RetroMagazine World

future days are back

Numero 25 Anno 4 - Settembre 2020 - WWW.RETROMAGAZINE.NET - Pubblicazione gratuita

FORTH: l'arma segreta!

GOLDEN AXE WARRIOR
(Sega Master System)

Intervista a Giuseppe Ettore Pintus
creatore di Freedom Fighter - Rise of the humans!

SENSIBLE WORLD OF SOCCER 2020
(PC / AMIGA)

Alien Attack!
un nuovo gioco in Locomotive Basic

- RetroMath: Come ti trasformo un'immagine...
- L'estensione MiniGrafik per il BASIC del Vic-20
- Cross-programmazione su Olivetti M20 in C
- Introduzione ad ARexx – quinta parte

Giappone 13a puntata:
Nintendo G&W, sfida all'immortalità

RUFF 'N' TUMBLE (Amiga - CD32)

Il motore del retrocomputing

La passione che tutti noi condividiamo e che chiamiamo retrocomputing (includendovi forse non totalmente a ragione il termine retrogaming), come ogni altra passione, ci porta ad impiegare molto tempo nella ricerca di hardware vintage, giochi e programmi originali, accessori e periferiche d'epoca e moderni, materiale bibliografico, libri e riviste.

Eppure, guidati da questa nostra profonda passione, talvolta impieghiamo risorse preziose quanto il tempo per raggiungere ciò che, personalmente o collettivamente, consideriamo il sacro Graal del retrocomputing. Può trattarsi di un particolare disk drive, raro perché in passato pochi pezzi sono stati immessi sul mercato, oppure di una particolare versione di un integrato, magari in ceramica, oppure ancora di un home computer recante un basso numero di serie sulla targhetta originale del produttore. Qualunque sia l'oggetto del desiderio, come per molte altre forme di collezionismo, dall'amatore più moderato fino all'accumulatore seriale, fatto sta che ci scopriamo spesso disposti ad aprire il portafogli e spendere cifre elevatissime, paradossalmente anche più alte di quanto fossero quando i pezzi che inseguiamo uscirono sul mercato. Per fare un esempio, il prezzo di lancio in USA e in Europa di un C64 "biscottone" costava circa mezzo stipendio di vostro padre o anche di più, ve lo ricordate?* E oggi, nell'estate del 2020, dopo quasi 40 anni e nonostante i 17 milioni di esemplari venduti nel mondo, c'è chi non solo chiede ma ottiene cifre intorno ai 250-300 euro per lo stesso computer, se ben tenuto, funzionante e magari con la scatola originale. Basta effettuare l'operazione di cambio da EUR in valuta nazionale per ritrovarsi di fronte ad un valore simile a quello del prezzo di vendita di lancio. E se parlassimo di altri modelli Commodore (Amiga) o di altri marchi meno venduti sul mercato di allora e di conseguenza meno facili da trovare oggi, i prezzi dell'usato (non garantito) supererebbero di gran lunga il prezzo originale di vendita al dettaglio.

Stiamo parlando di un fenomeno di prezzi al rialzo che si è concretizzato soprattutto negli ultimi dieci anni, con una tendenza di crescita lenta ma continua. Naturalmente questi rincari generalizzati attirano anche gli "investitori", incentivati a comprare per rivendere in un secondo momento, gli speculatori e persino i truffatori, che sfruttando la debolezza degli appassionati veri e le carenze di garanzia dei sistemi di vendita on-line, contribuiscono in maniera decisiva a far sì che l'incontro fra offerta e domanda si possa incontrare soltanto sempre più in alto sulla scala dei prezzi. Fortunatamente resistono alcune nicchie di veri appassionati cui sta a cuore che il proprio hobby non muoia sotto i colpi delle cieche tendenze del mercato e che ancora praticano lo scambio diretto di hardware, senza coinvolgere pagamenti in denaro. Fortunatamente la maggior parte degli emulatori software per i nostri amati sistemi sono disponibili in forma gratuita quando addirittura non in forma open-source. Fortunatamente anche i nuovi prodotti hardware (per lo più basati su circuiti FPGA) che s'ispirano o riproducono le vecchie macchine giungono sul mercato a prezzi abbastanza abbordabili pur offrendo un'esperienza non dissimile a quella dei sistemi originali. Altrimenti, nonostante il sempre crescente numero di persone che si (ri)avvicina al mondo dei sistemi a 8 e 16 bit, siano essi home e micro computer o console per videogiochi, questa nostra passione di raccogliere, utilizzare, rimettere le mani sull'hardware originale e salvarlo dall'oblio, resterebbe appannaggio di pochi "benestanti" e col tempo sparirebbe del tutto.

Perché è l'energia della passione che tutti noi condividiamo che tiene in vita il retrocomputing, non i soldi.

David La Monaca

(*) USD 595, LIT 973.500, DM 1495: pari a circa USD 1'576 nel 2019 – fonte Wikipedia

SOMMARIO

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

- Alberto Apostolo
- Giuseppe Fedele
- Michael Jean
- David La Monaca
- Gianluca Girelli
- Davide Bucci
- Giorgio Balestrieri
- Michele Ugolini
- Carlo N. Del Mar Pirazzini
- Federico Gori
- Marco Pistorio
- Querino Ialongo
- Daniele Brahimi
- Flavio Soldani
- Francesco Fiorentini
- Supporto grafico **Irene G. Valeri**
- Copertina a cura di **Flavio Soldani**

Intervista a Giuseppe Ettore Pintus

sviluppatore di Freedom Fighter - Rise of the humans!

di Francesco Fiorentini

Da qualche settimana a questa parte, gli utenti MSX hanno potuto mettere le mani su un nuovo gioco, che ha colpito tutti per il suo scrolling estremamente fluido e per la sua difficoltà. Stiamo parlando ovviamente di Freedom Fighter - Rise of the humans!, creato dal talentuoso Giuseppe Ettore Pintus.
Il gioco, che partecipa all'edizione 2020 dell'MSXDev, ha tutte le carte in regola per essere uno dei protagonisti assoluti di questa edizione. Il suo programmatore e' un frequentatore di molti gruppi retro italiani e noi di RMW non potevano farci sfuggire l'opportunita' di fargli qualche domanda. Ne è venuta fuori una bella intervista, piena di spunti interessanti e da leggere tutto d'un fiato!

Ciao Giuseppe e grazie di aver accettato il nostro invito per questa intervista nelle pagine di Retro Magazine World. Prima di cominciare a parlare del tuo gioco 'Freedom Fighter - Rise of the humans', cominciamo con qualche domanda di rito per conoscere l'uomo prima che il programmatore.

Vuoi raccontarci qualcosa di te? Chi è Giuseppe Ettore Pintus e cosa fa nella vita?

