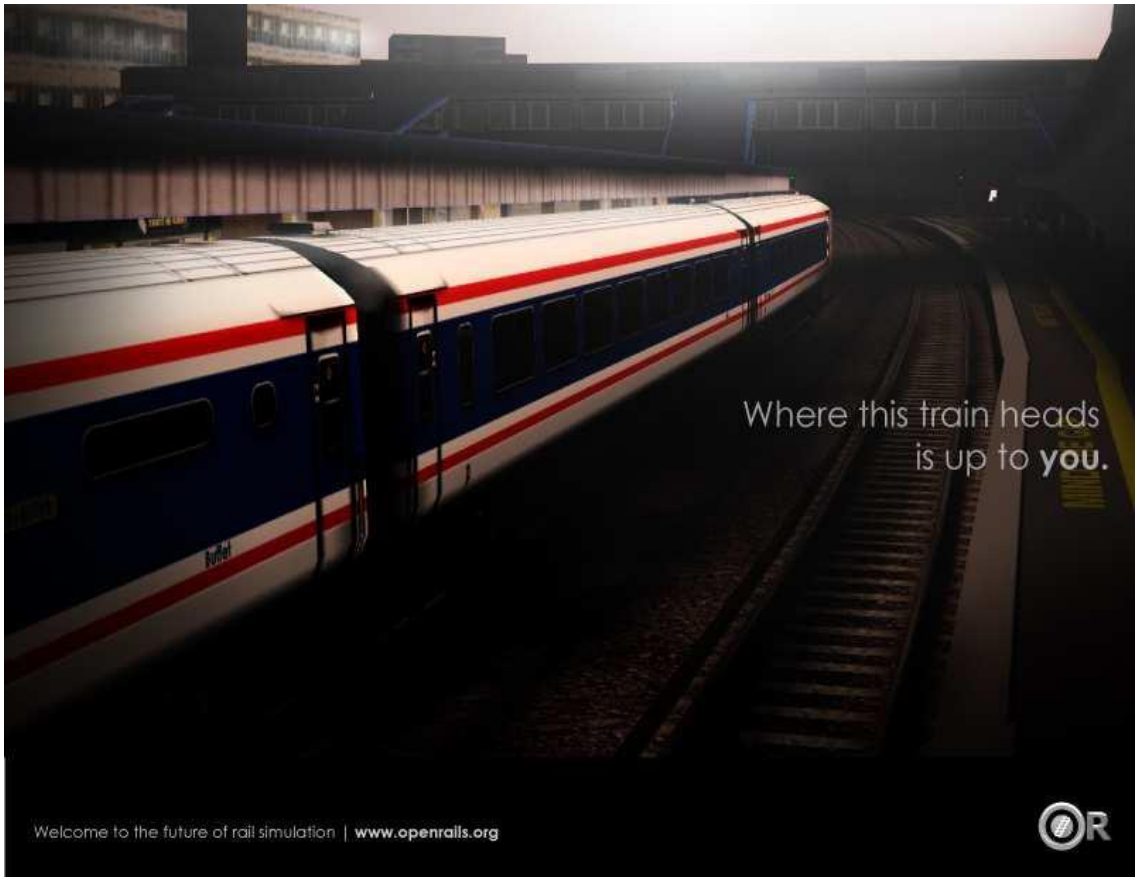


Draft Operations Manual

Version 1.0

1. Legal

1.1. Warranty

NO WARRANTIES. openrails.org disclaims any warranty, at all, for its Software. The Open Rails software and any related tools, or documentation is provided “as is” without warranty of any kind, either express or implied, including suitability for use. You, as the user of this software, acknowledge the entire risk from its use. See the license for more details.

1.2. Trademark Acknowledgment

Open Rails, Open Rails Transport Simulator, ORTS, Open Rails trademark, openrails.org, Open Rails symbol and associated graphical representations of Open Rails are the property of openrails.org. All other third party brands, products, service names, trademarks, or registered service marks are the property of and used to identify the products or services of their respective owners.

1.3. Copyright Acknowledgment and License

 This document is part of Open Rails.

2. New in This Release

Here are the features which have been added or substantially changed since v0.9 was released.

- Extremely high compatibility with MSTS content
- Train operation accordingly to timetables entered in .xls files
- Support for languages other than English
- Support of 3D cabs
- Train physics far more realistic than in MSTS

Some experimental features have been added which you can turn on; some of them may affect performance:

- Enhanced compatibility with MSTS activities
- Compatibility with MSTS environment files
- Extended AI train shunting
- Adhesion linked to weather
- Support for DDS textures
- Extended viewing distance

Contents

3. Introduction

3.1. What is Open Rails?

Open Rails software (OR) is a community developed and maintained project from openrails.org. Its objective is to create a new transport simulator platform that is first, compatible with routes, activities, consists, locomotives, and rolling stock created for Microsoft Train Simulator (MSTS); and second, a platform for future content creation freed of the constraints of MSTS.

Our goal is to enhance the railroad simulation hobby through a community-designed and supported platform built to serve as a lasting foundation for an accurate and immersive simulation experience. By making the source code of the platform freely available under the GPL license, we ensure that OR software will continually evolve to meet the technical, operational, graphical, and content building needs of the community. Open architecture ensures that our considerable investment in building accurate representations of routes and rolling stock will not become obsolete. Access to the source code eliminates the frustration of undocumented behavior and simplifies understanding the internal operation of the simulator without the time-consuming trial and error-prone experimentation typically needed today.

Open Rails software is just what the name implies – a railroad simulation platform that's open for inspection, open for continuous improvement, open to third parties and commercial enterprises, open to the community and, best of all, an open door to the future.

3.2. About Open Rails

To take advantage of almost a decade of content developed by the train simulation community, Open Rails software is an independent game platform that has backward compatibility with MSTS content. By leveraging the community's knowledge base on how to develop content for MSTS, Open Rails software provides a rich environment for both community and payware contributors.

The primary objective of the Open Rails project is to create a railroad simulator that will provide 'true to life' operational experience. The Open Rails software is aimed at the serious train simulation hobbyist; someone who cares about loco physics, train handling, signals, AI behavior, dispatching, and most of all running trains in a realistic, prototypical manner. While the project team will strive to deliver an unparalleled graphical experience, 'eye candy' is not the primary objective of Open Rails software.

By developing a completely new railroad simulator, Open Rails software offers the potential to better utilize current and next generation computer resources, like graphics processing units (GPUs), multi-core CPUs, advanced APIs such as PhysX, and widescreen monitors, among many others. The software is published so the user community can understand how the software functions to facilitate feedback and to improve the capabilities of Open Rails software.

Open Rails is published under the GPL license which is "copyleft"[1] to ensure that the source code always remains publicly available.

3.3. Does Open Rails need MSTS to run?

This is not a correctly set question. Open Rails is able to run a vast majority of MSTS content (routes, trains, activities). Open Rails does not need MSTS executable files (e.g. .exe or .dll files), neither it needs .ini files.

However, if the MSTS content uses content files originally delivered with MSTS, like tracks or general sounds (this applies in particular to routes), obviously to run such content OR needs such files.

If instead (and there are examples of this) the MSTS content does not use such original content files, again obviously OR does not need original MSTS files.

In both cases, MSTS content files (original and not) must be organized in a MSTS-compatible file structure. Within this manual such file structure will be called “MSTS installation” for clarity, even if this wording is not completely correct.

3.4. Community

At the present time, Open Rails software is offered without technical support. Therefore, users are encouraged to use their favorite train simulation forums to get support from the community.

- Train-Sim.Com http://forums.flightsim.com/vbts/
- UK Train Sim http://forums.uktrainsim.com/index.php
- Elvas Tower http://www.elvastower.com/forums/index.php?/index

For users interested in multiplayer, a forum is set up for you to seek and announce hosting sessions: http://www.tsimserver.com.

The Open Rails team is NOT planning on hosting a forum on the Open Rails website. We believe that the best solution is for the current train simulation forum sites to remain the destination for users who want to discuss topics relating to Open Rails software. The Open Rails team monitors and actively participates at these forums.

3.5. Highlights of current version

1 http://www.gnu.org/copyleft/

3.5.1. Focus on compatibility

With this release the announced goal has been reached to make as much of the existing MSTS content as possible run on the Open Rails. The development team's initial focus has been to provide a fairly complete visual replacement for MSTS that effectively builds on that content, achieving all the compatibility that is worthwhile, at the same time delivering a system which is faster and more robust than MSTS.

3.5.2. Focus on Operations

Release 1.0 clears the way to improving on MSTS in many ways which can be summed up as moving from Foundation to Realism and eventually to Independence, and already includes features that are beyond MSTS. Non-player trains can already have a first release movement orders (i.e. pickups, drop offs) based on files in MSTS format. Deadlocks between player and non-player trains, that are frequent in MSTS, have been practically eliminated.

3.5.3. Focus on Realistic Content

(To be reworked ???)

The physics underlying adhesion, traction, engine components and their performance will be based on a world-class simulation model that takes into account all of the major components of diesel, electric and steam engines. This includes elements never before simulated in a consumer train simulation game like DC vs AC traction motors, number of axles in the truck design, wheel creep and hop, electric shunting, tilting, momentum, thermodynamics of the steam cycle including compensation for fuel types, plus modeling of different electric traction systems. Open Rails will approach the level of physics realism only available in professional simulators.

Some of the major goals for our new physics are:

- Freedom from the constraints and limitations of the MSTS physics
- Ability to dynamically simulate component performance based on player input (traction motor degrading due to overheating); firing of steam locomotive including all major elements such as cylinders, blowers, grate, fuel types; component failure (turbo, electrical system, etc); or dynamic rail conditions on adhesion/traction physics
- Steam cycle implementation delivering an accurate simulation of the smallest to the largest compound engines as a single engine including different fuel types, firing strategies and steam usage (heating, injectors, etc.)
- Interaction of different diesel electric components to the overall physics covering first, second and third generation designs
- An electric traction model capable of simulating different electrical traction systems: AC, DC and different generations of electric traction engines
- New sound triggers to give audio complement to the new physics model

Existing models that do not have the upgraded Open Rails capabilities continue, of course, to perform well.

In the package of this version also ancillary programs (“tools”) are delivered, like:

- Track Viewer: a complete track viewer and path editor
- Activity Editor: a draft new activity editor to move beyond MSTS

4.MSTS File Format Compatibility

Open Rails software supports the MSTS file formats detailed below. The software uses a file parser to read the MSTS file information for use by the Open Rails software. Testing of the parser software indicates that it will locate many errors or malformation in these files that are not highlighted by the MSTS train sim software or by other utilities. In most cases, the Open Rails software will ignore the error in the file and run properly. Open Rails software logs these errors in a log file on the user's desktop. This log file may be used to correct problems identified by the Open Rails software.

4.1. Trainset

The software currently supports Shape (.s); Shape Definition (.sd); Sound (.sms); and texture Ace (.ace) files; including displaying the correct LOD, alpha and transparency attributes. Moreover it supports following file types: Engine (.eng); Wagon (.wag); it substitutes MSTS-style physics to enable the user to operate trains.

4.2. Consists

Open Rails software reads and displays Consist files (.con) used for Player Train, AI Train, and Loose Consists in activities.

4.3. Services

Open Rails software supports MSTS Service files (.srv) for the creation of both Player and AI services.

4.4. Paths

Open Rails software supports MSTS Path files (.pat) for determining the path of both Player and AI Trains.

4.5. Routes

Open Rails software supports the following MSTS Route files with the limitations noted.

- Route Database file (.rdb) - CarSpawner is supported.
- Reference File (.ref) – Open Rails does not provide a Route Editor in the current release.
- Track Database file (.tdb) – supported
- Route File (.trk) – Level Crossings and overhead wires are supported.
- Sigcfg (.dat) file – Signal & scripting capabilities are supported.
- Sigscr (.dat) file – Signal & scripting capabilities are supported.
- Speedpost (.dat) file – Supported
- Spotter (.dat) file – Supported

- Ssource (.dat) file – Supported
- Telepole (.dat) file – Supported
- Tsection (.dat) file – Supported
- Ttype (.dat) file – Supported
- Hazards (.haz) file - Supported

4.6. Environment

Open Rails software does not support advanced water dynamic effects at this time, while it supports first-level, player-driven dynamic weather effects.

Open Rails provides two types of environment representation, that can be selected by the player at game start: a MSTS compatible one and a native one.
In the native one Open Rails software uses its own sky, cloud, sun, moon and precipitation effects developed exclusively for it. In activity mode the starting parameters for time of day and weather are read from the activity file to determine the starting display in Open Rails software.

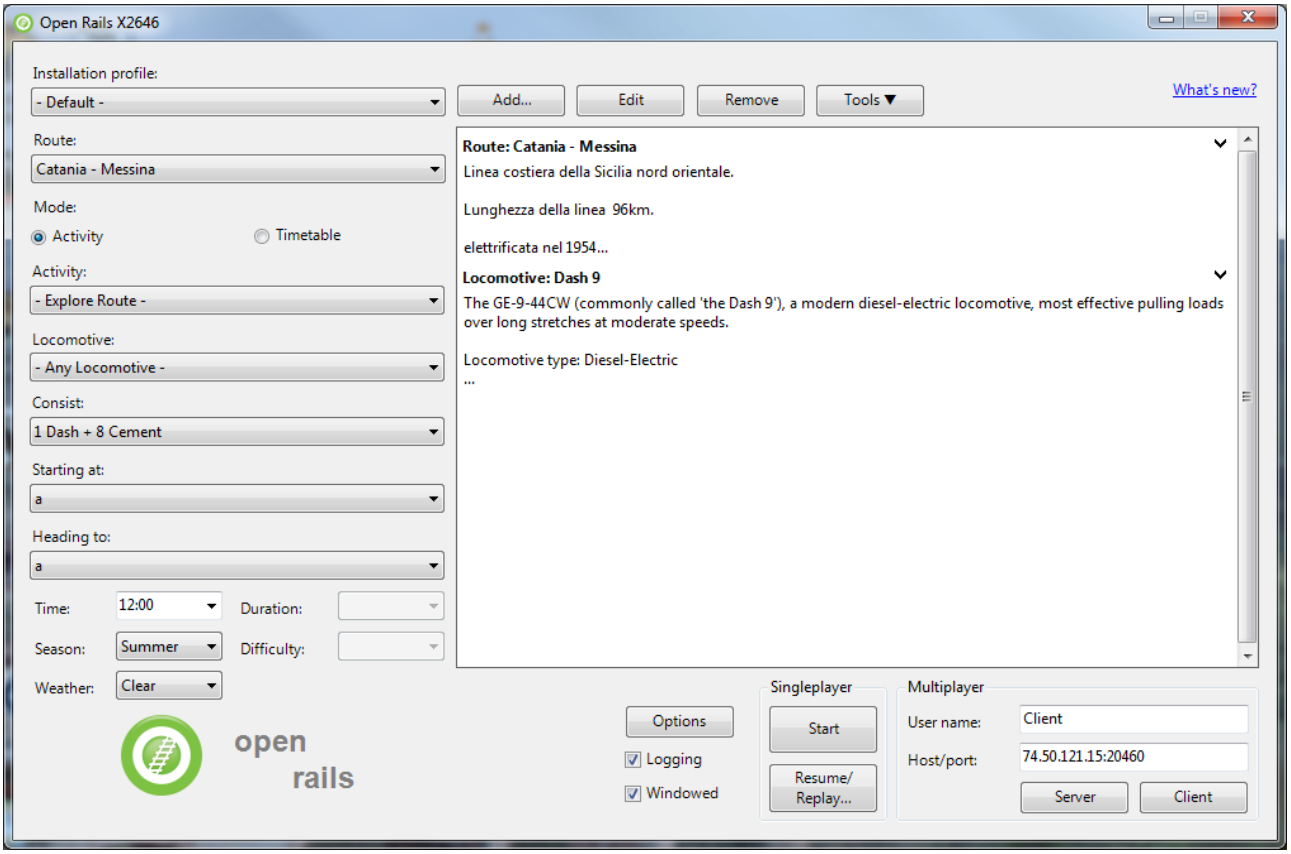
4.7. Activities

Open Rails software runs without problems a great percentage of the passenger and freight activities created using the MSTS activity editor, provided this option is checked.

5. Getting started

After having successfully installed Open Rails (see Installation Manual), to run the game you have to double-click on the Open Rails icon on the desktop or on OpenRails.exe file.

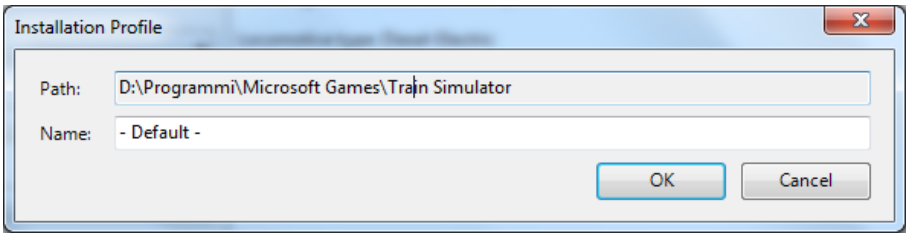
Following start window will appear:



5.1. Installation profiles

In the simplest case, where you have only a basic MSTS installation (see paragraph "Does Open Rails need MSTS to run?" for a precise definition of MSTS installation, OR should already correctly point at that. To check this, you should first of all see under "Installation Profile" the string " - Default - ".

Then, by clicking on "Edit" , you should get a window like the following one:



You can easily add further MSTS installations and switch between them (e.g. if you have the so-called “mini-routes” installed.)

To add a new installation (e.g. a “mini-route”) select “Add...” and select the root of the installation you want to add. You will also be asked to insert a name for the installation, to select it when you want to run the game.

With the “Remove” button you can remove from the OR list (not from the computer!) an installation.

5.2. Updating OR

When a new release of OR is available and the computer is online, a link “Update to release xnn” appears above the link “What's new” on the upper right part of the main menu window.

By clicking on the update link OR downloads the new release.

This way you are always up to date with your OR.

Various types of updates, named Update Channels, are available, depending from your OR-user profile. They are described here.

5.3. Preliminary selections

First of all, under “Route:” the route you want to play on has to be selected.

By checking the “Logging” checkbox, Open Rails will generate a log file named OpenRailsLog.txt that resides on your desktop. Such log file is very useful to document and investigate malfunctions.

At every restart of the game (that is after clicking “Start” or “Server” or “Client”) the log file is cleared and a new one is generated.

By checking the “Windowed” checkbox, Open Rails will run in a window instead of full screen.

If you want or need to fine-tune OR clicking on the “Options” button, you should read chapter “Open Rails options” about the rich OR options. It is recommended that you read such chapter.

5.4. Gaming modes

One of the plus points of Open Rails is the variety of gaming modes you can select.

5.4.1. Traditional “Activity” and “Explore mode” modes

As a default you will find the radio button “Activity” selected in the start window.

This will allow you to run an activity or run in explore mode.

If you use “Explore mode” (first entry selected under “Activity:”), you will have also to select the consist, the path, the starting time, the season and the weather with the relevant buttons.

To select the consist you have two possibilities: either you click under “Consist:”, and the whole list of available list of consists will appear, or you first click under “Locomotive:”, where you select the desired locomotive, and then click under “Consist:”, where only the consists led by that locomotive will appear.

If you instead select a specific activity, you won't have to perform further selections.

If you have selected the related Experimental Option, you can switch on and off at runtime Autopilot mode, where you can watch OR driving your train, like you were a trainspotter of a visitor in the cab.

5.4.2. Timetable mode

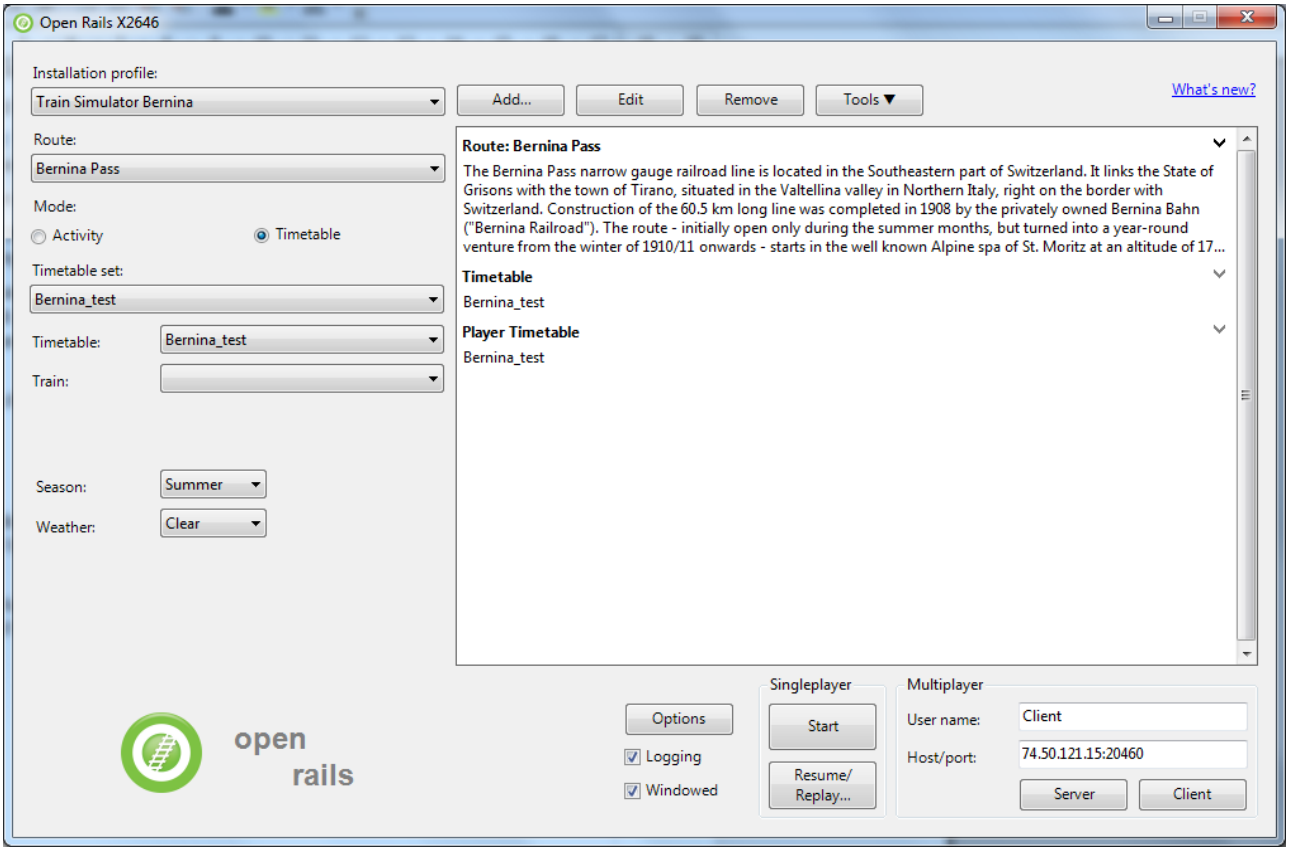
Timetable mode is based on a timetable that is available on a spreadsheet formatted in a predefined way, and defining trains and their timetable, their path, their consist, some operations to be done at end of train run, and some train synchronization rules.

Timetable mode allows to significantly reduce development time with respect to activities in case no specific shunting or train operation is foreseen. The complete description on the timetable mode can be found here.

The spreadsheet has a .csv format, but must be saved changed its extension into timetable_or within an “Openrails” subdirectory of the route's ACTIVITIES directory.

To the game player, one of the most interesting features of timetable mode is that any one of the trains defined in the timetable can be selected as player train.

If you select the radio button “Timetable”, the menu window will change as follows:



As of now buttons under “Timetable set:” and near “Timetable:” contain the same information. After having selected under “Timetable set:” the timetable you desire, the contents of the button near “Timetable:” will be automatically selected.

Now you can select at the right of “Train:” the train you will desire to run, among all trains of the timetable. Season and weather can be selected too.

5.4.3. Run!

Now, if you want to play, click on “Start”, and OR will start loading the data needed for your game. At end of the loading, you will be within the cab of your locomotive! You can read further in chapter “Driving a train”.

5.4.4. Multiplayer mode

Open Rails features also this exciting game mode: more players, each one on a different computer in a local network or through the Internet, can play together, each driving a train and seeing the trains of the other players, even interacting with them by exchanging wagons, under the supervision of a player that acts as dispatcher.

The multiplayer mode is described in detail here.

5.4.5. Replay

It is not a real gaming mode, but it is nevertheless another way to experience OR. After having run a game you can save it, and replay it: OR will save all commands you gave, and will automatically execute such commands during replay: it's like you are seeing a video on how you played the game.

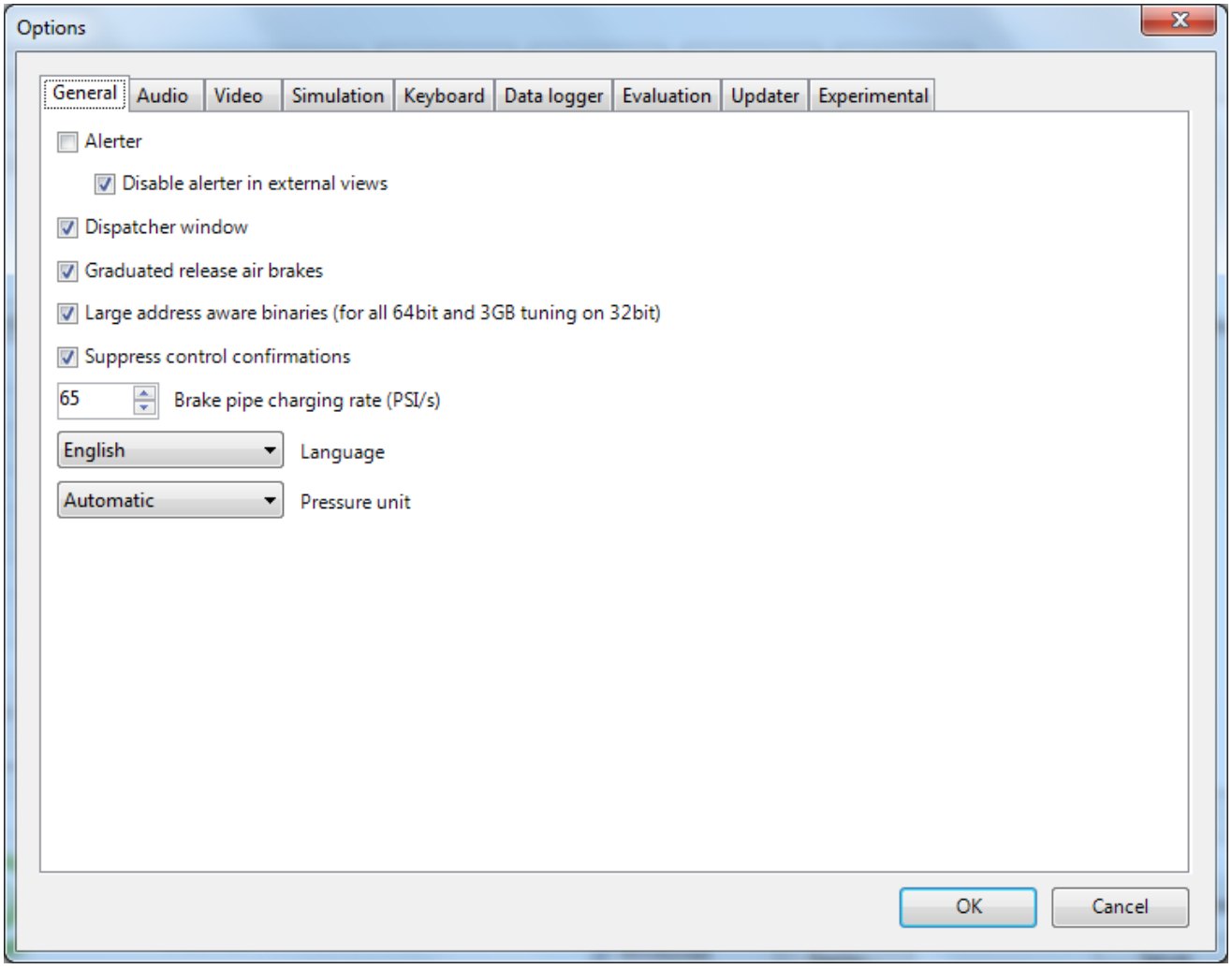
Replay is described later together with the save and resume functions.

6. Open Rails Options

The *Menu* > *Options* panels contain the settings which remain in force for the duration of your simulation. Most of the options are self-explanatory; you can adjust them according to your preference and system configuration. For example, you can turn off dynamic shadowing if your system has low FPS. The options configuration that you have selected is saved. When you will restart OR, it will use the last options configuration you selected.

There are 9 options panels, that are described here below.

6.1. General

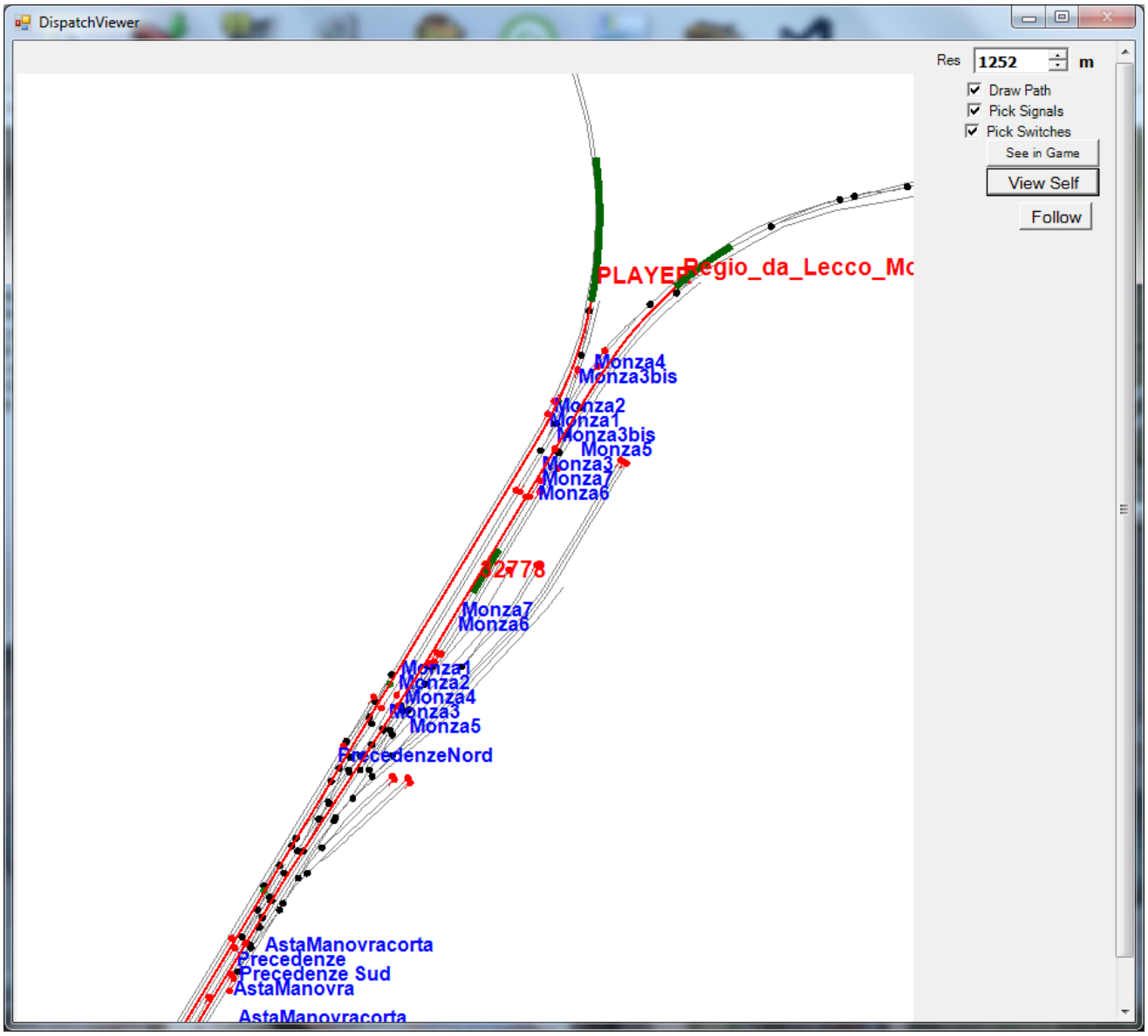


6.1.1. Alerter

As in real life, when this option is selected, the player driving the train is required to perform specific actions to demonstrate that he is “alive”. As sometimes the player uses views different from the cabview to follow his train, and therefore does not see the alerter warning, with the “Disable alerter in external views” he does not need to care about alerter in this situation.

6.1.2. Dispatcher window

It is suggested to always select this option. When this option is selected, at runtime pressing Ctrl-9 a new window like the following one appears.



This window coexists with the main OR window. Through the window you can monitor train circulation and also influence it, by setting signals and switches. A complete description of the dispatcher window can be found here

6.1.3. Graduated release air brakes

Selecting *Graduated Release Air Brakes* in *Menu > Options* allows a partial release of the brakes. Generally speaking checked is equivalent to passenger standard and unchecked is equivalent to freight standard. A complete description of this option can be found here .

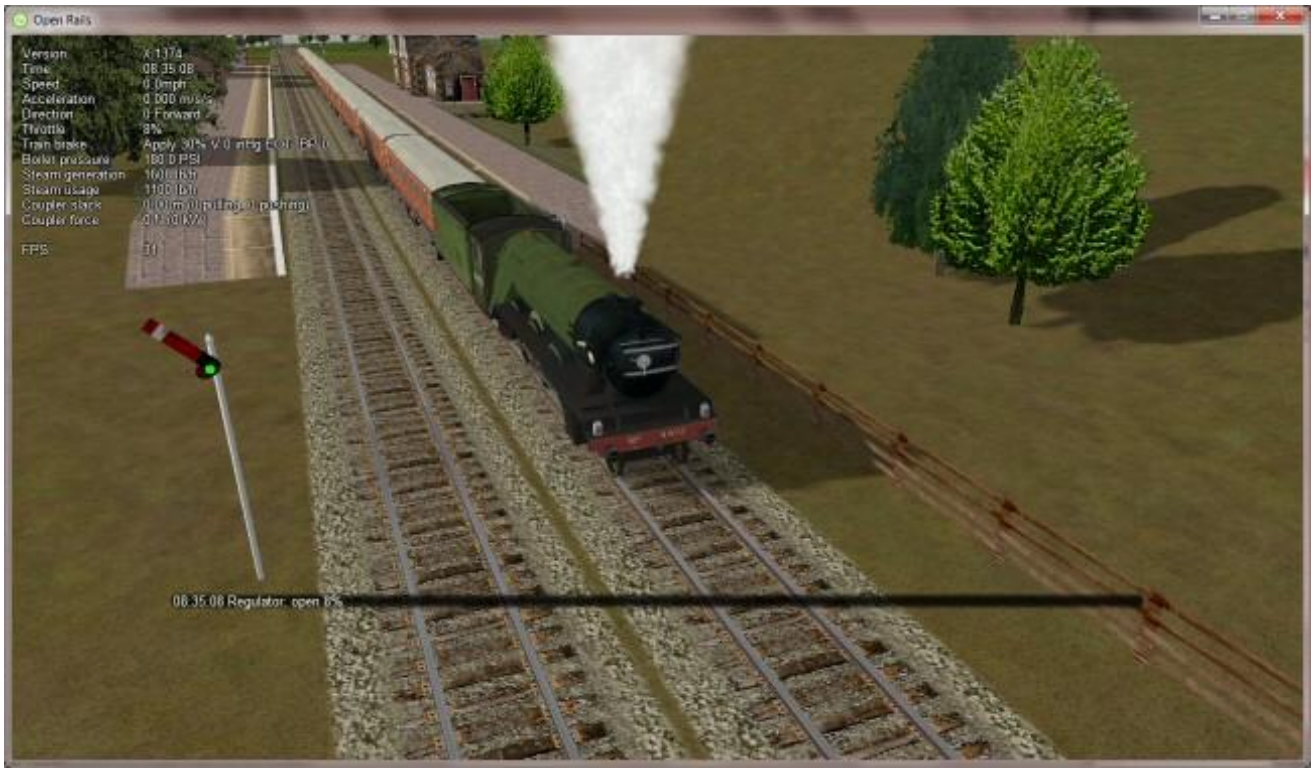
6.1.4. Large address aware binaries

It is suggested to leave this option checked. When it is unchecked, Open Rails may use as a maximum 2 GB of RAM. When it is unchecked, the maximum is 4 GB for 64-bit Windows systems, and 2 or 3 GB for 32-bit Windows systems. To extend from 2 to 3 GB the maximum RAM used by OR in 32-bit Windows systems you may read here http://knowledge.autodesk.com/support/autocad/troubleshooting/caas/sfdcarticles/sfdcarticles/How-to-enable-a-3GB-switch-on-Windows-Vista-Windows-7-or-Windows-XP-s.html .

Pls. Take note that the extension from 2 to 3 GB in 32-bit systems can slow down computer operation when not using OR.

6.1.5. Suppress control confirmations

Following MSTS practice, whenever you make adjustments to the train controls (e.g. open the throttle) OR briefly shows a message near the bottom of the screen.



This is helpful for operations that don't have visible feedback and also allows you to control the train without being in the cab.

Check this option if you prefer to monitor your cab instruments and don't want to see these messages.

OR uses the same message scheme for system messages such as "Game saved" or "Replay ended" but you can't suppress these system messages.

6.1.6. Brake pipe charging rate

Increasing the *Brake Pipe Charging Rate (PSI/Second)* value controls the charging rate.

Increasing the value will reduce the time required to recharge the train (that is releasing the brakes after a brake application), while decreasing the value will slow the charging rate. See also the paragraphs on the OR implementation of the braking system.

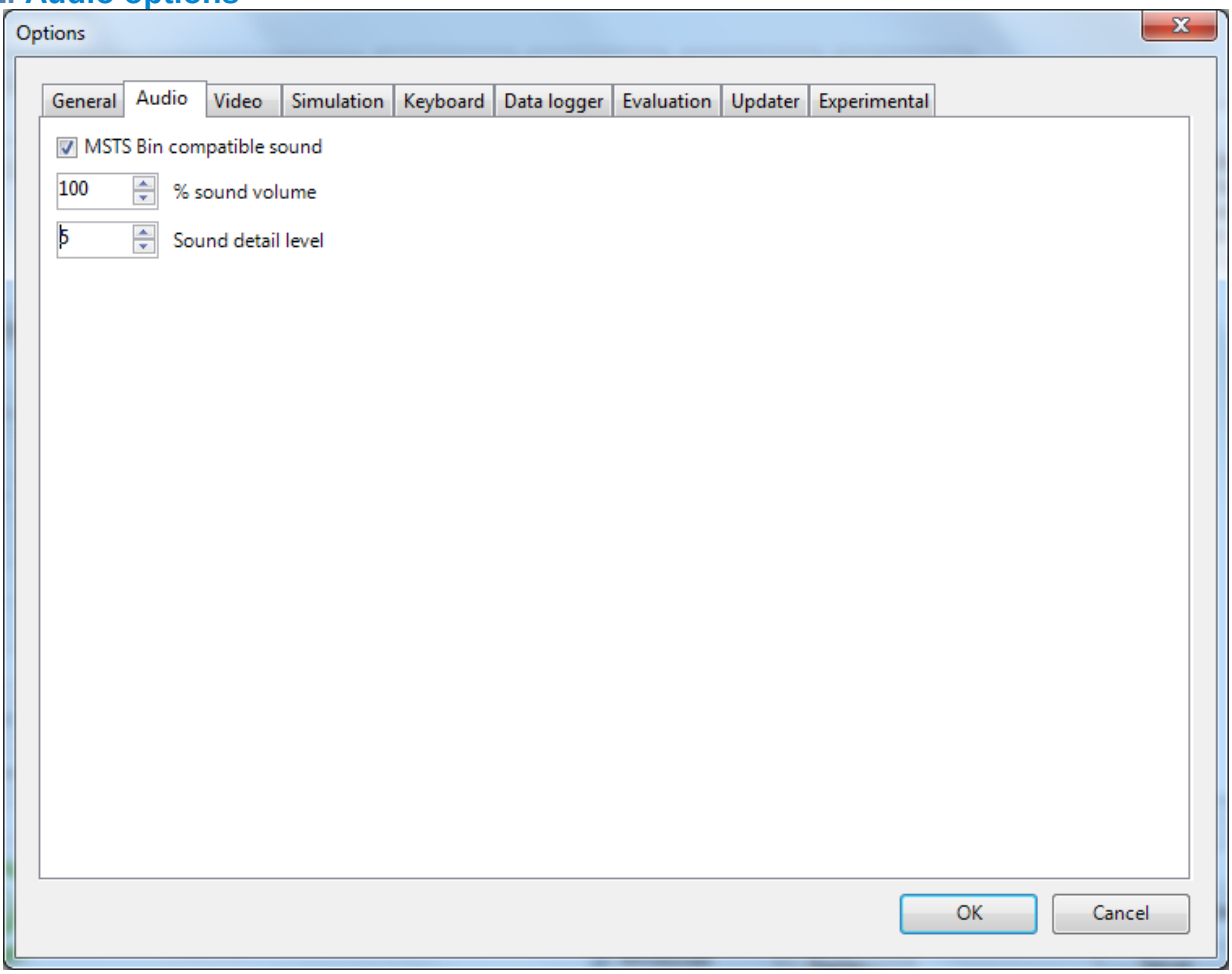
6.1.7. Language

OR is an internationalized package. It supports many languages, and other ones can be added by following instructions contained in the “Localization manual”.
When “System” is selected, OR automatically selects the language of the hosting Windows, if such language is available.

6.1.8. Pressure unit

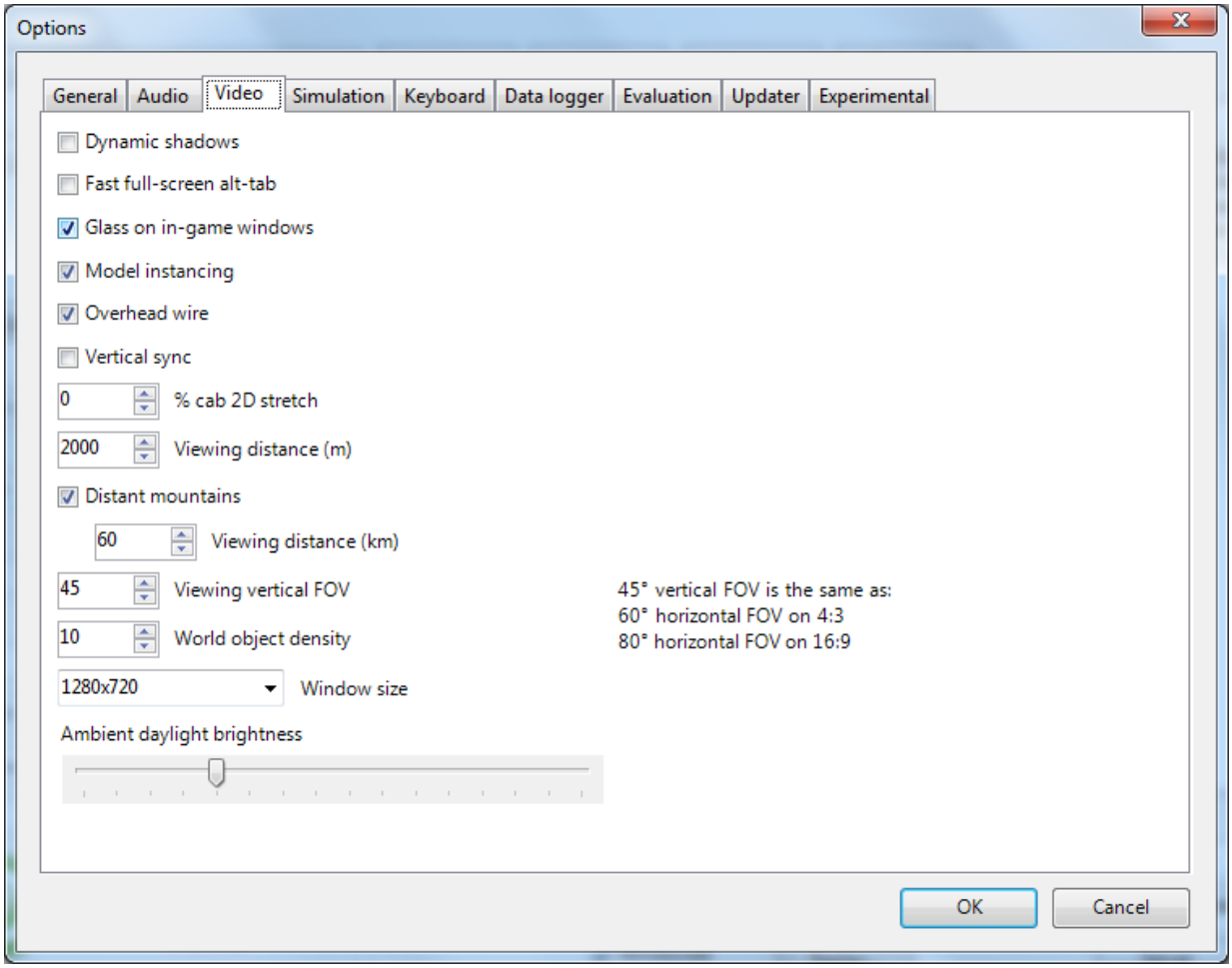
The player can select the unit of measure of brake pressure in the HUD display (see here for HUD).
When set to “automatic” the unit of measure is the same as used in the cabview of the loco.

