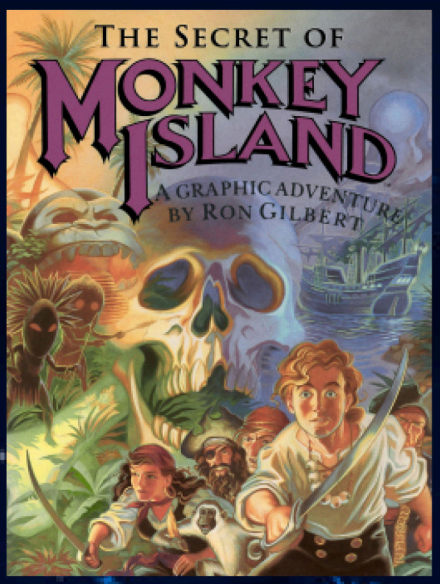
World RetroMagazine
future days are back
ISSN 3103-697X
Numero 26 Anno 4 - Novembre 2020 - WWW.RETROMAGAZINE.NET - Pubblicazione gratuita

Compatibilità software tra ZX 81 e Spectrum: mostri e fantasmi





DOOM

PC MS DOS...

Buon compleanno Monkey Island !

Il TMS9918A



CANNON FODDER

Amiga / PC

The Haunted House ...ovvero come giocare lo stesso gioco su 3 piattaforme diverse!

Un bit di rarità: Un dolcetto senza scherzetto!



MICRO MAGES (Nes)



Luigi's Mansion (Game Cube)

Una "terapia" per il Covid-19? Abbiamo la soluzione!

"Che paura mi fa...", cantavano i mostriciattoli nella sigla di "Carletto il principe dei mostri". Ve lo ricordate?

Sì, è vero, questo virus ci ha fatto e ci fa ancora paura. Ci paralizza in tanti modi diversi, perché si tratta anche di un virus "sociale" e "multimediale". Un virus che si è diffuso prima attraverso il terrore e poi arrivando effettivamente tra noi. Come possiamo curarlo? Al momento, non abbiamo una terapia definitiva, per quella ci vorrà l'aiuto del tempo, della scienza e della medicina. Ma possiamo "combattere" la parte psicologica del virus con i nostri amati videogame e i nostri gloriosi computer vintage.

Secondo l'OMS (l'Organizzazione Mondiale per la Sanità), il mondo dei videogiochi e tutto quello che gravita attorno ad esso hanno un efficace potere "terapeutico" per tutti i suoi utenti. Dedicarsi ai videogiochi consente di distrarsi e di impegnarsi in attività che coinvolgono e che portano interattività sociale (grazie, ad esempio, a piattaforme multiplayer come Kaillera). Giocare insieme e condividere le esperienze previene il contagio. Anzi, previene un particolare contagio, sottovalutato ma altrettanto pericoloso: quello della solitudine, della depressione e dell'ansia. Non si tratta solo di "giocare", ma di interagire con questo incredibile universo, grazie anche alle molte iniziative nate proprio durante questa folle e inattesa pandemia.
Nel mondo del retrocomputing, abbiamo assistito con interesse alle numerose sfide online sui vecchi classici, alle gare di programmazione in linguaggi come il BASIC (ad esempio, la competition basata sul tema di Alien Attack! che noi di RMW abbiamo promosso in prima persona) e all'incessante recupero di retro-hardware di tanti appassionati che riparano e rigenerano vere e proprie gemme del passato informatico. Tutto questo ci ha permesso di stare lontano da quei terribili mostri psicologici, ancor più spaventosi di quelli dei film horror o dei danni alla salute che il virus stesso può causare a noi e ai nostri cari. Sono queste paure i mostri che dobbiamo combattere, quegli stessi mostri che ci hanno accompagnato in questi duri mesi.

E pensare che proprio l'OMS, pochi anni fa, aveva dichiarato in modo allarmistico che la dipendenza da videogame fosse da considerare come una malattia, la cosiddetta "gaming disorder". Ed ecco che qualcosa un tempo etichettato come pericoloso, si trasforma oggi in un valente strumento in grado di aiutarci ad affrontare qualcosa di ben più rischioso. Riteniamo molto positivo che si riscoprano le qualità profonde insite nel videogioco, inteso come mezzo di comunicazione pensato per unire anziché isolare. Soprattutto in un periodo così difficile a cui siamo tutti indistintamente sottoposti.

Forza, allora! E' arrivato il momento di sconfiggere anche questo "mostro". Noi sappiamo come si fa! Cominciamo col premere il pulsante di avvio della nostra console o del nostro home computer preferito. Vedrete che lentamente ci potremo trasformare in un idraulico coi baffi, un pilota di astronave, un cavaliere in armatura, un guerriero... Potremo essere tutti questi personaggi altri ancora. E sapremo anche come sconfiggere il mostro di fine livello. In fondo, pensateci bene, non lo abbiamo già fatto centinaie di volte? ;-)

Carlo Nith Del Mar Pirazzini

SOMMARIO

◊ Il video chip TMS9918A — Pag. 3
◊ Compatibilità software tra ZX 81 e Spectrum: mostri e fantasmi — Pag. 6
◊ Alien Attack! dall'Amstrad CPC al Commodore 64 tramite il Simons' Basic — Pag. 9
◊ Bombs Away! — Pag. 15
◊ Un bit di rarità: un dolcetto senza scherzetto — Pag. 22
◊ The Haunted House... ovvero come giocare lo stesso gioco su 3 piattaforme diverse! — Pag. 25
◊ Come disattivare i tasti corrispondenti agli switch del joystick (C64) — Pag. 40
◊ Buon compleanno Monkey Island! — Pag. 42
◊ Billy Masters Was Right (PC) — Pag. 44
◊ Project Firestart (C64) — Pag. 45
◊ Luigi's Mansion (GameCube) — Pag. 47
◊ Cannon Fodder (Amiga/PC) — Pag. 49
◊ Micro Mages (NES) — Pag. 51
◊ Doom (PC) — Pag. 53
◊ Nintendo VS Takeshi (FamiCom) — Pag. 55
◊ Stormlord (C64) — Pag. 57
◊ True Lies (SNES) — Pag. 58

Hanno collaborato alla stesura di questo numero di RetroMagazine World (in ordine sparso):

- Alberto (Ghostbusters) Apostolo
- Gianluca Girelli
- Michele conte Ugolino
- Carlo N. Del Mar Pirazzini
- Daniele Brahimi
- Flavio Soldani
- Francesco Fiorentini
- Attilio Capuozzo/RPI Italia
- Querino Ialongo
- Leonardo Miliani
- Edoardo Ullo
- Supporto grafico Irene G. Valeri
- Copertina a cura di Flavio Soldani

Il video chip TMS9918A

di Leonardo Miliani

Inizio da questo numero una serie di articoli dedicati ai componenti che hanno costituito l'anima portante dei nostri amati retro-sistemi, siano stati essi computer o console. Ciò che rendeva distintivo un certo sistema era la sua architettura hardware: a differenza di oggi, dove i computer condividono la medesima architettura, diretta evoluzione di quella dell'omonimo computer messo in commercio da IBM nel lontano 1981, nei lontani anni '70 e '80 del secolo scorso ogni produttore metteva in commercio un computer elaborandolo secondo scelte personali. Partendo dalla scelta della CPU fino ai vari chip accessori, ogni componente andava ad assemblare un sistema che, bene o male, si distingueva dagli altri ed era, per così dire, unico. Nonostante ciò, molti computer si assomigliavano perché, come detto nella precedente serie di articoli dedicati al mio computer auto-costruito LM80C, non tutti i costruttori erano anche produttori di integrati per cui chi non lo era era gioco forza obbligato ad attingere a ciò che offriva il mercato. E dopo la CPU, uno dei componenti sicuramente più importanti era il processore video, a cui era demandato il compito di generare l'immagine video da inviare al monitor o alla TV collegati al sistema, e che spesso aveva anche il compito di gestire non solo il testo ma anche la parte grafica. Il chip che andremo ad analizzare è il TMS9918A di Texas Instruments, usato in un svariato numero di computer e console dell'epoca.

TMS9918 Video Display Processor

Nella seconda metà degli anni '70 del XX secolo comparvero sul mercato dei computer ad 8 bit non più destinati prettamente agli amatori perché da assemblare a cura dell'utente (come l'Apple I, l'Altair 8800 e simili) ma finalmente sistemi user-friendly, alloggiati in un case dotato di tastiera e già pronti all'uso: l'utente doveva infatti collegare solo l'alimentazione e un monitor o, più diffusamente, un comune TV di casa. Il Commodore PET, l'Apple II e il TRS-80 invasero le case di milioni di utenti. Texas Instruments (TI), visto il notevole successo riscontrato dai suddetti computer, decise di entrare in quel settore di mercato ed avviò, nel 1977, lo sviluppo di un proprio home computer domestico denominato TI-99/4: questo era simile ad una console, riprendendo la capacità di lanciare software su cartucce, a cui era stata poi aggiunta una tastiera ed una porta di espansione per collegare delle periferiche aggiuntive. Quel computer domestico aveva bisogno di un processore video. Per l'occasione TI sviluppò il **TMS9918** (notare l'assenza della "A" nel nome) detto **Video Display Processor** (VDP), un chip dalle buone caratteristiche tecniche: il TMS9918 generava un segnale video a 60 Hz in standard NTSC con una risoluzione grafica di 256 righe per 192 colonne e 15 colori. Poteva gestire anche una modalità testuale da 40x24 caratteri ed una multicolore da 64x48 pixel. I progettisti del chip decisero di dotarlo di una VRAM (la memoria video) separata da quella di sistema di 4 KB: a causa di questa scelta, la CPU poteva scrivere nel buffer video solo attraverso il VDP. A favore di questa architettura giocava però il fatto che il VDP non andava a sottrarre memoria dal sistema per gestire il buffer video.



Figura 1 - il TMS9918 visualizza solo 4 sprite per linea di scansione

Sprite

Il TMS9918 supporta in hardware gli sprite, elementi grafici usati nei giochi per gestire elementi quali il personaggio del giocatore, i nemici o altri oggetti movibili. Curiosamente, il termine "sprite" fu coniato da un manager TI, Dave Ackley, e comparve ufficialmente per la prima volta proprio nella documentazione del VDP. Gli sprite sono oggetti grafici gestiti dal chip video che si muovono in modo indipendente dalla grafica dello sfondo, senza alterarla. Inizialmente era stato deciso di supportare solo 4 sprite monocromatici per motivi tecnici (limitato tempo di accesso da parte del VDP alla VRAM e poca larghezza di banda di quest'ultima per passare più informazioni) ma Matthew Hagerty e Pete Macourek, un paio dei progettisti del VDP, riuscirono ad espanderne il numero, portandolo a 32, adottando uno stratagemma basato su uno "stack sprite" ed uno "sprite pre-processing": durante il passaggio del pennello video nei bordi a lato dello schermo (una zona dello schermo senza informazioni da visualizzare) il chip recupera i dati degli sprite (pre-fetching) che impila nello stack. Durante la generazione dell'immagine il VDP controlla nello stack quali sono i 4 sprite che possono essere visualizzati per ogni singola riga e ne recupera i dati. Resta, però, il limite dei 4 sprite per riga: in fig. 1 è possibile vedere come le parti degli sprite dal 5° in poi che occupano la stessa linea di scansione dei primi 4 non vengano renderizzate a video.
Gli sprite hanno dimensioni di 8x8 oppure 16x16 pixel (in realtà composti da 4 sprite affiancati), che possono essere raddoppiate: in questo caso i pixel assumono una dimensione di 2x2 pixel, per cui le dimensioni degli sprite passano, rispettivamente, a 16x16 e 32x32 pixel.

TMS9918A

Il TI-99/4 non riscosse molto successo. TI migliorò il progetto e rilasciò nel 1981 il TI-99/4A, una versione del

computer riveduta in molti aspetti e dotato anche di un nuovo chip grafico denominato **TMS9918A**. La "A" indicava che, rispetto al suo predecessore, esso ha una nuova modalità detta "Graphic 2" che può gestire l'immagine in modalità bitmap, ossia ogni singolo pixel è gestibile singolarmente. Il TMS9918, infatti, può gestire solo grafica in modalità tile, ossia a tessere (o tasselli), dove un tileset composto da 256 tessere di 8x8 pixel è usato per gestire la grafica. Il limite di questa modalità è che gli elementi, ripetendosi sullo schermo, non permettono ad esempio di tracciare grafici o immagini complesse. Per poter supportare questa nuova modalità il TMS9918A deve essere dotato di 16 KB di VRAM, necessari a mantenere in memoria non solo tutte le informazioni relative al buffer video ma anche tutti i dati relativi agli sprite.

Versioni e utilizzo

La prima versione del chip, il TMS9918 (senza la "A") è stato usato soltanto per il TI-99/4. Il TMS9918A (con la "A") è stato usato inizialmente per il TI-99/4A ma poi dato in licenza anche a terzi: grazie a questa scelta commerciale è stato ampiamente utilizzato in numerosi sistemi dell'epoca. È stato montato sulle console ColecoVision, CreatiVision e Sega SG-1000, sui computer Spectravideo e, soprattutto, sugli MSX prima serie. Alcune console più moderne, come il Sega Master System, il Sega Game Gear ed il Mega Drive montavano dei chip grafici derivati dal TMS9918A. Per adattare il VDP a diversi sistemi ed a diversi segnali video ne sono state rilasciate versioni specifiche: il TMS9928A genera sempre un segnale NTSC ma in versione YPbPr (a componenti di colore) mentre per i Paesi con standard PAL è stato realizzato il TMS9929A. In seguito il progetto è stato rivisto e sono stati rilasciati i nuovi TMS9118, TMS9128 e TMS9129: questi chip sono identici ai modelli che sostituiscono, a parte la funzione di un pin ed una diversa gestione della VRAM.

N.B: da qui in poi con "TMS9918A" per comodità si intenderanno globalmente sia il TMS9918A che tutti i suoi derivati, dato che essi condividono le caratteristiche tecniche.

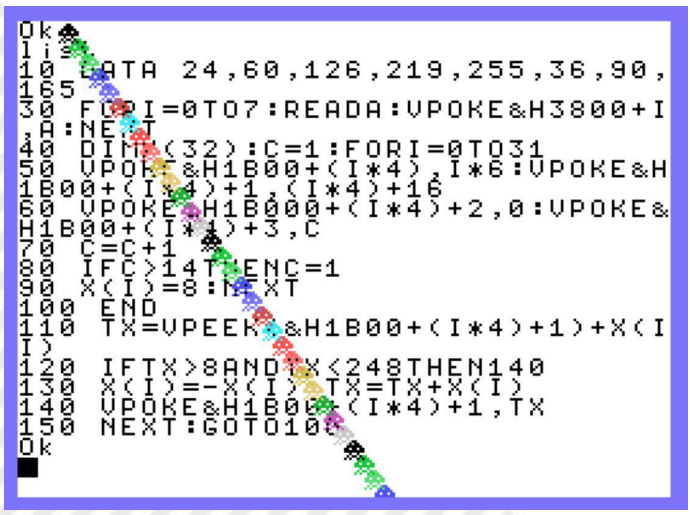


Figura 2: sprite renderizzati sopra al testo, che occupa uno degli ultimi piani dell'immagine

Caratteristiche tecniche

Il TMS9918A comunica con la CPU tramite un bus dati ad 8 bit e soltanto altri 3 segnali collegati ad altrettanti pin del chip: lettura, scrittura e modalità. I primi due sono chiari mentre l'ultimo è usato per indicare al VDP se i dati sono da scambiare con la VRAM oppure con uno dei suoi registri interni. Questi ultimi sono 8 in sola scrittura, con i quali si programmano le funzionalità del processore grafico, come ad esempio la modalità grafica o in quali porzioni della VRAM andare a memorizzare i dati del buffer video e degli sprite, ed 1 in sola lettura, il registro di stato, che fornisce informazioni ad esempio sulla collisione tra sprite o sull'interrupt. L'immagine video è suddivisa in "display planes". Questi "piani" sono in numero di 35, numerati da 0 a 34: più il numero è basso e più il piano è "vicino" all'osservatore ed ha priorità maggiore rispetto ai piani sottostanti. I primi 32 piani sono occupati dagli sprite, il 33° piano contiene l'immagine del fondo, il 34° è il "backplane", che è largo quanto l'intera immagine a video e forma lo sfondo di tutta l'immagine e, generalmente, si vede come bordo ai lati della schermata centrale. Dietro ad esso c'è un piano particolare detto "external VDP" che permetterebbe di introdurre nell'immagine il segnale video generato da una sorgente esterna: il VDP ha infatti un ingresso video (solo il modello TMS9918A) che, nei piani dei progettisti TI, sarebbe servito ad esempio a mettere in sovra-impressione degli oggetti e del testo sull'immagine generata da un altro VDP ma, a mia memoria, non sono a conoscenza di sistemi che abbiano usato questa funzionalità. L'ultimo piano, detto "default backplane", è una mera immagine nera che compare ad esempio quando il VDP viene alimentato ma non viene programmato dalla CPU. In figura 2 un esempio di sprite che si sovrappongono fra di sé e al testo, che occupa uno degli ultimi piani dell'immagine.

I colori disponibili sono 15, anche se alle volte vengono indicati come 16: in realtà, il 16° colore è "trasparente" e serve a far vedere l'immagine del piano dedicato al segnale esterno. Nel caso che non esista nessun pixel acceso in nessuno dei piani sottostanti e, in assenza di un segnale in ingresso, allora passerà il pixel dell'ultimo piano ("default backplane"): ecco perché, spesso, si pensa che questo colore sia anch'esso nero, perché passa il colore dell'ultimo piano che, appunto, è nero.

Relativamente alla gestione della VRAM, il VDP suddivide la memoria video in varie "tabelle" dove, a seconda della modalità grafica selezionata, vengono memorizzano i dati necessari alla generazione dell'immagine video e degli sprite. La "pattern table" contiene i motivi dei caratteri, la "name table" contiene i codici dei pattern da caricare per ogni cella dello schermo, la "color table" le informazioni sui colori. Anche i dati degli sprite sono memorizzati in 2 tabelle distinte: la "sprite attribute table", che contiene i dati principali degli sprite (come coordinate X e Y e colore) e la "sprite pattern table", che contiene invece i dati dei pixel che devono essere accesi o spenti.

Una grossa carenza del chip è l'assenza di scrolling, sia verticale che orizzontale. Se nel periodo in cui fu progettato,

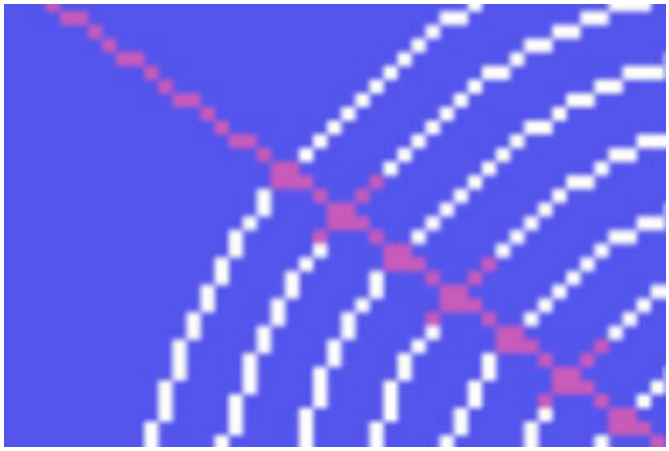


Figura 3: il difetto noto come "color spill"

la fine degli anni ‘70, i giochi erano a quadri singoli e tale caratteristica non era richiesta, agli inizi degli anni ‘80 comparve, soprattutto fra gli arcade, anche un nuovo genere di giochi detti a scorrimento: le conversioni dei giochi da bar risentono molto di questa limitazione, a cui devono sopperire via software.

Modalità grafiche

Il TMS9918A, come detto, supporta 4 modalità grafiche: testuale, grafica tileset, grafica bitmap, multicolore.

Modalità testuale:

in questa modalità lo schermo è diviso in 40x24 celle di 6x8 pixel, per una risoluzione dell’immagine di 240x192 pixel. Per gestire questa modalità il VDP utilizza una “pattern table” di 2.048 byte (256 pattern per 8 byte) e la “name table”, grande 40x24=960 byte, dove ed ogni byte rappresenta una cella di memoria: in ognuna di queste celle viene memorizzato il codice del carattere (da 0 a 255, per 256 possibili codici) che il VDP andrà poi a prelevare dalla pattern table per la visualizzazione. I colori disponibili sono solo 2 e valgono per tutto lo schermo: il colore principale, usato per i punti accesi dei singoli pattern, ed il colore di sfondo, usato per i punti spenti e per i bordi dello schermo. Siccome la cella video è larga solo 6 pixel, i 2 bit più significativi dei pattern vengono ignorati.

Modalità grafica 1:

questa modalità è una modalità grafica che può essere usata anche per rappresentare del testo. L’immagine è scomposta in 32x24 celle di 8x8 pixel, per una risoluzione di 256x192 pixel. La pattern table è grande 2.048 byte (256x8), la name table è 768 byte perché le celle ammesse sono, come detto, 32x24. A differenza di quella testuale, in questa modalità il colore è gestito in maniera diversa, e la color table è grande 32 byte: ogni byte contiene il colore dei pixel accesi e di quelli spenti per un blocco di 8 caratteri. Il colore del bordo è indipendente. Siccome i pattern disponibili sono solo 256 mentre le celle sono 768, questa modalità non permette di indirizzare i singoli pixel dell’immagine: per contro, questa modalità viene usata anche come testuale, perché si possono definire i pattern con i codici ASCII e poi visualizzarli sullo schermo con un semplice byte. Gli sprite sono supportati.

Modalità grafica 2:

La modalità grafica 2 è una modalità bitmap pura, dove il chip grafico può gestire ogni singolo pixel in maniera indipendente e gestire un’immagine di 256x192 pixel. A differenza della modalità grafica 1, in questa modalità l’immagine è gestita con 3 tabelle da 256 celle di 8x8 pixel l’una, per un totale di 768 pattern differenti. Anche il colore è gestito in maniera differente: ogni byte (8 pixel) dell’immagine permette di usare un colore primario ed uno per lo sfondo differenti. In questa modalità la pattern table occupa 6.144 byte: siccome ogni pattern occupa 8x8 pixel, servono 8 byte, e siccome ogni tabella contiene 256 pattern, servono 8x256=2.048 byte per ogni tabella. Stessa dimensione (6.144 byte) occupa la color table dato che, come detto prima, ogni byte della tabella indica i colori per un segmento video di 8 pixel. Infine, la name table è grande 768 byte (32x24 celle): lo schermo è diviso in 3 aree orizzontali (superiore, centrale e inferiore), ed ogni area è composta da 256 celle che possono contenere 256 pattern differenti. Anche la modalità grafica 2 supporta gli sprite. Siccome i colori dei pixel di ogni byte dell’immagine (8 punti orizzontali) sono memorizzati in un singolo byte, dove sono salvati il colore dei pixel accesi e quello dei pixel spenti, si capisce che per ogni cella di 8 pixel si possono avere solo 2 colori contemporaneamente: se l’utente prova a disegnare qualcosa con un terzo colore, quest’ultimo altera il colore primario di tutti i pixel di quel byte, come si vede dall’immagine in figura 3. Questo difetto è noto con il nome di “color spill”.

Modalità grafica multicolore:

Questa modalità permette di avere un’immagine a media risoluzione di 64x48 blocchi, dove ogni blocco occupa in realtà 4x4 pixel sullo schermo e può assumere uno qualunque dei 15 colori disponibili. Per questa modalità sono usate la name table e la pattern table: non viene usata la color table perché l’informazione sul colore è contenuta nella pattern table. La name table è composta da 768 byte che puntano ad un segmento di 8 byte nella pattern table (1.536 byte). Di questo segmento ogni area di 4x4 blocchi indica i colori per diverse righe dello schermo e sono usati solo 2 byte: il primo indica i colori dei 2 blocchi superiori ed il secondo quello dei blocchi inferiori. Anche questa modalità supporta gli sprite.

Modalità non documentate:

“giocando” con le impostazioni dei registri si possono ottenere alcune modalità non documentate del VDP. Ad esempio, è possibile ottenere una modalità intermedia fra la grafica 1 e la grafica 2, dove si hanno solo 256 pattern come in grafica 1 ma con la possibilità di gestire i colori dei singoli byte dei pattern come in grafica 2. Oppure impostare un modo diverso per ognuna delle 3 aree dei pattern sullo schermo, avendo ad esempio le prime 2 aree gestite in modalità bitmap e l’ultima in modalità tile, permettendo quindi di disegnare grafiche dettagliate per il quadro di gioco sui primi due terzi dello schermo e testi o motivi sull’ultimo, ad esempio per punteggi, messaggi e altro.

Bene, per quest’articolo è tutto. Alla prossima puntata.

Compatibilità software tra ZX 81 e Spectrum: mostri e fantasmi

di Alberto (Ghostbuster) Apostolo

Da ragazzo, mi capitava di analizzare ingegnosi piccoli programmi scritti per ZX 81 (quasi sempre giochi) ma spesso incontravo alcune difficoltà nella conversione delle istruzioni. Infatti non esiste una compatibilità al 100% tra ZX 81 e ZX Spectrum.

Differenze Hardware

I circuiti sono ovviamente diversi. L'espansione di memoria da 16 KB per ZX 81 non è utilizzabile su ZX Spectrum. I programmi salvati su cassette per ZX 81 non possono essere letti su ZX Spectrum. Solo la stampante ZX Printer si può collegare a entrambi i modelli [Bon83].

L'organizzazione della memoria è diversa e di conseguenza anche i comandi POKE e PEEK saranno diversi (così come le routine in linguaggio macchina e le variabili di sistema).

L'area di memoria riservata su ZX 81 per lo schermo è costituita da 768 bytes (24 righe per 32 colonne di caratteri) e grafica in bassa risoluzione (64 x 44 pixel).

Il modello ZX Spectrum ha 6912 bytes suddivisi in:

1) 6144 bytes di grafica (256 x 192 pixel oppure 24 x 32 caratteri),

2) 768 bytes (24 righe per 32 colonne) riservati ai colori dei caratteri e altri attributi video (luminosità e lampeggio).

Il set di caratteri per ZX 81 ha una sua codifica mentre i caratteri da 0 a 127 del set per ZX Spectrum sono ASCII. Per esempio, nel modello ZX 81 il carattere "A" e il carattere "A" in reverse (Fig. 2) hanno rispettivamente codice 38 e 166 [Bon82] mentre nel modello ZX Spectrum hanno lo stesso codice 65 (cambiano solo gli attributi video). Alcuni caratteri grafici non presenti nel set per ZX Spectrum si possono implementare tramite gli UDG (User Defined Graphics).

BASIC distinct...

Le differenze hardware hanno ripercussioni nelle rispettive versioni di BASIC cablate in ROM.

Il BASIC per ZX Spectrum si può considerare una estensione e una revisione di quello per ZX 81.

I comandi FAST e SLOW mancano su ZX Spectrum perché opera alla velocità FAST e trasmette i dati video in SLOW senza fare interferire le operazioni tra loro e non ha senso emularli (la gestione del video risulta indipendente).

Il comando PAUSE 0 emula PAUSE 40000 per ZX 81





Figura 1 (sopra) - Figura 2 (sotto)

```
10 POKE 23692,255
20 PRINT AT 21,31;''
```

Figura 3: l'apice si ottiene con Symbol_Shift+7.

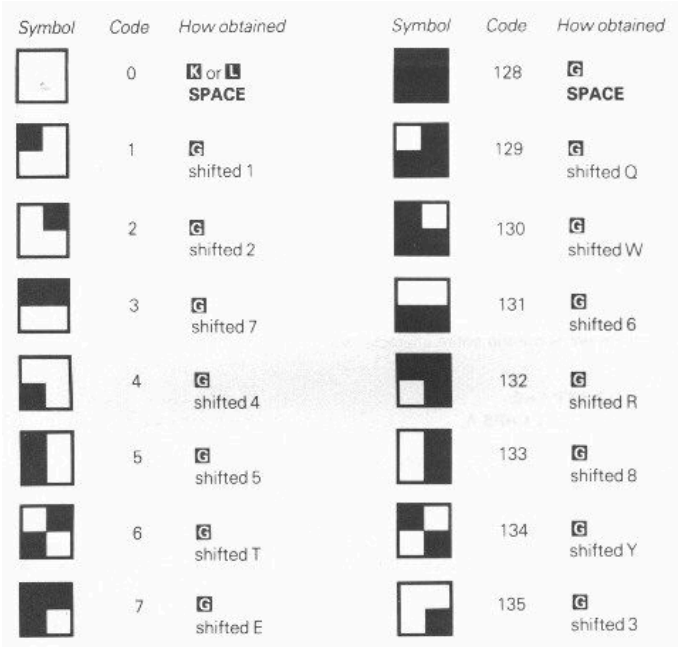


Figura 4

[Lis84b].

Manca il comando SCROLL perché su ZX Spectrum lo "scrolling" è automatico ma con controllo da parte dell'utente con la richiesta "scroll ?" che appare nella parte bassa dello schermo. Il comando POKE 23692,255 (variabile di sistema "SCR CT") seguito da PRINT AT 21,31;'' consente di emulare il comando SCROLL per ZX 81 (Fig.3).

