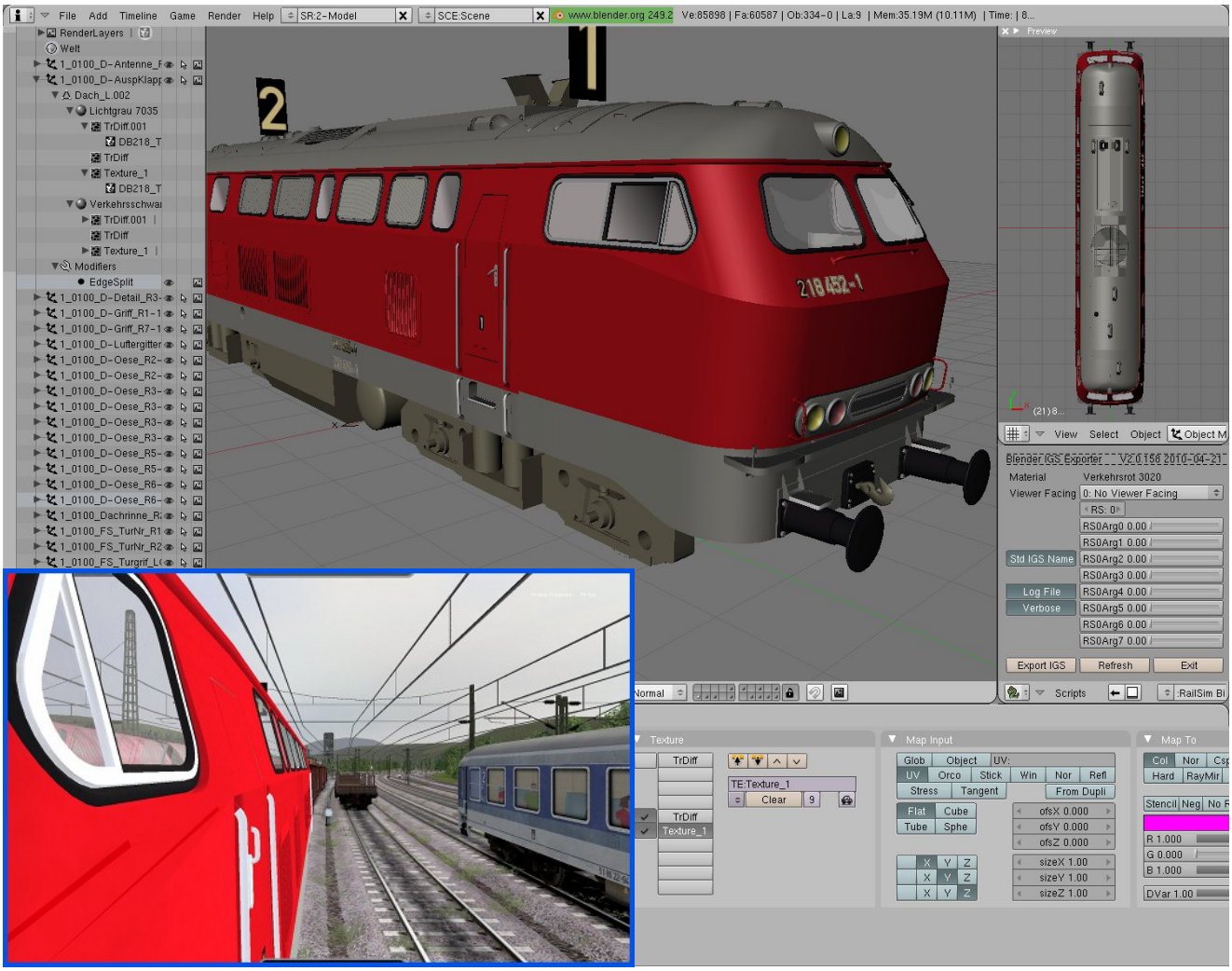


Blender IA/IGS Exporter (*Bigex*)

User Manual

HenningBR218

Version 2.0.158

1 Contents

2 Table of pictures

3 Summary

The Blender IA/IGS Exporter 2.0, (*Bigex*), is a Python Plugin for the very powerful 3D modelling and animation suite Blender. *Bigex* allows the export into Kuju's own IA and IGS intermediate formats. These file formats are required to import 3D models into Rail Simulator and Rail Works.

Note: The Blender IA/IGS Exporter version 2.0 does not support the export of animations into the IA format. This function will be implemented in a later version.

4 Introduction

The first Blender IA/IGS exporter by *steampsi* (see [1]) has got on in years. Like commonly known, it has some weak points and a few functions are missing completely. More importantly it no longer runs under the latest version of Blender.

Therefore, it was about time to write a new IA/IGS export plugin for Blender. Its name is Blender IA/IGS Exporter 2.0, or *Bigex* for short. It is completely new and developed from scratch. Like many other Blender plugins, *Bigex* is written in the programming language Python.

5 Copyright, Licence, Warranty, Contact, Credits

Many thanks for the help on manual translation, review and discussion to AndiS and Jeff Douglas (JADsHome).

6 About *Bigex*

6.1 Pupose of usage

The Blender IA/IGS export extension *Bigex* exports 3D objects created using Blender into the IA and IGS file formats. These file formats are required for importing 3D modells and animations into the simulation game software Rail Simulator (see [4]) or Railworks (see [3]).

6.2 General

At export time, as many parameters as possible are exported from Blender into the IA/IGS files. All parameters not in Blender but required by the IA/IGS format, are set to meaningful default values.

Note: *Bigex* does not check if all data to be exported are set to be Rail Simulator compliant (e.g. object names, shader names, etc.). Setting all parameter to get an IGS file which successfully exports to the Rail Simulator is in the responsibility of the user.

Note: All statements in this manual in relation to Rail Simulator are valid for Railworks too.

6.3 Feature overview

1. Runs under Linux and Windows versions of Blender
2. Export into IA (not in version 2.0) and IGS formats
3. Export of object hierarchies
4. Graphical input fields for IA/IGS specific parameter
5. Arbitrary lengths of names for objects, shaders, and textures
6. Automated manipulation of arbitrary IA/IGS data fields on export including support for regular expressions
7. Free choice to export all visible objects or all selected, visible objects
8. Multiple UV coordinate layers
9. Multiple texture layers
10. Free assignment of UV coordinate layer to texture layer
11. Simple system for parallel usage of the old export plugin
12. Support for Blender edge marker (SHARP, SEAM)
13. Support for Blender modifier (EDGE SPLIT)

7 Installation

7.1 Supported program versions

Bigex runs under the following Blender / Python combinations:

1. Blender 2.48 / Python 2.5
2. Blender 2.49 / Python 2.6

Once a full featured Blender 2.5 version is available, all future versions of *Bigex* will run under Blender 2.5 / Python 3.0.

Both programs are available for free in Linux and Windows versions.

Blender: see [2].

Python: see [5].

Note: *Bigex* version 2.0 does not run under Blender 2.5 / Python 3.0, because a full featured Blender 2.5 version is not available yet.

Together with an image processing program, like the free GIMP (see [6]), you have all the tools required to create 3D objects for Rail Simulator (see [4]) and Rail Works (see [3]). This devolopment environment formed by these tools runs under Linux as well as under Windows.

7.2 Required programs

To use *Bigex* the following programmes must be installed correctly:

- Blender, Version 2.48 or 2.49
- Python, Version 2.5 or 2.6
  (depending on the Blender version, see section 7.1 for further guidance)

Note: *Bigex* was not tested for Blender versions lower than 2.48. There is a chance that *Bigex* will run on such versions, as long as they use Python 2.5.

7.3 Installing the Bigex scripts

The installation of the *Bigex* scripts is very simple. Just copy all files from the ***Scripts*** folder of the package into the Blender scripts folder.

Under Linux, this is: <home>/.blender/scripts

e.g.: ~/.blender/scripts

Under Windows: <programme drive letter>\:<program folder>\<Blender folder>\.blender\scripts

e.g.: C\:Programme\Blender\.blender\scripts

After the next start of Blender the *Bigex* export script can be used.

7.4 Blender example files

The installation package includes example files. Blender file ***Bigex_ExampleObject_1.blend*** in folder ***Bigex_ExampleObjects*** contains complete examples for the usage of Rail Simulator shaders. The required texture files are in the subfolder ***Textures***.

A 3D model for the commonly used shaders show all the Blender settings needed for the export into IGS. The example objects also show how many texture layer you need for each shader, and which parts of the texture image are used for which effect in Rail Simulator.

