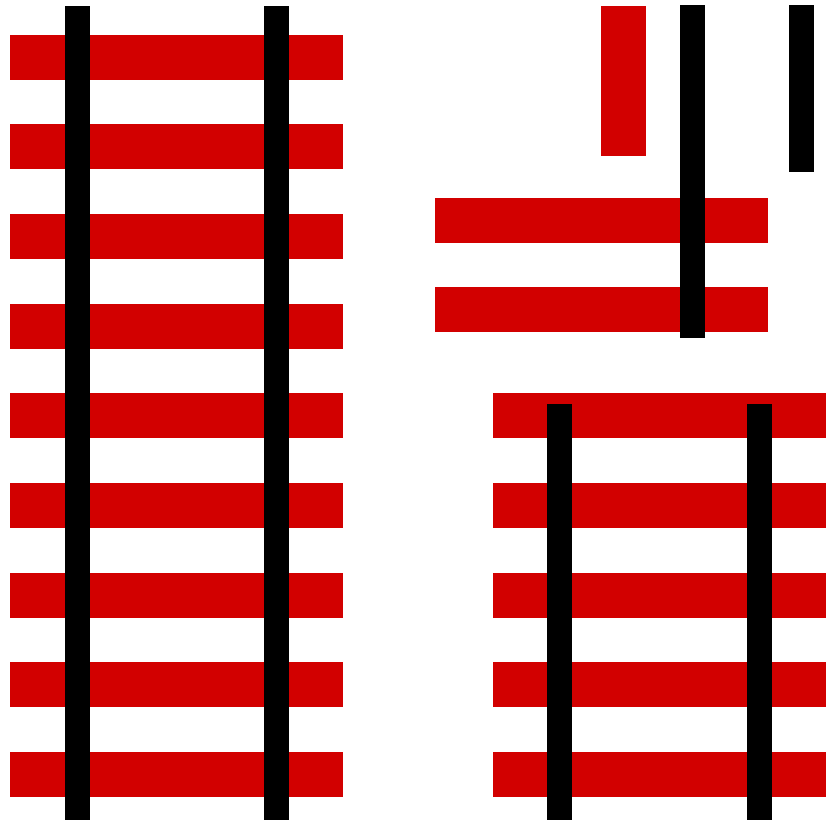




Raily 5 User Manual

10/17/2026

Contents

Welcome to Raily 5

Getting started

- Getting started 1: Build and look
- Getting started 2: Running trains
- Getting started 3: Annotations
- Getting started 4: Tunnels
- Getting started 5: Terrain
- Getting started 6: Leaving the flat
- Getting started 7: Decorating

Operation

- Conventions used in this manual
- Navigation in 2D and 3D
- Select objects
- Snapping
- Grid
- Rulers and Guidelines
- Work plane and coordinates
- Layers

Planning track

- Module selection
- Laying track faster
- Flex track
- Rail display
- Track labels
- Element info
- Embankments and ramps
- Viaduct
- Create helix
- Mirroring
- Track annotations in detail
- Create tunnel
- Signals, barriers and blocks

Trains, figures and cameras

- Train composition
- Placing trains
- Walking and cameras

Objects and shapes

- Move objects
- Rotate objects
- Scale elements
- Groups
- Drawing tools
- Polyline

- Image 2D
- Dimensions and measuring
- Baseboard
- Installing into multiple hosts
- Material selector
- Color selector
- Advanced tools

Scenery

- Terrain editing (Terrain & Sculptor)
- Import Heightmap
- Environment

Import, export and printing

- Importing files
- Printing
- Export SVG
- 3D export (STL, 3MF, glTF)
- Export DXF and HPGL
- Export to Raily 4
- Inventory

Setup and management

- Settings
- Plan Properties
- Backup and recovery
- Account and server
- License, free tier and subscription
- Platform differences (Web and desktop app)
- Desktop app lifecycle

Reference

- When something doesn't work
- Keyboard shortcuts
- Technical limits
- Performance tips
- Glossary

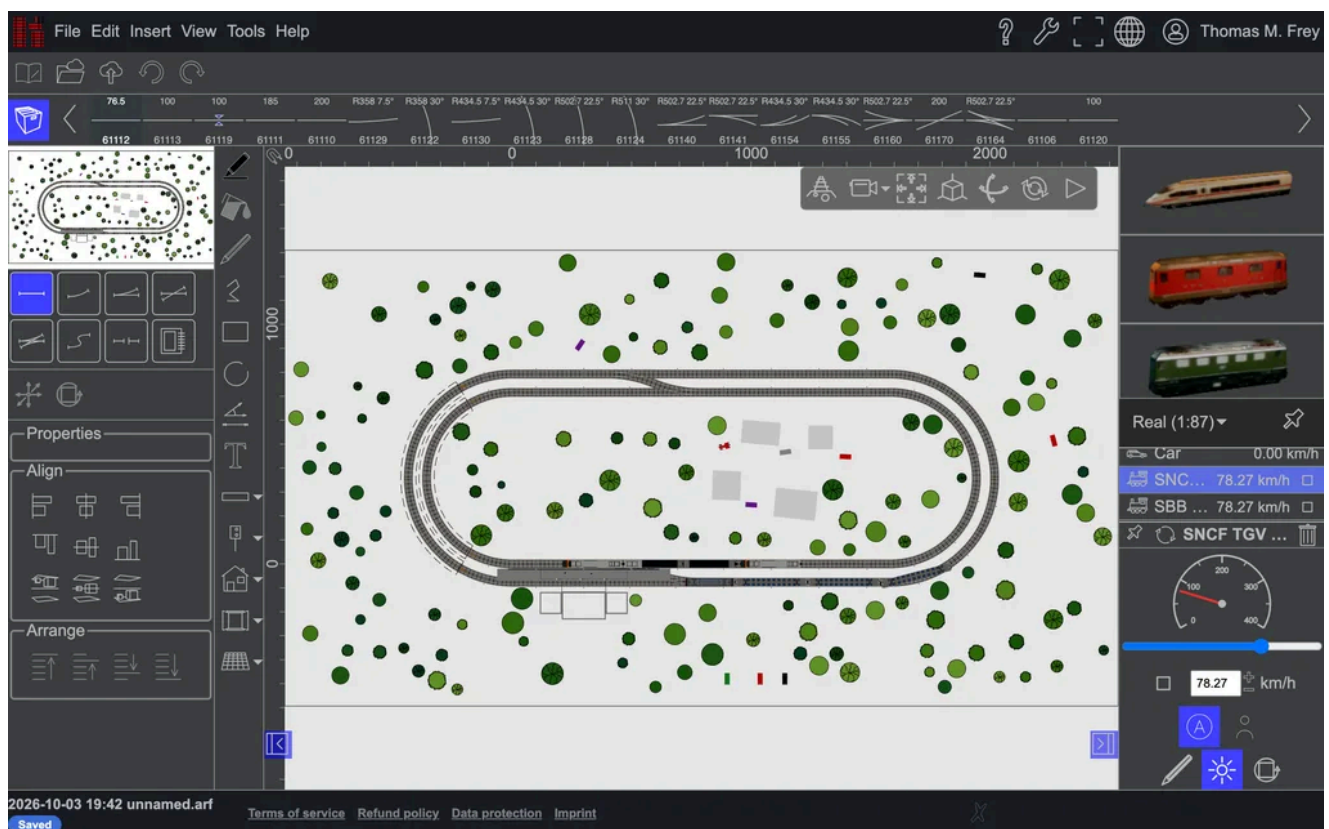
Welcome to Raily 5

You lay track, see what fits, **and know at the end which track material you need.**

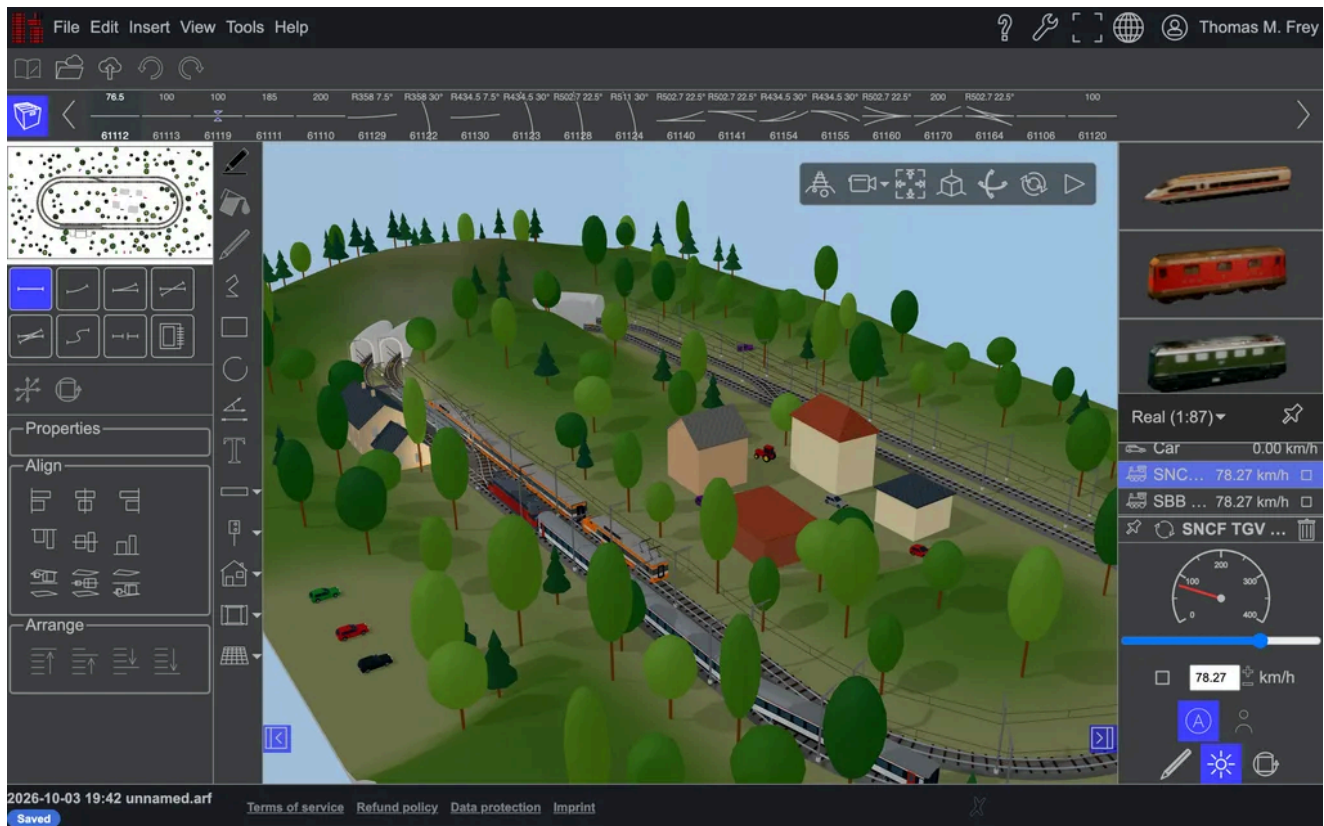
Raily began in 1991 as an "Electronic track template for Märklin plastic and metal track" — a program that replaced the plastic templates you'd otherwise lay out on the table. The original purpose has stayed the same; Raily has, however, learned a few things since. Today it supports track from more than seventy manufacturers. The layout can be shown in three dimensions, trains run on it, and terrain, tunnels and bridges appear right along with it.

What you can use it for

- **You have track and want to see what you can make of it.** Tick off your track system and get going. The [Inventory](#) tells you at the end what you're still missing.
- **You have an idea and want to see whether it works.** Lay track, try out different variants, and see right away what fits.
- **You don't have room for a real layout.** So build it virtually instead, with terrain, light, time of day and trains doing their rounds.



Planning happens flat, from above: this is where the dimensions are accurate, where you measure and print.



And the same plan in three dimensions, with terrain, buildings and trains running.

Here's where to start

[Getting started](#) leads you through seven steps, from the first rail to a fully built-out layout. Set aside the first hour for it; you'll find everything else on your own after that.

If you're looking for something specific

Chapter	Contents
Getting started	from the first rail to the second level, step by step
Basics	moving around, selecting, hitting a point precisely, layers
Planning track	modules, flex track, ramps, annotations, tunnels, signals
Trains, walkers and cameras	assembling a train, placing it on the track, riding along
Objects and shapes	everything that isn't track: moving, drawing, building solids
Terrain	terrain, sky and light
Import, export and printing	what goes into the plan, and what comes out of it
Setup and management	settings, backup, account, license
Reference	troubleshooting, keyboard shortcuts, limits, glossary

Getting started

Getting started 1: Build and look

The built-in tutorial starts by itself the first time you open Raily. It walks you through the same steps as this page; the best way is to follow both in parallel.

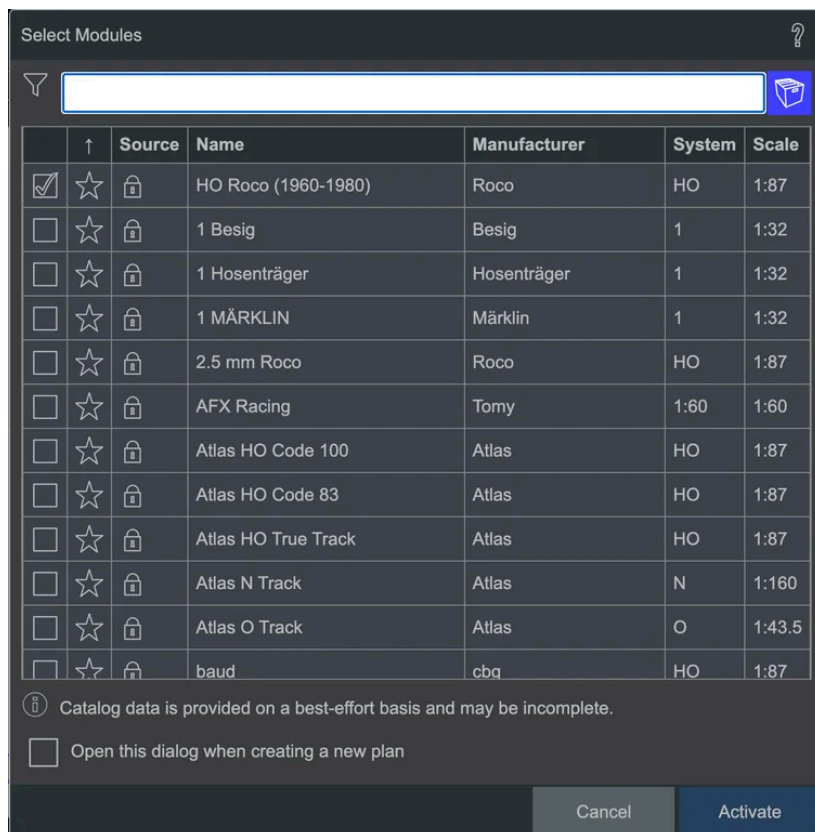
Clicked the tutorial away?

Help → **Reset tutorials** brings them all back. Then restart the program, or reload it in the browser.

File → **Example plans...** has, in the **Tutorial** folder, the plans the following steps use.

Choosing modules

Raily knows the track of many manufacturers. To keep the choice manageable, you work with the **modules** you have ticked; only their track is available.



The module selection dialog.

You can change the choice at any time, with **m** or through **View** → **Module selection...**

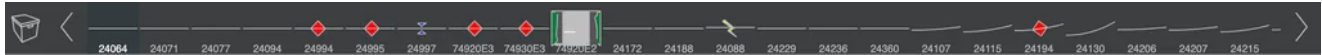
- The **star** keeps modules you use often at the top.
- The **archive box** also shows modules that are no longer manufactured.

Track and plan at the same scale

If you reach for track of a different scale than the plan, Raily asks: **Change** adopts the track's scale for the plan, **Keep** leaves it as is.

Placing track

The **quick-pick bar** holds the track of the active modules.

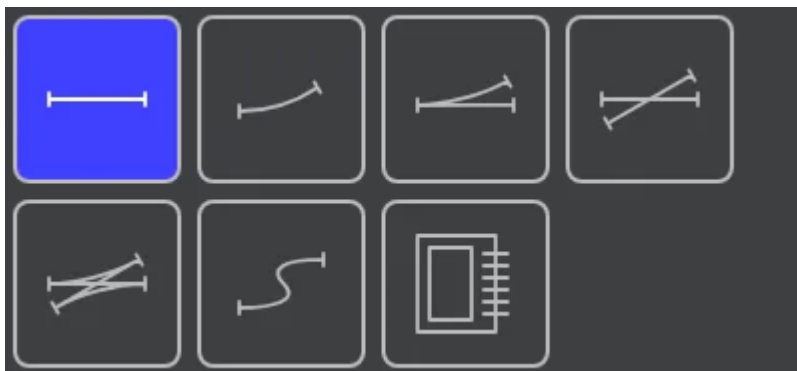


The quick-pick bar with the track of the active modules.

Drag a piece of track from the bar into the plan. Each further piece snaps onto a free end of the track once you drag it close enough.

While the track is still hanging on the pointer, **S** picks **which end** docks. On a curve, that decides whether the curve continues to the left or to the right.

The track buttons in the left toolbar jump straight to a kind of track in the bar: **Straight track**, **Curved track**, **Switch**, **Crossing**, **Crossing switch**, **Flex track**, **Special track**.



The seven buttons for the kinds of track.

There is a faster way for whole stretches of track

With **A** and **Enter**, without moving the mouse, see [Laying track faster](#).

Build a circle

Now it is your turn: lay a closed circle out of curved track. You will need it in the next step.

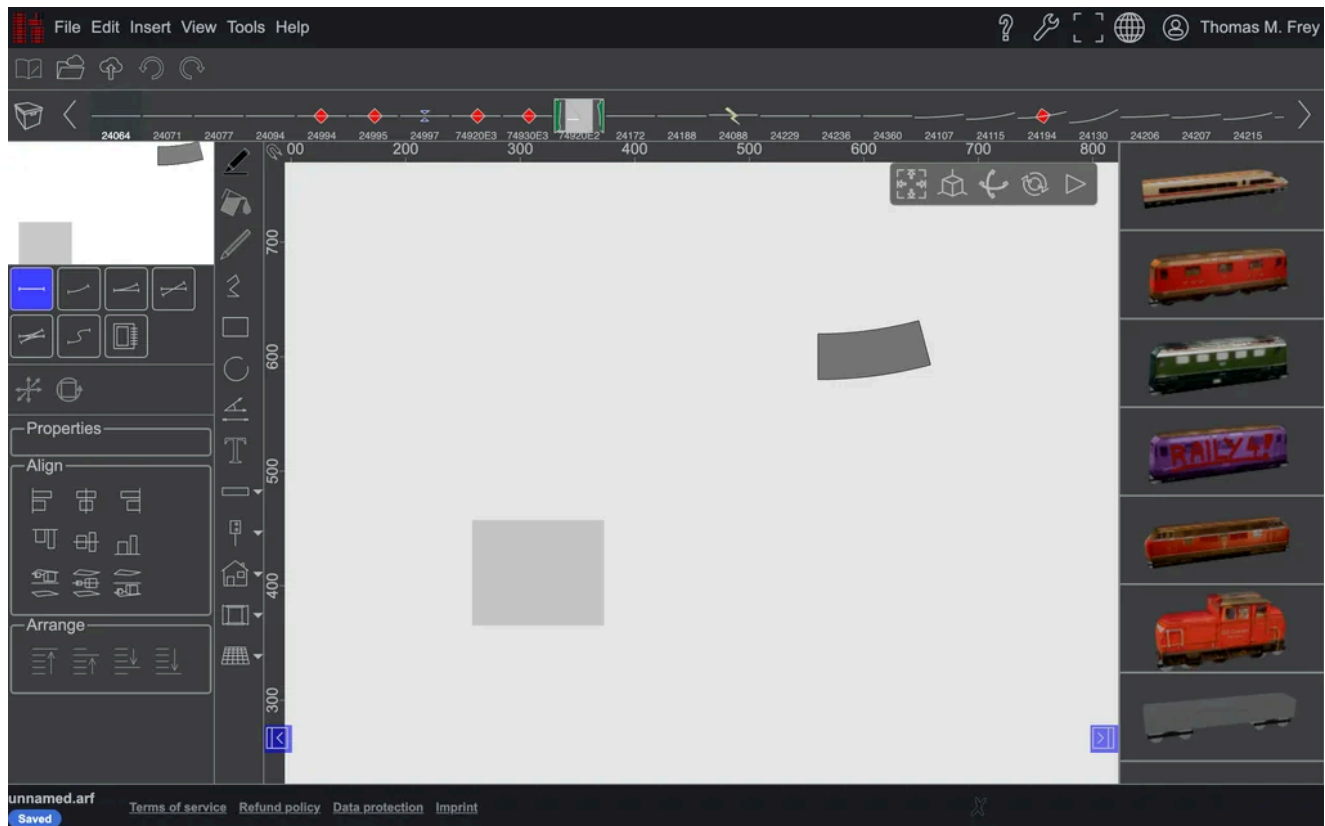
Bringing the whole plan into view

2D view switches to the top-down view and brings the whole plan into view. That is the button that brings you back when you have zoomed yourself away.

With the **mouse wheel** you zoom, with the **middle mouse button** you pan; on a trackpad or tablet it is two fingers. The **plan overview** at the top left shows the plan in miniature; a click inside it takes you to that spot. More under [Navigation](#).

Placing a building

Put a house beside the track with  **Add Building**. The button sits in the left toolbar in the **Buildings** multi-tool button; the small triangle next to it opens its list, see [Conventions](#).

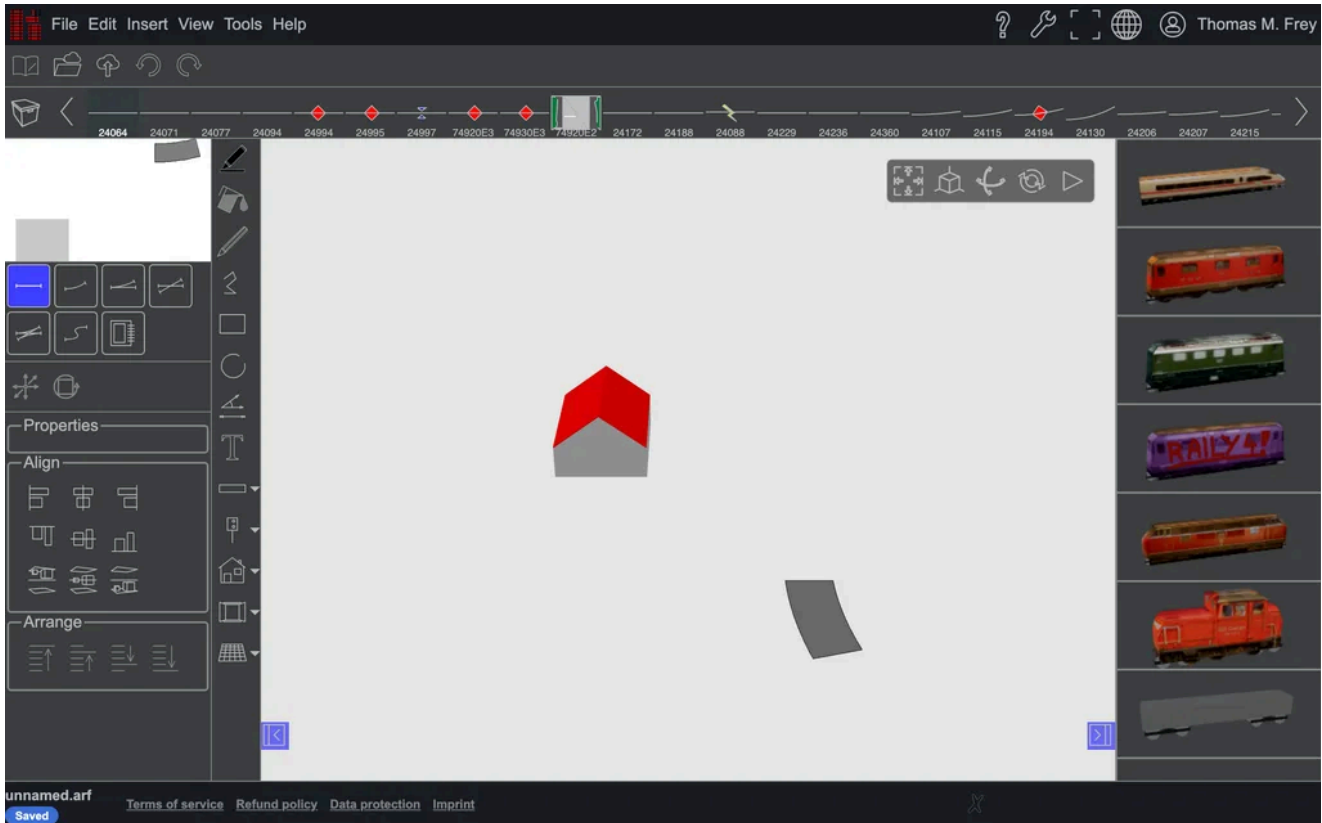


Track and building from above.

From above that is a fairly dull rectangle... time for the third dimension.

From above to 3D and back

Button	View
 2D view	flat from above, for building and measuring
 3D view	in space, the view stays aligned with the layout
 3D view (trackball)	the same, but freely turnable around the plan



The same building in the 3D view.

The rectangle has become a building.

- **Turn:** press the mouse button and move. On a selected object, dragging would move the object instead; **Shift** prevents that.
- **Center:** **Alt**-click an object to make that point the new center of rotation.

Selecting

- **Click** selects an object, **Ctrl**-click adds or removes one at a time.
- **Dragging a selection rectangle** selects everything inside it, 2D only.
- **Double-click** selects everything connected to the track, the whole circle for example.

How to efficiently select the stretch between two pieces of track is under [Selecting objects](#).

Saving

Ctrl + **S** saves. In the desktop app that is a file on your computer. In the browser, **File** → **Save As...** saves to the railly.ch server, for which you need to be [logged in](#), and **File** → **Save to Computer...** saves locally.

In between, Railly backs up in the background, see [Backup and recovery](#).

Next

- [Getting started 2: Running trains](#)

See also

- [Conventions used in this manual](#): keys, the right mouse button, and the second click that opens an editor
- [Snapping](#): why one piece of track locks onto another, and when it doesn't
- [Module selection](#)

Getting started 2: Running trains

Time for a bit of fun as a change of pace... Take the circle from [step 1](#), or load a ready-made one: **File** → **Example Plans...**, folder **Tutorial**, plan `tutorial2`.

Putting a train on the track

The vehicles are on the right, in the **object panel**. Drag one of them onto the circle. While you guide it over the track, Raily shows a **preview of the whole train** in the position it would take. That way you can see, before you let go, whether it's right.

The direction of travel follows the direction in which you move the train.

Running

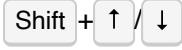
▷ **Start animation** sets everything moving; the same button stops it again. The train keeps going round for as long as the animation runs.

Switch the animation off while planning

You can plan with the animation running, but it's usually easier to switch it off. Some elements behave differently in planning mode and in "play" mode.

Riding along

 **Cockpit view** puts you in the train. If you have several trains, you can pick the "active" one from the list at the bottom right.

Key	Effect
	faster
	slower, then backwards
	much faster / much slower

The same keys also control a walker or a camera, see [Walking and cameras](#).

Throwing switches

Load `tutorial2b` from the same folder. The circle now has a switch, and with it a decision: where does the train go?

Drive up to the switch in the cockpit and press . That throws the **next switch ahead**.

Next

- [Getting started 3: Annotations](#)

See also

- [Placing trains](#)
- [Train composition](#)
- [Signals, barriers and blocks](#), so a train stops on its own

Getting started 3: Annotations

Annotations mark **stretches of track**. A single "marker" looks like an arrow and means "the annotation starts here at the arrow head and carries on from there". Raily works out how far the mark reaches. That calculation is the whole point: you say where something begins, not which twelve pieces of track belong to it. That same mark becomes a tunnel in step 4, and a cut in step 5.

Load **File** → **Example Plans...**, folder **Tutorial**, plan `tutorial3`. It holds three exercises: three straights at the top, a straight with a switch in the middle, and a circle at the bottom.

Exercise 1: the straights

The first annotation

Put a marker on one of the **upper straights** with  **Add highlight annotation**.

The marker has to sit on the track

An annotation lying beside the track is not anchored and is passed over when you refresh. Raily then reports *"The annotation is not anchored to a track — drag it onto one first."* Drag it onto the track again in that case.

Now **right-click** the marker and choose **Refresh annotation**.

On a tablet

Without a right mouse button, **long-press** the marker instead. The same applies everywhere the manual mentions the right mouse button, see [Conventions](#).

The result: the mark runs from the arrow **to the end of the track**. You have set a beginning but no end, so Raily takes everything that follows in the direction of the arrow.

Setting a counterpart

Put a second annotation on the same straight, further to the right, and have it **point back at the first**.

The arrow direction comes from the side of the track

Which **side of the track** you point at with mouse, finger or pen decides where the marker's arrow points. Drag it to the other side and the arrow turns round. **Flip direction** in the context menu does the same.

Refresh the mark. It now lies **exactly between the two arrows** — the second marker bounds what the first one opened.

Exercise 2: the switch

Move to the middle group: a straight, a switch, and two straights behind it.

Put an annotation on the straight at the left, with the arrow pointing **right, towards the switch**, and refresh.

The result: **every** element is marked — including the branches behind the switch. The region follows every path reachable from its beginning, and a switch simply has more than one continuation.

Now put a counterpart on each branch, pointing back, and refresh again. The mark is cleanly bounded.

Exercise 3: the circle

Put **a single** annotation on the circle at the bottom and refresh.

