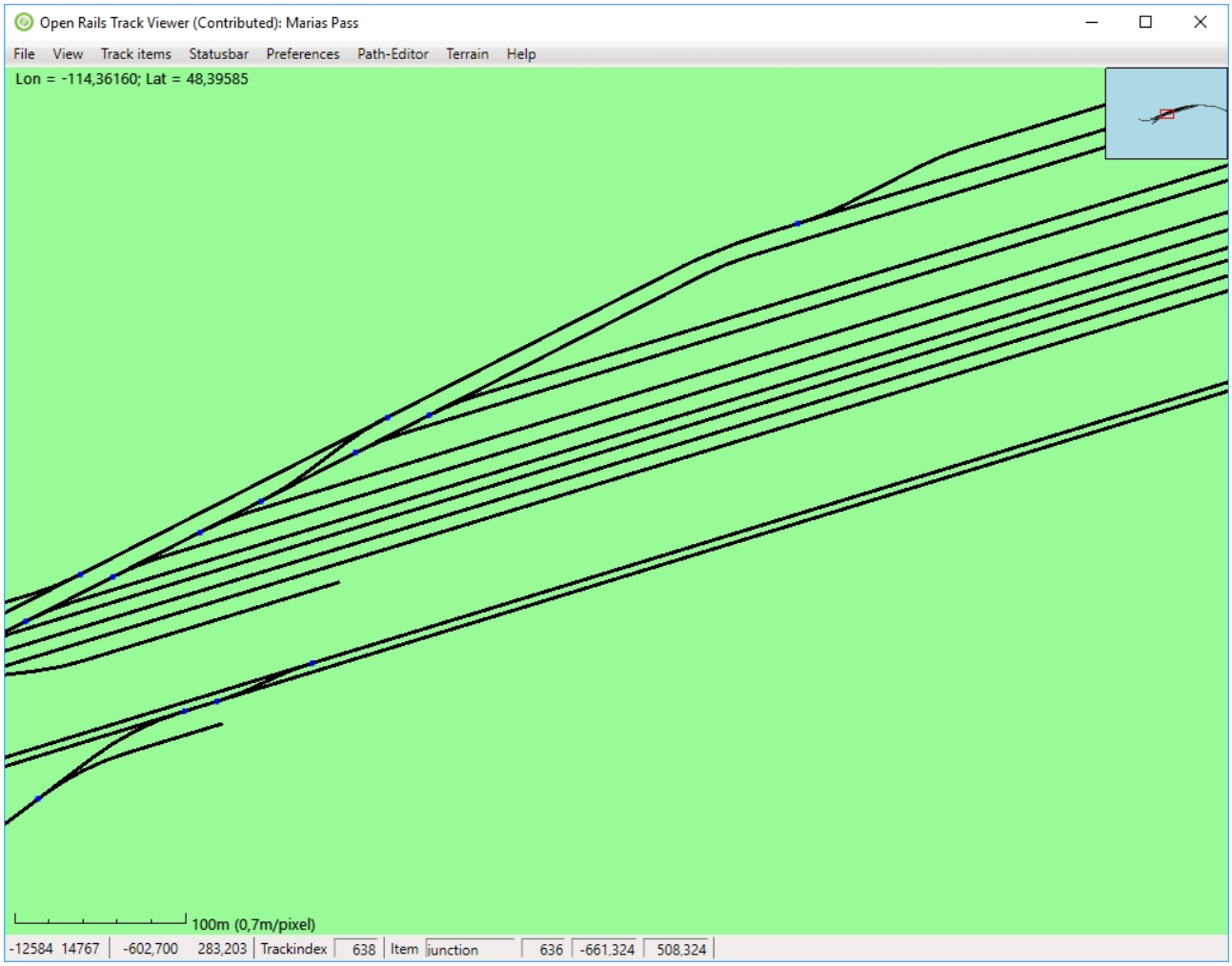
ORTS TrackViewer

Viewing MSTS tracks and editing MSTS paths



1. Introduction

ORTS TrackViewer is an open source program to view tracks and all track items from a MSTS (Microsoft Trains Simulator) route and to edit MSTS paths (as used in activities). The viewing part of TrackViewer is very similar to the program MSTS TrackViewer, which is no longer developed. The ability to edit paths is new.

2. Installation and support

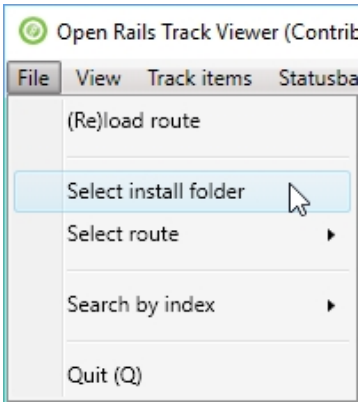
TrackViewer is current part of the Open Rails Transport Simulator (ORTS, see www.openrails.org). It is available both in source code and as a pre-compiled binary as part of the ORTS distribution. This means that if you installed ORTS, you will have TrackViewer as well. Its executable is called Contrib.Trackviewer.exe. TrackViewer will not run independently of ORTS, since it reuses parts of the code of ORTS. For more information on installing ORTS, see its web-site http://www.openrails.org/download/program/.

Although TrackViewer is part of the ORTS release, it does not have the same status. It is a so-called contributed package. There is no promise from the ORTS team to fix all issues. Since this is open source and based on spare-time contributions of one or a few coders, there are no guarantees. Support can be found in the same way as for ORTS itself. That is mainly via the forums on Elvas Tower (http://elvastower.com/, under SOFTWARE DEVELOPMENT → OPEN

RAILS)DEVELOPMENT, TESTING, and SUPPORT → Contributed Software → Track Viewer).

3. Loading a route

3.1. Via the menu

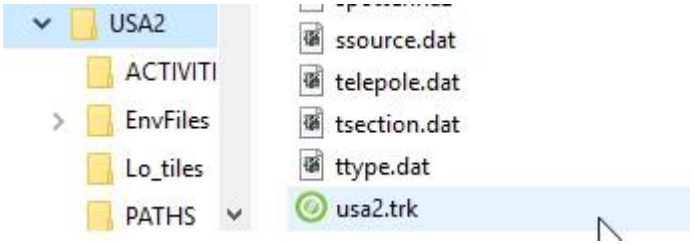


When running TrackViewer for the first time, it will try to find your MSTS installation. If it does not find it (or in case you did not install MSTS), you can select the install folder under the File menu. The install folder needs to be the folder that has 'ROUTES' as a sub-folder.

You can then select a route, again via the File menu. After the first use, you can reload your previously loaded route (it will load the first available route if you did not load a route before).

TrackViewer will keep most of your settings. This includes the install folder, the last route, and what you selected to be drawn on screen and what not. In other words, almost all selections work as preferences at the same time.

3.2. Via file explorer



It is also possible to open a route via the Windows file explorer (i.e. with a double click). This currently works for the following files:

- .trk.
- .tdb.
- .rdb (in which case roads will also be shown).
- .pat (in which case the path will be loaded and shown as well).

Do note that currently there is a little bit of a short-cut in the implementation. Normal route loading is done based on a folder. In that folder then the .trk file is opened which contains information on the filename that is to be used for the .tdb and .rdb files. That concept is not really changed. When opening a .trk, .tdb, or .rdb file, basically only the folder is taken and that folder is then first used to select the install folder and subsequently the route folder. The actual file name is neglected (which should only matter if you have multiple files with the same extension in the same folder).

4. Viewing a route

Most of the menu items are pretty self-explaining. You can select which items and which tracks to draw.

4.1. Moving around

By default the whole route is shown. This means all the available tracks in the track database (.tdb file).

Zooming can be done with the plus and minus keys ('=' and '-') or the mouse wheel. Zooming with the mouse is faster by default because it takes more zooming steps at once. Pressing shift during zooming with the mouse wheel gives you finer control. Zooming is centered around the mouse position normally. This means that if you point your mouse to a feature you want to zoom in to and scroll the wheel, the region of interest will be zoomed into.

Shifting the view window can be done using the arrow keys, or with the mouse (dragging with the

left-mouse button pressed).

Some other keyboard short-cuts related to viewing are described in section 30, as well as in the Help menu.

4.2. Roads

Next to the all-important track database for trains, also the roads can be shown. These are defined in the .rdb file.

4.3. Coloring

By default track is colored using black for straight sections and green for curved sections. Roads are drawn in grey for straight sections and olive green for curved sections.

The track section that is closest to the mouse is by default highlighted (e.g. in red). This allows you to get details on that particular section in the status bar (see later).

Using a different color for curved sections and highlighting can be turned on and off and via the menu View → Track Coloring. The actual colors used can be set via the Preferences menu.

4.4. TrackItems

There are many items along the tracks that are needed for the simulator (signals, sidings and platforms, interactives, speed limits and mile posts, …). All of these are coded in the track database (either .tdb or .rdb). Also end-points (where a track ends), junctions and cross-overs exist.

All of these can be either shown or hidden. What exactly is shown can be chosen via the Track items menu or via keyboard short-cuts (see later in this documentation or under the Help menu). Most items have a dedicated icon to show the kind of item (e.g. a bell for a soundregion, a signal post for a signal, a car for a carspawner, etc). Others are just shown using a disc (filled circle).

For some items not only an icon is shown, but also a text. This can be the name of a siding or platform, and it can be the speed limit or the value on a mile post.

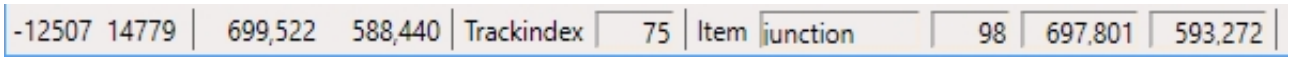
It is possible to highlight a certain track item. See for instance the yellow instead of orange bell in the picture above. In general a highlighted item will have a lighter color. The details of the highlighted item are shown in the status bar.

4.5. Status bar

The status bar gives some detailed information mostly relevant for route and path developers. The main information is always visible. The other information can be made visible from the menu. Multiple selections can be made, but at some point there is no longer room to show everything.

4.5.1. Main information

The main and always present information is show below:



These numbers mean the following (see below for details), from left to right

- Integer describing the number of the tile in x-direction
- Integer describing the number of the tile in z-direction
- x-offset within a tile
- z-offset within a tile
- Trackindex: the index of the piece of track closest to the mouse.
- TrackItem: the type of item closest to the mouse
- The index of the item
- the x-offset of the item within the tile
- the z-offset of the item within the tile

Locations in MSTS and hence ORTS are stored using a tile-based system. Tiles are square area's of 2048m × 2048m. The x-direction is along east-west axis. The z-direction is approximately along south-north axis. The y-directions is the height above the reference level (not used in TrackViewer).

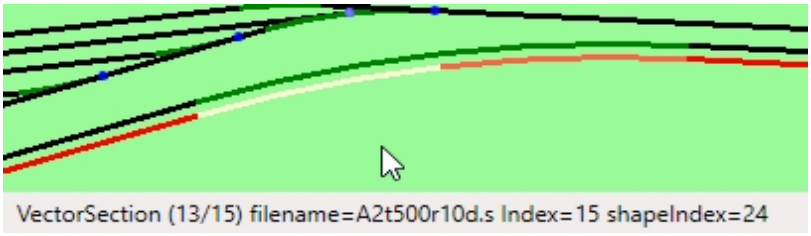
The tile is given by two integers (called tileX and tileZ). Within such a tile the location is given using an offset in the x- and z-directions from the middle of the tile (so running from -1024m to +1024m).

The tracks, junctions, end-nodes and various track items like sidings, crossings, signals etc are defined in the track data base (.tdb file). Tracks are pieces of track (or roads) along which a train or car can move. These tracks are bounded by either end-nodes or junctions. All of these things have indexes that are given at the bottom. Junctions, end-nodes and other track items also have a precise location (tracks, in contrast, are not at a certain point, but are basically curved lines).

4.5.2. Vector section information

A track in the track database is stored as a so-called vector node. Such a vector node consists of a number of sections, each one either straight or curved. When the information on the trackvector is

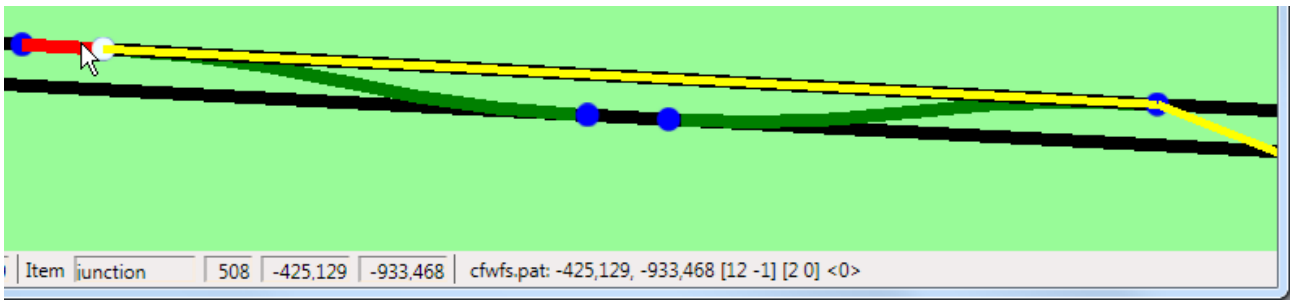
shown, the actual section is high-lighted even more (white section below). The number of the vector section in the track is shown. In this case, the red track consists of 15 sections, the 13th of which is shown in white. This section will be represented by the shape called A2t500r10d.s (there are other publications on the net to describe more details). The index of the section is 15 (in this case) whereas the index of the shape is 24. Obviously, this information might be relevant for a number of route developers, but not for general usage.



4.5.3. PAT file information

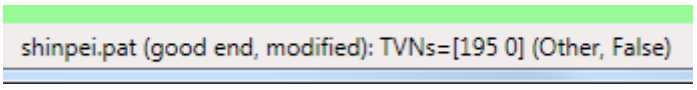
When the raw .pat file information is drawn, we mean that only information in the .pat file is used. Mostly this is the location of points and how they are connected. There is no relation made to the track database. Although, obviously, for most paths the nodes will be on logical places. Because no track information is used, only straight lines will be drawn.

The status bar information for a raw .pat file is therefore also pretty basic. First the filename of the path is shown. Then the exact location of the current node (meaning the last-drawn node). For this location only the offset in x- and z-direction within the current tile is used. Between the first square brackets the indexes of the next main and siding node are shown (-1 means no next node). Between the second square brackets the flags of the trackPDP are given (the last two numbers in the trackPDP definition, for an example see one of the .pat files). At last, between angled brackets, are the flags for the current node. Normally 0. For reversal, wait and some other nodes this will be different.



