GUIDA ALLA TRADUZIONE DEI VIDEOGIOCHI
(Le Avventure Grafiche, per lo più)

0. Premessa. In questa piccola, poco pretenziosa guida, cercherò di dare degli spunti di partenza a quanti vogliano apprendere gli strumenti e le nozioni basilari per provare (provare!) a "scardinare" i file che contengono i testi di videogiochi, soprattutto le avventure grafiche, i giochi per computer che più di altri hanno i dialoghi come base fondante della loro struttura, per offrirne una traduzione amatoriale.
Ci sono alcune necessarie nozioni che verranno date per scontate, mentre ad altre verrà dato un accenno nel corso della spiegazione. Le parti di codice sorgente che verranno presentate saranno in Delphi, chi conosce il Pascal, ovviamente, partirà avvantaggiato. Si danno per acquisite le nozioni base di informatica relative ai file e al sistema esadecimale, nonché al calcolo aritmetico in questo sistema. Non ci sono specifici strumenti applicativi particolarmente preferibili l'uno rispetto all'altro: sono necessari un hex editor (negli esempi viene usato UltraEdit®-32 della IDM Computer Solutions, Inc. http://www.ultraedit.com), un linguaggio di programmazione visuale (negli esempi viene usato Delphi, della Borland : http://www.borland.it. Esiste anche una distribuzione gratuita per scopi non commerciali), una calcolatrice scientifica (per i nostri scopi, quella di Windows, è più che sufficiente). Potrebbe rendersi necessario avere un programma per il confronto di file. UltraEdit®-32 contiene anche la funzionalità di confronto file.

1. "Perché devo dannarmi l'anima con puntatori?"

Nella maggior parte delle avventure grafiche, così come nella maggior parte dei videogiochi, nel codice sorgente, nelle parti dove si svolge un dialogo, le frasi non vengono referenziate esplicitamente, e vengono elencate invece in un file separato dall'eseguibile.
L'utilità di tale approccio è duplice: da una parte le eventuali modifiche possono venire apportate più velocemente e in maniera puntuale, senza dover perder tempo a cercare in tutto il codice le frasi da modificare. Quindi più o meno, la reazione di un personaggio di un'ag, di fronte al solito dilemma chiave/porta, potrebbe essere guidato dal seguente pseudocodice (pseudo!):

```
IF comando="USA <oggetto1> con porta"
THEN
   IF <oggetto1>="chiave d'ottone"
   THEN
      LEGGI frase1 IN buffer1
      VISUALIZZA buffer1
   ELSE
      LEGGI frase2 IN buffer2
      VISUALIZZA buffer2.
```

dove frase1 e frase2 non sono altro che indici che identificano due elementi del nostro "file" o tabella che contiene l'elenco di tutte le frasi e buffer1, buffer2 sono due stringhe di appoggio.
Sempre seguendo quest'esempio, potremmo ideare il seguente file:

```
1 | "Ehi! Funziona. La porta adesso è aperta."
2 | "Non funziona."
```

composto da due frasi, indicizzate da un intero nella prima colonna.

I vantaggi derivanti dall'adozione di un simile metodo sono evidenti, svincolando completamente il codice sorgente dalle modifiche ai testi, frequenti in fase di sviluppo. Inoltre, avere tutte le righe di dialogo concentrate in un'unica risorsa, aumenta notevolmente la facilità di modifica e di sviluppo dei dialoghi.
Questo modo di operare, comunque, non è esattamente, quello che viene adottato nella stragrande maggioranza dei casi, perché, non sapendo a priori qual è la lunghezza di una frase, volendo creare una tabella di tutte le frasi, saremmo costretti ad allocare un gran numero di byte per essere sicuri di "poterci fare entrare" anche la frase più lunga.
In termini tecnici, questo tipo di file si dice a "tracciato fisso".

Quello che invece comunemente viene fatto è di far corrispondere in una zona del file a parte, una specie di indice, di sommario: ogni elemento di questa specie di sommario, indica l'indirizzo del file, ossia il byte preciso, dal quale tale frase inizia; l'elemento successivo dell'indice individua l'inizio della frase successiva e quindi la fine della frase precedente.
In questo modo, il videogioco può, magari livello per livello, o nel caso delle ag, locazione per locazione, caricare in memoria, tutte le frasi leggendole con una routine che accede per indirizzo ad ogni frase.

Fatta questa premessa, ecco che arriviamo noi, cinque, dieci, quindici anni dopo, e decidiamo di intraprendere la traduzione di una qualche avventura grafica, rimasta sino ad allora in lingua originale (in genere l'inglese) per qualche inspiegabile motivo (di solito i soldi, o per meglio dire, la loro mancanza).