Il comando UNPLOT per ZX 81 è stato sostituito dal

comando PLOT OVER 1 ma occorre fare attenzione alla risoluzione grafica e ai programmi per ZX 81 nei quali PLOT e UNPLOT modificano i caratteri grafici (Fig.4) in sostituzione dei comandi PRINT AT.

Il comando SAVE per ZX 81 esegue il salvataggio su cassetta dei programmi e i dati nello stesso file mentre su ZX Spectrum sono previste più opzioni per salvare array e sequenze di bytes (schermate, routine in linguaggio macchina). Altra differenza è il salvataggio dei programmi BASIC con autostart. Su ZX Spectrum si ha il comando SAVE s LINE k dove s è una stringa e k rappresenta il numero di riga al quale saltare dopo il caricamento con il comando LOAD. Invece su ZX 81 è un po' più complicato (Fig. 5). L'esempio è preso dal manuale originale per ZX81 (capitolo 16). Dopo avere avviato il registratore in modalità REC, con il comando RUN 100 sarà salvato il programma (come effetto collaterale benigno, l'ultima lettera del nome apparirà in reverse). Dopo avere riposizionato il nastro della cassetta, per caricare il programma occorrerà dare il comando LOAD "USELESS". Terminato il caricamento, l'elaborazione proseguirà alla linea 110.

in alternativa a RUN si può dare il comando GOTO se non si vogliono cancellare le variabili (mentre GOSUB è sconsigliato perché non funziona correttamente).

```
  5 REM "USELESS"
 10 PRINT "THIS IS ALL IT DOES"
 20 STOP
100 SAVE "USELESS"
110 GOTO 10
```

Figura 5

```
 5  LET S=0
10  LET I=15
15  PRINT AT 0,I;
20  IF PEEK (1+PEEK 16398+256*PEEK 16399)=60
                              THEN GOTO 90
25  PRINT "■■■"
30  IF INKEY$<>"X" THEN GOTO 55
35  FOR J=-8 TO 8
40  PRINT AT 8-ABS J,I+1;
          ("T" AND J<0)+("□" AND J>=0)
45  NEXT J
50  LET S=S-10
55  IF INKEY$="N" AND I>0 THEN LET I=I-1
60  IF INKEY$="M" AND I<29 THEN LET I=I+1
65  PRINT AT 14,RND*31;"W"
70  SCROLL
75  SCROLL
80  LET S=S+1
85  GOTO 15
90  PRINT S
```

Figura 6

```
  5>LET S=0
 10 LET I=15
 15 PRINT AT 0,I;
 20 IF SCREEN$ (0,1+I)="W" THEN
GO TO 90
 25 PRINT "▀▀▀"
 30 IF INKEY$<>"x" THEN GO TO 5
5
 35 FOR J=-8 TO 8
 40 PRINT AT 8-ABS J,I+1;("T" A
ND J<0)+(" " AND J>=0)
 45 NEXT J
 50 LET S=S-10
 55 IF INKEY$="n" AND I>0 THEN
LET I=I-1
 60 IF INKEY$="m" AND I<29 THEN
LET I=I+1
 62 PAUSE 5
 65 PRINT AT 14,RND*31;"W"
 70 POKE 23692,255: PRINT AT 21
,31;""
 75 POKE 23692,255: PRINT AT 21
,31;""
 80 LET S=S+1
 85 GO TO 15
 90 PRINT S
```

Figura 7

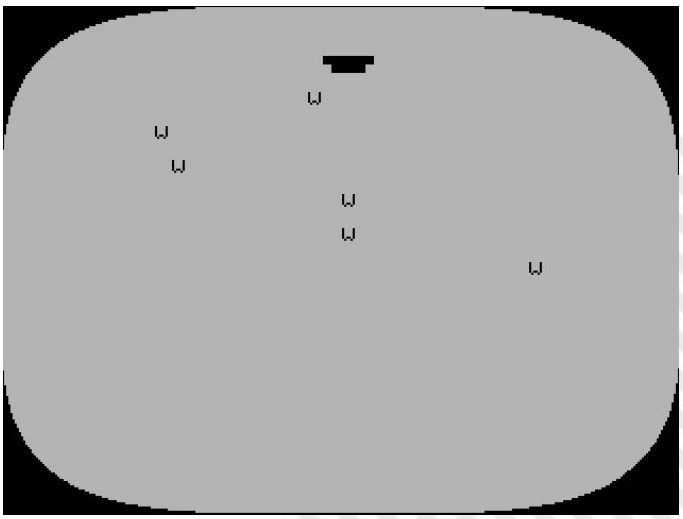


Figura 8: schermata del gioco di Fig.7

Gestire le collisioni nei giochi scritti in ZX BASIC

Generalmente, in un gioco che fa uso di caratteri alfanumerici come elementi grafici, le collisioni sono gestite con istruzioni BASIC e non con strane routine in linguaggio macchina.

Una difficoltà che si può incontrare nella conversione di un programma scritto per ZX 81, è dovuta alla tecnica di usare POKE e PEEK per il posizionamento di caratteri nell'area di memoria riservata allo schermo.

In Fig. 6 è dato un piccolo gioco per ZX 81, apparso sulla rivista italiana LIST n.1 [Lis84]. L'astronave nella parte alta dello schermo deve schivare i nemici (simboleggiati dalla lettera "W"). I comandi sono "N" per andare a sinistra, "M" per andare a destra e "X" per sparare (ogni sparo sottrae 10 punti alla variabile S che memorizza il punteggio). In Fig.7 è presente la versione per ZX Spectrum (con le modifiche alle linee 20, 70, 75 e l'istruzione PAUSE aggiunta per migliorare la giocabilità). Per gestire la collisione tra la parte centrale dell'astronave e i nemici, alla linea 20 è stata usata la funzione SCREEN$(row,col) che restitutisce il carattere stampato alle coordinate (row,col) dello schermo.

Tuttavia la funzione SCREEN$ ha un punto debole. Per esempio, se un pixel in alta risoluzione (disegnato con PLOT) "sporca" il carattere stampato alla posizione (row,col) allora SCREEN$ restituirà una stringa vuota perché non in grado di riconoscerlo.

E' un vero peccato che SCREEN$ funzioni solo con i

Una curiosità

In [Bon83] a pag. 230, c'è un errore. Gli indirizzi alla riga 20 sono per ZX 81 e non per Spectrum!

caratteri ASCII e non possa riconoscere gli UDG e gli altri caratteri grafici. Il mitico Dilwyn Jones suggeriva di aggirare l'ostacolo colorando gli elementi grafici e usando la funzione ATTR(row,col) che restituisce un numero associato alla posizione (row,col) pari a Flashing*128 + Brightness*64 + Paper*8 + Ink. Inoltre l'elaborazione della funzione ATTR è il 25% più veloce della funzione SCREEN$. In Fig.9 è riportato un piccolo programma scritto da Dilwyn Jones per ZX Spectrum nel quale sono usati gli UDG [Jon83]. I tasti per comandare l'astronave sono: "5" per spostarsi a sinistra e "8" per spostarsi a destra.

```
   5 GO SUB 1000
  10 RANDOMIZE
  20 FLASH 0 : BRIGHT 0: INK 0:
BORDER 0: PAPER 0: CLS
  30 LET SCORE=0
  40 LET ACROSS=INT (RND*32)
  50 LET SCORE=SCORE+1
  60 POKE 23692,255: REM AUTO-SC
ROLL
  70 PRINT INK 7;AT 21,RND*31;"A
";AT 21,RND*31;"A";AT 21,RND*31;
"A"
  80 PRINT AT 10,ACROSS;" ";AT 2
1,31;""
  90 LET ACROSS=ACROSS-(INKEY$="
5" AND ACROSS>0)+(INKEY$="8" AND
 ACROSS<31)
 100 IF ATTR (10,ACROSS)=0 THEN
PRINT AT 10,ACROSS; INK 4;"B": G
O TO 50
 110 PRINT AT 0,0; INK 7;"YOU SC
ORED ";SCORE;AT 10,ACROSS; INK 4
; FLASH 1;"B"
 120 INK 9: STOP
1000 REM MAKE GR. A INTO STAR
1010 FOR A=0 TO 15
1020 READ UDG
1030 POKE USR "A"+A,UDG
1040 NEXT A
1050 DATA 16,56,254,124,56,108,1
30,0,195,165,90,66,36,36,24,24
1060 RETURN
```

Figura 9

Conclusioni

Perché tradurre un programma da ZX 81 a ZX Spectrum al giorno d'oggi?

Nel mondo del retrocomputing esistono competizioni di programmazione per realizzare software con il minor numero di righe di programma. I piccoli programmi realizzati per ZX 81 sono una valida palestra per aumentare la propria abilità.

Oppure si desidera far girare su ZX Spectrum (reale o emulato) un programma riscritto per sfruttare le caratteristiche superiori (colore, grafica, suono).

Non esistono "regole" fisse di conversione. Per non andare a caccia di mostri e fantasmi durante la fase di debugging, è conveniente un "reverse engineering" preliminare per comprendere il comportamento del programma e successivamente migliorare solo le parti relative all'Input-Output, lasciando inalterate quelle costituite dai calcoli.

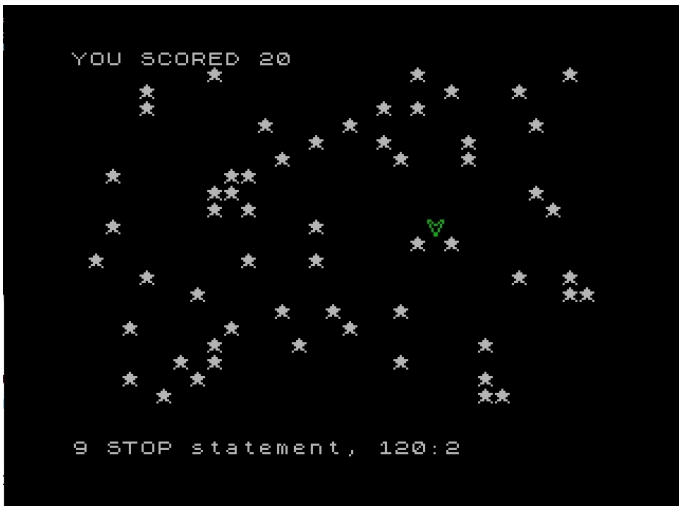


Figura 10: schermata del gioco di Fig.9

Bibliografia

[Bon82] R.Bonelli, "Guida al Sinclair ZX81", Gruppo Editoriale Jackson, 1982,
https://archive.org/details/guidaalsinclairzx81zx80enuovarom

[Bon83] R.Bonelli, "Alla scoperta dello ZX Spectrum", Gruppo Editoriale Jackson, 1983,
https://archive.org/details/allascopertadellozxspectrum

[Hew84] A.Hewson, "ZX equivalents", Sinclair User Magazine, n. 25, Apr.1984, pagg.122-124,
https://archive.org/details/sinclair-user-magazine-025

[Jon83] D.Jones, "Delving deeper into your ZX Spectrum", Interface Publications, 1983.

[Lis84] LIST n.1, Jan-Feb 1984, pag.28,
https://archive.org/details/LIST1984-01

[Lis84b] LIST n.5, Sep-Oct 1984, pag.92,
https://archive.org/details/LIST1984-05

Raccolta pubblicazioni LIST (non completa):
https://archive.org/details/listprogrammi

Alien Attack!

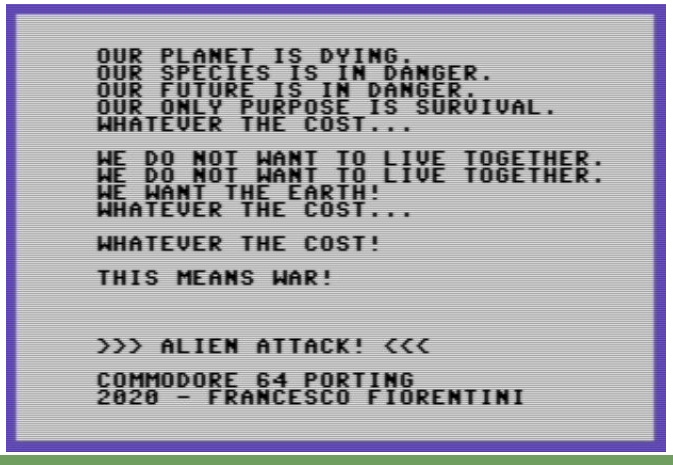
dall'Amstrad CPC al Commodore 64 tramite il Simons' Basic

di Francesco Fiorentini

Dopo avervi parlato nel numero scorso del gioco Alien Attack!, scritto in Locomotive Basic per Amstrad CPC, mi sono reso conto che era ormai diverso tempo che non scrivevo qualcosa per il mio primo computer, il Commodore 64. Effettivamente era una lacuna che andava colmata ed armato di buona volontá ho deciso di convertire il gioco per il biscottone. A prima vista potrebbe sembrare un'operazione piuttosto semplice, in fin dei conti si tratta di due macchine ad 8bit, quasi contemporanee, molto simili come resa grafica ed entrambe equipaggiate di un dialetto Basic. Ma, come i vecchi adagi insegnano, il diavolo si nasconde nei dettagli.

Come ho gia' avuto modo di scrivere diverse volte, il Locomotive Basic che equipaggia l'Amstrad CPC e' un dialetto Basic veramente evoluto, niente a che vedere con le limitate capacitá del Basic V2 che equipaggia invece il Commodore 64. I comandi che i due dialetti Basic hanno in comune sono pochi ed io, ovviamente, per scrivere il mio gioco avevo usato buona parte di quelli specifici del Locomotive. Dopo una rapida analisi ero comunque ragionevolmente convinto che sarei riuscito a trasferire tutto il codice abbastanza agevolmente, con la sola eccezione della ridefinizione dei caratteri.

Ridefinire i caratteri sul Commodore 64 é un'operazione piuttosto lunga e noiosa, mentre e' decisamente semplice sull'Amstrad CPC grazie all'istruzione SYMBOL. Ovviamente avrei evitato di ridefinire l'intero character set, ma per generare i tiles del gioco ed il suo protagonista, nonche' la bomba, avrei comunque dovuto ridisegnare alcuni caratteri. Quando ero ormai rassegnato a fare questo sforzo, pur di convertire il gioco, mi e' venuta in mente un'idea che rispondeva al nome di Simons' Basic.



Simons' Basic

La limitatezza dei comandi del Basic V2 e la conseguente relativa facilita' con cui questi potevano essere aggiunti, ha fatto si' che fiorissero nel corso degli anni diverse estensioni Basic per il Commodore 64. Indubbiamente una delle piu' famose e' il Simons' Basic.

Creato da un talentuoso ragazzino di soli 16 anni, questo software fu messo in commercio nel 1983 e sin da allora ha riscosso un successo incredibile. Chiunque abbia mai posseduto un C64 ha sentito almeno una volta nominare il Simons' Basic.

Il programma, generalmente distribuito su cartuccia, é oggigiorno facilmente reperibile anche in formato D64 ed aggiunge ben 114 comandi al Basic standard!

Scopo di questo articolo non é quello di illustrare tutte le capacitá del Simons' Basic, non basterebbe un libro, quanto quello di indicare alcune peculiaritá che mi sono state utili per il porting del mio gioco.

Ridefinizione dei caratteri

Come accennato, la ridefinizione dei caratteri sul biscottone é operazione piuttosto lunga, per fortuna invece il Simons' Basic facilita notevolmente il tutto: non ai livelli del Locomotive, ma quasi.

Innanzitutto prima di poter ridefinire i caratteri, questi devono essere copiati dalla ROM alla RAM. Questa operazione che in Basic V2 é alquanto complicata, si traduce in Simons' Basic nella sola istruzione **MEM**.

Riporto fedelmente dal manuale: *il comando MEM trasferisce una copia del set di caratteri da ROM su RAM, sotto il controllo del sistema operativo Kernal.* ***Lo schermo viene spostato alla locazione $CC00***.

Tenete bene a mente la parte evidenziata, piú tardi vi spiegheró il perché.

Dopo aver copiato i caratteri da ROM a RAM, questi possono finalmente essere ridefiniti tramite l'istruzione **DESIGN** e il conseguente comando **@**. Si vedano nel codice allegato le righe da 6000 a 6200.

Non semplice come il comando SYMBOL del Locomotive, ma ci si avvicina abbastanza. Ho anche creato un foglio Excel per automatizzare la scrittura delle righe necessarie per la ridefinizione di un singolo carattere: chi fosse interessato puó richiedermelo direttamente.



PRINT AT

Un'altra peculiaritá del Simons' Basic é quella di aggiungere al Basic standard la funzione PRINT AT, paragonabile al comando LOCATE del Locomotive.

La scelta del Simons' Basic era stata fatta anche per questo motivo, visto che il mio gioco Amstrad fa largo uso di questa funzione per stampare gli oggetti a schermo. Purtroppo devo ammettere che, per ragioni di performance, ho potuto beneficiare di questo comando soltanto per alcune parti del codice, mentre per le animanzioni del gioco ed il controllo delle collisioni ho dovuto riscrivere la logica da zero.

PEEK e POKE

Nel programma per Amstrad CPC il movimento dell'astronave viene simulato da una serie di caratteri stampati sullo schermo e posizionati tramite la funzione LOCATE. Sebbene fosse possibile fare lo stesso in Simon's Basic usando l'istruzione PRINT AT, per aumentare la velocitá ho preferito stampare i caratteri a video scrivendoli direttamente in memoria tramite il comando POKE. Si potrebbe pensare che sia difficile calcolare tutte le posizioni in memoria, ma quello che a prima vista potrebbe sembrare un calcolo complicato é in realtá piuttosto semplice se si tiene bene a mente la mappa della memoria video del Commodore 64. Lo memoria video del Commodore 64 e' formata da 25 righe e 40 colonne che generalmente partono dalla locazione di memoria $400 (1024). Ma se vi ricordate bene la puntualizzazione del comando MEM nel manuale del Simons' Basic, dopo l'esecuzione di tale comando la memoria video del Commodore 64 viene spostata alla locazione **$CC00** (**52224**).

Il movimento dell'astronave aliena é giá gestito da due cicli FOR, uno per le righe e uno per le colonne. A questo punto credo che sia abbastanza evidente che per stampare a video l'astronave aliena sia possibile utilizzare le coordinate fornite dai due cicli FOR ed sommarle al valore iniziale della memoria video.

Si veda la riga 1110 che valorizza la variabile AM.

Una volta calcolata la posizione dell'astronave, basta scrivere il carattere corrispondente direttamente nella memoria video tramite il comando POKE.

Si veda la riga 1200 che stampa la parte anteriore dell'astronave.

La riga 1220 stampa invece la parte posteriore dell'astronave (vi ricordo che l'astronave é formata da due caratteri).

La riga 1230 invece si preoccupa di ripulire l'eventuale scia dell'astronave, stampando a video il carattere 32 (uno spazio). E' evidente come ripetendo questa operazione si crei l'effetto del movimento da sinistra verso destra dell'astronave.

La stessa logica é stata adottata per la gestione della bomba.

E le collisioni?

In Locomotive Basic le collisioni erano gestite tramite il comando COPYCHR$(), capace di rivelare il carattere presente in una ben determinata posizione raggiunta tramite la funzione LOCATE.

Ma aspetta un momento, io la posizione della mia astronave la conosco di giá... forse posso leggere il valore del carattere presente in quella determinata posizione leggendo direttamente il valore della memoria video. Come? Usando l'istruzione PEEK ovviamente.

Si veda di nuovo la riga 1110.

Dopodiché basterá controllare questo valore e, se diverso da uno spazio o dall'astronave stessa, significa che c'é una collisione.

Si veda la riga 1140.

IF THEN :ELSE

Un'altra ragione che mi ha spinto a utilizzare il Simons' Basic é la possibilitá di usare il costrutto IF THEN :ELSE. Come tutti sappiamo, il Basic V2 é carente della condizione ELSE nel comando IF. A mio parere é una mancanza piuttosto importante perché costringe il programmatore a fare degli inutili controlli ed appesantire la giá scarsa leggibilitá del programma Basic. Per fortuna il nostro



amico Simons' Basic mette una pezza a questa mancanza implementando la condizione ELSE tramite la sintassi :ELSE. Ho dovuto riscrivere parte delle condizioni IF THEN ELSE per adattarle alla nuova sintassi, ma almeno non ho dovuto riscrivere la logica di questa parte del codice.

Bene, il gioco a questo punto é pronto. Il porting verso il C64 mi é costato solo un paio di pomeriggi di adattamento, grazie soprattutto alla flessibilitá del Simons' Basic.

Ah, dimenticavo... La bomba si sgancia con il tasto '1'. Buon divertimento!

Link utili:

Da questa pagina potete scaricare il Simons' Basic:
https://www.c64-wiki.com/wiki/Simons%27_BASIC

Qui invece potete trovare il manuale in italiano:
https://archive.org/details/simonsbasic_it

```
10 rem **********************************
11 rem alien attack! by francesco fiorentini
12 rem commodore 64 - simons' basic
15 rem settembre 2020
16 rem **********************************
17 hi=5000
18 gosub 10000: rem intro
20 print chr$(147)

29 rem definizione caratteri
30 gosub 6000
37 rem np=numero palazzi iniziale - th=top high
38 rem vm=starting point of video memory after mem
39 rem it is 52224 but my iterations start at 1...
40 np=6:th=6:lv=1:lf=3:pt=0:ex=15000:vm=52223

69 rem disegno dello sfondo
70 gosub 7000
99 rem inizializzazione parametri di gioco
100 bf=0:i=0

998 rem movimento dell'alieno
999 rem r=row am=alienmovement
1000 for r = 1 to 23
1100 for i = 1 to 40
1110 am=vm+i+((r-1)*40): ck=peek(am)
1120 rem controllo collisione
1140 if ck<>32 and ck<>254 and ck<>255 then goto 3000
1200 poke am,af: poke am+3072,0
1220 if i>1 then poke am-1,ab
1230 if i>2 then poke am-2,32
1239 rem controllo della bomba
1240 gosub 5000
1280 next i
1285 poke am,32: poke am-1,32
1290 pt=pt+50: if pt>=ex then 1291:else: goto 1298
1291 ex=ex+15000:lf=lf+1
1292 print at(1,23) "level:"; lv ;"- lives:"; lf ; "- points:"; pt
1293 goto 1299
1298 if lv< 10 then print at(30,23) pt :else:print at(31,23) pt
1299 if sf$="0000000000000000000000000000000000000000" then goto 1330
1300 rem print at(1,23) sf$
1301 next r
1309 rem level completed!!! move to next one
1310 gosub 6500
1320 goto 70
1329 rem livello completato anzitempo
1330 pt=pt+(23-r)*100: goto 1310

3000 rem crash
3005 cr$="***":pr$="   ":x=i-2:if x<=0 then x=1:cr$="**":pcr$="  "
3010 print at(x-1,r-1) cr$
3020 lf=lf-1
3030 if lf=0 then goto 3500
3040 for t=1 to 1000: next t
3050 print at(x-1,r-1) pr$
3060 print at(1,23) "level:"; lv ;"- lives:"; lf ; "- points:"; pt
3061 rem print at(10,25) "- hi-score:"; hi; "-"
3070 goto 99
```

```
3499 rem game over
3500 print chr$(147)
3510 print at(9,6) "----  game over  ----"
3511 print at(13,8) "score: "; pt
3525 for t=1 to 1000: next t
3530 print at(9,10) "-- play again? y/n --"
3531 if pt>hi then hi=pt
3550 get re$
3560 if re$="y" or re$="y" then goto 20
3570 if re$="n" or re$="n" then goto 4999
3580 goto 3550
4998 rem fine del gioco
4999 end

5000 rem gestione bomba - br=bombrow - bl=bombline bf=bombflag(0/1=n/y)
5005 get kp
5010 if kp=1 and bf=0 then 5020:else: goto 5025
5019 rem nuova bomba
5020 br=r+1: bl=i: bf=1: goto 5190
5024 rem controllo se esiste una bomba o meno
5025 if bf=0 then 5200:else: br=br+1
5030 if br=24 then goto 5210
5040 poke vm+((br-2)*40)+bl,32
5190 poke vm+((br-1)*40)+bl,bo
5200 return
5210 br=0:bf=0:poke vm+(22*40)+bl,32:sf$=inst("0",sf$,bl)
5220 goto 5200

6000 rem ridefinisce i caratteri
6001 rem **** attenzione importante *****
6002 rem $0400-$0757 video-ram before mem
6003 rem video memory starts at 1024
6004 rem $cc00-$cfff video-ram after mem
6005 rem video memory starts at 52224
6009 mem
6010 design 2,$e000 + 254 * 8
6011 @......bb
6012 @.....bbb
6013 @...bbbbb
6014 @..bbb..b
6015 @b.bb.bbb
6016 @bbbbbbbb
6017 @..b...b.
6018 @.b.b.b.b
6020 design 2,$e000 + 255 * 8
6021 @bb......
6022 @bbb.....
6023 @bbbbb...
6024 @b..bbb.b
6025 @bb.bb.bb
6026 @bbbbbbbb
6027 @.b...b..
6028 @b.b.b.b.
6030 design 2,$e000 + 253 * 8
6031 @........
6032 @..b..b..
6033 @...bb...
6034 @..bbbb..
6035 @..bbbb..
6036 @..bbbb..
6037 @..bbbb..
6038 @...bb...
6039 rem disegna i piani del palazzo char (230,231,232,233,234,235)
6040 design 2,$e000 + 230 * 8
6041 @bbbbbbbb
6042 @bb.bb.bb
6043 @bbbbbbbb
6044 @bb.bb.bb
6045 @bbbbbbbb
6046 @bb.bb.bb
6047 @bbbbbbbb
6048 @bb.bb.bb
6050 design 2,$e000 + 231 * 8
6051 @bbbbbbbb
```

```
6052 @bbbbb.bb
6053 @bbbbbbbb
6054 @bb.bbbbb
6055 @bbbbb.bb
6056 @bbbbbbbb
6057 @bb.bbbbb
6058 @bbbbbbbb
6060 design 2,$e000 + 232 * 8
6061 @bbbbbbbb
6062 @bb.bb.bb
6063 @bbbbbbbb
6064 @bb.bbbbb
6065 @bbbbb.bb
6066 @bb.bbbbb
6067 @bbbbbbbb
6068 @bbbbbbbb
6070 design 2,$e000 + 233 * 8
6071 @bbbbbbbb
6072 @bbbbbbbb
6073 @bb.bb.bb
6074 @bbbbbbbb
6075 @bb.bb.bb
6076 @bbbbbbbb
6077 @bbbbbbbb
6078 @bbbbbbbb
6080 design 2,$e000 + 234 * 8
6081 @bbbbbbbb
6082 @bb.bb.bb
6083 @bbbbbbbb
6084 @bbbb.bbb
6085 @bbbbbbbb
6086 @bb.bb.bb
6087 @bbbbbbbb
6088 @bbbbbbbb
6090 design 2,$e000 + 235 * 8
6091 @bbbbbbbb
6092 @bb.bbbbb
6093 @bbbbbbbb
6094 @bbbb.bbb
6095 @bbbbbbbb
6096 @bbbbbbbb
6097 @bb.bbbbb
6098 @bbbbbbbb
6099 rem disegna top1 del palazzo ridefinendo il char (240)
6100 design 2,$e000 + 240 * 8
6101 @........
6102 @........
6103 @........
6104 @........
6105 @...bb...
6106 @..bbbb..
6107 @.b.bb.b.
6108 @b.bbbb.b
6109 rem disegna top2 del palazzo ridefinendo il char (241)
6110 design 2,$e000 + 241 * 8
6111 @b......b
6112 @b......b
6113 @b......b
6114 @b......b
6115 @bb....bb
6116 @bbb..bbb
6117 @bbbbbbbb
6118 @bbbbbbbb
6119 rem disegna top3 del palazzo ridefinendo il char (242)
6120 design 2,$e000 + 242 * 8
6121 @...bb...
6122 @.bbbbbb.
6123 @.bbbbbb.
6124 @...bb...
6125 @...bb...
6126 @..bbbb..
6127 @.bbbbbb.
6128 @bbbbbbbb
6129 rem disegna top4 del palazzo ridefinendo il char (243)
6130 design 2,$e000 + 243 * 8
```

```
6131 @...bb...
6132 @...bb...
6133 @...bb...
6134 @...bb...
6135 @...bb...
6136 @..bbbb..
6137 @.bbbbbb.
6138 @bbbbbbbb
6139 rem af=alienfront-ab=alienback-bo=bomb
6190 af=255:ab=254:bo=253
6200 return

6499 rem gestione dinamica dei livelli
6500 np=np+1:lv=lv+1:pt=pt+500:th=th+1
6501 if pt>=ex then ex=ex+15000:lf=lf+1
6504 if th>14 then th=15:if np>19 then np=19
6505 print at(9,4) "----------------------"
6510 print at(9,5) "-- go to level";lv;"---"
6520 print at(9,6) "----------------------"
6525 for t=1 to 1000: next t
6530 print at(9,7) "-- r u ready? y/n ---"
6540 print at(9,8) "----------------------"
6550 get re$
6560 if re$="y" or re$="Y" then goto 6591
6565 goto 6550
6591 print chr$(147)
6600 return

7000 rem disegna lo sfondo
7001 rem al=aleatorio - np=numero palazzi - p=colore della penna
7002 sf$="00000000000000000000000000000000000000000000"
7003 print at(1,24) "level:"; lv ;"- lives:"; lf ; "- points:"; pt
7004 rem print at(1,25) "- hi-score:"; hi; "-"
7010 x=20-(np/2)
7020 for n=1 to np
7022   rem cambia il top del palazzo x=int(rnd(1)*(high-low))+low
7025   a=int((rnd(1)*4)+1)
7026   pp=239+a
7030   hi=int(rnd(1)*(th-2))+2
7031   ox=x
7041   sf$=inst("1",sf$,x)
7042   x=ox
7050   for i=hi-1 to 0 step -1
7051     rem cambia il piano x=int(rnd(1)*(high-low))+low
7052     a=int((rnd(1)*6)+1)
7053     pf=229+a
7063     pk=vm+((23-i-1)*40)+x
7065     if i=hi-1 then 7066:else:goto 7070
7066     poke pk,pp: poke pk+3072,2
7067     goto 7090
7070     poke pk,pf: poke pk+3072,2
7090   next i
7091 x=x+1
7100 next n
7300 return

10000 rem intro grafica
10001 ? chr$(147)
10060 print at(5, 2) "our planet is dying."
10061 print at(5, 3) "our species is in danger."
10062 print at(5, 4) "our future is in danger."
10063 print at(5, 5) "our only purpose is survival."
10064 print at(5, 6) "whatever the cost..."
10065 print at(5, 8) "we do not want to live together."
10066 print at(5, 9) "we do not want to live together."
10067 print at(5, 10) "we want the earth!"
10068 print at(5, 11) "whatever the cost..."
10069 print at(5, 13) "whatever the cost!"
10070 print at(5, 15) "this means war!"
10075 for i=1 to 2000:next i
10080 print at(5,19) ">>> alien attack! <<<"
10085 print at(5,21) "commodore 64 porting"
10090 print at(5,22) "2020 - francesco fiorentini"
10100 for i=1 to 3000:next i
10110 return
```

Bombs Away!

di Gianluca Girelli

Come scritto da Francesco Fiorentini nel suo articolo sul N.25, un paio di mesi fa i nostri cuori da retrocoders hanno battuto molto forte durante la "challenge" realativa alla realizzazione di un clone di Air Attack, vecchio gioco degli anni '80 realizzato praticamente per ogni piattaforma di allora. A distanza di tanti anni, è incredibile come questo semplice gioco non solo abbia resistito allo scorrere del tempo ma sia di fatto diventato un'icona in grado di attirare interesse anche nel mondo moderno.
E' ancora più straordinario come, in tempi non sospetti (circa un mese prima della challenge) abbia deciso di realizzarne una mia personale versione da inserire come minigioco all'interno di un gioco più grande di derivazione next-gen.
Mentre questa idea mi frullava per la testa, la sfida ha preso corpo sulle pagine di RPI e quindi, invece di programmare il gioco usando tecniche "avanzate" (layers o double-buffering su uno schermo grafico), ho pensato di implementarlo usando il più possibile un codice che ne mimasse il comportamento originale (schermo di testo e caratteri ridefiniti).