6.2. Audio options



Except for very slow computers, it is suggested to leave the “MSTS Bin compatible sound” option checked and to set to 5 the Sound detail level.
The “% sound volume” option allows to regulate the volume of the OR sound.

6.3. Video options



6.3.1. Dynamic shadows

With this option it is possible to enable or disable display of dynamic shadows. Disabling can be of interest if low frame rates are experienced.

6.3.2. Fast full-screen alt-tab

When the option is selected, by pressing Alt-Tab at runtime with OR at full screen, OR disappears from screen (but remains alive).

6.3.3. Glass on in-game windows

When the option is checked, the in-game windows are displayed in a semitransparent mode.

6.3.4. Model instancing

When the option is checked, in case that more instances of the same object have to be drawn, only a draw call is sent to the GPU. This means lower CPU load. It is suggested to always check this option.

6.3.5. Overhead wire

This option allows to enable and disable display of the overhead wire.

6.3.6. Vertical sync

When this option is selected, OR update rate can't be higher than the monitor vertical sync frequency (often 60 Hz). This spares CPU energy consumption in fast PCs.

6.3.7. % Cab 2D Stretch

OR manages both cab interiors using 2D images in a MSTS-compatible way, but supports also 3D models. Most 2D cab images follow MSTS practice, being 1024 x 768 pixels to suit monitors with a 4:3 aspect ratio.

So, the problem arises on how to do to display these 4:3 cabs on a 16:9 or 16:10 monitor.

One possibility is to stretch these images horizontally to match other aspect ratios, as shown in the image below.

To respect proportions however OR by default has no stretching and shows the full width of the cab interior thus losing a portion from the top and bottom of the image. You can use the *Up* and *Down Arrow* keys to pan and reveal these missing portions.

Therefore the setting *% Cab 2D Stretch* has a default of 0 providing no stretching and a maximum of 100 which stretches the picture so as to cover the complete display. Values in between provide a blend of panning and stretching.

6.3.8. Viewing distance

This option defines the maximum distance where terrain is displayed. At higher distances Distant Mountains will be displayed (see below). This parameter increases CPU and GPU load; moreover some routes are optimized with the standard MSTS maximum viewing distance (2000m).

6.3.9. Distant Mountains

Distant mountains are supported in a way compatible to MSTS. Distant mountains are present in the route, if the latter has a folder called LO_TILE. You can turn the feature on by checking the “Show Distant Mountains” checkbox. In addition to MSTS, you can select the viewing distance of the distant mountains.

6.3.10. Viewing vertical FOV

This value defines the vertical angle of world that is shown. Higher values roughly correspond to a zoom out effect. The default is 45 degrees.

6.3.11. World object density

This value can be set from 0 to 10; when 10 is selected, all objects as defined in the route files are displayed. Lower values don't display some categories of objects.

6.3.12. Window size

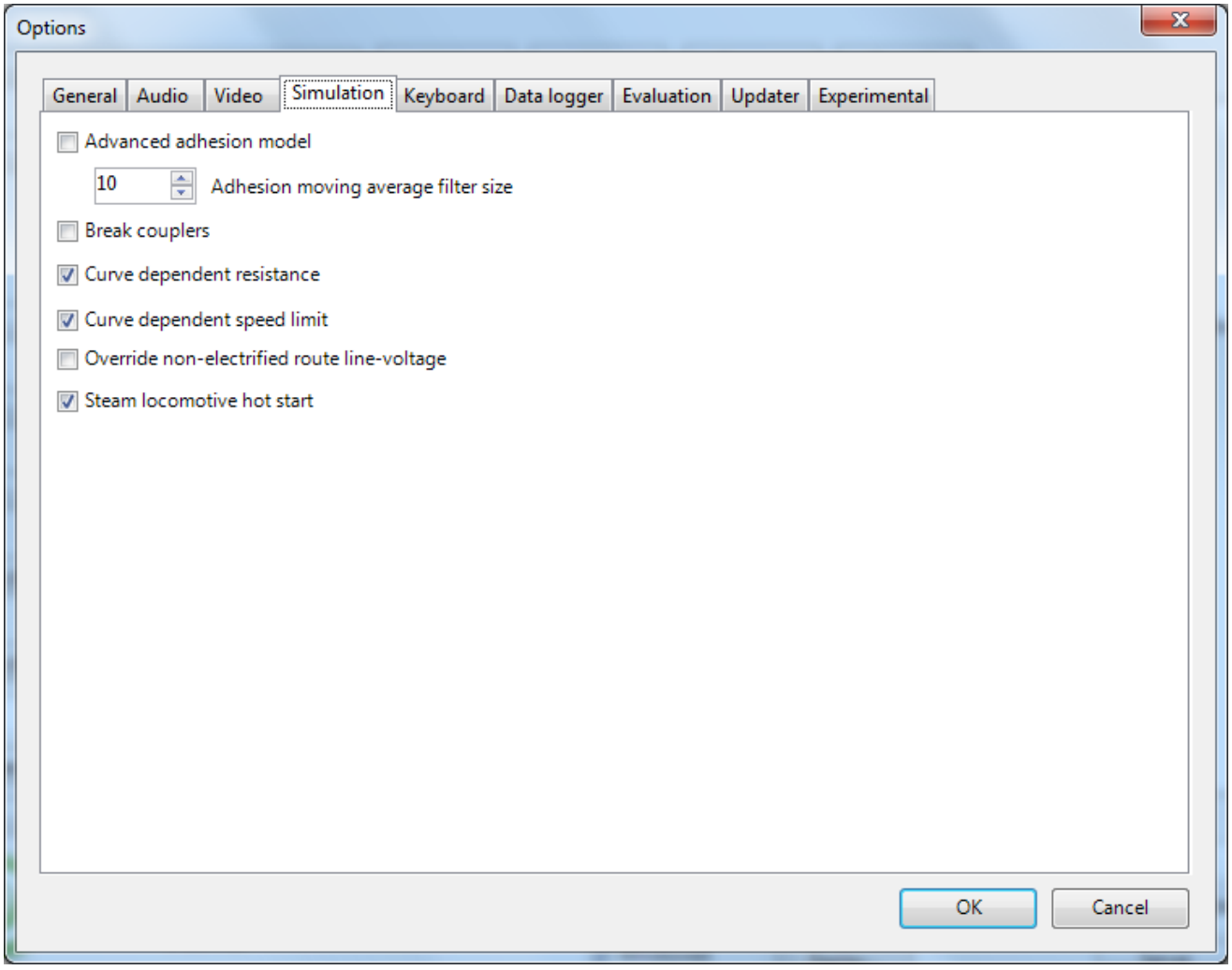
This couple of values defines the size of the OR window. There are some preconfigured couples of values, however you can also manually enter a different size to be used.

6.3.13. Ambient daylight brightness

With this slider you can set the daylight brightness.

6.4. Simulation options

The biggest part of these options defines train physic behaviors.



6.4.1. Advanced adhesion

OR supports two adhesion models: the basic one is similar to the one used by MSTS, while the advanced one is based on a model more similar to reality.
To know more you can read paragraph “ Adhesion model” later in this manual.

6.4.2. Adhesion moving average filter size

The computations related to adhesion are passed through a moving average filter. Higher values cause smoother operation, but also less responsiveness. 10 is the default filter size.

6.4.3. Break couplers

When this option is selected, in case the force on a coupler is higher than the threshold set in the .eng file, the coupler breaks and the train is divided in two parts.

6.4.4. Curve dependent resistance

When this option is selected, resistance to train motion is influenced by the radius of the curve it is running on. This option is described in detail in this paragraph (theory) and in this one (OR application).

6.4.5. Curve dependent speed limit

When this option is selected, OR computes if the train is running too fast in curves, and in such case a warning message is logged and displayed on the monitor. This option is described in detail in this paragraph (theory) and in this one (OR application).

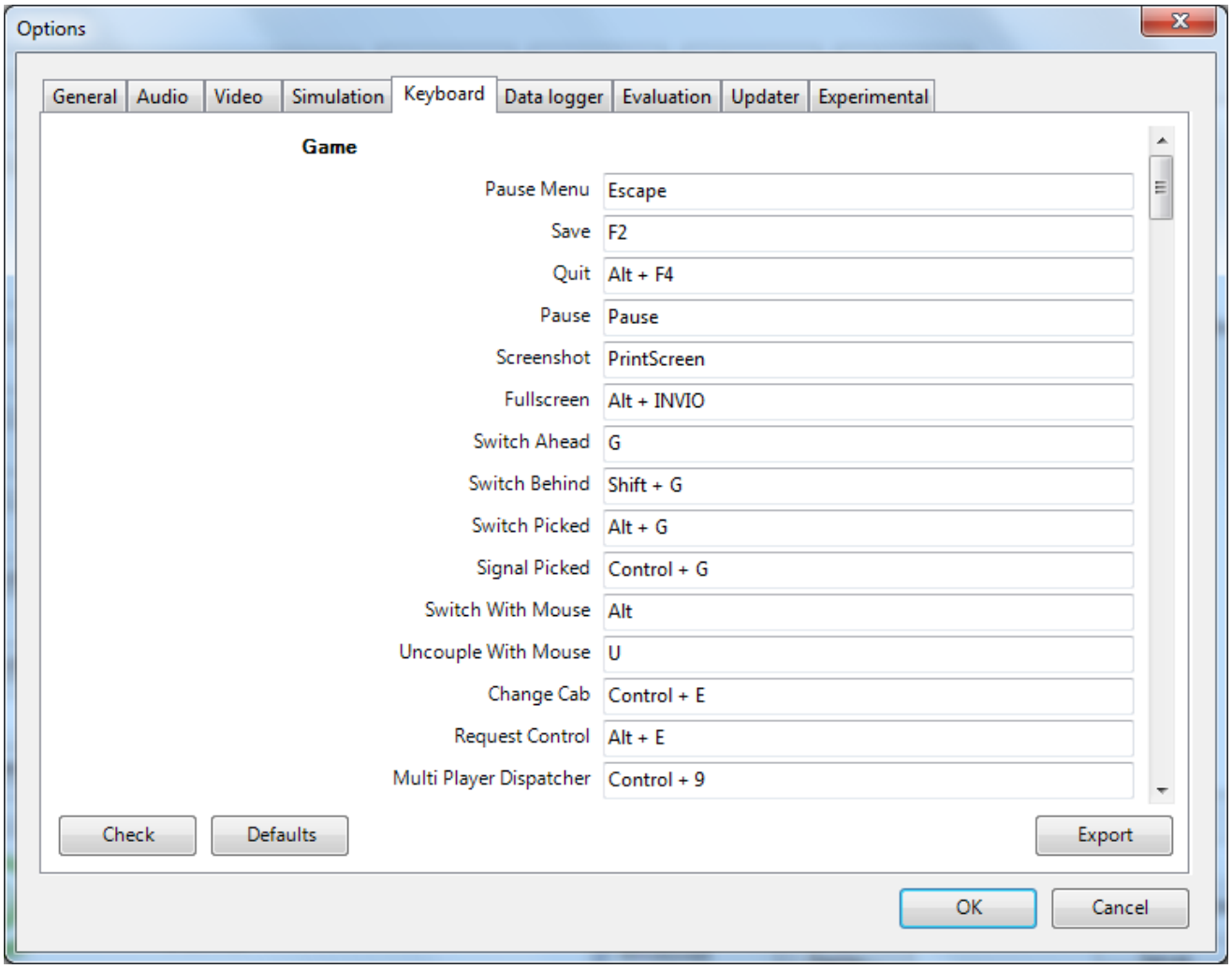
6.4.6. Override non-electrified route line-voltage

This option allows to run (in an unprototypical way) electric locos on non-electrified routes.

6.4.7. Steam locomotive hot start

This option allows to start the game with the water temperature already at a value that allows to run the locomotive. If not set, you will have to wait until the water temperature reaches a sufficient value.

6.5. Keyboard options



In this panel you find the keyboard keys that are associated to all OR commands.

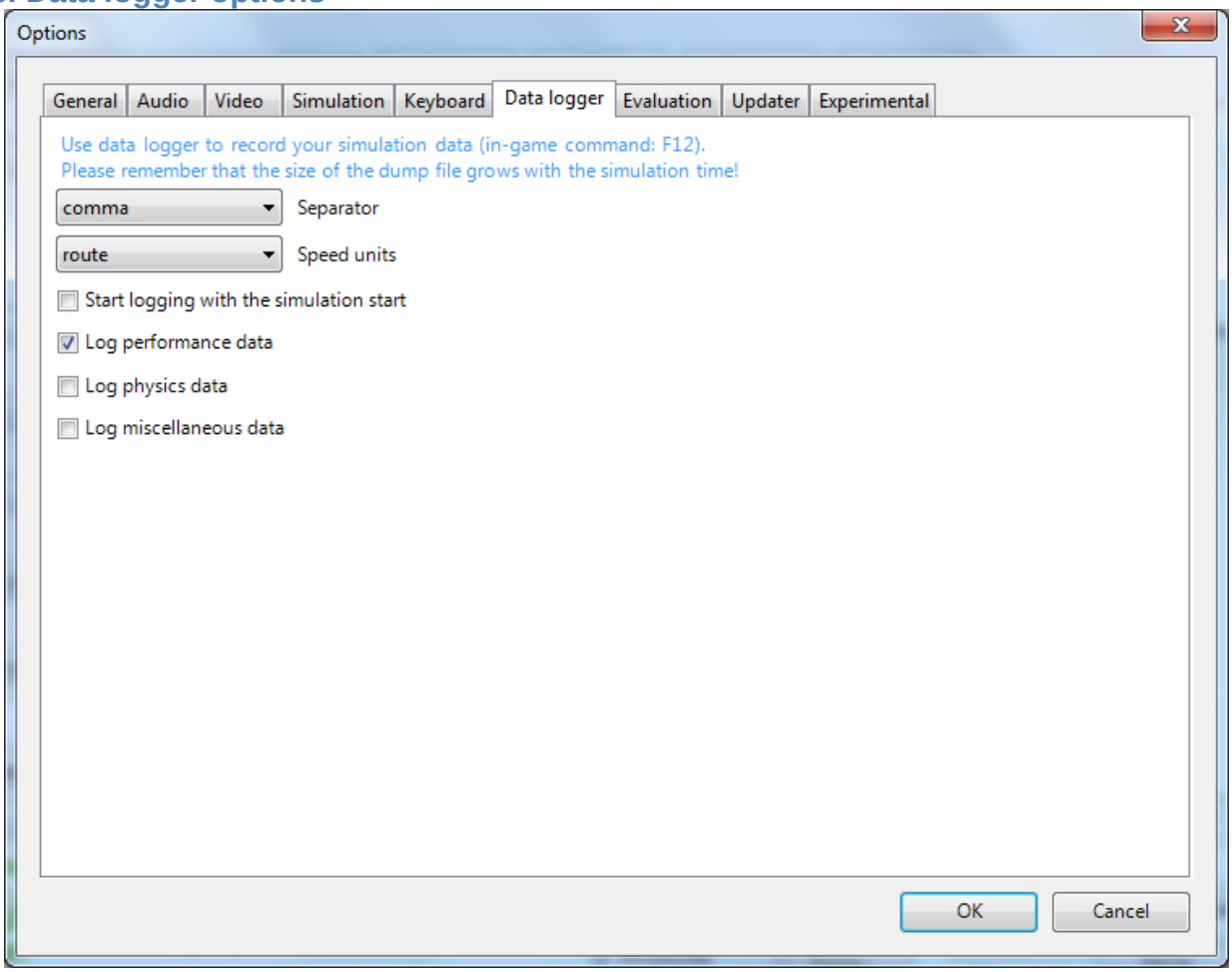
You can modify them by clicking on a field and pressing the new desired key. Three symbols will appear at the right of the field: with the first one you validate the change, with the second one you cancel it, with the third one you return to the default value.

By clicking on “Check” OR verifies if the changes made are compatible, that is that there is no key that is used for more than a command.

By clicking on “Defaults” all changes made are reset, and the default values are reloaded.

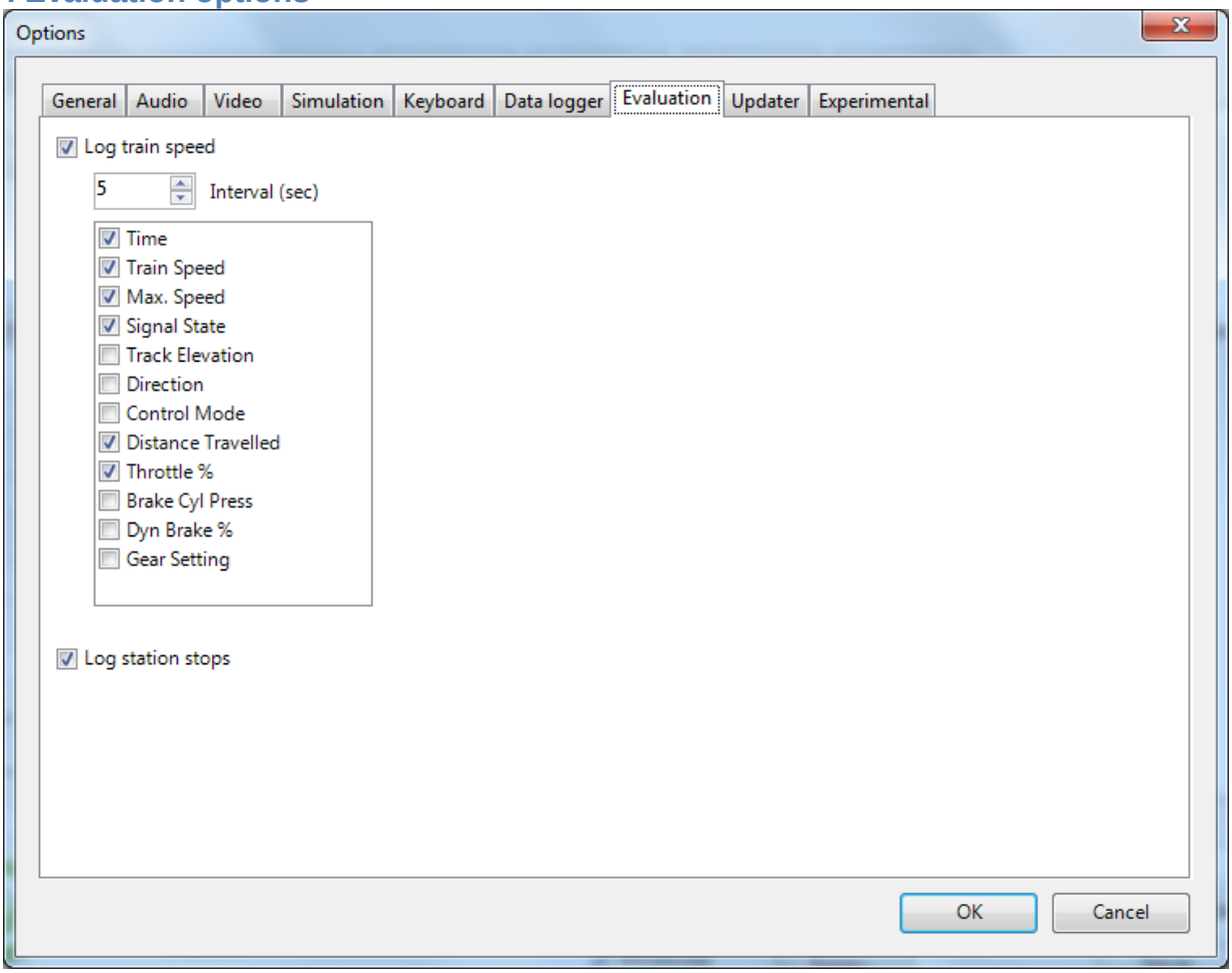
By clicking on “Export” a printable file “Open Rails Keyboard.txt” is generated on the desktop, with all links between command and keys.

6.6. Data logger options



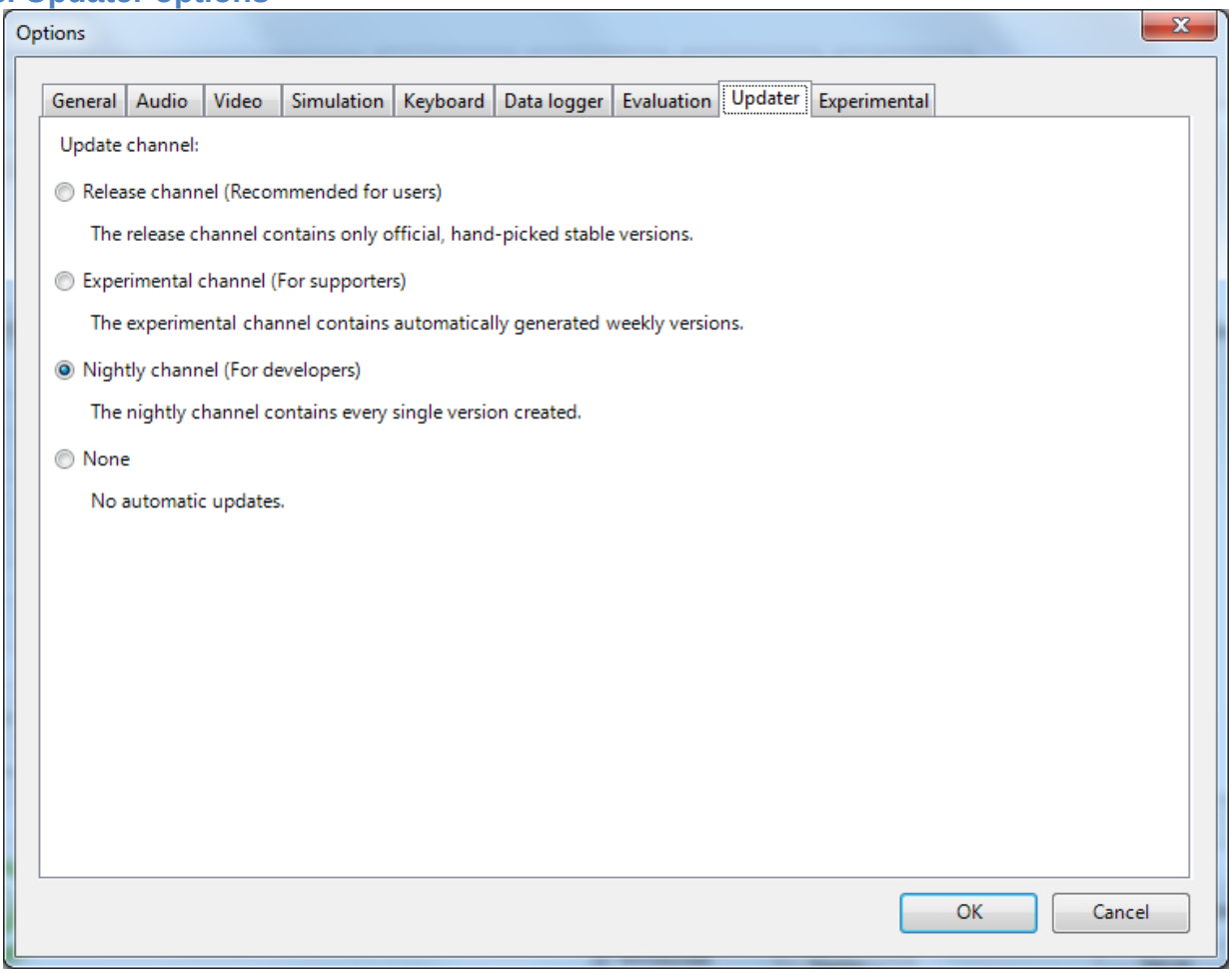
By selecting option “Start logging with the simulation start” or by pressing F12 a file with name dump.csv is generated within the Open Rails root folder. This file can be used for later analysis.

6.7. Evaluation options



When data logging is started (see preceding paragraph) also data selected in this panel are logged, allowing a later evaluation on how the activity has been executed by the player.

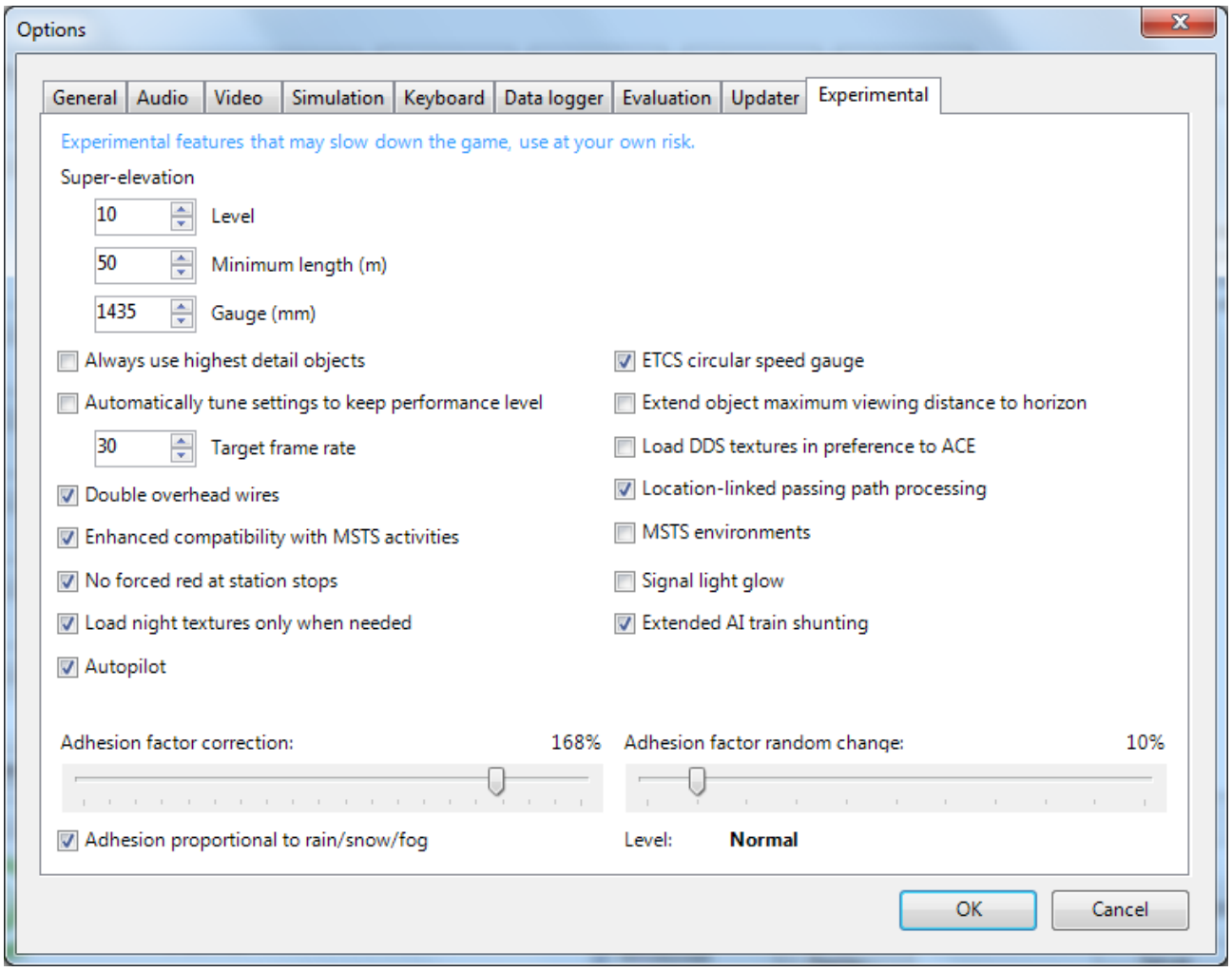
6.8. Updater options



These options control which update channel is active (see also here). The various options available are self-explanatory.

6.9. Experimental Options

There are some experimental features being introduced that may be turned on and off through the "Experimental" tab of the Options window, as shown below:



6.9.1. Super-elevation

OR supports super elevation for long curved tracks, by giving the "Amount" larger than 0. "Min Length" determines the length of the shortest curve to have super-elevation. You need to choose the proper gauge for your route, otherwise, some tracks may not be properly shown.

When super-elevation is selected, two viewing effects occur at runtime:

1) if an external camera view is selected, tracks and the above running train will be shown as inclined towards the internal part of the curve

2) if the cab view is selected, the cab itself will be shown as inclined towards the internal part of the curve, while the external world will be shown as inclined towards the external part; the

ratio of these two inclinations can be changed at runtime by repeatedly pressing Alt-R. Four

possible ratios are possible.

OR implements super elevated tracks using Dynamic Tracks. You can change the appearance of tracks by creating a TrProfile.sft in the TrackProfiles folder of your route. Discussions about this can

be found from http://www.elvastower.com/forums/index.php?/topic/21119-superelevation/page__view__findpost__p__115247.

6.9.2. Always use high detail objects

When this option is selected, object are always shown at the highest detail, even if they are distant (technically speaking always the highest LOD is used for rendering). Selecting this option can improve viewing quality but can slow down frame rate.

6.9.3. Automatic tune settings to keep performance level

When this option is selected OR tends to keep the selected frame per seconds rate. To do this it decreases or increases the viewing distance of the standard terrain.

6.9.4. Double Overhead Wires

MSTS uses single wire for electrified routes, you can check this box so OR can show two overhead wires that are most common.

6.9.5. Enhanced compatibility with MSTS activities

When this option is selected OR interprets activity related files in a way that is more similar to MSTS.

It is suggested to select this option when executing activities developed for MSTS.

6.9.6. No forced red at station stops

In case a signal is present subsequently to a station platform and in the same track section (no switches in between), as default OR sets the signal to red until the train has stopped and from that moment up to two minutes before starting time. This is useful to organize train meets and takeovers, however it does not always correspond to reality neither to MSTS operation. So with this option the player can decide which behavior the start signal must have. The option is operating only if also the Enhanced compatibility with MSTS activities option is selected and no Timetable mode operation is under way.

6.9.7. Load night textures only when needed

As default OR loads night textures together the day textures also at daytime.

To reduce loading time and spare memory used, when this option is selected, night textures aren't loaded at daytime and are loaded only at sunset (if the game proceeds up to sunset time).

6.9.8. ETCS circular speed gauge

When this option is selected, it is possible to add to the cabview a circular speed gauge accordingly to the European standard train control system ETCS.

for content developers: The gauge is added with the insertion of a block like the following in the .cvf file:

```
Digital (
        Type ( SPEEDOMETER DIGITAL )
        Style ( NEEDLE )
        Position ( 160 255 56 56 )
        ScaleRange ( 0 250 )
        Units ( KM_PER_HOUR )
)
```

6.9.9. Extend object maximum viewing distance to horizon

With this option selected all objects viewable up the viewing distance as defined in the Video Options are displayed. As a default ORTS displays only objects up to 2000 m. distance. Selecting this option improves display quality but may reduce frame rate.

6.9.10. Load DDS textures in preference to ACE

Open Rails is capable to load ACE and DDS textures. If only one of both is present, it is loaded. If both are present, the ACE texture is loaded unless this option has been selected.

6.9.11. Location-linked passing path processing

When this option is NOT selected, ORTS acts in a similar way to MSTS, that is if two trains meet whose paths share some track section in a station, but are both provided with passing paths as defined with the MSTS Activity Editor, one of them will run through the passing path, therefore allowing the meet. Passing paths in such case are available only to the trains whose path has passing paths.

When this option is selected, ORTS makes available to all trains the main and the passing path of the player train. Moreover, it takes into account the train length in selecting which path to assign to a train in case of a meet.

for content developers: A more detailed description of this feature can be found under the paragraph with same name in chapter “Open Rails train operation”.

6.9.12. MSTS Environments

As a default ORTS uses its own environment files and algorithms, e.g. for night sky and for clouds.

With this option ORTS applies the MSTS environment files. This includes support of Kosmos environments, even if the final effect may be different from the MSTS one as of now.

6.9.13. Signal light glow

When this option is set, to signal semaphores a glowing is added when seen at distance, so that they are visible at greater distance. There are routes where this effect has already been natively introduced; for these ones, it is not advised to use this option.

6.9.14. Extended AI train shunting

When this option is selected, further AI train shunting functions are available. This allows for more interesting and varied activities. If an activity is run which makes use of these function, the option must be selected.

Following additional shunting functions are available:

- AI train couples to static consist and restarts with it
- AI train couples to player or AI train and becomes part of it; coupled AI train continues on its path
- AI train couples to player or AI train and leaves to it its cars; coupled and coupling train continue on their path
- AI train couples to player or AI train and “steals” its cars; coupled and coupling train continue on their path
- AI train uncouples any number of its cars. AI train uncouples any number of its cars; the uncoupled part becomes a static consist. With the same function it is possible to couple any number of cars from a static consist.

This option is active only outside Timetable mode and if the “Enhanced compatibility with MSTS activities” option has been selected.

for content developers: A more detailed description of this feature can be found under the paragraph with same name in chapter “Open Rails train operation”.

for content developers: Selecting this option enables also the waiting points indicating an absolute time-of-day instead of a waiting point duration. A more detailed description of this feature can be found under the related paragraph in chapter “Open Rails train operation”.

6.9.15. Autopilot

With this option enabled and when in activity mode, it is possible to stay in the cab of the player train, but to let Open Rails move the train, respecting path, signals, speeds and station stops.

It is possible to switch at run time the player train from autopiloted to player driven. The Autopilot mode is described here.

6.9.16. Adhesion factor correction

This percentage factor is multiplied with the adhesion. Therefore lower values of the slider reduce adhesion and cause more frequent wheel slips and therefore a more difficult, but more challenging driving experience.

6.9.17. Adhesion factor random change

This percentage factor randomizes the adhesion factor corrector. The higher the factor, the higher the adhesion variations.

6.9.18. Adhesion proportional to rain/snow/fog

When this option is selected, adhesion becomes dependent from the intensity of rain and snow and from the density of fog. Intensities and density can be modified at runtime by the player.

7. Driving a train

7.1. Game loading

Once you have pressed “Start”, OR loads and processes all data needed to run the game. During this phase, the route splash screen is shown. If the same session was already loaded precedently, a bar showing loading progress is shown at the bottom of the display. During loading the log file OpenRailsLog.txt file, if selected, already begins storing data.

7.2. Entering simulation

At the end of the loading phase, you are within the cab of the train he drives. Depending from the configuration of the activity (in case of activity mode), your train will be in motion or stopped. In this second case, if the train is led by an electric loco, as first operation you have to raise the pantograph (key P).

7.3. Open Rails Driving Controls

Open Rails follows MSTS very closely, providing controls to drive steam, electric and diesel locos, both on their own or working together.

A very wide range of systems and instruments are supported as specified in the ENG and CVF files.

To drive the train, you have at your disposal a set of keyboard commands that is equivalent to those of MSTS, plus some new ones. You can get a previously printed version of the command set as described in paragraph 6.5 (Keyboard options), or you can press F1 to immediately get a scrollable window as shown below.

Alternatively, you can activate the cabview controls with mouse click (buttons) and mouse drag (levers).

7.3.1. Throttle Control

Steam locomotives have a continuous throttle or regulator, but many diesel and electric locos have a notched throttle which only moves in steps. To avoid jerks, some of these steps may be "smooth", where the power is gradually and automatically adjusted to achieve the setting.

7.3.2. Dynamic Braking

Dynamic braking is the use of the traction motors of a loco (electric or diesel-electric) as generators to slow the train. Initially, dynamic braking was applied in mountainous territory where conventional freight-car brakes were prone to overheating on long downgrades. It was also limited to speeds above 10mph. Dynamic braking controls are usually notched.

In OR, the dynamic brake (via the keys , and .) is not available unless the throttle is fully closed and similarly the throttle is not available unless the brake is fully released.

As defined in the CVF file, the tractive and braking forces may be shown on two instruments, on one instrument with two needles or on a single instrument where the braking is shown as a negative value.

7.3.3. Combined Control

Some locos are fitted with "combined control" where a single lever is used to provide throttle and brake control together, with negative throttle positions used to apply the brake. The brake element may be either dynamic or conventional train brakes.

There may be a delay changing between throttle and brake.

7.3.4. Refill

Diesel and steam locomotives must refill their supplies of fuel occasionally, perhaps daily, but steam locomotives need water more frequently and have a range of little more than 100 miles. Use the "T" key to refill with fuel or water.

If the loco or tender is alongside the pickup point, e.g. a water column, then the refilling takes place as the key is held down. If the loco is further away, then the distance to the nearest pickup is given instead.

7.3.5. Examples of Driving Controls

for content developers:

For continuous throttle, see MSTS model `...\TRAINS\TRAINSET\ACELA\acela.eng`

For a notched non-smooth throttle, see `...\TRAINS\TRAINSET\GP38\gp38.eng`

For a combined throttle and dynamic brake, see `...\TRAINS\TRAINSET\DASH9\dash9.eng`

For a combined throttle and train brake, see
`...\MSTS\TRAINS\TRAINSET\SERIES7000\series7000.eng`

7.4. Driving aids

Open Rails provides a rich amount of driving aids, which support the player during train operation.

7.4.1. Basic Head Up Display (HUD)

By pressing F5 you get at the top left of the display some important data are displayed in the so-called Head Up Display (HUD). If you want the HUD to disappear, press F5 again.

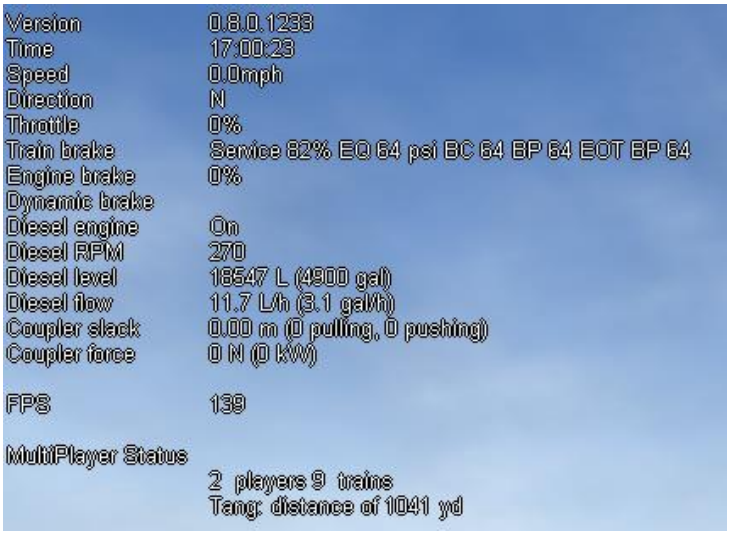
The HUD has 6 different pages. The basic page is shown at game start. To sequentially switch to the other pages Shift-F5 has to be pressed. After having switched through all extended HUD pages, the basic page is displayed again.

The basic page shows fundamental info. The other pages go into detail, and are used mainly for debugging or to get deeper information on how OR behaves. They are listed in the "Analysis tools" subchapter.

Selecting the F5 key displays the HUD information in the current version of Open Rails software. The following information is displayed:

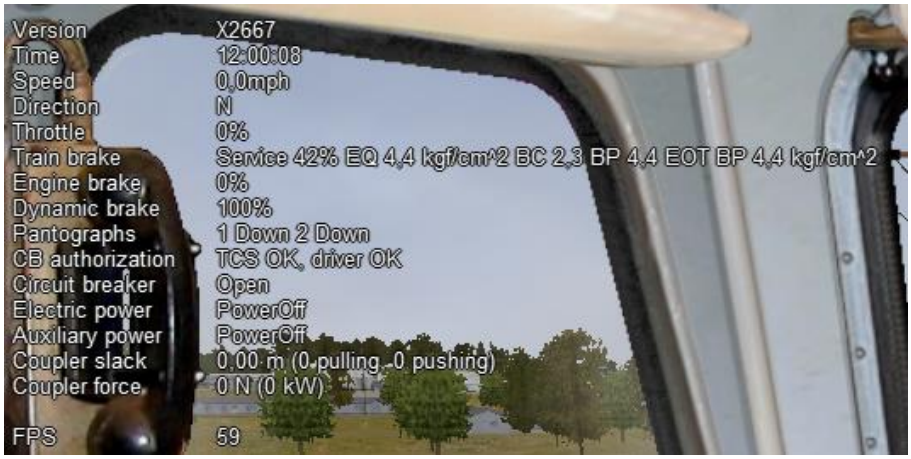
- Version = The version of the Open Rails software you are running
- Time = Game time of the Activity
- Direction = Position of the Reverser - Electric, Diesel and Steam.
- Throttle = Displays the current position of throttle expressed as a percentage of full throttle. Throttle correctly uses Notches and configured % of power for Diesel engines or
- % of throttle for steam engines.
- Train Brake = Shows the current position of the train brake system and the pressure value of the train brakes. Braking correctly reflects the braking system used; hold/release, self- lapping or graduated release. The Train brake HUD line has two Brake Reservoir pressure numbers: the first is Equalization Reservoir (EQ) and Brake Cylinder (BC) pressure numbers. The BP numbers reflect the brake pressure in the lead engine and the second is at the last car of the train. The unit of measure used for brake pressure is defined by option 6.1.8 "Pressure unit".
- Engine Brake = percentage of independent engine brake. Not fully releasing will impact train brake pressures.
- Speed = the speed in Miles/Hr or Kilometers/Hr
- Coupler Force = Force exerted on couplers in Newtons. (1 Lbs of force = 4.45 Newtons)
- FPS = Number of Frames rendered per second
- Multi-player status = the name of other players and the distance of their trains (in case the game has been started in multiplayer mode)

An example of the basic HUD for Diesel locomotives follows:



In case of a gear-based engine, at the right of the Diesel RPM the actual gear is displayed; N means no gear inserted.

7.4.2. Electric loco – Additional information



For electric locos also the info about pantograph state is present. In release 1.0 circuit breaker state and power state are directly consequence of the pantograph state.

7.4.3. Steam Engine HUD – Additional Information

When using a steam engine the following additional information is displayed in the HUD:

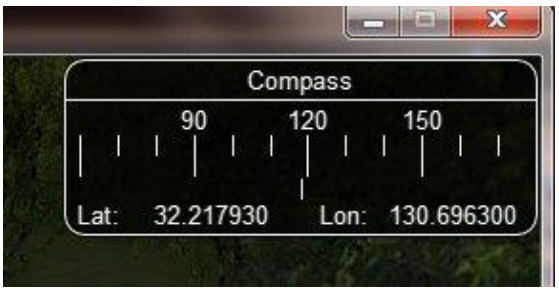
- Boiler pressure
- Steam Generation in lbs/h and based on key physics data in the ENG file. Steam generation is assumed to be ‘perfect’ from an operational perspective – fire temp, water level, etc. and is limited by the “MaxBoilerOutput” parameter in the ENG file
- Steam Usage in lbs /h and based on entirely new physics code developed by the Open Rails team. It is calculated by parsing the eng file for the following parameters: number of cylinders; cylinder stroke; cylinder diameter; boiler volume; maximum boiler pressure; maximum boiler output; exhaust limit; and basic steam usage
- Water level.

An example of the basic HUD for Steam locomotives follows:

The default setting is automatic firing. If manual firing is engaged (with Ctlr+F), then additional information is included:

7.4.4. Compass Window

Open Rails software displays a compass that provides a heading based on the direction of the camera together with its latitude and longitude.



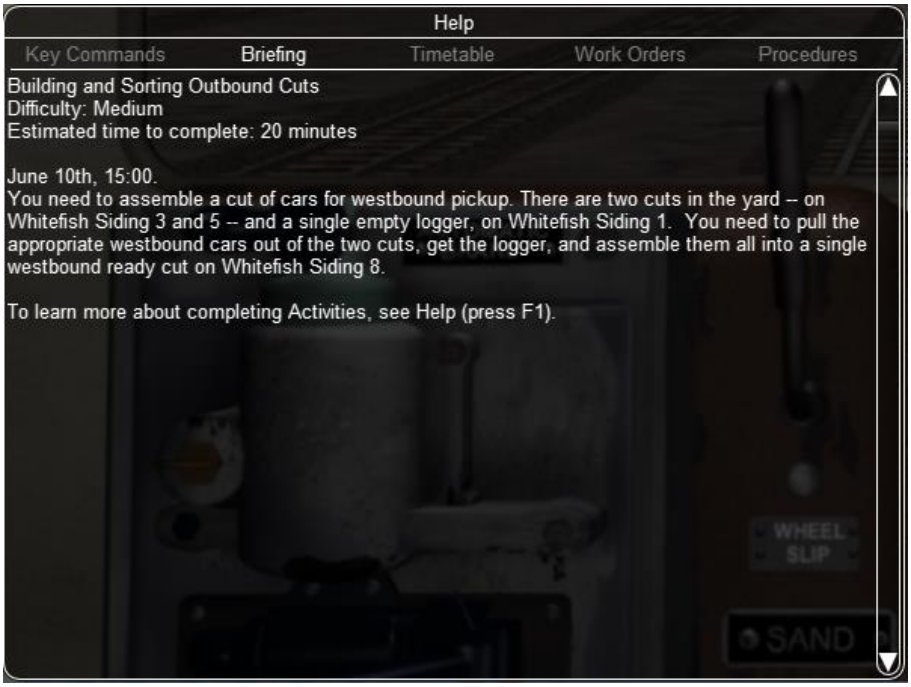
To activate the compass window press the 0 (zero) key. To deactivate the compass window, press the 0 (zero) key a second time.

