*iTS*MAIT
The Innovative Transportation Simulator by MAIT international User's guide (Version its-0.51)

by Jörg Schweizer, MAIT international
Email: info@MAITint.org
URL: www.MAITint.org

July 17, 2007

Preface

This is a guide to the *iTS*MAITsoftware package.

LINKS FOR DOWNLOADING ARE PROVIDED IN THE INSTALLATION SECTION
Look under the subsections that correspond to your operating system. Goto Sec. 2.2 for Windows, NT, XP etc. and Sec. 2.3 for Linux and UNIXEs in general.

Although the design and running the simulation of a PRT/GRT network appears straight forward, you can save a lot of time reading at least Sec 3!!.

Copyright statements

Contents

1 Introduction

1.1 What is *iTS*MAIT?

The innovative Transport Simulator by MAIT international e.V. is an easy-to-use micro-simulator software, intended to simulate and evaluate a set of "*innovative transportation systems*". The software has been developed according to the MAIT system specifications, and software specifications, presented at the MAIT Meeting 2001, Loughborough, UK. its-0.51 also satisfies the specifications for the PRT simulation software articulated by the EUNITRANS group during the Automated People Mover conference APM'99, Copenhagen, Denmark.

Even though the ultimate intention for this software is the simulation of a complete MAIT system, its current use is focused on PRT or AGV systems.

Essentially, **the software predicts performance and cost figures of a specified system.** The input data will include the network topology and assumed demand-patterns, maximum accelerations, line-speed, boarding-times, component-costs, etc. Output data will include waiting times, journey times, running costs and capital costs.

The performance-relevant technologies implemented are

- a novel version of a *vehicle follower control* algorithm. This control system is thought to allow close-to-physically achievable results (see also the discussion under "Carriers" in Sec. 5.1).
- Acceleration at stops, diverge and merge behavior.
- Boarding behavior at stops. Currently are two modes implemented:
  - *Synchronous mode,* where the stop is divided into a loading and unloading area.
  - *Asynchronous mode,* where users can queue up at all berth of the stop and enter the next available vehicle.
- *Shortest estimated travel-time routing.* (See Sec. 1.3 under logistics.) Considerable higher throughputs in high-density traffic operation are expected from a global logistic operation system, which is not yet implemented.

The software has been designed to obtain a *holistic view* of the simulated transportation system. For this reason it covers (or will cover) a variety of system aspects such as:

- Network design and planning, i.e traffic flow/ traveling speed/ waiting times etc. for a given network layout and origin-destination demand patterns. (See scheme of Fig. 1.)
- Cost- and performance analysis i.e prediction of passenger km, vehicle usage, expected investment and operating&maintenance costs, economy of scale, etc.
- Support for technological development of vehicles and infrastructure i.e to incorporate upgrades to modularization and interfaces, control systems, server-networks, logistics, management systems, etc.

- Support for implementation, test and integration, so that simulated hardware modules, such as vehicles and track, can be replaced by their real physical systems, while the logistics/management modules of the software can continue to control the real network— with almost no modifications.
- Support for geometrical design, study of visual impact and promotion i.e to provide 3D output of simulated network.A simple, but realtime 3D-viewer has been implemented.

For currently implemented features see Sec 1.3.

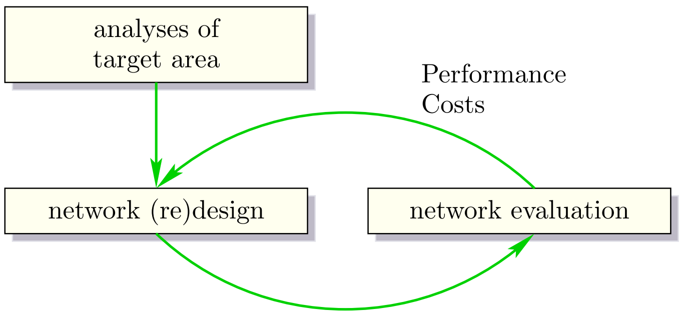


Figure 1: Network design cycle.

1.2 Who should use *iTS*MAIT?

Because the simulator covers a large number of development and implementation issues, it is of interest to:

- system developers,
- manufacturers,
- transport experts, consultants, town planners, architects,
- potential operators and
- **anybody who dreams of a better way to transport people and goods**.

1.3 Currently implemented features

- *Easy to use transportation network editor:* Track elements, vehicles and users can be selected from a library and placed and edited on the canvas with a graphical 2D network representation. Scanned maps of the target region can be used as background images. The number of different network components (track elements, vehicle-types, users) is still limited but is expected to grow rapidly. Anyway, new track elements can be easily created by copying and editing library text-files.
- *Vehicle dynamics:* The vehicles are using a *vehicle-follower* control algorithm. Speed-limits, maximum comfort/emergency/safety acceleration, brake actuation time are respected. Jerk is not simulated explicitly, but jerk limits can be introduced implicitly such that the average vehicle distances are represented correctly—as if jerks

were simulated. All parameters can be modified via GUI interfaces (in this case called "control-panels"). For more information on control issues, see Sec. 5.1 under carriers and Sec. 5.3.

- *Passenger-behavior:* Currently there are two types of passengers implemented:
  - *generic user:* this user is making one trip during the simulation, where the origin and destination stop are indirectly determined by a *origin-to-destination matrix*. See Sec. 4.1 on how to use this feature.
  - *test-driver:* This user makes one random trip after another. He has been mainly created for test purposes.
- *Logistics:* Currently three management modules are implemented:
  - Passenger management, which is the only one that interfaces directly with passengers.
  - Carrier management, allocates a vehicle to the passenger management. Optimized empty vehicle management.
  - Track management, knows the network topology, instructs all diverge points to direct the vehicle on the fastest way to the desired destination.

  For more information, see management objects in Sec. 5.1 and Sec. 5.2. .
- *Analysis and validation:* The current data about each module, i.e vehicles, users, track, managements, can be displayed via control-panels. The data contains module-dependent information about
  - Performance, e.g throughput, average speed, waiting times, traveled passenger km, etc.
  - Economics, e.g initial investment, annual costs, trip costs, etc. and **Importantly: Economy of scale.** Costs, that depend on the quantity (or length) of modules can be edited in a quantity/prices table. The software will then automatically compute the price, dependent on the size of the network, number of vehicles, etc.
  - important parameters, such as accelerations, line-speed, etc. Most of these parameters can be interactively changed.
- *Export results:* the most significant data of the current simulation, i.e parameters, performance and costs, can be exported as a tab-separated text file, which is easily imported into all spread-sheet applications such as Excel, kspread, star-office or Gnumeric.
- *Save and load:* the current state of the simulated network, including vehicles, users and managements, can be saved at any time into a simulation file. This simulation file can be reloaded and the simulation continued from the state when the network was saved.

- *Command-line mode:* the simulation of a previously edited and saved network can be launched also without graphics. In the Command-line mode simulation times can be reduced considerably. It also allows to run the simulation within scripts or as a batch process. Read more in Sec 4.3.
- *Scripting:* the simulation of a previously edited and saved network can be used in scripts. Parameterized vehicles and passengers can be added within the script and results can be saved automatically, see Sec 4.2.

2 Installation and Download locations

This section explains the installation for different operating systems (OSs). As *iTS*MAITis heavily based on the open source software packages Python2 (http://www.python.org), Python Numeric and Tcl/Tk (http://dev.scriptics.com/), it is necessary to install those packages before *iTS*MAITcan be launched[1].

DIRECT LINKS FOR DOWNLOADING ALL REQUIRED SOFTWARE ARE PROVIDED IN THE SUBSECTION THAT CORRESPOND TO YOUR OPERATING SYSTEM!. For windows, there is just one zipped file to download, which contains all the software needed!

2.1 System requirements

As *system requirement*, an Intel Processor with at least 500MHz (or equivalent), 64Mb of RAM and 15MB of disk space is recommended. The simulator can also be used with less powerful computers. However, in the graphic mode, vehicles movement becomes very slow and jerky. However, a previously edited network can be simulated in command-line mode, which is significantly faster.

2.2 Installation for MS Windows

All free packages run on 95, 98, NT, 2000, XP and Vista.

2.2.1 Instructions if open source packages are not installed

Since code.enthought.com offers a complete Python suit for windows, including all required packages, it is easiest to install just this software (instead of the individual packages):

1. Download the latest version at http://code.enthought.com/enthon/.
2. Double-click on the downloaded file and follow instructions.

Now you are ready to install the *iTS*MAITsoftware, (see Sec. 2.2.2).

[1]Even though the install instructions and downloads are only given for Linux and MS windows, the open source packages can be downloaded from the respective home-pages and compiled on almost all known platforms. For Macintosh, all binaries are available except for the Python Numeric package, but maybe somebody compiled it already, check the web-site www.cwi.nl/˜ jack/macpython.html.

2.2.2 Open source packages are already installed

Use these instructions if you have Python 2.3.4 and Python Numeric-23.1 or greater installed on your system (or followed installation in Sec. 2.2.1).

1. Download http://www.trasporti.ing.unibo.it/personale/schweizer/mait/projects/sim/downloads/its-0.51 -win.zip into a directory of your choice i.e. `My Documents \its`. On this level, a directory (containing all simulation files) with the name its-0.51 will be created.

2. Rightclick on file `My Documents \its` and select: extract all...

3. Open the its-0.51 directory and start the simulator by clicking on the file `its` (with the snake icon). If you can see the main window of the simulator (see Fig. 2), you are ready to proceed with the tutorials in Sec. 3.

Remark:
(ii): If you have already used a previous version of *iTS*MAITyou may have also previous version of Python and Numeric. These versions should work as well. However, Python 2.3.4 has been a major bug-fix release and runs indeed much more stable and also faster. So if you intend to make large simulations with several thousands of users you are encouraged to reinstall all packages. Then follow the instructions of Sec. 2.2.1.

(ii): Several *iTS*MAIT-version can coexist in separate directory trees, always named by its version. Executing the simulator of one or the other version is done by diving into the respective directory and clicking on the file `its`. The simulation files created be an older version (usually in the "projects" sub-folder), can be copied into the "projects" sub-folder of the newer version. Simulation files created by an older version, will be automatically upgraded when opened with the newer version of the simulator.

2.2.3 Installation of PyOpenGL

In order to view your network in 3D and real-time, you need to have the PyOpenGl-package installed (!! Does not work with Vista - sorry):

1. install one more Python module: download exactly this file PyOpenGL-2.0.1.09.py2.3-numpy23.exe (for Python 2.3) or PyOpenGL-2.0.1.09.py2.4-numpy23.exe (for Python 2.4). from
`http://sourceforge.net/project/showfiles.php?group_id=5988&package_id=6035`
and install this self-extracting file by double-clicking and following instructions.

2. Then you need the Microsoft's OpenGL driver, which is no Open Source but you get the binary for free from Microsoft or an open-source clone at
http://www.trasporti.ing.unibo.it/personale/schweizer/mait/projects/sim/downloads/glut-3.7.6-bin.zip
Unzip the file, and browse into the directory. You will see a file called glut32.dll. Just copy this dll-file in the main directory of the previously installed OpenGL Python package which is usually in `C:\ Programs\Python23\Lib\Site Packages\OpenGL`.

2.3 Installation for Linux

Linux installation is in the best case very easy because Python is part of most Linux distributions. If you have all the packages or if you have installed a previous version of *iTS* MAITyou can directly go to Sec. 2.3.1.

However, for the same reason it can get messy if you have some packages installed, other not, or if you are using older packages. Therefore, first check out whether the above mentioned open source packages are already installed on your system:

1. Open a terminal and type:`python`, or `python2`. If you get an error message or if you have a version lower than 2.2 then you need to install/upgrade python and all other packages. Go to Sec. 2.3.2.

2. If you have a valid version of python then type: `import Tkinter` behind the python prompt. If this produces an error message, you need to install TKinter.Go to Sec. 2.3.2.

