RubyTMIX:
A TrainzXML Exporter
for Google™ SketchUp®

User's Guide

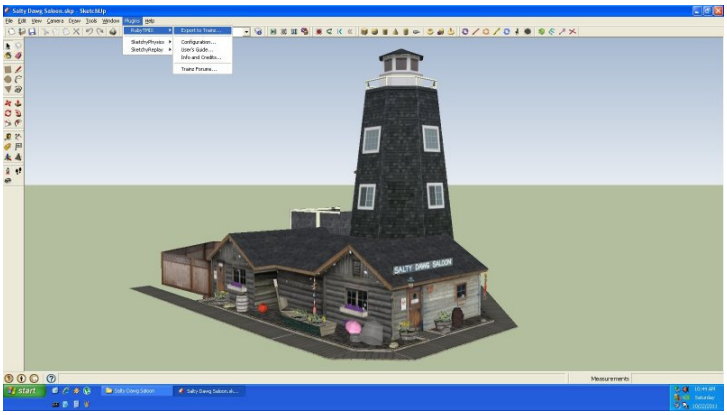
by Mike Jenkins (ModelerMJ / 606026)

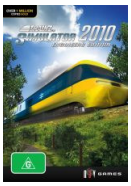
December 2012

This manual describes Version 1.2 of *RubyTMIX*, a plug-in for Google™ SketchUp® that provides direct export of SketchUp models to the Trainz™ railroad simulator environment with all geometry, materials and textures intact and the polygon count minimized.

Using a combination of its own built-in functionality as well as fully-automated interaction with various freely-available external utility programs, the plug-in produces all files required to import a SketchUp model directly into Trainz Content Manager, where it is immediately available for use in Surveyor.

The plug-in is compatible with Google SketchUp 7 and 8; with Windows XP and Windows 7; and with TS2009, TS2010 and TS12. It's named for the programming language in which it's written (**Ruby**) and the intermediate file format used during the export process (**T**rainz **M**esh **I**mporter **X**ML).





Initial Release / 1.0.0: November 2011
This Version / 1.2.0: December 2012

The information in this document is subject to change without notice and should not be construed as a commitment by the author. Although every effort has been made to ensure accuracy, the author assumes no responsibility for any errors that may appear herein.

If You Have Problems with RubyTMIX…

Please contact me at mike@dhobh.net (or post in the Trainz forums, or send me a PM from there) and I will do my best to assist you. DO NOT send attachments to me unless I ask you to; if I receive an e-mail with attachments from a source I don't know, it will be deleted unread.

If You Have Problems with SketchUp…

I am NOT your go-to guy for this. There is a wealth of information related to SketchUp on the internet, including tutorials and how-to videos, so you would do well to seek a solution to your issue(s) from one of these sources. I also heartily recommend the book *Google SketchUp 8 for Dummies*; more information about this can be found at http://www.aidanchopra.com/.

Acknowledgements

Thanks to Peter Villaume (PEV) for generously sharing his knowledge of TrainzXML files and Trainz geometry, and for allowing me to link to his Trainz Mesh Viewer as part of my tool.

Thanks to philskene for letting me use his *Port Ogden & Northern* layout as a backdrop for my screen shots.

Thanks to those who participated in the Beta test phase and provided feedback (especially Arraial P, who found many obscure issues and whose observations and suggestions played a large part in the development of this tool).

And, finally, thanks to everyone in the Trainz community for their kind words and support.

Regards,
- Mike Jenkins (ModelerMJ / 606026)

Contents

This page intentionally left blank.

Preface

This manual describes Version 1.2 of RubyTMIX, a plug-in for Google™ SketchUp® that provides direct export of SketchUp models to the Trainz™ railroad simulator environment with all geometry, materials and textures intact and the polygon count minimized. Using a combination of its own built-in functionality as well as fully-automated interaction with various freely-available external utility programs, the plug-in produces all files required to import a SketchUp model directly into Trainz Content Manager, where it is immediately available for use in Trainz Surveyor.

This manual contains 4 chapters:

- Chapter 1, *Introduction*, provides a general description of RubyTMIX and its current capabilities.
- Chapter 2, *Installation and One-Time Configuration,* describes how to install RubyTMIX and how to perform initial setup of the plug-in from within Google SketchUp.
- Chapter 3, *Exporting SketchUp Models to Trainz,* describes the basic process by which models are exported from Google SketchUp and subsequently imported into Trainz via Content Manager.
- Chapter 4, *Making Simple Spline Objects,* describes the process by which simple (one-piece) spline-type objects can be created in SketchUp and exported to Trainz with RubyTMIX.
- Chapter 5, *Troubleshooting and Advanced Topics*, describes how to deal with common problems that may arise when attempting to export SketchUp models to Trainz, and also provides information for advanced users who may wish to perform some or all of the post-processing steps themselves.

It is anticipated that this will be a living document that grows and evolves as new features are added and existing features are enhanced or changed. It is also my intention that tutorials will be developed separately to address the actual use of SketchUp to produce content for Trainz, so that type of material has intentionally been left out of this document.

This page intentionally left blank.

Legal Considerations

RubyTMIX is being made freely available to the Trainz community and may be used for any purpose, personal or commercial, within the terms stated on the inside cover of this document. **If you don't agree to those terms… don't use the software.**

Of somewhat greater importance is the way in which the Google SketchUp *Terms Of Use* apply to RubyTMIX. I have exchanged several e-mails with the correct person at Google regarding this, and the following list summarizes what we discussed.

1. The free edition of Google SketchUp may not be used for commercial purposes. This means that you cannot sell the models you create with the free edition – it does not mean you cannot use it to create models for the Trainz environment. What this boils down to is that the only acceptable way of distributing Trainz models created with the free edition of Google SketchUp is to post them on the Trainz DLS so that they are freely available to everyone (the need for a First-Class-Ticket not withstanding).

2. Payware items created for Trainz, e.g. routes, may reference models created with the free edition of Google SketchUp that have been posted on the DLS, but may not directly include any such models in the distributed payware package. This is (I believe) in line with the existing policies of N3V/Auran regarding the use of freeware within payware.

3. Materials, textures, components and Google Street View or Google Earth images that are accessible from within Google SketchUp or other Google products may be used in the creation of Trainz models. However, all such materials must remain part of the model and may not be separated from it for use in any other content creation tool. This means that, even though individual images that represent textures and photos may be accessible at various points in the export/import process, those images must remain part of the single object asset that is held in Trainz and cannot be re-used in any other non-SketchUp-produced asset.

4. Section 8.2 of the Google 3D Warehouse Terms of Service (which can be found at http://sketchup.google.com/intl/en/3dwh/tos.html) includes the statement "For the avoidance of doubt, you may modify, distribute, and create derivative works of Content uploaded by other users in 3D Warehouse." I have interpreted this to mean that models can be downloaded from the Google 3D Warehouse, converted via RubyTMIX, and placed on the Trainz DLS in either altered or original unchanged form as desired. Proper etiquette dictates, of course, that the permission of the original author should be sought and that they (and the Google 3D Warehouse) should be named as the original source(s).

In summary, I was told that nothing I intended to do required a special agreement with Google; therefore I feel safe in saying that we are allowed to use the free edition of Google SketchUp to produce content for Trainz as outlined above.

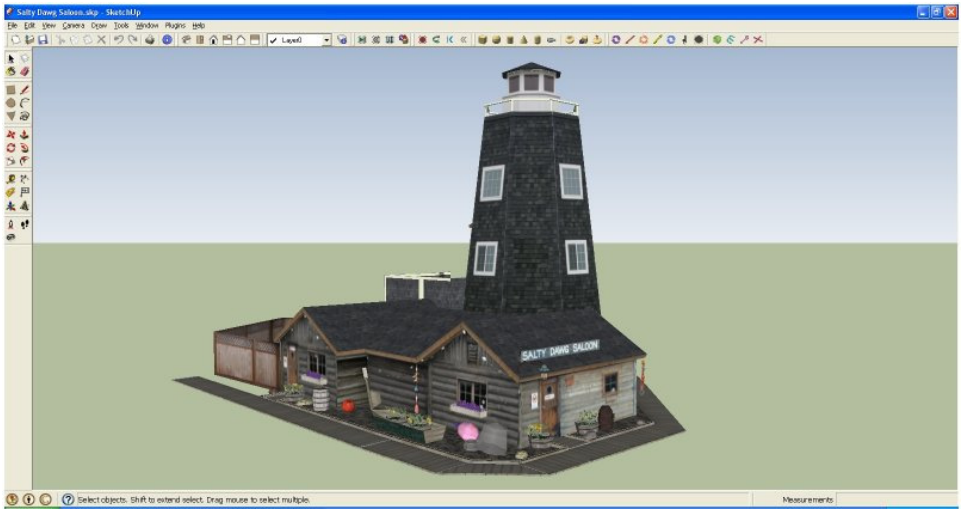
If you disagree with my assessment… don't use the software.

This page intentionally left blank.

1 Introduction

RubyTMIX is a plug-in for Google™ SketchUp® that provides direct export of SketchUp models to the Trainz™ railroad simulator environment with all geometry, materials and textures intact and the polygon count minimized. Using a combination of its own built-in functionality as well as fully-automated interaction with various freely-available external utility programs, the plug-in produces all files required to import a SketchUp model directly into Trainz Content Manager, where it is immediately available for use in Surveyor. The plug-in is compatible with Google SketchUp 7 and 8; with Windows XP and Windows 7; and with TS2009, TS2010 and TS12.

A typical result of the process is shown below. The upper screen shot is a model from the Google 3D Warehouse (*Salty Dawg Saloon* by JCHarrist), in Google SketchUp; while the lower screen shot is the same model in TS12 Surveyor (in the *Port Ogden & Northern* route, used as a backdrop courtesy of philskene), after being processed by RubyTMIX and imported via CM.



1.1 Features & Limitations

RubyTMIX provides a range of features that are designed to make it flexible in terms of assisting the content creator only as much as may be desired. At one end of the spectrum this allows the export process to be used in an entirely automated fashion, while at the other end the content creator has the ability to tweak many aspects of the exported model before it is imported into Trainz.

Implementation

- Ruby plug-in, runs inside Google SketchUp 7 or 8 & integrates into the menu bar.
- User interface is provided via Ruby-SketchUp 'Web Dialogs'.
- Requires Trainz Mesh Importer (included).
- Requires the Microsoft .NET 2.0 runtime (usually already part of your Windows OS).
- Supports preview of produced IM file in PEV's Trainz Mesh Viewer (obtained separately) prior to import into Trainz Content Manager.

Main Features

- Automated production of all files required for use with TS2009, TS2010 and TS12.
- Immediate error- and warning-free import of all files into Content Manger from a single folder.
- Imported assets are ready for immediate use in Surveyor.
- No other content-creation tools required (e.g. Blender or G-Max).
- **IT'S FREE!**

How It Works

- Requires each SketchUp model (**.skp** file) to be in its own folder.
- Automatically creates all required files and folders based on file name information from SketchUp
- Exports geometry and materials descriptions directly to TrainzXML format.
- Automatic geometry selection logic minimizes poly count by exporting only the faces required.
- Manual selection of front / back / both / textured-only geometry is also provided.
- Export full model or selection only.
- Automatically compensates for many mirrored component configurations.
- Correctly handles projected and positioned textures.
- Exports SketchUp colors as non-textured materials, fully defined in the TrainzXML file.

- Exports SketchUp textures as image files with auto-resizing to closest-power-of-2 dimensions and 1:8 max ratio as needed.
- Textures are exported using controlled file names to eliminate issues with Unicode characters that would cause Trainz Content Manager to reject the images.
- Original texture images are saved separately, with their actual names, along with a cross-reference that tells which controlled file name goes with which actual texture.
- Auto-converts JPG and BMP material and photo-texture files to 24-bit TGA format.
- Auto-converts PNG and TIF textures with transparency to 32-bit TGA format and maintains alpha information.
- Automatic creation of 'config.txt' file -- you specify the defaults for all required items.
- Provides options to automatically add tags to the 'config.txt' file as required to support the creation of surveyor-only and simple (one-piece) spline-type objects.
- Creates a JPG thumbnail from the SketchUp model and auto-resizes it to 240 x 180.
- Automatically runs Trainz Mesh Importer at conclusion of process (also creates CMD file for manual re-run).
- All post-processing options can be turned on or off, so you can tweak the results yourself.
- Retains all configurable options separately for every model.
- Creates a log file for each model that documents its most recent export run to aid in troubleshooting.

There are also some limitations that you need to be aware of.

- Currently the plug-in is capable of processing only static scenery objects and simple splines. I hope to add other object types in the future.
- No support is currently provided for attachment points or animations. Attachment points can, however, be added using PEV's *Attachment Maker* tool, which is freely available at http://members.optusnet.com.au/~villaump/pevsoft.htm.
- The plug-in currently exports SketchUp textures individually. If your model has a large number of textures this can result in poor performance when it is used in Trainz. There are features in SketchUp that can be used to reduce the number of textures exported; this is, however, a complex subject that is best covered in a separate Tutorial.
- The plug-in only operates under Windows. It does not operate on the Mac.
- Versions of SketchUp prior to 7.1 have not been tested and there is no guarantee of proper functionality in those versions.

1.2 Audience

This manual is written for those who are already familiar with how Google SketchUp works, and are at least somewhat proficient at creating models. I also expect that (as a content creator) you will have a good understanding of Trainz KUIDs and 'config.txt' files, and that you know how to import content into Trainz Content Manager.

1.3 What You Need

In order to use RubyTMIX, you will need the following items.

- Trainz software (TS2009, TS2010 or TS12) installed and operational on your PC.
- Google SketchUp software installed and operational on your PC. To download this for free, go to http://sketchup.google.com/intl/en/download/index.html; this document references Version 8.
- PEV's Trainz Mesh Viewer software installed and operational on your PC. To download this for free, go to http://members.optusnet.com.au/~villaump/pevsoft.htm. After download, simply run its installer; you can change the installation location if you like (although there is usually no reason to). Some additional setup steps may be needed the first time you try to preview a model that you have exported via RubyTMIX, but the process is very simple and we'll cover that when it's required.
- The Trainz Mesh Importer utility, which converts TrainzXML files into Trainz *Indexed Mesh* (IM) files. **This is included as part of the RubyTMIX installation,** and if you install RubyTMIX as instructed no configuration will be necessary in order for it to operate properly.
- The Microsoft .NET 2.0 Runtime is required to support the image post-processor that comes with RubyTMIX. This should already be present in all operating systems, and should certainly be on your Trainz PC since Trainz itself requires this. If needed, you can download this at http://www.microsoft.com/download/en/details.aspx?id=19.
- Last but not least, you need the RubyTMIX plug-in for Google SketchUp. This is distributed as a ZIP archive which contains the PDF manual you are reading now, and also contains a second (embedded) ZIP archive that provides the actual RubyTMIX installation. We'll talk more about that shortly.

1.4 Recommended Additional Utilities

Other utilities exist that can make it easier to use RubyTMIX and the files it produces.

- IrfanView is commonly used to resize images and perform format conversions. To download this for free, go to http://www.irfanview.com/. **However, you should be aware that IrfanView does not preserve alpha channel (transparency) information if you open a PNG or TIF image and then save it as TGA**.
- PEV's Images2TGA Utility will be useful if you want to perform your own image post-processing operations; get this from the same place as the Trainz Mesh Viewer.

- Notepad++ is a tabbed editor that includes color-coding for many types of files. It is freely available at http://notepad-plus-plus.org/. You may wish to change the file association for files of type **XML** to open with Notepad++ instead of with the default program (which is usually Internet Explorer), so that you can view the TrainzXML files produced by RubyTMIX. For instructions, search the Internet for "how to set file associations" and choose a link that is appropriate for your operating system.

1.5 Notes About the Recent Transfer of SketchUp Ownership

Since RubyTMIX was originally developed, Google has sold its interests in SketchUp to a company named Trimble. The free version of SketchUp continues to remain available as it was under Google ownership, and the terms of use remain unchanged. Therefore this transition of ownership of the SketchUp product should have no effect on the use of SketchUp in relation to Trainz or RubyTMIX.

This page intentionally left blank.

2 Installation and One-Time Configuration

This chapter describes how to install RubyTMIX and how to perform initial setup of the plug-in from within Google SketchUp.

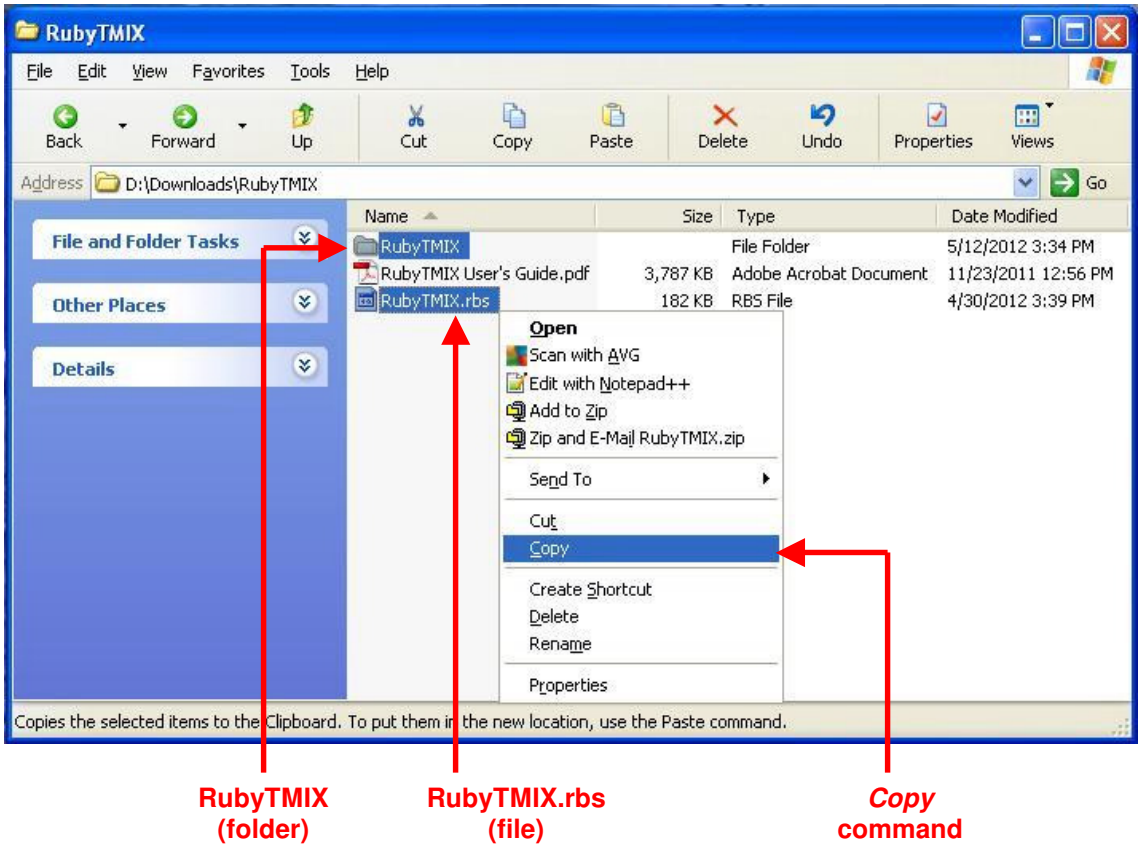
2.1 Installing RubyTMIX Into Google SketchUp

When you obtained RubyTMIX, you should have downloaded a single ZIP file that contains the PDF document you are reading now, plus two other items:

- A file named *RubyTMIX.rbs*. **This is the plug-in executable and must be copied into the Google SketchUp *Plugins* folder**; we'll discuss this in a moment.
- A folder named *RubyTMIX*. **This folder contains support files required by the RubyTMIX executable, and must also be copied into the Google SketchUp *Plugins* folder**.

Installing RubyTMIX into Google SketchUp is a straight-forward **manual process**.

1. In the folder where you unzipped the RubyTMIX download, select *RubyTMIX.rbs* and the folder named *RubyTMIX*; to select multiple items at once, hold down **Ctrl** while you click the items until they are both highlighted at the same time.
2. Type **Ctrl+C**, or **right-click** on either selected item and choose *Copy* from the context menu that appears.



