Trolleybus User Guidance

6 Mar 2023 | 此指南有中文版
This file is generated with Markdown PDF

Intorduction

This is a user guidiance for making realistic trolleybus system in Teardown.

This guidance is divided into 3 sections:

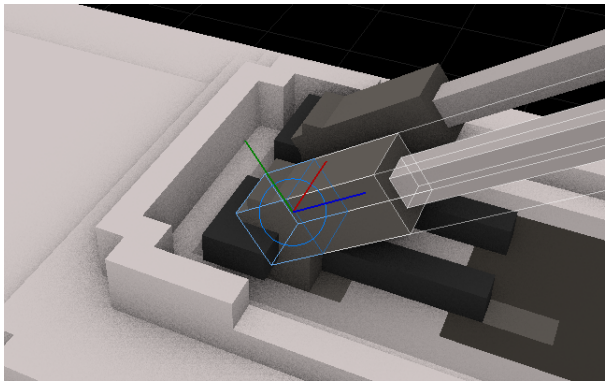
- Basic
- Advanced
- Limitations

Basic

Trolleybus

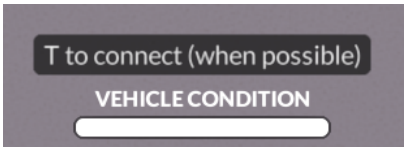
To make/convert/modify a regular vehicle to trolleybus vehicle, you need to ensure:

- `trolleybus.lua` is a parent script of a single vehicle.
- Trolleybus **poles** are children of the vehicle (**DO NOT** use `prop=true` for poles!) and each pole is a seprate dynamic body.
- Pivot of each pole body **MUST** at centre of joint which connects this body to vehicle. (as shown below)



- **Shoe** of a pole should be a seprate voxel, have tag `trolleyShoe` and is recommended to be small.

If you setup your vehicle correctly, you should see this when driving.



☑**NOTICE**

- This mod have took articulated bus into consideration. You can follow same operations mentioned above to make an articulated trolleybus.
- This mod is compatable with spawning.

Route

Trolleybuses always driving along routes.

To setup a trolleybus route, you need to ensure:

- Route is made up of several **overhead lines**, consists of nodes (shapes) and ropes.
- Nodes should be as small as possible and tagged with `trolleyNode`
- Ropes should **ONLY** connect to bottom of nodes and is recommended to always in tension for better visual effects.
- Lines does not intersect, except at juntions[1].
- Route could be easily reached by trolleybus and have enough space in between.
- Curved sections[2] should not have larger turning radius (compared to vehicle's turning radius) and have shorter section length[3].

Notes

[1]: See Advanced for more information.

[2]: Rope with nodes at both ends is a Section.

[3]: To prevent unexpected disconnect.

☑**NOTICE**

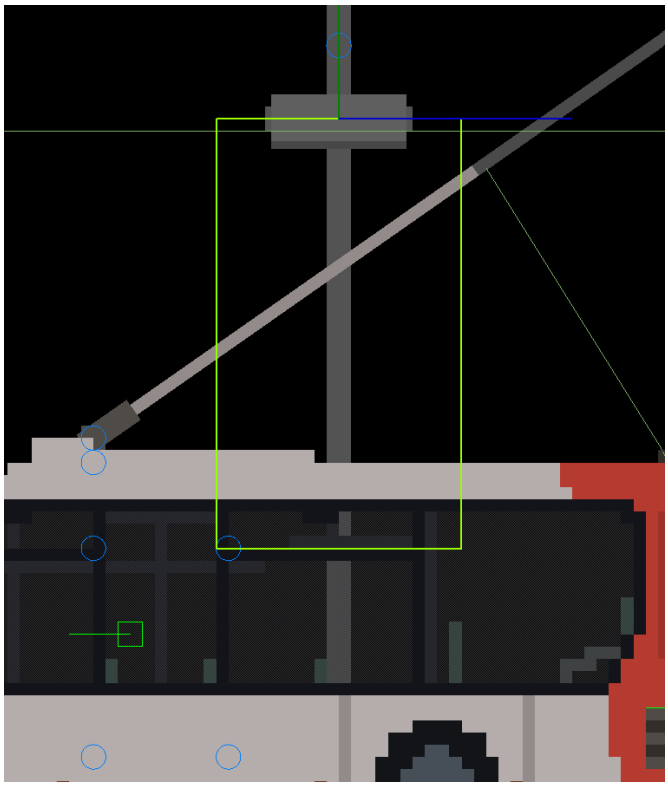
- This mod is compatable with flexible suspended overhead lines, see Advanced: Ignored Ropes for more information.

