

The Big Book of SnowRunner Map Making

Revision 1.1 (2021-09-24)

Copyright 2021 Chris Nelson



This work is licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](#).

If you own the Windows version of SnowRunner, then you can make and publish your own custom maps.

This guide documents every feature of the SnowRunner map editor. You can read (or skim) sequentially through the guide to learn everything there is to know, or you can use it as a reference when you want to learn more about a particular feature.

If I had to experiment to figure out how something worked, why should you have to run that experiment again for yourself? Everything useful that I've learned went into this book. Is it all a bit much? Yes. But is it what you need to know to make a map that matches your vision? Also yes.

What I put in this book that I can't find elsewhere:

- The quickest of [quick starts](#).
- Detailed information about **all** editable map properties.
- Full instructions for complex editing tasks.
- Built-in differences in behavior among the various objects and object types.
- Bugs to avoid. (So many!)
- Reference tables, including a concise chart of the capabilities of trucks and trailers.
- And more!

This book does **not** cover the creating of modded trucks, only maps.

If you are viewing this book on my [Google Drive](#), I encourage you to download the PDF (20 MB). The PDF includes clickable cross-references and is generally easier to flip through.



Table of Contents

Quick Start	10	Dev Tools: Mod Tools	43
Run the Map Editor	10	Revealing the Map	44
Create a Fresh Map	11	Discovery of Vehicles	44
Move the Camera	14	Time	45
Make Hills and Depressions	15	Take a Screenshot	46
Place a Truck	18		
Save and Pack	22	Introduction to the Map Editor Window	47
Make a Backup Copy of your Campaign	23	Menu Bar	47
Test Your Map	23	Toolbar	49
Re-editing a Map	26	Main Panel	50
Example Maps	26	Navigate in the Main Panel	50
Introduction	27	Output Panel	51
How to Use This Book	28	Terrain Panel	52
What Kinds of Maps Can I Make?	28	File View Tab	53
Sources of Information	29	Controls for Hierarchical Lists	53
Is This Book Up To Date?	29	Scene View Panel	53
Literals and Variables	29	Introduction to Features	54
Editor and Game Bugs	30	Object Groups	55
Broken Assets	31	Property Panel	55
Design Tips	31	Edit a Property using a Widget	57
If You Made Maps in MudRunner	31	Edit a Property by Typing Directly	57
Useful Files and Archives	33	Edit a Property Using a Dropdown Menu	58
Media Directory	33	Edit a Property Using a Slider	58
Map Source Files	33	Edit a Property That Selects an Asset	59
Derived Map Files	34	Edit a Property that Selects Brushes	62
Game Files	34	Rearranging Panels	63
Upload Files	35		
Asset Icons	35	The Context Menu	64
Limited Backups	35	Reload	65
Editor Configuration Files	37	Rebuild Terrain	65
Game Save Data	37	Add Feature	66
Test Your Map	39	Fly To	66
Pack Your Map	39	Feature Context	66
Dev Tools	39	Category Context	68
Dev Tools: Create	40		
Dev Tools: Reload	41	Map Features	69
Dev Tools: Information	41	Introduction to Top-Level Properties	70
Dev Tools: Virtual Garage	41	Map Name	70
Dev Tools: Free Camera	42	Map Description	70
Dev Tools: Cargoes	43	Map Dimensions	70
Dev Tools: Repair & Refuel	43	Map Borders	71
Dev Tools: Change Time	43	Maximum Height	72
		Trucks and Trailers	74
		Move and Rotate a Truck	75
		Move and Rotate in the Main Panel	75

Move a Truck in the Scene View	76	Plan with Colorization	117
Land Property	77	Adding Natural Tints	117
Local Transform	80	Models	119
Select Truck Components	80	Broken Model	120
Vehicle Type	80	Move, Rotate, or Scale a Model	120
Identify Compatible Truck Parts in the Virtual		Scale	122
Garage	82	Model Physics	123
Engine, Gearbox, Winch, and Suspension	82	Collision Notification	124
Wheels	83	Freeze Physics	125
Color	84	Special Models	125
Trailer	85	Cargo	125
Add-ons	85	Country-Specific Models	126
Truck ID	88	Rocks and Ice	126
Starting Damage and Fuel	88	Winch Attachment Points	127
Locked and Unlocked Vehicles	89	Lights	127
Reserved Vehicles	90	Farplane Models	130
The Active Truck	91	Animated Models	130
Hand Edit the XML	91	Animals	130
Terrain Height	92	Multi-Stage Models	131
Use Height Data from an External Source	93	Elevated Overlay Bases	133
Height Brush Mode	93	Other Aspects of Appearance	133
Autofade	95	Shadows	134
Flatten Brush Mode	96	Dynamic Shadows	134
Smooth Brush Mode	97	Static Shadows	136
Brush Keyboard Shortcuts	98	Occlusion	138
Automatic Terrain Rebuild	98	Individual Plants	139
Height Information	98	Move, Rotate, or Scale a Plant	140
Deceptive Height Cues	100	Terrain Height Adjustment	144
Navigate While Painting	101	Plant Physics	144
Restore a Previous Terrain State	102	Country/Season-Specific Plants	145
Undo/Redo	103	Winch Attachment Points	145
Undo of Object and Node Changes	103	Other Aspects of Appearance	145
Undo of Paint Changes	104	Distributions	147
Mud	105	Choose Plants	147
Extrudes Mode	105	Choose a Bitmap Filename	148
Extrudes to Wetness	108	Paint the Distribution Region	148
Automatic Mode	109	Plant Density Restrictions	149
Automatic Mode With Size > 2.5	109	Scale-Dependent Brushes	150
Automatic Mode With Size \leq 2.5	110	Grass	150
Decrease Mud	111	Automatic Distribution after Painting	151
Restore a Previous Mud State	111	Random Seed	151
QuickMud	112	Interaction with Roads, Water, and Mud	151
Colorization	114	Interaction with Material Textures	152
Choose a Color	114	Model Occlusion	153
Paint with Colorization	115	Distribution Summary	153
Colorization of Features	115	Materials	154

Material Properties	155	Ruler	204
Material Layer Properties	156	Manipulating Nodes	205
Paint a Material Layer	158	More Ruler Stuff	206
Plants, Grass, and Debris	160	Rivers	207
Blending	160	River Node Properties	207
Value Blending	160	River Speed	208
Edge Blending	161	Foam	210
Blend of Grasses	162	Cascades	211
Additional Materials	162	River Junctions	211
Texture Edges Between Materials	165	Flow Map	212
Snow Layer	166	Water Color	213
Procedural Snow	167	Background Water	214
Delete a Material	167	River Sounds	215
Winter Map	169	Wetness	217
Snow Layers	169	Breakable Ice	218
Soft Snow and Crust Snow	170	Point Sounds	220
Snow Depth	171	Point Sound Visualization	220
Update Snow Layer	172	Name	220
Interaction with Plants	173	Sound Location	221
Procedural Snow	173	Sound Volume and Radius	221
Procedural Snow on Models	174	Sound Files	222
Procedural Snow on Plants	177	Built-in Sound Files	222
Procedural Snow on Trucks	177	Custom Sound Files	223
Procedural Snow on 3-D Overlays	177	Delay Between Sounds	224
Procedural Snow on Landmarks	178	Sound Conditions	224
Procedural Snow on Material Layers	178	Ambient Sound Domains	225
Interaction with Colorization	179	Sound Domain Visualization	225
Overlays	180	Name	226
Manipulating Nodes	181	Sound Domain Location and Shape	226
Overlay Properties	184	Update Vertices	228
Type	184	Add a Vertex	228
Flatten	184	Delete a Vertex or a Sound Domain	229
ApplyOffset	184	Duplicate a Sound Domain	229
Periodic Models	184	Sound Files	229
Roads	186	Overlay	229
Road Ends	186	Volume and Fading Distance	230
Road Dropouts	187	Sound Conditions	230
Flatten	189	One-Shot Sound Domains	231
ApplyOffset	190	Name	231
Road Node Properties	191	Sound Domain Location and Shape	231
Conforming Overlays	194	Sound Files	231
Cliffs	196	Volume Range	232
Elevated Overlays	197	Radius Range	232
Wires	198	Delay Between Sounds	233
Model Attachment Points	201	Weather Intensity Threshold	233
Other Aspects of Appearance	202		

Sound Conditions.....	233	Zones and Objectives.....	265
No-Music Sound Domains.....	234	Zones.....	266
Name.....	234	Zone ID.....	266
Sound Domain Location and Shape.....	234	Move or Rotate a Zone.....	267
Hysteresis.....	234	Zone Name and Icons.....	267
Fade In and Out.....	235	Zone Perimeter.....	269
Reference Maps.....	236	Perimeter Shape and Size.....	270
Create a Reference Map.....	237	Vertical Perimeter.....	271
Reference Merge Map.....	237	Flat Perimeter.....	272
Combined Features.....	238	Other Perimeter Properties.....	273
Inter-Reference Conflicts.....	244	Hilly Zones.....	274
References within References.....	245	Zone Perimeter on Models.....	275
Mutator.....	245	Zone Settings Editor.....	276
Toggle No References.....	246	JSON Files.....	276
Groups.....	247	Manage the Zone ID.....	276
Group Property of a Feature.....	248	Add or Delete a Zone.....	276
Group Context.....	248	Rename a Zone ID in the Main Editor.....	277
Change Group.....	249	Rename a Zone ID in the Zone Settings Editor.....	277
Top-Level Editor Properties.....	251	Clear the Zone ID.....	278
Animation Speed.....	251	Uncommitted Id Edits.....	278
Mud Type.....	251	Duplicate IDs.....	279
Mutator.....	251	Zones Used by Objectives.....	279
Procedural Snow Coverage.....	251	Zone Settings Editor Controls.....	280
Sun Static Direction.....	252	Zone Settings Editor Properties.....	281
Sky Preset.....	252	UI Text.....	282
Ambient Sounds.....	252	Character Limit.....	282
Daytime Presets.....	253	Formatting Tags.....	283
Daytime Presets over Time.....	254	Localization.....	284
Force Daytime Preset.....	255	Global or Regular Zone ID.....	285
Extrudes to Wetness.....	255	ModMapError.....	287
Background Water.....	255	Map Initialization.....	288
Details of the Map Editor Window.....	256	Map Introduction.....	288
Menu Bar.....	256	uiTitle.....	288
Toolbar.....	256	uiText.....	289
Main Panel.....	258	uiIcon.....	289
Select Multiple Objects.....	259	Automatically Activated Objectives.....	289
Manipulate Multiple Objects.....	259	Map Images.....	290
Output Panel.....	260	Map Image in Global View.....	290
Terrain Panel.....	260	Map Image in Save Slot.....	291
File View Tab.....	260	Dev Tools Menu.....	291
Controls for Hierarchical Lists.....	262	Ignore Country in Garage Store.....	291
Keyboard Controls.....	263	Map Cloaking.....	292
Scene View Panel.....	263	Zone Types.....	292
Property Panel.....	264	Objectives.....	292
		Content Settings.....	292

Lock/Unlock Content	293	Reward Description	327
Starting Parameters	294	Multi-Stage Models	328
Custom Prices	294	Potential Model Collisions	330
Zone Types	295	Objective Stages	331
Garage Entrance	296	Conflicting Properties	331
Garage Exit	297	Cargo Delivery Goal	332
Recovery Zone	299	Goal Description	332
Trailer Store	299	Cargo to Deliver	333
Fuel Station	300	Delivery Zones	333
Service Hub	300	Manual Unloading	334
Watchpoint	300	Required Truck	338
Upgrade Zone	302	Multi-Stage Models	338
Automatic Cargo Loading Zone	303	Spawn Cargo for Stage	340
Multi-Stage Model	305	Cargo Spawn Location	342
Manual Cargo Loading Zone	305	Cargo Spawn Height	342
Cargo Generation Location	306	Logs	342
Cargo Generation Height	307	Highlighting of Cargo Sources and Destinations	343
Cargo Generation Conflicts	307	Revealed, Unrevealed, and Invisible Cargo Sources	343
Logs	308	Highlighting of Cargo Sources	344
Crafting Zone & Storehouse	308	Highlighting of Cargo Destinations	345
Energy Generation Zone	311	Recommended Cargo Sources	345
Living Area	312	Cargo Sources when Spawning Cargo Requires the Metal Detector	348
Gateway	312	Truck Delivery Goal	350
Multiple Zone Types	312	Truck Repair Goal	355
Objectives	313	Change Truck Goal	357
Objective Status Terminology	313	Visit Goal	359
Custom Employers	314	Seismic Vibrator Goal	360
New Employer	314	Living Area Goal	361
Modified Employer	314	Multiple Goals	362
Add Objective	314	Zone Visibility	363
Requirements	315	Visibility Based on Zone Type	364
Offer Zone (Task or Contest)	316	Visibility Based on Objective and Goals	365
Employer (Contract)	317	Regions	367
Objective Description	317	Create a New Region	367
Recommended Equipment	318	Region Name	368
Rewards (Task or Contract)	319	UI Text	369
Experience	319	Add and Arrange Maps	369
Money	320	Save a Region	370
Item	320	Pack a Region	371
Access to Zone	320	Open a Region	371
Multiple Rewards	322	Region Properties	371
Rewards (Contest)	322	Region Name and Description	371
Fail Reasons (Contest)	324	Region Images	372
First-Person View Requirement	325		
Damage Limit	325		
Time Limit	325		
Contest Hours	326		

Dev Tools Menu	373	Build Bridge	413
Ignore Country in Garage Store	374	Build Other Structure	414
Map Cloaking	374	Specialty Models	418
Region Content Settings	374	Appendix B: Hand Edit Text Files	420
Region Zone Types	374	Position Coordinates	420
Gateway	375	Position Coordinates in Meters	420
Regional Contracts	379	Terrain Block Coordinates	421
Cross-Map Contract Restrictions and Notes	380	Orientation Coordinates	421
Publish Your Map	382	2-D Orientation of Reference Maps	421
Subscribe	384	2-D Orientation of Zones	422
Activate	385	3-D Orientation of Trucks, Models, and Plants in the Editor	423
Play	386	3-D Orientation of Trucks, Models, and Plants in the XML File	424
Appendix A: Reference Information	387	3-D Orientation of the Camera	425
Truck Reference	388	3-D Orientation of Wire Nodes	426
Table Column Meanings	388	SnowRunner Built-In Assets	426
Scout Trucks	390	Map XML Files	427
Highway Trucks	391	Map and Region JSON Files	429
Heavy Duty Trucks	391	Distribution Brushes: brushes.xml	430
Offroad Trucks	392	Mutators: mutators.xml	431
Heavy Trucks	393	Mutator Definition	431
Trailer Reference	394	Mutations	432
Table Column Meanings	394	Ambient Music: music_presets.xml	433
Scout Trailers	394	Level Definition	433
Regular Trailers	394	Music While Driving	434
Low-Saddle Semi-Trailers	395	Music in the Garage	436
High-Saddle Semi-Trailers	395	Appendix C: Hand Edit Bitmap Files	437
Other Trailers	395	Bitmap Coordinates	437
Cargo Types	396	Bitmap Dimensions	438
Material Layer Reference	398	Color Channels	438
Employers	401	Channel Depth	439
Zone Icons	403	Use an Edited Bitmap	439
Travel Services	403	Use a Different File Type	439
Objectives	404	Colorization & Wetness: _base_tints.tga	440
Cargo Areas	405	_merge_mud_wetness.tga	440
Vehicles	406	Snow Depth: _snow.tga	441
Ruins	407	Water Speed and Foam: _water.tga	442
Miscellaneous	408	Water Flow Direction: _water_flow.tga	443
Additional Icons	408	Default Water Flow Direction: default_water_flow.tga	444
Rock Model Reference	409	Water Color: _water_mud.tga	445
Multi-Stage Model Reference	410	Material Opacity: mtrl_layout.tga	446
Remove Barrier	411		
Open Barrier	411		
Deconstruct Structure	412		
Repair Structure	413		

Fix the Material Opacity Alpha Channel in Gimp	Change Log	460
		446
Material Layer: mtrl_base.blends.tga		448
Accurately Paint Snow Variations with the Aid of Gimp		448
Distribution Density & Scale: dstr_*.tga		452
Fix Distribution Scale Randomness in Gimp		452
Terrain Height: _height.dds		454
River Height: _water_height.png		456
Mud		457
QuickMud: _quickmud.tga		458
Reference Merge Map: _ref_merge.tga		459

Quick Start

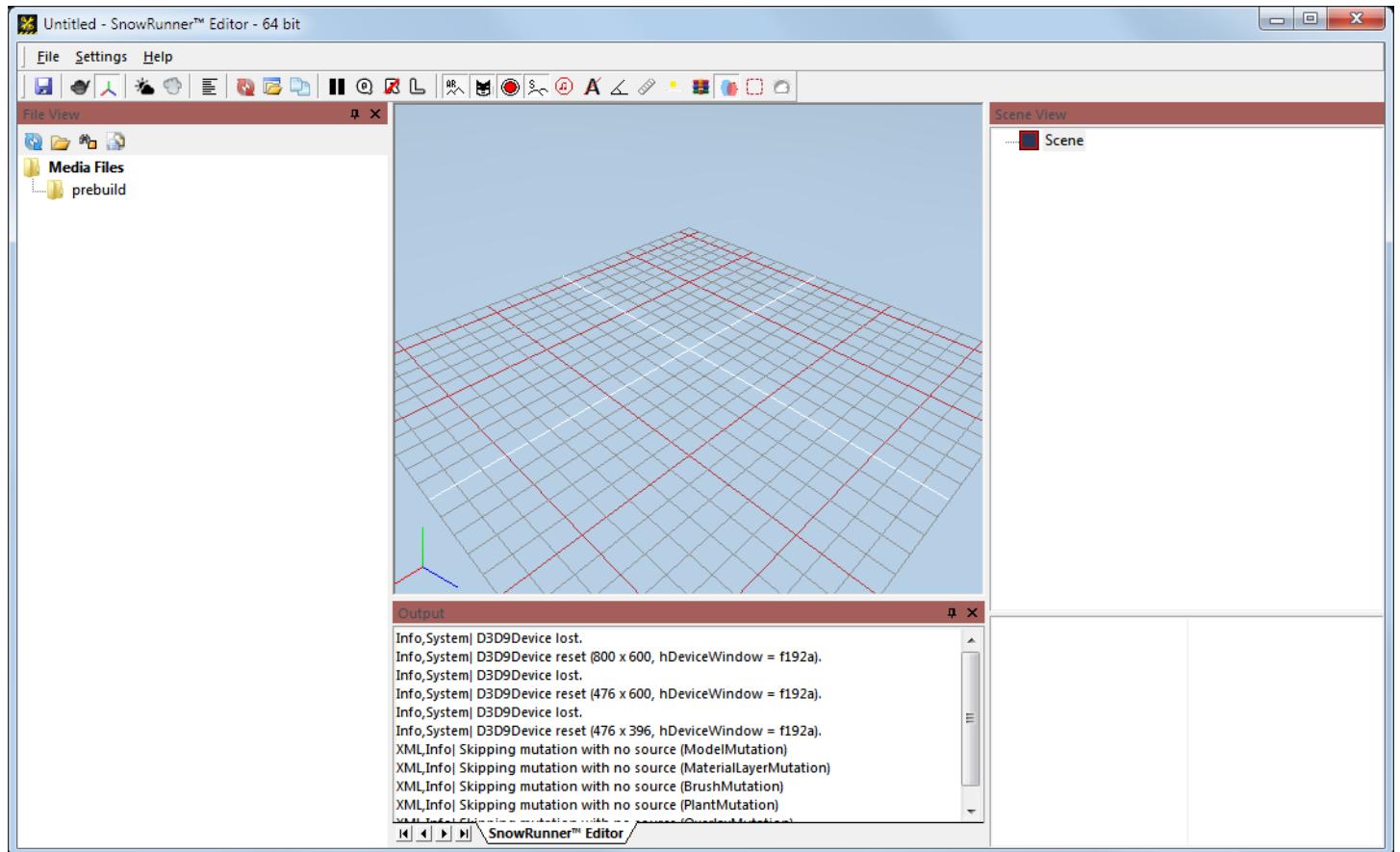
Before getting into the nitty gritty of map editing, you'll want to create and test your first simple map as quickly as possible.

Run the Map Editor

Installing the SnowRunner game also automatically installs the map editor, but it doesn't install a convenient shortcut icon. You can start the map editor from a path that looks something like this, depending on whether you have the Epic or Steam version of the game:

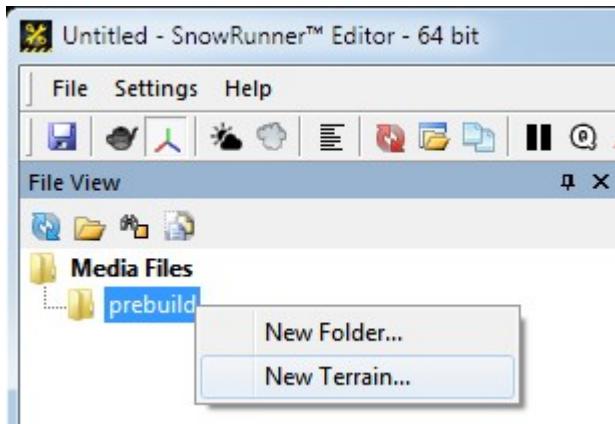
```
C:\Program Files\Epic Games\SnowRunner\en_us\Sources\BinEditor\SnowRunnerEditor.exe  
C:\Program Files (x86)\Steam\SteamApps\common\SnowRunner\Sources\BinEditor\SnowRunnerEditor.exe
```

The initial map editor window looks like this:



Create a Fresh Map

To create a new map, right-click on **prebuild** under **Media Files**, then select **New Terrain...**



A dialog box pops up asking what to name the map and what size to make the map.

The map name **must** begin with **level**. This is how the game recognizes it as a complete playable map and not an unplayable map section. After the **level** part, the rest of the name must be all lowercase letters, digits, and underscores. E.g. no capital letters or spaces. In this example I'll call it **level_test**.

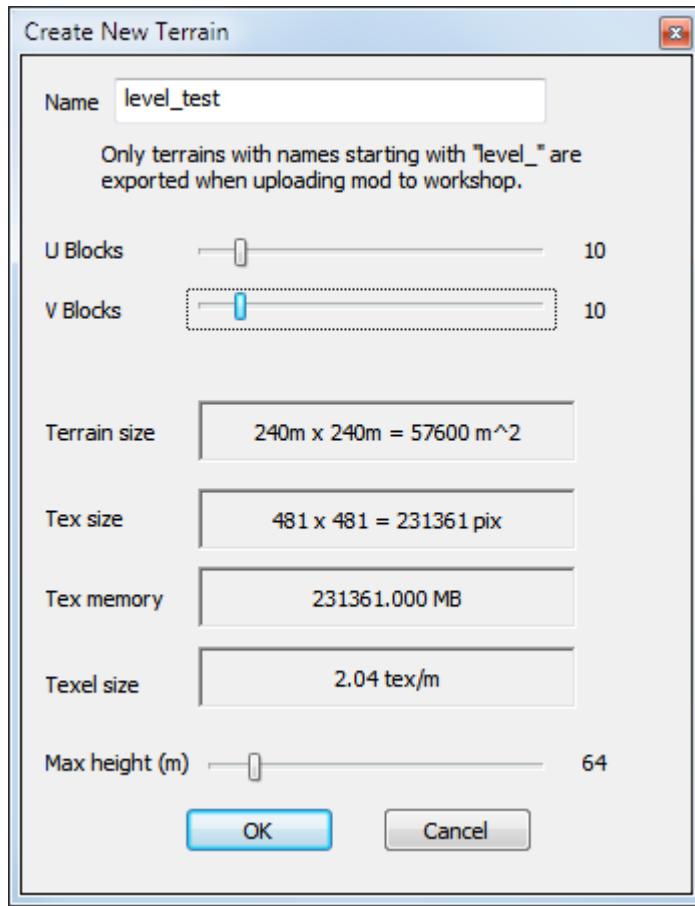
The default is a rather small map 48 meters on a side, so you might want to bump up the sliders for **U Blocks** and **V Blocks** to something like 10 and 10, which equates to 240 meters on a side. The default max height of 64 meters is fine.

Don't go wild with either of these values. A map with large U and V dimensions is very slow to work with, and a large max height causes stair-stepping of the terrain.

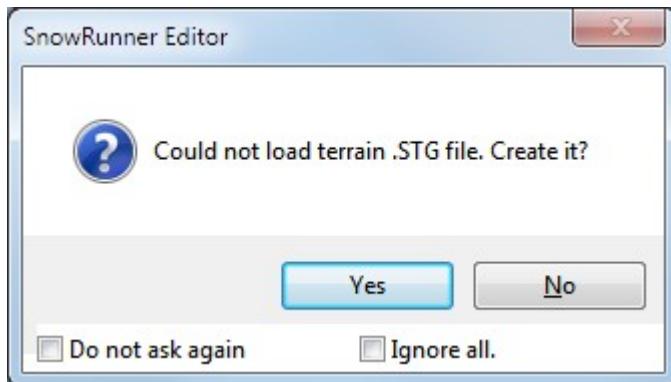
The U and V dimensions must be a multiple of 2. The max height can be any integer. For fine control of the value, you can use the left and right arrow keys after clicking a slider.

Bug: I don't know what the **Tex memory** value is supposed to be measuring, but SnowRunner doesn't actually require 1 megabyte per texture pixel. So don't worry if this value seems excessively large.

When you're ready, click **OK**.

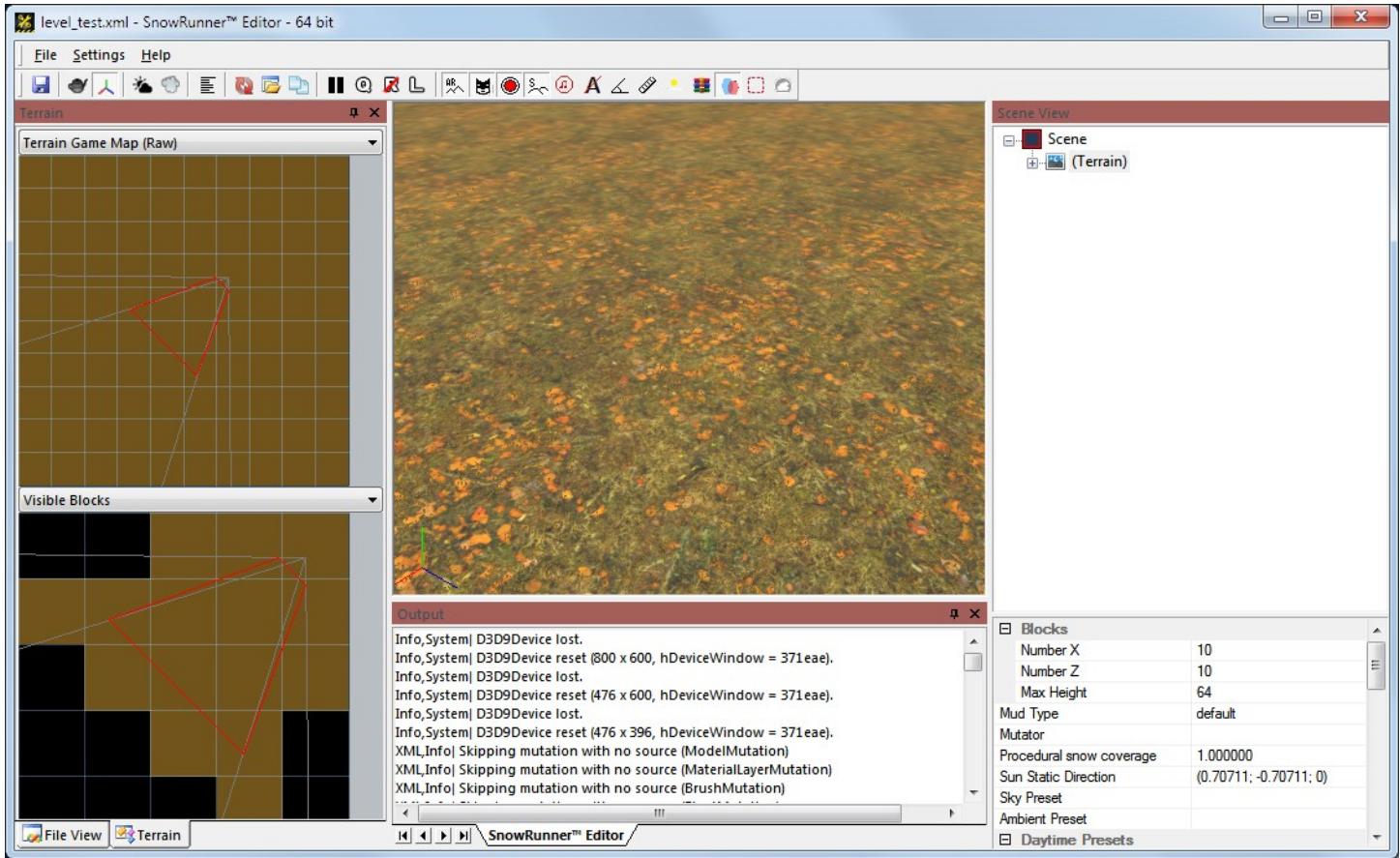


Because this is a new map, it does not yet have a terrain .STG file. Click Yes to create one.



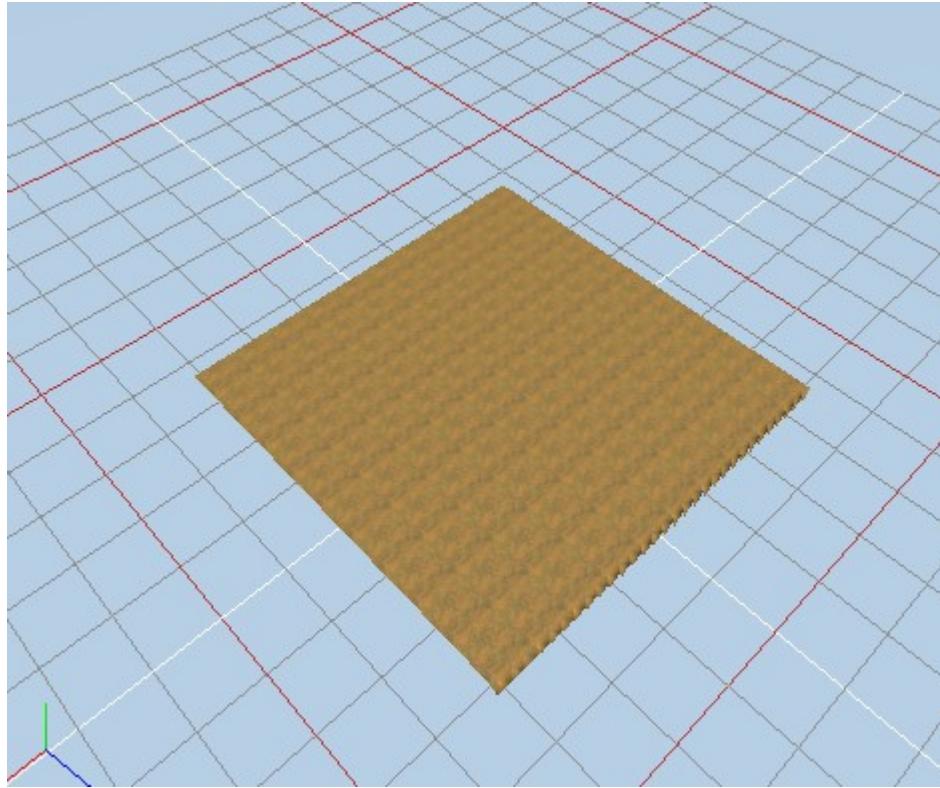
Bug: [New Terrain...](#) sometimes fails to create certain initial files such as [_snow.tga](#) or [mud](#). If this happens, click away the warning dialogs, close the map, and try again with a different map name. Since we're trying to get you started quickly, don't worry for now about deleting the files left behind by the broken map.

Bug: Your first time running the Editor, the camera is pointed in completely the wrong direction to see your terrain. To point the camera in the correct direction, double-click in the **Terrain Game Map** panel on the left. The main (center) panel now shows you a close-up look at the ground.



Move the Camera

Click in the main (center) panel, then rotate the mouse scroll wheel up and down to zoom in and out. When you zoom out far enough, you can see the edges of the map.



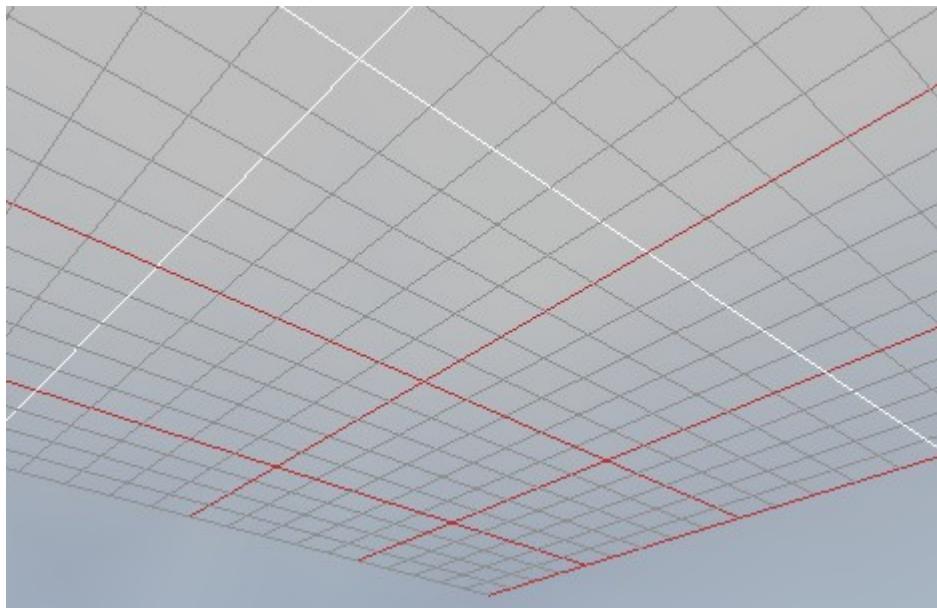
When you zoom in, the camera moves toward where the mouse is pointing. So an easy way to move the camera around the map is to zoom out, point the mouse at the area of interest, and zoom back in.

If you lose track of your terrain, double-click in the **Terrain** panel on the left to restore the camera to a reasonable location.

You can point the camera in different directions by holding down the left mouse button in the main panel and moving the mouse. The camera pivots around the point that you were pointing at when you clicked the button, moving left, right, up, and down.

It is possible to move the camera below the ground. You're not expected to look at the terrain from underneath, and the terrain is only drawn from the top side. So when the camera is underground, all you see is the reference grid above. Rotate the camera back above the ground to restore your vision.

When the camera is pointed up, the background is shaded cloud gray. When the camera is pointed down as is usual, the background is shaded a pale blue.

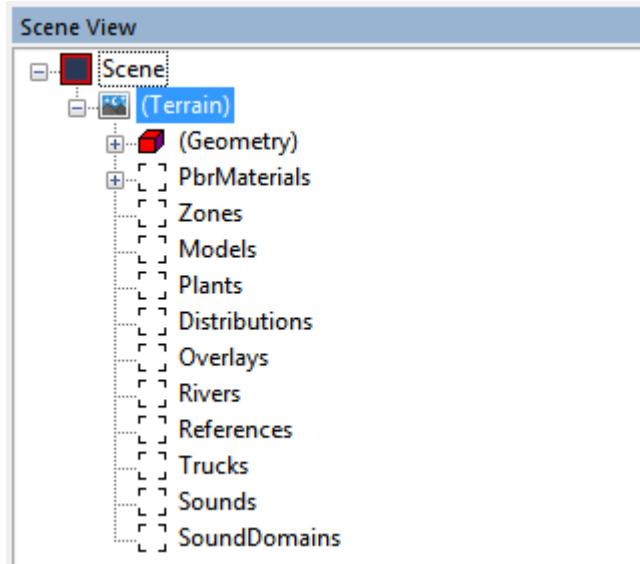


Bug: While moving the mouse to rotate the camera, the mouse pointer also moves. If the mouse pointer moves outside of the main panel, camera rotation stops.

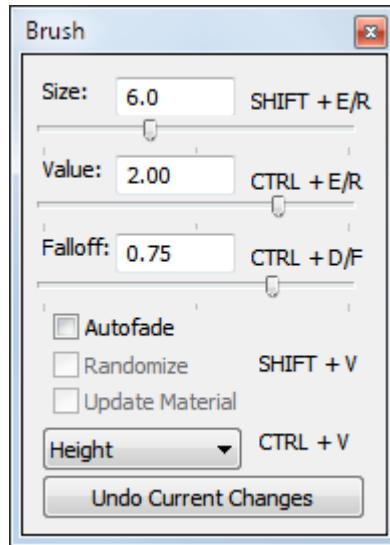
Make Hills and Depressions

The initial terrain is perfectly flat, which isn't very interesting. You can raise the terrain in some places and lower it in other places to make it more interesting to drive around in.

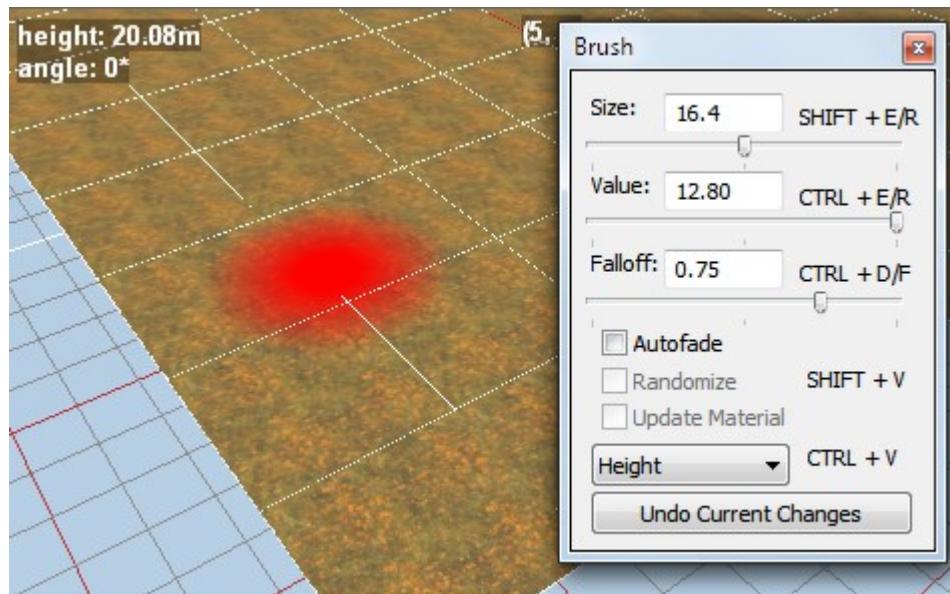
To edit the terrain height, first expand the hierarchy under [Scene → \(Terrain\)](#) to show the many editable aspects of your map, including its [\(Geometry\)](#). There is additional hierarchy within [\(Geometry\)](#), but you don't need to look at that yet.



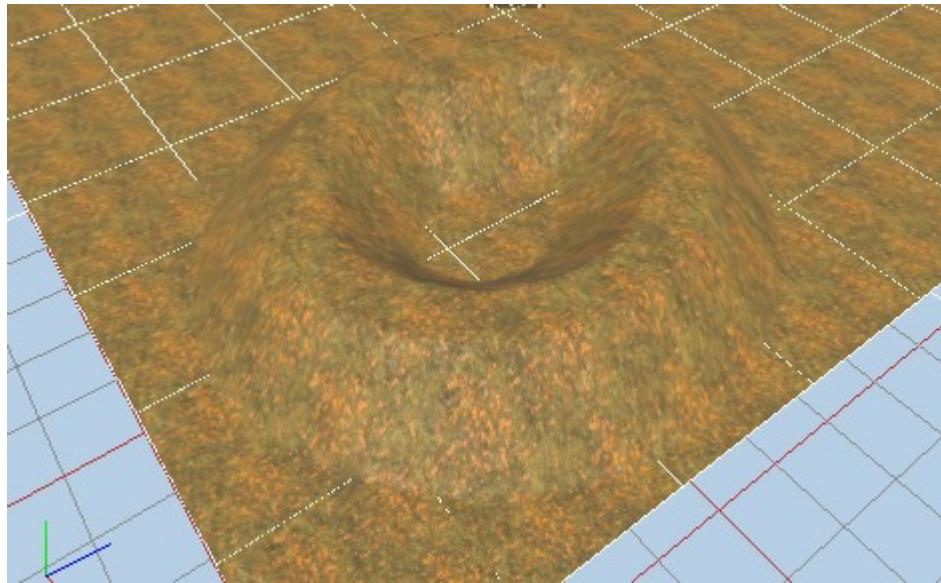
Left click on the **(Geometry)** element in the **Scene View**. This pops up a **Brush** dialog box. It's called a "brush" because you "paint" your terrain height changes onto the map. You can move the **Brush** dialog box to the side if you don't want it to obscure the main panel.



By default, your paintbrush shows up as a red dot on the terrain. Below I've increased the size and value of the brush to make it stand out more.



When the brush is active, you can zoom the camera in and out with the scroll wheel and rotate the camera with the left mouse button as usual. In addition, you can now adjust the terrain height with the right mouse button. With the mouse over the terrain in the main panel, hold down the right mouse button while moving the mouse. The terrain increases in height by the amount of the brush **Value** if the value is positive, and it decreases in height if the value is negative.



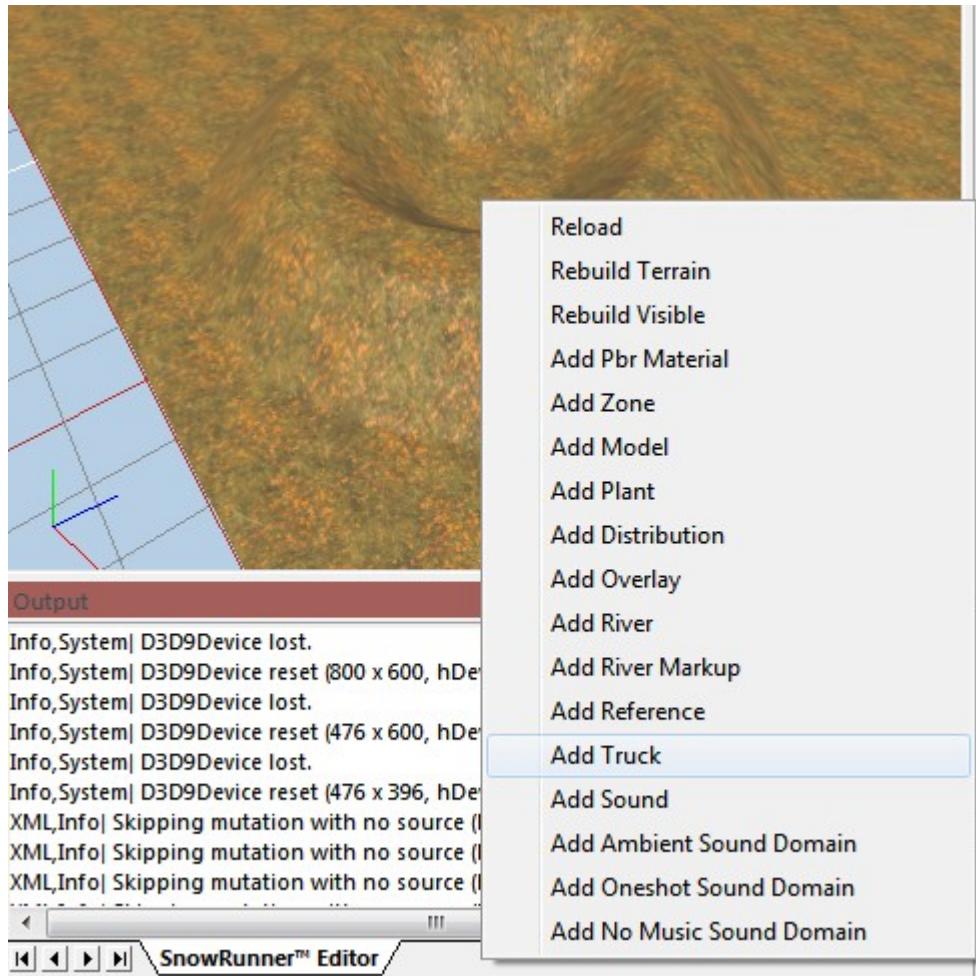
To make larger changes to the terrain, you can adjust the size and value of the brush, or you can make multiple passes with right click and drag.

When you are done editing the terrain height, left click in the main panel to exit painting mode. Note that **left click and drag** moves the camera while remaining in painting mode, while **left click with no mouse movement** leaves painting mode.

Place a Truck

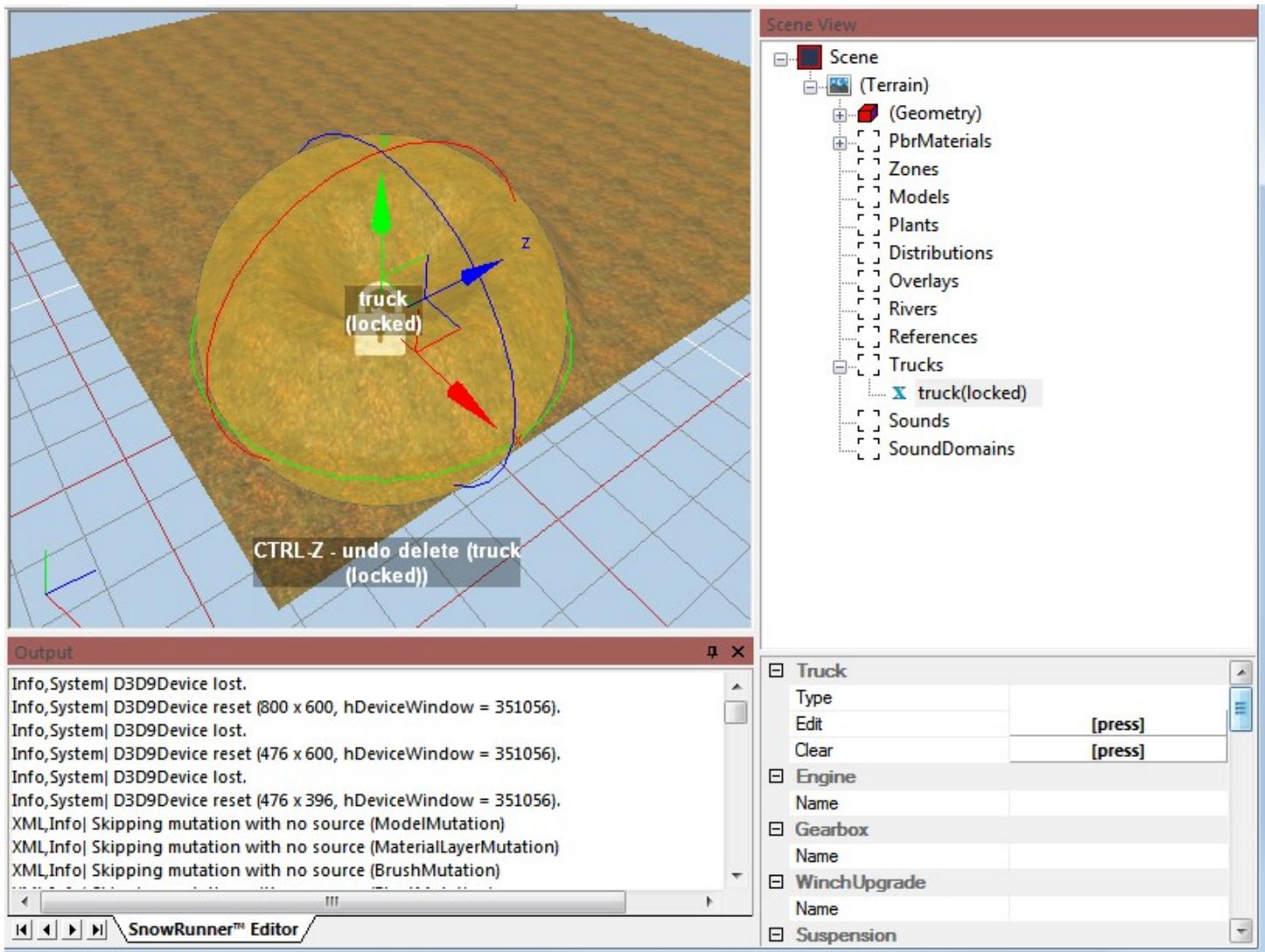
Now that you have some interesting terrain, you'll need a truck to drive around in it.

If necessary, left click to exit painting mode. Then point the mouse over the terrain in the main panel, right click, and select Add Truck from the context menu.

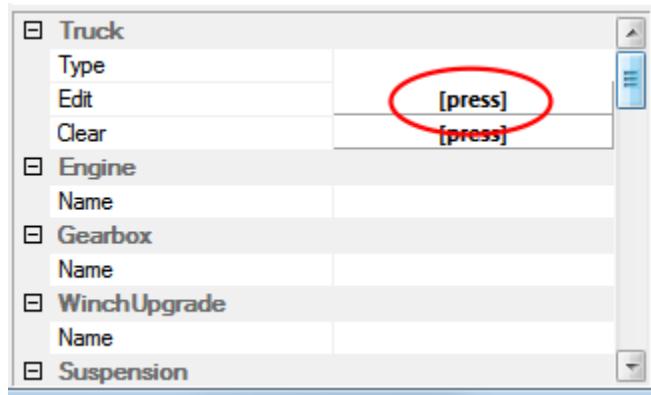


Bug: Unlike in most other programs, the context menu does not appear until you release the right mouse button.

A new truck appears in both the main panel and in the **Scene View** panel. Below the **Scene View** panel is an unlabeled panel that shows various parameters for the truck. I call this panel the property panel.



The truck that you've added doesn't yet have a type. To tell the Editor which kind of truck it is, click in the property panel next to **Truck → Edit** where it says **[press]**.



Bug: The first time you click here, the Editor may lock up for 5 or more seconds as it creates icons for all possible trucks. It will be faster for subsequent uses.

The Editor pops up a dialog listing all of the available kinds of trucks, as well as other things (such as trailers) that the game puts in the truck category.

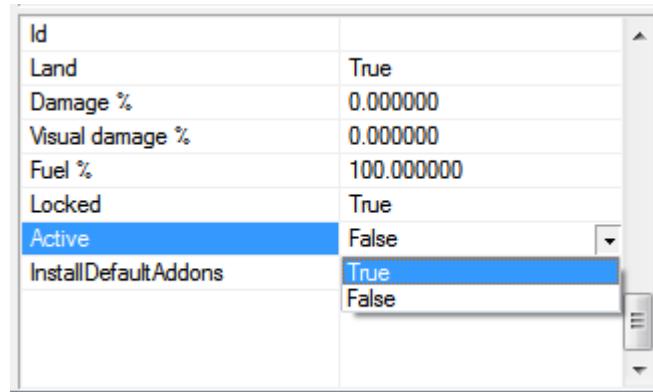


For testing purposes, I like the **ank_mk38** because it is a very capable truck in its base form, and because I don't have to scroll to find it. Click to select the truck that you like and click **OK**.

If you have experience with the MudRunner Editor, you may remember that changed feature properties did not commit until you changed the selection (e.g. by clicking elsewhere). However, the SnowRunner Editor mostly fixes this bug so that property changes take effect immediately. The few exceptions where the commit is delayed are described where appropriate.



There is one more step to make this truck drivable. If necessary, re-select the truck by clicking it in the **Scene View** panel. Then scroll to the bottom of the property panel. Where it says **Active: False**, click on **False**, and change the dropdown value to **True**.

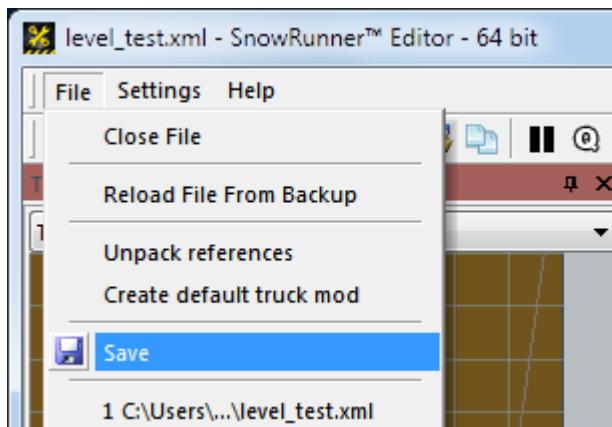


The truck is now active.



Save and Pack

To save your work so far, select **File → Save** from the top menu bar or press the keyboard shortcut: **Ctrl+S**.



Now pack your map for testing. Click the stack of books in the toolbar. The toolbar is just under the top menu bar, and the stack of books is the fourth icon from the right end.



A dialog box warns that packing the level takes time. Click **Yes** to continue and wait for it to complete. For a small and simple map, it takes only a few seconds.

Packing your map also automatically saves it. So the previous save step wasn't strictly necessary, but saving often is a good habit to get into.

Make a Backup Copy of your Campaign

In the course of working with the Editor, you will start a lot of new games on your maps. SnowRunner gives you four save slots, so hopefully you have room to create these new games without disturbing the save files for your campaign, but there is always the possibility that you make a mistake.

Bug: When I leave SnowRunner idling on the main menu in the background, it will occasionally generate spurious events as if I am pressing the **Enter** key or other keys. This may cause it to start a new game and overwrite a save slot! I always leave a slot empty so that it is the default location of a new save. This seems to reduce the frequency of disaster, but it's not a failsafe. Even with my precautions, the game once deleted all of my saves while idling in the background.

To avoid losing your campaign progress, I recommend making a copy of your data.

If you bought SnowRunner from the EPIC Store, your save files are in **%USERPROFILE%/Documents/My Games/SnowRunner/base/storage**. You might know **%USERPROFILE%/Documents** as your “My Documents” folder. That directory has a randomly numbered subdirectory with your save data in it.

If you bought SnowRunner from Steam, your save files are in a path something like **C:/Program Files (x86)/Steam/userdata**. That directory has a randomly numbered subdirectory. This subdirectory contains directories for each of your Steam games which are also unhelpfully labeled with random strings of digits. If you’ve played SnowRunner recently, then its folder should be one of the most recently modified. Your save data consists of the **remote** directory and associated **remotecache.vdf** file.

Make a backup copy of all files in this directory. If you ever need to restore your files, the steps are on page 38.

Test Your Map

Time to take your map for a spin. Start the SnowRunner game.

Tip: You have to wait through the epilepsy warning every time, but you can skip the rest of the ads by pressing the **Esc** key as soon as the epilepsy warning starts to fade out.

From the game’s main menu, select **New Game** and then **Custom Scenarios**.



The game lists all available mods. Most likely this is only the one you created, e.g. [mod_level_test](#). Select your map and a save slot for your custom game. The save slot must be separate from your main campaign's save slot because progress for each mod is saved separately.

Bug: The game does not reliably replace an existing save with a new game. It asks for confirmation as if it is replacing the existing save, but then it sometimes loads the old game instead of the new game. This bug only occurs when a new game is the first game loaded after SnowRunner is started. Exit the (faulty) game and start a new custom scenario again, and it works. This may be a Steam integration bug, in which case perhaps Epic users are safe from it.

Bug: I often leave SnowRunner running in the background as I repeatedly make edits to a map and test it. That means I don't have to wait for the game to start up, **and** it works properly when I create a new game. If your PC is limited on memory, however, you may not be able to run the game and the Editor simultaneously.

The game now starts in the truck that you chose and in the terrain that you painted.



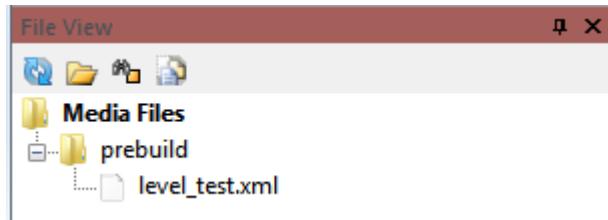
The game controls are all as usual, with one exception. There is now a menu of **Tools** in the upper right of the screen. The **Tools** menu provides actions useful for testing such as spawning new trucks, allowing truck modification as if you are in a garage, etc. See the Dev Tools section on page 39 for more information about this menu.

Bug: The default grass and leaves texture on the terrain looks unnaturally large compared to the size of your truck. That can be fixed once you are familiar with the material layers (page 156).

Re-editing a Map

If you quit and restart the SnowRunner Editor, it does not automatically return to the map you were previously editing.

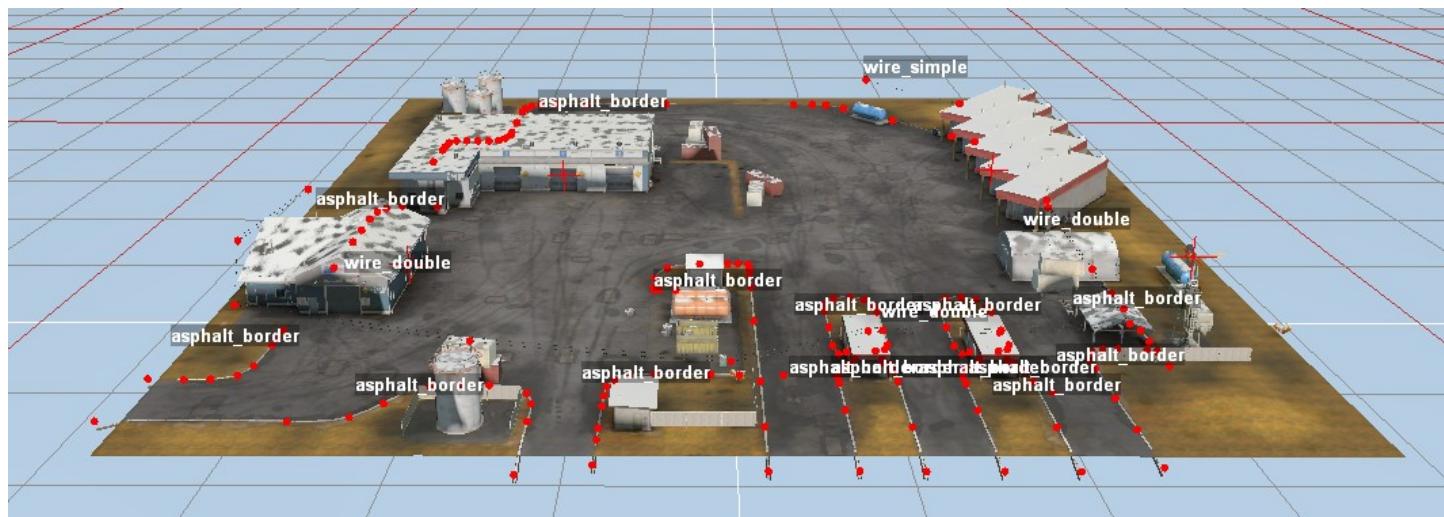
The maps that you have created are listed in the [File View](#) panel under [Media Files → prebuild](#). To edit a map, double click it.



Example Maps

Saber has not published any of their maps in an editable format. However, their maps use a number of shared map sections that Saber has kindly made available for modders to use. [File → Unpack references](#) unpacks these map sections from the installation directory and puts them in the [Media/prebuild](#) directory where they can be read into the SnowRunner Editor.

Bug: [Unpack references](#) takes a very long time (over a minute on my system) with no indication of progress. Be patient, or watch the files as they appear in the [Media/prebuild](#) directory.



Introduction

Before diving into all of the details of creating a map, the following sections create a base of knowledge that you can use for all editing tasks.

How to Use This Book

This book is designed so that you can read it straight through from beginning to end. Each section builds on the information from previous sections to form a complete picture without too many instances of “I’ll tell you later”. However, you can also just start experimenting with the Editor while reading only the sections of this book that strike your interest. To accommodate this, I’ve done my best to label all sections so that you can easily find the section you need in the table of contents.

If you’ve downloaded the PDF, all page numbers are hyperlinks. Click on the page number to jump directly to it.

What Kinds of Maps Can I Make?

Maps can be as small or as large as you like, even over than 2 km square. (Believe me, that’s plenty large.) You can also combine up to four maps into a global region. Unfortunately, there isn’t a known method for making a multi-region campaign similar to the one built into SnowRunner.

I recommend that you start small, though. If you’ve played any of the built-in campaign regions, you know how long they took to play. Design takes much longer, and you’ll want to test every aspect of your map at least once, plus at least once all the way through.

Each map or region that you publish stands alone and cannot share progress or saved games with the built-in campaign or other mods.

Custom maps can be played single-player or multi-player.

Custom maps can only be run in normal mode. The game does not allow custom maps to be started in hard mode.

Maps can use almost all assets that are used in the built-in campaign’s maps and DLC, including models, trees, roads, terrain materials, etc. The only limitation is that trucks from DLC can only be used by players who own the same DLC.

If you’re feeling ambitious, you can also create your own 2-D, 3-D, and sound assets. The SnowRunner map editor allows you to integrate custom assets into your maps, but you’re on your own for the tools necessary to create them. You can even create your own trucks and trailers to use on your map, but custom vehicles are not covered by this guide.

To reach an international audience, you can localize the text in your maps into many languages.

Maps can easily be published on [mod.io](#), the official exchange for custom SnowRunner maps and trucks.

Do you have ideas for how a map should be designed? For challenges that you haven't seen before? For vistas that beg to be seen? If so, make a map!

Sources of Information

I should point out that this is not your only source of information regarding SnowRunner map making. There are various other guides on the web in either text or video form, and Saber have themselves published a reasonably complete guide. You can find Saber's guide by selecting **Help → Guides** in the Editor. This opens a window with all of the built-in SnowRunner guides. Saber's map editing guide is in [SnowRunner_Editor_Guide.pdf](#).

Advantages of the Big Book of SnowRunner Map Making:

- Faster “quick start” to editing and testing.
- Includes topics not included in the Saber guide.
- Covers all topics in exhaustive detail.
- Acknowledges bugs and describes how to avoid them.

Advantages of the Saber guide:

- More likely to stay up to date as the game and map editor are updated.
- Not as exhausting to read.

By the way, if you think this book is useful, please give it a “thumbs up” wherever you found it. Those thumbs up help more people find and enjoy this book. More educated modders means better mods!

Is This Book Up To Date?

There are a lot of websites that will “helpfully” copy this book for their readers, but won’t keep it up to date. The official release location is on Google Drive:

https://drive.google.com/file/d/1x_CtAODJ8bd9d5wilTKIZIzIYaE8C-JM/view?usp=sharing

The title page of this book has the date on which I last edited it. However, Saber doesn’t notify me when they change something, so it’s quite possible for me to edit the book after Saber makes a change and yet not include the change. I keep a change log for my book updates on page 460.

I’ll add comments to my [announcement post](#) in the Focus forums when I make major updates. If you notice anything missing or wrong, let me know in the comments.

Literals and Variables

Text in the Editor (or game) that this book quotes literally is formatted as follows: **example literal text**. It is typically used for file names, feature hierarchy, property names, and built-in property values.

Variable text is text in the Editor (or game) that can have different values, which this book formats as follows: **example variable text**. I'll usually use variable text that both describes the value in question and also conforms to the style that the text must obey.

For example, when this book refers to your map's XML file, the map name is whatever you assigned (which should be lowercase and without spaces), but the file extension is always the same. Thus, the map's XML file is **map_name.xml**. Note that the period after **.xml** is not highlighted because it is not part of the filename.

Editor and Game Bugs

No mod editor has ever been designed to be as consumer friendly as the game itself for a number of reasons:

- Only a fraction of game buyers even try mod editing.
- People who edit mods tend to be more tech savvy and better able to work around any bugs that exist.
- The universe of possibilities is much larger when editing mods than when playing a game. That makes it harder to test every corner of the design and iron out the bugs.

In most cases, a mod editor is tested only to the extent that it was used to make the game levels themselves. If a feature wasn't used in the game, then it probably wasn't tested in the Editor.

If it sounds like I'm making excuses for Saber, I totally am. I have complete respect for Pavel's original Editor and for all of the quirks that Saber has smoothed out of it. But Saber has also added a pile of new features which come with their own bugs.

For any insufficiently tested feature, there can be a bug in the Editor (so the Editor can't edit that feature properly) or a bug in the game (so that the game doesn't work correctly with that feature). I've chosen to highlight either type of bug as follows:

Bug: This is an example bug warning. I'll explain what triggers the bug, how to recognize it, how to avoid it (if possible), and how to fix any problems it causes (if possible).

I generally put each bug warning immediately after the related text that describes how to use a feature. So if you see a bug warning, read it!

And if Saber wants to use these bug reports while updating the Editor, I wouldn't mind that, either. :)

I have encountered many more bugs in the Editor than are documented here. However, many bugs occur only occasionally without any known cause, which makes them difficult to document. If the Editor is behaving oddly, try restarting it to see if that fixes the problem.

Broken Assets

SnowRunner comes with a **lot** of assets, i.e. specific 2-D, 3-D, or sound features that you can build into your map. Any assets that Saber used in their own maps are generally fairly well debugged, as are trucks, trailers, and add-ons since the player can add them to any map. However, Saber also created some map assets that they didn't end up using, and these are less well debugged. I suspect that some experimental assets were even released accidentally, and these may be completely broken.

Since someone might choose to use a released asset in their map, potential warts and all, Saber is understandably reluctant to change any assets once released. Thus, buggy assets tend to accumulate, with fixed versions being released under a different name (if they are fixed at all).

So if you find a buggy asset, check whether there is an equivalent fixed asset. If not, and if the bugs can't be worked around, just write it off and don't use that asset. So it goes.

Design Tips

I'll admit right now that I am not the best map designer. I'm an engineer at heart, and my artistic eye is limited. Therefore, I'm hesitant to give a lot of design tips in this book. Where I do give a tip, it is often highlighted as follows:

Tip: This is an example design tip. You don't have to do it this way, but this is probably a good way to do it.

Some of these tips I've directly borrowed and expanded from Saber's guide. Other tips come from my experience with making maps and/or wrestling with the Editor for long periods of time. I try to stay away from totally obvious tips. You already know what a natural world looks like, so I'll leave it up to you to decide whether rocks should go in trees.

If You Made Maps in MudRunner

If you made maps in MudRunner, then the Editor should be mostly familiar to you. I recommend reading through the Quick Start and Useful Files and Archives sections since those highlight some important differences from MudRunner.

In particular if you already read my Big Book of MudRunner Map Making, you'll find a lot of familiar material in this book. On the other hand, a lot of things have changed from MudRunner, and a lot of things are new. If you have the patience for it, I recommend at least a skim through the entire book to avoid frustrations later.

The MudRunner Editor included the ability to import maps from SpinTires. However, the SnowRunner Editor doesn't include any ability to import maps from SpinTires or MudRunner. You may be able to import selected

features of previous maps by manually copying bitmaps and XML sections. However, enough has changed that you're unlikely to have much success. Sorry.

Useful Files and Archives

It is useful to understand the directory structure used by the SnowRunner Editor since you need to occasionally interact with it. And you should keep backups of your map source files, just in case.

Media Directory

The primary files and directories of interest are in

`%HOMEPATH%/Documents/My Games/SnowRunner/Media`.

You might want to make a shortcut from your desktop for easy clicking.

Map Source Files

The files that describe your map are saved as follows:

- `prebuild/map_name.xml` – the main XML file and the point of reference for all other files.
- `prebuild/map_name/*` – additional XML files, bitmap files, and other files that describe portions of the map.
- `levels/map_name/*.json` – these files are created when you first use the Zone Settings Editor (page 276).

The format and contents for these files is described in the appendices, beginning on page 420. However, for the most part you don't need to worry about it since the Editor handles editing of these files automatically.

One useful thing to understand is when data is saved to disk:

- `*.xml` files are saved when you explicitly save in the main SnowRunner Editor.
- `*.json` files are saved when you explicitly save in the Zone Settings Editor or the Region Settings Editor.
- Bitmap files are saved as soon as you finish painting a feature in the Editor. I.e. these files always record your most recent changes without waiting for you to save.

Because your map data is saved in your user area and not the game area, it remains safe even if you uninstall SnowRunner. However, there is always the possibility of losing or corrupting your files either through user error or an Editor bug. For this reason, **keeping archives of your work is important**.

The easiest archival method is to occasionally copy the necessary files and directories from the above locations to another directory. Even better, copy it to a different drive in case you have a drive failure. Keep in mind that the Editor may add more files to the `prebuild/map_name/` directory or its subdirectories from time to time, so make sure to back up all of these files and subdirectories, not just those files that were there at the start.

Derived Map Files

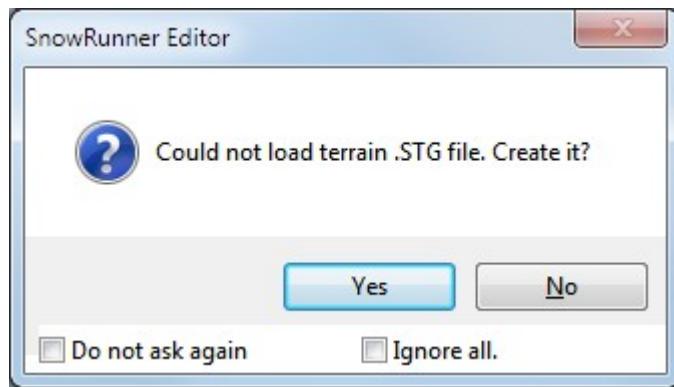
The Editor maintains other files that are derived from the source files above. These derived files are generally optimized for display or gameplay purposes, whereas the source files are optimized for easy editing. The derived files are saved as follows:

- `levels/map_name/data.stg` – a giant binary blob used by the rendering engine
- `levels/map_name/*.dds` – various derived bitmap files
- `levels/map_name/user_name_of_map.txt` – an embarrassing hack that no one likes to think about

Note that these files are mixed together with the `*.json` files that are part of your map **source** files.

The derived files are saved when you explicitly save in the main SnowRunner Editor.

If you delete any of these derived files (but not the `*.json` files!), the Editor offers to recreate them from your map source files when you load the map. When the Editor makes the offer, it generically refers to all of these derived files as the “.STG file”.



Game Files

When you pack your map, the Editor further optimizes and compresses the files needed by the game. These are the files that the game loads when you test your map. They are saved as follows:

- `Mods/map_name*.pak`

Each `*.pak` file is an archive of files that can be opened with a full-featured archive manager such as 7-Zip. However, the contents of these archives are mostly opaque since vital information is saved in a format that only SnowRunner can read.

The SnowRunner Editor recreates these files whenever you pack your map.

Upload Files

When you pack your map, the Editor also prepares a single file suitable for upload to mod.io, named as follows:

- `levels/map_name.zip`

This `map_name.zip` file is an archive of the `*.pak` files described above. (Yes, an archive of archives, in competing formats no less.) Having a single file makes the upload easier.

The SnowRunner Editor recreates these files whenever you pack your map.

Asset Icons

The Editor keeps a cache of icons showing the appearance of the available map assets as follows:

- `icons/*.dds`

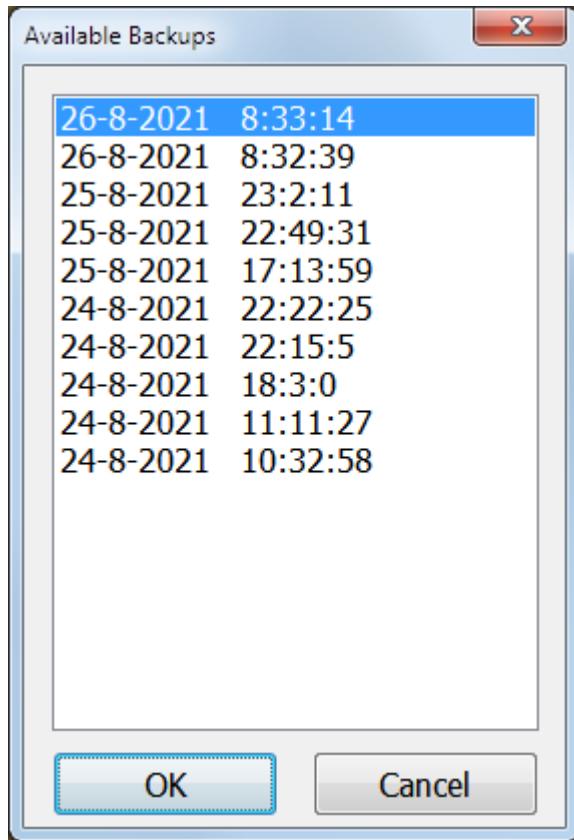
Asset selection is explained on page 59.

Limited Backups

The Editor automatically keeps backups of your XML files **only** (not JSON or bitmap files). Backups are kept in the following location for the most recent 10 times that you've had the map open for editing:

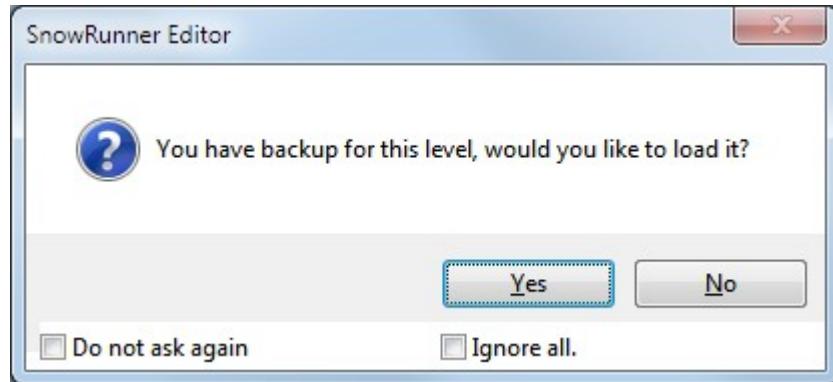
- `%appdata%/SnowRunner 2 Editor/backup/map_name/*`

You don't need to memorize this directory location, however, because the Editor can automatically restore your choice of backup files when you select [File → Reload File From Backup](#).



Bug: When you restore from backup, this creates unsaved changes that get their own redundant backup, which causes the oldest backup to expire. If you're not sure which backup you want to restore, you might want to backup your backups before you start experimenting with restoration.

Of note, the most recent backup always has your most recent changes, even if you haven't saved them yet! So restoring from the most recent backup usually won't have any effect. However, if the Editor crashes while you are editing a map, it offers to restore your most recent backup the next time you load that map into the Editor.



If you click **Yes**, your most recent unsaved edits are restored, but still aren't saved. It is your decision whether or not to save them. Note that your camera position is not restored, so it may deceptively point at an unchanged portion of your map. Move the camera to where your latest changes were, and you should see them.

If you click **No**, your unsaved edits are permanently discarded. So if in doubt, click **Yes** and review your edits before deciding whether to keep or discard them.

To summarize, the Editor's automatic backups essentially provide two functions:

- The most recent backup has your most recent edits, saved or unsaved, which means that it's in sync with your latest bitmap files. When the Editor crashes, this backup is fantastically useful.
- Older backups are not in sync with your latest bitmap files, so they're less useful, but they still have value in limited circumstances.

Editor Configuration Files

The Editor keeps track of its past activity and settings in the following files:

- `%appdata%/SnowRunner 2 Editor/LastStates.xml` – used to detect when the Editor crashes (because the file isn't updated as expected before the Editor closes).
- `%appdata%/SnowRunner 2 Editor/CustomColor.xml` – records custom colors that were set in the color picker, e.g. in the brush dialog for **(Geometry) → (Colorization)**.
- `%appdata%/SnowRunner 2 Editor/ViewTerrain.xml` – keeps for each map the most recent Editor camera position.
- `%appdata%/SnowRunner 2 Editor/Log.txt` – keeps the text from the **Output** panel for all sessions (so this file eventually gets quite large).
- `%HOMEPATH%/Documents/My Games/SnowRunner/base/config/editor.cfg` – records the current settings for **Show Snow By Up Vector** (page 173) and for many of the toggle buttons on the toolbar (page 256).

Game Save Data

If you bought SnowRunner from the EPIC Store, your save files are in `%USERPROFILE%/Documents/My Games/SnowRunner/base/storage`. You might know `%USERPROFILE%/Documents` as your "My Documents" folder. That directory has a randomly numbered subdirectory with your save data in it.

If you bought SnowRunner from Steam, your save files are in a path something like `C:/Program Files (x86)/Steam/userdata`. That directory has a randomly numbered subdirectory. This subdirectory contains directories for each of your Steam games which are also unhelpfully labeled with random

strings of digits. If you've played SnowRunner recently, then its folder should be one of the most recently modified. Your save data consists of the `remote` directory and associated `remotecache.vdf` file.

Make a backup copy of all files in this directory. If you ever need to restore some files from backup, you can either restore them all, overwriting all four save slots, or you can pick and choose. But you definitely want to archive all of the files now so that you don't discover later that you missed an important one.

Restoring Game Save Data

The easiest way to restore your game save data is to delete the current save data and copy your archived files in its place. Quit the game first to ensure that it doesn't still have old data in memory.

If you are on Steam, be sure to restore the `remotecache.vdf` file as well. It may be necessary to disable cloud saves to ensure that Steam reads the local data.

After restoring your archived save data and restarting the game, you might find that your objective progress is correct, but all of your maps are cloaked. This may be a problem particular to Steam. If you have this problem, these two comments may help you solve it:

<https://steamcommunity.com/app/1465360/discussions/0/4811511685076364199/#c4811511685077529182>

<https://steamcommunity.com/app/1465360/discussions/0/4811511685077534336/#c3111403360718313694>

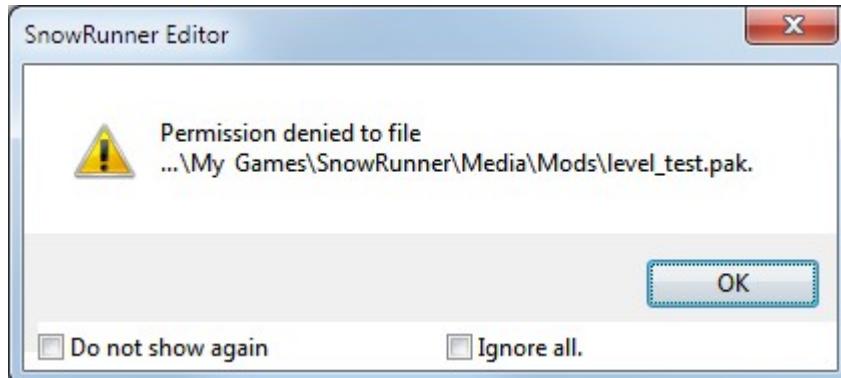
Test Your Map

I expect that you're already familiar with how to play SnowRunner. However, the information in this section details tools that aren't normally available or that you might not have bothered with during regular gameplay. Understanding these tools is absolutely necessary in order to effectively design and test your map.

Pack Your Map

Before testing any changes to your map, you must pack your map. Click the **Pack terrain** button in the toolbar to pack your map.

When the game has your map loaded, it locks your map to prevent changes. If you attempt to pack your map while it is locked by the game, the Editor will display a failure message. Return to the game's main menu to unlock the map and allow the Editor to make changes.



Likewise, if the Editor is packing a map, it locks the map to prevent it from being played. If you attempt to load the map in the game while the map is locked, the game will exit back to the main menu.

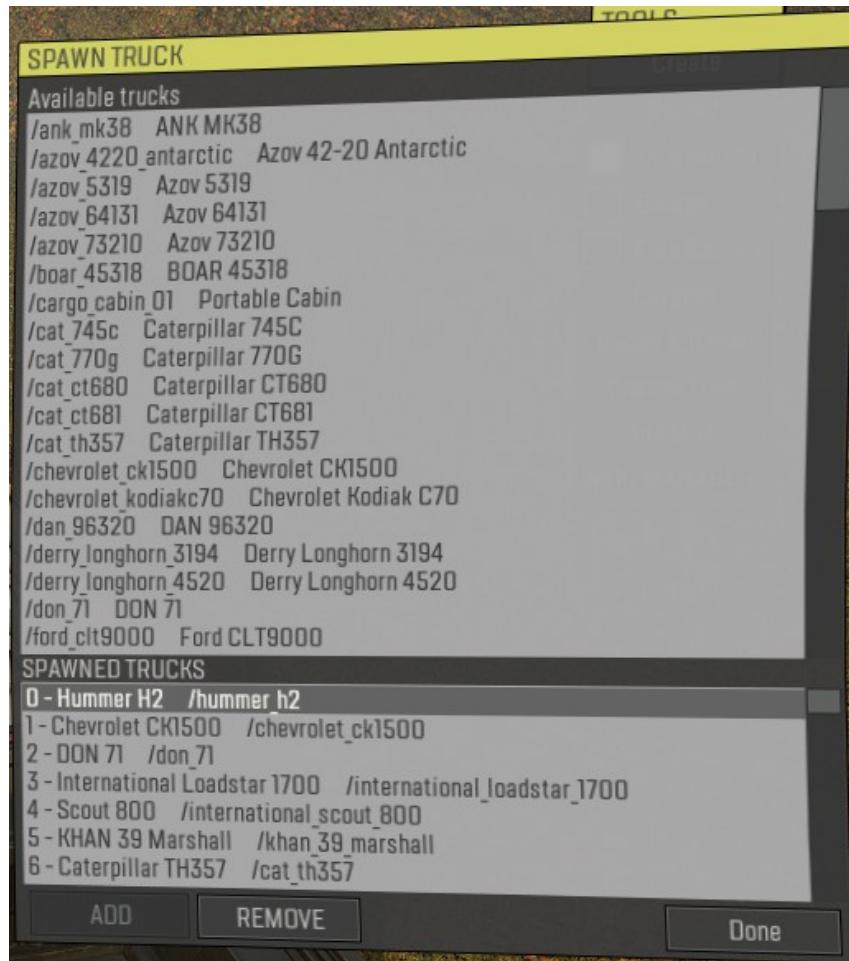
Dev Tools

While running the game in your local custom map, a **Tools** menu is shown. This menu of development tools (abbreviated as “Dev Tools”) gives you the ability to quickly try different combinations of trucks, trailers, and add-ons.

The various options in the **Tools** menu are described below.

Dev Tools: Create

Clicking **Create** pops up a dialog that allows you to create a truck and/or jump to any existing truck.



To add a truck, click on any truck under **Available trucks**, then click somewhere on the visible terrain. A ghost image of the selected truck appears. Rotate the truck with left click and drag. When you are satisfied with its location, click **ADD**.

Warning: the created truck may intersect hazards on the map such as models or other trucks. As soon as you get close enough to enable its physics, the truck may take (or cause) significant damage.

To remove the truck currently being driven, click **REMOVE**.

Warning: if you close the **Create** dialog while not in control of a truck, you won't be able to drive any truck or even move the camera. However, you can still use the **Tools** menu to jump into another truck.

To jump into another truck from the **Create** dialog, click any truck listed under **SPAWNED TRUCKS**.

Bug: While stranded outside of a truck, the navigation map won't list any trucks, so you can't jump to a truck via the navigation map.

Dev Tools: Reload

The **Reload** option is not particularly useful to map development. It assists truck modders by reloading truck assets without leaving the map. It is not possible to change a packed map while it is being used by the game.

Dev Tools: Information

The **Information** checkbox puts some velocity information in the center of the HUD. Again, this is useful only to truck modders.

Dev Tools: Virtual Garage

The **Garage** checkbox pops up additional menus that allow you to customize the current truck with add-ons and trailers. This virtual garage allows any compatible part to be installed regardless of whether you unlocked the part or reached the necessary rank.

Bug: You can open the navigation map while in the virtual garage, but it does not give you the usual options to view objectives or change trucks. Close **Garage** mode and re-open the map to get the usual map options.

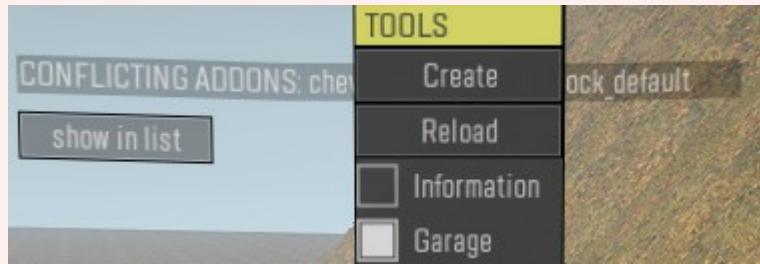
All parts in the virtual garage are listed by the lowercase name used by the Editor. Thus, the **Garage** option is extremely useful for getting the names of compatible parts to add to trucks in the Editor.

The virtual garage only lists trailers that can potentially be bought at the trailer store. You cannot use the virtual garage to attach specialty trailers that are typically only used for objectives.

To install or uninstall an item, double-click it.

If there is an installation conflict, a message appears on the screen along with a button, **show in list**. Clicking **show in list** moves the part selection to the conflicting part (or one of the conflicting parts if there are multiple). The conflicting part may need to be uninstalled (to make room) or installed (to provide an attachment point, such as for a semi-trailer). Double-click the conflicting part to install/uninstall it before re-attempting your original change.

Bug: The message describing the installation conflict is obscured by the **Tools** menu. Therefore, you'll probably need to click the **show in list** button to find the (first) conflicting item.



Bug: The virtual garage doesn't always detect when a combination of add-ons conflicts with an existing trailer. But it does detect when a new trailer conflicts with existing add-ons. So always add the trailer last.

If your truck is too close to another object, installing a part may fail because there isn't enough room to install it without causing a collision. In this case, the part is not installed. This can be distinguished from a conflict with another part by the lack of a conflict message.

Tires, Rims, and Suspension

Tires and rims must match. To make this easier, installing any tire automatically installs a compatible rim if necessary. This helping hand only goes one way, though. If you attempt to install a rim that is incompatible with the current tire, the install silently fails.

The tire and rim size must also be compatible with the suspension. If you attempt to install a tire size that is incompatible with the suspension, the install silently fails.

Bug: After installing a raised suspension and a large tire and rim, the virtual garage allows you to install a shorter suspension while keeping the oversized tire and rim. Do not attempt to use this incompatible combination in the map Editor.

Dev Tools: Free Camera

Clicking the **Free Camera** checkbox separates the camera from the truck. There is no immediate change to the camera view, but you can now drive away from the camera. Or you can move the camera freely using the following controls:

- right click and drag: rotate the camera up, down, left, or right.
- **I** or **NumPad 8**: move the camera forward.
- **K** or **NumPad 2**: move the camera backward.

- **J** or **NumPad 4**: move the camera left.
- **L** or **NumPad 6**: move the camera right.
- **O**, **NumPad 3**, or **Shift**: move the camera up.
- **U**, **NumPad 1**, or **Z**: move the camera down.

All free camera movements are much faster when the left mouse button is held down. This is quite handy for larger movements, but it is *extremely* fast, so use it with caution.

The free camera is useful for looking at an angle not normally allowed, for getting a closer look at something, or for arranging an artistic screenshot. More tips for screenshots are on page 46.

Dev Tools: Cargoes

Click **Cargoes** in the **Tools** menu adds some information to the upper left showing the usage of cargo points on the truck.

Unfortunately, there is no way to artificially spawn cargo using the **Tools** menu.

Dev Tools: Repair & Refuel

Click **Repair & Refuel** to fully repair and refuel your current truck. The parts and fuel are not taken from any truck; they are gifted to you by the **Tools** menu.



If your truck has flipped, this option does not set it back upright. The quick way to restore a flipped truck is to spawn a new truck of the same type, then deleted the flipped truck.

Dev Tools: Change Time

Click **Change Time** to cycle through the lighting conditions at different times of day. This is useful for testing the various sky presets (page 252).

Bug: Using the **Change Time** option to switch to nighttime does not correctly trigger lights to shine. It also doesn't actually change the time for time-based features such as ambient sound domains. Instead, use the **Skip Time** feature from the navigation map to change the time (page 45).

Dev Tools: Mod Tools

The **Add Mod** and **MOD MANAGER** buttons allow modded trucks to be managed and spawned.

Revealing the Map

The map is initially an unexplored land. As the player drives around and visits watchpoints, more details are revealed in the map view, and vehicles and zones are discovered. By default, the map view is accessed by pressing the **M** key in the game.

Full Reveal

To fully reveal an area, the player must drive a truck near it. Areas within 39 meters of the active truck are drawn in color in the map view. The player can also visit a watchpoint, which reveals the map within the radius set by the watchpoint's property. Vehicles and zones are usually labeled on the map if they are within a fully revealed area.

Viewing an objective's details can also label unrevealed vehicles and zones used by the objective. Zone visibility is described in more detail on page 363. If an objective is being tracked or the objective's details are viewed, and a vehicle is needed for the current stage, that vehicle is always labeled on the map, even if it hasn't yet been revealed.

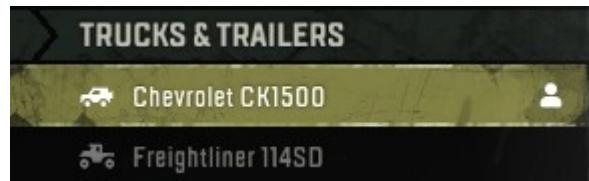
Partial Reveal

If the map is set to be uncloaked (page 292), then the portions of the map that have not been fully revealed are partially revealed. These areas of the map are drawn in gray shades. The shapes of vehicles can be seen in these partially revealed areas, but they are not usually labeled on the map. Zones are also not usually labeled on the map if they are only partially revealed.

If the map is set to be cloaked, then most of the map is initially black. As the player drives around, areas of the map within about 140 meters of the active truck are partially revealed. Visiting a watchtower fully reveals some portion of the map, but it does not partially reveal any additional area.

Discovery of Vehicles

If a truck or trailer is set to be locked (page 89), then even after it is revealed, its label is initially gray in the list of **TRUCKS & TRAILERS** on the left side of the map.



A player can “discover” the vehicle by driving within 15 meters of it (as measured between the truck centers). This discovery rewards the player some experience points, and its label changes to white on the left side of the map. If a locked truck has not yet been discovered, the player can't jump into it, even if the target truck is

clearly visible. Once the player discovers the truck, she can jump into it by using either the map view or the **CHANGE TRUCK** function.



If the vehicle **type** is locked (page 293), discovering it also unlocks the type so that it can be bought at the garage or trailer store. Note that if a vehicle type is locked, but an individual vehicle is unlocked, then that vehicle cannot be discovered, and it can't be used to unlock its type.

If a truck or trailer is set to be unlocked, then its label is white on the left side of the map as soon as it is revealed. There is no separate “discovery”, and nothing new happens when the player drives near it. As soon as an unlocked truck is revealed, the player can jump into it from the map view. Alternatively, the player can use the **CHANGE TRUCK** function to jump into an unlocked truck. In this case, the target truck does not necessarily need to be revealed as long as it is currently within about 50 meters from the camera. (I.e. moving the camera affects whether you can jump into a truck.)

Once a locked truck or trailer is discovered, it can be purchased in the garage or trailer store even if its rank requirement hasn't been met. On the other hand, an unlocked vehicle can never be discovered, so it isn't offered for sale until its rank requirement is met, even if the player has driven it.

Time

One hour in player time equals one day in game time. Thus, $2\frac{1}{2}$ minutes equals 1 hour, and 1 minute equals 24 minutes.

SnowRunner labels the various portions of the day as follows:

- Night: 00:00 – 06:00
- Morning: 06:00 – 13:00
- Afternoon: 13:00 – 20:30
- Evening: 20:30 – 00:00

When you start a new game, the time starts at 08:00, mid-morning.

In the navigation menu, you can press a key (default: **T**) to skip time. SnowRunner advances the time to the next of these times: 08:00, 13:00, 20:30, and 00:00. Changing to different times lets you check how your map looks under different lighting conditions. It also lets you quickly test time-dependent features such as sounds and music.

Take a Screenshot

To take a screenshot, simply press the **PrtScn** key on your keyboard. This makes a copy of the framebuffer that you can then paste into an image editor of your choice, such as IrfanView. Alternatively, if you’re running SnowRunner in a window, press **Alt+PrtScn** to copy only the currently selected window.

If you’re running the Steam version, you can also press Steam’s screenshot key, which by default is **F12**. In this case, the screenshot is saved to a folder managed by Steam, and you can review and manipulate it from the Steam library.

If you’re trying to set up a pleasing view for the screenshot, a few options are available.

To hide the main HUD and certain other HUD features, open the game settings and turn off the various options under **HUD**. Or you can leave the HUD enabled and then use an image editor to crop the screenshot to a central part of the screen where the main HUD elements aren’t in the way. Potentially you can adjust the **Third-Person Camera FOV** under **Video** settings so that the central part of the screen shows approximately the field of view that you want.

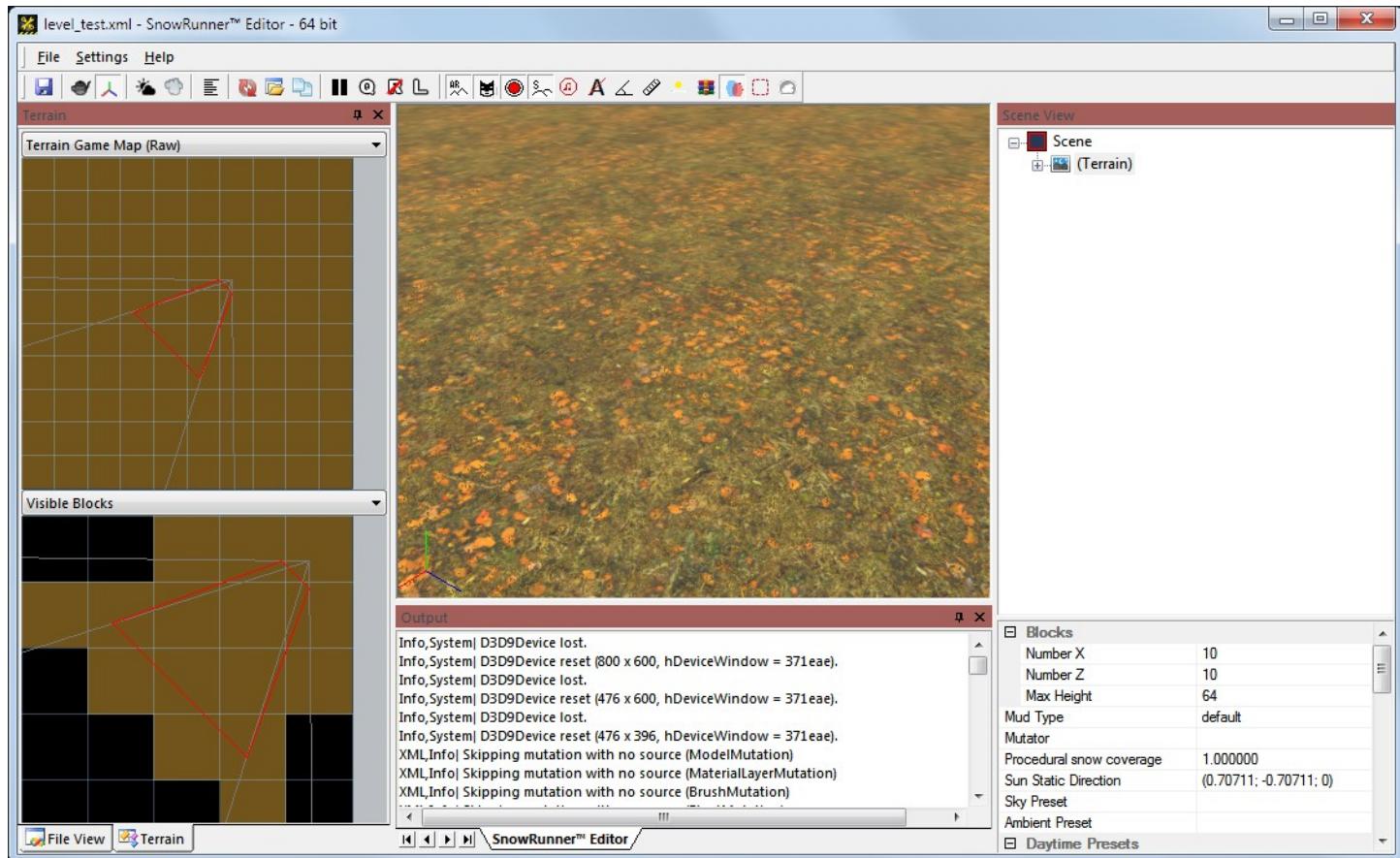
The **Free Camera** option in the Dev Tools is useful for getting a different angle.

Bug: Hiding the HUD disables the free camera. So you can’t get precisely the angle you want **and** get a pleasing full-screen view.

Alternatively, you can open the game settings, and under **Video**, turn on the **Legacy Camera**. With the legacy camera enabled, click the left mouse button to enter camera mode, then push the mouse up to move the camera to over the truck so that only the roof of the truck is visible at the bottom of the screen. In this mode, you can hide the HUD and get a good view across most of the window. If you want to cut out the truck, you only need to cut off a small portion of the screen shot.

Introduction to the Map Editor Window

The Editor window contains a number of panels that I describe here. I've moved descriptions of the less useful functions to page 256, but that still leaves a quite a pile of interaction options that are useful to understand.



Menu Bar

The menu bar at the top of the window has **File**, **Settings**, and **Help** menus.

File Menu

Possible actions in the **File** menu are as follows.

Close File closes the map and returns to the **File View** panel. If there are unsaved changes, the Editor asks if you want to save first.

Bug: Edits in the property panel are not considered to be unsaved changes until the edits are committed by changing the selection, e.g. by left-clicking elsewhere.

Reload File From Backup opens a dialog that allows you to restore XML files from a selected backup. Note that bitmap files are not backed up and cannot be restored from backup. More information about the limited backups is on page 35.

Saber used a large number of map reference sections that they have kindly made available for modders to use. The Example Maps section on page 26 describes how to use **Unpack references** to get these references.

Create default truck mod is used for creating truck mods. Truck mods are not covered in this book.

Save (shortcut: **Ctrl+S**) saves the map's named properties from memory to disk as XML files. This differs from bitmaps painted with a brush, which are updated on disk as soon as a change is committed.

Exit exits the Editor.

The **File** menu also includes a list of recent edited maps. Selecting one closes the current map and opens the selected map (with a save confirmation dialog if necessary).

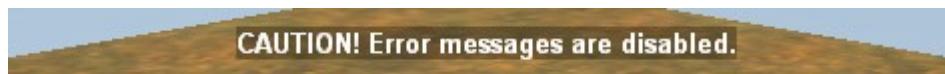
Settings Menu

Possible actions in the **Settings** menu are as follows.

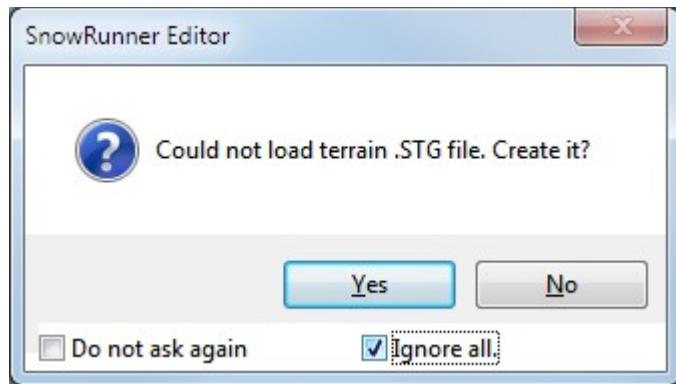
Ignore Warnings disables all warning dialog boxes. A checkmark appears next to the menu item when warnings are being ignored. Selecting **Ignore Warnings** again re-enables warnings.



The Editor also prominently warns you in the main panel when warnings are being ignored.



The **Ignore Warnings** setting can also be enabled by selecting **Ignore All** in any dialog warning box.



The **Ignore Warnings** setting applies across all maps, but it **does not** persist across Editor sessions.

Show Snow By Up Vector is another toggle setting. When enabled, it shows procedural snow cover on models and plants (page 173). This setting applies across all maps and **does** persist across Editor sessions.

Help Menu

The only option in the **Help** menu is **Guides**. This opens Windows Explorer with the folder containing the official guide PDFs that were installed with the game.

Toolbar

Under the menu bar is a toolbar with a row of buttons for various features. You can hover over each button to get a description of what it does.



This introduction covers just the vital buttons. Additional buttons are introduced in the relevant sections of this book, and the full set of toolbar buttons is summarized on page 256.

Save (shortcut: **Ctrl+S**) saves the map's named properties from memory to disk as XML files. This differs from bitmaps painted with a brush, which are updated on disk as soon as a change is committed.

Night is a toggle button. When enabled, it changes the lighting within the Editor to night mode. This is useful for testing how everything looks in the dark, especially lights.

Duplicate selected (shortcut: **Ctrl+D**) makes a duplicate copy of the currently selected feature.

Use Ruler is a toggle button. When enabled, a line segment or path can be overlaid on the terrain, and the Editor reports its length. More detail is on page 204.

Pack terrain converts the map to a format that can be played in the game and uploaded to mod.io. More detail is on page 39.

Zone settings opens the Zone Settings Editor. More detail begins on page 266. **Caution:** while the Zone Settings Editor is open, the main SnowRunner Editor window is completely unresponsive. Close the Zone Settings Editor to return to the main SnowRunner Editor.

Region settings opens the Region Settings Editor. More detail is in 367. **Caution:** while the Region Settings Editor is open, the main SnowRunner Editor window is completely unresponsive. Close the Region Settings Editor to return to the main SnowRunner Editor.

Main Panel

Much of your editing time is spent in the main panel. This panel is unlabeled, but you can find it centered between the **Terrain** panel on the left and the **Scene View** panel on the right. It shows a “camera eye” view of your map.

Navigate in the Main Panel

The main panel shows the 3-D map from the perspective of a virtual camera. To view different parts of the map, the camera can be moved in all three dimensions, it can be rotated left and right, and it can be pitched up and down. To prevent disorientation, the camera is never tilted left or right.

A small icon in the lower left of the main panel shows the coordinate axes as they appear from the current camera angle. The red line shows the X axis and points east. The green line shows the Y axis and points up. The blue line shows the Z axis and points north. The SnowRunner coordinate system is described in more detail on page 420.



You can move the camera around as follows:

- left click and drag – Move the camera around whatever terrain you clicked on and rotate the camera to continue pointing at that spot. This is my favorite method for rotating the camera.
- mouse wheel up or down – Push the camera closer to the terrain under the mouse or pull it further away. (Caveat: the most recent click or drag must have been in the main panel.)
- **Ctrl** + left click and drag – Keep the camera pointed the same direction while moving it relative to whatever terrain you clicked on. This effectively drags the terrain around under the mouse, and it’s my favorite method for moving the camera across the map.
- **Shift** + left click and drag – Keep the camera fixed in space and rotate it to point in new directions.
- **Alt** + left click and drag – If a feature is selected, move and rotate the camera to keep that feature in the same spot in the view (or off view). If no feature is selected, the **Alt** modifier is ignored.

It is easy to get completely lost in the main panel so that no terrain is visible. If that happens, double click in the **Terrain** panel (page 52) to recenter your camera over the terrain.

If the camera is below the terrain, the display engine also won't draw it. In that case, double clicking in the **Terrain** panel moves the camera close to the terrain, but still pointing up at it, so the terrain is not drawn. In that case, left click in the main panel and drag downwards to tilt the camera downward while moving the camera itself upward until the terrain comes into view.

Extremely rarely you may end up with the camera pointed straight up or down, and left click and drag won't pitch it back to a proper angle. The Editor tries to prevent this by keeping the camera just off of vertical, but its algorithm can occasionally fail. **Shift** + left click and drag may work when other methods fail. Otherwise, you'll need to quit the Editor and edit the file that stores the camera position:

`%appdata%/SpinTiresEditor/ViewTerrain.xml`. Find your map in the file and delete its entry. The next time you open the level, the camera resets to its default position and angle. (This may be underground, but that's more easily fixable.)

You can select a target in the main panel as follows:

- left click – Select the terrain block, model, plant, or object vertex under the mouse. Other types of objects cannot be selected in the main panel. This notably includes trucks and zones.

Bug: A model, plant, or object vertex cannot be clicked on in the main panel if it is off the edge of the map. Instead, select it from the **Scene View** (page 53).

- double left click – Select the terrain block, model, plant, or object vertex under the mouse and move the camera (without rotation) to center the selected item in the main panel.

Right click brings up a context menu. The context menu is described on page 64.

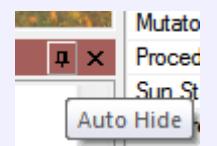
Output Panel

The **Output** panel (below the main panel) shows logging information and error messages from the Editor and the embedded display engine. You may refer to it occasionally when things go wrong.

The **Output** panel often includes lines saying "D3D9Device lost" and "D3D9 Device reset". These messages look ominous, but they seem to be part of normal operation, and you can ignore them.

Bug: The **Output** panel does not automatically scroll to keep the most recent messages in view. That, plus the general inscrutability of the logging info makes the **Output** panel rather useless. By default, any important messages pop up a warning dialog anyway.

Tip: You can close the **Output** panel to reclaim some screen space. Be aware that the only way to restore the panel is to restart the Editor. Alternatively, you can hide the **Output** panel by clicking the pushpin icon in its upper right. A hidden panel can be restored without needing to restart the Editor.



Terrain Panel

The **Terrain** panel on the left includes two subpanels looking into the scene from a top-down view.

By default, the upper panel shows a view of the entire map. If the map is not square, it is stretched into a square for display in the terrain panel. A truncated triangle shows the field of view of the main panel's camera.

The **Terrain** panel only shows the colors that are painted on the terrain itself. So, for example, it shows a road overlay, but not a house model. On the other hand, it may show a shadow for the house, since the shadow is painted onto the terrain.

A dropdown menu above the upper panel changes it to show many other map textures used by the display engine. I don't find these to be very useful. Once you're done looking around, flip it back to the default **Terrain Game Map (Raw)** at the end of the dropdown list.

The lower panel is similar to the top one, but by default it zooms in on the terrain blocks that are visible to the camera. (I'll describe terrain blocks later in the guide.) A truncated triangle again shows the field of view of the main panel's camera. Only those terrain blocks within this view (or very near it) are drawn in the lower panel.

Another dropdown menu allows the lower panel to be switched to show only the **Selected Blocks**. Since selecting multiple terrain blocks is a pain (page 259), I find the **Visible Blocks** default to be the more useful view for the lower panel.

Changes made from the dropdown menus in the **Terrain** panel do not persist across Editor sessions.

Below the two map panels is a block of text that gives various information about the selected terrain blocks. The information isn't particularly useful, though, so I never pay any attention to it.

Left click in either part of the terrain panel to select a terrain block.

Double click in either part of the terrain panel to select a terrain block and fly the camera in the main panel to that block.

It is tempting to rotate the scroll wheel up and down within the **Terrain** panel to zoom in and out, but this actually moves the main panel camera. And because the mouse is positioned outside the main panel, the target point for the zoom is off the edge of the view, so zooming in isn't very effective. Move the mouse back to the main panel before zooming.

File View Tab

At the bottom of the **Terrain** panel are two tabs which you can use to switch between the **Terrain** panel and the **File View** panel.

The **File View** panel lists selected files and folders in the **Media** directory, generally only the ones immediately necessary for map or truck mod editing. To view all files and folders, see page 260.

Double-click any map's XML file in the **prebuild** folder to open that map. Only one map can be open at a time, so if you have a map open already, the Editor prompts you to save it before opening a new one.

Right-click the **prebuild** folder to open a context menu that allows you to create a new map (**New Terrain...**).

Remember that you can open a recently edited map by selecting it from the **File** menu, but you can open any map at all by double clicking its XML file within the **prebuild** directory.

Controls for Hierarchical Lists

The **File View** tab, the **Scene View** panel, and the property panel all display hierarchical lists that share a number of controls in common.

A few common controls are performed by the mouse:

- Left click an item to select that item.
- Left click on a **+** to the left of a folder or container to expand that container, showing its contents.
- Left click on a **-** to the left of a folder or hierarchical container to contract that container, hiding its contents.
- Or double click on a folder or hierarchical container to expand or contract that container.

Bug: Double click has no effect in the **Scene View** panel.

- Right click in the **File View** or **Scene View** brings up a context menu if an appropriate item is clicked.

Keyboard controls are also supported, but not particularly useful. The full set of keyboard controls is described on page 263). Some useful keys for the property panel are described starting on page 57.

Scene View Panel

The **Scene View** panel on the right contains a categorized list of all the features on the map. It uses the standard controls for hierarchical lists, above.

You can add an additional layer of grouping to some feature categories. Groups are described on page 55.

- Left click on a feature or category to select it.
- Right click on a feature or category to bring up a context menu. The context menu is described on page 64.
- Double click to select a feature and fly to it.

Bug: Double click in the **Scene View** requires that the second click is neither too fast nor too slow. Right around half a second between clicks seems to do it. If you can't get the hang of it, single click instead and use the **F** shortcut key to fly to the feature.

Introduction to Features

This book groups features into three types:

- A truck, model, plant, sound, reference, or zone is a “simple object”. It has a single location and orientation (even if it has no visual presence).
- An overlay, river, or river markup or the ruler is a “path object”. It is a path that links a number of nodes, each with an ordered position but not an orientation.
- A sound domain is a “polygon object”. It is a polygon that encloses a number of nodes, each with only an unordered position.
- Everything else is just a feature, not an object. This includes geometry features, materials, and distributions.

Path objects and polygon objects are together referred to as multi-node objects.

The bare term “objects” encompasses simple objects, path objects, and polygon objects.

