Guida completa alla traduzione di ROM in lingua straniera

Autori: D-Chan, mickey, Fraka, Duke, BGies, MJL, _Ombra_
Versione: 3.0
Ultimo aggiornamento: 29/01/2002

Dannazione, non esiste nessun fottuto manuale per le traduzioni! Nessun nome, nessuna tecnica! Soltanto un ammasso uniforme di valori esadecimali legati da regole decise da quei pazzi dei programmatori, e la vostra intelligenza!!

- Introduzione -

Intendete tradurre una ROM, o semplicemente volete modificarne il contenuto, ma non avete la più pallida idea di come si faccia, non siete pratici di editor esadecimali o cose del genere e soprattutto non sapete come è fatto l'interno di una ROM?

Allora iniziamo dalle basi. Il testo di una ROM, salvo in rari casi, non utilizza lo standard adottato dal DOS e da Windows, il codice ASCII. In questo codice (come in tutti quelli esistenti anche nelle ROMs), ogni lettera corrisponde ad un numero. Per esempio, la lettera A in ASCII corrisponde al numero esadecimale $41, la B al $42 e così via (i numeri esadecimali si indicano generalmente col simbolo "$" che li precede, ma per comodità in questa guida li lasceremo indicati senza). Una ROM ha anch'essa ogni lettera associata ad un numero, che generalmente viene espresso in numeri esadecimali (per chi lo ignorasse, il sistema esadecimale è un sistema di numeri in cui si va da 0 a 15, dove i numeri dal 10 al 15 vengono indicati con le lettere dalla A alla F).

Un tempo era necessario procurarsi un Hex Editor, cioè un editor esadecimale, ed imparare a memoria il codice della ROM che si traduce. Quindi per tradurre la parola "Arma" in FF3 era necessario cercare i numeri 00 2B 26 1A e sostituirli con quelli equivalenti alla parola desiderata. Fortunatamente adesso esistono delle utility che visualizzano e permettono di modificare il testo come se fosse in ASCII (o quasi) e rendono molto più semplice la modifica della ROM, anche se non riducono certo i problemi di spazio che si incontrano.

Già, per chi non lo sapesse in una ROM non si può superare il numero di caratteri in una riga presenti nella versione originale. Questo rappresenta il più grosso problema di traduzione, specie nelle ROMs in Giapponese (anche se un metodo per aumentare lo spazio per il testo esiste, ma è estremamente complesso). Comunque ricordatevi che è meglio cambiare di molto il testo anziché renderlo più vicino all'originale ma facendolo diventare difficilmente comprensibile.

La grafica in una ROM è sempre gestita in tile, ma di questo parleremo dopo. Premessa: ricordate che in questa guida le lettere della ROM verranno definite "font" oppure "tile". Le tile ad ogni modo non contengono esclusivamente il testo ma anche la grafica, per cui non fate confusione (vedere il capitolo "**I FONT**" per ulteriori delucidazioni).

Detto questo... buona lettura! :)

- La modifica del testo -

Per iniziare a tradurre una ROM è necessario conoscere la sua tabella dei fonts, ovvero le corrispondenze valore esadecimale -> carattere.
Chi ha un minimo di conoscenze informatiche saprà per certo che i dati sui calcolatori vengono

memorizzati ed elaborati sotto forma di bit (cioè una sequenza di 0 e di 1), ma che per comodità vengono letti usando il sistema numerico esadecimale (se non lo conoscete cercate documentazione online a riguardo).
Per memorizzare del testo, vengono decise una serie di convenzioni, secondo le quali a ogni byte corrisponde una lettera dell'alfabeto. Ad esempio, nel codice ASCII (utilizzato praticamente da tutti i computer moderni) il numero 41 corrisponde alla lettera "A", il 42 alla "B", etc. Ovviamente anche gli editor esadecimali usano questo sistema, purtroppo i giochi per console no. I creatori di videogiochi infatti il più delle volte creano delle tabelle nuove di zecca per ogni gioco rendendovi più difficile il lavoro, quindi prima di iniziare a tradurre dovete trovare la tabella dei font per quel gioco.
Questo è un esempio di tabella:

```
  0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1   A B C D E F G H I J K L M N O
2 P Q R S T U V W X Y Z a b c d e
3 f g h i j k l m n o p q r s t u
4 v w x y z 1 2 3 4 5 6 7 8 9 0 .
5 , : ; - ! ? =
6
7
8
9
A
B
C
D
E
F > <
```

La tabella si legge come un piano cartesiano: prendendo prima il numero della riga, poi quello della colonna troverete a che codice esadecimale corrispondono le varie lettere (in questo caso: A=11, B=12, 9=4D, >=F0).
Così se vogliamo trovare nella ROM la parola "Arma" basta cercare "11 3C 37 2B" e il gioco è fatto.
Questo metodo almeno si usava prima che esistessero editor come il Thingy...
La particolarità di questo editor infatti è che vi permette di modificare il testo direttamente da tastiera, come se fosse scritto in ASCII, basta che voi gli passiate come parametro la tabella dei fonts, cosi che possa fare la conversione.
Il Thingy ha molte altre feature utili, vi consiglio di leggere la sezione della **Guida all'uso del Thingy** per approfondimenti.