Capturer

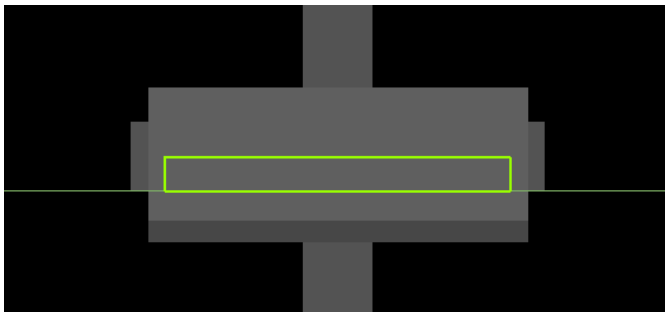
Disconnected vehicles can only reconnect to a route with capturer.

To make a capturer, you need to ensure:

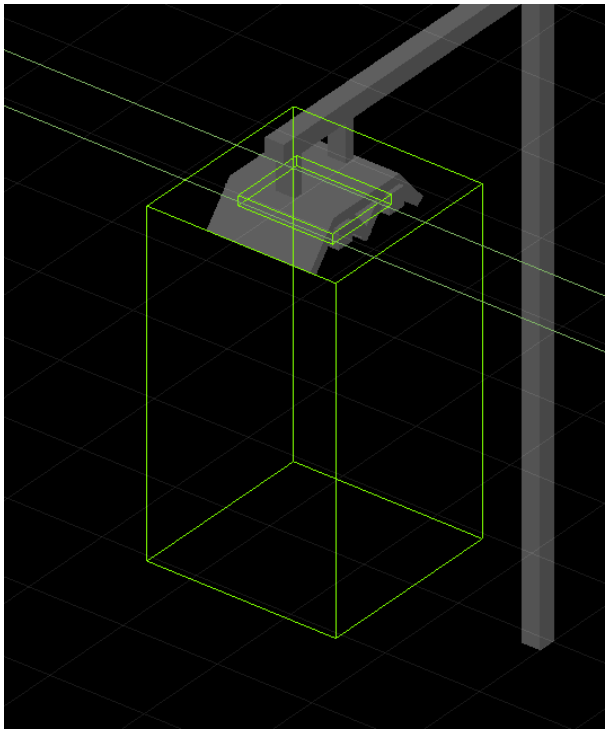
- `capturer.lua` is a parent script of a single capturer。
- **ALL** nodes of the capturer are children of a body tagged with `trolleyCapture`
- Lines does not intersect at capturer.
- **ONLY** one node on each line have additional tag `trolleyCapture`
- A trigger, tagged with `trolleyCapture`, covering **ALL** nodes & ropes inside capturer, and a slightly larger area which can make contact with vehicle. (as green rectangle shown below)



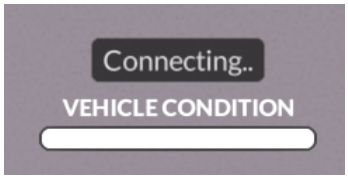
- A second trigger, tagged with `trolleyCheck`, **ONLY** covering **ALL** ropes inside capturer, and is recommended to made as small as possible. (as green rectangle shown below)



- The capturer should looks like this, first and second trigger are shown seprately as larger and smaller green box.



If you setup capturer correctly, you should see this when driving a trolleybus which is waiting for connection.



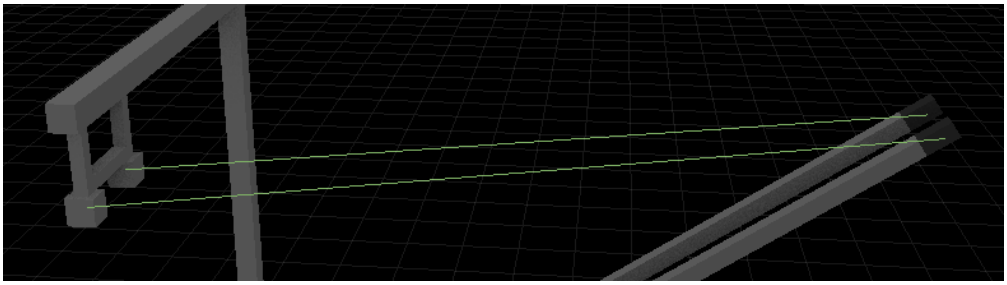
Advanced

⚠**Please make sure you have read Basic section.**

Initial Connection

You can make a trolleybus to connect with a route on load.