4.5.4. Path information

When editing paths it is sometimes useful to get more detail on the path itself. That is shown when you select the path-information in the statusbar. First the filename of the path is stored. The file will be in ROUTES/<your route>/PATHS. Between brackets details on the path are shown:



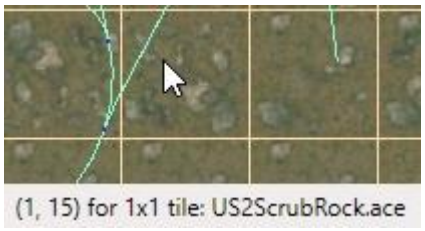
- Good end: this means a well-defined end-node has been defined.
- Modified: this means the path has been modified (in TrackViewer, when editing has been enabled). Note that when loading a path and then enabling editing sometimes some edits are done automatically to make the path well-defined (according to TrackViewer of course).
- Broken: If the path is broken, this will be shown. Broken means that the .pat file is (no

longer) in line with the track database. It might have points defined that do not correspond to junction locations, etc. Or it might have a flag set in the .pat file indicating it is broken (even though the path might be fine itself). After fixing a broken path this denomination should go away.

- Stored tail: a tail is stored and is waiting for reconnect (see below for details on how to use this).

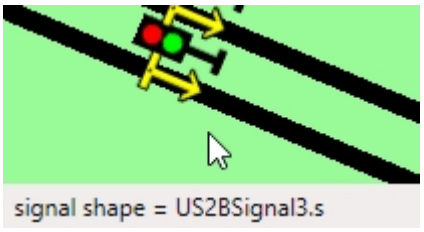
Next some information is shown about the last drawn node (so not the active node during editing, but the last node that has been drawn). First there are the TVNs (Track Vector Nodes, meaning trackNodes that are not junctions or endnodes) of the next main and the next siding path are shown. Between the last brackets the type of node is shown (Other means the there is nothing special about the node). If a node is broken, this will be shown together with the reason why it is broken. At last, for wait points also the wait-time will be shown.

4.5.5. Terrain information



The status bar can show the information on the location of the patch within the tile, and the name of the .ace file that is used for that patch. A 1×1 tile is a normal tile (128m×128m). An 8×8 tile is a DM tile (1024m×1024m).

4.5.6. Signal information

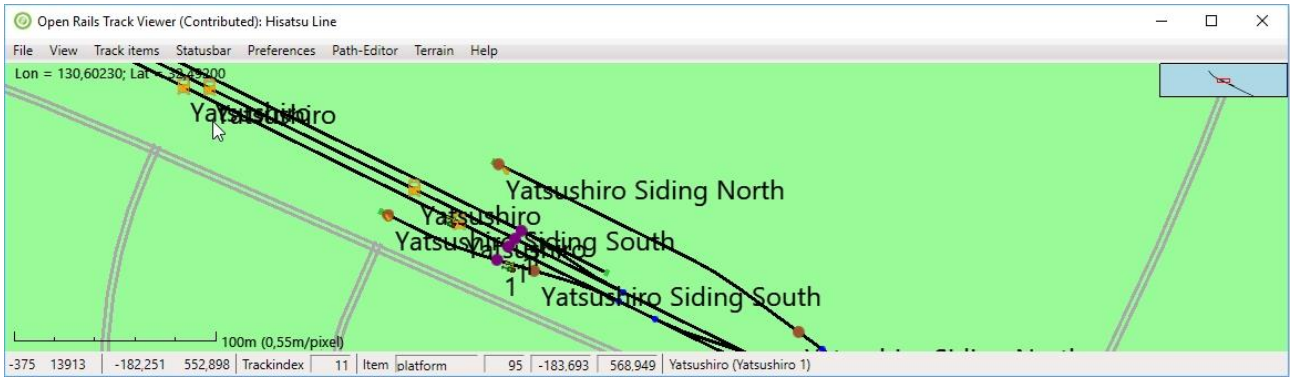


If signals are shown, the status bar can show information on the signal shape. In particular it will show the .s file that contains the signal shape.

Note that the actual location of the signal is given by the black dot in the middle of the base of the yellow arrow.

4.5.7. Station and platform names

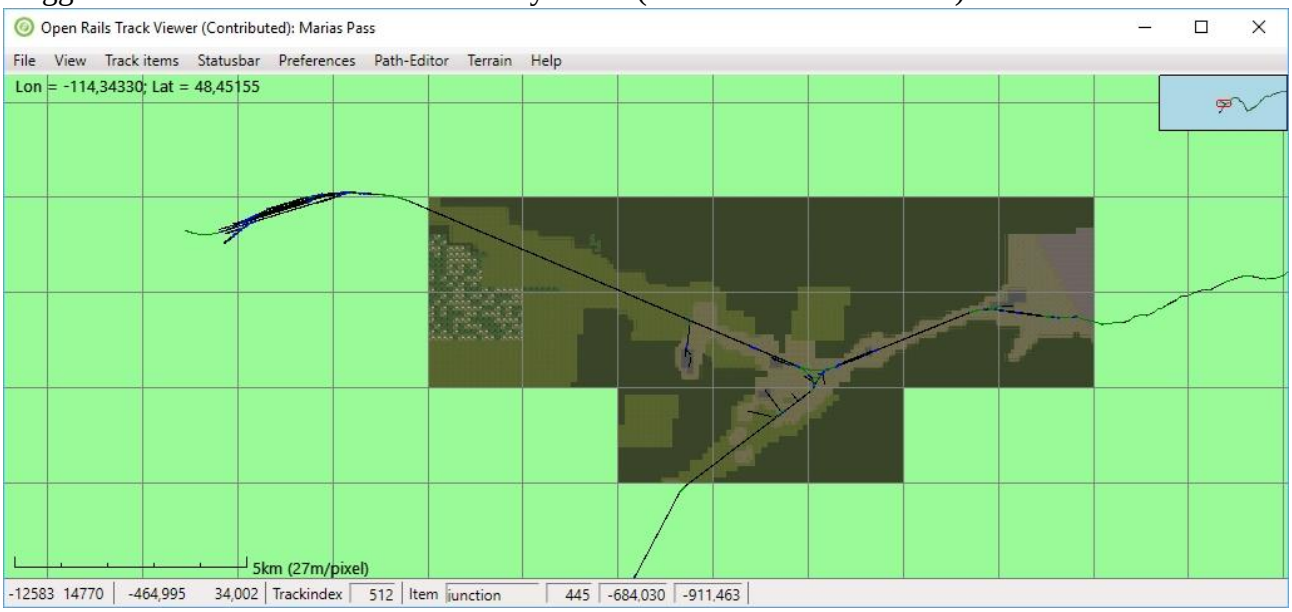
To make timetables in ORTS, you need information on station names as well as track information (i.e. you need the platform name). In crowded scenes the information on station names and platform names are sometimes difficult to see. Therefore these names can be shown on the status bar (if the track item closest to the mouse is a platform.



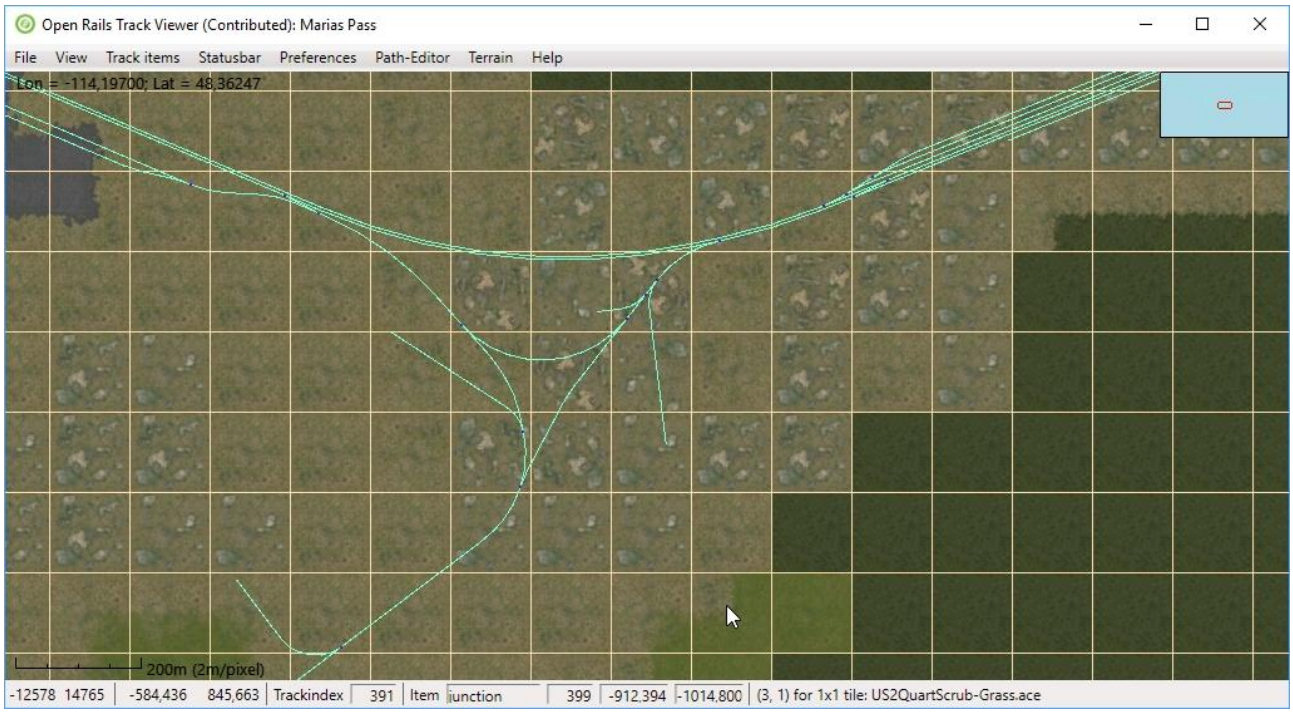
4.6. Terrain

It is also possible to show terrain tiles. This is like the helicopter view of what the ground looks

like. The textures are loaded from the TerrTex .ace files. There are two kinds of these terrain tiles: the normal ones and the Distance Mountain (DM) ones. The latter have less detail but cover a bigger area and are intended for far-away views (not all routes have them).

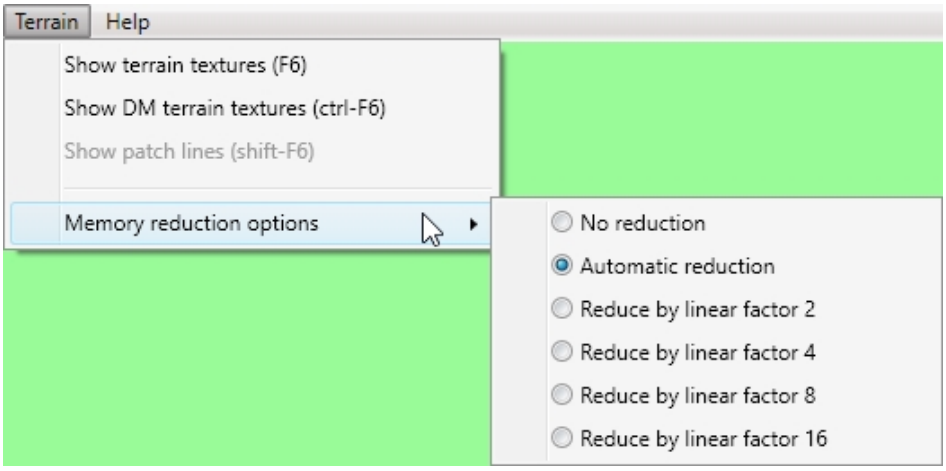


Terrain tiles need quite a big amount of memory if you load them all at the same time. It is therefore best to zoom in to the region of interest and press F6 (or use the menu item). TrackViewer will then load the necessary tiles for that region alone. Zooming out again will not load additional terrain and you can get a view like in the picture above.



If you zoom in and also show the patch lines, you get the view as above. Each patch is an area of 128m×128m and is made using one of the .ace files (possibly by rotating and/or flipping). Note that the status bar shows the information on the location of the patch within the tile, and the name of the .ace file that is used for that patch.

Note that in the picture above track coloring has been turned off. In that case the tracks get a different color for better visibility (all colors can be modified via the Preferences menu).

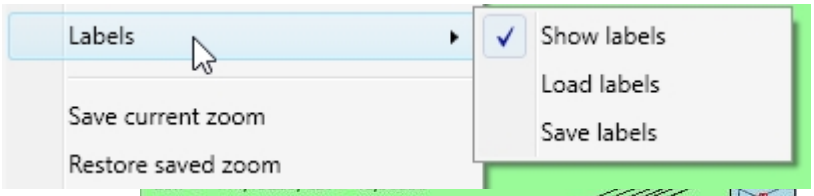


The reason for the large memory consumption is that all the terrain textures need to be loaded for them to be visible, in contrast to what happens in a normal simulator run where only the relevant terrain textures are loaded (plus a lot of other textures of course). To reduce the amount of memory there is the possibility to scale down the textures. This means, for instance, that a 256×256 texture with a reduction factor of 4 will be mapped onto a 64×64 texture. The original texture is removed from memory, so the memory consumption goes down with a factor 16 in this case. It is possible to go to a higher reduction factor by just remapping the texture that is in memory to an even smaller texture. No reloading is needed. But if you want to go to a lower reduction factor, the textures have to be reloaded, since the detailed information is no longer in memory.

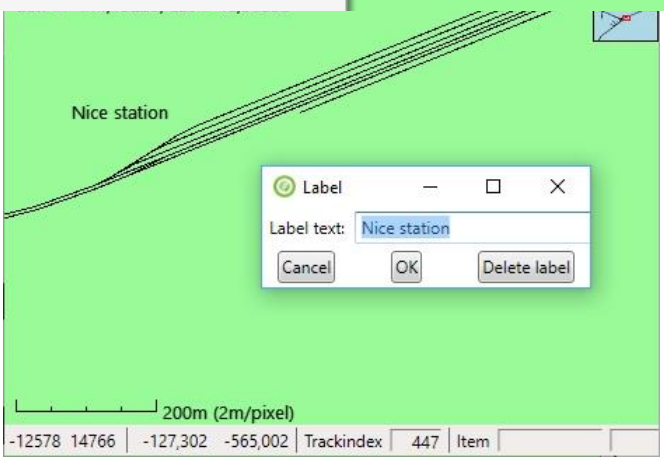
The effect of memory reduction can of course be seen if you zoom in, but for viewing a large part of the route it should not matter. Automatic reduction will consider the amount of .ace files that have been read and increase the reduction factor as more textures are loaded. Also previously loaded textures will be re-scaled if needed. The largest reduction factor will be used when more than 6400 textures are present.

4.7. Labels

Labels allow you to make it easier to present your route to users. They are just pieces of text that you can show on the screen. This is similar to the text for sidings, platform and station names. But in contrast to those, labels are not stored in any of the standard route files, but in an independent file. As such, they also do not at all influence the ORTS simulation.



For labels there is a new entry in the View menu. Obviously you first have to select Show labels to see anything. You can save the labels you have created (see below) to a file. This file will be in the .json format. You can save that file anywhere you like, since currently there will not be any automatic loading of that file. Once you have a .json file with labels, you can load that file again.