Come trovare la giusta tabella dei font

1) Metodo del confronto

Se il gioco che volete tradurre permette di cambiare il nome al proprio personaggio (Es: Chrono Trigger) procedete in questo modo:
Chiamate Crono con il nome "Crono", giocate e salvate. A questo punto rinominate il file del salvataggio chrono.srm in chrono.1. Ora iniziate un'altra partita e chiamate Crono "AAAAA", giocate e salvate. Rinominate chrono.srm in chrono.2 e fate un bel confronto tra i file chrono.1 e chrono.2 con il comando dos fc così:

```
fc /b chrono.1 chrono.2 > cfr.txt
```

Adesso lanciate edit cfr.txt e andate ad esaminare il file.

file: cfr.txt Confronto dei file chrono.1 e chrono.2 in corso...
0000059C: 01 02
000005B0: **A2 A0**
000005B1: **CB A0**
000005B2: **C8 A0**
000005B3: **C7 A0**
000005B4: **C8 A0**
000005E3: 0E 16
000005E4: 12 18
000005FA: 0A 0B
00001FF0: 73 29
00001FF1: 8C 42

Nel file chrono.2 si notano 5 caratteri uguali uno di seguito all'altro che corrispondono al nome "AAAAA", quindi la A = A0 e di conseguenza la C = A2, r = CB, o = C8, n = C7. Ora si può costruire la tabella delle fonts seguendo l'ordine alfabetico (se A = A0 allora B = A1, C = A2 e così via...).
Trucchetto: se A = 41, allora la ROM è in ASCII e si può modificare con un editor esadecimale.
Un'attenzione particolare va riservata ai caratteri che non compongono il normale alfabeto A..Z/a..z, come numeri, segni di punteggiatura o altri simboli particolari (come i disegnini vicino ai nomi degli oggetti in Final Fantasy e altri RPG). Per conoscerne il valore nella text table il più delle volte basta:

- Aprire la ROM in un editor grafico come Tile Layer Pro, Naga, Visor o XLate
- Trovare le tiles che compongono la text table A..Z/a..z
- Scovare i simboli e osservarne la posizione

Naturalmente se il simbolo che stiamo cercando, mettiamo sia il disegno di una faccina, si trova 2 tiles prima della A e il codice della A è A0, il codice per faccina è A0 - 2 = 9E.

Un altro sistema è invece quello di sostituire alla prima frase del gioco dei numeri esadecimali a caso, per vedere quali simboli escono fuori. Il piu' delle volte sarete costretti a fare così per scoprire tiles che non sono numeri o lettere, e quindi difficilmente rintracciabili in altro modo.
Se avete completato tutto l'alfabeto ma appunto vi mancano tiles come la punteggiatura o delle icone, potete trovarli ad intuito, studiandone la posizione. Giocate fino ad incontrarli, ad esempio trovate la frase:

Si, oggi e' una bella giornata! :)

Aprendo la ROM col Thingy cercate questa frase, vi si presenterebbe una situazione del genere:

Si<0B> oggi e<0C> una bella giornata<0E> <0F><10>

Ovviamente i bytes 0B,0C,0E,0F e 10 sono rispettivamente virgola, apostrofo/accento, punto esclamativo, due punti e parentesi ;)

2) Con il programma SearchR dei Dejap

Innanzitutto consiglio di procurarsi l'utility chiamata SearchR che potete scaricare alla sezione Utilità. Essa permette di cercare stringhe di testo all'interno della ROM.

Come esempio utilizziamo la ROM di Final Fantasy III, proviamo a cercare il nome della

protagonista Terra. Nella stessa directory mettiamo la ROM e il SearchR. Lanciamo SearchR, che dalla versione 2 presenta una comoda interfaccia, apriamo la ROM, e scriviamo "terra" nel campo "search for:". Come risultato otteniamo:

Offset: 0047C0
Hex Bytes: 93 84 91 91 80 FF 8B 8E
Ascii text: [.....]
Rel.Offset: a=80 A=?? 1=??
Rel.Text: [terra.lo]

Se hai giocato a FFIII saprai che quando ti viene chiesto di inserire il nome di un personaggio, il nome di default è proposto tutto in lettere maiuscole. Così i numeri trovati in precedenza corrispondono tutti a lettere maiuscole: 93 = T, 84 = E, 91 = R, 80 = A.

Se nella colonna con la scritta "Ascii text" comparirà lo stesso testo di quella "Rel. Text", significa che il testo nella ROM è in ASCII e pertanto è facilmente modificabile con un editor esadecimale. Stessa cosa se si trova A=41.

Se invece non trovate niente, tenete in considerazione i vari metodi di compressione e codifica spiegati più sotto, soprattutto quello a 16 bit.
In questo caso, quando fate le ricerche, semplicememnte scrivete le parole con un punto interrogetivo fra' le lettere, il ? corrisponde ad un carattere jolly che puo' essere qualsiasi byte hex. Per esempio la parola "Yes" la cercherete come "Y?e?s" ed il gioco e' fatto.