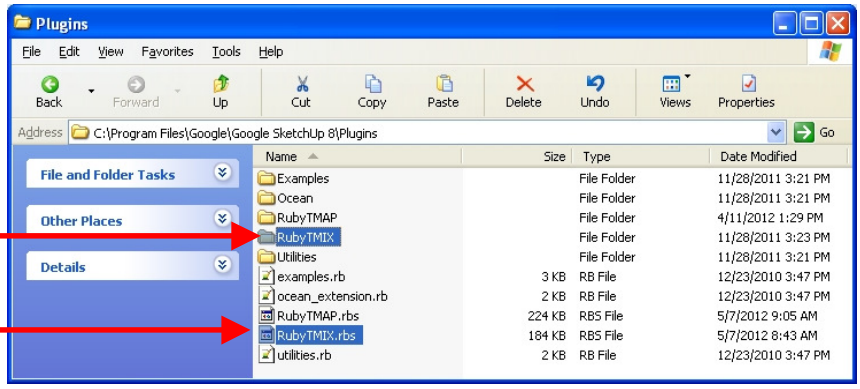
3. Find the location of the Google SketchUp *Plugins* folder. One easy way to do this is to start by finding the location of Google SketchUp itself.
    - On Windows XP systems, right-click the Google SketchUp icon your desktop, and select *Properties* from the context menu. In the *Google SketchUp 8 Properties* dialog box that appears, click the *Find Target* button; this will open Windows Explorer at the location where Google SketchUp is installed.
    - On Windows 7 systems, right-click the Google SketchUp icon on your desktop, and select *Open file location* from the context menu; this will open Windows Explorer at the location where Google SketchUp is installed.
4. Look in the Windows Explorer view that you just opened and find the folder named *Plugins*. Double-click this folder to open it; you are now at the location where the RubyTMIX files need to be installed.
5. Type **Ctrl+V**, or right-click in an empty space in the Windows Explorer view and choose *Paste* from the context menu. This will copy the RubyTMIX files into the *Plugins* folder, and you should now see something like the following illustrations.

Example of the *Plugins* folder in Windows XP.

The highlighted items are the ones you must copy into this folder.

RubyTMIX (folder)

RubyTMIX.rbs (file)

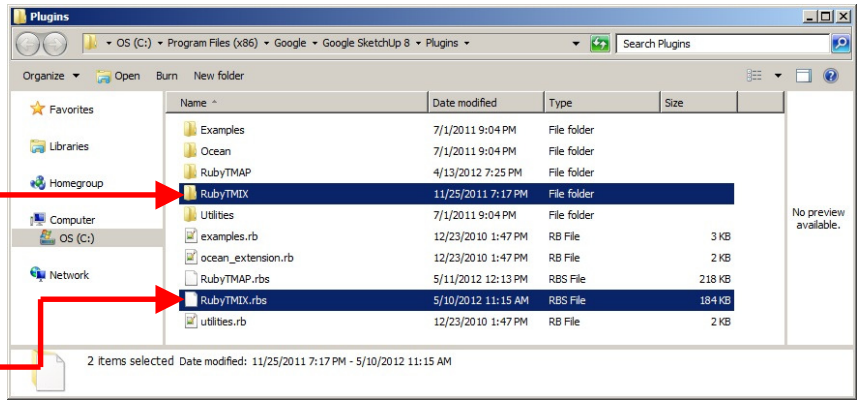


Example of the *Plugins* folder in Windows 7.

The highlighted items are the ones you must copy into this folder.

RubyTMIX (folder)

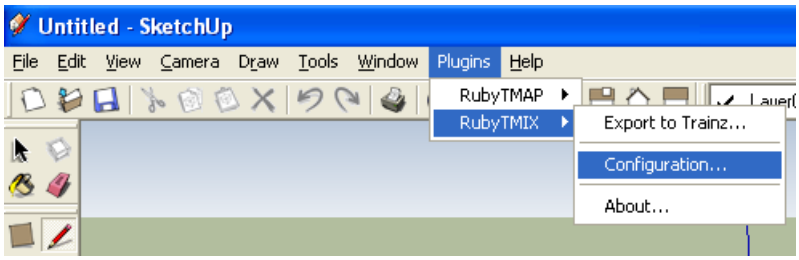
RubyTMIX.rbs (file)



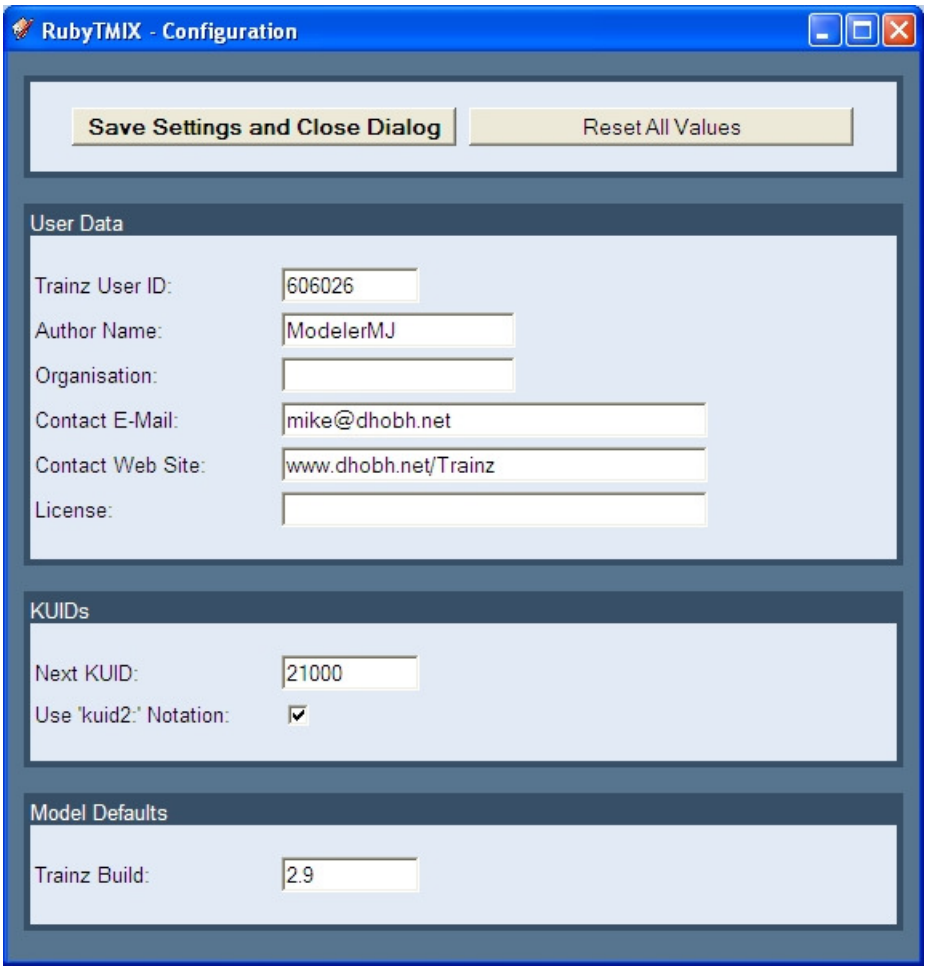
Installation of RubyTMIX into Google SketchUp is now complete.

2.2 Initial RubyTMIX Configuration

Now that the plug-in is installed into Google SketchUp, and you have all the necessary utility programs installed on your computer, the last thing that needs to be done is to configure the RubyTMIX plug-in itself. To begin, start Google SketchUp and click the *Plugins* menu. Highlight *RubyTMIX*, and you will see a set of sub-menus that looks like the example at the top of the next page. *If you don't see this menu, then you probably did not install RubyTMIX correctly; go back to Section 2.1 and check to be sure you performed all the steps exactly as described.*



Click the item named *Configuration*. This will display a dialog similar to the following.



An extra note is in order about the way dialogs like this work in Google SketchUp. They are not ordinary dialogs like you see inside of many programs; instead they are known as **Web Dialogs**, meaning that they are actually web pages that are being displayed by your default internet browser. That's why you will see minimize and maximize buttons (for security reasons, these cannot be disabled from within a Ruby script); although it won't hurt anything, you should avoid clicking them.

The *Configuration* dialog is divided into 4 main sections. These are as follows.

1. At the top are two buttons. As its name implies, *Save Settings and Close Dialog* will save any changes you have made and will remove this dialog from your screen. *Reset All Values* will set all the values in the entire dialog to defaults that are built into RubyTMIX; you will be asked to confirm that you want to do this before it actually takes place.
2. The *User Data* section allows you to enter information that will be placed into the 'config.txt' files that RubyTMIX automatically creates when you export a model for the first time (after the 'config.txt' file has been created, RubyTMIX will not overwrite it unless you tell it to, so you can manually make any changes needed and they will remain intact). The fields in this section are as follows.
    - Trainz User ID is your unique numeric identifier, which is embedded into the KUID of any models you create. This field is mandatory.
    - Author Name is the name you use on the Trainz forums. This will appear in Content Manager as your author name. This field is mandatory.
    - Organisation is the name of any 'company' that you might be operating in relation to Trainz. This field can be left blank if desired.
    - Contact E-Mail is a real-world e-mail address where you can be reached by people who use your models. This field can be left blank if desired.
    - Contact Web Site is the URL of the web site associated with your 'company'. This field can be left blank if desired.
    - License is the license text that you want to use for your models; you need to enter it all on one line. This field can be left blank if desired.
3. The *KUIDs* section allows you to define how RubyTMIX uses KUID numbers when it creates a 'config.txt' file.
    - Next KUID defines the KUID number that RubyTMIX will use for the next 'config.txt' file that is generated. After the file is generated, this number is incremented and saved for use with the next new model; that new number will appear here if you look at this dialog again. This field is mandatory, but you can override this number for any model on an individual basis.
    - Use 'kuid2' Notation determines if generated KUID numbers will be versioned (*kuid2* format) or not (*kuid* format). This is also something that you can override for any model on an individual basis.

4. The *Model Defaults* section currently contains only one item.

   - Trainz Build defines the Trainz build number that RubyTMIX will use for the next 'config.txt' file that is generated. **2.9** is the default value, which provides compatibility with TS2009, TS2020 and TS12. This field is mandatory.

When you've set all the required values, click *Save Settings and Close Dialog* to save your changes.

Note 1: While RubyTMIX does check to be sure you have filled in the mandatory fields (and will display a message if you missed one), it does not specifically check your entries for validity as far as what Trainz expects to see. So, it's up to you to ensure that your entries actually represent valid data; if you make a mistake, it won't show up until you try to import the model into Content Manager and find yourself faced with an error.

Note 2: Starting with version 1.2 of RubyTMIX, the master configuration settings are saved to the *Application Data* folder on your computer, rather than being saved within the *RubyTMIX* folder inside the Google SketchUp *Plugins* folder. This change was made to address security issues under Windows 7, which does not (normally) allow data to be written to the location(s) where executable programs are stored. If you previously had RubyTMIX installed on your system, the plugin will still use the old master configuration file, so your existing settings will not be lost, but if you change the settings and save them, they will then be written to the *Application Data* folder instead.

You are now ready to start exporting Google SketchUp models with RubyTMIX.

This page intentionally left blank.

3 Exporting SketchUp Models to Trainz

This chapter describes the basic process by which models are exported from Google SketchUp and subsequently imported into Trainz via Content Manager.

3.1 Important File Location Requirements

RubyTMIX expects that the .skp file for each model you want to process will be in its own individual folder. This is part common sense and part necessity:

- RubyTMIX uses the name of the model's **.skp** file to determine the names of other files that it creates, many of which will go into the same folder where the **.skp** file is located. If there were multiple **.skp** files in the same folder, you would soon find that folder had a large number of different files, all associated with different models, which would quickly become confusing (and inconvenient).
- The files you need to import into Trainz Content Manager always go into a folder named **RubyTMIX_Output,** which is created *within the folder where the current open model's* ***.skp*** *file resides*. This is not unlike the convention that every single asset in Trainz has a file named 'config.txt', regardless of what its actual name is; and it also helps to simplify the process of locating and dealing with the files later.

So: Before you try to process a SketchUp model with RubyTMIX, you must place its .skp file into a folder by itself, and then open it in SketchUp from that folder.

WARNING

FAILURE TO PLACE EACH MODEL'S .skp FILE IN ITS OWN FOLDER PRIOR TO PROCESSING IT WITH RubyTMIX WILL ALMOST CERTAINLY CAUSE YOU TO LOSE WORK RELATED TO OTHER MODELS IN THE SAME FOLDER. This is an important requirement that MUST be adhered to.

3.2 Prerequisites for Exporting a Model

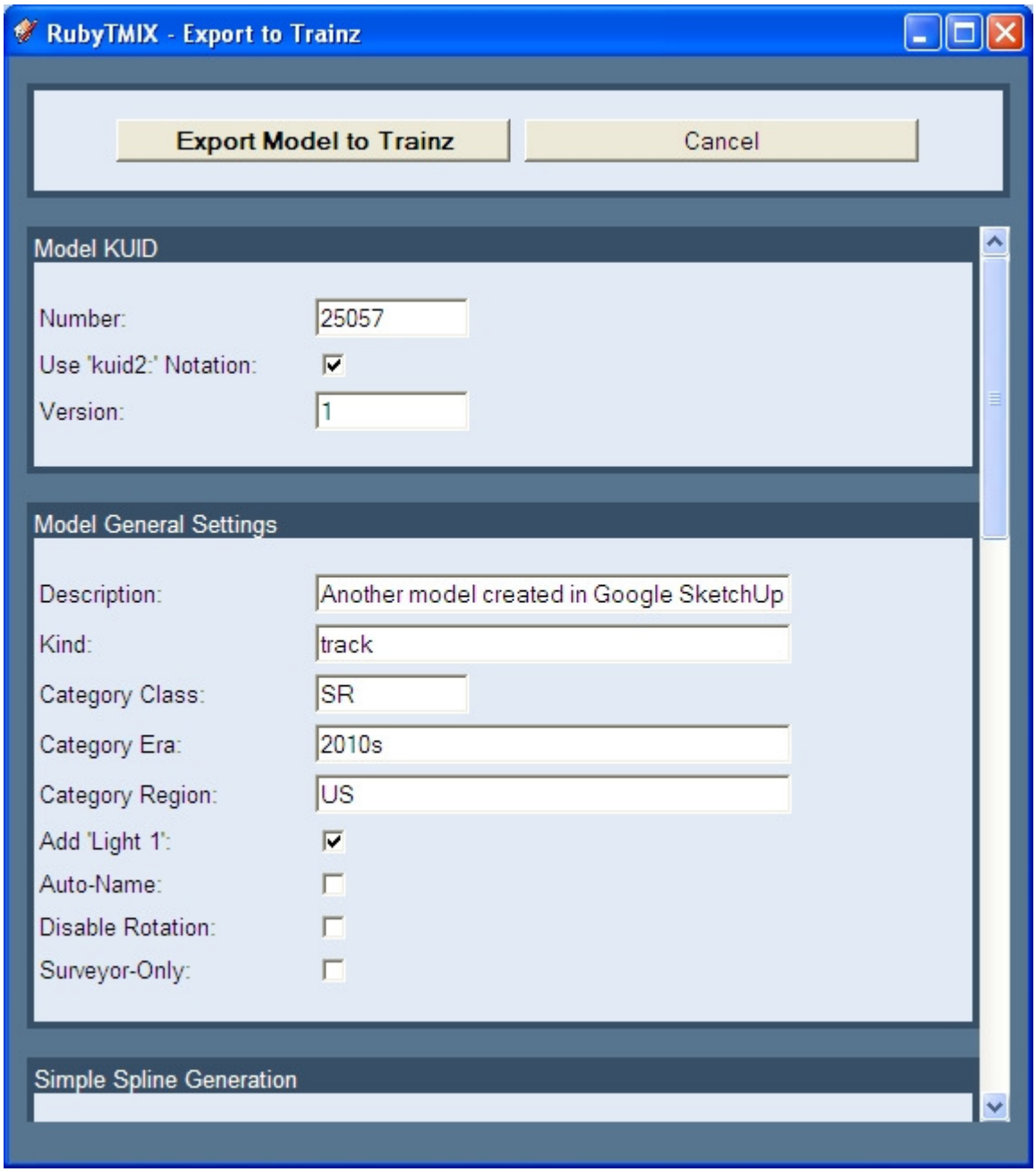
In order to export a model from SketchUp to Trainz using RubyTMIX, you must have already completed the one-time configuration process described in Chapter 2. If you haven't done so, a message a displayed reminding you that this is required, and you will not be able to start the export process until the configuration step has been completed.

You must also have a model open and it must have been saved at least once (so that it has a name); and it must also contain some geometry! If any of these conditions is not met, a message will be displayed and you will not be able to start the export process.

Finally, before you start an export, be sure to set up the view of your model so that it looks the way you want its thumbnail image to appear in Trainz Content Manager.

3.3 Starting the Export Process

Assuming that the aforementioned prerequisites have been met, you begin the export process by selecting *Export to Trainz* from the RubyTMIX menu. When you do, you will see a dialog similar to the example on the next page.



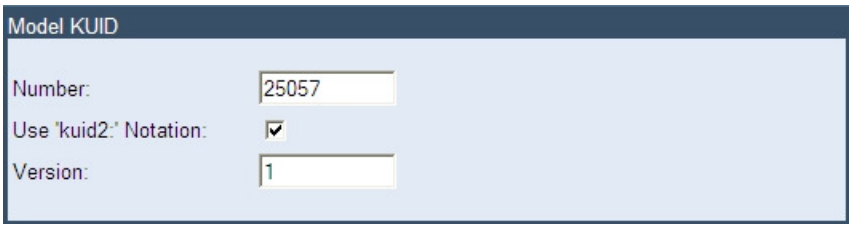
The *Export to Trainz* dialog is divided into two major parts. The top part, which does not scroll, contains two buttons:

- Click Export Model to Trainz to start the actual export process.
- Click Cancel to close this dialog without starting the process.

The bottom part, which scrolls, contains 8 sections where you actually select the options you want to use for the export of the current model. These sections are as follows.

Model KUID

The *Model KUID* section allows you to set the KUID information for the model. If this is the first time this model is being exported, or if you have elected to replace the existing 'config.txt' file on this export run, you can change the KUID settings and the values you supply will be used when the 'config.txt' file is generated for the model.

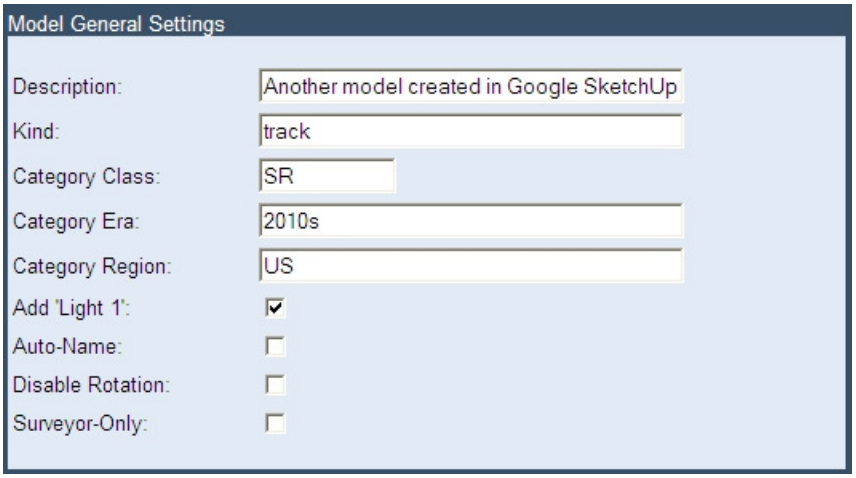


The fields in this section are as follows.

- Number is the unique number that you want to assign to the model. This field is mandatory and will start out with the next available value, which you can replace if desired by typing a new number here.
- Use 'kuid2' Notation determines if generated KUID numbers will be versioned (*kuid2* format) or not (*kuid* format). If this check box is ticked, 'kuid2' notation is used; if it's not ticked, 'kuid' notation is used.
- Version indicates the version number that will be part of the generated KUID, if the *Use 'kuid2' Notation* check box is ticked; in this case the *Version* field is mandatory. By default the version number is '1'.

Model General Settings

The *Model General Settings* section allows you to specify values for several 'config.txt' items that should be quite familiar to any Trainz content creator. As with the KUID values, you can change these settings and the values you supply will be used whenever a 'config.txt' file is generated for the model.



The fields in this section are as follows.

- Description allows you to type a simple description for the model. This capability is limited to a single line of text; if you need a more complex description that spans multiple lines, then you will need to edit the 'config.txt' file directly.
- Kind allows you to specify the type of asset being created. Typically this will be the word "scenery", but you can type any valid Trainz assets kind here.