7.4.5. F1 Information Monitor

Open Rails software now offers in a tabbed format using the F1 key following panels:

- Keyboard commands

- Briefing: displays what the activity creator has entered as info to be provided to the player about the activity



- Timetable: shows the list of the station stops, if any, with scheduled and actual arrival and depart
- Work order: if defined by the activity creator, lists the coupling/uncoupling operations to be performed; when an operation has been completed, the string “Done” appears in the last column

7.4.6. F4 Track Monitor

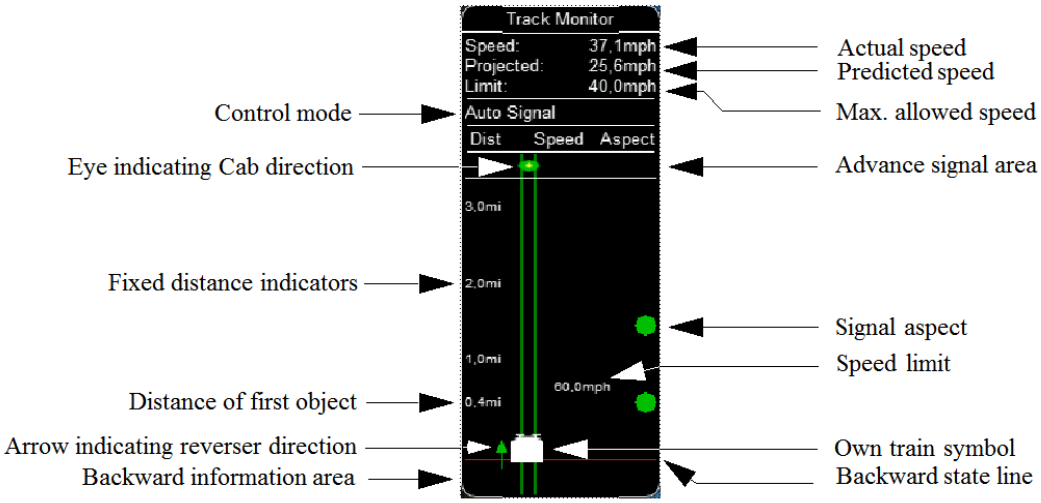
This window, that is displayed by pressing F4 has two different layouts according the the train’s control mode: Auto mode or Manual mode / Explorer mode (it is strongly suggested to follow the link and read the related paragraph).
Auto mode is the default mode running activities or timetables.
There are however two main cases where you have to pass to “Manual” mode pressing Ctrl-M:

- when the activity foresees shunting without a predefined path
- when the train runs out of control due to SPAD (passing red signal) or by error exits the predefined path; if such situations occur you usually get an emergency stop. To reset such emergency stop and then move to correct the error, you first have to pass to Manual mode.

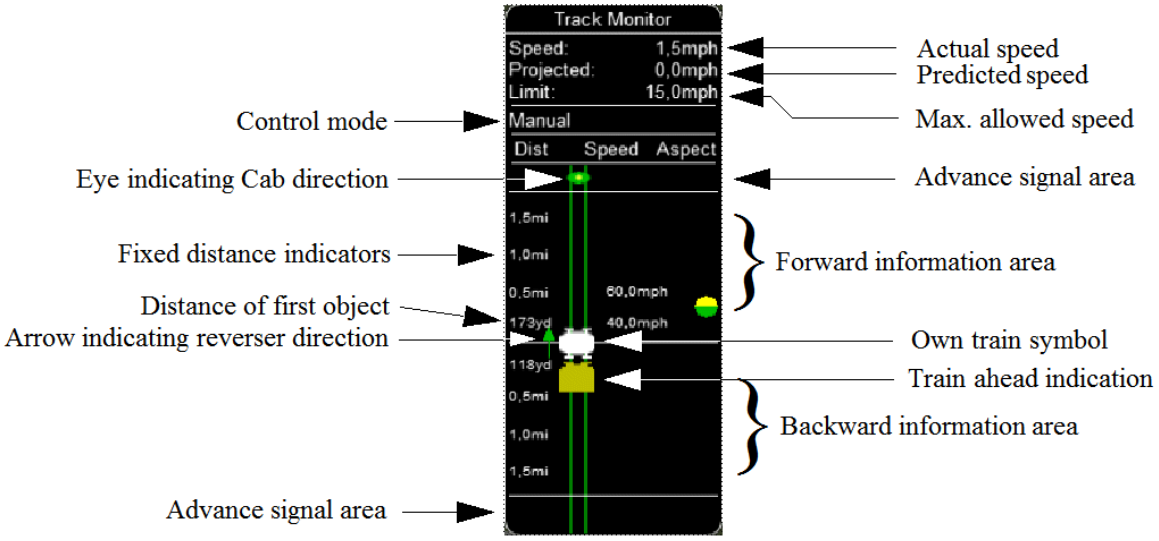
Passing to manual mode is achieved by pressing Ctrl-M with stopped train.

You can return to auto mode by pressing Ctrl-M again when the head of the train is again on its correct path, with no SPAD situation. In standard situations you can return to auto mode also with moving train. Details are described in the paragraph of the above link.

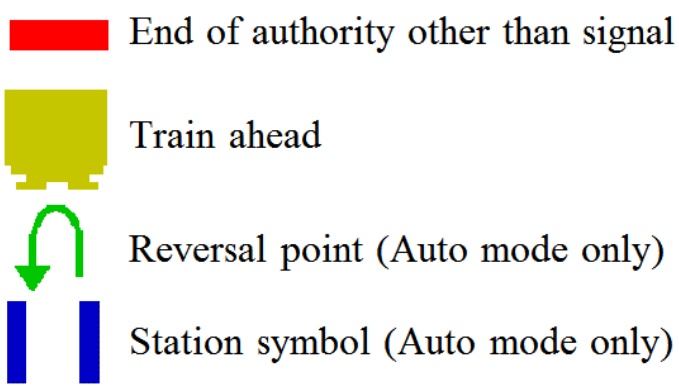
Display in Auto mode:



Display in Manual mode / Explorer mode:



Displayed symbols (common for Auto and Manual mode unless indicated otherwise) :



Notes:

- Distance value is displayed for first object only, and only when within distance of first fixed marker.
  Distance is not shown for next station stop.
- When no signal is within the normal display distance but a signal is found at a further distance, the signal aspect is displayed in the advance signal area. The distance to this signal is also shown.
  This only applies to signals, not to speedposts.
- For Auto mode :
  if the train is moving forward, the line separating the Backward information area is shown in red, and no Backward information is shown.

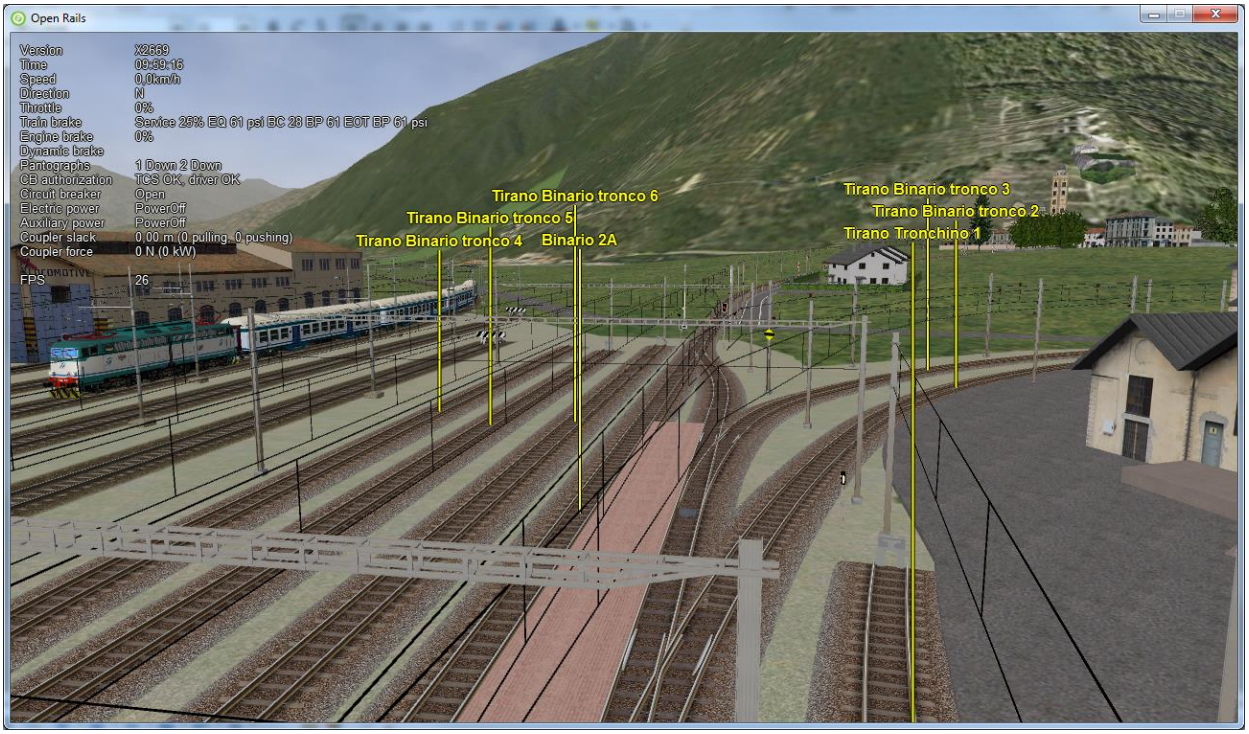
If the train is moving backward, the separation line is shown in white, and Backward information is shown if available.

- For Manual mode :
  if the train is on it's defined path (and toggle back to Auto control is possible), the own train symbol is shown in white, otherwise it is shown in red.
- The colour of the track-lines are an indication of the train speed with regards to the maximum allowed speed :
  - Dark green : low speed, well below allowed maximum
  - Light green : optimal speed, just below maximum
  - Orange : slight overspeed but within safety margin
  - Dark red : serious overspeed, danger of derailment or crashing

Note that the placement of the objects with respect to the distance offset is indicative only. If multiple objects are placed at short intermediate distances, the offset in the display is increased such that the texts do not overlap. As a result, only the first object is always shown at the correct position, all other objects are as close to their position as allowed by other objects closer to the train.

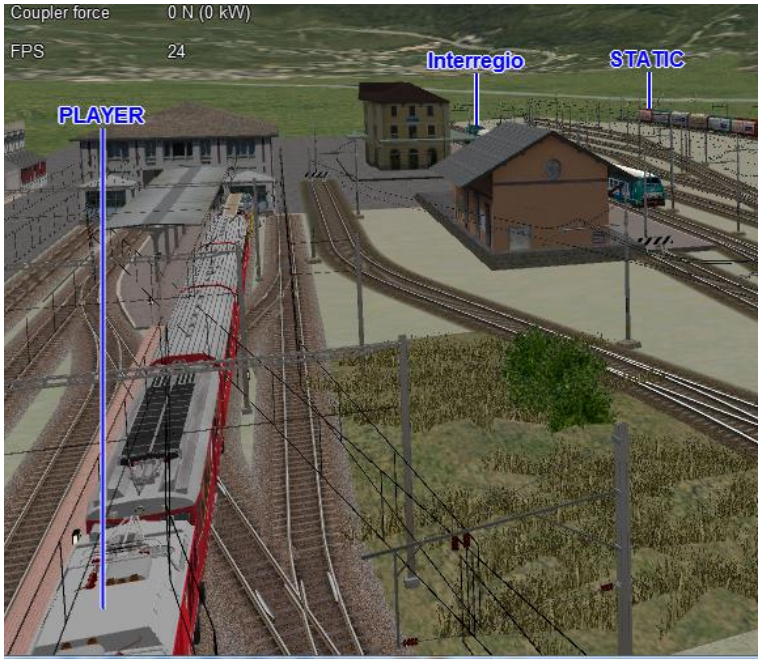
7.4.7. F6 Siding and Platform Names

Hit the F6 key to bring up the siding and platform names within a region. These can be crowded so hitting Shift-F6 will cycle through platforms only, sidings only, showing and showing both. Hitting F6 again removes both siding and platform names.

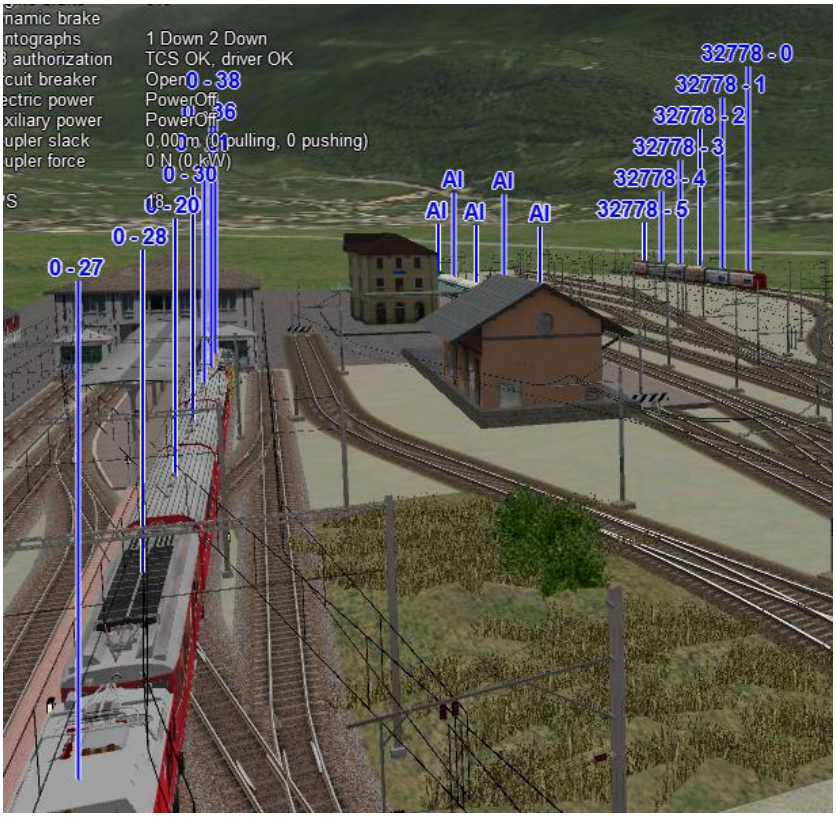


7.4.8. F7 Train Names

Hitting F7 key displays train service names (player train always has “Player” as identification).



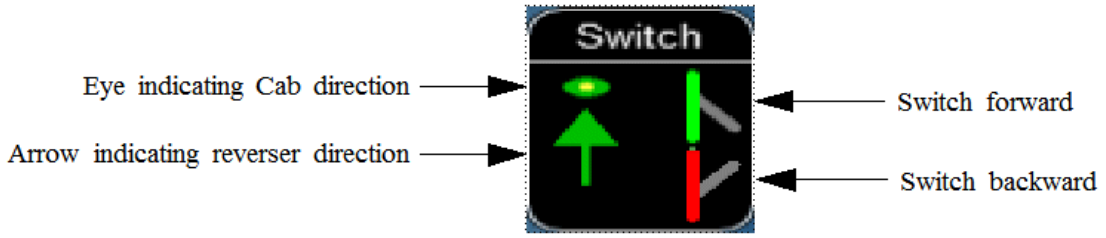
Hitting Shift-F7 displays the rolling stock IDs.



In a multiplayer session, player-controlled trains will have the id specified by the player:

7.4.9. F8 Switch Monitor

Use the Switch Monitor to see the direction of the turnout directly in front and behind the train.



There are 4 ways to change that direction:

- Click on the turnout icon in the Switch Monitor.
- Press the G key (or, for the turnout behind the train, the Shift+G key)
- Hold down the Alt key and use the left mouse button to click on the switch in the Main Window
- Use the dispatcher window, as described here

Pls. note that with the last two methods you can throw any switch, not only the one in front and the one behind the train.

Pls. note also that not all switches can be thrown: in some cases the built-in AI dispatcher holds the switch in a state, to allow trains (especially AI trains) to follow their predefined path.

Arrow and eye have the same meaning as in the track monitor. The switch is red when it is reserved or occupied by the train, and green when it is free.

Switch shown in green can be operated, switch shown in red is locked.

7.4.10. F9 Train Operations Monitor

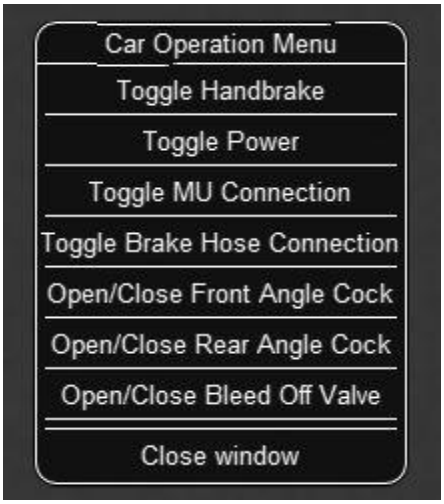
Open Rails Train Operations window is similar in function to the F9 window in MSTS, but includes additional features to control the air brake connections of individual cars. For instance, it is possible to control the connection of the air brake hoses between individual cars, or to

uncouple cars with air brakes released so that they will coast.
The unit which the player has selected as the unit from which to control the train, i.e. the lead unit, is shown in red.
Cars are numbered according to their UiD in the Consist file (.con) or UiD in the Activity file (.act). Scrolling is accomplished by clicking on the arrows at the left and right bottom corner of the window.

Clicking on the coupler icon between any two cars uncouples the consist at that point.
You can also uncouple cars from your player train by pressing the U key and clicking with the mouse on the couplers on the main window.

By clicking on any car in the above window, the Car Operation Menu appears.

Through this menu it is possible:
- to apply and release the handbrake of the car
- to power on or power off the car (if it is a loco). This applies both for electric and diesel locos
- to connect/disconnect loco operation from that of the player loco.
- to connect or disconnect the air hoses from the rest of the consist
- to toggle the angle cocks on the air hoses at either end of the car between open and closed
- to toggle the bleed valve on the car to vent the air pressure from the car's reservoir and release the air brakes in order to move the car without brakes (e.g. humping, etc.)



By toggling the angle cocks on individual cars it is possible to close selected angle cocks of the air hoses so that when the cars are uncoupled, the air pressure in the remaining consist (and optionally in the uncoupled consist) is maintained. (The remaining consist will not go into "Emergency" state.)
In Open Rails, opening the bleed valve on a car or group of cars performs two functions: it vents

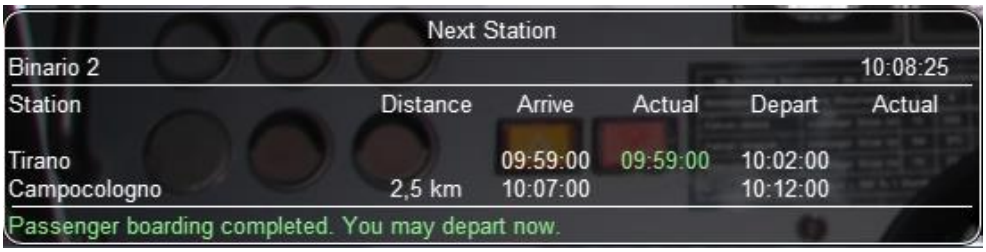
the air pressure from the brake system of the selected cars, and also bypasses the air system around the cars if they are not at the end of the consist so that the rest of the consist remains connected to the main system. In real systems the bypass action is performed by a separate valve in each car.

More information about manipulating the brakes during coupling and uncoupling can be found here .

7.4.11. F10 Activity Monitor

The Activity Monitor is similar in function to MSTS. It records the Arrival time of your train versus the Actual time as well as the Departure Time.

A text message alerts the engineer as to the proper departure time along with a whistle or other departure sound.

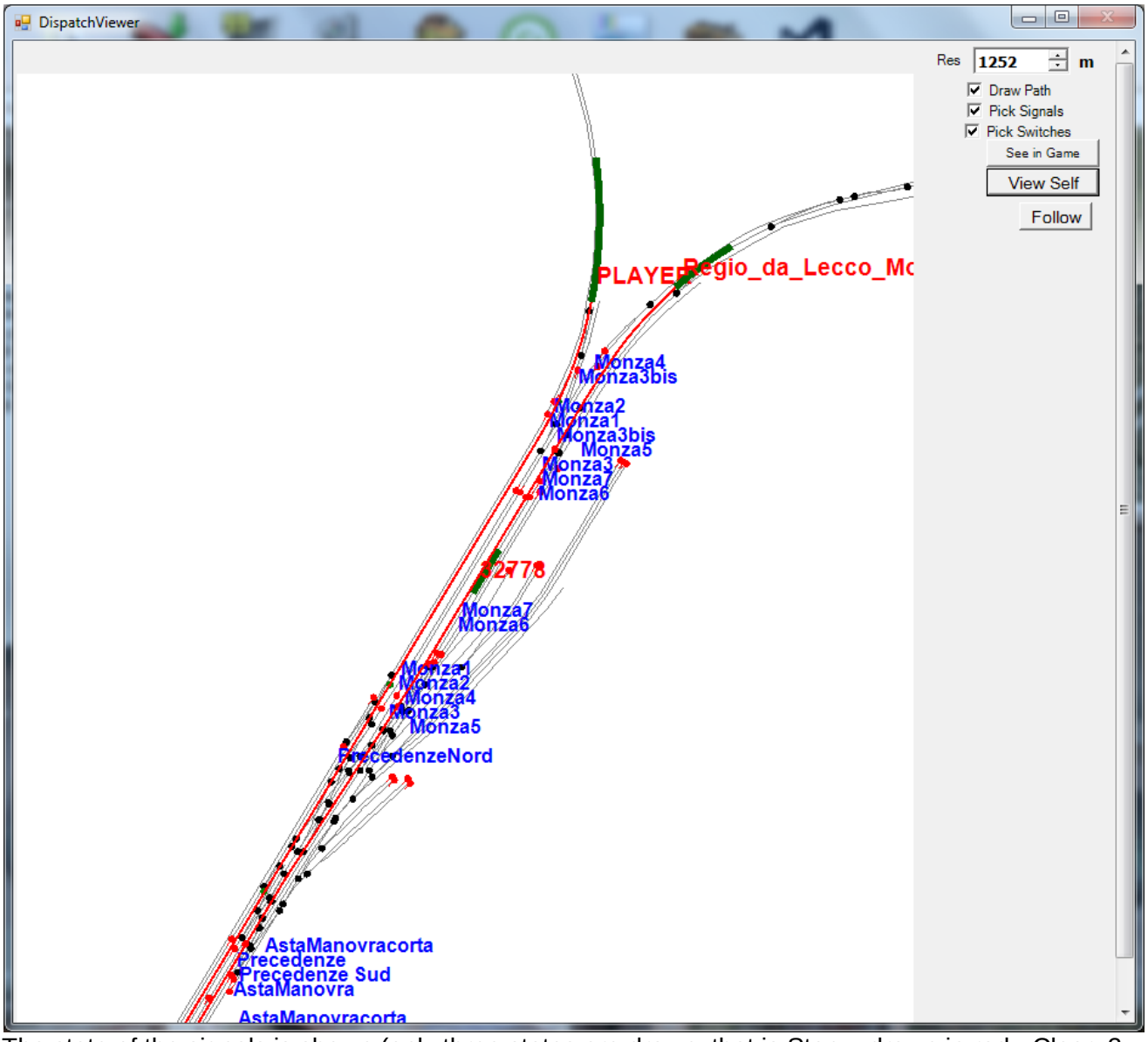


7.5. Dispatcher window

The dispatcher window is a very useful tool to monitor and control train operation.

The dispatcher window is opened by pressing Ctrl-9. The window is opened in a minimized way, so you have to click on one of the OR icons in the taskbar to open it. The window is resizable and can also be maximized, e.g. on a second display. You can define the level of zoom either changing the value within the “Res” box or using the mousewheel. You can pan through the route by moving the mouse pressing the left button. You can hold shift key while click the mouse in a place in the map, which will quickly zoom in with that place in focus. You can hold Ctrl while click the mouse in a place in the map, which will zoom out to show the whole route. Holding Alt and click will zoom out to show part of the route.

The dispatcher window shows the route layout and monitors the movement of all trains. While the player train is identified by the “PLAYER” string, all other trains are identified by their service name.

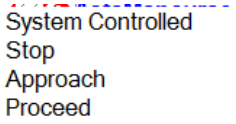


The state of the signals is shown (only three states are drawn, that is Stop – drawn in red-, Clear_2 - drawn in green -, while all signals with restricting aspect are drawn in yellow.

Also the state of the switches is shown. A switch shown with a black dot indicates main route, while a grey dot indicates side route.

When the “Draw path” is checked, the first part of the path that the train will follow is drawn in red. If a trailing switch in the path is not in the correct position for the path, a red X is shown on it.

When left- or right-clicking on a signal, following pop-up menu appears:



You can force the signal to Stop, Approach or Proceed. Later you can return it to System Controlled mode.

When left- or right-clicking on a switch, a small pop-up menu with the two selections “Main route” and “Side route” appears. Clicking on them you can throw the switch, provided the OR AI dispatcher allows that.

Referring to AI train, as a general rule you can command their signals, but not their switches, because AI trains aren't allowed to exit their path.

The two checkboxes “Pick Signals” and “Pick Switches” are checked as default. You can uncheck one of them when a signal and a switch are superimposed in a way that it is difficult to select the desired item.

You can click a switch (or signal) and press **Ctrl+Alt+G** to jump to that switch with the free-roam camera.

If you click on “View Self” the dispatcher window will center on the player train. However, if the train moves, centering will be lost.

You can select a train by left-clicking with the mouse its green reproduction in the dispatcher window, more or less half way between the train's head and its name string. The train body becomes red. At that moment, if you click on “See in game” the main window will show such train (you had to be using cameras 2, 3 or 4 before). Display of the new train could need some time if the train is far away from the previous camera view.

Take into account that continuous switching from train to train, especially if trains are far away, can lead to memory overflows.

If after a train selection you click on “Follow” the dispatcher window will remain centered on the train.

7.6. Additional train operation commands

OR supports an interesting range of additional train operation commands. Some significant ones are described here below.

7.6.1. Diesels power on/off

With key Y the player diesel engine is alternatively powered on and off. At game start the engine is powered on.

With keys Shift-Y the helper diesel locos are alternatively powered on and off. At game start the engines are powered on.

Take note that with use of the Car Operation Menu you can also power on and off the helper locos individually.

7.6.2. Initialize brakes

Entering this command at any moment the brakes are fully released. Check the keyboard assignment for the keys to be pressed. The command can be useful in three cases:

- a good number of locos do not have correct values for brake parameters in the .eng file; MSTS doesn't care about this; however OR uses all these parameters, and it may happen that they don't allow the brakes to fully release; of course it would be more advisable to correct such parameters
- it may happen that the player does not want to wait the time needed to recharge the brakes; the use of such command in this case is not prototypical of course
- to immediately connect brake lines and recharge brakes after a coupling operation; again, the use of the command is not prototypical.

7.6.3. Connect/disconnect brake hose

This command should be used after a coupling/decoupling. As the code used depends on keyboard layout, check the keys to be pressed as described in paragraph 6.5 or by pressing F1 at runtime.

More on connecting brakes and manipulating the brake hose connection can be found here and here.

7.6.4. Doors and mirror commands

Take note that the standard keys for these commands are different from those of MSTS.

7.6.5. Wheelslip reset

With keys Ctrl-X you get an immediate wheelslip reset.

7.6.6. Toggle advanced adhesion

Advanced adhesion can be enabled/disabled by pressing Ctrl-Alt-X.

7.6.7. Request to clear signal

When the player train has in front of it or at its rear a red signal, it is sometimes necessary to ask for authorization to pass the signal. This can be requested by pressing Tab for the signal in front and Shift-Tab for the signal on the rear. You get back a vocal message declaring if you got authorization or not. On the Track monitor window the signal colours change from red to red/white if there is authorization.

7.6.8. Change Cab - Ctrl+E

All locos and also some passenger cars have a forward-facing cab which is configured through an entry in the ENG file. For example, the MSTS Dash9 file *TRAINSET\DASH9\dash9.eng* contains the entry:

```
CabView ( dash9.cvf )
```

Where a vehicle has a cab at both ends, the ENG file may also contain an entry for a reversed cab:

```
CabView ( dash9_rv.cvf )
```

OR will recognise the suffix `_rv` as a rear-facing cab and make it available as follows.

When double-heading, banking or driving multiple passenger units (DMUs and EMUs), your train will contain more than one cab and OR allows you to move between cabs to drive the train from a different position. If you change to a rear-facing cab, then you will be driving the train in the opposite direction.

If there are many cabs in your train, pressing Ctrl+E moves you through all forward and rear-facing cabs up to last cab in the train. If you end up in a rear-facing cab, your new “forward” direction will be your old “backward” direction. So you will now drive the train in the opposite direction.

A safety interlock prevents you from changing cabs unless the train is stationary and in neutral with the throttle closed.

7.6.9. Train oscillation

You can have train cars oscillating by hitting Ctrl-V; if you want more oscillation, click Ctrl-V again. Four levels, including the no-oscillation level, are available.

7.7. Autopilot mode

Autopilot mode is not a simulation of a train running with cruise control; instead, it is at first place a way to test activities more easily and faster; it can however also be used to run an activity (or part of it, as it is possible to turn on and off autopilot mode at runtime) as a trainspotter or a visitor within the cab.

Autopilot mode is enabled with the related checkbox in the Experimental Options. It is active only in activity mode (no explorer, no timetable mode).
Starting the game with any activity you are in player driving mode. If you press Alt-A, you enter the autopilot mode: you are in the loco's cabview with the train moving autonomously accordingly to path and station stops and of course respecting speed limits and signals. You still have control over horn, bell, lights, doors, and some other controls that do not affect train running. The main levers are controlled by the autopilot mode, and indications are correct.
You can at any moment switch back to player driven mode, by pressing ALT/A, and can again switch to autopilot mode by pressing again ALT/A.
When in player driven mode you can also change cab or direction. However, if you return to autopilot mode, you must be on the train's path; other cases are not managed. When in player driven mode you can also switch to manual, but before returning to autopilot mode you must first return to auto mode.
Station stops, waiting points and reverse points are synchronized as far as possible in the two modes.
Cars can be uncoupled also in autopilot mode (but check that the train will stop enough time, else better pass to player driven mode). A static consist can also be coupled in autopilot mode.
Request to clear signal (TAB) works in the sense that the signal opens. However in autopilot mode by the moment the train stops. You have to pass to player driven mode, to pass the signal and then you can return to autopilot mode.

Please note that if you run with Advanced Adhesion enabled, you can have wheelslips when passing from autopilot mode to player driven mode.

The jerky movements of the levers in autopilot mode are due to the way OR pilots the train.

7.8. Changing viewpoint

You have all MSTS camera views available. The free roaming camera is selected with key 8, and is always selectable.

Pressing Alt-9 (Numkey) when in front or rear outside camera or passenger camera shifts the position of the camera 1 car further towards the end. Pressing Alt-3 (Numkey) shifts the camera 1 car towards the head of the train. Pressing Alt-1 (Numkey) moves the camera to the first car, while pressing Alt-7 (Numkey) moves the camera to the last car.

Pressing Alt-9 when in front, rear or trackside camera moves the camera to another train and so on.

Pressing 9 returns the camera to the player train.

7.9. Toggling from windowed mode to full-screen

You can toggle at any moment from windowed mode to full-screen by pressing Alt-Enter.

7.10. Modifying the Game Environment

7.10.1. Time of Day

When in activity mode Open Rails software reads the StartTime from the MSTS .act file to determine what the game time is for the activity. In combination with the longitude and latitude of the route and the season, Open Rails computes the actual sun position in the sky. This provides an extremely realistic representation of the time of day selected for the activity. For example, 12 noon in the winter will have a lower sun position in the northern hemisphere than 12 noon in the summer. Open Rails game environment will accurately represent these differences.

Once the activity is started, Open Rails software allows the player to advance or reverse the environment “time of day” independently of the movement of trains. Thus, the player train may sit stationary while the time of day is moved ahead or backward. The keys to command this depend from the national settings of the keyboard, and can be derived from the key assignment list obtained pressing F1.

In addition, Open Rails offers similar functionality to the time acceleration switch for MSTS. Use Alt + Page Up or Alt + Page Down keys to change the speed of the game clock.

In a multiplayer session, all clients time, weather and season selections are overridden by those set by the server.

7.10.2. Weather

When in activity mode Open Rails software determines the type of weather to display from the Weather parameter in the MSTS Activity file. In the other modes the weather can be selected in the start menu.

7.10.3. Modifying weather at runtime

Following commands are available at runtime (keys not shown here can be derived from the key assignment list obtained pressing F1):

- Overcast increase/decrease: increases and decreases the amount of clouds
- fog increase/decrease
- precipitation increase/decrease.

This demonstrates Open Rails software's foundation for dynamic weather effects in the game. Moreover pressing Alt-P weather can change from clear to raining to snowing and back to clear.

7.10.4. Seasons

In activity mode Open Rails software determines the season, and its related alternative textures to display from the Season parameter in the MSTS Activity file. In other modes the player can select the season in the start menu.

7.11. Screenshot - Print Screen

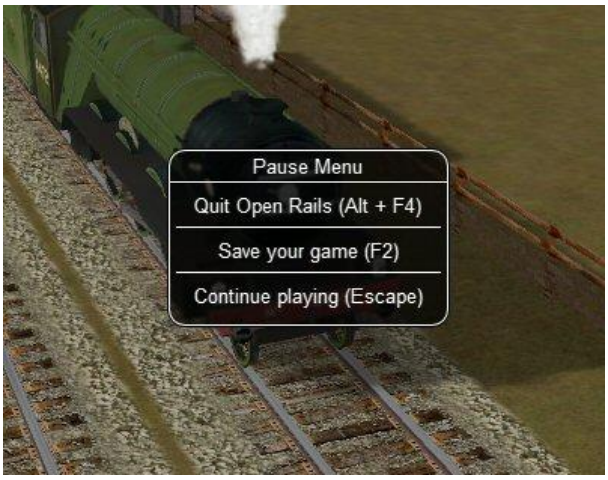
Press the Print Screen to capture an image of the game window. This will be saved, by default, in the file C:\Users\<username>\Pictures\Open Rails\Open Rails <date and time>.png[2]

Although the image is taken immediately, there may be a short pause before the confirmation appears. If you hold down the Print Screen key, then OR takes multiple images as fast as it can.

The key to capture the current window - Alt+Print Screen - is not intercepted by OR.

7.12. Suspending or exiting the game

You can suspend or exit the game by pressing the ESC key at any moment. Following window will appear.



The window is self-explanatory.

If you are running OR in a Window, you can also exit OR simply clicking on the x on the left top of the OR window.

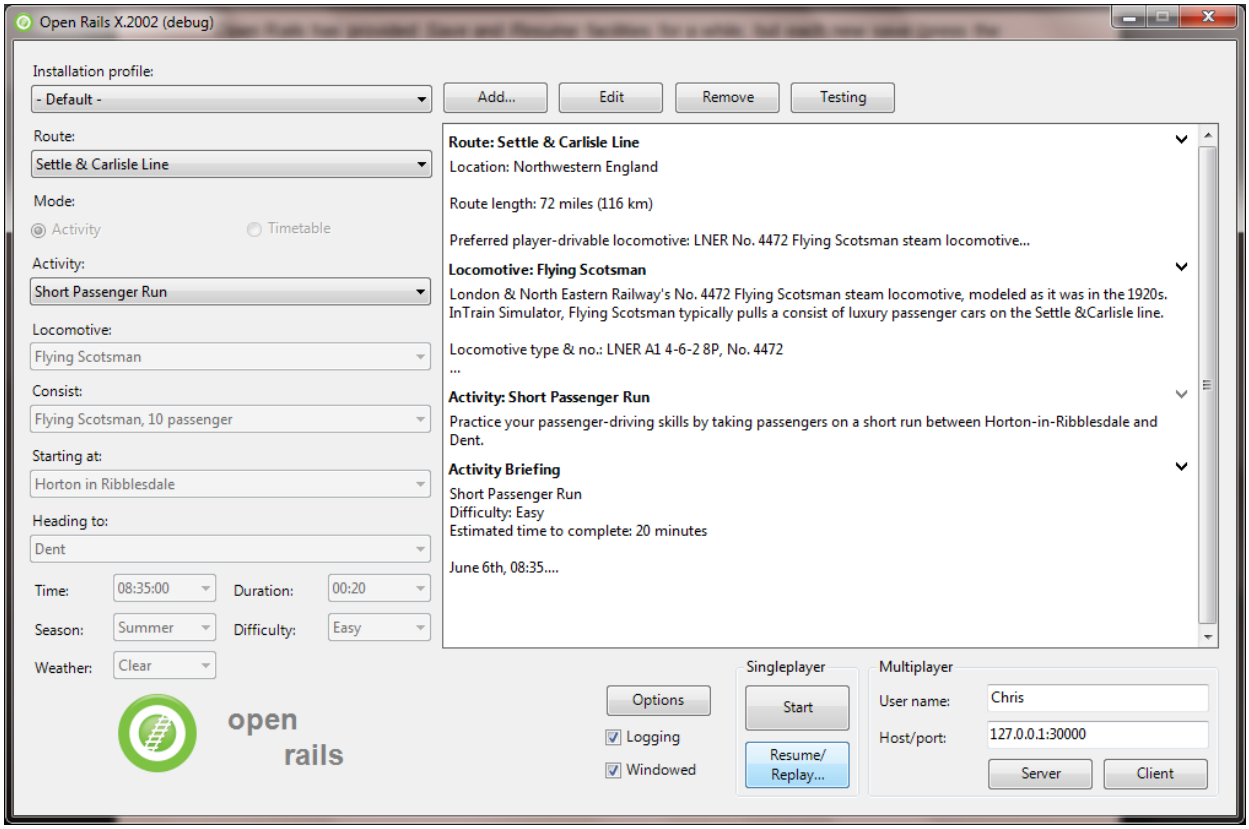
7.13. Save and Resume

Open Rails provides *Save* and *Resume* facilities and keeps every save until you choose to delete it.

During the game you can at any time save your session by pressing F2.

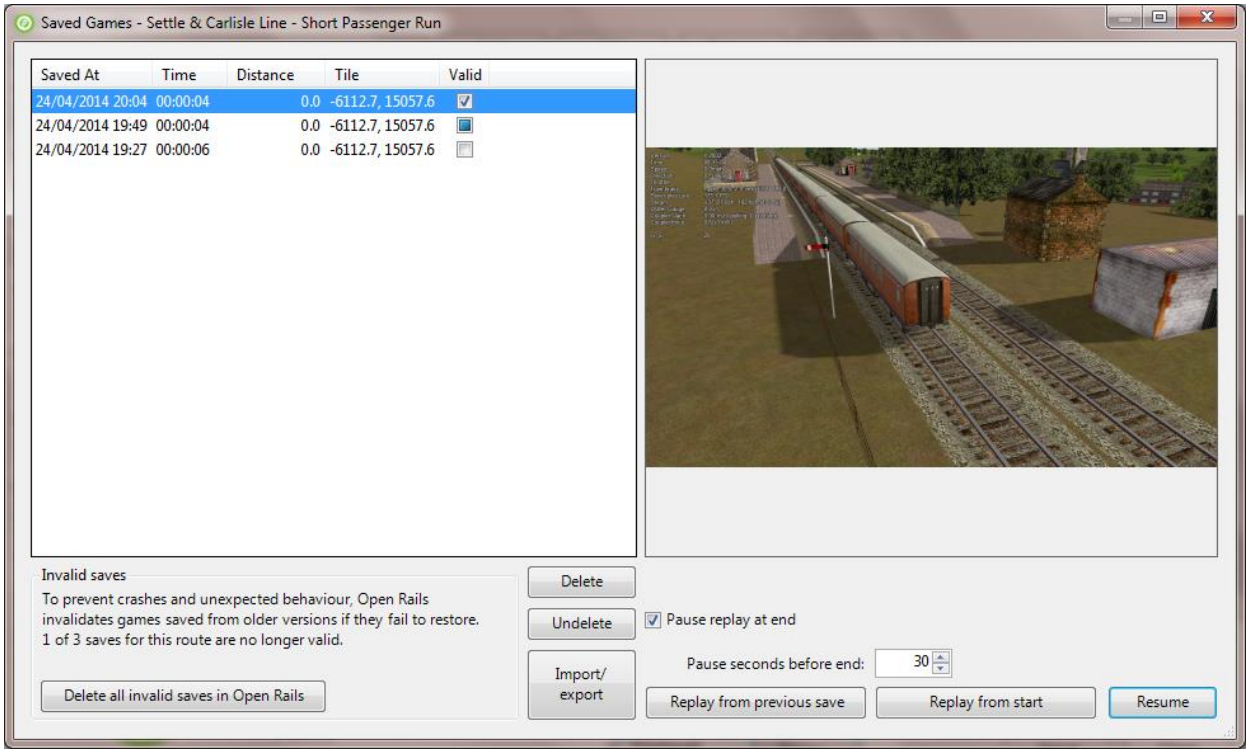
You can view them by choosing an activity and then pressing the *Resume/Replay...* button.

[2] Windows also knows the Pictures folder by the name My Pictures



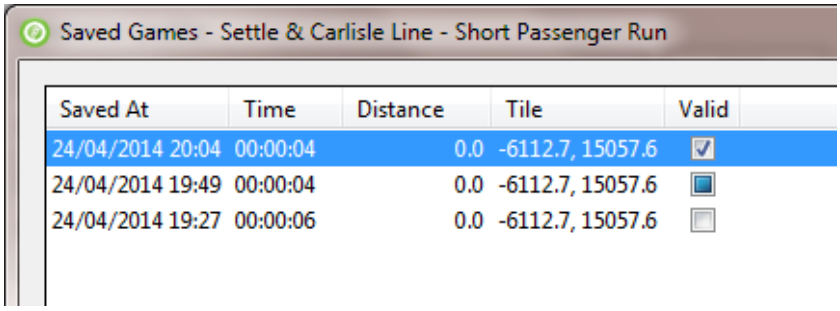
This leads to a list of any Saves you made for this activity.

To help you identify a Save, the list provides a screenshot and date and also the time and distance travelled in metres and the position of the player's train.

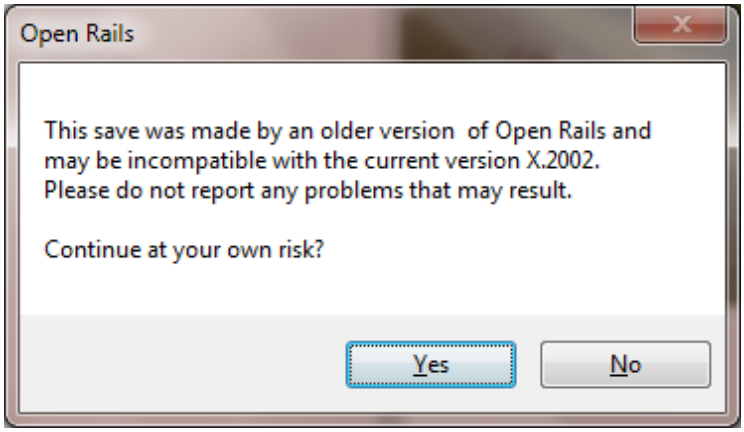


7.13.1. Caution

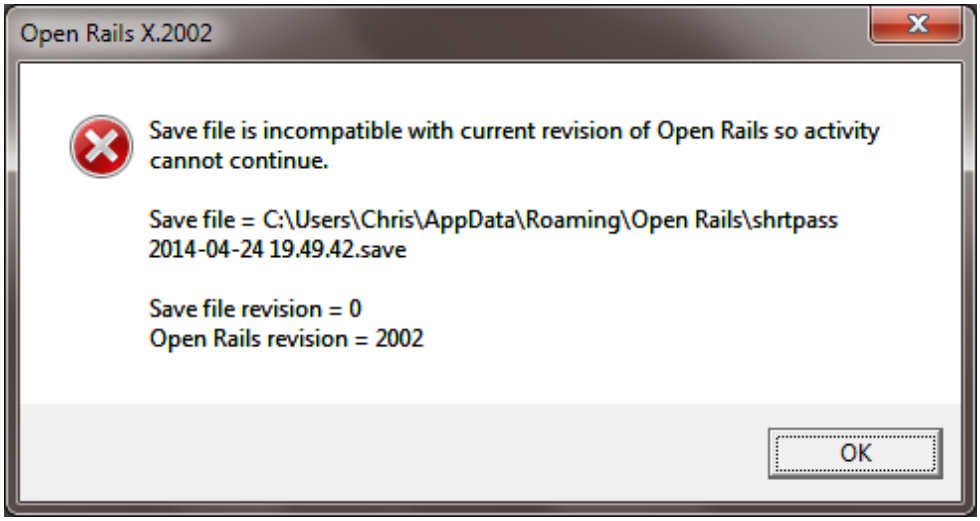
You should be aware that these Saves will only be useful in the short term as each new version of Open Rails will mark Saves from previous versions as potentially invalid (e.g. the second entry in the list below).



