Advanced Trains for Minetest

The ADVTRAINS mod for Minetest

September 4, 2024

Contents

Introduction

ADVTRAINS is a mod with the goal of introducing realistic trains and rail equipments. It also has features allowing automated trains on a large scale, including interlocking and a few methods of scripting.

This manual is written as there does not appear to be much up-to-date information on ADVTRAINS, despite a number of people being interested in the mod. There has been a number of attempts to create various guides, but these only cover certain topics instead of being a comprehensive manual. This manual is therefore created as a contribution to the ADVTRAINS community.

This manual is typeset using LaTeX; its source code can be found here: `https://codeberg.org/y5nw/advtrains-doc`.

Acknowledgments

A notable portion of the manual is influenced by various other guides and manuals for ADVTRAINS. In particular:

- The format of the API documentation is influenced by R^{7}RS.
- Section 3 is written with reference to orwell's interlocking guide and Blockhead's video explaining the three-station setup. The links to both sources can be found in that section.
- The manuals related to railway time are modifed from the railway time API documentation written by orwell. Part of the section is also taken from `advtrains_line_automation/railwaytime.lua`
- The cover image is created by Blockhead and licensed under CC BY-SA 4.0. The original image can be found on: `https://forum.minetest.net/viewtopic.php?p=418745#p418745`.

Conventions

- Unless otherwise specified, the unit of length is meter (or, more technically, nodes).
- Unless otherwise specified, the unit of speed is meter per second.
- Unless otherwise specified, the unit of acceleration is meter per second squared.
- Arguments to functions and ATC commands are delimited using angle brackets $\langle$*like this*$\rangle$.
- Argument names are generally lowercase, but the first letter may be capitalized.
- Lua string constants are written without quotation marks when the content does not resemble any numeric or symbolic constant and when used as enumerators or table indices.
- The action of left-clicking a node with a trackworker is called *rotation*. The action of right-clicking a node with a trackworker is called *adjustment*.

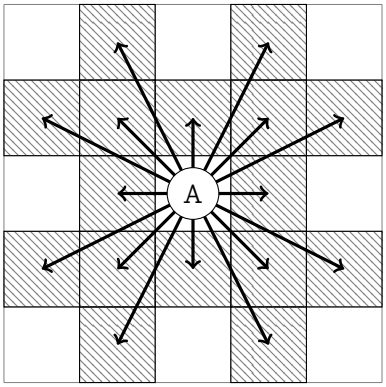
Interlocking notations

- The $\overleftarrow{\top}$ symbol stands for a TCB assigned to the track.
- The $\overset{\bullet}{\top}$ symbol stands for a TCB assigned to the track with a signal assigned to the TCB side that the light is facing (here: the signal is assigned to the left side of the TCB).
- In the context of track sections, $\overline{\mathrm{PQ}}$ indicates the section containing the tracks between P and Q.
- In the context of (interlocking) routes, the notation $\mathrm{P} \to \mathrm{Q}$ indicates a part of a route where the train starts at P and continues at Q, while the notation $\mathrm{P} \xrightarrow{\mathrm{T}\to\mathrm{Q}} \mathrm{R}$ indicates the same route with the turnout T explicitly set up to face Q.
- $\perp$ is used to indicate the end of the track.

1 Tracks

1.1 Angles

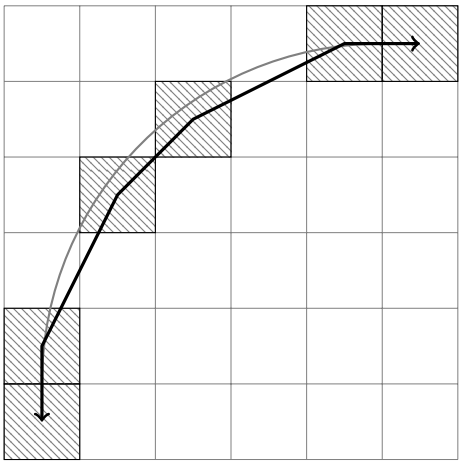
Unlike regular carts in Minetest, tracks can connect in 16 cardinal directions instead of four:



In the figure above, the tip of the arrows, along with the marked square, indicate positions where tracks can be placed to connect to the position A.

Tracks that are not axis-aligned are sometimes referred to as 30° (although it is technically $\arctan 0.5$) or 45° tracks. However, the specific meaning of the terms may depend on the context of the discussion.

The use of 16 cardinal directions instead of four allows the creation of curves that are more realistic than their counterparts made using minecart rails. For example, the following figure shows the placement of the smallest (*tightest*) curve with a 90° turn, with the gray line indicating a perfect arc of the turn radius. Note that tracks can only turn one "step" at a time.



However, this is still notably smaller than in real life. For comparison, the smallest curve for the H0 model (1:87) has a radius of 360mm, which is 31.32m when scaled to real life.

1.2 Track placement

The usual way of placing a track is with a *track placer*, a tool that places the tracks and, in many cases, automatically adjusts them to connect to nearby tracks. In particular, tracks automatically adjust to curves when placed.

You can get track placers from the creative inventory or by crafting. Please refer to the crafting guide for the recipe.

To place the track, simply right click the node to place the track on with the trace placer.

1.3 Track adjustment

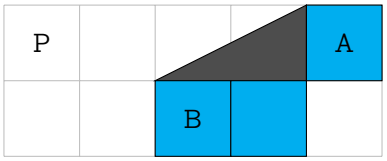
You can use a trackworker to make certain adjustments to tracks. Specifically,

- Left-clicking on a track with the trackworker toggles between the variants of the particular track, if any.
- Right-clicking on a track with the trackworker rotates the track counterclockwise.

Typical examples of "variants" of tracks include turnouts and curves. Please note that Y-turnouts and different types of crossings have their own variants. These variants are described below. Additionally, special tracks (ATC track, station tracks, etc.) are usually only straight, and certain mods that provide additional tracks (e.g. linetrack) may not provide a full set of tracks available in the official track set.

1.4 Slopes

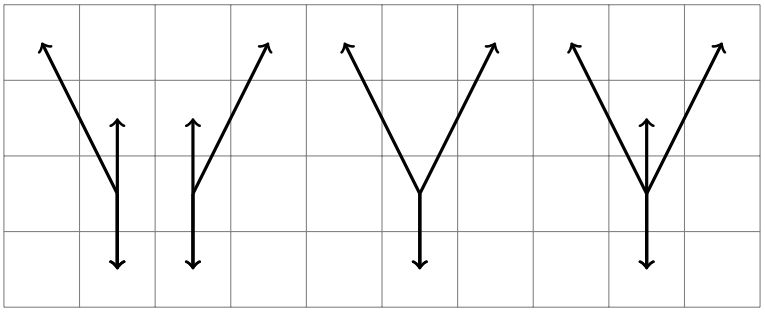
To place slopes, face the upper end of the slope and click on the nth node before the upper end of the slope for a slope with an incline of $1/n$. For example, if you want to build a slope with an incline of 50% (the slope marked gray), you would look at node A and click on node B while standing in such a way that the direction you are looking at is also the direction in which the slope goes up (e.g. at node P):



Please note that, in the figure above, the three colored nodes must already be placed.

