Table of Contents

(CTRL+ALT+A→H for document outline)

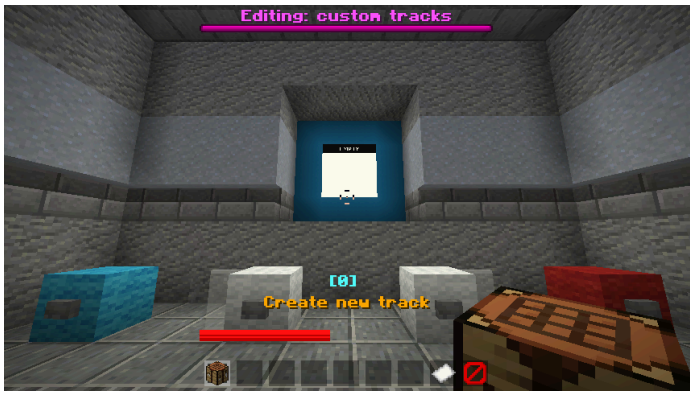
Video guide

ReflectedMantis made a track editor guide that covers basic race track creation with AI pathfinding included.

Check it out here:
Minecraft: Sprint Racer Custom Track Tutorial (v1.4)

Creating and configuring a track

(this section is for v1.1.0 and later)



In the custom track manager, select the crafting table on your hotbar and right click it. A clean, new track will be created.

If you want your creation to be a race track, press the stone button on the light blue wool to make sure you're in the "Race Tracks" list before using the crafting table.

If you want your creation to be a battle arena, press the stone button on the red wool to make sure you're in the "Battle Arenas" list before using the crafting table.

Now, select the lectern on your hotbar titled "Track config" and right click it. Here you'll see all important settings for your track. This will look different depending on if your track is a race track or battle track. Most important here are **Track Name** and **Coordinates**.

For **Track Name**, click the [CHANGE] button in the text window and a command will be suggested to you. Replace TRACKNAME with whatever text you want (spaces are fine here). Press ENTER to run the command and rename the track.

By default your track's coordinates are set to x=0 z=0. For **Coordinates**, click [SET COORDS] and you'll be able to fly to wherever you want in the world. Right click on the structure block in the middle of your hotbar to set the track location to where you are. Right click the barrier to return once you're done setting the coordinates.

Making nodes

Nodes?

Nodes are what make your track functional. These are your checkpoints, finish lines, speed boosts, item chests, AI waypoints, player start positions, etc. These come from spawn eggs that you can place while in Editor Mode.

Editor Mode basic function

To enter Editor Mode, drop the gray wool in your first inventory slot (you must be in Creative Mode to do this). Some tools will appear on your hotbar in the rightmost 5 slots. The three empty slots on your hotbar are free, you can put whatever you want in them.

Open your inventory and go to your “Survival Inventory”. Here you will see spawn eggs for every node. Drag them to your hotbar to start using them! If you lose some, don’t worry-- just exit and re-enter Editor Mode and all spawn eggs will be replenished.

Simply use a spawn egg on the ground to create a new node. Most nodes can also be configured before placing if you drop them.

Drop nodes for config!

If a node’s spawn egg says “Drop for Config”, simply press your drop key (Q by default) while holding the item and a text menu will be brought up. Always check out the config menu before placing a node you’re unfamiliar with!

Player Starts, Item Chests, Checkpoints, Finish Lines, Scripted Sequences, Teleporters, and AI Breadcrumbs have very important config menus-- be sure to use them!

Activators (IMPORTANT!)

The most important node is the “Activator Node”, the first egg in your inventory. Without this node, no other nodes in your track will function at all.

Be sure to put at least one Activator node within 100m of where you set your track's coordinates to. Upon loading the track, the Activator nearest to the coordinates will be enabled.

All nodes must be within 100m of an Activator node in order to function. This prevents other tracks from running any game logic if they happen to be in loaded chunks. If you need to cover a larger area, you can chain Activators together (Activators enable other Activators!). Just make sure you don't connect to another track's Activator by accident!

Node eraser

Deploy the creeper spawn egg in your inventory on the ground and it will delete all nodes within a small radius.

If you use your eraser while nodes are selected, only the selected nodes will be deleted. It will delete all selected nodes regardless of how far away they are. Nifty!

Selecting nodes

Throwing the snowball on your hotbar will select the nearest unselected node. More operations will become available on your hotbar once you do this.

Some nodes have a **face direction** which is indicated by particles above the node. This can be changed using the tools on your hotbar.

It's important to set the face direction of player spawns, checkpoints, and finish lines!

