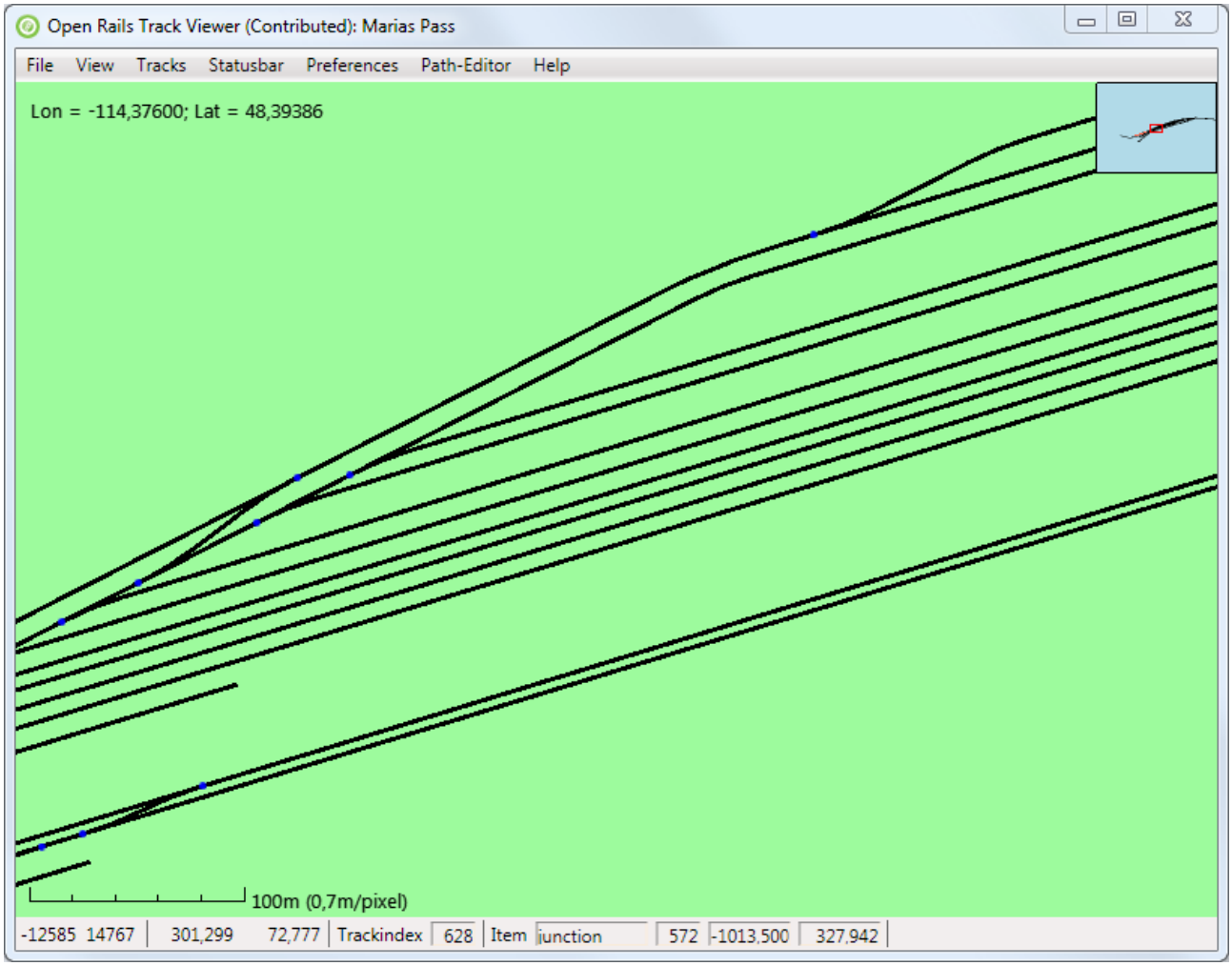
ORTS TrackViewer

Viewing MSTS tracks and editing MSTS paths



Contents

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.

Note that also this documentation is still under development.

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 a recent (experimental) version of 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 an experimental version of ORTS, see its web-site http://www.openrails.org/experimental.html.

Although TrackViewer is part of the ORTS release, it does not have the same status. 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 Open Rails you can use the sub-forums *Discussion* or *Maybe it's a bug*).

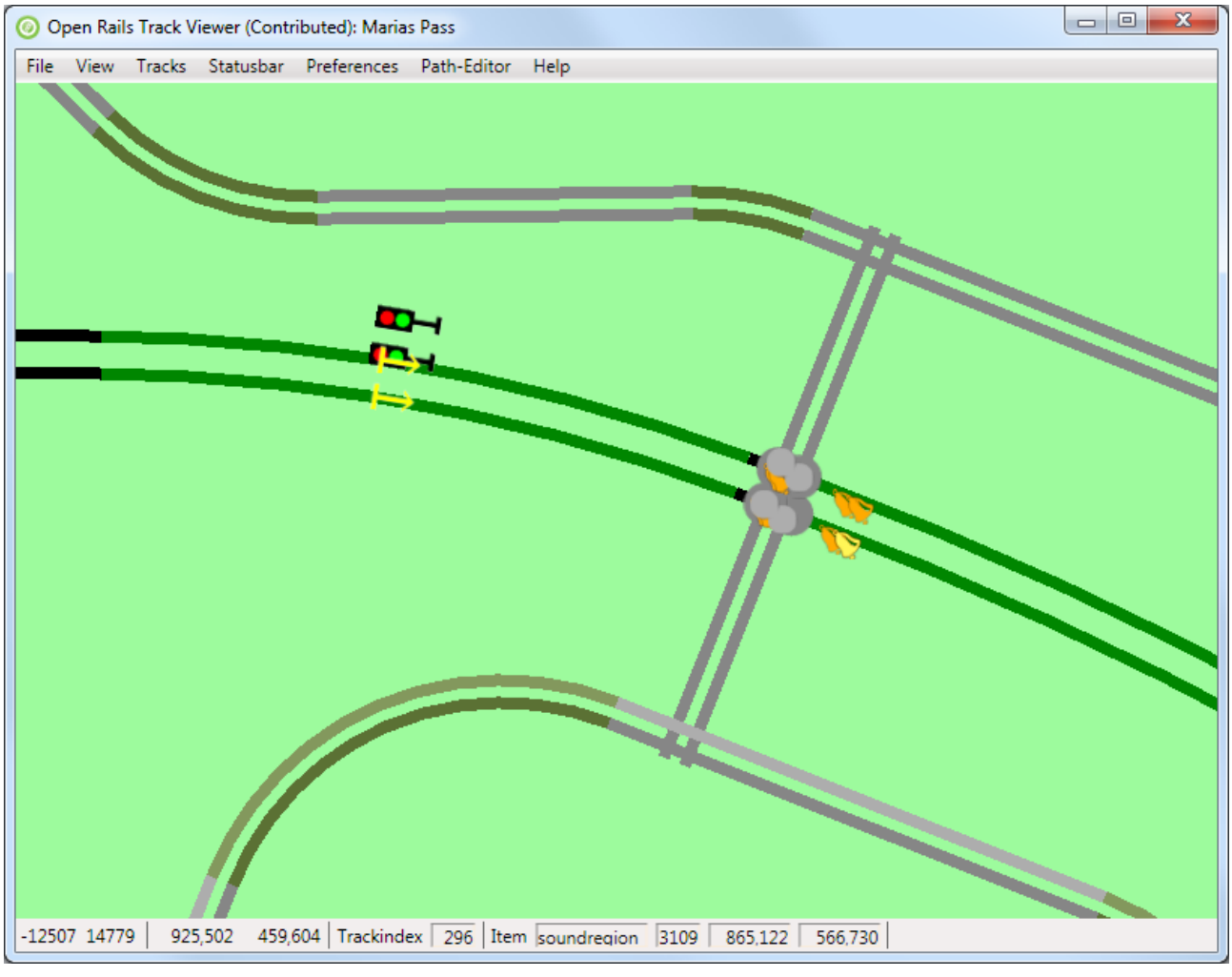
3. Viewer manual

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 directory under the File menu. You can then load a route, again via the File menu. Selecting a different route can also be done via the File menu. After the first use, you can reload your previous route (this was saved).

