Course research simulator

Level: basic

CONTENTS

Workflow

When you want to create an experiment with the research simulator software, the most important components to create or use are:

- subject Id
- data to be stored and analyzed
- a set of scenario scripts
- a database of the world in which the experiment is executed

The typical and recommended workflow is:

1. Start with a world. You can either choose a world database from the set of existing databases or you can create a new one. The program RoadToolVC.exe creates or modifies world databases. See "RoadDesign.pdf" for the documentation. For most experiments the existing databases offer sufficient functionality and this will save a lot of time in your experiment design.
2. Create a scenario script. You can either choose an existing scenario script and modify that or create a new one. See "Scenario.pdf" for the documentation. The TextPad editor is used to create scenario scripts and the document "ScenarioTool.pdf" describes how you can configurate TextPad to check for syntactical errors and to compile a binary script file. The binary scripts are used in the experiment. There are lot of scenario scripts that can be found in the "Scripts" folder under "DriverTraining" and "Research". In the main script (*.scn) the database is loaded as the first statement in the file using : **Set RoadNet "databasename"**, and the elements of the roadnet are accessed in the script via interface functions.
3. Create a data description file and and experiment file. The documentation for this can be found in "ExperimentSpecification.pdf". The experiment specification file has an *.ext extention. It contains information on the subject Id (for datafile name that contains the data that are measured in the experiment), the path of the data definition file and the path of the binary scenario file.

   So, based on the requirements for the experiment you have to decide:
   - which type of road (environment) you want, and if a database already exists that you can use
   - what traffic interactions and experimental manipulation you want and if an existing scenario script can be modified and used for that
   - what data you want to sample

   Always test the scripts, databases and stored data extensively before you start the actual experiment with real subjects.
4. Run the experiment on all subjects. Start the simulator and select an *.exp file to run it.
5. Process the data. Data can be stored as binary datafiles of ascii datafiles with a fixed sample frequency (raw data) and/or as processed data stored in ascii files.

Run an experiment

If you have created or modified all required files, and all files are stored in the correct folders, the experiment can be started:

1. Start the simulator

Double-click the DriveSim3 desktop icon. This starts the StartSim.bat file. This file starts:

a. **control.exe** program, which basically is the user interface and control program that controls all Inter Process Communication via shared memory and the internal Ethernet port.
b. **traffic.exe** (which runs the scenario script , the traffic model, data sampling etc.)
c. Rendering of central channel (runs mainC.py)
d. Rendering of left and right channels (runs mainLR.py)