Checkpoints, Finish Lines, and position calculation

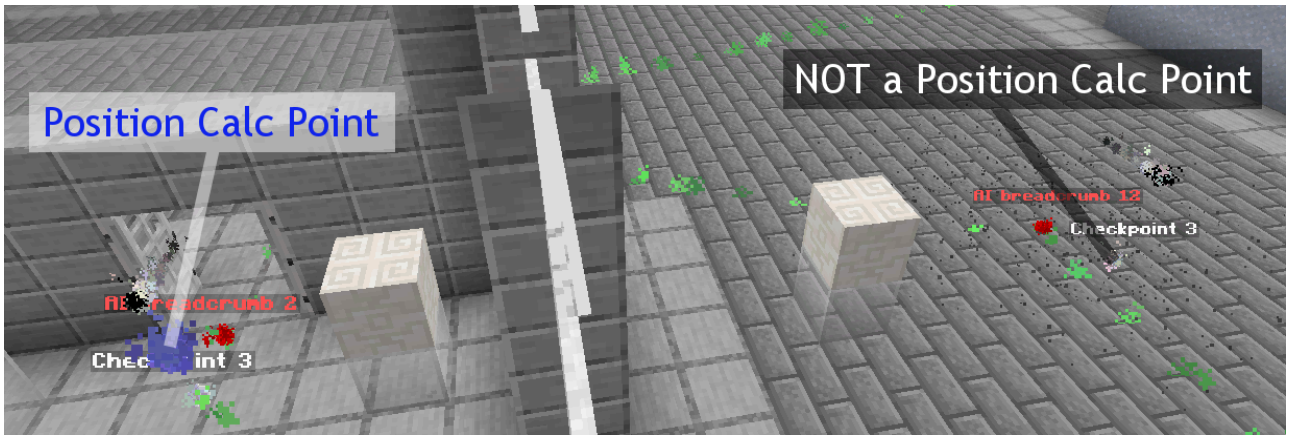
This is only important for race tracks! Checkpoints and finish lines do nothing in battle tracks.

Before placing a node for a checkpoint or your finish line, drop the spawn egg to bring up the config menu. As you might expect, checkpoints go in a sequence of 1, 2, 3, 4, 5, 6, etc. To meet the track's checkpoint requirement and complete a lap at the finish line, the player must enter the hitbox of each checkpoint *in the proper order*.

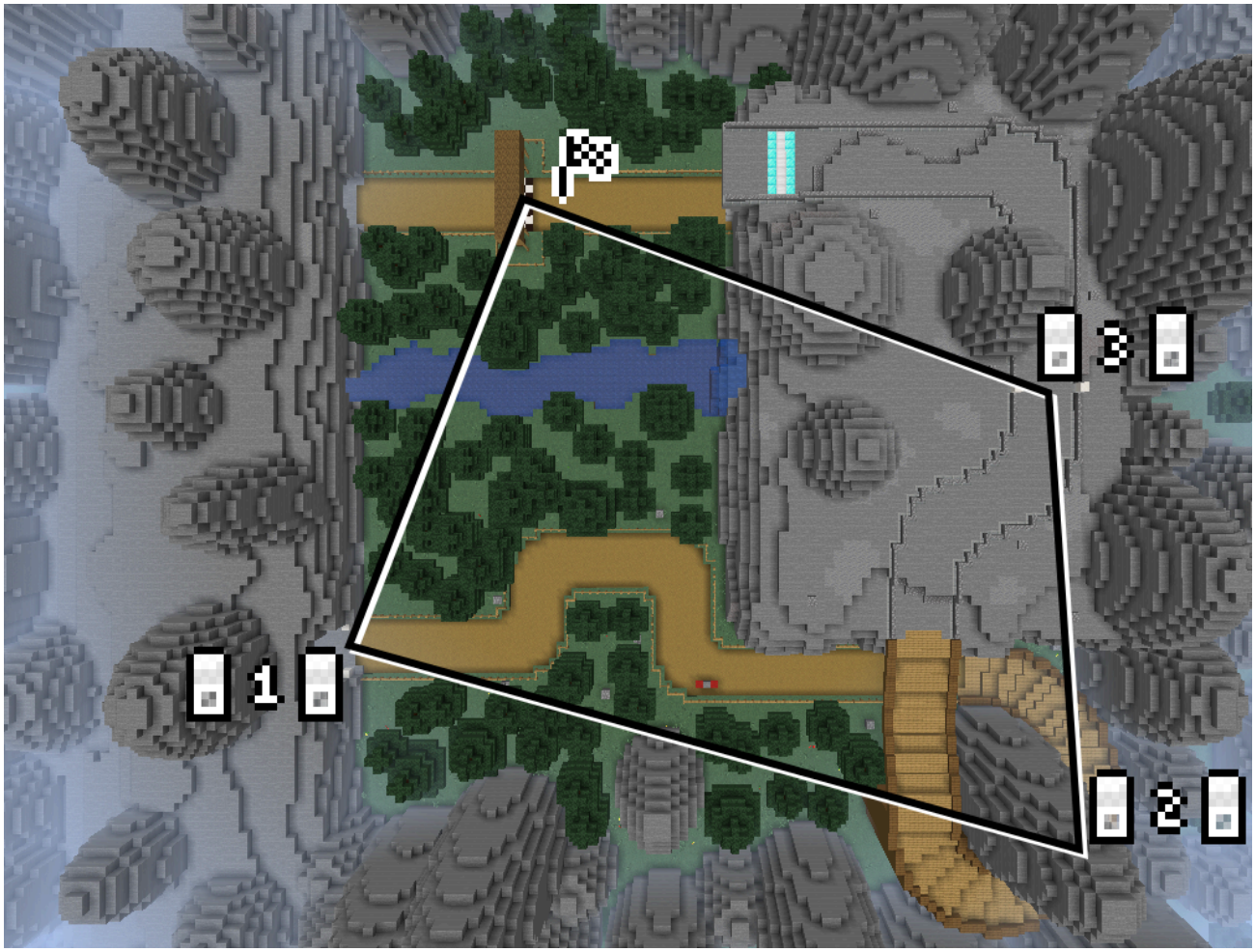
Checkpoints and your finish line are responsible for position calculation. This is how the game knows who is in 1st, 2nd, 3rd, etc. **Note the setting: Position Calc Point? [Yes] [No]**. For each number of checkpoint & the finish line, you must set ONE of the nodes to be a position calc point (only ONE pos calc node for checkpoint 1, ONE for checkpoint 2, ONE for each following checkpoint, then ONE for the finish line).

