FAQ for Virtual Test Drive

Preface / Vorwort

May we kindly ask you to provide your entries in English language. This will facilitate the use of this Wiki by our international users. Thank you!

Wir bitten darum, Einträge auf dieser Seite vorzugsweise in englischer Sprache zu verfassen, da der Markt für VTD zunehmend außerhalb des deutschsprachigen Raums liegt. Gerne übernimmt VIRES die Übersetzung neu eingestellter Einträge. Vielen Dank!

Preface / Vorwort
Hardware and Operating System Configuration
How can I install VTD for a multi-user environment?
How can I create a Bug Report?
General Definitions
GUI/IOS
- GUI does not start and prints: FATAL (VTGUI): "ComHandler::CTOR: could not get own host address"
- How can I change the font size of the GUI, so that all elements are readable on my machine?
- Operation via GUI or SCP?

ImageGenerator
TaskControl
- How do I maximize the performance of a SiL simulation?
  - VTD 2.x:
  - VTD 1.x:
- How do I retrieve player configuration details (i.e. names, IDs, categories)
- Video Streaming Issues
  - Live video-streaming via TaskControl (TCP/IP network port)
  - Video/Image transfer from the ImageGenerator via Shared Memory (SHM)
    - Low-Dynamic-Range (LDR) operation
    - High-Dynamic-Range (HDR) operation
  - How to read image data from the ImageGenerator's Shared Memory (SHM)
    - General Description
    - Code Example - General
    - Full Example: explicit triggering of VTD and video readback via shared memory, HDR
- Replay ignores my GUI camera settings
- How can I trigger the simulation (and control real-time behavior)?
  - Internal Triggering
  - External Triggering
    - Triggering via ImageGenerator
    - Triggering via RDB
    - Triggering via SCP
  - Overview of trigger modes and image generation modes
  - Using a moduleManager Plug-in for Explicit Trigger Intervals
  - Internal Loop of the TaskControl
- How can I synchronize the simulation?
  - Basic Synchronization to Vehicle Dynamics
  - Synchronization to 3rd Party Sources via RDB
    - Register for Synchronization
    - Provide Synchronization Signal
    - Un-register from Synchronization
  - Synchronization to 3rd Party Sources via Shared Memory
    - Register for Synchronization
    - Provide Synchronization Signal
    - Un-register from Synchronization
- Driving with joysticks and game wheels
  - Settings for the Logitech G27 game wheel
- Performance Monitoring

ModuleManager
Sound
- How can I play an arbitrary sound sample via SCP?
- How can I configure the sound to use an interface other than eth0?
- How can I configure the sound to use TCP instead of UDP?
- How can I use own sound samples for my vehicle's sound?
  - Recording
  - Registration

Databases
- How can I add detailed surface information to an existing road network?
  - Visual Database
  - Logical Database
- How does VTD support Geo-Referencing
  - Projection Information
    - OGC-WKT Interpretation for Transverse Mercator
    - Direct Definition of the proj4 String
  - Conversion Methods
    - TaskControl
    - ModuleManager

RoadDesigner (ROD)
Traffic and Scenario
- Driving on the "left side"
- A vehicle starts automatically although it shouldn't or vice versa
- A lane change is not executed completely and the driver steers back to the original lane.
- How can I define a steering trajectory for a player that also uses a waypoint path (route)?
- Is there a specification of the scenario description?
- Configuration of entities (3d-models of players and objects)
  - Adding a trailer to a vehicle
  - How can I add new vehicle models to my VTD configuration?

- Description for versions VTD.1.4 and higher
- Description for versions VTD.1.2 to VTD.1.3
- Description for versions prior VTD.1.2
- How do I obtain the model number for a given vehicle model?
- How can I add new objects to my VTD configuration?
- How can I convert an object's 3d-model so that it will be loaded by v-IG?
- How can I place a general 3d-object in my scenario at run-time?
- How can I place a custom sign in my scenario at run-time?
- Driver Behavior
- How can I request control commands from a VTD driver for my own vehicle?
- Requesting Pedal Positions
- Selective Control
- Steering Target Information
- How are the normalized driver parameters linked to physical values?
- How can I realize an "automated driving" simulation?
- Adapt driver behavior to vehicle type
- What are the influencing factors for the desired acceleration?
- Lane Change Model
- General
- Is there a relationship between the distanceKeeping coefficient and the realization of autonomous lane changes on multi-lane roads?
- How does urgeToOvertake influence overtaking behavior?
- Configuration of "track-controlled" driver mode
- How to produce a sudden traffic jam
- How can I show a perspective map in the ScenarioEditor?
- How do I configure a custom trajectory for a player?
- Display additional road information in ScenarioEditor
- How can I debug my trajectory or how can I see my steering path?
- How do I configure a looping path for a player?
- Where is the default eyepoint location of a player?
- Traffic Lights and Traffic Signs
- How are the state masks defined?
- Collision of Players
- How can I create Triggers/Actions executed by each player of a group reaching the trigger condition
- How do I use an external trigger
- Scenario Editor
- Launching the ScenarioEditor from command line

Data and Video Recording

- Data Recording
- How can I create a data recording?
- Replaying a recording
- Status Display
- Recording to a buffer instead of a file
- I have a data recording with pedestrians but I don't have a pedestrian license. What will happen?
- Video Recording
- Live video recording
- Recording via GUI
- Alternative Command Sequences
- "Offline" video recording
- Are there any restrictions in connection with the video recording?

Vehicle Dynamics

- VIRES Vehicle Dynamics in the ModuleManager
- VIRES Simple Dynamics
- VIRES Complex Dynamics
- Changing parameters
- Implementation Overview
- Truck Cabin Movement
- Using the complex dynamics
- Using RDB
- Configuration
- Data computed within the plug-in
- How do I connect my vehicle dynamics (and other components) to VTD?
- How to run a vehicle dynamics which is connected via RDB only
- 3rd Party Vehicle Dynamics: veDYNA
- Installation
- Basic Steps
- Access Rights
- Project / Setup-specific settings
- Vehicle Name
- VTD configuration
- Scenario configuration
- Operation
- Legacy Operation
- Troubleshooting: veDYNA refuses to start
- Vehicle Characteristics using the RDB version of veDYNA
- 3rd Party Vehicle Dynamics: others
- How can I get contact point information for my vehicle dynamics?
- Contact points via TaskControl (low frequency)
- Contact points via OdrGateway (high frequency)
- General
- Installation
- Process Configuration
- RDB Configuration
- Communication
- Operation in Relative Position Mode
- Offroad Handling
- Manual Control of the Vehicle

Pedestrians
- DI-Guy Pedestrians
  - Installation and Configuration
  - Additional remarks for VTD installations prior to VTD 1.4.x
    - Configuration of the TaskControl
    - Plug-in of the Image Generator
  - Operation
    - Character Viewer of Scenario Editor
- Vires Pedestrians
  - Database creation
  - ImageGenerator configuration

Common Questions
- Which configuration data is used by VTD?
- What data do I have to backup?
  - Setups
  - Projects
  - ROD databases
  - General Note
- Which SW-packages do I need for video generation?
- Which data can I expect from VTD?
- How can I locate a project outside the VTD tree?
- How can I document my own source code with Doxygen if I'm including VTD header files (viRDBIcd.h or scpIcd.h)?
- How do I format a command for immediate mode of the SCP Generator?
- How do I define Umlaute in SCP commands?
- Communication
  - How to configure the communication so that only loopback device is used?
  - How to use a specific Ethernet device for communication?
  - My UDP communication does not work
- How can I influence the information about traffic signs?

Task and Environment Configuration
- How can I customize environment variables used in VTD?
- How can I initialize VTD for each simulation?
- How do I integrate a 3rd party component (here: ADTF) into the VTD mechanisms (load components, start etc.)?
- How can I assign a task to a dedicated CPU?
- How can I run VTD on a remote PC?
  - Running VTD via remote login
  - Running individual tasks on a remote computer
- How can I configure multiple simulators for co-operation?
  - Step 1: Preparing the installation(s)
  - Step 2: Configuring the TaskControl
  - Step 3: Configuring the Environment
  - Step 4: Configuring the Image Generator
  - Step 5: Creating a scenario
  - Step 6: Preparing the Simulators
  - Step 7: Starting the Simulation
  - Step 8: Having fun

Performance
- My system gets stuck when I run imageGenerator, scenarioEditor and other outputs on the same display.

Runtime Data Bus (RDB)

Cockpit IOS (CIOS)

Debugging
- How can I create a core dump?
- How can I identify the reason of a crash?

Hardware and Operating System Configuration

see VTD Configuration Issues.

How can I install VTD for a multi-user environment?

This issue has been moved to the following location: Multi-User Installation

How can I create a Bug Report?

Starting with VTD 1.4.1, it is possible to create a bug report which will include most of the relevant information that might be needed to resolve an issue. When reporting an issue, please attach the file generated by the bug report tool to your ticket. Here's how to generate the file:

- Open a shell
- cd into the directory Script VTD/Runtime/Tools/Bugreport
- execute `./vtdBugreport.sh`
- after running the script, you will see a line like `********** Please send the file "/tmp/vtdBugreport.20141004_1129.tgz" to VIRES.`
- send the indicated file to VIRES or attach it to your ticket

General Definitions

If you want to know more about co-ordinate systems etc. please go to the General Definitions page.

GUI/IOS

GUI = Graphical User Interface
IOS = Instructor Operating Station

Here: both terms mean the same software: VtGui

GUI does not start and prints: FATAL (VTGUI): "ComHandler::CTOR: could not get own host address"

The problem ist related to your network configuration. In order to connect to the taskcontrol the IP address of the taskcontrol host has to be retrieved. This is done in the following way:

- By default localhost is assumed as the hostname.
- The default can be overwritten with the environment variable TASKCONTROL_HOST.
- The GUI retrieves the address by the system call gethostbyname(MyTaskcontrolHost).
- If the hostname can not be resolved (i.e. "ping MyTaskcontrolHost" is not successful) the GUI gives up printing the error message.

Solution:

- Make sure, the hostname can be resolved either by DNS, DHCP or simply by writing the IP to the hosts file.
- Check with ping hostname

```
sherman.site|wuni>6: hostname
 sherman.site
```

```
sherman.site|wuni>7: ping sherman
 PING sherman.site (192.168.100.90) 56(84) bytes of data.
 64 bytes from sherman.site (192.168.100.90): icmp_seq=1 ttl=64 time=0.027 ms
```

How can I change the font size of the GUI, so that all elements are readable on my machine?

The appearance of the GUI may vary with the availability of fonts on your machine, with screen resolution etc. On some machines, the default font may be too large so that some elements of the GUI cannot be operated comfortably. In this case, you may have to adapt the font size:

1. In the GUI, select the pop-up window *Edit->Preferences* from the main menu bar
2. Open the tab *Parameters*
3. Press the button *Select Font*
4. Choose the appropriate font from the ones available on your machine

Operation via GUI or SCP?

The command interface between the GUI and the other elements of VTD is the *Simulation Control Protocol (SCP)*. Using this protocol and connecting via the SCP port to the TaskControl, other components may also take over control of the simulation.

If you control parts of the simulation by your own components and have the GUI running concurrently, problems may arise since the GUI will not be able to reach all of its internal states as would be the case if it operated as the only control means.

Examples:

- setting the name of a recording via GUI and starting the recording via SCP will fail. In this case, provide also the name of the recording from your application
- setting a new camera position via SCP will not be recognized by the GUI. In the application, the camera position will change, but the GUI will still indicate its own active camera settings

ImageGenerator

Some issues involving the IG have accumulated over the past; therefore, the IG FAQs are found in a separate file

VTD FAQ ImageGenerator

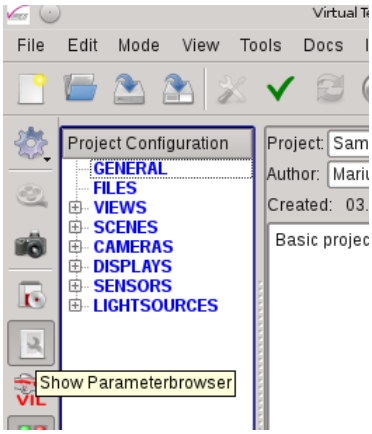
TaskControl

How do I maximize the performance of a SiL simulation?

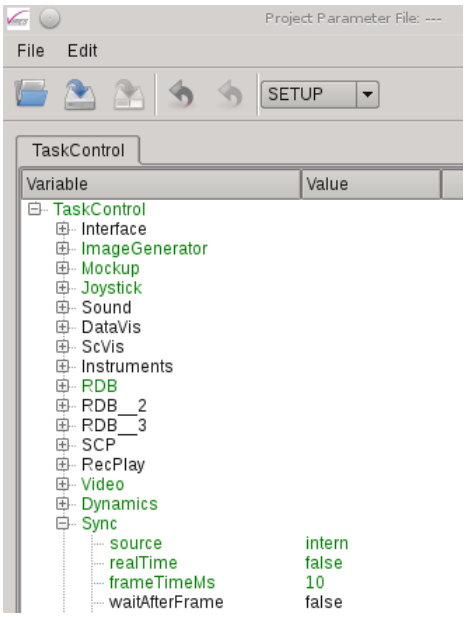
If you'd like to run a software-in-the loop test as fast as possible (i.e. non realtime), then you should explicitly turn off the realtime feature and the synchronization to the IG. Also, you have to indicate the desired duration of a simulation frame.

VTD 2.x:

- Open the Parameter Browser



- Set the browser to the mode "SETUP" (combo box on top)
- Configure the "Sync" parameters as required
- Press the "Save" button



- Apply the configuration, press INIT and then START

VTD 1.x:

Edit the taskControl configuration file (typically: *Setups/Current/Config/TaskControl/taskControl.xml*) as in the following example:

```
<TaskControl>
  ...
  <Sync source="intern" realTime="false" frameTimeMs="40" />
  ...
</TaskControl>
```

How do I retrieve player configuration details (i.e. names, IDs, categories)

You may query a player's ID / name / category by means of an SCP command.

In order to retrieve the information from a known ID, use the command

```
<Query entity="player" id="n"/>
```

The reply, if successful, will be

```
<Reply entity="player" name="name of player n" id="n" category="category of player n"/>
```

If not successful, it will be

```
<Reply entity="player" id="n"/>
```

In order to retrieve the information from a known name, use the command

```
<Query entity="player" name="name of player n"/>
```

The reply, if successful, will be

```
<Reply entity="player" name="name of player n" id="n" category="category of player n"/>
```

If not successful, it will be

```
<Reply entity="player" name="name of player n"/>
```

If you want to retrieve the above information for all players in a loaded scenario, just send

```
<Query entity="player"/>
```

You will receive individual reply messages (one for each registered player).

Video Streaming Issues

Video data may be retrieved by a custom task in two ways:

- from the **TaskControl** (TC) via the RDB image port
  - uses a TCP/IP connection
  - works in low-dynamic range (LDR) only
  - works for a single IG instance only
- directly from the **ImageGenerator** (IG) via Shared Memory (SHM) on the machine running the IG
  - works for low-dynamic range (LDR) **and** high-dynamic range (HDR)
  - works for multiple instances of the image generator
  - works with the post-processing pipeline

Live video-streaming via TaskControl (TCP/IP network port)

Note: this chapter applies to the **low-dynamic-range** (LDR) configuration **only!**

The live video streaming via a TCP socket of the *Runtime Data Bus (RDB)* must be turned on explicitly in the TaskControl configuration file (VTD 1.x, typically: *Data/Setups/Current/Config/TaskControl/taskControl.xml*) or in the Parameter Server settings of the current setup (VTD 2.x, *Data/Setups/Current/Config /vtdParamServerAutoLoad.stp.xml*).

The following parameters must be adjusted:

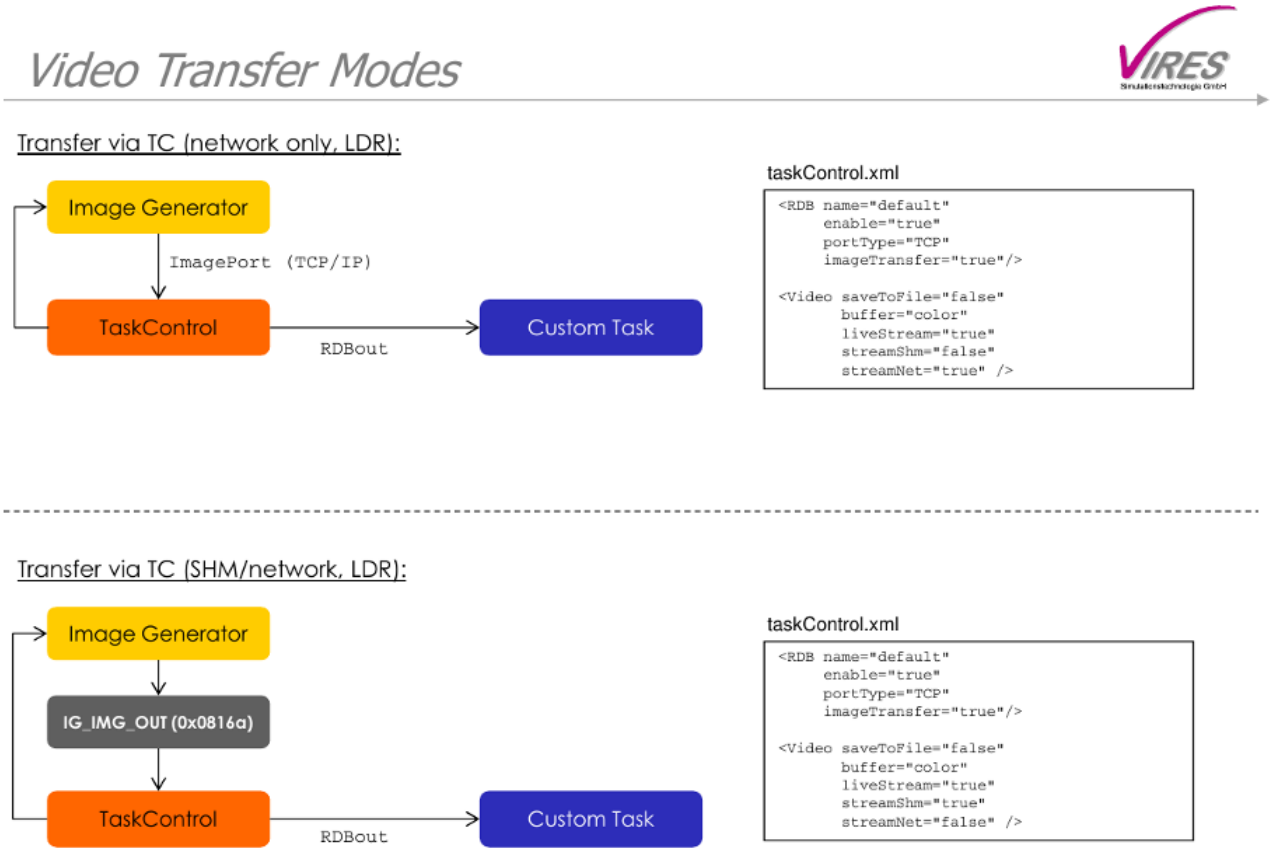
- the attribute *imageTransfer* within the `<RDB/>`-tag must be set to `true`
- the attribute *liveStream* within the `<Video/>`-tag must be set to `true`

Example:

```
<TaskControl>
  ...
  <RDB enable="true"
       portType="TCP"
       imageTransfer="true" />
  ...
  <Video saveToFile="false"
         buffer="color"
         liveStream="true" />
  ...
</TaskControl>
```

Unless otherwise specified, the (internal) image transfer between IG and TC will be performed via a TCP/IP network connection. This connection is **different** from the RDB connection from which you may be reading the resulting image data. Alternatively and for better bandwidth, the (internal) image transfer between IG and TC may also be configured as SHM communication.

See also the following figure:



In order to activate (internal) SHM video transfer between TC and IG, the above settings have to be modified in the following way:

```
<TaskControl>
  ...
  <RDB enable="true"
       portType="TCP"
       imageTransfer="true" />
  ...
  <Video saveToFile="false"
         buffer="color"
         liveStream="true"
         streamShm="true"
         streamNet="false" />
  ...
</TaskControl>
```

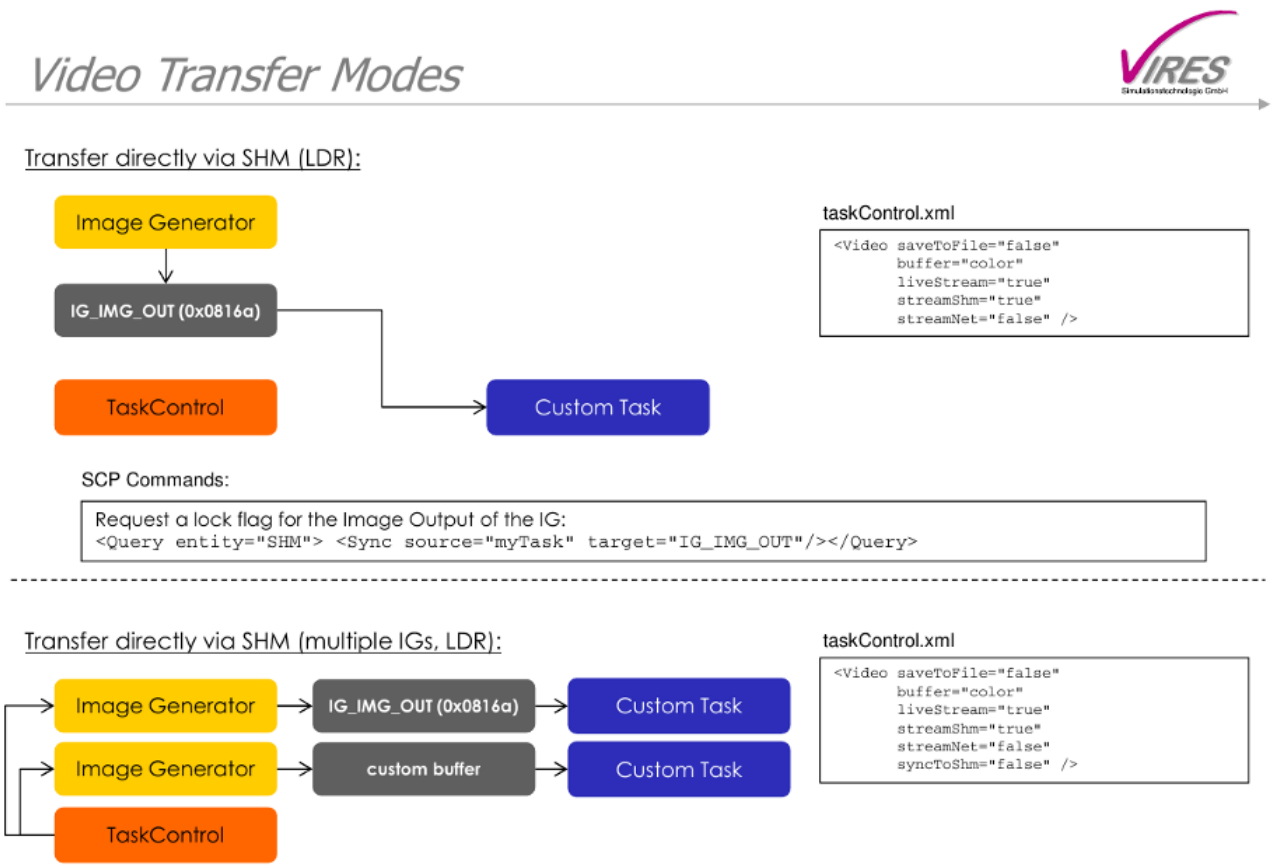
Notes:

- a custom task will stay in sync with the TC automatically since it receives the data via the TCP/IP connection of the RDB output channel. The internal communication between TC and IG is not relevant for external tasks
- for (internal) SHM communication between TC and IG, both must run on the same machine, of course
- a custom reader **must not** be connected to the (internal) SHM connection between IG and TC!

Video/Image transfer from the ImageGenerator via Shared Memory (SHM)

Low-Dynamic-Range (LDR) operation

The video data may be retrieved directly from the IG's shared memory segment. See the following figure for setups involving one or multiple image generators in LDR mode.



In order to activate the image transfer via SHM, the attributes *liveStream* and *streamShm* should be set to `true`.

Example:

```
<TaskControl>
   ...
   <Video saveToFile="false"
          ...
          liveStream="true"
          streamSHM="true"/>
   ...
</TaskControl>
```

The ImageGenerator writes per default to the SHM segment with the ID *RDB_SHM_ID_IMG_GENERATOR_OUT* as specified in the RDB header file.

Note: The above flag has no influence on the SHM writing of the OptiX extension into *RDB_SHM_ID_CUSTOM_01*.

Important Note: With a single IG in LDR operation, the synchronization may be realized using the TC's sync mask functionality (see figure above). For a multi-IG setup operation, the user has to take care of correct synchronization with the respective IG instances / shared memory segments. In this case, the TC should be informed that it does not have to watch the SHM state

```
<TaskControl>
   ...
   <Video saveToFile="false"
          ...
          liveStream="true"
          streamSHM="true"
          syncToShm="false"/>
   ...
</TaskControl>
```

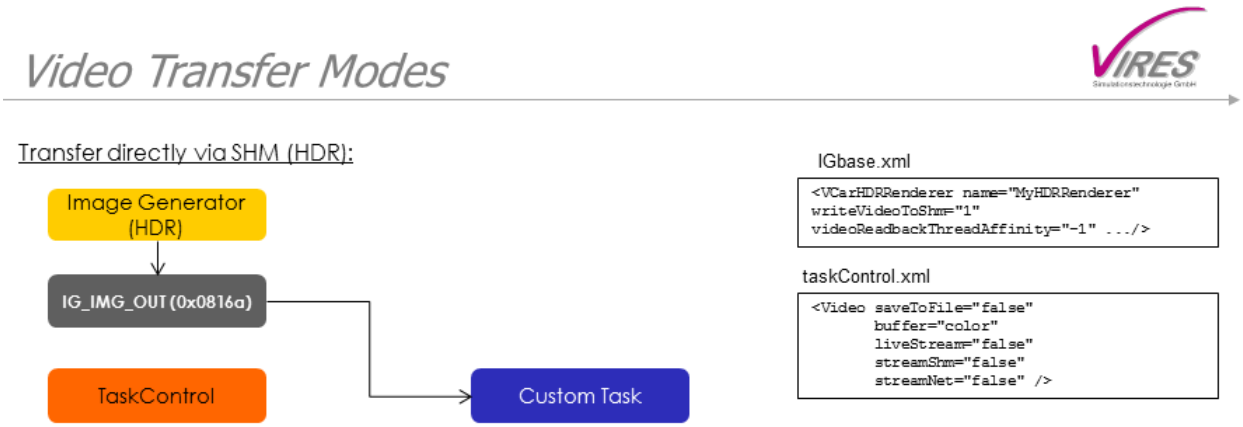
High-Dynamic-Range (HDR) operation

Preface: fo the shared memory configuration in HDR modoe please read first the chapter IG FAQ HDR.

The High-Dynamic Range mode supports only an image dump to the shared memory (SHM). Images are stored as RDB messages (`RDB_PKG_ID_IMAGE`) accompanied by the corresponding camera information (`RDB_PKD_ID_CAMERA`). In order to enable the video transfer to shared memory in general, register the component RDBInterface within the IG configuration file (typically: `VTD/Data/Setups/Current/Config/ImageGenerator/IGbase.xml`).

For enabling a per-frame image dump to shared memory, set the attribute `writeVideoToShm` within the tag `<VCarHDRRenderer>` to `true`. For this tag, see also chapter High-Dynamic-Range Rendering. The TaskControl **must not** be involved in this mechanism of image transfer!

