is also possible to configure the settings of your codec to influence the video quality and the resulting size of the avi file.
5. Press OK to start the generation of the video file. You can watch the video during the export process. It is also possible to stop the export at any time.

5.7.2 Option 2: Save Image

Activate the menu option **File** | **Save Image** and choose a file name. The current frame is grabbed and saved as an image. DYNAanimation supports the image formats TGA and BMP.

6 Remote Control via ActiveX

DYNAanimation can be controlled completely via ActiveX controls. This provides a powerful programming interface for Matlab or any other programming language. Matlab versions 6.5 and higher are supported.

To access the COM-Server interface, enter the following command in the Matlab command window:

```
h=am_start(1);
```

This command opens DYNAanimation, establishes the connection to DYNAanimationn and generates a handle. If DYNAanimation is already open, only the the connection is established. The ActiveX methods can then be accessed by calling `h.`⟨`Method`⟩

The function `h.methods` provides an overview of the installed methods. The following table shows the DYNAanimation functions and the names of the corresponding ActiveX methods.

6.1 Example for Remote Control

The entire animation setup and animation run can be controlled from Matlab. In principle, all actions that are performed directly via the user interface have to be described by means of appropriate functions. This includes creating the scenery, creating objects and displays, and setting view options.

The supplied Matlab procedure `demo_translation.m` demonstrates the usage of the COM-interface of DYNAanimation. We recommend to have a closer look at this procedure and to take it as a starting point for your own remote control applications.

6.2 Function Overview

The following table provides a comprehensive overview of the ActiveX methods. Please note that some methods that manipulate the displays can only be used for a subset of the available displays.

In the table the displays that can be controlled by a method are indicated by capital letters with the following meaning:

`T` Text

`G` Gauge

`VB` Vertical Bar

`HB` Horizontal Bar

`A` Accleration

`G` Gears

`RV` Rear View

`SV` Side View

`C` Cockpit

`L` Logo

`P` Plot

`Tr` Trace

`R` Ruler

is indicated by

ActiveX Method	Short Description
About()	Displays About Dialog
AddCamera(BSTR szCameraName, [out, retval] short *output)	Adds camera after current with Name [string szCameraName], returns camera ID [short ID]
AddDisplay(BSTR szDisplayName, [out, retval] short* output)	Adds display after current with Name [string szDisplayName], returns display ID [short ID]
AddObject(BSTR szObjName, BSTR szFilename, [out, retval] long *output)	Adds object after current with Name [string szObjName], VRML-geometry [string szFilename] returns object ID [long ID]
AdditionalView()	Adds a view window
BlockInput(short iViewID, short onoff)	Toggles reaction to user inputs for view number [short iViewID] (-1 for all) on or off [short onoff]
CascadeViews()	Arranges view windows in a cascade
CenterObject(short iObjID)	Moves camera to focus on object number [int iObjID]
ChangeObject(long iObjID, BSTR szFilename)	Changes object's [long iObjID] geometry into [string szFilename]
ClearMotionBlurBuffer()	Clears motion blur buffer of previous frames
CloseProject()	Closes current project
DeleteCamera(short iCamID)	Removes camera [short iCamID]
DeleteDisplay(short iDispID)	Removes display [short iDispID]
DeleteObject(short iObjID)	Removes object [short iObjID]
DuplicateCamera(short iCamID)	Duplicates Camera [short iCamID]
DuplicateDisplay(short iDispID)	Duplicates Display [short iDispID] and enters *MoveDisplays* mode
DuplicateObject(short iObjID, [out, retval] long *output)	Duplicates Object [short iObjID] and returns [long ID] of the clone
Exit()	Closes DYNAAnimation
FlushOpenGL(short iViewID)	Redraws scene in View [short iViewID]
FlushSignals()	Applies Signal buffer
GetAntialiasing([out, retval] short *output)	Gets status of *Use Antialiasing*-feature [short onoff]
Camera	
GetCameraAngle(int iCamID, [out, retval] double *output)	Gets angle [double angle] of camera [int iCamID]
GetCameraID([out, retval] short *output)	Gets [short ID] of active camera
GetCameraIDbyName(BSTR szID, [out, retval] short *output)	Gets [short ID] of camera named [string name]
GetCameraList([out, retval] VARIANT *output)	Gets names [string_array CameraNames] of existing cameras
GetCameraListID([out, retval] VARIANT *output)	Gets IDs [int_array CameraIDs] of existing cameras
GetCameraName(short iCamID, [out, retval] BSTR *output)	Gets name [string CameraName] of camera [short ID]
GetCameraOffset([in] int iCamID, [out, retval] VARIANT *output)	Gets offset vector [double (x,y,z)] of camera [int iCamID]
GetCameraRotation(int iCamID, [out, retval] VARIANT *output)	Gets rotation vector [double (x,y,z)] of camera [int iCamID]
GetCameraSettings([out, retval] short *output)	Gets visibility status of *Camera Settings*-dialog [short onoff]
GetCameraSignalID(int iCamID, [out, retval] short *output)	Gets signal [short SignalNumber] camera [int iCamID] is connected to

