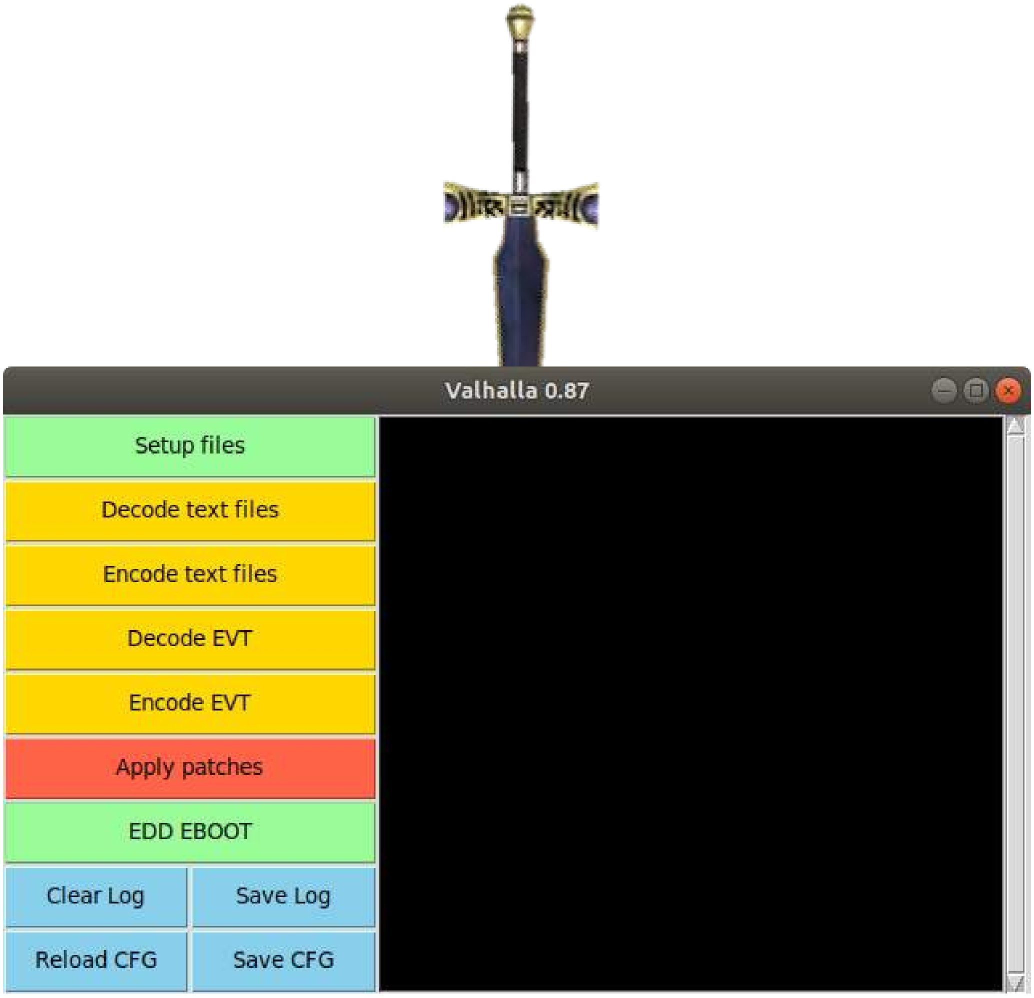
Quick Reference Guide

Rev. 0.87

Valhalla

Summary

Valhalla is a tool for editing the Final Fantasy Tactics psp game. (ASM, Text, Data and Events)



Contents

1) ASM

1.1) Overview

The PSP uses a MIPS R4000-based CPU. So all the games run with the MIPS instruction set.

FFT WotL, unlike other PSP games, was built from original FFT for PSX. So it shares with it many things (sprites, events instructions, data structures, etc).

In other words, FFT Wotl (from an ASM point of view) does all what the original game does and more. Plus the 32MB of RAM (vs the 2MB of PSX) gives this game an overall hacking advantage.

1.2) Notes for making ASM hacks

1.2.1) Disassembly the BOOT.BIN with Valhalla.

Use ‘EDD EBOOT’.