3. If you have the right version of python then type `import Numeric` after the python prompt. If this produces an error message, you need to install Python Numeric. Go to Sec. 2.3.3.

Some quick installation method for python and the required packages are briefly described below. If you have all packages you can continue with Sec. 2.3.1.

2.3.1 Installing *iTS* MAIT

Once all packages are in place the installation of *iTS* MAITis easy:

1. Download
   http://www.trasporti.ing.unibo.it/personale/schweizer/mait/projects/sim/downloads/its-0.51 .zip.

2. place file `its-0.51 .zip` in a directory of your choice. Open a Terminal, open this directory and unpack zip archive with `$ unzip -a its-0.51 .zip`.

3. A new subdirectory called "its-0.51 " has been created; open it and start the simulator with`$ python its.py`. For command line option type `$ python its.py -h`. If you see the main window of the simulator (see Fig. 2) then the installation is successfully terminated. Otherwise, check out the messages on the terminal to see whether there is a package missing.

Remark 1: If you intend to install *iTS* MAITfor all users then you should put the entire its-0.51 directory in `$PYTHONHOME/site-packages/`. Read the documentation on how to make the new package visible to the Python interpreter.

Remark 2: If you intend do more Python programming, I recommend to use the Python text editor (and development tool) IDLE (which you can find in the Python installation tree if it is installed). There are also syntax highlighting packages for the Emacs and FTE editors. All tools can be found at www.python.org.

2.3.2 Installing/upgrading Python and Tkinter

What you need is the version Python 2.2 or higher (2.3 recommended) with Tkinter support. There are different methods to install/upgrade which are more or less easy, dependent on your present system configuration.

- *Installing with CDs of distribution:* This is easiest. Since Python and Tkinter are part of any Linux distribution that I know of. Take these CDs and install the following packages in this sequence (if not already installed):

  1. `python-2.3.3-6`
  2. `python-devel-2.3.3-6` (only if you have no RPM binary for Numeric)
  3. `tcl-8.4.5-7` (required for Tkinter)
  4. `tk-8.4.5-8` (required for Tkinter)
  5. `tix-8.1.4-96.1` (required for Tkinter)
  6. `tkinter-2.3.3-6`

  This list is made off the Fedora Core 2 distribution. In other distributions version names may slightly differ, but the important thing is that they are consistent and satisfy internal dependencies.

  RPM package installation for Linux newcomers: login as root, open a terminal, cd into the RPM-directory of the distribution CD (usually: `$ cd /mnt/cdrom/RPMS`) and type:
  ```
  # rpm -i packagename.rpm
  ```
  Alternatively use your favorite graphical RPM management application which is usually somewhere in the applications launch-bar under System tools, Admin tools or similar).

- *Upgrading:* If you have Python version >= 2.2 the current *iTS*MAITversion should work. However, Python 2.3 has been a major bug-fix release and runs indeed much more stable and also faster (up to 30% on some benchmarks). So if you intend to make large simulations with several thousands of users you are encouraged to upgrade Python. Several Python versions can actually coexist on your system, you just need to make sure which one is called when you type `$ python` or click on a python file.

  If you have an RPM based Linux system, make sure to get the upgrade RPMs for your specific distribution, otherwise you will get crazy with package dependencies. You can find upgrades on one of the following sites:
  http://www.rpmfind.net/linux/RPM/(all major distributions),
  http://freshrpms.net/packages/(RedHat oriented),
  http://dag.wieers.com/home-made/apt/(RedHat oriented).
  Other distributions like Debian or Gentoo do maintain their own database with upgrades, have a look at their home page.

- *Compile from source:* Compiled code usually runs faster than binaries because it will be compiled for your processor and not for the (lowest) standard processor i386. Download the source code for Python 2.3.4 from the download page at www.python.org/ftp/python/2.3.4/Python-2.3.4.tgz and compile. Even though the compilation process is well done and straight forward to use, people who never compiled a source package may find it difficult to go through all steps.

2.3.3 Installing Python Numeric

If you have Python up and running, choose one of these methods to get python-numeric-23.1 installed:

- *RPMs:* This is easiest, but recommended ONLY if you find the package python-numeric-23.1-xxxx.rpm for your particular Linux distribution. Have a look at the following mirrors: www.rpmfind.net/linux/RPM/(all major distributions), http://freshrpms.net/pac oriented), http://dag.wieers.com/home-made/apt/(RedHat oriented) Other distributions like Debian or Gentoo do maintain their own database with upgrades. Just have a look at the respective home page.

- *Compile from source:* This is actually the "Python way" to install Python modules, because it is platform independent.

  1. First make sure you have the Python header files installed, which should be the case when you have compiled Python from source. Otherwise,install the RPM package `python-devel-2.3.x-x` from your Linux distribution.
  2. Download source Numeric-23.1.zip at http://sourceforge.net/projects/numpy and unzip.
  3. Open terminal as root in the unzipped directory `Numeric-23.1` and type:
     ```
     # python setup.py install
     ```

  If you have problems consult the Python Numeric homepage (http://www.pfdubois.com/numpy/)

2.3.4 Installation of PyOpenGL

In order to view your network in 3D and real-time, you need to have the PyOpenGL-2.0 (or newer) installed:

- PyOpenGL is often available as binary RPM or Debian package for you Linux distribution. Sometime the package is called "python-opengl". Check out the respective repositories, see Sec. 2.3.3

- Download and compile the source from `http://sourceforge.net/project/showfiles.php?group_id=5988&package_id=6035`

3 Short Tutorials

Go through these tutorials if you are using the simulator for the first time, they help you to quickly get a simulation designed and running. For professional studies the advanced topics in Sec. 4 are highly recommended.

Start up *iTS*MAITin GUI mode (GUI= Graphical user interface) by typing `python its.py` behind the terminal prompt or by double clicking on the its-snake in your file browser. Then you should see the empty main window as shown in Fig. 2. Now you can continue with one of the short tutorials below.

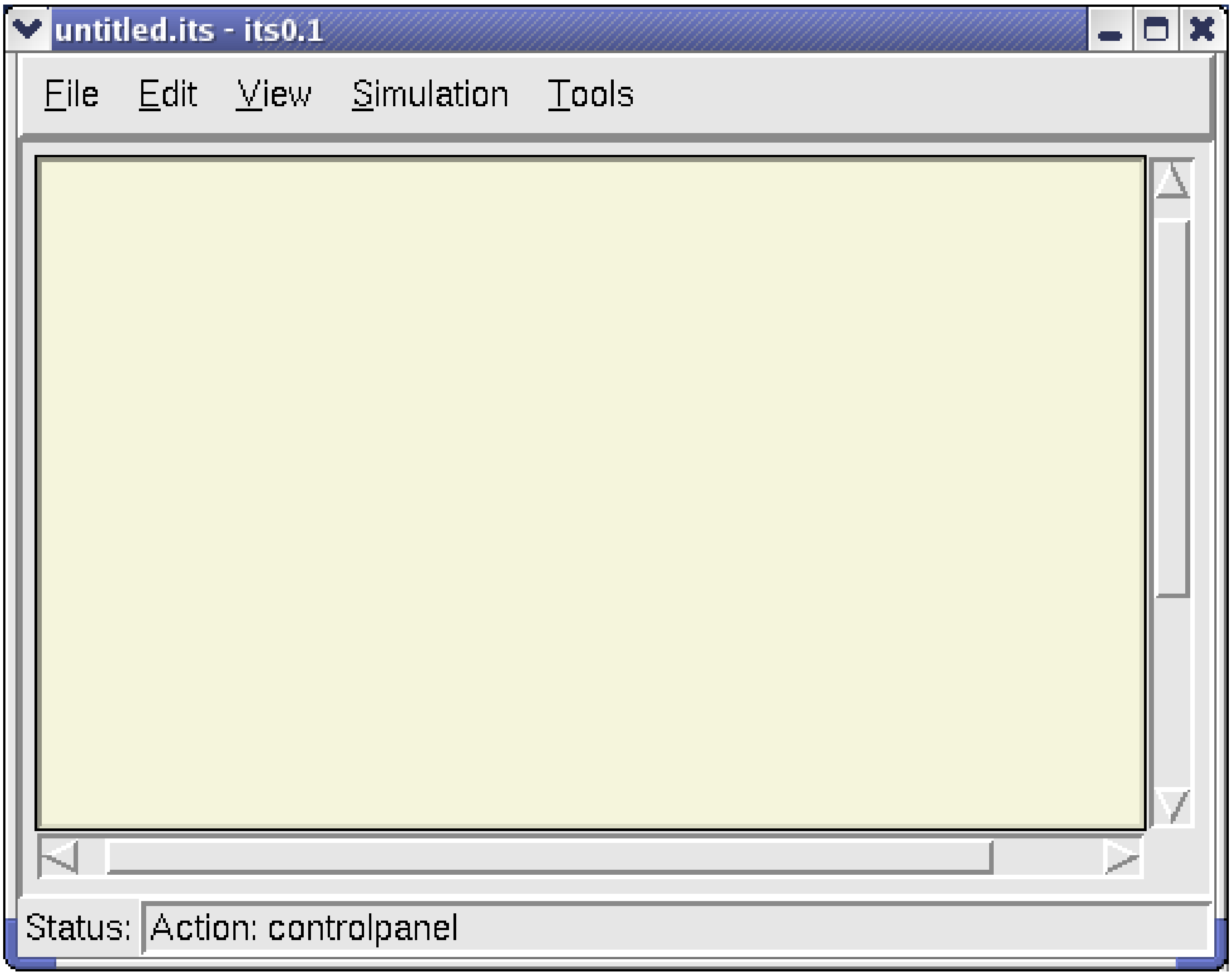
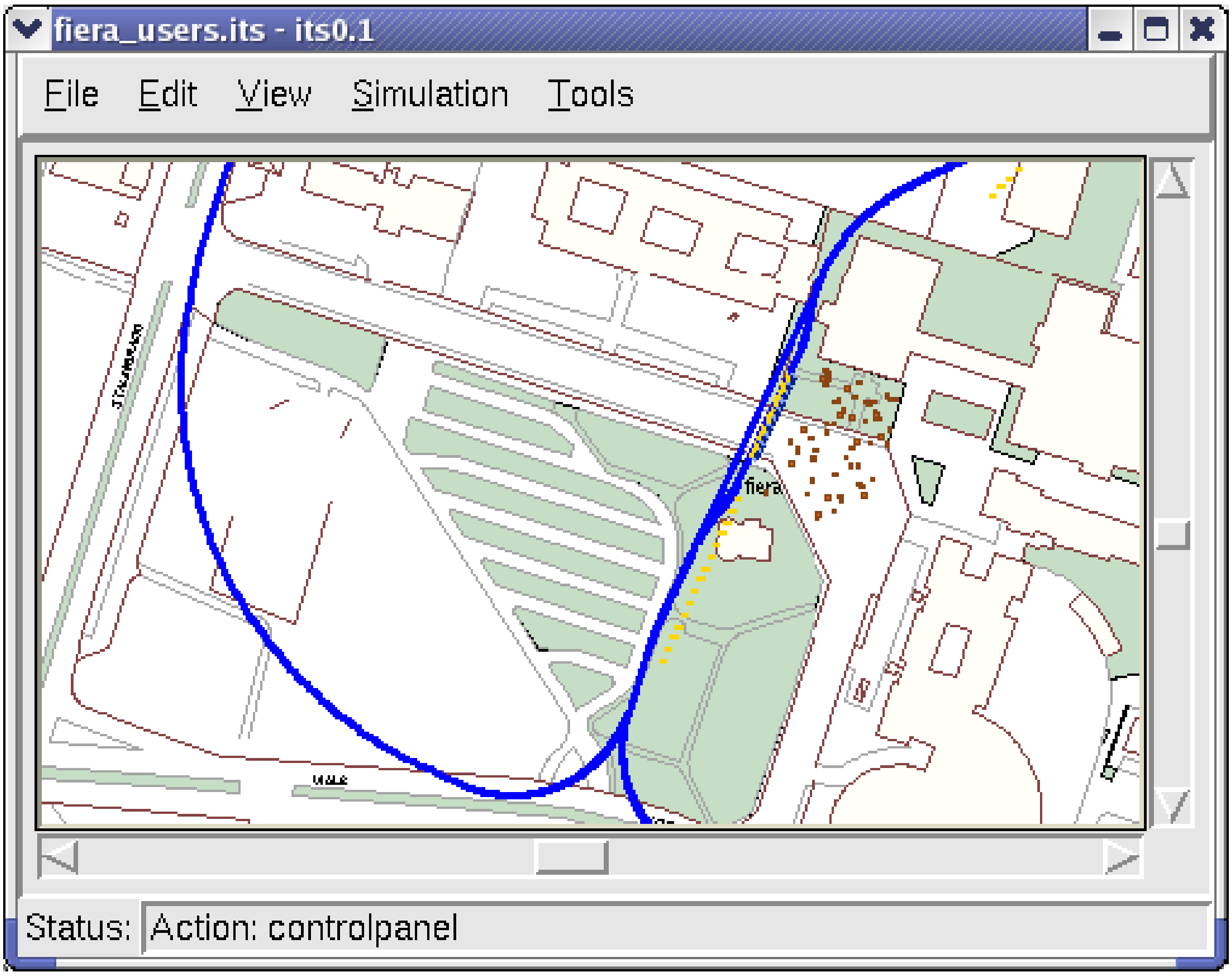


