SnowRunner:
Console Requirements for Mods.
Optimization

Version 0.9.6 of Feb 2, 2022

Contents

1. Introduction

Mods on consoles pass a selection and approval process before we make them available for consoles.

This guide provides some requirements and recommendations that can help in making mod approved for consoles and in its optimization in general.

2. Primary Requirements

The basic list of requirements for a mod to be eligible for mods on console:

- No offensive or inappropriate content.
- Mod titles need to be in English
- Special characters are not allowed (such as ^ / % etc.)
- A mod cannot surpass 1GB in size.
- A mod must have no visible glitches or render bugs.
- A vehicle mod must allow the player to progress through the game to some extent and must be possible to play without any blocking issues. Before making a mod available, we will test this on our side.
- In regards to licenses and branded vehicles, we will provide a list of the brands that allow the modding of their vehicles. All other mods of licensed vehicles are at risk of not being accepted. This list is not intended to be final and should grow over time.

Link to the post with these requirements on the official forum:

https://forums.focus-home.com/topic/55567/modding-in-patch-10-and-beyond

However, along with these basic requirements, there are a lot of technical requirements and recommendations, see below.

3. Technical Requirements

3.1. Naming of Textures. Format of Textures

Textures used for custom trucks and custom models must follow the requirements described below.

3.1.1. Correct format for all textures

All texture files must be in the **.tga** format.

3.1.2. Correct naming scheme for all textures

All textures must be named according to the correct naming scheme. Particularly, the naming of textures is strictly defined, postfixes in their file names indicate the type of data inside them. The Editor and the game itself use these postfixes to correctly convert the initial texture and pack the mod correctly.

The name of the file is formed according to the following pattern:

name__postfix.tga (e.g. azov_4220_antarctic_front__d.tga)

Where:

- **name** – any appropriate name that does not include two underscores in a row (no "__" inside it).
- **postfix** – defines the type of data within a texture according to the table below.

__postfix	Type of Texture	Parameter in XML
__d	Albedo Map	AlbedoMap
__d_a	Albedo Map + Alpha	AlbedoMap
__n_d	Normal Map	NormalMap
__sh_d	Shading Map	ShadingMap
__em_d	Emissive Map	EmissiveMap

WARNING: The naming scheme for textures described above is mandatory. This naming scheme is necessary to enable streaming for textures of the mod, which may be essential. Particularly, mods with the wrong naming of textures result in memory leaks on consoles and will not receive approval for them.

For descriptions of XML parameters and other details, see "***7.2. <Material>***" in the "**Integration of Trucks and Addons**" guide (**Integration_of_Trucks_and_Addons.pdf** in the same documentation package).

3.1.3. Textures for Color Customization

Textures used for color customization of the trucks (base color and tint) are diffuse maps and, because of that must have **__d** postfix. However, for easier identification of these textures, we recommend adding auxiliary suffixes (**_wt** and **_cc**) to their names, as shown in the table below.

_suffix__postfix	Type of Texture	Parameter in XML
_wt__d	The map with the base color of the surface.	AlbedoMap
_cc__d	The map that defines the paint zones.	TintMap

For details on these textures, see "***15. Color Customization***" in the "**Integration of Trucks and Addons**" guide (**Integration_of_Trucks_and_Addons.pdf** in the same documentation package).

3.2. Polycount: Suggested Average Values

Values of the **polycount** for mods of trucks should have **200K** triangles as the upper limit.

Particularly, approximate values of polycount for different parts of the truck model are the following:

- **External part of the truck** (without wheels) – up to **100K** triangles and not more.
- **High-res interior** of the truck (for the cabin view) – up to **40K** triangles and not more.
- **Other elements** (including wheels, visual customization, etc.) – up to **60K** triangles and not more.

As you can see, the total value for all parts of the truck should be lower than **200K** triangles. If the polycount of a truck mod exceeds this value, this may be the reason for not receiving approval for consoles.