Di solito le frasi che cerchiamo sono "nascoste" dentro un file di risorse, vale a dire un file che può contenere anche altri tipi di dati, come suoni, immagini, musiche, se siamo fortunati dentro un file di testo, se siamo particolarmente sfortunati in un file criptato.

Mettiamoci nel caso di moderata sfortuna: i testi sono "in chiaro".
Bene, se proviamo ad aprire il nostro file e a tradurre direttamente con il text editor le frasi contenute, magari in modalità "inserimento" ci accorgeremo presto che o il gioco non partirà con il file modificato, restituendo un qualche messaggio d'errore, o se parte, avremo un comportamento "strano": frasi incomplete, troncate e così via.
Ecco che salta dunque agli occhi la necessità di dovere modificare oltre alle frasi, anche i relativi "puntatori", ossia gli indici alla posizione relativa di ciascuna frase all'interno del file: per ogni frase modificata in lunghezza, sia in aumento che in diminuzione, dovremo contraggiornare i puntatori relativi a tutte le frasi seguenti!!!

1.1. Subito un controesempio "semplicissimo".

Ecco da Rent-A-Hero (neo Software, 1998) un'eccezione alla regola appena formulata.
Quello seguente è una parte di uno dei file dei dialoghi, praticamente in chiaro, con una evidente chiave di accesso:
021_SPR_01="The village elder Soromann recognised the power and importance of these stones and called them 'Gloomstones'."

021_SPR_02="And their lives were about to change..."

023_SPR_01="The shiny gold created envy and greed amongst the other peoples. But after the war came peace. Some, however, loved the battle, so they were outlawed and banished onto a number of ships. After many years of peace and unity fate raised its ugly head again to remind us of those that had been forgotten..."

024_ST2_01="Is there nobody who can help us?"

Come si può subito intuire, nel codice sorgente il programmatore referenzierà il nome della stringa (024_ST2_01, 023_SPR_01, etc.) piuttosto che la stringa stessa: questa verrà (probabilmente) estratta dal file da una routine apposita, che leggerà la stringa e la porrà in un buffer di lettura. Potrebbe leggere solo il contenuto tra le virgolette " o semplicemente, e più univocamente, arrestarsi al doppio comando di "carriage return" che non è visualizzato nel testo sopra riportato, ma è presente con la coppia di valori x'0D0A'.

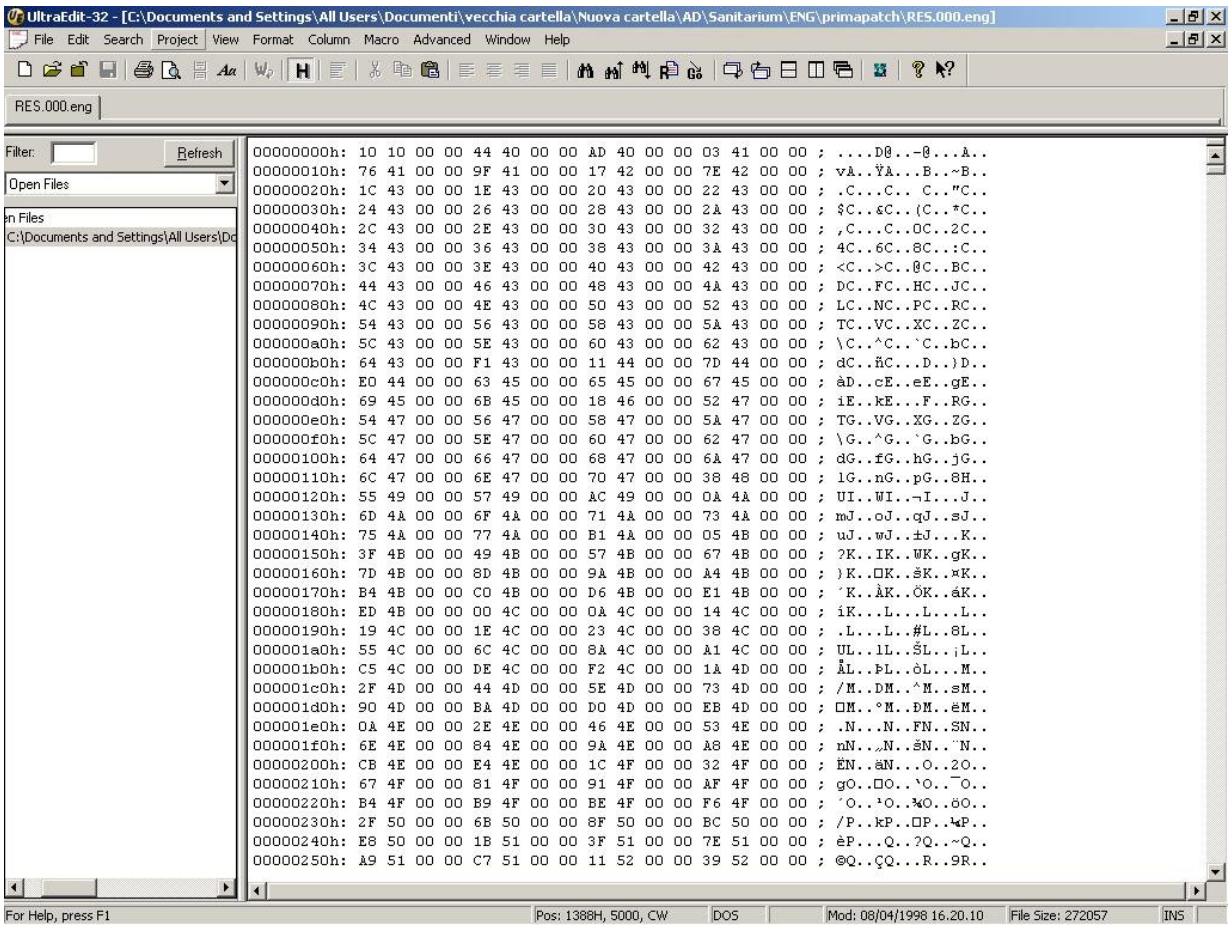
Qui, non dobbiamo proprio far niente: possiamo pure permetterci di editare il file con un word editor, premurandoci di salvarlo come file ASCII, senza contraggiornare nessun puntatore, e stando attenti a rispettare i valori di carriage return (ossia l'andata a capo).

