FSDS xTweak

VERSION 2.0

Manual Edition 2.8

TABLE OF CONTENTS

TABLE OF FIGURES

1. Acknowledgements

A big thanks to the following good folks:

- Rick Piper, Felix Rodriguez, Mark Harper David Bushnell and Bobby Hayes for their testing and contributions with ideas and general encouragement.
- The regulars at the Freeflight Design Shop forums for their support, bug reports and ideas (http://www.freeflightdesign.com)
- AVSIM for maintaining a great free library service which allows the distribution of this tool

These tools are written in C++, using Microsoft Visual Design Studio 2008 Professional Edition. The config files are processed using Allegro (v4.2.0), a great open source game programming library (http://www.talula.demon.co.uk/allegro/). The polar matrix decomposition code is by Ken Shoemake, and is taken from Graphics Gems IV (Edited by Paul Heckbert, 1994, Morgan Kaufman Publishing).

2. Getting help and support

If you need help, you can go to the development blog (http://davenunez.wordpress.com) or the FSDSxTweak support forum on the Freeflight Design Shop forum (http://www.aerodynamika.com/cgi-bin/yabb/YaBB.cgi?board=FSDSTweak). If you do not find any help there, you can email me directly (dave.nunez.za@gmail.com). Please note that I support these tools as a hobby, so it may take me a while to respond to requests.

3. License and terms of use

I would appreciate it if you would email me (dave.nunez.za@gmail.com) a screenshot of your project, be it freeware or commercial. Knowing people use the tools keeps me updating them!

4. Donations

If you enjoy or use these tools, please consider making a charitable donation to Amnesty International (http://www.amnesty.org/en/donate) or the International Committee of the Red Cross (http://www.icrc.org/Web/eng/siteeng0.nsf/htmlall/helpicrc).

5. Installation

Before installing, you need to have the following components installed:

- The FSX SDK (which is found on your FSX Deluxe installation DVDs). Update it to the current standard by downloading updates from FS Insider: http://www.fsinsider.com

- Flight Simulator Design Studio (FSDS) v3.5.1. If you don't have the latest patches for FSDS, get them from (updating from v3 to v3.5.1 is free): http://www.abacuspub.com/fsds35/fsds35main_default.html

- The Microsoft Visual C++ 2008 Redistributable Package, which you can download from here: http://www.microsoft.com/downloads/details.aspx?familyid=9B2DA534-3E03-4391-8A4D-074B9F2BC1BF&displaylang=en

- The Microsoft .NET 3.5 network, which you can download form here: http://www.microsoft.com/downloads/details.aspx?FamilyID=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en

Once you have installed these, you can go ahead and run the installer. The installer provided should put everything into the right place automatically. In case it doesn't, or you wish to move things around, all the files are best placed into your FSDS folder (by default, this folder is c:\program files\abacus\FSDS_V3.5). Then move the fsdsxtweak_pluing.exe file into the FSDS plugins folder (by default this is c:\program files\abacus\FSDS_V3.5\plugins). If you are running Vista or Windows 7, you should not install FSDS to Program Files (see below).

6. Compatibility with Vista, Windows 7 and 64 bit

Versions of FSDSxTweak before 2.6 had some compatibility issues with Vista/Windows 7. This has been fixed in version 2.6 (and subsequent versions). Here is the breakdown on compatibility, and how to work around any outstanding issues:

1. FSDSxTweak is designed to be run by a ***non-administrator*** user, with ***UAC turned on***. It is not a good idea to use your computer day-to-day as administrator, or to turn UAC off, as this can open you up to viruses and other attacks.

2. Vista and Windows 7 prevents programs from writing to the Program Files folder. This will break FSDSxTweak. To run FSDSxTweak, you should install FSDS into some directory *outside* of Program Files, and the following the installation instructions detailed above (e.g. to C:\FSDS_v3_5).

3. From version 2.6 onwards, FSDSxTweak will happily write to the registry to save your settings without having to be an administrator user.

4. All of the FSDSxTweak suite tools are fully compatible with 32 and 64 bit versions of Windows, provided you have installed the correct version of the Microsoft .NET framework.

7. The build pipeline

To properly understand what FSDSxTweak does, a little explanation of how FSDS builds models is required. A normal build on FSDS is a two step process – the geometry data which you create with FSDS (stored in the FSC file) is compiled by FSDS into two .X files (one for the interior/VC model, and the other for the exterior model). These .X files are then handed off to XtoMDL (which is part of the FSX SDK), which builds a separate MDL file for the interior and exterior models (see Figure 1):

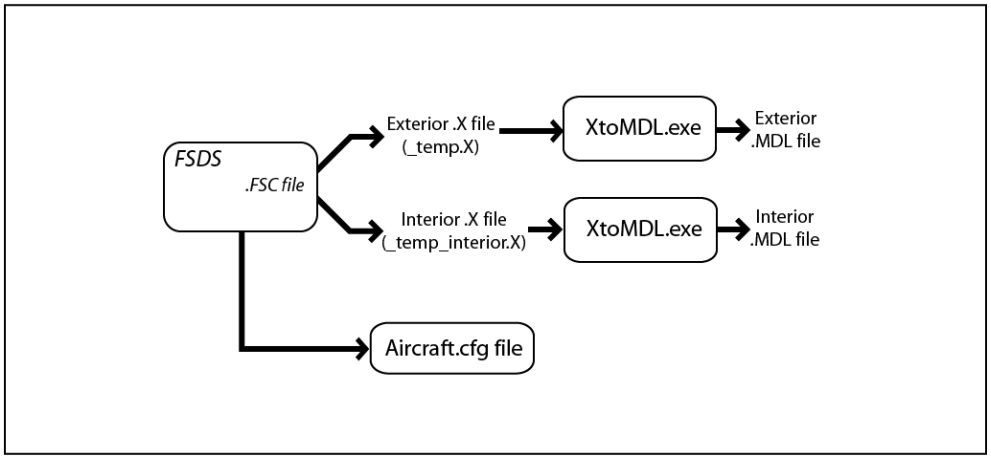


Figure 1:The default FSDS build pipeline

FSDS also generates two .xanim file (one for interior and one for exterior), which contain all the animation/keyframe data (these xanim files are also used by XtoMDL to build the MDLs). FSDS also modifies the aircraft.cfg file, to enter the viewpoint, gear positions, etc.

FSDSxTweak uses a modified pipeline, by inserting itself between FSDS and XtoMDL (see Figure 2):

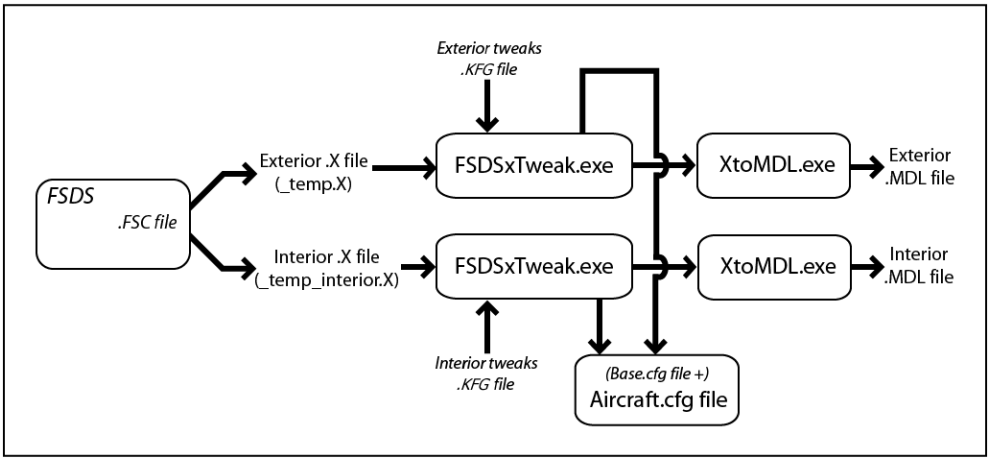


Figure 2:The FSDSxTweak build pipeline

FSDSxTweak takes the two .X files produced by FSDS, and applies the tweaks defined in the KFG files to them, to create new .X files. These are then handed off to XtoMDL, to build MDL files. After generating the .X files, FSDSxTweak also adds information to the aircraft.cfg file (camera positions, gear parameters, etc).

7.1 *Overview of the FSDSxTweak suite*

Four tools are included in this package:

fsdsxtweak.exe: All that is required to tweak models is this tool. It is command-line driven and reads plain ASCII text config (KFG) files that define the tweaks to apply (see section 13 for a list of commands)

fsdsxtweak_plugin: In order to facilitate the use of fsdsxtweak.exe, this tool allows for single-button-press compilation of models. It is effectively a graphical frontend to *fsdsxtweak.exe*, and also adds some extra housekeeping tools (see section 8 for details)

fsdsxtweak_edit: This tool allows for easy creation and editing of the tweak files used by fsdsxtweak (KFG files). It allows for fast easy editing, and reduces errors associated with syntax errors found in hand-edited config files (see section 13 for details).

modeldef_edit: This tool allows you to edit the animation and parts dictionary file (modeldef.xml) to add your own custom animations to a model

Essentially, FSDSxTweak does the actual work in tweaking the .X files. FSDSxTweak_Plugin makes your life easier by eliminating the need to enter command line arguments, and FSDSxTweak_Edit makes your life easier by eliminating the need to edit KFG tweak files in a text editor (which requires remembering all the options and commands).

8. FSDSxTweak_Plugin user's guide

This tool can be launched from the **Plugins** menu of FSDS. It is used to launch the build process with the new pipeline (see section 7 above). The interface is shown in Figure 3, Figure 4 and Figure 5.

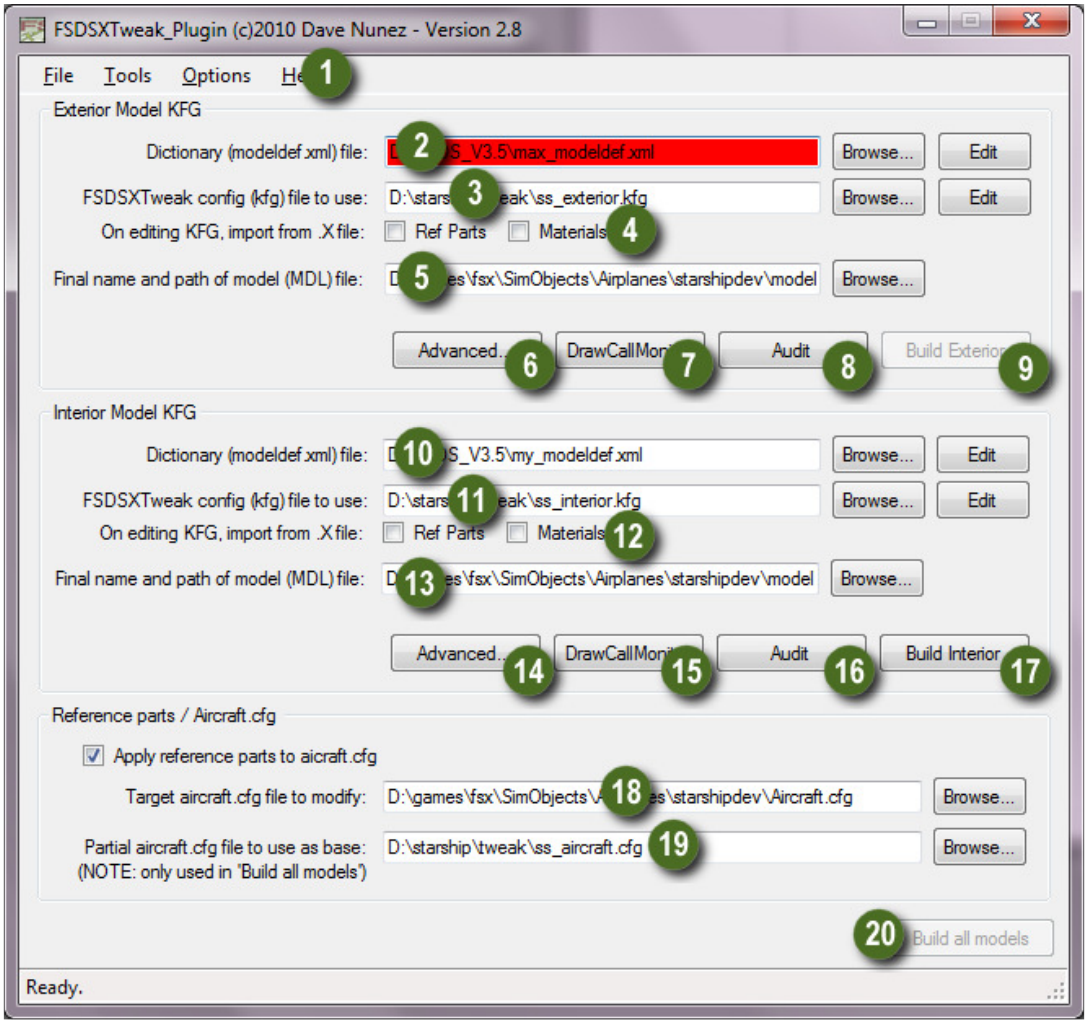
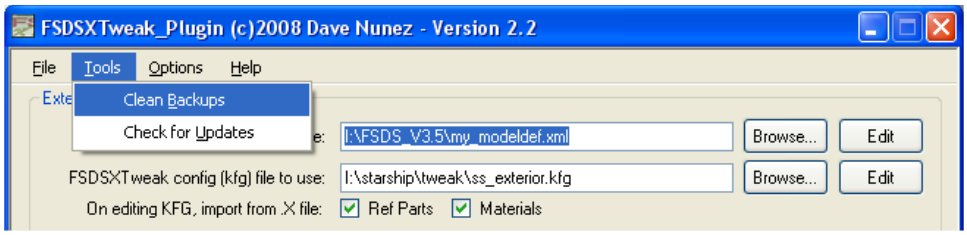


Figure 3: FSDSxTweak_Plugin interface

The components of the main interface are:

1.***Menus:*** The following features are available in the menu:

File menu: The following is available in this menu:



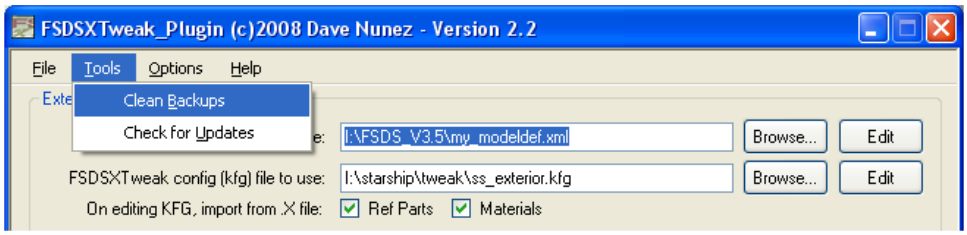
New – Clear all the settings to begin a new configuration (XCF) file.

Open – Open a configuration (XCF) file

Save as – Save the current configuration as a XCF file. Because the target model folder is saved in the XCF file, you will probably have a separate XCF file for each of your projects, so that the models, once compiled, end up in the correct aircraft folders.

Exit – Exit FSDSxTweak_Plugin

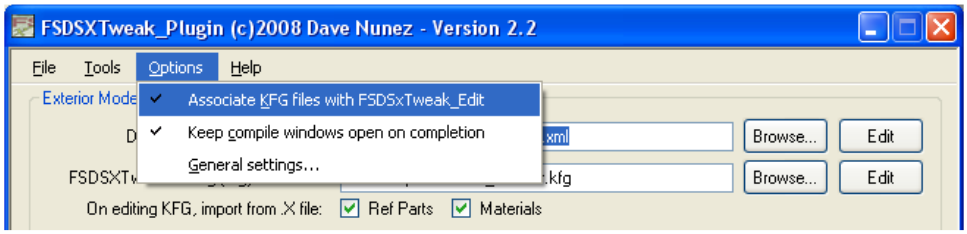
Tools menu: The following is available in this menu:



Clean backups – The backup files generated by FSDS (which are named .m00, .m01, .m02… and .c00, .c01, m02… will be moved to the backup folder you have defined (it does the same thing as when you have selected **Clean backup files to** – see above).

Check for updates – Connects to the update news server to check if a newer version of the FSDSxTweak tools or documentation is available for download.

Options menu: The following is available in this menu:

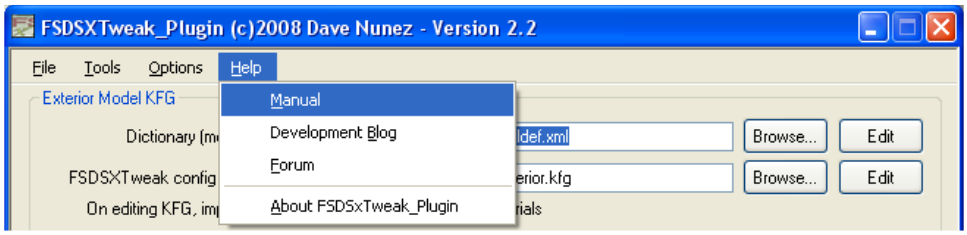


Associate KFG files with FSDSxTweak_Edit – When this feature is turned on, double clicking a KFG file from anywhere in Windows Explorer (or selecting Open on it), will open it up in FSDSxTweak_Edit (this is only possible if you have defined a location for FSDSxTweak_Edit. Turn this off to remove this functionality.

Keep compile windows open on compilation – When this feature is turned on, FSDSxTweak_Plugin will keep the DOS windows which display feedback from the compilation open (press any key to close them). Turning it off automatically closes the window, and removes any feedback. It is advisable to keep this on, or you will not be able to read any error messages from a broken compilation. Please note that when you select 'Build all models', you will need to press a key to close the exterior model build window before the interior model build process begins.

General settings — This opens the settings interface (see below).

Help menu: The following is available in this menu:



Manual – open this document in the PDF reader

Development blog – opens your web browser at the development blog: http://davenunez.wordpress.com

Forum – opens your web browser at the Free Flight Design Studio FSDSxTweak forum:
http://www.aerodynamika.com/cgi-bin/yabb/YaBB.cgi?board=FSDSTweak

Please note that you must register at the forum (for free) to make use of it. The registration page is at:
http://www.aerodynamika.com/cgi-bin/yabb/YaBB.cgi?action=register

About FSDSxTweak_Plugin – displays the current version number

2. ***Dictionary (modeldef.xml) file (Exterior Model KFG):*** This is the location of the dictionary file used by xtomdl to process the part names in your model. Note that this will be the dictionary used to process your exterior model (i.e. not the VC). The dictionary usually resides in `C:\Program Files\Microsoft Games\Microsoft Flight Simulator X SDK\SDK\Environment Kit\Modeling SDK\bin\modeldef.xml`. Clicking on the **Edit** button will open up two files in the selected text editor: The dictionary file itself, and the `partdefsdata.txt` file which is present in the FSDS folder (if one exists) – this allows the inclusion of custom part names. This textbox will turn red if the dictionary entered is not readable (or does not exist), and the compile buttons will remain disabled until a valid dictionary is entered.

3. ***FSDSxTweak config (KFG) file to use (Exterior Model KFG):*** Point this to the KFG file you have defined to tweak your external model. Clicking on the **Edit** button will launch FSDSxTweak_Edit with that KFG file, to allow easy editing. This textbox will turn red if the KFG file entered is not readable (or does not exist), and the compile buttons will remain disabled until a valid KFG file is entered.

4. ***On editing KFG, import from .X file: (External Model KFG):*** This option will import all the existing new reference parts and/or materials from the last built version of your model when you click the **Edit** button associated with the exterior KFG file (see section 0 for a list of new reference parts). Note that you must have a temporary build file in your FSDS folder (called `_temp.x`, which is created when you build the model in FSDS with the **Delete Temp Files** option turned off). Also note that when importing from the .X file, no part or material which already exists in the KFG file is overwritten (see section 9.5 for more details).

5. ***Final name and path of model (MDL) file (Exterior Model KFG):*** This will be the location of your final, compiled FSX model. Normally you will point this to your aircraft's model folder, so that it will load in the simulator.

6. ***Advanced… (Exterior Model KFG):*** This opens up the advanced options for the exterior model build (see the description of the advanced options interface below).

7. ***DrawCallMonitor (Exterior Model KFG):*** If you have defined a location for DrawCallMonitor and a final build of your model exists, this will open DrawCallMonitor and automatically load your model in it to show a performance report.

8. ***Audit (Exterior Model KFG):*** This will produce an audit report of the exterior model. For more information on what an audit produces, see section 10.2 below.

9. ***Build exterior (Exterior Model KFG):*** This will tweak and compile your model, and the resulting build will be placed in the target folder you have specified. If you have turned on **Apply reference parts to aircraft.cfg** and there are suitable parts in the model, these will be merged into your `aircraft.cfg` file. Please note that this will ***not*** make use of a base partial `aircraft.cfg` file, even if one is defined (see point 19 below for an explanation). This button will be disabled if all the settings for the project are not valid (one or more of the textboxes will be red indicating the problem).

10. ***Dictionary (modeldef.xml) file (Interior Model KFG):*** This is the location of the dictionary file used by xtomdl to process the part names in your model. Note that this will be the dictionary used to process your interior model (i.e. the VC). The dictionary usually resides in `C:\Program Files\Microsoft Games\Microsoft Flight Simulator X SDK\SDK\Environment Kit\Modeling SDK\bin\modeldef.xml`. Clicking on the **Edit** button will open up two files in the selected text editor: The dictionary file itself, and the `partdefsdata.txt` file which is present in the FSDS folder (if one exists) – this allows the inclusion of custom part names. This textbox will turn red if the dictionary entered is not readable (or does not exist), and the compile buttons will remain disabled until a valid dictionary is entered.