Per le compressioni come il DTE/MTE invece l'unico metodo efficace per scovarle è cercare per esempio una decina di parole, cosi' da riuscire a costruire una tabella. Aprire la ROM in un editor come il Thingy e leggere il testo che avete trovato con le tabelle che avete creato. A questo punto cercate di "intuire" quali sillabe/parole mancano e far corrispondere queste supposizioni ad 1 byte.

Per esempio, se nel Thingy vedete le parole "Mario#out", puo' voler dire che il byte # corrisponde al MTE " is ", quindi nella vostra tabella non fate altro che aggiungere "#= is " dove # corrisponde al byte hex (naturalmente il tutto senza le virgolette)

Metodi particolari di codifica del testo

Il metodo con cui è sistemato il testo cambia da ROM a ROM. Si basano tutte sul sistema delle tile, ma con delle varianti. Vediamole:

1) DTE/MTE

Lo spazio che una cartuccia mette a disposizione è sempre molto poco, e questo gli sviluppatori lo sanno molto bene. Una cartuccia per SNES arriva a non più di 4 Megabytes e considerando che sono decisamente costose, meno spazio occupava un gioco meglio era. Per questo i programmatori delle varie software house sviluppavano potenti routine di compressione dei dati, cosi da risparmiare quanto più spazio possibile. E mettere in crisi i ROM hackers, anche :)

La grafica è la parte più consistente, ma esistono metodi di compressione anche per il testo, e si basano tutti sull'assegnazione di piu' lettere a un solo valore esadecimale.
Il piu' famoso è il DTE della Squaresoft, in cui a un valore esadecimale corrispondono due lettere (l'acronimo significa infatti Dual Tile Encoding), le sillabe scelte sono quelle che compaiono con maggior frequenza nel gioco.
Quando a un solo valore esadecimale corrispondono più di 2 lettere, ci troviamo di fronte al cosiddetto MTE, che sta per Multiple Tile Encoding.

Un esempio di ROM che fa uso di MTE è Chrono Trigger, dove la scritta "the" corrisponde al byte n.21, "her" al byte n.30 e così via. La maggior parte di queste sillabe sono state sostituite dal traduttore, poiché in italiano non sono di nessuna utilità. Così, il byte 27 che in origine era "you" ora è "gli".
Questo però può causare dei problemi nel testo ancora in inglese, perché ora ogni volta che ci dovrebbe essere scritto "you" adesso c'è scritto "gli", per cui anche una parola come "yourself" diventa "glirself" ^__^
Questo è il motivo per cui è meglio scrivere nel "readme.txt" che il rimanente testo in inglese NON sarà più leggibile con la patch in italiano.

Per sostituire le sillabe, bisogna trovarle all'interno della ROM. Generalmente utilizzano la stessa tabella degli altri fonts, per trovarle potete cercarle col SearchR. Spesso sono separate da un altro byte, ad esempio in Chrono Trigger sono separate da un valore che rappresenta il numero di lettere che compone la sillaba: 02 se la sillaba è di due lettere). In Final Fantasy III invece è il valore esadecimale 7F e tutte le sillabe sono composte da due lettere. Quindi ponendo il caso in cui le tre tile consecutive fossero "he", "it" e "is", bisognerà scrivere he?it?is nella schermata principale del SearchR.

L'MTE può presentarsi in diverse forme:

1. Semplicemente, a un byte corrispondono 3 o più lettere
2. Dizionari: questa tecnica di compressione è analoga alla precedente, assegna più lettere a uno stesso valore esadecimale. La differenza sostanziale e' un byte "trigger" (grilletto, diverso da ROM a ROM) prima di ogni tile doppia, con lo scopo di segnalare che appunto ciò che segue non è una normale tile. Inoltre, questa routine viene utilizzata quasi sempre per parole intere (e non sillabe).
   I dizionari funzionano più o meno così: in un punto della ROM c'è un "dizionario" con le parole più usate, scritte di seguito e di solito separate da un byte, a cui si può accedere mettendo nel testo del gioco il byte trigger e di seguito il byte indice del dizionario.
   Ad esempio:

   Ciao come <00><A3>

   00 è il byte trigger, A3 indica la parola da usare. In un altro punto della ROM avremo:

   parola<byte fine>parola 2<byte fine>stai<byte fine>parola 3 etc

   E' possibile editare i dizionari, ma non si può superare il numero di byte originali, salvo difficili hacking a livello di assembler.
   Alcune ROM utilizzano i dizionari per cose particolari come i nomi dei personaggi o frasi che scompaiono dopo che il giocatore preme un tasto.
   Per esempio in Secret of Mana il nome dei vari personaggi è formato da due bytes: il byte n.57 e un byte che può essere 00, 01 o 02 a seconda di quale dei tre personaggi vi era scritto.

3. Substring: questo sistema si discosta un po' da quelli precedenti. Infatti, le substring sono dei puntatori! (se non sapete cosa sono, leggete l'ottima guida ai puntatori di Anus P..
   Anche in questo caso c'è un byte trigger, però vengono letti i 2 (o 3, dipende dalla ROM) bytes successivi, che sono appunto un puntatore. Questo significa che le substring permettono di puntare in un qualunque punto del banco, cosi da utilizzare eventuale spazio inutilizzato.
   Un esempio di ROM che usa le substrings (e anche i dizionari) è Terranigma della Enix, in cui il byte trigger per le substrings è CB.