Magari tutte le ag fossero fatte così …

2. Le cose iniziano a farsi meno banali.

Premessa doverosa: nei sistemi operativi MSDOS, Windows, etc. (ma potrebbe essere una proprietà dei processori INTEL, dovrei pensarci o indagare, e può darsi che lo farò, prima di rilasciare questo documento…) i numeri non vengono registrati dai programmi all'interno dei file da sinistra verso destra, bensì i byte che li compongono sono scritti da destra verso sinistra. Tanto per fare un esempio, il numero esadecimale x'0000AC4E' lo troviamo scritto come "4EAC0000".

Ciò detto, prendiamo subito un esempio in cui ci viene "sbattuto" praticamente in faccia questo famoso "indice" di tutte le frasi.
Questo è il file RES.000 dove sono contenuti tutti i testi di Sanitarium (ASC Games, 1998).



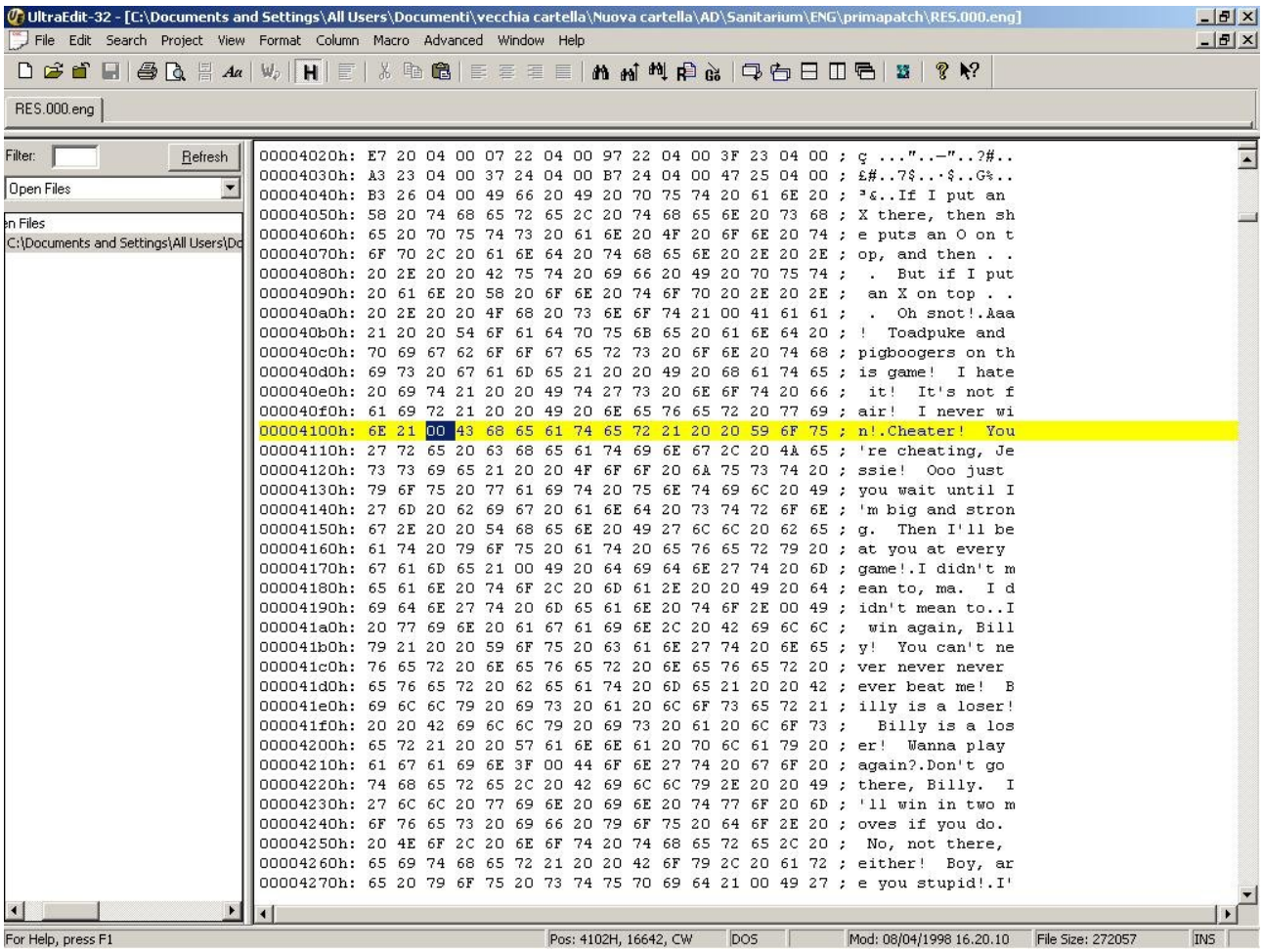
Non fatichiamo molto a capire che i primi 4 byte (x'00001010) scritti come abbiamo appena spiegato come "10100000") esprimono il numero (in esadecimale!) delle frasi, e quindi dei puntatori, contenuti nel file: lo iniziamo a sospettare osservando l'indirizzo esadecimale al quale terminano i puntatori: x'00004043" (l'indirizzamento di tutti i file inizia da x'00', non da x'01'!). Quindi, x'00004043'- 4 (cioè i byte iniziali) + 1 (dobbiamo contare il byte x'00'!) danno x'00004040') che, diviso per i 4 byte che compongono ogni puntatore, da' proprio x'00001010'.