11. ***FSDSxTweak config (KFG) file to use (Interior Model KFG):*** Point this to the KFG file you have defined to tweak your internal (VC) model. Clicking on the **Edit** button will launch FSDSxTweak_Edit with that KFG file, to allow easy editing. This textbox will turn red if the KFG file entered is not readable (or does not exist), and the compile buttons will remain disabled until a valid KFG file is entered.

12. ***On editing KFG, import from .X file: (Interior Model KFG):*** This option will import all the existing new reference parts and/or materials from the last built version of your model when you click the **Edit** button associated with the exterior KFG file (see section 0 for a list of new reference parts). Note that you must have a temporary build file in your FSDS folder (called `_temp.x`, which is created when you build the model in FSDS with the **Delete Temp Files** option turned off). Also note that when importing from the .X file, no part or material which already exists in the KFG file is overwritten (see section 9.5 for more details).

13. ***Final name and path of model (MDL) file (Interior Model KFG):*** This will be the location of your final, compiled FSX VC model. Normally you will point this to your aircraft's model folder, so that it will load in the simulator.

14. ***Advanced… (Interior Model KFG):*** This opens up the advanced options for the interior model build (see the description of the advanced options interface below).

15. ***DrawCallMonitor (Interior Model KFG):*** If you have defined a location for DrawCallMonitor and a final build of your model exists, this will open DrawCallMonitor and automatically load your model in it to show a performance report.

16. ***Audit (Interior Model KFG):*** This will produce an audit report of the interior model. For more information on what an audit produces, see section 10.2 below.

17. ***Build interior (Interior Model KFG):*** This will tweak and compile your VC model, and the resulting build will be placed in the target folder you have specified. If you have turned on 'apply reference parts to aircraft.cfg' and there are suitable parts in the model, these will be merged into your `aircraft.cfg` file. Please note that this will ***not*** make use of a base partial aircraft.cfg file, even if one is defined (see point 19 below for an explanation). This button will be disabled if all the settings for the project are not valid (one or more of the textboxes will be red indicating the problem).

18. ***Target aircraft.cfg file to modify:*** This specifies the `aircraft.cfg` file to add the new reference parts to. Note that this will only occur if the 'Apply reference parts to aircraft.cfg' option is turned on.

19. ***Partial aircraft.cfg to use as base:*** The `aircraft.cfg` modification code in FSDSxTweak is only able to modify or add information to an `aircraft.cfg` file; it cannot remove existing information. During the aircraft development process, there is a good chance that incorrect data will be written to the file (for instance, a camera definition which is no longer needed). It is therefore safer to separate out the `aircraft.cfg` sections which you modify by hand, from those which are automatically updated by FSDSxTweak, and then bring them together for use in the aircraft. This option lets you have a partial `aircraft.cfg` file (which contains only the sections of the file which you create by hand), which is automatically merged with the reference parts in your model, and written to your aircraft folder. Note that the partial `aircraft.cfg` is used only with the **Build all models** command, and not with the **Build Exterior** or **Build Interior** commands.

20. ***Build all models:*** This build first the interior and then the exterior models. If a partial aircraft.cfg has been specified, it will apply all the new reference parts to that partial (first the parts defined in the exterior model, then the parts defined in the interior model). This button will be disabled if all the settings for the project are not valid (one or more of the textboxes will be red indicating the problem).

The settings interface (accessible through the menu: **Options->General Settings**) contains the following features:

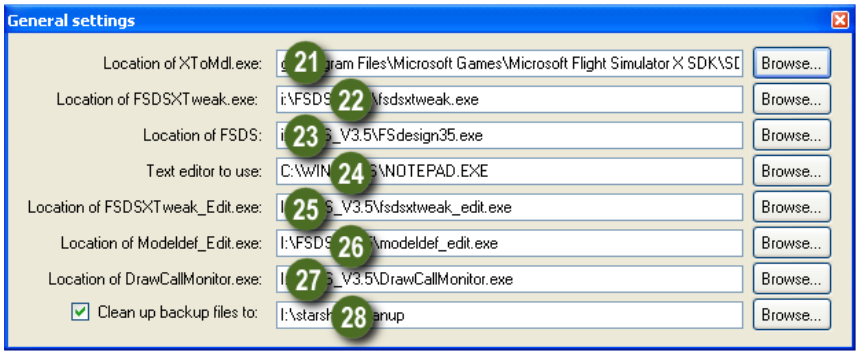


Figure 4: FSDSxTweak_Plugin settings interface

21. ***Location of XtoMdl.exe:*** Point this to the location of the mdl compiler – this usually resides in `C:\Program Files\Microsoft Games\Microsoft Flight Simulator X SDK\SDK\Environment Kit\Modeling SDK\3DSM7\Plugins\XToMdl.exe`

22. ***Location of FSDSxTweak.exe:*** Point this to the location of fsdsxtweak.exe – this usually resides in your FSDS folder.

23. ***Location of FSDS:*** Point this to the location of FSDS – this usually resides in `c:\program files\abacus\fsds_v3.5\`

24. ***Text editor to use:*** If you wish to edit the dictionary (modeldef.xml) files without using Modeldef_Edit, you need to point this to the location of a text editor on your system. The best bet is to point it to notepad.exe, which usually is `c:\windows\notepad.exe`

25. ***Location of FSDSxTweak_Edit.exe:*** Point this to the location of fsdsxtweak_edit.exe – this usually resides in your FSDS folder.

26. ***Location of Modeldef_Edit.exe:*** If you are going to be editing custom animated parts, you will need to edit the part dictionary (usually this is `modeldef.xml`). Modeldef_Edit makes this process easier than using a text editor. Modeldef_Edit will be called when you click on the Edit button associated with the dictionary on the main interface. Point this to the location of modeldef_edit.exe – this usually resides in your FSDS folder (if you do not want this functionality, you can leave this field blank).

27. ***Location of DrawCallMonitor.exe:*** If you want a reliable estimate of your model's performance, you should consider downloading DrawCallMonitor by Arno Gerretsen, which reports the number of DrawCalls made by your model. FSDSxTweak_Plugin can call DrawCallMonitor directly to give you this report. Once you have downloaded (from `http://www.fsdeveloper.com/forum/showthread.php?t=10953`) and installed DrawCallMonitor, Point this to DrawCallMonitor.exe (if you do not want this functionality, you can leave this field blank). Please note that this functionality requires Version 1.1 or later of DrawCallMonitor.

28. ***Clean up backup files to:*** Turn this check box on if you want to clean up the backup files generated by FSDS on compilation. When compiling an aircraft, FSDS normally backs up the aircraft.cfg and model files, in your actual aircraft folder. If you are making a lot of builds of your model, you will find that quickly your aircraft folder fills up with backups,

which can be awkward. By selecting this option (and giving it a folder to which it can move these backups), you can have FSDSxTweak_Plugin automatically move the backups to the folder you specify when you build a model. Note that even if you turn off this feature, you can still force a once-off cleanup of backups by selecting **Clean Backups** in the **Tools** menu (see below).

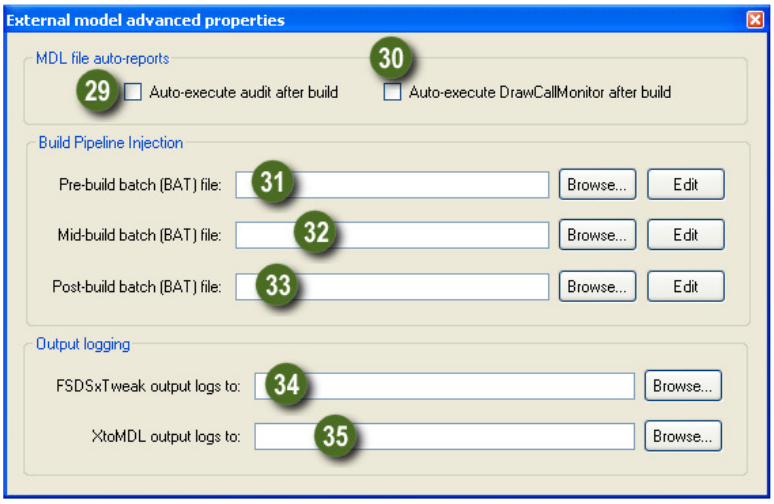


Figure 5: FSDSxTweak_Plugin advanced KFG settings interface

The advanced build options interface (accessible through the **Advanced…** button in the interior and exterior KFG sections of the main interface) contains the following features (note that the interior and exterior models have separate advanced options settings):

29. ***Auto-execute audit after build:*** This will run an audit automatically when the model build is completed.

30. ***Auto-execute DrawCallMonitor after build:*** This will run DrawCallMonitor with your model loaded when the model build is completed.

31. ***Pre-build batch (BAT) file:*** If you need to add to the build pipeline by introducing some other step (copying files, running some other tool etc), then this option allows you to execute your own batch file before `_temp.x` or `_temp_interior.x` is run through fsdsxtweak.

32. ***Mid-build batch (BAT) file:*** This allows you to run a batch file which will execute after fsdsxtweak is run, but before XtoMdl is run. Note that after running fsdsxtweak, the .x file which is compiled is called `_exterior.x` or `_interior.x`.

33. ***Post-build batch (BAT) file:*** This allows you to run a batch file once the XtoMdl build process is completed.

For more information on batch files, see http://www.computerhope.com/batch.htm and http://www.merlyn.demon.co.uk/batfiles.htm

8.1 *A typical session with FSDSxTweak_Plugin*

This is a typical scenario illustrating the use of FSDSxTweak_Plugin. On starting a new FSDS project (Let's say a Boeing 787), set up your FSDS viewports with the three-view drawings, and then go to the FSDS **Plugins** menu, and start **FSDSxTweak_Plugin.exe**. Set all the SDK

and tool paths (to XtoMdl, FSDSxTweak, FSDS, Text editor, FSDSxTweak_Edit, the dictionary file for the internal and external models), and a location to clean up backups to.

Now, create a blank tweak file for both your internal and external models (do this even if you do not plan to tweak the file – an empty tweak file will make no changes, and chances are good you will end up with tweaks in both your models). To do this, enter the name of a new KFG file in the **FSDSxTweak (KFG) file to use** slots for both models. In this example, I will use `c:\program files\abacus\fsds_v3.5\projects\b787_ext.kfg` and `c:\program files\abacus\fsds_v3.5\projects\b787_int.kfg`. Also, set up your target `aircraft.cfg` location, and your target folder for your models. Now, save this config in your projects folder, and call it b787.xcf. Close FSDSxTweak_Plugin. Next time you open it, it will remember all your settings. Begin to build your model as always.

When you come to a place where you want a tweak (for instance, you want to add a bump map to a material on your external model), simply run FSDSxTweak_Plugin.exe from the plugins menu again, click **Edit** next to the external KFG filename box, and enter your tweak using FSDSxTweak_Edit. Once you have entered the tweak, save the file in FSDSxTweak_Edit, and close it. Rinse and repeat! If you want to save yourself the hassle of remembering the names of parts or materials you want to tweak, use the 'Import from .X file' option to automatically take all reference parts and/or materials into FSDSxTweak_Edit.

Come time to test your model, compile your model as normal in FSDS, but be sure to TURN OFF the **Delete Temp Files** option in the *Flight Model Selection* window. Once the compilation is finished, open up FSDSxTweak_Plugin once again, and hit 'Build all models'. This will apply all your tweaks, re-build the models, and copy them to the correct location. You can then run FSX to check your progress. For more detail in this build procedure, see the tutorial in section 14.1.

9. FSDSxTweak_Edit user's guide

This tool can be launched via the **Edit** buttons associated with KFG files in FSDSxTweak_Plugin, or it can simply be launched as a normal application by double-clicking its icon (if you have selected the **Associate KFG files with FSDSxTweak_Edit** option in FSDSxTweak_Plugin – see point 23 in section 8). It is used to write the KFG files which store the tweaks for the interior and exterior models. The interface is shown in Figure 7.

9.1 *Tweak categories*

FSDSxTweak_Edit allows you to tweak many aspects of an aircraft project. Some of the tweaks are applied to the .X file, and some are applied to the *aircraft.cfg* file. FSDSxTweak_Edit makes this process transparent by providing all the tweakable options as a set of categories in a tab control. All the tweakable categories are shown in Figure 6.

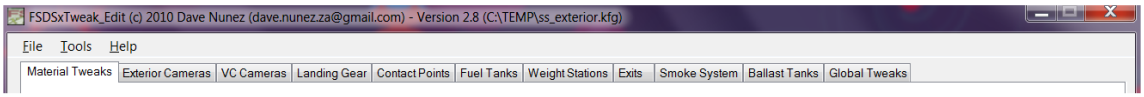


Figure 6: Tweak categories available in FSDSxTweak_Edit

Note that there are some new reference parts which do not appear in the FSDSxTweak_Edit interface, as these have no parameters to set. These are engines, helicopter specific parts (`_liftaerocenter`, `_mainrotor` and `_secondaryrotor`) and anemometers. See sections 12.5, 12.13 and 12.15 for more information.

9.2 *Adding new tweaks*

With the exception of globally applied tweaks (see section 9.4), all tweaks in FSDSxTweak_Edit are added to category lists (each category has its own tab in the main window). There is a list of material tweaks, gear point tweaks, fuel tank tweaks, and so on. You do not need to add tweaks to every list – simply add the ones you need, and leave the rest empty.

In each category, you see the same basic interface layout (see Figure 7). On the left is the list of existing tweaks, with buttons to add a tweak, remove the currently selected tweak, or edit the currently selected tweak. On the right is the interface for editing the tweak (which is different for each category, although all have an OK and Cancel button). To add a new tweak, click the **Add a new <whatever category you are working with>** button. A new tweak will then appear in the list, with an automatically generated name. You can then select that tweak on the list and click **Remove** to delete it (you will get an 'are you sure prompt' to confirm), or **Edit** to set the specific parameters you need. Note that until you select a tweak and click **Edit**, the interface on the right will remain greyed out and inactive. For details on what each parameter does, please consult the relevant section of the FSX SDK.

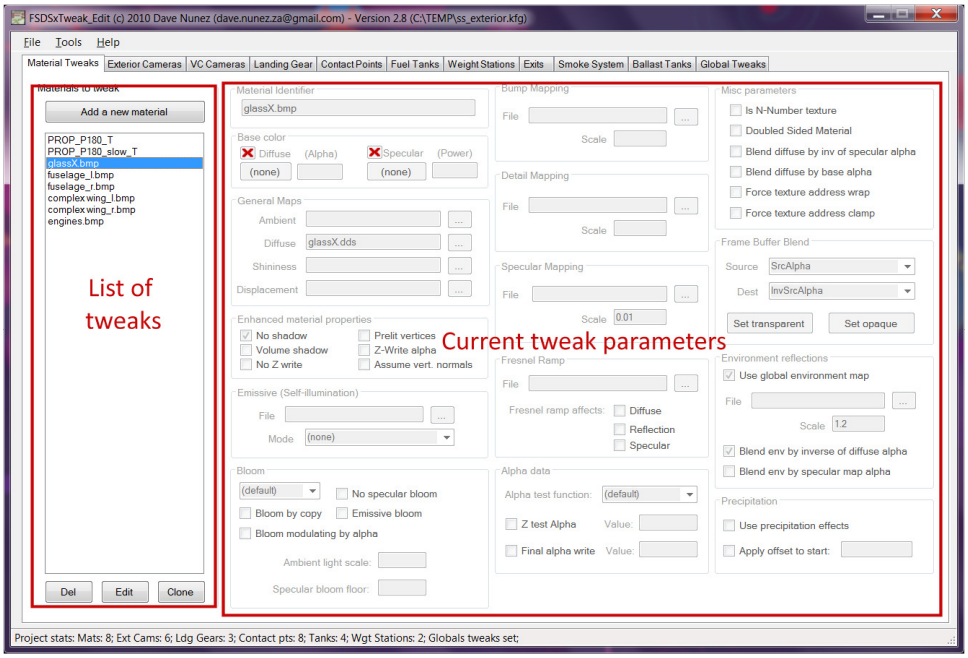


Figure 7: Basic layout of FSDSxTweak_Edit interface

While you are editing parameters, the list on the left and its associated buttons will be greyed out and made inactive, and so will the **File** menu. You must finish editing the current tweak before you can use those functions, either by pressing **OK**, which will incorporate your edits into the tweak, or pressing **Cancel**, which will scrap all the edits for this tweak since you last pressed the **Edit** button.

Note that you do not need to set all the values in the interface for each tweak; in most cases, you will only set a couple of the available options for each tweak (for example, although a material tweak can incorporate more than a dozen options, you need only apply those which you need). There is only one constraint in editing – each tweak must have a unique name (e.g. you cannot have two materials called *glass.bmp*, or two gear points called *gear.0*). FSDSxTweak_Edit automatically checks for any name clashes and will warn you of any conflicts as you go.

9.3 *Naming new tweaks*

Different categories of tweak have different naming styles. There are three patterns of naming:

1. *Freeform names* – this applies to materials. A new tweak of this type can have any name (materials are named after the identifier bitmap in your model you wish to operate on, e.g. fuselage.bmp). These can be up to 128 characters long, and can have any form you wish, provided they are unique in this KFG file.

2. *Numbered names* – this applies to cameras, VC cameras, landing gear points, contact points, weight stations, exits and smoke emitters. A new tweak will always have the name <category>.NUMBER, where you are free set NUMBER to any (positive integer) number (it does not have to start at any particular number, and the different numbers do not have to follow in sequence from each other). You will notice that in the interfaces to edit these categories, there is a textbox to set

number; this will automatically be appended to <category name> when you click **OK**. For example, if you create a new camera, the automatically generated name will be something like *camera.0* - You can edit the number parameter to 7, and on clicking **OK**, you will see on the tweak list that the tweak has been named *camera.7*.

3. *Specially named tweaks* – this applies to fuel tanks and ballast tanks. Some tweaks have special names defined by the SDK (e.g. fuel tanks must be named as per the table in section 12.11). For this type of name, the interface on the right will provide a drop-down list of legal names, and will warn you of any naming conflicts when you click **OK**.

9.4 *Globally applicable tweaks*

Tweaks are not associated with any particular reference part or material are called *globally applicable*. These are the ShadowMapReady and UnitsMetric switches (found on the **Global Tweaks** tab), the *global gear settings* which do not apply to any particular gear point (found on the **Landing Gear** tab), and the *global contact settings* which do not apply to any particular contact point (found on the **Contact Points** tab).

You can set these points independently of any other tweaks. For example, you can set some of the *global contact settings* in your KFG file even if you have not defined any contact points in your model.

9.5 *Importing parts and materials from a .X file*

This feature was introduced in version 2.1. To import, go to the **File** menu, then select **Import/Merge from X file…** and then select if you wish to import reference parts or materials (if you want to import both, you will need to do this twice, one for each). An import allows you to give FSDSxTweak_Edit a .X file which it will scan for either material names (the texture .BMP filenames you defined in FSDS) or the new reference part names which can be tweaked (see section 0). Once the scan is complete, FSDSxTweak_Edit will add the items it finds to the appropriate categories. The addition is a non-destructive merge; that is, if a part or material is found in the KFG file with the same name, the existing item will not be overwritten. It is therefore safe to always import into your KFG file, even if you have done work already. Note however that the import only *adds* to the KFG file; if you delete a reference part or material in your model which you have previously entered into your KFG file, a later import will not remove that part. It is generally safe to leave such 'ghost parts' and 'ghost materials' in your KFG file, as FSDSxTweak is smart enough to ignore any such parts it finds.

The import can be done either by the menu option described above, or by selecting the appropriate option in the FSDSxTweak_Plugin interface (see points 25-28 in section 8).

9.6 *Helper tools in FSDSxTweak_Edit*

The FSDSxTweak_Edit tools menu provides several helpers which can be helpful during the construction of tweak files, even if you are editing your tweak files by hand.

Dynamic compression calculator: This tool (Figure 8) simplifies the calculation of the *static compression* and *static-to-dynamic compression ratio* parameters for landing gear definitions in the aircraft.cfg file. If you are making your models in FSDS and running them through FSDSxTweak, these parameters can be automatically calculated (see section

14.6). This tool is intended mostly for those using GMax, or those who want to tweak the parameters by hand.

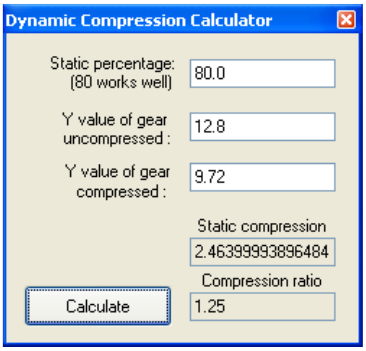


Figure 8: FSDSxTweak_Edit dynamic compression calculator

The *static percentage* value is the percentage of the static (total) compression which the gears display at maximum weight. A value of 80 is a good default.

The *Y value of gear (uncompressed)* parameter represents the Y (vertical) value of the bottom of the gear when there is no weight on the aircraft (this value corresponds to the `_gearu_N` new reference part – see 12.6)

The *Y value of gear (compressed)* parameter represents the Y (vertical) value of the bottom of the gear when there is maximum weight on the aircraft (this value corresponds to the `_gearc_N` new reference part – see 12.6)

Unit converter: This tool (Figure 9) allows for the quick conversion between units. Note that the conversion between Gallons of fuel and pounds of weight uses 6.0 as a conversion factor (which is the value used by FSX).