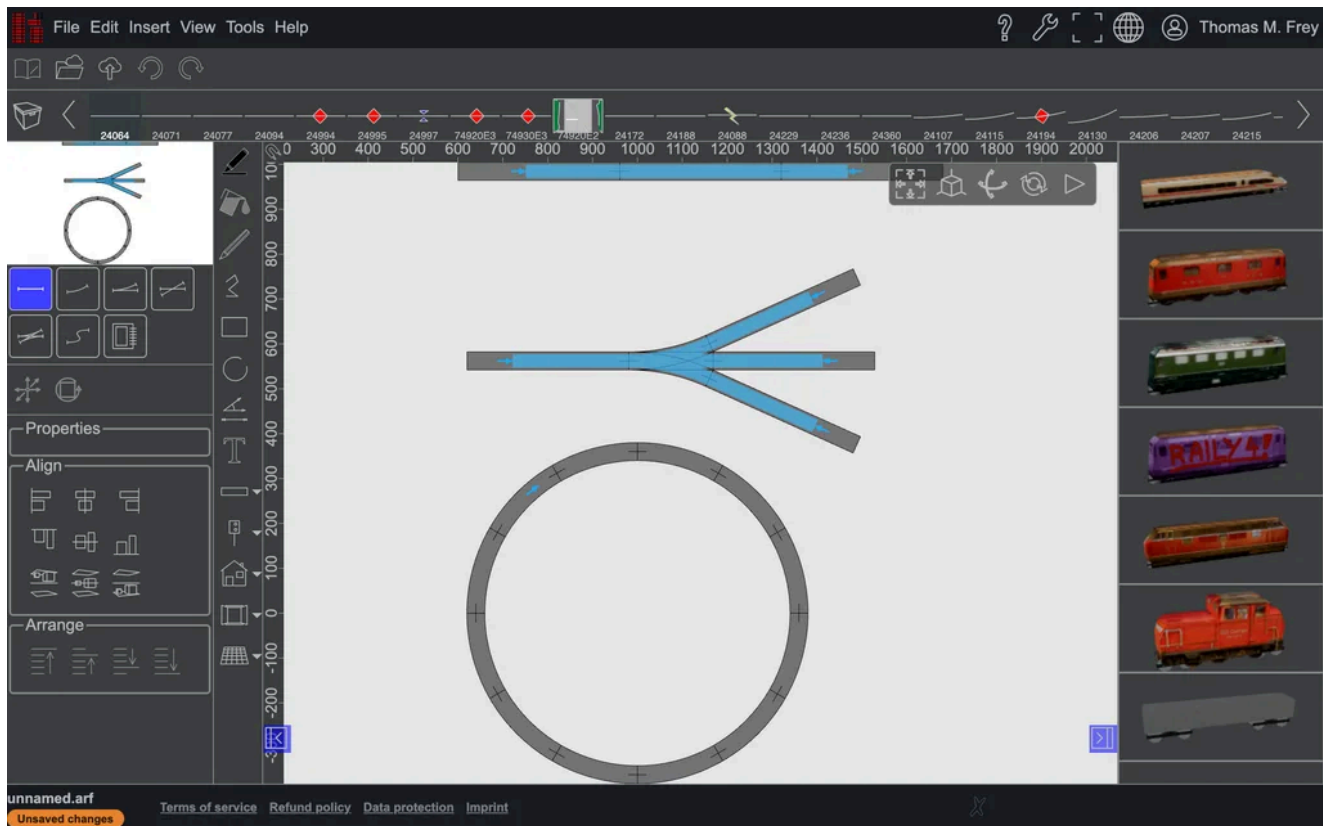
Nothing happens — instead a red message appears briefly: *"Annotation is inconsistent: the region loops back into an entrance. It is not applied."*

That is not a fault but the nature of a circle: the region runs round and meets itself from behind. Raily cannot know where it should stop, so it does not apply it at all.

On a closed loop you always need two annotations — one for the beginning, one for the end. Set a counterpart and refresh: the mark now sits on the stretch between them.

For comparison

The plan `tutorial3b` in the same folder holds the markers of all three exercises. After a **Refresh annotation** it looks like this:



The three exercises after refreshing: the pair at the top, the bounded switch in the middle, the loop at the bottom — with no mark.

At the top the mark lies between the two arrows. In the middle the switch and its branches are marked and bounded at all three exits — which takes **four** markers there: one as a beginning and three as counterparts. At the loop below there is deliberately only **one** marker, and that is exactly why the loop stays grey: the inconsistent case from exercise 3.

Next

- [Getting started 4: Tunnels](#)

See also

- [Track annotations in detail](#) — refreshing several annotations at once, naming annotations
- [Signals, barriers and blocks](#): the same markers, but as a block or a speed limit

Getting started 4: Tunnels

There are two ways to a tunnel. The first is quick, the second is the one you will use later.

Load **File** → **Example Plans...**, folder **Tutorial**, plan `tutorial4` — the same three exercise groups as in the last step.

The direct way: from the selection

Select the three upper straights (with a double-click, for instance) and choose **Create tunnel...** with the right mouse button. The dialog asks for the cross section and the dimensions, and Raily pulls the profile along the track.

The catch: a tunnel made this way always runs **exactly from element boundary to element boundary**. It begins and ends where a piece of track begins and ends — not where you want the ground in front of the portal. And if you later swap or move a piece of track, the tunnel knows nothing about it.

The recommended way: tunnel annotations

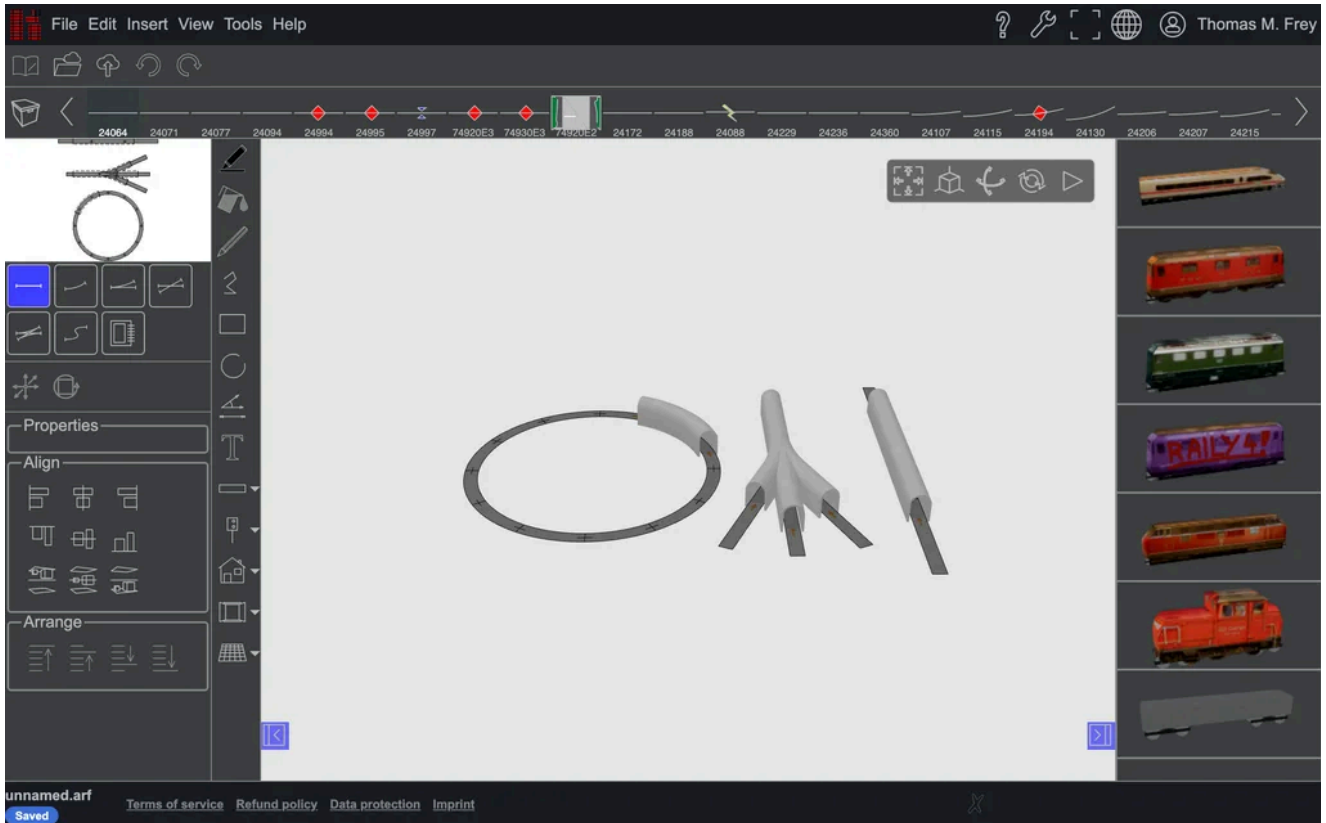
A tunnel is a stretch of track in just the same way as the mark from [step 3](#) — and is set in just the same way.

In the annotations' multi-tool button, choose  **Add tunnel annotation**.

From here the rules you already know apply:

- The marker has to sit **on the track**.
- A single marker reaches **to the end of the track**; a counterpart bounds it.
- On a **circle** you always need **two**.
- **Right mouse button** → **Refresh annotation** recomputes the region and builds the tunnel.

Set a pair on the upper straights, refresh, and switch to  **3D view**: the tunnel is there, with portals exactly where your markers are.



The same three exercises become three tunnels, including over the switch, where one region turns into a portal with three exits.

The advantage over the direct way: the portals sit freely on the track rather than at element boundaries, and when the track changes, refreshing again brings the tunnel with it. A further advantage is that going through an annotation can also "build" the tunnel into a terrain automatically.

For comparison

tutorial4b holds the tunnel annotations as set, tutorial4c the tunnels built from them.

Next

- [Getting started 5: Terrain](#)

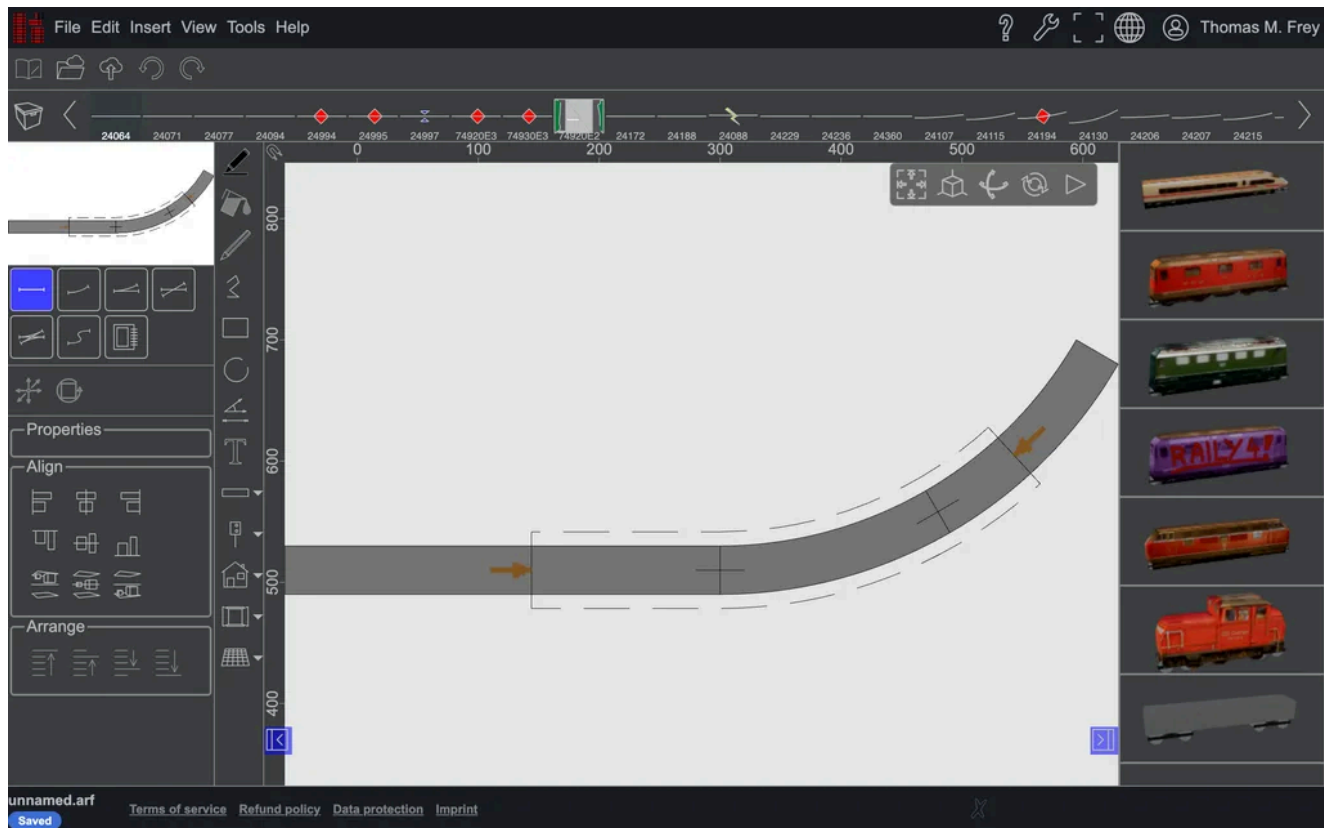
See also

- [Create tunnel](#) — cross sections, dimensions, profile offset
- [Track annotations in detail](#)
- [Editing terrain](#)

Getting started 5: Terrain

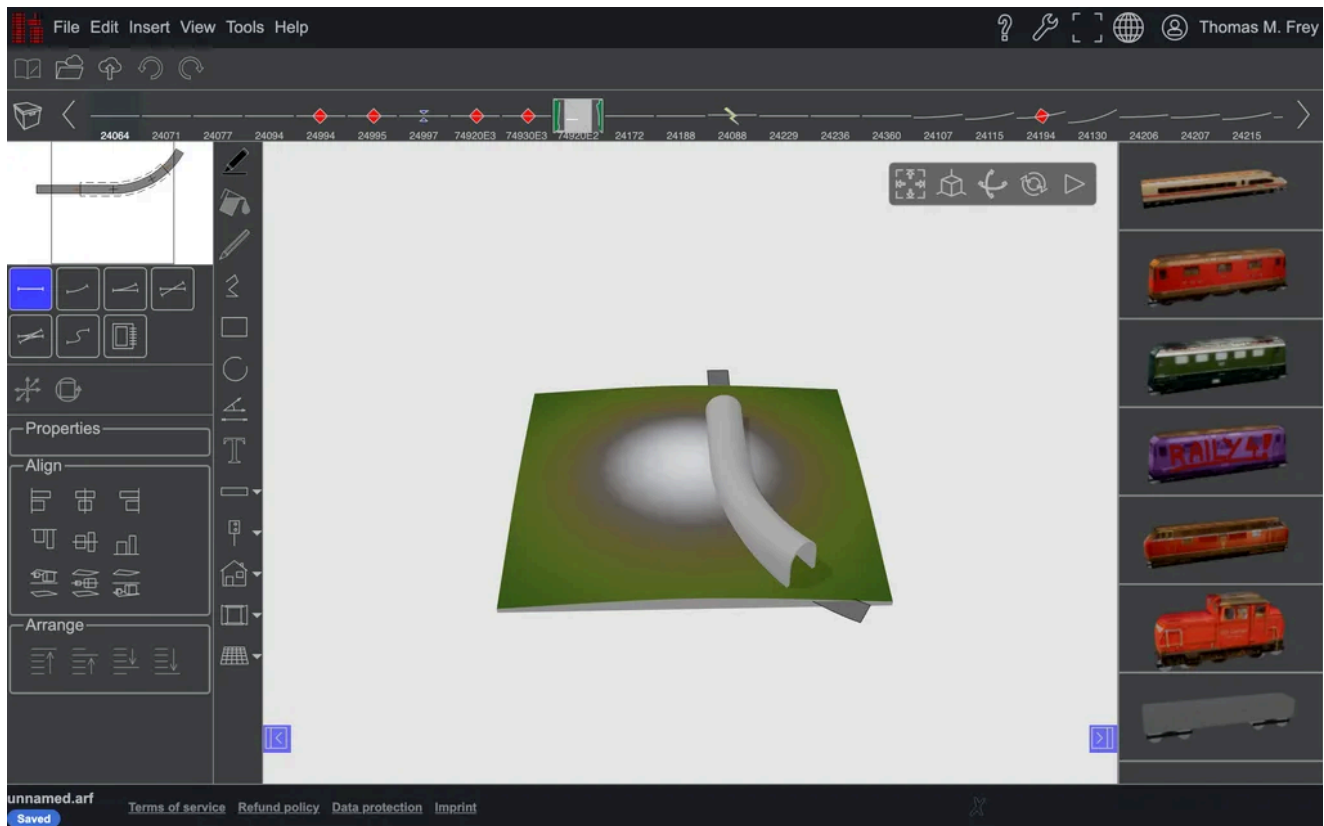
The tunnel from [step 4](#) still stands rather lost in space. In this step it gets a mountain around it.

Load **File** → **Example plans...**, folder **Tutorial**, plan `tutorial5a`: one track, one tunnel, and the two tunnel annotations from the last step.



`tutorial5a`: the track with the tunnel and the two markers that produced it.

With **Add Terrain** you put a block of terrain into the plan. At first it's a slab with a small rise; `tutorial5b` shows this state.



tutorial5b : the slab is there, the tunnel still stands free on top of it.

Select the terrain block and click it **again** to enter the terrain editor. This is how you reach the editor of any object, everywhere — see [Conventions](#).

The terrain sculpting tools

The **Terrain sculpting tools** panel holds four brushes at the top. They differ only in *what* the brush does to the heights beneath it:

	Tool	Effect
	Raise terrain	raises, most strongly at the center of the brush
	Lower terrain	lowers
	Flatten to clicked height	pulls everything to the height you clicked first
	Smooth terrain	averages out differences and removes edges and spikes

Below that sit **Brush radius** and **Strength**.

Gently, and several times

A small **Strength** value and several strokes give a more even landscape than one heavy stroke. With  **Smooth** you can fix afterwards whatever came out too jagged.

Now it's your turn: drag a mountain up over the tunnel, until the tunnel disappears inside it.

Bringing the tunnel into the mountain

The mountain stands, but the tunnel doesn't know anything about it yet. It was built when the terrain didn't exist yet.

Finding markers under the mountain

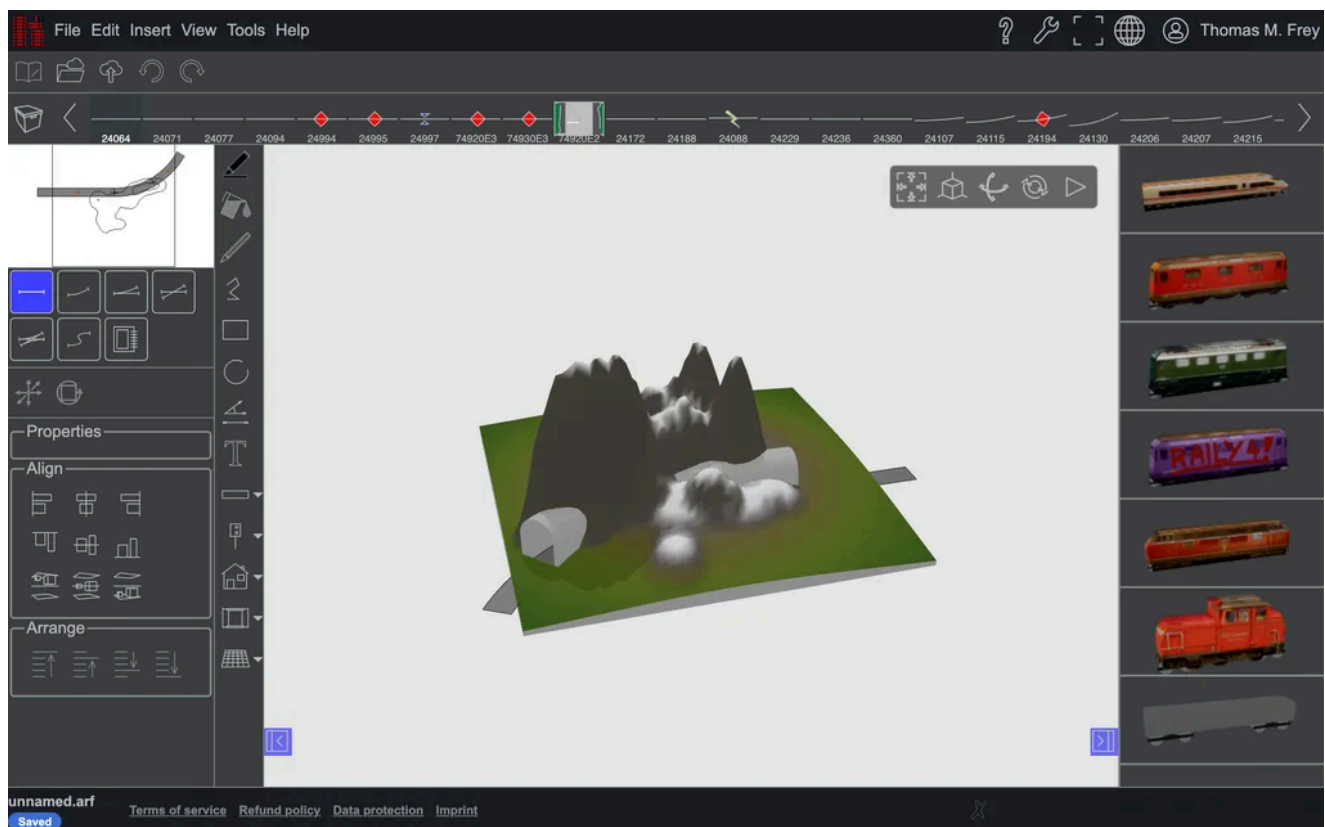
And here you run into a practical problem: the two tunnel markers now lie **inside the mountain**. In the 3D view they can no longer be clicked.

Annotations are found in 2D

Switch to  **2D view**. From above, nothing lies over the markers — they're just as easy to grab there as before the terrain existed. That's the normal case as soon as a plan has landscape: **you build in 3D, you edit annotations in 2D**.

Refreshing them all at once

But you don't have to hunt for the markers one by one at all. **Tools → Track annotations...** lists every annotation in the plan. Whatever no longer matches the layout shows **stale**, and **Refresh** rebuilds it.



tutorial5c : after refreshing, the tunnel sits inside the mountain, and the mountain has been given a hole.

That's the payoff of the annotation: it describes a **stretch of track**, and when refreshing, Raily itself looks for whatever stands in the way there. A tunnel made by hand is a body, and knows nothing.

What's still stuck in the ground: cut and fill

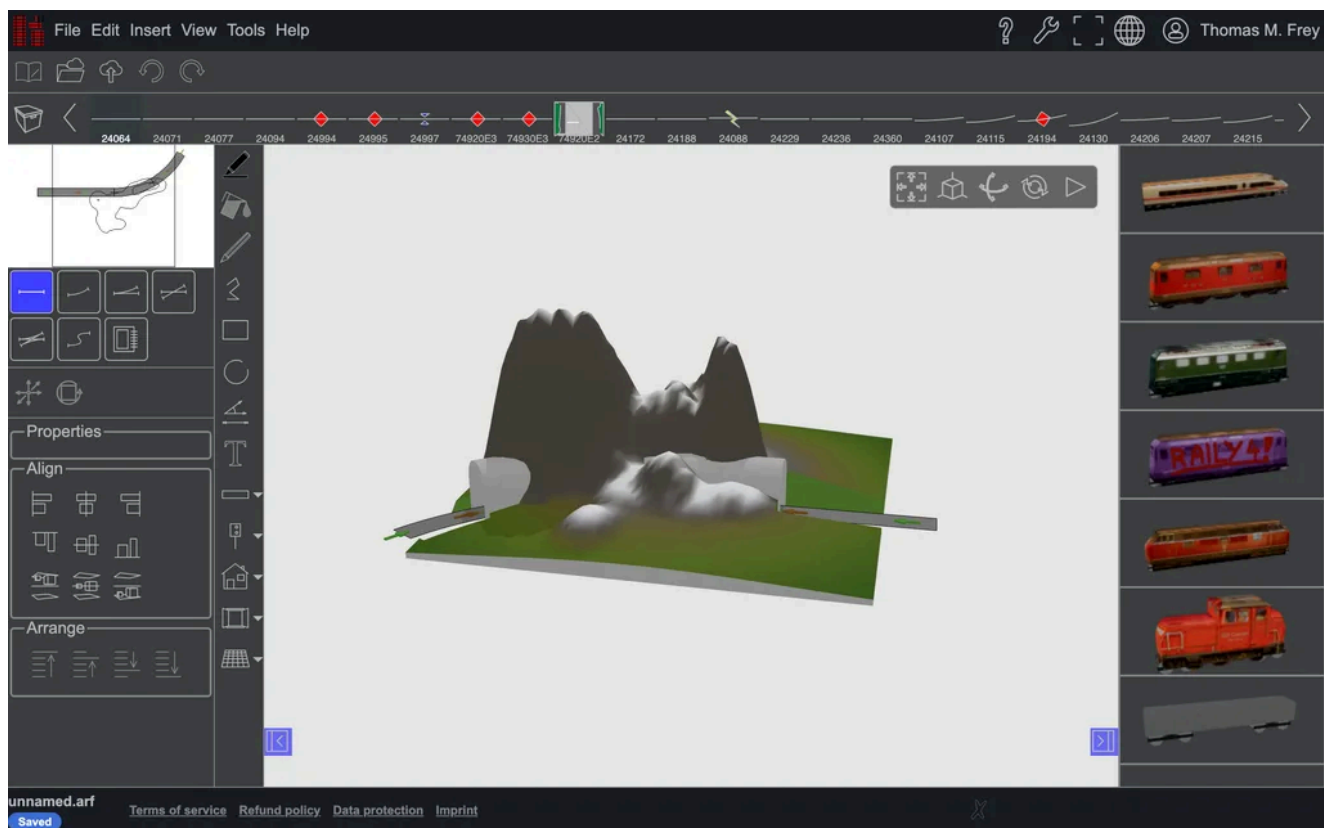
Look at `tutorial5c` at the **tunnel mouths**. The tunnel sits cleanly in the mountain, but the track in front of it and behind it still runs straight through the terrain. What's missing is the forecourt a real portal has.

There's a second kind of annotation for that. In the annotations' multi-tool button, choose ✕ **Add cut & fill annotation**.

It works exactly like the tunnel annotation, and the rules from [step 3](#) apply unchanged: lie on the track, a counterpart bounds the region, then **refresh**.

The difference is what gets built. The cut & fill annotation cuts away an area above, down to the given height, and inserts an embankment of the given height below.

Set a pair at each tunnel mouth, from the portal side outwards, and refresh.



`tutorial5d` : with one cut at each end, the track runs freely out of the mountain.

You set the shape of the cut on the annotation: **Trackbed width**, **Slope** of the batter, **Depth**, and the materials for batter and trackbed. More on that under [Track annotations in detail](#).

Color on the terrain

A terrain block starts out colored "Alpine": meadow at the bottom, rock above it, snow on top. Other palettes are under **Surface colouring** in the property panel.

Match colours to terrain

For the colors to fit your mountain, Raily needs to know what height range they run over. The button at the bottom of the tool panel reads it from the actual heights. One press, and the snow lies on the summit instead of on the plain.

More under [Terrain editing \(terrain & sculptor\)](#).

Next

- [Getting started 6: Leaving the flat](#)

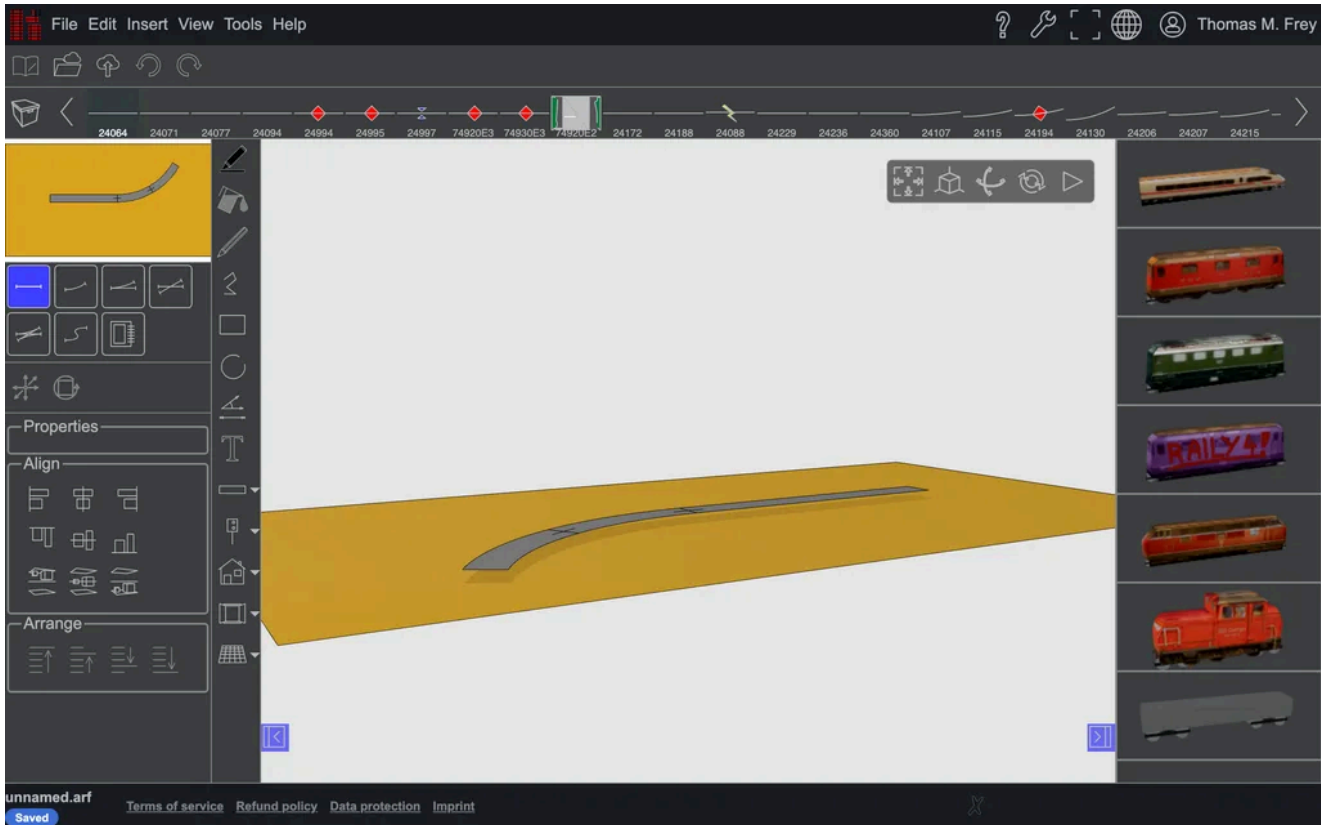
See also

- [Getting started 4: Tunnels](#)
- [Track annotations in detail](#): the refresh dialog and the parameters of cut and fill
- [Embankment and ramp](#): the track heights the terrain adapts to
- [Import a heightmap](#): a terrain from a greyscale image, instead of shaping it by hand

Getting started 6: Leaving the flat

We have mountains now, but our track is flat. In this step we bring it up, and stumble over the same thing twice along the way.

Load **File** → **Example plans...**, folder **Tutorial**, plan `tutorial6a` : a table with a track on it.



`tutorial6a` : a grey band on the table. Look closely at the right-hand end: it floats there.