A label can be added using the short-cut key L. You will get a small popup where you can add the text of the label.

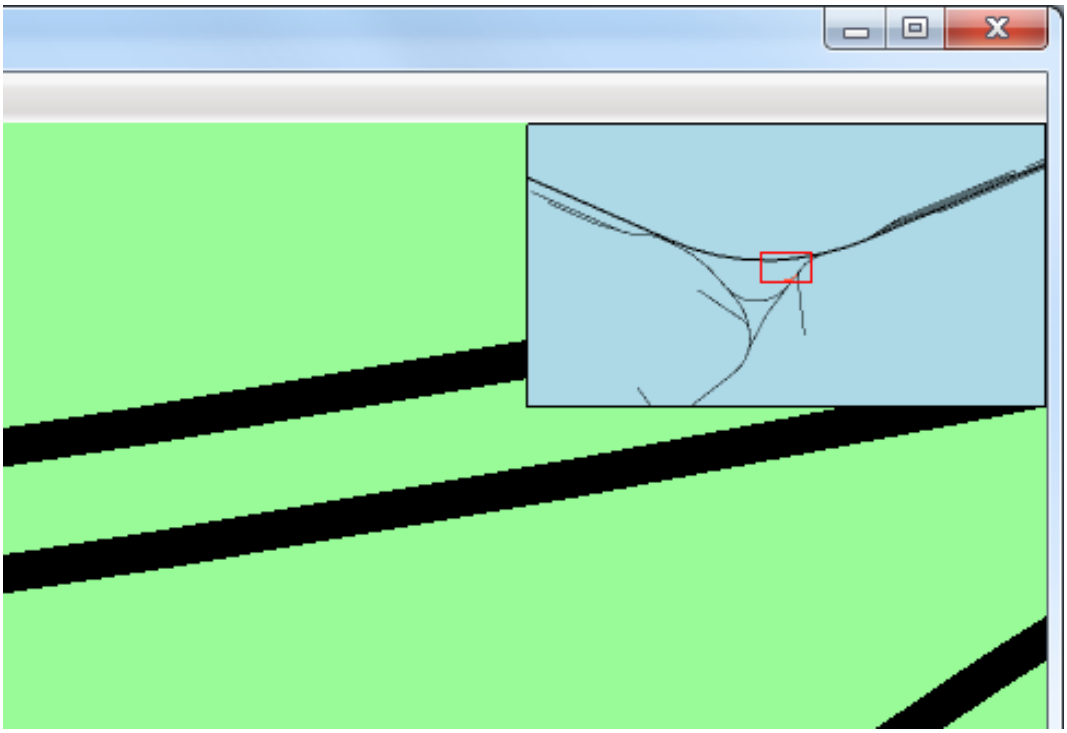
To modify the text of a label or to delete a label you can use the context menu (right-mouse click, but make sure you are not also editing a path), and select ‘edit label’.

To move the label to a different position, you can drag it with the ctrl-key pressed (so press ctrl, then left-click the mouse and move it around). Only the label that is closest to the mouse pointed will be dragged.

4.8. Additional things on screen

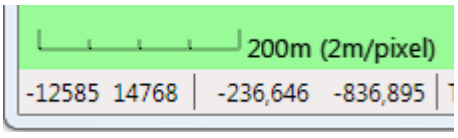
4.8.1. Inset

The inset on the top-right shows you a bigger part of the track but at a smaller scale (ratio 1:10) such that you can see where you are when zoomed in. The red rectangle shows where the main window is located.



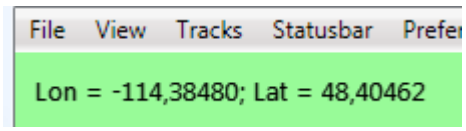
4.8.2. Scale ruler

The scale ruler shows the current scale. There is both a ruler as well as a description of how many meters a single pixel describes. Both of these obviously change while zooming in or out. Note that when pressing Shift during zooming you have more control on zooming, to make it easier to reach the same zoom-level as in a previous run.



4.8.3. World location

MSTS (and hence ORTS) internally does not store the location in real-world longitude and latitude values, but in tiles (see also the section on status bar above). The projection that MSTS uses to translate real world-locations on a sphere to the flat tile-based locations is called Goode Homolosine projection. This is a complicated and not-often used projection. We can use this projection to calculate the approximate real world location in Longitude (negative is West, positive East) and Latitude (negative is South, positive is North). Note that because of the complexity in the projection the ORTS numbers shown here might differ slightly from those in MSTS .



It is also important to realize that 'up' in TrackViewer is not exactly North.

4.8.4. World tiles

MSTS only defines its world in so-called tiles. Below the world-tiles that are actually used are shown. On the white rectangles there is nothing within MSTS (and hence ORTS). Only on the green tiles there are tracks and other objects. As mentioned above, instead of just showing the world-tiles it is also possible to show terrain.

4.9. Searching and finding

If a developer is working on a track database (.tdb or .rdb) then the index of a trackNode or trackItem is available. Via the File → Search by index it is possible to find where exactly this element is. Once it is found, the view will center around that particular location. It might help to first zoom in a bit to make sure it is clear where the actual element is. For detailed finding it helps to have the element highlighted and use the information on the status bar.

Next to that one can go to (i.e. center around) a certain named station, platform or siding via the menu View→ Center around … .

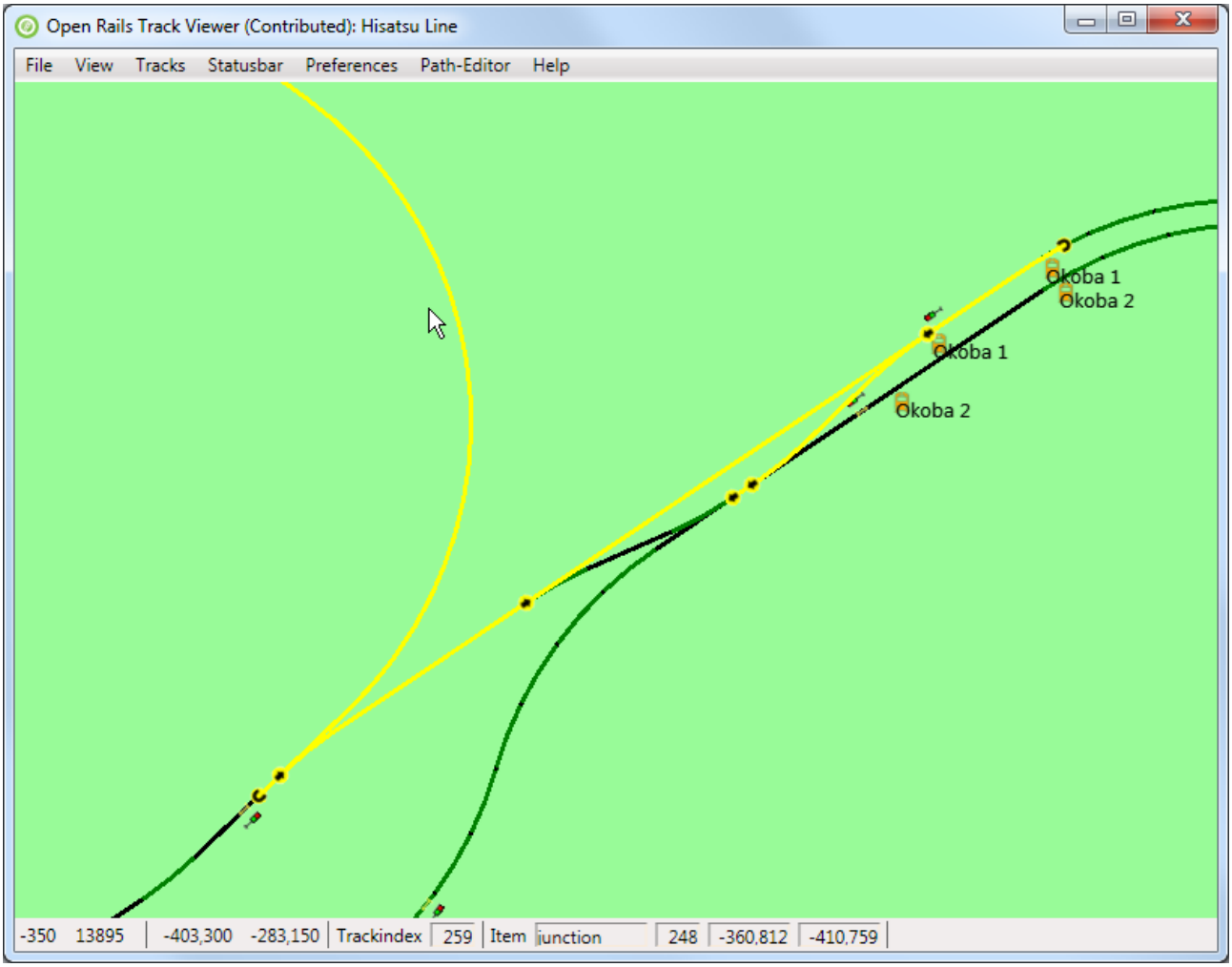
5. Viewing a path

TrackViewer also supports the drawing of paths, as used by activities for both player train and AI trains. These paths are defined in the PATHS folder of the route and have extension .pat.

5.1. Drawing a path

Drawing can be done in two ways.

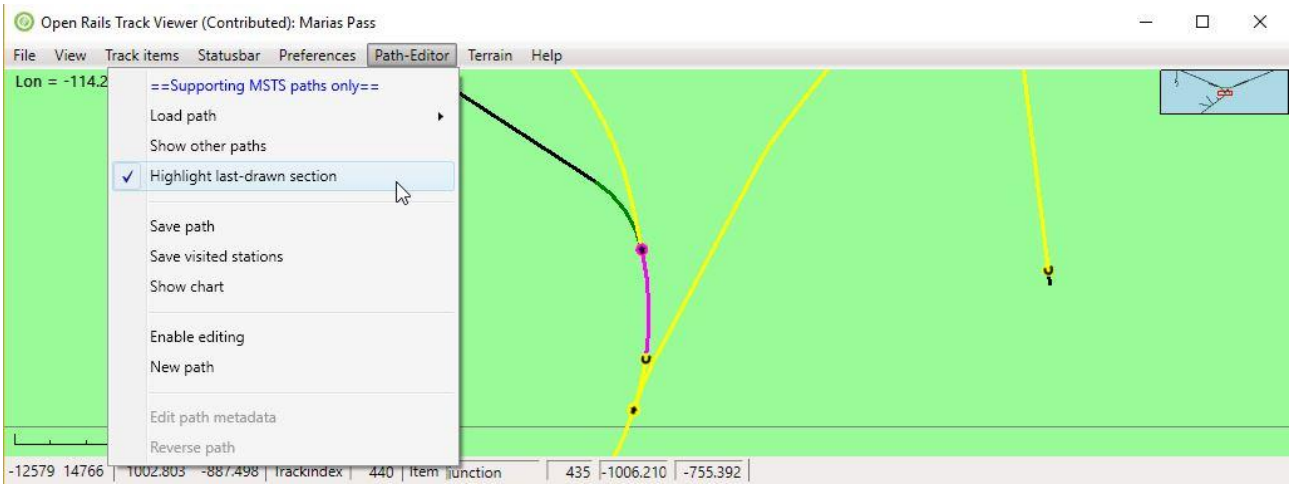
1. Drawing the raw .pat file itself. This will draw a very crude path, containing only straight lines. The only information that is used is the information from the .pat file (so without link to the track database). This is more of a legacy feature and perhaps a fall-back in case of other issues.
2. Drawing 'processed path' is the more advanced and perhaps currently the most sensible feature. It really draws the path over the track, following all curves.



Initially the whole path is drawn. How much of the path is drawn can be changes as follows:

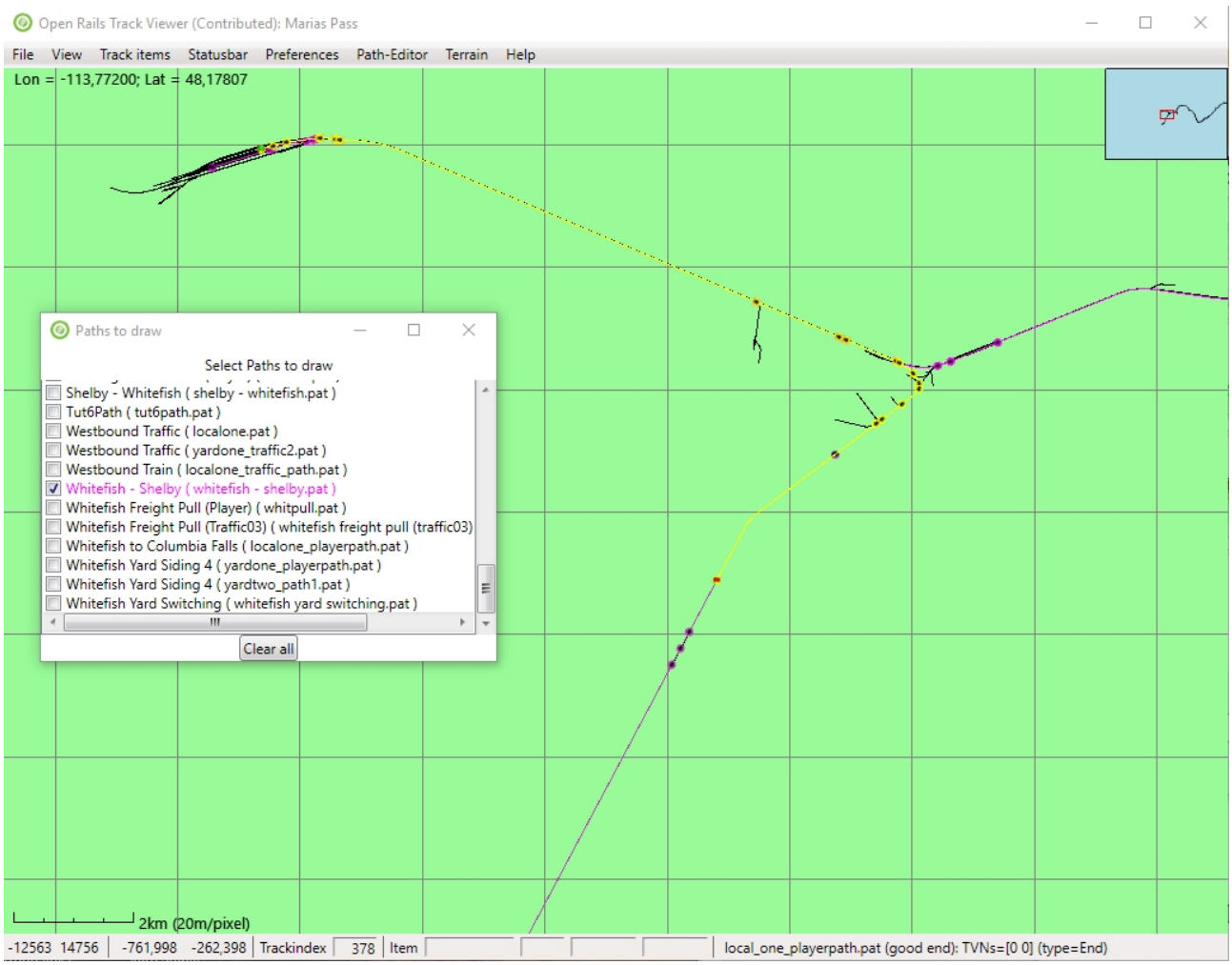
- PageUp draws more of the path (until the full path is drawn)
- PageDown draws less of the path.
- Shift-PageUp will draw the full path,
- Shift-PageDown will only draw the starting point.
- Using the key 'c' will center the view window on the last drawn point of the path. It can be kept pressed during zooming or during the decreasing/increasing the path length.