Figure 2: *iTS*MAITMain window

3.1 Opening, starting and viewing simulation files

Choose from the main menu File / Open. You will *always* be asked whether you want to save the present simulation. Press "NO" if there is no simulation or if you have just saved it. Select the simulation file `fiera_users.its` and press the "OK" button. After the transportation network has appeared in the main window (similar to Fig. 3), choose from the main menu Simulation / Start. The vehicles and users should now start moving. Stop the simulation with menu Simulation / Stop. You can *continue* the simulation with Simulation / Continue. If you select again Simulation / Start the simulation will apparently continue, but time and all statistical data will be set to zero.

There is also a *step-by-step simulation* option: click somewhere in the canvas and press the `return` key. For each `return`-key hit, the simulator will advance 500ms. Before continuing the simulation in normal mode, you need to set the `End of simulation time` to a desired value by selecting Simulation / Parameters.



Status: Action: controlpanel

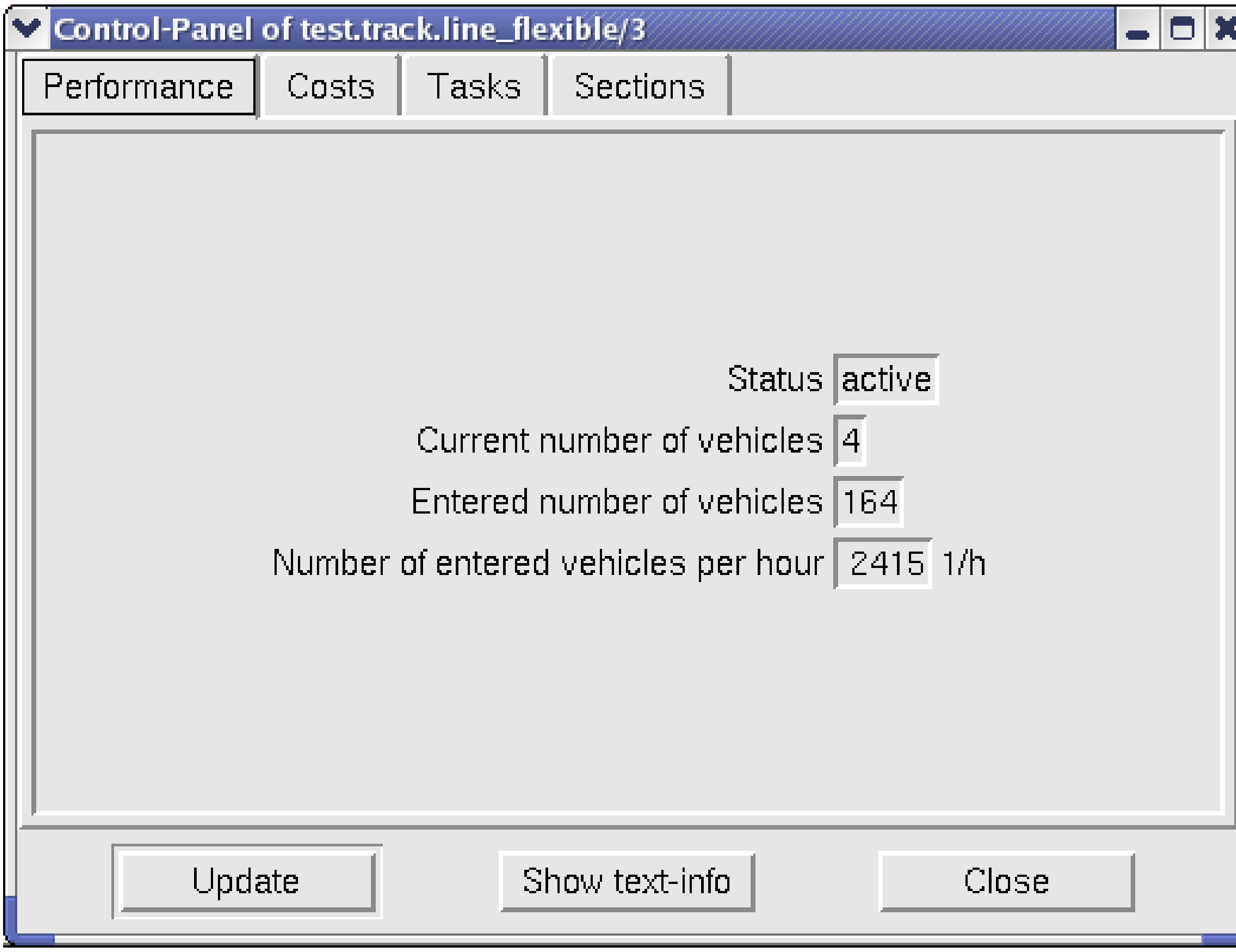
Figure 3: The simulation `fiera_users.its`. The brown dots beside the off-line stop are people (users) who will have a ride. The vehicles are the yellow rectangles.

Use the scroll-bars or cursor-keys to *scroll the viewing area* of the canvas. *Zoom in and out* with the PgDown and PgUp buttons.*Attention:* for large maps, enlargements are computationally intensive and require a huge amount of memory!! More zoom options are available on the View Menu. Choose View / indicate ghosts to visualize how vehicles are merged and de-merged. *Attention*, this option will slow down the display.

Network evaluation methods are explained in Sec. 3.2, for Network creation and editing see Sec. 3.3.

3.2 Network evaluation

Open a complete simulation file (including vehicles and users as demonstrated in Sec. 3.1) and simulate it for at least 10 minutes (see status bar below canvas). There are currently


Control-Panel of test.track.line_flexible/3

Status: active
Current number of vehicles: 4
Entered number of vehicles: 164
Number of entered vehicles per hour: 2415 1/h

Figure 4: The Performance page of the control-panel of a track element.

the following methods available to evaluate the current state of the simulation:

The Control-panel is an interactive graphical user interface which displays various information about a selected module (see Fig. 4 with an example of a track element). The information are ordered by subjects (for example performance, costs, etc.) and presented on different pages which can be chosen on the top border of the control-panel window. Many fields and buttons are interactive, values can be changed by:

- clicking in the field,
- modifying value with keyboard and
- **pressing the RETURN or ENTER button**.

There are many possibilities to open control-panels:

- Select Edit/ Module/ Control-panel and click on a module (user, track-element) within the canvas.

- Select Edit / Browse.... This will open a window with a list of all simulation objects, also the “hidden objects” (for example managements), which are not visible on the canvas. Select an object and press the Control-panel button. Some objects contain a number of individual modules of the same type (for example the object `base.user.test_driver` contains all users of the type “test-driver”). These modules are displayed on the right. In this case, select one of the modules and press the Control-panel button. A double click has the same effect as pressing the Control-panel button.
- The control-panel can also be opened from within other control-panels if there is a list-box with module names. For example, there is a list of vehicles in the `Section` page of each track element which are currently on the selected section (select a section in the section list to show vehicles). Click directly on the vehicle ID to open the control-panel of the respective vehicle.

Export results allows to write structured information about the state of the simulation into a file. Currently the data is written in table-form using a tab-separated text format. The resulting file can be directly imported into text processing documents or spread sheet applications.

To export results, select Tools / Export results... from the main menu, insert a file name with extension (for example .txt) in the upcoming dialog-box and press OK. When imported with the spread-sheet application kspread (KDE e.V) the data looks as shown in Fig. 5.

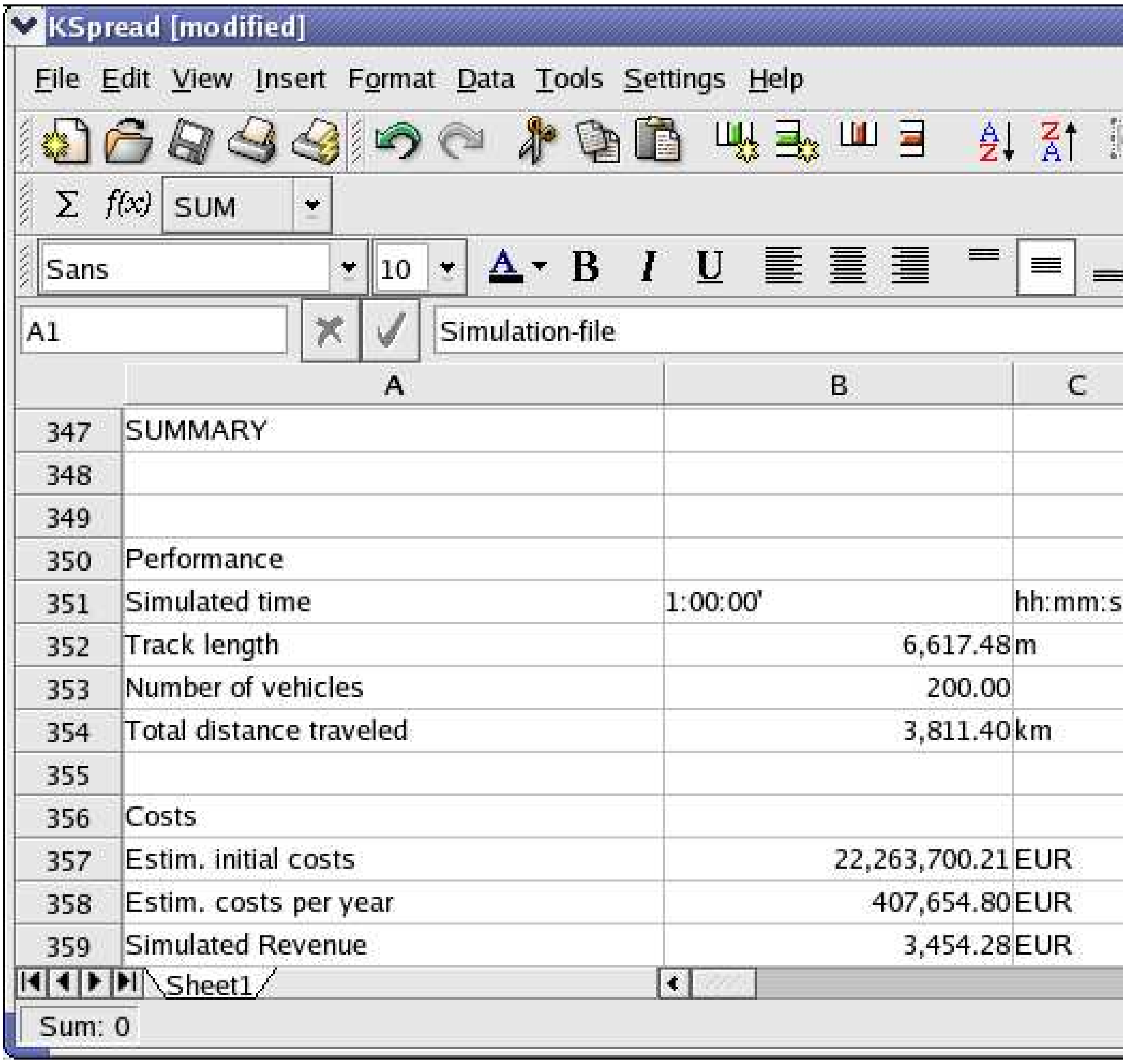
Export canvas: see Sec. 3.4.