When you resume from such a Save, there will be a warning prompt.



The Save will be tested during the loading process. If a problem is detected, then you will be notified.



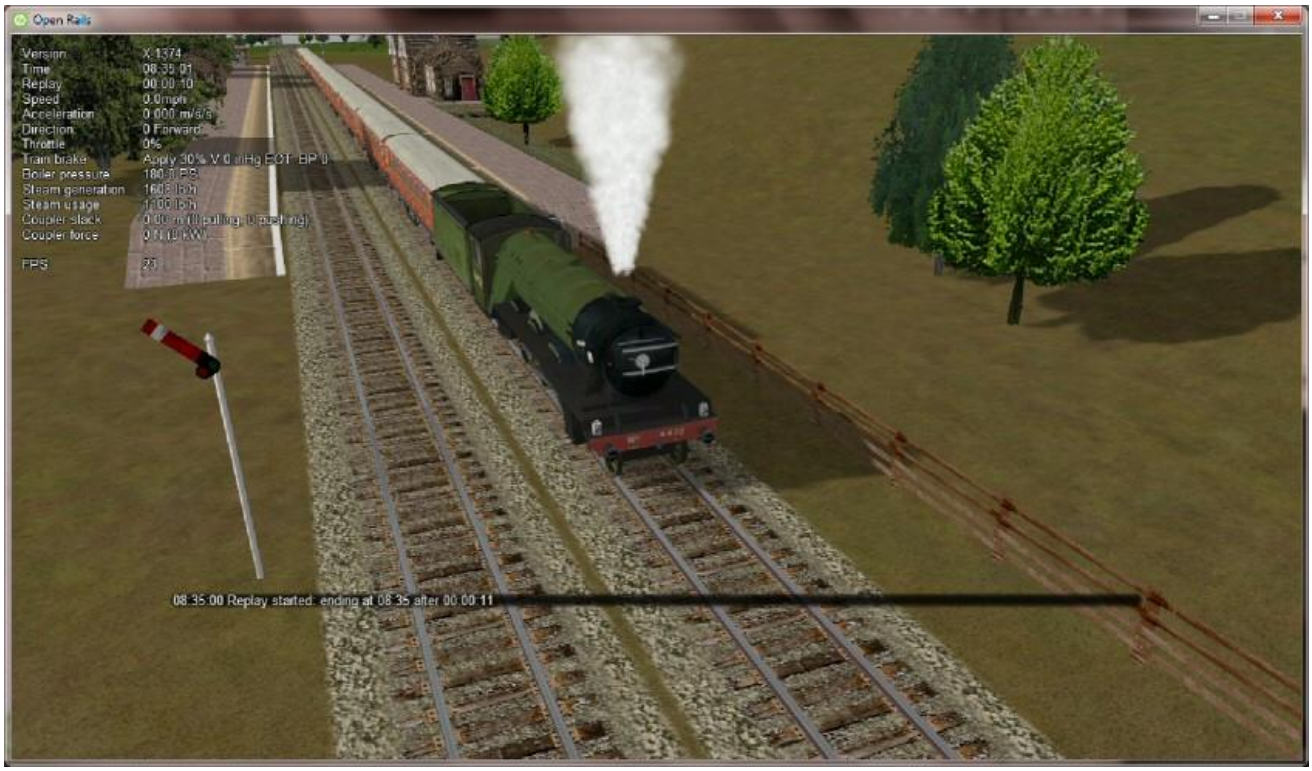
This Save and any Saves of the same age or older will be of no further value and will be marked as invalid automatically (e.g. the 3rd entry in the list). The button in the bottom left corner deletes all the invalid Saves for all activities in Open Rails.

7.14. Save and Replay

As well as resuming from a Save, you can also replay it just like a video. All the adjustments you made to the controls (e.g. opening the throttle) are repeated at the right moment to re-create the activity. As well as train controls, changes to the cameras are also repeated.

Just like a "black box flight recorder" Open Rails is permanently in recording mode, so you can save a recording at any time just by pressing *F2 Save.*

You choose the replay option using *Menu* > *Resume* > *Replay from start*

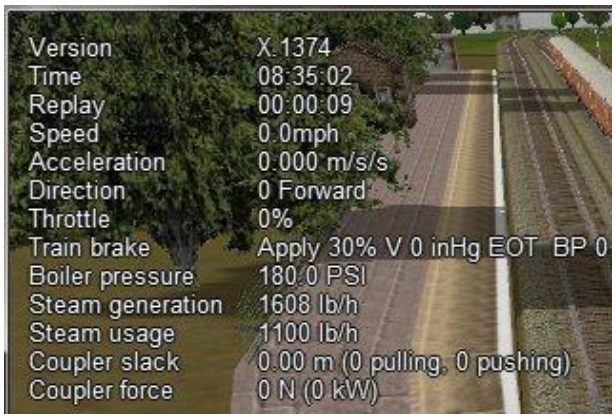


A second option *Menu* > *Resume* > *Replay from previous save* lets you play back a shortened recording. It resumes from the most recent Save it can find and replays from that point onwards. You might use it to play back a 5 minute segment which starts an hour into an activity.

A warning is given when the replay starts and a replay countdown appears in the *F5 Head Up Display*.

Warning

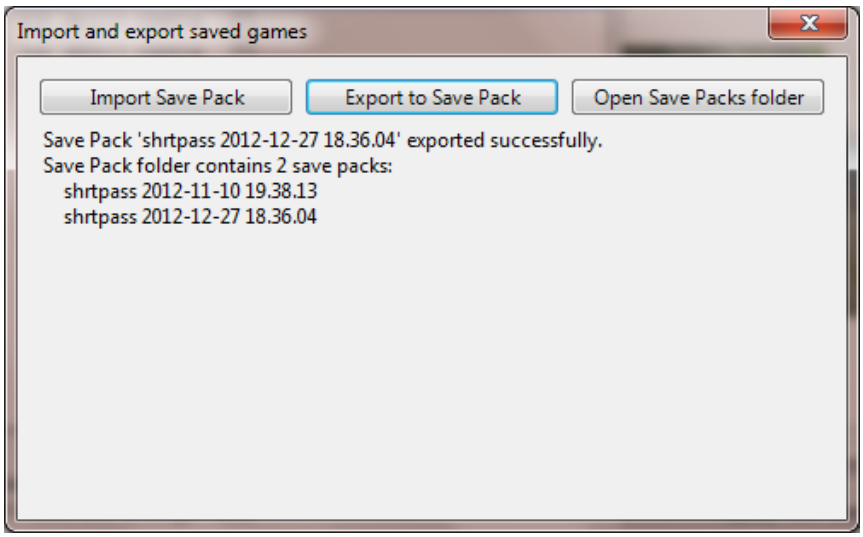
Countdown



By default, the simulation pauses when the replay is exhausted. Use *Pause replay at end* on the Saved Games window to change this.

Little can usefully be achieved by adjusting the train controls during replay, but the camera controls can be adjusted without limit. If changes are made (e.g. switching to a different camera view or zooming out), then replay of the camera controls is suspended while replay of the train controls continues. The results is a bit like editing a video. To resume the replay of the camera controls, just press *Esc* to open the *Pause Menu* and then choose *Continue playing.*

A possible development may be to edit the replay file to adjust times or to add messages to provide a commentary. This would allow you to build demonstrations and tutorials.



Replay is a feature which is unique to Open Rails. You can use it to make your own recordings and Open Rails provides a way to exchange them with other players.

To export a Save file, use *Menu > Options > Resume > Import/export saves > Export to Save Pack*

OR will pack the necessary files into a single archive file with the extension "ORSavePack" in the folder *Open Rails\Save Packs*

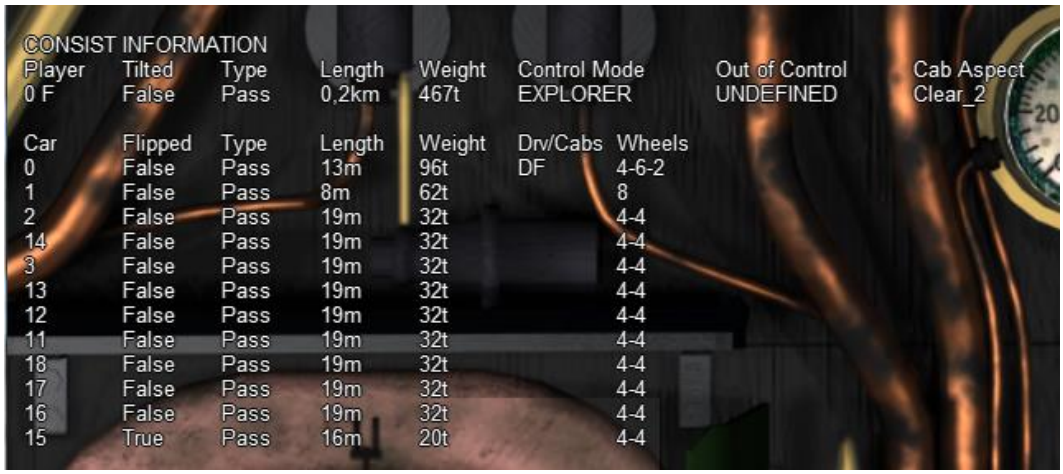
This ORSavePack file is a zip archive which contains the replay commands, a screenshot at the moment of saving, a Save file (so that Open Rails can offer its *Resume* option) and a log file. This arrangement means that the ORSavePack archive is ideal for attaching to a bug report.

You can use the *Import Save Pack* button on the same window to import and unpack a set of files from an ORSavePack archive. They will then appear in your *Saved Games* window.

7.15. Analysis tools

The extended HUDs provide a rich amount of info for analysis, evaluation and troubleshooting.

You move from one HUD to the other by repeatedly pressing Shift-F5.



7.15.1. Extended HUD for Consist Information

This page shows in the first line data about the whole train. Under “Player” you find the train number as assigned by OR followed by an “F” if the forward cab is selected, and an “R” if the rear cab is selected.

“Tilted” is true in case the consist name ends with “tilted” (e.g. ETR460_tilted.con), in which case it means that it is a tilting train.

“Control mode” shows the actual control mode. Read more on this here.

Cab aspect shows the aspect of next signal.

In the other line data about the train cars are shown. Data are mostly self-explanatory. Under Drv/Cabs a D appears if the car is driveable, and an F and/or a R appear if the car has a front and/or a rear cab.

7.15.2. Extended HUD for Locomotive Information

The next extended HUD display shows Locomotive information.

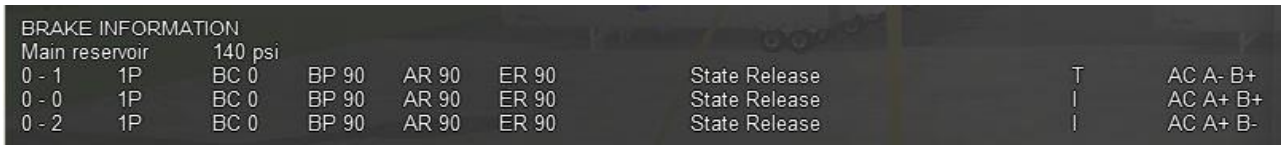
As can be seen from this screenshot related to a fictitious train with a diesel, an electric and a steam loco, info about diesel and electric locos is contained on a single line, while info about steam locos includes a large set of parameters, which shows the sophistication of OR's steam physics.

In the bottom part of this HUD two moving graphs show the evolution in time of the throttle value and of the power of the player locomotive (the one where the active cab resides).

7.15.3. Extended HUD for Brake Information

This extended HUD display includes all information in the basic HUD plus Brake status information. Info is shown for all cars. The first number shows the car UiD in the train, as found in the consist file or the activity file; the following alphanumeric string shows the brake

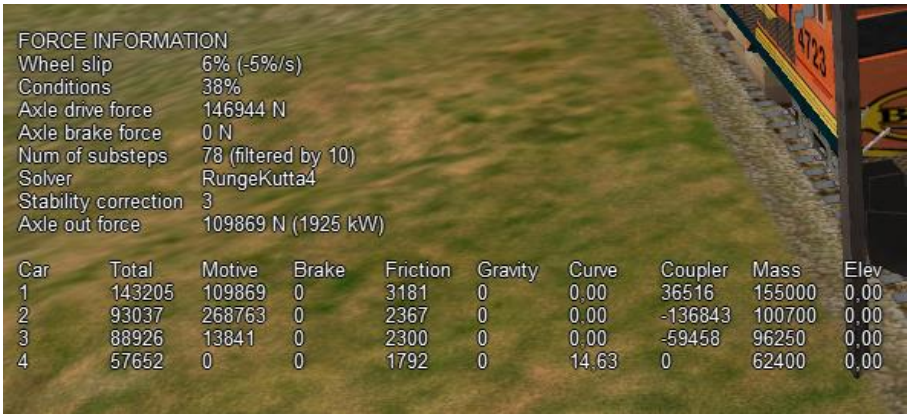
system (1P: single-pipe system, V: vacuum etc.) and the current state of the air brakes on the unit. More information on this display can be found in 8.5 Open Rails Braking.



7.15.4. Extended HUD for Train Force Information

In the first part of this display some info related to the player locomotive is shown. The info format differs if advanced adhesion has been selected or not (option 6.4.1).

The table part shows total force for up to ten locos/cars in the train. The first number shows the position of the car in the train. The second number is the total force acting on the car. This is the sum of the other forces after the signs are properly adjusted. The next number is the motive force which should only be non-zero for locomotives, and that becomes negative during dynamic braking. Next number is the brake force. Follows the friction force calculated from the Davis equation. The following value is the force due to gravity. Next value is the friction force due to the car being in a curve. The next value is the coupler force between this car and the next (negative is pull and positive is push). The mass in kg and the track elevation in % under the car follow. All the force values are in Newtons. Many of these values are relative to the orientation of the car, but some are relative to the train. If applicable, two further fields appear; the first is is "True" if the car is flipped with respect to the train and otherwise it's "False", while the second



signals coupler overload.

At the bottom of the picture two moving graphs are displayed.

The upper graph displays the motive force in % of the player locomotive. Green colour means tractive force, red colour means dynamic brake force.

The lower graph refers – roughly speaking - to the level of refinement used to compute

axle force.

7.15.5. Extended HUD for Dispatcher Information

The next extended HUD displays Dispatcher Information. It is very useful to troubleshoot activites or timetables. The player train and any AI trains will show in the Dispatcher Information, a line for each train.

A detailed explanation of the various columns follows.

Train : Internal train number, with P=Passenger and F=Freight.

- Travelled : distance travelled.
  Gives a indication if all is well. If a train started an hour ago and 'travelled' is still 0.0, something's clearly wrong.
- Speed : present speed.
- Max : max. allowed speed.
- AI Mode : gives an indication of what the AI train is 'doing'.
  Possible states :
  - INI : train is initializing. Normally you would not see this.
  - STP : train is stopped other than in a station. The reason for the stop is shown in "Authority".
  - BRK : train is preparing to stop. Does not mean it is actually braking, but it 'knows' it has to stop, or at least reduce speed, soon.
    Reason, and distance to the related position, are shown in "Authority" and "Distance".
  - ACC : train is accelerating, either away from a stop or because of a raise in allowed speed.
  - RUN : train is running at allowed speed.
  - FOL : train is following another train in the same signal section.
    Its speed is now derived from the speed of the train ahead.
  - STA : train is stopped in station.
  - WTP : train is stopped at waiting point.
  - EOP : train is approaching end of path.
  - STC: train is Static train, or train is in Inactive mode if waiting for next action.

- AI data : shows throttle (first three digits) and brake (last three digits) positions when AI train is running, but shows departure time (booked) when train is stopped at station or waiting point, or shows activation time when train is in inactive mode (state STC).
- .
- Mode :
    - SIGN (signal)
    - NODE
    - MAN: train is in manual mode (only player train, see here)
    - OOC: train is out of control
    - EXPL: train is in explorer mode (only player train)

      When relevant, this field also shows delay (in minutes), e.g. S+05 mean Signal mode, 5 mins. delay.
- Auth : End of "authorization" info - that is, the reason why the train is preparing to stop or slow down.
  Possible reasons are :
    - SPDL : speed limit imposed by speedsign.
    - SIGL : speed limit imposed by signal.
    - STOP : signal set at state "STOP".
    - REST : signal set at state "RESTRICTED" (train is to reduce speed at approaching this signal).
    - EOA : end of authority - generally only occurs in non-signalled routes or area, where authority is based on NODE mode and not SIGNAL mode.
    - STAT : station.
    - TRAH : train ahead.
    - EOR : end of train's route, or subroute in case the train approaches a reversal point.

  When the control mode is “NODE” the column Auth can show following strings:
    - EOT: end of track
    - EOP: end of path
    - RSW: switch reserved by another train
    - LP: train is in loop
    - TAH: train ahead
    - MXD: free run for at least 5000 meters
    - NOP: no path reserved.

  When the control mode is “OOC” the column Auth can show following strings:
    - SPAD: passed signal at danger
    - RSPD: passed signal at danger running backwards
    - OOAU: passed authority limit

    - OOPA: out of path
    - SLPP: slipped into path
    - SLPT: slipped to end of track
    - OOTR: out of track
    - MASW: misaligned switch.

- Distance : distance to the authority location.
- Signal : aspect of next signal (if any).
- Distance : distance to this signal.
  Note that if signal state is STOP, and it is the next authority limit, there is a difference of about 30m between authority and signal distance. This is the 'safety margin' that AI trains keep to avoid accidentally passing a signal at danger.
- Consist : the first part of the train's service name. Only for the player, always the “PLAYER” string is displayed.
- Path : the state of the train's path.
  The figure left of the "=" sign is the train's present subpath counter : a train's path is split into subpaths when its path contains reversal points.
  The details between { and } are the actual subpath.
  Following the final } can be x<N>, this indicates that at the end of this subpath the train will move on to the subpath number N.

  Path details :
  The path shows all track circuit sections which build this train's path. Track circuit sections are bounded by nodes, signals or cross-overs, or end-of-track.
  Each section is indicated by its type :
    - - is plain train section.
    - > is switch (no distinction is made for facing or trailing switch).
    - + is crossover.
    - [is end-of-track.

  Following each section is the section state. Numbers in this state refer to the train numbers as shown at the start of each row.
  Below, <n> indicates a such a number.
    - <n> section is occupied by train <n>.
    - (<n>) section is reserved for train <n>.
    - # (either with <n> or on its own) section is claimed by a train which is waiting for a signal.
    - & (always in combination with <n>) section is occupied by more than one train.
    - deadlock info (always linked to a switch node) :
        - * possible deadlock location - start of a single track section shared with a train running in opposite direction.

- ^ active deadlock - train from opposite direction is occupying or has reserved at least part of the common single track section.
  Train will be stopped at this location - generally at the last signal ahead of this node.
- ~ active deadlock at that location for other train - can be significant as this other train can block this train's path.

The dispatcher works by reserving track vector nodes for each train. An AI train will be allowed to move (or start) only if all of the nodes up to the next potential passing location are not reserved for another train. If this condition cannot be met, the AI train will not spawn.

There are other reasons that an AI train might not appear. The current dispatcher assumes that all routes are unsignaled. The dispatcher issues a track authority (which is similar to a track warrant) to all trains. For an AI train to start the tracks it needs must not already be reserved for another train. The dispatcher compares the paths of the trains to identify possible passing points and then reserved tracks for a train up until a passing point. When a train gets near the next passing point the reservation is extended to the next one. The end result is that an AI train cannot be placed on a track if that section of track is already occupied by or reserved for another train. A section of track is any track bounded by either a switch or a signal. Exception to this rule can be forced in activity mode by selecting the user option "Enhanced MSTS Compatibility".

7.15.6. Extended HUD for Debug Information

The last extended HUD display replaces the Dispatcher information with Debug information.
The first line (logging enabled) refers to logging as described in paragraphs 6.6 and 6.7.

Wide varieties of parameters are shown from frame wait and render speeds in milliseconds, to number of primitives, Process Thread resource utilization and number of Logical CPUs from the system’s bios. They are very useful in case of OR stuttering, to find out where the bottleneck is.

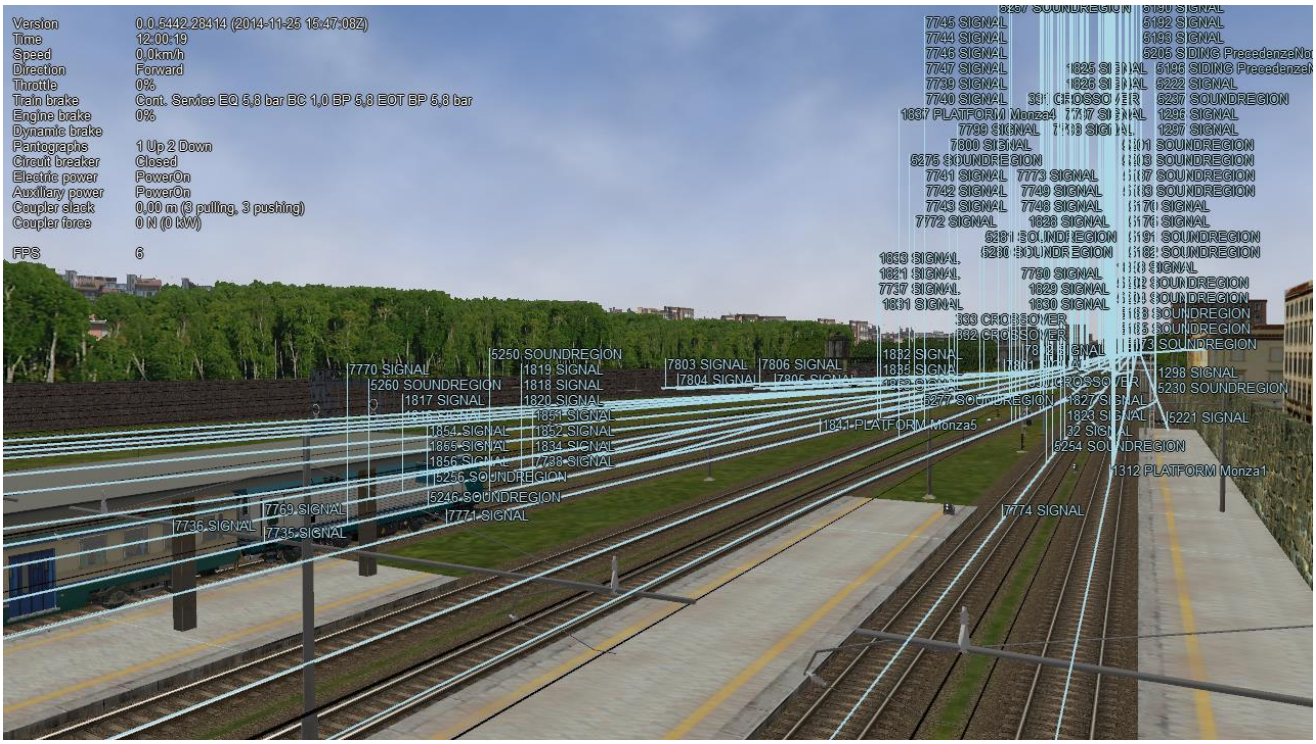
The values in the “Camera” line refer to the two tile coordinates and to the three coordinates within the tile.

At the bottom of the picture, some moving graphs are displayed that show the actual load of the computer.

Referring to memory occupation, about at least 400 MB must remain free to avoid out-of-memory exceptions.

7.15.7. Viewing interactive track items

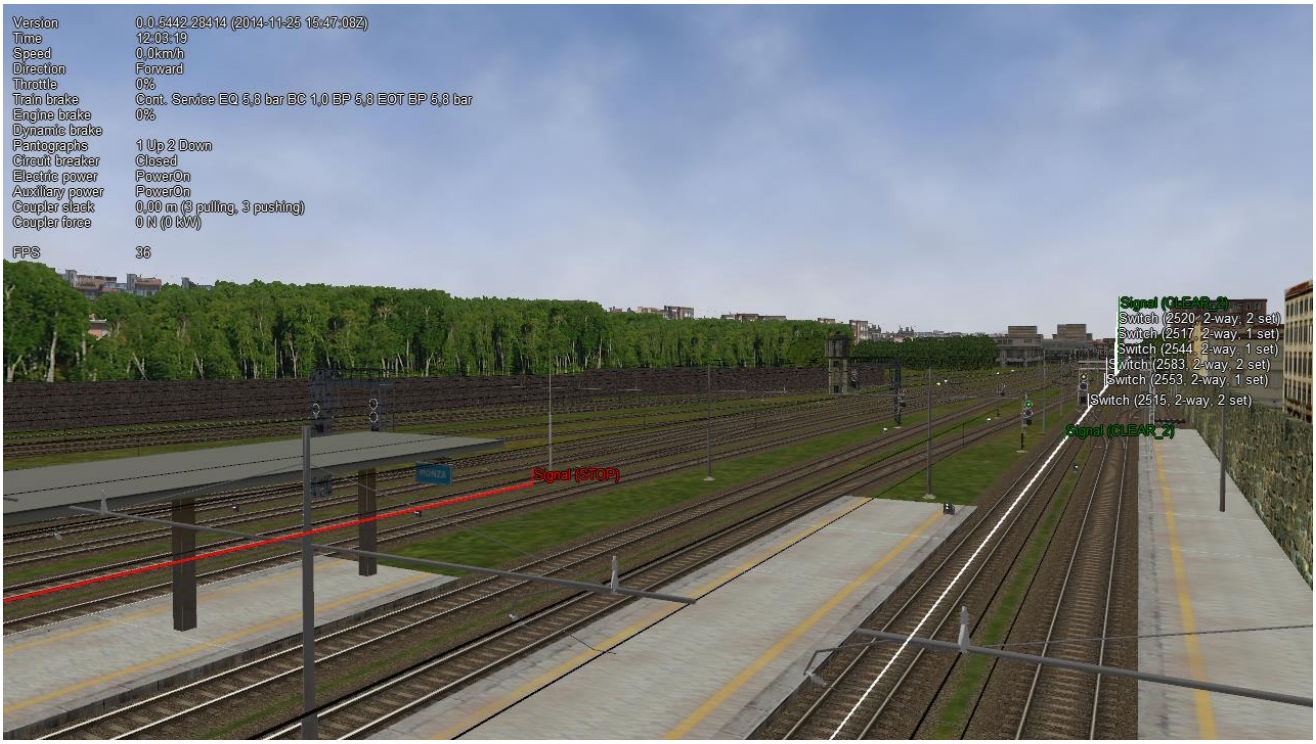
By pressing Ctrl-Alt-F6 at runtime you get a picture like this one



that allows you to take note of the interactive IDs for debugging purposes.

7.15.8. Viewing signal state and switches

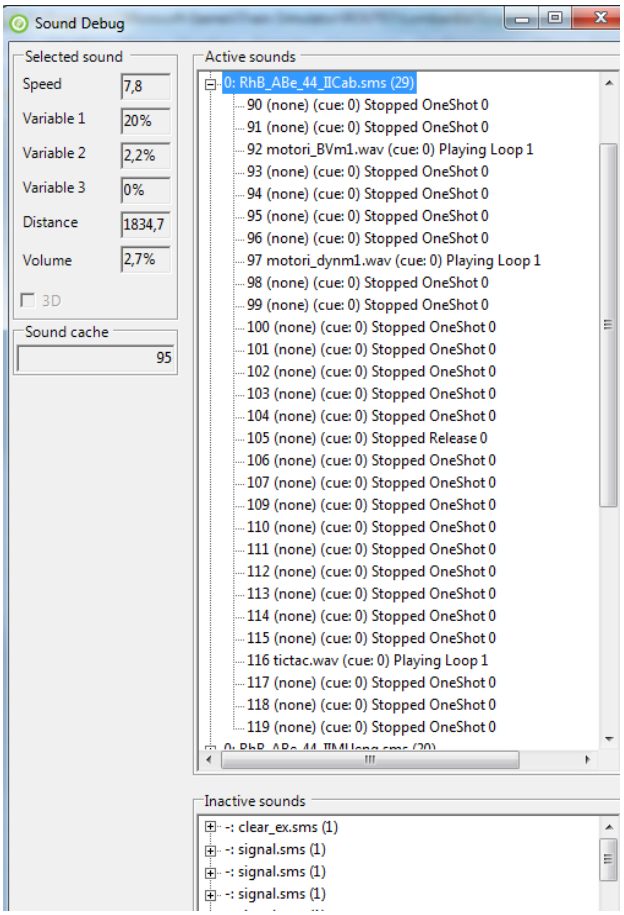
By pressing Ctrl-Alt-F11 you get a picture like the following one



that shows you the state of the signal and the switches on the path.

7.15.9. Sound debug window

By pressing Alt-S following additional window opens:



It shows on the above part the list of all active .sms files; by entering into detail of a specific .sms file, the list of all sound streams is displayed, as well as their state. On the left the value of the analogic sound variables is displayed for the selected .sms file. The volume refers to the first stream of the selected sound file.

Active and inactive sounds toggle passing from internal to external views and vice-versa.

7.16. OpenRailsLog.txt logfile

When in the main window the “Logging” option is checked, a logfile named OpenRailsLog.txt file is generated. This file contains rich information about the execution of the game session, allowing to identify critical problems. This file should always be attached to requests of support in case of problems.

Contents of the file are often self-explanatory, and therefore can be evaluated by the same contents developer.

7.17. Code-embedded logging options

OR source code is freely downloadable; check the www.OpenRails.org website for this. Within the code there are some debug options that, when activated, generate specific extended log files, e.g. for analysis of signal and of AI train behavior. Short specific info on this can be provided to people with programming skills.

7.18. Autopilot mode

Autopilot mode is a powerful tool to test activities.

8. Open Rails Physics

Open Rails physics is in an advanced stage of development. The physics structure is divided into logical classes; more generic classes are parent classes, more specialized classes inherit properties and methods of their parent class. Therefore, description for train cars physics is valid also for locomotives (because a locomotive is a special case of train car). All parameters are defined within the WAG or ENG file. The definition is based on MSTS file format and some additional ORTS based parameters. To avoid possible conflicts in MSTS, *ORTS* label is added to every OpenRails specific parameter (such as ORTSMaxTractiveForceCurves).

The WAG or ENG file may be placed as in MSTS in TRAINSET\TRAINS*TrainCar*\ folder (where *TrainCar* is the name of the train car folder). If OR-specific parameters are used, or if different WAG or ENG files are used for MSTS and OR, the preferred solution is to place the OR-specific WAG or ENG file in folder TRAINSET\TRAINS*TrainCar*\OpenRails (see here for this).

8.1. Train Cars (WAG, or "Wagon" part of ENG file)

The behavior of a train car is mainly defined by a resistance / resistive force (a force needed to pull a car) defines behavior of train cars. Train car physics also includes coupler slack and braking. In the description below, the Wagon section of the WAG / ENG file is discussed.

8.1.1. Resistive forces

Open Rails physics calculates resistance based on real world physics: gravity, mass, rolling resistance and optionally curve resistance. This is calculated individually for each car in the train. The program calculates rolling resistance, or friction, based on the Friction parameters in the Wagon section of WAG/ENG file. Open Rails identifies whether the WAG file used the FCalc utility or other friction data. If FCalc was used to determine the Friction variables within the .wag file, Open Rails compares that data to the Open Rails Davis equations to identify the closest match with the Open Rails Davis equation. If no-FCalc Friction parameters are used in the WAG file, Open Rails ignores those values, substituting its actual Davis equation values for the train car.

Basic (simplified) Davis formula is used in the following form:

Where *res_force* is the friction force of the car. The rolling resistance can be defined either by FCalc or ORTSDavis_A, _B and _C components. If one of the ORTSDavis components is zero, FCalc is used. Therefore, if the data doesn't contain e.g. the B part of the Davis formula, a very small number should be used instead of zero.

When a car is pulled from steady state, an additional force is needed due to higher bearing forces. The situation is simplified by using a different calculation at low speed (5 mph and lower). Empiric static friction forces are used for different classes of mass (under 10 tons, 10 to 100 tons and above 100 tons). In addition, if weather conditions are poor (snowing is set), the static friction is increased.

When running on a curve and if the “Curve Resistance Speed Dependent” option is enabled, additional resistance is calculated, based on the curve radius, rigid wheel base, track gauge and superelevation. The curve resistance has its lowest value at the curve's optimal speed. Running at higher or lower speed causes higher curve resistance. The worst situation is starting a train from zero speed. The track gauge value can be set by ORTSTrackGauge parameter, otherwise 1435 mm is used. The rigid wheel base can be also set by ORTSRigidWheelBase, otherwise the value is estimated. Further details are discussed later.

When running on a slope (uphill or downhill), additional resistance is calculated based on the car mass taking into account the elevation of the car itself. Interaction with the “car vibration feature” is a known issue (if the car vibrates the resistance value oscillate).

8.1.2. Coupler slack

Slack action for couplers is introduced and calculated the same way as in MSTS.

8.1.3. Adhesion of locomotives – settings within the Wagon section of ENG files

MSTS calculates the adhesion parameters based on a very strange set of parameters filled with even stranger range of values. Since ORTS is not able to mimic the MSTS calculation, a standard way based on the adhesion theory is used with some known issues in use with MSTS content.

MSTS “Adheasion” parameters are not used in ORTS. A new set of parameters can be used instead:

ORTSAdhesion(ORTSCurtius_Kniffler (A B C D))

The A, B and C values are coefficients of a standard form of various empirical formulas, e.g. Curtius-Kniffler or Kother. The D parameter is used in the advanced adhesion model described later.

From A, B and C a coefficient CK is computed, and the adhesion force limit is then calculated by multiplication of CK with the car mass and the gravity acceleration (9.81), as better explained later in the paragraph.

The adhesion limit is considered in the adhesion model of locomotives only.

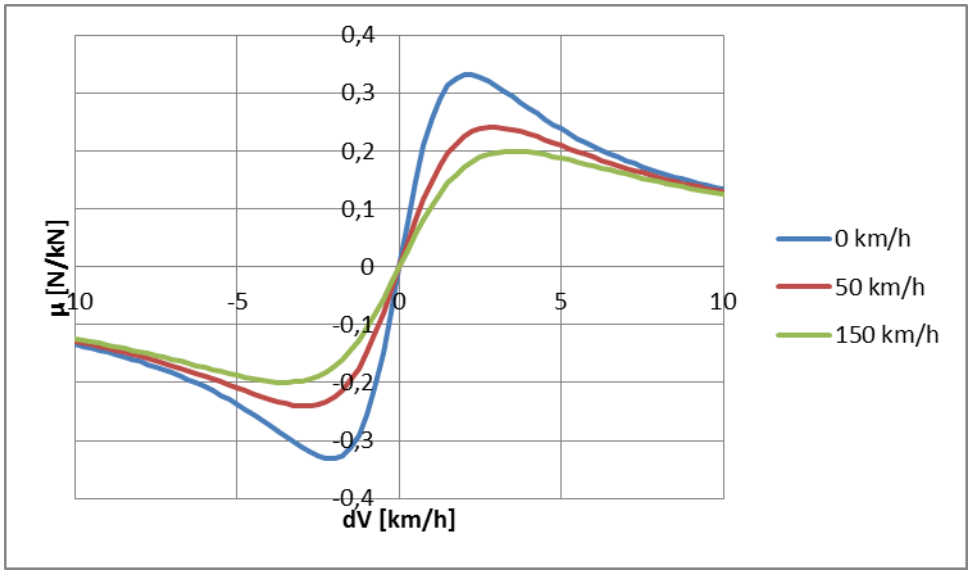
The adhesion model is calculated in two possible ways. The first one – simple adhesion model – is based on a very simple threshold condition and works similar to the MSTS adhesion model. The second one – advanced adhesion model – is a dynamic model simulating the real world conditions on a wheel-to-rail contact and will be described later. The advanced adhesion model uses some additional parameters such as:

- ORTSAdhesion(ORTSSlipWarningThreshold (T)) , where T is the wheelslip percentage considered as a warning value to be displayed to the driver.
- ORTSAdhesion(Wheelset (Axle (ORTSInertia (Inertia)))) , where Inertia is the model inertia in $kg.m^2$ and can be set to adjust the advanced adhesion model dynamics. The value considers the inertia of all the axles and traction drives. If not set, the value is estimated from the locomotive mass and maximal power.

The first model -simple adhesion model - is a simple tractive force condition-based computation. If the tractive force reaches its actual maximum, the wheel slip is indicated in HUD view and the tractive force falls to 10% of the previous value. By setting the throttle down adherence is

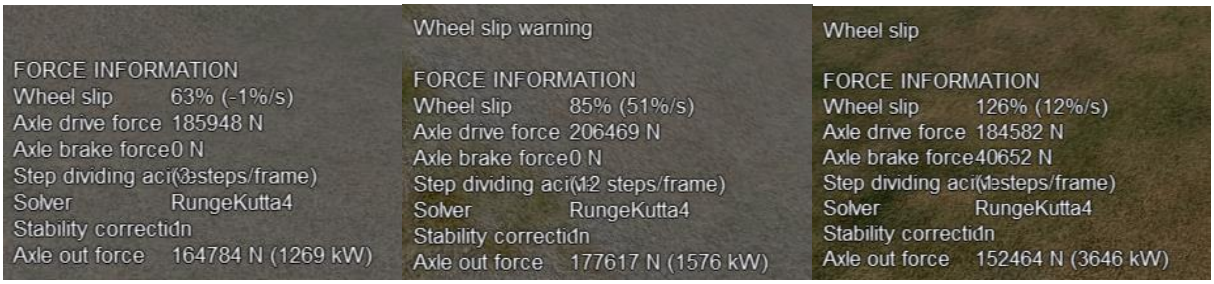
regained. This is called the simple adhesion model.

The second adhesion model (advanced adhesion model) is based on a simplified dynamic adhesion theory. Very briefly, there is always some speed difference between the wheel speed of the locomotive and the longitudinal train speed when the tractive force is different from zero. This difference is called “wheel slip / wheel creep”. The adhesion status is indicated in the HUD “Force Information” view by the “Wheel Slip” parameter and as a warning in the general area of the HUD view. For simplicity, only one axle model is computed (and animated). A tilting feature and the independent axle adhesion model will be introduced in the future.
The heart of the model is the slip characteristics (picture below).



The “wheel creep” describes the stable area of the characteristics and is used in the most of the operation time. When the tractive force reaches the actual maximum of the slip characteristics, force transition falls down and more power is used to speed up the wheels, so called “wheel slip”.

To avoid the loss of the tractive force, use the throttle in combination with sanding for returning to the stable area (wheel creep area). Possible sequence of the wheel slip development is shown on the pictures below. The “Wheel slip” value is displayed as a value relative to the best adhesion conditions for actual speed and weather. The value of 63% means very good force transition. For values higher than “Wagon (ORTSadhesion (ORTSSlipWarningThreshold))” or 70% by default, the “Wheel slip warning” is displayed, but the force transition is still very good. This indication should warn you to use the throttle very carefully. Exceeding 100%, the “Wheel slip” message is displayed and the wheels are starting to speed up, what can be seen on the speedometer or the external view 2. To reduce the wheel slip, use “throttle down”, sanding or a locomotive brake.



The “actual maximum” of the tractive force is based on the Curtius-Kniffler adhesion theory and can be adjusted by a following parameter of the ENG file: Wagon (OR_adhesion (Curtius_Kniffler (A B C D))), where A, B, C are coefficients of Curtius-Kniffler, Kother or similar formula. By default, Curtius-Kniffler is used.

This means that the maximum is related to the speed of the train, or to the weather conditions. “D” parameter is used in advanced adhesion model and should be always 0.7.

There are some additional parameters in the “Force Information” HUD view. The axle/wheel is driven by the “Axle drive force” and braked by the “Axle brake force”. The “Axle out force” is the output force of the adhesion model (used to pull the train). To compute the model correctly the FPS rate needs to be divided by “Solver dividing” value in a range from 1 to 50. By default, the Runge-Kutta4 solver is used to get the best results. When the “Solver dividing” value is higher than 40, the Euler-modified solver is used instead to reduce CPU load.

Anyway, in some cases when the CPU load is higher, the time step for the computation may become very high and the simulation may start to oscillate (the “Wheel slip” rate of change (in the brackets) becomes very high). There is a stability correction feature that modifies the dynamics of the adhesion characteristics. Higher instability can cause a huge wheel slip. You can use “DebugResetWheelSlip” (“Ctrl+X” by default) command to reset the adhesion model. If you experience such behavior for most of time, please use the basic adhesion model instead by pressing “DebugToggleAdvancedAdhesion” (“Ctrl+Alt+X” by default).
Another option is to use a Moving average filter available in the Simulation Options. The higher value, the more stable the simulation will be. However, the higher value the slower dynamic response. Recommended range is between 10 and 50.

To match some of the real world features, the “Wheel slip” event can cause automatic zero throttle setting. Use the “Engine (ORTS (ORTSWheelSlipCausesThrottleDown))” Boolean value of the ENG file.

8.2. Engine – Classes of Motive Power

Open Rails software provides for different classes of engines: diesel, electric, steam and default. If needed, additional classes can be created with unique performance characteristics.

8.2.1. Diesel locomotives in general

Diesel locomotive model in ORTS simulates the behavior of diesel engine driven locomotives of two basic types – diesel-electric and diesel-mechanic. The diesel engine model is the same for both types, but acts differently because of the different type of load. Basic controls (direction, throttle, dynamic brake, air brakes) are common across all classes of engines. Diesel engines can be started or stopped by pressing the START/STOP key (Y in English keyboards). The starting and stopping sequence is driven by a "starter" logic, which can be customized, or is estimated by the engine parameters.

8.2.1.1.Starting the diesel engine

To start the engine, simply press the START/STOP key once. The direction controller must be in the neutral position (otherwise, a message pops up). The engine RPM will increase according to its speed curve parameters (described later). When RPM reaches 90% of StartingRPM (67% of IdleRPM by default), the fuel starts to flow as well as the exhaust emission. RPM continues to increase up to StartingConfirmationRPM (110% of IdleRPM by default) and the demanded RPM is set to idle. The engine is started and ready to operate.

8.2.1.2.Stopping the diesel engine

To stop the engine, press the START/STOP key once. The direction controller must be in the neutral position (otherwise, a message pops up). The fuel flow is cut-off and the RPM will start to decrease according to its speed curve parameters. The engine is considered as fully stopped when RPM is zero. The engine can be restarted even when stopping (RPM is not zero).

8.2.1.3.Starting or stopping helpers' diesel engines

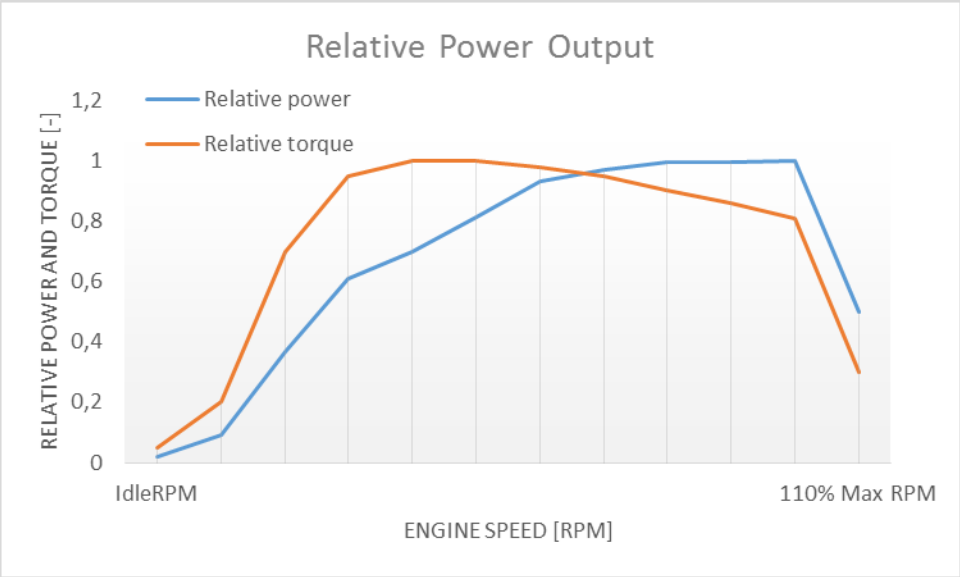
By pressing Diesel helper START / STOP key (Shift + Y in English keyboards), diesel engines of helper locomotives can be started or stopped. Consider disconnecting from multiple-unit (MU) signal instead of stopping the engine (see here, "Toggle MU connection").