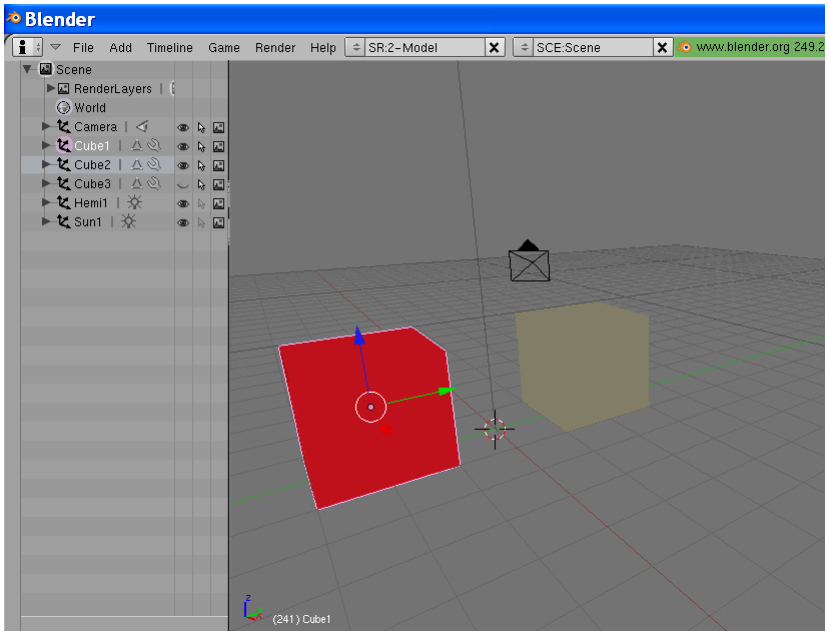
All examples are ready to be exported. To export a single example model, select it using the mouse. If no objects are selected, all objects will be exported.

There are two more sample files in folder ***Bigex_ExampleObjects***. These contain examples for the search and replace expressions for the IGS data manipulation at export time. More on this later in section *11 Data modification during export*.

8 Usage

8.1 Exportable objects

As a general rule, only ***visible*** objects of type ***Mesh*** or ***Empty*** are exported. Objects, which are set on invisible, will not be exported. This means all objects that are marked with the greyed eye in the Blender *Outliner* window will not be exported.



Picture 8.1: Exported objects

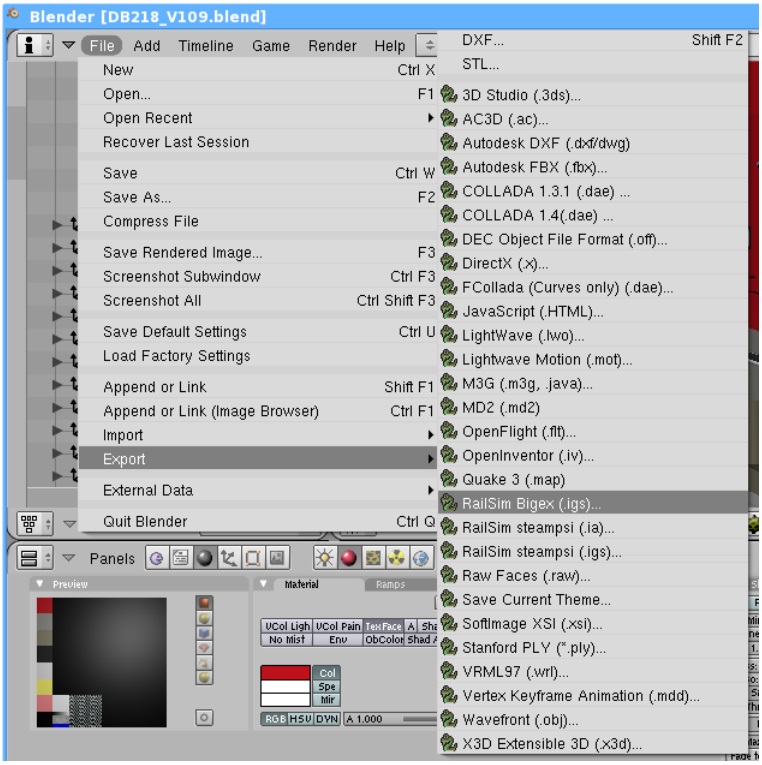
In this example the selected mesh object *Cube1* is exported only. The not selected object *Cube2*, the hidden object *Cube3*, and all lamp and camera objects are not exported.

If exportable objects are selected, only these are exported. Otherwise, all visible objects are exported.

8.2 Starting the graphical user interface (GUI)

The export of the objects is done in two steps. In the first step the *Bigex* GUI shown in picture 8.1 of section 8.3 is started. From there the button *Export* starts the real export process in the second step.

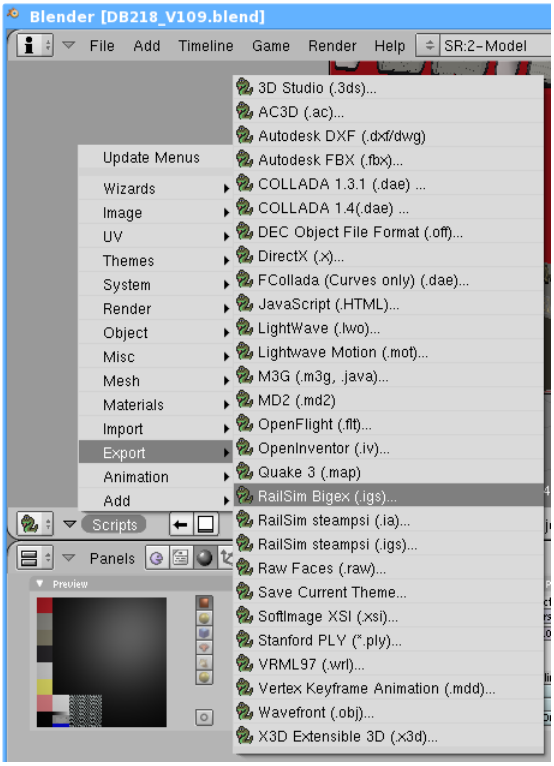
There are several ways to start the user interface. The simplest is using the Blender menus as shown on pictures 8.2 and 8.3 below. But of course, you can load the *Bigex* start script into the text window just like any other script in Blender and then press ALT-P.



Picture 8.2: Starting the user interface 1

To start *Bigex* from a Blender menu, there are several alternatives.

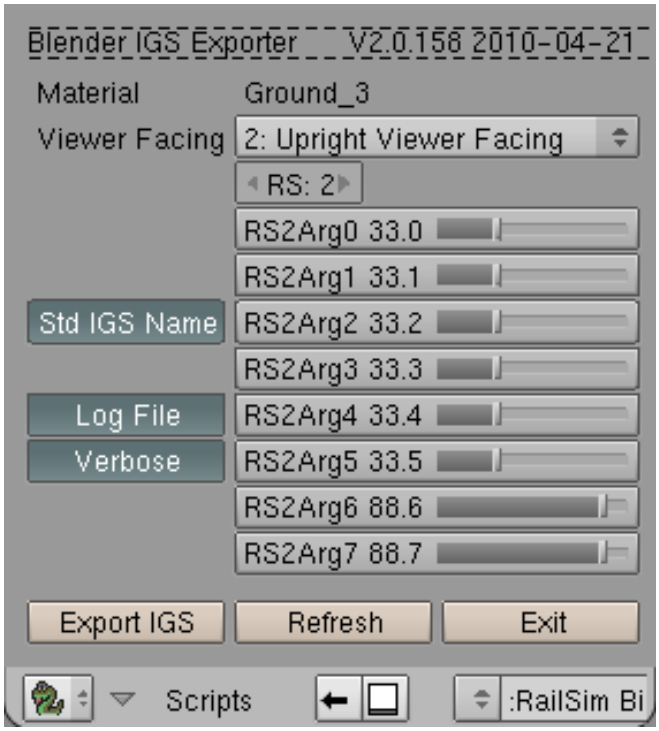
The first is via the menu
File => ***Export*** => ***RailSim*** *Bigex* ***(IGS***.



Picture 8.3: Starting the user interface 2

A second way is via the menu
Scripts => Export => RailSim *Bigex* ***(IGS.***

8.3 Graphical user interface (GUI)



Picture 8.4: User interface

The user interace of *Bigex*. On the right hand the input fields for the IA/IGS specific parameter, left the option buttons for the export and at the bottom the action buttons.

On the right side of the *Bigex* user interface are the input fields and sliders for the IA/IGS specific parameters. All those parameter belong to a certain material and will be exported into the IA/IGS format. The parameters for the current active Blender material are always automatically shown. The parameters for all Blender materials are saved within the .blend file and are thus available again next time that .blend file is loaded.

Note: Which values from Blender will be mapped on which fields of the IA/IGS format file is shown in the table in section 14.1.

The export option buttons can be found on the left side of the *Bigex* user interface. The ***Log File*** option enables output logging to the console window to be written into a log-file. The option ***Verbose*** dumps out a lot more detailed information than shown in the console window. More on log-files can be found in the following chapters.

If the option *Standard IGS file name* (***Std IGS Name)*** is not selected, then as the first step of the export a file selector window opens and asks for the file name for the IGS file to be exported. Otherwise the currently opened *.blend* file name with the extension of *.igs* is used as the name for the IGS file to be exported. In this case no file selector window appears.

Existing IGS files are replaced without asking for confirmation. To increase safety against unintended overwriting of an existing *.igs* file, an "_exp" is added to the filename.

Example: If your current Blender file is *MyEngine.blend,* your IGS file name would be *MyEngine_exp.igs*.