See also the following figure:



Note: TC will not block automatically; triggering and locking needs to be done via Custom Task if full frame-sync operation is to be achieved

Important notes:

- if you are using a *post-processing pipeline* and the output of that pipeline is dumped into SHM, then the post-processing pipeline is taking care of the video transfer. Thus, `writeVideoToShm` must be **`false`**; otherwise the PP-output will be overwritten!
- The HDR SHM configuration interferes with the LDR video stream (TC) settings.
  - Enabling the **writeVideoToShm** attribute **and** using the video streaming feature of the TC will end in an **invalid** state.
  - Therefore, set the TC's configuration parameters `liveStream` and `streamToSHM` to **`false`**.
  - Ticket #2695 addresses the problem to merge the HDR SHM configuration.

How to read image data from the ImageGenerator's Shared Memory (SHM)

General Description

An extensive description of all aspects of the SHM mechanisms is given in

FiguresManualIO_SHM.pdf

Code Example - General

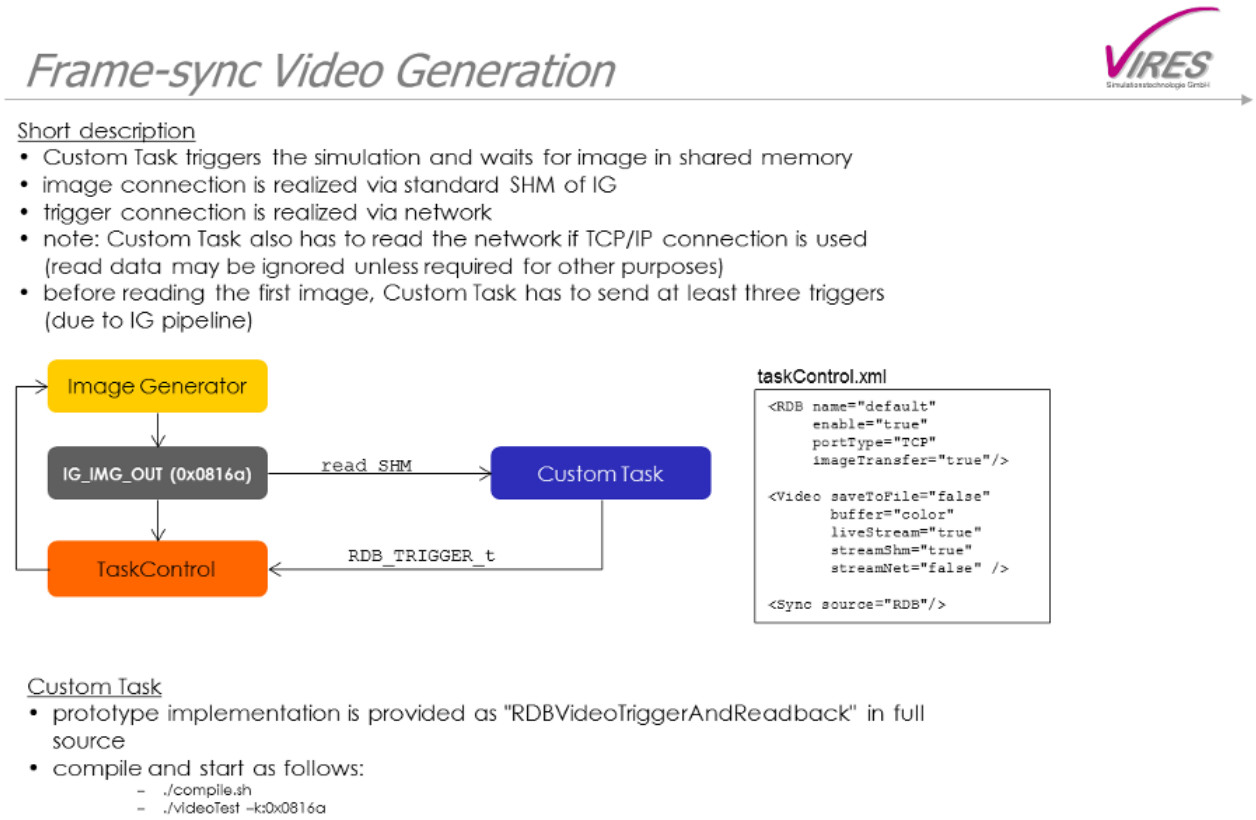
A coding example for attaching to the SHM of the ImageGenerator and reading its contents is given here:

RDB Data Transfer (Receiver Side) via Shared Memory

This example will read all data contained in the SHM segment, i.e. it will, typically, retrieve image data as well as camera data. The processing of the data itself has to be implemented by the user.

Full Example: explicit triggering of VTD and video readback via shared memory, HDR

This example provides the complete set of *setup*, *project* and custom reader. The layout is given in the following image:



Brief description

- an HDR video with fish-eye distortion is generated
- the TC is triggered via a custom component
- the custom component reads back the HDR image and the depth image via shared memory

What you need in addition:

- an HDR database and a scenario

The complete package of this example for VTD is stored here:

videoGenAndReadback.tgz

Replay ignores my GUI camera settings

If replay is also playing back the SCP commands then GUI camera settings will be overwritten by the contents of the record file. You may set the camera via the GUI after the start of the replay but any recorded camera command will override this setting when being replayed. It's definitely not a bug but a feature.

In order to configure the replay of SCP commands, edit the TaskControl configuration file (typically: *Data/Setups/Current/Config/TaskControl/taskControl.xml*).

Example:

```
<TaskControl>
   ...
   <Replay playSCP="true"
      ... />
   ...
</TaskControl>
```

Set the attribute *playSCP* to `false` if you don't want SCP commands to be replayed. Per default (i.e. also if the attribute *playSCP* is missing), SCP commands will be replayed.

How can I trigger the simulation (and control real-time behavior)?

The main simulation loop may be triggered from external sources or internal sources. The configuration is to be provided in the file

Data/Setups/Current/Config/TaskControl/taskControl.xml

within the tag *<Sync/>* (for an overview of the options, see also the SCP documentation).

Internal Triggering

Internal triggering means that the simulation loop is triggered from within the taskControl. The following options apply:

```
<Sync  source="intern"
       realTime="true"
       frameTimeMs="40"/>
```

This causes internal triggering with real-time conditions (i.e. simulated time corresponds to the actually passed time of day) and with a step width of 40ms. If the actual computation of the simulation takes more time than the *frameTimeMs* foresees, then the real-time constraints cannot be met. So, make sure that the time interval in real-time conditions is not too restrictive.

```
<Sync  source="intern"
       realTime="false"
       frameTimeMs="40"/>
```

Same as above (in terms of source and frame time), but the simulation will run as fast as possible. So, if the computation of a frame takes less time than the given frame time, your simulation will run faster than real-time, otherwise it will be slower

Dependency to other components:

The taskControl's frequency may be influenced by other components, too, so that you may not be able to reach the full frequency unless you are adapting also the other components (see also below How to synchronize the simulation). One **implicit dependency** is given already in the "Standard" configuration of VTD if you are using it in connection with the VIRES vehicle dynamics plugin that is running in the moduleManager: the taskControl may not run faster in frame-synchronous mode than the dynamics data is computed by the moduleManager (i.e. not faster than 100Hz in default configuration). If you want to stay frame-synchronous with the moduleManager's vehicle dynamics but still run faster than 100Hz (e.g. setting `frameTimeMs="1"` in the above configuration examples), you have to set the moduleManager to a different frequency (see also Changing the moduleManager's frame rate).

External Triggering

Triggering via ImageGenerator

In order to trigger the simulation for real-time interactive applications, it should be linked to the trigger signal of the image generator. This is achieved in the following way:

```
<Sync  source="extern"
       realTime="true"/>
```

In the standard configuration, it is assumed that the image generator (and the XServer) runs at exactly 60Hz. For other frequencies (e.g. if you force the IG to run at 30Hz or if your XServer runs at a different frequency) you have to adjust the frame time sent from the IG to the TaskControl. This can be done in the file

Data/Setups/Current/Config/ImageGenerator/IGbase.xml

in the following section (example is given for 30Hz)

```
<TAKATA ....
    constantDeltaTime="0.0333333333"
    useConstantDeltaTime="1"
... />
```

Triggering via RDB

If you want to control the simulation frame from within a 3rd party module, then you may trigger the main simulation loop explicitly via RDB. Whether this is real-time or not depends on your external simulation control. RDB triggering is configured the following way:

```
<Sync  source="RDB" />
```

In order to trigger each frame, you will have to send periodically an RDB message of type *RDB_PKG_ID_TRIGGER*. The structure

```
typedef struct
{
    float   deltaT;
    int32_t frameNo;
    int32_t spare;
} RDB_TRIGGER_t;
```

has to be filled with the delta time of the next simulation step (*deltaT*), e.g. 0.03333333 for a 30Hz frame rate, and the unique frame number (*frameNo*). The simulation will always block until it receives the next trigger message.

NOTE: When triggering via RDB, the trigger package should be sent in a separate message, i.e. not in combination with the previous OBJECT_STATE. The trigger message does not necessarily have to be surrounded by the START_OF_FRAME and END_OF_FRAME packages.

Triggering via SCP

If you want to control the simulation frame from within a 3rd party module that doesn't have a connection to the RDB, then you may trigger the main simulation loop explicitly via SCP. However, since a text protocol is involved, this method is not recommended for real-time or high-frequency purposes (actually, it has been designed for test purposes only). SCP triggering is configured the following way:

```
<Sync  source="SCP" />
```

In order to trigger each frame, you will have to send periodically an SCP message of the following type:

```
<SimCtrl><Sync dt="0.03333333"/></SimCtrl>
```

The delta time of the next simulation step (*dt*) in the example is given for a 30Hz frame rate and may be any arbitrary number greater than 1ms.

Overview of trigger modes and image generation modes

The way of triggering the simulation will also influence what you see in the image generator. Generally speaking, the IG may be in sync with the simulation (i.e. drawing each simulation frame) or it may be out of sync (rendering less or more images than the available simulation frames). Depending on the trigger mode for the IG and the TC (i.e. simulation), the following combinations may occur:

```
          TC-<Sync>            IG                                result
     extern, realtime     sync to v-blank           real-time sim with IG frequency, IG and TC in sync
                          explicitTrigger="false"

     intern, realtime     sync to v-blank           real-time sim with TC frequency, IG and TC out of sync
                          explicitTrigger="false"

     RDB                  sync to v-blank           sim with frequency from outside, IG and TC out of sync
                          explicitTrigger="false"

     intern, realtime     no sync to v-blank        real-time sim with TC frequency, IG and TC in sync
                          explicitTrigger="true"

     RDB                  no sync to v-blank        sim with frequency from outside, IG and TC in sync
                          explicitTrigger="true"

     intern, realtime     no sync to v-blank        real-time sim with TC frequency, IG and TC out of sync
                          explicitTrigger="false"
                          trigger via SHM

     RDB                  no sync to v-blank        sim with frequency from outside, IG and TC in or out of sync
                          explicitTrigger="false"   depending on sync between RDB-trigger for TC
                          trigger via SHM           and SHM-trigger for IG
```

The configuration issues concerning the image generator are described at IG-FAQ - Synchronization and Triggering.

Using a moduleManager Plug-in for Explicit Trigger Intervals

Note: this feature is not available in all installations.

Let's assume you have a set of sensors which requires individual timings each and also needs up-to-date traffic and dynamics information in each frame.

Problem:

Your sensors need to be updated each 16ms, 10ms and 50ms respectively. The simulation does not have to run in sync with the image generator (i.e. the IG may run as a mere consumer of data).

Solution:

You'll have to setup two instances of the moduleManager, one for the sensors and one for generating the trigger signal. The taskControl has to be set to trigger mode `RDB` and, thus, non-realtime. It is recommended to use frame-synchronously running module managers for both, the sensors and the trigger (i.e. moduleManagers whose command lines include the parameter `-t 0.0`). The sensors will be set up as usual, for the moduleManager which provides the trigger signal, the following configuration is recommended:

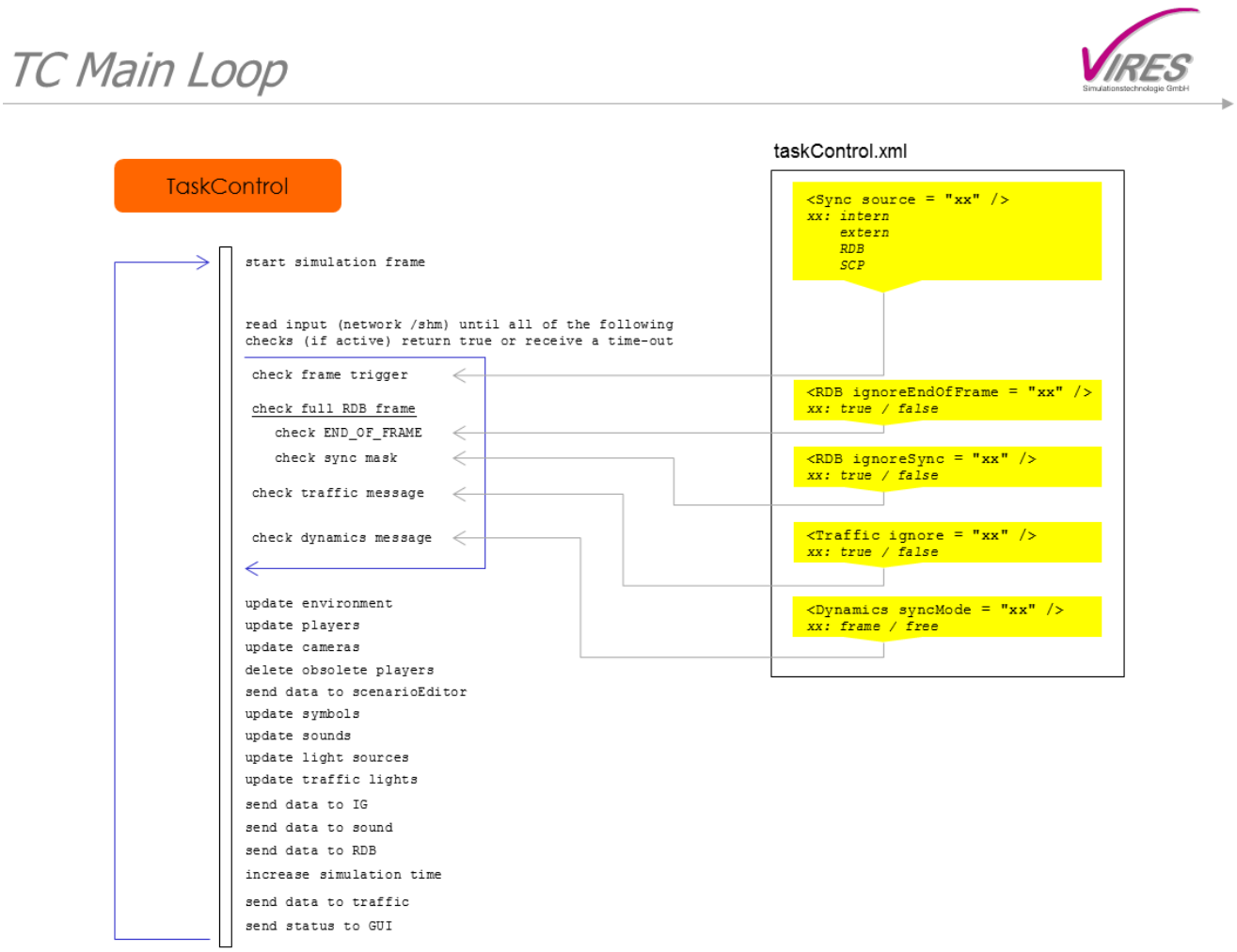
```
<RDB>
    <Port name="RDBraw" number="48190" type="TCP" />
</RDB>

<DynamicsPlugin name="viTrigger">
    <Load     lib="libModuleTrigger.so" path=""/>
    <Player   default="false" />
    <Config   verbose="false" />
    <Config   entity="timer" baseDelta="0.002" />  <!-- basic delta time is 2ms -->
    <Config   entity="timer" offset="0.0" delta="0.016" featureMask="0x0003"/>  <!-- IG -->
    <Config   entity="timer" offset="0.0" delta="0.010" featureMask="0x0001"/>
    <Config   entity="timer" offset="0.0" delta="0.050" featureMask="0x0001"/>
    <Debug    enable="false" />
</DynamicsPlugin>
```

You have to set a basic delta time (attribute `baseDelta`) which is used for minimum step size. The attribute `featureMask` in each timer configuration is currently (VTD 1.4.2) in a prototype state. It shall enable the user to identify parts of the entire simulation loop (within the taskControl) that shall be updated depending on the individual frame time that is applicable. In the example above, the IG (mask 0x0002) will be updated every 16ms whereas the traffic (mask 0x0001) will be updated in each frame.

With the trigger module specified as above, the simulation will be triggered at 10ms, 16ms, 20ms, 30ms, 32ms....

Internal Loop of the TaskControl

In order to better understand the overall simulation sequence, see the following figure for more details about the taskControl's internal loop and its dependency on various trigger / synchronization mechanisms



How can I synchronize the simulation?

Basic Synchronization to Vehicle Dynamics

The primary synchronization is between the vehicle dynamics (of the first *extern* vehicle) and the taskControl. You may configure this sync mode in the file

Data/Setups/Current/Config/TaskControl/taskControl.xml

within the section

```
<Dynamics  syncMode="xxxxx"/>
```

If *xxxx* is set to *frame*, the next simulation frame will only be triggered after the input from the vehicle dynamics has arrived.
If *xxxx* is set to *free*, the next simulation frame be triggered not taking into account whether a vehicle dynamics package has arrived. Any positions of external players will then be extrapolated from the last available information (position and speed) to the time of the current simulation frame.

Synchronization to 3rd Party Sources via RDB

If you're running one or more external processes that should each complete their internal frame before the next simulation frame is triggered, then you should register them with the TaskControl's sync mechanism and have them send explicit sync information upon completion of their frames. The technique is the same for each external process and should follow this pattern:

Register for Synchronization

Before or after INIT, register your component to act as a sync source. For this, send an SCP query with unique identification (i.e. name) of your source

```
<Query entity="RDB"> <Sync source="myComp123" /> </Query>
```

You will receive as an answer either:

```
<Reply entity="RDB"> <Sync source="myComp123" mode="free" /> </Reply>
```

```
or

<Reply entity="RDB"> <Sync source="myComp123" mask="128" mode="frame" /> </Reply>
```

In the former case, the simulation has been configured to run in free mode and explicit synchronization to external components is impossible. In the latter case, you receive a unique numeric sync mask (as integer value) that you need for the provision of the actual sync signal to the simulation.

Provide Synchronization Signal

After you have received the sync mask and whenever the simulation is in "RUN", it will wait for your mask to arrive via RDB in the package called *RDB_PKG_ID_SYNC*. With this package, you have to provide the following structure **in each simulation** frame (if you miss, the simulation will block!):

```
typedef struct
{
    uint32_t mask;
    uint32_t spare[3];
} RDB_SYNC_t;
```

Where *mask* holds the mask (i.e. the integer value) that you have received with the *<Reply.../>* command.

Un-register from Synchronization

You may terminate the dependency of the simulation from your component's sync signal by sending the following SCP command:

```
<Query entity="RDB"> <Sync source="myComp123" delete="true"/> </Query>
```

Note: upon receipt of as STOP command, the sync dependency will automatically be removed. Before the next start, you'll have to re-register your sync source again.

Synchronization to 3rd Party Sources via Shared Memory

This technique is quite similar to the one defined for the synchronization via RDB (see above). Instead of directly influencing the triggering of the next simulation frame, it will only influence the provision of new information to the shared memory by a producer (typically the IG). Indirectly, this may also influence the triggering, of course.

NOTE: The SHM synchronization of standard VTD components is currently only performed by the Image Generator.

In order to work with this technique, you should follow this pattern:

Register for Synchronization

Before or after INIT, register your component to act as a sync source of the Shared Memory. For this, send an SCP query with unique identification (i.e. name) of your compoenent

```
<Query entity="SHM"> <Sync source="myComp123" /> </Query>
```

You will receive as an answer:

```
<Reply entity="SHM"> <Sync source="myComp123" mask="256"/> </Reply>
```

which contains a unique numeric sync mask (as integer value) that you need for the provision of the actual sync signal to the simulation.

Provide Synchronization Signal

After you have received the sync mask and whenever the simulation is in "RUN", the producer of a given shared memory segment will wait for your mask to be cleared from shared memory before performing the next write operation. Note that in most cases SHM is multi-buffered (typically double-buffered), so that you may have to check various buffers before deciding which one to process. The following code snippet illustrates how this can be done for a component which is attached to the IG's shared memory:

The steps are

1. Attach to the shared memory (here: the SHM containing image data)
2. Lock the corresponding buffer while reading / processing it
3. Read the information
4. Release your Bit from the SHM mask (in this case, it's the bit that the TC has been statically defined; just replace it with your mask)
5. Unlock the buffer

```
bool
IfaceIG::checkForShmData()
{
    // attach to shared memory first
    ....

    // get a pointer to the shm info block located at the very beginning of the SHM
    RDB_SHM_HDR_t* shmHdr = ( RDB_SHM_HDR_t* ) ( mImgShm->getStart() );

    if ( !shmHdr )
        return false;

    // expecting double-buffered SHM!
    if ( ( shmHdr->noBuffers != 2 ) )
        return false;
```

```
    // allocate space for the buffer infos
    RDB_SHM_BUFFER_INFO_t** pBufferInfo = ( RDB_SHM_BUFFER_INFO_t** ) ( new char [ shmHdr->noBuffers * sizeof( RDB_SHM_BUFFER_
    RDB_SHM_BUFFER_INFO_t* pCurrentBufferInfo = 0;

    char* dataPtr = ( char* ) shmHdr;
    dataPtr += shmHdr->headerSize;

    for ( int i = 0; i < shmHdr->noBuffers; i++ )
    {
        pBufferInfo[ i ] = ( RDB_SHM_BUFFER_INFO_t* ) dataPtr;
        dataPtr += pBufferInfo[ i ]->thisSize;
    }

    // check which buffer to read
    if ( ( pBufferInfo[ 0 ]->flags & RDB_SHM_BUFFER_FLAG_TC ) && ( pBufferInfo[ 1 ]->flags & RDB_SHM_BUFFER_FLAG_TC ) )
    {
        ...
        // lock while reading
        pCurrentBufferInfo->flags |= RDB_SHM_BUFFER_FLAG_LOCK;
    }
    else if ( ( pBufferInfo[ 0 ]->flags & RDB_SHM_BUFFER_FLAG_TC ) )
    {
        ...
        // lock while reading
        pCurrentBufferInfo->flags |= RDB_SHM_BUFFER_FLAG_LOCK;
    }
    else if ( ( pBufferInfo[ 1 ]->flags & RDB_SHM_BUFFER_FLAG_TC ) )
    {
        ...
        // lock while reading
        pCurrentBufferInfo->flags |= RDB_SHM_BUFFER_FLAG_LOCK;
    }
    else
    {
        delete pBufferInfo;
        return false;
    }

    //process the actual data
    .......

    // release after reading
    pCurrentBufferInfo->flags &= ~RDB_SHM_BUFFER_FLAG_TC;  // this is for TC, 3rd party components should clear their own mask
    pCurrentBufferInfo->flags &= ~RDB_SHM_BUFFER_FLAG_LOCK;

    delete pBufferInfo;
    return true;
}
```

Un-register from Synchronization

You may terminate the dependency of the shared memory from your component by sending the following SCP command:

```
<Query entity="SHM"> <Sync source="myComp123" delete="true"/> </Query>
```

Driving with joysticks and game wheels

IMPORTANT NOTE: When driving with a joystick / game wheel / mockup, make sure that no other source provides the inputs of your ego vehicle.

IMPORTANT NOTE for VTD 2.0+: The configuration has moved from the file `taskControl.xml` to the ParamGui (and, thus, to the file `vtdParamServerAutoLoad.stp.xml` in *Data/Setups/Current/Config*. The parameters that need to be adjusted are the same as shown below for `taskControl.xml` but the interface is now the ParamGUI not the file itself. Please take this into consideration.

The first external vehicle may be controlled directly via a joystick or game wheel which is attached to the host running the TaskControl. In order to activate this feature, the TC configuration file (typically: *Data/Setups/Current/Config/TaskControl/taskControl.xml*) should be configured as follows:

```
<TaskControl>
    <Mockup type="Joystick".../>
</TaskControl>
```

For VTD 2.0+, the resulting config file (`vtdParamServerAutoLoad.stp.xml`) will contain the following entries (including all the additional information given in the paragraphs below):

```
<ViSimulation>
<Component name="TaskControl">

<TaskControl xmlns="http://www.vires.com/2015/VtdParamSchema/TaskControl">

  <Mockup>
    <type value="Joystick"/>
  </Mockup>
```

```
  <Joystick>
    <type value="custom"/>
    <debug value="false"/>
    <Axis__1>
      <name value="steering"/>
      <index value="0"/>
      <minRaw value="-32767"/>
      <maxRaw value="32767"/>
      <minNorm value="9"/>
      <maxNorm value="-9"/>
    </Axis__1>
    <Axis__2>
      <name value="throttle"/>
      <index value="2"/>
      <minRaw value="-32767"/>
      <maxRaw value="32767"/>
      <minNorm value="1"/>
      <maxNorm value="0"/>
    </Axis__2>
    <Axis__3>
      <name value="brake"/>
      <index value="3"/>
      <minRaw value="-32767"/>
      <maxRaw value="32767"/>
      <minNorm value="1"/>
      <maxNorm value="0"/>
    </Axis__3>
    <Button__1>
      <index value="4"/>
      <cmd value="&lt;Symbol name=mySymbol duration=2.0>&lt;PosScreen x=0.2 y=0.1/>&lt;Text size=40.0 data=Button_4_pressed/>&
    </Button__1>
    <Button__2>
      <index value="5"/>
      <cmd value="&lt;Symbol name=mySymbol duration=2.0>&lt;PosScreen x=0.2 y=0.2/>&lt;Text size=40.0 data=Button_5_pressed/>&
    </Button__2>
    <Button__3>
      <index value="0"/>
      <cmd value="&lt;Symbol name=mySymbol duration=2.0>&lt;PosScreen x=0.2 y=0.3/>&lt;Text size=40.0 data=Button_0_pressed/>&
    </Button__3>
  </Joystick>

</TaskControl>
</Component>
</ViSimulation>
```

Many joysticks and game wheels will be recognized automatically. However, if yours isn't recognized or if you're not satisfied with the calibration, you may configure the joystick and game wheel interface by yourself within the TaskControl configuration file.

Example:

```
<TaskControl>
    <Mockup type="Joystick".../>

    <Joystick type="custom" debug="false">
        <Axis name="steering" index="2" minRaw="-32767" maxRaw="32767" minNorm="1.0" maxNorm="-1.0" />
        <Axis name="throttle" index="0" minRaw="-28768" maxRaw="32767" minNorm="0.0" maxNorm="1.0" />
        <Axis name="brake"    index="1" minRaw="-17000" maxRaw="0"     minNorm="0.0" maxNorm="1.0" />
    </Joystick>

</TaskControl>
```

The joystick axes (attribute *index*) are mapped to logical axes (attribute *name*) and the valid raw values (as provided by the driver) are translated into normalized input values for the vehicle dynamics. For raw values out of the specified range, the normalized value of the respective logical axis will always be set to 0. If you want to find out which joystick axes and which raw data ranges are to be expected, set the attribute *debug* to `true` and watch the output of the TaskControl while operating your joystick. The simulation must be running in order to receive any values.