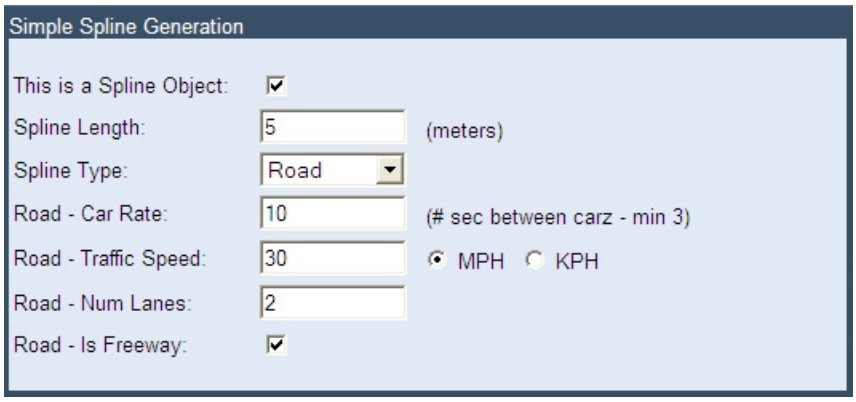
  Note: If you use the Spline options (which will be described shortly), then RubyTMIX will ignore anything you type in this field and will instead substitute a Kind of "track" when it creates the 'config.txt' file.
- Category Class is the 2-letter Trainz standard designation for the type of asset you are creating. Typically this will be letters "BR", but you can type any valid Trainz category-class designation here.

  Note: If you use the Spline options (which will be described shortly), then RubyTMIX will ignore anything you type in this field and will instead substitute the correct category-class designation for the type of spline being created.
- Category Era is one ore more decade identifiers, in the form "2010s" and separated by commas, that specify which general time period the asset is appropriate for.
- Category Region is the 2-letter Trainz standard designation for the area of the world the asset most usually appears in. Examples are "US" for United States, "AU" for Australia, and so on.
- Add 'Light 1' determines if the 'config.txt' file has the line 'light 1' added to it or not. If this check box is ticked, the line is added; if this checkbox is not ticked, the line is not added.

  Note: If you use the Spline options (which will be described shortly), then RubyTMIX will ignore the state of this check box and will not emit the 'Light 1' tag to the 'config.txt' file, as this tag is not valid for spline objects.
- Auto-Name determines if the 'config.txt' file has the line 'autoname 1' added to it or not, which in turn determines if Trainz will or will not automatically assign a name to the asset when it is placed in a route.. If this check box is ticked, the line is added, and auto-naming will take place; if this checkbox is not ticked, the line is not added, and auto-naming will not take place.
- Disable Rotation determines if the 'config.txt' file has the line 'rotation 0' added to it or not, which in turn determines if Trainz will or will not allow the asset to be rotated after it is placed in a route.. If this check box is ticked, the line is added, and rotation of the asset is disabled; if this checkbox is not ticked, the line is not added, and rotation of the asset will be permitted..
- Surveyor-Only determines if the 'config.txt' file has the line 'surveyor-only 1' added to it or not, which in turn determines if Trainz will or will not display the asset in Driver. If this check box is ticked, the line is added, and the asset will not appear in Driver; if this checkbox is not ticked, the line is not added, and the asset will appear in Driver.

Simple Spline Generation

The *Simple Spline Generation* section allows you to export your model with the appropriate 'config.txt' tags that make it a spline object. Only simple (one-piece) splines are supported. To create spline objects with initiators and terminators, additional manual editing of the 'config.txt' file as well as other modeling operations will be required.



The fields in this section are as follows.

- This is a Spline Object is a check box that determines if the object is or is not a Spline. Placing a tick mark in this check box indicates the object is a spline, while leaving the check box unticked indicates the object is not a spline. If the check box is unticked, the other values in this section have no effect.
- Spline Length allows you to specify the length of the spline, in meters. This value will be initially calculated for you, based on the extents of your model.

  Note: If this value comes up as 0.000, it means that RubyTMIX does not think your model represents a valid spline. Because splines must start at the origin and must extend only in the negative Y direction, this usually means that your model contains geometry in the positive Y direction from the origin, or it contains no geometry in the negative Y direction from the origin.
- Spline Type allows you to specify what kind of spline you are creating. It is important that you make the proper selection so that RubyTMIX will write the correct category-class identifier to the 'config.txt' file. The available spline types are Fence, Road, Platform, Structure and Vegetation.

  Note: Splines of type 'Track' are not supported by RubyTMIX due to their complexity. Track splines are best created in other tools, such as GMax.
- Road - Car Rate allows you to specify the minimum interval, in seconds, at which carz will be generated by Trainz when you are creating a Road spline. A value of 0 disables traffic, while values from 3 to 90 specify that number of seconds between carz. Values of 1 and 2 are not valid.
- Road - Traffic Speed allows you to specify the speed of travel of the carz when you are creating a Road spline. Type the value you want in the text field, and then choose MPH or KPH as appropriate.

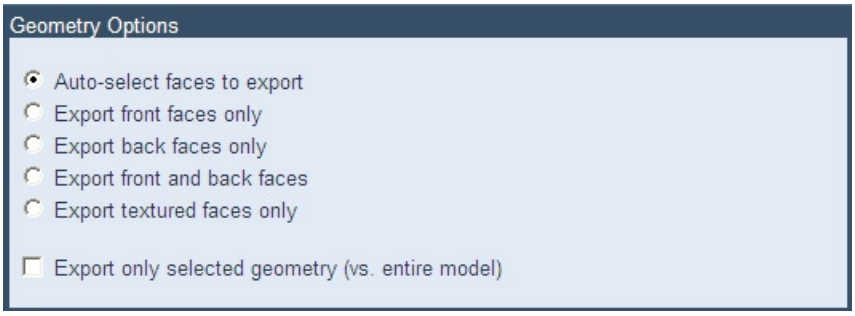
Note: Trainz expects the 'traffic-speed' value in the 'config.txt' file to be expressed as *meters per second*. RubyTMIX will automatically perform the required calculations and emit the proper value to the 'config.txt' file, based on the speed value you entered and your selection of MPH or KPH.

- Road - Num Lanes allows you to specify the number of travel lanes when you are creating a Road spline. Type the value you want in the text field.
- Road - Is Freeway determines if the 'config.txt' file has the line 'isfreeway 1' added to it or not, which in turn determines if carz on a Road spline will all travel in the same direction. If this check box is ticked, the line is added, and carz should all travel in the same direction; if this checkbox is not ticked, the line is not added, and carz will travel in both directions.

Chapter 4 provides a short tutorial for creating spline objects, and you are encouraged to look there for additional information on this topic.

Geometry Options

The *Geometry Options* section allows you to specify how polygons are extracted from the SketchUp model for output to Trainz.



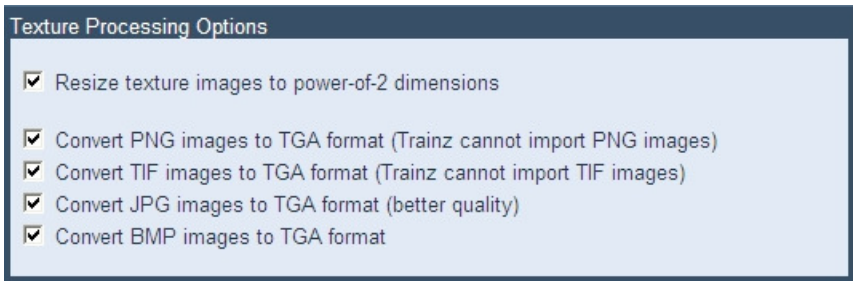
The fields in this section are as follows.

- Auto-select faces to export is the recommended setting for most applications. When this option is ticked, RubyTMIX analyzes the faces in the model and decides which ones to export based on which way they are facing and whether or not they have a material or texture applied to them. The end result is that RubyTMIX (usually) generates only the absolute minimum number of polygons that is required to represent all the visible faces of the model.
- Export front faces only will export all the faces in the model that are identified as 'front', and only those faces, regardless of whether or not they have any materials or textures applied to them.
- Export back faces only will export all the faces in the model that are identified as 'back', and only those faces, regardless of whether or not they have any materials or textures applied to them.

- Export front and back faces will export all the faces in the model, regardless of whether or not they have any materials or textures applied to them. Needless to say, this will result in the highest number of polygons being generated.
- Export textured faces only will export all the faces in the model that have a material or texture applied to them, regardless of their front/back orientation.
- The Export only selected geometry check box can determine whether or not the entire model is exported. If it is not ticked, or if it is ticked and no part of the model is selected, then the entire model is exported. If it is ticked and some part of the model is selected, then only the selected geometry is exported.

Texture Processing Options

The *Texture Processing Options* section provides you with a great deal of control over how the various textures in your model are processed. For a new model, all of these check boxes are ticked by default.



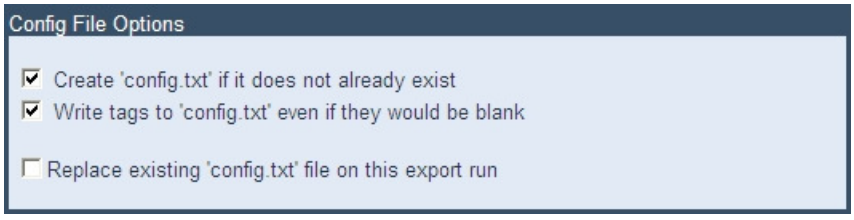
The fields in this section are as follows.

- Resize texture images to closest-power-of-2 determines if texture images are resized after being exported. Unlike Trainz, SketchUp has no strict requirement that image dimensions must be powers-of-2, and so the images that are exported may not be correctly dimensioned for Trainz. If ticked, this item causes RubyTMIX to check the dimensions of each image and to automatically resize it as needed to meet the Trainz requirements; it also respects the 1:8 maximum ratio. If not ticked, then images are not resized even if this would result in them being incompatible with Trainz; you must then resize them yourself prior to attempting to import the model into CM. This can have advantages if you want to resize the images differently than RubyTMIX does.
- Convert PNG images to TGA format and Convert TIF images to TGA format determine how SketchUp textures based on PNG, TIF and TIFF files are handled. These file types cannot be used by Trainz; when either item is ticked, its corresponding images will be converted to 32-bit TGA with alpha (transparency) information maintained. If either item is not ticked, its corresponding images are left as-is and you must convert them to TGA yourself prior to importing the model into CM. Regardless of these settings, images based on PNG and TIF/TIFF files are always given a .TGA extension and always have 'alphasource' set to 'true' in their Trainz Mesh Importer material definition.

- Convert JPG images to TGA format determines how SketchUp textures which are based on JPG or JPEG files are handled. Such textures are lossy and are not recommended for use with Trainz; when this item is ticked, all such files will be converted to 24-bit TGA. If this item is not ticked, these images are used as-is; they will still work with Trainz, but are not considered optimal. Regardless of the setting, JPG images always have 'alphasource' set to 'false'.
- Convert BMP images to TGA format determines how SketchUp textures which are based on BMP files are handled; when this item is ticked, all such files will be converted to 24-bit TGA. If this item is not ticked, these images are used as-is, and should work with Trainz with no issues. Regardless of the setting, BMP images always have 'alphasource' set to 'false'.

Config File Options

The *Config File Options* section allows you to control what happens in regard to the "config.txt" file that is generated for your model.

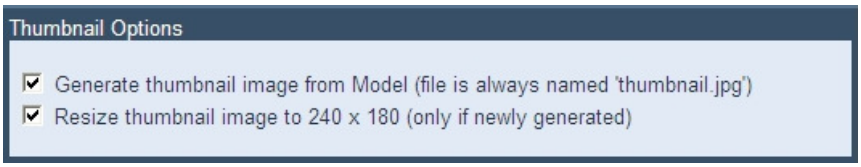


The fields in this section are as follows.

- Create 'config.txt' if it does not already exist determines if a 'config.txt' file is created for models when they are exported. If ticked, the file is created; if not ticked, the file is not created. Once a 'config.txt' file has been created, RubyTMIX never touches it again (unless you tell it to), so any changes you make to it yourself after that point won't get wiped out.
- Write tags to 'config.txt' even if they would be blank determines how certain tags are written to the 'config.txt' file if you haven't specified values for them. The tags that are affected are *Organisation*; *Contact E-Mail*; *Contact Website*; and *License*, all of which are defined via the one-time configuration described in Chapter 2. If this check box is ticked, these 4 tags are written to the 'config.txt' file even if they contain no values, which results in double quotes being emitted for each; if this check box is not ticked, then any of these 4 fields that do not contain values are not written to the 'config.txt' file at all.
- Replace existing 'config.txt' file on this export run can be used to either retain any existing 'config.txt' file for the current model (not ticked) or to erase the existing file and create a new one (ticked). This check box resets to **not ticked** every time you start an export, so if you forget about it the worst that will happen is that you won't lose any existing 'config.txt' file.

Thumbnail Options

The *Thumbnail Options* section allows you to control what happens in regard to the thumbnail image that is generated for your model.

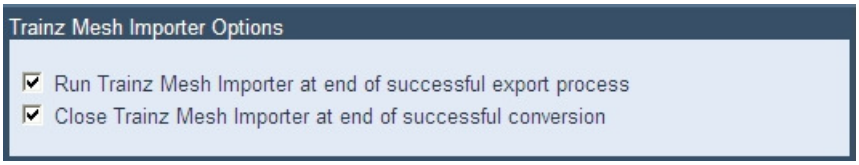


The fields in this section are as follows.

- Generate thumbnail image from Model determines if a Content Manager thumbnail is generated during export; when this is item is ticked, the view of the model at the time the export started is used to create a thumbnail for Content Manager. If this item is not ticked, no thumbnail is created; but the "config.txt" file will still contain a reference to one, so you will need to supply a thumbnail yourself.
- Resize thumbnail image to 240 x 180 determines if a newly-generated thumbnail image is resized to the 240 x 180 pixel dimensions required by Content Manager; if ticked, the image is resized. If not ticked, the image is not resized and you must resize it yourself prior to attempting to import the model into Content Manager or a warning will result.

Trainz Mesh Importer Options

The *Trainz Mesh Importer Options* section provides you with some control over how the Trainz Mesh Importer utility is invoked at the end of the export process.



The fields in this section are as follows.

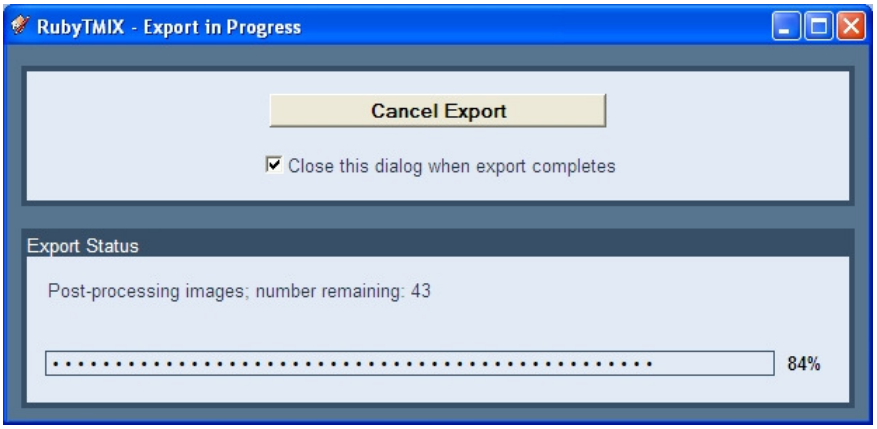
- Run Trainz Mesh Importer determines if the Trainz Mesh Importer utility is automatically invoked at the end of a successful export. If ticked, the utility runs and will attempt to create an **IM** file based on the contents of a TrainzXML file created by RubyTMIX. If not ticked, the Mesh Importer is not launched automatically, meaning you must run it yourself in order to create the **IM** file. Regardless of the setting of this item, a **CMD** file is always created that will invoke the Mesh Importer for you when double-clicked.
- Close Trainz Mesh Importer determines how the command line for the Mesh Importer is constructed. If this item is ticked, then the Mesh Importer will be set to terminate automatically after running if no errors occurred. If this item is not ticked, the Mesh Importer will be set to remain open until a keyboard key is pressed. The Mesh Importer command line is always constructed to keep the display open if a conversion error occurs.

When You Have Made All Your Selections

Once you have all of the options set according to your preferences, click *Export Model to Trainz* to start the actual export process. **All of these settings are remembered individually for every model and will be recalled if the same model is exported again.**

3.4 Monitoring the Export Process

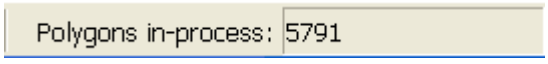
While an export operation is in progress, a status display appears that looks similar to the example below.



As you can see, this is a very simple dialog that contains:

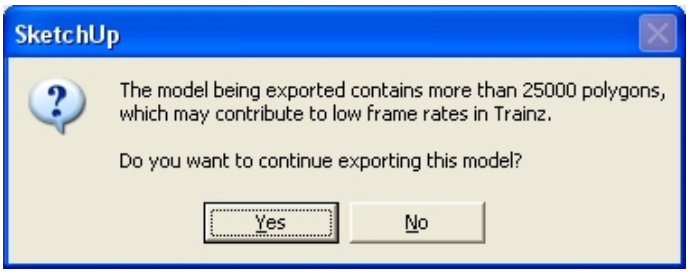
- A 'Cancel Export' button that you can click to abort the export process if desired. If you do so, please be patient; RubyTMIX must finish its current task before it can actually stop the process.
- A 'Close this dialog when export completes' check box that, if **not** ticked, will cause the *Export In Progress* dialog to stay open after the export completes.
- Export Status, which displays a series of message describing the current task within the export process; a progress bar (of sorts); and a percentage-of-completion value.

RubyTMIX also displays information in the SketchUp *status bar* and *value control box* (VCB) to indicate what's going on when it's involved in a lengthy operation that might cause the *Export Status* area to show the same display for a relatively long time. In particular, it displays a *Polygons in-process* count in the VCB, similar to the example below, that gives you a very real indication of how much work is going on:



The polygons in your model are counted during an early part of the export process. As the count of polygons increases during this time, it is displayed in the VCB. Later, as the actual geometry calculations for these polygons are completed, the count winds back down to show you how much work is left.

If the number of polygons found exceeds 25,000, RubyTMIX also issues a warning message, similar to the example shown at the top of the next page; this simply cautions you that exporting models which are extremely complex can result in poor performance in Trainz, which may include significant lagging when the model is in view.



To continue exporting the model, click **Yes**; this choice is then remembered for the current model and this warning will not be repeated if you export the model again. If you click **No**, the export process is aborted.

Here are the steps RubyTMIX goes through as it exports your model.

- Check for the content output folder (**RubyTMIX_Output**) and create it if needed. If it exists, delete all the files it contains (except 'config.txt', unless it is to be overwritten on this run) to ensure a clean import into CM later.

  Note: Starting with RubyTMIX 1.2, it is allowable for other folders to exist within the *RubyTMIX_Output* folder (i.e. if you have added them yourself after the initial export). Any such folders are skipped over and their contents are not affected by the clearing process.

- Check for the materials output folder (**Material_Texture_Images**) and create it if needed. If it exists, delete all the files it contains to ensure that only the textures actually used in the current model are presented for later use.

  Note: Starting with RubyTMIX 1.2, it is allowable for other folders to exist within the *Material_Texture_Images* folder (i.e. if you have added them yourself after the initial export). Any such folders are skipped over and their contents are not affected by the clearing process.

- Pre-process the model to identify all the nodes in the model hierarchy, and save references to their lists of entities and transforms.

- Select the faces to process based on the Geometry options. Accumulate textures for these faces and count the number of polygons to be exported. Display the number of textures and polygons found in the SketchUp status bar and VCB.

- Write the original texture image files to the **Material_Texture_Images** folder; *note that this is an internal SketchUp operation that must be done all-at-once, and may therefore cause the exporter to appear to pause for several seconds.*

- Create a cross-reference file in TXT format (editable with Windows Notepad) that shows the association of the real texture names with the sequential ones that will be created later.

- Scan the images in the folder and tabulate their actual X-Y dimensions for later use in image resizing.
- Create the TrainzXML file, write its header, and start the **<mesh>** section.
- Calculate vertex, normal and UV coordinates based on model geometry and component or group transforms. Attempt to auto-detect mirrored geometry and projected textures and adjust data as needed to prevent 'inside-out' geometry in the final exported model. Create sequential 'controlled' names for all textures that are guaranteed to be acceptable to Trainz Content Manager.
- Write the **<materials>** section of the TrainzXML file, changing file extensions to account for PNG-to-TGA, TIF-to-TGA, JPG-to-TGA and BMP-to-TGA conversions that will happen later, and then close the TrainzXML file.
- Create the 'config.txt' file if it does not already exist (and if set to do so), using the values supplied in the main RubyTMIX configuration dialog as well as the start-of-export dialog.
- Perform image post-processing via command-line calls to **rtmipp.exe**. This is where textures are resized to nearest-power-of-2 dimensions and also where the actual format changes from PNG, TIF, JPG or BMP to TGA are performed.

