Development:Tools:UnitConverter

From VsWiki

Contents

- 1 Introduction
  - 1.1 Important Note
- 2 Requirements
- 3 Quick Start Guide
  - 3.1 Configuration
  - 3.2 Integrating Models
  - 3.3 Converting Hud Image
  - 3.4 Using Uncompressed Images
  - 3.5 Adding Thruster, Docks, Mounts
  - 3.6 Editing Stats
  - 3.7 Comparing Units
  - 3.8 Getting Help
- 4 UnitConverter Tool
  - 4.1 Installation and Starting
  - 4.2 Models Workspace
  - 4.3 Textures
    - 4.3.1 Advanced Usage - Converting faction textures
  - 4.4 HUD Image
  - 4.5 Unit Editor
    - 4.5.1 Advanced Usage - Independent Unit Editor
  - 4.6 View
  - 4.7 Configuration
    - 4.7.1 Advanced Configuration
  - 4.8 About
- 5 Code Reference
  - 5.1 Python Modules
  - 5.2 Program Flow
  - 5.3 Data Elements

Introduction

Raising from the stone age of workflows, in which every task, from texture conversion to editing of unit stats, had to be accomplished manually and only by quite knowledgeable magistrates who knew the alchemy of units.csv columns, xmesh files, and had a magic with meshers. Now, UnitConverter shines upon us, still in the distance, but unmovable like a beacon on solid rock, to be the church of the artists, to accommodate the apostles of Blender, CineMut, and LaGrande, to bring the holy spirit of salvation to this compassionate universe of ours.

Some of the main Unit Converter features:

- bi-directionally converts meshes between obj and bfxm files
- assign textures to (sub)models

- convert model and shield meshes
- compress default and faction input textures to dds output textures
- saves the complete workspace in portable config files
- change model techniques and blend mode
- checks and corrects invalid obj files, recreates mtl files
- creates a model directory if none exists and copies all required files
- compresses hud image and creates a .sprite file
- units.csv entry editor and creator (with default unit stats for new units)
- displays model using VS
- reads mesh dimensions from obj file
- placement of thrusters, turrets, mounts, docks, and blink lights from marker obj file
- on-screen instructions for each workflow module
- production tested

Important Note

> Due to the way *mesher* handles command line options (separates different inputs by spaces), it will not allow to use folders or file names with spaces (as common on Windows). Should you be getting runtime errors during the conversion, you should rename your folder to something like "UnitName" or "Unit_Name_working_folder", i.e. substitute spaces with underscores "_". Do the same for the input files, i.e. "Unit_Name.obj".

Requirements

In order to run UnitConverter on your system is to have Python 2.5 or 2.6 installed on your system. Python 3.x has different syntax and UnitConveter will not work as long as it hasn't been ported to Python 3. Tkinter library is required to run UnitConverter, however it comes installed with most python distributions. Some Linux distributions require a separate installation of Tkinter.

A requirement for the **model obj files** is that no mtl file is provided. UnitConverter will generate its own texture information based on the input given under the **Model** panel. The best way for exporting files from blender is to use the following three options *only*: **Edges**, **UVs**, **Objects**. You may also enable Normals, and Groups. However the **Materials** export option should be explicitly disabled.

Especially when reworking existing units, it is recommended that the model **obj** file is named the same way as the unit in *units.csv*, in particular what concerns the letter case, usually following the CamelCase convention.

You will further require master textures for the object, which must be on the **png** file format and Power-Of-Two. Master textures for factions should have the faction name prefixed to the texture name, e.g. "confed_hull.png".

Optionally, an **obj** file with markers can be supplied and must contain the name "marker" in the file name.

Optionally, a shield **obj** file can be supplied, containing the name "shield" in the file name.

Quick Start Guide

Configuration

You must first configure the paths under Config including the application in the path:

- *path\data* (to edit unit stats and view model in Vega Strike)
- *path\mesher.exe* (to convert model to bfxm) - get mesher (http://vegastrike.svn.sourceforge.net/viewvc/vegastrike/trunk/win32/bin/)
- *path\nvcompress* (to compress textures) - see here for instructions on nvcompress

(There is no need to set the working dir, as this field is purely informational)

Integrating Models

Assuming you have exported:

- ship mesh to *Ship.obj*
- shield to *Ship-Shield.obj*
- helpers to *Ship-Mounts.obj*

... the steps to convert your model are:

- In UnitConverter, under "Model" add all three obj files to workspace.
- Then select the main Ship model in the drop down.
- Under "Textures" you add the ones you have provided.
- Press "Save and Convert"
- Press "Convert Textures"
- Press "Copy Data"
- Under "Unit" press "Save"
- Under "View" press "View"

This will call vegastrike and the modelview.mission with your new unit enabled.

Converting Hud Image

Under "HUDImage":

- Open the "*Ship-hud.png*" image using the "..." button
- Press "Convert Image"
- Press "Copy Data"
- Under "Unit" press "Save"
- Under "View" press "View"

Using Uncompressed Images

- Under "Config" check the "Bypass compression" option.
- Under "Textures" press "Save and Convert"
- Press "Copy Data"
- Under "View" press "View"

Adding Thruster, Docks, Mounts

- Under "Model" press "Add Model"
- Select the "*Ship-Mounts.obj*" file with your mount markers.
- Under "Unit" and "Unit2" review the stats
- Under "Unit" press "Save"
- Under "View" press "View"

Editing Stats

- Under "Unit", in "Unit name" write the units.csv key (e.g. *Llama.begin*)
- Press "Load"
- Edit fields under "Unit" and "Unit2"
- Under "Unit" press "Save"

Comparing Units

- Under **Unit** tab, in **Unit name** write the units.csv key (e.g. *Llama.begin*)
- Press **Load**
- Use **<** and **>** to browse backward and forward.
- After editing a parameter make sure to press **Save**

Getting Help

In case of problems, please report to the Unit Converter thread (http://vegastrike.sourceforge.net/forums/viewtopic.php?f=20&t=11885) .