For complex paths it is sometimes difficult to see until where a path is drawn. To improve this it is possible to highlight the last drawn section of the path. This makes it much clearer what exactly PageUp and PageDown are doing. It also then makes it less likely that you do not have enough of the path drawn when doing editing (see below).



5.2. Drawing other paths

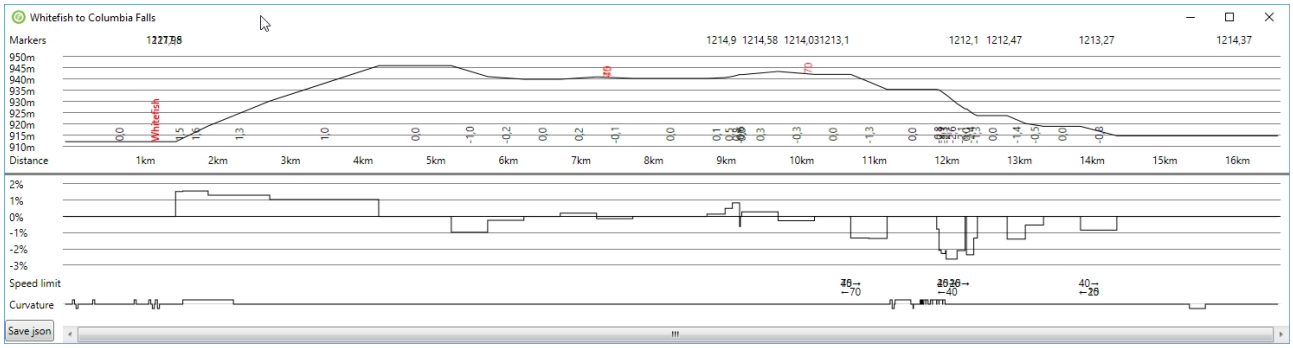
Normally you would only look at a single path at a time. However, to get better context, and perhaps to show both a player path and one or more paths for AI trains, it is possible to draw other paths as well.



The other parts are drawn in another color (purple/pink in this case). When drawing multiple alternative paths, each of them gets a slightly different color (automatically). In the image above, in total three paths are shown.

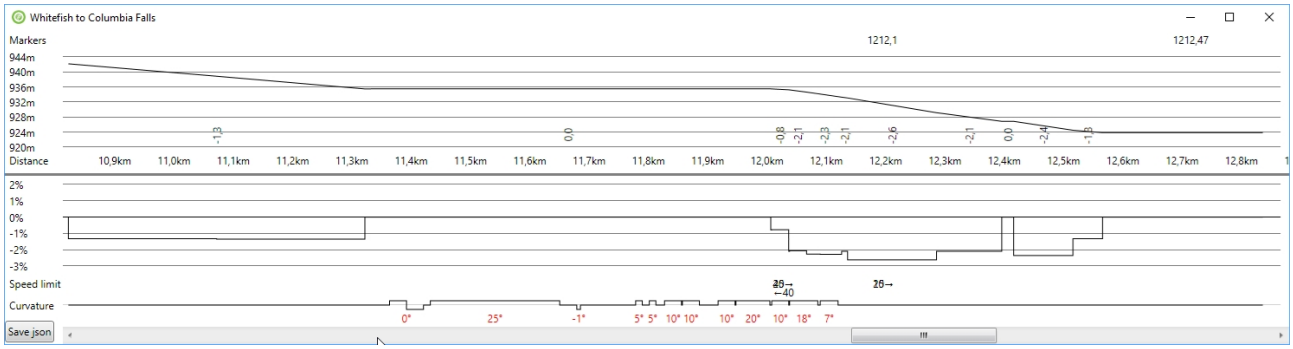
5.3. Path chart

For a given path it is possible to show a so-called path-chart.



The top half shows the vertical profile (height about sea-level), together with station names (in red). On top of it markers are shown (mileposts?). The gradient (in %) is shown with numbers that are rotated by 90°. Below that is the distance along the path.

The top half shows the gradient again but now graphically. Below that are the speed limits (with arrows indicating the direction. The bottom part shows the curvature. A deviation above the line shows a curve to the right, below the line shows a curve to the left.



Note that zooming in and out (e.g. with the mouse) works on this path-chart as well. The distance along the track is dynamically updated to always show where along the path you are. When the zoom level is large enough, the bottom part not only shows the direction of the curve, but also the amount of curve (in degrees, shown in red).

For more information on these charts, see http://www.elvastower.com/forums/index.php?/topic/26623-two-suggestions-for-trackviewer/.

At the bottom left is a button called ‘Save json’. This will save the chart information into a .json file. That .json file can then be used to create a web-page (e.g. as part of a route distribution), showing the chart information. Creating that web-page has not been released as software in the ORTS source code, but an initial version is available in the elvastower forum above.

6. Editing a Path

The Path editor is intended to be an editor of MSTS paths. It was not intended to be an editor for new ORTS-only paths and activities (because when TrackViewer was created the idea was that ORTS-specific path editor and activity editor would be built from scratch including different file formats and functionality). The editor is also not intended to be an activity editor or consist builder for MSTS.

6.1. Path-editor menu

The path-editor menu contains various items to deal with paths.

Without editing, you can still load a .pat file. Once loaded the initial part of the path will be shown. You can center the last drawn node in the visible window with 'c'. You can also save the path. Note that the .pat file so created will have a different ordering of nodes than the original .pat file (the path editor will create a normal ordering).

To enable editing you either have to load a path from file and then enable editing (under the path-editor menu), or you have to create a new path (which will enable editing for you). It is also possible to edit the meta-data of the path (like name, ID, start location and end-location).

On the Path-editor menu there are also a few preferences that are related to the path-editor (the same preferences are also in the Preferences menu).

6.2. What is a path?

A path is defined using a set of nodes. All of these nodes should be on a track location. Most of the nodes will be on top of a junction in the track. But start, end, reversal, and wait points will be nodes that are located on vector sections.

A normal path will just be a series of nodes. Think of it as a directed graph. But some paths have siding paths that are like a parallel path for a certain distance (between two nodes). The MSTS format only allows two parallel paths (so maximum 1 siding path).

6.3. Extending a path: making it longer

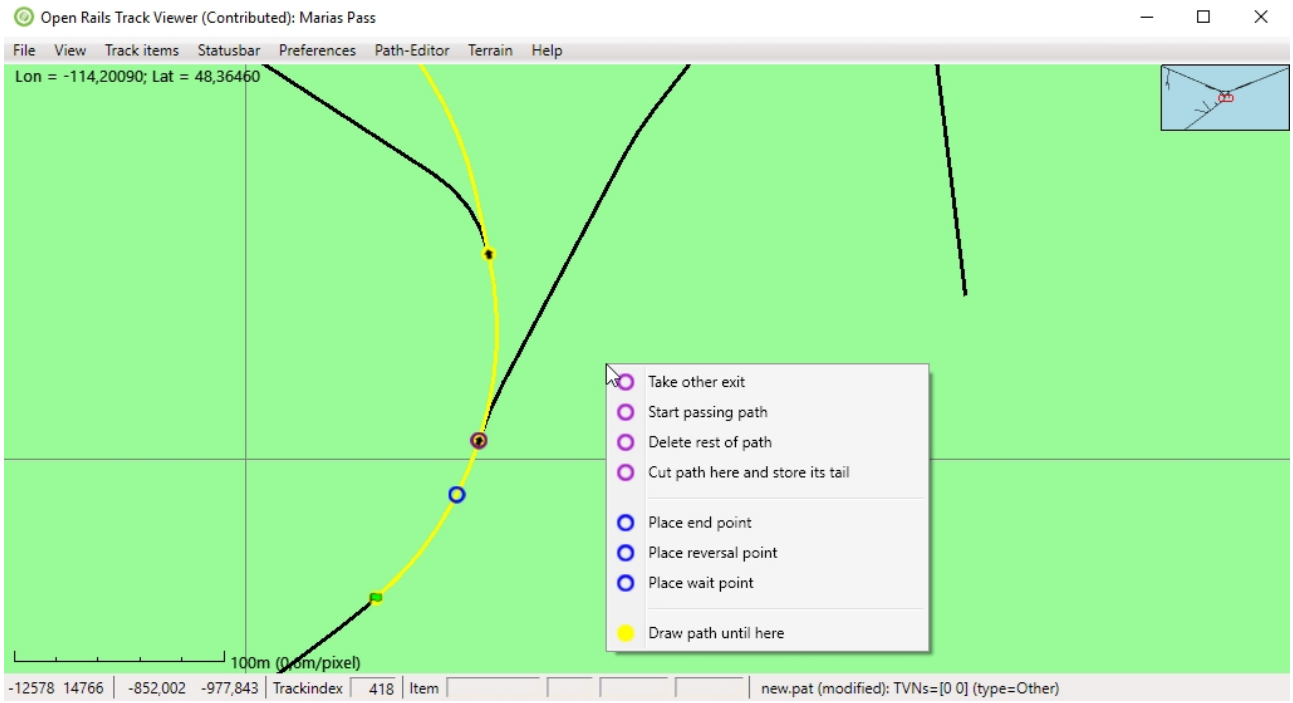
Originally making a path longer was done in the same way as was done for just drawing a path, see section 12: Shift-PgUp would show the whole path that already was created, without making it longer. And PgUp would draw one additional node at a time, and if the path was not yet long enough, add 1 node. In other words, the only way to make a path longer was using PgUp repeatedly.

This also shows up when creating a new path: originally when placing a the start-point, only 1 additional node was drawn. Possibly, this feature of TrackViewer was the least intuitive for new users. Especially since other editors did not show this behavior: there when placing a start-point immediately a long stretch of path would appear.

For that reason, now, it is possible to change that behavior. There is a new preference ‘Shift-PgUp also extends path’. Selecting this preference and then pressing shift-PgUp will extend the path with a lot of nodes. In many cases this will be as far as it is possible to go (so until the track ends somewhere). But to prevent a run-away situation in the case of loops, the maximum number of added nodes is 100. The same happens when placing a start point or reversing the start-direction: immediately a long path will be visible.

Please note that when the path has become too long for you to make it easy to edit, it is always possible to use the action ‘Delete rest of path’ from the context menu (see below).

6.4. Editing actions and context menu



The editor recognizes two kinds of 'active' locations (both identified with a small ring around the relevant location)

1. The node that is closest to the location of the mouse is called the 'active node'. Note that only actually drawn nodes can be active. The reason for this is that this makes it possible to deal with nodes that have the same location, for instance when the path has been reversed and goes over the same junction(s) again.
2. The location on the track that is closest to the mouse is called the 'active track location'. If there are no nodes defined (so for a new path), this can be any location. If there are nodes defined, only drawn track can contain an active track location.

The context menu will popup when clicking on the right mouse while editing is enabled. It will contain actions that can be performed

- for the current active node (purple)
- for the current active track location (blue)
- related to broken paths
- for the complete path (yellow)

6.4.1. Active track location

Actions related to the path itself

- Place start point. This will create the first point of a new path.
- Place reversal point. Place a point where the train will need to reverse. This will remove any activity-related points that have been defined further along the path.
- Place end-point. Note that we make a distinction between what happens to be the last node, and has actually been defined as an end-node. Once an end-node has been defined, actions that would invalidate this end-node are not allowed (with the exception of course of

removing the end-point itself). This is to prevent accidental changes.

Actions related to activities on the path.

- Place wait point. Place a point where the player doing the activity will have to wait. A popup will appear where you can edit the meta data of this point, which consists of the time to wait. For MSTS that is the only thing a wait point does. There are, however, a number of ORTS-specific advanced shunting options that you can define.

For all of these four points (start, end, reversal, wait) it is possible to change the location by dragging the point. To do this, first press the Control button, and then drag the point with the left mouse button pressed. Release the mouse button before you release the Control button, otherwise the action will be canceled.

6.4.2. Active node

Actions related to the path itself

- Change start direction. The initial direction of the path, once a start point has been added, depends on the default in the track database. There is about 50% chance that you would like the other direction.
- Take other exit. For default direction through a facing junction is the main route as defined in the track database (actually, tsection.dat). Taking the other exit simply changes the current exit taken from the junction. When there is no end-point defined, you have a lot of freedom to do this. When there is an end-point taking the other exit is only allowed in those situations where the editor can find a way to reconnect to the existing path itself. For more complex situations, use Cut and store tail (see below). For nodes on passing paths 'Take other exit' is also allowed, but only when the point where the passing path and the main path reconnect is not changed.
- Add passing path. Instead of taking the other exit, it is also possible to add a passing path (also called siding path). This is also only allowed in those situations where the editor can find a way to reconnect to the existing path itself. For more complex situations, use Start passing path (see below).
- Start passing path. Start a complex passing path here. The path will be broken until the passing path has been reconnected. For details see below.
- Reconnect passing path. Reconnect the passing part started with 'Start passing path' to the main path. For details see below.
- Cut and store tail. For complex restructuring of paths (including dealing with broken paths), this will basically cut the path into two at the current location. This location and the rest of the path will be stored (we call this the 'tail'). You can then edit the first part of the path in any way you like, until it is time again to recombine the first part with the tail. For an example, see below.
- Auto-connect to tail. Use this to reconnect from the current node to the tail that has been stored using 'Cut and store tail'. Note that any nodes after the current active node will be lost (which is great if contains broken nodes).

Actions related to activities on the path

- Edit point data. This will popup a menu to edit the meta data of a wait point.

Actions related to removing a point

- Remove end point: This will remove the end-point. Note that paths loaded from file will

always have an end-point when loaded.

- Remove reversal point. Simply remove the reversal point. Obviously this will impact the track quite significantly.
- Remove wait point. All meta-data will be lost.
- Remove passing path. Remove the passing path that starts at this junction.
- Remove start point. This obviously will lead to an empty path. But the tail (if present) will be kept.
- Clear path (keep tail). One of the things you might want to do, after you created a tail, is removed the whole path before the tail. This is especially useful if you want to create a new start to a long path. This action will do the same as 'remove start point', but it does not need that the active node is actually the start point.
- Delete rest of path. This deletes, from here, the rest of the path. This will remove the end-point as well, and therefore open up editing actions that are not allowed when the end-point is defined (like taking an other exit that would otherwise be destructive for a large part of the path). Note that the more-or-less the same effect can be reached by removing first the end-point and then taking an other exit, remove reversal, or other action. But for this you do not first to move the end-point and then back.