1.2.2) MIPS instructions supported by Valhalla:

noop	No operation
j I26	Jump with immediate 26 bit value
jal I26	Jump and link with immediate 26 bit value
jr S	Jump to register value
jr.hb S	Jump to register value with hazard barrier
jalr D,S	Jump and link register
jalr.hb D,S	Jump and link register with hazard barrier
beq S,T,I16	Branch if equal
bne S,T,I16	Branch if not equal
blez S,I16	Branch if less or equal zero
bgtz S,I16	Branch if greater than zero
bltz S,I16	Branch if less than zero
bgez S,I16	Branch if greater or equal than zero
bltzal S,I16	Branch if less than zero and link
bgezal S,I16	Branch if greater or equal than zero and link
beql S,T,I16	Branch if equal likely
bnel S,T,I16	Branch if not equal likely
blezl S,I16	Branch if less or equal zero likely
bgtzl S,I16	Branch if greater than zero likely
bltzl S,I16	Branch if less than zero likely
bgezl S,I16	Branch if greater or equal than zero likely
bltzall S,I16	Branch if less than zero and link likely
bgezall S,I16	Branch if greater or eq than zero and link likely
lui T,I16	Load upper immediate
lb T,I16(S)	Load byte
lbu T,I16(S)	Load byte unsigned
lh T,I16(S)	Load halfword
lhu T,I16(S)	Load halfword unsigned
lw T,I16(S)	Load word
lwl T,I16(S)	Load word left
lwr T,I16(S)	Load word right
sb T,I16(S)	Store byte
sh T,I16(S)	Store halfword
sw T,I16(S)	Store word
swl T,I16(S)	Store word left

swr T,I16(S)	Store word right
mfhi D	Move from HI
mflo D	Move from LO
mthi D	Move to HI
mtlo D	Move to LO
movz D,S,T	Move if zero
movn D,S,T	Move if not zero
addi T,S,I16	Add immediate
addiu T,S,I16	Add immediate (No trap OF)
add D,S,T	Add
addu D,S,T	Add (No trap OF)
sub D,S,T	Sub
subu D,S,T	Sub (No trap OF)
mult S,T	Mult
multu S,T	Mult (No trap OF)
div S,T	Divide
divu S,T	Divide (No trap OF)
madd S,T	Mult and add
maddu S,T	Mult and add (No trap OF)
msub S,T	Mult and sub
msubu S,T	Mult and sub (No trap OF)
andi T,S,I16	And immediate
and D,S,T	And
ori T,S,I16	Or immediate
or D,S,T	Or
xori T,S,I16	Xor immediate
xor D,S,T	Xor
nor D,S,T	Nor
slti T,S,I16	Set if less than immediate
sltiu T,S,I16	Set if less than immediate unsigned
slt D,S,T	Set if less than
sltu D,S,T	Set if less than unsigned
clz D,S	Count leading zeroes
clo D,S	Count leading ones
sll D,T,I5	Shift left logical
sllv D,T,S	Shift left logical variable
srl D,T,I5	Shift right logical

srlv D,T,S	Shift right logical variable
sra D,T,I5	Shift right arit
srav D,T,S	Shift right arit variable
ext T,S,I5,I5	Extract bits
ins T,S,I5,I5	Insert bits (buggy)
seh D,T	Sign extend halfword
seb D,T	Sign extend byte
wsbh D,T	Word swap bytes between halfwords

1.2.3) Valhalla special instructions:

noop*X	No operation X times
encodeText;	Encode from ‘;’ to the end of line as text
encodeBits;	Encode 8 bits after ‘;’
encodeHalf;	Encode 4 hex digits after ‘;’
$-name:	Patch separator
$-uuid:	Used to identify the patch
$-requires:	To specify if requires other patches
$-define:	To create special references like: %Ref_Name,value These will be valid for the current patch.
$-description:	Used to write comments
$-file:	Specifies the file to write. Only two options: ‘boot.bin’ or ‘fftpack.bin’
$-type:	Specifies the type of addressing. Four options: ‘ram’ or ‘file’ or ‘path to file’ or ‘relative’ There is fifth option, used to clear space: ‘NULL_FILE’
$-offset:	This sets the starting point to write and/or the ending point. If ‘ram’ or ‘file’ or ‘path to file’ are used, then just write the numerical value of the address in hexadecimal. Attach ‘&INDEXED’ if required. Else if ‘relative is used’ write base offset, attach ‘&INDEXED&’ if required, and write the extra offset required. Else for ‘NULL_FILE’ just write the numerical value of the address in hexadecimal and attach ‘&INDEXED’ or ‘file_size(decimal)’
xxx:	It creates an special reference ‘xxx’. It will be valid for the current location.
@xxx:	It creates an special reference ‘@xxx’. It will be valid for the entire file.