3.3 Designing an automated transport network

These are step-by-step instructions on how to design a network from scratch. **It is highly recommended to follow these instructions** at least for the first time you create a network. Keep in mind that this is not a fully-developed commercial software with all the conveniences like undo, group, mark, copy, paste, etc. Furthermore, some parameters cannot be changed in later design steps and need to be set at the beginning. However, following the instructions below even larger networks can be rapidly and conveniently designed, see also design sequence in Fig. 6:

1. Create a new simulation with File / New from the main menu.
2. Optionally, change currency in Simulation / Parameters (default is EUR). Confusing results may occur when changing the currency later.
3. Optionally, put one or more scanned maps on the canvas: Select Edit/ Map/ Add... from the main menu, select the graphic file with the desired map in the browser-dialog window and press OK. You should see the map on the canvas, attached to the pointer. Move map to the correct position and place it with a click. The add-and-place operation can be repeated for any number of maps.

   Note that *iTS*MAITis a micro-simulator, showing the movement of individual people and vehicles. This means the map’s resolution must be high enough to show structures such as individual houses or cars, which requires considerable memory



Simulation-file

	A	B	C
347	SUMMARY		
348			
349			
350	Performance		
351	Simulated time	1:00:00'	hh:mm:s
352	Track length	6,617.48	m
353	Number of vehicles	200.00	
354	Total distance traveled	3,811.40	km
355			
356	Costs		
357	Estim. initial costs	22,263,700.21	EUR
358	Estim. costs per year	407,654.80	EUR
359	Simulated Revenue	3,454.28	EUR

Sum: 0

Figure 5: Example of file with exported results, when imported in a spread sheet (here kspread from KDE e.V).

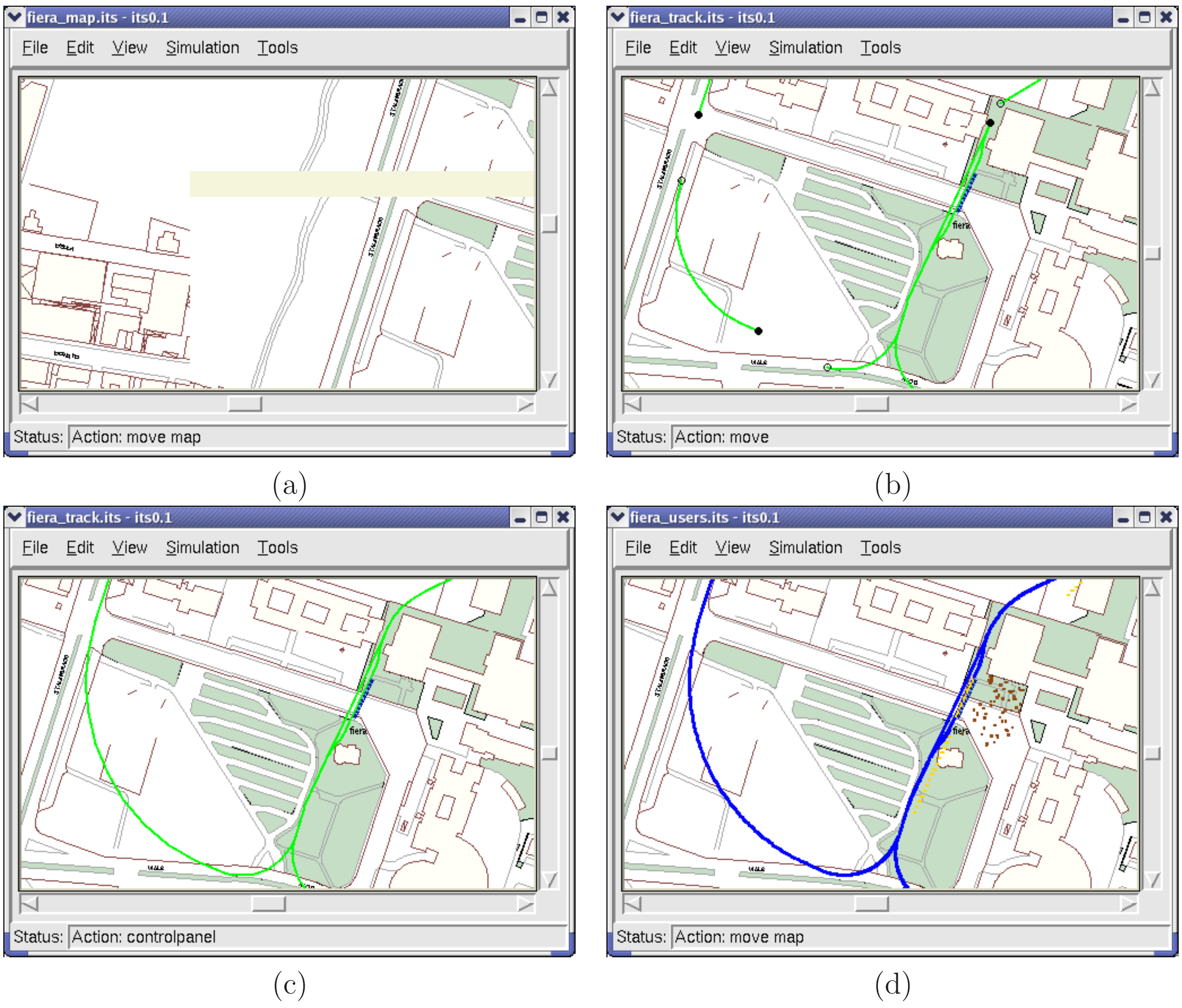


Figure 6: Example of a step-by-step network design.(a) Moving and placing background maps. (b) Placing stops and other track element. (c) Closing the gaps with "flexible" track elements. This design step is available in the its-0.51 distribution, just open simulation file `fiera_track.its`. (d) Track is "activated" and carriers as well as users have been added to the stops. The simulation is now ready to run.

and processing time for map images. In order to obtain realistic results using maps, please read carefully the following instructions:

- Scale the map to *one pixel-per-meter* (for example if you want to include a 5km^2 large map than it must have 5000×5000 pixels $= 25,000,000$ pixels). This scaling is not done by the simulator, it is required that you do this with a graphics program *before* including it in the simulator. Note that the width of vehicles, track and persons is slightly over-sized (roughly factor 1.3) in order to improve visibility at 100% viewing scale.
- Save the map in GIF-format into the same directory as the simulation file, default is its-0.51 `/projects`.
- Multiple background maps are recommended if the target area is not rect-

angular. Try to avoid “empty” map space as *maps occupy a lot of memory, disk-space and slow down zoom operations.*
*Hint:*If you are designing a bigger network it may be useful to delete all maps after designing the layout of the network and before running the simulation. This will reduce the memory occupation and increase simulation speed (in particular zoom operations). However, keep a copy of the the simulation with the maps in case you want to redesign/expand the network.

- The maps are not included into the simulation file (only their names are). This saves you a lot of disk space but if you want to copy the simulation for a friend, make sure to copy all map files as well and do not change their names or location once they are included.
- Best suited are aerial or satellite photographs of the target area. Road maps have the disadvantage that the streets are drawn wider than they are in reality. Aerial photographs give also a more realistic impression when presenting a demo-simulation.

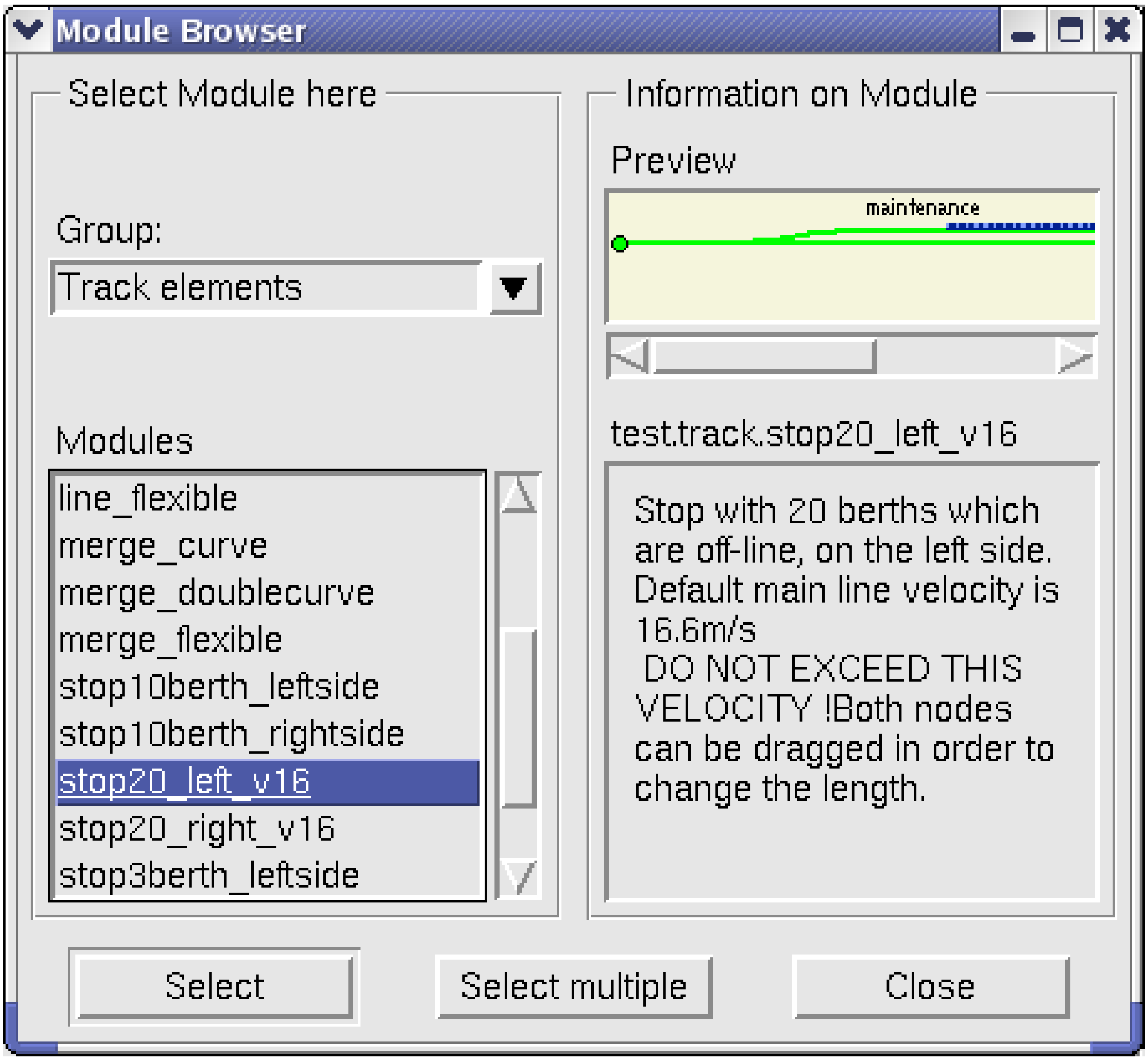
4. For your convenience: Some menus have a dashed separator on the top (for example Edit / Module). This means they can be torn-off and placed permanently anywhere on the screen as permanent window. Just select this separator to transform the menu into a window.

