

Hey,

This guide is based on the existing copy, but is being updated for TPF 2. Except for the files, most of it is updated.

If someone uses the article to learn how to mod, please give me feedback. Otherwise, please feel free to make suggestions, requests and criticisms. Thank you very much!

Before I started modding, all I really wanted was my favourite vehicles. I had no idea about modding, how mods work and how to create them.

I would like to give an overview here to explain the basics to future modders and give an idea of what is needed for a vehicle mod.

For the time being, this is only about vehicles - stations and placeable assets only differ in the last steps (script). Information on the structure of the different mod types and their components can be found in the [TPF 2 Wiki](#) and partly valid in the [old TPF 1 Wiki](#).

Here is a very good guide as a supplement to Seamon with more concrete instructions for action: <https://www.transportfever.net...sten-Mod-erstellen-2-pdf/>

From 0 to ready - very good video (3min):

https://www.youtube.com/watch?v=_LfzbB3YYDA&list=LL&index=43

A slightly older modding guide with video material is here [Modding guide](#)

Here you can also find videos by Zargom: [Modding Tutorials for TPF Vidcasts/Livestreams](#)

In French (but with pictures) there is a version here:

<https://docs.google.com/docume...a5KtNCVKEkKVDm930e90/edit>

First of all, some information about the time required. I need a few weeks for a tram, I worked on my first railcar for more than half a year. After 2 years of practice, I worked about 120 hours on my P-car tram package with some repaints.

Transport Fever mods are quite complex and do not take two hours to complete. It requires a lot of **motivation, will and determination**.

However, seeing your own model running (also with others) is a great confirmation of the work you have put in.

P Car details: 1 day model, 2 days texture, 1 day rest textures +test export, 1 day repaint, 1 day export + animations, 1 day sidecar+passengers+displays, 1 day repaints, 2 days Christkindl, 1 day night versions+format+small things+sound, 1 day rest things +release
= 12 x 10h = 120h

1. Structure and Content Of a Mod

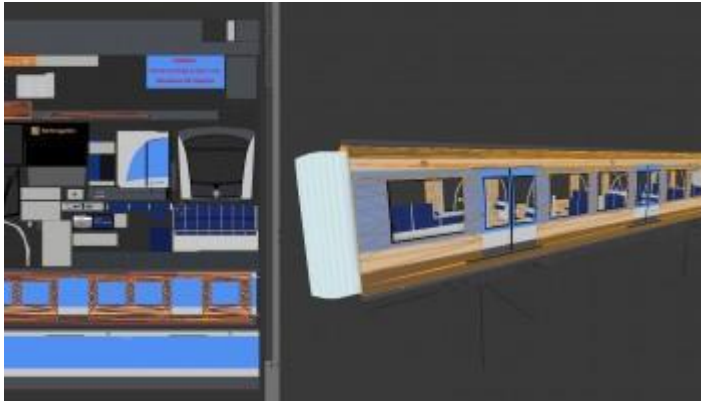
In general, you can learn a lot from existing mods. The structure is always the same.

You have a 3D model, which is modelled with the help of CAD programmes, such as Blender.

This requires a texture, i.e. colours, inscriptions, etc. The texture is a normal image that is created in image editing programmes such as GIMP.

So that the game knows which part of the image belongs to which part of the model, the model is "mapped" with the help of a UV map. This combines model and texture.

The whole thing looks like this, for example. The orange are the surfaces of my model, which I put on top of the texture.



Now you define special materials for individual areas. These tell Transport Fever which properties the surface should have. Windows are transparent and require a different material than, for example, the walls or the luminous spotlights.

Materials are used to ensure not only effects such as luminosity (for spotlights) but also the correct representation of transparency.

This model is now converted into Transport Fever readable files using a special exporter.

These are the .msh and the .msh.blob files. The files are generally based on the programming language Lua and are easy to understand and edit, e.g. with Notepad++.

The .msh files are hardly relevant, information is stored there but we usually do not need to change it.

In the msh.blob. files our model is integrated with UV Map, they consist of cryptic strings and we can care less about them.

Each object (e.g. seats, outer shell, wheels) is exported individually, gets its own .msh and .msh.blob file.

These object files are now assembled with further information in the script. There are differences between trams, railcars and stations. For now, we will only look at the normal wagons, rail cars or buses/trams.

- Multiple Unit for railcar(.lua)

--- Model (.mdl)

----- Group

----- Mesh (.msh/.msh.blob)

+ Material and animation

We already know the mesh/object.

These meshes are now specified in the .mdl so that the game knows which individual objects belong to the wagon. One .mdl is then the first wagon. A second .mdl is a middle wagon.

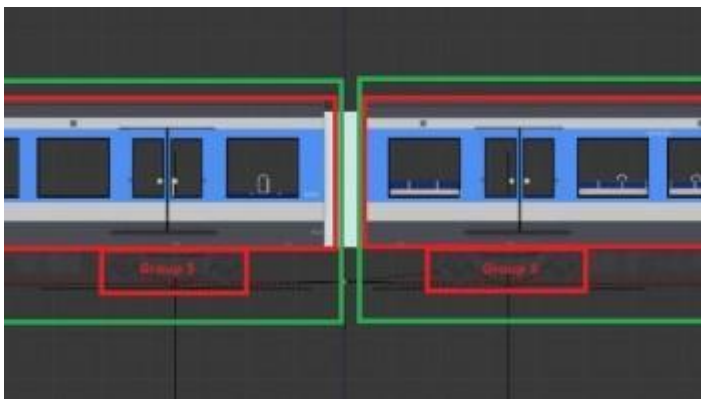
For each mesh we also have to write which material is to be used and, if applicable, which animation this mesh has.

Since we want our car to turn nicely, these meshes are grouped together. These groups indicate where the contact with the rail will be in play.

Each Mdl e.g. "Wagon 1" then has two further subgroups here "Bogie 1" and "Bogie 2", which define the actual contact with the rail, as we want our wagon to be movable on the bogies.

Sometimes [fake bogies](#) are offered as alternatives, especially for trams and buses.

However, we are looking at the normal bogies here and fake bogies need not interest us any further.



Usually you create several copies of the same group, which contain differently detailed meshes. As soon as you move the camera further away, the second group (the second Lod = Level of detail) is displayed, in which very detailed meshes such as grab rails and interior fittings no longer appear. For this, the individual objects must not be merged in the CAD programme, but must exist as a separate mesh after export (separate .msh/.msh.blob file).

You can also replace the wagon with a cuboid. For this you have to model the wagon again with less details, I'm usually too lazy for that.

In the .mdl, the different lods are then sorted with the groups and meshes and further settings such as price, weight, capacity or loading speed are made.

For railcars, you can then string together several .mdl in the MultipleUnit.lua, so that you can build a train of any length with two wagons (two .mdl).

Furthermore, there are folders where our materials are located, in which we link the texture. For these there are extra folders, as well as for sound effects, for example.

Besides our normal texture, there are others, for example for reflections, dirt or colourability.

Since Transport Fever models as resource-efficiently as possible and does not model each rivet individually, there is also an extra texture (normal map) that makes the train look even more plastic without having to model it separately. More about this later.

Each piece has its own folder, is linked via files and then assembled like a puzzle. This allows good adaptability to always have the same structure, regardless of whether it is a railcar, locomotive or tram. The only differences between the vehicle types are in the settings or the folder name (tram instead of train, for example).

2. The Model

For the modelling and the UV map, as well as the export, I use Blender, as it is free and very comprehensive. With version 2.8, the interface has been greatly revised, making it less shortcut-heavy and less convoluted.

