PROTON BUS SIMULATOR

www.protonbus.com.br

Map modding system

Updated with v252 - 2020 April **(TRANSLATION IS UNDER REVISION)**

by Marcos Elias

Introduction

The second step of map modding support on **Proton Bus** allows maps to have passengers, traffic and more! This brings the simulator to another level. Basically there are two Protons: before and after this feature. It's almost as a new game!

Due to our very limited time, we cannot provide individual support about converting or editing mods. We recommend that you try it by yourself, by looking at the samples and other maps and checking their file structure. Please post your questions and related problems at our Facebook groups, so solutions can be shared between more people, and other members may also help you.

If you are still not subscribed to the founder's channel, please, join at:

www.youtube.com/marquinhosxp

English channel:

https://www.youtube.com/channel/UC_Us4lbIWRvQPNzJK5UwAtQ

Official PBSU fanpage on Facebook: https://www.facebook.com/protonbusoficial

And for PBSR: https://www.facebook.com/protonbusrod.oficial/

Facebook Groups

Special group for mod creators:

https://www.facebook.com/groups/pbsmods/

Please, avoid posting unrelated things there, it's only for the support between other creators.

Groups for users:

These groups may have more variated posts, like screenshots, gameplays, tips and so on:

International group:

https://www.facebook.com/groups/1264008273712815/

Official Brazilian group:

https://www.facebook.com/groups/2238192089740569/

Older fans groups:

https://www.facebook.com/groups/624645387735090/

https://www.facebook.com/groups/1678009835826675/

Feel free to join to every group! But the chances to receive support in English are higher on the international one.

Tip of a great website to get textures for mods:

http://www.textures.com

Some notes related to the first step

Before creating a map on the second step, it's important to understand all things related to the basics of the first step. If you do not know it yet, don't worry, it is possible to follow this guide from scratch while you check the sample map. The most important requisite is knowing well basic commands of Blender, because we will use this program to export maps.

Older documentation refering the first step is here, for maps:

http://blog.protonbus.com.br/2019/09/experimental-mods-de-mapas-no-pbs.html

And for buses:

http://blog.protonbus.com.br/2018/09/primeira-fase-do-sistema-de-mods-de.html

It is better having always the latest version of the game! Older versions are not supported anymore, since the beta is constantly changing, we always consider only the most recent one. You may find recent builds at http://pbsu.busmods.com for the urban and http://pbsr.busmods.com for the coach edition.

Basically we will use **Blender 2.79** to produce the scene, export the model using the 3ds format and create the directory structure for folders and files for Proton Bus.

IMPORTANT: USE BLENDER 2.79!

The needed exporter do not work on current versions of Blender 2.8, it seems abandoned and no one could get it working yet. It maybe that someday in the future it will come back renewed for new Blender versions, but for now we all must use 2.79. You may download Blender 2.79 here:

https://download.blender.org/release/Blender2.79/

The latest version generally ends with a "b" letter. Platform is up to you, depending on the system that you are using (Windows, Linux or Mac) and architecture (32 or 64 bit).

If you have already started creating the map on 2.8, we would suggest that you export it to some intermediate format that 2.79 could open and then export from it to the game.

VERY IMPORTANT: MODIFY THE 3DS EXPORTER FOR PROTON BUS!

The default 3ds exporter from Blender truncates all objects and textures names up to 12 characters. Normally we will need more than that to avoid problems with some pieces and/or textures.

Download this edited plugin for Blender to export it (this file is for Blender 2.79):

http://proton.viamep.com/coisas/export_3ds_protonbus_blender279.zip

Tutorial on how to replace the plugin (this video is in Portuguese, sorry, but we hope that you can follow it): https://youtu.be/0EokFmSjGdA

If you do prefer, instead of downloading this changed file you may edit yourself the **export_3ds.py** file located at **scripts\addons\io_scene_3ds** of your Blender (locate it there at Program files, or wherever you have extracted it, if you have used the zip without installer).

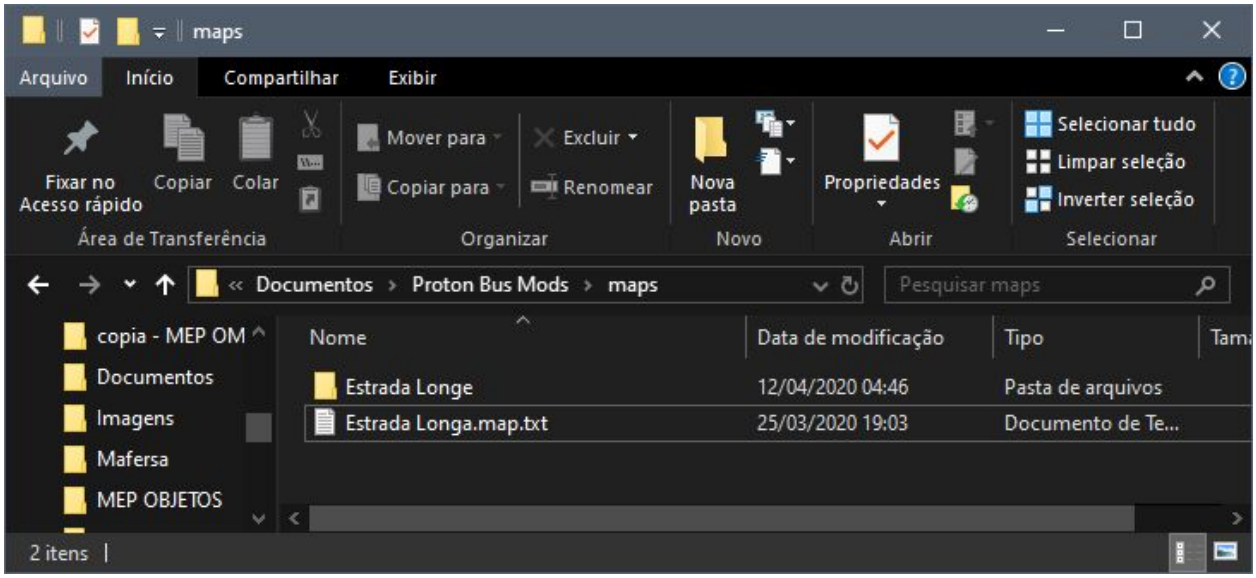
Search for **[:12]** and replace that **12** with a bigger number, like **999**. This is the character limit. WIthout doing this step exporting to the game may fail, since it limits all objects and textures names to a max of 12 characters by default. This is very important, do not forget it!

TIP: WHEN DOWNLOADING BLENDER YOU MAY CHOOSE THE PORTABLE VERSIONS AS ZIP, TO KEEP MANY VERSIONS AT THE SAME TIME. GENERALLY THE INSTALLER SETUP WILL REPLACE IT, KEEPING ONLY ONE. DOWNLOADING IT AS ZIP, YOU COULD KEEP MANY.

The file structure for the map mod

Maps in Proton are composed of many txt files (pure text), and some folders. The primary structure is a txt file that defines some properties and a folder. Both must be placed inside the “maps” directory, and it is inside the mods folder. The mods directory generally is Proton Bus Mods inside the user’s documents folder, on PC; and the folder com.viamep.p.../files inside Android - data, on mobile.

A sample of the structure on PC:



The txt definition file must ends with **.map.txt**. The game searches for all .map.txt files that are inside this folder to list all available maps at the selection screen. From the selected file it will read all definitions for this map and so it will know where are the needed files placed at: models, textures, and so on. Generally you may use the name of the map or region, route etc, as on the sample above, **Estrada Longa.map.txt** (note: this name in Portuguese means something like “Long road”). It should be a unique name, that other people will not use! If someone else creates a map with the same name of an existing one, it will be replaced when they install it. Avoid

common names. A great tip is including a specific prefix or suffix, like the author or team name, for example.

VERY IMPORTANT: NEVER USE ACCENTS OR SPECIAL CHARACTERS ON FILENAMES! THEY MAY WORK ON YOUR COMPUTER, BUT PROBABLY THEY WILL FAIL TO BE LOADED WHEN USED IN SYSTEMS WITH OTHER LANGUAGES.

Inside this txt we have a basic structure like this:

[map]

baseDir=Estrada Longe << The base map directory, that stays together with its .map.txt

modelsDir=Rota 1 << The models directory, that stays inside the **tiles** folder, this one inside the baseDir defined above

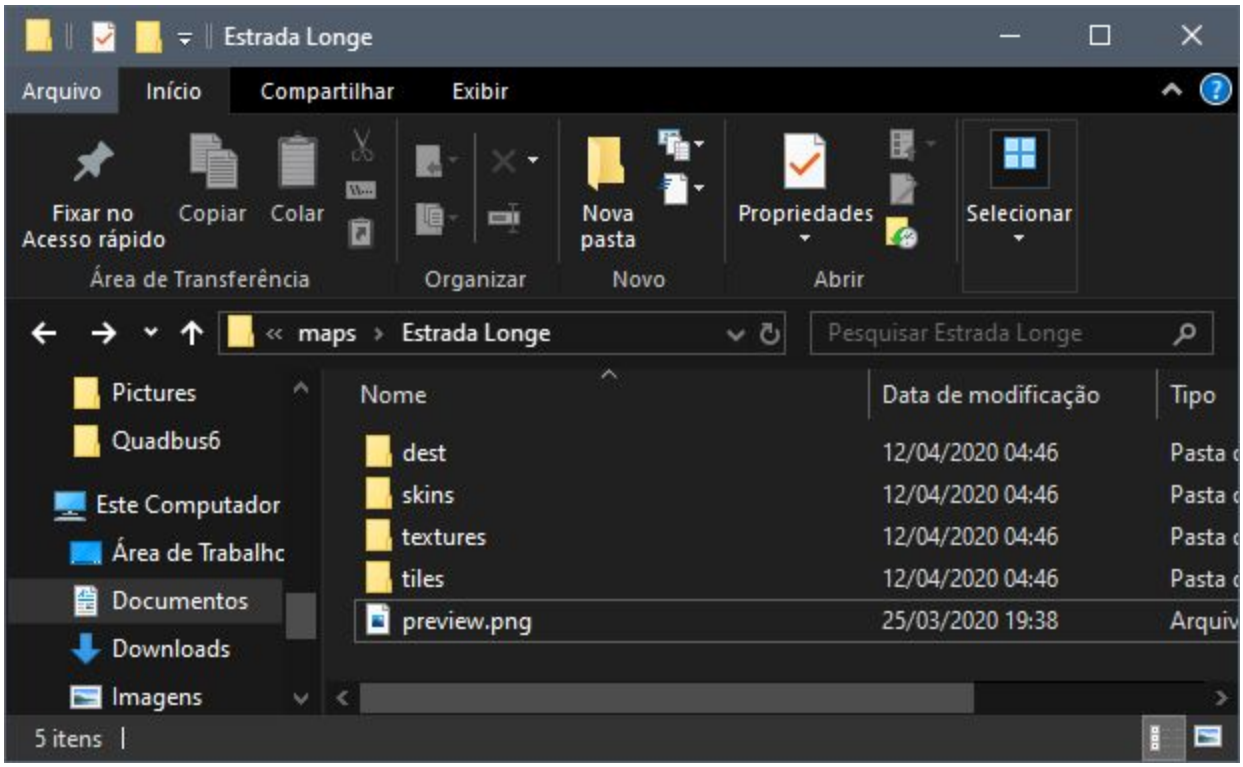
textures=textures << The textures folders, that must be inside the baseDir

mapModVersion=2 << The version of the modding system, that informs to the game to show a message for users if they try to load a map of the future with an old game version

IMPORTANT: FOR THE SECOND PHASE USE NUMBER 2 HERE!

preview=preview.png << The picture that will be shown at the selection screen. It must be placed inside the base directory, not on textures. If you do not use this item, the game will look for a picture named as “preview.png”. We recommend the 16:9 proportion for this picture. A good size is 640x360 pixels. It do not need to be too large.

Inside the base directory of the map there should be other folders, like these ones:



The **dest** directory will keep destinations displays and route plates. It will have subfolders with destinations defined later, and inside each destination folder, all its related pictures.

The **skins** folder, as the name suggests, will store the paintings for buses. It should exist one subdir named **0** (zero) inside it, and inside the 0 folder there will be folders of all buses (only **pbc**, for now). Skins will be randomly selected inside the running game.

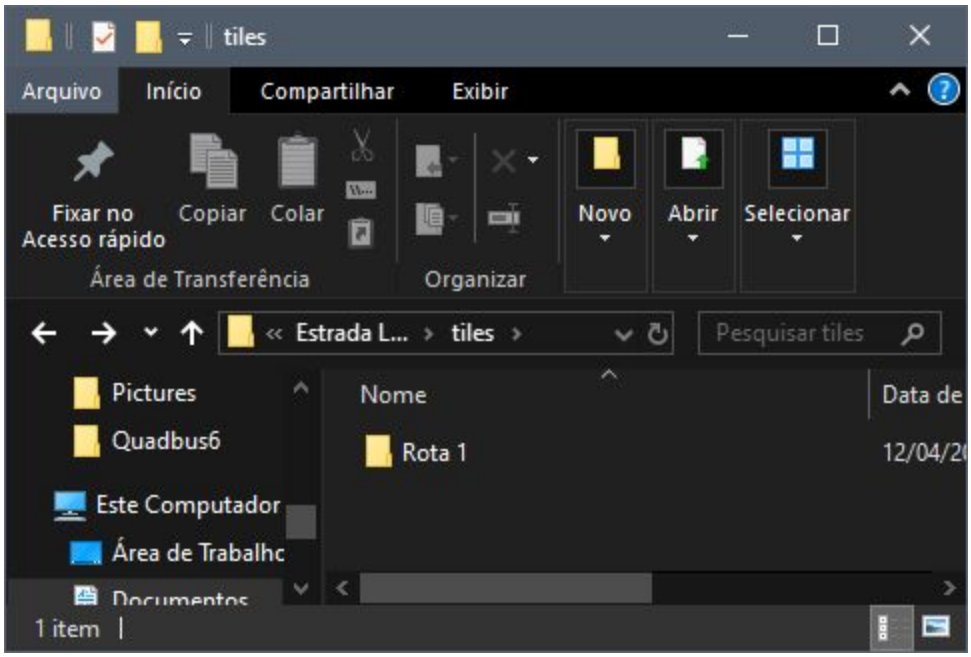
The **textures** folder, as its name suggests, will keep all textures of the objects.

VERY IMPORTANT: AVOID JPG TEXTURES! THE GAME MAY TRY TO LOAD THEM, BUT IT WORKS BETTER WITH PNG. SOME JPG MAY BE PINK OR CANNOT BE LOADED, SPECIALLY ON SOME MOBILE DEVICES.