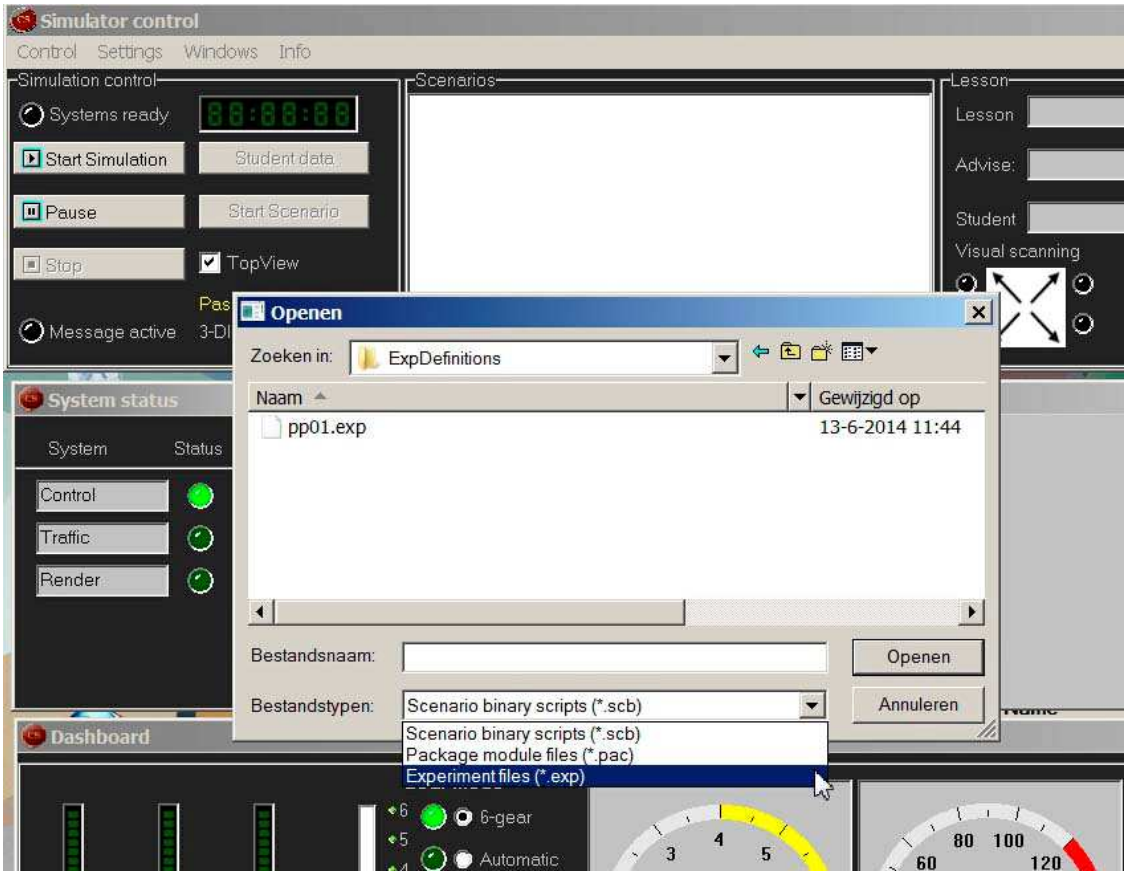
2. Load the experiment file

When the simulator is started, an experiment is selected via the "StartSimulation" button. As you can see in the next figure, two types of files can be read:

- *.scb files. These are the binary scenario scripts that are the basis of all simulations.
- *.exp files. These are experiment files created with the ExpPrep program, but can easily be edited via TextPad. This will probably be used most often when you do an experiment. The *.exp file contains the name of an *.scb file that is read by control.exe. Control.exe uses the subject Id and communicates this with traffic.exe which exposes this to the script function "SubjectIdent()", used in the "DataProc.sci" file. Control.exe also reads the data definition filename in the *.exp file, and exposes this (via traffic.exe) in the script function "DataDef()". This is used in the "DataProc.sci" file as:

```
EXT_DataDef = DataDef();  // get the data specification filename
Proc( OpenFileRead, EXT_DataDef);  // open that file for reading
```

The folder where the *.scb or *.exp files are stored, and where the file selection box initially opens in control.exe can be set in the file "scriptfolder.def", in the SimCarnet folder. So, before you start the experiment, it saves time and confusion if you set the scriptfolder.def file to point to the folder where your *.exp files are stored.



After the selection of the *.exp file for the respective subject, the experimental trial starts. If you use the *.exp mechanism and have a data definition file defined in the *.exp file, the data sampling starts and ends automatically. For example, in Distraction.scn, in Scenario 100 (Scen[100]), Action[1], there's:

```
a := ExpDataDefined();
If ( a = 1 ) {
  a := StartDataSampling();
}
```

The function ExpDataDefined() gives TRUE as a return value if a data definition file has been read (via the *.exp file). If that's the case, the StartDataSampling() function is called (which is defined in the "DataProc.sci" file). ExpDataDefined() is a 'system defined function', while StartDataSampling() is a used defined function.

Later, in Action 14, it is specified to end data sampling:

```
Define Action[14] {
  Start {
    When ( Fase = 14 );
    a := SendMessage( 102, SEND_ALWAYS, 0 ); // end of practice
    a := SendMessage( 120, SEND_ALWAYS, 0 );  // we are going to stop
  }
  End {
    When ( Action[].Duration > 1.0 );
    Fase := 15;
    a := ExpDataDefined();
    If ( a = 1 ) {
      a := StopDataSampling();
    }
  }
}
```