  Note: 'rtmipp.exe' is the RubyTMIX Image Post-Processor, which is included as part of the RubyTMIX installation.
- Create a JPG thumbnail for the model and resize it to 240 x 180, again via **rtmipp.exe**.
- Run the Trainz Mesh Importer to process the TrainzXML file. Also create a **CMD** file that will allow the Mesh Importer to be run again manually if needed, and creator an additional CMD file that will allow the model to be easily viewed in PEV's *Trainz Mesh Viewer* program.

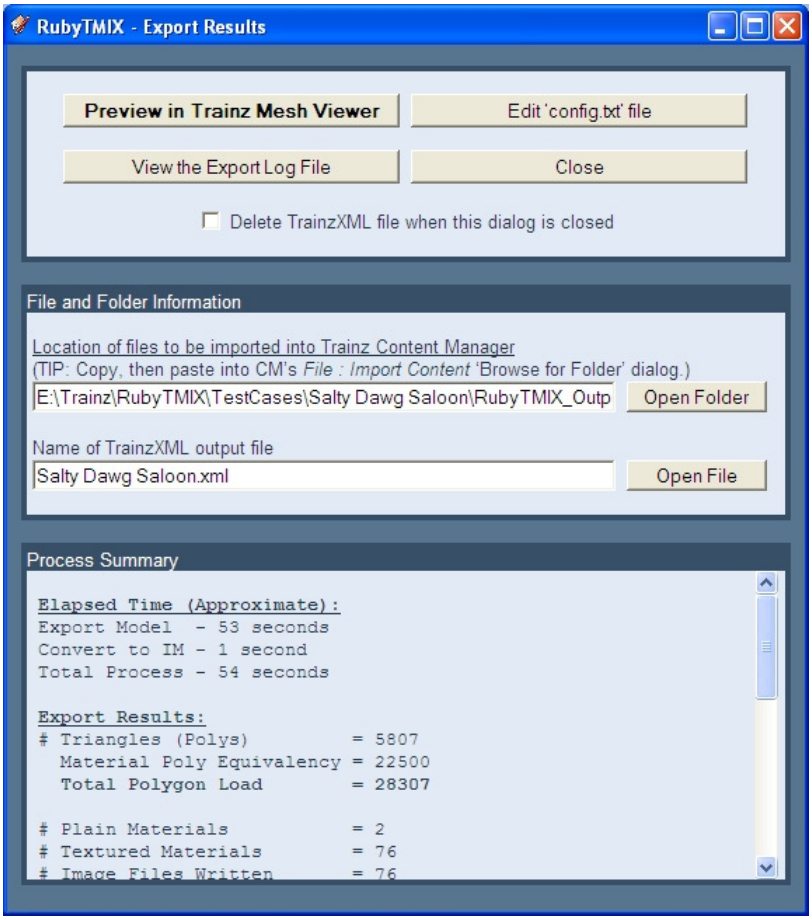
  Note: The Trainz Mesh Importer is included in the RubyTMIX installation.
- Close the *Export In Progress* dialog and display the *Export Results* dialog.

I strongly recommend that you leave your PC alone and do not try to do other things while the export process is underway. Taking the focus away from SketchUp while it's running lengthy Ruby script operations (that's *any* Ruby script, not just RubyTMIX) is known to cause lock-ups and/or to cause Windows to think that SketchUp is "not responding" – which in turn may result in you losing some or all of your work.

3.5 Assessing and Previewing the Export Results

When the export operation completes, the *Export Results* dialog is displayed, and will appear similar to the example shown at the top of the next page.

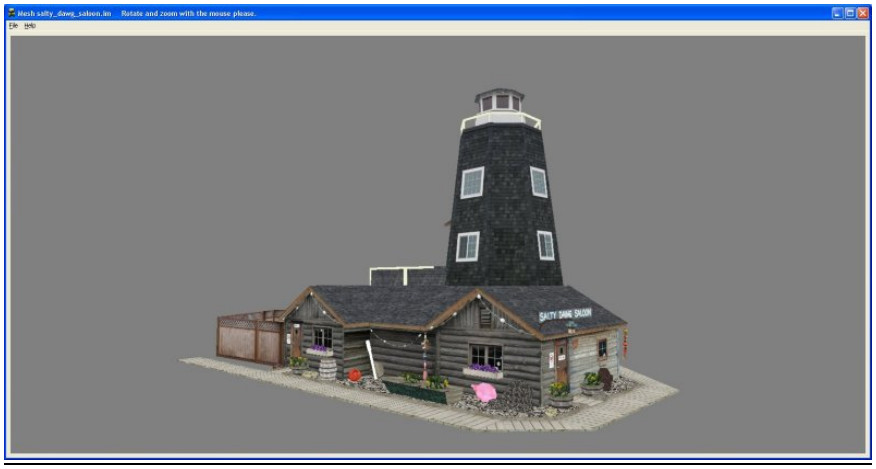
This dialog not only summarizes the results of the export process, but also provides multiple options that allow you to preview, tweak and otherwise manipulate the exported files before you move on to importing the model into Trainz via Content Manager.



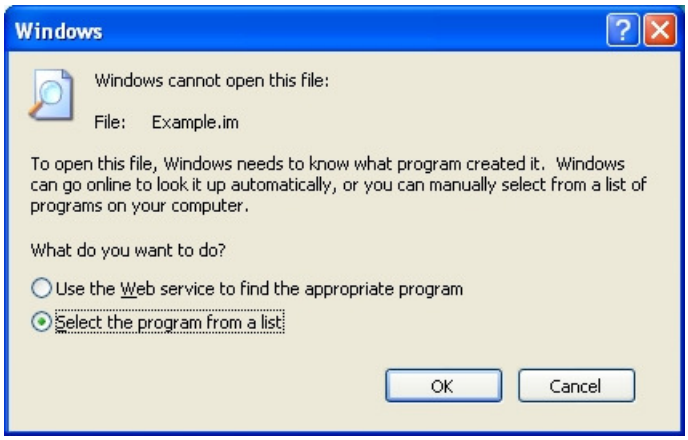
The *Export Results* display consists of 3 main sections, which are as follows.

1. At the top are four buttons and a check box.

    - Preview in Trainz Mesh Viewer allows you to use PEV's Trainz Mesh Viewer to preview the model immediately and get some idea of what it will look like in Trainz – *before* you actually import it. In the case of the model being used as an example here, this results in a display similar to the image below.



If this is the first time you have tried to preview a model, you may instead see a message from Windows stating that it doesn't know how to open the file.



This is no great cause for alarm – it simply indicates that the **IM** file extension has not yet been associated with the Trainz Mesh Viewer program.

This is easy to fix. Tick the option *Select the program from a list* and then click *OK*. Windows will display an *Open With* dialog. In that dialog, click *Browse*, and navigate to the location where Trainz Mesh Viewer is installed.

In Windows XP systems, this will usually be:
C:\Program Files\PEVSoft\Mesh Viewer2

In Windows 7 (64-bit) systems this will usually be:
C:\Program Files (x86)\PEVSoft\Mesh Viewer2.

Once you find the file **Mesh_Viewer.exe**, select it and click *Open*; this will return you to the *Open With* dialog. Tick the check box *Always use the selected program to open this kind of file* and then click *OK*. Trainz Mesh Viewer should then open and display the preview of your model as shown above.

- Edit 'config.txt' file allows you to open the model's 'config.txt' file and view its contents (or alter them if desired). Typically this file opens in Windows Notepad.
- View the export log file allows you to open the log file that was created during the export process; this can aid in troubleshooting problems with the exporter. Typically this file opens in Windows Notepad as well.
- Delete TrainzXML file when this dialog is closed will, as its name implies, delete the intermediate XML file that was used as the input to the Trainz Mesh Importer. To retain this file, untick this checkbox; to delete the file automatically, tick this checkbox. **Note that you must retain this file (check box not ticked) if you have chosen to perform any of the post-processing steps yourself,** since you will need to run the Mesh Importer manually to generate the **IM** file for Trainz after completing your own post-processing work.
- Close removes this dialog from the screen.

2. The *File and Folder Information* section shows you where the output of the export process is located, and allows you to view the contents of that folder. It also allows you to view the TrainzXML file itself if you like.

    - Location of files to be imported into Trainz Content Manager shows you where all the files are located that need to be imported into CM to bring this model into Trainz. They are all in one folder, which makes it very easy to import the model; if you click *Open Folder*, that folder will be displayed in Windows Explorer.

      You'll also notice the dialog suggests a TIP, which is to copy the displayed text to the Windows clipboard so that you can later paste it into CM's *Browse for Folder* dialog. To do this, right-click the displayed text and choose *Select All* from the pop-up menu; then right-click the selected text and choose *Copy* from the pop-up menu. I wish you didn't have to do this manually, but there is currently no functionality within the SketchUp Ruby interpreter that provides access to the Windows Clipboard (or I would do it for you).

    - Name of TrainzXML output file shows you the name of the intermediate XML file that was created to provide input to the Trainz Mesh Importer. If you'd like to view this file, click *Open File.* However, unless you have changed the association for XML files, this will usually result in the file opening in Internet Explorer – if you're lucky. The file can be very large and IE might actually crash instead. If you want to view these files, it's much better to obtain Notepad++ as I mentioned earlier, and then change the association for XML files so that they open with this application instead.

3. The *Process Summary* section shows you some general information about what was exported and what actions were taken. It includes:

    - Elapsed Time, which shows you approximately how long the process took.
    - Export Results, which lists the quantities of certain important items that were involved in the export process. Included in this is the number of polygons (triangles); the number and types of materials; the number of image files written; and the number of image file type conversions performed.
    - Export Selections provides a simple summary of the selections you made at the start of the export process, as a sort of reminder. Scroll the bottom section of the dialog down to see the entire list.

After you're done with this dialog, click *Close*, and then exit from Google SketchUp.

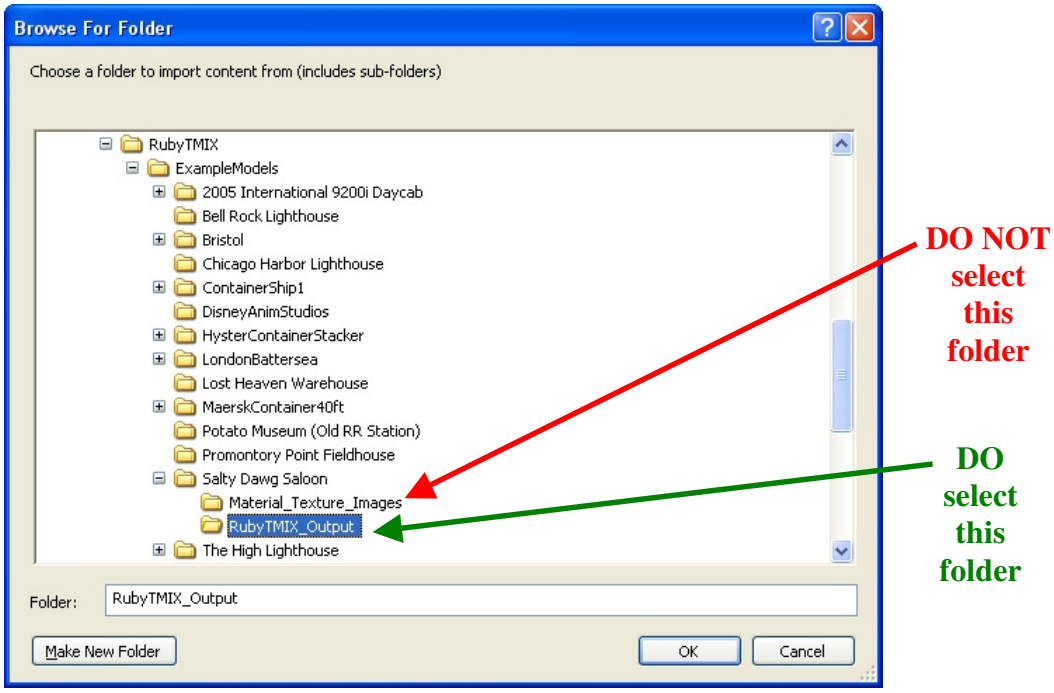
Assuming there were no issues with the export process and that you did not elect to perform certain aspects of the process yourself, you are now ready to import the model into Trainz Content Manager.

3.6 Importing the Model into Trainz Content Manager

Importing the model into Trainz Content Manager is straight-forward and operates in the same way as the process for any other content.

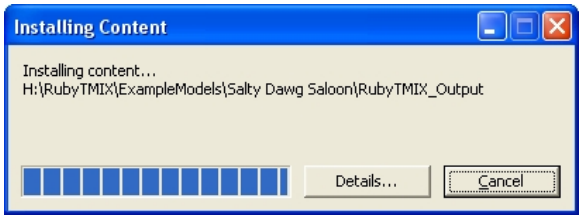
1. Select **File** → **Import Content**. In the *Browse for Folder* dialog that appears, navigate to the folder that was created by RubyTMIX and which contains all the data for your model. This will be located within the same folder as your model's **.skp** file, and will always be named **RubyTMIX_Output**.

   An example of selecting this folder is shown below. Note that you must be careful to select the proper folder – which is *not* **Material_Texture_Images**.

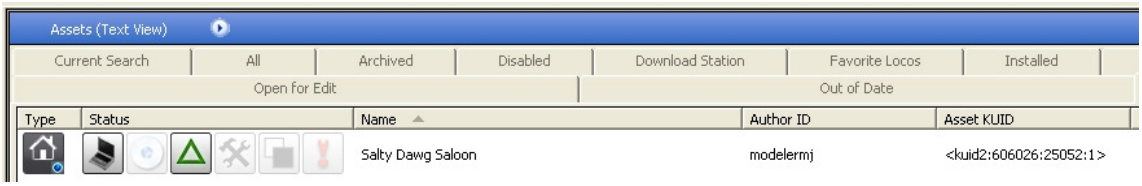


   If you copied the location text from the RubyTMIX *Export Results* dialog (as suggested by the **TIP** there), you can paste it into the *Browse for Folder* dialog's *Folder* field. To do so, simply click in the *Folder* field and type **Ctrl+V**; or right-click in the *Folder* field and choose *Paste* from the context menu that appears.

2. Click *OK*, and the import process should start. Depending on the complexity of your model, this may take some time; you will usually see a display similar to the following example.



3. After import completes – assuming you have your search options set properly in Content Manager – you should see the new model pop up in the display (if you don't see it, try selecting the **Today** tab, or refresh the display). If you used all of the automatic options of RubyTMIX, the model will normally be error- and warning-free, although it will of course show as 'modified'.



If the model has errors, it may mean that you elected to perform some of the operations yourself (like resizing images) and missed something; right-click the entry and check the *Errors & Warnings* report to see what's wrong.

Models downloaded from the Google 3D Warehouse may not always work. I have found a few examples where texture sizes are reported by SketchUp as having dimensions that are powers of 2 when in fact they are not, and so RubyTMIX doesn't know it needs to resize them. In this case you will get errors in CM and will need to resize these images manually.

4. If you select the new model in the list, you should also see its thumbnail and description in the *Asset Details* area of Content Manager (assuming it is displayed). In the case of the example model being used here, this looks like the example below – the image you see was automatically created by RubyTMIX and resized to CM's requirements of 240 x 180 pixels.

3.7 Process Complete

The model is now an asset in Trainz. If you launch Surveyor, you will find it on the *Objects* tab as a static scenery item and you can place it and manipulate it just as you would any other static scenery object.

This page intentionally left blank.

4 Making Simple Spline Objects

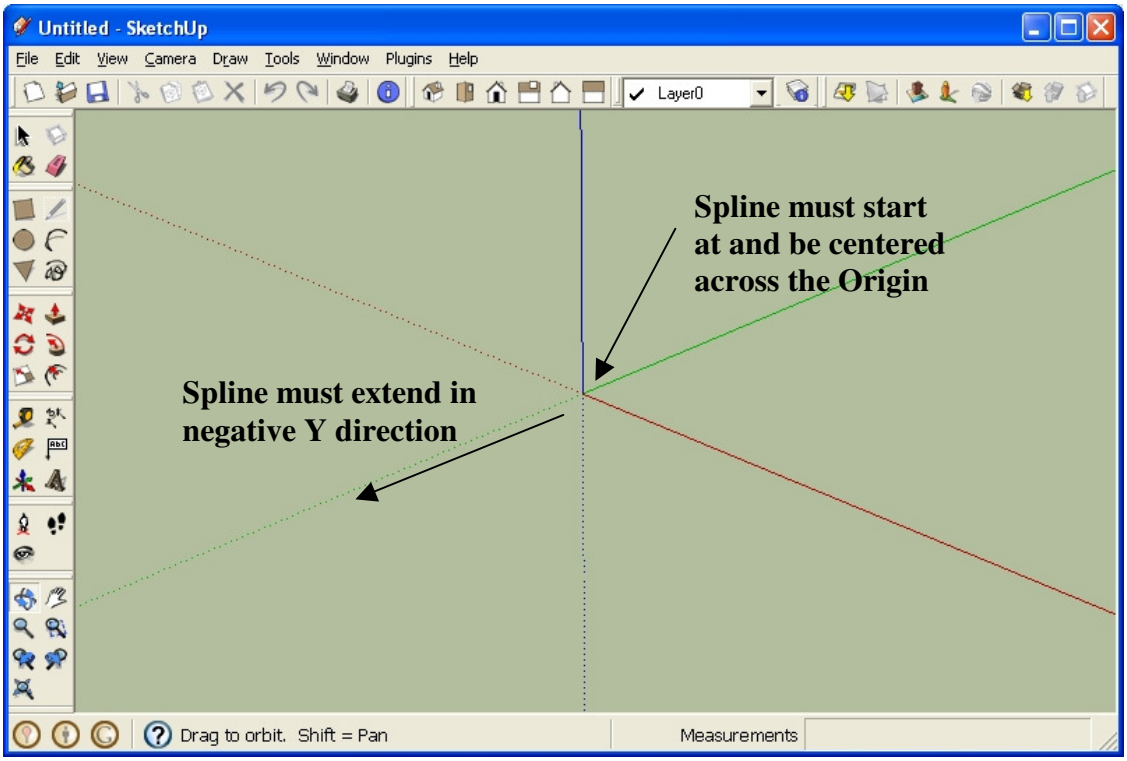
This chapter describes process by which simple (one-piece) spline-type objects can be created in SketchUp and exported to Trainz with RubyTMIX.

In terms of the capabilities of RubyTMIX, there are two basic types of spline objects: Simple and Complex. Simple splines are things like roads, fences and so on, which have only one part that repeats. Complex objects, on the other hand, have extra pieces called *initiators* and *terminators* that appear only at the ends of the main spline.

RubyTMIX only supports the direct generation of simple spline objects. It is certainly possible to make complex spline objects in SketchUp, and to export them via RubyTMIX, but in order to get them into Trainz some extra manual effort on your part will be required. Therefore, only simple splines are discussed in this chapter.

4.1 Spline Basics

Splines must always start at the SketchUp origin; must always extend in the negative Y (green) direction; and must also be centered on the Origin in the X (red) direction.



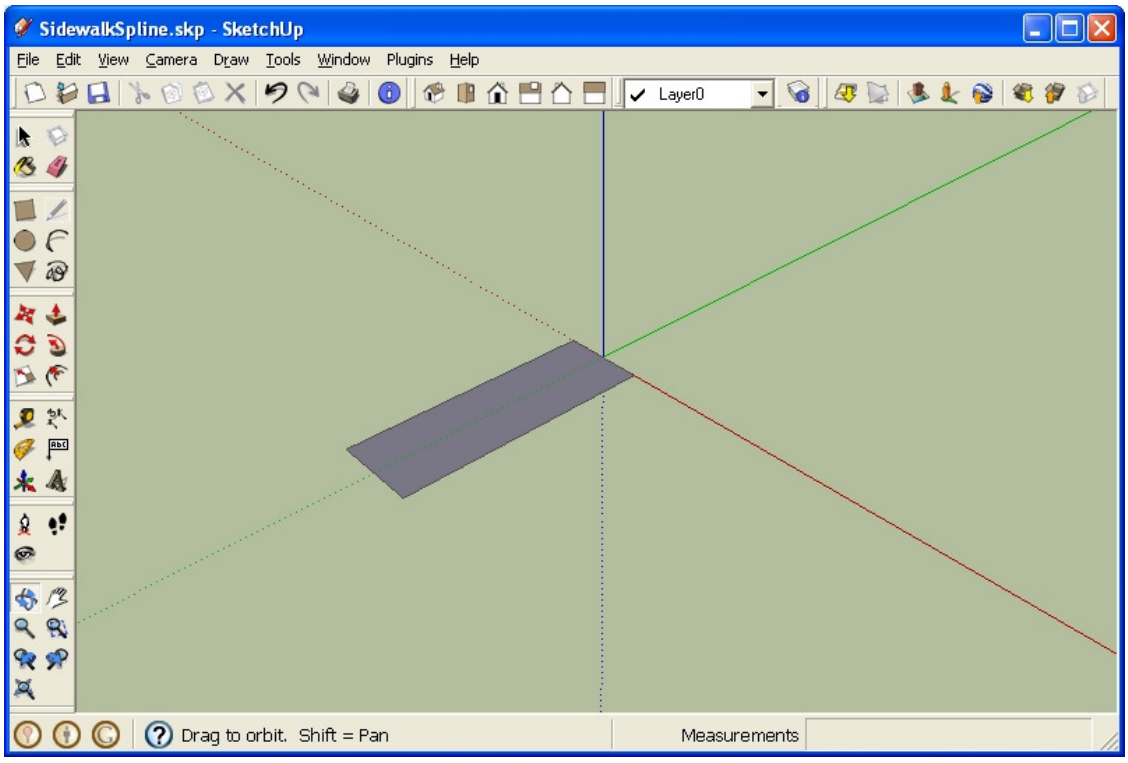
As long as you follow these rules, you should be able to easily create any kind of simple spline you need.