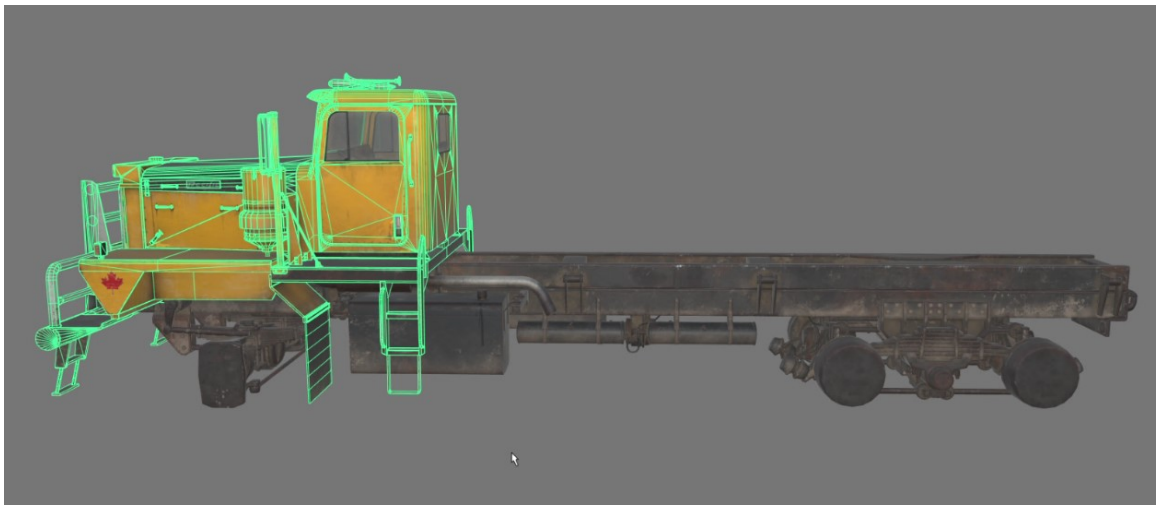
> **TIP**: Keep the polycount small for parts of the model that cannot be seen or are hardly visible for the player (e.g. engine).

Polycount of custom models can vary greatly depending on the model. However, in general, you should not use a large amount of custom models for the level. The less is the amount of custom models, the better the level is optimized for consoles (see 3.5. Reuse of Standard Resources below).

3.3. Size/Texel of Textures: Suggested Average Values. Mapping.

Suggested sizes/texel values of textures used for the standard are the following:

- **External part of the truck and its low-poly cabin**
  - For a regular truck – **1 texture**, **2048x2048**.
  - For a large truck (e.g. Pacific, CAT745, etc.) – **2 textures, each of 2048x2048 size:** one for the front part of the truck (selected part on the picture below) and one for the rest of the model (non-selected part on the picture below, i.e. chassis, and so on).