UnitConverter Tool

Installation and Starting

You can download the latest version of UnitConverter from the Vega Strike svn repository:

- View repo trunk/modtools/UnitConverter (http://vegastrike.svn.sourceforge.net/viewvc/vegastrike/trunk/modtools/UnitConverter/) .
- Download with

```
svn co https://vegastrike.svn.sourceforge.net/svnroot/vegastrike/trunk/modtools/UnitConverter
```

To install, copy all files to a directory of your choice.

To run UnitConverter, open command line/shell, change to the installation directory, and depending your operating system type (without the ">" which denotes the command line prompt:

Windows:

```
> uc
```

Linux:

```
> ./uc
```

Other:

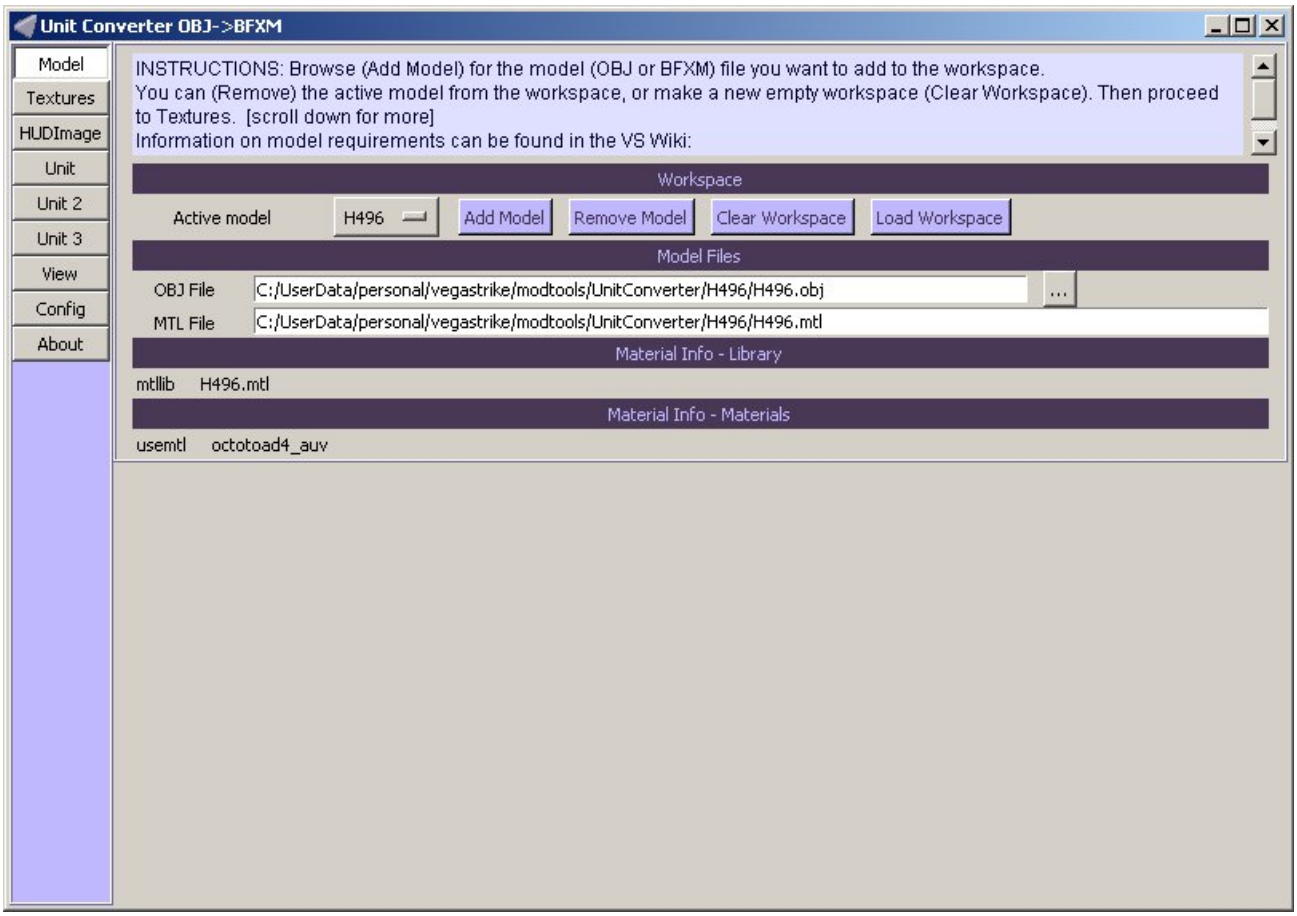
```
> python unitconverter.py
```

First thing you'll need to do is to configure the paths for Vega Strike data directory, mesher and nvcompress.

> **Important:** To be fully able utilize UnitConverter functionality, please configure the paths under **Config**.

Models Workspace

Models Workspace Screen:



The following operations are available:

- **Clear Workspace** - to start a new model integration
- **Add Model** - to add a new model to workspace. The following files are accepted:
  - *.obj* file containing the model mesh
  - *?-Shield.obj* file containing the shield mesh
  - *?-Marker.obj* file containing the marker objects as described in the objects marker modeling section
  - *.bfxm* files to backward convert meshes to obj files
- **Clear Model** - to remove the selected model from the workspace
- **Load Workspace** - to load an *.ini* configuration file of the workspace
- **Active Model** - change active model on the dropdown to edit its textures