4.2 A Simple Scenery Spline

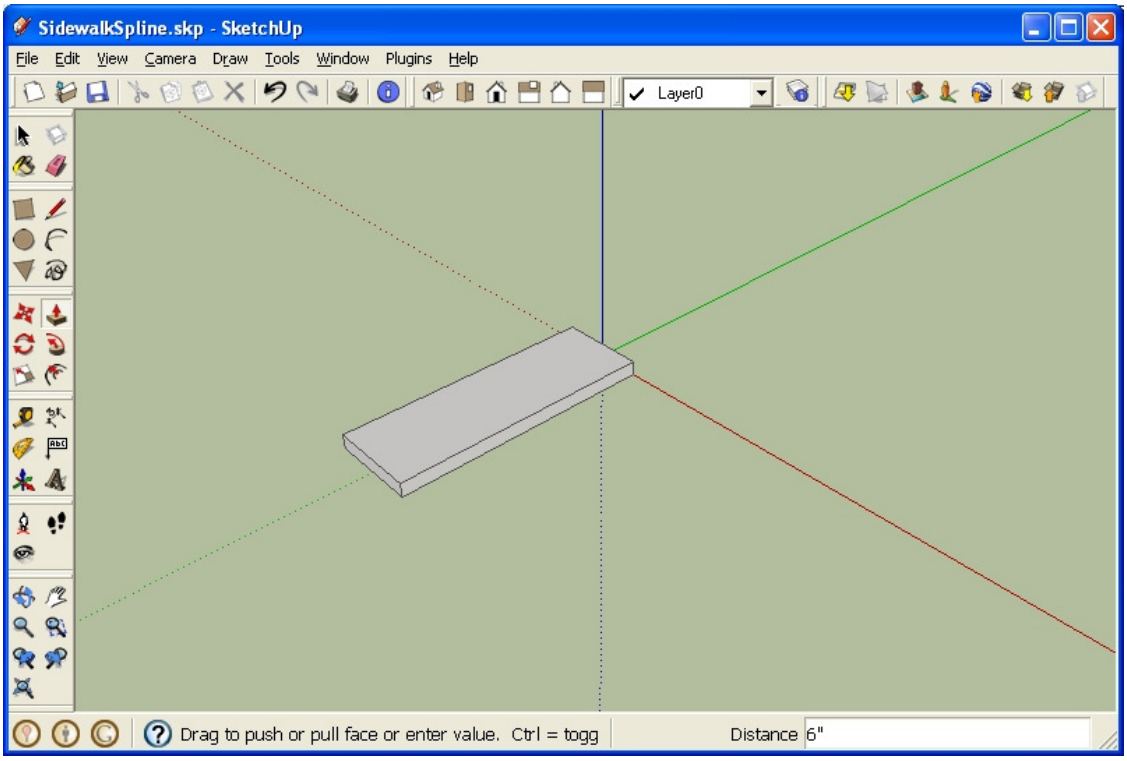
The following steps will take you through the process of creating a sidewalk spline with very basic geometry. The purpose of this is to demonstrate the common techniques that are involved in the creation of any kind of spline. Our sidewalk will be 3 feet wide and 9 feet long.

- Start a new model in SketchUp and save it in its own folder.
- Use the Line tool to draw a line that starts at the Origin and extends 1.5 feet in the negative X (red) direction..
- From the end of that line, draw another line that extends 9.0 feet along the negative Y (green) axis.
- From the end of that line, draw another line that extends 3.0 feet in the positive X (red) direction.
- From the end of that line, draw another line that extends 9.0 feet in the positive Y (green) direction.
- From the end of that line, draw another line (the last one) that extends back to the Origin point.

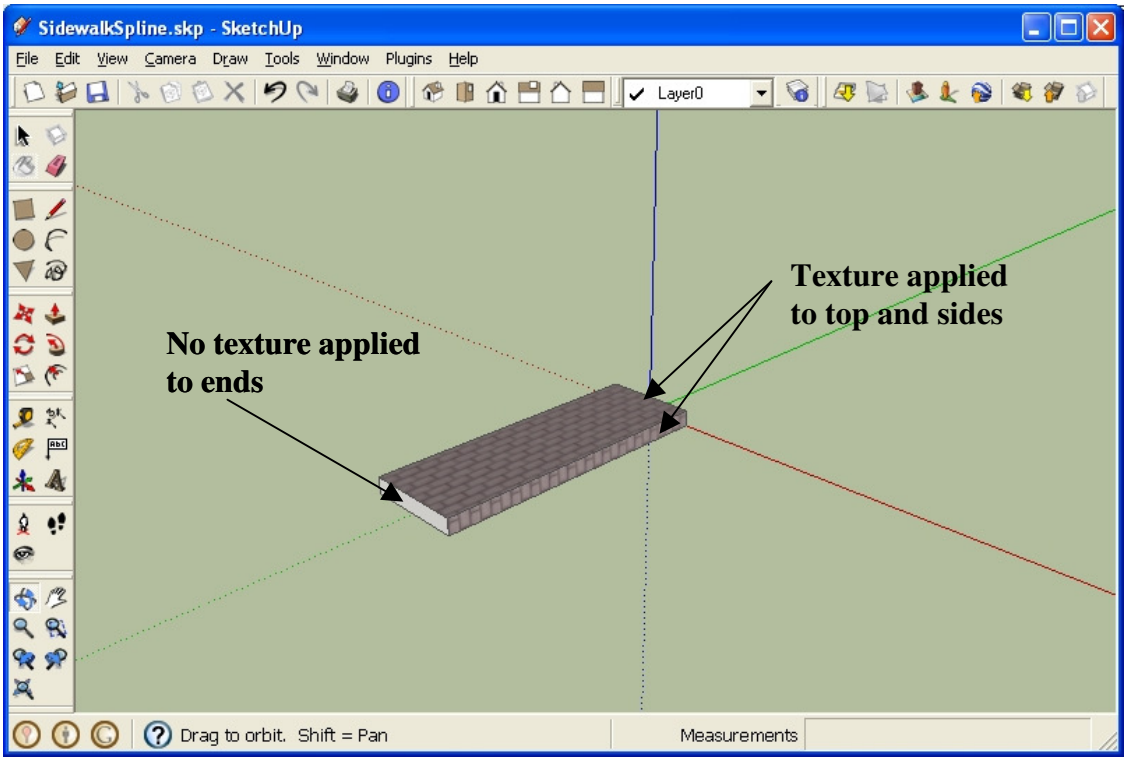
When you draw this last line, SketchUp will fill the rectangle you have created, and the result should look similar to the example below.



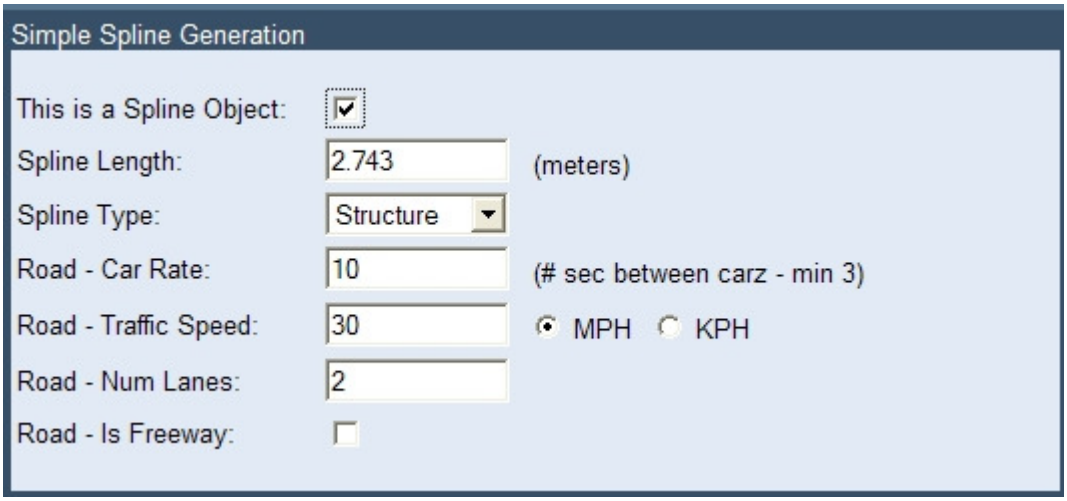
Using the Push-Pull tool, raise the surface of the rectangle 6 inches. This should give you a result similar to the following example.



Apply a suitable texture to the top and sides. Do not apply any texture to the ends.



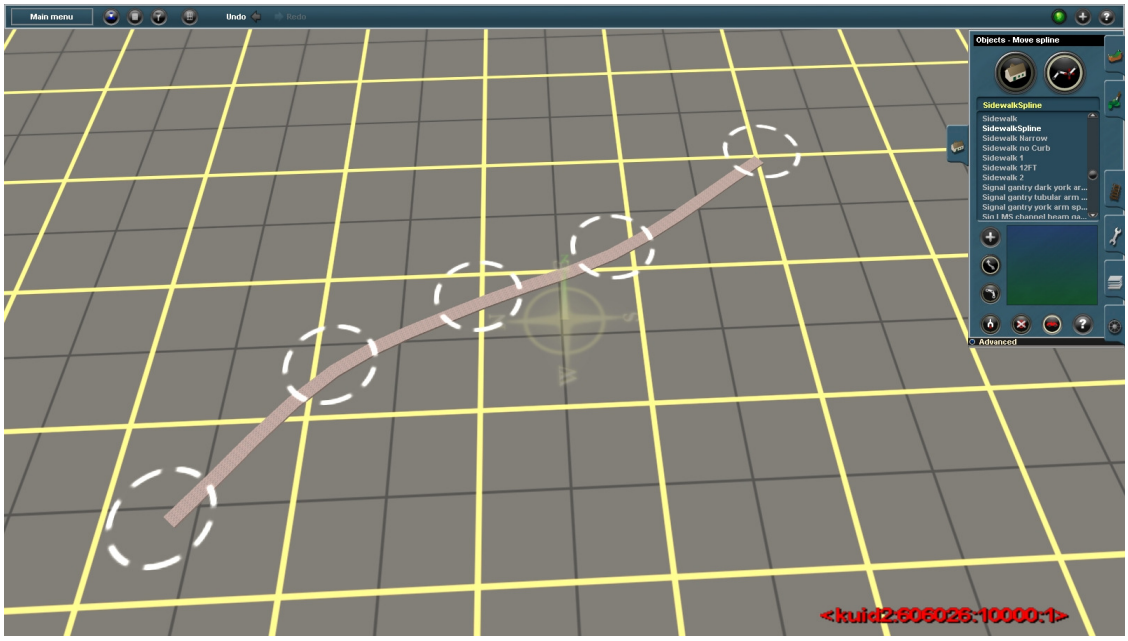
Save the model, then select the menu item at **Plugins ► RubyTMIX ► Export to Trainz**. This will cause the RubyTMIX *Export to Trainz* dialog to display, and the *Simple Spline Generation* section should look similar to the example below.



Note that the length of the spline has been calculated for you, and is already displayed in the *Spline Length* field. The default *Spline Type* is "Structure" so we can leave that as-is for now.

To generate this spline object, all you need to do is place a tick mark in the *This is a Spline Object* check box; fill in any other export information (kuid number, description, etc.) and then click *Export to Trainz*.

Once the model has been exported, import it into Content Manager as described in Chapter 3; it should appear error- and warning-free, ready to use. Then start Trainz, go into Surveyor, and place the object; naturally enough, you'll find it on the *Object* tab as a spline. Here's a screen capture of the sidewalk spline we just created.

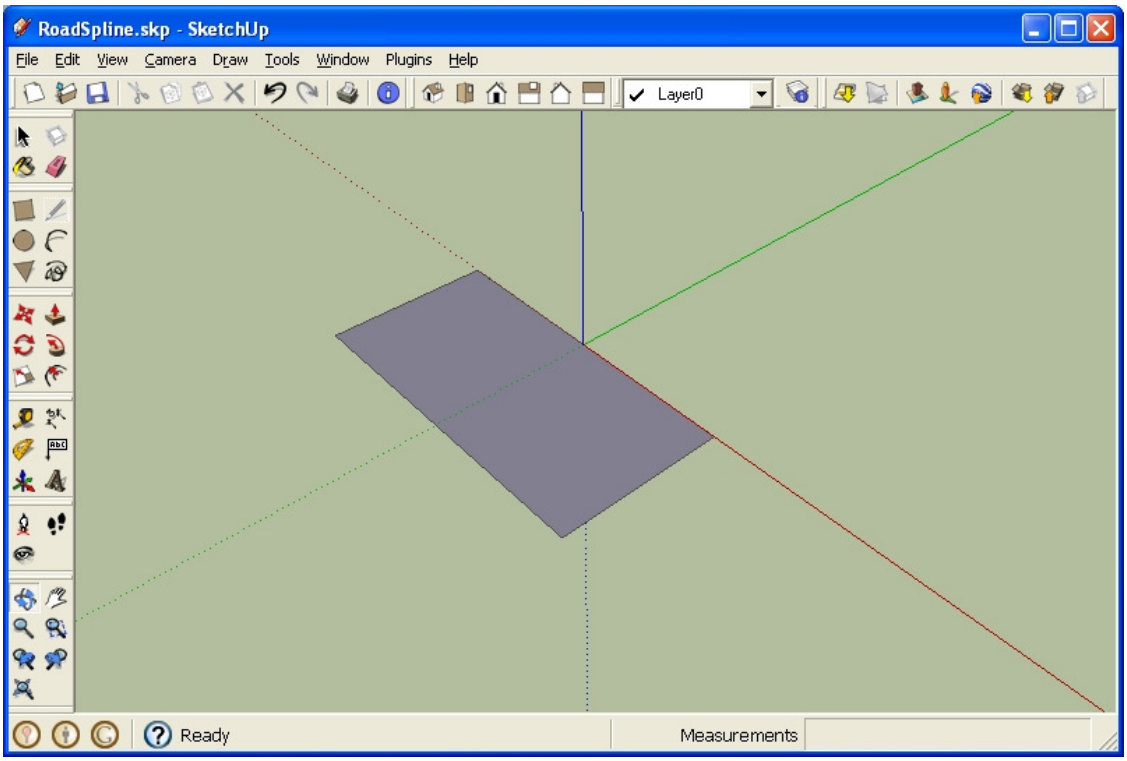


4.3 A Simple Road Spline

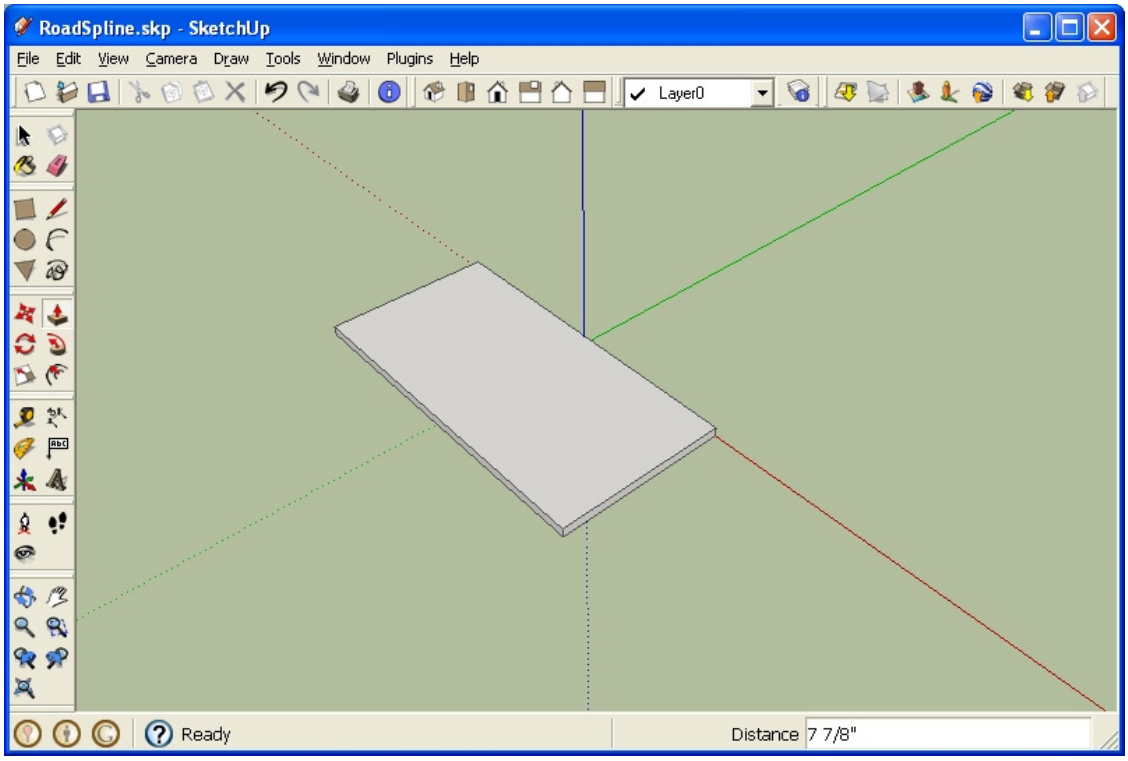
The following steps will take you through the process of creating a simple road spline with light 2-way traffic traveling at 30 miles per hour.

- Start a new model in SketchUp and save it in its own folder.
- Use the Line tool to draw a line that starts at the Origin and extends 12 feet in the negative X (red) direction..
- From the end of that line, draw another line that extends 12 feet along the negative Y (green) axis.
- From the end of that line, draw another line that extends 24 feet in the positive X (red) direction.
- From the end of that line, draw another line that extends 12 feet in the positive Y (green) direction.
- From the end of that line, draw another line (the last one) that extends back to the Origin point.

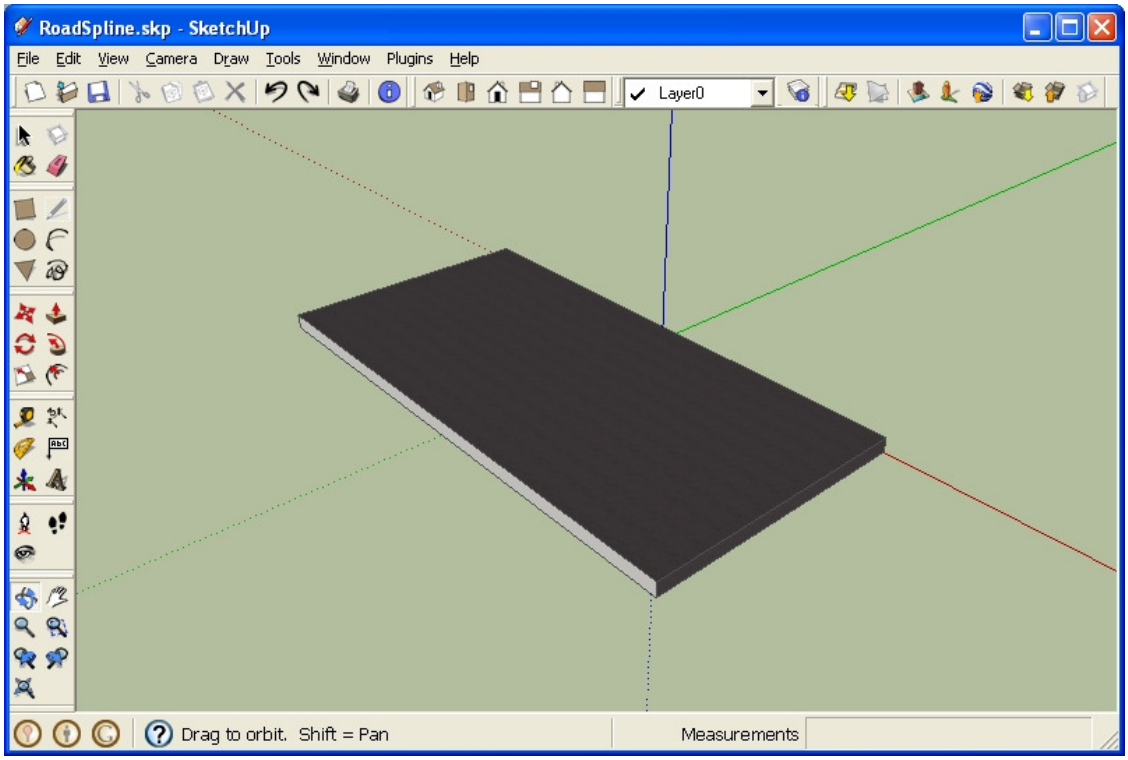
When you draw this last line, SketchUp will fill the rectangle you have created, and the result should look similar to the example below.