Most (Internet and video) tutorials will therefore not be entirely correct. I also use the old Blender 2.79 here. If you have questions about Blender 2.8, ask [EISFEUER](#). Some also use Sketchup for modelling, as it is clearer and easier to use. However, it is limited when it comes to functions and especially curves. For UV Map and export, it must then also be exported to Blender.

I personally recommend Blender, because you have everything here. The entry hurdle is higher here, but in my opinion it pays off.

For the actual modelling, it is best to look at the basic functions in normal Blender tutorials.

It is a matter of learning by doing. Especially the key combinations, operation and menu are not easy.

Everyone has their own approach, functions and strategy here.

To explain the actual modelling here would go beyond the scope, but in the following I briefly explain my approach with the most important steps.

[Here you can find information about blending itself](#)

Before modelling, however, the most important thing is to have a good starting point. You need exact measurements, preferably with technical drawings and distortion-free pictures. Technical drawings can often be found on the manufacturer's website, in books, from other modders (https://docs.google.com/spread...un4ELYfkL_fGcBtL-a2c/edit here or just ask), or on request from the manufacturers. Youtube videos also help to see details.

Drawings can be inserted as backgrounds in Blender and thus the shape can be compared. You should pay attention to the quality of the drawing/image and not blindly copy it. Perspective and the level of detail have a great influence. Therefore, you should have as much material as possible to be able to compare from different views. So far, not one of my drawings has been correct. Often, proportions, views or angles/details are not correct. This can have a significant influence on the shape.



My approach:

I usually start with a cuboid that I set to the correct dimensions. It is important that the coordinate system is correctly aligned. X points to the right and Z to the top. The train later moves from left to right, so the front part is on the right (press numpad 1 and possibly 5).

I add section lines with Ctrl + R, move them back and forth a little, round off the edges with Ctrl + B, extend surfaces with E and thus assemble my model from various cuboids, cylinders and surfaces. Cut/connect with K, connect/fill with F.

You should make the walls double so that you can make the interior cladding a different colour and not have the exterior texture inside as well (e.g. blue). To do this, I take the finished outer wall, copy it (Shift + D) and scale it down a bit. This is now my inner wall.

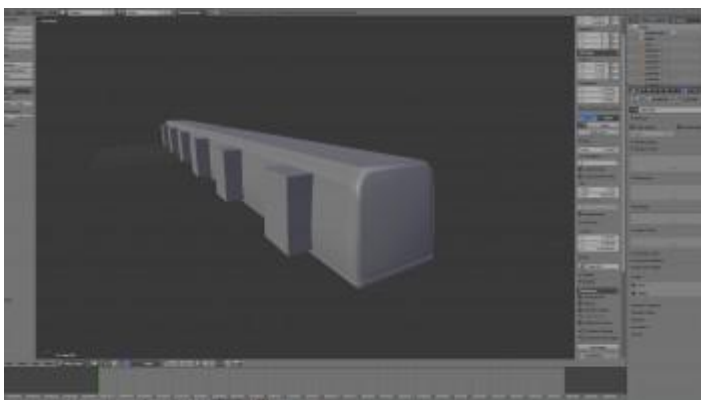
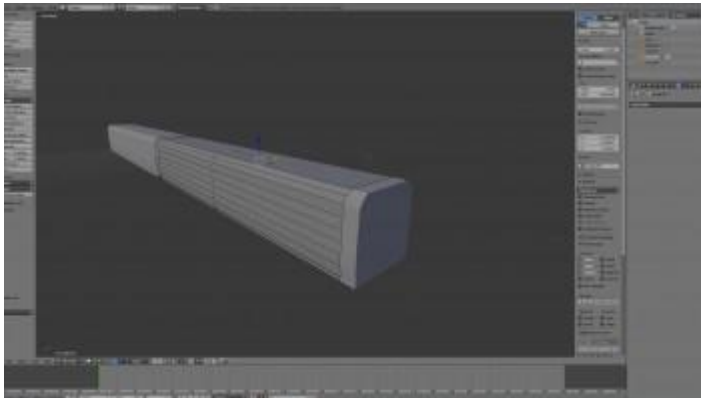
You can also start with a surface, make it a little thicker with E (approx. 0.02- 0.05 m) and then round it off. Then you save on scaling because the inner wall is already included, but you have to make sure that you work the outside and inside in the same way.

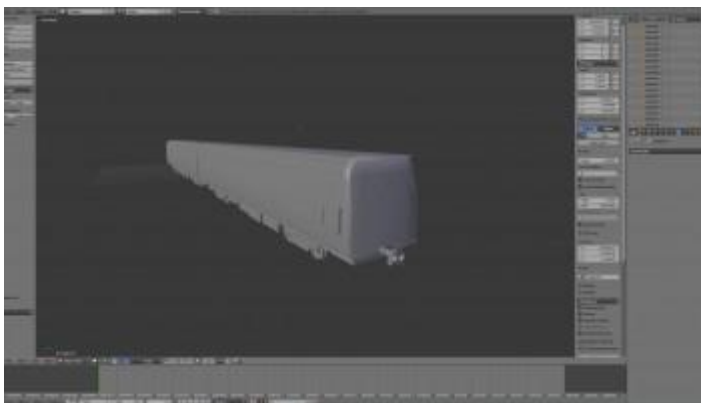
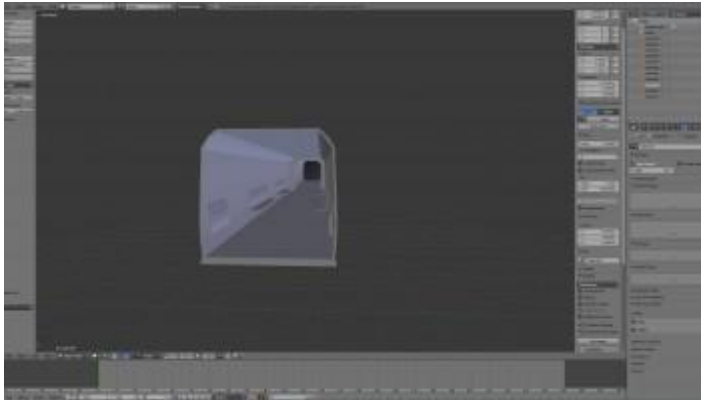
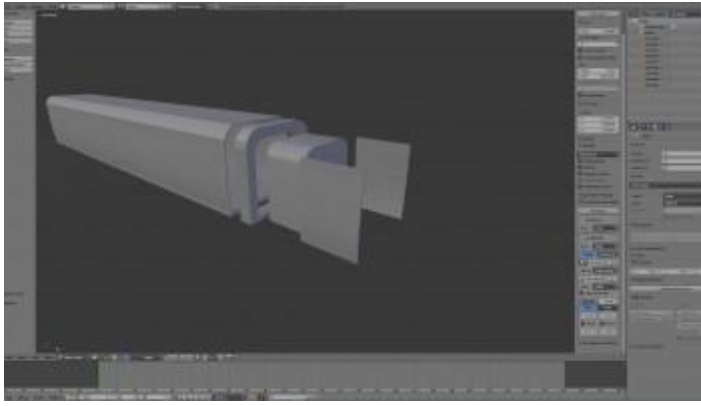
For symmetrical parts, you can also model only one half and then mirror it with the Mirror Modifier. The modifier should be applied before the UV map and texturing.