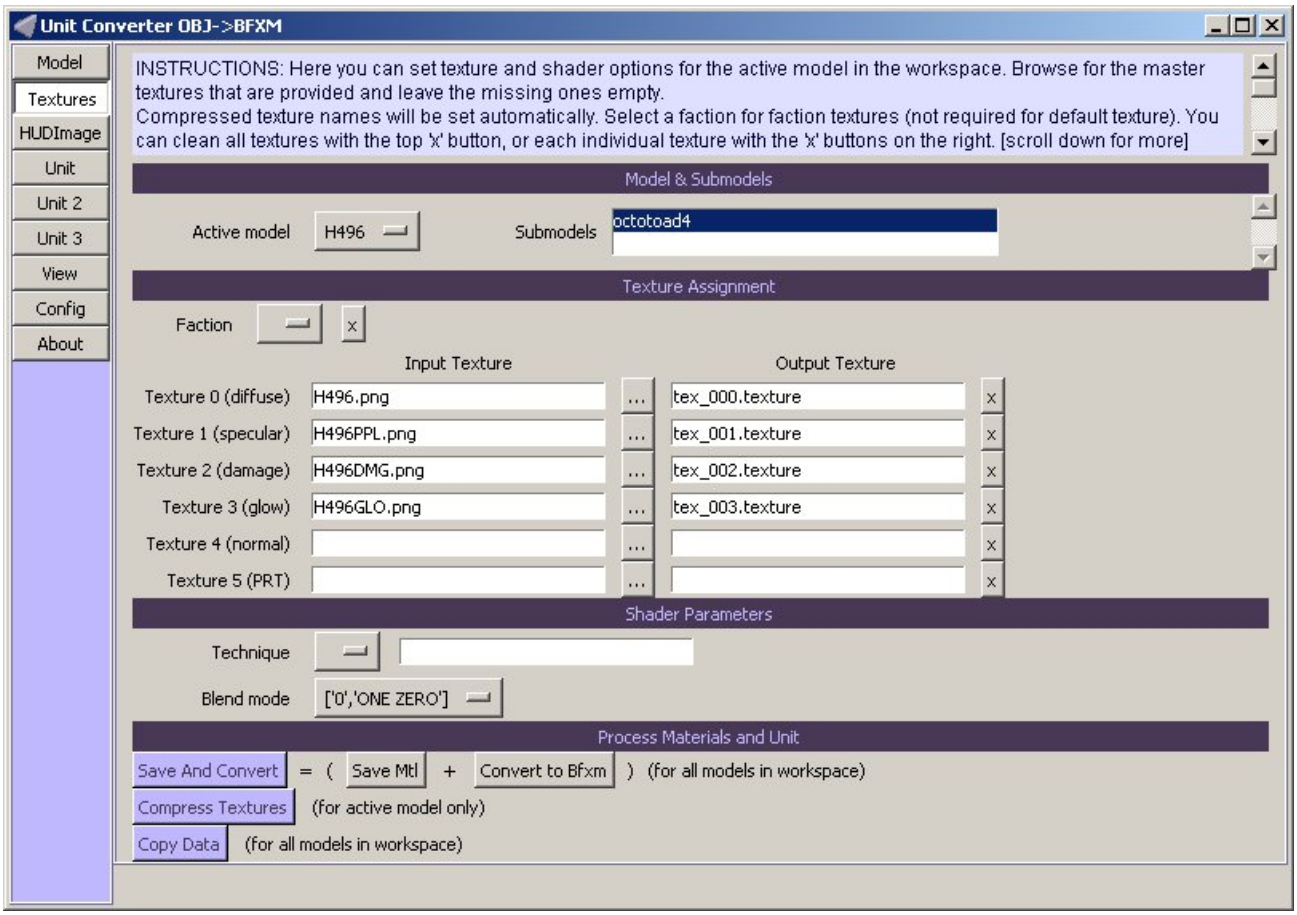
It is important that your first model in workspace should be the main model, as further information like directory name and units.csv entry name will be derived from the main model name.

Under **Model/Files** the *.obj* and *.mtl* files for the active model will be displayed. Under **Material Info** you can find information on the *.obj* file's material library and material names.

To edit texture assignments, proceed to the **Textures** tab on the left.

Textures

Textures Screen:



In this tab you can assign textures to model submeshes:

- Under **Submodels** select the submesh you'd like to assign textures to. By default the first submesh is selected when the model is activated.
- Add the available **Textures 0-5** by pressing the **...** buttons between the **Input Texture** and **Output Texture** columns and browse to the appropriate file. The output texture name will be set (naming convention: **tex**_*model_submodel_texture*.**texture**)
- Select the appropriate **Technique**, i.e. leave it empty for conventional shading approach or choose e.g. *cinemut_opaque* if your input textures are provided with CineMut shader family compatible packing.
- Select **Blend mode**.
  - For main meshes leave it at *ONE ZERO*
  - For shield mesh and blink light submeshes choose *ONE ONE*

Once you have assigned textures to all your submeshes, you can now start the conversion:

- Press **Save and Convert** to save the texture assignment information and convert your model to Vega Strike compatible *bfxm* format.
- You may **Compress Textures** using nvcompress if you have configured the command path previously
- Finally, '*Copy Data* will copy textures and *bfxm* files to the unit directory

To add the HUD image, proceed to the **HUDImage** tab on the left.

Advanced Usage - Converting faction textures

Faction textures must follow the naming of the default textures with prefixed faction name. When your default diffuse texture is *dif.png* and *tex_000.texture*, the faction input and output textures for the Uln faction will be *uln_dif.png* and *uln_tex_000.texture* and so on for all the other textures and factions.