A differenza del buon Francesco, non ho fatto in tempo a completare il lavoro entro la data del 24 Settembre, ma spero che lo sforzo possa comunque risultare di vostro gradimento.

Per questioni di brevità, mi concenterò qui solo sulle differenze di questo port specifico (scritto in Hollywood) lasciando invece al lettore il compito di studiare nel dettaglio l'implementazione originaria (cfr: RMW edizione italiana n.25, pag. 22).

Partiamo innanzitutto dai vincoli fissati. Poichè come detto si trattava di un minigioco da attivare all'interno di un gioco più complesso e funzionale al suo proseguimento, questa particolare implementazione di "Air Attack" (che ho rinominato "Blast Away!") non poteva costituire una vera e propria sfida di per sé, pena un livello complessivo di difficoltà troppo elevato per l'utente occasionale. Ecco perchè ho quindi optato per:
- 1 solo livello;
- 1 sola vita;
- numero limitato di palazzi da distruggere;



Figura 1 - Bombs Away! splashscreen

- nel caso la bomba vada a segno distruggerà il palazzo per intero (a differenza di altre implementazioni);
- livello di difficoltà gioco mediamente facile (high score non importante).

Il passo successivo è stato quello di definire la grafica: tramite un qualsiasi programma (io ho usato Gimp) ho disegnato i vari elementi di base, cercando di dar loro un aspetto il più possibile da carattere ridefinito (vedi anche gli articoli di Francesco sull'argomento).

Anche nel mio caso ho usato alcune tessere diverse (tiles) per le forme base dei grattacieli, una serie di "tetti" che potessero ulteriormente diversificarli, una base a scacchi per simulare un prato (o comunque fondamenta della città) ed il fatidico "bombardiere" che, visto il tema spaziale del gioco in cui Blast Away! è "affogato", ho deciso di disegnare a similitudine dei "Viper Coloniali" della vecchia serie TV "Battlestar Galactica" degli anni '80. Completa l'opera un omino stilizzato con paracadute aperto nel caso in cui la navetta vada ad impattare contro un palazzo. Lo splash screen iniziale è invece opera dell'amico Marco Riva.

Il codice allegato al presente articolo è abbastanza commentato da essere autoesplicativo, tuttavia ci sono alcuni punti su cui è necessario fare luce. Innanzitutto è da notare la presenza di due procedure, p_Update_input() e p_MouseState(), che non sono presenti nell'implementazione originaria in BASIC. Il motivo della loro introduzione è che i sistemi odierni sono, ovviamente, estremamente più veloci che nel passato ed un semplice click del mouse corrisponde in realtà a molteplici "stati" che vengono immagazzinati nel buffer di input/output.

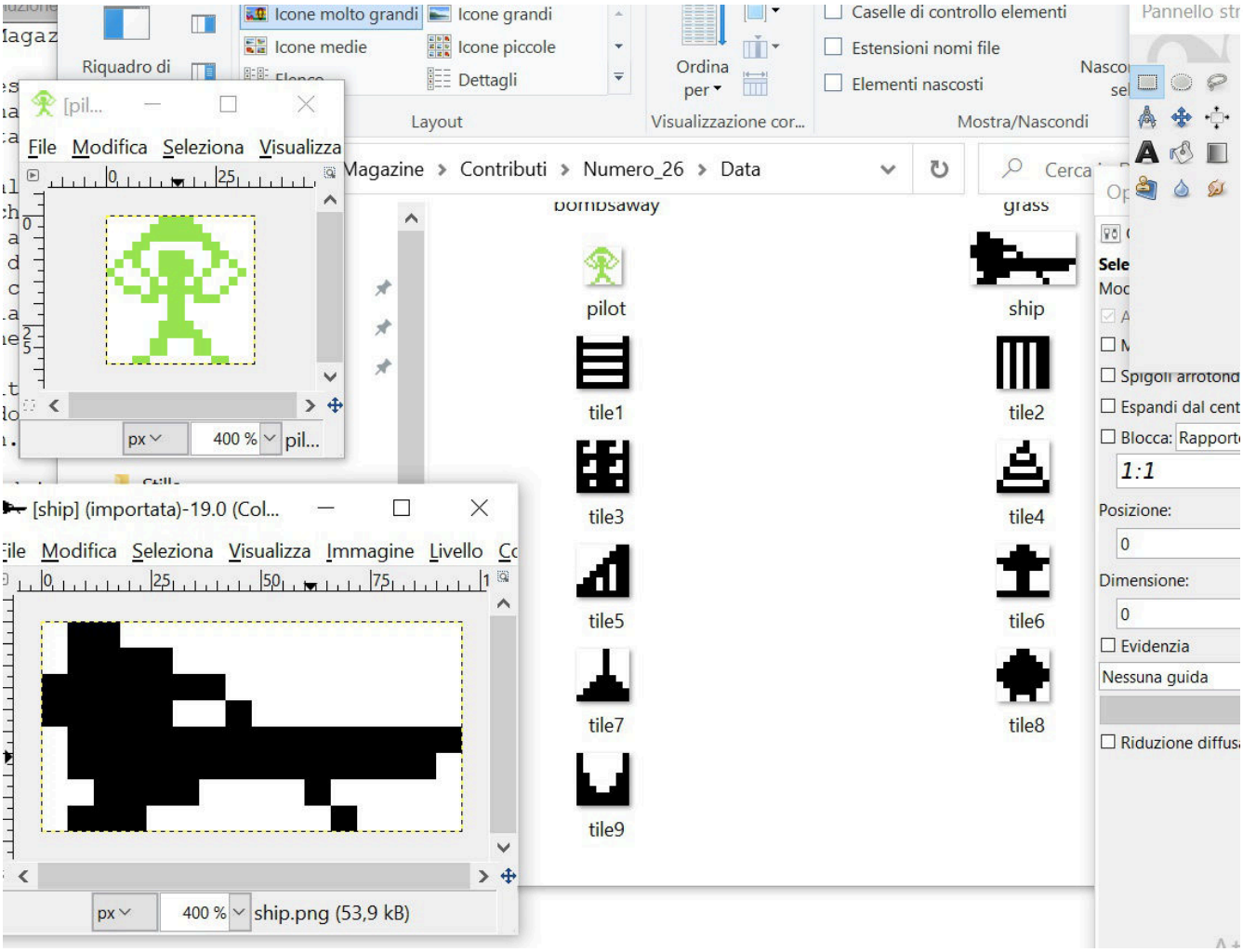
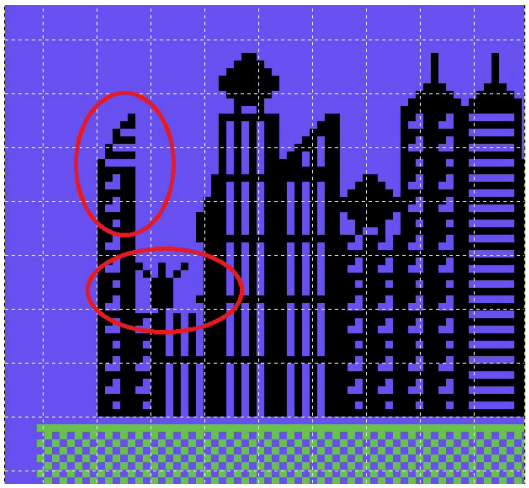


Figura 2 - La grafica del gioco

La conseguenza, a prescindere dalla velocità dell'utente nel premere e rilasciare il pulsante, è che il sistema continuerà a reagire come se il giocatore stesse continuamente clickando per sparare. Senza entrare nel dettaglio, le predette procedure rendono "discreto" uno stato che altrimenti non lo sarebbe e limitano il programma a reagire una sola volta per click.

Un'altra cosa che è stato necessario limitare è il "range" di posizioni orizzontali da dove la bomba può essere sganciata.

Mi spiego meglio: a differenza dell'implementazione classica sullo schermo testuale, dove la navetta si sposta di colonna in colonna, una alla volta, su un numero di colonne limitato (ad esempio 40 sul C64/C128, ognuna della larghezza di 8 pixels), in questo caso la nostra bomba può cadere a partire da un qualsiasi punto sullo schermo alla risoluzione di un singolo pixel, trattandosi di un'implementazione in ambiente completamente grafico. Questo problema comporta il fatto che un palazzo possa essere distrutto solo in parte poiché la bomba, non cadendo esattamente all'interno della colonna che contiene il palazzo in questione, non fa altro che "affettarlo" secondo la direttrice vericale (vedasi figura).



Ciò risulta in una moltiplicazione virtuale del numero dei palazzi a schermo che finisce con l'invalidare le ipotesi iniziali. La soluzione che ho individuato è stata quella di mimare, in modo trasparente all'utente, l'effetto dell'esistenza delle singole colonne: grazie a Gimp ho quindi calcolato e memorizzato in un vettore le coordinate degli estremi sinistro e destro di ogni singolo palazzo. Il codice quindi, al momento in cui rileva la pressione del

mouse e decide che è possibile sganciare la bomba (nessuna altra bomba già in volo) ne rileva la coordinata Y, la confronta con quelle all'interno dell'array e, dopo averla corretta di un adeguato "offset", la lascia partire.

Per quanto riguarda il main loop del gioco, esso è abbastanza semplice e segue i principi già teorizzati nell'articolo sui "Cenni di Game Coding" pubblicato sul numero 17: l'obiettivo è mantenere il tutto più semplice e lineare possibile, in modo da facilitare manutenzione e future espansioni del codice. La sezione di controllo del gameplay è virtualmente staccata dal modo di implementarla a video, in modo da poter più facilmente adattare il programma a diversi ambienti di sviluppo. Più nel dettaglio, prima si leggono tutti gli input e si calcolano le reazioni; solo alla fine si realizza la grafica con un singolo ciclo di rendering. In tale modo il movimento risulta più fluido e privo degli "hang ups" che si percepiscono quando il render avviene in più fasi (es: calcola posione navetta; disegna navetta; calcola posione bomba; se c'è un impatto mostra animazione esplosione; disegna bomba) perchè durante l'esecuzione di una singola fase di animazione il resto è bloccato.

Messo in "linguaggio naturale":

```
Ripeti
   leggi input
   determina se è possibile sganciare una bomba
   calcola la prossima posizione della
navetta
   se la bomba è in volo, calcola la prossima
posizione della bomba
    calcola nuovo punteggio in caso di collisioni
bomba/palazzo
   disegna lo schermo (navetta, bomba, frame di
animazione esplosione etc)
   controlla le condizioni per game over/game won
Fino a quando exit=True
```

Per quanto riguarda le condizioni di fine gioco il meccanismo è molto semplice: se la prossima posizione della navetta è già occupata (il controllo viene effettuato verificando il colore del primo pixel oltre il muso della navetta: se è nero abbiamo davanti a noi un palazzo) siamo in collisione (game over); mentre se tutti i palazzi sono stati distrutti abbiamo vinto. Come Francesco, anch'io ho deciso di legare tale controllo al contenuto di un array il cui numero di celle è pari alle colonne virtuali su schermo ed il contenuto della singola cella è "1" se un palazzo esiste ed è ancora in piedi.



Figura 3 - Grafica in-game

Il codice contiene anche controlli aggiuntivi, la cui spiegazione non è però importante in questa sede.

Infine, poichè i giochi moderni girano in full-screen ed era invece necessario simulare un vecchio monitor dotato di bordi attorno allo schermo, ho usato il comando Hollywood "clip region":

```
CreateClipRegion(1, #BOX, 190, 53, 928, 623)
SetClipRegion(1)
```

Questo comando serve per creare ed attivare una zona sullo schermo (che può avere anche forme complesse) che agisce come maschera: tutto ciò che è al suo interno viene visualizzato a video, mentre il resto viene nascosto. Ecco che dunque la nostra navetta comparirà emergendo dal bordo sinistro dello schermo per poi scomparire gradualmente mentre attraversa il bordo destro.

Il codice di Francesco, relativo all'implementazione del suo Alien Attack su Amstrad, finisce con la ridefinizione dei caratteri necessari per stampare a video istruzioni e punteggi. Mentre un port di tale codice per C64 e C128 verrá pubblicato su uno dei prossimi numeri di RMW, ho optato per usare un font futuristico già pronto. Tale font, chiamato "Neuropol", è gratuito e liberamente scaricabile per scopi non commerciali dal link a fondo pagina.
Nonstante il codice riportato non sia stato sviluppato per sistemi retro, spero ne apprezzerete comunque lo spirito e spero che le tecniche di programmazione qui dimostrate possano essere di vostra utilità per il futuro. Blast Away! gira su vari sistemi, il primo dei quali è l'AmigaOS4.1. Tramite Hollywood è facilmente compilabile per sistemi "Classic" (AmigaOS3.x) a patto però di abbassare le dimensioni della grafica.

Sorgente e compilato per Win32 si possono scaricare da: www.gdg-entertainment.it/rmw/bombs_away.zip
Buon divertimento!

```
/*************************************************************
**                                                          **
** Name:        Bombs Away!                                 **
** Author:      Gianluca Girelli                            **
** Original Code and Graphics: Gianluca Girelli             **
** Additional Graphics: Marco Riva                          **
** Version:     1.0                                         **
** Date:        08.09.20                                    **
** Last update: 19.10.20                                    **
** Interpreter: any                                         **
** Licence:     Creative Common 4.0 (CC BY-NC-SA 4.0 INT)   **
** Function:    Hollywood implementation of "Air Attack"    **
**              8-bit videogame.                            **
** History:                                                 **
** 1.0: (22.09.20)                                          **
** -game complete                                           **
** 0.1: (08.09.20)                                          **
** -Initial test                                            **
*************************************************************/

@DISPLAY 1, {X=#CENTER, Y=#CENTER, Width=1280, Height=720, HideTitleBar=True,
   Sizeable=True, ScaleMode=#SCALEMODE_AUTO, Mode = "ask", FitScale = True, KeepProportions = True}

@FONT 1, "neuropol", 36,{Engine=#FONTENGINE_INBUILT}

state=0 ;must be global variable

Function p_Update_input()
;store previous mouse state and read the new one
        laststate=currentstate
        currentstate=IsLeftMouse()
EndFunction

Function p_MouseState()
; compare present and previous mouse state to determine action to take.
; Events are triggered only if mouse button was just release (mouse state=3)
        If laststate=True
                If currentstate=True
                        state=4         ;still pressed
                Else
                        state=3         ;just released
                EndIf
        Else
                If currentstate=True
                        state=2         ;just pressed
                Else
                        state=1         ;still released
                EndIf
        EndIf
EndFunction

Function p_AdjustCoordinates(bomb_x_coord)
;mimics a text-only screen by creating "virtual columns" to channel bomb's flight
        If bomb_x_coord>=184 And bomb_x_coord<232 Then result=0
        If bomb_x_coord>=232 And bomb_x_coord<280 Then result=1
        If bomb_x_coord>=280 And bomb_x_coord<328 Then result=2
        If bomb_x_coord>=328 And bomb_x_coord<376 Then result=3
        If bomb_x_coord>=376 And bomb_x_coord<424 Then result=4
        If bomb_x_coord>=424 And bomb_x_coord<472 Then result=5
        If bomb_x_coord>=472 And bomb_x_coord<520 Then result=6
        If bomb_x_coord>=520 And bomb_x_coord<568 Then result=7
        If bomb_x_coord>=568 And bomb_x_coord<616 Then result=8
        If bomb_x_coord>=616 And bomb_x_coord<664 Then result=9
        If bomb_x_coord>=664 And bomb_x_coord<712 Then result=10
        If bomb_x_coord>=712 And bomb_x_coord<760 Then result=11
        If bomb_x_coord>=760 And bomb_x_coord<808 Then result=12
        If bomb_x_coord>=808 And bomb_x_coord<856 Then result=13
        If bomb_x_coord>=856 And bomb_x_coord<904 Then result=14
```

```
        If bomb_x_coord>=904 And bomb_x_coord<952 Then result=15
        If bomb_x_coord>=952 And bomb_x_coord<1000 Then result=16
        If bomb_x_coord>=1000 And bomb_x_coord<1048 Then result=17
        If bomb_x_coord>=1048 And bomb_x_coord<1096 Then result=18
        If bomb_x_coord>=1096 And bomb_x_coord<1114 Then result=19
        Return(result)
EndFunction

Function p_minigame()
        LoadBrush(140, "Data/44_2.jpg")
        LoadBrush(141, "Data/tile1.png", {Transparency = #WHITE})
        LoadBrush(142, "Data/tile2.png", {Transparency = #WHITE})
        LoadBrush(143, "Data/tile3.png", {Transparency = #WHITE})
        LoadBrush(144, "Data/tile4.png", {Transparency = #WHITE})
        LoadBrush(145, "Data/tile5.png", {Transparency = #WHITE})
        LoadBrush(146, "Data/tile6.png", {Transparency = #WHITE})
        LoadBrush(147, "Data/tile7.png", {Transparency = #WHITE})
        LoadBrush(148, "Data/tile8.png", {Transparency = #WHITE})
        LoadBrush(149, "Data/tile9.png", {Transparency = #WHITE})
        LoadBrush(150, "Data/ship.png", {Transparency = #WHITE})
        LoadBrush(151, "Data/bomb.png", {Transparency = #WHITE})
        LoadBrush(152, "Data/pilot.png", {Transparency = #WHITE})
        LoadBrush(153, "Data/bombsaway.png", {Loadalpha = True})
        LoadBrush(154, "Data/grass.png", {Transparency = #WHITE})

        Local x=328             ;coordinates of first tile to be drawn
        Local y=558
        Local altpal=0          ;height of current palace
        Local whichtype=0       ;defines which kind of palace to be drawn (3 different types)
        Local whichtop=0        ;defines which kind of top to be drawn (6 different types)
        Local ship_x=94         ;ship initial coordinates
        Local ship_y=100
        Local bomb_x=bomb_y=0   ;initial bomb coordinates
        Local exit=False
        Local fire_enabled=True
        ;all buildings standing + virtual columns
        Local a = {0,0,0,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,1,0,0,0,0}
        ;virtual columns horizontal boundaries (coordinate in pixels)
        Local b =
{184,232,280,328,376,424,472,520,568,616,664,712,760,808,856,904,952,1000,1048,1096,1114}
        Local column=0
        score=0
        sum=0                   ;transit variable for standing buidings

        DisplayBrush(140, #CENTER, #CENTER)   ;background
        DisplayBrush(153, #CENTER, #CENTER)   ;title

        WaitLeftMouse()

        Cls
        SetFont("neuropol", 48)
        SetFontStyle(#BOLD)
        DisplayBrush(140, #CENTER, #CENTER)   ;background
        CreateTextObject(1, "How to play")
        CreateTextObject(2, "Your ship is slowly flying into the ground. Destroy the buildings for a safe
landing. Press left mouse button to drop a bomb.", {Align=#JUSTIFIED, wordwrap=750})
        CreateTextObject(3, "Press left mouse button to start.", {Align=#JUSTIFIED, wordwrap=650})
        DisplayTextObject(1, #CENTER, 80)
        DisplayTextObject(2, #CENTER, 200)
        DisplayTextObject(3, #CENTER, 520)

        WaitLeftMouse()

        Cls
        DisplayBrush(140, #CENTER, #CENTER)   ;background

        SetFillStyle(#FILLCOLOR)
        SetFormStyle(#NORMAL)
```

```
        CreateClipRegion(1, #BOX, 190, 53, 928, 623)
        SetClipRegion(1)

        ;generating world
        For Local i=280 To 1000 Step 48
                DisplayBrush(154, i, 612)
                Wait(3)
        Next
        For Local i=1 To 13
                altpal=Rnd(5)+1
                whichtype=Rnd(3)+1
                For Local j=1 To altpal
                        DisplayBrush(140+whichtype, x, y)
                        Wait(3)
                        y=y-48
                Next
                whichtop=Rnd(6)+4
                DisplayBrush(140+whichtop, x, y)
                y=558
                x=x+48
        Next

        SetFont("neuropol", 32)
        SetFontStyle(#BOLD)
        SetFontColor(#WHITE)
        Locate(200,45)
        Print("Score: "..score)

        ;commencing game
        Repeat
                laststate=currentstate=0 ; flush mouse buffer

                ;read input
                p_Update_input()
                p_MouseState()

                ;react to input
                If state=3 And fire_enabled And ship_x>184 ;fire has been pressed and bomb is still
unreleased
                        column=p_AdjustCoordinates(ship_x+24)
                        bomb_x=b[column]
                        bomb_y=ship_y+48
                        fire_enabled=False     ;bomb released. no more firing until it hits ground
                        state=0                ;flush mouse buffer
                EndIf

                ;compute next ship position
                If ship_x<1115         ;1015
                        ship_x=ship_x+4
                Else
                        ship_x=94      ;190
                        ship_y=ship_y+48
                EndIf

                ;if bomb's away, compute next bomb position
                If bomb_y<607 And fire_enabled=False Then bomb_y=bomb_y+8

                ;updates score in case of impact with standing building (p_GameProgress())
                If fire_enabled=False And a[column]=1
                        a[column]=0
                        score=score+50
                EndIf

                ;render screen
                DisplayBrush(150, ship_x, ship_y)
                If fire_enabled=False Then DisplayBrush(151, bomb_x, bomb_y)
                Wait(1)
                Box(ship_x, ship_y, 96, 48, $6953f5)
```

```
		If fire_enabled=False Then Box(bomb_x, bomb_y, 48, 48, $6953f5)

		If bomb_y+48>=606
			fire_enabled=True
			Box(200, 45, 500, 48, $6953f5)
			Locate(200,45)
			Print("Score: "..score)
		EndIf

		;termination conditions
		sum=0
		For Local i=0 To 12
			sum=sum+a[i]
		Next
		If score=650 And sum=0 And bomb_y+48>=606    ;game won
			;move ship out of screen boundaries on current row
			For Local i=ship_x To 1120 Step 4
				DisplayBrush(150, i, ship_y)
				Wait(1)
				Box(i, ship_y, 96, 48, $6953f5)
			Next
			;land ship
			MoveBrush(150, 190, ship_y+48, 700, 558,{Speed = #SLOWSPEED})
			Box(700, 558, 96, 48, $6953f5)
			MoveBrush(150, 700, 558, 900, 558,{Speed = #SLOWSPEED})
			SetFont("neuropol", 64)
			SetFontStyle(#BOLD)
			SetFontColor(#WHITE)
			CreateTextObject(1, "Game Won !", {Align=#JUSTIFIED, wordwrap=750})
			DisplayTextObject(1, #CENTER, #CENTER)
			If level=1 Then arcade_game_won=1 Else arcade_game2_won=1
			Wait(150)
			exit=True
		EndIf

		If (ship_x+97)<1015    ;avoid the following checks go out of boundaries
			If ReadPixel(ship_x+97, ship_y+26)=0  ;game over
				Box(ship_x, ship_y, 96, 48, $6953f5)
				While ship_y+34<607           ;pilot ejecting and gliding toward ground
					DisplayBrush(152, ship_x+34, ship_y)
					Wait(1)
					Box(ship_x+34, ship_y, 34, 34, $6953f5)
					ship_y=ship_y+2
				Wend
				SetFont("neuropol", 64)
				SetFontStyle(#BOLD)
				SetFontColor(#WHITE)
				DisplayBrush(152, ship_x+34, ship_y) ;fix pilot on ground
				CreateTextObject(1, "Game Over", {Align=#JUSTIFIED, wordwrap=750})
				DisplayTextObject(1, #CENTER, 250)
				Wait(150)
				exit=True
			EndIf
		EndIf

		WaitTimer(1, 40)  ; script timed at 25fps
	Until exit=True

	;resets game state
	exit=False

	FreeClipRegion(1)
	For Local i=140 To 154 Do FreeBrush(i)

EndFunction

StartTimer(1)
p_minigame()
```



Un bit di rarità

rovistando tra varie ed eventuali

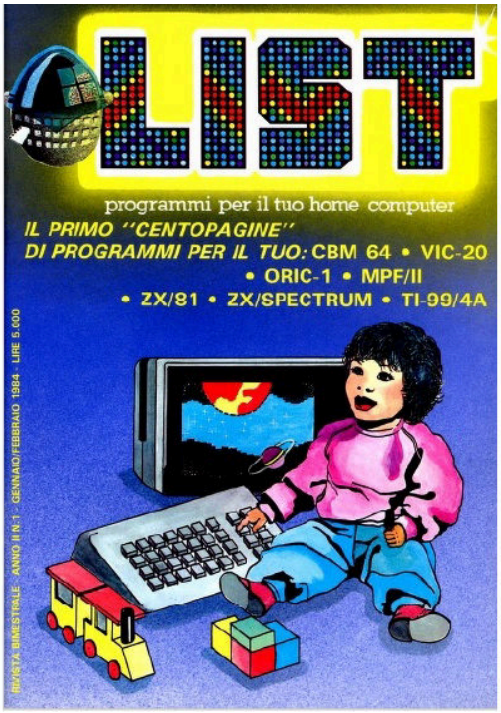


Un dolcetto senza scherzetto!



di Alberto (Ghostbuster) Apostolo

In passato, qualcuno si è lamentato perché RetroMagazineWorld parla spesso di Commodore e Sinclair. Non smetteremo mai di ripetere che RetroMagazineWorld è una rivista aperta a chiunque abbia qualcosa di interessante da raccontare su hardware e software di qualsiasi modello di computer. In questa occasione ho composto una lista ragionata di alcuni programmi e articoli pubblicati sulla rivista italiana LIST per ORIC-1, Sega SC-3000, Sharp MZ-700 e MicroProfessor II .