Note that the finish line behaves pretty much the same as a checkpoint, except it can send the player within its trigger area to the next lap... if they've met the checkpoint requirements!

Position Calculating nodes are marked via a blue cloud, as shown here:



Players' positions are based on their distance to the next checkpoint or finish line node (only those marked as "Pos Calc" points). As an example, this is how the position calculator sees River Valley. Note that it doesn't care what the actual shape of the track is-- it only cares where the position calculating Checkpoint and Finish Line nodes are located.



Scripted sequences

Some tracks have sequences that take away player control for a short time and force the player to move a certain way. Examples include the loop-de-loop in Laeto Forest and the cannon in Miner's Island. These are done via the Scripted Sequence node.

Sequences need a **"start"** and an **"end"** node. When the player is near the start node, they are pulled into the sequence. When they hit the end node, they regain regular control. Use the config menu to specify your start and end points.

While in the sequence, players rapidly teleport to the next nearest node in the sequence. The pattern goes like this: 1 → 2 → 3 → 4 → 1 → 2 → 3 → 4 → etc.

Also note that every single node in the sequence has a face direction. This is the direction the player is forced to face when they teleport to that particular node in the sequence. Best course of action is to select all of them at once (after placement) and then set the direction using the dispenser tool on your hotbar… assuming you want them all to face the same way.

Quick edits for Item Chests, AI Breadcrumbs, and Checkpoints

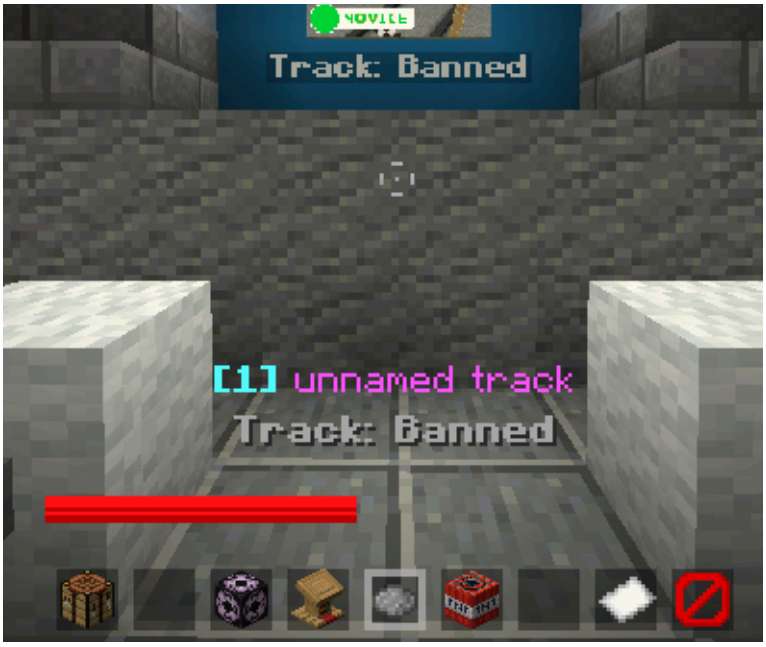
If you go near an Item Chest, Checkpoint, Finish Line, or AI Breadcrumb node, you'll see some special tools (colored dye) appear on your hotbar. Drop them and you'll be able to edit the properties of the nearest Item Chest or AI Breadcrumb node. This is very useful as these nodes often need to be edited *after* they've been placed.

Enabling your track

(this section is for v1.1.0 and later)

In the custom track manager, select the dye in the middle of your hotbar and right click it.

Your track will be toggled between being “Banned” and “Allowed”.



What does “Allowed” mean?

> When allowed, your track will be eligible to appear randomly. This means it can be a voting option in the lobby if Sprint Racer is set to [VOTE] or [ROULETTE] mode, or it can be selected randomly if Sprint Racer is set to [RANDOM] mode.
>
> Note that custom tracks do not appear in [IN ORDER] mode. This mode plays the stock tracks only!
>
> If using [CHOOSE] mode, you can select and play the custom track regardless if it’s tagged as allowed or not.

AI pathfinding

Out of all the nodes, AI Breadcrumbs are the scariest and most complicated by far. In order to use them effectively, you'll need to know how the AI pathfinding system works, as there are some hidden nuances that influence the decision-making of the AI.

In a nutshell...

The AI has no way of learning from its environment or generating navigation paths, it is entirely dependent on user-placed waypoints. The AI moves in a straight line from point-to-point in order to navigate. These points are the AI breadcrumb nodes you can place using Editor Mode. Each breadcrumb has directions that the AI will use to find the next breadcrumb. It's a cycle!