continued on the next page

continued from previous page

GetCameraVisible(int iCamID, [out, retval] double *output)	Gets max visibility distance [double Visibility] of camera [int iCamID]
GetCameraZRotate(int iCamID, [out, retval] short *output)	Gets status of *ZRotate*-feature [short onoff] of camera [int iCamID]
GetComplexBackground([out, retval] short *output)	Gets status of *Use Complex Background*-feature short onoff]
GetConnectToSignals([out, retval] short *output)	Gets status of signal connection short onoff]
GetConnectionMode([out, retval] short *output)	Gets mode of signal connection [short mode] (0=offline, 1=online local, >1=extern)
GetConnectionModeList([out, retval] VARIANT *output)	Gets List of available connection TYPES [string_array Types]
GetConnectionType([out, retval] short *output)	Gets type of signal connection [short type] (0=Net,1=Bus)
GetConnectionTypeList([out, retval] VARIANT *output)	Gets list of available connection MODES [string_array Modes]
GetControlToolbar([out, retval] short *output)	Gets status of Control Toolbar [short onoff]
Display	
GetDisplay([out, retval] VARIANT *output)	Gets ID and name of selected display [multi_array ID_name]
GetDisplayCaption(short iDispID, [out, retval] BSTR *output) **T, G, VB, HB**	Gets Caption [string caption] of display [short iDispID]
GetDisplayCaptionUnit(short iDispID, [out, retval] BSTR *output) **T**	Gets Unit [string unit] of display [short iDispID] value (see list of display functions!)
GetDisplayColor(short iDispID, [out, retval] VARIANT *output) **T, G**	Gets color [double_array (r,g,b)] of display [short iDispID]
GetDisplayFile(short iDispID, [out, retval] BSTR *output) **L, P**	Gets input [string filename] filename of display [short iDispID]
GetDisplayIDbyName(BSTR szID, [out, retval] short *output)	Gets ID of display [BSTR szID]
GetDisplayLegend(short iDispID, [out, retval] short *output) **P**	Gets status of *Legend* feature [short onoff] of plot display [short iDispID]
GetDisplayLength(short iDispID, [out, retval] double *output) **R**	Gets length [double length] of ruler display [short iDispID]
GetDisplayLimitX(short iDispID, [out, retval] double *output) **P**	Gets X-Axis limit [double limit] of plot display [short iDispID]
GetDisplayList([out, retval] VARIANT *output)	Gets names [string_array DisplayNames] of all displays in all views
GetDisplayListID([out, retval] VARIANT *output)	Gets IDs [int_array IDs] of all displays in all views
GetDisplayListViewID([out, retval] VARIANT *output)	Gets list of view IDs [int_array Views] each display is assigned to
GetDisplayMax_1(short iDispID, [out, retval] double *output) **G, VB, HB**	Gets maximum for 1st value [double max] of display [short iDispID]
GetDisplayMin_1(short iDispID, [out, retval] double *output) **G, VB, HB**	Gets minimum for 1st value [double min] of display [short iDispID] (OR in case of *Ruler*-display: X-Rotation!)
GetDisplayName(short iDispID, [out, retval] BSTR *output)	Gets name [string name] of display [short iDispID]
GetDisplayPosition(short iDispID, [out, retval] VARIANT *output)	Gets position [double (x,y)] of display [short iDispID], screen coordinates are -1<x<1 and -1<y<1