NB: sebbene queste siano le tre varianti con cui più spesso si presenta il MTE ogni gioco è diverso dagli altri e potrebbe usare tecniche di compressione diverse, quindi non prendete per oro colato tutto quello che leggete. Usate la vostra intelligenza e fate esperimenti :)

Può capitare che le varie sillabe non abbiano la stessa tabella del testo (ovvero che questi byte speciali non siano una specie di comando per concatenare più lettere), ma che in ogni tile doppia siano effettivamente disegnati più caratteri. Questo genere di tile è generalmente chiamato "squished tiles" e l'unica soluzione per utilizzarle è fare un hacking della grafica.
Sconsiglio ai principianti di iniziare a tradurre una ROM che sfrutti tile multiple, per fare pratica è meglio iniziare con una più semplice.

2) La codifica a 16 bit

Non so se sia il nome esatto per questo metodo, mi sono basato sul readme del Thingy ^__^
In breve, tra una parola e l'altra viene interposto un altro byte. Per esempio, al posto di "YES" ci sarà scritto "YxExSx" dove "x" può essere un byte qualunque. Anche gli eseguibili per Windows sfruttano questo sistema, nei quali di frequente si può trovare le lettere separate da un punto (c.o.s.ì. .a.d. .e.s.e.m.p.i.o).
Pochi giochi, come per esempio Harvest Moon, utilizzano questo sistema, tenetelo però in considerazione quando con la ricerca relativa non avete nessun risultato, perchè il testo cercato (specialmente quello dei menù) potrebbe essere codificato in questo modo.

3) Interleaved Genesis ROM

Le ROM per Megadrive in formato SMD hanno una strana disposizione dei bytes nella ROM e di conseguenza anche delle lettere. Per questo motivo, non sono traducibili senza essere prima convertite in formato BIN. Diverse utility (uCON, GenConv, SegaTool) sono capaci di farlo. Ad ogni modo, il SearchR riesce a leggerle de-interlacciandole on-the-fly (la ROM comunque non subirà alcun cambiamento).

4) Byteswap

Le ROM per Nintendo64 dumpate con il Doctor64 (formato V64) sfruttano questo sistema. Praticamente i byte nella ROM sono invertiti a due a due (byteswapping), perciò la parola "MARIO " diventa "AMIR O". Procuratevi un'utility per convertire la ROM su **Dextrose** ("http://www.dextrose.com").

- La grafica delle console -

Il sistema a tile

Il SNES, così come la maggior parte delle console, disegna la sua grafica con il metodo delle tile (mattonelle), ovvero dei rettangolini di pixel che vengono uniti insieme in modo da formare una immagine. L'immagine originale viene scomposta in tasselli che sono memorizzati singolarmente nella ROM, nel momento di mostrare l'immagine a video tali tasselli vengono richiamati e ricomposti attraverso una mappa (tilemap grafica).

Ora verrebbe da chiedersi "perchè?". Senza dubbio disegnare le immagini pixel per pixel sarebbe stato più agevole (è il metodo usato dai nostri PC, per dirne una), soprattutto quando è necessario disegnare dei singoli pixel sullo schermo: il motivo fondamentalmente è lo spazio, è possibile riusare in immagini diverse le stesse tile risparmiando un sacco di spazio. Se ad esempio in due schermate abbiamo lo stesso sprite in cui però cambia un dettaglio (che ne so, un buco dovuto a un proiettile in uno sparatutto) è sufficiente memorizzare una sola volta tale

sprite, sostituendo semplicemente la tile "bucata" :)

C'e' da dire che il metodo delle tile incasina parecchio il lavoro del romhacker (e quando mai?), perchè spesso si fa fatica a ritrovare le immagini: se volete modificare con un tile editor il title screen di un gioco ad esempio, potreste trovare le sue tile tutte affiancate e quindi lavorare agevolmente, avendo l'intera immagine sotto gli occhi. Le tile però potrebbero anche stare sparpagliate nella ROM (valle a ritrovare!) o peggio ancora, alcune tile potrebbero essere utilizzate piu' volte nell'immagine, quindi ritoccandole provochereste modifiche anche ad altre zone dell'immagine.
Il primo caso è teoricamente risolvibile sfruttando le mappe usate dalla unità grafica per ricomporre le immagini (le tilemap menzionate all'inizio) in modo da rintracciare le tile e ricomporre l'immagine al volo, è comunque una tecnica per programmatori SNES esperti e per quanto ne so, non è mai stata implementata.