Each feature has built-in properties based on its type that can't be edited. Besides the feature's obvious appearance, it may have type-specific interactions with trucks, or cast shadows in certain ways, or make certain noises, or many other behaviors that are entirely determined by the game designers.

Each feature also has a set of named properties that are defined in the Editor. This almost always includes a position and may include other properties such as whether standard physics are disabled for the object, or whether it should cast shadows, or play certain music, or many other behaviors that can be chosen by you, the map designer.

Each **named property** has a name and a value, and it is displayed in the properties panel at the bottom of the **Scene View**. Named properties are recorded in your map's XML file in the **prebuild** directory. Unless otherwise stated, references to “properties” in this guide refer to named properties. Named properties can be edited in the property panel, described below.

Every named property is presented in the following standardized format. If you need to look up any property, you can search this book using the property name or (if necessary for disambiguation) its hierarchy.

hierarchy.example feature: category.example property: example type of value

The property does something as described here.

Default: the default value; if its meaning isn't obvious (e.g. for a blank default), it is explained further.

In place of a location, a feature may have one or more bitmaps that describe where the feature is present and/or how the feature's attributes change across the entire map.

A **bitmap** is painted onto the terrain using a brush. Each pixel in the bitmap corresponds to a local region of the terrain and describes some aspect of the feature in that region. Each bitmap is recorded in a file in your map's directory within the **prebuild** directory. The name of the bitmap file is implicit for some features, and it is a named property for other features.

Object Groups

Subsets of objects can be grouped together, allowing some amount of organization in maps which may contain a very large number of objects. But since it will take you a while to accumulate that many objects in a map, I'll hold off describing groups until page 247.

Property Panel

At the bottom of the **Scene View** panel is an unlabeled sub-panel that I call the property panel. It shows the named properties of the feature selected in the main **Scene View** panel and allows those properties to be edited where applicable.

Most named properties can be edited in the properties panel, either by directly typing a new value or by some other means of selecting a new value. The editing conventions are different depending on the property type. Editing methods for each property type are described in the sections below.

When a property is edited, the changed values are displayed with bold text in the properties panel. The text remains bold until you change the selection or until you change a value back to its previous value.

Group	MainGroups
Brand	air_conditioner_01
Tag	
CollisionType	STATIC
Position	
X	118.876549
Y	20.078430
Z	-74.301620
Rotation	
X	0.000000
Y	-0.000000
Z	-0.000000
Scale	2.000000
Disable Day Static Shadow	True
Disable Night Static Shadow	True
Freeze Physics	True
Animation Camera Frame Name	<No anim camera frames with...

If the changed values result in a visual change to the feature, it is usually redrawn with its new appearance as soon as you confirm the change, e.g. by pressing return. Some property changes, however, don't take effect until you rebuild the terrain (page 65).

When selecting named properties, the **Tab** key switches from the left column to the right column, and the **Enter** key confirms your edits and returns to the left column.

Bug: For an editable text value, pressing the **Tab** key confirms the change (and updates the main panel if appropriate), but it does not return to the left column. Instead, neither column is selected. For ease of keyboard navigation, I recommend that you use the **Enter** key to confirm the change and return to the left column.

A few named properties are for display only or otherwise cannot be edited once the feature is created (e.g. the map size).

Other controls for navigating through the list of properties are summarized in the Controls for Hierarchical Lists section on page 53.

Edit a Property using a Widget

A selected object's position, orientation, and scale properties or a node's position and width properties can be modified by manipulating a widget in the main panel. The new values are updated in the property panel when you release the mouse button, but unlike manually entered values, the new values are not displayed in bold.

Edit a Property by Typing Directly

For some properties, you can directly type a new value:

- To replace the value with a new one, click the property name in the left column of the property panel and begin typing. The cursor automatically moves to the property value in the right column with your new text.
- To edit the existing value, click anywhere in the property value in the right column to begin editing at that position.

Press the **Enter** key or left click somewhere else to complete the data entry.

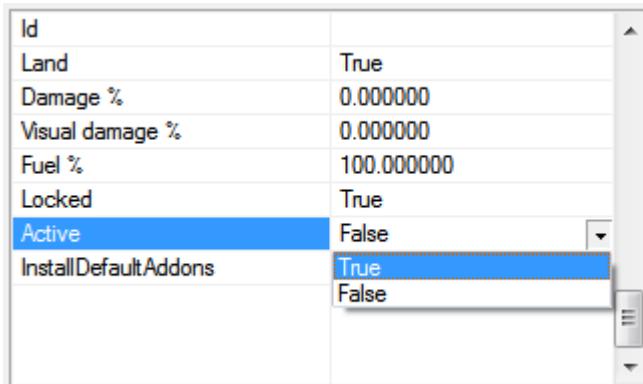
Group	MainGroups
Org X	118.834381
Org Z	-72.565430
Dir	(1; 0)
Id	I am typing directly...
Icon 30x30	
Icon 40x40	
Name	

Many named properties look like they have a value that you can type directly, but when you click the property some other interface appears. These other interfaces are described below.

Edit a Property Using a Dropdown Menu

For a property with a dropdown menu, a small downward-pointing arrow appears when you select the property. Click the arrow to get a menu of values that you can choose among. You can also click at the right end of the property where the arrow will appear to select the property and invoke the dropdown menu with one click.

Properties that can be either **True** or **False** are the most common use of the dropdown menu.

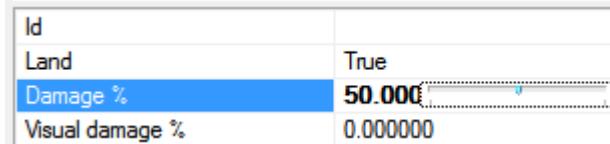


Double-click the property (either the name or the value) to cycle among the values. This is particularly handy for switching between **True** and **False**.

Tip: You have to click once in the **right** column before a double click can register. However, a double-click in the **left** column cycles the value without requiring an extra click. (Now aren't you glad you're reading this book? That's the kind of secret knowledge that could shave **seconds** from your map editing time. Ha ha.)

Edit a Property Using a Slider

For a property with a defined range of numerical values, clicking in the right column of the property causes a slider to appear to the right of the property value.



The slider can be manipulated using the mouse or keyboard:

- Drag the slider indicator to the right or left to quickly increase or decrease the value.
- Click on the slider to the right of the indicator or press the **Page Down** key to increase the value by a large step.
- Click on the slider to the left of the indicator or press the **Page Up** key to decrease the value by a large step.

- Press the right arrow key to increase the value by a small step.
- Press the left arrow key to increase the value by a small step.
- Scroll the mouse wheel up or down to increase or decrease the value by a moderate amount.
- Press the **End** key to set the value to the maximum value.
- Press the **Home** key to set the value to the minimum value.
- Press the **Enter** key or left click elsewhere to confirm the selected value.

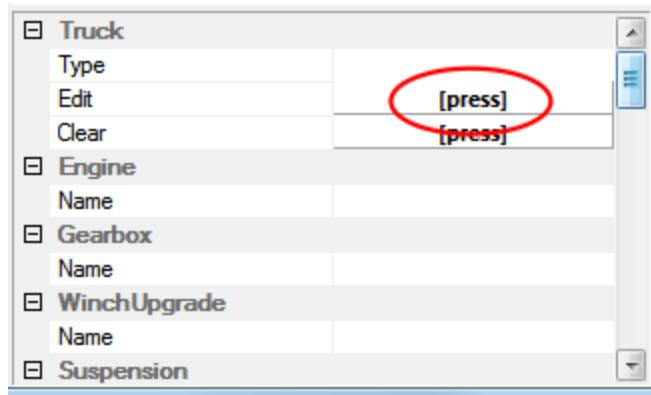
There is no way to type a value directly into a slider property.

Edit a Property That Selects an Asset

For properties that choose among things that can be represented visually, the Editor uses a **Select Asset** window.



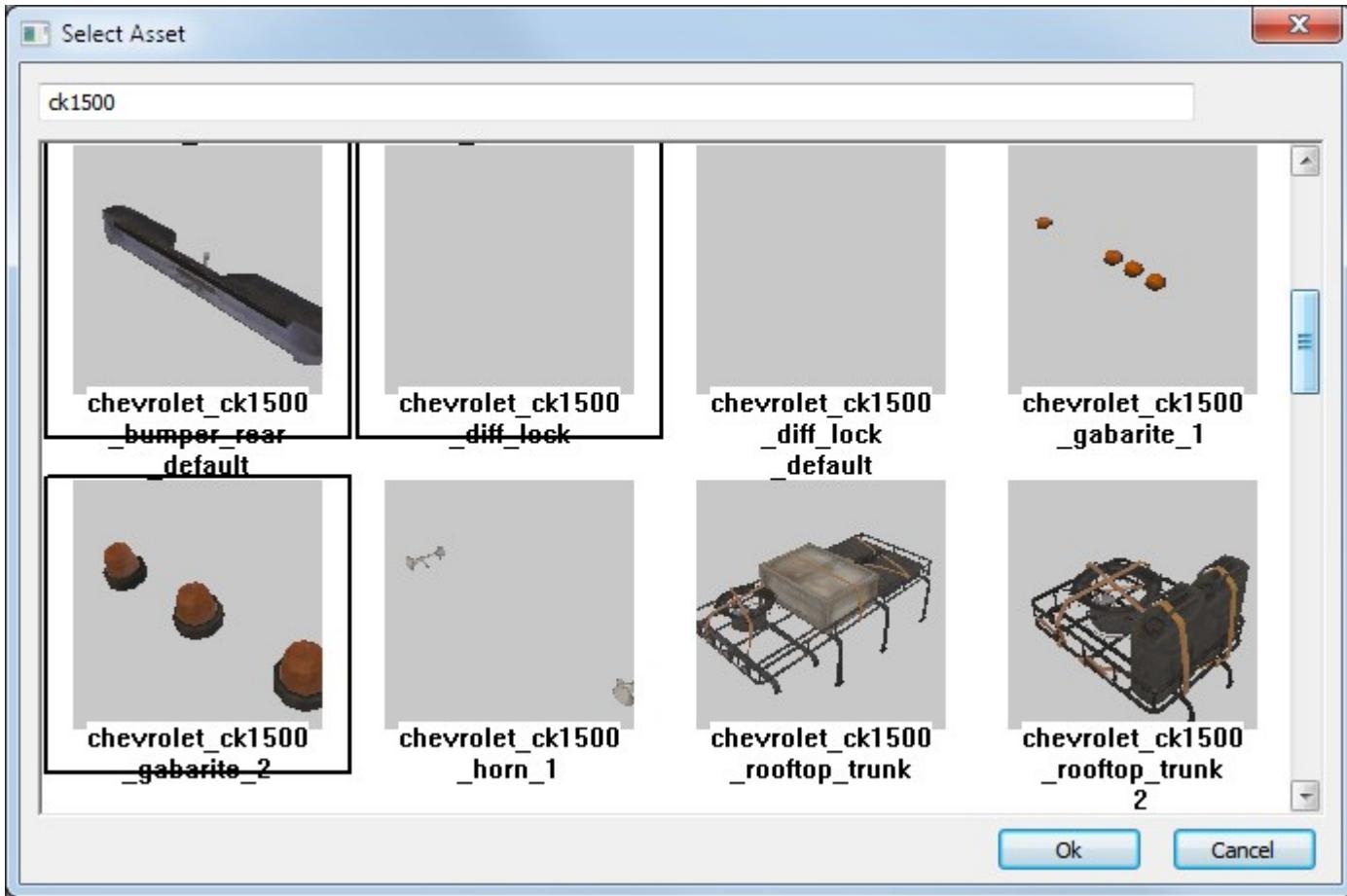
For some objects, the **Select Asset** window pops up when the object is created in order to choose its type. For other objects, an asset can be selected at any time. For these, the Editor displays the current selection as an uneditable value, but it adds a second pseudo-property that allows you to edit the asset selection. The name of the pseudo-property is **Edit**, and its value displays as **[press]**. Left click the **[press]** text to open the **Select Asset** window. (The **[press]** text acts like a button, even though it doesn't look like one.)



Bug: The first time you open a **Select Asset** window for a particular asset category, the Editor may lock up for 5 or more seconds as it creates icons for all possible assets in that category (page 35). Later uses may pause again if the Editor decides to check for any changes, but never as long as the first time.

Bug: In most programs, when you click in the scrollbar above or below the current scroll position, the window scrolls up or down by exactly one page. However, in the **Select Asset** window, it instead scrolls by a fixed percentage of the total number of entries. This makes it hard for your eye to predict where to look for assets that have newly scrolled into view. And when the list of assets is very long (such as for truck add-ons), the fixed scroll percentage scrolls so far that it actually skips over many of the available assets. When you click the up or down arrow buttons at the ends of the scrollbar, it scrolls by a smaller fixed percentage. This has a similar bug, but at least it scrolls by a small enough percentage that it shouldn't skip over any assets. You can also filter the asset list (described below) to reduce the number of entries to scroll through.

You can filter the list of assets by typing in the blank panel near the top of the dialog box. Only assets that include your text somewhere in the name are shown in the dialog box.



To select an asset, left click it in the **Select Asset** window and then click **OK** or press the **Enter** key. Or double-click an asset to choose it and immediately confirm your choice.

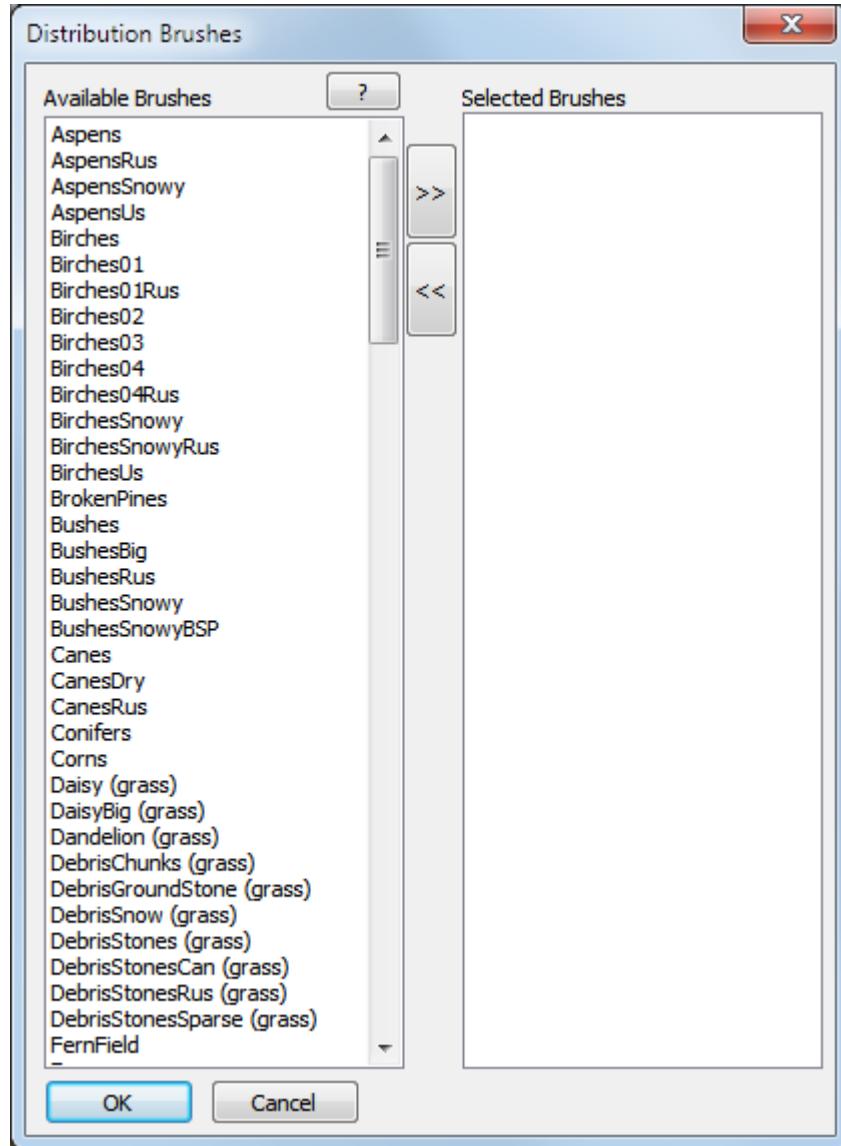
Click **Cancel** or press the **Esc** key to cancel the asset selection. If the **Select Asset** window was opened during the creation of a new object, the object creation is canceled. If you click **OK** or press **Enter** when no asset is selected, it is the same as clicking **Cancel**.

For properties where an asset is not required, you can click **[press]** next to the **Clear** pseudo-property in order to clear the selected asset.

Bug: The **Clear** property is shown for many properties where an asset is required. For these properties, clicking **[press]** next to **Clear** does nothing.

Edit a Property that Selects Brushes

The appearance of some features depends on which brushes are attached to the feature. Zero, one, or many brushes can be selected using the Select Brush window. The Editor displays the names of the current brushes as the value of a **List** property. To edit the **List** property, click [press] next to the following **Edit** pseudo-property.



Move a brush from the left **Available Brushes** column to the right **Selected Brushes** column by clicking its name and then the **>>** button. Move it back by selecting it from the right column and clicking **<<**. Or quickly change a brush from one side to the other by double clicking it in either column.

When you are done, click **OK** or press **Enter** to confirm your selection of brushes. Or click **Cancel** or press **Esc** to cancel your changes.

Rearranging Panels

The Editor has extensive features for rearranging its panels, i.e. moving them to different positions, moving them out to a separate window, or hiding them. However, any nice arrangement you come up with is lost when you restart the Editor, so these features just don't seem worth documenting.

If you do accidentally mangle your panels (e.g. by closing one), you can restart the Editor to get back to its default panel arrangement.

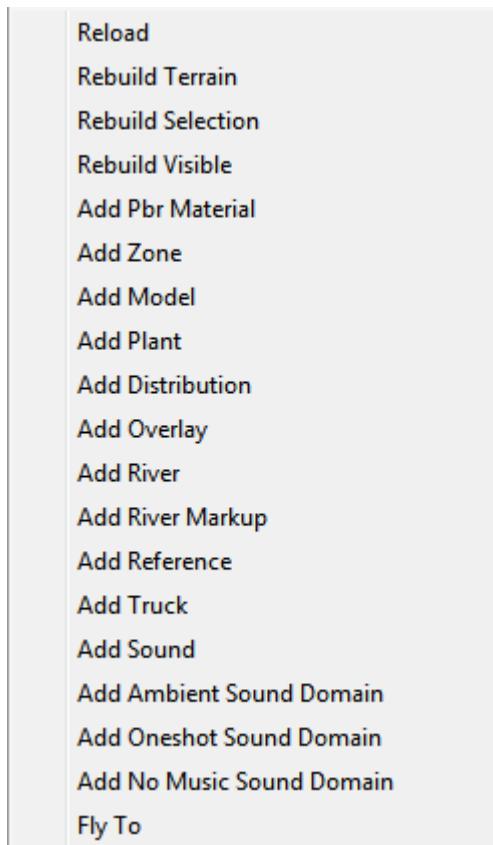
The Context Menu

The SnowRunner Editor has a context menu that can be invoked by right clicking in a couple of different places. The context menu always includes a number of common entries, and more entries are added for the context of the chosen feature.

In most cases, if you right click anywhere in the main panel, the context menu appears for the currently selected feature. However, if a feature with a bitmap property is selected, right click in the main panel operates the paint brush instead.

If you right click on a feature in the **Scene View** panel, the context menu appears for that feature. If you close the context menu without selecting an item from the menu, the feature is selected as if you left-clicked on it.

The rest of this section describes the common context menu entries, plus a few entries that are used by a number of features.



Bug: The context menu only works in the main panel when something is selected at the **(Terrain)** level or below. Right click in the main panel is ignored if the top-level **Scene** is selected or if nothing in the **Scene View** panel is selected (so the property panel is blank). This most commonly occurs after the map is saved, after the terrain is rebuilt, after a feature is deleted, or after a paint brush is deselected. My habit is that if right click doesn't work in the main panel, I just left click and then right click again.

Reload

Reload discards all unsaved changes and reloads the map from disk.

Bug: The Editor asks for confirmation before reloading, but it doesn't explicitly say whether your map has any unsaved changes. On the other hand, if you really want to reload, it's probably because you want to discard your unsaved changes, so I guess it's not so bad.

Rebuild Terrain

As a performance optimization, the Editor doesn't always fully update the appearance of the map as you make changes. Instead, you must manually tell the Editor when to take the time to rebuild the terrain, and you have the option rebuild all of the terrain or only a selected portion.

Each of the **Rebuild** actions first commits any property edits in progress, and it clears any current feature selection.

Rebuild Terrain rebuilds everything in the map. For a large, complex map, this can take a very long time.

Rebuild Visible rebuilds the visible terrain blocks. I'll sometimes use this when editing a large map so that I don't have to wait for a full rebuild. The following shortcuts are applied:

- It omits grass from the rebuilt terrain blocks.
- It distributes plants without regard to roads, rivers, and mud.
- It doesn't update the map in the **Terrain** panel.

Rebuild Selection rebuilds only the selected terrain block(s), using the same shortcuts as **Rebuild Visible**.

Because selecting multiple terrain blocks is a pain (page 259), I find this option to be not very useful.

The toolbar has a toggle button for **Quick mode**. When enabled, **Rebuild Terrain** omits grass and shadows from the rebuilt terrain. This can speed up the rebuild quite a bit, especially on large maps. Note that **Quick mode** has no effect on **Rebuild Selection** and **Rebuild Visible**, which omit grass anyway, but which generate shadows regardless of the **Quick mode** setting.

Pack terrain always rebuilds the terrain before packing the map. It performs this rebuild without taking any shortcuts, and it ignores the **Quick mode** setting. I.e. it ensures that the packed map is complete and correct.

Add Feature

Add Feature adds the named feature type to the map and selects it. The feature's position is initialized to the terrain in the center of the main panel. If there is no terrain visible at that spot, the feature is added at a corresponding point off the map.

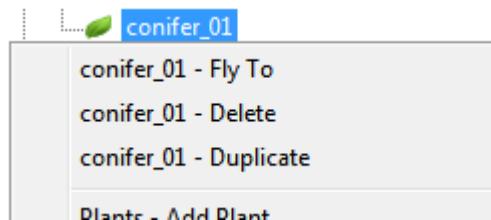
Fly To

Fly To moves the camera (without rotation) to a position where it can see the selected terrain block(s).

This is distinct from **Feature - Fly To**, described below. The **Fly To** option is only available when no **Scene View** feature is selected. The secret keyboard shortcut is the **F** key. It's usually faster to double click in the **Terrain** panel, however.

Feature Context

When an individual map feature is selected, extra options are added to the context menu to interact with that feature.



Fly To

Feature - Fly To moves the camera (without rotation) to point at the selected feature. The secret keyboard shortcut is the **F** key.

You can also double click a model or plant in the main panel to select and fly to it, but this method lets you select and fly to it from the **Scene View**.

Delete

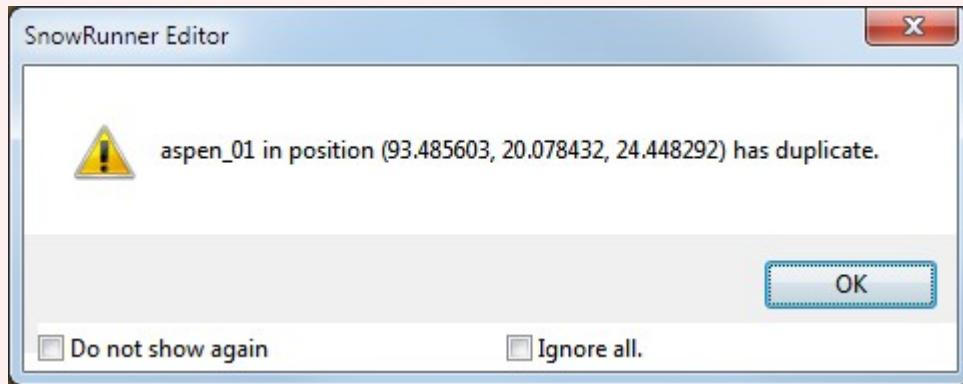
Feature - Delete deletes the chosen feature and clears the current selection. The secret keyboard shortcut is the **Delete** key.

Bug: After a feature is deleted, nothing is selected. If you then press the **Delete** key to delete another feature when none is selected, a confirmation dialog pops up anyway. If you click **Yes** in the confirmation, the Editor crashes.

Duplicate

Feature - Duplicate makes a copy of an object and selects the copy. This copies all of object's properties to the new object and selects it. The object can also be duplicated with a shortcut key, **Ctrl+D**.

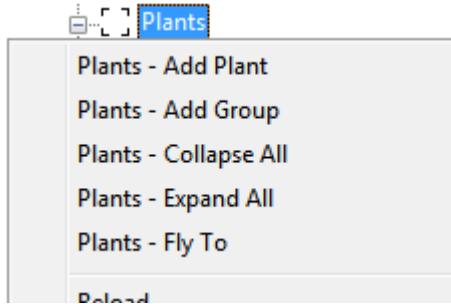
Bug: Duplicating a model or plant displays a warning that the object is duplicated. This dialog may be repeated even as you delete the duplicate object. The checkbox for **Do not show again** only works if the next warning dialog has the exact same text, so it can suppress repeated warnings about the same objects in the same positions, but it won't suppress the warning when duplicating a different object.



Tip: You might want to cultivate a habit of pressing **Ctrl+D** to duplicate and then **Enter** to dismiss the warning dialog for models and plants. Then immediately move the duplicated object away so that the warning does not recur.

Category Context

When a feature category (such as **Models**) is selected, extra options are added to the context menu to interact with that feature category.



When an individual feature is selected, the same category options are added to the context menu in addition to the options for the individual feature as described above.

Add Feature

Features - Add Feature is the same as the generic **Add Feature** menu item, but is a little closer to the mouse when the context menu is invoked on another feature of the same type.

Add Group

Features - Add Group creates a new group under the feature category. Groups are described on page 247.

Collapse and Expand

Features - Collapse All contracts the feature category and all groups within the category so that their contents are hidden in the **Scene View**.

Features - Expand All expands the feature category and all groups within the category so that their contents are shown in the **Scene View**.

Fly To

Bug: **Features - Fly to** flies to the selected **terrain block**. (This is distinct from **Feature - Fly To** for an individual feature.)

Map Features

The following sections describe all of the Editor features that are not directly related to zones and objectives. Zones and objectives are described beginning on page 265.

Introduction to Top-Level Properties

The following properties apply to the map as a whole. This is not a complete list; the remaining top-level properties will only make sense after exploring the other map features. Page 251 reviews all top-level properties in the Editor.

Note that I use a standard format for describing named properties and their values. See page 54 for more information about this format.

Map Name

Scene.(Terrain): Description level.Id: UI text (page 282); formatting tags are not supported; max ~30 characters

Specifies the map name to display in the navigation map and global view.

Default: blank; the name of the map defaults to `mod_map_name`.

The name of the map is also used as the name of the region if a region isn't specified.

Despite setting a map name here, the map is still listed as `mod_map_name` in the SnowRunner menus, i.e. when you start a custom scenario or view a saved game. However, when someone subscribes to your published map, the SnowRunner menus are finally updated with the published mod name (page 382).

Map Description

Scene.(Terrain): Description level.Description: text

No known purpose.

Default: blank.

Map Dimensions

Scene.(Terrain): Blocks.Number X, Z: read-only integer in the range 2 – 84; even numbers only

Indicates the number of terrain blocks in the X and Z directions, as set by the **U Blocks** and **V Blocks** parameters when the map was created.

The map dimensions cannot be changed in the Editor once the map has been created, and it is difficult or impossible to change them by hand while retaining the map geometry. Choose carefully.

The dimensions of a single terrain block are described on page 421.

Map Borders

The game puts an impassable border near each edge of the map. The border is drawn about 8 meters from the map edge, but the player's truck is stopped at about 14 meters from the edge (as measured at the **center** of the truck). Make sure that the player never needs to drive closer to the edge of the map than that.



A secondary system physically blocks a trailer when it is backed too close to the edge. I can't figure out the algorithm, but it lets at least one trailer overhang the edge, while other trailers can't even reach the drawn border.



A map tends to look better if its border is rarely or never visible, so it is useful to understand how the border is drawn.

Each border is drawn at the height of the terrain under the part of the border nearest the truck. E.g. the entire north border is drawn at the height of the terrain at the X position of the truck and 8 meters from the north edge of the map. If there is a river at that point, the border is drawn at the height of the river. The border is 4 meters high.

Tip: You can hide the border by dropping the height of the terrain 8 meters from the edge of the map. If you do this, though, you'll want another visual obstacle such as trees or an uncrossable ridge.

A border is only visible when the player's truck is within about 64 meters of the edge of the map. Note that this is based on the **truck** position (measured at its center), not the **camera** position.

Tip: You can hide the border by keeping trucks more than 64 meters from the edge of the map (e.g. by uncrossable water). It is not possible to hide the border under water.

Maximum Height

Scene.(Terrain): Blocks.Max Height: integer in the range 1 – 512, in meters

Indicates the maximum possible height of the terrain.

The maximum height of a map cannot be changed in the Editor once the map has been created, and it is difficult or impossible to change it by hand while keeping other map features in place. Choose carefully.

Max Height also determines the **precision** of the terrain height. The terrain height is expressed internally as 4096 divisions of the **Max Height** value.

A large value such as 512 for **Max Height** means that terrain rises in 0.125 meter (5 inch) steps. This is smoothed out across the 0.5 meter terrain tiles, but if every 1 or 2 meters includes a 0.125 meter ramp, it makes for a bumpy ride. If you use a large **Max Height** value, you'll need to avoid or heavily disguise areas that have moderate slopes. It only looks decent and drives reasonably where the terrain is completely flat or where it is fairly steep.



Tip: Instead of creating map with a large mountain that requires a large **Max Height** value, split it up into a series of difficult ridges that require less total height.

Tip: You can get a bit more headroom for mountainous terrain by lowering the base height of your terrain before you start adding ridges. E.g. with **Max Height** of 64 meters, the default terrain height is 20.08 m. You can reduce the height of some part of the map to ~5 meters, then use a large **Flatten** brush (page 96) to flatten the entire map down to that height, giving you an extra 15 meters of height to work with. I like 5 meters since that still allows enough depth to dig a truck-drowning river or lake.

The SpinTires series has traditionally used a **Max Height** of 64 meters, which works well. 128 meters is also reasonable.

There is no advantage to making the value a power of 2. It's just tradition.

Default Height

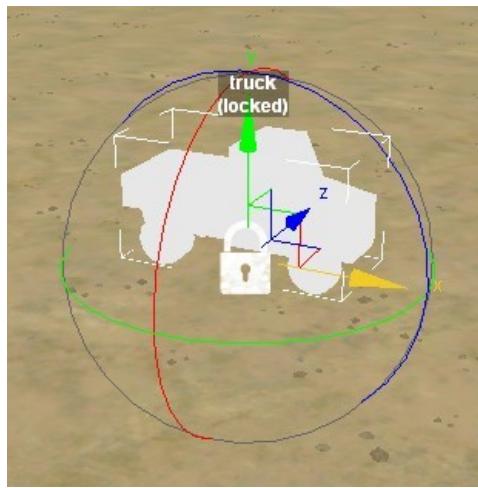
Each map starts with the terrain at a default height that depends on the **Max Height** value. Specifically, the default height is $\text{Max Height} \times 1285 / 4096$. E.g. with a max height of 64 meters, the default height is 20.078 meters.

Trucks and Trailers

The minimum requirement to publish a working map is that it has a starting truck, so we'll begin with that.

Right click anywhere in the main panel or on any feature in the [Scene View](#) and select [Add Truck](#). (If it doesn't work, review the context menu and its caveats and bugs starting on page 64.) This same option is used to add a trailer. Page 80 describes how to specify what kind of truck or trailer you want.

The new truck comes with a widget in the main panel with a number of fancy interface elements. If these are all jumbled together, zoom in until you can distinguish the individual elements.



The text above the truck indicates its type, which for now is just a generic "truck". The text also indicates that the truck is locked, and the white padlock over the truck indicates the same thing. Locked trucks and all other truck details are discussed below.

The colored widget elements allow you to move and rotate the truck. These manipulations are described in the next section.

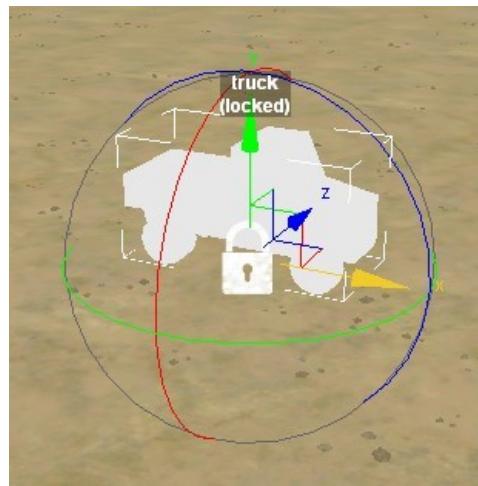
When a truck is created, it is automatically selected for editing. Unfortunately, you cannot click on a truck in the main panel to select it. You can only select a truck by left clicking it in the [Scene View](#).

Move and Rotate a Truck

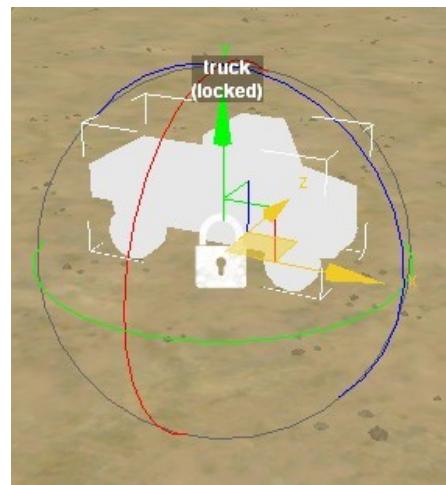
Move and Rotate in the Main Panel

When a truck is selected, the Editor draws an interface widget surrounding the truck in the main panel.

The widget includes a set of colored arrows corresponding to the X/Y/Z coordinate axes. Left click and drag an arrow to move the truck back and forth along that axis. By default, a truck is attached to the ground and cannot be raised into the air, so dragging the green Y arrow does nothing.

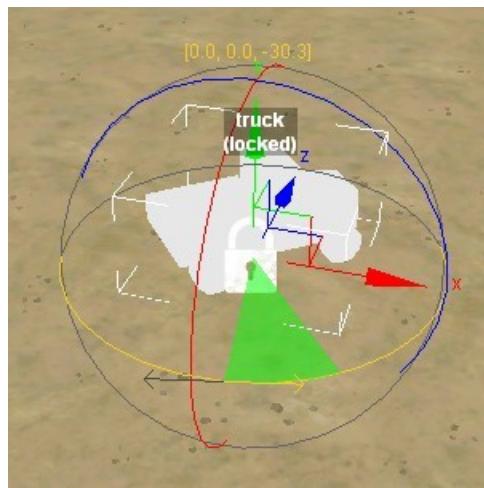


Connecting the bases of the arrows are three squares in 3-D space that designate the XZ plane (ground plane), XY plane, and YZ plane. Dragging one of these squares moves the truck around in the 2-D space associated with the plane. The squares often partially overlap on the screen, which can make it difficult to grab the right one. However, moving in the ground plane is the most generally useful, and the Editor always prioritizes that one when there's an overlap. Because a truck is attached to the ground by default, dragging the XY plane or the YZ plane only moves the truck in the X or Z directions, respectively.



The interface widget also includes a hollow sphere composed of three colored circles. Left click and drag the green circle to rotate the truck around its vertical axis (the Y axis). By default, a truck is attached to the ground and cannot tilt away from it, so the red and blue circles do nothing. You can also grab and drag any part of the sphere that isn't another interface element. This allows rotation not constrained to any plane, although it is still constrained when the truck is attached to the ground.

When dragging the mouse to rotate a truck, your intuition is likely to drag the mouse in a circle, but that is not how the Editor works. When you start dragging, the Editor draws an opposing pair of arrows tangent to the circle at the location where you first clicked. Drag the mouse linearly in the direction of one of the arrows to rotate the truck. The more you drag, the more it rotates (even past 360°), although rotation stops when the mouse leaves the main panel.



The truck is surrounded by a virtual box that designates the boundaries of the truck in the X, Y, and Z axes. The corners of this box are shown with white lines. Because the box has a fixed orientation, its boundaries expand and contract to follow the corners of the truck as the truck rotates.

Move a Truck in the Scene View

A truck's position and orientation can also be edited in the property panel at the bottom of the [Scene View](#).

Scene.(Terrain).Trucks.type: `Position.X`, `Position.Y`, `Position.Z`: numeric, in meters

Specifies the initial location of the truck.

Default: ground level in the center of the main panel.

Scene.(Terrain).Trucks.type: `Rotation.X`, `Rotation.Y`, `Rotation.Z`: numeric, in degrees of rotation

Specifies the initial orientation of the truck.

Default: 0, 0, 0; pointing east (+X).

Position	
X	-27.393368
Y	20.078434
Z	-31.153299
Rotation	
X	-0.000000
Y	-0.000000
Z	-0.000000

The position of a truck is listed in the **Position** group with three properties **X**, **Y**, and **Z**. The values can be edited directly and are measured in meters.

The orientation of a truck is listed in the **Rotation** group. The **X**, **Y**, and **Z** properties specify the truck's rotation around each axis, measured in degrees.

Bug: If you edit the position or rotation properties of a truck by hand, the Editor won't save those changes. Instead, the truck reverts to its previous position and orientation the next time you load the map, or when you rebuild the terrain. When hand editing a truck's position or rotation, also edit at least one other property, even if you change it right back to its original value. Or use the widgets in the main panel to move and rotate the truck. Changes made by these widgets are saved correctly.

Only trucks have this bug. Other types of objects are safe.

Bug: **Rotation.X** and **Rotation.Z** are swapped for trucks. I.e. increasing the **Rotation.X** value rotates the truck counterclockwise around the **Z** axis, and increasing the **Rotation.Z** value rotates the truck counterclockwise around the **X** axis. This is not the same axis swap as for models and plants!

For any given orientation, there are many combinations of **X**, **Y**, and **Z** that achieve the same rotation. You might notice that when you click away from the truck and then re-select it, the Editor has changed the **Rotation** property values. However, they still result in the same orientation.

The coordinate systems for position and orientation are described on page 420.

Land Property

Scene.(Terrain).Trucks.type: Land: True/False

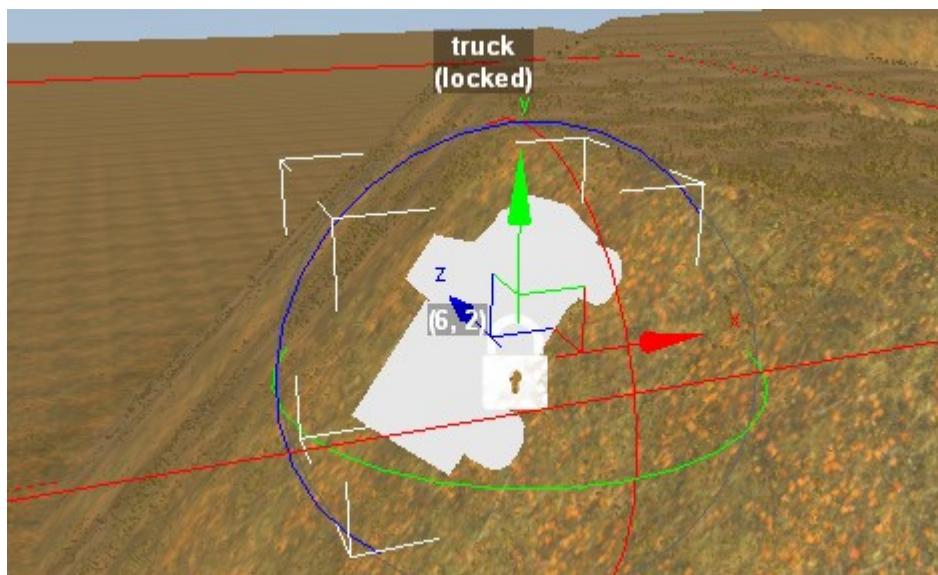
Specifies whether the truck's position and orientation are tied to the ground position and angle.

Default: **True**.

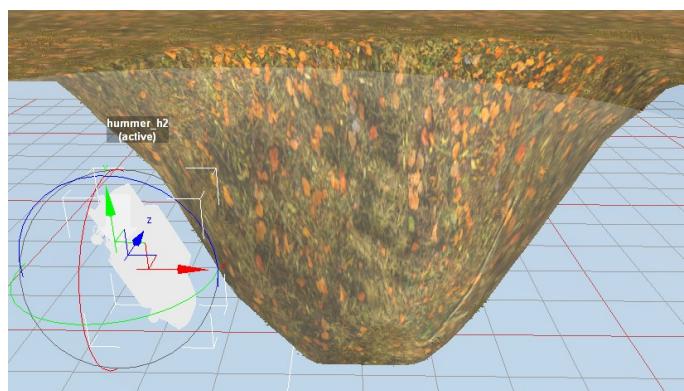
Remember that you can double click a **True/False** property to invert its value. The full set of property panel controls is described on page 55.

Position	
X	-27.393368
Y	23.546810
Z	-31.153299
Rotation	
X	44.999996
Y	44.999996
Z	-0.000001
Id	
Land	False
Damage %	True
Visual damage %	False

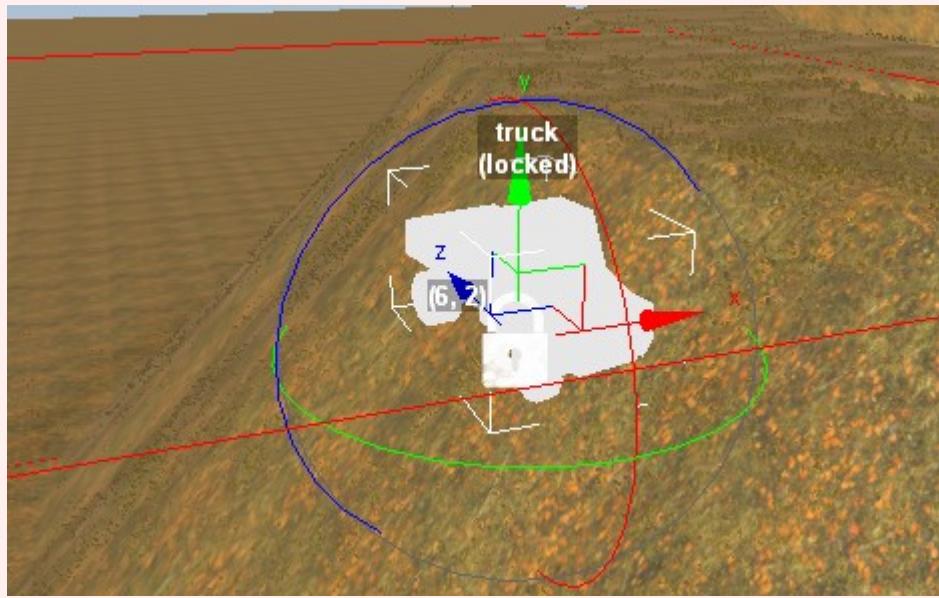
When the **Land** property is **True**, the truck is attached to the ground. The Editor tilts the truck so that its wheels make maximum contact with the ground.



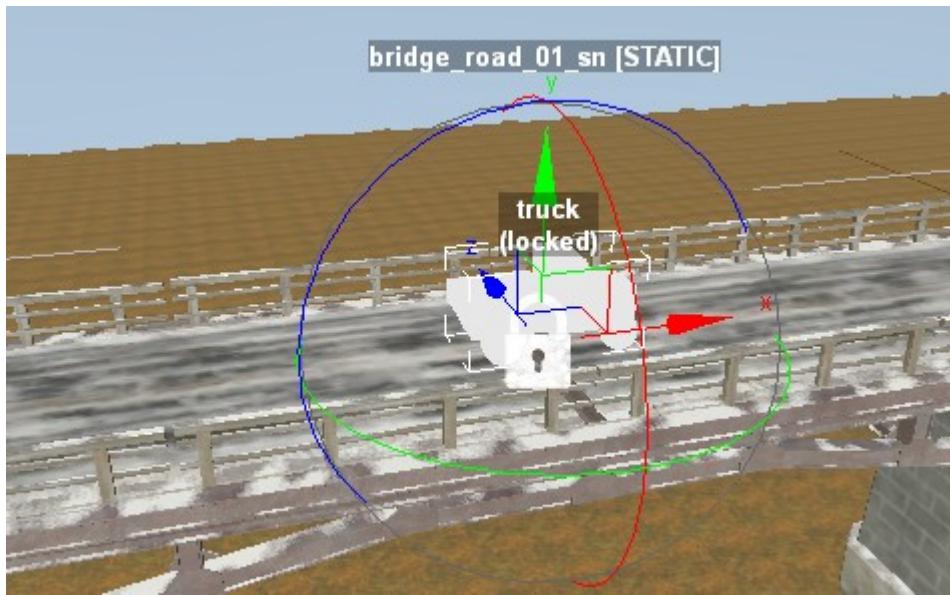
Although there generally isn't a reason for doing so, you can place a truck off the edge of the map. In this case, the **Land** property places the truck at the height and orientation of the closest edge of the map. This general behavior is followed by all ground-following features that are placed off the edge of the map.



Bug: The Editor shows the truck pitched correctly up and down, but it does not show it tilted correctly side to side. However, its initial position will be tilted correctly in the game.

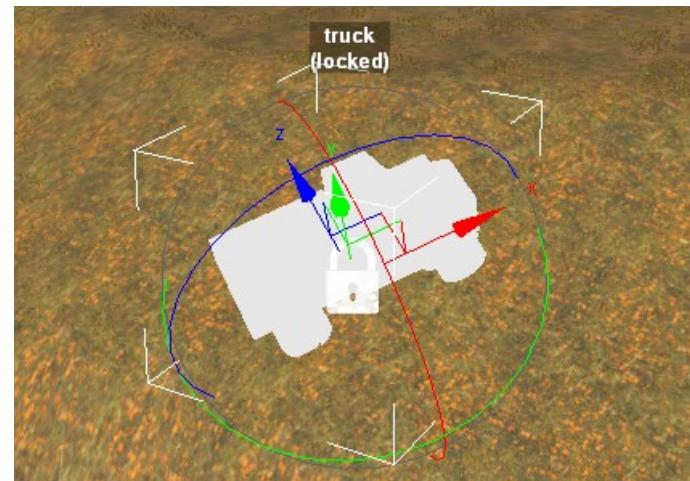
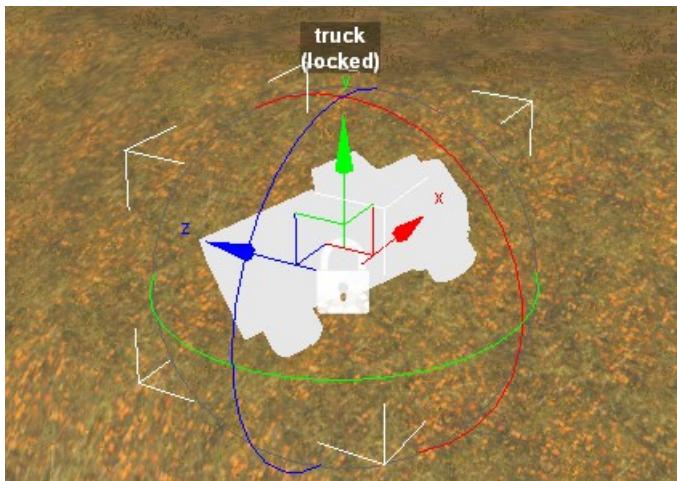


When the `Land` property is `False`, the truck can be moved and rotated freely. It can be raised into the air, buried in the ground, and tilted to any orientation. In most cases, this looks unnatural and results in strange physics. But it can also be useful, e.g. to place a truck on a bridge, or to present the player with a flipped-over truck.



Local Transform

When moving and rotating a truck, it can be helpful to manipulate it using its local axes rather than the global axes. The **Local Transform** option in the toolbar can be toggled by clicking it or pressing **Ctrl+L**. When enabled, **Local Transform** sets the axes of the interface widget to match the current orientation of the truck. Using local axes generally makes setting the pitch and tilt of a truck much more intuitive.



When **Local Transform** is enabled, manipulating the rotation widget updates the local axes as soon as you let go of the widget. If you manually edit the rotation property values, however, the the local axes aren't immediately updated to match. To force the local axes to update, just move the camera a smidgen. When the Editor redraws the view, it updates the widget, including the new local axes.

Select Truck Components

Vehicle Type

Scene.(Terrain).Trucks.type: **Truck.Type:** asset selection

Specifies the type of truck or trailer.

Default: blank; vehicle does not appear in the game.

The primary property for any truck is its type. When a truck is selected, this property is the top one in the property panel. To select the truck type, click **[press]** for the **Edit** pseudo-property in the **Truck** group. This brings up a list of trucks from which you can choose your truck type.



Bug: The first time you click here, the Editor may lock up for 5 or more seconds as it creates icons for all possible trucks. Later uses may pause again if the Editor decides to check for any changes, but never as long as the first time.

Bug: The **Truck** group includes a **Clear** pseudo-property, but clicking [press] next to **Clear** does nothing. You can change the truck to a different type, but you cannot revert it to the blank truck type.

In the **Select Asset** dialog window, you can type in the white box near the top to filter the list of trucks as desired. More information about this dialog is on page 59.

Once a truck type has been selected, the type appears in the properties panel, and the ghost view of the truck in the main panel changes to reflect the selected truck type. The truck is also called by its type in the list of **Trucks** in the **Scene View**.

Bug: Although the truck silhouette updates immediately, the name of the truck type isn't displayed correctly in the main panel until you deselect the truck.

Note that the truck asset list includes trucks from all possible DLC even if you do not own that DLC. If a player tries to play a custom map with one of these trucks without owning the corresponding DLC, the truck will not appear. Keep this in mind when selecting trucks for your map and when advertising your map to others. Trucks

and truck skins (page 84) are the only features that require DLC ownership. All other features from DLC can be used freely in any map.

You can choose a different truck type by clicking next to **Edit** again and choosing a new truck. Choosing a new truck type discards the previous selection, but keeps all other properties including trailers and addons.

The Editor uses lowercase truck names that differ slightly from the names used in the game, but it's always obvious how each name in the Editor correspond with a truck name in the game.

In the truck selection dialog you can choose to select a trailer instead of a truck. If you select a trailer as the truck type (instead of putting it the separate Trailer property below), the trailer is parked by itself as if it is a truck. Cargo can be loaded on this trailer as described in the Add-ons section below. Trailers that were introduced in DLC can be freely used even without DLC ownership.

By the way, because wheels are a separate asset that are attached to the truck by the game, they are not shown in the images in the asset selection dialog. They also aren't shown on the silhouette of a truck in the main panel, although they are shown for trailers.

Identify Compatible Truck Parts in the Virtual Garage

Since each truck generally has many available parts, few of which are shared with other trucks, there are a huge number of part names to keep track of. Since each truck can only install a subset of parts, and only in certain combinations, the best way to find compatible truck parts is in the game itself.

The various truck parts and trailers have lowercase names in the Editor that differ from the names used in the normal game. However, when playing in your own custom map or in any proving grounds map, the **Garage** option opens a virtual garage that allows you to easily discover the lowercase names of compatible parts. For more information on the dev tools and the virtual garage, see page 41.

When selecting a truck part, the Editor does not check whether it is compatible with the truck or the other parts. If it is not compatible, the game simply leaves the part uninstalled when it loads the map.

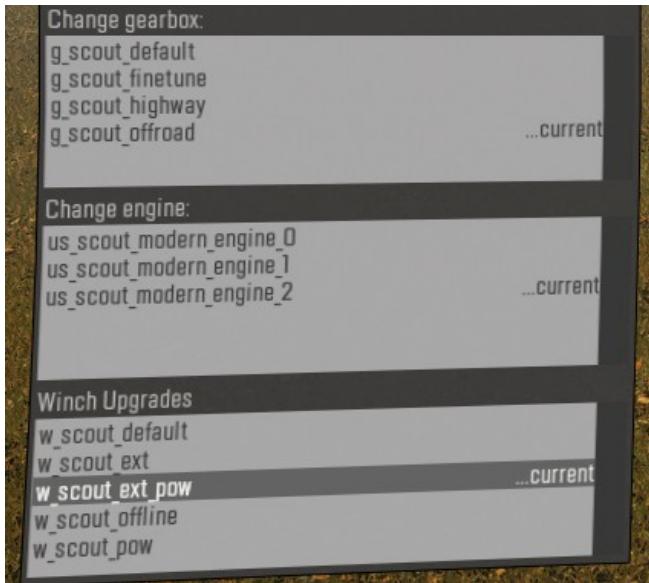
Engine, Gearbox, Winch, and Suspension

Scene.(Terrain).Trucks.type: Engine.Name: text
Scene.(Terrain).Trucks.type: Gearbox.Name: text
Scene.(Terrain).Trucks.type: WinchUpgrade.Name: text
Scene.(Terrain).Trucks.type: Suspension.Name: text

Specifies the engine, gearbox, winch, and suspension add-on types.

Default: blank; if **Name** is blank or invalid, the truck uses a default component.

Enter the name of a compatible truck part into the corresponding property value for that part to be installed on the truck.



<input type="checkbox"/> Engine	
Name	us_scout_modern_en...
<input type="checkbox"/> Gearbox	
Name	g_scout_offroad
<input type="checkbox"/> WinchUpgrade	
Name	w_scout_ext_pow

Wheels

Scene.(Terrain).Trucks.type: Wheels.Type: text

Specifies the wheel type.

Default: blank; if **Type** is blank or invalid, the truck uses its default wheels.

Scene.(Terrain).Trucks.type: Rim.Type: text

Specifies the rim type.

Default: blank; if **Type** is blank or invalid, the truck uses the default rim that is compatible with the wheels.

Scene.(Terrain).Trucks.type: Tire.Type: text

Specifies the tire type.

Default: blank; if **Type** is blank or invalid, the truck uses the default tire that is compatible with the wheels.

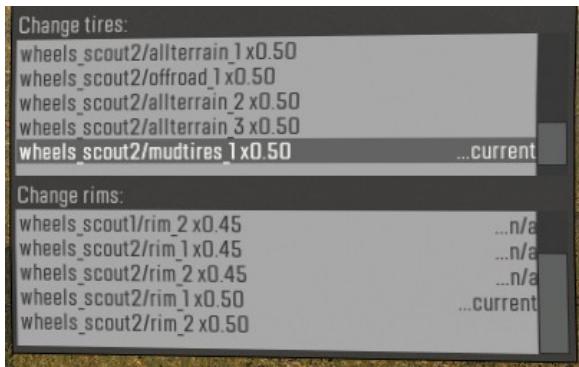
Scene.(Terrain).Trucks.type: Scale.Type: numeric, in meters

Specifies the wheel diameter.

Default: 0; if **Type** is 0 or invalid, the truck uses the nearest size that is compatible with the rim and tire types.

Type values for Wheels, Rim, Tire, and Scale can be determined from the tire and rim names shown in the virtual garage.

- The Wheels.Type value is the text in the virtual garage prior to the / in the name of the tire and rim. (It is always the same for both.)
- The Rim.Type value is the text after the / and before the x in the name of the rim.
- The Tire.Type value is the text after the / and before the x in the name of the tire.
- The Scale.Type value is the number after the x in the name of the tire and rim. (It is always the same for both.)



Wheels	
Type	wheels_scout2
Rim	rim_1
Tire	mudtires_1
Scale	0.500000

Color

Scene.(Terrain).Trucks.type: Customization.PresetId: integer

Specifies the color scheme for the truck.

Default: -1; if PresetId is -1 or invalid, the truck uses its default color scheme.

You can test color schemes in the Customization section of the virtual garage. The PresetId value is the number after the word Preset in the virtual garage.

Note that you can deselect all choices in the virtual garage to return to the default color scheme, which is different than all presets. The default PresetId property value of -1 chooses this color scheme.

The standard color schemes are numbered 0 – 31. Many of these are a single solid color, but some are multi-toned. Experiment in the virtual garage to find a pleasing scheme.

Special skins for trucks are numbered 32 and higher. Note that they are typically available only with DLC installed, and only for certain trucks.



If you set a truck's color scheme to use a DLC skin that the player doesn't own, the truck uses its default color scheme instead. Unless your objective descriptions rely on the truck's color to distinguish it, this shouldn't be a big deal.

Trailer

Scene.(Terrain).Trucks.type: Trailer.Type: asset selection

Specifies the type of trailer to attach to the truck.

Default: blank; trailer is not attached in the game.

Selecting a trailer is performed much like selecting a truck, this time using the pseudo-properties in the [Trailer](#) section.

Bug: The game cannot initialize a truck with an attached trailer, so this property is useless.

If you select a trailer that is not compatible with the truck type, the Editor simply records the choice without any error notification. Make sure to find a compatible trailer in the virtual garage.

The trailer is not drawn in the main panel. You have to use your intuition about its length to ensure that it doesn't overlap any other objects.

Add-ons

Scene.(Terrain).Trucks.type: Addons.Type: multiple asset selection

Specifies the add-ons to attach to the vehicle.

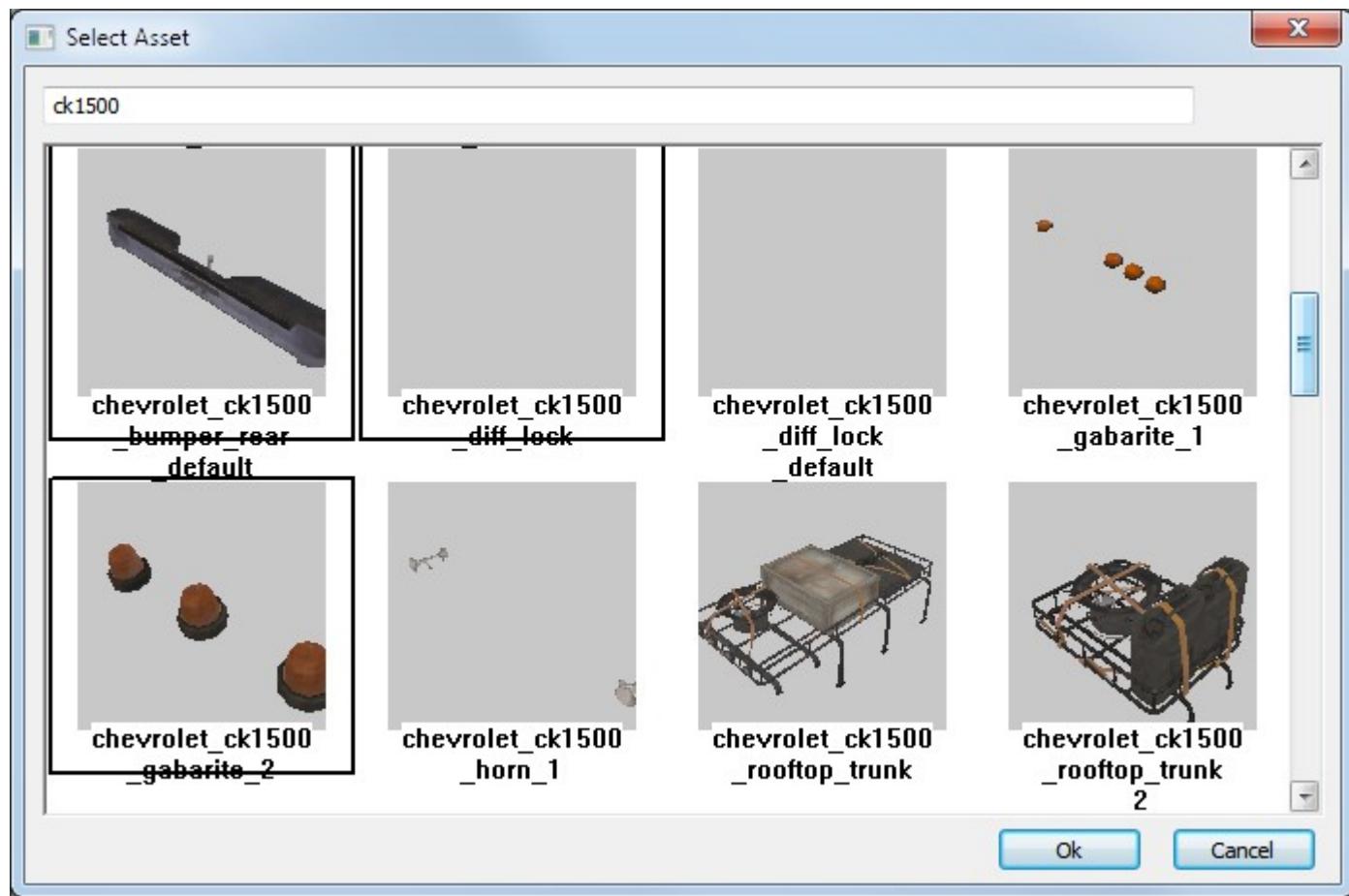
Default: blank; only default add-ons are installed.

Selecting add-ons is performed using the pseudo-properties in the **Addons** section. Unlike for trucks and trailers, however, this dialog box allows you to select multiple items before clicking **OK**. Clicking an add-on in the dialog once selects it. Clicking it again deselects it. Selected add-ons are indicated with a black border.

Bug: The first time you open the add-on assets, the Editor may lock up for **30** or more seconds as it creates icons for all of the many add-ons. Later uses may pause again if the Editor decides to check for any changes, but never as long as the first time.

You can filter the list of add-ons by typing in the blank panel near the top of the dialog box. Only add-ons that include your text somewhere in the name are shown in the dialog box.

Bug: When you change the filter text, the Editor discards any prior assets you selected in the window.



You may have noticed that the pseudo-property name for add-ons is **Add** instead of **Edit**. Once some add-ons are selected and confirmed with **OK**, click **[press]** again to **Add** to add more items to the list.

Unfortunately, there is no way to selectively remove add-ons from the list. Instead, click **[press]** next to **Clear**, then re-add the desired add-ons.

The Editor does not check the compatibility of add-ons. The game quietly ignores any incompatible add-ons and does not attach them to the truck. Use the virtual garage to find the names of add-ons that are compatible with the truck and can be used together. The order of compatible add-ons doesn't matter, as long as the necessary prerequisites are somewhere in the list.

Add-ons are not drawn on the truck in the main panel.

The list of addons may be longer than fits in the properties panel. Hover your mouse over the list of add-ons to get the full list in hover text.



Default Add-ons

Scene.(Terrain).Trucks.type: InstallDefaultAddons: True/False

Specifies whether to install non-required default add-ons.

Default: **True**; all default add-ons are installed where there is no conflicting part.

Most default add-ons are part of a required group. It is possible to uninstall required add-ons in the virtual garage (e.g. to leave a truck without any exhaust pipes), but the game always installs (or re-installs) the necessary defaults when the map is loaded. E.g. if no exhaust is included in the add-ons list, then the default exhaust is installed regardless of the **InstallDefaultAddons** property.

However, a few trucks have add-ons that are default but not required. If **InstallDefaultAddons** is **False**, then these non-required default add-ons are not installed. E.g. the default spare wheel is not installed on the KHAN 39 Marshall.

Cargo

Cargo can be loaded onto the truck as an "add-on". All cargo add-ons can be found in the addon dialog by filtering for the word "cargo". Note that in a few cases (particularly for logs), the cargo is shaped differently for a particular truck and so has a unique name. Be sure to pick the right add-on for your truck. Unfortunately, the dev tools cannot help test the compatibility of cargo.

Warning: Unlike other add-ons where order doesn't matter, the appropriate truck bed for the cargo must appear in the add-on list before the cargo. This means that typically you must add the truck bed first, confirm and close the dialog box, and then re-open the dialog box to add the cargo. If you select multiple add-on types from one invocation of the dialog box, they are added to the list in alphabetical order, which likely isn't what you want.

To load multiple copies of the same cargo on a truck, add one copy, exit the asset dialog, then add another copy. Each copy is added to the list of add-ons, and the game loads all the cargo that fits onto the truck. In the case of cargo of different types, cargo earlier in the list is loaded onto the truck closer to the front.

Truck ID

Scene.(Terrain).Trucks.type: Id: text

Specifies a unique identifier for this vehicle for use by objectives.

Default: blank; objectives cannot refer to this vehicle.

The **Id** property is optional. If filled, it should provide a unique identifier for this exact vehicle. This ID allows the vehicle to be linked to an objective (page 313).

Another vehicle should not have the same **Id** value in the map or region. The game triggers a ModMapError (page 287) if it finds a duplicate truck ID.

For safety, I recommend that the truck ID be limited to a combination of lowercase **a – z**, uppercase **A – Z**, **0 – 9**, and **_**. It should not include other characters, including spaces.

Bug: A truck loses its ID if the player ever drives it into a garage or recovers the truck to a garage.

A truck keeps its ID if the player recovers the truck to a recovery zone or if the player takes the truck through a gateway.

Starting Damage and Fuel

Scene.(Terrain).Trucks.type: Damage %: numeric slider from 0 – 100

Specifies the initial damage to the truck's components as a percentage.

Default: 0.

Each component is damaged by the same percentage, including wheels.

Any repair add-ons or repair trailer always start out full of parts and are unaffected by the **Damage %**. The **Damage %** property has no effect on a trailer since it has no components that can take damage.

Scene.(Terrain).Trucks.type: Visual damage %: numeric slider from 0 – 100

Specifies the initial damage to the vehicle's body panels as a percentage.

Default: 0.

Visual damage is only visual; it has no effect on the operation of the truck. Visual damage is evenly distributed across the truck body. Even 5% visual damage can look rather significant (shown below), so don't overdo it.



The Visual damage % property also applies to a trailer. Note that a trailer cannot be repaired.

Scene.(Terrain).Trucks.type: Fuel %: numeric slider from 0 – 100

Specifies the initial fuel in the vehicle's internal fuel tanks as a percentage.

Default: 100.

Any refueling add-ons always start out with full fuel and are unaffected by the Fuel %. If a trailer is placed on its own, the Fuel % property specifies the initial fuel level of any refueling tank on the trailer.

Locked and Unlocked Vehicles

Scene.(Terrain).Trucks.type: Locked: True/False

Specifies whether a truck or trailer is locked at the start of the map.

Default: True.

When the Locked property is True, the Editor adds a (locked) suffix to its name and draws a padlock icon on its silhouette in the main panel. A vehicle starts locked by default.

A locked vehicle has three attributes of interest:

- The player cannot drive a locked truck.
- When the player discovers a locked vehicle, she gains an experience bonus.
- When the player discovers a locked vehicle, it becomes available for sale in the garage or trailer store (if it wasn't already available).

See the Discovery of Vehicles section on page 44 for more information about the discovery process.

Reserved Vehicles

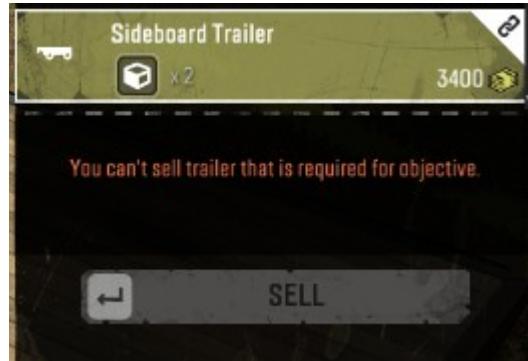
A truck or trailer can be reserved for use in a vehicle delivery goal (page 350), or a truck can be reserved for a truck repair goal (page 355). An objective may reserve a vehicle permanently, or the vehicle may be awarded to the player upon completion of the objective.

Bug: To avoid bugs, a reserved truck should (almost) always start as locked. See page 353 for details.

A player cannot “discover” a truck while it is reserved, nor can she enter or otherwise drive it. A player can repair or refuel a reserved truck and can tow it with a winch from another truck. If the player attempts to change to a reserved truck via the functions menu, the game displays a helpful error message based on the objective that is reserving it.



A player cannot “discover” a trailer while it is reserved, but she can otherwise attach it and use it a regular trailer (e.g. pack and unpack goods). The only thing a player can’t do with a reserved trailer is sell it.



The Active Truck

`Scene.(Terrain).Trucks.type: Active: True/False`

Specifies whether the player starts the map in the truck.

Default: `False`.

The `Active` property determines which truck the player starts in. If you set the `Active` property to `True` for a truck, the Editor automatically sets all other trucks on the map to `False` so that there is only one active truck. When the `Active` property is `True`, the Editor adds an `(active)` suffix to its name and draws an arrow pointing down at it. The `Locked` property is ignored when the `Active` property is `True`.

Bug: When an active truck is selected, the Editor draws a 32-meter gray disk around it in the main panel. This is a holdover from the MudRunner Editor, and it has no useful meaning in SnowRunner.

Although the Editor attempts to allow only one truck to be active, it is possible to end up with multiple active trucks, e.g. by duplicating an active truck. The game triggers a `ModMapError` (page 287) if it finds more than one active truck.

If no truck is marked as active, then the player spawns without a truck and with no ability to move the camera or enter a truck. If the `Tools` menu is enabled in the game, you can use it to jump into an existing truck. See the Dev Tools section on page 39 for details.

If the active “truck” is actually a trailer, the player spawns in the trailer. From here, the player can move the camera or enter a different truck, but the trailer cannot be driven. (No engine or drivetrain.)

If the active truck is reserved by an objective, the player still starts in that truck. I’m not going to try to document the behavior in this case. Don’t do it.

Hand Edit the XML

The properties for all of your trucks (and other data) is saved in your map’s XML file in the `prebuild` directory. Editing the XML by hand is useful for tasks that are difficult or impossible to do in the SnowRunner Editor. For example, you can rearrange the trucks so that the active truck is first. Or you can use a text editor’s find and replace function to change every `Locked` property to `False` for testing purposes.

More detail about hand-editing the map XML files appears in Appendix B beginning page 420.

Terrain Height

Clicking any of the items under **(Geometry)** in the **Scene View** allows you to edit the terrain. Terrain features don't have named properties, and they are not saved in the map's XML file. Instead, the terrain is divided up into small squares, and data for each square is recorded as a pixel in a bitmap. The "brightness" for each pixel in the bitmap determines the strength of a corresponding feature in the terrain. For example, if the terrain height increases smoothly from minimum height on the left to full height on the right, it is recorded in the height bitmap as an image that fades from black on the left to white on the right.

This concept of terrain features as bitmaps is reflected in the metaphor for editing those features: painting the terrain using a brush.

There are many features within the **(Geometry)** category, but this section is devoted to the **(Geometry)** feature itself.

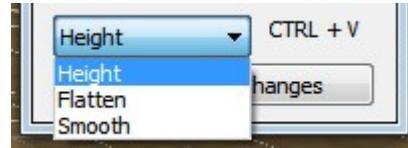
Click **(Geometry)** under **(Terrain)** in the **Scene View** to edit the terrain height. A **Brush** dialog pops up that allows you to edit your brush parameters. Then paint on the map by holding down the right mouse button while moving the mouse (right-click and drag).

If you are satisfied with the results, left click in the main panel or on something else in the **Scene View** to deselect the **(Terrain)** tool. This commits your changes and writes them to the underlying bitmap file. (No separate save step is necessary for terrain changes.)

On the other hand, if you don't like your changes, click **Undo Current Changes** in the **Brush** dialog to revert your changes without updating the underlying bitmap file. More information about undo of paint changes is on page 104.

Bug: If you left click on a non-brush feature in the Scene View, it cancels the brush painting tool, but does not select the clicked feature. On the other hand, if you left click a different brush feature, that feature is selected, and the brush immediately changes to the new type.

The **(Geometry)** brush has three modes: **Height**, **Flatten**, and **Smooth**. Each of these modes is described in the sections below.



Use Height Data from an External Source

The Editor can read the terrain height from a separate file, described on page 455.

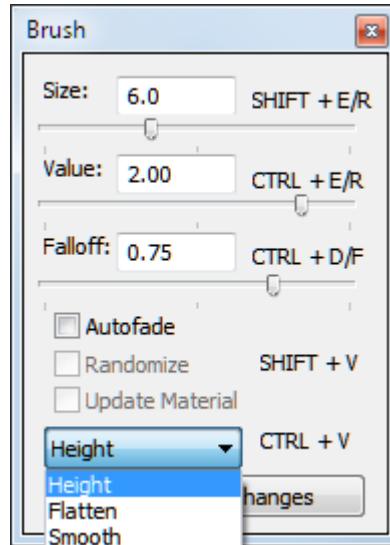
Bug: The **Copy Source Heightmap** item is directly under the mouse whenever you open the context menu on any **(Geometry)** feature. If you accidentally select this context menu item, it will destroy all height data on your map with no confirmation dialog or ability to revert the change.

Tip: Delete `prebuild/map_name/_height_source.png` immediately after creating a new map to prevent **Copy Source Heightmap** from doing anything. The default contents of this file are not useful for any purpose, anyway.

Some third-party tools allow you to generate terrain for a fictional place or extract terract height from real-world data. If you'd like to start your map with height data from an external source, see the section on editing the Editor's bitmap files on page 437 and the subsection for terrain height on page 454.

Height Brush Mode

The default **Height** mode in the **Brush** dialog increases or decreases the terrain height under the brush.

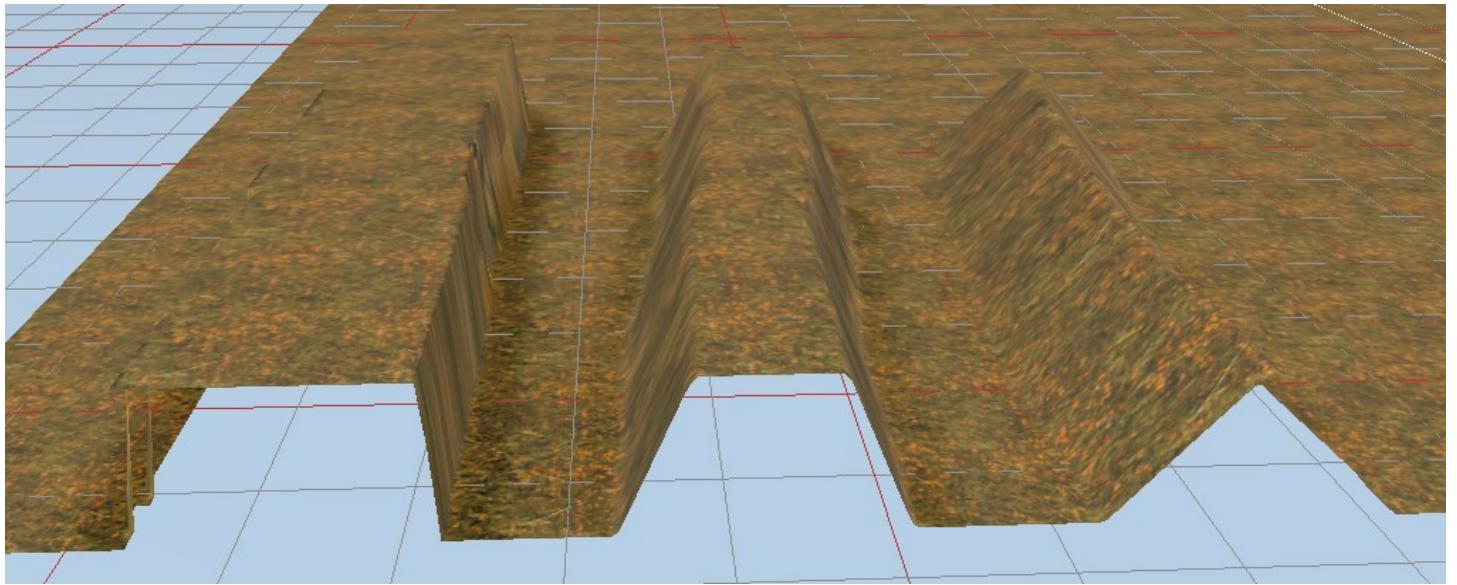


The **Size** parameter in the **Brush** dialog sets the radius of the brush in meters. The slider is approximately logarithmic, so you have fine control of small brush sizes and coarse control of large brush sizes.

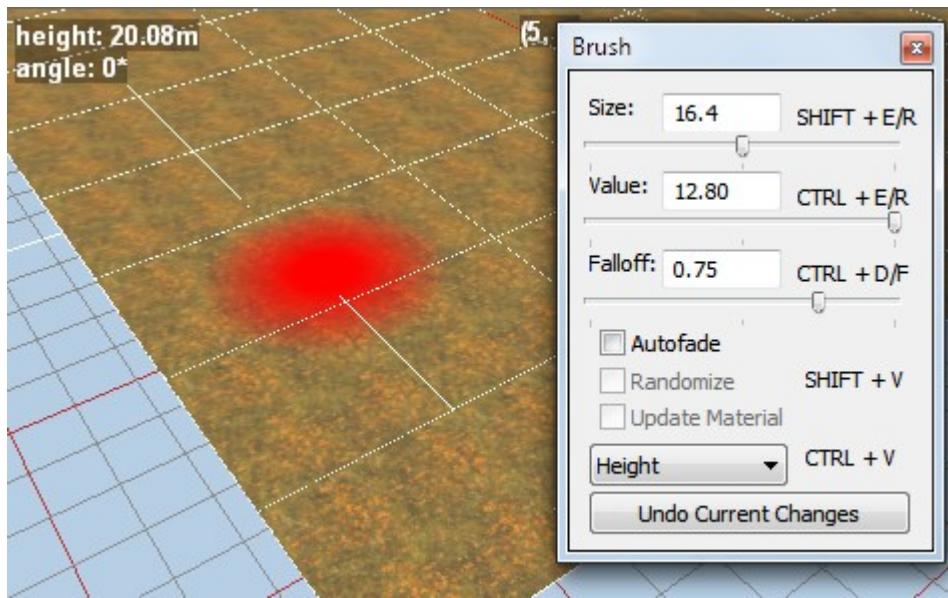
The **Value** parameter sets how much the terrain height changes at the center of the brush during a paint stroke. The value is measured in meters. Positive values increase the terrain height, and negative values decrease the terrain height.

The brush has an inner region in which the terrain height is changed by the full **Value** and an outer region where the change falls off to zero. The **Falloff** parameter determines the fraction of the brush radius that is dedicated to the outer brush region, where the change falls off. With a **Falloff** of 0.00, there is no outer region, and the entire brush paints the terrain with the full change set by **Value**. With a Falloff of 1.00, falloff is spread across the entire brush, so only the point at the center paints the terrain with the full change. In most cases, some intermediate value is desirable.

The figure below shows the effect of different **Falloff** values, with 0.00 on the left, 0.50 in the middle, and 1.00 on the right.



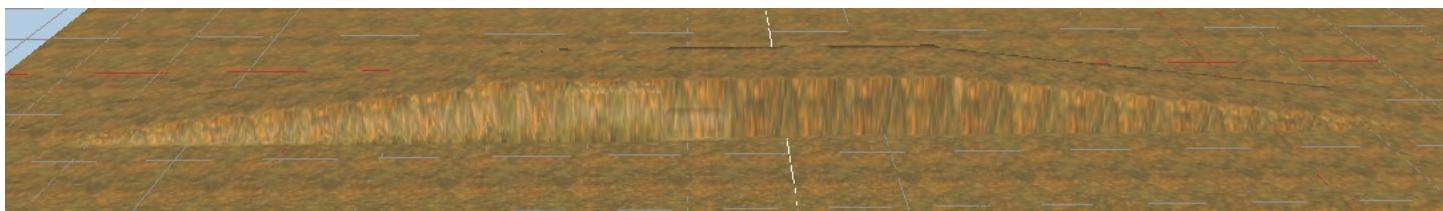
While in **Height** mode, a disc is drawn on the terrain under the mouse to preview the brush parameters. A red disc is drawn for a large positive **Value**, and a black disc is drawn for a very negative **Value**. The brush **Size** and **Falloff** are also represented on the disc. This provides a quick sanity check that your parameters make sense before you start to paint.



The disc is less strongly colored at reduced values. Unfortunately, when painting with a small **Value**, the disc can be nearly invisible.

Autofade

The **Autofade** checkbox is mostly useful for painting tire tracks in mud, not for painting height, but you can experiment with it here. With **Autofade** enabled, when you right click and begin dragging in some direction, the Editor adds a tail to your brushstroke that fades in as if you had slowly lowered the brush onto the terrain while approaching the spot where you actually clicked. And when you let go of the right mouse button, the Editor adds a tail that fades out as if you had slowly lifted the brush while continuing to drag in the most recent direction. The length of the tail is about 10× the width of the brush.



I find Autofade very difficult to control, but your mileage may vary.

Autofade is available for **(Geometry)** only when using the **Height** mode.

Flatten Brush Mode

Once you've painted some height changes onto the terrain, it is very difficult to get it flat again using the **Height** mode. To easily flatten terrain, change the **Brush** dialog to the **Flatten** mode. In this mode, the terrain at the point where you start your brush stroke determines the target height. Painting in this mode shifts the height of the painted terrain toward the target height.

The Editor keeps track of the most recent brush parameters you used in each mode. When you switch modes, the parameter values are reset to whatever you last used in that mode. It does not save these parameters across Editor sessions, however.

The **Size** parameter for **Flatten** mode still has the same purpose; it sets the radius of the paint brush.

The **Value** parameter is now a value between 0 and 1, and it determines how quickly the painted terrain should change toward the target height. With a **Value** of 1.00, the terrain at the center of the brush jumps fully to the target height. With a lesser **Value**, the terrain under the brush shifts only the specified fraction toward the target height. But this is deceptive! For every pixel that you drag the brush, the Editor re-applies the flatten effect across the entire brush. For example, with a **Value** of 0.50, as you start dragging the brush, the terrain near the center of the brush, shifts halfway to the target value, then shifts half of the remaining amount, etc. This is very different than the **Height** mode, where the brush effect is re-applied only if you start a new stroke.

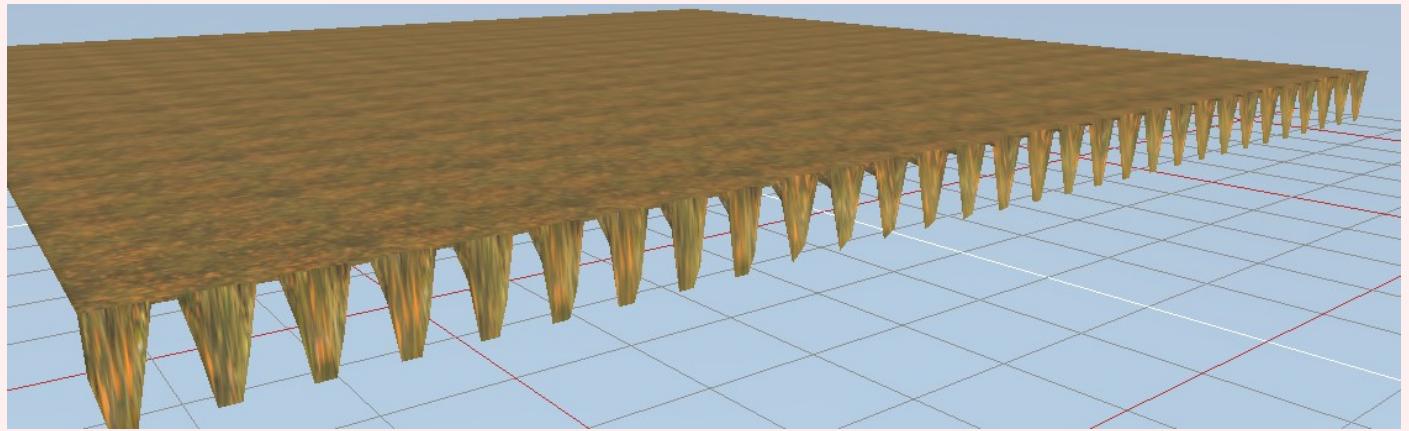
The **Falloff** parameter still has the same purpose; it sets how much of the brush is dedicated to the falloff region.

If you just want everything to be perfectly flat, it is fine to set the **Value** to 1.00 and the **Falloff** to 0.00. However, if you want a more subtle effect (e.g. at the edges of the flattened region), I recommend a reduced **Value** and a higher **Falloff**. For example, a **Value** of 0.50 and a **Falloff** of 0.80 allows me to stroke overlapping loops with my brush that result in perfectly flat terrain where I've been looping and smooth slopes at the edges.

In the **Flatten** mode, the brush preview disc is drawn in red for a **Value** above 0.50, black for a **Value** below 0.50.

Bug: Normally the brush preview disc increases in intensity as the **Value** changes away from some neutral value that has no effect when painted. In **Flatten** mode, however, the brush preview disc treats 0.5 as the neutral **Value** when 0.0 is the actual neutral **Value**. This means that the preview disc is invisible near a **Value** of 0.50 even though this **Value** causes a large effect.

Bug: A freshly created map is mostly at a single middling height, but one edge often has some weird height drops. You can use **Flatten** mode to fix it.



Smooth Brush Mode

In the **Smooth** mode, the target height at each point under the brush is a weighted average of the heights around the edge of the brush. E.g. the target height in the middle of the brush is exactly the average of the heights around the edge of the brush, but the weighting function sets the target height near the left side of the brush closer to the height of the left edge than the right edge. In other words, **Smooth** mode tries to create a smooth slope.

The **Size** parameter still has the same purpose; it sets the radius of the paint brush.

The **Value** parameter indicates the magnitude of the effect. In this case, a **Value** of 1.00 does not fully smooth the terrain as soon as you click, but it takes very little dragging of the brush to achieve a completely smooth slope.

The **Falloff** parameter again specifies the falloff characteristics of the brush. In this mode, however, falloff is counter-productive. The results can be very non-smooth if terrain is fully smoothed at the center of the brush while the terrain at the edges stays close to its original heights.

Because of the way the **Smooth** brush works, it is perfectly reasonable to set the **Value** to 1.00 and the **Falloff** to 0.00.

Bug: Normally the brush preview disc increases in intensity as the **Value** changes away from some neutral value that has no effect when painted. In **Smooth** mode, however, the brush preview disc treats 0.5 as the neutral **Value** when 0.0 is the actual neutral **Value**. This means that the preview disc is invisible near a **Value** of 0.50 even though this **Value** causes a large effect.

Brush Keyboard Shortcuts

Keyboard shortcuts are listed on the **Brush** dialog box. E.g. **Shift+E** decreases the **Size** by a step, and **Shift+R** increases it by a step. Each step is 5% of the slider length, but remember that the slider has an exponential relationship to the **Size** value.

Shift+V is for the Randomize checkbox, which is disabled for all **(Geometry)** brush modes.

Ctrl+V cycles among the **(Geometry)** brush modes: **Height**, **Flatten**, and **Smooth**.

Ctrl+Z is an unlisted undo shortcut that is different than the **Undo Current Changes** button. See page 104 for details.

The keyboard shortcuts can be used not just when the **Brush** dialog box has focus but also when the main panel or the **Scene View** have focus. In other words, they can be used essentially any time you are using a brush. If you learn and use these shortcuts, your painting speed will be much faster with all brushes.

Automatic Terrain Rebuild

By default, the terrain is rebuilt whenever you commit a change to a **(Geometry)** feature. Only the terrain blocks that were touched are rebuilt, and the rebuild uses the same shortcuts as **Rebuild Selection** (page 65). The main effects of this quick rebuild are to update shadows but also to remove grass from the affected terrain blocks.

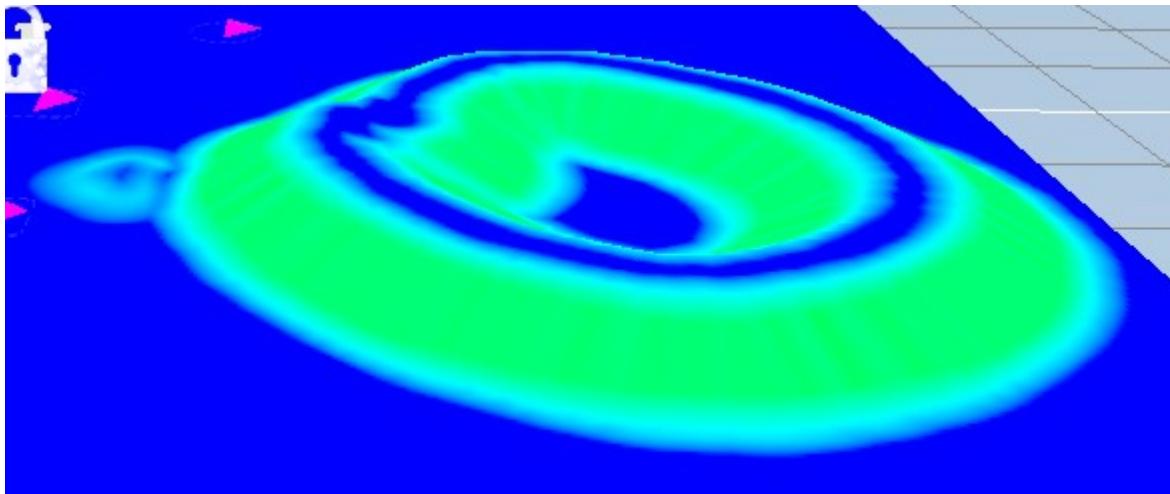
You can disable this automatic terrain rebuild by disabling the **Enable autorebuild terrain** toggle button on the toolbar (page 256).

Height Information

Whenever the mouse is over the terrain in the main panel, the Editor annotates the current terrain height and angle in the upper left corner. The angle is 0° for flat terrain, increasing to near 90° for a cliff.

height: 12.95m
angle: 28*

Alternatively, you can get a quick read of the terrain angles at all points at once by enabling [Show HeightMap Angles](#) in the toolbar (page 256).



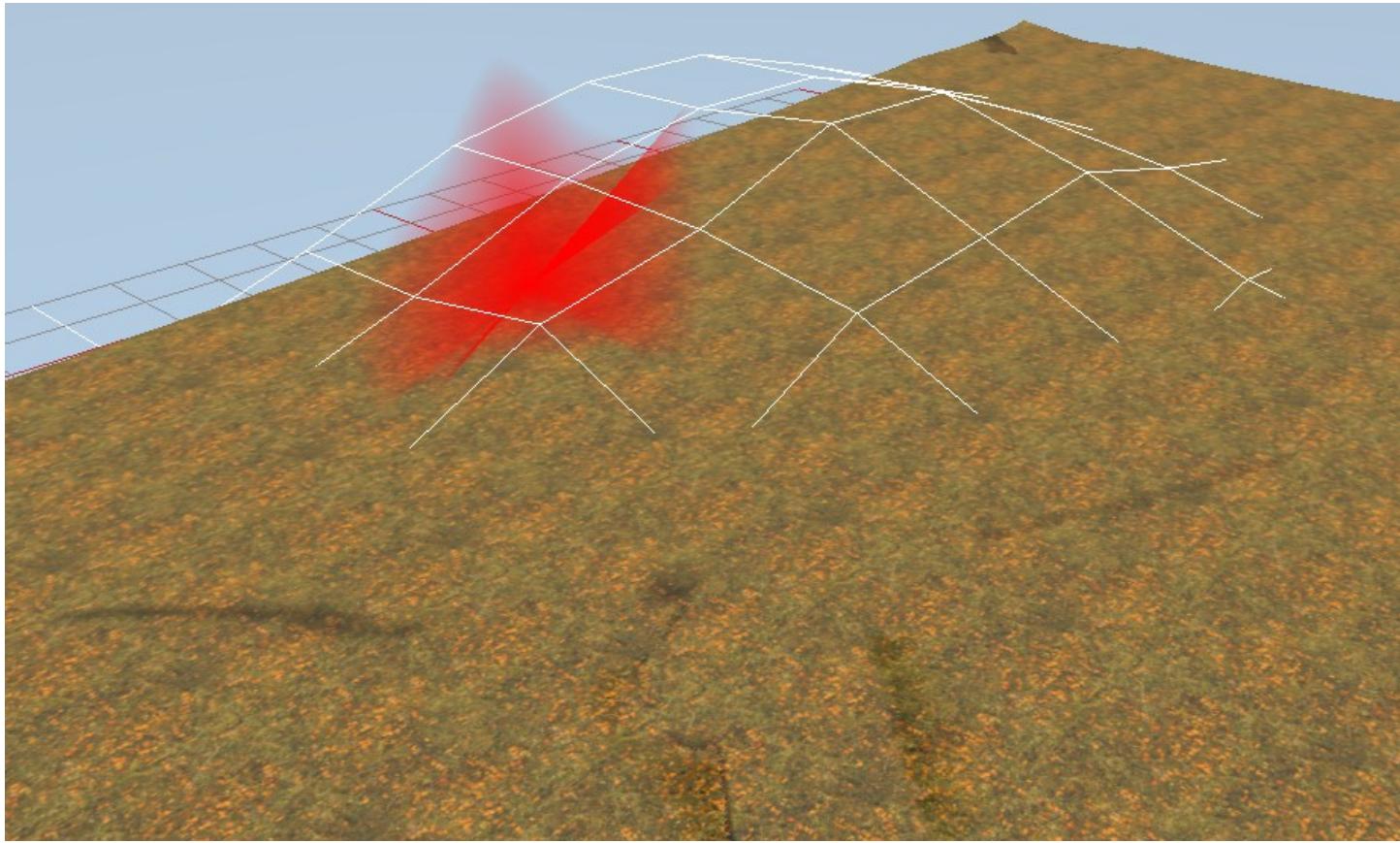
The heightmap colors correspond as follows:

- dark blue: 0°
- cyan: $1^\circ - 24^\circ$
- green: $25^\circ - 34^\circ$
- yellow: $35^\circ - 44^\circ$
- orange: $45^\circ - 59^\circ$
- red: $60^\circ - 90^\circ$

In ideal conditions in SnowRunner, a well-prepared truck can climb a smooth slope up to about 60° . For comparison, according to the Guinness World Records, the steepest street in the world is 19° . So anything green through orange is definitely off-roading.

Deceptive Height Cues

While you are editing the terrain height, the Editor continuously redraws the terrain with its new height. However, it doesn't change the way it draws everything else until you commit your changes and rebuild the terrain. For example, I have (almost) completely flattened the terrain in the screenshot below, but it doesn't look flat at all.



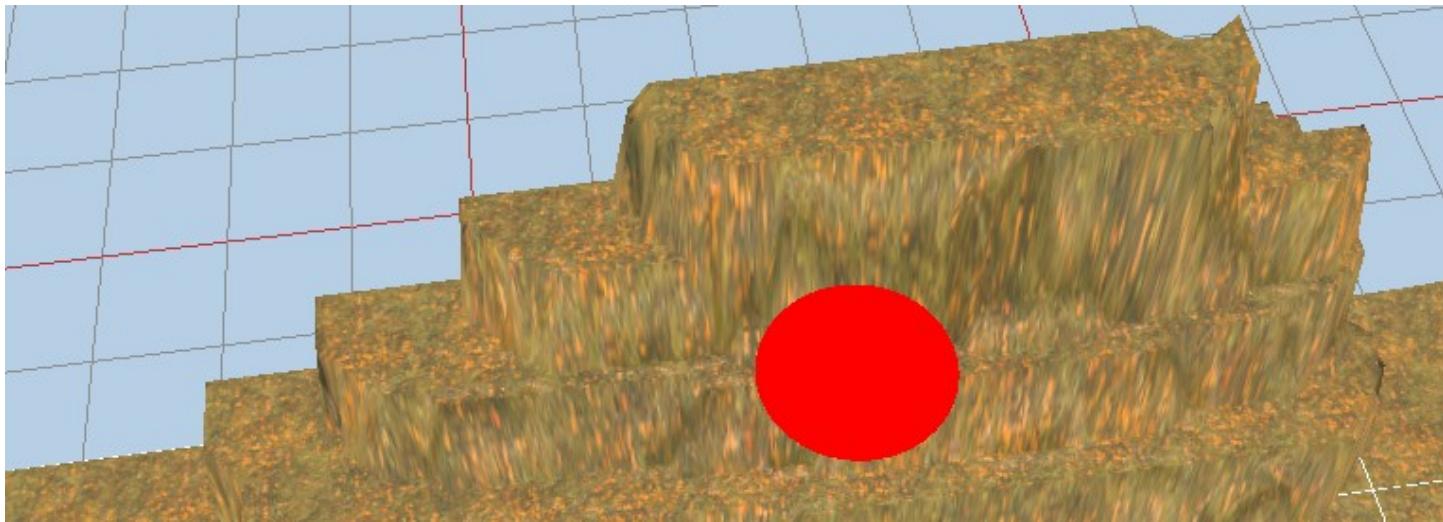
The deceptive cues above are as follows:

- The terrain is still colored with its original shadows, showing where steep slopes used to exist.
- The terrain blocks (the white wireframe mesh) remain at their original height.
- The brush preview disc is warped to follow the contours of the original terrain.

Another deceptive cue not shown here is the height and angle displayed in the upper left of the main panel, which continue to show the values for the original terrain.

The only cue that isn't deceptive is the edges of the map. The map edges are straight, which implies that the terrain must be flat (or at least smooth). A couple of hitches near the corner imply that I didn't quite get it flat there.

The brush preview disc is particularly deceptive because its horizontal position is determined by where I'm pointing on the original terrain, which can be very different from the horizontal position where I'm pointing in the edited terrain. In the screenshot below, I've built up the terrain from a flat plane. The brush preview disc appears to be pointed at the middle of the slope, but it actually represents a position right at the edge of the map. (Compare the disc position to the virtual line connecting the flat edges of the terrain at the left and right.) Drawing at this position actually draws at the very edge of the map at the top of the slope.



Rotating the camera and watching the motion of the terrain can help a little. But really the best solution is to commit your big changes, see how they look, and then make smaller changes to fix up the details.

Navigate While Painting

While right-click-dragging, it can be frustrating to run into the edge of the main panel. If the mouse leaves the main panel and re-enters somewhere else, the brush paints a straight stroke from where it left to where it re-entered.

To avoid this, you can navigate while a brush stroke is in progress. While continuing to hold down the right mouse button, you can then press the control key and the left mouse button to switch to dragging the terrain around. If you then release the left mouse button while continuing to hold the right button down, you can resume your paint stroke at the same position on the terrain, but a new position within the main panel.

This trick only works well with **Ctrl** + left-drag because the mouse moves exactly with the terrain. With a regular left-drag or with **Shift** + left-drag, the terrain moves away from the mouse, so you end up with a stray brush stroke when you release the mouse.

Restore a Previous Terrain State

Click [Undo Current Changes](#) in the [Brush](#) dialog to revert your changes without updating the underlying bitmap file. Changes are reverted across all brush modes back to the point when you first entered the ([Geometry](#)) brush. However, the Editor doesn't keep any further backups of previous versions of the bitmap files.

I recommend that if you're happy with your terrain, manually make a copy of your map source files to an archival location (page 33). Then, if you accidentally mess something up, you can simply restore one or all of the archived terrain bitmap files back into your [Media](#) directory.

Unlike when you update the map's XML file, the SnowRunner Editor won't automatically notice when you change bitmap files outside the Editor. Rebuild the terrain (page 65) to redraw the map from the files.

Undo/Redo

The Editor has two separate undo facilities: one for object and node changes, and one for painting.

See also the backup system for XML files (page 35).

Undo of Object and Node Changes

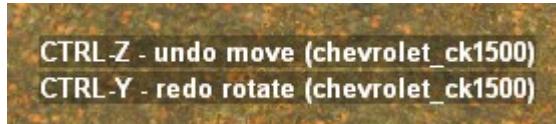
The Editor has a limited undo facility for object and node changes with a multi-entry history. Only the following changes can be undone:

- Move, rotate, or scale when using the main panel widget (but not when the properties are edited directly).
- Delete a feature or path node.

Bug: Undo of delete is buggy for most features. The Editor may crash, it may restore a zombie object that doesn't actually exist, or it may do something even more subtly wrong. Undo specifically of the deletion of a model, plant, or node **appears** to be stable, but I can't say that I trust it.

When undo is possible, the Editor indicates what action can be undone at the bottom of the main panel. To undo, press **Ctrl+Z**.

After one or more undo actions, the Editor indicates that redo is possible at the bottom of the main panel. To redo, press **Ctrl+Y**.



Bug: An undo/redo of an object's manipulation may not correctly display the change until the object is selected.

Bug: The undo history is retained through a rebuild of the terrain, but undo/redo does not actually work across the rebuild boundary.

Undo and redo can get confusing if you've done some actions that support undo mixed with other actions that don't support undo.

Undo and redo are definitely useful, e.g. when you accidentally drag the wrong widget handle. But it's hard to develop habits for when undo is appropriate and when it may result in bugs.

Undo of Paint Changes

While painting any bitmap feature, **Ctrl+Z** and **Ctrl+Y** operate on brush strokes. To undo a brush stroke, press **Ctrl+Z**. To redo a brush stroke, press **Ctrl+Y**.

Bug: While painting a bitmap feature, the main panel still shows a message that **Ctrl+Z** and **Ctrl+Y** operate on objects.

The undo history for brush strokes is 10 strokes. The history is lost whenever you change the brush mode (e.g. from **Height** to **Flatten**) or when you exit painting mode.

The brush dialog box also has an **Undo Current Changes** button. Click this button to revert all changes that were made since painting mode was entered. This includes changes across different brush modes.

Mud

Mud is tricky to get right. Expect to experiment a lot to get a feel for it.

Note that many of the terrain materials are already a little soft, including the **default** material. This means that if a truck drives over it repeatedly, it eventually cuts ruts into the terrain and loses traction. Adding mud makes the material even softer and more slippery.

Mud can be painted in four different ways:

- The **(Geometry) → (Mud)** brush in **Extrudes** mode paints mud that is hard to see.
- The **(Geometry) → (Mud)** brush in **Automatic** mode can paint wide swaths of visible mud.
- The **(Geometry) → (Mud)** brush in **Automatic** mode can also paint narrow muddy ruts that look like a truck drove through.
- The **(Geometry) → (QuickMud)** brush paints a wide area of mud that appears as if churned by many trucks.

Extrudes Mode

Expand the **(Geometry)** category in the **Scene View** and click **(Mud)** to start using the mud brush. The default mode is **Automatic**, but we'll start with the **Extrudes** mode.

The mud brush allows painting with relatively fine detail. For a particular **Size** value, a mud brush is $2.5 \times$ smaller than the size of any other terrain brush. I.e. the radius of the brush is the **Size** value $\times 40$ cm.

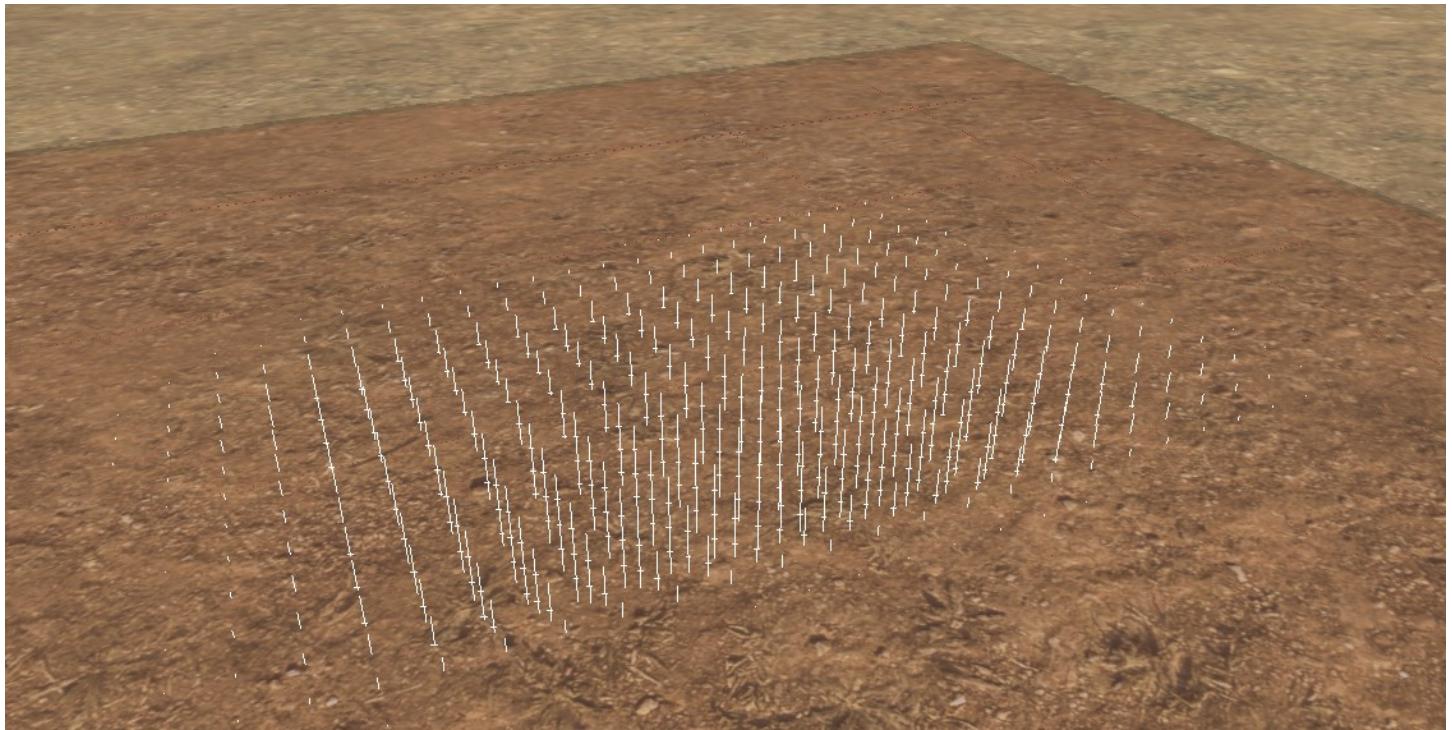
A **Value** greater than 0.50 increases the amount of mud, and a **Value** less than 0.50 decreases the amount of mud. When increasing mud in **Extrudes** mode, the brush preview disc is red. When decreasing mud, the brush preview disc is black.

Bug: Using 0.50 as the neutral **Value** is a holdover from SpinTires and MudRunner where all brushes used the same slider range. For brushes that can draw or erase paint, it would make more sense to have a range of -1.00 to $+1.00$ with a neutral value of 0.00 . But the neutral value of 0.50 is common to many of the brushes, so you'll eventually get used to it.

While the mud brush is active, the Editor indicates the presence of mud with two visual cues:

- A brown tint covers a large block of terrain, indicating that mud is present.
- Thin white lines extend down from the terrain surface where mud was painted. The length of each line represents the softness of the mud in that area. A tick mark is placed on the line for every 25% of the

maximum mud softness that is present. E.g. if a line has one tick mark near the bottom, the mud is just over 25% of the maximum possible softness.

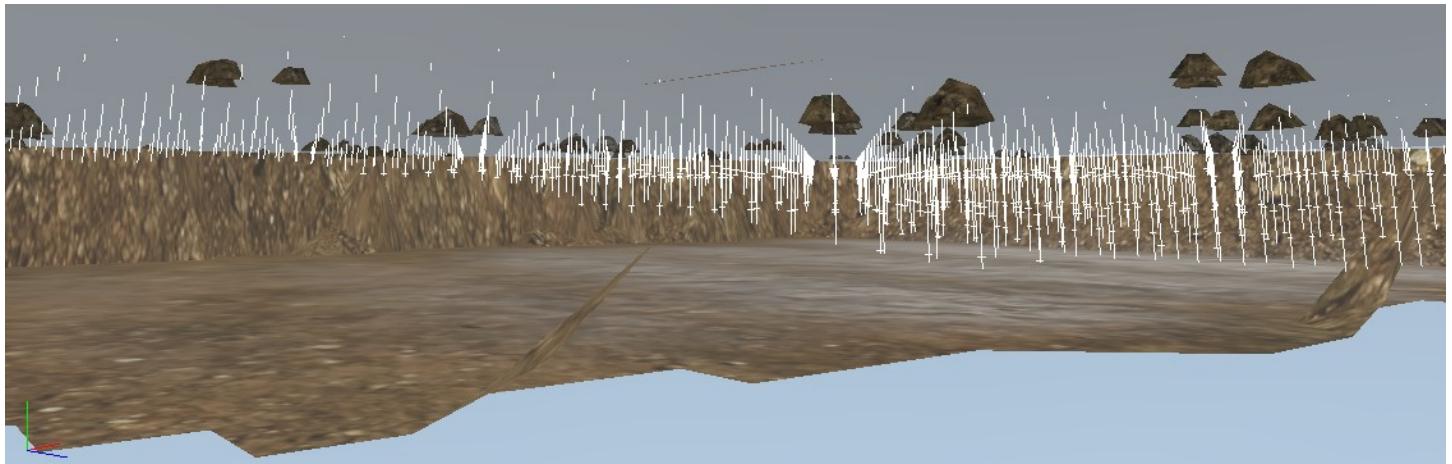


Bug: Mud that is maximum depth does not show a fourth tick mark, only a full-length fourth segment.

When the mud brush is not active, mud painted in **Extrudes** mode is entirely invisible. Most mud should also include some areas painted with the **Automatic** mode to make it much more visible. **Automatic** mode is described on page 109.

Mud painted in **Extrudes** mode also has very little effect on truck behavior unless it is wet. Wetness is discussed in the next section.

While a mud brush is activated, you can move the Editor camera slightly underground to see a bit more about how mud is handled in the game. Wherever mud is present, a second terrain layer is added beneath the main surface. This second layer provides another surface for both visual rendering and to support the physics when the truck breaks through the main surface and starts churning up the mud.