Action buttons are found at the bottom of the user interface. The ***Exit*** button closes the *Bigex* user interface. The ***Refresh*** button will update the current material if *Bigex* does not do this automatically. Another option is to move the mouse pointer over a few other sub-windows of Blender. This will trigger the automatic user interface update function of *Bigex*.

The ***Export IGS*** button starts the export process.

8.4 The export process

After clicking on the ***Export IGS*** button, *Bigex* starts the mapping of the objects and parameter from the Blender internal data structures into the IA/IGS data structures. Depending on the number of objects, vertices, polygons and materials as well as your PC hardware, this process can take seconds, minutes or hours.

During the export process *Bigex* works on copies of the Blender objects only. Neither objects nor parameters of the Blender data structure are modified!

The progress of the export can be monitored on the bar on the top of the Blender window, permitting a rough guide to how long the export will take.

The console window shows important warning and error messages throughout the export process. If the Log File button is selected on the *Bigex* user interface all this information, and some more are written into a log file. This is created in the same folder than the IGS file and has the same name as the *.igs* file but with the extension of *.log*.

Example: With the opened Blender file *MyLoco.blend*, the resulting *Bigex* log-file name would be *MyLoco_exp.log*.

This log file lists in great detail and in readable format what is written into the binary IGS file.

Note: For complex objects to be exported, the log file can become quite large. It can get a size of more than 100MB!
If this behaviour is not desired, the option button ***Verbose*** and, if required, additionally the button ***Log File*** need to be deselected.

Bigex does not check any parameter values for correctness in the sense of a successful and error free import into Rail Simulator.

Note: It is the responsibility of the user that object names and values are in keeping with the Rail Simulator specifications.

However, warning messages are created to hint on possible root causes of incompatibilities in the exported IGS files and with the Asset Editor.

For missing parameters, for example ambient light colour, sensible default values will be exported. If important information like the name of a texture file is missing, warnings are issued and the object will be exported partially or not at all.

Therefore if whole objects or parts of them are not exported, check the log file for warnings and error messages.

If there are no major errors in the Blender objects, an IGS file is written. Otherwise, the export process terminates prematurely with an error message.

Finally *Bigex* writes a status report and a set of statistical values to the console window and the log file. They tell you how much time each of the export steps took and how many objects, vertices, polygons and materials were written into the IGS file.

Note: Minimising the console window reduces the total time the export process takes. Printing and scrolling this window significantly adds to the total duration of the export process.
To see messages already dumped during the export, open the log file in an external editor and reload it from time to time.

In some cases the import of the IGS file into the asset editor of Rail Simulator fails. In other cases the conversion in the asset editor into the *.GeoPcDx* file terminates with an error message. To discover the cause it is a good idea to have a closer look at the warning messages in the *Bigex* log-file. If *Bigex* generates no warnings, the import into Rail Simulator should be error free.

9 Creating objects for the IA/IGS export in Blender

9.1 Object types

In basic only two types of object are exported by *Bigex*. These are Blender *Mesh* objects, containing the geometry data of a 3D model and *Empty* objects, which are used amongst others for grouping of the mesh objects. All other Blender objects like splines, cameras and lamps are not exported.

9.2 Object orientation

The coordinate systems of Blender and Rail Simulator are not the same. Rail Simulator uses a left handed coordinate system (positiv values point: X right, Y up, Z forward) and Blender is using a right handed coordinate system (positiv values point: X right, Y forward, Z up). The conversion between these is done automatically by *Bigex* at export time.

This means, if for example you build a loco in Blender in a way that the X axis (red arow) points to the right, the Y axis (green arrow) points in driving direction and the Z axis (blue arrow) points upward, then the loco has the correct orientation in the Rail Simulator.

9.3 Object hierarchies and other export limitations

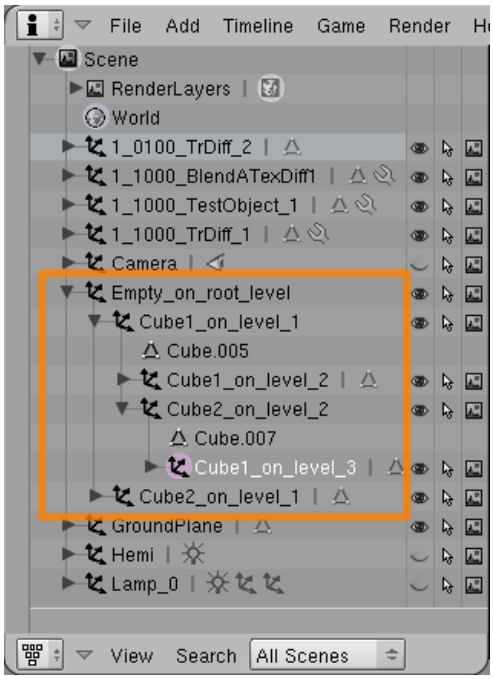
In basic no object hierarchies can be transferred from Blender into Rail Simulator. The problem which prevents this is on the Rail Simulator side. Thus the following precondition must be taken into account in Blender for a successful export into Rail Simulator.

Generally the IGS format is designed to hold information on object hierarchies. *Bigex* supports the export of object hierarchies into the IGS format.

Unfortunately the *ConvertToGEO.exe* tool, used by the asset-editor, does not support object hierarchies when converting the IGS files into the GeoPcDx format. If the IGS file contains more than one hierarchy level it terminates with the error message *"Instancing not supported"*. The import into Rail Simulator will fail.

One possible way to prevent this is to move all objects into one hierarchy level. Another is to group the objects on the root level with Blender *Empty* objects. If then only the level below the *Empty* objects is used for the mesh objects, you just need to select all objects on that level for export. By using this last method the hierarchy level is reduced back to a depth of one.

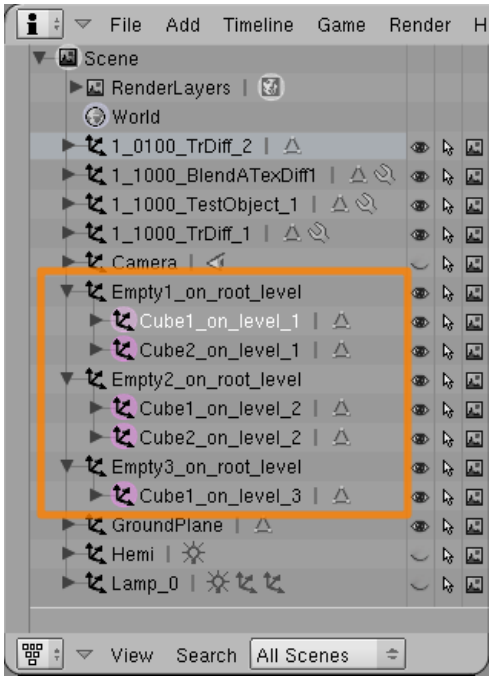
It is more simple than it sounds, once you take a closer look at both example pictures.



Picture 9.1: Object hierarchies 1

Object hierarchies in Blender with a depth of four. The import into Rail Simulator will fail if you try to export the whole hierarchy.

The example in picture 9.1 shows an object hierarchy at which the object *Cube1_on_level_3* is on the third level below the root level. The whole hierarchy structure is exported into the IGS file. But the export from the asset editor into Rail Simulator fails later with the current version of the converter tool.



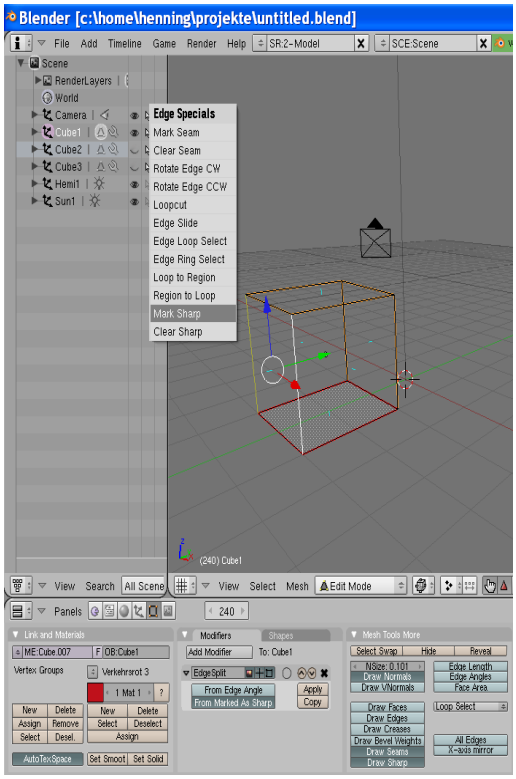
Picture 9.2: Object hierarchies 2

Object hierarchies in Blender with reduced depth. All objects from lower levels are moved on the first level. Just the selected objects are exported. This way the IGS file contains the main level only.

Another limitation with the converter tool *ConvertToGEO.exe* is the maximum count of objects. It is limited to 256. At an object count above this, the tool terminates with a corresponding error message. The only workaround for this is to combine some meshes to reduce the object count.