Connect all shoes of a trolleybus to nodes, which are part of the route, with ropes tagged with `trolleyConnect`.



The trolleybus would automatically connect to this route on load and delete ropes.

Ignored Ropes

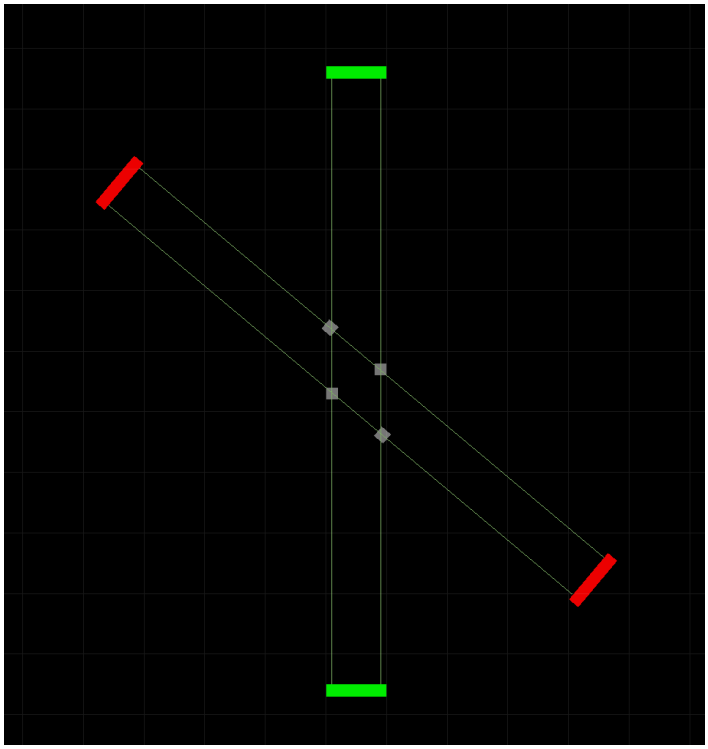
Adding tag `trolleyIgnore` can make script ignore a rope. This would be helpful while making cable suspended overhead lines.

Intersection

Same as real intersections, you can make routes cross with each other without causing problems.

To make a proper intersection, you need to ensure:

- Sections are grouped according to directions.
- **Ropes** in same direction are tagged with `trolleyGroup=[group]`, where `[group]` is a unique id to identify groups. Different groups **MUST** use different id[1].
  ⚠**CANNOT USE** `-` *(Hyphen-Minus)* **IN GROUP ID**
  ⚠**CANNOT USE Lua magic characters IN GROUP ID**
- **ALL** lines in any group **MUST** include at least one rope ahead and after an intersection (as shown below)

  In this pictrue, two routes (red and green) intersect and form an intersection, all ropes in this picture have grouped with `trolleyGroup`
- If there're several intersections close to each other, treat them as one large intersection[2].

Notes

[1]: Only for interscting groups, non intersecting groups can have same id.

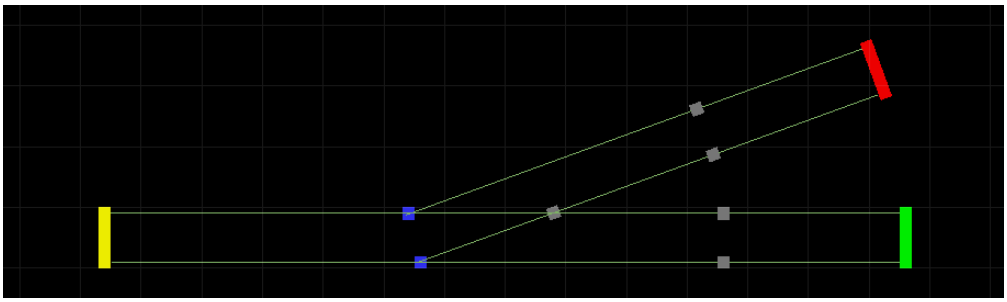
[2]: Since different groups on same line must have at least 2 regular sections in between otherwise grouping would recognize later one as another group and refuse to travel on it. Regular sections have no group therefore it can be picked by any group. See Complex Intersections.

🛠Additional Notes

Merge Switch

You can treat merge switch as a special type of intersection: two lines entering but only one exiting.