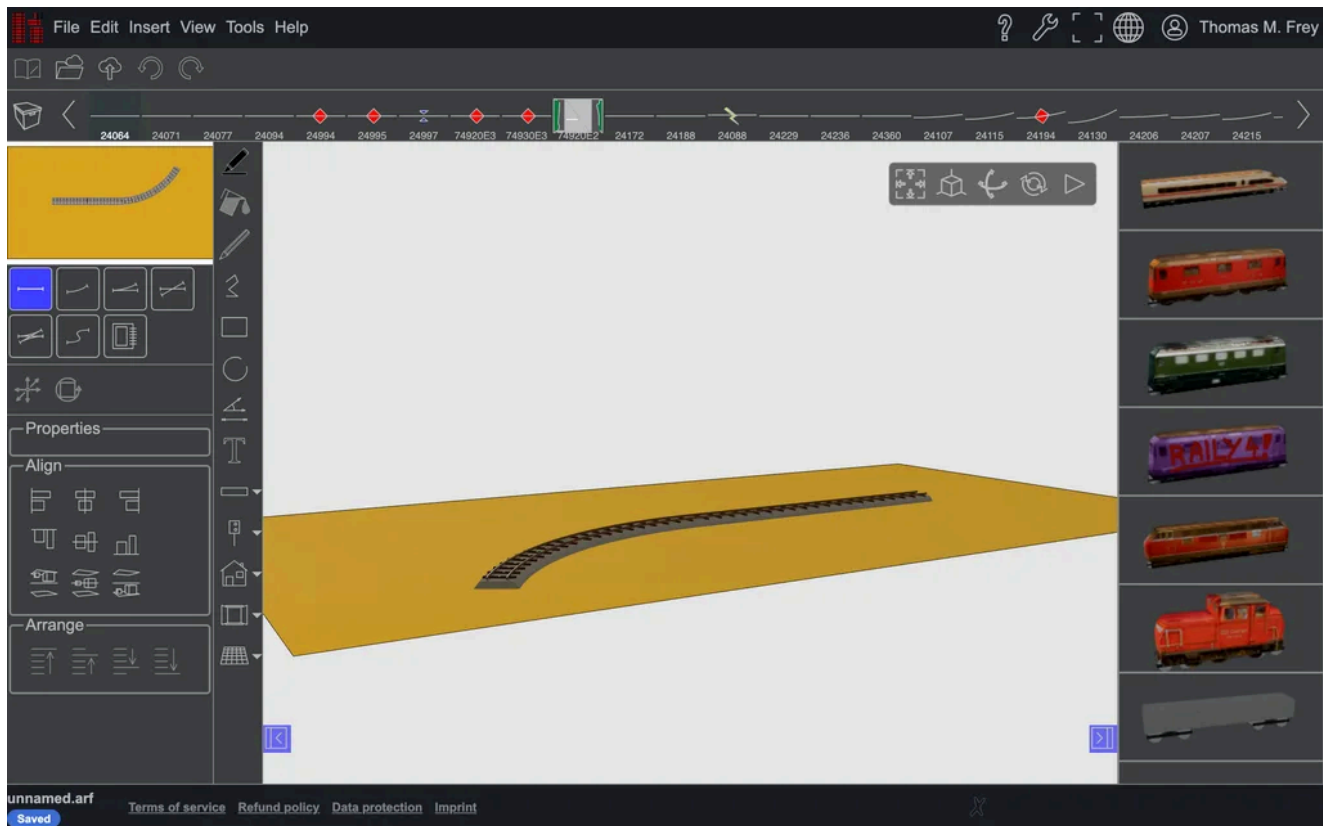
Hold on... why is the track above the ground?

Because the height Raily stores for a track is **not** its underside.

The **reference point** of a track, the point whose height Raily keeps, sits at the **railhead**. Everything a track has below that hangs under this height: rail, sleeper, bedding. When placing it at level 0, Raily therefore added the build depth; in the example plan the railhead sits at 10.3 mm. And because a track is drawn in 2D as a flat band, that band floats above the table.

Showing the track as a body

Select the track, call up **Edit Rail Display...** with the **right mouse button**, and switch on the **3D** option under **Rendering**.



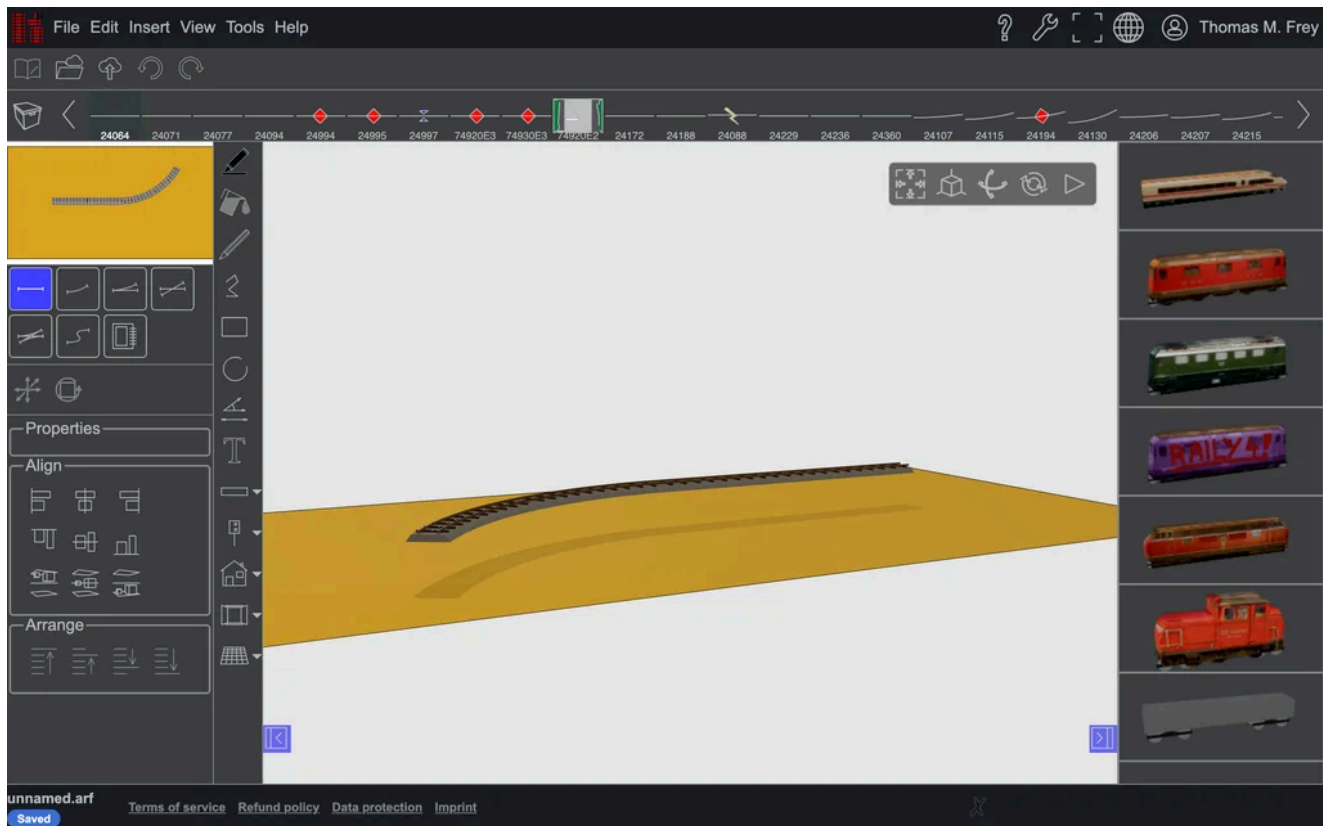
tutorial6b : now with bedding, sleepers and rails, and the body stands cleanly on the table.

The band has become a track, and you can see what was invisible before: under the railhead sits the **build depth**.

For the whole plan at once, that is available via [Track display](#).

Moving it up

With **Translate**, or the **t** key, you move the selection in **three** dimensions, including upward.



tutorial6c : the track lies up top.

How do you get from the bottom to the top?

The track is up there now, but nothing leads up to it. It needs a **gradient**.

The **r** key, or **Rotation**, would be the obvious move: take hold of the track, tip its front up, done.

Not with the rotate tool

A tilted piece of track is rigid and does not hold the gradient through a curve. There is a dedicated tool for a gradient.

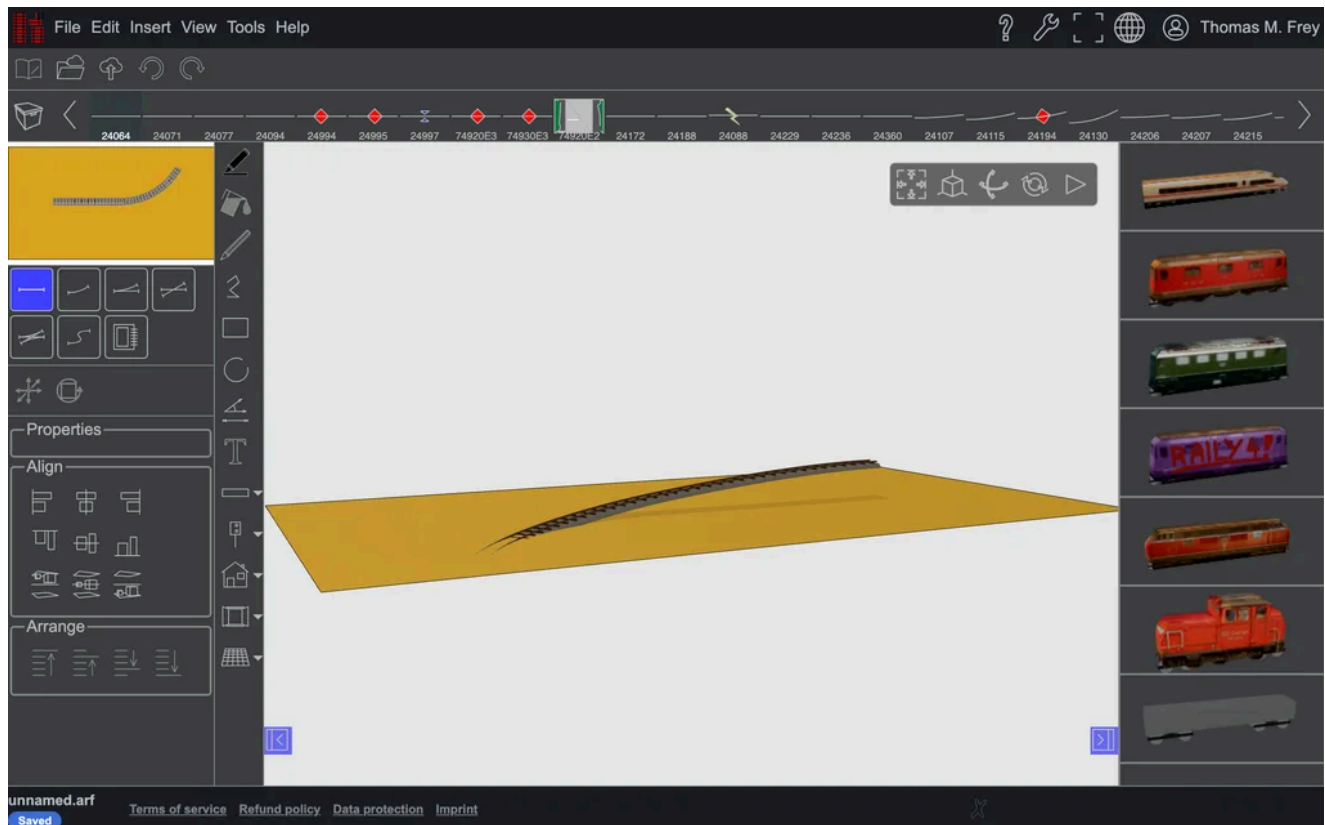
Select the track pieces of the ramp instead, and call up **Edit ramp...** with the **right mouse button**. The **ramp editor** asks for three things:

Field	Meaning
Start height	the height at one end of the chain
End height	the height at the other end
Apply	works the heights out onto the track in between

The editor can do more: transition, cant, maximum grade. The three are enough to begin with; everything else is under [Embankments and ramps](#).

Enter **0** as the start height and the height you want as the end height, and press **Apply**.

Oops, now the track is stuck in the ground



tutorial6d : at the bottom, the track has sunk into the table.

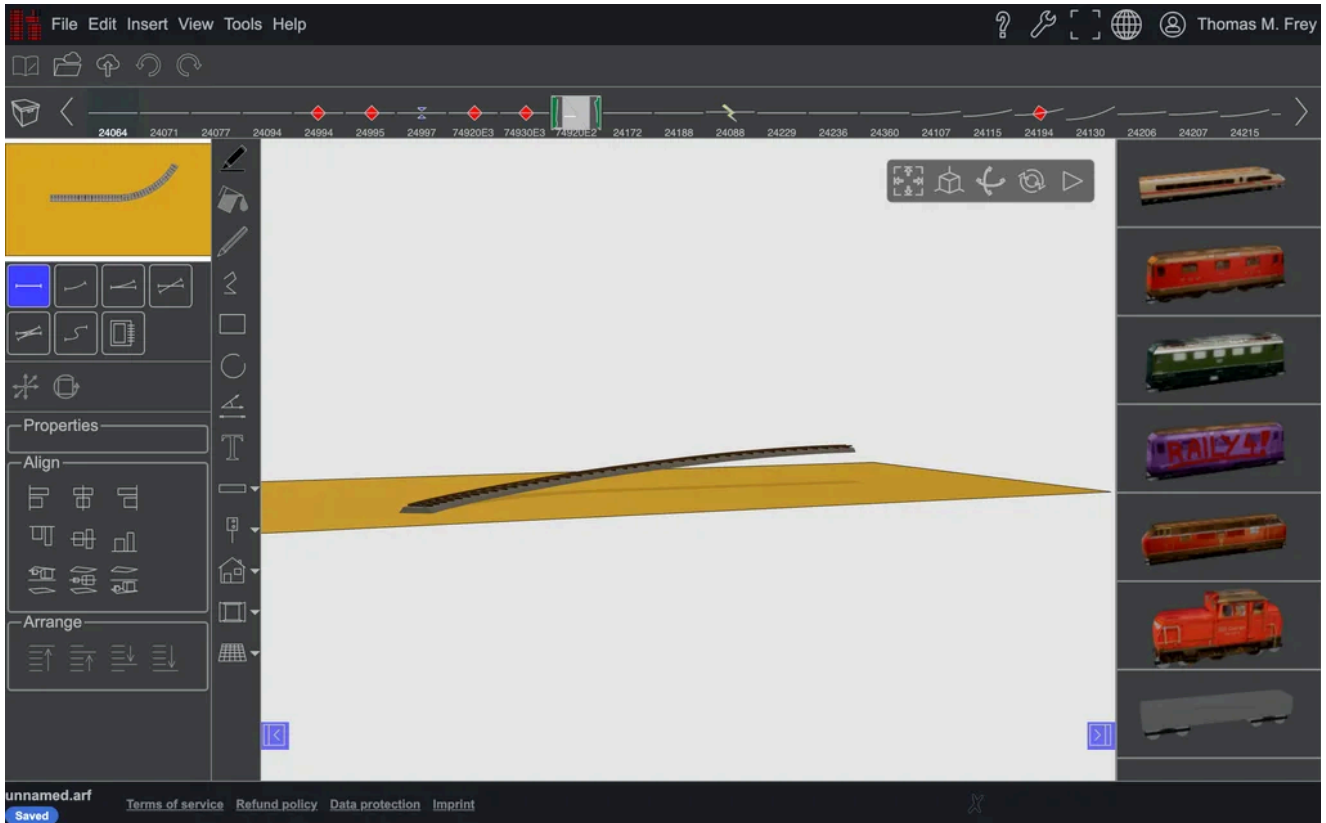
The same reason as at the start: **0 is the height of the railhead**, not that of the substructure. Start at 0 and you sink the bedding and sleepers into the table.

So the start value has to be the track's **build depth**. Where to find it: select the track, **right mouse button** → **Info...**, row **Build depth**. It is broken down there: *rail + sleeper + bedding = total height*. For the Märklin C track in this plan that is **10.3 mm**.

Catalogue data, no guarantee

Build depths come from catalogues and measurements, and can be off in individual cases. If your build depends on the figure, measure it on real track; where the numbers come from is under [Technical limits](#).

Open the ramp editor once more, enter **10.3 mm** as the **start height**, and press **Apply** again.



tutorial6e : the track sets off on the table and climbs from there.

And when there is not enough room: the helix

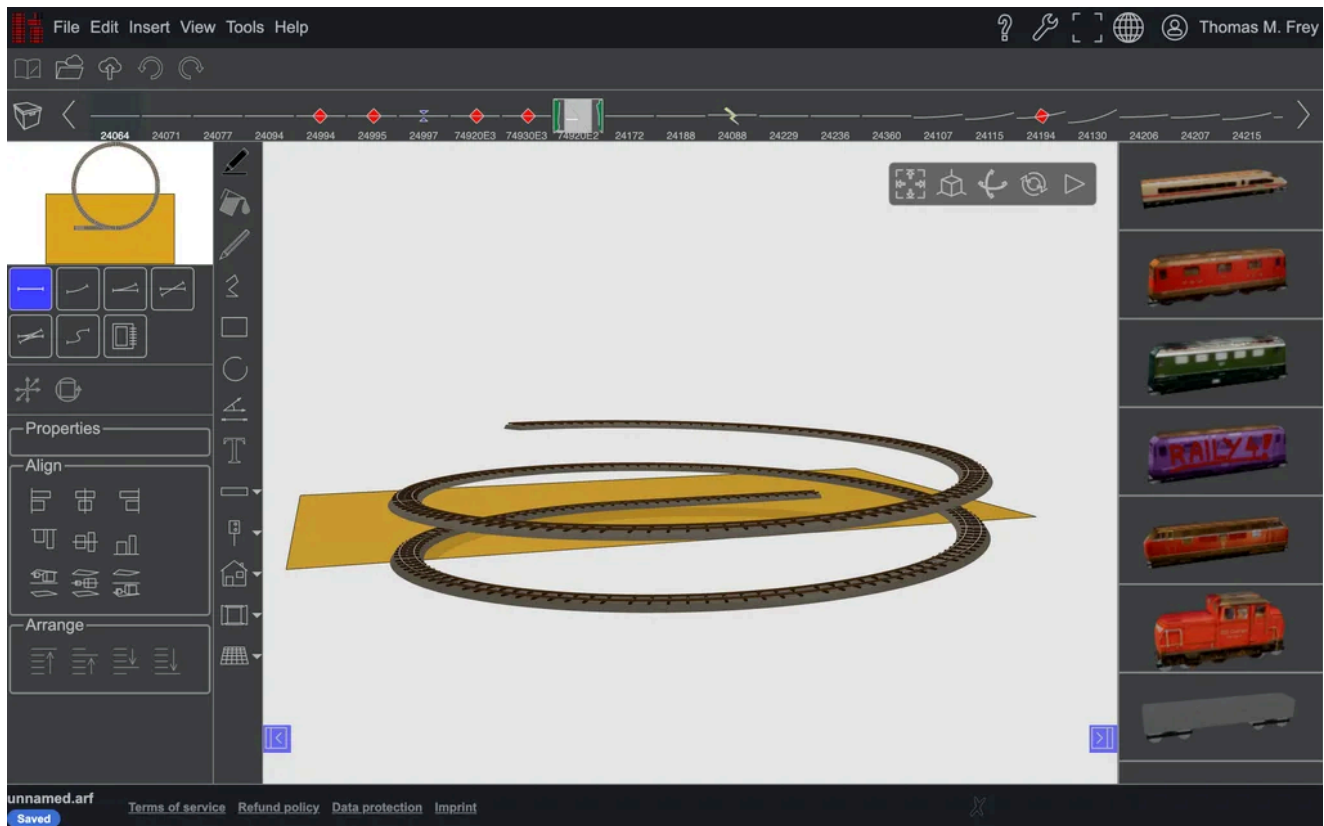
A ramp needs length. Where that is missing, you turn upwards instead.

Go back to `tutorial6b`, select the **curved track at the end** of the track and call up **Create helix....**
 Raily repeats that curve and stacks the turns.

The selection is the whole instruction: **one curve with one end still free**. The helix grows out of the free end, and every turn is a copy of that curve.

No Create helix... in the menu?

With several pieces of track, a straight track, a switch, or a curve with track on both ends, the entry is not offered at all.



tutorial6f : a curved track becomes a helix.

You enter **turns** and **rise per turn**.

Count the board in

The **rise per turn** is the distance from turn to turn, and that distance has to fit not only the train, but also the **board that the track of the upper turn rests on**, together with whatever supports it. Put in only the height of the rolling stock, and you build a helix that nothing fits through in the model.

More on this under [Create helix](#).

Next

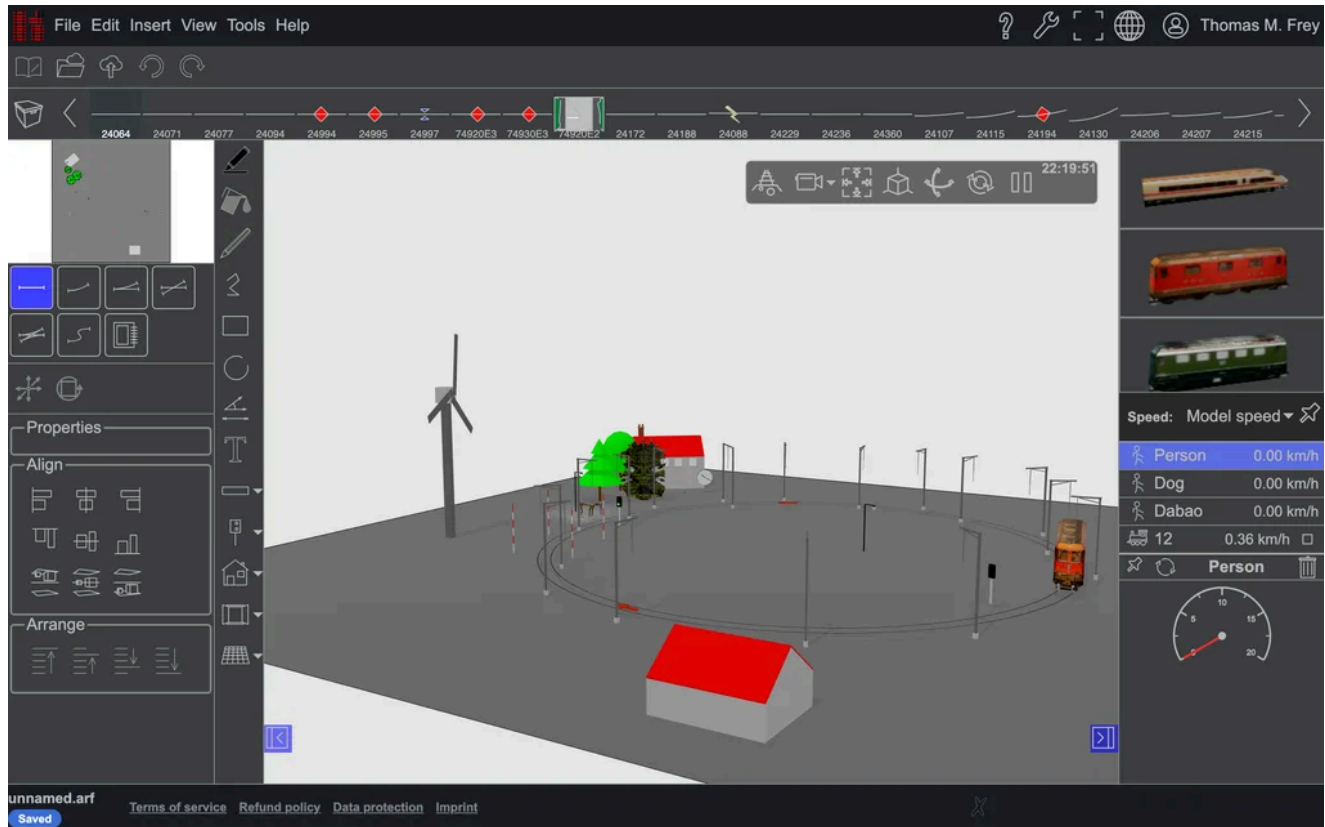
- [Getting started 7: Decorating](#)

See also

- [Embankments and ramps](#)
- [Create helix](#)
- [Track display](#)
- [Element info](#)
- [Work plane & coordinate systems](#)
- [Getting started 5: Terrain](#)

Getting started 7: Decorating

Track, tunnel, mountain and gradient are in place. What is missing is everything beside them: houses, trees, lamps, poles. This step shows where those objects come from and which two peculiarities they have that track does not.



tutorial7 : the goal of this step.

The buttons with the objects

Down in the left toolbar are four **multi-tool buttons**. Each shows the tool last used from it, and the small triangle beside it opens its list — as does a long press on the button itself, which is the easier way on a touchscreen:

Button	What it contains
Railway	what belongs to the track: signal, pole, barrier, bridge, platform, station, engine shed
Buildings	what stands as a whole: building, tower, wind turbine
Building parts	what goes onto or into a building: window, door, balcony, chimney, stairs, clock
Scenery and life	everything around it: terrain, tree, lamp, shape, walker, camera

A click on the tool hangs the object on the pointer; the next click in the plan sets it down. The tool stays on the button afterwards: ten trees are the same click ten times.

Set it up first, then place it

The **Insert** menu carries the same four lists. There, each entry first opens the object's dialog, and only after **OK** does it hang on the pointer. Either way, Raily remembers how you last set up an object of that kind.

Shaping a building

Set a building beside the track. At first it is a box with a roof.

Select it and click it **again**. On the left, the building's values appear — **Variant, Length, Height, Width, Roof shape** — and green and yellow handles for the same dimensions appear on the object itself. Type a number when you know one, drag a handle when you are looking for it.

That is how **all** these objects work, from tree to station hall.

Windows and doors belong on a wall

This is the first peculiarity, and it applies to almost everything under **Building parts**. Windows  and doors  do not lie on the table, but on a **surface**:

- **Set them in a 3D view.** From above you see no wall, only the roof.
- **They hang themselves on the wall you point at**, with its tilt and its direction. A window on a gable wall stands upright in it without you having to rotate it.

A window that has been set travels **along the wall** when dragged, so also upward and downward.

From a sticker to an opening

A window on the wall at first sits only **in front of it**. A hole appears only once you **install** it:

1. Select the window.
2. Add the **building** too with -click. Whatever was clicked last is the host.
3. Press  + .

Raily cuts the opening out of the wall, and from then on the window belongs to the building: it travels with it when you move the house. The same three steps install a staircase, a tunnel or any other solid into another.

From then on a click hits the building, not the window anymore. To get hold of the window again, select the building and **enter** it with ;  leads back out, see [Groups](#). Inside,  +  uninstalls the window again.

Why not right away, when it is set down?

Because the cutting has to be computed, and because you will usually move a window once more first. Installing is the moment at which you say: now it is in the right place.

Lamps

 **Add lamp** sets a light fitting that really does light. **Mounting** chooses between post, wall bracket, suspended, bollard and the fitting alone. With **Wall bracket** it behaves like a window and hangs itself on the wall; change that setting **before** you place it.

Lit when is a small condition. The default is `night|dusk`, from dusk until morning; `true` means always, `false` never. The same kind of condition sets a signal and closes a barrier, see [Signals, barriers and blocks](#).

All lamps light up, but only a few cast light

Every lamp that is on is bright in itself. But at most **four** lamps and headlights at a time cast their **glow** onto the ground and walls — the nearest ones, because light costs computing time. On a fast computer, set **3d lamps and headlights** to 8 in [Settings](#), group **Rendering**.

And when is it night?

In a fresh plan it is always midday, so the lamps stay off. What decides when it gets dark is the [environment](#).

Poles remember their track

This is the second peculiarity, and it applies to poles and signals. Set  a pole beside a track and it binds itself to it. Which **side** it stands on is decided by the side your pointer was on when you set it down.

Move the track later and the pole goes with it. Drag it far enough away and the binding is released again.

Next

That was the getting-started path. From here the remaining chapters describe the individual tools, the [table of contents](#) is the starting point.

See also

- [Snapping](#): how objects hang onto surfaces and tracks
- [Installing into multiple hosts](#)
- [Material selector](#)
- [Environment](#): time of day and position of the sun

Operation

Conventions used in this manual

Four things come up on almost every page. They are covered here once.

Keys

Raily accepts either key of a pair on every system. The manual names the first:

In this manual	Windows and Linux	Mac
		 ⌘
		 ⌥
		 ⇧

Mouse and finger

Right mouse button means: open the context menu. Without a right mouse button, **long-press** the object — with two fingers on a trackpad.

A single finger holds no key. If **Optimize UI for touch** is on in [Settings](#), toggles for it appear in the upper toolbar: Multi-select, Snap off, Angle lock, No link. They **latch on** until you switch them off again; see [Snapping](#).

The second click opens the editor

Many objects have their own editor. You reach it with a second click, not a double-click:

1. **Click** the object so that it alone is selected.
2. **Click it again**.

Below the track type buttons in the left toolbar are all the editors for the current selection; the second click opens the most common one. There are always two, even for multiple objects:  [Translate](#) and  [Rotation](#).

Multi-tool buttons

A small triangle on a button means: there is a list behind it. A click on the triangle, or a long press on the button, opens it. The button shows the tool last used from that list, and a short click picks that one.

Icons

An icon like  in the text is the button to look for in the program.

See also

- [Keyboard shortcuts](#): the full list, also for changing them

- Snapping

Navigation in 2D and 3D

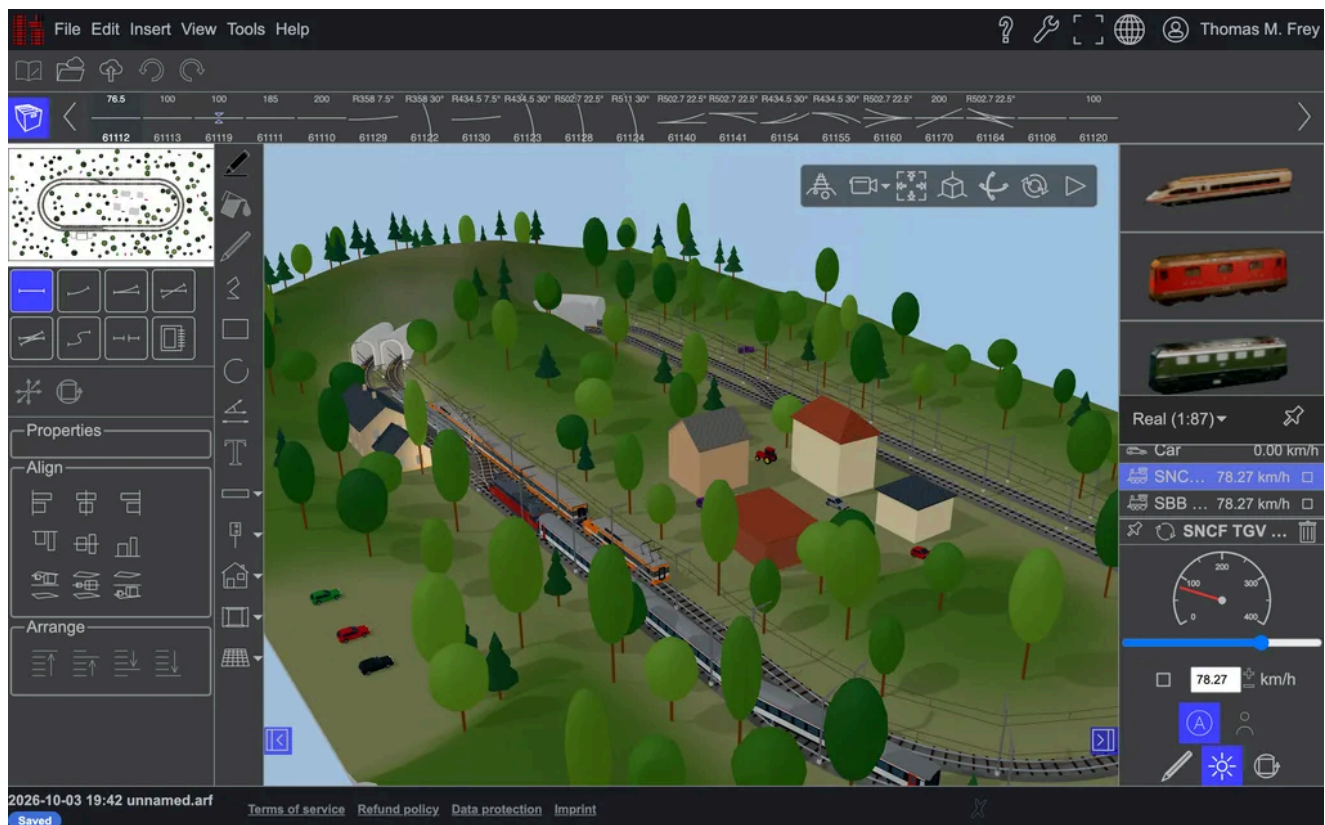
How to move around the plan is shown step by step in [Getting started 1](#). This page is for looking things up.