Bug: The first time you paint a new section of mud, the main surface is temporarily erased and only the second, underground layer is rendered. If you exit the mud brush and re-select it, the main surface is drawn correctly.

This second terrain layer highlights an important point: although the thin white lines imply a mud depth, they really only indicate its softness. If a heavy truck persists in churning up the mud, it can always dig all the way to the second layer.



In fact, the game automatically creates a second layer wherever the truck starts churning up dirt. With enough persistence, it is possible to dig deep ruts even without mud.

Tip: Avoid sudden transitions between deep mud and no mud. Otherwise, when the truck hits the edge, it bounces unnaturally as it is suddenly forced up and over the solid dirt.

Extrudes to Wetness

The default for a new map is that mud is dry. Even the maximum amount of mud has very little effect on truck behavior unless it is also damp.

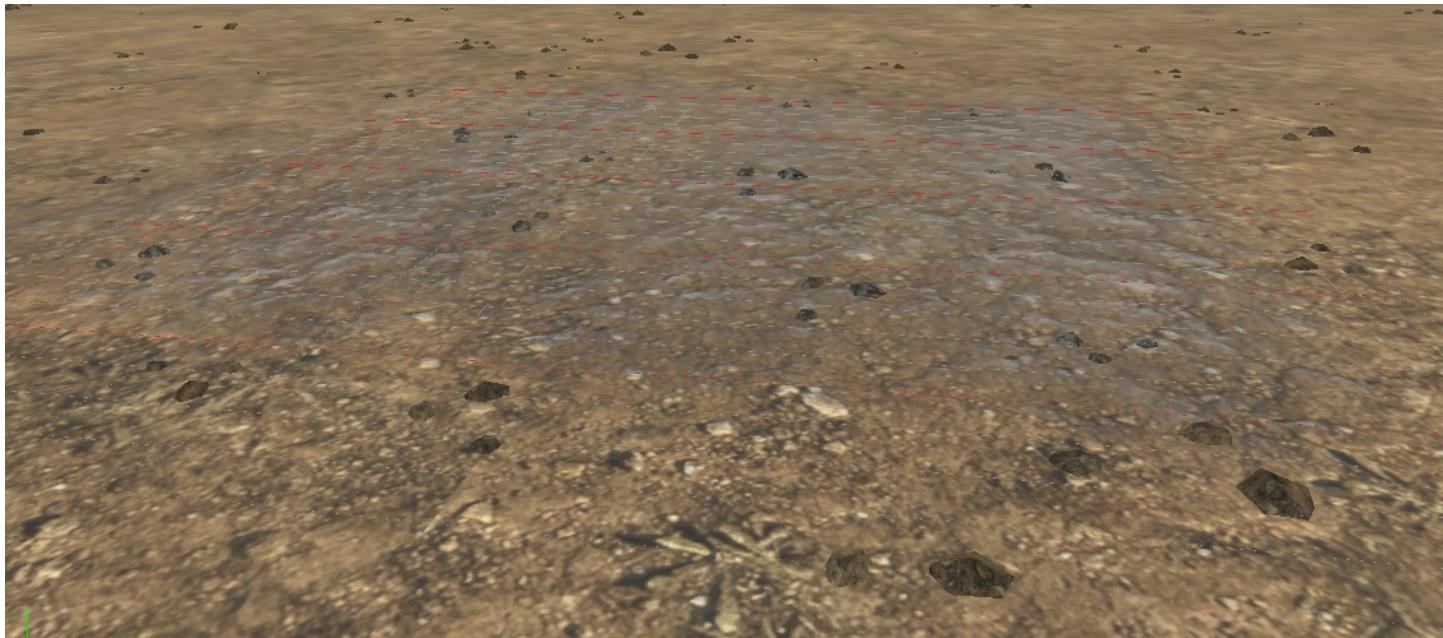
The terrain can be made automatically damp where mud exists with the [Extrudes to Wetness](#) property.

Scene.(Terrain): Extrudes to Wetness: numeric

Specifies the amount of wetness to add, depending on the depth of mud.

Default: 0.0.

The [Extrudes to Wetness](#) property causes wetness to be added to the terrain in proportion to the mud softness. Wetness provides a subtle visual indication that mud is present. It also makes the mud softer and more slippery.



Wetness can also be applied manually, as described on page 217. Wetness from the [Extrudes to Wetness](#) property is calculated dynamically when the terrain is rebuilt, so it doesn't touch the manual wetness bitmap.

The [Extrudes to Wetness](#) value is multiplied by the mud softness (as a fraction of the maximum mud softness) to get the derived wetness (as a fraction of the maximum wetness). The resulting calculated wetness is capped at 100% if necessary.

Tip: Saber recommends an **Extrudes to Wetness** value of 1.0:

- Mud with 25% softness (1 tick mark) is generally passable even by 2WD vehicles with highway tires.
- Mud with 50% softness (2 tick marks) is generally passable by AWD vehicles with offroad tires.
- Beyond that, only the toughest, most mud-capable trucks can find passage.

As long as **Enable autorebuild terrain** is enabled (page 256), new mud painted in **Extrudes** mode is rendered with the current **Extrudes to Wetness** value. But to apply the parameter to existing mud you must manually rebuild the terrain (page 65).

Scene.(Terrain): -threshold: numeric

Specifies the minimum mud depth before **Extrudes to Wetness** is applied.

Default: 0.0.

Mud that is less deep than the **-threshold** value (as a fraction of the maximum mud softness) has no wetness automatically applied. For mud that is deeper than the threshold, wetness is applied as usual.

Tip: Saber recommends a **-threshold** value of 0.20 for a winter environment and a smaller value such as 0.05 for a warmer environment.

Automatic Mode

The **(Mud)** brush also has an **Automatic** mode that is used to paint visually obvious mud. Its behavior is different if the **Size** is greater than 2.5 meters vs. less than or equal to 2.5 meters.

As in the **Extrudes** mode, a **Value** greater than 0.50 increases the amount of mud, and a **Value** less than 0.50 decreases the amount of mud. When increasing mud in **Automatic** mode, the brush preview disc is white. I'll describe decreasing/erasing **Automatic** mud in a later section.

The **Falloff** parameter is disabled in **Automatic** mode because paint strokes have their own unique behavior in this mode.

Automatic Mode With Size > 2.5

Automatic mode is called “automatic” because the brush does multiple things at once. With **Size** > 2.5 it does the following.

- It increases the softness of the mud in the same way as the **Extrudes** brush.
- It adds a dark tint to the painted area to indicate where subsurface soil has been exposed.



Since the **Automatic** mode records mud softness in the same way as the **Extrudes** mode, the **Extrudes to Wetness** property applies to both. So for mud that is deep enough, it can have both the dark tint of subsurface soil, plus additional tint and shininess from wetness.

Automatic Mode With Size ≤ 2.5

With size less than or equal to 2.5 meters, the Editor adjusts **Automatic** mode to paint a rut as if a truck drove through. This increases the softness of the mud and adds a dark tint as above, and it also adds lumpiness to the surface where the mud is pushed down in the center of the rut and pushed up slightly on each side of the rut.

Tip: Saber recommends a **Size** of 0.5 – 0.8 meters to represent a rut from a truck with narrow to wide tires.

Tip: Saber recommends a **Value** of 0.54 – 0.7 to represent the range from very shallow to very deep ruts.

Tip: A pair of parallel ruts best simulates the passage of a single truck (or of multiple trucks driving the same path). Position a truck of the appropriate size near the ruts to gauge how far apart they should be.

Tip: The **Autofade** checkbox is well suited for painting a rut that begins and ends with a smooth transition. However, it takes a practiced hand to get the **Autofade** tails to be parallel for each pair of ruts. It may help to be only press and release the right mouse button while the mouse is in motion.



With the recommended values, the mud in the ruts is not particularly soft, and it won't generate much wetness from the **Extrudes to Wetness** parameter. A truck driving through these ruts is more effected by the lumpiness of the ruts than the softness of the mud. You can use the **Extrudes** mode to increase the softness of the mud and/or manually add wetness to make it more slippery.

Decrease Mud

Mud can be decreased or erased with a brush **Value** less than 0.5. The effect is different depending on whether the brush is in **Extrudes** or **Automatic** mode.

Erasing in **Extrudes** mode initially only effects the mud softness. The lumpiness and dark tint are unaffected until the mud softness reaches zero, at which point the lumpiness and tint are removed. However, the lumpiness and tint remain recorded and will return if the **Extrudes** mode is used to soften the mud again.

The **Automatic** mode reduces the mud softness, its lumpiness, and the dark tint simultaneously. However, it does a really bad job reducing the lumpiness and tint. Depending on how you apply the brush, you can end up removing the tint while leaving the lumpiness, or remove the deep part of the rut while leaving the pushed up parts, etc.

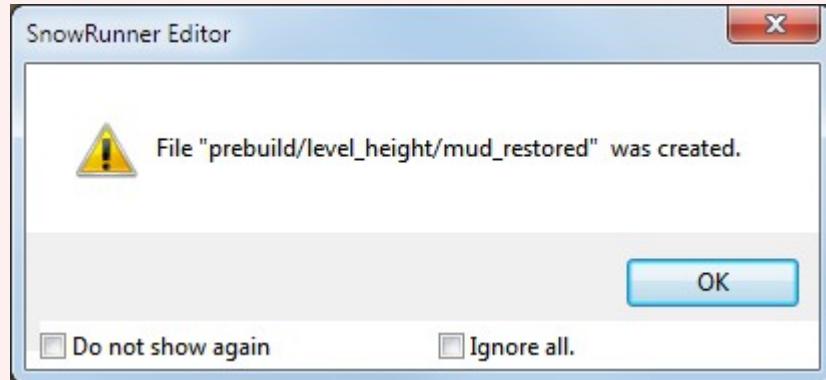
If you aren't satisfied with the visual appearance of a rut, it is generally easier to entirely erase it and start again. Use **Automatic** mode with **Value** 0.0 and a **Size** large enough to entirely cover the affected area. This also erases the mud softness, so you'll need to reapply that with the **Extrudes** mode if desired.

Conversely, if you like how your ruts look but want the mud to be more or less soft, you can paint or erase freely in **Extrudes** mode. As long as you finish with any mud softness, the existing ruts are unaffected.

Restore a Previous Mud State

As with other terrain brushes, you can click undo mud changes using the **Undo Current Changes** button or with **Ctrl+Z**. These facilities are described in more detail on page 104.

Bug: All **(Geometry)** items in the **Scene View** have a context menu item called **Restore version "mud" from "data.stg"**. Since **data.stg** is only updated when the map is manually saved, this appears perfect, but unfortunately it doesn't work. Selecting the action pops up a dialog to indicate that it is done:



Whether or not I replace my **mud** file with the new **mud_restored** file, and whether or not I reload the map, and whether or not I rebuild the terrain, the mud is never restored to the last manually saved state.

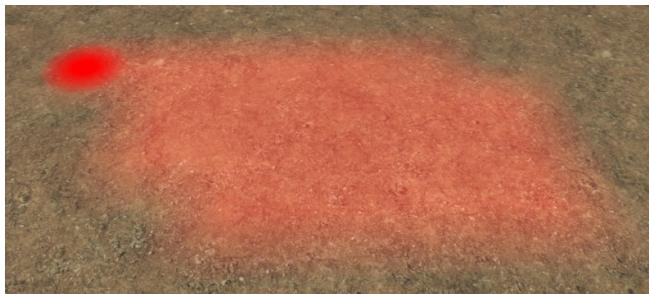
Tip: I recommend that if you're happy with your mud, manually make an archival copy of your map in progress. Then, if you accidentally mess something up, you can simply copy the archived **mud** file back into your **prebuild/map_name** directory.

QuickMud

Expand the **(Geometry)** category in the **Scene View** and click **(QuickMud)** to quickly paint a large swath of heavily churned mud. This feature works differently than other terrain brushes because you don't have direct control of the results. Instead, you simply indicate the broad areas where you want mud, and the Editor decides how it should look.

A **Value** greater than 0.50 increases the amount of mud, and a **Value** less than 0.50 decreases the amount of mud. When increasing mud, the brush preview disc is red. When decreasing mud, the brush preview disc is black.

The figures below show the act of painting with the **QuickMud** brush and the result.



Essentially, you paint an area with a red tint, and the Editor creates many pairs of truck ruts at random orientations in order to churn that area (and somewhat beyond, since it uses autofade).

The mud generated by the **QuickMud** brush is kept separate from the mud created by the **Mud** brush, so you can't directly observe the softness of the generated mud. However, experimentation reveals that where the **QuickMud** tint is brightest, the mud randomly ranges in softness from 0% to 35% of the maximum softness. Where the **QuickMud** tint is weaker, the maximum softness is reduced. You can add additional softness with the **Extrudes** brush if desired.

The brightness of the **QuickMud** tint also appears to have a subtle effect on the visual depth of the generated ruts, but I'm not completely positive about that. At the very weakest tints, **QuickMud** generates light patches of mud tint without any ruts.

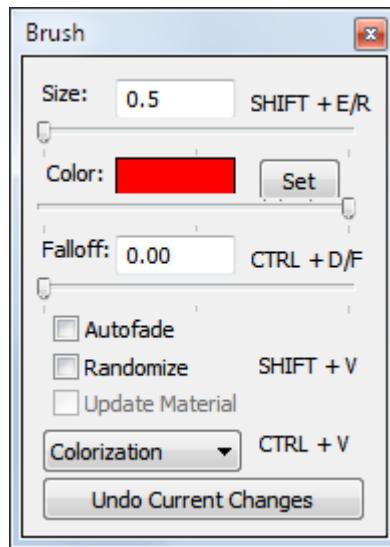
Colorization

Colorization is used to tint the terrain and plants on your map. In the early stages of creating a map, colorization is a great tool for planning the layout of your map. When your map is nearly complete, you can use colorization to make subtle color variations to add naturalism and variety to your map.

Expand the **(Geometry)** category in the **Scene View** and click **(Colorization)** to start using the colorization. The only brush mode is **Colorization**.

Choose a Color

The colorization brush dialog is distinct from other brush dialogs in that in place of a **Value**, the dialog has a **Color**.



Click the **Set** button next to the color to choose a different color. Another dialog pops up to pick a color. It lists a common set of basic colors, and up to 16 custom colors that you can choose yourself. Or you can create an unsaved custom color for one-off use. The Editor saves your custom colors so that you can use the same colors on different maps and in different Editor sessions.

Bug: When you click the **Set** button, the color selection dialog pops up with the color black selected, not the color you were just using. If you create a custom color, be sure to save it in one of the custom slots. Then if you need to edit the color, click the **Set** button, then select your custom color's slot before editing its values.

When painting with a color, the map is tinted with that color. But the Editor doesn't directly mix some amount of the colorization with some amount of the map's natural color. Instead, it looks at the **difference** of the colorization from a medium gray and applies that to the map. For strong colors, the distinction is minor: a

colorization of black or white or red or other primary color will tint the map closer to that color. But for colors close to a medium gray, instead of being tinted a medium gray, the map is instead tinted very lightly or not at all.

Paint with Colorization

In the colorization brush dialog, the slider under the **Color** selects how strongly the color is applied. It starts in the middle, which neither paints nor erases. Move the slider to the right to paint the selected color. Move the slider to the left to erase all colors.

I find that there is only the narrowest range in the middle of the slider where the paint or eraser is only weakly applied. It's much easier to use the extreme values with a weak paint that to apply a strong paint weakly. (Tips for creating weak paint are below.)

Bug: Normally the brush preview disc increases in intensity as the **Value** changes away from some neutral value that has no effect when painted. In **Colorization** mode, however, the brush preview disc treats 0.0 as the neutral **Value** when 0.5 is the actual neutral **Value**. This means that the preview disc is invisible near a **Value** of 0.0 even though this **Value** causes a large effect.

Erasing paint is equivalent to painting with medium gray. If you'd prefer to see the brush preview disc when erasing, change the color to medium gray and move the slider to the extreme right.



Colorization of Features

Of course, colorization applies to the material texture applied to the terrain, but it also applies to some other map features.

Roads, mud, and snow are colorized.

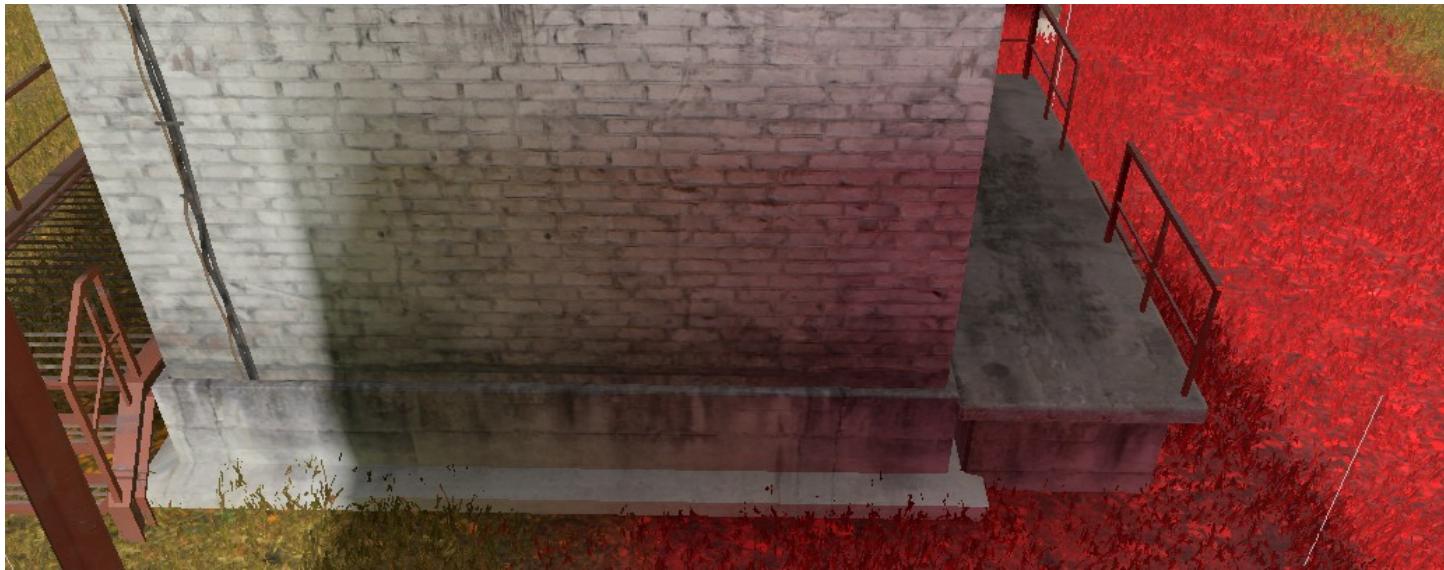
Grass is colorized, although the uniform tint of most grasses allows them to accept some colors better than others.



Plants are colorized using the color at the plant base. To see the change, the terrain must be rebuilt (page 65) after the color is applied, but the color then immediately applies to any trees placed in the colorized area. Trees with pale trunks are particularly vulnerable to colorization.



A model or conforming overlay is colorized, but only where a daytime static or dynamic shadow falls on it. To see the change, the terrain must be rebuilt after the color is applied, but the color then immediately applies to any models or overlays placed in the shadowed and colorized area. The elevated and wire overlays are all quite dark, so it's unclear whether they are affected by colorization. A model placed by an overlay brush is colorized the same as regular models.



Trucks are never colorized, even when shadowed.

Colorization has no effect on the surface or volume of a river, but it does affect the material layer on the river bottom as usual.

Plan with Colorization

When using colorization to plan the layout of your map, subtlety is unnecessary. Use strong colors and splash paint everywhere. Use a large brush to designate large areas, and use a small brush to draw lines or write text on the map. The colorization resolution is only ~1 meter, though, so text must be fairly large.



Colorization does not automatically appear in the [Terrain](#) view. Rebuild the terrain (page 65) to update the [Terrain](#) view.

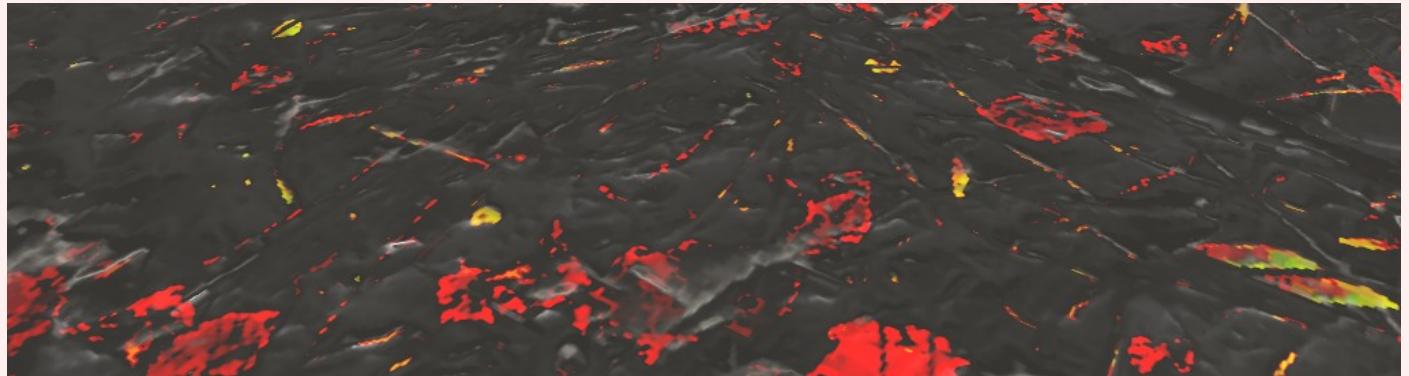
Adding Natural Tints

Adding natural tints is an art form that I seem to be terrible at, so I can't give a lot of tips here.

As discussed above, it is very difficult to apply the color paint weakly; it is much easier to strongly apply a weak color. E.g. to make the terrain a bit more of a dark brown, apply a paint that is a mix of dark brown and medium gray, e.g. R = 112, G = 96, B = 64.



Bug: Any color with a saturated color channel cannot be darkened in that channel. E.g. the `default` material layer contains bright leaves that turn pure red or yellow when darkened. This dramatically reduces the ability of colorization to apply a uniform natural tint.

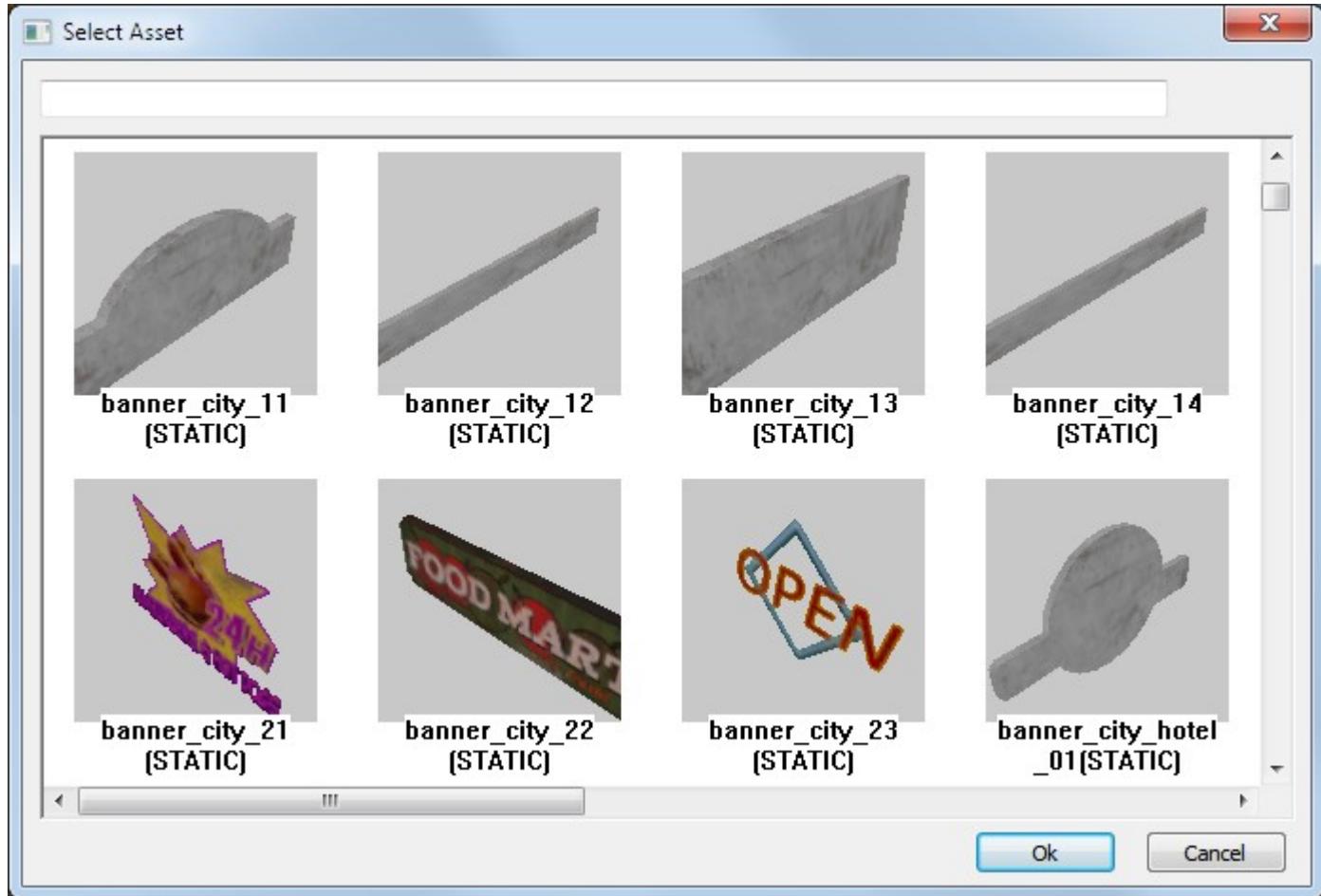


Consider also the separate wetness feature (page 217). Not only does wetness tint the terrain slightly darker, it also adds other beneficial features.

Bug: Particularly dark tints increase the softness of mud much like wetness does. Roughly speaking, if the R+G+B values add up to 192 or more, the colorized mud is unaffected. But if the R+G+B values add up to 128 or less the colorized mud is significantly softer. Applying a full black colorization makes mud as soft as full wetness.

Models

To add a model, choose **Add Model** from the context menu. A window appears where you can select what type of model you wish to create.



Bug: The first time you click here, the Editor may lock up for **30** or more seconds as it creates icons for all of the many add-ons. Later uses may pause again if the Editor decides to check for any changes, but never as long as the first time.

The model type is displayed in the **Brand** property, but it cannot be edited.

Scene.(Terrain).Models.type: **Brand:** read-only text

Indicates the model type.

Conveniently, you do not need to own any DLC in order to use models that were released as part of DLC. Nor does a player need to own any DLC as long as her game is up to date. This is unlike trucks, which do require you to own the necessary DLC (page 80).

Bug: DO NOT USE the Replace action in the context menu. It doesn't work right; it can't reliably be canceled; it may flicker between old and replacement models; it may silently corrupt your models until you quit the Editor and restart. Instead, create a new model of the desired type, then delete the old one.

If you accidentally select **Replace**, cancel it immediately. If cancel doesn't work, select a new model (not the original one), save your work if necessary, then entirely quit the Editor. When you reload your map, the XML may have the wrong model, but at least it is a consistent, error-free model. Also verify that the model types are correct for any models you had touched recently and the most recent models that you added (listed last in the **Scene View**).

Unlike a truck, you can left click a model in the main panel to select it.

Broken Model

If the Editor does the wrong thing when packing a map, then the game may not be able to render a model correctly. When this happens, the game renders a default “broken model” instead, a white exclamation point on a red pedestal.



If you see this, entirely quit the Editor, reload, and repack your map. In my experience, this should correct the problem.

Move, Rotate, or Scale a Model

Scene.(Terrain).Models.type: **Position.X**, **Position.Y**, **Position.Z**: numeric, in meters

Specifies the initial location of the model.

Default: ground level in the center of the main panel.

Scene.(Terrain).Models.type: `Rotation.X`, `Rotation.Y`, `Rotation.Z`: numeric, in degrees of rotation

Specifies the orientation of the model.

Default: 0, 0, 0; facing east (although what counts as the “front” of a model varies among models).

Move or rotate a model in much the same way as for trucks (page 75). The coordinate systems for position and orientation are described on page 420.

Bug: `Rotation.X` and `Rotation.Y` are swapped for models. I.e. increasing the `Rotation.X` value rotates the model counterclockwise around the `Y` axis, and increasing the `Rotation.Y` value rotates the model counterclockwise around the `X` axis. This is not the same axis swap as for trucks!

Unlike trucks, models cannot be tied to the terrain by way of a `Land` property. Instead, select `Do Land` from the context menu to move the model up or down to an appropriate height. The model may include parts below its base height, in which case `Do Land` moves these parts underground.

`Do Land` works best when the terrain is completely flat under the model. Otherwise, the Editor moves the model so that all terrain points are at or below the model’s base height within the model’s bounding box. Since the Editor checks only center of each terrain point, this may allow the edges of the model to intersect terrain where the terrain is interpolated between its defined grid of points.

You’re on your own to set the orientation of the model. Again, it works best if the ground is completely flat under the model so that the default model orientation looks natural.

If the terrain is not completely flat, be sure to carefully check around the base of the model for gaps and either re-orient the model or adjust its height until it has a solid connection.



Local Transform

When moving and rotating a model, it can be helpful to manipulate it using its local axes rather than the global axes. The **Local Transform** option in the toolbar (**Ctrl+L**) sets the axes of the interface widget to match the orientation of the model.

Scale

Scene.(Terrain).Models.type: Scale: numeric

Specifies the size of the model as a multiplier/fraction of the default size.

Default: 1.0.

Each model has a default size, but you can adjust its size to make it larger or smaller. This adjustment factor is the **Scale** property.

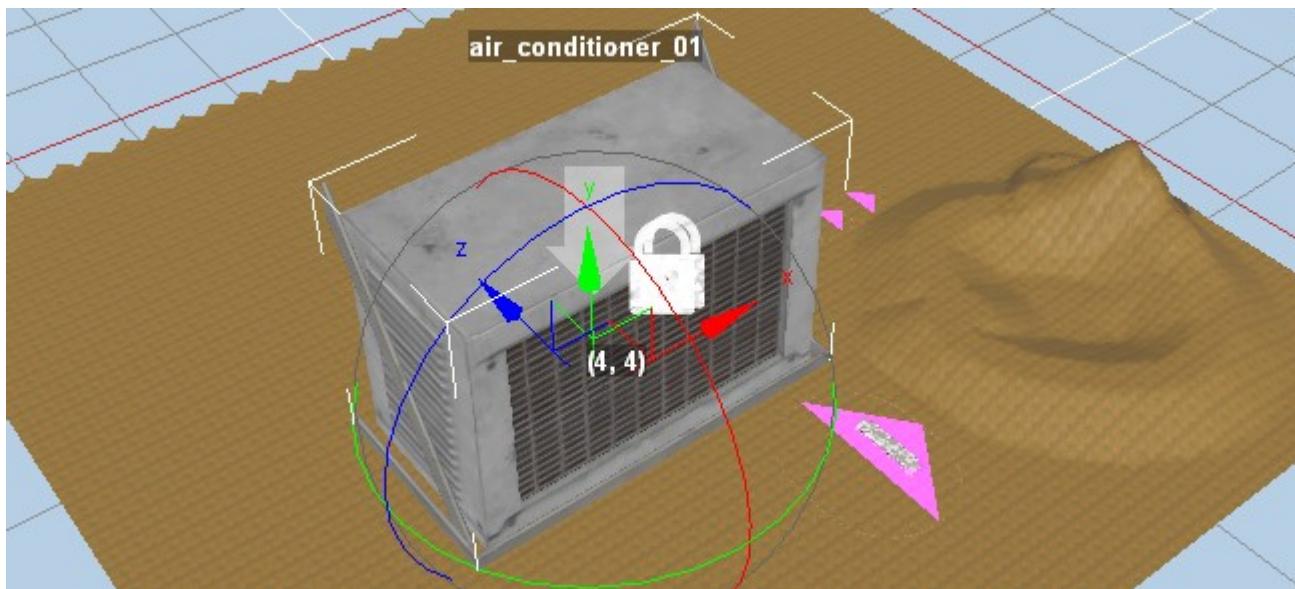
Tip: For many models, changing the size from the default can make it look unnatural. E.g. a larger building may look mostly fine, but its doors will look like they're made for giants. On the other hand, a bigger-than-usual windmill or a half-size bag of cement will look fine.

Tip: Many rock models are best used at larger scales. See the rock model reference on page 409.

Tip: `barrel_01` and `barrel_02` should always use `Scale` = 2 in order to appear as the industry-standard size. The other barrels are fine with `Scale` = 1. `barrel_06` is the same as `barrel_02` but with the correct default scale.

You can adjust the `Scale` value directly in the property panel. Alternatively, in the main panel you can left click in the diamond at the center of the model's axes and drag up or down to increase or decrease the scale, respectively.

The Editor limits the `Scale` to be no smaller than 0.01. There is no explicit upper limit, but if a model is much larger than anyone would consider reasonable, the Editor may crash. Be careful when typing values directly into the `Scale` field.



Model Physics

`Scene.(Terrain).Models.type`: Collision Type: read-only text

Displays the default physics behavior of the model.

The Editor annotates each model with what type of physics it implements:

- **[STATIC]**: The model never moves when a truck hits it.
- **[DYN]**: The model can move and/or deform when a truck hits it.
- **[DESTR]**: The model can break into pieces when a truck hits it. This is a special case of a dynamic model.

A static model can be one that a truck runs over, runs into, or passes through (e.g. because it is only a visual effect). The physics of static models are very efficient for game performance.

A dynamic model is one that moves when a truck hits it (or comes very close to it). Since a dynamic model can move, it also has to model the effects of momentum and gravity and check for collisions with the terrain and other obstacles.

For efficiency, the game enables physics for a dynamic model only when a truck gets very close to it. After a short period of activity, a dynamic model sinks into the ground and is removed from the game. It is reset to its initial position only when the player gets far enough away that she probably won't see it reset.

A destructable model is one that can break into multiple pieces when a truck hits it. Destruction may be trivial or may require considerable force. Once destroyed, each piece of the original model then acts like its own dynamic model.

If a static model is placed such that it intersects the ground or another static model, nothing special happens. Indeed, it is quite common to place a static model slightly depressed into the ground so that there is no visible gap below it.

On the other hand, if a dynamic model isn't placed well, it may react with the environment before the truck touches it. This can cause the model to fling itself into space, or even fling itself at the player's truck, causing unexpected damage.

Tip: It is generally best to place a dynamic model exactly on level terrain or just above bumpy terrain.

Mud and snow are a special case since a dynamic model sinks slowly into these surfaces when the player's truck gets near. You can consider slightly depressing the model's initial position into the mud or snow. E.g. for deep mud, [barrel_06](#)'s natural buoyancy is about 40 cm deep, and for deep snow (without mud underneath), 40 cm is coincidentally the maximum depth that anything can sink. Be sure to test nudging the model with a truck in the game to see how it reacts.

Collision Notification

If [Collision Notification](#) is enabled in the Editor's toolbar, then the Editor highlights a dynamic model that is in a position to collide with another object or the terrain. The highlight changes the bounding box of a deselected model from its normal gray to violet.

Bug: Because the bounding box is only colored when a model is deselected, you can't see potential collisions while you are manipulating a model, only when you are done.

Bug: The bounding box color is changed only for the model that is being deselected. The color does not change for other models that the deselected model is now or was previously colliding with.

Bug: The Saber guide indicates that the bounding box should be red or violet depending on whether the dynamic model collides with the terrain or another model. However, in the current Editor, all collisions are highlighted with violet.

A collision notification doesn't necessarily mean that the model will react violently when approached. The forces may be small enough that it only moves gently away from the collision, or friction may even prevent it from moving at all.

Conversely, the lack of a collision notification doesn't mean that the model will stay put when approached. If it isn't supported properly, gravity could cause it to fall when its physics are enabled.

Freeze Physics

`Scene.(Terrain).Models.type: Freeze Physics: True/False`

Specifies whether to force the model to behave as a static model.

Default: `False`.

You may prefer to use a dynamic or destructable model as if it is a static model, e.g. when it is clustered with other models in a static scene. This can improve performance (particularly if there are a lot of models) and also be less confusing than having some models move while other models stay put.

To treat a model as a static model, change its `Freeze Physics` property to `True`. A truck can still collide with the model, but the model won't move as a result. When a model has `Freeze Physics` enabled, its bounding box is black when deselected.

`Freeze Physics` has no effect on a static model other than to change its bounding box color. It does not stop animations.

Special Models

The following categories of models benefit from additional explanation.

Cargo

As of season 5, Saber has removed cargo models from the list of model assets. To create a true cargo 'model', you now need to hand edit the map XML file. However, Saber removed the cargo models for a good reason, and I don't recommend their use.

The preferred way to spawn cargo on the map is via an objective, as described on page 340.

Country-Specific Models

Many models are labeled with `us` (or `usa`), `ca`, or `rus` in their names, representing the US, Canada, and Russia.

In many cases, a model is only available for one country. Since Saber generally builds models for use in a specific map, many models associated with a specific type of location may only be available for one country (e.g. rocket factory buildings in Russia). Keep this in mind when planning a map.

Not all country-specific models are labeled as such. Check each model carefully for any writing on it that may tie it to a specific country.

If two models have the same name except for the country label, then the models often (but not always) have the same shape with different applied textures.



Nothing prevents you from using a mix of models from different countries on the same map if you feel that they look good together.

Rocks and Ice

Many rock models are best used at larger scales. See the rock model reference on page 409.

In addition to the rocks as models, there are also rocks that are implemented as “plants” in order to allow their use in distributions (page 147). Similarly, different ice shapes are implemented as either models or plants.

Winch Attachment Points

Some models include winch attachment points (similar to trees). Besides the familiar power poles and streetlights, many other pole-like structures have winch attachment points. (Other structure shapes can't support winch attachment points as well because nothing prevents the winch line from passing through the structure.)

Lying Trees

A couple of models appear to perfectly duplicate trees of the same name: [pine_lying_a](#) and [spruce_lying_02](#). The major difference that I've found is that the tree plants are firmly rooted to the ground (despite being tipped over), while the tree models are easily pulled out of position. That makes the models much less useful as winch attachment points.

Lights

Some models include lights. SnowRunner has two types of lights, and a model often includes both types.

A “flare” is an ambient glow in the air. For an omnidirectional light, this is usually a ball of diffusely brighter color around the light source. For a directional light, this may be a cone of brightness that fades with distance from the light.

Bug: The SnowRunner Editor does not display flares, even in [Night](#) mode.

The screenshot below shows two red flares for the truck's marker lights, two orange flares for the running lights on the roof, and two streetlamps with slightly different colors of flares below and in front of them. These are distinct from the bright white bulbs of the streetlamps, which are simply a super-white part of the 3-D model.



A “light” illuminates objects and terrain within range of the light. A light cannot be seen directly; it can only be detected by the change in brightness of other objects. A light can be omnidirectional or directional. The screenshot below shows both the flare of streetlights and the light that they cast on the ground. (The poles in the background are models of broken streetlights.)



Lights are visible in the evening (starting at 20:30) and night (ending at 06:00), although their effects are hard to see in the first and last half hours.

Bug: In the game, using the **Change Time** option in the **Tools** menu to switch to nighttime does not trigger lights to shine. Instead, use the **Skip Time** feature from the navigation map to change the time to **Night**.

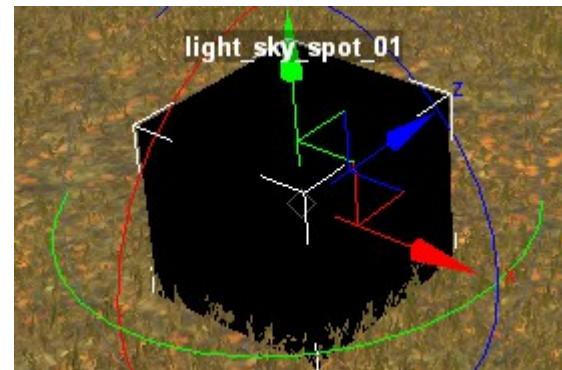
To view the effect of lights in the Editor, you must explicitly rebuild the terrain.



light_sky_spot_01

You might run across the `light_sky_spot_01` model and wonder what it does. It includes a model of a black cube, but that shouldn't be visible in the game because you're expected to bury the model underground.

The model comes with a **very** dim spotlight that hovers 25 meters above the cube and shines down on it. It's basically a nightlight for a small area of the map so that terrain and objects in it can be seen without other sources of illumination (e.g. when a truck's headlights aren't pointed at it).

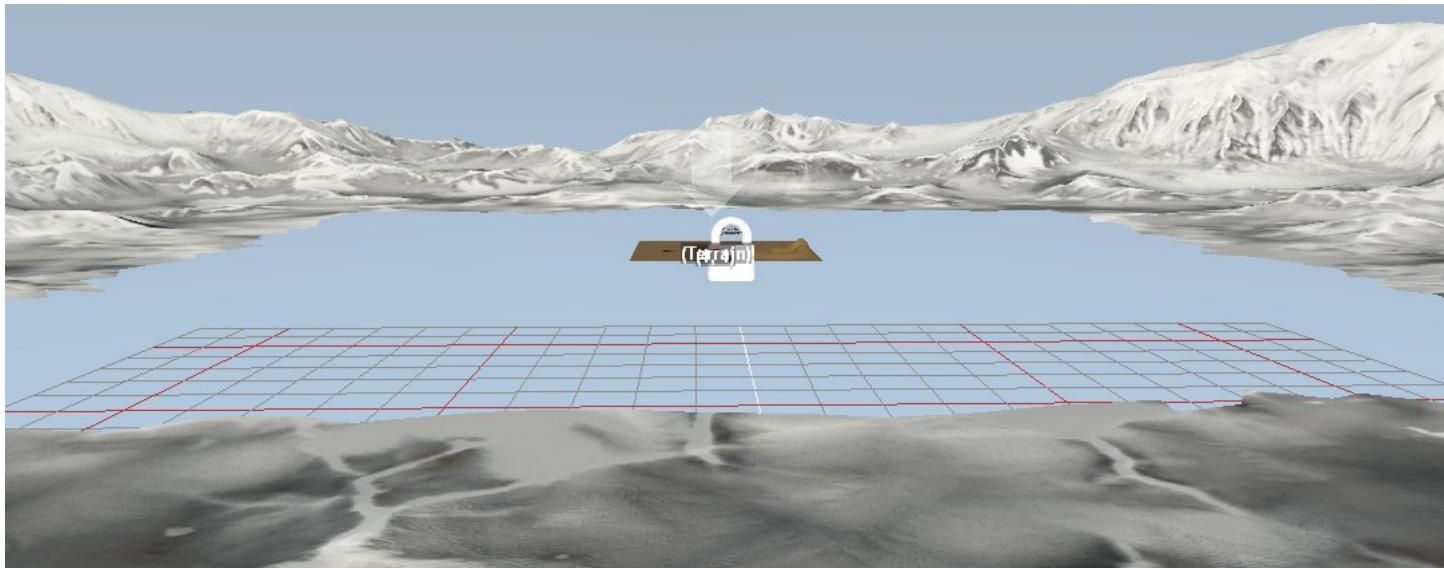


In the screenshot below, I have buried the `light_sky_spot_01` model about 20 meters deep, so its light source is about 5 meters above the ground. Since it's a directional light, it lights a circle below it. I placed a large, snow-covered flat surface below it to make the light effect more visible (including glints of reflection off the snow). It's very subtle in the Editor. If it's visible in the game, it requires a darker sky than I have yet tested with.



Farplane Models

A few models can be used as a backdrop beyond the edges of your map. These models have the names `farplane_*`.



You can scale or shift the position of a farplane model as you like, although it is not intended to match the edges of your terrain. Instead, the idea is that your map's terrain is high near the edges so the player is never looking more than slightly downwards when looking beyond the terrain edges, and the farplane model is then visible in the distance.

Some farplane models have a low side. These are appropriate when used with background water that extends to the horizon (page 214).

Animated Models

Some models have animated motion, e.g. the country flags mentioned above. Animation is shown in the Editor only when the model is deselected. Animation can be sped up, slowed down, or paused in the Editor with the **Time speed** property (page 251) and the **Pause** toolbar button.

Freeze Physics has no effect on animations.

Animals

I am aware of the following animal models:

- **bear_01**: a stationary white bear. It can only be seen at a distance. It disappears (fades away) if the player drives closer. The game does this by fudging the levels of detail for the model, described on page 133.

- **wolf2**: an unblinking pair of not-at-all wolf-like eyes. These appear at closer distances than the bear above, but only at night. The wolf eyes are bidirectional and face along the model's +Z and -Z axes.

Tip: Place the **wolf2** model where it is unlikely that the player will shine her truck's headlights on it. Otherwise it looks ridiculous when the light reveals that nothing is there. The inside of a curve or up a protected slope are potentially good locations.

- **birds_flying_01** and **birds_sea_01**: a flock of animated birds flying in circles and figure-8 patterns. Make sure to give them some height so that they aren't just skimming over the ground. Visible at any distance, day or night. The bird models are silent. To get associated sound effects, see pages 220 or 225.
- **butterfly_flying_01**: a cluster of butterflies flying in large circles and figure-8 patterns (similar to the birds above). Visible at any distance, day or night. The butterfly model isn't very good; unlike the birds, the butterfly wings are one sided, so the player can typically see only one wing at a time. The default scale is also way off. A **Scale** value of about 0.2 seems to strike a decent balance between being too big to be plausible and too small to be visible.

Bug: The **wolf**, **hunter_01**, and **yeti_snow_man** models appear to be broken. I can see the **yeti_snow_man** in a narrow range of distances in the Editor, but never in the game.

None of the animal models can be collided with.

Multi-Stage Models

Multi-stage models change state in response to completion of an objective stage (page 328) or delivery of cargo (page 338). These models have **_objective** in their name. The state change can be purely visual, or it can remove a barrier that was blocking trucks from passing. Some models have a fancy animation to show a change, while others simply swap to the new shape with a sound effect and a cloud of dust.

When viewed in the Editor, a multi-stage model shows all possible states simultaneously (but not the animations between the states).

Model ID

To associate a multi-stage model with an objective, you must give it a model ID in the **Tag** property.

Scene.(Terrain).Models.type: **Tag**: text

Specifies a unique identifier for this model for use by objectives.

Default: blank; objectives cannot refer to this model.

Each model must have a **Tag** value that is unique among models. The ID allows the model to be linked to the progress of an objective.

Another model in the map should not have the same `Tag` value. The game may hang or crash or exhibit strange behavior if it finds a duplicate model ID within a map. However, another map in the same region may reuse the same model ID.

For safety, I recommend that the model ID be limited to a combination of lowercase `a – z`, uppercase `A – Z`, `0 – 9`, and `_`. It should not include other characters, including spaces.

Use One Stage of a Multi-Stage Model

Some of the multi-stage models have an associated model without the `_objective` in its name. These single-stage models have only a single appearance, generally the “intact” stage of the associated multi-stage model. These are your best bet if you just want the model without its multi-stage progression.

Other multi-stage models don’t have an equivalent single-stage model, however. In that case, you can use a single stage of the multi-stage model with a bit more hassle. If a multi-stage model is not connected to an objective, then the game renders it in its default stage. A reference guide on page 410 describes the general appearance of each multi-stage model in each state. If you instead need the model in one of its other stages, then you’ll need to connect it to an objective.

Tip: To use a single stage of a multi-stage model, associate that model stage with objective stage 0 of any objective (page 328).

Bug: If a multi-stage model is not connected to an objective, its landmark may not be rendered on the navigation map in the correct default stage.

Animation Camera

`Scene.(Terrain).Models.type`: `Animation Camera Frame Name`: dropdown menu

Specifies one of the model-specific camera angles to use for animation between model stages.

Default: depends on the model type.

If a multi-stage model has an animation between stages, it can designate a camera position and movement from which to watch each animation. The Editor automatically fills the `Animation Camera Frame Name` property with a valid camera ID for the model, which is almost always `objective_camera_0` for models that have an associated camera. The property can be edited, but the only model with more than one available camera is `bridge_road_01_a_objective`. Its cameras all sweep along the same sides of the bridge at each stage, but each views a different amount of the surrounding terrain. They range from a low position that gives a long field of view in `objective_camera_0` to a high position with a short field of view in `objective_camera_5`.

Elevated Overlay Bases

The pipe_oil_us_base and conveyor_belt_base models are intended to be instantiated as part of an associated elevated overlay (page 186). These models therefore have an unusual appearance on the navigation map (page 202) that likely makes them unsuitable as stand-alone models.

Other Aspects of Appearance

Shadows

The behavior of shadows for all objects is described starting on page 134.

Procedural Snow

A winter map adds snow to certain surfaces of a model. See page 173 for details.

Level of Detail

For performance reasons, most models have multiple levels of detail. When viewed up close, all details are visible. But from a medium or far distance, a simpler model is used, or the model disappears entirely. The models from Saber generally have well designed levels of detail so that you don't see much difference as you approach or get further away. But you can occasionally notice some small differences.

Landmark

Large models comes with a simplified 3-D model, called a “landmark”, that is used on the game’s navigation map. Small models do not have landmarks and are not shown on the navigation map.

Note that the **Terrain** view shows only the colors of the terrain itself, not any of its 3-D landmarks.

Shadows

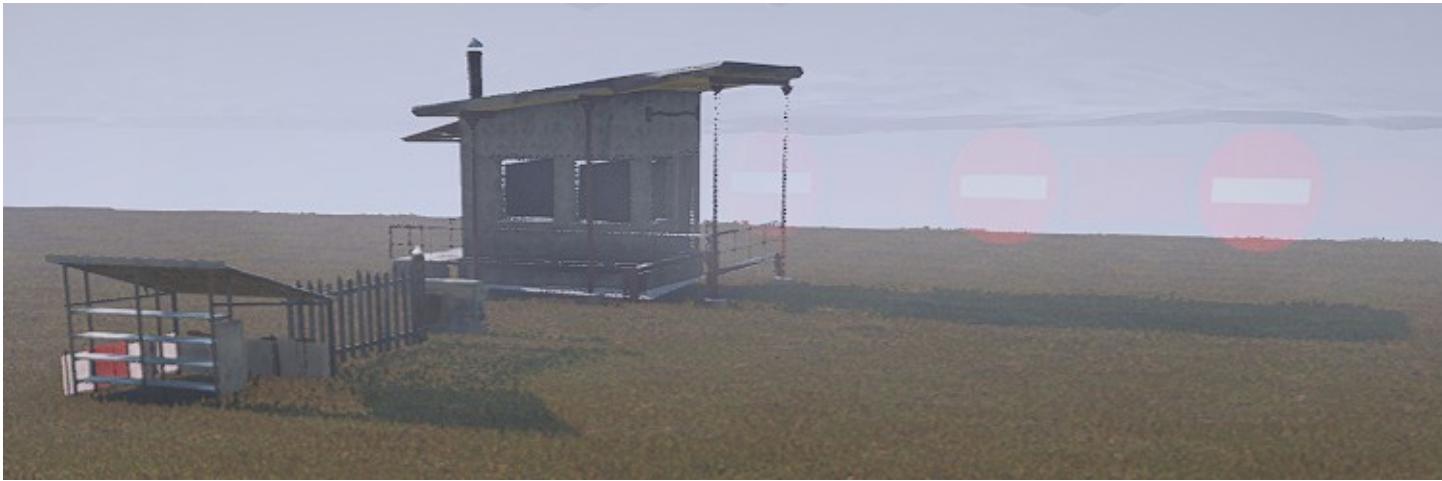
Shadows come in three kinds, described below.

Shadows can be cast by trucks, models, plants, 3-D overlays, and even the 3-D shape of the terrain itself.

Dynamic Shadows

Dynamic shadows are the best looking shadows: they have high detail and take into account the dynamic positions of the sun and of objects. E.g. a low sun casts long shadows, and shadows move when the object casting the shadow moves. Dynamic shadows have a heavy performance cost, so the game only renders dynamic shadows within 50 meters of the player camera.

Dynamic shadows can only be seen in the game, not the Editor. The Editor only has static shadows, described in the next section.



Daytime Shadows

During the day, the game only has to deal with one light source (the sun), so it is able to display dynamic shadows for all objects.

The color of the daytime shadow on a 3-D object is influenced by the average color of the ground underneath, as if the shadowed area is being indirectly illuminated by light bouncing off the ground. This gives a greenish tinge to shadows above grassy terrain. It also means that colorization of the terrain has an indirect effect on shadowed areas.



Nighttime Shadows

At night, the game has to deal with multiple light sources. The moon is not a light source, but the various streetlamps and other models can be light sources. The game therefore simplifies by casting a dynamic shadow at night from only one object: the player's truck. Other trucks and other types of objects don't cast a dynamic shadow at night. The player's truck's shadow appears to be cast by a weighted average of the nearby light sources. I.e. there is only one shadow, not many.

Truck headlights do not cast dynamic shadows. Although the back sides of model surfaces are less illuminated than the front sides (relative to the truck's headlights), the model does not actually block light from reaching other surfaces. E.g. in the screenshot below, those window openings are not being lit by an internal light, but by the truck's headlights.



Unlike daytime shadows, nighttime shadows have a uniform color and are not tinted by the terrain underneath.

Static Shadows

Static shadows are similar to dynamic shadows, but are based on fixed positions for light sources and objects. They are computed separately for day and night when you rebuild terrain, and the static shadows are saved with the map. Static shadows can be seen in the Editor and are also used in the game for distances above 50 meters.



Note that the 50-meter cutoff is the distance between the camera and the shadow, regardless of the distance to the model. This occasionally results in visible artifacts such as a shadow that appears short in the distance but which resumes as a long shadow near the player's camera.

Daytime static shadows are computing using a fixed sun position. As with dynamic shadows (above), daytime static shadows are tinted with the color of the terrain underneath.

As with dynamic shadows (above), nighttime static shadows are computed at each position using a weighted average of the neary by light sources.

Not all objects cast static shadows. See the below subsections for details.

After moving an object or changing the shape of the terrain, rebuild the terrain (page 65) to update the static shadows. Note that static shadows are omitted when rebuilding in **Quick mode**.

Static Shadows for Models

Static shadows for models are controlled by properties.

`Scene.(Terrain).Models.type`: **Disable Day Static Shadow**: True/False

`Scene.(Terrain).Models.type`: **Disable Night Static Shadow**: True/False

Specifies whether to disable static shadows for the model during the day/night.

Default: depends on the model.

You can manually enable or disable shadows during the day and/or night using the **Disable Day Static Shadow** and **Disable Night Static Shadow** properties. For most models, the default **Disable Day Static Shadow** and **Disable Day Static Shadow** values are **False**, so shadows aren't disabled (and therefore are enabled). Some

models instead default one or both values to **True**. Models that can move (dynamic models and multi-stage models) sometimes disable their static shadows to avoid a ridiculous-looking shadow.

Bug: Many multi-stage models are missing this default disable of static shadows, so you have to do it manually for each one. Otherwise, for example, you could get a static shadow for a tower that hasn't been built yet. In general, a missing shadow looks better than an extra shadow.



Bug: Many dynamic models are also missing the default disable, although this tends to be less obvious simply because the model and its shadow are smaller. In the below screen shot, I've used a truck to push the sedan and barrel a little bit away. Note that their shadows are now on the wrong side compared to the static tower to the left.



Static Shadows for Other Objects

Whether static shadows are displayed for plants and 3-D overlays is a built-in property of the object type and cannot be changed in the Editor. Some extra notes regarding plant shadows are in the plants section on page 145.

Trucks and trailers are highly mobile and therefore never cast static shadows.

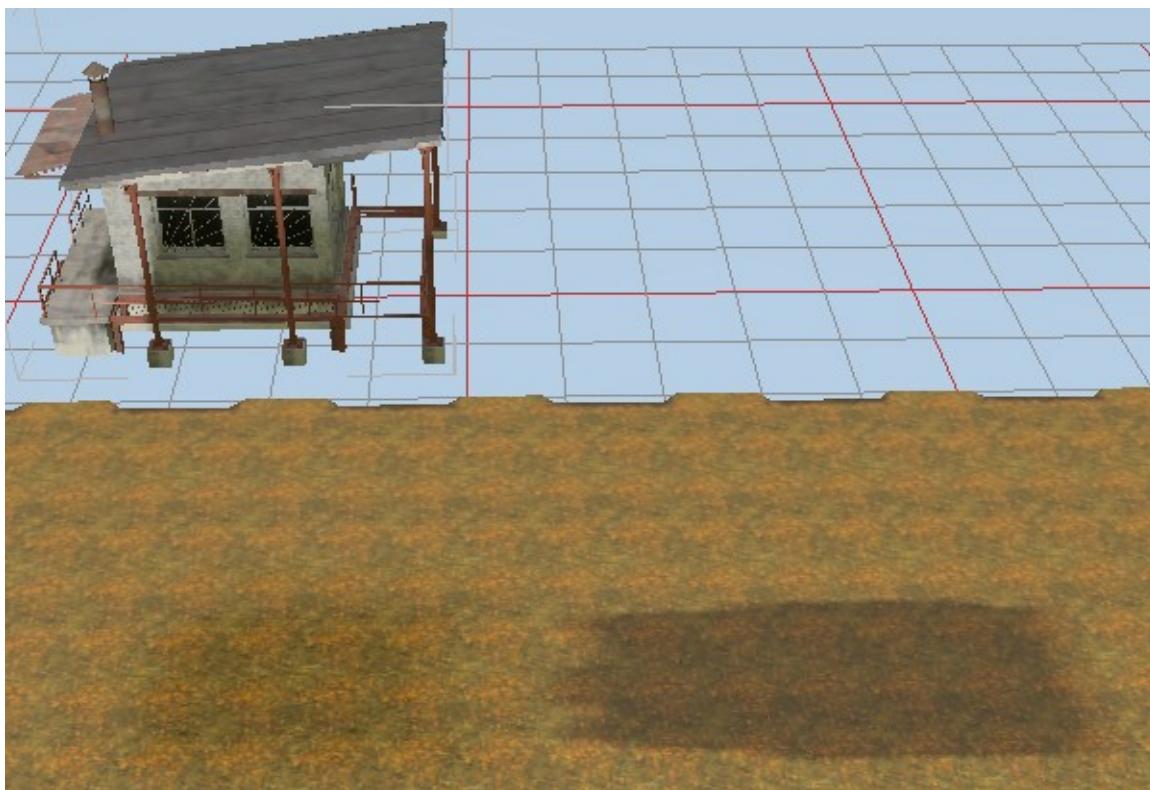
High terrain casts static shadows on lower terrain and objects.

Occlusion

An occlusion shadow results from blocking the omnidirectional glow from the sky rather than the directional glow from a light source. As such, they are generally less dark and more diffuse.

Occlusion shadows are essentially unchanged between day and night, although they tend to be less noticeable at night.

Occlusion shadows are drawn directly beneath the object, not offset by the sun angle. The size and tint of the occlusion shadow is not affected by the object's height. If the object is below the ground surface, the occlusion appears on the terrain surface above it. The screenshot below shows the occlusion shadow directly below the floating shed, and the static shadow offset to the right.



Occlusion shadows are statically calculated like static shadows. After moving an object, rebuild the terrain (page 65) to update the occlusion shadows. Note that occlusion shadows are omitted when rebuilding in **Quick mode**.

Occlusion Shadows for each Object Type

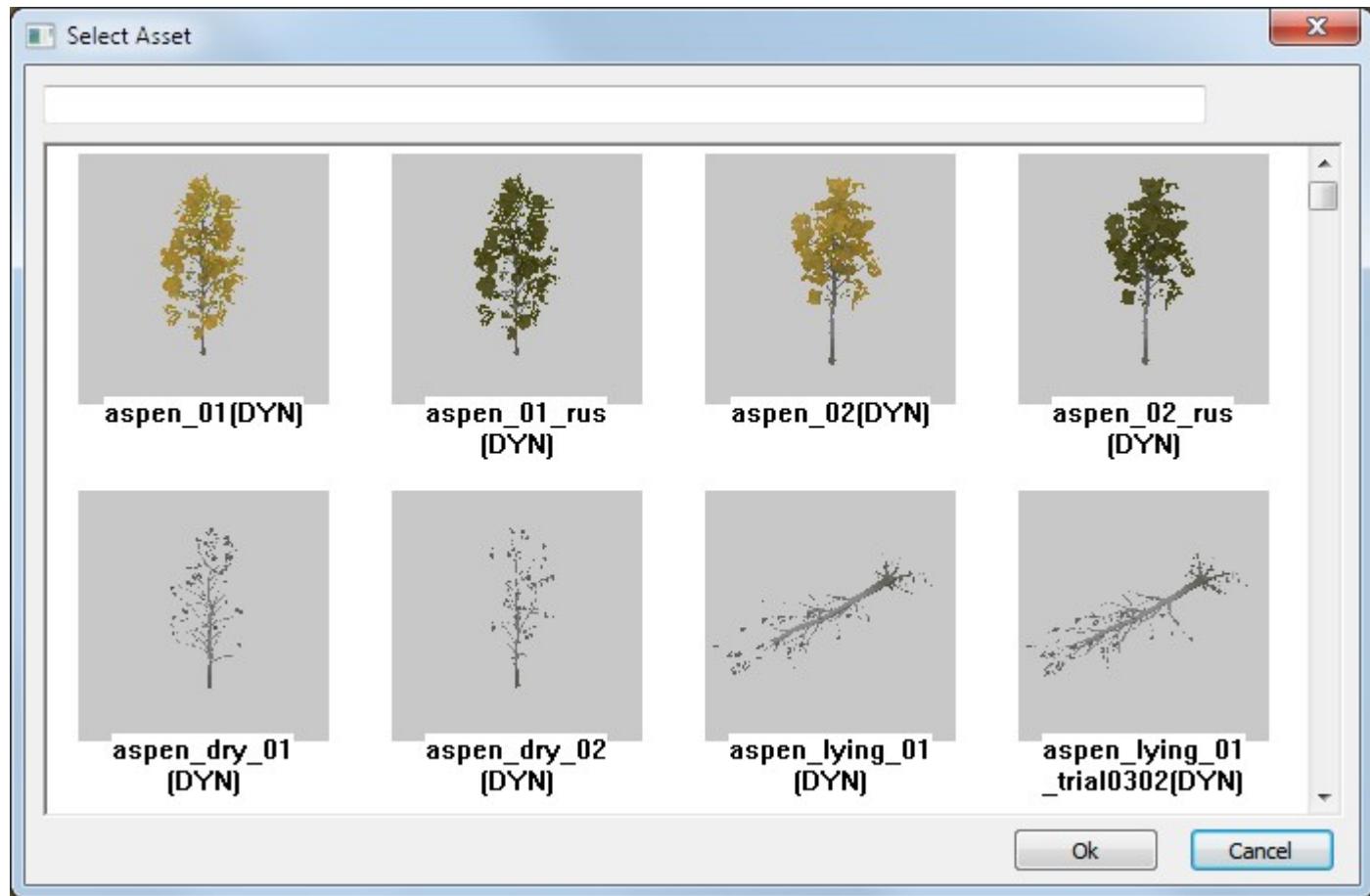
The occlusion shadow for all objects is enabled or disabled by the built-in properties of the model. You have no control over it from the Editor.

Trucks and trailers are highly mobile and therefore never cast occlusion shadows.

Individual Plants

You will add most plants to your map using distributions, described on page 147. Distributions are great for randomly placing many plants, but you don't have exact control of them. Alternatively, you can add an individual plant in much the same way as a model. This lets you place a plant exactly where and how you want it.

To add a plant, choose **Add Plant** from the context menu. A window appears where you can select what type of plant you wish to create.



Bug: The first time you click here, the Editor may lock up for 5 or more seconds as it creates icons for all plants. Later uses may pause again if the Editor decides to check for any changes, but never as long as the first time.

There are a few non-plants listed in the asset window, particularly rocks and ice chunks. These are grouped with the plants because they can be distributed in large numbers like plants. Distributions are described beginning on

page 147. There are also some plants (e.g. grasses) that can only be placed by distributions or materials. Since they cannot not be individually placed, they are not included in the plant asset selection window.

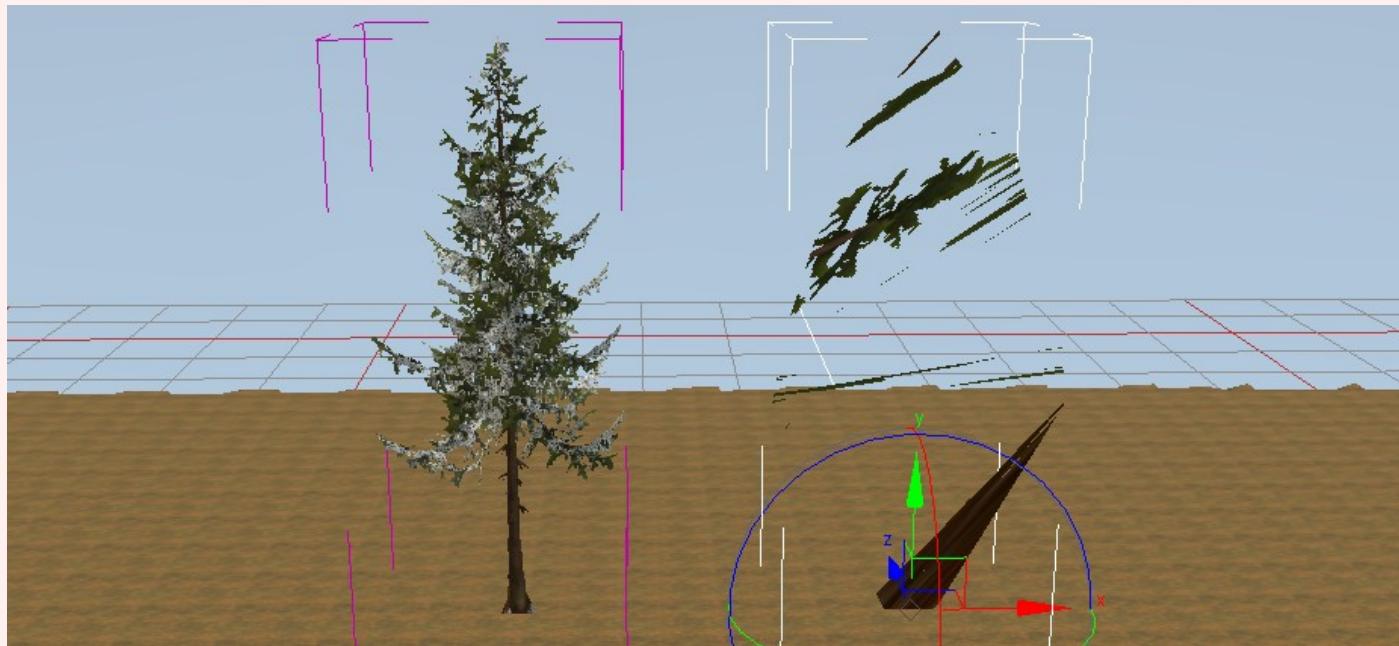
Once the type of plant has been selected, you cannot change it. You can only delete the plant and add another.

Scene.(Terrain).Plants.type: **Brand:** read-only text

Indicates the plant type.

As with a model, you can left click a plant in the main panel to select it.

Bug: Something is very wrong with the Editor's rendering of selected plants. Many trees and bushes disappear or are extremely distorted at lower levels of detail, and some rocks radically change height when selected. The deselected appearance seems to be fine in all cases.



Bug: `cane_chunk_a` displays numerous warning dialogs about missing billboards and then displays as red rectangles at lower levels of detail. This plant isn't used in any distributions, and with this bug you shouldn't place it as an individual plant, either.

Move, Rotate, or Scale a Plant

Scene.(Terrain).Plants.type: **Position.X**, **Position.Y**, **Position.Z:** numeric, in meters

Specifies the initial location of the plant.

Default: ground level in the center of the main panel.

Scene.(Terrain).Plants.type: `Rotation.X`, `Rotation.Y`, `Rotation.Z`: numeric, in degrees of rotation

Specifies the orientation of the plant.

Default: 0, 0, 0; facing east (although what counts as the “front” of a plant is rather arbitrary).

Move or rotate a plant in the same way as for trucks (page 75). The coordinate systems for position and rotation are described on page 420.

Bug: `Rotation.X` and `Rotation.Y` are swapped for plants. I.e. increasing the `Rotation.X` value rotates the plant counterclockwise around the `Y` axis, and increasing the `Rotation.Y` value rotates the plant counterclockwise around the `X` axis. This is not the same axis swap as for trucks!

Scene.(Terrain).Plants.type: `Scale`: numeric

Specifies the size of the plant as a fraction/multiplier of the default size.

Default: 1.0.

Scale a plant in the same way as for models (page 122). Unlike models, all plants look fine with a `Scale` near 1. As with models, you are free to adjust the scale to get a pleasing size.

Local Transform

When moving and rotating a plant, it can be helpful to manipulate it using its local axes rather than the global axes. The **Local Transform** option in the toolbar (`Ctrl+L`) sets the axes of the interface widget to match the orientation of the plant.

Do Land

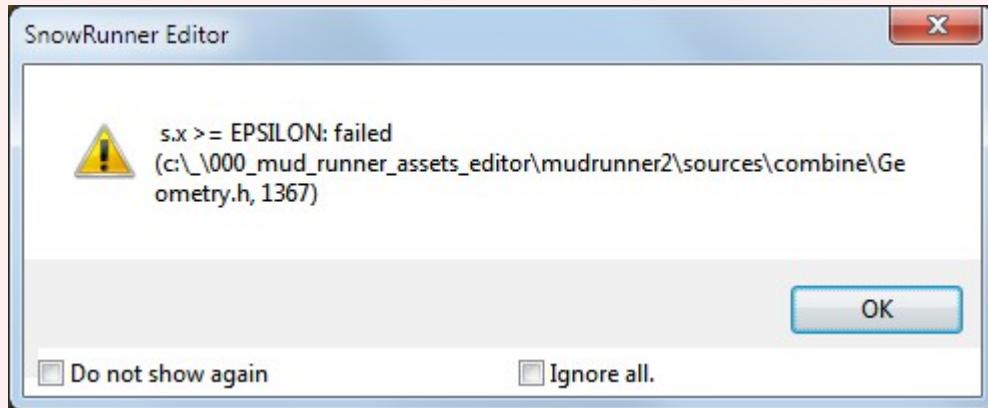
Scene.(Terrain).Plants.type: `Do Land`: `True/False`

Specifies whether the plant’s position is tied to the ground.

Default: `True`.

The **Do Land** property for plants is similar to the **Land** property for trucks (page 77).

Bug: Changing **Land** to **False** often changes a plant's **Scale** to 0. I haven't figured out the pattern yet, so I guess you can hold your breath and give it a try. The 0 **Scale** causes a few identical warning dialogs to pop up whenever you do anything else. The Editor repeatedly changes the **Scale** to 0 if you try to fix it, so the only known workaround is to delete the plant and try again.



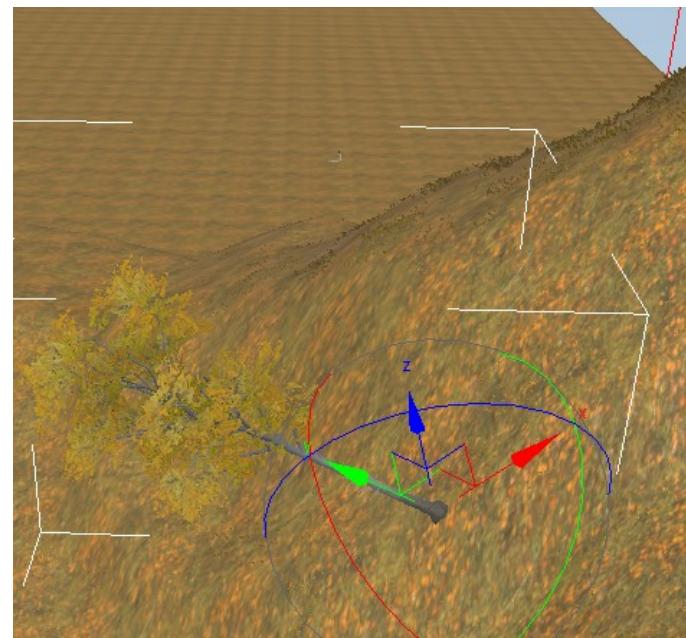
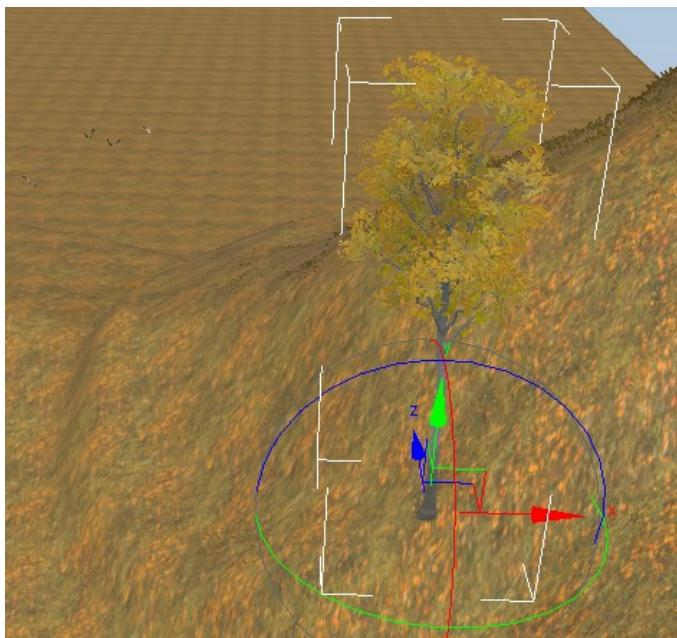
Perpendicularity

Scene.(Terrain).Plants.type: Perpendicularity: numeric slider from 0.0 – 1.0

When **Do Land** is **True**, this specifies the fraction of the terrain angle that should be transferred to the plant's orientation.

Default: 0.0.

By default, plants are oriented vertically, even when **Do Land** ties them to angled terrain. Especially for bushy type plants, it may make sense to have them grow perpendicular to a hillside rather than vertical. You can rotate the plant by hand, but for this task it may be easier to increase a plant's **Perpendicularity** property.



A perpendicularity near 1.0 (100%) doesn't make sense for large trees, but it may make sense for pumpkins. And slight perpendicularity may also improve the appearance of larger plants.

Bug: Decreasing the **Perpendicularity** property doesn't stand the tree back up again. You have to set the **Perpendicularity** value to 0, then set the **Rotation.Y** and **Rotation.Z** values to 0 by hand, then increase **Perpendicularity** to the desired value.

Bug: The rotation widgets don't work when the **Perpendicularity** is greater than 0. If you try to use any rotation (even a local rotation around the plant's axis), the plant rapidly tilts toward 100% perpendicularity. The rotation widgets only work when the **Perpendicularity** is exactly 0.

Tip: If the bugs in **Perpendicularity** give you a headache, just set **Perpendicularity** to 0 and rotate the plant by hand. With **Local Transform** enabled, you can rotate the plant to face the slope, and then it's easy to tilt it downslope.

Terrain Height Adjustment

Some trees raise the terrain around their base point. The nominal height increase is 30 cm, but it may be less if the tree is positioned between terrain vertices. The terrain is raised even if the tree is not touching the ground.



Bug: If two or more trees are planted close together, their terrain height adjustments are cumulative. You probably want to avoid that.

The height adjustment is separate from the base terrain height. If you move the tree, the height reverts to its original value. However, these changes to terrain height (whether to raise it or revert a previous increase) are not performed until you rebuild the terrain (page 65). If a tree has its **Do Land** property set to **True**, the new terrain height changes the height of the tree.

Smaller plants such as bushes do not raise the terrain height.

Plant Physics

The Editor annotates each model with what type of physics it implements, the same as for models (page 123). Most plants are **[DYN]** or **[DESTR]**, but there are a couple of **[STATIC]** rocks.

Unlike most dynamic models, most plants (except chunks) are physically attached to their base point, even if the plant's base is floating in air. The main consequence is that you don't have to worry about plants sinking into mud or snow. They also generally bend in place rather than being pushed around, although that changes if the plant breaks into chunks.

Plants also don't physically interact with models or other plants, so plants can be placed close to obstacles without worrying that they may have adverse collision events. Although **Collision Notification** highlights "collisions" by plants, these serve at most as a warning that the plant may look odd growing through another object.

Freeze Physics

`Scene.(Terrain).Models.type: Freeze Physics: True/False`

Specifies whether to force the plant to behave as a static object.

Default: `False`.

Like models, you can change a plant into a static object by setting its `Freeze Physics` property to `True`. It still blows in the wind, but it won't budge after a truck impact or bend in response to winching. Since plants are quite performance efficient even in great numbers, there is little reason to use `Freeze Physics` with them, and it has the potential to greatly confuse the player.

Country/Season-Specific Plants

Because the initial US maps were set in autumn, many of the 'default' plant names use autumn colors, and those with `rus` in their names have summer colors. Feel free to use whatever mix of plants looks good to you.

Winch Attachment Points

Trees have winch attachment points. Mature trees (regardless of scale) tend to be unbreakable and therefore solid winch points. Younger trees are easy to bend and may break when the winch stress becomes large.

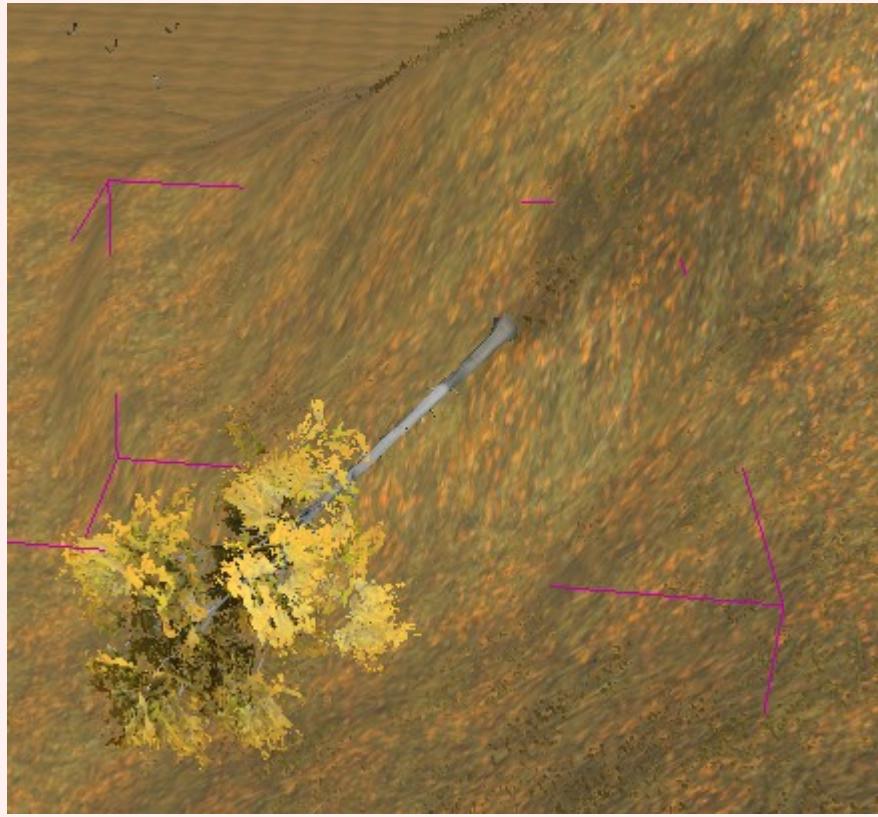
Other Aspects of Appearance

Shadows

The behavior of shadows for all objects is described starting on page 134.

Plants cast dynamic shadows like all other objects. Whether static shadows and occlusion are displayed for plants is strictly a property of the plant type and cannot be changed in the Editor.

Bug: Daytime static plant shadows are calculated as if the plant is standing in its default orientation. Nighttime static plant shadows are calculated using the actual plant orientation.



Grass never casts shadows of any type.

Procedural Snow

A winter map adds snow to the foliage and to certain solid parts of a plant. See page 173 for details.

Level of Detail

As with models (page 133), plants and grasses have multiple levels of detail.

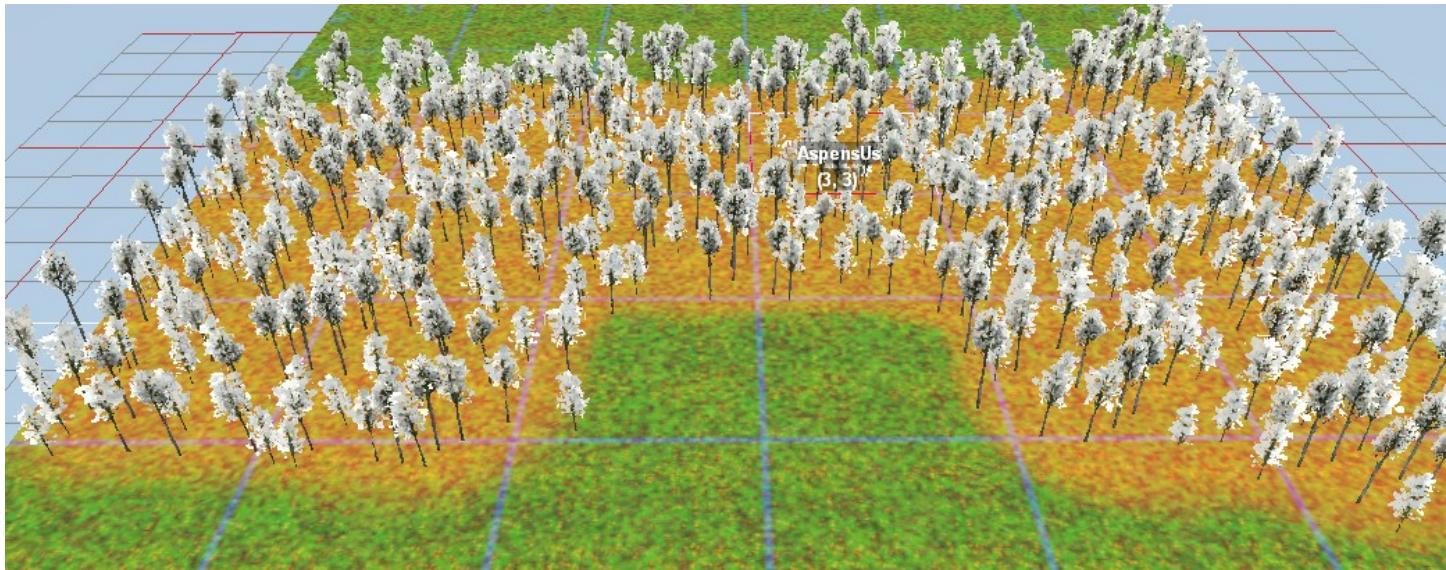
The level of detail changes for grass tends to make it look patchy when viewed in the Editor. However, when viewed at lower angles as is usual in the game, it looks fine.

Landmarks

Large trees come with a simplified 3-D model, called a “landmark”, that is used on the game’s navigation map. Smaller plants and grass do not have landmarks and are not shown on the navigation map.

Distributions

Plants, grass, and debris can be quickly created in large quantities using distributions. To create a new distribution, select **Add Distribution** from the context menu.



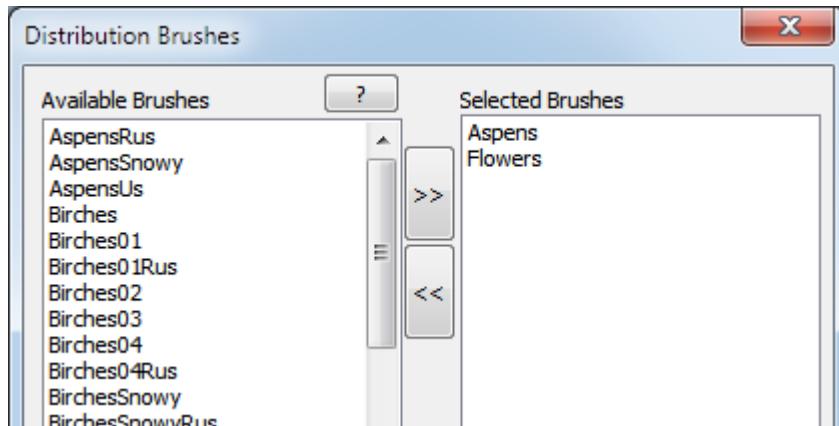
Choose Plants

Scene.(Terrain).Distributions.*brush_list*: **Brushes:** brush list

Specifies the plants to randomly place in the painted distribution areas.

Default: blank; no plants are placed.

Choose what plants will be in the distribution by clicking **[press]** next to the **Edit** property in the **Brushes** category. A window appears allowing you to choose as many distribution brushes as you desire.



Move a brush from the left **Available Brushes** column to the right **Selected Brushes** column by clicking its name and then the **>>** button. Move it back by selecting it from the right column and clicking **<<**. Or quickly change a brush from one side to the other by double clicking it in either column.

Each selected brush includes one or more plants in the distribution. Which plants are included is usually somewhat apparent from the brush name, but the distinction between, say, **Birches**, **Birches01**, **Birches02**, **Birches03**, and **Birches04** can only be discovered in the Editor by experimentation. You can also inspect the **brushes.xml** file outside the Editor. Its format is described on page 430.

The list of brush names becomes the label for the distribution in the **Scene View**.

Choose a Bitmap Filename

Scene.(Terrain).Distributions.*brush_list*: Map: filename

Specifies the bitmap file where the painted distribution is stored.

Default: blank; the distribution cannot be painted.

Each distribution must be assigned an associated bitmap file in its **Map** property.

To enter the bitmap filename, click on the property, then click the **...** to the right of the value field. A file chooser opens to show your **prebuild/map_name** directory. For a new distribution, presumably you won't be selecting an existing file. Instead, type the new filename into the file chooser. The filename must be in the form **dstr_name.tga**. If your filename doesn't have the required **dstr_** prefix or **.tga** suffix, or if you change the path to something other than the **prebuild/map_name** directory, the Editor will complain and revert your change.

Bug: If you try to type in a filename directly, the Editor complains that it is in the wrong folder and reverts your change. It is possible to manually enter the folder and filename as **map_name/dstr_name.tga**, but it's probably easier to use the method above.

You can also select an existing bitmap file for the distribution as long as it is within the **prebuild/map_name** directory. For example, you might select a file that was previously used by a deleted distribution. You can even select the same file as another distribution in use. Normally you'd just add more plant brushes to an existing distribution, but adding a new distribution with the same painted area can be useful for certain purposes described in the sections below.

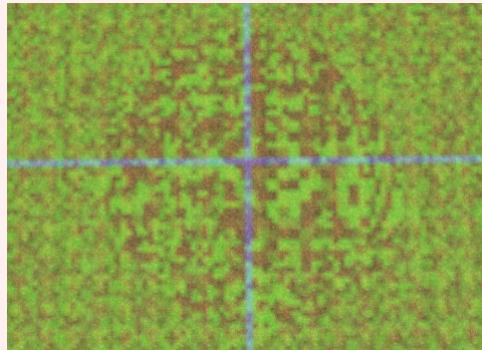
Paint the Distribution Region

Once you have chosen one or more brushes, use the distribution brush's **Density** mode to increase the density for the chosen plants where desired on the map. Density is painted using the red channel.

It is not unusual to paint most regions with 100% density, which actually corresponds to “100% of typical density”. You can paint with lighter density to distribute plants more sparsely than typical.

You can also switch the brush to the **Scale** mode to change the scale of plants in the distribution. Scale is painted using the green channel. By default, the scale is randomized across the entire range of scales configured for the distribution brushes, which is why the painted distribution is initially a speckled green pattern. You can bias the scale of your distribution by painting different scale values onto the map.

Bug: Once you have altered the painted scale channel, there is no way within the Editor to restore it to its original random distribution. If you repaint the scale with the **Randomize** checkbox enabled and the **Value** set to 1, this paints one third of the pixels with the minimum value, and most of the remaining pixels are painted with the same value across multi-pixel chunks. To restore true randomness using Gimp, see page 452.



After distributing plants onto the map, you can edit your distribution by changing what plants are part of the distribution and/or by painting or erasing different areas. This flexibility allows you to quickly and easily distribute plants across the landscape in a natural-looking fashion.

Plant Density Restrictions

A plant’s distribution is limited by the density specified by the distribution brush and also by the minimum spacing between plants in the same family, e.g. big trees, small trees, grass, and debris. Thus, if your distribution includes many different plants in the same family, the frequency of each plant is often reduced.

The order of distributions and the order of brushes within a distribution influences how many plants each brush gets to distribute. The first distribution and the first brush in a distribution have the first distribution opportunity. The remaining brushes and distributions can only place plants where they can find an appropriate gap (if any).

There are a few plants (such as mushrooms) which default to a low density even when the distribution is drawn with maximum density. For these cases, overlaying a second distribution with the same plant can increase the overall density. For most large plants, however, using the same plant in multiple distributions will result in little or no increase in density.

Tip: If a plant's density is not limited by its minimum spacing, you can increase the plant's density by setting up two or more distributions using the same plant brush **and** the same bitmap filename.

Besides avoiding other distributed plants, the Editor avoids distributing plants near individually placed plants in the same category. So you can manually place plants as needed in critical locations (e.g. as winch points), and the randomly distributed plants will fill in around them without overlap.

Scale-Dependent Brushes

Many of the distribution brushes limit the plant choices at each location based on the painted scale at that position. The effect is hard to notice with the default randomization of scale, but you'll notice a shift in the proportion of plant types if you bias the scale up or down (or toward the middle).

Grass

This book tends to lump plants and grasses together, but I should mention that they are completely different categories of objects. The category of “grasses” also includes small debris such as rocks and mud chunks.

Grass cannot be individually place. Instead, grass can come from three sources:

- A grass selected as a distribution brush. Grasses are labeled in the brush list with a **(grass)** suffix.
- A grass that is silently included by a plant distribution brush. E.g., the **Birches** brush automatically includes the **GrassTallSummerB** grass brush.
- A grass that is invoked by a material layer. E.g. the **default** material texture includes **GrassShort** and **GrassShortAu**.

Bug: Sometimes, grasses are placed extremely sparsely. It doesn't always happen, and when it happens, it doesn't always happen everywhere on the map. But when and where it does happen, it makes the grass look like ass. I can't figure out what triggers the bug, but it seems to have something to do with multiple grasses in the same distribution, whether selected as a grass brush or included automatically from a plant brush, or with multiple grasses associated with the same material.

Tip: Rather than include multiple grasses in one distribution, you can put them in separate distributions. These distributions can share the same bitmap filename to paint the grasses in the same places.

Most grasses are planted in tufts that have a relatively large footprint. This may mean that a tuft of grass is placed where its base point is legal, but the grass extends into areas where it shouldn't (e.g. outside of the distribution or onto the edges of a road). If this bothers you, you'll have to trim back the distribution region to avoid it.

Automatic Distribution after Painting

By default, plant locations are updated whenever you commit a change to the painted bitmap. Only the terrain blocks that were touched are rebuilt, and only the main plants are placed or removed, not the grass. This is only a redistribution of plants, not a rebuild of terrain, so shadows are not updated.

You can disable this automatic distribution rebuild by disabling the [Enable autorebuild terrain](#) toggle button on the toolbar (page 256).

To fully update the map, manually rebuild the terrain (page 65). A manual rebuild is also required whenever you change the distribution's named properties.

Random Seed

Plants are distributed randomly up to the mapped density. (There's no reason to use the Randomize checkbox for density because doing so just decreases the average density.) The random distribution is not directly under your control, but the Editor initializes the random function with a fixed value which you can control. This value is kept in the [Seed](#) property. If you don't like the random placement of plants in the distribution, you can choose a different seed value.

Scene.(Terrain).Distributions.*brush_list*.[Seed](#): [Seed](#): integer

Specifies the seed value for the random distribution.

Default: small random integer.

You might want to edit the seed value if the random distribution has a random gap in a highly visible location. Different seed values will not help you if your density bitmap isn't quite right, however. In most cases, the best way to fix the distribution is to carve out a region of zero density where plants are undesired, draw a larger region of maximum density to get more plants, or add individual plants where desired.

Be aware that even with a particular seed value, any changes to the painted distribution region or to other distributions may cause a different random placement of plants. Plan to review the critical areas of your distributions for misplaced plants before publishing.

Interaction with Roads, Water, and Mud

By default, plants are not distributed in water, on a road, or in mud. However, you can tell the distribution to ignore the road, water, or mud and therefore distribute the plants at that location anyway.

Scene.(Terrain).Distributions.*brush_list*: Ignore Overlays: True/False

Specifies whether the distribution can be placed on roads.

Default: **False**.

The **Ignore Overlays** property applies only to road overlays. To avoid other overlays (especially conforming overlays), you have to manually paint a region of zero density in your distribution around each overlay.

Scene.(Terrain).Distributions.*brush_list*: Ignore Water: True/False

Specifies whether the distribution can be placed underwater.

Default: **False**.

A river only prevents distribution if **Ignore Water** is **False** and the river is visible above ground. I.e. a portion of river underground does not prevent distribution of plants above it.

Scene.(Terrain).Distributions.*brush_list*: Ignore Mud: True/False

Specifies whether the distribution can be placed in automatic mud.

Default: **False**.

The **Ignore Mud** property only applies to mud created with the **Automatic** brush (visible mud). This also includes automatic mud created using the **QuickMud** feature. Mud created with the **Extrudes** brush never affects distributions.

The above properties apply only to newly painted distributions over an existing road, river, or automatic mud. If you place a new road or river, you have to manually rebuild the terrain (page 65) to exclude plants from that region.

You may occasionally see plants being drawn in water, on a road, or in mud even when these checkboxes are not checked. This often occurs because the Editor has rebuilt the terrain using one of the shortcut methods (page 65). To fix it, manually rebuild the terrain.

Interaction with Material Textures

Material textures can automatically include grass and/or exclude certain plants and grasses. Details are on page 160.

Model Occlusion

If the bounding box of a model touches the terrain anywhere (so it is not entirely above or below ground), then plant and material distributions are excluded from the region of its occlusion shadow. Note that not all models have an occlusion shadow.



Distribution Summary

Plants may be distributed:

- Where a distribution includes the plant and is painted with non-zero density, or
- Where included by a material layer (grass only).

Plants are never distributed:

- On a road, in water, or in automatic mud, if the corresponding distribution has `Ignore Type` = `False`.
- On a material layer that excludes the plant.
- In deep snow.
- In the occlusion shadow of a model.
- In a scale-dependent distribution where the painted scale is outside the range for the plant.
- Near an individually placed plant or previously distributed plant in the same family.

Materials

The **PbrMaterials** category in the **Scene View** holds one or more materials that make up the surface of the terrain. These materials include surfaces such as grass, dirt, gravel, sand, asphalt, and snow. Each possible surface is referred to as a material layer.

A material layer describes the texture (2-D appearance) of the terrain, i.e. the bitmap that is applied to the terrain. It also adds 3-D grass (or debris) to distribute on the terrain surface and/or limits plants and grass from other distributions. These positive and negative effects on distribution are described on page 160.

The material layer also describes intangible aspects that can't be seen in the Editor, e.g. whether dust can be kicked up from the terrain and what it sounds like when tires skid on the terrain. For ease of description, I'll mostly refer to the materials' impact on the texture of the terrain. The impact on other aspects is implied.

One terrain block (24m×24m square of the terrain) can use only one material, but that material may include up to four material layers arranged however you like across that terrain block. I'll begin by describing how material layers are handled in one material before we start adding more materials.

Material Properties

Your map is initialized with its first material in [Scene.\(Terrain\).PbrMaterials.\(PBR Terrain Material\)](#). Click on it to bring up its properties. This also brings up a brush tool, but we'll deal with that in the next section.

Name	
Material	
Albedo wetness mult	1.000000
Roughness wetness mult	1.000000
Mask channel index	0
Layer 1 (base)	
File	default
[Choose File]	[press]
Tiling scale	1.000000
Layer 2	
File	
[Choose File]	[press]
Tiling scale	1.000000
HM blending contrast	0.900000
Mask channel index	-1
Layer 3	
File	
[Choose File]	[press]
Tiling scale	1.000000
HM blending contrast	0.900000
Mask channel index	-1
Layer 4	
File	
[Choose File]	[press]
Tiling scale	1.000000
HM blending contrast	0.900000
Mask channel index	-1

[Scene.\(Terrain\).PbrMaterials.name](#): Name: text

Specifies the name of the material. This is only displayed in the [Scene View](#) and has no other purpose.

Default: blank; displays as [\(PBR Terrain Material\)](#).

To help keep your materials straight, you can assign a name to each material in its [Name](#) property. The name has no game purpose, however. Names do not need to be unique.

[Scene.\(Terrain\).PbrMaterials.name](#): Albedo wetness mult: numeric

Specifies how much that wetness darkens the tint of the terrain. A higher value leads to darker wetness. A value of zero removes the tinting effect of wetness.

Default: 1.0.

Scene.(Terrain).PbrMaterials.name: `Roughness wetness mult`: numeric

Specifies how much that wetness makes the terrain look smoother. A higher value leads to more smoothness. A value of zero removes the smoothing effect of wetness.

Default: 1.0.

Note that the wetness-related properties are per material, not per material layer. This makes it difficult to use different values in different parts of the map, and you will likely end up using the same values for all materials on your map.

Tip: Saber recommends for winter environments `AlbedoWetnessMult` = 1.0 and `RoughnessWetnessMult` = 1.0 (representing ice). They recommend for warmer environments `AlbedoWetnessMult` = 2.0 and `RoughnessWetnessMult` = 0.5 (representing water).

100% wetness reaches the maximum darkness even with `AlbedoWetnessMult` = 0.5, so higher values only affect how regions of partial wetness are drawn. Negative values of `AlbedoWetnessMult` are treated as 0.0.

100% wetness reaches the maximum smoothness with `AlbedoWetnessMult` = 2.0. Negative values create a bright ring around the edge of wet areas, so you should keep your values to 0.0 or higher.

Scene.(Terrain).PbrMaterials.(PBR Terrain Material): `Mask channel index`: read-only integer

Displays the material number, starting with 0 for the first material.

Material Layer Properties

The remaining properties for the material are specific to each material layer.

Scene.(Terrain).PbrMaterials.name: `Layer 1 (base).File`: asset selection

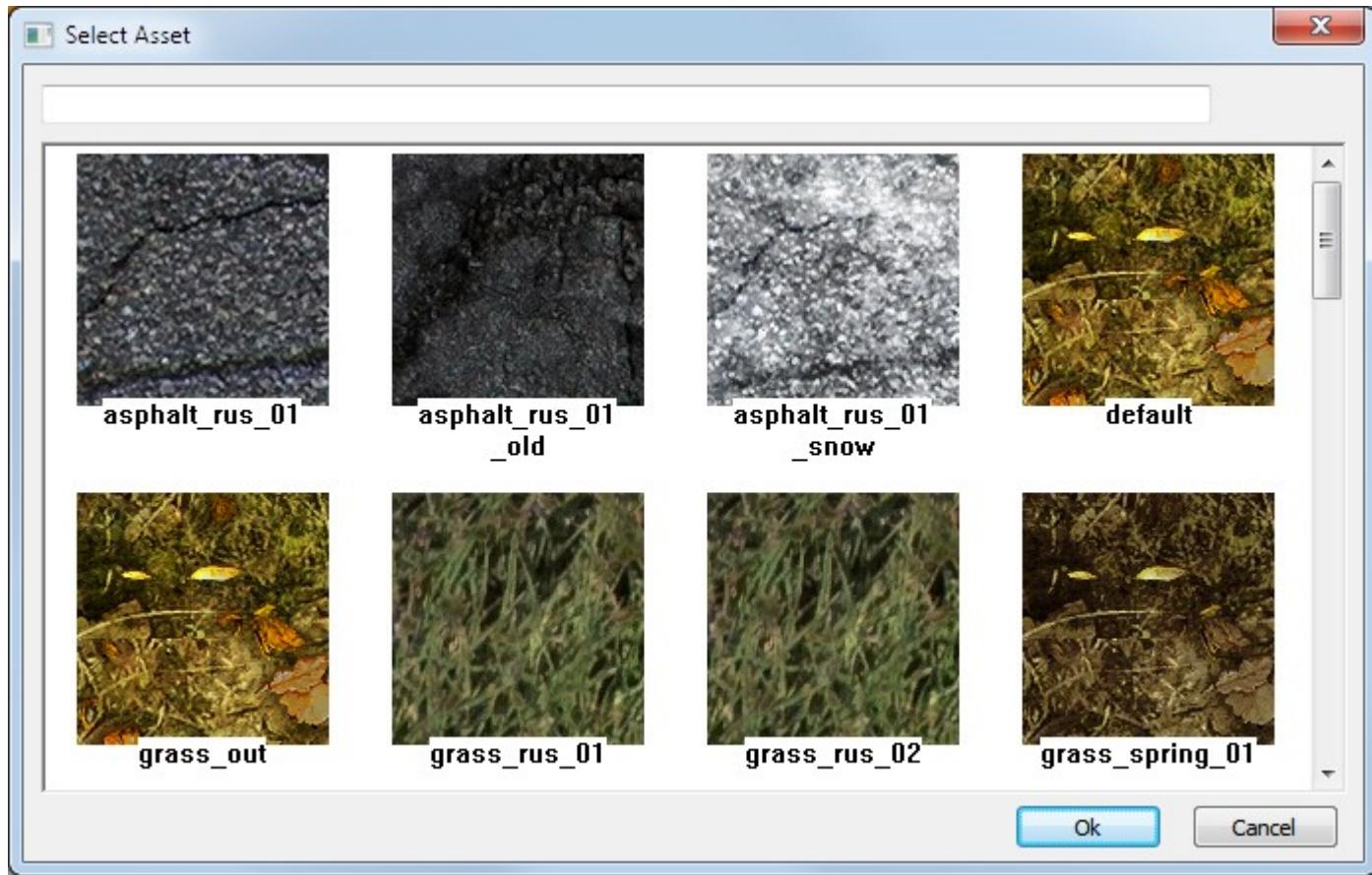
Specifies the texture for the layer.

Default: `default`; grass and autumn leaves.

Scene.(Terrain).PbrMaterials.name: `Layer n.File`: asset selection

Specifies the texture for the layer.

Default: blank; using this material layer is like looking into the inky depths of space.



If you skip a layer when you assign textures, the Editor will shuffle the other material layers to fill in the gap the next time you rebuild the terrain (page 65).

Scene.(Terrain).PbrMaterials.name: Layer 1 (base).**Tiling scale:** numeric
Scene.(Terrain).PbrMaterials.name: Layer n.**Tiling scale:** numeric

Specifies the factor by which the texture pattern is shrunk.

Default: 1.0.

By default, the texture is tiled in a 2×2 pattern to fill each terrain block. But for textures that require finer detail, we want smaller tiles, with more of them to fill a block. The **Tiling scale** property specifies how much smaller each copy of the texture should be. E.g. a value of 5 shrinks the texture by a factor of 5, so it takes a 10×10 pattern of tiles to fill a terrain block. The value does not need to be an integer. If the edge of a terrain block is reached in the middle of a tile, the tile continues smoothly into the next terrain block as long as it uses the same texture.

Tip: A good-looking **Tiling scale** is **not** 1.0 for most terrain layers. See the Material Layer Reference section on page 398 for recommended values.

If you're so inclined, you can use a negative value to flip the terrain texture.

Layers 2 – 4 have additional properties that are not used in the base layer.

Scene.(Terrain).PbrMaterials.*name*: Layer *n*. HM blending contrast: numeric

Specifies the sharpness of the blend line between different textures within a material.

Default: 0.9.

When different textures within the same material are placed next to each other on the terrain, the edges of those textures are blended together to avoid a sharp demarcation. The **HM blending contrast** property specifies the sharpness of that blend line. The effects of this property are described in more detail on page 161.

Values are constrained to an effective range of 0.0 to 1.0.

Tip: Saber recommends the following **HM blending contrast** values for various material layers:

- soft surfaces: 0.7 – 0.8
- hard surfaces: 0.8 – 0.9

Tip: Arrange a harder layer above a softer layer so that its greater **HM blending contrast** is used.

Scene.(Terrain).PbrMaterials.*name*: Layer *n*. Mask channel index: integer

Displays the channel number for the material within the **mtrl_base.blends.tga** bitmap (page 448): 0 for the red channel, 1 for the green channel, 2 for the blue channel, or -1 if a texture is not assigned.

The **Mask channel index** is always 2 less than the layer number once the Editor has closed any gaps in the layer usage. E.g. layer 3 uses **Mask channel index** 1. This per-layer property is separate from the per-material **Mask channel index** described above.

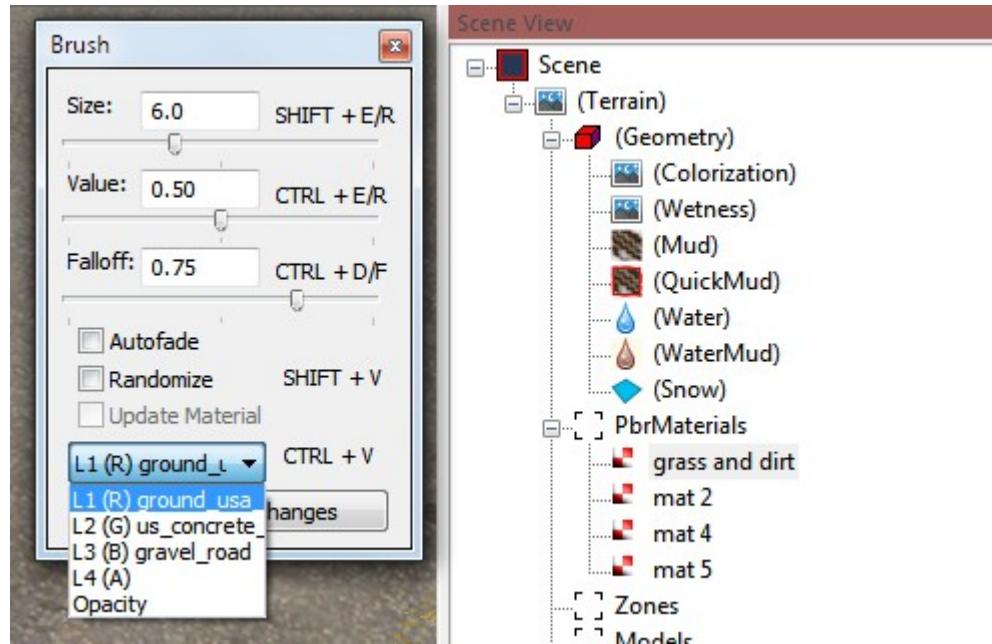
Paint a Material Layer

Texture is painted onto the terrain in 0.5×0.5 meter squares. Each square typically displays the characteristics of a single material layer, but some blending can occur as described beginning on page 160.



Textures are painted onto the terrain in layers like separate coats of paint. Each texture uses a stencil so that it is painted in some squares but in others. Whichever texture is painted last is usually the most visible, but a weakly painted texture allows color to bleed through from the textures underneath.

To begin painting, click the desired material under **PbrMaterials**, then select the desired texture in the mode selection dropdown of the **Brush** dialog box.



The bottommost texture is always applied across the entire terrain block, so it cannot be selected in the **Brush** dialog. Each of the three other textures in the material can be selected and painted over top.

Bug: The three other textures are labeled **Layer 2**, **Layer 3**, and **Layer 4** in the property panel, but they are confusingly abbreviated **L1**, **L2**, and **L3** in the Brush dialog.

The **L4 (A)** brush mode is used only for snow (page 169). The **Opacity** brush mode is used to select which material to apply (page 162).

A **Value** greater than 0.50 increases the strength of the texture, and a **Value** less than 0.50 decreases the strength of the texture. When increasing the strength, the brush preview disc is red (**R**), green (**G**), or blue (**B**), depending on which layer is selected. When decreasing the strength of any layer, the brush preview disc is black.

Remember that the paint is layered, so if you paint a strong **Layer 4 (L3)**, nothing you paint in the other layers is immediately visible. However, if you then erase some of the paint from **Layer 4**, your changes to the lower layers become visible. (Computer graphics: the only time you can paint **under** existing paint.)

Plants, Grass, and Debris

An exposed material layer can include 3-D grass and debris in addition to its obvious surface texture. The Editor classifies all such 3-D objects as “grass”, regardless of their appearance. Grass from a material is always excluded from roads, water, and automatic mud. This is unlike grass from distributions, where you have a choice about their exclusion (page 151).

Tip: Grass from materials tends to grow through the roadbed of a bridge where it meets the terrain. Extend the road overlay under the lip of the bridge to block grass from being placed there.

An exposed material layer can also block plants and/or grass from distributions (page 147). The exclusions mostly make sense (`us_concrete` blocks most grass and plants, but fallen leaves and cones are allowed), but they include some real head scratchers (`us_concrete` allows pine trees).

The above two paragraphs refer to “exposed” material layers. If a material layer is entirely painted over at some location, then it neither contributes grass nor blocks distributions at that location. Only the top-most layer counts. Only if the top-most layer is not painted at full strength can one or more lower layers be blended in. Blending is described in the next section.

After painting or removing a material layer, rebuild the terrain (page 65) to redraw plants and grass in the correct places.