continued on the next page

continued from previous page

GetDisplayPrecision_1(short iDispID, [out, retval] long *output) **T**	Gets precision for 1st value [double precision] of display [short iDispID]
GetDisplayRatio(short iDispID, [out, retval] short *output) all except **Tr,R**	Gets status of *Keep Aspect Ratio*-feature [short onoff] of display [short iDispID]
GetDisplayScale(short iDispID, [out, retval] VARIANT *output)	Gets scale factors [double_array (x,y)] for display [short iDispID], overridden by *Keep Aspect Ratio*
GetDisplayScaleX(short iDispID, [out, retval] double *output) **P**	Gets X-axis scale factor [double_array (x,y)] for plot display [short iDispID]
GetDisplayScaleY(short iDispID, [out, retval] double *output) **P**	Gets Y-axis scale factor [double_array (x,y)] for plot display [short iDispID]
GetDisplaySettings([out, retval] short *output)	Gets visibility status of *Display Settings*-dialog [short onoff]
GetDisplayShadow(short iDispID, [out, retval] short *output) **P**	Gets status of *shadow*-feature [short onoff] for plot display [short iDispID]
GetDisplaySignal_1(short iDispID, [out, retval] short *output) all except **L**	Gets number of 1st input signal [short number] for display [short iDispID]
GetDisplaySignal_2(short iDispID, [out, retval] short *output) **A, C, P, R**	Gets number of 2nd input signal [short number] for display [short iDispID]
GetDisplaySignal_3(short iDispID, [out, retval] short *output) **A,C**	Gets number of 3rd input signal [short number] for display [short iDispID]
GetDisplaySteps(short iDispID, [out, retval] double *output) **R**	Gets number of steps [double number] for ruler display [short iDispID]
GetDisplayTransparency(short iDispID, [out, retval] float *output) **T,G,L**	Gets value for *transparency*-feature [double transparency] for display [short iDispID]
GetDisplayType(short iDispID, [out, retval] BSTR *output)	Gets type description [string type] for display [short iDispID]
GetDisplayViewID(short iDispID, [out, retval] short *output)	Gets view number [short number] for display [short iDispID]
GetDisplayWidth_1(short iDispID, [out, retval] long *output) **T**	Gets max width of numbers [long width] in text display [short iDispID]
GetEndFrame([out, retval] long *output)	Gets the number of the last time step
GetFog([out, retval] short *output)	Gets status of *Use Fog*-feature [short onoff]
GetFullscreenMode([out, retval] short *output)	Gets status of *Fullscreen*-feature [short onoff]
GetGameController([out, retval] short *output)	Gets status of *Enable Game Controller (Joystick)*-feature [short onoff]
GetGlow([out, retval] short *output)	Gets status of *Use Glow*-feature [short onoff]
GetLicensePath([out, retval] BSTR *output)	Gets path of currently used license file [string path]
GetLighting([out, retval] short *output)	Gets status of *Use Lighting*-feature [short onoff]
GetLoopResults([out, retval] short *output)	Gets status of *Loop Result Animation*-feature [short onoff]
GetMainToolbar([out, retval] short *output)	Gets status of Main Toolbar [short onoff]
GetMatlabPath([out, retval] BSTR *output)	Gets path of active Matlab-Version [string path]
GetMaxFrame([out, retval] long *output)	Gets total animation length in frames [long frame]
GetMotionBlur([out, retval] short *output)	Gets status of *Use Motion Blur*-feature [short onoff]
GetObject3DSettings([out, retval] short *output)	Gets visibility status of *Object Settings*-dialog [short onoff]