The views

Button	View
 2D view	flat from above, for building, measuring and printing. It brings the whole plan into view, so it's the way back whenever you've lost the overview
 3D view	in space, the viewing direction aligned with the layout
 3D view (trackball)	the same, but freely turnable around the plan
 Cockpit view	from inside a train or a walker, see Getting started 2
 Follow	from outside instead of from inside, see below

Switching trains

Each further press of the cockpit view button switches to the **next** controllable object (e.g. a train).



A plan in the 3D view.

Zooming and panning

Device	Pan	Zoom
Mouse	drag with the middle mouse button	mouse wheel
Trackpad	two-finger drag	pinch open and closed
Touchscreen	two-finger drag	pinch open and closed

2D only: **dragging along the rulers** pans exactly horizontally or vertically (see [Rulers and Guidelines](#)), and **Shift** + **dragging a rectangle** zooms to exactly that area.

Also only in 2D: **scrollbars** along the right and bottom edge. Drag one to move the view, or click beside it to move by one screen. The x in their corner hides them; **View** → **Scrollbars in 2D** brings them back.

Raily tells a mouse from a trackpad apart by the kind of event: a genuine mouse wheel zooms, a two-finger swipe on a trackpad pans. So you don't need to change any setting when you switch between the two.

If the mouse wheel pans instead of zooming

The **Zoom on scroll-wheel in 2d** setting ([Settings](#), group **Input & Interaction**) decides what the mouse wheel does in the 2D view. If it's off, the wheel pans and you zoom with **Ctrl** + wheel. In the 3D views, the wheel always zooms.

Rotating in 3D

Press and drag the mouse to rotate the camera around the point you're looking at. **Alt** -click a spot in the plan to make that point the new **rotation center**. That lets you get close to a detail without the camera pushing it back out of view as you rotate.

Dragging moves a selected object

If the drag starts on a selected object, it moves the object instead of the camera. Holding **Shift** always rotates only the camera.

Following from outside

Next to the cockpit view button sits a second one: **Follow** shows the active object (e.g. the train or walker) from outside. The small triangle beside it opens three camera styles, and the one you last picked stays on the button:

	Camera style	
	Follow from behind	behind and above the vehicle
	Follow and orbit	same distance, but you choose the direction you're looking with  -drag
	Watch from the trackside	the camera stays put, the train passes by, then it finds a new spot

What a walker or a camera is as an object, and how you control either, is covered under [Walking and cameras](#).

The plan overview

The **plan overview** at the top of the left toolbar shows the whole plan in miniature. Clicking into it centers the view on that spot; you can also **drag inside it** to slide the view across the plan that way. In the 3D views, the same point becomes the point you're looking at, and with it the rotation center. On a large layout, this is the fastest way from one end to the other.

See also

- [Selecting objects](#)
- [Laying track faster](#): pick up a track from the plan instead of looking for it in the bar
- [Snapping](#)
- [Rulers and Guidelines](#)
- [Grid](#)
- [Keyboard shortcuts](#)

Select objects

Almost every command in Raily acts on the **selection**. You already know the main actions from [Getting started 1](#); here they're gathered together, plus a few more: selecting by rail type, groups, and the two settings that turn this behavior off.

The actions

Action	Effect
Click	selects an object and discards the previous selection
 -click	adds it individually, or removes it again
Click on empty space	clears the selection
Drag a selection rectangle	everything inside it (2D view only)
Double-click a rail	everything connected to it, directly and indirectly (can be turned off, see below)
 -click a second rail	the stretch between both, see below

Without a keyboard

On a tablet, you reach  -click via the toggle  **Multi-select** in the top toolbar. It **latches**: as long as it's on, every click adds to the selection or removes from it. The toggle only appears when **Optimize UI for touch** is on in the [Settings](#).

The double-click can be turned off

Select connected elements on double-click in the [Settings](#), group **Input & Interaction**, controls it. Turned off, double-click selects only the rail you clicked.

The stretch between two rails

If there are several paths between the two clicked rails — over a switch, a runaround track, a loop — Raily looks for up to **eight** of them and offers them by length, shortest first. Each further  -click on the same second rail switches to the next path; after the last one it starts over from the beginning. So you keep clicking until the path you mean is marked.

If Raily finds no path, only the rail you chose first remains selected.

By type, not by place

Right mouse button → **Select all of type** selects every element in the plan that comes from the same module and the same rail type as the one you clicked. That way you can, for instance, change the color of all bumper blocks at once, without hunting for them.

Groups

Objects that belong together can be combined into a **group**, which is selected and moved as one, see [Groups](#).

See also

- [Navigation in 2D and 3D](#)
- [Rotate objects](#) and [Scale elements](#)
- [Groups](#): treat several objects as one
- [Track annotations in detail](#): mark stretches of track permanently, instead of selecting them anew every time
- [Keyboard shortcuts](#)

Snapping

While dragging, an object snaps to nearby **targets**: a grid line, a guide line, a special point on another object, a free track end.

Configuring snapping

The magnet icon  in the **top-left corner of the rulers** opens the window for snapping and grid:

Toggle	Effect
Snap to Grid	to the nearest grid line
Snap to Guide Lines	to the nearest guide line
Snap to Special Points	to corners, endpoints and midpoints of other objects, and to free track connectors
Snap direction	also adopts the target's orientation
Snap to Ground	places the dragged object on whatever lies under the pointer

The magnet icon only exists in the 2D view, and only while the rulers are on. The settings still apply everywhere, though; the path that's always open is [Settings](#), group **Snapping**. That's also where the **snap distance** is.

Snap distance counts in screen pixels

The further you're zoomed out, the more of the plan the same range covers. If a click misses the target you meant, zoom in closer.

If several targets are in range, Raily decides per axis, separately: guide lines before grid before points and connectors.

Snap to Ground

The other toggles decide **where on the map** an object lands; this one decides **how high**: on the topmost surface under the pointer, be it terrain, a roof, or a bridge. It's on by default and also applies when placing an object. Without it, an object arrives at the height of the work plane — inside a mountain, for instance, right in the middle of it.

The ground has no say when something else already holds the object (a track, a wall), when you're dragging on a tilted work plane, or when you turn off snapping with .

Keys while dragging

Key	Effect
	Turns off all snapping. Move the object first, then press  — otherwise the view rotates
	turns off Snap direction only
	forces fixed angles
 + 	prevents a connection from forming when you drop

On tablet, there are toggles for this in the upper toolbar; see [Conventions](#). They **latch on**, unlike a held key.

Surfaces and track

- A window, door, balcony, and a lamp on a wall bracket snap **only to surfaces**. Point at the wall, not next to it.
- A pole, signal, and [track annotation](#) snap **to the track** and remember it: move the track and they follow. The **side** is decided by the pointer when you drop it. Pulled far enough away, the binding lets go.

Highlight snap points on selection, in Settings, shows the point that the next drag would use.

See also

- [Grid](#)
- [Rulers and Guidelines](#)
- [Work plane and coordinates](#)

Grid

The grid is the graph paper under the plan: fine lines in the 2D view that you can [snap](#) to.

Turning it on and off

The magnet icon  in the top-left corner of the rulers opens the window for snapping and grid (2D view only). At the top of it are:

- **x:** and **y:** are the grid's **base spacing**, in millimeters, set separately per direction.
-  **Lock** keeps the ratio of x to y fixed while you change one of them. A 100 × 50 mm grid then keeps the same aspect ratio when enlarged.
- **Show Grid** toggles the display on and off.

Snap to Grid is a separate switch in the same window: display and snapping are independent of each other. Both are also in [Settings](#), the path that stays open in 3D as well.

The visible spacing adapts, the base values don't

When zooming out, a 10 mm grid would eventually merge into a grey area. Raily therefore multiplies the spacing by **10** repeatedly, until there are at least **10 pixels** on screen between two lines: 10 mm becomes 100 mm, then 1000 mm.

Two consequences:

- Your entered base values stay unchanged; only the display gets coarser.
- It only goes **upward**. When zooming in, the grid does not get finer than the base value; if you want to work more finely, enter a smaller value.

See also

- [Snapping](#)
- [Rulers and Guidelines](#)
- [Work plane and coordinate systems](#)

Rulers and Guidelines

In the 2D view, two rulers sit at the edges: one along the top for X, one on the left for Y. They don't just show measurements — they're also the tool you use to pan the plan in a straight line and to place **guidelines**.

You can turn the rulers off with **View** → **Rulers in 2D**, or with **Show rulers in 2d.** in [Settings](#), group **Appearance & Workspace** — that also hides the magnet icon in their corner, and with it the quick route to the [Snapping](#) settings. Their spacing follows the [grid](#), including its automatic coarsening when you zoom out.

Panning the plan in a straight line

Drag **along** a ruler:

- top ruler → pans **horizontally**
- left ruler → pans **vertically**

This is the easiest way to follow a long stretch of track without drifting sideways.

Placing a guideline

A guideline is an infinitely long horizontal or vertical line in the plan that you can snap to. Two ways:

- **Drag it out of the ruler:** drag into the plan, across the ruler's direction. Releasing creates the line. Drag the pointer back into the ruler and the operation is cancelled.
- **Double-click the ruler:** places a guideline at that spot right away.

It's best to pull out the most important ones right at the start: the edge of the layout, or module joints.

The direction you drag decides: dragging across the ruler creates a guideline, dragging *along* it pans the plan.

Changing a guideline

Every guideline has a colored **handle** in the ruler that runs across it — a vertical guideline's handle sits in the top ruler, a horizontal one's in the left ruler. If the line lies outside the visible area, its handle is missing too.

- **Dragging the handle** moves the line.
- **Clicking the handle** without dragging opens a small window with three things:
 - the **Position** in millimeters, exact to type in,
 - a **color picker**, to tell guidelines apart by purpose (edge of the layout, track centerline, module joint),
 -  **Delete**.

Snapping to guidelines

Guidelines only take effect when **Snap to Guide Lines** is turned on, in the same window as the [grid settings](#), reachable via  in the ruler corner or via [Settings](#). When snapping, guidelines take [priority over the grid](#): wherever you've placed a guideline, that's the one you mean.

See also

- [Grid](#)
- [Snapping](#)
- [Navigation in 2D and 3D](#)

Work plane and coordinates

A plan is three-dimensional; your pointer is not. A click is a *ray* through the scene, and the **work plane** decides at what height it arrives.

As long as you build flat on the table, it lies horizontal at height 0, and you don't need to worry about it. It gets interesting on a second level: a mountain station, a mezzanine.

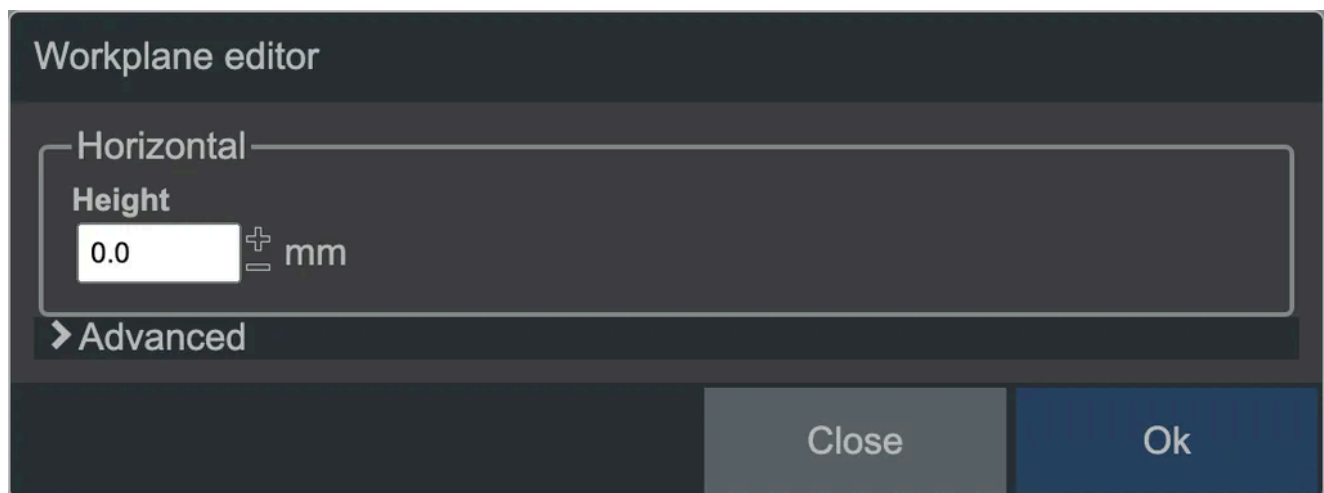
What it determines

- **Where drawn things land:** polylines, rectangles, and paths lie on the work plane.
- **How an object travels while dragged:** parallel to the work plane, so it keeps its height.
- **Where heights are measured from:** [embankments](#) and [ramps](#) measure from it.

Anything attached to a **surface**, a window for instance, goes by its wall instead of the work plane.

Setting the work plane

Edit → Edit Workplane...



The workplane editor: the normal case above, "Advanced" below it for tilted planes.

The **Height** field is almost always enough: 0 for the table, 400 for the level above it. Entering a value there also levels out a tilted plane.

Under **Advanced** you can specify an arbitrarily tilted plane, via origin and normal, three points, or its equation. This rarely helps while planning and quickly gets confusing.

Coordinates

Raily calculates in **millimeters**: **X** to the right, **Y** to the back (upward in the top-down view), **Z** upward.

Model scale or reality

Raily shows lengths either in model scale or at real-world size. Most editors carry the switch at the top of the properties area; otherwise **Model scale for lengths and sizes** in the [settings](#) applies. If a number is off by roughly the factor of your scale, that's almost always why. The work plane's height is always in model scale.

Every element also has **its own** axes. When [rotating](#) and [moving](#), you choose between world and object.

The display at bottom right

On the right of the status bar are the X, Y, and Z of the point under the pointer, plus, for a track, its item designation. If something is selected, next to it you'll see:

120.0 × 80.0 × 35.0 mm	width, depth, and height of the selection, as it needs space in the plan
⊥ 42.0	the height of its underside

Putting one thing on top of another

Underside plus height of the lower object is its top edge. Enter that number for the upper object as **Z** in [Translate mode](#).

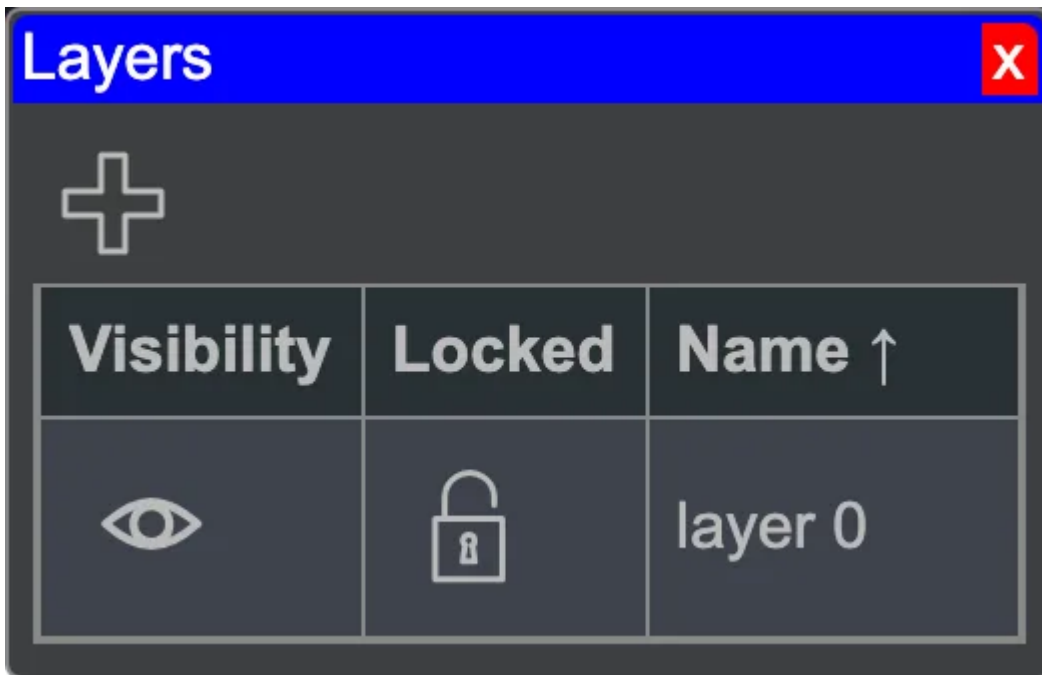
See also

- [Snapping](#)
- [Move objects](#)

Layers

A **layer** groups together objects that you want to **hide** as a set or **lock** against changes: the room with the table, the imported template, the finished station. An object sits on at most one layer. As long as everything lies flat on the table, you don't need any; as soon as things start to overlap, they save you from mis-clicks.

Tools → **Layer tool...** opens the window, as does **Shift + L**. It stays open while you keep working.



The layer tool window.

The list

Column	
Visibility	👁 visible, 👁 hidden
Locked	🔓 unlocked, 🔒 locked
Display	normal, faint, wireframe, mesh, x-ray: see below
Name	the name you give it

Hover over a row, and two buttons appear by the name: ✎ **Rename** and 🗑 **Delete Layer**. The ➕ button below creates a new layer, visible and unlocked.

The last layer stays

The delete button disappears once only one layer is left. Deleting a layer doesn't delete its objects. They then belong to no layer any more, and are always visible and selectable.

What hidden and locked mean

- **Hidden** is not drawn, neither in 2D nor in 3D, and doesn't come along when [printing](#) either.
- **Locked** is drawn, but cannot be selected: no click, no selection rectangle, no **Select all**. Whatever was already selected drops out of the selection.

Locked is often more useful than hidden

If you just don't want to move an object by accident, lock its layer instead of hiding it. That way you can still see what you need to keep clear of.

How a layer is drawn

Faint draws the layer pale, as a background to work against. **Wireframe** draws only its lines and outlines, without fills; a curved surface gets rings and meridians, like a globe. **Mesh** draws every triangle its surfaces are made of, the way Raily 4 drew a wireframe layer. **X-ray** makes its surfaces see-through, so what lies beneath or inside shows. The display applies in 2D and in 3D; printing and export draw every visible layer as normal. Plans from Raily 4 keep their wireframe and faint layers.

Putting an object on a layer

Newly added objects are on no layer. To put them on one, select them and call up **Edit → Set Layer...**, or press . The dialog shows the plan's layers and creates a new one on request. Next to the name is the same pair of icons, so you can see what you're taking on if you put something on a hidden or locked layer.

What separate layers are worth it for

- **Room and table**: room walls, doors, dimension sketches. Gone with one click for the 3D view and for printing.
- **Templates**: the imported [image](#) you trace the plan over. Lock this layer as soon as the template is scaled correctly.
- **Finished sections**: the station that's done. Locked, it no longer gets in the way while you build the line next to it.
- **Labels and legends**: things you may need for printing, but that otherwise get in the way while planning.

Annotations bring their own layers

As soon as you place the first [annotation](#), Raily creates two layers by itself: **annotation-markers** for the markers, and **annotations** for what's generated — bands, tube outlines and the like. That lets you hide both with a single click, instead of hunting down every marker one by one. If you want an annotation's results on a different layer, you can set that yourself per marker group, under **Layer of the output**.

Layers are no substitute for groups

A layer says what you can **see and touch**; it doesn't group anything together. Whatever needs to move together belongs in a [group](#).

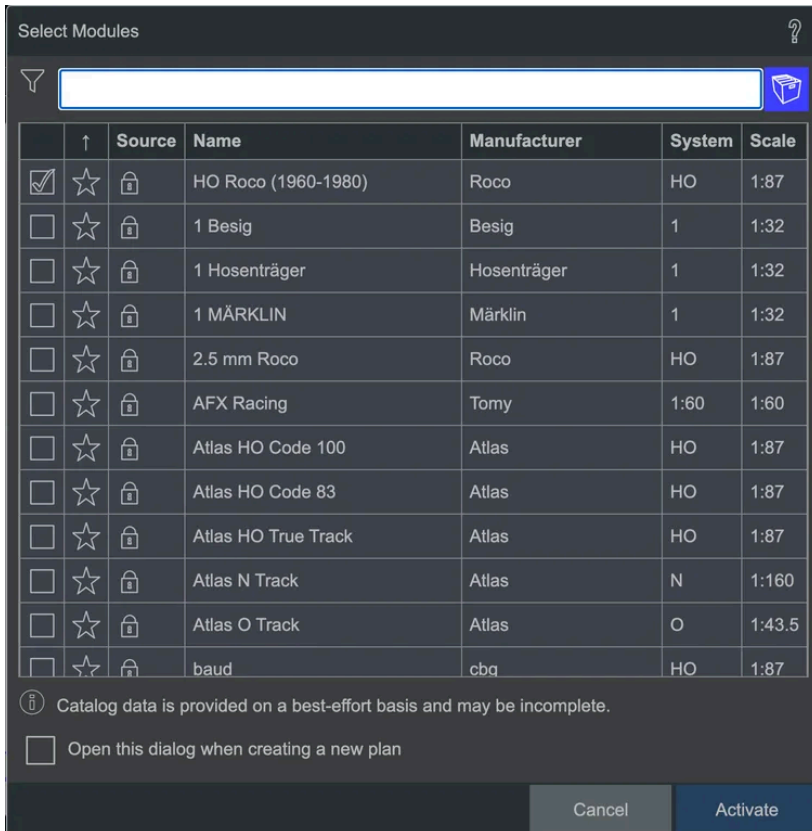
See also

- [Selecting objects](#)
- [Groups](#)
- [Printing](#)
- [Image 2D](#)
- [Track annotations in detail](#): the two layers that annotations create on their own
- [Keyboard shortcuts](#)

Planning track

Module selection

Raily knows the track of many manufacturers — too many to offer them all in one bar at once. **Module selection** decides which track systems you're currently working with: only the track of the checked modules appears in the **Quick Pick bar**.



The "Module selection" dialog.

Open it via **View** → **Module selection...** or the m key. The selection isn't a one-time decision: you can change it at any time, even in the middle of a plan.

«Module» here means track system

In Raily, a module is a **track system**: all the elements of one manufacturer's system, for example *Märklin C track* or *Roco Line with roadbed*. It has nothing to do with the *modules* of modular layout building (Fremo, standard segments).

Selecting

A **click on a row** activates the module or switches it off. Several can be active at the same time; that's the normal case as soon as two systems occur in the plan.

Mastering the list

The list is long. Five things help:

-  **Search field**: typing filters by name. Press  to jump straight from the search field into the table.
-  **Star**: marks a module as a favorite.
- **Archive box**: the box to the right of the search field. Closed, it hides clearly discontinued modules; open, all of them are shown. Open it for a plan with older track.
-  **Only your own modules**: the pencil to the right of the archive box narrows the list to the modules in your library. It only appears once you have modules of your own. The module editor, which opens a module from the same list, remembers its setting separately.
- **Column sorting by Source, Name, Manufacturer, System and Scale**. Without a sort of your own, favorites come first, then active modules, then alphabetical order.

Where a module comes from

The first column, **Source**, carries an icon; hover over it and it explains itself:

Icon	Meaning
	Comes with Raily
	From your own module library, or your version that replaces one that comes with Raily. The icon is a button , see below
	Comes with the open plan, or was loaded for this session only; it is not stored in your library

Where your library lives

Desktop app	as files on your computer. Help → About... (macOS: Raily → About...) shows the Your own modules folder under <i>"Where your files are kept"</i> as soon as you've saved your first module
Browser	logged in: on the raily.ch server; otherwise in the browser's storage

Back to the Raily version

If you've edited a module that comes with Raily, your version now stands in its place, and the  in the Source column becomes a **button**: clicking it asks whether the Raily version should apply again, and removes yours from the library. If the bundled version has changed since you edited it, the hint on the icon says so.

For a module of your own with no bundled version behind it, the click removes it from your library after asking. The open plan keeps it as its own as long as it is built with it, and takes it along when saved.

Saved plans keep their version

A plan you saved with your version still carries it. Raily asks, when you open it, which of the two should apply, and remembers the answer for exactly these two versions. So only your library gets tidied up — old plans stay as they were.

Opening when creating a plan

The checkbox **"Open this dialog when creating a new plan"** shows the module selection automatically every time you start a new plan.

About the catalog data

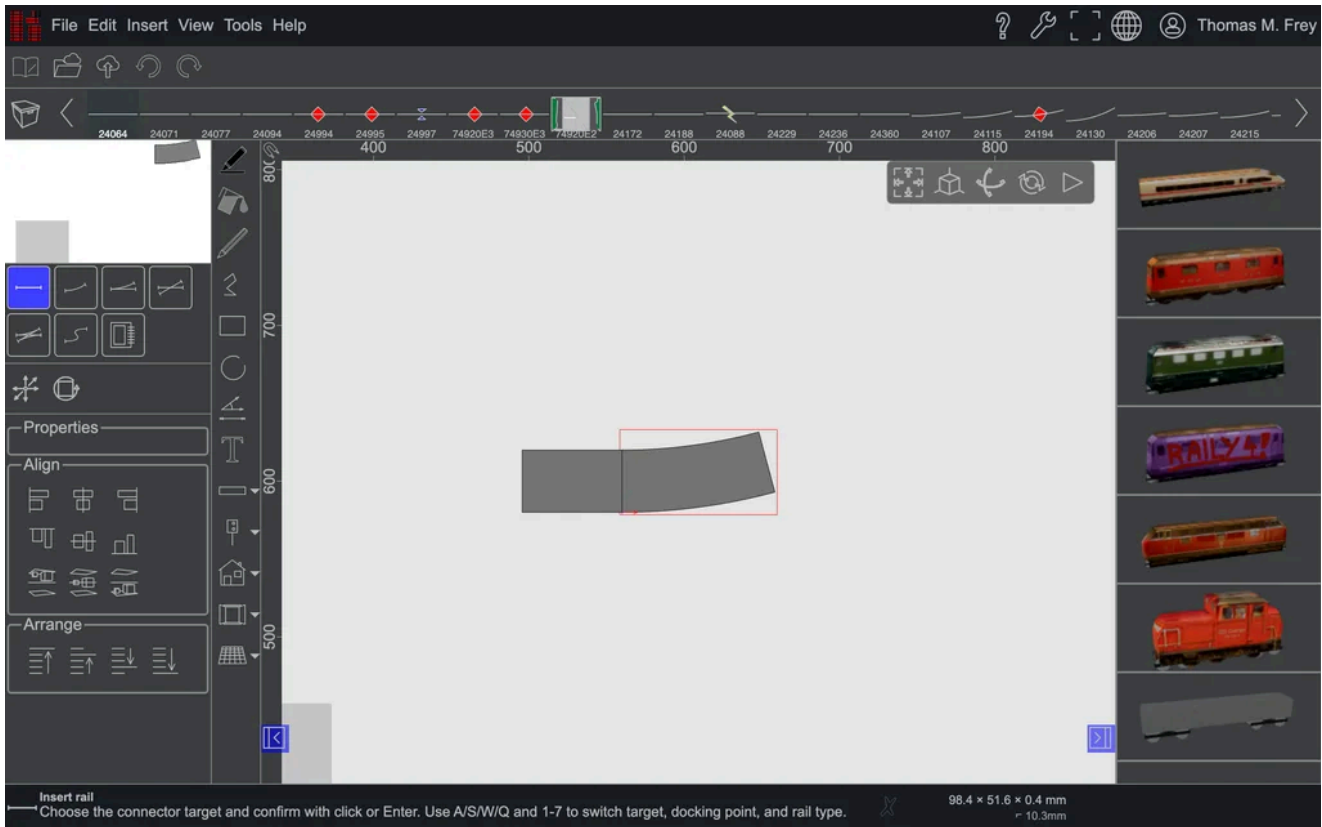
At the bottom is the note *"Catalog data is provided on a best-effort basis and may be incomplete."* Clicking it explains in detail where the measurements come from and where they run into limits, under [Technical limits](#).

See also

- [Getting started 1: Build and look](#)
- [Element info](#): the catalog data of a single piece of track
- [Flex track](#): when no track from the module fits
- [Inventory \(dialog\)](#): which parts your plan needs
- [Technical limits](#): the origin and accuracy of the catalog data

Laying track faster

Dragging a piece of track from the quick-pick bar into the plan is the way from [Getting started 1](#): it explains itself and is enough for a few pieces. You lay a whole section of track faster with the keyboard: the mouse stays put, and each piece first appears as a preview.



After **A**: the selected track is outlined in red, the preview of the next piece hangs off the free end. Nothing is placed yet.

The flow

1. **Click** the track you want to build onto.
2. **A**: the preview of the next piece appears, docked to a free end of the selection.
3. If it doesn't fit, change it with the keys below.
4. **Enter** places the piece and immediately arms the next one at the end that was just created.
5. **Esc** ends it.

A section of track is built as **A**, **Enter**, **Enter**, **Enter** ... **Esc**. Whichever piece is placed is always the one chosen in the quick-pick bar.

The keys

Key	Effect
A	next free end of the selection to build onto
S	end of the new piece that docks there
W / Q	next / previous track of the same kind
1 ... 8	kind of track: Straight, Curve, Switch, Crossing, Crossing switch, Flex track, Transition track, Special element
Enter or click	place and keep building at the new end
Esc	end

The **right mouse button** does the same as **S**, in case you're already at the mouse anyway.

A cycles in a circle

If your selection has several free ends — a station throat, say — **A** moves around their shared center. So the next key press is always the neighboring end, never one at the far end of the layout.

When nothing is selected

Then there is no end to keep building onto, and the piece **hangs off the pointer**. It docks as soon as you come near any free end in the plan, whichever one. The keys above apply unchanged: **S** chooses the docking end, **W**, **Q** and **1–8** change the piece, a click or **Enter** places it. After that the piece you just placed is selected, and the flow above continues.

Picking up a track from the plan

You don't have to look for the track you want to keep building with in the quick-pick bar: take it where it already lies. **Alt + right mouse button** on a track in the plan switches the quick-pick bar to **exactly this** track, kind and marking included. The next track you place is then the same one.

This is the fast way to keep building in the middle of a plan without knowing which element was used there. It also works **while** placing: you can continue a section of track at any of its elements without breaking the key sequence.

While moving a track

S also applies outside of placing: dragging a track, or a whole selection of tracks, through the plan, **S** chooses the **free end the dragged piece hangs from**. That decides which end docks when you let go, and therefore where the selection ends up.