6.4.3. Broken nodes

Actions related to fixing the path

- Fix invalid point. A point can be defined as invalid in the .pat file using a certain flag. This will lead to a broken path by definition. But when the path is not actually broken, one can just remove the flag making it a valid point again.
- Autofix broken nodes. This is the main way to fix broken nodes. When this option is available TrackViewer found a way to connect the last good point to the next good point. Selecting this option will create that path. For an example, see below.

Actions to find where the path is broken

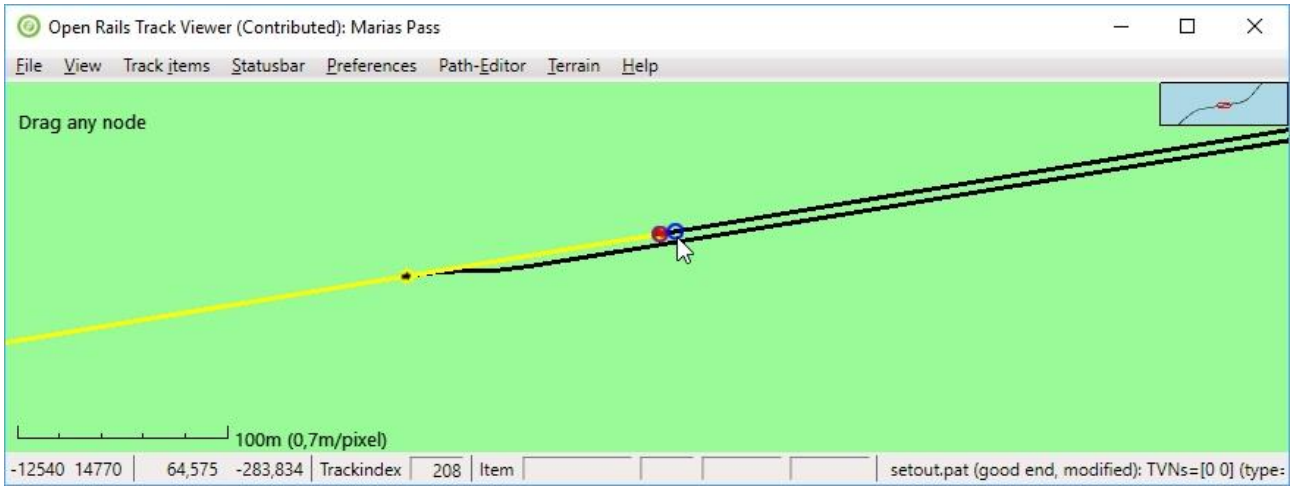
- Draw to next broken point. This will extend the drawing of the path such that it just includes the next broken nodes. For long broken paths it is not always easy to find where all the broken parts are. This allows you to search for it. It is not always working perfectly, especially for broken passing paths, so you might need to extend or shorten the drawn path a bit (PageUp and PageDown).

6.4.4. All nodes

Drawing related

- Draw path until here. This does not affect the path itself, only which part of the path is drawn. When the full path is drawn, for each location only the last drawn node can be active. In situations where the path goes multiple times over the same track (e.g. due to reversal nodes), this can be an issue. It can easily be solved by reducing the number of nodes actually drawn. However, for long paths it is very inconvenient if the path can only be extended or reduced by only one node at a time. Draw path until here allows you to select until which node the path will be drawn (and reduced or extend again from that node).

6.5. Dragging with the mouse

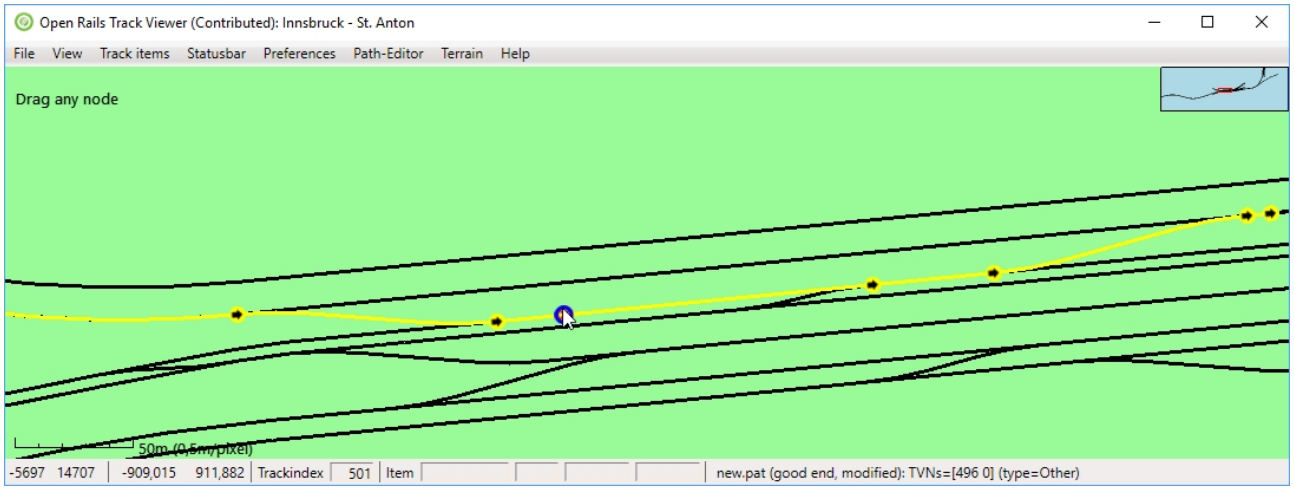


Dragging happens when you first press ctrl and then click the left-mouse button and move the mouse. The dragging action can be canceled by releasing the ctrl-key before the left-mouse button. If you press ctrl, on the top-left of the window there will be a description of the action that will be taken when start to press the left-mouse button. There is a preference you can set to make this action visible. The most important dragging is to drag a 'vector' node (start, end, reversal or wait point). This allows you to fine tune the position of any of these nodes without having to delete and replace them. In the picture above this would mean moving the start point left or right along the track it is on.

But, dragging is actually more advanced than just that. You can also drag the point to a nearby parallel track. In this case you could drag it directly to the siding below the current start point.

That dragging to a nearby track is also possible in the middle of a path (so away from a start, end, reversal, or wait point). This makes it possible to do an action like 'take-other-exit' using dragging. So you can do bigger changes in the path in this way in one go. What really happens is that a temporary node is added once you press the left-mouse button (it will show as a broken node, see the figure below. That temporary node can then be dragged, just as the start point above. Dragging it along the track does not make sense, because the path will not change. But you can drag it to a nearby path (in the example below to the track just below it), thus making the path take another exit.

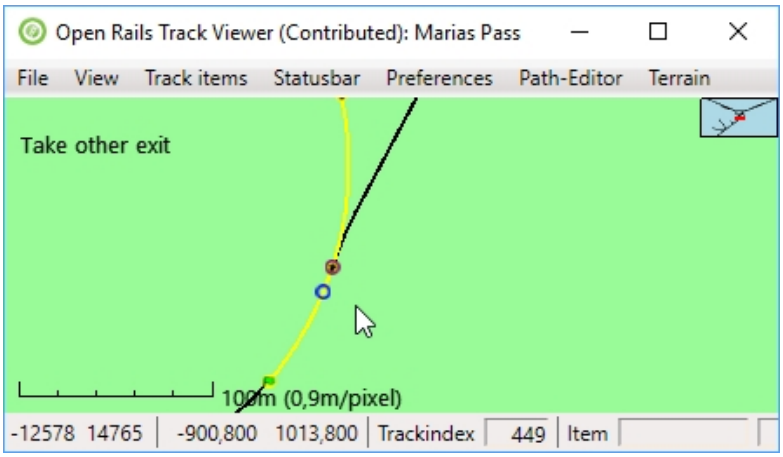
Above you can see the situation at the beginning of the drag operation. Below is what happened after you dragged the mouse a little big down.



6.6. Performing actions with the mouse

It is also possible to do a single action with a left-mouse click. For this you need to keep the shift-key down, and then click the left-mouse button. The possible actions are:

- Add start point
- Change the start direction
- Take other exit
- Take other exit for a passing path
- Auto fix broken nodes
- Remove passing path
- Edit a wait point
- Remove a reversal point
- Remove the end point



Although there is a list of possible actions, only one of these actions is the current active action. As with dragging, on the top-left of the window there will be a description of the action that will be taken when clicking the left-mouse button. In the figure this is ‘Take other exit’. The list above is also the priority list: the first action that is possible will be the active action.

6.7. Modifying long paths

To be able to modify long paths and prevent having to redefine large portions that you do not want to change, a number of features are available. Since, however, they might not all be completely intuitive, here is a short manual

6.7.1. Changing the start

Changing the start but leaving the rest can be accomplished as follows

- First go to the first node that you still want to keep. Select 'cut path here and store its tail'. This will effectively save the rest of the path. You are now free to do edits to the initial part.
- If you really want a different start, the easiest is to now select, on any node, 'Clear path, keep tail'.
- Then you have to place a new start point.

- Increase the path as you want.
- Finally, you can select 'Reconnect to tail'. TrackViewer is able to find a re-connection over a distance of 20 junctions. So if you just want the 'default' connection, you can use this as soon as you are within this distance. If you only change the starting track in a station, you can do this immediately after placing the start point (and even if the start direction is oriented in the wrong direction.

6.7.2. Changing something in the middle

Changing something in the middle is not very different from changing the start, and actually also has similarities with changing the end. The easiest steps are

- Go to the first node that you still want to keep, after the part you want to change. Select 'cut path here and store its tail'.
- Then go to the last node before the change that you still want to keep. Select 'Remove rest of path' (this should still keep the tail).
- Do the path editing as you need.
- Finally, select 'Reconnect to tail'.

6.7.3. Changing the end

Changing the end by 'take other exit' is often not possible. The reason for this is that 'destructive' changes (changes that would remove the end-point) are not allowed when there is a defined end-point. Once you know this, the solution is simple:

- Remove the end-point.
- The do all the editing you need.
- Add an end-point.

Sometimes the end point is still far away. To make it simpler, you can also do:

- Remove rest of path. This will remove the end as well, and therefore opens up all kinds of editing activities.
- Again, do all the editing you need.
- Add an end-point.

6.8. Creating passing paths

Passing paths are alternative routes along sidings that are next to the main path. In TrackViewer they are drawn in orange.

There are two ways to create passing paths (both described in more detail below):

- 'Add passing path': this will create a passing path as long as a path can be found following only the main track of every switch (and as long as not too many junctions have to be crossed. You have no freedom to determine where it will reconnect to the main path.
- Complex passing paths can be made using first defining where the passing path must start and then defining where the passing path needs to reconnect.

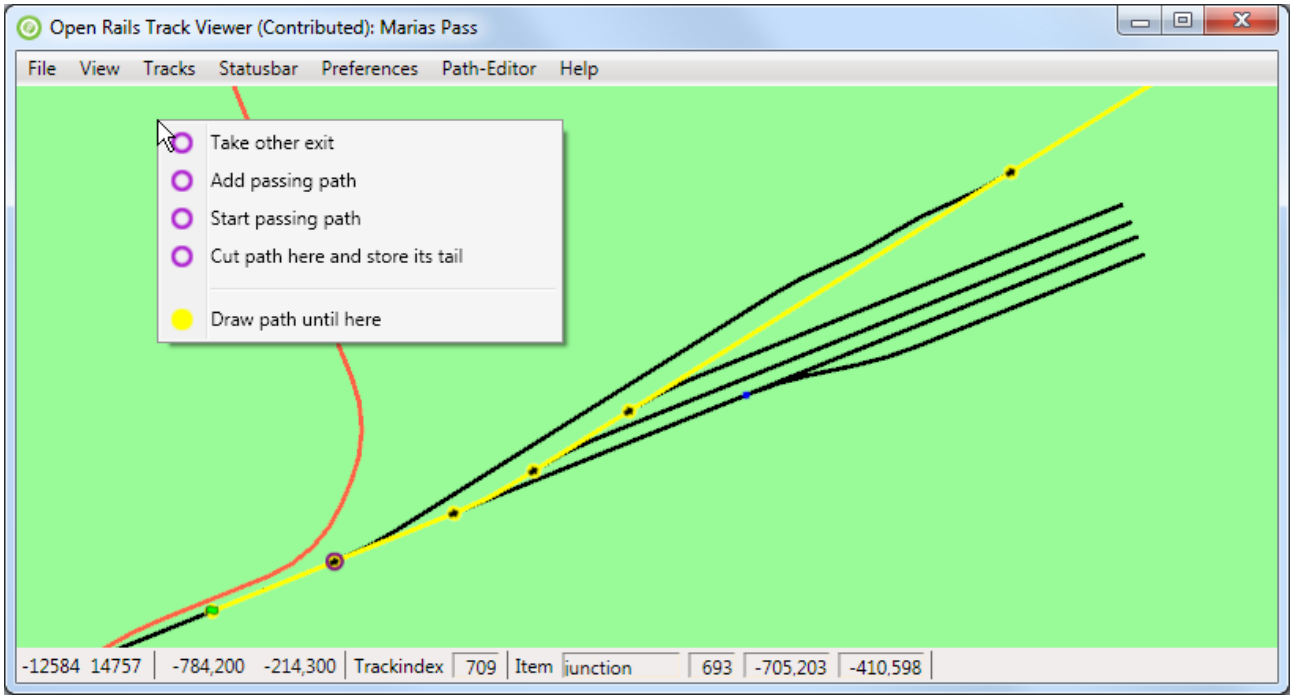
Some limitations related to passing paths:

- Once a passing path has been created, it can be changed but only as long as the start and end of the passing path remain at the same place (always junctions)

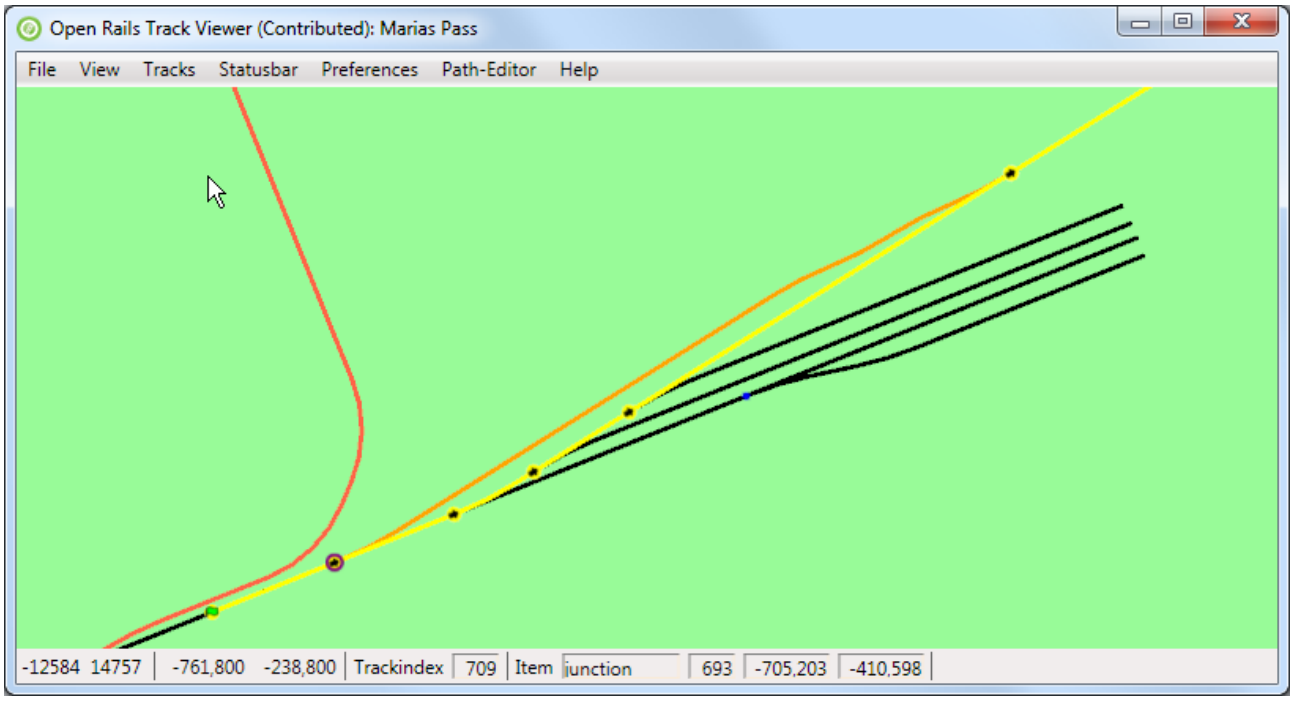
- There are no start, end, wait and reversal points allowed on either the passing path or the main path that runs in parallel.