Qualche spiegazione tecnica, i bitplane

In che modo può essere utile sapere tutto ciò al romhacker?
La modifica grafica ha parecchi utilizzi, il più comune e forse anche il più utile è il cambio del font del gioco, infatti le lettere sono immagini a tutti gli effetti e quindi sono memorizzate nella ROM sotto forma di tile. Cambiare il font è un'operazione simpatica quando quello originale non è di vostro gradimento, mentre diventa indispensabile se si traduce un gioco dal giapponese, poichè bisogna inserire nella ROM le lettere del nostro alfabeto. E' possibile comunque modificare qualsiasi oggetto grafico, come il title screen per metterci la firma del traduttore ;) oppure sostituire lo sprite di Super Mario con quello di Sonic e altre cose simili ^_^

Per modificare le tile oggi esistono programmi appositi detti tile editor (uno dei migliori è probabilmente il Tile Layer Pro), che permettono di visualizzarle e modificarle direttamente, come se si stesse lavorando con un programma di grafica tradizionale, sono comunque convinto che un minimo di conoscenze tecniche sulle tile faccia sempre bene.

Bisogna innanzitutto sapere che una tile viene memorizzata come una stringa di byte (scontato:P), ognuno dei quali rappresenta una riga. Ad esempio:

```
0 1 1 1 1 1 0 0 -> 7C

1 1 0 0 0 0 1 0 -> C2

1 1 0 0 0 0 1 0 -> C2

1 1 0 0 0 0 1 0 -> C2

1 1 0 0 0 0 1 0 -> C2

1 1 0 0 0 0 1 0 -> C2

0 1 1 1 1 1 0 0 -> 7C

0 0 0 0 0 0 0 0 -> 00



0 = pixel vuoto

1 = pixel pieno
```

Immaginate di guardare questa figura come se fosse "disegnata", vedreste una O. Basta convertire ogni riga in un numero esadecimale ed si ottiene la stringa da memorizzare nella

ROM, in questo caso 7C C2 C2 C2 C2 C2 7C 00. Un rettangolo come questo è chiamato bitplane (piano di bit). Come vengono gestiti però i colori? Un metodo come questo permette solo di disegnare immagini monocromatiche (0=pixel vuoto, 1=pixel pieno) mentre sappiamo bene che i giochi usano i colori, e tanti anche! La risposta è sovrapporre i bitplane :)
Per spiegare meglio uso un esempio preso direttamente dal doc di **Peekin**.

```
3 3 3 3 3 3 3 3 ------- bitplane 3

3 2 2 2 2 2 2 2 2 ----- bitplane 2

3 2 1 1 1 1 1 1 1 --- bitplane 1

3 2 1 0 0 0 0 0 0 0 - bitplane 0

3 2 1 0 0 0 0 0 0 0

3 2 1 0 0 0 0 0 0 0

3 2 1 0 0 0 0 0 0 0

3 2 1 0 0 0 0 0 0 0

  2 1 0 0 0 0 0 0 0

    1 0 0 0 0 0 0 0

      0 0 0 0 0 0 0
```

Pensate a una tile come a una sequenza di bitplane sovrapposti uno sull'altro, per sapere il colore di un pixel bisogna leggere i bit sovrapposti su quel punto. Ad esempio, avendo una tile di 8x8 pixel formata da due bitplane, se voglio sapere il codice del colore per il pixel in alto a sinistra (quello che i programmi di grafica identificano con le coordinate 0,0), leggerò il bit in alto a sinistra del primo bitplane e poi il bit in alto a sinistra del secondo bitplane. I due bit ottenuti sono il colore di quel pixel.
Il modo in cui i byte dei vari bitplane sono disposti nella ROM varia a seconda del metodo grafico utilizzato, per i quali vi rimando alla già citata guida di Peekin sulla grafica del SNES, che a mio parere è la piu' completa a riguardo.
Torniamo invece ai colori, se siete riusciti a capire il metodo dei bitplane sovrapposti potreste essere giunti da soli a una importantissima considerazione: il numero dei colori di una tile dipende dal numero di bitplane. Con tre bitplane, il codice di ogni colore è formato da 3 bit e dunque si hanno a disposizione 2^3=8 colori, se fossero stati 4 avremmo avuto 16 colori e via via fino a un massimo di 8 bitplane, per un totale di 256 colori.

Le tile su SNES sono sempre di 8x8 pixel. Le tile di dimensioni diverse di cui parlano altri docs non sono altro che degli "assemblaggi" formati da più tile, ad esempio con 4 tile 8x8 si può ottenere una tile 16x16. Questo è il caso di Go! Go! Ackman, in cui le lettere del testo occupano una tile 16x16 ciascuna.

Primi passi, alcuni problemi

L'editing della grafica può dare parecchi grattacapi a chi ci si avvicina la prima volta. Alcune delle domande sentite piu' frequentemente sono "come mai i colori nel tile editor non corrispondono a quelli del gioco? Sono tutti sballati!" oppure "cosa è tutta quella roba che si vede tra una zona di grafica e un'altra?".

Andiamo con ordine. Innanzitutto bisogna precisare che il SNES cosi come tantissime altre console utilizza le palette per memorizzare i colori, chi smanetta con immagini in formato GIF o

PCX saprà bene di cosa si tratta. Il problema di fondo è che per rappresentare un colore RGB sono necessari un sacco di dati, per fare un esempio tangibile prendiamo il formato BMP a 24 bit per PC (le comuni immagini .bmp salvabili col paint di windows). Per ogni pixel vengono salvati tre byte, uno per ogni componente della tripletta RGB (insomma uno per la componente rossa, una per la verde e una per la blu). Con una rapida moltiplicazione si capisce perchè tali immagini hanno sempre dimensioni spaventose! Un'immagine di appena 50x50 pixel occupa 50*50*8=7500 byte!