While in Editor Mode, particle trails are drawn that show the connections between nodes and the direction they move in. Pay attention to these!

In v1.2 and prior: there are only 25 possible ID values for AI breadcrumbs. This means there will probably have to be multiple with the same ID-- this is okay. When the AI is told to go to breadcrumb 1, for instance, it will always move toward the *nearest* breadcrumb with an ID of 1. Note that when the AI moves from breadcrumb to breadcrumb, it only cares about the breadcrumb it is chasing. Moving through an unrelated breadcrumb will not cause the AI to re-route.

In v1.3 and later: there are 625 possible ID values for AI breadcrumbs. It's recommended you take full advantage of this and give every breadcrumb a unique ID. This will avoid headaches down the road, especially if you're making a battle map!

Also in v1.3 and later: You have a tool called the “Breadcrumb Connector”. This is immensely useful for quickly drawing paths without having to fiddle around in the edit panel for every single node. It can be used to set any of the 4 directions of the Breadcrumb, and can also be used to set the special 5th direction if an applicable event is being used.

Direction bias / probability

AI zombies will depart in one of four directions from each node they visit (each direction is assigned a value that dictates what # breadcrumb the AI will chase). There is a set probability for each of the relative four directions. The AI favors the direction it is already traveling when choosing a new direction. Think of the four directions as a compass:

Light Green
Dark Blue -|- Light Blue
Dark Green

Here are the probabilities for movement directions:
8/13 - Keep going in the direction the AI previously took
2/13 - Take the direction adjacent to the previous direction (clockwise)
2/13 - Take the direction adjacent to the previous direction (counter-clockwise)
1/13 - Take the polar opposite direction (unless the AI recently deployed a trap)
NOTE: If a direction is defined as 0, it will NEVER choose that direction when departing.

You can mix and match combinations of directions to produce various pathing probabilities. It is also perfectly fine for multiple directions to point to the same destination.

Recommended design approach

If you're making AI navigation for a race map, you can get away with using just 1 direction in most situations. The lower probabilities for switched directions can be used to create alternate paths that are less frequently taken.

If you're making AI navigation for a battle map, a good strategy is to treat your 4 directions like a compass. For example, all north-bound routes can be light green, all east-bound routes can be light blue, all south-bound routes can be dark green, and all west-bound routes can be dark blue. Unlike in race mode, your connections between nodes should probably be two-way instead of one-way. This is how all of the stock battle tracks handle AI navigation.

Notable quirks and nuances

Once the AI gets within 2 meters of the breadcrumb it's chasing, it will then move on to the next breadcrumb. This means that the particle trails aren't always perfectly accurate-- the AI will typically take a turn one block before the particles suggest it will. This can have consequences!

When the AI spawns at a Player Start node, respawns at a checkpoint after a death, or teleports to a Teleport Destination node, it will automatically target the breadcrumb closest to itself. Keep this in mind when placing breadcrumbs near checkpoints and the finish line.

When near an item chest, the AI will veer off course in order to get the pickup, then return to chasing whatever breadcrumb it was going after. There are some instances where this might get the AI stuck, like if it tries to grab a chest on the other side of a wall. The AI will get bored and move on if it cannot reach a chest, but this still eats up a lot of time. In the config menu for Item Chest nodes, there is an option to make the chest invisible to the AI.

If the AI gets stuck, it will first attempt to jump over whatever is blocking its path. If this fails, it will reset to the last checkpoint it hit. In battle mode, the AI will give up on whatever node it's currently chasing and go after a different one nearby (randomly chosen). Note: just like players, the AI is not allowed to reset in battle mode!

The AI can only function while in loaded chunks. If the AI enters an unloaded chunk, it will force itself to respawn and the old entity that entered an unloaded chunk will be deleted when its chunk is reloaded. This can cause problems on large tracks where areas of the track might not be loaded all the time. In that case, you might consider disabling the AI altogether.

In battle mode, when the AI gets near an opponent, it will forget about the breadcrumbs and chase down whatever opponent it wants to fight. After defeating the opponent, getting bored, or getting scared and running away, the AI will return to following breadcrumbs starting with the breadcrumb nearest to itself.

As of v1.3, AI can now follow breadcrumbs while flying with elytra. Keep in mind that when making turns mid-air, the previous momentum from the elytra will affect the flight path.

Events

Events are executed the instant the AI reaches the breadcrumb with the event. There are many different events that can inspire complicated behaviors such as making precise jumps, using ender pearls to take a shortcut, using elytra, taking a general shortcut, etc.

The “Force redirect nearby AI” event is very useful for re-routing an AI that failed a jump or got stuck. Note that touching an untargeted breadcrumb normally does not cause a re-route.

Random spread

In order to make AI movements more organic, AI breadcrumbs can be given a “spread”. This means that when the AI targets and moves towards a node with spread, they may chase a point that is offset by some amount from the center. The offset is random and can happen in both the x and z directions, with the maximum spread being the value you set.

Be careful about when and where you use an increased spread, as it may sometimes cause the AI to get stuck.

Hook

Changing the hook alters the path the AI takes when traveling to the next breadcrumb. Straight is the default. Curving left or right will cause the AI to take a path in the shape of a rainbow rather than a straight line. This can be used to make the AI take smooth-looking turns in races.