As mentioned in note [2] in previous section, regular line sections can be picked by any group, therefore we just need to group incoming routes till intersection point (shown as below)



- Lines between blue nodes and red end are group 1
- Lines between blue nodes and green end are group 2
- Lines between blue nodes and yellow end are not grouped

Note

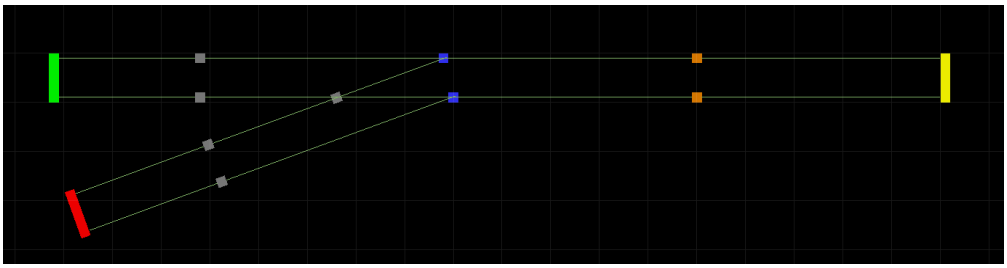
- Minimum grouping on each branch is **one section from closest intersecting/merging point**. Which means in this picture, ropes connecting green or red end blocks doesn't have to be grouped.

⚠**NOTICE**
Merge switch **CANNOT** act as split switch

🛠**Additional Notes**

Split Switch

Although visually similar (shown below), split switch is different from merge switch. Player are required to choose a direction before entering this type of switch (same as reality).



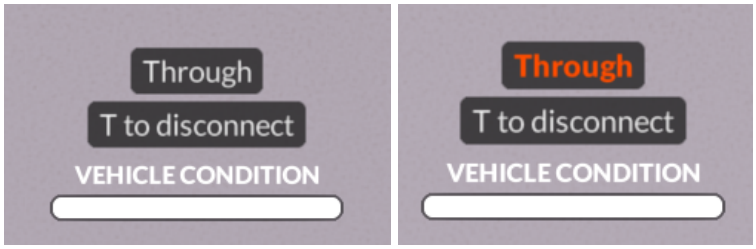
Different from merge switch, which can have unlimited route connections in theory, split switch only split a route into two branches:

- **Through** route
- **Turnout** route

Through route is default direction a trolleybus would go, while **turnout** require manual triggering. This mod slightly modified this logic so that this operation is a toggle operation.

To setup a split switch, you first need to make a merge switch. After that, make sure:

- Nodes which routes splits are tagged with `trolleySwitch=[group1]-[group2]`, where `[group1]` and `[group2]` are group id of each route, `[group1]` is group id of through direction.
  e.g. (tag value only):
  - group 1 (default): `10`; group 2: `12`; nodes: `10-12`
  - group 1 (default): `asd`; group 2: `6`; nodes: `asd-6`
  - group 1 (default): `7t`; group 2: `p2`; nodes: `7t-p2`

- [*Optional*] Nodes before splitting point can have tag `trolleySwitch` with no tag value to extend notification zone. (Nodes colored orange in this picture)
  ⚠ Nodes tagged with `trolleySwitch` **should** be continuous.
  

  Within notification zone, selected direction is displayed; Text would change red and bold, and selection would be locked before approaching switch.

Notes

- Minimum grouping on each branch is **one section from closest intersecting/splitting point**. Which means in this picture, ropes connecting green or red end blocks doesn't have to be grouped.
- Minimum distance between two split switch is: **Two regular sections, excluding one side of nodes.** (`N -- N --`, `N` is node, `--` is rope)

🛠 Additional Notes

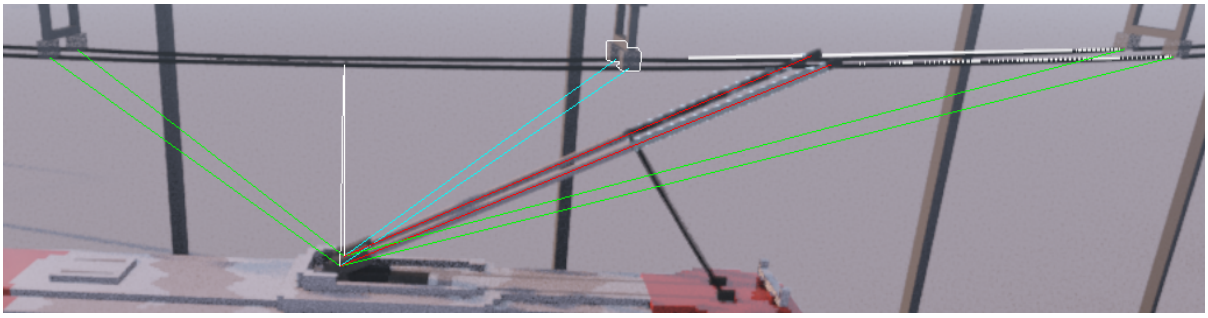
Additional Notes

Only depend on sections discussed above might not be enough to make more complex trolleybus system.