1.2.4) Valhalla default asm hacks:

Type	Name	US/EU	JP	Notes (*)
Core	Slowdown fix v2	✓	✓	
	Add spellquotes back to the game	✓	N/A	
	Spell quotes always pop up	✓	☒	
	Unlock novels	✓	N/A	
	Smart Encounters	✓	✓	
	Battle Initial Camera v2	✓	✓	
	Fix gil amount needed for the lip rouge quest	✓	✓	
Text	Remove dupe font data	✓	✓ *	Partial implementation.
	Replace font file ENG version	✓	N/A	
	Replace font width file ENG version	✓	N/A	
	Replace spell.mes file	✓	☒	
	Replace snplmes.bin file	✓	☒	
	Replace wldmes.bin file	✓	✓	
	Replace all tutoX.mes files	✓	✓	
	Rewrite all quick access text	✓	☒	Fixed.
	Change tutorial soldier names	✓	✓	
Data	Replace all data files	✓	✓	
Custom	Unlocked jobs v2	✓	☒	
	Battle jobs list	✓	☒	
	Formula 36 is now +PA_(X) and +MA_(Y)	✓	☒	
	Mighty sword v2	✓	☒	
	Crits always deal 50 percent bonus damage	✓	☒	
	Abilities in Arith skillset can be reflected	✓	✓	
	Null slps file in fftpack	✓	☒	
	Replace TEST.EVT file	✓	✓	
	Treasure Hunter 9-bit rare items	✓	☒	
	ENTD 9-bit equipment items	✓	☒	
	Balthier gets no special ENTD items	✓	☒	
	Poach 9-bit items	✓	☒	
	Onion items can be equipped if lip rouge flag	✓	☒	
	Treasure Hunter rare minimum brave	✓	☒	
	Arithmeticks supports up to 20 skillsets	✓	☒	
	Party Roster Extension (PRE)	✓ *	☒	Not fully tested.
	Switch formation and absorb	✓	☒	

	Monster eggs in last roster slot only	✓	❌	
	Level cap	✓	❌	
	Monster do not count as Casualties nor Injured	✓	❌	
	Treasure Hunter is Player only	✓	❌	
Experimental	Test entd 9b items	✓ *	❌	For testing purposes.

2) Text (no events)

2.1) Overview

All the text for this game (without events) is in 4+1 folders (Data/patches/files/):

a) wld-mes	Big text file with 121 hard sections.
b) spell-mes	One section file with 512 entries for spell quotes
c) snpl-mes	Novels text file with 6 hard sections.
d) tuto-mes	7 one section files for the first tutorial menu.
c) rest-mes	Contains all dispersed and duped text entries.

2.2) Text compression

FFT natively supports (for both PSX and PSP) a type of text compression. This compression is done by 3 bytes word (starting with 0xF0, 0xF1, 0xF2 or 0xF3 or ‘111100’ in binary), that can point back up to 8KB and read up to 35 bytes.

Valhalla supports this type of compression in a read-only way.

FFT does not natively support DTE. However FFT uses a font file that contains around 2k2 symbols, and 256 symbols can be encoded in one byte. Since english only have around 26 lower case chars plus numbers, then it is possible to use around 120 symbols slots to place 2-4 chars while only using one byte to encode them. This is a type of byte pair encoding, which will help to compress text data.

Valhalla supports this type of compression and creates those 120 custom symbols. This not fully supported in the jp version.

2.3) Text formats and modifications

FFT has two types of text format hard sectioned files and header sectioned files.

The former ones have a fixed size and are filled with zeros (v.g. novels) and the latter ones have a 128 byte header which contains pointers for 32 fixed sections (v.g. LZW's).

Sadly FFT has dispersed and duped text all over the boot.bin and the fftpack.bin. Which makes this game painful for editing text.

Valhalla requires the 'default-text-hacks.asm' to fully edit all in-game text. This patch condense all FFT dispersed and duped text in 3 files.

2.4) Notes for editing text files

2.4.1) Open and save the files as 'utf-8'.

2.4.2) FFT special chars:

{0x3F}	Prints '...'
{Ramza}	Prints Ramza custom name.
{Wait Press}	Prints the white Hand and waits for player input.
{0xF5XX}	Used for sound effects.
{0xF6XX}	Used for sound effects.
{Newline}	Creates a new line in the text entry.
{LS}	Print a max width white space (LongSpace).
{Begin List}	Used for giving options to the player.
{End List}	Used for giving options to the player.
{Keep Open}	Used for dialogs
{MClose}	The text entry will be closed if player input.
{AClose}	The text entry will be closed after last char is printed
{Color XX}	Changes font color.
{Delay XX}	Default delay is 1. Is the time used for printing each char.
{0xE8XX}	Used in menus.
{0xECXX}	Used in menus.
{Tab XX}	Like the 'tab' key, but with XX size.