Mid-air elytra path

Nodes marked as “mid-air elytra path” have a special property: they will NOT be used for any situation where the AI would need to automatically target the breadcrumb nearest to itself. This is relevant in situations such as: respawning, resetting to checkpoint, leaving a teleporter, getting stuck in battle mode, getting teleported by an ender pearl or warp item, landing after using elytra, or returning to breadcrumb-following after chasing an opponent in battle mode.

You don’t necessarily have to use the “mid-air elytra path” marker for elytra paths. Sometimes it’s useful to apply it to regular nodes on the ground that you don’t want the AI to target unless explicitly told to. Funnily, this tag doesn’t actually have anything to do with being in the air (it doesn’t require being in the air or using elytra). Elytra pathing is generally just its best application.

Hardcore parkour

If you need the AI to execute a complicated series of jumps, it is helpful to use the “Force Jump” event combined with the “Require OnGround: [Yes]” setting. If you don’t require the OnGround tag, the AI will usually just get within radius of the node and then air-strafe off into the cliff you want it to jump over. The AI needs to be touching the ground in order to initiate a jump!

Performance warning

AI waypoints, like all other nodes, are entities. Having a large number of entities (over several hundred) loaded at the same time will bring Minecraft to its knees. Even if the entities are not being used, they will still tank tick performance. **If your track is really long or has a super complicated layout, you may have to give up on AI support.**

Two particularly large stock tracks, Hurricane Islands and The Gauntlet, do NOT support AI due to entity lag concerns. Other stock tracks like Redstone Factory and Scorched Sands DO support AI, but they’re noticeably worse on performance than other tracks due to the high number of nodes. Dead Canyon is a small track, but suffers from similar performance issues due to being really close to other tracks (more entities are loaded at once). Be aware of these issues! If you want AI on your track, try to plan better than I did.

Sprint Racer | Custom Track Guide

Navigation direction map

The “Wrong way!” and “You’re off course!” texts on the stock tracks aren’t just magic, there’s something going on under the hood (literally under the track).

If you’ve flown below any of the stock tracks in spectator mode, you probably already recognize it. There’s a sheet of wool at y=0 with different color patterns that match the shape of the track. This is a navigation map that can tell when players are going the wrong way, going off course, or missing a checkpoint. It is also the source of the navigation arrow that is shown when players choose to use the navigator.

Basic function

All active players check the block at y=0 directly below themselves. The color of the wool block found determines how the navigator reacts.

In race mode:
Lime Wool: Player should be going ***north***. If facing south, say “Wrong Way!”
Light Blue Wool: Player should be going ***east***. If facing west, say “Wrong Way!”
Green Wool: Player should be going ***south***. If facing north, say “Wrong Way!”
Blue Wool: Player should be going ***west***. If facing east, say “Wrong Way!”
Red Wool: Off course!
Light Gray Wool: Missed a checkpoint! *(see checkpoint overrides below)*
Black Wool: Force player to reset (also kills thrown ender pearls)
Yellow Wool: Nothing (actionbar navigator will display a “?”)

In battle mode:
Red Wool: Off course! The player is given a few seconds to return to safety, otherwise they die.
Black Wool: Kill player (force respawn) (also kills ender pearls)
Yellow Wool: Nothing

Layers / checkpoint overrides (race mode only)

In addition to checking y=0, players also check y=1, 2, 3, or etc depending on what checkpoint they’re on. So if a player is going for checkpoint 1, they check y=1. If a player is going for checkpoint 2, they check y=2 (not y=1 anymore). So on…

The block players find at y=1 or higher overrides the one found at y=0.

Custom preview image

In the custom track config menu, there's a setting that allows you to set the "Preview Image". There are 50 dedicated map.dat preview images that you can overwrite within the world file's "data" folder.

In v1.6 and later, you can use any # map.dat file, so it's recommended you create a new one instead of overwriting one of the built-in 50 map files. With that, you'll avoid file conflicts with other custom tracks.

Here are some resources for creating preview images similar to the stock tracks:
https://drive.google.com/open?id=10gq4M6HGX_vxnOSPH7X1tq_qcLn4ZEky
GIMP 2 (a free program) is needed to use the included template. Sample images are also included if you want to try and emulate them with another program.

To export your image to a map.dat file, you can use MC Map Item Tool.
https://mc-map.djfun.de/
Be sure to give the file the correct number as its name! (ex: 11001, 11002, 11050)

To get the image in the game, simply drag your newly exported map.dat file into the world file's "data" folder. You'll need to restart the world to make your changes show up.

Moving builds between worlds

Bringing in a build or track from another world is tricky in a post-1.13 Minecraft, but it's possible! Here are 3 different ways to get it done:

A. (recommended) Use the Axiom mod. It does a great job pasting larger builds and doesn't corrupt world data.

B. Copy the region files (and, optionally, entities files) into your world. You may need to rename some files to move the build into the region you want it in. Press f3 to see what region files something is in. Make sure you're not replacing an existing region!