Starting with VTD 1.4, you may also assign SCP commands to the joystick buttons within the configuration entry. Here's an example for assigning reverse and driving gear to two buttons (a message will also be shown on-screen for a second when the gear is switched).

```
<TaskControl>
    <Mockup type="Joystick".../>

    <Joystick type="custom" debug="false">
        <Axis name="steering" index="2" minRaw="-32767" maxRaw="32767" minNorm="1.0" maxNorm="-1.0" />
        <Axis name="throttle" index="0" minRaw="-28768" maxRaw="32767" minNorm="0.0" maxNorm="1.0" />
        <Axis name="brake"    index="1" minRaw="-17000" maxRaw="0"     minNorm="0.0" maxNorm="1.0" />
        <Button index="4">
            <![CDATA[<Symbol name="dummy" duration="1.0"><PosScreen x="0.2" y="0.1"/><Text data="gear_D" colorRGB="0x00ff00" s
                    <Player name="Ego"><DriverInput gear="D"/></Player>]]>
         </Button>
        <Button index="5">
```

```
            <![CDATA[<Symbol name="dummy" duration="1.0"><PosScreen x="0.2" y="0.1"/><Text data="gear_R" colorRGB="0xff0000" s
                    <Player name="Ego"><DriverInput gear="R"/></Player>]]>
        </Button>
    </Joystick>

</TaskControl>
```

Settings for the Logitech G27 game wheel

These are the settings for the Logitech G27 game wheel that have to be performed within *taskControl.xml*:

```
    <Joystick type="custom" debug="false">
        <Axis name="steering" index="0" minRaw="-32767" maxRaw="32767" minNorm="1.0" maxNorm="-1.0"/>
        <Axis name="throttle" index="1" minRaw="-32767" maxRaw="0" minNorm="1.0" maxNorm="0.0"/>
        <Axis name="brake"    index="1" minRaw="0" maxRaw="32767" minNorm="0.0" maxNorm="1.0"/>
    </Joystick>
```

NOTE: The axis indices may be different depending on your configuration of the G27. The example above is valid for the "out-of-the-box" operation under OpenSuSE 11.4.

HINT
If you want to use the full performance of the G27 under Linux (i.e. the 900deg rotation angle and separate raw axes for throttle and brake), you will have to re-configure it. A tool for doing this can be found at

https://github.com/TripleSpeeder/LTWheelConf

The corresponding description is at

http://en.jo1jo.com/vb2/showthread.php?t=457672

Performance Monitoring

availability: VTD.2.0.3+

The performance of the *taskControl* and the *moduleManager* may be monitored using the following mechanism:

Provide an SCP command like

```
<RealtimeMonitor entity="taskControl" active="true" tolerance="0.1" verbose="true" live="true" bruadcast="true"/>
```

entity: "taskControl" or "moduleManager"

tolerance: fraction of nominal frame time that is tolerated before issuing a warning message

verbose: if true, output is a bit more detailed

live: if true, permanent monitoring data will be reported in shell

broadcast: if true, taskControl will also send RDB message `RDB_PKG_ID_PERFORMANCE`, so that external entities can monitor the real-time behavior of the taskControl (**note:** this attribute is only available for the taskControl, not for the moduleManager)

The output of the monitoring with live reporting will look similar to the following:

```
NOTICE 17:29:49:295.69(taskCtrl): "Performance::print: mRealTimeTolerance = 0.10000l"
NOTICE 17:29:49:295.79(taskCtrl): "Performance::print: mNoMeasurements / mNoUnderruns  /  mNoOverruns = 1123 / 19 / 15"
NOTICE 17:29:49:295.83(taskCtrl): "Performance::print: simTime = 18.716s, measured real-time = 18.802s, delta = -86ms, last dt
```

ModuleManager

see ModuleManager

Sound

The sound module is an addOn to the standard VTD distribution. If you want to use the sound system make sure that you have a license and the appropriate executables. The sound module is usually installed at `Runtime/AddOns/Sound.x.y`

How can I play an arbitrary sound sample via SCP?

If you want to play an individual sound sample based on an SCP command, you have to do the following (in the sample, two sounds are defined which may be activated independently):

a) Edit the sound config file (typically: `Runtime/AddOns/Sound.x.y/sound.xml`)

```
    <BufferManager>
        :
        <Sample     name="MY_SAMPLE_A"  file="MySound/sampleA.wav"   />
        <Sample     name="MY_SAMPLE_B"  file="MySound/sampleB.wav"   />
        :
    </BufferManager>

    :
```

```
<SimCar>
    :
    <SoundSource id="0" buffername="MY_SAMPLE_A" gain="1.0" loop="false" />
    <SoundSource id="1" buffername="MY_SAMPLE_B" gain="1.0" loop="false" />
    :
</SimCar>
```

b) Send the corresponding play command via SCP

```
<Sound name="mySound_A"><Start id="0"/></Sound>
<Sound name="mySound_B"><Start id="1"/></Sound>
```

How can I configure the sound to use an interface other than eth0?

If you want to use a different ethernet interface for your sound module, you will have to adjust the TaskControl and the Sound settings. In the following example, *eth2* shall be used instead of eth0.

TaskControl (e.g. Data/Setups/Current/Config/TaskControl/taskControl.xml):

```
<TaskControl>
  :
  <Sound enable="true"
         interface="eth2" />
  :
</TaskControl>
```

Sound (e.g. Runtime/AddOns/Sound.3.2/sound.xml):

```
<CarSoundSystem
  timeout="true"
  netIf="eth2"
  netType="UDP"
  tcHost="name_of_host_running_taskControl"
  useScp="true">
```

The *name_of_host_running_taskControl* must be contained in your local hosts file.

How can I configure the sound to use TCP instead of UDP?

Per default, TaskControl and Sound communicate via UDP connection. If this spams your network or you don't want to see broadcasts for any other reason, you may configure the sound to use TCP. Please perform the following steps:

TaskControl (e.g. Data/Setups/Current/Config/TaskControl/taskControl.xml):

```
<TaskControl>
  :
  <Sound enable="true"
         portType="TCP" />
  :
</TaskControl>
```

Sound (e.g. Runtime/AddOns/Sound.3.2/sound.xml):

```
<CarSoundSystem
  timeout="true"
  netType="TCP"
  tcHost="name_of_host_running_taskControl"
  useScp="true">
```

The *name_of_host_running_taskControl* must be contained in your local hosts file.

How can I use own sound samples for my vehicle's sound?

You may add your own sound samples to the library of sounds and use them instead of the ones furnished with VTD. Most probably, you will only want to change the engine's sound; therefore, we are describing only this procedure in the following lines.

Recording

You have to record sound samples (as perceived by the driver) for various engine speeds (RPMs) and engine loads. For our standard BMW_3.5 liter engine, our sampling has been:

- RPM: 1200 / 2000 / 3000 / 4000 / 5000 / 6000
- Load: 0 % / 30 % / 60 % / 100 %

That'll give you a total of 24 sound samples (see directory `AddOns/Sound.3.2/Engine/BMW_35i`). Of course, you may also use other engine speeds and loads.

Registration

Once you have done the recording, edit the file `sound.xml` furnished with VTD or create your own copy. In this file, go to the section `<Engine>` and edit the entries for `<Rpm...>` and @<Load...>. For the example given above, the entries are:

```
<Rpm idx="0" rpm="1200.0" />
<Rpm idx="1" rpm="2000.0" />
<Rpm idx="2" rpm="3000.0" />
<Rpm idx="3" rpm="4000.0" />
<Rpm idx="4" rpm="5000.0" />
<Rpm idx="5" rpm="6000.0" />
<Load idx="0" load="0.0" />
<Load idx="1" load="0.3" />
<Load idx="2" load="0.6" />
<Load idx="3" load="1.0" />
```

Note: The number of indices must correspond to the number of entries for engine speed and load (as you may have guessed anyway).

In the next step, assign specific sound files to specific combinations of an engine speed index and an engine load index each. In our example, this is:

```
<Sample idxRpm="0" idxLoad="0" file="Engine/BMW_35i/engine_1200_000.wav" />
<Sample idxRpm="1" idxLoad="0" file="Engine/BMW_35i/engine_2000_000.wav" />
<Sample idxRpm="2" idxLoad="0" file="Engine/BMW_35i/engine_3000_000.wav" />
<Sample idxRpm="3" idxLoad="0" file="Engine/BMW_35i/engine_4000_000.wav" />
<Sample idxRpm="4" idxLoad="0" file="Engine/BMW_35i/engine_5000_000.wav" />
<Sample idxRpm="5" idxLoad="0" file="Engine/BMW_35i/engine_6000_000.wav" />
<Sample idxRpm="0" idxLoad="1" file="Engine/BMW_35i/engine_1200_030.wav" />
<Sample idxRpm="1" idxLoad="1" file="Engine/BMW_35i/engine_2000_030.wav" />
<Sample idxRpm="2" idxLoad="1" file="Engine/BMW_35i/engine_3000_030.wav" />
<Sample idxRpm="3" idxLoad="1" file="Engine/BMW_35i/engine_4000_030.wav" />
<Sample idxRpm="4" idxLoad="1" file="Engine/BMW_35i/engine_5000_030.wav" />
<Sample idxRpm="5" idxLoad="1" file="Engine/BMW_35i/engine_6000_030.wav" />
<Sample idxRpm="0" idxLoad="2" file="Engine/BMW_35i/engine_1200_060.wav" />
<Sample idxRpm="1" idxLoad="2" file="Engine/BMW_35i/engine_2000_060.wav" />
<Sample idxRpm="2" idxLoad="2" file="Engine/BMW_35i/engine_3000_060.wav" />
<Sample idxRpm="3" idxLoad="2" file="Engine/BMW_35i/engine_4000_060.wav" />
<Sample idxRpm="4" idxLoad="2" file="Engine/BMW_35i/engine_5000_060.wav" />
<Sample idxRpm="5" idxLoad="2" file="Engine/BMW_35i/engine_6000_060.wav" />
<Sample idxRpm="0" idxLoad="3" file="Engine/BMW_35i/engine_1200_100.wav" />
<Sample idxRpm="1" idxLoad="3" file="Engine/BMW_35i/engine_2000_100.wav" />
<Sample idxRpm="2" idxLoad="3" file="Engine/BMW_35i/engine_3000_100.wav" />
<Sample idxRpm="3" idxLoad="3" file="Engine/BMW_35i/engine_4000_100.wav" />
<Sample idxRpm="4" idxLoad="3" file="Engine/BMW_35i/engine_5000_100.wav" />
<Sample idxRpm="5" idxLoad="3" file="Engine/BMW_35i/engine_6000_100.wav" />
```

h3 Restart

Once the recording and registration are done, re-start the sound module with your new configuration file as command line attribute, example:

```
./vroom -f myFile.xml
```

That's it.

Databases

How can I add detailed surface information to an existing road network?

Detailed surface information may be used for improving the inputs to the vehicle dynamics simulation. In the visualization, it will provide far more variation in the display of a road's surface. For the time being, OpenCRG http://www.opencrg.org is the format which is required to introduce detailed surface information into the VTD environment.

Visual Database

For the visual database, the tools for generating the road surface are currently available internally only. They will be made part of ROD in an upcoming release.

Logical Database

The logical database (OpenDRIVE file) may be tweaked manually to refer to a surface description. In combination with the latest OpenDRIVE Manager (V 1.3.5 and higher) which is being used by all VTD components, a surface description file may influence elevation or friction of a given road.

The easiest way of applying a surface description to a given road in the OpenDRIVE file is the so-called **attached** mode which replaces the reference line of the OpenCRG data with the one of the OpenDRIVE road.

Here's a recipe (replace with your applicable file names etc.):

1. Open your OpenDRIVE file in a text editor

   ```
   kwrite enhancedNetwork.xodr
   ```

2. Identify the road which is to be enhanced with the OpenCRG data
   1. search for *road id="<id of the road>"*, e.g.

      ```
      <road id="1" junction="-1" name="testRoad" length="5.0416614452820000e+03">
      ```

   2. search for the closing tag of the road

```
</road>
```

3. Insert the surface description tags as given in the OpenDRIVE specification **before** the closing tag

```
<road id="1" junction="-1" name="testRoad" length="5.0416614452820000e+03">
:
    <surface>
      CRG file="Data/crgMasterFileA.crg" sStart="0" sEnd="5.0417000000000000e+03" orientation="same" mode="attached"
          sOffset="0" tOffset="0" zOffset="0" zScale="1.0000000000000000e-01" hOffset="0" purpose="friction"/>
    </surface>
</road>
```

   Per default, OpenCRG data will be used for the road elevation (i.e. as the delta to the macroscopic elevation described by the OpenDRIVE `<elevation>` tag). If you specify the optional attribute `purpose="friction"` (as shown in the example), the OpenCRG data will be interpreted as friction values instead.
4. Make sure, your OpenCRG file can be applied to the road. The easiest way is to define all OpenCRG values repeatable in u- and v- direction and, thus, make it applicable over the entire OpenDRIVE road. For this, the header of the OpenCRG file should contain the following elements:

```
$ROAD_CRG_OPTS
*
* CRG elevation grid border modes in u and v directions
* BORDER_MODE_U = 0             ! return NaN
* BORDER_MODE_U = 1             ! set zero
*  BORDER_MODE_U = 2             ! keep last (default)
  BORDER_MODE_U = 3             ! repeat
  BORDER_OFFSET_U = 0.0         ! z offset beyond border (default: 0)
*
* BORDER_MODE_V = 0             ! return NaN
* BORDER_MODE_V = 1             ! set zero
* BORDER_MODE_V = 2             ! keep last (default)
  BORDER_MODE_V = 3             ! repeat
  BORDER_OFFSET_V = 0.0         ! z offset beyond border (default: 0)
$!*************************************************************
$ROAD_CRG_MODS
 SCALE_Z_GRID = 1.00
 GRID_NAN_MODE = 2
 SCALE_SLOPE = 1.0
 SCALE_BANKING = 1.0
 REFLINE_OFFSET_Z = 0.0
$!*************************************************************
```

These sections should be introduced no later than before the section `$KD_DEFINITION`. If the sections `$ROAD_CRG_OPTS` and `$ROAD_CRG_MODS` already exist in your file, then adjust them accordingly.

That should be it! Now you should be able to drive on the OpenCRG-enhanced road.

Further topics are handled in VTD FAQ Road Designer

How does VTD support Geo-Referencing

Note: the following lines will fully apply to VTD 2.0 and higher. Prior to its release, preliminary versions may be available upon individual request.

All geo-referencing is performed via the OpenDRIVE Manager library which is an inherent part of the respective VTD modules (TaskControl, ModuleManager, OdrGateway). With version 1.4 of OpenDRIVE, geo-referencing has become an option within a database definition. In order to facilitate the work with geo-referenced databases within VTD, some hints are provided here and will be updated with on-going development.

Projection Information

In an OpenDRIVE file, the information about the projection from, typically, a WGS84 system to, e.g. UTM32N, (and vice versa) is described as a child of the tag `<geoReference>`. This information is contained in an OGC-WKT string. The OpenDRIVE Manager and, thus, VTD use the free library "proj4" https://github.com/OSGeo/proj.4 for co-ordinate conversion. This library provides its own means for defining a projection rule, and converting OGC-WKT strings into proj4 strings is not always straightforward. Therefore, VTD has come up with various ways of handling this issue:

OGC-WKT Interpretation for Transverse Mercator

In order to increase compatibility of OpenDRIVE databases between different tools and in order to address some tool-specific features like a preference for small inertial co-ordinates (due to accuracy constraints), we propose using an OGC-WKT string according to the following pattern:

```
PROJCS["My Projection",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            AUTHORITY["EPSG","6326"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199328,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4326"]],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
```

```
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin", pLat ],
    PARAMETER["central_meridian", pLon ],
    PARAMETER["scale_factor", pScale ],
    PARAMETER["false_easting", pX],
    PARAMETER["false_northing",pY],
    AUTHORITY["EPSG","32632"],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH]]
```

The parameters *pLat*, *pLon*, *pScale*, *pX* and *pY* are user-defined and may be adapted as necessary. They provide a good means to keep the co-ordinates within the actual OpenDRIVE nodes small and have a valid projection setting nonetheless.

The above OGC-WKT can easily be transformed into the corresponding proj4 string:

```
+proj=tmerc +lon_0=pLon +lat_0=pLat +ellps=WGS84 +k_0=pScale + +x_0=pX + y_0=pY
```

Direct Definition of the proj4 String

The projection string defined in an OpenDRIVE file may be overwritten in the corresponding VTD components by a user-defined proj4 string. This will help adapting / correcting the interpretation of OGC-WKT strings if the internal conversion fails. The following SCP command has been introduced:

```
<SimCtrl><LayoutFile originX="aaa" originY="aaa" originZ="aaa" proj4CfgInertial="+proj=utm +zone=32 +ellps=WGS84 +datum=WGS84
+units=m +no_defs"/></SimCtrl>
```

Typically, the values for *originX*, *originY* and *originZ* should all be zero since offsets may be defined in the proj4 string itself. All VTD components will be adapted for handling this command, thus enabling the entire tool-chain to work in a very flexible way with individual projection settings.

Conversion Methods

TaskControl

You may feed the TaskControl with geo-graphic co-ordinates contained in the position information of `RDB_PKG_ID_OBJECT_STATE` packages (VTD 1.4.3+). Make sure that you set the co-ordinate type within the message to `RDB_COORD_TYPE_GEO` for a conversion to take place. The following assignments are foreseen:

```
pos.x: longitude [deg]
pox.y: latitude  [deg]
pos.z: altitude  [m]
pos.h: heading   [rad]
pos.p: pitch     [rad]
pos.r: roll      [rad]
```

Upoon receipt, these data will be converted to the corresponding inertial data.

ModuleManager

If you are writing ModuleManager plugins, you may use the following two methods for converting geo-coordinates into inertial co-ordinates and vice versa:

```
        /**
        * convert inertial position into geo position
        * @param inX  inertial position, x co-ordinate [m]
        * @param inY  inertial position, y co-ordinate [m]
        * @param inZ  inertial position, z co-ordinate [m]
        * @param inH  inertial position, heading [rad]
        * @param geoLong  resulting geo position, longitude [deg]
        * @param geoLat   resulting geo position, latitude  [deg]
        * @param geoZ     resulting geo position, z co-ordinate [m]
        * @param geoH     resulting geo position, heading [rad]
        * @return true if successful
        */
        bool inertial2geo( const double & inX, const double & inY, const double & inZ, const double & inH,
                           double & geoLong, double & geoLat, double & geoZ, double & geoH );

        /**
        * convert inertial position into geo position
        * @param geoLong  eo position, longitude [deg]
        * @param geoLat   eo position, latitude  [deg]
        * @param geoZ     eo position, z co-ordinate [m]
        * @param geoH     eo position, heading [rad]
        * @param inX  resulting ginertial position, x co-ordinate [m]
        * @param inY  resulting ginertial position, y co-ordinate [m]
        * @param inZ  resulting ginertial position, z co-ordinate [m]
        * @param inH  resulting ginertial position, heading [rad]
        * @return true if successful
        */
        bool geo2inertial( const double & geoLong, const double & geoLat, const double & geoZ, const double & geoH,
                           double & inX, double & inY, double & inZ, double & inH );
```

RoadDesigner (ROD)

see VTD FAQ Road Designer.

Traffic and Scenario

Driving on the "left side"

Per default, the traffic module will be configured for "right side" driving which is found in many countries including Germany, the USA etc. But, of course, countries using the opposite side for driving (UK, Japan, India etc.) are also covered by our traffic module, provided that you have a logical database (.xodr file) with correct connections in intersections, correct placement of traffic signs etc.

In order for the traffic module to work on these databases, just set the command line argument `-leftHand` when you start the executable `ghostdriver`. For VTD 2.x, just modify the entry in `Data/Setups/Current/Config/SimServer/simServer.xml` as follows:

```
<Process>
  name="Traffic"
  :
  cmdline="-leftHand -interface vt -seed 9"
  :
</Process>
```

A vehicle starts automatically although it shouldn't or vice versa

This behavior is identical for internal and external vehicle if the ModuleManager is configured to run external vehicle. Otherwise it will only apply to internal vehicles. Two simple rules apply:

1. A vehicle will start automatically,
   - if it has a *driver* **AND**
     - *no action* has been assigned to it **OR**
     - an action of type *autonomous* **OR**
     - an action of type *speed change* with target speed != 0
       has been assigned to it and the corresponding trigger becomes valid upon start of the simulation

1. A vehicle will *NOT* start automatically,
   - if it has *no driver* **OR**
   - if it has a *driver* **AND**
     - *at least one action* has been assigned to it **AND**
       - no action of type *autonomous* **AND**
       - no action of type *speed change* with target speed != 0
         has been assigned to it whose trigger becomes valid upon start of the simulation

So, if you want a vehicle which already carries an action to start autonomously upon start of the simulation, you have to assign a *driver* and an action *autonomous* which is executed at the very beginning of the simulation.

A lane change is not executed completely and the driver steers back to the original lane.

If the time for a lane change is too short, the driver may not make it to the target lane and after termination of the action will steer back to the original lane. In this case, increase the time for the lane change.

How can I define a steering trajectory for a player that also uses a waypoint path (route)?

1. Just assign a route to the player as usual.
2. Use the path shape tool to create a trajectory. Please make sure the shape starts and ends on a valid lane of your route.
3. Add the shape to the watched trajectories via a SCP "ActionNominalTrajectory".

The player will start on his route following it until he reaches the beginning of the trajectory. After using the trajectory as steering input, he will continue driving on the route from his current position on.

Note: Users who want to continue processing the steering target received via RDB may consider using either spline curves or clothoids as trajectory. Polylines are more likely to create undesired input as of their discontinuous nature.

Is there a specification of the scenario description?

For the schema file of the scenario description please check the manuals section.

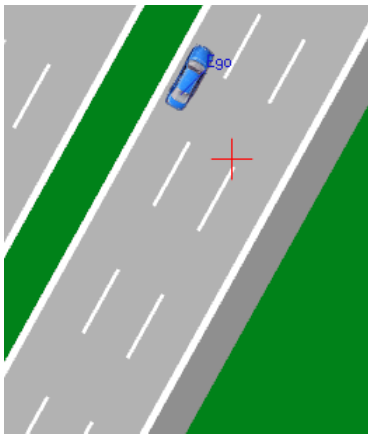
The scenario editor is described in a separate manual within the *Doc* directory of the distribution.

Configuration of entities (3d-models of players and objects)

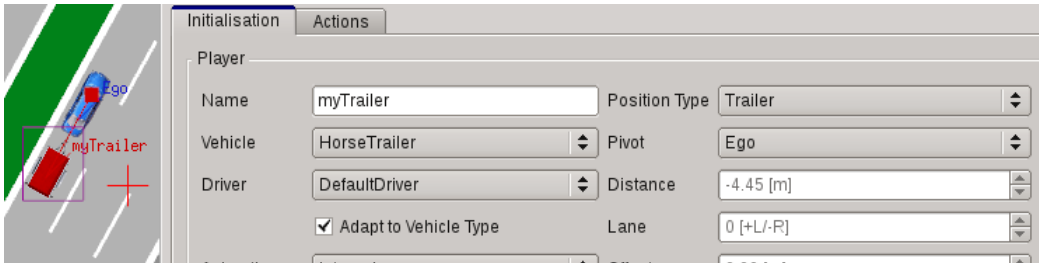
Adding a trailer to a vehicle

Basically, any vehicle may be used as truck or trailer but, of course, only a subset really makes sense. It is the user's responsibility to set up the correct combination of models. The steps for setting up a truck-trailer combination in the ScenarioEditor is as follows:

- Create the truck/parent (internal or external vehicle):



- Create the trailer
- Set its *Position Type* to `Trailer`
- Set its *Pivot* to the truck (here: `Ego`)
- That's it!



Recommended combinations of trailers vs trucks are as follows:

- *HorseTrailer, Trailer*: any passenger car / SUV / van
- *Site Vehicle Trailer*: TGL Site Vehicle
- *TGX Trailer**: TGX
- *LKW Trailer*: MerrcedesActros_11

How can I add new vehicle models to my VTD configuration?

Description for versions VTD.1.4 and higher

For adding new vehicles to VTD, the procedure of adding configuration files hasn't changed (see Description for versions VTD.1.2 to VTD.1.3). However the content of the configuration files has changed and is described here:

Vehicle Configuration Files
Material System

Description for versions VTD.1.2 to VTD.1.3

In order to be able to load a new model into the IG and select it for players in the ScenarioEditor, the following procedure has to be performed.

Note: For this example we assume the model file to be named "MyCar.ive"

- Prerequisite:
  - The vehicle must be created as IVE-file and is located in the Databases/Cars folder of the appropriate project. Please consider the manual for a description of this process.
- Edit the vehicle database:
  - The vehicle database is located at *$VTD/Distros/Current/Config/Players/Vehicles* (where $VTD is the path to the user's local VTD installation) or in *$VTD/Projects/Current/Config/Players/Vehicles*. It is always recommended to create a copy of the original distro database in the user project where the new vehicle shall be used.
    This is a folder containing a file for each vehicle model, that can itself contain multiple dynamic and color descriptions.

> *1. Section <`PlayerConfig`><`Graphics`>:*

```
<Model ID="Car_MyCar" filename="MyCar.ive"
```

The model id can be freely chosen, but must be unique among all models.

```
<VehicleTemplate name="MyCar - green" DBEntityName="Car_MyCar" >
```

Different material descriptions can be added to the template if they are supported by the user model. If needed, please consider the IG manual for a description.

```
    <Appearance baseColor="0.1 0.8 0.0" model="MyCar 2001" />
</VehicleTemplate>
```

With this definition a new model named "MyCar - green" appears in the ScenarioEditor *ScenarioProperties/VehicleDefinitions* dialog in the Model list.

> *2. Section <Scenario><VehicleDynamics>:*
> *Copy an existing VehicleDef entry and change all attributes to match the desired car.*

```
<VehicleDef Name="MyCarDefinition" .../>
```

> *3. Section <Scenario><VehicleModels>*
> *Create an own vehicle model that references the template and dynamics of the user model.*

```
<Model Name="MyCar - Vehicle" Appearance="MyCar - green" VehicleDynamics="MyCarDefinition" IconColor="auto" Swarm="1" />
```

> *IconColor="auto" means that the preview color of the appearance is used.*
> *Swarm="1" enables this vehicle model for swarm players. If set to zero the user model is not automatically used by swarm players.*

Info:

- In case you still have old user vehicle databases *vehicleCfg.xml* and do not want to split them into separate files, it is possible to move this file into directory *Vehicles* where it will also be parsed and interpreted.
- If you also port user **Setups** to VTD.1.2+, please edit the file *Config/ScenarioEditor/scenarioEditor.xml.ini* and replace:
    1. at *VehicleModels* "vehicleCfg.xml" with "Vehicles"
    2. at *ObjectModels* "objectCfg.xml" with "Objects"
- After that and a restart of ScenarioEditor old scenarios just need to be opened and saved again in order to reference the new DB paths.

Description for versions prior VTD.1.2

Note: For this example we assume the model file to be named "MyCar.ive" and the next unused type id of all vehicle templates is e.g. 80.

- Prerequisite:
    - The vehicle must be created as IVE-file and is located in the Databases/Cars folder of the appropriate project. Please consider the manual for a description of this process.
- Edit the vehicle database:
    - The vehicle database is located at *$VTD/Distros/Current/Config/Players/vehicleCfg.xml* (where $VTD is the path to the user's local VTD installation) or in *$VTD/Projects/Current/Config/Players/vehicleCfg.xml*. It is always recommended to create a copy of the original distro database in the user project where the new vehicle shall be used.

> *1. Section <PlayerConfig><Graphics>:*

```
<Model ID="Car_MyCar" filename="MyCar.ive"
```

The model id can be freely chosen, but must be unique.

```
<VehicleTemplate name="MyCar" Model="Car_MyCar" type="80" />
```

Different material descriptions can be added to the template if they are supported by the user model. If needed, please consider the IG manual for a description.