Blending

Value Blending

Where a material layer is not painted at full strength, the display engine blends it with the lower material layers. There is some blending at the pixel level, but mostly the engine looks for the implied high spots in each texture and gives them priority for display

The below screenshot shows an extreme close-up of a gravel texture that is painted only weakly over a grass and leaves texture in the center left and a concrete texture in the center right. The images of the gravel pebbles dominate the grass and leaves, but the grass and leaves texture is stronger in the gaps between gravel pebbles. Interestingly, even though the gravel is painted over the concrete, our eyes interpret it as a thin layer of concrete partially covering the low spots in the gravel.



Edge Blending

When a layer of paint has little falloff between regions of “paint” and “no paint”, then blending with the lower layers is based on the [HM blending contrast](#) property of the upper layer. The below screenshots show the same edge in which the concrete has an [HM blending contrast](#) of 0.6 on the left and 1.0 on the right. With a value of 1.0, the edge wiggles a bit to follow the implied high spots in each texture, but colors are never blended between the two textures.



These screenshots also illustrate that even though paint is applied in 0.5×0.5 meter squares, the display engine rounds off corners to avoid an unnatural-looking sharp bend. It also uses diagonal lines to connect paint squares that are painted in a stairstep fashion.

When the **HM blending contrast** property value is below about 0.55, lower material layers start to bleed through even where the upper layer is at full strength. See the Material Properties section on page 155 for recommended values.

Blend of Grasses

Where two material layers are smoothly blended, grasses from each texture can be intermixed depending on the strength of each paint.

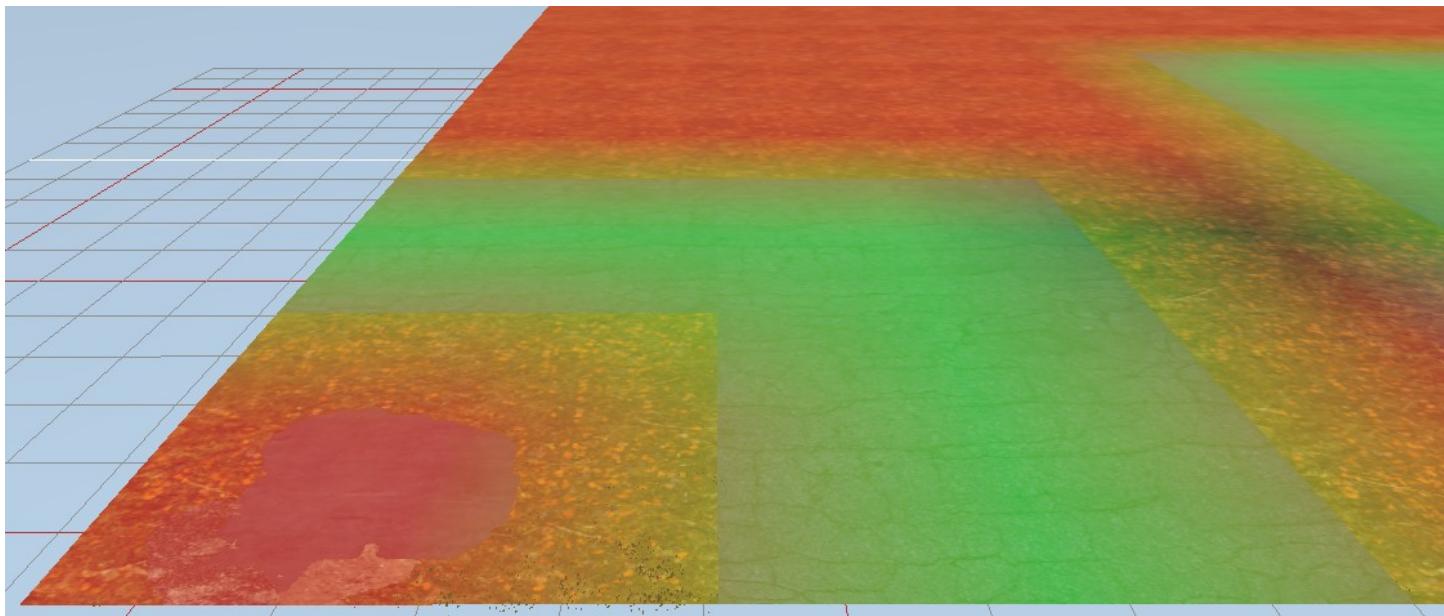
Where two material layers have a sharp line dividing them, a grass from one may intrude on the other simply because of the size of the grass tuft (or other “grass” object). This is especially noticeable when viewed closely. However, it is much less noticeable with the typically high camera position while driving in the game.



Additional Materials

To add additional materials, select **Add Pbr Material** from the context menu. Each material can have its own combination of material layers.

Each terrain block can use only a single material. The material used in each terrain block is chosen by painting the material onto that terrain block. To begin painting, click the desired material under **PbrMaterials**, then select **Opacity** in the mode selection dropdown of the **Brush** dialog box. Because the default material layer may be shared across different materials, the Editor helps distinguish which material is used where by tinting them in different colors.



Each map can use up to four different materials. While painting with [Opacity](#), each material is tinted with a different color:

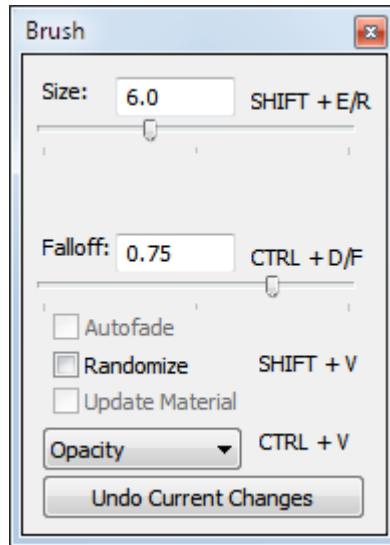
- The first material is tinted red.
- The second material is tinted green.
- The third material is tinted blue.
- The fourth material is tinted black.

Bug: Although each material has a distinct boundary at the edge of a terrain block, the tinting is smoothly blended between the centers of terrain blocks. This can make it hard to discern the true tint of small regions of material.

Although you can add more than four materials and attempt to paint with these additional materials, the changes revert as soon as you [Rebuild Terrain](#).

Bug: Saber documents that the Editor support up to four materials on a map, but painting with the fourth material doesn't work correctly. If you [Rebuild Terrain](#), these terrain blocks revert to showing the first material instead. This can be fixed up by hand with some bitmap channel editing outside of the Editor. See page 446 for details.

Painting with **Opacity** is all or nothing, so there is no **Value** parameter. The **Size** allows you to potentially paint many terrain blocks at once. Oddly, the **Falloff** parameter can still be adjusted, but it does nothing.



To paint the selected material onto the terrain with **Opacity**, right click on the desired block. Or right click and drag to quickly paint multiple blocks. There is no brush preview disc. As long as the **Size** is small, it's simply the terrain block under the mouse. If the **Size** is large, it is the many terrain blocks within that range of the mouse.

Material **Opacity** is somewhat different than other painting modes in that selecting a different material requires a new selection from the **Scene View** rather than a quick change to the **Value** or brush mode.

Bug: While painting with **Opacity**, right click updates both the artificial tinting and the textures on the terrain to match the new material. However, **Undo Current Changes** only reverts the changes to the tint. To correctly revert the textures on the terrain you must then select **Rebuild Terrain** from the context menu.

When you change the material assigned to a terrain block, the paint stencils for layers 1 – 4 are retained, although they may now point to different textures in the new material.

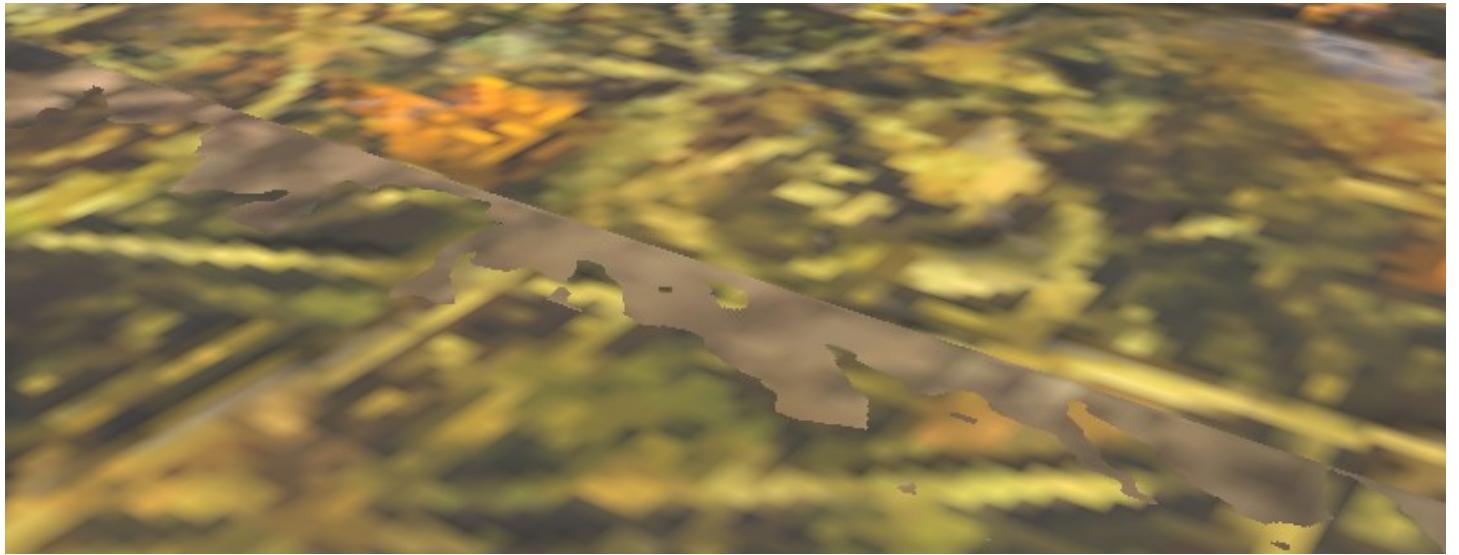
Texture Edges Between Materials

Different terrain blocks can use different materials and thus different combinations of material layers. However, adjacent terrain blocks must have at least one matching material layer which is used along the boundary between the terrain blocks. Otherwise, a sharp line results where the texture suddenly switches from one block to the other.



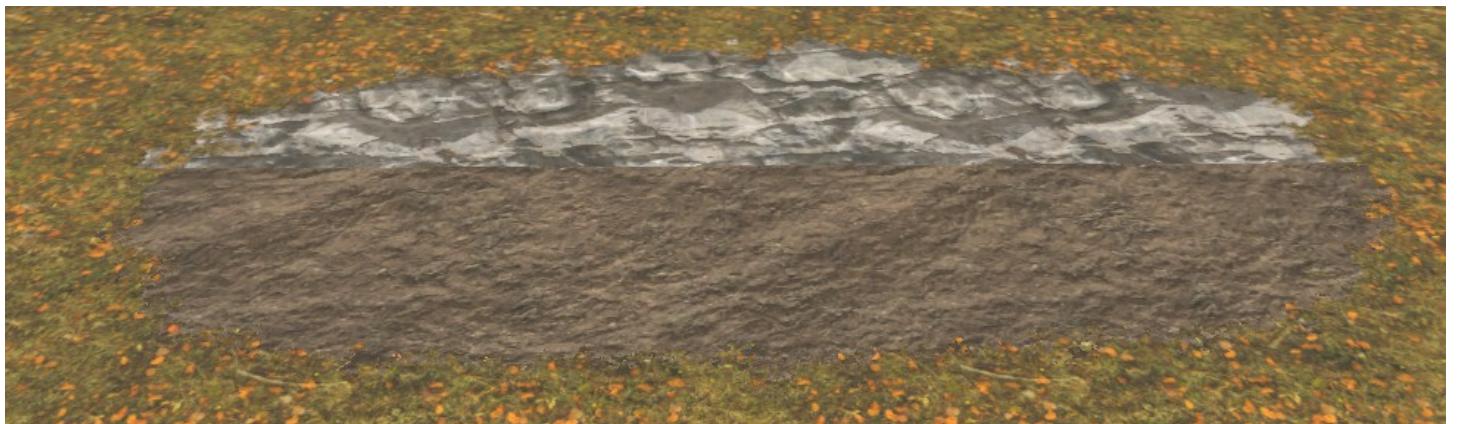
Multiple material layers can be blended at a boundary between terrain blocks, but only if the materials in both blocks use the same textures in all blended layers. Needing to share multiple layers across materials sharply limits how many different material layers can be used throughout the map, so typically you will share only one material layer and avoid any blending of layers near material boundaries.

BTW, if two adjacent materials use the same texture, but in different layers, the display engine blends between the layers separately on each side of the boundary. I.e. if a block uses grass in layer 0 adjacent to grass in layer 1 in an adjacent block, the first block displays a blend of layer 0 and layer 1 near the boundary. Since layer 1 is some other texture, this results in color bleed from an unwanted texture and a sharp line at the block boundary. Therefore, textures shared across materials should always be shared in the same layer.



If four materials are used in a map, and each material holds up to four material layers, but at least one texture is shared across each pair of materials, then the maximum total number of textures per map is $4+3+3+3 = 13$.

Caution: Although you select a particular material to begin painting a material layer, the brush actually paints all materials with the selected material layer number. So if you paint across the boundary of two materials, the painted texture takes on the characteristics of whatever texture is assigned to that layer in each material.



Snow Layer

The snow_layer texture is special. See page 169 for details.

Procedural Snow

A winter map adds snow to the `stone_snowy` and `rocks_rus_01_snowy` material layers. See page 173 for details.

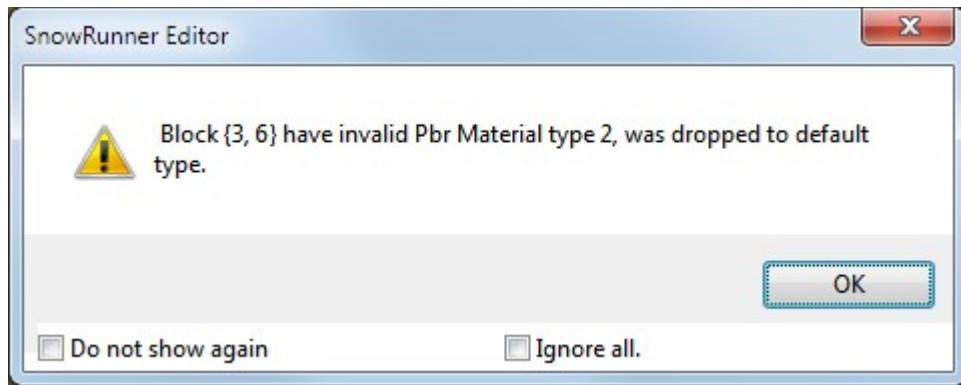
There are other `snow`-related material layers, but these are the only ones that support procedural snow.

Delete a Material

You can delete a material using the context menu. Deleting the last material behaves somewhat differently than deleting an earlier material. Each scenario is discussed below.

Bug: Unless you have only recently loaded your map into the Editor, deleting the last material will likely crash the Editor.

When you delete the last material, the Editor does not update the terrain, and you have the opportunity to add a new material to take the place of the deleted one. But if you rebuild the terrain, the Editor displays a warning dialog for each terrain block that used the deleted material. Although the Editor says it drops the terrain block to the default material, this is only temporary during the rebuild. Repaint the opacity of the terrain blocks so that they use valid materials.



When you delete any material other than the last, the Editor makes the following changes:

- Any terrain blocks that use that material revert to using the first material instead. (This is the new first material if the previous first material was deleted.)
- The `Mask channel index` of all following layers are updated.
- The hidden bitmap that assigns materials to terrain blocks is updated with the new channel index values.

Bug: The Editor may render the terrain oddly after you delete a material. Rebuild the terrain to fix it.

Bug: The Editor is guaranteed to crash if you rebuild the terrain when there are no materials. If the map has only one material, and you want to delete it, I recommend that you first add a new material, then delete the old one.

Winter Map

SnowRunner draws a fundamental distinction between a winter map and a map set in another season. A winter map includes full support for snowy features, but it also somewhat reduces the support for non-snowy features.

A map is recognized as a winter map when it includes the `snow_layer` texture in **Layer 3** of any material (whether or not that material or material layer is ever used on the map).

The following features are enabled only for a winter map:

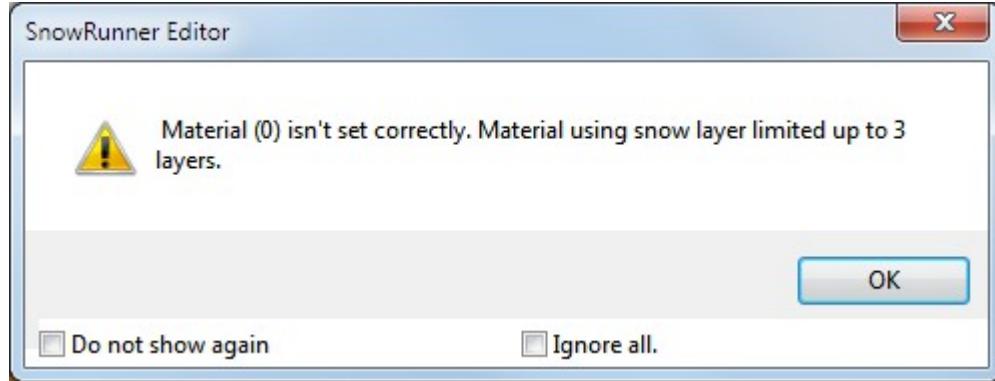
- Snow depth (page 171).
- Procedural snow coverage (page 173).

Snowy weather (precipitation) is not explicitly tied to a winter map, but you'll probably want the right weather for your map. Weather is selected via the daytime presets, described on page 253.

When working with a lot of snow, all that white on the screen can be hard on your eyes. The **Terrain brightness** button on the toolbar displays a dialog box that lets you adjust the brightness of the terrain within the Editor only. Other map features are not affected.

Snow Layers

The `snow_layer` texture is special and is officially supported only in **Layer 3** of a material. In order to assign `snow_layer` to **Layer 3**, **Layer 4** must not have any texture assigned to it.



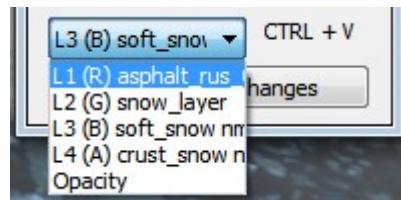
Bug: There is no way in the Editor to remove a texture that has been assigned to a layer, so once **Layer 4** has a texture assigned, **snow_layer** cannot be added to that material. Instead, you must create a new material and then delete the old one. See page 167 for details and cautions regarding deleting material.

Bug: When a **snow_layer** texture is used by any material, **Layer 4** is not drawn for other materials. I.e. materials without a **snow_layer** are limited to three material layers on a winter map.

Once assigned, the **snow_layer** texture can be painted like any other material layer. It can also be painted automatically while painting snow depth (page 172).

Soft Snow and Crust Snow

The **snow_layer** texture comes with two additional snow variations that can also be painted as layers. To get these additional layers, first assign the **snow_layer** texture to **Layer 3**, then rebuild the terrain (page 65). If you re-select the material, the brush now shows two additional paint layers: **soft_snow** and **crust_snow**.



To paint with soft snow or crust snow, first paint with **snow_layer**, then paint the **soft_snow** or **crust_snow** modifier on top of it. These are not true material layers, but modifiers of the **snow_layer**.

Bug: If **soft_snow** is painted outside the **snow_layer** area, it has no effect on the 2-D texture, but it does prevent the non-snow material in **Layer 1** or **Layer 2** from distributing grass. This bug applies only to **soft_snow**, not **crust_snow**. So be careful where you paint your **soft_snow**.

Soft snow is snow in which patches are partially melted (softened). It is mostly associated with plants since plants tend to retain heat longer than the air, and plants also provide a dappled mix of sun and shade.

Crust snow is snow in which chunks of snow have partially melted and then refrozen, so it is a mix of snow and chunks of air-filled ice. It is mostly associated with snow that has been plowed or blown off the road. It may also appear near fast running stream banks where water has splashed out and frozen.

Bug: **L4 (A)** is initialized to fully painted. You must erase **L4 (A) crust_snow** across your entire map if you want anything other than **crust_snow** anywhere.

Tip: If you previously drew anything in L3 (B), you should erase that, too. Once you have a snow_layer anywhere, you can't use Layer 4 for anything except soft_snow anyway, and you don't want any leftover invisible paint from another material to apply soft_snow where you don't want it.

The snow types differ only in how the display engine renders the light bouncing off the snow. Unfortunately, the display engine used by the Editor doesn't model this effect, so it can only be seen in the game.

Bug: The Editor could assist in distinguishing the snow types, e.g. by applying a tint while a snowy texture is selected, but it does not. This means that there is absolutely no way in the Editor to tell where the different snow types have been painted. So paint with care, and try to do it only once, e.g. once everything else is nailed down. See also page 448 for how to use colorization with the aid of an image editor to help paint and edit these snow layers more accurately.

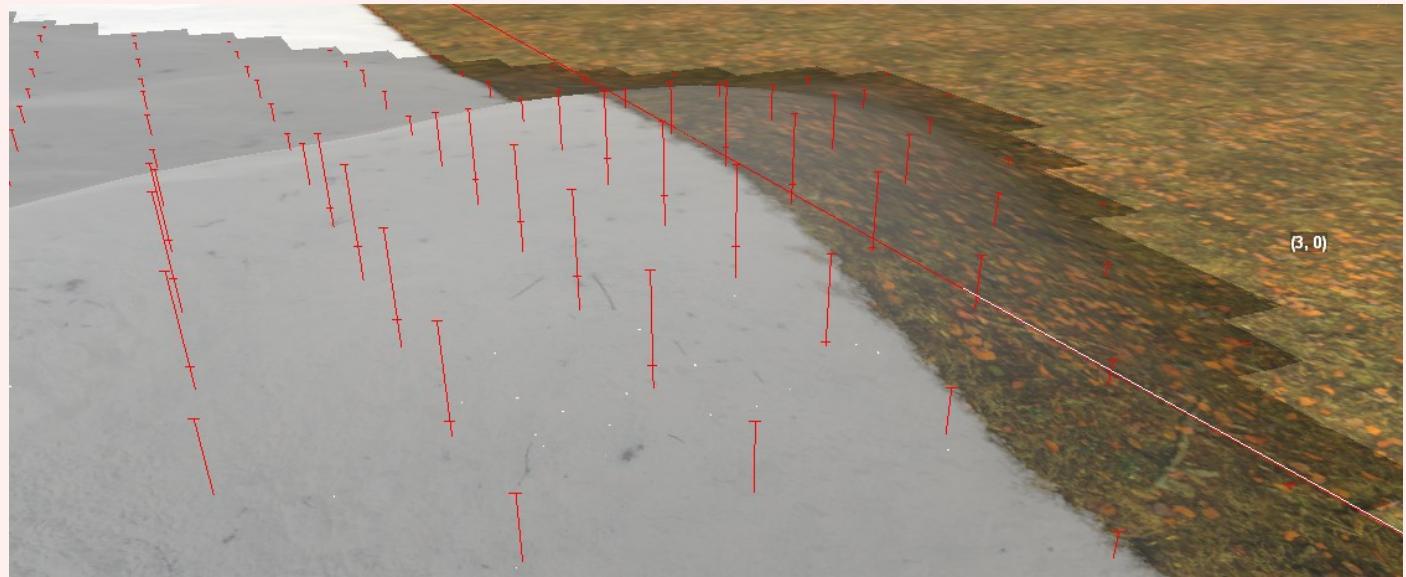
The difference between types of snow is very subtle in the game. I arranged the below screenshot with the best lighting and camera angle that I could find, using sharp edges between the snow regions to make them more apparent. There is a slanted vertical stripe of soft_snow on the left and another slanted vertical stripe of crust_snow on the right. The normal snow_layer is in the center and at the edges.



Snow Depth

Select (Geometry) → (Snow) to paint deep snow on the terrain. By default, deep snow can only be painted where the snow_layer texture is present. The snow depth is ignored everywhere else.

Bug: Snow depth is ignored when it is painted elsewhere on a material that uses the `snow_layer`. However, if it is painted on a material that doesn't use `snow_layer`, it artificially raises the height of the terrain texture (as if there is snow there), but doesn't create any visual or physics effect associated with snow.



Paint snow depth by selecting `Depth` as the brush mode. This could alternatively be called snow height since the snow is piled on top of the base terrain height.

The `Value` range is -4.0 to +4.0. As with terrain height, whatever `Value` you set determines how much the snow depth changes with a single right-click and drag, regardless of how many times you drag over the same area with the mouse. You have to release the mouse button and make another pass to change the height further.

Bug: Unlike anywhere else, 1.0 in the `Value` here corresponds to 20 centimeters, so +4.0 corresponds to +80 centimeters of snow depth.

While painting, the terrain is dynamically updated, but the Editor also draws thin red lines to help illustrate the depth of the snow. Tick marks are spaced every 0.5 meters along each line, starting from the top.

Although snow can be painted up to 3.2 meters deep, a truck can never sink more than about 40 cm into it. To allow a truck to sink deeper, you must paint mud under the snow. If the snow is deeper than the mud plus about 40 cm, then only snow is churned up as a truck sinks deep. But if the mud is deep enough to allow the truck to reach the base terrain, then it starts to churn up mud and soil as well.

Update Snow Layer

You can paint snow depth and the `snow_layer` material layer simultaneously onto the terrain. To do this, enable the `Update Material` checkbox in the `(Snow)` brush dialog. Any snow depth you add then automatically paints

`snow_layer` onto the terrain as well. Even the smallest amount of snow paints the `snow_layer` with an effective `Falloff` of zero, so it may need some touch-up afterward.

If the terrain material does not include the `snow_layer` texture in [Layer 2, Update Material](#) has no effect.

Bug: If you press `Ctrl+Z` or click [Undo Current Changes](#) in the brush dialog, it undoes the snow depth changes, but it doesn't undo the changes to the `snow_layer` material.

Interaction with Plants

Deep snow excludes plant distributions. Depending on the plant, the depth limit may be 50 cm or 200 cm. I can't find a property that determines how the snow depth translates to distribution exclusion, so this is just something you'll have to experiment with for yourself.

Conversely, some plants automatically remove snow depth around themselves. But if the plant is part of a distribution, it must be able to be placed (i.e. in shallow snow) in order to reduce the snow depth around itself.



Procedural Snow

A winter map can automatically add procedural snow to 3-D objects and certain material layers. This procedural snow adds a white tint and snow-like reflective glints to appropriate parts of an object, but it does not add any “thickness” to the snow.

Scene.(Terrain): Procedural Snow Coverage: numeric

Specifies the amount of snow on 3-D objects.

Default: 1.0, but the value is used only on a winter map.

Procedural snow is applied to models, plants (but not grasses), 3-D overlays and overlay brushes, trucks, and certain material layers.

There is only a single **Procedural Snow Coverage** value for each map, so there is no way to specify more snow coverage in some parts of the map and less coverage in other parts.

The Editor doesn't show procedural snow when an object is selected; it only shows procedural snow on deselected objects.

Bug: Although the game draws procedural snow only in a winter map, the Editor makes no such distinction. Instead, the Editor draws procedural snow based on whether the menu item **Settings → Show Snow By Up Vector** is enabled. Since this Editor setting applies to all maps, you have to toggle it yourself whenever you switch between a winter map and a non-winter map. Alternatively, you can set **Procedural Snow Coverage** to 0.0 on a non-winter map, but the Editor still applies a dim white haze to upper surfaces with this value.

Procedural Snow on Models

Procedural snow is initially drawn only on the upper surfaces of a model. Increasing values for **Procedural Snow Coverage** increase the opacity of the snow and the angles at which it sticks:

- For a surface that is perfectly horizontal and facing upward, values between 0.0 and 0.5 increase the opacity of the snow cover on that surface. At 0.5, the surface is entirely covered.
- Snow sticks to increasing angles from 0.0 until, at a value of about 1.95, it begins to stick to vertical surfaces.
- For a surface that is perfectly vertical (facing sideways), values between 1.95 and 2.5 increase the opacity of the snow cover on that surface.
- Snow sticks to increasing angles from 1.95 until, at a value of about 3.95, it begins to stick to horizontal surfaces facing **downward**.
- For a surface that is perfectly horizontal and facing downward, values between 3.95 and 4.5 increase the opacity of the snow cover on that surface. At 4.5, the surface is entirely covered, and greater values make no meaningful difference.

Snow coverage is patchy on large models, even with a high [Procedural Snow Coverage](#) value. The snow patches are more “streaky” than “patchy” at some angles. But the patches are based on a volumetric heuristic, so the degree of snow coverage is consistent within a local region, regardless of how the horizontal rotation of the surface. (Vertical rotation changes the amount of snow cover as described above.)



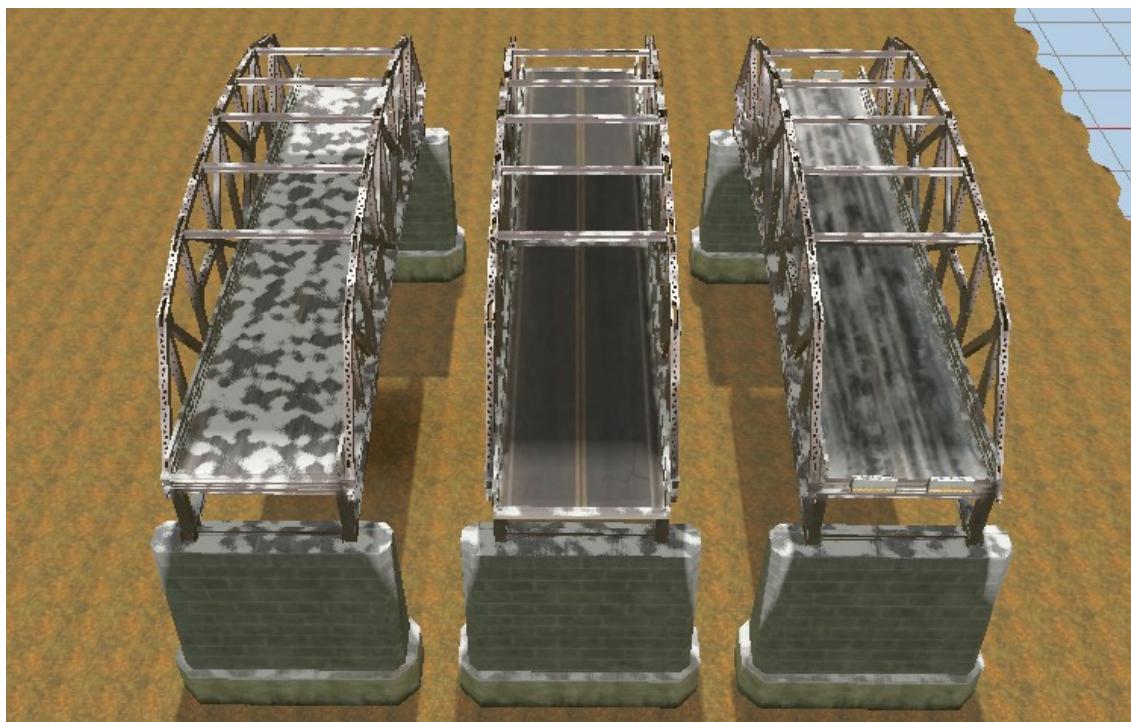
The size of snow patches depends on the built-in properties of the various model elements. E.g. some parts of a model may have larger snow patches, while other parts have smaller patches. For a given model element, the patches are always the same size, regardless of how the model is scaled.



A model can also disable procedural snow coverage for some elements of the model. Depending on the model, this may apply only at high levels of detail. (The level of detail scales with the model, which explains why one

building's doors are clear of snow in the screenshot above, while the smaller buildings have procedural snow applied.)

The default procedural snow coverage works pretty well for most models even without any special model properties. However, some models need those built-in properties to look good. For example, the screenshot below shows `bridge_road_01` (which has the default patchy snow on the road), `bridge_road_01_a` (which fixes the road to be snow-free, roughly like a standard asphalt road overlay) and `bridge_road_01_sn_objective` (which also leaves off the procedural snow, but adds hand-painted snow to the road, making it appear identical to the `us_asphalt_road_snowy` road overlay).



For dynamic objects, procedural snow is applied based on the model's default orientation. Unlike static models, the snow on a dynamic model sticks to the same original surfaces even when it is tilted in the Editor. Likewise, snow remains on the same original surfaces regardless of how the dynamic model is punted around in the game.

Procedural Snow on Plants

Procedural snow cover is added to the solid surfaces of a plant in the same way as for models, above. However, unlike with dynamic models, procedural snow is added to a dynamic plant based on its orientation in the Editor.



For plant foliage, snow cover on the leaves increase until they are fully covered at a [Procedural Snow Coverage](#) value of about 1.9. The pattern of increasing snow coverage on foliage varies for each plant. E.g. for `bush_b_rus` with [Procedural Snow Coverage](#) = 1.0, roughly every other leaf is fully snow covered, while the remainder are mostly snow free. This appears to be implemented in a similar manner to the patchy snow coverage on models.

Procedural Snow on Trucks

Procedural snow cover is added to trucks in a similar way as for models, above. Like dynamic models, snow is drawn on the truck as if it is in its default orientation, even if it is initialized or driven (or flipped over) in a different orientation.

The biggest difference with trucks is that they behave as if the [Procedural Snow Coverage](#) is half of its assigned value, with a max of ~2.5. I.e. the most that a truck can be covered is down to its vertical surfaces.

Snow is not applied to glass, the truck interior, wheels, or tires. If a truck has both mud and snow on it, the mud covers the snow where it is present.

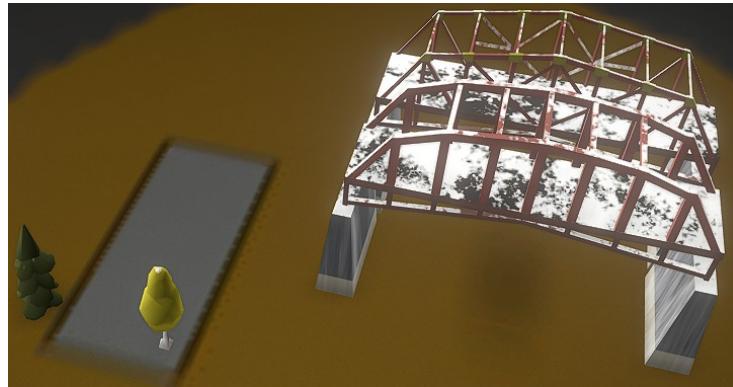
Procedural Snow on 3-D Overlays

Procedural snow cover is added to conforming overlays (page 194) and elevated overlays (page 197) in the same way as for models, above. It is also added to the periodic models created by the overlay brushes (page 184).

Procedural snow is not applied to roads or wires.

Procedural Snow on Landmarks

In a winter map, a moderate amount of procedural snow is added to all model and 3-D overlay landmarks, regardless of the [Procedural Snow Coverage](#) value or model properties.



Procedural snow is not added to plant landmarks.

Procedural Snow on Material Layers

Procedural snow coverage is added to the [stone_snowy](#) and [rocks_rus_01_snowy](#) material layers depending on the angle of the terrain, very much like what is done for models.



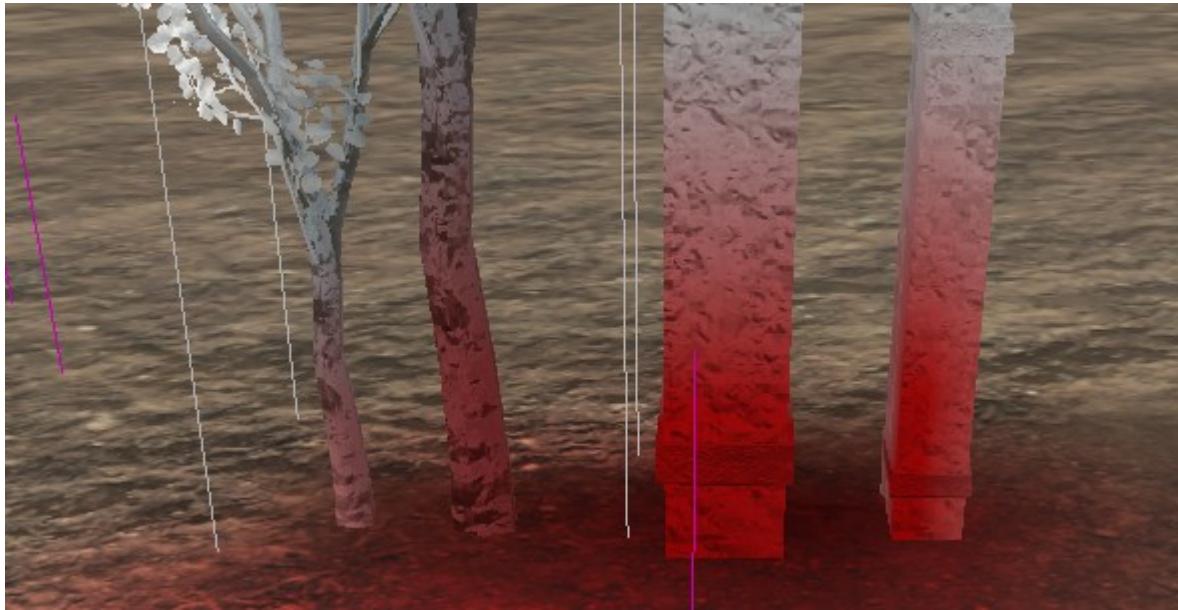
Both the game and the Editor draw procedural snow on these textures only for a winter map.

Bug: The [Show Snow By Up Vector](#) Editor setting has no effect on procedural snow drawn on material layers.

Interaction with Colorization

Plants can be colorized (page 115), but if there is enough procedural snow, it covers the colorization.

However, procedural snow on all objects is colorized within about a meter of the ground. For plants, this effectively recolorizes the bottom of the plant after the procedural snow hides the coverage elsewhere. The procedural snow colorization distance ignores the scale of a model, but is scaled for a plant.



The colorization of procedural snow applies only to the parts of the object directly over the colorized ground. This is true even for plants, which are normally colorized based on their base point's position.

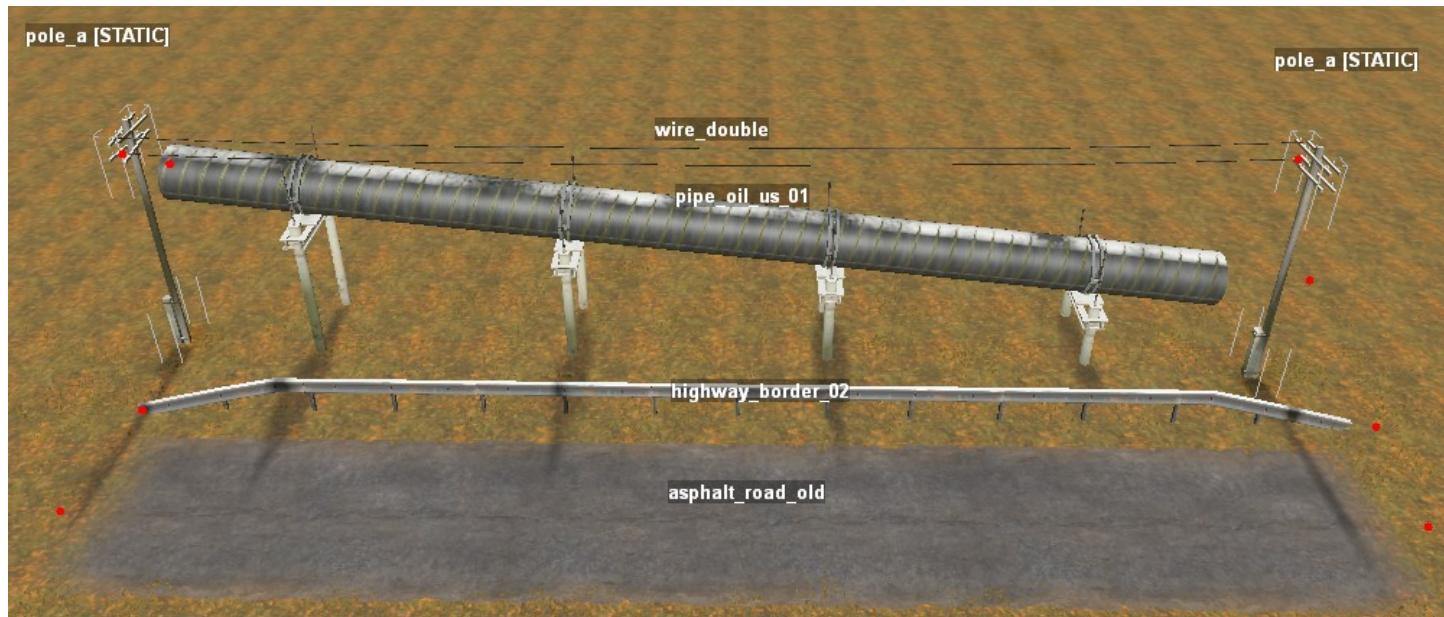
Overlays

Overlays are unique from other objects in that the elements of the overlay are repeated along a path stretched between two or more nodes. Overlays are so called because in SpinTires and MudRunner, all overlays were draped over the underlying terrain. SnowRunner adds new path-following objects that have heights independent of the underlying terrain, but they're still called overlays.

To add a road, choose **Add Overlay** from the context menu. A window appears where you can select what type of overlay you wish to create.

There are four kinds of overlays:

- Roads.
- Conforming overlays.
- Elevated overlays.
- Wires.



All overlays share the same methods for manipulating the nodes that define an overlay's path. They also (sort of) share a common set of overlay properties. These shared methods and properties are described first, with later sections devoted to the attributes of each kind of overlay and the particular properties for each kind's nodes.

Manipulating Nodes

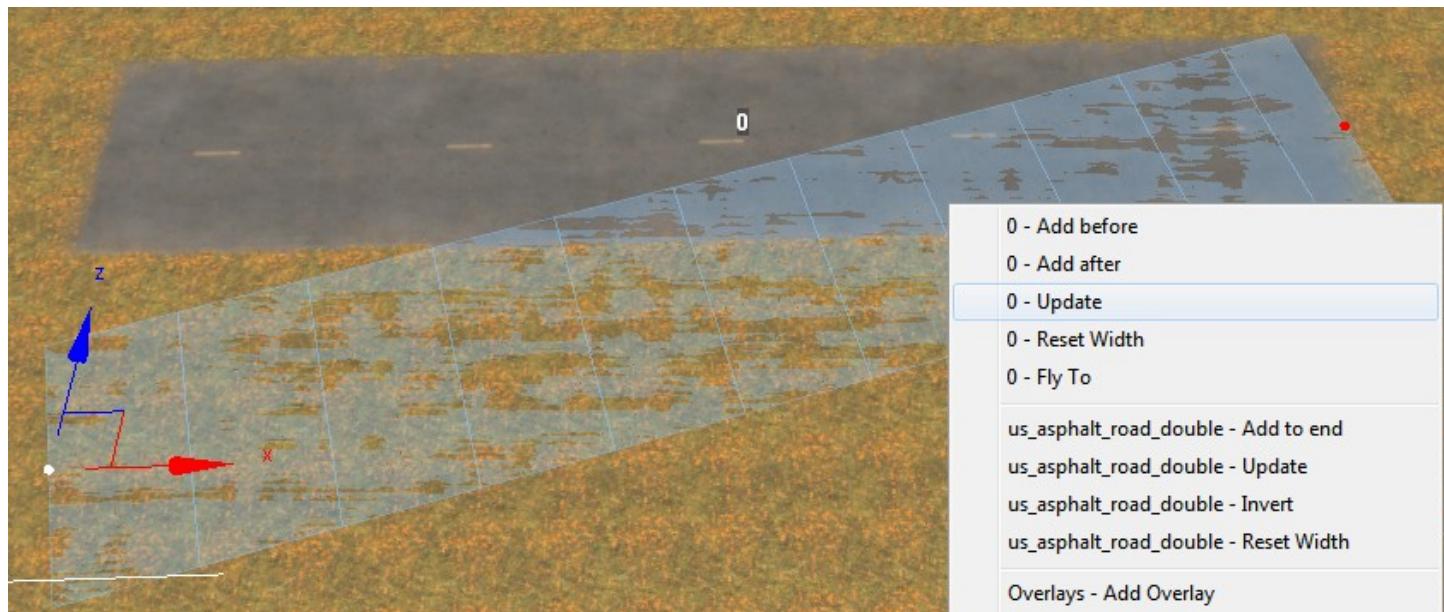
This section uses roads for illustration, but the concepts apply to all kinds of overlay.

A new overlay starts with two nodes, labeled 0 and 1. Each node can be individually selected and moved. You can add more nodes to make a more complex path. Depending on the kind of overlay, the Editor draws the overlay as a smooth curve or as straight segments between nodes.

You can click an overlay's node in the main panel to directly select that node. You can also expand the overlay in the **Scene View** and click one of its numbered nodes within. If the node is off the edge of the map, you can only select it from the **Scene View**, not in the main panel.

A node can be moved using the standard methods. If the overlay is a terrain-following kind, its nodes do not have a height property, and you can only move them in the X-Z plane. Otherwise, each node can be moved in all three dimensions.

When you move one or more nodes of an overlay, the Editor does not redraw it immediately, but instead draws the new overlay position in pale blue. To redraw the overlay at its new position, deselect the overlay, or select **Update** from the context menu.



Update actually appears on the context menu twice. This is because the context menu includes the option for both the selected node (*n* – **Update**) and the overlay as a whole (*overlay_type* – **Update**). **Update** for either the node or the overlay does the same thing.

It is also possible to move all nodes on the overlay at once by selecting the overlay itself in the **Scene View** (not just one of its nodes). You can then move it using a widget in the main panel, which updates the positions of all of its nodes by the same amount.

Bug: The Editor puts the interface for the widget at its original position prior to nodes being moved, not in the current center of the overlay. The Editor only repositions the widget to the center of the overlay when it re-evaluates the entire overlay, e.g. after rebuilding the terrain (page 65).



Each kind of overlay has a different method of terminating the overlay at each end. In most cases, the overlay needs to be a minimum length for it to be drawn at all.

Bug: The Editor occasionally forgets to draw non-road overlays. Select the overlay or rebuild the terrain to remind the Editor to draw it again.

Add a Node

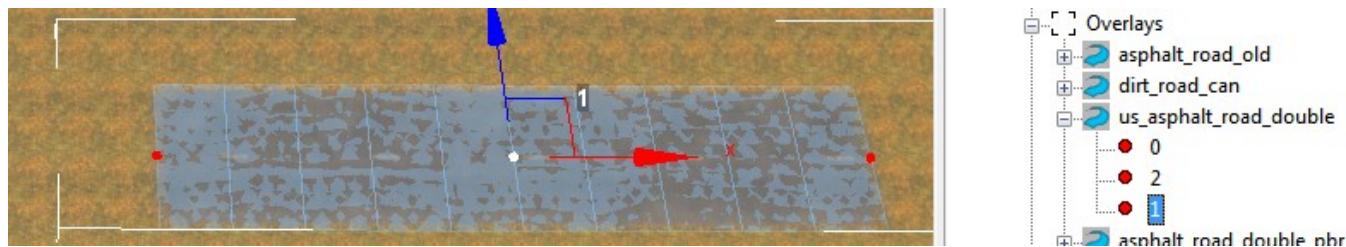
You can add a new node to an overlay using the context menu. Whenever a new node is created, it is automatically selected.

In the context of the overlay, `overlay_type – Add to end` creates a new node after the highest numbered node on the overlay so far. The new node is initialized only a meter out from the original end node, but you can move it wherever you want. There is no equivalent “add to beginning” option.

In the context of a node, `n – Add before` adds a new node midway between the selected node and the previous (lower numbered) node, and `n – Add after` does the same relative to the next (higher numbered) node. If `Add before` is used on the first node (0), it extends the overlay by half the distance between the first two nodes. Likewise, `Add after` extends the overlay from the last two nodes. Note that `n – Add after` and `overlay_type – Add to end` both extend the same end of the overlay, but by different distances.

Node Order

A new node is numbered so that it has the correct order with respect to the other nodes on the overlay. However, it is always added to the end of the list of nodes in the **Scene View**, regardless of its order on the overlay and numerical order. This can be confusing. The order of nodes in the **Scene View** is corrected if you save and reload the map or if you rebuild the terrain.



If you prefer the overlay to be numbered in the opposite direction, select **overlay_type – Invert** from the context menu.

Tip: You can **Invert** an overlay twice to quickly fix its node order in the **Scene View**.

Bug: The Editor sometimes incorrectly complains that there are fewer than two nodes. In my experience, it recovers and does the right thing once you've clicked away a warning dialog or two.

Delete a Node or an Overlay

To delete a node, select it and choose **n – Delete** from the context menu, or press the **Delete** key. After the node is deleted, the Editor selects the overlay as a whole.

If the overlay has only two nodes, there is no **n – Delete** option for a node on the context menu, and the **Delete** key does nothing when a node is selected (after a spurious confirmation dialog).

To delete an entire overlay, select the overlay in the **Scene View** and choose **overlay_type – Delete** from the context menu, or press the **Delete** key. To avoid accidents, the Editor does not include the **overlay_type – Delete** option when an individual node is selected.

Split an Overlay

To split a overlay into two overlays, select the node where you want the overlay to be split and select **n – Split in this node** from the context menu. **Split in this node** is not available for the first or last node of a overlay.

Bug: When you split an overlay, the Editor incorrectly complains that there are fewer than two nodes. In my experience, it recovers and does the right thing once you've clicked away a warning dialog or two.

Overlay Properties

The following properties apply to the overlay as a whole (not individual nodes).

Type

`Scene.(Terrain).Overlays.type`: `Type`: text

Specifies the overlay type.

You can change the type of an existing overlay by typing a new type into its `Type` property. If you mistype the type, the Editor displays a couple of warning dialogs and then reverts your change.

Manually entering a new `Type` value can be handy if you have a complex overlay shape that you think would work better as a different type. Be sure to change it to another overlay of the same kind, however, or the different node property names will cause problems for the Editor.

Flatten

`Scene.(Terrain).Overlays.type`: `Flatten`: `True/False`

Specifies whether to flatten the road from one side to the other. Has no effect on non-road overlays.

Default: `False`.

The `Flatten` property exists for all overlays, but it is only used for roads. Details are on page 189.

ApplyOffset

`Scene.(Terrain).Overlays.type`: `ApplyOffset`: `True/False`

Specifies whether to apply an offset to the terrain height based on the road overlay type. Has no effect on non-road overlays.

Default: `True`.

The `ApplyOffset` property exists for all overlays, but it is only used for roads. Details are on page 190.

Periodic Models

`Scene.(Terrain).Overlays.type`: `Brushes`: brush list

Specifies the models to place at intervals along the length of the overlay.

Default: blank; no models are placed.

Models can be automatically added along an overlay's path by clicking [press] next to **Edit** under the **Brushes** property category for the road. Brush lists are described on page 62.

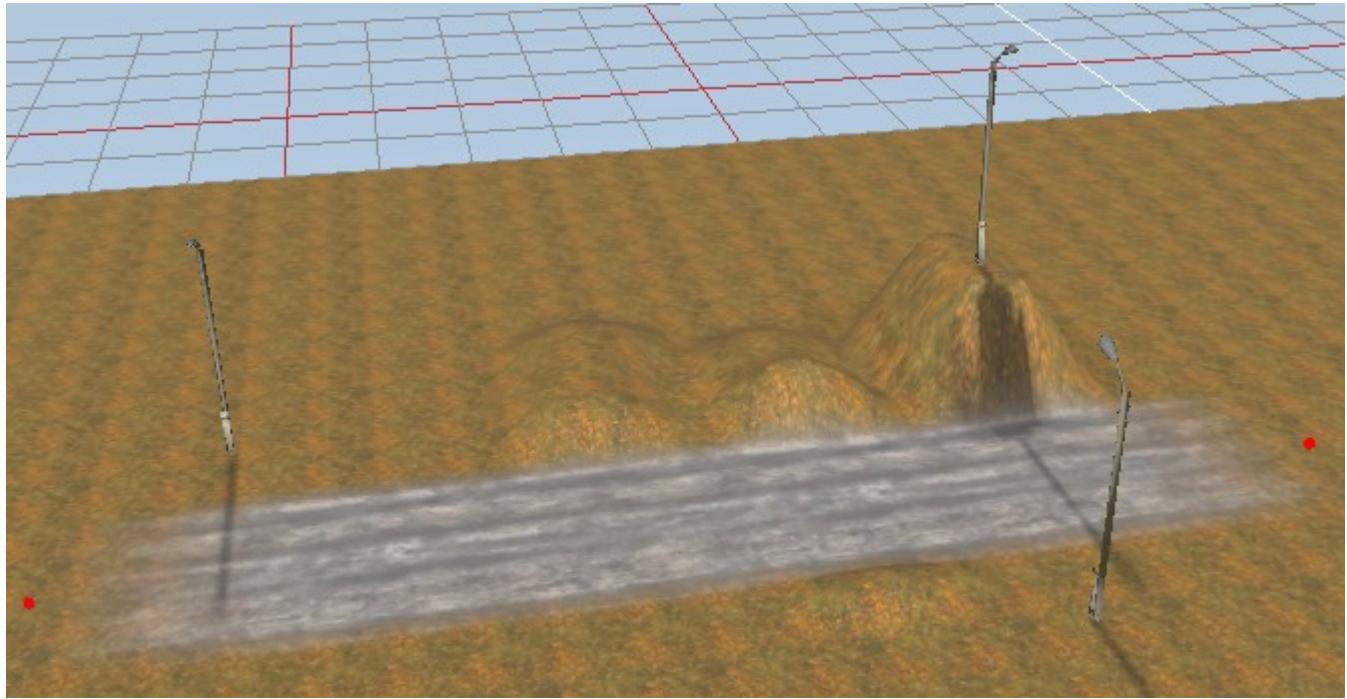
The brush list has only four entries:

- **Lampposts** are streetlights that run along both sides of the overlay. Appropriate for roads.
- **LamppostsSided** are streetlights that run along one side of the overlay. Appropriate for roads.
- **PipeBases** are poles and platforms that run under the overlay. Appropriate for the **pipe_oil_us_01** overlay.
- **ConveyorBase** are poles and platforms that run under the overlay. Appropriate for the **conveyor_belt_01** overlay.

Lampposts

When traveling along the road from node 0, **LamppostsSided** are put only on the left side of the road. **Invert** the road direction to cause the lampposts to switch sides.

The lampposts are placed at a fixed distance from the centerline of the road, regardless of the road width (page 191). They are placed to match the height of the terrain at that point.



Multiple types of lampposts can be added to one road, but doing so causes them to be drawn on top of each other.

Platforms

PipeBases and **ConveyorBase** are virtually identical except in some minor visual details. Although the two bases would seem to be interchangeable, the appearance of each base type on the navigation map (page 202) is tied to its expected overlay, so you probably shouldn't mix the two.

Each base consists of a platform that is attached to the elevated overlay and supporting poles that extend underneath. The supporting poles are fairly long, and for a moderate overlay height, the poles are embedded deeply in the terrain. However, their length is finite, so make sure that the elevated overlay is close enough to the terrain so that the poles aren't floating in air.

Roads

The primary visual characteristic of a road is the two-dimensional texture that it applies to the terrain. However, a road can also apply height changes to the terrain, and it has built-in behavior properties such as friction and softness.

Road overlays are those overlays with **road**, **path**, or **gravel_rail** in their names, with the following caveats:

- The **gravel_rail_*** overlays are designed for use with train rails but they're implemented the same as roads.
- The **iron_road*** and **log_road** overlays are conforming overlays, described on page 194.

Road Ends

The ends of the road fade out smoothly at each end, allowing you to blend in mud or other techniques to make a natural looking transition to wilderness. The length of the fade out depends on the road type.

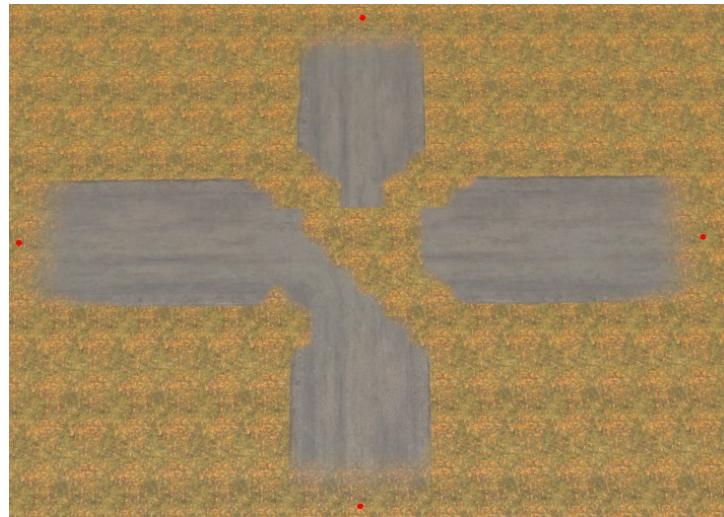
Most road types also blend smoothly with the terrain on each side of the road.

Bug: Some road types allow their side edges to get cut off along a grid boundary, resulting in a staircase edge for diagonal roads. Although some blending does occur to smooth this edge, consider adding features near the side of the road to distract from this staircase edge (e.g. mud, plants, guardrails, etc.)



Road Dropouts

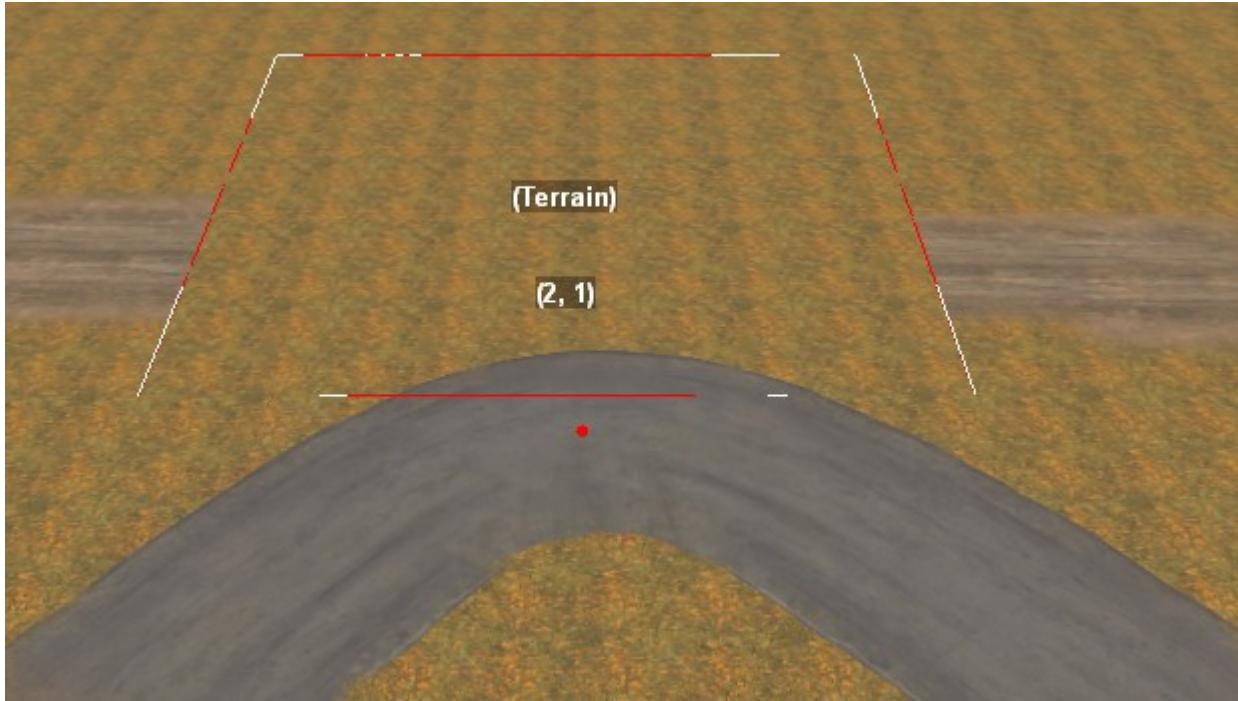
The game engine doesn't have a clean way to merge road overlays together. Instead, if any part of a road overlay is too close to another road of the same type, the road simply fades away. You can patch this missing road section using a material (such as asphalt or concrete) that roughly matches the color of the road, or you can hide it with mud or other effects. You may also need to edit plant distributions to avoid plants being placed in the missing road section.



If your road includes a tight curve, then the vertices on the inside of the curve end up too close together, which triggers the same fading effect. You can fix this by relaxing the curve or by patching it as above.

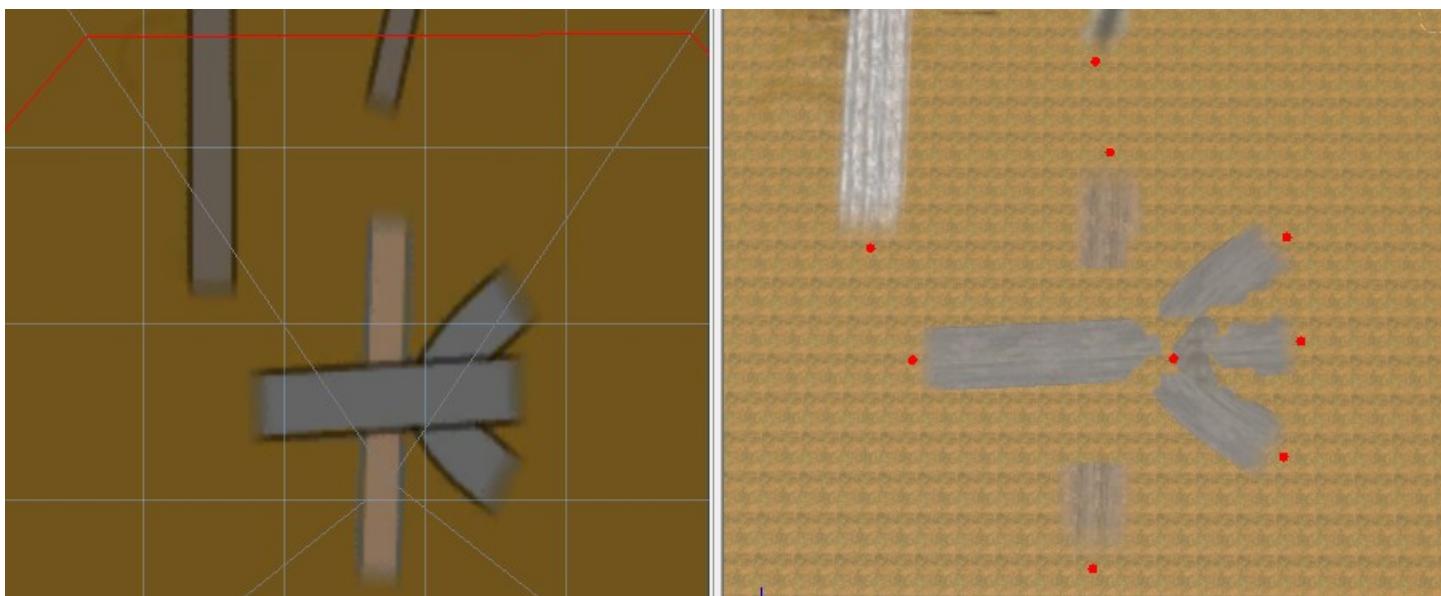


As a performance optimization, the game engine allows only one type of road per terrain block. If two types of road both enter a terrain block (even if they do not intersect), the lower priority road abruptly disappears. Priority is generally 2-lane asphalt > all other roads. If two roads are the same priority, the first road listed in the [Scene View](#) gets priority.



You will probably find yourself spending a lot of time fixing road intersections. If nothing else works (e.g. at an intersection), you're usually best off by ending a road with a standard fade out rather than having it cut off sharply at the edge of the terrain block.

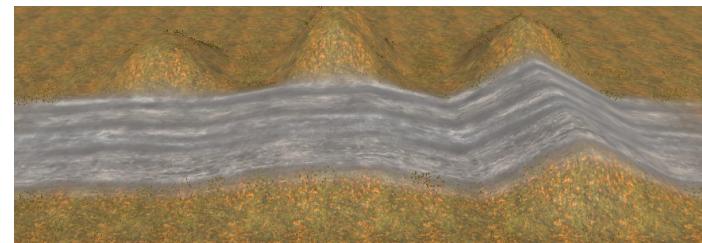
The navigation map (previewed in the [Terrain panel](#)) has no problem with drawing intersecting roads. However, it draws them with the first roads on the bottom and the last roads on the top, without any attempt to smooth the transition or merge road edges at intersections.



Flatten

The [Flatten](#) property is introduced on page 184. When [Flatten](#) is [True](#) for a road, the Editor flattens the terrain under the road. You have to manually rebuild the terrain (page 65) to apply a new [Flatten](#) value or to apply the effect after a change in road position.

[Flatten](#) removes the side-to-side “tilt” of a road, but it has no effect on the end-to-end “slope” of a road. For this reason, it’s a good idea to apply a terrain smoothing pass along the length of the road as described on page 97. Then enable [Flatten](#) for the road to remove any residual tilt.



The slope of a flattened road is based on the slope at the road’s centerline, so you can use a smoothing brush that is narrower than the road to avoid disturbing the surrounding terrain.

The flatten effect fades out at each end of the road in the same way that the road texture fades.

The width of the flatten effect is determined not just by the width of the road but also by the build-in properties of the road overlay. In particular, the `gravel_rail` overlays are flattened across only a portion of their width, and the `dirt_path_sn_01` road is not flattened at all!

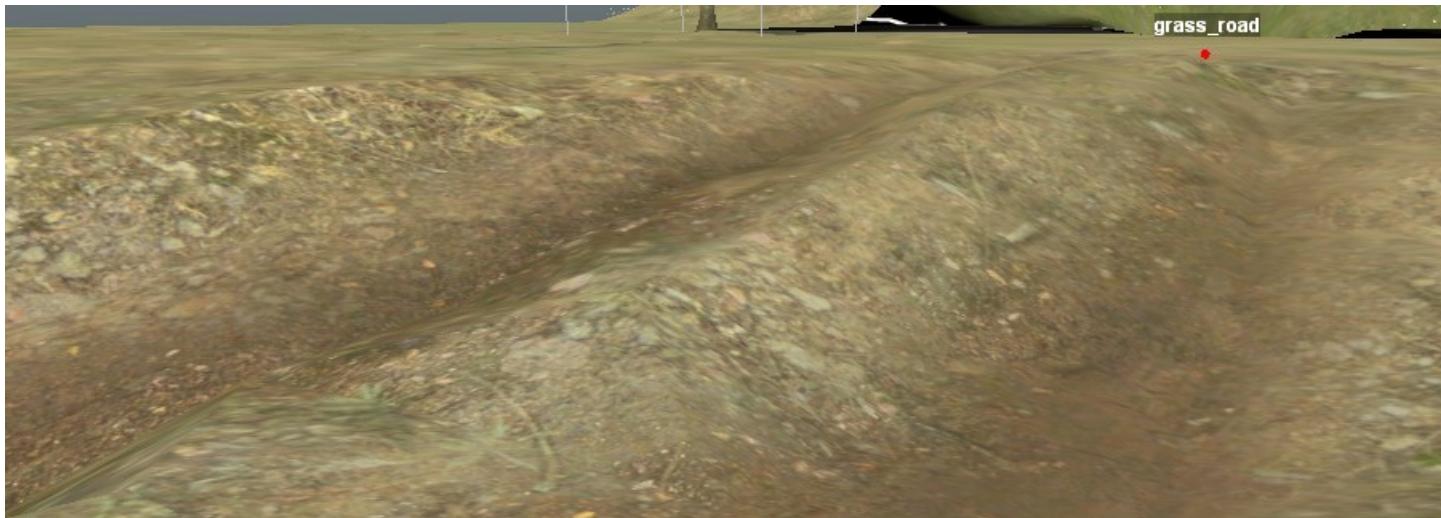
The `Flatten` effect is applied without changing the recorded terrain geometry, so you can experiment with flattening without destroying your terrain. Adjusting the terrain height below a flattened road can be very confusing, however, so it's best to set the `Flatten` property to `False` before working on the terrain geometry.

The `Flatten` property is applied to each road in its order in the `Scene View`. If roads intersect on a slope, this can result in ugliness. This is true even when a lower-priority road is suppressed from being drawn in the terrain block.

ApplyOffset

The `ApplyOffset` property is introduced on page 184. When `ApplyOffset` is `True` for a road, the Editor applies pre-defined bumps and ruts to the road based on its the road type. Changing `ApplyOffset` to `False` prevents these bumps. You have to manually rebuild the terrain (page 65) to apply a new `ApplyOffset` value or to apply the offset after a change in road position.

The screenshot below is actually from MudRunner. All of the road types in SnowRunner use a more subtle offset or no offset at all.



The offset is applied whether or not the road is actually drawn (e.g. due to `Erase Snow` being `False` or a higher priority road being present in the terrain block). Where roads intersect, the offset is applied for the last road listed in the Scene View, regardless of priority.

The offset is applied after the `Flatten` effect, of course. The road doesn't flatten itself!

Road Node Properties

The following properties apply to each node of a road overlay.

Node Position

`Scene.(Terrain).Overlays.type.n`: `Position X`, `Position Z`: numeric, in meters

Specifies the position of the node.

The position coordinate system is described on page 420.

Road Width

`Scene.(Terrain).Overlays.type.n`: `Width`: numeric, in meters

Specifies the width of the road at the selected node.

Default: determined by the road type.

Each type of road has a default width. For most asphalt roads, this width cannot be changed. For other road types, the width can be changed at each node. To change the width of the road at a node, select the node and drag up or down from the diamond around the node. Or type a new width directly into the node's `Width` property.

If the width can't be changed, the node has no diamond to drag from. If you type in a new `Width` value, the new value is ignored.

For widths less than the default, part of the road texture is dropped at the edges of the road. For widths greater than the default, the road texture is duplicated to extend the edges of the road. This may look particularly unnatural for some road types.



You can reset the width of the road to the default value in the context menu. If you select the `n – Reset width` option, only that node is reset. If you select the `overlay_type – Reset width` option, all nodes on the road are reset.

Some roads have a built-in property so that they are drawn in the navigation map (previewed in the [Terrain panel](#)) with their average width throughout, rather than showing a varied width.

Manually entering a new [Type](#) value can be confusing since it retains the width values from the old type, at least temporarily.

Erase Snow

`Scene.(Terrain).Overlays.type.n: Erase Snow: True/False`

Specifies whether snow has been plowed from the road.

Default: [True](#).

Roads are “plowed” by default, meaning that the [snow_layer](#) material and snow depth are removed from the road surface. You can change this on a per-node basis by changing [Erase Snow](#) to [False](#). This property affects the depth of the snow and the blend between the road surface and the material layer.



The depth of the snow blends smoothly from a node where [Erase Snow](#) is [True](#) (so snow depth is forced to 0) and an adjacent node where [Erase Snow](#) is [False](#) (where the snow is its full depth).

The road texture also blends smoothly with the [snow_layer](#) material layer, but over only a short distance from the node where [Erase Snow](#) is [True](#); the [snow_layer](#) dominates throughout the remaining distance to the node with [Erase Snow](#) is [False](#).

When `Erase Snow` is `False`, the material layer is actually brought to the foreground not just for the `snow_layer`, but for any material layer. This allows not just the texture to show above the road, but also any 3-D objects such as grass that are associated with the material layer.



When the material layer dominates because `Erase Snow` is `False`, it is very much as if the road has ended. E.g. the terrain is as soft as specified by the material layer, not hard like the road.

Bug: With a bit of work, the player can dig ruts through the snow to expose the dirt underneath, which shouldn't be possible if the road were actually present under the layer of snow.



There are two road features that are unaffected by `Erase Snow`:

- The road is still drawn on the navigation map.
- The height of the terrain is still affected by the `Flatten` and `ApplyOffset` properties.

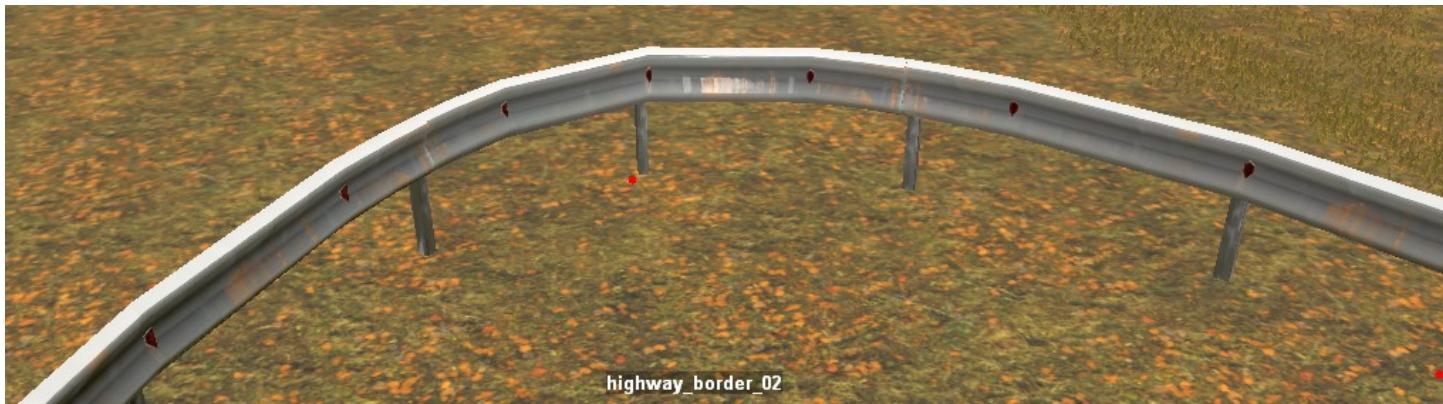
Tip: The **Erase Snow** property may be useful at intersections. Instead of allowing the lower-priority road to be cut off abruptly, and instead of ending the road early to achieve a smooth transition, you can instead use **Erase Snow** to smoothly end the visible road. But with this property, the road continues on the navigation map (instead of stopping short of the intersection), and the **Flatten** and **ApplyOffset** properties continue to apply.

Conforming Overlays

A conforming overlay consists of repeating, linked, three-dimensional objects that follow a curved path between nodes and conform to the underlying terrain. These overlay types have the following names:

- **iron_road***: a train rail.
- **log_road**: a road built of whole 3-D logs.
- **asphalt_border**: a low curb.
- **highway_border***: a guardrail.
- **cliff***: an overlay designed to build up the side of a steep section of terrain into a sharp-edged cliff.

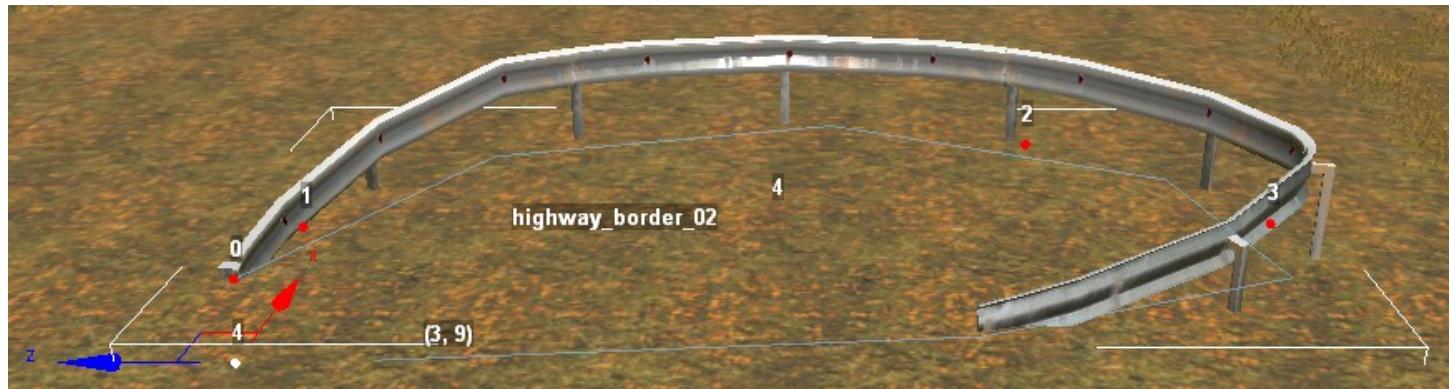
The 3-D object that is drawn for each segment of the overlay is composed of vertices and polygons, like any other 3-D object. The vertices are displaced and skewed to follow the curve of the overlay, but the polygons that form the object surfaces are drawn straight between vertices, ignoring the curve of the overlay. The 3-D objects are generally modeled with an appropriate number of vertices so that the object takes on a smooth curve or a segmented appearance as appropriate.



In addition, every vertex is displaced vertically by the terrain underneath it. For most conforming overlays, anything more than gentle terrain undulations make the objects look very weird as parts of the object are vertically displaced by different amounts. By contrast, the cliff overlays must be placed on steep terrain in order to look natural.

Many of the conforming overlays have a distinct left side vs. right side. E.g. one side of a highway guardrail is designed to face the road, and the other side faces away. If the overlay is drawn the wrong way around, use the context menu to invert the order of its points, which causes the overlay to flip the other way.

Each end of the conforming overlay is terminated by sinking the last segment of the overlay into the ground. Since each segment is a fixed length, the total length of the drawn overlay is not able to exactly match the length of the overlay path. Instead, only as many segments as can fit are drawn, starting from node 0. For the guardrail overlays in particular, the segments are quite long, which means that the final segment is not always placed as desired near the last overlay node. Usually you can fix this by adjusting the points slightly to change the total guardrail length until the final piece is placed as desired. The long segment lengths also mean that you may need to make the path longer for any segments to show up at all.



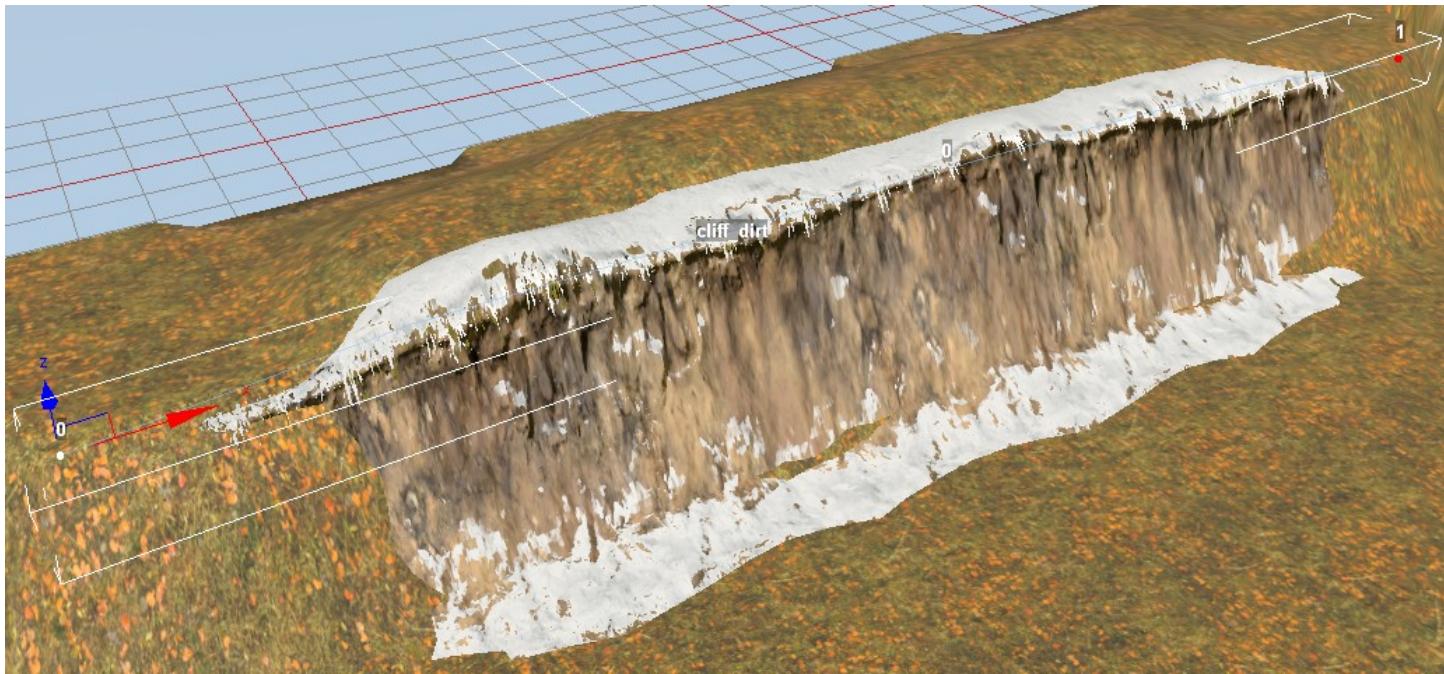
You may end up with grass or plants on top of your overlay. Road borders, cliffs, and roots are all subject to this problem, but it looks especially weird for the highway border (guardrail). Although the distribution may have **Ignore Overlays** set to **False** (page 151), that property applies only to road overlays. The only way to prevent grass and plants on top of a conforming overlay is to clear the distribution density under the overlay.



Cliffs

A cliff overlay is implemented the same as the other conforming overlays, but because it is intended to be placed on steep terrain, it takes some practice in shaping the terrain to get a cliff to look natural.

The overlay path is intended to be draped along the top of a steep section of terrain, where it creates a distinct top lip for the cliff. Most of the cliff overlay is drawn on only one side of the path, where it forms the face and foot of the cliff.



The overlay has a fixed width that depends on its type, but none is intended for a cliff more than a few meters tall. Although the overlay conforms to any amount of height change within the width of the overlay, excessive height causes the overlay's texture to stretch and look unnatural.

The cliff overlay uses a different texture for the top of the cliff than for the cliff face, and occasionally another for the foot of the cliff. You should use a matching material layer for the terrain to blend smoothly with the top and bottom of the cliff. You may need the underlying material layer to match the face of the cliff as well since the cliff overlay often allows portions of the underlying terrain to be exposed.

When facing the cliff from below, the cliff overlay should be laid out with node 0 on the left and the highest number on right. Otherwise the overlay will be upside down. Use the context menu to invert the node numbers and thus also the cliff orientation.

The cliff overlays are built with a short kink upward at the top of the cliff edge to get the overlay clear of the underlying terrain. However, you generally don't want this upward kink in your terrain. I find that it looks best to include gently rising terrain at the top of the cliff to raise the kinked top of the cliff overlay to a flatter aspect.

The only properties for a confirming geometry overlay node are for position.

`Scene.(Terrain).Overlays.type.n`: `Position X`, `Position Z`: numeric, in meters

Specifies the position of the node.

The position coordinate system is described on page 420.

Elevated Overlays

An elevated overlay consists of repeating, linked three-dimensional objects that follow a curved path between elevated nodes. These overlay types have the following names:

- `pipe_oil_us_01`: a large-diameter pipe (e.g. for oil).
- `conveyor_belt_01`: a flattened tube with an implied conveyor belt inside.

Unlike the previously described kinds of overlay, each node of an elevated overlay can be positioned freely in 3-D space.

Bug: When the elevated overlay as a whole is selected, the interface widget only allows it to be moved horizontally. The only way to lift the entire overlay is to lift each node individually.

Unlike the conforming overlays described above, an elevated overlay doesn't sink into the ground at each end. It simply cuts off at the end of a segment. For best results, place an appropriate `conveyor*` or `pipe*` model to terminate each end of an elevated overlay. There are also `pipe*` models that raise the height to allow truck traffic between two sections of elevated overlay.

The elevated overlays look best with a support structure added periodically using the `Brush` property. See page 184 for details.

The only properties for an elevated geometry overlay node are for position.

`Scene.(Terrain).Overlays.type.n`: `Position X`, `Position Y`, `Position Z`: numeric, in meters

Specifies the position of the node.

The position coordinate system is described on page 420.

Wires

Unlike the segmented curves of other overlays, wires (`wire_*`) are stretched straight from node to node. Wires are generally strung from pole to pole, but the wire overlay is just the wires. You have to add the pole models separately.

The poles that you attach the wire to can be created by one of the lamppole brushes applied to a separate road overlay. Do not try to use the lamppole brushes directly on a wire overlay; the poles will always be off to the side of the wire, and when you try to attach the wire, the poles move away the next time you rebuild the terrain.

`Scene.(Terrain).Overlays.type.n`: `Position X`, `Position Y`, `Position Z`: numeric, in meters

Specifies the position of the node.

The position coordinate system is described on page 420.

Poles have built-in attachment points to which you can snap your wire nodes. Hold down the `V` key while moving a wire node near a pole's attachment point to snap the wire node to the attachment point. The snap occurs when the node is within 1 meter of 3-D distance from the attachment point.

Tip: It can be hard to move a wire node to the correct position in 3-D space. If you orient the camera to point straight down at a pole, you can easily move a wire node to the right horizontal position. Then re-orient the camera to a side view and hold `V` while moving the wire node vertically until it snaps into place.

You can also hold down the `B` key to snap at a larger distance. The snap occurs when the node is within 3 meters of horizontal distance and **any amount** of vertical distance from the attachment point.

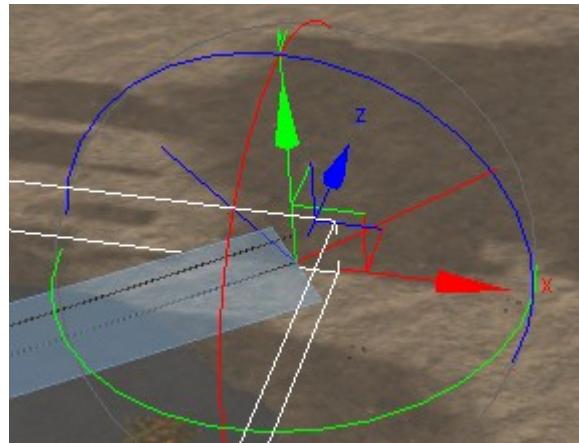
Bug: The `B` key is super flaky when snapping to the `power_lines_01` model. Sometimes it refuses to snap at close distances, but can snap when the horizontal distance is **greater** than 3 meters. Sometimes it doesn't snap at all. If it doesn't work for you either, just ignore it and use the `V` key.

Note that wires do not actually attach to a pole; they merely snap to the pole's location. If you move the pole, the wires are left behind.

Building models do not supply a natural-looking attachment point for wires, but simply running a wire into a building can look good enough.

Wires can't be bumped by a truck, no matter how low the wires are. The truck simply passes through them.

Each wire node has an orientation. When the wire node is selected, the orientation is drawn as a set of three orthogonal colored lines without arrowheads in the interface widget. Unlike the other elements of the interface widget, these lines are not always the same length on the screen. Instead, each line is 4 meters long, so its length on the screen varies depending on how close the camera is to it.

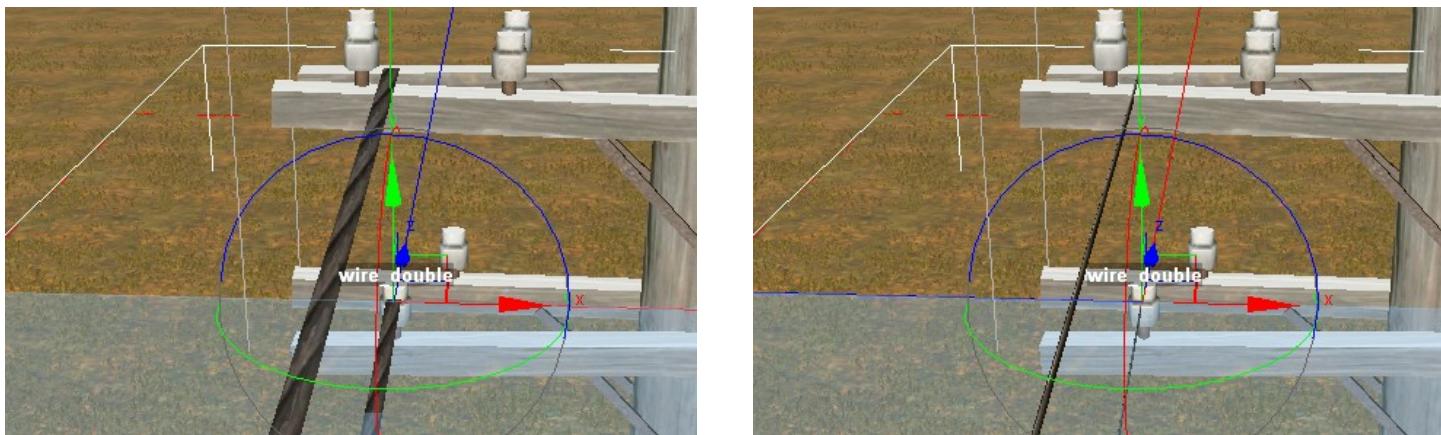


Scene.(Terrain).Overlays.type.n: Orientation: quaternion (page 426)

Specifies the orientation of the node.

Quaternions aren't really fit for humans to contemplate. Unless you're extremely mathematically minded, you should stick to the rotation widget on the main panel. The coordinate system for the orientation of wire nodes is described on page 426.

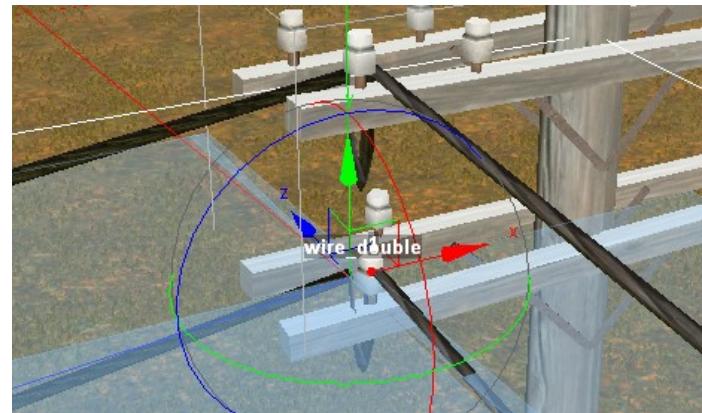
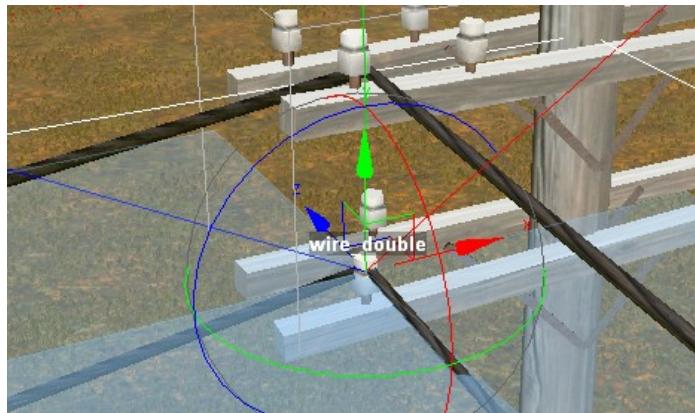
A wire overlay looks most natural when the Z axis (blue line) of each node is oriented parallel to the wire. If the Z axis is oriented perpendicular to the wire, the 3-D shape of the wire is flattened at that node.



Wires snap not only to the position but also the orientation of a model. This allows them to connect cleanly to the model and to have a natural orientation between models. Note however, that while it may seem harmless to twist a power pole 180°, if it doesn't match the orientation of the next pole in the line, the wire gets twisted between the poles, resulting in a flattened spot midway along the wire.

Where the wire bends at a node, the orientation cannot be perfect for both segments of wire on either side of the node. If it's a minor bend, the orientation discrepancy causes only a small amount of flattening that is hardly noticeable. As the orientation discrepancy exceeds 60°, however, the wire is flattened by over 50%, and it starts to become noticeable. (Admittedly, though, it's more noticeable when you get in close with the Editor than when the player is watching where she's driving in the game.)

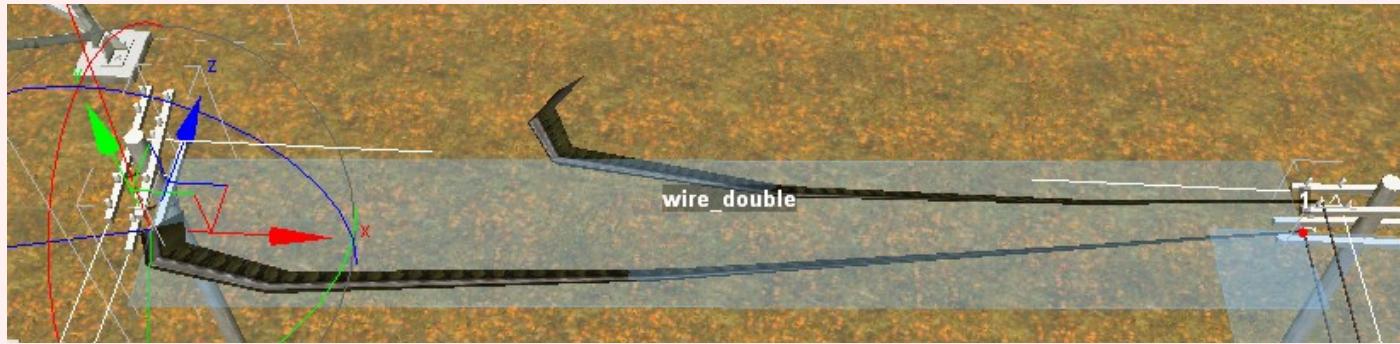
The easiest way to mitigate this issue is to orient each pole to split the difference in orientations of the wire on each side. Then snap the wire to the pole to make its orientation match. Or, after snapping the wire, you can manually adjust the node's orientation to split the difference, as shown in the lower left screenshot. Here, the node orientation is off by only 45° for each wire segment, so the wire is flattened by only ~30%.



Alternatively, you can add a new node very close to the bend so that each node on either side of the bend can be naturally oriented for the neighboring long stretch of wire, and the twist is only in the short wire between the adjacent nodes. Since every wire has a fixed droop, however, the droop in the short wire can look odd, as shown in the upper right screenshot.

Bug: The Editor crashes if you snap two adjacent wire nodes to the same attachment point or otherwise give them the same position.

Bug: If you repeatedly rotate a wire node, rounding errors will accumulate in the orientation quaternion, causing it to not be the expected unit length. Since the Editor does not correctly re-normalize the quaternion, the wire gets scaled oddly, causing it to completely miss its specified position. Hand edit the orientation property to (0; 0; 0; 1) and try again with fewer rotations. But probably the easiest method is to create a temporary model to which you can snap the wire node's position and orientation, then delete the model once the node is in place.



Tip: Wait to add wires until the rest of your map is nearly complete so that you don't have to repeatedly readjust them.

Model Attachment Points

The following models have defined attachment points for wires:

- [lamppole_a*](#)
- [pole_a](#)
- [pole_a_w_light_01](#)
- [pole_b](#)
- [power_lines_01](#)
- [station_02](#)
- [us_light_pole*](#)
- [us_power_pole*](#)

The [pole_a_objective](#) model doesn't have attachment points because it can move between different positions depending on an objective progress (page 131). Instead it shows wires in a fixed position when it is standing, and it is up to you to align adjacent poles and wires to form a visible connection.

Most models have more visible points where attachments look possible than configured attachment points. You can either ignore these extra points or make a manual adjustment from one of the attachment points.

You may have noticed in the above screenshots that the [wire_double](#) overlay has a stacked pair of wires. These are the perfect spacing for the pole models (at 1.0× scale), but the insulators aren't actually vertically aligned on

the poles. You can either accept that the player is unlikely to notice, or you can use a pair of `wire_simple` overlays for more accurate routing.

The `power_lines_01` model is a high-tension tower and is best suited for the `wire_big` overlay. The attachment points are strangely off. They're so high, it's hard for the player to notice, but after snapping each node into place, you can manually move it 0.82 meters down (when the model is at 1.0× scale) to get a better spot.

I'm not sure what the `wire_big_double` overlay is for; the `power_lines_01` model doesn't have anywhere to attach the vertically stacked second wire.

Other Aspects of Appearance

Shadows

3-D overlays have shadows similar to model shadows (page 133). However, whether or not static shadows are displayed for overlays is a built-in property of the overlay type and cannot be changed in the Editor.

Procedural Snow

A winter map adds snow to certain surfaces of a 3-D overlay. See page 173 for details.

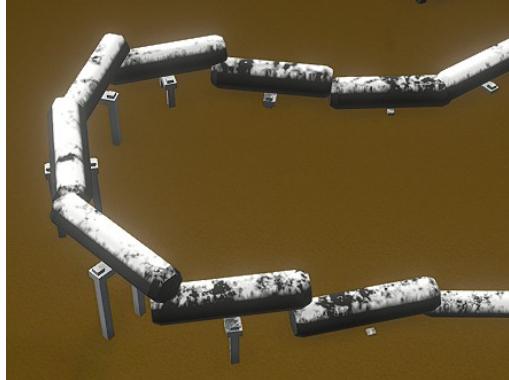
Level of Detail

As with models (page 133), 3-D overlays have multiple levels of detail.

Landmarks

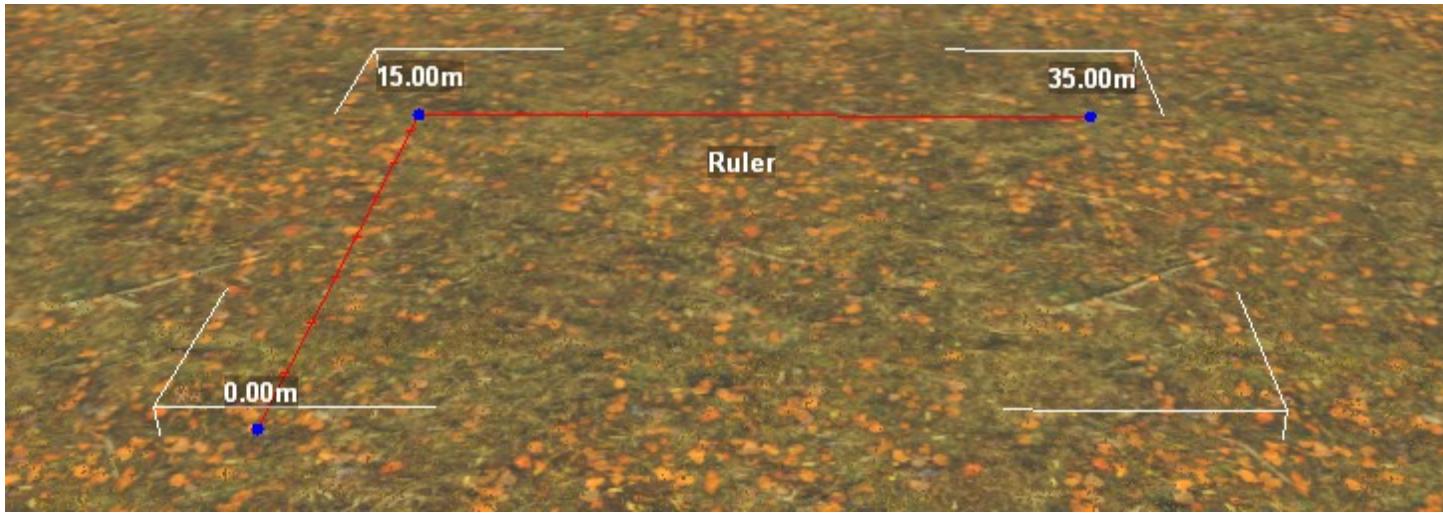
Only roads are drawn on the navigation map as a color on the terrain. 3-D overlays don't get a landmark or any other indication on the navigation map. (This is an improvement over MudRunner, where cliff overlays tended to look like paths on the navigation map.)

The **PipeBases** and **ConveyorBases** overlay brushes (page 184) instantiate models that **do** create 3-D landmarks on the navigation map. In particular, the landmark shows not only the base structure itself, but also a representative section of pipe or conveyor. However, the representative section used for the landmark does not show how the actual overlay tilts and bends, so the segments don't join cleanly on the navigation map.



Ruler

The Editor has a ruler that can be used to measure distances or lengths. The ruler is manipulated somewhat like a road overlay, but it only exists in the Editor and is not saved in any way. Only one ruler can exist at a time.



Click the **Use Ruler** icon on the toolbar to enable the ruler.

When the ruler is enabled, a left click on the terrain selects or otherwise interacts with the ruler. Selection of terrain blocks is disabled in ruler mode. Selecting any other feature in the main panel or the **Scene View** ends ruler mode. However, the ruler is saved, so re-entering ruler mode restores the previous ruler nodes.

The ruler starts out with no nodes. The easiest way to add a node is to double click on the terrain in the main panel. Each double click adds a new node to the end of the ruler's path at the location of the click. The new node is not automatically selected.

The ruler is composed of straight lines connecting however many nodes you choose to add. Each node is labeled with its total distance along the path formed by the ruler. The first node is always labeled "0.00m". The path of the ruler is also drawn with a red line connecting the nodes. The path is drawn along the ground, following the terrain. All measurements are relative to this path, so they include the 3-D distances generated by changes in terrain height.

The red line's length is always rounded to the nearest integer number of meters. I.e. if the last node has a measurement that ends in .0 – .4, the red line stops before reaching that node. If the last node has a measurement that ends in .5 – .9, the red line is extended slightly beyond the point. Strange, but true.



The properties of the ruler and its nodes are shown in the property panel, but they are **not** shown in the **Scene View**. To select a ruler node, click on it in the main panel. To select the ruler as a whole, click anywhere on the terrain in the main panel.

Manipulating Nodes

Each node of the ruler can be individually selected and moved. You can also add more nodes, and the Editor draws the ruler with straight segments between each pair of nodes.

You can click a road's node in the main panel to directly select that node.

Bug: For some reason it can be difficult or impossible to select a particular node in some cases. If this happens, you're out of luck. You can only clear the ruler (described further below) and start again.

(while a ruler node is selected): **Position X**, **Position Y**, **Position Z**: numeric, in meters

Specifies the position of the node.

A node can be moved using the standard methods. Because the ruler is always attached to the ground, you can only move nodes in the X-Z plane. Each node displays its **Y** height in the property panel, but the Editor rejects any edits to this value.

The position coordinate system is described on page 420.

It is not possible to move the ruler as a whole. Only individual nodes can be moved.

Each node has a **Width** property and a diamond widget to allow the width to be scaled, but it does nothing.

(while a ruler node is selected): **Width**: numeric

No known purpose.

Add a Node

The context menu allows you to add a new node to the road. Whenever a new node is created, it is automatically selected.

Ruler – Add to end creates a new node after the last one on the ruler's path. The first node is added to the terrain directly below the camera (not where the camera is pointed). The second node is added 1 meter east and 1 meter north of the first node. A third or later node is placed one meter out in the direction of the last segment of the ruler's path. It is usually easier to double click the terrain to add a new node to the end exactly where desired.

With any node selected, **Add after** adds a new node midway between the first two nodes on the path. **Add before** adds a new node to the beginning of the path, extending the ruler in that direction.

Bug: **Add before** and **Add after** always behave as if the first node was selected, regardless of which node was actually selected.

Ruler nodes have an implicit order, but unlike overlays, ruler nodes are never explicitly numbered.

Delete a Node or the Ruler

To delete a node, select it and choose **Delete** from the context menu, or press the **Delete** key. Every node can be deleted, leaving the ruler with no nodes. After a node is deleted, the Editor selects the ruler as a whole.

To delete all nodes at once, select **Ruler - Clear** from the context menu. There is no keyboard shortcut to clear all nodes. Note also that the ruler itself is never actually deleted. It continues to exist, but with no nodes.

More Ruler Stuff

You can invert the order of nodes in the ruler (and thus the measurement origin) by selecting **Ruler – Invert** from the context menu.

The ruler and its nodes have an **Update** option in the context menu, but it does nothing.

(while the ruler as a whole is selected): **Step**: integer, in meters

Specifies the distance between hash marks on the ruler path.

Default: 2.

Bug: Setting **Step** to 0 crashes the Editor.

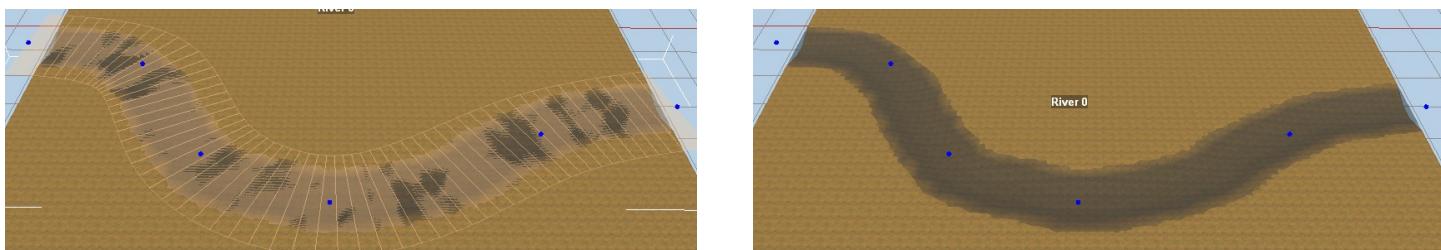
Rivers

A river is something like a combination of a road overlay with an elevated overlay. It has a 3-D path, and each node of the path specifies a width. A river can have various speeds of flow, including stopped. Depending on how the river is constrained by the terrain, it can appear to be a river, a creek, a rivulet, a puddle, a pond, a lake, or a sea, but they're all implemented as rivers.

Create a river by selecting **Add River** from the context menu. The river is created stretching from left to right across the center of the main panel, and its height is the maximum height of the terrain between the initial two nodes.

Rivers are easiest to create in a channel or depression carved out of flat or nearly flat terrain. The game engine draws steep rivers very poorly. Vertical waterfalls are impossible, although steep cascades can be created under controlled conditions.

In general, you want the river feature to be wider than the corresponding depression in the terrain, with a height somewhere between the top and the bottom of the depression. Adjust the height and width at each point accordingly. When the river is selected, its full width can be seen. But when the river is deselected, the river is shown only where the river is higher than the terrain. This hides the edges of the river feature where they extend beyond the depression and into higher terrain.



The display engine handles sharp corners well, although there's not much it can do to make a fast-flowing river look natural if the water bends back on itself.

You can make the river arbitrarily wide, but if the terrain is lower elsewhere, the river may start to intrude where you don't want it. Also be aware that if river water is close under the ground surface, a truck will be able to dig ruts down to water level, or it may even make splashing noises when the river is barely hidden under the terrain.

River Node Properties

The following properties apply to each node of a river.

Scene.(Terrain).Rivers.River *n.m*: **Position X**, **Position Y**, **Position Z**: numeric, in meters

Specifies the position of the node.

The position coordinate system is described on page 420.

Scene.(Terrain).Rivers.River *n.m*: **Width:** numeric, in meters

Specifies the width of the river at the selected node.

Default: 16.

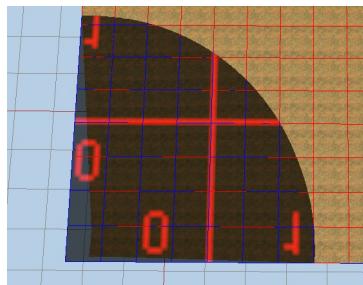
River Speed

The speed of flow can be painted for any location within a river. To begin painting, select **Scene.(Terrain).(Geometry).(Water)** in the **Scene View**, then select **Speed** in the mode selection dropdown of the **Brush** dialog box.

To increase the speed of flow, paint with a **Value** greater than 0.5. To decrease the speed of flow, paint with a **Value** less than 0.5. While the **(Water)** tool is selected, the **Speed** paint is drawn as an artificial blue tint applied to the visible areas of the river.

Caution: the paint is applied at the position of the **terrain** under the mouse, even though the paint is drawn at the height of the river. If the river is deep (so the terrain is a very different height), this can result in a large discrepancy in brush position depending on your camera angle in the Editor. Pay attention to the position of the brush preview disc to see where paint will be applied.

You'll notice that the **(Water)** tool shows a red grid over your river with upside down numbers at the edges. This has no real meaning, but if you're curious, it's explained on page 437.



When the **(Water)** tool is deselected, the Editor shows water speed in the same way as the game does: as moving waves on the water surface. This makes it easy to evaluate the results of different amounts of **Speed** paint.

By default, the water flows along the curved path laid down by the river's nodes, in the direction of increasing node numbers. Remember that you can use the context menu to invert the direction of node numbering. Page 212 describes how to get more precise control of the flow direction.

In addition to its visual effect, the speed of the river also influences how strongly it pushes against the player's truck.