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

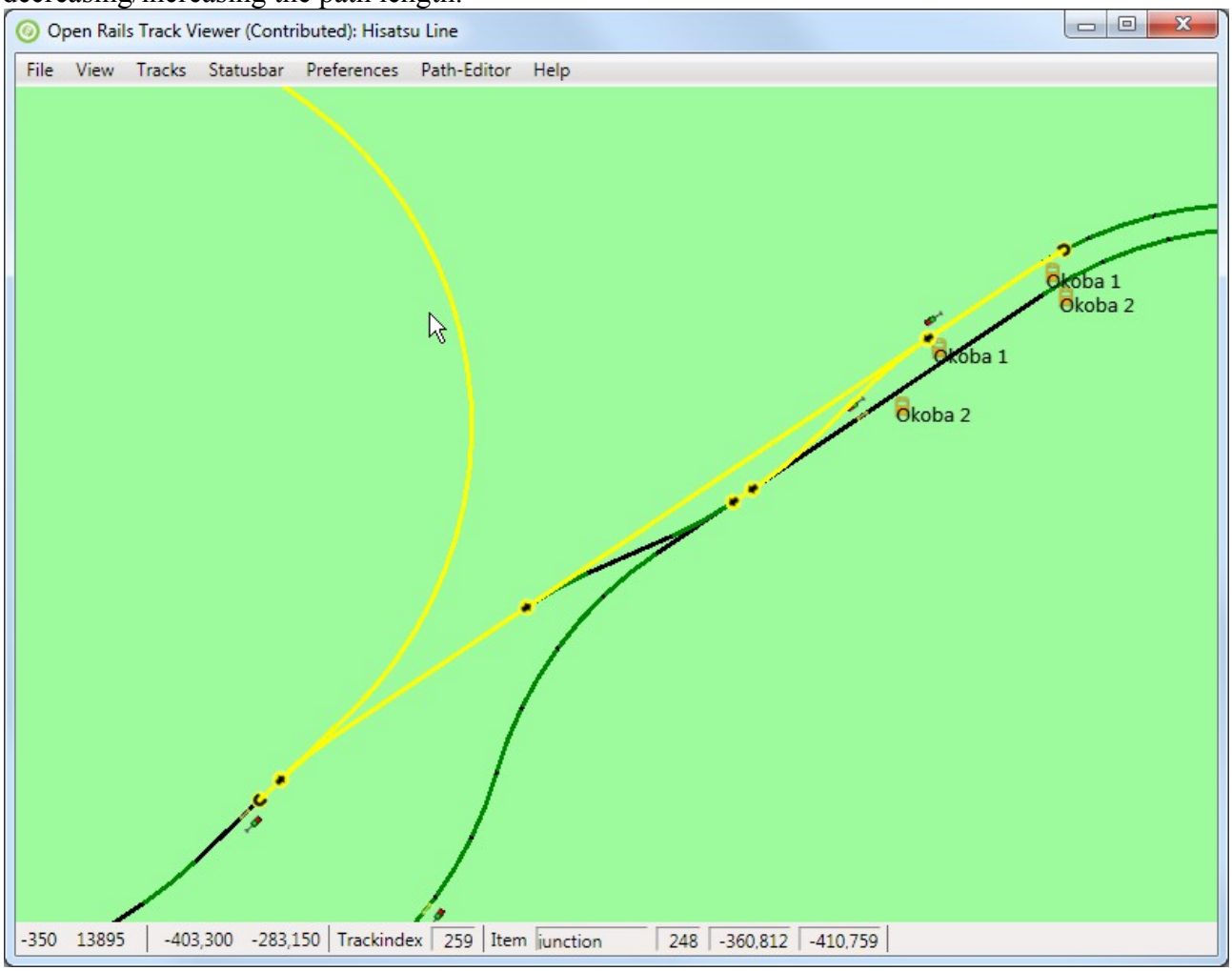
TrackViewer will keep most of your settings. This includes the install directory, the last route, and what you selected to be drawn on screen and what not. In other words, almost all selections work as preferences at the same time.

A number of menu items have keyboard shortcuts (also mentioned on the menu items). Other shortcuts can be found under Help. Zooming and shifting works best using plus and minus keys ('=' and '-') or the mouse wheel. Pressing shift during zooming with the mouse wheel gives you finer control. Pressing shift Zooming is centered around the mouse position normally. Shifting the view



window can be done using the keys a-s-d-w, or with the mouse (left-mouse button pressed).

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 directory of the route and have extension .pat. Drawing can be done in two ways. Drawing the raw .pat file itself will draw a very crude path, containing only straight lines. Drawing 'processed path' is the more advanced feature and uses the ORTS AIPath code. Initially the wholle path is drawn. The length of the drawn path can be decreased/increased with the PageDown and PageUp keys. 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.

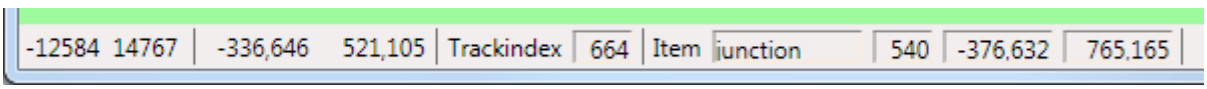


3.1. Status bar

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. Please note that the other information might still change in the future, since the information is not necessarily very relevant for many people.

3.1.1. 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

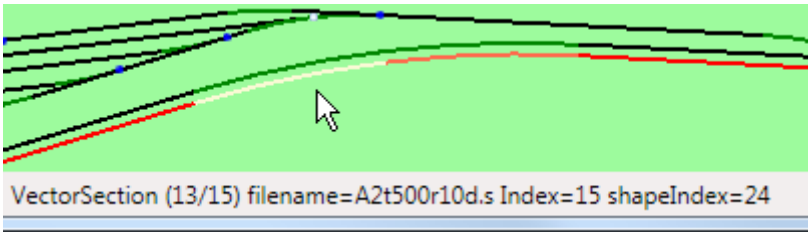
Locations in MSTS and hence ORTS are stored using a tile-based system. Tiles are square area's of 2048m × 2048m. The x-direction is along east-west axis. The z-direction is approximately along south-north axis. The y-directions is the height above the reference level (not used in Track Viewer).

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).

3.1.2. Trackvector information

A track in the track database is stored as a so-called vectornode. Such a vector node consists of a number of sections, each one normally 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 developpers, but not for general usage.

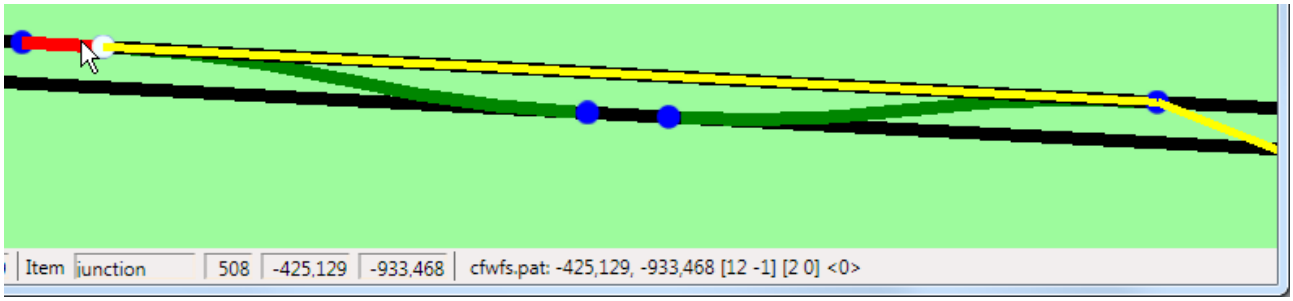


3.1.3. Raw .pat file information

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 statusbar information for 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.

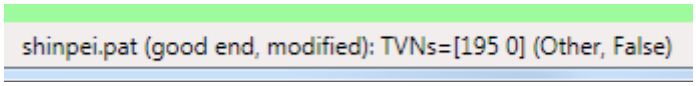


3.1.4. Information on the path

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. Note that when loading a path and then enabling editing sometimes some edits are done automatically to make the path well-defined (according to Track Viewer of course).
- Broken: If the path is broken, this will be shown. 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).