C. Use WorldEdit.

1. Install WorldEdit on your server as a plugin. Don't use the forge mod! The mod will almost certainly corrupt your scoreboard data and make parts of the map unplayable.
2. It's recommended you dedicate extra ram to minecraft to prevent crashes.
3. Select your build and save it as a schematic using //schematic save <name>.
4. Load your Sprint Racer world. Go to where you want the track pasted and type //schematic load <name>. Note that when you paste the schematic, it retains the relativity from your position when you copied it. If the schematic is over a few million blocks, it might be a good idea to split it into smaller schematics.

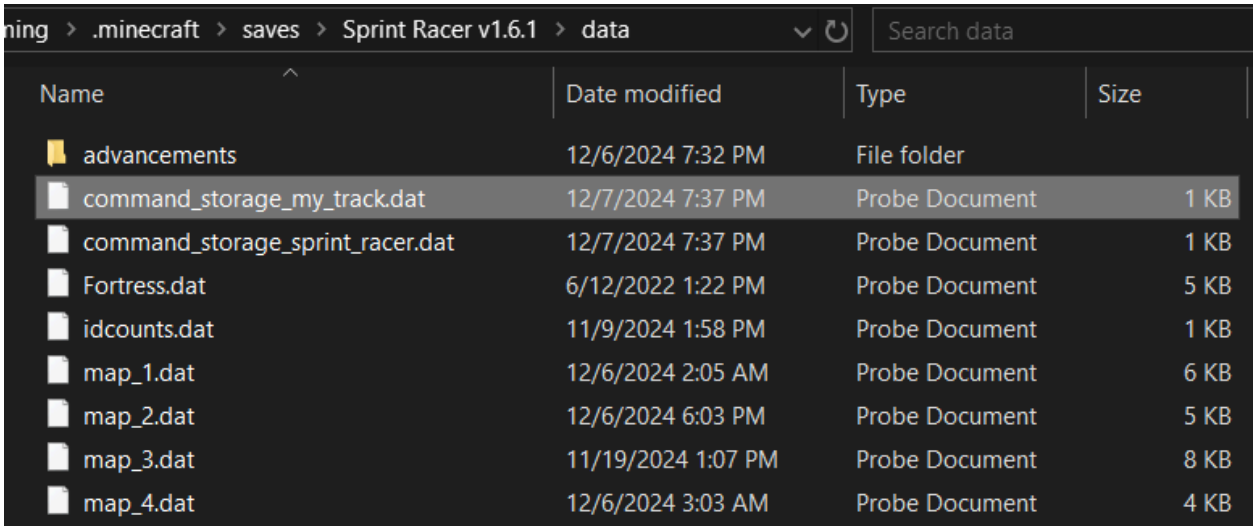
Exporting and sharing your track (v1.6+)

Exporting

In the custom track manager, navigate to the second page and click [EXPORT TRACK].

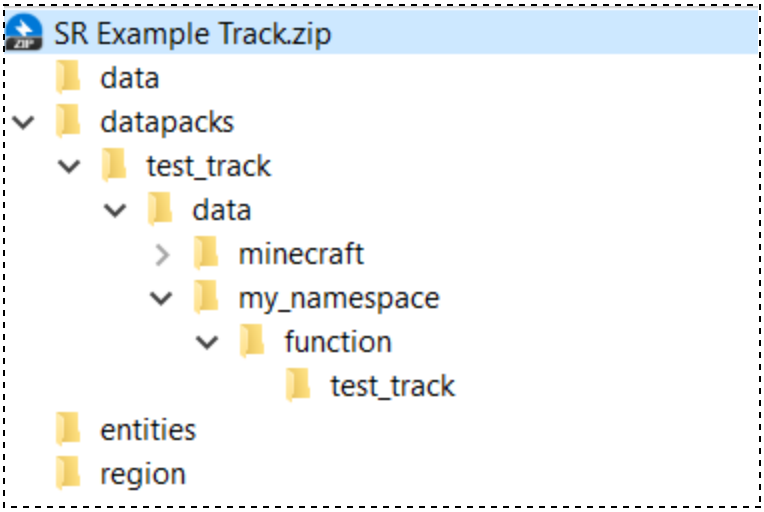
Before running the suggested command, enter a filename. This should probably be the name of your track. The filename must be all lower-case with no spaces.

After running the command, your track's metadata will appear in saves/Sprint Racer/data as a command_storage.dat file. You must close your world for the file to appear. Keep track of this file-- it's required for other people to import your track!