> *2. Section <Scenario><VehicleAppearance>:*

```
<Appearance Name="MyCar - green" Model="Vendor Year" Color="0.1 0.8 0.0" MovingModel="80"  />
```

With this definition a new model named "MyCar - green" appears in the ScenarioEditor *ScenarioProperties/VehicleDefinitions* dialog in the Model list.

> *That is all to be able to select a user model for any vehicle definition. In order to have a permanent vehicle definition in the ScenarioEditor with the right physical properties presented in the vehicle definitions list, these additional entries can be done:*
>
> *3. Section <Scenario><VehicleDynamics>:*
> *Copy an existing VehicleDef entry and change all attributes to match the desired car.*

```
<VehicleDef Name="MyCarDefinition" .../>
```

> *4. Section <Scenario><VehicleModels>*
> *Create an own vehicle model that references the appearance and dynamics of the user model.*

```
<Model Name="MyCar - Vehicle" Appearance="MyCar - green" VehicleDynamics="MyCarDefinition" IconColor="auto" Swarm="1" />
```

> *IconColor="auto" means that the preview color of the appearance is used.*
> *Swarm="1" enables this vehicle model for swarm players. If set to zero the user model is not automatically used by swarm players.*

How do I obtain the model number for a given vehicle model?

Upon initialization of the simulation, the traffic module will generate the numeric model numbers for the available vehicle models definitions. The entire configuration is sent to the image generator. If you want to receive the same information, edit the file *Setups/Current/Config/TaskControl/taskControl.xml* and edit the section `ImageGenerator`:

```
<TaskControl>
:
    <ImageGenerator
    :
                    dynPlayerConfig="true"
                    forwardPlayerConfig="true"
    :
    />
</TaskControl>
```

You will receive the entire configuration information. Look for the attribute `templateID` within `<VehicleTemplate>` in order to retrieve the numerical model ID.

Example:

```
<ImageGenerator>
  <PlayerDef>
    <VehicleTemplate name="Audi A3 - black" DBEntityName="Car_Audi_A3" templateId="1">
    :
    </VehicleTemplate>
  </PlayerDef>
</ImageGenerator>
```

How can I add new objects to my VTD configuration?

In order to be able to load a new model into the IG and select it for objects in the ScenarioEditor, the following procedure has to be performed.

1. Replace the soft link `$VTD/Data/Projects/Current/Databases/ObjectsCommon` with a copy of `$VTD/Data/Distros/Current/Databases/ObjectsCommon`.
2. Copy all your models as optimized IVE files into this directory.
3. Copy directory `$VTD/Data/Distros/Current/Config/Players/Objects` to `$VTD/Data/Projects/Current/Config/Players/`
4. This directory already contains an example `Cube.xml` that can be copied and adapted to the new model. You can create a single file for each object or sum up multiple objects in one file as needed.

Explanation of the necessary elements of `ObjectConfig`:

- `<Model>` defines a unique name for the model and the path to the IVE file.
- `<Graphics>` `model` references the model description above, `icon` can be used to define a preview image for the ScenarioEditor.
- `<ObjectTemplate>` `name` defines a unique name which is used to instantiate the object in ScenarioEditor.
- `<Geometry>` (`length` / `width` / `height`) define the dimension of the object.

Other attributes are used for future extensions.

After a restart of all components including ScenarioEditor the new objects can be used.

Note: Working directly in the *Distro* folder is possible, but not recommended due to updates to new versions of VTD.

How can I convert an object's 3d-model so that it will be loaded by v-IG?

v-IG requires objects to be provided with texture information etc. included. For this purpose, we are using a script so that e.g. OpenFlight models become available as IVE files. You may alter the following script `conv.sh` to match your purposes.

Note: the script works best if you have a ROD installation which comes with the required OSG tools. If you don't have a ROD installation then you may either purchase one ;-) or you may install the OSG tools by yourself via the OpenSceneGraph website.

Command line: `conv.sh inFile.flt outFile.ive`

```
#!/bin/tcsh
###-----------------------------------
###--- convert model to .ive format
###-----------------------------------

setenv DB_TOOL_DIR         /home/<my user's path to ROD tools>/VIGDBTools
setenv LD_LIBRARY_PATH     .:${DB_TOOL_DIR}

echo "Database generation process is starting."

set currentDir=`pwd`

setenv OSGOPTIMIZE ${DB_TOOL_DIR}/osgoptimize
setenv OSGCONV     ${DB_TOOL_DIR}/osgconv
setenv OSG_FILE_PATH /:.

$OSGOPTIMIZE --make-all-static\
             --move-geode-statesets-to-drawables\
             --remove-all-transparency\
             --remove-texture-paths all\
             --process-node-comments\
             --files\
             $1 DB2.ive

if ( -e DB2.ive ) then
        echo "info: Step 1 succeeded..."
```

```
else
        echo "error: Step 1  failed..."
        exit
endif

$OSGOPTIMIZE --make-all-static\
             --remove-presort-groups\
             --remove-external-filename-comments\
             --flatten-static-transforms\
             --remove-external-filename-comments\
             --remove-redundant-nodes\
             --combine-adjacent-lods --share-duplicate-state --remove-redundant-nodes\
             --merge-geodes\
             --share-duplicate-state --remove-redundant-nodes\
             --merge-geometry\
             --merge-geodes\
             --remove-redundant-nodes --merge-geometry\
             --combine-adjacent-lods --share-duplicate-state --remove-redundant-nodes\
             --combine-adjacent-lods --merge-geodes --merge-geometry --remove-redundant-nodes\
             --combine-adjacent-lods --merge-geodes --merge-geometry --remove-redundant-nodes\
             --combine-adjacent-lods --merge-geodes --merge-geometry --remove-redundant-nodes\
             --files\
             DB2.ive DB3.ive

if ( -e DB3.ive ) then
        echo "info: Step 2 succeeded..."
else
        echo "error: Step 2  failed..."
        exit
endif

$OSGCONV --compressed-arb DB3.ive  DB4.ive

if ( -e DB4.ive ) then
        echo "info: "$1 "converted..."
        mv DB4.ive $2
        rm -f DB2.ive DB3.ive
else
        echo "error: Conversion  failed..."
        exit
endif
```

How can I place a general 3d-object in my scenario at run-time?

Objects which have not been included in the scenario file may be added during run-time by a series of SCP commands. Before you can use these commands, the 3d-model representing each instance of an object must be included in the configuration file (see previous two chapters).

1. Create the Object:

The object's instance requires a unique name. So make sure, you provide each instance with its own name. The `model` must be the identifier of one of the known models in the configuration files.

```
<Player name="Pylon1">
  <Create category="object" model="RdMiscPylon03-32cm"/>
</Player>
```

2. Position the Object:

For a complete set of instructions that may be used for positioning, please consult the SCP documentation. Here's an example for positioning the object along a track:

```
<Set entity="player" name="Pylon1">
  <TrackPos track="5" s="1296" lane="2" />
</Set>
```

3. Have fun!

How can I place a custom sign in my scenario at run-time?

The following description consists of two steps:

- making the sign available as an object
- placing the sign in a scenario at run-time

A) Create a sign as object for the IG and ScenarioEditor

1. Make your own sign by opening Sign40kmh.tdo in your RoadDesigner.

> *Edit the sign the usual way. It is also possible to attach several signs to one pole.*
> *Save the file with an appropriate name.*
> *Do not change the direction of the sign.*
> *Before generating the result you must remove the shadow database in the project settings.*

2. Copy the result file $VTD/Data/Projects/Current/Databases/ObjectsCommon/

> *In our example it is Sign40kmh.opt.osgb*

3. Take Sign40kmh.xml and copy it into $VTD/Data/Projects/Current/Config/Players/Objects

4. Rename the file appropriately and its contents according to your sign.

B) Insert a sign using SCP commands

5. Start the simulation. and the following SCP commands can be sent via Trigger or via the SCP Interface

- This SCP command creates the sign:

```
<Player name="Sign40kmhN"><Create category="object" type="sign" model="Sign40kmh"/></Player>
```

- This SCP command puts the sign at the required place:

```
<Set entity="player" name="Sign40kmhN"> <PosInertial hDeg="0" x="1637.3" y="855.4" z="0.65" rDeg="0" pDeg="0"/> </Set>
```

> *After these 2 commands your sign is visible.*

- After the following command the sign is also known to traffic and sensors:

```
<SimCtrl>
  <ModifyLayout>
     <Add>
        <Anchor>
          <PosInertial x="1637.3" y="855.4" z="0.0"/>
        </Anchor>
        <signal id="1" name="Sign40kmhL" dynamic="no" orientation="-" country="OpenDRIVE" type="274" subtype="54" value="
     </Add>
  </ModifyLayout>
</SimCtrl>
```

> *The orientation relates to the nearest track.*

Driver Behavior

How can I request control commands from a VTD driver for my own vehicle?

Implementing your own driver which acts correctly within a road network (i.e. in relation to traffic rules, signs, signals and other players) may be a challenging task. A quite sophisticated driver model has been developed for the traffic vehicles which are controlled by VTD and you may use this driver for your own vehicle, too. All you have to do is to issue the SCP message

```
<Player name="Ego"><Driver ctrlLatLong="ghostdriver" /></Player>
```

after receiving via SCP the RUN command by the simulation (or after receiving the first OBJECT_STATE packages via RDB). **Note:** the name `Ego` in the above command should be replaced with the name of your player. After the command has been executed, you will receive packages of type `RDB_PKG_ID_DRIVER_CTRL` with at least the following contents filled with correct data:

```
    playerId          ID of the player (check to identify your player)
    steeringTgt       desired steering angle on front wheels
    steeringSpeed     steering speed at front wheels
    accelTgt          desired acceleration (may be numerically very high)
    flags             turn indicator or similar may be set
    validityFlags     at least the flags RDB_DRIVER_INPUT_VALIDITY_TGT_STEERING and RDB_DRIVER_INPUT_VALIDITY_TGT_ACCEL will b
```

all other data of the package will be invalid unless the corresponding validity flags are set. The target acceleration and steering are typically translated within the own vehicle's dynamics simulation into throttle, brake and steering wheel positions within the applicable physical limits. So whenever the VTD driver has some wish for an extreme value, it need not necessarily be possible to fulfill this wish! The vehicle dynamics should just provide what is possible within its constraints.

Requesting Pedal Positions

If your vehicle dynamics does not support the targets, then you may also use estimated(!) throttle and brake pedal positions as well as the steering wheel angle. These may be activated by the following command (in addition to the one given above where you request the commands in general)

```
<Player name="Ego"><Driver sendPedals="true" brakePedalScale="1.0" throttlePedalScale="1.0" /></Player>
```

Important note: be aware that our driver does not know details of your vehicle dynamics, so the inputs via pedals and steering wheel may be far off the actually required values. It is highly recommended to use the targets instead of the pedals! You may tweak the pedals a using the additional attributes *throttlePedalScale*, *brakePedalScale*, *steeringWheelScale* and *pedalSensitivity*. **Setting *brakePedalScale* is mandatory!**

Selective Control

The lateral and longitudinal controls of the driver may be selectively turned off (e.g. if you implement your own, partial model). In order to switch the relevant controls, please provide any of the following commands:

```
CONTROL OFF:
<Player name="Ego"><Driver ctrlLat="false" /></Player>
<Player name="Ego"><Driver ctrlLong="false" /></Player>
```

```
CONTROL ON:
<Player name="Ego"><Driver ctrlLat="true" /></Player>
<Player name="Ego"><Driver ctrlLong="true" /></Player>
```

Starting with VTD 2.0, *ctrlLat* and *ctrlLong* may also be set to

- `default` (same as `true`, i.e. use whatever source is available)
- `ghostdriver` (force use of ghostdriver's inputs, e.g. for overwriting mockup inputs)
- `off` (same as `false`: invalidate the corresponding axis)

Steering Target Information

For receiving the steering target position of the driver in world coordinates, request a **ContactPoint** via SCP with type *steering target* and user defined *id*. The answer will be continuous packages of type `RDB_CONTACT_POINT` distinguishable by the specified id.

How are the normalized driver parameters linked to physical values?

see VTD Driver Characteristics

How can I realize an "automated driving" simulation?

By "automated driving" we mean that the internal driver commands are (temporarily) overwritten by a longitudinal or lateral control mechanism. Adaptive Cruise Control is a very simplistic example for this sort of task. In VTD, the original driver commands are either generated by the mockup (i.e. physical steering wheel and pedals) or by a driver model which is running within the traffic simulation (i.e. ghostdriver). For activating the latter one, please see the previous paragraph.

Both sorts of drivers generate `RDB_DRIVER_CTRL_t` messages which are transmitted to the vehicle dynamics simulation. In a period of "automated driving" you will have to overwrite these commands - either in parts or completely. In order to perform this, just send `RDB_DRIVER_CTRL_t` messages to the taskControl via RDB and set not only the validity flags for the controls that you want to override but also the validiy flag `RDB_DRIVER_INPUT_VALIDITY_ADD_ON`. This will cause the taskControl to overwrite the corresponding axis (longitudinal, lateral or both) with the `ADD_ON` commands, i.e. these have higher priority. The overwriting will be performed in each simulation step for which an `ADD_ON`-command has been received (i.e. you must send your commands at least with the frequency of the simulation). Once you stop sending `ADD_ON`-commands, the original driver will take over again. Note: depending on how much you are off the original driver's intentions upon terminating the overwrite commands, the transition may be more or (most probably) less smooth.

The following archive contains a full implementation of a dummy driver that operates the acceleration pedal with a sinusoidal function and applies a constant steering angle. Note: it's an example only and will have to be adapted considerably for your own purposes.

package: dummyDriver.20170601.tgz

It consists of the following files:

```
DummyDriver/
DummyDriver/inc/
DummyDriver/inc/CommonQmakeDefs.pro
DummyDriver/Communication/
DummyDriver/Communication/RDBClientSample/
DummyDriver/Communication/RDBClientSample/MyDriver.cpp
DummyDriver/Communication/RDBClientSample/compileMyDriver.sh
DummyDriver/Communication/Common/
DummyDriver/Communication/Common/RDBHandler.cc
DummyDriver/Communication/Common/RDBHandler.hh
DummyDriver/Communication/Common/viRDBIcd.h
```

For compilation, just cd into `DummyDriver/Communication/RDBClientSample/` and execute `./compileMyDriver.sh`. You will receive an executable *myDriver* in the same directory. As long as you execute it, it will take over control of the vehicle. If you stop, the VIRES driver will take over again.

Adapt driver behavior to vehicle type

The ScenarioEditor allows creating own drivers with special behavior, e.g. a small lane change dynamic coefficient (lateral speed for lane changes). This driver can now be used on every kind of vehicle and will behave the same, independently which type of vehicle it is applied to.
In reality the behavior of a driver sitting on a truck will not be the same as on a motorbike. Following the previous example the driver on the motorbike has a larger lane change dynamic coefficient.

During vehicle creation using the ScenarioEditor (1.6+) or the appropriate SCP command (GhostDriver 1.6+) one can automatically adapt the behavior of the driver to the type of vehicle.
On checking the checkbox "Adapt to Vehicle Type" in ScenarioEditor or setting "adaptDriverToVehicleType" to true in the SCP command, the normalized coefficients of the driver are adapted as follows:

	Adaptation for		
Adapted parameter	**trucks and buses**	**motorbikes**	**others**
desiredSpeed (desired longitudinal speed)	10% slower	10% faster	none
distanceKeeping (the distance to the vehicle in front)	20% more distance keeping	none	none
laneKeeping oscillation frequency (measure for driver's ability to stay in the center of the lane)	20% slower oscillation	10% faster oscillation	none
laneKeeping oscillation amplitude (measure for driver's ability to stay in the center of the lane)	10% less oscillation amplitude	10% more oscillation amplitude	none
speedKeeping oscillation frequency (measure for driver's variance in holding a given speed)	20% slower oscillation	10% faster oscillation	none

speedKeeping oscillation amplitude (measure for driver's variance in holding a given speed)	10% less oscillation amplitude	10% more oscillation amplitude	none
laneChangeDyn (maximum lateral speed for lane changes)	10% less dynamic	10% more dynamic	none
keepRightRule (measure for driver's desire to use the outmost lanes)	20% more right keeping	none	none
urgeToOvertake (measure for driver's desire to pass slower vehicles)	20% less overtaking	10% more overtaking	none
desiredAcc (desired acceleration)	10% less desired acceleration	10% more desired acceleration	none
desiredDec (desired deceleration)	10% less desired deceleration	10% more desired deceleration	none
steeringDist (look-ahead distance for the generation of the steering input)	none	25% more steering dist	none

The normalized values can never exceed the range from 0 to 1 (if you create a driver with a distanceKeeping value of 0.99999, the adaptation for trucks will only lead to 1.0).
Values on the boundaries (0.0 or 1.0) are not adapted, because some of the boundary values are handled special (e.g. a keepRightRule of 0.0 leads to keeping the left lane on a motorway, whereas 0.01 does not).

What are the influencing factors for the desired acceleration?

The acceleration value chosen by the driver is based on several points:

- current speed
- the driver's desired speed
- desired acceleration (driver behavior)
- desired deceleration (driver behavior)
- braking performance, primarily distance keeping to:
  - surrounding objects, vehicles and pedestrians, depending on
    - road type
    - overtaking pahse
  - traffic lights, depending on their phase and if available, distance to waiting line
  - junctions, depending on applicable traffic rules

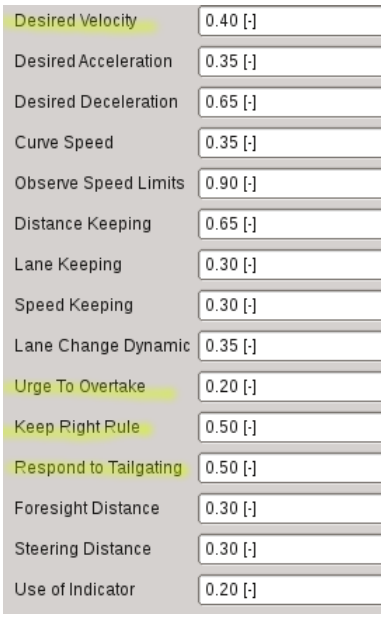
Lane Change Model

General

The following parameters of the VTD driver model influence the decision of the driver whether to perform a lane change

- desired velocity
- urge to overtake
- keep right rule
- respond to tailgating

These are made available in the ScenarioEditor GUI under "Properties->Driver Definitions"



When performing a lane change, the current offset from lane center of the vehicle in the "departure" lane is taken into account and the offset in the "arrival" lane will be predicted based on the time it takes for the lane change to be completed and based on the lane keeping dynamics.

The decision for lane change is based on a couple of criteria:

- overtaking
  - is there a reason to start overtaking?
    - is own speed sufficient?
    - is there a general urge for overtaking (driver parameter)?
    - is there an obstacle (slower vehicle) in current lane?
  - is it allowed to overtake?
    - do road marks permit overtaking?
  - is it safe to overtake?
    - is there oncoming traffic (from front or rear)?
- switch to side lane
  - is there a reason to switch to the side lane?
    - is own speed sufficient?
    - is a vehicle tailgating the driver's vehicle?
    - is the urge to keep right sufficiently high?
  - is it safe to switch lane?
    - is there oncoming traffic (from front or rear)?

Is there a relationship between the distanceKeeping coefficient and the realization of autonomous lane changes on multi-lane roads?

In deciding whether a lane change can be performed or not (precondition is a reason for a lane change, e.g. a slower vehicle in front), the hypothetical acceleration request of the "overtaken" driver and, at presence of a front vehicle, the hypothetical acceleration request of the own driver is calculated.

Simplified:

- If there is a front vehicle, we ask the own driver, what he would want to do regarding acceleration, if the lane change would already be done. If this hypothetical acceleration request falls below a limit*, the lane change is not performed (e.g. if the driver wishes to perform a full brake). If there is no front vehicle, there is no reason not to change lanes.
- The same question is asked to the potentially "overtaken" driver. If he would want to decelerate too much*, the lane change is not performed.

Both potential acceleration wishes are influenced by the distanceKeeping coefficient.
-> If the future follower has a small distance need, the lane change will be performed short in front of him.

*This limit depends on the own urgeToOvertake-coefficient. That is, the larger the value the more braking will be expected by the drivers.

How does urgeToOvertake influence overtaking behavior?

The urgeToOvertake parameter is a measure for driver's desire to pass slower vehicles.

One of the overtaking conditions is the difference of current speed (e.g. 80 km/h) to desiredSpeed (e.g. 100 km/h).
The desiredSpeed is multiplied by a factor (call it k) based on urgeToOvertake. If urgeToOvertake is 1.0 also k is 1.0 (urgeToOvertake = 0.5 does not lead to k = 0.5 but to 0.85!).
The result is compared with current speed, and if current speed is less, one of the overtaking conditions is met.

That means, using the example values current speed = 80 km/h and desiredSpeed = 100 km/h:

- urgeToOvertake = 0.00: => no overtaking
- urgeToOvertake = 0.25 => k = 0.60: current speed is more than 60 km/h => no overtaking
- urgeToOvertake = 0.50 => k = 0.85: current speed is less than 85 km/h => overtaking (if other conditions are met)
- urgeToOvertake = 1.00 => k = 1.00: current speed is less than 100 km/h => overtaking (if other conditions are met)

Configuration of "track-controlled" driver mode

Note: this chapter only applies to certain installations; if you don't know what it's about, the it is (most probably) not intended for you ;-)

For players which are being controlled by messages of type `RDB_PKG_ID_CUSTOM_OBJECT_CTRL_TRACK` the preview that is applied for path planning (lateral control) may be set using the command

```
<Player name="...."><Control trackCtrlPreview="1.5"/></Player>
```

with *trackCtrlPreview* being the preview along the track in `[s]`.

If you want to adjust the preview for longitudinal (speed) control, you have to use the following command:

```
<Player name="...."><Control trackCtrlPreviewLong="0.5"/></Player>
```

How to produce a sudden traffic jam

To generate a traffic jam at a location previously unknown with low lead time, it is best practice to do the following:

1. Create all players used for the traffic jam with the scenario editor at positions where they can safely be parked (e.g. on an auxiliary track)
   The SCP command to create players during runtime could in this case also be used if the system is fast enough to prevent frame delays.
2. When the condition to create the traffic jam is fulfilled, the players can be moved to the desired positions using the following SCP-commands of the *set* group.
   - *PathS*, if there is a valid route
   - *PosInertial*, if the world coordinates are known
   - *PosRelative* or *PosRelativeRoad*, if the position shall be related to the ownship
   - *TrackPos*, if the road coordinates are known

How can I show a perspective map in the ScenarioEditor?

Just press `@` in the ScenarioEditor to toggle between birdview and perspective map.

How do I configure a custom trajectory for a player?

Via ScenarioEditor:

- open the ScenarioEditor
- define a path shape for your player via Create PathShape dialog
- assign your player to the path shape via position type path shape (tab "Initialization" in player properties)
- determine start and end s-coordinate of the path shape; this is the range that will be used for following the path shape

Instead of setting the position type of the player to path shape, an action can be used. By entering this action, the player is searching for the path shape and is following this as soon as it is found:

```
<Traffic>
  <ActionNominalTrajectory command="add" radius="15" endAction="emd" name="pathShape" actor="Own"/>
</Traffic>
```

Via SCP during the simulation:

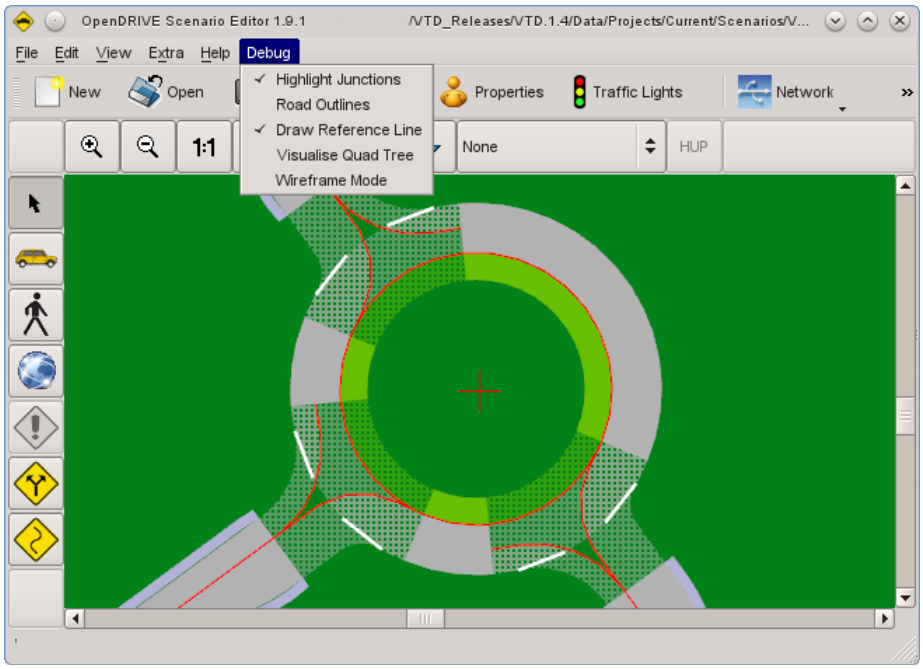
The path shape can also be created at run time via SCP. Following command will create a path shape relative to a player:

```
<Path name="pathShape" type="polyline">
  <Waypoint><PosRelative player="Own" dx="10" dy="0"dz="0"/></Waypoint>
  <Waypoint><PosRelative player="Own" dx="15" dy="2.0"dz="0"/></Waypoint>
  <Waypoint><PosRelative player="Own" dx="20" dy="0"dz="0"/></Waypoint>
</Path>
```

Following the path shape can be achieved with ActionNominalTrajectory described above.
See SCP documentation for further information to the SCP commands.

Display additional road information in ScenarioEditor

ScenarioEditor provides a "Debug-Mode", which is activated using the '-d' command line option. It allows displaying junctions (semi-transparent), track reference lines (red) and some additional information used for ScenarioEditor development:



Within an ordinary VTD installation, ScenarioEditor is started via the file `taskSettings.cfg` in `Data/Setups/Common/Config/Simulation`. The file contains an area for the ScenarioEditor and defines a variable called `CMD_ARGS`. Just add the '-d' option to the `CMD_ARGS`:

```
  set CMD_ARGS      = ( -d -f scenarioEditor.xml.ini $EXTRA_ARGS )
```

How can I debug my trajectory or how can I see my steering path?

First you have to enable sending the steering path via RDB:

```
<Query entity="player" name="Ego">
  <SteeringPath noPoints="10" offset="2.7" dt="0.3" minDeltaDistance="0.6"/>
</Query>
```

Then you enable debug trajectories:

```
<TaskControl><Debug trajectories="true"/></TaskControl>
```

How do I configure a looping path for a player?

- open the ScenarioEditor
- define the waypoint path for your player
- assign your player to the path (tab "Initialization" in player properties)
- determine start and end s-coordinate of the path; this is the range that will be used for looping
- set the `end action` to `loop`

If you want your vehicle to loop in a different range than the one determined by the start position of your vehicle, add an action like the following at the end of the path

```
<Set entity="player" name="Own">
  <PathS name="myPath" s="0" endAction="loop" lane="-1"/>
</Set>
```

Where is the default eyepoint location of a player?

The default eyepoint of a player (i.e. vehicle) is at the following position in player-coordinates:

`x = 1.1m, y = 0.3m, z = 1.1`

Traffic Lights and Traffic Signs

How are the state masks defined?