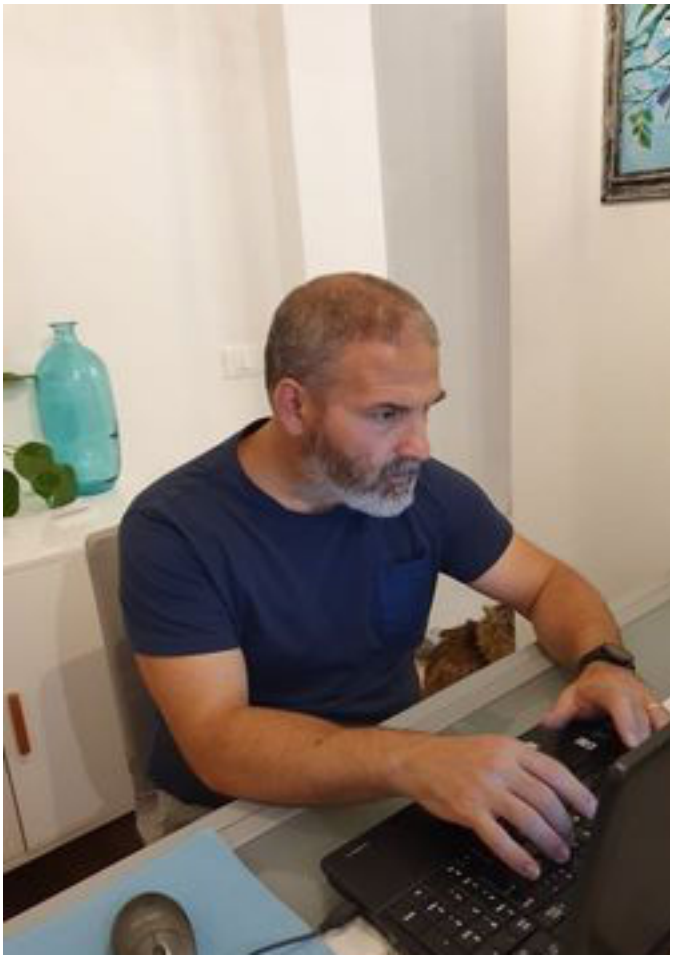


Figura 1 - Giuseppe Ettore Pintus all'opera sul suo gioco

Ciao a tutti e grazie a voi per l'invito. Giuseppe nella vita è prima di tutto un marito: se avete fra le mani Freedom Fighter lo dovete alla pazienza di mia moglie, che ha supportato (o sopportato?) la mia passione e il mio progetto. Sono elettricista di professione (un po' riduttiva come descrizione, perché non mi occupo solo di impianti elettrici, ma più o meno rende l'idea).

Il gioco che hai realizzato è per lo standard MSX, un computer che in Italia non ebbe il successo del Commodore 64 e dello Spectrum. Per quale ragione ti sei avvicinato a questa macchina?

L'MSX è stato il computer con cui sono cresciuto, unico in mezzo a tanti amici sessantaquattristi e qualche possessore di Amstrad CPC (ai tempi dalle nostre parti andava forte quello distribuito dalla Schneider insieme ai corsi di informatica). Ho avuto qualche "scambio di cassette" con compagni di scuola di amici ma poi conobbi di persona solo un paio di msxisti.

Nel caso l'MSX non sia stato il tuo primo computer, qual è stato e che ricordi ne hai?

Il primo computer ad entrare in casa è stato il Commodore Vic20. Posseggo e amo tuttora l'originale della mia infanzia! La passione per la programmazione è nata con lui. Arrivò a casa il giorno del mio decimo compleanno. Arrivò "nudo e crudo". Senza giochi, senza datassette. Per cui ogni giorno digitavo uno (ma a volte tutti) dei tre listati in fondo al manuale (erano tre giochi in basic di Duane Later). Quanti syntax error in quel periodo! Poi all'ora di cena spegnevo il Vic e non potendo registrare sapevo che avrei dovuto rifare tutto il giorno dopo!

Come hai conosciuto il mondo della programmazione? Sei un autodidatta, come molti di noi negli anni 80/90, oppure hai seguito qualche corso di formazione particolare?

Assolutamente autodidatta. Come ho detto prima tutto è iniziato col Vic. A furia di digitare, leggere e rileggere il manuale (e tutti gli esempi contenuti) ho preso dimestichezza. Poi col tempo comprammo il datassette e le prime riviste. Copiavo e studiavo i listati. Quando misi da parte il Vic-20 per soppiantarlo con l'MSX avevo all'attivo tre/quattro giochi in BASIC ideati e programmati da zero da me: un clone di Breakout, un gioco di corse (simil ten-liner), un platform alla PacLand (a cambio di schermo) e un gioco di corse simile agli schiacciapensieri

della GiG. Proseguii con il BASIC MSX (e comprai il libro sull'assembly che non misi mai a frutto in passato perché non trovai da nessuna parte un programma assembler!) e poi passai ad Amiga. Feci parecchie cosette in Amos, tra cui un motore di gioco completo per RPG simil Eye of the Beholder (ma più fluido della versione Amiga, più simile a Black Crypt per intenderci)

Qual è il tuo retro-linguaggio di programmazione preferito e per quale ragione?

Al momento l'assembly Z80, ci ho programmato il mio primo gioco ufficiale e internazionale!

Quali altri computer ti piace programmare oltre all'MSX?

Tutti quelli che ho menzionato prima (mi son fatto prendere dalla foga).

Come ti approcci al retro-computing? Possiedi molte retro-macchine o preferisci emulatori e/o FPGA?

Al momento possiedo un VCS2600 Jr (loose ed in riparazione), un Commodore Vic20 boxato, 3 MSX VG8020 (dei cui uno in riparazione, il primo che ho avuto) un Amiga 500+ (il mio, anche questo in riparazione causa maledizione Varta...) e recentemente ho acquistato un MSX2 VG8235 (il sogno che avevo prima dell'Amiga). Uso per praticità molto gli emulatori, soprattutto per testare il sofware velocemente...

Parliamo un po' di 'Freedom Fighter - Rise of the humans', uno shoot'em up a scorrimento verticale, come mai hai scelto questo format per il tuo gioco?

Eh! In realtà io volevo solo programmare uno scrolling verticale fluido per dimostrare che l'MSX può farlo (e che io sono in grado di farlo). Ma una volta pronta la versione quasi definitiva della routine di scrolling, dopo averla presentata sul forum di MSX Resource Center, sono stato spinto dagli altri coder ad andare avanti. Non sapevo nemmeno da che parte cominciare e che cross assembler

Figura 2 - Schermata iniziale di Freedom Fighter

utilizzare: sono partito usando un assembler scritto in basic su MSX emulato, che fornivano nella prima cassetta della rivista C16/MSX in edicola...

Da quali giochi hai tratto ispirazione per creare Freedom Fighter?