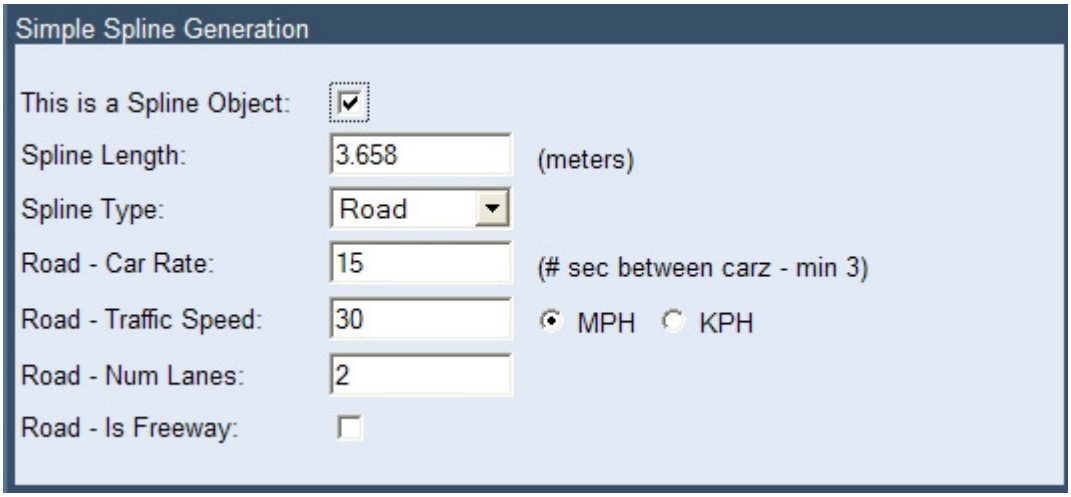
Using the Push-Pull tool, raise the surface of the rectangle 0.2 meters. This should give you a result similar to the following example.



Apply a suitable texture to the top and sides. Do not apply any texture to the ends.



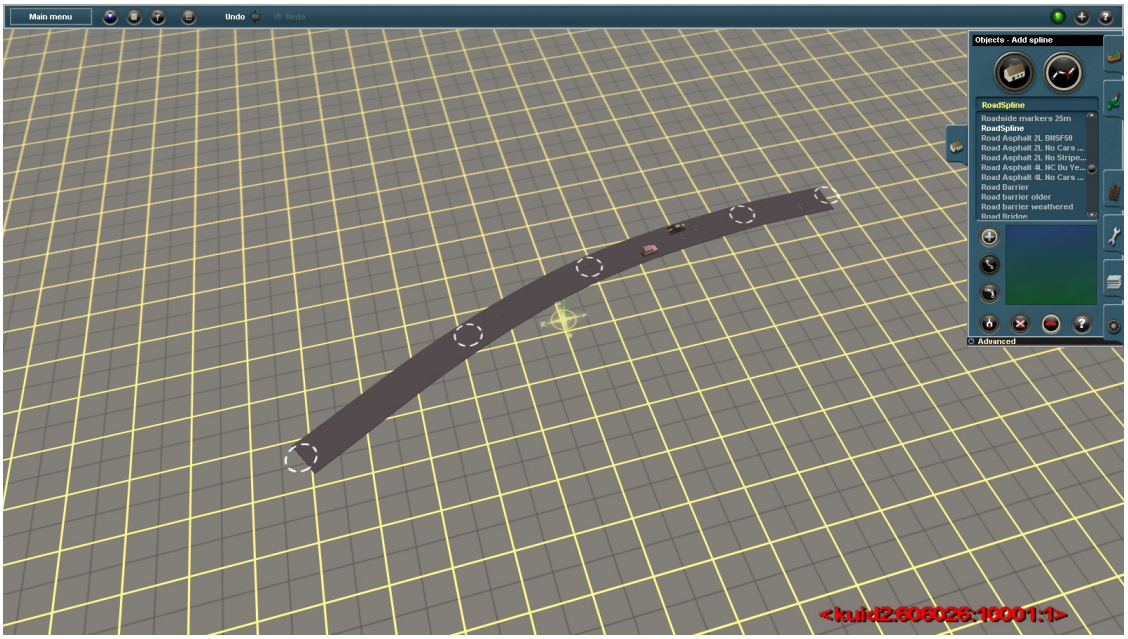
Save the model, then select the menu item at **Plugins ► RubyTMIX ► Export to Trainz**. This will cause the RubyTMIX *Export to Trainz* dialog to display, and the *Simple Spline Generation* section should look similar to the example below.



Once again, the length has been pre-calculated for you. In this example I have already placed a tick mark in the *This is a Spline Object* check box.

Note that I have also selected a *Spline Type* of "Road", and have entered "15" in the *Road – Car Rate* field. The other fields were left at their default values (30 mph speed, 2 lanes, and not a freeway).

Fill in any other export information (kuid number, description, etc.) and then click *Export to Trainz*. Import the model into Content Manager, start Trainz, go into Surveyor, and place the object; again, you'll find it on the *Object* tab as a spline.



Here's a closer view that shows how the 0.2-meter thickness of the road allows carz to sit properly on its surface. As you can see, the shadows are rendering properly under the carz, and there is indeed 2-way traffic being automatically generated.

That's all there is to it. As we've seen, creating simple splines is a fairly straight-forward process that works reasonably well as long as you follow the three simple rules given at the start of this section:

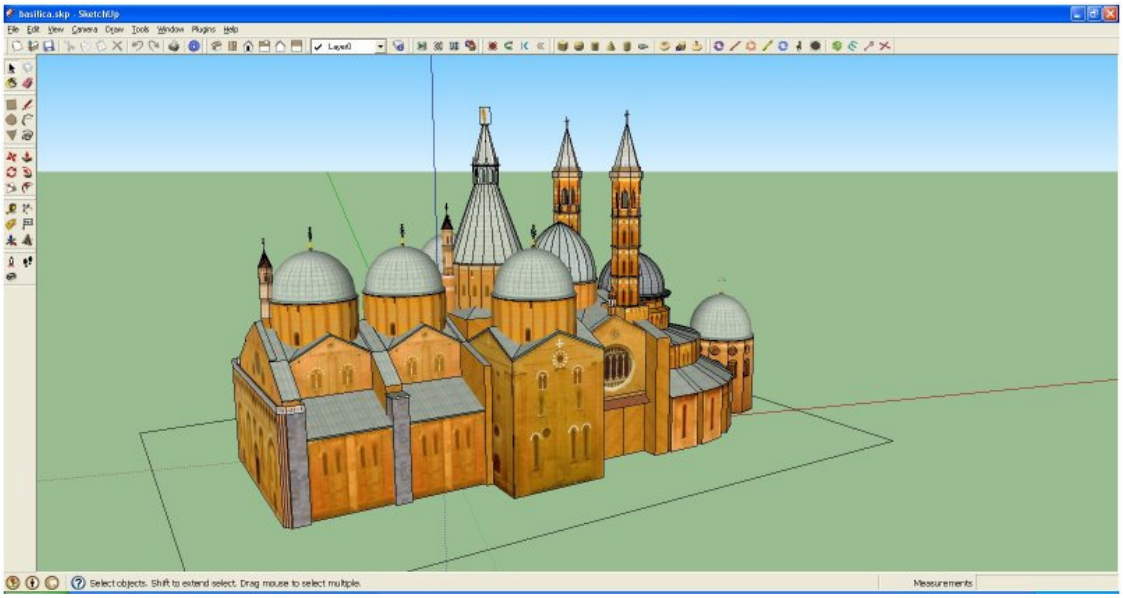
- Splines must always start at the SketchUp origin
- Splines must always extend in the negative Y (green) direction
- Splines must be centered on the Origin in the X (red) direction.

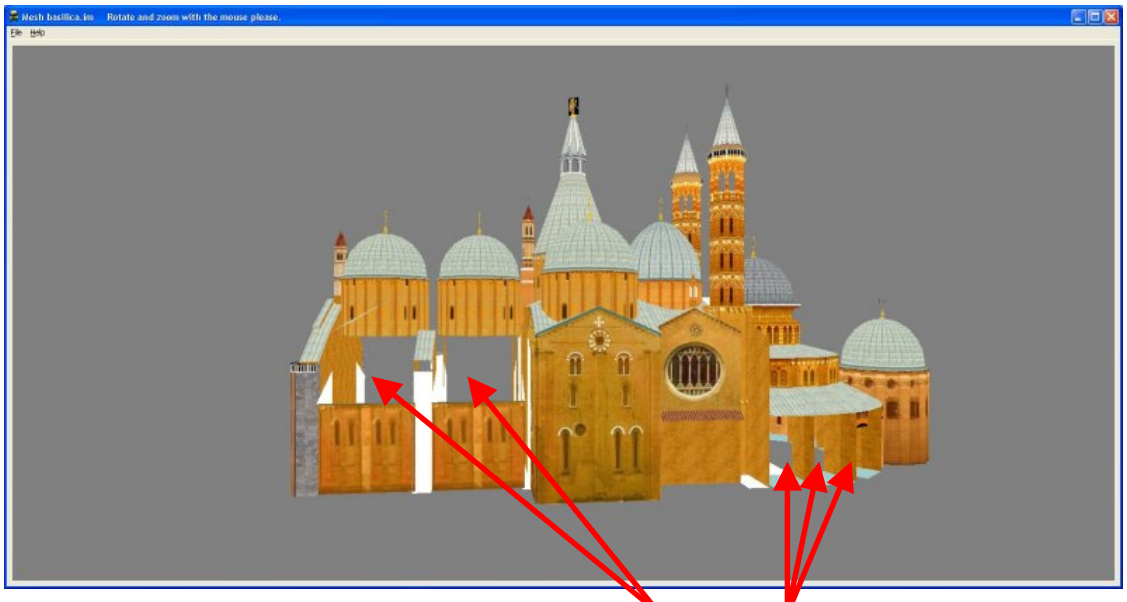
5 Troubleshooting and Advanced Topics

This chapter describes how to deal with common problems that may arise when attempting to export SketchUp models to Trainz, and also provides information for advanced users who may wish to perform some or all of the post-processing steps themselves.

5.1 Dealing with Missing or Inside-Out Geometry

One of the most common issues you will run into when exporting models from Google SketchUp with RubyTMIX is missing or inside-out geometry. Let's look at an example, using *The Basilica of Saint Anthony* by Arriga Silva (obtained from the Google 3D Warehouse). The first image below shows the model in SketchUp, while the second image shows the model in PEV's Mesh Viewer after being exported by RubyTMIX.



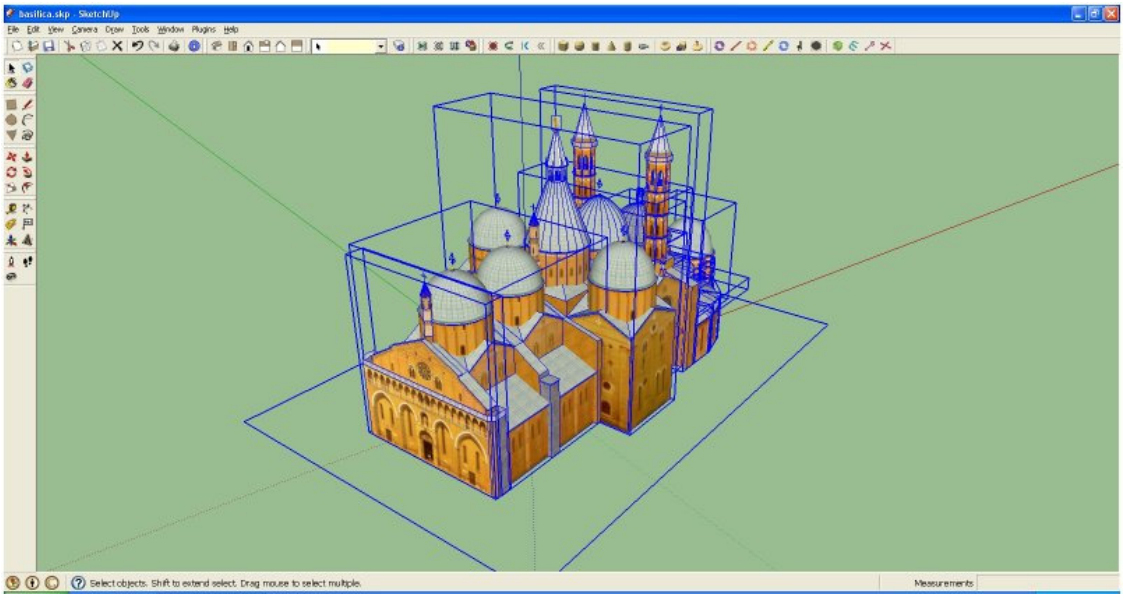


In the second image, you can clearly see that part of the roof is missing, as are many of the walls of the round section at the right. When this occurs for any particular model, it's primarily due to the way that model is constructed.

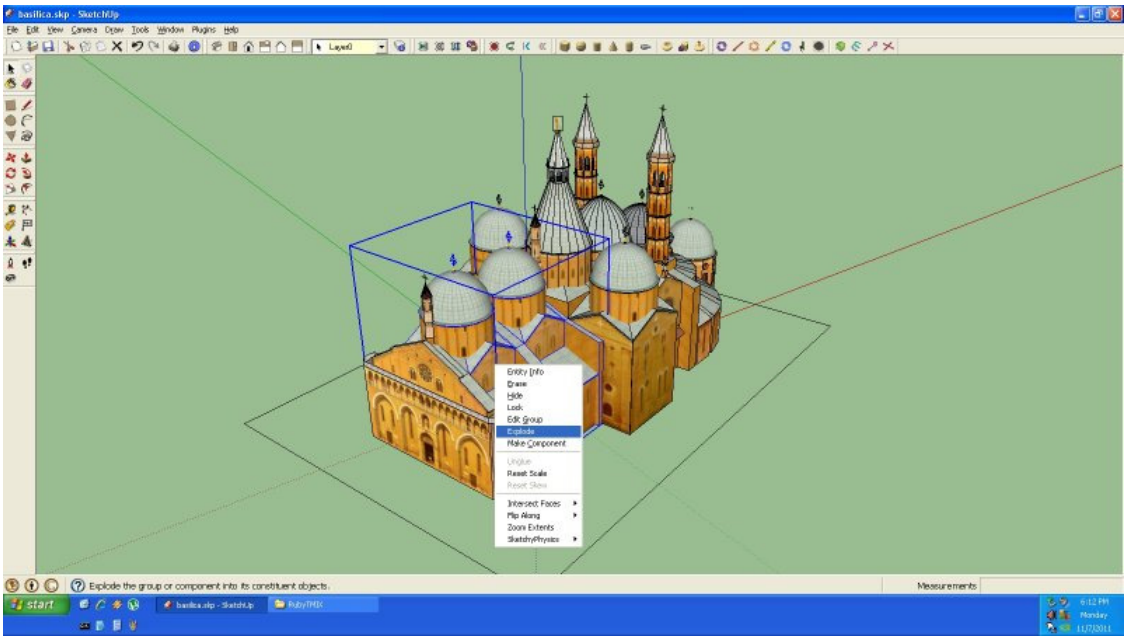
- When you are building a model in SketchUp that exhibits *bilateral symmetry* (two halves which are identical mirror images of each other) or *radial symmetry* (identical parts that repeat in a curve about a central point), you can save a lot of time and effort by making only one of those parts, turning it into a component, and then copying that component and flipping or rotating it to create the other identical parts of the model.
- This is a great feature of SketchUp – and is also something that is incredibly hard to deal with when writing an exporter, because when SketchUp reflects components in this way it sometimes *changes the winding order of the triangles* that make up the faces – which in effect means that some parts of the model follow the left-hand-rule for face normals, and some other parts follow the right-hand-rule. If an exporter is expecting one rule, and the data really follows the other, then the geometry ends up inside-out – and it is extremely difficult, if not impossible, to automatically detect this change in the order of the data via the Ruby API because there are very few clues.
- Even so, RubyTMIX can usually detect geometry that is flipped evenly across one of the main axes, and when it does it will automatically reverse the winding order of the associated data. But, if the geometry is flipped across multiple axes or in a circle, then the detection test fails (two wrongs make a right, so to speak) and RubyTMIX can't automatically determine that it needs to treat the data any differently than usual.

This is very easy to fix, using a technique known as "exploding a component instance." *Exploding a component effectively turns its faces into individual geometry elements that always have the correct winding order regardless of how the component was originally flipped, mirrored or rotated.* RubyTMIX can then export the model with no issues.

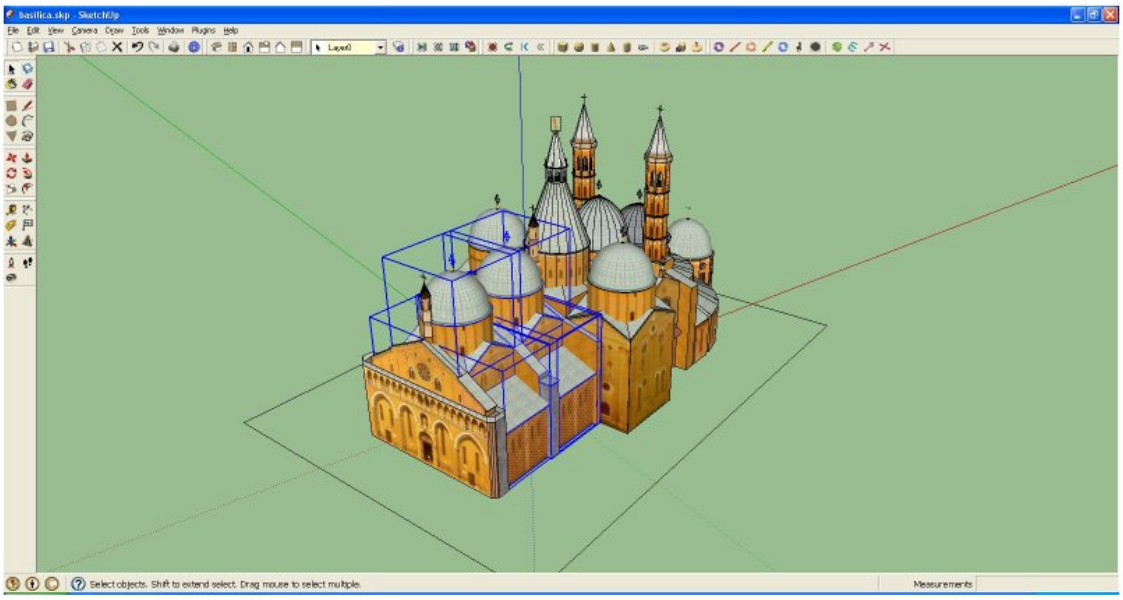
Let's see how this is done – starting with the image below, which shows all the top-level components in the model (bounded by blue boxes).