Adjust Speed Based on Depth

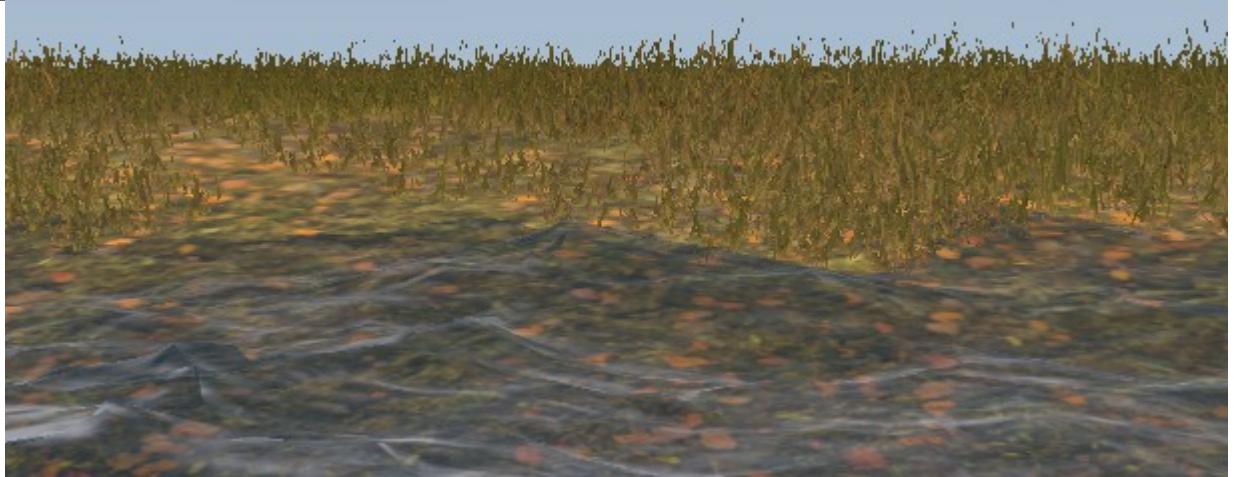
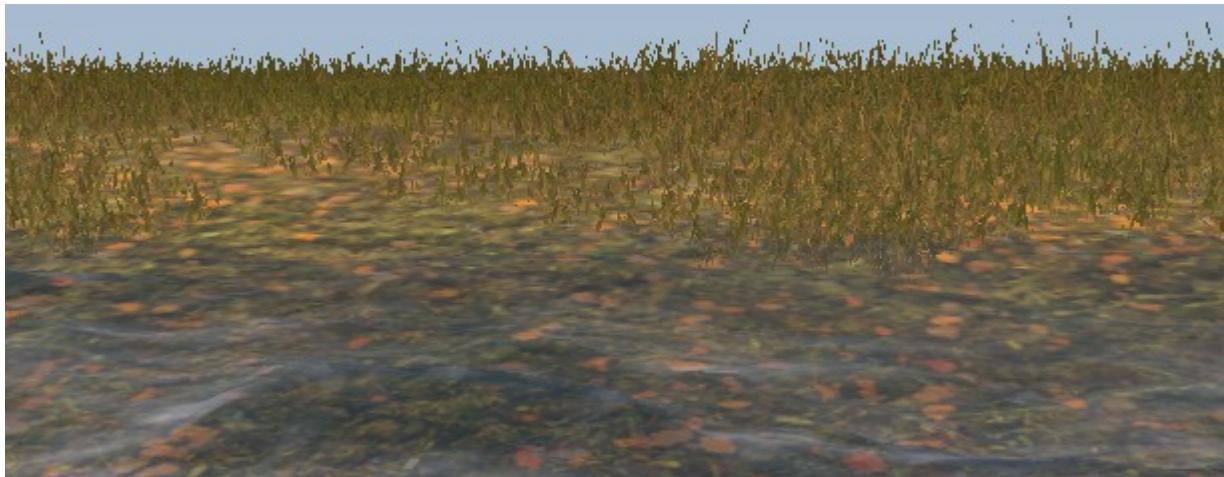
By default, the waves on the river surface are effected not just by the **Speed** paint but also by the depth of river. Specifically, areas of the river shallower than ~50 cm have decreased wave action.

Scene.(Terrain).Rivers.River *n*: AvoidEffectOpacity: True/False

Specifies whether the river depth is ignored when calculating wave height.

Default: **False**.

When **AvoidEffectOpacity** is **True**, waves can lap strongly onto shore.



Flow Property

The river has a Flow property. As far as I can tell, this does nothing.

`Scene.(Terrain).Rivers.River n: Flow: numeric`

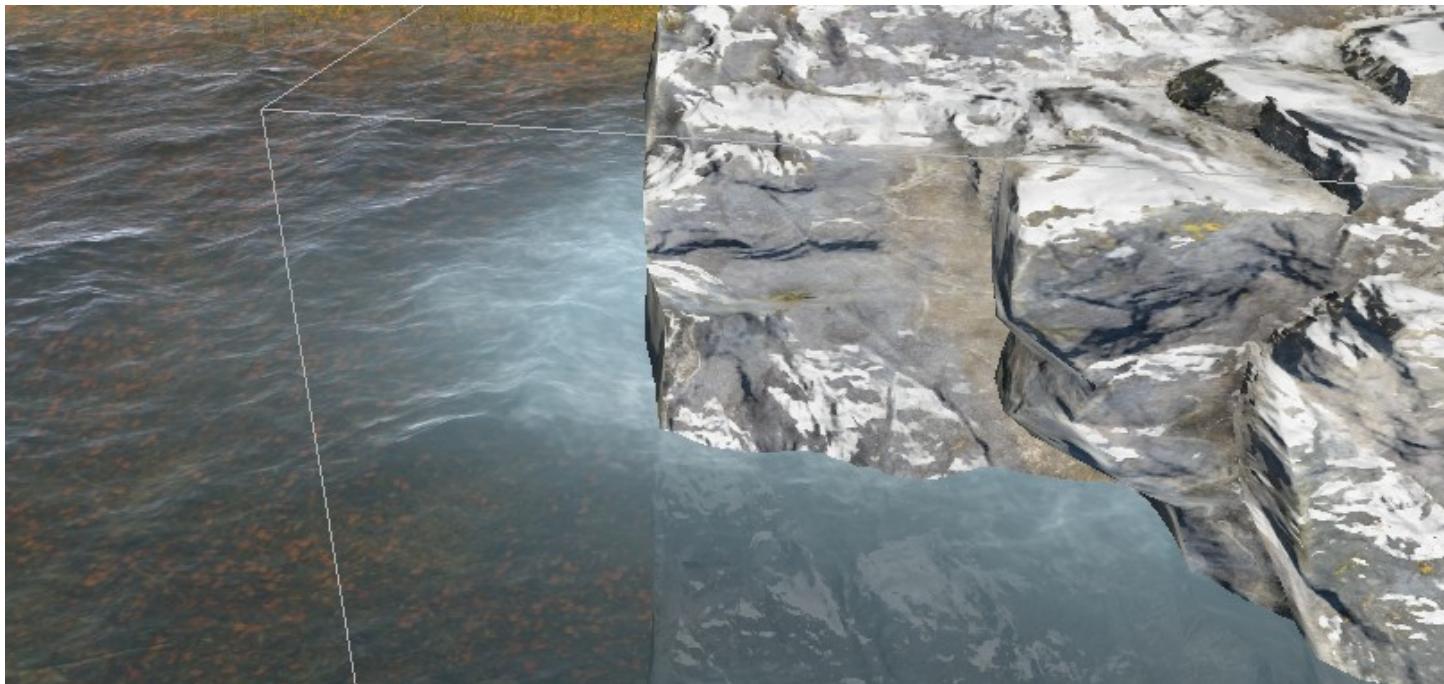
No known purpose.

Default: 0.0.

Foam

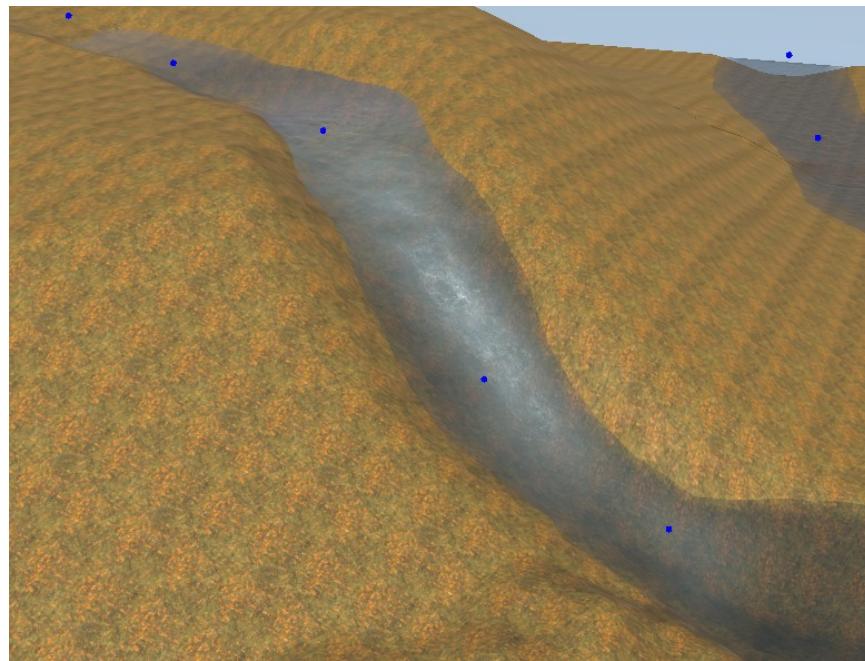
Foam can also be painted onto a river. To begin painting, select `Scene.(Terrain).(Geometry).(Water)` in the `Scene View`, then select `Foam` in the mode selection dropdown of the `Brush` dialog box.

Foam adds white air bubbles to the river surface. It looks best in high speed areas, and it looks most natural where fast-flowing water collides with an obstacle.



Cascades

Creating steep watercourses is difficult for a number reasons. It can be tough to cut the right depth of channel into a slope of the terrain. It can be difficult to bend the vertical curves in the river just right to pour over the edge. But most of all, it is simply awkward to properly maneuver the river points in 3-D space while looking at a 2-D window. In particular, more points tend to be needed to keep the river in its channel, and it's easy to have those points slightly out of line with each other which can result in the river making odd shapes as it curves through the closely spaced points. Keeping all of the river points lined up is the best way to keep your river in the proper shape.



Note that where the river transitions between a steep area and a flat area, the river may hump or dip as the display engine tries to smooth the transition. Try to control these humps and dips to avoid the river flowing too obviously uphill through them (or even overspilling the banks).

River Junctions

Overlapping rivers combine quite well as long as they have a similar height and slope at the junction. If they are both using the default flow direction (along the river path), their flow directions are smoothly averaged across the junction.

One area to watch out for is excess river width. This usually doesn't matter when it's hidden underground, but it can affect the river junction, especially if the other river is entering at a steep angle.

The Editor will occasionally create small gaps in a river at a junction or elsewhere as you perform local edits. Rebuild the terrain (page 65) to reconnect all the river pieces.

Flow Map

Rather than use the default direction set by the path of river nodes, you can paint the desired flow direction onto the map.

Scene.(Terrain).Rivers.River *n*: UseFlowMap: True/False

Specifies whether to use the painted flow map

Default: **False**.

Interestingly, changing **UseFlowMap** to **True** does not immediately change the water flow direction because where the flow map is not yet painted, the river still defaults to deriving its flow direction from the path of river nodes.

To paint the flow direction, select **Scene.(Terrain).(Geometry).(Water)** in the **Scene View**, then select **Flow** in the mode selection dropdown of the **Brush** dialog box.

Caution: the paint is applied at the position of the **terrain** under the mouse, even though the paint is drawn at the height of the river. If the river is deep (so the terrain is a very different height), this can result in a large discrepancy in brush position depending on your camera angle in the Editor. Pay attention to the position of the brush preview disc to see where paint will be applied.

When painting with **Flow**, a **Value** of 0.0 has no effect. Higher values of **Value** lay down stronger paint. The direction of water flow is determined by the direction that the mouse is moved while painting. Different colors are used for different directions, although the mapping of color to direction is mostly inscrutable. (See page 443 for a detailed description.)

Bug: There is no way within the Editor to erase **Flow** paint to restore the default flow direction. You can only overpaint the **Flow** with a different direction.

Bug: In **Flow** mode, the brush preview disc treats 0.5 as the neutral **Value** when 0.0 is the actual neutral **Value**. This means that the preview disc is invisible near a **Value** of 0.5 even though this **Value** causes a large effect.

Randomize has no effect in **Flow** mode.

Bug: **Autofade** paints the starting tail in the wrong direction.

The context menu on any **(Geometry)** feature includes a **Create Default Flowmap** option that can be useful when editing the flow directions outside the Editor. See page 444 for more information.

Water Color

Each river can be painted with a mix two water colors.

Scene.(Terrain).Rivers.River *n*: Clean Type: dropdown menu

Specifies the color for the river water where it is not painted as muddy.

Default: **blue**.

Scene.(Terrain).Rivers.River *n*: Muddy Type: dropdown menu

Specifies the color for the river water where it is painted as muddy.

Default: **brown**.

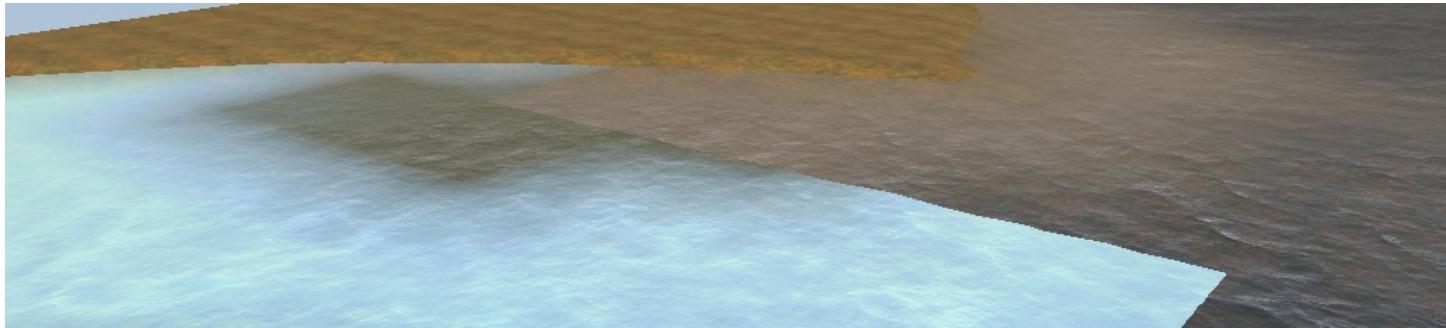
To paint parts of a river as “muddy”, select **Scene.(Terrain).(Geometry).(WaterMud)** in the **Scene View**. There is only one mode for this brush, **Mud**.

To increase the muddiness of the water, paint with a **Value** greater than 0.5. To decrease the muddiness, paint with a **Value** less than 0.5. Muddy water is tinted green while the **(WaterMud)** brush is active.



Caution: the **Mud** paint has very low resolution, only one pixel every 8 meters. It cannot be used for detail work.

Each terrain block allows only one color for **Clean Type** and one color for **Muddy Type**. If two rivers are within the same terrain block with different colors, the colors of one river are used throughout the entire terrain block.



Tip: If two rivers have a junction, make sure that they have at least one color in common, and use that color throughout the terrain block(s) that they share. If they share both colors, then you can mix those colors freely in the junction.

Tip: You can split a river and give each segment different colors. Even if they share a color in the middle, this allows three colors total across both segments. E.g. you can use one color for the wild upper reaches, another color for normal flow, and a third color for stagnant or deep lower sections.

Background Water

You can add water that extends from one or more of your map's edges to the horizon. Background water is flat, clean, and has no speed or foam.

Scene.(Terrain): Background Water.Type (Clean): text

Specifies the color of the background water.

Default: blank; prevents display of background water until set to a valid value.

Unlike the colors for river water, the background water color is not selected from a pulldown menu. Manually type a value to match one of the pulldown choices for a river.

Scene.(Terrain): Background Water.Height: numeric, in meters

Specifies the height of the background water.

Default: 0.0.

To ensure that a river's height matches the background water's height at the map edge, the display engine starts averaging the river's height with the background water's height 24 meters to the edge. It uses a weighted average, with full weight to the river's height at 24 meters from the edge and smoothly adjusting the weight until the background water's height has full weight at the edge. Although the weight changes smoothly across

the 24 meters, the sudden addition of weights starting at 24 meters from the edge can result in a noticeable vertical bend in the river. To avoid this, ensure that any river within 24 meters of the map edge is already at (or nearly at) the background water height.

Scene.(Terrain): Background Water.Enable for -X: True/False

Scene.(Terrain): Background Water.Enable for +X: True/False

Scene.(Terrain): Background Water.Enable for -Z: True/False

Scene.(Terrain): Background Water.Enable for +Z: True/False

Specifies whether to display background water on the corresponding edge of the map (west, east, south, and north, respectively).

Default: **False**.

Background water is also displayed off the corners of the map if it is displayed off either adjacent edge.

Tip: If you don't want the player to see the "no entry" border, use deep water, obstacles, and/or terrain to keep trucks at least 64 meters from the edge of the map. See page 71 for details.

Bug: The background water uses a standard reflection image with trees even when there is nothing on the horizon.



River Sounds

Unless specifically configured, rivers don't make noise. In order for a river to make noise (e.g. burbling water), a river markup must be created (described here) **and** an ambient sound preset must be assigned (page 252).

Create a river markup by selecting **Add River Markup** from the context menu.

When a river markup is selected, it is displayed as a dark blue polygon enclosing the path formed by its nodes. When a river markup is unselected, it is displayed only if the **Show all sound domains** toolbar button is enabled (page 256).

River Markup Node Properties

The following properties apply to each node of a river markup.

Scene.(Terrain).Rivers.RiverMarkup: **Position.X**, **Position.Y**, **Position.Z**: numeric, in meters

Specifies the position of the node.

A node can be moved using the standard methods. The river markup is a 2-D feature, so you can only move nodes in the X-Z plane. Each node displays its **Y** height in the property panel, but the Editor rejects any edits to this value.

Scene.(Terrain).Rivers.RiverMarkup: **Width**: numeric slider from 0.0 to 500.0

Specifies the width of the river markup at the selected node.

Default: 100.

Bug: The widget for a river markup node does not include an interface for adjusting the width. You can only adjust the width by using the slider for its **Width** property. The slider can only select a multiple of 5.0. If you want finer control than that, you'll need to edit the XML by hand (page 427).

River Markup Behavior

When the player's camera is within the polygon designated by the river markup, the river sounds are played at their default volume surrounding the player. As the player's camera gets further away, the river sounds decrease in volume and are spatially located toward the river markup.

The river sounds are defined by the ambient sound preset (page 252), which must be assigned in order to hear anything.

Wetness

Expand the **(Geometry)** category in the **Scene View** and click **(Wetness)** to start using the wetness brush. The only brush mode is **Tint**.

By default, the terrain is dry. Painting it with increased wetness makes the terrain more wet.

Visually, wetness makes terrain somewhat darker and somewhat glossier. The degree of change is determined by the **Albedo wetness mult** and **Roughness wetness mult** properties of the material, described on page 155.

Wetness makes all surfaces more slippery, and it makes soft surfaces softer. These two effects dramatically increase the difficulty of traversing muddy terrain.

Wetness has only a small visual effect on the **snow_layer** texture, and it makes little or no difference to vehicle behavior.

Wetness is treated as ice on the **ground_snowy**, **us_snow_asphalt**, **ice_01**, and **ice_02** material layers and on the **asphalt_road_snowy_small**, **us_asphalt_road_snowy**, **dirt_road_snowy**, and **dirt_path_sn_01** road overlays. I suspect that this makes the wetness even more slippery than usual.

Rivers (page 207) are not automatically wet. Consider making the river bottom wet as well as some distance up the banks.

Wetness can be automatically added as mud is painted. See page 108 for details.

Breakable Ice

A combination of features can transform mud into breakable ice.



The specific combination of features is as follows:

- **ice_02** material layer to supply an appropriate surface texture with low friction.
- **Extrudes** mud to allow a truck to sink.
- Wetness (manual or using **Extrudes To Wetness**) to make the mud softer.
- **IceChunks** distribution.
- A river just under the surface for splashing and additional resistance after a truck has broken through.

The **IceChunks** distribution is the key. It creates ice chunks just under the terrain surface that can support a light truck and can partially support a heavier truck. It makes appropriate crackling noises when driven over, and it turns up ice chunks when a truck breaks through.

The **IceChunks** distribution should be as dense as possible (or nearly so) to prevent a truck from finding a hole a distribution and sinking quietly into the ice without the necessary drama.

The mud depth and amount of wetness should both be fairly high. Without wetness, the mud isn't soft enough to allow a truck to break through. You can tune the mud depth and wetness for how stuck you want a truck to get.

ice_02 includes a subsurface layer so that it still looks icy underneath, rather than dirty. If you prefer the breakable ice to be over a muddy bog rather than clear water, you can substitute a different material layer (such as **snow_layer**) and use a distribution of **IceDirt** instead of **IceChunks**.

Note that the **[IceChunksSnow](#)** distribution puts its ice chunks on top of the terrain rather than underneath, and these ice chunks support much more weight than **[IceChunks](#)** or **[IceDirt](#)**.

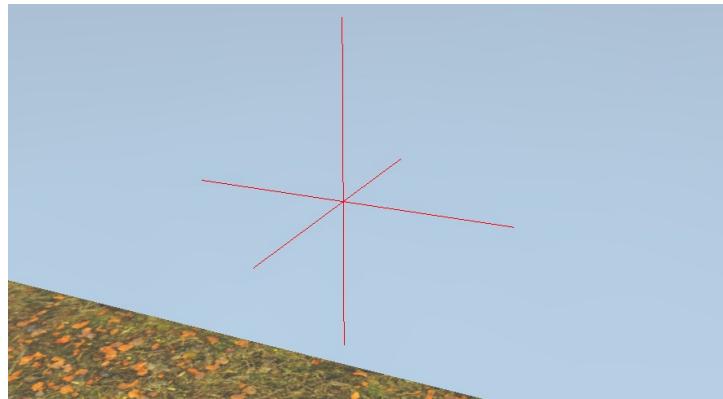
Point Sounds

The game can periodically play a sound when the player drives close to a particular location. To create such a sound, choose **Add Sound** from the context menu.

All point sounds and sound domains (page 225 and 231) play independently, and the game mixes their sounds together as appropriate.

Point Sound Visualization

When a point sound is unselected, it displays only as three intersecting red lines with no label. Show all sound domains (page 256) has no effect on the display of point sounds.



Click a point sound in the **Scene View** to select it. While a point sound is selected, the main panel displays its name and a movement interface widget for it.

Name

Scene.(Terrain).Sounds.*name*: **Name:** text

Specifies the name of the sound to display in the **Scene View** at all times and in the main panel when the sound is selected.

Default: blank; displays as **(Terrain Sound)**.

To help keep your sounds straight, you can assign a name to each sound in its **Name** property. The name has no game purpose, however. Names do not need to be unique.

Sound Location

`Scene.(Terrain).Sounds.name`: `Position.X`, `Position.Y`, `Position.Z`: numeric, in meters

Specifies the position of the source of the sound.

Default: ground level in the center of the main panel.

The sound doesn't have a scale or orientation, only a position. You can move it similar to a truck (page 75). Although the sound is initially created at ground level, there is no advantage to attaching it to the ground, so there is no equivalent to `Land` or `Do Land` for sounds.

The position coordinate system is described on page 420.

Keep in mind that the “microphone” for the sound is in the game’s camera, not in the truck. In 3rd-person view (outside the truck), you may need to move drive slightly away from the center of the sound in order to center the camera in its vicinity. Also note that in 1st-person view (inside the cockpit), all sounds are attenuated.

Sound Volume and Radius

`Scene.(Terrain).Sounds.name`: `Volume`: numeric in the range 0.0 to 1.0

Specifies the maximum volume of the sound as a fraction of its volume in the sound file.

Default: 1.0.

`Scene.(Terrain).Sounds.name`: `Distance.Min`: numeric, in meters

Specifies the distance from the point sound’s 3-D position at which the sound volume begins to fade to zero.

Default: 5.

`Scene.(Terrain).Sounds.name`: `Distance.Max`: numeric, in meters

Specifies the distance from the point sound’s 3-D position at which the sound volume becomes inaudible.

Default: 35.

The sound plays at the specified volume anywhere within the spherical radius given by `Distance.Min`. Outside this radius, the sound attenuates to zero at the radius given by `Distance.Max`. There is no visual indication of these distances in the main panel. Be sure to include the height of the sound in your calculations.

Bug: The Editor crashes when you pack your map if `Distance.Min` \geq `Distance.Max`.

Sound Files

`Scene.(Terrain).Sounds.name`: Sound file: text

Specifies the relative path and base filename of the sound file(s) to play.

Default: blank; triggers a **File not found** error when the map is packed.

You can use either the game's built-in sounds or your own custom sounds, and you can select a single file or a set of files to be played in a random order.

Built-in Sound Files

SnowRunner keeps its sound files in its `shared_sounds.pak` archive, described on page 426. If you've extracted these files somewhere, you can browse them to find a sound that you like. You'll need a sound player that can handle PCM format, such as VLC or Audacity.

Saber recommends that point sounds be in mono format (not stereo) to allow the aural direction of the sound to be correctly placed. If you are using a built-in sound file, make sure it's in the right format. In VLC, `Ctrl+J` shows the media format.

The best built-in point sounds are probably in the `[sound]/actors` directory. All of these are mono except in the `actor_construction`, `actor_rocket`, and `actor_train` subdirectories, which have sounds intended for multi-stage model animation (page 131).

Single File

Set the `Sound file` value to the sound file's path within the `shared_sounds.pak` archive, excluding the initial `[sound]` directory name and the trailing file extension. For example, if the sound is in `[sound]/actors/actor_ru_radio_rnd_set/actor_ru_radio_rnd_1.pcm`, then set the `Sound file` property to `actors/actor_rocket/actor_ru_radio_rnd_set/actor_ru_radio_rnd_1`. Either `/` or `\` can be used as directory separators.

File Set

The `shared_sounds.pak` archive includes many subdirectories with sets of files ending in two underscores and one or more digits, e.g. `_1` through `_8`. If you specify one of these files without that `_n` ending, then the game plays all of the files in a random order. E.g. `actors/actor_ru_radio_rnd_set/actor_ru_radio_rnd`.

The game never repeats a sound consecutively when playing a set in a random order. That means that a set with only two sound files always alternates between them.

Custom Sound Files

If you create your own sound file, it must be in 16-bit WAV format and have a `.wav` filename extension. The Editor converts it to PCM format when you pack your map.

(Technically, WAV files also internally use a PCM encoding, but these are not the same encoding as the built-in sounds. In fact, if you copy one of the built-in files to your own directory, the Editor will try to convert it and fail.)

Saber recommends an encoding frequency of 44.1 kHz, but any frequency that sounds good is fine. Saber also recommends the mono format for point sounds.

Put the `*.wav` file(s) in `%USERPROFILE%/Documents/My Games/SnowRunner/Media/sounds` or one of its subdirectories. You may need to create the `sounds` directory. The `*.wav` files are shared by all of your maps, so they don't go in `Media/prebuild`.

Tip: If your sound files are used by only one or a few maps, give them their own subdirectory to keep them separate from the sound files of other maps.

Some mistakes are better than others:

- If your `Sound file` refers to a file that the Editor can't find in your `sounds` directory (or in `shared_sounds.pak`), it pops up a warning dialog when you pack your map.
- If your `Sound file` refers to a `*.wav` file in your `sounds` directory that is not in the proper format, the Editor pops up some warning dialogs when you pack your map.
- If your `Sound file` refers to a file in your `sounds` directory with a different extension, the Editor doesn't pop up any warning dialogs, and the sound won't play in the game. Always use the correct file extension to avoid unnecessary frustration.

Single File

Set the `Sound file` value to the sound file's path within the `sounds` directory, excluding its extension. E.g. if your sound file is in `Media/sounds/test/evil.wav`, set `Sound file` to `test/evil`. Either `/` or `\` can be used as directory separators.

File Set

To play a number of sound files in a random order, end each file name with two underscores and one or more digits (`_n`). E.g. if your sound files are in `Media/sounds/test/good__1.wav` through `Media/sounds/test/good__7.wav`, set `Sound file` to `test/good`. The numbers at the end of the filename don't have to be consecutive.

Delay Between Sounds

`Scene.(Terrain).Sounds.name: Loop delay.Min`: numeric, in seconds

`Scene.(Terrain).Sounds.name: Loop delay.Max`: numeric, in seconds

Specifies the range of possible lengths for the random pause after each sound before the next sound is played.

Default: 0 – 0; results in continuous play of back-to-back sounds.

After playing a sound, the game inserts a pause of random length before playing the next sound. If the `Min` value is 0, then sounds can be played back to back. If the `Max` value is also 0, then sounds are always played back to back continuously.

Interestingly, the game appears to pick a random value between `Min` and `Max` even if `Min > Max`. You should stick to `Min ≤ Max`, however.

Sound Conditions

`Scene.(Terrain).Sounds.name: Conditions`: text

Specifies the conditions required to play the sound.

Default: blank; the sound is played unconditionally.

You can set a point sound to play only during the day or only at night by setting the `Conditions` value:

- `DAY_FOREST, DAY_WIND`: play the sound only during the day.
- `NIGHT_FOREST, NIGHT_WIND`: play the sound only at night.

Bug: In the game, using the `Change Time` option in the `Tools` menu to switch to day or night does not trigger sound conditions. Instead, use the `Skip Time` feature from the navigation map to change the time.

Note: A point sound works reliably only when the `DAY_FOREST` condition is used together with `DAY_WIND` or when `NIGHT_FOREST` is used with `NIGHT_WIND`. If only a `*_FOREST` condition or only a `*_WIND` condition is used, the sound might not work. When listing both the `*_FOREST` and `*_WIND` conditions, the comma is important, but the space is not.

Ambient Sound Domains

An ambient sound domain is similar to a point sound, but it plays in a certain region without a specific point source. Ambient sounds always play continuously back to back. To create an ambient sound domain, choose [Add Ambient Sound Domain](#) from the context menu (or [SoundDomains – Add SoundDomain](#)).

The Editor lumps the new ambient sound domain in the [SoundDomains](#) feature category along with the other types of sound domains:

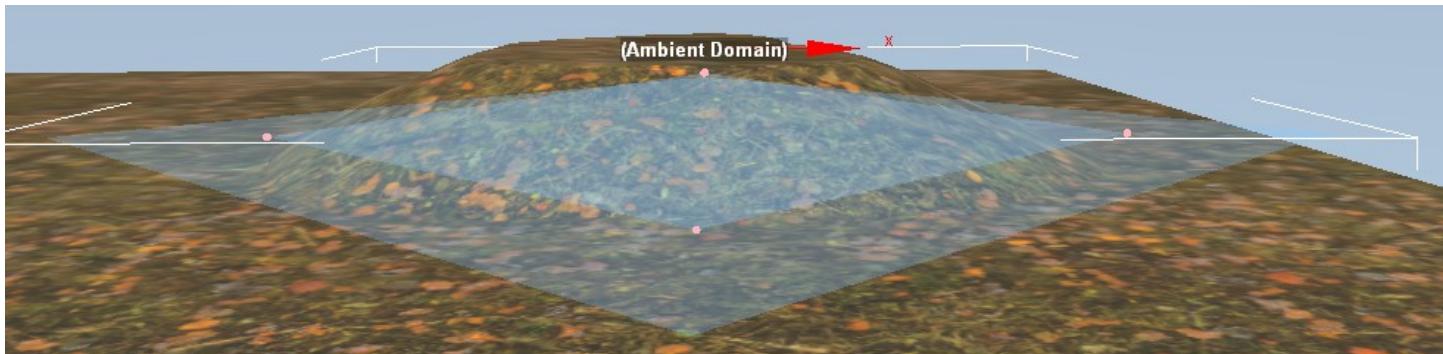
- An ambient sound domain uses a music notes icon ⓘ next to the name, and the domain shape is drawn in pale blue.
- A one-shot sound domain (page 231) uses a sound wave icon ⓘ next to the name, and the domain shape is drawn in tan.
- A no-music sound domain (page 234) uses a plain plane icon ⓘ next to the name, and the domain shape is drawn in dark gray.

Bug: The color of the domain shapes in the main panel has no relation to the color of their associated icons.

All point sounds (page 220) and sound domains play independently, and the game mixes their sounds together as appropriate.

Sound Domain Visualization

When a sound domain is selected, its shape is drawn as a translucent polygon stretched between its vertices. If the ground rises between the vertices, part of the polygon may be underground, but it is displayed without regard to occlusion.



When a sound domain is unselected, it is displayed and labeled only if the [Show all sound domains](#) toolbar button is enabled (page 256).

Name

`Scene.(Terrain).SoundDomains.n name: Name: text`

Specifies the name of the sound domain to display in the `Scene View` at all times and in the main panel when the sound domain is selected.

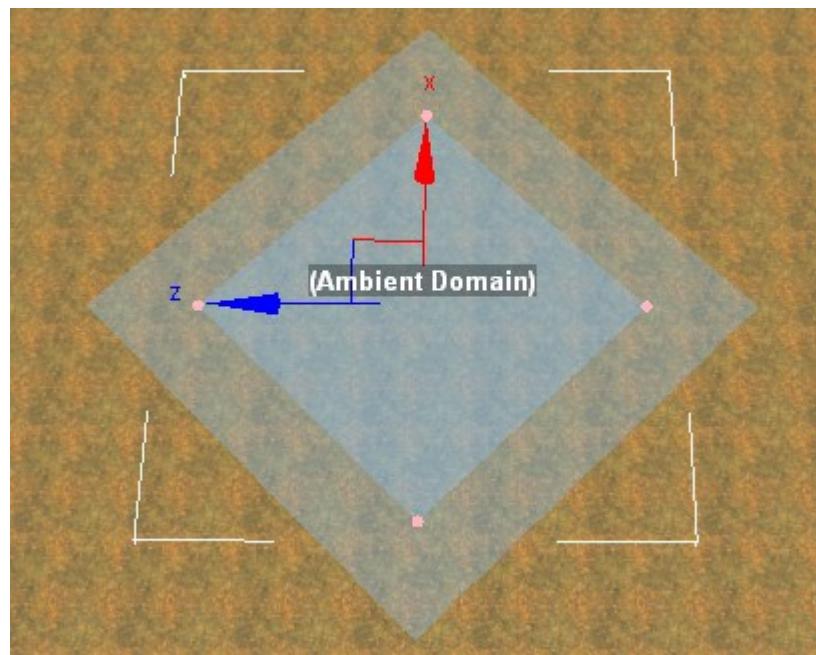
Default: blank; displays as `(Ambient Domain)`.

To help keep your sounds straight, you can assign a name to each sound domain in its `Name` property. The name has no game purpose, however. Names do not need to be unique.

Tip: If you have trouble remembering the meaning of the icons for each type of sound domain, consider including the sound domain type in its name.

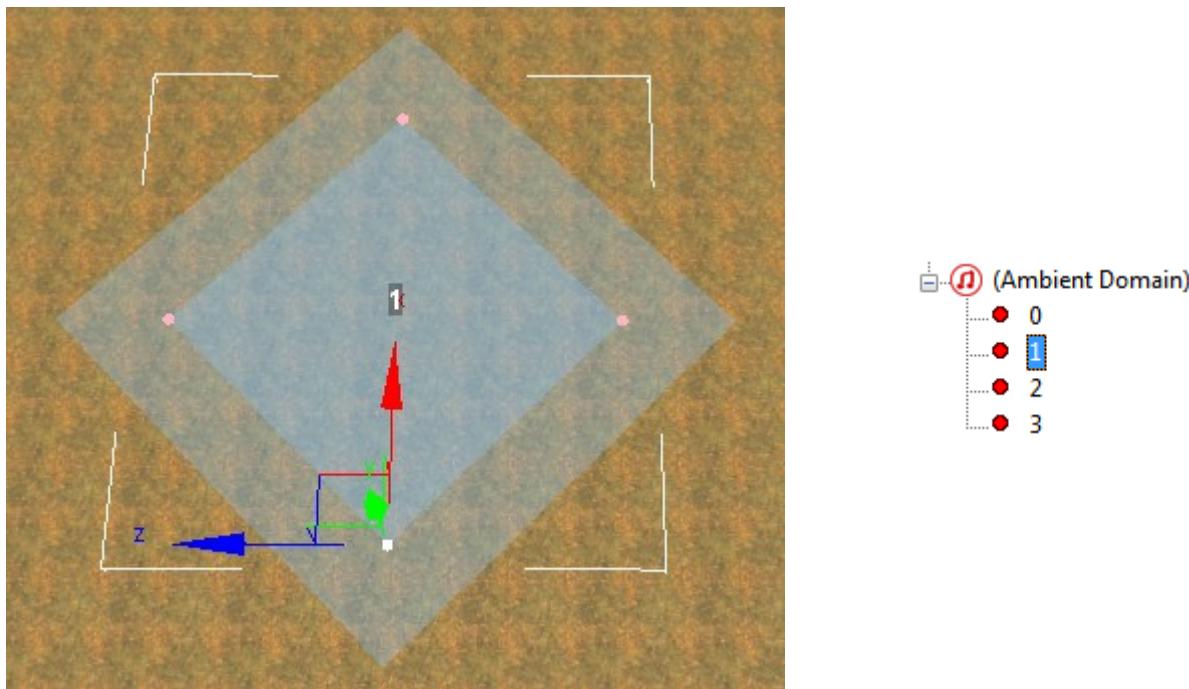
Sound Domain Location and Shape

The sound domain doesn't have a scale or orientation, only a position and a shape.



With the sound domain selected, you can move it in the ground plane. But because the shape encloses a 2-D area, you cannot change its height.

To change the shape of the sound domain, you can select a vertex from the main panel, or you can expand the sound domain in the **Scene View** and select a numbered vertex there. I find that it is sometimes difficult or impossible to click a vertex in the main panel, so I have to select it in the **Scene View** instead.



`Scene.(Terrain).SoundDomains.(n) name.n`: `Position X`, `Position Y`, `Position Z`: numeric, in meters

Specifies the position of the vertex.

Once the vertex is selected, you can move it in the usual ways, either by dragging in the main panel or by typing a new value into its `Position X` or `Position Z` property.

The position coordinate system is described on page 420.

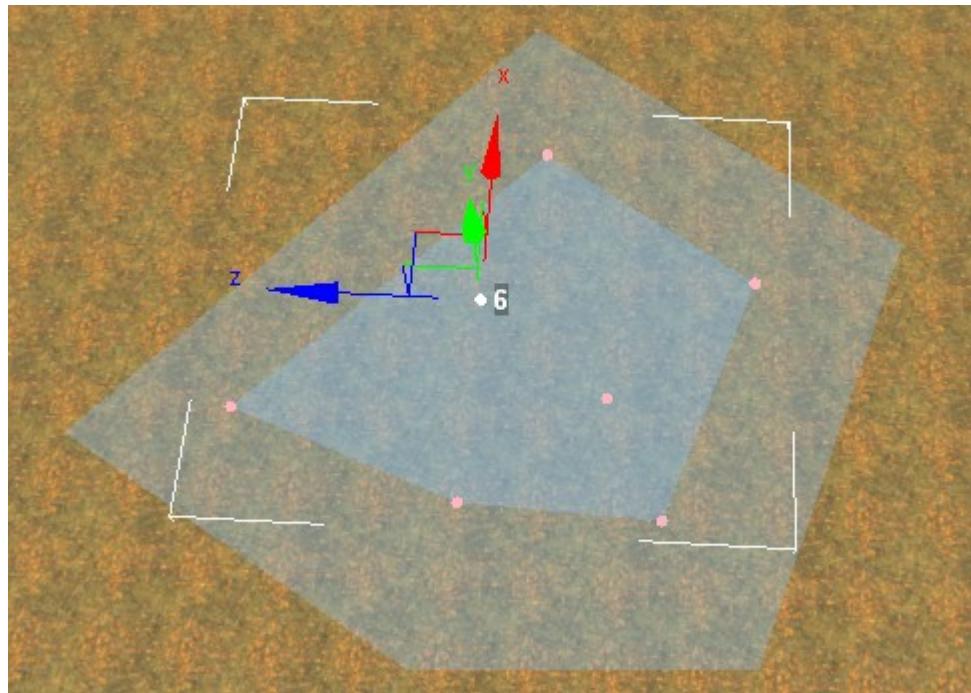
Bug: Despite having height control widgets in the main panel and a `Position Y` property, a sound domain vertex is always attached to ground.

Note that the sound domain as a whole does not have position coordinates in the property panel. When you move the whole sound domain in the main panel, you are actually moving all of its vertexes together.

The more transparent polygon outside of the vertices represents the fading distance, described on page 230.

Update Vertices

No matter how you arrange your vertices, the sound domain is always the convex shape formed by the outermost vertices, ignoring the order of vertices, and ignoring inner vertices.



With the whole sound domain or any vertex selected, you can choose **Update** from the context menu to rationalize the vertices. The Editor removes unnecessary inner vertices and renumbers the outer vertices to follow a clockwise path around the perimeter. The vertex with the smallest X coordinate becomes vertex 0.

(Exception: if the domain has only three vertices, **Update** does nothing, even if the vertices are currently laid out counterclockwise.)

Bug: If the domain has a triangular counterclockwise perimeter formed by the first three vertices and has additional higher-numbered vertices inside the perimeter, **Update** causes the Editor to crash. Similarly, if the domain has four or more vertices all exactly in a line, **Update** causes the Editor to crash.

Add a Vertex

If an outer vertex is selected, you can choose **Add Vertex** from the context menu to add a new vertex to the middle of the segment connected to the next vertex in clockwise order (regardless of the current vertex numbering). If an inner vertex is selected, or the whole sound domain is selected, **Add Vertex** behaves as if the vertex with the smallest X coordinate is selected.

(Exception: if the domain has only three vertices, the new vertex is added between the selected vertex and the next higher numbered vertex. If the whole sound domain is selected, the new vertex is added between vertices 0 and 1.)

Delete a Vertex or a Sound Domain

With a vertex selected, choose **Delete** from the context menu or press the **Delete** key to delete the vertex. Delete is not an option when the domain only has three vertices.

To delete an entire sound domain, select the sound domain and choose **Delete** from the context menu, or press the **Delete** key.

Bug: The context menu does not make an obvious distinction between the context of a selected vertex and a selected sound domain. Be careful which one you have selected.

Duplicate a Sound Domain

To duplicate a sound domain, select the sound domain and choose **Duplicate** from the context menu, or press **Ctrl+D**.

Sound Files

Scene.(Terrain).SoundDomains.^① name: **Sound file:** text

Specifies the relative path and base filename of the sound file(s) to play.

Default: blank; triggers a **File not found** error when the map is packed.

Choose a sound file or set of sound files in the same way as for a point sound (page 222).

Since it doesn't have an aural direction, either mono or stereo sound is permitted for an ambient sound domain.

Overlay

Scene.(Terrain).SoundDomains.^① name: **Overlay:** numeric slider from 0.0 – 1.0

No known purpose.

Default: 1.0.

Saber claims that the **Overlay** property determines the volume mix between the ambient sound domain and the map-wide ambient sound (page 252), but I don't hear any difference. I suspect that it doesn't do anything.

Volume and Fading Distance

Scene.(Terrain).SoundDomains.[②](#) name: **Volume**: numeric slider from 0.0 – 1.0

Specifies the maximum volume of the sound as a fraction of its volume in the sound file.

Default: 1.0.

Scene.(Terrain).SoundDomains.[②](#) name: **Fading distance**: numeric, in meters

Specifies the approximate distance from the edge of the shape through which the sound fades to zero volume.

Default: 5.

The sound plays at the specified volume anywhere within the defined shape. Outside this shape, the sound attenuates to zero approximately at a distance given by **Fading distance** (in meters).

The actual fading distance is shown by the more transparent outer portion of the polygon representing the sound domain in the main panel. There's some weird stuff going on with the way it handles corners, but what you see pictured is where the sound plays in the game, corners and all.

Bug: If **Fading distance** is zero, the game hangs when it tries to load the map.

Keep in mind that the “microphone” for the sound is in the game’s camera, not in the truck. Whether the sound plays and its volume is determined solely by the camera’s 2-D position, regardless of height.

Sound Conditions

Scene.(Terrain).SoundDomains.[②](#) name : **Conditions**: text

Specifies the conditions required to play the sound.

Default: blank; the sound is played unconditionally.

You can set an ambient sound domain to play only during the day or only at night by setting the **Conditions** value in the same way as for a point sound (page 224).

One-Shot Sound Domains

A one-shot sound domain is similar to an ambient sound domain, but it plays each sound in a different random direction from the player and at a different random volume. A one-shot sound domains is rather awkwardly implemented, and its behavior doesn't match what is described in Saber's documentation. It seem best suited to short sounds of only a few seconds duration that are well separated in time from each other. Saber uses the sound of a rockfall as an example.

To create a one-shot sound domain, choose **Add Oneshot Sound Domain** from the context menu.

The Editor lumps the new one-shot sound domain in the **SoundDomains** feature category along with other types of sound domains. A one-shot sound domain uses a sound wave icon next to the name: .

All point sounds (page 220) and sound domains play independently, and the game mixes their sounds together as appropriate.

Name

Scene.(Terrain).SoundDomains. name: **Name:** text

Specifies the name of the sound domain to display in the **Scene View** at all times and in the main panel when the sound domain is selected.

Default: blank; displays as **(Oneshot Domain)**.

To help keep your sounds straight, you can assign a name to each sound domain in its **Name** property. The name has no game purpose, however. Names do not need to be unique.

Sound Domain Location and Shape

Scene.(Terrain).SoundDomains. name.n: **Position X, Position Y, Position Z:** numeric, in meters

Specifies the position of the vertex.

The one-shot sound domain location and shape can be manipulated in the same way as for an ambient sound domain (page 226). The position coordinate system is described on page 420.

Sound Files

Scene.(Terrain).SoundDomains. name: **Sound file:** text

Specifies the relative path and base filename of the sound file(s) to play.

Default: blank; triggers a **File not found** error when the map is packed.

Choose a sound file or set of sound files in the same way as for a point sound (page 222).

I presume that one-shot sound domains should be in mono format (not stereo) to allow the aural direction of the sound to be correctly placed.

Volume Range

`Scene.(Terrain).SoundDomains.(•) name`: `Volume.Min`: numeric slider from 0.0 – 1.0

`Scene.(Terrain).SoundDomains.(•) name`: `Volume.Max`: numeric slider from 0.0 – 1.0

Specifies the range of values for the random maximum volume chosen for each sound playback.

Default: 0.4 – 1.0.

The game appears to pick a random value between `Min` and `Max` even if `Min` > `Max`. You should stick to `Min` ≤ `Max`, however.

Radius Range

`Scene.(Terrain).SoundDomains.(•) name`: `Radius.Min`: numeric, in meters

Specifies the approximate distance from the edge of the domain shape at which the sound remains at its randomly chosen volume.

Default: 10.

`Scene.(Terrain).SoundDomains.(•) name`: `Radius.Max`: numeric, in meters

Specifies a distance over which the sound decreases in volume.

Default: 35.

While a sound is playing, it never becomes entirely inaudible with distance, even at distances much beyond `Radius.Max`. The sound only stops playing at the end of the sound file. However, smaller `Max` values cause the volume to fall off more quickly than larger values.

If `Min` > `Max`, the game doesn't crash, but I have even less idea of what it's actually doing with the numbers. You should probably stick to `Min` ≤ `Max`.

The absolute aural direction of the sound (e.g. northeast) appears to be chosen randomly for each sound, and it stays the same throughout playback no matter where the player moves the truck and camera.

Delay Between Sounds

`Scene.(Terrain).SoundDomains.(•) name` : `Interval.Min`: numeric, in seconds

`Scene.(Terrain).SoundDomains.(•) name` : `Interval.Max`: numeric, in seconds

Specifies the range of values for the random pause after each sound before the next sound is played.

Default: 10 – 15.

After playing a sound, the game inserts a pause of random length before playing the next sound. If the `Interval.Min` value is 0, then sounds can be played back to back. However, this seems to cause poor volume mixing as the game transitions from the old sound to the new sound. I recommend a `Min` value of at least 1.

If `Interval.Min` and `Interval.Max` are both 0, then sounds are always played back to back continuously.

The game appears to pick a random value between `Min` and `Max` even if `Min > Max`. You should stick to `Min ≤ Max`, however.

Weather Intensity Threshold

`Scene.(Terrain).SoundDomains.(•) name` : `Weather intensity threshold`: numeric slider from –1.0 to 1.0

No known purpose.

Default: –1.0.

Saber recommends leaving this property value at its default value of –1.0.

Sound Conditions

`Scene.(Terrain). SoundDomains.(•) name` : `Conditions`: text

Specifies the conditions required to play the sound.

Default: blank; the sound is played unconditionally.

You can set a one-shot sound domain to play only during the day or only at night by setting the `Conditions` value in the same way as for a point sound (page 224).

No-Music Sound Domains

A no-music sound domain is set up similarly to other sound domains, but instead of playing sound in a particular location, it suppresses the map-wide ambient music (page 433) in that location.

To create a no-music sound domain, choose **Add No Music Sound Domain** from the context menu.

The Editor lumps the new no-music sound domain in the **SoundDomains** feature category along with other types of sound domains. A no-music sound domain uses a plain plane icon next to the name: .

Name

Scene.(Terrain).SoundDomains. name: **Name:** text

Specifies the name of the sound domain to display in the **Scene View** at all times and in the main panel when the sound domain is selected.

Default: blank; displays as **(No Music Domain)**.

To help keep your sounds straight, you can assign a name to each sound domain in its **Name** property. The name has no game purpose, however. Names do not need to be unique.

Sound Domain Location and Shape

Scene.(Terrain).SoundDomains. name.n: **Position X, Position Y, Position Z:** numeric, in meters

Specifies the position of the vertex.

The no-music sound domain location and shape can be manipulated in the same way as for an ambient sound domain (page 226). The position coordinate system is described on page 420.

Hysteresis

Scene.(Terrain).SoundDomains. name: **Fade params.Fading distance:** numeric, in meters

Specifies the approximate distance from the edge of the shape through which the no-music state doesn't change.

Default: 20.

The ambient music fades out as the camera enters the no-music zone defined by the domain vertices. The ambient music doesn't fade in again until the camera leaves the zone extended by the **Fading distance**. Placing the two boundaries sufficiently far from each other prevents the ambient music from repeatedly fading in and out due to small changes in camera position.

Tip: A **Fading distance** of 40 would prevent the ambient music from reacting more than once to any camera movements swinging around even a large truck, although if the player also switches the camera to trailer mode, the required distance would be even further. So perhaps just set the **Fading distance** to a moderate amount and let the player live with weird music changes that accompany wild camera movements.

Fade In and Out

Scene.(Terrain).SoundDomains.◆ *name*: **Fade params.Fade in time**: numeric, in seconds

Specifies the fade-in time for the ambient music when leaving the no-music zone.

Default: 5.

Scene.(Terrain).SoundDomains.◆ *name*: **Fade params.Fade out time**: numeric, in seconds

Specifies the fade-out time for the ambient music when entering the no-music zone.

Default: 3.

These fade-in and fade-out times are unrelated to the normal fade-in and fade-out times for ambient music. The ambient music also doesn't fade back in at the beginning of a track like usual, but may fade in to the middle.

Reference Maps

Your map can import another map by reference. The imported map can add features to the map and can modify the terrain within its boundaries.

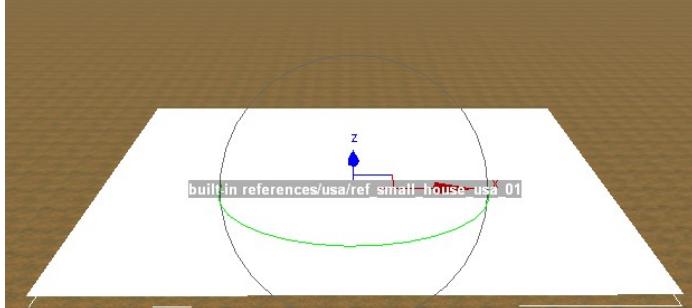
You can create your own reference maps (described further below). Or you can use the built-in reference maps supplied by Saber, described on page 26.

To add a reference to your map, select **Add Reference** from the context menu. A file chooser window appears, by default pointing to your prebuild directory. Navigate to the appropriate directory (if necessary) and select the top-level XML file for the reference map.

When any reference is selected, all references show a highlighted rectangle on the map. The appearance of the rectangle depends on whether each reference has an ***.stg** file. If the ***.stg** file is present, the reference is represented by its terrain colors (as seen in its **Terrain** panel, i.e. its navigation map without 3-D features). If the reference doesn't have an ***.stg** file, it is represented by a plain white rectangle instead. You can open any of the built-in reference maps and rebuild their terrain to generate an ***.stg** file for each one you are interested in.

When no references are selected, each reference rectangle is shown as a grid of black lines representing its terrain blocks.

Rebuild the terrain (page 65) to make all features from the reference visible in your map. You may also notice that if you reload your map, some features are updated from the reference, but not all. In general, rebuilding the terrain is the canonical way to fully update references after any change to reference properties.



The features imported from the reference map are not a permanent part of your map and do not disrupt any of the underlying map data. You can move or delete the reference, and your original map contents reappear when you rebuild the terrain.

Scene.(Terrain).References.ref_path: **STG**: read-only text

Indicates the path to the reference map, ignoring the **prebuild/** prefix and the **.xml** suffix.

Unlike for other features, a reference's position is presented as a single property with three values.

Scene.(Terrain).References.ref_path: Position: 3-D position

Specifies the position of the reference as $(X; Y; Z)$.

Default: ground level in the center of the main panel.

Bug: Despite having a Y value in its position, a reference is always attached to ground. Edits to the Y value are discarded.

Scene.(Terrain).References.ref_path: Angle: numeric

Specifies the clockwise rotation of the reference.

Default: 0.0.

Bug: In some cases, a reference that extends too far off the edge of the map causes the Editor to crash when it rebuilds the terrain. I can't figure out the pattern, but if happens to you, pull your reference further onto your map, and the crash should go away.

The coordinate systems for position and rotation are described on page 420.

Create a Reference Map

Any map that you have created can be used as a reference.

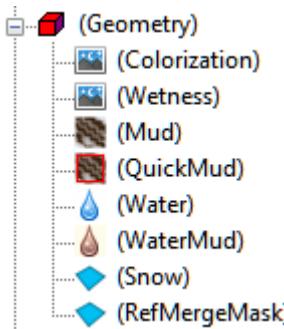
If you create a map only to be used as a reference, it doesn't need to be named starting with **level**. The example reference maps start with **ref**, which seems like a fine convention.

For best results, ensure that the edges of the reference map are at the default height (page 73). This allows the smoothest integration when the map is used as a reference.

Reference Merge Map

You can create a mask which defines which parts of the reference map to use and which parts to ignore. This mask is called the reference merge map. Depending on the properties applied to the reference, the merge map is applied to various painted properties in various ways described in the sections below.

By default, a reference map does not have a merge map. To create one, open the reference map and select **Create Ref Merge Map** from the context menu of any **(Geometry)** feature in the **Scene View**. This creates a new feature under **(Geometry)** called **(RefMergeMask)**, and **Create Ref Merge Map** is removed from the context menu.



To paint the merge map, select `Scene.(Terrain).(Geometry).(RefMergeMask)`. This brings up a brush tool whose only mode is `Flatten + Ref infl.`

The merge map is initialized with the entire map painted yellow, which means that the entire map is used as a reference. To allow areas of the reference map to be ignored, clear the yellow tint from those areas. You may want to soften the edges of the merge map to avoid a sudden change in terrain height at the edge of the reference.

Combined Features

Features from the reference map are combined with your map as described in the following sections.

Combined Terrain Height

The reference map measures its painted terrain height relative to its default height (page 73). When you include a reference map, its relative terrain height is added to the terrain height of your map at each point.

`Scene.(Terrain).References.ref_path`: `Flatten: True/False`

Specifies whether your map is flattened below the reference before the reference height is added.

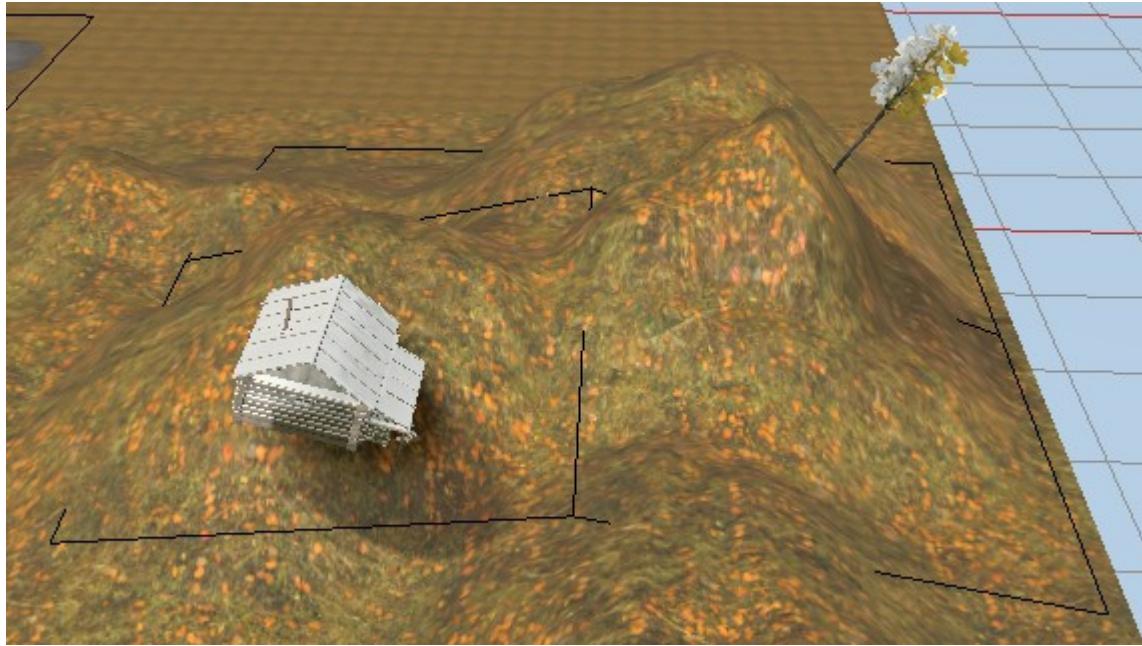
Default: `True`.

If `Flatten` is `True`, your map is flattened to its height at the center position of the reference map before the relative terrain height of the reference map is added. This ensures that the reference map keeps the shape of its terrain without distortion.

If `Flatten` is `False` (or at the edges of the reference when `Flatten` is `True`), the terrain from your map ends up tilting the terrain in the reference. If the base point of a model in the reference is on terrain that gets tilted, in most cases the model is equivalently tilted, even if the model is floating or buried. The exception is a few light pole models, which remain in their original orientation. Distributed plants are also tilted, but only to the limited extent that is generally allowed by their distribution.

Bug: Individual plants are tilted regardless of their `Perpendicularity` property value.

Bug: The tilt is only applied in the east-west direction of your map. If the terrain is tilted north-south, that tilt is not applied to reference models, individual plants, or distributed plants.



You can apply a further correction factor to the terrain height under the reference.

`Scene.(Terrain).References.ref_path: HeightOffset`: numeric, in meters

Specifies the additional height that is added (or subtracted) from the relative terrain height from the reference.

Default: 0.0.

Bug: Objects with a `Land` or `Do Land` property value of `True` follow the changes in terrain height. However, other objects (particularly models) are left at the height they would have been without the `HeightOffset`.

Terrain Height Mask

All terrain height changes caused by the reference are subject to a mask.

`Scene.(Terrain).References.ref_path: ApplyMergeMap`: `True/False`

Specifies whether the reference's merge map is used.

Default: `False`.

If `ApplyMergeMap` is `True` and the reference has a merge map, it is used as a mask when applying the terrain height of the reference. The strength of the paint in the merge map determines the amount of height that is applied by the `Flatten` property, the `HeightOffset` property, and the reference terrain height.

If `ApplyMergeMap` is `False` or the reference doesn't have a merge map, a default mask is used. The default mask uses applies the full reference height across most of the reference rectangle, but it slopes away over the last ~5 meters to the edges of the rectangle.

Combined Mud

Regardless of the `ApplyMergeMap` setting, mud and quickmud are applied by the reference map in addition to mud that already exists on your map.

Combined Colorization

Colorization is applied using the same mask as the terrain height. I.e. depending on the value of `ApplyMergeMap`, either the reference merge map or a default mask is used.

Combined Wetness

Wetness uses the same mask as the terrain height, either the reference merge map or a default mask. However, within the painted area of the mask, wetness can be applied from the reference map in a few different ways.

`Scene.(Terrain).References.ref_path: WentessBlendMode`: dropdown menu: `Replace` or `Add`

Specifies how wetness is applied from the reference map.

Default: `Replace`.

If the `WentessBlendMode` property is set to `Replace`, wetness on the original map is ignored where reference wetness is applied.

If the `WentessBlendMode` property is set to `Add`, wetness from the reference map is combined with the original wetness on your map.

Combined Snow

Snow uses the same mask as the terrain height, either the reference merge map or a default mask. However, within the painted area of the mask, snow can be applied from the reference map in a few different ways.

`Scene.(Terrain).References.ref_path: Snow`: dropdown menu: `Replace`, `Add`, or `Ignore`

Specifies how snow is applied from the reference map.

Default: `Replace`.

If the `Snow` property is set to `Replace`, snow on the original map is ignored where reference snow is applied.

If the `Snow` property is set to `Add`, the reference snow depth is added to the original snow depth.

If the **Snow** property is set to **Ignore**, the reference snow depth is not applied, and the snow on the original map is retained throughout the reference rectangle.

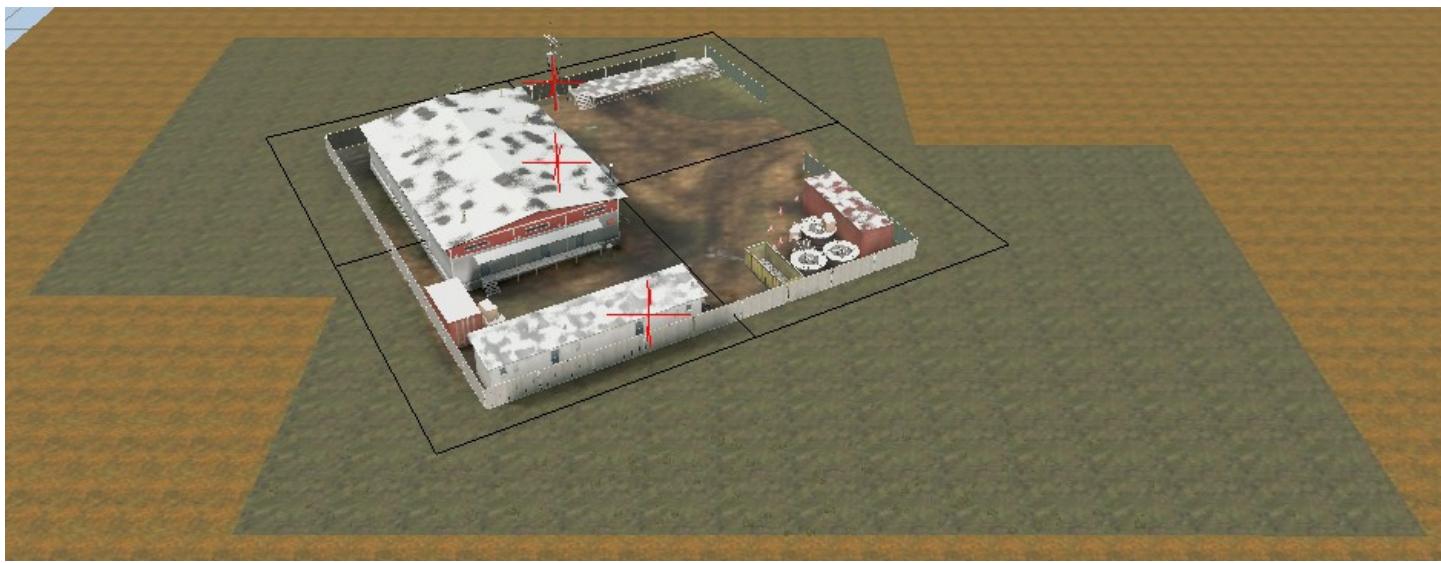
Combined Materials and Material Layers

By default, the materials from the reference have priority over your map materials for all terrain blocks touched by the reference, but you can choose to keep your original map materials instead.

Scene.(Terrain).References.ref_path: ApplyMaterials: True/False

Specifies whether materials from the reference map are applied.

Default: **True**.



Materials applied by references do not appear to count against the maximum number of materials that your map can use. This may be a sneaky way to get extra materials on your map if desired.

Bug: If the terrain blocks of the reference are exactly aligned with the terrain blocks of your map, and **ApplyMaterials** is **True**, the terrain blocks on your map use the reference map materials where they touch the edges of the reference map, even though they don't overlap.

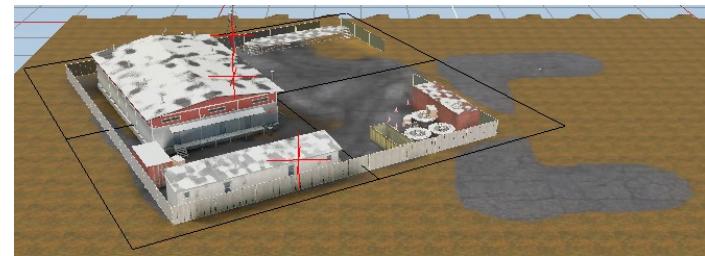
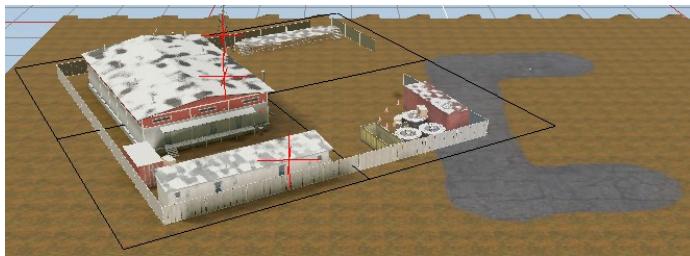
Independently, you can choose whether to use the reference's choice of material layers or to keep the original material layers from your map.

Scene.(Terrain).References.ref_path: MergeMaterialMasks: True/False

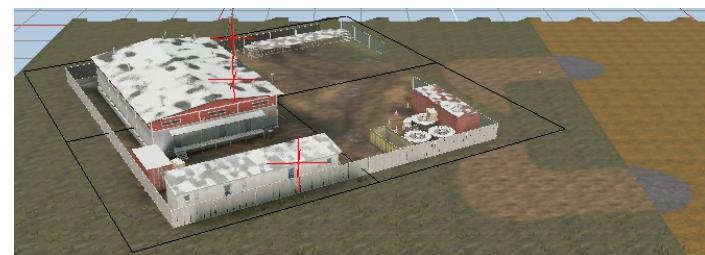
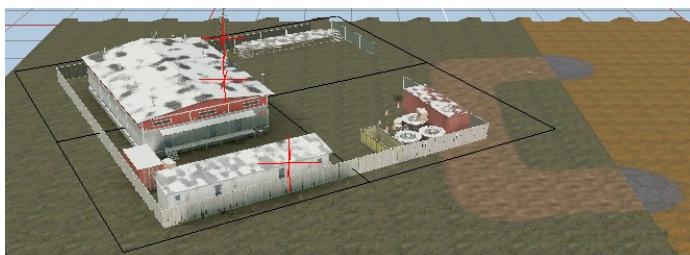
Specifies whether the choices of material layers from the reference map are applied.

Default: **True**.

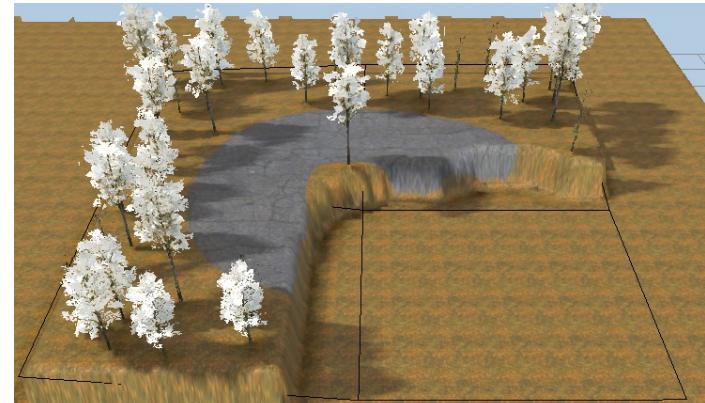
The below screenshots show the results when `ApplyMaterials` is `False` and `MergeMaterialMasks` is `False` or `True`. When `MergeMaterialMasks` is `True`, the choice of material layers is made only within the reference rectangle.



The below screenshots show the results when `ApplyMaterials` is `True` and `MergeMaterialMasks` is `False` or `True`.



Material layers use the same mask as the terrain height, either the reference merge map or a default mask. The reference map determines the material layers only where the merge map is painted.



Combined Distributions

You can independently choose whether to apply the reference merge map to distributions.

`Scene.(Terrain).References.ref_path: ApplyDistributionMergeMap: True/False`

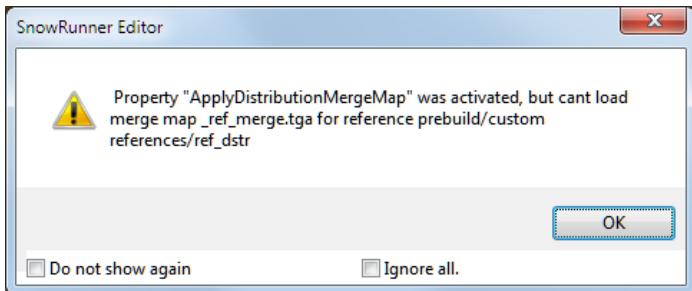
Specifies whether distributions are masked by the reference merge map.

Default: `False`.

If `ApplyDistributionMergeMap` is `False`, the reference distributions are applied **in addition to** the distributions on your base map. On the other hand, if `ApplyDistributionMergeMap` is `True`, then **only** the reference distributions are applied within the area painted by the merge map, and the reference distributions are not applied outside that area.



If `ApplyDistributionMergeMap` is `True`, but the reference does not have a merge map, then the Editor complains, then applies only the reference distributions within the reference rectangle.



Note that the reference distribution is always reapplied from scratch. Where the plants were placed on the standalone reference map has no bearing on where plants are distributed when the map is referenced.

Combined Objects

Models, individual plants, overlays (and their associated brush models), rivers, sounds, and sound domains from the reference map are imported without any effect on objects that already exist on your map.

On the other hand, trucks and zones in the reference map are always ignored.

Bug: The height of an elevated overlay is copied directly from the reference map without regard to the height of your terrain. If your map has a different maximum height than the reference or if the reference map is not placed at the same height as your map, the elevated overlay will appear at the wrong height. It's probably best to simply not use elevated overlays in a reference map.

Roads and rivers combine or collide in the same way as they would if they were normally placed on the same map.

Bug: MudRunner had an [AboveReferences](#) property that would ensure that the colors of a reference river don't override the colors of a river on your map, and Saber's documentation says that SnowRunner has the same property, but no such property exists in the SnowRunner Editor.

An object from a reference map is suppressed if it is off the edge of **your map**.

On the other hand, an object that is placed beyond the edges of **the reference map** is not cut off at the edge of the reference rectangle. In general you should probably avoid creating objects that extend beyond the edge of the reference map and even more strongly avoid placing objects entirely off the edge of the reference map.

A river or road can pose a challenge if you want it to merge with another river or road in the containing map. In most cases it is easiest to leave it off of the reference map and rely on the containing map to create the river or road through the reference map area.

Combined Water Speed, Foam, Flow, and WaterMud

Regardless of the [ApplyMergeMap](#) setting, water speed, foam, flow, and mud color are applied by the reference map in addition to these attributes that are already painted on your map. Don't ask me how the flow directions are combined. All I can say is that you probably want to avoid that.

Inter-Reference Conflicts

References are applied to the map one at a time. If two references overlap (or otherwise share a terrain block), features added by the first reference are treated as being part of the original map while the second reference is applied.

Scene.(Terrain).References.ref_path: Layer: integer

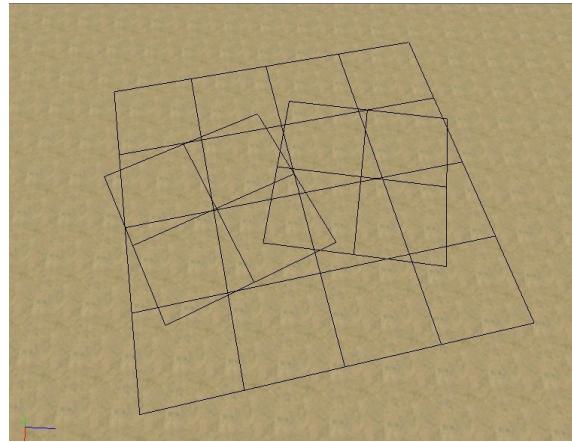
Specifies the priority of the reference when the order of references is determined

Default: 0.

References are applied from the lowest **Layer** to the highest **Layer**. **Layer** values do not need to be consecutive. If two references have the same **Layer** value, the references are applied in the order that they are listed in the **Scene View**.

References within References

If you include a reference that itself includes references, the Editor does not apply those subreferences. However, you can choose **Paste References from this Reference** from the context menu. This adds the referenced subreferences to your map with the appropriate position and orientation to match how they are used in the primary reference.



To remove the pasted subreferences, choose **Delete References from this Reference** from the context menu of the primary reference. Note that the Editor does not actually track which references came from where. It simply deletes any reference whose center lies within the rectangle of the selected reference.

Mutator

You can apply a mutator that changes selected reference features to better fit your map. E.g. a reference designed for a particular country and season can be mutated to match the scenery of a different country and season. A single mutator is applied across all references.

Scene.(Terrain): Mutator: text

Specifies the name of the mutator to apply.

Default: blank; if **Mutator** is blank or invalid, no mutator is applied.

The following built-in mutators are provided: **rus**, **wis**, **rus_sn**, **tayga_sn**, **can**, and **ru_05**. However, each of these was created by Saber to provide only the mutations that they needed for a specific map. Since none of the built-in mutators provides a complete set of mutations, I hesitate to recommend their use.

Instead, it is easy to create your own mutator to meet your needs. For details, see page 431.

Toggle No References

The toolbar has a toggle button called `No references`. When the toggle is enabled, rebuilding the terrain causes the Editor to hide references instead of rebuilding them. Only the materials from the reference are applied, and only if `ApplyMaterials` is `True`.

As with other terrain rebuild shortcuts, `No references` is ignored when packing the map.

Groups

Groups allow some amount of organization in maps which may contain a very large number of objects.

A group can only be created within a feature category. A group cannot span categories, and groups cannot be nested.

Groups are supported within the feature categories for models, plants, distributions, overlays, rivers, references, and zones. Groups are not supported for materials, trucks, sounds, sound domains, or river markups.

A new group is initially assigned a random integer as its name. This can be changed to any desired name.

```
Scene.(Terrain).Models.group name: Name: text  
Scene.(Terrain).Plants.group name: Name: text  
Scene.(Terrain).Distributions.group name: Name: text  
Scene.(Terrain).Overlays.group name: Name: text  
Scene.(Terrain).Rivers.group name: Name: text  
Scene.(Terrain).References.group name: Name: text  
Scene.(Terrain).Zones.group name: Name: text
```

Specifies the name to use for the group in the [Scene View](#).

Default: random integer.

Features within a group are recorded in a separate XML file which is saved in the map's `prebuild/map_name/subgroups` directory. The XML file is named based on the feature category and the group name. If you change the group name, the XML file is renamed the next time you save. (And the old filename is deleted.)

This XML file can be copied into another map's `subgroups` directory to effectively copy all of the grouped features to that map. Their locations from the original map may not be appropriate for the new map, but it is relatively easy to select all of the grouped features in the new map and move them together (page 259).

Group Property of a Feature

If a feature category supports grouping, every feature in that category includes a group property.

`Scene.(Terrain).Models.type`: **Group**: dropdown menu
`Scene.(Terrain).Plants.type`: **Group**: dropdown menu
`Scene.(Terrain).Distributions.brush_list`: **Group**: dropdown menu
`Scene.(Terrain).Overlays.type`: **Group**: dropdown menu
`Scene.(Terrain).Rivers.River n`: **Group**: dropdown menu
`Scene.(Terrain).References.ref_path`: **Group**: dropdown menu
`Scene.(Terrain).Zones.zone_id`: **Group**: dropdown menu

Specifies the group of features that this feature is part of.

Default: **MainGroups**; the feature is not in a group.

You can change what group the feature is in or remove it from all groups by changing this property value using its dropdown menu or by using the context menu (below). Selecting a new group from the property's dropdown menu immediately updates the feature's position in the **Scene View**.

Group Context

When a group or a feature within a group is selected, extra options are added to the context menu to interact with that group. This is in addition to the other context menu categories described beginning on page 64.

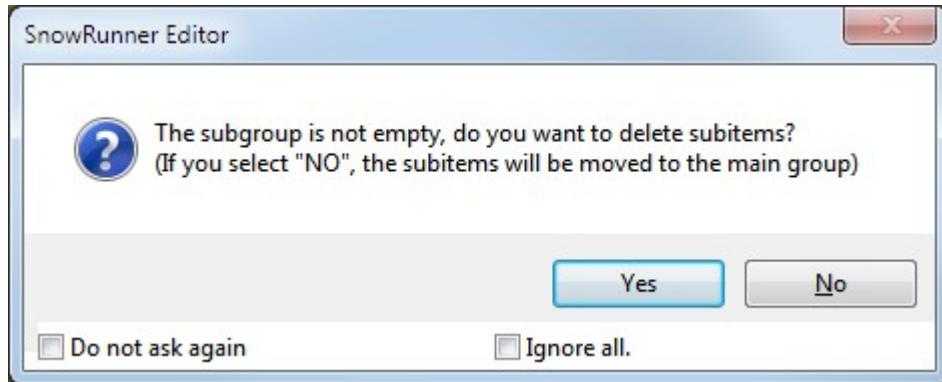
Group - Add Feature adds a new feature to the same group.

Group - Collapse All contracts only that group in the **Scene View** (and nothing else in the feature category).

Group - Expand All expands only that group in the **Scene View** (and nothing else in the feature category).

Bug: **Group – Fly to** flies to the selected **terrain block**.

When a group is selected (not a feature within a group), **Group – Delete** deletes the group. If the group isn't empty, the Editor asks whether you want to delete the features in the group.



Bug: If you select **No**, the Editor crashes. If you want to preserve the features in the group, you have to manually move features out of the group (below) before deleting the group.

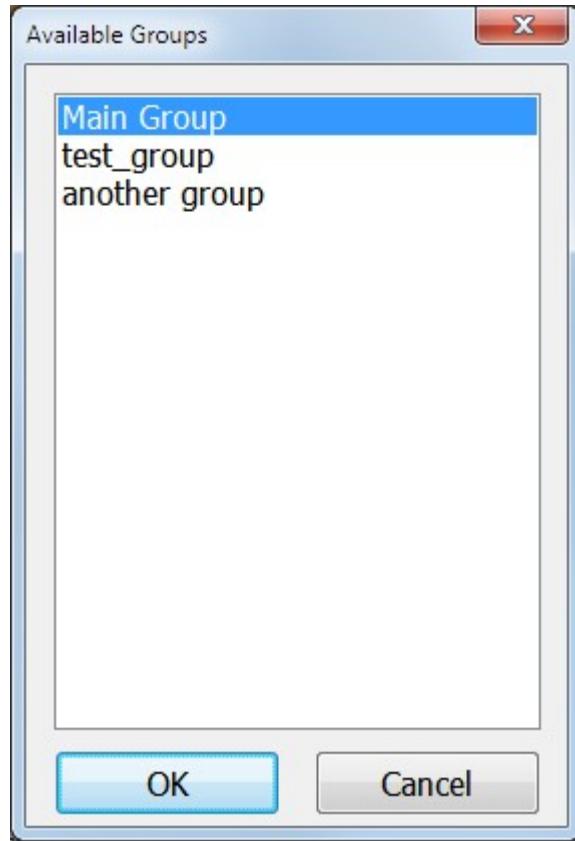
Bug: After a group is deleted, it remains as an option for the **Move to other group** context menu item or the **Group** property. A feature that is moved to a deleted group is lost. Another group can be renamed to the same name as the deleted group, which adds an extra layer of confusion. The Editor only fully removes the deleted group when you reload the map.

Change Group

When a feature category includes one or more groups, and a feature in that category is selected, then the context menu adds an option to move the feature to another group.

Bug: The **Move to other group** option is placed in the context of the feature category or the group name, which makes no intuitive sense since it's the feature itself that is being moved.

Features - Move to other group or *Group - Move to other group* pops up a dialog to move the selected feature to another group or back into the main category list (called **Main Group** in the dialog).



Bug: Using this context menu option does not immediately update the **Group** property value that is displayed for the moved feature. The feature must be deselected and reselected to see the new **Group** value.

You can also change what group a feature is in by changing the value in its **Group** property, described above.

Top-Level Editor Properties

The following properties apply to the map as a whole.

The map name, description, dimensions, and maximum height were covered in the introduction on page 70. Note that the Zone Settings Editor also has top-level properties for the map, described on page 288.

Animation Speed

Scene: Time speed: numeric slider from 0.0 – 2.0

Specifies how fast to display animations in the Editor.

Default: 1.0; normal speed

Animations include model animations (e.g. wind_mill_01) and the wind effect on plants.

Time speed affects only the Editor, not the game. It is actually applied to the Editor, not the map, so it applies to all maps that are loaded in the same Editor session. It is not saved anywhere, so it resets to its default value when you restart the Editor.

The **Pause** button on the toolbar is a shortcut to quickly set the **Time speed** value. If **Time speed** is non-zero, click **Pause** to set the **Time speed** to 0.0 (which also depresses the **Pause** button). If **Time speed** is zero, click **Pause** to set it to 1.0 (and release the **Pause** button).

Bug: When you press **Pause**, the **Time speed** value displayed in the property panel doesn't update until you click away and reselect the **Time speed** property. However, changing the **Time speed** to zero or non-zero correctly updates the **Pause** button state immediately.

Mud Type

Scene.(Terrain): Mud Type: dropdown menu with only one option

Does nothing.

Mutator

The **Mutator** property is described on page 245.

Procedural Snow Coverage

The **Procedural snow coverage** property is described on page 173.

Sun Static Direction

Scene.(Terrain): Sun Static Direction: direction vector

Specifies the direction in which the static sun shines.

Default: (0.70711; -0.70711; 0)

The static sun is used to create day static shadows (page 136). Direction vectors are described on page 424.

Make sure that the sun points down (negative Y), or it won't work very well.

Sky Preset

Scene.(Terrain): Sky Preset: text

Specifies a preset file that defines the sky.

Default: blank; a default sky is used.

The Saber documentation officially endorses `sky_ru_02`, `sky_us_01`, and `sky_us_02`, but other skies are also available:

- (blank): pale blue sky with two sets of counter-rotating low pale gray clouds
- `sky_main_menu`: very blue sky (almost purple), lots of clouds, large sun with halo
- `sky_main_menu_ru03`: blue sky, slow low white and pink clouds, overpowering sun
- `sky_ru_02`: very blue sky (almost purple), lots of clouds, large sun with halo; used in Taymyr; displays a warning when packing
- `sky_us_01`: blue sky, medium cloud density, small sun; used in Michigan
- `sky_us_02`: blue sky, slow low white and pink clouds, overpowering sun; used in Alaska

Bug: `sky_default`, `sky_us_01_ttt`, and `sky_us_02_ttt` display one or more warning dialogs when you pack the map, and the resulting skies are substandard.

It may be possible to create a custom sky preset. TBD.

Ambient Sounds

Scene.(Terrain): Ambient Preset: text

Specifies a preset file that defines the ambient sounds in various conditions.

Default: blank; no ambient sounds.

The Saber documentation officially endorses `snd_amb_ru_summer`, `snd_amb_us_autumn`, and `snd_amb_us_winter`, but other ambient sound presets are also available. Each of these has a constant background loop for each combination of day and night, forested or open. Each also defines river sounds that can be used by river markup (page 215). Each also adds various random noises such as generic birds, wind gusts, and trees creaking. Particularly distinguishing sounds are as follows:

- `snd_amb_ru_summer`: crow, woodpecker, owl, wolf
- `snd_amb_ru_autumn`: crow, woodpecker, magpie
- `snd_amb_ru_winter`: crow, owl, deer, wolf
- `snd_amb_ru_winter_menu`: owl only
- `snd_amb_ru_winter`: same as `_ru_winter`, but with tigers instead of deer; sounds legit
- `snd_amb_us_summer`: crow, insects, wolf
- `snd_amb_us_autumn`: crow, owl, wolf
- `snd_amb_us_autumn_01_01`: same as `_us_autumn`, but wolf is only in forested areas, varied river sounds
- `snd_amb_us_autumn_01_04`: same as `_us_autumn`, but varied river sounds
- `snd_amb_us_winter`: crow, woodpecker, wolf
- `snd_amb_yukon`: crow, woodpecker, owl
- `snd_amb_yukon_menu`: only constant background loops and river sounds; no creaking trees or animal noises

A custom set of ambient sounds can be created. TBD.

There is also ambient music, but it requires creating a custom file outside the Editor. Ambient music is described starting on page 433.

Daytime Presets

Daytime presets define the light, fog, and weather conditions at various times of day.

`Scene.(Terrain): Daytime Presets.Night`: text
`Scene.(Terrain): Daytime Presets.Night to Day`: text
`Scene.(Terrain): Daytime Presets.Day Early Variants (; - separator)`: text
`Scene.(Terrain): Daytime Presets.Day Mid Variants (; - separator)`: text
`Scene.(Terrain): Daytime Presets.Day Late Variants (; - separator)`: text
`Scene.(Terrain): Daytime Presets.Day to Night`: text

Specifies a preset file that defines the conditions at the corresponding time of day.

Default: blank; generic default conditions.

The game comes built-in with three sets of files that can be used with the daytime presets. Here are the values that the game uses for Michigan:

- Night: night_us_01
- Night to Day: night_to_day_us_01
- Day Early: day_1_us_01;day_1a_us_01
- Day Mid: day_2_us_01;day_2a_us_01
- Day Late: day_3_us_01;day_3a_us_01
- Day to Night: day_to_night_us_01

For each of these property values, you can replace us_01 with us_02 for Alaska, or with ru_02 for Taymyr. Note that the Alaska presets are for a winter map and include snowy weather. Michigan and Taymyr are for a non-winter map and include rainy weather.

You can mix up these presets to a certain extent, but there's a limit to how much you can practically accomplish. The game combines each preset with lighting appropriate to that time of day, so using a preset from a different time of day can result in an eye-searingly bright scene, or a completely black scene, or both in quick succession.

The three Day presets allow variants that the game chooses among, and the built-in presets supply two variants for each. The day_n variant is the base version, and the day_na variant defines more inclement weather conditions, e.g. more fog, higher chance of precipitation, and heavier precipitation. You can remove one of the variants for any or all times of day to restrict the preset conditions. Note that the variants are separated by a semicolon, and a space is **not** allowed after the semicolon. Semicolon-separated variants can only be used in the properties that explicitly support them.

Be careful typing the daylight presets. If any preset is not named correctly, the game crashes when it loads the map.

Custom daytime presets can be created. TBD.

Daytime Presets over Time

As time progresses, the game blends the conditions of adjacent presets together to smooth the transitions. You can patiently wait for an hour of real time to see the sequence for a whole day of game time, or you can use the Change Time function in the dev tools menu (page 39) to quickly review each of the unblended daytime presets.

By default, the game shows the blend for the current time. The first time you click Change Time, it switches to showing the Night preset. For each additional click, it sequences to the next preset or the next variant in the preset. After showing the Day to Night preset, the next click returns back to the blend for the current time.

Force Daytime Preset

Scene.(Terrain): Daytime Presets.Force.Name: text

Specifies a preset file to use at all times of day, regardless of other settings.

Default: blank; different daytime presets are used for different times of day.

Bug: If you use a custom daytime preset in **Force.Name**, the game instead shows a default daytime preset. Only built-in daytime presets can be forced.

Scene.(Terrain): Daytime Presets.Force.Night Factor: dropdown menu: **Day**, **Night**, or **Night to Day**

No known purpose. The game behaves as if it is daytime regardless of the setting.

Default: Day.

Bug: **Night Factor** resets to **Day** when you pack the map. The value saved to the map's XML file while packing correctly reflects the most recent edit, and future saves also have the correct value, so it appears to be just a display issue.

Extrudes to Wetness

The **Extrudes to Wetness** and **-threshold** properties are described on page 108.

Background Water

The **Background Water** properties are described on page 214.

Details of the Map Editor Window

The map editor window was introduced on page 47. This section gives full details for those parts of the window that were not fully described in the introduction. This information is mostly of use to diehard keyboard users and people having trouble sleeping. Everyone else can probably skip it.

Menu Bar

The menu bar is fully described in the introduction on page 47.

Toolbar

Under the menu bar is a toolbar with a row of buttons for various features. You can hover over each button to get a description of what it does.



Many of the buttons are toggles. Click once to enable, click again to disable. The Editor saves the enable/disable setting for many of these toggles between sessions (page 37). The remaining toggles reset to their defaults when you restart the Editor.

Save (shortcut: **Ctrl+S**) saves the map's named properties for all features from memory to disk. This differs from bitmaps painted with a brush, which are updated on disk as soon as a change is committed.

Wireframe (shortcut: **Ctrl+W**) is a toggle button. When enabled, it redraws everything using only colored outlines for all triangles, instead of filled shapes. The setting is not saved between sessions.

Grid (shortcut: **Ctrl+G**) is a toggle button. When enabled, it draws a reference grid below the terrain at height 0. The size of each square depends on the zoom level, so it can't be used as a measurement tool, but it is sometimes helpful to keep yourself oriented. The setting is not saved between sessions.

Night is a toggle button. When enabled, it changes the lighting to night mode. This is useful for testing how everything looks in the dark, especially lights. The setting is not saved between sessions.

Fog is a toggle button with no known purpose. The setting is not saved between sessions.

Statistics (shortcut: **Ctrl+T**) is a toggle button. When enabled, it puts a block of text in the upper left detailing the camera position and orientation (as *position > orientation*) and statistics regarding what is being drawn. The coordinate systems for position and orientation are described on page 420. The setting is not saved between sessions.

Reload Resources reloads most data used by the map (as opposed to data **about** the map). E.g. this includes shapes, texture bitmaps, etc. This is only useful if you are creating and editing custom assets that are used in your map.

Open File (shortcut: **Alt-Shift-O**) pops up a dialog box listing all files in your Media directory. When you select a file from this list, the Editor attempts to open the file in some manner:

- If you select a map or mesh XML file, it opens in the SnowRunner Editor. These can also be opened via the **File View** (page 53)
- If you select another XML file, it opens in whatever editor Windows lists as your default (usually some sort of text editor).
- If you select a bitmap/texture file, the Editor displays an error dialog requesting that you specify a texture editor path in the Editor settings. However, although the MudRunner Editor had such a setting, the Snow Runner editor does not, so the error message just looks silly now.
- Finally, if you open some other file type, the Editor displays an error dialog to say that it isn't supported.

Duplicate selected (shortcut: **Ctrl+D**) makes a copy of the currently selected feature. See page 67 for details.

Pause is a toggle button that toggles animations. See page 251 for details. The setting is not saved between sessions.

Quick mode is a toggle button. When enabled, rebuilding the terrain causes the Editor to rebuild large features as usual, but it hides grass, shadows, and other minor features rather than rebuilding them. This can speed up the rebuild quite a bit, especially on large maps. See page 65 for details. The setting is not saved between sessions.

No references is a toggle button. When enabled, rebuilding the terrain causes the Editor to hide references instead of rebuilding them. See page 246 for details. The setting persists between Editor sessions.

Local transform is a toggle button. When enabled, an object's interface widget is rotated to match the orientation of the object. This is described further on page 80. The setting is not saved between sessions.

Enable autorebuild terrain is a toggle button. When enabled, the terrain is rebuilt whenever you commit a change to a (**Geometry**) feature (page 98), and plant locations are updated whenever you commit a change to a distribution bitmap (page 151). The setting persists between Editor sessions.

Show all sound domains is a toggle button. When enabled, sound domains are displayed and labeled even when not selected. See page 225 for details. The setting persists between Editor sessions.

Hide Strings is a toggle button. When enabled, it hides the labels that normally display near each object. Instead, only the selected feature is labeled. The setting persists between Editor sessions.



Show HeightMap Angles is a toggle button. When enabled, it recolors the terrain to show the angle of the terrain at each point. Further information is in the Height Information section on page 98. The setting persists between Editor sessions.

Use Ruler is a toggle button. When enabled, a line segment or path can be overlaid on the terrain, and the Editor reports its length. More detail is on page 204. The setting is not saved between sessions.

Terrain brightness displays a dialog box where the brightness of the terrain can be adjusted. This affects the display in the Editor only, and other map features are not affected. This is particularly useful on snowy maps with a lot of white terrain. The setting is not saved between sessions.

Pack terrain converts the map to a format that can be played in the game and uploaded to mod.io. More detail is on page 39.

Collision Notification is a toggle button. When enabled, dynamic models are highlighted if they collide with the terrain, other models, or individual plants. More detail is on page 124. The setting persists between Editor sessions.

Zone settings opens the Zone Settings Editor. More detail begins on page 266. **Caution:** while the Zone Settings Editor is open, the main SnowRunner Editor window is completely unresponsive. Close the Zone Settings Editor to return to the main SnowRunner Editor.

Region settings opens the Region Settings Editor. More detail is in 367. **Caution:** while the Region Settings Editor is open, the main SnowRunner Editor window is completely unresponsive. Close the Region Settings Editor to return to the main SnowRunner Editor.

Main Panel

The main panel is described pretty extensively in its introduction on page 50, so this section describes only a few additional interface options of marginal utility.

A few comments regarding camera movement when you click somewhere off the terrain:

- left click and drag – If the initial click is not on visible terrain, the camera pivot is instead the center of the map at the lowest possible terrain height: (0; 0; 0).
- mouse wheel up or down – If the mouse is not over the terrain, the camera moves much slower.
- ctrl + left click and drag – If the initial click is not on visible terrain, the mouse drags an equivalent point at height 0.

A middle click (e.g. depressing the mouse wheel) displays a copy of the coordinate axes at the mouse position for as long as the middle button is held down. It performs no other action.

Select Multiple Objects

You can select multiple targets in the main panel as follows:

- Ctrl + left click – Add a terrain block, model, or plant to the current selection. It is not possible to select different types of targets at once. E.g. a model and a plant cannot be in a selection together. Ctrl-clicking a different type of target resets the selection as if it were clicked without the control key. Ctrl-click can also remove a terrain block from the selection. It is not possible to remove a model or plant from the selection in the main panel.
- right click and drag – Select all models within the dragged rectangle. This only works for models.

Bug: It is sometimes possible to right click and drag to select multiple terrain blocks. But it usually only works soon after a map is loaded. At some point during editing, something causes it to stop working, and then it won't work anymore until the map is reloaded.

When multiple objects are selected, the selection outline in the main panel encompasses all of the selected objects.

Manipulate Multiple Objects

Only a single interface widget is placed on the last selected object, but any widget manipulations apply to all selected objects. E.g. you can easily move all selected objects as a group. All movement, rotation, or scaling is performed relative to each object's initial position, orientation, or scale.

If multiple objects are selected, the context menu has **Selected – Action** instead of **Feature – Action**:

Selected – Do Land (for models) works fine.

Selected – Replace (for models) fills me with such fear that I refuse to even try it. (See page 119 for the source of my fear.)

Selected – Fly To moves the camera to bring all selected objects into view (at least partially).

Selected – Delete All works fine.

Selected – Duplicate All duplicates all of the selected features.

Bug: Duplicating multiple features at once may cause an extra one to spawn every time the duplicate warning dialog pops up with no obvious way to make it stop.

Output Panel

The **Output** panel (below the main panel) shows logging information and error messages from the Editor and the embedded display engine.

At the bottom of the **Output** panel are four arrows pointing left and right. I do not know what these do.

The angled tab at the bottom of the **Output** panel indicates that it is showing the main log from the SnowRunner Editor. On occasion, another tool within the Editor may create a separate tab in the **Output** panel in order to display its log. You can then click on the angled tabs to switch between logs.

Terrain Panel

The **Terrain** panel is described pretty extensively in its introduction on page 50, so this section describes only one additional option of marginal utility.

Hold the **Control** key while you left click to add or remove a terrain block from the current selection.

You cannot click and drag or shift-click in the terrain panel to select multiple terrain blocks.

File View Tab

At the bottom of the **Terrain** panel are two tabs which you can use to switch between the **Terrain** panel and the **File View** panel.

The **File View** panel lists selected files and folders in the **Media** directory.

Bug: Files are in alphabetical order, but folders are in reverse alphabetical order.

Double-click any map's top-level XML file in the **prebuild** folder to open that map. Only one map can be open at a time, so if you have a map open already, the Editor prompts you to save it before opening a new one.

Double-click a folder to expand or contract its level of hierarchy.

Right-click the **prebuild** folder or any sub-folder for a context menu that allows you to create a new directory (**New Folder...**) or a new map (**New Terrain...**) in that folder. Note that creating a folder hierarchy for maps is only for your own convenience. All maps are compiled into the top level of the **levels** and **Mods** directories, so they must still be named uniquely across the entire prebuild hierarchy.

Bug: **New Folder...** has no visible effect unless you enable **Show All Files** (below) to see it. You need to add a map (**New Terrain...**) to that folder to make it visible when **Show All Files** is disabled.

Bug: **New Folder...** enforces the same restrictions on a folder name as **New Terrain...** enforces on a map name, but these restrictions are unnecessary for folders. If you want to use a disallowed character in a folder name, create or rename the folder in Windows Explorer instead of within the Editor.

Bug: **New Terrain...** sometimes fails to create certain initial files such as **_snow.tga** or **mud**. While there are various options to recover from this state, the easiest is to simply delete all files associated with the map and start over.

Remember that you can open a recently edited map by selecting it from the File menu, but you can open any map at all by double clicking its XML file under the prebuild directory for its mod.

The **File View** tab has its own toolbar. You can hover over each button to get a description of what it does.



The **Update** button refreshes the file list.

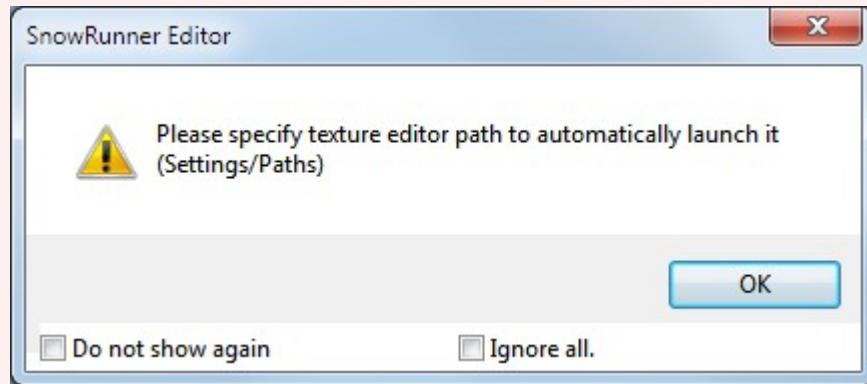
The **Show in Explorer** button opens the selected folder in Windows Explorer.

The **Find References** button finds all references within the **Media** directory to the selected XML file. For example, it can find all maps that use a reference map, or it can find all references to a custom asset. The **Find References** button remains depressed while the search is ongoing, and it releases when the search is complete. The search results are displayed in the **Output** panel in a new tab called **Find References Results**. Be aware that the **X** button in the corner of this tab closes the entire **Output** panel, not just the tab.

The **Show All Files** button is a toggle. When enabled, it shows all files and folders in the Media directory. In the default disabled state, it shows only the top-level XML file for each map, and folders are shown only if they contain at least one top-level XML file.

When all files are shown, double-click an XML file (that isn't a map's top-level XML file) to open it in your default XML editor (e.g. Notepad++).

Bug: When all files are shown, if you double-click a bitmap file, the Editor complains that no texture editor has been configured. This is a holdover from the MudRunner Editor; there is no way to specify the path to a texture editor in the SnowRunner Editor.



Bug: [File View](#) shows an icon next to each file corresponding to the default Windows application that opens that file type. However, double-clicking only starts the associated application for XML files. All other document types are unsupported.

Controls for Hierarchical Lists

The [File View](#) tab, the [Scene View](#) panel, and the property panel all display hierarchical lists that share a number of controls in common.

A few common controls are performed by the mouse:

- Left click item selects that item.
- Left click on a to the left of a folder (in the [File View](#) tab) or hierarchical container (in the [Scene View](#) panel or property panel) expands that container, showing its contents.
- Left click on a to the left of a folder or hierarchical container contracts that container, hiding its contents.
- Double click on a folder (in the [File View](#) tab) or hierarchical container (in the property panel) expands or contracts that container. Double click has no effect in the [Scene View](#) panel.
- Right click performs special actions specific to the panel in question.

Keyboard Controls

Bug: Although there are keys to move the selection highlight, only the property panel has a key to select the highlighted item. This deficiency unfortunately makes keyboard navigation rather useless for the **File View** and **Scene View**.

Certain actions can also be performed using the keyboard:

- The up and down arrows move the selection highlight up and down the list.
- The right arrow does different things in different contexts:
 - On an unexpanded folder or container, it expands the container.
 - On an expanded folder or container, it moves to the first item in the container.
 - On a non-container item in the property panel, it moves the selection down to the next item.
 - On a non-container item in the **File View** tab or **Scene View** panel, it does nothing.
- The left arrow does different things in different contexts:
 - On an expanded folder or container, it hides the items in the container.
 - On an unexpanded folder or container, it moves to the container of the current item (up one level in the hierarchy).
 - On a non-container item in the property panel, it moves the selection highlight up to the previous item.
 - On a non-container item in the **File View** tab or **Scene View** panel, it does nothing.
- The **Home** key moves the selection highlight to the top of the list.
- The **End** key moves the selection highlight to the bottom of the list.
- The **Page Down** key moves the selection highlight down by approximately one page.
- The **Page Up** key moves the selection highlight up by approximately one page.

In all cases, the panel automatically scrolls to keep the highlighted item in view.

Scene View Panel

The **Scene View** panel is described pretty extensively in its introduction on page 53. This section only describes how to select multiple objects at once.

Ctrl + left click on a truck, model, or plant in the same category as previous selections to add that feature to the selection. Features in different categories cannot be selected at the same time.

Ctrl + left click on a selected feature removes it from the selection.

Shift + left click on a feature in the same category as a previous to select all features from the first clicked item to the shift-clicked item. This discards from the selection any features outside the newly selected range.

Bug: Shift click does not correctly highlight the selected items, although the proper items are secretly selected. If you're not sure, shift-click the same item a second time, and all items seem to highlight correctly. If you've used ctrl-click to adjust the selection after the shift-click, it is not possible to tell exactly what is selected.

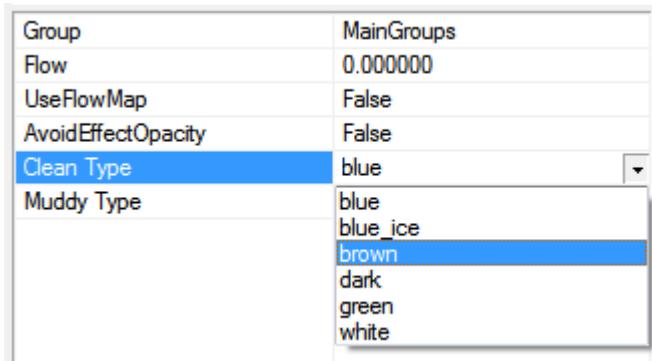
Once multiple objects are selected, they can be manipulated as described on page 259.

Property Panel

The property panel is described pretty extensively in its introduction on page 55, so this section describes only a few additional controls of marginal utility.

If the selection highlight is at the top of a set of related properties (e.g. **Position** or **Rotation** in the figure above), the **Enter** key expands or collapses the set.

When a property with a dropdown menu is selected, you can type the first letter of the desired value to immediately select that value. If the dropdown menu is currently open, the selection highlight moves to that value but doesn't select it yet. In either case, if there are multiple values that start with the same letter, typing the letter cycles among those values. Picking a dropdown value by typing is not sensitive to case, so e.g. you can press **t** for **True** and **f** for **False**.



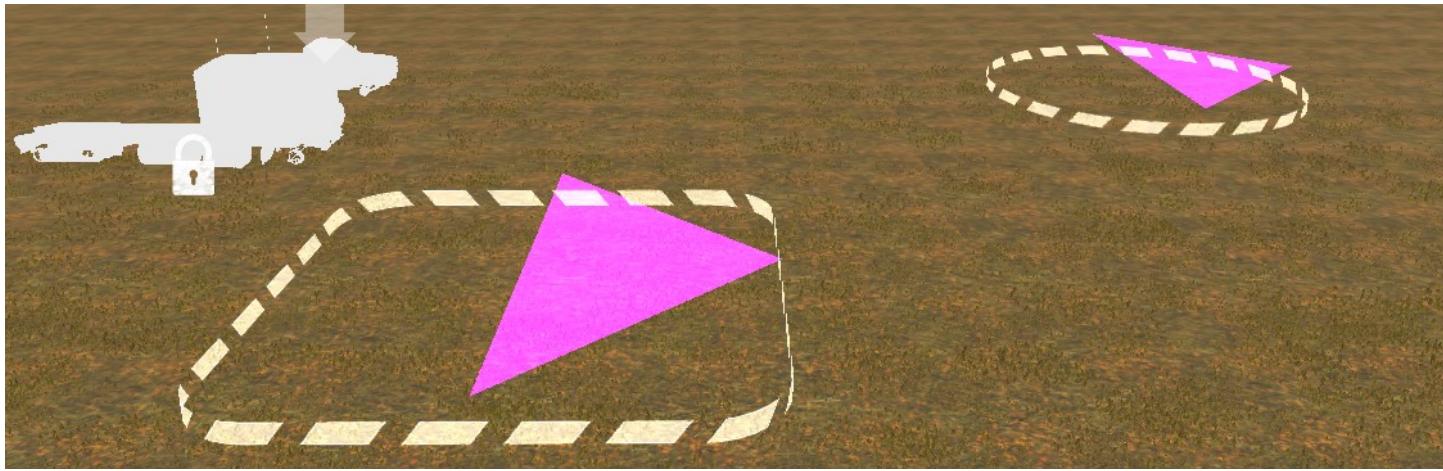
When a property that chooses a filename is selected (i.e. the distribution **Map** property), you can double click it to open the file chooser. This can be faster than finding and clicking on the **...** button after it appears. You have to click once in the right column before a double click can register. However, a double-click in the **left** column opens the file chooser without requiring an extra click.

Zones and Objectives

The following sections describe zones and their various functions. These include static zone types (e.g. “garage”) as well as objectives (tasks, contests, and contracts). Zone types and objectives can be edited at the map level using the Zone Settings Editor or at the region level using the Region Settings Editor.

Zones

Zones define locations on the map that are used for various functions such as truck services or objective destinations. Each zone is surrounded with a dotted line which I call the “perimeter”. In the game, the perimeter is animated to slide around the edges of the zone. In the Editor, the perimeter is static. The Editor also draws a pink triangle to point in the direction of the zone’s orientation. The orientation of the zone makes a difference for certain zone types.



Zones are commonly placed on the map in association with 3-D models such as garages, fuel pumps, etc., but the models are just set dressing. Only the zone has the associated game logic.

The main SnowRunner Editor can only specify the properties of the zone **locator**, i.e. its ID, location, size, and (potential) appearance. These properties are described below.

The **functions** of a zone are edited by the Zone Settings Editor (page 276) or Region Settings Editor (page 367). Permanent zone functions are specified by the zone type (page 295). Zone functions that apply only for the duration of an objective are specified by the objectives (page 313) and the objective stages (page 331).

Zone ID

Scene.(Terrain).Zones.zone_id: Id: text

Specifies a unique identifier for this zone for use by objectives and to act as a label within the Editor.

Default: blank; displays as **no_id** in the Editor and may confuse the game.

Each zone must have an **Id** value that is unique among zones. The ID allows the zone locator to be linked to its functions specified in the Zone Settings Editor or Region Settings Editor. The ID is also used to label the zone in the main panel and **Scene View** of the map Editor.

Another zone in the map should not have the same `Id` value. The game may hang or crash or exhibit strange behavior if it finds a duplicate zone ID within a map, including more than one zone with no ID specified.

Tip: If the game hangs when you bring up the navigation map, it's almost certainly zone related, and a missing or duplicate zone ID is the most common cause.

Another map in the same region may reuse the same zone ID.

For safety, I recommend that the zone ID be limited to a combination of lowercase `a – z`, uppercase `A – Z`, `0 – 9`, and `_`. It should not include other characters, including spaces.

Move or Rotate a Zone

`Scene.(Terrain).Zones.zone_id` : `Org X`, `Org Z`: numeric, in meters

Specifies the location of the center of the zone.

Default: ground position in the center of the main panel at the time of creation.

`Scene.(Terrain).Zones.zone_id` : `Dir`: 2-D direction vector (page 422)

Specifies the orientation of the zone.

Default: (1; 0); facing east.

Move or rotate a zone in the same way as for trucks (page 75) except that zones are always attached to ground, so the movement and rotation widget shows only the ground plane. The coordinate systems for position and orientation are described on page 420.

Bug: The direction vector should be unit length; otherwise the zone may be drawn with a distorted size, and the game may perform the zone's functions using a **different** distorted size.

Bug: The `Local Transform` toolbar option does not work for zones.

Zone Name and Icons

Zone Name

`Scene.(Terrain).Zones.zone_id` : `Name`: UI text (page 282); formatting tags are not supported; max ~30 characters

Specifies the name to display for the zone on the navigation map.