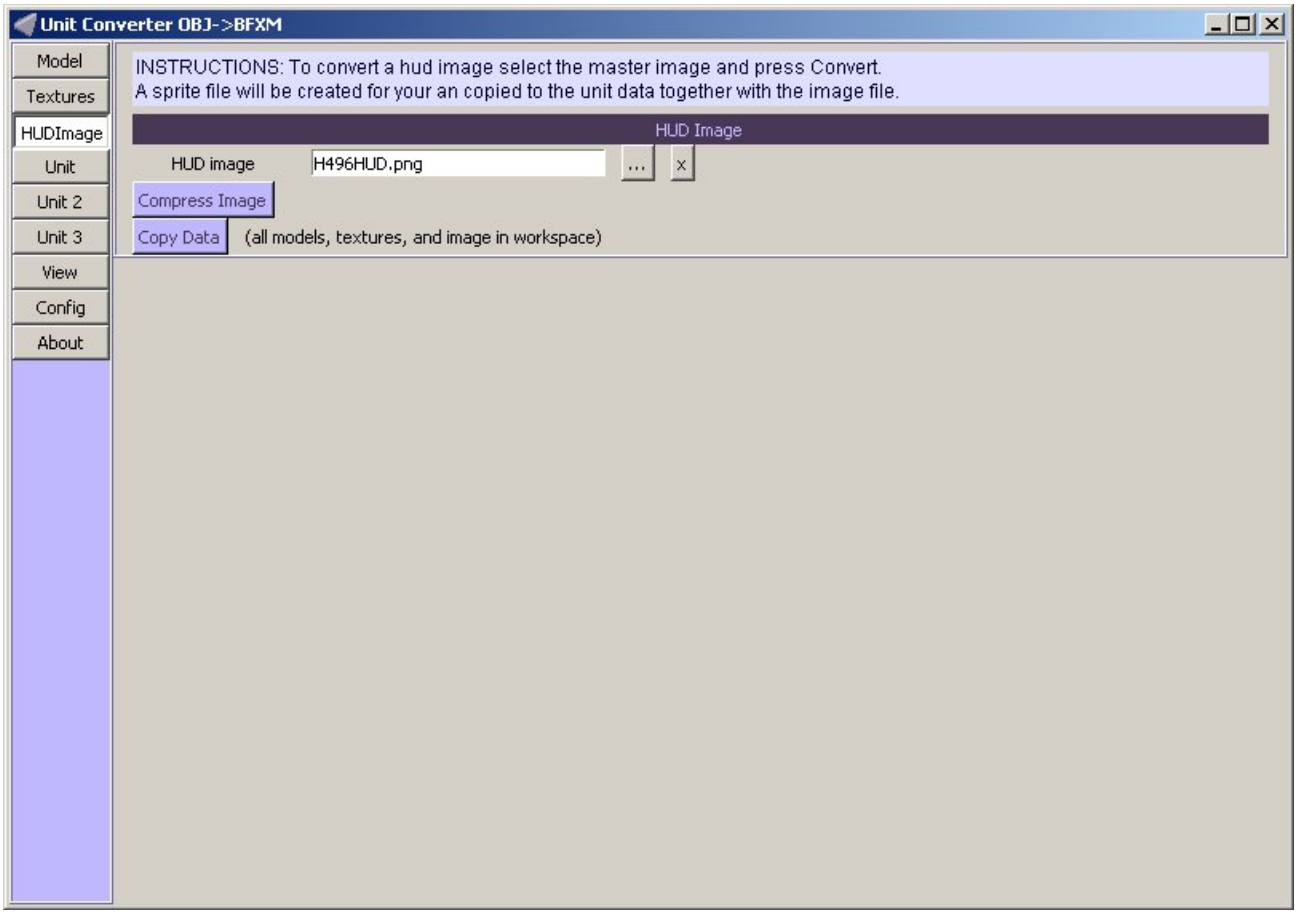
Unit converter can assist you with DDS conversion of the input textures provided the naming convention is

followed. To convert a faction texture:

- Select the **Faction** from the drop down, then
- **Compress Textures**, and when done
- **Copy Data** to put the faction textures into the unit data directory

HUD Image

HUD Image Screen



Adding a HUD image to the data is fairly easy, simply:

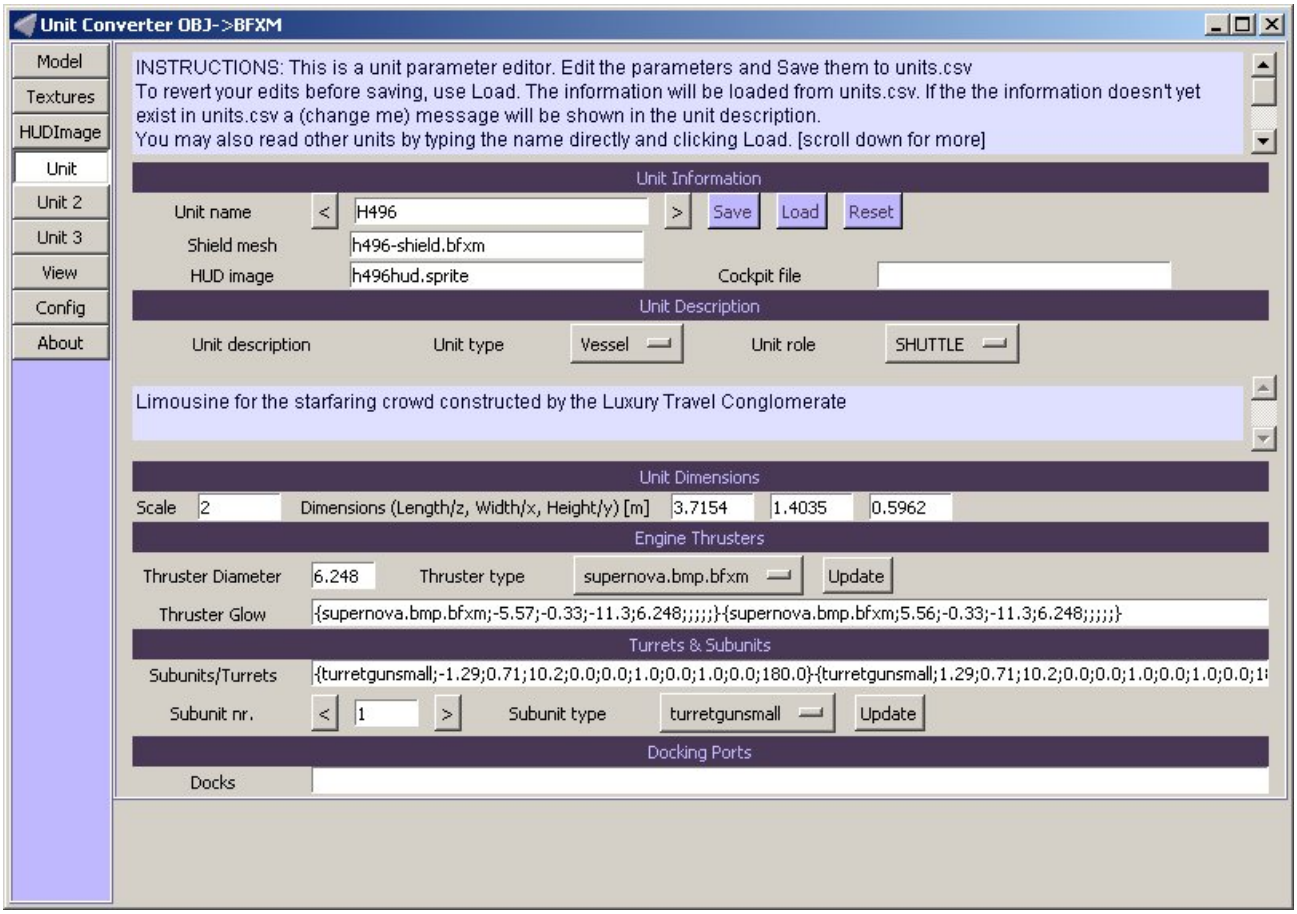
- Browse for the image using the **...** button. The appropriate entry will also be created in the **Unit** editor.
- **Compress Image** to make it DDS, and
- **Copy Data** to the unit data directory. This button evokes the same command as the one on the **Textures** tab. This means, the *image*, *sprite*, *bfxm*, and *texture* files will be copied together.