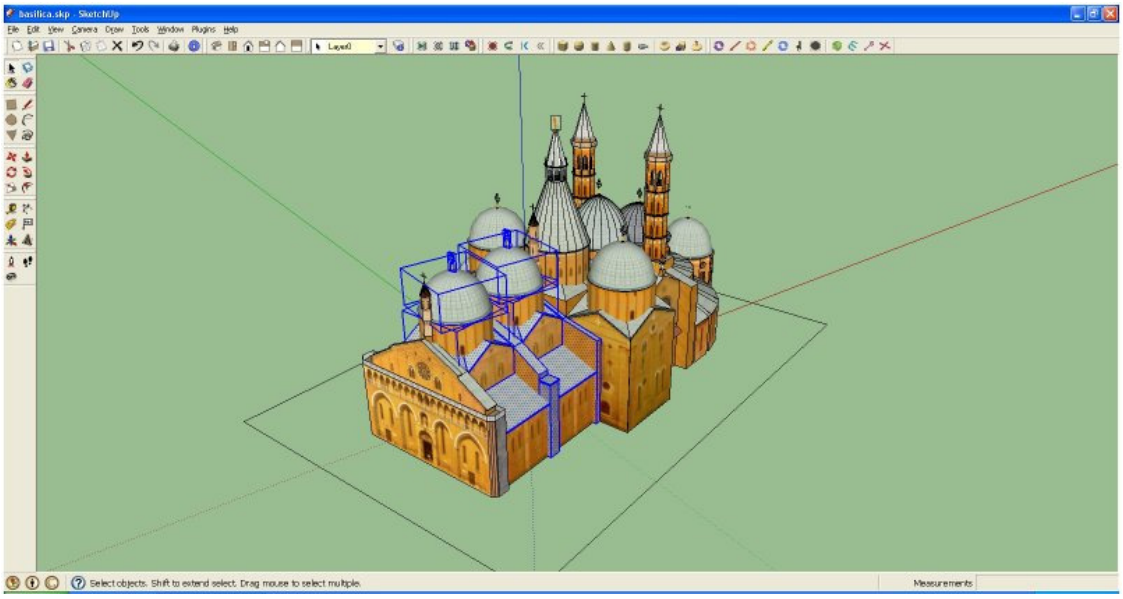
In selection mode, I clicked in the green background area so that none of the components were selected; then clicked on just the roof areas at the near end. This revealed that this entire section was one large component, as shown below.



Next, I right-clicked this component to access its context menu, and selected **Explode**, also as shown above. This resulted in the display below.

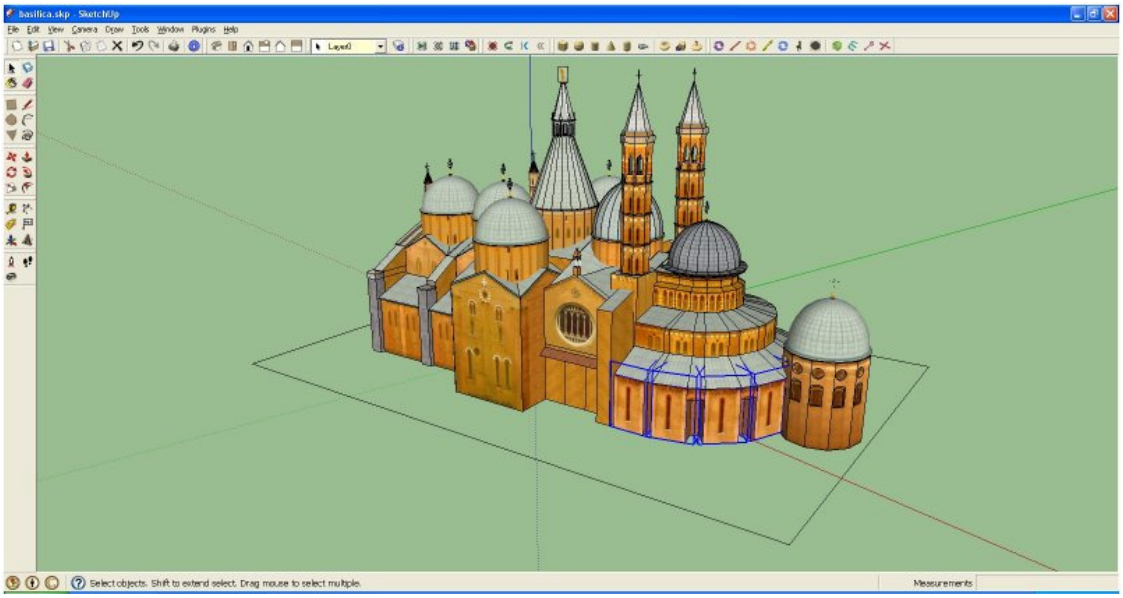


This shows that the larger component was actually made up of two smaller components – which I also needed to Explode, since the missing roof areas were still within those components. Since both of these components were still selected from the previous Explode operation, I right-clicked one of them and selected **Explode** from the context menu again. This gave me the view at the top of the next page.

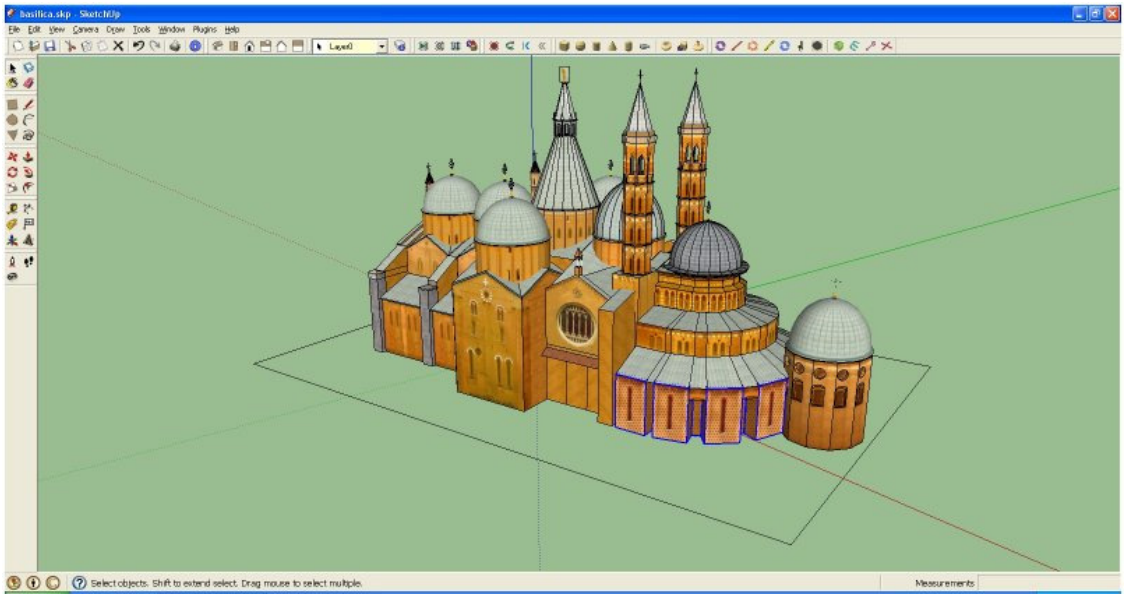


As you can see, the rooftop areas that were missing in our export attempt are now no longer within a component – they have been broken out into individual geometry elements, which is what we need.

Next, I selected the four walls in the rounded section that also did not export correctly before, using Shift+Click to select them one at a time. You can see the result of the selection process in the image below – clearly these walls are also contained within components that have been replicated in a radial fashion as described earlier.

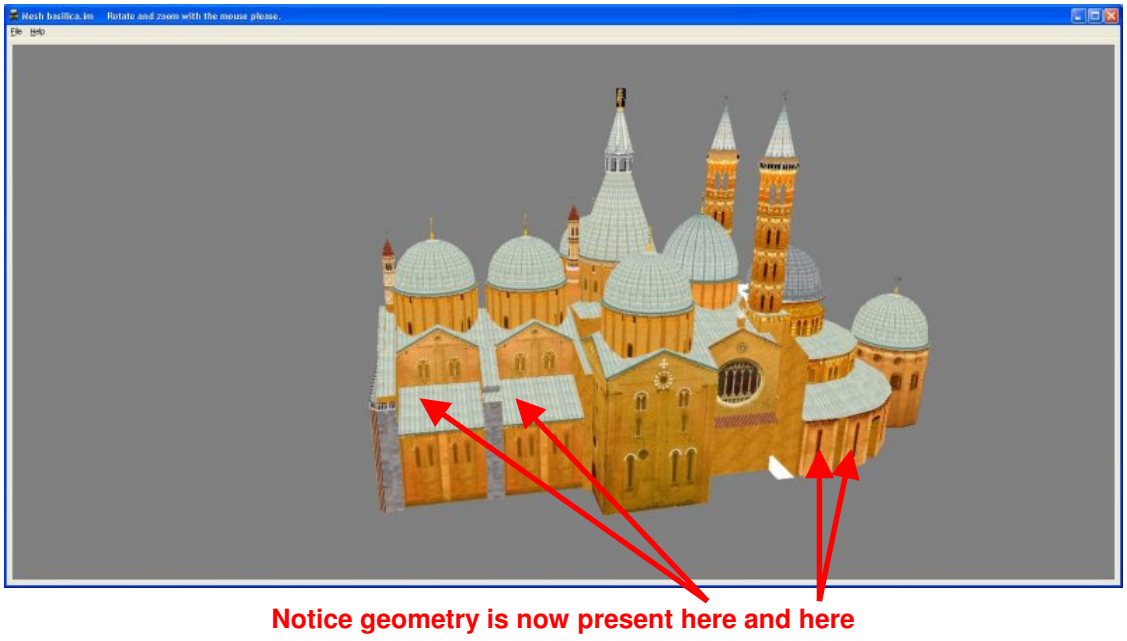


I then right-clicked one of the selected components and selected **Explode** from the context menu. This gave me the view at the top of the next page.



As you can see, the walls that were missing in our export attempt are now no longer within a component – they have been broken out into individual geometry elements, which is what we need.

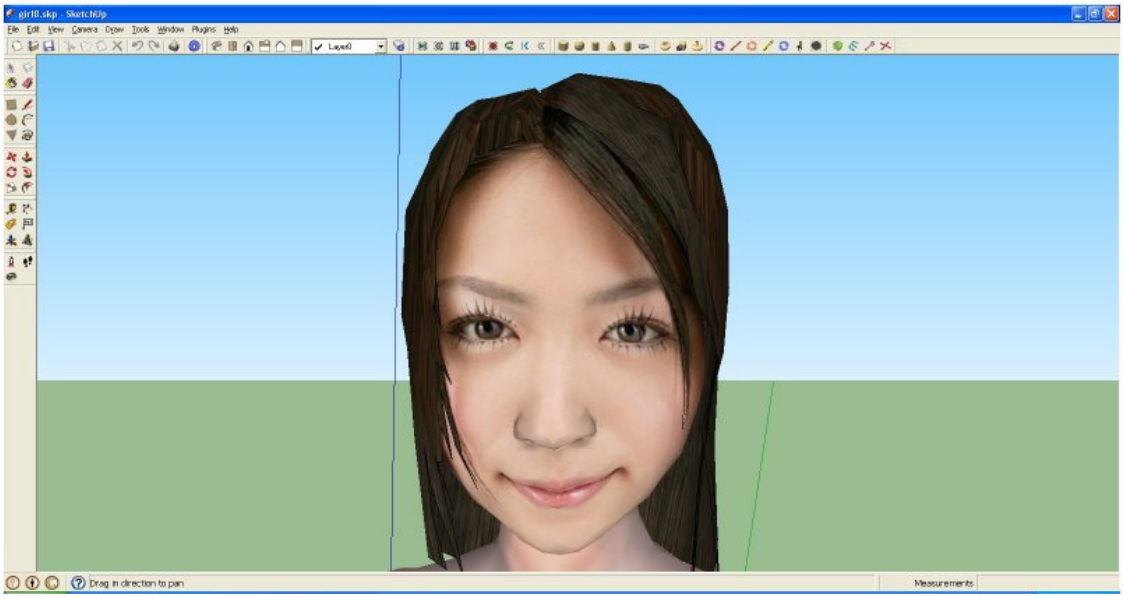
I continued in the same fashion to explode the other areas that did not render properly in the original export attempt. Then I exported the model again, and when I previewed it in PEV's Mesh Viewer I got the result shown below.



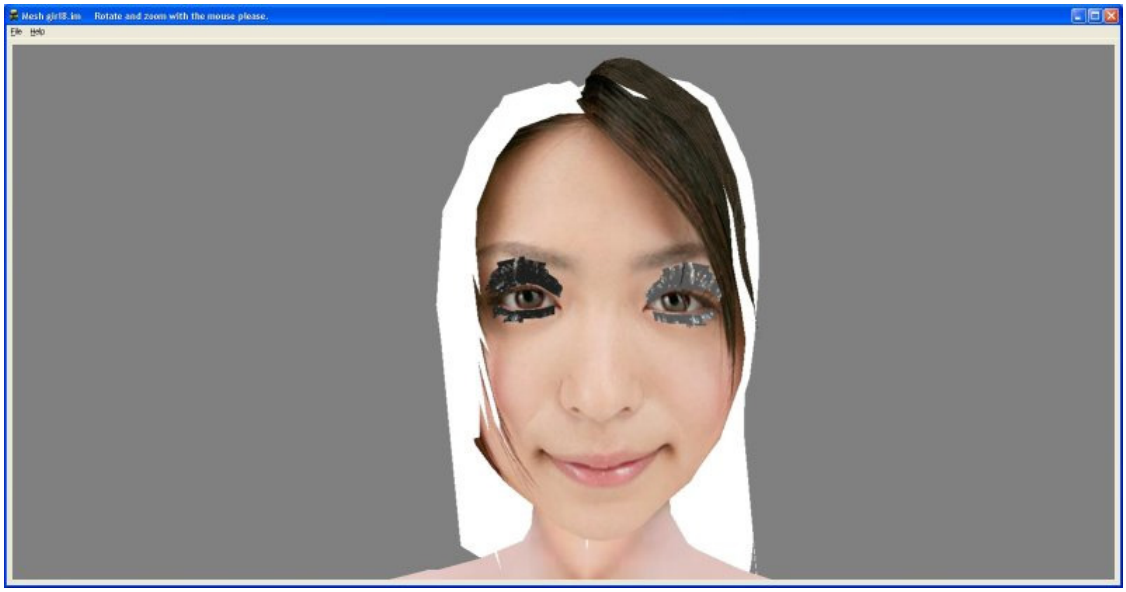
Notice geometry is now present here and here

In this image, you can clearly see that all of the previously missing geometry is now present and properly rendered. Problem solved!

This issue can also manifest itself in other ways that are not so obvious. Let's look at another example, using a 3D model of a girl that has a different but related export problem.



This image shows the girl in SketchUp. The image below shows how she looks in PEV's Trainz Mesh Viewer after being exported by RubyTMIX, with no special modifications.

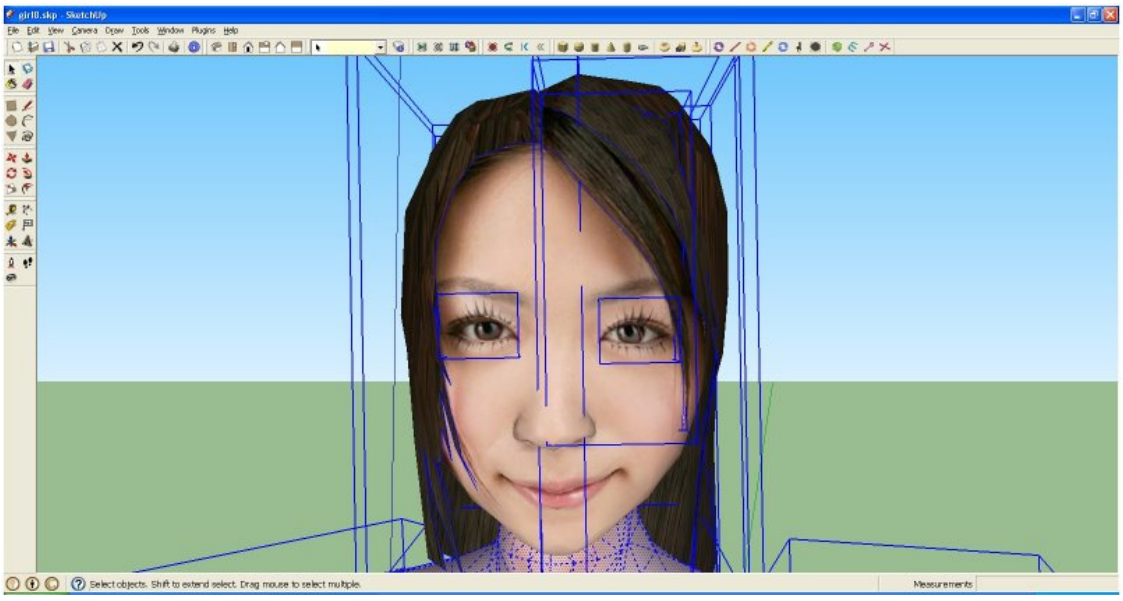


No, she's not wearing some kind of strange hat. Her hair didn't render correctly because, even though it looks like 'something' is there, the geometry is in fact inside-out and you are seeing the back faces of the polygons that make up her hair.

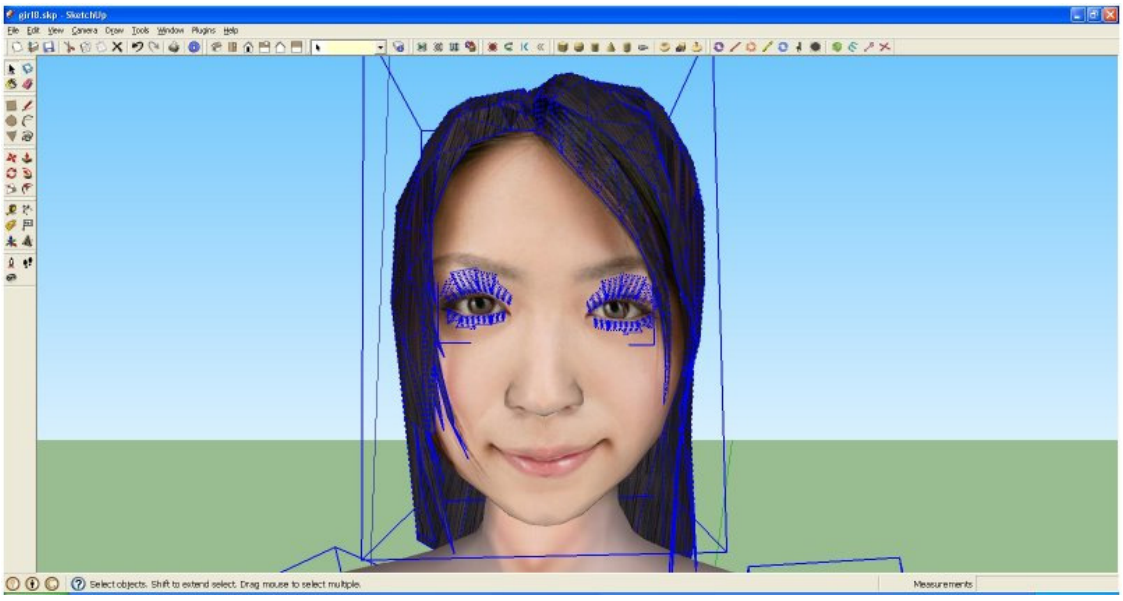
As before, this issue is caused by the way the model is constructed – components were used and mirrored in ways that RubyTMIX cannot automatically detect. And, as before, the solution is very simple: all we have to do is explode those components into their individual geometric elements, and everything will be fine.

You might notice something else in the image above, which are the dark areas where her eyelashes should be. The reason for this is currently unknown, so we'll preview the final results in Trainz itself at the end of this example.

As you can see in the following image, the girl's head is made up of several top-level components.



With these components selected, I right-clicked and selected **Explode** as previously described. This resulted in the display below.



As you can see, her hair and eyelashes are now no longer within a component – they have been broken out into individual geometry elements, which is what we need.

At this point I exported the model again; previewed it in Trainz Mesh Viewer to verify that her hair was rendering correctly (ignoring the black areas around her eyes for the reasons explained a moment ago) and then imported her into Content Manager. After opening Surveyor, I placed her on the grid and then zoomed in to see how she looked. This is shown in the image at the top of the next page.

And there she is, with her hair (and eyelashes) rendering properly.