2.4.3) Valhalla special chars:

{zeroes(amount,offset)}	Fills with zeroes (0x00) up to offset+amount.
{pad(byte,offset)}	Fills with byte (0-255) from the start of line up to offset.
{file XX}	Used for rest-mes to sort text entries.
{section XX}	Used for rest-mes to sort text entries.
{section XX-s}	Used for rest-mes to split common section parts.
{section XX-e}	Used for rest-mes to split common section parts.
{event XXX}	Used for test-evt.
{event text}	Used for test-evt.
{format}	Used for spell-mes-0-utt.txt at the beginning of each line to set FFT special chars.

2.5) wld-mes

2.5.1) Summary:

This is FFT WLDMES.BIN file. It has 121 hard sections.

2.5.2) Contains:

-Shop text.
-Chronicle text.
-Rumors text.
-Tasks text.

2.5.3) Files (write mode only):

0.BIN	WLDMES.BIN (extracted)
wld-mes-0-unt	Decoded WLDMES.BIN

2.5.4) Files (read mode only):

wld-mes-0-utt	Editable decoded WLDMES.BIN

2.5.5) Files (read and write mode):

wld-mes-0.bin	Encoded 'wld-mes-0-utt'.

2.6) spell-mes

2.6.1) Summary:

This is FFT SPELL.MES file. It has only one hard section with 512 entries.

On both US and UE versions, this file is blocked.

2.6.2) Contains:

-Spell quotes.

2.6.3) Files (write mode only):

0.MES	SPELL.MES (extracted)
spell-mes-0-unt	Decoded SPELL.MES without text.
spell-mes-0-unt.txt	Decoded SPELL.MES text.

2.6.4) Files (read mode only):

spell-mes-0-utt.txt	Editable decoded SPELL.MES text.

2.6.5) Files (read and write mode):

spell-mes-0-utt	Rebuilded decoded SPELL.MES.
spell-mes-0.bin	Encoded 'wld-mes-0-utt'.

2.7) snpl-mes

2.7.1) Summary:

This is FFT SNPLMES.BIN file. It has 6 hard sections.

On both US and UE versions, sections 1-4 and 6 are blocked.

2.7.2) Contains:

-Meas novel.
-Nanai novel.
-Wiyu novel.
-Enavia novel.
-Scriptures.
-Test book.

2.7.3) Files (write mode only):

0.BIN	SNPLMES.BIN (extracted)
snpl-mes-0-01-unt	Decoded SNPLMES.BIN 1º section w/out text.
snpl-mes-0-01-unt.txt	Decoded SNPLMES.BIN 1º section text.
snpl-mes-0-02-unt	Decoded SNPLMES.BIN 2º section w/out text.
snpl-mes-0-02-unt.txt	Decoded SNPLMES.BIN 2º section text.
snpl-mes-0-03-unt	Decoded SNPLMES.BIN 2º section w/out text.
snpl-mes-0-03-unt.txt	Decoded SNPLMES.BIN 3º section text.
snpl-mes-0-04-unt	Decoded SNPLMES.BIN 4º section w/out text.
snpl-mes-0-04-unt.txt	Decoded SNPLMES.BIN 4º section text.
snpl-mes-0-05-unt.txt	Dummy file.
snpl-mes-0-06-unt	Decoded SNPLMES.BIN 6º section w/out text.
snpl-mes-0-06-unt.txt	Decoded SNPLMES.BIN 6º section text.

2.7.4) Files (read mode only):

snpl-mes-0-01-utt.txt	Editable decoded SNPLMES.BIN 1º sec text.
snpl-mes-0-02-utt.txt	Editable decoded SNPLMES.BIN 2º sec text.
snpl-mes-0-03-utt.txt	Editable decoded SNPLMES.BIN 3º sec text.
snpl-mes-0-04-utt.txt	Editable decoded SNPLMES.BIN 4º sec text.
snpl-mes-0-06-utt.txt	Editable decoded SNPLMES.BIN 6º sec text.