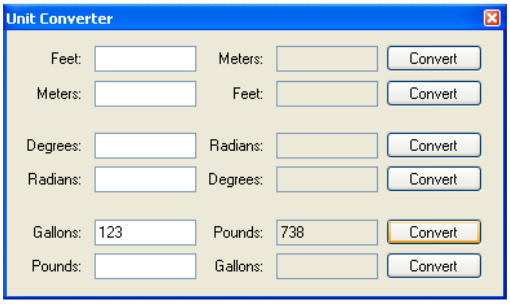


Figure 9: FSDSxTweak_Edit unit converter

GUID generator: This tool (Figure 10) generates FSX compatible GUIDs. Press **Generate GUID** for another value.

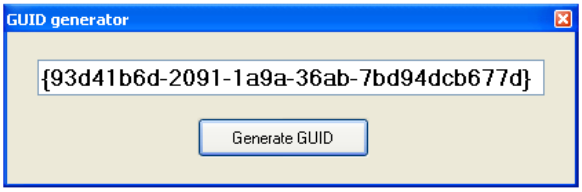


Figure 10: FSDSxTweak_Edit GUID generator

9.7 *Hybrid editing – using FSDSxTweak_Edit and a text editor on a KFG file*

It is possible to switch between editing a KFG file with a text editor and FSDSxTweak_Edit, provided that the hand-edits do not include any illegal section or parameter names (which would not compile with FSDSxTweak anyway). If you are doing a hybrid edit, be particularly careful of spelling errors in special SDK names (such as the names of fuel tanks, or the names of material blend modes). Illegal names will lead to unpredictable behavior in FSDSxTweak_Edit.

10. FSDSxTweak user's guide

This tool does the actual work of processing the .X files, and is operated using either command line parameters, or FSDSxTweak_Plugin. It is strongly recommended that you launch it from FSDSxTweak_Plugin, as this will prevent you from having to type in a lot of arguments.

If you would like to operate FSDSxTweak yourself, then this section provides a list of commands and arguments. Double-clicking on the icon or launching it with no arguments brings up a reminder of the arguments (Figure 11). FSDSxTweak runs in two modes: Tweak Mode, which processes .X file and tweaks it, and AUDIT mode, which provides an audit of a .X file.

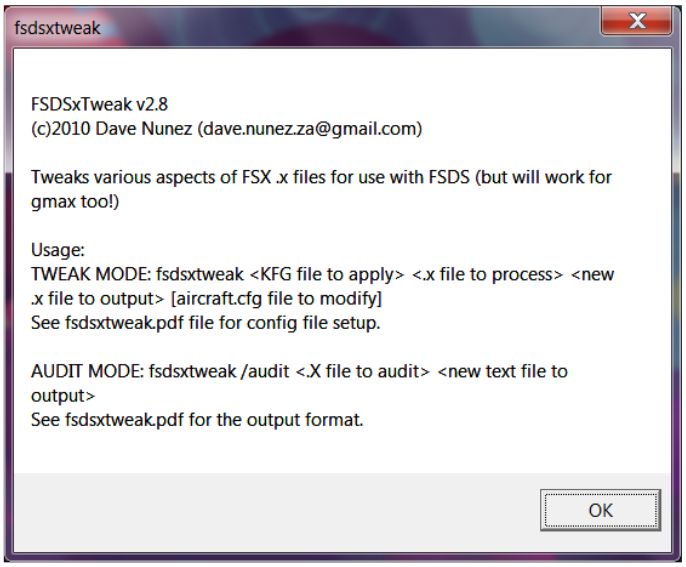


Figure 11: FSDSxTweak launched with no arguments

10.1 *Using Tweak mode*

Tweak mode is the default mode. In tweak mode, FSDSxTweak takes the following arguments, in the following order:

KFG file to apply: This is the KFG file which defines the tweaks to apply to the input .X file. For a description of how to create KFG files, see sections 9 and 13.

.X file to process: This is the path to the input .X file which will be tweaked. Note that this file is NOT overwritten or modified in any way.

New .X file to output: This is the path to the new .X file which will be created after tweaking. It is effectively a copy of the input .X file, except that it is tweaked in the ways defined by the KFG file. Note that if the file provided already exists, it will be overwritten with no prompt.

Aircraft.cfg file to modify: *Note: This is an optional argument*. If this argument is set, and the .X file contains parts named with the new reference part names (see section 0), then these parts will not be processed into the model, but rather will be injected into the `aircraft.cfg` file provided.

Tweak mode will parse the .X file, and outputs (to the stdout console) feedback about the tweaking process, including error messages where appropriate. It does not return an error or success code.

10.2 *Using audit mode*

To enter audit mode, the argument `/audit` must be passed as the first argument, followed by these two arguments:

> ***.X file to audit:*** This is the path to the .X file to be audited.
>
> ***New text file to output:*** This is the path to the text file which will be created with the audit results. Note that if this file already exists, it will be overwritten without any prompt.

The audit produces a text document which lists the texture files used by the .X file, together with usage statistics for each texture file. A typical audit output is shown in Figure 12. The audit provides three pieces of information:

List of maps used: This is a list of all the bmp and dds files which the .x file actually refers to. This is useful to package up your completed models, as it helps you ensure that you don't forget to include any textures, and also that you do not include any textures which your model does not actually use. You could also use the output to build a batch file for mass-processing your textures through a bmp format utility (like imagetool.exe).

Number of references per map: This lists all the textures as above, but also reports how many times each file is actually referenced. The number shown is not exactly how many parts make use of the texture; it could be higher if you have a part which uses more than one texture. This report is useful in evaluating performance issues (i.e. frame rate), and gives a hint about how to improve performance. Read Adrian Wood's blog posts on how performance in FSX is most affected by having more materials (rather than more polygons, as in the old days):

http://blogs.technet.com/torgo3000/archive/2007/06/01/performance-art.aspx

http://blogs.technet.com/torgo3000/archive/2007/06/14/performance-art-2-when-is-a-draw-call-not-a-draw-call.aspx

http://blogs.technet.com/torgo3000/archive/2007/06/19/performance-art-3-polygons-don-t-matter.aspx

By examining this section of the audit, you can determine which textures are being used only a few times, and try to optimize that by combining them into a single larger texture sheet (which reduces the number of materials).

Performance warnings: This is a list of the least used materials; specifically, it lists the bottom 2.5% in terms of references. This information is used in the same way as the number of references per map data, but this time you are shown only the bottom 2.5% of materials. The 2.5% number is just a rule-of-thumb I use, which seems to work well. Depending on your model, you should try to combine the materials shown in this list into single larger texture sheets to reduce the number of materials you use.

```
//The file 'F:\FSDS_V3.5\_exterior.x' uses 24 maps:
lights.bmp
engines.bmp
envmap.bmp
complex wing_r.bmp
complex_wing_r_bump.dds
complex wing_l.bmp
complex_wing_l_bump.dds
s2000_interior.bmp
fuselage_l.bmp
fuselage_r.bmp
cabin1.bmp
cabin instruments.bmp
wing_tl.bmp
sailcanard_li.bmp
sailcanard_lo.bmp
sailcanard_ri.bmp
sailcanard_ro.bmp
chrometube.bmp
s2000_tyre2.bmp
s2000_tyreb.bmp
pilot.bmp
glassX.dds
PROP_P180_T.bmp
PROP_P180_slow_T.bmp

//Number of references per texture (average 15.38 refs/map):
    5:lights.bmp
   10:engines.bmp
  137:envmap.bmp
   37:complex wing_r.bmp
   37:complex_wing_r_bump.dds
   38:complex wing_l.bmp
   38:complex_wing_l_bump.dds
   11:s2000_interior.bmp
    7:fuselage_l.bmp
    5:fuselage_r.bmp
    1:cabin1.bmp
    3:cabin instruments.bmp
    5:wing_tl.bmp
    3:sailcanard_li.bmp
    3:sailcanard_lo.bmp
    3:sailcanard_ri.bmp
    3:sailcanard_ro.bmp
    7:chrometube.bmp
    4:s2000_tyre2.bmp
    4:s2000_tyreb.bmp
    1:pilot.bmp
    5:glassX.dds
    1:PROP_P180_T.bmp
    1:PROP_P180_slow_T.bmp

//PERF WARNING: Bottom 2.5% of maps by reference (consider consolidating some of these into a
single map):
    5:lights.bmp
    7:fuselage_l.bmp
    5:fuselage_r.bmp
    1:cabin1.bmp
    3:cabin instruments.bmp
    5:wing_tl.bmp
    3:sailcanard_li.bmp
    3:sailcanard_lo.bmp
    3:sailcanard_ri.bmp
    3:sailcanard_ro.bmp
    7:chrometube.bmp
    4:s2000_tyre2.bmp
    4:s2000_tyreb.bmp
    1:pilot.bmp
    5:glassX.dds
    1:PROP_P180_T.bmp
    1:PROP_P180_slow_T.bmp
```

Figure 12: Sample output of FSDSxTweak file audit

11. Modeldef_Edit user's guide

Modeldef_Edit allows you to easily edit the modeldef.xml file which contains the data required to animate model parts in FSX. The standard FSX dictionary contains hundreds of parts, and it is only necessary to use Modeldef_Edit if you wish to change an existing part, or add a prt of your own.

Each part in the dictionary has a unique name (e.g. `r_flap_key`), and one or both of an **Animation block** and a **PartInfo block**. The Animation block is contained in an `<Animation>` XML tag, and indicates that the part is an animation. The PartInfo block is contained in a `<PartInfo>` XML tag, and contains the code required to animate and/or control the visibility of the part. It is important to note that if a particular part has both an Animation and a PartInfo block, the `<Name>` and `<Length>` parameters in these two blocks must tbe the same. Furthermore, if a part has an Animation block, that block must contain a GUID which is unique in the dictionary. For more information on these parameters and part dictionaries in general, see the *Using Modeling Tools* section of the SDK.

11.1 *Installation notes*

Before you do any editing with Modeldef_Edit, be sure that the SDK lookup files provided with this download (`eventids.txt`, `helpids.txt`, `units.txt` and `variables.txt`) are in the same folder as Modeldef_Edit. Also, open the settings (Menu: **Options ->Settings…**) and ensure that you configure the program as you need. **IMPORTANT NOTE FOR FSDS USERS:** If you are using FSDS, you must turn the **Auto Update FSDS with new parts** option on, and enter the location of your FSDS installation. Failure to do so could lead to horrible XtoMdl compilation errors.

11.2 *Modeldef_Edit interface*

The interface has two panes side by side – the left pane shows the list of parts in the animation, a tool to search/filter for a particular part, and the set of possible actions for a part selected in the list. The right pane shows the parameters for the currently selected part (see Figure 13).

Modeldef_Edit can be used as a quick browser of a modeldef.xml file by simply selecting the a part in the left pane, and examining the parameters on the right. Figure 14 below shows the details of the main window

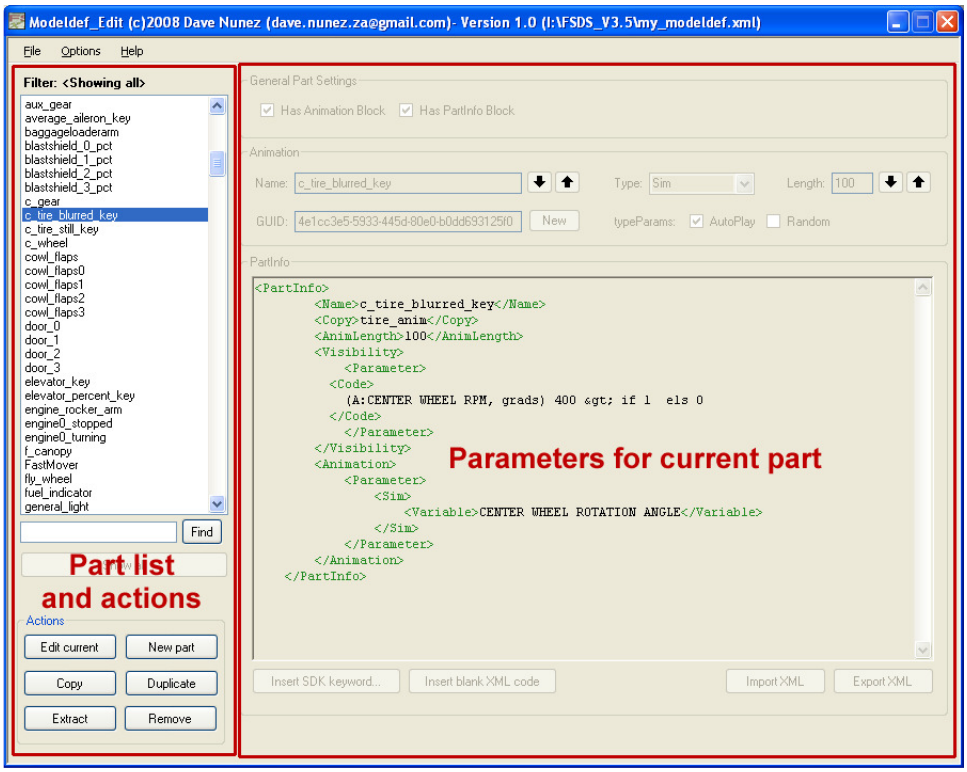


Figure 13: General layout of the Modeldef_Edit interface

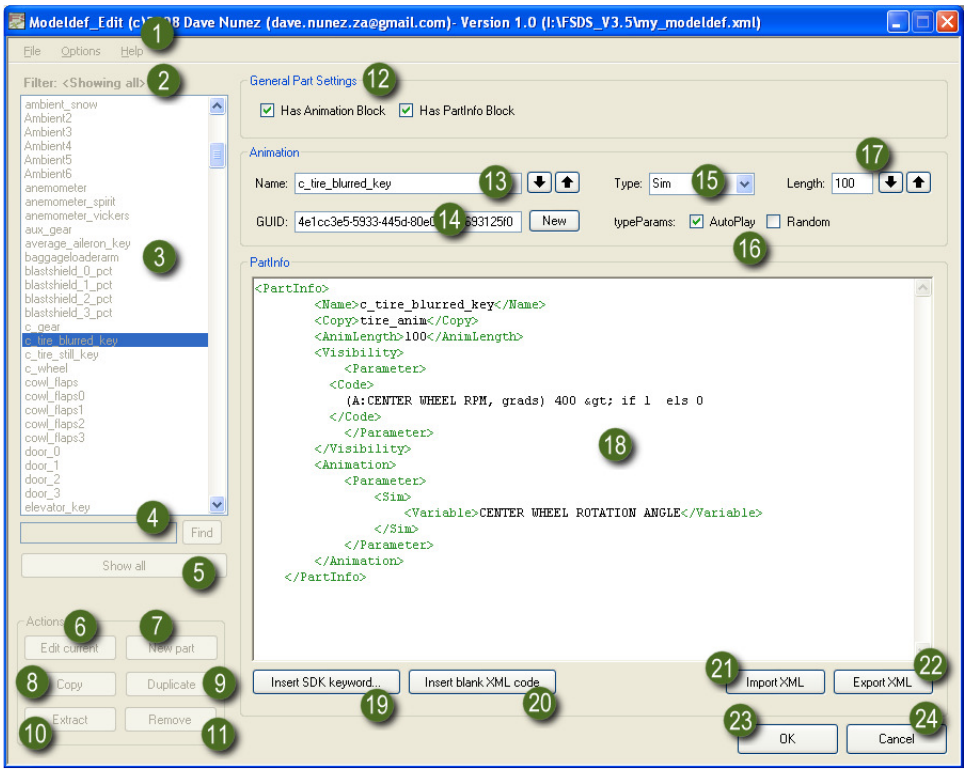


Figure 14: Modeldef_Edit interface main window

1.***Menus:*** The following features are available in the menu:

File menu: The following is available in this menu: **Open**, **Save**, **Save As** and **Exit**.

Options menu: The **Settings…** feature opens the general settings:

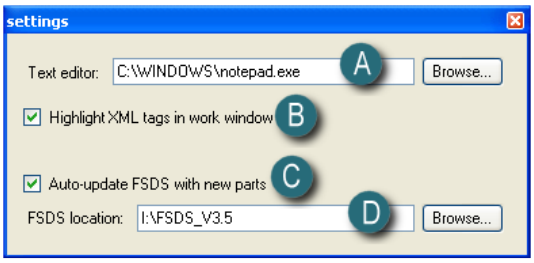


Figure 15: Modeldef_Edit settings window

A: Enter here the location of a text editor, which will be used when extracting a part (see10 below). If you have no preference, use `C:\WINDOWS\notepad.exe`

B: Select this to toggle color–coding of XML tags in the XML work window (see18 below).

C: If you are using FSDS to build your models, the changes you make to `modeldef.xml` need to be synchronized into the `partdefsdata.txt` file used by FSDS. Turning this option on will do so automatically each time you save you `modeldef.xml` file. **IMPORTANT NOTE:** FSDS only reads the `partdefsdata.txt` when it starts up, so if you have added a new part and are going to be compiling your model inside FSDS, you need to shut FSDS down and start it up again, or you will get XtoMdl compilation errors.

D: If you are using FSDS, enter here the folder in which FSDS is installed (this is where the `partdefsdata.txt` file will be written).

Help: Allows quick opening of the manual (this document), the development blog, and the Freeflight Design Forum.

2. ***Active filter:*** This shows what filter/search is being applied to the displayed list. When this displays `<Showing All>`, the list shows all parts in the modeldef.xml file. The filter can be changed using features 4 and 5 described below.

3. ***List of parts:*** This shows the list of parts which fit the active filter, or all parts is no filter is active. If no parts in the modeldef.xml file match the active filter, this list will be empty.

4. ***Filter entry box:*** This allows you to enter a filter to apply to the part list. The filter is a case insensitive substring search – it will show all parts whose name contains the entered filter anywhere in its name, with case being ignored. The **Find** button activates the filter (pressing enter in the text box will do the same). When a filter is active, the entered filter is shown in **active filter** caption above the list (see 2 above).

5. ***Show all (remove filter)***: This will deactivate the filter, showing all parts again. The **active filter** caption (see 2 above) will reflect this by reporting `<Showing All>`.

6. ***Edit current***: This will enable the right hand pane, and allowing editing of the currently selected part. Note that the left panel is ghosted until the use clicks on **OK** or **Cancel** in the right hand pane (see 21 and 22 below).

7. ***New part:*** This will create a new part. The new part is automatically named `NewPart_000` (the second part `NewPart_001`, and so on). Editing mode is automatically activated. Note that the new part will have neither an Animation block nor a PartInfo block turned on (see 12 below). If the user clicks on **Cancel** while editing a newly created part, the new part is automatically deleted and its name is removed from the parts list.

8. ***Copy:*** This creates a new part which is intended to be a copy of the currently selected part. The new part will be named according to the selected part. So if the selected part is `c_gear`, the new part will be named `Copy_of_c_gear_000`. The Animation block of the new part will match the animation block of the selected part, and the PartInfo block will contain the minimum correct XML code for a part copy, which for a copy of c_gear would be:

```
<PartInfo>
	<Name>Copy_of_c_gear_000</Name>
	<Copy>c_gear</Copy>
</PartInfo>
```

Note that as per the SDK, the intention of a part copy is to allow the user to quickly make a part which duplicates an existing part except for small changes (such as responding to a different token variable). If you wish to make a new part which involved major changes to the PartInfo block, make a part duplicate rather than a part copy (see 9 below).

9. ***Duplicate:*** This creates a new part which is an exact duplicate (except for GUID and name) of the currently selected part. The new part will be named according to the selected part. So if the selected part is `c_gear`, the new part will be named `Duplicate_of_c_gear_000`. If the Animation or PartInfo blocks are available, they will be duplicated.

10. ***Extract***: This will extract the XML for the currently selected part (both Animation and PartInfo blocks if available) to the text editor which has been defined in the settings window (see 1 above). The purpose of this feature is to extract out a single part to email or post on a forum easily.

11. ***Remove***: This will remove (delete) the currently selected part from this file. There is no undo, so be careful!

12. ***General Part Settings***: Here you can set whether this part has an Animation or PartInfo block, or both. Ticking a both will enable the appropriate section to edit. Note that when you create a new part (see 7 above), it has neither of these blocks turned on. Normal aircraft animations have both blocks turned on, but if all you want to do is control part visibility (rather than animation), then you do not need the animation block.

13. ***Name***: The unique name for this part. Clicking the **down arrow** will cause the name entered in this box to be copied into the PartInfo block, if it is available. Clicking the **up arrow** will cause the name entered in the PartInfo block (if available) to be copied into

this box. The arrows allow for easily ensuring that the names in both blocks match exactly, which is required by the SDK.

14. ***GUID***: This is the generic universal ID for this part, as required by the SDK. You can generate a new one by clicking the **New** button next to the text box.

15. ***Type***: This is the type of part, `Sim` or `Standard`. Normal aircraft animations are of type Sim. See the SDK for a discussion of these types.

16. ***TypeParams:*** These are the parameters for the Animation. You can legally have both, one or neither of these in an Animation. Normal aircraft animations usually have only **AutoPlay** turned on.

17. ***Length***: The number of frames that this animation plays over. Clicking the **down arrow** will cause the number entered in this box to be copied into the PartInfo block, if it is available. Clicking the **up arrow** will cause the number entered in the PartInfo block (if available) to be copied into this box. The arrows allow for easily ensuring that the numbers in both blocks match exactly, which is required by the SDK.