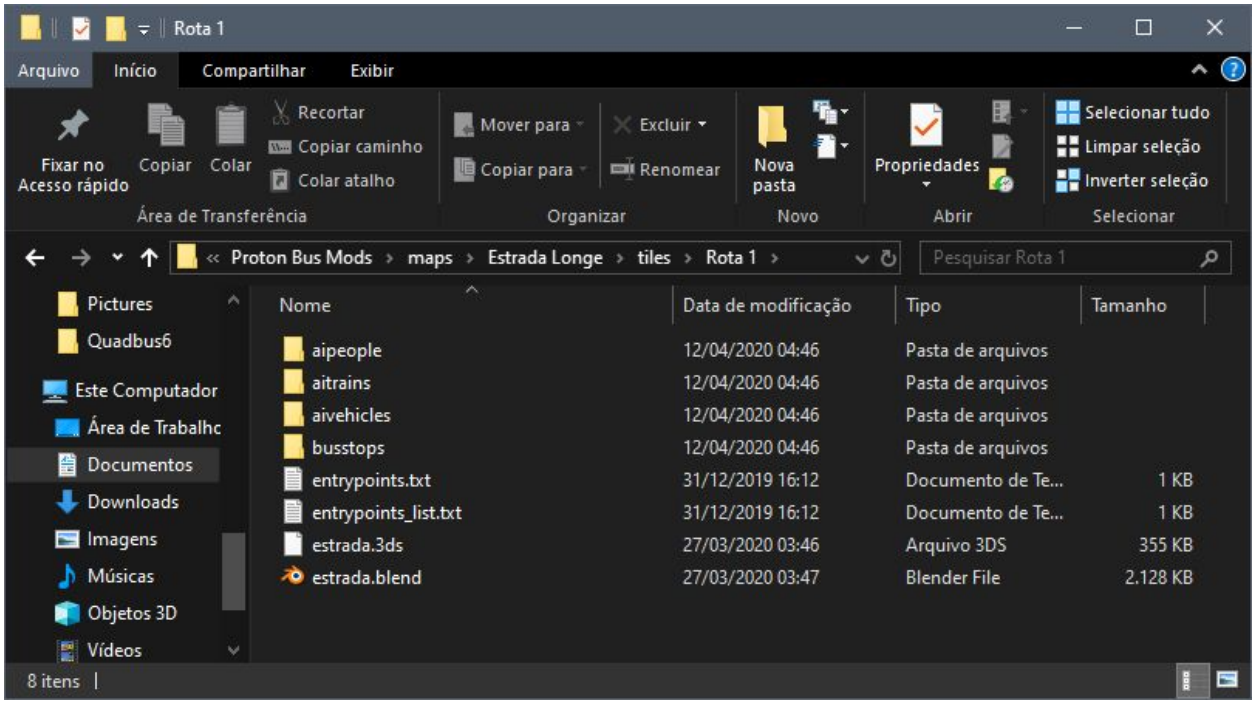
The **tiles** directory must have this specific name. The models directory will be placed inside it. The models directory name is that defined on the .map.txt file on the **modelsDir** setting. This structure was made thinking in an eventual continuous loading in the future. It won't be guaranteed nor promised, it's just an internal study. We will avoid making streaming for map mods due to hiccups that may occur when loading them.

IMPORTANT: ALL MAP MODS ON PROTON WILL BE LOADED AS A WHOLE MAP! THEY CANNOT BE TOO HEAVY, WITH MANY ROUTES. IF YOU PLAN TO MAKE MANY DIFFERENT BUS ROUTES, PLEASE, DIVIDE THEM INTO PARTS, BY SEPARATING EACH ONE TO A SPECIFIC "MAP". LEAVE EACH ONE WITH ITS OWN .MAP.TXT FILE, SO USER WILL LOAD ONLY THE NECESSARY OBJECTS. THIS MAKES IT MORE PERFORMANT TO RUN BETTER, SPECIALLY ON MOBILE.

At our sample, inside the tiles folder there is a folder named **Rota 1**, that was defined at modelsDir inside the main txt file of the map.



Inside that models directory is where the exported map objects must be placed at (3ds files). Also there will be two txt files that will hold information about the available places where the bus could be positioned, and other specific folders:



All map scenery must be exported to .3ds in Blender and placed inside this directory. You may divide your model into layers or different files, and place them in this directory.

TIP: SAVE THE BLEND FILE DIRECTLY TO THIS FOLDER, SO IT WILL BE EASIER WHEN EXPORTING IT! DELETE THE BLEND FILE WHEN RELEASING YOUR MAP, IF YOU DO NOT WANT THAT OTHER PEOPLE EDIT IT.

BY THE WAY, VERY IMPORTANT: MODS FOR PROTON BUS ARE NOT PROTECTED! TECHNICALLY, NO GAME COULD PROTECT ITS MODS! IF IT APPEARS ON THE SCREEN, THERE ARE PROGRAMS THAT COULD EXTRACT ALL MODELS DIRECTLY FROM THE GRAPHICS CARD MEMORY. SO WE WILL NOT ENCRYPT OR MAKE A PROTECTION SYSTEM, IT WOULD BE USELESS, IT'S JUST A MATTER OF TIME TO SOMEONE BREAK IT. THE ONLY WAY TO NOT HAVE YOUR MAP EDITED BY OTHER PEOPLE IS NOT RELEASING IT.

The content of this folder, besides 3ds and blend files, are these items:

aipeople << Folder with settings for pedestrians, walking people

aitrains << Folder with settings for trains

aivehicles << Folder with settings for cars on the AI traffic

busstops << Folder with settings for bus stops and passengers

entrypoints.txt << Folder with settings for entry points and positions where the bus may be placed at

entrypoints_list.txt << A simple list with names of all entry points

Items inside these directories are directly related to some named objects inside the 3D files. The game will read those files and search for objects with suggested names when linking these information. For example, inside the aivehicles there are txt files with information about the traffic system paths for cars, where each road or avenue, or segment, will have its own name. The game will read this name and search on the loaded models where they are at, so it will generate the code that will make that cars could appear at those places. It's your responsibility keeping all names unique and readable, without generating confusion when setting them up. We will see more tips soon.

Setting up entry points

Right after selecting your map users may select the route, bus line or destination, or a place to enter (it could be a bus depot, some road etc, not necessarily a route). This entry point will be identified as **entrypoint**. You could have as many as you want of them inside the map.

IMPORTANT: PLEASE REMEMBER TO NOT CREATE MAPS WITH MANY ROUTES! FOCUS MORE ON SCENERY THAT USER MAY PASS! OTHERWISE PERFORMANCE MAY BE COMPROMISED, SPECIALLY FOR MOBILE PHONES, SINCE THE ENTIRE MAP WILL BE LOADED AT ONCE.

Entrypoints must be named in a unique form, in such a way that no other object on the map have the same name, to avoid that the code find the wrong object. A suggestion is to include something that identifies the route and the direction of the trip, for example, as on native maps:

351F-10 TP << The initial bus stop of the route, the main terminal

351F-10 TS << The last bus stop of the route, the secondary terminal

Garagem Maria Amalia << A sample of a bus depot name

IMPORTANT: NEVER USE ACCENTS OR SPECIAL CHARACTERS ON THESE NAMES! THIS AVOIDS MANY PROBLEMS WITH THE GAME WHEN USED FOR PEOPLE WITH OTHERS LANGUAGES. IT IS NOT ALLOWED ALSO USING FORWARD OR BACKSLASH, SINCE THEY ARE USED AS DIRECTORY SEPARATOR ON MOST OPERATING SYSTEMS.

USE ONLY LETTERS FROM A TO Z, NUMBERS FROM 0 TO 9 OR A SIMPLE DASH OR UNDERSCORE.

If you do not want to follow the same São Paulo style names that we use, it's ok, you may use something like the Uipe map:

77-01 << The route 77, turn 1, going to its final destination

77-02 << The route 77, turn 2, going back to where it started

You cannot use repeated names, like “77” only. By the way you may use “77 going” and “77 coming back”. Those names will also be used for destination display directories.

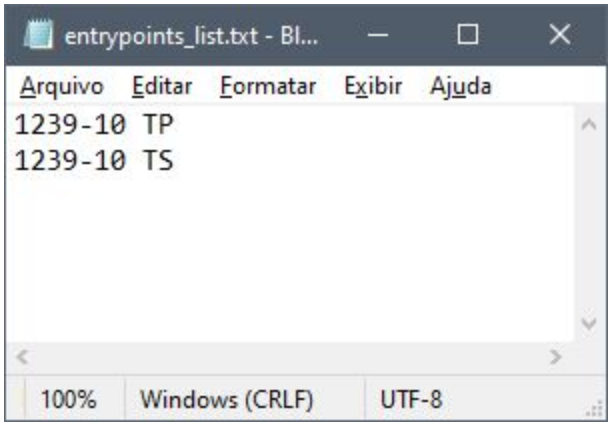
IMPORTANT: NO OTHER OBJECT INSIDE THIS MAP COULD HAVE THESE NAMES! OTHERWISE THE GAME MAY SELECT THEM WHEN TRYING TO LOOK FOR THE ENTRY POINT. SO IT IS REALLY GREAT USING AN UNIQUE NAME.

Do not use, for example, 001, Cube, Road, etc. It must be an unique name for the specific entry point.

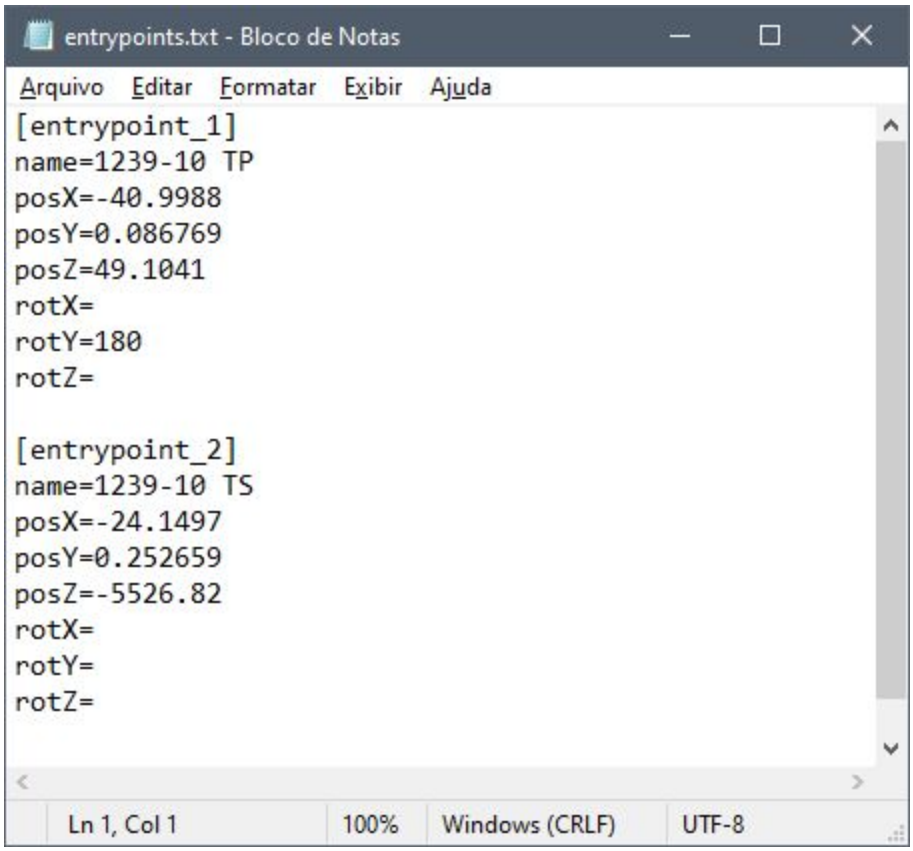
AGAIN, WE ASK: DO NOT USE SPECIAL CHARS, QUOTES, DIFFERENT CHARS OF ANY KIND! IT MAY WORK FOR YOU ON YOUR DEVICE BUT THOSE CHARS MAY BRING PROBLEMS FOR OTHER PEOPLE WITH SYSTEMS USING OTHER LANGUAGES. THEY MAY BREAK THE GAME.

So, defined all names, let’s go setting up the files!

The **entrypoints_list.txt** file must keep a simple list of all defined names, one at each line. It will be used by the game to show the routes list on the selection screen dialog for the user, as well as on the repositioning bus screen selection too.



But the **entrypoints.txt** is a little bit more complex. It will mark, in fact, the physical place where the bus should appear.



VERY IMPORTANT: PLEASE REMEMBER THAT Y AND Z AXIS ARE INVERTED BETWEEN BLENDER AND UNITY! ON BLENDER THE Y AXIS IS FOR FORWARD/BACK, AND Z IS FOR UP/DOWN. BUT ON UNITY, THE ENGINE THAT WE USE, Y IS FOR UP/DOWN AND Z IS FOR FORWARD/BACK! ALWAYS REPLACE Y WITH Z AND VICE VERSA WHEN COPYING POSITIONS BETWEEN BLENDER AND PROTON. X IS NOT CHANGED.

The **entrypoints.txt** file will have a list with incremental sections of type entrypoint_1, entrypoint_2, entrypoint_3 etc… As many as needed. On the sample map shown, there are only two of them.

IMPORTANT: IF YOU ADD OR REMOVE NEW ENTRYPOINTS LATER, PLEASE UPDATE THIS FILE, KEEPING THE NUMBER IN A INCREMENTAL MANNER, STARTED AT 1.

The entrypoint structure is as follows:

[entrypoint_1] << The entrypoint identifier for this item

name=1239-10 TP << The name of the entrypoint, the same used at entryponts_list.txt and also it will be the same of the folder for destination displays

posX=-40.9988 << The position on X of the entrypoint on the 3D model

posY=0.086769 << The position on Y of the entrypoint on the 3D (Z on Blender, for the height!)

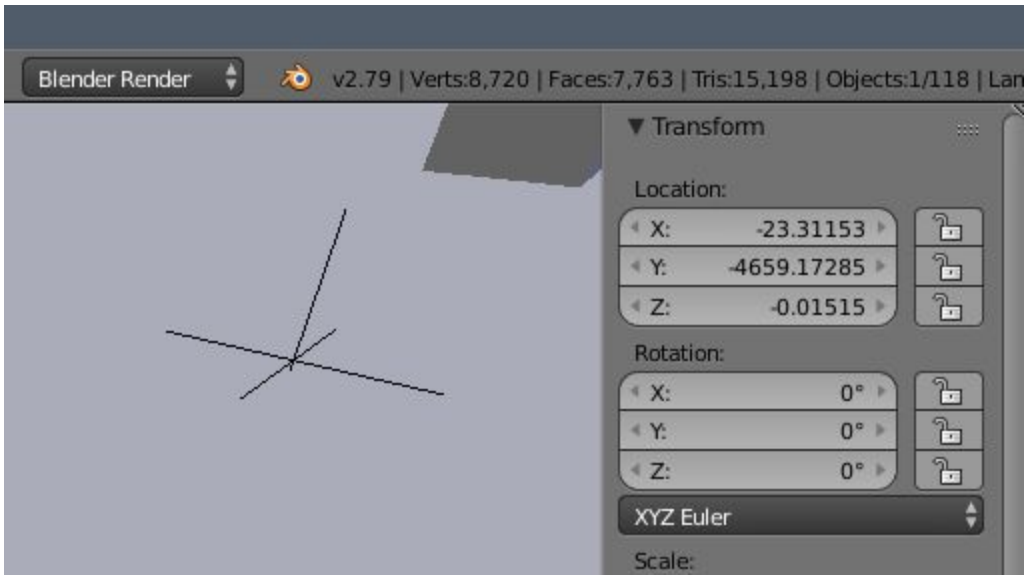
posZ=49.1041 << The position on Z of the entrypoint on the 3D (Y on Blender!)

rotX= << The rotation on X on the 3D, generally this will be always 0

rotY=180 << The rotation related to the Y axis (Z on Blender), which indicates the direction of the bus, where it should be pointed at

rotZ= << The rotation on Z on the 3D, generally this will be always 0

To know exactly what these coordinates should be, you may use an empty object on Blender, it is one of the easiests ways to get it. For example:



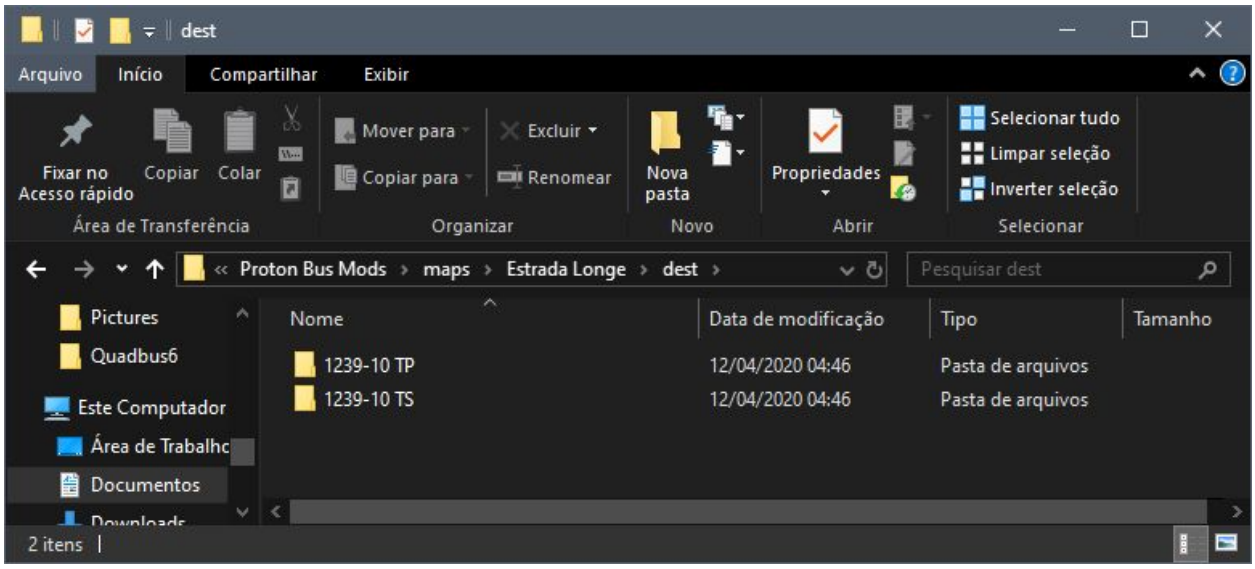
IMPORTANT: ALWAYS USE THE DOT AS THE DECIMAL SEPARATOR, NOT THE COMMA, THAT IS USED IN SOME LANGUAGES (INCLUDING OUR PORTUGUESE). THE GAME WILL TRY TO READ THE DOT. IF YOU USE THE COMMA IT MAY BREAK THE LOADING PROCESS.