See also

- [Getting started 1: Build and look](#)
- [Module selection](#): which tracks are even available to choose from
- [Snapping](#): why a track docks and when it doesn't
- [Navigation in 2D and 3D](#)
- [Keyboard shortcuts](#): the full list, also to change it

Flex track

Sooner or later no catalog piece fits: the transition between two track ends that don't line up, or the fitting piece that has to be exactly 137 mm long. That's what **flex track** is for — the only track whose shape you determine yourself.

Like any track, it comes from the **quick-pick bar**; the button  **Flex track** jumps there. Not every module contains one. Freshly placed, it has the shape it had in the module. What becomes of it is decided by the **Flex** context menu:

Entry	Result
Convert to Bezier	freely shapeable, with handles
Replace with curve...	an arc with Radius and Angle
Replace with straight...	a straight with Length

Any shape can be converted back into any other.

Bending freely

The converted track shows three handles at each end:

Handle	
Square	the endpoint
Circle next to it	how far the curve follows the end's direction before it turns away
Circle further along	rotates the direction the track runs out of the end

Drag an **endpoint** onto a free track end and it snaps onto it, taking on that end's **direction**. So you don't have to hit the angle for a kink-free transition yourself — [snapping](#) sets it for you.

Lengthening a straight

A straight flex track — freshly placed, or made with **Replace with straight...** — shows a **double arrow** at each end. Drag one along the track to lengthen or shorten it; the other end stays where it is. The end never leaves the track's line, wherever the pointer wanders, so the straight stays straight while it [snaps](#) to a track end in line with it.

With numbers

The free-form shape has no radius you could write down. If you need a number, enter the shape via **Replace with curve...** or **Replace with straight....** The dialog opens with the current values, so you correct them instead of typing from scratch.

A new arc is almost never as long as what it replaces. The end that's attached to another track stays put; otherwise it's the first end. So build a fitting piece starting from the side that's already in place.

Smallest curve radius

If a flex track is bent tighter than allowed, a **red bar** appears over the spot, in the plan and in 3D, though not on printouts.

The allowed radius is set in [Plan Properties](#): **762 mm** by default for H0, the figure for long-distance coaches. Short cars manage 360 mm too; **0** turns the warning off. **Flex** → **Smallest curve radius...** overrides the value for a single track — for an industrial siding, say.

The circle segment button doesn't draw track

 **New Circle Segment** is a [drawing tool](#). Its arc is a line in the plan, nothing a train could run on.

Closing a gap

Select the two rails whose free ends face each other and right-click. Two entries close the gap between them, and one undo takes the result back:

- **Auto-flex** lays flex track: a straight and an arc, an S of two arcs where the ends are offset, or a U where they point the same way. The entry is only offered when there is such a path. It keeps to the plan's smallest curve radius if it can; where that leaves no path, it bends as tight as the system's own curves go, and the red bar shows where. The flex comes from the active modules, or from any loaded module that has one — Märklin C has none of its own.
- **Auto-close with pieces...** closes with the straights and curves of the active modules only. A dialog lists the combinations found, fewest pieces first, each with how far its last joint misses the other rail; the gap is closed when that is within 3 mm. The plan shows the selected one; ◀ ▶ or the arrow keys step through them. No combination within that, and the entry says so: try Auto-flex, or move one end.

Both lay flat. Where the two ends differ in height, give the new track its grade afterwards with [Embankments and ramps](#).

See also

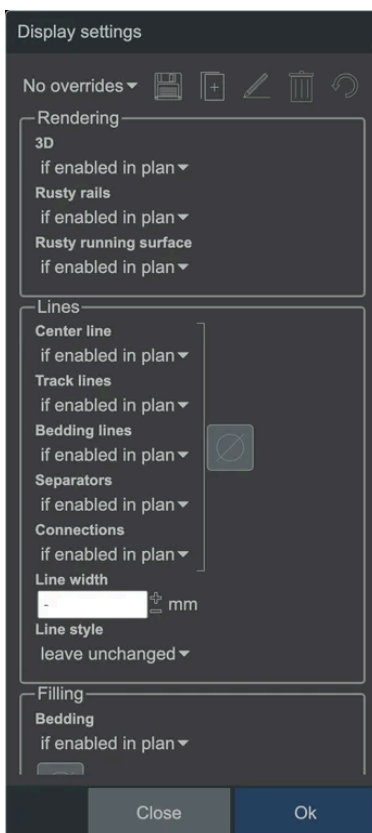
- [Embankments and ramps](#): giving a flex track a grade
- [Dimensions and measuring](#)

Rail display

How a rail is drawn — with bedding, with sleepers, with a center line, with a name — is not a fixed property, it's something you can adjust. And on two levels, at that, which you need to keep apart:

	Where	Applies to	Stays in the plan
Rail display	A rail's context menu	the selected rails	yes , it's saved
Display settings	View → Display overrides...	the whole plan	no , only the current view

The first is how your plan is *meant* to look. The second is how you want to view or print it **right now**, without touching the plan itself.



The rail display settings.

Individual rails: Rail display

Select one or more rails and use the **right mouse button** to call up **Edit Rail Display...**

Typical reasons: highlighting a stretch of track, a power circuit, a service track, the part that's being built next.

For a stretch of track: use a highlight instead

A color on the rail applies only to the rails you currently have selected. With a [2D highlight](#) you mark stretches of track independently of the individual rail elements.

Lines

- **Line width** in millimeters. **0** means **hairline**: always one screen pixel, no matter how far you zoom out. That's the default, and it's usually right — a line with a true width disappears in the overview.
- **Line style**: **solid**, **dashed**, or **dotted**. Dashed is the simplest way to tell planned track apart from track that's already built.

Filling

Bedding and **Sleepers**, each with its own color.

Markings: the symbols the catalogue draws into an element: insulated and uncoupler track, rerailers, a turntable's pit, the outline of a module board. On by default; turn them off where they clutter the picture.

Name

- **none / short / long**: whether and how fully the element name appears on the rail.
- **horizontal text**: keeps the label horizontal, instead of letting it rotate with the rail.

More on that under [Track labels](#).

Rendering

- **3D**: draws the rail as a solid body instead of flat lines.
- **Rusty rails** and **Rusty running surface**: the weathering. On the prototype the running surface stays worn bright while the rest rusts; that exact combination gives the most realistic impression.

Use for new track elements

The **Use for new track elements** checkbox makes the display you've set the default for everything you place afterwards. The way to build a whole plan in your own look throughout, instead of matching every rail one by one.

The whole plan: Display settings

View → Display overrides... lays a layer over the whole plan. The stored values of the individual rails stay untouched — they are only overruled for as long as an override is active.

Every option has three states:

State	Effect
if enabled in plan	doesn't interfere; whatever is set on the rail applies
always	forces the option for the whole plan
never	turns it off for the whole plan

Colors, line width, and text height additionally know **leave unchanged**: then the individual rails' values apply again.

What can be overridden: **Center line**, **Track lines**, **Bedding lines**, **Bedding**, **Separators**, **Connections**, **Sleepers**, **Markings**, **Line width**, **Line style**, **Rail names**, **Horizontal text**, **3D**, **Rusty rails**, and **Rusty running surface**.

Presets

At the top of the dialog sits a bar of presets. Three are built in:

Preset	What it does
No overrides	everything back to "if enabled in plan"
Print	center line, separators and markings on, track lines, bedding and sleepers off, short names, the readable blueprint
Animation	3D on, rails rusty down the sides with a bright running surface, names and markings off, the attractive picture

You can also **save** your own combination alongside them, name it, and load it again later.

The fastest way to a usable printout

Choose **Print**, [print](#), then **No overrides**. The plan itself hasn't changed in the process.

When something doesn't show up

- **Bedding turned on but not visible**: an override is probably set to **Bedding: never**. Check the override dialog first, not the rail.
- **Turntable pit or track symbols gone**: **Markings** is set to **never**, which is what the **Animation** preset does. Choose **No overrides** to get them back.
- **Lines too thin when printing**: the hairline (width 0) is one *screen* pixel, and comes out correspondingly fine on a high-resolution printer. For printing, specify a true width in millimeters.
- **Names missing**: the rail name is set to **none**, either on the rail or through an override.

See also

- [Track labels](#)

- [Print dialog](#)
- [Track annotations in detail](#): mark sections instead of recoloring them
- [Layers](#)

Track labels

Every piece of track in the plan can carry its **name**: R2 , 24230 , Turnout left 15° . That's the label that turns a printed plan into a set of building instructions: you see which piece goes where, without having to measure.

Turning it on

The label isn't a property of the plan, but is set **per track piece**, under **Name** in [Rail display](#) — which also lets you set it for the whole plan at once.

For printing, **short** is usually the right choice: the full name of an element isn't readable on a 30 mm piece of track.

Moving the label

Once a track piece shows its name, clicking it brings up two handles:

- the **text anchor**: it moves the label. On closely spaced track, this is how you pull overlapping texts apart.
- a **rotation handle**: it appears only when **Text horizontal** is *off*.

 **Reset text position and rotation** restores the default.

The rotation handle snaps to 15°

While dragging, the angle jumps in 15° steps, snapping either to the **world** or to the **track**, whichever of the two is closer. That way a text ends up either exactly horizontal in the plan or exactly parallel to the track. Hold  or  while dragging to rotate freely.

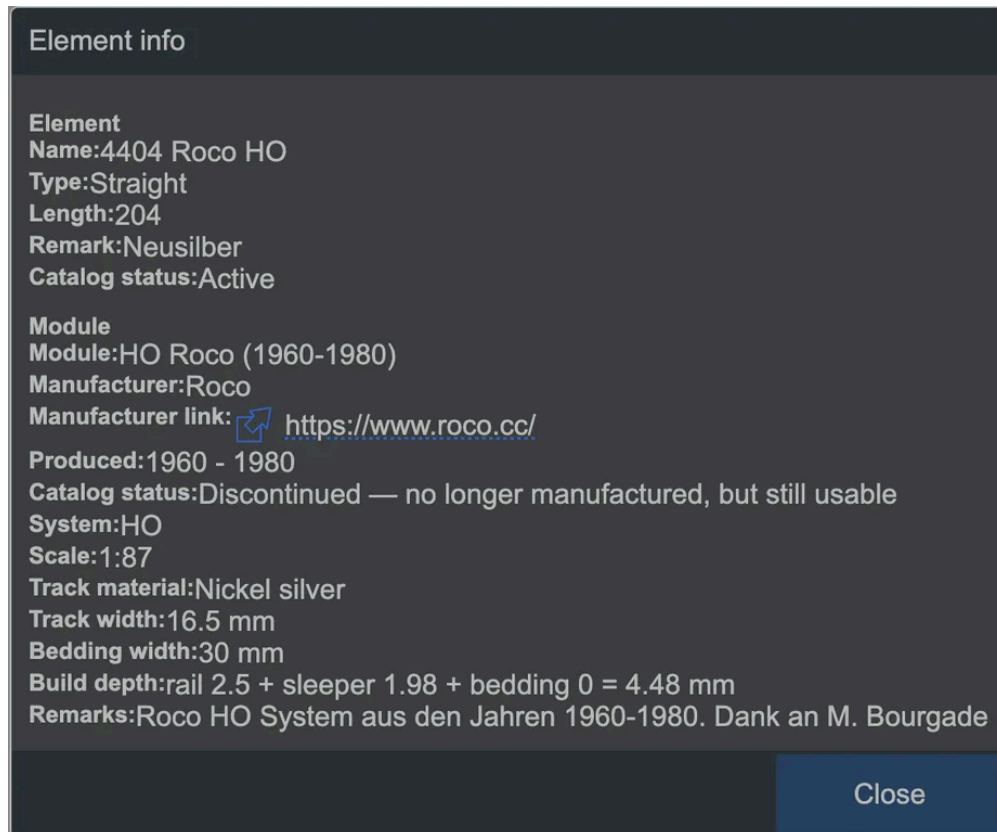
Text horizontal is the simpler choice: the text stays level no matter how the track lies, so it stays readable in every view. A text that rotates with the track looks better on a printed plan, but is awkward to read on upside-down sections.

See also

- [Rail display](#): override the label for the whole plan
- [Element info](#): the catalog data behind the name
- [Printing](#)

Element info

The element info answers the question "What kind of track is this?": for a selected element, it shows the manufacturer's catalog data, not the settings in the plan.



The element info.

Open it via **Info...** in the context menu of a selected element.

Element

- **Name** and **Type**: what the part is called and what it is (straight, curve, switch ...).
- **Length**, **Radius**, **Angle**: the key figures of the track piece. Which of these appear depends on the type: a straight has no radius.
- **Track width**: the measurement between the rails.
- **Bedding width** and **Build depth**: the height is broken down for you: *Rail + sleeper + bedding = total height*. That's the value you need when tracks from different systems are meant to meet.
- **Catalog status**: whether the part is still produced, plus **Produced From** and **Produced Till**.
- **Remark**: notes from the manufacturer or the module.

Module

Below that are the details of the **module** the element comes from: **Manufacturer** with **Manufacturer link**, **System**, **Scale**, **Track material**, and the **Catalog status** of the module as a whole.

Planning transitions

If two systems are meant to connect, compare the build depths here. A difference of one millimeter shows up as a visible step in the finished track.

See also

- [Module selection](#)
- [Inventory](#)

Embankments and ramps

The **ramp** distributes heights along a chain of track, the **embankment** builds the sub-structure underneath it. Heights first, then the embankment: it reads the heights off the track.

Ramp

Select a connected chain of track and call up **Edit ramp....** Whatever hangs off the track, catenary poles for instance, may be selected along with it.

The most important choice comes right at the top:

- **Bend the track** (default) keeps the plan. The track becomes **longer** than the plan for it, as with a real ramp.
- **Keep elements flat** tilts each piece of track as a whole, as in Raily 3 and 4. Its footprint becomes **shorter**, it cannot be canted, and on a curve it does not hold the grade. Why, see [Technical limits](#).

The display names **Track** and **Ground**, along with the difference. On a steep helix that can easily be 20% more track.

Field	
Start height, End height	the heights at the ends of the chain. Also available as handles in the plan
Transition length	the vertical easement curve at each end, so the ramp runs into flat track without a kink
Max grade	a warning only

The **average grade** of the middle section is shown. It is higher than height divided by length, because the eased ends are flatter, and it decides what a locomotive can still pull up. Model railways tolerate 2 to 4%; at 3%, 100 mm of height needs roughly 3300 mm of track.

Flatten removes all heights and cant from the selection.

Cant

With a **target speed**, the track twists through the curve. Without one (0) it stays flat.

For the model or for the prototype

You enter the speed as a **prototype speed**. **For the model**, the cant that keeps a coin lying still on the coach floor usually comes out to a fraction of a degree. **For the prototype** works with the curve the model represents, and almost always hits the maximum cant: a matter of appearance.

Max cant and **Cant ramp 1:N** limit it. The cant already builds up on the straight **before** the curve. If the straight isn't long enough, it becomes smaller, and the display says so.

Helix instead of a ramp

Where there isn't room, the [helix](#) helps. It is **not eased**; for the transition into flat track, add a short ramp. If its turns should be canted, open the ramp editor on it after creating it.

Embankment

The **cut & fill annotation** builds the embankment: place the markers, refresh. After every change to the track, refreshing again is enough. See [Getting started 5](#).

Where the embankment ends at the bottom is set on the annotation:

- **Embankment height**: the same height throughout, the underside following the gradient of the track. The right choice in terrain, where the embankment fills a dip and the rest disappears in the ground.
- **Embankment down to a level** with the **Level height**: the embankment stands on a horizontal level, such as the baseboard. Under a ramp it grows with the track, and where the track lies on the level there is none.

Heights are measured from the **formation**, not from the top of the rail: Raily subtracts the construction depth of the track itself. A track lying on the baseboard therefore gets no embankment, although its rails stand a few millimetres above it.

With a **Depth** of 0 there is no cutting, which leaves a plain embankment.

Create cut & fill in the context menu builds the same body from the **selected track**, without markers. It is set up the same way, in the property panel.

It's a body, not a rule

A body made this way knows nothing about the track. Move the track, and it stays where it was.

See also

- [Viaduct](#)
- [Technical limits](#)
- [Terrain editing](#)

Viaduct

The **viaduct annotation** builds what carries a track where the ground is far below it: a deck and piers under it, in masonry with arches or as a deck on piers. It follows the track round curves and up grades. You place and update it like any track annotation, with two markers at the ends of the stretch, see [Getting started 3](#).

Style

- **Masonry arches** (default): piers, arches and the spandrel walls over them, one body with the deck. The model is the masonry viaducts of the Rhaetian Railway; the defaults are those of the Landwasser Viaduct.
- **Deck on piers**: a deck on separate piers, for a steel or concrete bridge or a ramp on posts.

Spans and piers

Span length is the clear opening you would like. The stretch is divided into equal spans as close to it as possible: a viaduct ends where its annotation ends, not after a whole number of spans.

With **Abutments** (default) the viaduct ends on a full pier at either end. Without, it ends on half a pier, for a viaduct that runs on into an embankment.

Pier offset slides the piers along the track until a span lies over whatever passes underneath, a road, a stream, another track. The track plan shows the piers dashed.

Field	
Pier thickness, Pier width	along and across the track; width 0 means as wide as the deck
Pier batter	how much a pier widens towards its foot
Pier projection	how far a pier stands proud of the masonry. The ledge at the springing is the impost
Plinth height, Plinth projection	a plinth under every pier taller than it; 0 = none

Height

The piers stand on a horizontal level, the **Level height**, such as the baseboard. With the level under a terrain, the feet of the piers disappear in the ground, however the terrain lies there.

The top of the deck lies under the track: Raily takes the track's construction depth off itself. **Vertical offset** moves it further.

Arches

Arch shape is round, segmental or pointed, **Rise ratio** the height of the last two. **Ring thickness** is the masonry between the crown of an arch and the deck.

Every arch is built level. Up a grade the arches therefore step from pier to pier, as built ones do, and each stays below the deck all along. Where the deck comes down, an arch grows flatter and finally becomes a wall.

Ring projection makes the voussoirs on the face stand proud of the wall, in a material of their own. It never stands further out than the pier the arch springs from; 0 = flush.

Deck

Field	
Deck width, Deck thickness	
Parapet height, Parapet thickness	the parapet on both sides; height 0 = none
Cornice height, Cornice projection	the band at the top of the face that the parapet stands on; 0 = none
Lateral offset	puts the deck beside the track, positive to the right of travel

One deck under two tracks

Put the markers on one of the two tracks, move the deck by half the track spacing with **Lateral offset**, and make it wider by the track spacing.

Materials

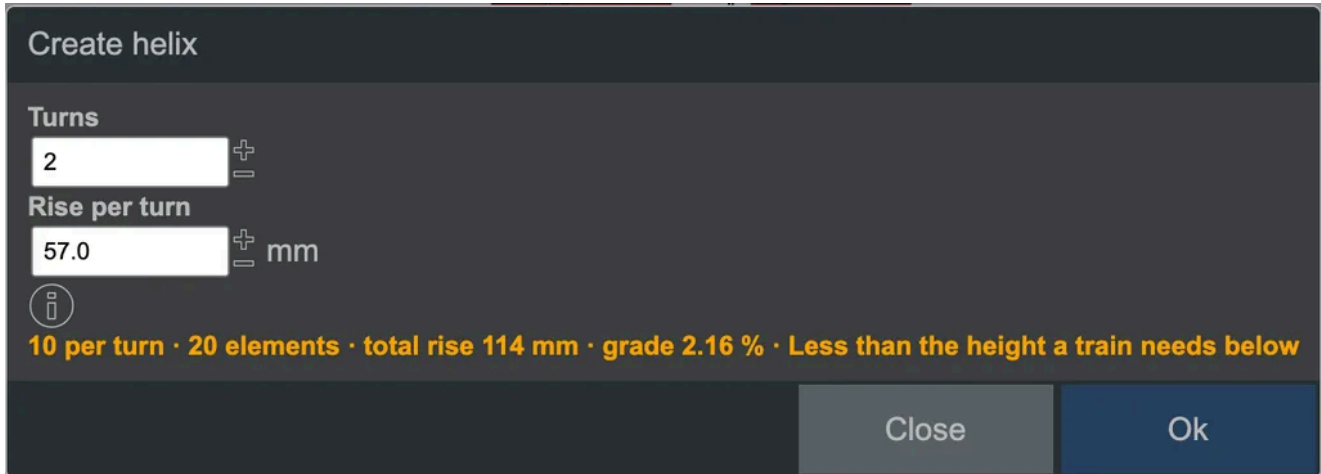
Structure material for arches and spandrels, **Deck material**, **Pier material**, **Plinth material** and **Ring material** can each be chosen. A plinth of rough stone under piers of dressed stone is as common on real viaducts as one built all of a piece.

See also

- [Track annotations in detail](#)
- [Embankments and ramps](#)

Create helix

A helix takes the track from one storey to the next within a small footprint. The dialog builds it from a curve you selected beforehand.



Create helix

Turns

2

Rise per turn

57.0 mm

10 per turn · 20 elements · total rise 114 mm · grade 2.16 % · Less than the height a train needs below

Close Ok

The dialog for creating a helix.

How to go about it

1. Select **exactly one curve with one end still free**. It is the element the helix repeats, and it grows out from that free end.
2. Call up **Create helix...** in the context menu.
3. Set the turns and the rise per turn.

No Create helix... in the menu?

The entry only appears for exactly one curve with a free end. With several tracks selected, with a straight track, with a turnout, or with a curve that has track attached at both ends, it is not offered at all.

Inputs

- **Turns**: how many full circles the helix makes.
- **Rise per turn**: the height gained per turn. The preset is 5 m at prototype scale, around 57 mm in H0.

From these the dialog calculates and shows:

- **per turn**: how many curves make a circle. This follows from the curve itself — 12 pieces at 30°, 16 at 22.5° — which is why the dialog never asks for a radius.
- **elements**: how many track pieces the whole helix needs.
- **total rise**: what it climbs in total.
- **grade**: as a percentage, calculated as the height gained per element over its plan length — the number that decides what a locomotive can still pull up.

400 elements is the limit

So the helix cannot accidentally overload the system, it is capped at 400 elements.

Clear height

Choose the rise so that enough room remains between the turns for track, train and baseboard. If it drops below **5 m at prototype scale**, around 57 mm in H0, the display changes color and reports *"Less than the height a train needs below"* as a hint that it is probably getting tight.

Check the numbers before you build

The warning is a rule of thumb. Check it against your own equipment: rail profile, the tallest vehicle, the thickness of the board plus its underlay and, if you have one, the catenary.

Raily builds the helix even with the warning showing. See also [Technical limits](#).

See also

- [Technical limits](#)
- [Embankments and ramps](#): the transition from the helix into the level track, and the cant of its turns

Mirroring

Edit → Mirror horizontally and **Edit → Mirror vertically** flip the plan over. Horizontal swaps left and right, vertical swaps top and bottom. Heights stay as they are.

Mirroring applies to the **selection**, or to the **whole plan** if nothing is selected. The axis sits at the center of whatever is being mirrored, so the plan stays where it is.

Select bound objects too

A pole or a signal only mirrors along if you select it together with the track.

A symmetric track plan

Duplicate one half and mirror the copy — faster than rebuilding it by hand.

What happens to the tracks

A track can't always simply be flipped: a left switch becomes a right switch in the mirror, and that's a different item. Raily therefore **replaces** each track with its counterpart wherever possible. The search is based on geometry; if there are several candidates, the most similar item number wins.

After mirroring, a message reports what happened: how many tracks were mirrored, how many became individual pieces for lack of a counterpart, which could only be moved, and which pieces may have come out wrong.

See also

- [Rotating objects](#)
- [Selecting objects](#)
- [Module selection](#)

Track annotations in detail

How to set and refresh an annotation is in [Getting started 3](#). This page covers what goes beyond that.

All annotations at once

When the track layout has changed, every region running over the changed spot is stale. Bringing them up to date one at a time soon becomes tedious. **Tools** → **Track annotations...** opens the **Refresh Track Annotations** dialog, which shows them all at once.

Refresh Track Annotations

☒	Region ↑	Name	Kind	Track	Mouths	Status
☒	1 		2D highlight	24360 Märklin HO	2	stale
☒	2 		2D highlight	24360 Märklin HO	4	stale
☒	3 		2D highlight	24130 Märklin HO	1	inconsistent

Refresh recomputes the marked regions from the current track layout.

Cancel

Select in plan

Refresh

The dialog for refreshing track annotations.

Each row is one region:

Column	Meaning
Region	Its number. The spanner beside it opens its parameters
Name	The region name, if the kind knows one (see below) — editable here
Kind	2D highlight, Tunnel, Cut & fill, Viaduct ...
Track	The track the marker is anchored to
Mouths	How many markers bound the region
Status	See below

Status	What it means
applied	The region agrees with the track layout
stale	Track has changed, the region has not been recomputed yet
inconsistent	The region loops back into its own mouth — usually a ring with only one mouth
unresolved	The region could not be computed
not on a track	The marker is not anchored to any track

The tick on the left chooses what gets refreshed. **Refresh** recomputes the ticked regions from the current track layout and then reports how many were applied and how many were not, for one of the reasons above. **Select in plan** jumps to the annotations of the ticked rows.

It says «stale» after loading

A loaded plan shows what it last drew; regions are not recomputed when it opens. So "stale" here does not mean anything is wrong — a click on **Refresh** is enough.

Two dedicated layers

Raily puts annotations on two [layers](#) automatically, as soon as it needs them:

Layer	Carries	State
annotation-markers	the markers themselves	visible and unlocked — you're still meant to touch them
annotations	the generated result: everything Refresh builds	visible, locked , so a mark does not get moved by accident

That lets you show or hide the whole thing with one click in the layer tool — the markers alone, when they are in the way while you build, or the generated result, when you want to see only the bare track.

A different layer for the output

Open a marker's parameters and expand **Layer of the output**: Automatic means the automatic layer **annotations**; a chosen value applies only to **this** region.

Naming regions (advanced)

Two marks on the same stretch of track are not a contradiction — one may overlap the other or lie entirely inside it. So that Raily knows which marker belongs to which region, markers carry a **name**.

The rule behind it: a marker only bounds regions of **its own name**. A marker with a different name is not a boundary — it simply does not belong there. Without this mechanism every second mark would cut the first one in two.

So you need the name exactly when two regions of the same kind are meant to overlap — a mark for the station area, say, and a second, shorter one for the approach inside it.

Not every kind knows names

2D highlight, **Block** and **Speed limit** are told apart by name. **Tunnel**, **Cut & fill** and **Viaduct**, on the other hand, are bounded by their **placement**: there, two markers facing each other form a region, and two regions may **not** overlap — one bore cannot sit inside another.

A copy with no connection (advanced)

Generate standalone in an annotation's context menu rebuilds the result as an ordinary object and detaches it from the annotation: refreshing later no longer changes that copy.

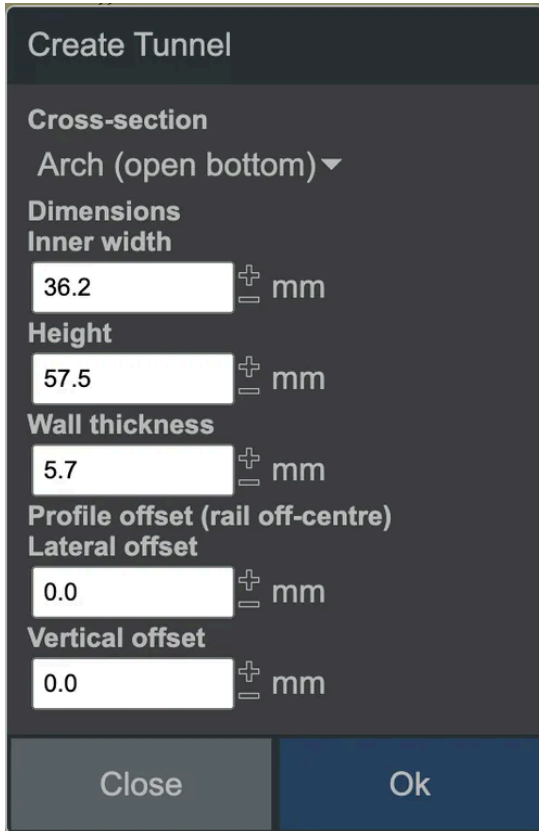
It needs an annotation that sits on a track and resolves to a consistent region — otherwise there is nothing to generate.

See also

- [Getting started 3: Annotations](#)
- [Signals, barriers and blocks](#): what a block and a speed limit do
- [Getting started 4: Tunnels](#)
- [Create tunnel](#)

Create tunnel

The dialog turns the selected tracks into a tunnel tube: Raily sweeps the chosen cross-section profile along the track path and cuts it out of the terrain.



Create Tunnel

Cross-section
Arch (open bottom) ▼

Dimensions

Inner width
36.2 mm

Height
57.5 mm

Wall thickness
5.7 mm

Profile offset (rail off-centre)

Lateral offset
0.0 mm

Vertical offset
0.0 mm

Close Ok

The dialog for creating a tunnel.

First select the tracks the tunnel should run through, then choose **Create tunnel...** from the context menu.

Cross-section

Three profiles are available:

Profile	Use
Arch (open bottom)	The usual tunnel: vertical walls, vaulted ceiling, open floor
Round (closed bottom)	Closed tube, for instance for a bore through rock
Rectangular	Box profile, for instance for an underpass or gallery section

Dimensions

- **Inner width:** the width on the inside — the space the train actually has. Not the component's outer width.
- **Height:** the height of the interior above the track plane.
- **Wall thickness:** the thickness of the masonry. It is added on top of the inner width.

The **suggested inner width** is already what a reference vehicle sweeps on the curve of this track, in a curve considerably more than the width of the car. What that figure does not know, and why it is a lower bound, is under [Technical limits](#). If you build tight, also run your longest vehicle through the curve by hand.

Profile offset (off-center track)

Where the track does not lie in the center of the tube, **Lateral offset** and **Vertical offset** shift the profile against the track. Use this to build double-track tunnels where each track has its own distance to the wall, or a tunnel with a side escape route.

See also

- [Technical limits](#): what the width suggestion does not know
- [Terrain editing](#)
- [Embankments and ramps](#)

Signals, barriers and blocks

A block publishes, under its **name**, whether a train is standing in it. A signal, a barrier, or a lamp reads this name and sets itself accordingly.

Blocks

A **block** is an annotation like a highlight or a tunnel: you place markers on the track, and Raily works out how far the block extends, through switches and across element boundaries.

1. In the toolbar's annotation button, choose  **Add block annotation**.
2. Place two markers on the track that are meant to bound the block.
3. **Right-click** → **Refresh annotation**.

All markers are initially named `block`, so a pair immediately bounds a block. Only once you need a **second** block do you give its markers their own name, under **Region name**, such as `approach` or `block2`. A marker only bounds blocks **of its own name**, so two blocks may overlap or lie inside one another.

A block with no name cannot be addressed

The name is the only way a signal finds the block. Delete it, and the signal reads an unknown name — and an unknown name is **false**, so the block counts as clear, no matter how obviously a train is standing in it.