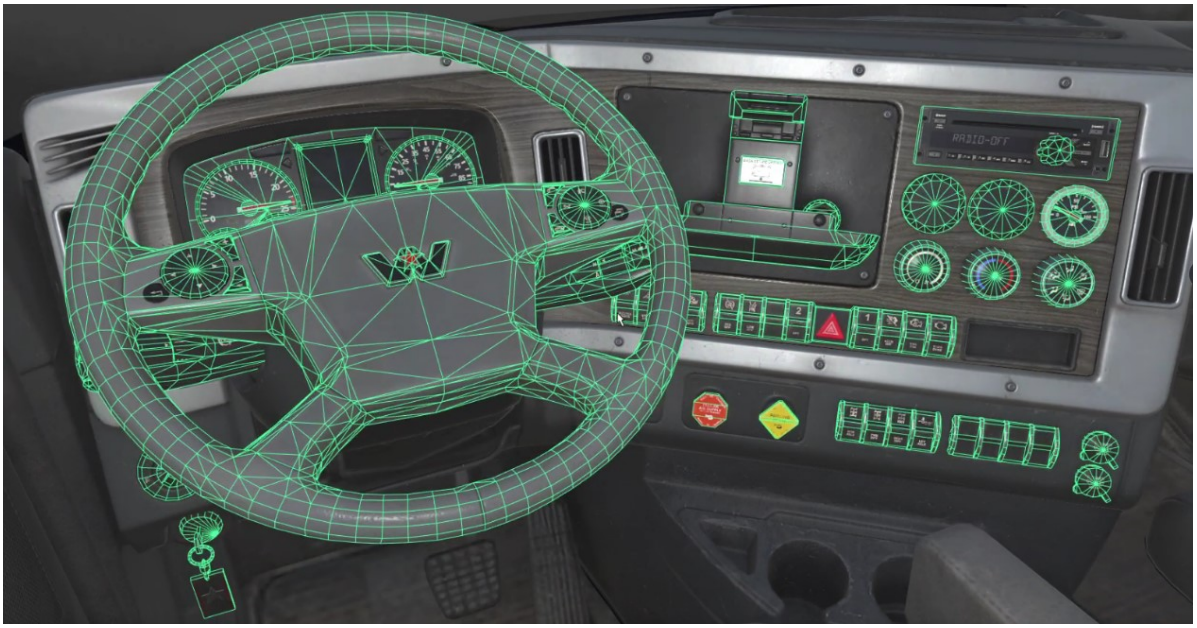


- **Glass** (all glass part of the truck) – **1024x1024** or **2048x1024** (depends on the format of the glass)
- **High-res Cabin** – **2K**.
- **Gauges** – **1024x1024** or **2048x1024** (depends on the number of elements).
- **Glass of the gauges** – **512x512**.
- **Tire** – **1024x512**.
- **Rim** – **512x512**.
- **Addons** – may vary (depends on size/complexity), texel is **~230**

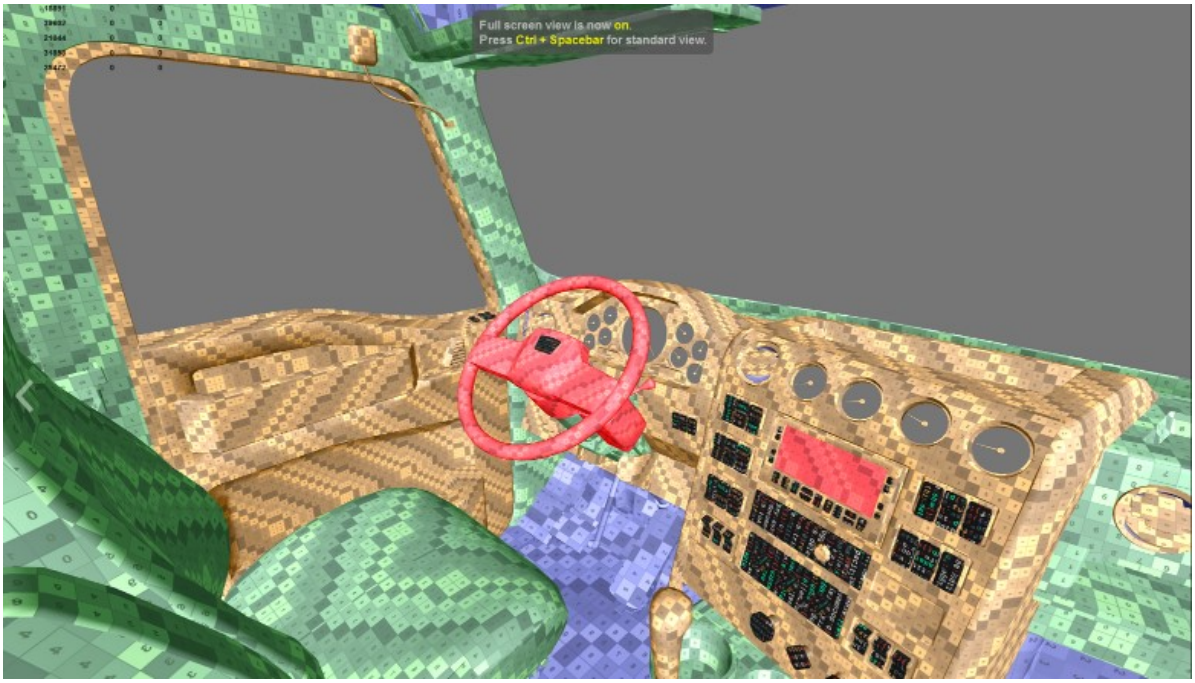
The size of textures for custom models can vary depending on the model. However, in general, you should not use a large amount of custom models for the level. The less is the amount of custom models, the better the level is optimized for consoles (see 3.5. Reuse of Standard Resources below).