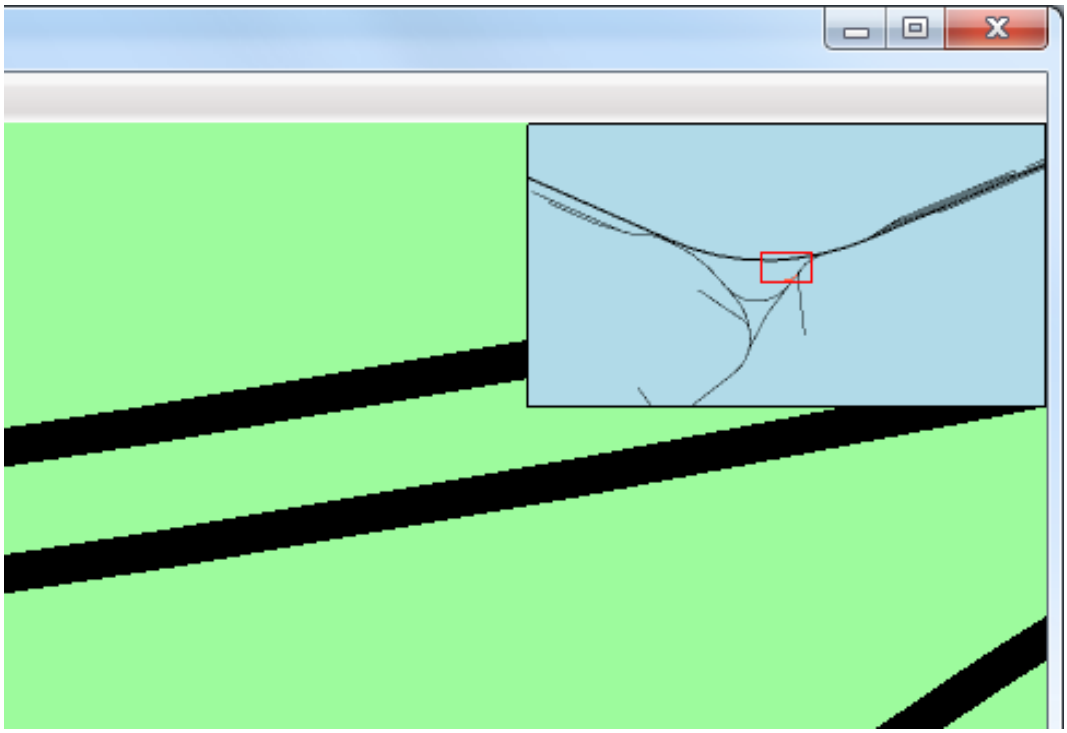
Next some information is shown about the last drawn node (so not the active node during editing, but the last node that has been drawn). First there are the TVNs (Track Vector Nodes, meaning trackNodes that are not junctions or endnodes) of the next main and the next siding path are shown. Between the 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.



3.2. Additional things on screen

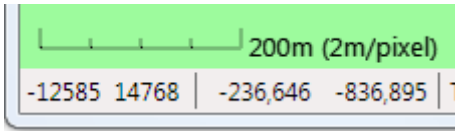
3.2.1. Inset

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



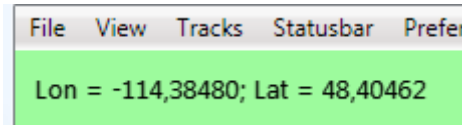
3.2.2. Scale ruler

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.



3.2.3. World location

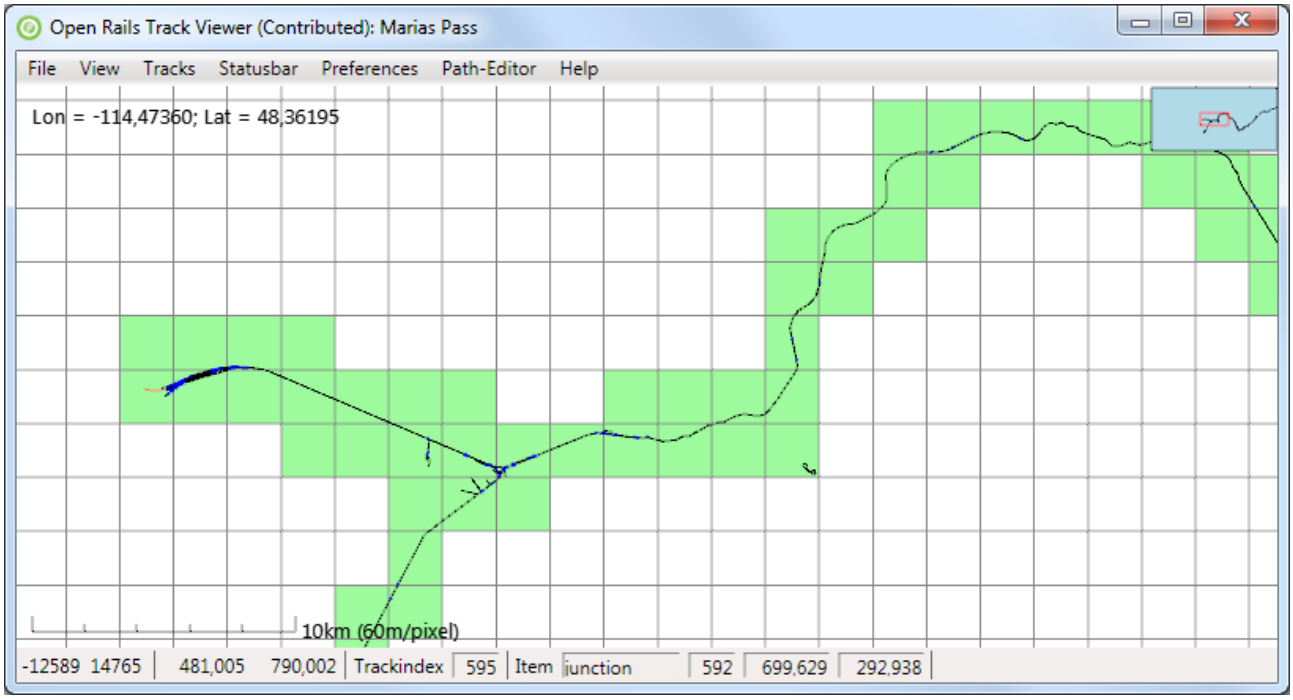
MSTS (and hence ORTS) internally does not store the location in real-world longitude and latitude values, but in tiles (see also the section on status bar above). The projection that MSTS uses to translate real world-locations on a sphere to the flat tile-based locations is called Goode Homolosine projection. 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 numbers shown here might differ slightly from those in MSTS.



It is also important to realize that 'up' in the track viewer is not North.

3.2.4. 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. For some developpers this information might be useful.



4. Path editor manual

At this point in the editor is still under construction. It is the intention to have a working version always, although it might not contain all features yet.

The Path editor is intended to be an editor of MSTS paths. It is not intended to be an editor for new ORTS-only paths and activities (because that will be build from scratch including different file formats and functionality). The editor is also not intended to be an activity editor for MSTS (it will not contain a consist editor for instance).

4.1. 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).

4.2. 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 to the current active node or the current active track location.

4.2.1. 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 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.

4.2.2. 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 start point. This obviously will lead to an empty path.
- Remove passing path. Remove the passing path that starts at this junction.

4.2.3. Broken nodes

Actions related to fixing the path

- Autofix broken nodes. This is the main way to fix broken nodes. When this option is available Track Viewer 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).

4.2.4. 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).

4.3. Performing actions with the mouse

A number of actions are also possible using the mouse (press Control and click the left-mouse button). Only those actions that are the logical and most-used actions are possible. Note that you can always undo a change.

The actions that are possible with the mouse are:

- Dragging a start, end, reversal or wait point.
- Add start point
- Take other exit
- Auto fix broken nodes
- Remove passing path
- Draw until here.