LIST è stata una rivista di Informatica molto sottovalutata (quando ero studente di Informatica alle Scuole Superiori, alcuni professori ne sconsigliavano l'acquisto). Quasi tutti gli articoli pubblicati erano di facile comprensione e alcuni orientati alla didattica.
Spero di avere reso un po' di giustizia a LIST e di avere fornito materiale interessante ai possessori di computer che hanno avuto poco spazio nel mercato. Chi non comprende la lingua italiana può tradurre facilmente gli articoli e le diciture nei programmi con i traduttori automatici diffusi in Rete.
La raccolta (non completa!) di LIST si trova alla pagina
https://archive.org/details/listprogrammi

https://archive.org/details/LIST1984-01
Pag. 08-08 Sega SC-3000 (technical sheet)
Pag. 50-51 ORIC-1 "La fontana" (The fountain, graphics)
Pag. 56-56 Sharp MZ-700 (technical sheet, part 1)
Pag. 57-61 ORIC-1 "Entertainer Rag" (music)
Pag. 64-67 MPF II "Oroscopo" (Horoscope, astrology)
Pag. 68-70 MPF-II "Enalotto" (Italian horse-racing pools)
Pag. 94-95 "Dentro il computer" (Inside the computer, education, part 1)

https://archive.org/details/LIST1984-02
Pag. 11-12 ORIC-1 "Tris" (Tic-Tac-Toe, game)
Pag. 15-16 ORIC-1 "Rally" (game)
Pag. 23-27 MPF II "High Driver" (game)
Pag. 37-38 MPF II "Banca" (Bank account, business)
Pag. 39-40 MPF II "Istogrammi" (Histograms, utility)
Pag. 45-45 ORIC-1 "La torre" (The Tower, math game)
Pag. 46-46 Sharp MZ-700 (technical sheet, part 2)
Pag. 47-48 Sharp MZ-700 "Mastermind" (game)
Pag. 48-49 Sharp MZ-700 "Armonia" (Harmony, astrology)
Pag. 60-62 ORIC-1 "Zhorick" (game)
Pag. 81-82 ORIC-1 "Mosca cieca" (Blindman's bluff, game)
Pag. 83-85 MPF II "O-X" (Othello,Reversi, game)

Pag. 86-86 ORIC-1 "Conversione da numero binario a decimale" (bin-to-dec, utility)
Pag. 92-94 "Dentro il computer" (Inside the computer, education, part 2)

https://archive.org/details/LIST1984-03

Pag. 13-13 ORIC-1 "Coder-Decoder" (cryptography)
Pag. 14-14 ORIC-1 "Renumber Line" (utility)
Pag. 21-22 ORIC-1 "Falciatrice" (The mower, game)
Pag. 36-39 "Dentro il computer" (Inside the computer, education, part 3)
Pag. 40-40 Correzioni al programma "Tris" (Bug-List of program "Tris", see List n. 1)
Pag. 41-41 Sharp MZ-700 "Caccia al tesoro" (Treasure hunt, game)
Pag. 42-42 ORIC-1 "Istogrammi" (Histograms, utility, Sharp MZ-700 printed in header is a mistake)
Pag. 43-45 ORIC Atmos 48K (technical sheet)
Pag. 57-58 Sega SC-3000 "Cascatutto" (Everything Falls, game)
Pag. 59-60 Sega SC-3000 "Corso Basic" (BASIC course, education, part 1)
Pag. 69-75 MPF II "Fatturazione magazzino" (Inventory invoice, business, part 1)

https://archive.org/details/LIST1984-04

Pag. 18-18 ORIC-1 "Tabella" (printing price lists, utility)
Pag. 32-33 Sega SC-3000 "Corso Basic" (BASIC course, education, part 2)
Pag. 34-35 Sega SC-3000 "Bioritmi" (Biorythm, utility)
Pag. 38-38 Sharp MZ-700 "Istogrammi di percentuali" (Histograms and percentage, utility)
Pag. 39-40 Sharp MZ-700 "Riunisci...la musica" (Collecting music, music)
Pag. 42-43 "Dentro il computer" (Inside the computer, education, part 4)
Pag. 68-69 ORIC-1 "Calendario" (Calendar utility)
Pag. 73-74 MPF II "Quattro in fila" (Four-in-a-row, game)
Pag. 88-90 MPF II "Fatturazione magazzino" (Inventory invoice, business, part 2)

https://archive.org/details/LIST1984-05

Pag. 14-14 Sega SC-3000 "Totocalcio" (Italian football pools)
Pag. 30-30 Sharp MZ-700 "Risoluzione di equazioni con il metodo di Cramer" (Cramer's method, math)
Pag. 45-47 "Dentro il computer" (Inside the computer, education, part 5)
Pag. 64-65 MPF II "Snake" (game)
Pag. 72-73 Sega SC-3000 "Corso di Basic" (BASIC course, education, part 3,4)

https://archive.org/details/LIST1984-06

Pag. 20-21 MPF II "Viaggio nello spazio" (Space journey, game)
Pag. 29-30 ORIC-1 "Gran Prix" (game)
Pag. 31-34 Sega SC-3000 "Caccia al sottomarino" (Submarine hunt, game)
Pag. 35-36 Sharp MZ-700 "Incontri di calcio" (handling football championship results)
Pag. 56-57 "Dentro il computer" (Inside the computer, education, part 6)
Pag. 66-67 Sharp MZ-700 "Tombola" (handling the Bingo)
Pag. 77-78 Sega SC-3000 "Video Picture" (graphics)
Pag. 87-89 Sega SC-3000 "Corso di Basic" (BASIC course, education, part 5)
Pag. 90-91 Sega SC-3000 "Outline" (how changing your character set)

https://archive.org/details/LIST1985-01/mode/2up

Pag. 08-09 Sega SC-3000 "Morra" (Chinese Morra, game)
Pag. 10-11 Sega SC-3000 "Mastermind" (game)
Pag. 15-17 Sharp MZ-700 "Su e giù per la penisola" (part 1, education, geography of Italy)
Pag. 22-24 Sharp MZ-700 "Attacco aereo" (Air attack, game)
Pag. 27-29 ORIC-1 "Levrieri" (Greyhounds, game)
Pag. 45-48 MPF II "Piramide" (Marienbad, game)
Pag. 49-50 Sharp MZ-700 "Istruzioni POKE" (education)
Pag. 53-56 Sega SC-3000 "Corso di Basic" (BASIC course, education, part 6,7)
Pag. 74-77 ORIC-1 "Break-Out" (game)
Pag. 78-79 Sega SC-3000 "Astro War" (game)
Pag. 80-82 Sega SC-3000 "Paroliamo" (Countdown word game)
Pag. 89-90 ORIC-1 "Salta la rana" (Frog, game)

Pag. 91-93 Sharp MZ-700 "Electronic Mastermind" (game)
Pag. 94-96 Sharp MZ-700 "Saliscendi" (Rise-and-Fall, game)

https://archive.org/details/LIST1985-02

Pag. 26-26 Sharp MZ-700 "Load and Run" (education)
Pag. 28-31 Sega SC-3000 "Corso di Basic" (BASIC course, education, part 8,9,10)
Pag. 32-35 ORIC-1 "Char-constructor" (utility)
Pag. 40-42 ORIC-1 "Smash" (The Wall, game)
Pag. 43-45 Sharp MZ-700 "Su e giù per la penisola" (part 2, education, geography of Italy)
Pag. 57-58 Sega SC-3000 "Uova Spaziali" (Space Eggs, game)
pag. 59-61 Sharp MZ-700 "JAZZI" (Jahtzee, game)
Pag. 67-69 Sharp MZ-700 "Boxe" (game)
Pag. 70-72 Sega SC-3000 "Tiro a volo" (Shooting, game)
Pag. 85-88 Sharp MZ-700 "Slot machine" (game)
Pag. 89-90 Sega SC-3000 "Pianoforte" (music)

https://archive.org/details/LIST1985-04

Pag. 31-33 Sega SC-3000 "Super Master Mind" (game)
Pag. 37-38 Sharp MZ-700 "Sci alpino" (Skiing, game)
Pag. 40-41 Sega SC-3000 "Real golf" (game)
Pag. 46-48 Sharp MZ-700 "Dispersioni termiche" (Heat loss, science)
Pag. 49-50 MPF II "Contraerea" (Flak, game)
Pag. 53-54 Sharp MZ-7000 "Bioritmi" (Biorythm, utility)
Pag. 55-56 ORIC-1 "Cubi" (Cubes, game)
Pag. 57-58 MPF II "Mastermind" (game)
Pag. 84-85 Sega SC-3000 "Archer" (game)

https://archive.org/details/LIST1985-05

Pag. 37-39 Sega SC-3000 "Grafica in 3D" (graphics)
Pag. 40-42 Sharp MZ-7000 "Caccia all'U-boot" (U-boot hunting, game)
Pag. 48-49 ORIC-1 "Oric Sequencer" (music)
Pag. 52-57 MPF II "Black-Jack" (game)
Pag. 62-63 Sega SC-3000 "Decisioni" (Choices, mathematical logic)
Pag. 64-65 Sharp MZ-700 "Analisi" (Calculus, math utility)
Pag. 70-71 ORIC-1 "Equazioni di 1° e 2° grado" (math)
Pag. 72-73 ORIC-1 "Semina" (Seeding, game)

https://archive.org/details/LIST1985-08-09

Pag. 74-78 Sega SC-3000 "Math Software" (math utility)
Pag. 79-81 Sega SC-3000 "Fasi Lunari" (Moon phases, astronomy)
Pag. 82-83 Sharp MZ-700 "Ferma il totale" (Stop the total, math game)
Pag. 84-85 Sharp MZ-700 "Biglietti da visita" (business cards printing, utility)

https://archive.org/details/LIST1985-08-09-speciali

Pag. 06-07 Sega SC-3000 "Sega Graphics" (graphics)
Pag. 19-20 Sega SC-3000 "Simulatore di volo" (Flight Simulator, game)
Pag. 21-21 Sharp MZ-700 "Archivio Sharp" (Archive, utility)

https://archive.org/details/LIST1985-10-11

Pag. 67-70 Sega SC-3000 "Bosco maledetto" (The cursed wood, adventure)
Pag. 70-71 Sega SC-3000 "Calcolo dei solidi" (Solid geometry, math)
Pag. 72-75 Sharp MZ-700 "Elenco fornitori" (Suppliers' List, business)
Pag. 75-78 Sharp MZ-700 "Sette e mezzo" (7 and half Italian BlackJack, game)

https://archive.org/details/LIST1986-04

Pag. 52-54,63-64 Sega SC-3000 "Drawer" (graphics)
Pag. 65-69 Sharp MZ-700 "Gestione Magazzino" (Inventory management, business)

The Haunted House

...ovvero come giocare lo stesso gioco su 3 piattaforme diverse!

di Francesco Fiorentini

Nell'articolo di chiusura del numero scorso vi avevo accennato che mi sarei dilettato a fare il porting di codice Basic su diverse piattaforme. Quello che avete tra le mani é soltanto il primo di questi lavori e spero che incontri il vostro gradimento.

Haunted House é un gioco, un'avventura testuale, pubblicata sul libro '**Write your own Adventure Programs for your microcomputers**' pubbicato da **Usborne Publishing** nel 1983.
Il libro mirava ad insegnare come progettare e scrivere un'avventura testuale in Basic, ed il gioco Haunted House era uno degli esempi piú significativi del volume.

Potete reperire il libro su archive.org a questo indirizzo: https://archive.org/details/Write_Your_Own_Adventure_Programs_1983_Usborne/

Mentre la versione **GW Basic** di Haunted House l'ho trovata a questo indirizzo: https://github.com/robhagemans/hoard-of-gwbasic/blob/master/AllBasicCode/ADV-OLD.BAS

Amstrad CPC
Dopo aver esaminato il codice mi sono reso conto che avrebbe potuto funzionare senza modifiche sull'**Amstrad CPC**. Ho quindi copiato ed incollato il codice su WinAPE e come per magia, l'avventura si é avviata senza problemi.

Ho provato a giocare un po' il gioco e mi é sembrato proseguire senza particolari problemi.
Possiamo quindi affermare che il porting per Amstrad CPC é stato veramente semplice. :-D

Commodore 64
Sulle ali dell'entusiasmo ho quindi deciso di lavorare sul porting per **Commodore 64**... E qui sono cominciati i dolori.
Dopo aver incollato il codice 'as is' nel VICE ho ricevuto una serie di errori che mi hanno costretto a rivedere il listato e riscrivere parte della logica.



Figura 1. Haunted House funzionante su Amstrad CPC

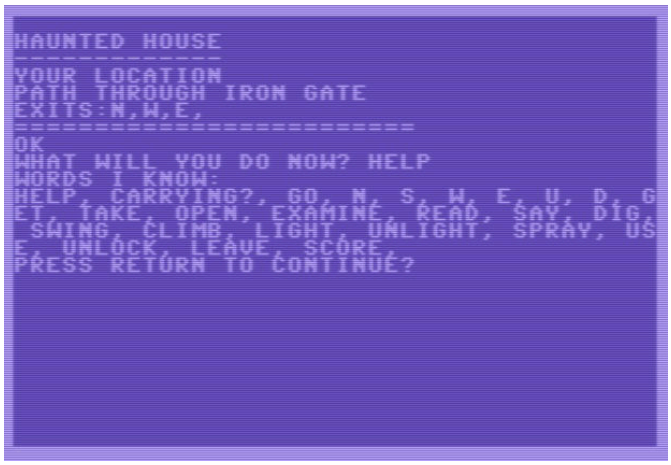


Figura 2. Haunted House porting per Commodore 64

Per fortuna il programma faceva giá uso di variabili lunghe soltanto due caratteri e quindi perfettamente compatibili con la limitazione del Basic V2, il problema principale del codice invece era che alcune righe di programma eccedevano gli 80 caratteri di lunghezza.

Alcune delle righe per fortuna potevano essere compresse semplicemente rimuovendo degli spazi inutili, ma altre invece dovevano per forza essere divise per essere accorciate ed in alcuni casi ho dovuto rivederne leggermente la logica.
Si veda per esempio la riga 270 originale che ha prodotto nella versione Commodore 64 le righe 270, 271 e 275.

Un altro esempio eclatante é la riga originale 460, che gestisce un salto GOSUB condizionato dalla variabile VB. In questo caso ho dovuto sia suddividere la riga che gestire anche il valore della variabile VB affinché il salto condizionato rimanesse efficiente.
Si vedano quindi le righe 460, 461 e 465 nel codice per Commodore 64.

Tutto sommato la conversione per il Commodore 64 é risultata abbastanza agevole, una volta capito come e dove agire.

Visual Basic 6.0
A questo punto ho voluto alzare l'asticella della difficoltá e provare a fare un porting decisamente piú impegnativo: Visual Basic 6.0.

Qualcuno potrebbe obiettare che il VB6 non é propriamente retro, ma visto che é stato rilasciato nel 1998 ed ha giá 22 anni di vita, direi che puó essere considerato retro a tutti gli effetti.

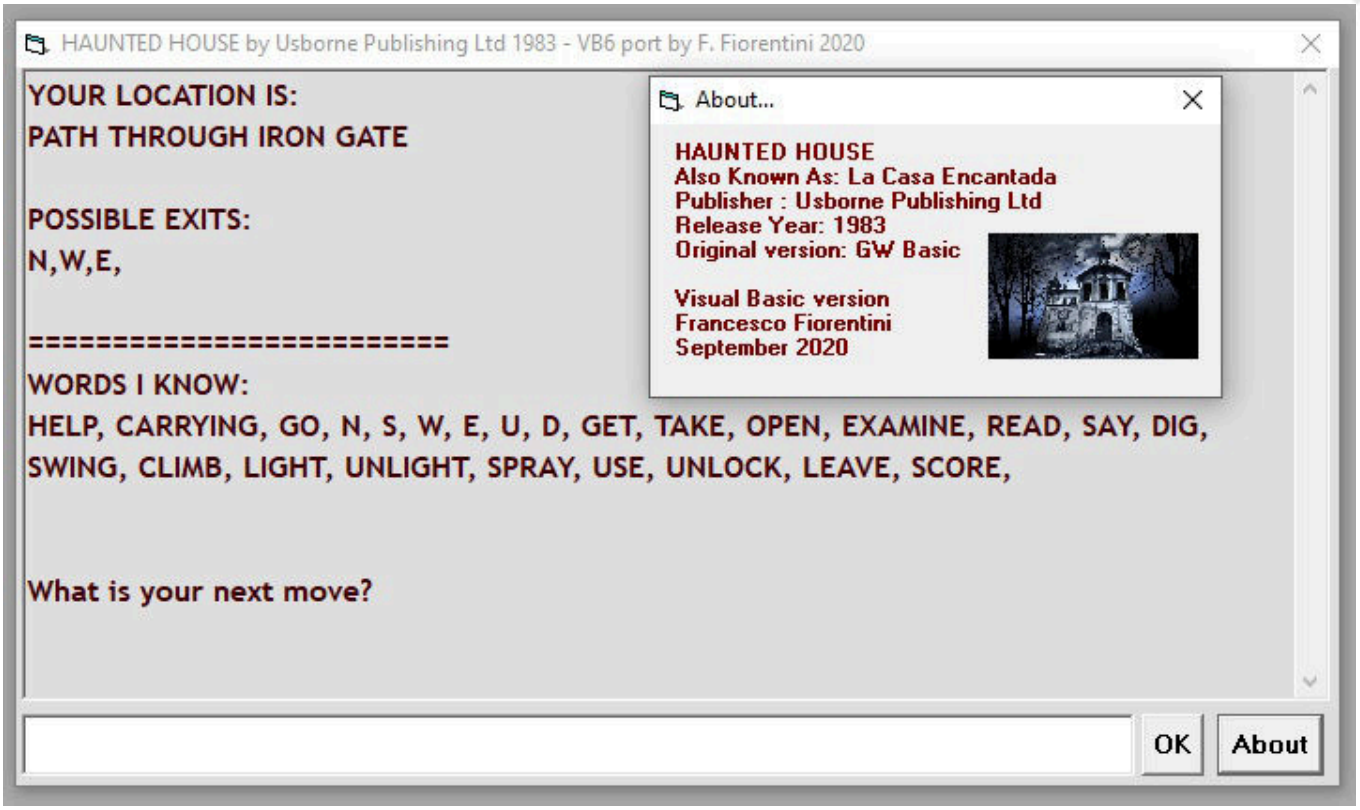


Figura 3. Haunted House porting su Visual Basic 6 con About... form

Dei 3 porting questo é stato decisamente il piú impegnativo. Adattare il codice GW Basic nato come programmazione 'destrutturata' ad un Basic 'event driven' come il Visual Basic 6, devo ammettere che non é stato uno scherzo. Mi sono voluto comunque cimentare in questa conversione per capire che livello di diffcoltá comportasse e se il risultato, una volta raggiunto, valesse o meno lo sforzo.

Ecco una lista delle azioni che ho dovuto intraprendere:
- ho dovuto creare una distinzione tra l'area descrittiva e l'area di digitazione dei comandi
- a causa della rimozione della numerazione delle righe, ho dovuto modificare tutti i salti condizionati (GOSUB) per puntare a delle label (per fare prima ho mantenuto il numero di riga come nome della label)
- il VB non gestisce READ/DATA, ho dovuto quindi trasformare i DATA in Array
- i vecchi Basic facevano distinzione tra i nomi delle variabili se queste contenevano l'indicazione del tipo; es. WA, WA$ e WA% erano tre variabili totalmente diverse. In questo caso ho dovuto rinominare tutte le variabili per consentire al programma di funzionare correttamente
- ho inoltre aggiunto un form 'About' per dare i dovuti crediti a Usborne Publishing

Il risultato é tutto sommato gradevole. L'avventura testuale rispecchia lo spirito dei giochi del passato, ma con un look and feel decisamente piú moderno.

Inoltre il VB6 offre diverse potenzialitá per arricchire il nostro software, e con relativamente poco sforzo. Magari, in una delle successive uscite di RetroMagazine potremmo anche pensare di rilanciare **Haunted House Enhanced**!

Potrebbe essere aggiunta di una lista per rappresentare l'inventario, oppure aggiunta di una serie di pulsanti per l'esecuzione dei comandi elementari (invece di doverli digitare di volta in volta).
Potrebbe anche essere aggiunta una mappa che si auto-completi man mano che il nostro eroe avanza nell'esplorazione. Per non parlare di trasformare un'avventura testuale, in un'avventura testuale con grafica, aggiungendo un'immagine per ogni luogo visitato.

Ovviamente queste sono solo delle idee che ho buttato giú al momento di scrivere l'articolo. Ma la base c'é ed implementarle, come ho scritto in precedenza, sarebbe relativamente semplice. Mi prudono giá le mani per mettermi all'opera... :-D

Spero che questo primo articolo sul porting dei programmi Basic possa incontare il vostro interesse e soprattutto possa essere di stimolo per riscoprire i vecchi programmi.

Tornando all'avventura Haunted House: se siete esperti avventurieri, risolverla sará ragionevolmente semplice, se invece avete bisogno di aiuto, qui potete trovare un comodo walkthrough che vi permetterá di completare il gioco: https://solutionarchive.com/file/id%2C9320/

Non mi resta quindi che augurarvi buon divertimento e perdonatemi per la lunghezza eccessiva del codice allegato, ma é, spero, per una buona causa. :-)