1.5 Turnouts

Turnouts are tracks that connect tracks from more than one direction. Some examples of turnouts are shown below.



The figure above aligns the tracks in a way that they always merge into the track that connects to the south. This is mainly done for intuitivity — like tracks, you can rotate the turnouts freely in any of the 16 directions.

Turnouts are placed by using regular track placers and adjusting the tracks with the trackworker.

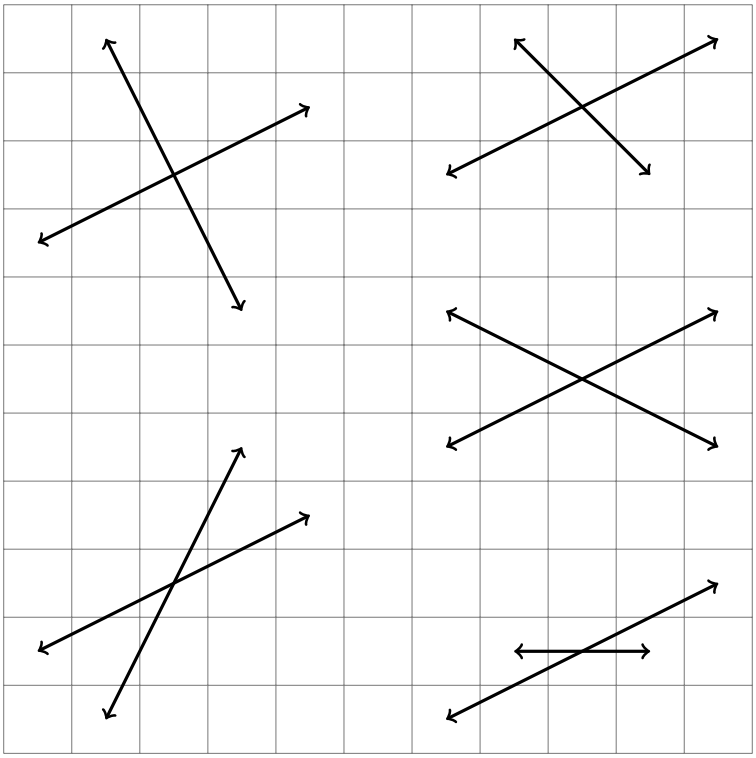
1.6 Crossings

There are a few crossing nodes that allow you to create intersections:

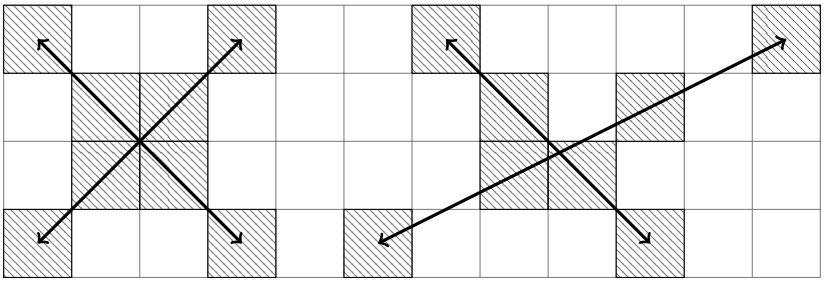
- Perpendicular crossings: intersections where two tracks are perpendicular to each other
- $90° + x$ crossings: intersections where only one of the two tracks is axis-aligned
- diagonal crossings: other crossings that do not fit in the two categories mentioned above

Each type of crossing has its own track placer. The trackworker can not be used to switch between the different types of crossing tracks.

The figure below shows a few examples of crossings that are available.



It is also possible to create track layouts without using crossings. Some examples are shown below.



As an exercise, build a T junction when you finish reading this section. You will need this later in section 3.10.1. You can also look at the scheme in that section if you need a hint.

1.7 Special tracks

ADVTRAINS also have a few special tracks, some of which are explained in their own sections. Note that these tracks are only available as straight tracks.

ATC track item

Track that sends ATC commands to a train (see section 2.7 for more information)

Loading track item

Track that makes a train load as many items as possible from a chest below the track if the train is moving in the direction indicated by the arrow on the track at a speed lower than 2m/s.

Unloading track item

Track that makes a train unload as many items as possible into a chest below the track if the train is moving in the direction indicated by the arrow on the track at a speed lower than 2m/s.

Temporary speed restriction rail item

Track that forces a train to pass with an arbitrary speed limit.

Station track item

Track that makes trains temporarily stop at a given point (see section 3.8 for more information)

LuaATC track item

Track that can be programmed in Lua (see section 5 for more information).

2 Train controls

To get on a train, click on the train while its doors are open, or click on the train while holding the Sneak key. You need to be in the driver stand in order to control the train.

Train movement currently depends on two factors: the lever of the train and the percentage of locomotives in the train. Details on train physics can be found in section A.

Unlike in real life, levers are not moved continuously. Instead, there are a few lever "positions" that are recognized by ADVTRAINS:

- The lever 0 triggers the emergency brake. It can be triggered by the ATC BB command (see section 2.7).
- The lever 1 triggers the default brake.
- The lever 2 ("roll") slows down a train as if it was on a horizontal surface without any traction
- The lever 3 does not affect the speed of the train.
- The lever 4 accelerates the train.

2.1 Train formspec

If you have access to the driver stand of the wagon, right-clicking the train will open up a formspec will a few buttons, depending on the status of the train:

Passenger area — UI element

Move to the passenger area.

Driver stand — UI element

Move to the driver stand.

Onboard computer — UI element

Open the onboard computer.

Wagon properties — UI element

Allow setting wagon properties.

Get off — UI element

Get off the train.

(Doors closed) — UI element

Get off the train by right-clicking the train while holding the Sneak key.

2.2 Onboard computer

The onboard computer allow setting certain properties of the train. Feel free to explore it yourself. A few fields are explained here in detail:

Line number — UI element

The line name of the train. Despite the label, you can set this to anything. This field is mainly used for interlocking.

Routing code — UI element

The routing code of the train. This field is mainly used for interlocking. Unlike line numbers, you can specify multiple entries in this field, separated by spaces.

Train overview — UI element

This section is only shown when the train is not moving. It allows you to couple and decouple wagons from the train.

2.3 Wagon properties formspec

The "wagon properties" formspec allows setting properties that apply to each wagon. The following fields are present in the formspec:

Whitelist UI element

("Allow these players to access your wagon") A list of other player that may access the inventory and the driver stand of the wagon.