18. ***XML work window***: This window allows editing the XML associated with the PartInfo block. If the option **Highlight XML tags in work window** is turned on in the settings window (see 1 above), the work window will display XML tags in green, and other text in black. Note that no XML error checking is provided, but if the XML entered is not properly formatted (e.g. a tag is not followed later by a closing tag) then the XML will not be allowed to be entered into the part. Note that you can cut from and paste into the XML work window, so you can easily transfer an XML part you see in a forum (or email) into your modeldef.xml file (see 14.7 below for a tutorial).

19. ***Insert SDK keyword***: This opens the SDK keyword lookup tool, which lets you easily insert important keywords into your XML:

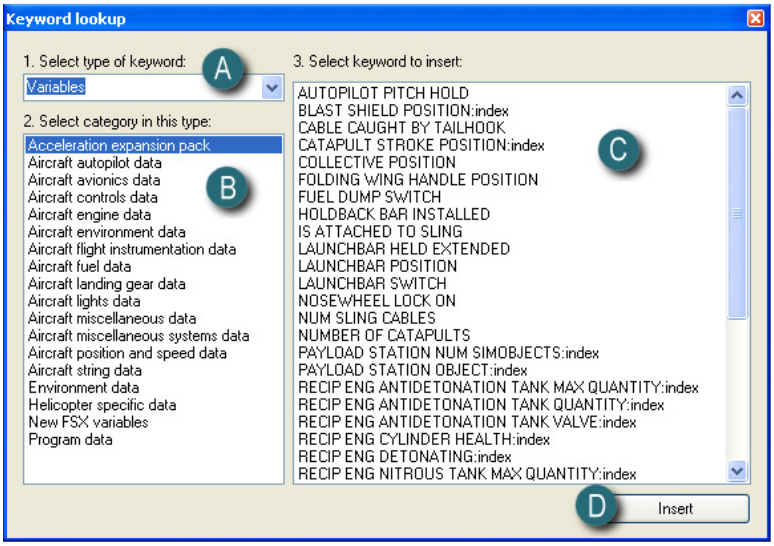


Figure 16: Modeldef_Edit keyword lookup tool

To insert a new keyword, first find the type of keyword (A), then the category (B). Finally select the keyword from the list (C) and click insert (D). The keyword will be entered in the

XML work window at the point where your typing cursor was. The types and categories are taken from the SDK, and the keyword list is sorted alphabetically.

Note that this feature requires the files `eventids.txt`, `helpids.txt`, `units.txt` and `variables.txt` (provided with this download) to be in the same folder as `modeldef_edit.exe`. Also note that the keywords provided here are the most inclusive set (i.e. they include the keywords for the Acceleration pack), so check that you are not using Acceleration only keywords if you are building a model for SP2.

20. ***Insert blank XML code***: This will replace the contents of the XML work window with a minimal set of XML PartInfo tags:

```
<PartInfo>
	<Name>NewPart_000</Name>
	<AnimLength></AnimLength>
	<Animation>
		<Parameter>
			<Sim>
				<Variable></Variable>
				<Units></Units>
			</Sim>
		</Parameter>
	</Animation>
</PartInfo>
```

This allows for quick set up of a new PartInfo block.

21. ***Import XML:*** This will load a text file with XML code into the XML work window.

22. ***Export XML:*** This will save the contents of the XML work window to a text file. Note that because you cannot add incorrect or faulty XML into a modeldef.xml file, you cannot save bad XML with your modeldef.xml. The **Export XML** feature lets you save your XML with errors until you can debug your XML and save it into the modeldef.xml file.

23. ***OK:*** This will attempt to insert the part you are editing into the XML document. If the XML is not properly formatted or some other error occurs, you will be prompted to correct it.

24. ***Cancel:*** This will quit editing and revert the part to its state before you began editing. If you begin the edit with the New Part button (see 7 above), the temporary part is removed from the document.

12. New FSDS reference parts

FSDSxTweak supports adding new spatial information into the `aircraft.cfg` file beyond that which is possible with FSDS. For example, the position of the engines for the `[generalenginedata]` section can be automatically set by creating a reference part in the model called `_engine_0`.

New in version 2.7: Version 2.7 added the ability to attach the following to models:

1. Scenery library objects, via the `_attachbyguid_[GUID]` part
2. Generic attach parts such as carrier catapults, carrier arresting wires, etc. via the `_attachbyname_[attach part name]` part
3. Effects via the `_attacheffect_[effect filename]` part
4. Landable platforms via the `_attachplatform_[platform type]` part
5. NoCrash nodes via the `_nocrash` part
6. MouseRects node via the `_mouserect_[mouserect name]` part
7. Visibility nodes via the `_visibility_[condition name]` part

Details on what these parts do is in the FSX SDK.

12.2 *Creating and naming reference parts*

A reference part in FSDS can be any primitive (block, cylinder, polygon, etc.). I prefer to use a small 3-sided polygon, as it interferes the least with modeling the actual visible geometry. Once you have created the reference part, give it the appropriate name, and then make sure the '***Hierarchy Node (no geometry)***' checkbox is turned on. **NOTE: Do *NOT* use the 'Reference part (not generated)' checkbox** – this will lead to your reference part not making it into the .X file, and will therefore not processed by fsdsxtweak (see the screenshots below for examples).

For classes of parts where multiple instances can exits (for example, there can be multiple cameras in a model), name them by using a number after the special name, starting with zero. So the first camera in your model would be `_camera_0`, the second camera would be `_camera_1`, then `_camera_2`, and so on. There is an exception here however: Fuel tanks are not named with numbers in this way, but with special names which are defined by the SDK (see 12.11 below for details).

NOTE this exception: For the `_attachbyguid_`, `_attachbyname_`, `_attacheffect_`, `_attachplatform_`, `_nocrash`, `_mouserect_` and `_visibility_` parts, do ***not*** check the '***Hierarchy Node (no geometry)***' checkbox – simply leave the part as a default piece of geometry.

12.3 *Oriented and non-oriented reference parts*

Some of the new parts (such as the engines, gear and fuel tanks) require only the position in space; the reference parts are ***non-oriented*** (i.e. you don't care which way the part is pointing, as long as its position is correct. Figure 17 shows a non-oriented part (again, note you must check the 'Hierarchy Node' check box and not the 'reference part' checkbox

For some other parts, however (cameras and some helicopter parts) it is important to also set ***the direction or orientation of the part***. For cameras, for example, it is important to define not only where the camera is (position), but also which way the camera is pointing (its orientation). For oriented parts, you need to rotate your part so that the ***Z axis is pointing in the direction you want.*** For example, for cameras, the direction the Z axis is pointing is, is the direction the camera is looking at – see Figure 18.

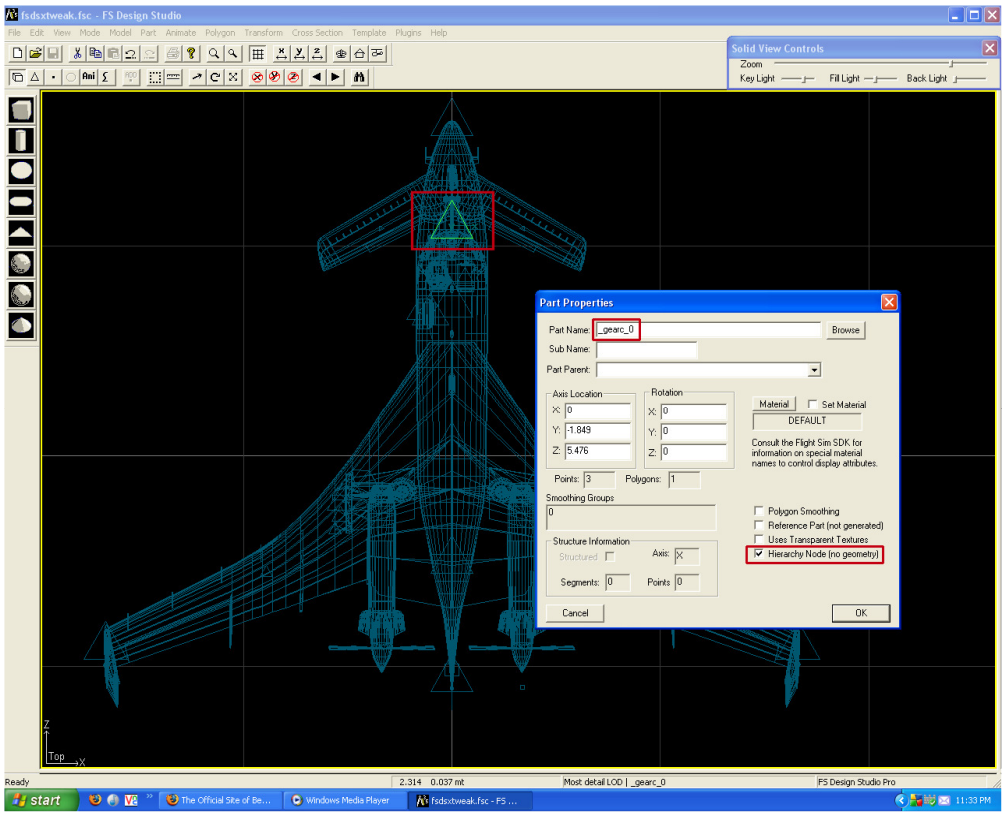


Figure 17: A non-oriented part.

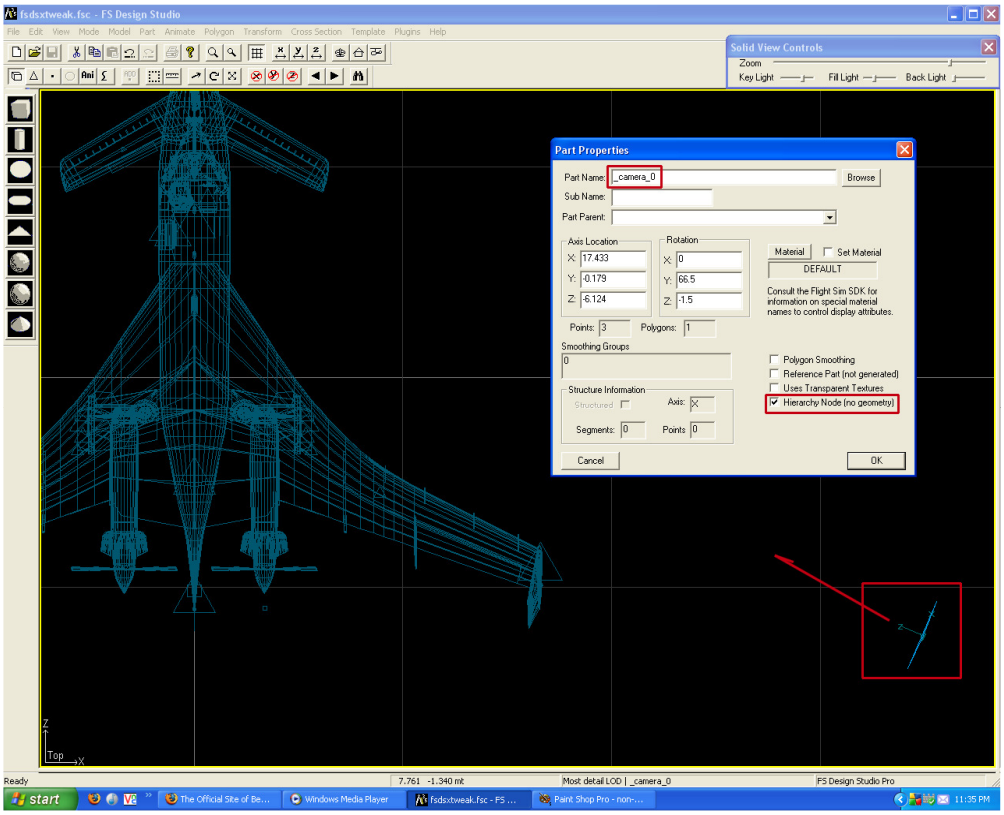


Figure 18: An oriented part.

12.4 *Providing additional information for a reference part*

The new reference part provides the location of the part, which is written into the `aircraft.cfg` file. In most cases, the reference parts can have more information associated with it (for example, for a contact point, one can also add the type of contact point, the impact velocity which causes damage, and the sound to be played, as laid out in the SDK). To add this additional information, you must create a KFG tweak file, which will contain these details. See sections 9 and 13 and the tutorial in section 14.1 for how to do this.

12.5 *Engines*

Special part name: `_engine_0` to `_engine_11`

These parts are written into the `[generalenginedata]` section of the `aircraft.cfg` file, and provide the location of the engine. Note that although FSDSxTweak can create up to 12 engines, FSX actually only recognizes a maximum of four engines in an aircraft.

12.6 *Landing Gear*

Special part names: `_gearc_N` and `_gearu_N`

These parts are written into the `[contact_points]` section of the `aircraft.cfg` file, and provide data about the gear. Each gear should have one `_gearc_N` and one

_gearu_N part; _gearc_N represents the bottom of the gear when fully compressed, while _gearu_N represents the gear completely uncompressed – see the gear compression tutorial in section 14.6 for more information.

12.7 *Contact points*

Special part name: _contact_N

These parts are written into the [contact_points] section of the aircraft.cfg file, and provide data about a non-landing gear contact point (i.e. a scrape point, float, water rudder, etc).

12.8 *Exits*

Special part name: _exit_N

These parts are written into the [exits] section of the aircraft.cfg file, and provide data about the exits of the aircraft, which is used to animate the doors and allow aligning of jetways at airports.

12.9 *Cameras*

Special part names: _camera_N for external cameras, and _vccamera_N for VC cameras

This is an oriented part; Z axis should point along direction of camera view

Each of these parts is written into their own [cameradefinition.N] section in the aircraft.cfg file to define a new camera.

12.10 *Weight/Load Stations*

Special part name: _stationload_N

These parts are written into the [weight_and_balance] section of the aircraft.cfg file, and provide information about a seat or cargo space for loading the aircraft.

12.11 *Fuel Tanks*

Special part name: Unlike other special parts, fuel tanks are not indexed using numbers. Each one has a unique name:

Legal Fuel Tank part names		
_fueltank_leftmain _fueltank_leftaux _fueltank_lefttip	_fueltank_center1 _fueltank_center2 _fueltank_center3 _fueltank_external1 _fueltank_external2	_fueltank_rightmain _fueltank_rightaux _fueltank_righttip

These parts are written into the `[fuel]` section of the `aircraft.cfg` file, and provide the location of a fuel tank. When designing these parts in FSDS, you should place the axis of your fuel tank part in the center of the part (in FSDS: **Part** menu, then **Center Axis to Part**).

12.12 *Smoke emitters*

Special part name: `_smoke_N`

These parts are written into the `[smokesystem]` section of the `aircraft.cfg` file, and define the location of the smoke points on the aircraft.

12.13 *Helicopter specific parts*

Special part names: `_liftaerocenter`, `_mainrotor` and `_secondaryrotor`

`_mainrotor` is an oriented part, with the Z axis pointing forward.

These parts are written into the `[helicopter]`, `[mainrotor]` and `[secondaryrotor]` sections of the `aircraft.cfg` file respectively. What they define is fairly self-explanatory.

12.14 *Water ballast tanks*

Special part name: `_ballast_N`

Note that only 6 ballast tanks may be defined. Each tank number is tied to a location on the aircraft:

Water ballast tank numbers and locations	
0	Front fuselage
1	Rear fuselage
2	Left outboard
3	Left Inboard
4	Right Inboard
5	Right outboard

These parts are written into the `[water ballast system]` section of the `aircraft.cfg` file, and define the location of the ballast tanks. When designing these parts in FSDS, you should place the axis of your ballast tank part in the center of the part (in FSDS: **Part** menu, then **Center Axis to Part**).

12.15 *Anemometers*

Special part name: `_anemometer_N`

These parts are written into the `[anemometer]` section of the `aircraft.cfg` file, and define the location anemometers.

12.16 *Generic attachpoint*

Special part name: `_attachbyname_[attach part]`

These parts cause the special attachment to be generated at the origin of the part. These parts can be used to attach rotating beacons or carrier parts (from the Acceleration SDK) to a model. For example, to attach an arresting cable to a model, create a part called `_attachbyname_attachpt_cable_1_1` at one end of the cable, and another called `_attachbyname_attachpt_cable_1_2` at the other end of the cable. Note that the part itself will not appear on the model.

12.17 *Library object attachpoint*

Special part name: `_attachbyguid_[library object GUID]`

These parts cause the library object specified by the GUID to appear attached at the origin of the part. Note that the part itself will not appear on the model.

12.18 *Effect attachpoint*

Special part name: `_attacheffect_[effect filename]/[optional effect parameters]`

These parts cause the effect to appear at the origin of the part. When specifying the effect to use, use the filename of the effect file, but leave off the .FX extension. For example, to attach the FX_SMOKE_R.FX file to the model, create a part called `_attacheffect_fx_smoke_r` - the effect will appear instead of the part. Optionally you can add paramters (as defined in the FSX SDK) to the effect. If you want to add these optional parameters, add them as described in the SDK after a forward slash (/). Put a semicolon (;) after each parameter, including the final parameter. For example, to make the red smoke not appear at night, name your part like this: `_attacheffect_fx_smoke_r/NIGHT=0;`

12.19 *Platform attachpoint*

Special part name: `_attachplatform_[platform type]`

These parts cause a landable platform to appear at the origin of this part. When specifying this part, you must state the platform type which must be one of the following strings:

Legal platform types for attachment	
ASPHALT	BITUMINOUS
BRICK	CONCRETE
CORAL	DIRT
FOREST	GRASS
GRASS_BUMPY	GRAVEL
HARD_TURF	ICE
LONG_GRASS	MACADAM
OIL_TREATED	PLANKS
SAND	SHALE
SHORT_GRASS	SNOW
STEEL_MATS	TARMAC
URBAN	WATER
WRIGHT_FLYER_TRACK	

Note that the part itself will not appear on the model.

12.20 *Visibility node*

Special part name: `_visibility_[condition name]`

These parts define a visibility condition node, which controls the visibility of all child parts. To control the visibility, give this node a condition name, and then via an XML part in the ModelDef.xml define the visibility condition. Note that the part itself will not appear on the model.

12.21 *MouseRect node*

Special part name: `_mouserect_[mouserect name]`

These parts define a MouseRect which allows manipulation in the VC by the user. To allow user control, give this node a mouserect name, and then via an XML part in the ModelDef.xml define the mouserect. Note that the part itself will not appear on the model.

12.22 *NoCrash node*

Special part name: `_nocrash`

These parts apply to scenery only. They remove the crash detection for all child parts. Note that the part itself will not appear on the model.

13. Manual editing of KFG files

Although it is strongly recommended that you use FSDSxTweak_Edit to edit your KFG files, you may also edit these with a text editor if you wish. If you prefer to edit your KFG files by hand, you can use this section as a reference for the names of KFG sections and parameters to use. Please refer to the FSX SDK topic "Aircraft Configuration Files" for the actual meaning/effect of each parameters (the connection between these parameters and the relevant section in the SDK should be fairly obvious). The parameters are presented in alphabetical order.

13.1 *Creating a KFG file by hand*

KFG files are simply plain text documents with a .KFG extension; you can use Notepad to create them. The file has a format very similar to the `aircraft.cfg` file – the file consists of numerous sections headed by a name in square brackets (e.g. `[gear.0]`), followed by the parameters for that section. Each parameter consists of a parameter name, and equals sign, and the value of that parameter (e.g. `damping_ratio=0.2` or `title=My new camera`). Only one parameter may exist per line of text. See the file `example.kfg` for a sample of a legal KFG file.

A note about section names: For parts which can have multiple instances (such as contact points and landing gears), the section name is listed as, for instance [gear.N] – the .N should be replaced by the number used in naming the actual part. So if you named the part _gearc_2, then your section should be named [gear.2].