Listato originale per GW Basic (MS DOS) - compatibile anche con Locomotive Basic (Amstrad CPC)

```
10 REM HAUNTED HOUSE ADVENTURE
20 REM ***********************
30 REM THIS VERSION FOR "MICROSOFT" BASIC
40 REM REQUIRES A MINIMUM OF 16K
50 REM SELECT "TEXT MODE" IF NECESSARY
60 REM ***********************

70 V = 25: W = 36: G = 18
73 DIM R$(63), D$(63), O$(W), V$(V)
77 DIM C(W), L(G), F(W)
80 GOSUB 1600 '----> DO INITIALISATION

85 REM DESCRIPTION AND FEEDBACK
90 CLS : PRINT "HAUNTED HOUSE"
100 PRINT "-------------"
110 PRINT "YOUR LOCATION"
120 PRINT D$(RM)
130 PRINT "EXITS:";
140 FOR I = 1 TO LEN(R$(RM))
150 PRINT MID$(R$(RM), I, 1); ",";
160 NEXT I
170 PRINT
180 FOR I = 1 TO G
190 IF L(I) = RM AND F(I) = 0 THEN PRINT "YOU CAN SEE "; O$(I); " HERE"
200 NEXT I
210 PRINT "========================="
220 PRINT M$: M$ = "WHAT"
225 REM INPUT AND INPUT ANALYSIS
230 INPUT "WHAT WILL YOU DO NOW"; Q$
240 V$ = "": W$ = "": VB = 0: OB = 0
250 FOR I = 1 TO LEN(Q$)
260 IF MID$(Q$, I, 1) = " " AND V$ = "" THEN V$ = LEFT$(Q$, I - 1)
270 IF MID$(Q$, I + 1, 1) <> " " AND V$ <> "" THEN W$ = MID$(Q$, I + 1, LEN(Q$) - 1): I = LEN(Q$)
280 NEXT I
290 IF W$ = "" THEN V$ = Q$
300 FOR I = 1 TO V
310 IF V$ = V$(I) THEN VB = I
320 NEXT I
330 FOR I = 1 TO W
340 IF W$ = O$(I) THEN OB = I
350 NEXT I
355 REM ERROR MESSAGES OVERRIDE CONDITIONS
360 IF W$ > "" AND OB = 0 THEN M$ = "THAT'S SILLY"
370 IF VB = 0 THEN VB = V + 1
380 IF W$ = "" THEN M$ = "I NEED TWO WORDS"
390 IF VB > V AND OB > 0 THEN M$ = "YOU CAN'T '" + Q$ + "'"
400 IF VB > V AND OB = 0 THEN M$ = "YOU DON'T MAKE SENSE!"
410 IF VB < V AND OB > 0 AND C(OB) = 0 THEN M$ = "YOU DON'T HAVE '" + W$
420 IF F(26) = 1 AND RM = 13 AND FIX(RND(1) * 4) <> 3 AND VB <> 21 THEN M$ = "BATS ATTACKING!": GOTO 90
430 IF RM = 44 AND FIX(RND(1) * 3) = 1 AND F(24) <> 1 THEN F(27) = 1
440 IF F(0) = 1 THEN LL = LL - 1
450 IF LL < 1 THEN F(0) = 0
455 REM BRANCH TO SUBROUTINES
460 ON VB GOSUB 500, 570, 640, 640, 640, 640, 640, 640, 640, 980, 980, 1030, 1070, 1140, 1180, 1220,
1250, 1300, 1340, 1380, 1400, 1430, 1460, 1490, 1510, 1590
470 IF LL = 10 THEN M$ = "YOUR CANDLE IS WANING!"
480 IF LL = 1 THEN M$ = "YOUR CANDLE IS OUT!"
490 GOTO 90

495 REM VERB 1
500 PRINT "WORDS I KNOW:"
510 FOR I = 1 TO V
520 PRINT V$(I); ", ";
530 NEXT I
540 M$ = "": PRINT
550 GOSUB 1580
560 RETURN

565 REM VERB 2
570 PRINT "YOU ARE CARRYING:"
580 FOR I = 1 TO G
590 IF C(I) = 1 THEN PRINT O$(I); ", ";
600 NEXT I
610 M$ = "": PRINT
620 GOSUB 1580
630 RETURN

635 REM VERBS 3 TO 9 INCLUSIVE
640 D = 0
650 IF OB = 0 THEN D = VB - 3
660 IF OB = 19 THEN D = 1
```

```
670 IF OB = 20 THEN D = 2
680 IF OB = 21 THEN D = 3
690 IF OB = 22 THEN D = 4
700 IF OB = 23 THEN D = 5
710 IF OB = 24 THEN D = 6
720 IF RM = 20 AND D = 5 THEN D = 1
730 IF RM = 20 AND D = 6 THEN D = 3
740 IF RM = 22 AND D = 6 THEN D = 2
750 IF RM = 22 AND D = 5 THEN D = 3
760 IF RM = 36 AND D = 6 THEN D = 1
770 IF RM = 36 AND D = 5 THEN D = 2
780 IF F(14) = 1 THEN M$ = "CRASH!  YOU FELL OUT OF THE TREE!": F(14) = 0: RETURN
790 IF F(27) = 1 AND RM = 52 THEN M$ = "GHOSTS WILL NOT LET YOU MOVE": RETURN
800 IF RM = 45 AND C(1) = 1 AND F(34) = 0 THEN M$ = "A MAGICAL BARRIER TO THE WEST": RETURN
810 IF (RM = 26 AND F(0) = 0) AND (D = 1 OR D = 4) THEN M$ = "YOU NEED A LIGHT": RETURN
820 IF RM = 54 AND C(15) <> 1 THEN M$ = "YOU'RE STUCK!": RETURN
830 IF C(15) = 1 AND NOT (RM = 53 OR RM = 54 OR RM = 55 OR RM = 47) THEN M$ = "YOU CAN'T CARRY A BOAT!":
RETURN
840 IF (RM > 26 AND RM < 30) AND F(0) = 0 THEN M$ = "TOO DARK TO MOVE": RETURN
850 F(35) = 0: RL = LEN(R$(RM))
860 FOR I = 1 TO RL
870 U$ = MID$(R$(RM), I, 1)
880 IF (U$ = "N" AND D = 1 AND F(35) = 0) THEN RM = RM - 8: F(35) = 1
890 IF (U$ = "S" AND D = 2 AND F(35) = 0) THEN RM = RM + 8: F(35) = 1
900 IF (U$ = "W" AND D = 3 AND F(35) = 0) THEN RM = RM - 1: F(35) = 1
910 IF (U$ = "E" AND D = 4 AND F(35) = 0) THEN RM = RM + 1: F(35) = 1
920 NEXT I
930 M$ = "OK"
940 IF F(35) = 0 THEN M$ = "CAN'T GO THAT WAY!"
950 IF D < 1 THEN M$ = "GO WHERE?"
960 IF RM = 41 AND F(23) = 1 THEN R$ = "SW": M$ = "THE DOOR SLAMS SHUT!": F(23) = 0
970 RETURN

975 REM VERBS 10 AND 11
980 IF OB > G THEN M$ = "I CAN'T GET " + W$: RETURN
985 IF L(OB) <> RM THEN M$ = "IT ISN'T HERE"
990 IF F(OB) <> 0 THEN M$ = "WHAT " + W$ + "?"
1000 IF C(OB) = 1 THEN M$ = "YOU ALREADY HAVE IT"
1010 IF OB > 0 AND L(OB) = RM AND F(OB) = 0 THEN C(OB) = 1: L(OB) = 65: M$ = "YOU HAVE THE " + W$
1020 RETURN

1025 REM VERB 12
1030 IF RM = 43 AND (OB = 28 OR OB = 29) THEN F(17) = 0: M$ = "DRAWER OPEN"
1040 IF RM = 28 AND OB = 25 THEN M$ = "IT'S LOCKED"
1050 IF RM = 38 AND OB = 32 THEN M$ = "THAT'S CREEPY!": F(2) = 0
1060 RETURN

1065 REM VERB 13
1070 IF OB = 30 THEN F(18) = 0: M$ = "SOMETHING HERE!"
1080 IF OB = 31 THEN M$ = "THAT'S DISGUSTING!"
1090 IF (OB = 28 OR OB = 29) THEN M$ = "THERE'S A DRAWER"
1100 IF OB = 33 OR OB = 5 THEN GOSUB 1140
1110 IF RM = 43 AND OB = 35 THEN M$ = "THERE'S SOMETHING BEYOND..."
1120 IF OB = 32 THEN GOSUB 1030
1130 RETURN

1135 REM VERB 14
1140 IF RM = 42 AND OB = 33 THEN M$ = "THEY ARE DEMONIC WORKS"
1150 IF (OB = 3 OR OB = 36) AND C(3) = 1 AND F(34) = 0 THEN M$ = "USE THIS WORD WITH CARE 'XZANFAR'"
1160 IF C(5) = 1 AND OB = 5 THEN M$ = "THE SCRIPT IS IN AN ALIEN TONGUE"
1170 RETURN

1175 REM VERB 15
1180 M$ = "OK '" + W$ + "'"
1190 IF C(3) = 1 AND OB = 34 THEN M$ = "*MAGIC OCCURS*": IF RM <> 45 THEN RM = FIX(RND(1) * 64)
1200 IF C(3) = 1 AND OB = 34 AND RM = 45 THEN F(34) = 1
1210 RETURN

1215 REM VERB 16
1220 IF C(12) = 1 THEN M$ = "YOU MADE A HOLE"
1230 IF C(12) = 1 AND RM = 30 THEN M$ = "DUG THE BARS OUT": D$(RM) = "HOLE IN THE WALL": R$(RM) = "NSE"
1240 RETURN

1245 REM VERB 17
1250 IF C(14) <> 1 AND RM = 7 THEN M$ = "THIS IS NO TIME TO PLAY GAMES"
1260 IF OB = 14 AND C(14) = 1 THEN M$ = "YOU SWUNG IT"
1270 IF OB = 13 AND C(13) = 1 THEN M$ = "WHOOSH"
1280 IF OB = 13 AND C(13) = 1 AND RM = 43 THEN R$(RM) = "WN": D$(RM) = "STUDY WITH A SECRET ROOM": M$ =
"YOU BROKE THE THIN WALL"
1290 RETURN

1295 REM VERB 18
1300 IF OB = 14 AND C(14) = 1 THEN M$ = "IT ISN'T ATTACHED TO ANYTHING!"
1310 IF OB = 14 AND C(14) <> 1 AND RM = 7 AND F(14) = 0 THEN M$ = "YOU SEE THICK FORREST AND CLIFF
```

```
SOUTH": F(14) = 1: RETURN
1320 IF OB = 14 AND C(14) <> 1 AND RM = 7 AND F(14) = 1 THEN M$ = "GOING DOWN!": F(14) = 0
1330 RETURN

1335 REM VERB 19
1340 IF OB = 17 AND C(17) = 1 AND C(8) = 0 THEN M$ = "IT WILL BURN YOUR HANDS"
1350 IF OB = 17 AND C(17) = 1 AND C(9) = 0 THEN M$ = "NOTHING TO LIGHT IT WITH"
1360 IF OB = 17 AND C(17) = 1 AND C(9) = 1 AND C(8) = 1 THEN M$= "IT CASTS A FLICKERING LIGHT": F(0) = 1
1370 RETURN

1375 REM VERB 20
1380 IF F(0) = 1 THEN F(0) = 0: M$ = "EXTINGUISHED"
1390 RETURN

1395 REM VERB 21
1400 IF OB = 16 AND C(16) = 1 THEN M$ = "HISSSS"
1410 IF OB = 16 AND C(16) = 1 AND F(26) = 1 THEN F(26) = 0: M$ = "PFFT!  GOT THEM"
1420 RETURN

1425 REM VERB 22
1430 IF OB = 10 AND C(10) = 1 AND C(11) = 1 THEN M$ = "SWITCHED ON": F(24) = 1
1440 IF F(27) = 1 AND F(24) = 1 THEN M$ = "WHIZZ - VACUUMED THE GHOSTS UP!": F(27) = 0
1450 RETURN

1455 REM VERB 23
1460 IF RM = 43 AND (OB = 27 OR OB = 28) THEN GOSUB 1030
1470 IF RM = 28 AND OB = 25 AND F(25) = 0 AND C(18) = 1 THEN F(25) = 1: R$(RM) = "SEW": D$(RM) = "HUGE
OPEN DOOR": M$ = "THE KEY TURNS"
1480 RETURN

1485 REM VERB 24
1490 IF C(OB) = 1 THEN C(OB) = 0: L(OB) = RM: M$ = "DONE"
1500 RETURN

1505 REM VERB 25
1510 S = 0
1520 FOR I = 1 TO G
1530 IF C(I) = 1 THEN S = S + 1
1540 NEXT I
1550 IF S = 17 AND C(15) <> 1 AND RM <> 57 THEN PRINT "YOU HAVE EVERYTHING": PRINT "RETURN TO THE GATE
FOR FINAL SCORE"
1560 IF S = 17 AND RM = 57 THEN PRINT "DOUBLE SCORE FOR REACHING HERE!": S = S * 2
1570 PRINT "YOUR SCORE = "; S: IF S > 18 THEN PRINT "WELL DONE! YOU HAVE FINISHED THE GAME": END
1580 INPUT "PRESS RETURN TO CONTINUE"; Q$
1590 RETURN

------------------------------------------------------------------------------
1595 REM GAME INITIALISATION ROUTINE
1600 REM DIM R$(63), D$(63), O$(W), V$(V)
1610 REM DIM C(W), L(G), F(W)
1620 DATA 46,38,35,50,13,18,28,42,10,25,26,4,2,7,47,60,43,32
1630 FOR I = 1 TO G
1640 READ L(I)
1650 NEXT I
1660 DATA HELP,CARRYING?,GO,N,S,W,E,U,D,GET,TAKE,OPEN,EXAMINE,READ,SAY
1665 DATA DIG,SWING,CLIMB,LIGHT,UNLIGHT,SPRAY,USE,UNLOCK,LEAVE,SCORE
1680 FOR I = 1 TO V
1690 READ V$(I)
1700 NEXT I

1705 '----> Possible movements (they are associated with the location, sharing the same index)
1710 DATA SE,WE,WE,SWE,WE,WE,SWE,WS
1720 DATA NS,SE,WE,NW,SE,W,NE,NSW
1730 DATA NS,NS,SE,WE,NWUD,SE,WSUD,NS
1740 DATA N,NS,NSE,WE,WE,NSW,NS,NS
1750 DATA S,NSE,NSW,S,NSUD,N,N,NS
1760 DATA NE,NW,NE,W,NSE,WE,W,NS
1770 DATA SE,NSW,E,WE,NW,S,SW,NW
1780 DATA NE,NWE,WE,WE,WE,NWE,NWE,W
1790 FOR I = 0 TO 63
1800 READ R$(I)
1810 NEXT I

1815 '----> Description of the location
1820 DATA DARK CORNER,OVERGROWN GARDEN,BY LARGE WOODPILE,YARD BY RUBBISH
1830 DATA WEEDPATCH,FOREST,THICK FOREST,BLASTED TREE
1840 DATA CORNER OF HOUSE,ENTRANCE TO KITCHEN,KITCHEN AND GRIMEY COOKER,SCULLERY DOOR
1845 DATA ROOM WITH INCHES OF DUST,REAR TURRET ROOM,CLEARING BY HOUSE,PATH
1860 DATA SIDE OF HOUSE,BACK OF HALLWAY,DARK ALCOVE,SMALL DARK ROOM
1865 DATA BOTTOM OF SPIRAL STAIRCASE,WIDE PASSAGE,SLIPPERY STEPS,CLIFFTOP
1880 DATA NEAR CRUMBLING WALL,GLOOMY PASSAGE,POOL OF LIGHT,IMPRESSIVE VAULTED HALLWAY
1885 DATA HALL BY THICK WOODEN DOOR,TROPHY ROOM,CELLAR WITH BARRED WINDOW,CLIFF PATH
1900 DATA CUPBOARD WITH HANGING COAT,FRONT HALL,SITTING ROOM,SECRET ROOM
```

```
1905 DATA STEEP MARBLE STAIRS,DINING ROOM,DEEP CELLAR WITH COFFIN,CLIFF PATH
1920 DATA CLOSET,FRONT LOBBY,LIBRARY OF EVIL BOOKS,STUDY WITH DESK AND HOLE IN WALL
1925 DATA WEIRD COBWEBBY ROOM,VERY COLD CHAMBER,SPOOKY ROOM,CLIFF PATH BY MARSH
1940 DATA RUBBLE-STREWN VERANDAH,FRONT PORCH,FRONT TOWER,SLOPING CORRIDOR
1945 DATA UPPER GALLERY,MARSH BY WALL,MARSH,SOGGY PATH
1960 DATA BY TWISTED RAILING, PATH THROUGH IRON GATE,BY RAILINGS,BENEATH FRONT TOWER
1965 DATA DEBRIS FROM CRUMBLING FACADE,LARGE FALLEN BRICKWORK,ROTTING STONE ARCH,CRUMBLING CLIFFTOP
1980 FOR I = 0 TO 63
1990 READ D$(I)
2000 NEXT I

2010 DATA PAINTING,RING,MAGIC SPELLS,GOBLET,SCROLL,COINS,STATUE,CANDLESTICK
2012 DATA MATCHES,VACUUM,BATTERIES,SHOVEL,AXE,ROPE,BOAT,AEROSOL,CANDLE,KEY
2014 DATA NORTH,SOUTH,WEST,EAST,UP,DOWN
2016 DATA DOOR,BATS,GHOSTS,DRAWER,DESK,COAT,RUBBISH
2018 DATA COFFIN,BOOKS,XZANFAR,WALL,SPELLS
2060 FOR I = 1 TO W
2070 READ O$(I)
2080 NEXT I
2090 F(18) = 1: F(17) = 1: F(2) = 1: F(26) = 1: F(28) = 1: F(23) = 1: LL = 60: RM = 57: M$ = "OK"
2100 RETURN
```

Listato per Commodore 64 - adattamento di F. Fiorentini

```
10 rem haunted house adventure
20 rem ***********************
30 rem this version for Commodore 64
40 rem requires a minimum of 16k
50 rem F.Fiorentini - Ottobre 2020
60 rem ***********************

70 v = 25: w = 36: g = 18
73 dim r$(63), d$(63), o$(w), v$(v)
77 dim c(w), l(g), f(w)
80 gosub 1600 '----> do initialisation

85 rem description and feedback
90 print chr$(147): print "haunted house"
100 print "-------------"
110 print "your location"
120 print d$(rm)
130 print "exits:";
140 for i = 1 to len(r$(rm))
150 print mid$(r$(rm), i, 1); ",";
160 next i
170 print
180 for i = 1 to g
190 if l(i) = rm and f(i) = 0 then print "you can see "; o$(i); " here"
200 next i
210 print "========================="
220 print m$: m$ = "what"
225 rem input and input analysis
230 input "what will you do now"; q$
240 v$ = "": w$ = "": vb = 0: ob = 0
250 for i = 1 to len(q$)
260 if mid$(q$, i, 1) = " " and v$ = "" then v$ = left$(q$, i - 1)
270 if mid$(q$, i + 1, 1) <> " " and v$ <> "" then goto 275
271 goto 280
275 w$ = mid$(q$, i + 1, len(q$) - 1): i = len(q$)
280 next i
290 if w$ = "" then v$ = q$
300 for i = 1 to v
310 if v$ = v$(i) then vb = i
320 next i
330 for i = 1 to w
340 if w$ = o$(i) then ob = i
350 next i
355 rem error messages override conditions
360 if w$ > "" and ob = 0 then m$ = "that's silly"
370 if vb = 0 then vb = v + 1
380 if w$ = "" then m$ = "i need two words"
390 if vb > v and ob > 0 then m$ = "you can't '" + q$ + "'"
400 if vb > v and ob = 0 then m$ = "you don't make sense!"
410 if vb < v and ob > 0 and c(ob) = 0 then m$ = "you don't have '" + w$
420 if f(26)=1 and rm=13 and fix(rnd(1)*4) <> 3 and vb <> 21 then goto 425
421 goto 430
425 m$ = "bats attacking!": goto 90
430 if rm = 44 and fix(rnd(1) * 3) = 1 and f(24) <> 1 then f(27) = 1
440 if f(0) = 1 then ll = ll - 1
450 if ll < 1 then f(0) = 0
455 rem branch to subroutines
456 if vb > 14 then goto 465
460 on vb gosub 500,570,640,640,640,640,640,640,640,980,980,1030,1070,1140
```

```
461 goto 470
465 on vb-14 gosub 1180,1220,1250,1300,1340,1380,1400,1430,1460,1490,1510,1590
470 if ll = 10 then m$ = "your candle is waning!"
480 if ll = 1 then m$ = "your candle is out!"
490 goto 90

495 rem verb 1
500 print "words i know:"
510 for i = 1 to v
520 print v$(i); ", ";
530 next i
540 m$ = "": print
550 gosub 1580
560 return

565 rem verb 2
570 print "you are carrying:"
580 for i = 1 to g
590 if c(i) = 1 then print o$(i); ", ";
600 next i
610 m$ = "": print
620 gosub 1580
630 return

635 rem verbs 3 to 9 inclusive
640 d = 0
650 if ob = 0 then d = vb - 3
660 if ob = 19 then d = 1
670 if ob = 20 then d = 2
680 if ob = 21 then d = 3
690 if ob = 22 then d = 4
700 if ob = 23 then d = 5
710 if ob = 24 then d = 6
720 if rm = 20 and d = 5 then d = 1
730 if rm = 20 and d = 6 then d = 3
740 if rm = 22 and d = 6 then d = 2
750 if rm = 22 and d = 5 then d = 3
760 if rm = 36 and d = 6 then d = 1
770 if rm = 36 and d = 5 then d = 2
780 if f(14)=1 then m$="crash! you fell out of the tree!":f(14)=0:return
790 if f(27)=1 and rm=52 then m$="ghosts will not let you move":return
800 if rm=45andc(1)=1andf(34)=0 then m$="a magical barrier to the west":return
810 if (rm=26 andf(0)=0) and (d=1 or d=4) then m$="you need a light":return
820 if rm = 54 and c(15) <> 1 then m$ = "you're stuck!": return
830 if c(15)=1 and not(rm=53 or rm=54 or rm=55 or rm=47) then goto 835
831 goto 840
835 m$="you can't carry a boat!": return
840 if (rm > 26 and rm < 30) and f(0) = 0 then m$="too dark to move":return
850 f(35) = 0: rl = len(r$(rm))
860 for i = 1 to rl
870 u$ = mid$(r$(rm), i, 1)
880 if (u$ = "n" and d = 1 and f(35) = 0) then rm = rm - 8: f(35) = 1
890 if (u$ = "s" and d = 2 and f(35) = 0) then rm = rm + 8: f(35) = 1
900 if (u$ = "w" and d = 3 and f(35) = 0) then rm = rm - 1: f(35) = 1
910 if (u$ = "e" and d = 4 and f(35) = 0) then rm = rm + 1: f(35) = 1
920 next i
930 m$ = "ok"
940 if f(35) = 0 then m$ = "can't go that way!"
950 if d < 1 then m$ = "go where?"
960 if rm = 41 and f(23) = 1 then goto 965
961 goto 970
965 r$ = "sw": m$ = "the door slams shut!": f(23) = 0
970 return

975 rem verbs 10 and 11
980 if ob > g then m$ = "i can't get " + w$: return
985 if l(ob) <> rm then m$ = "it isn't here"
990 if f(ob) <> 0 then m$ = "what " + w$ + "?"
1000 if c(ob) = 1 then m$ = "you already have it"
1010 if ob>0 and l(ob)=rm and f(ob)=0 then goto 1015
1011 goto 1020
1015 c(ob)=1:l(ob) = 65: m$ = "you have the " + w$
1020 return

1025 rem verb 12
1030 if rm = 43 and (ob = 28 or ob = 29) then goto 1035
1031 goto 1040
1035 f(17) = 0: m$ = "drawer open"
1040 if rm = 28 and ob = 25 then m$ = "it's locked"
1050 if rm = 38 and ob = 32 then m$ = "that's creepy!": f(2) = 0
1060 return

1065 rem verb 13
```

```
1070 if ob = 30 then f(18) = 0: m$ = "something here!"
1080 if ob = 31 then m$ = "that's disgusting!"
1090 if (ob = 28 or ob = 29) then m$ = "there's a drawer"
1100 if ob = 33 or ob = 5 then gosub 1140
1110 if rm = 43 and ob = 35 then m$ = "there's something beyond..."
1120 if ob = 32 then gosub 1030
1130 return

1135 rem verb 14
1140 if rm = 42 and ob = 33 then m$ = "they are demonic works"
1150 if (ob=3 or ob=36) and c(3)=1 and f(34)=0 then goto 1155
1151 goto 1160
1155 m$="use this word with care 'xzanfar'"
1160 if c(5) = 1 and ob = 5 then m$ = "the script is in an alien tongue"
1170 return

1175 rem verb 15
1180 m$ = "ok '" + w$ + "'"
1190 if c(3)=1 and ob=34 then goto 1195
1191 goto 1200
1195 m$="*magic occurs*": if rm<>45 then rm=fix(rnd(1)*64)
1200 if c(3) = 1 and ob = 34 and rm = 45 then f(34) = 1
1210 return

1215 rem verb 16
1220 if c(12) = 1 then m$ = "you made a hole"
1230 if c(12) = 1 and rm = 30 then goto 1235
1231 goto 1240
1235 m$ = "dug the bars out": d$(rm) = "hole in the wall": r$(rm) = "nse"
1240 return

1245 rem verb 17
1250 if c(14) <> 1 and rm = 7 then m$ = "this is no time to play games"
1260 if ob = 14 and c(14) = 1 then m$ = "you swung it"
1270 if ob = 13 and c(13) = 1 then m$ = "whoosh"
1280 if ob=13 and c(13)=1 and rm=43 then goto 1285
1281 goto 1290
1285 r$(rm)="wn":d$(rm)="study with a secret room":m$="you broke the thin wall"
1290 return

1295 rem verb 18
1300 if ob=14 and c(14)=1 then m$="it isn't attached to anything!"
1310 if ob=14 and c(14)<>1 and rm=7 and f(14)=0 then goto 1315
1311 goto 1320
1315 m$="you see thick forrest and cliff south": f(14)=1: return
1320 if ob=14 and c(14)<>1 and rm=7 and f(14)=1 then goto 1325
1321 goto 1330
1325 m$ = "going down!": f(14) = 0
1330 return

1335 rem verb 19
1340 if ob=17 and c(17)=1 and c(8)=0 then m$="it will burn your hands"
1350 if ob=17 and c(17)=1 and c(9)=0 then m$="nothing to light it with"
1360 if ob=17 and c(17)=1 and c(9)=1 and c(8)=1 then goto 1365
1361 goto 1370
1365 m$="it casts a flickering light": f(0)=1
1370 return

1375 rem verb 20
1380 if f(0) = 1 then f(0) = 0: m$ = "extinguished"
1390 return

1395 rem verb 21
1400 if ob = 16 and c(16) = 1 then m$ = "hissss"
1410 if ob=16 and c(16)=1 and f(26)=1 then f(26)=0: m$="pfft!  got them"
1420 return

1425 rem verb 22
1430 if ob = 10 and c(10) = 1 and c(11) = 1 then goto 1435
1431 goto 1440
1435 m$ = "switched on": f(24) = 1
1440 if f(27) = 1 and f(24) = 1 then 1445
1441 goto 1450
1445 m$ = "whizz - vacuumed the ghosts up!": f(27) = 0
1450 return

1455 rem verb 23
1460 if rm = 43 and (ob = 27 or ob = 28) then gosub 1030
1470 if rm=28 and ob=25 and f(25)=0 and c(18)=1 then goto 1475
1471 goto 1480
1475 f(25)=1: r$(rm)="sew": d$(rm)="huge open door": m$="the key turns"
1480 return
```

```
1485 rem verb 24
1490 if c(ob) = 1 then c(ob) = 0: l(ob) = rm: m$ = "done"
1500 return

1505 rem verb 25
1510 s = 0
1520 for i = 1 to g
1530 if c(i) = 1 then s = s + 1
1540 next i
1550 if s=17 and c(15)<>1 and rm<>57 then goto 1555
1551 goto 1560
1555 print "you have everything": print "return to the gate for final score"
1560 if s = 17 and rm = 57 then goto 1565
1561 goto 1570
1565 print "double score for reaching here!": s = s * 2
1570 print "your score = "; s: if s > 18 then goto 1575
1571 goto 1580
1575 print "well done! you have finished the game": end
1580 input "press return to continue"; q$
1590 return

1595 rem game initialisation routine
1600 rem dim r$(63), d$(63), o$(w), v$(v)
1610 rem dim c(w), l(g), f(w)
1620 data 46,38,35,50,13,18,28,42,10,25,26,4,2,7,47,60,43,32
1630 for i = 1 to g
1640 read l(i)
1650 next i
1660 data help,carrying?,go,n,s,w,e,u,d,get,take,open,examine,read,say
1665 data dig,swing,climb,light,unlight,spray,use,unlock,leave,score
1680 for i = 1 to v
1690 read v$(i)
1700 next i

1705 rem possible movements (associated with the location, same index)
1710 data se,we,we,swe,we,we,swe,ws
1720 data ns,se,we,nw,se,w,ne,nsw
1730 data ns,ns,se,we,nwud,se,wsud,ns
1740 data n,ns,nse,we,we,nsw,ns,ns
1750 data s,nse,nsw,s,nsud,n,n,ns
1760 data ne,nw,ne,w,nse,we,w,ns
1770 data se,nsw,e,we,nw,s,sw,nw
1780 data ne,nwe,we,we,we,nwe,nwe,w
1790 for i = 0 to 63
1800 read r$(i)
1810 next i

1815 rem description of the location
1820 data dark corner,overgrown garden,by large woodpile,yard by rubbish
1830 data weedpatch,forest,thick forest,blasted tree, corner of house
1840 data entrance to kitchen,kitchen and grimey cooker,scullery door
1845 data room with inches of dust,rear turret room,clearing by house,path
1860 data side of house,back of hallway,dark alcove,small dark room
1865 data bottom of spiral staircase,wide passage,slippery steps,clifftop
1880 data near crumbling wall,gloomy passage,pool of light
1881 data impressive vaulted hallway, hall by thick wooden door
1885 data trophy room,cellar with barred window,cliff path
1900 data cupboard with hanging coat,front hall,sitting room,secret room,closet
1905 data steep marble stairs,dining room,deep cellar with coffin,cliff path
1920 data front lobby,library of evil books,study with desk and hole in wall
1925 data weird cobwebby room,very cold chamber,spooky room,cliff path by marsh
1940 data rubble-strewn verandah,front porch,front tower,sloping corridor
1945 data upper gallery,marsh by wall,marsh,soggy path,by twisted railing
1960 data path through iron gate,by railings,beneath front tower
1965 data debris from crumbling facade,large fallen brickwork
1966 data rotting stone arch,crumbling clifftop
1980 for i = 0 to 63
1990 read d$(i)
2000 next i

2010 data painting,ring,magic spells,goblet,scroll,coins,statue,candlestick
2012 data matches,vacuum,batteries,shovel,axe,rope,boat,aerosol,candle,key
2014 data north,south,west,east,up,down
2016 data door,bats,ghosts,drawer,desk,coat,rubbish
2018 data coffin,books,xzanfar,wall,spells
2060 for i = 1 to w
2070 read o$(i)
2080 next i
2090 f(18)=1:f(17)=1:f(2)=1:f(26)=1:f(28)=1:f(23)=1:ll=60:rm=57:m$="ok"
2100 return
```