The rotation on the vertical axis (Z on Blender, Y inside the game) is very important to define the orientation, where the bus will be “looking at”: straight, 90 degree, -90, 180, 15, 32 etc.

TIP: these position coordinates must be placed slightly above the floor, not inside it, otherwise the bus may fall forever when loaded! It doesn’t need to be too high, but some centimeters of the floor may help. The 0,0,0 point of the bus will coincide with this position when it is loaded or repositioned at the map.

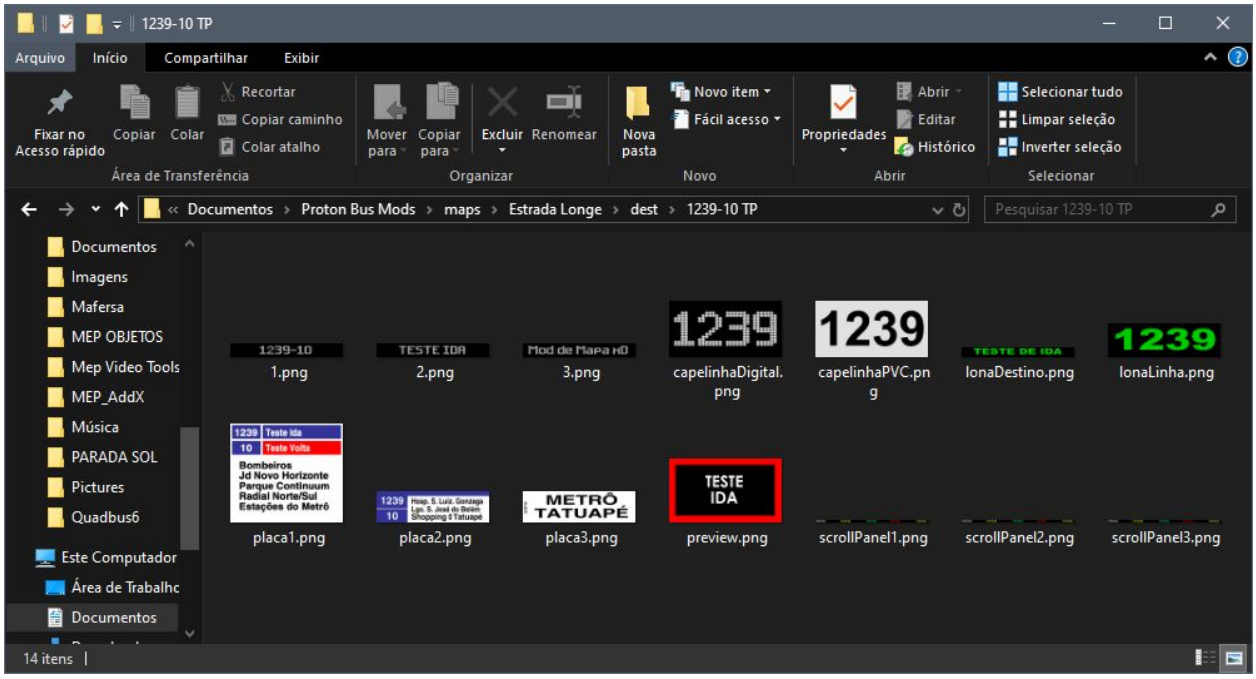
Inside the base directory of the map there must be a folder named **dest**, and inside it there will be more folders with all entrypoints names, exactly as defined inside the **entrypoints_list.txt** file and on the **name** property of **entrypoints.txt**.

IMPORTANT: It is mandatory keeping a good consistency with those names! If there is a different character or space, it may be broken.



That's why it is not allowed using the slash on entrypoints names: they are used on the file system, and Windows and other operating systems do not allow to use this special char because it is used as a directory separator. When they find a forward or backslash they treat it as a folder division, not as if it was a part of the filename itself.

Inside the desired destination folder there will be files with all specific pictures for destinations and plates, together with a picture named **preview.png** that will be used on the route selection screen on the game.



IMPORTANT: AVOID PICTURES LARGER THAN 2048 PIXELS OF WIDTH OR HEIGHT FOR MOBILE MODS! MOST MOBILE GPUS CANNOT HANDLE WELL THOSE TEXTURES. IT MAY MAKE THE GAME BECOME UNSTABLE OR CRASH VERY OFTEN.

All names and suggested sizes are the same for destination displays mods at the user level, as these ones:

1.png Main frontal electronic display, message 1 (1024x128)

2.png Main frontal electronic display, message 2 (1024x128)

3.png Main frontal electronic display, message 3 (1024x128)

capelinhaDigital.png Electronic display at back or side with the route number (256x128)

capelinhaPVC.png Plastic or paper display with the route number (256x128)

lonaDestino.png Old fabric destination display with the route name (256x128)

lonaLinha.png	Old fabric destination display with the route number (1024x128)
placa1.png	Side plate, generally next to the door (400x350)
placa2.png	Frontal plate 1 (512x142)
placa3.png	Frontal plate 2 (512x142)
scrollPanel1.png	Main scrolling destination display, used on some buses (2048x32)
scrollPanel2.png	Second scrolling display, optional (2048x32)
scrollPanel3.png	Third scrolling display, optional (2048x32)

Sizes are only suggestions to keep the proportion, but they may be larger or smaller. Just really avoid anything larger than 2048.

Besides those, there must be a **preview.png** too, that will be shown at the route screen selection dialog. We recommend using the 16:9 proportion for this preview picture. A great size is 640x360. It does not need to be too big.

Route definition: urban or intercity?

New from v253: if the route is an intercity route where people should get off the bus only at the last stop, place an empty txt file in the destination directory named as **intercity.txt**. All routes that have this file will make people to get off the bus only when they pass on the deboarding trigger, ideally at the latest stop. Without this txt the route will be treated as a normal urban route, where people ask to get off after some minutes that they have boarded (the amount of time is randomly selected for each person).

Destination: is it out of service?

New from v253: if this destination should not allow passengers to enter the bus, place an empty txt file in the destination directory named as **outofservice.txt**. This will prevent people of getting on the bus. It is designed for destinations like OUT OF SERVICE, DEPOT etc.

IMPORTANT: FILE NAMES ARE CASE SENSITIVE! IT IS NECESSARY KEEPING THE CASE DIFFERENCES BETWEEN THE LETTERS, SO THEY CAN WORK FOR ALL SYSTEMS. IF YOU IGNORE THIS IT MAY WORK ON YOUR SYSTEM BUT NOT ON OTHERS.

That’s all about entrypoints for now. In the future they will have an additional setting for the GPS or route helpers, but this is not made for this version of the system.

Setting up Passengers

Currently in Proton people will enter the bus at any bus stop where they are, no matter what the destination display shows. In the future this behaviour may be changed to get more realistic, for instance, avoiding entrance when the destination shows another route.

Bus stops settings are divided into two parts: some objects on the 3D model and some params inside a txt file. Objects will define the place where people will be waiting for the bus and the bus stop location itself, while the txt file will hold some properties like the bus stop name, amount of passengers, their rotation (where they will be looking at), and so on.

IMPORTANT: EACH BUS STOP ON PROTON MUST HAVE AN UNIQUE NAME, EXCLUSIVE, THAT IS NOT USED BY ANY OTHER 3D MODEL ON THE MAP! AT THE LOADING PROCESS THE GAME WILL SEARCH FOR ALL OBJECT NAMES. REUSING SOME COMMON NAMES ON DIFFERENT PARTS MAY CAUSE PROBLEMS, SINCE THE GAME MAY SELECT THE WRONG OBJECT. TO AVOID THIS BAD BEHAVIOUR IT IS RECOMMENDED CREATING YOUR OWN NAMING SCHEME, ALWAYS USING A SPECIFIC PREFIX FOR BUS STOPS THAT WILL HELP TO AVOID CONFUSION LATER. THE SAME IS VALID FOR THE AI TRAFFIC SYSTEM, THAT WILL BE EXPLAINED SOON. DO NOT USE ACCENTS OR SPECIAL CHARS ON THESE NAMES TOO, THE SAME RULES FOR FILES APPLY!

It’s recommended that the internal name uses a unique word, without spaces, with some prefix that identifies that it is a bus stop, followed by an exclusive name for this bus stop. For example, some suggestions for bus stop names:

zzStop1, zzStop2, zzSquareStop, zz5AvenueStop1, zzCentralParkDoor3...

Samples of bad names, that could cause problems with other objects on the map:

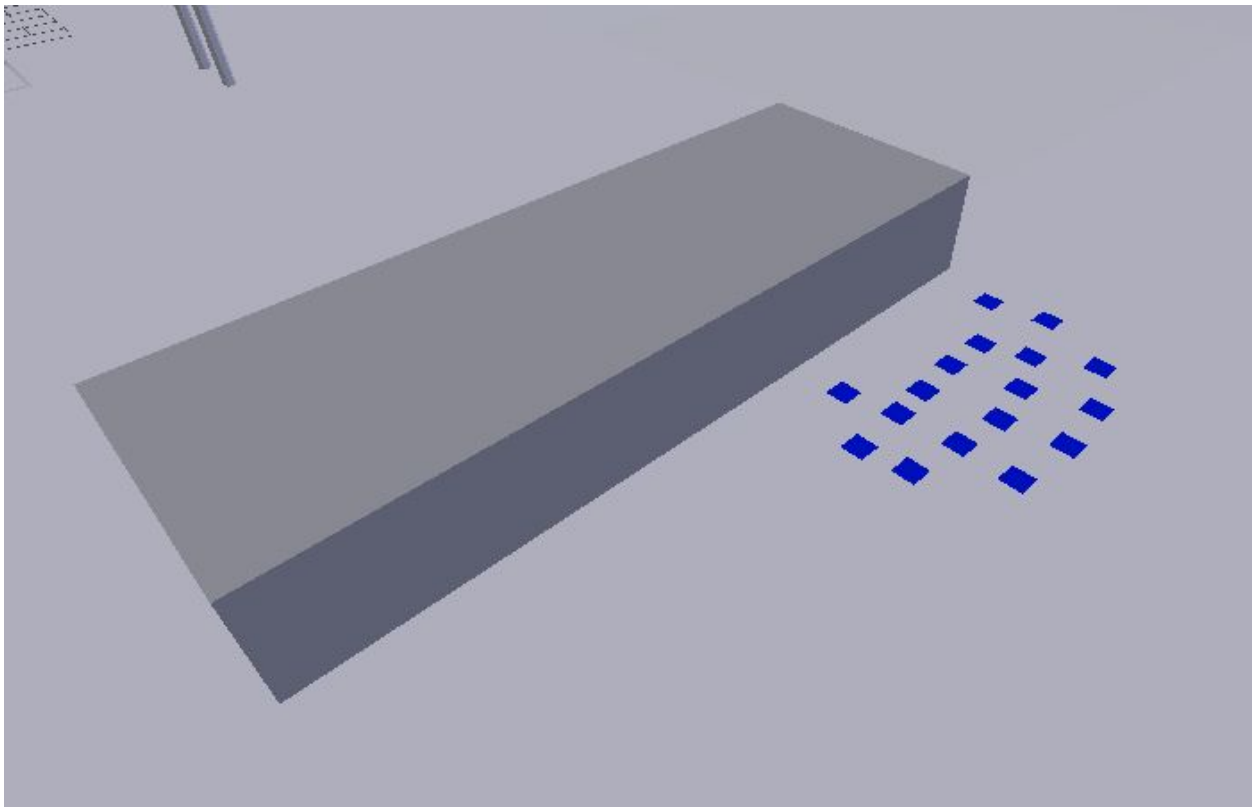
road, street, cube, Cube, collider, stop, bus…

It is not mandatory using “zz”, it’s only a suggestion. It makes it easier when looking for bus stops on Blender later, in case you want to edit or remove some of them. Names without any prefix may make it harder to find them in large maps.

On the 3D model the bus stop will have two kinds of objects: a squared box where the bus must be with at least a little part overlapped, to identify that it is on this bus stop; and some smaller objects on the ground that will be used to mark the positions of the people that are waiting for the bus.

A LITTLE NOTICE: CURRENT VERSIONS DO NOT SUPPORT SITTING WAITING PASSENGERS, ONLY STANDING!

Bus stop settings on the 3D model



The cube, or better, parallelepiped, must be created from a cube, without too much unnecessary vertices. It will be a trigger inside the game, an invisible collider that will know whether the bus is inside it or not. All little dots will be objects to store positions of waiting passengers.

TIP: TO KEEP THE MAP MORE LIGHTWEIGHT AND FASTER TO LOAD, USE SIMPLE SMALLER PLANES FOR GETTING POSITIONS INSTEAD OF COMPLETE CUBES. PLANES HAVE ONLY 2 TRIANGLES, WHILE CUBES HAVE TWELVE (2 TRIANGLES PER FACE, 6 FACES). THOSE OBJECTS WILL BE DISCARDED AFTER LOADED, SO THEY DO NOT HAVE TO BE DETAILED! THEY WILL BE USEFUL ONLY TO GET POSITIONS. THIS DOES NOT WORK WITH EMPTY OBJECTS ON BLENDER, IT MUST HAVE A MESH, SO IT'S BETTER MAKING THE SIMPLEST MESH POSSIBLE.

The collider name must be **busstopname_trigger**. Positions for passengers must have the bus stop name followed by a dot and incremental numbers starting from 000. This is a convenience, since we utilize Blender. It is enough duplicating the first object named with the .000 ending to Blender create the next names like .001, .002, .003 etc. This will be used also for paths for the AI traffic.