To edit the unit properties and parameters, proceed to the **Unit** tab on the left.

Unit Editor

The Unit Editor provides facilities for editing your most important parameters of the unit:

- **Load** - loads the unit parameters
- **Save** - saves the edited parameters
- **Reset** - reloads unit parameters for active model without previously saving changes



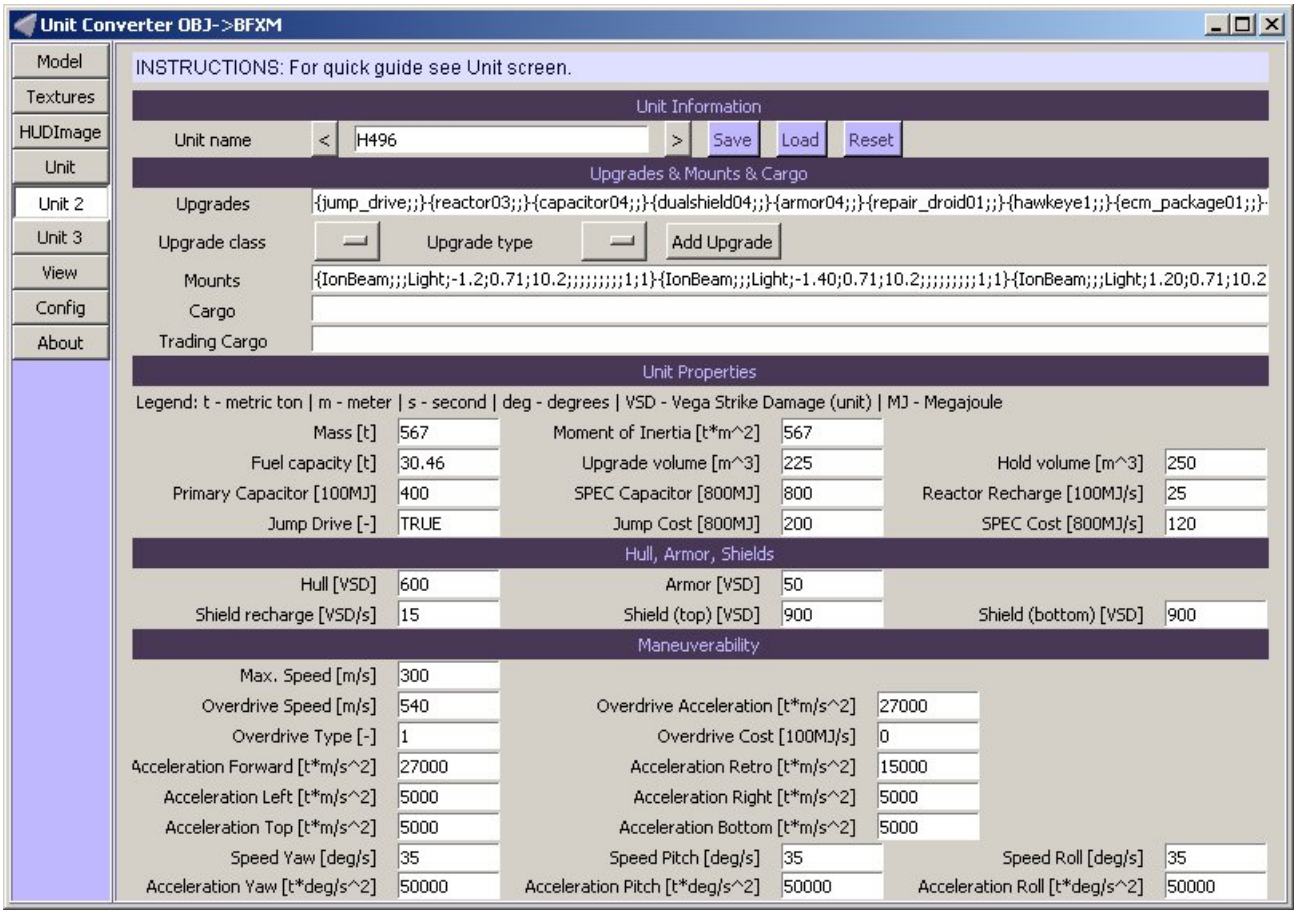
The most critical parameters for successful integration of the unit are:

- **Unit description** - the text that will be displayed in-game about the unit
- **Unit type** and **Unit role** - used in-game and to correctly view vessels and installations as such
- Under **Unit dimensions** you will find the
  - **Dimensions** as derived from the *obj* mesh file.
  - To bring the model up to the expected scale, you'll need to set the **Scale** parameter. The final size of the model in-game will be the multiplication result of dimensions and the scale.