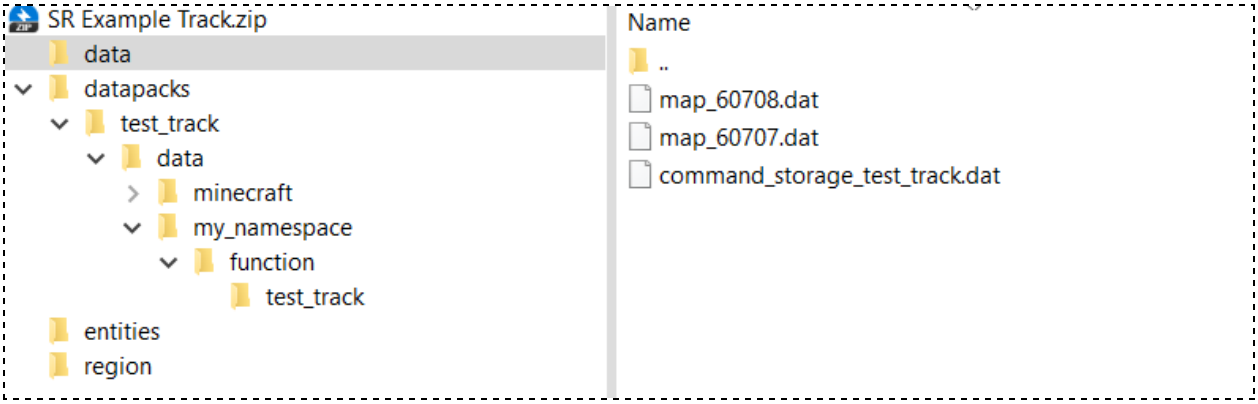


Format (packaging your custom track)

Your custom track should be packaged as a .zip containing only the files needed for your track to work. When people install your custom track, they will extract the .zip onto their Sprint Racer world folder. Here is an example pack, which can be downloaded from https://tinyurl.com/sr-test-track.

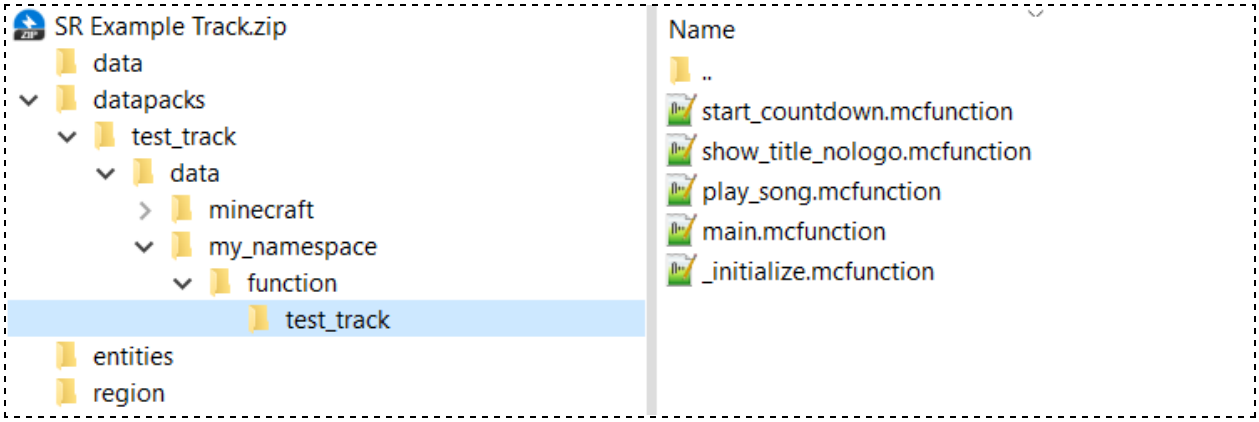


The data folder should contain your track's metadata (command_storage_<track>.dat) and any map.dat files from your world folder that your track needs.



⚠️If you were using one of the built-in blanks previews (such as map_11001.dat), please switch to something else not currently used in Sprint Racer's data folder. This way, you'll avoid file conflicts with other custom tracks.

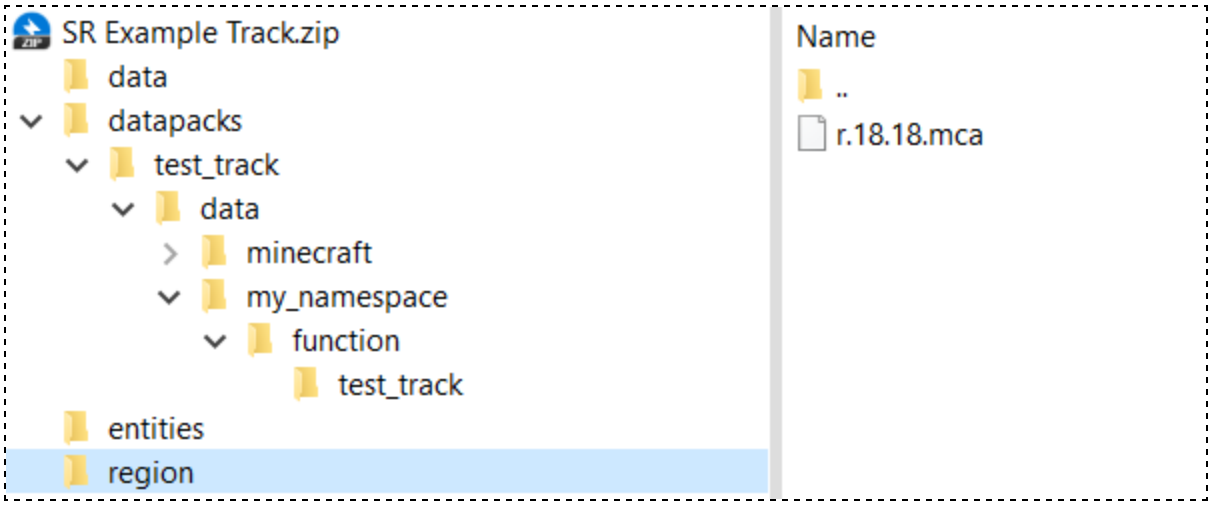
The datapacks folder is optional, and should contain any custom functions you're using.