5. Add, adjust and connect track elements: Open first the Module browser with Edit/ Module/ Add.... The module browser, as shown in Fig. 7, has a group-pull down menu and module selection dialog box. Choose track-elements at the `group` pull down menu. Select a particular track element from the `Modules` selection box that you want to add to the simulation. An example of a selected type is shown in the preview window together with some additional text information. Note the little circles on each extremity of track elements. An empty circle indicates an input node (where vehicles enter), whereas a filled box marks an output node (where vehicles leave).

   Press the `Select` of the module-browser bottom and move pointer over canvas (alternatively, double click on the module type). The selected track element should now be attached to the pointer. As long as the track element is attached to the pointer with one of its nodes, the following operations can be used:

   - Press key “a” or shift-“a” to change the angle of the track element with respect to its attachment point (currently in one degree steps).
   - Press key “n” to cycle with the mouse pointer through the nodes. This becomes important later on when the track element must be connected to other track elements.
   - Left click to place the track element on the canvas. When input node comes close to an output node, or vice versa, the two track elements connect and the circles disappear. Note that the newly attached element may change orientation in order to guarantee a continuation from the previous element. It is just like the assembly of a model railway.

     There is immediately another track element of the same type attached to the pointer which you can place again.



Stop with 20 berths which are off-line, on the left side. Default main line velocity is 16.6m/s
DO NOT EXCEED THIS VELOCITY !Both nodes can be dragged in order to change the length.

Figure 7: The module browser. A selection panel for adding track-elements, vehicles and users. To see this panel go Edit/ Module/ Add...

- Right click or press the ESC-key to get rid of the track element attached to the pointer. Alternatively, you can click on a module in the selection box to pick a different one. You can pick up again and manipulate the geometry of the track element by selecting Edit/ Module/ Move and by clicking on the desired element.

When creating the network, it is recommended to minimize the number of track elements. This will considerably improve simulation speed. This means: use few track elements and connect them by stretching their extremities. This step requires a bit of experience. However, with the following hints one can rapidly build up a network:

- Place first all stops. Maybe you have done some traffic demand analysis of the target region and you know the rough position and capacity of the stops. Make sure that the orientation (adjust with “a” and “A”) is right and input nodes (open circles) and output nodes (filled circles) are at the desired extremities of the stop.

- Place diverge and merges elements with curves, circular curves, and line elements wherever possible.
- Change radius and length of existing elements in order to cover the maximum possible path:
  - Go in move-mode by right-click when placing or by selecting the Edit/ Module/ Move menu.
  - Click on the node you want to move or stretch.
  - Move the mouse to change shape of track element.
  - Left-click to complete operation.

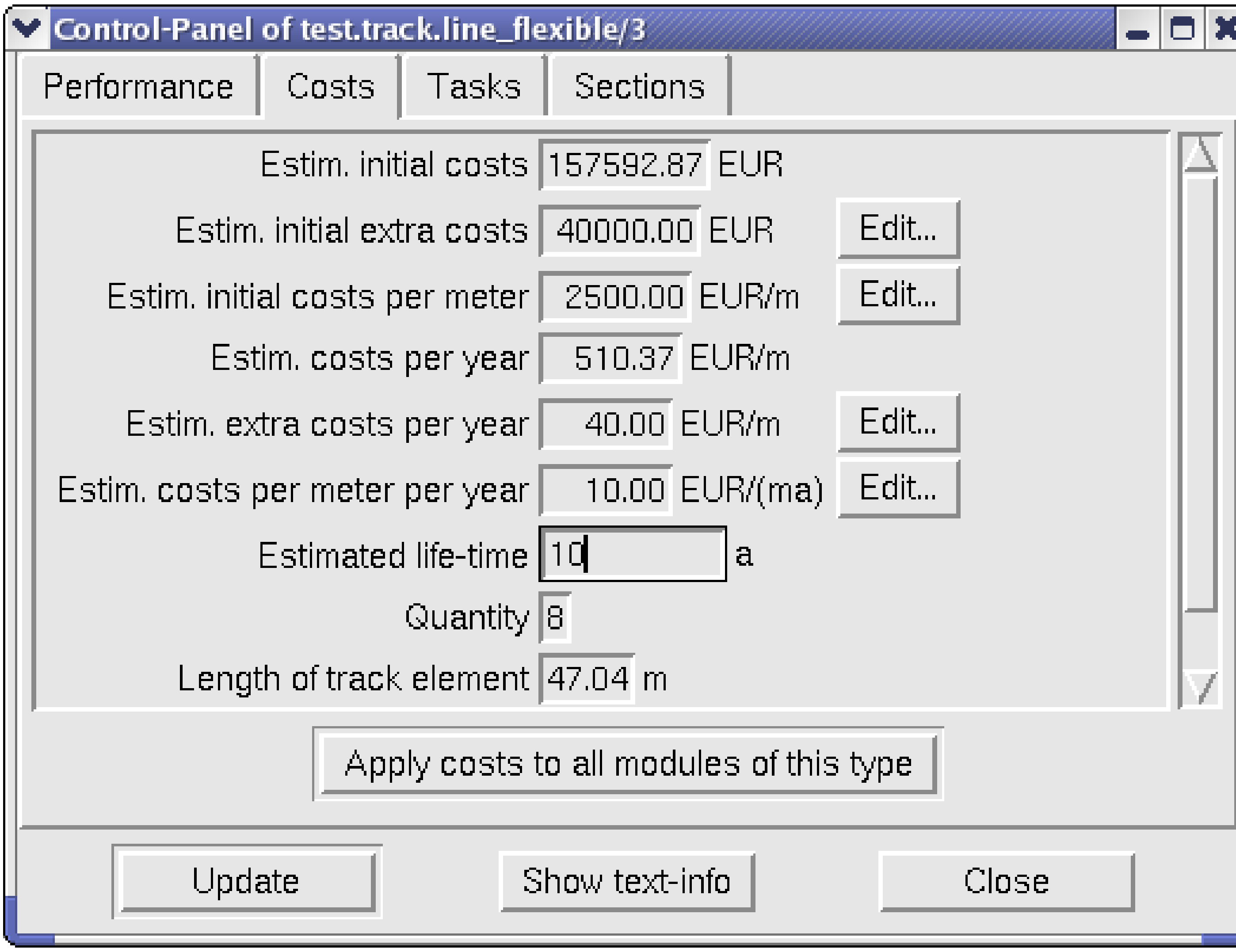
  Stretching can also be used to make connections, when a filled and empty node are close enough together.
- **WARNING: there is currently no design rule checker!** You must make sure that the entry sections of all merges are sufficiently long, this means length greater than $T_B v + \frac{1}{2a_c} v^2$, where a_c is maximum comfort deceleration, T_B is the brake actuation time and v the maximum line velocity.

  The nominal line velocity is a property of the section and can be changed on the control-panel’s `Section`-page. However, nominal line velocity is not maximum line velocity. If the previous section has a higher nominal line velocity then you must leave the vehicles additional space to reduce their speed to adapt to the nominal line velocity of the merge section. Do never increase the line velocity of stops, otherwise they are no more safe.

  The accelerations are a property of carriers and can be changed on their control-panel.
- Place diverges and merges with *flexible* extremities.
- Use again the move-mode to stretch the track elements and to close as many gaps as possible by making connections. Pay again attention to the length of the entry sections of merges.
- Close remaining gaps by placing and stretching flexible lines.
- An alternative to adding track elements one can “clone” existing elements by selecting the Edit/ Modules/ Clone tool and by clicking on the track elements to be cloned. This is similar to the copy/paste operation of text editing applications. The Edit/ Modules/ Delete tool allows to delete track-elements.

6. Optionally, configure track-elements:

   - Select Edit/ Modules/ Control-panel and click on a track-element of the canvas.
   - Click on the `Sections` page, and select an individual sections of the track-element on the table of all sections. Change nominal line velocity for the selected section. Do not forget to hit the RETURN or ENTER key after editing a number in order to make the changes valid for the simulator.

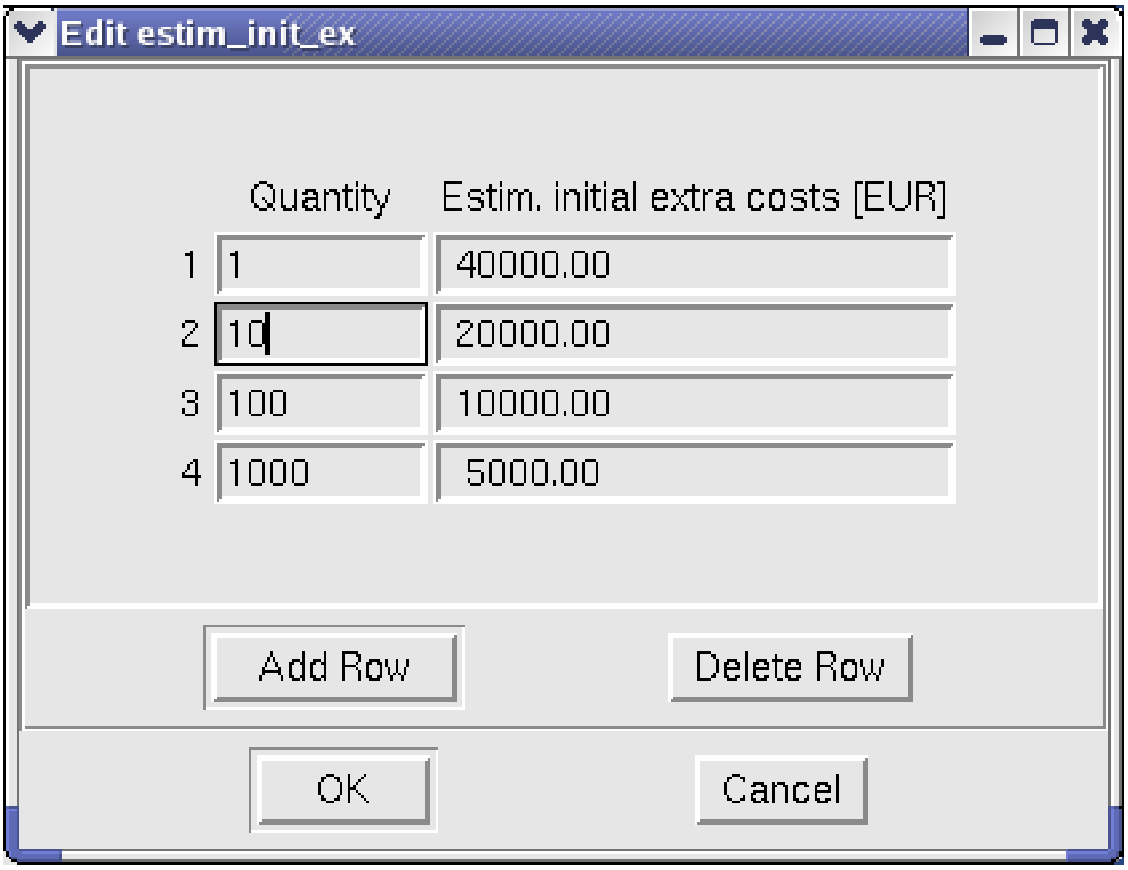

Control-Panel of test.track.line_flexible/3
Performance
Costs
Tasks
Sections
Edit...
Edit...
Edit...
Edit...
Apply costs to all modules of this type
Update
Show text-info
Close

Estim. initial costs	157592.87	EUR
Estim. initial extra costs	40000.00	EUR
Estim. initial costs per meter	2500.00	EUR/m
Estim. costs per year	510.37	EUR/m
Estim. extra costs per year	40.00	EUR/m
Estim. costs per meter per year	10.00	EUR/(ma)
Estimated life-time	10	a
Quantity	8	
Length of track element	47.04	m

Figure 8: The costs page of the control-panel of a track element.