4.4. Creating passing paths

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

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

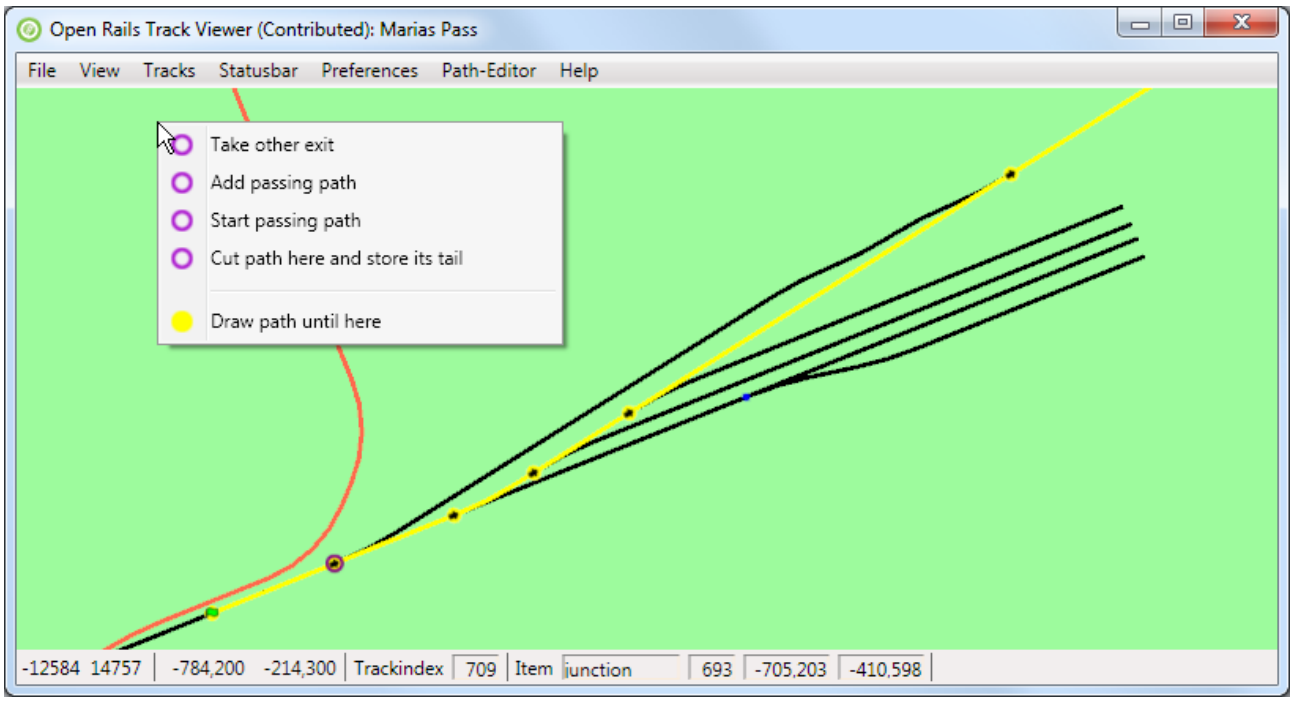
- 'Add passing path': this will create a passing path as long as a path can be found following only the main track of every switch (and as long as not too many junctions have to be crossed. You have no freedom to determine where it will reconnect to the main path.
- Complex passing paths can be made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

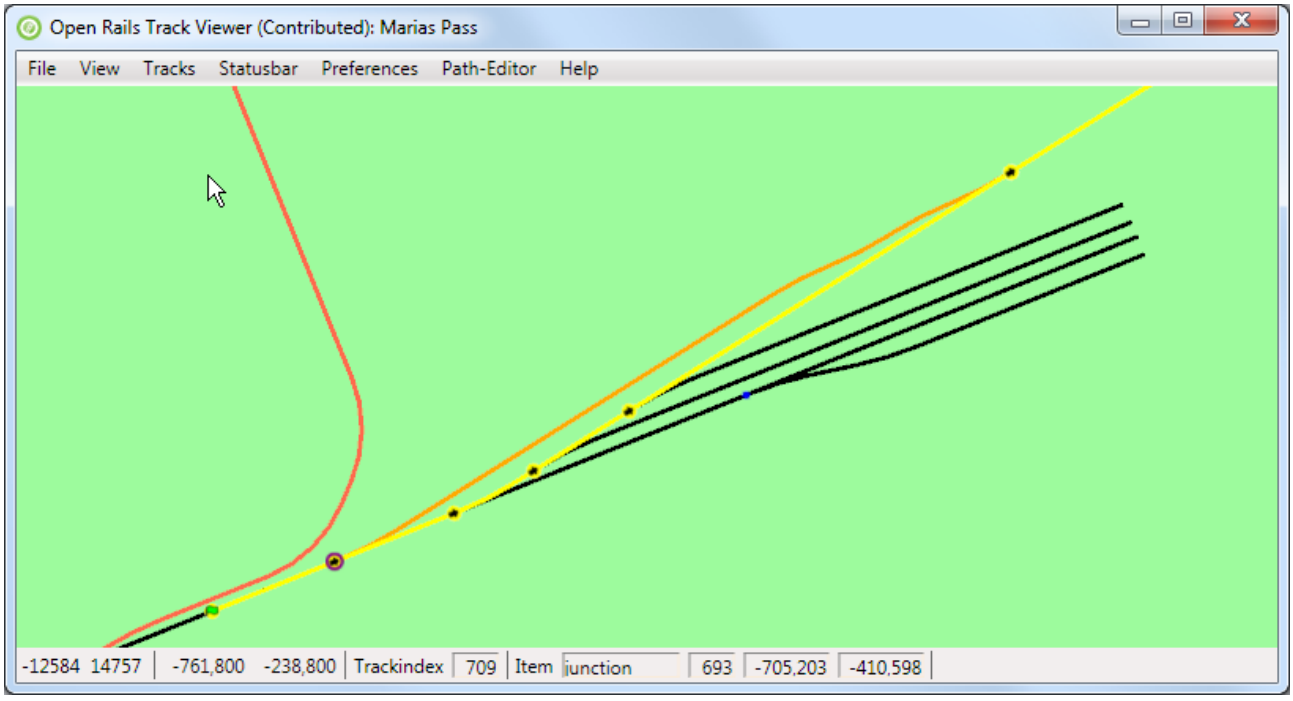
- Once a passing path has been created, it can be changed but only as long as the start and end of the passing path remain at the same place (always junctions)
- There are no start, end, wait and reversal points allowed on either the passing path or the main path that runs in parallel.

4.4.1. 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':

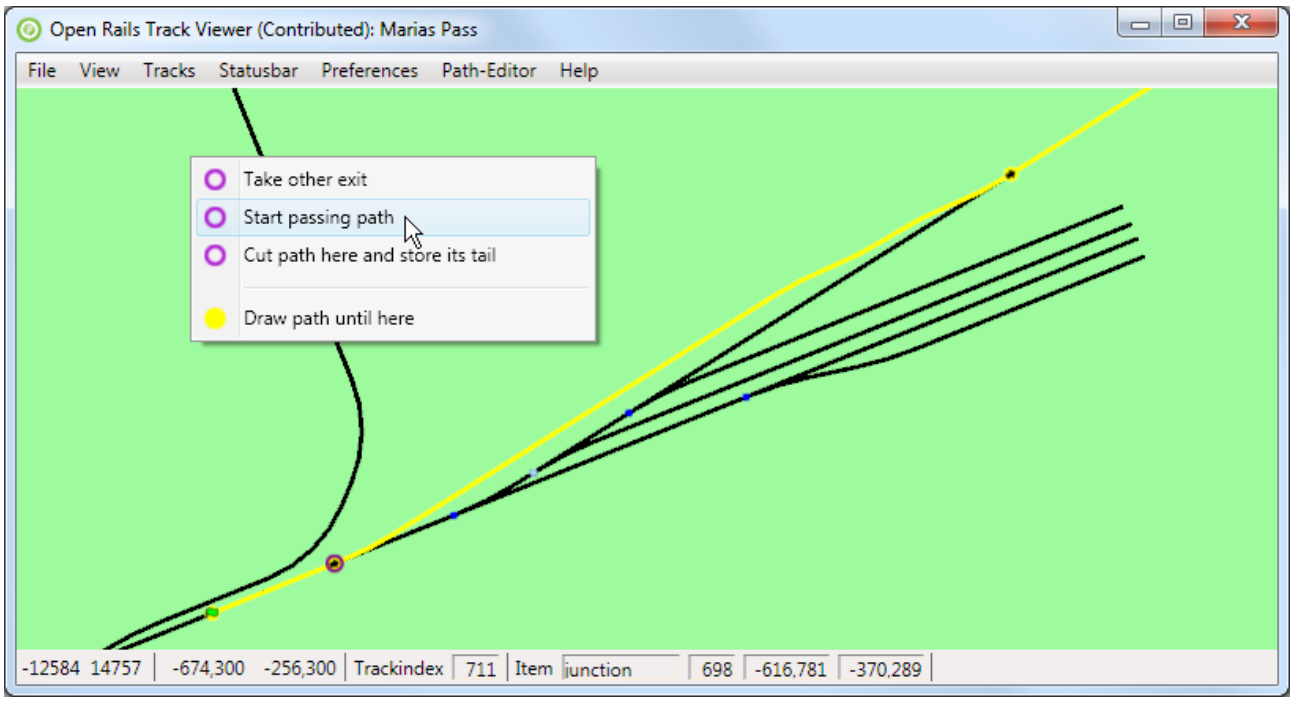


The result is shown below: the passing path has been added (in orange). In this case it is a very simple passing path without any nodes.