⚠️If you're using the example pack as a base, please rename "test_track" and "my_namespace" to something original.

⚠️If you were using the namespace "sr_c" for custom functions, you should instead move your functions to your own original namespace. This way, you'll avoid file conflicts with Sprint Racer, as well as other custom tracks.

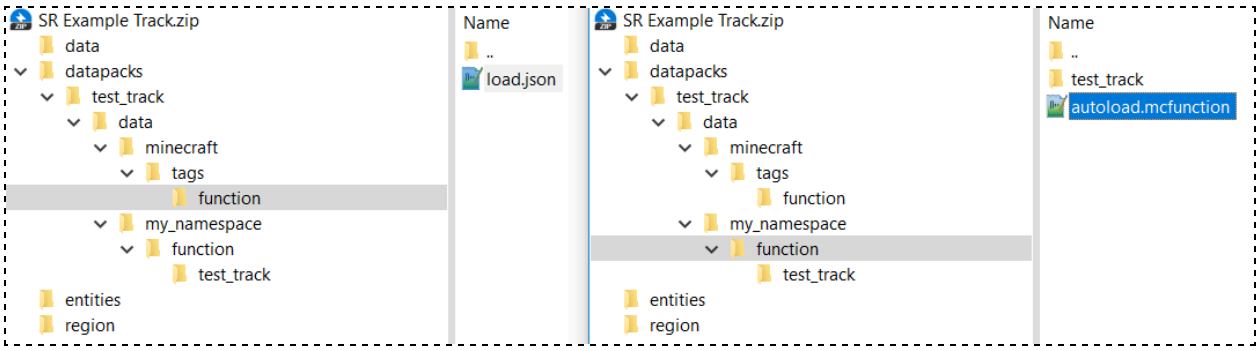
The entities and region folders should contain the corresponding files from your world folder that involve your custom track.



Press F3 to view what region files your custom track occupies. As an example, River Valley is in region r.2.1.mca. If your track is really big, or was built near the border between two regions, you will need to include multiple regions.

Making your track import automatically on world bootup

The provided example custom track file contains two important files inside its datapack: load.json and autoload.mcfunction. You'll need to set both of these up within your own datapack to make your track auto-import.



Step 1: Edit load.json so that it references the autoload function inside your namespace.

```
{
    "values": [
        "my_namespace:autoload"
    ]
}
```

Step 2: Edit autoload.mcfunction so that the namespace used on line 5 is your own namespace. The purpose of this line is to keep delaying the import command from running until the world is loaded and ready to accept a new track.

```
#don't do anything until world spawn is loaded

scoreboard players set #test value 0
execute as @e[tag=w,type=armor_stand] at @s if loaded ~ ~ ~16 if entity @a run scoreboard players set #test value 1
execute if score #test value matches 0 run return run schedule function my_namespace:autoload 10t
#=====

#import track
#(note: this will work once, then doing nothing on subsequent import attempts)
function sprint_racer:game_logic/10/storage/import_track {filename:"test_track"}
```

Step 3: Edit autoload.mcfunction so that the filename argument on the last line is the same filename that you exported your track with. This will make Sprint Racer read from data/command_storage_<filename>.dat

```
#don't do anything until world spawn is loaded

scoreboard players set #test value 0
execute as @e[tag=w,type=armor_stand] at @s if loaded ~ ~ ~16 if entity @a run scoreboard players set #test value 1
execute if score #test value matches 0 run return run schedule function my_namespace:autoload 10t
#=====

#import track
#(note: this will work once, then doing nothing on subsequent import attempts)
function sprint_racer:game_logic/10/storage/import_track {filename:"test_track"}
```

Sharing your track

Feel free to share your custom tracks wherever. Individual tracks are pretty small, so they're easy to share.

Currently, our hub for sharing custom tracks is the ▫ custom-tracks forum on the Team Scripted Discord server. This is the same server that's advertised in the credits area of Sprint Racer. You can join our server using the link https://discord.gg/jbFWf3x. We might make a website at some point, too.

⚠️Before publishing your track, please look at the Sprint Racer Regions & Maps sheet to make sure your regions and map.dat files don't conflict with anyone else's custom tracks. If you do run into a conflict, you can easily rename your region files and map.dat files to something safe. Just remember to update your track config data (and custom functions if you're using them), test your track, and then export it again.

⚠️ If you are publicly publishing a full-on track pack with the entirety of Sprint Racer's files, that's awesome! But please clear it with us first!

⚠️Please remember that your import-able custom track .zip files will only work on Sprint Racer worlds that are v1.6.0 or newer. For reference, that's the version we released when Minecraft 1.21.4 came out.

Hard-coded tracks

https://docs.google.com/document/d/1RZfQCow5IFfFo93sylkCM99Ol5JUmUsjeYhYdIv6W6E/edit?usp=sharing

This document details how to create a custom track *without* the built-in custom track editor (mostly for my own use when I'm officially adding new tracks). This approach allows you to have dedicated functions, track mini-maps, track preview images, and everything else just like the stock tracks.

The custom track manager has gotten more powerful over time, so making tracks this way is super not recommended.

All the nodes and Editor Mode stuff still applies here.