Attention, if you want to changes velocities then it is absolutely required to read the warnings of the previous design step, in particular concerning the merge elements. It may happen that you must extend the entry section of a merge when increasing the speed!!

- Click on the `Costs` page of the control-panel, as shown in Fig. 8. You see a list of costs plus life-time and some other relevant quantities:
  - Estim. initial extra costs = All initial investment costs which are not proportional to the track length (extra poles and structures, elevators, etc.). This cost item can be edited in a quantity/cost table by pressing the Edit... button, see Fig. 9. The quantity is here the number of this type of track element in the current network, which will be automatically determined when calculating the costs.
  - Estim. initial costs per meter = All initial investment costs proportional to the track length (rails, beam-structure, poles, etc.). This cost item can be edited in a quantity/cost table by pressing the Edit... button. The



	Quantity	Estim. initial extra costs [EUR]
1	1	40000.00
2	10	20000.00
3	100	10000.00
4	1000	5000.00

Figure 9: Cost-table editor allows automatic *economy of scale calculations* for network.

quantity is here the length in meter of the total network.

- Estim. extra costs per year = All annual operating and maintenance costs which are not proportional to the track length (control, cleaning, painting, repair). This cost item can be edited in a quantity/cost table by pressing the Edit... button. The quantity is here the number of this type of track element in the network.
- Estim. costs per meter per year= All annual operating and maintenance costs proportional to the track length (cleaning, painting, repair). This cost item can be edited in a quantity/cost table by pressing the Edit... button. The quantity is here the length in meter of the total network.
- Estim. initial costs = Estim. initial extra costs+Estim. initial costs per meter $\times$ length of track-element. This cost item is a function of other costs and cannot be edited.
- Estim. costs per year = Estim. extra costs per year+Estim. costs per meter per year $\times$ length of track-element. This cost item is a function of other costs and cannot be edited.
- The life time can be changed directly in the text box, but has currently no further use.

Press the `Apply`-button to apply costs to the present track-element or the "`Apply costs to all modules of this type`"-button to copy the costs to all track-element of the same type that exist in the entire network.

- In case of stops, you can click on the `Parameters` page of the control-panel. The following parameters can be modified:
  - `Name` of the stop. By default the simulator gives unique numbers to the stops. If you want to give names to stop, try to avoid whitespace! These names may be compared later with the names in your origin-to-destination demand pattern table and a stop-name with two space-characters is different from a name with a single space-characters. You could use underscores instead, use `kings_road` for example instead of `kings road`
  - `Zone`: You can also assign the stop to a so called "zone". The concept of zones is later used to add vehicles and users. For example 50 users or vehicles can be distributed over all stops which are in the same zone. The default zone of all stops is `maintenance`.
  - `Berths to be kept clear`: This is the desired number of berth to be kept clear for newly arriving vehicles. This means an empty vehicle will not be diverted into the stop if the number of free berth is below the number specified in this field. Default is that half of the berth must be kept clear.

The above parameters can be changed at any design step, but it is recommended to do it before starting the simulation. Otherwise the obtained simulation results are produced with a changing set of system parameters.

7. **Save track layout!** You will definitely want to change your track later on. For this reason it is a good idea to save this intermediate result now because after the next steps it becomes increasingly difficult (and sometimes impossible) to remove vehicles and users from the simulation in order to get back to the plain track. Therefore, Select File / Save as... on the main menu and insert the name of the simulation. The suggested name is `mycity_track.its`, to indicate that this simulation file contains track information only.

8. Activation of track elements: The activation of track elements is somehow similar to switching on the real track after installation. To activate the track go to menu item Edit/ Module/ Activate. Then click on the track element that you want to switch on. Alternatively you can activate the entire network at once with Edit/ Module/ Activate all.

   Note: Only track elements that are connected on all nodes can be activated. However, single section of a track element can be activated if both ends are connected. Furthermore, only inactive track elements can be moved, stretched or deleted. For inactivation go Edit/ Module/ Inactivate and click on the track elements to inactivate.

9. Add vehicles: click on module-browser and select `Carriers` (carrier is MAIT terminology). You could double-click on a carrier type (currently only `experim` is available), drag it over a stop on the canvas and click to place it into a berth. However, a more effective method is to select a carrier and press the `Add multiple` bottom. A dialog box will appear with the following options:

   - Number of vehicles to be added

- The zone to where the vehicles are distributed. The default is `everywhere`.
- Maximum absolute comfort, emergency and failure acceleration rates.
  The emergency and failure deceleration define the minimum achievable headway dependent on velocity. The headway in turn will ultimately limit line capacity, see discussion in Sec. 5.3.

You can repeat this operation to place any amount of vehicles to stops in different zones.

10. Optionally, edit costs of carriers: Select Edit / Browse..., and `test.car.experim` in the simulation objects list. Click on any particular carrier in the Modules list and press the `Show control-panel`-button. Click on the `Costs` page of the control-panel. You will see a lists of costs plus life-time and some other relevant quantities:

    - Estim. initial costs = All initial investment costs This cost item can be edited in a quantity/cost table by pressing the Edit... button. The quantity is here the number of this carrier type in the current network.
    - Estim. costs per year= All annual operating and maintenance costs (control, cleaning, painting, repair).This cost item can be edited in a quantity/cost table by pressing the Edit... button. The quantity is here the number of this carrier type in the current network.

    Press the `Apply`-button to apply costs to the present carrier or the "`Apply costs to all modules of this type`"-button to copy the costs to all carriers of the same type that exist in the entire network.

11. It is recommended to save the state of the simulation file after adding vehicles with the name `mycity_cars.its`

12. Add users: click on module-browser window and select `Users`. For this exercise we will select the user "`test_driver`". Double-click on `test_driver`, drag him/her close to a stop on the canvas and click to place. However, a more effective method is to select the `test_driver` in the browser window with a single click and press the `Add multiple` bottom. A dialog box will appear with the following options:

    - Number of users to be added
    - The zone to where the users are distributed. The default is `everywhere`.
    - The boarding time, including door opening entering the vehicle and door closing.
    - The exit time. Same as boarding time, but passenger leaves the vehicle.

    You can repeat this operation to place any amount of test drivers to stops in different zones.

13. Optionally, change `End of simulation time` in Simulation / Parameters (default is $3600s = 1h$).

14. It is recommended to save the state of the simulation file after adding users with the name `mycity_users.its`

15. Run simulation with Simulation / Start.

3.4 Graphics export and printing of canvas

Unfortunately there is currently no platform-independent printing scheme for graphics implemented. However, there are two methods to export the transport-network in a printable graphics format: Postscript snapshot and screen-shots.

3.4.1 Postscript(R) Export

Postscript(R), is the Adobe(R) file-format for printers. You may be able to drag and drop a postscript file directly into a postscript compatible printer symbol, on almost all Unix systems you simply type at the prompt: `lp postscriptfile.ps` and the file will be printed directly to the postscript line printer or automatically converted to a proprietary format and then printed.

Postscript files can also be viewed and edited. There is commercial software, such as Adobe Acrobat Distiller/Reader and Photoshop. The most widespread free software is Ghostview and the Gimp. Windows users can download the free software at
`http://www.cs.wisc.edu/~ghost/gsview/index.htm`

3.4.2 Postscript(R) snapshot

To make postscript snapshots, simple select Tools / Make PS snapshots. If you now run the simulation, a snapshot will be made every $500ms$ by default. The files will be saved in the current directory under the name(s): `mysimfile_shot001.ps`, `mysimfile_shot002.ps`, `mysimfile_shot003.ps`, ...

Note: the PS snapshots cover only the area of the canvas that you actually see in the window on the screen (but without window borders). If you want the whole network, you may want to zoom out first.

3.4.3 Screen-shots

Screen-shots are probably the quickest and simples way to export graphics:

MS windows:
1. Maximize window with canvas.
2. Press the “Print Screen”-key
3. Open MS Paint or other MS text/graphics applications
4. Select Edit / Paste, or in MS word Edit / Paste Special....
5. Optionally, edit graphics (cut off windows frame and menu, resize, compress, etc.)

Linux:
1. Maximize window with canvas.
2. Take a snapshot using your favorite snap-shooter (for example ksnapshot that comes with the KDE desktop) and save window as graphics file.
3. Optionally, edit graphics with gimp (cut off windows frame and menu, resize, compress, etc.)

4 Advanced topics

In contrast with the tutorials, the advanced topics will explain how to use the simulator to a fuller extend. Many topics are very useful for professional studies. The Principles of operation in Sec. 5.1 are not fundamental for the usage of the simulator, but may help to get a deeper understanding.

4.1 Add users according to a predefined origin-to-destination matrix

The evaluation of any planned transportation system starts usually with an origin-to-destination matrix (ODM). From this matrix we know how many passengers want to travel from and to a specific place in a certain time window (usually during peak hours). There are planning-methods that allow to get from the ODM (together with other data on street-layouts and geographical data) to a transportation-network layout with nodes, links and stations, where the links and station must have certain capacities in order to cope with the expected flow of vehicles and passengers. However, note that PRT/GRT networks have some special properties such as off-line stops. Usually the first layout is based on static flow averages, which is a crude, but useful assumption.

In any case, one would start with an initial topology and then verify with the micro-simulator whether the respective performance satisfies our expectations. If not, bottlenags need to be identified and the topology modified accordingly.

Here, we explain how to use *iTS*MAITfor this iterative optimization process, in particular on how to include a predefined ODM in a network layout.

1. Design of initial network: In order to simulate a transportation network with a specific ODM, you obviously need to design the network first. Follow the steps in Sec. 3.3 up to the point where you have saved the simulation with carriers, for example under the name `mycity_cars.its`.

2. Prepare file with the ODM data: The required ODM file is an ordinary text-file with a comma-separated table:

   ```
   ,SN1,SN2,SN3,SN4,SN5,SN6,SN7,SN8
   SN1,P11,P12,P13,P14,P15,P16,P17,P18
   SN2,P21,P22,P23,P24,P25,P26,P27,P28
   SN3,P31,P32,P33,P34,P35,P36,P37,P38
   SN4,P41,P42,P43,P44,P45,P46,P47,P48
   SN5,P51,P52,P53,P54,P55,P56,P57,P58
   SN6,P61,P62,P63,P64,P65,P66,P67,P68
   SN7,P71,P72,P73,P74,P75,P76,P77,P78
   SN8,P81,P82,P83,P84,P85,P86,P87,P88
   ```

   In this case we have 8 stops, with names `SN1`...`SN8`. These are the stop names that you have inserted at the control-panel of each stop. If not, do so by selecting Edit/ Module/ Control-panel; clicking on the stop on the canvas you want to give a name; choose the `Parameters` Tab at the control-panel, fill in the `Name` field AND

HIT THE RETURN KEY to take over the value. [2] By default the stop-names are unique numbers $(1, 2, 3, \ldots)$. During the simulation `Pmn` passengers will travel from stop with name `SNm` to the stop with name `SNn`. You can find an example ODM file under `projects/fiera_odm.txt`

You may have the ODM data already available in electronic form, in a spreadsheet for example. In this case, just save it as comma separated text file and you are done.

3. Add users to simulation: Now we want to add the users to the already existing simulation, containing track and vehicles. Open the simulation into *iTS*MAIT(for example `fiera_cars.its`). Open the module browser with Edit/ Module/ Add... and select group "`users`". Select the "`generic`" user-type and click on the `Add multiple` bottom. Then a dialog window shows up with the following options:

   File with O/D matrix Insert path of ODM file or select the file using the file browser.

   Start trip earliest,Start trip latest These two fields specify the time-interval in seconds when the trip of the users must start. The actual start of the trip of a particular user will be picket from this interval (using uniform distribution).