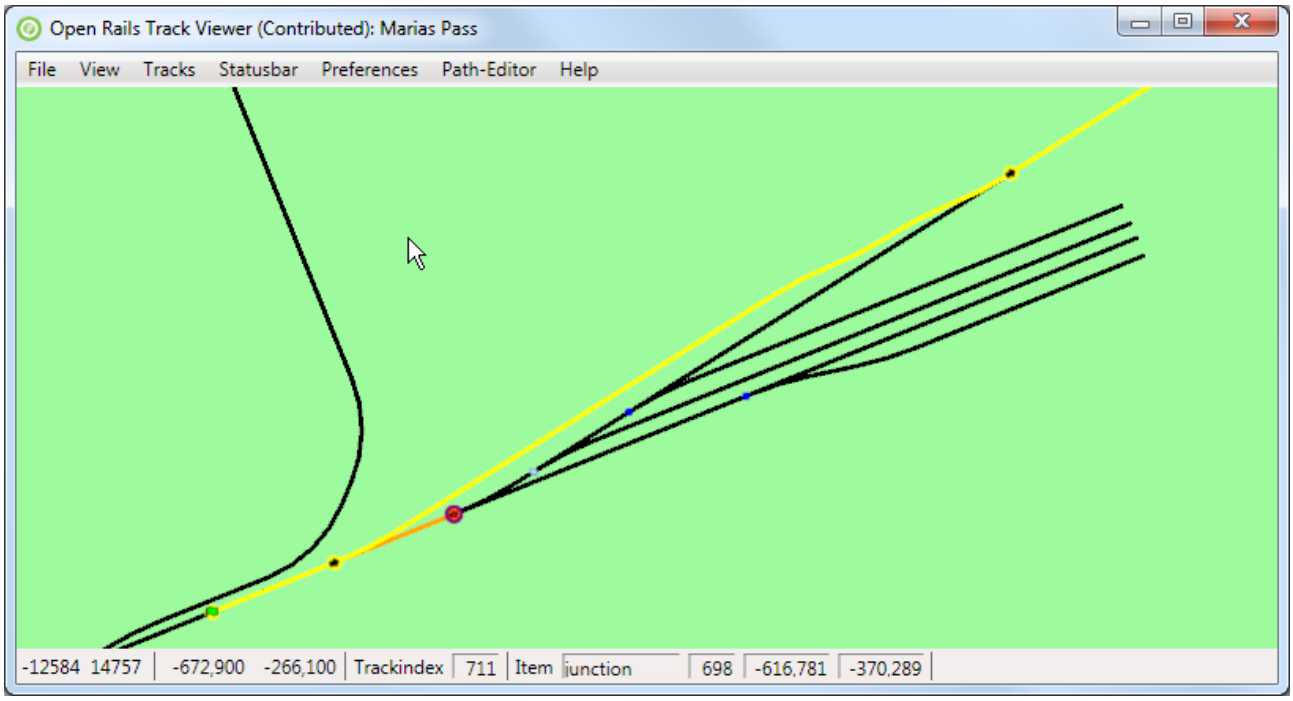


4.4.2. Complex passing paths

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:

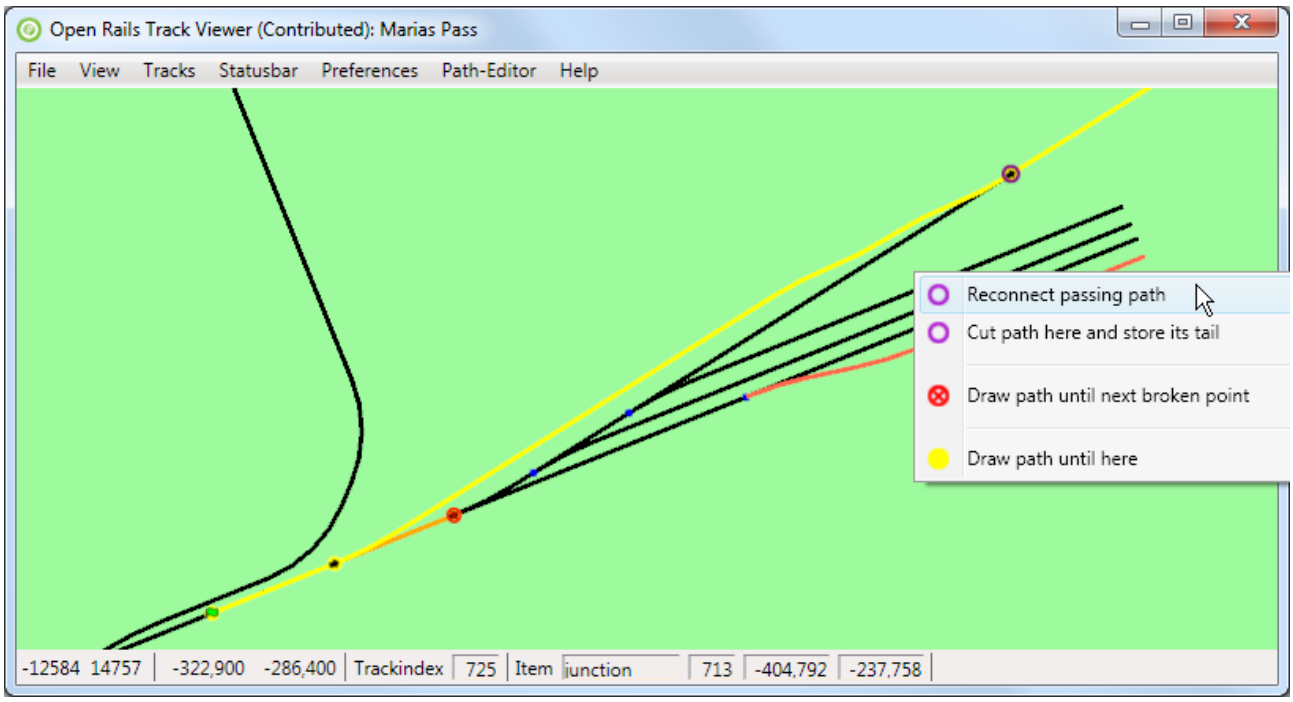


Starting the passing path at the same point, and following only main tracks at each junction would not lead to a reconnect-point, but only to one of the sidings. To create a passing path here the first step is to 'Start passing path':