A block **draws nothing**, or a layout full of blocks would be unreadable. If you want to see it, give it colors in its parameters.

Signals

 **Add Signal**, in the **Railway** multi-tool button of the left toolbar, places a two-aspect signal, stop or clear. It attaches to the track next to where you place it, and the side your **pointer** is on when you drop it decides **which direction of travel** it controls (the one the arrow on the track points in). If it ends up controlling the wrong direction, drag it once across the track to the other side.

Left-hand traffic: use a negative offset across track

Offset across track defaults to positive, so the signal stands to the **right** of the direction it controls — right-hand traffic, as on the continent. Where traffic drives on the left, in Great Britain or in Japan, enter a **negative** value; then the mast stands on the left, and the pointer side at drop time means the opposite accordingly.

The value that decides is called **Clear when**:

Clear when	Meaning
<i>(empty)</i>	always clear: a signal nobody has configured yet stops no one
<code>!block</code>	clear as long as no train is standing in the block <code>block</code> , the normal case
<code>!block !approach</code>	clear as soon as either of the two blocks is clear
<code>false</code>	always stop

The signal sets itself while the animation runs: what you see is always the current state.

A signal is only read automatically by trains when it is bound to the track

No matter where it stands, the signal shows whether the block bound to it is clear, but trains driving automatically only see it when it is connected to the track.

Barriers

— **Add barrier**, from the same multi-tool button, places a level crossing. It reads the same kind of condition, just named the other way round, **Open when**:

- *(empty)*: always open.
- `!block` : closes as soon as a train enters the block, and opens again once it has left it.

Swing time (s) is the time the boom needs to swing. So place the block marker a little **ahead of** the crossing.

Speed limits

A **speed limit** is again an annotation, placed like a block: ↻ **Add speed limit annotation**, two markers, refresh. Its value is a real **km/h** figure, the number that would be on the sign, not a number in plan units; Raily converts it via the scale, the same way the speedometer in the cockpit does.

Limits may overlap and nest inside one another; a 40 through the station within a 60 on the line is the normal case. Where several apply, the **lowest** one governs.

Mandated speed turns the limit into a target: a train running automatically then not only brakes for it, but also accelerates up to this value as soon as it is allowed to. Useful for an acceleration zone after the station.

What a train does with this

In the train panel, every train has **Signals**: with two states:

Setting	Behavior
Automatic	The train brakes on its own for signals at danger and for limits, and moves off again when the signal clears. This is what lets a layout run unsupervised.
Manual	You drive; Raily only reports: <i>“A train passed a signal at danger,” “A train is over the speed limit.”</i>

A name set by hand: the switch

☞ **Add Switch**, in **Building parts**, sets the name found under **Drives condition**: latching (one click on, the next off) or as a push-button, which holds it only for as long as it is pressed. It is pressed in the plan for as long as the animation runs. A lamp with **Lit when** `light` is a light switch of this kind, a barrier with **Open when** `!gate` a gate switch.

The language, in one paragraph

Wherever a condition appears — **Clear when**, **Open when**, **Lit when** — the same rules apply:

- **Names** are whatever the plan knows: the names of your blocks, and `day`, `night`, `twilight`, `dawn`, `dusk` from the [sun's position](#).
- **true** always means yes, **false** never.
- **!** not, **&** and, **|** or, plus parentheses: `!block & !approach`.
- **Empty** means *true*. A signal with no condition is clear, a barrier with no condition open.
- An **unknown** name is *false*. Deleting a block therefore does not turn a signal red — it leaves it clear.

See also

- [Getting started 3: Annotations](#): how a marker is placed and refreshed
- [Track annotations in detail](#): naming regions, refreshing them all at once
- [Environment](#): where `night` and `dusk` come from
- [Getting started 7: Decorating](#): the same condition on a lamp
- [Train composition](#) and [Placing trains](#)

Trains, figures and cameras

Train composition

The editor shows the train as a row of **wagon cards** in running order.

- **Insert:** The plus buttons at the ends and between the cards offer every vehicle of the object panel, [your own](#) among them, right at that point.
- **On the card:** move forward or backward, **Duplicate wagon** and **Delete wagon** (as long as there is more than one).

What a wagon looks like, what it runs on and how close it couples to the next are its vehicle's to say: to change them, [update the vehicle](#), and the trains made of it follow.

A change to the train can be undone like any other.

See also

- [Placing trains](#)

Placing trains

Drag a vehicle from the **object panel** onto a track. As long as the pointer is over the track, the whole train sits there exactly as it will land once you let go. Releasing sets it down.

- **Direction of travel:** the direction you drag in. If the train faces the wrong way, drag it back a bit.
- **Gauge:** the train takes on the size of the track it stands on. Dragged from the H0 main line onto an H0e branch line, it becomes smaller.
- **Starting point:** wherever you set it down is where it starts after every reset and the next time the plan is opened, see [start point](#).

The first train in the plan automatically becomes the active one, for the cockpit view and for following. You pick up a placed train the same way and drag it elsewhere; a plain click only selects it.

Vehicles you build yourself

What makes something a vehicle is the **chassis** in it, and nothing else. The ready-made wagons in the object panel are a group of solids with a chassis grouped in; build your own the same way and it is a vehicle too. To run it, save it to [your vehicles](#).

You get one from **Insert ▶ Railway ▶ Add Chassis**: place it, group it with whatever solids make up the body, and you have a vehicle.

The chassis carries the numbers the vehicle runs on, such as the track gauge of the prototype and the length over buffers. They are prototype millimetres, so one body serves every scale — the vehicle is rebuilt for the gauge of the track you put it on.

Gap when pushed and **Gap when pulled**, among the chassis's details, say how far the vehicle stands from the one coupled to it — for a screw coupling, buffers touching and 10 cm. A coupling between two different vehicles takes the mean of theirs.

Gangway gives the vehicle its passage to the next: rubber rolls as on a UIC coach, or a bellows. Which ends have one, and its size, show once one is chosen.

Coupling says what the vehicle is coupled by: a **Screw coupling**, which is what a vehicle has unless it says otherwise, or a **Centre coupler**. Between two vehicles with **None** there is nothing to click to part the train.

What it stands on is the **Axle groups** list: one entry per bogie or single axle, each with its position along the vehicle, how many axles it has, how big its wheels are and whether they are driving wheels. Add and remove them there; the list keeps them in their order along the vehicle by itself.

What the vehicle wears is three more lists of the same kind. **Lamps** says which end a lamp is at, how high and how far out it sits and what it shows — a lamp *with the direction of travel* is white at the end that leads and red at the end that trails. The lamps light with the train's **Headlight** switch, and only at the two ends of the train. **Chimneys** says where smoke leaves the vehicle, and **Pantographs** where a pantograph stands, which way it folds and whether it is raised.

The vehicle runs where its chassis stands, so the chassis belongs under the middle of the body. To fit it by hand, [enter](#) the vehicle's group and open the chassis with a [second click](#): handles appear for the length, the buffers, each axle group and a bogie's wheelbase, and each lamp, chimney and pantograph. A lamp you drag stays on the face of the body, a chimney or a pantograph on its roof — you only choose where.

Fit chassis to body in the context menu of the vehicle or of its chassis does all of it at once: it moves the chassis under the middle of the body, makes it as long as the body plus its buffers, keeps the axles at their share of the length and puts the lamps, chimneys and pantographs onto the body.

My vehicles

A vehicle you build runs once it is one of **your vehicles**. Its context menu offers **Save to my vehicles...**, which asks for a name; once it is saved, **Put on track** sets the vehicle down as a drag from the object panel does. Your vehicles are listed in the object panel ahead of the ready-made ones and are kept where your own modules are: in the browser or on your computer and, while you are logged in, in your account. A name has to differ from those of your other vehicles.

The group on the plan stays your workbench for the vehicle. Once you change it, its context menu offers **Update "name"**: every train of the vehicle in the open plan is rebuilt with the new version, and **Put on track** is offered again. **Save as new vehicle...** keeps the changed one under a name of its own and leaves the saved one as it was.

A right click on a vehicle in the object panel gives you a copy to work on, which you put down on the plan: **Edit** for one of yours, which the copy stays linked to, and **Edit a copy** for a ready-made one, which becomes a new vehicle of yours when you save it. **Remove from my vehicles...** deletes one of yours; plans that use it keep their copy. **Earlier versions...** brings back one of the last ten saved versions; the current one goes to the trash in turn.

A plan takes along the vehicles of yours its trains are made of, so that it runs the same wherever it is opened — also for someone who does not have them. It keeps the version its trains were built with until you update the vehicle. While the plan is open, its vehicles are listed in the object panel too.

When you open a plan with vehicles you do not have, Raily asks whether to add them to your vehicles or keep them **Only in this plan**. For one you have in another version, it offers **Keep both**, **Take the plan's version** — which replaces yours — and **Use my version**, which rebuilds the plan's trains with yours. Closing the question changes nothing. What you leave as it is, Raily remembers for these two versions and asks again only once one of them changes.

Coupling and derailling

If a train runs into another one **slowly**, Raily couples the two into one, without asking. Your view then jumps to the remaining train. To separate them, stop and click the coupler between the parts.

If the approach is too fast, at least one train **derails**. The cockpit bar then shows the buttons that bring it back to the start; **B** does the same.

See also

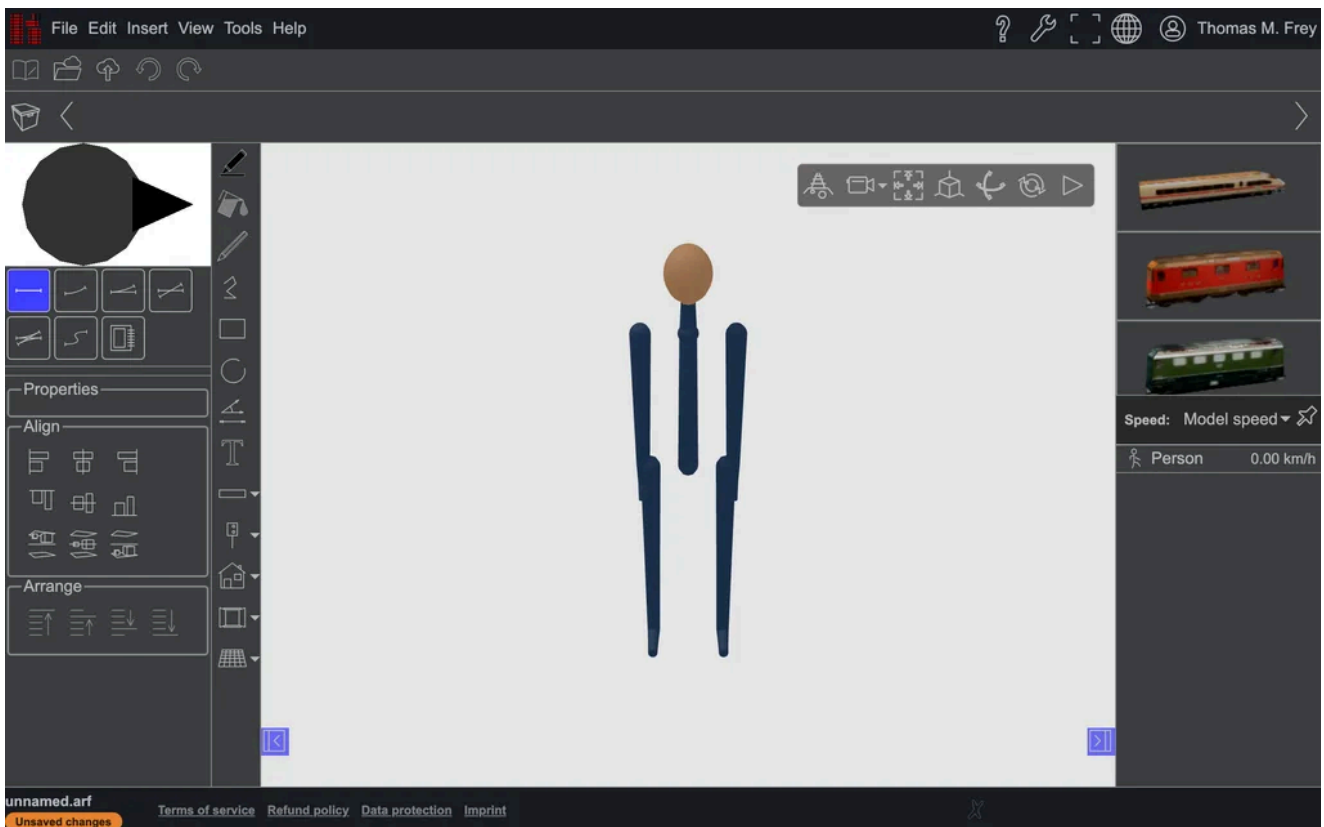
- [Train composition](#)
- [Getting started 2: Running trains](#)

- Signals, barriers and blocks

Walking and cameras

Besides trains, there are three objects you can look through: a **walker** that goes on foot, a freely flying **camera**, and a **drone**. They are in **Scenery and life**, and once placed they appear in the **object panel** next to the trains. How to switch to their view is covered under [Navigation](#).

 Walker	snaps its feet to the surface below it when placed. It walks over terrain and buildings, takes steps, refuses slopes over 45°, and falls off an edge. Person, dog, cat, horse, or giraffe
 Camera	a viewpoint for any angle neither walker nor train can reach. It flies straight through everything
 Drone	the same thing as a visible flying craft that climbs and descends with momentum. It launches two meters above the clicked point

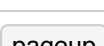
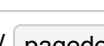


A freshly placed walker.

Speed, facing, and size are in the properties. How a drone banks into its turns is chosen with **Flying controls** in [Settings](#).

Controls

The animation must be running. Then the same keys apply as in the cab:

Key (<i>hold</i>)	
 / 	forward / backward
 / 	turn
 / 	climb / descend, camera and drone only
	the walker runs, the drone flies faster
 / 	look up and down without changing direction

The cockpit view button switches to the **next** controllable object with every press, whether train, walker, camera, or drone.

Boarding

Standing beside a vehicle, the walker boards it with  and rides along, also while the vehicle is moving; a second  sets it down beside it again. In the object panel  does the same for the selected walker. In a wagon and on a boat's deck it walks on, held by floor, walls and ceiling, on deck by the rail. In a car, a bus with a low ceiling or the tank it sits, looks out ahead, and cannot walk about. There are no doors yet, so the key is the only way in and out. If the vehicle tips over, the walker stands beside it. The **pin** stores, with the starting point, what the walker is riding in: after opening, it boards it again as soon as the vehicle is there. For a train the train itself has to be stored, which  at the top of the list does for everything at once.

Starting point

Walker, camera, drone, and train each have a stored starting position. That is where they begin when the plan is opened; wherever they have moved to since is not stored in the plan. Move one like any other element, and its starting point moves with it.

In the object panel, each entry has two buttons:

- The **pin** stores the current position as the starting point.  at the top of the list does the same for **everything** at once, the simulation clock included. That covers whether the animation is running: pinned while it runs, the animation starts by itself when the plan is opened and after .
-  **brings** the object back to its starting point.  does the same for the object you are currently controlling, for example after a crash.

 next to **Start animation** brings everything in the plan back, and sets the clock to its stored start.

See also

- [Navigation in 2D and 3D](#): cockpit view and following
- [Placing trains](#)
- [Environment](#): the simulation clock

Objects and shapes

Move objects

Most of the time you just drag an object into place. **Translate mode** is for when it has to be exact. You reach it with  or via  **Translate** in the left toolbar — also for multiple objects.

Set a position

With  **World axes**, **X**, **Y** and **Z** appear at the top. Change a number and the object moves there.

Which point of the object the numbers refer to is set by the **anchor**. The default is the **center of the underside**: Z is then the height the object stands at. Expanded, the anchor shows a box with corners, edge midpoints and face centers; choose, say, the left edge if something needs to sit flush against a wall.

With multiple objects, the whole selection moves until its anchor sits at that point. Spacing and orientation stay the same.

Move by an amount

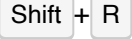
The fields below take a **distance**; **Apply** moves by exactly that amount, as many times as you like. With  **Object axes**, each object moves along its own axes: 100 mm in X means "100 mm in its own direction" for every track.

The arrow keys move in small steps,  in large steps, and  moves up and down.

See also

- [Rotating objects](#)
- [Scale elements](#), with the same anchor
- [The bottom-right display](#)

Rotate objects

 or  **Rotation** in the left toolbar opens rotate mode for the selection;  rotates around the object's own axes.  leaves it.

Drag or type

- In **2D**, a curved double arrow appears as the handle, with a cross at the center of rotation.
- In **3D**, there are three rings, one per axis: X red, Y green, Z blue.

The property panel shows the same rotations as numbers: in 2D the **angle**, in 3D **Heading**, **Pitch** and **Roll**. For a single object, these are its angles. For multiple objects, it is the amount to rotate by; the field then jumps back to 0.

The arrow keys rotate in steps of 1°, or 15° with ; see [Keyboard shortcuts](#).

Multiple objects: one center or each on its own

 Selection center	all rotate together around the center of the selection
 Individual centers	each rotates around its own center. The amount only takes effect with Apply

World or object

In 3D,  and  choose whether the rotation happens around the plan's fixed axes or around the object's axes. With the shared center, the plan's axes always apply.

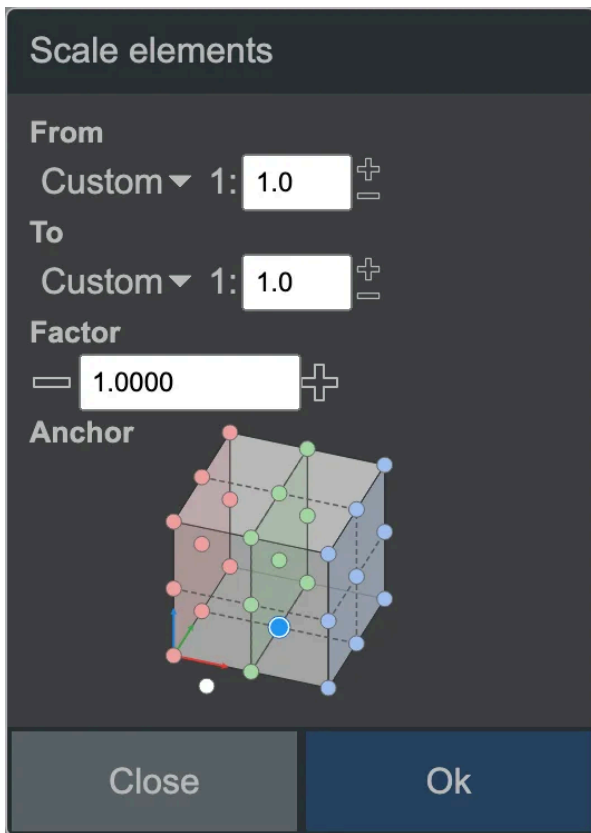
 **Reset** sets the orientation to 0; the position stays.

See also

- [Move objects](#)
- [Mirroring](#)

Scale elements

Scale... in the context menu enlarges or shrinks the selected elements by a factor.



The Scale dialog.

The factor

You can enter it directly, or have Raily work it out: **From** and **To** take two scales, and the **factor** between them follows. An H0 house (1:87) headed for an N-scale (1:160) plan is shrunk to 0.544 this way, without you doing the math.

That doesn't make it an N-scale house yet

The conversion shrinks the geometry. A model built for the smaller gauge would also have fewer details and thicker walls, so it can still be printed. For the footprint on the plan the conversion is correct; for 3D printing it is a starting point.

The anchor

The **anchor** is the point that stays put; everything else moves toward it or away from it. The default is **bottom center**: whatever stands on the table stays standing on the table. If the selection has a height, the picker offers all three axes; otherwise only the area.

If you grab the wrong point, nothing is lost: move the element to its place afterwards.

What can be scaled

The entry only appears if **every** selected element can be scaled. If it is missing, something in the selection has fixed dimensions.

Track does not scale

A piece of track is a catalog part with the manufacturer's dimensions. A plan in a different gauge is not scaled — it is built with the track of that gauge instead, see [Module selection](#).

Scaling a group

A [group](#) is scaled as a whole: as the parts grow, so do the **distances between them**. A station with its lamps keeps its proportions, instead of the lamps growing larger while staying where they were. The same applies to groups within groups.

See also

- [Rotate objects](#)
- [Groups](#)
- [Selecting objects](#)

Groups

A **group** combines several objects into one: a single click selects all of them, and they are moved or rotated together — for example a station building with its lamps, trees, and figures.

Forming and ungrouping

Edit → Group ( + )	turns the selection into a group
Edit → Ungroup ( +  + )	ungroups the selected groups again

After grouping, the new group is selected; after ungrouping, it is the objects that were inside it — so you can carry straight on working.

Groups can also be nested: a group can be combined with other objects into a larger one, as deep as you like.

Ungrouping restores the position, not the order in the plan

A group has its own position, and its members sit relative to it. When you ungroup it, Raily resolves that position back onto the objects, so they stay exactly where they were shown. The objects are then placed back at the spot where the group stood in the plan.

Editing a single object inside it

You do not have to ungroup a group to get at one of its members: you **enter** it.

1. Select the group.
2. Press .

While you are inside, clicks act on the group's members instead of the group. A note outlined in orange then appears at the top center of the plan, *"Editing inside another element — Esc or click here to leave"* — which is also the way out.

Way out	
	leaves the level
Click on the note	the same, without the keyboard
	the same

Each of these leaves **one** level; how deep you are is shown by the number in the note once there is more than one.

Not just groups

The same note and the same way out appear wherever one element contains others — for example in a solid that has had something subtracted from it, or in an assembled part.

See also

- [Selecting objects](#)
- [Rotate objects](#) and [Scale elements](#)
- [Keyboard shortcuts](#)

Drawing tools

At the top of the left toolbar are the tools for everything that is not track: roads, building footprints, water surfaces, walls, labels.

	Tool	What it's for
	New path	any free-form shape, angular or curved. The main tool
	New rectangle	rectangle, L, U, and O shape, with dimensions
	New circle segment	circle and arc
	Add text	a text box
	Measure elements	see Dimension lines and measuring
	New polyline	the tool from Raily 4, see Polyline

Drawing happens on the [work plane](#), in the colors of the two color swatches at the top of the toolbar. All these shapes are paths and share two things: a **height** turns the area into a solid, and two paths can be [combined](#), with the , , and  keys.

Path

Click sets a corner point. **Drag** while clicking instead sets a curve point with handles. A **double-click** ends the path.

The [second click](#) on a finished path lets you edit it: drag points and handles, and in the property panel

- turn a point into a **corner**, **smooth** it, or make it **symmetric**, or **delete** it,
- **split** a segment, **straighten** it, or turn it into a Bezier curve,
- **join** two endpoints, which closes an open path, or **separate** a point.

Below that are the **height** and the **fill**: the fill color, or in 3D a **material**.

The fill can be **hatched** or **crosshatched**: lines at an angle and spacing of your choice, in their own colour and width, drawn over the fill colour. Without a fill colour they give the look of a technical drawing.

Rectangle

Click one corner and drag to the opposite one. Releasing creates the rectangle and selects it; handles and properties come with the second click.

- **Shape**: **Rectangle**, **L shape**, **U shape**, **O shape**, or **Free path**. The first four are the same frame with legs on the sides of the box; switching keeps the box. The [baseboard](#) is the same shape with a thickness.
- **Corner radius**: only for the rectangle.
- **Free path**, or **Convert to paths** in the context menu for several at once, turns the shape into a path with individual points. There is no way back.

Text

The new text box hangs from the pointer; a click places it. It initially has neither border nor fill. While typing, a dedicated bar appears for font, formatting, color, alignment, and lists.

- **Size mode:** **Unlimited** grows with the text, **Fixed width** wraps, **Fixed size** holds both. Dragging a handle sets the matching mode.
- **Padding** and **border width:** the margin around the text and the box's outline.

Text does not appear in 3D by itself

The 3D view and the 3D export need **3d text (experimental)** in the advanced display settings.

See also

- [Track labels](#): the name on the track is not a text box
- [Flex track](#): the circle segment does not draw track
- [Material Selector](#)

Polyline

The **polyline** is the drawing tool from Raily 4: straight runs from point to point. For new work, use the **Path** in [Drawing tools](#); it can do the same, plus curves.

New polyline (): Click for each point, a double-click ends input. In the editor you drag the corner points, the **plus handles** between two points insert one,  removes the selected one.

Switch	
Close	joins the last and first point
Fill	fills the area. Only works on a closed shape
Height	turns the area into a solid

Convert to path in the context menu turns a polyline into a path.

See also

- [Drawing tools](#)

Image 2D

An image in the plan serves as a template: the floor plan of a room, a scanned sketch. It comes in via **File** → **Import...** or by dragging the file onto the window, see [Import files](#).

Selected, it shows handles for scaling. In the editor ([second click](#)), width and height are available as numbers, a button swaps the image, and  **Restore aspect ratio** un-skews it again.

Scale to a known distance

1.  **Start measuring** and click two points in the image whose distance you know, such as two corners of a room.
2. Enter the real distance as the target length.
3. Apply the scaling.

The image is then to scale. It's best to place it on its own, locked [layer](#).

See also

- [Dimensions and measuring](#)
- [Baseboard](#)

Dimensions and measuring

A **dimension line** is an element like any other: movable, adjustable, deletable. There are two ways to create one.

By hand

 **Measure elements**, among the drawing tools in the left toolbar: click the start point, click the end point. If you hit a track or another element while doing so, the point binds to it, and the dimension line moves along with it.

Turn a length into an angle

Open the dimension line's properties ([second click](#)) and set **Measurement** to **Angle**. A third handle appears in the middle; drag it to the vertex. **Show radius** adds the radius.

For a whole track

Measure in the context menu of selected tracks creates a dimension line for every straight segment (length) and every curved one (angle and radius). Very short pieces are left out, as is the freely curved path of a [flex track](#): it has no radius.

The dimension lines are anchored to the track. Move, rotate, or bend it, and they move along.

The entry only appears when the selection consists exclusively of tracks.

Adjust

The context menu of a dimension line flips it with **Switch dimension side**. Text size, arrows, and alignment are in its properties. The **Annotation size factor** in the [Plan Properties](#) enlarges all dimension lines together.

See also

- [Image 2D](#): scale an image to a known distance
- [Track labeling](#)

Baseboard

Insert → **Table...** creates the board the layout sits on. For a plan meant to be built later, it is the first item: it tells you how much space is really there.

Four shapes

At the top of the dialog, four **shapes** are available: Rectangle, L shape, U shape, and O shape. They are the same as for the [Rectangle](#) drawing tool.

The dimensions

The **box** — the length and width everything fits inside — comes from the frame you drag out in the plan, or change at the handles. Within this box, the remaining fields describe the individual **runs**:

Field	Meaning
Left run, Right run	Width of the two side runs
Bottom run, Top run	Depth of the two cross runs
Left leg length, Right leg length	how far a side run extends, only for the U shape
Height	the thickness of the board

Changing it later

The baseboard stays adjustable for as long as it exists: select it, [click it again](#), and the same fields appear again. The shape can be changed here just as it can when creating it.

The baseboard is a path

That is why it can do what paths can: carry a line color and a fill color, sit on its own [layer](#), and be included when [printing](#) or not. The **height** turns the outline into a solid.

See also

- [Image 2D](#): place a floor plan of the room in the background and align the baseboard to it afterwards
- [Work plane and coordinates](#)
- [Layers](#)

Installing into multiple hosts

[Getting started 7](#) shows  +  for a window and a building. The same key goes further, in both directions.

Anything with a solid can be installed. Whether it cuts an opening while doing so depends on the element: a window or a tunnel cuts, an embankment does not.

Multiple elements into one host

Select the elements, add the host last with -click, and press  + . Raily cuts all the openings at once.

One element into multiple hosts

The same cutout should appear on several hosts, for instance on several identically aligned wall segments. Select the element first, then the hosts, and choose **Install into selected hosts** from the context menu.

Only the first host gets the visible solid

All the others only get the cutout. The command matches one opening across several hosts — it does not duplicate an object.

Move the element afterwards, and all the cutouts follow.

If only one of the selected elements can cut at all, Raily finds it even if you did not click it first. Otherwise the order matters.

Uninstalling

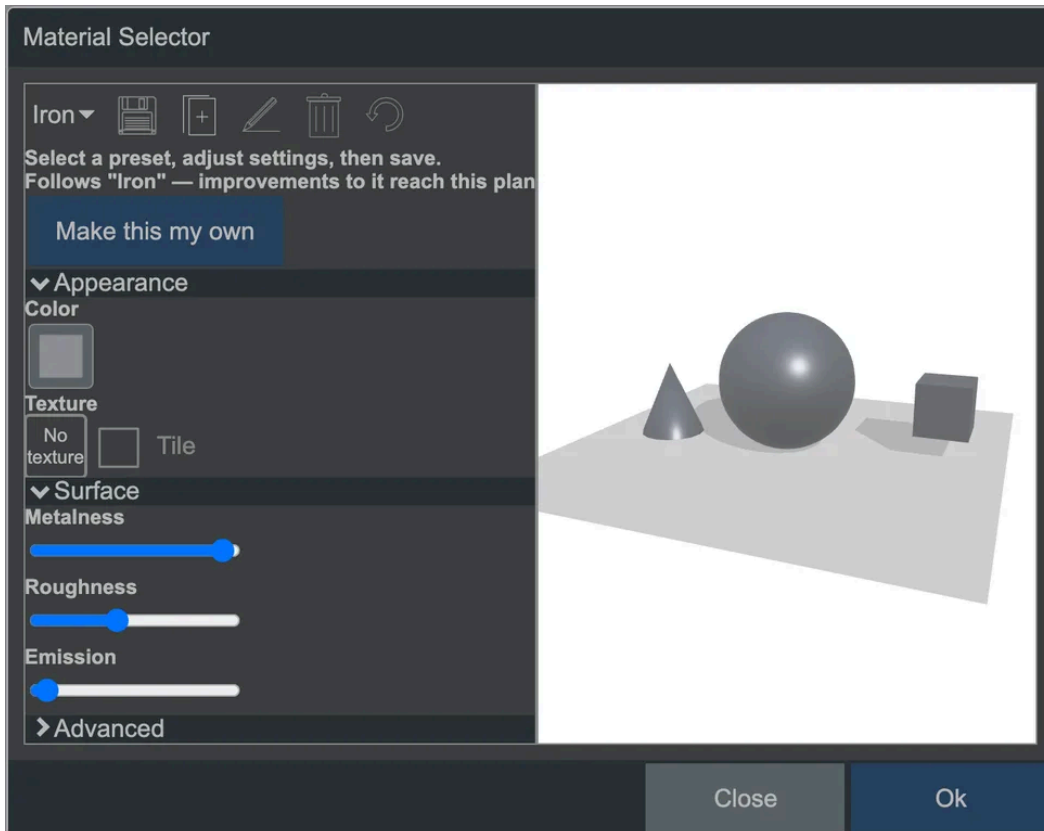
 +  or Take out	releases the one selected relationship; other cutouts remain
Uninstall from all hosts	releases the whole assembly, from any one of its parts. Appears only with more than one host

See also

- [Solid operations \(CSG\)](#)
- [Selecting objects](#)

Material selector

The material selector determines how a surface looks: color, finish and, where needed, a texture. It appears wherever an element has a material — on buildings, terrain surfaces, building parts and imported models.