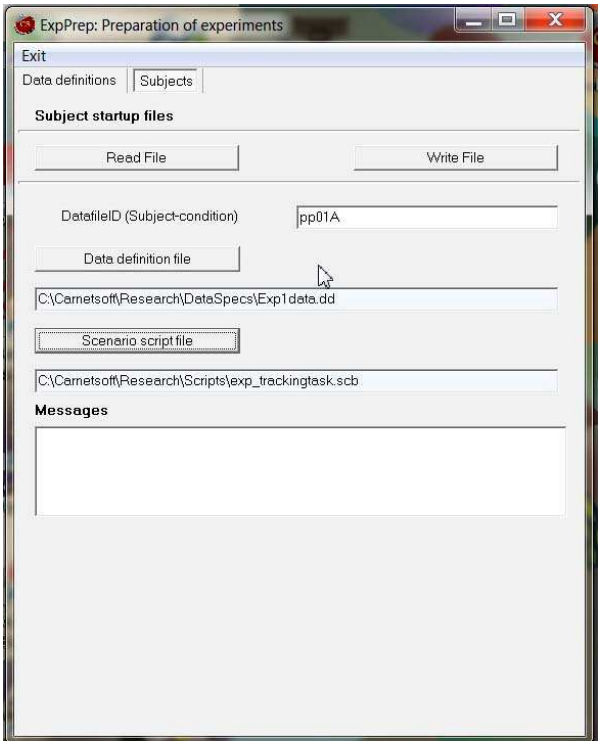
StopDataSampling() is defined in "DataProc.sci" as well.

Create the experiment files

Experiment files can best be stored in a single folder, one for each experiment. Each subject/condition combination should have a separate experiment file. Suppose you want to run 20 subjects and each subject has 2 separate runs (for example Condition A and Condition B) you will need to create 40 experiment files, for example:

- s01A
- s01B
- s02A
- s02B
- etc.

Each file can have a different scenario script or maybe even have a different data definition for the data to be sampled and stored.



As an example, suppose this is the contents of s01A.exp:

"s01A"
"C:\DriveSim3\Carnetsoft\ResearchSim\ExpPrep\DataSpecs\Exp1data.dd"
"C:\DriveSim3\Carnetsoft\Lessons\MYEXP\MyExpA.scb"

When the experiment is started with s01A.exp, make sure that the Exp1data.dd file (data definitions) is stored in the folder as indicated (C:\DriveSim3\Carnetsoft\ResearchSim\ExpPrep\DataSpecs\) and the scenario script file is located in the folder as indicated.
The datafile (s01A.da0 if you store as a binary file or s01A.dat, if you store as an ascii file) will be stored in the \data folder, directly below the script folder (C:\DriveSim3\Carnetsoft\Lessons\MYEXP\data).