9.4 Edge marker

Blender offers some very useful edge markers. Marked edges are highlighted in different colours. On the *Mesh Tools More* tab in Blender you determine which types of edges are shown in colours. For the marker and modifier shown below, these are *Draw Seams* and *Draw Sharp.* Which edges will be highlighted can be adjusted in edit mode on the *Editing (F9)* menu and the *Mesh Tools More* tab - e.g. the edge marker *Draw Seams* and *Draw Sharp*.



Picture 9.3: Edge marker

In this example, in menu *Mesh* Tools *More* the face normals (little blue lines on the visible side of the face), the SEAM (orange) and SHARP (red) edge markers are selected to be visible.

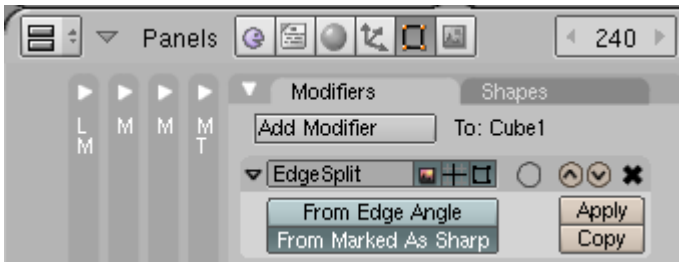
Selected edges can be marked with the *Edge Specials* pop-up menu (CTRL-E key).

As a help for flexible unwrapping, Blender provides the edge marker *SEAM, which are highlighted* in orange. This marker is supported by *Bigex*. This means that additional vertices are generated as required for the export of all UV coordinates.

Another very useful edge marker is *SHARP*, which highlights edges in red. This one is used together with the modifier *EdgeSplit* (see section 9.5) to obtain hard edges on an object that is set to smooth (*Set Smooth*) - for example at a closed half of a sphere. The *EdgeSplit* modifier should be set to *From Marked as Sharp. From Edge Angle* should be deselected in this case.

9.5 Modifier

Nearly all modifiers are supported by *Bigex*. During export, all modifiers for an object are applied to a copy of the original object. This copy is then exported and deleted afterwards. The original object remains as is with all its modifiers.



Picture 9.4: Modifier

Example: The very useful modifier *EdgeSplit*.

Refer also to section 10.1 Overview.

10 UV coordinates and Texture layer

10.1 Overview

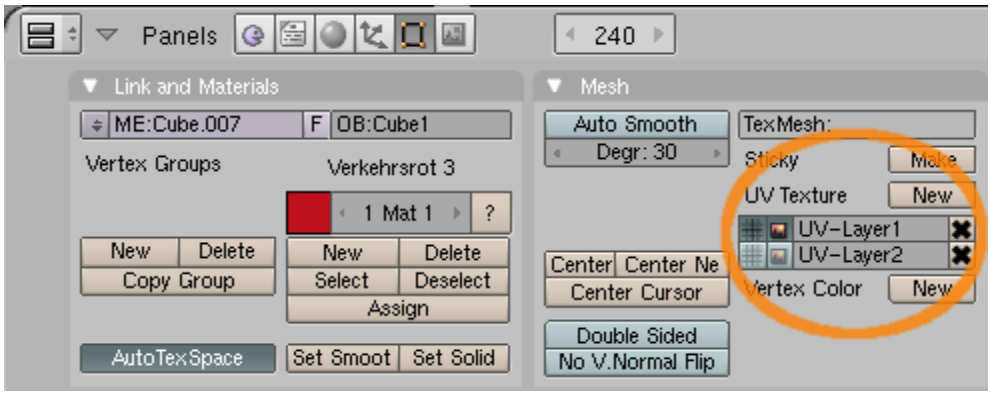
To texture a 3D model, you need ***UV coordinates***, a ***texture image*** and a ***shader name***.

The following explains these terms in brief. The following three chapters describe the details and the usage more in depth.

UV coordinates define for each vertex the position on the texture it is mapped to. Because a *texture image* has two dimensions, only one pair of coordinates are required. These are the X and Y position on the *texture image*, which is mapped to the relevant vertex. For preventing confusion with the X, Y and Z coordinates of the vertex's geometry data, the coordinates on the texture image are called U and V.

A list containing all the UV coordinates for all vertices of a mesh is called the UV (coordinate) layer. In Blender it is possible to set a name for each UV layer for easier identification. Each mesh object in Blender can have multiple UV layers each with their own unique names.

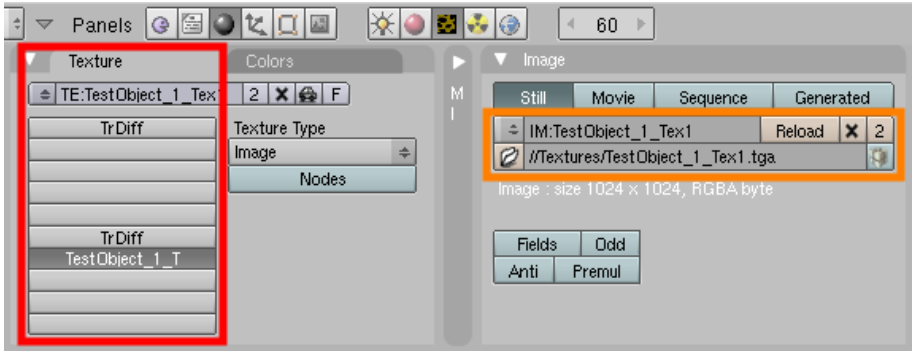
Unfortunately Blender calls this UV-layer "UV Texture". This should not be confused with the term *texture layers,* described below. In this document, we will stick with the term *“UV layer”* for a set of UV coordinates of a mesh.



Picture 10.1:
UV Texture

Two UV-Layer with a set of UV coordinates each of a Blender mesh object.

A ***texture image*** in Blender is a reference to a path and file name pointing to a texture image file. A *texture image* can be specified via a t*exture layer* for Blender materials. Each texture layer can hold one *texture image* reference. Furthermore a texture layer holds information about what *UV layer* is to be used to map the *texture image* on the mesh vertices. Of course it is possible to have multiple texture layers for a material, each with it’s own unique name.



Picture 10.2: Texture Layer

The *Textur Layer* stack of Blender in the *Texture Menue* (red border) containing three defined layer. Layer number seven *TestObject_1_T* has the textur image *TestObject_1_Tex1.tga* assigned (orange border).

Important note: Images, selected in the *UV/Image Editor* window of Blender at UV coordinate creation, are totally independent from the *texture images* of the *texture layer*! Texture images set in the *UV/Image Editor* window are exclusively used in Blenders 3D windows.
For the export into the IGS files and the later use in Rail Simulator, as well as for rendered pictures in Blender, the *texture images* specified in the *texture layers* are used.

A ***Shadername*** must be specified for Rail Simulator to display the 3D object. This means, that the shader names are specified by Rail Simulator. Depending of the chosen shader it requires one, two or three *texture layers*, each one with a *texture image* assigned. The Rail Simulator shader names are passed to Blender via one of the texture layer name (see section 10.3 Texture layer).

10.2 UV (coordinate) layer

UV layers are lists with two dimensional coordinates of points on a texture image, to which the vertices of a mesh are mapped on.

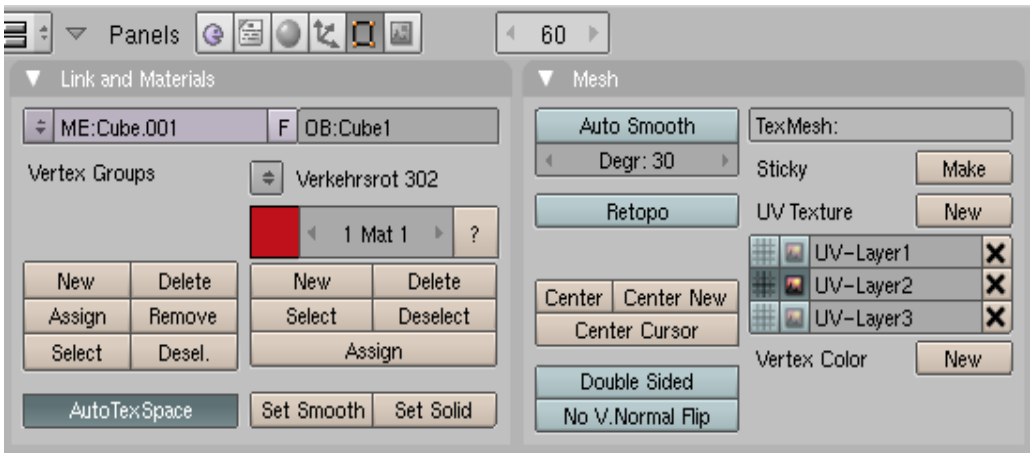
Usually one UV layer is sufficient. In exceptional cases it can be necessary to have a second or third UV layer for a mesh. This may be the case for a second texture - for example normal mapping data, which shall be used for a few faces only or lays differently on the mesh.

To create UV-coordinates in Blender, select those faces in edit mode for which you want to set the UV coordinates. Then use the “unwrap” method of Blender for these faces by pressing the U-key. This brings up the *UV Calculation* menu, which offers a list of unwrapping methods. Simply experiment with the available alternatives to see which one delivers the best results for your needs.