continued on the next page

continued from previous page

GetObjectAutoReload(long iObjID, [out, retval] short *output)	Gets status of *Auto Reload*-feature [short onoff] of object [long iObjID]
GetObjectCenter(long iObjID, [out, retval] short *output)	Gets status of *Center*-feature [short onoff] of object [long iObjID]
GetObjectChrome(long iObjID, [out, retval] short *output)	Gets status of *Chrome*-feature [short onoff] of object [long iObjID]
GetObjectColor(long iObjID, long iColorID, [out, retval] VARIANT *output)	Gets color [double (r,g,b)] number [int number] of object [long iObjID]
GetObjectColorList(long iObjID, [out, retval] VARIANT *output)	Gets List of all colors [double_array] of object [long iObjID]
GetObjectContour(long iObjID, [out, retval] short *output)	Gets status of *Contour*-feature [short onoff] of object [long iObjID]
GetObjectDoubleSided(long iObjID, [out, retval] short *output)	Gets status of *Double Sided*-feature [short onoff] of object [long iObjID]
GetObjectID([out, retval] long *output)	Gets ID [long ID] of active object
GetObjectIDbyName(BSTR szID, [out, retval] long *output)	Gets object ID of object
GetObjectList([out, retval] VARIANT *output)	Gets list of object names [string_array names] in the scene
GetObjectListID([out, retval] VARIANT *output)	Gets list of object IDs [int_array IDs] in the scene
GetObjectName(long iObjID, [out, retval] BSTR *output)	Gets name [string name] of object [long iObjID]
GetObjectOffset(long iObjID, [out, retval] VARIANT *output)	Gets Offset [int_array (x,y,z)] of object [long iObjID]
GetObjectRotation(long iObjID, [out, retval] VARIANT *output)	Gets Rotation [int_array (x,y,z)] of object [long iObjID]
GetObjectScale(long iObjID, [out, retval] VARIANT *output)	Gets Scale factors [int_array (x,y,z)] of object [long iObjID]
GetObjectSignalID(long iObjID, [out, retval] long *output)	Gets Signal number [long number] object [long iObjID] is linked to
GetObjectSignalList([out, retval] VARIANT*output)	Gets names of all object signals [string_array signals] in the scene
GetObjectSignalListID([out, retval] VARIANT *output)	Gets numbers of all object signals [int_array signals] in the scene
GetObjectTransparency(long iObjID, [out, retval] double *output)	Gets value of *Transparency*-feature [double transparency] of object [long iObjID]
GetObjectVisible(long iObjID, [out, retval] short *output)	Gets state of *Visible*-feature [short onoff] of object [long iObjID]
GetProductKey([out, retval] BSTR *output)	Gets Product Key [string key]
GetRealtimeMode([out, retval] short *output)	Gets state of *Real Time Mode*-feaure [short onoff]
GetRecordDownsampling([out, retval] long *output)	Gets number [long downsample] of skipped frames when recording an animation
GetRecordResults([out, retval] short *output)	Gets recording state [short onoff]
GetRunResults([out, retval] short *output)	Gets playing state [short onoff]
GetShadows([out, retval] short *output)	Gets state of *Use Shadows*-feature [short onoff]