The "usual" traffic lights (i.e. three light bulbs - red / yellow / green) may be controlled by providing a statemask. This statemask is a 32bit integer value resulting in an eight digit hex value. The states of individual light bulbs are controlled by the half-bytes from left to right. Each light bulb may - theoretically - be controlled individually in color with the following coding

- 0 = off
- 1 = red
- 2 = yellow
- 3 = green

Due to the modelling of the current traffic lights in VTD, not all of this flexibility is available (i.e. the colors are typically not adjustable). So, basically, the state mask just resembles which lights are on and off. Here are some examples:

- Red traffic light: stateMask = 0x10000000
- Yellow traffic light: stateMask = 0x02000000
- Green traffic light: stateMask = 0x00300000

Collision of Players

The collision of players can be handled in two ways:

- detection only
- detection and action

For the former case, please check the documentation about the "CrashSensor" in this wiki. For the latter one, there is a means in the traffic module to compute a crash impulse and apply it to the traffic vehicles. At the same time, an SCP message like the following will be issued

```
<Player name="Own"><Collision omega="1.92958" vFinalLocalX="3.73906" vFinalLocalY="-1.71303" vFinalLocalZ="0" /></Player>
```

In order to activate the collision detection in the traffic module, use the command

```
<Traffic><Collision enable="true"/></Traffic>
```

For additional attributes, see the SCP documentation.

How can I create Triggers/Actions executed by each player of a group reaching the trigger condition

One can create Trigger/Action combinations for groups of players (e.g. all pulk players), which are only executed for single players. The groups are called "selections", there are predefined ones, but you also can create your own selections.
E.g. you can place a pulk traffic selection trigger somewhere, which executes an ActionSpeedChange. This ActionSpeedChange will be executed for every pulk player reaching that position.

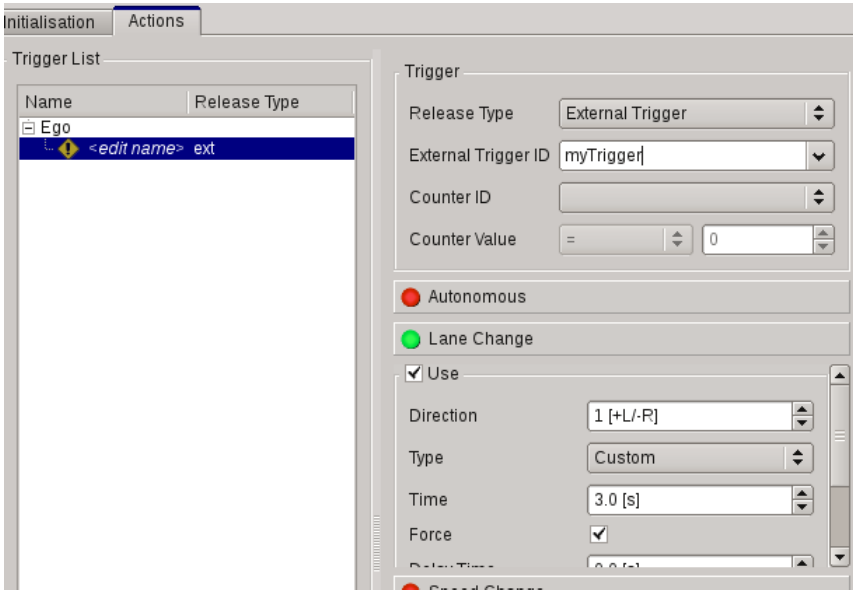
1. double click on one of your players, e.g. "Ego"
2. create a new Trigger
3. move the Trigger somewhere you'll find it in the following steps
4. right click on that Trigger and select "Transfer to" -> e.g. "ALL_SWARM_PLAYERS"
5. click on the Trigger in the scenario
6. change "Pivot" to "$owner"
7. create the Action and set the "Number of Executions" to "infinite"
8. if you want to use a SCP-Action with an "actor", set the "actor" to "$owner"

How do I use an external trigger

External triggers may be used to define an action within the scenario but trigger its execution by an external (SCP) command.

Example:

The following Lane Change action is executed upon arrival of the external trigger named "myTrigger" (you may use any name you like).



In order to execute the trigger during runtime, just send the SCP command

```
<Traffic><Trigger id="myTrigger" active="true"/></Traffic>
```

Note: for repeated activation of a trigger, make sure it is set to "deactivated" in-between. The command for the above trigger would be `<Traffic><Trigger id="myTrigger" active="false"/></Traffic>`

Scenario Editor

Launching the ScenarioEditor from command line

Usually, the ScenarioEditor will be launched from the VTD GUI using the mechanisms of the SimServer. If you want to start it from command line, you may do so using the following script ("startEditor.sh").

```
#!/bin/bash

export VTD_ROOT="../../../"
export VI_RUNTIME_DIR="$VTD_ROOT/Runtime"
export VI_CORE_DIR="$VI_RUNTIME_DIR/Core"
export VI_DATA_DIR="$VTD_ROOT/Data"
export VI_SETUP_DIR="$VI_DATA_DIR/Setups"
export VI_PROJECT_DIR="$VI_DATA_DIR/Projects"
export VI_DISTRO_DIR="$VI_DATA_DIR/Distros"
export VI_CURRENT_SETUP="$VI_SETUP_DIR/Current"
export VI_CURRENT_PROJECT="$VI_PROJECT_DIR/Current"
export VI_CURRENT_DISTRO="$VI_DISTRO_DIR/Current"

export VI_FILE_PATH=".:$VI_CURRENT_PROJECT:$VI_CURRENT_SETUP:$VI_DATA_DIR/Setups/Common:$VI_CURRENT_DISTRO:$VI_DATA_DIR"
export VI_FILE_SUB_PATH=".:Databases:Scenarios:Scripts:Recordings:Config/TaskControl:Config/ImageGenerator"
export VI_FILE_SUB_PATH="$VI_FILE_SUB_PATH:Config/ScenarioEditor:Config/SensorManager:Config/DynamicsManager"
export VI_FILE_SUB_PATH="$VI_FILE_SUB_PATH:Config/ModuleManager:Config/Players:Plugins/SensorManager:Plugins/DynamicsManager:P

./scenarioEditor -noSCP -f $VI_CURRENT_SETUP/Config/ScenarioEditor/scenarioEditor.xml.ini $*
```

Please do the following:

1. Save above script as `startEditor.sh` in "VTD.2.1/Runtime/Core/ScenarioEditor"
2. Make it executable ("chmod a+x startEditor.sh")
3. Launch, for example, "./startEditor.sh ../../../Data/Projects/Current/Scenarios/crossing8Demo.xml"
4. That's it.

Data and Video Recording

Data Recording

How can I create a data recording?

VTD-recordings (*.dat*-files, CAUTION: not to be confused with ADTF-Dat-files!) are per default located in the sub-directory *Recordings* within the current project directory. In order to create a new recording, please perform the following steps:

1. In the GUI, select the *FILES* branch from the tree view in the *Project Configuration* window
2. Click with the right mouse button on the entry *Recordings* in the center tree view
3. Select "New Recording"
4. Specify a file name for the recording. Now, the red *Record*-button in the action toolbar will be available
5. Press the red *Record* button to start the recording (it may also be activated before starting the simulation)

6. Press the red *Record* button again to stop the recording

If the record button remains pressed, then a new recording will be started upon each start of the simulation. If the *overwrite* option has been selected in the *Recording* panel, then only the latest run will be recorded; otherwise, a new recording with a unique number will be created for each run.

Replaying a recording

Status Display

During playback, there will be an additional line in the IG showing the playback status.

You may disable this line by providing the corresponding entry in the TC configuration or by sending the same SCP command online.

```
<TaskControl>
  <RecPlay ... showInfo="false" ... />
  :
</TaskControl>
```

Recording to a buffer instead of a file

availability: VTD 2.0.2+

Instead of recording data to a file, an internal buffer may be used. It will be available for instantaneous replay. The buffer is a ringbuffer holding the tbd. last seconds of simulation data. The configuration is achieved as follows:

```
<Record>
  <Config writeMem="true" recBufferSize="20"/>
  <Start/>
</Record>
```

The attribute *writeMem* activates buffer recording and *recBufferSize* determines the size of the buffer in `[s]`.

The recording is stopped by

```
<Record> <Stop/> </Record>
```

For replaying the buffer, use the following command sequence:

```
<Replay> <Start/> </Replay>
```

The playback is stopped by

```
<Replay> <Stop/> </Replay>
```

The buffer content may also be **dumped to a file** using the following syntax:

```
<Record>
  <File path="/home/vtd/VTD.2.0/Data/Projects/Current/Recordings" name="testRec.dat" overwrite="true"/>
  <Dump/>
</Record>
```

I have a data recording with pedestrians but I don't have a pedestrian license. What will happen?

If you don't have a valid license for pedestrians (here: DI-Guy) you may either purchase one ;-) or you will just see simplified pedestrian during the replay. These simplified pedestrian will not show any nice animation but it will be moving along the recorded positions nevertheless. Once you have a pedestrian license, the high fidelity models will be used again.

Video Recording

There are two ways to create videos. Starting with VTD 1.1, it is possible to have videos directly, i.e. live, written by the image generator. In previous versions, videos could only be generated from data recordings in a sort of post-processing. Both methods will be described here.

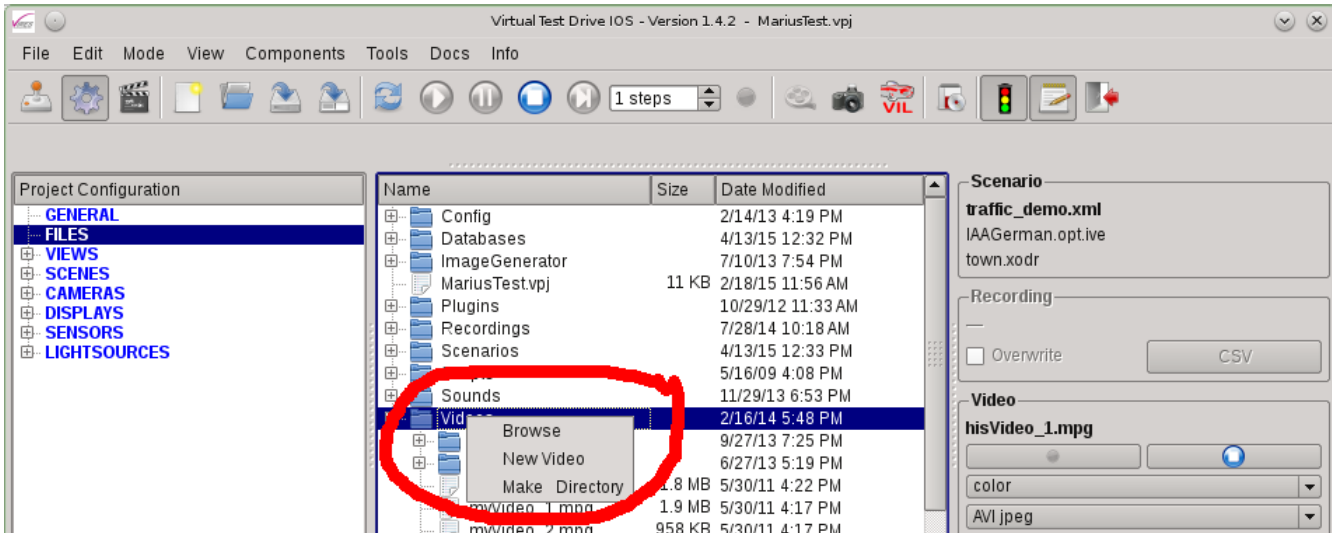
Live video recording

Note: Live video recording can generate MP4 files and AVI files (AVI for VTD 2.0.2+) only. For other video formats, please follow the instructions below for the "offline" video recording.

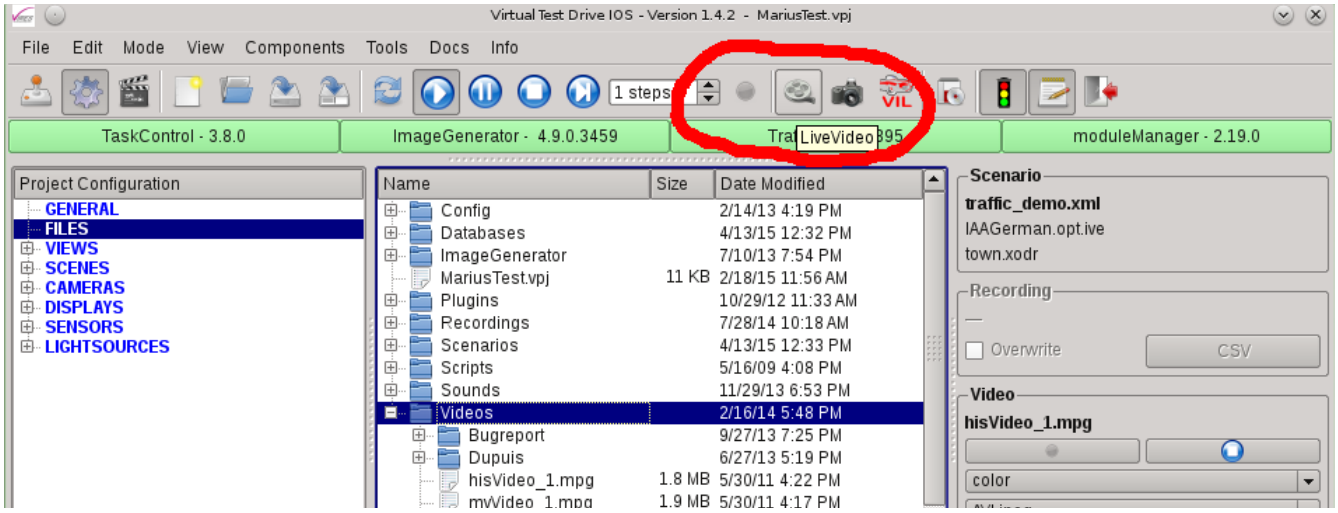
Recording via GUI

Per default (VTD 1.4+), the following method using the GUI should work:

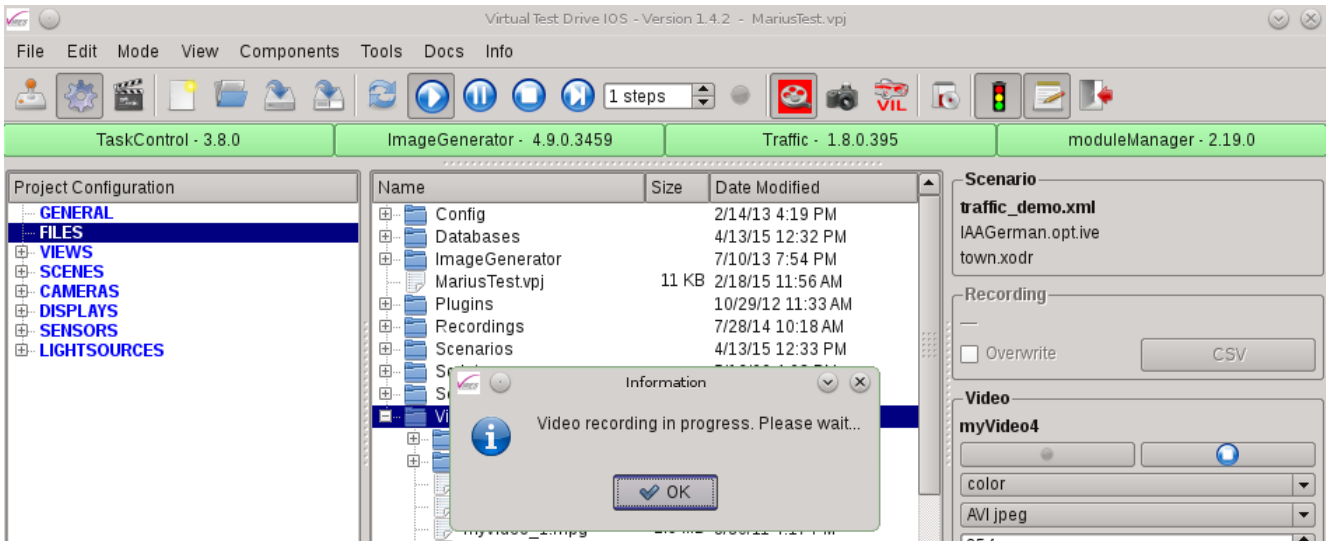
1) Create a new file (in sub-dir `Videos`) where the resulting video shall be stored



2) While the simulation is running, press the "live video" button:



3) The live video recording will start and a pop-up message will appear; the live recording button will turn red



4) In order to stop the video recording, press the "live video" button again

5) That should be it

Alternative Command Sequences

For live video recording, the *Image Generator (IG)* must be configured accordingly. Please edit the file *Data/Setups/Common/Config/ImageGenerator/CommonIGConfig.xml* (formerly: *Data/Setups/Current/Config/ImageGenerator/IGbase.xml*) so that it contains one of the two lines

MP4:

```
<FramebufferReader name="MyFBReader">
    <VideoEncoder filename="/tmp/video.mpg" frameDrops="1" kbps="4000" cpuAffinity="2"/>
</FramebufferReader>
```

AVI:

```
<FramebufferReader name="MyFBReader" >
  <VideoEncoder filename="/tmp/foo.avi" frameDrops="2" cpuAffinity="2" kbps="6000"/>
</FramebufferReader>
```

You may change the parameters accordingly, but note that the name *MyFBReader* **MUST NOT** be changed!

The TaskControl should run in sync with the IG, otherwise frame drops will occur and the video timing will be distorted. Please perform the following settings in *Data/Setups/Current/Config/TaskControl/taskControl.xml*:

```
<Sync source="extern"
      realTime="true" ..../>
```

or

```
<Sync source="extern"
      realTime="false"
      frameTimeMs="20"/>
```

During the simulation, the video-recording may be operated in two ways

1. By pressing the key "t" while having the input focus on the IG output window
    1. The video will be recorded using the name specified in *IGbase.xml*, followed by a unique number
    2. Recording will be stopped by pressing "t" again
2. By sending SCP commands:
    1. Configure the video output:

```
<Video> <Output path="/tmp/" name="myVideo" /> </Video>
```

    2. Start the live recording

```
<Video> <Start live="true"/> </Video>
```

    3. Stop the live recording

```
<Video> <Stop/> </Video>
```

"Offline" video recording

Videos may be generated from recordings upon replay. This method also provides a means to generate videos in formats other than MP4. Please perform the following steps, using the GUI:

1. Record your scenario (see FAQ)
2. Select the *Replay* mode in the GUI
3. Select the *FILES* branch from the tree view in the *Project Configuration* window
4. Select the appropriate record file from the list of recordings
5. Click with the right mouse button on the entry *Videos* in the center tree view
6. Select *New Video*
7. Specify the name of the video file
8. Specify the video format and other parameters in the far right *Video* panel of the GUI
9. Press the red record button within this panel
10. The video recording and conversion will start and may take a while; now it's time for a coffee break
11. A pop-up window will appear after the video conversion is finished.
12. The video will be located in the *Videos* directory of your current project

Important Notes:

- Video recording and conversion are two separate steps. That the former is running does not mean that the latter will also do so (at least if you haven't tried before). Therefore, after the video recording, please watch the messages in the *TaskControl* window and look out for warnings / errors due to missing software packages.
- Length and height of the IG window (video image) must each be a multiple of 16 so that all formats can be generated. Problems will be reported in the *TaskControl* window.
- The ratio of the simulation frame rate and the video frame rate should be an integer number. Otherwise, a time scaling or jitter may occur in the resulting video.

Yet Another Note: The offline video conversion requires (temporarily) considerable space on the hard disk. Using the default settings of VTD, the data amount may be up to 35 to 40MB per second of the video. If the default location which is included in the default filename for temporary video files (*/tmp/vidSample*) does not provide enough space for the temporary data, you may specify a more convenient location using the environment variable *VTD_TMP_VIDEO_FILE*. **Note:** with this variable you do not only set the file path but the full name of the video file including its path!

Are there any restrictions in connection with the video recording?

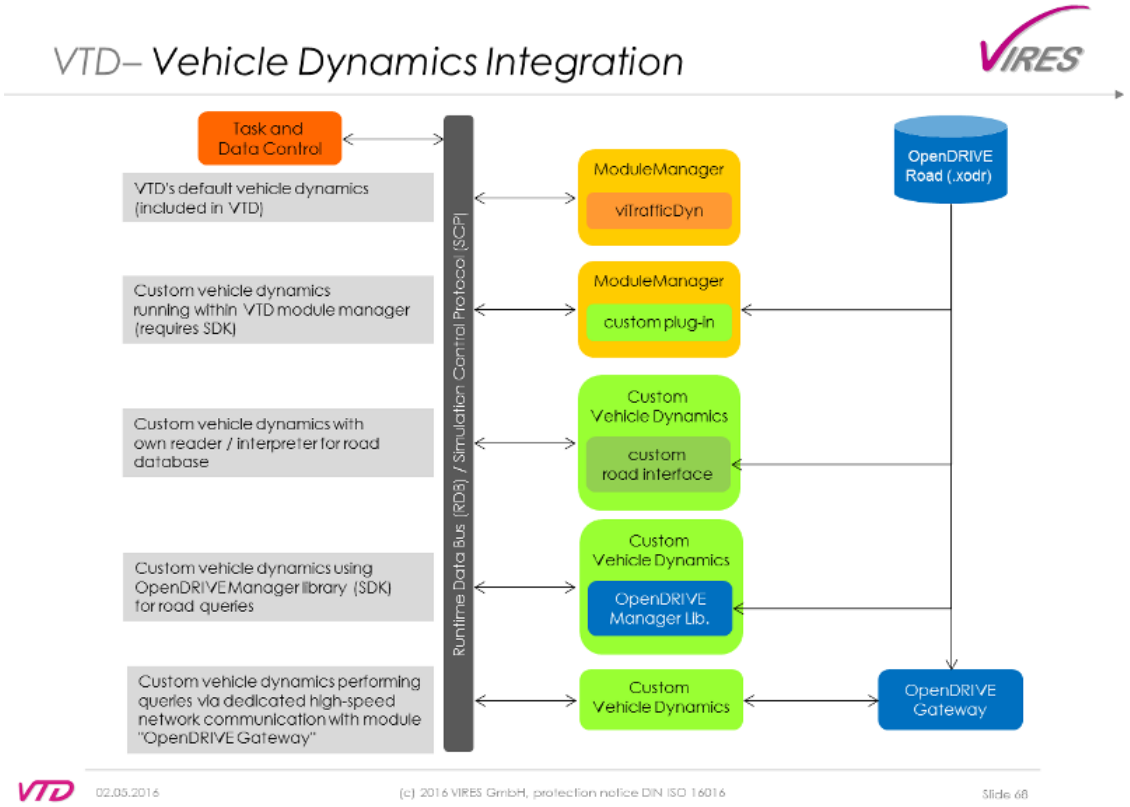
Definitely yes!

- Offline video recording may be limited by time and disk space (since raw images are stored as intermediate format)
- Codecs used by VTD may not run in some media players (e.g. Windows Media Player). We recommend *VLC* as playback tool.
- Depending on the video format, not all resolutions, frame rates etc. might be available.

Vehicle Dynamics

For the vehicle dynamics, various options are available. You may run a vehicle with the preparation panel (this will control the selected external vehicle defined in a scenario and provides no actual dynamics), you may run it via a plug-in of the ModuleManager or you may control it via RDB with a 3rd party vehicle dynamics simulation.

The following slide gives a quick overview of the different ways of connecting a vehicle dynamics simulation to VTD:



VIRES Vehicle Dynamics in the ModuleManager

There are two Vehicle Dynamic Models delivered with VTD. The simpler version (VIRES Simple Dynamics) can be used as external dynamics module and is also used to compute the dynamic behavior of internal vehicles. The more complex version (VIRES Complex Dynamics) is only available as Module Manager Plugin.

VIRES Simple Dynamics

The VIRES Traffic Dynamics is used for internal vehicles (internal cars and swarm traffic) simulated by the traffic component of VTD (ghostdriver). Additionally it can be used as dynamics model for one or more external car(s). For this purpose the Simple Dynamics is available as Module Manager Plugin (libModuleTrafficDyn.so). Per default the first external vehicle in a scenario is using this Plugin for computing dynamic behavior of the external vehicle.

A document describing our simplified vehicle dynamics is given here: VTD Vehicle Dynamics

VIRES Complex Dynamics

Unlike the Simple Dynamics, the Complex Dynamics uses a physics engine to compute the movement of the vehicle. The engine that is used is the Bullet Physics Library. Bullet already provides a so called RayCast Vehicle Model. The VIRES Complex Dynamics is based on that RayCast Vehicle with several modifications.
The main difference to the Simple model is, that there are four contact points, one per wheel, which allow a realistic driving on curbstones, speed bumps, etc. Each wheel is attached to a suspension with adjustable behavior for spring and damper. The main modification of the Bullet RayCast Vehicle is the ability to emit rays into the OpenDRIVE/OpenCRG world and therefore allow driving on an OpenDRIVE/OpenCRG road network.
Powertrain and gearbox are not part of the Bullet Vehicle, they are improved versions of the Simple Dynamics powertrain and gearbox models and allow a lot more parameters to be modified. Result of the power train calculation are forces, which are applied to the physical model of the vehicle in the Bullet world.

Main intention for the development of the Complex Dynamics was a realistic behavior regarding the driving on OpenCRG objects like cobblestone. It is not a validated vehicle dynamics model for any specific car!