Here I'd like to share some tips and several possible routing setup which could outherwise hard to figure out without take look into scripts.

Debug Mode

Setting a parameter (`param`) of a `trolleybus.lua` script to `DEBUG=true` can enable debug mode on this vehicle.



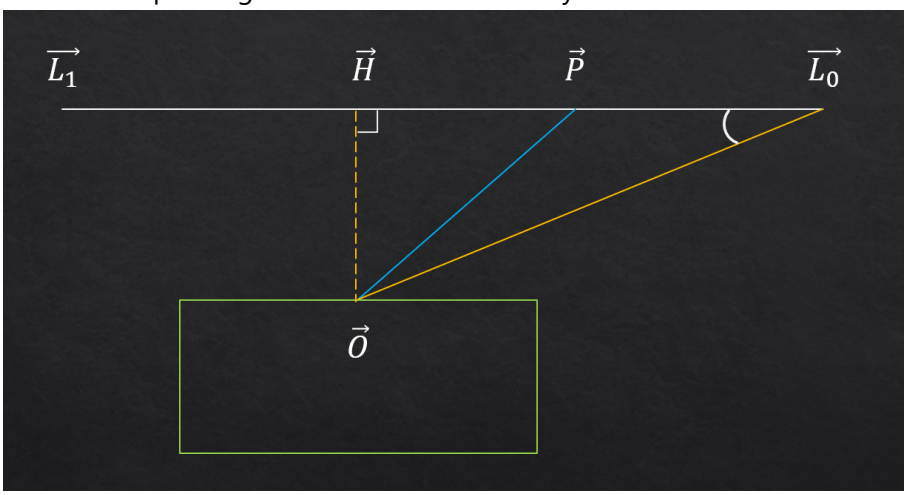
- ***Blue*** lines point towards ***closest node*** from shoe.
- ***Green*** lines point towards ***neighbouring nodes***, neighbouring nodes are always 2 for each closest node if trolleybus can keep travelling on route.
- ***Red*** lines point towards ***calculated connecting point***, this line should always overlap with pole.
- ***Thin White*** lines point towards ***closest point on line***, see Distance Limitation for more information.
- ***Thick White*** lines indicate ***connected line***, this line should always overlap with a section of route.
- ***Origin*** of Blue, Green, Red and Thin White lines is ***pivot point of pole body***.

You can always keep a debug vehicle in scene to check route setup.

Distance Limitation

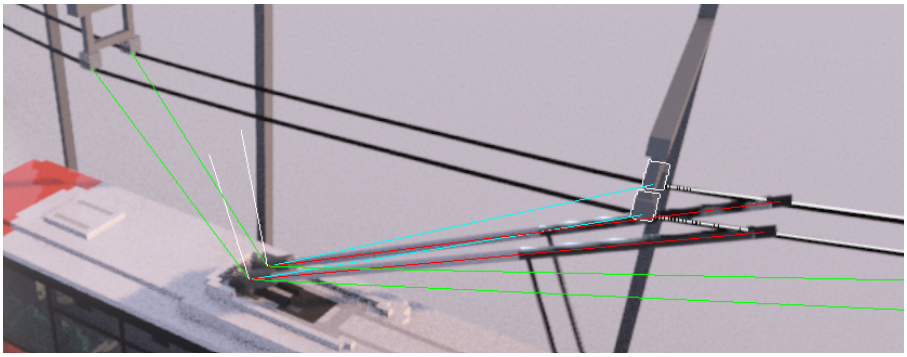
There's a maximum perpendicular distance allowed for any trolleybus poles to maintain connection. This value is **90% of pole length**

Here's a simplified geometric model for trolleybus and a line.



Line $\vec{OP}$ is pole and $\vec{OH}$ is perpendicular distance. When $|\vec{OH}|$ is longer than $|\vec{OP}| \times 0.9$, the pole would disconnect.