The material selector dialog.

Presets

The fastest way there is a **preset**: polished steel, nickel silver, iron, cast iron and other ready-made materials. The approach is **select a preset, adjust settings, then save**.

Appearance

- **Color** and **texture**, the image on the surface.
- **Tile size in real-world metres (not model scale)**: how large the texture is in reality, set separately for **U (X)** and **V (Y)**. The value is a length, not a unitless number: a brick wall at 1 m per repeat looks right at any scale.
- **Tile**: whether the texture repeats or is laid down once.

Textures Raily draws itself

The texture dialog doesn't only list images. **New generated...** builds a texture from its description, and **Adjust...** takes a generated one as a starting point.

Texture	
Brick	Stretcher, English, cross and Flemish bond, new or weathered, brick and mortar color
Roof	Plain tile, pantile or slate
Stone	Ashlar, setts, granite, sandstone, marble
Wood	Sawn boards with width and grain
Corrugated sheet	Corrugated or box profile, true to its real pitch
Surface	Concrete, precast, asphalt, sand, earth, roofing felt
Rust, Patina	the two weathering effects

All of them compute from **real-world dimensions**: you specify how wide a brick is, not how many pixels it has. That way the same texture fits at any scale, and the **preview** shows it singly or **tiled**.

Adjust creates a new texture

A generated texture is its description: change a value and it becomes a different texture, and the old one stays wherever it's already in use. So you never modify a model just because it happens to use the same texture.

Show legacy

The **Show legacy** switch brings in the images from Raily 4. For a new plan, it's worth looking at the generated ones first: they're sharper, because they're drawn at the size they're needed at.

Surface

- **Metalness**: how strongly the surface reflects like metal. High for rail profiles, zero for masonry.
- **Roughness**: how matte the reflection is. Low gives a mirror gloss, high a dull material.
- **Emission**: how strongly the surface glows on its own. For a lit window or a signal; not a substitute for a lamp, since a glowing surface doesn't light up anything but itself.

Special

A few materials follow their own rendering instead of the normal surface calculation:

Special	Effect
None	Normal material
Water	Water surface with its own reflection
Crystal (fallback)	Translucent material

Under **Special Parameters** are the values each special material needs.

Preview

The preview shows the material either on the element itself (**Element preview**) or on a neutral standard shape (**Standard preview**). **Drag to rotate, scroll to zoom.**

See also

- [Color selector](#)
- [Rail display](#)

Color selector

The dialog appears everywhere a color is chosen. It has four expandable routes to a color:

- the **Web palette**,
- **Custom** with sliders for RGB and HSV, and for opacity where that fits,
- **Bookmarked colors**, to keep a plan consistent,
- **Used colors** of the open plan.

Wherever a color may also be absent, such as the fill color, it can be cleared here.

See also

- [Material selector](#)

Advanced tools

What else there is, and what it's for — without step-by-step instructions.

Module editor

Tools → **Module editor...** ( + ) edits track systems instead of plans: create your own, or change a bundled one.

Create Module sets up an empty module, **Select Module** loads an existing one. **Add New Rail** builds an element from a basic shape (straight, curve, switch, crossing, turntable, bumper block ...) complete with segments and connectors; for more freedom, use individual segments and connectors from the editor's toolbar. **Make Element Drivable** turns the geometry into running paths — for a switch, with several named positions.

Draw Polygon adds decoration such as a platform, a pit or a buffer: click its corners and double-click to finish. A new polygon is closed and filled, like nearly all of the catalogue's. The stroke and fill colour buttons in the toolbar set the colours of the selection and of what you draw next; the tool window sets whether a polygon is closed and filled, and a double-click on it lets you move its corners.

New Path and **New Rectangle** draw decoration a polygon cannot: curves, holes and hatched areas, set in the tool window. Raily 4 does not show such decoration.

New Regular Polygon... draws a circle or an arc as decoration: it asks for the radius, the arc and the number of corners, and a click places the polygon. With 360° the corners are spaced evenly; a smaller arc puts a corner on each of its ends. The dialog remembers the last values.

Mirror horizontally and **Mirror vertically** flip the selection, or the whole element when nothing is selected. **Create Mirrored Counterpart...**, next to the element list, adds a copy of the current element mirrored lengthwise under a new ID: a left switch becomes a right one, decoration included. Mirroring a plan then uses it. **Mirror image** in the element details names the mirror image by hand: another element of the module, or the element itself, turned.

Next to the module name sit the buttons for the metadata (manufacturer, scale, system, catalog status ...), the **Track Profile**, the **Module Diff** against the saved version, and the **download** as `.arf`.

A change applies to every plan that uses the module from now on

A module belongs to its name, not to a plan. Plans already saved keep their own version, and Raily asks when opening which one should apply (see below). **Delete Element** removes the track piece from the module; the dialog tells you what that means for plans.

A module loaded into the editor is only in your library after you **save** it. Until then you edit a copy: **Discard** leaves the module as it was.

A note on the module says where it comes from. For a module from your library, next to it sit **Earlier versions...**, which brings back one of the last ten saved versions (the current one goes to the trash in

turn), and the way out: **Use Raily's version...** for a module that replaces a bundled one, otherwise **Remove from my library....**

Module library and reconciliation

If a plan brings along a module that differs from your library, **Module in this plan** appears.

- If you don't have it: **Use for this plan only** or **Add to my library**.
- If you have your own version, the dialog shows the differences and offers **Keep both**, **Replace my library version with the plan version**, or **Use my library version**.

Replacing can remove elements from your library

The dialog names them and asks for a second confirmation. **Keep both** changes nothing and is the default.

Raily remembers your answer for exactly these two versions: the plan asks again only once one of them has changed. **Use my library version** reaches the plan file with the next save.

The way back to the bundled version is described in [Module selection](#).

Object repository

Insert → Object repository... offers ready-made objects from bundled collections: stations, trees, bridges, buildings, signals. If the scale doesn't match, **From scale** and **To scale** work out the **factor**.

The geometry is scaled, not the level of detail

After conversion, an H0 house has the size of an N-scale model, but not its wall thicknesses.

Primitive shapes and solid operations (CSG)

Insert → Scenery and life → Add 3D shape... places a box, sphere, cylinder, cone, pyramid, torus and more. They become a custom part once you combine several of them: a wall with a niche, a pipe with a bore.

Select two or more solids. The **CSG** context menu and three keys do the same thing:

Command	Key	Result
Subtract	-	A without B
Union	+	A and B as one solid
Intersect	/	what A and B have in common

B is whatever you clicked last. Holding **Alt** also leaves B standing. The calculation runs in the background; the status bar shows it.

The result remembers what it's made of. **Mesh** → **Create mesh** in the context menu turns it into a solid triangle mesh, for example for the [3D export](#). The same keys also combine two [drawn paths](#).

Editing the parts of a body

Click a combined solid a second time and the **Parts** strip opens: a tree of every part, nested bodies included, with a sign in front of a part that says what it does — a plain minus for one that is cut away, nothing for one that is added. Choose a part in the tree, in the flat views (**Ortho views**) or in the scene, and the editor stands on that part: move and rotate it with **T** and **R**, drag it, delete it, or change its values in the properties panel. Something inserted meanwhile becomes a part of the body. The strip's arrows change the order of the parts, which is what decides what is cut from what; **Take out** (**Shift** + **O**) lifts a part out of the body it sits in. **Esc** closes the strip.

The element groups

At the bottom of the left toolbar, and in the **Insert** menu, sit the same four groups: **Railway**, **Buildings**, **Building parts** and **Scenery and life**. What's in each one is shown by the list itself. Two rules decide where an element ends up:

- Almost everything under **Building parts** attaches to a **surface**, as does a lamp on a wall bracket.
- **Pole** and **signal** bind to a **track**.

Both are explained by [Snapping](#). Every element is adjusted the same way; see [Shaping a building](#).

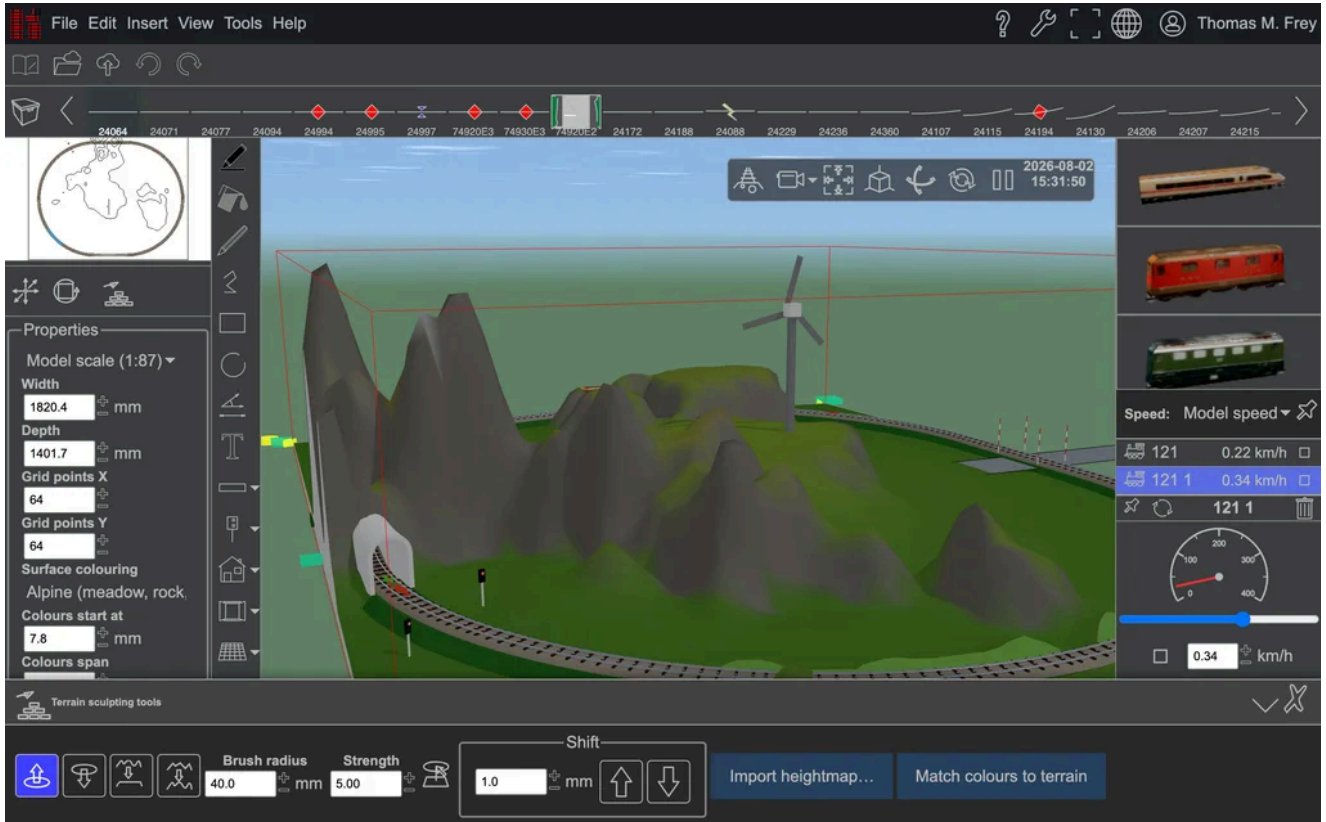
See also

- [Module selection](#)
- [Getting started 7: Decorating](#)

Scenery

Terrain editing (Terrain & Sculptor)

How a mountain comes about is shown by [Getting started 5](#): place a patch, [click it again](#), shape it with the four brushes. What follows here goes beyond that.



The sculptor on a terrain patch, in the 3D view.

The patch

Grid points X and **Y** are the patch's most important value. The brushes only move heights that exist as support points: a patch that's too coarse can't be shaped finely, one that's too fine costs computing time for detail nobody sees. Start coarse and create finer patches where you need detail.

Alongside that are **Width**, **Depth**, and one material each for the top, the sides, and the bottom.

Brushes

- **Aim by ground plane** helps on steep slopes, where the flank is in the way and you mean the foot of it. The usual choice is **Aim by surface**, aiming by surface picking.
- **Shift** raises or lowers the **whole patch** by an amount, without touching its shape: the fastest way to bring a finished terrain up to the height of the track.
- Brush radius and shift are lengths and follow **Model scale for lengths and sizes**. **Strength** is a proportion.

Surface colouring

Surface colouring colors the surface by height and steepness, instead of the material: **Alpine**, **Meadow**, **Moorland**, **Desert**, **Autumn**, or **None**. **Colours start at** and **Colours span** set the height range the palette runs across.

Match colours to terrain in the tool window sets both from the actual heights. That way the snow ends up on the peak.

With multiple patches, select them all together

The button sets **one** height range for all selected patches. Adjusted one at a time, a color band ends up cutting straight across continuous terrain.

Contour lines

The patch can draw contour lines, on request, **in the plan** and **in 3D**, with adjustable spacing, start height and color. On paper they replace the 3D view.

Contour lines know nothing of cutouts

They show the slope as it would run **without** a cutting, and in 3D they run straight across the open pit. For a tunnel that's correct, since the mountain above it is still there.

Multiple patches

Split terrain... in the context menu cuts a patch into blocks, **Merge terrain blocks** joins them back together. This gets you different resolutions in different places, and blocks that individually fit your printer.

A second click always opens only **one** block, and a brush stroke stops at the seam. So select all the blocks and call up **Sculpt terrain blocks together...**: the brush then runs across the seams.

Merging is only offered when the blocks form a rectangle and nothing built in would be lost, such as a tunnel.

Terrain last

Settle the track and its heights first ([Embankments and ramps](#)) and only then adjust the terrain. Tunnels and cuttings from [annotations](#) build themselves in on refresh.

See also

- [Import a heightmap](#)
- [Create tunnel](#)
- [Performance tips](#)

Import Heightmap

A heightmap is an image whose brightness is read as height: **light becomes high, dark low**. **Import Heightmap...** in the [terrain editing](#) tool window fills the selected terrain patch from it.

Where the image comes from

- **Choose heightmap image...** loads an image from your computer. Besides **Greyscale**, the dialog reads the two common color encodings for real elevation data, **Mapbox Terrain-RGB** and **Mapzen Terrarium**.
- **Fetch from elevation data...** cuts out a piece of real terrain: choose a place, set the extent. This needs you to be [logged in](#).

The dimensions

Field	Meaning
Peak height	for greyscale: what height white corresponds to
Ground width of the image	how wide the depicted terrain really is
Terrain width in the model	how wide it should become in the plan. Match plan scale calculates this
Vertical exaggeration ×	stretches the heights. Real terrain often looks too flat in the model
Grid resolution X / Y	the fineness of the patch. Match image resolution sets it to fit

Normalize sets the lowest point to zero, **Set boundary heights to zero** lets the terrain taper off at the edges. The preview shows the result before you apply it.

Image material

Compression artefacts in a JPG turn into bumps in the terrain; use PNG. An image that only uses mid grey tones results in flat terrain.

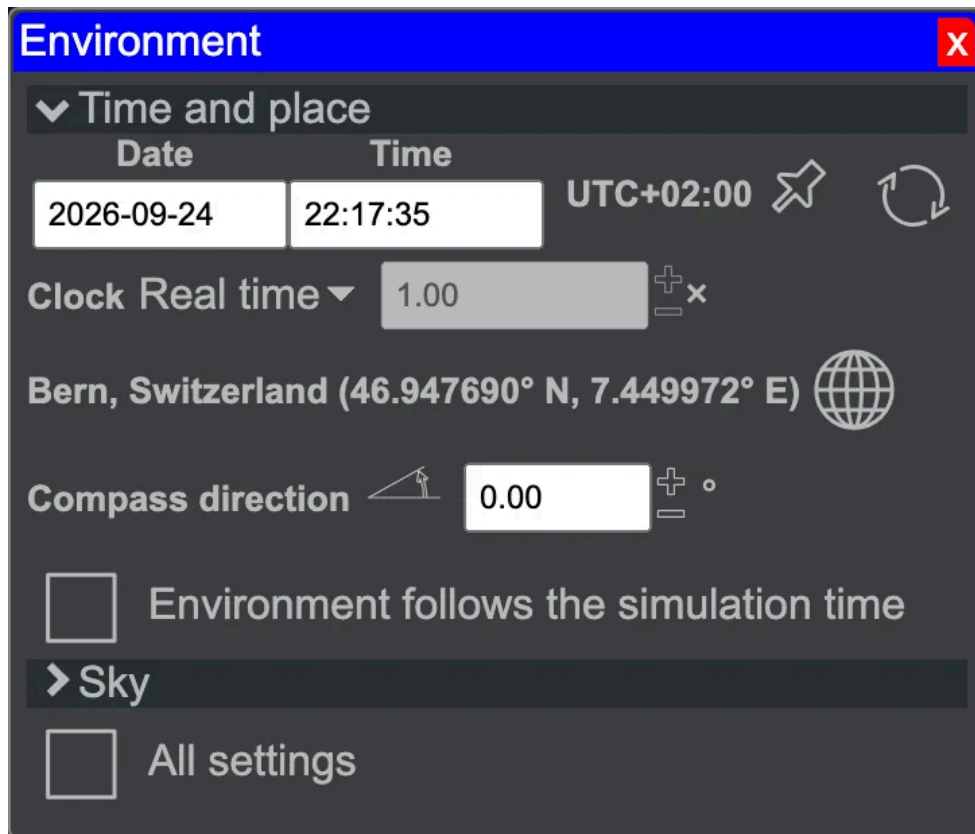
The result is a starting point: afterwards you cut in the trackbed with the brushes or a [cut & fill annotation](#).

See also

- [Terrain editing \(terrain & sculptor\)](#)

Environment

Tools → Environment... (**Shift** + **E**) determines **when and where** your layout stands and what the sky above it looks like. The settings belong to the plan. The window stays open, and every change shows up immediately in the 3D view.



The Environment tool window.

Usually the two sections **Time and place** and **Sky** are enough. **All settings** at the very bottom unfolds the rest: sun and moon light, ambient light, fog, exposure.

Time and place

The decisive question sits at the very bottom: **Environment follows the simulation time**. When it is set, sun, moon and sky follow from **Date**, **Time** and location. When it is off, you set the sun's position by hand.

- The **Clock** runs in **Real time**, stands still **Fixed**, or runs **Custom** faster or slower.
 - **Select location...** offers cities; otherwise you enter coordinates. **Compass direction** sets where north lies on your layout, and with it, from which side the sun comes.
 - ⚓ stores the clock as it stands now as the plan's start, running or not; ↺ resets it back to that.
- They are the same buttons as for the [start point](#) of a train or a walker.

Pick a Location

Search

Country

Type to filter by name

United States

Brooklyn 40.650° N, 73.950° W · 15 m

Los Angeles 34.052° N, 118.244° W · 89 m

New York City 40.714° N, 74.006° W · 10 m

Coordinates

lat,lon OR 47.37 N, 8.63 E

Name

Location name

Height above sea level (m)

0

Close

Ok

The location picker, opened from **Select location...**

The clock shows your time zone, the sky shows the location

Someone planning a Swiss layout from Canada sees the right sun at an hour that looks wrong.

Sky

Background decides how much the section offers at all: **Plain background**, **Background follows the sun** (the same color, darkening with the time of day), or **Sky with sun, moon and stars**. Only the computed sky brings stars, clouds with a wind direction, and the **endless flat ground**, so the layout doesn't float in nothing. The clouds move with the clock.

What the sun's position switches

From the sun's altitude, Raily derives five conditions that apply throughout the plan:

Condition	When it applies
day	the sun stands above 10°
night	it stands below −6°
twilight	in between
dawn / dusk	the same span while the sun is rising or setting

A lamp with `night|dusk` then turns on at sunset. How a condition is written is described under [Signals, barriers and blocks](#). If the sun stands still, use `twilight : dawn` and `dusk` only exist with a sun that moves.

Too dark or too bright?

Check in order: does the environment follow the simulation time, and is it currently night there? Is the sun's position set very low by hand? Is **Exposure** off, or **Ambient light** set to zero?

See also

- [Getting started 7: Decorating](#): the lamps that wait for `night`
- [Plan properties](#): the background color

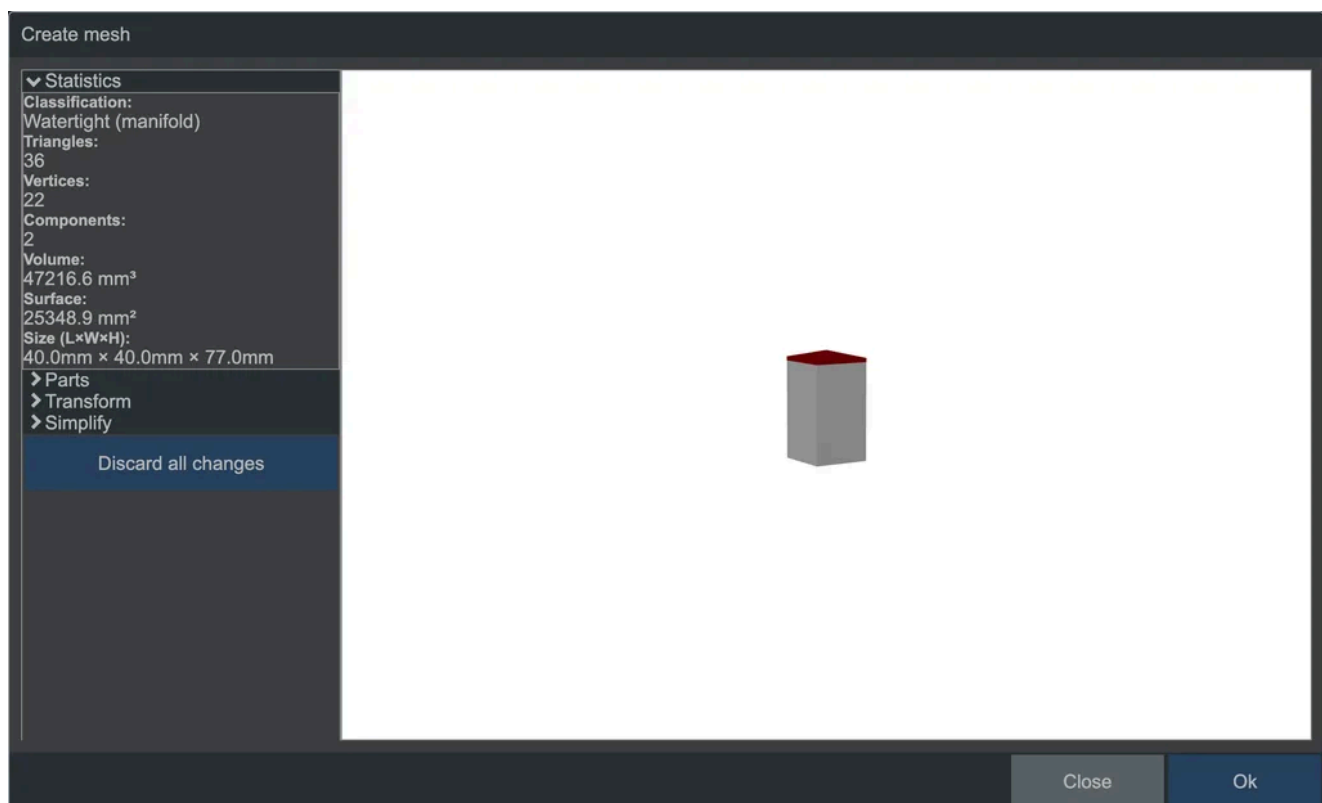
Import, export and printing

Importing files

📁 **File** → **Import...**, or drag the file onto the window. Raily recognizes the format from the extension. Drag files that belong together at the same time, such as an `.obj` with its `.mtl`, otherwise the model stays without color.

Format	Comes in as
STL	a solid, after a preview dialog
3MF, glTF/GLB	a solid, after a dialog for scale and up axis; lamps and cameras come along
OBJ, 3DS	a solid, directly in the plan
DXF, HPGL, SVG	lines, arcs, polylines and areas
PNG, JPG, GIF, WebP	a 2D image at the pointer
R16	a texture from Raily 4
.arf, .rly	a plan or module, see below

Checking 3D models



The dialog when importing a mesh file.

Two things are often wrong in models from other sources:

- **The unit.** STL stores none. Compare the displayed **Dimensions** with what you expect: a house 120,000 mm tall was saved in meters. Models for 3D printing are often 1:1 and need to be scaled down to your layout scale.
- **The up axis.** If the model lies on its side, rotate it 90° about X.

The **number of triangles** tells you what the model will cost to render. A model for 3D printing can have more triangles than an entire track layout; **Simplify** reduces them. The same dialog repairs holes and duplicate faces, also afterwards via **Process mesh...** in the imported model's context menu.

Plans and modules

An `.arf` file with a plan asks "**Replace old plan?**": **Replace**, **Insert** (the elements hang from the pointer) or **Open in new tab**. A plan from Raily 1 to 3 (`.rly`) can only be replaced or opened in a new tab.

If a module is in the file, Raily asks "**Install module?**": **Install** puts it permanently in your library, **This session only** does not. If you already know the module, Raily first shows which elements were added and which were dropped.

See also

- [Import a heightmap](#): terrain from an image
- [3D export \(STL, 3MF, glTF\)](#)
- [Performance tips](#)

Printing

File → Print... ( + ) prints the plan to scale, spread over several pages if needed, which can then be assembled. What gets printed is whatever sits on visible [layers](#).

- **Paper Options:** format, orientation, and margins.
- **Scale 1:n:** enter the factor, for example 1 for a full-size template.  **Fit on page** picks a round value that fits the whole plan onto one page. The dialog shows how many pages that comes to.
- **Info box:** a title block with plan, author, and description from the [plan properties](#) and the scale of this print, in one corner of the assembled plan. Empty fields are left out.
-  **Export (HTML)** writes the same pages to a file that can be printed from any browser.
-  **Save as PDF** writes the same pages to a PDF: at exactly the scale set here and without the header and footer a browser adds when printing. Print the PDF at actual size (100 %), not "fit to page".

The readable blueprint

The **Print** preset of the [display overrides](#) shows center lines and short rail names instead of bedding and sleepers. Hairlines become very fine on a printer; specify a line width for printing.

Without a license, every sheet carries a notice, see [License](#).

See also

- [Export SVG](#)
- [Inventory](#)

Export SVG

The SVG export produces a 2D vector drawing of your plan, suitable for further work in a drawing program, for printing at any size, or for publishing on the web.

What gets exported is what is visible in the 2D view: hidden layers do not appear in the file.

Options

- **Include zoom/pan script:** places a small script in the SVG file that lets the drawing be zoomed and panned in the browser. For an image someone is meant to explore on screen; unnecessary for printing.
- **Wrap in HTML:** writes a complete HTML page instead of a plain SVG file. Useful together with the zoom/pan script.
- **Precision (digits):** number of decimal places for the coordinates. Fewer digits give a noticeably smaller file; 2 to 3 is usually enough for a model railway plan.
- **Absolute coordinates:** writes every point as an absolute coordinate instead of relative to the previous one. Increases the file size, but helps with target programs that interpret relative paths imprecisely.
- **Bake in transform:** folds transformations (translation, rotation, scaling) directly into the coordinates instead of leaving them as a `transform` attribute. Needed for programs that do not resolve nested transformations correctly.

When the result looks wrong in the target program

Enable "Bake in transform" and "Absolute coordinates". That removes almost everything that makes different SVG interpretations diverge.

Notes

Fonts and fill patterns may differ depending on the target program: SVG references fonts instead of embedding them. If text reflows or the typeface changes, replace it in the target program or convert it to paths there.

See also

- [Print dialog](#)
- [Export to Raily 4](#)

3D export (STL, 3MF, glTF)

Format	What it's for
STL	3D printing. Pure geometry, no colors
3MF	3D printing with colors
glTF	Display and further processing, for example in Blender

All three are in a solid's context menu under **Export**; glTF exports the selection from there, and the whole plan under **File ▶ Export ▶ glTF**.

Check the dimensions in the dialog before you print.

STL



The STL export dialog.

Binary – no color is the normal case. The two color variants are two incompatible interpretations of an unofficial extension; you will have to try which one your slicer understands. For color, 3MF is the better route.

3MF: multi-color printing



The 3MF export dialog.

A slicer does not assign filaments to a triangle's color, but to **parts** of an object or to a painted surface. That is why the **color mode** decides whether it sees your colors:

Color mode	
Standard 3MF materials	the standard, and the default. Viewers show the colors, Bambu Studio and OrcaSlicer do not read them
One filament per part	each solid becomes a part with its own filament. The most reliable way, when each color is its own solid
Painted per triangle	the color sits as paint on the surface. Needed when one solid carries several materials

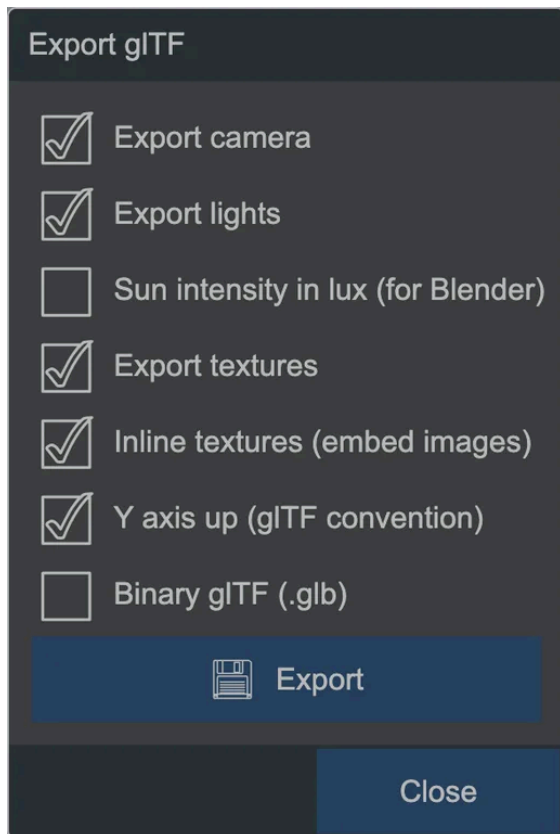
How to tell you have the right mode

If **Mixed-material triangles** shows 0 and each color is its own solid, use **One filament per part**. Otherwise, **Painted per triangle**.

Distinct colors (filaments) counts how many filaments the print needs.

Before exporting, Raily repairs the geometry if needed. If that required adding or removing surface, it shows a message — take a look at the result before you print.

glTF



The glTF export dialog.

You choose whether the camera, lights, and textures go along, and whether everything sits in one `.glb` file. Leave **Y axis up (glTF convention)** on, or the model will lie on its side in the target program. **Sun intensity in lux** is meant for Blender.

See also

- [Import STL](#)
- [Solid operations \(CSG\)](#)

Export DXF and HPGL

Both formats are under **File** → **Export**. DXF delivers your plan as a drawing for CAD programs, HPGL for a cutting or drawing plotter.

Both export line work only: text (labels, dimension-line numbers) and filled areas are in neither file, only outlines. For a drawing with labels, the [SVG export](#) is better suited.