Sicuramente il primo che mi viene in mente è Zanac, della Compile. Una pietra miliare negli shoot'em up per MSX. Infatti in molti durante lo sviluppo hanno accomunao i due giochi. Ho reso omaggio a Zanac nella musica del primo livello (derivata da quella originale) e in uno dei tre segreti del gioco. Sbloccando il secondo segreto, infatti, si può cambiare la propria astronave con una di quelle dei più famosi shoot'em up per MSX: Zanac, Twinbee, Star Soldier, Star Force e Hype.

Quali difficoltà hai incontrato durante la scrittura del codice?

Debuggare un progetto assembly da solo, soprattutto se è il tuo primo progetto, se hai imparato tutto strada facendo e se sei partito direttamente con un ambizioso progetto megarom (1 Megabit, cioè 128KB di gioco su cartuccia) non è affatto una passeggiata. Per il resto, quando avevo qualche dubbio avevo a disposizione un'intera community di coder a cui chiedere consiglio!

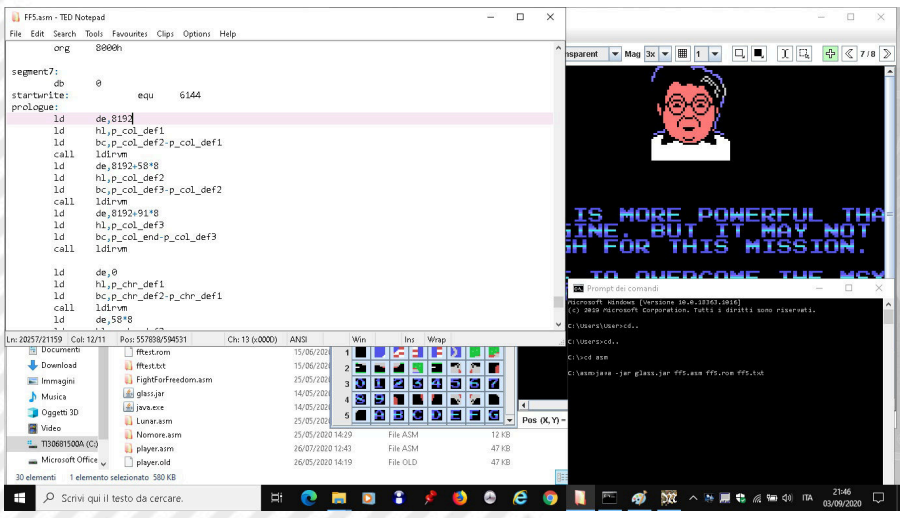


Figura 3 - l'ambiente di sviluppo di Freedom Fighter

Non sono un profondo conoscitore della programmazione MSX, ma leggo spesso di problemi di scrolling su questa macchina. Il tuo gioco inevce è fluidissimo, come sei riuscito a realizzare questa magia?

Non è che l'MSX abbia problemi di scrolling... Non ha proprio lo scrolling!

Figura 4 - il primo livello di Freedom Fighter

E' una funzione che il suo chip video (per intenderci lo stesso del TI99/4A e del Colecovision) non ha. Per cui tutto quello che scorre è puramente un trucco realizzato via software. Bisogna conoscere bene l'hardware con cui si ha a che fare

Se potessi tornare indietro ed approcciare il lavoro con "il senno di poi", cosa faresti diversamente?

Non so se cambierei qulcosa. Mi sono divertito molto a realizzarlo. Magari eviterei direttamente di accettare l'aiuto offerto da persone dubbie (che infatti sono state escluse dal progetto) e magari avrei cercato qualche grafico MSX nella community di cui sopra (anche se mi pare di essermela cavata discretamente con tutta quella grafica e il codice!). Quello che invece sono sicuro che rifarei è lavorare insieme a Phaze101 sulla parte audio. E' stata un'esperienza fantastica, credo si sia avvicinata molto a quello che doveva essere lavorare in una software house negli anni 80 (o almeno a come immaginavo io).

Ho notato che sei molto attento ai feedback che ricevi dai giocatori e che hai pubblicato diversi aggiornamenti del tuo gioco. Alcuni di questi erano mirati ad abbassare la difficoltà iniziale di Freedom Fighter. Avevi volutamente creato un gioco difficile da completarsi per rendere omaggio al format dei giochi anni 80?

Si, inizialmente ho pensato: Ehi! I giochi per gli 8-bit negli anni 80 erano spaventosamente difficili! Devo ricreare quel feeling. E poi sono solo 5 livelli, se lo faccio troppo facile finisce in un attimo! Del resto io giocavo e finivo tranquillamente il primo livello senza perdere nemmeno una vita. Ma poi ho letto che davvero TUTTI lo trovavano troppo difficile, così ho ascoltato la voce dei giocatori e ho iniziato a smussare qua e là…

Siamo arrivati al momento 'Marzulliano' dell'intervista: "Si faccia una domanda e si dia una risposta".

"Chi me lo ha fatto fare?" può andare bene come domanda? La risposta è per tutti: ragazzi (cresciuti, in effetti, dato che siete amanti di retrocomputing), le passioni vanno seguite. Ci ho messo due anni a sviluppare il gioco. Ma era il mio primo gioco. L'assembly era territorio sconosciuto per me ma volevo davvero farlo. Fatelo anche voi! Se volete programmare e non sapete farlo dovete semplicemente iniziare. Si possono avere tante soddisfazioni anche dal lentissimo BASIC. Si impara a ragionare da programmatori, a risolvere problemi, a sfruttare l'hardware. Con le attuali "conoscenze diffuse", tra coder a cui chiedere e libri del passato da scaricare in pdf, manuali e tutto ciò che una volta non c'era potete davvero farcela. Fatelo! Potete anche iscrivervi al nostro gruppo Facebook, Retro Programmers Inside, qui:

https://www.facebook.com/groups/RetroProgrammersInside/?ref=share

Prima di lasciarti, vorrei ringraziarti nuovamente per la tua disponibilità e soprattutto per avere rilasciato il tuo gioco con licenza freeware.
In bocca al lupo per la tua partecipazione all'MSXDev 2020!

Grazie a voi e viva il lupo!