13.2 *Global tweaks*

Section name: `[model]`

`UnitsMetric=` *{0 for imperial, 1 for metric} Note – this is required to properly process the new reference parts. Set to 1 if your model is designed using meters.*

`ShadowMapReady=` *{0 = off, 1 = on}*

Automatically set: Nothing

13.3 *Global contact point settings*

Section name: `[contact_global]`

`emergency_extension_type=` *{number: 0 = None, 1 = Pump, 2 = Gravity}*

`gear_system_type=` *{number: 0 = Electrical, 1= Hydraulic, 2 = Pneumatic, 3 = Manual, 4 = None}*

`static_pitch=` *{number; pitch of model during slew mode in degrees}*

`static_cg_height=` *{number; height of model above ground in slew mode in feet}*

`tailwheel_lock=` *{0 = available, 1 = unavailable}*

Automatically set: Nothing

13.4 *Materials*

Section name: `[texture.N]`

`identifier=` *{your texture filename, including .bmp or .dds extension} Note – this is a unique name which exists inside your model, and is used to identify the material for tweaking. This should always be the first entry in each section.*

`AssumeVerticalNormal=` *{0 = off, 1 = on}*

`ambient=` *{.bmp or .dds filename for this map}*

`BlendEnvironmentByInverseDiffuseAlpha=` *{0 = off, 1 = on}*

`BlendEnvironmentBySpecularAlpha=` *{0 = off, 1 = on}*

`bump=` *{.bmp or .dds filename for this map}*

`bump_scale=` *{number, 0 to large, number of times the texture repeats}*

`DestinationBlend=` *{string, one of Zero , One , SrcColor , InvSrcColor , SrcAlpha , InvSrcAlpha , DestAlpha , InvDestAlpha , DestColor , InvDestColor}*

`detail=` *{.bmp or .dds filename for this map}*

`detail_scale=` *{number, 0 to large, number of times the texture repeats}*

`diffuse=` *{.bmp or .dds filename for this map}*

`displacement=` *{.bmp or .dds filename for this map}*

`emissive=` *{.bmp or .dds filename for this map}*

`EmissiveMode=` *{string, one of AdditiveNightOnly , Blend , MultiplyBlend , Additive , AdditiveNightOnlyUserControlled , BlendUserControlled , MultiplyBlendUserControlled , AdditiveUserControlled}*

`environment_level_scale=` *{number, 0-1. 0 = matte, 1=mirror}*

`fresnel=` *{.bmp or .dds filename for this map}*

`FresnelDiffuse=` *{0 = off, 1 = on}*

`FresnelEnvironment=` *{0 = off, 1 = on}*

`FresnelSpecular=` *{0 = off, 1 = on}*

`NoShadow=` *{0 = off, 1 = on}*

`NoZWrite=` *{0 = off, 1 = on}*

`PrecipitationOffset=` *{0 = off, 1 = on} Note: if you have set PrecipitationUse=1, then you can set this to 1 tocreate a time delay between the precipitation starting and the effect occuring.*

`PrecipitationOffsetValue=` *{0 = off, 1 = on} Note: If you have set PrecipitationUse=1 AND PrecipitationOffset=1, then this value (a floating point number) controls how much of a time offset to apply. I am not sure what the legal range of values is here - play with it and see.*

`PrecipitationUse=` *{0 = off, 1 = on}*

`PrelitVertices=` *{0 = off, 1 = on}*

`reflection=` *{.bmp or .dds filename for this map}*

`shininess=` *{.bmp or .dds filename for this map}*

`SourceBlend=` *{string, one of Zero , One , SrcColor , InvSrcColor , SrcAlpha , InvSrcAlpha , DestAlpha , InvDestAlpha , DestColor , InvDestColor}*

`specular=` *{.bmp or .dds filename for this map}*

`SpecularMapPowerScale=` *{number, 0 to large}*

`UseGlobalEnvironment=` *{0 = off, 1 = on}*

`VolumeShadow=` *{0 = off, 1 = on}*

`ZWriteAlpha=` *{0 = off, 1 = on}*

`DoubleSided=` *{0 = off, 1 = on}*

`IsNNumber=` *{0 = off, 1 = on}*

`AllowBloom=` *{0 = off, 1 = on}*

`BloomByCopy=` *{0 = off, 1 = on}*

`BloomModulatingByAlpha=` *{0 = off, 1 = on}*

`EmissiveBloom=` *{0 = off, 1 = on}*

`NoSpecularBloom=` *{0 = off, 1 = on}*

`SpecularBloomFloor=` *{number, 0-1. 0=None, 1=Maximum}*

`AmbientLightScale=` *{number, 0-1. 0=None, 1=Maximum}*

`diffuse_r=` *{red component of diffuse, number, 0-1}*

`diffuse_g=` *{green component of diffuse, number, 0-1}*

`diffuse_b=` *{blue component of diffuse, number, 0-1}*

`diffuse_a=` *{alpha component of diffuse, number, 0-1}*

Note: If you are going to tweak the diffuse color of the material, then you must set all four of the diffuse_ tweaks. Setting only one (e.g. diffuse_r only), will lead to fsdsxtweak.exe skipping the tweak.

`specular_r=` *{red component of specular, number, 0-1}*

`specular_g=` *{green component of specular, number, 0-1}*

`specular_b=` *{blue component of specular, number, 0-1}*

Note: If you are going to tweak the specular color of the material, then you must set all three of the specular_ tweaks. Setting only one (e.g. specular_r only), will lead to fsdsxtweak.exe skipping the tweak.

`BlendDiffuseByInverseSpecularMapAlpha=` *{0 = off, 1 = on}*

`BlendDiffuseByBaseAlpha=` *{0 = off, 1 = on}*

`ForceTextureAddressWrapSetting=` *{0 = off, 1 = on}*

`ForceTextureAddressClamp=` *{0 = off, 1 = on}*

`ZTestAlpha=` *{0 = off, 1 = on}*

`AlphaTestValue=` *{number, 0 to large}*

`AlphaTestFunction=` *{string, one of Never, Less, Equal, LessEqual, Greater, NotEqual, GreaterEqual, Always}*

`FinalAlphaWrite=` *{0 = off, 1 = on}*

`FinalAlphaWriteValue=` *{number, 0 to large}*

Automatically set: Nothing

13.5 *Engines*

There are no settings for this reference part.

13.6 *Landing Gear*

Section name: `[gear.N]`

Legal parameters:

`airspeed_damage=` *{number representing speed, kts}*

`airspeed_limit=` *{number representing speed, kts}*

`brake_map=` *{number: 0 = None, 1 = Left Brake, 2 = Right Brake}*

`class=` *{number: 0 = None, 1 = Wheel, 2 = Scrape, 3 = Skid, 4 = Float, 5 = Water Rudder}*

`damping_ratio=` *{number representing ratio, 0-1}*

`extension_time=` *{number representing time, seconds, 0=fixed gear}*

`impact_damage_threshold=` *{number representing speed, ft/sec}*

`max_static_compression_ratio=` *{number representing Ratio of Maximum Compression to Static Compression, 0-1}*

`retraction_time=` *{number representing time, seconds, 0=fixed gear}*

`sound_type=` *{number: 0 = Center Gear, 1 = Auxiliary Gear, 2 = Left Gear, 3 = Right Gear, 4 = Fuselage Scrape, 5 = Left Wing Scrape, 6 = Right Wing Scrape, 7 = Aux1 Scrape, 8 = Aux2 Scrape, 9 = Tail Scrape.}*

`static_compression=` *{number representing compression}*

`steering_angle=` *{number representing angle, degrees}*

`wheel_radius=` *{number representing radius of wheel, ft}*

`x=` *{longitudinal distance of the point from the origin, ft}*

`y=` *{lateral distance of the point from the origin, ft}*

`z=` *{vertical distance of the point from the origin, ft}*

Automatically set: The `max_static_compression_ratio`, `static_compression`, `x`, `y` and `z` parameters are automatically set (no entry in kfg file required) if the model has parts named `_gearu_N` and `_gearc_N`. If you have these parts and place the parameters in your kfg file, your kfg file parameters will override the values automatically derived from the model.

13.7 *Contact points*

Section name: `[contact.N]`

`class=` *{number: 0 = None, 1 = Wheel, 2 = Scrape, 3 = Skid, 4 = Float, 5 = Water Rudder}*

`impact_damage_threshold=` *{number representing speed, ft/sec}*

`sound_type=` *{number: 0 = Center Gear, 1 = Auxiliary Gear, 2 = Left Gear, 3 = Right Gear, 4 = Fuselage Scrape, 5 = Left Wing Scrape, 6 = Right Wing Scrape, 7 = Aux1 Scrape, 8 = Aux2 Scrape, 9 = Tail Scrape.}*

x= *{longitudinal distance of the point from the origin, ft}*

y= *{lateral distance of the point from the origin, ft}*

z= *{vertical distance of the point from the origin, ft}*

Automatically set: The x, y and z parameters are automatically set (no entry in kfg file required) if the model has parts named _contact_N. If you have these parts and place the parameters in your kfg file, your kfg file parameters will override the values automatically derived from the model.

13.8 *Exits*

Section name: [exit.N]

Legal parameters:

speed= *{number representing opening/closing speed, seconds}*

type= *{number: 0 = Main, 1 = Cargo, 2 = Emergency}*

x= *{longitudinal distance of the point from the origin, ft}*

y= *{lateral distance of the point from the origin, ft}*

z= *{vertical distance of the point from the origin, ft}*

Automatically set: The x, y and z parameters are automatically set (no entry in kfg file required) if the model has parts named _exit_N. If you have these parts and place the parameters in your kfg file, your kfg file parameters will override the values automatically derived from the model.

13.9 *Cameras*

Section name: For external cameras, [camera.N]; for VC cameras, [vccamera.N]. Note that the N numbers are separate for both of these (i.e. both begin at camera.0 and vccamera.0).

Legal parameters:

allowpbhadjust= *{number: 1=on, 0=off}*

allowzoom= *{number: 1=on, 0=off}*

category= *{string; one of Aircraft, AirTraffic, Cockpit, Custom, Outside, Multiplayer, Runway, Scenery, Tower}*

chasealtitude= *{number: relative altitude in feet -1000 to 3000}*

chasedistance= *{number: distance to object in meters; 0 to 30000}*

chaseheading= *{number: relative angle -180 to 180}*

`chasetime=` *{number: seconds 0 to 200}*

`clipmode=` *{string; one of Normal, Minimum, Spot, Tower}*

`cyclehidden=` *{number: 1=on, 0=off }*

`cyclehideradius=` *{number: nautical miles 0 to 100}*

`description=` *{string}*

`guid=` *{string; guid of camera}*. Note: If creating a camera with FSDSxTweak_Edit, the GUID is generated automatically.

`fixedaltitude=` *{altitude in feet, -500 to 30000000}*

`fixedlatitude=` *{latitude in degrees, -90 to 90}*

`fixedlongitude=` *{longitude in degrees, -180 to 180}*

`headingpanrate=` *{ number: rate 0 to 100 }*

`hotkeyselect=` *{ number: 1 to 10 indicating hot key slot }*

`initialxyz_x=` *{ x position of camera, in design units}*

`initialxyz_y=` *{ y position of camera, in design units}*

`initialxyz_z=` *{ z position of camera, in design units}*

`initialpbh_b=` *{bank angle of camera, in degrees}*

`initialpbh_h=` *{heading angle of camera, in degrees}*

`initialpbh_p=` *{ pitch angle of camera, in degrees}*

`initialzoom=` *{number; zoom value}*

`instancedbased=` *{number: 1=on, 0=off }*

`momentumeffect=` *{number: 1=on, 0=off }*

`origin=` *{string; one of Cockpit, Virtual Cockpit, Center, Pilot, Tower, Fixed, WorldObject }*

`panacceleratortime=` *{ number: seconds }*

`panpbhadjust=` *{string; one of None, Ordinal, Swivel, Orthogonal}*

`panpbhreturn=` *{number: 1=on, 0=off}*

`pitchpanrate=` *{ number: rate 0 to 100 }*

`showaxis=` *{string; one of Yes, No, FrontOnly}*

`showlensflare=` *{number: 1=on, 0=off}*

`showpanel=` *{number: 1=on, 0=off}*

`showweather=` *{number: 1=on, 0=off}*

`smoothzoomtime=` *{number; seconds}*

`snappbhadjust=` *{string; one of None, Ordinal, Swivel, Orthogonal}*

`snappbhreturn=` *{number: 1=on, 0=off}*

`targetcategory=` *{string; one of None, AI Planes, Fixed}*

`title=` *{string}*

`track=` *{string; one of None, FlyBy, Track, TrackBank, FlatChase, FlatChaseLocked}*

`transition=` *{number: 1=on, 0=off }*

`xyzadjust=` *{number: 1=on, 0=off}*

`xyzrate=` *{number: speed in meters/second}*

`xyzaccelleratortime=` *{number: seconds }*

`zoompanscalar=` *{number: scalar 0 to 100 }*

Automatically set: The `initialxyz_x`, `initialxyz_y`, `initialxyz_z`, `initialpbh_p`, `initialpbh_b` and `initialpbh_h` parameters are automatically set (no entry in kfg file required) if the model has **oriented** parts named `_camera_N` or `_vccamera_N`. If you have these parts and place the parameters in your kfg file, your kfg file parameters will override the values automatically derived from the model.

13.10 *Weight/Load Stations*

Section name: `[stationload.N]`

Legal parameters:

`name=` *{string}*

`weight=` *{number representing max allowed weight; lbs}*

`x=` *{longitudinal distance of the point from the origin, ft}*

`y=` *{lateral distance of the point from the origin, ft}*

z= *{vertical distance of the point from the origin, ft}*

Automatically set: The x, y and z parameters are automatically set (no entry in kfg file required) if the model has parts named _weightstation_N. If you have these parts and place the parameters in your kfg file, your kfg file parameters will override the values automatically derived from the model.

13.11 *Fuel Tanks*

Section name: [tank.SPECIALNAME]

The section name of a fuel tank includes its special FS name, which is one of center1, center2, center3, leftmain, leftaux, lefttip, rightmain, rightaux, righttip, external1, or external2. So to create parameters for the right tip tank, the section name would be [tank.righttip]

Legal parameters:

usable= *{number representing usable gas, gallons}*

unusable= *{number representing unusable gas, gallons}*

x= *{longitudinal distance of the point from the origin, ft}*

y= *{lateral distance of the point from the origin, ft}*

z= *{vertical distance of the point from the origin, ft}*

Automatically set: The x, y and z parameters are automatically set (no entry in kfg file required) if the model has parts named according to the table in 12.11 above. If you have these parts and place the parameters in your kfg file, your kfg file parameters will override the values automatically derived from the model.

13.12 *Smoke Generators*

Section name: [smoke.N]

effect= *{string; effect file name without .fx extension – use fx_smoke_w for default}*

x= *{longitudinal distance of the point from the origin, ft}*

y= *{lateral distance of the point from the origin, ft}*

z= *{vertical distance of the point from the origin, ft}*

Automatically set: The x, y and z parameters are automatically set (no entry in kfg file required) if the model has parts named _smoke_N. If you have these parts and place the parameters in your kfg file, your kfg file parameters will override the values automatically derived from the model.

13.13 *Ballast*

Section name: `[ballast.N]`

Note that the N values in these section names have a special meaning. N ranges from 0 to 5, and each number represents a specific location on the aircraft. See the table in section 12.14 for the meaning of each.

`capacity=` *{number: tank capacity in gallons}*

`valve_index=` *{number: which valve controls dump on this tank}*

`x=` *{longitudinal distance of the point from the origin, ft}*

`y=` *{lateral distance of the point from the origin, ft}*

`z=` *{vertical distance of the point from the origin, ft}*

Automatically set: The `x`, `y` and `z` parameters are automatically set (no entry in kfg file required) if the model has parts named _ballast_N. If you have these parts and place the parameters in your kfg file, your kfg file parameters will override the values automatically derived from the model.

13.14 *Helicopter parts*

There are no settings for this reference part.

13.15 *Anemometers*

There are no settings for this reference part.

14. Tutorials

14.1 *Your first tweak – setup and adding a new reference part*

Objective: To get the FSDSxTweak suite up and running, and include one of the new reference parts (a fuel tank) to your model so that it is included in the `aircraft.cfg` file automatically.

Needed: One of your FSDS models (this example will use the Jenny_MP.fsc sample model from FSDS).

Procedure: For each project, you need to set up FSDSxTweak_Plugin. In FSDS, click on the **Plugins** menu, and then click on **FSDSxTweak_Plugin.exe.**

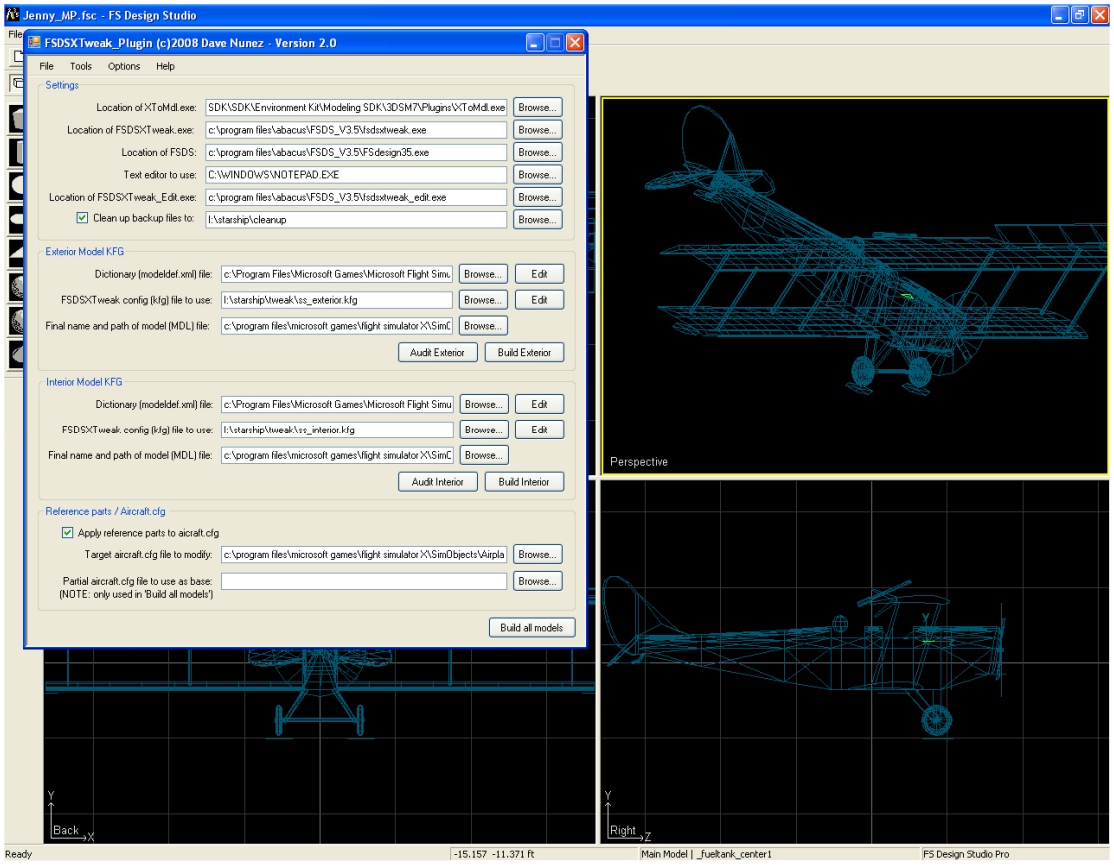


Figure 19: Setting up FSDSxTweak_Plugin for this project

Once you have the window, fill out all the paths to the tools, as well as the target paths. The target paths are where your compiled/modified files will end up. Normally, these will point directly to your aircraft folder in the `SimObjects` folder in FSX. See section 8.1 for more hints on how to fill these in. Once you have completed all the paths (check Figure 19 for a reference), save this configuration in your project folder. Be sure to fill in the Close FSDSxTweak_Plugin, and continue modeling.

Now place your fuel tank in FSDS. A good way to do this is to add a single polygon. Click on the **Part** menu, then **Add** then **Polygon...** Select 3 sides, and radius 0.5 (the perpendicular axis can be anything you like as fuel tanks are not an oriented parts). Click on OK, and the move the part to the correct location. If you are modeling a very large fuel tank (say, on a 747), place your reference part in the center of the tank (so that FSX models the location of the weight correctly).

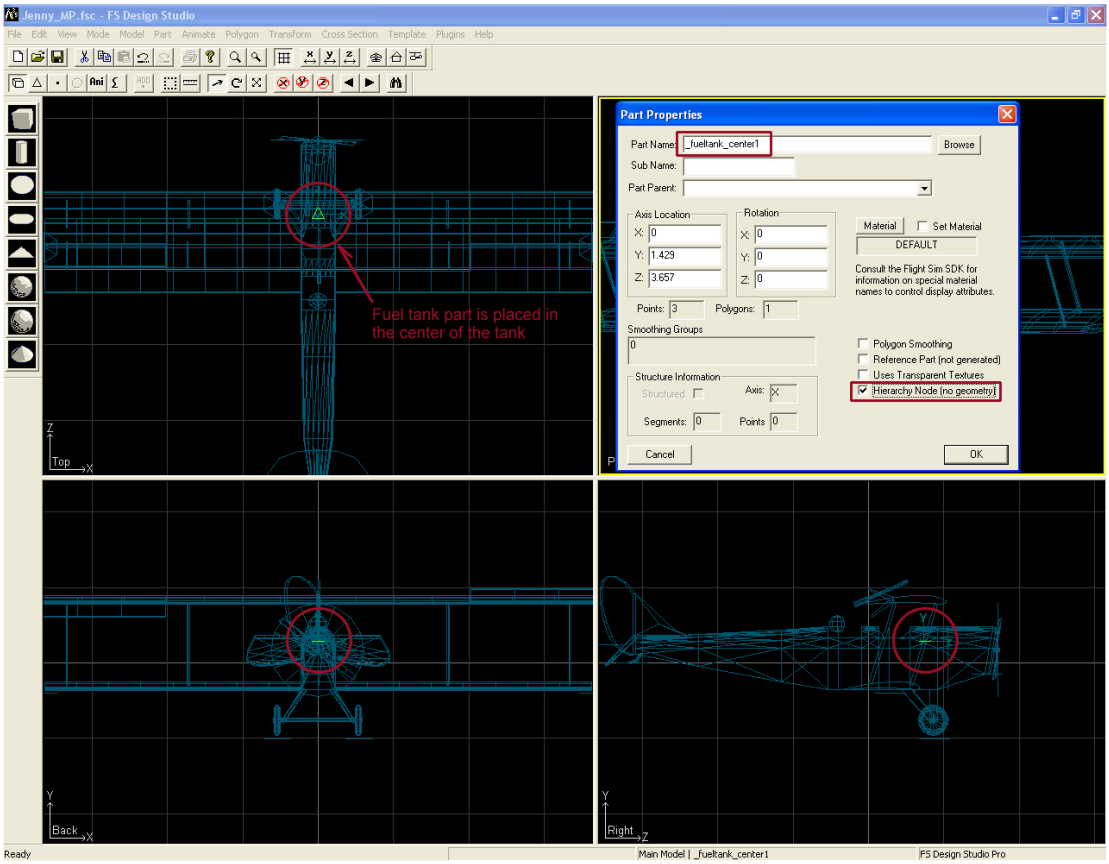


Figure 20: Adding a new reference part in FSDS

Now you need to name and tag the part correctly (see Figure 20). Open up the part properties by pressing F2, and set the name part to be `_fueltank_center1`. Then, turn on the **Hierarchy Node (no geometry)** checkbox. Leave all the other checkboxes off.

