MapEditor Mod – User Guide for Railroader

Version: 1.0.25128.1912
Audience: Experienced Railroader players with at least moderate modding experience, new to this MapEditor mod

---

Introduction

Contents

The **MapEditor mod** for *Railroader* is a work-in-progress but highly capable modding tool that allows real-time editing of railway maps from within the game. It enables direct manipulation of track nodes, segments, trestles, telegraph poles, and scenery objects through an integrated interface that builds on the engine's internal data structures.

This tool is intended for users who are already proficient with Railroader's gameplay and have moderate experience working with mods. If you are comfortable reading JSON, understand basic game file structures, and want to take control of your map at the segment level, this editor unlocks most of the internal track system for live editing.

There is currently no official documentation for this mod. This guide is intended to help users understand the editor's feature set and interaction model by walking through each tool individually. The structure of this guide follows a typical editing workflow: from mod selection to node and segment editing, followed by trestle generation, telegraph pole placement, and scenery manipulation. Relevant details about saving, exporting, and current limitations are included throughout. the editing tools in the order you would typically use them when building or modifying a map. Every button and workflow is described in functional detail, with notes about editor behavior and known limitations.

1. Selecting and Saving Mods

All changes made using the MapEditor are stored in a specific mod, which must be selected before any editing can begin. Internally, this mod is referred to as a **graph**. Without an active mod, no edits can be saved.

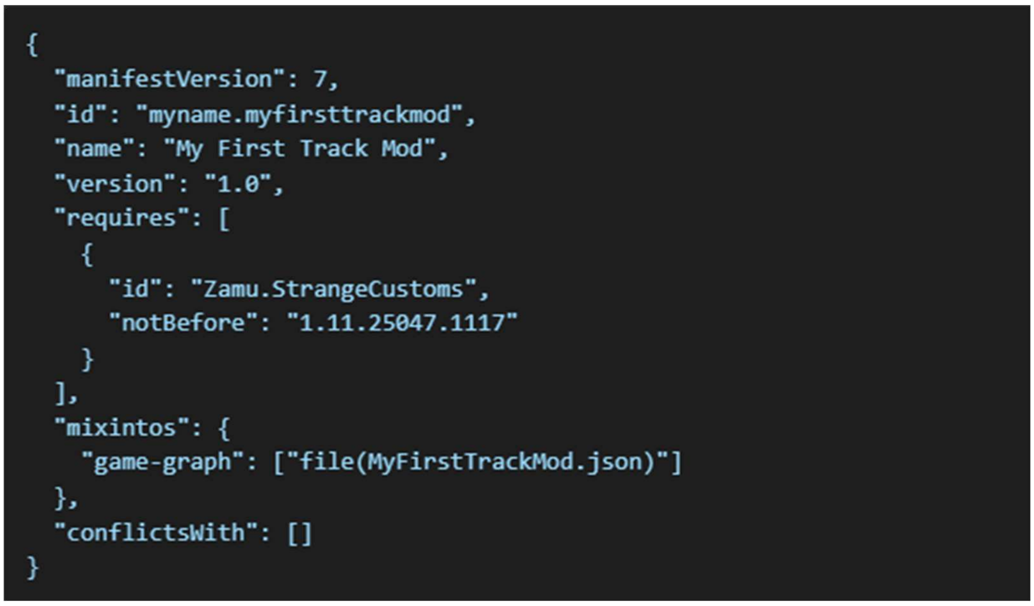
Choosing or Creating a Mod

- Use the **"Select Mod"** dropdown to choose an existing mod. This list is only populated at game startup based on mods that contain a mixintos.game-graph entry.
- To create a new mod:
    - Use the **Create Simple Mod** button in the editor interface (if available).
    - Alternatively, create one manually:
        1. Create a new folder (e.g., MyFirstTrackMod/).
        2. Add a definition.json file with a unique id, a name, and a reference to your patch file in the mixintos.game-graph field.
        3. Add an empty Data/ folder.
        4. Include an empty .json patch file (e.g., MyFirstTrackMod.json) with valid structure for tracks, nodes, and segments.
        5. Zip the folder and place it in your Railroader/mods/ directory.