Vi ricordo che potete giocare a Freedom Fighter online qui: https://www.file-hunter.com/MSXdev/index.php?id=freedomfighter

Oppure scaricarlo gratuitamente da qui: https://download.file-hunter.com/Games/MSXdev/

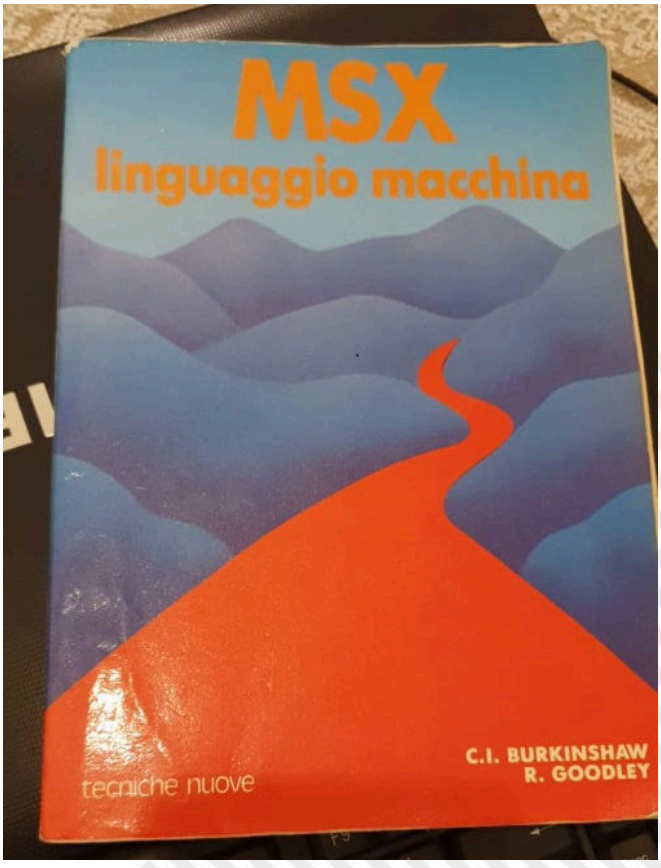


Figura 5 - il libro sull'Assembly dell'MSX

Sproteggere i programmi scritti in GW-BASIC

di Alberto Apostolo

Leggendo il brillante articolo di Giuseppe Fedele sulla Crittografia (RMW 23 ita, RMW 01 eng), mi sono ricordato di quando (nel 1998) fui incaricato di effettuare il porting di alcuni vecchi programmi per conto di una azienda di Pordenone che produceva macchine tessili.

I programmi erano stati scritti in GW-BASIC da un dipendente di quell'azienda ma i listati cartacei erano andati perduti e il salvataggio con l'opzione P di protezione rendeva i file sorgente illeggibili.

All'epoca non ero riuscito a trovare documentazione su come effettuare la sprotezione (essendo stato autorizzato dai proprietari del software, operavo in modo assolutamente legale). Potevo solo fare il "dump" dei file sorgente con il buon vecchio PCTOOLS.EXE sul mio 386 per cercare di capire quale fosse la codifica utilizzata.

Soluzione con la forza bruta

Imitando i leggendari crittografi britannici di Bletchley Park, dovevo cercare di sfruttare un punto debole di molte codifiche e osservare il comportamento quando si ha a che fare con una stringa molto lunga formata dallo stesso carattere.

Come primo caso di test, provai a salvare in modo protetto un semplice programma (Listato 1) e a visualizzarlo con un Hex Editor (in questo articolo ho usato il programma free HxD).

In Figura 1 si nota che il carattere "A" è codificato in modo diverso in funzione della posizione nella riga. Allo stesso tempo si nota che dopo 143 byte la codifica si ripete (un vero colpo di fortuna!).

Ripetendo lo stesso esercizio per ogni carattere si può ottenere una matrice di decodifica 256 x 143 (in Figura 2 si vede solo una piccola parte). Siccome esistono metodi assai più semplici ed eleganti, vi

```
1 REM AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
```

Listato 1

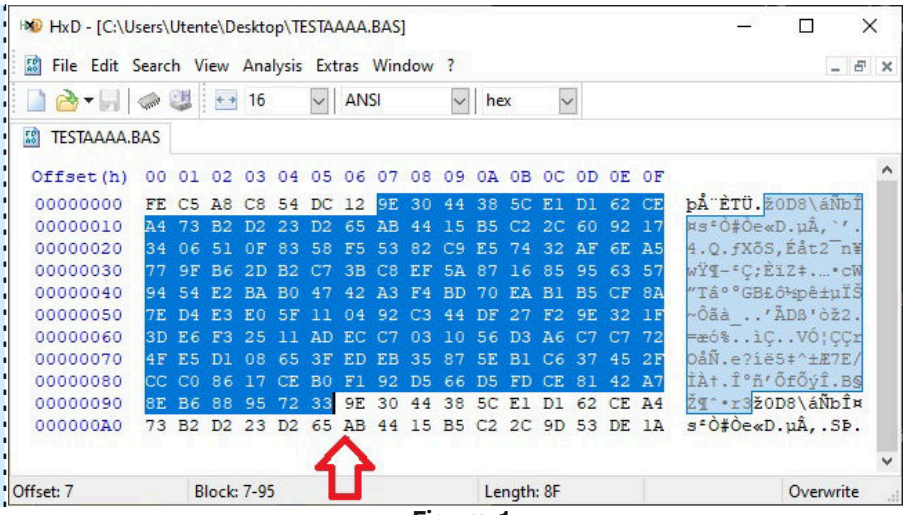


Figura 1

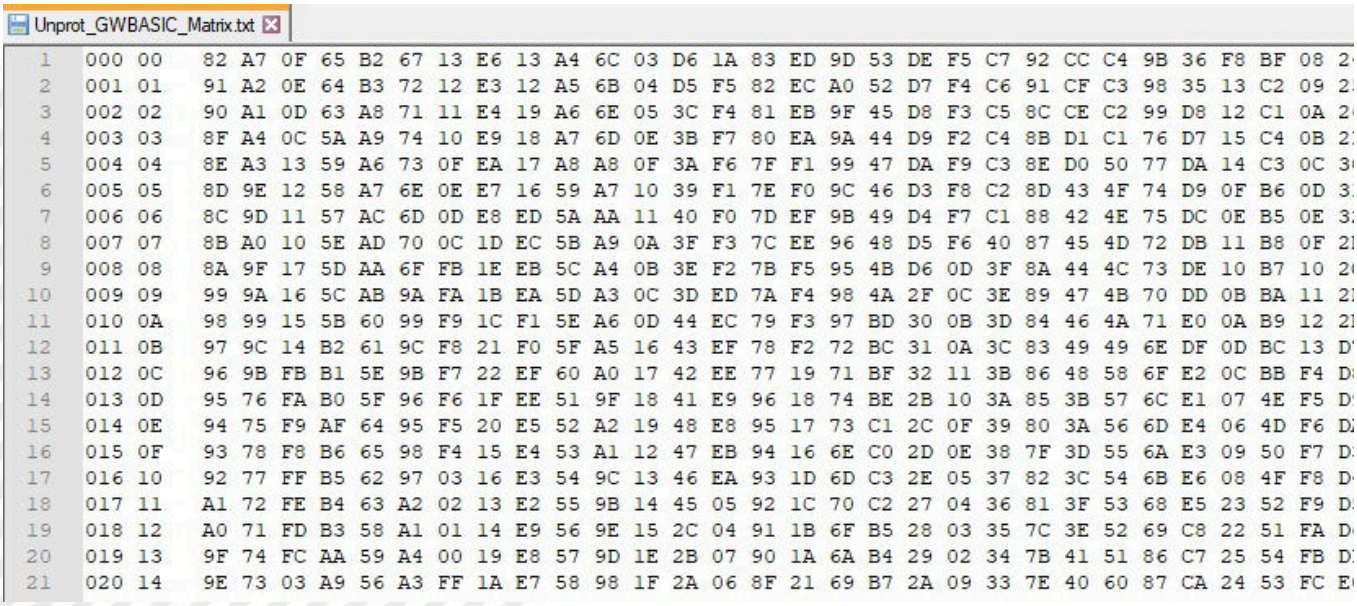


Figura 2

risparmio la pubblicazione dei programmi scritti all'epoca in COBOL che mi hanno permesso di arrivare al risultato.