So far you have the set the position of the fuel tank, but not its other parameters, such as its capacity. Open up FSDSxTweak_Plugin, then click on the Edit button next to the **FSDSxTweak config (KFG) file to use** line in the **Exterior Model KFG** section. This opens up FSDSxTweak_Edit. Close the project summary mini window, and then click on the **Fuel Tanks** tab (see Figure 21). Click on **Add a new fuel tank**, then select the new tank in the list, and click **Edit**. In the Capacities group set your tank capacities in gallons, and then click on OK. Save the KFG file, and close FSDSxTweak_Edit, and then close FSDSxTweak_Plugin.

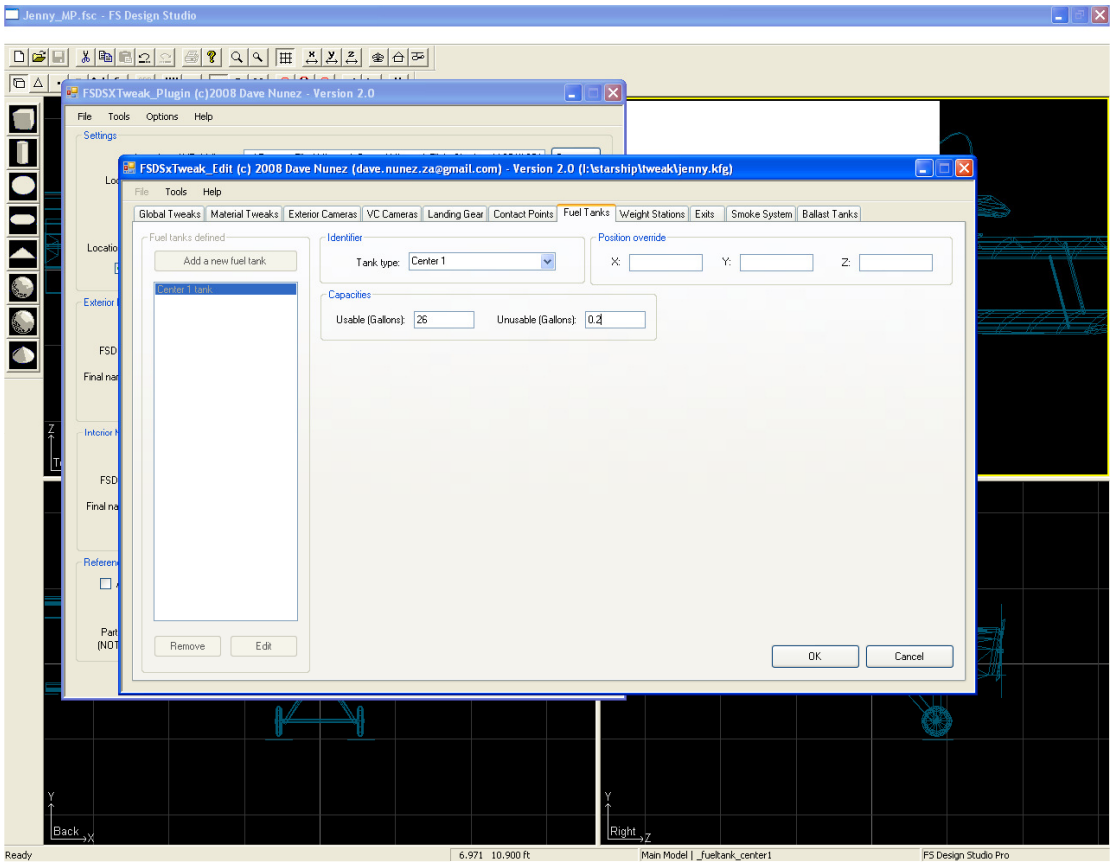


Figure 21: Adding extra parameters with FSDSxTweak_Edit

Now it is time to build and test your model. In FSDS, go to the **File** menu, then **Create FS Object File, then Aircraft File (.mdl)…** In the Flight Model Selection window, choose the model on which to base your own, and be sure to turn OFF the **Delete Temp Files** checkbox. Then click OK, and wait for the build to finish. At this point, you have a fresh build of your model in FSX, but it does not have any of your tweaks applied. To finish applying the tweaks, open up FSDSxTweak_Plugin again, and then click **Build All Models**. Once the builds are completed, your tweaked model is done! You can load your aircraft in FSX and test it.

14.2 *Adding a bump map to a material*

Objective: To add a bump map to a material to simulate rivets, panel lines, etc. This can be done to both your exterior model, and your interior (VC) model.

Needed: Your aircraft model, the base texture, and a bump map which you will apply to your base material. Your bump map must meet the following criteria: (1) it is the same size (in pixels) as your base texture (2) it is saved in DDS format (3) it is a correct bump map rather than a normal map. If you are using CrazyBump (http://www.crazybump.com) to create your bitmaps, you have a normal map and not a bump map – see Bill Leaming's notes on how to convert a normal map into a proper bump map (http://www.fsdeveloper.com/forum/showthread.php?t=7501). If you used The Gimp to create your bump map, the you do not need this correction (see http://davenunez.wordpress.com/2008/07/21/using-the-gimp-to-create-bump-maps-for-fsx/). Figure 22 below shows an example of a base texture (left) and the bump map that goes with it (right). Note that the color of the bump map will be in weird shades of blue and purple –this will ***not*** affect the actual final color of your model.

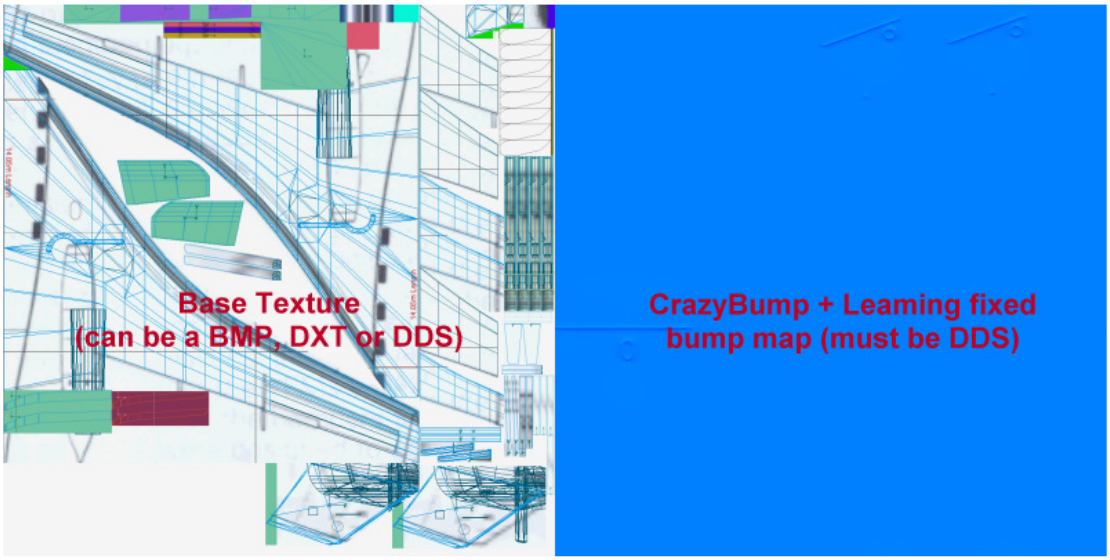


Figure 22: Base texture and corresponding bump map

Required settings: If you are editing your kfg file by hand, add this single switch to your material:

```
bump=your bump map file name.dds
```

You can also set, if you like, a scale parameter for the bump map (the default value is 1.0). Do this by adding the following line to your kfg file (note that 1.5 is purely used as an example, you can have any number larger than 0)

```
bump_scale=1.5
```

See the explanation below for a note on what this scale parameter does.

If you are using FSDSxTweak_Edit, set the following for your material (Figure 23 - only the section in the areas marked red is essential, with the Scale parameter being optional)

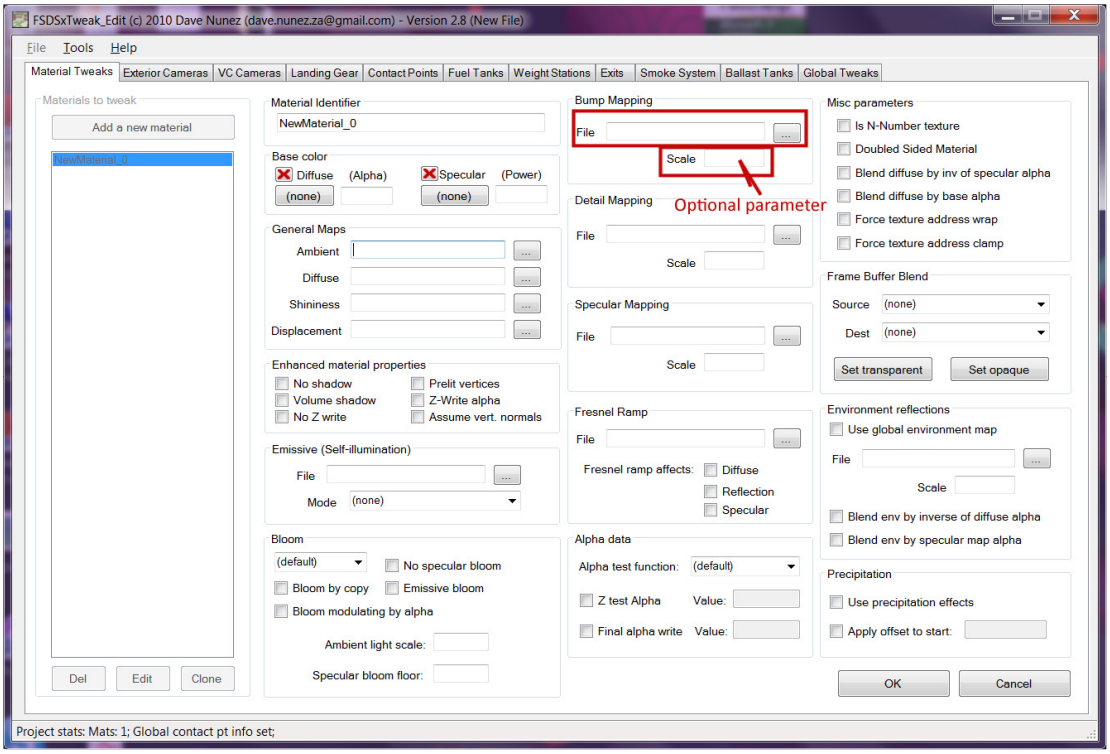


Figure 23: FSDSxTweak_Edit setup for bump mapping

Notice that I have named my bump map complex_wing_l_bump.dds. Your bump map may have any name, but it is good practice to name it the same as the base texture with _bump added at the end (it will help you figure out which bump map belongs to what later).

Explanation: The bump map is used by FSX to render bumps and pits in your model. It does this by changing the shading of the base texture depending on the spatial information contained in the bump map. The scale parameter essentially overemphasizes the heights of the bumps; values less than 1 will make the bumps flatter, while values higher than 1 will make the bumps more extreme. Generally, small changes in these values make a big difference visually, so make these adjustments with care.

14.3 *Great glass/Transparent materials*

Objective: To get a good transparent material, which includes some reflections (i.e. glass as opposed to matte transparent plastic), for example, see Figure 24.

Figure 24:Transparent material environmental reflection/mapping

Needed: An FSDS model which is built with the materials you want to be transparent with an a material diffuse alpha less than 255 (I use 128 and that works well), as well as being texture mapped with a 'reflective' texture (see the FSDS help file on how to do this). You will also need a DDS texture to use for the transparent materials - the main (RGB) channels should contain the colour of the material you want (i.e. make this green if you want green glass), and the alpha channel will control the reflectivity of the material. Note that this texture MUST have a DDS extension (not BMP).

Required settings: If you are editing your kfg file by hand, add these switches to your config file:

```
diffuse=glass.dds
SourceBlend=SrcAlpha
DestinationBlend=InvSrcAlpha
UseGlobalEnvironment=1
BlendEnvironmentByInverseDiffuseAlpha=1
environment_level_scale=1.2
```

If you are using FSDSxTweak_Edit, set the following for your glass material (Figure 25 - only the sections in the areas marked red are essential)

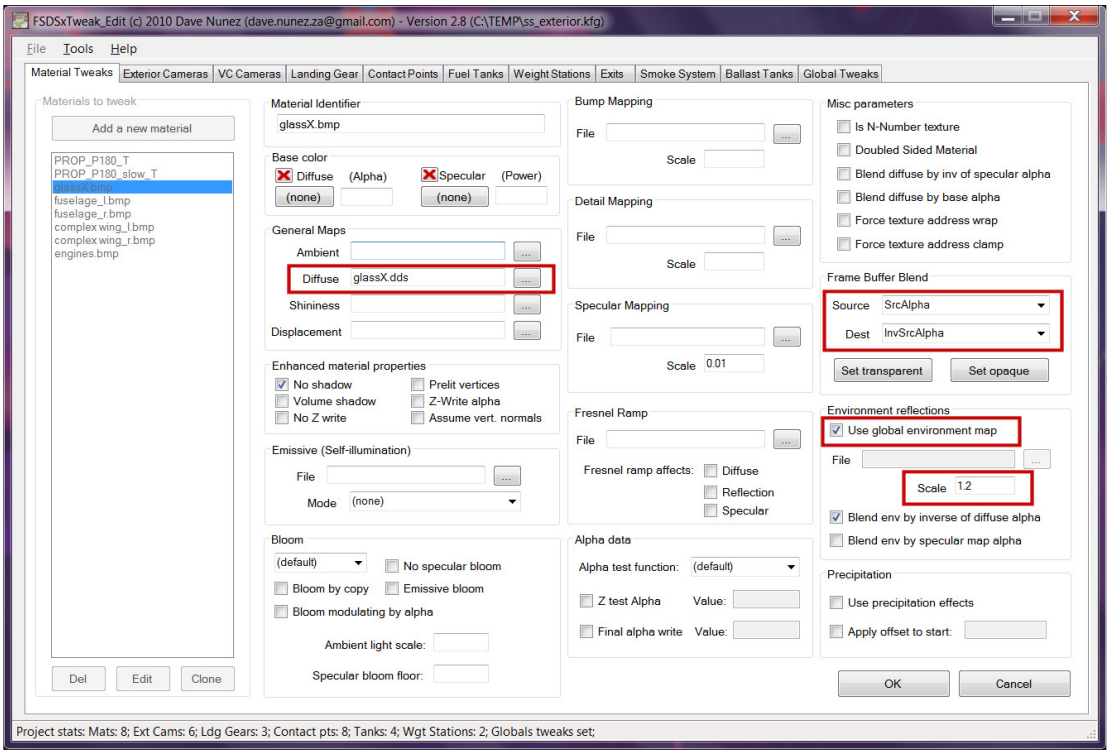


Figure 25:FSDSxTweak_Edit setup for transparency with reflections

Note that I am assuming that your glass texture is called *glassX.dds*; change the first line if your texture has a different file name. You can also set your `environment_level_scale` value higher to get a more glossy finish.

Explanation: This tweak applies both the transparency given by the FSDS material (you control the transparency with that value in the FSDS editor - 255 = opaque; 0 = invisible), as well as the global environment map reflection (the amount of reflection is controlled by the color of the DDS alpha map - black - mirror, white - matte). You can also control how much the environment map affects the material using the `environment_level_scale` parameter; 0 = no effect, large numbers = exaggerated effects (best results are from values around 1.0).

Note that the two lines:

```
SourceBlend=SrcAlpha
DestinationBlend=InvSrcAlpha
```

Actually implement the transparency. If you remove these, you will have an opaque material.

14.4 *FS9 style alpha based reflectivity*

Objective: To get a specular shine/reflectivity on your models controlled by the texture Alpha channel (as was done in FS9).

Needed: An FSDS model which is built with the appropriate textures tagged as 'reflective' (see the FSDS help file on how to do this). Your texture which contains the actual diffuse texture in the RGB channels, and the transparency in the alpha channel (see Figure 26 for examples). Note that in the alpha channel, darker greys are more shiny (black is a mirror, white is total matte). This texture should be saved as a 32 bit bmp, or as a DXT or DDS file which contains an alpha channel.

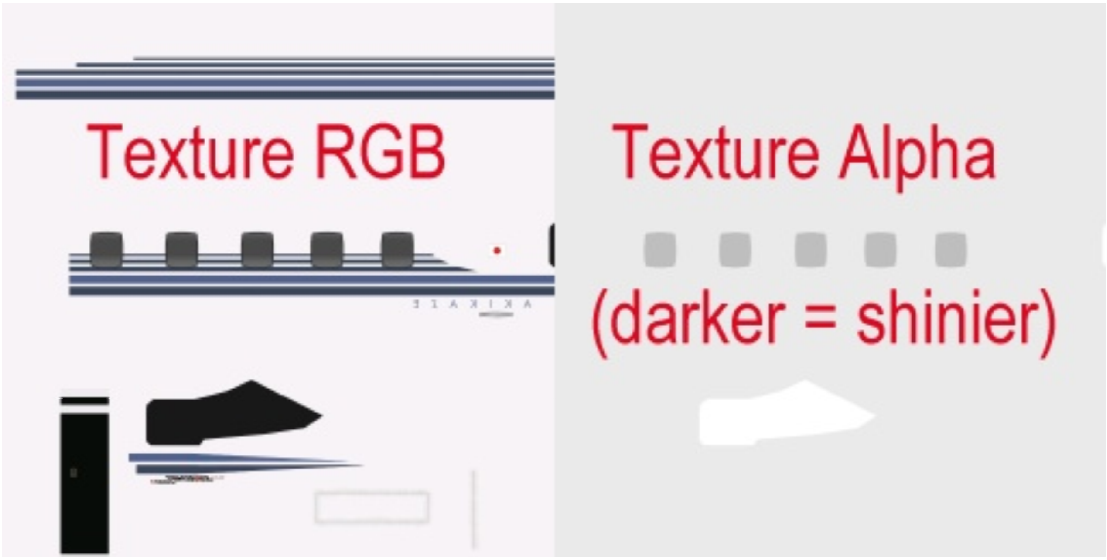


Figure 26:Required bitmap (RGB data and alpha data) for FS9 style shine

Required settings: If you are editing your kfg file by hand, add these switches to your config file:

```
environment_level_scale=1.0
UseGlobalEnvironment=1
BlendEnvironmentByInverseDiffuseAlpha=1
```

If you are using FSDSxTweak_Edit, set the following for your material (Figure 27 - only the sections in the areas marked red are essential):

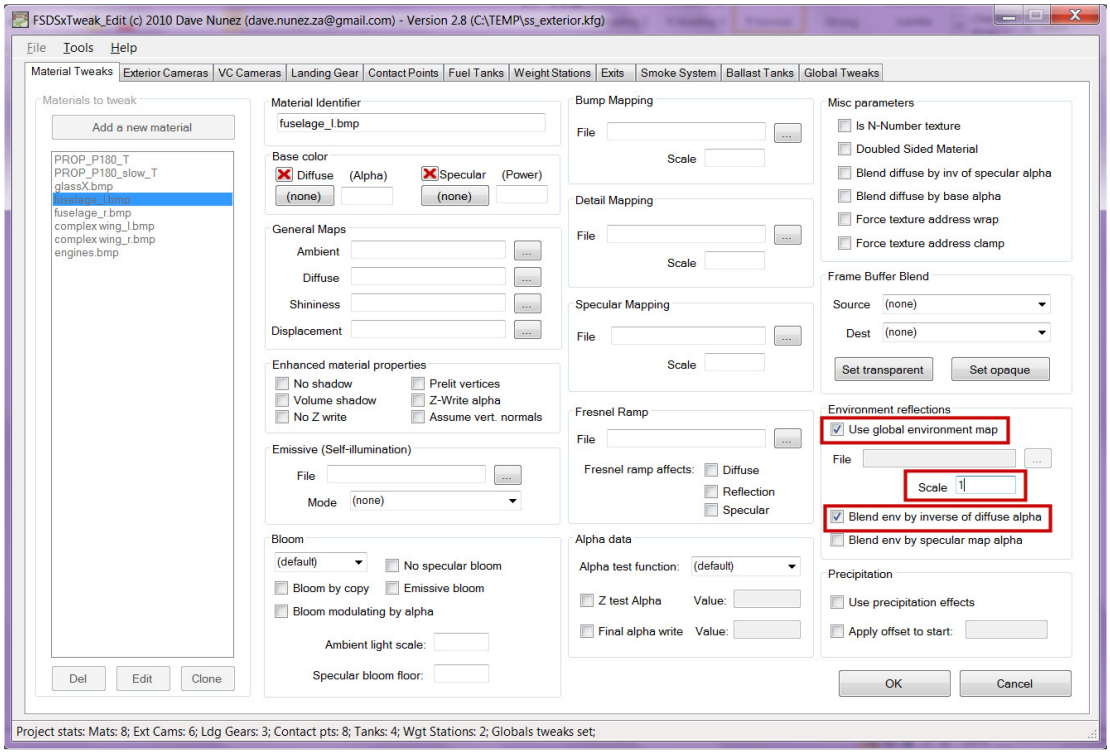


Figure 27: FSDSxTweak_Edit setup for FS9 style shine

Explanation: This tweak applies the Global environment map (a reflection of the outside world) blended to your texture. The amount it is blended depends on the *inverse* of the value in the alpha map - so for black (value 0), you get only the environment map, at mid grey (value 128), you get half environment map/half your texture, and at white (value 255), you only get your texture. You can further modify this by adjusting the `environment_level_scale` parameter - higher values give the environment map preference.