After this step, the UV coordinates of the selected vertices are added to the active UV layer. If there was none before, a new one is created. The new UV layer can be found, with a selected mesh object, under *Editing (F9)* => tab *Mesh,* directly underneath *UV Texture*.

Further UV layers can be created using the *New* button. Note that the UV coordinates are always added to the ***active*** UV layer.

Names for UV layers can be chosen arbitrarily. In the example below on picture 10.3, *UV-Layer* 1 and *UV-Layer3* are inactive while *UV-Layer* 2 *is* activated.



Picture 10.3:
UV layer

UV layer, in Blender to be found in the *Mesh* menu under the term *UV Texture*.

10.3 Texture layer

Texture layers, also called texture channels in Blender are used to hold all parameters for the single render stages of a Rail Simulator shader. Texture layers are part of a Blender material.

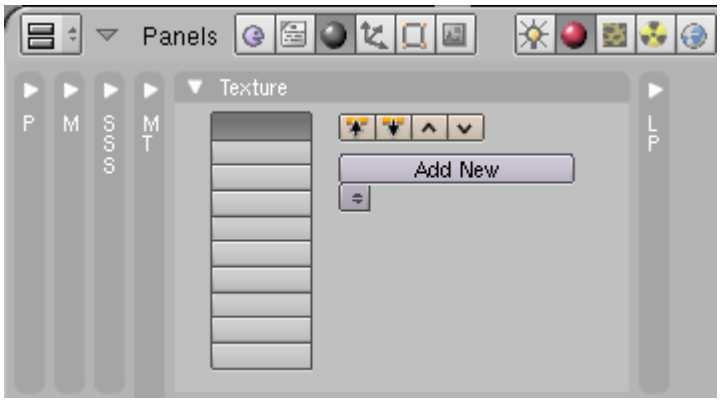
Each mesh object and each face in Blender must have a material assigned to it, otherwise it cannot be exported to IGS. The assigned material prescribes the way the object is displayed in Rail Simulator later, i.e., the visual properties of the object.

In Blender, there are many parameters to define these properties. However, the IGS format can hold a small subset of them only.

One of the most important properties are *texture layers*. These define the shader which is used by Rail simulator later, as well as the *texture images* to be used for the render stages of that shader.

For each texture layer, one file name can be specified, which points to the image for the texture. The images should be in *.tga* or *.png* format because those both formats can hold alpha channel data. The *.tga* format is preferred and should be used.

The material panel (*Shading (F5)* => *Materials Button* => tab: *Texture*) contains a stack of texture layers for each material. To set one of the texture layers, simply choose an existing one or create a new one by selecting *Add New*. Afterwards set *Texture Type* to *Image* and select your texture file with the file selector on the *Image* tab like shown on picture 10.4.



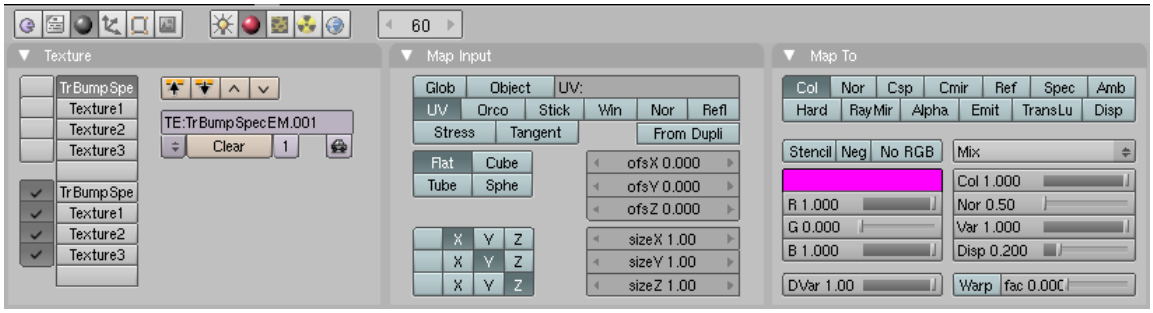
Picture 10.4: Texture Layer Stack

The empty *Texture Layer* stack in Blender, this time in the *Material* menu.

Rail Simulator requires it's own, specific shader names to display the 3D models correctly. Thus a specific Blender texture layer is used to set the shader name for the later IGS export. More on this in section 10.5 Shader

Important note: The shader name for a material which is used for the export into the IGS format is specified via the name of the **first active** texture channel.

Important note: Due the name length in Blender being limited some shader names need to be set using a short form. During the export, those are replaced by the full Rail Simulator shader names (see section 10.5).



Picture 10.5: Active Texture Layer

In the example in picture 10.5 the sixth *texture layer* of the list is the first active one. This one defines the shader name *TrBumpSpecEM* in this example. The following three active texture layers define the other texture files required by this shader.

Exclusively the name of the first active texture layer is used. No further settings are used for export into the IGS file. By this, there is a clean separation between shader name and texture layer definitions. For this reason it is required only to define one texture layer with a shader name for each Rail Simulator shader. This one can be reused as often as needed for other materials.

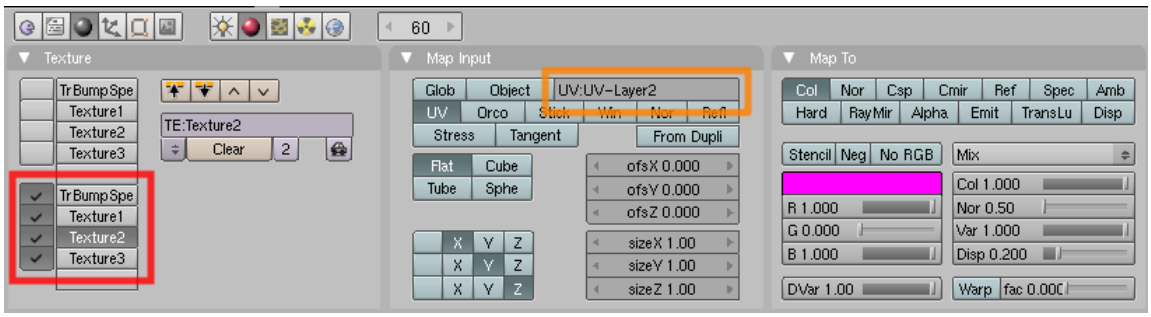
The texture layer specifying the texture images for the chosen shader are defined in one of the following entries. Depending upon how many the chosen shader expects, the corresponding number of texture layers need to be entered and activated. Due to only the texture image references and settings being used and not the names, all of that texture layer can be reused for other material too.

Important note: Empty and/or inactive texture layer entries (i.e., without a tick mark) will be ignored by *Bigex*.

This means, the next active texture layer, following the shader name texture layer, is for the first render stage of the shader, the next active one for the second, and so on. All active texture layers will be exported, independent of what number of the layer the shader expects.

The following parameters of the single texture layer are used for the export into IGS:

Tab ***Map Input***, Button ***UV***	Must always be selected.
Tab ***Map Input***, Text field ***UV:***	If there are multiple UV layers, the name of the intended UV layer must be specified here. If this field is left blank, then the UV coordinates from the active UV layer are used. Usually there is only one UV layer required and this field remains blank.



Picture 10.6: Render Stages

The example in picture 10.6 shows four activated texture layers (red border). For the second texture pass of shader *TrBumpSpecEM* it is specified that the UV coordinates from the UV layer with the name *UV-Layer2* shall be used.

Some further parameters of this texture layer are exported into the IGS file too. For a detailed list refer to the table insection 10.5 Shader.

10.4 Backward compatibility with old IGSexport plugin

If you would like to have the choice to export your 3D model with the old IA/IGS exporter plugin instead there exists an easy solution. This may be required as long as *Bigex* does not support the IA format and you may want to export animated objects.

The solution consists of two separate texture layer groups. One for the old IGS export plugin and one for *Bigex*. Both exporters expect the shader name as the name of a specific texture layer. But in Blender it is not possible to have two texture layers with the same name but different settings.

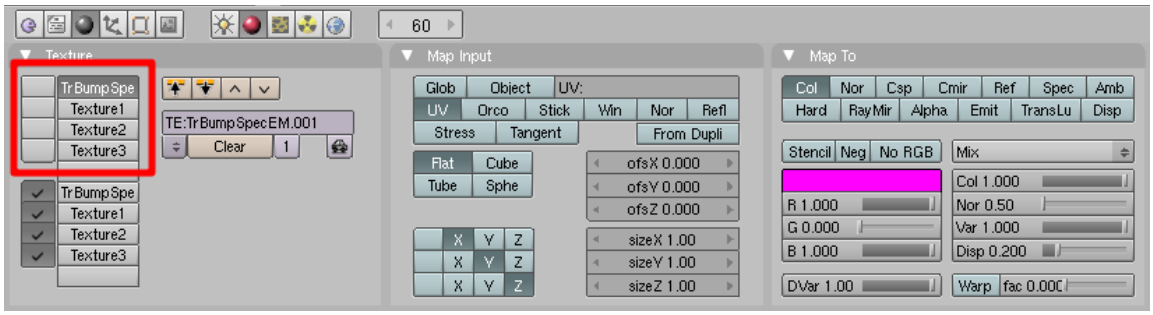
There is however an easy workaround for this problem. The old IGS export plugin for Blender (see [1]) expects the shadername in the ***very first*** entry of the texture layer list. Fortunately it accepts the short form of the shader names (see table in 10.5) for Rail Simulator, but also accepts an extension of the shader name of the form „**.001**, **.002**, …“. By this extension of the shader name / texture layer name it is possible to have a clear separation between both texture layers with the same shader names.