6.8.1. Simple passing paths

Creating a simple passing path is simple: Just make sure the node where you want to start the passing path is active, right click, and select 'Add passing path':

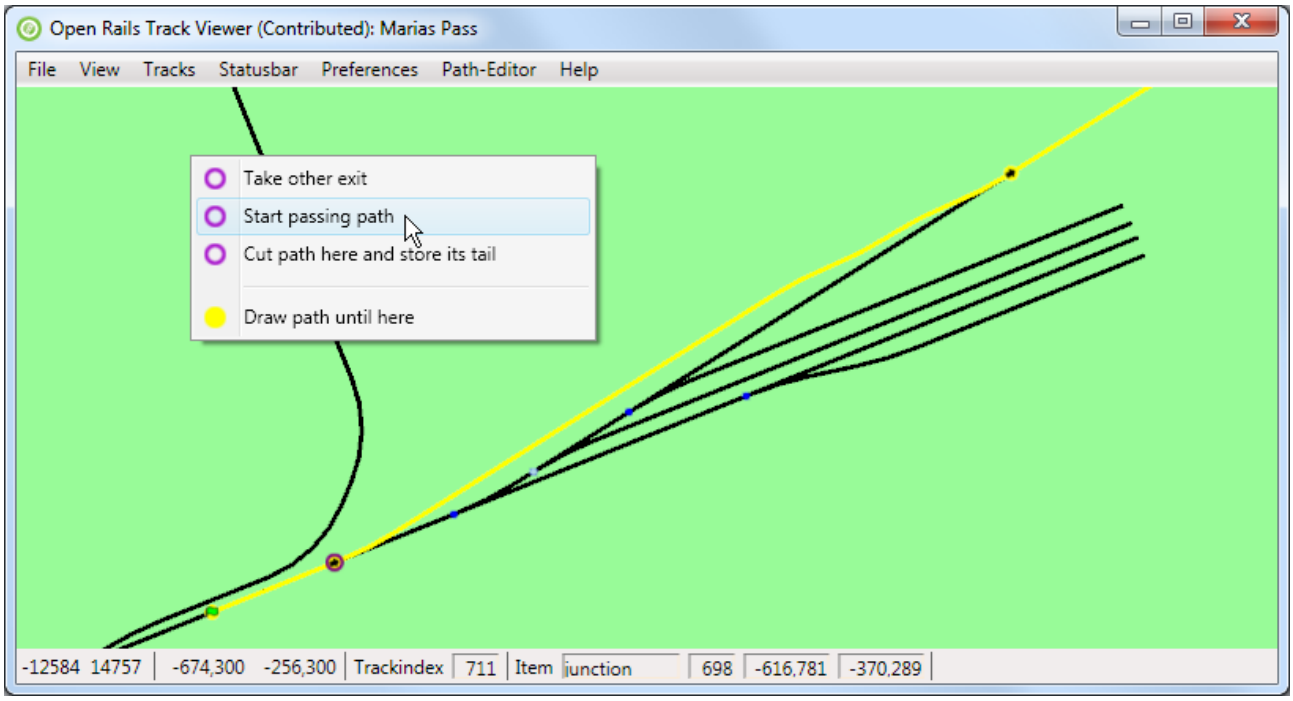


The result is shown below: the passing path has been added (in orange). In this case it is a very simple passing path without any nodes.

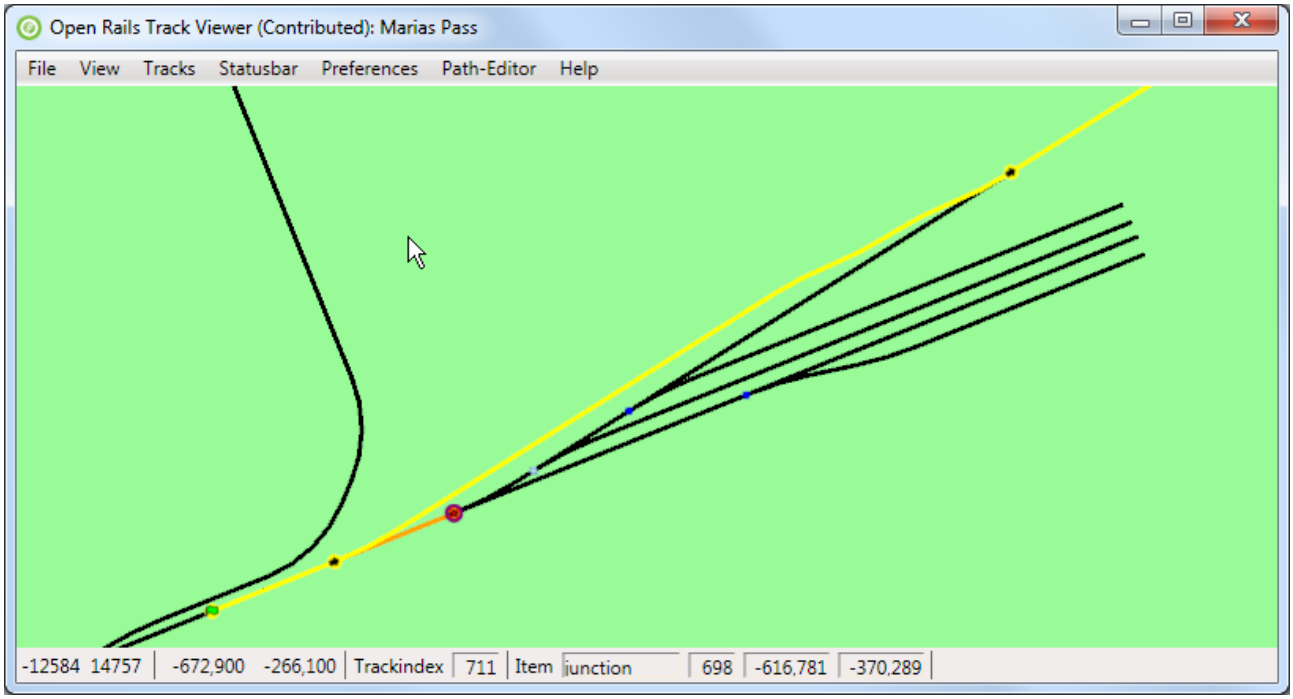


6.8.2. Complex passing paths

When we start in the same area as above, but with a path along the top track, we can not add a simple passing path:



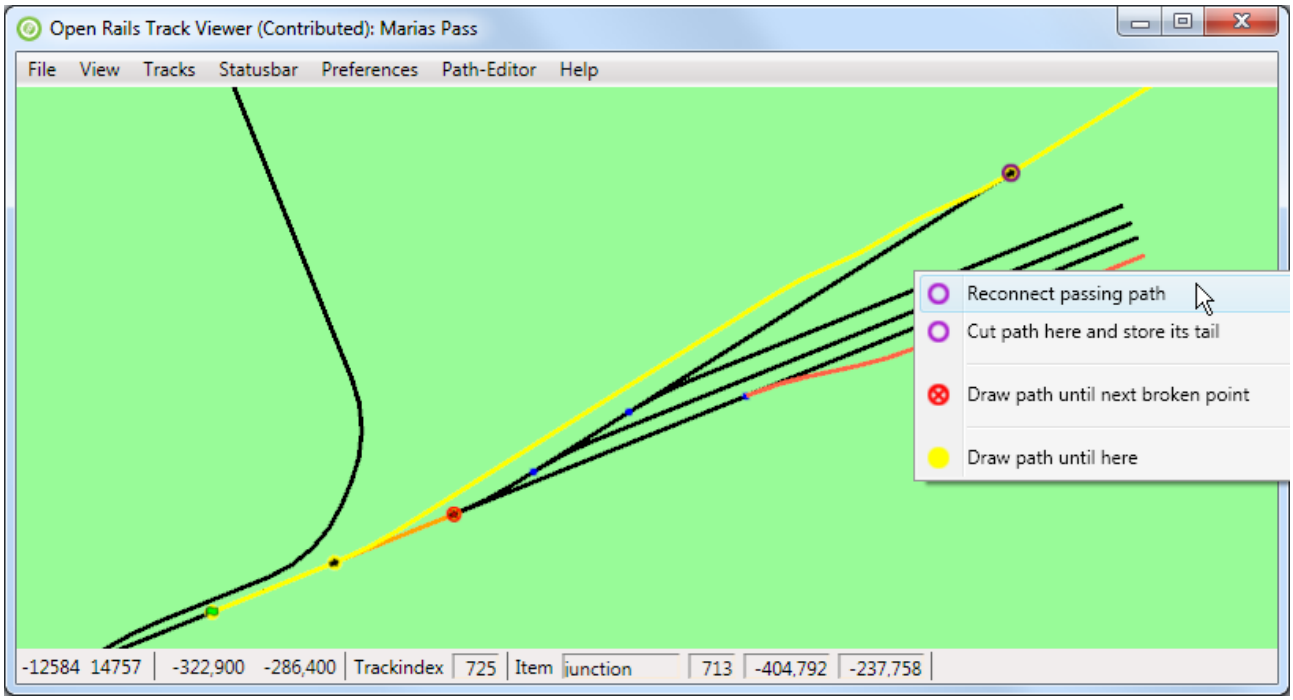
Starting the passing path at the same point, and following only main tracks at each junction would not lead to a reconnect-point, but only to one of the sidings. To create a passing path here the first step is to 'Start passing path':



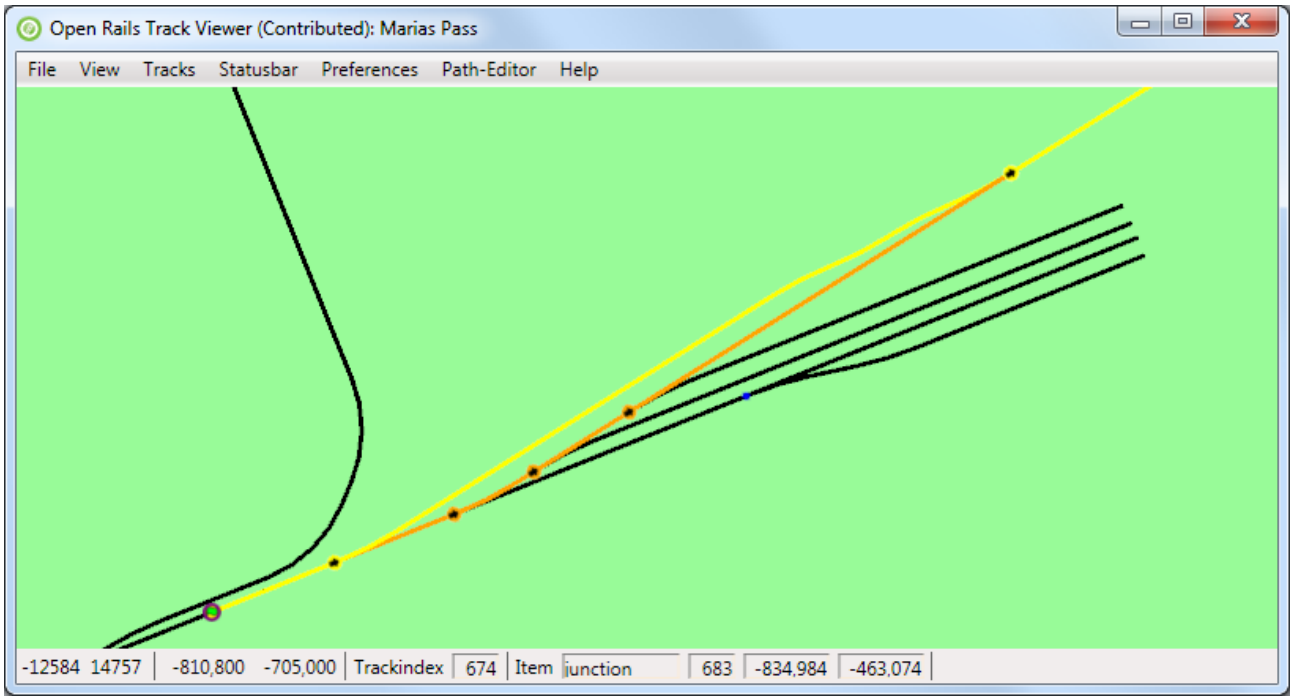
This will create only the first part of a passing path. The path is now broken (as can be seen from a cross through the just created node on the passing path.

The next step is to go to the node on the main path where you want the passing path to reconnect. The command here is 'Reconnect passing path'. TrackViewer will search for a path between the start

of the passing path (the node just created) and the point where you want to reconnect. Following main tracks at each junction has preference. Actually, when there is the option to 'Reconnect passing path', TrackViewer has already found a path. So if you do not see the option to reconnect, apparently it is not possible.



After reconnecting you get the passing path as shown below.



6.9. Broken nodes and paths

The MSTS .pat file stores the various nodes using the location. This makes it independent of the index a junction or track happens to have in the track database, which is good. Sometimes, however, the track database has been updated (tracks have been updated or moved, junctions have been added

or removed, …). In those cases a stored .pat file might suddenly be broken: a defined path node can no longer be linked to a correct track location. Broken nodes will be indicated by a cross through the node. Furthermore, the path can no longer be drawn along the track: straight lines will be used.