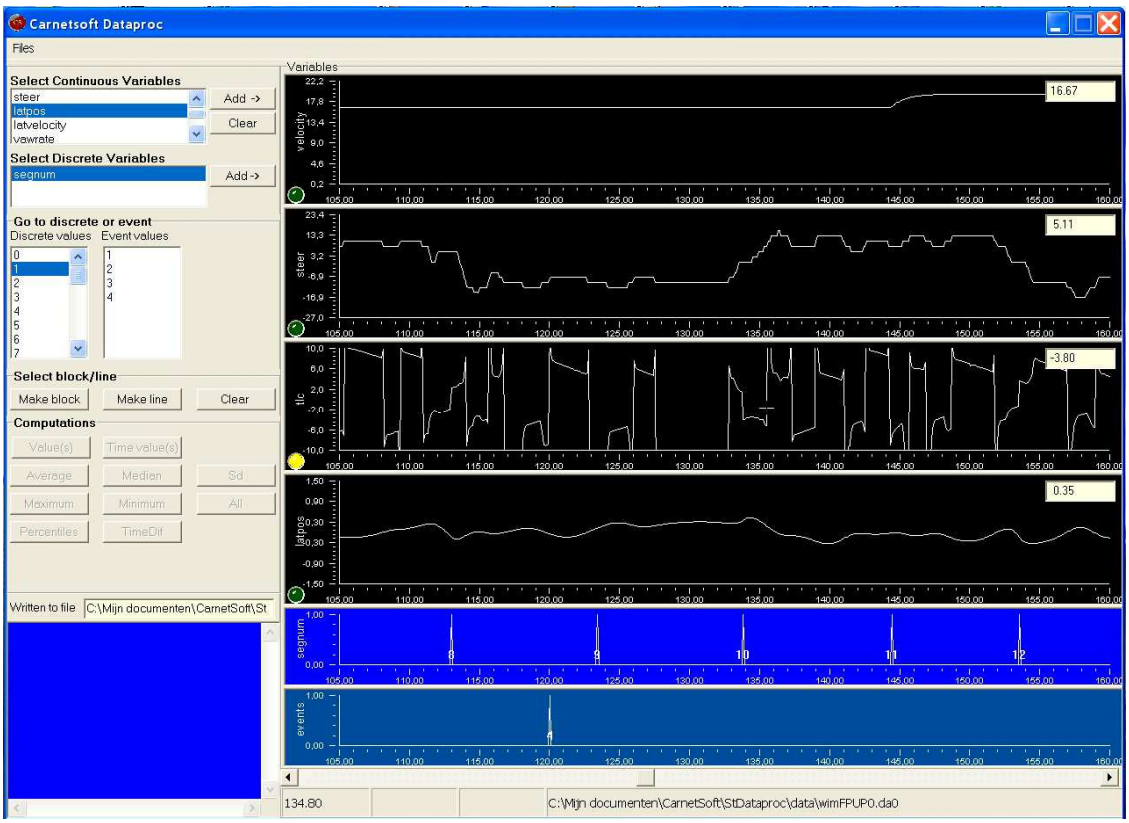
Process the data

After the experiment is completed, the data can be inspected and processed with “Dataproc.exe”. This program is for binary data files (raw data) only. In the file “DataProc.sci”, there’s a function named StartDataSampling() with the following code:

```
Define Function StartDataSampling() {
  Var { a; DataType; DataCode;}
  Proc( DataAsciiMode, 0 );  // store data as ascii if 1, else store data binary
  If ( D_DataSamplingOn = True ) {
    Proc( CloseData );
  }
  Proc( ClearDataVariables );
  Proc( SetSampleFrequency, 10 ); // sample with 10 Hz to binary file
  … etc
```

The script procedure Proc(DataAsciiMode, 0) tells the system to store the data as binary data, that can be read by Dataproc.exe. In that case, *.da0 binary files are created, together with *.evt ascii event files. The data specification is stored in the header of each *.da0 file, so Dataproc.exe knows which data have been sampled, together with the sample frequency. For more information on data storage, read “DataStorage.pdf”. When data is stored in binary *.da0 files, you can read these into DataProc.exe, and the save this as an CSV file (comma seperated ascii file), that can be read into excel or any other data analysis tool. So, usually its best to store data as binary *.da0 files.

All data files are stored in the \data folder below the script file. In the script file you can tell the system when to store events, which are stored in the *.evt files and are synchronized with the binary data.



Create or modify scenario scripts

There are a large number of scenario scripts to choose from. The folder \ResearchSim\Scripts contains all *.scn and *.sci files of the driver training program and a number of scripts with specific research simulator tasks (examples for communication with external computers, wordcounting secondary task, car following task, detection-response task or pdt, etc). “Scenario.pdf” gives detailed documentation on the script language.

Scripts are loaded by double-clicking on the *.scn or *.sci file. Make sure these file types are by default opened with TextPad and that TextPad has been configurated properly (see “ScenarioTool.pdf”).

The main scenario file has the extention *.scn. This file can have any number of other scenario files included, all with the extention *.sci. You cannot include an *.scn file. As an example (from P-Highway1.scn):

```
Set RoadNet "highwayKSA"

 // the following are defined constants
Assign ManageRoute              20
Assign RemoveClose              21
Assign AchterOpKomendVerkeer            22
Assign ConflictInvoeger      23

/*************************************************************
            Global variables
*************************************************************/

Var { NrVehicles; MaxPrio; StartRun;
    ...
}
/*************************************************************
            Include files
*************************************************************/

#Include "GenTraffic.sci"
#Include "DA_DrivingTasks.sci"   // assess driving behaviour
#Include "DataProc.sci"
....
```

So, first a database is set. Then a number of defined constants are defined and a set of global variables. And then a number of include files are referred to that contain specific script functionality used in the simulation. To explain how global variables work: for example, since the global variable “NrVehicles” is defined before “GenTraffic.sci” is loaded, all functions in GenTraffic.sci can use the NrVehicles variable.