It is also possible to operate with the own engine off and the helper's engine on.

8.2.1.4.ORTS specific diesel engine definition

If no ORTS specific definition is found, asingle diesel engine definition is created based on the MSTS settings. Since MSTS introduces a model without any data crosscheck, the behavior of MSTS and ORTS diesel locomotives can be very different. In MSTS, MaxPower is not considered the same way and you can get much "better" performance than expected. In ORTS, diesel engines cannot be overloaded.

No matter which definition of the engine is used, the diesel engine is defined by its load characteristics (maximal output power vs. speed) for optimal fuel flow and/or mechanical characteristics (output torque vs. speed) for maximal fuel flow. The model computes output power / torque according to these characteristics and the throttle settings. If the characteristics are not defined (as instead they are in the example below), they are built up based on the MSTS data and common normalized characteristics.



In many cases the throttle vs. speed curve is customized because power vs. speed is not linear. The default linear throttle vs. speed characteristics is built in to avoid engine overloading at lower throttle settings. Nevertheless, it is recommended to adjust the table below to get more realistic behavior.

In ORTS, single or multiple engines can be set for one locomotive. In case there is more than one engine, other engines act like helpers' engines (start / stop controls, shift+Y by default). The power of each active engine is added to the locomotive power. The number of diesel engines is not limited.

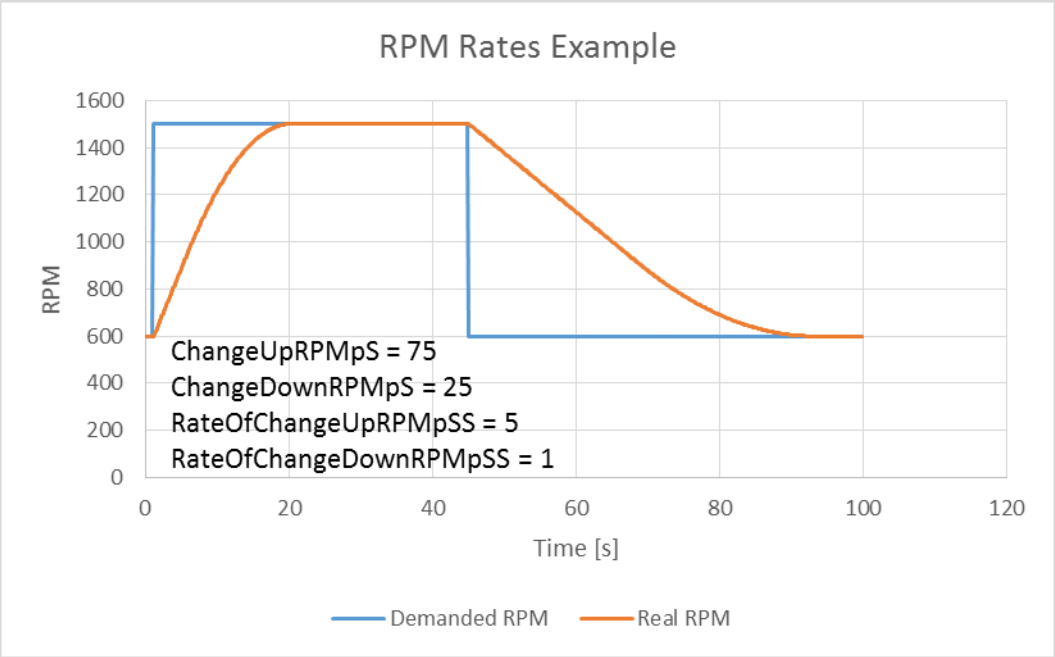
If ORTS specific definition is used, each parameter is tracked and if one is missing, it is estimated from the MSTS specific settings. In case DieselPowerTab and/or DieselTorqueTab are missing, the missing table is computed based on the existing one, or both are filled by default tables.

Engine(…	
ORTSDieselEngines (2	Num of engines
Diesel (	
IdleRPM (510)	Idle RPM
MaxRPM (1250)	Maximal RPM
StartingRPM (400)	Starting RPM
StartingConfirmRPM (570)	Starting confirmation RPM
ChangeUpRPMpS (50)	Increasing change rate RPM / s
ChangeDownRPMpS (20)	Decreasing change rate RPM / s
RateOfChangeUpRPMpSS (5)	Jerk of ChangeUpRPMpS RPM / s^2
RateOfChangeDownRPMpSS (5)	Jerk of ChangeDownRPMpS RPM / s^2
MaximalPower (300kW)	Maximal output power
IdleExhaust (5)	Num of exhaust particles at IdleRPM
MaxExhaust (50)	Num of exhaust particles at MaxRPM
ExhaustDynamics (10)	Exhaust particle multiplier at transient
ExhaustColor (00 fe fe fe)	Exhaust color at steady state
ExhaustTransientColor (00 00 00 00)	Exhaust color at speeding up
DieselPowerTab (	Diesel engine power tab
0 0	RPM Power in Watts
510 2000	

520 5000 600 2000 800 70000 1000 100000 1100 200000 1200 280000 1250 300000) DieselConsumptionTab (0 0 510 10 1250 245) ThrottleRPMTab (0 510 5 520 10 600 20 700 50 1000 75 1200 100 1250)) Diesel (///the same as above or different))	Diesel engine fuel consumption tab RPM Specific consumption g/kWh Diesel engine RPM vs. throttle tab Throttle % Demanded RPM

8.2.1.5.Diesel engine speed behavior

The engine speed is calculated based on the RPM change rates and its change rates. Usual setting and corresponding result is shown below. ChangeUpRPMpS means the slope of RPM, RateOfChangeUpRPMpSS means how fast RPM approaches demanded RPM.



8.2.1.6.Fuel consumption

Following the MSTS model, ORTS computes the diesel engine fuel consumption based on ENG file parameters. The fuel flow and level are indicated by the HUD view. Final fuel consumption is adjusted according to the current diesel power output (load).

8.2.1.7.Exhaust

The diesel engine exhaust feature can be modified as needed. The main idea of this feature is based on the general combustion engine exhaust. When operating in a steady state, the color of the exhaust is given by the new ENG parameter “engine(ORTS(Diesel(ExhaustColor)))”. The amount of particles emitted is given by a linear interpolation of “engine(ORTS(Diesel(IdleExhaust)))” and “engine(ORTS(Diesel(MaxExhaust)))” in the range from 1 to 50. In a transient state, the amount of the fuel increases but the combustion is not optimal. Thus, the amount of the particles is temporary higher, multiplied by engine(ORTS(Diesel(ExhaustDynamics))) and colored by engine(ORTS(Diesel(ExhaustTransientColor))). The format of the “color” value is (aarrggbb) - aa = power of light; rr = red color component; gg = green color component; bb = blue color component in HEX number format (00 to ff).

8.2.1.8.Cooling system

ORTS introduces a simple cooling and oil system within the diesel engine model. The engine temperature is based on the output power and the cooling system output. Maximum value of 100°C can be reached with no impact on performance. It is just an indicator, but the impact on the engine’s performance will be implemented later. The oil pressure feature is simplified and the value is proportional to the RPM. There will be further improvements of the system later on.

8.2.2. Diesel-electric locomotives

The Diesel-electric locomotives are driven by electric traction motors supplied by a diesel-generator set. The gen-set is the only power source available, thus the diesel engine power supplies also auxiliaries and other loads. Therefore, the output power should be always lower than the diesel engine rated power.

In ORTS, the diesel-electric locomotive can use ORTSTractionCharacteristics or ORTSMaxTractiveForceCurves tables to provide better performance in comparison with the real world. If no table is used, the tractive force is limited by MaxForce, MaxPower and MaxVelocity. The throttle setting is passed to the ThrottleRPMTab, where the RPM demand is selected. The output force increases with the Throttle setting, but the power follows maximal output power available (RPM dependent).

8.2.3. Diesel-hydraulic locomotives

Diesel-hydraulic locomotives are not implemented in ORTS. However, using ORTSTractionCharacteristics or ORTSMaxTractiveForceCurves tables, desired performance can be achieved, while no gearbox is in use and DieselEngineType is “electric”.

8.2.4. Diesel-mechanic locomotives

ORTS features a mechanical gearbox feature that mimics the MSTS behavior, including automatic or manual shifting. Some features not well described in MSTS are not yet implemented, such as GearBoxBackLoadForce, GearBoxCoastingForce and GearBoxEngineBraking.

Output performance is very different in comparison with MSTS. The output force is computed using the diesel engine torque characteristics to get results that are more precise.

8.3. Electric locomotives

At the present time, diesel and electric locomotive physics use the default engine physics. Default engine physics simply uses the MaxPower and MaxForce parameters to determine the pulling power of the engine modified by the Reverser and Throttle position. The locomotive physics can be replaced by traction characteristics (speed in mps vs. force in Newtons) as mentioned below.

Some OR-specific parameters are available in order to increase the realism of the electric system.

Since the simulator doesn’t know when the pantograph in the 3D model is up or down, you can set some additional parameters in order to add a delay between when the raise of the pantograph is commanded and when the pantograph is up.
In order to do that, you can write in the Wagon section of your .eng file or .wag file (since the pantograph can be on a wagon) this optional structure:

```
ORTSPantographs(
    Pantograph(                          << This is going to be your first pantograph.
        Delay( 5 s )                     << Example : a delay of 5 seconds
    )
    Pantograph(
        … parameters for the second pantograph …
    )
```

)

Other parameters will be added to this structure later such as power limitations or speed restrictions.

By default, the circuit breaker of the train closes as soon as power is available on the pantograph. In real life, the circuit breaker doesn't close instantly, so you can add a delay with the ORTSCircuitBreakerClosingDelay() optional parameter.

The power-on sequence time delay can be adjusted by the optional ORTSPowerOnDelay( ) value (for example : ORTSPowerOnDelay(5 s)) within the Engine section of the .eng file (in seconds). The same delay for auxiliary systems can be adjusted by the optional parameter ORTSAuxPowerOnDelay( ).

A scripting interface is available in order to create a customized circuit breaker or a customized power supply system (it will be useful later when the key bindings will be customizable for each locomotive).

The power status is indicated by "Electric power" value of the HUD view. The pantographs of all locomotives in a consist are triggered by "Control Pantograph First" and "Control Pantograph Second" command ("P" and "Shift+P" by default). The pantographs status is indicated by the "Pantographs" value of the HUD view.

The electric, diesel and default locomotive physics can be extended by the following Boolean parameters: ORTS(EmergencyCausesPowerDown; EmergencyCausesThrottleDown; EmergencyEngagesHorn) (are these operating ???). All these parameters are related to an emergency braking situation. These parameters are used to match some of the real world features of modern locomotives.

8.4. Steam locomotives

Steam engines employ a more developed physics model. Open Rails steam physics parses the eng file for the following parameters: NumCylinders; CylinderStroke; CylinderVolume; BoilerVolume; MaxBoilerPressure; MaxBoilerOutput; ExhaustLimit; and

BasicSteamUsage. These parameters are used as input values for Open Rails independently developed steam locomotive physics equations.

8.5. Engines – Multiple Units in Same Consist or AI Engines

In an OR player train one locomotive is controlled by the player, while the other ones are as default or controlled by the train's MU (multiple unit) signals for braking and throttle position, etc. The player-controlled loco generates the MU signals which pass along to every unit in the train. For AI trains, the AI software directly generates the MU signals - there is no player-controlled loco. In this way, all engines use the same physics code for power and friction.

This software model will ensure that non-player controlled engines will behave exactly the same way as player controlled ones.

8.6. Open Rails Braking

Open Rails software has implemented its own braking physics in the current release. It is based on the Westinghouse 26C and 26F air brake and controller system. Open Rails braking will parse the type of braking from the ENG file to determine if the braking physics uses passenger or freight standards, self-lapping or not. This is controlled within the Options menu as shown below.

Selecting Graduated Release Air Brakes in *Menu* > *Options* allows a partial release of the brakes. Some 26C brake valves have a cut-off valve that has three positions: passenger, freight and cut-out. Checked is equivalent to passenger standard and unchecked is equivalent to freight standard.

The *Graduated Release Air Brakes* option controls two different features. If the train brake controller has a self-lapping notch and the *Graduated Release Air Brakes* box is checked, then the amount of brake pressure can be adjusted up or down by changing the control in this notch. If the *Graduated Release Air Brakes* option is not checked, then the brakes can only be increased in this notch and one of the release positions is required to release the brakes.

Another capability controlled by the *Graduated Release Air Brakes* checkbox is the behavior of the brakes on each car in train. If the *Graduated Release Air Brakes* box is checked, then the brake cylinder pressure is regulated to keep it proportional to the difference between the emergency reservoir pressure and the brake pipe pressure. If the *Graduated Release Air Brakes* box is not checked and the brake pipe pressure rises above the auxiliary reservoir pressure, then the brake cylinder pressure is released completely at a rate determined by the retainer setting.

Following brake types are implemented in OR:

- Vacuum single-pipe
- Air single-pipe
- Air twin-pipe
- EP (Electro-pneumatic)
- Single-transfer-pipe (air and vacuum).

Here below in general the operation of air single-pipe brakes is described.

The auxiliary reservoir needs to be charged by the brake pipe and, depending on the WAG file parameters setting, this can delay the brake release. When the *Graduated Release Air Brakes* box is not checked, the auxiliary reservoir is also charged by the emergency reservoir (until both are equal and then both are charged from the pipe). When the *Graduated Release Air Brakes* box is checked, the auxiliary reservoir is only charged from the brake pipe. The Open Rails software implements it this way because the emergency reservoir is used as the reference pressure for regulating the brake cylinder pressure.

The end result is that you will get a slower release when the *Graduated Release Air Brakes* box is checked. This shouldn't be an issue with two pipe air brakes because the second pipe can be the source of air for charging the auxiliary reservoirs.

Open Rails software modeled most of this graduated release car brake behavior based on the 26F control valve, but this valve is designed for use on locomotives. That valve uses a control reservoir to maintain the reference pressure and Open Rails software simply replaced the control reservoir with the emergency reservoir.

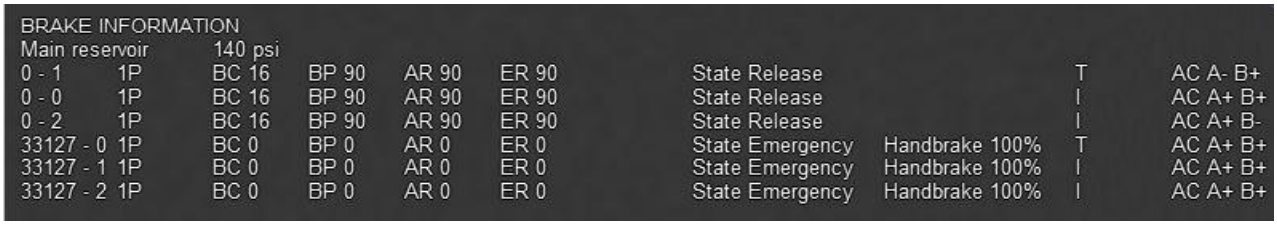
Increasing the Brake Pipe Charging Rate *(PSI/Second)* value controls the charging rate. Increasing the value will reduce the time required to recharge the train; while decreasing the value will slow the charging rate. However, this might be limited by the train brake controller parameter settings in the ENG file. The brake pipe pressure can't go up faster than the equalization reservoir.

The default value, 21, should cause the recharge time from a full set to be about 1 minute for every 12 cars. If the *Brake Pipe Charging Rate (PSI/Second)* value is set to 1000, the pipe pressure gradient features will be disabled and will disable some of the other new brake features, but not all of them.

Brake system charging time depends on the train length as it should, but at the moment there is no modeling of main reservoirs and compressors.

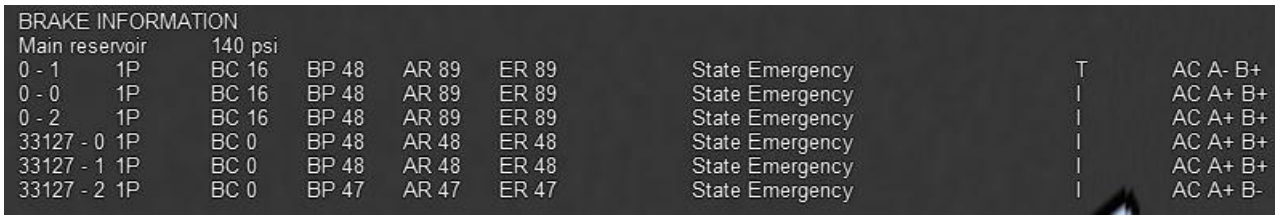
8.6.1. Using the F5 HUD Braking information in the Game

This helps users of Open Rails to understand the status of braking within the game and assists in realistically coupling and uncoupling cars. Open Rails braking physics is more realistic than MSTS, as it replicates the connection, charging and exhaust of brake lines. When coupling to static consists, please note that the brake line for the newly added car doesn't have any values. This is because the train brake line/hose has not yet been connected. The last four columns of each line shows the condition of the air brake hose connections of each car. The columns following "AC" describe the condition of the "Angle Cock" on the end of each brake hose: A is the front cock, B is the rear cock. The Angle Cock is the physical connector between the air lines of adjacent cars. The symbol "+" indicates that the cock is open and the symbol "-" that it is closed. The column headed by "T" indicates if the hose on the loco or car is interconnected: "T" means that there is no connection, "I" means it is connected to the train air pressure line. If two consecutive angle cocks are B+ and A+ respectively, they will pass the main air line pressure between the two cars. In the example below note that the locomotive air brake lines start with A- (closed) and end with B- (closed) before the air hoses are connected to the newly coupled cars. All of the newly coupled cars in this example have their angle cocks open.



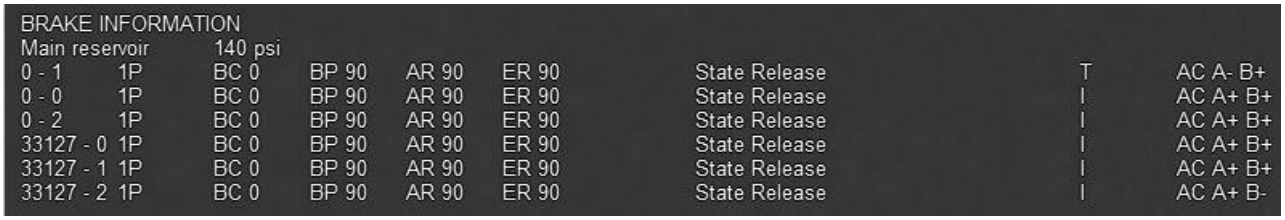
Note that, upon coupling, all the newly added cars have their handbrakes set to 100%. Pressing “Shift+;” (Shift plus semicolon in English keyboards) will release all handbrakes on the consist as shown below. (Pressing “Shift+'” (Shift plus apostrophe on English keyboards) will set all of the handbrakes.)

Pressing the Backslash key (\) (in English keyboards; please check the keyboard assignments for other keyboards) connects the brake hoses between all cars that have been coupled to the engine. The HUD display changes to show the new condition of the brake hose connections and angle cocks:



All of the hoses are now connected; only the angle cocks on the lead loco and the last car are closed as indicated by the “-”. The rest of the cocks are open (“+”) and the air hoses are joined together (all “I”) to connect to the air supply on the lead locomotive.

Upon connection, charging of the brake train line commences. Open Rails uses a default charging rate of about 1 minute per every 12 cars. The HUD display reports that the consist is in “Emergency” state; this is because the air pressure dropped when the empty car brake systems were connected. Ultimately the brake pressures reach their stable values:



Pressing the Shift+/ (in English keyboards) will immediately fully charge the train brakes line to the final state if you don’t want to wait for the train brake line to charge. This action is not prototypical, however.

The state of the angle cocks and the air brake pressure of individual coupled cars can be manipulated by using the F9 display. For example, this will permit more realistic shunting of cars in freight yards. (see F9 Train Operations Monitor.)

When uncoupling cars from a consist, using the Braking F5 HUD in conjunction with the F9 Car Operations display allows the player to do so without losing the air pressure in the remaining cars. This avoids the braking system going into “Emergency” state and requiring reinitialization. The player must first close the angle cock at the rear of the car ahead of the first car to be uncoupled before uncoupling so that the air pressure in the remaining consist is not lost when the air hoses to the uncoupled cars are disconnected.

8.6.2. Dynamic Brakes

Open Rails software supports dynamic braking for engines. To increase the Dynamic brakes

press Period (.) and Comma (,) to decrease them. Dynamic brakes are usually off at train startup (however this can be overridden by the related MSTS setting in the .eng file), the throttle works and there is no dynamic brake line in the HUD. To turn on dynamic brakes set the throttle to zero and then press Period; Pressing Period successive times increases the Dynamic braking forces. If the value n of the MSTS parameter DynamicBrakesDelayTimeBeforeEngaging (n) is greater than zero, the dynamic brake will engage only after n seconds. The throttle will not work when the Dynamic brakes are on.

The Dynamic brake force as a function of control setting and speed can be defined in a DynamicBrakeForceCurves table that works like the MaxTractiveForceCurves table (see here). If there is no DynamicBrakeForceCurves defined in the ENG file, than one is created based on the MSTS parameter values.

8.6.3. Native Open Rails Braking Parameters

Open Rails has implemented additional specific braking parameters to deliver realism in braking performance in the simulation.


The Open Rails team intends to provide separate documentation detailing the usage, syntax, and methodology of how to use these new braking and performance parameters. Check the Open Rails web site periodically for current status *(this should be done now I think ???)*.

Following are a list of specific OR parameters and their default values. The default values are used in place of MSTS braking parameters; however, two MSTS parameters are used for the release state: MaxAuxilaryChargingRate and EmergencyResChargingRate

wagon(brakepipevolume - Volume of car's brake pipe in cubic feet (default .5). This is dependent on the train length calculated from the ENG to the last car in the train. This aggregate factor is used to approximate the effects of train length on other factors.


Strictly speaking this value should depend on the car length, but the Open Rails Development team doesn't believe it's worth the extra complication or CPU time that would be needed to calculate it in real time. We will let the community customize this effect by adjusting the brake servicetimefactor instead, but the Open Rails Development team doesn't believe this is worth the effort by the user for the added realism.

engine(mainreschargingrate - Rate of main reservoir pressure change in PSI per second when the compressor is on (default .4).

engine(enginebrakereleaserate - Rate of engine brake pressure decrease in PSI per second (default 12.5).

engine(enginebrakeapplicationrate - Rate of engine brake pressure increase in PSI per second (default 12.5).

engine(brakepipechargingrate - Rate of lead engine brake pipe pressure increase in PSI per second (default 21).

engine(brakeservicetimefactor - Time in seconds for lead engine brake pipe pressure to drop to about 1/3 for service application (default 1.009).

engine(brakeemergencytimefactor - Time in seconds for lead engine brake pipe pressure to drop to about 1/3 in emergency (default .1).

engine(brakepipetimefactor - Time in seconds for a difference in pipe pressure between adjacent cars to equalize to about 1/3 (default .003).

8.6.4. Brake Retainers

The retainers are controlled using the left and right Brace ({ and }) keys on an English keyboard. The left Brace ({) key will reset the retainer on all cars to exhaust (the default position). The retainer setting is increased each time the right Brace (}) key is pressed. First the number of cars is increased (25%, 50% and then 100%) and then the retainer setting is changed (low pressure, high pressure and then slow direct). For 25% the retainer is set on every fourth car starting at the rear of the train. 50% is every other car and 100% is every car. These changes can only be made when stopped. When the retainer is set to exhaust, the ENG file release rate value is used, otherwise the release rates are hard coded based on some AB brake documentation used by the Open Rails development team.

8.7. Dynamically evolving Tractive Force

The Open Rails development team has been experimenting with max/continuous tractive force, where it can be dynamically altered during game play using the ORTSMaxTractiveForceCurves parameter as shown earlier. The parameters were based on the Handbook of Railway Vehicle Dynamics. This says the increased traction motor heat increase resistance which decreases current and tractive force. We used a moving average of the actual tractive force to approximate the heat in the motors. (*does this still apply ???*)Tractive force is allowed to be at the maximum per the ENG file, if the average heat calculation is near zero. If the average is near the continuous rating than the tractive force is de-rated to the continuous rating. There is a parameter called ContinuousForceTimeFactor that roughly controls the time over which the tractive force is averaged. The default is 1,800 seconds.

8.8. Curve Resistance - Theory

8.8.1. Introduction

When a train travels around a curve, due to the track resisting the direction of travel (ie the train wants to continue in a straight line), it experiences increased resistance as it is “pushed” around the curve. Over the years there has been much discussion about how to accurately calculate curve friction. The calculation methodology presented (and used in OR) is meant to be representative of the impacts that curve friction will have on rolling stock performance.

8.8.2. Factors Impacting Curve Friction

A number of factors impact upon the value of resistance that the curve presents to the trains movement, and these are as follows:

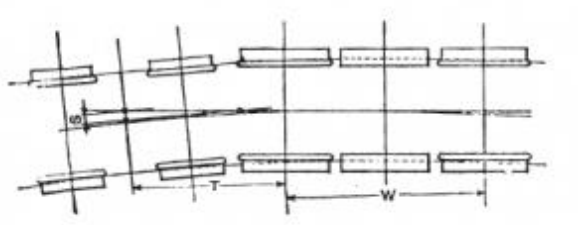
- Curve radius – the tighter the curve radius the higher the higher the resistance to the train
- Rolling Stock Rigid Wheelbase – the longer the rigid wheelbase of the vehicle, the higher the resistance to the train. Modern bogie stock tends to have shorter rigid wheelbase values and is not as bad as the older style 4 wheel wagons.
- Speed – the speed of the train around the curve will impact upon the value of resistance, typically above and below the equilibrium speed (ie when all the wheels of the rolling stock are perfectly aligned between the tracks). See the section below “Impact of superelevation”.

The impact of wind resistance on the curve is ignored.

8.8.3. Impact of Rigid Wheelbase

The length of the rigid wheelbase of rolling stock will impact the value of curve resistance. Typically rolling stock with longer rigid wheelbases will experience a higher degree of “rubbing” or frictional resistance on tight curves, compared to stock with smaller wheelbases.

Steam locomotives usually created the biggest problem in regard to this as their drive wheels tended to be in a single rigid wheelbase as shown in Fig 1. In some instances on routes with tighter curve the “inside” wheels of the locomotive was sometimes made flangeless to allow it to “float” across the track head. Articulated locomotives, such as Shays, tended to have their drive wheels grouped in bogies similar to diesel locomotives and hence were favoured for routes with tight curves.



The value used for the rigid wheelbase is shown as W in Fig 1.

(Diagram Source - The Baldwin Locomotive Works - Locomotive Data - 1944)

Figure 1 - Example of Rigid Wheelbase in steam locomotive

8.8.4. Impact of SuperElevation

On any curve whose outer rail is super-elevated there is, for any car, one speed of operation at which the car trucks have no more tendency to run toward either rail than they have on straight track, where both rail-heads are at the same level (known as the equilibrium speed). At lower speeds the trucks tend constantly to run down against the inside rail of the curve, and thereby increase the flange friction; whilst at higher speeds they run toward the outer rail, with the same effect. This may be made clearer by reference to Fig. 2, which represents the forces which operate on a car at its centre of gravity. With the car at rest on the curve there is a component of the weight **W** which tends to move the car down toward the inner rail. When the car moves along the track centrifugal force **Fc** comes into play and the car action is controlled by the force **Fr** which is the resultant of **W** and **Fc**. The force **Fr** likewise has a component which, still tends to move the car toward the inner rail. This tendency persists until, with increasing speed, the value of **Fc** becomes great enough to cause the line of operation of **Fr** to coincide with the centre line of the track perpendicular to the plane of the

rails. At this equilibrium speed there is no longer any tendency of the trucks to run toward either rail. If the speed be still further increased, the component of **Fr** rises again, but now on the opposite side of the centre line of the track and is of opposite sense, causing the trucks to tend to move toward the outer instead of the inner rail, and thereby reviving the extra flange friction. It should be emphasized that the flange friction arising from the play of the forces here under discussion is distinct from and in excess of the flange friction which arises from the action of the flanges in forcing the truck to follow the track curvature. This excess being a variable element of curve resistance, we may expect to find that curve resistance reaches a minimum value when this excess reduces to zero, that is, when the car speed reaches the critical value referred to. This critical speed depends only on the super-elevation, the track gauge, and the radius of track curvature. The resulting variation of curve resistance with speed is indicated in Fig 3.

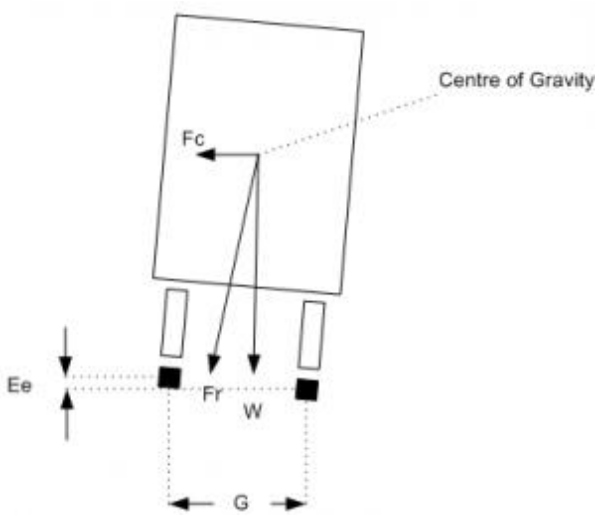


Figure 2 - Description of forces on rolling stock transitioning a curve

8.8.5. Calculation of Curve Resistance

$$R=WF(D+L)2r$$

Where R = Curve resistance, W = vehicle weight, F = Coefficient of Friction, u = 0.5 for dry, smooth steel-to-steel, wet rail 0.1 - 0.3, D = track gauge, L = Rigid wheelbase, r = curve radius.

Source: The Modern locomotive by C. Edgar Allen - 1912

8.8.6. Calculation of Curve Speed Impact

The above value represents the least value amount of resistance, which occurs at the equilibrium speed, and as described above will increase as the train speed increases and decreases from the equilibrium speed. This concept is shown pictorially in the following graph. Open Rails uses the following formula to model the speed impact on curve resistance:

$$SpeedFactor=ABS((EquilibriumSpeed-TrainSpeed)(EquilibriumSpeed)* \text{ResistanceFactor@start})$$

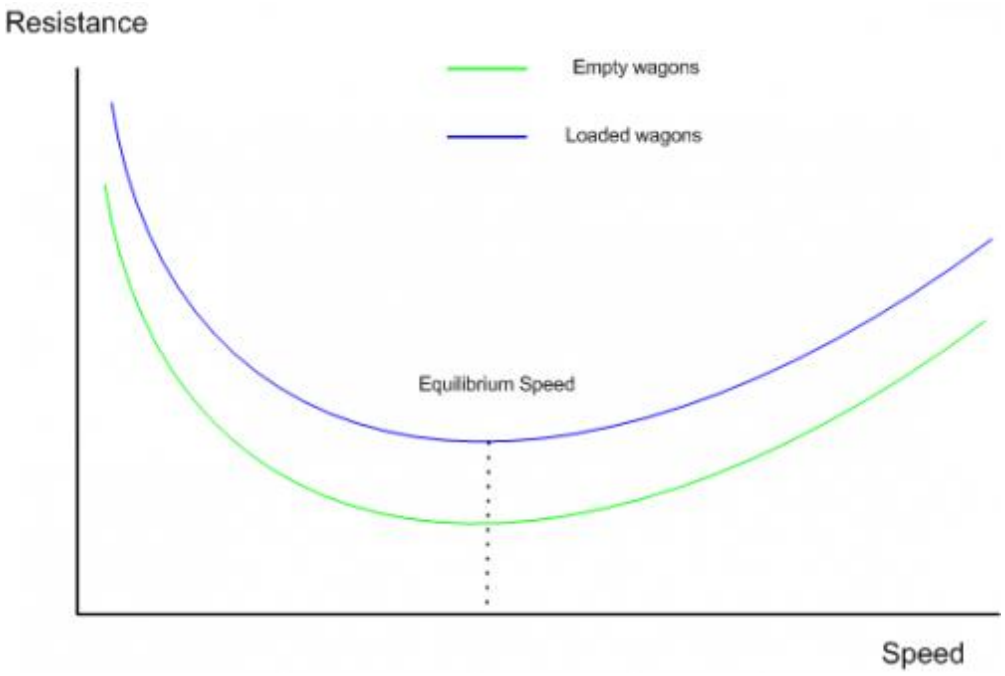


Figure 3 - Generalisation of variation of curve resistance with speed

8.8.7. Further background reading

http://en.wikipedia.org/wiki/Curve_resistance_(railroad)

8.9. Curve Resistance - Application in OR

Open Rails models this function, and the user may elect to specify the known wheelbase parameters, or the above "standard" default values will be used. OR calculates the equilibrium speed in the speed curve module, however it is not necessary to select both of these functions in the simulator options TAB. Only select the function desired. By studying the "Forces Information" table in the HUD, you will be able to observe the change in curve resistance as the speed, curve radius, etc vary.

8.9.1. OR Parameters

Typical OR parameters may be entered in the Wagon section of the WAG or ENG file, and are formatted as below.

```
ORTSRigidWheelBase ( 3in )
ORTSTrackGauge ( 4ft 8.5in) (also used in curve speed module)
```

8.9.2. OR Default

The above values can be entered into the relevant files, or alternatively if they are not present, then OR will use default values as described below.

Rigid Wheelbase – as a default OR uses the figures shown above in the "Typical Rigid Wheelbase Values" section. Starting curve resistance value has been assumed to be 200%, and has been built into the speed impact curves. OR calculates the curve resistance based upon actual wheelbases provided by the player or the appropriate defaults. It will use this as the value at "Equilibrium Speed",

and then depending upon the actual calculated equilibrium speed (from the speed limit module) it will factor the resistance up as appropriate to the current train speed.

Steam locomotive wheelbase approximation – the following approximation is used to determine the default value for the fixed wheelbase of a steam locomotive.

$$WheelBase = 1.25 * (axles - 1) * DrvWheelDiameter$$

8.9.3. Typical Rigid Wheelbase Values

The following values are used as defaults where actual values are not provided by the player.

Rolling Stock Type	Typical value
Freight Bogie type stock (2 wheel bogie)	5’ 6” (1.6764m)
Passenger Bogie type stock (2 wheel bogie)	8’ (2.4384m)
Passenger Bogie type stock (3 wheel bogie)	12’ (3.6576m)
Typical 4 wheel rigid wagon	11’ 6” (3.5052m)
Typical 6 wheel rigid wagon	12’ (3.6576m)
Tender (6 wheel)	14’ 3” (4.3434m)
Diesel, Electric Locomotives	Similar to passenger stock
Steam locomotives	Dependent on # of drive wheels, Can be up to 20'+ , eg large 2-10-0 locos

Modern publications suggest that an allowance of approximately 0.8 lb. per ton (us) per degree of curvature for standard gauge tracks. At very slow speeds, say 1 or 2 mph, the curve resistance is closer to 1.0 lb. (or 0.05% up grade) per ton per degree of curve.

8.10. SuperElevation (Curve Speed Limit) . Theory

8.10.1. Introduction

When a train rounds a curve, it has a tendency to want to travel in a straight direction and the track must resist this movement, and force the train to move around the curve. The opposing movement of the train and the track result in a number of different forces being at play.

8.10.2. 19th & 20th Century Vs Modern Day Railway Design

In the early days of railway construction financial considerations were a big factor in the route design and selection. Given that the speed of competing transport, such as horses and water transport was not very fast, speed was not seen as a major factor in the design process. However as railway transportation became a more vital need for society, the need to increase the speed of trains became

more and more important. This lead to many improvements in railway practices and engineering. A number of factors, such as the design of the rolling stock, as well as the track design, ultimately influence the maximum speed of a train. Today's high speed railway routes are specifically designed for the speeds expected of the rolling stock.

8.10.3. Centrifugal Force

Railway locomotives, wagons and carriages, hereafter referred to as rolling stock, when rounding a curve may be considered as coming under the influence of centrifugal force. Centrifugal force is commonly defined as:

- The apparent force that is felt by an object moving in a curved path that acts outwardly away from the centre of rotation.
- An outward force on a body rotating about an axis, assumed equal and opposite to the centripetal force and postulated to account for the phenomena seen by an observer in the rotating body.

For this article the use of the phrase centrifugal force shall be understood to be an apparent force as defined above.

8.10.4. Effect of Centrifugal Force

When rolling stock rounds a curve, if the rails of the track are at the same elevation (ie two tracks are at the same level) the combination of centrifugal force Fc and the weight of the rolling stock W will produce a resulting force Fr that does not coincide with the centre line of track, thus producing a downward force on the outside rail of the curve that is greater than the downward force on the inside rail (Refer to Figure 1). The greater the velocity or the smaller the radius of the curve becomes (some railways have curve radius as low as 100m), the farther the resulting force Fr will move away from the centre line of track. Equilibrium velocity was the velocity a train could negotiate a curve will the rolling stock weight equally distributed across all the wheels.

If the resulting force Fr approaches the outside rail, then the rolling stock is at risk of “falling” off the track or overturning. The following drawing, illustrates the basic concept described. Lateral displacement of the centre of gravity permitted by the suspension system of the rolling stock is not illustrated.

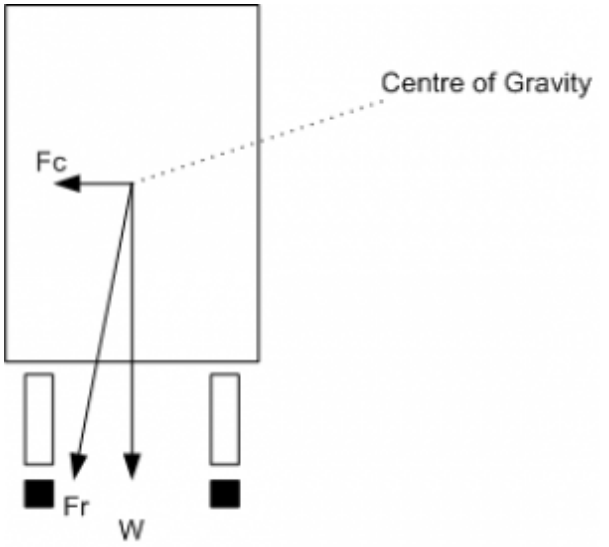
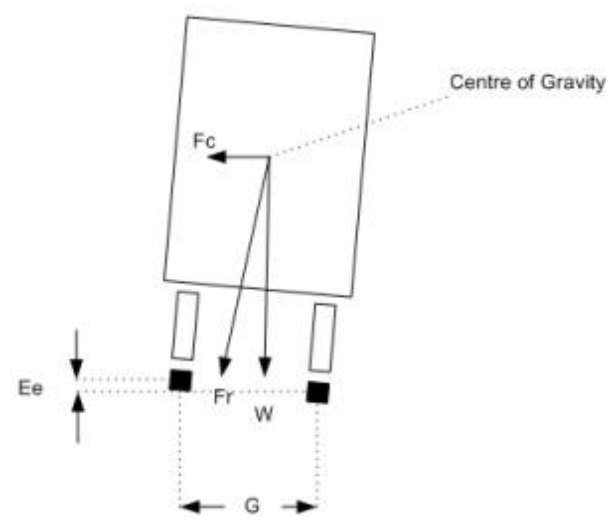


Figure 1 - Forces at work when train rounds a curve

8.10.5. Use of SuperElevation

In order to counteract the effect of centrifugal force Fc the outside rail of the curve may be elevated above the inside rail effectively moving the centre of gravity of the rolling stock laterally toward the inside rail.

This procedure is generally referred to as superelevation. If the combination of lateral displacement of the centre of gravity provided by the superelevation, velocity of the rolling stock and radius of curve is such that resulting force Fr becomes centred between and perpendicular to a line across the running rails the downward pressure on the outside and inside rails of the curve will be the same. The superelevation that produces this condition for a given velocity and radius of curve is known as



the balanced or equilibrium elevation.

Figure 2 - This illustrates the above concept.

8.10.6. Limitation of Superelevation in Mixed Passenger & Freight Routes

Typical early railway operation resulted in rolling stock being operated at less than equilibrium velocity (all wheels equally sharing the rolling stock weight), or coming to a complete stop on curves. Under such circumstances excess superelevation may lead to a downward force sufficient to damage the inside rail of the curve, or cause derailment of rolling stock toward the centre of the curve when draft force is applied to a train. Routine operation of loaded freight trains at low velocity on a curve superelevated to permit operation of higher velocity passenger trains will result in excess wear of the inside rail of the curve by the freight trains.

Thus on these types of routes, superelevation is generally limited to not more than 6 inches.

8.10.7. Limitation of Superelevation in High Speed Passenger Routes

Modern high speed passenger routes, do not carry slower speed trains, nor expect trains to stop on curves, so it is possible to operate these routes with higher track superelevation values. Curves on these types of route are also designed to be relatively gentle radius, and are typically in excess of 2000m (2km) or 7000m (7km) depending on the speed limit of the route.

Parameters	France	Germany	Spain	Korea	Japan
Speed (km/h)	300/350	300	350	300/350	350
Horizontal curve radius (m)	10000 (10km)	7000 (7km)	7000 (7km)	7000 (7km)	4000 (4km)
Superelevation (mm)	180	170	150	130	180
Max Grade (mm/m)	35	40	12.5	25	15
Cant Gradient (mm/s)	50	34.7	32	N/A	N/A
Min Vertical radius (m)	16000 (16km)	14000 (14km)	24000 (24km)	N/A	10000 (10km)

Table 1 - Curve Parameters for High Speed Operations *(Railway Track Engineering by J. S. Mundrey)*

8.10.8. Maximum Curve Velocity

Maximum velocity on a curve may exceed equilibrium velocity, but must be limited to provide a margin of safety before overturning velocity is reached or a downward force sufficient to damage the outside rail of the curve is developed. This velocity is generally referred to as maximum safe velocity or safe speed. Although operation at maximum safe velocity will avoid overturning of rolling stock or rail damage, a passenger riding in a conventional passenger car will experience centrifugal force perceived as a tendency to slide laterally on their seat creating an uncomfortable sensation of instability. To avoid passenger discomfort maximum velocity on a curve is therefore limited to what is generally referred to as maximum comfortable velocity or comfortable speed. Operating experience with conventional passenger cars has led to the generally excepted, circa 1980, of designating the maximum velocity for a given curve to be equal to the result for the calculation of equilibrium velocity with an extra amount added to the actual superelevation that will be applied to the curve. This is often referred to as unbalanced superelevation or cant deficiency. Tilt trains have been introduced to

allow faster train operation on tracks not originally designed for “high speed” operation, as well as high speed railway operation. The tilting of the passenger cab allows greater values of unbalanced superelevation to be used.

8.10.9. Limitation of Velocity on Curved Track at Zero Cross Level

The concept of maximum comfortable velocity may also be used to determine the maximum velocity at which rolling stock is permitted to round curved track without superelevation and maintained at zero cross level. The lead curve of a turnout located between the heel of the switch and the toe of the frog is an example of curved track that is generally not superelevated. Other similar locations would include yard tracks and industrial tracks where increased velocity made possible by superelevation is not required. In such circumstances the maximum comfortable velocity for a given curve may also be the maximum velocity permitted on tangent track adjoining the curve.

8.10.10. Height of Centre of Gravity

Operation on a curve at equilibrium velocity results in the centre of gravity of the rolling stock coinciding with a point on a line that is perpendicular to a line across the running rails and the origin of which is midway between the rails. Under such condition the height of centre of gravity is of no consequence as resulting force Fr coincides with the perpendicular line described. When rolling stock stops on a superelevated curve or rounds a curve under any condition of non-equilibrium resulting force Fr will not coincide with the perpendicular line previously described and the height of the centre of gravity then becomes consequential in determining the location of resulting force Fr relative to the centre line of the track. The elasticity of the suspension system of rolling stock under conditions of non-equilibrium will introduce a roll element that effects the horizontal displacement of the centre of gravity that must also be considered when determining the location of resulting force Fr.

8.10.11. Calculation of Curve Velocity

The generic formula for calculating the various curve velocities is as follows:

$$V=EgrG----\sqrt{}$$

Where: E = Ea (track superelevation) + Ec (unbalanced superelevation)

g = acceleration due to gravity

r = radius of curve

G = track gauge

8.10.12. Typical SuperElevation Values & Speed Impact - Mixed Passenger & Freight Routes

The values quoted below are “typical” but may vary from country to country.

Track superelevation typically will not be more than 6 inches (150mm). Naturally depending upon the radius of the curve speed restrictions may apply.

Normally unbalanced superelevation is typically restricted to 3 inches (75mm), and is usually only allowed for passenger stock.

Tilt trains may have values of up to 12 inches (305mm).