You have to cut out doors so that they can be animated later. I use the modifier Boolean here. If you have exterior sliding doors, you can combine the right doors on the right side (as well as the left doors on the right side, the same with the left side and the second wagon) as one object. Then you have, for example, three right doors as one object that you can move together (later for the animation). If you want to rotate them or animate them independently, they must be defined as a single object.

Windows can be painted on later, so they don't have to be cut out (although I did that here). If you want to use an interior camera, you can remove the windows inside (only one-sided windows on the outside). If this is too much work for you, you can use a parameter later on that will also display it correctly, but then you may get fly screen effects.

Here is an example of my BVG H.





First I created a cuboid again according to the measurements and the drawing. I rounded it off, divided it into more areas and then adapted it to the model in the picture. In picture 3 I cut out the doors with the blocks. Next the bellows and the first seats. I reused the bogies and substructure from my other mod as the models are similar. I also added details like

windscreen wipers or notches to the front in picture 6. In picture 7, I cut out the windows from the inside (for the interior camera).

The aim is to use as few triangles (Tris) as possible, but just enough so that the object still looks nice and characteristic. The more triangles you use, the more resources the vehicle will need. Guidelines for a tram or a wagon of a railcar are 10 to 20 thousand triangles. If it's double that, it's not the end of the world. However, five times should be avoided.

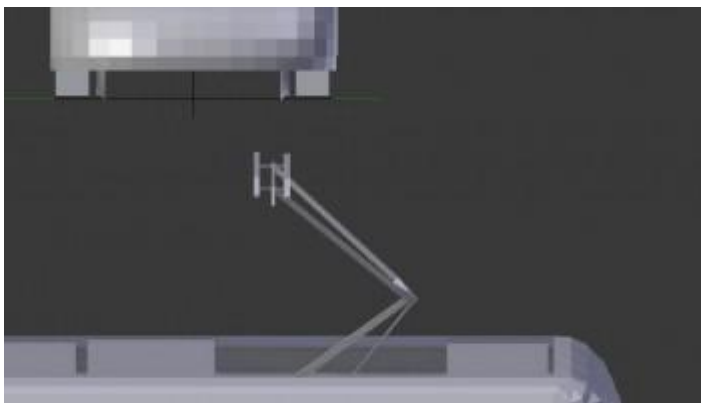
You can also hide details later on if you move the camera further away (lods - levels of detail), so that the many tris do not matter.

In terms of performance, you can also make sure to use a few individual objects (not each seat as one object), otherwise more position calculations have to be done.

However, you should not group too much together so that you can hide individual objects in different lods. For the reversing feature, it is also important to define all objects that can be used for reversing (red headlights, white headlights, left turn signals if present, right turn signals, brake lights) as an extra object.

They are then automatically shown or hidden by the game.

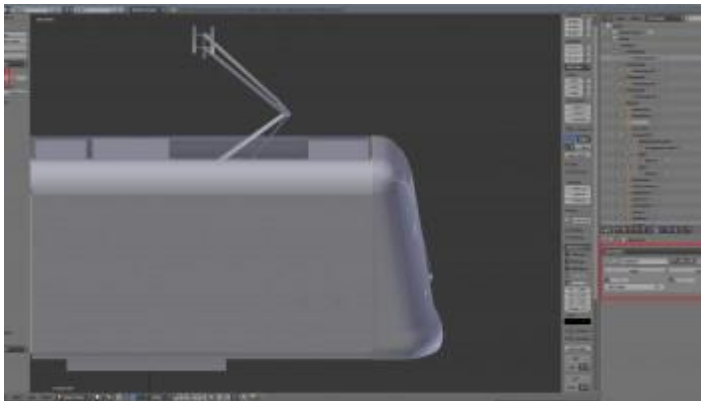
Here the Variobahn with seats. You can see the individual surfaces very clearly - it doesn't need to be more precise.



You can also round off without using more triangles, namely with shading. So that not every edge is smoothed away, I set the Edge Split Modifier to e.g. 40 degrees. However, I do not use this (otherwise it separates the edges, which means that the shading is no longer correct)! Alternatively, you can also select only individual surfaces/faces and make Shade smooth.

You should make sure that the shading in Blender does not have any strange spots, flickering or edges (the surfaces must be connected). Otherwise try Ctrl + N (direction of the normals - outside/inside), Mesh - Vertices - Remove Doubles, or Mesh - Clean up - Delete loose. If nothing helps, select the separated points manually and join them with W - Merge.

After shading:



3. The UV Map

When our model is ready, we need the UV map and our texture.

Each object is now assigned a place on the texture with its surfaces.

In Blender it is best to open a second window. On one side you have the object, on the other side the UV Map (UV Image Editor).

The function for creating the UV Map is called "Unwrap" (U in Edit Mode) and offers different ways of dividing the areas.

Here you simply have to try it out and decide, depending on the case, which of the functions offer the best distribution. You can select only certain areas of the object and unwrap area by area, or the entire object at the same time.

It is important that the surface is as little distorted as possible, otherwise you cannot texture cleanly. With round objects, you usually have to split the surface into several parts.

You should be careful not to place the surface groups flush/too close to each other, otherwise pixel mud will occur later and it will no longer be possible to clearly separate which colour belongs to which surface. This does not mean the individual triangles, but the different objects that do not belong together.

The space on the texture should also be used as completely as possible.

The larger the UV map (and the higher the resolution of the texture), the more detailed you can texture later.

Areas (wagon, doors, ...) on which you later want to apply lettering, pictograms or details should therefore be mapped larger than, for example, the grey interior panelling.

When mapping, you can vary from "map and texture each grey interior panel individually" to "slap all the grey on one spot" → quote pro modder "Nooooooo". 😄

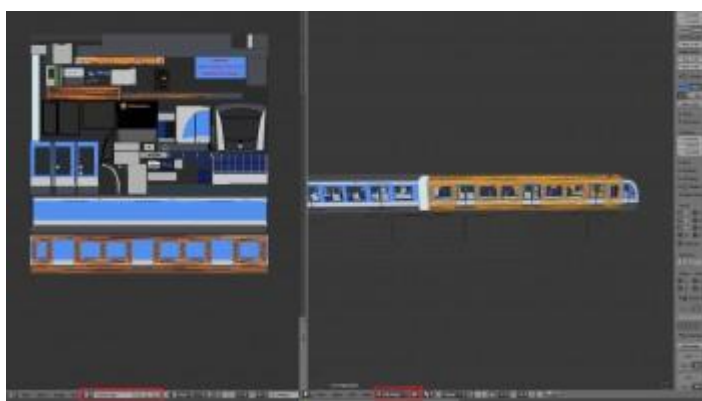
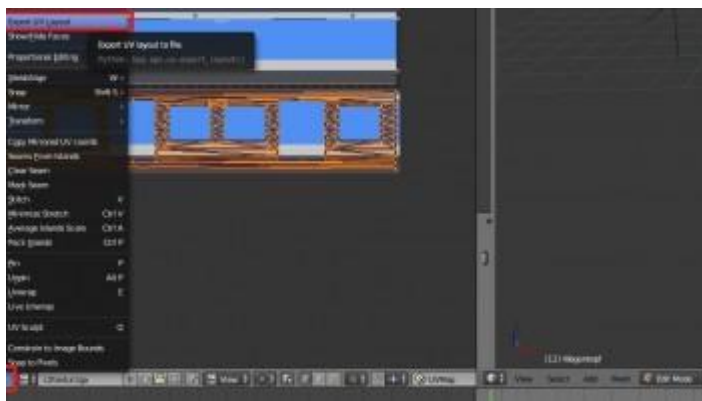
The more precisely you map, the more you can change repaints afterwards.