Wagon road number UI element

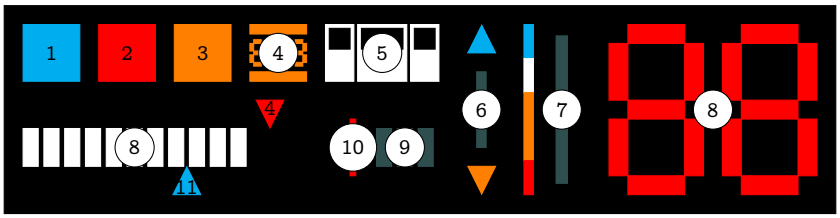
An identifier that is applied to the wagon. It is mainly intended to be a human-readable alternative to wagon IDs.

Freight code UI element

The freight code of the wagon. This is mainly used by LuaATC and allows, among other things, different wagons to be shunted together for certain parts of a journey even if they have different destinations.

2.4 HUD

The HUD is a visual representation of certain information about the train. An overview of the HUD is as follows:



1. ATC indicator
2. LZB indicator
3. Shunt indicator
4. Aspect of and distance to the upcoming LZB entry (usually a signal)
5. Door indicator
6. Reverser
7. Lever
8. Speed indicator
9. Max speed indicator
10. Speed restriction indicator
11. ATC target speed indicator

2.5 Keybindings

Hitting any movement key as a passenger will drop you off the train when a door is open or when the wagon has no doors. Alternatively, the Sneak+Aux1 keystroke will drop you off a train, regardless of whether doors are open.

The following keybindings are available to the driver, regardless of the mode of the train.

- Sneak+Forward: Set the ATC target speed to the maximum speed

- Sneak+Backward: Set the ATC target speed to 0
- Sneak+Left: Set the ATC target speed to 4
- Sneak+Right: Set the ATC target speed to 8
- Sneak+Jump: Open the onboard computer
- Jump+Backward: Emergency brake

2.6 Manual control

Manual control makes the user in charge controlling the train. The following keybindings are available when manually controlling the train:

- Forward: Accelerate.
- Backward: Roll when the train is moving, reverse otherwise.
- Left: Open/Close the door on the left side and, if the door on the right side is open, close the door on the right side.
- Right: Open/Close the door on the right side and, if the door on the left side is open, close the door on the left side.
- Jump: Brake or, in ATC mode, switch to manual control

These keybindings (except for the jump key) do not work in ATC mode. The lever is moved to a specific level only as long as the corresponding key is pressed.

2.7 ATC

ATC is a method of automatically controlling trains. Certain modes can be directly activated using specific keybindings, but more common ways of using ATC include:

- Directly using ATC tracks,
- Using station/stop tracks, and
- Using LuaATC

ATC tracks can be used to send custom ATC commands to the train passing the track. The ATC commands available are listed below:

`A0` lexical syntax

Disables ARS.

`A1` lexical syntax

Enables ARS.

`BB` lexical syntax

Actives the emergency brake until the train stops.

`B`⟨*speed*⟩ lexical syntax

Brakes until ⟨*speed*⟩ is reached.

`Cpl` lexical syntax

Enables autocouple mode.

D⟨*time*⟩ lexical syntax

Waits for ⟨*time*⟩ seconds before continuing execution.

I⟨*condition*⟩⟨*consequent*⟩; lexical syntax

Executes ⟨*consequent*⟩ if ⟨*condition*⟩ is met.

I⟨*condition*⟩⟨*consequent*⟩E⟨*alternate*⟩; lexical syntax

Executes ⟨*consequent*⟩ if ⟨*condition*⟩ is met, ⟨*alternate*⟩ otherwise.

K lexical syntax

Kicks all passengers (players not driving the train) off the train only when the train is stopped and its doors are open.

OC lexical syntax

Closes doors on both sides.

OL lexical syntax

Opens doors on the left side and closes doors on the right side (if open).

OR lexical syntax

Opens doors on the right side and closes doors on the left side (if open).

R lexical syntax

Reverses the train only when the train is not moving.

S⟨*speed*⟩ lexical syntax

Accelerates to ⟨*speed*⟩ or roll if the train is faster then ⟨*speed*⟩.

W lexical syntax

Waits until the target set by S or B command is reached.

For the B, D, and S commands, the argument should be written immediately after the command, without any whitespace.

The following ⟨*condition*⟩s are available for the ATC I command:

\+ lexical syntax

True when the train is driving in the same direction as the arrows on the ATC rail.

\- lexical syntax

True when the train is driving in the opposite direction of the arrow on the ATC rail.

<⟨*speed*⟩ lexical syntax

True when the train is slower than ⟨*speed*⟩.

>⟨*speed*⟩ lexical syntax

True when the train is faster than ⟨*speed*⟩.

<=⟨*speed*⟩ lexical syntax

True when the train is not faster than ⟨*speed*⟩.

>=⟨*speed*⟩ lexical syntax

True when the train is not slower than ⟨*speed*⟩.

2.8 LZB

> The explanation of LZB is specific to ADVTRAINS. Please refer to other sources, such as Wikipedia, for information on the use of LZB in real-life rail lines.

In ADVTRAINS, LZB is a system that makes sure that trains realistically brake to the desired speed, overriding ATC and manual control if necessary.

LZB speed targets are temporary and associated to a specific point - it only makes sure that the front part of the train passes that point at the desired speed. For example, a point speed restriction rail with a speed target of 3 will only make sure that the front of the train passes the PSR rail at the speed of 3, but the train is allowed to accelerate up to the speed limit (or the maximum speed, if there is no speed limit) after the head of the train passes the PSR rail.

2.9 ARS

Signals using ARS will select the appropriate route for the train only if ARS is also enabled on the train. This mode is enabled by default. This feature is mainly used by station tracks, which disables ARS for the train when it arrives at the station and re-enables ARS before departure.

2.10 Autocouple mode

If enabled, the train automatically couples with any train it collides with. This mode is disabled by default.

Historically, a train would also automatically couple with any train is collides with. This behavior was changed with the introduction of the autocouple mode.

2.11 Shunt mode

This mode restricts the velocity of the train to 6m/s and, allows a train to proceed past a signal at danger if the signal allows the train to proceed in shunt mode. It is disabled by default and automatically enabled when the train reverses in an interlocked section.

3 Interlocking and line automation

Interlocking is a set of equipments employed to prevent train collisions while allowing trains to go to their destinations. This chapter will explain some basic concepts and includes a task at the end that you can try to do yourself.

Please note that this chapter is also intended to be used as a reference, so some things may be explained in a way that is not easy to understand at first. If that is the case, the following links may be helpful, albeit possibly outdated:

- The online interlocking guide: `https://advtrains.de/interlocking`
- Blockhead's Youtube video showing a simple three-station setup (see section 3.11): `https://www.youtube.com/watch?v=ylG1vHj4zjg`

It is also recommended for new players to read through the entire section to understand various concepts of the interlocking system before following the instructions in the subsections. Section 3.9 includes certain examples that may be helpful.