8.10.13. Typical SuperElevation Values & Speed Impact - High Speed Passenger Routes

As freight trains normally don't run on high speed routes, track superelevation values may be up to 12 inches (300mm), depending upon the country. Track designs ensure that curve radius is designed to the relevant route speed without the need to exceed this value. Typical values for some specific countries are shown in the table below.

	Cant D (SuperElevation) (mm)	*Cant deficiency (Unbalanced SuperElevation) I (mm)*
CEN (draft) – Tilting trains	180-200	300
Czech Rep. – Tilting trains	150	270
France – Tilting trains	180	260
Germany – Tilting trains	180	300
Italy – Tilting trains	160	275
Norway – Tilting trains	150	280
Spain – Tilting trains (equivalent for standard gauge)	160 (139)	210 (182)
Sweden – Tilting trains	150	245
UK – Tilting trains	180	300

Table 2 - Superelevation limits *(source - Tracks for tilting trains - A study within the Fast And Comfortable Trains (FACT) project by B. Kufver, R. Persson)*

8.11. SuperElevation (Curve Speed Limit) Application in OR

Open Rails implements this function, and has “standard” default values applied. The user may elect to specify some of the standard parameters from the above formula.

8.11.1. OR Parameters

Typical OR parameters can be entered in the *Wagon* section of the WAG or ENG file, and are formatted as below.

```
ORTSUnbalancedSuperElevation ( 3in )
ORTSTrackGauge( 4ft 8.5in)
```

8.11.2. OR Default

The above values can be entered into the relevant files, or alternatively OR will default to the following functionality.

OR will firstly use the speed limit value from the **TRK** file to determine whether the route is a conventional mixed freight and passenger route or a high speed route.

Speed limit < 200km/h (125mph) – Mixed Freight and Pass route

Speed limit > 200km/h (125mph) – High speed passenger route

“Default” values of *track superelevation* will be applied based upon the above separations.

Track gauge will default to the standard value of 4’ 8.5” (1435mm).

Unbalanced superelevation (Cant Deficiency) will be determined off the value entered by the user, or will default to the following values:

- Conventional Freight – 0” (0mm)
- Conventional Passenger – 3” (75mm)
- Engines & tenders – 6” (150mm)

Tilting trains require the addition of the relevant unbalanced superelevation information into the relevant rolling stock files.

8.12. OR-specific File Inclusions for ENG and WAG files

for content developers:

In preceding paragraphs many references have been done to OR-specific parameters and tables to be included in the ENG and WAG files. MSTS is in general quite tolerant if it finds unknown parameters and even blocks within ENG and WAG files, and continues running regularly. However this way of operating is not encouraged by the OR team. Instead, a cleaner approach has been made possible, that is described here.
Within the trainset folder containing the ENG and WAG files to be upgraded an OpenRails subfolder has to be generated. Within this subfolder a text file named xxxx.eng or xxxx.wag, where xxxx.eng or xxxx.wag are the names of the original files, has to be created.
Referring to the contents of this new file there are two possibilities:
– either such file contains all info included in the original file (except for modified parts of course) plus OR-specific parts if any
– or this new file contains at its beginning only a reference to the original file, plus the modified parts and the OR-specific parts.

An example of an OR-specific bc13ge70tonner.eng file to be placed into the OpenRails subfolder and that uses the second possibility is as follows:

```
include ( ..\bc13ge70tonner.eng )

Wagon (
  MaxReleaseRate ( 2.17 )
  MaxApplicationRate ( 3.37 )
  MaxAuxilaryChargingRate ( .4 )
  EmergencyResChargingRate ( .4 )
  BrakePipeVolume ( .4 )
  ORTSUnbalancedSuperElevation ( 3in )
Engine (
  AirBrakeMainresvolume ( 16 )
  MainResChargingRate ( .5 )
  BrakePipeChargingRate ( 21 )
  EngineBrakeReleaseRate ( 12.5 )
  EngineBrakeApplicationRate ( 12.5 )
  BrakePipeTimeFactor ( .00446 )
```

```
BrakeServiceTimeFactor ( 1.46 )
BrakeEmergencyTimeFactor ( .15 )
ORTSMaxTractiveForceCurves (
  0 (
      0 0 50 0 )
  .125 (
    0 23125
    .3 23125
    1 6984
    2 3492
    5 1397
    10 698
    20 349
    50 140 )
  .25 (
    0 46250
    .61 46250
    1 27940
    2 13969
    5 5588
    10 2794
    20 1397
    50 559 )
  .375 (
    0 69375
    .91 69375
    2 31430
    5 12572
    10 6287
    20 3143
    50 1257 )
  .5 (
    0 92500
    1.21 92500
    5 22350
    10 11175
    20 5588
    50 2235 )
  .625 (
    0 115625
    1.51 115625
    5 34922
    10 17461
    20 8730
    50 3492 )
  .75 (
    0 138750
    1.82 138750
    5 50288
    10 25144
    20 12572
    50 5029 )
  .875 (
    0 161875
    2.12 161875
    5 68447
```

```
        10 34223
        20 17112
        50 6845 )
      1 (
        0 185000
        2.42 185000
        5 89400
        10 44700
        20 22350
        50 8940 )

      )
    )
  )
```

The ORTSMaxTractiveForceCurves are formed by blocks of pairs of parameters representing speed in meters per second and tractive force in Newtons; such blocks are each related to the value of the throttle present on top of each block. For intermediate values of the speed an interpolation is computed to get the tractive force, and the same applies for intermediate values of the throttle.

It is not possible to replace only a part of the Lights() block. It must be replaced in its entirety.

8.13. Testing OR rolling stock

An interesting tool to test OR rolling stock is described and available here .

9.Further Open Rails rolling stock features

9.1. Train Engine Lights

OR supports the whole set of lights foreseen by MSTS.

9.2. Tilting trains

OR supports tilting trains. A train tilts when its .con file contains the “tilted” string: e.g. ETR460_tilted.con.

10. Open Rails train operation

10.1. Open Rails Activities

OR has the aim of running in a compatible way most of the activities written for MSTS. To achieve this the "Enhanced Compatibility with MSTS activities" option has to be selected. This option will be mentioned often in this chapter, and will be shortly called "MSTS Compatibility" option

10.1.1. Player Paths, AI Paths, and How Switches Are Handled

If the player path requires a switch to be aligned both ways, the alignment that is the last on the path is used. If an AI train crosses the player path before the player train gets there, the AI train will leave the switches aligned for the main route (the default setting for most switches)

If you throw a switch to move into a siding, the switch at the far end of the siding is aligned to let you leave when your train first occupies the siding. But after that it is not changed back to its original setting. If the switch gets thrown the other way, you can leave the siding with the switch aligned incorrectly. If you uncouple and re-couple to the train while it occupies the misaligned switch, the rear end of the train will switch tracks.

10.2. Open Rails AI

10.2.1. Basic AI Functionality

- OR supports AI trains. In time, the AI system will become more advanced with new features.
- OR supports two distinct ways of controlling trains: it supports activities in compatibility with MSTS, and it also supports Timetable control. Note that various options and settings are sometimes limited to either activity or Timetable control.
- AI trains can meet if both paths have passing sections defined at the same place.
- Waiting points and reverse points work. Reverse points can be used in both Activity and Timetable modes, while waiting points can only be used in Activity mode. Reversing could take place at a point before the point this takes place in MSTS, when the "MSTS Compatibility" option has not been selected.
- AI trains throw switches not lined properly before engaging them.
- In activity mode AI trains can perform shunting actions, provided the "Extended AI shunting" and "Enhanced Compatibility with MSTS activities" options have been selected.
- Priorities: AI trains should start as scheduled as long as there is no other AI train already on a conflict path.

10.3. Open Rails Signalling

Note that this document details behaviour while in single-player mode only. For multi-player mode, different rules may apply.

10.4. Control Mode

Control Mode defines what interactions there are between the player and the control system, and the level of control of the player on signals and switches.

There are two basic modes: Auto Mode and Manual Mode.
Use the Ctrl-M key to toggle between these modes.

10.4.1. Auto Mode

In Auto Mode the control system sets the train's path and signals, and the player cannot change the setting of the switches or request for signals at danger to clear. The train's route is taken from the path as defined in the Activity Editor or timetable definition, and the system will attempt to clear the route ahead of the train according to the signalling rules and interaction with other trains.

No route is cleared in the reverse direction as the train is assumed to not run in reverse. Selecting a reverse cab or changing the position of the reverser does not change the direction of the route. In fact, the route will not be reversed other than at reversal points as defined in the train's path. At these reversal points, the route will reverse automatically as soon as the train stops.
If the train does accidentally run backward, e.g. due to slipping or setting back after overshooting a platform, only safety checks are performed for the rear end of the train with respect to signals, switch alignment, other trains and end of track. There is no check on speedlimits behind the train.

Setting switches using the F8 window or G/Shift-G is not allowed. Setting switches using Alt-left mouseclick is possible, but not allowed for switches in the train's path. However, any switches set manually will automatically be reset by an approaching train according to that train's path. So, in Auto Mode the train can not deviate from the defined path.

A request to clear a signal ahead of the train using the Tab command is only allowed when the track ahead is occupied by another train which is at a stand-still, and when that track is in the train's route. A request to clear a signal which would lead the train off its route is not allowed. A request to clear a signal behind the train using Shift-Tab is also not possible.

Auto Mode is intended for normal running under control of signals or traffic control. Shunt moves can be performed if fully defined in the train's path, using reversal points etc..

10.4.1.1.Details on Auto Mode

There are two sub-modes to Auto Mode: Auto Signal and Auto Node.
Auto Signal is the normal mode on signalled routes. The train's route is generally cleared from signal to signal. Only in specifically defined situations can routes be cleared short of a signal as detailed below.

Auto Node is set when the train has not encountered any signals yet, e.g. on unsignalled routes or at the start of the route when there is no signal along the path of the train as far as it can be cleared - e.g. in yards where the train starts but has no clear route yet to the first signal.

Auto Node can also be set if the route ahead cannot be fully cleared up to the next signal, and partial clearing is allowed.

A number of sub-states are defined in Auto Node, depending on the reason clearance is terminated. In the list below, (***A***) indicates a subtype which can occur if no signal has yet been encountered, (***B***) indicates a subtype when a route from a signal is partially cleared.

The following states are possible :

- *(**A**)* route ahead is clear to the maximum distance for which the track is cleared. The control mode is set to Auto Node - Max Distance.

- *(**A**)* route ahead is blocked at a switch which is aligned for and occupied or reserved by another train. Control mode is set to Auto Node - Misaligned Switch.

- *(**A**)(**B** - only if signal allows access to occupied track, or after Tab command)* route ahead is occupied by a stationary train or train moving in the same direction. Control mode is set to Auto Node - Train Ahead.

  Note that, for (***A***), it should not be possible that the route ahead is occupied by a train moving in opposite direction - in that case, there should always be a misaligned switch in the train's path.

  For (***B***), a signal will never clear when the train ahead is moving in the opposite direction, nor will the Tab request be granted.

- *(**A**)(**B**)* the train's defined path terminates short of the next signal, or there is a reversal point short of the next signal, and there is at least one switch between this point and the next signal.
  The control mode changes to Auto Node - End of Path.

  Note that if there is no switch between the terminating or reversal point and the next signal the route is automatically extended to the next signal.

- *(**A**)(**B**)* the train has passed the last signal before the end of the track, or the train has reached the end of track without encountering any signal. The control mode changes to Auto Node - End of Track.

Changes from Auto Node to Auto Signal and vice-versa are automatic and cannot be influenced by the player.

10.4.2. Manual Mode

When it is required for a train to move off its defined path, a player can switch his train to Manual Mode. This will allow the player to set switches and request to clear signals off its path. However, there are a number of restrictions when running a train in Manual Mode.

In Manual Mode, a route is cleared from the train in both directions, ahead of and behind the train. The route is cleared to a shorter distance as compared to Auto Mode, and is never cleared automatically beyond the first signal. If a train is moving and passes a signal in the opposite direction, the route behind the train will automatically retract to that signal as that is now the next signal in the reverse route. Similarly, of course, when train is running in reverse with respect to signals ahead.

The route orientation will not change whatever direction the train is running. It is fixed to the orientation of the route as it was the moment the player switched to Manual Mode. So, changing to a reverse-facing cab or changing the position of the loco's reverser does not change the direction of the route orientation. This is no limitation on the train's behaviour, as routes are always cleared in both directions. It does, however, affect the display of the F4 and F8 windows, as the top/bottom direction of these windows is linked to the route direction and will therefore not change if the train reverses. To assist the player in his orientation in which direction the train is moving, an 'eye' has been added to these displays symbolizing the direction of the cabview, and an 'arrow' has been added to symbolize the direction of the reverser.

The player can set all switches in the train's path using the F8 window or the G/Shift-G keys. The G key will set the first switch ahead of the train (as defined by the route direction), Shift-G sets the switch behind the train. It is also possible to set switches as required using the Alt-left mouseclick command. Switches can be set even if they are in the train's path and a signal has been cleared over that path. Switches can, of course, not be set if already set as part of a cleared route for another train.

The following rules apply to the setting of switches :

- all switches will remain in the position in which they were set by the last train passing over that switch. If no train has yet passed over the switch, it is in its default position.
- when in Manual Mode, trailing switches will not be automatically aligned for the approaching player train, **except** :
- when a route is cleared through a signal while in Manual Mode, any trailing switches in the train's path up to the end of authority (e.g. next signal) will be aligned.
  Note that in this case, trailing switches in the path cleared by the signal can no longer be reset.

Signals which the train approaches will not be cleared automatically. The player must request clearance of all signals encountered using the Tab or Shift-Tab keys.
The Tab key will clear the signal ahead of the train (according to the route direction), the Shift-Tab key will clear the signal behind the train. Repeated use of (Shift-)Tab will clear the next signal beyond the first cleared signal etc., but only up to the maximum clearing distance.

Signals will always clear on request except when the section immediately behind the signal is already cleared for a train from the opposite direction. The normal route-setting limitations etc. are ignored. The signal will only clear to the first available most restrictive aspect above Stop.
Note that, in contrast to the situation in Auto Mode, as the signal will clear even if the full route behind the signal is not available, a cleared signal is no indication of the cleared distance beyond that signal. It may be that the first switch beyond the signal is already cleared for another train.

Therefore, when in Manual Mode, use of F4 window or of the Dispatcher window to check on the route availability is essential when running in an area with AI traffic.

When in Manual Mode, deadlock prevention processing is switched off. This is because of the changes in the train's route and direction which are likely to occur in Manual Mode could jeopardise the stability of the deadlock processing. So care should be taken when using Manual Mode in an area with AI traffic, specifically on single track sections.

The only requirement to switch from Auto Mode to Manual Mode is that the train is at a standstill. The Ctrl-M key toggles between Auto Mode and Manual Mode. When switching from Auto Mode to Manual Mode, all signals already cleared will be reset, and new routes are cleared ahead of and behind the train for the maximum distance if possible, or up to the first signal.

To switch back from Manual Mode to Auto Mode the front of the train must be on the path as defined in the Activity Editor. If the path contains reversal points, the train must be in between the same reversal points as where it was when it switched to Manual Mode (same subpath).

If the train is moving in the direction as the path defines, switch back to Auto Mode can be done while the train is moving. The rear of the train need not be on the defined path, only the front.

If the train is moving in the opposite direction, it must be at a standstill in order to switch back to Auto Mode. If the orientation of the train's route somehow was reversed (e.g. by moving through a balloon-line or a Y-section) and differs from the direction in the defined path, both front and rear must be on the defined path. In this situation, the orientation will switch back to the direction as defined in the path.

10.4.3. Out-of-Control Mode

This is a special mode. Normally, the player train should not be in this mode.
The out-of-control mode is activated when the player violates a security rule.
Such incidents are :

- when the player train passes a signal at danger;
- when the player train passes over a misaligned switch;
- when the player train runs beyond the end of the authorised path.

Such actions will place the player train in out-of-control mode.
In this situation, the emergency brake is activated and maintained until the train is stopped. The player has no control over his train until it is at a standstill.

Once the train has stopped, the player can switch to Manual Mode to try to restore to a correct situation (e.g. set back to in front of the signal at danger, authorised path etc.). Once a normal situation has been restored, the player can switch back to Auto Mode. If the action led the player train onto a section of track already cleared for another train, that train too is stopped.

10.4.4. Explorer Mode

When Explorer Mode is started instead of an activity, the train is set to Explorer Mode.

The player has full control over all switches. Signals will clear as normal but signals can be cleared over routes which are not normally available using the Tab or Shift-Tab commands.

10.5. Track Access Rules

All trains clear their own path. When in Auto Signal mode, part of that function is transferred to the signals.

In Auto Node mode, trains will clear their path up to 5,000 metres, or the distance covered in 2 mins. at the max. allowed speed, whichever is furthest. In Auto Signal mode, the number of signals cleared ahead of the train is taken from the value of the SignalNumClearAhead parameter as defined in the sigcfg.dat file for the first signal ahead of the train.

In Manual mode, the distance cleared is 3,000 metres maximum, or as limited by signals.
Distances in Explorer Mode are similar to those in Auto Mode.
If a train is stopped at a signal it can claim the track ahead ensuring it will get priority as next train onto that section, but to avoid needless blocking of other possible routes, no claim is made if the train ahead is also stopped.

No distinctions are made between any type of train, and there are no priority rules.

10.6. Deadlock Processing

When a train is started, it will check its path against all other trains (including those not yet started). If a section is found on which this train and the other train are due in opposite directions, the boundaries of that total common section are determined, and ‘deadlock traps’ are set at those boundaries, for each train in the appropriate direction. These boundaries are always switch nodes. When a train passes a node which has a ‘deadlock trap’ for that train, the trap is sprung. When a train approaches a node which has an active deadlock, it will stop at that node, or at the last signal ahead of it if there is one. This train will now also spring its deadlock traps, and will claim the full common section of that deadlock to ensure it will be the next train allowed onto that section. The deadlock traps are removed when a train passes the end node of a deadlock section.

When a train is started, and the train’s path includes one more reversal points, deadlocks are only checked for the part of the path upto the first reversal point. On reversal, deadlocks are checked for the next part etc..
Deadlock traps are removed when a train switches to Manual mode. When the train switches back to Auto mode, the deadlock check is performed again.

There are no deadlock checks in Explorer Mode as there are no AI trains when running in that mode.

If an alternative path is defined (using the Passing Path definition in MSTS Activity Editor), and the train is setting a route to the start node of this alternative path, it will check if a deadlock is set for the related end node. If so, and the alternative path is clear, it will take the alternative path, allowing the other train to use the main path. If the alternative path is already occupied, the train will wait short of the node where the path starts (or the last signal in front, if any); this is to prevent blocking both tracks which would leave the opposite train nowhere to go.

Further rules for use of the alternative paths :

- Trains from both direction must have the same main path through the area.
- If only one train has an alternative path defined, and trains are to pass, that train will always use the alternative path, the other train will always use the main path, regardless of which train arrives first.
- If both trains have an alternative path defined, and trains are to pass, the first train to clear its route will take the alternative path. Note that this need not always be the first train to arrive - it could be that the train which first clears its path takes much longer to actually get to the passing loop.

10.7. Reversal Points

If a reversal point is defined, the path will be extended beyond that point to the end of the section, this is to the next switch or signal or the end of track.

The ‘diverging’ point is determined - this is the switch node where the reverse route diverges from the incoming route. From this point, a search is made for the last signal facing the reverse direction which is located such that the full train will fit in between the signal and the end of the path. If there is such a signal, this will become the ‘diverging’ point. In order for a train to be able to reverse, the rear of the train must be clear of this ‘diverging’ point.

From this point on reversal points behave differently if the “Enhanced compatibility with MSTS activities” option is selected or not.

10.7.1. Reversal Points when no MSTS compatibility option set

When a train has cleared the ‘diverging’ point, the reversal will take place immediately as the train stops. Note that the train need not have reached the actual reversal point, nor that it needs to be clear of any switches between the ‘diverging’ point and the end of the path. The ‘diverging’ point is shown in the F4 track monitor window as the position which the front of the train must have reached to ensure the rear end is clear.

Double reversal points will generally work as in MSTS, except that the train must stop in the section containing the double reversal points.

Because reversal points are activated as the train is stopped clear of the diverging point, reversal points placed in the same section as the starting point (i.e. before the first signal or switch) are immediately activated on start of the activity. This also applies to double reversal points in such a position, which therefore don’t work as intended. Note that this is only necessary for double reversal points placed immediately in front of the starting point.

10.7.2. Reversal Points when MSTS compatibility option set

In this case for AI trains reversal takes place like under MSTS, that is when the AI train reaches with its first car the reversal. If at that point the rear of the train has not yet cleared the diverging point, the reversal takes place later, when such diverging point is cleared.

For player trains the reversal can take place starting from 50 meters before the reversal point provided the diverging point is cleared.
As in MSTS, double reversal points can be used to set at red a signal after such reversal points. However waiting points are recommended for this, as explained in next paragraph.

10.8. Waiting Points

10.8.1. General

Waiting points (WP) set in a path run by an AI train are regularly respected by the train, and executed when the head of the train reaches the WP.
Differently from MSTS, waiting points don't influence the length of the reserved path, except when the WP is followed by a signal in the same track section (no nodes – that is switches – in between).
WPs set in a path run by a player train have no influence on the train run, except – again when the WP is followed by a signal in the same track section.
In such case, both for AI trains and player train, the signal is set to red when the train approaches the WP.
For AI trains the signal returns to green (if the block conditions after the signal allow this) one second after expiration of the WP.
Player trains must stop BEFORE the WP (else they get an emergency stop). In this case the signal returns to green 5 seconds after expiration of the WP.
If there are more WPs in the track section where the signal resides, only the last one influences the signal.
WPs cannot be used in Timetable mode.

10.8.2. Absolute waiting points

When the “Extended AI shunting” option is selected and OR is not in Timetable Mode, waiting points with a waiting time between 30000 and 32359 are interpreted as absolute time-of-day waiting points, with a format 3HHMM, where HH and MM are hour and minute of day in standard decimal notation.
If the AI train will reach the WP before such time of day, the WP will expire at HH:MM. If the AI train will reach the WP later, the WP will expire after a second. This type of WP can be used also in conjunction with a signal in the same track section, as explained in preceding paragraph.
Again, such waiting points won't have effect on a player train if there is no signal in the same section; if instead there is the signal, it will stay on red until the WP has expired or until the train will stop in front of the WP (the later of the two events will be considered).

Absolute waiting points are a comfortable way of synchronizing and scheduling train operation.

10.9. Signals at Station Stops

If the signal at the end of a platform protects a route which contains switches, that signal will be held at danger up to 2 mins. before the booked departure. If the station stop is less than 2 mins., the signal will clear as the train comes to a stand. This applies to both AI train and player trains.
However, if the platform length is less than half the train length, the signal will not be held but will clear as normal to allow the train to properly position itself along the platform. Signals which only protect plain track will also not be held.
In some railway control systems trains don't get a red at the station starting signal when they have to stop in that station.

To achieve this, with MSTS compatibility option set, you can select option "No forced red at station stops" to disable this signal behavior.

Signals at waiting points for player trains will be held at danger until the train has stopped and the waiting point has expired. For signals at waiting points for AI trains, see preceding paragraph.

10.10. Speedposts and Speed Limits Set by Signals

Speedlimits which raise the allowed speed, as set by speedposts or signals, only become valid when the rear of the train has cleared the position of speedpost or signal.

When a speedlimit set by a signal is lower than the speedlimit set by the last speedpost, the speedlimit is set to the lower value. However, when a speedlimit as set by a signal is higher than the present speedlimit set the last speedpost, the limit of the speedpost will be maintained. If a lower speedlimit was in force due to a limit set by another signal, the allowed limit is set to that as defined by the speedpost.

If a speedpost sets a limit which is higher than that set by the last signal, the limit set by the signal is overruled and the allowed limit is set to that as defined by the speedpost.

Instead, when the MSTS compatibility option is set, the valid speedlimit is always the lower between that of the last signal and that of the last speedpost.

10.11. Further features of AI train control

- AI trains always run in Auto control mode.
- AI trains will ignore any manual setting of switches and will reset all switches as defined in their path.
- AI trains will stop at stations and will adhere to the booked station departure times if possible.
- AI trains will stop at a platform such that the middle of the train stands in the middle of the platform. If the train is longer than the platform, it means both front and rear of the train will stand outside the platform. If the platform has a signal at the end, and this signal is held at danger (see above), and the train is too long for the platform, it will stop at the signal. But if the train length is more than double the platform length, the signal will not be held.
- AI trains will adhere to the speed limits.
- AI trains will stop at signal at approx. 30 m. short of a signal at danger.
- At waiting points, the AI trains will stop at the waiting point. Any signal beyond the waiting point is kept at danger until the required departure time.
- Where AI trains are allowed to follow other trains in the same section passing permissive signals, the train will adjust its speed to that of the train ahead, and follow at a distance of approx. 300m. If the train ahead has stopped, the train behind will draw up to a distance of

about 50m. However, if the train ahead is stopped in a station, and the train behind is also booked to stop at that station, the train will draw up behind the first train up to a distance of a few metres.

- The control of AI trains before the start of an activity is similar to the normal control during an activity, except that the update frequency is reduced from the normal update rate to just once per second. But all rules regarding speedlimits, station stops, deadlock, interaction between AI trains (signals etc.) are followed. The position of all AI trains at the start of an activity therefore is as close as possible to what it would have been if the activity had been started at the start time of the first AI train.

10.12. Location-linked passing path processing

for content developers:

Passing paths can be used to allow trains to pass one another on single track routes. The required passing paths are defined per train path in the MSTS Activity Editor or in the native ORTS path editor included within TrackViewer.

This present version is an 'intermediate' stage leading to complete new processing. The data structure and processing have already been prepared for the next stage, when 'alternative paths' (so not just a single passing path but multiple paths through a certain area) will be defined per location, and no longer per train.

This version, however, is still based on the MSTS activity and path definition, and therefore still based on definition of alternative paths per train.

The setup of this version is as detailed below :

- Passing paths defined for the *player* train are available to **all** trains - in both directions.
  The 'through' path of the player train is taken to be the "main" path through that location. This only applies to Activity mode, as there is no predefined player train when running in Timetable mode.
- Each train can have definitions for additional passing paths, these will be available to that train only.
  Note that this implies that there can be more than one passing path per location.
- When possible passing locations are determined for each pair of trains, the train lengths are taken into consideration.
  A location is only 'valid' as a passing location if at least one of the trains fits into the *shortest* of the available passing paths.
- The order in which passing paths are selected :
  - If no train is approaching from the opposite direction (through route) :
    - Train's own path.
    - "Main" path.

- Any alternative path.

- If train is to pass another train approaching from the opposite direction (passing route) :
  - Train's own path (if not the same as "main" path).
  - Alternative path.
  - "Main" path.

  However, in the situation where the train does not fit on all paths, for the first train to claim a path through the area, preference is given to the paths where the train will fit (if any).

The setting of the 'deadlock' trap (the logic which prevents trains from getting on a single track from both directions) has also been changed.
In the 'old' version, the trap was 'sprung' as a train claimed it's path through a possible passing area. However, this often lead to quite early blocking of trains in opposite direction.
In this version the trap is 'sprung' when a train actually claims it's path in the single track section itself.
One slight flaw in this logic is that this can lead to the train which is to wait will be allocated to the "main" path, while the train which can pass is directed over the "loop". This can happen when two trains approach a single track section at almost the same time, each one claiming it's path through the passing areas at either end before the deadlock trap is actually sprung.

One word of warning : if a passing location contains platforms and there are passenger trains which are booked to stop there, passing paths should not be used for that location. A passenger train directed onto an 'alternative' path will miss it's station stop and, for AI trains, it means it will also miss all further station stops as the missed stop is not removed from the list, and further station stops are therefor not processed.

Selecting this type of passing path with the related experimental option processing can lead to considerable changes in the behaviour of trains on single track routes - and behaviour that is certainly significantly different from that in MSTS.

10.13. Further differences between activity running with MSTS compatibility option set or not

10.13.1. End of run of AI trains

- MSTS compatibility option set: AI trains end their run where the end point of their path resides, as in MSTS
- option not set: AI trains end their run at the next signal or at the next node (the nearest of the two)

10.13.2. “Default Performance” and “Performance” parameters

- MSTS compatibility option set:
    - if the AI train does not make station stops, its maxspeed (not considering signal, speedpost and route speed) is given by the first MaxVelocity parameter in the .con file, expressed in meters per second, multiplied by the "Default performance" parameter (divided by 100) that can be found and modified in the MSTS AE in the "Service editor". Such parameter divided by 100 is written by the AE in the .srv file as "Efficiency".
    - If the AI train makes station stops, its maxspeed depends from the "Performance" parameter for every route section, as can be seen and defined in the AI train timetable (that is maxspeed is the product of the first MAxVelocity parameter by the "Performance" parameter divided by 100).
    - Such performance parameter list is written (divided by 100) by the AE in "Service_Definition" block in the activity editor, again as "Efficiency" (for every station stop).
    - from the starting location of the AI train up to the first station, the "Performance" linked to such station is used; from the first station to the second one, the "Performance" linked to the second station is used and so on. From the last station up to end of path the "Default performance" mentioned above is used.
    - This corresponds to MSTS behaviour
    - Moreover the Efficiency parameter is used also to compute acceleration and braking curves.
- Option not set: the Efficiency parameter is used only to compute acceleration and braking curves.

10.13.3. Start of run of AI train in a section reserved by another train

- MSTS compatibility option set: the AI train is created, as in MSTS. It is to the activity creator not to generate deadlocks. Creation of a train in a section where another train resides is possible only if the created train is “behind” the pre-existing train
- option not set: the AI train is not created until the section becomes free. Creation of a train in a section where another train resides is possible only if the created train is “behind” the pre-existing train.

10.13.4. Stop time at stations

- MSTS compatibility option set:
    - The platform passenger number as defined by the MSTS activity editor is considered.
    - Each passenger requires 10 seconds to board. Such time must be divided by the number of passenger wagons within the platform boundaries. Also locos with the line PassengerCapacity in their .eng file count as passenger wagons (EMU, DMU). The criterium to define if a passenger wagon is within the platform boundaries is different for player train and AI train. For player train an individual check is made on every passenger wagon to check if it is within the plaform boundaries (I have supposed that

this is OK if at least two thirds of the wagon are within). For AI train instead the number of wagons+engines within the platform is computed, and all of them, up to the amount of the passenger wagons in the consist, are considered as passenger wagons. The player or AI train boarding time is added to the real arrival time, giving a new departure time; such new departure time is compared with the scheduled departure time and the higher is selected as real departure time.

- AI freight trains stop 20 seconds at stations.
- A train is considered as a passenger one if at least one wagon (or engine) carries passengers.
- AI real freight trains (0 passenger cars) stop 20 seconds at stations as MSTS if scheduled starting times are not present. If they are present the freight trains will stop up to the scheduled starting time or up to the real arrival time plus 20 seconds, selecting the higher of the two.
- A special behaviour has been introduced for trains with more than 10 cars and having a single passenger car. This type of trains has been used in MSTS to have the possibility of defining schedules also for freight trains. These trains are managed - like MSTS - as passenger trains with the rules defined above. However a simplification for the player has been introduced for the player train: if such train stops with the single passenger car outside of the platform, the stop is anyhow considered valid.
- All this is compatible with MSTS operation; only the fact that also scheduled departure time is considered for AI trains differs, as it is considered an improvement.

- Option not set:
  - In place of the platform passenger number, the boarding time in seconds is as defined for such platform in parameter PlatformMinWaitingTime ( ) within the route's .tdb file
  - such value is added to the real arrival time and the highest between such addition and the departure time is selected as real departure time.

10.14. Extended AI train shunting

for content developers:

10.14.1. General

When this option is selected together with the “Enhanced compatibility with MSTS activities” option within the “Experimental Options” menu window , further AI train shunting functions are available.

Following additional shunting functions are available:

1. AI train couples to static consist and restarts with it
2. AI train couples to player or AI train and becomes part of it; coupled AI train continues on its path
3. AI train couples to player or AI train and leaves to it its cars; coupled and coupling train continue on their path
4. AI train couples to player or AI train and “steals” its cars; coupled and coupling train continue on their path
5. AI train uncouples any number of its cars; the uncoupled part becomes a static consist. With the same function it is possible to couple any number of cars from a static consist.

This option is active only outside Timetable mode and if the “Enhanced compatibility with MSTS activities” option has been selected.

10.14.2. Activity Design for extended AI train shunting functions

Activity design can be performed with the MSTS Activity Editor, and doesn't need post-processing of the created files.

First functions 1 to 4, that always involve a coupling, are discussed.
It is not desired that AI trains couple to other trains because of e.g. timing problems, in case they were designed to run separately. So, coupling is activated only if some conditions are met.
In general the signal protection rules apply, that is an AI train will find a red signal if its path leads it directly to another train. So in general these functions can be used only if there are no signals between coupling train and coupled train. However, at least in some cases, this can be overcome by the player by forcing the signal to clear state with the dispatcher window.
Coupling with static consist is not subject to other conditions, as if the activity designer decided that the path would lead an AI train up to against a static consist, he also wanted that the AI train coupled.
Coupling with another AI train or with the player train is subject to following conditions, that is
1) either coupling happens in the last path section of the coupling AI train, and the path end point is under the coupled train or beyond in same section
2) or coupling happens in the last section before a reverse point of the coupling AI train, and reverse point is under the coupled train or beyond in same section.
This way undesired couplings are avoided in case the AI train has his path running in same direction beyond the coupled train.

Just after coupling there is another check to define what happens.
In case the coupled train is static:
1) If there is at least one reverse point further in the path or if there are more than 5 track sections further in the path, the coupling train couples with the static train, and then the so formed train

restarts following the path of the coupling train
2) if not, the coupling train couples with the static train and becomes part of the static train itself (is absorbed by it), stopping movement.

In case the coupled train is a player train or an AI train:
a) if there is at least one reverse point further in the path of the coupling train, the coupling train couples with the coupled train; at that point there are two possibilities:
a1) the trainset coupling to the coupled train is a wagon: in this case the coupling train leaves to the coupled train all the cars between its loco and the coupled train, decouples and moves further in its own path (it can only reverse due to above conditions). The coupled train follows its own path.
a2) the trainset coupling to the coupled train is a loco: in this case the coupling train “steals” from the coupled train all the cars between the coupled train's loco and the coupling train, decouples and moves further in its own path (it can only reverse due to above conditions). The coupled train follows its own path.
b) if there is no reverse point further in the path of the coupling train, the coupling train couples with the coupled train and becomes part of it (is absorbed by it). The coupled train follows its own path.

Now on how to design paths:
a) if one wants that the coupling train is absorbed by the coupled train: simply put the end point of the path of the coupling train below the coupled train
b) if one wants that the coupling train moves further on in its path after having coupled with the coupled train: put in the path of the coupling train a reverse point below the coupled train. If one wants also that the coupling train does not immediately restart, but that it makes a pause, a waiting point has to be added in the path of the coupling train, subsequently to the reverse point. Such waiting point has to be put below the coupled train if this is a static consist, and below what remains of the coupling train just after having coupled with the coupled train, in case the latter is an AI train; in this case the position of the WP can be a bit difficult to be determined if the coupling train left its cars to the coupled train, because in the standard case what remains of the coupling train after coupling/decoupling is only the loco.
If the coupled train is an AI train, obviously it must be stopped on a waiting point when it has to be coupled by the coupling train.

Here function 5 (AI train uncouples any number of its cars) is described.
To uncouple a predefined number of cars from an AI train, a waiting point has to be inserted.
The format of the waiting point (in decimal notation) is 4NNWW, where NN is the number of cars in front of the AI train that are NOT uncoupled, loco included, and WW is the duration of the waiting point in seconds. It must be noted that "front" of the AI train is the part which is in front of the train in the actual direction. So, if the consist has been created with the loco at first place, the loco will be at the front up to the first reverse point. At that point, "front" will become the last car and so on.
Following possibilities arise:
1) AI train proceeds and stops with loco at the front, and wants to uncouple and proceed in the same direction: a WP is inserted where the AI train will stop, with the above format, counting cars starting from the loco.
2) AI train proceeds with loco at the rear, and wants to uncouple and proceed in reverse direction: a reverse point has to be put in the point where the train will stop, and a WP has to be put sequentially after the reverse point, somewhere under the part of the train that will remain with the train, formatted as above. As the train has changed direction at the reverse point, again cars are counted starting from the loco.
3) AI loco proceeds and couples to a loose consist, and wants to get only a part of it: a reverse point is inserted under the loose consist, and a WP is inserted sequentially after the reverse point, somewhere under the part of the train that will remain with the train, formatted as above.

What is NOT possible as of now is to couple the AI train to the player train or to another AI train, and to "steal" to it a predefined number of cars. With the already available functions it is possible to steal only all cars or to pass all cars. If it is desired that only a certain amount of cars is passed from an AI or player train to the other, the first AI train has to uncouple such cars as described above, has to move a bit forward, and then the second AI train couples to those cars.

10.15. Signal related files

for content developers:

OR manages signals as defined in files sigcfg.dat and sigscr.dat in a way that is highly compatible to MSTS.

10.15.1. SignalNumClearAhead

Specific rules however apply to sigcfg.dat parameter SignalNumClearAhead (), that is not managed in a consistent way by MSTS.

If for a SignalType only one SignalNumClearAhead () is defined (as standard in MSTS files), such parameter defines the number of NORMAL signal heads (not signals!) that are cleared down the route, including the signalheads of the signal where the SignalType resides.

If for a SignalType a second SignalNumClearAhead () parameter is added just before the existing one, OR interprets it as the number of NORMAL SIGNALS that are cleared down the route, including the signal where the SignalType resides.

MSTS will skip this first SignalNumClearAhead () and will consider only the second. This way this change to sigcfg.dat does not affect its use in MSTS.
However, instead of modifying the copy of file sigcfg.dat residing in the routes' root, the approach described in next paragraph is recommended.

10.15.2. Where to put OR-specific sigcfg and sigscr files

OR-specific sigcfg and sigscr files have to be put in a subfolder “OpenRails” residing within the main folder of the route. Sigcfg.dat must maintain its name, while the sigscr files can also have other names, provided that within sigcfg.dat there is a reference to such other names.

10.16. OR-specific signalling functions

A set of powerful OR-specific signalling functions are available. Sigcfg and sigscr files referring to these functions must be located as described in previous paragraph.

10.16.1. SPEED signals – new signal function type

The SPEED signal function type allows a signal object marker to be used as a speed sign.

The advantages of such use are :

- The signal object marker only applies to the track on which it is placed. Original speed signs always also affect any nearby lines, making it difficult and sometimes impossible to set a specific speed limit on just one track in complex areas.
- As signal object, the SPEED signal can have multiple states defined and a script function to select the required state, e.g. based on route selection. This allows different speed limits to

be defined for different routes through the area, e.g. no limit for the main line but specific limits for a number of diverging routes.

The SPEED signal is fully processed as a speed limit and not as a signal, and it has no effect on any other signals.

Limitation : it is not possible to define different speeds related to type of train (passenger or freight).

10.16.1.1. Definition and usage

Definition is similar as for any other signal, with SignalFnType set to "SPEED".

It allows definition of drawstates and aspects like any other signal. Different speed values can be defined per aspect as normal.

An aspect can be set to not have an active speedlimit. If this aspect is active, the speed limit will not be changed. This can, for instance, be used if a route-linked speed limit is required. This aspect can then be set for a route for which no speed limit is required.

An aspect can also be set to not have an active speedlimit but with a special signal flag : OR_SPEEDRESET.

If this flag is set, the speed limit will be reset to the limit as set by the last speedlimit sign. This can be used to reset any limit imposed by a specific signal aspect. Note that this does not overrule any speed limits set by another SPEED signal as those limits are processed as if set by a speedlimit sign.

Example :

```
SignalType ("SpeedSignal"
        SignalFnType ( SPEED )
        SignalLightTex ( "ltex" )
        SignalDrawStates ( 5
                SignalDrawState ( 0
                        "speed25"
                        )
                )
                SignalDrawState ( 1
                        "speed40"
                        )
                )
                SignalDrawState ( 2
                        "speed50"
                        )
                )
                SignalDrawState ( 3
                        "speed60"
                        )
                )
                SignalDrawState ( 4
                        "speed70"
                        )
                )
        )
        SignalAspects ( 5
                SignalAspect ( APPROACH_1       "speed25"       SpeedMPH ( 25 ) )
                SignalAspect ( APPROACH_2       "speed40"       SpeedMPH ( 40 ) )
```

```
                SignalAspect ( APPROACH_3       "speed50"       SpeedMPH ( 50 ) )
                SignalAspect ( CLEAR_1          "speed60"       SpeedMPH ( 60 ) )
                SignalAspect ( CLEAR_2          "speed70"       SpeedMPH ( 70 ) )
            )
            SignalNumClearAhead ( 2 )
    )
```

Notes :

- The SignalNumClearAhead value must be included to satisfy syntax but has no function.
- The actual speed can be set either using fixed aspect selection through user functions, or can be route linked.

  The actual use is defined in the related script and the related shape definition.

Example 2 :

```
SignalType ( "SpeedReset"
    SignalFnType ( SPEED )
    SignalLightTex ( "ltex" )

    SignalDrawStates ( 1
        SignalDrawState ( 0
            "Red"
        )
    )
    SignalAspects ( 1
        SignalAspect ( STOP     "Red"  signalflags (OR_SPEEDRESET) )
    )
    SignalNumClearAhead ( 2 )
)
```

This example resets the speed to the limit as set by the last speedsign, overruling any speed limits set by signal aspects.

10.16.2. Approach control functions

Approach control signals are used, specifically in the UK, to keep a signal at 'danger' until the train is within a specific distance ahead of the signal, or has reduced its speed to a specific value. Such control is used for diverging routes, to ensure the speed of the train is reduced sufficiently to safely negotiate the switches onto the diverging route.

For use in OR, two script functions have been defined which can be used to control the signal until the train has reached a specific position or reduced its speed.

These functions are :

- APPROACH_CONTROL_POSITION(position)
- APPROACH_CONTROL_SPEED(position, speed)

These functions are Boolean functions, returned value is 'true' if a train is approaching the signal and is within required distance of the signal and, for APPROACH_CONTROL_SPEED, has reduced its speed below the required values.

Parameters :

position : required distance of train approaching the signal, in meter

speed : required speed, in meter/sec.

Note that the speed is checked only when the train is within the defined distance.

Important note : although the script uses 'float' to define local variables, these are in fact all integers. This is also true for the values used in these functions : if direct values are used, these must be integer values.

The values may be set directly in the signalscript, either as variables or as numbers in the function call.

However, it is also possible to define the required limits in the sigcfg.dat file as part of the signal definition.

The syntax definition for this is :

ApproachControlLimits (<definitions>)

Allowed definitions :

- Position :
    - Positionm : position in meter.
    - Positionkm : position in kilometer.
    - Positionmiles : position in miles.
    - Positionyd : position in yards.

- Speed :
    - Speedkph : speed in km / hour.
    - Speedmph : speed in miles / hour.

These values are referenced in the script file using the following variable names :

- Approach_Control_Req_Position
- Approach_Control_Req_Speed

These variables must not be defined as floats etc., but can be used directly without prior definition.

Note that the values as defined in the sigcfg.dat file will be converted to meter and meter/sec and rounded to the nearest integer value.

Example

This example is a three-head search light signal, which uses Approach Control if the route is set to the 'lower' head.

Route selection is through 'dummy' DISTANCE type route-selection signals.