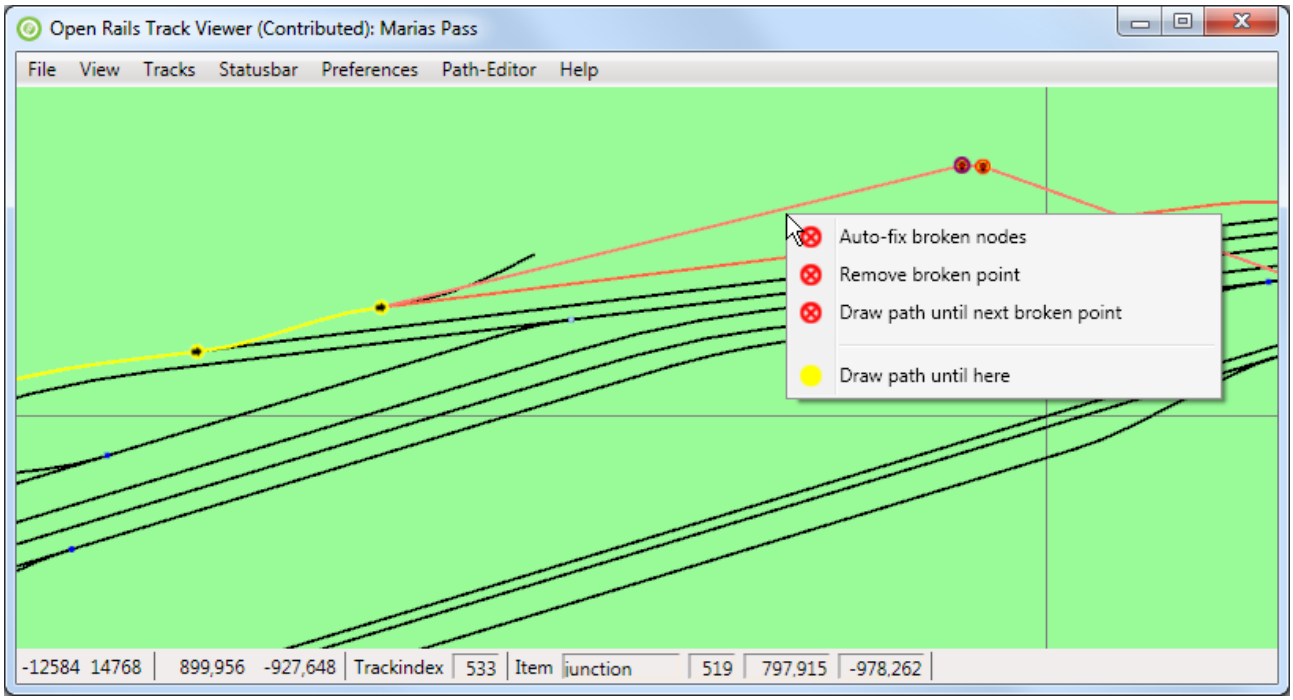
To correct broken paths, the following three options are available:

- For situations where the passing path is broken, the solution is to remove the passing path and rebuild it.
- For situations where the fix is pretty straight forward, TrackViewer can auto-correct the broken path itself.
- For other complex situations, use the 'cut-and-store-tail' functionality.

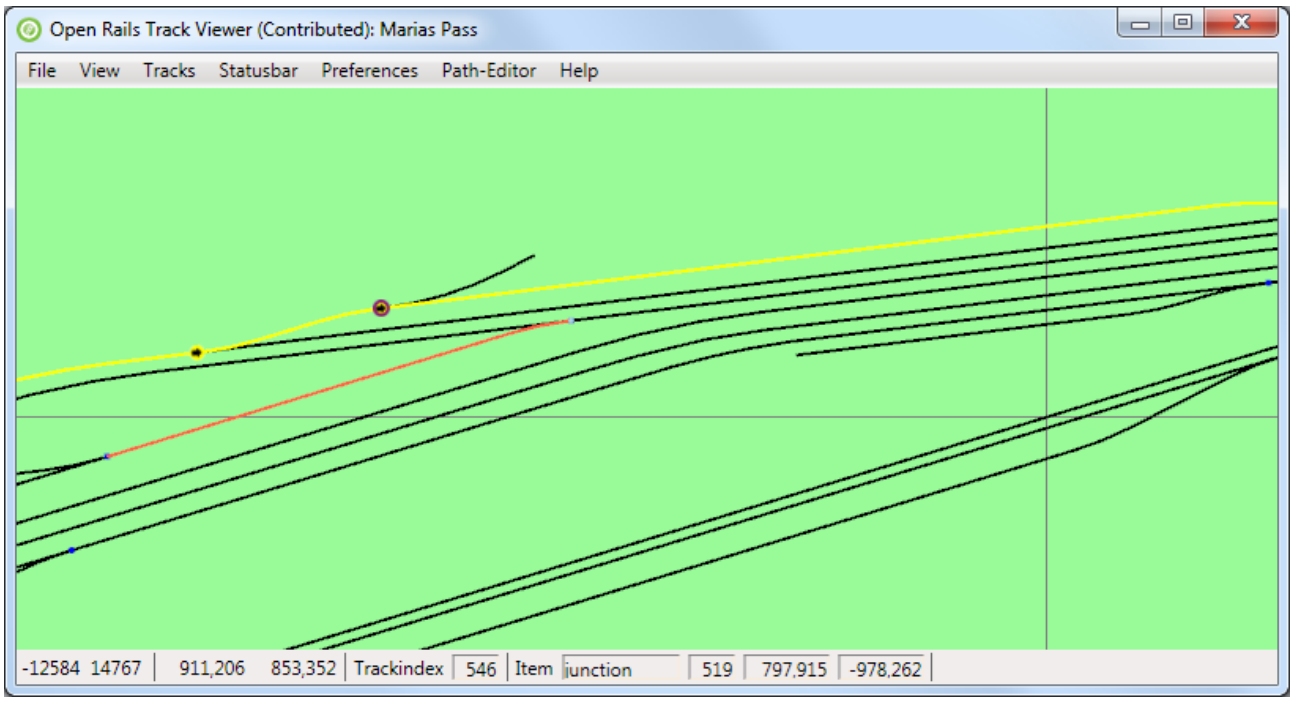
Both are described in more details below.

6.9.1. Auto correct broken nodes

Consider the following situation:



As one can see the path is fine and drawn in yellow until a certain node. Then, on the top right there are two broken nodes. Apparently the tracks have been changed quite a bit after this path was created. Right clicking on one of the relevant nodes (that would be either the broken nodes themselves or the good nodes just before or after the broken path) will have, in the popup-menu, the item 'Auto-fix broken nodes'. Selecting this will auto-fix the broken nodes. The result is then:

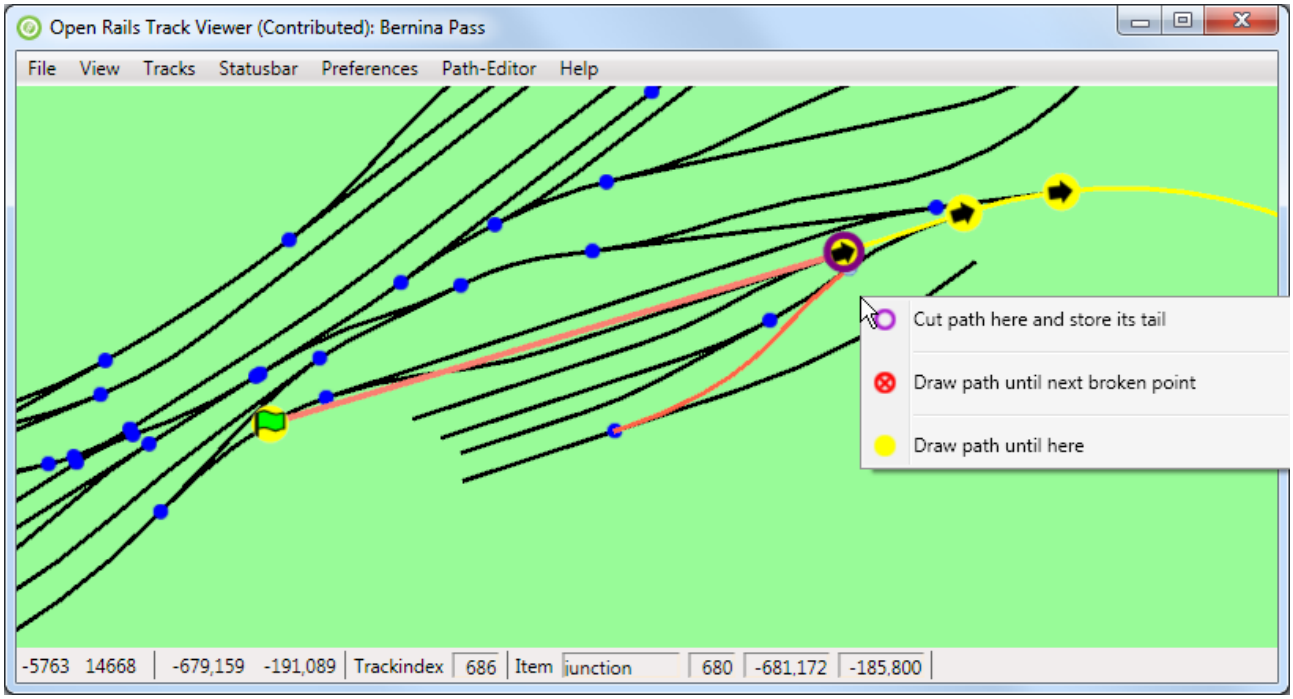


As one can see, the path has been fixed.

6.9.2. Fixing broken nodes in more complex situations

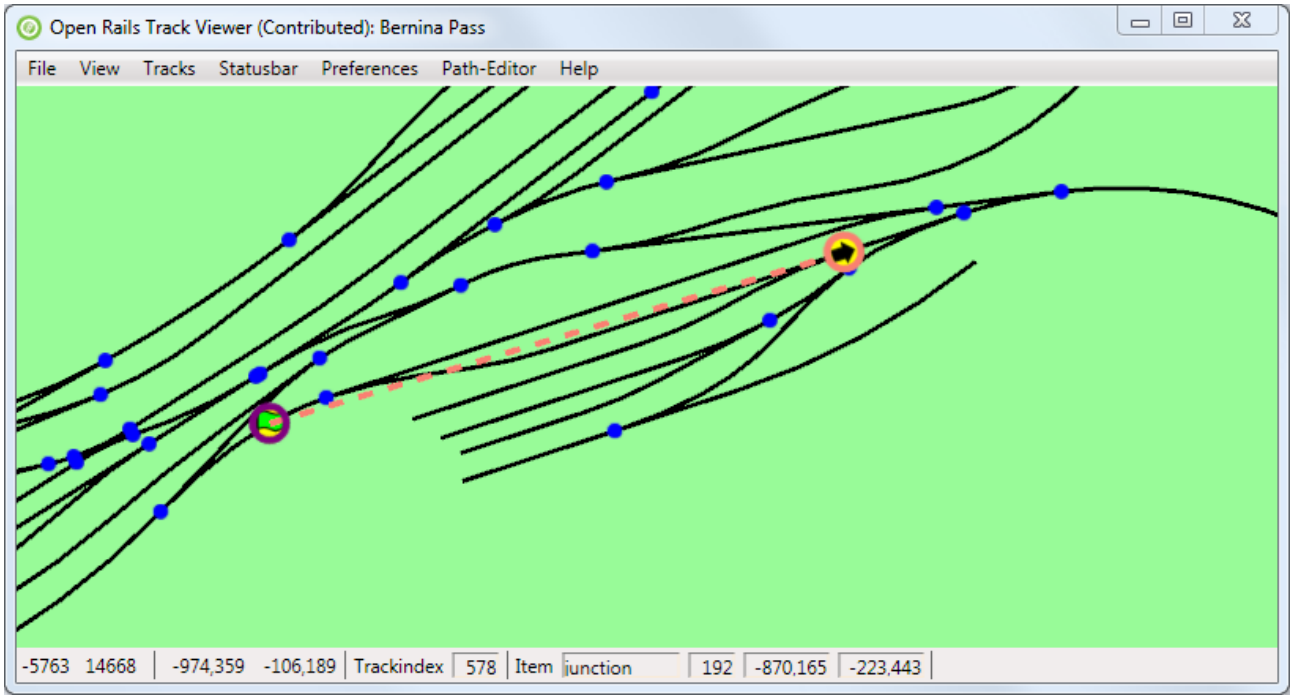
Sometimes it is not possible to auto-fix the broken nodes. Most likely this is due to a situation where either the node before or after the broken section is not directed correctly. The direction of the path on the track can not always be determined when the path is broken. Then there is a 50% chance of the direction being not what you would like. And then auto-fix won't work.

As an example, see the broken path below:



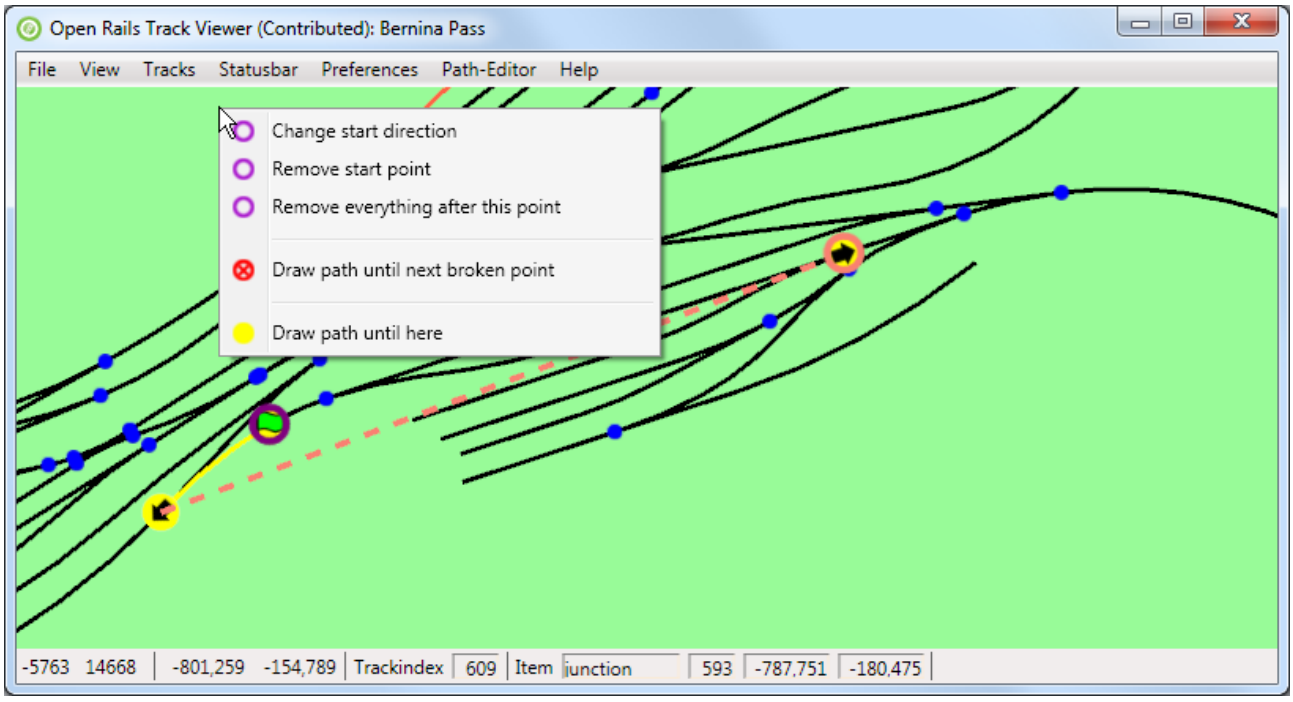
It is obvious that there is a simple connection between the start point and the first good junction

node. The path is broken because the junction just after the start point is not in the path. To fix this path, right click on one of the good nodes after the broken path and select 'Cut path here and store its tail'. This will store the rest of the path (tail) so you can reconnect it later. But at the same time, you have freedom to correct the path in almost any way you like. Note that this also works in case the path is not broken, but you just want to change the initial part of a path in a non-trivial matter.