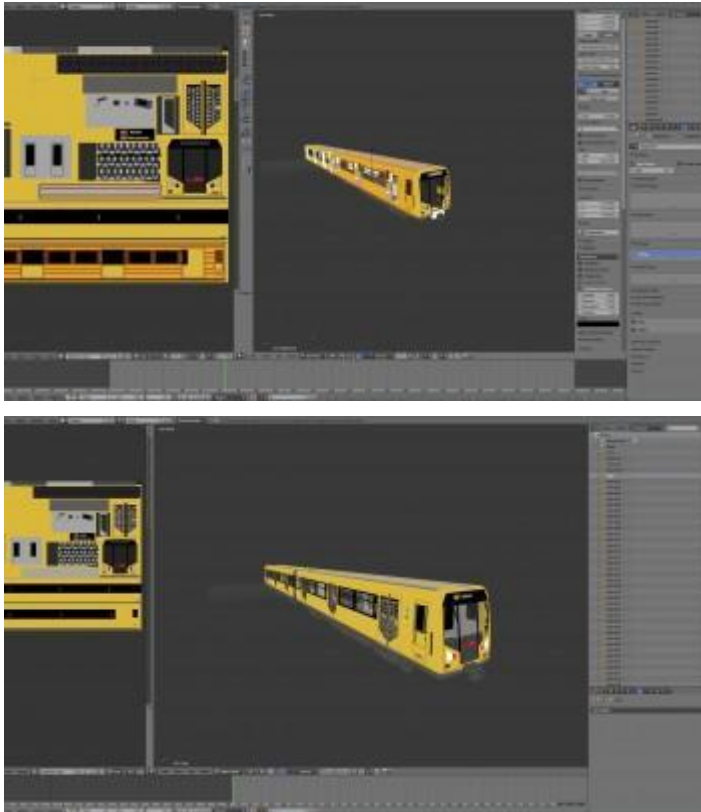
If you put everything on top of each other on a monochrome surface, you cannot make any reflections, colouring, dirt, or subsequent changes there that go beyond a colour change.

There are different approaches to the sequence.

- You can temporarily combine/merge all objects (Ctrl + J), create the UV map for everything, export it and then make the entire texture based on the UV map. The advantage here is that you can use the space on the texture optimally. At the end, however, you have to separate the objects again (P in edit mode). To make this easier, it is advisable to place the objects at different heights so that they do not overlap.
- You create the UV map for one object, texture only this one and repeat this for each object. I work with this method, often switching between texture and Blender, and can thus more easily make corrections to the texture and UV map (size). But here you have to estimate how big the individual objects will be, so that in the end everything fits and no gaps remain. For the outer textures I use about half to two thirds of the texture.

You can export the UV map in edit mode via Uvs - Export UV Layout (image 1). I have inserted the texture directly (image 2), as I am texturing at the same time. If you work according to "1.", you do not have it yet, but only place the UV map on the empty background.





For the BVG H, I added train destination displays and lights in the rightmost image. Doors are also shown there.

4. The Texture

For the texture, just about any drawing programme that goes beyond Paint is suitable. I use the free GIMP (2.8 - not the new one), but many don't get along with it. Some also use PaintShop, Substance Painter, Photoshop, or Paint.net.

The programme used should support different layers, for me 5 are usually enough. Others texture with 100 different layers.

At the end of the day, the colour has to be on it. Layers can help to make the whole thing variable and subsequently changeable. If you do everything on one layer, you can't recycle much for other mods.

An important decision is the resolution of the texture.

The higher the resolution, the more detailed you can texture. Especially with curves, the pixels limit the quality. Here, too, you have to find a middle ground between sufficient quality and consumed power. As small as possible, as large as necessary.

1024x1024 or 2048x2048 pixels are common. I almost always use 2048x2048 and get along with it.

With GIMP, you save the texture as a Gimp file, as all layers remain there. Here you can easily make changes later.

For Transport Fever, we export - this combines everything visible into one layer. It is best to use .tga as the format (uncheck RLE compression!). Later we will switch to DDS.

If we have not yet inserted the texture into Blender, we will do so now (see picture above).

5. Export Model

The most important thing is now in place.

Now comes the export.

That is a bit difficult now. There are two paths and currently no single one. Both have advantages and disadvantages.

The previous way is better for vehicles as of 05.05.2020, as the new way still contains errors. UPDATE: As of 09.01.2020 The FBX way works quite well, but still has limitations. The professional for the exporter is [themeatballhero](#).

New mods that are not to be made for TPF 1 I would make directly with the FBX Importer.

1. new path

Here you skip TPF 1 directly and do it with the new Model Editor.

To do this, the model is exported as an .fbx file (also for 2.8) and imported directly with the Model Editor. This then directly makes the .mdl for us.

Info:

Currently, however, groups cannot be exported correctly from Blender. Only one "level" is possible, so to speak. The sub-grouping has to be done by hand.

The mesh names that are exported are not the ones that can be changed. Blender has internal names for each mesh (with .002, for example) which then look like crap in the MDL.

Also, groups are sometimes rotated 90 degrees, so you have to adjust that accordingly. Only one material per mesh is possible.

Instructions:

We have to use objects of the type "Empty" instead of the extra objects (Model, Group from the plugin).

Export the object as Fbx. Then import with the Model Editor, save and you're done.

How exactly this works is explained here:

<https://steamcommunity.com/sha...iledetails/?id=1970798355>

Theoretically, that would be great and much easier... if it weren't for the problems mentioned, among others 😄

It might work better with the next patch.

It continues with "Configure materials

2. Old way via TPF 1

You export for Transport Fever 1, make the vehicle halfway executable and convert it to [TPF 2](#) with the [TPF1 --> TPF2](#) [Konvhttps://www.transportfever.net/filebase/index.php?entry/4809-tpf1-tpf2-konverter/erter](https://www.transportfever.net/filebase/index.php?entry/4809-tpf1-tpf2-konverter/erter) and add a few new functions there.

The Blender Addon from Merk is required for exporting. It does not yet work with the new Blender 2.8. Animations cannot be exported either (possibly wrong info, animations might work), creating them in Blender is unnecessary. We create them manually later in the script.

Tutorial: [Blender Import/Export Addon](#)

Download: [Blender Import/Export Addon \(currently version 0.5.3 - alpha test\)](#)

Once it is installed, we still have to take some precautions. Otherwise we will get error messages.

In Blender you can already set all the properties of the train, link textures and configure materials. Since I'm still missing a lot of things and some functions don't work (animation export) and I only want to have the files for now, I'm only exporting dummies for now and will set up all the rest later.

I should only think about the number of materials, naming and allocation beforehand, as changing them later is awkward (you can also re-export later).

I usually use one material each for the outer wall, the outer transparent wall (window), the interior incl. inner wall (nothing transparent as window inside removed/cut away) alternatively one texture for inner transparency (window not cut out) and for the headlights (glowing).