continued on the next page

continued from previous page

GetSignalValue(short iSignalID, [out, retval] float *output)	Gets current value [float value] of signal [short iSignalID]
GetSmoothShading([out, retval] short *output)	Gets state of *Use Smooth Shading*-feature [short onoff]
GetSoftShadows([out, retval] short *output)	Gets state of *Use Soft Shadows*-feature [short onoff]
GetStartFrame([out, retval] long *output)	Gets number of Start Frame [long number]
GetStatusbar([out, retval] short *output)	Gets state of Statusbar [short onoff]
GetStepMode([out, retval] short *output)	Gets state of *Step Mode*-feature [short onoff]
GetSynchronisationMode([out, retval] short *output)	Gets Synchronisation state [short onoff]
GetTextures([out, retval] short *output)	Gets state of *Use Textures*-feature [short onoff]
GetUserDefinedBackground([out, retval] short *output)	Gets state of *Use UserdefinedBackground*-feature [short onoff]
GetVideoCompressionList([out, retval] VARIANT *output)	Gets list of available video codecs [string_array codecs]
GetViewMaximized(short iViewID, [out, retval] short *output)	Gets maximized state [short onoff] of view [short iViewID]
GetWireFrame([out, retval] short *output)	Gets state of *Use Wireframe*-feature [short onoff]
GetWorkPath([out, retval] BSTR *output)	Gets path of working directory [string path]
HoldOpenGL(short iViewID, short iHold)	Freezes [short iHold] Graphical output of view [short iViewID]
Initialize()	Initializes possibly running instance of DYNAAnimation
IsOpen([out, retval] short *output)	Detects if DYNAAnimation is running [short onoff]
LoadProject(BSTR szProjectName)	Loads project file [string filename]
LoadResults(BSTR szResultName)	Loads result file [string filename]
MergeResults(BSTR szResultName1, BSTR szResultName2, BSTR szResultNameMerge)	Merges two result files [string szResultName1, string filename_source2] into one [BSTR szResultNameMerge]
NewDefaultProject()	Opens default project file
NewProject(BSTR szProjectName)	Opens a new project [string szProjectName]
NextCamera()	Chooses next camera from list
PreviousCamera()	Chooses previous camera from list
RefreshAllViews()	Refreshes all views
ReloadAllGeometries()	Reloads all geometries, (discarding *replace colors* settings)
ResetDisplayImage(short iDispID)	restes default image of display type ”logo”
ResetResults(short onoff)	Resets animation to start frame [short onoff]
SaveAvi(BSTR szAviName, BSTR szCodec, short iViewID)	Writes animation to AVI-file [string szAviName, string szCodec, short iViewID]
SaveImage(BSTR szImageName, short iViewID)	Saves screenshot to bitmap [string szImageName, short iViewID]
SaveProject()	Saves current project file
SaveProjectAs(BSTR szProjectName)	Saves current project file as [string szProjectName]
SaveProjectAsDefault()	Saves current project as default project
SaveResultsAs(BSTR szResultsName)	Saves recorded results as [string szResultsName]
SetAlwaysOnTop(short onoff)	Toggles *Always on Top*-feature [short onoff]
SetAntialiasing(short onoff)	Toggles *Use Antialiasing*-feature [short onoff]
SetCamera(short iCamID)	Chooses camera [short iCamID] as active
SetCameraAngle(short iCamID, float fAngle)	Sets field of view angle [short iCamID] of camera [float fAngle]

continued on the next page

continued from previous page

SetCameraOffset(short iCamID, float x, float y, float z)	Sets camera offset [short iCamID, float x, float y, float z]
SetCameraRotation(short iCamID, float x, float y, float z)	Sets camera rotation [short iCamID, float x, float y, float z]
SetCameraSettings(short onoff)	Toggles visibility of *Camera Settings*-dialog [short onoff]
SetCameraSignal(short iCamID, short iSignalID)	Links camera [short iCamID] to an object signal [short iSignalID]
SetCameraVisibility(short iCamID, float fDistance)	Sets maximum viewing distance [float fDistance] of camera [short iCamID]
SetCameraZRotate(short iCamID, short onoff)	Toggles *Z-Rotate*-feature [short onoff] of camera [short iCamID]
SetComplexBackground(short onoff)	Toggles *Use Complex background*-feature [short onoff]
SetConnectToSignals(short onoff)	Toggles *Connect to signals*-option [short onoff]
SetConnectionMode(short iModeID)	Sets mode of connection [short iModeID] (0=offline, 1=online local, etc.)
SetConnectionType(short iTypeD)	Sets type of connection [short iTypeD] (0=Net, 1=Bus)
SetControlToolbar(short onoff)	Toggles visibility status of control toolbar [short onoff]
SetCurrentFrame(long iFramePos)	Sets animation to frame [long iFramePos]
SetDisplay(short iDispID)	Chooses current display [short iDispID]
SetDisplayCaption(short iDispID, BSTR szCaption) **T,G,VB,HB**	Sets caption [string szCaption of display [short iDispID]
SetDisplayCaptionUnit(short iDispID, BSTR szCaptionUnit) **T**	Sets caption unit [string szCaptionUnit] of display [short iDispID]
SetDisplayColor(short iDispID, double fR, double fG, double fB) **T,G**	Sets color [double fR, double fG, double fB] of display [short iDispID]
SetDisplayFile(short iDispID, BSTR szFile) **L,P**	Sets input file [string szFile] of display [short iDispID]
SetDisplayLegend(short iDispID, short onoff) **P**	Toggles legend visibility [short onoff] of display [short iDispID]
SetDisplayLength(short iDispID, double fValue) **R**	Sets length [float fValue] of display [short iDispID]
SetDisplayLimitX(short iDispID, double fValue) **P**	Sets X-axis limit [float fValue] of display [short iDispID]
SetDisplayMax_1(short iDispID, double iMax) **G, VB, HB**	Sets maximum value [float iMax] of display [short iDispID]
SetDisplayMin_1(short iDispID, double iMin) **G, VB, HB**	Sets minimum value [float iMin] of display [short iDispID]
SetDisplayPosition(short iDispID, double iX, double iY) all except **L**	Sets screen position [double iX, double iY] of display [short iDispID]
SetDisplayPrecision_1(short iDispID, long iPrecision) **T**	Sets precision [long iPrecision] of display [short iDispID]
SetDisplayRatio(short iDispID, short onoff) all except **Tr,R**	Toggles *Keep aspect ratio*-feature [short onoff] of display [short iDispID]
SetDisplayScale(short iDispID, double iX, double iY)	Sets on-screen scale factors [float x, float y] for display [short iDispID]
SetDisplayScaleX(short iDispID, double fValue) **P**	Sets x-axis scale factor [double fValue] for display [short iDispID]
SetDisplayScaleY(short iDispID, double fValue) **P**	Sets y-axis scale factor [double fValue] for display [short iDispID]
SetDisplaySettings(short onoff)	Toggles visibility of *Display Settings*-dialog [short onoff]