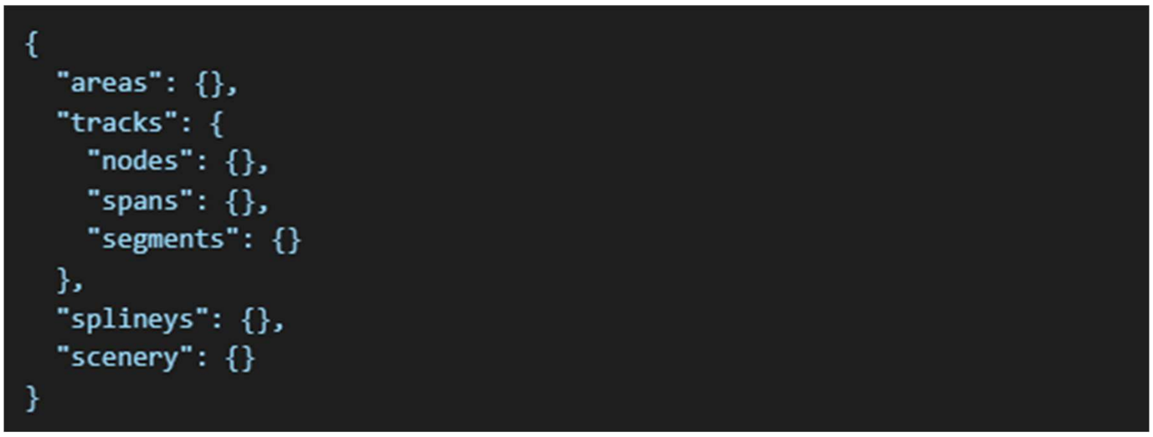
Once a valid mod is in place and selected, the editor can write changes to the specified patch file. Note that newly created mods will **not** appear in the dropdown list until you restart the game. The dropdown is static for the duration of a session.

File Structure Example

A minimal definition.json may look like:



```
{
  "manifestVersion": 7,
  "id": "myname.myfirsttrackmod",
  "name": "My First Track Mod",
  "version": "1.0",
  "requires": [
    {
      "id": "Zamu.StrangeCustoms",
      "notBefore": "1.11.25047.1117"
    }
  ],
  "mixintos": {
    "game-graph": ["file(MyFirstTrackMod.json)"]
  },
  "conflictsWith": []
}
```



```
{
  "areas": {},
  "tracks": {
    "nodes": {},
    "spans": {},
    "segments": {}
  },
  "splineys": {},
  "scenery": {}
}
```

This ensures the editor can load and write edits cleanly.

Runtime Behavior

When the editor opens, it loads the specified patch file into memory using the internal PatchEditor. Any changes are made to this in-memory version until explicitly saved. If the editor is closed without saving, all changes are lost.

The patch is saved only when the user presses **Save Changes**. There is no autosave. Saved data is written to the .json patch file located in the mod's Data/ folder. If this folder is inaccessible (e.g., read-only archive), fallback exports are written to UserData/MapEditor/Exports/.

Tip: Always verify the correct mod is selected before beginning any edit session.

2. Editing Track Nodes

The Track Node Editor enables placement, movement, rotation, duplication, and removal of track nodes. Each node represents a control point in the rail network and determines the shape and connection of track segments.

Available Tools and Buttons

- **Create** – Duplicates the currently selected node and places the new node 2.5 meters to the left, inheriting the source node's rotation.
- **Move** – Allows direct repositioning of a node using the 3D gizmo. This automatically triggers a visual update of any connected segments.
- **Rotate** – Enables rotation gizmos around the X, Y, and Z axes to adjust a node's orientation.
- **Split** – Inserts a new node halfway along the selected segment. This operation maintains continuity and updates the geometry accordingly.
- **Remove** – Deletes the selected node and any segments that originate or terminate at that node.
- **Flip Switch Stand** – Toggles the visible switch stand direction. This affects visual orientation only and is only available when editing a switch node.
- **Show** – Centers the camera on the selected node for quick navigation.

Additional Rotation Features

- **Set Rotation** – Assigns an absolute rotation to the selected node using euler angles (X, Y, Z).
- **Copy Rotation** – Stores the current rotation of a node to the clipboard.
- **Paste Rotation** – Applies the previously copied rotation to another node.

Tips and Behavior

- Holding **Shift** while scrolling vertically increases vertical movement speed from 0.1 to 1.0.
- All user-modified fields in the editor panel are highlighted in **white**. This visual cue helps track what changes are pending.
- All actions performed in the node editor are added to the undo/redo history unless explicitly canceled.

Node creation and editing are non-destructive until saved. You can experiment freely and revert using the undo system.

3. Editing Track Segments

The Track Segment Editor enables creation and modification of segments between nodes. Segments define the visible rails and their physical behavior, such as curvature, speed limits, and associated structures like trestles.