The remaining entries will be created from your marker object:

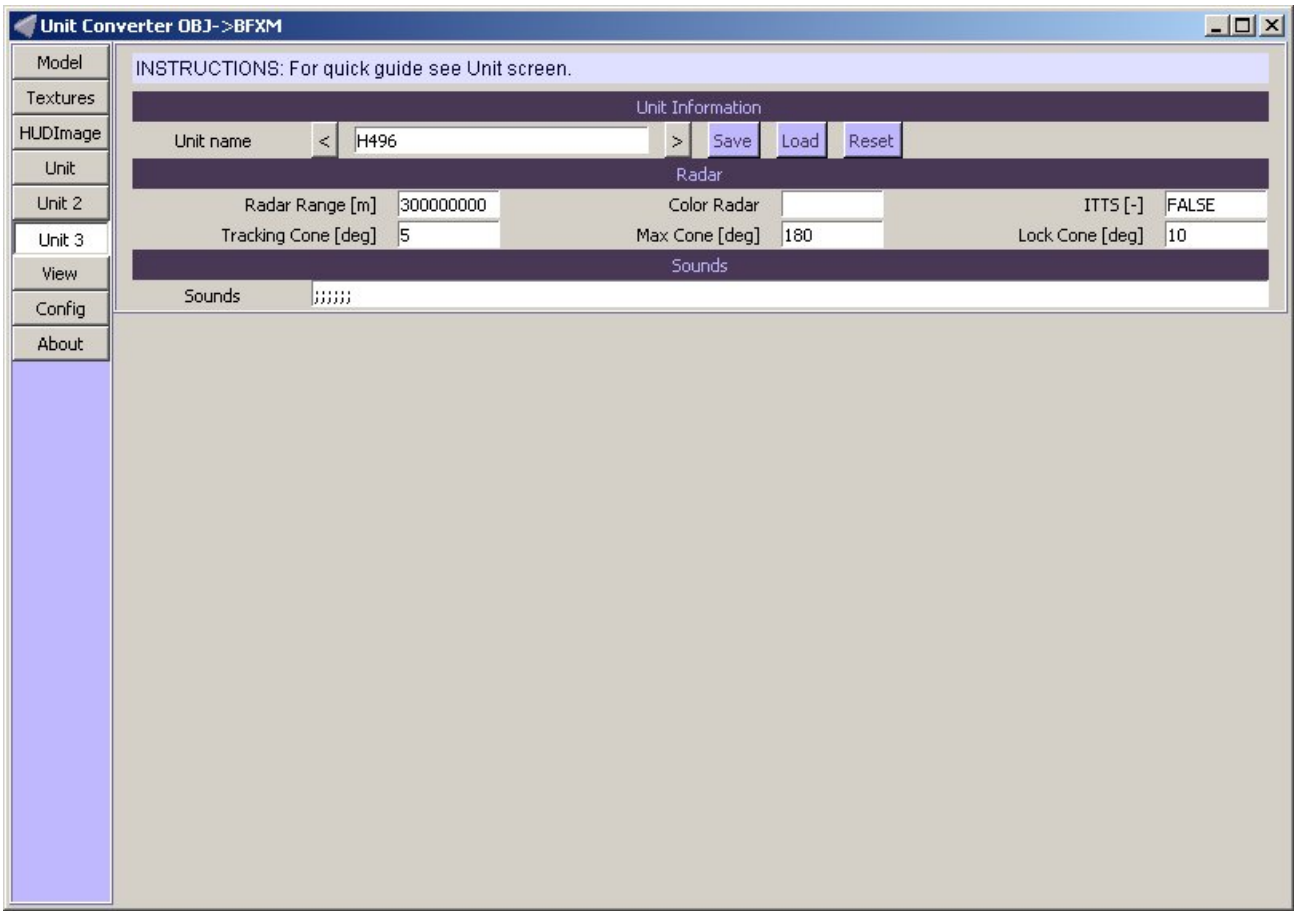
- **Engine Thruster** will be derived from markers named *engine_** . You may **Update** the **Thruster type** to a different than the default one (*supernova.bmp.bfxm*).
- **Turrets & Subunits** will be derived from markers named *turret_** or *subunit_** . You may browse the **Subunit nr.** (number) by using the **<** and **>** buttons and change the turret type from the **Subunit type** dropdown and confirmation by **Update**.
- **Docking Ports** will be derived from markers named *dock_**

To continue editing unit parameters, proceed to the **Unit 2** tab on the left.



To add upgrades, under **Upgrades'** ***select the*** *Upgrade class* ***and*** *Upgrade type* ***and then*** *Add Upgrade* to the unit. To delete an upgrade, simply remove the entries between (and including) the {} brackets.

All other parameters on this tab and the following **Unit 3** tab can be manually edited



Don't forget to actually **Save** your changes before proceeding to **View** the unit in-game.