D'altra parte, che i puntatori siano formati da 4 byte (a priori avrebbero potuto essere più corti) lo si vede "a occhio" osservando il file con l'hex editor per righe di 16 byte ciascuna. Un'ultima verifica a campione, ci convince che proprio di puntatori si tratti: prendiamo un indirizzo a caso dalla lista e posizioniamoci all'indirizzo indicato (Ultraedit offre una comoda funzionalità a questo proposito): andremo proprio all'inizio di una frase.

A questo punto, ci rimane poco da fare: il nostro programma (perché pensare di fare questo lavoro "a mano" sarebbe disu-mano…) dovrà effettuare poche operazioni fondamentali:

1) aprire il file in I/O;
2) consentire la scrittura della frase nuova;
3) sostituirla a quella vecchia;
4) contraggiornare tutti i puntatori seguenti quello relativo alla frase modificata, incrementandoli o decrementandoli a seconda che la nuova frase sia più o meno lunga della vecchia.

Per chiarezza, ecco l'immagine relativa alla parte delle prime frasi del file:



3. File come matrioske.

Quello di Sanitarium è un caso piuttosto raro. Le cose, molto spesso, non vanno così bene.
Quando troviamo delle frasi “in chiaro” in un file, da ciò che abbiamo visto finora, ci verrebbe la tentazione di dire: “Bene, ora osservo l’indirizzo di una qualsiasi frase, lo “rivolto” ed avvio la ricerca all’interno del file per quel valore, per trovare dove sono gli indici dei puntatori.”
Bene, non faticheremo molto a verificare che questo discorso spesso non funziona.
(CONTINUA…)