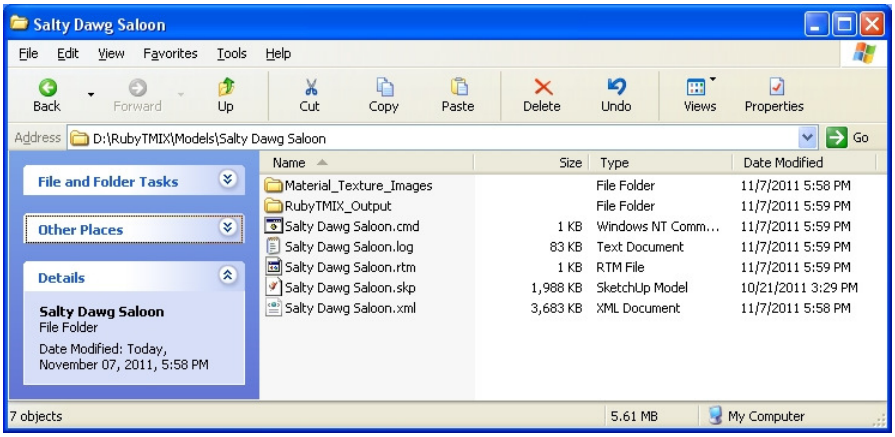
Therefore, if you run into issues with missing or inside-out geometry – or if things just don't seem to be rendering correctly for any reason – always try the 'Explode' techniques described here first. As shown, they are simple and very easy to perform.

5.2 A Closer Look at the Output of RubyTMIX

When RubyTMIX exports a model, it creates a series of files and folders at three different locations. This section, meant for Advanced Users, describes these locations and the files they contain after an export has taken place.

5.2.1 Files Created at the Model's Original Location

When RubyTMIX exports a model, it typically creates four files and two folders within the same folder where the model's **.skp** file was located. This is one reason why each **.skp** file needs to be in its own folder, as emphasized at the start of Chapter 3.



The files of interest in this example are as follows. Note that, in actual practice, all of the file names will be the same as that of the model's **.skp** file – *Salty Dawg Saloon* just happens to be the one used as an example here.

- **Salty Dawg Saloon.xml:** This file is the input to the Trainz Mesh Importer utility. As its extension implies, it is an XML file that meets the requirements set forth by N3V for input to the utility. Note that this file will only be present after export if you chose to retain it by **not** ticking the check box labeled *Delete TrainzXML file when this dialog is closed* in the RubyTMIX *Export Results* dialog.
- **Salty Dawg Saloon.cmd:** This file is created to allow you to manually re-run the Trainz Mesh Importer after the export has been completed and you have exited from SketchUp. To re-run the Importer, you must have retained the XML file, as noted above; then, just double-click the CMD file and the Mesh Importer will execute in a command line window with the XML file as its input. After execution completes, the Mesh Importer command line window will remain open so you can see the results; type any key to close the window.
- **Salty Dawg Saloon.log:** This is a text file that contains a log of the export process. Each line in the file has a date and time stamp and describes a particular operation that took place during the export process. To view this file, double-click it and it will typically open in Windows Notepad. If you are having trouble exporting a particular model with RubyTMIX, I may ask you to send me this file as part of the process of diagnosing the issue.
- **Salty Dawg Saloon.rtm:** This is a text file that contains the saved RubyTMIX settings for this particular model. It follows the standard Windows 'INI' file format, with sections delineated by headers and containing name=value pairs. You can view this file in Windows Notepad by opening it explicitly via **File** → **Open**.

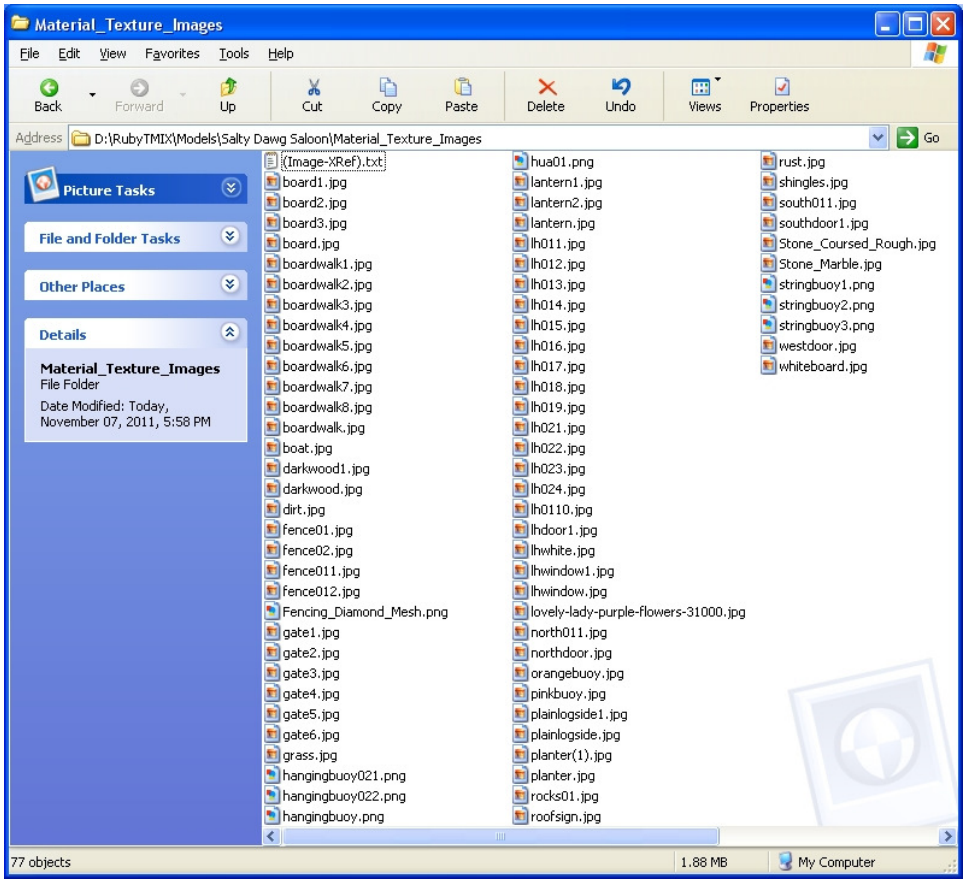
WARNING

NEVER EDIT THIS FILE YOURSELF. If you make a mistake, it may prevent RubyTMIX from processing this model again until the issue is corrected.

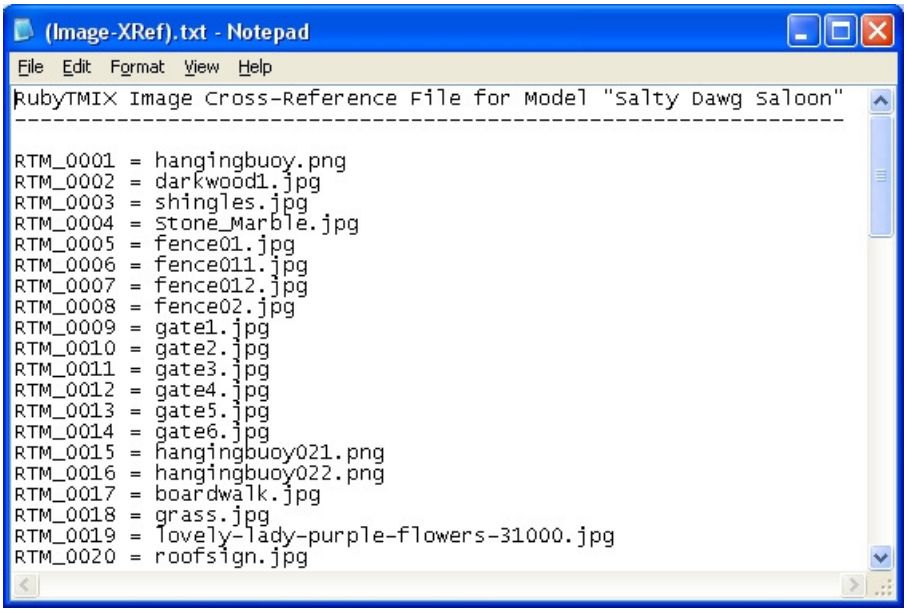
The folders of interest in this example are described in the following sections; note that these folders will always have the same names regardless of the name of the **.skp** file.

5.2.2 Files Created in the Material_Texture_Images Folder

The **Material_Texture_Images** folder receives all of the original image files that represent the textures used in the model. The files will have their original names (including Unicode characters if present) and original extensions, so you can easily post-process them yourself if desired; more on that later. For the Salty Dawg Saloon example model, the contents of this folder appear as shown in the image on the next page.



This folder also receives a cross-reference file that shows you which 'controlled' file name RubyTMIX associated with which original image file. This file is always named **(Image-XRef).txt**; the parenthesis are added to the name to ensure it appears first in the folder display when the list is sorted in ascending order by name. If you double-click this file it will open in Windows Notepad, and will appear similar to the example below.



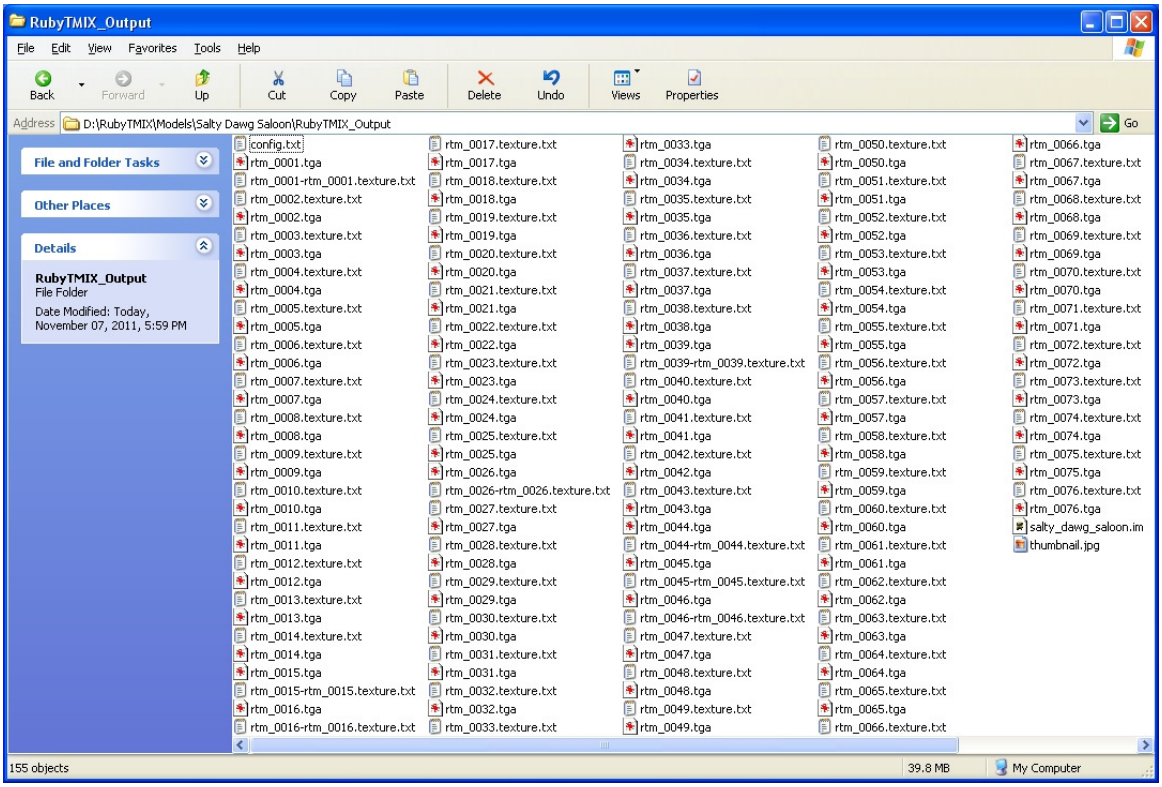
This example shows that the controlled file name **RTM_0001** is associated with the actual texture named **hangingbuoy.png**, and so on. The extensions are not listed for the **RTM_xxxx** files since they are always the same as the extension of the original image file.

5.2.3 Files Created in the RubyTMIX_Output Folder

The **RubyTMIX_Output** folder receives all of the files that need to be imported into Trainz Content Manager in order to create an asset that can be used in Surveyor.

As shown in the image at the top of the next page, you will usually find five types of files in this folder:

- Various image files incrementally named **RTM_0001**, **RTM_0002**, and so on, that represent the texture images. These will have a TGA extension if automatic conversion took place, or will have the same extension as the original texture file (PNG, JPG or TIF) if auto-conversion to TGA was not specified.
- The **config.txt** file that was automatically generated by RubyTMIX.
- The **IM** (indexed mesh) file that was generated by the Trainz Mesh Importer.
- Various **.texture.txt** files that were also generated by the Trainz Mesh Importer. The names of these files are chosen by the Importer and may sometimes have the image name repeated; this is apparently normal behavior and is no cause for concern.
- A file named **thumbnail.jpg**, which is the thumbnail image generated by RubyTMIX.



5.3 Converting and Resizing Texture Images Manually

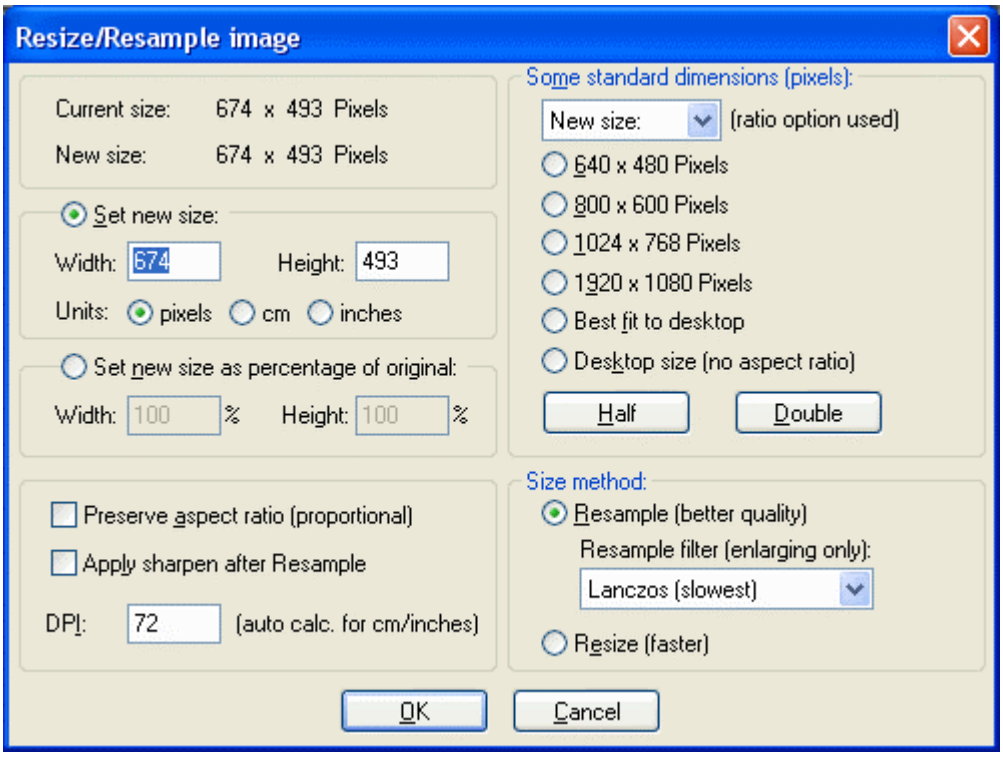
If you elect to perform some of the image post-processing steps yourself, you will need to deselect certain options (described in Section 3.3) at the start of the export process:

- If you want to convert PNG images to TGA yourself, deselect *Convert PNG images to TGA format*. This will cause all PNG images produced by RubyTMIX (e.g. rtm_xxxx.png) to retain their original image data format, even if they are resized. However, since Trainz cannot import PNG files, all images with a PNG extension will still have a TGA extension and will still be identified as having an alpha source as part of the material definition that is written to the TrainzXML file. This means that you must convert all such images to 32-bit TGA; if you do not want to use the alpha channel, you must also edit the associated .texture.txt file to remove the alpha source parameter.
- If you want to convert TIF images to TGA yourself, deselect *Convert TIF images to TGA format*. This will cause all TIF images produced by RubyTMIX (e.g. rtm_xxxx.tif) to retain their original image data format, even if they are resized. However, since Trainz cannot import TIF files, all images with a TIF extension will still have a TGA extension and will still be identified as having an alpha source as part of the material definition that is written to the TrainzXML file. This means that you must convert all such images to 32-bit TGA; if you do not want to use the alpha channel, you must also edit the associated .texture.txt file to remove the alpha source parameter.
- If you want to convert JPG images to 24-bit TGA yourself, or want to use the JPG images in Trainz as-is, deselect *Convert JPG images to TGA format*. This will cause all JPG images produced by RubyTMIX (e.g. rtm_xxxx.jpg) to retain their original image data format, even if they are resized. JPG images **can** be used with Trainz, as long as they are sized to meet the power-of-2 requirements.
- If you want to convert BMP images to 24-bit TGA yourself, or want to use the BMP images in Trainz as-is, deselect *Convert BMP images to TGA format*. This will cause all BMP images produced by RubyTMIX (e.g. rtm_xxxx.bmp) to retain their original image data format, even if they are resized. BMP images **can** be used with Trainz, as long as they are sized to meet the power-of-2 requirements.
- If you want to resize images yourself, deselect *Resize texture images to closest-power-of-2*. Note that RubyTMIX will then not resize **any** images at all, meaning you must resize **all** the images yourself.

IrfanView is commonly the tool of choice to resize images. However, be warned: **IrfanView does not preserve the alpha channel information for PNG and TIF files if you save them as TGA**. Therefore, for PNG and TIF images, you should use IrfanView to resize the image and save it back to its original format – being sure to preserve alpha information via the options dialog that appears during file saves, if applicable – and then use PEV's *Images2TGA* utility to convert the files to 32-bit TGA. For JPG and BMP images, you can safely use IrfanView to create 24-bit TGA files directly.

Remember to delete the original PNG, TIF, JPG or BMP image(s) before importing the model into Trainz, or errors may result in CM.

To resize an image in IrfanView, select **Image** → **Resize/Resample** when the image is loaded, which brings up the *Resize/Resample Image* dialog box:



In order to resize the image to power-of-2 dimensions as required by Trainz, you will usually need to un-tick the check box named *Preserve aspect ratio*. Then be sure *Set new size* is selected, and type the new dimensions you want in the *Width* and *Height* fields. Then click *OK* and the image will be resized. For details about other options in this dialog, refer to the IrfanView help file.

When doing this, it may be useful for you to know what values RubyTMIX uses when it automatically resizes images; these values are listed in the table below (in pixels):

Original Dimension (X or Y)	Power-of-2 Value Used by RubyTMIX
0 to 2	2
3 to 5	4
6 to 11	8
12 to 23	16
24 to 47	32
48 to 95	64
96 to 191	128
192 to 383	256
384 to 767	512
768 to 1535	1024
1536 or greater	2048

5.4 Resizing the Thumbnail Manually, or Supplying Your Own

You can also elect to resize the thumbnail that is automatically generated by RubyTMIX, or you can supply your own image entirely. To do so, you will need to deselect certain options (described in Section 3.3) at the start of the export process:

- If you want to resize the auto-generated thumbnail yourself, deselect *Resize thumbnail image to 240 x 180*.
- If you want to supply your own thumbnail image entirely, deselect *Generate thumbnail image from model*. If you supply an image named **thumbnail.jpg** then it will be referenced correctly in the 'config.txt' file that is automatically generated by RubyTMIX; if you call your thumbnail something else, you will also need to edit the 'config.txt' file as needed to reference this new image file name.

You might want to resize the thumbnail yourself, or supply an image of your own, because the thumbnails that SketchUp generates may not always be of very good quality – for large models, they tend to be somewhat blurry – and because the automatic resizing process may produce an odd-looking image due to the way it is stretched. If you choose to manually resize the image, you could for example crop it instead, and/or place a clearer image with the proper perspective on a 240 x 180 background.

5.5 Future Coverage of Other Topics

Because SketchUp is such a powerful tool with so many features, and because its uses in relationship to Trainz may vary widely, it is impossible to cover every possible eventuality in advance in a manual such as this. For that reason, I have really only touched on the main points that you need to know about.

It is anticipated that a series of separate Tutorials will be developed over time to address not only various modeling techniques, but also issues that may arise and whatever their work-arounds are found to be. As noted in the Preface, this manual is also considered a living document, and is intended to be updated periodically to include more details about common issues or additional information about topics that prove to be of special interest.

5.6 Assistance with Other Issues

If you have problems with RubyTMIX, please contact me at mike@dhobh.net (or post in the Trainz forums, or send me a PM from there) and I will do my best to assist you. DO NOT send attachments to me unless I ask you to; if I receive an e-mail with attachments from a source I don't know, it will be deleted unread.

If you have problems with SketchUp, I am **NOT** your go-to guy for this. There is a wealth of information related to SketchUp on the internet, including tutorials and how-to videos, so you would do well to seek a solution to your issue(s) from one of these sources. I also heartily recommend the book *Google SketchUp 8 for Dummies*; more information about this can be found at http://www.aidanchopra.com/.

This page intentionally left blank.

End of Document.