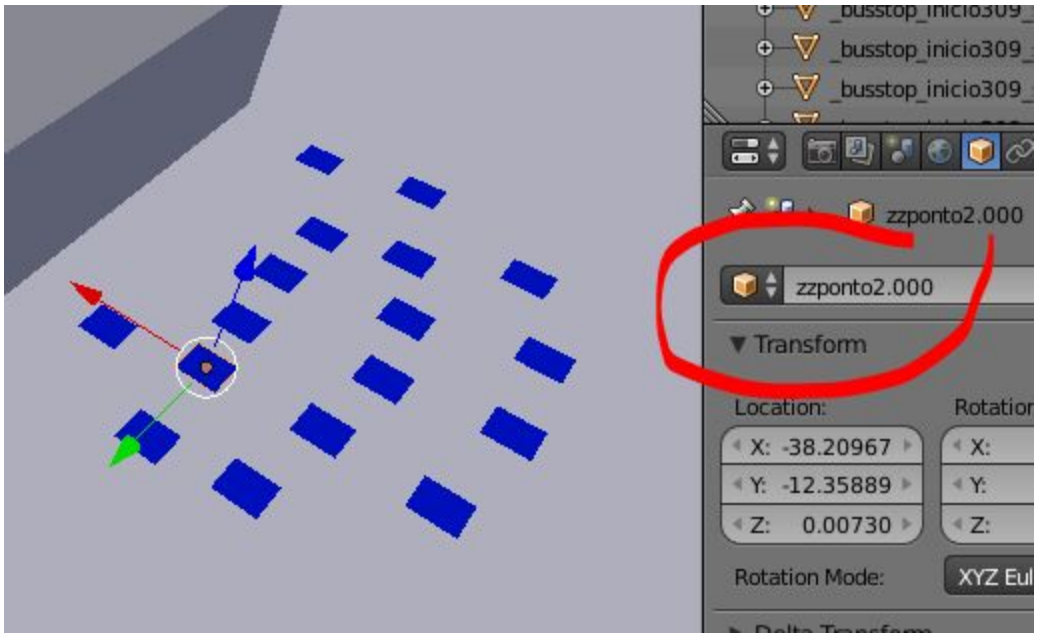
For example:

For the bus stop with an internal name zzStop1:

zzStop1_trigger, zzStop1.000, zzStop1.001, zzStop1.002, zzStop1.003 etc

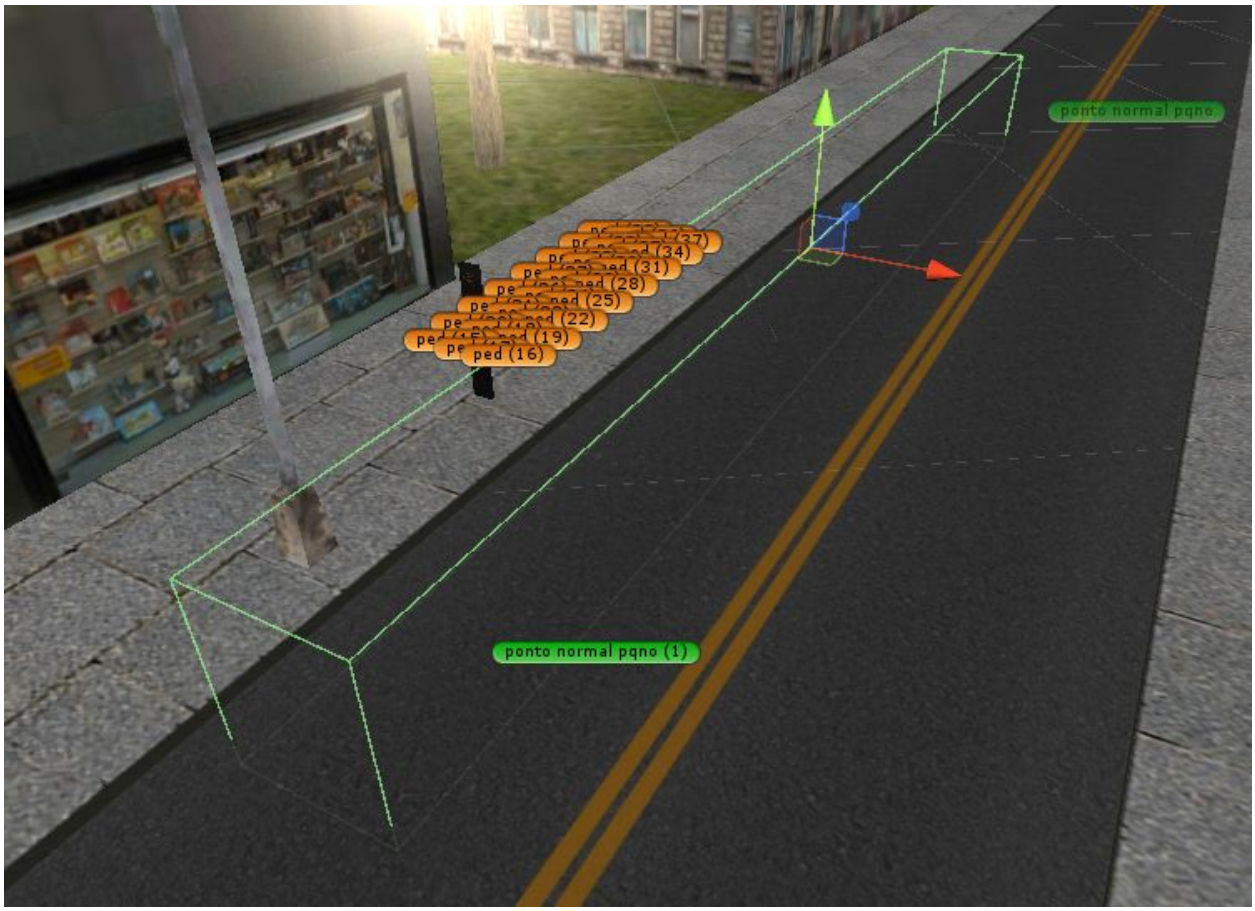
For some bus stop named as zzCentralParkDoor3:

zzCentralParkDoor3_trigger, zzCentralParkDoor3.000, zzCentralParkDoor3.001, zzCentralParkDoor3.002 etc

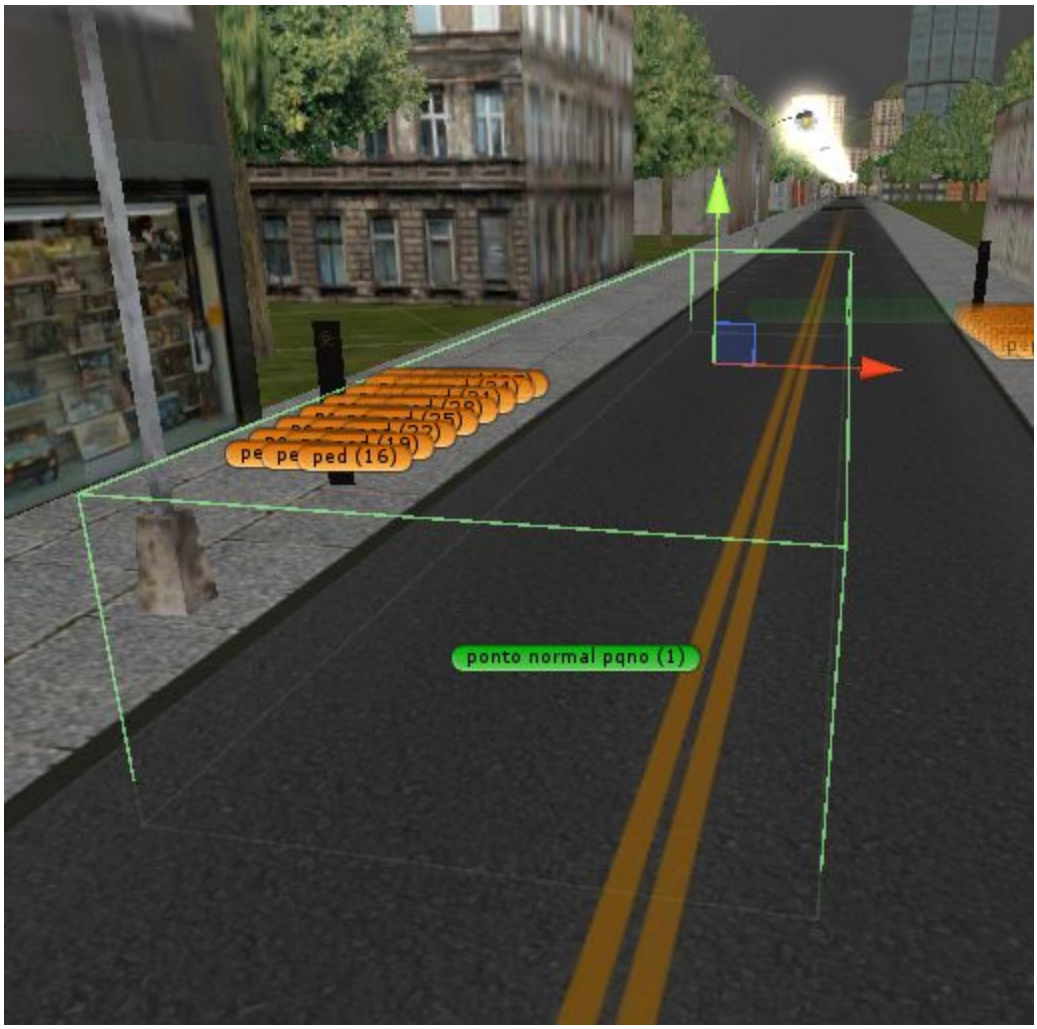


IMPORTANT: THE BUS STOP COLLIDER MUST GET A GREAT AREA INSIDE ITS PLACE, BUT IT CANNOT LEAK TO THE OTHER SIDE OF THE STREET, IF THERE WILL BE TRAFFIC IN THE OPPOSITE DIRECTION COMING. IT IS GREAT PLACING IT WITH A OUTSIDE PART ON THE SAME DIRECTION OF THE TRAFFIC FLOW, THUS ALLOWING THAT THE PASSENGERS COMPLAIN IF THE DRIVER DON'T STOP WHEN THEY WANT TO GET OFF THE BUS. THE BUS MUST BE WITH SOME PART INSIDE THE BUS STOP TRIGGER FOR THE SYSTEM TO WORK.

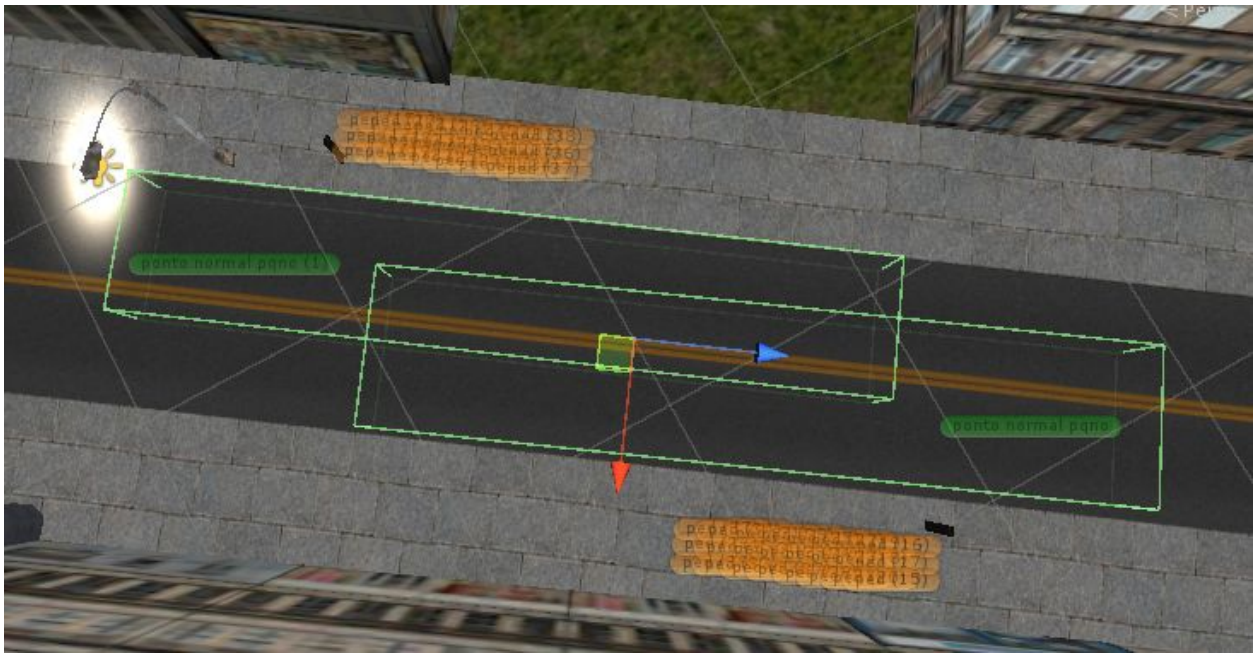
A sample of a perfect, well positioned trigger:



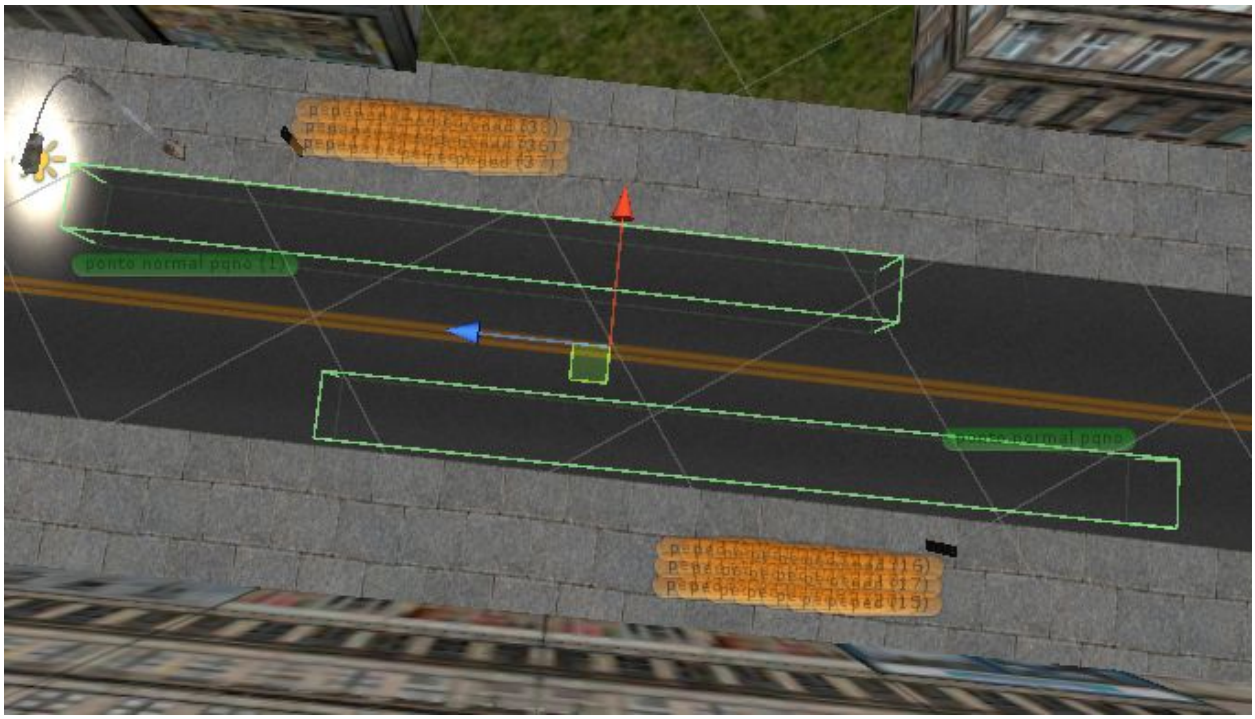
Now a bad one, it cannot leak to the other side of the road:



Sample of bad triggers, with overlapped bus stops, this cannot be used:



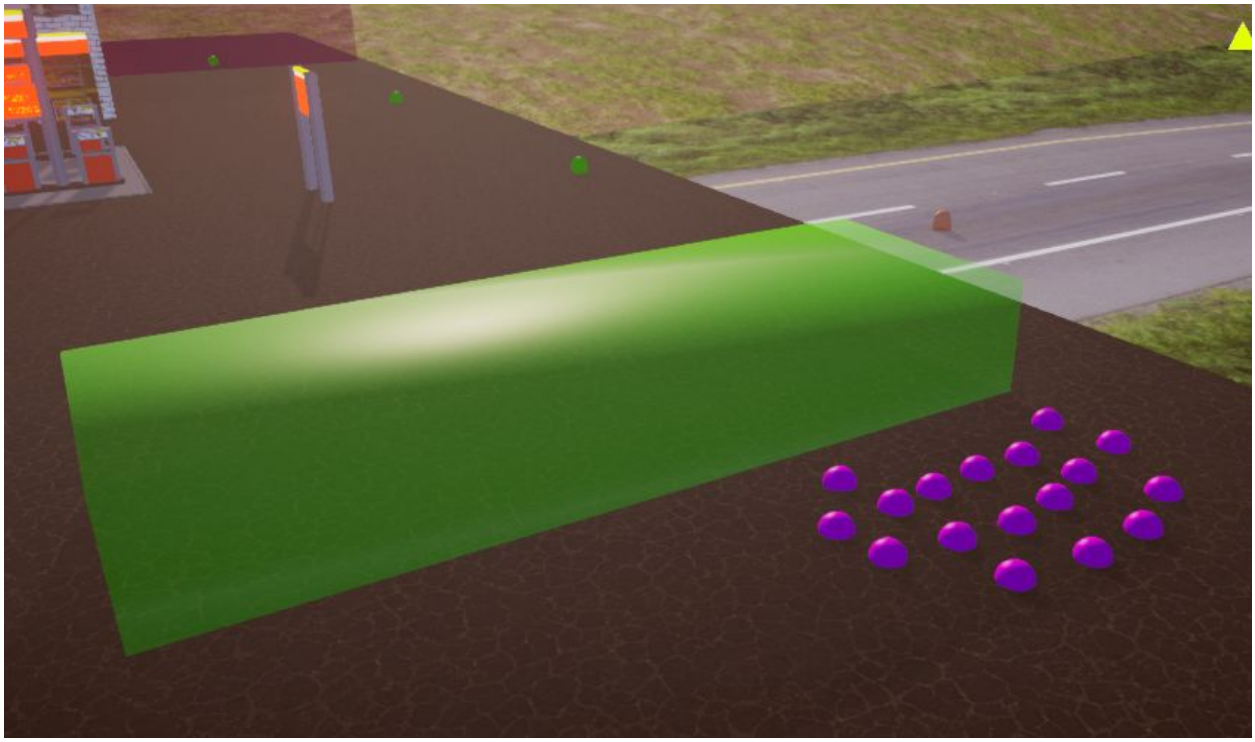
Sample of two great bus stops, non overlapping:



Note that the bus cannot be at two bus stop triggers at the same time, otherwise some bad things could happen, like the people coming from the other side of the road trying to enter the bus.

TIP: ENABLE THE DEBUG OPTIONS TO VERIFY IN GAME WHERE BUS STOPS, TRIGGERS AND OTHER POSITION HELPERS ARE. THIS ALLOWS TO FIND SOME ERRORS ON YOUR MAP THAT MAY BE HARDER TO FIND BY USING BLENDER ONLY.

A sample of the view in debug mode:



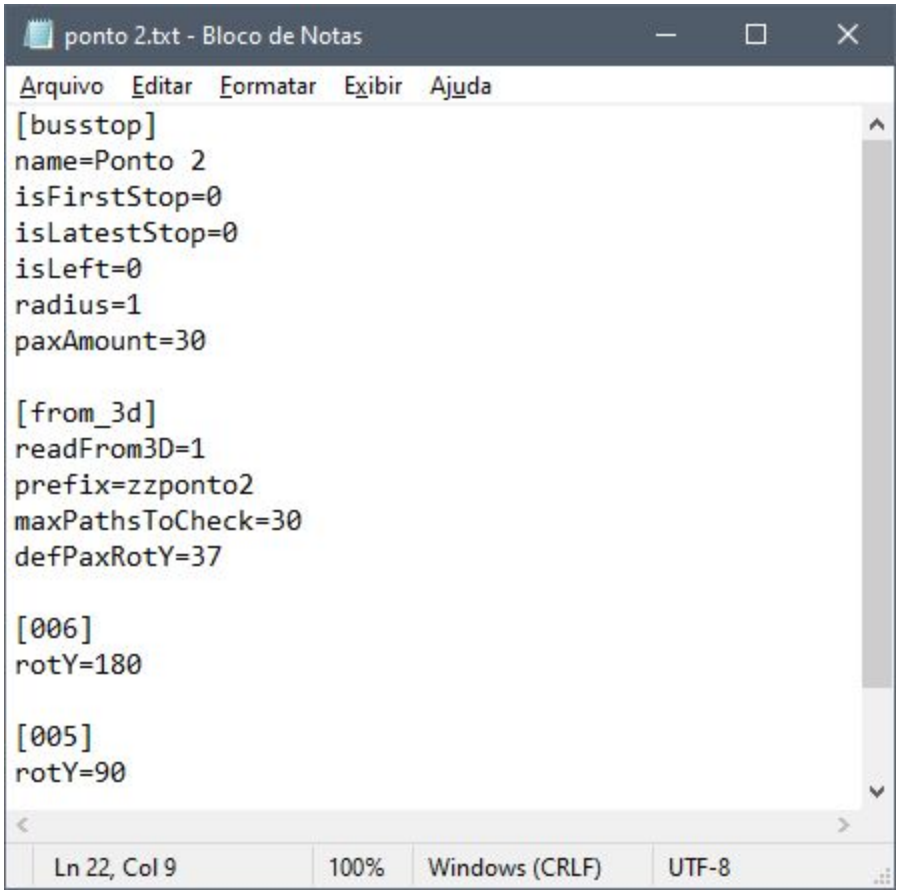
TIP: THOSE SPECIAL TRIGGER AND POSITION OBJECTS DO NOT MUST TO HAVE A MATERIAL NOR TEXTURE. SINCE THEY WILL BE DESTROYED ON LOADING AFTER THEIR POSITIONS ARE GET, IT IS BETTER LEAVING THEM WITHOUT A MATERIAL TO MAKE THE LOADING PROCESS FASTER.

Bus stop settings on txt files

For each bus stop there must be a txt file inside the **busstops** folder, that is inside the **modelsDir** defined on the main txt file. If you place the bus stop on the 3D model and forget the txt, or if there is some inconsistency between used names, it may make the bus stop to not work inside the game or appears as a pink object, like one without material.

A great practise is naming txt file with the same name of the bus stop, that name used on trigger and for positions. This is not mandatory on current versions, but it is great for organization purposes.

The txt content is like this:



The first **[busstop]** section defines some basic properties of the bus stop.

The **name** item defines a name that may appear for the user in the future. Generally it could be a human readable name, with spaces, something that may be used in a future list of routes. Currently it is not being used, but you are free to set it up. It's recommended the same basic rules: no special characters or accents.

Both properties **isFirstStop** and **isLatestStop** will probably be removed in future versions, or replaced with something better. You may use the 0 value for false/negative and 1 for true/positive. Those are boolean values.

isFirstStop would define that this is the first stop of the route, telling the system to not let people get off the bus if the driver stays parking there for a while (like when player is talking in a gameplay video, for example). And **isLatestStop** is for the last one, forcing all passengers to get off the bus. By the way the deboarding is managed with a new trigger, completely independent of the bus stop. So this setting is obsolete.

NOTICE: IN FUTURE VERSIONS THE FIRST AND LAST BUS STOPS WILL BE IDENTIFIED BY SOME LIST WITH ALL BUS STOPS FOR EACH ROUTE, IGNORING THESE SETTINGS. FOR NOW THEY COULD BE USED NORMALLY, BUT THEY ARE NOT VERY IMPORTANT. ON INTERMEDIATE BUS STOPS, THAT ARE NEITHER THE FIRST NOR THE LAST ONE, LEAVE BOTH WITH THE VALUE OF ZERO. YOU COULD ALSO OMIT THOSE SETTINGS THAT THEY WILL ASSUME THEIR DEFAULT VALUES (ZERO).

isLeft is for a left bus stop, no matter if it is at some elevated platform or at the sidewalk level. The default value is 0 (false), showing that the default is a right bus stop. Use **isLeft=1** for left bus stops, like on some bus corridors and BRTs. It is possible using both right and left bus stops on the same route. When entering the bus stop trigger passengers will try to switch to the other side of the bus.

The **radius** property is not used anymore, it was used in internal early versions. Just ignore it or leave it always with the default value (1).

Concluding this section, **paxAmount** defines the amount of passengers that will be waiting for the bus on this specific bus stop. This value will not be used literally: the game may apply a random value based on this one depending on the user settings, that percentage of passengers at the game options. To use a higher number it is necessary having empty “slots” on the bus stop (that smaller planes .000, .001 etc on Blender), otherwise some people may appear over the others or it may occur an unexpected bug.

TIP: FOR SOME VERY IMPORTANT PLACES WITH LOTS OF PEOPLE, LIKE BUS TERMINALS, SHOPPING CENTERS, PARKS ETC, USE MORE PEOPLE! BUT SET IT TO A MUCH LOWER VALUE FOR SOME ROADS TO MAKE IT MORE REALISTIC.

The **[from_3d]** section defines some settings that will be used together with some 3D models defined inside Blender.

readFrom3D=1 must be always 1. In older prototypes this value was 0 because the 3D coordinates would be manually defined inside the txt, but reading them from the 3D model is way better for the map creator. We will not have to copy and paste hundreds or thousands of numbers, only rotations.

prefix=nameofthestop must contain the internal name, that one that is used on the 3D: the same name before _trigger or before .000, .001 etc.

IMPORTANT: THE VALUE OF THE PREFIX ITEM MUST COINCIDE EXACTLY WITH THE NAME USED ON THE 3D MODEL! IF IT IS DIFFERENT, THE BUS STOP MAY BE SKIPPED OR SOMETHING ELSE UNEXPECTED COULD HAPPEN. TO AVOID PROBLEMS IN THIS SETTING, ALL BUS STOP NAMES MUST NOT CONTAIN SPACES, SPECIAL CHARACTERS, ACCENTS ETC. USE THE FRIENDLY NAME ON THE “NAME” PROPERTIE TO GIVE THE BUS STOP A NICE NAME, LEAVING THE INTERNAL NAME FOR THE SYSTEM.

maxPathsToCheck=30 defines the amount of positions to verify for this bus stop. If your bus stop will have more than 30 passengers positions, increase this value here to avoid that the exceeding people could be ignored. This was made for optimization purposes. The game cannot guess how many positions it should reserve, so it must read all, starting from 0 and adding 1…

Until it cannot found the next. This item allows that the counting process stops at the defined number, avoiding spending unnecessary processing for most bus stops that will have less passengers.

defPaxRotY=37 defines the default rotation for passengers. It is about the vertical rotation, indicating where they will be pointing at. Generally on Blender this will be the Z rotation, please remember that we use Unity and on Unity it is on the Y axis. This sets the direction of where people will be looking at, so the value will change between each road. We recommend that you try values like 0, 90, -90 or 180, and make the necessary adjustments based on what you see.

At the end we have some optional sections, that will be basically defined by the number of the path between brackets. These will have only the **rotY** param, allowing you to define an individual rotation for each person. For example:

[006]

rotY=180

[005]

rotY=90

This indicates that the passenger on the position identified by 006 will be rotated at 180 degree, while the 005 will be at 90. All other people on this bus stop without its position manually defined here will use the **defPaxRotY** defined inside the [from_3d] section. Generally you don't have to worry about this. Most of the time the defPaxRotY is enough. You may delete or do not even have to place these specific individual numbers.

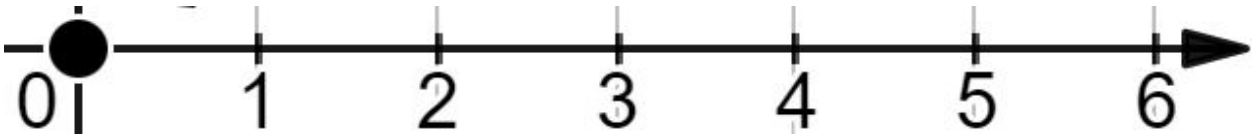
TIP: DELETE THOSE INDIVIDUAL ROTATIONS IF YOU COPY THE TXT FROM THE SAMPLE MAP! THOSE ITEMS BETWEEN BRACKETS MUST BE CONFIGURED FOR EACH CASE. IF YOU WILL NOT USE THEM, THEY SHOULD BE DELETED. THE SAME WILL BE VALID FOR PATHS FOR CARS, BUSES AND TRAINS LATER. THEY ARE ONLY EXAMPLES FOR NOW.

Setting up pedestrians

Let's go with the theory frist. It is essential to understand how the system works. All marks of the areas where people walk or cars and trains drive have a very similar setting. There will be some different params among them, but the general idea is the same for both.

They move in a predefined way, a linear way like a path defined by many points. Basically all points or dots are linked in sequence. Paths always work with a direction: from the previous to the next one.

It would be like a positive segment of the Cartesian plane:



But in this case it is allowed to make curves, get climbs etc. But always respecting the sequence order. By knowing this you already know how you should make the paths inside the game: by positioning objects with some specific names and keeping it as a sequence!

This is a manual process that take a long time to make. It is not really difficult, just time consuming. We dream that in the future we will have a better tool to generate those paths, we know that currently it is not great. The same difficulty we've got in the past with our native maps. But do not be discouraged, it is awesome watching the cars and people moving! Generally the result of our dedication while creating the map is worth it!

Some points may also be a "spawner", that makes it "born" an object there that starts to move to the next point. When getting to the end of the path they have two choices: disappear… Or assume the next path ahead, if they find one that begins exactly where the previous one finishes.

For optimization purposes the game will only show moving objects (people, vehicles or trains) that are close to the player within a certain radius of distance. It would be a huge waste of

processing resources making the engine to render a walking object that is 5km away. On the other hand, it is too bad spawning objects right ahead of the player's vision. They must appear naturally, coming from some place. So generally they do not appear right in front of the player.

There is a great ramdominess level on spawners so they do not spawn all at the same time or too overlapped. You may control the frequency of spawning to create more or less dense areas. It is possible also to reduce the amount of spawners, allowing that it appears less objects over the time. This will be useful for trains, since it will normally pass one after some minutes that the last one came, not all together like pedestrians in huge cities.

NOTE: PEDESTRIANS AND TRAINS ON PROTON DO NOT IDENTIFY COLLISION WITH OBJECTS AHEAD BY DESIGN, DIFFERENTLY FROM VEHICLES LIKE CARS AND BUSES. THIS IS MADE FOR OPTIMIZATION PURPOSES, ALLOWING US TO PLAY WITH LOTS OF PEDESTRIANS. THE CURRENT SYSTEM IS NOT PROJECTED TO ALLOW CROSSINGS ON THE ROAD, OTHERWISE IT COULD PLACE MANY PEDESTRIANS, MAKING IT HARDER TO PASS WITH THE BUS (AS SEEN ON METRÔ CARRÃOZINHO ON THE ARICANDUVA MAP). PLEASE TRY TO AVOID THESE PEDESTRIAN CROSSINGS.