2.7.5) Files (read and write mode):

snpl-mes-0-01-utt	Rebuilded decoded SNPLMES.BIN 1º section.
snpl-mes-0-02-utt	Rebuilded decoded SNPLMES.BIN 2º section.
snpl-mes-0-03-utt	Rebuilded decoded SNPLMES.BIN 3º section.

snpl-mes-0-04-utt	Rebuilded decoded SNPLMES.BIN 4º section.
snpl-mes-0-05-unt	Decoded SNPLMES.BIN 5º section.
snpl-mes-0-06-utt	Rebuilded decoded SNPLMES.BIN 6º section.
spell-mes-0.bin	Encoded all 'snpl-mes-0-0X-utt' plus '5-unt'.

2.8) tuto-mes

2.8.1) Summary:

These are FFT TUTOX.MES files. 'X' goes from 1 to 7. Each has only one hard section.

2.8.2) Contains:

-The script for each sub-tutorial in the first tutorial menu.

2.8.3) Files (write mode only):

X.MES	TUTOX.MES (extracted)
tuto-mes-X-unt	Decoded TUTOX.MES

2.8.4) Files (read mode only):

tuto-mes-X-utt	Editable decoded TUTOX.MES

2.8.5) Files (read and write mode):

tuto-mes-X.bin	Encoded 'tuto-mes-X-utt'.

2.9) rest-mes

2.9.1) Summary:

Custom Valhalla text files that contains all the other non-event text. It is divided in 3 files:

rest-mes-0	8*128 byte headers plus {MClose}*28 section. (1428 bytes)
rest-mes-1	All non compressible text entries. (220264 bytes)
rest-mes-2	All compressible text entries. (209824 bytes)

2.9.2) Contains in 1:

-Jobs names.
-Skillset names.
-Abilities names.
-Items names.
-Tiles names.
-Tiles descriptions.
-Names dictionary.
-Menus (almost all, if not all).
-Tavern text.
-Etc.

2.9.3) Contains in 2:

-Jobs descriptions.
-Items descriptions.
-Abilities descriptions.
-Skillset descriptions.
-Locations descriptions.
-Jobs requirements.
-Help descriptions.

2.9.4) Files (write mode only):

X.BIN	RESTX.MES (extracted)
rest-mes-X-unt	Decoded RESTX.MES

2.9.5) Files (read mode only):

rest-mes-X-utt	Editable decoded RESTX.MES

2.9.6) Files (read and write mode):

rest-mes-X.bin	Encoded 'rest-mes-X-utt'.

3) Data

3.1) Overview

As almost any game, FFT has some data tables dispersed over the boot.bin and the fftpack.bin.

Valhalla is not intended to be a replacement of FFTPatcher for editing data tables, but it can apply the files from it.

3.2) data-pack

3.2.1) Summary:

This is just all the files extracted from a clean '.fftpatch' file.

3.2.2) Contains:

-All FFT data table files.

3.2.3) Files (read mode only):

pspItems pspItemAttributes moveFindItems abilities jobs skillSets monsterSkills actionMenus statusAttributes poaching jobLevels storeInventories propositions abilityEffects itemAbilityEffects reactionAbilityEffects abilityAnimations items inflictStatuses itemAttributes entd1 entd2 entd3 entd4 entd5	Editable binary files.

4) Events

4.1) Overview

Even though there is a great tool for editing FFT event file, I found it a bit slow and troublesome to open those 744 files.

With that in mind, Valhalla 0.83+ supports editing events through a single big file.

4.2) test-evt

4.2.1) Summary:

This is FFT TEST.EVT file. It has 744 events (10240 bytes each, 6144 in the jp version) .

4.2.2) Contains:

-All event data.

4.2.3) Files (write mode only):

TEST.EVT	TEST.EVT (extracted)

4.2.4) Files (read and write mode):

test-evt-unt	Editable decoded TEST.EVT (backup after)
test-evt.bin	Encoded 'test-evt-unt'.

5) Configuration file

5.1) Format

BASIC SECTION	
["$-patchFolder:", "../Data/patches/"],	**Description:** Set the ‘patch’ folder. **Value:** “../Data/patches/”
["$-filesFolder:", "files/"],	**Description:** Set the ‘patch files’ folder. It is relative. **Value:** “files/”
["$-disBOOT:", "../Data/boot-dis.txt"],	**Description:** Set the file for disassembly the BOOT file. **Value:** “../Data/boot-dis.txt”
["$-cleanISO:", "../Data/fft-wotl.iso"],	**Description:** Set the ISO file. **Value:** “../Data/fft-wotl.iso”
["$-cleanBOOT:", "../Data/boot-decrypted.bin"],	**Description:** Set the BOOT file. **Value:** “../Data/boot-decrypted.bin”
["$-cleanPACK:", "../Data/fftpack.bin"],	**Description:** Set the FFTPACK file. **Value:** “../Data/fftpack.bin”
["$-modBOOT:", "../Data/boot-patched.bin"],	**Description:** Set the modded BOOT file. **Value:** “../Data/boot-patched.bin”
["$-modPACK:", "../Data/fftpack-patched.bin"],	**Description:** Set the modded FFTPACK file. **Value:** “../Data/fftpack-patched.bin”
["$-modISO:", "../Data/fft-wotl-patched.iso"],	**Description:** Set the modded ISO file. **Value:** “../Data/fft-wotl-patched.iso”
["$-logFile:", "../Data/Valhalla.log"],	**Description:** Set the file to save the log. **Value:** “../Data/Valhalla.log”
["$-fontSize:",	**Description:** Set font size behaviour.