The script refers to the data in the road database. A road database of the world contains of a number of files. The script functions only refer to the information in the “logical” database, the *.net files. It is very important that the corresponding *.net file (highwayKSA.net in this example) is stored in the “SimCarnet\scenegraphs” folder. This *.net file contains all information about intersections, Cnodes, segments, paths, lanes, traffic lights, traffic signs and roadmarkings that are needed for the simulations. For example, all vehicles use this information in their models, the information is used to position the vehicle, or to measure the distance to the road edge etc.
As an example, to position the simulator car initially you could specify the following:

```
  Part[MainTarget].PathNr       := 6;
Part[MainTarget].Lane         := DefaultLane;
Part[MainTarget].DisToInter    := 0.5*Path[6].Length;
```

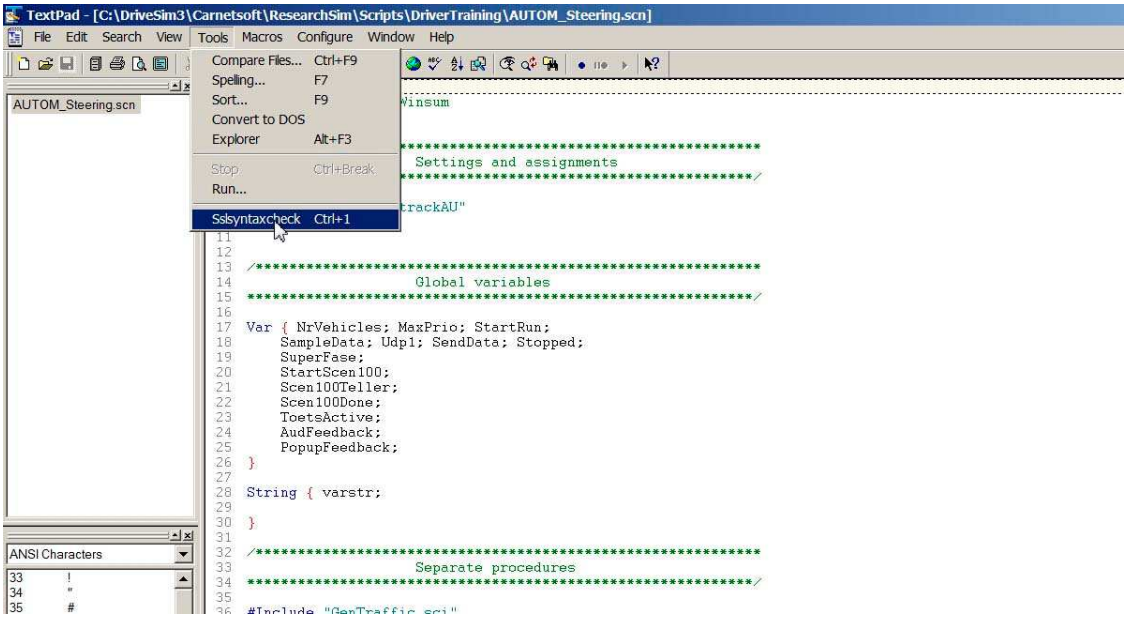
A “Part” is a traffic participant, and “MainTarget” is the simulator car you are driving in. It is positioned on Path number 6, on the default lane at a distance to the next intersection of half the path length. A path is a stretch of road between 2 intersections. So, in order to create the simulations, knowledge of the roadnet is important. You have to know the path numbers. See next paragraph.

The script files contain a lot of examples of the script language and you can look these up in the documentation. Basically, the script consists of a set of scenarios, that are defined as :

```
Scen[number] {
   Start {
        When ( condition-1);
        .....
   }
  End {
        When ( condition-2);
         .....
   }
}
```

All scenarios run in parallel, so you can have any number of scenarios active at the same time. Scenarios can have sub-scenarios (called Actions), they can have a Do block (in which the code is executed each frame). There can be specific scenarios attached to a traffic participant (called PartScen[]), etc.
To learn more about the script language, check the code and the documentation, and if there are specific questions, send a mail to info@carnetsoft.com.

The folder with the script you are working in contains a file named “encrypt.def”. If it contains the value 1, then the file is converted into a single *.scb file, a binary script file. This *.scb file must be copied to the folder where the *.exp file refers to. To create the *.scb file select SslSyntaxCheck from the Tools menu in TextPad.