Basically each circuit or path has its own settings, independent of the others. Each one must have therefore an unique name. It is valid the same rule for bus stops: avoid using common names that may be confused with other map objects. It is great having an own prefix for each kind of path, followed by a small name.

Pedestrian paths are set with 3D objects on the model to get their positions, starting with .000. For example:

xxSidewalk1.000, xxSidewalk1.001, xxSidewalk1.002 etc

anotherRoad.000, anotherRoad.001, anotherRoad.002 etc

Please try to avoid using names that may exist as a part of other objects, like cube, door, street etc. Some exclusive prefix at the beginning of the name may help it. We do not want to force using a specific prefix, so it is up to you managing it on your maps. If the name of some object

coincide with the name of a path it may break the game, since the engine could not know which one is the correct one for the path to continue. For example, if there is a part of a road named as street.003 as a result of duplicating an object… And a path was set as street.000, street.001, street.002… When making the path the game may read the position of the wrong object as if it were part of the path, thus generating an undesired behaviour: the pedestrian, car or train could be moved to a different place and/or the road piece would disappear.

Like the position setters for bus stops, prefer to use simple planes without materials whenever possible, to keep it more performant. But if the map is small, there is no problem by using cubes. If some path need to pass through another one, it may be great creating a colored material without texture for each one, just to separate them visually inside Blender. All the models and materials of those paths will be destroyed when loading the map, so the more lightweight they are, the better it is for the overall experience.

TIP: THE INTERNAL TECHNICAL NAME FOR THESE POINTS IS “WAYPOINT”. THEY ARE LITERALLY A “WAY POINT”.

About spawners, an important thing: leaving all waypoints as spawners is great to develop faster and try faster too, or to fill dense areas like important squares, places, bus terminals etc. But this may have a negative impact on performance, including both memory as well as processing power.

Each spawner must make many calculations at each interval (a custom time that may be defined for each one). Those calculations are not heavy, but it may get complicated if there are thousands of them at the same time. In very large maps or with many long roads with many traffic lanes, if all paths were spawners, performance could become gradatively worse. Due to internal engine limitations, all of those calculations must run on the main thread of the application, disputing processing of one core only of the processor. The more simultaneous operations it must do, the more heavy it gets, taking it longer to render the next frame.

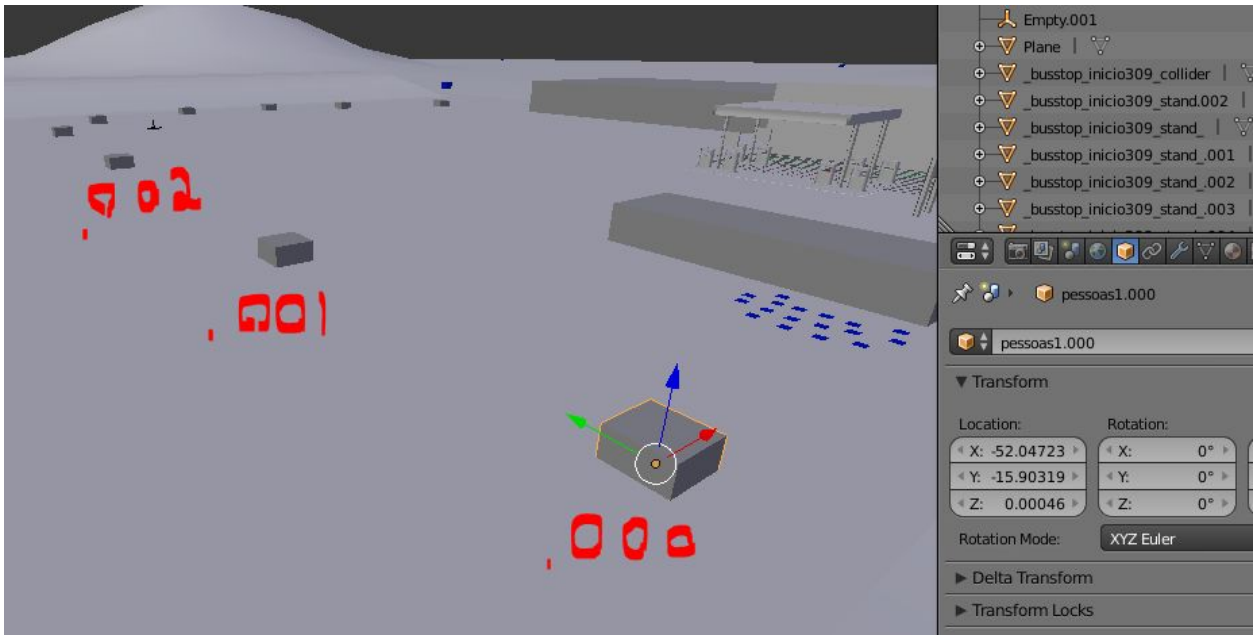
Generally you should not worry for the most situations, but be advised. If the map is huge with many kilometers or many roads, this may become a serious problem. At our experience with Proton we could see that for most common situations it won’t be a real problem. The old map

with the bus corridor of the route 4520 on the Aricanduva city was made entirely in one scene, like mods, it have got three lanes in each direction… And performance were not affected due to the traffic system there. But if it were a map with ten times that size, things could get worse. So stay tuned: generally the less spawners, the better.

At curves it is necessary defining more points so the objects may make a smooth movement. At straight areas you may give a higher space between each point. The system that makes the vehicle movements will receive lots of improvements over the lifetime of the project, but paths probably won't change.

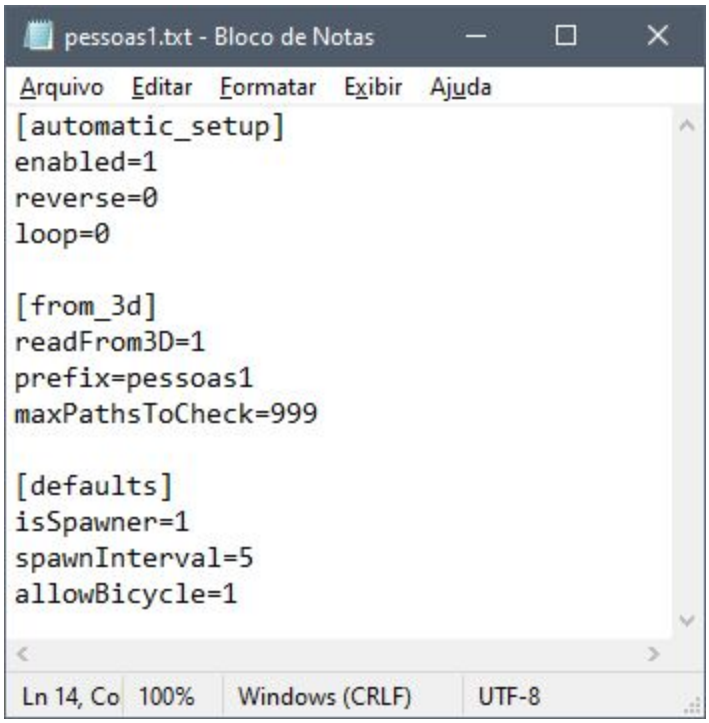
Now, let's make it!

Basically there is no secret when placing objects on Blender: create a plane or a cube, reduce its scale so it won't disturb viewing other objects, and name it with a unique name followed by .000. For example, xxSidewalk1.000 (let's keep the xx to avoid mixing it with the zz from bus stops, but it's only a suggestion or example). Later, duplicate this object (CTRL + D on Blender) and position it a little bit ahead. And keep doing it… It will be filling up the correct numbers for us. So that's why we use the naming convention finishing in .000, it helps a lot! At least in Blender 2.79, we hope that it will always continue doing it.



After doing it, let's go to the txt! Create a new txt file inside the **aipeople** folder, that must be inside the models directory (the **modelsDir** that is defined on the main txt of the map). The name for this txt is not important, but it is good having the same path name to keep it organized, this will help if you have plans to make more changes later.

This file has the following structure:



The first **[automatic_setup]** section has these items:

enabled=1 is very important, leave it always as 1. The system will make an automated setting linking the points together, up until the last one. This helps a lot the setting up process, otherwise we would be manually linking each position.

The **reverse=0** param is optional. Leaving it as 1 will make the path to bet set at the opposite direction, from the last to the first points.

NOTICE: IT SEEMS THAT THERE IS A BUG AND THIS SETTING MAY BE IGNORED. WE DO NOT RECOMMEND SETTING REVERSE TO 1 RIGHT NOW, BY THE WAY IT WAS CREATED FOR THIS PURPOSE.

Concluding this section, **loop** defines if the path will be looped. Leaving it at 0 (false, the default value) it means that not: when the path ends, it really ends. The person (or car, train etc) will

disappear when getting to the last waypoint. Leaving it as loop=1 the last waypoint will be linked to the first one. This is very useful for bus terminals and public squares, where people could keep continue walking. But it is not recommended for most normal linear roads and sidewalks. If the loop has the value of 1, people will walk on the entire path and repeat it, without disappearing when it gets to the finish.

WARNING: USING LOOPS FOR SMALLER PATHS IS NOT RECOMMENDED! SPAWNERS WILL BE ALIVE, SO THEY MAY SPAWN A LOT OF PEOPLE... SINCE THEY WON'T DISAPPEAR AT THE END OF THE PATH... BECAUSE THEY WILL BE CLOSE TO THE PLAYER... LOOP IS GREAT FOR LARGE AREAS, WHERE THE DISTANCE FROM THE PLAYER MAKES PEOPLE DISAPPEAR, FOR EXAMPLE, AT THE CARRÃOZINHO BUS TERMINAL ON THE ARICANDUVA MAP. THIS IS NOT RECOMMEND FOR SMALLER BUS TERMINALS.

The **[from_3d]** section is very similar to the bus stops:

readFrom3D=1 will always be 1, to read the paths from the 3D file.

prefix defines the name of the path objects, without the dot and the numbers. For example, xxSidewalk1, xx69Avenue1, etc. The name of the txt file is not important, but this prefix must match exactly the name defined on the 3D objects.

maxPathsToCheck=999 is very similar to the bus stop too, but here it is with a higher limit by default. Generally you do not have to change it, unless your path has more than 999 waypoints. Otherwise the last ones could be ignored.

The waypoints settings has a new and very important section: **[defaults]**. It will define the default values for many settings of the waypoints of this path. This is great so you do not have to set it up individually for each path, since there will be hundreds or thousands of them around the entire map.

isSpawner defines if paths will be spawners or not. If it got the value of 1, yes, paths will be spawners, making people, cars or trains to appear at their positions. If it got the value of 0, so no, they won't be spawners. So each individual path may become an independent spawner later.

Generally for pedestrians and vehicles we use isSpawner=1, but for trains, 0, so they not appear overlapped.

spawnInterval defines the basic time interval in which the spawner code will run, in seconds. Leaving it at 5, at each 5 seconds (with a small random variation) all waypoints of this path will try to spawn a walking object. It is not recommended setting it to a too low value since it could bring performance problems. Increasing it may make that it lasts longer to appear people, allowing to set up less dense areas, like secondary roads or calm villages.

allowBicycle defines whether it could spawn or not cyclists on this path. Leave the value 1 to allow, or 0 to disallow cyclists. Generally paths inside bus terminals would have the 0 value, to avoid bicycles inside their platforms. By the way, if another path makes a link with the terminal and the previous one allows bicycles, cyclists could finish their paths by passing through the bus terminal normally.

NOTICE: GENERALLY CYCLISTS DRIVE ON THE ROAD TOGETHER WITH CARS, BY THE WAY ON PROTON THEY ARE PROGRAMMED TO DRIVE TOGETHER WITH PEDESTRIANS, MAKING THEM TO APPEAR ON SIDEWALKS. THIS IS NOT SO DIFFERENT OF OUR REALITY SOMETIMES… CURRENTLY IT IS NOT POSSIBLE CREATING EXCLUSIVE BIKE LANES.

The **[defaults]** section defines the behaviour for all waypoints of that path. If you would like to make a different part for some segment, for example, by reducing its density, you could change settings for a specific waypoint too. At the end of the txt file, create a new entry with the number of the desired path between brackets, followed by its individual properties. For example, let's assume that the path of number 512 must have less people. So we make it as follows:

[512]
isSpawner=1
spawnInterval=50
allowBicycle=0

Repeat it for other paths that may be useful for you. This will be used very often for trains, where generally isSpawner of the [defaults] section will be set to 0, and only a few waypoints inside the path will have its own param isSpawner=1.

To create more non integrated paths is easy: just repeat all procedures, choosing another name for them!

IMPORTANT: IF THE LAST WAYPOINT OF A PATH IS OVERLAPPED WITH THE FIRST WAYPOINT OF ANOTHER PATH, THE SYSTEM WILL UNDERSTAND AS IF IT THERE IS A CONNECTION BETWEEN THEM! THIS IS USEFUL TO JOIN TOGETHER DIFFERENT AREAS WITHOUT HAVING TO CREATE PATHS WITH LOTS OF WAYPOINTS.

VERY IMPORTANT: DO NOT PLACE WAYPOINTS TOO CLOSE! IF THEY ARE TOO CLOSE TO EACH OTHER, MOVING OBJECTS MAY STOP OR CANNOT TURN WELL. ON CURVES OR CORNERS YOU MUST USE MORE POINTS TO MAKE IT SMOOTH, BUT DO NOT EXAGGERATE. TOO MANY CLOSE WAYPOINTS ON THE SAME PATH MAY BREAK THE SYSTEM.

Setting up the traffic system for vehicles

After learning how to place pedestrians, it is much easier now to understand how car paths are made! It's basically the same thing, with just some new properties. The folder to store all txt related to vehicles is named **aivehicles**, that also must be inside the models directory.

IMPORTANT: PLEASE REMEMBER TO USE UNIQUE NAMES! DO NOT MIX PEOPLE WITH CAR OR TRAINS WHEN NAMING, TO AVOID CONFUSION OR OBJECTS DRIVING TO THE WRONG PLACE. OUR DEVELOPMENT TEAM DO NOT HAVE ENOUGH TIME TO FIX PROBLEMS ON YOUR MAPS CAUSED DUE TO MISCONFIGURATION. PLEASE, BE PATIENT WHEN ORGANIZING YOUR STRUCTURE. BEFORE RELATING SOME POTENTIAL BUG, DOUBLE CHECK IF IT IS REALLY A GAME BUG OR A MAP BUG.

For vehicles, both **[automatic_setup]** and **[from_3d]** will be the exact same thing as they were for pedestrians. Most needed changes are made on individual properties. They may be set up at **[defaults]** to be applied for all waypoints of the path, as well as for some specific path with **[number]**.

isSpawner is the same thing for pedestrians: 0 for a normal point without spawner and 1 to spawn vehicles there.

isBusSpawner is a different one: it's a secondary spawner just for buses. If you would like to place lots of buses, you may leave isBusSpawner=1 on [defaults] so all waypoints could spawn a bus. Generally this is not recommended since it is not realistic. We recommend that you set isBusSpawner=0 to not spawn buses by default, and later select just a few individual waypoints and set them as bus spawners. So there will be more cars than buses on the road.