This tweak is nice because it saves a lot of video RAM on your graphics card compared to using the specular map. In the specular map, you use three bytes per pixel, while in the alpha channel you only one byte per pixel - so you use 1/3 the memory. It also means that you a moving less data across the video memory bus, so you should be speeding up the loading times, and perhaps even the frame rate (not sure about that though).

14.5 *Emissive panel lighting*

Objective: Emissive map based night lighting for the virtual cockpit gauge materials, as shown in Figure 28.

Figure 28: Emissive lighting on a VC panel

Needed: A texture of your VC gauges (which you should have anyway for mapping the VC polygons), and a new emissive map. The emissive map must be a ***DXT1 bitmap (no alpha)***, which is blended over the gauge texture. Darker colours in the emissive map will produce less light, while lighter colours will produce more light (i.e. black gives you total darkness, white gives you total light). You can use colours to show coloured lights (i.e. use red in the emissive map to produce a red light, etc). Figure 29 shows an example of a emissive map (left) which goes over a VC gauge texture (right). Note that the emissive map must be UPSIDE DOWN (even if you defined your gauges in the panel.cfg file 'the right way up').

Required settings: If you are editing your kfg file by hand, add these switches to your vc gauge material:

```
emissive=vc1_l.bmp
SourceBlend=SrcAlpha
DestinationBlend=InvSrcAlpha
EmissiveMode=MultiplyBlendUserControlled
```

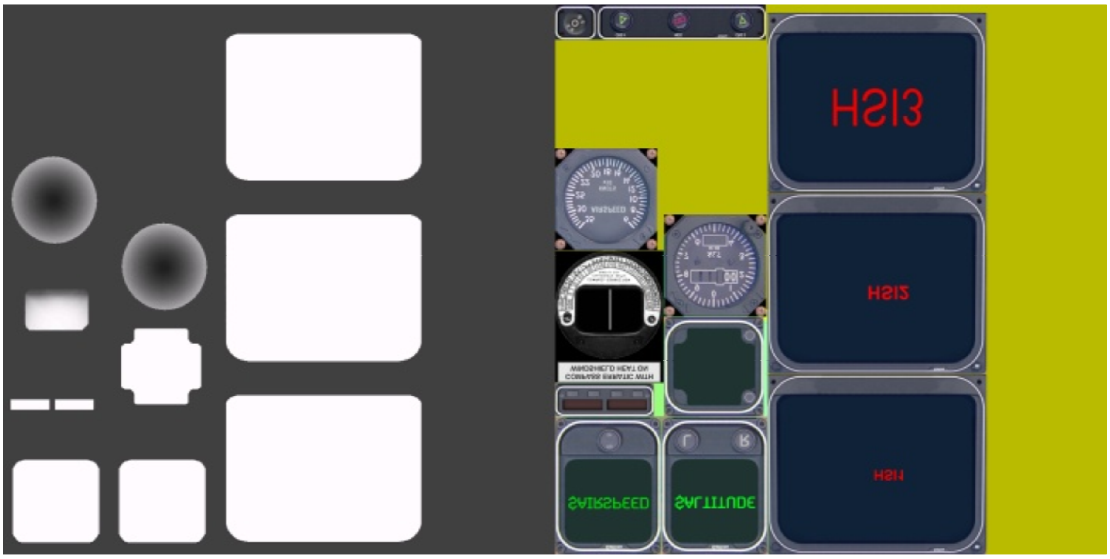


Figure 29: Emissive map (left) used to light the VC gauge sheet (right)

If you are using FSDSxTweak_Edit, set the following for your VC gauge material (Figure 30 - only the sections in the areas marked red are essential)

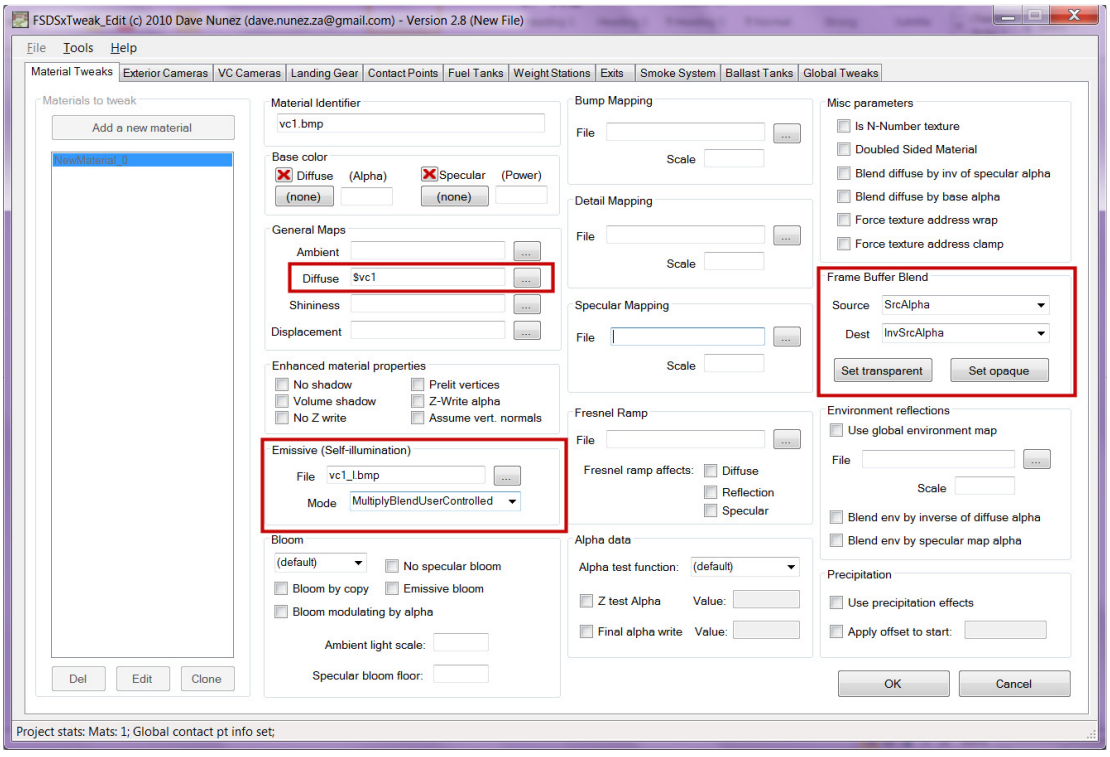


Figure 30: FSDSxTweak_Edit setup for emissive lighting of gauges

Notice that I have named my emissive map *vc1_l.bmp* - yours can be called anything (the _l is not necessary, but it is convenient to easily identify your light maps).

Explanation: An emissive map is essentially blended over the base texture map under particular conditions (set by the `EmissiveMode` switch). In this case, we want the colour of the base texture map and of the emissive map multiplied together before they are put on the actual polygon. Also, the mode is `usercontrolled` meaning that its use is linked to the panel lights switch in the simulator (the `KEY_PANEL_LIGHTS_TOGGLE` event) - it will only be applied when then user hits the panel lights switch.

14.6 *Automatic correct gear compression*

Objective: To have dynamic compression on the aircraft landing gear which responds to weight, braking, etc, but without the wheels sinking into the ground.

Needed: Your aircraft model, with existing animation keyframes for the compression. At keyframe 0, you should have the gear retracted (or not, if your aircraft does not have retracting gear!). At keyframe 100, you should have the gear extended, and completely uncompressed, as if there were no weight at all on the wheels (see Figure 31). At keyframe 200, you should have the gear extended, and the gear completely compressed (as it would if the plane were on the ground and loaded up to maximum weight and fuel – see Figure 32).

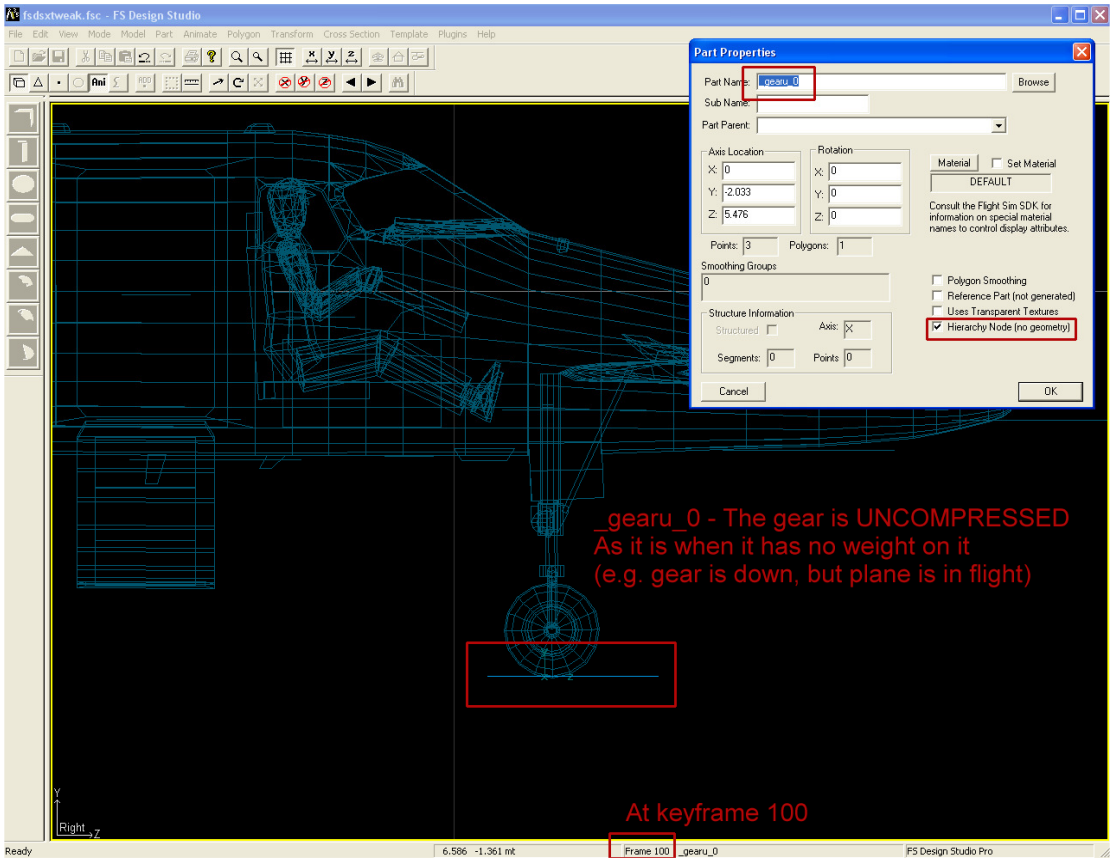


Figure 31: The part `_gearu_0` aligned to gear at frame 100

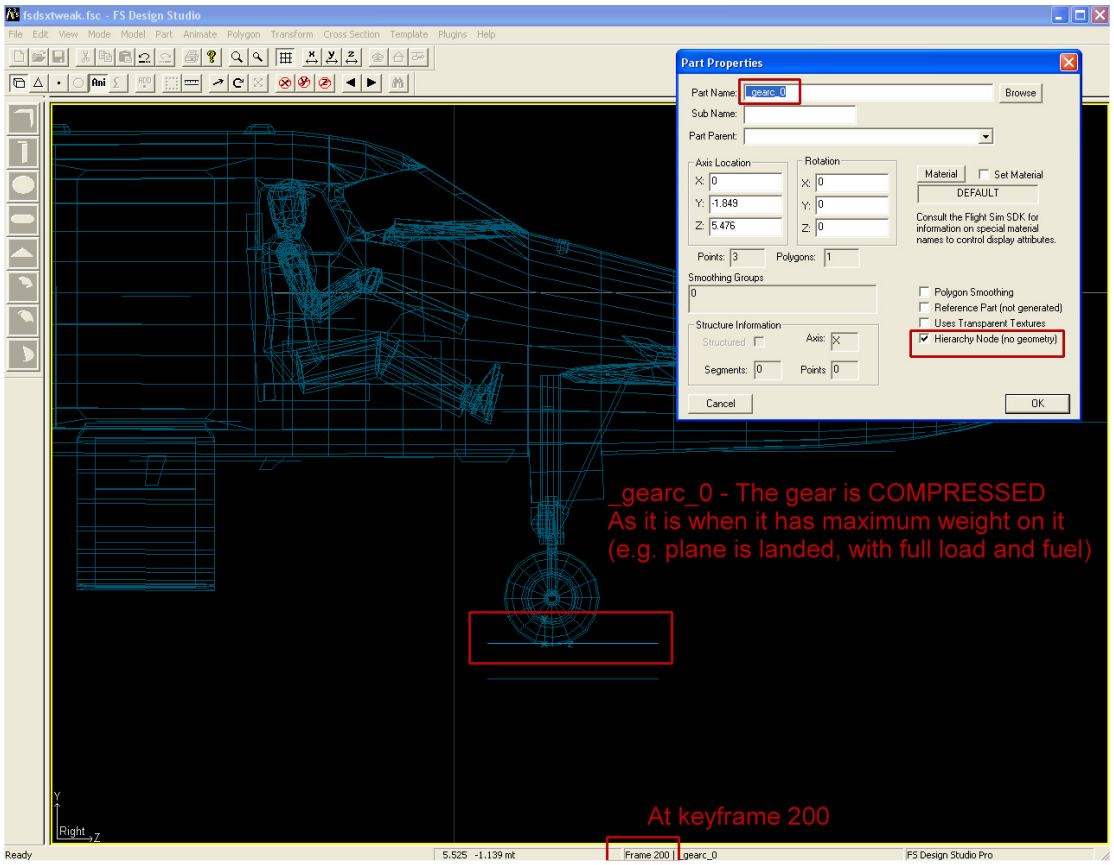


Figure 32: The part _gearc_0 aligned to gear at frame 200

Required settings: *Note: All of the word required for this tweak is done inside FSDS.* For each gear point, you will need to add two special reference parts: one called `_gearc_0` and another called `_gearu_0`. Note that the zero at the end of the name indicates that this is the first gear point – your second point would need parts `_gearc_1` and `_gearu_1`, while the third point would have parts `_gearc_2` and `_gearu_2`, and so on.

To create these special parts, create a new 3 sided polygon (in FSDS menu: **Part** then **Add** then **Polygon…** then select 3 sides). Scale it down till it is a reasonable size (see the images above for the right sort of size) – note that the size does not really matter as this polygon will not be placed into your model; just scale it so it is a good size to work with.

Once the part is made, open the part properties (by pressing the F2 key), name it `_gearc_0`, and make sure the "Hierarchy Node (no geometry)" box is checked (see the top screenshot). Once this is done, copy and past this part to make another, and rename it to be `_gearu_0`.

14.7 *Adding a new animated part using Modeldef_Edit*

Objective: To add a new animated part to your model using Modeldef_Edit. For the purposes of this tutorial, I will use the 3D compass logic by Robert Bruce, which you can find on http://www.aerodynamika.com/cgi-bin/yabb/YaBB.cgi?num=1164752978

Needed: Your aircraft model, with the new part modeled as you want it. According to Robert's instructions, this part will have 360 frames in its animation. You should animate it so that it shows the main compass points at keyframes 0, 90, 180, 270 and 359. Apart form that, you will need the <PartInfo> data provided by Robert on the forum:

```
<PartInfo>
     <Name>compass_dial_mk5</Name>
     <AnimLength>360</AnimLength>
     <Animation>
     <Parameter>
          <Sim>
            <Variable>PLANE HEADING DEGREES GYRO</Variable>
              <Units>degrees</Units>
              <Scale>1</Scale>
              <MinValue>0</MinValue>
              <MaxValue>360</MaxValue>
          </Sim>
     </Parameter>
     </Animation>
</PartInfo>
```

Procedure: Be sure that you back up your modeldef.xml file before you begin editing it. Also, if you are using FSDS, go to the **Options** menu of Modeldef_Edit, click **Settings…** and in the settings window, turn on the **Auto-update FSDS with new parts**, and enter the path of FSDS into the **FSDS location** textbox. Close the settings window, and then open your modeldef.xml file in Modeldef_Edit:

Now we need to create the new part, and give it the correct settings. Click the **New part** button below the part list. Notice that a new part called `NewPart_000` has been created, and you are automatically in edit mode. The part we want to create is an animated part, so it will have both an Animation block (which contains the animation parameters) and a PartInfo block (which contains the actual XML code). Turn on both the **Has Animation Block** and **Has PartInfo Block**.

As we already have the XMl code from the forum, we will deal with the PartInfo first. Copy the XML code (all of it, including the `<PartInfo>` and `</PartInfo>` tags) and then paste it into the XML work window by clicking in it, and then pressing **Ctrl-V** on your keyboard. You will notice that the XML code has been color coded.

Now we can deal with the Animation block. Have a look at the XML code you have pasted – you will notice that it has a legal name (in between the `<Name>` and `</Name>` tags), and length of the animation (in between the `<AnimLength>` and `</AnimLength>` tags). The SDK requires that the length and name sections of the Animation block and PartInfo blocks match exactly, so to ensure this, click the **Up arrow** button next to the **Name** textbox in the Animation section. Notice that the part's name (`compass_dial_mk5`) has been automatically copied into the Name textbox. Now do the same for the length parameter by clicking the **Up arrow** button next to that textbox. Now all we need is a unique GUID for this part. Click the New button next to the GUID textbox, and a GUID will be created for you.

We are done. Before you click OK, check Figure 33 to make sure that you have filled in all the fields as they are in the image. Then click OK.

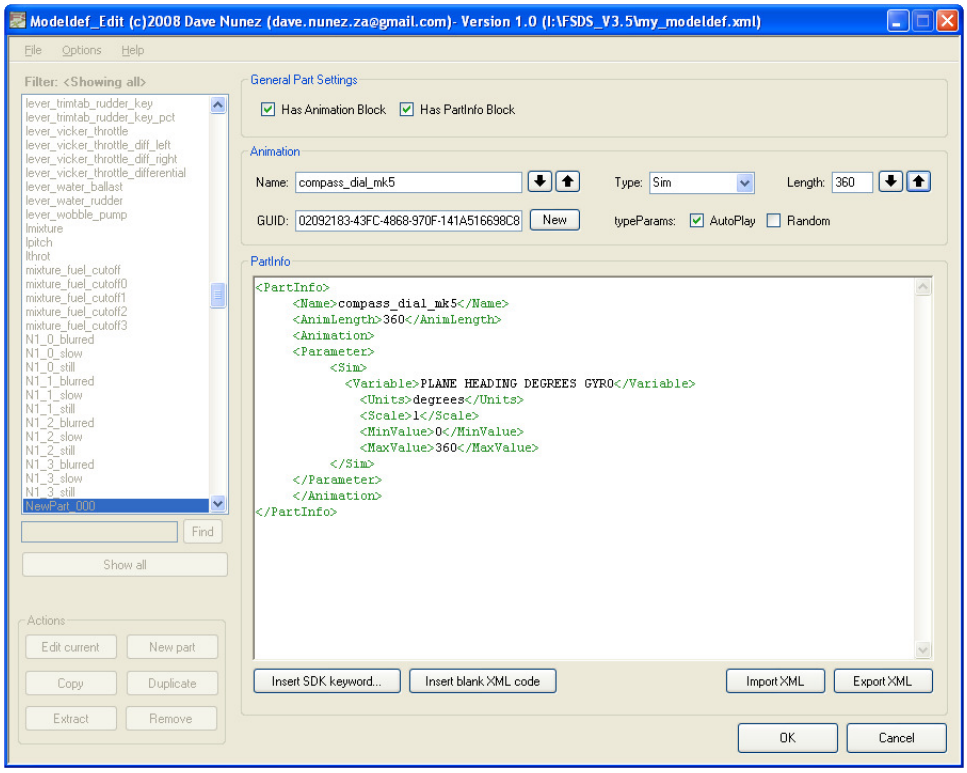


Figure 33: Modeldef_Edit in edit mode with a new part

Let's check that your part came through OK. Because the list of parts is long, it is easier to use the filter feature rather than scrolling through it manually. We know that the word `dial` was in the new part's name somewhere, so in the textbox under the part list, type `dial` and press enter or click the **Find** button. The list will then change to show all parts that contain `dial` somewhere in their names. To return to the complete part list, click the **Show All** button under the part list.

Now that we have our new part ready, save your modeldel.xml file. You should be able to compile your model, and if you have included a part named `compass_dial_mk5` in your model, it will be animated in the correct way.

14.8 *Attaching an effect to a model*

Objective: To attach smoke effects to a model. Although one can add smoke effects via the aircraft.cfg file, attaching them directly to the model allows you to have smoke effects attached to animated parts (such as swing wings, etc).

Needed: Your aircraft model, and effect files (.fx files) to attach. For this tutorial we will use two of the default FSX effects – `fx_smoke_r.fx` (red smoke) and `fx_smoke_w.fx` (white smoke).

Procedure: Each effect we are going to attach will need a new part, named _attacheffect_[effect name]. The parts you create can have any shape or size, as they will not appear in the final model (the part will be replaced by the effect). Create this is a normal part (see the part properties in Figure 34).