⚠Please note that the line is the one with pole connected, not visually closest section of a route. You can easily spot this when driving a trolleybus with debug mode enabled at curved sections.



The two thiner white lines indicate perpendicular distance, you can see these lines would intersect with line which have poles connected, rather than those directly above

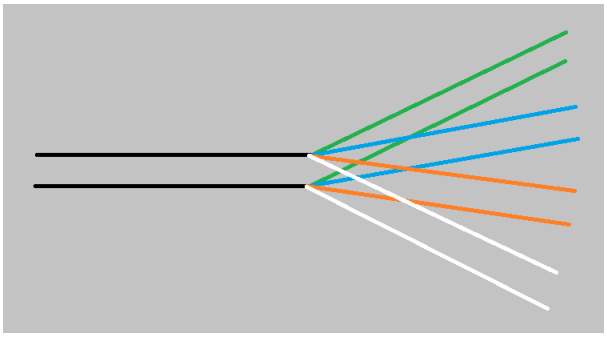
Dead End & Auto Disconnect Zone

When trolleybus travells into an dead end, poles would automatically disconnect after shoes passed end of the route.

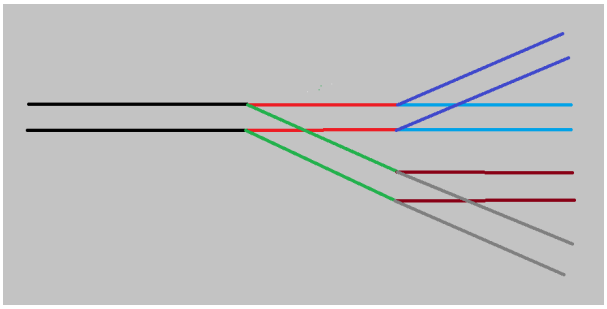
Use along with `trolleyIgnore` tag mentioned at here, we can make a section of lines which is visually normal but not acting as overhead lines. This would automatically disconnect trolleybuses once drive into this zone.

Continuous Merge Switches

Although theoretically, you could connect many routes to one and make a multiple-to-one merge switch, it's not an easy task to wire up these routes nice and clean.



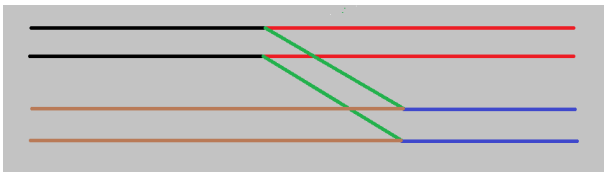
Instead, we can put multiple regular merge switches together to form a continuous merge switch.



Mod also allows inclusive grouping.
For example, `1a` and `2` merge into `1a2`, which is not hard to see that `1a`+`2`=`1a2`; This line can then merge into `1a4f2`, which is `1a`+`4f`+`2`.
Moreover, inclusive grouping does not requrie regular sections to seprate different merge switches. So continuous merge switches can be realy compact and simple.

Crossover Switches

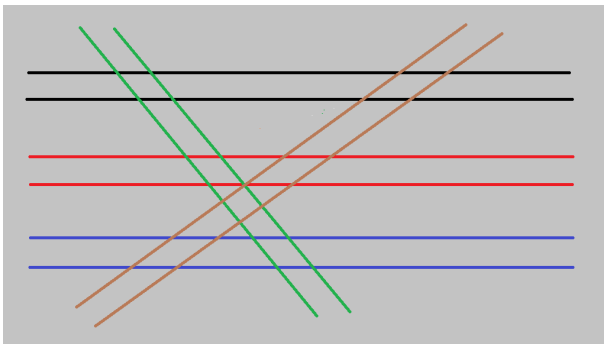
Although there is a minimum gap required between different split swtiches, we can put merge switches directly on branches of a split switch. This allows us to make a single crossover switch.



With proper design, double or scissors crossover are also possible.

Complex Intersections

As shown in picture below, we can make more complex intersections with proper grouping setup.



A simple trick for this is: always make different routes seprate groups while groups on same route remains same.

Limitations

- **No power simulation**
  Currently there's no power simulation, which means trolleybus are more like hybrid vehicles.
- **Limited pole control**
  Poles completely relies on it's own weight, joint springs and/or pull from ropes to retact.
- **Loose Rope**
  It's impossible to get path of rope under existing modding framework, therefore this mod uses nodes to determine path of ropes and that's also the reason I suggest ropes to keep under tension.