TIP: FOR EXCLUSIVE BUS CORRIDORS OR LANES , LEAVE isSpawner=0 ON DEFAULTS AND isBusSpawner=1, SO THERE WILL BE ONLY BUSES THERE, WITHOUT NORMAL VEHICLES!

rightBlinker defines if the vehicle should enable its right blinker when it has this waypoint as a target. Generally this will always be 0, but it should become 1 on specific paths of corners or turns, setting a new property at the bottom of the file with the number of the waypoint inside brackets.

leftBlinker is very similar, but for the left. Leave also as 0 on defaults and set it to 1 for individual waypoints.

spawnInterval defines the time interval of the spawner, in seconds. Increase this value to give more time between one and another car that could be spawned there, thus allowing to reduce

the traffic density. It is great increasing the spawnInterval for secondary roads, or areas where the bus must enter in a sharp turn, to avoid conflicts with lots of cars on the corner.

applySpeedMultiplier sets whether the default speed of vehicles will be changed or not, based on a multiplier defined just below. By default it is 0, disabled. If you would like to increase or decrease the speed for this path, leave this value at 1 and set the speedMultiplier below.

speedMultiplier defines a factor for the speed on this path. It could be defined right on defaults, where it will be applied to all waypoints of the path, or for individual paths, allowing that cars change speed only for some specific region like a school area or a climb, for example.

Vehicle speeds on Proton is still something to be improved. By the way this multiplier factor allows changing it. Do not use zero, use some value between 0 and 2. Please do not exaggerate too much. If you make them to move very fast, you may encounter some problems like cars driving on top of others or unable to properly brake.

If you set speedMultiplier to 0.5 then the speed limit of that region will be half of the default value, and 2 will make it double the default value. Note that each vehicle has a random internal customization for speeds, so the simulation gets more real: acceleration and max speed is different for each car. You may observe it on some crossings like at the entrance of Metrô Carrãozinho on Aricanduva: when a traffic light is set to green, not all cars behave exactly like every other. This is not definitive or finished, but it is already interesting.

Remember that for the speed multiplier to be applied, **applySpeedMultiplier** must be set to 1.

TIP: SET UP A HIGHER SPEED FOR EXPRESS LANES ON HIGHWAYS, FOR EXAMPLE, AND SLOWER SPEEDS FOR LOCAL LANES!

Setting up AI buses repaints

Currently it is not possible choosing which vehicles should run on the traffic system. This is also true for buses. This feature to mod vehicles is on our plans, but it is not confirmed, given its complexity and potential performance problems.

By the way, you can change repaints for the PBC bus that is used on the AI traffic!

You may place all desired skins inside the folder at **skins/0/pbc**. Skin for visstaLO is not confirmed yet, we consider removing this bus to save more RAM for other map objects.

Just place all desired pictures inside the skins/0/pbc folder that the game will randomly choose one of them when a pbc appear.

TIP: USE ONE REPAINT ONLY, OR JUST A FEW! IF YOU PLACE LOTS OF REPAINTS GAME WILL NEED MORE RAM, CONSIDERING THAT REPAINTS ARE GENERALLY HUGE IMAGES... USERS COULD EASILY CHANGE REPAINTS FOR WHATEVER THEY LIKE DIRECTLY ON THAT FOLDER.

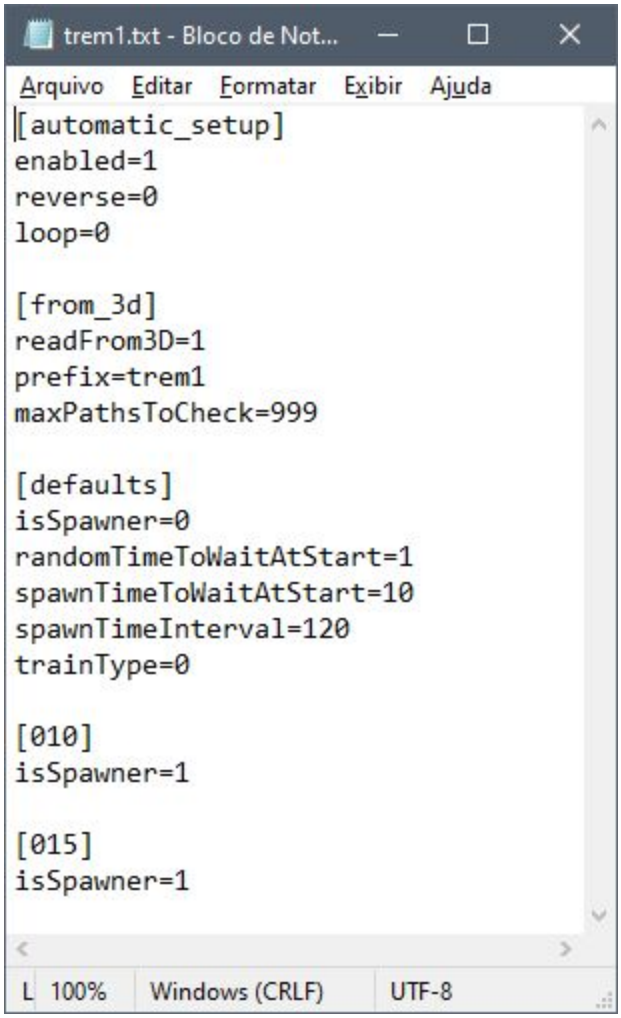
Skins must be preferably PNG images, keeping that observation to avoid accents or special characters for filenames.

The template to create a repaint for pbc is here:

http://omsi.viamep.com/proton/base-skins-pbc-protonbussimulator.zip

Setting up trains

Settings for trains are almost identical to pedestrians and cars, with one big difference: it is great setting isSpawner to 0 at [defaults], and manually placing some spawners at very specific paths, giving a huge space between them. Txts files for trains paths must be inside the **aitrains** folder, that one that is inside the models directory.



New params for trains are as follows:

randomTimeToWaitAtStart must have the value 0 or 1, it's a boolean (true/false). If it is true (1), the system will wait for a random time between 0 and the spawn interval defined later to load the first train once the map is loaded. For example, if the time interval is 120 seconds (two minutes)... If randomTimeToWaitAtStart is equal to 1 then the first train could appear at any moment after the map was loaded, from between some few seconds to one minute, almost two… It will be random inside this interval. Generally this is better to allow that the train does not appear always at the same time when the map is loaded with the player close to a spawn point, giving it a little bit of more realism to the game. Sometimes train will appear faster, others it will take longer to appear. So useful!

spawnTimeToWaitAtStart will be used if randomTimeToWaitAtStart is set to 0. So the first train on this spawner will appear after these seconds passes. For example, if you set it to 20, the first train on this spawner may appear 20 seconds after the map was loaded.

spawnTimeInterval defines the repeating time interval between trains on this path (slightly randomized), after the first one has appeared. We recommend a value like 120 (two minutes) or more, this is a great interval for modern trains or metros. At some places you may increase it to a higher value. Please note that the time is defined in seconds, so multiply or divide it to 60 if you would like to calculate it in minutes.

These settings allow a great randomness level, since player will see the train at different moments in different places. Note that for the first timer to take effect, player must have started the map close to a train spawner. If player spawns too far and gets close to a train spawner by driving, randomness will be even better.

To finish it, **trainType** defines the kind of train that will appear. The default value is 0, we recommend that you leave it at 0 until some future update brings more trains. This value will allow you to customize the type of train like suburban, subway or freight.

IMPORTANT: TRAINS DO NOT HAVE PHYSICS DETECTION LIKE CARS! THEY WILL NOT STOP IF THERE IS A CAR OR PEDESTRIAN ON ITS PATH. SO IT IS NOT SUPPORTED TRAIN LEVEL CROSSINGS.

BE CAREFUL: PLEASE PLACE TRAIN SPAWNERS WITH A GREAT SPACE BETWEEN THEM! LIKE ONE AT EACH ONE OR TWO KILOMETERS OR SO, NOT TOO CLOSE! IF YOU PLACE LOTS OF TRAIN SPAWNERS OR SET isSpawner=1 ON DEFAULTS, A LOT OF TRAINS MAY APPEAR OVERLAPPED. THE SYSTEM WAS DESIGNED TO HAVE SPACED TRAIN SPAWNERS TO WORK WELL.

Special commands and names on 3D models

Transparent parts for grids, trees

Pieces that must have some transparency may be named with the **_transparent_** keyword as a part of the name. This keyword may be before or after the object name, it does not matter the order. For example, plate_transparent_ and _transparent_plate will produce the same effect.

For the transparency to work in fact, the texture used on the object material must have an alpha channel with some transparent value. Generally you may set this by using a graphics program; this is out of the scope of this guide.

Currently this command utilizes a simplified transparent shader to be more performant. It may not emit shadows. In the future it could have some more commands related to transparency.

Parts with colliders for roads, sidewalks, platforms, floors

All parts that must have collision must have the **_gencol_** keyword as part of its names. By having it, the game will try to generate a collider when the object is loaded by using some internal engine function to do it.

For optimization purposes, please, use this command only when really needed! If there are lots of useless colliders or complex models with colliders, the game may have a bad performance drop or be too heavy to render.

We recommend that you use colliders only for floor parts, like roads, bus terminals platforms, columns where player may hit, etc. Avoid using it on normal houses, buildings, detailed walls, etc, it will make the game runs slower.

IMPORTANT: COLLIDER GENERATION IS AN ENGINE FUNCTION THAT WE DO NOT HAVE DIRECT ACCESS TO. FUTURE VERSIONS MAY BREAK COMPATIBILITY WITH CURRENT COLLIDERS. IT MAY BE NECESSARY MAKING A MAP UPDATE WHEN THE ENGINE IS UPDATED. TO MINIMIZE THE RISKS, PLEASE AVOID USING COMPLEX COLLIDERS, WITH LOTS OF POLYGONS. IF NECESSARY, GENERATE A SIMPLE GEOMETRIC FORM AND SET IT AS AN INVISIBLE COLLIDER, AS THE FOLLOWING COMMAND WILL SHOW, LEAVING THE COMPLEX GEOMETRY WITHOUT COLLIDERS.

VERY, VERY IMPORTANT: HUGE COLLIDERS MAY BE HEAVY TO PROCESS OR EVEN PROBLEMATIC, MAKING ALL THE PHYSICS CALCULATIONS TO BECOME UNSTABLE. IF YOUR ROAD OR OBJECT IS TOO LARGE, PLEASE DIVIDE IT INTO SMALLER PIECES WITH A FEW DOZENS METERS (OR A FEW HUNDREDS). DO NOT MAKE TOO HUGE KILOMETRIC OBJECTS WITH COLLIDERS, SINCE THEY MAY BRING PROBLEMS SOON OR LATER.

Invisible parts with colliders for blocking areas, complementary floors

Sometimes you would like to use an invisible collider just to avoid that the player threspasses some unfinished area, but do not want making the place “too ugly” with a visual blocking object. In this scenario you may use the **_invisible_** keyword together with **_gencol_**. It basically will remove the viewing system, the render command of the object. For these objects, materials and textures are not needed, just the mesh. For example, **blockingblablabla_gencol_invisible_**.

This is also great to generate simple colliders for complex models. For example, a detailed light pole. It is a bad idea using the _gencol_ for those objects with lots of details, since they will add a lot of unnecessary geometry to the physics system, increasing the complexity of all calculations. The more polygons on colliders, heavier to render the game will be!

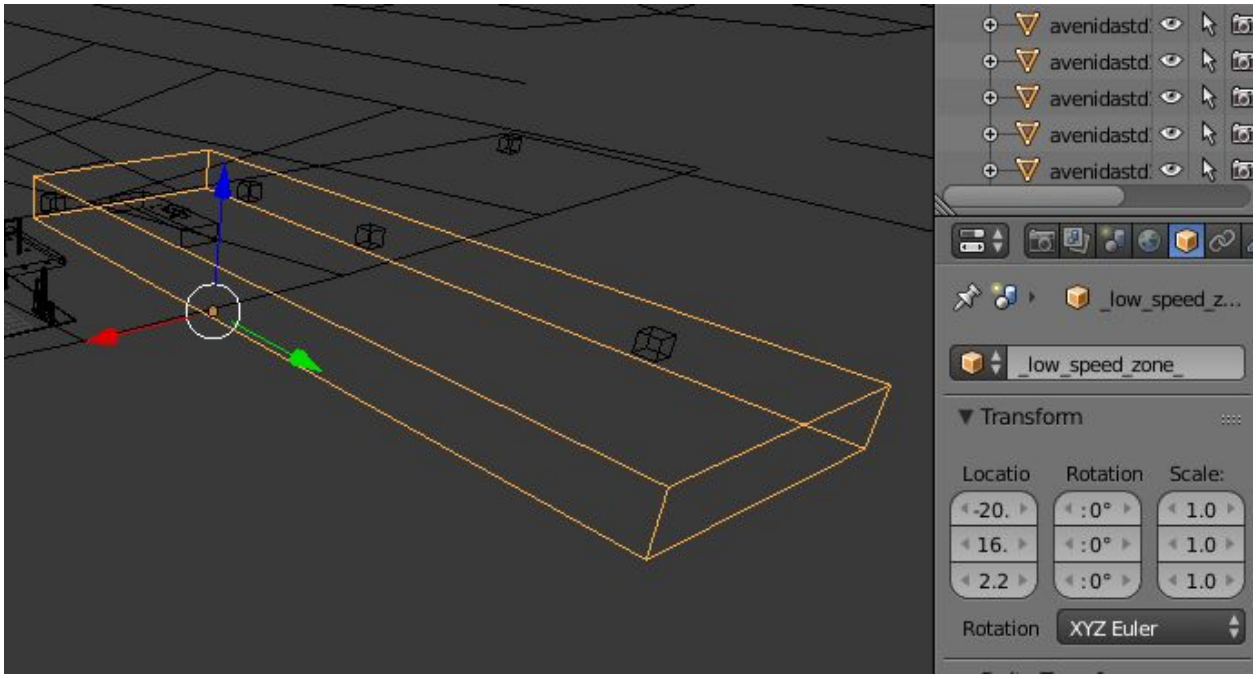
In this example you may leave the 3D model of the light pole without any collider, and create a new simplified one only for its main body with the name **_gencol_invisible_,** using a cube or a simple cylinder with very few faces (6, 8 etc, not many).

Light emitter always on panels, like for advertising

The **_emissive_** keyword change the object shader to leave it as if it were filled with a light. This is useful for luminous panels, like ads on bus stops or stores, reflective traffic barriers, electronic displays etc.

Low speed zone, so passengers won't complain on terminals

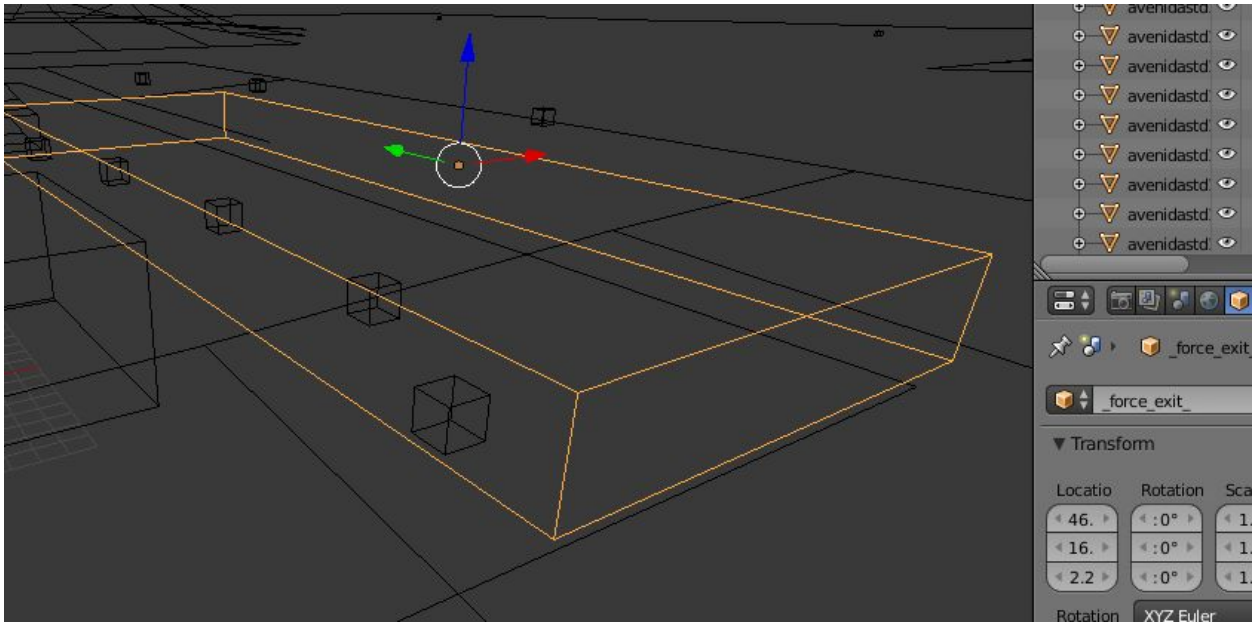
If the driver is too slow, people may complain about its speed asking to run faster. But in some cases it is mandatory keeping a low speed in real life, like inside bus terminals. To avoid those complaints on these places, you may create a cubic object filling the entire desired area, named **_low_speed_zone_**.