3.3.1. Correct mapping of a dashboard and gauges

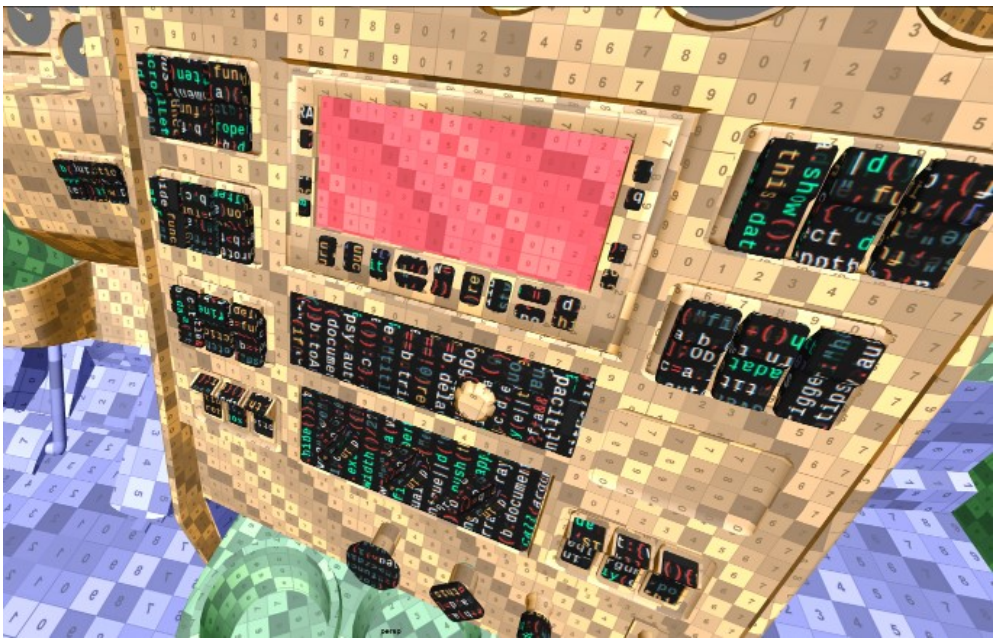
It is important to make a correct mapping for the dashboard and gauge elements, otherwise the text on the dashboard will be hardly visible and unreadable. To avoid that, you need to map all text elements with the large texel. As a rule, all these elements are mapped within the gauges texture, along with a steering wheel. The gauge texture is typically 1K; if necessary, it can be increased to 2x1K.



Small elements of the cabin are mapped taking into account the typical camera position. I.e., elements near the camera should have large texel, distant elements (or hardly visible elements) – with lower texel.