For the red smoke, we will create the part and name it `_attacheffect_fx_smoke_r` and for the white smoke we will name the part `_attacheffect_fx_smoke_w`

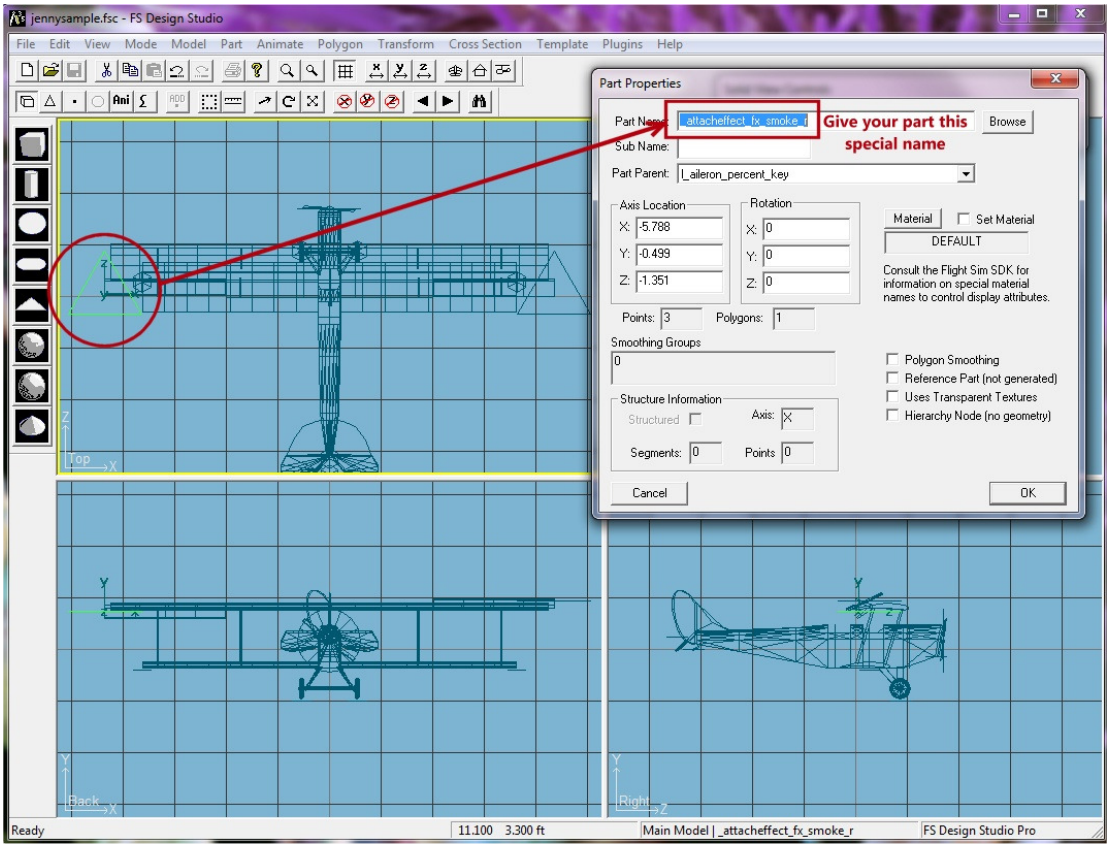


Figure 34: Setting up for attaching effects

This is all you need to do. Compile the model and run it through FSDSxTweak. Open the model in FSX to see the magic (see Figure 35).

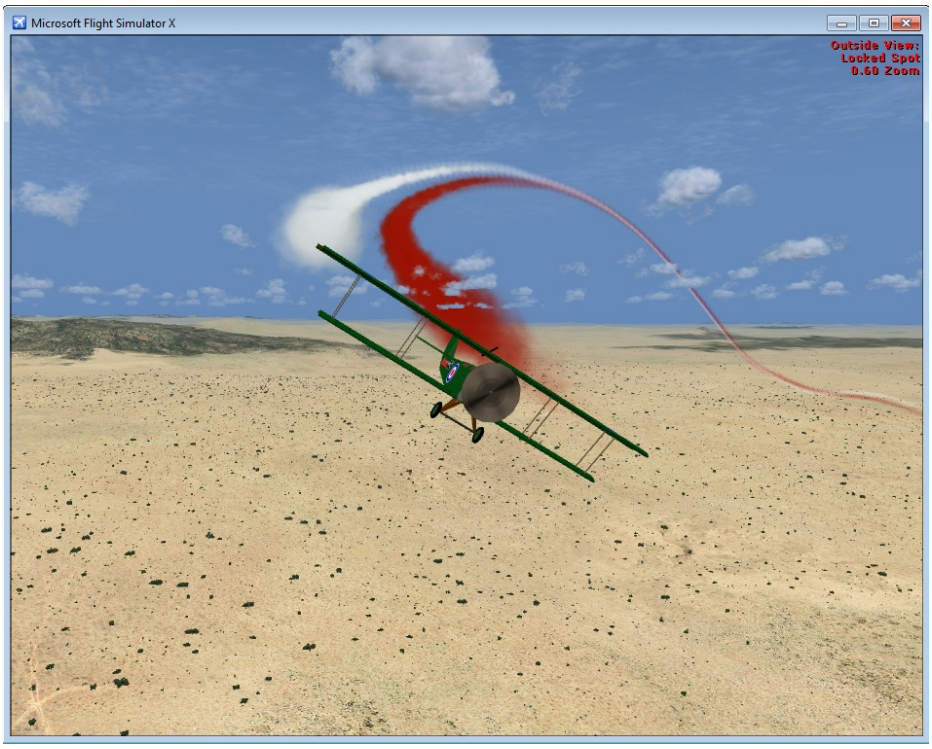


Figure 35: The smoke effects in FSX

14.9 *Attaching a landing light to a model*

Objective: To attach a landing light to an aircraft. Landing lights cannot be attached to via the aircraft.cfg file, so attach points are a simple way to solve this problem.

Needed: Your aircraft model and the landing light effect file (fx_landing.fx) which comes with FSX.

Procedure: A new part with the special name `_attacheffect_fx_landing` which can have any shape or size, as it will not appear in the final model. Ensure the Y axis of the part axis is pointing away from the direction you want the light to point in (see the part properties and setup in Figure 34).

This is all you need to do. Compile the model and run it through FSDSxTweak. Open the model in FSX and you should see your landing lights.

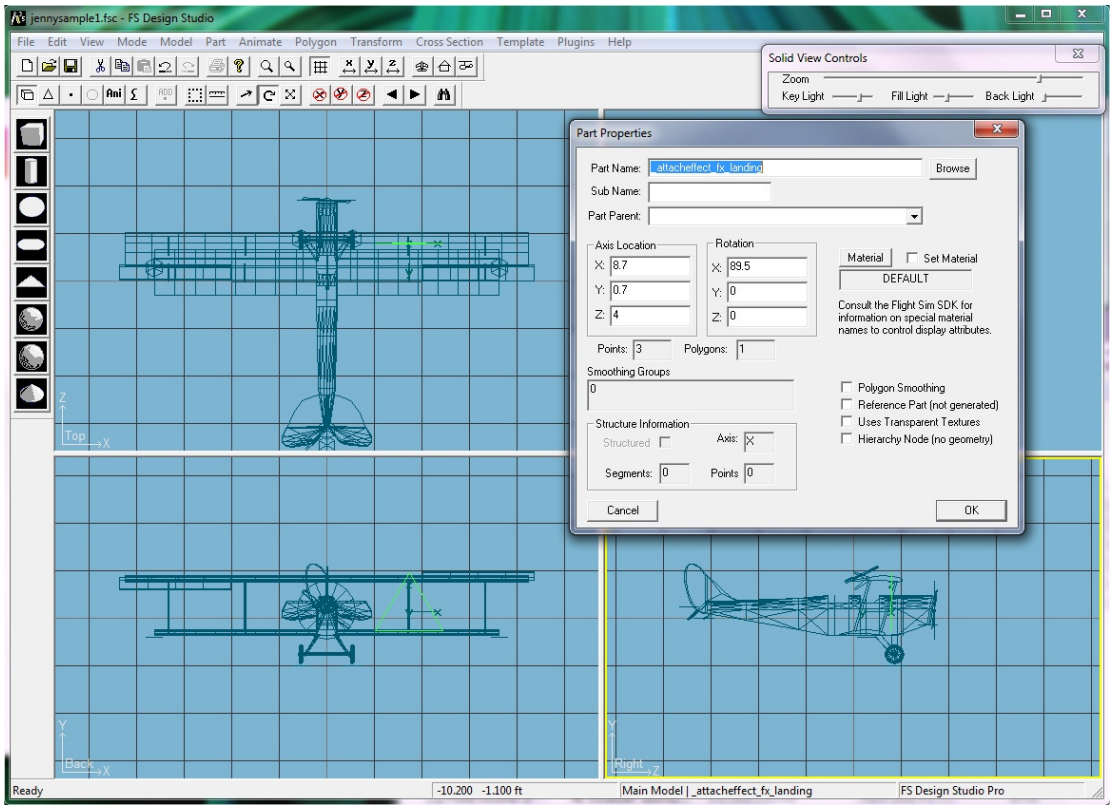


Figure 36: Setting up to attach a landing light

14.10 *Attaching arresting wires and catapults to an aircraft carrier*

Objective: To attach arresting wires and catapults to an aircraft carrier model. This requires the FSX Acceleration pack (carrier parts are part of the Acceleration SDK).

Needed: A carrier model with a landable platform to act as a runway.

Procedure: Each arresting wire needs two special parts. For the first wire we will create two special parts `_attachbyname_attachpt_cable_1_1` and `_attachbyname_attachpt_cable_1_2` (parts labeled A and B in Figure 37). The wire will be strung between these two points. The parts you create can have any shape or size, as they will not appear in the final model (the part will be replaced by the effect). Use the same settings as you did for the smoke effects (see Figure 34), but use these new names.

For the second arresting wire, create two parts again, but name them `_attachbyname_attachpt_cable_2_1` and `_attachbyname_attachpt_cable_2_2` (parts C and D in Figure 37). To add a third wire add the parts `_attachbyname_attachpt_cable_3_1` and `_attachbyname_attachpt_cable_3_2`, and so on.

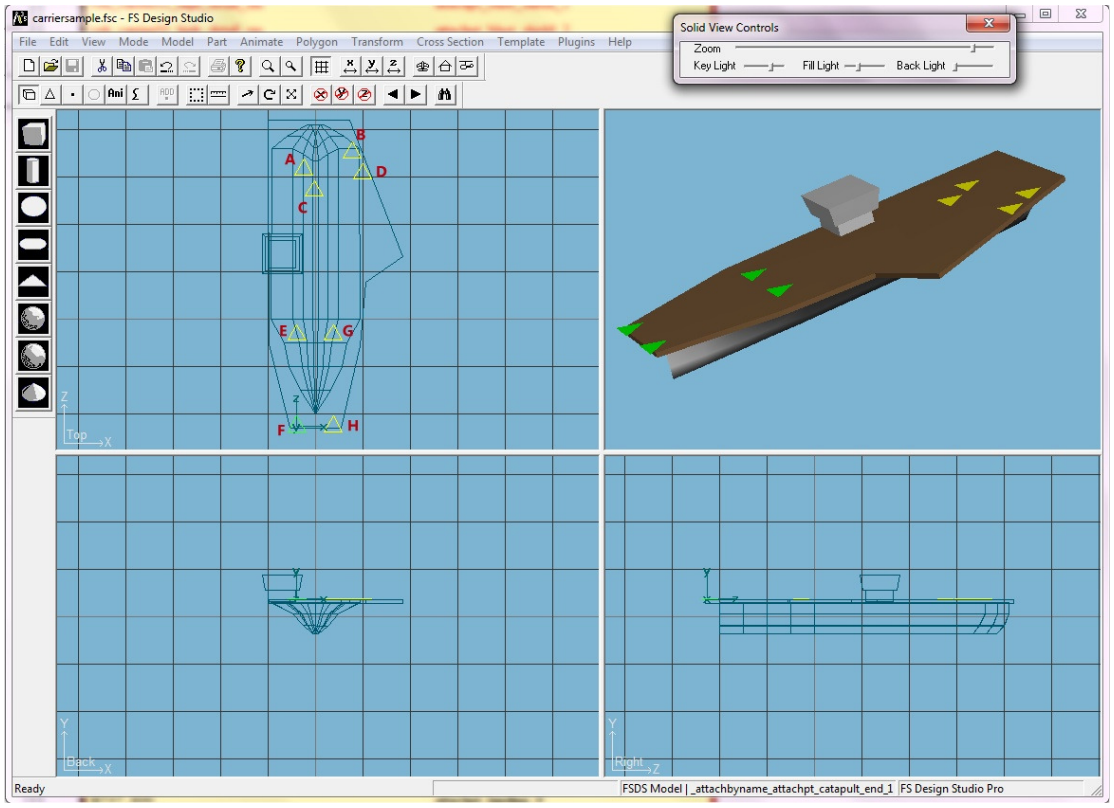


Figure 37: Setting up the carrier wires and catapults

The catapults are done in a similar way. The first catapult uses two parts named `_attachbyname_attachpt_catapult_start_1` and `_attachbyname_attachpt_catapult_end_1` (parts E and F in Figure 37). The second catapult will use parts named `_attachbyname_attachpt_catapult_start_2` and `_attachbyname_attachpt_catapult_end_2` (parts G and H in Figure 37).

This is all you need to do. Compile the model and run it through FSDSxTweak.

15. Release history

15.1 *FSDSxTweak*

V2.8: Added the material properties: BlendDiffuseByInverseSpecularMapAlpha, BlendDiffuseByBaseAlpha, ForceTextureAddressWrapSetting, ForceTexture AddressClamp, ZTestAlpha, AlphaTestValue, AlphaTestFunction, FinalAlphaWrite and FinalAlphaWriteValue

V2.7: Added the ability inject attachpoints, visibility nodes, nocrash nodes and mouserect nodes via the PartData{} chunk in the .x files. This removes the need to manipulate the PartDataDef.txt file in FSDS. Also, it adds the ability to add carrier parts, which is not otherwise possible in FSDS.

V2.6: Fixed a missing block (AllowBaseMaterialSpecular) in the .x file output which was breaking the base material specular feature in textured materials.

V2.5: [This version number was skipped because I didn't notice before releasing!]

V2.4: Added the ability to set the base material specular properties (intensity and colour)

V2.3: Fixed a bug where even if a material was set to "no bloom" it would still set allowbloom to 1 in the .x file.

V2.2: (V2.1 was not released – 2.2 is used to match the number of the other tools)

Added bloom and misc (NNumber and doulesided) parameters to materials

Added diffuse and specular material color tweak

Fixed a bug where precipitation parameters of materials were written incorrectly in the output .x file

V2.0: Added support for inserting part info into aircraft.cfg

Added support for oriented parts in aircraft.cfg via decomposition of .X file 4x4 transformation matrix

Added support for processing aircraft.cfg files by adding a new optional argument with the name of the aircraft.cfg file to update.

v1.6: Added audit mode

Added version reporting for compatability with FSDSxTweak_Plugin - use "/version <filename>"

v1.5: Added the ShadowMapReady model tweak

v1.41: Test version, not released; I rushed to ship the last one and horrible bugs were introduced. Thanks to Rick Piper for all his testing contributions!

v1.4: Built with a new version of MSVC, and the new version of the Allegro library
Fixed a bug which reset the specular map power scale to 0 if not defined
Now prints out the fsdsxtweak version number when it begins a run
Added three function selectors for advanced effects:
SourceBlend and *DestinationBlend*
EmissiveMode, which allows one to set proper advanced night lighting on VCs and such

v1.3: Added full set of base material properties:
UseGlobalEnvironment
BlendEnvironmentByInverseDiffuseAlpha
BlendEnvironmentBySpecularAlpha
FresnelDiffuse
FresnelSpecular
FresnelEnvironment
PrecipitationUse
PrecipitationOffset
PrecipitationOffsetValue
SpecularMapPowerScale

v1.2: Added full set of material maps; now supports:
diffuse
specular
ambient
emissive
reflection
shininess
bump
displacement
detail
fresnel
Added the full set of enhanced material properties
AssumeVerticalNormal
ZWriteAlpha
NoZWrite
VolumeShadow
NoShadow
PrelitVertices

V1.1: Fixed some bugs in v1 which lead to 'uncompilable' .x files
Added *environment_level_scale* parameter
Added *detail_scale* and *bump_scale* parameters

v1.0: Initial release

15.2 *FSDSxTweak_Plugin*

V2.8: (Forced version name to 2.8 to sync up with the other tool numbers)

Added a status strip to the main window to give some feedback of the last action.

Fixed the 'check for updates' feature so that it actually works now!

Added autodetect of invalid parameters on a timer so that in real time the compile buttons are enabled/disabled to indicate that settings need to be fixed before compilation.

V2.4: Fixed issues with writing to the registry for the verbose and KFG file associations which broke the feature under Vista and Windows 7.

V2.3: Added the additional switches for XtoMdl under the Advanced page separately for interior and exterior models.

V2.2: Decluttered the gui by moving the general settings (location of tools etc) to a separate window which is accessed by the options menu now.

Fixed a bug in the 'check for updates' code which always told you to updated to current version

Added ability to spawn DrawCallMonitor v1.1

Added the pre/mid/post build batch files

Added the output-redirection to file feature

Added the auto post build DrawCallMonitor and/or audit feature

V2.1: Changed the saving of default config from default.xcf to settings in the registry key HKEY_CURRENT_USER\Software\FSDSxTweak\FSDSxTweak_Plugin. Note that default.xcf is not used anymore at all.

V2.0: Added console integration for KFG files
Moved some support fuctions to menu
Added interface for new aircraft.cfg tweaking
Added partial aircraft.cfg functionality
Added option to auto-close compile windows on completion

v1.3: Added support to call FSDSxTweak_Edit directly for cfg files
Added the backup cleanup feature

v1.2: Added support for the 'audit' feature of fsdsxtweak 1.6

Added the 'check for updates' functionality (also checks if fsdsxtweak is up to date)

Added the 'go to FFDS forum' button to make it quicker to post queries!

v1.11: Fixed a bug where 'build all models' would actually build the exterior model twice

v1.1: Added the 'view FSDSxTweak readme' button to help with editing the cfg files

Added the 'edit' buttons to reduce the tedium associated with finding all those files for editing!

Added a lot of error checking and warnings to prevent crashes etc.

v1.0: Initial release

15.3 *FSDSxTweak_Edit*

V2.8: Added Material properties for BlendDiffuseByInverseSpecularMapAlpha, BlendDiffuseByBaseAlpha, ForceTextureAddressWrapSetting, ForceTexture AddressClamp, ZTestAlpha, AlphaTestValue, AlphaTestFunction, FinalAlphaWrite and FinalAlphaWriteValue.

Added the 'Clone Material' button to the material list pane

Double-clicking on any tweak list will now put the tweak into edit mode.

Made the material tab be the first tab (as editing materials seems to be what people do most)

Got rid of the status/project stats popup window and replaced it with the status strip at the bottom of the window

Fixed some bugs around diffuse and specular color were leading to values getting lost under some conditions

[Some version info has been lost]

V2.3: Changed the 'allow bloom' control to a drop-down with a '(default)' option to better support the case where people want to keep the material properties set by FSDS.

V2.2: Added diffuse and specular color tweak to materials

Fixed a bug with saving global tweaks (ShadowMapReady and UnitsMetric)

Removed custom GUID generation code and replaced is with .NET code

Added context menus (right click) to lists (with edit, remove, add)

Fixed a bug on landing gears which displayed empty string on the "are you sure you want to delete" dialog

Added bloom and misc parameters to the materials tab

Fixed a bug where File->New did not clean up contact points and gears

V2.1: Fixed a big where renaming numbered names would lead to some items being overwritten

Load/Save of contact points was not implemented; now loads/saves contact points

Load of gear points was not implemented

Gears were not being read on KFG load

read_ballast() was not updating num_active_ballast properly and had a number reading bug

Weight station gui did not include identifier number

Rewrote command line arg code, now order of args is not important, although version mode remains the same (two args, first is /version, second if <filename>)

Added /parts:<filename> and /materials:<filename> switches to allow import form .x file

V2.0: Added support for the new aircraft.cfg tweaks (fuel tanks, etc).

Added disaster recovery

Added tools (guid generator, etc)

Improved help menu options (manual, etc)

Changed default file type for KFG

Fixed many gui related bugs (passing null args to functions, etc).

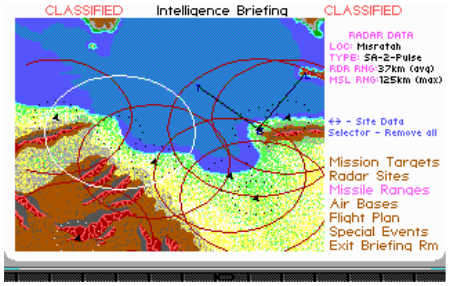
V1.0: Initial release

15.4 *Modeldef_Edit*

V1.1: Fixed a bug where the braces ({ and }) were not being shown in the XML work window when XML highlighting was turned on.

V1.0: Initial release

In case you were wondering, this document was written in Oranjezicht, South Africa, at Gate B02 at Cape Town International Airport, at Gate E05 at Amsterdam Airport Schiphol, Redmond, Washington, and a little wooden house in Bellevue Washington.



The choice of this font is an homage to those great Microprose simulations of the 1980s (F-19 Stealth Fighter, F-15 Strike Eagle II, Red Storm Rising, etc) which also used this font in their great manuals, and got so many of us hooked on simulation.

Alright, enough already – go build a plane!