3.1 Basic concepts of interlocking

Unlike road traffic, trains are not always able to brake to a complete stop within a distance that the driver can see. It is therefore required that the track ahead of a train is secured using technical means[1] in such a way that the train can run on the tracks safely even when the driver has limited sight. Among other things, it is necessary that

- The train does not run into another train unless this is intended for the purpose of shunting.[2]
- Track components (in particular, turnouts) are not modified while the train is running on the tracks.

A path along such tracks, along with the states that various components should be put into for safe operation, is called a *route*. A route can be *set* only if the above conditions can be met. The system used for setting routes is called *routesetting*. After setting a route, track components can no longer be adjusted, and conflicting routes for other trains are not allowed to be set.

Note that setting conflicting routes is not *allowed* in the sense that the interlocking system would not allow such paths to be set.[3] However, this alone does not directly prevent a train from proceeding along a potentially dangerous path. Therefore, the signaling system is needed to make sure that trains only follow routes that are set for the train to travel along. Signals are described in detail in section 4.

Note that there is a subtle but important difference between rail and road traffic. A rail signal showing that the route ahead is clear indicates that the train may safely proceed to the next signal and that there is no obstacle along the route. A green light in road traffic does not guarantee this - it can be green even in a traffic jam, in which case blindly driving ahead likely has undesirable results.

To guarantee that the route ahead is clear, ADVTRAINS uses a system known as *fixed block signaling*. Tracks are divided into *sections*, and each section should not be occupied by more than one train at a time. Routes are then set by *reserving* the track sections along the route until the train passes through the section. A route can therefore only be set if none of the sections along the route is occupied by or reserved for a different train.

[1] In terms of ADVTRAINS, the track is secured by the interlocking system.

[2] In addition to preventing collisions from the front (and, implicitly, the rear) end of the train, real-life rail operation additionally involves preventing accidents in which one train (usually a runaway train) collides into the side of another train, especially at junctions. While prevention against such accidents (known as *flank protection*) is possible with ADVTRAINS' interlocking system, this currently comes at a cost of lower capacity due to internal limitations of the interlocking system, while the risk of side collision is usually low enough to be tolerable.

[3] There are also Minetest-specific situations that the interlocking system cannot guard against, such as manually placing wagons along a set route.

3.2 TCBs

Track circuit breaks (TCBs) are nodes that can be assigned to two-way tracks and indicate the limits of track sections. TCBs have two sides - each corresponding to a direction of the track that the TCB is assigned to - that can be assigned to two track sections.

To assign a TCB to a track:

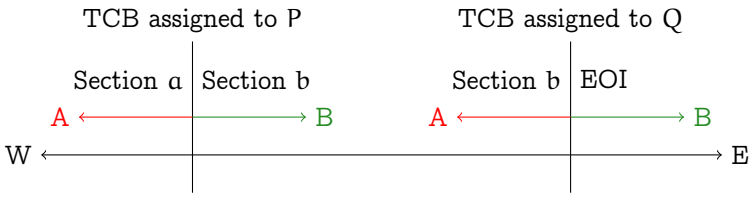
- Place a TCB, ideally adjacent to the track it will be assigned to.
- Right-click the TCB.
- You will be told in the chat to punch a track to assign the TCB to. Punch the track to assign the TCB to - this track will, in later subsections, be the boundary of track sections.
- A TCB marker showing the two sides (A and B) will appear on the track you have assigned the TCB to. This will be explained in further detail below.

After assigning the TCB, you can right click it to open the TCB formspec. This will be used in the following sections. Please note that the TCB formspec has a section for side A and one for side B - when following the instructions in the following subsections, make sure you click the button for the right TCB side. The two sides of the TCB are assigned based on the orientation of the tracks. In particular, straight tracks with different orientations can still appear visually the same. For this reason, most graphs in the rest of this chapter will use cardinal directions instead of actual TCB sides, or omit TCB sides if these are irrelevant.

For new players, it is recommended to set up all TCBs before creating new track sections.

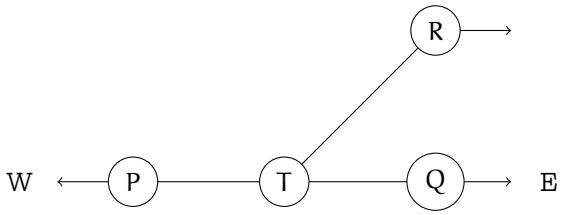
3.3 Track sections

In terms of interlocking, track sections are segments of tracks that can be occupied by a train or reserved for a train to pass. The following graph shows TCBs and sections along a line segment. For simplicity, letters are used instead of actual coordinates and IDs.



Section a begins at the side A of P (sometimes written as P/A) and extends to the west beyond the graph. Section b is limited by P/B and Q/A, effectively spanning the line segment $\overline{PQ}$. Q/B is not part of any track section and is therefore known as *end of interlocking* or *EOI*, as the track beyond it is not part of the interlocking system.

The graph below shows a track segment where P, Q, and R form a section around the turnout T. Notice that a train entering from Q cannot directly reach R without reversing (or vice versa), while a train entering from P can reach both Q and R.



To create a track section, click on "Create new track section" in the TCB formspec. This should create a new track section and add the TCB side to the track section. Other track sides that face the track side you created the TCB with will also be added to the track section. In the graph above, creating a track section at the west

side of Q will automatically add the south-west side of R and the east side of P to the track section. In addition, TCBs placed inside track sections are automatically added to the adjacent track sections.

After the track section is created, you can click on "Show track section" to open the track section formspec, or "Remove from section" to remove the TCB side from the track section.

In some cases, such as crossings, you may need to manually add TCB sides to a track section. This can only be done by creating a track section for the TCB side and then merging the two track sections.

To merge two track sections:

- Click "Join into other section" in the formspec of the section that needs to be merged.
- Click "Join with ⟨*section*⟩" in the formspec of the section to merge the other section into, where ⟨*section*⟩ is the ID of the section that needs to be merged.

To abort the procedure above, click on the "X" button on the right.

The track section formspec also has a button labeled "Dissolve section". Clicking on that button will remove the track section entirely.

In most cases, the two sides of the TCB should belong to different sections, or at least one side belongs to EOI. If this is not the case (i.e., both sides of the TCB belong to the same track section), it likely means that the presence of the TCB is irrelevant for the track section. With the current implementation of the section occupation system, this is problematic because the section behind the train is always freed, even when the train passes a TCB with both sides belonging to the same section. You will likely also receive a warning that a TS lock is encountered during a real run of the routesetting routine.
If you run into such a situation, dissolve the track section and make sure that every TCB is assigned to a track and delimits the track sections that the two sides of the TCB are assigned to, and then create the sections again.

If you want to insert a TCB inside a track section (i.e. split the track section in two), it is recommended to do so by dissolving the track section before placing the TCB. Make sure to recreate the track sections and adjust the routes as needed.