["disable", 14, 12]],	The first value sets if custom font size is ‘disable’ or not. The second and third value sets the button and text font size, only if not ‘disable’. **Value:** “disable”, “button_font_size”, “text_font_size”
["$-windSize:", "640x360"],	**Description:** Set initial window size. **Value:** “width x height”
["$-type:", "us"],	**Description:** Set the type of ISO to patch. “us” or “eu” values work for both regions, since “us” and “eu” versions were coded identically by SE. **Value:** “us”, “eu” or “jp”
["$-bkl:", "blacklist.asm"],	**Description:** Set the file for excluding asm hacks by UUID, that will be read from the patch folder. **Value:** “blacklist.asm”
["$-patches:", [["default-core-hacks.asm", "enable"], ["default-text-hacks.asm", "enable"], ["default-data-hacks.asm", "enable"], ["default-cust-hacks.asm", "enable"], ["default-expe-hacks.asm", "disable"]], ...],	**Description:** Set the patch files that will be read from the patch folder. They can be disable with a second value. The second list is for the jp version. **Value:** “name_of_the_file”, “enable” or “disable”
["$-DPFont:",	**Description:** Set if patched font will be used for decoding

false, false],	text. The second one is for the jp version. **Value:** false
["$-tFiles:", [["spell-mes", "enable", [[0]], 0], ["snpl-mes", "enable", [[1, 2, 3, 4]], 40960], ...],	**Description:** Set the text file folders that will be read and writen from the patch files folder when decoding/encoding text or event files. They can be disable with a second value. A third value indicates which section can be compressed, a zero means no section and a minus one means dummy file. The fourth values is the size of the section for that file. Zero means one section. And the fifth value is only for ‘rest-mes’ and contains offsets to make ‘rest-mes-0.bin’. The last one is reserved for the events. The second list is for the jp version. **Value:** “name_of_folder”, “enable” or “disable”, “[section_to_be_compressed, another_section_to_be_compressed]”, “size_of_section”, “[reserved_offset, another_ reserved_offset]”
ADVANCED SECTION	
["$-dTFiles:", [["0.MES"], ["0.BIN"], ["0.BIN"], ["1.MES", "2.MES", "3.MES", "4.MES", "5.MES",	**Description:** Set the text and event file names for each folder that will be written when building Valhalla requirements. The second list is for the jp version. **Value:** “reserved_name_of_file”

"6.MES", "7.MES"], ["0.BIN", "1.BIN", "2.BIN"], ["TEST.EVT"]], ...],	
["$-eTFiles:", [[["INDEXED", 172]], ...],	**Description:** Used for extracting the text and event files. The second list is for the jp version. **Value:** Reserved.

6) Graphical user interface

6.1) Layout

	➔ **Shorcut:** ‘F1’ ➔ **Normal output:** Extracting fftpack.bin: Done. Extracting EBOOT.BIN: Done. Decrypting EBOOT.BIN: Done. Extracting files: spell-mes 0.MES Done. Extracting files: snpl-mes 0.BIN Done. Extracting files: wld-mes 0.BIN Done. Extracting files: tuto-mes 1.MES 2.MES 3.MES 4.MES 5.MES 6.MES 7.MES Done. Extracting files: rest-mes 0.BIN 1.BIN 2.BIN Done. Extracting files: test-evt TEST.EVT Done. Extracting font glyphs file: Done. Writing font widths file: Done. Copying boot.bin: Done. Copying fftpack.bin: Done. Copying fft-wotl.iso: Done. Rebuilding font glyphs and widths: Done.