Soluzione con un tocco di fioretto

Cercando su Internet al giorno d'oggi si trovano pagine web su qualsiasi argomento. Così ho potuto sperimentare un trucco semplice per sproteggere un programma "ingannando" l'interprete GW-BASIC.
Per prima cosa si deve creare un programma dummy in GW-BASIC composto da solo due byte: 0FFh e 1Ah. Nel mio caso ho utilizzato l'Hex Editor HxD (Figura 3).
Poi dall'interprete occorre eseguire la sequenza di operazioni (Figura 4):
1) caricare il programma protetto,
2) caricare il programma dummy,
3) salvare il programma (che ora sarà sprotetto).
L'inganno avviene perché il primo byte di un programma salvato vale 0FFh se il programma è sprotetto (con le parole chiave "tokenized"), 0FEh se è codificato. Il programma dummy sovrascrive soltanto il primo byte e il GWBASIC provvederà diligentemente alla decodifica delle successive istruzioni in memoria.
Questo semplice trucco ha lo svantaggio che si può applicare su un programma alla volta, attivando l'interprete GW-BASIC.

Il mistero svelato

Durante la ricerca per la stesura dell'articolo, ho trovato un interessante post scritto da Christophe Lenclud [Len18] nel quale si mostra il sistema di decifratura usato dal GW-BASIC.

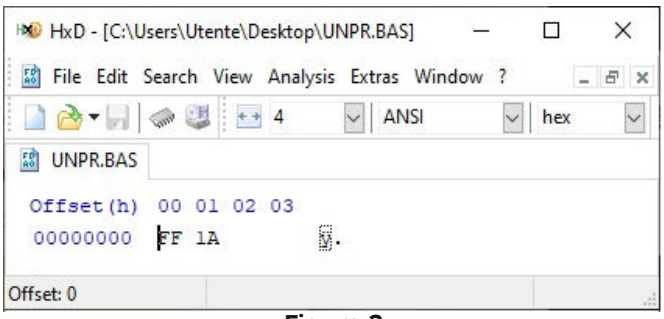


Figura 3

Si applica lo XOR tra ciascun byte del programma e due chiavi incorporate nel file GWBASIC.EXE. Le chiavi hanno lunghezza rispettivamente 13 e 11 byte e ne consegue che si ripete la cifratura ogni 13 x 11 = 143 byte (Listato 2).
In [Len18] si trova una lista (incompleta) delle versioni sulle quali è applicato questo sistema che si basa sulla ricerca delle chiavi all'interno del file eseguibile dei vari interpreti BASIC:
1) BASICA.EXE (size 54272 bytes, 13 May 1983 12:00:00, MD5 = 28E22CAA7EC534A78D37AA3314690758) from "The COMPAQ Personal Computer DOS, Version 1.11" Rev E.
2) GWBASIC.EXE (size 59728 bytes, 05 June 1984

```
; Input: DS:SI -> Data to be deciphered.
;          DS:DX -> After end of data.
Decipher_GWBASIC proc near
    mov     cx, 0D0Bh
    mov     di,si
    mov     bh,0
    cld
@@LoopDecipher:
    cmp     si,dx
    jz      short @@EndOfFile
    ; Decipher one byte...
    mov     bl,ch
    lodsb
    sub     al,cl
    xor     al, [bx + offset Key1 - 1]
    mov     bl,cl
    xor     al, [bx + offset Key2 - 1]
    add     al,ch
    stosb
    ; Next byte...
    dec     cl
    jnz     short @@NotZ1
    mov     cl,0Bh
@@NotZ1:
    dec     ch
    jnz     @@LoopDecipher
    mov     ch,0Dh
    jmp     @@LoopDecipher
@@EndOfFile:
    ret
Decipher_GWBASIC endp

Key1    db    9Ah, 0F7h, 19h, 83h,  24h, 63h,
43h, 83h,  75h, 0CDh, 8Dh, 84h, 0A9h
Key2    db    7Ch,  88h, 59h, 74h, 0E0h, 97h,
26h, 77h, 0C4h,  1Dh, 1Eh
```

Listato 2

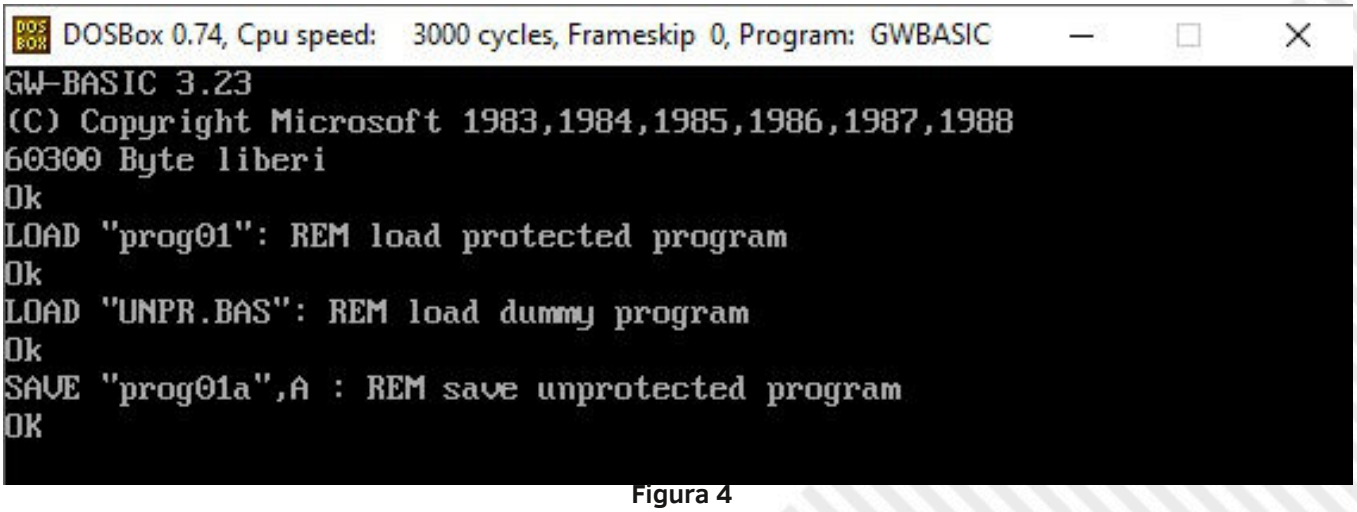


Figura 4

01:25:00, MD5 = 2FB3EB25944C27267626836435DE7369) "BASIC Interpreter - Version 1.12.03 - Copyright (C) 1984 Corona Data Systems, Inc" from MS-DOS 1.25.
3) Floppy disk images of Compaq MS-DOS 1.10, 1.11, 1.12, 3.00, 3.31.
4) Floppy disk images of MS-DOS 1.25, 2.11, 3.10, 3.30.