To export:

- The model may only contain surfaces with triangles or quadrilaterals (not >4). To do this, select an object, export it, if the error message "contains ngons" is displayed and you then switch to Edit Mode, the causal faces are already marked. Then simply select Mesh - Faces - Triangulate faces and they will be correctly separated into triangles. With the Triangulate modifier, all faces are cut, even if they were allowed quadrilaterals, so the first variant is better.
- Most modifiers must be applied before export. The Edge Split modifier should/should not be applied. Triangulate does not need to be applied, but it can help if you want to fix hard edges caused by the modifier.
- We create a model, a group for each wagon and set it to the zero point. Now we also create a group for each bogie. The origin of the bogie group must be in the middle of the bogie (it rotates around the Z-axis). Shift + S and "Set Origin" help here.
- The bogie group is defined as the parent of the bogie meshes (select meshes, then group, then Ctrl + P). The wagon group is the parent of the wagon meshes and the bogie group. The model is the parent of a wagon group. For two wagons we have two models.
- For the wheel to rotate correctly, the origin must be on the Y-axis (axis of rotation) of the wheels. For the wheels, the "Axle" box is selected in the "Object Data" settings (to the left of Material) (could also be added later). For all other meshes (not bogie and wheels) the origin should be on the zero point. To do this, I transfer the position (possibly rotation and scaling) with Ctrl + A.
- Each mesh/surface needs a material assigned to it. In the Material tab, create a new one, name it, create a new texture in the tab next to it (the same texture for each material) and link our created texture as an image. In the material settings, our texture must also be specified for map_color_reflect for the standard material "Reflective". In Edit Mode, you can now assign different materials to the individual surfaces using Assign.
- We select Wagonmodell1, Wagongroup1, everything underneath (meshes, 2 bogie groups and their meshes) and export. In the export settings we select train as the type, you can also specify a subfolder (e.g. the name of the train) and so that we do not export the second model at the same time, we select only NGons and not Unselected.
- Later we do this with the second wagon and the second model. To simplify matters, we will use only one for now.
- Finished exporting. In the path you will now find the folder res, where our files are located.

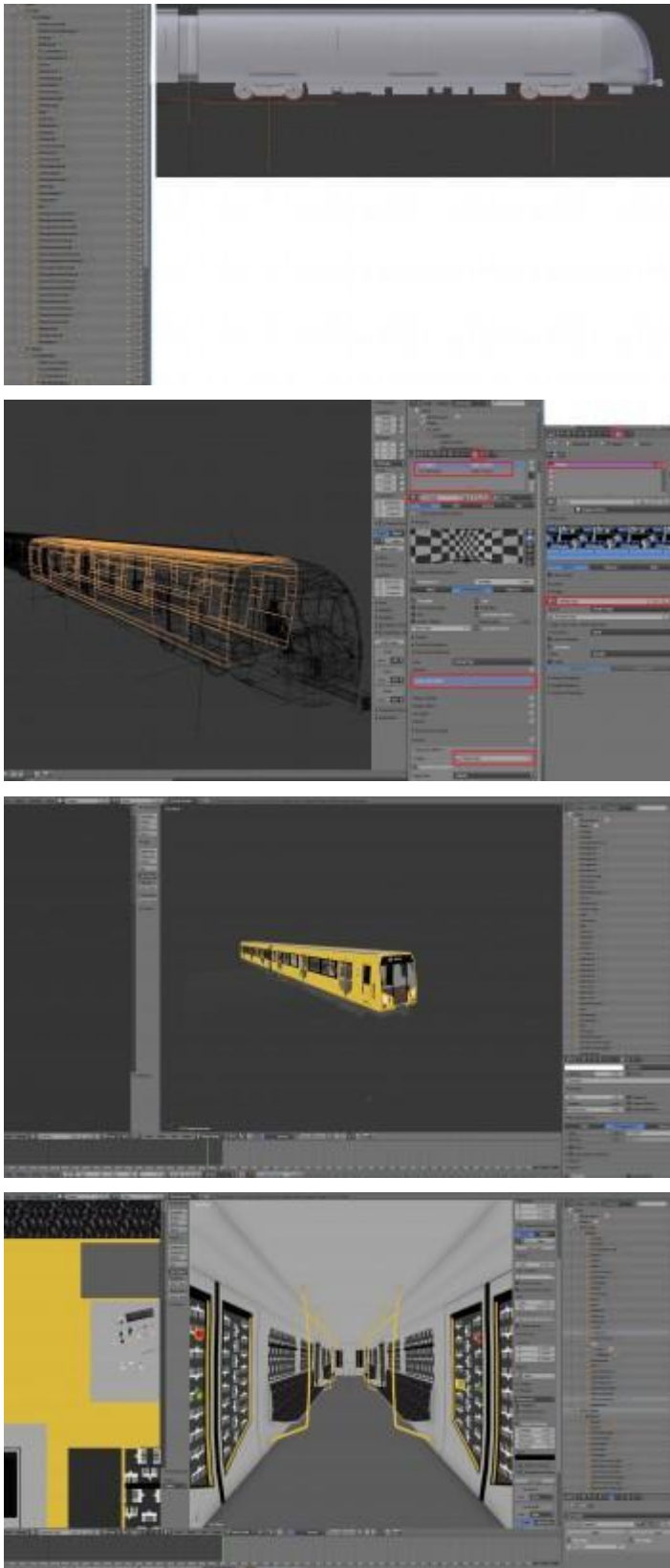


Figure 1: Origin for wheel and bogie (group) and correct parent assignment, but with many individual meshes (can be combined).

Figure 2: Material "Wagon" assigned to the outer surfaces and material "Interior" assigned to the inner surfaces. Taken over with Assign. Textures are also set for the material.

Image 3: With BVG H I have now named the objects correctly (Mesh is the most important, for an overview also the Blender object).

Picture 4: There is now also the correct grouping on the right side of the pedigree, I spontaneously added holding poles as well.

2.1 Check exported files

Let's now have a look at the individual files. We open the .msh files (models - mesh - vehicle - train - possibly subfolders) with a text editor (e.g. Notepad++) and find a link to our material there. In TPF 2 the material is linked directly in the .mdl for each mesh, in TPF 1 it was still in each .mesh file. Here it is helpful if you have already assigned and named the materials correctly in Blender. Then we don't have to change anything.

Next we look at the groups (models - group - vehicle - train - possibly subfolders). We also open these with the editor. There we make sure that the paths to our meshes are correct. Here we should also pay attention to upper and lower case. This is irrelevant for Windows, but not for Linux, which is why some users may experience crashes. With TPF 2, the group is no longer defined as an extra file, but directly in the .mdl with a kind of paragraph (bracket block).

The bogie groups are then also available and linked in the wagon groups. The many zeros and ones determine the alignment of the models. You could still change them here with the [transformation matrix](#). However, I recommend making changes directly in Blender and then exporting them again, so that everything in Blender always remains up to date.

If this fits, we open our model (models - model - vehicle - train). There we check whether the path to our wagon group is correct. We also have to insert values in the metadata. This includes for example availability, displayed name and description, capacity, speed, lifetime, or weight. It is best to copy this part from another mod and replace the values. Here is an example: [Mdl_Example_Basis.zip](#)

-1 means automatic calculation and is entered for purchase price and maintenance costs.

If we have not yet marked the wheels with "Axle" in Blender, we have to manually add the path to our wheel meshes.

Later we will add extras like passengers, turn traction, camera positions, or sound file here (after converting to TPF 2).

However, the basic metadata information must be available for the converter to convert correctly.

6. Configure Materials

Next, we take care of the materials. [Here is a list and their properties.](#) REFLECTIVE is no longer supported in TPF 2.

The material files are located at material - vehicle - train. If you want to put them in a subfolder, you have to make sure that the correct path to the material is in each .msh file (TPF 1 path) or in the mdl (directly TPF 2).

For the material Wagon Transparent I use
PHYS_TRANSPARENT_NRML_MAP_CBLEND_DIRT , for Wagon and Interior (no transparency) PHYSICAL_NRML_MAP_CBLEND_DIRT and for Emissive (luminous) EMISSIVE.