Listato per Visual Basic 6.0 - adattamento di F. Fiorentini

```
VERSION 5.00
Begin VB.Form About
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "About..."
   ClientHeight    =   2175
   ClientLeft      =   45
   ClientTop       =   390
   ClientWidth     =   4560
   LinkTopic       =   "Form2"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   2175
   ScaleWidth      =   4560
   ShowInTaskbar   =   0   'False
   StartUpPosition =   1  'CenterOwner
   Begin VB.Image Image1
      Height          =   1000
      Left            =   2700
      Picture         =   "About.frx":0000
      Stretch         =   -1  'True
      Top             =   870
      Width           =   1680
   End
   Begin VB.Label Label1
      Caption         =   "Label1"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   8.25
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00000080&
      Height          =   1935
      Left            =   200
      TabIndex        =   0
      Top             =   120
      Width           =   4095
   End
End
Attribute VB_Name = "About"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Form_Load()
Label1.Caption = "HAUNTED HOUSE" & vbCrLf & "Also Known As: La Casa Encantada" & vbCrLf & "Publisher :
Usborne Publishing Ltd"
Label1.Caption = Label1.Caption & vbCrLf & "Release Year: 1983" & vbCrLf & "Original version: GW Basic"
& vbCrLf & vbCrLf & "Visual Basic version"
Label1.Caption = Label1.Caption & vbCrLf & "Francesco Fiorentini" & vbCrLf & "September 2020"
End Sub

VERSION 5.00
Begin VB.Form HauntedHouse
   BackColor       =   &H00E0E0E0&
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "HAUNTED HOUSE by Usborne Publishing Ltd 1983 - VB6 port by F. Fiorentini 2020"
   ClientHeight    =   5730
   ClientLeft      =   8295
   ClientTop       =   5670
   ClientWidth     =   10575
   ForeColor       =   &H00000040&
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   5730
   ScaleWidth      =   10575
   ShowInTaskbar   =   0   'False
   Begin VB.CommandButton Command1
      Caption         =   "About"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
```

```
      EndProperty
      Height          =   495
      Left            =   9600
      TabIndex        =   3
      Top             =   5160
      Width           =   855
   End
   Begin VB.CommandButton Command_Check
      Caption         =   "OK"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   9000
      TabIndex        =   2
      Top             =   5160
      Width           =   495
   End
   Begin VB.TextBox InputString
      BeginProperty Font
         Name            =   "Verdana"
         Size            =   14.25
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   50
      TabIndex        =   1
      Top             =   5160
      Width           =   8895
   End
   Begin VB.TextBox Testo
      BackColor       =   &H00E0E0E0&
      BeginProperty Font
         Name            =   "Trebuchet MS"
         Size            =   12
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00000040&
      Height          =   5055
      Left            =   50
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   2  'Vertical
      TabIndex        =   0
      Text            =   "Haunted House.frx":0000
      Top             =   0
      Width           =   10455
   End
End
Attribute VB_Name = "HauntedHouse"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Dim V, W, G, I, VB, OB As Integer
Dim R, D, O, S
Dim Varray, RArray, DArray, OArray, LArray
Dim M$
Dim V_$
Dim W_$
Dim R_$
Dim U$
Dim Q$
Dim RM, LL
Dim FArray
Dim C(36)
Dim F(36)
```

```
Dim RL
Dim AddString As String

Private Sub Command_Check_Click()
Q$ = UCase(InputString)
M$ = ""

'---------- Controllo dell'azione
V_$ = "": W_$ = "": VB = 0: OB = 0
For I = 1 To Len(Q$)
    If Mid$(Q$, I, 1) = " " And V_$ = "" Then V_$ = Left$(Q$, I - 1)
    If Mid$(Q$, I + 1, 1) <> " " And V_$ <> "" Then W_$ = Mid$(Q$, I + 1, Len(Q$) - 1): I = Len(Q$)
Next I
If W_$ = "" Then V_$ = Q$
For I = 1 To V
    If V_$ = Varray(I) Then VB = I
Next I
For I = 1 To W
    If W_$ = OArray(I) Then OB = I
Next I

Rem ERROR MESSAGES OVERRIDE CONDITIONS
If W_$ > "" And OB = 0 Then M$ = "THAT'S SILLY"
If VB = 0 Then VB = V + 1
If W_$ = "" Then M$ = "I NEED TWO WORDS"
If VB > V And OB > 0 Then M$ = "YOU CAN'T '" + Q$ + "'"
If VB > V And OB = 0 Then M$ = "YOU DON'T MAKE SENSE!"
If VB < V And OB > 0 And C(OB) = 0 Then M$ = "YOU DON'T HAVE '" + W_$
If FArray(26) = 1 And RM = 13 And Fix(Rnd(1) * 4) <> 3 And VB <> 21 Then M$ = "BATS ATTACKING!": GoTo
Stampa
If RM = 44 And Fix(Rnd(1) * 3) = 1 And FArray(24) <> 1 Then FArray(27) = 1
If FArray(0) = 1 Then LL = LL - 1
If LL < 1 Then FArray(0) = 0

Rem BRANCH TO SUBROUTINES
On VB GoSub 500, 570, 640, 640, 640, 640, 640, 640, 640, 980, 980, 1030, 1070, 1140, 1180, 1220, 1250,
1300, 1340, 1380, 1400, 1430, 1460, 1490, 1510, 1590
If LL = 10 Then M$ = "YOUR CANDLE IS WANING!"
If LL = 1 Then M$ = "YOUR CANDLE IS OUT!"
GoTo Stampa

500:
Rem VERB 1
AddString = "WORDS I KNOW:" & vbCrLf
For I = 1 To V
AddString = AddString & Varray(I) & ", "
Next I
M$ = ""
AddString = AddString & vbCrLf
Return

570:
Rem VERB 2
AddString = "YOU ARE CARRYING:" & vbCrLf
For I = 1 To G
If C(I) = 1 Then AddString = AddString & OArray(I) & ", "
Next I
AddString = AddString & vbCrLf
M$ = ""
Return

640:
Rem VERBS 3 TO 9 INCLUSIVE
D = 0
If OB = 0 Then D = VB - 3
If OB = 19 Then D = 1
If OB = 20 Then D = 2
If OB = 21 Then D = 3
If OB = 22 Then D = 4
If OB = 23 Then D = 5
If OB = 24 Then D = 6
If RM = 20 And D = 5 Then D = 1
If RM = 20 And D = 6 Then D = 3
If RM = 22 And D = 6 Then D = 2
If RM = 22 And D = 5 Then D = 3
If RM = 36 And D = 6 Then D = 1
If RM = 36 And D = 5 Then D = 2
If FArray(14) = 1 Then M$ = "CRASH!  YOU FELL OUT OF THE TREE!": FArray(14) = 0: Return
If FArray(27) = 1 And RM = 52 Then M$ = "GHOSTS WILL NOT LET YOU MOVE": Return
If RM = 45 And C(1) = 1 And FArray(34) = 0 Then M$ = "A MAGICAL BARRIER TO THE WEST": Return
If (RM = 26 And FArray(0) = 0) And (D = 1 Or D = 4) Then M$ = "YOU NEED A LIGHT": Return
If RM = 54 And C(15) <> 1 Then M$ = "YOU'RE STUCK!": Return
If C(15) = 1 And Not (RM = 53 Or RM = 54 Or RM = 55 Or RM = 47) Then M$ = "YOU CAN'T CARRY A BOAT!":
```

```
Return
If (RM > 26 And RM < 30) And FArray(0) = 0 Then M$ = "TOO DARK TO MOVE": Return
FArray(35) = 0: RL = Len(RArray(RM))
For I = 1 To RL
U$ = Mid$(RArray(RM), I, 1)
If (U$ = "N" And D = 1 And FArray(35) = 0) Then RM = RM - 8: FArray(35) = 1
If (U$ = "S" And D = 2 And FArray(35) = 0) Then RM = RM + 8: FArray(35) = 1
If (U$ = "W" And D = 3 And FArray(35) = 0) Then RM = RM - 1: FArray(35) = 1
If (U$ = "E" And D = 4 And FArray(35) = 0) Then RM = RM + 1: FArray(35) = 1
Next I
M$ = "OK"
If FArray(35) = 0 Then M$ = "CAN'T GO THAT WAY!"
If D < 1 Then M$ = "GO WHERE?"
If RM = 41 And FArray(23) = 1 Then R_$ = "SW": M$ = "THE DOOR SLAMS SHUT!": FArray(23) = 0
Return

980:
Rem VERBS 10 AND 11
If OB > G Then M$ = "I CAN'T GET " + W_$: Return
If LArray(OB) <> RM Then M$ = "IT ISN'T HERE"
If FArray(OB) <> 0 Then M$ = "WHAT " + W_$ + "?"
If C(OB) = 1 Then M$ = "YOU ALREADY HAVE IT"
If OB > 0 And LArray(OB) = RM And FArray(OB) = 0 Then C(OB) = 1: LArray(OB) = 65: M$ = "YOU HAVE THE " +
W_$
Return

1030:
Rem VERB 12
If RM = 43 And (OB = 28 Or OB = 29) Then FArray(17) = 0: M$ = "DRAWER OPEN"
If RM = 28 And OB = 25 Then M$ = "IT'S LOCKED"
If RM = 38 And OB = 32 Then M$ = "THAT'S CREEPY!": FArray(2) = 0
Return

1070:
Rem VERB 13
If OB = 30 Then FArray(18) = 0: M$ = "SOMETHING HERE!"
If OB = 31 Then M$ = "THAT'S DISGUSTING!"
If (OB = 28 Or OB = 29) Then M$ = "THERE'S A DRAWER"
If OB = 33 Or OB = 5 Then GoSub 1140
If RM = 43 And OB = 35 Then M$ = "THERE'S SOMETHING BEYOND..."
If OB = 32 Then GoSub 1030
If OB > 0 And M$ = "" Then M$ = "NOTHING REALLY USEFUL FOUND..."
Return

1140:
Rem VERB 14
If RM = 42 And OB = 33 Then M$ = "THEY ARE DEMONIC WORKS"
If (OB = 3 Or OB = 36) And C(3) = 1 And FArray(34) = 0 Then M$ = "USE THIS WORD WITH CARE 'XZANFAR'"
If C(5) = 1 And OB = 5 Then M$ = "THE SCRIPT IS IN AN ALIEN TONGUE"
Return

1180:
Rem VERB 15
M$ = "OK '" + W_$ + "'"
If C(3) = 1 And OB = 34 Then M$ = "*MAGIC OCCURS*": If RM <> 45 Then RM = Fix(Rnd(1) * 64)
If C(3) = 1 And OB = 34 And RM = 45 Then FArray(34) = 1
Return

1220:
Rem VERB 16
If C(12) = 1 Then M$ = "YOU MADE A HOLE"
If C(12) = 1 And RM = 30 Then M$ = "DUG THE BARS OUT": DArray(RM) = "HOLE IN THE WALL": RArray(RM) =
"NSE"
Return

1250:
Rem VERB 17
If C(14) <> 1 And RM = 7 Then M$ = "THIS IS NO TIME TO PLAY GAMES"
If OB = 14 And C(14) = 1 Then M$ = "YOU SWUNG IT"
If OB = 13 And C(13) = 1 Then M$ = "WHOOSH"
If OB = 13 And C(13) = 1 And RM = 43 Then RArray(RM) = "WN": DArray(RM) = "STUDY WITH A SECRET ROOM": M$
= "YOU BROKE THE THIN WALL"
Return

1300:
Rem VERB 18
If OB = 14 And C(14) = 1 Then M$ = "IT ISN'T ATTACHED TO ANYTHING!"
If OB = 14 And C(14) <> 1 And RM = 7 And FArray(14) = 0 Then M$ = "YOU SEE THICK FORREST AND CLIFF
SOUTH": FArray(14) = 1: Return
If OB = 14 And C(14) <> 1 And RM = 7 And FArray(14) = 1 Then M$ = "GOING DOWN!": FArray(14) = 0
Return

1340:
```

```
Rem VERB 19
If OB = 17 And C(17) = 1 And C(8) = 0 Then M$ = "IT WILL BURN YOUR HANDS"
If OB = 17 And C(17) = 1 And C(9) = 0 Then M$ = "NOTHING TO LIGHT IT WITH"
If OB = 17 And C(17) = 1 And C(9) = 1 And C(8) = 1 Then M$ = "IT CASTS A FLICKERING LIGHT": FArray(0) =
1
Return

1380:
Rem VERB 20
If FArray(0) = 1 Then FArray(0) = 0: M$ = "EXTINGUISHED"
Return

1400:
Rem VERB 21
If OB = 16 And C(16) = 1 Then M$ = "HISSSS"
If OB = 16 And C(16) = 1 And FArray(26) = 1 Then FArray(26) = 0: M$ = "PFFT!  GOT THEM"
Return

1430:
Rem VERB 22
If OB = 10 And C(10) = 1 And C(11) = 1 Then M$ = "SWITCHED ON": FArray(24) = 1
If FArray(27) = 1 And FArray(24) = 1 Then M$ = "WHIZZ - VACUUMED THE GHOSTS UP!": FArray(27) = 0
Return

1460:
Rem VERB 23
If RM = 43 And (OB = 27 Or OB = 28) Then GoSub 1030
If RM = 28 And OB = 25 And FArray(25) = 0 And C(18) = 1 Then FArray(25) = 1: RArray(RM) = "SEW":
DArray(RM) = "HUGE OPEN DOOR": M$ = "THE KEY TURNS"
Return

1490:
Rem VERB 24
If C(OB) = 1 Then C(OB) = 0: LArray(OB) = RM: M$ = "DONE"
Return

1510:
Rem VERB 25
S = 0
For I = 1 To G
If C(I) = 1 Then S = S + 1
Next I
If S = 17 And C(15) <> 1 And RM <> 57 Then AddString = AddString & vbCrLf & "YOU HAVE EVERYTHING":
AddString = AddString & vbCrLf & "RETURN TO THE GATE FOR FINAL SCORE"
If S = 17 And RM = 57 Then AddString = AddString & vbCrLf & "DOUBLE SCORE FOR REACHING HERE!": S = S * 2
AddString = AddString & vbCrLf & "YOUR SCORE = " & S: If S > 18 Then AddString = AddString & vbCrLf &
"WELL DONE! YOU HAVE FINISHED THE GAME": End
Return

1590:
Return

Stampa:
'----------- Stampa risultato del comando -----------------
Testo.Text = "YOUR LOCATION IS:"
Testo.Text = Testo.Text & vbCrLf & DArray(RM)
Testo.Text = Testo.Text & vbCrLf & vbCrLf & "POSSIBLE EXITS:" & vbCrLf
For I = 1 To Len(RArray(RM))
    Testo.Text = Testo.Text & Mid$(RArray(RM), I, 1) & ","
Next I
Testo.Text = Testo.Text & vbCrLf
For I = 1 To G
    If LArray(I) = RM And FArray(I) = 0 Then Testo.Text = Testo.Text & "YOU CAN SEE " & OArray(I) & "
HERE"
Next I
Testo.Text = Testo.Text & vbCrLf & "========================="
If AddString <> "" Then
    Testo.Text = Testo.Text & vbCrLf & AddString
    AddString = ""
Else
    Testo.Text = Testo.Text & vbCrLf & M$
End If
Testo.Text = Testo.Text & vbCrLf & vbCrLf & "What is your next move?"
InputString.Text = ""
InputString.SetFocus
End Sub

Private Sub Command1_Click()
About.Show
End Sub

Private Sub Form_Load()
'Load parameters
```

```
V = 25: W = 36: G = 18
LArray = Array(0, 46, 38, 35, 50, 13, 18, 28, 42, 10, 25, 26, 4, 2, 7, 47, 60, 43, 32)
Varray = Array("", "HELP", "CARRYING", "GO", "N", "S", "W", "E", "U", "D", "GET", "TAKE", "OPEN",
"EXAMINE", "READ", "SAY", "DIG", "SWING", "CLIMB", "LIGHT", "UNLIGHT", "SPRAY", "USE", "UNLOCK",
"LEAVE", "SCORE")
RArray = Array("SE", "WE", "WE", "SWE", "WE", "WE", "SWE", "WS", "NS", "SE", "WE", "NW", "SE", "W",
"NE", "NSW", "NS", "NS", "SE", "WE", "NWUD", "SE", "WSUD", "NS", "N", "NS", "NSE", "WE", "WE", "NSW",
"NS", "NS", "S", "NSE", "NSW", "S", "NSUD", "N", "N", "NS", "NE", "NW", "NE", "W", "NSE", "WE", "W",
"NS", "SE", "NSW", "E", "WE", "NW", "S", "SW", "NW", "NE", "NWE", "WE", "WE", "WE", "NWE", "NWE", "W")
DArray = Array("DARK CORNER", "OVERGROWN GARDEN", "BY LARGE WOODPILE", "YARD BY RUBBISH", "WEEDPATCH",
"FOREST", "THICK FOREST", "BLASTED TREE", "CORNER OF HOUSE", "ENTRANCE TO KITCHEN", "KITCHEN AND GRIMEY
COOKER", "SCULLERY DOOR", " ROOM WITH INCHES OF DUST", "REAR TURRET ROOM", "CLEARING BY HOUSE", "PATH",
"SIDE OF HOUSE", _
"BACK OF HALLWAY", "DARK ALCOVE", "SMALL DARK ROOM", "BOTTOM OF SPIRAL STAIRCASE", "WIDE PASSAGE",
"SLIPPERY STEPS", "CLIFFTOP", "NEAR CRUMBLING WALL", "GLOOMY PASSAGE", "POOL OF LIGHT", "IMPRESSIVE
VAULTED HALLWAY", "HALL BY THICK WOODEN DOOR", "TROPHY ROOM", "CELLAR WITH BARRED WINDOW", "CLIFF PATH",
" CUPBOARD WITH HANGING COAT", _
"FRONT HALL", "SITTING ROOM", "SECRET ROOM", "STEEP MARBLE STAIRS", "DINING ROOM", "DEEP CELLAR WITH
COFFIN", "CLIFF PATH", "CLOSET", "FRONT LOBBY", "LIBRARY OF EVIL BOOKS", "STUDY WITH DESK AND HOLE IN
WALL", "WEIRD COBWEBBY ROOM", "VERY COLD CHAMBER", "SPOOKY ROOM", "CLIFF PATH BY MARSH", "RUBBLE-STREWN
VERANDAH", "FRONT PORCH", _
"FRONT TOWER", "SLOPING CORRIDOR", "UPPER GALLERY", "MARSH BY WALL", "MARSH", "SOGGY PATH", "BY TWISTED
RAILING", "PATH THROUGH IRON GATE", "BY RAILINGS", "BENEATH FRONT TOWER", "DEBRIS FROM CRUMBLING
FACADE", "LARGE FALLEN BRICKWORK", "ROTTING STONE ARCH", "CRUMBLING CLIFFTOP")
OArray = Array("", "PAINTING", "RING", "MAGIC SPELLS", "GOBLET", "SCROLL", "COINS", "STATUE",
"CANDLESTICK", "MATCHES", "VACUUM", "BATTERIES", "SHOVEL", "AXE", "ROPE", "BOAT", "AEROSOL", "CANDLE",
"KEY", "NORTH", "SOUTH", "WEST", "EAST", "UP", "DOWN", "DOOR", "BATS", "GHOSTS", "DRAWER", "DESK",
"COAT", "RUBBISH", "COFFIN", "BOOKS", "XZANFAR", "WALL", "SPELLS")
FArray = Array(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0)
FArray(18) = 1: FArray(17) = 1: FArray(2) = 1: FArray(26) = 1: FArray(28) = 1: FArray(23) = 1: LL = 60:
RM = 57: M$ = "OK"

Testo.Text = "YOUR LOCATION IS:"
Testo.Text = Testo.Text & vbCrLf & DArray(RM)
Testo.Text = Testo.Text & vbCrLf & vbCrLf & "POSSIBLE EXITS:" & vbCrLf
For I = 1 To Len(RArray(RM))
    Testo.Text = Testo.Text & Mid$(RArray(RM), I, 1) & ","
Next I
Testo.Text = Testo.Text & vbCrLf
For I = 1 To G
If LArray(I) = RM And FArray(I) = 0 Then Testo.Text = Testo.Text & "YOU CAN SEE " & OArray(I) & " HERE"
Next I
Testo.Text = Testo.Text & vbCrLf & "========================="
Testo.Text = Testo.Text & vbCrLf & M$
Testo.Text = Testo.Text & vbCrLf & vbCrLf & "What is your next move?"
M$ = "WHAT"
InputString.Text = ""
End Sub

Private Sub InputString_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
    Call Command_Check_Click
End If
End Sub
```

Come disattivare i tasti corrispondenti agli switch del joystick

di Attilio Capuozzo Fondatore RetroProgramming Italia – RP Italia

Nella 4° puntata del Tutorial su come sviluppare un gioco interamente in BASIC V2, che trovate sul gruppo RetroProgramming Italia – RP Italia, Felice Nardella ha inserito un'utilissima Tabella che, per comodità, alleghiamo al presente articolo (Fig. 1).

La Tabella mostra i tasti – o la combinazione di tasti – che corrispondono ai 5 switch del Joystick collegato alla Control Port 1 o 2 del C64.

	PORTA 1	PORTA 2
SU	1	SPAZIO + F1/F2
GIU'	← (Freccia in alto a sx)	SPAZIO + Z
DESTRA	2	SPAZIO + B
SINISTRA	CTRL	SPAZIO + C
FIRE	SPAZIO	SPAZIO + M

Figura 1

I 5 switch sono le 4 direzioni di movimento (Nord, Sud, Est, Ovest) + il Fire Button.

Lo scopo del presente Trick è quello di disattivare completamente l'utilizzo di questi tasti nei programmi che gestiscono i movimenti tramite, appunto, il Joystick. Iniziamo col dire che la prima operazione da fare è quella di disabilitare la scansione della tastiera che avviene, di norma, ogni 1/60 secondo.

La scansione della tastiera, come anche la gestione del lampeggiamento del cursore (Cursor Blinking) nonché l'aggiornamento del cosiddetto Jiffy Clock, l'orologio software del C64 preciso al secondo (le Variabili riservate TI e TI$, del BASIC V2, si riferiscono al Jiffy Clock), sono gestiti tramite una Routine di Interrupt.

A generare ("triggerare") l'Interrupt è un Contatore a 16 bit, detto Timer A, che costituisce una delle 5 sorgenti di Interrupt di uno dei 2 CIA – il CIA #1 – acronimo di Complex Interface Adapter (6526), i chip specializzati deputati al colloquio con le periferiche di I/O ossia di Input/Output.

Il chip CIA #1 è collegato, tramite una linea di tipo IRQ, al Pin 3 della CPU 6510.

IRQ è l'acronimo di Interrupt ReQuest ed è uno dei 2 tipi di Interrupt Hardware contemplati dall'architettura del C64. L'altro Interrupt Hardware è l'NMI acronimo di Non-Maskable Interrupt; il CIA #2 è collegato, tramite una NMI line, al Pin 4 del microprocessore 6510.

Le 5 sorgenti di Interrupt del CIA #1 possono essere rilevate tramite uno dei 16 Registri interni del chip, l'Interrupt Control Register (CIAICR) schematizzato in Fig. 2.

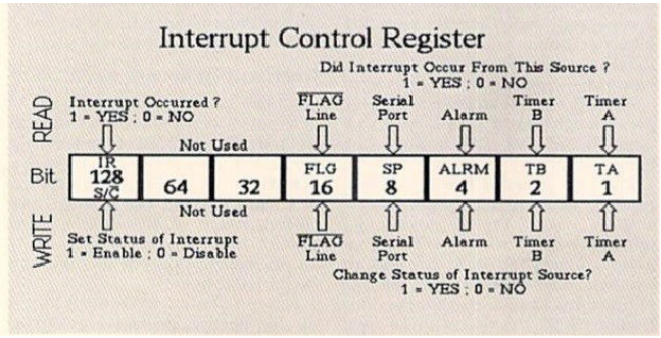


Figura 2

L'Interrupt Control Register è mappato in memoria alla locazione 56333 ($DC0D).

Per disabilitare una sorgente di Interrupt, porremo a 0 (ossia resetteremo) il bit alto del Registro (bit number 7; bit value 128) e scriveremo un 1 nel bit corrispondente all'Interrupt che intendiamo disabilitare.

Nel caso specifico il nostro scopo è di disabilitare la generazione dell'IRQ Interrupt da parte del Timer A che nel Registro 56333 è controllato dal bit 0.
Quindi è necessario utilizzare la seguente istruzione:
poke 56333,1

Ricordiamo brevemente che il Timer A effettua un countdown partendo da un valore di start predefinito – detto LATCH VALUE – fino ad arrivare a 0. Quando arriva a 0, se il corrispondente bit nel predetto Registro è abilitato, si genererà la richiesta di Interrupt.

Di default il CIA #1, come detto, genera un IRQ Interrupt (tramite il Timer A) 60 volte al secondo.

Prima di uscire dal programma dovremo ricordarci di riabilitare il Timer A (e in definitiva di riattivare la scansione della tastiera) scrivendo un 1 nel bit 7 del CIAICR e, di nuovo, un 1 nel bit 0 che rappresenta la nostra sorgente dell'Interrupt: poke 56333,129

La modifica del valore contenuto nell'Interrupt Control Register NON è ancora sufficiente a raggiungere il nostro scopo ossia quello di fare in modo che l'utente NON possa utilizzare le combinazioni di tasti alternative ai movimenti del Joystick collegato a una delle 2 Control Port del C64. I 64 tasti fisici del Commodore 64, sono contenuti in una matrice di 8 Righe per 8 Colonne ossia la Keyboard Matrix (Vedi Fig. 3).

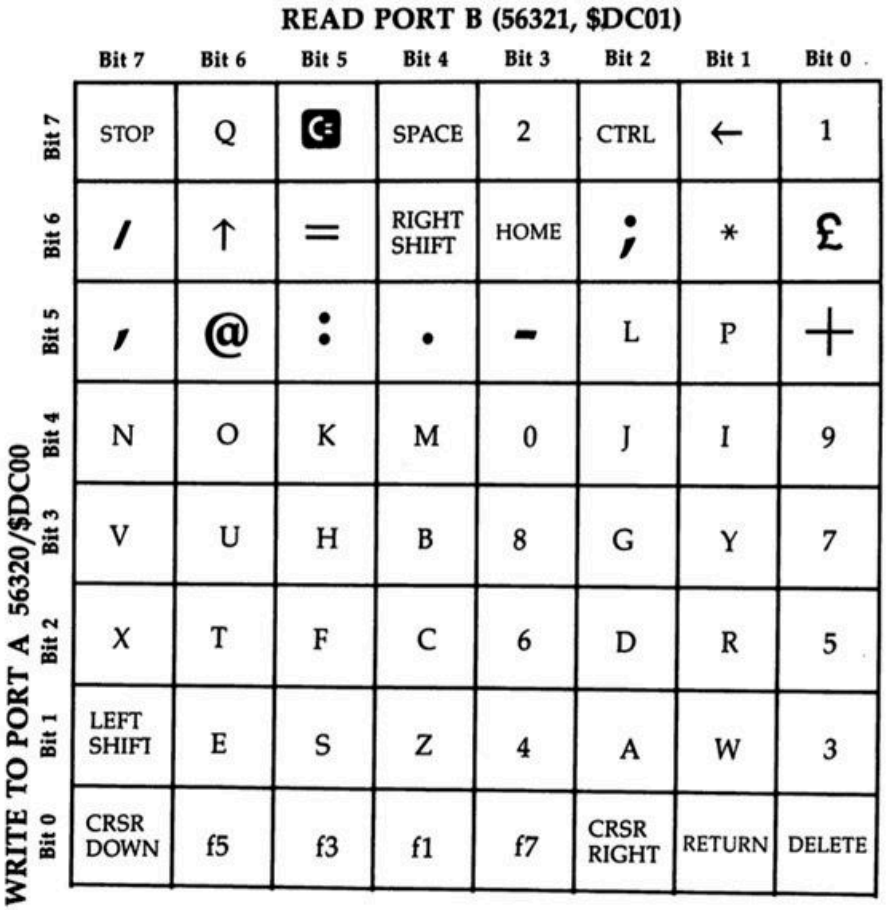


Figura 3

La Keyboard Matrix è collegata al CIA #1 tramite 2 dei suoi 16 Registri interni: il Data Port Register A mappato all'indirizzo 56320 ($DC00) e il Data Port Register B corrispondente alla locazione di memoria 56321 ($DC01). In realtà I tasti fisici di un C64 sono in tutto 66; nella Keyboard Matrix mancano il tasto SHIFT LOCK che non ha bisogno di essere rilevato in quanto viene sostituito dal normale SHIFT di sinistra, il LEFT SHIFT (di cui ne simula la pressione), e il tasto RESTORE la cui pressione genera un Interrupt di tipo NMI.

Il Registro 56320 (Data Port A) viene utilizzato in scrittura per selezionare le Colonne da leggere della Keyboard Matrix mentre il Registro 56321 (Data Port B), usato in lettura, serve a leggere le Righe della Matrice relative alla Colonna precedentemente selezionata tramite il Data Port Register A.

Giusto per complicare la vita di noi RetroProgrammatori del C64, negli stessi Registri vengono memorizzati – nei primi 5 bit -i valori corrispondenti ai 5 switch del Joystick come ben descritto dal già citato Tutorial di Felice Nardella. Dunque quando andremo a leggere il contenuto di uno dei 2 predetti Registri NON avremo modo di sapere se i bit dei Registri sono stati modificati dal movimento del Joystick (e/o dalla pressione del Fire Button) oppure dalla pressione di 1 tasto (o di una combinazione di tasti) che simulano i 5 switch del Joystick.

Per poter essere sicuri di leggere, in uno dei 2 Data Port Register, un valore che provenga dal Joystick non ci rimane che andare preventivamente a scrivere nel Registro 56320 (il Data Port Register utilizzato, come detto, in scrittura) un 1 nei suoi 8 bit corrispondenti alle 8 Colonne della Keyboard Matrix. Infatti, scrivendo 1 nei bit del Registro 56320 ignoreremo la lettura delle corrispondenti Colonne della KeyBoard Matrix (l'effetto opposto lo si ottiene, invece, ponendo a 0 i bit del Registro).
In BASIC V2, pertanto, andremo a digitare la seguente istruzione: poke 56320,255

In definitiva le 2 istruzioni che andranno eseguite PRIMA delle successive letture di una delle 2 Control Port a cui è collegato il Joystick (Registro 56320 = Control Port 2 o Registro 56321 = Control Port 1) sono le seguenti:
poke 56333,1:poke 56320,255

Finalmente abbiamo raggiunto il nostro agognato scopo! That's all folks!

Vi ricordo che potete raggiungere il gruppo **RetroProgramming Italia - RP Italia**:
https://www.facebook.com/groups/retroprogramming/

Buon compleanno Monkey Island

di Edoardo "Edward mani di forbice" Ullo

Ci sono anniversari importanti che non possono essere mandati in secondo piano. Trent'anni fa, era il 15 ottobre del 1990, **LucasFilm** pubblicava **The Secret of Monkey Island**, avventura punta e clicca di **Ron Gilbert** e di altri sviluppatori che avrebbero poi lasciato un segno nell'industria (leggasi Tim Schafer e Dave Grossman).



Questo è un gioco che, nei sui quattro dischetti della versione Amiga ma anche nella sua edizione Pc, è entrato nella storia per tanti motivi. A partire dallo strampalato protagonista, quel **Guybrush Threepwood** che incurante di tutto e di tutti voleva diventare un temibile pirata ma che – puntualmente – veniva preso in giro da tutti. Ingiuriato in tutti i modi e preso per i fondelli sia dal vecchio mercante sia dal guardiano dell'isola di Melée (la fiabesca isola nel profondo dei Caraibi che fa da sfondo alla prima parte del gioco) che grazie alla sua vista di falco alla "geometra Filini" seppe riconoscere subito la persona che aveva davanti apostrofandola con un "Ehilà, damerino"! Per non parlare delle musiche. Tante, d'atmosfera ed orecchiabili agli strampalatissimi dialoghi, puzzle, duelli e robe varie.

Ma, vi chiederete voi, cosa c'entra The Secret of Monkey Island in un numero dedicato ad Halloween? La risposta ve la diamo immediatamente. A parte l'importanza ed il



peso di questo capolavoro assoluto, infatti, il gioco ha qualche suo momento horror. Sia pure, ovviamente, mitigato da tantissimo umorismo: "A nessun animale è stato fatto del male durante la produzione di questo gioco", si legge in un avviso.

Vi ricordate la scena **splatter** in cui l'antagonista della serie, il **pirata fantasma** LeChuck (lui si che è davvero temibile, ndr), smembra il corpo dello sceriffo Fester Shinetop di cui aveva preso le sembianze per sbarazzarsi del nostro "eroe"? Beh, se non è horror questo. Inoltre, sia pure in modo comico e leggerissimo, si parla molto di voodoo. Ed è così che il nostro biondo eroe riesce a portare la nave sgangherata a Monkey Island nonostante la ciurma si fosse auto ammutinata in modo molto simpatico. Certamente non vi spieghiamo come: i più grandi se lo ricorderanno, i più giovani potranno giocarlo e divertirsi.



C'è anche un **troll**… o qualcosa di simile che ci ostacola e non ci fa passare il ponte, salvo farci andare avanti dietro "lauto" pasto.



E che dire dei **cannibali**? Di norma sono da temere. E per forza: sono nemici spietati, agguerriti, che mangiano altri esseri umani. Ma in The Secret of Monkey Island li troviamo salutisti ed attenti a sale e colesterolo. Encomiabile. Tuttavia, il nostro Guybrush dovrà anche affrontare un labirinto infernale ma per superarlo avrà bisogno di una



bussola speciale: la **testa del navigatore**. Unica in grado di trovare la via in questo dedalo dantesco dove crescono i funghi. Questa è davvero una testa parlante tenuta in vita grazie ad una speciale magia dei cannibali di Monkey Island.

E poi? Un'altra sfumatura potenzialmente horror è senza dubbio quella che si vive nel passaggio in cui si arriva alla nave del pirata fantasma LeChuck per salvare Elaine Marley e farla in barba ad una ciurma festante. Bisogna diventare **invisibili** per fregare questi pirati non morti così fortunati dall'aver incontrato in vita LeChuck che li ha uccisi ed arruolati da non morti.

Insomma, tutto quello che tradizionalmente ad Halloween dovrebbe esserci: i fantasmi, i cannibali, una testa parlante, la magia voodoo.

Ingredienti che se fossero mischiati tra loro darebbero come logico risultato un gioco o una produzione horror di prim'ordine. In The Secret of Monkey Island, invece, li troviamo mischiati all'ironia ed alla spensieratezza.



Il merito? Una scrittura super, dialoghi fantastici, geniali, scritti tanto con cuore che con la testa.



Questo è il prototipo dell'avventura punta e clicca perfetta, almeno per le produzioni di inizio anni '90.

Il sequel è altrettanto spettacolare e più longevo ma forse ha un pizzico di carisma in meno rispetto al suo predecessore che celebriamo in questo nostro numero dedicato ad Halloween.

BILLY MASTERS WAS RIGHT

Anno: 2020
Sviluppatore: Paco Garcia
Genere: Avventura Punta e clicca.
Piattaforma: Windows

Nessuno crede a Billy Masters.
Un sedicenne americano degli anni 80/90 accusato di aver assunto droghe al liceo e di aver caluniato il suo insegnate per gravi crimini.
Stanchi del suo atteggiamento strafottente, i suoi genitori gli hanno imposto il coprifuoco più totale fino a quando non si scuserà con il suo insegnante.