The following parameters are influencing the behavior of the Dynamics Models:

Parameter Name	Default Value	Simple	Complex	Description	Config File	SCP
maxSpeed	60.0 m/s	X		upper speed limit of the vehicle, n/a for complex dynamics	X	X
maxSteering	0.5 rad	X	X	maximum steering angle (+/-) on the front wheels	X	X
mass	1400.0 kg	X	X	overall vehicle mass, used for powertrain and physics simulation	X	X
maxTorque	240.0 Nm		X	maximum engine torque, only applicable for complex dynamics	X	X
enginePower	75.0 kW	X	X	maximum engine power, used for powertrain calculation in simple and complex dynamics	X	X

overallEfficiency	75 %	X		overall efficiency factor, used in simple model for max. acceleration calculation	X	X
wheelDiameter	0.6 m	X	X	diameter of wheels	X	X
wheelDiameterDynamic	0.62 m			unused	X	X
rollingResistance	0.013	X		wheel rolling resistance, influencing current driving resistance in simple dynamics	X	X
wheelSkewStiffness	12.0			unused	X	X
airDragCoefficient	0.35	X	X	air drag coefficient, supported by simple and complex model	X	X
frontSurfaceEffective	2.0 m²	X	X	effective front surface, supported by simple and complex model	X	X
maxDecel	-9.5 m/s²	X		maximum deceleration	X	X
distFront	3.5 m	X	X	vehicle dimensions, measured from vehicle reference point (rear axis, bottom, neutral)	X	X
distRear	1.0 m	X	X	vehicle dimensions, measured from vehicle reference point (rear axis, bottom, neutral)	X	X
distLeft	1.0 m	X	X	vehicle dimensions, measured from vehicle reference point (rear axis, bottom, neutral), without mirrors	X	X
distRight	1.0 m	X	X	vehicle dimensions, measured from vehicle reference point (rear axis, bottom, neutral), without mirrors	X	X
distHeight	1.5 m	X	X	vehicle dimensions, measured from vehicle reference point (rear axis, bottom, neutral)	X	X
wheelBase	2.7 m	X	X	wheel base	X	X
gradientRollAngle	-1	X		optional gradient for roll movement, set to 0 if no roll movement is desired (e.g. for trains); if set to -1, simple dynamics will determine reasonable roll movement; ignored in complex dynamics	X	X
gradientPitchAngle	-1	X		optional gradient for pitch movement, set to 0 if no pitch movement is desired (e.g. for trains); if set to -1, simple dynamics will determine reasonable pitch movement; ignored in complex dynamics	X	X
wheelDrive	wheel_drive_front	X		wheel drive type: "wheel_drive_front", "wheel_drive_rear" or "wheel_drive_all"; currently ignored by complex dynamics (complex dynamics is always front wheel driven)	X	X
minEngineRpm	1200 1/Min		X	automatic gearbox will change gear if engine RPM falls below this value; only supported by complex dynamics		X
maxEngineRpm	6500 1/Min		X	automatic gearbox will not allow RPM above this value; only supported by complex dynamics		X
maxEngineBrakeTorque	30 % of maxTorque		X	maximum engine brake torque; only supported by complex dynamics		X
shiftingTime	0.2 s		X	time required for gear change; only supported by complex dynamics		X
keepGearTime	2.0 s		X	minimum time a gear is kept; only supported by complex dynamics		X
idleRpm	800 1/Min		X	engine RPM in idle state; only supported by complex dynamics		X
creepForce	100 N		X	minimum force on wheels with gear and closed clutch; only supported by complex dynamics		X
maxDrivingBrakeForce	15000 N		X	maximum force of the driving brake; only supported by complex dynamics		X
wheelHalfWidth	0.2 m		X	half width of the wheels; only supported by complex dynamics		X
frictionSlip	2.2		X	slip friction coefficient; only supported by complex dynamics		X
suspensionStiffness	60000.0 N/m		X	suspension stiffness; only supported by complex dynamics		X
wheelsDampingCompressionFactor	0.2 Ns/m		X	damping factor for compression; only supported by complex dynamics		X
wheelsDampingRelaxationFactor	0.2 Ns/m		X	damping factor for relaxation; only supported by complex dynamics		X
rollInfluence	0.5		X	reduce roll influence, "anti-roll bar"; 0.0 = no roll influence; only supported by complex dynamics		X
steeringTorqueStart	0.3 Nm		X	minimum torque at which the steering angle starts moving; only supported by complex dynamics		X
steeringTorqueHoldMax	1.8 Nm		X	torque at which the maximum steering angle is just kept; with falling below this value, steering angle decreases again.; only supported by complex dynamics		X
steeringTorqueMax	2.0 Nm		X	maximum torque, leading to maximum steering angle; only supported by complex dynamics		X

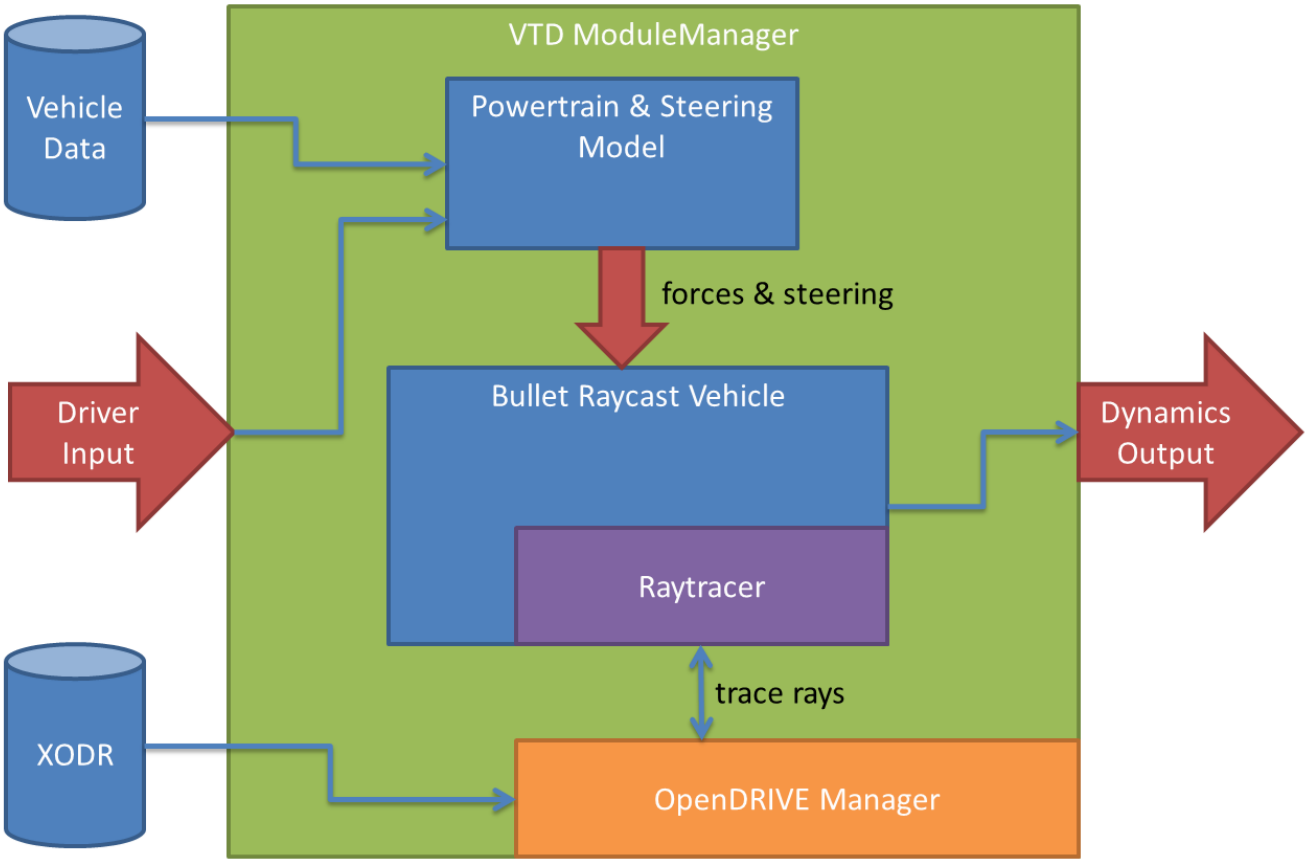
Changing parameters

The parameters previously described usually are changed via config file. Most of the parameters used for the complex model are currently only changeable by SCP protocol.
In order to change them, you have to use the following SCP-command:

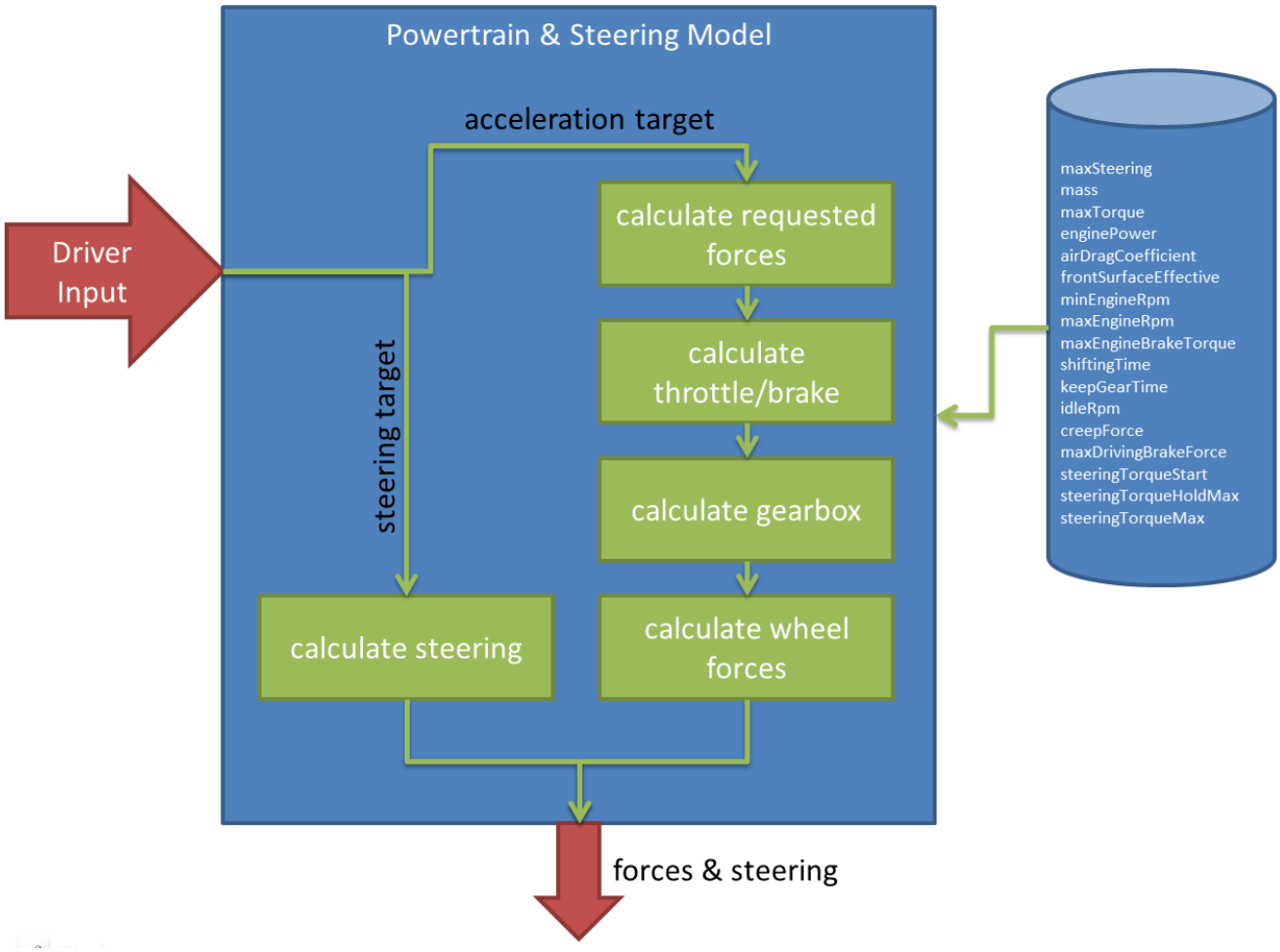
```
<Player name="Ego"><VehicleDef suspensionStiffness="30000.0"/></Player>
```

Implementation Overview

The complex model is implemented as VTD ModuleManager Plug-In. As every dynamics Plug-In it reads driver input (steering wheel and pedals or steering target and acceleration target) and creates vehicle dynamics output (position, attitude, steering angle, velocity, acceleration, suspension values, powertrain values).



The powertrain and steering model uses a set of input parameters and simulates steering wheel, throttle/brake and a gearbox. The result is a steering angle and wheel forces.

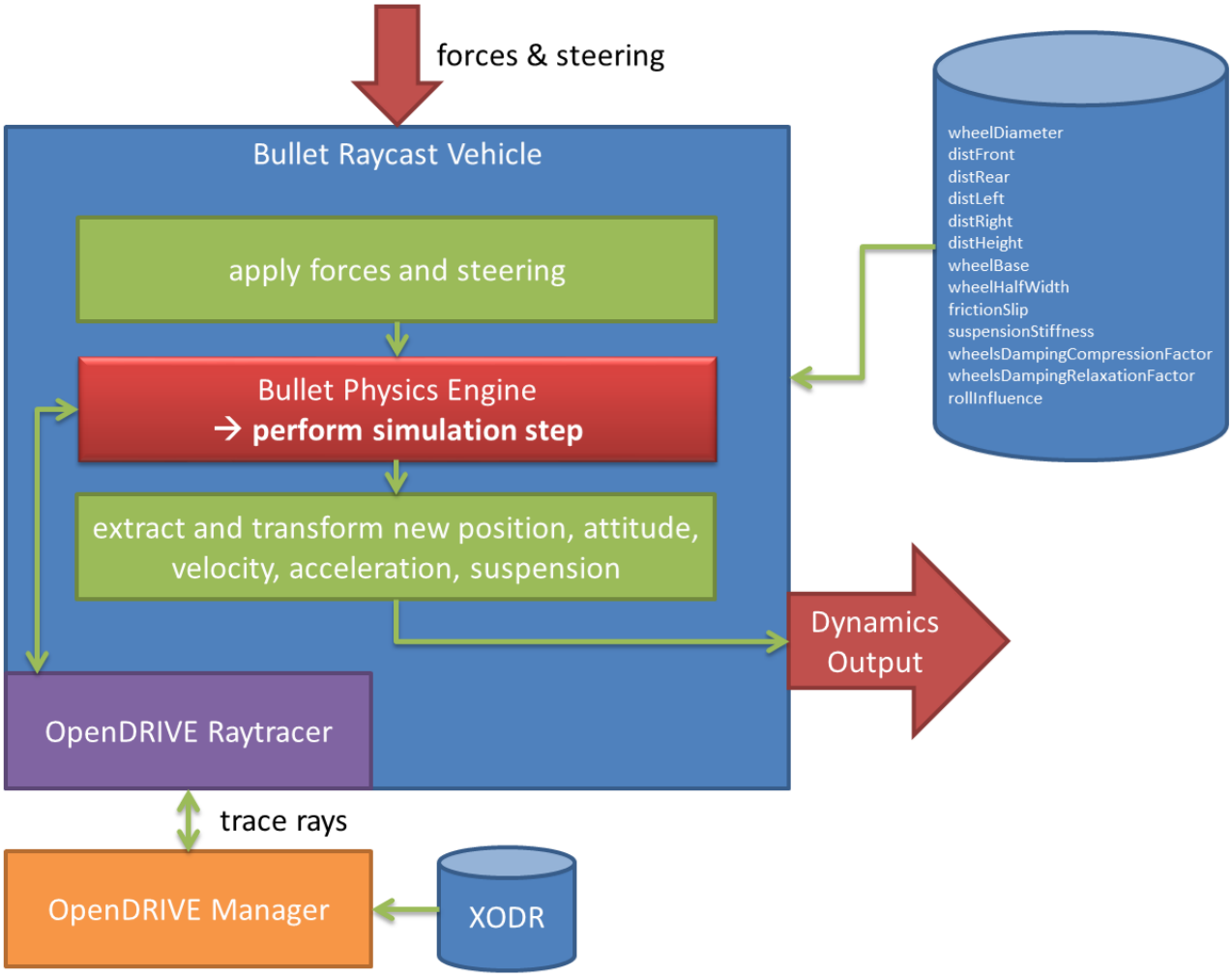


After that, forces and steering are applied to the Bullet Raycast Vehicle. The execution of one simulation step in the Bullet Physics Engine leads to four traced rays. The adapted raytracer is using OpenDRIVE methods to determine suspension length and suspension force, depending on several Bullet Vehicle parameters, like spring compression and spring relaxation damping coefficients.

Bullet Raycast Vehicle

- Casts a ray for each wheel
- Calculates suspension length and suspension force based on the ray’s intersection point
- Suspension force is applied to the chassis
- Suspension is provided by the spring force plus a damping force
- There are two coefficients for damping:
    - spring compression
    - spring relaxation

The new position, attitude, velocity, etc. is extracted and transformed back into the VTD coordinate system and afterwards written as standard vehicle dynamics output.



Truck Cabin Movement

In this section two ways will be described to move a truck cabin relative to a truck chassis.

The attached package contains two projects one using an extended version of the VTD complex dynamics the other one using RDB input.

vtd.2.1.0.addOns.truckCabinMovement.20180824.tgz

Both projects are using a split Mercedes Actros 2011 model:


v0 t00:23.367


v0 t00:00.017

General notes:

- two external vehicles are set up, one for cabin the other one for chassis
- the chassis player is the "main player", the cabin player is set invisible for traffic (otherwise the VTD driver will see the cabin as vehicle and stop)

Using the complex dynamics

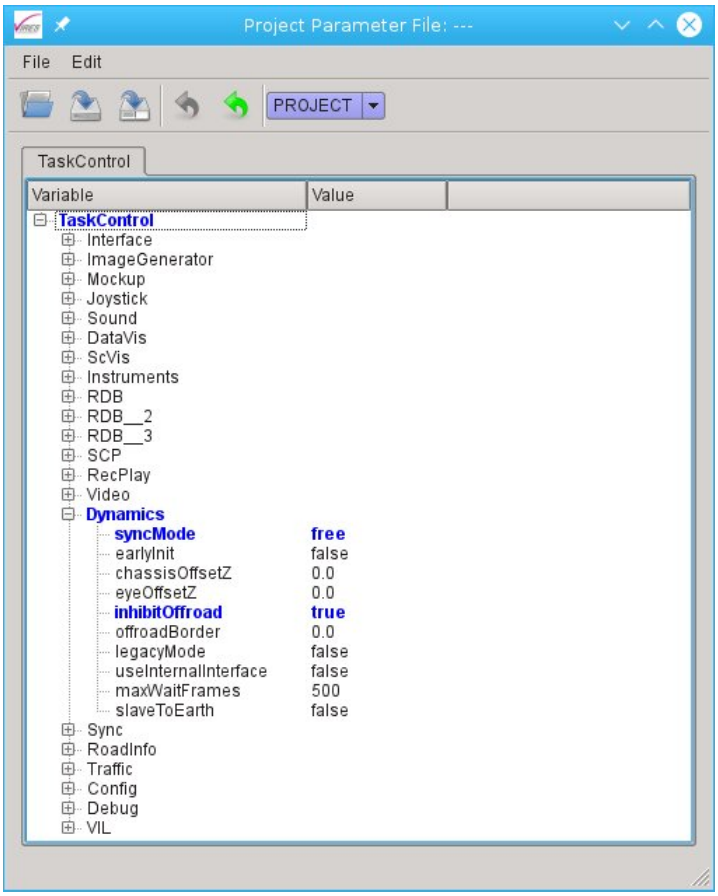
There is a prototypical implementation of visual realistic movements of a truck cabin attached to a truck chassis, using the Bullet Physics library, which is already used in the so called complex dynamics model of VTD.
The corresponding VTD project is called "TruckCabinWithModuleManagerDyn". A sample scenario is included (TruckCabinExample.xml).

Using RDB

There is a prototypical implementation of a RDB receiver/sender based on the ExampleVehDynInteg.cpp in VTD.2.1/Develop/Communication /RDBClientSample called ExampleTruckCabin.cpp and compiled with compileTruckCabin.sh.
The corresponding VTD project is called "TruckCabinWithRDB". A sample scenario is included (TruckCabinExample.xml).

Additional notes:

- following parameters are recommended
- dynamics input in free mode
- inhibitOffroad to true (TaskControl will not try to reset the vehicle(s) to the road; the example will just move straight forward)



Configuration

A simplified vehicle dynamics model may run as plug-in in the module manager. This plug-in is provided as standard feature in VTD. For the configuration, please see the standard setup file of the moduleManager in `Data/Distros/Current/Config/ModuleManager/moduleManager.xml`

For the data exchanged between the vehicle dynamics and the taskControl, please see

How do I connect my vehicle dynamics and other components to VTD?

Data computed within the plug-in

The VIRES Vehicle Dynamics computes and transmits the following information (excerpt of the source code):

```
RDB_OBJECT_STATE_t:
    objState->base.id   = mOwnPlayerId;
    objState->base.type = RDB_OBJECT_TYPE_PLAYER_CAR;
    objState->base.geo.dimX = 4.60;
    objState->base.geo.dimY = 1.86;
    objState->base.geo.dimZ = 1.60;

    objState->base.geo.offX = 0.80;
    objState->base.geo.offY = 0.00;
    objState->base.geo.offZ = 0.30;

    objState->base.pos.x     = mInertialPos.getX();
    objState->base.pos.y     = mInertialPos.getY();
    objState->base.pos.z     = mInertialPos.getZ();
    objState->base.pos.h     = mInertialPos.getH();
    objState->base.pos.p     = mInertialPos.getP();
    objState->base.pos.r     = mInertialPos.getR();
    objState->base.pos.flags = RDB_COORD_FLAG_POINT_VALID | RDB_COORD_FLAG_ANGLES_VALID;

    objState->ext.speed.x     = mInertialSpeed.getX();
    objState->ext.speed.y     = mInertialSpeed.getY();
    objState->ext.speed.z     = mInertialSpeed.getZ();
    objState->ext.speed.h     = mInertialSpeed.getH();
    objState->ext.speed.p     = mInertialSpeed.getP();
    objState->ext.speed.r     = mInertialSpeed.getR();
    objState->ext.speed.flags = RDB_COORD_FLAG_POINT_VALID | RDB_COORD_FLAG_ANGLES_VALID;

    objState->ext.accel.x     = mInertialAccel.getX();
```

```
    objState->ext.accel.y     = mInertialAccel.getY();
    objState->ext.accel.z     = mInertialAccel.getZ();
    objState->ext.accel.flags = RDB_COORD_FLAG_POINT_VALID;

    objState->base.visMask    =  RDB_OBJECT_VIS_FLAG_TRAFFIC | RDB_OBJECT_VIS_FLAG_RECORDER;

RDB_VEHICLE_SYSTEMS_t:
    vehSys->playerId  = mOwnPlayerId;
    vehSys->lightMask = miLightMask;
    vehSys->steering  = mSteeringAngle;

RDB_ENGINE_BASE_t:
    engine->playerId  = mOwnPlayerId;
    engine->rps       = mEngineSpeed;
    engine->load      = mEngineLoad;

RDB_WHEEL_BASE_t (4x):
        wheel->playerId          = mOwnPlayerId;
        wheel->id                = i;
        wheel->radiusStatic      = mpIDynVeh->getVehicleDefinition().wheelDiameter * 0.5;
        wheel->slip              = 0.0;
        wheel->springCompression = mWheelDeflection[i];
        wheel->rotAngle          = mWheelAngle;
        wheel->steeringAngle     = mSteeringAngle;

RDB_DRIVETRAIN_BASE_t:
    drivetrain->playerId       = mOwnPlayerId;
    drivetrain->gearBoxType    = RDB_GEAR_BOX_TYPE_AUTOMATIC;

    switch ( mpIDynVeh->getVehicleDefinition().wheelDrive )
    {
        case DynVeh::IDynVeh::WHEEL_DRIVE_FRONT:
            drivetrain->driveTrainType = RDB_DRIVETRAIN_TYPE_FRONT;
            break;
        case DynVeh::IDynVeh::WHEEL_DRIVE_REAR:
            drivetrain->driveTrainType = RDB_DRIVETRAIN_TYPE_REAR;
            break;
        case DynVeh::IDynVeh::WHEEL_DRIVE_ALL:
            drivetrain->driveTrainType = RDB_DRIVETRAIN_TYPE_AWD;
            break;
    }

    if ( mGear == 0 )
        drivetrain->gear = RDB_GEAR_BOX_POS_N;
    else if ( mGear < 0 )
        drivetrain->gear = RDB_GEAR_BOX_POS_R1 + ( 1 + mGear );
    else
        drivetrain->gear = RDB_GEAR_BOX_POS_D + mGear;
```

Please let us know if you are missing any additional information.

How do I connect my vehicle dynamics (and other components) to VTD?

See next paragraph or Data I/O of VTD.

How to run a vehicle dynamics which is connected via RDB only

Note: This paragraph applies to VTD prior version 1.4 only.

For a way to connect your own vehicle dynamics, please see Data I/O of VTD.

If you want to run your own vehicle dynamics via RDB and want to avoid interference with the default dynamics, you should disable the default internal (dummy) dynamics provided with VTD. Please perform the following steps:

- In *Data/Setups/Current/Config/Simulation/simSettings.cfg* disable the dummy dynamics by deleting the corresponding line or commenting it

```
#    set TASK_LIST = "$TASK_LIST dummyDynTraf"
```

- In *Data/Setups/Current/Config/TaskControl/taskControl.xml* disable the internal interface

```
#    <Dynamics ...
       useInternalInterface="false"
      ... />
```

3rd Party Vehicle Dynamics: veDYNA

veDYNA is a vehicle dynamics package provided by TESIS Dynaware. It is not part of the basic VTD distribution and must be licensed separately.It was first linked to VTD with an internal interface and has been upgraded to an RDB connection. Both versions still exist. This manual concentrates on the latest (RDB) version but also provides some instructions for the legacy version.

You need a dongle and a valid license file to operate veDYNA.

Installation

Basic Steps

veDYNA is installed in *VTD/Runtime/AddOns/DynamicsVeDYNA* (classic version) or *VTD/Runtime/AddOns/DynamicsVeDYNA.RDB* (RDB-compliant version).

1. Unpack veDYNA to the target directory (preferably *VTD/Runtime/AddOns/DynamicsVeDYNA/* or *VTD/Runtime/AddOns/DynamicsVeDYNA.RDB*)
2. Copy the license file to this directory and link it to the name *license_dynaware.ini*. You may first have to convert it to a readable format using the command *dos2unix <licensefile>*

Access Rights

For full performance, veDYNA should run with root permissions. Also, problems with the licensing have been experienced on Ubuntu systems if veDYNA is not started with root permissions. In order to grant these permissions also for non-root users, do the following after installation of the package:

1. open a shell and go to veDYNA installation directory (see above)
2. become root user
3. execute *chown root.root StartVedyna.sh StopVedyna.sh vedyna_traffic_vtd.1.10* (or whatever version of the executable you have)
4. execute *chmod a+s StartVedyna.sh StopVedyna.sh vedyna_traffic_vtd.1.10* (or whatever version of the executable you have)
5. close the shell or exit the root mode

Project / Setup-specific settings

Vehicle Name

The RDB version of veDYNA can only run if it knows which vehicle of the scenario is to be controlled. Therefore you have to provide the name of your car explicitly within the veDYNA configuration.

- Open the file *dynaware.ini* and provide your own vehicle's name at the configuration parameter *SCP/ownVehicleName*

VTD configuration

No other vehicle dynamics must be active during operation of veDYNA. Therefore, perform the following steps

1. copy the file *moduleManager.xml* from *VTD/Data/Distros/Current/Config/ModuleManager* to either your current setup or your current project into the path *Config/ModuleManager* (a file in the current project will supersede a file in the current setup).
    1. remove the entry *DynamicsPlugin*
2. edit the file *simSettings.cfg* in *VTD/Data/Setups/Current/Config/Simulation*:
    1. remove the entry *dummyDynTraf*
    2. activate *veDYNA*
3. edit the file *taskControl.xml* in *VTD/Data/Setups/Current/Config/TaskControl*:
    1. add the attribute *useInternalInterface*="*false*" under *Dynamics*
    2. adjust the sync source, if necessary (e.g. *intern* for SiL and *extern* for human in the loop)
    3. add the attribute *cleanDriverCtrl=“true“* under *RDB*

Scenario configuration

As noted above, the own vehicle's name in the scenario MUST match the name specified in the veDYNA configuration file (see above).

1. adjust the name of your own vehicle in the scenario file

SIL-specific settings

For SiL-applications, veDYNA may be controlled by the driver in the VIRES traffic simulation (ghostdriver). Therefore, add a command to request driver information either in your scenario or in any other script you might be using and which might be sending SCP data_

1. add the command *<Player name="EGO"><Driver ctrlLatLong="ghostdriver" /></Player>* as an action in your scenario that is immediately executed after start of the simulation
2. assign a path to the own vehicle

Operation

veDYNA will only run currectly if it has received an explicit *init* command. So, each time you run with veDYNA first issue *<SimCtrl><Init mode="operation"/></SimCtrl>* and then issue *<SimCtrl><Start mode="operation"/></SimCtrl>*

Legacy Operation

If you're using the non-RDB version of veDYNA, please edit the file

Data/Setups/Current/Config/TaskControl/taskControl.xml

as follows:

```
<Dynamics .....
   legacyMode="true"
/>
```

Reason: the init sequence has changed with VTD 1.1 and the "old" vehicle dynamics needs to be initialized in a bit different fashion. This will be obsolete once the RDB connection to veDYNA has been realized.