Default: blank; displays the zone ID in the Editor and displays nothing in the game.

Unlike the **Id** property above, the **Name** does not need to be unique.

The **Name** property isn't displayed while the player is driving. E.g. if the player unlocks a garage named "Fred's Garage", the game only displays a generic "Garage Unlocked" message. It is only displayed in the navigation map.

Zone Icons

Scene.(Terrain).Zones.zone_id: Icon 40x40: text

Specifies the name of an icon that is displayed above the zone when the player drives close to it. The same icon is also displayed on the navigation map.

Default: blank; no icon is displayed.

Scene.(Terrain).Zones.zone_id: Icon 30x30: text

Specifies the name of an icon that is displayed in the list of locations on the left side of the navigation map.

Default: blank; no icon is displayed.



Unfortunately, there is not a convenient menu of icon images to choose from in the Editor. Refer to the Zone Icons reference section on page 403 for a table of icons and their corresponding names. Be sure to copy each name exactly as it appears in the table, even if it looks like the name has a typo. If an icon name is specified incorrectly, the game displays a small red and yellow placeholder image (☒) instead.

If a 40×40 icon is used in place of a 30×30 icon or vice versa, it looks generally OK, but it isn't properly centered.

If an icon property value is left blank, then no icon is drawn. However, other cues are present as usual to indicate the presence of the zone. The screenshot below shows two fuel stations on the map, one with associated icons and one without.



Zone Visibility on Navigation Map

`Scene.(Terrain).Zones.zone_id: Is Visible On Minimap: True/False`

Specifies whether the zone is displayed on the navigation map.

Default: `True`; the zone is displayed on the navigation map.

Bug: The SnowRunner Editor and its documentation consistently misuse the term “minimap” when they mean the full-size navigation map. SnowRunner does not have a minimap.

If you change `Is Visible On Minimap` to false, then the zone is never displayed on the navigation map or on the list to the left of the map. In that case, the `Name` property doesn’t matter because it is never used. The zone perimeter still appears during normal driving. However, if `Is Visible On Minimap` is `False`, the icon above the zone is also suppressed.

SnowRunner automatically suppresses any display of the zone in most circumstances where doing so is appropriate (e.g. when a zone is only used for an inactive objective), so there is usually no reason to change `Is Visible On Minimap` from its default value. Any exceptions are noted where appropriate.

Tip: A good reason to suppress a zone on the minimap is when you have two zones with the same purpose in very close proximity (e.g. two sides of a fuel station). But you should also consider simply merging the two zones into one.

For more information on zone visibility, see page 363.

Zone Perimeter

`Scene.(Terrain).Zones.zone_id: Shape Type: pulldown menu to choose Rectangle or Circle`

Specifies whether the zone is displayed as a rectangle (with rounded corners) or a circle.

Default: `Rectangle`.

Scene.(Terrain).Zones.zone_id: Dimensions.Length or diameter: numeric, in meters

Specifies the length of a rectangular zone's sides parallel to the zone's direction, or the diameter of a circular zone.

Default: 10.0.

Scene.(Terrain).Zones.zone_id: Dimensions.Width: numeric, in meters

Specifies the length of a rectangular zone's sides perpendicular to the zone's direction; not used for a circular zone.

Default: 10.0.

Scene.(Terrain).Zones.zone_id: Border.Rounding radius: numeric, in meters

Specifies the radius of a rectangular zone's rounded corners; not used for a circular zone.

Default: 1.5.

Note that the **Shape Type** is only there for convenience. A **Circle** is drawn exactly like a **Rectangle** that has its **Width** equal to its **Length or diameter** and its **Rounding radius** equal to half its **Length or diameter**.

Bug: When the **Shape Type** is **Circle**, the game ignores the **Width** value. However, the Editor still draws the zone's front-facing arrow using the **Width** as its width. For better visuals in the Editor, you might want to set the **Width** equal to the **diameter** for a **Circle** zone.

Perimeter Shape and Size

Scene.(Terrain).Zones.zone_id: Wall.Is wall: True/False

Specifies whether the zone perimeter lies flat against the ground (**False**) or is drawn vertically (**True**).

Default: **False**.

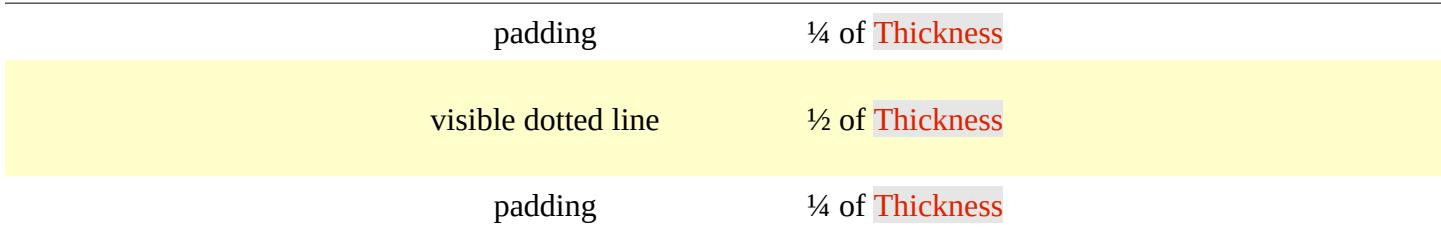


Scene.(Terrain).Zones.zone_id: Border.Thickness: numeric, in meters

Specifies **twice** the width of the visible dotted line that designates the zone perimeter.

Default: 0.75.

The visible dotted line that designates the zone perimeter is conceptually drawn upon a larger invisible surface that I call its “padded ribbon”. The width of this ribbon is the **Thickness** value, in meters. The visible dotted line is half the width of the ribbon, and the quarter width of ribbon on either side of the dotted line is invisible padding. I.e.



Vertical Perimeter

When **Is wall** is **True**, the layout of the perimeter in the top-down view is determined by the **Length or diameter**, **Width**, and **Rounding radius** properties as described above. Independently, the layout of the perimeter in the side view is determined by the **Thickness** and **Height** properties.



The **Rounding radius** isn't constrained by the **Thickness** of the perimeter, so the minimum **Rounding radius** is 0, entirely removing the rounding from the corners. The maximum **Rounding radius** is half the minimum of **Length** and **Width**. If the **Rounding radius** is larger, the rendering engine reduces it as necessary.

Scene.(Terrain).Zones.zone_id: Wall.Height: numeric, in meters

Specifies the height of a vertically drawn zone perimeter above terrain level; not used when **Is wall** is **False**.

Default: 2.0.

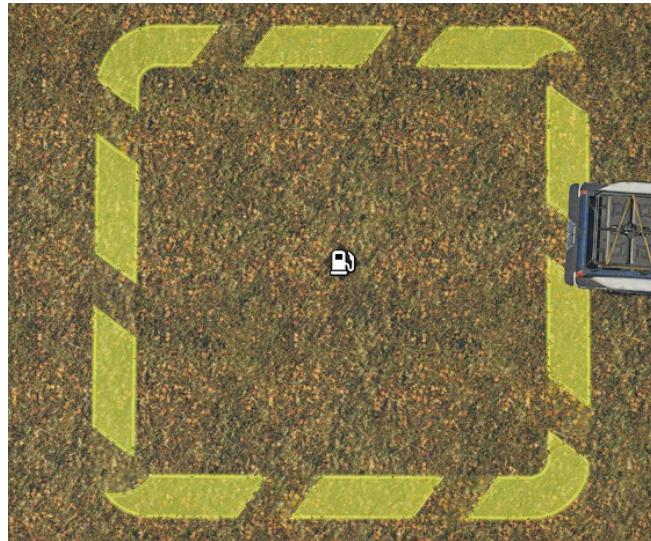
When the **Height** value is 0, the vertical perimeter is centered at ground level. Its height is adjusted by the **Height** value, measured in meters.

Bug: The Editor places the forward-facing pink arrow (visible only in the Editor) at **half** the height given by the **Height** value.

Since the width of the visible dotted line is half the **Thickness** value, the **Height** value must be set to one-quarter of the **Thickness** value to align the bottom of the visible perimeter to exactly ground level.

Flat Perimeter

When **Is wall** is **False**, the layout of the perimeter in the top-down view is determined by a combination of its **Length or diameter**, **Width**, **Rounding radius**, and **Thickness** properties.



Bug: Although the game draws the perimeter either flat or vertical, depending on the **Is wall** property, the Editor always draws the perimeter vertically.

When drawing a perimeter with **Is wall** set to **False**, however, the Editor still ignores the **Height** property and instead draws the bottom of the perimeter ribbon at exactly ground level (so the visible dotted line is slightly above the ground). On the other hand, the Editor still uses the **Height** property to set the height of the forward-facing arrow.

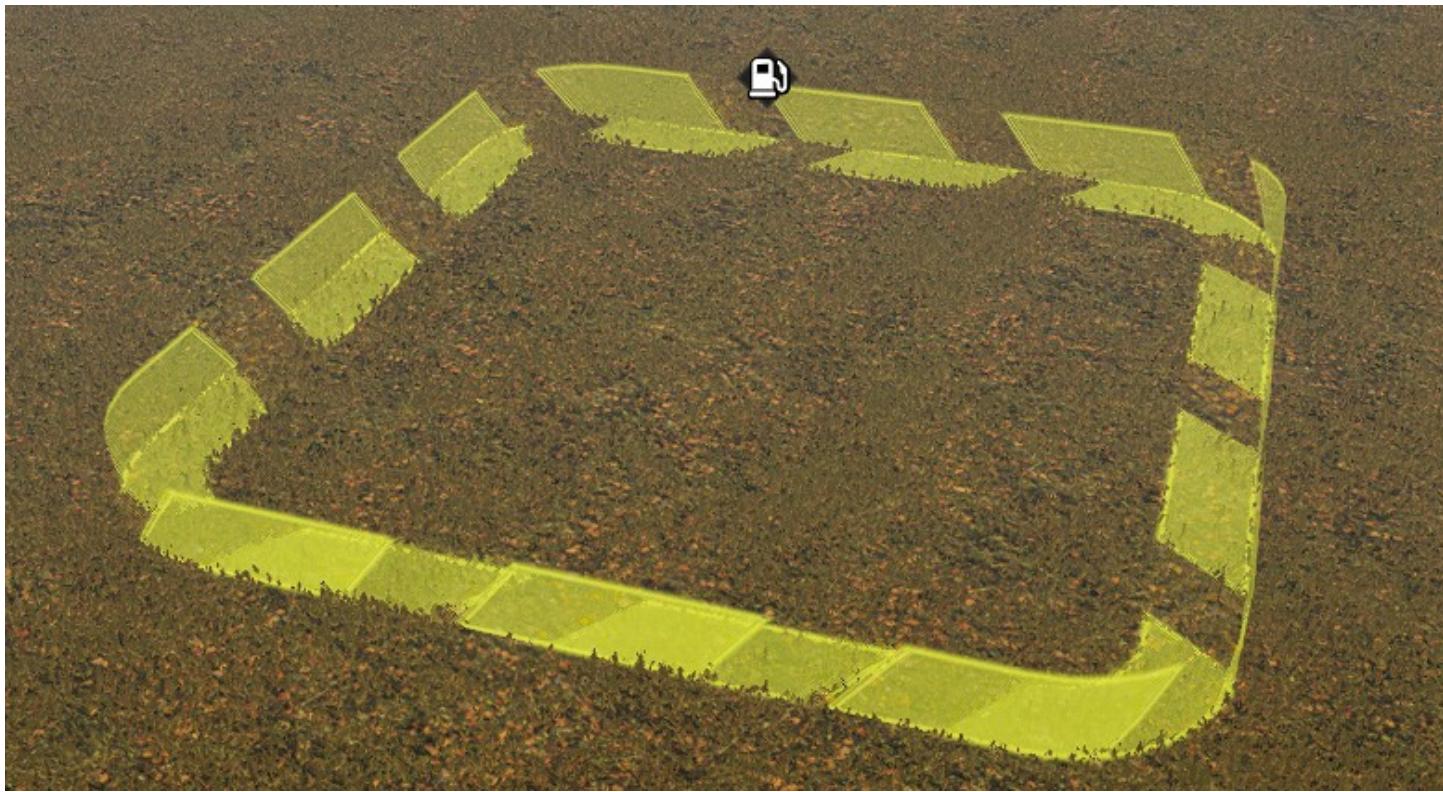
This Editor deficiency makes it much harder to experiment with different property values to achieve the desired look. The calculations below that determine the exact perimeter placement may be useful to you if you don't mind the math.

The perimeter is placed so that the **outer edge** of its **padded ribbon** aligns with the dimensions provided by **Length or diameter** and **Width**. This means that the visible dotted line is inset by half its width.

The **Rounding radius** sets the **outer radius** of the **padded ribbon**. The inner edge of the ribbon necessarily has a smaller radius. To avoid a negative radius, the game automatically reduces the padded ribbon width (**Thickness**) to no greater than the **Rounding radius**. This means that the visible dotted line width is no more than half the **Rounding radius**, and the inner edge of the dotted line is always rounded, never a sharp corner.

The actual detection area for the zone is determined by the outer edge of the padded ribbon, which is exactly set by the **Length or diameter**, **Width**, and **Rounding radius** properties. However, you might prefer to set the outer edge of the visible dotted line to particular dimensions. If so, use the calculations below:

- **Length** = desired visible outer length + $0.5 \times \text{Thickness}$
- **Width** = desired visible outer width + $0.5 \times \text{Thickness}$
- **Rounding radius** = desired visible outer radius + $0.25 \times \text{Thickness}$



Other Perimeter Properties

Scene.(Terrain).Zones.zone_id: Border.Scroll speed and direction: numeric, in clockwise meters per second
Specifies the rotation speed of the zone perimeter.
Default: -1.0.

The rotation is counter-clockwise if the value is negative. If **Is wall** is **False**, the speed is measured at the outer edge of the padded ribbon.

`Scene.(Terrain).Zones.zone_id: Border.Opacity`: numeric

Specifies the brightness of the zone perimeter.

Default: 1.0.

Since the perimeter color is added to the ground color, it is never truly opaque, but larger `Opacity` values make it appear more opaque, and smaller values make it appear more translucent.

Bug: The `Opacity` property only affects how the perimeter looks in the Editor. In the game, the perimeter's color and brightness are determined by the zone's function(s), and the `Opacity` property is ignored.

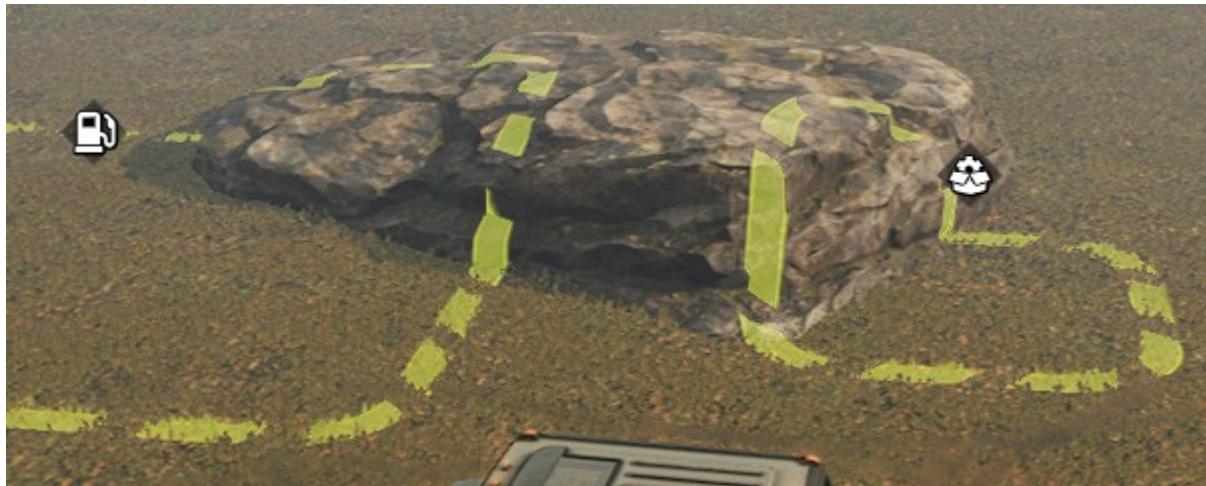
Hilly Zones

If the terrain in a zone is not flat, the zone perimeter gets distorted. If `Is wall` is `False`, the perimeter conforms to the terrain surface. If `Is wall` is `True`, the perimeter follows a straight line from the height at one corner to the height at the next corner (modulo the `Rounding radius`). The below screenshot shows the difference between the two styles. They look goofy when seen juxtaposed like this, but each one individually is fine.



Zone Perimeter on Models

A flat zone perimeter not only conforms to the terrain, but also to most rock models. This is only visible in the game, not the Editor.



The zone perimeter also crawls over the legs of the log trestle model ([log_scavange_02](#)).

Many bridges and platforms also have XML that says the zone perimeter should be placed on the model, but it only works if the model surface is placed at exactly ground level. It's unclear how this is useful for bridges, and it is not at all useful for platforms.



Other miscellaneous models also claim to support a zone perimeter. I have not tested that assertion.

Zone Settings Editor

The Zone Settings Editor allows you to edit various properties that control the gameplay on your map. Most of these are related to zones, some of which allow actions at any time, others of which are tied to specific objectives. Other properties help present the map to the player or set up the starting conditions for the map.

To start the Zone Settings Editor from the main SnowRunner Editor, click the **Zone settings** button on the toolbar (the second button from the right). **Caution:** while the Zone Settings Editor is open, the main SnowRunner Editor window is completely unresponsive. Close the Zone Settings Editor to return to the main SnowRunner Editor.

JSON Files

Unlike the main SnowRunner Editor, which saves its data in XML and bitmap files in the **prebuild** directory, the Zone Settings Editor saves its data in JSON files in the **levels/map_name** directory:

- **level_desc.json** records properties associated with the map introduction (page 288) and images (page 290).
- **content_settings.json** records content settings (page 292).
- **zone_settings.json** records properties associated with zone types (page 295) and a few other miscellaneous properties.
- **objective_settings.json** records properties associated with objectives (page 313).

Manage the Zone ID

The zone locators are defined in the main SnowRunner Editor, but the zone functions are defined in the Zone Settings Editor. The zones are listed in the Zone Settings Editor under **map_name.TERRAIN LOCATORS LIST**.

The loose connection between the zone ID in the main Editor and the properties in the Zone Settings Editor is a conceptual minefield. Although the two Editors attempt to keep your changes in sync, it is easy to lose confidence in what they are doing. I'll do my best to explain so that you can always understand what is going on.

Add or Delete a Zone

The main Editor saves most of its property data in **Media/prebuild/map_name.xml**, and it specifically saves its zone properties in **Media/prebuild/map_name/_zones.xml**. Meanwhile, the Zone Settings Editor saves its zone properties in **Media/levels/map_name/zone_settings.json**.

If you add or delete a zone in the main Editor, it doesn't touch the `zone_settings.json` file directly. Instead, when you start the Zone Settings Editor, the main Editor tells it what zones currently exist. Thus, the Zone Settings Editor allows you to edit the same zones that are present in the main Editor.

Note that if you don't select `Save` in the Zone Settings Editor, no changes are made to the `zone_settings.json` file.

Tip: If you delete a zone in the main Editor the zone and its properties won't be shown in the Zone Settings Editor. However, as long as you don't select `Save` in the Zone Settings Editor, the zone settings can be recovered. Close the Zone Settings Editor (without saving), re-create the zone in the main Editor, restart the Zone Settings Editor, and the zone's properties should be there once more.

It is not possible to add a zone from within the Zone Settings Editor. You can delete a zone from the Zone Settings Editor, but it has no effect on the zone locators in the main Editor, and it will reappear the next time you start the Zone Settings Editor.

Rename a Zone ID in the Main Editor

If you change the `Id` of a zone in the main Editor, the Editor keeps track of this rename. The next time you start the Zone Settings Editor, the main Editor tells it to change the name(s) in the `zone_settings.json` file before it opens the window where you can edit the zone settings.

This is weird, so let me say it again. If you renamed a zone `Id` in the main Editor, the `zone_settings.json` file can change even though you never selected `Save` in the Zone Settings Editor.

This oddity only applies to renamed zone IDs. If there are also new or deleted zones, those still aren't updated in `zone_settings.json` until you select `Save` in the Zone Settings Editor.

Note that if you then close your map in the main Editor without saving, your zone IDs will be inconsistent between the `_zones.xml` file (which was not manually saved) and the `zone_settings.json` file (which was automatically updated).

This inconsistency between files doesn't happen in the other direction, however. If you save your work in the main Editor while it is tracking a renamed zone ID, the main Editor quietly starts the Zone Settings Editor in the background and tells it to change the name in the `zone_settings.json` file so that both files are in sync.

Rename a Zone ID in the Zone Settings Editor

If you change the ID of a zone in the Zone Settings Editor, it does not attempt to make a corresponding change in the main Editor. If you save your change in the Zone Settings Editor, the two files will be out of sync.

After closing the Zone Settings Editor, you can immediately make the same change in the main Editor. The Editor attempts to make a corresponding rename in the `zone_settings.json` file, but since the old ID no longer exists in that file, it is harmless.

After you manually change and save both files in this order, the two are correctly in sync. However, it's easier to rename the zone ID in the main Editor first and let it deal with synchronizing the change.

Clear the Zone ID

If you clear the `Id` of an existing zone, the main Editor tracks this as a rename event. The next time you save your changes in the main Editor or start the `Zone Settings Editor`, you'll get an error that a name is missing.



This is harmless. All of the other zones are presented correctly, including any that have been renamed to valid IDs. Of course, the zone cannot be used for any purpose until an ID has been assigned.

Uncommitted Id Edits

When you type a new name into a zone's `Id` property in the main editor, the change is not committed right away. This uncommitted status can be seen in the `Scene View`, which doesn't update the zone name even when you press return. To commit your edit, left click in the main panel or click something else in the `Scene View`.

If you create a new zone, assign it an `Id`, and then start the Zone Settings Editor while the `Id` change is uncommitted, then the Zone Settings Editor won't know the new zone's ID, and it won't include it in the list of zones.

Bug: If you accidentally do this, then after you close the Zone Settings Editor, the main Editor no longer has the zone selected, but it still hasn't committed the `Id` change. Select the zone again, then click elsewhere to deselect it, and it should finally be committed. You can now restart the Zone Settings Editor to see and edit the new zone.

Things are worse if you have an existing zone in the `zone_settings.json` file, and you rename its `Id` in the main Editor without committing the change. If you then start the Zone Settings Editor, the main Editor tells it to

rename the zone according to the uncommitted change, but the main Editor **also** says that the current list of zones includes the zone with its old ID.

Bug: Oh yeah, that's a bug.

As an example, assume you have an uncommitted ID change from `x` to `y`, and you start the Zone Settings Editor. the ID is changed in `zone_settings.json` file from `x` to `y`. However, the Zone Settings Editor is also told that the list of zones includes `x` and not `y`. Since the Zone Settings Editor no longer has any properties for zone ID `x`, it lists `x` without any properties. If you select `Save` now, you'll lose your properties for the zone. If you realize the situation, close the Zone Settings Editor without saving. Select the zone again the main Editor, then click elsewhere to commit the change, then restart the Zone Settings Editor.

Tip: Make a habit of clicking elsewhere to commit editing changes before starting the Zone Settings Editor.

Duplicate IDs

Now that you have some understanding of how ID changes are synchronized, it is apparent how confusing it is to ever have two zones with the same ID.

Bug: If you change the `Id` of a zone multiple times in the main Editor, the Editor tracks this as a series of renames, not as a single change from the original ID to the newest ID. If any of the intermediate IDs were a duplicate of another zone's ID, this causes problems even though the Zone Settings Editor was never invoked while duplicate names were present.

Of course, if you choose `Duplicate` on a zone, then the new zone has the same ID. You could avoid using `Duplicate`, but it can be convenient for duplicating zone locator properties or aligning zones near each other.

If you use `Duplicate` to make one zone into multiple, whichever one you rename first ends up with the original zone's properties. Although the main Editor presents the other renames to the Zone Settings Editor, there is no longer a zone with that ID to rename.

Tip: You can actually take advantage of the above bug. After duplicating a zone, rename the original zone, then rename the duplicates, then rename the original zone back to its original name. When you start the Zone Settings Editor, the original zone has its original name **and** its original properties.

Zones Used by Objectives

Some properties under the `objectiveSettings` category in the Zone Settings Editor take a zone ID as a value. However, the Editors make no attempt to keep these IDs in sync as you make changes. If you change the zone ID in one place, it's up to you to also change it in the other place.

Zone Settings Editor Controls

The hierarchical list in the Zone Settings Editor is similar to the ones in the main Editor, but some of the controls are different. The complete list of confusing, context-sensitive Zone Editor Controls is below. Controls that are useful in all contexts are **highlighted**.

Some mouse controls can be performed with either left or right click:

- Click on a **+** to the left of a hierarchical container to expand that container, showing its contents.
- Click on a **-** to the left of a hierarchical container to contract that container, hiding its contents.
- Double click on **editable text** to edit it. Property values are generally editable, as are zone IDs and **ZoneProperty** names.
- Double click on a non-editable container to expand or contract a container.

Others can only be performed with a left click:

- Left click on a green plus **+** to the right of a property to add a sub-property within it.
- Left click on a red minus **-** to the right of a property to delete it and all of its contents.

Certain actions can also be performed using the keyboard:

- The up and down arrow keys move the selection up and down the list.
- The right arrow key does different things in different contexts:
 - On an unexpanded folder or container, it expands the container.
 - On an expanded folder or container, it moves to the first item in the container.
 - On a non-container item, it does nothing.
- The left arrow key does different things in different contexts:
 - On an expanded folder or container, it hides the items in the container.
 - On an unexpanded folder or container, it moves to the container of the current item (up one level in the hierarchy).
 - On a non-container item, it does nothing.
- F2 edits the selected item. This can only edit its name if the item was selected via the keyboard. It can edit the value if you selected the item by clicking in the value field.
- The **Home** key moves the selection to the top of the list.
- The **End** key moves the selection to the bottom of the list.
- The **Page Down** key moves the selection down by approximately one page.

- The **Page Up** key moves the selection up by approximately one page.
- **Ctrl+S** saves changes.

Bug: A change isn't saved if it hasn't been committed. If the cursor is still visible in a text field, the edits to that field aren't yet committed. This is particularly devilish after you've selected something from a dropdown menu; you feel like you're done, but the field is actually still open for edits. Commit your changes by pressing **Enter** or selecting another item before saving.

- **Ctrl+Q** exits the Zone Settings Editor.

Bug: If the only change hasn't yet been committed, the Zone Settings Editor won't prompt you to save before quitting.

In all cases, the panel automatically scrolls to keep the selected item in view.

Zone Settings Editor Properties

Unlike the main Editor, where only features are added to the **Scene View** list via the context menu, the property list in the Zone Settings Editor is much more dynamic.

If a green plus **+** is to the right of an entry, then you can click the **+** button to add one or more items within that entry. Depending on the property type, a numbered item may be added immediately, or a dialog box may request the new item name or type.

If a numbered item is added automatically, then the entry is just a normal container that requires no additional documentation. However, if the new item requires a name or type, the entry is documented in the following format.

hierarchy.example entry: **+**

Specifies the name and/or type of a new item.

In most cases, the new item really needs only a name or only a type, but the dialog box requests both. If the dropdown menu lists only one type, then it must be the name that needs to be specified. If the dropdown menu lists multiple types, then most likely the name is unimportant. Selecting a type causes the name to update to match that type. In a few rare cases it is useful to add multiple items of the same type. In this case, they must be given distinct names only to keep them distinct in the property list. The name is not otherwise used.

When items are added within an entry with **+**, each added item has a red minus **-** to the right of it. Click the **-** to remove the item from the entry.

If a black arrow ▼ is to the right of an entry, then you can click the ▼ button to activate the entry, which then opens with appropriate subproperties. Click the ▼ to deactivate the entry and remove its subproperties. A dialog box pops up to confirm the activated entry type or **null** for a disabled entry. An entry of this type is just a container that requires no additional documentation; only its contained item is documented.

Once you've drilled down sufficiently within the expandable entries, the actual properties are documented in the same way as in the main Editor (page 54).

UI Text

I use the term “UI text” to refer to most text that the game shows to the player, such as a zone name or objective information. The Zone Settings Editor and Region Settings Editor have many properties with UI text values. The main Editor has only a couple such properties.

UI text can include almost any characters, including Unicode characters. (For the truly esoteric characters, you'll need to test to verify whether they are supported by the game's font.)

Special whitespace characters such as a tab or linefeed won't display properly, but normal spaces are fine.

Although the double quote " and backslash \ have special meaning in the **objective_settings.json** file, you don't need to worry about them when using the Zone Settings Editor. The Zone Settings Editor escapes them appropriately.

SnowRunner uses the square brackets [and] and the vertical bar | for its own purposes, so they should never be used in UI text. Using them risks screwing up the message displayed to the player.

Character Limit

Most UI text fields have a fixed number of lines (often just 1) with a fixed width for each line. A different number of characters may fit in a UI text field depending on the widths of the characters and the positions of line breaks between words.

If a UI text value has more text than will fit in a field, the game cuts off the extra text. It usually does this after a word boundary, but some UI text fields can be cut off in the middle of a character.

For each UI text property, I've estimated the approximate number of characters that can fit. If the UI text is used in multiple places with different amounts of space, I've based my estimate on the smallest available field that the player will commonly see. (I.e. I ignore the abnormally small UI text fields on the player profile screen.) My estimate should help you decide whether your text will obviously fit or obviously not fit, but if your text is near the estimated character limit, you'll just have to try it to see if it fits.

Formatting Tags

SnowRunner allows certain HTML-style tags to format some types of UI text.

Tip: Saber has not documented these formatting tags, and the only tag that Saber uses in the main campaign is `<y>`. I would guess that all formatting tags are safe to use where I indicate that they supported, but you should pick your own risk tolerance.

The portion of text to format is surrounded by an opening tag and a closing tag. E.g. if a goal description is `Deliver to the <y>Swamp</y>`, then it is formatted as below:



The formatting tags are as follows:

- `<y>` starts yellow text, and `</y>` ends it.
- `<r>` starts red text, and `</r>` ends it.
- `` starts blue text, and `` ends it.
- `<g>` starts gray text, and `</g>` ends it.
- `<i>` starts italic text, and `</i>` ends it.
- `<u>` starts underlined text, and `</u>` ends it.

Tags can be nested, e.g. to get yellow italic underlined text. In that case, tags must be closed in the reverse order of how they were opened. E.g. `<y><i><u>SHOUTING</u></i></y>`. Unrecognized tags are quietly suppressed.

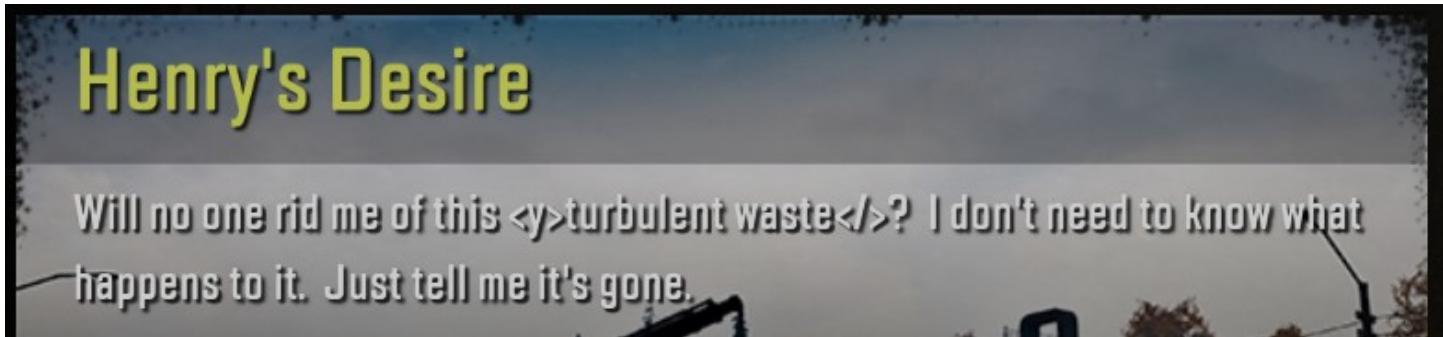
Bug: If the UI text includes a `<` without a following `>`, the game crashes when it tries to display the text. This may happen even when formatting tags are not (fully) supported in that text type.

The following special characters are also supported when formatting tags are supported:

- `<` is replaced with `<`.
- `>` is replaced with `>`.
- `&` is replaced with `&`.

I tried a bunch of other common HTML codes, and they aren't supported, so these may be the only supported ones. `&` doesn't generally need to be escaped; if the game encounters an ampersand code that it doesn't understand, it just displays it unchanged.

This book explicitly states which UI text properties support these formatting tags. If nothing is stated, then tags are implicitly not supported. E.g. in the objective description, tags have **no effect** on the offer text, but they do **properly format** the text in the objective details, but then the **raw tags** are displayed in the completed task in the player's profile. So I consider tags to be unsupported in the objective description.



Localization

Any UI text can be localized into different languages. By default, a property value simply specifies the text to display. However, the property value can instead be an ID that is cross-referenced into a file that is different for each supported language. For example, the property value FREDS_GARAGE can display as “Fred’s Garage” for someone playing in English or “Гараж Фреда” for someone playing in Russian.

To support localization, a file for each desired language must be placed in the [Media/prebuild/map_name/texts](#) directory. If you don’t have this directory yet, you’ll have to create it. The filenames for each language are as follows:

```
strings_brazilian_portuguese.str  
strings_chinese_simplified.str  
strings_chinese_traditional.str  
strings_czech.str  
strings_english.str  
strings_french.str  
strings_german.str  
strings_italian.str  
strings_japanese.str  
strings_korean.str  
strings_polish.str  
strings_russian.str  
strings_spanish.str
```

These files must use the little-endian UTF-16 encoding. The easiest way to ensure this encoding is to copy and modify an existing file. Windows Notepad or Emacs can read little-endian UTF-16 and will automatically write the file back out in the same format. To get an example file to start with, choose [File → Create default truck mod](#) in the Editor, then copy one of the files from [Media/Mods/truck_name/texts](#).

Each line of a localization file is an ID followed by the localized text enclosed in double-quotes, e.g.

FREDS_GARAGE	"Fred's Garage"
--------------	-----------------

For safety, I recommend that the localization ID be limited to a combination of lowercase **a – z**, uppercase **A – Z**, **0 – 9**, and **_**. It should not include other characters, including spaces. The localization ID appears to be case insensitive, but I recommend using the same case everywhere.

The separator between the ID and the localized text can be any mix of spaces and tabs, although Saber recommends tabs only.

The localized text can be almost any Unicode text, but with the following restrictions. If you want to use a standard double-quote **"** in the quoted localized text, escape it with a backslash, i.e. **\"**. A backslash should also be escaped with another backslash, i.e. **\\"**.

As with non-localized text, special whitespace characters such as a tab or linefeed won't display properly, but normal spaces are fine. SnowRunner uses the square brackets **[** and **]** and the vertical bar **|** for its own purposes, so they should not be used here, even with a backslash escape. If this book says that formatting tags and/or **\n** are supported in the text, then they are equally supported in the localization. The left angle bracket **<** should not be used except as part of a supported formatting tag.

Bug: If the UI text includes a **<** without a following **>**, the game crashes when it tries to display the text. This may happen even when formatting tags are not (fully) supported in that text type.

If a UI text ID isn't found in the appropriate localization file, the ID itself is displayed. Thus, you can always simply specify the desired text directly in the property value rather than using a localized ID.

Bug: If SnowRunner can't find the desired localization for a particular ID, it displays the ID itself even if a localization exists in English or another language.

Tip: If you've localized your map into only a subset of languages, the best option for the other languages may be to simply copy over the English file. It's a bit hegemonic, but it's better than showing the player your internal IDs.

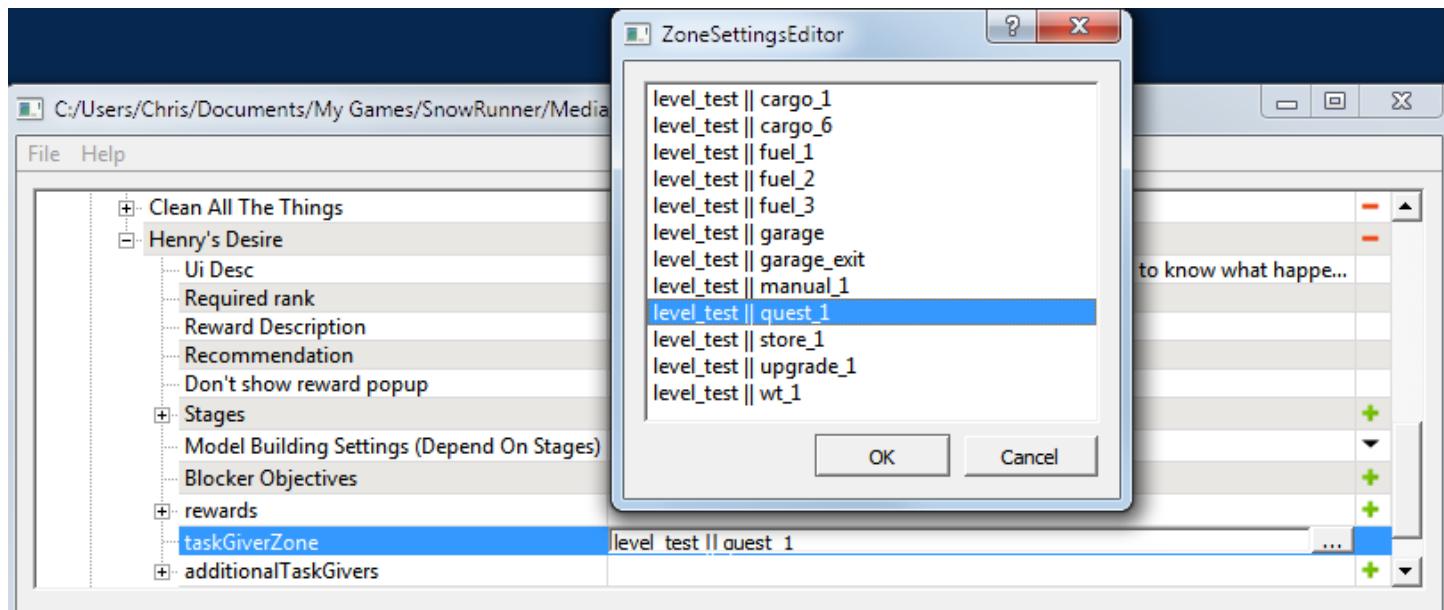
Global or Regular Zone ID

Objectives can cross maps within a region. As such, certain properties require a global zone ID in which both the map name and the zone ID are specified. The map name is separated from the following zone ID by two vertical pipe characters, e.g. **map_name || zone_id**.

Tip: When using the Zone Settings Editor to edit a map, all global IDs must refer to that map. They cannot refer to another map, even if the maps are always used together in a region. Only global IDs entered at the region level using the Region Settings Editor (page 367) can cross map boundaries.

If the map is implicit, only a regular zone ID is needed. In this case, the map name should **not** be attached to the zone ID.

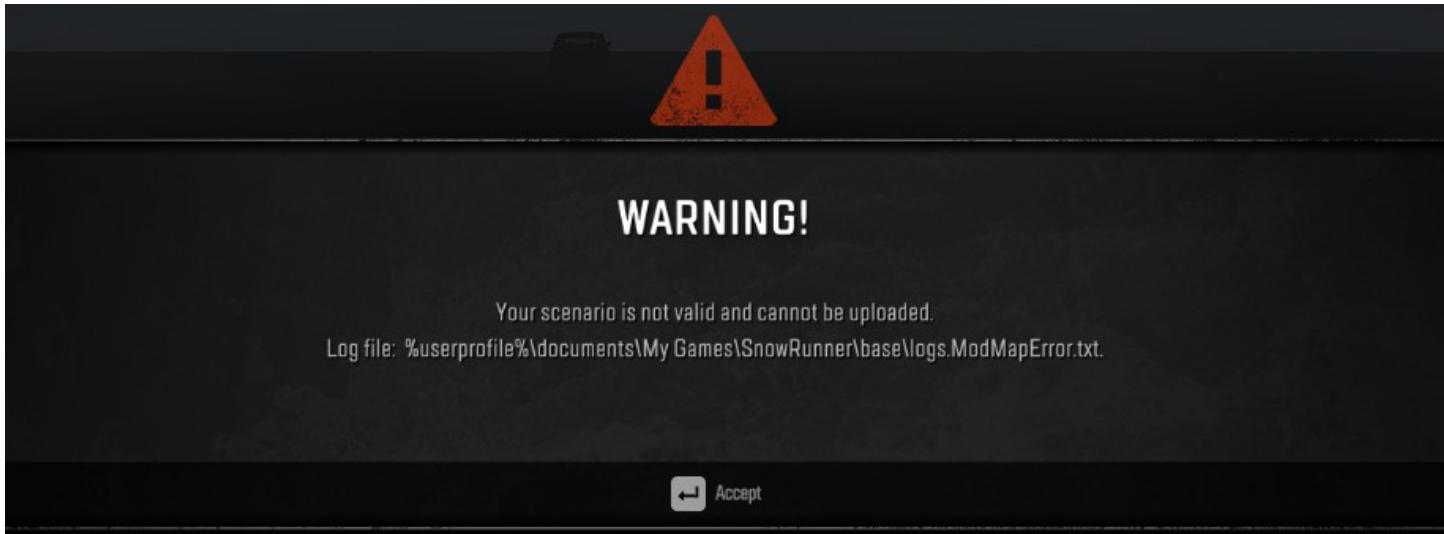
For either case, when you select the zone ID value for editing, a small button labeled appears to the right. Click the button to open a dialog listing the known global or regular zone IDs, as appropriate. From this dialog, you can simply select the desired zone without worrying about its format.



Bug: A few properties in the Zone Settings Editor require a zone ID but do not provide a button to allow easy entry of the zone ID in the appropriate format. All such properties require only a regular zone ID, which you must type in by hand.

ModMapError

The game has a limited ability to detect errors in the zone and objective specifications. When it detects such an error, it displays an error message when it loads the map.



The error message gives the name of a log file that you can look at for more information. The log file is only cleared when you restart the game, so pay attention to the time stamps in the log. For example, the below log shows one error detected at 4:14 pm and two more detected at 4:20 pm:

```
16:16:04.003 2777.794 msg    scripts/modMapError           Invalid level  [] : contract
visit 1 and 2
16:20:45.394 3059.185 msg    scripts/modMapError           visit 1 and 2: More than one
map for contract : \ maps: mod_level_map2, mod_level_map1 \ (in zone to visit and
makeActionInZone zones list)
16:20:45.394 3059.185 msg    scripts/modMapError           Invalid level  [] : contract
visit 1 and 2
```

The log messages can be a bit inscrutable, but you can usually make sense of them when you consider what changes you've made to your zone types and objectives recently.

After displaying the error message, the game does load and start your map, but all zones and objectives are disabled.

Map Initialization

The Zone Settings Editor includes a number of miscellaneous properties that apply to the map as a whole. These properties generally describe how the map should be initialized and how it fits into its region, but there isn't really a coherent difference between these properties and the map properties that are specified within the main Editor (page 251).

Map Introduction

When the player enters a fresh map, an optional window introduces the map. The appearance and behavior of this window are described by the properties below.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris bibendum congue mauris, nec euismod lectus dictum eu. Suspendisse in elit nulla. Aliquam ac congue ipsum. Sed turpis ligula, porta sit amet mi sed, venenatis molestie est. Integer est libero, facilisis eget ipsum sit amet, condimentum volutpat enim. Duis eu risus elementum, finibus arcu vitae, lobortis justo. Curabitur sed ante non orci fringilla consequat. Vestibulum porta sollicitudin ultricies. Duis gravida orci molestie, posuere nunc eu, auctor dui. Suspendisse non molestie justo. Vestibulum hendrerit tristique nisi sit amet consequat. Donec pharetra felis quis nibh pharetra laoreet. Sed justo nulla, imperdiet at ex non, fermentum commodo lorem. Cras quis suscipit dolor, eget auctor elit. Quisque justo orci, vulputate posuere vulputate at, elementum eget magna. Mauris sed viverra enim, eget vulputate diam. Vivamus vel lectus at leo tristique rhoncus. Aenean a justo at nunc vulputate tristique. Donec ut ex sed tellus tempus auctor. Sed egestas malesuada erat, nec malesuada turpis venenatis a. Nam dignissim ligula congue dui ultricies, quis ornare orci bibendum. Curabitur in luctus massa, in iaculis nulla. Sed sapien.

uiTitle

popupSettings.uiTitle: UI text (page 282); formatting tags are not supported; max ~72 characters

Specifies the title to use at the top of the introduction window.

Default: blank.

In order for the introductory window to display, **uiTitle** must be specified as well as at least one of **uiText** or **uiIcon**. If **uiTitle** is blank or if both **uiText** and **uiIcon** are blank, the window does not appear.

uiText

popupSettings.uiText: UI text (page 282); supports a special set of formatting tags, described below

Specifies a description of the map to use at the bottom of the introduction window.

Default: blank; no text is displayed.

UI text and the standard formatting tags are described on page 282. **popupSettings.uiText** supports all of the standard formatting tags except **<i>**. In addition, **popupSettings.uiText** supports **\n** or **
** as a line break. It also supports bulleted lines surrounded by **** and ****.

When **** and **** are used for bulleted lists in HTML, the complete list is supposed to be surrounded by **** and ****. However, the one example in Saber's texts uses **** and **** instead. I find that either combination works, as well as leaving off surrounding tags altogether. Use whatever you feel most comfortable with.

The space allocated to **uiText** is a fixed size of eight lines, regardless of how much text there is. Each line holds about 150 characters. If there is more **uiText** than can fit in 8 lines, the remaining text is cut off.

uiIcon

popupSettings.uiIcon: text

Specifies the image to use in the center of the introduction window.

Default: blank; no image is displayed.

The image should be 1188×326 and should be saved as a PNG file in
prebuild/map_name/ui/textures/image_name.png

Set the **uiIcon** value to the **image_name** without the path or the **.png** extension.

if the image is too big or too small, it overfills or underfills the window and is incorrectly centered.

If the image is not found with the specified name, a filler image is displayed instead. (Weirdly, **uiIcon** is the opposite of other properties in using the filler image in place of a broken image, and not in place of an unspecified image.)

Automatically Activated Objectives

popupSettings.objectives.[n]: text

Specifies the name of an objective to activate as the map is initialized.

Default: blank; ignored.

Any number of objectives can be activated when the map is entered, but **only** if the introductory window displays as described for `uiTitle`, above. If the introductory window isn't displayed, the game doesn't activate any objectives.

The objectives to be activated can be contracts, tasks, or contests:

- A contract is what is typically activated via this property. The player can activate a contract any time that it is being offered, but this saves her the step.
- If a task is activated this way, its offer zone is only useful for restarting the task.
- Activating a contest this way is a bit rude to the player since the timer starts right away without any warning. The player can still restart the contest in its offer zone.

The game also begins tracking the first activated objective, specified in `popupSettings.objectives[0]`.

Note that this feature can activate an objective even if that objective depends on a blocker (page 315). You may find this useful for testing.

Map Images

You can specify images to use to represent the map in the global view and in a save slot.

Unfortunately, there is no known method to specify an image to use on a loading screen. The game always uses a generic SnowRunner screen for a custom map.

Map Image in Global View

`levelDesc.globalMapPreview`: text

Specifies the image that represents the map in the global view.

Default: blank; a default image is used.

The image should be 360×216 and should be saved as a PNG file in

`prebuild/map_name/ui/textures/image_name.png`

Set the `globalMapPreview` value to the `image_name` without the `.png` extension.

If the image is too big, only the upper left portion of the image is shown. If the image is too small, then black borders are added to the right and below the image.

If the image is not found with the specified name, a black rectangle is displayed instead.

Map Image in Save Slot

`popupSettings.saveSlotMapPreview`: text

Specifies the image to use in a save slot when the active truck is on the map.

Default: blank; a default image is used.

The image should be 360×203 and should be saved as a PNG file in

`prebuild/map_name/ui/textures/image_name.png`

Set the `saveSlotMapPreview` value to the `image_name` without the .png extension.

If the image is too big or too small, it overfills or underfills the save slot window and is incorrectly centered.

If the image is not found with the specified name, a default image is used instead.

Tip: Because the game fades the bottom of the image to black, a 360×216 image works just as well, which means that you can reuse the `globalMapPreview` image. However, you're taking a chance that the game may change and break your image.

Dev Tools Menu

You can choose whether the Dev Tools menu (page 39) is shown or hidden when the map is published.

Normally it would be hidden, but you can choose to show it if your map is intended as a “proving grounds” type map.

`isEnableDevMenu`: checkbox

If checked, the dev tools menu is shown when playing the published map.

Default: unchecked; the dev tools menu is hidden in the published map.

The Dev Tools menu is always shown for a map in development. See page 382 for information about publishing your map.

Ignore Country in Garage Store

`isIgnoreCountryPurchaseBlock`: checkbox

If checked, the player can buy trucks from any country in the garage.

Default: unchecked; the player can only buy trucks from a single country in the garage.

Bug: There is no way to specify what country a map is associated with. If `isIgnoreCountryPurchaseBlock` is unchecked, the player can only buy US trucks in the garage. This checkbox is more useful at the regional level.

Trailers are not associated with a particular country, so all trailers are available from a trailer store.

Map Cloaking

`map_name.isNeedToBeCloaked`: checkbox

If checked, the navigation map starts completely black (except for the regions revealed by the active truck).

Default: unchecked; the navigation map starts dimly revealed in gray everywhere.

The process of revealing the map is described on page 44.

Zone Types

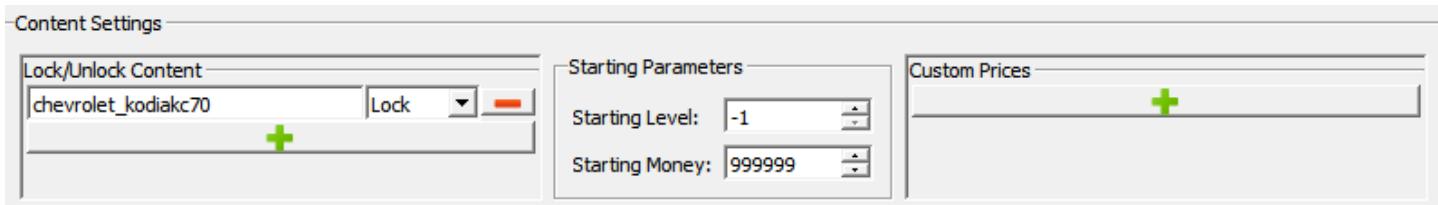
`map_name.TERRAIN LOCATOR LIST` lists each of the zone locators that were set up in the main Editor. Zone types can be assigned to each zone as described on page 295.

Objectives

`objectiveSettings` is used to set up objectives: contracts, tasks, and contests. These settings are described starting on page 313.

Content Settings

The Zone Settings Editor has a few extra widgets in the middle of the window for adjusting the availability of vehicles and add-ons.



Bug: The Zone Settings Editor won't warn you if there are unsaved changes in these widgets. Make sure to save your changes.

Lock/Unlock Content

By default, vehicles and add-ons are unlocked when the player reaches a certain rank, after which they can be purchased in the garage (trucks and add-ons) or the trailer store (trailers). But if the player discovers a locked vehicle on the map (page 89), she can then purchase its type in the garage or trailer store even if its rank requirement hasn't been met. Similarly, if an add-on is awarded to the player, she can purchase more of that add-on without needing to meet the rank requirement.

The whole rank system can be disabled for an item, however, effectively giving a vehicle or add-on an infinite rank requirement. Do this by clicking the **+** below **Lock/Unlock Content** and typing the truck type or add-on name into the blank field. When content is locked in this way, the player can only unlock it by discovering a locked vehicle (page 44) or by being awarded an add-on at an upgrade zone (page 302). If the player doesn't do this, or if the vehicle or add-on isn't made available for discovery or reward on the map, then it remains locked forever.

You can change the **Lock** field to **Unlock**. This reverts the item to its usual rank requirement. This is equivalent to removing the item from the list, but is easier to reverse (e.g. for testing purposes).

Wheels, rims, tires, and stickers cannot be locked. These items always unlock according to their built-in prerequisites. The various **saddle_low** and **saddle_high** addons clearly also aren't intended to be locked or upgraded. A **saddle_low** simply cannot be locked at all. A **saddle_high** can be locked, but it is still supplied for free when a trailer that requires it is purchased from the trailer store.

There is no way to lock all content by default. Even if you listed every item in the game, the player may install a new DLC or mod, which is then unlocked by default. So you have to rely on the rank system and the economy and the player's own sense of fair play to keep overpowered content out of her hands.

Tip: The real purpose of locked content seems to be to avoid a situation where a discovery or reward is anti-climactic because the player has already purchased the item. So considering locking every vehicle and add-on that the player can discover or be awarded.

Bug: If an item is locked, but there is nowhere on the map to unlock it, the game displays a garbled message to the player. Perhaps this means that you should only lock items that the player can unlock somewhere.



Starting Parameters

You can specify the player's starting rank (**Starting Level**) and savings (**Starting Money**). The initial values of -1 set the player's rank and money to the default values of 1 and 5000, respectively.

Custom Prices

The system for custom prices works similarly to the system for locked content (above), but instead of locking a vehicle or add-on, it sets its price. The minimum price is 0, and it applies to both buying and selling. (Custom maps cannot use hard mode, so the buy price and sell price are always the same.)

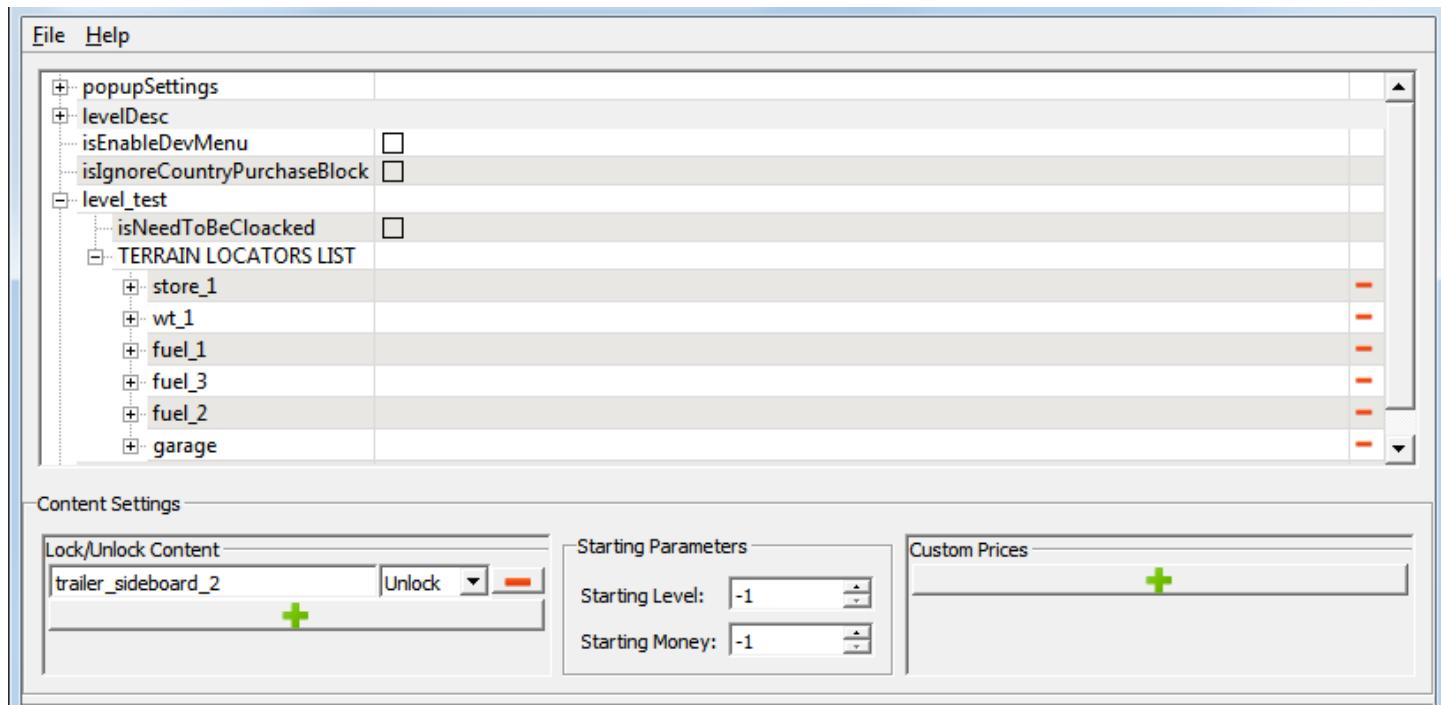
Note that for trucks, the custom price is the **base** price. The garage store price includes the prices of all default add-ons, which can significantly increase a truck's price above its base price.

The prices of wheels, rims, and tires cannot be changed, and stickers are always free. Any price on these items is ignored. A price can be put on the various **saddle_low** and **saddle_high** add-ons in the garage, but they are still supplied for free when a trailer that requires one is purchased from the trailer store.

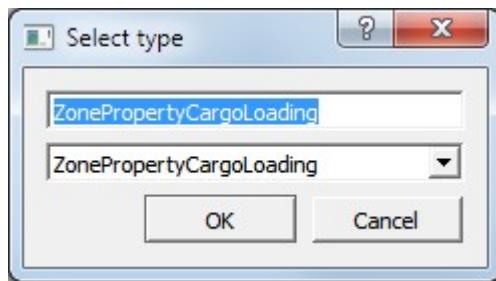
Zone Types

A zone's types specify the services the zone provides to the player's trucks. Most zones have no more than one zone type, but more are allowed. Zone types are specified in the Zone Settings Editor within the `map_name.TERRAIN LOCATORS LIST` property category.

The main SnowRunner Editor specifies the properties of the zone locator (page 266). The **TERRAIN LOCATORS LIST** in the Zone Settings Editor is automatically populated with the ID of each zone locator. If a zone does not have an ID, it is not listed.



To add properties to a zone, open its container, then click the green plus to the right of `props`. Then select the desired zone type from the dropdown menu in the dialog box and click **OK**.



Confusingly, the dialog box has a field that you can type in and a dropdown menu, and selecting an item from the dropdown menu updates both fields. However, after you've chosen an item from the dropdown menu, you can optionally type a different name in the upper field if you so desire.

`map_name.TERRAIN LOCATORS LIST.zone_id.props:` +

Specifies a zone type to assign to the zone and optionally assigns a name to that type. The name has no purpose except for display in the Zone Settings Editor.

Tip: If you can remember the meaning of each `ZoneProperty` type, there is no particular reason to assign a different name after choosing the type. If you find the `ZoneProperty` types confusing, however, you might want to assign a better English name after choosing a type (e.g. “trailer store” instead of `ZonePropertyTrailerAttach`).

The sections below describe the various zone types and the corresponding property name. Depending on the zone type, the Zone Settings Editor may make a number of subproperties available.

Objective-related zones do not need to have an assigned zone type. Objectives are specified and assigned to zones via a separate system, described beginning on page 313.

This section primarily describes how to assign zone types when the map is initialized, using `map_name.TERRAIN LOCATORS LIST.zone_id.props`. However, zone types can also be assigned when an objective is completed, using `objectiveSettings.typeSettings.objective name.rewards`. `ObjectiveRewardOpenZoneProperties.zoneSettings.ZonePropertiesSettingsObjectiveReward.props` (page 320).

Garage Entrance

A zone with the `ZonePropertyGarageEntrance` type allows the player to enter a garage. The orientation of the zone only matters if the zone is also used as a garage exit.

Each map may contain only one garage, and Saber says that a map can contain only one garage entrance. However, my testing shows that a map can contain multiple garage entrances, although they all enter the same garage. Also, while the player is in the garage, the building list next to the navigation map shows the player icon in both of the garage entrances. This is confusing enough to the player that you should probably avoid it.



If a separate garage exit is not specified, then the garage entrance also acts as the garage exit. In this case, the zone’s orientation is important, as described in the following section.

Garage Exit

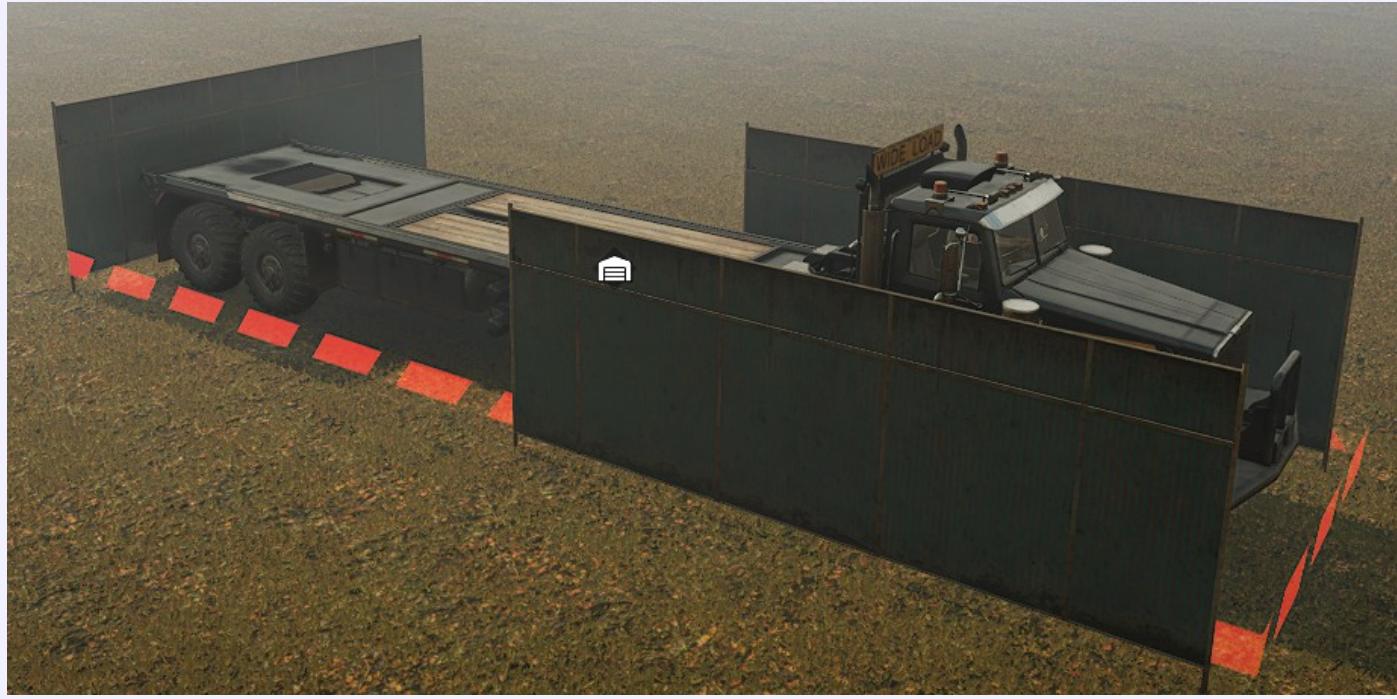
When the player exits the garage, the truck is placed in a zone with the [ZonePropertyGarageExit](#) type. If there is no such zone, then the truck is placed in a zone with the [ZonePropertyGarageEntrance](#) type instead.

A [ZonePropertyGarageExit](#) zone has a red perimeter. It is automatically suppressed from appearing on the navigation map or building list, regardless of the [Is Visible On MiniMap](#) property value specified for the zone locator.

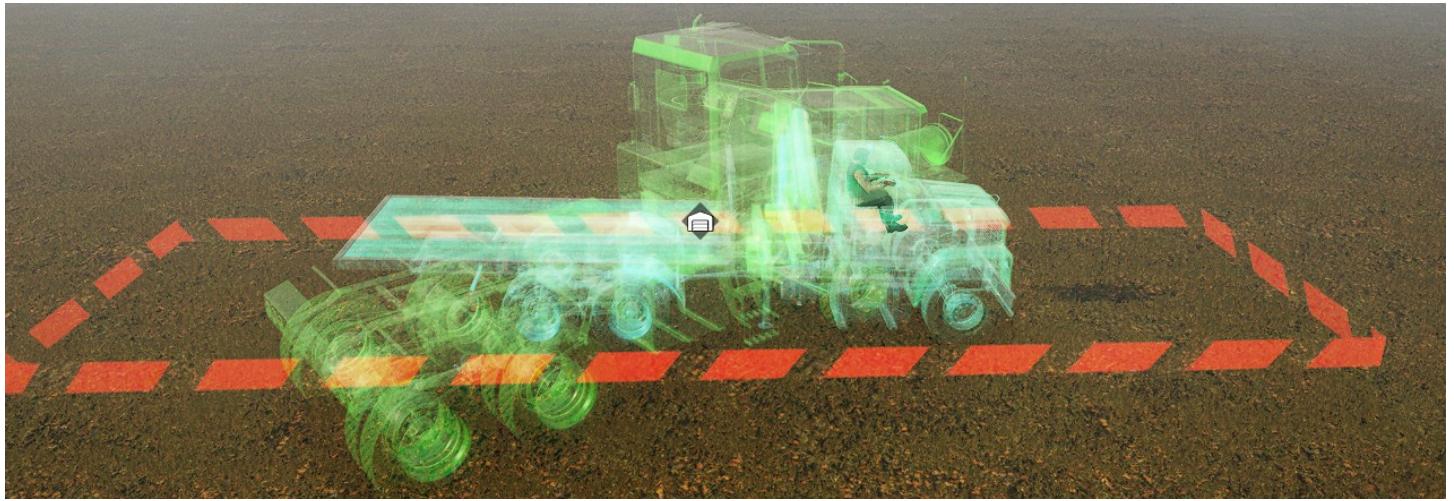
Tip: For consistency with the built-in maps, clear any icons from [Icon 30x30](#) and [Icon 40x40](#) to prevent an icon from appearing over the zone in the driving view.

The truck is centered in the exit zone, facing in the same direction as the zone's orientation. To avoid a catastrophe, make sure that the largest truck can exit without intersecting an obstacle. If you make a zone of the corresponding dimensions, it not only helps you keep the area clear of stray models, but it also helps guide the player to keep the area clear of parked trucks.

Tip: My measurements indicate that a zone with a [Length](#) of 17 meters and a [Width](#) of 5 meters (pictured below) accommodates the longest and widest trucks (including the year 1 DLC). However, extrapolating from Saber's recommendations, 21 meters by 9 meters may be safer for future DLC and/or player mods.



If another truck is in the garage exit zone, both trucks become “ghost” (virtual) trucks that ignore collisions with each other. As soon as the player drives clear of the blocking truck, both trucks become fully “real”.



Similarly, if loose cargo is in the garage exit zone, the player’s truck appears as a “ghost”, and it ignores collisions with the cargo until it moves away from the cargo.

If a model or plant is in the garage exit zone, the player’s truck may immediately collide with it and is likely to get stuck on it. Unlike a recovery zone (below), SnowRunner makes no attempt to avoid the collision, so it’s up to you to keep the area clear of obstacles on the map.



Saber says that a map “can” contain only one garage exit. My testing shows that a map can contain multiple garage exits, but the game chooses one of them for where to respawn the truck (the same one every time).

Recovery Zone

When the player chooses **Recover** from the **Functions** menu, the truck is recovered to the garage if one has been discovered. Otherwise, the truck is recovered to the zone with the **ZonePropertyRecover** type.

No zone perimeter appears in the game for a **ZonePropertyRecover** zone, and no icon appears in the driving view. The zone is also automatically suppressed from appearing on the navigation map or building list, regardless of the **Is Visible On MiniMap** property value set for the zone locator.

The recovery zone should be placed to avoid obstacles the same as for the garage exit, above. However, the game uses a very different heuristic for placing the truck than it does for the garage exit. When the truck is recovered to the recovery zone, the game automatically shifts the truck’s position to try to avoid collisions with obstacles, including other trucks. It never creates a ghost truck.

Tip: Place the recovery zone where the active truck starts so that the player returns to where she started rather than warping to somewhere new.

***map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyRecovery.priority*:** integer

Specifies the priority for use of the the recovery zone. Higher values have higher priority.

Default: 0.

You can create multiple recovery zones, and the game uses the one with the highest value in the **priority** property. It never uses a recovery zone of a lower priority, even where there are blockages that force the truck out of the recover zone, so you might as well create only one recovery zone.

If the garage is not yet discovered, and there is no zone with the **ZonePropertyRecover** type, then recovery fails. It’s not nice to do this to the player.

WARNING!

Evacuation is not available. The garage is not found and evacuation area is not established.

Trailer Store

A zone with the **ZonePropertyTrailerAttach** type allows the player to buy or sell a trailer attached to the truck. The orientation of the zone doesn’t matter.

When the player buys a trailer, it doesn't need to fit within the zone, but the purchase fails if the trailer would intersect an obstacle. Make sure there is sufficient room in at least one direction for a very long trailer such as the superheavy semi-trailer (which adds about 20 meters to the truck length).

Tip: In most cases, there will naturally be plenty of room for a long trailer in the direction of the store entrance. However, the player would often prefer to drive forward out of the trailer store after purchasing a new trailer, so consider leaving enough room for the trailer in the direction opposite the store entrance.

Fuel Station

A zone with the `ZonePropertyFuelStation` type allows the player to refuel her truck. The orientation of the zone doesn't matter.

Service Hub

A zone with the `ZonePropertyAutoRepairAndRestore` type repairs the player's truck. It also restocks both repair parts **and** fuel in any add-ons and trailers, but it doesn't add fuel to the truck's internal tanks. The orientation of the zone doesn't matter.

Unlike the fuel station, the service hub doesn't cost money, so it happens automatically without confirmation.

Watchpoint

A zone with the `ZonePropertyWatchpoint` type reveals the surrounding region on the map. The forward orientation of the zone doesn't matter.

A watchpoint zone type has a number of subproperties.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyWatchpoint.range`: integer, in meters

Specifies the radius that the watchpoint reveals on the map.

Default: 0, so nothing gets revealed.

See page 44 for more information about revealing the map. Areas within 39 meters of the active truck are revealed without the aid of a watchpoint, so the watchpoint range should be significantly larger than that unless the watchpoint is only there to launch an observation from.

The remaining properties specify the camera behavior if the player chooses to [Launch Observation](#). The camera starts at one position and orientation, then swings over to another position and orientation.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyWatchpoint.Delay at start`: integer, in milliseconds

Specifies how long the camera waits at the start position before it starts moving toward the end position.

Default: 0 ms.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyWatchpoint.cameraPosStart`: 3-D position

Specifies where the camera starts at the beginning of the observation animation.

Default: (0, 0, 0); not a useful location.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyWatchpoint.cameraDirStart`: 3-D orientation

Specifies where the camera points at the beginning of the observation animation.

Default: (0, 0, 0); the player cannot launch an observation unless the orientation has at least one non-zero value.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyWatchpoint.Duration of transition`: integer, in milliseconds

Specifies how long the camera movement takes from the start position to the end position.

Default: 0 ms; the game won't quite hang, as it does eventually respond to a press of [Esc](#) to end the observation, but it definitely has a hard time of it. The minimum effective duration is 1 ms.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyWatchpoint.cameraPosEnd`: 3-D position

Specifies where the camera stops at the end of the observation animation.

Default: (0; 0; 0); not a useful location.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyWatchpoint.cameraDirEnd`: 3-D orientation

Specifies where the camera points at the end of the observation animation.

Default: (0; 0; 0); the player cannot launch an observation unless the orientation has at least one non-zero value.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyWatchpoint.Delay at end`: integer, in milliseconds

Specifies how long the camera waits at the end position before the observation ends.

Default: 0 ms.

When editing the times, the “ms” text is uneditable. The time is always in milliseconds (thousandths of a second), so e.g. `7000 ms` is 7 seconds.

When editing the camera positions and orientations, the three values split into separate `x`, `y`, and `z` fields, each of which can be edited. The coordinate systems for position and orientation are described on page 420.

Tip: The easiest way to get a good camera position and orientation is to position the camera in the Editor, then enable the `Statistics` toggle button in the toolbar. (Remember to close the Zone Settings Editor first.) The statistics list the camera position followed by its orientation: `(X; Y; Z) > (X; Y; Z)`.

`Camera: (-26.68; 30.19; -25.96) > (0.88; -0.24; 0.40)`

Upgrade Zone

A zone with the `ZonePropertyUpgradesGiver` type grants the player a free truck add-on part and unlocks the ability to buy more. The orientation of the zone doesn’t matter. The zone disappears after being used.

The upgrade zone goes well with locked parts, described on page 293.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyUpgradeGiver.upgrades.[0]`: text

Specifies the add-on to unlock and award to the player.

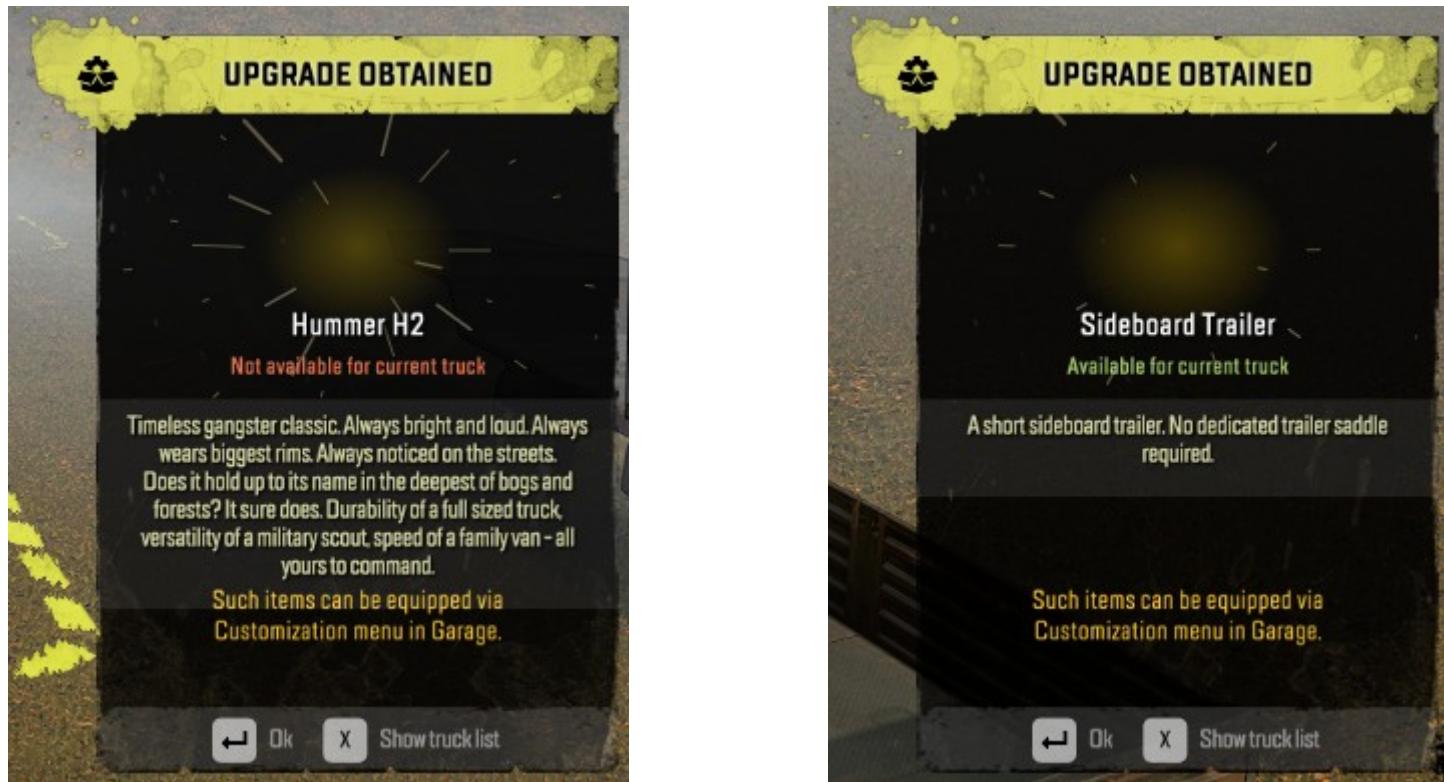
Default: blank; the upgrade zone cannot be activated.

The awarded part is available to be installed the next time the player visits a garage. If the add-on is already unlocked, then the upgrade still grants the player a free one, even if she already owns one or more.

Bug: `ZonePropertyUpgradesGiver` implies plural upgrades, and any number of numbered properties can be added within the `upgrades` subcategory, but the game grants only the first upgrade to the player. Everything after `upgrades.[0]` is ignored.

An add-on can also be awarded for completing an objective (but it cannot be unlocked this way). See page 319.

You can name a truck type or trailer type in `upgrades.[0]`, and the game will unlock it (but not grant a free copy). However, this feature is not documented and displays incorrect messages to the player with a missing icon, so it clearly isn't intended to be used. The normal way to unlock a truck or trailer is to find it on the map.



Wheels, rims, and tires cannot be unlocked or awarded by an upgrade zone. Stickers can be awarded, but since they are always unlocked and free, there's no benefit to it.

If `upgrades.[0]` has an invalid part name, then the game appears to grant the part to the player, although there is no way to buy or install the invalid part.

If no upgrade parts are listed for the upgrade zone, the zone is displayed in the game, but the player cannot activate it.

Automatic Cargo Loading Zone

A zone with the `ZonePropertyCargoLoading` type allows the player to generate packed cargo directly onto a truck or attached trailer. The zone perimeter is yellow. The orientation of the zone doesn't matter.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyCargoLoading.platformId`: regular zone ID (page 285)

Specifies the manual cargo loading zone (page 305) to associate with this automatic loading zone.

Default: blank; if the zone ID is not found on the map, no manual cargo loading zone is associated.

Bug: The `...` button to easily select a zone ID is missing from this property. Fittingly, the zone ID for the manual loading zone must be typed in manually.

The `cargoSettings` subcategory allows any number of cargo types to be available from the loading zone.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyCargoLoading.cargoSettings.[n].name`: text with dropdown menu for common choices

Specifies a cargo type that can be loaded at the automatic cargo loading zone or its associated manual zone.

Default: `CargoServiceSpareParts`.

When editing the `cargoSettings.[n].name` value, you can either type in a name or select a name from the dropdown menu. Note that the menu is missing some names, so you'll need to type those in by hand. Reference information for cargo types is on page 396. If a `name` is mistyped, that cargo type is ignored.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyCargoLoading.cargoSettings.[n].Amount`: integer

Specifies the amount of the specified cargo that is available from the automatic cargo loading zone or its associated manual zone.

Default: --1; represents an unlimited quantity of the corresponding cargo type.

If the initial `Amount` is 0, the player cannot activate the cargo zone.

On the other hand, if the player depletes the stock of any cargo type to zero, the cargo remains active in the cargo zone, but with the message "No more cargo left."

Tip: If there isn't another source for goods with a limited quantity, player mistakes can potentially ruin her game. E.g. if the player generates goods that aren't needed yet, her only option is to use a crane to set them aside for later. If a crane isn't available or the player chooses instead to destroy the goods, then a later objective may be undeliverable. On the other hand, if there is a limited source for goods nearby and an unlimited source further away, the punishment for a mistake is the usual one for SnowRunner: more driving.

The available cargo types are presented to the player in a random order. If the same cargo type is listed twice, only one of them is used; their amounts are not cumulated. If there are no entries in `cargoSettings`, the cargo zone is not displayed in the game.

If the automatic loading zone is enabled on the navigation map, it helpfully shows which cargo types are available at the zone. However, the map can list no more than 5 cargo types (chosen at random). If there are more than 5 cargo types, the map tells you how many more are available.



Multi-Stage Model

Some DLC campaign regions show a change to a multi-stage model (page 131) when an associated cargo zone is emptied of supplies. E.g. when all bricks are removed from a cargo zone, the associated brick ruins are stripped to their foundation.

Bug: This feature is not yet supported in the Editor. There is a clear stub for the function in the `zone_settings.json` file, but whenever I try to put something there, the Editor changes it back to `null` when I pack the map.

Manual Cargo Loading Zone

A zone with the `ZonePropertyManualLoading` type allows the player to generate unpacked cargo onto the ground. The player can then use a crane to manually load the cargo onto a truck. The zone perimeter is red. The orientation of the zone determines the orientation of the generated cargo.

The manual cargo loading zone must have the `ZonePropertyManualLoading` type and must be associated with an automatic cargo loading zone in order to appear in the game. Enter the manual zone's ID in the `platformId` property of the automatic zone to associate the two zones. The player can then enter the automatic loading zone and choose whether to generate goods directly onto the truck or into the manual loading zone.

Since the manual cargo loading zone is entirely controlled by the paired automatic cargo loading zone, it has no additional properties of its own. It is not possible to have a manual cargo loading zone without an associated automatic cargo loading zone. The only way to require cargo to be loaded manually is to spawn cargo as part of an objective (page 340).

If a cargo type has limited quantity in the automatic loading zone, then the available quantity is reduced whether the goods are generated in the automatic or manual loading zone.

The manual loading zone always has a “hook” icon above it in the driving view, regardless of any icons assigned to the zone properties. (Oddly, this hook is different than the ones available for named icons.) This icon always appears in the driving view, regardless of the **Is Visible On MiniMap** property value. On the other hand, the zone is automatically suppressed from appearing on the navigation map or building list, regardless of the **Is Visible On MiniMap** property value set for the zone locator.

If the manual cargo loading zone is not paired with an automatic cargo loading zone, it is entirely suppressed from the navigation map and the driving view.

A manual loading zone can be associated with more than one automatic cargo loading zone, although this probably isn’t very useful.

Cargo Generation Location

Caution: Cargo is generated with a different orientation than cargo that is spawned for an objective (page 342)

If we assign a “forward” direction for cargo based on how it is loaded on the truck, then the cargo is generated in the manual loading zone pointing 90° to the right of the zone orientation. I.e. long goods are oriented with the long end perpendicular to the zone’s direction.

Tip: The zone’s dimensions have no effect on where the cargo is generated. But setting the dimensions to the maximum size of potential cargo helps inform the player where to expect the cargo to appear and where to avoid parking a truck. A raised platform can also keep the player’s truck clear of the cargo zone.



Cargo Generation Height

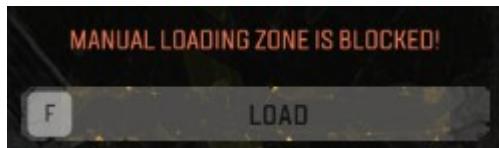
The game attempts to generate cargo just above ground level. If the ground is hilly, it tries to generate the cargo high enough to not be inside the ground. If it detects a model in the way, it generates cargo just above the model. Its heuristic isn't very good, though. It does best when the ground is very flat and any model underneath is well centered and also flat (such as the various **platform** models).

Tip: To perfectly center a model in the zone, simply copy the zone's **Org X** and **Org Z** values to the model's **X** and **Z** properties. Unfortunately, the zone's direction isn't even in the same format as the model's, so you're on your own to align their orientations. The default orientation for zones (pointing along the X axis) happens to match the default orientation for platform models, however, so you're all set if that orientation works for you.

The zone's **Height** property has no influence on the cargo generation height. However, if **Is Wall** is **True**, the **Height** property is useful to position the zone perimeter just above a platform model.

Cargo Generation Conflicts

If a truck or trailer intrudes into the manual loading zone, the game won't allow cargo to be generated.



If unpacked cargo of intrudes into the manual loading zone, and that cargo is produced by the loading zone in unlimited quantity, the game quietly deletes it when generating new cargo.

But if the intruding cargo is not produced by the loading zone or is produced only in limited quantity, the game asks for confirmation before deleting the intruding cargo to make space for new cargo.



Logs

The height heuristic for generated cargo performs poorly with the standard log trestle (`logs_scavange_02`), but the game compensates by generating a log 2 meters higher than it would for other cargo. This gives the log room to drop into place. Unlike other cargo types, logs can share space with other logs of the same type in the manual loading zone.



Once three logs are in the zone, the game won't allow more logs to be spawned until at least one is removed.



When other cargo is generated (including logs of a different type), it deletes any number of logs from the zone.

Bug: If logs are available from the loading zone in limited quantity, the automatic loading zone creates a full load with one log, but the manual loading zone requires three logs to be generated to pack a full load.

Crafting Zone & Storehouse

A zone with the `ZonePropertyStorehouseCraft` type allows the player to craft cargo from other inputs and then load it like an automatic cargo loading zone. It also allows the player to store cargo to be retrieved later. For brevity, I'll usually just call it a crafting zone; the storehouse is implied.

The zone perimeter is yellow. The orientation of the zone doesn't matter.



The crafting zone acts as an automatic loading zone for cargo in its storehouse inventory. The player can alternatively send cargo to the associated manual loading zone.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyStorehouseCraft.platformId`: regular zone ID (page 285)

Specifies the manual cargo loading zone (page 305) to associate with the crafting zone.

Default: blank; if the zone ID is not found on the map, no manual cargo loading zone is associated.

Bug: The `...` button to easily select a zone ID is missing from this property. Type in the zone ID manually instead.

The game asks for confirmation before deleting intruding cargo in the manual loading zone to make space for new cargo.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyStorehouseCraft.energyZoneId`: global zone ID (page 285)

Specifies the energy generation zone (page 311) to associate with the crafting zone.

Default: blank; no energy generation zone is associated.

If the crafting zone is assigned when the map is initialized, I find that the energy generation zone is given the `ZonePropertyEnergy` type automatically, even if that type is not assigned in the Zone Settings Editor. However, if the crafting zone is assigned by an objective reward, the associated energy zone must either already have the `ZonePropertyEnergy` type or the type must also be assigned by the objective reward.

Bug: If the crafting zone is assigned when the map is initialized, and its `energyZoneId` matches the zone ID of the crafting zone, the game hangs when it loads the map.

Tip: A separate energy generation zone can be created with the exact same position and perimeter as the crafting zone in order to allow energy to be supplied in the same area as the crafting zone. The doubled zone perimeter appears extra bright, but it's not too bad. Alternatively, set the `Thickness` of either zone perimeter to 0 to make it invisible, leaving only one perimeter displayed. Set the zone locator's `Is Visible in Minimap` property to `False` for the energy generation zone to suppress it from the navigation map and its icon from the driving view.

Bug: The `energyZoneId` value is specified as a global zone ID, and that zone can be used as an energy zone even if it is on a different map. However, the generator trailer shuts off whenever the player leaves the map, so the storehouse can never use that energy.

The `craftSettings` subcategory allows any number of cargo types to be manufactured by the crafting zone. If no items are added, the zone cannot be activated.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyStorehouseCraft.craftSettings.[n].name`: text with dropdown menu for common choices

Specifies a cargo type that can be manufactured at the crafting zone.

Default: `CargoServiceSpareParts`.

When editing the `cargoSettings.[n].name` value, you can either type in a name or select a name from the dropdown menu. Note that the menu is missing some names, so you'll need to type those in by hand. Reference information for cargo types is on page 396. If a `name` is mistyped, that cargo type is ignored.

Once the player has manufactured one or more cargo items, they go to the crafting zone's storehouse inventory. From there, the player can load them onto her truck or send them to the associated manual loading zone.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyStorehouseCraft.craftSettings.[n].craftCount`: integer

Specifies the amount of the specified cargo that is manufactured each time.

Default: -1.

Each craftable cargo type can optionally require one or more input components (cargo).

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyStorehouseCraft.craftSettings.[n].cargoComponents[x].name`: text with dropdown menu for common choices

Specifies a cargo type that must be in the storehouse in order to craft cargo.

Default: `CargoServiceSpareParts`.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyStorehouseCraft.craftSettings.[n].cargoComponents[x].Amount`: integer

Specifies the amount of the specified cargo that is consumed from the storehouse in order to craft cargo.

Default: 1.

Each craftable cargo type can also optionally require fuel from a generator trailer.

`map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyStorehouseCraft.craftSettings.[n].energy`: integer

Specifies the amount of fuel that is consumed from the generator trailer in order to craft cargo.

Default: 0.

If the `energy` value is non-zero, an energy generation zone must be assigned in the `energyZoneId`, or the game will trigger a ModMapError (page 287). The player must bring a generator trailer to the associated energy generation zone and start the generator from the truck functions menu. Once the trailer is running, the player can detach the trailer, with the caveat that the generator stops whenever the player reloads the map.

The game can display up to four required input types, where the energy/fuel input counts as one of them if the `energy` value is non-zero. If there are more than four required input types, the game only lists the first four, but all are still required. That leaves the player in the dark as to what inputs may be missing, so don't do that.

If the crafting zone requires no energy or cargo components, then it essentially acts as an automatic cargo loading zone (page 303) with an associated storehouse.

Energy Generation Zone

A zone with the `ZonePropertyEnergy` type allows the player to turn on a generator trailer in the zone. The zone appears in the game simply by having the `ZonePropertyEnergy` type, but it is only useful if it is associated with a crafting zone. Enter the energy generation zone's global zone ID in the `energyZoneId` property of the crafting zone to associate the two zones (page 308).

The energy generation zone perimeter is yellow. The orientation of the zone doesn't matter.

The energy generation zone is automatically given a lightning bolt icon in the driving view, overriding any assigned icon. However, the navigation map uses the icons assigned in [Icon 30x30](#) and [Icon 40x40](#).

Bug: The built-in campaign uses a lightning bolt icon in the navigation map, but the name of this icon isn't documented, and I haven't been able to guess it.

Living Area

Bug: The [ZonePropertyLivingArea](#) zone type is not yet supported.

Gateway

A gateway only makes sense at the regional level (page 375). A gateway between zones in the same map results in weird behavior such as duplicated trucks and trucks that aren't shown on the map.

Multiple Zone Types

Multiple types can be assigned to a single zone. For zone types that do something automatically, all automatic effects are invoked. For other zone types, the player can switch among the various interaction options.



Tip: The player might not notice that she can switch to another interaction option. The built-in campaign uses multiple interaction options only when the additional option is for a cargo delivery goal (page 332), which adds an icon to the navigation map and driving view to help make the extra option more obvious.

Tip: A fuel station and a service hub can be combined cleanly since the service hub is automatic and doesn't require an interaction option. There is also an icon that represents the combined fuel and service function so that the player can easily understand the purpose of the zone.

There is never a reason to assign the same zone type more than once to a zone, and it probably isn't supported cleanly, so you shouldn't do it.

Objectives

Because a SnowRunner objective often involves visiting multiple zones (or occasionally none at all), objectives are not specified within a zone. Instead, objectives get a separate `objectiveSettings` category in the Zone Settings Editor.

An objective can take one of three forms:

- A contract can be activated without the player needing to visit a particular zone. Contracts are listed in the game under the employer offering the contract.
- A task is offered when the player visits a particular zone.
- A contest is also offered when the player visits a particular zone, and it offers different rewards depending on the speed of completion.

All three objective types are very similar. Any differences are discussed where appropriate in the sections below.

Objective Status Terminology

I use the following terms to describe the status of objectives:

- Locked: The player does not have the required rank to activate the objective, or it is blocked by another objective.
- Unlocked: The player meets all of the requirements for an objective to be offered or activated. An unlocked task or contest is offered when the player drives to the starting zone. The player can activate an unlocked contract from list of contracts in the navigation map.
- Offered: The game offers an unlocked task or contest when the player visits its starting zone. If the player accepts the offer, the objective is activated.
- Activated: The player can perform the stages of the task. Multiple objectives can be activated at once with different levels of progress in each.
- Tracked: The goals of the objective's current stage are displayed in the upper right corner of the screen. Only one objective can be tracked at a time, and the player can get credit for completing an objective's goal only when the objective is being tracked.
- Completed: The player has completed all stages of the task and received the reward.

Custom Employers

Contract objectives are associated with employers. The game has a number of built-in employers, described on page 401. Or you can create your own employer using one of the two method below.

New Employer

`objectiveSettings.employer:` +

Specifies the name of a new custom employer as UI text (page 282); formatting tags are not supported; max ~30 characters.

The built-in employers use all capital letters in their names, but you can use lowercase if you want.

It's OK to define new employers that you never use. If no contract uses an employer, the game does not display that employer in the contract list.

`objectiveSettings.employer.Employer Name.Employer Logo:` text

Not supported; the new employer uses a default image identical to Husky Forwarding (page 401).

Default: blank.

`objectiveSettings.employer.Employer Name.Employer Background:` text

Not supported.

Default: blank.

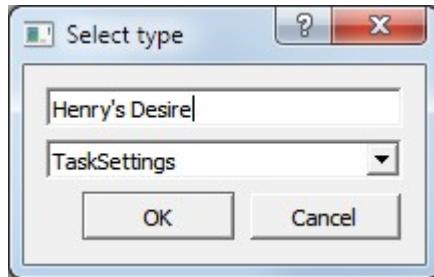
Modified Employer

If you like the icon of one of the built-in employers, you can use localization (page 284) to change its name while keeping its icon. The localization ID for each employer is listed on page 401. If you include this localization ID in your map's localization files, it overrides the default name (in each language).

Add Objective

To add an objective, start the Zone Settings Editor and open the `objectiveSettings` category. Add a new objective by clicking the + to the right of `tasksSettings`, `contractsSettings`, or `contestSettings`.

The resulting dialog box looks similar to the one used for [ZoneProperties](#), but is used very differently. In the top field, enter the name for the objective. The dropdown menu has only a single entry which cannot be changed.



Bug: The dialog box is called “Select Type”, but you actually use it to name your objective.

objectiveSettings.typeSettings: +

Specifies the name of a new objective as UI text (page 282); formatting tags are not supported; max ~36 characters.

The objective name is shown to the player, but it is also used by the game to internally track the status of the objective, so the name must be unique among all objectives, regardless of type. After the objective is added, you can edit its name by double-clicking it.

Tip: Localization potentially allows you to show the player the same name for objectives with different internal names, although this could be somewhat confusing to the player, especially since the game rearranges the order of the objective list depending on what the player has clicked recently.

Requirements

An objective is not offered and cannot be activated until the player meets the required rank and until all blocker objectives are complete.

objectiveSettings.typeSettings.objective name.Required rank: integer

Specifies the minimum rank required to unlock the objective.

Default: 0; the player always meets the rank requirement.

objectiveSettings.typeSettings.objective name.Blocker Objectives.[n]: name

Specifies the name of another objective that must be completed before this one can be activated.

Default: blank; an invalid name triggers a ModMapError (page 287).

A contest can be listed as a blocker objective, in which case it unblocks the dependent objective the first time it is completed with any trophy.

Offer Zone (Task or Contest)

A task or contest requires an offer zone. When the task or contest is unlocked and the player enters this zone, the game offers the objective to the player.

A task or contest can be activated as soon as the player enters the map (page 289), but it still requires an offer zone where the player can restart the objective if desired.

The primary offer zone is specified in the `taskGiverZone` property. Additional zones can be specified by adding properties under the `additionalTaskGivers` category.

`objectiveSettings.typeSettings.objective name.taskGiverZone`: global zone ID (page 285)

Specifies the primary zone where the objective is offered.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

`objectiveSettings.typeSettings.objective name.additionalTaskGivers.[n]`: regular zone ID (page 285)

Specifies an additional zone where the objective is offered.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

When a task or contest is unlocked, its offer zone appears with a yellow perimeter. If the offer zone is shared with a zone type that normally has a red perimeter, the perimeter turns yellow until the objective is completed.

Bug: If two objectives are given by the same zone, accepting one may accept the other, **even if the other should not be offered** (e.g. because it is blocked or completed).

Tip: One workaround is to create duplicate zones with the same name and location, but different zone IDs, but that still leaves the next problem...

Bug: If two objectives are simultaneously offered in the same location, the game only offers one to the player. The player cannot activate the second until she completes the first.

Tip: Avoid offering objectives in the same location at the same time. E.g. use **Blocker objectives** to offer them at separate times.

Bug: If the player's truck straddles the offer zones of objectives with adjacent locations (the same objective in both zones or two different objectives), the game sometimes gets confused and may offer neither objective until the player leaves both zones and then returns to just one of them.

Tip: Make sure that objectives are offered sufficiently separate in space to prevent straddling or use **Blocker objectives** to offer them at separate times.

Tip: It is also helpful to offer objectives separate from other interactive zones so that the user doesn't need to switch among interaction types. This separation doesn't need to be as far, however, since the player can more easily identify the problem and the fix.

A contract doesn't have an offer zone. Instead, a contract is offered from the navigation map by an employer (below).

Employer (Contract)

A contract requires an employer who offers the objective. The employer can be one of the built-in employers or a new employer.

objectiveSettings.contractsSettings.objective name.employer: text with dropdown menu for common choices

Specifies the employer who offers the contract objective.

Default: **Husky**.

The names and other reference material for each employer are listed on page 401. Note that some built-in employers are missing from the dropdown menu, but you can still type in their names manually.

A new employer requires additional configuration (page 314).

Bug: If a new employer is not properly configured or if an employer name is mistyped, the game **might** hang when the navigation map is opened. Or it might wait until you've edited the map a few more times, so it can be very mysterious what is causing the game to hang. Be very careful with your employer names.

Objective Description

objectiveSettings.typeSettings.objective name.Ui Desc: UI text (page 282); formatting tags are not supported; max characters > 600, but the text gets smaller and smaller

Specifies the objective's description that is shown to the player.

Default: blank.

Bug: If the **UI Desc** value is blank, the objective details won't show a description, but the offer popup shows the description of the previous offered objective!

When the player views the objective offer, the game shows an overview of the objective, including its description. The description is also displayed when the player views the objective details from the navigation map.



The objective offer shows an icon for each goal type required by the objective. If an objective has multiple goal types, multiple icons are displayed.



Recommended Equipment

The objective can inform the player what equipment is recommended to complete the objective.

`objectiveSettings.typeSettings.objective name.Recommendation`: dropdown menu

Specifies the recommended equipment for the objective.

Default: **NONE**.

The recommendation options are as follows:

- **NONE**: The area for recommendations is left blank.
- **CRANE**: A truck with a crane is recommended.
- **OFFROAD**: An offroad truck is recommended.
- **SCOUT**: A scout truck is recommended.
- **SEISMIC**: A truck with a seismic vibrator is recommended.
- **METAL_DETECTOR**: A truck with a metal detector is recommended.

Bug: A metal detector is used to detect loose cargo, so it implies that a crane is also needed. However, there is no way to tell the game to recommend both a metal detector and a crane.

Bug: The Zone Settings Editor doesn't have a menu item to allow recommendation of a truck with a log-loader crane. To recommend a log-loader crane, hand edit the `levels/map_name/objective_settings.json` file and change the `recommendedTruck` value to a numeric (unquoted) `6`. This displays as a blank value in the Zone Settings Editor (distinct from the default `NONE` text). As long as you don't edit the `Recommendation` value, the recommendation is retained as you make other edits in the Zone Settings Editor.

Rewards (Task or Contract)

Any number of rewards can be added to a task or contract.

`objectiveSettings.typeSettings.objective name.rewards`: +

Specifies the type of reward to add to the objective, chosen from a pulldown menu.

The choices of reward type are as follows:

- **ObjectiveRewardExperience**: awards some amount of experience toward the next rank.
- **ObjectiveRewardMoney**: awards some amount of money.
- **ObjectiveRewardItem**: awards an item.
- **ObjectiveRewardOpenZoneProperties**: adds a new type to a zone.

Experience

`objectiveSettings.typeSettings.objective name.rewards.ObjectiveRewardExperience.amount`: integer

Specifies the amount of experience to award to the player when the objective is complete.

Default: 0.

Money

`objectiveSettings.typeSettings.objective name.rewards.ObjectiveRewardMoney.amount`: integer

Specifies the amount of money to award to the player when the objective is complete.

Default: 0.

Item

`objectiveSettings.typeSettings.objective name.rewards.ObjectiveRewardItem.Item name`: text

Specifies the type of truck or add-on to award to the player when the objective is complete.

Default: blank; no item is awarded.

If a truck type is awarded, the new truck is sent to the garage storage area. Note that a truck can also be awarded via a truck delivery goal (page 350). Alternatively, the rewarded item can be an add-on, in which case it goes into the player's garage inventory.

Tip: Use the reward description (below) to remind the player to check the garage storage area for an awarded truck.

Bug: If you award a trailer type, then the next time that the player views her garage storage area, the game will crash, presumably because there is an unexpected trailer in the garage. The safe way to award a trailer is via a truck delivery goal (page 354).

Wheels, rims, and tires cannot be awarded by an objective. Stickers can be awarded, but since they are always free, there's no benefit to it.

Unlike an upgrade zone, an objective award does not unlock the specified item. If a truck is awarded, but locked, the player can retrieve the awarded truck from storage, but cannot buy more of that type until it is unlocked. If an add-on is awarded, the part goes to the player's inventory, but it cannot be installed until it is unlocked.

Access to Zone

The `OpenZoneProperties` reward type is typically used to add a type to a zone that did not have one before. This makes the zone visible with the new type. Any number of zones can have types added as part of this reward type. Zone types are described starting on page 295.

`objectiveSettings.typeSettings.objective name.rewards.ObjectiveRewardOpenZoneProperties.zoneSettings:` +

Specifies a name for the new zone settings. The name has no purpose except for display in the Zone Settings Editor.

Default: `ZonePropertiesSettingsObjectiveReward`.

Instead of specifying a global zone ID (page 285), separate subproperties are created to hold the map name and zone ID.

`objectiveSettings.typeSettings.objective name.rewards.ObjectiveRewardOpenZoneProperties.zoneSettings.ZonePropertiesSettingsObjectiveReward.zoneId`: regular zone ID (page 285)

Specifies the zone for which one or more types are being added.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

Bug: The `...` button to easily select a zone ID is missing from this property. Type in the zone ID manually instead.

`objectiveSettings.typeSettings.objective name.rewards.ObjectiveRewardOpenZoneProperties.zoneSettings.ZonePropertiesSettingsObjectiveReward.mapId`: text; editable in Region Settings Editor; read-only in Zone Settings Editor

Specifies the name of the map that contains the zone specified above.

Default: blank in Region Settings Editor; an invalid map name triggers a ModMapError (page 287).

`objectiveSettings.typeSettings.objective name.rewards.ObjectiveRewardOpenZoneProperties.zoneSettings.ZonePropertiesSettingsObjectiveReward.props`: +

Specifies a zone type to assign to the zone, exactly as zone types are normally assigned (page 295).

Bug: A new watchpoint type permits the player to launch an observation, but it does not actually reveal any of the map, regardless of the `range` specified. My testing suggests that the other reward zone types work exactly like the permanent zone types.

The game tells the player that a new zone type is unlocked, but not where to find it or what type it is.



The new zone type is mentioned in the objective offer and objective details, but not in the reward window. The new zone name is briefly mentioned in the log of accomplishments that scrolls rapidly past on the right side of the driving view, but speaking for myself, I almost never notice what goes by in that log.

Tip: Use the reward description (page 327) to let the player know that the new zone is open.

Multiple Rewards

Any number of rewards can be awarded. However, the game can list no more than one **Experience** reward and one **Money** reward, and which reward it lists of each type is not consistent throughout an objective. There is no reason to give more than one experience reward and one money reward, and you should avoid doing so.

The game separately lists additional awards such as items and access to zones. The game can list up to three additional awards in the objective details, but no more than one in the offer window or reward window. Consider limiting yourself to one additional award.

The dialog to add a new reward type allows you to change the name used for the reward. If you wish to award multiple rewards of the same type, you must assign unique names for the additional rewards. The name is only used for display in the Zone Settings Editor, so pick something that makes sense to you.

If multiple new zones are configured within `rewards.ObjectiveRewardOpenZoneProperties.zoneSettings` (i.e. one reward with multiple zones), then the game groups all of them into one “Access to Location” message. However, if the new zones are configured as separate entries under `rewards` (i.e. multiple rewards of one zone each), then the game gives each one a separate “Access to Location” line in the rewards list in the objective details (up to the limit of three items and/or zones). As described above, however, only one of these is listed in the offer window and reward window.



Rewards (Contest)

A set of rewards must be added to a contest. The reward set consists of separate rewards for a gold, silver, or bronze completion time.

Bug: If you don't have any rewards for a contest (i.e. no `ObjectiveRewardsByTime` property), the game hangs when it tries to load your map.

objectiveSettings.contestSettings.objective name.rewards: +

Specifies a name for the reward set. The name has no purpose except for display in the Zone Settings Editor.

Although you can have multiple reward sets, they suffer from the same complications with multiple rewards as described in the section above.

Each trophy has an associated **Time limit**, but only the gold and silver time limits are used. The bronze time limit is ignored:

- If the player completes the contest within the gold time limit, the gold rewards are awarded.
- Otherwise, if the player completes the contest within the silver time limit, the silver rewards are awarded.
- Otherwise, if the player completes the contest in any amount of time, the bronze rewards are awarded.

objectiveSettings.contestSettings.objective name.rewards.ObjectiveRewardsByTime.timeSettings.TROPHY.

Time limit: integer, in seconds

Specifies the contest time that the player must beat in order or to win the associated trophy rewards.

Default: 0 s; the trophy is impossible to win unless the objective goals are deliberately trivial.

The silver time limit should be greater than the gold time limit. Otherwise, the silver trophy is unachievable, and the contest offer looks weird.

Experience, money, and item rewards are the same as in the section above. A contest cannot open a zone as a reward.

objectiveSettings.contestSettings.objective name.rewards.ObjectiveRewardsByTime.timeSettings.TROPHY.

Experience: integer

Specifies the amount of experience to award to the player when the trophy is won.

Default: 0.

objectiveSettings.contestSettings.objective name.rewards.ObjectiveRewardsByTime.timeSettings.TROPHY.

Money: integer

Specifies the amount of money to award to the player when the trophy is won.

Default: 0.

objectiveSettings.contestSettings.objective name.rewards.ObjectiveRewardsByTime.timeSettings.TROPHY.

Possible Item reward: text

Specifies the type of truck or add-on to award to the player when the trophy is won. See page 320 for details.

Default: blank; no item is awarded.

Bug: The game awards the listed rewards regardless of how many times the player has achieved any trophies. This is the only case in which the player can “grind” an objective to get multiple rewards. It would be much better if the game only awarded the **difference** between an improved result’s rewards and the previous best rewards.

Bug: An item reward associated with trophy is not awarded if the player gets a better trophy. You should probably have only one possible item reward which you specify as a reward for each trophy at the desired level and above. I.e. reward the item for GOLD only; or for GOLD or SILVER; or for GOLD, SILVER, or BRONZE.

The contest offer dialog includes the required times, the experience and money rewards. Item rewards are not listed in the offer or the reward dialog, so you probably want to include them in the contest introduction **and** reward description.



Tip: Consider giving cheap customization add-ons as part of the contest awards. This encourages the player to install cool add-ons that they might otherwise never bother with, but it doesn’t make the contest feel mandatory.

Fail Reasons (Contest)

There are three fail reasons that are always present and cannot be deleted:

- **RecoveryFailReason:** the player fails the contest if she recovers her truck to the garage or starting area.
- **ChangeTruckFailReason:** the player fails the contest if she changes trucks.
- **GarageFailReason:** the player fails the contest if she enters a garage.

Additional fail reasons can be added.

`objectiveSettings.contestSettings.objective name.failReasons: +`

Specifies a new fail reason type from a dropdown menu.

The options for fail reason types are described in the sections below.

Although you can repeat a fail reason type more than once by giving it different names, there is never any useful reason to do so.

The game lists up to three of the additional fail reasons in the contest offer dialog. If there are more than three, it does not list them all.



First-Person View Requirement

When `CockpitFailReason` is added to the `failReasons` list, the game automatically switches to cockpit view when the contest starts, and the player fails if she changes views.

Damage Limit

When `DamageFailReason` is added to the `failReasons` list, the player fails if she accumulates more than damage than allowed.

`objectiveSettings.contestSettings.objective name.failReasons.DamageFailReason.maxDamage: integer`

Specifies the limit of damage points that the player can accumulate during the contest.

Default: 0; any damage fails the contest.

Time Limit

When `TimeFailReason` is added to the `failReasons` list, the player fails if she takes more time than allowed.

`objectiveSettings.contestSettings.objective name.failReasons.TimeFailReason.timeLimitSec: integer, in seconds`

Specifies the maximum time that the player can take to complete the contest.

Default: 0; the player immediate fails the contest unless the objective goals are deliberately trivial.

This is the real lower bound to achieve the bronze trophy, although the game never presents it as such.

Bug: If `TimeFailReason` is given, the game says only “Time limit present” without specifying the limit.

Contest Hours

When `HoursFailReason` is added to the `failReasons` list, the contest runs at a certain time of day.

`objectiveSettings.contestSettings.objective name.failReasons.HoursFailReason.startHour`: integer, in hours after midnight

Specifies the time at which the contest starts.

Default: 0 (midnight).

`objectiveSettings.contestSettings.objective name.failReasons.HoursFailReason.finishHour`: integer, in hours after midnight

Specifies the time at which the contest ends.

Default: 0 (midnight).

When the contest is accepted, the game automatically switches to the time given by `startHour`, and the player fails if time reaches the `finishHour`.

Hours are integers and should be from 0 – 23, inclusive. Values outside this range are treated as 0 (midnight). If `finishHour` is less than `startHour`, the contest ends at the finish hour in the next day. The game fails with a `ModMapError` (page 287) if `finishHour` is equal to `startHour`.

One hour of game time is equivalent to 150 seconds (2m30s) of real-world time.