Directly following, starting from the second entry of the texture layer list the old IGS exporter plugin expects one or more further texture layer entries, if applicable. For all following unrequired texture layers the old exporter reports a warning message, but ignores it. In difference to *Bigex* the old exporter ignores if the texture layer is activated or not.

The texture layer group with the red border in picture 10.7 shows the expected texture layer for the old exporter.

The activated group underneath is the texture layer *Bigex* expects.

Like mentioned before, the old exporter runs under Blender 2.48 and Python 2.5 only. If you have this combination running, then you can export by choice with both exporters ***without*** having to change anything.



Picture 10.7: Settings for export with both exporter plugins.

Note: All texture layers, except those for the shader names, can be used in both groups one or more times. The old exporter uses just the texture image file name in difference to *Bigex,* which uses further settings (see section 11.4 Mapping of Blender data on the IGS format)

10.5 Shader

The table below lists almost all of Rail Simulator's shaders. The first column shows the official, full name, which need to be used for the IGS files. The second lists the short form for use in Blender. The following columns contain a short description and the usage of the texture slots.

Further information about Rail Simulator's shaders can be found in the Rail Simulator developer documentation (see [12]) or in the Rail Simulator WIKI (see [10]).

Shaders are grouped into ***Non-fx*** shaders and ***fx*** shaders.

Textures come in two forms: RGB images using 24Bit per pixel (8Bit red, 8Bit green, 8Bit blue) and RGB images with 8Bit alpha channel (RGBA) using 32Bit per pixel. RGBA contains a 24Bit colour image and an 8Bit gray scale image.

Non-fx Shader					
Shader Name (Non-fx)	**Short Name**	**Description**	**Texture 1**	**Texture 2**	**Texture 3**
AddAlphaDiff	AddAlphaDiff	No texture, additive vertex alpha with diffuse colour.	none	none	none
AddATex	AddATex	Texture mapped, no lighting applied, using additive alpha from texture's alpha channel	RGB: Color A: Transp.	none	none
AddATexAlphaDiff		Texture mapped, with diffuse colour, using additive alpha from texture's alpha channel combined with vertex alpha	?	?	?
BlendATexDiff	BlendATexDiff	Texture mapped, with diffuse colour, using additive alpha from texture's alpha channel	RGB: Color A: Transp.	none	none
AddDiffuse		No texture, with diffuse colour, using additive alpha	?	?	?
AddTex	AddTex	Texture mapped, no lighting applied, using additive alpha	RGB: Color	none	none
AddTexAlphaDiff		Texture mapped, with diffuse colour, with additive vertex alpha	?	?	?
AddTexDiff		Texture mapped, with diffuse colour, using additive alpha	?	?	?
BlendAlphaDiff		No texture, vertex alpha blending with diffuse colour	?	?	?
BlendATex		Texture mapped, no lighting applied, using alpha blending from texture's alpha channel	?	?	?
BlendATexAlphaDiff		Texture mapped, with diffuse colour, using alpha blending from texture's alpha channel combined with vertex alpha	?	?	?
BlendATexDiff		Texture mapped, with diffuse colour, using alpha blending from texture's alpha channel	?	?	?
BlendATexDiffTrans		Texture mapped, diffuse colour, alpha blending from texture's alpha channel, pixels with alpha=0 are transparent (e.g.	?	?	?

Non-fx Shader					
Shader Name (Non-fx)	**Short Name**	**Description**	**Texture 1**	**Texture 2**	**Texture 3**
		alphaed fences).			
BlendTexAlphaDiff´		Texture mapped, with diffuse colour, with vertex alpha blending	?	?	?
BridgeSplit		Not drawn. Use to define areas where track crosses over itself.	?	?	?
Diffuse		No texture, just diffuse colour	?	?	?
DualAddATexDiffDestBlend		Dual textured, diffuse colour, first pass additive, and second pass blended alpha with the alpha of the first texture (e.g. puddles).	?	?	?
DualBlendATexDiffAdd		Dual textured, with diffuse colour, using alpha blending for first pass and additive alpha for second pass	?	?	?
DualTexDiffAdd		Dual textured, with diffuse colour, using additive alpha for second texture	?	?	?
DualTexDiffAddWithLightIntens		Add second pass to first pass, brightness of second pass affected by lightmaps if used	?	?	?
DualTexDiffAddWithoutLightIntens		Add second pass to first pass, brightness of second pass not affected by lightmaps if used	?	?	?
DualTexDiffInvisibleStencilBlend		Dual textured, with diffuse colour, first pass invisible, second pass alphaed using alpha of first pass texture	?	?	?
DualTexDiffStencilAdd		Dual textured, with diffuse colour, using additive alpha for second texture only where first texture has solid alpha	?	?	?
DualTexDiffStencilBlend		Dual textured, with diffuse colour, using blended alpha for second texture only where first texture has solid alpha	?	?	?
DualTexDiffTAlpha		Dual textured, with diffuse colour, using second texture's alpha channel to blend between textures	?	?	?
DualTexDiffTrans		Dual textured, with diffuse colour, using second texture's transparency	?	?	?
DualTexDiffVAlpha		Dual textured, with diffuse colour, using vertex alpha to blend between textures	?	?	?
EmbossBumpmap		Bumpmap for Train 2 prototype or something like that	?	?	?
Invisible		Nothing is drawn - use for invisible collision barriers	?	?	?

Non-fx Shader					
Shader Name (Non-fx)	**Short Name**	**Description**	**Texture 1**	**Texture 2**	**Texture 3**
Tex	Tex	Texture mapped, no lighting applied	RGB: Color	none	none
TexDiff	TexDiff	Texture mapped with single texture, diffuse colour applied	RGB: Color	none	none
TripleGlossMap		Triple texture, 2nd pass contains gloss map in alpha channel, 3rd pass (reflection) texture drawn additively	?	?	?
TripleGlossMapWithLightIntens		Triple texture, 2nd pass alpha channel gloss map, 3rd pass drawn additively affected by lightmaps if used	?	?	?
TripleGlossMapWithoutLightIntens		Triple texture, 2nd pass alpha channel gloss map, 3rd pass drawn additively not affected by lightmaps if used	?	?	?
TripleTexDiffAddAdd		Triple textured, 2nd and 3rd passes are drawn additively	?	?	?
TripleTexDiffTAlpha		Triple textured, with diffuse colour, using each texture's alpha channels to blend between each pair of passes	?	?	?
TripleTexDiffTAlphaVAlpha		Triple textured, with diffuse colour, pass 2 uses texture alpha for blending, pass 3 uses vertex alpha for blending	?	?	?
TripleTexDiffVAlpha		Triple textured, with diffuse colour, using same vertex alpha to blend between each pair of passes	?	?	?
TripleTexDiffVAlphaTAlpha		Triple textured, with diffuse colour, pass 2 uses vertex alpha for blending, pass 3 uses texture alpha for blending	?	?	?

fx-Shader					
Shader Name (fx)	**Short Name**	**Description**	**Texture 1**	**Texture 2**	**Texture 3**
TrainEnv.fx	TrEnv		RGB: Color	RGB: Dummy	none
LoftTexDiff.fx	LoftTexDiff		RGB: Color	none	none
LoftTexDiffTrans.fx	LoftTexDiffTr		RGB: Color A: Transp.	none	none
LoftBump.fx		Diffuse texture and normal map	?	?	?
LoftBumpAlpha.fx		Diffuse texure with alpha and normal map	?	?	?
LoftBumpTrans.fx		Diffuse texture with 1-bit alpha and normal map	?	?	?
SkinAmbient.fx		Single colour skinned	?	?	?