Svariati anni fa venne introdotto il metodo della palette, che permetteva di risparmiare un sacco di spazio: perchè ripetere tre byte per ogni pixel che deve essere di un determinato colore?
La palette è una specie di tabella contenente un numero limitato ma arbitrario di colori (la palette piu' classica ne ospita 256), per ogni pixel dell'immagine viene usato UN SOLO byte per indicare il colore, questo byte altro non è che un indice da usare per selezionare un colore dalla palette. La tripletta RGB viene memorizzata una sola volta ed è "richiamata" per ogni pixel che si vuole fare di quel colore. Ingegnoso, no?
Questo sistema si castra da solo però ^_^ infatti la palette deve avere dimensioni limitate (come detto poco fa, solitamente non piu' di 256 colori) perchè altrimenti non e' piu' sufficiente un solo byte per fare da indice e tutta la tecnica si rivela inefficace. Questo spiega anche perchè provando a salvare una fotografia in formato GIF, l'immagine si impoverisce di colori: vengono limitati a 256, contro i 16 milioni del formato BMP ^_^
Per una console è comunque un limite più che accettabile, tanto che con soli 256 colori si riescono ad ottenere giochi dalla grafica splendida. (Per evitare malintesi, bisogna precisare una cosa: 256 è il numero di colori visualizzabili contemporaneamente a video, ma il SNES potenzialmente ne sa gestire un numero molto maggiore!)

Ora, tutto questo discorso è stato necessario per dire che, per disegnare un'immagine, c'è bisogno dell'immagine stessa (i byte indice che indicano i vari pixel) e della palette, per determinare la giusta colorazione di ogni pixel. Per quale motivo le immagini nei tile editor hanno i colori sballati? Il problema è che i nostri poveri tile editor non hanno la più pallida idea di dove si trovi la palette! Ogni gioco potrebbe memorizzarla in una zona differente e con un metodo differente. Tutto ciò che si può fare è "tentare" di leggere la ROM come se fosse codificata con un determinato metodo grafico (2bpl, 3bpl etc) affibbiandogli una palette inventata, e vedere se si riesce a individuare immagini familiari. Può sembrare di "tirare a indovinare", in effetti è proprio cosi :)

Questo causa anche un effetto collaterale: quelle zone confuse e ingarbugliate che compaiono tra una zona di grafica e un altra all'interno di un tile editor. Che roba è?
Il tile editor non ha la piu' pallida idea di dove si trovi la grafica del gioco, si limita a leggere L'INTERA ROM come se fosse un immenso file grafico. Cosi vengono interpretate come grafica anche cose come programma, sonoro, archivi di dati etc etc, tutta questa roba va a formare le misteriose zone "casinose" di grafica ;)

Un'altra cosa importante da sapere è che una ROM spesso utilizza più di un metodo grafico, per cui è necessario provare i vari metodi e scandagliare di volta in volta il file.
Non è certo il caso del Gameboy ^_^ dove la grafica è TUTTA da 2 bitplane (guarda caso il Tile Layer Pro ha chiamato la modalità a 2bpl "modalità gameboy"). Nel SNES il formato più usato è quello a 4bpl, con cui troverete la maggior parte degli sprite. I title screen possono arrivare anche a 8bpl, mentre i caratteri del testo sono memorizzati quasi sempre a 2bpl. Se ci pensate ha senso, 4 colori sono più che sufficienti per visualizare il testo e una eventuale ombreggiatura. Quando riconoscete immagini che però appaiono leggermente distorte o confuse provate a visualizzarle con un'altra modalità, potreste semplicemente aver sbagliato quella.

- L'hacking della grafica -

La modifica degli sprite ed in generale della grafica in una ROM è particolarmente utile a chi vuole intraprendere una traduzione dal giapponese, ed è per questo che mi soffermerò specialmente sulla sostituzione delle tile che compongono le righe di testo. Il resto della grafica, comunque, è modificabile circa nella stessa maniera.
Per sostituire le tile che costituiscono la grafica nella ROM, bisogna utilizzare un editor di tile. I più famosi possono essere scaricati dalla sezione **Utilità** del nostro sito. Effettivamente, il capitolo relativo alle tile è già stato completato e ulteriori informazioni possono essere trovate nei readme di ogni tile editor, comunque farò una piccola spiegazione con la ROM di "Zelda - Link's Awakening" per Gameboy (che è già in inglese, ma essendo una ROM per GB è particolarmente facile da modificare).

1) Con il NaGa sprite editor

Aprite la ROM scegliendola dal menu e premendo ESC. Cercate se riuscite a vedere in una certa zona della ROM delle figure che possono sembrare quelle presenti nel gioco. Se non ci riuscite, premete i tasti 1, 2 o 4 per cambiare metodo di visualizzazione (1Bit per pixel, tiles 8x8, ecc) ed utilizzate le altre opzioni possibili descritte nelle istruzioni di ogni utility (anche se il default del NaGa va bene per le ROM del Gameboy). Tra queste tiles, oltre a quelle che compongono i vari sprite del gioco, sicuramente vi saranno anche i font (lasciate perdere scritte intere come CONTINUE, SAVE, RAFT, SHOP, o altre che sono "disegnate" per intero e non sono sequenze di tile raffiguranti le lettere). Una volta visualizzate, cliccate su ogni tile per modificarla a piacimento. Questo editor scrive direttamente sulla ROM che state modificando.