Signal definition :

```
SignalType ( "SL_J_40_LAC"
        SignalFnType ( NORMAL )
        SignalLightTex ( "bltex" )
        SigFlashDuration ( 0.5 0.5 )
        SignalLights ( 8
                SignalLight ( 0 "Red Light"
                        Position ( 0 6.3 0.11 )
                        Radius ( 0.125 )
                )
                SignalLight ( 1 "Amber Light"
                        Position ( 0 6.3 0.11 )
                        Radius ( 0.125 )
                )
                SignalLight ( 2 "Green Light"
                        Position ( 0 6.3 0.11 )
                        Radius ( 0.125 )
                )
                SignalLight ( 3 "Red Light"
                        Position ( 0 4.5 0.11 )
                        Radius ( 0.125 )
                )
                SignalLight ( 4 "Amber Light"
                        Position ( 0 4.5 0.11 )
                        Radius ( 0.125 )
                )
                SignalLight ( 5 "Green Light"
                        Position ( 0 4.5 0.11 )
                        Radius ( 0.125 )
                )
                SignalLight ( 6 "Amber Light"
                        Position ( 0 2.7 0.11 )
                        Radius ( 0.125 )
                )
                SignalLight ( 7 "White Light"
                        Position ( 0 2.7 0.11 )
                        Radius ( 0.125 )
                )
        )
        SignalDrawStates ( 8
                SignalDrawState ( 0
                        "Red"
                        DrawLights ( 1
                                DrawLight ( 0 )
                        )
                )
                SignalDrawState ( 1
                        "TopYellow"
                        DrawLights ( 1
                                DrawLight ( 1 )
                        )
                )
                SignalDrawState ( 2
                        "TopGreen"
                        DrawLights ( 1
```

```
                                DrawLight ( 2 )
                        )
                )
                SignalDrawState ( 3
                        "TopYellowMidGreen"
                        DrawLights ( 2
                                DrawLight ( 1 )
                                DrawLight ( 5 )
                        )
                )
                SignalDrawState ( 4
                        "MidYellow"
                        DrawLights ( 2
                                DrawLight ( 0 )
                                DrawLight ( 4 )
                        )
                )
                SignalDrawState ( 5
                        "MidGreen"
                        DrawLights ( 2
                                DrawLight ( 0 )
                                DrawLight ( 5 )
                        )
                )
                SignalDrawState ( 6
                        "LowYellow"
                        DrawLights ( 3
                                DrawLight ( 0 )
                                DrawLight ( 3 )
                                DrawLight ( 6 )
                        )
                )
                SignalDrawState ( 7
                        "LowWhite"
                        DrawLights ( 3
                                DrawLight ( 0 )
                                DrawLight ( 3 )
                                DrawLight ( 7 SignalFlags ( FLASHING ))
                        )
                )
        )
        SignalAspects ( 8
                SignalAspect ( STOP                             "Red" )
                SignalAspect ( STOP_AND_PROCEED                         "LowWhite" SpeedMPH(25) )
                SignalAspect ( RESTRICTING                      "LowYellow" SpeedMPH(25) )
                SignalAspect ( APPROACH_1                       "MidYellow" SpeedMPH(40) )
                SignalAspect ( APPROACH_2                       "TopYellowMidGreen" )
                SignalAspect ( APPROACH_3                       "TopYellow" )
                SignalAspect ( CLEAR_1                          "MidGreen" SpeedMPH(40) )
                SignalAspect ( CLEAR_2                          "TopGreen" )
        )

    ApproachControlSettings (
        PositionM  ( 500 )
        SpeedMpH   ( 10  )
    )

        SignalNumClearAhead ( 5 )
)
```

Signal function (reduced to show use of approach control only).

This function uses approach control for the 'lower' route.

```
////////////////////////////////////////////////////////////////////////////
SCRIPT SL_J_40_LAC

// Searchlight Top Main Junction
        extern float        block_state ();
        extern float        route_set ();
        extern float        def_draw_state ();
        extern float    next_sig_lr ();
        extern float    sig_feature ();
        extern float        state;
        extern float        draw_state;
        extern float        enabled;

//
// Returned states
// drawn :
//      SIGASP_STOP
//
//   Top Cleared :
//      SIGASP_APPROACH_3
//      SIGASP_APPROACH_2
//      SIGASP_CLEAR_2
//
//   Middle Cleared :
//      SIGASP_APPROACH_1
//      SIGASP_CLEAR_1
//
//   Lower Cleared :
//      SIGASP_RESTRICTING
//      SIGASP_STOP_AND_PROCEED
//

//
// User Flags
//
// USER1 : copy top approach
// USER2 : top approach junction
// USER3 : copy middle approach
// USER4 : no check block for lower
//

    float       clearstate;
    float       setstate;
    float       diststate;
    float       adiststate;
    float       nextstate;
    float       routestate;
    float       blockstate;

    blockstate = 0;
    clearstate = 0;
    routestate = 0;
    setstate   = 0;
    nextstate  = next_sig_lr(SIGFN_NORMAL);
    diststate  = next_sig_lr(SIGFN_DISTANCE);
    adiststate = diststate;

    if (diststate ==# SIGASP_CLEAR_1)
    {
```

```
      diststate = SIGASP_CLEAR_2;
    }
    if (diststate ==# SIGASP_APPROACH_1)
    {
      diststate = SIGASP_APPROACH_3;
    }

// get block state
   if (!enabled)
    {
       clearstate = -1;
    }

    if (block_state () ==# BLOCK_JN_OBSTRUCTED)
    {
       clearstate = -1;
    }

    if (block_state() ==# BLOCK_OCCUPIED)
    {
       blockstate = -1;
    }

// check if distant indicates correct route
    if (diststate ==# SIGASP_STOP)
    {
       clearstate = -1;
    }

// top route
    state = SIGASP_STOP;

    if (blockstate == 0 && clearstate == 0 && diststate ==# SIGASP_CLEAR_2)
    {
       // aspect selection for top route (not shown)
       .......
    }

// middle route
    if (blockstate == 0 && clearstate == 0 && diststate ==# SIGASP_APPROACH_3)
    {
       // aspect selection for middle route (not shown)
       .......
    }

// lower route
    if (blockstate == 0 && clearstate == 0 && diststate ==# SIGASP_RESTRICTING)
    {
       if (Approach_Control_Speed(Approach_Control_Req_Position, Approach_Control_Req_Speed))
       {
          state = SIGASP_RESTRICTING;
       }
    }

// Get draw state
            draw_state = def_draw_state (state);
```

10.16.3. TrainHasCallOn function

This function is intended specifically to allow trains to 'call on' in Timetable mode when allowed to do so as defined in the timetable.
The use of this function allows a train to 'call on' into a platform in Timetable mode without jeopardizing the functionality in normal Activity mode.

It is a Boolean function and returns state as follows :

- Activity Mode :
  - Returns true if :
    - Route from signal is not leading into a platform.
- Timetable Mode :
  - Returns true if :
    - Route from signal is not leading into a platform.
    - Route from signal is leading into a platform and the train has a booked stop in that platform, and either of the following states is true :
      - Train has $CallOn command set for this station.
      - Train has $Attach command set for this station and the train in the platform is the train which it has to attach to.
      - Train is part of RunRound command, and is to attach to the train presently in the platform.

The use of this function must be combined with a check for blockstate ==# BLOCK_OCCUPIED.

Note : this function must <u>NOT</u> be used in combination with blockstate ==# JN_OBSTRUCTED.
The state JN_OBSTRUCTED is used to indicate that the route is not accesible to the train (e.g. switch set against the train, opposite movement taking place etc.).
Some signal scripts allow signals to clear on blockstate ==# JN_OBSTRUCTED. This can lead to all kinds of incorrect situations.
These problems are not due to programming errors but to route signal script errors.

Example (part of script only) :

```
if (enabled && route_set() )
{
    if (block_state == #BLOCK_CLEAR)
    {
    // normal clear, e.g.
          state = #SIGASP_CLEAR_1;
    }
    else if (block_state == #BLOCK_OCCUPIED && TrainHasCallOn() )
    {
    // clear on occupied track and CallOn allowed
         state = #SIGASP_STOP_AND_PROCEED;
    }
    else
    {
    // track is not clear or CallOn not allowed
         state = #SIGASP_STOP;
    }
}
```

10.16.4. TrainHasCallOn_Restricted function

This function has been introduced because signals with call-on aspects can be used as entrance signals for stations, but also on 'free line' sections, that is away from stations.
TrainHasCallOn always allows call-on if the signal is on a 'free-line' section. This is to allow proper working for USA-type permissive signals.
Some signal systems however use these signals on sections where call-on is not allowed. For this case, the TrainHasCallOn_Restricted function has been introduced.

When approaching a station, both functions behave the same, but on 'free line' sections, the TrainHasCallOn_Restricted() will never allow call-on.

So, in a nutshell :

- Use on approach to stations :
    - TrainHasCallOn() and TrainHasCallOn_Restricted() :
        - Activity : call-on not allowed
        - Timetable : call-on allowed in specific situations (with $callon, $stable or $attach commands)
- Use on 'free line' :
    - TrainHasCallOn() :
        - Activity and Timetable : call-on always allowed
    - TrainsHasCallOn_Restricted() :
        - Activity and Timetable : call-on never allowed

10.16.5. How to lay down these signals on the route

These signals can be laid down with the MSTS RE. In the .tdb file only a reference to the SignalType name is written, an in the world file only a reference to the signal head is written. As these are accordingly to MSTS standards, no need to manually edit route files exists.

11. Timetable mode

11.1. Introduction

The timetable concept is not a replacement for the activity definition, but is an alternative way in defining both player and computer-controlled (AI and Static) trains.
In an activity, the player train is defined explicitly, and all AI trains are defined in a traffic definition. Static trains are defined separately.
In a timetable all trains are defined in a similar way. On starting a timetable run, the required player train is selected from the list of available trains. In the timetable definition itself, no distinction is made between running trains - all running trains can be selected as player train, and if not selected as such they will be run as AI trains. Static trains are also defined in the same way but cannot be selected as player train.
As a result, the number of different 'activities' that can be played using the same timetable file is equal to the number of trains which are defined in the timetable.

The development of the timetable concept is still very much a work in progress. This document details the state as it is at the moment, but also includes items yet to be produced, or items which have yet to be developed further.
To distinguish between these items, the following styles are used.
Items shown in black italics are available but only in a provisional implementation, or in a limited context. Further development of these items is still required.
Important aspects where use of specific OR or MSTS items for timetables differ significantly from its use in an activity are shown in bold.
Apart from the items indicated as above, it should be realised that as work continues, all items are still subject to change.

11.2. General

11.2.1. Data definition

The timetable data is defined in a Spreadsheet, and saved as a *.csv file (character separated file).
As separation character, either ',' (comma) or ';' (semi-colon) must be used.
Do not select space or tab as separation character.
As ';' or ',' are possible separation character, these symbols must not be used anywhere within the actual data. Enclosure of text by quotes (either single or double) has no effect.

11.2.2. File structure

The saved *.csv files must be renamed with extension *.timetable_or. The timetable files must be placed in a OpenRails subdirectory in the route's Activities directory.

11.2.3. File and train selection

When starting a timetable run, the required timetable file must be selected.
After selecting the required timetable, a list of all trains contained in that timetable is generated and the required train can be selected.
Season and weather can also be selected, these are not preset within the timetable definition.

11.3. Timetable definition

11.3.1. General

A timetable consists of a list of trains, and, per train, the required timing of these trains. The timing can be limited to just the start time, or it can include intermediate times as well.
At present, intermediate timings are limited to 'platform' locations as created using the MSTS Route Editor.
Each column in the spreadsheet contains data for a train, each row represents a location. A cell at the intersection of a train and location contains the timing data for that particular train at that location.
Special rows and columns can be defined for general information or control commands.
The first row for each column contains the train definition.
The first column for each row contains the location definition.
The cell at the intersection of the first row and first column **must be empty**.

This paragraph only lists the main outline, full detailed description will follow in the next paragraphs.

11.3.2. Column definitions

A column is defined by the contents of the first row.
Default, the first row defined the train name.
Special columns can be defined using the following syntax :

#comment	: column contains comment only and is ignored when reading the timetable
<blank>	: column is extension of preceding column

11.3.3. Row definitions

A row is defined by the contents of the first column.
Default, the first column defines the stop location.
Special columns can be defined using the following syntax :

#comment	: row contains comment only and is ignored when reading the timetable
<blank>	: row is extension of row above
#path	: defines train path
#consist	: defines train consist
#start	: defines time when train is started
#note	: defines general notes for this train
#dispose	: defines how train is handled after it has terminated

11.3.4. Timing details

Each cell which is at an intersection of a train column and a location row, can contain timing details for that train at that location.
Presently, only train stop details can be defined. Later on, passing times can also be defined, these

passing times can be used to determine a train's delay.
Control commands can be set at locations where the train stops, but can also be set for locations where no timing is inserted as the train passes through that location without stopping.

11.4. Timetable data details

11.4.1. Timetable description

Although #comment rows and columns are generally ignored, the contents of the cell at the intersection of the first #comment row and first #comment column is used as the timetable description.

11.4.2. Train details

The train name as defined in the first row must be unique for each train in a timetable file.
This name is also used when referencing this train in a train command, see details below.
The sequence of trains is not important.

11.4.3. Location details

At present, the possible locations are restricted to 'platforms' as defined in the MSTS Route Editor.
Each location must be set to the 'Station Name' as defined in the platform definitions.
The name used in the timetable must exactly match the name as used in the route definition (*.tdb file), otherwise the location can not be found and therefor not processed.
Also, each location name must be unique, as otherwise its position in the train path could be ambiguous.
The sequence of the locations is not important, as the order in which the stations are passed by a train is defined in that train's path. For the same reason, a train's path can be set to just run in between some of the locations, or be set to bypass certain stations.

11.4.4. Timing details

Each cell at an intersection of train and location can contain the timing details of that train at that location.
Times are defined as HH:mm, and the 24-hour clock must be used.
If a single time is inserted it is taken as the departure time (except at the final location).
If both arrival and departure time are to be defined, these must be separated by '-'.
Additional control commands can be included. Such commands can also be set for locations where the train does not stop and therefor has no timing details, but the train must pass through that location for the commands to be effective.
Although a location itself can be defined more than once in a timetable, it is not possible to define timing details for trains for a location more than once. If a train follows a route which takes it through the same location more than once, the train must be 'split' into separate train entries.

11.4.5. Special columns

- **#Comment column**

A column with the #comment definition in the first row is a comment column and is ignored when reading the timetable, except for the cell at the intersection of the first comment column and the first comment row.

- **<Blank> column**
  A column with a blank (empty) cell in the first row is taken as a continuation of the preceding column. It can be used to insert control commands which apply to the details in the preceding column. This can be useful when timings are derived automatically through formula in the spreadsheet as inserting commands in the timing cell itself would exclude the use of such formula.

11.4.6. Special rows

- **#Comment row**
  A row with the #comment definition in the first column is a comment row and is ignored when reading the timetable, except for the cell at the intersection of the first comment column and the first comment row.

- **<Blank> row**
  A row with a blank (empty) cell in the first column is taken as a continuation of the preceding row.

- **#Path row**
  The #path row defines the path of that train. The path must be a *.pat file as defined by the MSTS Activity Editor, and must be located in the route's Path directory. This field is compulsory.
  The timetable uses the same paths as defined for activities.
  However, waiting points must not be defined in paths for use in timetables as the processing of waiting points is not supported in the timetable concept.
  Waiting points within a timetable must be defined using the specific control commands.

- **#Consist row**
  The #consist row defined the consist used for that train. This field is compulsory.
  However, if the train is run as an AI train and it is 'formed' out of another train (see below), the consist information is ignored and the train uses the consist of the train out of which it was formed.
  For the player train, the consist is always used even if the train is formed out of another train.
  The consist definition must be a *.con file as defined by the MSTS Activity Editor, and must be stored in the defined consist directory.

- **#Start row**
  The #start row defines the time at which the train is started. It must be defined as HH:mm, and the 24 hour clock must be used. This field is compulsory.

  Use of start time for AI trains :

- When a train is formed out of another train and this other train is included to run in the timetable, the time defined in #start is only used to define when the train becomes active.

Use of start time for player train :

- The time as defined in #start is used as the start time of the 'activity'.

If a train is formed out of another train and this train is included in the timetable, then if this train is delayed and has not arrived before the defined start time, the starting of this train is also delayed until the train out of which it is formed has arrived. **This applies to both AI and player train.** This means that the start of the player activity can be delayed.
For details on starting and running of trains around midnight see special paragraph.

- **#Note row**
  The #note row can be used to defined control commands which are not location related but apply to the full run of the train. It can also be used to set commands for trains which do not stop at or pass through any defined location. This row is optional.

- **#Dispose row**
  The #dispose row defines what happens to an AI train when it has reached the end of its run, i.e. it has reached the end of the defined path.
  The information in the #dispose row can detail if the train is to be formed into another train, and, if so, how and where. For details see the commands as described further down.
  This row is optional and if included, the use per train is also optional.
  If the row is not included or the field is not set for a particular train, the train is removed from the activity after it has terminated.
  The #dispose row presently does not affect the end of the run for the player train.

11.4.7. Control commands

11.4.7.1. General

Control commands can be set to control train and signalling behaviour and actions.
There are four sets of commands available :

- Location commands
- Train control commands
- Create commands
- Dispose commands

11.4.7.2. Command syntax

All commands have the same basic syntax.
A command consists of :

- Syntax name : defines the control command

- Syntax value : set the value related to the command
  Not all commands take a value.
- Syntax qualifiers : adds additional information to the command
  Not all commands have qualifiers.
  Some qualifiers may be optional but others may be compulsory, or compulsory only in combination with other qualifiers.
- Syntax qualifier values : a qualifier may require a value

Command syntax :

$name = value /qualifier=value

Multiple values may be set, separated by '+'.
Note that any qualifiers always apply to all values.

11.4.7.3.TRAIN REFERENCE.

Many commands require a reference to another train.
This reference is the other train's name as defined in the first row.

11.4.7.4.LOCATION COMMANDS.

Location commands are :

- $hold
- $forcehold

These commands are also available as train control commands and are detailed in that paragraph.

11.4.7.5.TRAIN CONTROL COMMANDS.

All available train control commands are detailed below.
These commands can be set for each timing cell, i.e. at each intersection of train column and location row. The commands will apply at and from the location onward (if applicable).
Some commands can also be set in the #note row, in which case they apply from the start of the train. These commands are indicated by an asterisk (*) behind the command name.
The commands $hold and $nosignalwait can also be set as location commands.

$hold, $nohold and $forcehold

If $hold is set, it defines that the exit signal for that location must be held at danger up to 2 mins. before train departure.
An exit signal is allocated to a platform if this signal is beyond the end platform marker (in the direction of travel), but is still within the same tracknode - so there must not be any points etc. between the platform marker and the signal.
Default, the signal will not be held.
If set per location, it will apply to all trains, but can be overridden for any specific train by

defining $nohold in that train's column.
If set per train, it will apply to that train only.

$forcehold will set the first signal beyond the platform as 'hold' signal, even if this signal is not allocated to the platform as exit signal. This can be usefull at locations with complex layout where signals are not directly at the platform ends, but not holding the signals could lead to delay to other trains.

$callon

This will allow a train to 'call on' into a platform occupied by another train.
For full details, see paragraph on relation between signalling and timetable.

$connect

Syntax : $connect=<train> /maxdelay=n /hold=h

Defines that a train is to wait at a station until another train has arrived, so as to let passengers make the connection between the trains.
The train will be timetabled to allow this connection, and the $connect command is set to maintain this connection if the arriving train is running late.
Note that the $connect command will not lock the signal. If the paths of this train and the arriving train conflict before the arriving train reaches the station, additional $wait or $hold commands must be set to avoid deadlock.

Command value : reference to train which is to be waited for, this is compulsory.

Command qualifiers :

/maxdelay=n : n is the maximum delay (**in minutes**) of the arriving train for which this train is held.
If the delay of the arriving train exceeds this value the train will not wait.
The maximum delay is independent from this train's own delay.
This qualifier and its value are compulsory.

/hold=h : h is the time (**in minutes**) the train is still held after the other train has arrived, and relates to the time required by the passengers to make the connection.
This qualifier and its value are compulsory.

$wait *

Syntax : $wait=<train> /maxdelay=n /notstarted

Defines that a train is to wait for the referenced train to allow this train to proceed first.
The referenced train can be routed in the same or the opposite direction as this train itself.
A search is done for the first track section which is common to both trains, starting at the location where the $wait is defined, or at the start of the path if defined in the #note row.
If the start location is already common for both trains, then first a search is done for the first

section which is not common to both trains, and the wait is applied to the next first common section beyond that.
If the wait is set, the section will not be cleared for this train until the referenced train has passed this section. This will force the train to wait. The referenced train must exist for the wait to be valid.
However, if /notstarted is set, the wait will also be set even if the referenced train has not yet been started. This can be used where the wait position is very close to the start position of the referenced train, and there is a risk that the train may clear the section before the referenced train is started.

Care should be taken when defining a $wait at a location where the train is to reverse. As the search is performed for the active subpath only, a $wait defined at a location where the train is to reverse will not be effective as the common section will be in the next subpath after the reversal. In such a situation, the train should be 'split' into two separate definitions, one up to the reversal location and another starting at that location.

Command value : referenced train, this is compulsory.

Command qualifiers :

/maxdelay=n : n is the maximum delay (**in minutes**) of the referenced train for which the wait is still valid.
This delay is compensated for any delay of the train which is to wait, e.g. if maxdelay is 5 minutes, the referenced train has a delay of 8 minutes but this train itself has a delay of 4 minutes, the compensated delay is 4 minutes and so the wait is still valid.
This parameter is optional, if not set a maxdelay of 0 minutes is set as default.

/notstarted : the wait will also be applied if the referenced train has not yet started.

$follow *

Syntax : $follow=<train> /maxdelay=n

This command is very similar to the $wait command, but in this case it is applied to each common section of both trains beyond a part of the route which was not common.
The train is controlled such that at each section where the paths of the trains rejoin after a section which was not common, the train will only proceed if the referenced train has passed that position. The command therefor works as a $wait which is repeated for each such section.
The command can only be set for trains routed in the same direction.
When a wait location is found and the train is due to be held, a special check is performed to ensure the rear of the train is not in the path of the referenced train or, if it is, the referenced train has already cleared that position. Otherwise, a deadlock would result, with the referenced train not being able to pass the train which is waiting for it.

Command value : referenced train, this is compulsory.

Command qualifiers :

/maxdelay=n : n is the maximum delay (**in minutes**) of the referenced train for which the wait is still valid.
This delay is compensated by any delay of the train which is to wait, e.g. if maxdelay is 5 minutes, the referenced train has a delay of 8 minutes but this train itself has a delay of 4 minutes, the compensated delay is 4 minutes and thus the wait is still valid.
This parameter is optional, if not set a maxdelay of 0 minutes is set as default.

$waitany *

Syntax : $waitany=<path> /both

This command will set a wait for any train which is on the path section as defined.
If the qualifier /both is set, the wait will be applied for any train regardless of its direction, otherwise the wait is set only for trains heading in the same direction as the definition of the path.
The path defined in the waitany command must have a common section with the path of the train itself, otherwise no waiting position can be found.
This command can be set to control trains to wait beyond the normal signal or deadlock rules. For instance, it can be used to perform a check for a train which is to leave a siding or yard, checking the line the train is to join for any trains approaching on that line, for a distance further back than signalling would normally clear, so as to ensure it does not get into the path of any train approaching on that line.
With the /both qualifier set, it can be used at the terminating end of single track lines to ensure a train does not enter that section beyond the last passing loop if there is another train already in that section as this could lead to irrecoverable deadlocks.

11.4.7.6.DISPOSE COMMANDS.

Dispose commands can be set in the #dispose row to define what is to be done with the train after is has terminated.
See special notes below on the behaviour of the player train when it is formed out of another train by a dispose command, or when the player train itself has a dispose command.

$forms

Syntax : $forms=<train> /runround=<path> /rrime=time /setstop

$forms defines which new train is to be formed out of this train when the train terminates.
The consist of the new train is formed out of the consist of the terminating train and any consist definition for the new train is ignored.
The new train will be 'static' until the time as defined in #start row for that train. This means that the new train will not try to clear its path, signals etc., and will not move even if it is not in a

station.
If the incoming train is running late, and its arrival time is later as the start time of the new train, the start of the new train is also delayed but the new train will immediately become active as soon as it is formed.
For loco hauled trains, it can be defined that the engine(s) must run round the train in order for the train to move in the opposite direction. The runround qualifier needs a path which defines the path the engine(s) is to take when performing the runround. If the train has more than one leading engine, all engines will be run round. Any other power units within the train will not be moved.
For specific rules and conditions for runround to work, see paragraph on relation between signalling and timetable concept.
If runround is defined, the time at which the runround is to take place can be defined. If this time is not set, the runround will take place immediately on termination of the incoming train.

Command value : referenced train, this is compulsory.

Command qualifiers :

/runround=<path> : <path> is the path to be used by the engine to perform the runround. This qualifier is optional; if set, the value is compulsory.

/rrtime=time : time is the definition of the time at which the runround is to take place. The time must be defined in HH:mm and must use the 24 hour clock. This qualifier is only valid in combination with the /runround qualifier, is optional but if set, the value is compulsory.

/setstop : if this train itself has no station stops defined but the train it is to form starts at a station, this command will copy the details of the first station stop of the formed train, to ensure this train will stop at the correct location.
For this qualifier to work correctly, the path of the incoming train must terminate in the platform area of the departing train.
This qualifier is optional and takes no values.

$triggers

Syntax : $triggers=<train>

$triggers also defines which new train is to be formed out of this train when the train terminates.
However, when this command is used, the new train will be formed using the consist definition of the new train and the existing consist is removed.

Command value : referenced train, this is compulsory.

$static

Syntax : $ static

The train will become a 'static' train after it has terminated.

Command value : none.

$stable

Syntax : $stable /out_path=<path> /out_time=time /in_path=<path> /in_time=time /static /runround=<path> /rrtime= time /rrpos=<runround position> /forms=<train> /triggers=<train>

$stable is an extended form of either $forms, $triggers or $static, where the train is moved to another location before the related command is performed. In case of /forms or /triggers, the train can move back to the same or to another location where the new train actually starts. Note that in these cases, the train has to make two moves, outward and inward.
A runround can be performed in case /forms is defined.
If /triggers is defined, the change of consist will take place at the 'stable' position. Any reversal(s) in the inward path, or at the final inward position, are taken into account when the new train is build, such that the consist is facing the correct direction when the new train is formed at the final inward position.
The $stable can be used where a train forms another train but when the train must clear the platform before the new train can be formed to allow other trains to use that platform. It can also be used to move a train to a siding after completing its last duty, and be 'stabled' there as static train.
Separate timings can be defined for each move; if such a time is not defined, the move will take place immediately when the previous move is completed.
If timings are defined, the train will be 'static' after completion of the previous move until that required time.

If the formed train has a valid station stop and the return path of the stable command (in_path) terminates in the area of the platform of the first station stop of the formed train, the 'setstop' check (see setstop qualifier in $forms command) will automatically be added.

Command value : none.

Command qualifiers :

/out_path=<path> : <path> is the path to be used by the train to move out to the 'stable' position. The start of the path must match the end of the path of the incoming train.

/out_time = time : time definition when the outward run must be started. Time is defined as HH:mm and must use the 24 hour clock.

/in_path=<path> : <path> is the path to be used by the train for the inward run from the 'stable' position to the start of the new train.

The start of the path must match the end of the out_path, the end of the path must match the start of the path for the new train.

/in_time = time : time definition when the inward run must be started.
Time is defined as HH:mm and must use the 24 hour clock.

/runround=<path> : <path> is the path to be used by the engine to perform the runround.
For details, see the $forms command.

/rrtime=time : time is the definition of the time at which the runround is to take place.
The time must be defined in HH:mm and must use the 24 hour clock.

/rrpos = <runround position> : the position within the 'stable' move at which the runround is to take place.
Possible values :
- out : the runround will take place before the outward move is started.
- stable : the runround will take place at the 'stable' position.
- in : the runround will take place after completion of the inward move.

/static : train will become a 'static' train after completing the outward move.

/forms=<train> : train will form the new train after completion of the inward move. See the $forms command for details.

/triggers=<train>: train will trigger the new train after completion of the inward move. The train will change to the consist of the new train at the 'stable' position.
See the $triggers command for details.

Use of command qualifiers :

In combination with /static :

/out_path	: compulsory
/out_time	: optional

In combination with /forms :

/out_path	: compulsory
/out_time	: optional
/in_path	: compulsory
/in_time	: optional
/runround	: optional
/rrtime	: optional, only valid if /runround is set
/rrpos	: compulsory if /runround is set, otherwise not valid

In combination with /triggers :

/out_path	: compulsory
/out_time	: optional
/in_path	: compulsory
/in_time	: optional

11.5. ADDITIONAL NOTES.

11.5.1. TIMETABLE COMMANDS AND PLAYER TRAIN.

11.5.1.1.TERMINATION OF A TIMETABLE RUN.

On reaching the end of a timetabled run, the program will not be terminated automatically but has to be terminated by the player.

11.5.2. SIGNALLING REQUIREMENTS AND TIMETABLE CONCEPT.

11.5.2.1.General.

The timetable concept is more demanding of the performance of the signalling system than 'normal' activities. The main reason for this is that the timetable will often have AI trains running in both directions, including trains running ahead of the player train in the same direction as the player train. There are very few activities with such situations as no effort would of course be made to define trains in an activity which would never be seen, but also because MSTS could not always properly handle such a situation.
Any flaws in signalling, e.g. signals clearing the path of a train too far ahead, will immediately have an effect on the running of a timetable.
If signals clear too far ahead on a single track line, for instance, it means trains will clear through passing loops too early, which leads to very long waits for trains in the opposite direction. This, in turn, can lead to lock-ups as multiple trains start to converge on a single set of passing loops.
Similar situations can occur at large, busy stations - if trains clear their path through such a station too early, it will lead to other trains being kept waiting to enter or exit the station.
If 'forms' or 'triggers' is used to link reversing trains, the problem is exacerbated as any delays for the incoming train will work through on the return working.

11.5.2.2.Call On signal aspect.

Signalling systems may allow a train to 'call on', i.e. allow a train onto a section of track already occupied by another train (also knows as permissive working).
The difference between 'call on' and 'permissive signals' (STOP and PROCEED aspects) is that the latter is also allowed if the train in the section is moving (in the same direction), but 'call on' generally is only allowed if the train in the section is at a standstill.
When a signal allows 'call on', AI trains will always pass this signal and run up to a pre-defined distance behind the train in the section.
In station areas, this can lead to real chaos as trains may run into platforms occupied by other trains

such that the total length of both trains far exceeds the platform length, so the second train will block the 'station throat' stopping all other trains. This can easily lead to a complete lock-up of all traffic in and around the station.
To prevent this, calling on should be blocked in station areas even if the signalling would allow it. To allow train to call on when this is required in the timetable, the $callon command must be set which overrules the overall block. This applies to both AI and player train
In case the train is to attach to another train in the platform, calling on is automatically set.
Because of the inability of AI trains in MSTS to stop properly behind another train if 'called on' onto an occupied track, most signalling systems do not support 'call on' aspects but instead rely on the use of 'permission requests'. AI trains cannot issue such a request, therefore in such systems $callon will not work.
In this situation, attach commands can also not work in station areas.
Note that the 'runround' command also requires 'call on' ability for the final move of the engine back to the train to attach to it. Therefor, when performed in station areas, also the runround can only work if the signalling supports 'call on'.

Special signalling functions are available to adapt signals to function as described above, which can be used in the scripts for relevant signals in the sigscr file.
The function "TRAINHASCALLON()" will return 'true' if the section beyond the signal upto the next signal includes a platform where the train is booked to stop, and the train has the 'callon' flag set.
This function will also return 'true' if there is no platform in the section beyond the signal.
The function "TRAINHASCALLON_RESTRICTED" returns 'true' in similar conditions, except it always returns 'false' if there is no platform in the section beyond the signal.
Both functions must be used in combination with BLOCK_STATE = BLOCK_OCCUPIED.

11.5.2.3.Wait commands and Passing paths.

From the location where the 'wait' or 'follow' is defined, a search is made for the first common section for both trains, following on from a section where the paths are not common.
However, on single track routes with passing loops where 'passing paths' are defined for both trains, the main path of the trains will run over the same tracks in the passing loops and therefor no not-common sections are found. As a result, the waiting point cannot find a location for the train to wait and therefor will not work.
If waiting points are used on single track lines, the trains must have their paths running over different tracks through the passing loop in order for the waiting points to work properly.
It is a matter of choice by the timetable creator to either preset passing locations using the wait commands, or let the system work out the passing locations using the passing paths.

11.5.2.4.Wait commands and Permissive signals.

The 'wait' and 'follow' commands are processed through the 'blockstate' of the signal control.
If at the location where the train is to wait permissive signals are used, and these signals allow a 'proceed' aspect on blockstate JN_OBSTRUCTED, the 'wait' or 'follow' command will not work as the train will not be stopped.

11.5.2.5.Running trains around midnight.

A timetable can be defined for a full 24 hour day, and so would include trains running around midnight.

The following rules apply for the player train :

- Train booked to start before midnight will be started at the end of the day, but will continue to run if terminating after midnight.
- Trains formed out of other trains starting before midnight will NOT be started if the incoming train is delayed and as a result the start time is moved after midnight. In this situation, the activity is aborted.
- Trains booked to start after midnight will be started at the beginning of the day.

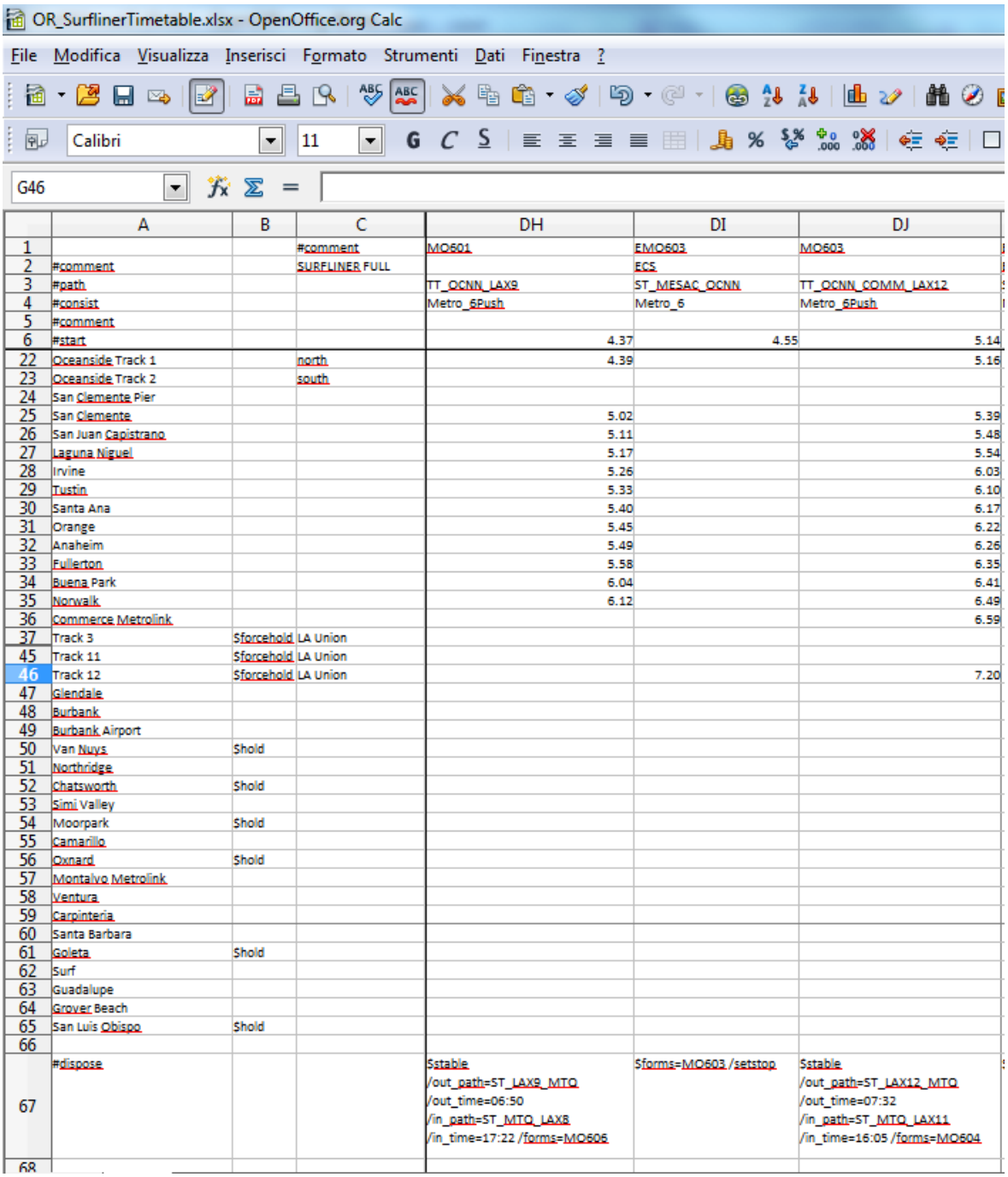
The following rules apply for AI trains :

- Trains booked to start before midnight will be started at the end of the day, but will continue to run if terminating after midnight.
- Trains formed out of other trains starting before midnight will still be started if the incoming train is delayed and as a result the start time is moved after midnight.
- Trains booked to start after midnight will be started at the beginning of the day.

As a result of these rules, it is not really possible to run an activity around or through midnight with all required AI trains included.

11.6. Example of a timetable file

Here is an excerpt of a timetable file:



	A	B	C	DH	DI	DJ
1			#comment	MO601	EMO603	MO603
2	#comment		SURFLINER FULL		ECS	
3	#path			TT_OCNN_LAX9	ST_MESAC_OCNN	TT_OCNN_COMM_LAX12
4	#consist			Metro_6Push	Metro_6	Metro_6Push
5	#comment					
6	#start			4.37	4.55	5.14
22	Oceanside Track 1		north	4.39		5.16
23	Oceanside Track 2		south			
24	San Clemente Pier					
25	San Clemente			5.02		5.39
26	San Juan Capistrano			5.11		5.48
27	Laguna Niguel			5.17		5.54
28	Irvine			5.26		6.03
29	Tustin			5.33		6.10
30	Santa Ana			5.40		6.17
31	Orange			5.45		6.22
32	Anaheim			5.49		6.26
33	Fullerton			5.58		6.35
34	Buena Park			6.04		6.41
35	Norwalk			6.12		6.49
36	Commerce Metrolink					6.59
37	Track 3	$forcehold	LA Union			
45	Track 11	$forcehold	LA Union			
46	Track 12	$forcehold	LA Union			7.20
47	Glendale					
48	Burbank					
49	Burbank Airport					
50	Van Nuys	$hold				
51	Northridge					
52	Chatsworth	$hold				
53	Simi Valley					
54	Moorpark	$hold				
55	Camarillo					
56	Oxnard	$hold				
57	Montalvo Metrolink					
58	Ventura					
59	Carpinteria					
60	Santa Barbara					
61	Goleta	$hold				
62	Surf					
63	Guadalupe					
64	Grover Beach					
65	San Luis Obispo	$hold				
66						
67	#dispose			$stable /out_path=ST_LAX9_MTQ /out_time=06:50 /in_path=ST_MTQ_LAX8 /in_time=17:22 /forms=MO606	$forms=MO603 /setstop	$stable /out_path=ST_LAX12_MTQ /out_time=07:32 /in_path=ST_MTQ_LAX11 /in_time=16:05 /forms=MO604
68						

11.7. Macro tool to ease generation of timetable spreadsheets

A very useful macro tool can be downloaded from here (free registration to Elvas Tower Forum needed).

12.Open Rails Multi-Player

12.1. Goal

The Multi-Player mode implemented in this stage is intended for friends to play OR together, each assuming the role of a train engineer operating a train. There is a built-in way to compose and send text messages, but there is no built-in tool for chatting, thus players are encouraged to use Ventrillo, Skype, MSN, Yahoo, Teamspeak or other tools to communicate vocally.

The current release utilizes a peer-to-peer mode, thus each player must start and run OR on their computer. A special server was deployed so you may not need to set up a server from your own computer.

12.2. Getting Started

One player starts as the server, and then others connect in as clients. Each player will choose and operate their own consist (and locomotive), but also can jump to watch others' consists, or couple with others to work as lead and DPU through a tough route, or even act as a dispatcher to control signals and switches manually.

12.3. Requirements

The server can start an activity or choose to explore. Clients MUST choose to explore (or a simple activity with timetable but no AI trains).

The client must select the same route played by the server.

It is *not* required for everyone to have the same set of paths, rolling stocks and consists.

12.4. Technical Issues

If you start the server at home, it will be necessary for you to learn your public IP address. You may also need to configure your router for port forwarding. Details to accomplish these are given in sections that follow.

It is recommended that you do not run a server for a prolonged period as the code has not been tightened for security. Only tell people you trust that you have a server started.

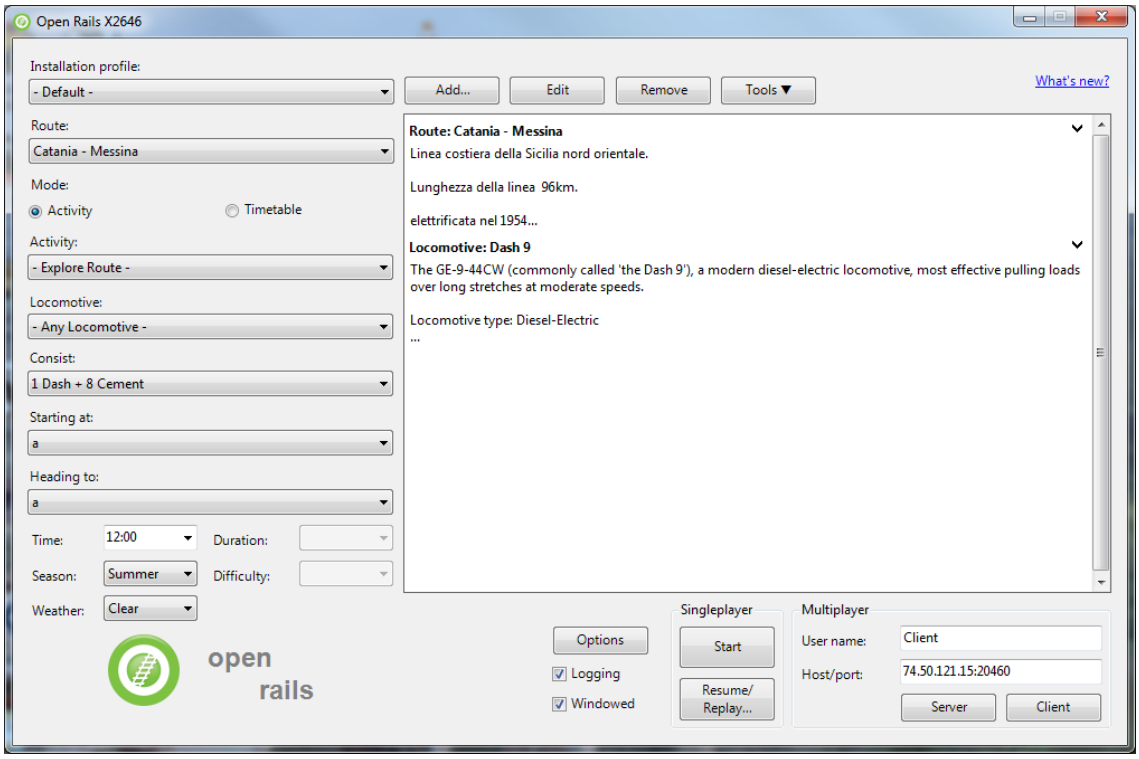
12.5. Technical Support

You can ask questions in the following forums: trainsim.com, elvastower.com, uktrainsim.com, etc.

A web forum has been set for you to post questions and announce servers. You can also request a private club so that only your friends know of your server. The forum is free to join and post: http://www.tsimserver.com/forums

12.6. Starting a Multi-Player session

12.6.1. Starting as server



On the OR main menu you select in a standard way as described in the "Getting started" chapter on the left side Route, activity or explore route, and in case of explore route you select as usual locomotive, consist, path, time, season and weather.

On the lower right side you enter your User Name and the host and port address. If you want to run as standalone server, or if you want to have more than instance of OR running in MP mode on the same computer, you must set Host/port to 127.0.0.1:30000. 30000 is the default port, but you can change to any integer between 10000 and 65536.

If you want to run in a local area network usually valid host addresses are 192.168.1.2 or 192.168.1.1.

After having inserted the Username and Host/port data you click on "Server"

When server starts, Windows Firewall may ask if you want to allow OR access to the Internet. If so, click **Allow**. If you use other firewall software, you may need to configure it to allow OpenRails to access the Internet.

There is no built-in limit of how many players can connect; a server with good Internet upload bandwidth can be expected to handle at least 10 client connections.

12.6.2. Starting as client

On the left side of the main menu you must enter only route, path and consist. The other parameters are received from the server.
On the right side you enter your username, IP address and port of the server, and click on "Client"

12.7. In-Game Controls:

1. Once the server and clients start and connect, MultiPlayer status will be shown on the left of the screen. You can watch how many players and trains are present and how far away you are from others. You can also look if you are acting as dispatcher (the server always is the dispatcher) or as client.
2. A player joined will have the same weather, time and season as the server, no matter what are the original choices.
3. The player train may join the world and find it is inside another train. Don't panic, you have two minutes to move your train out before OR thinks you want to couple with that train.
4. AI trains are added by the server and broadcast to all players. As a client, do not start an activity with AI trains; moreover it is recommended that you start in Explore mode on the client.
5. You can jump to see other trains in sequence by pressing **Alt+9**. OpenRails will cycle through all trains active on the server with each key press. As some trains may be far away, OpenRails may need a few seconds to load the surrounding scenery. Thus you may temporarily see a blank screen. You can press **F7** to see train names. You can press **9** to return to seeing your own train.
6. Locations of trains from other players are sent over the Internet. Because Internet routings vary moment to moment there may be some lag, and trains may jump a bit as OpenRails tries to update the locations with information received.

7. You can couple/decouple as usual. As coupling is controlled in the server, a player needs to drive slowly so that the server will have accurate information of train positions. If two player trains couple together, one of them will become a helper, and a message will be shown on the left indicating that the player is in Helper mode. A player in Helper mode cannot control their consist as it falls under control of the lead locomotive. By pressing Shift+E you can swap Helper status with another player on the train. Always press \ and Shift+/ to reset brakes each time after coupling/uncoupling.