fx-Shader					
Shader Name (fx)	**Short Name**	**Description**	**Texture 1**	**Texture 2**	**Texture 3**
SkinDiffuse.fx	Skin	Textured skinned.	RGB: Color A: Transp.	none	none
SkinGloss.fx		Textured, normal mapped, specular with gloss map, and skinned.	RGB: Color	RGB: Normal Map	RGB: Gloss Map
SkinNormal.fx		Textured, normal mapped, specular and skinned.	RGB: Color	RGB: Normal Map	none
SkinSpecular.fx		Textured, specular and skinned.	RGB: Color	none	none
StencilShadow.fx	Shadow	Stencil shadow objects, material must begin with shadow_ to be detected	RGB: Color	none	none
TrainBasicObjectDiffuse.fx	TrDiff	Single texture, dynamic lighting.	RGB: Color	none	none
TrainBasicObjectSpecular.fx	TrSpec	Texture, colour modulated specular.	RGB: Color A: Transp.	RGB: Spec color map	none
TrainBumpEnv.fx		Textured, normal mapped, environment mapped.	RGB: Color	RGB Normal Map	RGB: Dummy (Cubic Env)
TrainBumpEnvMask.fx		Textured, normal mapped, masked environment map.	RGB: Color A: Env Mask	RGB: Normal map	RGB: Dummy (Cubic Env)
TrainBumpSpec.fx	TrBumpSpec	Textured, normal mapped, specular.	RGB: Color A: Transp.	none	none
TrainBumpSpecEnv.fx	TrBumpSpecEM	Textured, normal mapped, environment map and specular.	RGB: Color	RGB Normal Map	RGB: Dummy (Cubic Env)
TrainBumpSpecEnvMask.fx		Textured, normal mapped, masked environment map and specular.	RGB: Color A: Env & Spec Mask	RGB: Normal map	RGB: Dummy (Cubic Env)
TrainBumpSpecMask.fx		Textured, normal mapped, masked specular.	RGB: Color A: Env Mask	RGB: Normal map	none
TrainFlora.fx	TrFlora	Ambient lighting, single texture.	RGB: Color	none	none
TrainGlass.fx		Screen space refractive glass with normal map and diffuse.	RGB: Color	RGB: Normal map	Back buffer copy
TrainLightMapWithDiffuse.fx	TrLightMap	Diffuse tex, lightmap, dynamic lighting.	RGB: Color	RGB Lightmap	none
TrainSkyDome.fx	Sky	Skydome	RGB: Color	RGB: Dummy (Cubic Env)	none
TrainSpecEnv.fx		Textured, vertex environment mapped with specular.	RGB: Color	RGB: Dummy (Cubic Env)	none
TrainSpecEnvMask.fx	TrSpecEM	Textured, masked vertex environment mapped with specular.	RGB: Color A: Env & Spec Mask	RGB: Dummy (Cubic Env)	none
TrainUprightViewFacingFlora.fx	TrUpVFaceFlora	Single texture, globally lit, upright view facing	RGB: Color A: Transp.	none	none
TrainVertexLit.fx		Diffuse tex, vertex lighting only.	RGB: Color	none	none
TrainVertexLitWithDiffuse.fx		Diffuse tex, vertex lighting, dynamic lighting.	RGB: Color	none	none

fx-Shader					
Shader Name (fx)	**Short Name**	**Description**	**Texture 1**	**Texture 2**	**Texture 3**
TrainViewFacingFlora.fx	TrVFaceFlora	Single texture, globally lit, view facing	RGB: Color A: Transp.	none	none
WaterCubeMap.fx	Water	Splish	RGB: Color A: Transp.	RGB: Normal map	none
SkinRESERVED1.fx		RESERVED FOR FUTURE USE			
SkinRESERVED2.fx		RESERVED FOR FUTURE USE			
SkinRESERVED3.fx		RESERVED FOR FUTURE USE			
TrainBumpEnv.fx		Textured, vertex environment mapped.	RGB: Color	RGB: Dummy (Cubic Env)	none
TrainBumpEnvMask.fx		Textured, masked vertex environment map.	RGB: Color A: Env. Mask	RGB: Dummy (Cubic Env)	none

11 Data modification during export

11.1 Overview

Bigex has a very powerful function built in for data manipulation during the export which allows the setting of ***every*** entry in the IA/IGS file to an arbitrary value. For example it can be used for replacing the short form of the shader names into the full shader name or replacing the path and name of a texture file e.g. for dynamic numbering.

The Blender internal data structures remain unchanged. Only the exported data values are changed.

Basically all data can be manipulated when exported into the IA/IGS format file. These are not only strings like shader and object names but also all int and float number values. This for example enables the manipulation of the floating point numbers of the world colour in the IGS file.

Manipulating the data follows the search and replace principle. The built in search & replace function also supports regular expressions which follow the Python syntax (see [11]). This means that the search text can contain search blocks, which can be referenced/reused in the replace text.

Bigex reads the search and replace expressions from two text files. Both these files are expected to be in the chosen target folder for the IA/IGS file. If one or both exist then *Bigex* reads them and applies the search and replace expressions to the exported data.

Right at beginning of the export process *Bigex* writes some status information into the console window, and if enabled into a log file. These contain information about where both search & replace text files were found and how many valid expressions were read.

Both text files with the replacement expressions separate in one with a fixed name and one with a fixed extension to the IA/IGS export file name. The first with the fixed name ***IGS_ExpModFile.txt*** resp. ***IA_ExpModFile.txt*** can be used for replacements which are the same over all 3D models – such as shader names.

The name for the model specific text files is built from the chosen IA/IGS export file name plus the fixed extension ***_IGSExpModFile.txt***, resp. ***_IGSExpModFile.txt***.

Example: If the IA/IGS export file name is *MyEngine.igs*, then the resulting filename for the text file with the replacement expressions is *MyEngine_IGSExpModFile.txt* resp. *MyEngine_IAExpModFile.txt*

Two example search and replace text files are in the ***Bigex_ExampleObjects*** subfolder of the installation package.

11.2 Contents of the search and replace files

The syntax of the expressions in the base and model specific search and replace files are identical. Both can contain blank lines, comment lines, lines with search domain header or lines with search and replace expressions.

Note: Comment lines do have as **first character in the line** a **#** and are ignored by *Bigex* like blank lines.

11.3 Search domain header

Replacement commands are preceded by a search domain specification, consisting of the name of an object list and the name of the attribute to be modified.

> **Note:** Lines containing a search domain start with a **[** as **very first character in the line**, followed by the object list name, a colon **:**, the name of the attribute of a list entry and finally a **]** as **last character in the line**.

All replace expressions until the next search domain header, resp. until the end of file, are applied for the search range only which is specified by the search domain header.

Example:

```
...
[Materials:ShaderName]
# Shader Name (fx)
TrDiff=TrainBasicObjectDiffuse.fx
TrSpec=TrainBasicObjectSpecular.fx

[Objects:Name]
A_FSSeitFenst_Rg=1_0200_Seitenfenster_Rechts
...
```

The first two search and replace expressions are applied on the *ShaderName* attribute of all materials in the *Materials* object list.

The third search and replace expression will be applied on the *Name* attribute of all objects in the O*bjects* object list.

11.4 Search and replace expressions

One line contains one search and replace expression.

> **Note:** Lines with search and replace expressions consist of a search string, starting at the beginning of the line, ending before the **=**, the **=** itself and a replacement string, starting after the **=**, ending with the end of the line.

Three different cases are differentiated for search strings.

1. ***Unconditioned replacement:*** If the search string is blank, which means the assignment character **=** is the first one in a line, then the specified attribute of all objects of an object list is replaced. This is indenpent of the value it holds currently.
2. **Limited on certain objects of a list:** If the search string starts with **[n]** or **[n-m]**, then the **n-th** resp. the **n-th to m-th** objects of an object list are searched and replaced where applicable.
3. **Limited on certain content of attributes:** If the search string does not start with **[** or **=**, then it is searched for values of attributes, which are **exactly** matching the search string.

```
Example:
...
# Search domain: Materials object list, attribute: ShaderName
[Materials:ShaderName]
# 1. Replaces all shader names of all materials with SName2
=SName2

# 2. Replaces only shader names of all materials with SName2 if the ShaderName is SName1
SName1=SName2

# 3. Replaces shader name of material 99 with SName2 but only if it is SName1
[99]SName1=SName2

# 4. Replaces shader name of materials 88-99 with SName2 but only if it is SName1
[88-99]SName1=SName2

# 5. Replaces shader name of material 99 with SName2 regardless what it was before
[99]=SName2
...
```

11.5 IGS and IA object lists

All objects from the following object lists can be manipulated at the export into the IGS format:

- IGfHeaders
- IGfObjects
- IGfMeshLODs
- IGfMeshes
- IGfVerts
- IGfVertexBoneBindings
- IGfTriangles
- IGfBones
- IGfMaterials
- IGfLights
- IGfSplines
- IGfGenericItems
- IGfGenericData
- IGfDataBlocks
- IGfData
- IGfFaceTagDescriptions
- IGfTextureNames

All objects from the following object lists can be manipulated at the export into the IA format:

- IAfHeaders
- IAfTriggers
- IAfAnimationNodes
- IAfMotionControllers
- IAfMotionSets

11.6 Attributes of objects

All the different objects have many attributes. There are hundreds different attribute names. To find out the exact name spelling, it is best to run the export once ***without*** any replacement. Afterwards you can copy and paste the exact attribute name you want to modify from the log file.

Example: Replacing an object name.

Excerpt of the log file after an export ***without*** replacement:

```
...
================================================================================
IGfObjects                    : [ 42 ]
--------------------------------------------------------------------------------
IGfObjects [ 1  /  42  ] (390932) : 'AB_FSSeitFenst_Rg'
------------------------------
Name                          : 'AB_FSSeitFenst_Rg'
NameUniqueID                  : 0
MeshLODCount                  : 4
MeshLODListStartOffset        : 390708 ==> IGfMeshLODs[0]
...
```

If the following replacement command is used:

```
...
[Objects:Name]
AB_FSSeitFenst_Rg=1_0500_Seitenfenster_Rechts
...
```

then the object name `1_0500_Seitenfenster_Rechts` instead of `AB_FSSeitFenst_Rg` will be written into the IGS file.