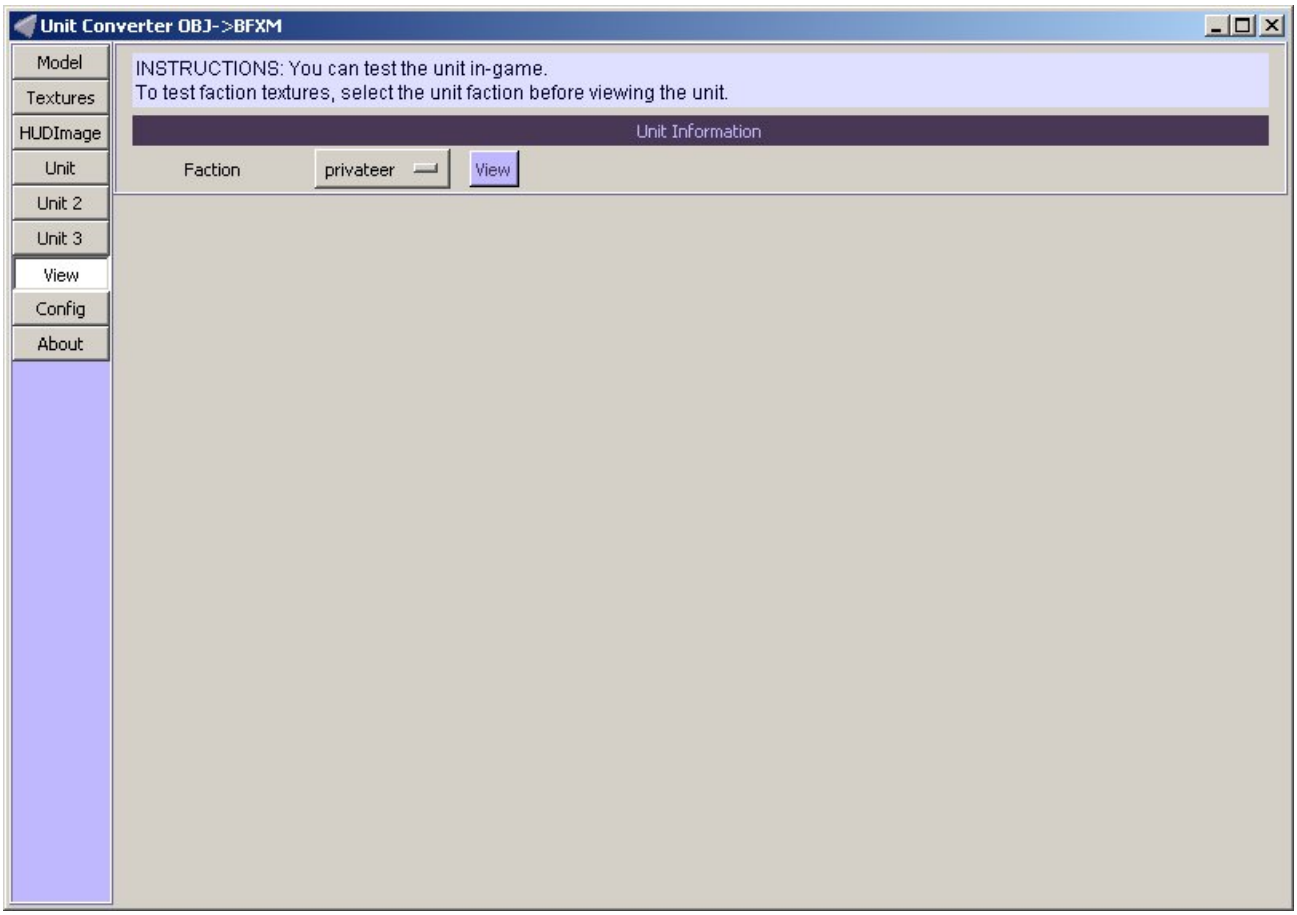
Advanced Usage - Independent Unit Editor

You can use built-in Unit editor of the **Unit Converter** to edit units without having to load the mesh files. To do so:

- On the **Model** tab do **Clear Workspace**
- On the **Unit** tab, manually write the unit key under **Unit name**
- **Load** the unit parameters

You can also browse the units.csv file by using the **<** and **>** buttons on the sides of the **Unit name** to row-wise move back and fore between the entries. Browse, **Load**/**Save** buttons are also available on **Unit 2** and **Unit 3** tabs.

View



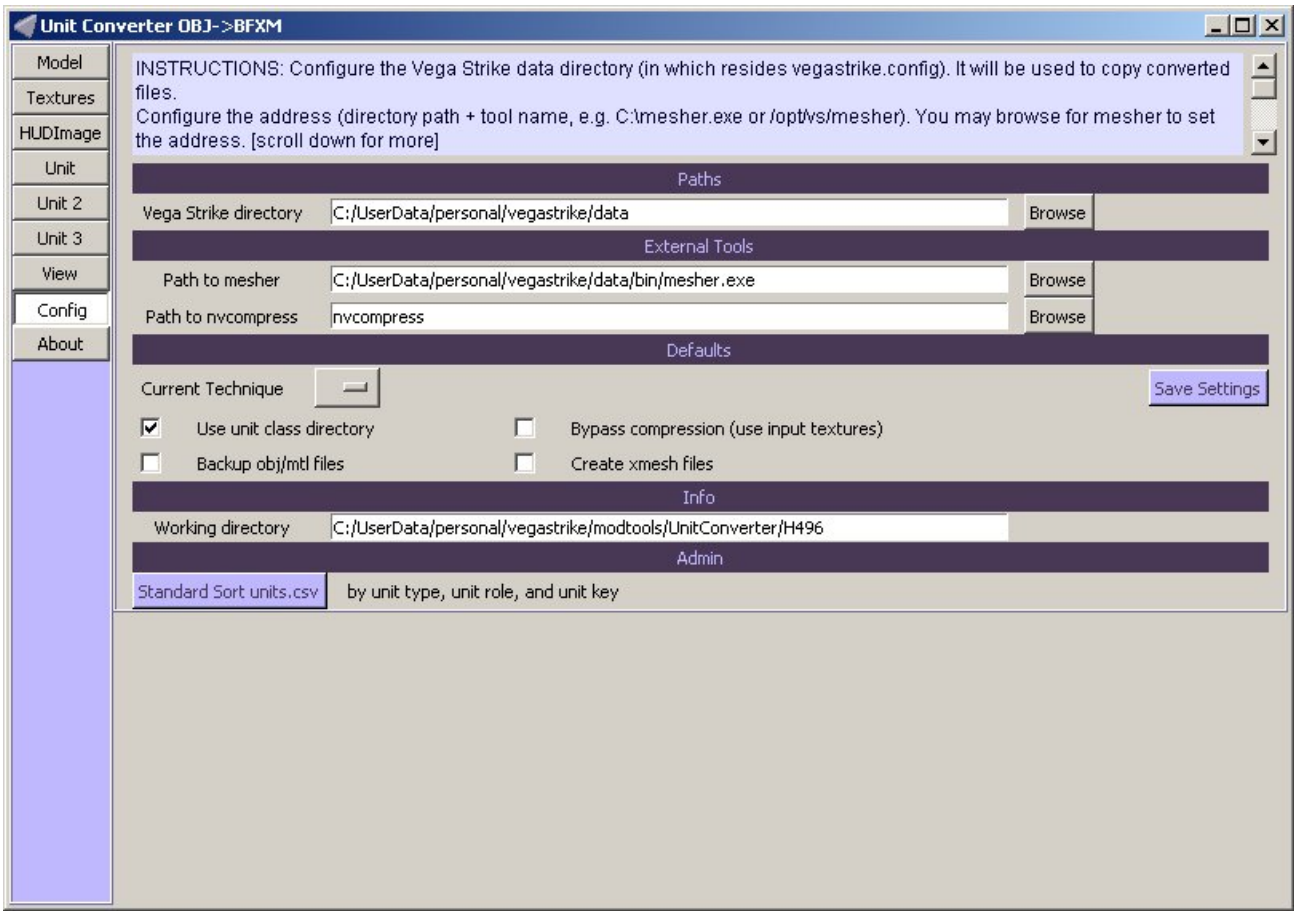
By pressing **View** Vega Strike can be loaded and you can test your unit in-game.