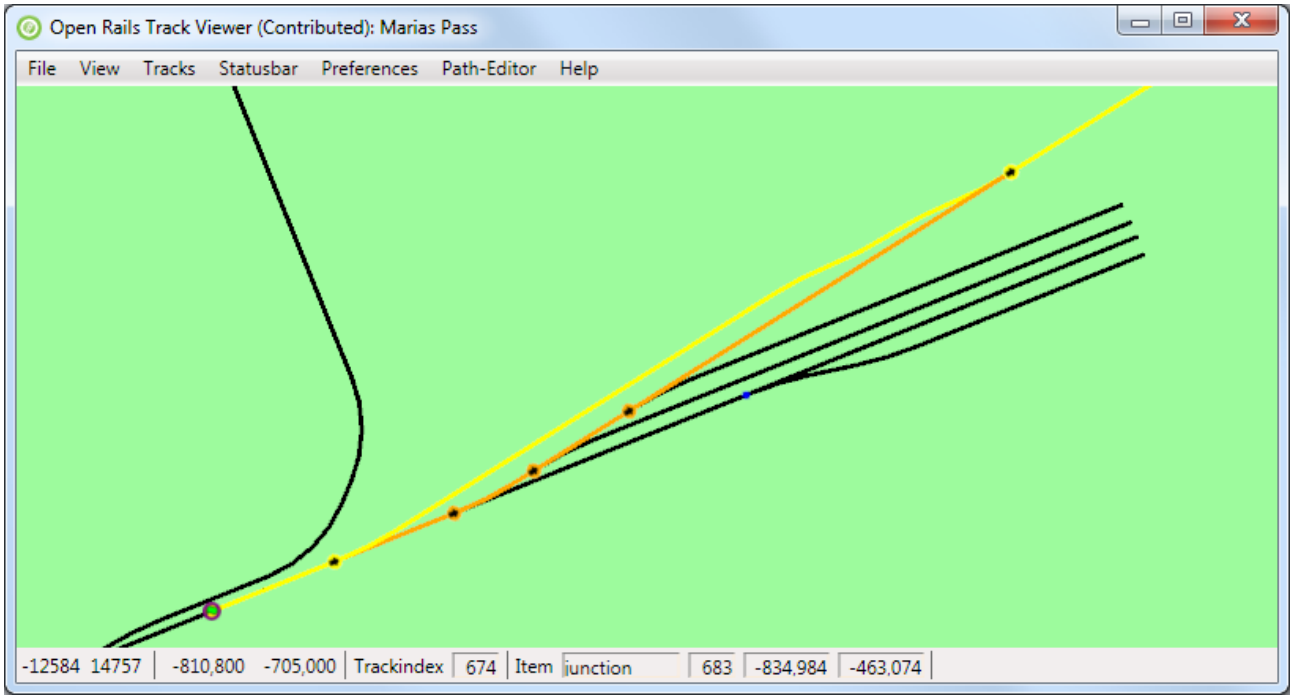


This will create only the first part of a passing path. The path is now broken (as can be seen from a cross through the just created node on the passing path.

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'. Track Viewer 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', Track Viewer 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.



4.5. 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.

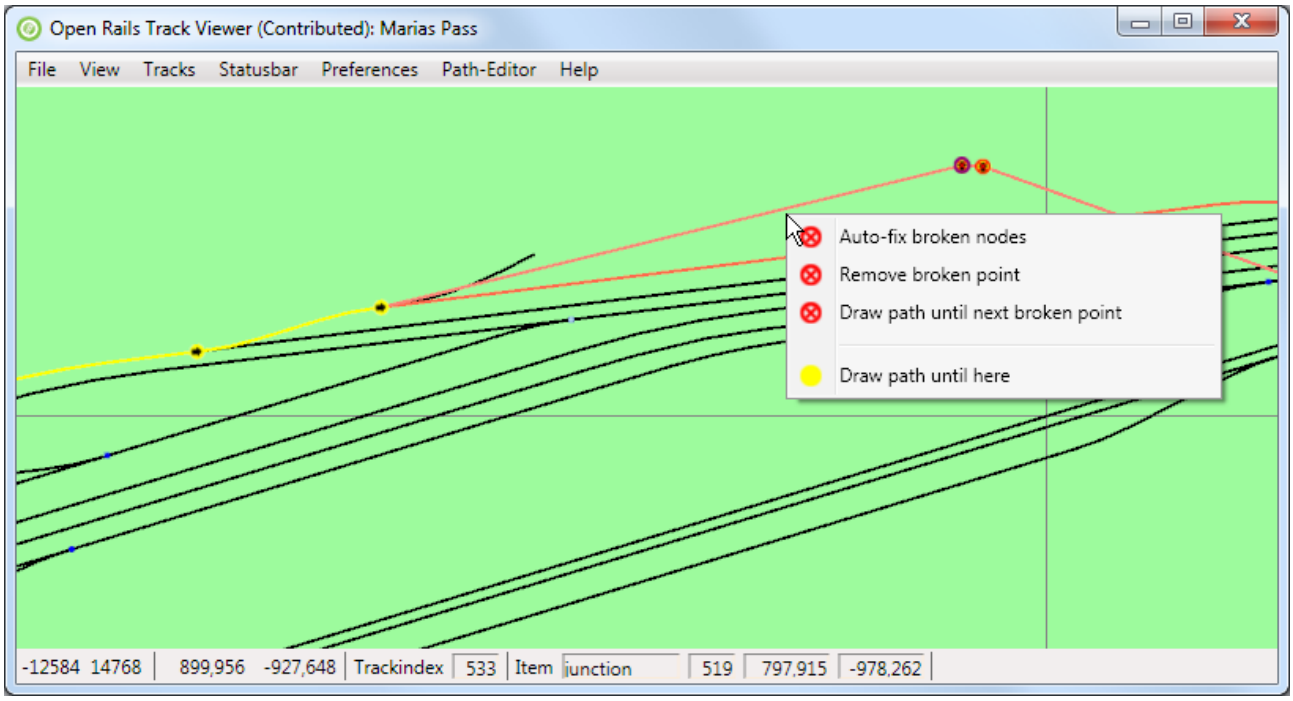
To correct broken paths, the following three options are available:

- 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, Track Viewer can auto-correct the broken path itself.
- For other complex situations, use the 'cut-and-store-tail' functionality.

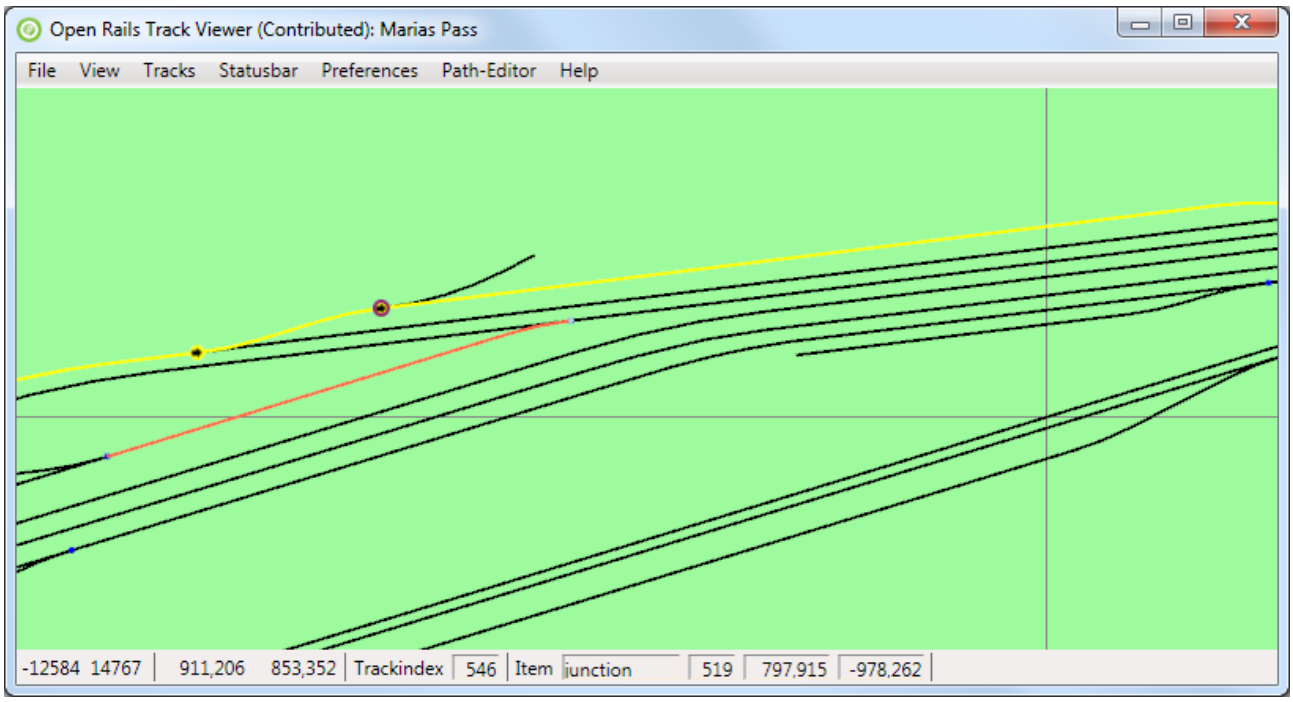
Both are described in more details below.