Excerpt of the log file ***with*** replacement:

```
...
================================================================================
IGfObjects                    : [ 42 ]
--------------------------------------------------------------------------------
IGfObjects [ 1  /  42  ] (390932) : '1_0500_Seitenfenster_Rechts'
------------------------------
Name                          : '1_0500_Seitenfenster_Rechts'
NameUniqueID                  : 0
MeshLODCount                  : 4
MeshLODListStartOffset        : 390708 ==> IGfMeshLODs[0]
...
```

Note: Further examples for search and replace expressions can be found in the example files ***IGS_ExpModFile.txt*** and ***Bigex_ExampleObject_1_IGSExpModFile.txt*** in the subfolder ***Bigex_ExampleObjects*** of the installation package.

12 Running Bigex in command line mode

As well as it's use as an IA/IGS exporter from Blender, the *Bigex* Python script can be used as a IA/IGS file analyser. This operation mode can be started from command line console window.

The analyser mode allows you to dump out the contents of a IA/IGS file into the console window or optionally into a log file. It is also possible to generate a comma separated value (csv) file which can easily be imported as a spreadsheet like Openoffice Calc or Microsoft Excel.

The behavior of the *Bigex* script can be controlled with options on the command line.

Running the *Bigex* script under Python from the command line is quite easy. You simply need to call ***bigex25.pyc*** if you are using Python 2.5, resp. ***bigex26.pyc*** if you are using Python 2.6.

Under Linux, open a new console window via the Start menu and type the command:

`python ~/.blender/scripts/bigex25.pyc --help` (for Python 2.5.x) or
`python ~/.blender/scripts/bigex26.pyc --help` (for Python 2.6.x)

Under Windows, open a new console window via the Start menu (Start => Run: "cmd" => OK) and type the command:

`C:\Program Files\Python\python.exe <PathToBigex>\bigex25.pyc --help` (for Python 2.5.x) or
`C:\Program Files\Python\python.exe <PathToBigex>\bigex26.pyc --help` (for Python 2.6.x)

It may be required to adjust the path names to the Python executable. The Blender scripts are usually in folder:

<programme drive letter>\:<program folder>\<Blender folder>\.blender\scripts

e.g.: C\:Programme\Blender\.blender\scripts

Once successfully started, the option '--help' dumps out the following help text:

```
Usage: bigex25.pyc [Options] (IGS-filename.igs|IA-filename.ia)

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -l, --logging         Enables logging. Writes all information from IGS/IA
                        file into a log file.
                        usage -l [LogFileName]
                        If LogFileName is omitted, default output file name is
                        <IGS/IA-filename>_imp.log
                        Default is False.
  -m, --modify          Enables IGS/IA file modifying. IGS/IA file will be
                        read in, modified and written back. Modify expressions
                        are read from specified modify file.
                        usage -m [ModFileName]
                        If ModFileName is omitted, default file name
                        '[IGS|IA]_ImpModFile.txt' is used.
                        If a modify file name is specified, it is
                        used additionally
                        Default is False.
  -k, --headeronly      Dumps out header information of IGS/IA file only.
                        Default is False
  -t, --timestamps      Prints timestamps with information about elapsed time.
                        Default is False
```

```
-o OBJECTLIST, --objectlist=OBJECTLIST
                      Writes all elements of OBJECTLIST list into a csv-
                      file. Default csv-file name is:
                      <IGS/IA-filename>_<objectlist>.csv
                      -o help       shows a list with all object list names
                      -o OBJECTLIST dumps out object list OBJECTLIST
                      -o all        dumps out all object lists
                      -o nonempty   dumps out all non-empty object lists
                      Default is False
-a, --addresslist     Dumps out addresses of objects in IGS/IA file.
                      Default is False
-s, --summary         Dumps out a summary at the end of the export.
                      Default is False
-v, --verbose         Dumps out very detailed information.
                      Default is False
-q, --quiet           Dumps minimum information only. Writes no log file,
                      no time stamps, no summary.
                      Default is False
```

If requested, new files will be created in the same folder where the source IA/IGS file resides.

Some examples for calling the script:

```
... bigex26.pyc -k -s MyIGSfile.igs
```

dumps out only the header and a short summary of the contents of *MyIGSfile.igs*. This gives a quick overview of what is in the IGS file.

```
.... bigex26.pyc -l -s -t -v MyIGSfile.igs
```

dumps complete info about the contents of the IGS file. Including all details on all objects plus summary and the elapsed time for each step.

```
... bigex26.pyc -o help MyIGSfile.igs
```

lists all object list names of the IGS file.

```
... bigex26.pyc -l -s -t -v -o all MyIGSfile.igs
```

dumps out detailed info about the contents of the IGS file. Plus one csv file is generated for each object list which holds at least one object.

13 Sending problem reports

No piece of software is without errors. Thus, if the export script is aborting e.g. with an error message, please send an error report to the mail address listed in section 5 Copyright, Licence, Warranty, Contact, Credits. This helps to improve *Bigex*. Before sending please consider the following:

- verify that your runtime environment is as described in this manual
- verify that the problem is repeatable at least three times
- reduce your Blender model to objects only, which are causing the problem at export. This may either objects which cause the exporter to abort with an error message in the console window or objects which are not propperly exported or not exported at all.

To do a root cause analysis of the problem, the following information need to be available in a ZIP file:

- Output of the console window containing the whole export process which causes the problem.
- Log file, created with option *Verbose* enabled in the user GUI.
- Blender .blend file containing the object only which causes the problem.

14 Annex

14.1 Mapping of Blender data on the IGS format

During the export into the IGS format all necessary 3D model data from Blender needs to be mapped into the equivalent IGS object attributes. Because Blender has lots of sliders, input fields and buttons, it is not obvious which settings in Blender relate to which IGS file attributes.

The following table shows which input fields of Blender are mapped to which entries of the IGS files.

If you set the value of a certain input field in Blender, e.g. the name of a texture image file, then you will find that value again in the correct place in the log file i.e. in TextureNameList -> TextureName[0].

Blender				**IGS File**			
Blender window	**Panel**	**Tab**	**Field**	**IGS Object**	**Attribute**	**Comment**	**Activ**
Button Window	Scene(F10)	Format	FPS	Material	RSx.FPS	The Scene value is set for all active Render Stages.	yes
Scripts Window	*Bigex* GUI	Material	RSxArgy	Material	RSx.UVArguments	The IGS format can hold the first 6 of the 8 arguments from the *Bigex* GUI.	yes
Scripts Window	*Bigex* GUI	Material	Viewer Facing	Material	Viewer Facing		yes
Button Window	Shading(F5)	Shaders	LBias	Material	RSx.MipLODBias	The Shading value is set for all active Render Stages.	no (not accessible via Python)
Button Window	Shading(F5)	Links and Pipeline	Zoffs	Material	ZBias		yes
Button Window			ZTransp				yes
Button Window	World Buttons	World	AmbR AmbG AmbB	Header	AmbientColor Red Green Blue		yes
- - -	- - -	- - -	- - -	Header	AmbientColor Alpha	Set fix on 0.0.	yes
- - -	- - -	- - -	- - -	Material	AmbientColor RGB	Set fix on 1.0	yes
Button Window	Material Buttons	Material	Col RGB	Material	DiffuseColor RGB		yes
Button Window	Material Buttons	Material	Spe RGB	Material	SpecularColor RGB		yes
Button Window	Material Buttons	Material	Mir RGB	Material	EmissiveColor RGB		yes
Button Window	Material Buttons	Material	A	Material	DiffuseColor SpecularColor EmissiveColor Alpha		yes

Blender				IGS File			
Blender window	**Panel**	**Tab**	**Field**	**IGS Object**	**Attribute**	**Comment**	**Activ**
Button Window	Material Buttons	Shaders	Spec	Material	SpecularPower	To get a wider range for SpecularPower, the both Blender fields are multiplied.	yes
Button Window	Material Buttons	Shaders	Hard				
Button Window	Material Buttons	Texture	Erster aktiver Textur Channel Name	Material	ShaderName		yes
- - -	- - -	- - -	- - -	Material	RenderStageCount	Automatical: Active Texture Layer count minus 1.	yes
Button Window	Material Buttons	Shaders	Emit	Material	EmissiveStrength		yes

14.2 References

[1] http://home.exetel.com.au/randomsoftware

[2] http://www.blender.org/

[3] http://store.steampowered.com/app/5287

[4] http://www.railsimulator.com/

[5] http://www.python.org oder http://www.python.de

[6] http://www.gimp.org

[7] http://www.gnu.org/licenses/gpl.html, http://www.gnu.de/documents/gpl-3.0.de.html

[8] http://rail-sim.de, http://rail-sim.de/railsim/forumnew/index.php?board=8.0

[9] http://forums.uktrainsim.com, http://forums.uktrainsim.com/viewforum.php?f=291

[10] http://www.railsimdownloads.com/wiki/tiki-index.php

[11] http://docs.python.org/library/re.html

[12] http://www.railsimulator.com/en/railsimfiles