2) Con il Tile Layer Pro

Aprite la ROM e mettete l'editor in modalità GameBoy cliccando su "View", poi "Format" e poi "Game Boy". A questo punto cercate la zona della ROM dove sono situate le tile. Prendete quelle che volete modificare, copiatele sulla clipboard a destra e modificatele. Una volta modificate, sarà necessario salvare le modifiche cliccando su "File" e poi su "Save", oppure su "Save as..." se volete salvarla senza modificare quella originale. Quella clipboard è molto comoda se si vuole modificare uno sprite le cui tile sono disposte alla rinfusa nella ROM.

[**NOTA:** Le tile contenute nelle ROM per Super Nintendo sono leggermente più difficili da modificare, perché possono essere codificate in formati differenti. Consiglio di provare i diversi metodi di visualizzazione se non sono visibili da subito. In alcune ROM compresse (come Seiken Densetsu 3) la grafica non è visibile in nessun modo, almeno non con i nostri sistemi. Chi è molto bravo con la programmazione e conosce l'assembly del SNES può scriversi un decompressore su misura, ma bisogna essere veramente veramente esperti :)
Un esempio di questo tipo di hacking lo trovate con il programma di FuSoYa, che abbiamo usato per tradurre alcune parole e parti grafiche di Zelda: A Link To The Past.]

Nelle traduzioni del Game boy c'e' un piccolo trucco per scovare facilmente la tabella dei font, il tutto con l'emulatore No$gb. Il trucco consiste nell'aprire il suddetto emulatore, la ROM che si vuole tradurre e giocare finché non compare del testo, a questo punto bisogna premere "Esc" e apparira il menu "Debugger". Una volta qui, clicca sull'opzione "Window" ed entra nel "Tile viewer". Qui potrai vedere gli sprite che stà usando l'emulatore nel momento che hai premuto Esc. Ora con il puntatore del mouse vai sulle tile che contengono le lettere di cui volete sapere il codice. Una volta sulla tile, guardate sul lato destro in basso, ci sara' scritto "Tile Nº --", dove -- corrisponde al codice della tile. In questa maniera si puo' creare una tabella "quasi" completa senza complicazioni.

Anche con il NES si puo' usare lo stesso trucco solo che va fatto in una maniera diversa. Come emulatore si usa il Nesticle, caricate la ROM e quando appare il testo premete ALT+P e apparirà "Cpu emulation paused". Qui premete F2 e vedrete una schermata piena di lettere,

grafica e altre cose. Una volta qui cercate la lettera A o quella di cui volete sapere il codice, cliccateci sopra e vedrete una finestra con lo sprite che avete scelto, poi, nell'angolo della finestra che e' apparsa, ci sara' una scritta come #?? dove ?? e' il codice hex corrispondente alla tile che avete scelto.

In realtà è possibile usare questo trucchetto anche per le ROM Megadrive, usando il Genecyst. Come al solito apritelo e caricare la ROM, poi premete ALT+P e anche qui apparirà "Cpu emulation paused". Una volta qui premete la barra spaziatrice e scegliete l'opzione "View", poi sull'opzione "Patterns" e qui potrete vedere gli sprite che sono in quel momento sullo schermo. L'unico problema è che non potrete vederne il codice :))))
Se qualcuno sa' come fare, scrivete :)

- La creazione della patch -

Per creare le patch da distribuire, si usa normalmente il formato IPS (International Patching Standard). Per creare le vostre patch IPS dovete procurarvi il programma SnesTool. Una volta avviato usate l'opzione "Create IPS", selezionate la ROM "genuina" (quella che non avete modificato) e poi la ROM modificata. Il programma costruirà il file IPS con tutte le differenze che avrà il nome dell'ultima ROM selezionata.
Adesso siete pronti a distribuire la vostra patch! Createvi un bello zip con:

- IPS (Versione che può solo usare i file ips e non crearli)
- Un bel README.TXT
- La vostra patch (Ma dai?? :))

Inoltre non guasterebbe un bel file batch che, prima crea una ROM di backup, e poi lancia ips con la vostra patch.
Lo ZSNES dalla versione 0.800 in poi supporta inoltre un sistema di patching automatico che patcha temporaneamente la ROM. Basta mettere il file IPS nella directory dei salvataggi (in alcuni readme errati delle nostre patch c'è scritto in quella delle ROM, ci scusiamo per l'errore), dopo averla eventualmente rinominata con lo stesso nome della ROM.
Scrivete anche questo nel readme se traducete una ROM per SNES.