3.4 Signals

Each TCB side can have a signal assigned to the side. The signal will then indicate whether the train is allowed to enter the section to which the same TCB side is assigned to. Signals need to be set up in order to be able to set up routes. In the previous graph, in order to let trains move from P to Q or R, a signal needs to be set up on the side of P that faces the track section (i.e. the east side of P). For an overview of the signals available in ADVTRAINS, refer to section 4. Signals should conventionally face the opposite direction of the side of the TCB so that the driver can see the signals.

Every signal can optionally have an influence point. This is the point where the aspect of the signal becomes effective, and should be located before the train passes the TCB side that the signal is assigned to. The influence point for signals that are independent of track sections is irrelevant, but should conventionally be close to the signal.

To assign a signal to a TCB:

- Click on "Assign a signal" in the TCB formspec.
- You will be asked to punch a signal. Punch the signal to assign to the TCB side.
- If you are prompted to set the influence point, you can do so by left clicking the track you want to assign the influence point to while looking in the same direction that the train drives in.

You can also set the influence point of a signal without assigning it to the TCB. To do so, right click the signal while holding the Aux1 key. For signal signs, right-clicking the signal will bring up the same prompt.

After assigning a signal to a TCB side, you can right click it to open up the signal formspec. The "Influence point" button will open a formspec let allows you to change or remove the influence point. The "Unassign signal" button will unassign the signal from the TCB side while keeping the influence point (if present).

3.5 Routes

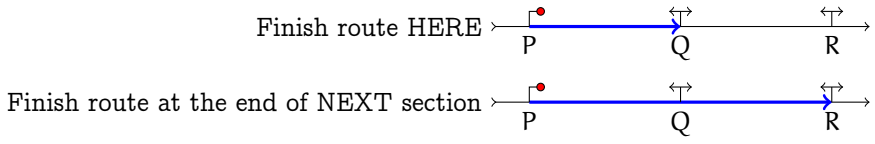
Routes contain information on where a train goes to, which section should be reserved for the train to pass, and how certain components, such as turnouts, should be set up. Routes are bound to TCB sides, but they usually need to start at ones with a signal assigned. In most cases, routes should also end in such a way that a train leaving the route can immediately enter the next route, with the most common exception being situations where dead ends are involved, such as train yards and termini where the line physically ends (e.g. Stuttgart Hauptbahnhof).

To create a route:

1. Click on "New route" in the formspec of the signal assigned to the starting TCB side.
2. Set up passive components (e.g. turnouts) in the section appropriately and lock them by punching, if necessary.
3. Punch the next TCB that the train should pass. The track that the TCB is assigned to should have a marker showing "END ROUTE".
4. If the following track section is the last track section in the route and does not require any setup require in step 2 or have more than two TCB sides, click "Finish route at the end of NEXT section".
5. If the route leads to the end of the physical line, click "Finish route at the end of NEXT section".
6. If the route ends at the TCB, click "Finish route HERE".
7. If the route continues beyond the TCB, click "Advance to next route section" and repeat the above steps, starting from step 2.

You can also click "Finish route at the end of NEXT section" at the starting TCB. This can be useful when the last part of a route does not involve setting up further passive components or when the route continues to a dead end.

If the TCB is not considered suitable for route continuation, please check that the side of the TCB from which the train passes the TCB is part of a track section and that the train can reach there without passing a point assigned to another TCB. If you see "Advancing over next section is impossible at this place. End of interlocking.", please check that the side of the TCB corresponding to the driving direction is part of a track section.



After finishing the route, you are prompted for the name of the route. It is recommended to use a sensible name. You should then see the route formspec. If you don't, click on the name of the route you just created and click "Edit route", and proceed to the next section regarding ARS.

3.6 Automatic routesetting

Automatic routesetting (ARS) is a method of choosing a route based on a set of matching patterns. In the previous section, you learned to set up a route, and there is a text input box where you can enter the ARS rules. This area is empty by default, which means that the route is not selected in any case.

ARS rules are delimited by a newline. Each line can be one of the following:

`# ⟨comment⟩` pattern

Comment

`LN ⟨line⟩` pattern

Matches trains with the exact line name ⟨*line*⟩.

`RC ⟨routing code⟩` pattern

Matches trains that contain the routing code ⟨*routing code*⟩.

`!LN ⟨line⟩` pattern

Matches trains with a different line name than ⟨*line*⟩.

`!RC ⟨routing code⟩` pattern

Matches trains that do not contain the routing code ⟨*routing code*⟩.

`*` pattern

Matches all trains

A whitespace is required before the argument.

ARS rules are designed to short-circuit. The route is selected if a train matches any of the rules.

In some cases, you may want to disable ARS. To do so, click on "Disable ARS" in the signal formspec. Clicking on the same button will enable ARS.

3.7 Automatic working

Automatic working is a system that sets the route after the train passes the signal. To enable automatic working, click on "Enable Automatic Working" in the signal formspec after the route is set. Clicking on the same button again will disable automatic working.

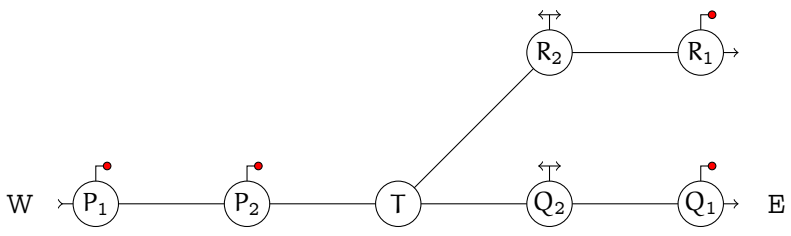
3.8 Station/stop tracks

Station/stop tracks (sometimes simply called *station tracks*) are special tracks that register a station where a train should stop. The interface of the formspec should be self-explanatory, with a few things to notice:

- You need to click "Save" after changing the settings.
- The station code is used to identify the station. The station name is shared among all tracks with the same station code (not vice versa).
- Station tracks disable ARS for the specific train when the train arrives at the station and enables ARS on the train before departure.