Bug: If `HoursFailReason` is given, the game always says “Complete during night”, even if the hours given are daytime hours. The game also doesn’t tell the player how many hours the contest runs.

Reward Description

When the player completes an objective, by default a window displays the objective rewards along with a configurable message.



`objectiveSettings.contestSettings.objective name.Don't show reward popup`: checkbox

Specifies whether to display or suppress the reward window.

Default: unchecked; the window is displayed.

If the reward window is suppressed, music still plays when the player completes the objective, and the rewards are awarded without an associated message.

`objectiveSettings.contestSettings.objective name.Reward Description`: UI text (page 282); formatting tags are not supported; max ~300 characters

Specifies the message to display in the reward window.

Default: blank; an error message is displayed in the reward window instead.

Despite its property name, the reward description is probably used more often to comment on the completed job than on the reward. The `Reward Description` property isn't used if `Don't show reward popup` is checked.

If multiple objectives are completed simultaneously, the reward for only one of them is shown.

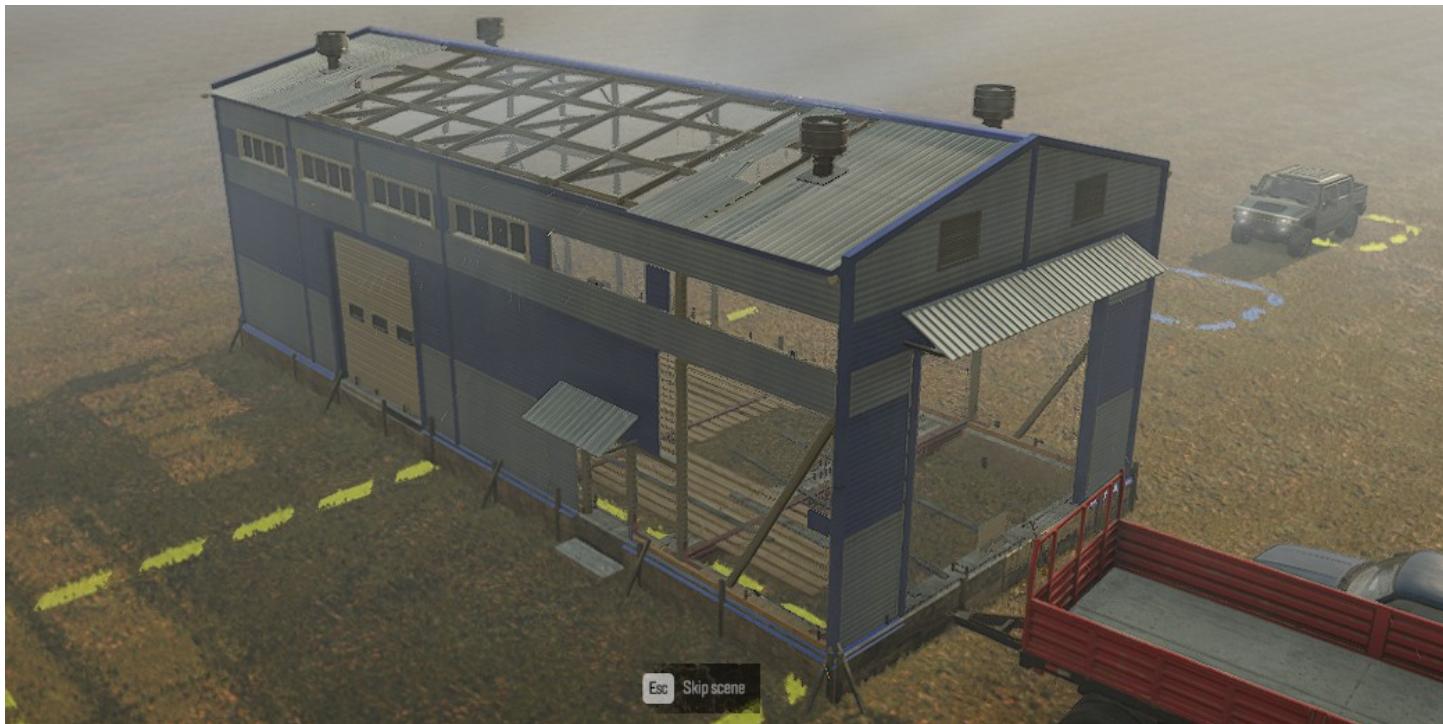
Tip: Physically separate the objective destination zones or use blocker objectives (page 315) or other means to separate them in time.

Once an objective is completed, it is removed from the list of objectives next to the navigation map. Any zones that were only used for a completed task or contract are hidden.

Since a contest can be accepted again after being completed, it remains in the list of **Task Givers** next to the navigation map, and the offer zone remains visible in the driving view. The offer zone is also shown on the navigation map when the contest is selected.

Multi-Stage Models

Multi-stage models (page 131) change state in response to completion of an objective stage (this section) or delivery of cargo (page 338). The change can be purely visual, or it can remove a barrier that was blocking trucks from passing. Some models have a fancy animation to show a change, while others simply swap to the new shape with a sound effect and a cloud of dust.



Most multi-stage models have the suffix **_objective** and so can be easily found in the model asset list. Reference information for the available multi-stage models is on page 410.

Click the ▼ to the right of **Model Building Settings (Depend On Stages)** to connect a multi-stage model to the player's progress in the objective's stages.

`objectiveSettings.typeSettings.objective name.Model Building Settings (Depend On Stages).levelName`: text; editable in Region Settings Editor; read-only in Zone Settings Editor

Specifies the name of the map that contains the multi-stage model.

Default: blank in Region Settings Editor; an invalid map name triggers a ModMapError (page 287).

`objectiveSettings.typeSettings.objective name.Model Building Settings (Depend On Stages).modelTag`: text

Specifies the ID of the model that changes state as objective stages are completed (page 131).

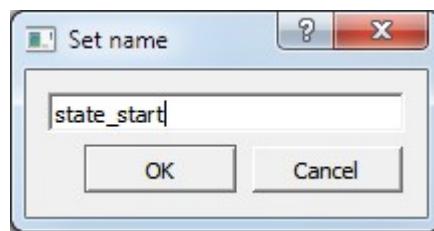
Default: blank; triggers a ModMapError (page 287).

An objective can change only a single model in response to completion of the objective stages. Also, a model ID cannot be used by more than one objective, nor by more than one property within an objective. An invalid model ID is ignored.

`objectiveSettings.typeSettings.objective name.Model Building Settings (Depend On Stages).stagesProgress`:



Specifies a model stage name to an association for.



As a reminder, after you've added any name, you can change the name by double clicking it in the property list.

The easy way to get model stage names is by checking the reference on page 410. The hard way is to open the `initial.pak` archive in the SnowRunner installation directory (page 426), find the model XML file under `[media]/classes/models` (or in one of the `[media]/_dlc` directories), and then find the name attributes specified in each `<Subset>` tag. But you shouldn't need to use the hard way unless you want to use a newer model that I haven't yet added to my reference.

Once a model stage name is added, you can associate it with an objective stage number.

<code>stagesProgress</code>		
<code>build_stage_0</code>	0	
<code>build_complete</code>	1	

`objectiveSettings.typeSettings.objective name.Model Building Settings (Depend On Stages).stagesProgress.
model stage name`: integer

Specifies the objective stage number to associate with the named model stage.

Default: 0; the model is initialized to the named model stage when the map is started and when the objective is restarted.

Because property names must be unique, a model stage name can only be associated with a single objective stage number. However, that same model stage can be retained for any number of consecutive objective stages until the next objective stage that has an associated model stage.

If you don't associate any model stage with objective stage 0, then the game initializes the model to its default stage (the first stage listed in the reference).

The game may behave unpredictably if a model stage name is mistype or if multiple model stages are associated with the same objective stage.

Bug: If no model stage name is associated with objective stage 0, the model's landmark may not be rendered on the navigation map in the correct default stage.

Potential Model Collisions

When a model changes shape, there is a chance that its new shape could collide with a truck. Most of the built-in models are safe from this, as they either don't change their collision boxes or they only remove collision boxes. But a few models add collision boxes, so you may want to place obstacles or use impassable terrain to keep the player from parking a truck there. And certainly keep the stage completion zone(s) away from where the model is about to move or appear.

You don't have to use model stages in their default order. However, there is no animation or sound effect when a model makes a non-standard transition. Also, reversing model stages has an increased chance of causing a collision (e.g. if you reverse the stage order to add a barrier instead of removing a barrier).

If an objective removes a model barrier and then continues with additional stages, the player may choose to restart the objective. If the player has parked a truck where the barrier was, then this can also cause trouble. In this case, though, the player should be able to rescue her truck (perhaps with significant damage) by completing the stage to remove the barrier again.

Objective Stages

An objective is split across one or more stages which the player must complete sequentially. Add a stage by clicking the  to the right of the **Stages** container.

Each stage is comprised of one or more goals which the player may complete in any order. The goals in a stage can all be the same type or a mix of types.

For each added stage, all of the available goal types are listed. The method of adding each type of goal and the details for each type are described below.

In addition to having goals, a stage can also spawn cargo as described on page 340.

Conflicting Properties

When a goal refers to a particular truck, it can give it a distinctive name that is used on the navigation map. If multiple goals refer to the same truck, only one of those names can be applied at a time. The property that gets applied is often random, and in some cases the random result is not consistent. E.g. the truck can have one name on the map and another name in the objective details. I recommend that multiple goals refer to the same truck, all should assign it the same name.

Similarly, if the game groups multiple goals into one entry, I recommend that they all share the same goal description.

Cargo Delivery Goal

A stage can require the player to deliver certain cargo to one or more zones. Expand the actions property for the stage with the black arrow ▼, then click the + to the right of `zoneToFill` to add a cargo delivery goal. If no members are added to `zoneToFill`, the objective cannot be accepted.

Stages	[0]	+
actions	ActionPackSettings	-
zoneToFill	[0]	+
[0]	Ui Desc	Deliver to the <y>Swamp</y>:
Truck need to visit Uid:		
globalZoneId	level_test store_1	
Manual Unloading Settings	null	-
cargo	CargoPack	-
name	CargoRadioactive	-
Amount	1	
Model Building Settings (Depend On Cargo)	null	-
additionalUnloadZones		+
truckDelivery		+
visitAllZones	null	-
changeTruck	null	-
repairTruck		+
Seismograph Settings	null	-
spawnCargoOnStageActive		+
livingAreaInfo	null	-
Model Building Settings (Depend On Stages)	null	-
Blocker Objectives		+

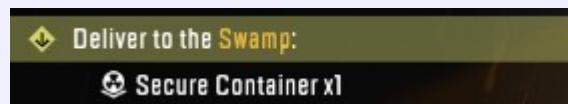
Goal Description

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].Ui Desc`: UI text (page 282); formatting tags are supported; max ~125 characters

Specifies a message that describes the goal in the objective details and tracking info.

Default: blank; an error message is displayed instead.

Tip: The built-in campaign uses the `<y>` tag to highlight named map locations in the manner shown below, and you may wish to do the same. Notice how the colon : sets up the cargo to deliver, which the game always writes immediately after the goal description.



Cargo to Deliver

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].cargo.name`: text with dropdown menu for common choices

Specifies the cargo type required by the cargo delivery goal.

Default: `CargoServiceSpareParts`.

When editing the `cargo.name` property value, you can either type in a name or select a name from the dropdown menu. Note that the menu is missing some names, so you'll need to type those in by hand. Reference information for cargo types is on page 396.

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].cargo.Amount`: integer

Specifies the amount of cargo required by the cargo delivery goal.

Default: 0; the objective cannot be accepted.

The player is allowed to deliver less than the full amount at once, so she can make multiple trips to fulfill a large goal.

The properties don't support multiple cargo types within a single goal. Instead you must add more goals under `zoneToFill`, each with their own description, etc. There is a trick to make them appear to be a single goal to the player, described in the next section.

Delivery Zones

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].globalZoneId`: global zone ID (page 285)

Specifies the primary zone where the cargo should be delivered.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].additionalUnloadZones.[n]`: regular zone ID (page 285)

Specifies an additional zone where the cargo can be delivered.

Default: blank; an invalid zone ID or a zone ID that is not on the same map as the `globalZoneId` triggers a ModMapError (page 287).

The `globalZoneId` property specifies the global zone ID for the primary delivery zone. If the player is allowed to deliver the cargo to other zones instead, these are specified with regular zone IDs in `additionalUnloadZones`.

Tip: `additionalUnloadZones` is useful for making a pair of delivery zones on either side of a blocked passage.

If there are multiple delivery goals in the same stage, all of them get the same yellow delivery icon on the navigation map. The player can select each goal description in the objective details to center the navigation map on that goal's primary delivery zone. However, the player has no way to determine which additional delivery zones apply to which goals. Consider making all deliveries share the same set of additional delivery zones or otherwise keep things simple for the player.

If multiple cargo delivery goals in a stage have the same primary delivery zone, the game combines them all under a single goal description ([UI Desc](#)). In this case, you should use the same goal description for all such goals.



If [Manual Unloading Settings](#) (below) is `null`, the player can only complete the delivery while the required cargo is packed on her truck or attached trailer.

Manual Unloading

`objectiveSettings.typeSettings.objective.name.Stages.[n].actions.zoneToFill.[x].Manual Unloading Settings.`
[Interaction zone Uid:](#): global zone ID (page 285)

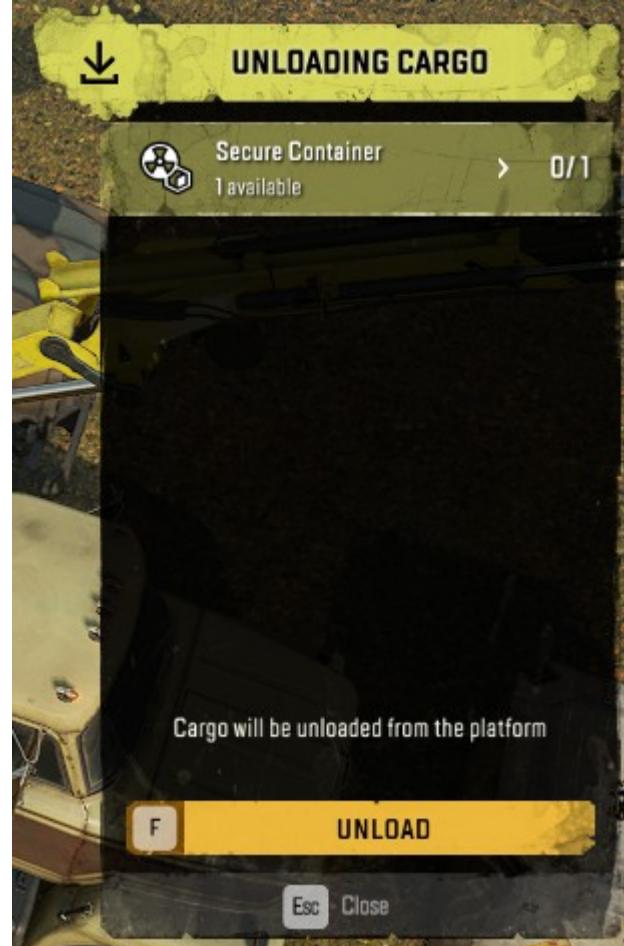
Specifies the zone where the the truck must be in order to deliver unloaded cargo.

Default: blank; an invalid zone ID or a zone ID that is not on the same map as the [globalZoneId](#) triggers a ModMapError (page 287).

In order to complete the cargo delivery goal, the player's truck must be in the interaction zone ([Interaction zone Uid:](#)), **and** the cargo must be unpacked and within the **primary** delivery zone ([globalZoneId](#)).

Bug: When manual unloading is required, the zones in [additionalUnloadZones](#) are made visible as if they are additional **delivery** zones, but they offer the cargo management dialog as if they are additional **interaction** zones. However, these zones don't actually work for either function: cargo unloaded in them is not recognized as deliverable, and their cargo management dialog never recognizes when the cargo is correctly placed in the delivery zone. So keep [additionalUnloadZones](#) blank when using manual unloading.

The below screenshots show the cargo management dialog before and after the cargo is correctly positioned in a manual delivery zone. Your milage may vary, but I found the grammar of the first dialog extremely confusing until I noticed the tiny red period that ends the first sentence.



The delivery zone is made visible in the driving view while the objective is activated.

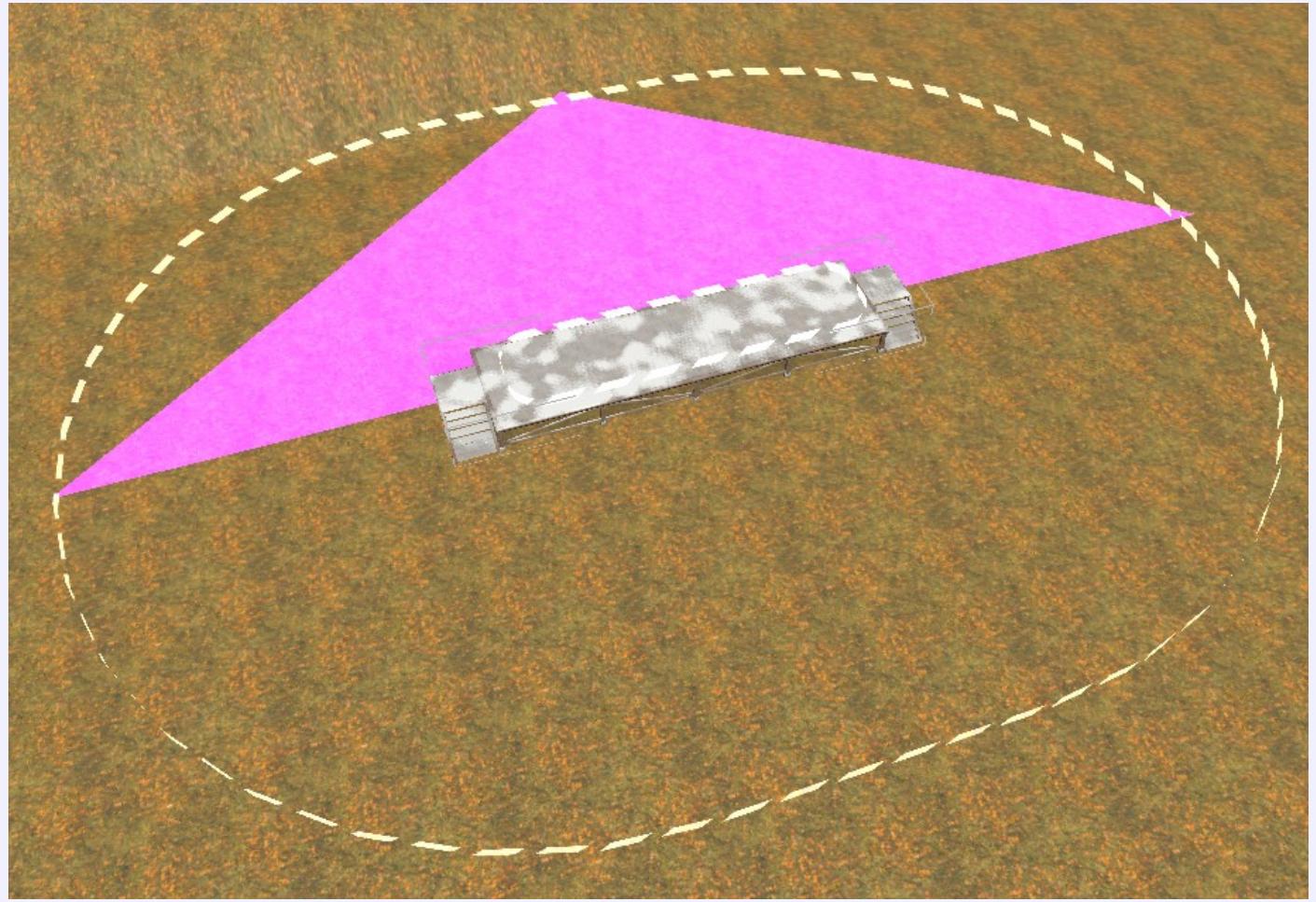
Warning: The interaction zone ([Interaction zone Uid:](#)) is not visible in the driving view (unless the zone has another purpose that makes it visible).

Tip: Saber recommends that the interaction zone entirely surrounds the delivery zone so that as soon as the player finishes unloading the cargo, she can make the delivery from the cargo management dialog offered by the invisible interaction zone.

Tip: The interaction zone should be large enough so that the delivery can happen even after a long stretch with a big crane, but small enough so that the delivery zone is obvious when the cargo management dialog is offered. I find that the longest crane stretch for both US and Russian trucks is about 13.5 meters.

Appropriate interaction zone dimensions can thus be calculated as follows:

- interaction zone **Length** = delivery zone **Length** + 27
- interaction zone **Width** = delivery zone **Width** + 27
- interaction zone **Rounding radius** = delivery zone **Rounding radius** + 13.5



If the player can drive into the delivery zone, she can potentially make the delivery by simply unpacking the cargo within her truck without actually unloading it.

Tip: Place the delivery zone on a raised platform (such as the various **platform** models) that the truck cannot climb onto.



The interaction zone and delivery zone are both shown on the navigation map when the objective is being tracked or when the objective details are viewed.



Tip: To avoid a pile-up of names and icons, set the interaction zone locator's **Is Visible on Minimap** property to **False**.



Manual unloading goes well with cargo that is spawned for the stage and thus requires manual loading. See page 340 for information about cargo spawning.

Required Truck

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].Truck need to visit Uid:`: text

Specifies a truck ID. The player can only complete the cargo delivery goal while driving this truck.

Default: blank; no particular truck is required.

Note that the ID refers to a specific truck, not a truck type. If a truck type is put in this field, then it won't match a truck ID, and so no truck can make the delivery.

The specified truck should not be reserved for another goal. Reserved trucks are described on page 90.

Tip: If a truck is reserved, use the `Blocker Objectives` property to ensure that it is rewarded before offering an objective that requires the player to drive it.

Bug: The game never tells the player which truck is required, or even that a specific truck is required at all. Instead, when the player arrives at the delivery zone with the cargo in the wrong truck, the game either doesn't offer the cargo management dialog, or it allows the player to select the cargo to deliver and then ignores the keypress to deliver it. You should tell the player in the `UI Desc` which truck to use.

Bug: If the player ever visits a garage with that specific truck, its truck ID is lost. I.e. when the player leaves the garage, the game doesn't restore the original truck, but instead spawns a new truck of the same type. So to use `Truck need to visit Uid` with any safety, you need to make sure that no garage is unlocked. (Even if the truck is unable to drive to a garage, the player could still recover the truck to a garage.) Note that the truck ID is **not** lost if the truck is moved through a gateway or the player recovers the truck to the start zone.

Tip: It's probably safest to never use this property.

If the `Truck need to visit Uid` property specifies the ID of a trailer, the delivery goal cannot be completed, even if the load is on the required trailer.

Multi-Stage Models

Multi-stage models (page 131) can change state in response to delivery of cargo. This is very similar to model state changes in response to completion of objective stages (page 328), where they are explained in detail.

Most multi-stage models have the suffix `_objective` and so can be easily found in the model asset list. Reference information for the available multi-stage models is on page 410.

Click the ▼ to the right of [Model Building Settings \(Depend On Cargo\)](#) to connect a multi-stage model to the player's progress in delivering cargo.

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].Model Building Settings (Depend On Cargo).levelName`: text; editable in Region Settings Editor; read-only in Zone Settings Editor

Specifies the name of the map that contains the multi-stage model.

Default: blank in Region Settings Editor; an invalid map name triggers a ModMapError (page 287).

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].Model Building Settings (Depend On Cargo).modelTag`: text

Specifies the ID of the model that changes state when cargo is delivered (page 131).

Default: blank; triggers a ModMapError (page 287).

An objective can change only a single model in response to the delivery of cargo. Also, a model ID cannot be used by more than one objective, nor by more than one property within an objective. An invalid model ID is ignored.

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].Model Building Settings (Depend On Cargo).stagesProgress`: 

Specifies a model stage name to add as an association with an objective stage.

The easy way to get model stage names is by checking the reference on page 410. The hard way is to open the `initial.pak` archive in the SnowRunner installation directory (page 426), find the model XML file under `[media]/classes/models` (or in one of the `[media]/_dlc` directories), and then find the name attributes specified in each `<Subset>` tag. But you shouldn't need to use the hard way unless you want to use a newer model that I haven't yet added to my reference.

Once a model stage name is added, you can associate it with cargo being delivered or not delivered.

`objectiveSettings.typeSettings.objective name.Stages.[n].actions.zoneToFill.[x].Model Building Settings (Depend On Cargo).stagesProgress.model stage name`: integer

Specifies the cargo delivery state to associate with the named model stage.

Default: 0; the model is initialized to the named model stage when the map is started and when the objective is restarted.

Cargo is either delivered or not delivered, so it can only switch a model between two states. The integer value representing the cargo state must therefore be either 0 or 1. A model stage name associated with a value of 0 sets the initial state of the model. A model stage name associated with a value of 1 specifies the state of the model after cargo is delivered.

stagesProgress		+
build_stage_0	0	-
build_complete	1	-

If you don't associate any model stage with objective stage 0, then the game initializes the model to its default stage (the first stage listed in the reference).

Bug: If no model stage name is associated with objective stage 0, the model's landmark may not be rendered on the navigation map in the correct default stage.

See also the notes about potential model collisions on page 330.

Simultaneous Activation of Multi-State Models

An objective can include a multi-stage model tied to its stages and also one or more multi-stage models tied to cargo deliveries. If two multi-stage models are simultaneously updated by the completion of a cargo delivery that completes a stage, both are animated simultaneously. If both have an animation camera, the camera follows only the model animation tied to stage completion while the other animation progresses in the background.

Spawn Cargo for Stage

Loose cargo can be spawned when a particular objective stage begins. This is the best mechanism for encouraging the player to manually load cargo for a cargo delivery goal. Since the cargo is respawned if the objective is restarted, the player can never get stuck without sufficient cargo.

Tip: Cargo is best spawned in the same stage as an associated cargo delivery goal. For details, see the information about cargo highlighting and recommendations starting on page 343.

To spawn cargo when a stage is started, click the to the right of `spawnCargoOnStageActive`. You can spawn cargo in as many zones as you like.

<code>spawnCargoOnStageActive</code>	
<code>[0]</code>	
<code>Don't show reward popup</code>	<input type="checkbox"/>
<code>zone</code>	<code>level_test manual_1</code>
<code>cargos</code>	
<code>[0]</code>	
<code>name</code>	<code>CargoServiceSpareParts</code>
<code>Amount</code>	<code>1</code>
<code>Should be discovered by metalldetector</code>	<input type="checkbox"/>

Bug: There is a checkbox for `Don't show reward popup`, but it doesn't belong here and doesn't do anything.

`objectiveSettings.typeSettings.objective name.Stages.[n].spawnCargoOnStageActive.[x].zone`: global zone ID (page 285)

Specifies the zone where the cargo is spawned.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

Unlike a manual cargo loading zone (page 305), there is no confirmation dialog to delete other cargo in the zone. Instead, cargo fails to spawn if other cargo is in the way.

Bug: The game does not indicate to the player when cargo failed to spawn, nor does it indicate **where** it failed to spawn, so the player has little hope of figuring out how to correct the situation. If the cargo spawns near where the stage starts (e.g. the task or contest offer zone), the player **might** figure out that the stray cargo she can see is obstructing the spawn of cargo she can't see, but then again she might not.

Interestingly, if a truck is in the spawn zone, it doesn't prevent cargo from spawning. If the cargo intersects a truck, it is displayed as a "ghost", and only becomes real when the truck is moved out of the way. The ghost cargo is shown on the map the same as if it is real.

Tip: Spawn cargo only in an area where it is highly unlikely that the player will drop (other) loose cargo. Make the spawn zone's length and width very small or perhaps 0, although cargo can obstruct even a zero-size zone if the cargo is close enough.

`objectiveSettings.typeSettings.objective name.Stages.[n].spawnCargoOnStageActive.[x].cargos.[0].name`: text with dropdown menu for common choices

Specifies the cargo type that is spawned.

Default: `CargoServiceSpareParts`.

When editing the `cargos.[0].name` property value, you can either type in a name or select a name from the dropdown menu. Note that the menu is missing some names, so you'll need to type those in by hand. Reference information for cargo types is on page 396.

Only one cargo item can be effectively spawned in a zone since any others will be blocked by the first. The `Amount` is hard-coded to 1, and there is no point to adding more entries to `cargos` after [0].

The game works best when the spawned cargo types match the cargo delivery goals for the same objective stage and when there is no other source for the same cargo type. In particular, nothing stops the player from ignoring the spawned cargo and instead delivering some other cargo of the same type. If spawned cargo is not needed, it remains on the map after the stage is completed.

If the checkbox next to `Should be discovered by metalldetector` is enabled, then the spawned cargo cannot be seen on the map until discovered by the metal detector. Details are on page 348.

Cargo Spawn Location

Caution: Cargo is spawned with a different orientation than cargo that is generated in a manual loading zone (page 306)

If we assign a “forward” direction for cargo based on how it is loaded on the truck, then cargo is spawned in the target zone with the cargo pointing in the same direction as the zone orientation. I.e. long goods are oriented with the long end parallel to the zone’s direction.

Cargo Spawn Height

The game attempts to spawn cargo just above ground level. If the ground is hilly, it tries to generate the cargo high enough to not be inside the ground. The zone’s `Height` property has no influence on the cargo spawn height.

Bug: Unlike a manual cargo loading zone (page 305), spawned cargo is not placed above any conflicting model.

Logs

The height heuristic for generated cargo performs poorly with the standard log trestle (`logs_scavange_02`), but the game compensates by generating a log 2 meters higher than it would for other cargo. This gives the log room to drop into place.

Bug: If the player restarts the objective, there is not enough room to spawn a new log above the old one in the log trestle. Even worse, the game turns the **old** log on the trestle into a ghost image that cannot be touched, so the player now has no logs available to her.

If a log is instead spawned without a trestle, the old log has enough time to become “real” as it falls. However, the old log(s) remain as distracting ghosts. Therefore, I recommend against spawning logs as part of a stage.

Highlighting of Cargo Sources and Destinations

Cargo, cargo loading zones, and cargo delivery zones are highlighted and/or recommended in various ways when the player views the details for an objective with a cargo delivery goal or when the player is tracking the objective. Understanding this highlighting can help you decide how simple or complex you want to make the cargo situation on your map.

I’ve tried to break down my description into understandable pieces, but the topic is quite complex nonetheless. I first describe highlighting and recommendations when the metal detector is not required. A separate section then describes the differences when spawned cargo requires the metal detector.

Revealed, Unrevealed, and Invisible Cargo Sources

Before we can talk about highlighting, let’s review the normal visibility of cargo on the map:

- ‘Revealed’ cargo has a 3-D shape on the map with an icon and label. If unpacked cargo is in a revealed area of the map, it is revealed.



- ‘Unrevealed’ cargo has a 3-D shape on the map, but has no icon or label. Unpacked cargo that is in the gray, unrevealed part of the map is unrevealed.



- ‘Invisible’ cargo cannot be seen on the map at all. Cargo that is in a cloaked part of the map is invisible.

A truck or trailer can be ‘revealed’, ‘unrevealed’, or ‘invisible’, similar to unpacked cargo. Cargo that is packed on a truck or trailer is not drawn on the navigation map, but the player can click on a revealed truck or trailer to see what cargo is packed on it.

A cargo loading zone doesn't have a 3-D shape on the map, but it can be 'revealed' with an icon and label if it is in a revealed area of the map and **Is Visible on Minimap** is not **False**. Otherwise it is invisible.

Whether a cargo source is revealed, unrevealed, or invisible actually makes no difference to whether it is highlighted, but it is interesting to consider exactly how highlighting breaks the conventions for unrevealed cargo.

Highlighting of Cargo Sources

A cargo source can only be highlighted only if it has an associated cargo delivery goal that is eligible for highlighting. A **cargo delivery goal is eligible for highlighting** if

- an objective is being **tracked**, and the cargo delivery goal is in the **current stage** of that objective, or
- details are being viewed for an **activated objective**, and the cargo delivery goal is in the **current stage** of that objective, or
- details are being viewed for an **offered objective**, and the cargo delivery goal is in the **first stage** of that objective.

A **cargo loading zone is highlighted** as a cargo source if

- it has cargo **in stock** of the same type as a cargo delivery goal that is eligible for highlighting, and
- the player's truck and attached trailer are not already carrying enough cargo to satisfy the goal.

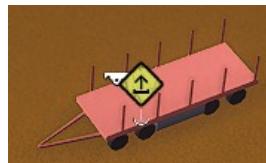
An **unattached trailer is highlighted** as a cargo source if

- it is carrying packed cargo of the same type as a cargo delivery goal that is eligible for highlighting, and
- the player's truck and attached trailer are not already carrying enough cargo to satisfy the goal.

Unpacked cargo is highlighted as a cargo source if

- it is the same type as a cargo delivery goal that is eligible for highlighting, and
- **cargo of the same type was (or will be) spawned** in the same stage as the goal.
- (It doesn't matter whether the player already has enough cargo to satisfy the goal.)
- (It also doesn't matter whether the cargo was actually spawned by the objective.)

A highlighted cargo source is marked with a yellow 'load' icon on the navigation map, but only if it is not a zone for which **Is Visible on Minimap** is **False**.



The highlight and label can be seen on the map even if the cargo source is otherwise unrevealed or invisible, although the 3-D shape is not drawn for an invisible source.



If a trailer is highlighted, the player can click on it to find out what cargo is packed on it, even if the trailer is otherwise invisible or unrevealed.

A highlighted cargo source is also marked with a similar icon in the driving view, whether or not **Is Visible on Minimap** is **False**.

A truck or attached trailer is never highlighted, even if it is carrying packed cargo of an appropriate type.

Bug: Unpacked cargo that was spawned by the current objective stage is highlighted, as is other unpacked cargo of the same type, even if that type is not required by a current delivery goal. Spawning cargo that isn't needed may also disrupt highlighting of unpacked cargo that is required, although at this point it gets so confusing that I'm not sure anymore.

Highlighting of Cargo Destinations

The **delivery zones** of a cargo delivery goal are highlighted as destinations **on the navigation map** if

- the goal is eligible for highlighting.

However, the delivery zones of a cargo delivery goal are highlighted as destinations **in the driving view** if

- the goal is eligible for highlighting, and
- the player's truck and attached trailer are **carrying enough cargo to satisfy the goal**.

Recommended Cargo Sources

When an objective's details are viewed, the game displays **at least** the goals for the current stage (or the first stage for an offered objective), but it may also display the goals for one or more **later stages**. For each displayed goal, the game may recommend one or more cargo sources for that goal.

Each cargo delivery goal recommends either unpacked cargo sources or a single cargo loading zone, never a mix of the two source types.

A cargo delivery goal recommends **unpacked cargo sources** if

- details are being viewed for an **activated** objective, and
- the cargo delivery goal is listed in the objective details, and
- cargo **was (or will be) spawned** in the same stage as the goal and with the same cargo type.

A cargo delivery goal recommends a **cargo loading zone** if

- details are being viewed for an **activated or offered** objective, and
- the cargo delivery goal is listed in the objective details, and
- the goal is not recommending unpacked cargo sources (as above).

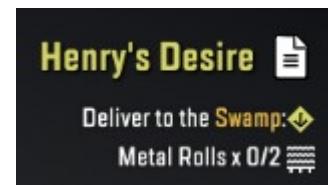
In other words, a cargo delivery goal can recommend a cargo loading zone if cargo of the same type was not (or will not be) spawned in the same stage or if details are being viewed for an unactivated objective.

Recommended Unpacked Cargo Sources

An unpacked cargo source is recommended if

- a cargo delivery goal **recommends unpacked cargo sources**, and
- the unpacked cargo has the same cargo type as the goal.

Multiple unpacked cargo sources can be recommended at a time, and the objective details list each recommended unpacked cargo source without an associated quantity. Selecting a recommended cargo source from the list centers the map on that unpacked cargo. Note that the number of recommended cargo sources may be different the number required by the associated cargo delivery goal.



Unpacked cargo can be recommended even if it is not highlighted. This occasionally means that the recommendation centers the map on a cargo source that is invisible or unrevealed and thus unlabeled.

The player can often benefit by zooming out and/or panning the map to find any highlighted cargo loading zones or detached trailers that aren't being recommended.

Tip: The recommended cargo sources make the most sense when exactly the right amount of cargo is spawned as is required for delivery, and there are no other sources for that cargo type.

If the same cargo type is required for more than one delivery goal, then each unpacked cargo of that type may be listed once for each goal. E.g. two goals for metal rolls and three unpacked metal rolls on the map results in six entries for metal rolls in the objective details. This is a bit confusing, but ultimately harmless.

Recommended Cargo Loading Zone

A cargo loading zone is eligible for recommendation if

- a cargo delivery goal **recommends a cargo loading zone**, and
- the cargo loading zone carries cargo of the same type as the goal (whether or not that cargo type is in stock).

If a single cargo loading zone is eligible for recommendation, it is recommended. If more than one cargo loading zone is eligible, **only one of them is recommended**. This is not necessarily the zone closest to the player's truck. I haven't figured out the pattern; it may be entirely random.

When a cargo loading zone is recommended, the objective details list the **quantity** of cargo **required for delivery** on a single line. Selecting that line centers the map on the single recommended cargo loading zone.



Bug: If `Is Visible on Minimap` is `False` for the recommended cargo loading zone's locator, then the recommendation does not center the map on anything, even if other eligible cargo loading zones are visible on the navigation map.

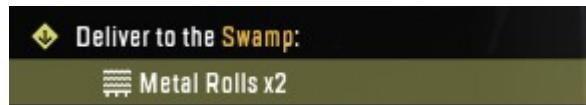
A cargo loading zone can be recommended even if it is not highlighted. This occasionally means that the recommendation centers the map on a loading zone that is invisible or unrevealed and thus unlabeled.

The player can often benefit by zooming out and/or panning the map to find any highlighted cargo loading zones or detached trailers that aren't being recommended. There may also be unpacked cargo that would satisfy the goal that is left unhighlighted and unrecommended.

Behavior when No Cargo is Recommended

If a cargo delivery goal recommends unpacked cargo, but no unpacked cargo sources meet the requirements, or if the cargo delivery goal recommends a cargo loading zone, but no cargo loading zone meets the requirements, then no cargo source is recommended.

If no cargo source is recommended, the objective details list the quantity of cargo required for delivery on a single line, even if the cargo delivery goal prefers to recommend unpacked cargo. However, selecting that line does not center the map on any cargo source.



Behavior when Multiple Goals Share a Delivery Zone

If multiple cargo delivery goals in a stage have the same primary delivery zone, the game combines them all under a single goal description (page 333). This may result in a mix of unpacked cargo and cargo loading zones being listed below the goal description, but remember that these are still for different goals and follow the rules for each goal as stated above.



Bug: Cargo loading zones cannot be recommended if any unpacked cargo is recommended. Selecting the unspawned cargo type fails to center the map on any source.

Tip: The recommended cargo sources work best when cargo is spawned for all cargo delivery goals in a stage or for none of them.

Cargo Sources when Spawning Cargo Requires the Metal Detector

The **Should be discovered by metalldetector** property has independent effects for each cargo type:

- If **Should be discovered by metalldetector** is **False** for all spawned cargo items of a particular type, then the visibility, highlighting, and recommendations for that type are handled as in the sections above.
- If **Should be discovered by metalldetector** is **True** for any spawned cargo items of a particular type, then the visibility, highlighting, and recommendations for that type are handled as below.

If **Should be discovered by metalldetector** is **True** for a spawned cargo item, I abbreviate that item as “MD cargo”, and its type is an “MD cargo type”. There can also be non-MD cargo (spawned or unspawned) with an MD cargo type.

Depending on the player’s progress, the MD cargo type can be “hidden”, “partially hidden”, or “revealed”. This is best thought of as a status that applies to all cargo in that type, and the status remains until some event changes it.

When MD cargo is spawned, all unpacked cargo of its MD cargo type becomes “hidden”. This applies to all spawned and unspawned cargo of that type, including spawned cargo for which **Should be discovered by**

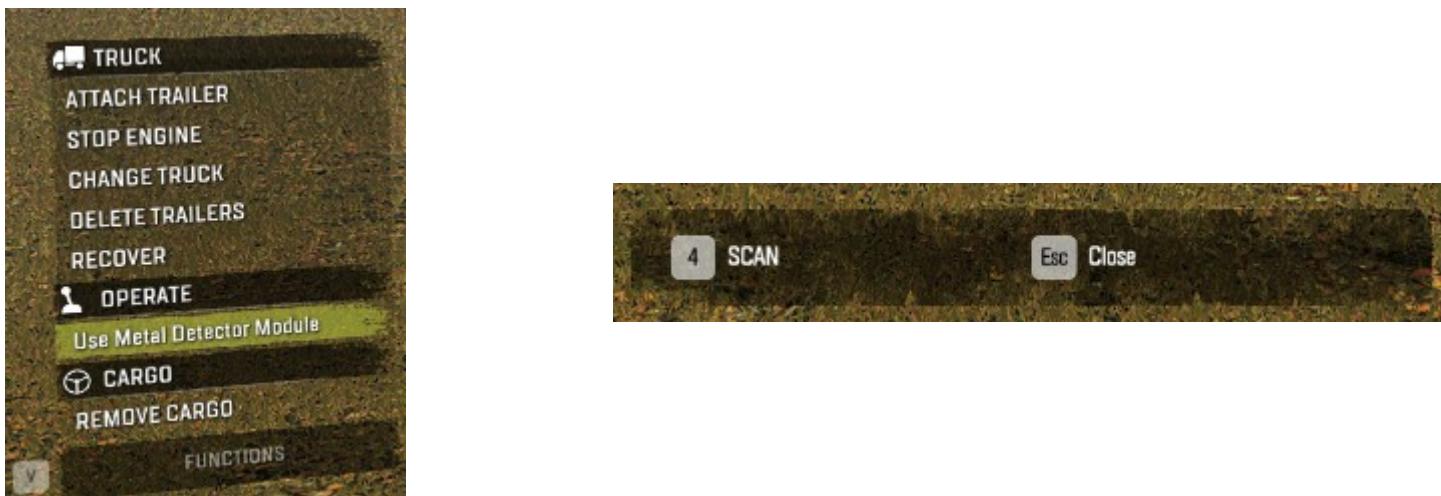
metallodetector is **False**. It also applies to **future** unpacked cargo that appears on the map, and continues to apply until the game removes the “hidden” status from the MD cargo type.

Hidden cargo is completely removed from the navigation map. It is also not highlighted on the map or in the driving view. Although it is still recommended and thus listed in the objective details, selecting one of the recommendations does not center the map on its location. Hidden cargo does remain visible and manipulable in the driving view as usual.

When the player is tracking the relevant objective and uses a truck’s metal detector within about 50 meters of MD cargo, the metal detector beeps three times, and the MD cargo is detected. The metal detector does nothing if the objective is not being tracked or if it detects only non-MD cargo, even if it is of the MD cargo type.

To clarify through anthropomorphism: if any cargo was spawned with **Should be discovered by metallodetector** enabled, then these are the bad guys who throw a cloak over all cargo of the same type, even innocent bystanders. The only way to remove the cloak is by using the metal detector to detect the bad guys. Using the metal detector on the innocent bystanders does nothing.

BTW, it can be difficult to decipher how to use the metal detector. When the metal detector is installed, you can operate it from the functions menu on the left. A useless confirmation then asks if you want to scan or not. The metal detector folds out and makes some noises. If it folds back up without further ceremony, your truck isn’t close enough to the cargo. If it beeps three times and cargo icons show up in the driving view, you found something.



If the player detects **any** MD cargo of a cargo type but has **not yet** detected **all** MD cargo of that type, then the MD cargo type changes from “hidden” to “partially hidden”:

- Unpacked cargo of the MD cargo type is allowed to be drawn on the navigation map, but not labeled.
- Unpacked cargo of the MD cargo type is allowed to be highlighted in the driving view, but still isn’t highlighted on the map.

- Selecting a cargo source from the recommended list in the objective details centers the map on it (even though it isn't highlighted or labeled).

I.e. once the player has used the metal detector to find any MD cargo item, it becomes much easier for her to find the remaining MD cargo.

If the player has detected **all** MD cargo of a cargo type, then the MD cargo type changes to “revealed”, and all cargo of that type regains its normal revealed, highlighted, and recommended statuses.

If the player completes the objective stage after detecting at least some MD cargo, but not all of it, this also changes the MD cargo type to “revealed”.

Bug: If the player completes the stage without using the metal detector to detect any MD cargo, then the MD cargo type remains “hidden”. This status can only be changed if there is a later metal detector stage with the same cargo type (but the player could again complete the stage without using the metal detector).

Tip: As with regular spawned cargo, MD cargo makes the most sense when there are no other sources for that cargo type. In addition, since the MD cargo type may still be hidden at the end of the stage, that cargo type should not become available in the future, either, unless as part of another metal detector stage.

If two stages are simultaneously active with the same MD cargo type, then any change to visibility or highlighting by one objective affects all cargo of that type as described above. You probably want to avoid doing this.

Should be discovered by metaldetector has no effect on the visibility or highlighting of cargo loading zones and packed cargo on unattached trailers.

Truck Delivery Goal

A stage can require the player to deliver one or more particular vehicles (trucks or trailers) to specified zones. This section initially discusses only trucks. Any differences with trailers is described on page 354.

To create a truck delivery goal, click the green **+** to the right of **truckDelivery**. This adds one truck delivery goal.

truckDelivery	
0	
Truck Uid	clean_truck_1
Truck UI Name	The <y>Dirty</y> Truck
afterStageFinished	REWARD
Ui Desc	Deliver to the <y>Quest Zone</y>:
globalZoneDeliveryId	level_test quest_1
additionalDeliveryZones	

Truck ID

`objectiveSettings.typeSettings.objective name.Stages.[n].truckDelivery.[x].TruckUid`: text

Specifies the truck ID (page 88) of a truck to be delivered.

Default: blank; an invalid truck ID triggers a ModMapError (page 287).

When a truck's ID is used in a truck delivery goal, that truck is reserved (page 90), and the player cannot drive it unless and until the objective awards it to the player. Since a player cannot drive the specified truck, the player can only complete the truck delivery goal by towing it with another truck.

If the player restarts an objective with a truck delivery goal, the reserved truck respawns at its original location. If another vehicle is in the way, the truck remains “virtual” until space is cleared for it.

Bug: If the truck delivery goal is for a DLC truck that the player doesn't own, then the truck won't initially appear on the map. But if the player restarts the objective with the delivery goal, it **does** spawn at its designated location.

A particular truck ID can be reserved only by a single objective, although it can be reserved by multiple goals in one or more stages of that objective.

Bug: The reserved truck should have its `Locked` property set to `True`. Otherwise the player can change into the truck as soon as all goals that reserve the truck are complete, which opens the possibility of various glitches and exploits. The one exception to this rule is a particular scenario involving the change truck goal, described on page 357.

Truck Name

`objectiveSettings.typeSettings.objective name.Stages.[n].truckDelivery.[x].Truck UI Name`: UI text (page 282); formatting tags are supported; max ~35 characters

Specifies a name for the truck to use on the navigation map, in tracking information, and in the objective details.

Default: blank; the game refers to the truck by its type instead.



If a truck ID is used by multiple goals and/or multiple stages in an objective, it should have the same name for all of them.

Goal Description

`objectiveSettings.typeSettings.objective name.Stages.[n].truckDelivery.[x].Ui Desc`: UI text (page 282); formatting tags are supported; max ~125 characters

Specifies the text that describes the goal for the player. This text is displayed when the objective is tracked or when the objective details are viewed.

Default: blank; an error message is displayed instead.



Delivery Zone

`objectiveSettings.typeSettings.objective name.Stages.[n].truckDelivery.[x].globalZoneDeliveryId`: global zone ID (page 285)

Specifies the zone where the truck should be delivered.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

If multiple truck delivery goals in a stage have the same delivery zone, the game combines them all under a single goal description (`UI Desc`). In this case, you should use the same goal description for all such goals.



`objectiveSettings.typeSettings.objective name.Stages.[n].truckDelivery.[x].additionalDeliveryZones.[n]`: regular zone ID (page 285)

Specifies an additional zone where the truck can be delivered.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

Bug: If any IDs are specified in `additionalDeliveryZones`, then the goal cannot be completed (even if the truck is delivered to all zones simultaneously).

Stage Complete

`objectiveSettings.typeSettings.objective name.Stages.[n].truckDelivery.[x].afterStageFinished`: dropdown menu

Specifies what to do with the delivered truck when the stage is complete.

Default: **REWARD**.

The `afterStageFinished` values have the following meanings:

- **DO NOTHING**: Nothing special happens at the end of the stage.
- **DETACH**: The truck is detached from the winch at the end of the stage. This has no effect if the truck is not attached or if a different truck is attached at the end of the stage.
- **DESTROY**: The truck is removed from play at the end of the stage.
- **REWARD**: The truck is awarded to the player at the end of the objective.

Bug: It would make a lot more sense to detach or destroy the truck when the **goal** is complete, rather than when the **stage** is complete.

Unlike the other after-stage actions, the truck is awarded when the **objective** is complete. The player can drive the truck once it has been awarded. The truck retains its **Truck UI Name** when selected on the navigation map, although it is simply listed by its type in the list of trucks on the left, and the name is lost the next time the player takes the truck to a garage.

If the individual truck is locked (page 89), it is unlocked when the truck is awarded. However, this does not count as discovery (page 44), and there is no discovery experience bonus.

Bug: The reserved truck should have its **Locked** property set to **True**. Otherwise the player can change into the truck as soon as all goals that reserve the truck are complete, which opens the possibility of various glitches and exploits. The one exception to this rule is a particular scenario involving the change truck goal, described on page 357.

If the truck **type** is locked content (so that it cannot be bought in the garage, page 293), it is discovered when it is awarded, and the player gets a discovery experience bonus. Unlike normal truck discovery, this happens regardless of whether the individual truck was locked or not.

Bug: Since a contest can be restarted after its objective is complete, a truck should never be awarded by a contest. Otherwise, if the player is in the truck when she restarts the contest, the truck would respawn and leave the player hanging in air.

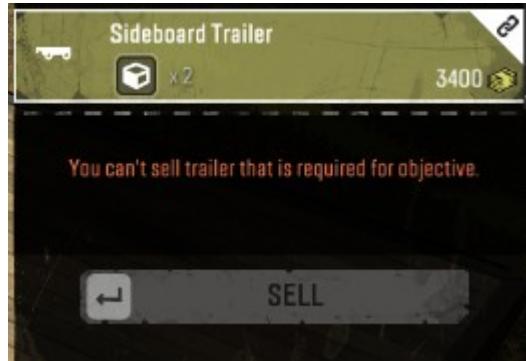
If multiple goals reserve the same truck in a stage, only one of the `afterStageFinished` properties is applied to the truck at the end of a stage. You should set the `afterStageFinished` property the same for all such goals.

Awarded trucks are listed as additional rewards in the objective offer, objective details, and reward window. See page 322 for cautions regarding multiple rewards.

Trailer Delivery

A truck delivery goal can equally be applied to a trailer by specifying the “truck ID” of a trailer. There are a number of trailer types that are only useful for the trailer delivery goal. These are indicated in the trailer reference section on page 394.

The player can complete the goal by towing the trailer by a winch as with a truck, or the player can instead attach the trailer to her truck in the normal manner. In fact, the player can use the trailer as much as she wants, transporting cargo, etc. The only thing she can't do with it is sell it.



Unlike with a drivable truck, the player can only sell an awarded trailer when the entire objective is complete, regardless of whether it was locked.

Bug: If the reserved trailer starts with cargo on it, the player can take the cargo, then restart the objective to respawn the trailer with more cargo again.

Bug: If the player loads cargo on the reserved trailer, the cargo is lost if she restarts the objective. For this reason, a cargo trailer probably shouldn't be used for a trailer delivery goal. That goes double for a contest, since the contest can also be restarted after it is complete.

If the trailer type is locked content (so that it cannot be bought in the trailer store, page 293), it is discovered when it is awarded, and the player gets a discovery experience bonus. Unlike normal trailer discovery, this happens regardless of whether the individual trailer was locked or not.

Truck Repair Goal

A stage can require the player to fully repair **and refuel** one or more particular trucks. A truck repair goal can be combined in the same stage as a truck delivery goal so that the player must repair, refuel, and deliver the specified truck.

To create a truck repair goal, click the green  to the right of `repairTruck`. This adds one truck repair goal.

Truck ID

`objectiveSettings.typeSettings.objective name.Stages.[n].repairTruck.[x].Truck Uid`: text

Specifies the truck ID (page 88) of the truck to be repaired and refueled.

Default: blank; an invalid truck ID triggers a ModMapError (page 287).

When a truck's ID is used in a truck repair goal, that truck is reserved (page 90), and the player cannot drive it unless and until the objective awards it to the player. Since a player cannot drive the specified truck, the player can only complete the truck repair goal by repairing and/or refueling it from another truck. If the specified truck is already fully repaired and refueled when the objective stage starts, the player must still perform a “repair” action on it to complete the goal. “Refuel” is disabled if the target truck is fully refueled, but “repair” is still possible when it is fully repaired. The game may also treat the goal as completed if the truck is fully repaired when the player enters its location via a gateway.

If the player restarts an objective with a truck repair goal, the truck respawns at its original location. If another vehicle is in the way, the truck remains “virtual” until space is cleared for it.

Bug: If the truck repair goal is for a DLC truck that the player doesn't own, then the truck won't initially appear on the map. But if the player restarts the objective with the repair goal, it **does** spawn at its designated location.

A particular truck ID can be reserved only by a single objective, although it can be reserved by multiple goals in one or more stages of that objective.

Bug: The reserved truck should have its `Locked` property set to `True`. Otherwise the player can change into the truck as soon as all goals that reserve the truck are complete, which opens the possibility of various glitches and exploits. The one exception to this rule is a particular scenario involving the change truck goal, described on page 357.

If the **Truck Uid** is set to a trailer, then the goal is impossible to complete because a trailer can never be repaired or refueled.

Truck Name

objectiveSettings.typeSettings.objective name.Stages.[n].repairTruck.[x].Truck UI Name: UI text (page 282);
formatting tags are supported; max ~35 characters

Specifies a name for the truck to use on the navigation map.

Default: blank; the game refers to the truck by its type instead.

Unlike for a truck delivery goal, the truck's name is never displayed by the objective. It is only shown when the player clicks on the truck in the navigation map.



If a truck ID is used by multiple goals and/or multiple stages in an objective, it should have the same name for all of them.

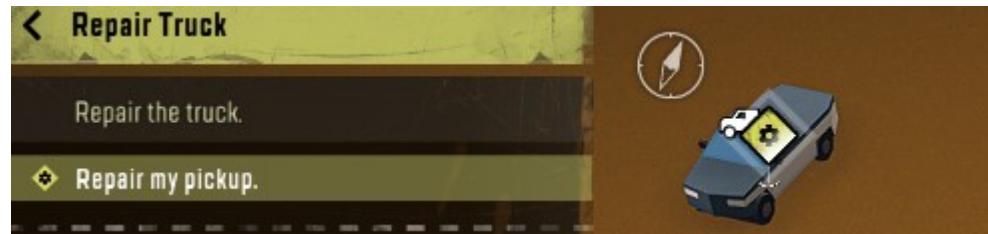
Goal Description

objectiveSettings.typeSettings.objective name.Stages.[n].repairTruck.[x].Ui Desc: UI text (page 282);
formatting tags are supported; max ~125 characters

Specifies the text that describes the goal for the player. This text is displayed when the objective is tracked or when the objective details are viewed.

Default: blank; an error message is displayed instead.

Unlike for a truck delivery goal, the goal description is not followed by the truck name, so you might want to provide your own description of the target truck in the goal's **UI Desc**.



Stage Complete

`objectiveSettings.typeSettings.objective name.Stages.[n].repairTruck.[x].afterStageFinished`: dropdown menu

Specifies what to do with the repaired truck when the stage is complete.

Default: **REWARD**.

The `afterStageFinished` property is exactly the same as described for the truck delivery goal (page 353).

Change Truck Goal

A stage can require the player to change into a specific truck.

To create a change truck goal, click the ▼ to the right of `changeTruck`. A stage can have only one change truck goal.

Truck ID

`objectiveSettings.typeSettings.objective name.Stages.[n].changeTruck.[x].TruckUid`: text

Specifies the truck ID (page 88) of the truck that the player must change into.

Default: blank; an invalid truck ID triggers a ModMapError (page 287).

The goal is completed as soon as the player changes to the designated truck.

The designated truck is not reserved by the change truck goal, but if the truck is reserved (page 90) by a goal in a different objective, then the game throws a ModMapError (page 287). The truck to change into can be reserved by another goal in the **same** objective, as long as it is rewarded to the player before the change truck goal's stage. (Otherwise, the player will not be able to enter the truck and complete the goal.)

If the `truckUid` is set to a trailer, then the goal is impossible to complete because the player can never change to driving a trailer.

If the player restarts an objective with a change truck goal, the truck is not respawned (unless it is reserved for another goal in the same objective).

Bug: If the player ever visits a garage with the truck needed for the change truck goal, its truck ID is lost. I.e. when the player leaves the garage, the game doesn't restore the original truck, but instead spawns a new truck of the same type.

Bug: If the player is already in the designated truck when the goal is activated, the goal is **not** completed. The player must change to a different truck and then change back to complete the goal.

Because of the above bugs, and because the player is not required to **do** anything in the designated truck, the change truck goal really only makes sense as a way to add flavor in a final stage when awarding a truck to the player from a truck delivery goal (page 350) or truck repair goal (page 355). In this case, the truck is guaranteed to have never visited a garage, and the player is guaranteed to not be in the truck when the goal is activated.

Tip: This is one case where it is acceptable (and necessary) to set the **Locked** property to **False** for a reserved truck. The second to last stage of the objective should have only goals that reserve the truck (so the player cannot enter the truck before the end of the stage), and the last stage should have a single goal to change into the truck (so that the objective ends and the truck is awarded as soon as the player enters the truck).

Bug: The player cannot unlock a truck that is specified by a change truck goal. The truck must have its **Locked** property (page 89) set to **False** to allow the player to enter the truck and complete the goal. This bug applies whether the truck was reserved and rewarded or was never reserved at all.

Truck Name

objectiveSettings.typeSettings.objective name.Stages.[n].changeTruck.[x].Truck UI Name: UI text (page 282); formatting tags are supported; max ~35 characters

Specifies a name for the truck to use on the navigation map, in tracking information, and in the objective details.

Default: blank; the game refers to the truck by its type instead.

If a truck ID is used by multiple goals and/or multiple stages in an objective, it should have the same name for all of them.

Goal Description

objectiveSettings.typeSettings.objective name.Stages.[n].changeTruck.[x].Ui Desc: UI text (page 282); formatting tags are supported; max ~125 characters

Specifies the text that describes the goal for the player. This text is displayed when the objective is tracked or when the objective details are viewed.

Default: blank; an error message is displayed instead.

The goal description is always followed by the truck name.



Stage Complete

`objectiveSettings.typeSettings.objective name.Stages.[n].repairTruck.[x].afterStageFinished`: dropdown menu

This property has no purpose.

Default: **REWARD**.

Bug: The `afterStageFinished` property is presented in the Zone Settings Editor, but it has no function for a change truck goal. (No, you can't destroy the truck that the player just changed into.)

Visit Goal

A stage can require the player to visit one or more zones.

To create a visit goal, click the ▼ to the right of `visitAllZones`, then click the + to the right of `zones` to add one or more destination zones. If no members are added to `zones`, the objective cannot be accepted.

Goal Description

`objectiveSettings.typeSettings.objective name.Stages.[n].visitAllZones.zones.[x].Ui Desc`: UI text (page 282); formatting tags are supported; max ~125 characters

Specifies the text that describes the goal for the player. This text is displayed when the objective is tracked or when the objective details are viewed.

Default: blank; an error message is displayed instead.



Destination Zone

`objectiveSettings.typeSettings.objective name.Stages.[n].visitAllZones.zones.[x].globalZoneId`: global zone ID (page 285)

Specifies the zone that should be visited.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

Required Truck

`objectiveSettings.typeSettings.objective name.Stages.[n].visitAllZones.zones.[x].Truck need to visitUid:: text`

Specifies a truck ID. The player can only complete the visit goal while driving this truck.

Default: blank; no particular truck is required.

See the identical property for cargo delivery (page 338) for the long list of cautions and caveats.

Tip: It's probably safest to never use this property.

Bug: If the `Truck need to visitUid` refers to a trailer, the visit goal cannot be completed, even if the designated trailer is attached to the player's truck. Use a truck delivery goal to require a trailer to be brought to a destination.

Seismic Vibrator Goal

A stage can require the player to use a seismic vibrator module in one or more zones.

To create a seismic vibrator goal, click the ▼ to the right of `Seismograph Settings`, then click the + to the right of `zones` to add a one or more destination zones. If no members are added to `zones`, the objective cannot be accepted.

`objectiveSettings.typeSettings.objective name.Stages.[n].visitAllZones.zones.[x].Ui Desc`

`objectiveSettings.typeSettings.objective name.Stages.[n].visitAllZones.zones.[x].globalZoneId`

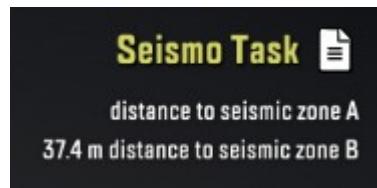
`objectiveSettings.typeSettings.objective name.Stages.[n].visitAllZones.zones.[x].Truck need to visitUid:`

The properties of a seismic vibrator goal are identical to those for a visit goal, described above.

Unlike a visit goal, the destination zones are not made visible in either the driving view or the map. Instead, the distance to each zone is listed when the objective is tracked and the player has a seismic vibrator installed on her truck. The player must then find each destination zone by blind triangulation.

Tip: Don't make the seismic vibrator target zone too small, or the player may get very frustrated trying to maneuver her truck to find it. The built-in campaign maps from Saber appear to set the zone to 20 – 30 meters in diameter, which seems to work well. Putting the zone in an obvious clear area with congested areas around it also helps direct the player to the correct area.

When the player's truck is within a target zone, the goal description is listed in the objective tracking information without embellishment. But when the player's truck is outside a zone, the distance to the target zone is awkwardly prepended to the goal description. I haven't yet found a good way to phrase the goal description to deal with both possibilities.



BTW, it can be difficult to decipher how to use the seismic vibrator. When the seismic vibrator is installed, the objective tracking information in the upper right shows the distance to each destination zone. If you are not in a destination zone, there is no way to operate the seismic vibrator. Once you reach a zone, open the functions menu and press the key indicated at the **bottom** of the screen (not with the other functions on the left).



Living Area Goal

Bug: The living area goal is not yet supported. If you attempt to add a living area goal type, it reverts to **null** when you pack your map (or sooner).

Multiple Goals

Most goal types allow multiple goals of that type to be specified in a stage. In addition, a stage may have goals of multiple types. To complete the stage, the player must complete all goals of all types in any order.

As goals are completed, the game removes them from the objective tracking info. Note that this does not affect the objective details, which always lists all goals, complete or incomplete.

Zone Visibility

Understanding zone visibility can be helpful when debugging your zones. E.g. is a zone not visible because it isn't working, or is that normal? It is also useful to understand which zones can be easily found on the map vs. which ones require more searching.

Driving View

Whether a zone is “available” for a purpose depends on its zone type (page 364) and/or its use by an objective (page 365). If a zone is available for any purpose, its zone perimeter is generally visible in the driving view (except where noted). Otherwise, it is “unavailable”, and it does not show up in the driving view.

If a zone is available in the driving view for multiple purposes, and those purposes prefer different colors of zone perimeter, then the pattern seems to be that blue beats yellow and red, or yellow beats red.

If a zone is available, the zone’s icon is also typically displayed in the driving view above the zone’s center, but only if the player’s **camera** is closer than about 50 meters and the truck is **not** in the zone.

If a zone can display a white icon (based on its type) or a yellow icon (based on an objective), priority goes to a yellow icon. The symbol in a yellow icon is based on the objective details. The symbol in a white icon is (usually) specified by the zone locator properties. If multiple icons of the same color are possible, the game picks one. I haven’t tried to figure out the pattern.

Bug: The game occasionally gets confused and stops displaying a zone’s icon in the driving view. This condition may continue until the map is reloaded.

If **Is Visible On Minimap** is **True**, a white icon for a zone is normally suppressed from the driving view. **Is Visible On Minimap** has no effect on a zone perimeter or on a yellow icon in the driving view. The exception is the hook icon for a paired manual unloading zone, which is always shown in the driving view regardless of **Is Visible On Minimap**.

Navigation Map

Whether a zone is “visible” on the navigation map depends on the its zone type (page 364), the player’s activity around the zone, and/or the zone’s use by an objective (page 365). An unavailable zone is generally not visible on the navigation map, except where noted. If a zone is not visible, it is “invisible”.

If a zone is visible, it is usually listed with white text in the zone list to the left of the navigation map. In certain cases, it is listed with gray text, although that doesn’t necessarily mean anything interesting.

A zone is marked on the navigation map with the highest visibility among all of its purposes. If multiple highlighting icons could be applied, the game picks one.

Regardless of anything else said in these sections, if **Is Visible On Minimap** is **False** for a zone locator, it is never visible on the navigation map or in the list to the left of the navigation map.

Visibility Based on Zone Type

Some zones become visible on the navigation map when they are available and in a revealed area of the map. Other zones change visibility status based on how close the player brings a truck. In this case, the distance is always measured from the center of the truck to the center of the zone.

A **garage entrance** is always available in the driving view with a yellow perimeter. It is invisible on the map until revealed. Even when revealed, it is still listed in gray until the player “discovers” it by driving within 20 meters. The player can only use “recover to garage” after discovering the garage entrance. Observation from a watchtower does not count toward discovering a garage.

If the garage entrance zone is large enough, the player may be able to enter the garage before it is even revealed. In this case, it counts as discovered (so the player can recover to it), but it still isn’t revealed on the map until the player drives closer to its center.

A **garage exit** is always available in the driving view with a red perimeter. It is never visible on the map.

A **recovery zone** is never available in the driving view or visible on the map. If a recovery zone is made visible in the driving view because the zone has another purpose, the zone perimeter may have an exotic color. (Pink?)

A **trailer store, fuel station, crafting zone, or energy generation zone** is always available in the driving view with a yellow perimeter. It is invisible on the map until revealed.

An **automatic cargo loading zone** is usually available in the driving view with a yellow perimeter, but it disappears from the driving view if it runs out of stock of all cargo types. It is invisible on the map until revealed. It remains revealed on the map even if it runs out of stock.

A **manual cargo loading zone** is available only when it is linked to an automatic cargo loading zone. If it is, it is visible in the driving view with a red perimeter. Its icon on the driving map is a loading hook (ignoring whatever icon was specified for the zone). Its icon can be overridden by a yellow icon for an objective, however. If the linked cargo loading zones run out of stock of all cargo types, the manual cargo loading zone disappears from the driving view at the same time as the automatic zone. A manual cargo loading zone is never made visible on the map.

A **service hub** is always available in the driving view with a yellow perimeter, and it is always visible on the map. It is listed in gray until it is in a revealed area of the map, but that makes no difference to anything else.

A **watchtower** is always available in the driving view with a blue perimeter, and it is initially visible on the map. Unlike other zones, the map usually shows only the watchtower's icon. Its name is shown on the map only when the watchtower is selected by clicking on its icon or on its name in the list on the left. The watchtower is listed in gray until it is in a revealed area of the map, but that makes no difference to anything else. When a watchtower is visited, it is removed from the navigation map. It remains available in the driving view, and the player can use its "launch observation" function as many times as she wants.

An **upgrade zone** is initially available in the driving view with a yellow perimeter. However, it only becomes visible on the map when the player drives within 10 meters of it, or when the upgrade zone is revealed by a watchtower. When an upgrade zone is activated, it is removed from the driving view and the navigation map.

Visibility Based on Objective and Goals

The offer zone for an unlocked **task or contest** is available in the driving view with a yellow perimeter. It is "discovered" when the player drives within 10 meters of it, or when the zone is revealed by a watchtower. When a task or contest is discovered, its offer zone is listed under **Task Giver** to the left of the navigation map.

From the time it is discovered, a **contest** is also listed under **Contests** to the left of the navigation map. Its details can be examined even if its offer zone has never been visited. It remains listed under **Contests** until the contest is accepted and completed. Even after completion, however, its offer zone remains visible and listed under **Task Giver**, and the player can visit it again to retry the contest.

When the player visits a **task**'s offer zone and accepts the task, it is listed under **Tasks** to the left of the navigation map. The offer zone remains available and visible under **Task Giver** until the task is completed, at which time it is removed. Likewise, the task itself is listed under **Tasks** until the task is completed.

A **contract** does not have an offer zone, but it may use other zones which are visible as described below.

When an objective is active, is being tracked, or the player is viewing its details, its goals make their zones visible as described below. If the player views an objective's details before accepting it, it makes zones temporarily visible and highlighted as if its first stage were active.

If a **cargo delivery goal** or **truck delivery goal** is in the current stage of any active objective, its delivery zones are available in the driving view with a yellow perimeter. If the objective is being tracked or its details are viewed, then its delivery zones in the current stage are visible on the navigation map and are highlighted with an "unload" icon in a yellow diamond. Cargo sources are visible as described on page 343.

The interaction zones of a **manual cargo delivery goal** in the current stage are also available in the driving view with a yellow perimeter. If the objective is being tracked or its details are viewed, then the interaction zones of the current stage are visible on the navigation map and are highlighted with an "unload" icon in a yellow diamond.

If a **visit goal** is in the current stage of any active objective, its destination zones are available in the driving view with a blue perimeter. If the objective is being tracked or its details are viewed, then its destination zones in the current stage are visible on the navigation map and are highlighted with a magnifying glass in a yellow diamond.

A **seismic vibrator goal** does not make its target zones visible in either the driving view or the navigation map. However, if a zone is already visible in the driving view for some other purpose, its zone perimeter changes to blue. In either case, selecting a zone from the objective details list does not center the navigation map over it or otherwise give a clue to its location.

When the objective details are viewed, zones used in later stages are sometimes listed. This listing does not make those zones visible on the map. Selecting a zone from a future stage centers the map on where it would be if visible.

Regions

One to four maps can be combined into a region, and the player can drive from one map to another via gateway zones. Alternatively, once the player has discovered garage entrances on multiple maps, she can move trucks among maps through the garage.

Contract objectives can span multiple maps in a region, but each map has its own tasks and contests.

It is trivial to design a map that works on its own and can also be used as part of a region. A map can even be used in multiple regions. Certain “global” properties for a map are replaced by new properties when used as part of a region. Because the individual maps within a region can also be played on their own, you can often test a map without needing to repack the region.

A region is edited by the Region Settings Editor. This can be started from the main Editor by clicking the **Region settings** button in the toolbar. The Region Settings Editor has much in common with the Zone Settings Editor (page 276), but because it is not tied to a specific map, it can be usefully started even when no map is open.

Tip: If you’re a power user, you can run the Region Settings Editor independently of the main Editor so that both can be used at once. To do this, create a shortcut for the following command line:

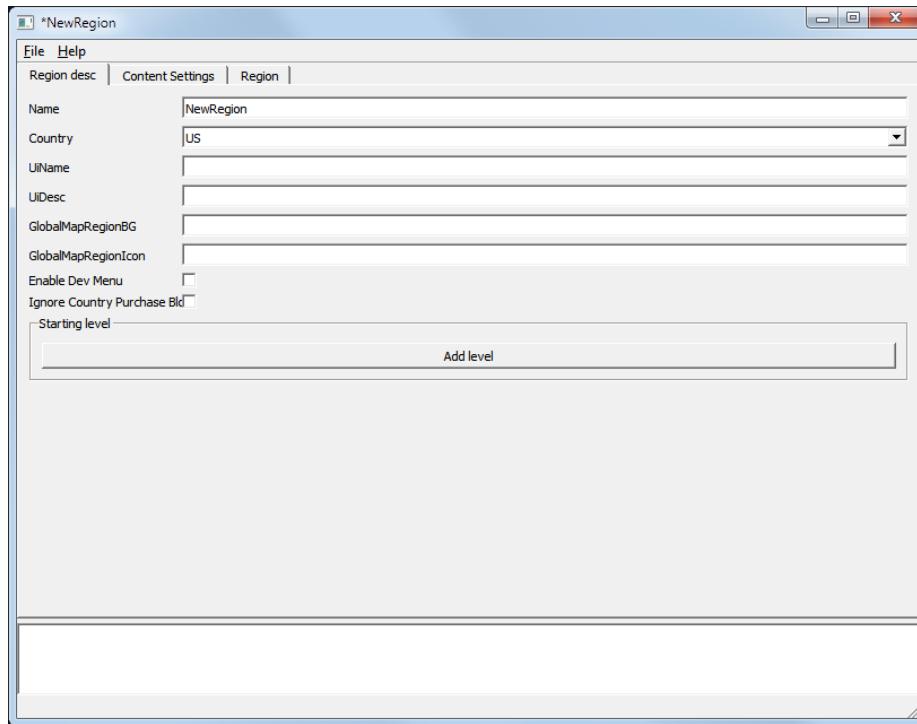
```
"path_to_snowrunner_installation/Sources/BinEditor/ZoneSettingsEditor.exe" -r  
"path_to_media_dir/prebuild" -l "path_to_media_dir/Mods" -u ui/textures/
```

Be sure to re-open a region whenever you pack one of its maps to keep the Region Settings Editor in sync.

Create a New Region

To create a new region, you must first have at least one map. The Region Settings Editor works entirely with packed map data, so pack your map(s) before starting the Region Settings Editor.

The Region Settings Editor starts completely blank except for its menu bar. Select **File → New Region** to start a new region.



Note that there is no menu item to close the Region Settings Editor. Use the red Windows close button in the upper right.

Region Name

Before saving your new region, give it a name (in the **Name** field of the **Region desc** tab). The Region Settings Editor uses the region name when naming directories and files, so the name should not include any of the following characters: <, >, :, “, /, \, |, ?, *. Spaces are fine, although you might want to use a localizable ID for UI text, described below.

Bug: The Region Settings Editor does not warn you when you are about to overwrite a region file that you did not open, so make sure you use a unique name.

If you save before adding any maps to the region, the Region Settings Editor complains, but despite the complaint it does save your changes.



UI Text

A region supports UI text and localization in the same way that a map does (page 282), with the caveat that the region does not have its own `*.str` files. Instead, the region combines the `*.str` files of all maps in the region.

Since the region does not have its own `*.str` files, any UI text at the region level must be localized using one of the `*.str` files of one of the maps. Remember that the region cannot make use of any map changes until the map has been (re)packed.

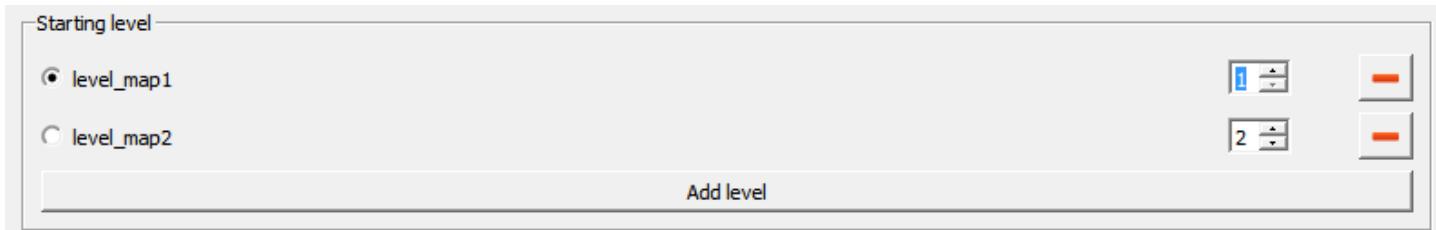
Because the `*.str` of all maps are combined for use in the region, conflicts among the UI text IDs may result in changes to the localization used within the maps.

Add and Arrange Maps

In the middle part of the `Region desc` tab is a box labeled `Starting level`. This is a bit confusing because you can do three things within the box:

- Add and delete maps.
- Choose which map the player starts on.
- Choose how the maps are arranged on the region navigation map.

To add a map, click the `Add level` button. A dialog box opens with a list of **packed** maps (in the `Media/Mods` directory). Select a map and click OK to add it to your region. You can add up to four maps to a region.



Once you have the maximum of four maps, the `Add level` button disappears.

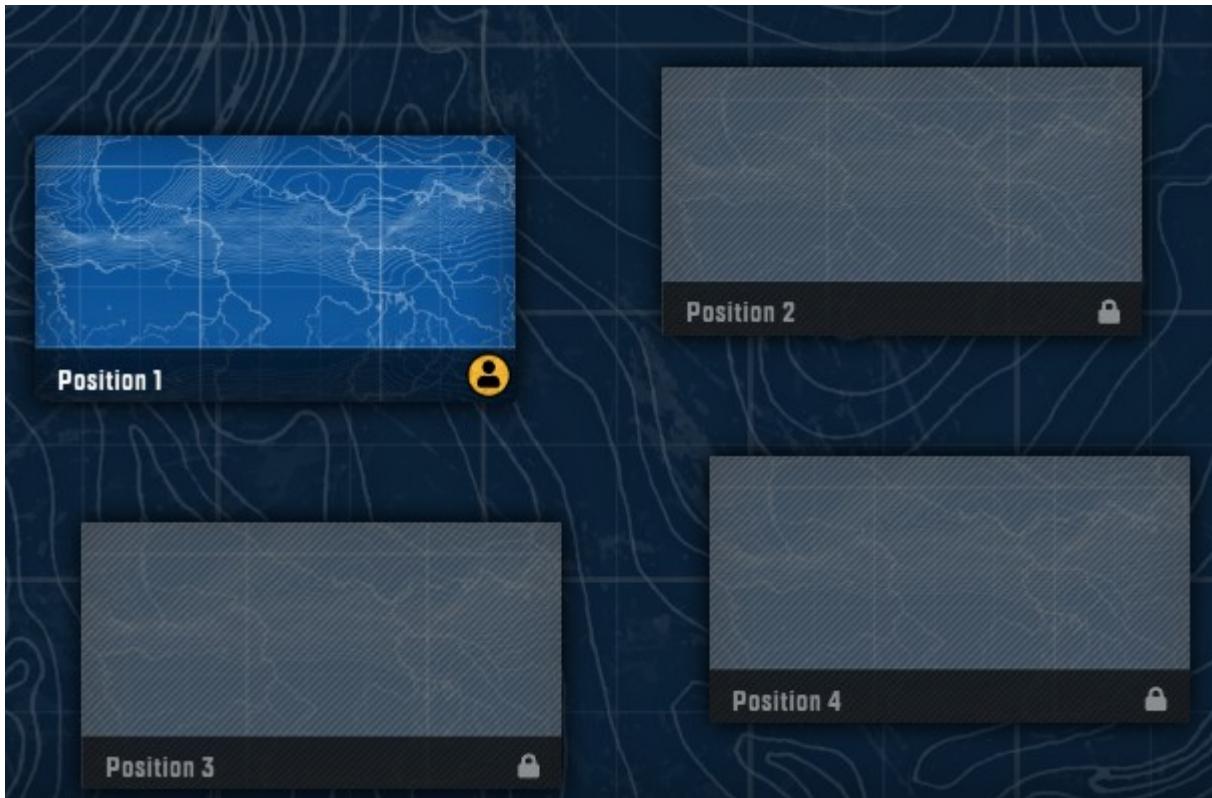
To delete a map, click the red minus `-` to the right of it.

Bug: If you have four maps, deleting a map does not restore the `Add level` button. Save the region and re-open it to get the button back.

To the left of the map names is a set of radio buttons, one for each map. Whichever map is selected is the map that the player starts on. Make sure that there is an active truck on that map for the player to start in.

The game treats active trucks on the non-starting maps as inactive, unlocked trucks (regardless of their **Locked** property value). However, the **Locked** bug for the change truck goal (page 357) still applies, even for an active truck on a non-starting map. There may be other inconsistencies, as well.

To the right of the map names is a integer increment/decrement widget. This sets the position of each map within the regional navigation map. You can click the tiny up and down arrows next to a number or type in a new value from 1 to 4. The Region Settings Editor does not allow duplicates; it adjusts the other map position numbers to remain unique. The position that corresponds to each value is shown below.



Once you've added maps to your region, an additional tab appears at the top of the window for each map. Each tab displays a **read-only** list of the zone types and objectives for the map. This is for your convenience so that you don't have to quit the Regional Settings Editor to consult the Zone Settings Editor, but you cannot make changes here.

Save a Region

Select **File → Save** or press **Ctrl+S** to save your region. The region information is saved in the **Media/prebuild/Region Name** directory.

If you rename your region, future saves will be at a new location corresponding to its new name.

Pack a Region

Select **File → Pack region mod** to pack your region for the game. The region information is saved in **Media/Mods/Region Name.zip**. The pack process is very quick for a region because all it has to do is copy and compress a few files.

When you pack a region, it is also automatically saved.

If you change any of the maps in the region, you must repack the changed maps, then repack the region.

Note that regions never use the **Media/levels** directory.

Open a Region

Select **File → Open region** or press **Ctrl+O** to open an existing region. A dialog opens with a list of saved regions. Select one and click **OK**. You'll need to do this pretty much every time you open the Region Settings Editor.

While a region is open, you can close the region and open a different region using the same procedure.

Region Properties

Region properties are specified in the **Region desc** tab.

Region Name and Description

Name: UI text (page 369); formatting tags are not supported; max ~55 characters

Specifies the name of the region used on the navigation map.

Default: NewRegion

The region name is described in more detail on page 368.

Country: dropdown menu

Specifies the country/continent that the region is part of.

Default: US

If the **Country** is **US**, the region is displayed as being in North America. If the **Country** is **RU**, the region is displayed as being in Russia.

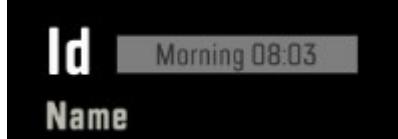
The choice of **Country** may also affect what trucks the player can purchase in garages in the region, depending on the **Ignore Country Purchase Block** property below.

UiName: UI text (page 369); formatting tags are not supported; max ~40 characters (max ~15 on player profile screen)

Specifies the name of the region used in the global view.

Default: blank; no name is displayed

Note that the region name is specified twice: once for the (local) navigation map, and once for the global view. The screenshots below show how it appears when the property values are set to reflect the corresponding property names.



As an example of how the two names are used, the built-in campaign has one region with a **Name** of “Alaska, USA” and a **UiName** of “ALASKA”, and it has another region with a **Name** of “Taymyr, Russian Federation” and a **UiName** of “TAYMYR”.

UiDesc: UI text (page 369); formatting tags are not supported; max ~600 characters

Specifies the description of the region used in the global view.

Default: blank; no description is displayed

Region Images

GlobalMapRegionBG: text

Specifies the image that appears in the background of the global view.

Default: blank; a default image is used.

The image should be 1920×1080 and should be saved as a PNG file in

`prebuild/Region Name/ui/textures/image_name.png`

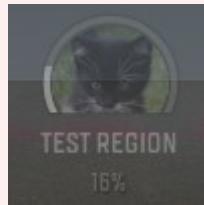
Set the **GlobalMapRegionBG** value to the `image_name` without the `.png` extension.

GlobalMapRegionIcon: text

Specifies the icon for the region in the list of regions in the global view and in the player profile.

Default: blank; the icon is blank.

Bug: A campaign with multiple regions is not supported, so there is no list of regions in the global view, and the region icon appears only dimly on the player profile.



The icon should be 85×85 and should be saved as a PNG file in
`prebuild/Region Name/ui/textures/icon_name.png`

Set the `GlobalMapRegionIcon` value to the `icon_name` without the `.png` extension.

Bug: If `icon_name` has more than 17 characters, the icon is not used.

Tip: The icon fits best into its container when it uses transparency to remove the corners and bottom portion. Saber provides a standard alpha mask at
https://drive.google.com/file/d/1aR_Og0xrtq_ATgi_B93hqdI_qvNUsS/view



Dev Tools Menu

You can choose whether the Dev Tools menu (page 39) is shown or hidden when the region is published. Normally it would be hidden, but you can choose to show it if your region consists of “proving grounds” type maps.

Enable Dev Menu: checkbox

If checked, the dev tools menu is shown when playing the published maps.

Default: unchecked; the dev tools menu is hidden in the published maps.

This setting overrides the per-map `isEnableDevMenu` setting in the Zone Settings Editor (page 291).

The Dev Tools menu is always shown for a region in development.

Ignore Country in Garage Store

Ignore Country Purchase Block: checkbox

If checked, the player can buy trucks from any country in a garage.

Default: unchecked; the player can only buy trucks from the country specified by **Country**.

This setting overrides the per-map `isIgnoreCountryPurchaseBlock` setting in the Zone Settings Editor (page 291).

Trailers are not associated with a particular country, so all trailers are available from a trailer store.

Map Cloaking

Watchpoints are still cloaked on a per-map basis, as described on page 292. There is no regional override.

Region Content Settings

You can adjust the availability of vehicles and add-ons in the **Content Settings** tab. These settings are identical to the map content settings set in the Zone Settings Editor (page 292).

Bug: The Region Settings Editor won't warn you if there are unsaved changes in the **Content Settings** tab. Make sure to save your changes.

When a map is part of a region, the map content settings are ignored, and the region content settings are used instead. If you've put a lot of work into the content settings of a map, you can directly copy the map's content settings from `Media/levels/map_name/content_settings.json` in place of the region's content settings at `Media/prebuild/Region Name/content_settings.json`. Then re-open the region to populate its **Content Settings** tab with the new information.

You can also copy content settings from a region to a map, from a region to another region, or from a map to another map.

Region Zone Types

You can add more zone types to a map in addition to the ones specified by the Zone Settings Editor. The primary use for this is to add gateway zones, which cannot usefully be added within an individual map. However, any zone types can be assigned at the region level. Zone types are added within the **Region** tab.

Unlike the Zone Settings Editor, the Region Settings Editor does not list all of the zones in the hierarchical list. Instead, you can add individual zones to the list that you wish to add types to.

MAPS LIST.map_name.TERRAIN LOCATORS LIST: +

Specifies a zone ID for which one or more zone types will be added. The zone ID is selected from a dropdown menu of zones in the map.

Once a zone is added to the list, you can add properties (zone types) to it in the usual way (page 295).

Gateway

A zone with the **ZonePropertyGateway** type acts as a gateway between maps. The zone perimeter is yellow. The forward orientation of the zone matters when it is used as a destination.

A gateway zone is half of a two-way passage between zones on two different maps. Confusingly, some properties are set on the origin zone, and some are set on the destination zone. Gateways are typically set up in pairs so that the player can travel back and forth and so that each gateway has both origin and destination properties.

Bug: You can set up a gateway as an origin without setting it up as a destination, but you can't set up a gateway as a (primary) destination without also setting it up as an origin.

Gateway Destination

MAPS LIST.map_name.TERRAIN LOCATORS LIST.origin_zone_id.props.ZonePropertyGateway.levelZoneLink: global zone ID (page 285)

Specifies the destination zone that the truck teleports to when the gateway is activated.

Default: blank; an invalid zone ID triggers a ModMapError (page 287).

This property is specified in the gateway properties of the **origin zone**, and the destination is specified in the **levelZoneLink** value. The destination zone does not need to also be a gateway, but you can only specify its destination properties (in the following sections) if it is.

The gateway must link to a zone on a different map. A gateway to a zone on the same map results in weird behavior such as duplicated trucks and trucks that aren't shown on the map.

When the player uses the gateway, the game places the truck in the destination zone with its nose touching the forward face of the zone, facing in the same direction as the zone. Note that this is unlike a garage exit or recovery zone, where the truck is centered in the zone.

To avoid a catastrophe, make sure that the largest truck can exit without intersecting an obstacle. Gateway collisions are handled in a similar manner to garage exits (page 297). However, because a truck can exit a gateway with a trailer in tow, it needs more space than it does when it exits from a garage.

Tip: Saber recommends that the gateway zone be at least 40×9 meters to accommodate the largest potential truck and trailer combination. But see the section below for more options.

If another truck obstructs the destination zone, both trucks become virtual until they are separated.

If the player tows another truck through the gateway, both trucks are placed at the front of the destination zone, and they become virtual until they are separated. The winch is detached in this case.

Extra Destination Zones

MAPS LIST.map_name.TERRAIN LOCATORS LIST.destination_zone_id.props.ZonePropertyGateway.extraZoneToPlaceTrucks: global zone ID (page 285)

Specifies a first alternative destination zone for when the primary destination zone is too small.

Default: blank; no alternative zone is specified.

MAPS LIST.map_name.TERRAIN LOCATORS LIST.destination_zone_id.props.ZonePropertyGateway.extraZoneToPlaceWinch: global zone ID (page 285)

Specifies a second alternative destination zone for when the primary destination zone and first alternative destination zone are too small.

Default: blank; no alternative zone is specified.

These properties are specified in the gateway properties of the (primary) **destination zone**, and they specify alternative destination zones in the property values. The alternative destination zones are not visible in the driving view or navigation map.

If either extra zone is specified, it must be on the same map as the destination zone, or the game will trigger a ModMapError (page 287).

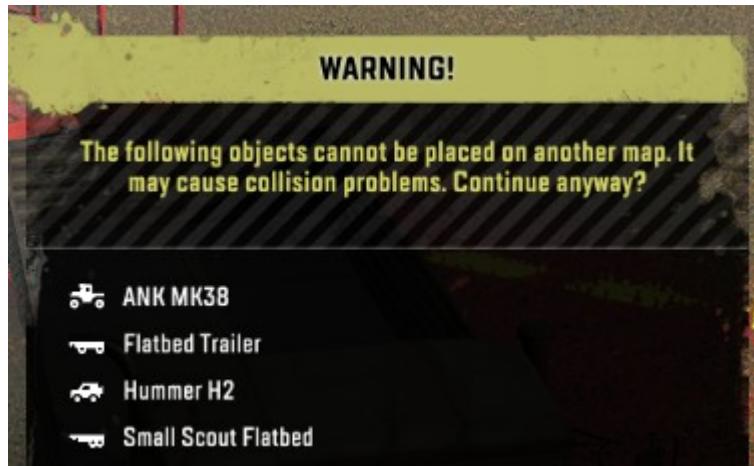
If the player's truck and trailer or a towed truck and trailer are too large to fit in the primary destination zone, and the first alternative destination zone is specified, the game tries to put the truck(s) there instead. If the player's truck and trailer or a towed truck and trailer are also too large to fit there, and a second alternative destination zone is specified, the game puts the truck(s) there instead. The last specified zone is used whether or not it is large enough.

When evaluating whether a truck and attached trailer fit in a zone, both the length and width are considered.

Tip: If an alternative destination zone is specified, the last one should be at least 40×9 meters to accommodate the largest potential truck and trailer combination.

Tip: The various destination zones can overlap. In particular, you can make the visible primary gateway zone a reasonable visible size and make the invisible alternative zone be larger. If the gateway is close to a road junction, you may be able to fit a smaller truck prior to the junction so that the player has a choice of where to go, while a larger combination may need to be placed after the junction to have sufficient room.

If the last specified zone is not large enough, the game checks whether the player really wants to attempt passage through the gateway. Note that the game checks only the last specified zone before displaying the confirmation dialog. You should ensure that the last specified zone is the largest.



Despite the property names, Saber's documentation agrees that they act as first and second alternative zones, not as separate zones for the player's truck and a towed truck. The player's truck and the towed truck are always placed in the same location and made virtual, even when the destination zone is plenty large enough for them to be arranged separately.

Saber claims that the extra zones are not fully supported yet and recommends that they be left blank, but they seem to work fine to me. Saber also says that the gateway may stop working properly if either alternative zone has any zone types assigned, but I haven't had a problem with that, either. But if you would like another zone function in the same area, consider making a separate overlapping zone for that purpose.

Animation at Destination

The animation after a gateway transfer is much like that used for a watchpoint (page 300), except that the camera movement always ends at the position and orientation of the player's truck.

MAPS LIST.map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyGateway.Delay at start:
integer, in milliseconds

Specifies how long the camera waits at the start position before it starts moving toward the truck position.
Default: 0 ms.

MAPS LIST.map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyGateway.cameraPos: 3-D position

Specifies where the camera starts at the beginning of the gateway animation.

Default: (0, 0, 0); not a useful location.

MAPS LIST.map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyGateway.cameraDir: 3-D orientation

Specifies where the camera points at the beginning of the gateway animation.

Default: (0, 0, 0); no animation is played.

MAPS LIST.map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyGateway.Duration of transition: integer, in milliseconds

Specifies how long the camera movement takes from the start position to the truck position.

Default: 0 ms; the game won't quite hang, as it does eventually respond to a press of **Esc** to end the animation, but it definitely has a hard time of it. The minimum effective duration is 1 ms.

MAPS LIST.map_name.TERRAIN LOCATORS LIST.zone_id.props.ZonePropertyGateway.Delay at end: integer, in milliseconds

Specifies how long the camera waits at the truck position before the animation ends.

Default: 0 ms.

When editing the times, the “ms” text is uneditable. The time is always in milliseconds (thousandths of a second), so e.g. **7000 ms** is 7 seconds.

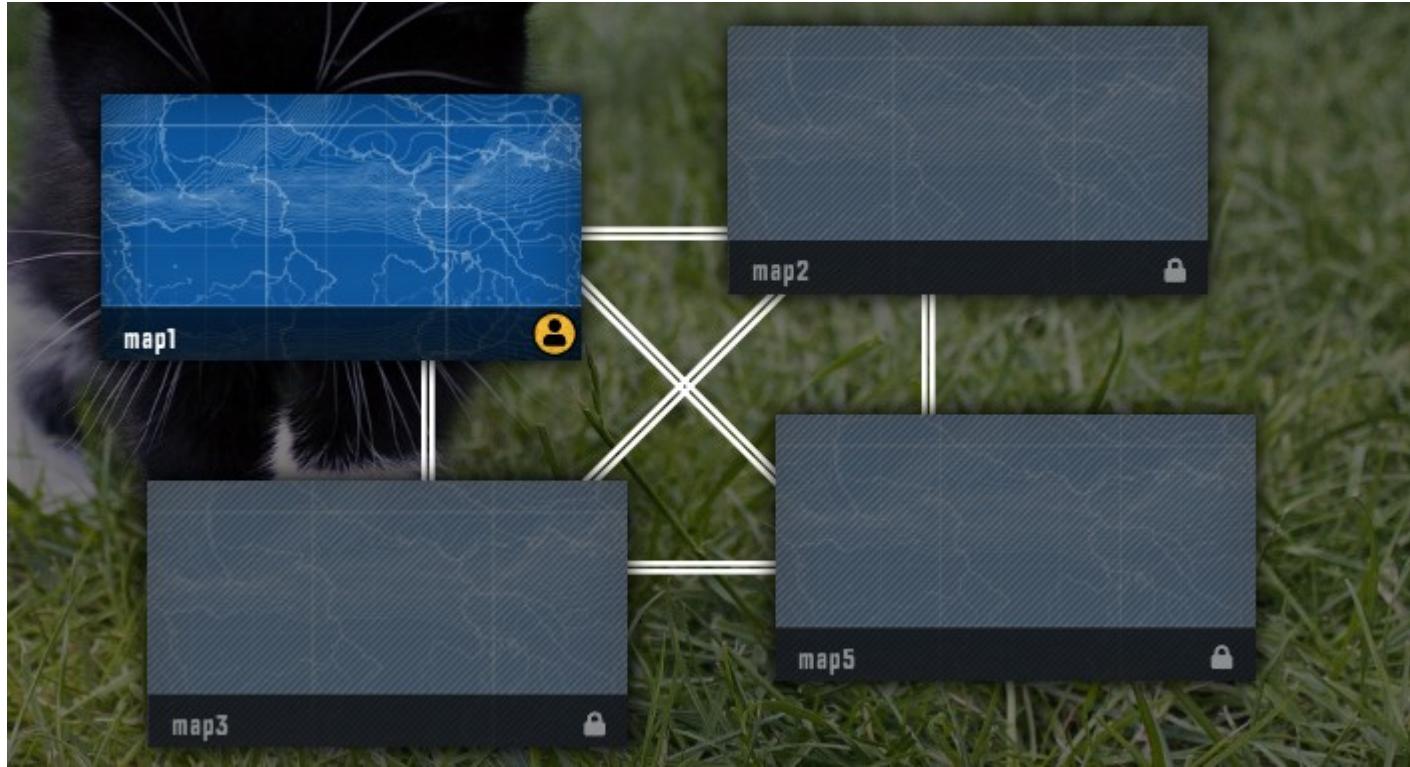
When editing the camera positions and orientations, the three values split into separate **x**, **y**, and **z** fields, each of which can be edited. The coordinate systems for position and orientation are described on page 420.

Tip: The easiest way to get a good camera position and orientation is to position the camera in the Editor, then enable the **Statistics** toggle button in the toolbar. The statistics list the camera position followed by the orientation: **(x; y; z) > (x; y; z)**.

Camera: (-26.68; 30.19; -25.96) > (0.88; -0.24; 0.40)

Gateways in the Global View

If a pair of maps is connected by a gateway in either direction, a link is drawn between the maps in the global view, even if the player hasn't reached the gateway yet.



Locked Gateway

The built-in campaign includes locked gateways that can be unlocked via objectives. While it's possible to add a gateway type to a zone via a contract (page 320), there are no clues in the Editor or Saber documentation how the zone can show up as a locked gateway before that.

Regional Contracts

Additional contracts can be added at the regional level. A regional contract can use zones on one map or on multiple maps. All map-level and region-level contracts must have names that are distinct from all other contracts, tasks, and contests.

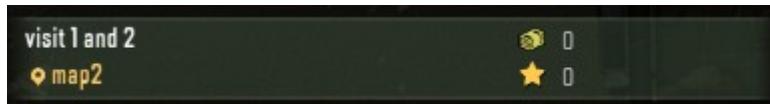
Tasks and contests are not supported at the regional level. They can only be created in the Zone Settings Editor using zones on a single map. Tasks and contest names only need to be unique within each map. Multiple maps can use the same task or contest name.

To add a regional contract, go to the **Region** tab and click the **+** to the right of **objectiveSettings.contractsSettings**. Regional contracts are set up identically to those at the map level with the addition of one property that only exists at the region level:

objectiveSettings.contractSettings.objective name.Map name for player profile: map name

Specifies which map is associated with the contract when it is viewed in the player profile.

Default: blank; the game defaults to whichever map is listed first among the contract's properties. If none of the goals refers to a map, the map is left blank in the player profile.



When editing the property value, you can click the **...** button on the right side to get a list of map names to choose among.

Cross-Map Contract Restrictions and Notes

Visit and Seismic Vibrator Goals

Within a single stage, all visit goals (page 359) and seismic vibrator goals (page 360) must specify destination zones within the same map. Otherwise, the game triggers a ModMapError (page 287).

Truck ID and Spawning Cargo Restrictions

In order for the player to restart a contract, the game requires the player to travel to the map where trucks and/or cargo will respawn. I.e. this applies to any contract with a truck delivery goal (page 350), a truck repair goal (page 355), or where cargo is spawned for a stage (page 340).

Bug: If an objective includes a change truck goal (page 357), the player can only restart if she travels to the map where the truck was initialized, even though that truck will not respawn when the objective is restarted.

Bug: If an objective includes a visit goal (page 359) or seismic vibrator goal (page 360) that requires the zone to be visited by a particular truck, the player can only restart if she travels to the map where the truck was initialized, even though that truck will not respawn when the objective is restarted.

Bug: If an objective includes a cargo delivery goal (page 332) that requires the truck to be delivered by a particular truck, the player can only restart if she travels to the map with the cargo delivery zone (**not** the map where the truck was initialized), even though that truck will not respawn when the objective is restarted.

You should ensure that the game requires the player to travel to no more than one map in order to restart. I.e. all reserved trucks and spawned cargo (and other bug-required areas) are on the same map. Otherwise the game will keep sending the player back and forth between maps without ever allowing her to restart the objective.



Note that when a truck respawns, although the player must be on the map where the truck appears, it can disappear from any map, and the game handles it correctly.

Access to Zone Reward

When rewarding the player with access to one or more zones, those zones can be on any map(s).

Multi-Stage Models

A multi-stage model will correctly change state (pages 328 and 338) even if the player triggers the state change from another map. If the model state change includes an animation, the animation is only shown if the player is on the same map as the model at the time of the change.

Publish Your Map

To publish your packed map to mod.io, visit SnowRunner's upload page at <https://snowrunner.mod.io/add>

Your map or region will be published as a “mod”. Most SnowRunner mods are custom trucks, but the process is just as easy to publish a map mod.



The publishing process is mostly performed on the mod.io website, and your published mod will have its own page on the website where people can subscribe to it.

Profile

Here you can set up a page for your mod by entering the required data:

- Mod name

Tip: The mod name must be unique among SnowRunner mods.

- Summary

Tip: I recommend stating the primary goal of the mod here, e.g. rock-crawling, exploration, or cargo hauling.

- Description

Tip: If your map makes use of DLC trucks, be sure to mention it prominently here. Players will be mad if trucks are mysteriously missing from the map.

Tip: mod.io seems to be a mostly English site, so I recommend using English for the mod name and summary. If you've created a multi-lingual map, you may want to highlight this and include translations in the description section.

Tip: I recommend including the overall size and/or complexity of the map here. E.g. “three maps totalling 5 square km” or “36 objectives to complete, some of which are required to unlock portions of the map”.

- Visibility: public or hidden

Tip: Only people you designate can subscribe to a hidden mod. Once you are satisfied, you can change it to public.

- Dependencies (on other mods)

Bug: There is no way to assert a dependency on DLC. If your map requires DLC trucks (or otherwise uses them at all), be sure to mention it in the description section above.

- Preview image; click the placeholder to open a file browser dialog
- Tags

For a map, the following tags are typical:

- Installation: subscribe
- Type: map
- Vehicle: <leave blank>
- Map: <season>

Tip: If your mod has a distinct season, specify it here to allow players to find it when searching for in their preferred season. If your mod uses multiple seasons or doesn't have an easily categorized season, leave this field blank.

Most map mods are automatically appropriate for single player. Multiplayer requires a garage for the additional players to join through, so only check the multiplayer box if the player can quickly unlock a garage.

The “Changes” section is appropriate only for trucks, so leave those checkboxes blank.

Media

You can upload additional images and YouTube URLs to make your map mod look appealing.

There is also a section to add Sketchfab models, but that is really only useful for truck mods.

Files

Finally, here is where you upload the mod file.

Click **Choose File** to open a file browser. Navigate to your **Media/levels** directory and choose the ***.zip** file. You can click **Upload** on the right to upload the file immediately, or it will upload automatically when you go to the next step.

You can release multiple version of your mod. To help keep them straight, you can assign a version number and a changelog to each.

By default, the latest upload is the “primary release”, which is what someone will get unless they make the extra effort to select a different version.

Finally, make sure to check the “I agree” checkbox, or mod.io won’t let you upload your file.

Team

If you are collaborating with other mod.io members, you can add them to the mod team here. Each team member can have various levels of access to see, edit, or manage the mod.

Alternatively, if you just want someone to help test your map, you can get a secret URL to your hidden map from the mod profile page.

This mod is hidden ([make public](#)). Only team members and moderators can view it (you can share this [preview URL](#) for early access). ×

Subscribe

If your map is public, you should be able to find it via the SnowRunner mods search bar (and you probably do want to check that it shows up correctly there).

However, the easiest way to find your mod is through your profile. Simply click your name in the upper right of any mod.io page to go to a list of your mods.



Click on a mod to go to its page. Initially, it will have a big “Subscribe to install” button on the right.

Subscribe to install

0

Click the button to subscribe to your mod.

Subscribed

1

Activate

SnowRunner automatically downloads any mod that you are subscribed to, but it doesn’t add it to the list of available maps until you activate it.

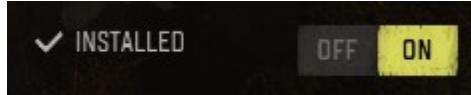
From the SnowRunner main menu, select **MOD BROWSER**. The Mod Browser lets you search for public mods, but the only way to find a hidden mod is to change the filter.



Click on the filter to change it, and check the box next to **Subscribed Mods Only**.



Once your mod is displayed, you can select it and flip the **INSTALLED** switch to **ON**.



Play

From the SnowRunner main menu, you can finally select **NEW GAME**, **CUSTOM SCENARIOS**, and then your mod name.

Appendix A: Reference Information

The following sections contain various lists and tables that are handy for reference but would have cluttered up the main text. Once you're familiar with the Editor, you may find yourself using this appendix the most often.

Truck Reference

The table below provides overview information for all trucks so that you don't have to try them all individually. To get detailed information about a particular truck, use the dev tools to create the truck and view it in the virtual garage to discover all allowed add-ons for the truck.

Another source of vehicle information is the [SpinTires fandom wiki](#).



Table Column Meanings

Truck is the name of the truck in the Editor and the dev tools. The name of the truck in the game is similar. (In fact, the truck names are virtually identical across all languages, even Russian.)

Reg. indicates the region that the truck is associated with.

- US is the United States/North America.
- Rus. is the Russian Federation.

AWD indicates the truck's all-wheel-drive capability.

- no: AWD cannot be enabled
- upgrade: AWD can be installed after the appropriate upgrade is obtained
- yes: AWD can be enabled by the player
- always: AWD is always enabled

Diff Lock indicates the truck's diff-lock capability.

- no: diff lock cannot be enabled
- upgrade: a locking differential can be installed after the appropriate upgrade is obtained
- yes: diff lock can be enabled without an upgrade, but only in a low gear
- yes+no: yes by default, but can be removed with an "upgrade"
- always: the differential is always locked

Tires indicate what types of tires the truck can use.

- H: highway
- A: all-terrain
- O: offroad
- M: mud
- C: chains

Trailer indicates what kinds of trailers the truck can pull.

- S: scout trailers
- T: regular trailers
- L: low-saddle semi-trailers
- H: high-saddle semi-trailers
- *: excludes semitrailer_oiltank and semitrailer_heavy_oiltank
- †: excludes semitrailer_m747
- ‡: excludes semitrailer_stepdeck_5 and semitrailer_goose_neck_4
- R: special rock semi-trailer

Bed indicates what kind of equipment and non-log cargo the truck can carry in its bed (or on the roof).

- B: big crane
- R: repair parts; if followed by a *, can be combined with a saddle
- r: a small number of repair parts; sometimes can be combined with other bed items. Spare wheels are not covered in this reference.
- F: fuel tank
- f: a small amount of fuel; sometimes can be combined with other bed items
- M: maintenance parts (repair + fuel). Blocks use of a trailer.
- m: a small amount of maintenance parts (repair + fuel); sometimes can be combined with other bed items
- D: metal detector. Can be combined with some bed items on some trucks, but that looks like be a bug.
- V: seismic vibrator. Blocks use of a trailer.
- T: towing package
- C: mini-crane. If followed by a '+', can be combined with any following options (usually only cargo units). The combination may prevent attachment of trailers. The mini-crane itself sometimes conflicts with semi-trailer saddles or specific fuel semi-trailers.
- 1–4: 1 to 4 generic cargo units
- G: can carry a cargo container, but not other cargo units

Logs indicates what kind of logs and/or logging crane the truck can transport.

- C+: log crane; can be combined with short, medium, or long logs if available; conflicts with other bed items
- C/: log crane; can be combined with medium log trailer if available, but not S or L or other bed items
- S: short logs in the bed; can be combined with medium log trailer if available
- M: medium logs in a trailer
- L: long legs stretching from the bed to a trailer

DLC indicates the DLC that must be owned to use the truck.

- S1: Season 1: Search & Recover
- S2: Season 2: Explore & Expand
- S3: Season 3: Locate & Deliver
- S4: Season 4: New Frontiers
- S5*: Season 5: Build & Dispatch
- AD is the Anniversary DLC (free)
- TD* is the Tatra Dual Pack
- JD* is the Jeep Dual Pack
- CL* is the Classico Pack
- GB* is the GMC Brigadier DLC
- N5* is the Navistar 5000-MV DLC (limited availability)
- WS* is the Western Star 49X DLC
- * indicates that I don't own these DLCs, so my data quality might not be as good as for other trucks.

Some truck add-ons were released as part of a DLC package, but you don't need to own the DLC to get the add-ons. DLC is required only for the truck itself.