If there are any syntactical errors in the *scn or included *.sci files, the editor loads the file that contains the error and brings up the line with the error and the error type. If all errors are fixed, copy the *.scb file to the proper folder before you use it in the simulator.

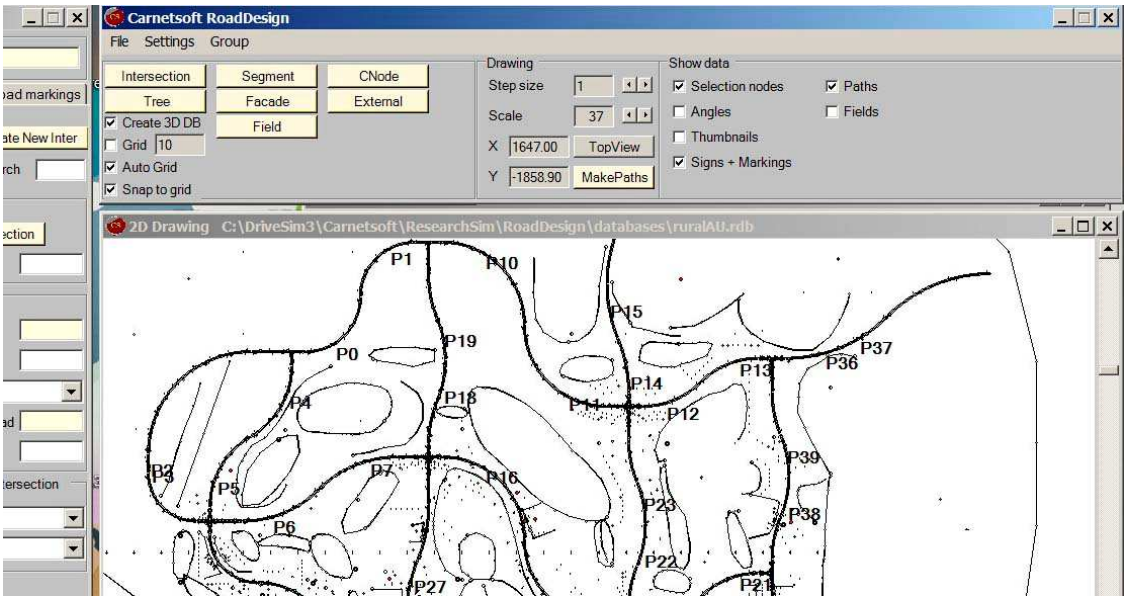
The world or road database

The world is created or modified with RoadToolVC, a database design software program. All databases are stored in the \databases folder below the folder where the RoadToolVC program is located.

The following databases are already available to start with. These can be modified, or you can create new databases.

- afbVoorrang a small village database
- curveTrack a curved track with speed limit roadsigns, no intersections, driving straight on continuously
- curveTrackNS same but without road signs
- dorp small town, zebra crossings, traffic lights, different right of way situations
- highway highway database with 3 exits and entrances
- highwayLong long continuous roadnet of highway
- inters small village database with a large number of intersections for crossing traffic in which you can drive straight on continuously (or for left and right turns), no traffic lights
- roundaboutA small town database with small roundabout
- roundaboutB large database with motorways and large roundabout
- roundabouts small database with small roundabout where you can drive 'straight on' continuously
- rural large rural database with rural roads
- ruralSnow same but with snow landscape
- ruralLong large rural database where you can drive straight on continuously
- vot small database with lots of turns

For more detailed information the reader is referred to the "RoadDesign.pdf" document.



To find out the path numbers, make sure you install a program to create snapshots (for example IrfanView), disable “ThumbNails”, enable “Paths”, and press the MakePaths button, see figure. Then the path numbers are displays on the 2D drawing. Each connection between 2 intersections (of connection nodes) has a path from A->B and from B->A: one is even and the opposite path is the next odd (for example path 2 is in one direction and path 3 is in the other direction). Even better is to make a snapshot or multiple snapshots if it’s a big database, and print the picture so you have a map.

If the database is saved, a number of files are created. One of these files is a *.path file which specifies all paths (length in meters) and the intersections or cnodes they start from and end to. The intersections can also be found in the road designer.