4.5.1. 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:

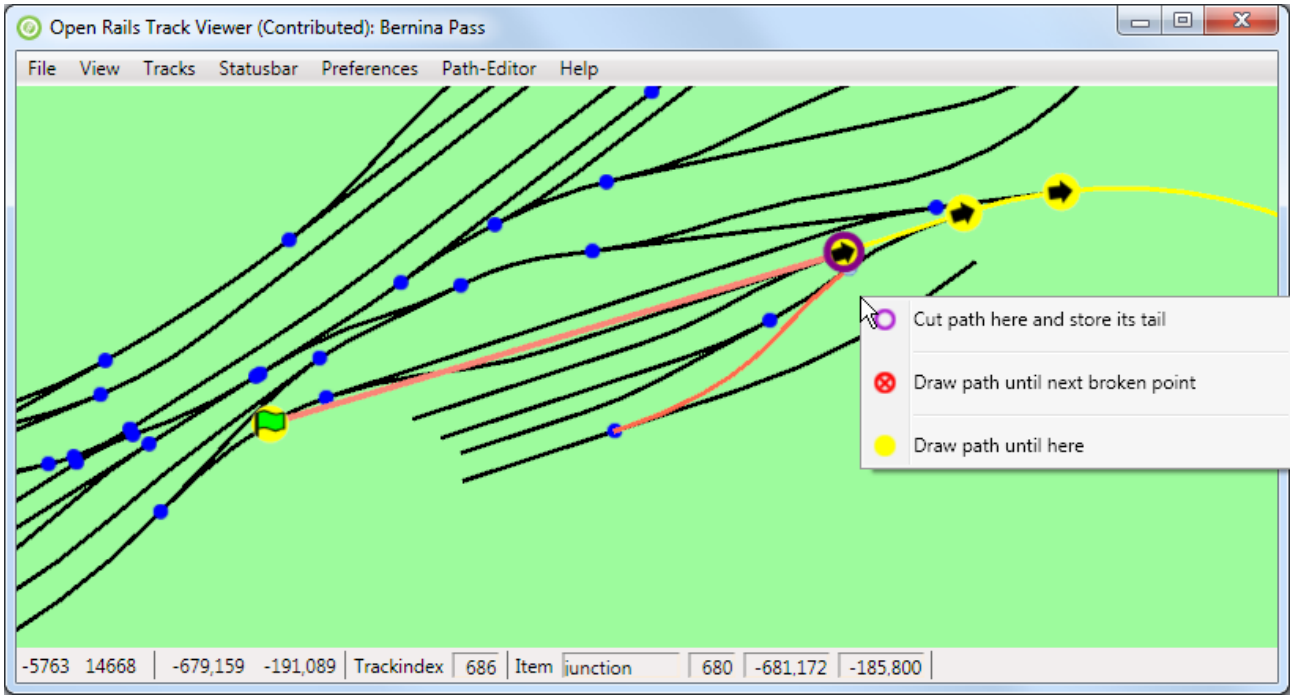


As one can see, the path has been fixed.

4.5.2. Fixing broken nodes in more complex situations

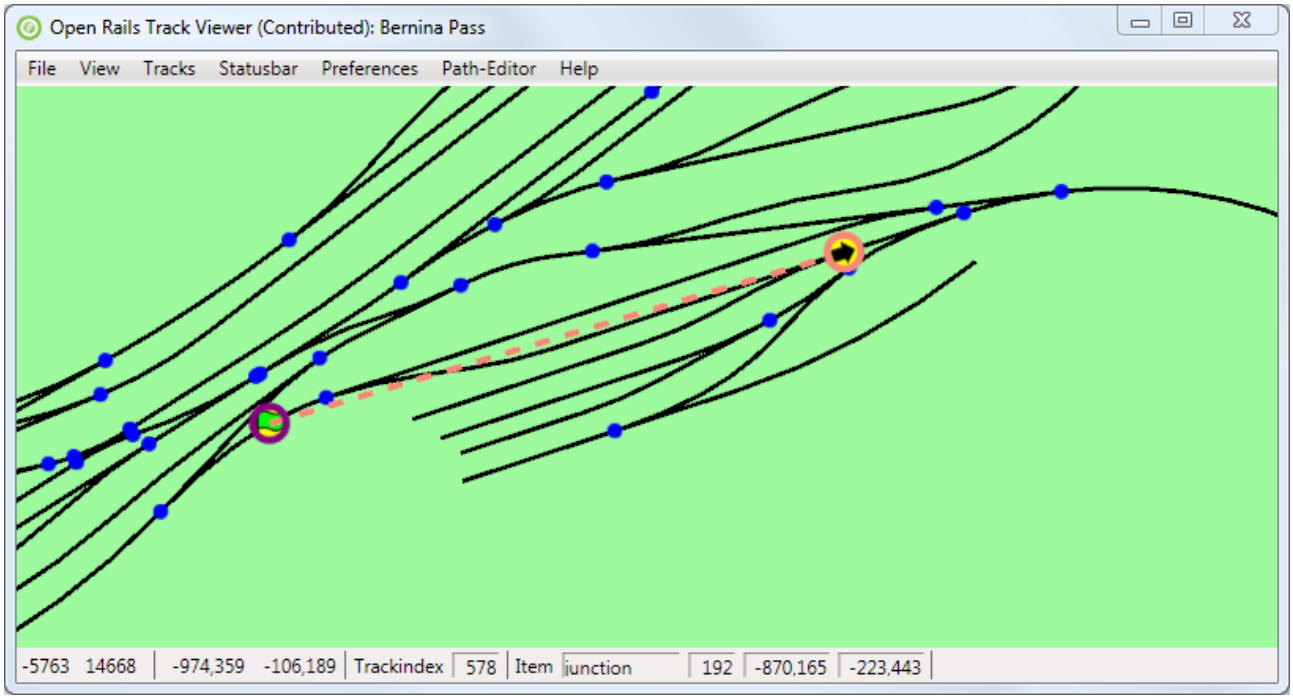
Sometimes it is not possible to auto-fix the broken nodes. Most likely this is due to a situation where either the node before or after the broken section is not directed correctly. The direction of the path on the track can not always be determined when the path is broken. Then there is a 50% chance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

As an example, see the broken path below:



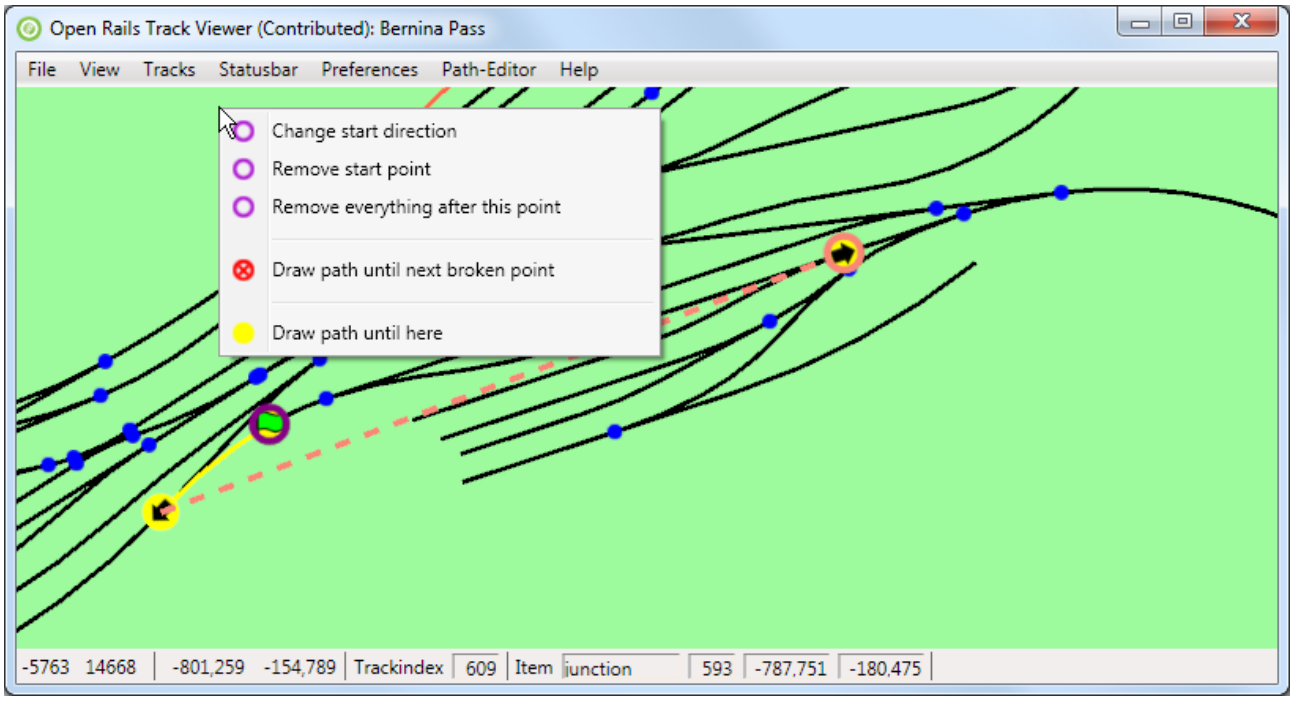
It is obvious that there is a simple connection between the start point and the first good junction

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.