Invece la cifratura avviene con un sistema simile dove i registri in SUB e ADD sono invertiti [Kit18]:
1) SUB AL,CH
2) XOR AL,Key1(pos. CH)
3) XOR AL,Key2(pos. CL)
4) ADD AL,CL

Conclusioni

Nel Listato 3 si trova la mia versione in GW-BASIC dell'algoritmo di decifratura (con alcuni aggiustamenti dovuti alla simulazione del registro AL con il GW-BASIC).
Il file sprotetto è ancora "tokenized" ma si può caricare e poi salvare con l'opzione A di conversione in modalità testo.
Lenclud non lo dice ma la decifratura comincia dal secondo byte in poi (il primo è riservato al flag di codifica).

```
1000 REM UNPROT2.BAS
1010 REM UNPROTECTING GW-BASIC PROGRAMS
1020 DIM K1(13):DIM K2(11)
1030 FOR J=1 TO 13:READ K1(J):NEXT J
1040 FOR J=1 TO 11:READ K2(J):NEXT J
1050 REM NAME OF PROTECTED PROGRAM
1060 OPEN "test1.BAS" FOR INPUT AS #1
1070 REM NAME OF UNPROTECTED PROGRAM
1080 OPEN "test1u.BAS" FOR OUTPUT AS #2
1090 CH=13:CL=11
1100 AL=ASC(INPUT$(1,#1))          :REM SKIP FIRST BYTE
1110 PRINT #2,USING"!";CHR$(255); :REM WRITE 0FFhex
1120 IF EOF(1) THEN 1280
1130 AL=ASC(INPUT$(1,#1))
1140 AL=AL-CL                      :REM SUB AL,CL
1150 IF AL < 0 THEN AL=AL+256:REM IT'S BASIC NOT ASSEMBLY
1160 U=AL:V=K1(CH):GOSUB 1310:REM XOR AL,K1(CH)
1170 AL = X                        :
1180 U=AL:V=K2(CL):GOSUB 1310:REM XOR AL,K2(CL)
1190 AL = X                        :
1200 AL = AL + CH                  :REM ADD AL,CH
1210 AL = AL MOD 256               :REM IT'S BASIC NOT ASSEMBLY
1220 PRINT #2,USING"!";CHR$(AL);
1230 CL=CL-1
1240 IF CL = 0 THEN CL=11
1250 CH=CH-1
1260 IF CH = 0 THEN CH=13
1270 GOTO 1120
1280 CLOSE #1
1290 CLOSE #2
1300 END
1310 REM X = U XOR V
1320 X=0
1330 FOR J=0 TO 7
1340 BITU = U MOD 2:BITV = V MOD 2
1350 X = X + (BITU-BITV)*(BITU-BITV)*(2^J)
1360 U=INT(U/2):V=INT(V/2)
1370 NEXT J
1380 RETURN
1390 DATA 154,247,25,131,36,99,67,131,117,205,141,132,169
1400 DATA 124,136,89,116,224,151,38,119,196,29,30
```

Listato 3

Bibliografia

[Bas85] AA.VV. "GW-BASIC User's Manual", 1985.
[Kit18] S.Kitt, "How were Microsoft GW-BASIC “protected” files encoded?", https://retrocomputing.stackexchange.com/questions/7104/how-were-microsoft-gw-basic-protected-files-encoded consultato il 2020_05_23.
[Len18] C.Lenclud, "Deciphering GW-BASIC / BASICA protected programs", https://slions.net/threads/deciphering-gw-basic-basica-protected-programs.50/ consultato il 2020_05_23.

RetroMath: Come ti trasformo un'immagine...

di Giuseppe Fedele

In computer graphics molte applicazioni richiedono di modificare un'immagine cambiando, ad esempio, dimensioni, posizione e orientamento. Queste modifiche, per lo più, possono essere fatte applicando una trasformazione geometrica alle coordinate dei punti dell'immagine. In questo articolo vogliamo analizzare alcune trasformazioni geometriche di base nello spazio 2D.

Tipi di trasformazione

Traslazione

Una delle trasformazioni più semplici è quella di muovere l'immagine in una nuova posizione. In Fig. 1, i punti dell'immagine originale sono disegnati in nero

$$P_1 = \begin{bmatrix} 2 \\ -5 \end{bmatrix} \quad P_2 = \begin{bmatrix} 3 \\ 6 \end{bmatrix} \quad P_3 = \begin{bmatrix} 7 \\ 10 \end{bmatrix}$$

mentre i corrispondenti punti traslati sono disegnati in rosso.

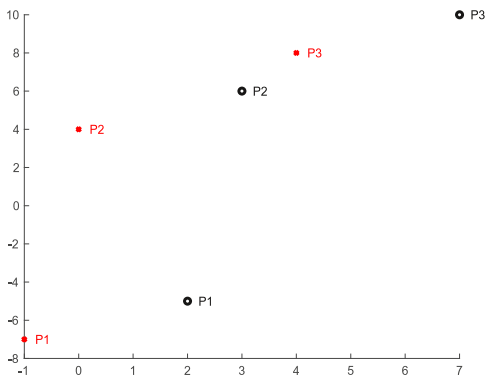


Figura 1. Traslazione.

Per effettuare la traslazione di un punto è necessario aggiungere alle coordinate *x* ed *y* del punto un valore costante, come mostrato in Fig. 2. Le nuove coordinate del punto saranno dunque:

$$\begin{cases} x' = x + \mathrm{t_x} \\ y' = y + \mathrm{t_y} \end{cases}$$

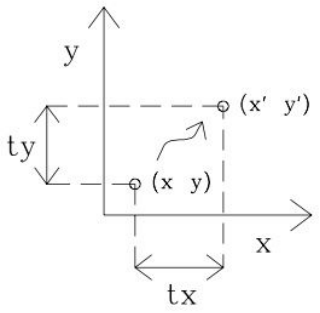


Figura 2. Traslazione di un punto.