Hidden layers count too

Unlike the [SVG export](#), the DXF and HPGL exports do not check which layers are currently visible: the whole plan always goes out, hidden layers included. Hiding something therefore does not keep it out of the file; for that you actually have to delete it from the plan. Both exports also only know the whole plan; there is no export of just the current selection for DXF or HPGL.

DXF

File → **Export** → **DXF...** opens a dialog with six checkboxes before the file is downloaded:

- **Header+Tables** (on by default), also writes the DXF header data (`HEADER`), the layer table (`TABLES`) and the file terminator (`EOF`). Switched off, the export delivers only the plain drawing elements, without header, tables, or terminator — a fragment for tools that insert it into an existing drawing rather than expecting a standalone file.
- **TrueColor** (on), writes each element's actual RGB color instead of relying on the classic 16-color palette. If your target program does not read TrueColor, only black and white remains, so switch it off.
- **LineWeight** (on), writes each element's stroke width into the file (in 1/100 mm, 0 to 2.11 mm).
- **Triangles (3DFACE)** (on), exports areas as `3DFACE` entities. Switched off, the export leaves out all areas; only lines and polylines remain.
- **Prefer LWPOLYLINE (2D)** (on), writes a consistently flat polyline as a compact, purely two-dimensional `LWPOLYLINE` instead of a classic `POLYLINE` with its own entry per point. This only affects polylines that are flat anyway; a genuinely spatial polyline always stays a `POLYLINE`, regardless of this setting.
- **Force 2D (z=0)** (off), sets the height of every point to 0, even for spatial geometry. For a pure floor plan, or when the target program does not know a third dimension.

For a cutting plotter or a pure 2D program

Switch on **Force 2D** and leave **Prefer LWPOLYLINE** on. The result is a flat, compact drawing without height values. With **Force 2D** off, a spatial layout instead turns into a DXF file that contains heights a 2D tool does not expect.

HPGL

File → **Export** → **HPGL...** has no options; a click on **Export** is enough.

HPGL is the language of pen and cutting plotters. A cutting plotter uses it to cut your track plan as a stencil out of cardboard, film, or thin wood.

Circles and arcs also turn into lines: HPGL has no third dimension, the plan is output flat. The file uses a single pen throughout; the colors in your plan play no role for the export. Very large plans are automatically scaled down by the export as far as needed to fit the coordinate range of the format (around 819 mm per axis at full resolution) — there is no setting for a specific paper or workpiece size. So check the actual size on your plotter before you cut.

See also

- [Export SVG](#)
- [3D export \(STL, 3MF, glTF\)](#)
- [Import files](#)

Export to Raily 4

Produces a file that Raily 4 can open, for exchanging with someone still using the previous version.

The export loses information

Raily 4 does not know all of Raily 5's element types and properties. The dialog points this out explicitly when it opens: whatever Raily 4 cannot represent is simplified or left out.

How to go about it

1. **Save your Raily 5 plan first.** The export is an additional file, not a replacement — keep working with the original file in Raily 5.
2. Export.

Elements that did not exist in Raily 4 are affected most: parametric objects, extrusions, terrain and CSG results.

See also

- [Export SVG](#): a lossless alternative when the recipient only needs a drawing

Inventory

View → Inventory... counts which rails your plan needs, and checks that against what you already have.

Inventory

Element ↑	Module	Preview	Used	Available	Remaining	Price
14534R Roco HO	HO Roco (1960-1980)		1	0	-1	0.00
4400 Roco HO	HO Roco (1960-1980)		9	0	-9	0.00
4404 Roco HO	HO Roco (1960-1980)		29	0	-29	0.00
4408 Roco HO	HO Roco (1960-1980)		1	0	-1	0.00
4417 Roco HO	HO Roco (1960-1980)		1	0	-1	0.00
4421 Roco HO	HO Roco (1960-1980)		12	0	-12	0.00

total length = 13893.1mm
inventory value = 0.00, missing elements price = 0.00

Close

The inventory dialog.

For each rail it shows how many times it is **used**. You fill in **Available** and **Price** yourself; **Remaining** and the totals in the footer are calculated from them: total length, inventory value, price of the missing elements.

Your stock is not part of the plan; it applies to all plans. Raily remembers it on this device and, when you are logged in, in your account: the browser and the desktop app then show the same stock, and the one changed last wins. You can also keep it as a file:

Button	
Save inventory as JSON / Load inventory from JSON	Your stock with prices. Loading replaces the current values
Import Raily 2/3/4 inventory	an <code>.inv</code> file
Print	the list as a printout or an HTML file
Export inventory data	the table as CSV or JSON, as a file or to the clipboard
Settings	show unused stock, show prices, filename and date on the printout

If prices are hidden, they are also left out of the printout and the export. In the CSV, numbers always use a decimal point; for a spreadsheet that expects a decimal comma, use semicolon or tab as the separator.

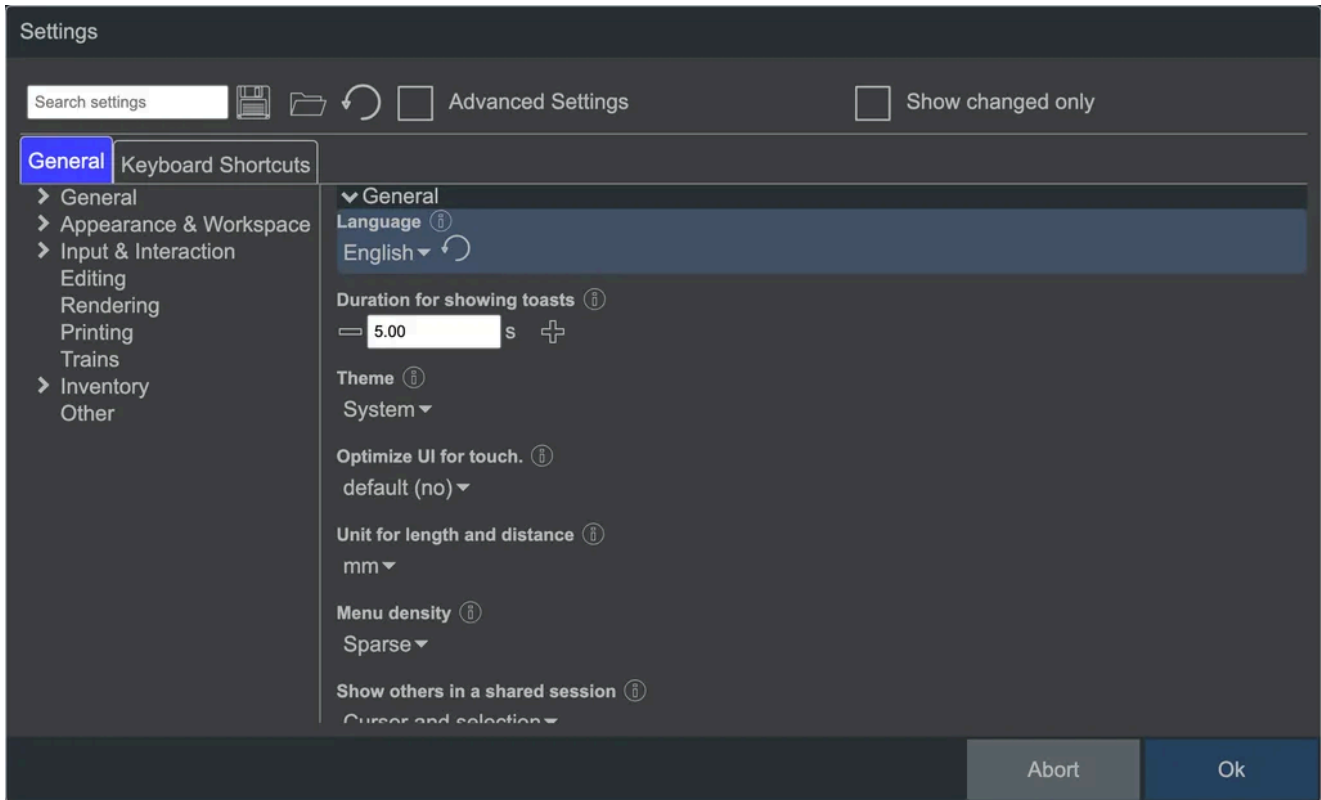
See also

- [Element info](#)
- [Module selection](#)

Setup and management

Settings

Settings are grouped by area, and each one carries its own explanation. This page therefore does not describe them one by one. It answers the question that, when in doubt, actually gets asked: **How do I get back if I have changed something?**



The Settings dialog.

What have I actually changed?

Show changed only hides everything still at its default value. What remains is exactly what you have touched at some point — usually a short list, even if the dialog feels large. Changed settings are also highlighted.

Resetting a single setting

Reset to default appears next to a changed setting. The button is only visible where the value differs from the default; it is therefore also the indicator that something was changed here.

This is the normal case: first turn on **Show changed only**, then reset the one row causing the trouble.

Resetting everything

If that does not help, calling

```
.../raily.html#reset-config
```

resets **all** settings. Raily asks at startup and only deletes them after your confirmation.

This affects only settings

Your plans remain untouched: they live in files, not in the settings. What is lost is every customization: language, appearance, keyboard shortcuts, defaults.

Transferring and backing up

Export settings writes everything to a file, **Import settings** reads it back in. This is the way to a second computer, and the precaution before you change something whose effect you do not yet know: export first, then try it out.

See also

- [Keyboard shortcuts](#)
- [Backup and recovery](#)

Plan Properties

File → Plan Properties... collects what applies to the whole plan: its scale, its look in the background, the rule for curves that are too tight, and the details by which you'll recognize it later.

Field	What it does
Scale	the ratio of prototype to model, e.g. 87 for H0 (1:87). It affects every length given in the plan
Smallest curve radius	how tightly a flex track may be bent; tighter spots are marked red in the plan. 0 turns the warning off
Annotation size factor	enlarges dimension lines and the compass together, without changing the plan
Background color	the color behind the plan. In the 3D view it can be replaced by the sky
Author and Description	free text, for sharing and documentation; the description accepts formatted text

Set the scale first

It decides what an entered length means, and the default smallest curve radius is calculated from it. Set it when you create the plan, not once track is already laid. The value you leave the dialog with also becomes the default for your next new plan.

The smallest curve radius is a default, not a lock

Raily does not stop you from building tighter; it only tells you where. The default is 762 mm for H0, the same radius converted to scale in other scales. It is measured on freely-shaped flex track; a catalogue curve has the radius it was manufactured with. A single piece of flex track may deviate from it; see [Flex track](#).

Time, location, sky, light and fog are **not** here — they live in the [Environment](#) tool window.

See also

- [Flex track](#): where the smallest radius matters
- [Environment](#)
- [Work plane and coordinate systems](#)

Backup and recovery

Raily backs up every open plan in the background, even one that has never been saved: at most every 30 seconds, and only when something has changed. A floppy disk briefly blinks in the status bar with *"Backup"*.

A backup is not a save: the plan file stays as it was, and **Unsaved changes** remains shown.

Every backup from the last ten minutes is kept; after that it's thinned out: first one every five minutes, after a day one every four hours, after a month one per week, after a year one per month.

Where they live

	Location
Browser	in the browser's own storage, separate per browser and profile. <i>Clear browsing data</i> removes them too
Desktop app	in the app data folder. Help → About... (macOS: Raily → About...) shows the path under <i>"Where your files are kept"</i> and opens the folder

On startup

If a plan has a backup newer than its last save, Raily reports *"Recover Unsaved Work"*: after a crash, a power outage, a closed tab.

The arrow in the row restores the **newest version**; clicking the plan shows the older ones. The recovered plan shows **Unsaved changes** until you save it, and no backup is deleted in the process. **Close** leaves everything as it is; the prompt comes back at the next start.

By hand

In the **Help** menu, right at the bottom:

Entry	Shows
Versions of This Plan...	the versions of the open plan. Greyed out for a plan that was never saved
All Backed-Up Plans...	every plan with backups, including ones never saved and ones from earlier sessions

Restoring opens the version in a new window or tab. The current work is left untouched.

Cleaning up

In **All Backed-Up Plans...**, **Trim to latest state** trims the checked plans to their newest version, **Delete selected plans** removes their backups. The file of a saved plan is left untouched.

Plans that were never saved, whose backups are older than a week and together take up space, Raily offers to delete on its own at startup. **No** deletes nothing and never asks about these plans again.

Save under a name you can find again

The backup is a safety net against crashes, not a substitute for **Save As....** A plan that exists only as a backup can disappear during cleanup.

See also

- [When something doesn't work](#)
- [Platform differences](#)

Account and server

With an account at raily.ch you save plans on the server, publish and share them, and a purchased offline license finds its own way back to you. Without an account, everything stays local.

In the browser the server is the usual storage location, because there is no file system there that **Save** could access on its own.

Login

	Where
In the browser	the button furthest right in the upper toolbar:  Login.... Once logged in, it shows your name there, with Account... (opens the website's account management) and Logout...
In the desktop app	 File → Login... and  File → Logout

The dialog asks for email and password, and offers **Create account** and a way to reset a forgotten password.

Buying works without an account too

A subscription needs an account, the offline key does not. See [License, free tier and subscription](#).

Opening and saving

The menu entries are named differently depending on the platform, because the normal case differs:

Action	In the browser	In the desktop app
Open from server	 File → Open...	 File → Open from Server...
Save to server	 File → Save As...	 File → Save to Server...
Open local file	 File → Open from Computer...	 File → Open...
Save locally	 File → Save to Computer...	 File → Save As...

Save without "As..." overwrites the open document, wherever it is stored. For a plan that has never been saved, it asks for a name.

The server's file browser knows folders, a trash, and **Rename**, and also accepts files dropped onto it. Before overwriting, it asks first.

Without logging in, the server entries are greyed out, and the tooltip says *"Not logged in"*.

Storage full

Raily then shows the used and the available quota. Save locally, empty the trash (this deletes permanently), or switch to a larger quota. Simply closing the dialog leaves nothing saved.

Sharing folders

Sharing happens on the website, in your account's file management: choose a folder, **Share folder**, enter an email address, and choose **Can edit** or **Read only**. The invited person sees the invitation after their next login.

In Raily, accepted folders appear in the file browser under **Shared with me**. You cannot save into a read-only folder.

Publishing

 **File → Publish** has two entries:

-  **Publish Plan...** puts the plan into the public gallery. Needs a login.
-  **Create Share Link...** creates a link without publicly listing the plan. Needs a login and a subscription, otherwise it says *"Needs a subscription"*.

 **File → Plan Gallery...** shows the published plans of all users, sorted by **Newest** or **Best**, even without logging in.

See also

- [License, free tier and subscription](#)
- [Backup and recovery](#)
- [Platform differences](#)

License, free tier and subscription

Without a license, Raily can be used as long as a plan stays small. It is unlocked in two ways:

	In the browser	In the desktop app
Subscription , tied to your account	yes	yes
License key , purchased once, without internet	no	yes

In the browser, then, there is only the subscription. The desktop app takes both: you sign in with an account that has a subscription, or you enter a license key.

The size limit

Without a license, a plan may have **200 objects**; every element is counted, including inside groups (some of the built-in objects consist of groups of triangles, which can cause editing to be unexpectedly blocked).

The warning only appears when you **enlarge** a plan past the limit, not when opening a large plan.

Above the limit, only select and delete still work

The plan stays visible and printable. Shrink it, sign in with an account that has a subscription, or enter a license key in the desktop app. **Check again** in the warning bar helps right after a purchase or signing in.

Watermark and printout

Without a license, the plan shows *"Unlicensed copy"* in the lower right; clicking it opens the purchase dialog. Every printed sheet carries *"printed with unlicensed raily.ch"* in the lower left.

Buying a license

The purchase dialog is opened by the watermark, **Get Raily 5** in the warning bar, or **About...** (in the browser under the Raily icon, in the desktop app under **Help**, on macOS under **Raily**).

Offer	Commitment	Applies to
Monthly subscription	cancel anytime	Browser and installed app
Yearly subscription	one year	the same, cheaper
License key	once, permanent	only the installed app for Windows, macOS and Linux, without account or internet

You are billed in euros including tax; at checkout it is converted to your currency. A subscription also brings more storage space on the server, the purchase dialog names the amounts.

When a subscription ends, your plans stay readable, and the size limit applies again.

The license key is not valid in the browser or on tablets. Try the app on your system before buying; there is no guarantee for future operating systems.

Only clicking a purchase button contacts the payment provider Paddle, and counts as consent to its cookies.

License key vs. subscription

	License key	Subscription
Applies to	this one installation of the desktop app	your account, wherever you sign in: in the browser and in the desktop app
Activate	Help → About... (macOS: Raily → About...) → Enter licence key . Only in the desktop app	sign in
Internet	never needed	needed. After a confirmation, the desktop app stays licensed offline for 48 hours
Purchasable without an account	yes	no

About... shows the current status, under **License details**.

See also

- [Account and server](#)
- [Platform differences](#)

Platform differences (Web and desktop app)

Raily exists both in the browser and as an installed **desktop app**. The tools themselves — editing, selecting, rotating, snapping, layers — are the same on both, and plan files open the same way on both. What differs comes down to the **window and the file**, not the drawing.

Desktop app only

- Opening a plan file from Explorer, Finder or a Linux file manager, by double-click or by dragging it in from outside.
- A confirmation prompt when closing a window that has unsaved changes.
- The automatic update check with a selectable channel (Stable/Beta/Alpha), see [Desktop app lifecycle](#).

The same on both

- **Backup and recovery**: Both back up in the background, keep the same versions, and offer unsaved work back at the next start. Only the location differs: browser storage on one side, the app's data folder on the other, see [Backup and recovery](#).
- **License**: An unlicensed copy shows the same watermark on both; no platform asks about it separately at startup, see [License, free tier and subscription](#).

See also

- [Desktop app lifecycle](#)
- [Backup and recovery](#)
- [License, free tier and subscription](#)

Desktop app lifecycle

What follows applies only to the installed **desktop app**. In the browser there is no program window to close, and no installer to update.

Closing the program with unsaved changes

Closing the window of a modified plan triggers a confirmation prompt: confirming closes the window, cancelling leaves the plan open.

Updates

When the main instance starts, the app checks for a new version — this can be turned off under **Settings**, group **Updates**. If there is one, an information bar appears at the top, whose urgency distinguishes between a regular, a significant, and a security-relevant update. You can open the new version, ignore just this one version, or dismiss the bar. Which **channel** is checked, **Stable**, **Beta**, or **Alpha**, is set in the same group.

Opening files from the operating system

Opening a plan file from Explorer, Finder or a Linux file manager, by double-clicking or "Open with", starts it in a **new window** — unless the app was not open at all yet, in which case it fills that first window. A file that is not a plan but is importable (STL, 3MF, an image ...) instead lands in the **currently open** window, just as with import from the menu.

See also

- [Backup and recovery](#)
- [Platform differences](#)
- [License, free tier and subscription](#)

Reference

When something doesn't work

Almost everything that suddenly “stops working” is a toggle that got switched at some point. This page is organized by what you see.

Nothing snaps anymore

What you see	Cause
A pole, signal or annotation won't bind to the track	 Snap to Special Points is off. This toggle applies to all targets that belong to an object, including the track under a pole
Still not, even then	The track is on a hidden layer . Whatever isn't drawn is not a snap target
Everything just hangs off the pointer	Snapping off is latched, one of the toggles in touch mode
The target isn't offered	The snap distance counts in screen pixels. Zoom in closer
The magnet icon is missing	It only exists in 2D, and only with rulers switched on. The settings are also in Settings

See [Snapping](#).

The object is in the wrong place

What you see	Cause
It's stuck in the hill, or floating	Snap to Ground is off
Everything new arrives too high up	The work plane is not at zero
It needs to sit at an exact height	Translate mode , anchor on the underside

I can't select it

Check	
Layer locked	A locked layer is drawn, but not selectable. See Layers
Layer hidden	Then it's simply not there
You're one level down	The orange notice at the top of the plan means clicks only hit the parts of <i>one</i> element.  takes you back out; see Groups

The numbers don't add up

Every length is off by the factor of my scale. The **Model scale / Real-world scale** toggle at the top of the property panel is set differently than you think; without an editor open, **Model scale for lengths and sizes** in the settings applies.

The plan scale is wrong. Set it early, in the [Plan Properties](#). Scaling does not rescue a plan set to the wrong scale: tracks are catalog parts and [do not scale](#).

The plan looks different than yesterday

A track looks the same despite a changed rail display, bedding is missing everywhere, or names appear everywhere. These are the **display overrides** under **View**. They take priority over the individual track, apply to **every** plan, and survive a restart. See [Rail display](#).

A red bar lies over a flex track. It is bent tighter than the plan's [smallest curve radius](#).

Nothing moves

Check  **Start animation**. As long as it isn't running, nothing moves — even a figure's arrow keys have no effect, and the clock stands still. Whether it's running is saved with the plan; an older plan opens paused.

A train disappears while running

You slowly ran into another train, and Raily **coupled** the two. Run into it too fast, and one **derails**. See [Placing trains](#).

The view won't move

You're stuck in **cockpit view** or **follow view**, where the camera travels with the vehicle.  **3D view** brings back the free camera.

If the mouse wheel pans instead of zooming, that's a toggle; see [Navigation](#).

I don't see anything in the plan

 **2D view** always zooms to the whole plan.

If the plan shrinks to a few pixels doing this, an element is sitting far outside the layout — often from a transposed digit while moving something.  +  usually catches the move. Earlier states are under **Help → Versions of This Plan...**, see [Backup and recovery](#).

Raily won't let me continue

“Free tier size limit reached.” See [License](#).

A track won't mirror. For a left-hand turnout, the catalog must contain a matching right-hand turnout. The message shown after [mirroring](#) tells you what happened instead.

See also

- [Technical limits](#)
- [Performance tips](#)

Keyboard shortcuts

Every shortcut can be changed under *Settings > Keyboard Shortcuts*. This list shows the defaults. **Ctrl** is **Cmd** on a Mac.

Everywhere

Command	Key
Cancel current operation	Esc
Open settings	Ctrl+,
Save	Ctrl+S
Save as	Ctrl+Shift+S
Open plan	Ctrl+O
Print	Ctrl+P
Show layer tool	Shift+L
Show environment tool	Shift+E
Delete selection	Del, Backspace
Undo	Ctrl+Z
Redo	Ctrl+Shift+Z, Ctrl+Y
Duplicate selection	Ctrl+D
Rotate selection	R
Rotate object	Shift+R
Move selection	T
Move object	Shift+T

In the plan

Command	Key
Move selection to layer	L
Add new rail	A
Draw polyline	K
Draw circle	C
Switch between 2D and 3D	V
Select track libraries	M
Open module editor	Shift+M
Copy selection	Ctrl+C
Cut selection	Ctrl+X
Paste	Ctrl+V
Drive or walk forward	↑ <i>(hold)</i>
Drive or walk backward	↓ <i>(hold)</i>
Steer left	← <i>(hold)</i>
Steer right	→ <i>(hold)</i>
Climb	E <i>(hold)</i>
Descend	Q <i>(hold)</i>
Look up	pageup <i>(hold)</i>
Look down	pagedown <i>(hold)</i>
Run or full throttle	Shift <i>(hold)</i>
Switch next turnout ahead	Space
Put what you ride back to its start	B
Board the vehicle you stand beside, or step out	Enter
Subtract solids or paths	-
Subtract solids or paths, keep originals	Alt+-
Merge solids or paths	+
Merge solids or paths, keep originals	Alt++
Intersect solids or paths	/

Command	Key
Intersect solids or paths, keep originals	Alt+/
Install into host	Shift+I
Select all	Ctrl+A
Group selection	Ctrl+G
Ungroup selection	Ctrl+Shift+G
Enter group	I
Leave group	O
Take out of the container being edited	Shift+O
Create reference	Ctrl+Shift+D

Placing track

Command	Key
Switch target connector	A
Switch docking connector	S
Next rail of this type	W
Previous rail of this type	Q
Choose straight track	1
Choose curved track	2
Choose turnout	3
Choose crossing	4
Choose crossing with turnout	5
Choose flex track	6
Choose transition track	7
Choose special element	8
Confirm insertion	Enter

Dragging elements

Command	Key
Switch docking connector	S

Rotating

Command	Key
Rotate +1° about X	↑
Rotate +15° about X	Shift+↑
Rotate -1° about X	↓
Rotate -15° about X	Shift+↓
Rotate -1° about Z	←
Rotate -15° about Z	Shift+←
Rotate +1° about Z	→
Rotate +15° about Z	Shift+→
Rotate -1° about Y	Alt+←
Rotate -15° about Y	Alt+Shift+←
Rotate +1° about Y	Alt+→
Rotate +15° about Y	Alt+Shift+→

Moving

Command	Key
Move left	←
Move left (large step)	Shift+←
Move right	→
Move right (large step)	Shift+→
Move up	↑
Move up (large step)	Shift+↑
Move down	↓
Move down (large step)	Shift+↓
Raise	Alt+↑
Raise (large step)	Alt+Shift+↑
Lower	Alt+↓
Lower (large step)	Alt+Shift+↓

Technical limits

Heights, gradients and cant change the track geometry itself. This page states the rules that follow from that, so a surprising result can be recognized as a consequence rather than a bug.

What the plan stores

In plan view, a track is **always flat**. Height, gradient and cant are stored as numbers at its ends, and the track is bent in between. It follows that:

- Snapping, selecting, printing and the 2D view all work with the unmodified plan view.
- If the values at two ends don't match at a joint, that's a visible step.
- Raily 4 does not know these numbers and shows the line flat. They are not lost in the process.

Ramps

- **Bend the track is the default**, because a track piece tilted as a rigid whole cannot hold its gradient through a curve: it leaves a curve flatter than it entered it, flat outright after a quarter circle. The kink then sits at the joint. **Keep elements flat** exists for compatibility.
- **A ramp must have transitions at its ends**. Two tracks count as connected when their ends are at most **3 mm** apart and their directions match to within about **1°**, including in height. Even 2% gradient without a transition is more than that. The ramp editor rounds them off on its own.
- **At the upper, free end of a helix**, the track keeps climbing at full gradient. Flat track will not snap in there; place a short ramp in between.
- **A turnout is rigid**. It can be raised and tilted, but not twisted, and it gets no cant.
- **Poles and labels** remember their position as a length along the track. Because a bent track gets longer, they shift a little on a steep ramp — about 9 mm over 10 m.

Cant

The calculation is $\tan(\text{cant}) = v^2 / (r \cdot g)$, point by point, then clamped to the **maximum cant** (default 160 mm real-world, converted) and to the **cant ramp 1:N**. That's why the twist starts already **before** the curve.

If a curve is too short for its ramp, it gets **less cant**, not a steeper ramp. The display tells you.

Helix, tunnel and embankment

A track is a line and may come arbitrarily close to itself. A tunnel, an embankment or a cutting sweeps a cross-section along the line. If the turns of a helix lie closer together than the profile is tall, that solid does not exist, and Raily refuses it. The rise per turn is then to blame.

If a profile is wider than the curve's radius, its inner part collapses. The part goes flat there, but stays usable.

How wide a tunnel is suggested

It isn't the width of the car that decides whether it fits through. In a curve, a long car swings its middle inward and its ends outward, and on model radii that adds up to a lot: a 26.4 m car sweeps out almost double its own width on R1 (360 mm in H0).

Create tunnel... therefore suggests the inside width based on what reference vehicles sweep out at **this track's radius**. As a guideline, for a 4500 mm wide bore (real-world size, H0):

Vehicle	from radius
Passenger wagon, 26.4 m	725 mm
Engine, 15.4 m	235 mm
Two-axle freight wagon, 14 m	190 mm
Short wagon, 8 m	100 mm

The number is a lower bound

The calculation assumes a rigid block on a fixed line. It knows nothing of sway, suspension, buffers, footboards, or the gaps between coupled cars, and two tracks side by side each need the allowance separately. If you're building tight, push the longest car through the curve by hand before you pile the mountain on top.

Dragging in the 3D view

A dragged element lands where the ray through the pointer meets the work plane. Near the horizon there is no longer a usable spot for that: the element **stays put** until the pointer points at a sensible location again. Tilt the view steeper rather than dragging against the limit.

Very thin lines

A line with a width is built as a surface, and that costs more the thinner it is. Imported drawings often come in at 0.26 mm. If you don't need the width, set it to **0**: the hairline is always one screen pixel wide.

Catalog data

Manufacturer, sales period, track material and build depth come from catalogs, technical documentation and measurements of real track. Older products are documented patchily, and almost no manufacturer publishes sleeper heights; where they're missing, they are estimated. Hence: **to the best of our knowledge, without guarantee**. The track geometry is correct; the build depth may be off in individual cases.

If you measure it yourself, you'll know more precisely. We welcome corrections with the track library, item number and measured value at support@enigon.com.

See also

- [Embankments and ramps](#)
- [Create helix](#)

- [Create tunnel](#)

Performance tips

Large layouts can slow down the display. These measures help.

While editing

- **Hide layers.** What sits on a hidden layer never enters the scene at all — it costs no rendering, no lighting, and no shadows. Hide whatever you are not currently working on. See [Layers](#).
- **Work in the 2D view.** 2D draws without shadows and without lamp light, and is correspondingly faster. Switch to 3D only to check your work.
- **Turn off shadows and lamps.** In [Settings](#) under *Rendering*: turn off **3D directional shadow**, set **3d lamps and headlights** to *Off* or lower. Shadows are the single biggest item in 3D.

On the plan

- **Raise terrain resolution only where detail is needed.** The triangle count of a surface grows with the product of the **grid points** X and Y. See [Terrain editing](#).
- **Simplify detailed STL models.** The import dialog shows the triangle count; **Simplify** lowers it, even after the fact — see [Import files](#).

See also

- [Layers](#)
- [Settings](#)

Glossary

The words Raily uses in its own particular way.

- **Annotation:** a marking on the track that describes a stretch, not individual pieces of track. From it, **Refresh** builds a tunnel, a cutting, a block. See [Getting started 3](#).
- **Marker:** the arrow that sits an annotation on the track. **Region:** the stretch Raily computes from it.
- **Work plane:** the plane in which new things are created and along which dragged things travel. See [Work plane and coordinates](#).
- **Build depth:** rail, sleeper, and bedding together. A track's height means its **railhead**; the build depth hangs beneath it. See [Getting started 6](#).
- **Vertical easement:** the vertical transition curve at the ends of a ramp.
- **Layer:** groups objects that are hidden or locked together. See [Layers](#).
- **Install:** bind an element to a **host** so it travels along with it and, if it can, cuts an opening into it.
  key.
- **Snapping:** locking into place while dragging — onto the grid, guides, points, track ends, surfaces. See [Snapping](#).
- **Group:** several objects chosen and moved as one.
- **Multi-tool button:** a button with a small triangle, hiding a list behind it. See [Conventions](#).
- **Module:** one manufacturer's track system, for example *Märklin C Track*. The word has nothing to do with layout modules (Fremo, segments) despite the shared name.
- **Cant:** the outer rail sits higher than the inner rail on a curve.
- **Connector:** the end of a track where another one docks.
- **Linked copy:** shows the same element a second time and follows every change to the original. A duplicate, by contrast, becomes independent the moment it's copied.
- **Helix:** a curve that repeats while climbing. See [Create helix](#).