continued on the next page

continued from previous page

SetDisplayShadow(short iDispID, short onoff) **P**	Toggles *Shadow*-feature [short onoff] of display [short iDispID]
SetDisplaySignal_1(short iDispID, short iSignalID) all except **L**	Chooses 1st input signal [short iSignalID] for display [short iDispID]
SetDisplaySignal_2(short iDispID, short iSignalID) **A,C,P,R**	Chooses 2nd input signal [short iSignalID] for display [short iDispID]
SetDisplaySignal_3(short iDispID, short iSignalID) **A,C**	Chooses 3rd input signal [short iSignalID] for display [short iDispID]
SetDisplaySteps(short iDispID, double fValue) **R**	Sets number of steps [double fValue] for display [short iDispID]
SetDisplayTransparency(short iDispID, float fValue) **T, G,L**	Sets value of *Transparency*-feature [double fValue] of display [short iDispID]
SetDisplayType(short iDispID, BSTR szDispType)	Sets type [string szDispType] of display [short iDispID]
SetDisplayWidth_1(short iDispID, long iWidth) **T**	Sets width [long iWidth] of display [short iDispID] value
SetEndFrame(long iFramePos)	Sets *Last frame* marker at frame [long iFramePos]
SetFog(short onoff)	Toggles *Use Fog*-feature [short onoff]
SetFullscreenMode(short onoff)	Toggles *Fullscreen Mode*-option [short onoff]
SetGameControler(short onoff)	Toggles *Use Game Controller*-feature [short onoff]
SetGlow(short onoff)	Toggles *Use Glow*-feature [short onoff]
SetLighting(short onoff)	Toggles *Use Lighting*-feature [short onoff]
SetLoopResults(short onoff)	Toggles *Infinite Loop results*-option [short onoff]
SetMainToolbar(short onoff)	Toggles Main Toolbar visibility [short onoff]
SetMotionBlur(short onoff)	Toggles *Use Motion Blur*-feature [short onoff]
SetObject(short iObjID)	Chooses current object [short iObjID]
SetObject3dSettings(short onoff)	Toggles *Object Settings*-dialog [short onoff]
SetObjectCenter(short iObjID, short onoff)	Toggles *Center*-option [short onoff] for object [short iObjID]
SetObjectChrome(short iObjID, short onoff)	Toggles *Chrome*-option [short onoff] for object [short iObjID]
SetObjectColor(short iObjID, short iColorID, float r, float g, float b)	Sets color [int index] value [float r, float g, float b] for object [short iObjID]
SetObjectContour(short iObjID, short onoff)	Toggles *Contour*-option [short onoff] for object [short iObjID]
SetObjectDoubleSided(short iObjID, short onoff)	Toggles *Double Sided*-option [short onoff] for object [short iObjID]
SetObjectName(short iObjID, BSTR szName)	Changes name [string szName for object [short iObjID]
SetObjectOffset(short iObjID, float x, float y, float z)	Sets Offset [float x, float y, float z] for object [short iObjID]
SetObjectRotation(short iObjID, float x, float y, float z)	Sets Rotation [float x, float y, float z] for object [short iObjID]
SetObjectScale(short iObjID, float x, float y, float z)	Sets Scale factors [float x, float y, float z] for object [short iObjID]
SetObjectSignal(short iObjID, short iSignalID)	Links Object [short iObjID] to object signal [short iSignalID]
SetObjectTransparence(short iObjID, float fValue)	Sets value for *Transparency*-feature [float fValue] for object [short iObjID]