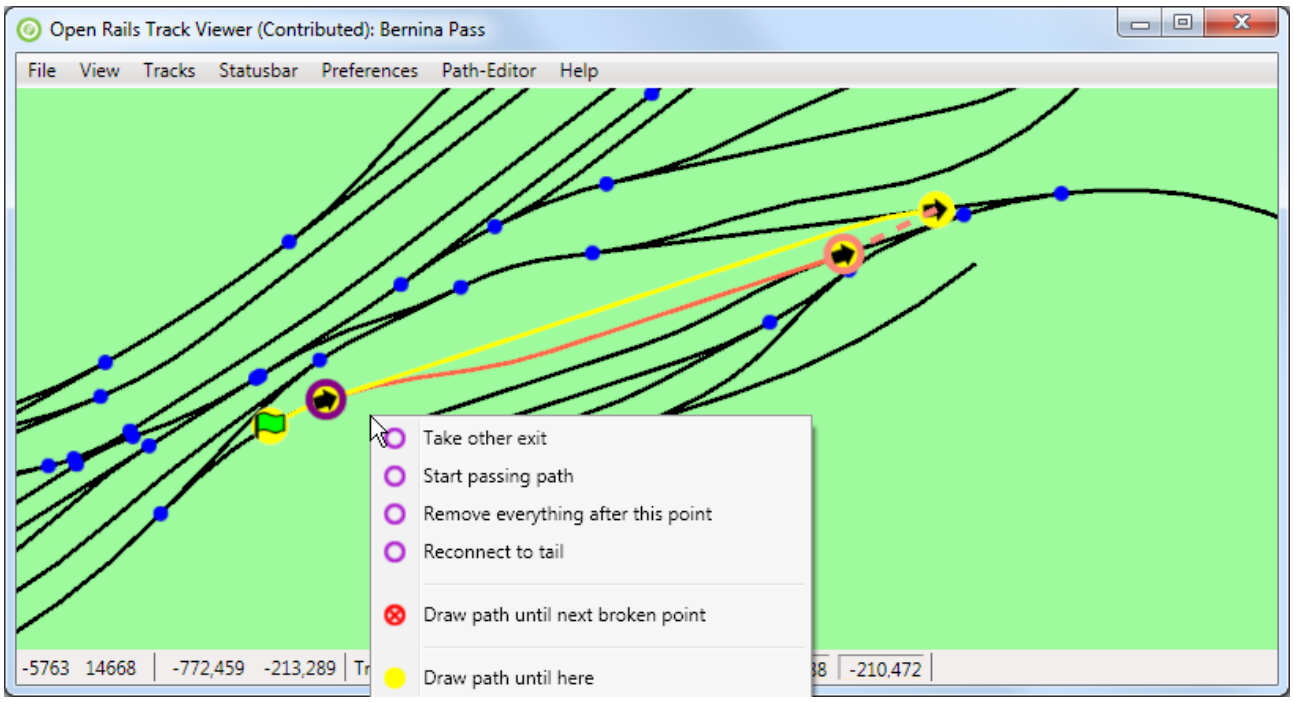


The tail is now drawn using only its first node. Furthermore, there is a dashed line from the last drawn node of the path to the tail. In this way you see where you need to connect again.

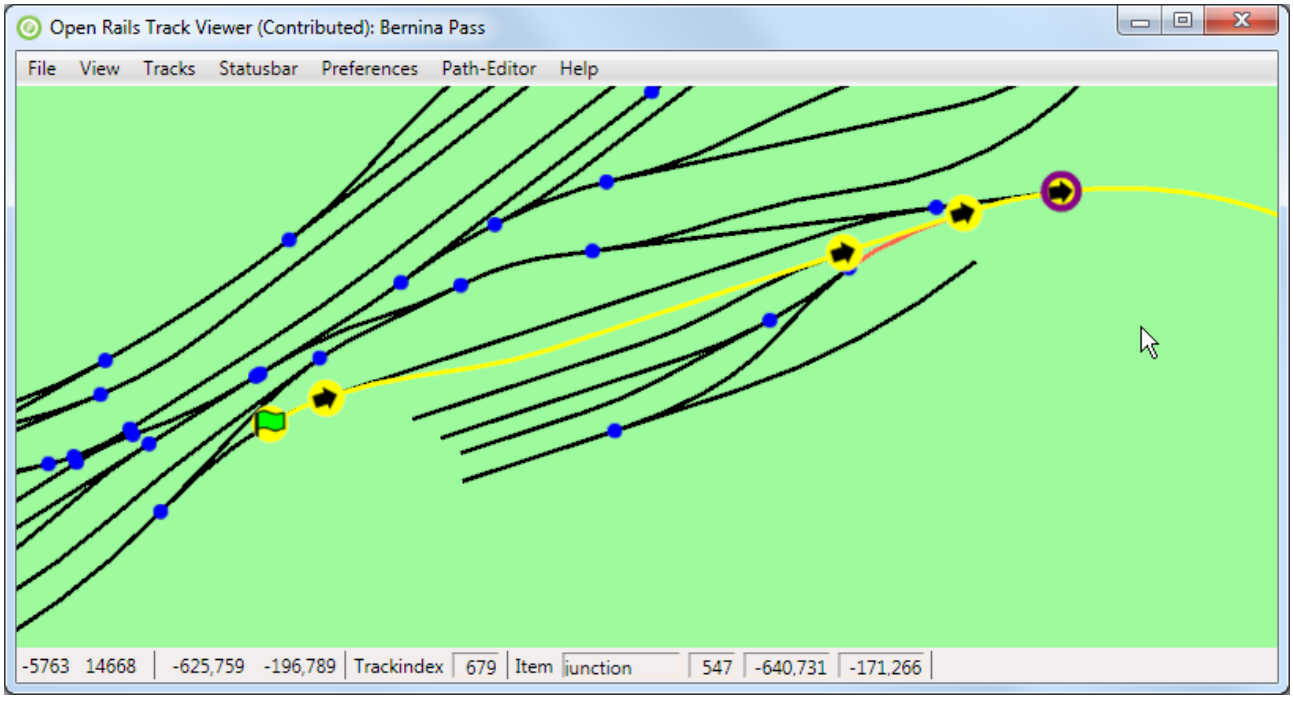
At this point only the start node of path is visible. Adding an extra node to the path (press Page-Up) makes the situation clear:



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.

4.6. 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.

Currently there has not been a large amount of testing the created or modified paths for use withing MSTS or ORTS. Any feedback is welcome.

5. 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.

5.1. Viewer

Zooming and moving the view window

- = Zoom-in. Keeping it pressed will keep on zooming in.
- Shift-= Zoom-in slowly, one small step at a time.
- − Zoom-out. Keeping it pressed will keep on zooming out.
- Shift-− Zoom-out slowly, one small step at a time.
- z Zoom-to-tile: zoom to a level where exactly one MSTS tile (2048m × 2048m is visible).
- r Zoom-reset: view the whole track.
- a Shift-left: Shift the view-window to the left.
- d Shift-left: Shift the view-window to the right.
- s Shift-left: Shift the view-window down.
- w Shift-left: Shift the view-window up.

Toggling what is visible/drawn

- F5 Show speed-limits.
- Shift-F5 Show mileposts.
- F7 Show signals.
- F8 Show platforms.
- shift-F8 Show platform-names.
- F9 Show sidings.
- Shift-F9 Show siding-names.
- 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.
- Scroll-wheel: Zoom-in or out.
- Shift-scroll-wheel: Zoom-in or out, but slower (to enable more precise control).

5.2. 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.
- 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.
- Shift-PgDn Draw only the start node of the path.
- Ctrl-z Undo the last edit.
- Ctrl-y Redo (only possible when at least one Undo has been done).

Mouse behavior

- Additional mouse button 1: Undo
- Additional mouse button 2: Redo

6. 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. Some of these might end up being implemented

- 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).
- 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.