3.9 Basic interlocking examples

3.9.1 Unidirectional diverging junction



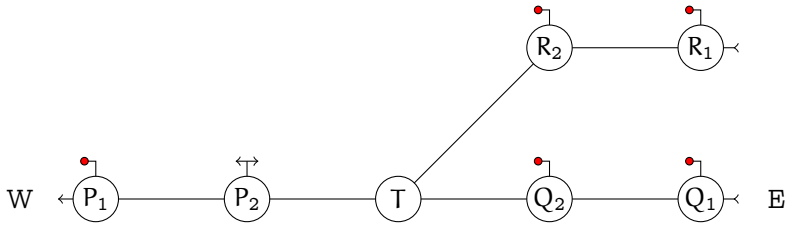
There are two ways for a train to travel from west to east:

- $P_1 \rightarrow P_2 \rightarrow T \rightarrow Q_2 \rightarrow Q_1$
- $P_1 \rightarrow P_2 \rightarrow T \rightarrow R_2 \rightarrow R_1$

Note that the train goes through $\overline{P_1P_2}$ in both cases. $\overline{P_2T}$ is also shared, but T needs to be set up based on the destination and the point cannot be assigned to a TCB. It is also encouraged to set up routes in a way that trains do not occupy the section containing any turnout while the train waits for a red signal (or stops in any way). This means that three routes can be set up for this example:

- $P_1 \rightarrow P_2$
- $P_2 \xrightarrow{T \rightarrow Q_2} Q_2 \rightarrow Q_1$
- $P_2 \xrightarrow{T \rightarrow R_2} R_2 \rightarrow R_1$

3.9.2 Unidirectional converging junction

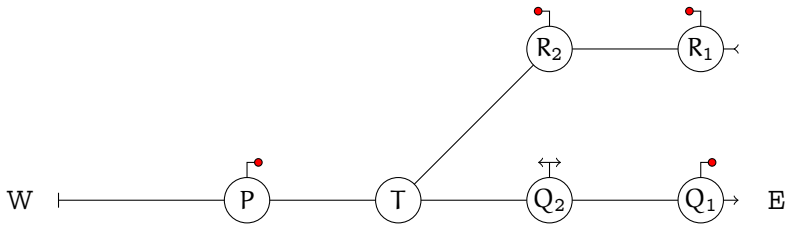


This setup is effectively the same junction as above, with trains running in the opposite direction. The following routes need to be set up:

- $R_1 \rightarrow R_2$
- $Q_1 \rightarrow Q_2$
- $R_2 \xrightarrow{T \rightarrow R_2} P_2 \rightarrow P_1$
- $Q_2 \xrightarrow{T \rightarrow Q_2} P_2 \rightarrow P_1$

Note that the turnout at T needs to be set up to point in the correct direction. The behavior of the interlocking system (and ADVTRAINS in general) is otherwise undefined.

3.9.3 Dead end



In this setup, trains from R_2 travel west and then reverses to continue toward Q_1. The following routes need to be set up:

- $R_1 \rightarrow R_2$
- $R_2 \xrightarrow{T \rightarrow R_2} P \rightarrow \perp$
- $P \xrightarrow{T \rightarrow Q_2} Q_2 \rightarrow Q_1$

Note that, for the second route, $\overline{P \perp}$ must be set up as a track section. Click on the "Finish route at the end of NEXT section" button at P. This is important as ending the route at P would allow the route to be set even if the track section $\overline{P \perp}$ is occupied.

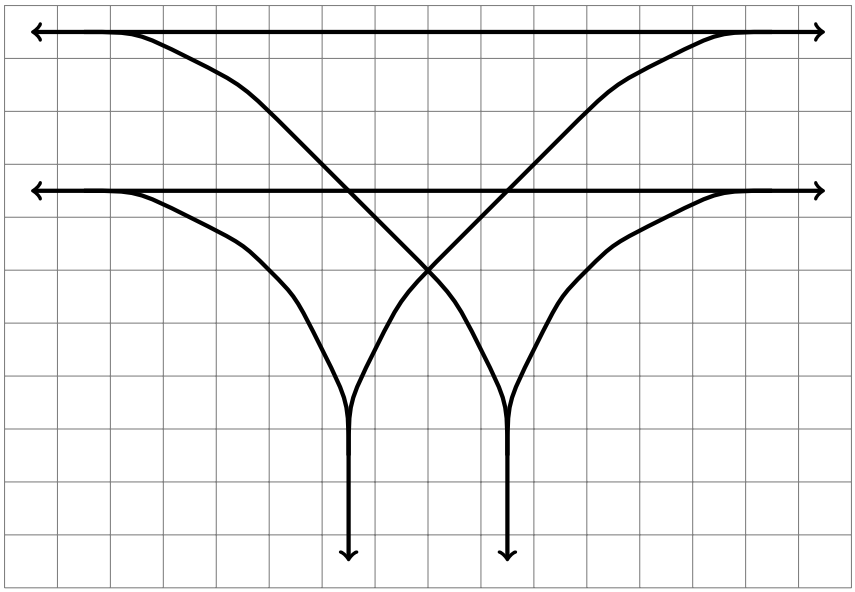
3.10 Considerations for interlocking

The previous sections were mainly theoretical in that the sections mostly introduced new concepts or described how to do things. This section will focus on the practical part of interlocking, in particular certain things to consider when setting up interlocking on a rail line.

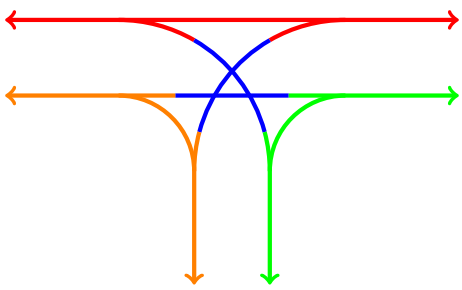
3.10.1 Junctions

One of the most important considerations when setting up interlocking at a junction is to make sure that multiple trains can go through the junction at the same time when possible.

As an exercise in section 1.6, you were asked to build a T junction. An example is shown below if you are stuck.[4] Try to interlock the junction based on what you have learned in the last few sections.



The lazy method would be to set up the entire junction as a single track section - you only need 6 TCBs for that. However, a train passing through would occupy the entire junction, even when two trains could use the junction at the same time - for example, a train going from east to west and another one from west to east. This can be solved by a slightly more complicated setup with four sections, each shown below with a different color:



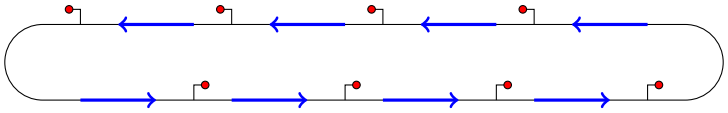
It is also possible to build the junction on multiple levels. This is known as grade separation. Note that grade-separated junctions may appear less realistic because slopes are significantly steeper than their real-life counterparts.

As an exercise, you can try to make the T junction grade-separated.

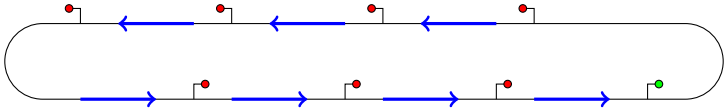
[4] Please note that, as in the rest of the manual, the curves in the diagrams only approximate their counterparts in the game. It is up to the reader to figure out which track to use.