Scaling (Scalatura)

Lo scaling è usato invece per cambiare le dimensioni di un oggetto. Lo scaling di un vettore riferito all'origine degli assi del piano *xy* si ottiene moltiplicando ciascuna componente del vettore per un fattore di scala:

$$\begin{cases} x' = x \cdot s_x \\ y' = y \cdot s_y \end{cases}$$

Se $|s_x|$ e $|s_y|$, ovvero i moduli dei fattori di scala, sono entrambi maggiori di 1, allora si ha un incremento delle dimensioni dell'oggetto, viceversa una diminuzione. In Fig. 3, ad esempio, il vettore che unisce l'origine del piano con il punto P_1 è scalato utilizzando i fattori $s_x = \frac{1}{4}, s_y = \frac{1}{2}$ (riduzione delle dimensioni) mentre il vettore relativo al punto P_2 è scalato con $s_x = \frac{3}{2}, s_y = \frac{6}{5}$ (aumento delle dimesioni).

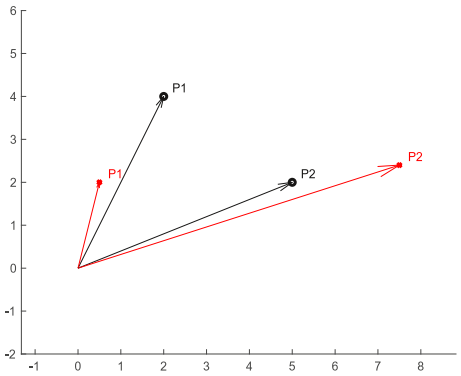


Figura 3. Scaling.

Se i fattori di scala sono gli stessi $s_x = s_y$, allora si ha uno scaling simmetrico, cioè l'immagine è ridotta o espansa della stessa quantità in ciascuna direzione (Fig. 4).

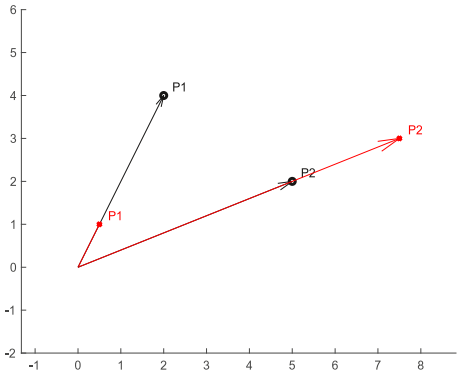


Figura 4. Scaling simmetrico.

Se il fattore di scala su x è negativo ($s_x < 0$), allora l'oggetto viene riflesso rispetto all'asse y, viceversa se $s_y < 0$, allora l'oggetto viene riflesso rispetto all'asse x. In Fig. 5, il vettore relativo a P_1 ha fattori di scala $s_x = -\frac{1}{2}, s_y = 1$ mentre il vettore relativo a P_2 ha fattori $s_x = 1, s_y = -\frac{1}{2}$.

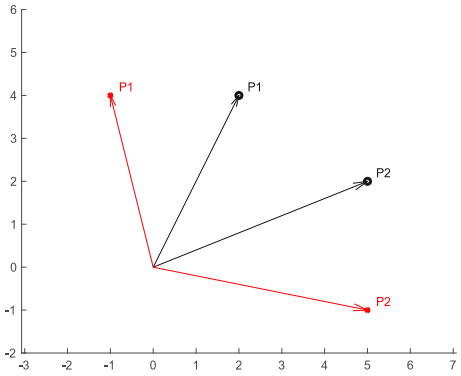


Figura 5. Scaling con fattori di scala negativi.

Rotazione

Un altro tipo comune di trasformazione è la rotazione che viene spesso utilizzata per orientare gli oggetti. La rotazione di un punto dell'oggetto è mostrata in Fig. 6.

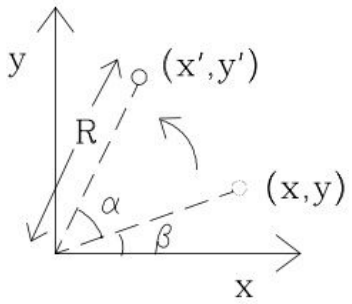


Figura 6. Rotazione di un punto.

Il segmento di lunghezza R che unisce il punto di coordinate (x, y) con l'origine, forma un angolo β con l'asse x, quindi:

$$\begin{cases} x = R\cos(\beta) \\ y = R\sin(\beta) \end{cases}$$

Dopo una rotazione di un angolo α, il nuovo punto avrà coordinate

$$\begin{cases} x' = R\cos(\alpha+\beta) \\ y' = R\sin(\alpha+\beta) \end{cases}$$

Espandendo i termini trigonometrici e sostituendo i valori di x e y, si ha:

$$\begin{cases} x' = R\cos(\alpha)\cos(\beta) - R\sin(\alpha)\sin(\beta) = x\cos(\alpha) - y\sin(\alpha) \\ y' = R\sin(\alpha)\cos(\beta) + R\cos(\alpha)\sin(\beta) = x\sin(\alpha) + y\cos(\alpha) \end{cases}$$

Shearing

Questa trasformazione ha l'effetto di distorcere la forma di un oggetto. Le nuove coordinate di un punto soggetto a shearing sono date da:

$$\begin{cases} x' = x + a\,y \\ y' = y + b\,x \end{cases}$$

Se $b = 0$, l'effetto è quello di distorcere l'immagine traslando la coordinata x in funzione dell'altezza dell'oggetto (con $a > 0$, man mano che l'ordinata del punto aumenta, l'oggetto subisce una maggiore distorsione sull'asse x). In Fig. 7 è riportata una distorsione lungo l'asse x con un fattore di shearing $a = \frac{3}{2}$.

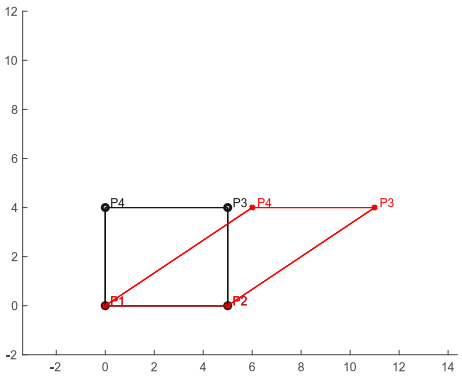


Figura 7. Shearing lungo l'asse x.

Analogamente, una distorsione lungo solo l'asse y si ottiene scegliendo $a = 0$ ed un valore di $b > 0$ (Fig. 8).

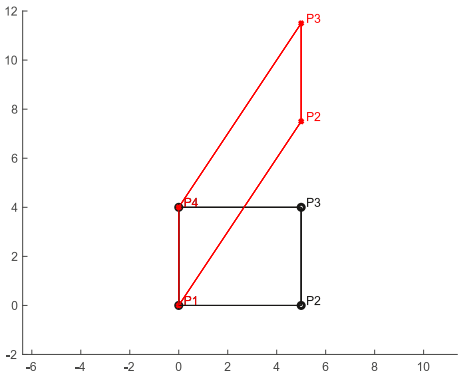


Figura 8. Shearing lungo l'asse y.