Creating Segments

- Select a node and **Shift + Click** on another node to create a directional segment.
- New segments inherit the type, style, and curvature rules from defaults or previous edits.
- Segments are visualized in yellow when selected.

Editing Segment Properties

Once a segment is selected, the right panel exposes editable properties:

- **Track Type** – Determines how the segment behaves visually and structurally (e.g., bridge, tunnel, siding).
- **Track Style** – Specifies the track mesh and tie layout.
- **Speed Limit** – Maximum allowable train speed, enforced in gameplay.
- **Priority** – Affects how routing decisions are made in dynamic pathfinding.
- **Group ID** – Used for grouping segments (e.g., for switch linkage or logical pairing).

Each field can be modified individually. Any change highlights the respective field in **white** to indicate unsaved modifications.

Geometry Editing Tools

- **Straighten** – Resets the segment's geometry to a straight line between nodes.
- **Move** – Activates a handler that allows dragging the curve in 3D space. This opens a Bézier curve manipulator for direct shape adjustment.
- **Inject Node** – Inserts a new node at the midpoint of the segment. This is useful for fine control or splitting a long segment.
- **Delete** – Removes the segment from the patch. Connected nodes remain untouched.
- **Update Properties** – Commits all pending field changes. Until this button is pressed, any changes remain in memory only.

Trestle Options

If the track type supports elevated construction (e.g., bridges), the following options become available:

- **Trestle Enabled** – A checkbox to toggle trestle generation on or off.
- **Trestle Style** – Defines visual support structure:
  - **Block** – Solid fill trestle
  - **Bent** – Vertical timber or steel bents

Changing trestle options does not immediately generate geometry. You must apply the settings by pressing **Update Properties**. Trestle generation then occurs automatically if the segment qualifies and the required parameters are set.

4. Managing Edits

Saving Edits

All track-related edits (nodes, segments, trestles, telegraph poles) are kept in memory until manually saved. Use the **Save Changes** button to persist edits to the .json patch file specified in your mod's mixintos.game-graph.

- Saves are written to the mod's Data/ directory.
- If that directory is locked (e.g., read-only archive), fallback is UserData/MapEditor/Exports.
- Saving clears the list of unsaved changes and resets visual indicators.

Scenery edits are **not** part of the patch file and must be exported separately.

Undo and Redo System

The editor maintains full undo/redo tracking for track editing operations. Scenery edits are excluded. Available actions:

- **Undo All** – Reverts all changes made during the current session.
- **Undo** – Reverts the most recent operation.
- **Redo** – Reapplies the last undone step.
- **Redo All** – Reapplies all undone steps since the last change.

All undo/redo state is managed by the MapStateEditor. Each edit is stored as a state delta and applied via internal replay logic. Exiting the editor discards unsaved changes and resets undo history.

Other Editor Actions

- **Rebuild Track** – Rebuilds all visual representations of track segments. This can resolve temporary glitches or mismatches caused by geometry edits or loading issues. It does not affect the actual data structure.
- **Refresh** – Rebuilds the editor interface for the currently active panel. This can be used if the UI becomes unresponsive or misaligned. It does not reload data from disk or discard unsaved changes.
-

5. Telegraph Pole Editor

⚠ Telegraph pole placement is currently not saved to the patch file. Changes will be lost when the game reloads. This feature appears to be a work in progress.

The Telegraph Pole Editor provides tools for placing and modifying telegraph poles along the railway line. Although not functionally essential, poles contribute to visual realism and help align with historical or thematic scenery.

Editor Modes

The following tools are confirmed to exist based on code and editor behavior. Some features, such as auto-placement logic, are not yet fully understood or operational:

- **Create** – Allows manual placement of a telegraph pole on a selected segment.
- **Update** – Adjusts existing pole positions or replaces pole types.
- **Destroy** – Removes a selected telegraph pole.

---

6. SceneViewer and Object Export

The SceneViewer allows users to inspect individual scenery objects in the map. It is strictly read-only: you cannot move, edit, or delete objects via the viewer or the in-game interface. However, you can view technical details such as identifiers and transform data, which can be reused in external mod files. Editing scenery via JSON is possible but falls outside the scope of this mod.

Navigating the Viewer

There are two ways the SceneViewer becomes active:

Limitation: The SceneViewer is an inspection tool only. All properties are static. To reposition or modify scenery, you must edit external files manually.

1. Browsing Existing Objects

- Open the **SceneViewer** via the MapEditor toolbar.
- Browse the loaded hierarchy of scenery objects.
- Selecting an entry shows technical details: transform matrix, prefab identifier, and optional masks.