Additional info for EMISSIVE:

Luminous parts are textured normally on the main texture (.tga or .dds). EMISSIVE works behind transparent surfaces with extra parameters see: [Emissive behind glass](#)

Depending on the material type, the file now has different contents. It is best to copy these from other mods and replace only the path to the texture. Here are [example materials.zip](#)

map_albedo (_opacity for transparency) is our normal texture for which we specify the path (located at textures - models - vehicle - train - possibly subfolder; the path is specified without textures).

With the TPF 2 way, you can also enter the materials directly in the model editor. But I always take my standard files and replace the path via Notepad++.

There are now other required textures. These are Glossmap/mga (metal gloss ao), Dirtmap/cbr (cblend dirt rust) and the Normalmap.

The gloss map specifies how strongly which area should reflect (metal effect, reflection and ambient occlusion = shading). On the dirt map, you determine how much rust and dirt appears in which places and which places can be coloured with the colouring function.

With the normal map we can make places look plastic that are actually flat. Since we have modelled in a resource-saving way, we can use it to artificially create small unevennesses.

You can either create these textures now or use a placeholder. Then you don't have any reflections or dirt for a first test.

Simply place the placeholder textures (resolution 2048x2048) with the other textures, link the path and skip the next two chapters. [placeholdertextures.zip](#)

7. Create More Textures

Now we have to create the other textures mentioned above.
There are some useful dictionary entries for this.

[Textures and the channels](#)

[Glossmap and textures \(a second attempt\)](#)

However, for someone who has no experience in image editing, these are sometimes too complicated. I work a bit sloppy and not professionally myself, so I explain it here in my very simple way.

Each colour consists of the components red, green and blue. These are the channels. Each colour determines how strongly a characteristic is expressed.

According to the article, red is responsible for the shimmering metallic effect in the glossmap (metal gloss ao). I open my colour setting and set the slider for red between 0 and 255. You have to try out how strong the value has to be. For modern high-gloss vehicles, you usually don't need any metal at all (maximum value 30). Instead, you need stronger reflections/ambient reflections. Green is responsible for that. I want a strong reflection and use 210, for example. Blue indicates Ambient Occlusion. With this you can create shading on the texture. For modern vehicles without large indentations, this is usually not needed. There you simply choose 255 for the full colour strength.

For older vehicles with a lot of twists and turns, the AO map can achieve more realism and more plasticity.

How to create them is described, for example, in Seamon's guide

<https://www.transportfever.net...sten-Mod-erstellen-2-pdf/>.

Depending on the area, I mix my colour together. Some colour codes as an example: High gloss walls/windows (you can give more green) 15d7db, interior/seats/non-reflective things (no reflection, no metal effect) 0000ff, middle thing for window frames e.g. 0065e7.

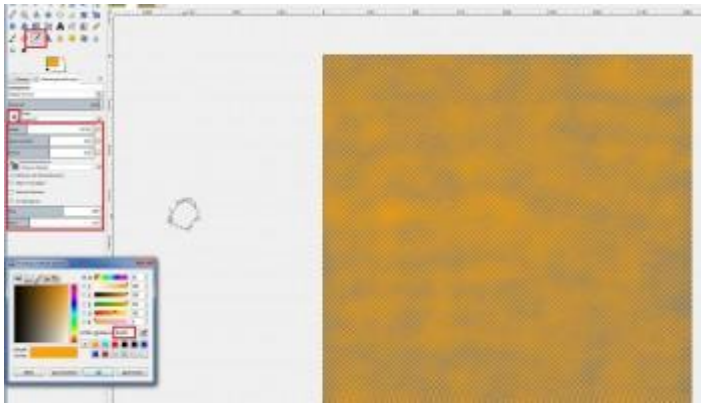
In the dirt map (cblend dirt rust), red is responsible for the colourability. I first take 255 (no colouring) and colour the first layer red. Then I take the normal texture in the second layer, select areas that can be recoloured and change them in the dirtmap (first layer) from red 255 to red 0 (black). You will then be able to colour these areas.

Now comes dirt (green) and rust (blue). I create dirt with the spray tool. For this I made a dirt layer that I can use again and again.

The colour is red 255 (cannot be coloured at first), green 162 (medium dirt) and blue 0 (no rust) → colour code ffa200. I choose blue 0 because I hardly have any rust on modern vehicles and it is negligible.

Personally, I also don't need dirt near the wheels, exhaust or on edges. For me, it is important that the whole vehicle becomes dark, dirty and blotchy. Therefore, quite simply a texture that I can use again and again.

In the picture are the settings for the spray tool. I move once horizontally and once vertically across the image at quite a fast speed.



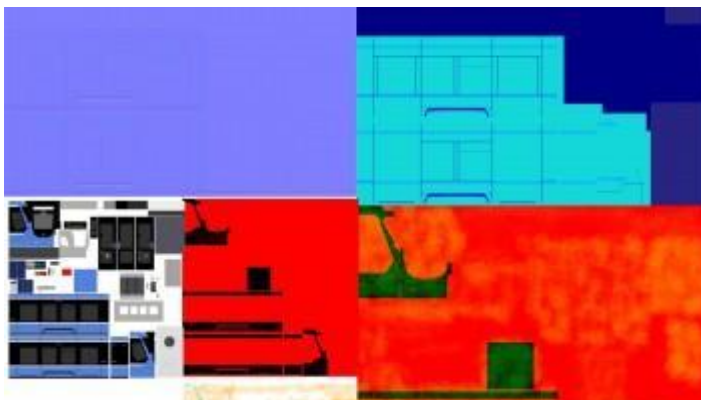
[dirtlayeruniversal.zip](#)

This dirt layer is now placed over the red/black layer. Now the problem would be that the colourable areas would be colourable, but the dirt with red 255 is not. We thus have orange on black and thus the areas on which dirt lies no longer have 0 with red. So we also select the colourable areas in the dirt layer and swap the colour (orange ffa200) with the colour without a red tone (turn red to 0, then we have 00a200).

Now the normal map. This is the purple map that adds artificial elevations.

Here, too, you could do it more professionally with a black/white texture (→ [Create NormalMaps](#)), but I make it easy for myself, take my normal texture, use the GIMP Normalmap plugin (download from the internet) and apply it. I make a few corrections (increase/decrease contrast for stronger/lighter effect; swap X and Y for indentation instead of highlighting or the other way round) and that's it.

Here are all the textures in the example and the dirt map divided into the two layers of origin (dirt and colourability).



Regarding the colouring function:

Often the colour in the game is not as strong as it appears in the selection menu. To do this, you have to find out the brightness of the colour to be coloured.

For me, blue areas can be coloured. With the colour picker I take the colour value and enter it into this converter to get the brightness.

<https://www.rapidtables.com/convert/color/rgb-to-hsl.html>

Now I calculate $0.5 * L$ value from the converter, for example $0.5 * 0.635$ and get the necessary factor. This is now inserted in the material, for example before the `polygon_offset{}`.

You can also play around in this range - you may find even better values. I then took 0.81, for example.

Code

```
1. colour_blend = { albedoScale = 0.787, },
```

8. Textures in DDS format

The .tga format used so far is OK, readable for Transport Fever and also works. With DDS, however, you have the advantage that the texture is compressed again and thus saves performance/storage space. Sometimes there is some pixel mud because several pixels are combined. Most of the time, however, this is not noticeable.

DDS textures

There is a DDS plugin for Gimp that you install. When exporting, you should merge all layers or delete unnecessary ones, otherwise the plugin gets confused.