Troubleshooting: veDYNA refuses to start

make sure that

- you have purchased a veDYNA license and you have the corresponding dongle (WiBu Key)
- the dongle is inserted in your system's USB slot
- you have installed the dongle driver, see VTD manual
- your license file *license_dynaware.ini* is a symbolic link pointing to the license file corresponding to your dongle's ID

- your license file contains the right line-end characters (in case of doubt, run *dos2unix license_dynaware.ini* in a shell)
- veDYNA can be run with root permissions (see *Access Rights* in FAQ)
- for RDB-version of veDYNA: your own player's name is listed correctly in *dynaware.ini*

Vehicle Characteristics using the RDB version of veDYNA

The configuration of veDYNA for VTD consists of two components: the parameter sets in the `work.current` directory and the actual internal characteristics of engine and gearbox. The parameter set can be modified by setting the link `data` to the corresponding sub-directory of `work.current` (e.g. `data_A7`).

The characteristics can be influenced by setting the parameter

```
[vdyctrl]
vehicle_type=3
```

in the file `dynaware.ini`. Known values and the corresponding vehicle types are:

```
NONE = 0
A4 = 1
Q4 = 2
R8 = 3
A1 = 4
A7 = 5
```

3rd Party Vehicle Dynamics: others

For other 3rd party vehicle dynamics packages (i.e. not veDYNA), the standard process of connecting a component via RDB to VTD is to be used. Parameterization of the specific vehicle dynamics package is to be performed by the respective supplier.

For more information about the data flow between a vehicle dynamics package which is connected via RDB to VTD and the VTD TaskControl, see

How do I connect my vehicle dynamics and other components to VTD?

How can I get contact point information for my vehicle dynamics?

Contact points via TaskControl (low frequency)

The taskControl can provide contact point information in sync with its own frequency. Typically, this will be no more than 60Hz for typical applications with image output.

Two steps are necessary to receive the contact point data:

- request contact points via SCP (once at beginning of simulation)
- read back the data stream via RDB (continuously)

The SCP command for the request of a contact point is formatted as follows:

```
<ContactPoint id="_unique numeric id by user_">
    <PosPlayer player="name of player to which cp is to be attached"
               dx="x-offset from player origin to contact point, player co-ordinates"
               dy="y-offset from player origin to contact point, player co-ordinates"
               type="standard"/>
</ContactPoint>
```

A separate command shall be issued for each contact point.

Example: vehicle "Own" with wheel base of 2.5m, track width 1.6m, four wheels (front left, front right, rear left, rear right)

```
<ContactPoint id="1">
    <PosPlayer player="Own" dx="2.0" dy="-0.8" type="standard"/>
    <PosPlayer player="Own" dx="2.0" dy="0.8"  type="standard"/>
    <PosPlayer player="Own" dx="0.0" dy="-0.8" type="standard"/>
    <PosPlayer player="Own" dx="0.0" dy="0.8"  type="standard"/>
</ContactPoint>
```

The data stream from TC to the vehicle dynamics will now contain four packages of type

`RDB_PKG_ID_CONTACT_POINT`.

In each of these packages, the following data will be contained:

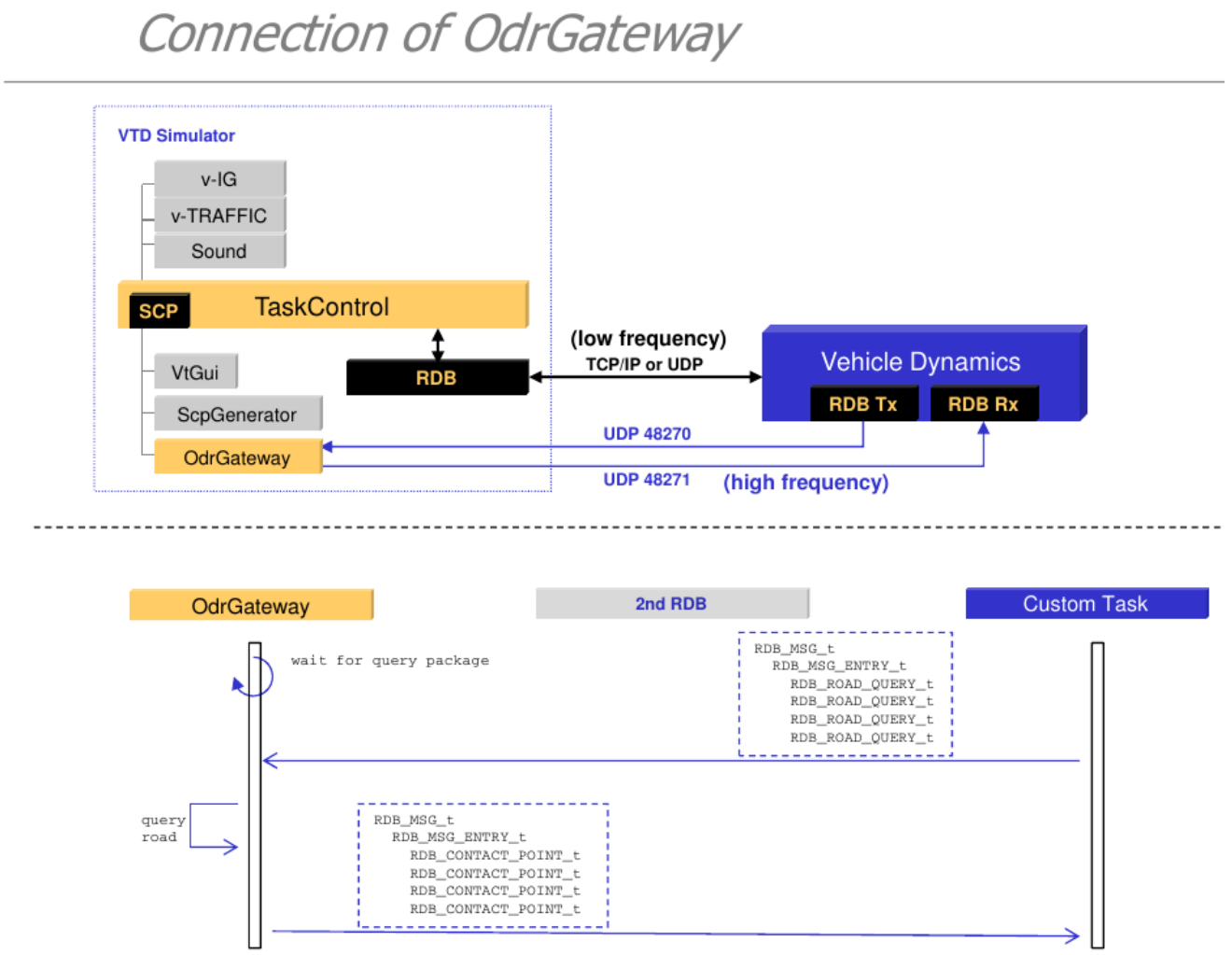
```
    uint16_t     id;            /**< unique ID of the contact point
    uint16_t     flags;         /**< various flags with contact point options
    RDB_COORD_t  roadDataIn;    /**< inertial position of contact point; heading=0; pitch and roll relative to vehicle axis
    float        friction;      /**< road friction at contact point
    int32_t      playerId;      /**< ID of the player to which CP belongs; only valid if corresponding flag is set
```

The absolute elevation of the road at the contact point will be given in `roadDataIn.z`, the friction will be given in `friction`. For more details, see the RDB documentation at `Doc/RDB_HTML/index.html` in your VTD distribution.

Contact points via OdrGateway (high frequency)

General

Many vehicle dynamics simulation packages will need contact point information at higher frequencies than the basic frequency of the taskControl. For these purposes, a separate tool, the so-called *OdrGateway* is available. It is to be connected to the vehicle dynamics via a dedicated RDB connection (TCP or UDP) - in order to not disturb traffic on other connection channels. The protocol for all communication is RDB.



Installation

The *OdrGateway* will be installed in the *AddOns* section of VTD:

```
VTD
├── Runtime
│   ├── AddOns
│   │   ├── OdrGateway
│   │   │   ├── odrGateway
│   │   │   ├── odrGateway.TCP.xml
│   │   │   ├── odrGateway.UDP.xml
│   │   │   ├── odrGateway.xml -> odrGateway.TCP.xml
```

It comes with the configuration files for TCP/IP and UDP connection. Please set the link *odrGateway.xml* to your desired configuration and adjust its contents accordingly (see below).

Process Configuration

VTD 2.x

In order to have the OdrGateway start automatically upon loading the components, edit the file *VTD/Data/Setups/Current/Config/SimServer/simServer.xml* and add the following paragraph:

```
      <Process
        name="OdrGateway"
        auto="false"
        explicitLoad="false"
        path="$VI_ADDON_DIR/OdrGateway"
```

```
        executable="odrGateway"
        cmdline="-f $VI_ADDON_DIR/OdrGateway/odrGateway.xml"
        affinitymask="0xFF"
        schedPolicy="SCHED_RR"
        schedPriority="20"
        useXterm="true"
        xtermOptions="-fg Green -bg Black -geometry 80x10+0+326"
        workDir="$VI_CURRENT_SETUP/Bin">
        <EnvVar name="LD_LIBRARY_PATH"      val="$VI_CORE_DIR/Lib:$LD_LIBRARY_PATH"                />
      </Process>
```

VTD 1.4.x

In order to have the OdrGateway start automatically upon loading the components, edit the file *VTD/Data/Setups/Current/Config/Simulation/taskSettings.cfg* and add the following paragraph:

```
# ......... OpenDRIVE Gateway
#
if { test $TASK_TYPE = "odrGateway" } then
  set BIN_NAME      = odrGateway
  set OBJ_DIR       = ( $VI_ADDONS_DIR/OdrGateway )
  set WORK_DIR      = ( . )
  set CMD_ARGS      = ( -c $OBJ_DIR/odrGateway.xml )
  set START_CMD     = ( source $ENV_SETTINGS ; cd $WORK_DIR ; $OBJ_DIR/$BIN_NAME $CMD_ARGS $RECORD_CMD ; sleep $XTERM_LIFETIME
  set USE_XTERM     = true
  set XTERM_OPTIONS = ( $XTERM_OPTIONS_COMMON -bg LightGoldenRod -geometry 80x10+508+163 -title $TASK_TYPE )
endif
```

Now, edit the file *VTD/Data/Setups/Current/Config/Simulation/envSettings.cfg* and add the line

```
set TASK_LIST = "$TASK_LIST odrGateway"
```

to the definition of tasks that are to be started.

This should be it.

Command Line Options (for all versions):

You may use the following command line options for the configuration of the process in the above entries:

```
odrGateway [-i interface] [-f filename] [-t frametime] [-h] [-x filename] [-m] [-v]
   -h :          show this help information
   -i interface: use the indicated interface for communication (e.g. \"eth1\")
   -f filename:  configuration file
   -t frametime: run with the given frametime instead of the std. 0.001s (1000Hz)
   -x filename:  load the given .xodr course file
   -m:           run in test mode (i.e. act as dummy sender)
   -v:           run in verbose mode
```

RDB Configuration

The RDB configuration of the OdrGateway is given in the corresponding config file.

Example:

```
<RDB>
    <Port name="RDBin"  number="48271" type="UDP" />
    <Port name="RDBout" number="48270" type="UDP" />
</RDB>

<Config verbose="false" forceRelativeMode="false"/>
```

In this example, a connection via UDP is defined with

- port 48271 used as port from vehicle dynamics to odrGateway
- port 48270 used as port from odrGateway to vehicle dynamics

In order to use UDP unicast instead of broadcast (version 2.1.1+), change the second configuration line to

```
<RDB>
    <Port name="RDBin"  number="48271" type="UDP" />
    <Port name="RDBout" number="48270" type="UDP" targetAddress="111.222.111.222" />
</RDB>
```

In the `<Config>` section, the verbose mode is set to `false` (you may enable it by setting it to `true`) and relative position mode is not forced (see below).

Communication

The OdrGateway will be waiting for incoming RDB messages (i.e. requests for contact point information) and will send back the corresponding contact point data. By this, latency is minimized and the frequency of the information transfer can be controlled via the requesting component. Frequencies up to 1kHz for 20+ contact points can easily be handled (usually, you should be fine with 4 points).

The component requesting the contact point information shall send one RDB package of type `RDB_PKG_ID_ROAD_QUERY` for each contact point. For a typical

case of four contact points, the complete RDB message will look as follows:

```
RDB_MSG_HDR_t
  RDB_MSG_ENTRY_HDR_t  (pkgId = RDB_PKG_ID_START_OF_FRAME)
  RDB_MSG_ENTRY_HDR_t  (pkgId = RDB_PKG_ID_ROAD_QUERY)
     RDB_ROAD_QUERY_t
     RDB_ROAD_QUERY_t
     RDB_ROAD_QUERY_t
     RDB_ROAD_QUERY_t
  RDB_MSG_ENTRY_HDR_t  (pkgId = RDB_PKG_ID_END_OF_FRAME)
```

In the structure `ROAD_QUERY_t` the following fields have to be filled correctly:

```
    uint16_t id;           /**< unique ID of the road query point (reflected in answer)
    uint16_t flags;        /**< additional query flags
    double   x;            /**< x co-ordinate (inertial) of query location
    double   y;            /**< y co-ordinate (inertial) of query location
```

The response will be an RDB message containing the corresponding contact points with the details as described above. In our example with four queries, the result will look as follows:

```
RDB_MSG_HDR_t
  RDB_MSG_ENTRY_HDR_t  (pkgId = RDB_PKG_ID_CONTACT_POINT)
     RDB_CONTACT_POINT_t
     RDB_CONTACT_POINT_t
     RDB_CONTACT_POINT_t
     RDB_CONTACT_POINT_t
  RDB_MSG_ENTRY_HDR_t  (pkgId = RDB_PKG_ID_END_OF_FRAME)
```

Operation in Relative Position Mode

availability: VTD 2.0.3+

Some vehicle dynamics packages prefer initialization at their internal origin (x/y/z/h = 0/0/0/0) and send their position back to the taskControl relative to this origin. In the scenario, however, the vehicle may - most probably - be located at an initial position different from the database origin. The taskControl will automatically take care of this fact if the vehicle dynamics sends `RDB_OBJECT_STATE_t` packages whose `pos` member's coordinate system is explicitly set to `RDB_COORD_TYPE_RELATIVE_START`. The received co-ordinates will be offset by the initial co-ordinates (and orientation) given in the scenario.

The OdrGateway also needs to be informed whether the co-ordinates transmitted in the `RDB_ROAD_QUERY_t`-packages are to be interpreted as inertial (absolute) or relative co-ordinates. For the latter, the user has to set the `flags` member in each `RDB_ROAD_QUERY_t`-package to `RDB_ROAD_QUERY_FLAG_RELATIVE_POS`. The initial position of the external vehicle (i.e. the offset which has to be applied for querying the road network) will be transmitted by the taskControl to the OdrGateway automatically upon receipt of the first relative position. So, per default, no further user action is required. However, if you wish to overwrite this initial position, then you may also issue the command

```
<Set entity="odrGateway" id="id_of_player"><PosInertial x="x_reset_pos" y="y_reset_pos" z="<_reset_pos" h="h_reset_pos"/></Set
```

legacy support: if your vehicle dynamics does not yet support the `flags` field in `RDB_ROAD_QUERY_t`, then you may also force the OdrManager to work in relative position mode by adding the following entry to the configuration file:

```
<Config forceRelativeMode="false"/>
```

Offroad Handling

Per default, VTD will have a watchdog running which becomes active once you leave the logical road area (i.e. OpenDRIVE lanes). This watchdog will create an *offroad* event and will re-position you automatically to the last known valid lane position. A corresponding *<Set>* command will be issued, so that also an external vehicle dynamics application may react to this signal.

In some cases, it might be more convenient to allow driving "everywhere", no matter whether there is an underlying road description. For these cases, please disable offroad handling by performing the following setting in the taskControl's configuration file (typically: *Data/Setups/Current/Config/TaskControl /taskControl.xml*):

```
<TaskControl>
  :
    <Dynamics  inhibitOffroad="true" />
  :
</TaskControl>
```

An alternative to disabling the offroad handling completely may be giving your own vehicle a bit more room for maneuver beside the road. In VTD, you may specify a virtual border that is added to the outer lane on each side of the road with the following command (here: 20m are added on each side):

```
<TaskControl>
  :
    <Dynamics  inhibitOffroad="false" offroadBorder="20" />
  :
</TaskControl>
```

Manual Control of the Vehicle

Since VTD 2.0.1, it is possible to provide the inputs to an externally computed vehicle also by keyboard. For this purpose, a separate tool "Vehicle Controller"

has been created. It uses SCP commands for performing this control. You may invoke the tool as follows:

```
usage: vehicleController [-p:x] [-s:IP] [-n:name]

    -p:x      Remote port to send to
    -s:IP     Server's IP address or hostname
    -n:name   name of controlled vehicle (default: Ego)

    Example: ./vehicleController -p:48179 -n:Ego
```

The shell where you started the tool will go blank; while keyboard focus is within the shell, you are able to use the four cursor keys to control the vehicle.

Pedestrians

DI-Guy Pedestrians

DI-Guy is a 3rd party library which is not contained in the standard delivery package of VTD. It requires a separate license.

If you don't have your DI-Guy license keys yet or wish to extend your license with DI-Guy functionality, please *contact us*. Please check your VTD license certificate if you are not sure what sort of license you have and whether your machine is licensed to run DI-Guy.

Installation and Configuration

All instructions for installation and basic configuration have been moved to the configuration pages of this wiki:

VTD_Configuration_Issues

Additional remarks for VTD installations prior to VTD 1.4.x

Configuration of the TaskControl

1. In *Data/Setups/* ***Current*** */Config/TaskControl/taskControl.xml* set the attribute *dynPlayerConfig* within the section `<ImageGenerator/>` to `true`

Plug-in of the Image Generator

DI-Guy is handled within a plug-in of the image generator. Usually, you don't have to worry about this configuration except for referring to the correct version of DI-Guy (see above). Even if you don't have a license, you may still leave the plug-in activated. If you wish to explicitly de-activate the DI-Guy plug-in of the IG, perform the following steps:

1. Open the file *Data/Setups/Current/Config/ImageGenerator/IGbase.xml*
2. In the Section *<Plugins>*, remove the entry *<Plugin file="PedestriansBDI_v10"/>* (delete the line or use XML comments to de-activate it)
3. In the Section *<Components>*, remove the tag *<PedestriansBDI_v10> ... </PedestriansBDI_v10>* (delete the lines or use XML comments to de-activate them)
4. That's it.

For re-activating the DI-Guy plug-in, just activate above entries again.

Operation

Character Viewer of Scenario Editor

From within the Scenario Editor, you may start the DI-Guy character viewer. If the installation is correct and you are running one of the recommended operating systems, the viewer should start without problems. However, if your installation deviates from the standard setup then you will have to adjust the script

startCharacterViewer

in the directory

Runtime/Core/ScenarioEditor

accordingly.

Vires Pedestrians

VTD supports databases with stationary pedestrians that are animated without the DI-Guy library and thus require no DI-Guy license.

Database creation

To insert an animated character inside a database, a static character in the form of an .flt object has to be placed in the database with RoadDesigner ROD. Characters suitable for animation can be found in category "character". Each of these static characters has a corresponding animated version, stored in a .osgb file.

ImageGenerator configuration

The animation of the pedestrians at runtime is handled by an optional ImageGenerator component called "AnimatedPedestrians". The configuration of this component is done in an IG configuration file, i.e. VTD/Data/Projects/UnitTest/Config/ImageGenerator/ProjectIGConfig.xml.
First, the corresponding plugin "AnimatedPedestrians" has to be inserted under the <Plugins> element. Second, the component "AnimatedPedestrians" must be registered with a name in the <Components> section.

The component replaces the static pedestrians by their animated .osgb counterparts at runtime and thus needs to find the .osgb files. The path to the directory containing the .osgb files is declared per <FilePath> element underneath the <SystemConfig> element. A sample ProjectIGConfig.xml may look like this:

```
<TaskControl>
```

```
<IGconfig>
  <SystemConfig>
    <FilePath path="IGLinks/Vig/data/Osgb" />
    ...
  </SystemConfig>

  <Plugins>
    <Plugin file="AnimatedPedestrians" />
    ...
  </Plugins>

  <Components>
    <AnimatedPedestrians name="MyAnimatedPedestrians" />
    ...
  </Components>
</IGconfig>
```

No further steps are required to animate the pedestrians in the database.

For performance reasons the AnimatedPedestrians component switches between the static and animated pedestrian depending on the distance between the observer and the pedestrian position. If the distance to the pedestrian is less than a certain minimum distance, the animated pedestrian is used. For distances greater than the minimum distance but below a maximum distance, the static version of the pedestrian is used. If the distance is even greater than the maximum distance, the pedestrian is not drawn at all.
These two distances can be specified with optional attributes "staticPedestrianMinDistance" and "staticPedestrianMaxDistance". Example:

```
<AnimatedPedestrians name="MyAnimatedPedestrians" staticPedestrianMinDistance="100" staticPedestrianMaxDistance="300" />
```

The value for attribute staticPedestrianMaxDistance must be greater than the value of staticPedestrianMinDistance. The default values for the attributes are "70" and "250" respectively.

Common Questions

Which configuration data is used by VTD?

VTD provides three levels of configuration:

- the current project
- the current setup
- the current distribution

In each of these levels configuration data may be provided in the directory `Config` and the corresponding sub-directories. Depending on the individual components, the configuration hierarchy will be as follows:

Config/Simulation

- `simSettings.cfg`
- `envSettings.cfg`
- `taskSettings.cfg`

For all of these files, the following rule applies:
1. The file in `Data/Setups/Current/Config/Simulation` will be read
2. It will trigger reading of the corresponding file in `Data/Setups/Common/Config/Simulation`
3. The settings may be overwritten or extended by the corresponding file in `Data/Projects/Current/Config/Simulation`

VTD components (except for ImageGenerator)

These components all use the FileFinder for locating their configuration files. The FileFinder works as follows:
1. The configuration file (e.g. `taskControl.xml`) will be searched in the current project (e.g. `Data/Projects/Current/Config/TaskControl/taskControl.xml`)
2. If not found, the configuration file will be searched in the current setup (e.g. `Data/Setups/Current/Config/TaskControl/taskControl.xml`)
3. If still not found, the configuration file will be searched in the current distro (e.g. `Data/Distros/Current/Config/TaskControl/taskControl.xml`)
4. If finally not found, the component will either start with default parameters or you will be doomed ;-)

ImageGenerator

The location of the ImageGenerator's configuration files is given in the its master configuration file. This file is typically `Data/Setups/Current/Config/ImageGenerator/AutoCfg.xml`. If you want to use a different master file, you have to adjust the IG's command line in your simulation configuration file `taskSettings.cfg` (in any of the above mentioned directories).

Conclusion:

You may actually configure the whole simulation (i.e. all components) within your project if you like. Usually, the main configuration is performed via the current setup and only some parts are re-configured in the project. However, the way you arrange your configuration is completely up to you.

What data do I have to backup?

Making backups of data is quite a crucial issue and should not be neglected. The directory structure of VTD allows you - to a considerable extent - to distinguish between system data (i.e. delivered with VTD distributions) and user data. Some things in this respect may not yet provide an ideal solution but we are working on it.

Setups

A setup contains the information about the actual configuration of VTD (distribution of IG channels etc.). Please include in your backup all sub-directories of `Data/Setups` that you have created or modified by yourself. In order to be sure you have not accidentally altered one of the common files, include

`Data/Setups/Common` in your backup.

Projects

A project contains all user data which is relevant for a given test or test series. Please include in your backup all sub-directories of `Data/Projects` (i.e. projects) that you have created or modified by yourself. Please check carefully that your project directory really contains the data that you want to backup. Databases delivered with VTD are usually placed in `Data/Distros/Current` and they are only symbolically linked to the respective projects. For your own databases, you should avoid doing this and place the databases (and other data) directly in your project's sub-directory instead. If any of your data which you have referring to from within your project is located outside of the project then you have to backup also this data separately.

ROD databases

Usually ROD is installed as a more or less separate tool in Runtime/AddOns. Any sub-directory of ROD to which you have added your own data should be included in your backup. In the standard installation this refers to your "Work" directory or your "Project" directories.

Some installations have the user data of ROD installed at `Data/ROD`. In this case, you should backup this directory. If you have added any data to other ROD directories, please don't forget to also those data.

In general, it should be sufficient to backup the setup files (i.e. all files `TT_xxxxx.DAT`) and the `.tdo` files since you can generate the OpenDRIVE and graphics files based on these data again. However, some algorithms may change over time and if you want to make sure that you have a backup of exactly the same OpenDRIVE and graphics files that you used in a certain test, then you should also backup those file.

General Note

If you make a backup, make also sure you can restore the data. In an ideal case, you will use an out-of-the-box installation of VTD and restore your data onto it. If everything performs well you will be able to continue working as usual. In a realistic case, you will not spend your time on doing a new installation but instead you should restore your data in a separate (temporary) directory and check that everything you consider your own data is contained.

Which SW-packages do I need for video generation?

If you want to perform off-line video generation, please check that you have installed all packages listed in the standard configuration procedure and follow also the instructions given in VTD_Configuration_Issues.

In SuSE 11.x, you may also have to symbolically link the installed *libpng...* to *libpng.so.3*, so that the *mencoder* delivered in the video tools of VTD will work.

Which data can I expect from VTD?

See Simulation_Data.

How can I locate a project outside the VTD tree?

In order to locate a project outside the VTD data tree and still be able to use them with VTD, you have to perform the following steps:

- In Data/Current/Config/Simulation/envSettings.cfg, set or adjust the environment variable *VI_PROJECT_DIR*
- Within the project, correct the following (relative) symbolic link
  - <Project>/Sounds

After a restart of VTD (incl. the GUI) only the projects within *VI_PROJECT_DIR* will be available, see also http://viresftp.dyndns.org/issues/83

How can I document my own source code with Doxygen if I'm including VTD header files (viRDBIcd.h or scpIcd.h)?

1. You need Doxygen >= V 1.6.3
2. Add the following lines in the Doxyfile

```
ALIASES = "unit=\par SI Unit: \n" \
          "msgid=\par Message ID: \n"
```

How do I format a command for immediate mode of the SCP Generator?

Problems may occur if a command containing the "-character is given as command line argument to the SCP Generator for immediate execution. In order to avoid these problems, enclose your command with single quotes (').

Example:

```
scpGenerator -i '<Symbol name="testText"> <Text data="Text"/> <PosScreen x="0.4" y="0.4"/> </Symbol>'
```

How do I define *Umlaute* in SCP commands?

If you're using *Umlaute* in SCP scripts (e.g. for text symbols), the scripts should be encoded '''ISO 8859'''. Otherwise interesting but quite strange symbols may appear on your screen.
Special characters of SCP messages inside scenarios can be masked according to the following table:

Character	Encoding
Ä	&#196;
Ö	&#214;
Ü	&#220;
ä	&#228;
ö	&#246;

ü	&#252;
ß	&#223;
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;
€	&#8364;

Communication

How to configure the communication so that only loopback device is used?

Background: the concurrent operation of multiple instances of VTD within a single network may cause interferences. Therefore, some sort of shielding needs to be done.

The following things should be adapted:

taskControl.xml

```
<TaskControl>
  <Interface name="lo" />
  <GSI portType="loopback" .../>
  <RDB portType="loopback" .../>
  <ScVis portType="loopback" .../>
</TaskControl>
```

envSettings.cfg

```
setenv TRAFFIC_VIS_ETHERNET lo
setenv SCVIS_ETHERNET lo
```

GSI / RDB will send UDP packages via the loopback device, so please configure your own components accordingly.

How to use a specific Ethernet device for communication?