2. Placing and Cloning Objects

- Click the **Place Object** button in the MapEditor.
- You are prompted to click a location in the game world.
- After clicking, the SceneViewer opens with a **Clone Object** button.
- Clicking **Clone Object** places the object at the previously clicked location.

Both views use the same SceneViewer layout to present the object structure.

Exporting Object Data

The export feature is accessed via the general *Settings* menu in Railloader—not from within the MapEditor. It generates a structured JSON-like file that contains:

- **Object Hierarchy**: Name, activation state, and nested children.
- **Components**: With fields like identifier, TypeName, and Unity-style Properties.
- **Transform Matrix**: A localToWorldMatrix per object. This encodes position and rotation but does not expose them directly.
- **Mask Data**: If present, fields like sizeX, degrees, and falloff.

Limitations

- No direct position, rotation, or scale fields.
- No metadata about placement time, author, or in-game selection.
- Not re-importable or usable for quick edits.

Use Cases

This export is mostly useful for mod developers and technical users. Potential purposes include:

- Debugging prefab placement.
- Recovering identifiers.
- Reverse-engineering placement logic.
- Comparing scenery structures across map versions.

Note: Extracting usable coordinates from the matrix requires scripting. This is outside the capabilities of the MapEditor.

7. Limitations

Despite its powerful features, the MapEditor mod comes with several limitations and caveats that users should be aware of:

Mod Selection

- The list of selectable mods is static per game session.
- Mods created after the game has launched will **not** appear in the dropdown until the game is restarted.

Track Editing

- Segments and nodes must be connected correctly. Invalid configurations may result in rendering glitches or unexpected behavior.
- AutoTrestle generation can fail if segment geometry is malformed or if track height is too low.

Export and Save Behavior

- Patch saves apply only to track systems. Scenery is excluded and handled separately.

Editor Lifecycle

- All changes are lost if the editor is closed without saving.
- Rebuilding assets may require manual refresh steps (e.g., toggling visibility).

Users are encouraged to test frequently and save often to avoid data loss.

---

8. Example Workflows

This section provides end-to-end editing scenarios designed to walk users through the entire workflow using the MapEditor. Each example assumes a working mod has been selected (see Chapter 1) and that the user is familiar with basic Railroader navigation. These workflows are validated step-by-step and are intended to produce a successful in-game result without requiring additional tools or external patch edits.

Follow these examples exactly to test your understanding of the editor. Each scenario builds on previously explained tools and behaviors.

A. Create a Branch Line

1. Open the **MapEditor** by clicking the icon in the top-right corner of the game interface.
2. Make sure a valid mod is created and selected. If you haven't created one yet, follow the steps in Chapter 1: *Selecting and Saving Mods*. Without a mod, no edits can be saved.
3. Set a custom prefix in the field provided. The default is Custom_, but you are strongly encouraged to enter your own identifier — and keep the underscore at the end (e.g., MyTest_). This helps distinguish your edits from those in other mods and prevents naming conflicts.
4. Open the **Track Node Editor** by clicking on an existing node in the world. Nodes are represented by bright cyan-blue triangular icons pointing along the track direction. Clicking a node opens the associated editor panel for that node.
5. Click **Create New** to create a new node. It will be placed 2.5 meters to the left of the selected one and will inherit its rotation.
6. Click **Move** to activate positioning mode. Note: the operation mode must be set to **Move**, as it defaults to **None**. Use the 3D gizmo to drag the new node to its intended position.
7. Adjust the node's rotation:
    - Option 1: Click on another node with the desired orientation. In its **Rotation** field, click **Copy**. Then return to your new node and click **Set** in the **Rotation** field to apply the copied rotation.
    - Option 2: Manually enter rotation values (X, Y, Z) in the **Rotation** field and click **Set** to apply.
    - Option 3: Click **Rotate** and use the on-screen gizmo to adjust the orientation interactively. The farther you move your cursor from the node while dragging, the finer the rotation adjustments. Holding **Alt** while rotating changes the active rotation axis (e.g., for banked curves). This is an advanced feature best explored through trial and error once you're comfortable with the basics.
8. To access the **Track Segment Editor**, first click on the node you just placed. This will highlight the connected segment in yellow. Then click on the yellow crossbar at the center of that segment to open its editor panel.