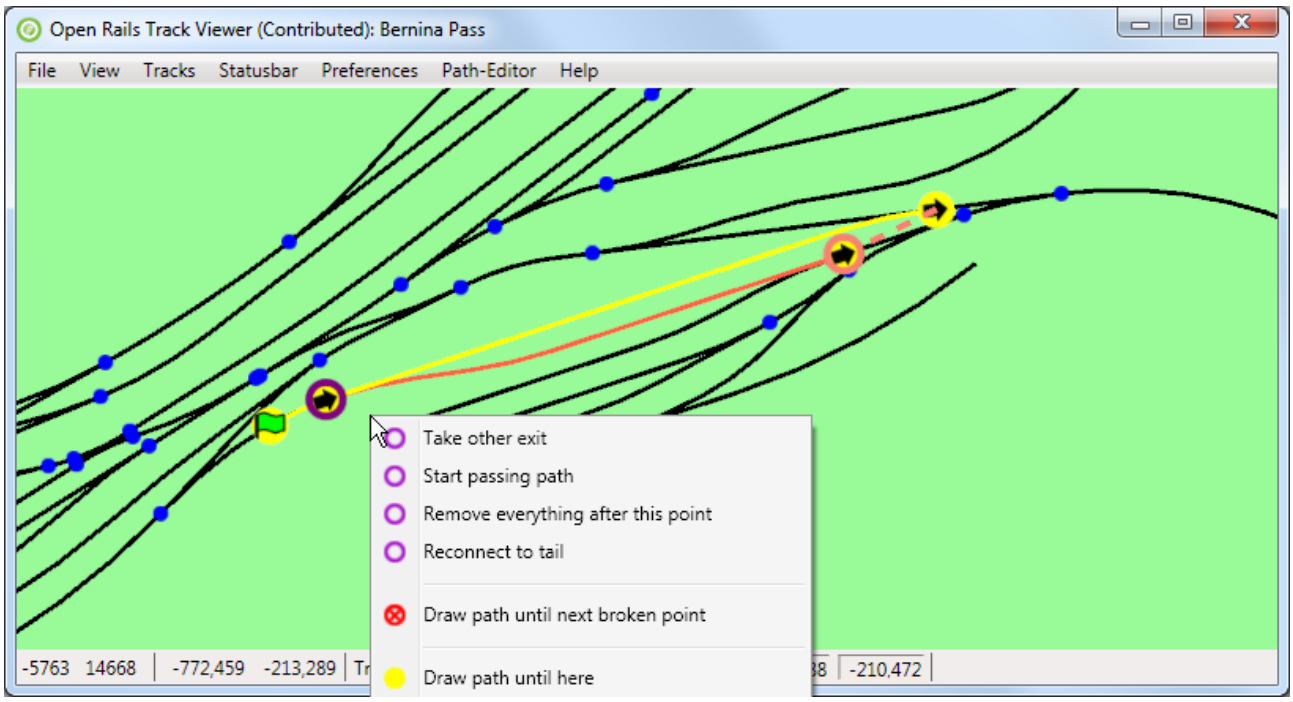


The tail is now drawn using only its first node. Furthermore, there is a dashed line from the last drawn node of the path to the tail. In this way you see where you need to connect again.

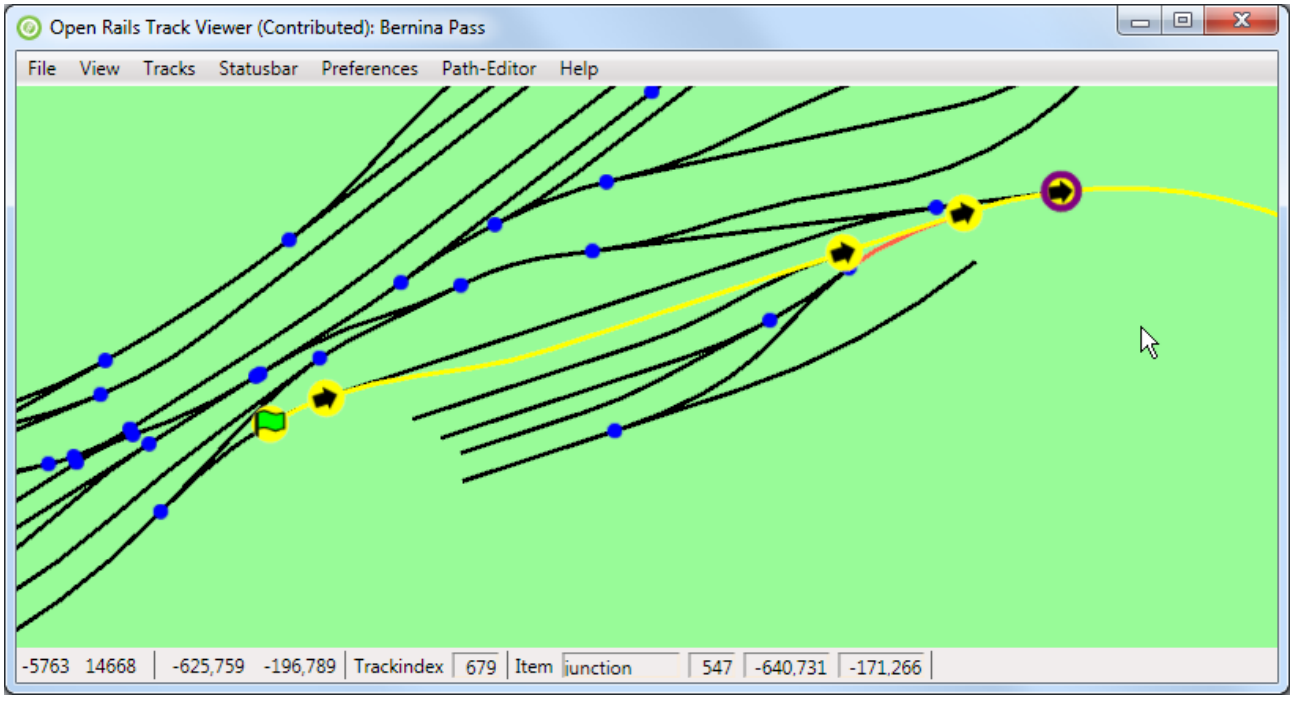
At this point only the start node of path is visible. Adding an extra node to the path (press Page-Up) makes the situation clear:



The direction of the start point is to the lower-left. This makes it impossible to auto-fix the path. The obvious solution is to 'Change start direction':



Now that the direction is reversed, it possible to reconnect to the tail. There is no need to first 'Take other exit'. Just press 'Reconnect to tail'. The result is a fine and completely connected path again:



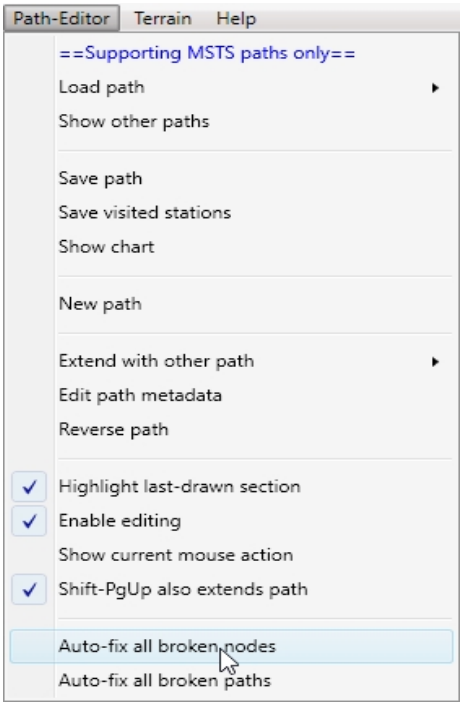
Note that now also all nodes are drawn again. This concludes fixing this part of the broken path.

6.9.3. Auto-fix all broken nodes

During the development of a route it is possible that a change in the track database makes a path broken. Possibly even in multiple places. Using the techniques described above it is possible to fix all broken points one by one. But it is also possible to (try to) auto-fix all broken nodes in a path via the Path-Editor menu.

Obviously there are limits to this. Not all broken situations can be automatically fixed. But both paths that are broken due to invalid points or due to reasonably limited track changes can be fixed.

6.9.4. Auto-fix all broken paths



Obviously, if a change in the track database makes that a certain path is broken, it is also likely that other paths are also broken. If you have many paths then it is tedious to go fix all of them one by one.

For that reason there is now the possibility to Auto-fix all broken paths. For this it is not needed that a path is loaded. The fixing will be done by loading all paths, one-by-one, and trying to auto-fix them. This will lead to a number of modified paths.

Once all paths are done, an overview will be shown on which paths have been modified. You can then decided to indeed save all the modified paths, either overwriting the original files, or popping up a save-window for each modified file.

Do note that sometimes a path that is loaded will immediately be modified. This happens in particular when an additional track node is needed between two junctions to clarify which junction exit is used. That intermediate node might not be present in the .pat file, but TrackViewer will add it. Such a modified path will show up as modified as well, even though it was never broken and nothing had to be fixed.

6.10. Limitations

The fact that there are limitations in the path editor is related to the limitations in the file format and capabilities of MSTS itself (obviously apart from bugs or the lack of someone willing to implement a feature). Since this editor is intended to yield MSTS-compatible paths (that can also run within ORTS), not all possible features you might think of can be supported. In the (near) future ORTS-specific path definitions and editors will be developed that are not limited by MSTS-compatibility. These, however, are not described here.

7. Keyboard commands and mouse behavior

Some of the keyboard comments can be seen from the menu as well. Here we try to document all of them.

7.1. Viewer

Zooming and moving the view window

- = Zoom-in. Keeping it pressed will keep on zooming in.
- Shift-= Zoom-in one step (for more precise control),
- − Zoom-out. Keeping it pressed will keep on zooming out.
- Shift-− Zoom-out one step (for more precise control),
- Z Zoom-to-tile: zoom to a level where exactly one MSTS tile (2048m × 2048m is visible).
- M Toggle between zooming around the mouse, and zooming around the center.
- R Zoom-reset: view the whole track.
- Ctrl-R Reload the route.
- L Add a label.

- Q Quit TrackViewer.
- Shift-C Shift to mouse: shift such that the current mouse location becomes the new center.
- ← Shift the view-window to the left (the route moves to the right).
- → Shift the view-window to the right.
- ↓ Shift the view-window down.
- ↑ Shift the view-window up.

Toggling what is visible/drawn

- F5 Show speed-limits.
- Shift-F5 Show mileposts.
- F6 Show terrain tiles.
- Ctrl-F6 Show DM terrain tiles (Distance Mountains).
- Shift-F6 Show patch lines of the terrain.
- F7 Show signals.
- F8 Show platforms.
- shift-F8 Show platform-names.
- F9 Show sidings.
- Shift-F9 Show siding-names.
- F10 Highlight tracks.
- Shift-F10 Highlight items
- F11 Show path (needed to be able to edit it!)
- Shift-F11 Show path from raw information in .pat file (which does not use track database). This will not be updated during editing.

Mouse behavior

- Shift & drag left mouse button: Shift the view-window with the mouse movement.
- Ctrl & drag left mouse button: Drag a label.
- Scroll-wheel: Zoom-in or out.
- Shift & scroll-wheel: Zoom-in or out, but slower (to enable more precise control).

7.2. Path Editor

Keys:

- C Shift the view window such that the last drawn-node is centered. You can keep this button down while drawing more or less of the path using the next keys.
- E Place the end point.
- W Place a wait point.
- Ctrl-Z Undo the last edit.

- Ctrl-Y Redo (only possible when at least one Undo has been done).
- PgUp Draw an extra node of the path (unless at the end-of-path)
- PgDn Draw one node less of the path (this does not change the path itself!)
- Shift-PgUp Draw all of the path. Depending of the setting of a preference, this will also make the path much longer.
- Shift-PgDn Draw only the start node of the path.

Mouse behavior

- Ctrl & drag left mouse button: Drag a node
- Shift & click left mouse button: Perform the currently active editor action
- Right mouse button: Open the context menu with editor actions
- Additional mouse button 1: Undo
- Additional mouse button 2: Redo

7.3. Menu

For the menu various alt-? keys are supported. Further menu navigation can then be done using arrow keys. The exact short-cut is unfortunately not always visible. Please press alt for a while to show these (they will have an underscore). Note that because TrackViewer is programmed as an XNA game, but with a WPF menu structure on top, not all standard WPF features work completely normal. Hence the minor issues with the menu.

8. Future development

Any future development depends on the wishes and needs of the community. I created ORTS TrackViewer as a debugging tool initially (working on paths), and it grew into something much more.

The following items have already been requested. It is unlikely that these will be added.

- Currently it is already possible to search for tracknodes and trackitems. Possibly it would be nice to have problems in the route directly available from the viewer (instead of getting this information from OpenRailsLog.txt).
    - For ORTS v1.3 the current plan is to have an independent command-line utility to check relevant files. In that case TrackViewer would not need to do something similar. But it could perhaps give the output of that utility (once available) in case a route cannot be loaded.
- Add possibility to import and export routes. So people can have a look at a route without having it installed.
    - Currently this is already possible by copying global tsection.dat, route-specific tsection.dat, <route>.tdb (and <route>.rdb), and for the moment also <route>.trk.
    - Making an export/import routine would basically mean to make a different file format to write and read (at least part of) the information in these files. It is not clear whether this is worth the effort.
- Make TrackViewer independent of XNA. This prevents people to have install XNA. Currently this is quite a big change, and it would probably also need the code to be independent of ORTS itself. Since XNA is needed anyway for ORTS, this is not likely to

happen.