3.10.2 Track capacity and deadlock

You can not infinitely add trains to a line. If you do, you will end up with a deadlock, where every train is stuck at a red light, waiting for the previous train to clear the track section ahead, while the previous train is also stuck at a red light, waiting for the train ahead of it to leave the section ahead, and so on:



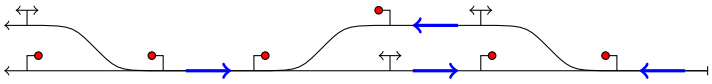
This problem can be trivially solved by removing a train and, if necessary, resetting the track section that the train previously occupied (this can be done in the track section formspec):



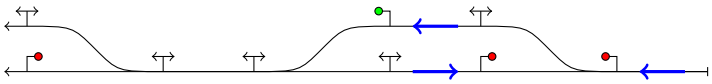
However, as you may notice, only the train at the bottom-right can move; other trains have to wait for the trains ahead to leave the section. The ideal situation is that the section length and the number of trains are decided in such a way that trains do not have to stop or slow down between stations.

When actually building a rail line, signals are spaced away much further than in the illustration above. On servers, the distance between signals typically range between about 50 nodes to a few hundred, depending on the speed at which trains run on the line and the number of trains using the line. The distance between signals in station areas usually depend on the size of the station.

Single-track sections can also be a source of deadlocks:



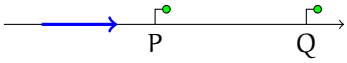
With single-track sections, routes should generally be set up in a way that the train does not stop in single track sections. Exceptions include the end of track, where the train has to stop and reverse:



Notice that the deadlock is prevented at the cost of lower capacity of the line.

3.10.3 Short routes

In some cases, there may be setups with very short routes:

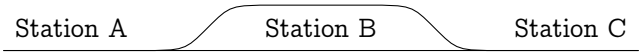


In the setup illustrated above, if $\overline{PQ}$ is short enough, or, more specifically, if $\overline{PQ}$ is shorter than the braking distance of the train at P, the signal at Q will be triggered to set a route.

If short routes are desired, such as at stations, it is generally recommended to disable ARS for the train, such as by using the `A0` ATC command, and keep some distance between the influence point of the signal and the point at which the train is expected to stop. Alternatively, you can also limit the speed of the train to trigger the routesetting system later, but this is not the ideal solution in most use cases of short routes.

3.11 Basic three-station setup

Now that you have learned to use interlocking, create a simple three-station setup shown below.



If you are stuck at some point or want something to compare your set up to, you can watch Blockhead's introduction video, where he creates a three-station setup. The link is provided at the beginning of this chapter.

4 Signals

Signals are devices that give trains certain instructions and/or impose certain limitations on trains. ADVTRAINS comes with a few signals, which are explained in the corresponding sections.

4.1 Types of speed restrictions

There are a few types of speed restrictions that a signal can give. These speed restrictions are independent of each other, and the strictest restriction is chosen as the speed restriction of the train. The speed restrictions need to be set by the specific signs that do so - every signal can only set (and lift) one type of speed restriction and do not affect other speed restrictions. The naming of the speed restriction types are mainly aesthetic and does not give any particular information for ADVTRAINS - in real life, these are mainly informative for the driver.

The following types of speed restrictions are used by signals in ADVTRAINS:

Permanent ("main") speed restriction (`main`) term

The default type of speed restriction. This one is most commonly given by signal lights.

Temporary speed restriction (`temp`) term

Speed restrictions that are temporarily set up, such as near construction sites.

Line speed restriction (`line`) term

Speed restriction that applies to the entire rail line.

The following is an example of a train running with speed restrictions. You may want to read the section on German signal signs first.

- The train starts without any speed restriction.
- The train drives past a Lf 1/2 sign with a speed restriction of 8. The train now has a speed restriction of 8.
- The train drives past a Ks 1 with the Zs 3 indicator showing 12. The train now has a main speed restriction of 12 and a temporary speed restriction of 8, so the train still has a speed restriction of 8.
- The train drives past a Lf 3 sign. The temporary speed restriction is lifted, and the train now has a speed restriction of 12, set by the main speed restriction (described above).
- The train drives past a Zs 10 sign. The main speed restriction is lifted, and the train now has no speed restriction.

4.2 Basic signals

The core mod of ADVTRAINS comes with wallmounted signals and a signal with two lights. These signals can only stop the train and lift the main speed restriction of the train.

4.3 Demo signals

The demo signals are provided by the `advtrains_interlocking` mod. It is similar to the basic signals described above but have three states instead of two.

Danger signal aspect

Red light: Stop.

Free signal aspect

Green light: Proceed at maximum speed.

Slow signal aspect

Yellow light: Proceed with the speed limit of 6m/s.

Demo signals do not allow trains in shunt mode to pass through.

4.4 German signals

The `advtrains_signals_ks` mod provides a set of signals used in Germany, including (but not limited to) Ks signals that the mod gets its name from.

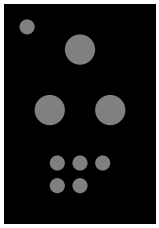
Due to historical reasons, the phrases *ex-DB* and *ex-DR* refer to the former Deutsche Bundesbahn (West Germany) and Deutsche Reichsbahn (East Germany) respectively.

The signal placement conventions assume right-hand traffic.

4.4.1 Ks signals

Ks signals are combined signals that combines the features of main signals and distant signals. The system was developed by the Deutsche Bahn AG after the Deutsche Bundesbahn in West Germany and the Deutsche Reichsbahn in East Germany merged, and the goal was to replace the other four signaling system in use in Germany, which still exist today.

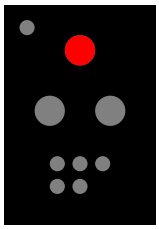
A real-life Ks signal looks similar to this:



Ks signals in ADVTRAINS have the following aspects:

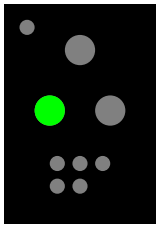
Hp 0 signal aspect

Red light: Stop.



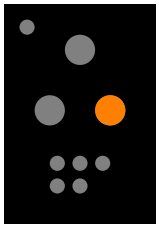
Ks 1 signal aspect

Green light: Proceed at maximum speed or with the speed limit shown on the Zs 3 indicator on the main signal (if present) and expect to pass the next main signal at maximum speed or, if the green light is flashing, with the speed limit shown on the Zs 3v indicator directly below the main signal.

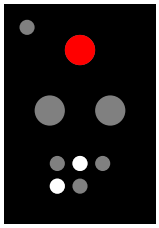


Ks 2 signal aspect

Yellow light: Proceed at maximum speed or with the speed limit shown on the Zs 3 indicator on the main signal (if present) and expect to stop at the next main signal.



In addition, Sh 1 (see below) may also appear along with Hp 0:

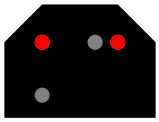


In this case, the train proceeds in shunt mode.

4.4.2 Shunt signals

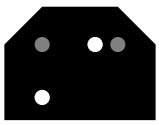
Sh 0 signal aspect

Two horizontally aligned red lights: Stop.