9. In the segment editor panel, adjust the segment parameters as needed (see *Chapter 3: Editing Track Segments* for details).
    - Set **Priority!** Negative priority is convention on mainline branches.
    - Click **Update Properties** to apply all changes.
10. Click **Create New** again after selecting the newly placed node. This creates a second node beyond the branch, adjust as needed.
11. Use **Shift + Click** from the second branch node to the new node to create a connecting segment.
12. After creating a switch, press **Clear Selection**, then click on the node involved in the switch. In the editor panel, locate the **Flip Switch Stand** checkbox. Toggling this flips the visible stand to the opposite side, useful for aligning switch visuals.
13. Use **Straighten** or **Move** to adjust the geometry of the newly created segment visually.

Use **Straighten** on segments that converge or diverge with caution. These segments form a merge or split and should retain their curvature to preserve correct geometry.

14. Use **Undo** to reverse the last segment change.
15. Use **Redo** to reapply it.
16. Press **Save Changes** to write your work to disk.

B. Build a Bridge Over a River

Note: This example assumes you have already created and selected a mod as described in Example A.

1. In the **Track Node Editor**, place or use one node on each riverbank, practice with A to familiarize yourself with editing and extending tracks by manipulating nodes and segments.
2. Click on a node to select it, then use **Ctrl + Scrollwheel** to raise it above the terrain. Make sure you are in **Move** mode, or vertical adjustment won't work. Alternatively, you can manually adjust the Z-position in the Position field, but this requires familiarity with world coordinates.
3. Create a segment between the two raised nodes using **Shift + Click** in the Track Node Editor, once their elevation is to your liking.
4. Click on the newly created segment to open its editor panel. In the **Track Style** section, enable **Trestle** if you want visible support structures. If you leave it disabled, only a bare bridge span will be created. Optionally, configure the **Head Style** and **Tail Style** to adjust how the bridge connects at its ends. Fill in any other segment properties as needed (see *Chapter 3: Editing Track Segments* for details). Optionally, configure the **Head Style** and **Tail Style** to adjust how the bridge connects at its ends.
5. Click **Update Properties** to apply settings. If height and type are valid, the trestle is generated automatically.
6. To adjust later, modify node height or segment shape and reapply **Update Properties**.

C. Reposition an Existing Track Segment

Note: This example assumes you have already created and selected a mod as described in Example A.

1. Select a segment in the **Track Segment Editor** that needs adjustment.
2. Click **Move** to open the Bézier curve editor. Drag to a new shape.
3. Use **Straighten** only if you want the segment to continue straight from the origin node in its current rotation. Note: this does not smooth out curves — it overrides the geometry and may cause the track to cut through terrain unexpectedly.
4. Use **Inject Node** to insert a control point at the midpoint of the segment. This is useful for creating more complex geometry — for example, adding an extra curve or defining a branching point along a long span.
5. If you want to make fine-grained adjustments to individual node positions or rotations, click on a node directly to edit it using the Track Node Editor.
6. Use **Update Properties** to apply segment changes.

These workflows demonstrate how MapEditor tools interlock: nodes define control points, segments define paths, and each edit builds toward a functional network structure.

9. TODOs (documentation)

This section outlines features and mechanisms that are partially implemented or require deeper inspection of the mod's internals.

Multi-Mod Support

- Investigate whether the editor supports cross-mod editing contexts or shared references.

Live Patch Reload

- Currently unsupported. Any change to a mod or patch file requires restarting the game for it to appear in the dropdown.

Unresolved Topics

- **Full explanation of and ** – The priority field appears to influence routing behavior, possibly affecting how AI or pathfinding algorithms choose between parallel tracks. Its precise weighting mechanism and interaction with groupId remain unclear. groupId likely serves to logically link segments (e.g., switches or track bundles), but its propagation and usage are not consistently documented in the patch structure. Further code tracing or runtime logging is needed to confirm their impact.

- **Multi-mod setups and cross-references** – Determine whether the editor can safely reference or extend track data across multiple mods, and how it handles conflicting definitions or shared node IDs.

- **Patch hot-reloading feasibility** – Assess whether changes to patch files can be reflected in-game without a full restart. This may depend on caching, patch injection timing, or UI event binding.

- **Scenery auto-placement (TelegraphPole logic)** – Investigate how poles are automatically placed along segments, including prefab assignment, spacing intervals, and snapping behavior. Code tracing required in placement logic.

- **Segment validation rules (e.g., geometry constraints)** – Clarify how the editor determines invalid segments, especially in cases of malformed curves, incomplete connections, or out-of-bounds geometry.

- **Editor rotation modifiers** – Confirm whether holding Ctrl + Alt while dragging rotation gizmos changes the active axis or coordinate space (e.g., local vs world).

Version Notes

This guide documents CzBuCHi MapEditor v1.0.25128.1912.