Il problema è che il suo insegnante è davvero uno psicopatico e sta tramando qualcosa nei confronti del vicinato.
Billy è messo alle strette e cercherà in tutti i modi di far valere le sue ragioni... Fino a quando non sarà chiaro che "Billy Masters aveva ragione".

Che bel gioco davvero! Forse non un capolavoro, ma una ventata di aria fresca (che poi così fresca, ricalca i bei tempi andati) tra i numerisi fps, rpg e soci ultra tecnologici fatti di framerate assurdo e 4k.
Ma davvero vi divertite con questi numeri? Vi interessa davvero sapere se divertirsi è a 60 fps o meno? Vabbé, lasciamo stare e torniamo al gioco.
Billy Masters Was Right è un breve gioco d'avventura in stile grafico Maniac Mansion e una trama ispirata a film come "The Burbs" o "Disturbia".
L'atmosfera del gioco è un misto tra nostalgia anni 80, i classici film di suspance dello stesso periodo e un teenage movie.
Classico punta e clicca che ricorda il sistema ideato appunto per Maniac Mansion dalla Lucas Film e devo dire che funziona ancora alla perfezione.
Divertente, mai banale, ben congeniato negli enigmi... irriverente nei dialoghi e nella trama e in alcuni punti anche sconcertante (una grafica cartoonesca con tematiche molto forti, clamoroso!).
Carina la realizzazione tecnica che ci catapulta su un Commodore 64 dei tempi di Maniac Mansion, ma veloce e precisa.
Il punto di forza per me è invece una colonna sonora adatta alle occasioni di gioco, mi ha davvero coinvolto.
Purtroppo non è lungo ne' impossibile, ma vi farà passare alcune orette di gioco.
Lo trovare gratuito sul sito del programmatore:
https://postmodernadventures.itch.io/billy
oppure donando una piccola cifra al programmatore.

Disponibile per Windows in Inglese, Spagnolo, Catalano e Tedesco (le localizzazioni sono ottime).

Questo Halloween caricate questo gioco e ricordatevi che... Billy aveva ragione!

di **Carlo N. Del Mar Pirazzini**







GIUDIZIO FINALE

» Giocabilità 80%
Sistema punta e clicca rodato che ricorda proprio Maniac Mansion.
Trama ben strutturata, buoni enigmi. Bello.

» Longevità 60%
Non troppo lungo purtroppo, ma divertente.
Sufficienza piena.



PROJECT FIRESTART

Anno: 1989
Sviluppatore: Dynamix
Editore: Electronics Arts
Genere: Action/avventura/ survival horror
Piattaforma: Commodore 64



"Diario del capitano - 13 febbraio 2061 - Astronave da ricerca PROMETHEUS.

La missione finanziata dalla Fondazione Science System è stata un fallimento. Gli esperimenti per la creazione di "super uomini" dalle capacità fisiche straordinarie è stato un completo fallimento. Come posso aver accettato di far parte di questo scempio?
La brana delle corporazioni in nome del potere e del denaro hanno fatto creare dei mostri. Stiamo morendo tutti e.. nel silenzio di queste pareti nessuno può sentirci, ma noi sentiamo loro e i loro artigli.

Eccoli, stanno entrando..."

Questo potrebbe essere l'inizio di Project Firestart, ambientato a bordo della Prometheus abbandonata e piena di cadaveri. L'agente protagonista, Jon, si muoverà all'interno della struttura cercando di capire cosa è accaduta attraverso i giornali di bordo, cercando di salvare l'unica sopravvissuta e fuggendo dalla stessa struttura prima di farla detonare.

Una trama incredibile per un gioco incredibile, spesso dimenticato ma che ha aperto realmente un genere anni prima di Resident Evil o Silent Hill. Project Firestart è un gioco d'azione con grafica in pseudo 3d-isometrico. L'astronavame è composta da numerose stanze, fisse e a scorrimento. Il nostro eroe può muoversi liberamente nella nave e con la possiilità di difendersi tramite una pistola laser efficacissima ma con un numero limitato di munizioni a disposizione.

Alcuni terminali che si possono incontrare nel corso del gioco contengono diari e giornali del personale a bordo della Prometheus, attraverso il quale il giocatore verrà a conoscenza di cosa è accaduto a bordo dell'astronave. Inoltre è possibile trovare alcuni oggetti in grado di ricaricare l'energia del personaggio o fornire nuove munizioni o altri oggetti utili per la risoluzione del gioco.
Occasionalmente, in seguito a un'azione del giocatore o in punti predeterminati, il gioco va in pausa per mostrare al giocatore una schermata in cui viene mostrato il

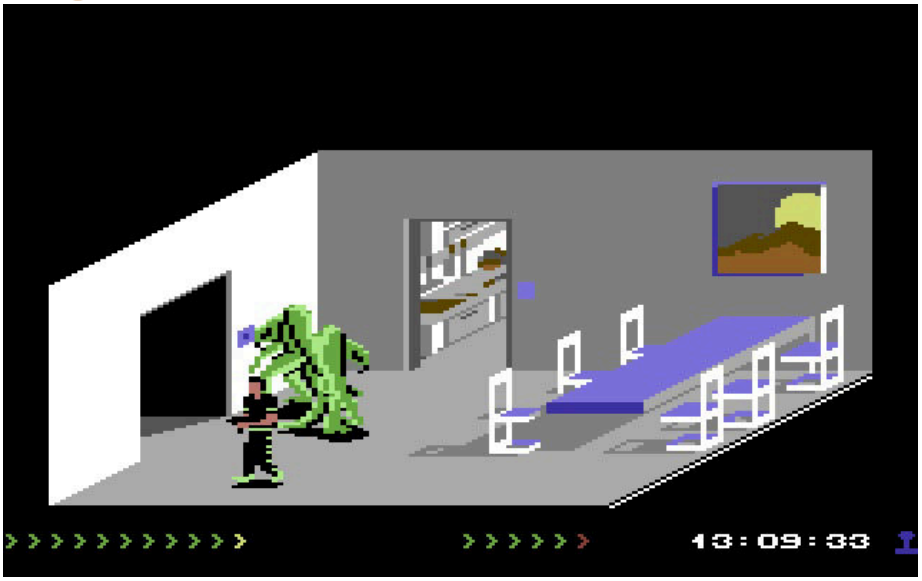












protagonista, un altro personaggio o uno dei mostri, dando al giocatore la possibilità di vedere meglio il loro aspetto.

Come dicevamo, Project Firestart è un videogioco d'azione nato su Commodore 64.
È stato disegnato da Jeff Tunnell e Damon Slye e pubblicato dalla Electronic Arts nel 1989.
A tutti gli effeti è da considerarsi il primo esempio di survival horror con finali multipli, concepito dieci anni prima dei più blasonati giochi del genere.

L'amosfera cupa, la sensazione che il nemico si nasconda dietro ogni angolo e quel senso di impotenza quando si è a corto di munizioni lo rese celere al tempo e gli diede un alone di mistico mistero. Le riviste lo osannarono per il concept e per il sistema di gioco, i giocatori rimasero di stucco per la realizzazione salvo poi lamentarsi per la difficoltà anomala ed un approccio totalmente diverso.

Graficamente sbalorditivo, vario e realizzato con questa grafica isometrica che dava ampio respiro al giocatore, una cosa senza precedenti in altri titoli. Un titolo che ci ricordava le atmosfere dei film horror e sci fi del tempo (Alien su tutti, ma soprattutto la cosa). I giocatori del tempo non erano abitatuati a dei nemici alieni così disgustori e coriacei rispetto allo standard.

Difficile, violento, angosciante...
Bellissimo.

Una vera e propria killer application per il Commodore 64. Mai uscito per altri sistemi, invidiato da possessori di macchine più potenti.

Provatelo chiusi nella vostra stanza... Col C64 acceso, il vostro joystick e il silenzio della stazione spaziale dove...

Nessuno potrà sentirvi urlare.

di **Carlo N. Del Mar Pirazzini**



GIUDIZIO FINALE

» Giocabilità 85%

Difficile e non adatto ai "puristi" degli arcade classici. Una volta dentro al gioco non smetterete tanto facilmente.

» Longevità 85%

La Prometheus non si liberarà in un secondo. Molte ore di gioco.





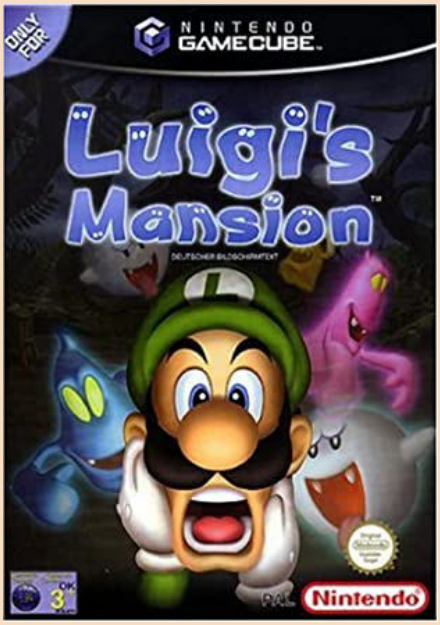






LUIGI'S MANSION

Anno: 2001
Editore: Nintendo
Sviluppatore: Nintendo
Genere: Survival Horror/Action
Piattaforma: Game Cube



"Mariooooooooo..... Mario!"
Quante volte premendo il tasto A del Cubo abbiamo fatto ripetere a Luigi questa frase? Io molto spesso.
Luigi's Mansion è un titolo concepito da Mamma Nintendo che ha dato il via ad una saga molto spesso sottovalutata, ma dotata di un appeal, una realizzazione e un gameplay eccellente. Questo primo capitolo uscí nel 2001 (2002 in Italia) come primo titolo per il neonato Game Cube.

Fa parte della macro-serie di titoli legati alla figura di Mario e ha avuto un remake su DS e un bellissimo terzo episodio sulla Switch.

Alla sua uscita fu criticato dalle riviste e dagli addetti ai lavori, che superficialmente avevano bollato il titolo come "fanciullesco" e semplice. Ma ogni cosa col tempo migliora (il vino ad esempio, ndN). Andiamo con ordine.

Il gioco è fondamentale per la storia del brand in quanto non solo da spessore ad un personaggio spesso in seconda fila rispetto a Mario, ovvero Luigi, che diventa protagonista inaspettato, ma fa fondamentalmente due cose: la prima si stacca dal concetto di platform classico della serie e soprattutto scimiotta i numerosi survival horror che imperversavano in quel periodo.
Un esperimento mai tentato prima da Nintendo e che non fu capito dalle riviste. Cosa che invece è da ritenere innovativa e incredibile!

In pochi sanno però che Luigi non era la prima volta che si metteva nei panni del protagonista. Dieci anni prima era stato protagonista con "Mario is Missing!".

La storia è semplice: Mario è stato rapito dai Boo, i popolari fantasmini della serie. Per questa loro caratteristica, sulla scia dei film dei Ghostbusters, Luigi si troverà ad affrontarli con un aspiratutto ed un Game Boy Horror con l'obiettivo di disinfestare le oltre 50 stanze del palazzo stregato dove si ritrova e liberare il fratello dopo aver ovviamente sconfitto il malvagio Re Boo, il boss finale. Ogni stanza racchiude segreti o trappole o momenti di paura spaventosi (fino a un certo punto, ndN).

E' nel gameplay che questo primo episodio si presentava e si presenta vario ed originale, con spunti che lo rendono tutt'ora incredibilmente moderno e mai noioso. La stessa formula si è rivelata vincente ed è rimasta pressoché identica nei nuovi capitoli. Questo è un punto di forza incredibile. L'uso del meraviglioso joypad del cubetto Nintendo è magistrale. Comodissimi i comandi e





completi con l'uso bilanciato di tutti i tasti e dello stick analogico.

Come nel mitico Ghostbusters al cinema, Luigi, più che combattere o saltare sugli avversari, li aspira utilizzando il fido Poltergust 3000, che aspira ogni sorta di fantasmi, dai semplici fantasmi generici a quelli in "cornice" che infestano la casa fino agli spettri più complicati e ai velocissimi Boo.

L'uso dell'aspirapolvere non si limita alla semplice aspirazione, ma potete sfruttare il Poltergust per spostare oggetti, risucchiare tende e coperte, aspirare monete, oggetti bonus e chiavi (fondamentali per aprire le stanze da esplorare).

Il vero punto di forza di questo capitolo però resta certamente l'utilizzo degli oggetti ed elementi che servono al nostro protagonista per avanzare nella storia e sempre più in alto nella villa, a partire dallo stesso aspiratutto dai poteri straordinari fino al Game Boy Horror, congegno che permette di osservare l'ambiente che circonda e dare informazioni sugli oggetti che incontriamo nel nostro cammino, ma funge anche da mappa dell'intero palazzo, aiutando il giocatore indicandogli le stanze chiuse a chiave, quelle già completate e quelle accessibili, perché abbiamo magari trovato una chiave speciale per potervi accedere.

Questo particolare oggetto può inoltre indicare la presenza di fantasmi e avere addirittura la funzione di inventario, indicando la quantità di soldi, gemme, spettri che possediamo.

La critica, come dicevamo, criticò il gioco bollandolo come un titolo bello ma "piccolo e per piccoli".
Certo le missioni non saranno miliardi e non impiegherete certamente migliaia di ore per finirlo, ma è un gioco bello, sereno e rilassante (nonostante gli spettri).

Bellissimo da giocare davanti alla tv al buio proprio ad Halloween, magari col proprio figlio/a tra uno spavento di un fantasma, una faccia spaventata di Luigi e le risate in casa. Questo lo rende speciale e, dopo molto tempo, gli stessi esperti del tempo lo hanno rivalutato.

Se fosse un alcolico sarebbe un buon brandy da gustare con calma accanto al fuoco, ma è un videogame ed è ben fatto, giocabile e molto bello da vedere.

Un perfetto esempio di come Nintendo è "differente".



Se avete un Cubo non posso non consigliarvelo, ma gira anche in emulazione con il Dolphin, lo trovate in versione remake su DS e sugli store di Nintendo.

Insomma, giocatelo!

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 95%
E' un prodotto ben studiato e giocabile. Il gameplay è ben calibrato, la storia veloce e accattivante e vi sentirete subito immersi nella missione alla ricerca di Mario. Ottimo uso del joypad.

» Longevità 80%
Non è un titolo eterno, ma è un titolo che vi terrà compagnia ogni volta che lo toglierete dallo scaffale per infilarlo nel Cubo.



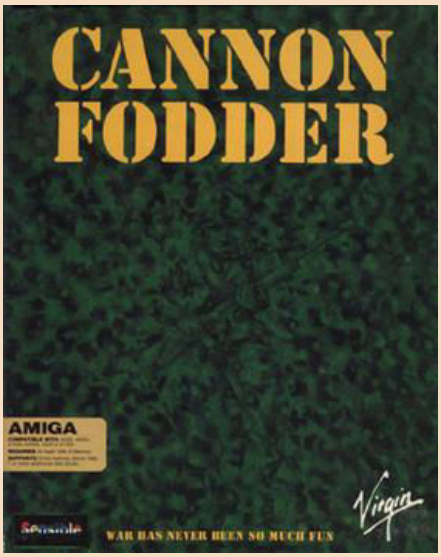




CANNON FODDER

Anno: 1993
Editore: Sensible Software/Virgin
Sviluppatore: Sensible Software
Genere: Sparatutto/Strategia in tempo reale
Piattaforma: Amiga





Chi nel 1993 possedeva un Commodore Amiga non poteva non aggiungere alla propria collezione videoludica questo piccolo capolavoro creato da quei geniacci che avevano regalato all'Europa (e al mondo) il gioco di calcio definitivo: Sensible Soccer.

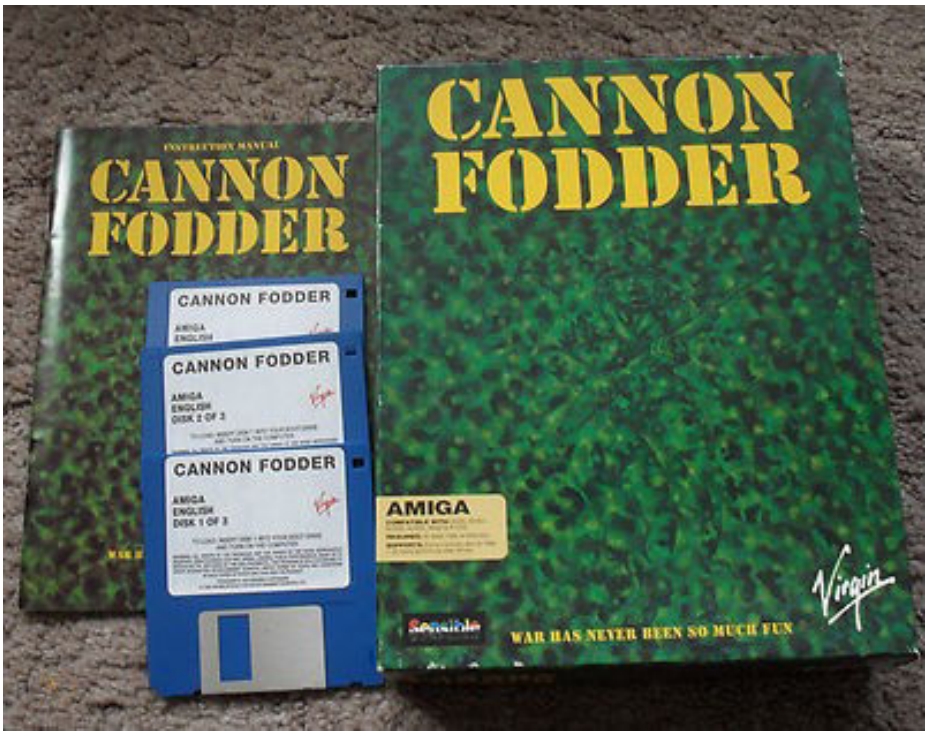
I coder inglesi ebbero un'altra fantastica idea: reclutare migliaia di piccoli omini in delicata pixel-art (come del resto lo erano i calciatori di Sensible Soccer) e piazzarli in altrettanto deliziosi scenari disegnati con visuale dall'alto per farli diventare come dice il titolo del gioco "carne da macello" ovvero un modo goliardico e diretto per farci provare direttamente sui nostri monitor che la guerra non era davvero mai stata così divertente "War has never been so much fun".

Giocabilità al top dentro serie di missioni di difficoltà crescente dove ovviamente lo scopo era nella maggior parte dei casi quello di eliminare la presenza nemica sul territorio (comprese basi e installazioni militari) cercando nel contempo di perdere meno soldati possibile (i sopravvissuti venivano aumentati di grado missione dopo missione) utilizzando in maniera geniale come unico sistema di controllo il fido mouse del Commodore Amiga.





Il gioco non era propriamente uno strategico in tempo reale (genere che nasceva proprio in quegli anni grazie a Dune 2 su Amiga) ma



un bel po' di strategia rapida dietro le nostre azioni di stampo arcade avrebbe comunque fatto la differenza tra la vita e la morte dei nostri piccoli soldati digitali.

Un capolavoro di giocabilità (e di game design) che contemplava l'utilizzo non solo di armi da fuoco standard ma anche di razzi, carri armati e tutto quanto ci sarebbe potuto servire per arrivare alla promozione delle nostre truppe evitando di farle diventare delle piccole croci sparse sulla collina dove si reclutavano nuove risorse tra un livello e l'altro.

Un capolavoro di game design che ci spingeva a sfruttare il territorio a nostro favore: nella giungla potevano mimetizzarci tra gli alberi (ma potevano farlo anche i soldati nemici) mentre nelle zone artiche si doveva fare attenzione al ghiaccio senza contare le mine, le trappole nascoste nella vegetazione, e decine di altri tocchi di classe (come la possibilità di ferire soltanto i nemici che poi si contorcevano e urlavano dal dolore in attesa di essere finiti) che rendevano il mondo di Cannon Fodder una esperienza immersiva ed entusiasmante trattando un tema crudo come la guerra in modo distaccato e divertente come in effetti era annunciato nella sigla iniziale composta e cantata dal progettista del gioco Jon Hare ovvero la canzone "War has never been so much fun" ciliegina sulla torta di un prodotto davvero meritevole di essere rigiocato ancora oggi.

di **Flavio Soldani**

GIUDIZIO FINALE

» Giocabilità 96%

Un riuscito mix tra strategia in tempo reale e sparatutto con talmente tanti tocchi di classe e con una giocabilità così alta da vincere a mani basse la sfida col tempo.

» Longevità 95%

Come detto sopra Cannon Fodder è un videogioco geniale che, insieme a molti altri, ha contribuito a fissare nei nostri cuori la creatività che permeava i primi anni '90 facendoci amare i sistemi Commodore Amiga.

Il gioco fu poi convertito sulla maggior parte delle piattaforme dell'epoca, ebbe un seguito Cannon Fodder 2 ed un rifacimento in 3D non realizzato da Sensible Software nel 2011 chiamato Cannon Fodder 3, purtroppo i seguiti non riuscirono ad avvicinarsi alla qualità del primo capitolo.





MICRO MAGES

Anno: 2019
Sviluppatore: Morphcat Games
Genere: Platform
Piattaforma: Nintendo NES

"Non essere un mago, sii magico!"
Leonard Cohen

Micro Mages per il Nintendo Entertainment System, sviluppato da Morphat Games, è un gioco di piattaforma a scorrimento verticale che ci vede nei panni di un mago appena giunto in una fortezza demoniaca antichissima per salvare la principessa dalle forze del male.

Il viaggio all'interno della fortezza porrà davanti al nostro eroe in tunica rosa grandi tesori, creature orribili e trappole mortali e solo grazie alle abilità magiche e alle sue capacità di scalatore il piccolo maghetto potrà raggiungere, con il nostro aiuto, la cima della torre.

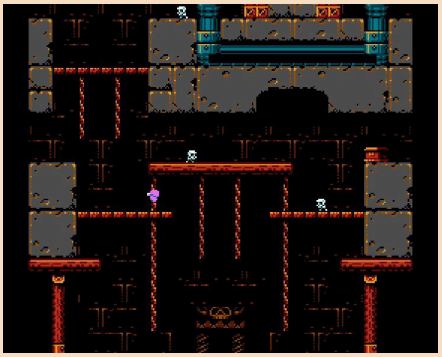
Sono 4 le torri da superare:
The Haunted Dungeon - una vecchia torre perduta piena di pipistrelli e scheletri e dove un enorme fantasma uccide urlando le sue vittime.
Torre Valhalla - dove i Goblin stanno costruendo una struttura che porta al Valhalla.
The Jungle Temple - costruito all'interno di un vulcano dormiente e governato dal Principe delle Tenebre... e infine
Last Tower- L'ultima torre è il quartier generale alquanto misterioso del male supremo che ha intrappolato la principessa.

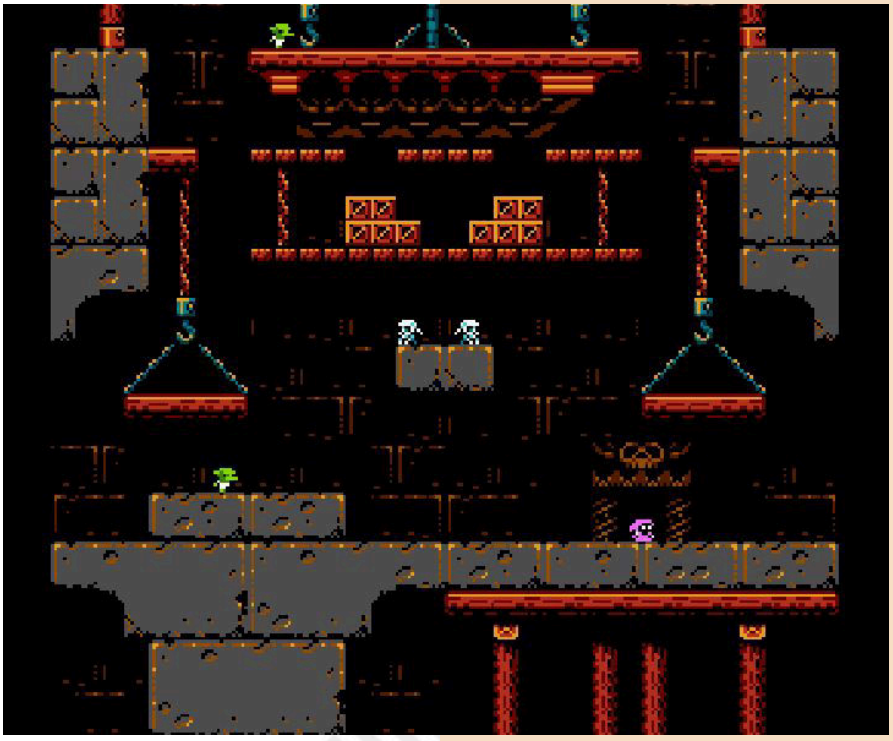
La meccanica principale del gioco è lanciare incantesimi contri i nemici col tasto A del joypad e utilizzare il salto (sul tasto B) per salire in cima ai vari livelli. L'azione di salto sul muro è abbastanza semplice da eseguire grazie ai controlli reattivi e al controllo perfetto del personaggio in ogni momento.

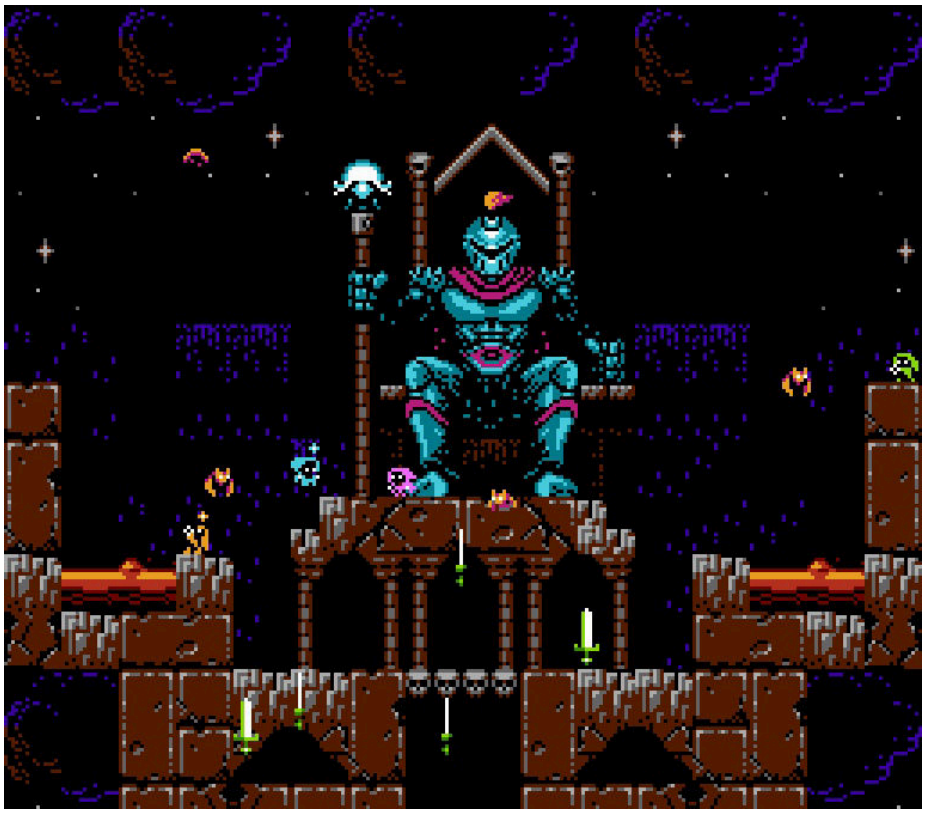
Il livello di sfida è piuttosto vario. Durante il percorso incontreremo diversi avversari e di trappole che ci renderanno il gioco mai noiso e molto avvincente. Inizialmente ci basterà un solo colpo per uccidere gli avversari semplice, ma con crescere dei livelli ci troveremo di fronte ad avversari più "coriacei" che necessiteranno di più colpi.
In nostro aiuto, durante la nostra esplorazione, troveremo numerose casse aperte e scrigni del tesoro. In ogni scatola troveremo diversi bonus che ci saranno molto utilie (la fata che ci donerà uno scudo per un colpo avversario, una piuma che ci darà la possibilità di volare e molto altro ancora).

Ogni 16.000 punti raccolti (attraverso i bonus e i nemici distrutti) ci verrà assegnata una vita extra.
Sono presenti anche i checkpoint, come in ogni platform che si rispetti, solitamente ubicati a metà del livello e davvero utili durante i percorsi più lunghi e complicati (il terzo mondo ad esempio).