To test faction textures (if provided) you may previously select the **Faction** from the drop down, otherwise leave it at *privateer* to use default textures.

Testing of units will happen in two different ways:

- For vessels, using a special mission, the *modelview.mission* that spawns the unit as a ship.
  - Use the **F6** key to view your own unit.
  - Use the **T** key to target a second unit of the same type and then press the **D** key to stop it from SPECing away, then turn around to it to verify the docking ports. Repeat pressing **D** every 10 seconds to prevent it from flying away suddenly.
- Use the **N** key to target the nearest station, fly to it and start shooting while switching to **F6** view to test your shield mesh.
- For installations, the *Modelview.system* will be equipped with the edited station.
  - Use the **N** key to target the station, then **D** to see the docking ports.

Configuration



The **Config** tab defines presets to supports some of the functionality of the **Unit Converter**:

- **Vega Strike directory** - is the directory where *Version.txt* and all data subdirectories reside (*ai*, *animations*, ...)
  - Use the **Browse** button to locate it
- **Path to mesher**:
  - Click the **Browse** button on the right to locate *mesher*
  - Find the *mesher(.exe)* file on your system, select the file and confirm with **Open**. The configuration will be saved so that you need to do it only once.
  - The path must be the complete address (directory path + executable) in the same way you would evoke it from command line / console. On windows it may or may not include the *.exe* extension suffix.
- **Path to nvcompress**:
  - Click the **Browse** button on the right to locate *nvcompress*
  - Find the *nvcompress(.exe)* file on your system, select the file and confirm with **Open**. The configuration will be saved so that you need to do it only once.
  - You will not require to install *nvcompress* if you just want to test your models. In this case please make sure you enable the **Bypass compression** option further below.
  - Techno-talk: The path must be the complete address (directory path + executable) in the same way you would evoke it from command line / console. On windows it may or may not include the *.exe* extension suffix. (If nvcompress is in your search path, you may only provide the command.)

Advanced Configuration

You may also change defaults that will be used anytime you reopen the Unit Converter:

- **Current Technique** - set the default technique you will be working with so that you don't need to

remember adjusting it when editing textures.
- **Use unit class directory** - used to calculate target for copying model and texture files.
  - when checked will store vessels in *data/units/vessels/unitname* directory and installations in *data/units/installations/unitname* directory. This is the default behavior for Vega Strike Upon The Coldest Sea game.
  - If your mod is using a flat directory structure, you can uncheck the option and all your units will be stored in *data/units/unitname* directory.
- **Bypass compression** - when checked the input texture will be used as output texture. This is useful when editing and testing textures without the need to reconvert the model to bfxm.
- **Backup obj/mtl files** - when checked will copy original *obj* and *mtl* files to *obj~* and *mtl~* before writing new information.
- **Create xmesh file** - though the xmesh format is obsolete, when converting from *obj* to *bfxm*, it might be useful to also create *xmesh* files to test the conversion process.
- Under **Admin** you can **Standard Sort units.csv** file. Though it is standard sorted by default, you might want to make sure it is after a manual editing of the file.

About

What's that about?

Code Reference

Python Modules

Admittedly, the structure of the program could be more modular.

Unit Converter is written in Python using only *Tkinter* UI building functionality. On Windows Tkinter is distributed with Python, on Linux, it's available for download in all repositories. No further additional modules are required to run Unit Converter.

Source modules for Unit Converter:

- *unitconverter.py* - main module containing the complete program
- *appwindow.py* - frame creator for the different tabs
- *objparse.py* - *.obj* and *.mtl* file reader, parser, and writer
- *vsunitcsv.py* - *units.csv* file reader, parser, and writer
- *vsmission.py* - *.mission* file reader, parser, and writer
- *vssystem.py* - *.system* file reader, parser, and writer
- *vector.py* - mathematical vector functions
- *png.py* - *.png* file format reader

Other files:

- *unitconverter.ini* - Unit Converter configuration file. Not distributed. Will be created upon first start.
- *unitconverter.ico* - Windows icon file.
- *uc* - Linux shell script for starting Unit Converter
- *uc.bat* - Windows batch file (shell script) for starting Unit Converter
- *uc_releasenotes.txt* - Unit Converter release notes
- *UnitWorkingDir/UnitName.ini* - Unit workspace configuration file

Program Flow

The GUI building section starts with:

```
    def __init__(self, master):
```

The code is visually separated into:

- variable initialization
- GUI building sections
- program execution, start-up, and shutdown

The GUI building section for each tab contains elements that have event triggers which call other functions within the first part of the *unitconverter.py* code.

Further, there a unique watch function that supervises the change of some of the contents of GUI element variables for elements that, in Tkinter, do not come with an event handler (option drop downs, text variables, option check boxes):

```
    def processEvents(self):
```

Data Elements

There is a lot of data being handled within the Unit Converter. The main data elements shall be explained here:

- *workspace[modelnr][submodelnr][parameternr]* - array containing the complete workspace information, where
  - modelnr is the index from *models[modelnr]* which stores the model names in workspace
  - submodelnr comes from the list *submodelsListBox*
  - *PARAMS* - array holding the parameter names for *workspace*

- *textures[submodelnr][counter]* - holds the textures for the active model only prepared for the mtl export in the array form *["map_0","tex_000.texture"]*
- *unit** - variables holding the unit editor values
- *textureN* - with N = 0-5, holds the output texture screen name
- *texmasterN* - holds the input texture name
- *activemodel* - name of the active (selected) model in workspace
- *unitname* - name of the key from units.csv
- *vegastrike* - Vega Strike installation directory root
- *workdir* - working directory

Most of the variables are commented within the code.