	Writing ENG font glyphs file: Done. Writing ENG font widths file: Done. --- 7.20 seconds ---
Decode text files	➔ **Shorcut:** ‘F2’ ➔ **Normal output:** CPU cores detected: 6 Decoding: spell-mes 0.MES Done. Decoding: snpl-mes 0.BIN Done. Decoding: wld-mes 0.BIN Done. Decoding: tuto-mes 1.MES 2.MES 3.MES 4.MES 5.MES 6.MES 7.MES Done. Decoding: rest-mes 0.BIN 1.BIN 2.BIN Done. --- 1.80 seconds ---
Encode text files	➔ **Shorcut:** ‘F3’ ➔ **Normal output:** CPU cores detected: 6 Encoding: spell-mes Rebuilding: Done. File: ../Data/patches/files/spell-mes/spell-mes-0 Section: 1 Done. Encoding: snpl-mes Rebuilding: Fixing newlines in: ../Data/patches/files/snpl-mes/snpl-mes-0-01 Fixing newlines in: ../Data/patches/files/snpl-mes/snpl-mes-0-02 Fixing newlines in: ../Data/patches/files/snpl-mes/snpl-mes-0-03 Fixing newlines in: ../Data/patches/files/snpl-

```
mes/snpl-mes-0-04
   Fixing newlines in: ../Data/patches/files/snpl-mes/snpl-mes-0-06
    Can't fix entry: 20.
   Done.
 File: ../Data/patches/files/snpl-mes/snpl-mes-0
 Section: 1
  Font Compression Success(7904).
 Section: 2
  Font Compression Success(8533).
 Section: 3
  Font Compression Success(3024).
 Section: 4
  Font Compression Success(2855).
 Section: 5
 Section: 6
 Done.
Encoding: wld-mes
 File: ../Data/patches/files/wld-mes/wld-mes-0
 Section: 1
...
 Section: 121
 Done.
Encoding: tuto-mes
 File: ../Data/patches/files/tuto-mes/tuto-mes-1
 Section: 1
 Done.
 File: ../Data/patches/files/tuto-mes/tuto-mes-2
 Section: 1
 Done.
 File: ../Data/patches/files/tuto-mes/tuto-mes-3
 Section: 1
 Done.
 File: ../Data/patches/files/tuto-mes/tuto-mes-4
 Section: 1
 Done.
 File: ../Data/patches/files/tuto-mes/tuto-mes-5
 Section: 1
 Done.
 File: ../Data/patches/files/tuto-mes/tuto-mes-6
 Section: 1
 Done.
 File: ../Data/patches/files/tuto-mes/tuto-mes-7
 Section: 1
 Done.
Encoding: rest-mes
 File: ../Data/patches/files/rest-mes/rest-mes-0
 Section: 1
 Section: 2
 Done.
 File: ../Data/patches/files/rest-mes/rest-mes-1
 Section: 1
 Done.
```

	File: ../Data/patches/files/rest-mes/rest-mes-2 Section: 1 Font Compression Success(55649). Done. --- 2.01 seconds ---
Decode EVT	➔ **Shorcut:** ‘F4’ ➔ **Normal output:** Decoding: test-evt CPU cores detected: 6 Done. --- 1.15 seconds ---
Encode EVT	➔ **Shorcut:** ‘F5’ ➔ **Normal output:** Encoding: test-evt CPU cores detected: 6 Event: 1 ... Event: 744 Done. --- 0.40 seconds ---
Apply patches	➔ **Shorcut:** ‘F6’ ➔ **Normal output:** Setting excluded hacks from: blacklist.asm Done. Applying patches in: default-core-hacks.asm: Applied. default-text-hacks.asm: Applied. default-data-hacks.asm: Applied. default-cust-hacks.asm: Applied. Done. Patching iso: /PSP_GAME/SYSDIR/EBOOT.BIN Patched. /PSP_GAME/USRDIR/fftpack.bin Patched. Done. --- 0.08 seconds ---
EDD EBOOT	➔ **Shorcut:** ‘F7’ ➔ **Normal output:** Extracting EBOOT.BIN: Done. Decrypting EBOOT.BIN:

	Done. Disassembling: boot-decrypted.bin CPU cores detected: 6 Done. --- 1.76 seconds ---
	➔ **Shorcut:** ‘F8’ ➔ **Normal output:**
	➔ **Shorcut:** ‘F9’ ➔ **Normal output:**
	➔ **Shorcut:** ‘Control’+‘D’ ➔ **Normal output:** [["$-patchFolder:", "../Data/patches/"], ...]
	➔ **Shorcut:** ‘Control’+‘S’ ➔ **Normal output:** Successfully saved. ➔ **Abnormal output:** Bad CFG. Configuration file loaded. ➔ **Abnormal output (2):** Bad CFG. Default configuration loaded.