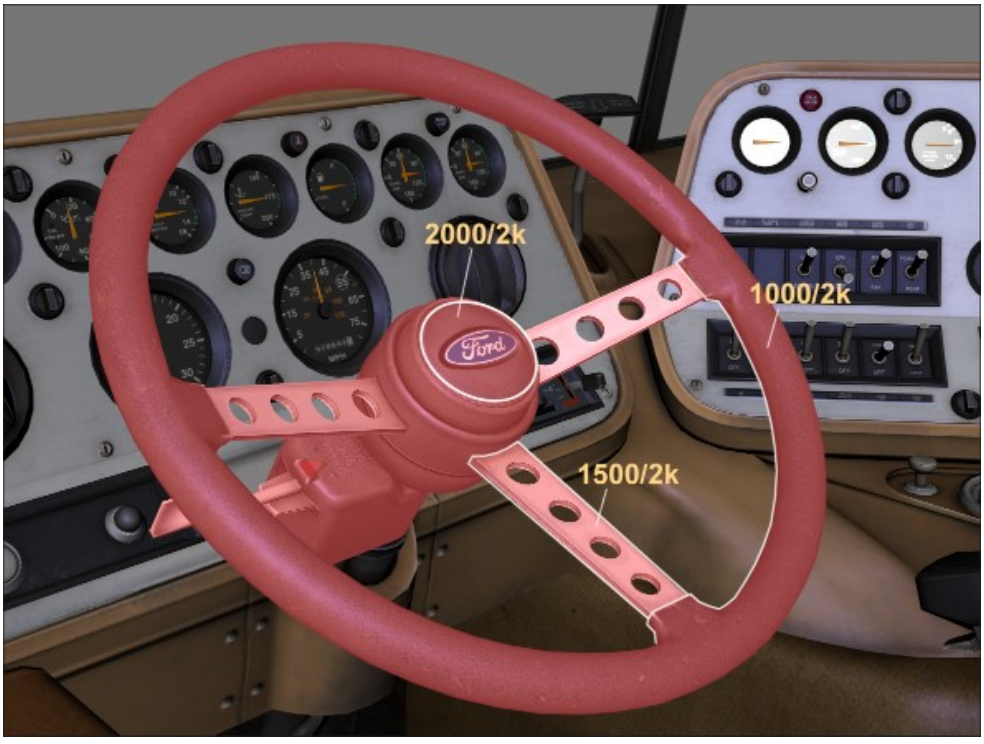


The process is typically as follows. First of all, using photo references, determine areas of the dashboard with graphical information (buttons, car radio panels, switchers, etc.). Use the corresponding texture for them in the 2K resolution for better readability. Or, stick to the minimum value of texel near 1500 for 2K.



After that, arrange the corresponding UV shells on the map.

Use the following texel values for the steering wheel (see red areas in the picture below).



Various gauges, displays, arrows are mapped within the separates gauges texture, along with a steering wheel. Then, you need to duplicate gauge faces/displays to imitate the protective glass (with an appropriate offset to avoid z-fighting).

3.4. Number of Physical Bones Synched in COOP

When creating an XML description of the truck or addon, and, particularly, the description of its physical model, one should take into account the synchronization of these bones in the COOP mode of the game.

By default, positions (**p**osition) and speeds (**v**elocity) of all bones of the physical model of the truck are synchronized. However, using the "**Legacy**" attribute of the **NetSync** tag (**`<NetSync Legacy="false" />`**), such total synchronization can be disabled. In this case, only the bones (Body) with the "**pv**" value of the **NetSync** attribute of the **Body** tag (**`NetSync="pv"`**) will be synchronized.

> **NOTE**: For details, see "***8.4.1. <NetSync>*** " in the "**Integration of Trucks and Addons**" guide (**Integration_of_Trucks_and_Addons.pdf** in the same documentation package).

The number of synchronized bones affects the network performance of the truck mod in COOP. The impact (and network conditions that result in issues) may vary greatly, however, in most cases, the **15** is the approximate upper limit for the number of these synchronized bones. I.e., when their number is higher than **10**, the issues may occur sometimes; when their number is higher than **20**, you can be almost sure that some issues will appear.

3.5. Reuse of Standard Resources

To optimize your mods during their creation, you should try to reuse standard resources provided by the game and Editor. All standard resources are already optimized and less frequently cause issues.

3.5.1. Higher amount of standard models

The lower the number of custom models used by your map, the better your map will be optimized for consoles and the less size it will have.

Try to use standard models provided by the SnowRunner Editor more frequently, combining them in different variations.

3.5.2. Higher amount of standard sounds

The usage of standard sounds for your map or truck will also make your mod more optimized for consoles.

Standard sounds are available within the **shared_sound.pak** in the installation folder of the game. They are available there in the PCM format. However, when you are using standard sounds you do not even need to extract them, all you need is to *link* to standard sounds that are available with the game.

I.e., you simply need to specify correct paths to them in the corresponding fields, see "***5.12. Adding Sounds***" in the "**SnowRunner Editor Guide**" guide (**SnowRunner_Editor_Guide.pdf** in the same documentation package).

The same method is valid for the music and ambient sounds.

3.6. Instanced Rendering for Custom Models

If you are using custom models in your map mods and the number of instances of these models is large, you need to use instanced rendering (instancing for these models).

For details, see "***2.3.1.1. Rule #1: Attributes for Instanced Rendering***" in the "**Custom Level Entities. Models, Overlays"** guide for details, available as **Custom_Level_Entities_Models_Overlays.pdf** of the same documentation package.