Scout Trucks

Reg.	Truck	AWD	Diff Lock	Tires	Trailer	Bed	Logs	DLC
US	cat_th357	always	always	M		C		S2
	chevrolet_ck1500	yes	upgrade	HAOMC	S	rfm		
	chevy_apache	yes	always	AOMC		m		CL*
	ford_f750	upgrade	always	HOC	T	C+1	M	S1
	hummer_h2	yes	upgrade	HAOMC	S	mm		
	international_loadstar_1700	always	always	HAOC	S	CRm		
	international_scout_800	yes	always	HAOMC	S	m		
	jeep_cj7_renegade	yes	yes	HAOMC		m		JD*
	jeep_wrangler	yes	yes	HAOMC	S	m		JD*
Rus.	don_71	yes	yes	HAOMC	S	fm		
	khan_317_sentinel	yes	yes	HAOMC	S	f		S4
	khan_39_marshall	always	yes	M		m		
	khan_lo4f	always	always	HAOMC	S	m		
	tuz_166	always	always	HAOMC	S			
	tuz_420_tatarin	always	always	O		m		
	yar_87	yes	always	AOMC	S	m		

Highway Trucks

Reg.	Truck	AWD	Diff Lock	Tires	Trailer	Bed	Logs	DLC
US	ford_clt9000	no	no	HAOC	TLH	BRFMDTC+2	C/SML	
	gmc_9500	upgrade	upgrade	HAOC	TLH	BRFMDVTC+2	C/SML	
	international_tristar_4070a	no	no	HAOC	TLH	BRFDTC+2	C/SML	

Heavy Duty Trucks

Reg.	Truck	AWD	Diff Lock	Tires	Trailer	Bed	Logs	DLC
US	cat_ct680	upgrade	yes+no	HAOC	TLH	BRFMDTC+2	C/SML	AD
	cat_ct681	upgrade	upgrade	HAOC	TLH	BRFMTC+2	C/SML	
	chevrolet_kodiakc70	upgrade	yes	HAOC	TLH	BRFDTC+2	C/SML	
	gmc_8000	upgrade	no	HAOC	TLH	BRFMDVTC+2	C/SML	GB*
	international_fleetstar_f2070a	upgrade	yes	HAOC	TLH‡	BRFMDTC+2	C/SML	
	international_hx_520	upgrade	no	HAOC	TLH	BRFMTC+2	C/SML	AD
	ws_4964_white	upgrade	upgrade	HAOC	TLH	BRFMTC+2	C/SML	

Offroad Trucks

Reg.	Truck	AWD	Diff Lock	Tires	Trailer	Bed	Logs	DLC
US	ank_mk38	always	always	HAOMC	T	2	M	
	freightliner_114sd	upgrade	yes	HAOC	TLH	BRFMDVTC+2	C/SML	
	freightliner_m916a1	yes	always	HAOC	TLH*	C	M	
	international_paystar_5070	yes	yes	HAOMC	TLH	BRFMDVTC+2	C/SML	
	royal_bm17	yes	no	HAOC	TLH	BRFMDVTC+2	C/SML	
Rus.	azov_5319	always	always	AOMC	TLH	BRFMVTC+2	C/SML	
	azov_64131	always	always	AOMC	TLH	BRFMVTC+2	C/ML	
	krs_58_bandit	always	always	HAOMC	TLH	FMVTC+Rr2	C/SML	S2
	step_310e	yes	yes	HAOC	TLH	BRFMDTC+2	C/SML	
	tatra_805	yes	upgrade	HAOMC	T	m2	M	TD
	tayga_6436	always	always	HAOMC	TLH	BRFVC+2	SML	
	tuz_16_actaeon	yes	yes	HAOMC	T	RFC+1	ML	S1
	tuz_108_warthog	yes	yes	HAOMC	TL	RFC1	M	
	voron_ae4380	always	always	HAOMC	TLH	BRFDTC+2	SML	
	voron_d53233	always	always	HAOMC	TLH	BRFMDTC+2	C/SML	
	zikz_5368	yes	upgrade	HAOMC	TL	RFMD2C+1	C/ML	

Heavy Trucks

Reg.	Truck	AWD	Diff Lock	Tires	Trailer	Bed	Logs	DLC
US	cat_745c	yes	yes	M		FG	SM	
	cat_770g	no	always	M		F	R	S2
	derry_longhorn_3194	always	always	HAOC	TLH		M	
	derry_longhorn_4520	yes	yes	HAOC	TLH		ML	
	navistar_5000mv	always	yes	HAOC	TLH	C	ML	N5*
	pacific_p12w	yes	always	AOMC	TH	BRMDT	C/SML	
	pacific_p16	no	always	O	TH†		ML	
	pacific_p512	no	upgrade	O	TLH	BFMDVTC+R2	C/SML	S3
	paystar_5600ts	upgrade	upgrade	HAOMC	TL	BR*FTC+3	C+SML	
	western_star_49x	upgrade	yes	HAOC	TLH	RFMVTC+2	ML	WS*
	ws_6900xd_twin	upgrade	yes	HAOMC		4		
Rus.	azov_4220_antarctic	always	yes	M	T	T2	C/ML	
	azov_73210	always	always	AOMC	TLH	BRFMVT3C+2	C/SML	
	boar_45318	yes	upgrade	HAOC	TH	RV	ML	S3
	dan_96320	always	always	AOMC	TLH	BRF2	ML	
	kolob_74760	always	always	M	TH		M	
	kolob_74941	yes	yes	M	TH		M	
	tatra_t813	yes	yes	AOMC	TH	mT2	M	TD*
	tatra_force_t815_7	yes	upgrade	AOMC	TH	BFT2	M	S5*
	tatra_phoenix	yes	upgrade	AOMC	TH	BF2	M	S5*
	zikz_605r	always	always	M	TH	BRrFV2	C/SM	S4

Bug: The cat_745c_log_bunk add-on can hold either short or medium logs, so for example a CAT 745 can be initialized with cargo_logs_cat_745c_log_bunk (short logs). However, due to a bug, the same short logs cannot be loaded onto the CAT 745 in the game.

Trailer Reference

Table Column Meanings

Trailer is the name of the trailer in the Editor and the dev tools.

Obj. indicates whether the trailer is intended only for objectives.

- no: the trailer can be bought and sold at a trailer store
- yes: the trailer cannot be bought or sold at a trailer store

Function indicates what functions and/or cargo capacity a trailer provides.

The player does not need to own any DLC to use all trailers.

Scout Trailers

Obj.	Trailer	Function
no	scout_trailer_flatbed_1	1 cargo unit
	scout_trailer_flatbed_2	2 cargo units
	scout_trailer_oiltank	900 fuel
	scout_trailer_radar	350m radar, 120 fuel

Regular Trailers

Obj.	Trailer	Function
no	trailer_addon_maintainer	350 repair parts, 2000 fuel
	trailer_flatbed_2	2 cargo units
	trailer_sideboard_2	2 cargo units
	trailer_flatbed_ramps_4	4 cargo units
	trailer_oiltank	2000 fuel
	trailer_service_2	1500 repair parts
	trailer_generator	generator, 1500 fuel
	trailer_log	medium logs
yes	trailer_flatbed_special_2	none (curtainside trailer)

Low-Saddle Semi-Trailers

Obj.	Trailer	Function
no	semitrailer_gooseck_4	4 cargo units
	semitrailer_flatbed_5	5 cargo units
	semitrailer_sideboard_5	5 cargo units
	semitrailer_stepdeck_5	5 cargo units
	semitrailer_oiltank	3700 fuel
yes	semitrailer_special_w_cat_770	none

High-Saddle Semi-Trailers

Obj.	Trailer	Function
no	semitrailer_m747	3 cargo units
	semitrailer_gooseck_8	8 cargo units
	semitrailer_stepdeck_plane_01	5 cargo units
yes	semitrailer_heavy_oiltank	5000 fuel
	semitrailer_coiled_tubing	none
	semiltrailer_heavy_construction_equipment	none
	semitrailer_oil_rig	none
	semitrailer_oil_refinery	none
	semitrailer_for_rocket	none
	semitrailer_rocket	none

Other Trailers

Obj.	Trailer	Function
no	trailer_log_pole	requires bunk_log_addon
yes	semitrailer_cat770g	can only be attached to cat_770g
	train	cannot be attached
	trailer_train_rocket	cannot be attached
	cargo_cabin_01	cannot be attached

Cargo Types

The following tables give the internal name used by the Zone Settings Editor for each of the cargo types. The number of cargo slots used by each is also listed.

Logs

Logs require a special logs carrier of the appropriate size, and each load fills the carrier.

Cargo Type	Editor Name	Logs per Load
short logs	CargoLogsShort	3
medium logs	CargoLogsMedium	3
long logs	CargoLogsLong	3

Pallets

Cargo Type	Editor Name	Cargo Slots
cement	CargoBags	1
packaged sand	CargoBags2	1
fuel	CargoBarrels	1
oil barrels	CargoBarrelsOil	1
bricks	CargoBricks	1
cellulose bricks	CargoCellulose	1

Crates and Containers

Cargo Type	Editor Name	Cargo Slots
consumables	CargoCrateLarge	1
vehicle spare parts	CargoVehiclesSpareParts	1
service spare parts	CargoServiceSpareParts	1
drilling spare parts	CargoServiceSparePartsSpecial	1
secure radioactive container	CargoRadioactive	1
cargo container	CargoContainerSmall	2
special cargo	CargoContainerSmallSpecial	2
oversized cargo container	CargoContainerLarge	4
drilling equipment	CargoContainerLargeDrilling	4

Frames and Straps

Cargo Type	Editor Name	Cargo Slots
wooden planks	CargoWoodenPlanks	1
concrete blocks	CargoConcreteBlocks	1
concrete slabs	CargoConcreteSlab	2
metal beams	CargoMetalPlanks	2
metal rolls	CargoMetalRoll	1
small pipes	CargoPipesSmall	2
medium pipes	CargoPipesMedium	2
large pipe	CargoPipeLarge	4
small cabin	CargoForkliftCaravanContainer2	2
large cabin	CargoForklift	4 (excludes semitrailer_sideboard_5)
rails	CargoIronRoad	4 (excludes semitrailer_sideboard_5)
industrial boiler	CargoBoiler	5 (excludes semitrailer_sideboard_5)
oil rig drill	CargoBigDrill	5
sequoia trunk	CargoSequoia	5

Vehicle Sections

Cargo Type	Editor Name	Cargo Slots
BA-20 armored car (rusted)	CargoBA20	2
BA-20 armored car (wrecked)	CargoBA20Add	2
rail section	CargoRailway	5
rocket engine assembly	CargoRocketEngine	1
stage 2 rocket fuel tank	CargoRocketPart2	3
stage 3 rocket fuel tank	CargoRocketPart1	3
airplane fuselage part	CargoPlane	5 (requires semitrailer_stepdeck_plane_01)
wing wreckage with engine	CargoWing1	5 (requires semitrailer_stepdeck_plane_01)
wing wreckage	CargoWing2	4 (requires semitrailer_stepdeck_plane_01)

Material Layer Reference

Below is reference information for the various material layers available in the Editor. Since the material layer dialog box shows a preview of the ground texture, I won't cover that. But many material layers share the same ground texture while having different 3-D models and/or different properties. I've highlighted some of the differences below.

The recommended range for the **Tiling scale** property is derived from the values used in Saber's reference maps. A few maps use values outside of the ranges I've listed; feel free to experiment and use a value that looks natural in your setting.

Some layers have weird suffixes like _test and _out that look like they were released by mistake. But most of these are used extensively in Saber's reference maps, so it's more likely that these materials simply got used in too many places and so weren't worth the trouble of renaming. I think you can feel comfortable using them in your own maps.

TBD: review special properties in XML files

Grass

Material Layer	Tiling scale	Notes
default	4 – 7	the default 3-D grass models
grass_out	4 – 7	clumpier 3-D grass models
grass_test	4 – 7	thick, tall 3-D grass, and a few flowers and weeds
grass_spring_01	4 – 7	brown, dry-looking 3-D grass and a few flowers
grass_rus_01	5	lush green 3-D grass and tall flowers and weeds
grass_rus_02	5	the same, but without flowers

Dirt

Material Layer	Tiling scale	Notes
ground_usa_01	3 – 5	with 3-D dirt clumps
ground_rus_01	4 – 5	the same 3-D dirt clumps as ground_usa_01
ground_spring_01	5	with different 3-D dirt clumps; allows much more hidden mud
ground_test	5	with 3-D stones
ground_snowy	5	with 3-D dirt clumps; wetness is ice; supports snow?

Sand

Material Layer	Tiling scale	Notes
sand_pbr_02	2.5 – 5	with 3-D dirt clumps
sand_pbr_03	2.5 – 5	only the ground texture differs
sand_test	2.5 – 5	only the ground texture differs; looks more like dunes

Mud

Material Layer	Tiling scale	Notes
mud	1.5	no 3-D objects
mud_flat	1.5	only the ground texture differs; is a bit less wet looking

Gravel

Material Layer	Tiling scale	Notes
gravel_road	3 – 5	with 3-D stones
gravel_road_rus	3 – 5	no 3-D objects
gravel_rus_01	2.5 – 20	no 3-D objects
gravel_pbr_test	3 – 5	no 3-D objects

Rocks

Material Layer	Tiling scale	Notes
rocks_rus_01	1 – 3	no 3-D stones
rocks_rus_01_snowy	1 – 3	no 3-D objects; wetness is ice; supports snow?

Stone

Material Layer	Tiling scale	Notes
stone_spring	1	no 3-D objects; wetness is ice
stone_snowy	1	no 3-D objects; supports snow?
stone_test	1	no 3-D objects

Concrete

Material Layer	Tiling scale	Notes
us_concrete	3 – 5	no 3-D objects
us_concrete_old	3 – 5	no 3-D objects
us_concrete_tile	3 – 5	no 3-D objects

Asphalt

Material Layer	Tiling scale	Notes
asphalt_rus_01	3 – 5	no 3-D objects
asphalt_rus_01_old	1 – 5	no 3-D objects
asphalt_rus_01_snow	1 – 5	no 3-D objects; wetness is ice
us_snow_asphalt	3 – 4	no 3-D objects; wetness is ice

Ice

Material Layer	Tiling scale	Notes
ice_01	3	no 3-D objects; wetness is ice; can see through to a second layer underneath
ice_02	3 – 4	wetness is ice; has a subsurface layer so that mud softness looks clean underneath; see page 218 for details

Snow

Material Layer	Tiling scale	Notes
snow_layer	2.2	supports soft_snow and crust_snow; supports snow depth
snow_spring	?	looks like snow_layer, but behaves like a normal terrain layer

Employers

The following built-in employers are available for contracts (page 317). Alternatively, you can create a custom employer (page 314).

Regardless of whether or not a company is given an English name below, you can use localization to assign your own name to a company. See page 314 for details.

Icon	Editor Name	Localization ID	English Name
	AmurFuelCompany	UI_CONTRACT_GIVER_COMPANY_AMUR_FUEL	AMUR PETROLEUM COMPANY
	AmurRegionalGov	UI_CONTRACT_GIVER_AMUR_REGGOV	AMUR REGIONAL ADMINIST.
	AttecaTownship	UI_CONTRACT_GIVER_COMPANY_ATTECA	STEEL RIVER TOWNSHIP
	BlackBird	UI_CONTRACT_GIVER_COMPANY_BLACK_BIRD	BLACK BIRD
	DysonDiesel	UI_CONTRACT_GIVER_COMPANY_DYSON	DYSON DIESEL
	GarlicOil	UI_CONTRACT_GIVER_COMPANY_GARLIC	THE COMPANY
	GoldhorseMining	UI_CONTRACT_GIVER_COMPANY_GOLD	GOLDHORSE MINING
	GreenLake	UI_CONTRACT_GIVER_COMPANY_GREENLAKE	GREENLAKE PAPER MILL
	GREnterprise	UI_CONTRACT_GIVER_COMPANY_GRE	GR ENTERPRISE
	Husky	UI_CONTRACT_GIVER_COMPANY_HUSKY	<no localization>
	HuskyForwarding	UI_CONTRACT_GIVER_COMPANY_HUSKY_FORWARDING	HUSKY FORWARDING

Icon	Editor Name	Localization ID	English Name
	KolaExpedition	UI_CONTRACT_GIVER_KOLA_EXPEDITION	KOLA EXPEDITION
	KolskyRegionalGov	UI_CONTRACT_GIVER_KOLSKY_REGGOV	KOLSKY ADMINISTRATION
	Maneuver29Agency	UI_CONTRACT_GIVER_COMPANY_MANEUVER_29	MANEUVER-29 AGENCY
	MorrisonMining	UI_CONTRACT_GIVER_COMPANY_MORRISON	MORRISON MINING
	Nonagon	UI_CONTRACT_GIVER_COMPANY_NONAGON	NONAGON INFRASTRUCTURES
	RostovRegionalGov	UI_CONTRACT_GIVER_COMPANY_RostovRegionalGov	ROSTOV ADMINISTRATION
	SeverMotorRepairs	UI_CONTRACT_GIVER_SEVMOT_REPAIRS	SEVMOT REPAIRS
	Stonecreek	UI_CONTRACT_GIVER_COMPANY_STONECREEK	STONECREEK LUMBER
	TaigaOil	UI_CONTRACT_GIVER_COMPANY_TAIGA	TAIGA OIL
	Tatra	UI_CONTRACT_GIVER_COMPANY_RostovRegionalGov	TATRA
	TransTaymir	UI_CONTRACT_GIVER_COMPANY_TRANS_TAYMIR	TRANSTAYMYR
	Voronoe12	UI_CONTRACT_GIVER_COMPANY_VORONOE	VORONOE 12
	WestRailways	UI_CONTRACT_GIVER_COMPANY_WestRailways	WESTERN RAIL ROADS
	WoodyWoodpecker	UI_CONTRACT_GIVER_COMPANY_WOODY	<no localization>

Zone Icons

The following icons can be used to mark zones in the game. In the table below, only the 40×40 icon image is shown. The 30×30 icon is similar.

Use the 40×40 icon name in the zone's **Icon 40x40** property value, and use the 30×30 icon name in the zone's **Icon 30x30** property value.

Travel Services

Icon Image	Purpose	30×30 Icon Name	40×40 Icon Name
	garage	garageImg30	garageImg
	trailer shop	trailerShopImg30	trailerShopImg40
	fuel station	fuelStationImg30	fuelStationImg
	service hub	serviceHubImg30	serviceHubImg
	watchtower	watchTowerImg30	watchTowerImg
	upgrade station	upgradeCreateImg30	upgradeCreateImg40
	crafting area	craftCrafting30	craftCrafting40
	gateway (tunnel)	gatewayImg30	gatewayImg
	locked gateway	gatewayLockedImg30	gatewayLockedImg

Objectives

Icon Image	Purpose	30×30 Icon Name	40×40 Icon Name
	contest	contestImg30	contestImg
	new task	taskNewImg30	taskNewImg40
	task	taskImg30	taskImg40
	subtask	taskSubImg30	taskSubImg40
	lost cargo	constructionCargoImg30	constructionCargoImg
	lost truck	troubleAltVehicleImg30	troubleAltVehicleImg

Cargo Areas

Icon Image	Purpose	30×30 Icon Name	40×40 Icon Name
	airport	airportImg30	airportImg
	bunker	bunkerImg30	bunkerImg
	drilling site	drillingSiteImg30	drillingSiteImg
	factory	factoryImg30	factoryImg
	farm	farmImg30	farmImg
	lumber station	sawmillImg30	sawmillImg
	storage	townStorage30	townStorage
	timber station	lumberjackImg30	lumberjackImg
	warehouse	constructionWarehouseImg30	constructionWarehouseImg
	manual loading	manualLoading30	manualLoadingMarker40
	manual unloading	manualUploading30	manualUploadingMarker40

Vehicles

Icon Image	Purpose	30×30 Icon Name	40×40 Icon Name
	trucks	allVehicleImg30	allVehicleImg
	scout truck	scoutVehicleImg30	scoutVehicleImg
	highway truck	highwayVehicleImg30	highwayVehicleImg
	heavy-duty truck	heavyDutyVehicleImg30	heavyDutyVehicleImg
	offroad truck	offroadVehicleImg30	offroadVehicleImg
	heavy truck	heavyVehicleImg30	heavyVehicleImg

Ruins

Icon Image	Purpose	30×30 Icon Name	40×40 Icon Name
	brick ruins	ruins30Bricks1	ruins40Bricks1
	brick ruins	ruins30Bricks2	ruins40Bricks2
	brick ruins	ruins30Bricks3	ruins40Bricks3
	metal ruins	ruins30Metals1	ruins40Metals1
	metal ruins	ruins30Metals2	ruins40Metals2
	metal ruins	ruins30Metals3	ruins40Metals3
	wood ruins	ruins30Woods1	ruins40Woods1
	wood ruins	ruins30Woods2	ruins40Woods2
	wood ruins	ruins30Woods3	ruins40Woods3

Miscellaneous

Icon Image	Purpose	30×30 Icon Name	40×40 Icon Name
	GPS marker	markerImg30	markerGpsImg
	living area	craftLivingArea30	craftLivingAreaMarker40
	road closed	constructionBlockageImg30	constructionBlockageImg
	checkpoint	constructionBorderBoothImg30	constructionBorderBoothImg
	hard hat	constructionHardHatImg30	constructionHardHatImg
	offroad 1	offroadAltImg30	offroadAltImg
	offroad 2	offroadAltAltImg30	offroadAltAltImg

Additional Icons

The Saber guide includes a few more icons for which only a 30×30 or 40×40 icon are available. Since none of these looks particularly useful, I didn't bother to document them.

Saber also documents a few alternative 40×40 icon names for which the icon is in an unoutlined style. These look worse than the other icons, so I see no reason to use them.

There are more icons available in the game, but without documentation from Saber, there's no way to discover their icon names.

There is no known way to create new icons and integrate them into the game.

Rock Model Reference

Rocks tend to look good at many scales. The much stronger constraint on rock scale is the accuracy of the collision mesh when the player is rock crawling. A lot of small, overly detailed rocks are bad for performance. On the other hand, scaling up a simplified collision mesh causes the truck's tires to visually float above or sink into the rock surface.

Thus, each rock model has a recommended range of scales that it works best at. Some of these values are derived from Saber's recommendations, and some are my own. I've checked them all, but with the caveat that I'm driving blind when checking the shape of collision meshes. You might find that some orientations work better for rock crawling than others.

* represents any continuation of the model name.

Models suitable for rock crawling; these have collision meshes that closely match the visual appearance:

- rock_03, rock_03_rus scale = 3 – 5
- rock_03_pile* scale = 2 – 4
- rock_03_*_objective scale = 2 – 4
- rock_04_rus scale = 6 – 10
- rock_04a* scale = 3 – 5
- rock_05 scale = 4 – 7
- rock_06* scale = 6 – 10
- rock_rus_ter_01 scale = 6 – 10

Models suitable for blocking a truck's progress; these have collision meshes that approximate the visual appearance:

- rock_01 scale = 1
- rock_02* scale = 1
- rock_07* scale = 1 – 2

Models unsuitable for any purpose; these have really bad collision meshes:

- rock_rus_ter_01_objective

In case you're curious, the rock_06* models are related to each other as follows:

- rock_06a_cap is the top of rock_06a
- rock_06_cap is one end of rock_06a_cap
- rock_06_cap is also the **bottom** of rock_06
- rock_06a_rus is a recolor of rock_06a
- rock_06a_rus_inst is identical to rock_06a_rus, but has some useful XML properties that Saber forgot to include in the rock_06a_rus model.

Multi-Stage Model Reference

Multi-stage models change state in response to completion of an objective stage (page 328) or delivery of cargo (page 338). The placement of the model itself is described on page 131.

Below are listed all of the available multi-stage models. The model asset list provides an overview of the model appearance, but this reference provides the specific names used for each model stage and summarizes its appearance in each stage.

For each model, the stages are listed in this format:

- <stage_name>: <description of stage>

The first stage listed is the default stage if the no stage is directly associated with an initial objective state.

Model stage transitions are animated only if marked with “[\(animated\)](#)”. By default, each animation is followed by an animated camera, although the player can interrupt it to return to the driving view as the animation completes. If an animation has “[no eagle-eye view](#)”, then the animation can only be watched from the driving view.

Bug: The animation camera sometimes mysteriously fails to operate, so you may at times see different models with no eagle-eye view than I did.

If the description of a stage is followed by an asterisk (*), then the new model stage could easily collide with a parked truck. There may be additional such cases that I didn’t notice.

There is often an increased chance of a collision if you use model stages in a different order than the default (e.g. to add a barrier instead of remove a barrier with the models below). Also, animations play only in the default stage order.

A couple of models cause the Editor to throw a warning when they are loaded and/or packed, but I lost my notes about which ones. The warnings seem to be harmless.

Remove Barrier

These models are simply visible in one state and hidden in the other, thus opening a path where the model used to be.

The first set of models all use the same pair of stage names:

concrete_block_03_objective
rock_03_objective
rock_06_objective
rock_rus_ter_01_objective
ghostcity_debris_01_objective
ruins_objective
rus_broken_pumptower_01_objective
rus_high_tower_01_objective
wooden_blockage_01_objective

- stage_visible: visible model blocks passage
- stage_hidden: model is hidden and does not block passage

container_03_objective

- build_complete: model visible (yes, this is the default stage)
- build_stage_0: model hidden

rocket_broken_01_objective

- state_start: model visible
- state_end: model hidden

pond_objective ([animated](#))

- stage_visible: visible piles of dead trees and an overturned railroad car, all partially submerged
- stage_hidden: model hidden

Open Barrier

These models open a path, similar to the above, but by moving things around in the model, rather than simply hiding them.

barrier_01_objective ([animated](#))

- stage_closed: barrier closed
- stage_opened: barrier open

pipe_01_objective

- state_start: barriers in front of a broken pipe
- state_end: barriers removed and pipe fixed

rus_factory_rocket_01_01_objective (animated)

- stage_0: barriers in front of closed doors
- stage_1: barriers pulled aside and doors open

rus_launchpad_hangar_01_doors (animated)

- stage_0: doors closed
- stage_1: doors open

Deconstruct Structure

These models remove part of a model, in most cases leaving only the foundation. These would presumably be a source of cargo rather than a destination.

abandonned_oil_derrek_01_objective (animated)

- build_stage_0: tower erected
- build_complete: only base remains

rus_constr_bricks_01_objective

- stage_0: partially constructed building with construction material ready to be used
- stage_1: building unchanged; construction material removed

constr_brick_01_objective**us_constr_brick_01_old_objective**

- state_start: derelict building on foundation pad
- state_end: foundation pad only

constr_metal_01_objective**us_constr_metal_01_old_objective**

- state_start: building frame on foundation pad
- state_end: foundation pad only

constr_wood_01_objective

- state_start: building frame on concrete foundation
- state_end: concrete foundation only

Repair Structure

These models repair a structure. This might make a path a bit easier to get through, but the broken model isn't really designed as an explicit barrier to passage.

pole_a_objective

- state_start: pole lying down
- state_end: pole standing up with wires attached

power_lines_01_objective

- state_start: distribution tower collapsed
- state_end: distribution tower repaired (no wires)

rus_broken_pumptower_02_objective

- state_start: pump tower collapsed
- state_end: pump tower repaired

pipe_01_rus_objective

- build_stage_0: broken supports with missing pipe section (not a barrier to passage)
- build_complete: fixed supports with high section of pipe

Build Bridge

These bridge models start with just barriers or with a non-supportive framework and finish with a complete bridge.

bridge_road_01_a_objective [\(animated\)](#)

- build_stage_0: bridge supports and steel underframe only; concrete barriers blocking road access
- build_stage_1: + steel topframe
- build_complete: + wood roadbed and side rails; concrete barriers removed

bridge_road_01_sn_objective [\(animated\)](#)

- build_stage_0: bridge supports and steel underframe only; concrete barriers blocking road access
- build_complete: + wood roadbed and side rails, steel topframe; concrete barriers removed

bridge_road_01_b_objective [\(animated\)](#)

- build_stage_0: bridge supports and steel underframe only; concrete barriers blocking road access
- build_stage_1: + steel siderails and roadbed frame, wood roadbed; concrete barriers removed

rus_bridge_road_01_b_objective [\(animated\)](#)

- build_stage_0: bridge supports and steel underframe only; concrete barriers blocking road access
- build_stage_1: + steel siderails and roadbed frame, wood roadbed; concrete barriers removed

rus_bridge_road_001_objective ([animated](#))

- build_stage_0: bridge supports and steel underframe only; concrete barriers blocking road access
- build_complete: + steel siderails and roadbed frame, wood roadbed; concrete barriers removed

rus_bridge_road_01_objective ([animated](#))

- build_stage_0: bridge supports and steel roadbed frame only; concrete barriers blocking road access
- build_stage_1: + partial steel topframe and roadbed tension cables
- build_complete: + remaining topframe and roadbed; concrete barriers removed

rus_bridge_road_01snow_objective ([animated](#))

- build_stage_0: bridge supports and steel roadbed frame only; concrete barriers blocking road access
- build_stage_1: + partial steel topframe and roadbed tension cables
- build_complete: + remaining topframe and roadbed; concrete barriers removed

rus_bridge_road_big_objective ([animated](#))

- build_stage_0: bridge supports and steel roadbed frame only; concrete barriers blocking road access
- build_stage_1: + partial steel topframe
- build_stage_2: + remaining topframe, railbed, and rails

bridge_wooden_big_02_objective ([animated](#))

- build_stage_0: bridge supports and wood underframe only; wooden barriers blocking road access
- build_stage_1: + wood roadbed; wooden barriers removed

bridge_logs_01_objective

- build_stage_0: wooden side rails only
- build_complete: + log roadbed

Build Other Structure

These models build a structure from a foundation or from nothing.

Drilling Sites

oil_derrek_01_objective ([animated](#))

- build_stage_0: base only
- build_stage_1: + lower structure and conveyer
- build_stage_2: + tower
- build_complete: + equipment

rus_drilling_rig_01_objective ([animated](#))

- build_stage_0: base only
- build_stage_1: + lower structure
- build_stage_2: + mid structure
- build_complete: + tower and conveyer

rus_drilling_rig_02_objective (animated)

- build_stage_0: frame and lower walls
- build_stage_1: + upper walls
- build_stage_2: + minor embellishments
- build_complete: + roof

rus_drilling_rig_03_objective (animated, but no eagle-eye view)

- build_stage_0: frame and end walls
- build_stage_1: + remaining walls
- build_stage_2: + roof
- build_complete: + embellishments

Paper Factory

Each section of the paper factory is split into two models: a static model, and an animated model that should have the same location and orientation. My guess is that Saber did this to reduce the size of each model. Unfortunately, the Editor throws warnings about the duplicate model locations.

Tip: You can use the ‘Do not show this warning again’ checkbox to disable the warnings, but they’ll come back the next time you restart the Editor. Instead, I suggest moving the **_objective** model down by 1 mm (decrease its Y coordinate value by 0.001).

paper_factory01_objective (animated) (should be placed with paper_factory01_stage_0)

- build_stage_0: nothing
- build_complete: everything

paper_factory02_objective (animated, but no eagle-eye view) (should be placed with paper_factory02_stage_0)

- build_stage_0: nothing
- build_complete: everything

The animation camera movement is strange for paper_factory03_objective. Presumably there should be a nearby building that the camera wants to include in the view.

paper_factory03_objective (animated) (should be placed with paper_factory03_stage_0)

- build_stage_0: nothing
- build_complete: everything

paper_factory04_objective (animated) (should be placed with paper_factory04_stage_0)

- build_stage_0: nothing
- build_complete: everything

paper_factory05_objective ([animated, but no eagle-eye view](#)) (should be placed with paper_factory05_stage_0)

- build_stage_0: nothing
- build_complete: everything*

paper_factory06_objective ([animated, but no eagle-eye view](#)) (should be placed with paper_factory06_stage_0)

- build_stage_0: nothing
- build_complete: everything

US Three-Part Factory

us_3part_factory_1_part_objective ([animated](#))

- build_stage_0: building and boilers
- build_stage_1: + rooftop and boilertop additions
- build_complete: + smokestacks and other details

us_3part_factory_2_part_objective ([animated](#))

- build_stage_0: main structures and pipe foundations
- build_stage_1: + connecting pipes and silotop additions
- build_complete: + more pipes and embellishments

us_3part_factory_3_part_objective ([animated](#))

- build_stage_0: main structures and foundations for silos and pipes
- build_stage_1: + silos and taller buildings
- build_complete: + pipes and rooftop stuff

Russian Fuel Factory

rus_fuel_factory_03_objective ([animated](#))

- build_stage_0: building and boilers
- build_stage_1: + rooftop and boilertop additions
- build_complete: + smokestacks and other details

rus_fuel_factory_04_objective ([animated](#))

- build_stage_0: main structures and pipe foundations
- build_stage_1: + connecting pipes
- build_complete: + more pipes and embellishments

rus_fuel_factory_05 is split into two models similar to the paper factory above. (In fact, these models essentially duplicate paper_factory03 except for the textures.)

rus_fuel_factory_05_objective ([animated](#)) (should be placed with rus_fuel_factory_05_stage_0)

- build_stage_0: nothing
- build_complete: everything

Russian Base

rus_base_01_building_objective ([animated](#))

- build_stage_0: metal building frame
- build_complete: building complete

rus_base_02_building_objective ([animated](#))

- build_stage_0: metal building frame
- build_complete: building complete

Russian Control Bunker

rus_controlbunker_02_1_part_objectives ([animation broken](#))

- build_start: building foundation
- build_stage_1: + walls, poles, and wires (after a delay caused by the broken animation)*
- build_end: + no change

rus_controlbunker_02_2_part_objectives ([animated](#))

- build_start: building foundation and derelict helicopter
- build_stage_1: building foundation (helicopter gone)
- build_end: fixed foundation and new helicopter

rus_controlbunker_02_3_part_objectives ([animation broken](#))

- build_start: derelict building
- build_stage_1: new building (after a delay caused by the broken animation)
- build_end: + no change

rus_controlbunker_02_4_part_objectives ([animation broken](#)) (works best on top of a building to avoid rotating through a truck)

- build_start: broken dish
- build_stage_1: fixed dish, stationary (after a delay caused by the broken animation)
- build_end: fixed dish, rotating (after a delay caused by the broken animation)

Other Buildings

conveyor_objective_01

- state_start: building frames and supplies
- state_end: complete buildings and connecting conveyer*

radio_station_rus_objective

- state_start: bare tower and concrete foundation for equipment
- state_end: complete tower and equipment*

us_factory_sorting_01_objective ([animated](#))

- build_stage_0: buildings
- build_stage_1: + rooftop structure and silos
- build_complete: + more rooftop stuff and connecting covered conveyer

us_mine_01_objective ([animated](#))

- build_stage_0: buildings
- build_stage_1: + more building height and some roofs
- build_complete: + final roofs, connecting conveyers, and embellishments

rus_pier_01_objective ([animated](#))

- build_stage_0: pilings only
- build_stage_1: pier complete[†]

rus_pier_02_objective ([animated](#))

- build_stage_0: pier missing most top slabs; some pilings leaning or missing
- build_stage_1: pier repaired, but only 110 m long compared to the original 145 m of broken pier[†]

tatra_facility_objective

- build_stage_0: first story complete, but old
- build_stage_1: + wood frame on second story
- build_stage_2: + second story covered; new paint everywhere

tatra_warehouse_objective

- build_stage_0: first story complete, but old
- build_stage_1: + wood frame on second story; first story partially repainted
- build_stage_2: + second story covered; new paint everywhere

[†] The game animates the pier being repaired from one end to the other, but it doesn't add any physical support until the very end of the animation. If the player doesn't wait for the animation to complete, she may attempt to drive onto a visually complete section of the pier too soon.

Specialty Models

I'm not sure if trailer_train_rocket_01_collision was a test model that got released accidentally, or whether it's intended to keep trucks out of trailer_train_rocket_01_objective prior to build_stage_1. If the latter, then you still need to advance trailer_train_rocket_01_collision to build_stage_1 somehow, e.g. by specifying it as the only model stage used by one of the initial contracts.

trailer_train_rocket_01_collision

- build_stage_0: no visible model; no collision box
- build_stage_1: no visible model; collision box for train only
- build_stage_2: no visible model; collision box for train only

trailer_train_rocket_01_objective ([animated](#))

- build_stage_0: idle gantry only
- build_stage_1: train pulls in and erects rocket in gantry*
- build_stage_2: rocket launches, leaving idle gantry and empty train

bridge_and_train_objective ([animated](#)) (triggers Editor warnings about the model, but it seems to work OK)

(requires a fair amount of straight, level track on either end for the train to move along)

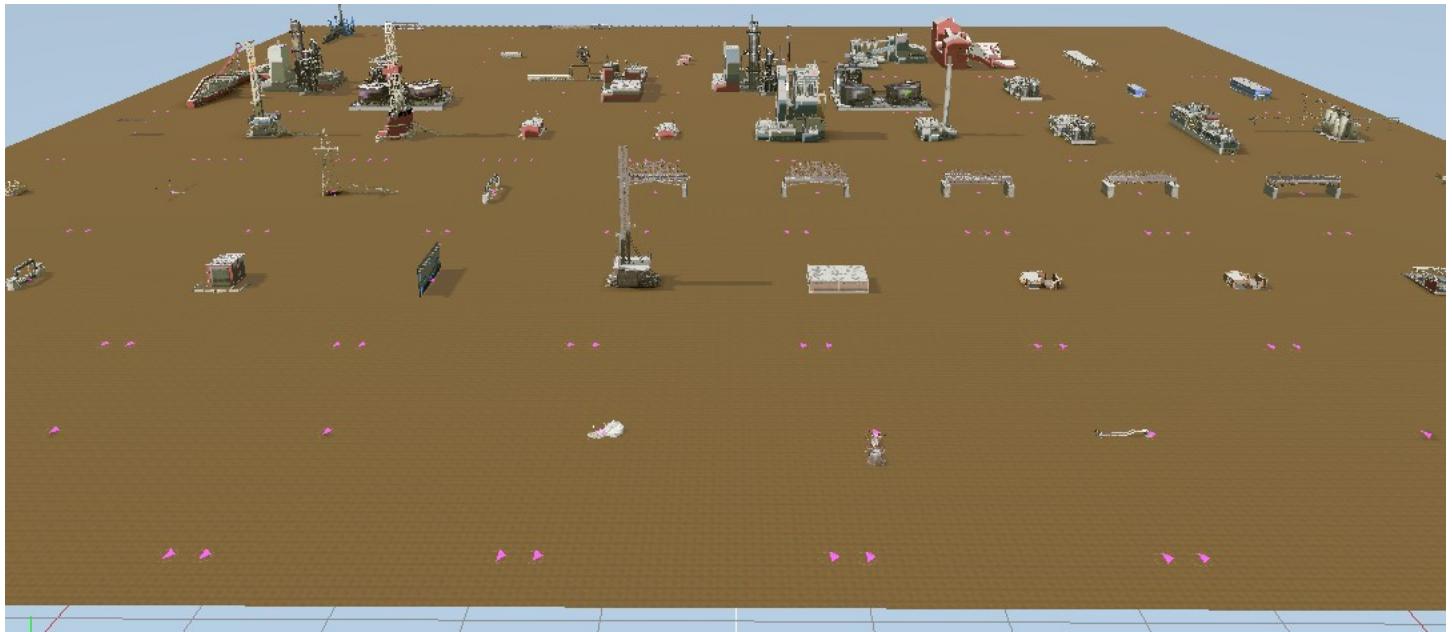
- build_stage_0: rail frame with steel upperframe at one end only
- build_complete: + finished frame and train rails; trail rolls through backwards and disappears in the distance
- train_hide: + no chage (train is already hidden)

rail_blocker_objective_01

- build_stage_0: train rail section with missing portion; barriers at each end of section
- build_complete: complete train rail section; no barriers

train_carriage_objective

- empty_carriage: empty train carriage
- cargo_carriage: train carriage carrying sequoia log
- hide_cargo_carriage: model hidden



Appendix B: Hand Edit Text Files

The SnowRunner Editor makes it easy to edit your map and immediately see your changes. However, you sometimes need to do something that requires more power than what the Editor offers or that is simply easier done outside the Editor. The following sections describe what you need to know in order to read and edit text files outside the SnowRunner editor. See also page 437 for how to edit bitmap files using an image editor.

Useful text files include the following:

- the XML and JSON files that are normally edited by the SnowRunner Editor.
- other miscellaneous files that you have to create for yourself in order to integrate with your map (e.g. localization and ambient music definitions).
- class files that define features that you can use on your map.

Position Coordinates

As mentioned on page 50, the X and Z axes define the ground plane, and the Y axis points up toward the sky. This section describes the coordinate systems in more detail.

Position Coordinates in Meters

An object or vertex that is always attached to the ground (such as a zone locator) has only X and Z coordinates in properties named `Position *` or `Org *`. Objects and nodes that can be detached from the ground have X, Y, and Z coordinates in properties named `Position.*` or `Position *`.

A camera position is formatted as `(x, y, z)` in properties named `cameraPos*`. Or the Editor's camera position is displayed as `(x; y; z)` when the `Statistics` toolbar button is enabled (page 256).

All object, node, and camera coordinates are measured in meters.

X and Z are measured from the center of the map:

- Positive X values are in the eastern half. Negative X values are in the western half.
- Positive Z values are in the northern half. Negative Z values are in the southern half.

Each position on the terrain surface has a Y coordinate that ranges from 0 at the lowest to the `Blocks.Max Height` value at the highest (page 72). Mud and snow can deform the terrain slightly beyond these limits.

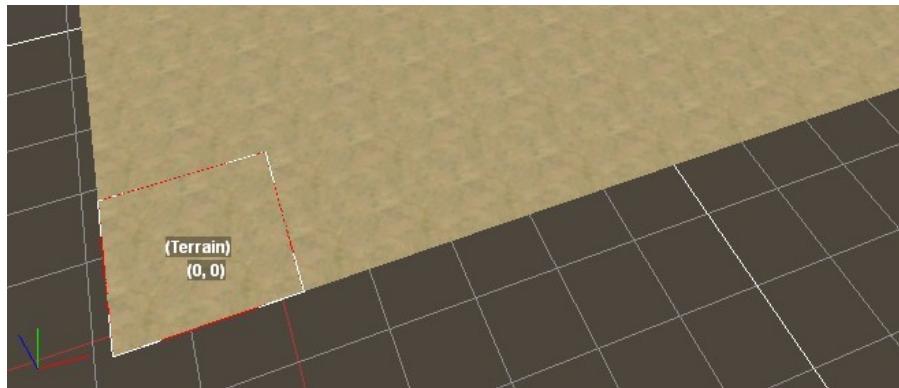
Object, node, and camera position coordinates are effectively unbounded in all three axes. The X and Z coordinates can be beyond the edges of the map, and the item is placed at that position. The Y coordinate can be

less than 0 (for an item sunken in the ground) or greater than the [Blocks.Max Height](#) value (for high-floating items).

Terrain Block Coordinates

The SnowRunner Editor doesn't directly use terrain block coordinates for anything. However, enough features have special cases based on terrain blocks that it's useful to understand their layout.

When you left click on the terrain, you select a terrain block. This block is 24 meters on a side (X and Z) and is displayed however tall (Y) the terrain is within that block. The terrain block is labeled with coordinates in (X, Z) format, with (0, 0) being the southwest corner of the map.



The map size is specified in the map's [Blocks.Number X](#) and [Blocks.Number Z](#) properties (page 70).

Orientation Coordinates

Not only do objects have a position, they also have an orientation. This section describes the orientation values in detail.

2-D Orientation of Reference Maps

A reference map is attached to the ground and can rotate only in a single plane. The Editor displays its orientation as an angle, [Angle](#). The angle is measured as degrees of clockwise rotation from the default orientation. A negative value represents counterclockwise rotation.

When you rotate the reference using the interface widget, the Editor cycles the Angle through the range of -180 to 180 (degrees).

If you type in new values for the orientation vector, you can enter values outside of the normal range. The Editor preserves the values though saves and loads of the XML file. Only when you rotate the object using the widget does the Editor revert the angle to its normal range.

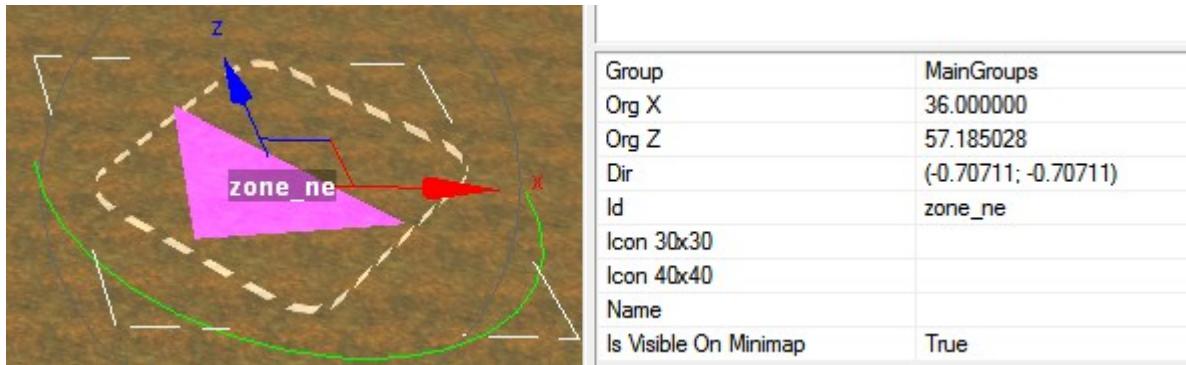
When the Editor saves the orientation angle in the XML file, it converts the angle to radians. E.g.

```
<Reference
  ...
  Angle="2.35619"
  ...
/>
```

Note that the saved value has fewer digits after the decimal point (5) than what is displayed value in the Editor (6). This is common for most numeric property values, regardless of whether they go through a conversion step when saving.

2-D Orientation of Zones

A zone is attached to the ground and can rotate only in a single plane. The Editor displays its orientation, **Dir**, as a pair of values in (X; Z) format.



These two values represent a unit vector specifying how close to the X axis the object points vs. how close to the Z axis it points.

If you type in new values for the orientation vector, you could create a vector that is not normalized to unit length. The Editor preserves the non-normalized values though saves and loads of the XML file. Only when you rotate the object using the main panel widget does the Editor recalculate a normalized orientation vector.

Bug: The orientation vector should be unit length; otherwise the zone may be drawn with a distorted size, and the game may perform the zone's functions using a **different** distorted size.

When the object is created, it initially points directly east. In other words, it points entirely along the X axis and not at all along the Y axis, so its direction vector is (1; 0).

Negative values in the direction vector means that the object points some amount to the west or south.

For the example zone in the screenshot above, the direction vector is $(-0.70711; -0.70711)$. This means that it points west and south. Note the normalized unit vector: $(-0.70711)^2 + (-0.70711)^2 = 1$.

The Editor saves 2-D orientation coordinates in the `zones.xml` file in the same format as it displays in the Editor. E.g.

```
<Zone  
...  
Dir="(-0.70711; -0.70711)"  
Id="zone_ne"  
...  
/>
```

3-D Orientation of Trucks, Models, and Plants in the Editor

For an object that can rotate freely in three dimensions, the Editor displays its orientation with three values measured in degrees: `Rotation.X`, `Rotation.Y`, and `Rotation.Z`.

Rotation	
X	44.999996
Y	90.000000
Z	45.000000

Each `Rotation` value specifies the rotation around the corresponding axis as degrees of counterclockwise rotation when looking in the direction of the axis or clockwise rotation when looking against the axis.

Bug: `Rotation.X` and `Rotation.Z` are swapped for trucks. I.e. increasing the `Rotation.X` value rotates the truck counterclockwise around the `Z` axis, and increasing the `Rotation.Z` value rotates the truck counterclockwise around the `X` axis.

For trucks, `Rotation.Y` is applied first (around the Y axis). Then, in the new local coordinate system after the first rotation, `Rotation.X` is applied (around the new Z axis orientation). Finally, `Rotation.Z` is applied (around the new X axis orientation).

Bug: `Rotation.X` and `Rotation.Y` are swapped for models and plants. I.e. increasing the `Rotation.X` value rotates the model or plant counterclockwise around the `Y` axis, and increasing the `Rotation.Y` value rotates the model or plant counterclockwise around the `X` axis.

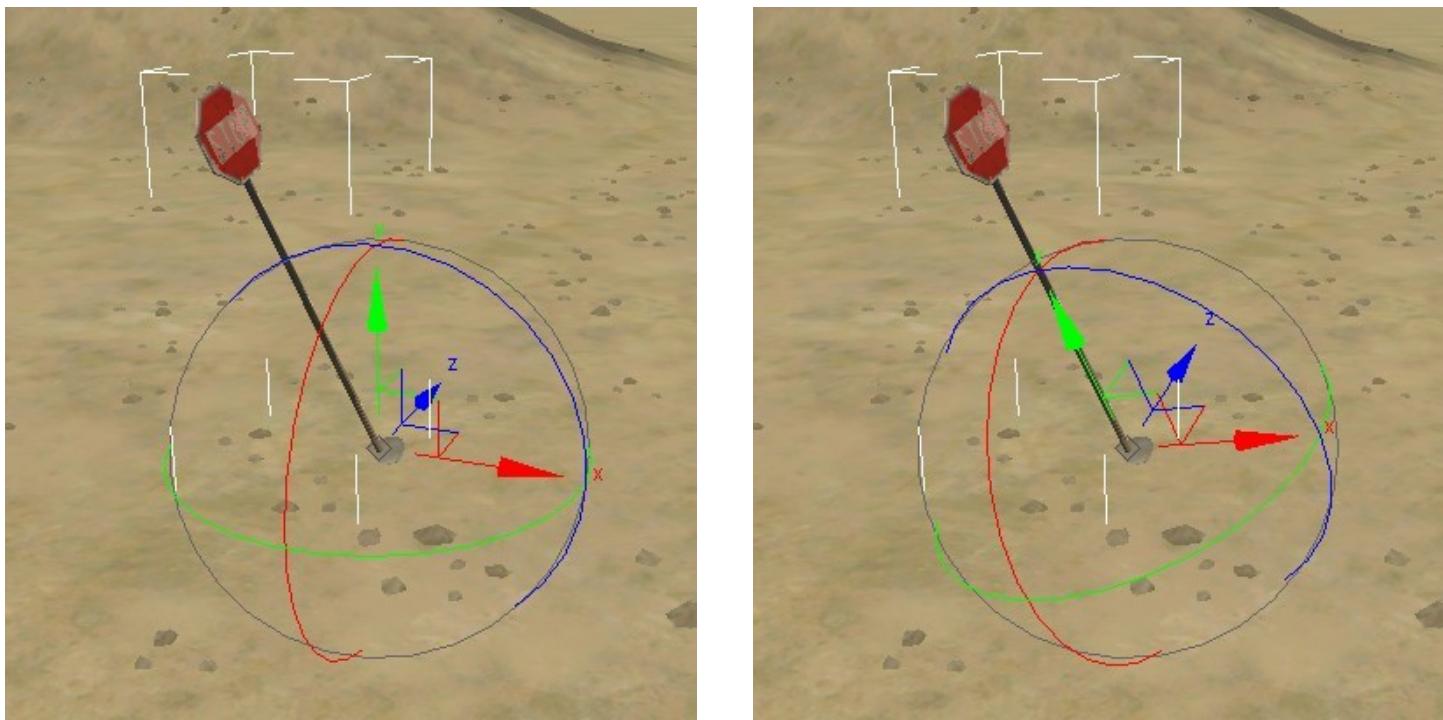
For models and plants, `Rotation.X` is applied (around the Y axis). Then `Rotation.Y` is applied (around the new X axis orientation). Finally `Rotation.Z` is applied (around the new Z axis orientation).

There are many combinations of `Rotation.X`, `Y`, and `Z` that result in the same orientation. If you use the main panel widget to adjust the rotation or if you reload the map, the Editor converts the `Rotation` values to its preferred combination.

3-D Orientation of Trucks, Models, and Plants in the XML File

Unlike what is displayed in the Editor, a 3-D object's orientation is saved in the XML file as a pair of normalized 3-D vectors called `Dir` and `Up`.

The screenshots below show the same view of a tilted stop sign with two coordinate axes superimposed. The left image shows the X, Y, and Z axes relative to the map (global space). The right image shows the X, Y, and Z axes on which the object was created (local space). These screenshots will be the reference for the following discussion.



The direction vector (`Dir`) is a normalized unit vector that specifies the direction of the blue Z **local** axis relative to the **global** coordinate space. The length of the direction vector doesn't matter, so the Editor normalizes it to unit length (1 meter).

When the object is initially created, its orientation is precisely aligned with the global coordinates. The direction of its Z axis therefore points entirely along the global Z axis and not at all in the direction of the global X or Y axes. Thus, its direction is $(0; 0; 1)$. Since this direction is the default, the Editor generally omits it from the XML file. So if you see a 3-D object in the XML file without a direction, assume $(0; 0; 1)$.

If the object is rotated or tilted, its local Z axis points along some combination of the global X, Y, and Z axes. For the stop sign example, its **Dir** is (0.061; 0.277; 0.959). Its local Z axis points mostly in the direction of the global Z axis, but also a little bit in the positive direction of the global X and Y axes (east and north). Note the normalized unit vector: $0.061^2 + 0.277^2 + 0.959^2 = 1$.

The 3-D direction vector indicates the direction of the local Z axis relative to the global coordinate space, but the object can rotate freely around its Z axis without changing this vector. More information is required to fix the object to a single orientation. The XML file includes this extra information as the **Up** vector. It is determined in the same way as the direction vector, but using the local Y axis instead of the local Z axis.

For the stop sign example, its **Up** is (-0.303; 0.921; -0.246). Its local Y axis points mostly in the direction of the global Y axis (up), but also a little bit west and south.

The default **Up** vector is (0; 1; 0). Many objects (such as houses and stop signs) will be rotated in the ground plane without being tilted. An object whose Y axis points directly up may have its **Up** vector omitted from the XML file.

It is possible to modify the **Up** vector in the XML file to be inconsistent with the **Dir** vector. In that case, the Editor uses the given **Dir** vector and modifies the **Up** vector to be consistent. However, the Editor preserves the inconsistent values though saves and loads of the XML file. Only when you rotate the object through the main panel widget does the Editor recalculate and save a consistent **Up** vector.

```
<Position  
    Org="-0.81737; 20.29656; -9.9487"  
    Up="(0.23974; 0.95614; 0.16827)"  
    Dir="(0.86656; -0.2889; 0.40695)  
/>
```

3-D Orientation of the Camera

A camera orientation is formatted as **(x, y, z)** in properties named **cameraDir***. Or the Editor's camera orientation is displayed as **(x; y; z)** when the **Statistics** toolbar button is enabled (page 256).

The **x**, **y**, and **z** values form a 3-D vector that specifies the direction that the camera faces. The length of the direction vector doesn't matter. The Editor normalizes its displayed camera position to unit length, but when entering a **cameraDir*** value, you can specify any length vector.

Unlike the 3-D object direction vectors specified in the section above, there is no "up" vector. Instead, the camera is always oriented level with the horizon. The only exception is when the camera is pointed straight up or straight down, in which case it is not possible to define a horizon level. Instead, when the camera is pointed straight down, "up" relative to the camera is forced to due west, and when the camera is pointed straight up, "up" relative to the camera is forced to due east.

Tip: The easiest way to get a good camera position and orientation is to position the camera in the Editor, then enable the **Statistics** toggle button in the toolbar. The statistics list the camera position followed by its orientation: $(X; Y; Z) > (X; Y; Z)$.

Camera: (-26.68; 30.19; -25.96) > (0.88; -0.24; 0.40)

3-D Orientation of Wire Nodes

The orientation of wire nodes (page 198) is displayed in the Editor and recorded in the XML file as quaternions with four values. Orientation quaternions are outside the scope of this documentation, but you can find plenty of documentation on the web. (TL,DR: they require a lot of math to understand.)

SnowRunner Built-In Assets

The SnowRunner installation directory is mostly inscrutable, but it does contain a few ***.pak** archives that contain files that include useful information regarding available assets. You can browse these files to help figure out how certain assets work, or you can make your own copy of certain files to package into your map.

The installed ***.pak** archives are in a path that looks something like this, depending on whether you have the Epic or Steam version of the game:

```
C:\Program Files\Epic Games\SnowRunner\en_us\preload\paks\client  
C:\Program Files (x86)\Steam\SteamApps\common\SnowRunner\preload\paks\client
```

The specific archives that I found interesting are as follows.

- **initial.pak**: contains all of the asset class definitions as well as localization text
- **shared_sounds.pak**: contains all of the ambient sounds used in the game

The ***.pak** archives are secretly just ZIP archives with a different file extension. You can extract the contents of a ***.pak** archive directly with a full-featured archive manager such as 7-Zip. Or you can copy an archive to a new name ending in **.zip**, then browse it using Windows Explorer.

Various sections of this book mention when useful information can be found in these archives.

Keep in mind that SnowRunner updates may change the contents of these archives at any time.

Map XML Files

The SnowRunner Editor makes it easy to edit your map and immediately see your changes. However, you sometimes need to do something that requires more power than what the Editor offers or that is simply easier done outside the Editor. The following sections describe what you need to know in order to edit files outside the SnowRunner Editor.

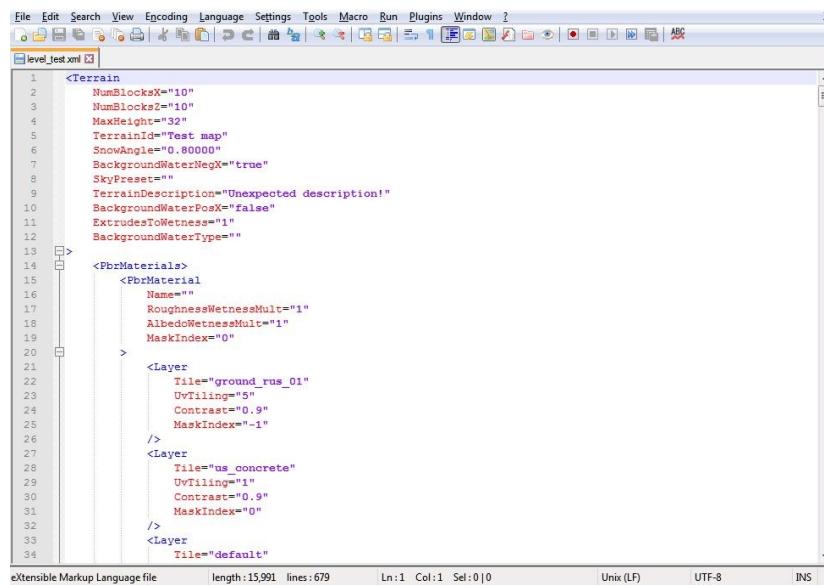
The main SnowRunner Editor keeps a copy of all named property values for a map in the following XML files:

- `prebuild/map_name.xml` – the top-level XML file; has all properties not included in one of the below files
- `prebuild/map_name/_sounds.xml` – ambient sounds, point sounds, sound domains, and river markups
- `prebuild/map_name/_zones.xml` – zone locators
- `prebuild/map_name/subgroups/Features_group_name.xml` (zero or more of this file type) – features in groups

Each XML file is just a text file, so it is easy to read and modify. Editing the XML outside the Editor is useful for tasks that are difficult or impossible to do in the SnowRunner Editor. For example, you can rearrange the trucks so that the active truck is first. Or you can write a script to arrange small fir trees in a grid to represent a Christmas tree farm.

Double-clicking the top-level XML file in the `File View` opens the map in the Editor. To open the top-level XML file in a different editor, select the `prebuild` directory in the `File View`, then click the `Show in Explorer` button icon in the `File View` toolbar. This opens the directory in Windows Explorer. Double click the XML file in Windows Explorer to open the file in your XML editor.

If you don't already have an XML editor installed, Wordpad works fine, or you can install Notepad++ or many other text editors if you want extra support for XML.



The screenshot shows the Notepad++ interface with the XML file 'level_test.xml' open. The code is as follows:

```

1  <Terrain
2    NumBlocksX="10"
3    NumBlocksZ="10"
4    MaxHeight="32"
5    TerrainId="Test map"
6    SnowAngle="0.80000"
7    BackgroundWaterRegX="true"
8    SkyPreset=""
9    TerrainDescription="Unexpected description!"
10   BackgroundWaterFoxX="false"
11   ExtrudedToWetness="1"
12   BackgroundWaterType=""
13   >
14   <PbrMaterials>
15     <PbrMaterial
16       Name=""
17       RoughnessWetnessMult="1"
18       AlbedoWetnessMult="1"
19       MaskIndex="0"
20     >
21     <Layer
22       Tile="ground_rus_01"
23       UV Tiling="5"
24       Contrast="0.9"
25       MaskIndex="1"
26     />
27     <Layer
28       Tile="us_concrete"
29       UV Tiling="1"
30       Contrast="0.9"
31       MaskIndex="0"
32     />
33     <Layer
34       Tile="default"
35     >

```

The XML code defines a terrain with specific properties like height, snow angle, and background water settings. It also includes material definitions and layer configurations for different parts of the map.

The internet has lots of resources to describe the format of XML files, but I'll give a short introduction. All properties are stored hierarchically in named tags. For example, truck properties are under the `<Trucks>` tag, with one `<Truck>` tag for each individual truck. If a tag contains sub-tags, then the container eventually includes a closing tag, e.g. `</Truck>`. If a tag contains only properties and no sub-tags, then the tag is opened and closed with a single tag, e.g. `<Position ... />`.

The properties for each feature should look familiar, although the exact names and formatting differ slightly from what is in the Editor window. The XML file is generally case sensitive, so you must use uppercase and lowercase exactly as you see it.

If a map property has its default value, the Editor may decline to include it in the XML file. To see how a property should appear in the XML, save a non-default value in the Editor before inspecting the XML file.

If you make changes to the top-level XML file while the SnowRunner Editor is open, the Editor pops up a dialog asking whether it should load the changes. Click `Yes` to do so. If you make changes to the other map XML files, you need to reload the map for the Editor to see the changes.

The SnowRunner Editor generally performs reasonably if the XML is well formed but the values in it don't make sense. It will make values self consistent where it can, and if some feature makes no sense at all, it simply won't display it. Thus, you can freely experiment with different values without fear that everything will get screwed up. (That said, "undo" is your friend in the XML editor.)

You can sometimes edit the XML to exceed the limits imposed by the Editor (e.g. `Fuel="200"`), but you do so at your own risk. Beware that something might be accepted by the Editor but not work in the game itself.

Map and Region JSON Files

The Zone Settings Editor saves its property values in JSON files:

- `levels/map_name/level_desc.json` – everything under `popupSettings` and `levelDesc`
- `levels/map_name/zone_settings.json` – zone types (under `map_name`) and the top-level checkboxes
- `levels/map_name/objective_settings.json` – everything under `objectiveSettings`
- `levels/map_name/content_settings.json` – everything in the `Content Settings` area

The Region Settings Editor also saves its property values in JSON files:

- `prebuild/region_name/regionDesc.json` – everything in the `Region desc` tab
- `prebuild/region_name/local_zone_settings.json` – zone types (under `Region` → `MAPS LIST`)
- `prebuild/region_name/local_objective_settings.json` – everything under `objectiveSettings`
- `prebuild/region_name/content_settings.json` – everything in the `Content Settings` tab

Don't try to hand-edit the following derived files:

- `prebuild/region_name/zone_settings.json` – combined zone types at the map and region levels
- `prebuild/region_name/objective_settings.json` – combined objectives at the map and region levels

If you make changes to a JSON file, you need to quit and reload the Zone Settings Editor or reload the region in the Region Settings Editor to see the changes.

Bug: The Settings Editors are quite intolerant of syntax errors in the JSON files and are likely to overwrite your changes with a blank file! I recommend that you make a copy of the files before making any changes.

Distribution Brushes: brushes.xml

You can make your own brushes for use in distributions (page 147).

Create new brushes in

`%USERPROFILE%/Documents/My Games/SnowRunner/Media/classes/editor/brushes.xml`. You may need to create the `classes/editor` directory.

Examples of distribution brushes can be found in the `classes/editor/brushes.xml` file within the built-in `initial.pak` package (page 426). This file also contains other types of brushes. Descriptions of these other brushes is TBD.

Tags and attributes in the `brushes.xml` file are TBD. (Sorry, this section is just a stub for now.)

The Editor ignores any distribution brushes that refer to a plant that doesn't exist.

Mutators: mutators.xml

Your map can mutate references to fit your scenery (page 245).

You can create your own mutators in

`%USERPROFILE%/Documents/My Games/SnowRunner/Media/classes/editor/mutators.xml`. You may need to create the `classes/editor` directory.

Caution: Once the Editor has read your custom `mutators.xml` file, you must reload your map to force it to re-read the file. Otherwise the Editor will continue to use a cached copy and won't see any changes you've made to the file. (The `Reload Resources` toolbar button does something, but whatever it does isn't reliable.)

The `mutators.xml` file contains a section for each mutator, and each mutator handles all mutations for one or more maps. Below is an example with one mutator:

```
<Mutators>
  <Mutator Name="rus_example">
    <Mutations>
      <ModelMutation      Source = "rock_03"          Target = "rock_03_rus"  />
      <ModelMutation      Source = "rock_06"          Target = "rock_04_rus"  />
      <ModelMutation      Source = "rock_06a"         Target = "rock_04a_rus" />
      <MaterialLayerMutation Source = "default"       Target = "grass_rus_01" />
      <BrushMutation       Source = "Birches01"        Target = "Birches01Rus" />
      <BrushMutation       Source = "Birches02"        Target = "Birches01Rus" />
      <PlantMutation       Source = "birch_01"         Target = "birch_01_rus" />
      <OverlayMutation     Source = "cliff_dirt"       Target = "cliff_dirt_rus" />
      <OverlayMutation     Source = "cliff_dirt_large" Target = "cliff_dirt_large_rus"/>
    </Mutations>
  </Mutator>
</Mutators>
```

More examples can be found in the `classes/editor/mutators.xml` file within the built-in `initial.pak` package (page 426).

Mutator Definition

```
<Mutators>
  <Mutator Name="mutator_name">
    <Mutations>
      ...
    </Mutations>
  </Mutator>
</Mutators>
```

mutator_name: text

Specifies the name of the mutator for use by the **Mutator** property (page 245).

Default: unspecified; the mutator cannot be applied by any map.

If you make multiple mutators, they must all be listed in the same **mutators.xml** file, even if the mutators are not related to each other.

Mutations

```
<Mutators>
  <Mutator Name="mutator_name">
    <Mutations>
      <ModelMutation      Source = "source" Target = "target" />
      <MaterialLayerMutation Source = "source" Target = "target" />
      <BrushMutation      Source = "source" Target = "target" />
      <PlantMutation       Source = "source" Target = "target" />
      <OverlayMutation     Source = "source" Target = "target" />
      ...
    </Mutations>
  </Mutator>
</Mutators>
```

ModelMutation declares a mutation from one model type to another.

MaterialLayerMutation declares a mutation from one material layer type to another.

BrushMutation declares a mutation from one distribution brush to another.

PlantMutation declares a mutation from one type of individually placed plant to another. This has no effect on distributed plants.

OverlayMutation declares a mutation from one overlay type to another.

source: text

Specifies the type to be mutated from.

Default: unspecified; no mutation is applied.

target: text

Specifies the type to be mutated to.

Default: unspecified; no mutation is applied.

Ambient Music: music_presets.xml

Your map can include ambient music that plays in the background. Different tracks can play at different times, and another track plays in the garage.

To set up your ambient music, create a file:

`%USERPROFILE%/Documents/My Games/SnowRunner/Media/classes/sounds/music_presets.xml`. You may need to create the `classes/sounds` directory. Note that this is **not** the same directory as where sound files are stored.

This XML file contains a section for each of your maps. Below is an example with music for only one map:

```
<MusicPresets>
    <LevelMusic LevelName="level_test">
        <Sound
            Name="bass solo"
            StartTime="0"
            EndTime="2"
            FadeInTime="15"
            FadeOutTime="7.5"
        />
        <Sound
            Name="drums"
            StartTime="19"
            EndTime="24"
            FadeInTime="15"
            FadeOutTime="7.5"
        />
        <GarageMusic
            SoundName="music/ipanema"
            FadeInTime="15"
            FadeOutTime="7.5"
            Volume="0.8"
        />
    </LevelMusic>
</MusicPresets>
```

More examples can be found in the `classes/sounds/music_presets.xml` file within the built-in `initial.pak` package (page 426).

Level Definition

```
<MusicPresets>
    <LevelMusic LevelName="map_name">
        ...
    </LevelMusic>
</MusicPresets>
```

`map_name`: text

Specifies the name of the map for which music is being defined.

Default: unspecified; the music presets are not applied to any map.

If you make multiple maps with music, they must all be listed in the same `music_presets.xml` file, even if the maps are not related to each other.

Notice that the `music_presets.xml` directly specifies which map each of its sections should apply to. Therefore, there is no need for a property in the Editor to specify which section of the `music_presets.xml` to use.

Music While Driving

Each `<Sound>` element defines music to play during a certain period of the day.

```
<MusicPresets>
    <LevelMusic LevelName="map_name">
        <Sound
            Name="sound_file"
            StartTime="start_hour"
            EndTime="finish_hour"
            FadeInTime="fade_in"
            FadeOutTime="fade_out"
        />
        ...
    </LevelMusic>
</MusicPresets>
```

The music generally plays continuously, but it can be interrupted, e.g. during a contest or while in a no-music sound domain (page 234).

Sound File

`sound_file`: text

Specifies the relative path and base filename of the sound file(s) to play.

Default: unspecified; no music plays for this `Sound` definition.

Choose a sound file or set of sound files in the same way as for a point sound (page 222). The built-in music files are in the `[sound]/music` directory of the `shared_sounds.pak` archive. The Editor displays a warning dialog if the sound file is specified but not found, but it does not display a warning dialog if the sound file is left unspecified.

Since it doesn't have an aural direction, stereo sound is officially permitted for ambient music.

In order to match the volume of the built-in campaign music (so that the player doesn't have to reach for the audio settings), Saber recommends that the music files have an internal volume of -7 dB.

Hour Range

`start_hour`: numeric, in hours after midnight, in the range 0.0 – 24.0, inclusive

Specifies the time when this **Sound** definition starts playing music.

Default: 0.0 (midnight).

`finish_hour`: numeric, in hours after midnight, in the range 0.0 – 24.0, inclusive

Specifies the time when this **Sound** definition finishes playing music.

Default: 0.0 (midnight).

The `*_hour` values can be fractional to start or finish in the middle of an hour. Values out of range are treated as 0 (midnight). The game does the right thing if `start_hour` is greater than `finish_hour`: it ends at the finish hour in the next day.

If you are using music that is tied to the time of day, refer to page 45 for when the light conditions change at the start and end of each part of the day.

If `start_hour` equals `finish_hour`, no music plays for this **Sound** definition. To play one track (or sequence) all day, set `finish_hour` to just under `start_hour`, e.g. 0.0 and 23.999. The music will awkwardly fade out and fade back in at the start of the track at midnight, but that's the best we can do.

If two **Sound** definitions overlap in time, then the game picks each track at random from either definition during the overlapping period.

Fade In and Out

`fade_in`: numeric, in seconds

Specifies how long it takes this **Sound** definition to fade in when it starts playing.

Default: ~20.

`fade_out`: numeric, in seconds

Specifies how long it takes this **Sound** definition to fade out when it stops playing.

Default: ~15.

In many cases, one **Sound** definition ends as another starts, e.g. at a particular hour, or when the player enters or leaves the garage. In this case, the fade-out of the previous music overlaps with the fade-in of the new music, so make sure they are arranged accordingly.

Bug: While not in the garage, the game fades in every track as it begins playing, even if it's still within the same the **Sound** definition. This bug does not apply to fade out, and it doesn't apply within the garage. If the sound files are short, the garage can even spread its fade in across multiple sound files.

Even with *fade_in* of 0, the music tracks are not played cleanly back to back, so don't try to make a clean loop from the end of a track to its beginning. Every track should have a built-in fade in and fade out at the beginning and end to avoid an abrupt transition.

Music in the Garage

The **<GarageMusic>** element defines music to play when the player is in the garage. If there are multiple **<GarageMusic>** definitions, the game uses only one of them.

```
<MusicPresets>
    <LevelMusic LevelName="map_name">
        <GarageMusic
            SoundName="sound_file"
            FadeInTime="fade_in"
            FadeOutTime="fade_out"
            Volume="volume"
        />
        ...
    </LevelMusic>
</MusicPresets>
```

sound_file, *fade_in*, and *fade_out* are the same as for driving music, above.

Volume

***volume*:** numeric in the range 0.0 to 1.0

Specifies the volume of the music as a fraction of its volume in the sound file.

Default: 1.0.

Depending on your choice of music, it might sound a little loud compared to the quiet of the garage. I presume that's why the garage has a volume control.

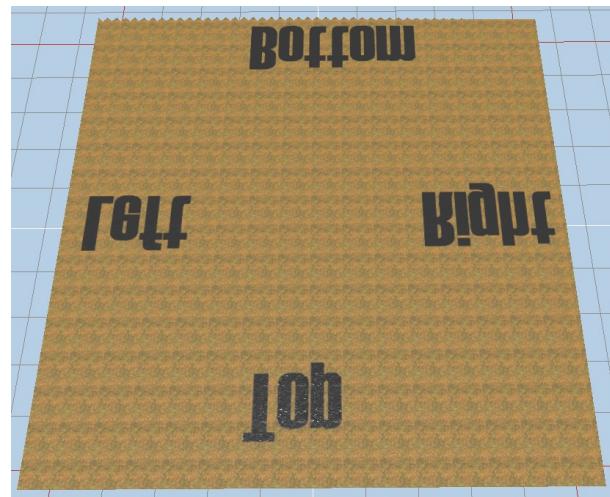
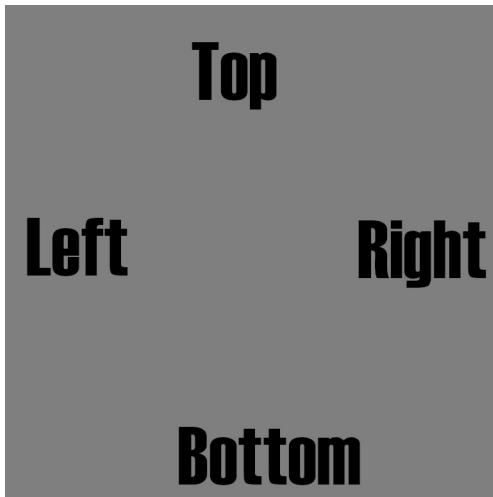
Appendix C: Hand Edit Bitmap Files

The Editor tracks the shape and attributes of the terrain using bitmaps. Using the SnowRunner Editor to paint directly on the terrain while all other terrain features are visible is very convenient, but sometimes you might want the features of an external application. This external application can be as simple as a free image editor (I use Gimp) or as sophisticated as a professional terrain modeling tool, as long as it can emit bitmaps.

This section describes the format of the bitmaps in general. Specific details for the bitmaps used by each terrain feature are in following sections.

Bitmap Coordinates

By common convention, pixels within bitmaps are numbered starting with (0,0) in the upper left and increasing to the right and down. However, the SnowRunner Editor chooses to map those increasing bitmap coordinates in the same direction as increasing object coordinates. This means that the Editor's Z coordinate increases in the northerly direction while the corresponding Y coordinate progresses **down** the bitmap. I.e. the bitmap appears upside down when viewed in the SnowRunner Editor.



West to east is assigned in the intuitive direction from left to right in the bitmap.

Tip: If the upside-down orientation of a bitmap is confusing you, feel free to flip the image vertically in your image editor while working on it. Just remember to flip it back before exporting the bitmap back into SnowRunner.

Bitmap Dimensions

The Editor has a preferred size for each bitmap that depends on the associated feature type and on the number of terrain blocks in each direction of the map. If you simply read, edit, and save one of the existing bitmaps, then your bitmap editor most likely will keep the bitmap in the preferred size. But if you create a bitmap from scratch, you should manually ensure that it is created in the preferred size.

The Editor divides each terrain block into pixels and assigns a bitmap coordinate to the **corners** of these pixels. Thus, if a terrain block has 16 pixels on a side, then the bitmap for that terrain block must be 17×17 . More generically, for a map with X terrain blocks in the east-west direction and Y terrain blocks in the north-south direction, if the number of pixels per terrain block is P, then the bitmap should be $(X \times P + 1)$ by $(Y \times P + 1)$.

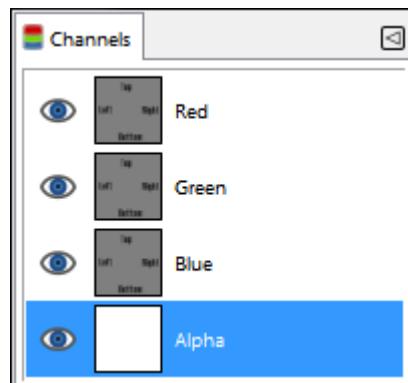
If you create a bitmap that is too large, the Editor throws away the extra pixels when building the terrain. However, it retains the dimensions specified in the file, so your map is forever burdened by a bitmap that is uselessly large.

If you create a bitmap that is too small, the Editor interpolates the values for the missing pixels when building the terrain. However, the Editor retains the bitmap dimensions specified in the file, and if you edit the bitmapped feature in the Editor, it only allows you to edit using the resolution of the bitmap file.

So do try to use the correct bitmap dimensions.

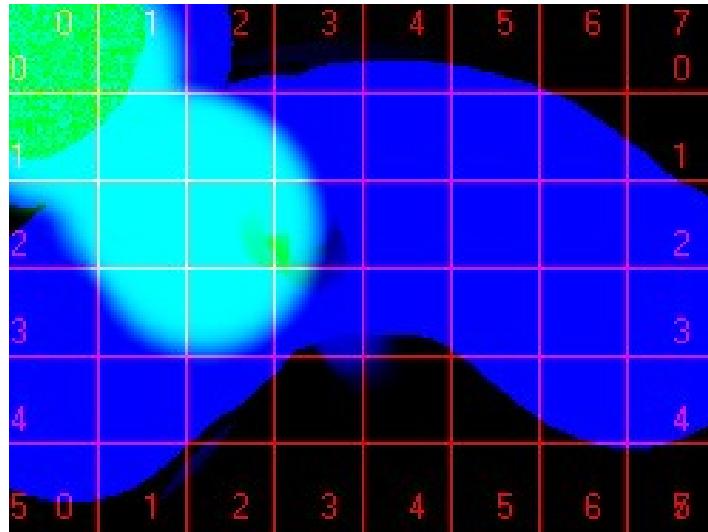
Color Channels

A bitmap typically records each pixel as 4 channels: red, green, blue, and alpha (opacity), but the Editor usually reuses each of these channels for a different purpose. For this reason, you probably want a bitmap editor that allows you to draw into only one color channel at a time while leaving the other channels unchanged. The below screenshot from Gimp shows the channel selection toolbox with just the alpha channel selected. (The eye icon for each channel means that all channels remain visible while the alpha channel is edited.)



Unused Color Channels

When a color channel is unused, the Editor sometimes initializes it with 0, sometimes with 255, and sometimes with a grid of terrain blocks and labeled at the edges with the terrain block coordinates.



When editing a bitmap externally, you may freely retain these initialized values, erase them, or use the channel for your own purposes.

Channel Depth

Each color channel is typically recorded as an 8-bit value (0 – 255). Some bitmaps, however, require a different channel depth. This is noted where each bitmap type is described.

In most cases, the Editor crashes if it reads a bitmap of an unexpected channel depth.

Use an Edited Bitmap

When you are done editing a bitmap, save (or export) it back to the game directory in place of the SnowRunner Editor's file. Then rebuild the terrain (page 65) to make your changes visible.

Use a Different File Type

Although the SnowRunner Editor expects most bitmap file to have a `.tga` extension, it uses a bitmap library that can read a variety of formats. For example, I have verified that it can read bitmaps in PNG, DDS, or even JPG format. If producing a TGA file is inconvenient, you can instead try creating your bitmap in some other common format, then rename its extension to `.tga` and see if the Editor can read it. (Warning: if the Editor fails to read a critical TGA file, it may hang or crash.)

Colorization & Wetness: `_base_tints.tga`

The [\(Colorization\)](#) (page 115) and [\(Wetness\)](#) (page 217) features are recorded in `prebuild/map_name/_base_tints.tga`.

This TGA file expects $P = 48$ pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R, G, B: [\(Colorization\)](#) color.
- Alpha: [\(Wetness\)](#) value.

The R/G/B color in the bitmap is a **modifier** on the tint of the terrain. To leave the terrain color unmodified, each channel value should be 127 or 128 (the middle of the 8-bit range). A lower value in any channel results in a darker tint with less of that color, and a higher value results in a brighter tint with more of that color.

Bug: The `_base_tints.tga` file is initialized with 127 in the R, G, and B channels, but if the [\(Colorization\)](#) brush is used to erase color, the Editor sets the R, G, and B channels to 128. The two values are visually indistinguishable, but c'mon, how sloppy is that?

The alpha channel ranges from 0 (no wetness) to 255 (maximum wetness).

`_merge_mud_wetness.tga`

Bug: When you edit mud on a map with a non-zero [Extrudes to Wetness](#) value (page 108), the Editor creates a temporary file in `prebuild/map_name/_merge_mud_wetness.tga`, and it doesn't clean up after itself.

`_merge_mud_wetness.tga` is the same format as `_base_tints.tga`, but uses only the Alpha channel to record the mud's contribution to wetness. The Editor only reads this file immediately after writing it, so there is no point in editing it externally.

Snow Depth: `_snow.tga`

The `(Snow)` feature (page 171) is recorded in `prebuild/map_name/_snow.tga`.

This TGA file expects P = 48 pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R: snow depth, as a fraction of the maximum of 3.2 meters.
- G: unused; initialized to a grid of terrain blocks.
- B: unused; initialized to a grid of terrain blocks.
- Alpha: unused; initialized to 0 (transparent).

Water Speed and Foam: `_water.tga`

The speed and foam modes of the ([Water](#)) feature (pages 208 and 210) are recorded in `prebuild/map_name/_water.tga`.

This TGA file expects P = 31 pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R: unused; initialized to a grid of terrain blocks.
- G: amount-opacity of foam.
- B: water speed, as a fraction of the maximum water speed of ~5 meters/second.
- Alpha: unused; initialized to 0 (transparent).

Water Flow Direction: _water_flow.tga

The flow mode of the (Water) feature (page 212) is recorded in `prebuild/map_name/_water_flow.tga`.

This TGA file expects $P = 31$ pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

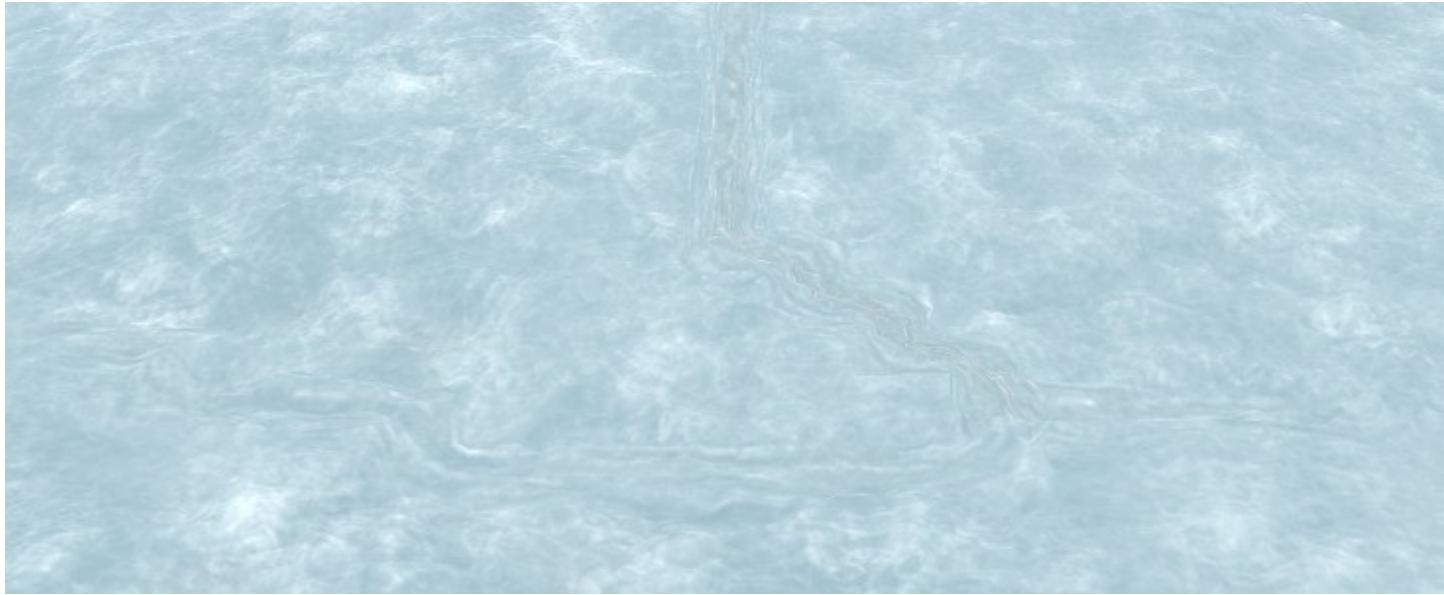
The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R: the west-east component of flow direction.
- G: the south-north component of flow direction.
- B: unused; initialized to 0.
- Alpha: unused; initialized to 255 (opaque).

The center value for R and G is 127.5. Values less than that indicate a west ($-X$) or south ($-Z$) component. Values greater than that indicate an east ($+X$) or north ($+Z$) component. The greater the deviation, the stronger the component.

Because the speed is recorded separately, the flow direction vector does not include a speed component, and the values are effectively normalized to a unit vector when used by the display engine. E.g. (128,128) effectively encodes the exact same northeast flow direction as (255,255).

Where water converges or diverges in different directions, the display engine draws a smooth transition up to about 15° of difference. Beyond that, the animation has a noticeable spot of differently animated water between the two regions. The discontinuity is much easier to see in animation than in a static screenshot.

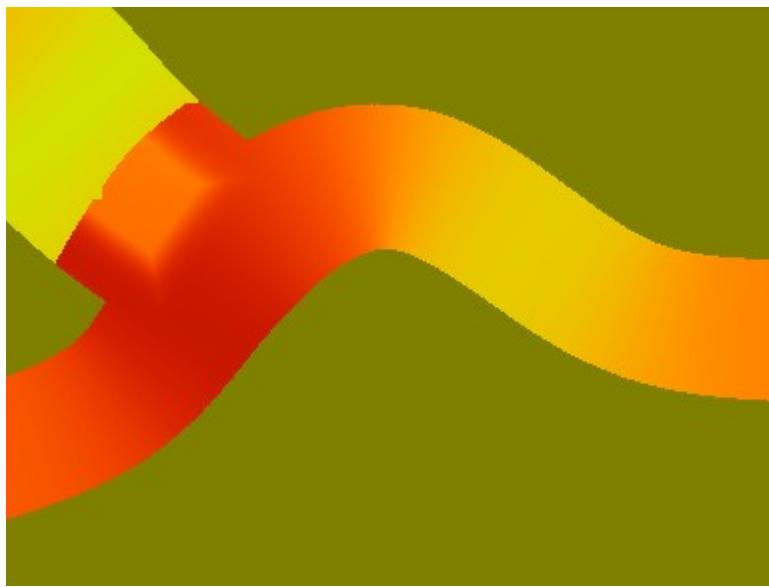


The R and G channels are initialized to 0. As a special case, the default river flow direction (set by its path of nodes) is used where the R and G channels are both 0.

Default Water Flow Direction: default_water_flow.tga

Select [Create Default Flowmap](#) from the context menu on any [\(Geometry\)](#) feature to create [prebuild/map_name/default_water_flow.tga](#) with normalized flow vectors for every point on a river showing the default river flow direction at that point.

Every point that is not in a river is given an (R, G) value of (128, 128).



Bug: The default flow map does not accurately reflect the default smoothly averaged flow directions around river junctions, but instead creates a mess of sharp direction boundaries and non-intuitive flow directions.

The [default_water_flow.tga](#) file can be copied to [_water_flow.tga](#) to serve as a starting point for modifying the river flow directions, or selective portions can be copied to restore the default directions where desired. However, it may be more effective to erase (paint to black) portions of [_water_flow.tga](#) to restore the default flow directions.

Water Color: _water_mud.tga

The **(WaterMud)** feature (page 213) is recorded in [prebuild/map_name/_water_mud.tga](#).

This TGA file expects P = 3 pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R: unused; initialized to 0.
- G: the amount of “mud” in the water.
- B: unused; initialized to 0.
- Alpha: unused; initialized to 255 (opaque).

Material Opacity: mtrl_layout.tga

The opacity of all materials (page 162) is recorded in [prebuild/map_name/mtrl_layout.tga](#).

[mtrl_layout.tga](#) records which material is used in each terrain block, and [mtrl_base.blends.tga](#) (in the next section) records which material layers within that material are used at each pixel.

[mtrl_layout.tga](#) expects 1 pixel per terrain block. Unlike other bitmaps that record a value at each pixel corner, the material opacity is recorded at the center of each terrain block, so the dimensions of the bitmap are expected to be exactly the number of terrain blocks in each direction.

The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R: a non-zero value maps to the first material.
- G: a non-zero value maps to the second material.
- B: a non-zero value maps to the third material.
- Alpha: a non-zero value maps to the fourth material.

If multiple channels are non-zero, an earlier material has priority over a later material. If all channels are zero, the Editor defaults to the first material.

When painting material opacity in the Editor, the tint that the Editor applies to the terrain is taken directly from the R/G/B channels, so low values and/or conflicting values in the bitmap could be deceptive. The Editor ignores the alpha channel when tinting.

Tip: I recommend that you use only bright red, bright green, and bright blue in the bitmap to avoid confusion.

Bug: When you paint opacity for the fourth material, the Editor fails to put a non-zero value in the alpha channel. Instead, it sets all channels to zero.

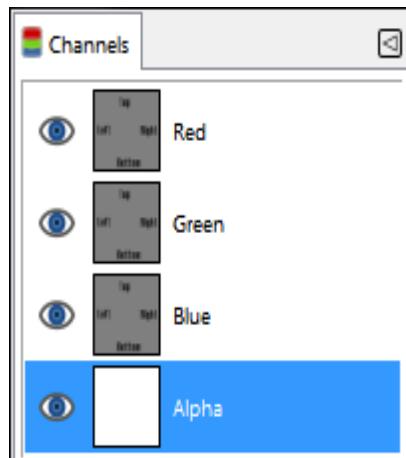
Tip: This bug can be fixed in an image editor by assigning a non-zero channel to the alpha channel for all pixels (e.g. as in the section below). Since the fourth material has lowest priority, the Editor uses the fourth material as we desire only when no other material is used.

Fix the Material Opacity Alpha Channel in Gimp

It can be difficult to figure out how to use an image editor to edit the alpha channel while leaving the other channels alone. Here is the procedure that I use in Gimp.

- Open [mtrl_layout.tga](#) in Gimp. Because the alpha channel is zero everywhere, the image is transparent.

- Select **Windows → Dockable Dialogs → Channels** to open or foreground the **Channels** tools.
- Click each of the **Red**, **Green**, and **Blue** channels so that only the **Alpha** channel is enabled for editing.



- Click the paint bucket to fill the **Alpha** channel with any color. The entire image is now opaque.
- Select **File->Overwrite mtrl_layout.tga**.

In the SnowRunner Editor, rebuild the terrain. Since the alpha channel is now non-zero, the fourth material is now correctly used where the R, G, and B channels are all zero.

Material Layer: mtrl_base.blends.tga

The painted areas for material layers in all materials (page 158) are recorded in [prebuild/map_name/mtrl_base.blends.tga](#).

[mtrl_layout.tga](#) (above) records which material is used in each terrain block, and [mtrl_base.blends.tga](#) records which material layers within that material are used at each pixel.

[mtrl_base.blends.tga](#) file expects $P = 48$ pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R: the strength of the [Layer 2](#) material.
- G: the strength of the [Layer 3](#) material.
- B: the strength of the [Layer 4](#) material.
- Alpha: the strength of the [Layer 5](#) material.

The bottom layer, [Layer 1](#), is implicitly full strength and is used if it is not entirely covered by above layers.

Accurately Paint Snow Variations with the Aid of Gimp

Because the [soft_snow](#) and [crust_snow](#) layers (page 170) are invisible in the Editor, they are difficult to paint accurately and nearly impossible to edit after painting. One alternative is to paint them in a different format that is visible in the Editor (such as colorization), then copy the painted channels into the corresponding channels of the [mtrl_base.blends.tga](#) file.

The procedure described here is rather cumbersome for quick experimentation. Consider directly (and blindly) painting [soft_snow](#) and [crust_snow](#) layers in a few small areas to find your preferred brush size and falloff value before going through the steps below.

This section describes how to perform the necessary manipulations in Gimp. I assume that you're starting with a complete map except for the [soft_snow](#) and [crust_snow](#) layers.

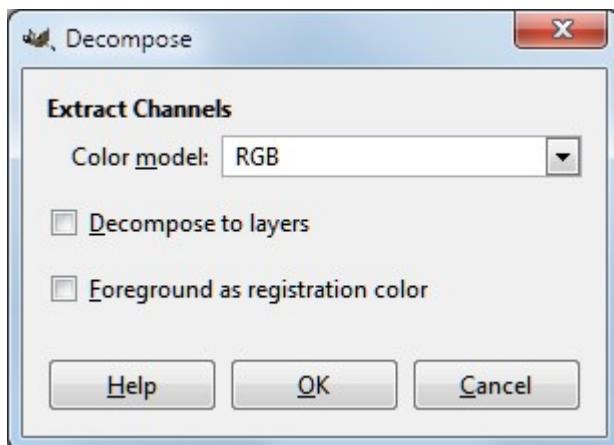
Colorize the Areas of Soft Snow and Crust Snow

- Save a copy of [_base_tints.tga](#) as [orig_base_tints.tga](#). You'll restore your original colorization data from this copy later. Also save a copy of [mtrl_base.blends.tga](#) as [orig_base.blends.tga](#). Hopefully you won't need it, but better safe than sorry.
- Set the colorization color to any shade of pure green, e.g. (0, 128, 0), and paint the entire map green. (Black also works, but green makes map features easier to see.) The important thing isn't the green color; the purpose here is to remove the red and blue colors.

- Set the colorization color to pure blue (0, 0, 255) and paint the snow regions where you want `soft_snow`. Don't worry about spilling outside the snowy regions; we'll restrict `soft_snow` to the `snow_layer` regions later.
- Set the colorization color to pure red (255, 0, 0) and paint the snow regions where you want `crust_snow`. Don't worry about spilling outside the snowy regions; `crust_snow` is ignored where the `snow_layer` isn't painted.
- Exit the colorization tool to commit your changes to the bitmap.
- Save a copy of `_base_tints.tga` as `snow_base_tints.tga`. You'll need this again if you want to make edits to your snow variations.

Separate the Color Channels

- Open `_base_tints.tga` and `mtrl_base.blends.tga` in Gimp.
- In `_base_tints.tga`, select `Colors → Components → Decompose...`
 - In the resulting dialog:
 - Set the `Color model` to `RGB`.
 - Uncheck `Decompose to layers`.
 - Uncheck `Foreground as registration color`.
 - Click `OK`.
 - This creates three new images, each containing just one of the original color channels.

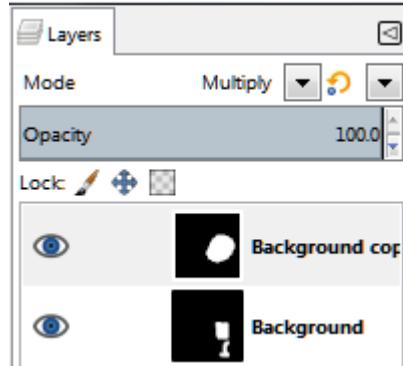


- Do the same for `mtrl_base.blends.tga`.

Mask the Soft Snow

- Select the `mtrl_base.blends-blue.tga` image. This is the `snow_layer`.
- Select `Edit → Copy`.
- Select the `_base_tints-blue.tga` image. This will eventually be the `soft_snow` layer.

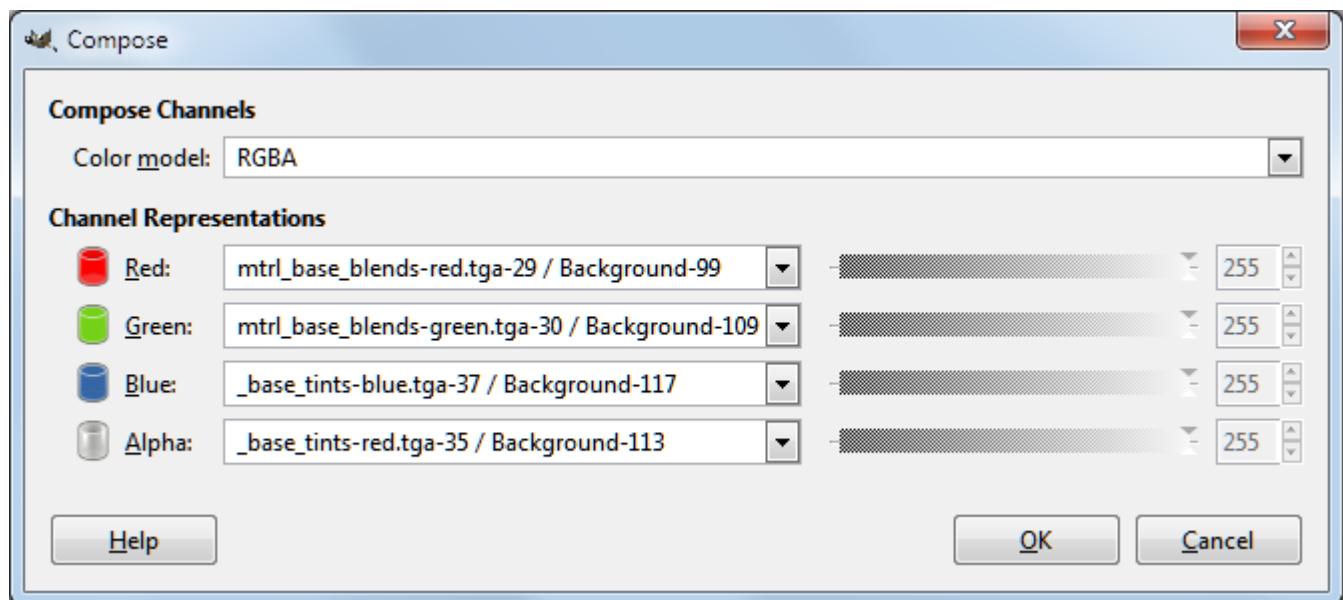
- Select **Edit → Paste As → New Layer**,
- Select **Windows → Dockable Dialogs → Layers** to open or foreground the **Layers** tools.
- At the top of the Layers toolbox, change **Mode** to **Multiply**.



- Select **Image → Flatten Image** to merge the two layers, leaving only the portions where both are painted.

Recombine the Color Channels

- In any of the decomposed images, select **Colors → Components → Compose...**
- In the resulting dialog:
 - Set the **Color model** to **RGBA**.
 - Set the **Red** source to **mtrl_base.blends-red...**; this is the original material **Layer 2**.
 - Set the **Green** source to **mtrl_base.blends-green...**; this is the original material **Layer 3**.
 - Set the **Blue** source to **_base_tints-blue...**; this is the new material **Layer 4 (soft_snow)**.
 - Set the **Alpha** source to **_base_tints-red...**; this is the new material **Layer 5 (crust_snow)**.
 - Click **OK**.
- The new image is automatically selected.



Export the Image

- Select **File → Export As...** and save the new image as **mtrl_base.blends.tga**.
- Delete **_base_tints.tga** and restore the **orig_base_tints.tga** copy you made earlier.
- In the SnowRunner Editor, rebuild the terrain to restore your original colorization.

Distribution Density & Scale: dstr_*.tga

A distribution (page 148) is recorded in [prebuild/map_name/dstr_name.tga](#).

This TGA file expects P = 48 pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap requires 8 bits per channel, with the channels assigned as follows:

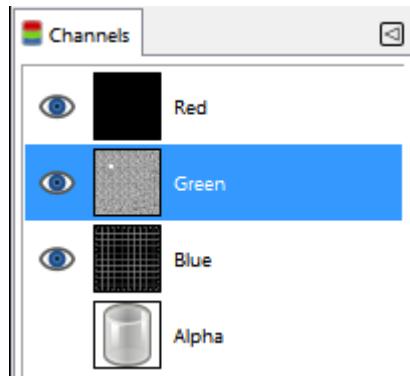
- R: distribution density.
- G: distribution scale.
- B: unused; initialized to a grid of terrain blocks.
- Alpha: unused; initialized to 0 (transparent).

Fix Distribution Scale Randomness in Gimp

Bug: The Editor's **Randomize** option for the **Scale** brush cannot restore a proper random distribution, so you need to use an external image editor to do so.

Here is the procedure that I use in Gimp.

- Open [dstr_name.tga](#) in Gimp. Because the alpha channel is initially zero everywhere, the image is transparent.
- Select [Windows → Dockable Dialogs → Channels](#) to open or foreground the **Channels** tools.
- Click each of the **Red**, **Blue**, and **Alpha** channels so that only the **Green** channel is enabled for editing.
- Click the eyeball next to the Alpha channel so that transparency is not displayed.



- If desired, use a selection tool to select the desired region to randomize. Otherwise, the entire bitmap is randomized.
- Select [Filters → Noise → Hurl...](#)
 - Change the **Randomization (%)** value to 100.0 and click **OK**.

- Select **File->Overwrite dstr_name.tga**.

To see the changes in the SnowRunner Editor, rebuild the terrain.

Terrain Height: _height.dds

The terrain height associated with the [\(Geometry\)](#) feature (page 93) is recorded in [prebuild/map_name/_height.dds](#).

This R16F DDS (Direct Draw) file expects P = 48 pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap prefers 16 bits in a single channel:

- R: height value, as a fraction of the maximum terrain height.

Accurately painting height values is very difficult because imperceptible differences in color translate to large differences in height. When I edit the height bitmap externally, I tend to either make large-scale edits that I'll adjust later in the Editor, or I make fine-scale edits using masks and filters to keep changes small and constrained.

Convert the DDS File to 16-bit PNG

DDS isn't natively supported by many image tools, and R16F isn't well supported even by some DDS-aware tools (such as Gimp), so your first step in editing the height bitmap is to convert it to a more convenient format.

A free and easy online converter I've found that handles 16-bit channels is here:

<https://online-converting.com/image/convert2png/>

When using this converter, make sure to set the [Color](#) to [48 \(True color, RGB\)](#) to get 16-bit channels in the output. The result is a PNG file with 16-bit channels that can be loaded into a 16-bit-per-channel bitmap editor such as Gimp (version 2.9 or above).

Depending on which format conversion tool you used, you may end up with a grayscale image. However, only the red channel is used by the SnowRunner Editor.

Cautions for 8-bit Channels

It may be tempting to use a more standard 8-bit PNG rather than a 16-bit PNG. However, this reduces the height precision by a factor of 16! For a map with a maximum height of 64 meters, each 8-bit step corresponds to a height difference of 25 cm (10 inches). It looks fine for perfectly flat ground, and it looks fine for impossibly steep terrain, but a smooth slope gets converted into big steps. It's bad.

If the tool you're using (such as a terrain-building tool) can only create a bitmap with 8-bit channels, the SnowRunner Editor automatically converts it to 16-bit precision. However, you'll likely need to go over the result with some fine sandpaper (the [Smooth](#) brush).

Use an Edited Height Map

Make sure to select 16-bit channels when exporting your edited image. If you are using Gimp 2.9+ and read a file with 16-bit channels to begin with, it automatically exports with the same channel depth.

Most likely you'll end up exporting a file with 3 or 4 channels (R, G, B, and alpha), but SnowRunner uses only the red channel.

You can copy the exported file into `_height.dds` and rely on the Editor's implicit format conversion, but for the height bitmap, the Editor provides a more straightforward method...

Export your height bitmap to `_height_source.png`, then select **Copy Source Heightmap** from the **(Geometry)** context menu. Note that while **(Geometry)** is selected, you can't open the context menu in the main panel because the right mouse button paints instead. So right click on **(Geometry)** in the **Scene View** to open the context menu.

Bug: The **Copy Source Heightmap** item is directly under the mouse whenever you open the context menu on any **(Geometry)** feature. If you accidentally select this context menu item, it will destroy all height data on your map with no confirmation dialog or ability to revert the change.

Tip: Rename or delete `_height_source.png` after importing it to prevent an accidental selection of **Copy Source Heightmap** from reverting later changes.

For some reason, the Editor creates a sample `_height_source.png` file when you first create your map. However, any edits that you make to terrain height in the Editor are not reflected in this file, so it can't serve as a starting point for external editing.

Tip: Delete `prebuild/map_name/_height_source.png` immediately after creating a new map to prevent **Copy Source Heightmap** from doing anything. The default contents of this file are not useful for any purpose, anyway.

Bug: Despite having a `.png` file extension, the initial `_height_source.png` file is actually in R16F DDS format, just like `_height.dds`.

River Height: _water_height.png

Select [Create Water Heightmap](#) from the context menu on any [\(Geometry\)](#) feature to create `prebuild/map_name/_water_height.png` with the height for every point on a river. Note that this is the absolute height of the river at each point, not its relative height above the terrain (the water depth).

This file has no particular use within the Editor or when hand editing, but Saber claims that some third-party tools can make use of it when creating water flow vectors.

This PNG file is created with $P = 31$ pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap uses 16 bits per channel, but the R, G, and B channels are identical to form a grayscale image:

- R: height value, as a fraction of the maximum terrain height.
- G: height value, as a fraction of the maximum terrain height.
- B: height value, as a fraction of the maximum terrain height.
- Alpha: 65535 (opaque) where a river is present, 0 (transparent) where a river is not present.

The water height values are limited to the same minimum and maximum heights as [_height.dds](#). If the river height is less than 0 at any point, that point is emitted with alpha value 0 (transparent).

Bug: If the river height is greater than the maximum terrain height at any point, that point is emitted as its height modulo the maximum height. I.e. the emitted river height starts counting up again from 0 after exceeding the maximum terrain height.

Bug: The water height is not sampled at every pixel. Instead, it is sampled at regular intervals between 1 and 2 pixels wide, and each sampled value is replicated (not interpolated!) to fill in the gaps between samples.

Mud

The **(Mud)** feature (page 105) is recorded in [`prebuild/map_name/mud`](#).

This file appears to include multiple bitmaps in DDS format. These are most likely a sparse representation of the mud so that space is not wasted on non-muddy areas.

Unfortunately, the method by which the DDS bitmaps are combined into one file is unknown, so mud cannot be edited outside the Editor at this time.

QuickMud: `_quickmud.tga`

The **(QuickMud)** feature (page 112) is recorded in `prebuild/map_name/_quickmud.tga`.

This TGA file expects P = 7 pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R: quickmud density
- G: unused; initialized to 0.
- B: unused; initialized to 0.
- Alpha: unused; initialized to 255 (opaque).

Reference Merge Map: _ref_merge.tga

The **(RefMergeMask)** feature (page 237) is recorded in [prebuild/map_name/_ref_merge.tga](#).

This TGA file expects P = 48 pixels per terrain block. (See page 438 for how to calculate bitmap dimensions.)

The bitmap requires 8 bits per channel, with the channels assigned as follows:

- R: unused; painted by the Editor the same as the G channel.
- G: the opacity of the mask.
- B: unused; initialized to 0.
- Alpha: unused; initialized to 0 (transparent).

Although the Editor paints the reference merge map in yellow (both red and green), it only uses the green channel when merging a reference.

Change Log

1.0

- The initial release.
- Includes new trucks released with season 5, but not yet other changes from season 5.

1.1

- Made a proofreading pass through the whole book and made minor improvements throughout.
- Corrected the description of the map border height (page 71).
- Added a note that DLC skins require DLC ownership (page 84).
- Fixed the organization of the Mud section and corrected a few misunderstandings (page 105).
- Added a bug warning to Colorization regarding the extra softness it may give to mud (page 117).
- Added bug warnings to trucks, models, and plants regarding the axis swaps of the Rotation properties (page 423, etc.)
- Corrected the recommended format of One-Shot Sound Domains to be mono (page 231).
- Added a bug warning regarding reference maps that extend off the edge of your map (page 236).
- Reorganized how groups are described to avoid inconvenient repetition when introducing new features (page 247).
- Added a section to describe the behavior of manual cargo generation conflicts (page 307).
- Added a bug warning for manual vs. automatic loading of logs in limited quantity (page 308).
- Reworked all mentions of reserved trucks to state that the truck should be locked to avoid bugs (page 353).
- Added text stating that a reserved vehicle can be discovered when awarded if its type is locked (page 353).
- Added a bug warning that a reserved truck or cargo trailer should not be awarded by a contest (page 353).
- Added warnings to the Publish section that DLC trucks should be mentioned prominently (page 382).
- Added CargoBigDrill that was missing from Appendix A (page 397).
- Added new sections in Appendix B:
 - Map XML Files (page 427).
 - Map and Region JSON Files (page 429).
- Added Change Log (page 460).
- Changes for the season 5 update that weren't included in 1.0:
 - Added new contract employers (page 401):
 - Tatra
 - Rostov Administration
 - Western Rail Roads
 - Added new cargo types (page 396):
 - CargoBoiler
 - CargoIronRoad
 - Added new multi-stage models (page 410):
 - pond_objective
 - rail_blocker_objective_rus_01
 - ruins_objective
 - rus_bridge_road_*_objective
 - rus_broken_pumptower_02_objective

- rus_pier_02_objective
- tatra_*_objective
- Added new truck add-ons (page 388):
 - Tow package.
- New game and Editor behavior:
 - Changed the description of cargo models since the Editor now removes them from the model asset list (page 125).
 - Removed the bug warning regarding a crafting zone not appearing in the driving view (page 308).
 - Removed the bug warning regarding a crafting zone requiring an energy zone (page 308).
 - Updated the watchtower's visibility: the watchtower zone remains in the driving view after a visit (page 364).