It may block your view while editing the 3D, just move it to another layer. This object will be an invisible trigger once inside the game. It does not need any material or texture.

End of the route, to force deboarding

When it is getting closer to the final destination, just a few meters before entering the road of the last bus stop or platform, you may use a cubic object with the name **_force_exit_** to force that all passengers gets off the bus on the next stop.



As the low speed zone, this object does not need to have a material.

IMPORTANT: IN FUTURE VERSIONS OF THE GAME THE BUS ROUTES MAY BE DEFINED FROM A BUS STOP LIST. SO THE GAME WILL KNOW WHAT IS THE LATEST STOP, WITHOUT HAVING TO RELY ON THIS OBJECT.

Extra details, to make a great experience for more players

Smartphones power is too variable, especially when looking for low cost models, popular on many places. Most worldwide players have a simple or old mobile phone. If we put a lot of details… The map could become too expensive to render and no one could play… It is very difficult finding a great balance between details and performance for mobile.

Trying to help with this problem, the game has the “Extra details” option at the map screen selection dialog, that players may enable or disable whenever they wish. Ideally some details that are not too important will be marked as an extra detail, so they will not be loaded for those players with simpler devices to make the gameplay experience better, to run smoothly.

The modding system also supports this feature!

Name the non very important 3D files with a **_det1** at the end of the name, before the extension. For example, **mainmap.3ds** for the map itself and **morethings_det1.3ds** for some extra objects. Please note that this naming must be applied to the file itself, not actual objects inside Blender!

Those that ends with **_det1.3ds** will only be loaded if the user enables that option to load extra details. It could save both graphics processing (with less triangles and textures) as well as RAM and video memory.

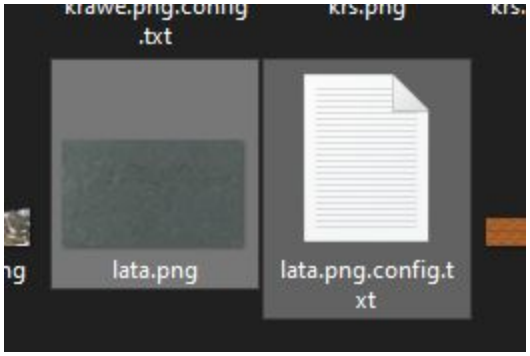
IMPORTANT: CURRENTLY THE EXTRAS DETAILS FILE MAY NOT SUPPORT PATHS FOR TRAFFIC, PASSENGERS ETC. USE IT FOR STATIC SCENERY OBJECTS ONLY, LIKE TREES, SOME BUILDINGS, PUBLIC TELEPHONES, OUTDOORS, PARK BENCHES, ETC.

Extra setting for floor, wall and roof textures to not blink

Some textures that fill a huge area may need a special treatment, otherwise they may appear blinking or glitching inside the game, making it appear that they are “alive” or shining.

For optimization purposes Proton does not make this treatment automatically, since it does not make a visual difference for most textures. So it is necessary setting them up manually for each texture. Generally this should be used only for textures that are used to fill large areas like grounds, terrains, asphalt, sidewalks, walls, huge ceilings etc.

Basically create an empty txt file with the same name of the texture, on the same folder, but ending with **.config.txt**. For example:



For example, for floor.png, the txt filename would be floor.png.config.txt.

IT IS RECOMMENDED ENABLING YOUR FILE MANAGER TO SHOW FILE EXTENSIONS FOR KNOWN FILES ON YOUR OPERATING SYSTEM TO AVOID ERRORS HERE.

If the texture had a blinking effect when driving or moving the camera, making this process may help to solve it. Again, remember: use this only when needed. This command makes the texture to consume a little bit more of RAM and graphics processing power. So if you apply it to every texture it could get worse performance.

Extras - Optimization tips

Generally, the less triangles and textures, the better.

As smaller textures are, its is better too. In many cases if you use a repeated texture on the object (tiled or seamless) you could set it to 256x256 or 512x512 pixels, for better performance. They do not have to be all 1024 if the level of details would not make much difference, depending on the actual size of the visual object. Use larger textures only when needed, like when you have to highlight some special smaller details as text or logos on bus repaints.

IMPORTANT: NEVER USE TEXTURES THAT ARE LARGER THAN 2048 PIXELS OF WIDTH OR HEIGHT FOR MOBILE DEVICES! MOST MOBILE GPUS CANNOT HANDLE THOSE TEXTURES, MAKING IT APPEAR STRANGE RANDOM GLITCHES OR EVEN CRASH THE GAME. IT IS NOT ABOUT WEAK PHONES, THIS HAPPENS ALSO ON SOME HIGH END MODELS.

If you have lots of objects with huge textures on the scene, like parked buses, some plates, outdoors etc, prefer leaving them at the extras details layer. So they will not be loaded for users that have that option disabled.

If needed, create two versions of the map: one normal, complete, for PC; and another one with resized textures for mobile phones. Depending on the map you may use the same files for both. If the map is too heavy, some users like to edit all textures by themselves and share it with others. Please do not worry if they do this for your map. The internal architecture of maps in Proton does not impose artificial restrictions, allowing that mods could be accessible to the highest amount of users whenever it is possible. If you do not “optimize” your map, probably others will try to do so. No one could have control about this.

Some objects may use a solid color, without any details. For this case you may use a very smaller texture like 4x4 pixels, with only the base color painted on it. Is is a waste of resources using a huge texture just to show one color only, since there is no more details inside it.

About colliders, as said before when dealing with the _gencol_ command, avoid generating colliders that are too complex or too huge. They increase a lot the number of mathematical calculations that the physics engine must do each frame. If your map is plane, prefer using one or more simple plane colliders for the ground on the same height than the roads, leaving the roads meshes without colliders. This also helps to make it more performant.

Objects of buildings like bus depots, terminals etc, do not need to have colliders for all vertices. You may use only on the ground and platforms, sidewalks, barriers etc, where the bus could naturally pass over. There is no need to place colliders on roof, ceiling or some columns. They would make the engine to do a lot of unnecessary calculations.

A large number of complex colliders may make the technical process of “floating origin” to become more intensive, increasing the delay that the game passes everytime when player stops the vehicle after driving for some kilometers.

Using only one object for the entire map may not be recommended, this is valid also for large complex objects, like walls with 1 or 2 km. They may have rendering problems with shadows or lights if their vertices are too far away from the others. On the other hand, using too many smaller objects may also be bad: more processing power will be spent to process and render all of them. There is no magic number that is valid for everybody, generally it is better testing a lot on real devices until you find a great balance between performance and quality. We recommend that you divide larger objects into smaller partes, especially if they have more than 40, 50, 100 meters. And this is even more important if they are complex objects, with more details on the mesh. When just a small portion of the mesh will appear on the screen, the entire object must be processed by the engine. This could cause an unnecessary performance drop if the engine have to handle a lot of triangles to show just a few of them. So it is really recommended to divide kilometric objects into smaller parts like roads, walls, roofs etc. This helps reducing the total number of triangles that must be rendered at each moment.

Avoid using too many materials on the same object. When there are two or three materials, the object must be rendered into many steps inside the engine, one for each material. This increases the time of the drawing process of each frame, causing fps drop. Prefer separating by

material. You could make some objects as child of others by parenting them on Blender (CTRL + P), it is better than making an entire object with everything together (CTRL + J). For example, in a bus terminal that uses many materials, leave each part separated from the others in a way that each one has only one material. Platform, floor, roof, etc. This generally is more efficient than leaving the entire bus terminal as an object only with 5 or 10 materials.

Combine smaller close objects that share the same material. This also helps reducing the amount of work that the processor and graphics card will have to do. A huge object is bad, but lots of smaller objects are also bad. Smaller objects that share the same material and are close to each other could be combined into one (CTRL + J on Blender) so they are rendered all together. This is valid for rocks, trees or even buildings. Be careful, because for this to work well, they must have one material only and be close in a small range of the map. Otherwise you would be facing the old problem of having an object that is too big, increasing the number of useless polygons that must be processed just to be discarded when only a small part of the object is visible.

If this sounds too confuse for you, try this: combine from 3 to 5 or 10 trees that are close to each other, if they share the same material. This way they will be rendered all together, saving processing resources when the engine must gets a list of all objects within the camera range that must be draw.

If the map is too large, it may appear some problems with precision for coordinates of positions inside Blender. Especially the more far away you are from the origin (the spot at 0,0,0). You could relax it by expanding to both sides of the axis, both negative and positive, keeping the map between the interval of -10 to 10km, for example. This generally is more efficient than going from 0 to 20km.

What are the impacts of making a very large map?

The system of map modding inside Proton bus is not suitable for large maps. We do not know if this will be possible in the future. We want it, but it is technically too hard to make. A large map must be divided into pieces, having those pieces loaded and unloaded as the player moves. The problem of this process is the huge impact on performance: every time that the game must load and integrate all the objects into the engine it will get a hiccup. This temporary lock is worse if the detail complexity increases. We do not want our game lagging due to this problem, so the map is entirely loaded at the startup. It may also gets some lag or hiccup for other reasons, but not for loading huge scenes. For example, spawners for people, cars, passengers etc must instantiate new objects as the player approaches them, and this has an impact on the main thread of the game. But we will try to avoid the worse hiccup that would happen if we try to load a new scene part while the player is driving.

So, due to this limitation, if the map has many kilometers, the amount of objects, polygons and textures tends to be too high. It may compromise performance. Or even worse, it could make the app to crash due to lack of RAM, especially on Android or 32-bit PCs.

It is not possible giving a fixed max size for the map, it will depend on the details level and many other factors. One specific thing to consider when making a large map is the spawner system for vehicles, people etc: they must be running a code at each time interval (the spawn interval). This code is distributed between some frames, but the more spawners on the scene, more frequently this code will run… This could make the game to run worse with less fps, both on mobile and on PCs. If the map has too many traffic points and some dozens or hundreds of kilometers, this may be the cause of the bad performance, especially if each road has many traffic lanes.

But it is still possible having great routes, with 20, 30, 40 minutes of driving: the old native map of the route 4520 was loaded all at once before it got integrated into Aricanduva. It had three traffic lanes at each side, with lots of spawner points, and it was still very performant for most mobile devices. It is worth making and testing it, being aware of this hard limitation.

If your map would have many bus routes, we recommend that you separate them in smaller maps, by creating one .map.txt for each region or route. They may share the same main folder to avoid duplicating all textures. This way it will be much more efficient since the game will load only the parts where the user will drive on.

The complex issue about open world in Unity

A side effect of driving on a long route in games made with Unity currently is the lack of precision of the numbers, and there is still no great solution for this limitation. Unity uses 32-bit floating point numbers to store data about positions, that can use only 7 digits of precision. If you drive for more than 1000 units (1000 meters or 1km) from the scene origin (the spot where x=0, y=0, z=0), there are left only three slots for the decimal part. After 10000 units (10000 meters or 10km) it gets only two decimal slots. Things get really bad here: shadow blinks, smaller objects appear shivering out of their original place, and so on… This engine does not have a great support for large open worlds in its classic programing model.

There is simple no magic solution, only some workaround. Proton Bus will move everything in the scene trying to keep the camera near the 0,0,0 spot, so the engine will never run out of slots for decimal parts of coordinates. This is the origin reset process, or floating origin. But this is an expensive operation. To move all objects, especially colliders, the physics engine must redo a lot of calculations since all positions changed all at once. Translating it to the real world: the game will stop for a moment. Generally a fraction of a second. But it is not good. We avoid doing it when player is driving, to avoid breaking the immersion of the experience. This will be done when the player stops the bus. So if you drive for some kilometers without stopping the vehicle for a moment, you will experience bad things happening.

We will continue trying to research to improve this limitation, but we are depending on changes on the engine itself, it is a closed box that we do not have full access. Other engines may apply other solutions, but in our reality there is nothing much else that we could do.

We all hope that someday Proton will support really large maps, with scenery streaming. But we do not know if this would be possible. It will not be a promised feature. If it works, great. If not, no problem, let’s enjoy the current system that is working well for most of us.

Debug tools

You can enable debug tools to see some things that are normally invisible, regarding mods. On versions like 252 or 253 these tools are at the bus and map selection screen, but in the future they may be moved to some item at Extras, to avoid that normal uses enable them.

Basically there will be outlines on special colliders like bus stops, triggers, as well as on the paths for pedestrians, vehicles and trains. This could help finding potential problems that could go unnoticed on Blender, like floating people or some object going to the wrong direction.

Never use pictures with width or height larger than 2048 pixels on current mobile devices!

Again, we ask, please: both for buses and maps, do not use larger images with dimensions larger than 2048 pixels for maps to run on mobile phones!

Even on PC these huge pictures are not normally necessary, except for repaints or plates with smaller logos or texts.

In current versions users may enable an option to force loading those huge textures, but this brings too many support problems that it could be removed someday. Please try to avoid them. On the compilation process the Unity engine itself resizes all native game textures to a max of 2048 pixels for Android, if they were larger than this limit. On mods no one have full control, it depends on the creators and users. One fact is that we cannot handle complaints and support cases about a “problem” that it is not ours, because it delays the development of new things. Many glitches or random crashes are caused due to use of these larger textures.

Here is an explanation of the process for users:

http://www.busmods.com/blog/static/1571653273imagens-grandes/

And here are some details during the discovery process (in Portuguese):

http://blog.protonbus.com.br/2019/10/importante-evitem-mods-com-texturas.html

We would like that you respect this rule.

Even when those textures do not cause glitch or crash, they have the disadvantage of occupying almost the entire bandwidth between GPU and CPU or storage and RAM. There is a physical limit on the transfer speed that vary depending on the device, as if it were an internet connection or a pipe with a certain water flow limit. By using lots of huge textures you make the system to wait more time between each step when they have to process them, which happens when some of them needs to appear on the screen or on the mirror, causing fps drops or hiccups. With smaller textures the bandwidth runs better, having more space to load and unload more textures at once.

Always test on a mobile device, if possible!

This is very important: for both mods of buses and maps, they all should be tested on some real mobile device before released, to check if everything is ok. Perhaps some performance problems or internal collider bugs generated by the engine only happen on some devices but not on PC… This happens a lot during our development process: it works well on the Unity editor and on PC, but throws many crazy errors on mobile devices...

There is no problem if you want to make a map for PC only, but in this case it is great telling a message when launching it to not disappoint mobile users.

More than 95% of Proton players are on mobile phones, it is worth it targeting these people.

That’s it for now!

Thank you very much for your interest in making maps for Proton Bus!

If you create a great free map, do not forget to show it on the game groups!

Have fun!