If you really only have one (visible) layer, you have to mirror it vertically (Image - Transformation - Mirror Vertically). Now export it with the file extension .dds and choose the following settings depending on the texture.

For the normal texture (has transparency) DXT5 and "generate mipmaps".

For the mga/Glossmap, as well as for cdr/Dirtmap (no transparency) DXT1 as well as "generate mipmaps".

For the 3DC normal map and "generate mipmaps".

The textures now replace the old tga textures or the placeholder textures. In the materials, the path must now be corrected with the correct file extension .dds.

The model, UV map and textures are now ready. We have already made basic settings in the script. Before we add additional features such as animations, lods, reversible traction, or passengers, we should test the current one.

9. The first tests + TPF 2 conversion

(TPF 1 way only)

When changing files, the world must always be reloaded in Transport Fever to apply the changes. The whole thing can also be done a little faster via the Model Viewer (TPF 1).

<https://www.transportfever.com...dding:developerinfo:tools>

The Model Viewer comes in C:\Program Files (x86)\Steam\steamapps\common\Transport Fever

In order for the Viewer and Transport Fever to recognise our mod, we have to add a few small things.

The res folder goes into a folder named according to the following scheme:
Name_Modname_1 . In my case (Steamname marc345) marc345_C2Inspiro_1.

In addition to the res folder, a mod.lua file must also be included. This provides general information such as mod name, mod description or author.

To do this, copy an existing one again and replace it with your own information. [mod lua example.zip](#)

We now place our mod in C:\Program Files (x86)\Steam\steamapps\common\Transport Fever\mods

Click on " ..." in the Model Viewer and it is then visible there. You can now select a model and display it. There you will also find settings for dirt, colouring or passengers.

You can now see the colours, reflections and previous work steps as they look in the game. If there are discrepancies, they can now be corrected.

If the game or the viewer crashes, you should look at the error message in the viewer (has more error messages) or in the [stdout.txt](#) (if tested in Transport Fever). If they are unclear, it is best to ask.

Now we have our model ready in the Model Viewer. It has the materials with textures, basic metadata and would be ready to drive in TPF 1.

We now convert it to TPF 2 with the [TPF1 --> TPF2 converter](#).

In TPF 2, the mod must be in the staging area for the editor to recognise it.

10. Completing the basic functions

So. We are now back at the same level, regardless of the path, and are in the Model Editor (TPF 2) with a functioning model.

In the depot, line manager etc., purple placeholders are currently still used for our train. In the Model Editor, we can use the Screenshots button to create the Ui images (they are saved directly and correctly).

If everything now fits, we can add the remaining functions.

11. Multiple Units

For multiple units (railcars) an extra file is created (text file with extension .lua Save), in which several .mdls are assembled to a railcar.

You can also mirror wagons with forward = false there.

It is located at config - multiple unit and could look like this:

Code

```
1. function data()
2. return {
3.   vehicles = {
4.     { name = "vehicle/train/C2_Head.mdl", forward = true },
5.     { name = "vehicle/train/C2_Wagon.mdl", forward = true },
6.     { name = "vehicle/train/C2_Wagon.mdl", forward = true },
7.     { name = "vehicle/train/C2_Wagon.mdl", forward = false },
8.     { name = "vehicle/train/C2_Wagon.mdl", forward = false },
9.     { name = "vehicle/train/C2_headback.mdl", forward = false },
10.
11.
12.   },
13.   name = _("Siemens C2 Inspiro"),
14.   desc = _("The Siemens C2 Inspiro is the newest train in the Munich underground
              system. It was built in 2013 and took 3 years to enter service. The train is the
              successor of the C1 and has received improvements in comfort and visibility. The
              'please stay back' announcement is missing.")
15. }
16. end
```

Display More

Multiple units can only be bought in the depot in Transport Fever - the Model Viewer does not support them.

In order for the individual wagons to no longer be purchasable in the depot, MultipleUnitOnly = true must be set in the .mdl files.

12. Animations

Attention: Animations have changed in TPF 2. You can use the old notation or outsource the animations with the new notation.

[Animate doors](#)

Animations are dangerous. If you haven't had a crash yet, you will have one now at the latest. As soon as an element is wrong here, it no longer works.

But animations are also nice. You can really put a lot of time into it, add glowing LEDs and create a door-closing spectacle.

In the C2 Inspiro, I have animated all the door lights (as an extra object so that they are independent) (6 per side: red inside, green inside, red right door, red left door, green right door, green left door).

But it can also be simple - let's start with that.

In the Mdl, an animation is automatically triggered for the events. This is written directly into the .mdl for the respective mesh.

You can rotate and move objects.

That's what we'll try now. We start in the mesh file of a door. If, for example, you have defined all the right doors of the right side as one object because you want to animate them immediately, we save work.

We then only have to enter the animation at one Msh.

Code

```
1. animations = {  
2.   close_all_doors = {  
3.     forward = true,  
4.     params = {  
5.       keyframes = {  
6.         {
```

7. red = {
8. 0, 0, 0,
9. },
10. time = 0,
11. transl = {
12. -0.7, -0.04, 0,
13. },
14.
15. }, {
16. red = {
17. 0, 0, 0,
18. },
19. time = 7000,
20. transl = {
21. -0.7, -0.04, 0,
22. },
23.
24. }, {
25. red = {
26. 0, 0, 0,
27. },
28. time = 8500,
29. transl = {
30. -0, -0.04, 0,
31. },
32.
33. }, {
34. red = {
35. 0, 0, 0,
36. },
37. time = 9200,
38. transl = {
39. 0, 0, 0,
40. },
41.
42. }, {
43. red = {
44. 0, 0, 0,
45. },
46. time = 9400,
47. transl = {
48. 0, 0, 0,
49. },
50.

```
51. },
52. },
53. origin = {
54. 0, 0, 0,
55. },
56.
57. },
58. type = "KEYFRAME",
59.
60. },
61. open_all_doors = {
62. forward = true,
63. params = {
64. keyframes = {
65. {
66. red = {
67. 0, 0, 0,
68. },
69. time = 0,
70. transl = {
71. 0, 0, 0,
72. },
73.
74. }, {
75. red = {
76. 0, 0, 0,
77. },
78. time = 500,
79. transl = {
80. 0, 0, 0,
81. },
82.
83. }, {
84. red = {
85. 0, 0, 0,
86. },
87. time = 1000,
88. transl = {
89. -0, -0.04, 0,
90. },
91.
92. }, {
93. red = {
94. 0, 0, 0,
```

```

95. },
96. time = 1100,
97. transl = {
98. -0, -0.04, 0,
99. },
100.
101. }, {
102. red = {
103. 0, 0, 0,
104. },
105. time = 2800,
106. transl = {
107. -0.7, -0.04, 0,
108. },
109.
110. },
111. },
112. origin = {
113. 0, 0, 0,
114. },
115.
116. },
117. type = "KEYFRAME",
118.
119. },
120.
121. },

```

Display More

Each block is a keyframe. There you specify at what time the object is located where. Starting position 0.

After some intermediate steps, the door is shifted by -0.7 metres in X and by -4 centimetres in Y after 2800 milliseconds. The best way to find out the values is to use Blender.

For the time being, only the start and end positions are sufficient for testing. You can always add intermediate steps for a rounder opening.

In the animation you sometimes see double entries where the door does nothing for a while. This can be done, for example, for the synchronisation with the door sound.

The Mdl becomes quite long because of the animations. To simplify it and make it easier to change afterwards, you can outsource the animations (recommended).