Sh 1/(ex-DR) Ra 12 signal aspect

Two white lights aligned on a slanted line: Shunting allowed.



4.4.3 Signal signs

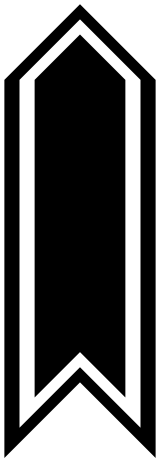
ADVTRAINS has a few types of signal signs, which can be placed by the corresponding placers. The aspect of the signal sign can be adjusted with the trackworker, but only in the range limited by the placer.

Zs 3 signal aspect

White number on a black background: Proceed with the main speed restriction shown on the sign.

Zs 10 signal aspect

An upward-pointing arrow: The speed restriction set by Zs 3 and regular signals is lifted.

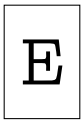


(ex-DR) Lf 1/2 signal aspect

Black number on an orange background: Proceed with the temporary speed restriction shown on the sign.

Lf 3 signal aspect

Black “E” on a white background: The temporary speed restriction set by Lf 1/2 is lifted.



Lf 7 signal aspect

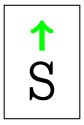
Black number on a white background: Proceed with the line speed restriction shown on the sign.

Ra 10 signal aspect

The text “Halt für Rangierfahrten” on a white semicircle: Do not proceed if the train is in shunt mode. This signal is conventionally placed on the left side of the track.

PAM signal aspect

The text “S” below a green arrow: Proceed without shunt mode. This sign does not have a real-life counterpart.



4.4.4 Differences from real-life signals

The signal book at `https://www.bahnstatistik.de/Signalbuecher/SB-DBAG_2006.pdf` is used for reference. This list is not comprehensive.

- The speed is indicated in m/s instead of multiples of 10km/h.
- Certain visual effects, such as making signal signs reflective or lit at night, are not implemented.
- Distant signaling is not implemented.
- The location of most signals are not checked. The location of the mounted Zs 3(v) indicators are checked relative to the main signal.
- The shunt signals described in this chapter are actually called *Schutzsignale*. The word *Rangiersignale* refers to a different set of signals (including acoustic signals) given by the person specifically responsible for train shunting.
- The ex-DB definition of Sh 1 is that the track section ahead is clear and does not imply that the driver is allowed to proceed. The driver is expected to ask for permission before proceeding.
- Distant signal signs (Zs 3v, Lf 6) are currently not available.
- The Lf 1/2 signal is no longer set up in new rail lines in real life. Lf 1 (expect temporary speed restriction) and Lf 2 (beginning of temporary speed restriction) are used instead.

5 LuaATC

LuaATC (sometimes also called *atlatc* for *Advtrains Lua ATC*) offers the ability to (among other things) use tracks to programmatically control trains. It is provided by the `advtrains_luaautomation` mod. The documentation can be found here: `https://git.bananach.space/advtrains.git/tree/advtrains_luaautomation/README.md`.

Players need the `atlatc` privilege to perform any operation with this mod (except punching operation panels to trigger an event). The Lua environment is only sandboxed with Lua's `pcall` and is therefore not protected against infinite loops or fork bombs. It is therefore advised to grant this privilege with care.

Here are some general recommendations when using LuaATC:

- Avoid using `while true`, as infinite loops effectively stall the server. Consider using numeric for loops with `breaks` instead so that the number of iterations is limited.
- Avoid using `interrupt`, as this can result in a fork bomb if multiple interrupts are queued at a time. Consider using `interrupt_safe` instead.
- Regular interrupts are based on ADVTRAINS `dtime`, which is not necessarily synced with real-life time, especially on servers with significant lag. Consider using `schedule` or `schedule_in` instead and configuring the `advtrains_lines_rwt_realtime` setting in `minetest.conf` to be in sync with real-life time if you want to schedule events based on real-life time. Note that the `advtrains_line_automation` mod needs to be enabled if you want to use the railway time API.

6 Contributing

6.1 Contributing code

At the time of writing, ADVTRAINS uses a mostly email-based workflow. It is therefore recommended to format patches with `git-format-patch` and send them with `git-send-email`. Alternatively, if you are unable to send patches via email, you can also post patches onto the forum thread.

When sending patches, please include binary files as well instead of linking those to external sources.

ADVTRAINS uses `busted`[5] and `mineunit`[6] for unittesting. Please include unittests along with your changes if possible.

If you want to make significant changes to ADVTRAINS, it is recommended to first discuss your proposal with the developers or, at least, with people who have sufficient knowledge of the part of ADVTRAINS that you want to work on.

6.2 Localization

Despite using Minetest's client-side translation system, the `l10n` branch manages locale data as PO files, which are then converted to Minetest's locale files at init time. This allows translators to use existing tools, such as POEdit, to work on translation files.

Translation files can be found in the `advtrains/po` directory of the `l10n` branch.

Please note that PO files should be named in the format of $\langle xx \rangle$`.po`, where $\langle xx \rangle$ stands for the language code of the target language. Minetest does not support regional variants for most target languages, so please do not append the country code to the name of the PO file.

A notable exception to the above rule is Chinese, which has `zh_TW` for the traditional variant and `zh_CN` for the simplified variant.

When translating, please keep in mind that

- Translations should be consistent. You can look at other entries or the translation in Minetest as a reference. If you know a third language (other than English and the target language), you can also use the translation file for that language as a reference.
- Translations do not have to fully correspond to the original text — they only need to provide the same information. In particular, translations do not have to have the same linguistical structure as the original text.
- Abbreviations or names may or may not need to be translated. This depends on certain aspects of the target language, such as (in particular) the script used for the written form.
- Native speakers are generally preferred. For non-native speakers, the CEFR level B2 is recommended as a reference. This is, of course, not mandatory.

[5] `https://olivinelabs.com/busted/`
[6] `https://github.com/S-S-X/mineunit`

A Physics

This section is mainly intended as a reference that is provided for convenience.

A.1 Movement

$$v = \int a \, dt$$
$$x = \int v \, dt$$

A.1.1 Constant acceleration

$$v = v_0 + at$$
$$x = x_0 + v_0 t + \frac{1}{2} a t^2$$

In certain cases, the starting velocity v_0 and the target velocity v_1 are known:

$$t = \frac{v_1 - v_0}{a}$$
$$s = \frac{v_1^2 - v_0^2}{2a}$$

A.1.2 Acceleration of a train

The acceleration of a train is calculate as follows:

$$a = a_{\text{all}} + a_{\text{locomotive}} \cdot \frac{n_{\text{locomotives}}}{n_{\text{wagons}}}$$

Please note that slopes are not taken into consideration.

A.1.3 Acceleration constants

Lever	a_{all}	$a_{\text{locomotive}}$
0	-10	0
1	-3	0
2	-0.5	0
3	0	0
4	0.5	1.5

Alphabetical Index