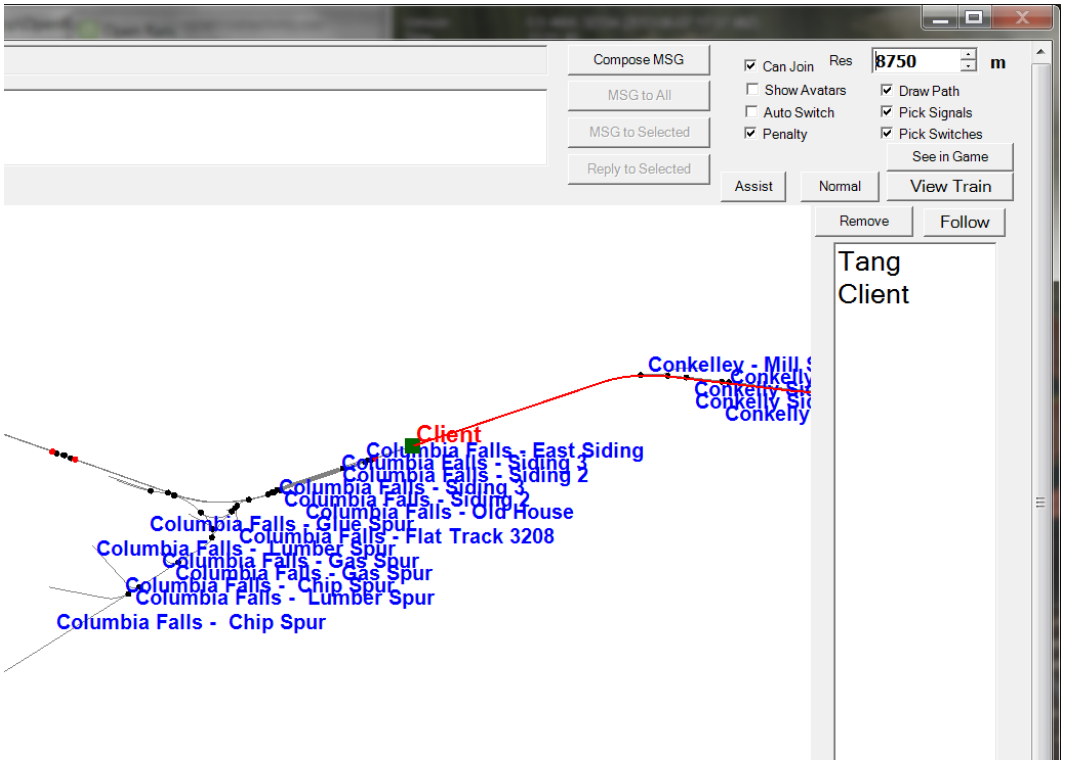
8. Players can uncouple their own trains. Players in the uncoupled trains may need to press

Shift+E to gain control; otherwise, the uncoupled trains may become a loose consist. Always stop completely before uncoupling, otherwise weird things may happen. **Players may also need to press keys for resetting brake state after uncoupling (see** here**).**

9. Players can throw switches by pressing **G** or **Shift-G** , and the switch state will change for all players on the server. The server has a choice to disallow clients to throw switches manually.
10. Both switches and signals are synchronized through the server (default every 10 seconds).
11. Player actions, such as sounding the horn or bell, turning on or off headlights, moving the pantograph up and down, opening and closing doors, moving the mirrors are broadcast to other players. Currently only the player controlled train has the cone of light shown.
12. A separate window showing the route, signals and trains can be activated by pressing **Ctrl+9**. By default, it is minimized and you must click on it on the Taskbar to make it active. You can hide it by pressing **Ctrl+9** again or by pressing **Esc** when that window has the focus. This window is an extended version of the Dispatcher Window.

You can zoom in and out by rotating the mouse wheel, or by holding both the left and right mouse

button and moving the mouse (if you do not have a mousewheel). You can hold shift key while click the mouse in a place in the map, which will quickly zoom in with that place in focus. You can hold Ctrl while click the mouse in a place in the map, which will zoom out to show the whole route. Holding Alt and click will zoom out to show part of the route.



A red line will be drawn for each train so you can find its intended path.

You can select a train either by clicking on the name in the right bar, or in the map by clicking the green train body. After that, you can click the “Remove” button to delete that train from the game.

You can pan the window by dragging it with the left mouse button.

One can click a switch (or signal) and press **Ctrl+Alt+G** to jump to that switch with the free-roam camera.

The Dispatcher player can click a switch (black dot) and choose “Main Route” or “Side Route” to switch. They can also click on a signal signal (green, red or orange dot) and choose to change the light.

The Dispatcher can choose a player and give the player right to throw switches and change signals, by clicking the button “Assist”. The right can be revoked by click the “Normal” button.

The Dispatcher can choose a player from the avatar list and remove that player from the game.

You can send a text message by typing in the top left text input area, and view the most recent 10 messages from the viewing area. One can send message to all after finishing it, or select some avatars and send a message to those selected.

12.8. Summation

1. Server can start an activity or Explore. Clients must choose to Explore the route or start with an activity without AI trains.

2. Missing rolling stock in other players' consists will be automatically replaced by existing cars from local directory.
3. You have two minutes after joining the game to move your train out of other trains.
4. Use **Alt+9** to see other trains, **9** to see your own train, **Ctrl+9** to view/hide the dispatcher window. Use the mouse wheel to zoom and left mouse button to pan the dispatcher window.
5. We can send and read messages from the dispatcher window
6. Use **Ctrl+Alt+F11** to see the path trains will follow, and **F7** to see train names
7. Move trains slowly when trying to couple.
8. Use **** and **Shift+/** (on English keyboards) just after your train is coupled or uncoupled, or when you just gain back the control of your own train.
9. Use **Shift+E** to gain control of your own train after uncoupling.
10. Use other communication tools (such as Ventrillo or Skype) to communicate with other players.
11. Always completely stop before uncoupling trains with two players coupled together

12.9. Possible Problems

1. A server may not be able to listen on the port specified. Restart the server and choose another port.
2. If you cannot connect to the server, verify sure you have the correct IP address and port number, and that the server has the port opened.
3. If other player have rolling stock you do not have, that train will automatically replace cars from your own folder, and this replacement may make the consist "interesting".
4. You may join the game and see you've selected the same start point as someone else and that your train is inside another train. Move the trains apart within two minutes and it will be fine.
5. If your train is moving too quickly when trying to couple the process may not work and weird things can happen.
6. As the server has absolute control, clients may notice the switch just changed will be changed back a few seconds later if the server controlled train wants to pass it.
7. Coupling/uncoupling the same set of trains may end up with weird things.
8. **Ctrl+E** locomotive switch may have train cars flipped.

12.10. Using the Public Server

A special public server is deployed so that you do not need to use your own computer as the server, avoiding the setup problems you may encounter. You can find the IP and port numbers here .

To connect to this public server you must act as described here, using IP and port numbers as found on the above link, with only a difference: the first player entering the session has to enter by clicking on “Client” and not on “Server”, even if he intends to be the dispatcher. If the port has no player yet, whoever connects first will be declared the dispatcher, others connected later will be normal players.

The public server runs a special code that is not part of OR. If you plan to run such a server for free, please contact the email listed in http://tsimserver.com/forums/showthread.php?2560.

12.10.1. Additional info on using the Public Server

1. If the computer of the player acting as dispatcher crashes or if the connection with it breaks down, the public server will try to appoint another player as dispatcher. Such player will receive on his monitor the following message: “`You are the new dispatcher. Enjoy!`”
2. If a client crashes or looses the connection, its position is held by the server for about two minutes. If the client re-enters the game within such time frame, it will re-enter the game in the position where he was at the moment of the crash.

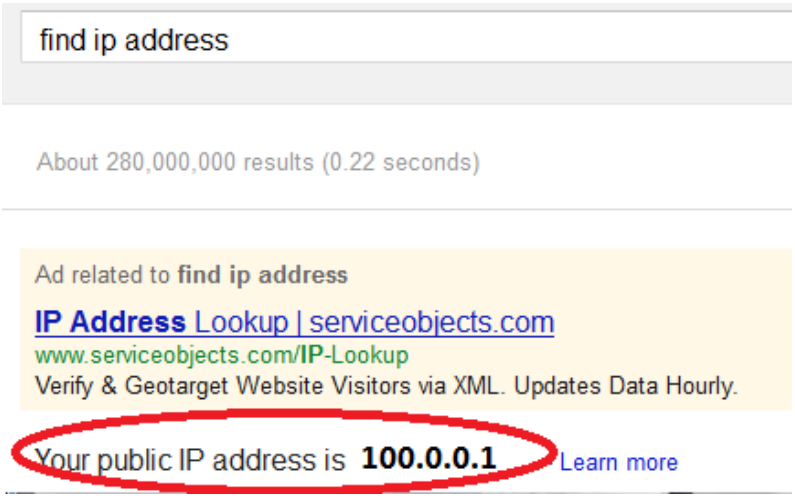
13. Multi-Player Setting up a Server from Your Own Computer

As any online game, you need to do some extra work if you want to host a multiplayer session.

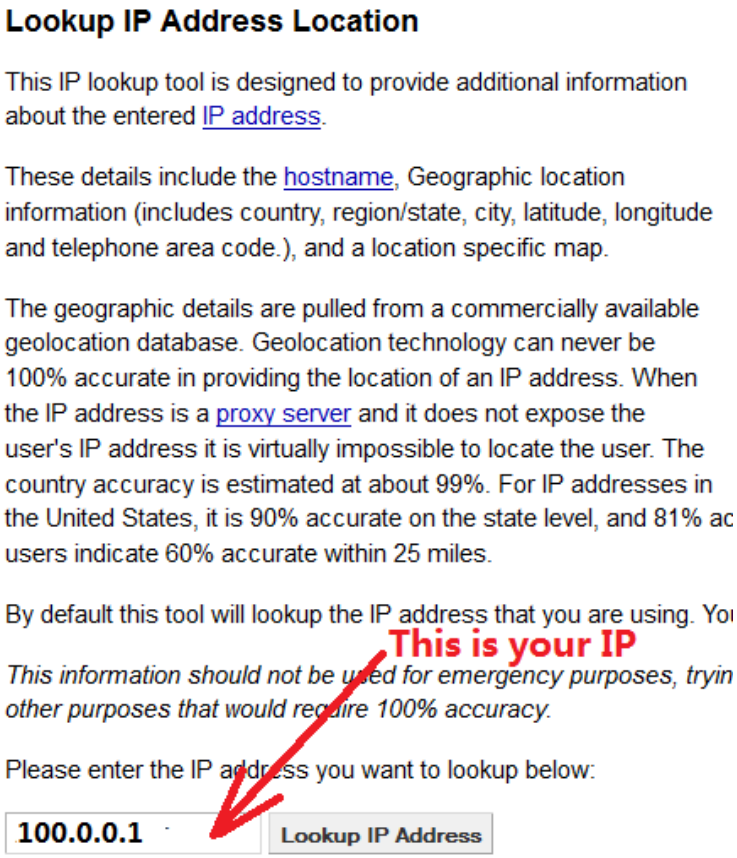
13.1. IP Address

If you are running at home and use a router, you may not have a permanent IP. Thus before you start as a server, you must find your IP. The quickest ways are the following:

1. Using Google: type in “find ip address”, then Google will tell you



2. If the above does not work, try http://whatismyipaddress.com/ip-lookup/, which shows your IP in the middle of the page

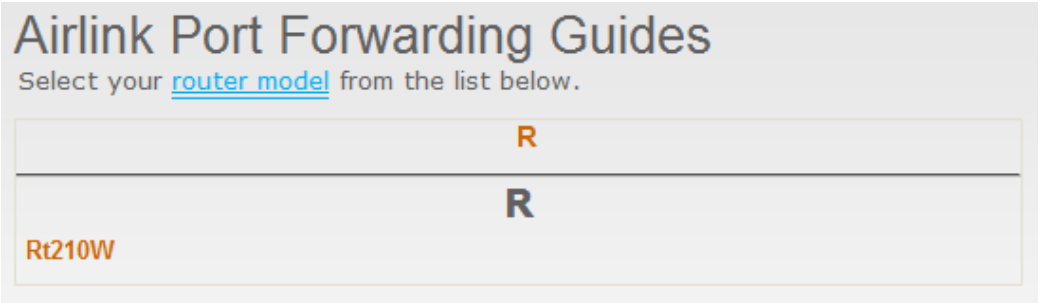


13.2. Port Forwarding

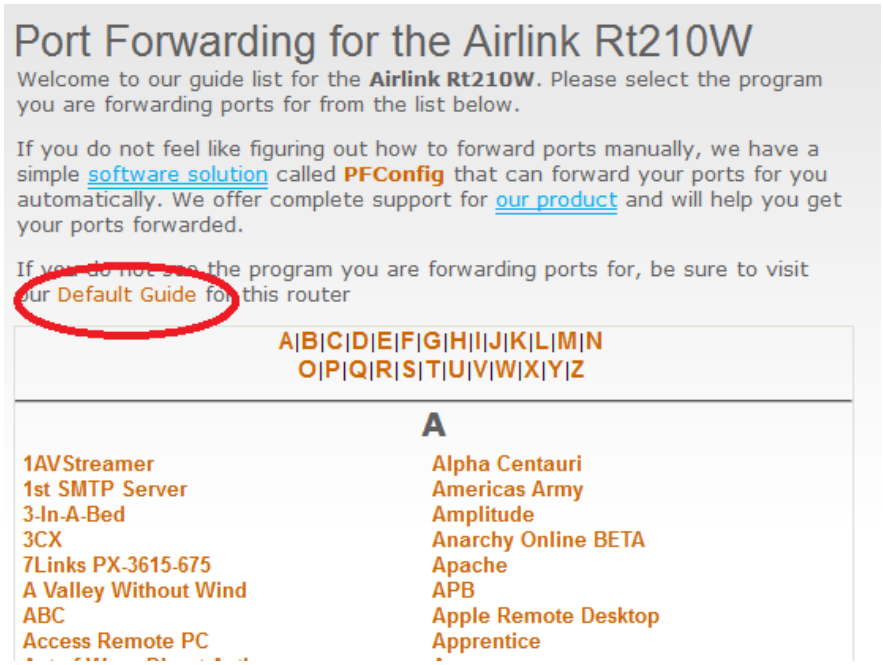
If you are using a router at home with several computers, your router needs to be told which computer on your home network should receive the network data OpenaRails needs. This is done by enabling Port Forwarding on the router. The default port OpenRails uses is 30,000. If you change that port number in the game you'll need to change the forwarded port number in the router as well. Your router must be told to forward data arriving from the internet on the correct port to the network IP address of the computer running OpenRails. For more information on Network Address Translation (NAT) and how Port Forwarding works, see this site: http://www.4remotesupport.com/4content/remote_support_NAT.html Here The following are the steps:

1. Go to http://portforward.com/english/routers/port_forwarding/, which contains a lot ads - just focus on the center of this page.
2. Locate the name of the manufacturer of your router, i.e. Airlink and click it:

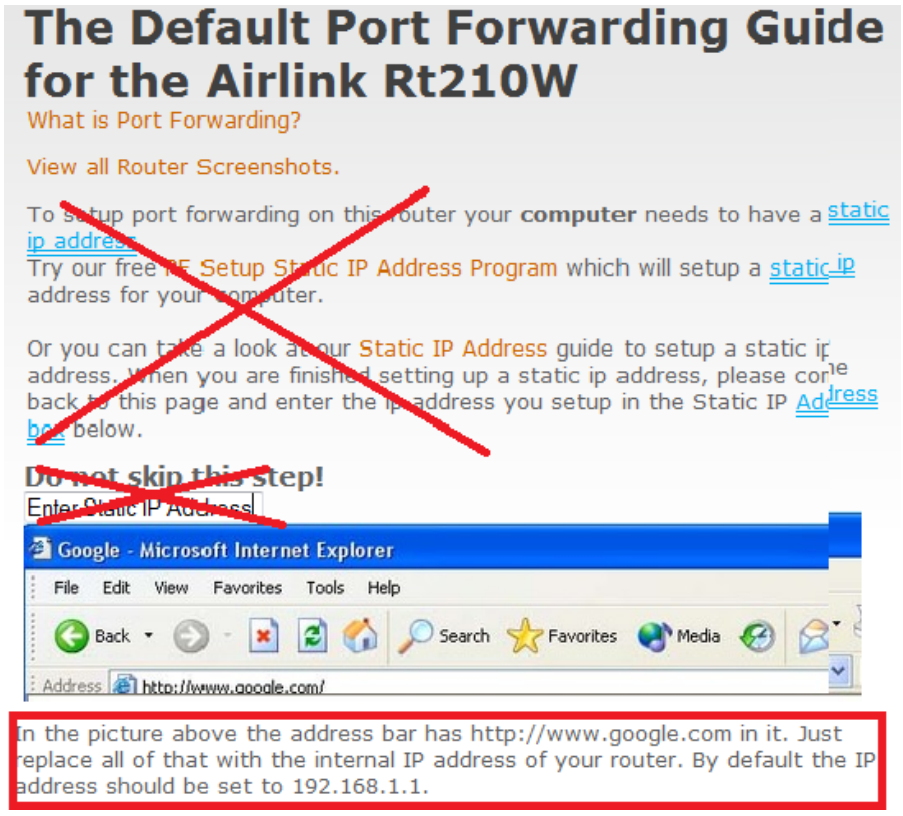
3. A page may appear allowing you to select your specific model of router:



4. It then shows all the programs (games) that you want to forward ports. Just click "Default Guide":



5. A page like the following should appear. Ignore the part crossed-out but pay special attention to the part enclosed in red:



Then follow the steps listed on the screen. Remember you want to forward port 30,000 by default, but if you change that you'll have to forward the correct port.

If you still cannot get others connecting to your computer, please go to www.tsimserver.com/forums and ask questions.

14.Open Rails sound management

14.1. OR vs. MSTS sound management

OR executes .sms files with a very high MSTS compatibility degree.

14.2. .sms instruction set

OR recognizes and manages the whole MSTS .sms instruction set, in general in a way compatible to MSTS. Differences are described here below.

The Activation () instruction behaves differently from MSTS regarding to cameras (CabCam, ExternalCam and PassengerCam): in general OR does not consider what cameras are explicitly activated within the .sms files. Instead, it uses a sort of implicit activation, that as a general rule works as follows:

2. when being in an inside view (cabview or passenger view) the related inside .sms files are heard, plus all external .sms files (with the exception of those related to the trainset where the camera is in that moment): the volume of those external files is attenuated by a 0.75 factor.
3. When being in an external view all external .sms files are heard.

For an .sms file to be heard, it must be within the activation distance defined in the related instruction.

A hack is available so as to hear only in cabview some .sms files residing outside the cabview trainset. This can be used e.g. to implement radio messages. For this to work the related .sms file must be called within a .wag file, must contain an Activation (CabCam) statement, and the related wagon must be within a loose consist, within a not yet started AI train or within the consist where the cabview trainset resides.

The ScalabiltyGroup () instruction behaves differently from MSTS for AI trains. While MSTS uses ScalabiltyGroup (0) for AI trains, OR uses for AI trains the same ScalabiltyGroup used for player trains. This way AI train sound can profit for the many more triggers active for AI trains in ORTS. E.g. Variable2 trigger is not active in MSTS for AI trains, while it is in ORTS.

14.3. Discrete triggers

Differently from MSTS, OR does not restrict operation of some discrete triggers related to locos only to the cabview related .sms file (usually named ...cab.sms file). They are all active also in the file related to external view (usually named ...eng.sms file).

OR manages following MSTS discrete triggers

2 DynamicBrakeIncrease (currently not managed)
3 DynamicBrakeOff
4 SanderOn
5 SanderOff
6 WiperOn
7 WiperOff
8 HornOn
9 HornOff

10 BellOn
11 BellOff
12 CompressorOn
13 CompressorOff
14 TrainBrakePressureIncrease
15 ReverserChange
16 ThrottleChange
17 TrainBrakeChange
18 EngineBrakeChange
20 DynamicBrakeChange
21 EngineBrakePressureIncrease
22 EngineBrakePressureDecrease
27 SteamEjector2On
28 SteamEjector2Off
30 SteamEjector1On
31 SteamEjector1Off
32 DamperChange
33 BlowerChange
34 CylinderCocksToggle
36 FireboxDoorChange
37 LightSwitchToggle
38 WaterScoopDown (currently not managed)
39 WaterScoopUp (currently not managed)
41 FireboxDoorClose
42 SteamSafetyValveOn
43 SteamSafetyValveOff
44 SteamHeatChange (currently not managed).
45 Pantograph1Up
46 Pantograph1Down
47 Pantograph1Toggle (currently not managed)
48 VigilanceAlarmReset
54 TrainBrakePressureDecrease
56 VigilanceAlarmOn
57 VigilanceAlarmOff
58 Couple
59 CoupleB (currently not managed)
60 CoupleC (currently not managed)
61 Uncouple
62 UncoupleB (currently not managed)
63 UncoupleC (currently not managed)

MSTS .sms files for crossings (crossing.sms), control error and permission announcements (ingame.sms) together with their triggers are managed.

MSTS triggers for derailment and fuel tower are currently not managed.

MSTS .sms files related to weather (clear_ex.sms, clear_in.sms, rain_ex.sms, rain_in.sms, snow_ex.sms, snow_in.sms) are managed.

The signal file (signal.sms) and its discrete trigger 1 is managed.

Moreover OR manages the extended set of discrete triggers provided by MSTSbin.

14.3.1. OR- specific discrete triggers

OR manages the following set of new discrete triggers that were not present under MSTS. If MSTS (or MSTSbin) executes an .sms where such discrete triggers are used, it simply ignores the related statements.

- triggers 101 - GearUp and 102 - GearDown for gear-based engines; they are triggered by the E resp. Shift-E key and they are propagated to all gear-based diesel engines of a train and run also for AI trains
- triggers 103 - ReverserToForwardBackward and 104 - ReverserToNeutral (valid for all loco types); this couple of triggers allows to distinguish if the reverser is moved towards an active or towards a neutral position, which is not possible under MSTS
- triggers 105 - DoorOpen and 106 - DoorClose (valid for all loco types); they are triggered by the Q and Shift-Q keys and are propagated to the wagons of the consist (that is also the .sms files of the wagons can refer to these triggers)
- triggers 107 - MirrorOpen and 108 - MirrorClose (valid for all loco types); they are triggered by the Shift-Q key.

Triggers from 109 to 118 are used for TCS scripting, as follows:

- triggers 109 and 110: TrainControlSystemInfo1 and -Info2
- triggers 111 and 112: TrainControlSystemActivate and -Deactivate
- triggers 113 and 114: TrainControlSystemPenalty1 and -Penalty2
- triggers 115 and 116: TrainControlSystemWarning1 and -Warning2
- triggers 117 and 118: TrainControlSystemAlert1 and -Alert2.

Triggers from 121 to 136 are used to synchronize steam loco chuffs with wheel rotation.
The sixteen triggers are divided into two wheel rotations. Therefore every trigger is separated from the preceding one by a rotation angle of 45 degrees.

Moreover OpenRails extends triggers 23 and 24 (electric loco power on/power off), that were introduced by MSTSbin, to diesel engines. Keys Y (for diesel player engine) and Shift-Y (for diese helpers), apart from physically powering on and off the diesel engines, trigger the above triggers.

14.4. Variable triggers

OR manages all variable triggers managed by MSTS. There can be some difference in the relationship between physical loco variables (e.g. Force) and related variable. This applies to Variable2 and Variable3.

14.5. Sound loop management

Sound loop management instructions are executed as follows by OR:

- StartLoop/ReleaseLoopRelease: the .wav file is continuously looped from beginning to end; when the ReleaseLoopRelease instruction is executed, the .wav file is played up to its end and stopped.
- StartLoopRelease/ReleaseLoopRelease: the .wav file is played from start up to the last CuePoint, and then continuously looped from first to last CuePoint; when the ReleaseLoopRelease instruction is executed, the .wav file is played up to its end and stopped.
- StartLoopRelease/ReleaseLoopReleaseWithJump: the .wav file is played from start up to the last CuePoint, and then continuously looped from first to last CuePoint; when the

ReleaseLoopReleaseWithJump instruction is executed, the .wav file is played up to the next CuePoint, then jumps to last CuePoint and stops. It is recommended to use this couple of instructions only where a jump is effectively needed, as e.g. in horns; this because this couple of instructions is more compute intensive and can lead to short sound breaks in case of high CPU loads.

14.6. Testing sound files at runtime

To this aim a useful tool is the sound debug window.

15. Open Rails cabs

OR supports both MSTS-compatible 2D cabs as well as native 3D cabs, even on the same locomotive.

15.1. 2D cabs

OR supports with a high degree of compatibility all functions available on MSTS for 2D cabs.

OR adds support for the ETCS circular speed gauge, as described here.

Moreover OR allows for configurable font family, font size selection, and selection among regular and bold style.

Here an example:

An optional line of the format ORTSfont (fontsize fontstyle "fontfamily") has to be inserted in the .cvf block of the digital controls or digital clocks, where fontsize is a float (default value 10), fontstyle an integer having value 0 (default) for regular and 1 for bold, and fontfamily is a string with the font family name (ex. "Times New Roman", default "Courier New").

This is an example that displays a bold font of size 12 and of family Sans Serif.

```
DigitalClock (
	Type ( CLOCK DIGITAL_CLOCK )
	Position ( 40 350 56 11 )
	Style ( 12HOUR )
	Accuracy ( 1 )
	ControlColour ( 255 255 255 )
	ORTSFont ( 12 1 "Sans Serif" )
```

Only the first parameter of ORTSFont can be present, or only the first two, or all three.

15.2. 3D cabs

The key to enter a 3D cab (provided the player loco has one) is Alt-1.

15.2.1. Development rules

for content developers:

1.The 3D cab is described by an .s file and the associated .ace or .dds files; all these files reside in a folder CABVIEW3D within the main folder of the locomotive

2.The 3D cab is associated to the .cvf file of the 2D cab
3. Instruments are named with the same convention, i.e., FRONT_HLIGHT, SPEEDOMETER, etc
4. A cab can have multiple instances of the same instruments, for example multiple clocks, speedometers
5. Instruments are first sorted based on their appearance in the .cvf file, for example SPEEDOMETER:0 corresponds to the first speedometer in the .cvf file, SPEEDOMETER:1 corresponds to the second one
6. An instrument can have multiple subgroups to make the animation realistic, for example, TRAIN_BRAKE:0:0 and TRAIN_BRAKE:0:1 belong to the instrument TRAIN_BRAKE:0;
7. However, if the instrument is a digital device, the second number will be used to indicate the font size used, for example SPEEDOMETER:1:14 means the second speedometer (which is digital as defined in .cvf) will be rendered with 14pt font. This may be changed in future OR releases. The important information for a digital device is its location, thus it can be defined as an object with a small single face in the 3D model.
8. Animation ranges must also be in agreement with the .cvf file

9. Within the Wagon section of the .eng file a block like the following one has to be generated:

```
ORTS3DCab(
    ORTS3DCabFile ( Cab.s )
    ORTS3DCabHeadPos ( -0.9 2.4 5.2 )
    RotationLimit ( 40 60 0 )
    StartDirection ( 12 0 0 )
    Sound ("DF11Gcab.sms" )
      )
```

10. It is also possible to animate the wipers, inserting in the .s file an animation named
`EXTERNALWIPERS:0:0`

A demo trainset with a 3Dcab, that may be useful for developers, can be downloaded from
http://www.tsimserver.com/Download/Df11G3DCab.zip

16. Developing OR contents

OR is defining and developing its own development tools.

However it is already possible to develop OR contents (rolling stock, routes, 3D objects, activities) using the tools used to develop MSTS contents, thanks to the high compatibility that OR has with MSTS.

Here some of the advantages developing OR specific contents are highlighted:

16.1. Rolling stock

1. OR is able to run shapes with much more polygons than MSTS. Shapes with more than 100.000 polys have been developed and run without problems.
2. Thanks to the additional physics description parameters, a much more realistic behavior of the rolling stock is achieved
3. 3D cabs add much to realism
4. OR graphics renders much better the results of the work of the rolling stock developers
5. Rolling stock driven and looked on super-elevated track improves gaming experience.

16.2. Routes

1. Routes are displayed in a much more crisp way
2. Thanks to the possibility of extending viewing distance, much more realism is possible
3. Double overhead wire adds much to realism of electrified routes
4. Extended signalling features allow for more realistic signal behavior.

16.3. Activities

1. Thanks to dispatcher monitor, dispatcher HUD, possibility to switch camera to any AI train, the player can monitor and control much deeper the execution of the activity, acting also as dispatcher, thus increasing fun
2. Timetable mode allows for fast and powerful development of timetable based gaming sessions
3. Extended AI shunting highly increases the palette of possible interactions between trains.

16.4. Testing and debugging tools

As listed here, a rich and powerful set of analysis tools eases the testing and debugging of contents under development.

16.5. Support

Support can be requested on the OR forum on www.elvastower.com/forums .

The OR development team, within the limits of its possibilities, is willing to support contents developers.

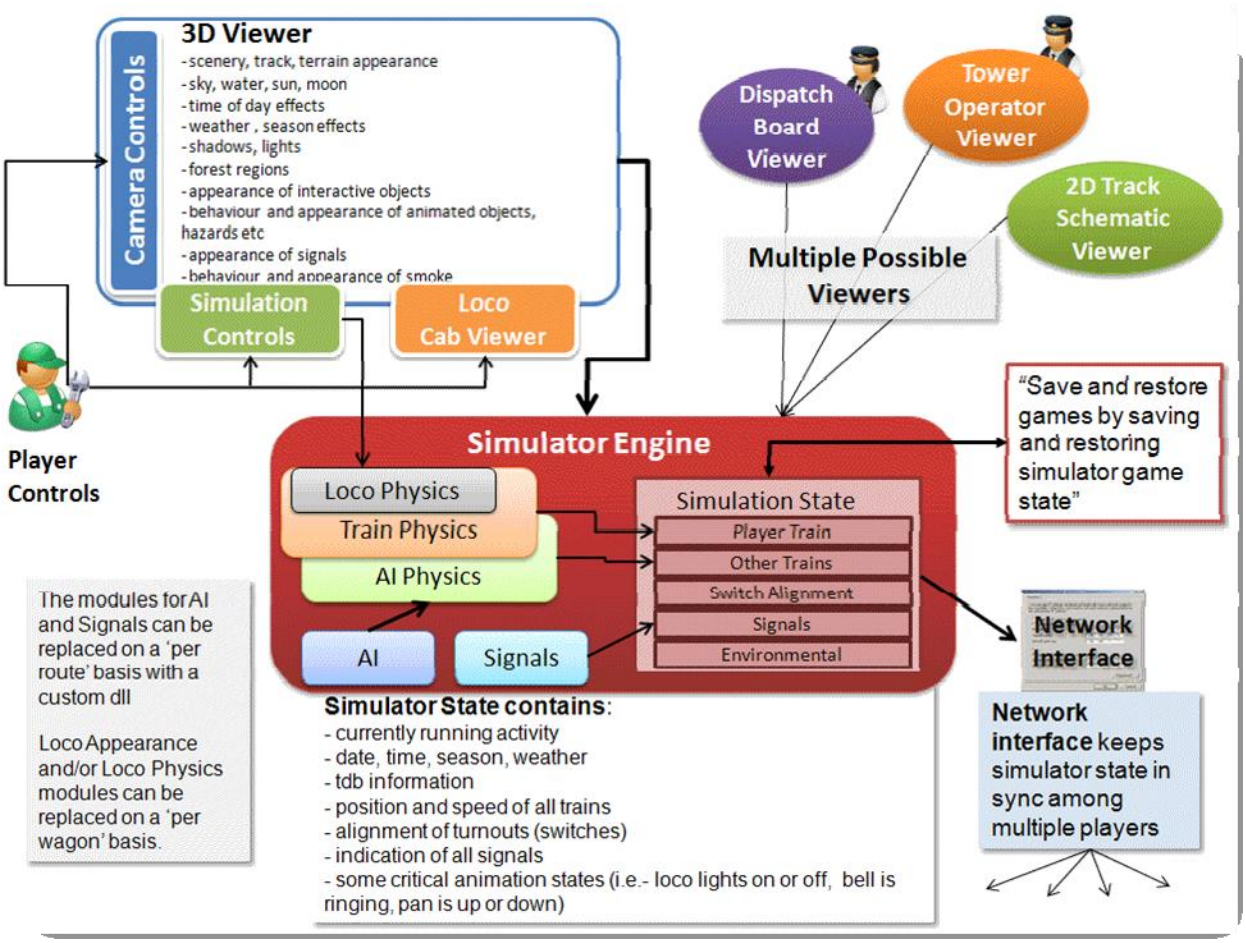
17.Open Rails Software Platform

Inside view:

17.1. Architecture

To better understand how the Open Rails game operates, performs, and functions, the architecture diagram below lays out how the software code is organized. The architecture of the Open Rails software allows for modular extension and development, while providing standardized methods to customize the simulation experience.

Please note that this diagram includes many capabilities and functions that are yet to be implemented.



17.2. Open Rails Game Engine

The Open Rails software is built on Microsoft's XNA game platform using XNA Framework 3.1 and .NET Framework 3.5 SP1. Source code is developed in Microsoft's Visual C#

programming language.

The XNA Framework is based on the native implementation of .NET Compact Framework for Xbox 360 development and .NET Framework on Windows. It includes an extensive set of class libraries, specific to game development, to promote maximum code reuse across target platforms. The framework runs on a version of the Common Language Runtime that is optimized for gaming to provide a managed execution environment. The runtime is available for Windows XP, Windows Vista, Windows 7, Windows 8, and Xbox 360. Since XNA games are written for the runtime, they can run on any platform that supports the XNA Framework with minimal or no modification of the Game engine.[1]

 A license fee is payable to Microsoft to use XNA Game Studio for Xbox 360 games. At this time, the Open Rails team has not investigated whether the Open Rails software is suitable for Xbox.

17.3. Frames Per Second (FPS) Performance

For the current release, Open Rails development team has untethered the FPS rate from the synch rate of the monitor. This allows the development team to more easily document performance improvements. The Open Rails team at a later date may decide to limit FPS to the synch rate of the monitor.

17.4. Game Clock and Internal Clock

Like other simulation software, Open Rails software uses two internal “clocks”; a game clock and an internal clock. The game clock is required to synchronize the movement of trains, signal status, and present the correct game environment. The internal clock is used synchronize the software process for optimal efficiency and correct display of the game environment.

The Open Rails team is dedicated to ensuring the game clock properly manages time in the simulation, so that a train will cover the proper distance in the correct time. The development team considers this vital aspect for an accurate simulation by ensuring activities run consistently across community member’s computer systems.

17.5. Resource Utilization

Because Open Rails software is designed for Microsoft’s XNA game framework, it natively exploits today’s graphics cards ability to offload much of the display rendering workload from the computer’s CPU.

17.6. Multi-Threaded Coding

The Open Rails software is designed from the ground up to support up to 4 CPUs, either as virtual or physical units. Instead of a single thread looping and updating all the elements of the

simulation, the software uses four threads for the main functions of the software.

Thread 1 - Main Render Loop (RenderProcess)
Thread 2 - Physics and Animation (UpdaterProcess)
Thread 3 - Shape and Texture Loading/Unloading (LoaderProcess)
Thread 4 – Sound

There are other threads used by the multiplayer code as each opened communication is handled by a thread.

The RenderProcess runs in the main game thread. During its initialization, it starts two subsidiary threads, one of which runs the UpdaterProcess and the other the LoaderProcess. It is important that the UpdaterProcess stays a frame ahead of RenderProcess, preparing any updates to camera, sky, terrain, trains, etc. required before the scene can be properly rendered. If there are not sufficient compute resources for the UpdaterProcess to prepare the next frame for the RenderProcess, the software reduces the frame rate until it can “catch up”.

Initial testing indicates that “stutters” are significantly reduced because the process (LoaderProcess) associated with loading shapes and textures when crossing tile boundaries do not compete with the main rendering loop (RenderProcess) for the same CPU cycles. Thread safety issues are handled primarily through data partitioning rather than locks or semaphores to maximise performance .

Ongoing testing by the Open Rails team and the community will determine what and where the practical limits of the software lie. As the development team receives feedback from the community, improvements and better optimization of the software will contribute to better overall performance – potentially allowing high polygon models with densely populated routes at acceptable frame rates.

18. Plans and Roadmap

Here are some highlights that the community can expect from the Open Rails team after v1.0. A fuller roadmap can be found at https://launchpad.net/or/+milestones

18.1. User Interface

A new Graphical User Interface (GUI) within the game.

Cab controls operated by mouse (similar to MSTS).

18.2. Operations

In addition to the new Timetable concept described in this document, some further improvements are planned:

• Extended ability to customize signals to accommodate regional, geographic, or operational differences
• Ability to use mixed signal environments - from dark territory to full automatic in-cab train control within the same route
• specifying random variations for AI trains in consist and delays.
• specifying separate speed profile for passenger or freight trains.
• AI trains which can split or combine
• schedule for AI trains which can depend on other trains (e.g. wait a limited time).

18.3. Activity Editor

The team has begun work on the Open Rails Activity Editor for setting up paths (routes through signaling and interlocking) for both Player and AI trains. Already this editor has moved beyond MSTS in providing the new concept Station Area. This will reduce the required total number of MSTS pathfiles - many pathfiles are just variations, differing from other pathfiles only by a route through a station (e.g. different platform) or yard. By using a Station Area, you can define a single main pathfile for all trains, with the different routes through the station given by notes in the timetable.

It will also make the whole MSTS system of passing paths redundant, with any passing paths no longer defined per train but per station. Finally, it will allow the definition of multiple and more complex passing paths which require additional switch settings, something that MSTS cannot handle.

18.4. Route Editor

Now that the project is moving beyond MSTS, we are at last able to specify the Open Rails Route Editor. This will free us from the constraints and fragility of the MSTS tool.

The editor will, of course, use GIS data, edit the terrain and allow objects to be placed and moved.

In particular, it will be possible to lay both track pieces and procedural track. The procedural track may bend up and down to follow the contours of the land and twist to build banked curves and spirals. There will be support for transition curves and it will be easy to lay a new track parallel to an existing one.

The new Route Editor will not be backwards-compatible with MSTS routes. It will work with Open Rails routes and there will be a utility to create an Open Rails route from an MSTS route.

No timetable is available for this work.

19. Acknowledgements

The Open Rails team would like to acknowledge the following for their work, without which Open Rails would not be possible.

- **John Sandford**, **Jim Jendro** and **Wes Card** for deciphering the MSTS tile files and providing computer algorithms that go a long way toward helping us do the same.
- **Riemer Grootjans** for his informative web tutorials and detailed code and for his book, "XNA 3.0 Game Programming Recipes."
- **Jan Vytlačil** for showing us how to make it rain and snow.
- **Paul Bourke** for the high-resolution star maps of the northern and southern hemispheres.

And **Wayne Campbell** for inspiring this improbable journey.

20.Appendices

20.1. ORTS Parameters

Open Rails aims for more realism than MSTS achieved and allows some data files to contain parameters which are specific to OR. These parameters all have a prefix of "ORTS" and will be ignored by MSTS.

The current list follows.

```
ORTSMaxViewingDistance
ORTSDLL
ORTSExhaustColor
ORTSExhaustTransientColor
ORTSDieselEngines
ORTSDynamicBrakesHasAutobailoff
ORTSMainResChargingRate
ORTSEngineBrakeReleaseRate
ORTSEngineBrakeApplicationRate
ORTSBrakePipeTimeFactor
ORTSBrakeServiceTimeFactor
ORTSBrakeEmergencyTimeFactor
ORTSBrakePipeChargingRate
ORTSMaxTractiveForceCurves
ORTSDynamicBrakeForceCurves
ORTSContinuousForceTimeFactor
ORTSSanderSpeedEffectUpTo
ORTSPowerOnDelay
ORTSEmergencyCausesPowerdown
ORTSEmergencyCausesThrottledown
ORTSEmergencyEngagesHorn
ORTSWheelslipCausesThrottledown
ORTSGrateArea
ORTSEvaporationArea
ORTSFuelCalorific
ORTSBurnRateMultiplier
ORTSForceFactor1
ORTSForceFactor2
ORTSCylinderPressureDrop
ORTSBackPressure
ORTSBurnRate
ORTSBoilerEfficiency
ORTSDiesel
```

```
ORTSAdhesion
ORTSCurtius_Kniffler
ORTSSlipWarningThreshold
ORTSAntislip
ORTSInertia
ORTSRadius
```

20.2. Units of Measure

Open Rails supports the same default units of measure as MSTS which are mostly, but not exclusively, metric.

When creating models just for Open Rails, we recommend you do not use defaults but specify units for all values that represent physical quantities.

As shown below, Open Rails provides a wider choice of units than MSTS.

Measure	Default unit	Applies to	OR accepts	MSTS accepts	Comment
Mass	kg		kg	kg	
			t	t	metric tonne (1,000 kg)
			lb	lb	
			t-uk		Imperial ton (2,240 lb)
			t-us		US ton (2,000 lb)
Distance			mm		
			cm	cm	
	m		m	m	
			km		
			in	in	
			in/2	in/2	half-inch - historic unit for tyre diameters
			ft		
			mile		
Area			m^2		
			*(m^2)	*(m^2)	
	ft^2		ft^2		
			*(ft^2)	*(ft^2)	
Volume	l	diesel fuel	l		litres
			m^3		
			*(m^3)		
			in^3		
			*(in^3)		
	ft^3	other	*(ft^3)	*(ft^3)	e.g. BoilerVolume
			g-uk		Imperial gallons
			g-us		US gallons
			gal		US gallons
			gals	gals	US gallons
Time	second		s		
			m		
			h		

Current	amp		a		
			amp		
Voltage	volt		v		
			kv		
Mass Rate			g/h		
			kg/h		
	lb/h		lb/h	lb/h	
Speed	m/s	other	m/s	m/s	metres per second
			km/h		
			kph	kph	kilometres per hour
			kmh	kmh	misspelling accepted by MSTS
	mph	dynamic brake	mph	mph	miles per hour
Frequency	hz		hz		Hertz
			rps		revolutions per second
			rpm		
Force	N		n	n	Newton
			kn	kn	
			lbf		Pounds force
			lb		
Power	w		w		Watt
			kw		
			hp		horsepower
Stiffness	n/m		n/m	n/m	Newtons per metre
Resistance	n/m/s		n/m/s	n/m/s	Newtons per metre per second
			ns/m		Newton seconds per metre
Angular Resistance	n/rad/s			n/rad/s	
Pressure	psi	air pressure	psi		pounds per square inch
			bar		atmospheres
			kpa		Kilopascals
	inhg	vacuum	inhg		inches of mercury
Pressure Rate	psi/s		psi/s		
			bar/s		
			kpa/s		
			inhg/s		
Energy Density	kj/kg		kj/kg		Kilojoules per kilogram
			j/g		

			btu/lb		Board of Trade Units per pound
Temperature Difference	degc		degc		
			degf		
Angle	radians		-		
				deg	
Angular Rate	rad/s		-	rad/s	
Other			-	lb/hp/h	e.g. CoalBurnage

20.3. Open Rails Best Practices

20.3.1. Polys vs. Draw Calls – What's Important

Poly counts are still important in Open Rails software, but with newer video cards they're much less important than in the early days of MSTS. What does remain important to both environments are Draw Calls.

A Draw Call occurs when the CPU sends a block of data to the Video Card. Each model in view, plus terrain, will evoke one or more Draw Calls per frame (i.e., a frame rate of 60/second means all of the draw calls needed to display a scene are repeated 60 times a second). Given the large number of models displayed in any scene and a reasonable frame rate, the total number of Draw Calls per second creates a very large demand on the CPU. Open Rails software will adjust the frame rate according to the number of required Draw Calls. For example, if your CPU can handle 60,000 Draw Calls per second and the scene in view requires 1000 Draw Calls, your frame rate per second will be 60. For the same CPU, if the scene requires 2000 Draw Calls, your frame rate per second will be 30. Newer design / faster CPU's can do more Draw Calls per second than older design / slower CPU's.

Generally speaking, each Draw Call sends one or more polygon meshes for each occurrence of a texture file for a model (and usually more when there are multiple material types). What this means in practice is if you have a model that uses two texture files and there are three instances of that model in view there will be six draw calls – once for each of the models (3 in view) times once for each texture file (2 files used), results in six Draw Calls. As an aid to performance Open Rails will examine a scene and will issue Draw Calls for only the models that are visible. As you rotate the camera, other models will come into view and some that were in view will leave the scene, resulting in a variable number of Draw Calls, all of which will affect the frame rate.

Model builders are advised that the best performance will result by not mixing different material types in a texture file as well as using the fewest number of texture files as is practical.

Players are advised to adjust camera positions to cull models from being in view whenever frame rates fall to unacceptable levels and to adjust the camera again to include more models when frame rates are high.