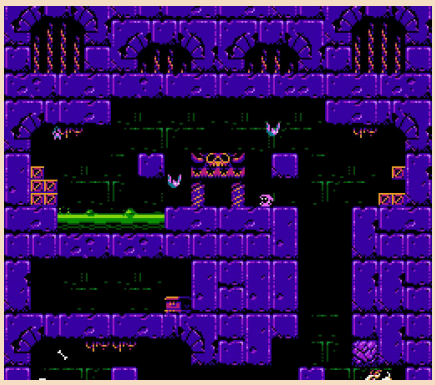


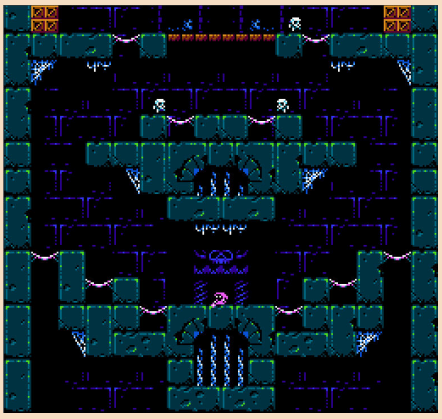


GIUDIZIO FINALE

» Giocabilità 90%
Fluido, dinamico e ben bilanciato sul grado di sfida. Non vi annoierà e vi terrà incollati al joypad per vedere tutti e 4 i mondi di sfida.

» Longevità 65%
...4 Mondi che però sono un po' pochini e una volta capiti i pattern di avversari ed enigmi vi verrà tutto più facile e troppo velocemente. Peccato.







Il gioco dispone anche di un comodo sistema di password per salvare la posizione che vi verrà fornito al termine di ogni torre.
Le boss fight le troveremo una volta raggiunta la cima della torre. I classici mostri presentano una sfida non semplice ma nemmeno troppo difficile una volta appresi gli schemi di attacco. Questo è un grosso punto a favore. Questa difficoltà ben bilanciata rendono Micro Mages un avvincente.

E' presente anche una modalità multi-player implementata molto bene e piuttosto divertente. Finché almeno un giocatore è vivo, non perderemo la vita del personaggio. I giocatori morti durante il percorso si trasformeranno in fantasi col potere di congelare gli avversari e aiutare i sopravvissuti. Possono anche distruggere casse e scrighi sperando di trovare una fata o un bonus che li riporterà in vita.
Come nei più classici platform anni 80, una volta completato il giro completo e sconfitto il boss finale sarà possibile ripartire da capo con un livello di difficoltà aumentato.
Che dire. Un gioco nuovo ben strutturato e con un ottimo level design e un gameplay perfetto.

I valori di produzioni su Micro Mages sono molto alti. La grafica è notevole. Dettagliata, animata con cura (il piccolo maghetto è dotato di una marea di animazioni incredibili, ndN) e tutto si muove con una grande fluidità. Un prodotto che sfrutta molto bene le capacità del Nes e che non presenta nemmeno un rallentamento o un flickering che spesso vediamo nei giochi per il Nintendo a 8 bit.
La colonna sonora e semplice e bilanciata al gioco. Non ossessiva ne ripetitiva, serve come sottofondo per procedere nell'avventura.
Note dolenti... poche ma ci sono. Il gioco non è impossibile una volta compresi i pattern dei livelli, degli avversari e delle trappole e solo 4 mondi sembrano pochi. Peccato perché con qualche livello in più sarebbe il "capolavoro" assoluto delle nuove produzioni per il NINTENDO a 8 bit.

Il gioco è disponibile in versione download su Steam e Itch.Io e gira su tutti gli emulatori per la console Nintendo (Mugen, Nostalgia....), ma vi consiglio di fare come ho fatto io e acquistare la cartuccia con tanto di manuale e package davvero molto curato.
La potete ordinare sul sito https://www.brokestudio.fr/product/micro-mages-nes/

di **Carlo N. Del Mar Pirazzini**

DOOM

Anno: 1993
Editore: Id Software
Sviluppatore: Id Software
Piattaforme: Pc MS DOS, Atari Jaguar, Super Nintendo, Sega Mega32x, Sega Saturn, Sony Playstation, 3DO, Amiga.
Genere: FPS

Fucili da caccia, motoseghe e demoni. Tutto questo è stato ed è Doom, e sebbene questo concetto sembri sciocco e privo di fantasia, in realtà è uno degli ingredienti chiave di questa leggenda dei videogiochi.

Ma perché tutto questo successo?
Nel 1993, diversi titoli controversi hanno avuto problemi. La maggior parte dei genitori e delle associazioni di categoria si è scagliata contro le violenze in Mortal Kombat o nel controverso Wolfestein 3D (il padre di Doom). Erano titoli violenti e sanguinari che erano sul mercato.
La controversia è scoppiata anche su Doom? Sì, ma in quantità minori. Il titolo è uscito sul mercato SHAREWARE e si è diffuso anche attraverso il passaparola e gli appassionati.

Le associazioni di categoria sono state messe a tacere dalle vendite del gioco e soprattutto dai giocatori SE NON ERA TOTALMENTE IMPORTANTE! Era un prodotto di nicchia per il periodo. Trama zero (ok, un pochino, ndN), il protagonista è importante? Neanche un po.

Doomguy, il nostro marine spaziale generale è stato un esempio di narrazione esperienziale, con il giocatore stesso invece di esservi semplicemente esposto. Ovviamente, l'elemento che lo rendeva così coinvolgente (oltre alla prospettiva in prima persona) erano i suoi fantastici effetti visivi simili a una cover di Brutal Death Metal, erano banali ma fantastici ... e violenti ... oh, erano ultra violenti !!!

Per un certo periodo di tempo, questo tributo pixelato alle classifiche dei film d'azione e dell'orrore è stata la prossima generazione di videogiochi. Ogni successivo gioco fpr su PC, console e persino Amiga era sullo stesso stile di Doom.
Nonostante la sua età, tuttavia, è ancora eccezionale. La grafica appare realistica, quasi grintosa, che si tratti di una semplice parete o di una superficie ultra decorata e dall'aspetto diabolico.

I vari livelli di luminosità danno origine a corridoi claustrofobici, con luci tremolanti o oscillanti o, addirittura, nere come la pece.

I film come Alien che sono evidenti influenze chiave, il tema principale del franchise è sempre stato ispirato da HR Giger, matrimonio immorale di carne e macchine e, come molti grandi giochi precedenti, il design dei livelli è astratto piuttosto che realistico, e sorprendente invece che banale. Per quanto riguarda i nemici, hanno un aspetto fantastico, hanno animazioni di morte incredibilmente dettagliate e raccapriccianti e, poiché sono in realtà sculture fotografate, hanno una sensazione quasi realistica per loro. Un gruppo eterogeneo di non morti e demoni che è diventato quasi iconico come i Goomba in Super Mario Bros o i fantasmi in PAC MAN.

L'aspetto è solo una parte dell'equazione, con il suono che completa il pacchetto nel miglior modo possibile. Oltre a una classica colonna sonora MIDI (ricordate il MIDI?), Ispirata - e leggermente strappata - da leggende del metal come Black Sabbath, Metallica, Megadeth e Judas Priest, tra gli altri, così come alcune melodie ambient più lente e che migliorano l'umore come la ciliegina sulla torta, tutti gli effetti sonori, dalle potenti esplosioni di armi, alle urla e ai ringhi di nemici allarmati, semplicemente perfetti - e aspetta un toro infernale carnivoro fatto di steroidi che vuole finirti! Per sentire l'eco del ruggito del gigantesco













Cyberdemon; il richiamo di accoppiamento tra ferocia e dolore.
Ma in tutto questo candore schizzato troviamo una macchia, una macchia che, soprattutto in questi tempi, ci fa capire l'età. Cosa riguarda? Ebbene ... il nostro eroe non può allacciarsi le scarpe, guardare il topo ... insomma, non può abbassare gli occhi e nemmeno alzarlo perché la tridimensionalità era ancora “un Prototipo "e non ancora in arrivo (arriverà qualche anno dopo con Quake e Descent).

Questo rende Doom vecchio, lo rende un pezzo di un'altra epoca.
Dove non abbiamo guardato al fotorealismo, 60 fps, 4k / 8k o motore ultra realistico.
Stavamo pensando di armarci di motoseghe per sventrare ogni Cacodemon presente.

Al momento della sua uscita, Doom fu criticato per aver sostenuto che si trattava di un semplice gioco in cui abbattere ondate di nemici dopo nemici, il che non poteva essere più lontano dalla verità.

Questo non è il solito Left for Dead, ma un gioco in cui troveremo una complessità di progettazione di livelli fantastici, pieni di trappole, con nemici posizionati in modo intelligente, tonnellate di segreti e molte situazioni in cui sarà necessario pensare di non essere massacrati da demoni.

La parola chiave quando si parla di design di alto livello è "equilibrio". Il conteggio dei nemici è alto, ma non tanto da fare ripetitivi scontri a fuoco; le trappole sono tante, ma lasciano abbastanza spazio al giocatore per reagire e affrontarle; le munizioni sono limitate, ma sufficienti per fare l'atto e, infine, mentre parte del divertimento sta cercando di trovare una chiave per aprire una porta o un interruttore per abbassare un ascensore, questo viene gestito in modo tale da non sentirti mai noioso, specialmente poiché ogni volta che viene raccolta una chiave, alcune aree verranno ripopolate.

I fan della saga nel corso degli anni hanno realizzato centinaia di mappe, mod, livelli aggiuntivi e sistemi per rendere questo prodotto eterno e alcuni di questi prodotti sono di incredibile fattura come Brutal Doom da cui è condiviso il riavvio della serie.

Era solo adrenalina! Nessuna salute rigenerativa, nessuna guida su dove andare dopo, nessuna intelligenza artificiale con script e nessun sistema di copertura specifico inoltre, beh ... nascondersi dietro qualcosa.

Corri come un matto tra i livelli, abbatti demone dopo demone, raccogli oggetti utili.

Questo era DOOM e questo è di nuovo DOOM.

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 95%
Immediato, intuitivo, veloce, frenetico. Molte armi, livelli e bonus nascosti. Ti lascia giocare e ti fa giocare come un matto.

» Longevità 95%
Molti credono che Doom sia solo una reliquia del passato; una reliquia importante, ovviamente, ma niente di più. Falso! La creazione di Id Software non è solo pionieristica, ma anche uno dei migliori videogiochi mai realizzati; un lavoro geniale di un'azienda la cui mentalità era, praticamente, abbastanza simile a quella degli sviluppatori indipendenti di oggi: un gioco di giocatori per giocatori. È bello, è veloce, è profondamente avvincente, immenso nei simboli e superbo in multiplayer, è fantastico...
É Doom, ed è qui per restare finché l'inferno non prende fuoco!







NINTENDO VS TAKESHI

Anno: 1986
Sviluppatore: Taito

Genere: Platform
Piattaforma: Nintendo FamiCom

Benvenuti in questa nuova puntata sul Giappone, secondo il nostro "**Halloween-calendario**" interno è la seconda parte della **666^ puntata** che corrisponde ufficialmente alla **14^ puntata** sul Giappone presente in RMW26! Buona lettura, quando avrete finito di leggere l'articolo, spero che sarete concordi con la mia più totale convinzione che la recensione di questo videogioco sia particolarmente attinente a questo numero speciale!

Parliamo di un videogioco ispirato, anzi creato, dalla frizzante mente del famoso "Beat Takeshi" (two beat: nel senso del ritmo della comicità poichè agli albori si esibiva in duetti comici). Takeshi Kitano è uno dei personaggi del cinema, più famosi a livello mondiale, soprattutto in Giappone. E' un personaggio pubblico molto autorevole, è un attore, regista, sceneggiatore, montatore, produttore cinematografico, conduttore televisivo, autore televisivo, conduttore radiofonico, comico, scrittore, pittore, cantante e... autore di videogiochi.
È considerato uno dei più importanti registi orientali viventi per l'inconfondibilità del suo stile, la radicalità e la forza innovativa del suo cinema. Spesso i film sono incentrati sulla mafia giapponese (Yakuza), non sempre apprezzati nè presi troppo sul serio.

Vista la sua fama era obbligatorio creare un videogioco dedicato al suo interessante personaggio.
Ho detto videogioco? Oppure un incubo ad occhi aperti? Ad ogni modo è presente nella top ten dei peggiori giochi giapponesi.

Il gioco difatti ha vinto addirittura il primo premio come peggior gioco per Nintendo FamiCom.

Alcune parti del gioco si riconducono alla filosofia del suo famoso programma televisivo "Takeshi's Challenge" o "Takeshi's Castle" conosciuto in Italia con il nome di "Mai dire Banzai", commentato dalla grandiosa "Gialappa's Band".

https://youtu.be/kyLzDuAkqn8

https://youtu.be/8t2T4jGI6G0

https://youtu.be/tsuj4XkadUg

Il programma era animato da spericolati nipponici che si prodigavano in sfide impossibili-comiche-pericolose: l'obiettivo finale era conquistare l'improbabile castello virtuale di Takeshi. Questa serie fu un successo, ma il videogioco ispirato alle opere di Takeshi fu proprio problematico.
Ritorniamo al videogioco. Agli albori di questa storia, Taito ricevette la licenza per creare un videogioco dedicato all'identità di Takeshi.

Il problema è che Takeshi fu coinvolto in maniera esagerata nel processo di elaborazione del gioco, senza lasciare spazio all'esperienza dei videogame maker: scelta molto pericolosa!

Si narra che Takeshi abbia creato tutta la trama del gioco in una notte, al bar, i famosi bar Izakaya, dove mille piattini colorati e bicchierini dalle surreali forme e contenuti si mescolano senza pietà all'interno dello stomaco, provocando bizzarre ed imprevedibili alchimie digestive nonchè psicosomatiche.

I produttori Taito nel bar registrarono tutto ciò che Takeshi proponeva. Takeshi voleva un gioco completamente diverso dagli altri dell'epoca: riuscì perfettamente nell'intento, purtroppo nel senso negativo della frase.

La presenza di una celebrità così famosa avrebbe sicuramente fatto vendere infinite copie di quel







videogioco, a prescindere dalla qualità del gioco stesso. Vi assicuro che lo possiedo e che amo il retrogaming, allo stesso tempo, purtroppo, anche io provo ribrezzo nei confronti di questa sfortunata opera. Ricordiamo che Takeshi odiava l'elettronica, telefonini, email e le sue idee si miscelavano sempre in un bizzarro connubio di tradizione e follia. Takeshi ha realmente creato una cartuccia imbarazzante per Nintendo FamiCom.

Oltretutto, durante lo sviluppo del gioco, Takeshi fu coinvolto in uno scandalo extraconiugale e finì sui giornali.
Il direttore del giornale che pubblicò la notizia fu addirittura picchiato da Takeshi in persona (assieme ai suoi seguaci) durante un violento raptus di follia: da quel giorno il grande Takeshi che apparteneva allo storico duo comico "Beat Takeshi" si arricchì della qualifica di "Beat Takeshi" nel senso di "to beat up": picchiare.

Lontano dalla televisione per qualche mese, ovviamente ritornò più famoso di prima e completò i suoi lavori lasciati in sospeso, programmando i numerosi impegni futuri.

Il gioco consiste in un platform molto semplice, con svariati elementi e bonus da scoprire. Sommariamente è un platform dove si devono assumere le classiche decisioni per avanzare di livello. La musica è frustrante, un loop senza armonia, per nulla orecchiabile. La traduzione in inglese, fortunatamente, è ottima.

https://youtu.be/j_RH518LyOk

Le particolarità del gioco sono tantissime, non riuscirò ad elencarle tutte visto che ci ho giocato tre volte e che sicuramente non rigiocherò in futuro a questo titolo:

1) se nello schermo dei titoli si tirano pugni in aria per 20,000 volte si arriva direttamente alla fine del gioco
2) ci sono livelli dove bisogna attaccare il secondo pad dotato di microfono e cantare come nelle sale Karaoke
3) possiamo divorziare e/o picchiare a sangue sia moglie che figli (ricordiamo che il gioco è ufficialmente rilasciato da Nintendo!)
4) possiamo picchiare il nostro datore di lavoro
5) possiamo picchiare la polizia e allo stesso tempo essere picchiati da essa, perdendo energia
6) possiamo picchiare tutti come pazzi scatenati ed essere odiati da tutti e soprattutto essere picchiati dalla gente che ci odia
7) ci sono situazioni porno, si può bere whisky e giocare a Pachinko
8) dentro la sala Pachinko potremo avanzare nei livelli successivi unicamente se uccideremo le persone che ci stanno picchiando
9) potremo pilotare dei velivoli, addirittura un aeroplano, che però non potrà atterrare e quindi ci schianteremo al suolo inutilmente, con conseguente game over
10) la conclusione del gioco è rappresentata da un piccolo disegnino con la faccia di Takeshi che dice "congratulazioni"
11) ulteriori numerose stranezze che non voglio assolutamente ricordare nè sapere nè descrivere...

Sembra proprio un opera tragicomica, violenta ed assurda, creata da una persona che odia i videogiochi: una logica tipica di Takeshi che realmente odia l'elettronica ed i videogiochi.
Forse questa logica è stata realmente voluta da Takeshi, con macabra coscienza.
Nessuno potrà mai saperlo. Nessuno chiederà spiegazioni a Takeshi. Pochi si ricorderanno di questo problematico gioco. Penso che Nintendo non sia neppure felice della presenza di questo articolo, pazienza, cercherò di farmi perdonare con una lettera di scuse!

Anche i creatori del "Strategy Guide" per completare il gioco hanno sudato freddo, affrontando un gioco così assurdo, irrazionale ed illogico.

Se troverete questa cartuccia nei negozi dell'usato di Tokyo, dovete assolutamente prenderla, è una mostruosità da collezione, con la conseguente mostruosa logica dei prezzi del collezionismo. Il manuale purtroppo è ancora più raro, neppure io lo possiedo.

Bene cari lettori, spero proprio che questa recensione abbia rispettato l'anima "horror" di Halloween, arrivederci al prossimo numero, vi prometto che torneremo nella normalità parlando della follia nipponica!

di **Michele conte Ugolino**

GIUDIZIO FINALE

» Giocabilità 1%
E' un'esperienza traumatizzante!

» Longevità 1%
Non vedrete l'ora di spegnere il FamiCom!





STORMLORD

Anno: 1989
Sviluppatore: Nick Jones, Raffaele Cecco, Huges Binns

Piattaforma: Commodore 64
Genere: Platform

Nuovo autunno e anche se molto diverso e particolare, nuovo Halloween in arrivo con alcune limitazioni dell'ultim'ora e nuova recensione di un gioco a tema di quest'ultima festa.
Festa che forse è una delle poche a far sentire lo spirito e la goliardia, dato che ogni anno, salvo pioggia torrenziale, intrattiene molti europei ed americani e soprattutto noi videogiocatori grazie a titoli horror che già all'epoca spopolavano sul biscottone. Ne avevo giocati e provati parecchi ma quello che mi colpì particolarmente, e in particolar modo per la colonna sonora che mi faceva paura, fu Stormlord.
Anche questo scoperto per puro caso, ammesso che il caso esista su una cassetta da edicola, di cui non mi ricordo il nome in italiano (però quello in inglese sì!).

Dopo la schermata iniziale, il gioco si presenta come un platform a scorrimento, in cui impersonerete un mago guerriero in età avanzata che salta, spara fiamme e si teletrasporta (e tutto ciò a dispetto dei suoi anni).

Lo scopo del gioco, nonché quello di superare ciascuno dei quattro tosti livelli, è di liberare le fate imprigionate in vari punti, risolvendo degli enigmi con i giusti oggetti; per esempio la chiave per aprire le porte, il miele per spostare lo sciame d'api ecc...
Nel mentre decine e decine di creature demoniache sono pronte a sbarrarvi la strada e rendere il gioco ancora più difficile.

Per fortuna avremo a disposizione ben otto vite sin dall'inizio per completare l'intero gioco, anche se, ripeto, non sarà una passeggiata, come la maggior parte dei giochi dell'epoca (scusate se sono un po' ripetitivo).
Alla fine di ciascun livello ci sarà un bonus che vi farà guadagnare parecchi punti e forse anche qualche vita extra.
Sta a voi scoprire di che tipo di bonus si tratta e come racimolare punti e vite per poi passare al livello successivo e liberare altre fatine: fatine che crescono numericamente di livello in livello.

Portare a termine il vostro compito non sará facilissimo, ricordo perfettamente quegli odiosi vermiciattoli che uscivano dal terreno alla velocità della luce e che mi facevano perdere un terzo delle vite a disposizione ed anche qualche enigma risolto in maniera sbagliata, che mi costringeva a ripartire dalla schermata dei titoli.

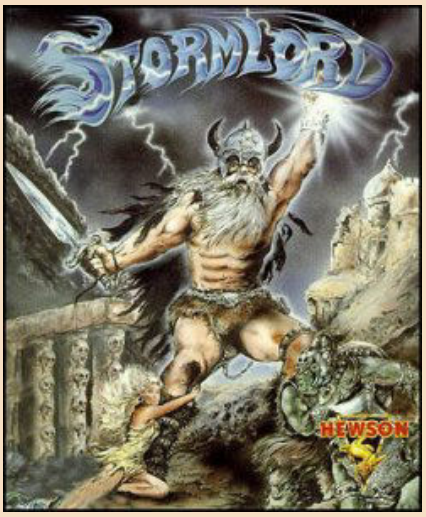
Il gioco uscì su diverse piattaforme e giocando le varie versioni ho potuto fare un paragone dei fondali e delle musiche; sono i due elementi che più caratterizzano questo gioco e come sempre la musica della versione del biscottone non è seconda a nessuno!

Stormlord ebbe anche un seguito molto simile al primo ma con lo scorrimento lineare e boss di fine livello; forse non ebbe lo stesso successo del primo ma valeva comunque il nostro tempo e i nostri joystick.

Se la notte delle streghe sarete costretti a rispettare il coprifuoco, procuratevi questo gioco oltre ai tanti altri horror belli e brutti che hanno fatto la storia sul C64 e mi raccomando, spegnete le luci!

Buona festa delle ombre

di **Daniele Brahimi**





Versione Commodore 64



Versione Amiga

GIUDIZIO FINALE

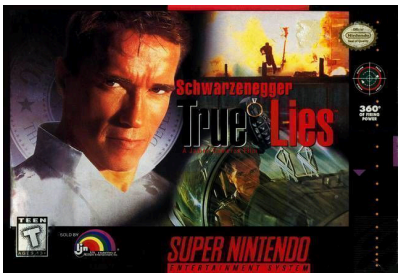
» Giocabilità 60%
Si muore facilmente, ma una volta presa la mano...

» Longevità 70%
Solo quattro livelli? Giocateci, poi mi direte!

TRUE LIES

Anno: 1994
Publisher: Acclaim
Piattaforma: Snes
Genere: Platform

Nella mia ultima recensione ho parlato di una rinata passione per le console a 16 bit e perciò voglio presentarvi ancora un titolo per Supernintendo scoperto un po' per caso: sto parlando di **True Lies.**



Il gioco è stato pubblicato dalla Acclaim Entertainment nel 1994, in concomitanza con l'uscita dell'omonimo film interpretato da Arnold Schwarzenegger.

Infatti il nostro eroe Harry Tasker ha le sembianze del mitico attore americano e la nostra avventura segue, livello per livello, la trama della pellicola.

Dal punto di vista grafico si tratta di un gioco top down, con la classica visuale isometrica dall'alto che ricorda tanto The Chaos Engine e Shadowrun.

All'inizio di ogni missione siamo armati soltanto di una pistola ma nel corso della nostra avventura possiamo raccogliere anche un fucile a pompa, una mitragliatrice, un lanciafiamme, delle granate e mine antiuomo e possiamo comodamente passare da un'arma all'altra con un semplice tasto.

Oltre alle armi possiamo anche raccogliere munizioni, chiavi per aprire passaggi nascosti e cassette del pronto soccorso per ricaricare la nostra energia.

Con un altro tasto invece possiamo compiere un salto in avanti per sfuggire ai proiettili o per ritrovarci faccia a faccia con il nemico.

Durante il nostro cammino è facile imbattersi in civili che spesso si trovano sulla nostra linea di fuoco e che dobbiamo assolutamente cercare di non colpire. Infatti dopo tre civili uccisi perdiamo una vita.

La grafica è ben curata con sprites colorati e ambientazioni sempre nuove e varie. Stupende sono anche le tante esplosioni.

I livelli sono molto lunghi, profondi e ricchi di dettagli e forse, essendo ben nove più uno bonus, a lungo andare potrebbero risultare un po' ripetitivi.

Niente da dire neanche sul comparto sonoro con musiche orecchiabili e diverse per ogni livello ed effetti sempre appropriati.

Ad arricchire il tutto ci sono screenshots del film, naturalmente pixellati, che introducono il tema di ogni missione.

In conclusione, se non conoscete questo titolo vi consiglio vivamente di riscoprirlo. Muovere il nostro Schwarzy tra proiettili, esplosioni e imboscate varie è davvero divertente e ci ritroviamo immersi in un'esperienza di gioco che non è stata replicata in altri titoli di questo genere per il supernintendo.

Querino Ialongo





GIUDIZIO FINALE

» Giocabilità 90%

Rispetto alla conversione megadrive, questa per super nintendo ha comandi che rispondono in maniera impeccabile, offrendo un'esperianza di gioco davvero divertente.

» Longevità 80%

Ci vogliono diverse ore per terminare questo titolo e forse nove livelli, più uno bonus, sono un pò troppo e per qualcuno potrebbero risultare ripetitivi.

100.000 Volte Grazie!

Qualche giorno fa abbiamo fatto sulla nostra pagina ufficiale di Facebook un annuncio per **celebrare i 100.000 download di tutte le nostre pubblicazioni**.

Purtroppo i post su Facebook sono piuttosto effimeri e spariscono velocemente, fagocitati dagli altri post e dalla frenesia con la quale ormai ci approcciamo a tutte le notizie che giornalmente vediamo. Insieme a tutta la redazione abbiamo quindi ritenuto doveroso soffermarci nuovamente su questo argomento, e ringraziare nuovamente i nostri letttori!

Chi ci segue ormai da tempo sa che non amiamo sbandierare numeri e statistiche, anche perché per noi sono piú importanti l'aspetto umano e le emozioni che possiamo suscitare con i nostri articoli ed i feedback o i complimenti ricevuti per un pezzo particolarmente apprezzato.

Ma questa volta mentirei se dicessi che questo numero non abbia generato in noi nessuna sensazione. Questo numero, bello, tondo, altisonante, ha risvegliato il nostro ego e... Volevamo festeggiarlo!

C'e' inoltre da notare che questo numero é soltanto una parte, sicuramente la piú grande, del totale assoluto, visto che alcuni siti fanno il mirroring dei nostri numeri e visto che ne abbiamo trovati alcuni anche su archive.org. Questo é sicuramente un altro punto di orgoglio per tutti noi della redazione.

Ma adesso basta parlare dei traguardi raggiunti, anche perché, una volta raggiunti fanno giá parte del passato, mentre noi, computer a parte :-D, siamo sempre orientati al futuro!

Durante l'ultima riunione interna della redazione, abbiamo discusso molti punti che vorremmo vedere completati in un futuro prossimo. Ci sono tanti progetti che speriamo di poter annunciare molto presto, ma quello che vorremmo vedere sempre di piú é la partecipazione attiva dei nostri lettori con i loro contributi.

Mai come adesso ci sono state cosí tante persone che lavorano a progetti dedicati al retrocomputing. Periferiche hardware, emulatori, giochi, programmi... vengono rilasciati quasi giornalmente ed é difficile stare dietro a tutto considerate la quantitá e la velocitá alla quale vengono presentati.

Se anche tu, che stai leggendo, fai parte di uno di questi progetti, contattaci. Vorremmo tanto dare voce e visibilitá a tutti.

Come abbiamo sempre detto RMW é la Vostra rivista!

Francesco Fiorentini

Disclaimer

RetroMagazine World (fanzine aperiodica) è un progetto interamente no profit e fuori da qualsiasi circuito commerciale. Tutto il materiale contenuto è prodotto dai rispettivi autori e pubblicato grazie alla loro autorizzazione.

RetroMagazine World viene concessa al pubblico con licenza: Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale (CC BY-NC-SA 4.0 INT) https://creativecommons.org/licenses/by-nc-sa/4.0/deed.it

In pratica sei libero di: condividere, riprodurre, distribuire, comunicare o esporre in pubblico, rappresentare, eseguire e recitare questo materiale con qualsiasi mezzo e formato, modificare, rielaborare, trasformare il contenuto e basarti su di esso per altre opere, alle seguenti condizioni:

Attribuzione
Devi riconoscere una menzione di paternità adeguata, fornire un link alla licenza e indicare se sono state effettuate delle modifiche. Puoi farlo in qualsiasi maniera ragionevole possibile, ma non con modalità tali da suggerire che il licenziante avalli te o l'utilizzo del materiale da parte tua.

NonCommerciale
Non puoi utilizzare il materiale per scopi commerciali.

StessaLicenza
Se rielabori, trasformi il materiale o ti basi su di esso, devi distribuire i tuoi contributi con la stessa licenza del materiale originario.

Il licenziante non può revocare questi diritti fintanto che tu rispetti i termini della licenza.

Divieto di restrizioni aggiuntive
Non puoi applicare termini legali o misure tecnologiche che impongano ad altri soggetti dei vincoli giuridici su quanto la licenza consente loro di fare.

RetroMagazine World
Anno 4 - Numero 26 - NOVEMBRE 2020

Direttore Responsabile
Francesco Fiorentini
Vice Direttore
Marco Pistorio
Coordinatore Redattori
David La Monaca
Responsabile Area Web
Giorgio Balestrieri