   Minimum boarding time, Maximum boarding time These two fields specify the boarding time-interval. The actual boarding time of a particular user will be picket from this interval (using uniform distribution).

   Minimum exit time, Maximum exit time These two fields specify the exit time-interval. The actual exit time of a particular user will be picket from this interval (using uniform distribution).

   Aura : This is simply the diameter of the dot of the 2D representation of this user-type.

   Color : By clicking on "Pick..." you can select the color of the 2D representation of this user type.

   Load user-profile, Save user-profile : You can save and reload the above parameters in as user-profile.

4. Press "OK".

With the above user-parameters you can simulate different user-types, as for example single adults, passengers with luggage or a user-group such as a family. In the latter case one user stands for the entire group.

Users of several user-types (with different ODMs) can be added sequentially by repeated applying the Edit/ Module/ Add... process.

Note that during the start trip time-interval, all passengers in the ODM matrix will start the trip. This means `Pmn` passengers will start the trip from stop with name `SNm` to the stop with name `SNn` from time `Start trip earliest` to `Start trip latest`.

Note that you will not see any user on the display unless the simulation time passes the time defined in `Start trip earliest`. Then users will start queuing up at the stop of their origin.

[2]Please do avoid white-space within the stop names, is can cause mismatches when the stop-names in the ODM file are compared with the stop names in the simulation file.

4.2 Scripting

Scripting is a powerful feature in order to automatize the process of an entire series of simulations. The simulator itself is written in Python which is a object-oriented script language. For this reason, it is straight forward to access most simulator internal variables and functions via a script.

You would use scripts without graphical interface for large scale simulations because:

- it is many times faster (the fiera simulation about 5 times as fast).
- it requires less memory.
- you can run the simulation within a script or in the background.
- you can automatically create and save reports.

A full description on what can be done with scripts would result in a manual by itself. Instead, there are some example scripts in the main directory that you can execute just as the main program. Python is an extremely clean language and you are invited to use the examples as a template which can copied and modified for your specific purposes. The simplest script is called `script_users`. It runs the simulation of the same transportation network with different number of users. `script_odms` is slightly more complicated as it automatically simulates the same network with a different number of vehicles, using different origin-to-destination matrices.

In order to view/modify the script, open it in a text editor (possibly with python syntax highlighting, such as emacs). The MS-windows distribution of Python contains already an appropriate editor called IDLE: Just right-click on the script file and select `edit with IDLE`. With IDLE you can also run the scrip: select menu item Run / Module.

4.3 The command-line mode

With the command-line mode you can run a simulation without graphical interface and animations. You will appreciate this simulation mode if you want to simulate a large network with many vehicles and passengers because:

- it is many times faster (the fiera simulation about 5 times as fast).
- it requires less memory.
- you can run the simulation within a script or in the background.
- you can automatically create and save reports.

Assumed you have edited a network, added vehicles and users and saved the simulation under `mycity_users.its`. Now you want to run the simulation for 1h, save the simulation afterward in file `mycity_1h.its` and a report of the results in file `mycity_1h.txt`. To do all this with just one command: open a terminal (or DOS-prompt under windows) and type:

```
python its.py -c -e3600 -o projects/mycity_1h.its -r projects/mycity_1h.txt
projects/mycity_users.its
```

If no simulation time is given (`-d` or `-e`) option then the simulation internal times are used as set in the menu Simulation / Parameters. If you do not trust what has happened during the command-line simulation, you can always have a look at the result by opening the output simulation file (here`projects/mycity_1h.its`) in the simulator.

Have also a look at other command-line options. For example with the `-d` combined with the `-f` option, one can iteratively advance the simulation by a given amount of time and save the output simulation file back into the input simulation file. In this example the simulation `mycity.its` is advanced by 1 minute each time the following command line is executed.

```
python its.py -c -d60 -f projects/mycity_users.its
```

The currently available command-line options can be retrieved with the command line `python its.py -h`.

Note: under MS-windows the command-line mode is painful because the standard DOS-prompt shell is not very sophisticated. You also need to add the python.exe in the application search path and you need to give the absolute path of the simulation and output file. Recommendation: better use scripting under windows.

4.4 How to simulate failures

Here is how you can simulate the most important failure: a vehicle stops moving on the track:

1. Open a complete simulation, including vehicles and users. You may also run the simulation for a certain time to bring the network in a realistic state. Then stop the simulation at the time when you want to simulate the vehicle breakdown.

2. Select Edit / ModulesControl-panel.

3. Click on a track-element, where you want to simulate the breakdown and select the "`Sections`" tab.

4. Select a Section of the Sections table where you want to simulate the event: On simple line or flexible line elements, there is just one section. On diverge elements, the first section is the input-sections, the others are output. On a merge-element, the first two elements are the input section, the last one is the output-section.

5. Click into the "`Line speed`" field and change it to zero, or a small value AND HIT THE RETURN BOTTOM, otherwise the new value will not be effective. Click on the `Update`-bottom to see if the new values have been registered. You can leave the control-panel open for further interaction.

6. Continue the simulation and watch how the vehicles are slowing down on the selected section.

7. After a certain simulation time (which is necessary to repair or remove the vehicle), stop the simulation again and use the above described method to restore the line-speed to its old value.

4.5 3D OpenGL monitor

This is only a very crude 3D visualizer, but it is in real-time. When sitting on a vehicle one can get a kind of feeling for what would happen on a real ride. This applies only to smaller networks (a few km) and only with a small number of users. Unfortunately, the OpenGL monitor is currently not integrated in the main GUI as it requires a different scheduler. You can double-click on its3d to see a demo test-track. However, to view your own network is a bit more complex.

Here is the way to view your own network in 3D:

1. Edit a simulation in 2D as described in Sec. 3.3, including vehicles and users.
2. Save the simulation.
3. Right-click on the script file “its3d” in the main directory and select `edit with IDLE`.
4. Look at the script file. After the Usage info you will find a section “DEFINE DEFAULTS HERE”.
5. change the variable `simfile_name` to the filename of your simulation.
6. Use Run / Module or press F5 to start the scene. After saving your changes, you can also quit and run the 3d monitor by a double-click on its3d, because now the default simulation is your network.

Here is how to use the monitor:

```
Initially you will see the network from above in observer mode.
You have the following function-keys:

Key      Function
r        switch to observer external observer mode.
u        sit on first vehicle
b        move with first vehicle but below the guideway
s        stop simulation
c        continue simulation
Esc      Quit application

The following navigation keys are only valid for observer mode.
8        move forward
2        move backward
6        move right
4        move left
9        move in
3        move out
To rotate around z and y axes: double-klick in 3D window and move mouse
while keeping left mouse-button pressed.
```

Tip: Use the Number-block on your key-board to navigate.

5 Principles of operation

The simulator as a whole is a complicated piece of software with many interacting objects. Even though details of the internal functions are hidden away behind a graphical user interface, it may be required to have a minimum understanding of the simulator's operating principles.

The basic principle of the simulator is quite simple: a simulation consists of a set of simulation objects. Each object contains:

- internal state variables.
- methods to change the internal state.
- methods to interact with other simulation objects.

For efficiency reasons some simulation objects handle and update all modules that are of the same type[3]. Hence, these simulation objects contain additionally:

- modules, all of same type. Modules are similar to simulation objects as they contain
  - internal states
  - methods to to change its state, and to interact with its parent simulation object and other modules.
- methods to interact with modules.

The methods to change the internal state of objects or modules are event-driven. There are principally two types of events:

1. During the simulation is running (after Simulation / Start or Simulation / Continue have been selected) the simulator calls cyclically the update method of each simulation object (see Fig. 10).The update method may then call various methods with parameters of other simulation objects and modules. The time step between two cycles is determined by the sampling-time which can be set in Simulation / Parameters.
2. Object methods are asynchronously called through graphical user interfaces (dialog boxes, control-panel etc.).
3. external calls of methods (parameter settings, etc.)

All (relevant) simulation objects and modules can be listed and inspected with the object browser: Select Edit / Browse.... As it can be seen in the lists, simulation objects as well as modules have unique identifications (IDs). The naming scheme for simulation objects is:

`family name . category . type name`

The naming scheme for modules is composed of the parent simulation object ID plus a serial number:

`family name . category . type name / serial number`

[3]this structure allows to use high speed numeric array operations

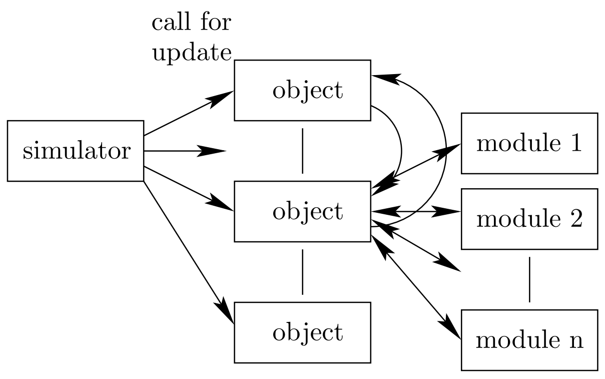


Figure 10: Simplified functioning of simulator: Simulator calls the update method of simulation objects. Some simulation objects contain a set of modules which are all of the same type. Because modules of the same type have the same functionality, very efficiently numeric- array oriented updates are possible.

5.1 Simulation objects, modules and their functions

This section gives an overview of the functions of simulation objects and modules and their relation to other objects, see Figure 11.

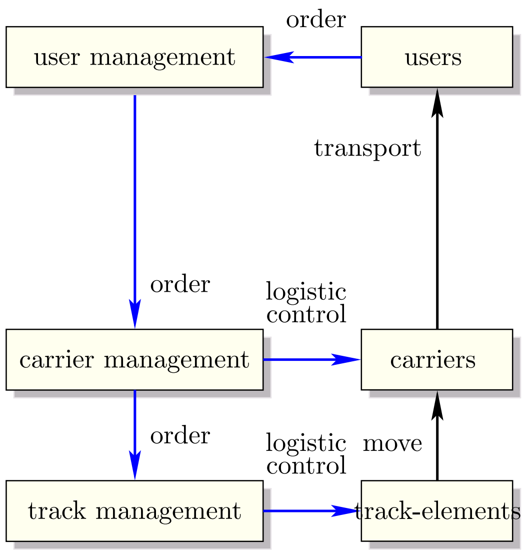


Figure 11: Simulation objects and their basic relations. Note that the users, carrier and track-elements are container objects for several modules.

users make trips with the transportation system by giving trip orders with destination to the user management and entering the carrier. There is currently just one user-type: the test-driver, making one random trip after the other.

carriers (vehicles), transport users along the track. They receive orders from the carrier management to let board a specific user. Carriers also control , together with the current track-element, the distance to other carriers and so called "ghost carriers" on parallel branches in conflict zones.

The *vehicle-follower* control algorithm is currently more optimized for simulation speed rather than for precise/optimized motion control. This means the following differences compared with a realistic control system: (i) there are no jerk-limits (in neither direction); (ii) speed changes of the first vehicle in a vehicle chain is almost instantly and fully propagated through the chain, which results in a reduced ride comfort; Differences (i) and (ii) result in a slightly higher line capacity of the simulated system compared with a real system.However additional time headway, necessary for jerk adaptation and be simulated by increasing the brake actuation time, which is one parameter of carriers.