7) Frequently asked questions (FAQ)

7.1) What is Valhalla?

In Norse mythology, is an special hall in Asgard. Joking aside, this “Valhalla” is a software, written in Python, that lets the user replace binary chunks of data/code for the psp game Final Fantasy Tactics WOTL with ease.

Nothing more nothing less.

7.2) So if i want to mod the WOTL psp game, this tool will help me?

Probably yes.

7.3) I started modding with the FFTPatcher, can I use it with Valhalla?

Yes and no. Since Valhalla is a binary chunk replacer, it only reads the original game images (US, EU and JP).

In other words, Valhalla doesnt even read the modifications done with it. That is why you should work with patch, data and text files. They are the source code of your modification.

7.4) What is the point of Valhalla if it can’t read its own modifications?

As I already wrote, Valhalla is a binary replacer for the original FFT WOTL psp game.

Valhalla doesn’t read its own modifications, because it is not an editor like FFTPatcher and for that reason it is not limited either.

When you apply asm hacks like the default text modification, you are changing the structure of the game code. Though its compatible with vanilla saves, the game is around 85% the same. And that is just with those hack examples.
So if an user wants to modify 40% of the game structure Valhalla can help that user, but won’t help reverse engineering those modifications by itself.

7.5) Applying all the defaults hacks sometimes bugs the game when saving?

Yes, prior version 0.87 there was a bug in the default text hack, that made the game randomly crash after a battle when saving.

7.6) The quotes sometimes go over the line limit and then can't be displayed in game?

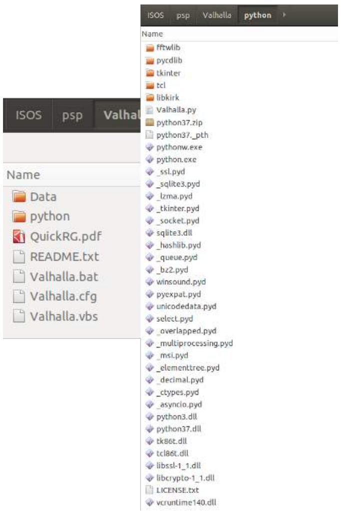
Well indeed that can happen if messing around without knowing anything. For this issue, there is special char “{format}” that must be placed at the beginning of each line in the quotes text file, to handle custom format for quotes.

7.7) I am a windows user, how do I run this app?

You need to download ‘Python (3.6+)’.

Install it (system-wide) or just download a portable archive.

Your Valhalla folder should look like this (see picture at right):



ISOS psp Valhal

Name

Data
python
QuickRG.pdf
README.txt
Valhalla.bat
Valhalla.cfg
Valhalla.vbs

ISOS psp Valhalla python

Name

fftwlib
pycdlib
tkinter
tcl
libkirk
Valhalla.py
python37.zip
python37._pth
pythonw.exe
python.exe
_ssl.pyd
_sqlite3.pyd
_lzma.pyd
_tkinter.pyd
_socket.pyd
sqlite3.dll
_hashlib.pyd
_queue.pyd
_bz2.pyd
winsound.pyd
pyexpat.pyd
unicodedata.pyd
select.pyd
_overlapped.pyd
_multiprocessing.pyd
_msi.pyd
_elementtree.pyd
_decimal.pyd
_ctypes.pyd
_asyncio.pyd
python3.dll
python37.dll
tk86t.dll
tcl86t.dll
libssl-1_1.dll
libcrypto-1_1.dll
LICENSE.txt
vcruntime140.dll

7.8) I got this error: “No such file or directory: '../Data/fft-wotl.iso'”. How do I fix it?

You need to place your Final Fantasy Tactics WotL image in Valhalla ‘Data’ folder and rename it as ‘fft-wotl.iso’.

You can change this in ‘Valhalla.cfg’ as you like.

7.9) Where do I get a Final Fantasy Tactics WotL image?

Buy it or borrow a copy from a friend. Valhalla supports the PSN version too.

7.10) What about the JP version?

Even though Valhalla supports the jp version, not all current default asm hacks are available for it.

7.11) Why ‘Python’?

Because, even if it’s not as good as C, it’s an amazing language. Fast, portable, compatible with C and mostly interpreted. IMHO better than C++ and Java (and way better than C#, NET, etc). Yet indentation is a pain without an IDE.

7.12) Is ‘Valhalla’ licensed?

Yes, it is licensed under WTFPL. Python and the libraries used have their own licenses.

7.13) If I want to support you, how can I do it?

By supporting ROMhacking.net