continued on the next page

continued from previous page

SetObjectVisible(short iObjID, short onoff)	Toggles visibility [short onoff] of object [short iObjID]
SetRealtimeMode(short onoff)	Toggles *Realtime*-mode [short onoff]
SetRecordDownsampling(short iRate)	Sets value for *Record Downsampling* feature [short iRate]
SetRecordResults(short onoff)	Toggles *Record Results*-mode [short onoff]
SetRunResults(short onoff)	Toggles *Run Results*-mode [short onoff]
SetShadows(short onoff)	Toggles *Use Shadows*-feature [short onoff]
SetSmoothShading(short onoff)	Toggles *Use Smooth Shading*-feature [short onoff]
SetSoftShadows(short onoff)	Toggles *Use Soft Shadows*-feature [short onoff]
SetStartFrame(long iFramePos)	Sets *First frame* marker at frame [long iFramePos]
SetStatusBar(short onoff)	Toggles visibility of status bar [short onoff]
SetStepMode(short onoff)	Toggles *Step*-Mode [short onoff]
SetSynchronisationMode(short onoff)	Toggles *Synchronisation*-Mode [short onoff]
SetTextures(short onoff)	Toggles *Use Textures*-Feature [short onoff]
SetUserDefinedBackground(short onoff)	Toggles *Use User-Defined Background*-feature [short onoff]
SetViewMaximized(short iViewID, short onoff)	Toggles *Maximized*-mode [short onoff] of view [short iViewID]
SetWindowSize(short x, short y)	Sets Window Size [short x, short y]
SetWireFrame(short onoff)	Toggles *Use Wireframe*-feature[short onoff]
SetWorkPath(BSTR szLabel)	Sets Work directory [string szLabel
ShowDisplays(short onoff)	Toggles visibility of all displays [short onoff]
ShowObjects(short onoff)	Toggles visibility of all objects[short onoff]
StartDYNAanimation()	Opens DYNAANinmation
StartIntegrated(BSTR szProdKey, BSTR szWork, BSTR szLic, BSTR szMatlab)	Starts DYNAAnimation as plugin for vedyna, example: StartIntegrated('vdy', 'Work-Directory', 'License-Path', 'Matlab-Path')
TileViewsHorizontal()	Arrange views with horizontal tiling
TileViewsVertical()	Arrange views with vertical tiling
WriteDisplaySignal(long iSignalID, short bOnce, float Value)	Writes a value [float Value] to display signal [long iSignalID] once [short bOnce=1] or recorded [short bOnce=0]
WriteObjectSignal(long iSignalID, short bOnce, VARIANT Values)	Writes a position/rotation vector [float_array Values] to signal [long iSignalID] once [short bOnce=1] or recorded [short bOnce=0]
WriteSignalName(short iSignalID, short iType, BSTR szName)	Gives the signal [short iSignalID, short iType(0=object,1=display)] a name [string szName]
SetObjectAutoReload(short iObjID, short onoff)	Toggles *Auto Reload*-option [short onoff] in object [short iObjID]

Imprint

TESIS DYNAware
Technische Simulation Dynamischer Systeme GmbH

Documentation

Baierbrunner Straße 15

D-81379 Munich

Germany

Telephone: +49 89 74 73 77-0

Telefax: +49 89 74 73 77-99

E-mail: info@tesis.de

Web: www.tesis.de/dynaware