Speed-limits, maximum comfort/emergency/safet acceleration are respected. All parameters can be modified via `Add multiple` bottom or control-panels.

track-elements are capable of guiding carriers along a predefined path. The instructions whether to diverge the carrier in a specific branch, to halt it or to load/unload it, are obtained from the track-management and passed on to the carrier when it enters the specific track element. The track-elements are organized in *track clusters* (currently all the net is one cluster). Each track cluster is controlled by one track-management and may contain one or more of the following track elements:

- *lines*, curves or flexible line that guide and control the carrier.
- *diverge*s, where one line splits into multiple lines.
- *merge*s, where multiple lines merge to one line.
- *stop*s, where people or freight gets loaded or unloaded. Stops are also organizing the halt position of carriers, and tell them where to load, to unload, to wait and when to start again. The stop does all that, dependent on which berths users are waiting.

user managements: provide a costumer friendly interface between user and the transportation system (this interface is currently not visible). User managements receive orders from the users and organizes carriers for this trip by giving orders to the carrier management. User managements would be operated by service providers (in the current version there is just one user managements).

carrier managements: handle the logistic control of a carrier fleet (in the current version there is just one carrier management). They allocate a carrier from the carrier fleet they control and give orders to the track management. They also optimize the redistribution of empty vehicles. Carrier managements would be operated by transport providers.

track managements: handle the logistic control of a track cluster (in the current version there is just one cluster). Track managements receive orders from the carrier managements on where to route the carriers. They estimate the path with the shortest trip time from origin to destination and give branch and halt instructions to all track-elements involved. Track managements would be operated by infrastructure providers.

5.2 Logistics and tasks

All managements together (see Fig. 11) perform the logistics of the network by exchanging information among each other, but also with the physical world (track-elements, carriers, users).

Apart from standard administrative operations, the managements currently optimize only the trip-time: They route the vehicle to its destinations on a path that has the shortest expected trip-time. If a branch is blocked, the vehicle is deviated, which will trigger a rerouting. Empty vehicles will be "intelligently" routes to the nearest stop with a higher user-demand.

If there is are no passengers at a stop, empty vehicles are send into the network. An empty vehicle is branched into a stop if there are waiting passengers at this moment. There is currently no predictive resource allocation—not for vehicles and neither for the network. This means traffic congestion can occur if the network has a bottleneck.

The organization of the entire logistic is based on the creation, exchange,and execution of tasks: If a management wants to get a service performed by a target (another or the same management or physical module), it must send an order. When the target receives the order, it will create a task for this order. The task inside the target will create one or several actions (or create orders for other managements and/or modules) which are necessary to fulfill the task. The task persist in the target until it is terminated, failed or explicitly killed. In either way the task will report the results to the entity which created the order.

Tasks are the instrument to control the network globally on large time scales. They are the bases for doing distributed, parallel computing in the system's heterogeneous computer network. Note that tasks are a inherent feature of most operating systems. The concept of tasks used in the simulator has been designed to fulfill the special requirements of the transportation network, including physical objects and a multi-level management. In later versions the simulation objects (in particular the management objects) could run on different processors, using the task mechanism of the respective operating system.

The task page of each control-panel shows task ID, status, the order ID and the object/module which created. Modules have usually only one task at a time, management units must handle many tasks simultaneously.

5.3 Local vehicle control

Before explaining the simulated system, a few words to the general PRT control-problem. A controller for short time head-ways needs to meet simultaneously criteria for safety, string stability, comfort and speed transitions.

The *safety criteria* can be expressed by means of the *minimum safety distance* d_{S} between vehicles that is required to avoid a collision between two consecutive vehicles. In general, d_{S} is proportional to the difference between the squared velocities of follower and lead-vehicle. In many control schemes, the level of safety is given by the safety factor K_{S} when the actually achievable distance is $K_{\mathrm{S}} \cdot d_{\mathrm{S}}$.

String stability means that a speed change of a vehicle in the string decreases in magnitude when propagating through the following vehicles.

The *comfort criteria* sets usually limits to the maximum allowed manœver accelerations and jerks.

Speed transitions do occur during various vehicle manœvers, such as approaching, following, merging, diverging, stopping or velocity changes. It is obviously required to guarantee all the above criteria during all possible speed transitions. One specific transition criteria is the "fixed end-point speed transition"(FEPST) criterion:all vehicles in the string must be at the lower safe speed at a fixed point of the track, not just the first vehicle.

The *optimization criteria* for lateral-controller is usually to maximize the line capacity or to minimize head-ways. The theoretical limits of carrying capacity and safety-level are generally set by the adopted *vehicle spacing policy.* The relevant policies are the *constant time headway policy* and *constant safety policy.* Constant time headway offers a constant, speed independent carrying capacity, while the safety factor drops below unity as velocity increases. Constant safety spacing guarantees a $K_{\mathrm{S}} \geq 1$. But for higher velocities, the carrying capacity drops below the one of the constant time headway spacing.

Now we introduce the necessary parameters and math: If vehicle $n-1$ has a maximum failure deceleration of a_F and the following vehicle n can guarantee an emergency deceleration of a_E by applying the emergency brake after time T_E then vehicle n is in a safe state if $d_n(t) > d_{\mathrm{S}}(v_n(t), v_{n-1}(t))$ with

$$d_{\mathrm{S}}(v_n, v_{n-1}) = T_E v_n + \frac{1}{2}\left(\frac{v_n^2}{a_E} - \frac{v_{n-1}^2}{a_F}\right). \tag{1}$$

Parameters a_E, a_F, T_E, vehicle length ℓ and the nominal line velocity V_{L} do ultimately determine the line capacity C in vehicles per hour, with

$$C = \frac{1}{\frac{d_{\mathrm{S}}(V_{\mathrm{L}}, V_{\mathrm{L}}) + \ell}{V_{\mathrm{L}}} + T_E} \times 3600.$$

The nominal line velocity V_{L} can be determined individually for each section of the network at the "Section Tab" of the control-panel of the corresponding track-element. In the general case if $a_F < \infty$ the plot of the line-capacity over the line-speed results in a curve with zero capacity at $V_{\mathrm{L}} = 0$, zero capacity if $V_{\mathrm{L}} \to \infty$ and a maximum capacity at

$$v_{\mathrm{opt}} = \sqrt{\frac{2\ell}{\frac{1}{a_E} - \frac{1}{a_F}}}.$$

The comfort criteria is introduced by limiting the acceleration $a_n(t)$ of each vehicle n at the comfort acceleration a_C. If we want to avoid collisions by braking only at at the comfort rate we must respect the *minimum comfort distance* $d_{\mathrm{C}}(\cdot,\cdot)$ with

$$d_{\mathrm{C}}(v_n, v_{n-1}) = T_E v_n + \frac{1}{2a_C}\left(v_n^2 - v_{n-1}^2\right) \tag{2}$$

The simulator can do both, *constant safety* and *constant time headway.* In fact, constant time headway is a special case of a constant safety, when $a_E = a_F$. The controller of the simulator does simply guarantee that the distance is at all times and for all vehicles grater than both, $d_{\mathrm{S}}(v_n, v_{n-1})$ and $d_{\mathrm{C}}(v_n, v_{n-1})$. If there is no vehicle in front, the vehicle will accelerate to the given nominal line velocity, which is a property of the section of the current track element.

Attention: at merge points you have to make sure that the legs of the merge element are at least as long as $d_{\mathrm{C}}(V_{\mathrm{L}}, V_{\mathrm{L}})$, otherwise vehicle won't have the chance to line up properly and crashes may occur.

All accelerations and the brake actuation time should be modified during the "add multiple" operation, when adding the carriers to the network. The parameters of individual vehicles can be modified through their control-panel.

The nominal line velocity can be modified for each section at the Sections-tab of the control-panel of each track-element. Section 4.4 explains how to do this.

There is also the possibility to introduce a comfort jerk j_C: simply add to the brake actuation time the term $2\frac{a_C}{j_C}$. This will add enough time-headway so that the acceleration can adapt at comfort jerk rate in the worst case scenario.

6 Bugs

At present, the software is not memory and speed optimized. **Tip if you are short of memory:** After you have completed the network design, delete your background maps with Edit/ Map/ Delete and safe this version under a different name. Then run the simulation. The results will be the same, but your computer does not have to deal with the memory of the maps. You can also much quicker zoom in and out without maps.

In any case, for larger networks it is recommended to run the simulation in command-line mode or as a script, without graphical support. This will reduce the used memory and make the simulations roughly 5 times faster, see Sec 4.3 and Sec 4.2.

Some "minor" problems have been discovered and will be improved in future versions:

- There may be problems with reloading already simulation files.Sometimes it cannot be seen immediately, but during the simulation when various WARNING messages do occur on the terminal.

  Please keep always a copy of the unactivated track as backup. This will always work. From there it is quick to reconstruct the simulation by adding different numbers of vehicles and users.

- Vehicles crash into each other at a certain merge points. The reason why this can happen is that there is no design rule checker. Most likely one of the branches of your merge element is too short in order to allow vehicles to adapt their speed.

  The only solution is to extend the input lines or to lower the line speed on the input sections (and on previous section in order to guarantee that the vehicles enter the merge at the desired speed.). In general one would set the same line-velocity for both input sections of a merge element (first two sections on the Section page of the control-panel).

- The deviation rate (number of deviation divided by the total number of switch operation$\times$100) on the track management control-panel can show 101% instead of 100%.

- It has been observed that vehicles stop inside stations and refuse to load passengers. This can occurs if stations are over-congested. Try to avoid over-congested networks. Add station or bypass or reduce the number of vehicles.

- Some error messages may occur after running and re-editing a simulation. But this should be avoided in any case: edit only unactivated tracks, save the network, activate it and then run the simulation.
- In some cases, a discontinuity can occur when connecting track elements. Usually this happens when the connecting algorithms cannot straighten the angle between both elements because it is too big.

  Workaround: just move the track elements apart with Edit/ Module/ Move and re-connect them. When still problems, try to change relative angle or position element or position of nodes. Or use flexible elements instead of curve elements or lines. If this still does not help it can be that you ask for a too small curve radius, which would also be a problem of the real system: Consider a different network-layout.
- Some track elements show "strange" stretch properties when placed with certain angles.

  Workaround: move (Edit/ Module/ Move) and turn the track element slightly if possible. When still problems, consider using alternative track elements.

7 Release notes for version its-0.51

Version its-0.51 :

- Empty vehicle management.
- Improved vehicle and boarding behavior for high capacity stops.
- Concept of user-profiles, see Sec. 4.1.
- Simple, real-time 3D OpenGL viewer (Requires PyOpenGL module installed), see Sec. 4.1.
- More export tools (Station statistics, trip-time tables etc.)
- Various bug fixes.

Version 0.3:

- Vehicle follower control algorithm has been enhanced by brake actuation time (which also allows to add extra inter-vehicle space for jerk adaptation).
- Improved boarding behavior for high capacity stops: if there is high demand, the stop is switching to a so called "synchronous mode", where the stop is divided in a unloading and loading area.
- All user types have now a boarding and an exit time as parameters.
- The new "generic user" can be place automatically according to a predefined origin-to-destination demand matrix.
- The new "generic user" has a trip start time. During the creation phase boarding and an exit time as well as start time can be randomized by giving time intervals as parameters.

- Simulator can run in command line mode, without graphical support. There are various command line options to control the most important simulation parameters and output/report file names.
- Scripting.
- Postscript snapshot tool: This tool can be “switched on” to make regular snapshots of the canvas. The snapshots will be saved as files in Postscript(R) format.

8 Features for future releases

Not all of these features will come with the next release...

- Improved task and new resource classes for management objects.
- Predictive Stochastic Network Control (PSNC) for global traffic. optimization.
- Running the simulator in a shell. This gives the same access to the simulator as within a scrip, but it is interactive.
- More sophisticated export tools (suggestions?).
- 3D OpenGL viewer and/or interface to renderer (most likely blender2.x).
- Improved vehicle follower control algorithm with explicit jerk simulation. More ride comfort, better implementability.
- Extension to MAIT simulator: global net-management, multiple track and carrier managements, multiple cabin managements.
- Import data of geographical data instead of bit maps (formats not yet decided, suggestions??)
- Display of graphs with speed profiles etc.
- Printer support.
- Balloon help support.
- Improved graphical network editor with mark, copy/ paste, undo etc.