Per default, VTD uses *eth0* for the communication. If you want to use a different interface (in the examples below: *ethX*), please adapt the following configuration entries:

taskControl.xml

```
<TaskControl>
  <Interface name="ethX" />
  <GSI interface="ethX" .../>
  <RDB interface="ethX" .../>
  <Sound interface="ethX" .../>
</TaskControl>
```

If the attribute "*interface*" is not defined for GSI and RDB, then the interface defined with the *<Interface>* tag will be used automatically.

envSettings.cfg

```
setenv TRAFFIC_VIS_ETHERNET ethX
setenv SCVIS_ETHERNET ethX
```

My UDP communication does not work

VTD may use UDP ports. This may interfere with your firewall. Either change your configuration that only TCP or loopback ports are used or open your firewall for UDP communication. On dedicated systems which are not connected to a public network, disabling the firewall completely is also an option (actually, it's the recommended one).

How can I influence the information about traffic signs?

Traffic signs are retrieved from the OpenDRIVE database. The taskControl performs a preliminary detection for the master vehicle (own vehicle) during the evaluation of the road information. In order to reduce the amount of data sent via network, a pre-filtering is done, so that only traffic signs of the following types will be detected:

```
( sig->mType > 100000 ) ||
( sig->mType == 205 )   ||
( sig->mType == 206 )   ||
( sig->mType == 274 )   ||
( sig->mType == 276 )   ||
( sig->mType == 277 )   ||
( sig->mType == 278 )   ||
( sig->mType == 280 )   ||
( sig->mType == 281 )   ||
```

```
( sig->mType == 282 )   ||
( sig->mType == 293 )   ||
( sig->mType == 294 )   ||
( sig->mType == 301 )   ||
( sig->mType == 306 )   ||
( sig->mType == 307 )   ||
( sig->mType == 310 )   ||
( sig->mType == 311 )   ||
( sig->mType == 330 )   ||
( sig->mType == 334 );
```

This filtering may be inappropriate for certain cases. In order to turn it off, change the configuration in `Data/Setups/Current/Config/TaskControl/taskControl.xml`:

```
<RoadInfo .... filterSignals="false" ../>
```

If you are using a sensor plug-in in the moduleManager in order to detect signals, then make sure that this plug-in is also configured correctly. In your moduleManager configuration file (typically `Data/Projects/Current/Config/ModuleManager/moduleManager.xml`) make sure that your sensor has the following entry:

```
<Filter type="trafficSign"/>
```

If you provide the filter entry standalone, all signs coming from the taskControl will be detected. If you want to restrict this to a certain subset, then provide the type number for each signal type that is to be detected, e.g.

```
<Filter type="trafficSign" signType="274"/>
<Filter type="trafficSign" signType="212"/>
<Filter type="trafficSign" signType="305"/>
```

Note: once you start using explicit filtering of sign types, you must provide type identifiers of all types that you want to detect!

Task and Environment Configuration

How can I customize environment variables used in VTD?

Environment variables can be set in the following three places:

1. In `Data/Setups/Common/Config/Simulation/envSettings.cfg`
2. In `~/.config/VIRES/VTD/envSettings.cfg` (with VTD 1.2.1 and later)
3. In `Data/Setups/Current/Config/Simulation/envSettings.cfg`

The files will be parsed in exactly this sequence upon start of a component. Therefore, you may override previous settings in the succeeding files. In order to stay consistent with the standard distribution, you should **NOT** alter the settings in the `Common` directory.

How can I initialize VTD for each simulation?

For this purpose a script `initProjects.sh` is located in `Data/Setups/Common/Scripts`. There you can add custom commands like calling SCP messages:

```
#!/bin/sh
#
# script called upon INIT of a simulation
#
# (c) 2015 VIRES GmbH, 02.01.2015 by M. Dupuis
#
# this script will be called with two arguments
# $1: project name
# $2: project path

# just a brief test

if [ $1 = unknown ]; then

echo  -e "initProject.sh: initializing project."

else

echo -e "initProject.sh: initializing project \"$1\" at \"$2\""

fi

# set the cameras for two channels to two different players

$VI_TOOLS_DIR/ScpGenerator/scpGenerator -i '<Symbol name="myText"><Text data="This is the screen" colorRGB="0xff0000" size="10
```

If you want to call the script only for a specific setup, copy the file (and maybe create the folder Scripts) to `Data/Setups/SetupName/Scripts`. It will be called automatically.

How do I integrate a 3rd party component (here: ADTF) into the VTD mechanisms (load components, start etc.)?

1. If required, create a dedicated setup (in *Data/Setups*) (see also VTD-User-Manual).

1. Copy an existing setup (e.g. *Standard.RDB*) with *cp*

```
cp -R Standard.RDB ADTFAutoStartSetup
```

2. Adjust the file *Config/Simulation/taskSettings.cfg* within the area "User-Defined tasks", so that your own task is registered as component of VTD.

```
# ----------------- ADTF -------------------------------------------------
#
if { test $TASK_TYPE = "ADTF" } then

  set BIN_NAME      = adtf_devenv
  set OBJ_DIR       = ( $ADTF_DIR/bin/debug )
  set WORK_DIR      = ( $VI_PROJECT_DIR )
#  set CMD_ARGS      = ( -p myproject.prj $EXTRA_ARGS )
  set CMD_ARGS      = ( $EXTRA_ARGS )
  set START_CMD     = ( source $ENV_SETTINGS ; cd $WORK_DIR ; $OBJ_DIR/$BIN_NAME $CMD_ARGS $RECORD_CMD ; sleep $XTERM_LIF
  set USE_XTERM     = true
  set XTERM_OPTIONS = ( $XTERM_OPTIONS_COMMON -bg DarkSeaGreen2 -geometry 80x10 -title ScenarioEditor )
endif
```

3. Add your own component to the file *Config/Simulation/simSettings.cfg* in the section "own tasks"

```
    set TASK_LIST = "$TASK_LIST ADTF"
```

Now, the new component (here: ADTF) will be loaded automatically if *Load Components* is executed in the GUi.

How can I assign a task to a dedicated CPU?

The CPU for each task (except for the ImageGenerator, see below) may be defined within the setup-specific file

`Data/Setups/Current/Config/Simulation/taskSettings.cfg`

In order to define a CPU, set the variable `VI_USE_CPU` to the corresponding value.

Example: assign the taskControl to CPU 4

Insert the following lines into your setup's `taskSettings.cfg`:

```
if { test $TASK_TYPE = "taskControl" } then
  set VI_USE_CPU = 4
endif
```

NOTE: all other start parameters of the taskControl are defined in `Data/Setups/Common/Config/Simulation/taskSettings.cfg` and don't have to be repeated here (unless they shall be changed).

For the IG, thread control is a bit more granular. Check VTD_FAQ_ImageGenerator for the detailed description.

How can I run VTD on a remote PC?

Running VTD via remote login

If you want to run VTD on a remote PC and just have the GUI on your computer for control purposes, all you have to to is to re-direct the GUI output to your own computer. Per default, all output windows will show up on the computer where VTD is started, therefore you'll have to perform a little tweak in the configuration. The safest way to do so is the following:

1. Log into the target computer with X re-direction: `ssh -X <username>@<remote-host>`
2. Check for (and remember) the correct value of the `DISPLAY` variable: `env | grep DISPLAY`
3. Open the file `Data/Setups/Current/Config/Simulation/taskSettings.cfg`
4. Check whether a configuration line for the task type `VtGui` is available
5. If not, copy the corresponding lines from `Data/Setups/Common/Config/Simulation/taskSettings.cfg`
6. Change the line defining the `START_CMD`, so that it overrides the setting of the `DISPLAY` variable with the correct value (as retrieved above).

Example:

```
# ......... GUI .........
#
if { test $TASK_TYPE = "VtGui" } then

  set BIN_NAME      = Vtgui
  set OBJ_DIR       = ( $VI_CORE_DIR/VtGui )
  set WORK_DIR      = ( . )
  set CMD_ARGS      = ( -p $PORT_VTGUI -c ../Config/VtGui/database.xml )
  set START_CMD     = ( source $ENV_SETTINGS ; setenv DISPLAY localhost:10.0 ; cd $WORK_DIR ; $OBJ_DIR/$BIN_NAME $CMD_ARGS $RE
  set USE_XTERM     = false
  set XTERM_OPTIONS = ( $XTERM_OPTIONS_COMMON -bg LightGoldenRod -geometry 80x10+508+163 -title VTGUI )
endif
```

1. Now go to `Data/Setups/Current/Bin`
2. Execute `./startGui.sh`
3. The GUI should show up on your local computer and if you execute `Load Components`, all components will run and occur on the remote computer.

Running individual tasks on a remote computer

This configuration is usually required for distributed image generators etc. but may of course be applied to any of the VTD components. For the tasks that you want to run on a remote computer, perform the following steps:

1. open the file `Data/Setups/Current/Config/Simulation/taskSettings.cfg`
2. check whether a configuration line for the task that you want to run remotely is available
3. if not, copy the corresponding lines from `Data/Setups/Common/Config/Simulation/taskSettings.cfg` or create a new entry
4. add the lines defining the variables `REMOTE_HOST`, `REMOTE_ARGS` and `REMOTE_USER`

Example:

```
# ......... example: IG CENTER CHANNEL on remote PC
#
if { test $TASK_TYPE = "igCtr" } then

  set BIN_NAME      = vigcar
  set OBJ_DIR       = ( $VI_CORE_DIR/ImageGenerator/bin )
  set WORK_DIR      = ( . )
  set CMD_ARGS      = ( $VI_SETUP_DIR/Config/ImageGenerator/AutoCfg.xml )
  set START_CMD     = ( source $ENV_SETTINGS ; cd $WORK_DIR ; $OBJ_DIR/$BIN_NAME $CMD_ARGS $RECORD_CMD ; sleep $XTERM_LIFETIME
  set USE_XTERM     = true
  set XTERM_OPTIONS = ( $XTERM_OPTIONS_COMMON -bg LightGoldenRod -geometry 80x10+508+163 -title IMAGEGENERATOR )
  set REMOTE_HOST   = "vtd-visual01"
  set REMOTE_ARGS   = "setenv DISPLAY :0.0 ; cd VTD/Data/Setups/Current/Bin ; "
  set REMOTE_USER   = "vtd"
  set VI_USE_CPU    = 1
endif
```

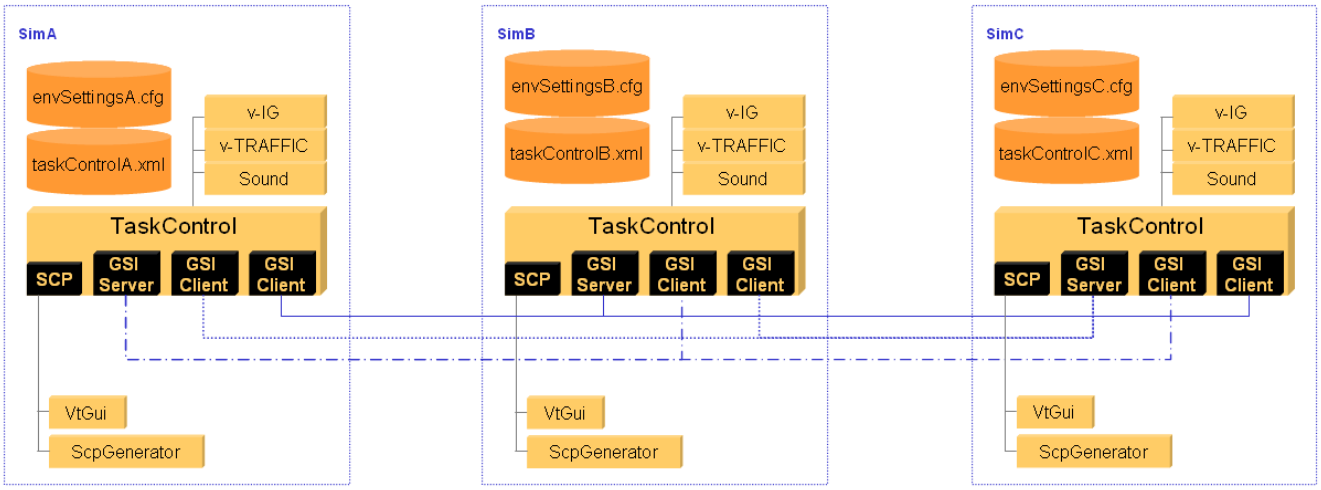
In order to being able to run the remote task, make sure that

1. the path to the remote directory is correct (see `REMOTE_ARGS`)
2. the remote host is known to your local host under the given name; see `/etc/hosts` or provide the IPv4 address
3. the remote user may login via ssh without password

Now you can start the GUI locally and upon executing `Load Components` the corresponding tasks will automatically be started on the remote computers.

How can I configure multiple simulators for co-operation?

Linking multiple simulators is more or less a configuration issue which lets various instances of the `taskControl` exchange information with each other. Let's assume, we want to link three simulators, so that they can all run concurrently in the same scenario. The following picture shows the basic setup:



Note: linking of the simulators is still performed via the *old* GSI interface. In some upcoming version of VTD, this may be migrated to RDB.

What steps are to be performed for the multi-sim environment?

Step 1: Preparing the installation(s)

A) Running on three different installations (i.e. on three simulators not sharing the same VTD installation):
1. On each installation (simulator) create a setup which you intend to use for the co-operation task and make this setup the `Current` setup

B) Running in one installation
1. create three different setups (e.g. `SetupA`, `SetupB` and `SetupC`)
2. make `SetupA` the `Current` setup in your original installation
3. create two additional installation directories (e.g. `VTD.2ndUser` and `VTD.3rdUser`)
4. in each of the additional installation directories:
    1. create a symbolic link to the original `Runtime` directory
    2. create a `Data` sub-directory
    3. in the `Data` sub-directory:
        1. create a symbolic link to the original `Projects` directory

2. create a symbolic link to the original `Distros` directory
3. create a sub-directory `Setups`
4. in the `Setups` sub-directory:
    1. create a symbolic link to the original `Common` directory
    2. create a link `Current` to the specific original setup directory (i.e. `SetupB` and `SetupC`)

Your installation will look similar to this:

Original installation

```
VTD/
  - Data
    - Distros
      - Current (symbolic link to local distro)
    - Projects
      - Current (symbolic link to local project)
    - Setups
      - Common
      - Current (symbolic link to SetupA)
      - SetupA
      - SetupB
      - SetupC
  - Runtime
```

2nd simulation:

```
VTD.2ndUser/
  - Data
    - Distros (symbolic link to VTD/Data/Distros)
    - Projects (symbolic link to VTD/Data/Projects)
    - Setups
      - Common (symbolic link to VTD/Data/Setups/Common)
      - Current (symbolic link to VTD/Data/Setups/SetupB)
  - Runtime
```

3rd simulation:

```
VTD.3rdUser/
  - Data
    - Distros (symbolic link to VTD/Data/Distros)
    - Projects (symbolic link to VTD/Data/Projects)
    - Setups
      - Common (symbolic link to VTD/Data/Setups/Common)
      - Current (symbolic link to VTD/Data/Setups/SetupC)
  - Runtime
```

Step 2: Configuring the TaskControl

Now that the installation directories are available, configure each taskControl so that co-operation (i.e. run-time data exchange) is enabled. In order to link three simulators, each taskControl must open one additional GSI server port and two GSI client ports (**note** these have to be named **receivePort** instead of **port**), so that each simulator is linked to each other. This must be done in each taskControl's config file (`Data/Setups/Current/Config/TaskControl/taskControl.xml`). For the following examples, let's assume, the simulators are running on computers with addresses 10.34.20.1, 10.34.20.2 and 10.34.20.3, respectively:

Original Simulation:

```
    <GSI enable="true" portType="TCP" imageTransfer="false"/>

    <GSI name="SimA" enable="true" portType="TCP"
         client="false" port="48121" receive="true"/>

    <GSI name="SimB" enable="true" portType="TCP"
         client="true" receivePort="48122" server="10.34.20.2" receive="true"/>

    <GSI name="SimC" enable="true" portType="TCP"
         client="true" receivePort="48123" server="10.34.20.3" receive="true"/>
```

2nd simulation:

```
    <GSI enable="true" portType="TCP" imageTransfer="false"/>

    <GSI name="SimA" enable="true" portType="TCP"
         client="true" receivePort="48121" server="10.34.20.1" receive="true"/>

    <GSI name="SimB" enable="true" portType="TCP"
         client="false" port="48122" receive="true"/>

    <GSI name="SimC" enable="true" portType="TCP"
         client="true" receivePort="48123" server="10.34.20.3" receive="true"/>
```

3rd simulation:

```
    <GSI enable="true" portType="TCP" imageTransfer="false"/>

    <GSI name="SimA" enable="true" portType="TCP"
         client="true" receivePort="48121" server="10.34.20.1" receive="true"/>

    <GSI name="SimB" enable="true" portType="TCP"
         client="true" receivePort="48122" server="10.34.20.2" receive="true"/>

    <GSI name="SimC" enable="true" portType="TCP"
         client="false" port="48123" receive="true"/>
```

If pedestrians are part of your simulation, then add the attribute `pedestrianAnimation="true"` to the "SimA" GSI-Konfiguration in the Original Simulation (see above).

Step 3: Configuring the Environment

The different simulators shall actually each run its own simulation and shall only be linked via the dedicated taskControl links defined above. Therefore, you have to make sure that all other ports don't interfere with each other. The easiest way to do this is to modify the environment settings for simulators B and C. In each simulator's directory `Data/Setups/Current/Config/Simulation` open the file `envSettings.cfg` and change the environment variables accordingly. Here's an example:

2nd simulation:

```
setenv PORT_TC_2_TRAFFIC 50711
setenv PORT_TC_2_IG      50712
setenv PORT_IG_2_TC      50713
setenv PORT_IG_CTRL      13212
setenv PORT_TC_2_OS      50714
setenv PORT_TC_2_SCVIS   50715
setenv PORT_SCVIS_2_TC   50716

setenv PORT_TC_2_VEDYNA  50717

setenv PORT_TC_2_RDB  50718
setenv PORT_RDB_2_TC  50719

setenv PORT_TC_2_SOUND 13507
setenv PORT_SOUND_2_TC 13508

setenv TRAFFIC_VIS_PORT_TC2IG 50712
setenv TRAFFIC_VIS_PORT_IG2TC 50713

setenv PORT_VTGUI       48169
setenv PORT_TC_2_SCP    48169
```

3rd simulation:

```
setenv PORT_TC_2_TRAFFIC 50811
setenv PORT_TC_2_IG      50812
setenv PORT_IG_2_TC      50813
setenv PORT_IG_CTRL      13312
setenv PORT_TC_2_OS      50814
setenv PORT_TC_2_SCVIS   50815
setenv PORT_SCVIS_2_TC   50816

setenv PORT_TC_2_VEDYNA  50817

setenv PORT_TC_2_RDB  50818
setenv PORT_RDB_2_TC  50819

setenv PORT_TC_2_SOUND 13607
setenv PORT_SOUND_2_TC 13608

setenv TRAFFIC_VIS_PORT_TC2IG 50812
setenv TRAFFIC_VIS_PORT_IG2TC 50813

setenv PORT_VTGUI       48159
setenv PORT_TC_2_SCP    48159
```

Note: Dedicated TCP/IP ports don't need a new port number unless two or more taskControl processes are running on one computer. If all simulators are running completely on different computers, you may omit changing the TCP/IP ports in the examples above.

Step 4: Configuring the Image Generator

If you change the ports of the image generator (e.g. `PORT_IG_CTRL`) you have to inform the IG about this change. In the respective setup, adjust the file `Config/ImageGenerator/IGbase.xml` accordingly (look for section `<TAKATA>` and change the values of the control port and the host address, if applicable).

Step 5: Creating a scenario

Under the current project, create a scenario which has three external vehicles (e.g. `OwnA`, `OwnB` and `OwnC`)

Step 6: Preparing the Simulators

Start all simulators by first invoking each GUI and then loading the components. Unless the taskControls of all simulators are loaded, the other ones will report connection errors (due to missing server ports for connection).

Step 7: Starting the Simulation

In a linked simulation environment, one simulator will act as master and the other ones will act as team members. You have to tell each simulator via SCP (e.g. using the scpGenerator) which role it has to fulfill. This can be done **AFTER** the simulation on each participant has started, so if you're not sure, wait for a couple of seconds before issuing the following commands.

Commands for original simulation (running as master, controlling `OwnA`):

```
<TaskControl><Traffic masterPlayer="OwnA"/></TaskControl>
<SimCtrl><LoadScenario filename="teamScenario.xml"/><Start mode="operation"/></SimCtrl>
<Team><Master master="SimA"/></Team>
```

Commands for 2nd simulation (running as slave, controlling `OwnB`):

```
<TaskControl><Traffic masterPlayer="OwnB"/></TaskControl>
<SimCtrl><LoadScenario filename="teamScenario.xml"/><Start mode="operation"/></SimCtrl>
<Team><Slave master="SimA"/></Team>
```

Commands for 3rd simulation (running as slave, controlling `OwnC`):

```
<TaskControl><Traffic masterPlayer="OwnC"/></TaskControl>
<SimCtrl><LoadScenario filename="teamScenario.xml"/><Start mode="operation"/></SimCtrl>
<Team><Slave master="SimA"/></Team>
```

Note: if the names of your external vehicles are permanently assigned to your simulators, you may also include the `<TaskControl>` configuration lines in the above examples in the respective `taskControl.xml` configuration files.

Step 8: Having fun

That should have been it and you should now be able to see each other in the simulation. Enjoy the race!

Performance

My system gets stuck when I run imageGenerator, scenarioEditor and other outputs on the same display.

This may happen due to an overload of the XServer and/or CPU. Make sure to balance the workload on the CPUs (use the configuration variable `VI_USE_CPU` in the config file `Setups/Current/Config/Simulation/taskSettings.cfg`) and have a good gfx card available. If the problems persist, close the scenarioEditor during a simulation run unless you really need it. The effect varies with machines and gfx drivers so that it's really hard to come by.

Runtime Data Bus (RDB)

For information about RDB, its formatting and contents, please see VTD's Simulation Data I/O

Cockpit IOS (CIOS)

This is an add-on to the VTD distribution. For more information, please check the page CIOS.

Debugging

How can I create a core dump?

Usually a crashing process will not create a core dump. You can configure individual modules to be authorized to create core dumps by modifying the respective tasks start parameters. For this, edit the file `Setups/Current/Config/Simulation/taskSettings.cfg`. If the process whose core dump feature you want to enable is not listed there, then copy the respective entry from `Setups/Common/Config/Simulation/taskSettings.cfg`. In the following example, the IG's core dump is enabled:

```
# ......... IG CENTER CHANNEL .........
#
if { test $TASK_TYPE = "igCtr" } then
  set BIN_NAME      = vigcar
  set OBJ_DIR       = ( $VI_CORE_DIR/ImageGenerator/bin )
  set WORK_DIR      = ( . )
  set CMD_ARGS      = ( $VI_SETUP_DIR/Config/ImageGenerator/AutoCfg.xml )
  set START_CMD     = ( source $ENV_SETTINGS ; cd $WORK_DIR ; limit coredumpsize unlimited; ipcrm -M 0x0000816a ; $OBJ_DIR
  set USE_XTERM     = true
  set XTERM_OPTIONS = ( $XTERM_OPTIONS_COMMON -bg LightGoldenRod -geometry 80x10+508+163 -title IMAGEGENERATOR )

endif
```

Note: the only modification is the one of the **START_CMD**. It is **; limit coredumpsize unlimited**. For other modules just insert these commands at the same location.

How can I identify the reason of a crash?

Usually, our software will not crash (;-)). In the rare case when you observe a crash during operation and wish to find out what internal routine has been

affected, we are providing some instructions here. These may be especially helpful if, for example, you wrote your own plugin to the ModuleManager and want to find out whether your component is the one that causes the crash.

- load the components as usual
- open a shell
- get the process ID of the process that you wish to watch (here: moduleManager)
  - type *ps -ea | grep moduleManager*
  - note the process ID (here: <procId>)
- start the debugger
  - type *gdb*
- attach to the running process
  - type *at <procId>*
- continue the execution of the process
  - type *cont*
- wait for the component to crash
- get the backtrace
  - type *where*
- collect all the text output in the shell and send it to us for investigation
- terminate the debugger
  - type *quit*
- unload the remaining components

rdb_roadmarks.png (47.3 KB) Marius Dupuis, 17.09.2012 10:56
objectCoord01.png (131 KB) Marius Dupuis, 21.01.2013 19:56
multiSim.png (31.3 KB) Marius Dupuis, 22.01.2013 13:10
sensors01.png (90 KB) Marius Dupuis, 29.01.2013 13:04
FiguresSHM.pdf (140 KB) Marius Dupuis, 15.07.2013 10:01
DI-Guy_LicInstall.pdf - DI-Guy manual for license installation (694 KB) Marius Dupuis, 16.12.2013 11:31
roadmarks.png (229 KB) Marius Dupuis, 17.04.2014 20:03
FiguresManualO6.png (56.7 KB) Marius Dupuis, 17.09.2014 09:15
vtd189.png (93.4 KB) Marius Dupuis, 02.01.2015 14:16
vtd190.png (117 KB) Marius Dupuis, 02.01.2015 14:45
FiguresManualO7.png (61.9 KB) Marius Dupuis, 05.01.2015 19:27
FiguresManualO8.png (35.6 KB) Marius Dupuis, 26.01.2015 21:05
vtd224.png (85.7 KB) Marius Dupuis, 21.05.2015 13:59
vtd226.png (82.2 KB) Marius Dupuis, 21.05.2015 13:59
vtd225.png (88.2 KB) Marius Dupuis, 21.05.2015 13:59
vtd231.png (80.2 KB) Marius Dupuis, 23.06.2015 08:17
se_debug_mode.png (55.1 KB) Andreas Biehn, 23.07.2015 15:01
videogen.png (133 KB) Marius Dupuis, 24.10.2015 08:29
Sign40kmh.tdo - RoadDesigner file to make a Sign with pole (8.58 KB) Esther Hekele, 27.10.2015 21:11
Sign40kmh.xml - The object definition (1.16 KB) Esther Hekele, 27.10.2015 21:17
instMultiUser.sh (6.72 KB) Marius Dupuis, 31.01.2016 08:13
vtd349.png (7.2 KB) Marius Dupuis, 28.03.2016 14:33
vtd350.png (21.2 KB) Marius Dupuis, 28.03.2016 14:35
vehDynOptions.png (106 KB) Marius Dupuis, 04.05.2016 21:51
vtd251.png (127 KB) Marius Dupuis, 15.05.2016 11:43
vtd480.png (23.2 KB) Marius Dupuis, 23.03.2017 20:54
vtd481.png (25.5 KB) Marius Dupuis, 23.03.2017 20:54
vtd488.png (25.2 KB) Marius Dupuis, 10.04.2017 00:30
dummyDriver.20170601.tgz (53.2 KB) Marius Dupuis, 01.06.2017 18:03
pedestrian.png (1.38 MB) Daniel Wiesenhuetter, 09.07.2018 10:39
vtd549_1.png (23.9 KB) Marius Dupuis, 08.08.2018 08:12
vtd.2.1.0.addOns.truckCabinMovement.20180824.tgz (8.09 MB) Andreas Biehn, 18.09.2018 11:38
dynParameter.jpg (51 KB) Andreas Biehn, 18.09.2018 11:40
actros11mvmnt.jpg (92.1 KB) Andreas Biehn, 18.09.2018 11:40
actros11cabin.jpg (85 KB) Andreas Biehn, 18.09.2018 11:40
actros11chassis.jpg (95.4 KB) Andreas Biehn, 18.09.2018 11:40
complexDyn2.png (121 KB) Andreas Biehn, 25.10.2018 12:21
complexDyn3.png (138 KB) Andreas Biehn, 25.10.2018 12:21
complexDyn1.png (80.8 KB) Andreas Biehn, 25.10.2018 12:21