Code

```
1.  animations = {
2.
3.  open_doors_right = {
4.  params = {
5.  id = "vehicle/tram/Avenio/open_doors_right/door-right-rear.ani",
6.  },
7.  type = "FILE_REF",
8.  },
9.  close_doors_right = {
10. params = {
11. id = "vehicle/tram/Avenio/close_doors_right/door-right-rear.ani",
12. },
13. type = "FILE_REF",
14. },
15. },
```

Display More

In the outsourced Ani file it looks like this:

Code

```
1.  function data()
2.  return {
3.  times = { 0, 800, 2300, 3000, 3200, 3800},
4.  transfs = {
5.  { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, -0.7, -0.04, 0.0, 1, },
6.
7.  { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, -0.7, -0.04, 0.0, 1, },
8.  { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0.0, -0.04, 0.0, 1, },
9.  { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0.0, 0.0, 0.0, 1, },
10. { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, },
11.
12. { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, },
13.
14.
15. },
```

```
16. }  
17. end
```

Display More

Now you can test the animation in the model editor. If it crashes (animation end error), the number of keyframes is probably not equal to the number of timesteps.

If everything works, you can programme the rest of the door objects. Between the meshes, usually only the sign changes during animation.

13. Model: LODs, Passengers, Camera Positions, Turning Train Feature, Train Destination Displays

In LOD 2/the second copy, detailed meshes are deleted that are not seen at further distance anyway, e.g. stop bars, interior displays, driver's table, or seats.

In the .mdl, the block at lods is then copied for each LOD.

I usually only do two lods because I don't have a lot of details that need to be/can be hidden. You can also do 3 or more.

At visibleFrom/To you enter at which distance which LOD is visible.

I usually use 0-250 and 250-2500.

With the axles you also have to add a second lod, otherwise the train won't go round the bend smartly when you look from further away. (Example file at the end of the chapter)

Passengers are so indicated:

Code

1. seats = {
2. { group =2, transf =
transf.scaleRotZYXTransl(vec3.new(0.97,0.97,0.97),transf.degToRad(90.0, 0.0, 0.0),
vec3.new(13.9, -1, 0.95)),standing = false, crew = true},
3. { group =2, transf =
transf.scaleRotZYXTransl(vec3.new(0.97,0.97,0.97),transf.degToRad(90.0, 0.0, 0.0),
vec3.new(13.9, -1, 0.95)),standing = true, crew = false },
4. },

You can add as many entries as you like here. The group is a MeshId. If the passenger is in the first car, you should enter a mesh here that is in the group of the first car.

You can see the MeshId in the Model Editor.

Just open the .mdl file with it and it shows all the Ids of the groups/meshes.

with transf.scale... you can scale the passengers, rotate them with degtoRad, place them with vec3.new (positions).

Standing = true, whether they should stand (otherwise they sit)

crew = true, whether it is the driver (otherwise normal passenger).

We also need to specify this (lines 1 and 4 are already there) at the beginning of the mdl:

Code

1. function data()
2. local vec3 = require "vec3"
3. local transf = require "transf"
4. return {

Here is an extra encyclopaedia entry: [Passengers and crew members in vehicles](#)

Camera positions can be set with this:

Code

```
1. cameraConfig = {  
2.   positions = {  
3.     {  
4.       group = 0,  
5.       transf = transf.rotZYXTransl(transf.degToRad(0.0, 0.0, 0.0), vec3.new(16, 0.0, 2.3)),  
6.       fov = 25  
7.     },  
8.     {  
9.       group = 0,  
10.      transf = transf.rotZYXTransl(transf.degToRad(180.0, 0.0, 0.0), vec3.new(18.5, 0.0,  
11.        2.4)),  
12.      fov = 40  
13.    },  
14.  },  
}
```

Display More

Group specifies the MeshId/GroupId (Model Editor).

fov is the field of view - the larger, the larger the image section up to the fisheye.

Rotation and position of the camera as before in the rear brackets.

Now the **reversible train feature**.

Here is a Wikipedia entry <https://www.transportfever.com...models:reversiblevehicles>

Normally, trains are turned around at the terminus and do not simply change direction.

With the reversible train feature, this can be changed.

To do this, set reversible = true in every .mdl that has an entry for engine. It is best to write this in all of them.

You can also use it to set changing pantographs, for example.

Instead of configuring a front wagon with white lights and a rear wagon with red lights, we now use a universal wagon.

Insert white and red lights here. Without additional settings, everything would be visible at the same time.

They can be used to determine which objects are hidden and which are shown again at the right time. An explanation can be found in the linked Wikipedia entry.

The Ids from the Model Editor are used again as Ids.

Train destination displays are explained here: [Destination displays](#)

Unfortunately you can't test them in the Model Editor 😞

You always have to restart the game.

With the Common API, you can also make variable displays here.

Here is the **finished Mdl as an example**. [Mdl_Example_Finished.zip](#)

The soundset is already entered here and is explained below.

14 Sound

[Sound Set](#)

The sound adds the final touch and gives the vehicle its own character.

This requires audio files and a sound configuration file.

The configuration file (.lua file) comes in config - sound_set.

We then link the name in our .mdl file (already done in the example).

Before we configure the sound file, we need the audio files.

Ideally, you record them yourself.

There are also videos on Youtube where you can use sound recordings with the author's permission.

If all else fails, you can simply use the default files from the game.

You can include the following sound files:

Unique sounds:

1. door open
2. door closed
3. horn

Repeatable sounds. A short section is repeated in an endless loop.

The difficulty is to find a range that sounds good in the transition from the end to the beginning and has no jerks or noise. This means that you cannot have accelerations in a sound.

We therefore have to split our acceleration into several constant pitches and then combine them as nicely as possible so that we get an even acceleration sound. [Here is a tutorial on how to make good loop transitions in Audacity.](#)

These files are used for:

1. standing noises
2. driving noise
3. acceleration noises

To create such audio files, I use Audacity - also free of charge.

All files must have only one mono track. To do this, you can use Tracks - Convert Stereo Track to Mono in Audacity.

With transport - loop playback, you can sample the repeatable sounds.

The paths for the audio files are now specified in the sound file. Events such as door open, close and horn can be easily specified with volume and audible distance.

A track is created for each of the repeatable sounds. There you specify the volume and the pitch depending on speed or power.

Notes:

You can also leave out the horn. Then it is not annoying.

In order for the door beep to be heard evenly everywhere, it could be played in every carriage. But then the sounds would overlap. I have therefore placed a dummy MDL in the exact centre of the railcar, which consists of a transparent rectangle and only plays sound.

The standard clacks and brake squeals can also be added.

When configured as a function of power: Power is only positive power here, i.e. acceleration. Braking is 0 and therefore does not lead to an adjustable sound.

Here is a commented example of the Avenio, everything is together here: [Soundconfig.zip](#)

15. Publish Mod

[Steam Workshop - Publish Modification](#)

For the Steam upload, this is a good guide.

Here in the forum you only need to upload the packed version of the mod.

After all the work, it would now be wasted time if we skimmed on the description and pictures.

Take some screenshots with the Model Viewer and insert them in the entry. A lot can also be done in-game with the camera tool.

Some Youtubers also do mod shows more often.

I hope this guide is helpful and could contribute to the understanding of Transport Fever mods and provide guidance for new modders.

I would be very pleased to receive feedback, suggestions for improvement, comments on understanding etc. so that this guide can be constantly improved and kept up to date and all the work is not in vain.

Thank you for reading.

Greetings, Marc