- FAQ -

Q: Ho rinominato la ROM in formato TXT, perché non vedo il testo? (domanda realmente posta)
A: È necessario leggere il resto della guida per tradurre una ROM. Perciò se pensavi che le sole FAQ bastassero, lascia perdere :P

Q: Ora ho trovato la tabella dei font, ma cosa me ne faccio?!?
A: Ti scarichi un editor, consiglio il Thingy e ti leggi le istruzioni.
Non chiedetemi di spiegare come funziona ogni singolo editor di testo! Il readme è sempre molto completo.
(Per il Thingy, se dopo aver letto il readme avete ancora problemi, potete leggere la nostra Guida all'uso del Thingy.
Generalmente è necessario inserire in un file di testo la tabella che avete trovato, ma il sistema cambia da editor a editor.

Q: Perché non riesco a trovare certe parole col SearchR o le parole compaiono "troncate"?
A: Probabilmente la ROM utilizza un sistema di compressione come il DTE e gli altri elencati qui sopra. Se siete particolarmente sfigati è possibile che lo script sia compresso con qualche

algoritmo "tradizionale", come ad esempio Huffman (avete presente il winzip?). In tal caso o siete molto bravi con l'ASM o non c'è niente da fare.

Q: Il SearchR non trova alcuna parola! Perché?
A: Assumendo che tu abbia tenuto in considerazione tutti i sistemi di compressione precedentemente elencati nella guida, è possibile che il testo sia compresso in una maniera particolare (provate a comprimere un file di testo con il WinZip, per intenderci). Ci sono poi alcune ROM nelle quali col SearchR non si trova nulla per motivi apparentemente inesistenti (come Bart Simpson VS the World per NES).

Q: Perché il SearchR trova corrispondenze diverse cercando la stessa parola (nel senso che, cercando una parola e trovandone varie corrispondenze nella ROM, trova che la lettera A risulta associata a diversi numeri esadecimali)?
A: Il SearchR prova tutte le combinazioni secondo cui una successione di numeri può formare la parola cercata. È possibile che, per puro caso, nel codice della ROM ci fossero dei numeri che messi in un certo modo formassero la parola cercata (questo capita in particolar modo con le parole più piccole). Può anche darsi che la ROM abbia più di una tabella (Final Fantasy III ne ha una per l'inventario e una per il resto del gioco).

Q: Come mai cercando la stessa parola con le maiuscole e con le minuscole ottengo gli stessi risultati? Significa che quella ROM utilizza allo stesso modo maiuscole e minuscole?
A: No, semplicemente, come detto sopra, il SearchR trova le stesse corrispondenze se la sequenza delle lettere è la stessa (infatti, per esempio, il rapporto fra la lettera A e la lettera C, in numeri, è sempre C=A+2 sia che le lettere siano maiuscole, sia che siano minuscole... spero di essere stato chiaro in questo caso ^__^)

Q: Posso modificare le ROM in ASCII direttamente con un editor di testo come il WordPad?
A: No, la ROM si rovinerebbe. Una serie di caratteri vengono visti come specie di "quadratini", che vengono considerati tutti uguali, e di conseguenza vengono salvati tutti con lo stesso numero. Ovviamente dopo la ROM non funziona più.

Q: Voglio ottenere più spazio nella ROM! Come devo fare?
A: E' necessario fare un hacking dei puntatori in modo da dirottarli in una zona libera che aggiungete in fondo alla ROM. Il procedimento è spiegato dettagliatamente nel documento sull'hacking dei puntatori di Anus P.

Q: Ho tradotto una ROM del Gameboy, ma quando faccio partire l'emulatore dà un errore di "checksum". Che fare?
Le ROM del Gameboy hanno un particolare sistema secondo cui, qualunque cosa venga modificata nella ROM, bisogna cambiare dei byte anche che nell'header (la parte iniziale della ROM che la fa identificare dalla console). Pare che non tutti gli emulatori siano in grado di far funzionare ROM con il cheksum incorretto (cioe' una ROM tradotta), di conseguenza vi conviene usare un programma come l'HebeGB per modificare il checksum e farlo risultare corretto all'emulatore.

Q: Il SearchR non parte! Quando lo avvio scrive solo "Unable to open 640x480x8bpp VESA2L screen".
A: La tua scheda grafica non supporta i driver VESA in versione 2.0 o superiore. È possibile ottenerli al sito web del produttore oppure con lo SCITech Display Doctor.

Q: Ok, ora parte, ma se clicco su GFX EDIT scrive "Unable to open 320x240x8bpp VESA2L screen". Eppure ora la mia scheda supporta i drivers VESA.
A: Si, ma non alla risoluzione di 320x240. In questo caso, non è possibile ovviare al problema

in alcun modo, a meno che non esca una nuova versione dello SCITech DD che doti la tua scheda di questi drivers anche a risoluzioni più basse.

Q: Le patch di traduzione sono legali?
A: Fintanto che la patch viene distribuita senza la ROM, risulta legale. È invece illegale distribuire una ROM alla quale sia già applicata la patch. Questo perché per scaricare una ROM è necessario possedere il gioco su supporto originale (per le console fino a 16-bit, generalmente la cartuccia), e nessuno può avere su cartuccia una ROM tradotta.

Bene, se ci sono domande che riteniate vadano aggiunte a queste FAQ o se semplicemente avete ulteriori problemi, scriveteci specificando di aver letto la presente guida e dando informazioni precise di cosa non riuscite a fare.