La Fig. 9 mostra invece una distorsione su entrambi gli assi con $a = \frac{3}{2}, b = \frac{4}{3}$.

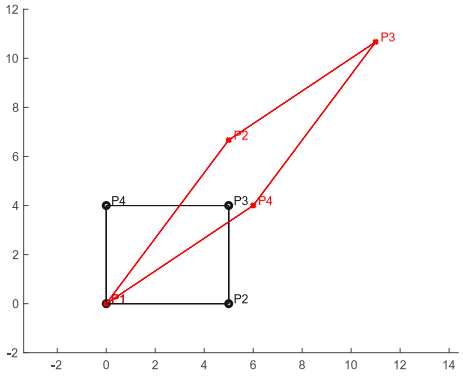


Figura 9. Shearing sugli assi x ed y.

Rappresentazione matriciale

Una formula generale che comprende tutte le trasformazioni viste in precedenza è

$$\begin{cases} x' = a\,x + b\,y + c \\ y' = d\,x + e\,y + d \end{cases}$$

dove a, b, c, d, e, f sono costanti. Con l'artificio di inserire una equazione fittizia, la formula precedente può essere riscritta, in forma matriciale, come:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Le quattro trasformazioni possono essere riscritte con questa notazione come:

- Traslazione

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Scaling

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Rotazione

$$\begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Shear

$$\begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ovviamente la matrice che lascia invariato l'oggetto è la matrice identità

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Poiché il prodotto matriciale non è commutativo, cioè $M_1\ M_2 \neq M_2\ M_1$, allora è importante l'ordine con cui si eseguono le trasformazioni. In Fig. 10 viene mostrata, a sinistra, l'applicazione di una traslazione e poi di una rotazione con $t_x = 2, t_y = 4, \alpha = \frac{\pi}{4}$, mentre, a destra, l'effetto ottenuto con l'ordine invertito delle trasformazioni.

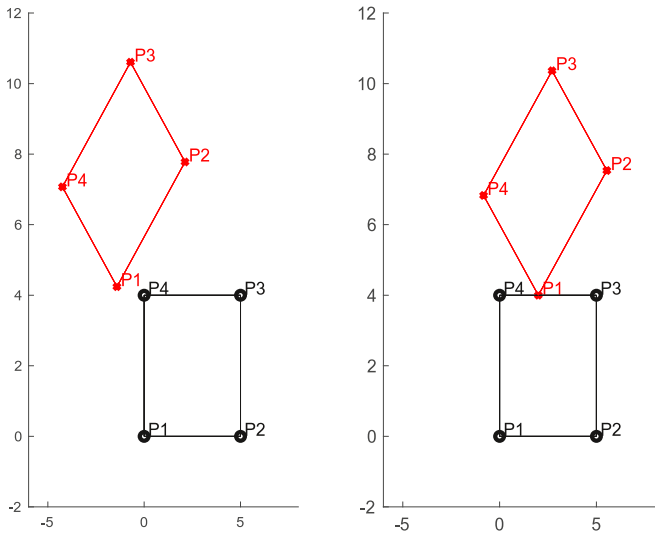


Figura 10. Non commutatività delle trasformazioni.

Bibliografia

[1] J. Vince, Mathematics for Computer Graphics, fifth ed., Springer, 2017.

[2] D. Kothari, G. Awari, D. Shrimankar, A. Bhende, Mathematics for Computer Graphics and Game Programming, Mercury Learning and Information, 2019.

FORTH: l'arma segreta!

di Michael Jean (traduzione in italiano a cura di Francesco Fiorentini)

Nel settembre 1982, la societa' britannica Jupiter Cantab, fondata da due ex membri dello staff della Sinclair Company, ha lanciato il computer **Jupiter Ace**, un diretto concorrente dello ZX-81 e dello Spectrum prodotti dalla loro ex societa'.

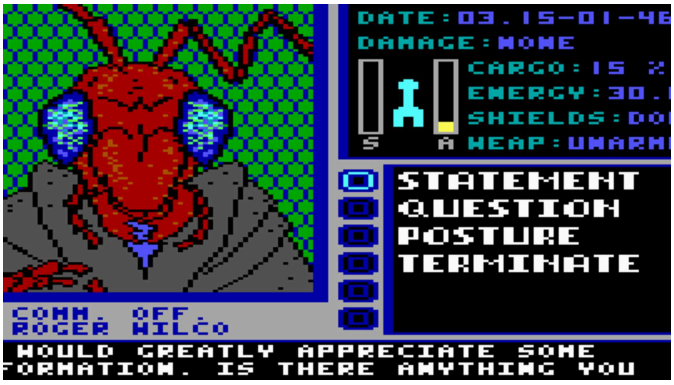
Il Jupiter Ace ha un case simile allo Spectrum e lo stesso microprocessore, ma la piccola azienda ha in mano un jolly: e' programmabile con il **Forth**. Mentre il BASIC e' il linguaggio usato da tutti i concorrenti in questo momento, la societa' britannica ha accettato la scommessa di distribuire la sua macchina equipaggiandola con il linguaggio Forth. Questo linguaggio occupa la meta' dello spazio di memoria e permette un'esecuzione da sei a dieci volte piu' veloce del BASIC, rendendo il Jupiter Ace la macchina piu' efficiente nella sua fascia di prezzo.

All'epoca, il Forth non e' molto usato nei microcomputer, ma non e' nemmeno del tutto sconosciuto.

Gia' nel 1980 la rivista Byte gli dedico' un numero intero. In uno degli articoli, ipotizza addirittura che la societa' Atari stia sviluppando una versione del Forth che gli permettera' di programmare i giochi arcade piu' velocemente: « (...) Atari ha sviluppato una propria versione personalizzata del linguaggio, chiamata game-FORTH, che e' in attesa del suo primo utilizzo per sostituire il codice macchina come linguaggio utilizzato per creare giochi arcade. Un giorno, presto, si potra' giocare a un gioco a gettoni senza sapere che si sta effettivamente eseguendo un programma FORTH.»

Da questa affermazione e' nata la leggenda, ancora oggi in circolazione nella comunita' Forth, che il Forth sarebbe stata l'arma segreta dell'Atari.

Anche se non ci sono prove che le principali case di produzione di videogiochi abbiano fatto un uso esteso e segreto di tale linguaggio, diversi software hanno approfittato della velocita' di sviluppo messe a disposizione dal Forth. Ad esempio, e' documentato che il gioco **Starflight**, pubblicato da Electronic Art nel 1986, sia stato sviluppato in Forth: "Il team ha codificato il gioco principalmente in Forth con alcune routine chiave scritte in x86 assembler. Forth e' stato scelto perche' e' piu' facile da usare rispetto all'assembly e piu' compatto. Questo era importante perche' il gioco doveva entrare in 128K di RAM" .



Ma cos'e' Forth? Forth è un linguaggio sviluppato negli anni sessanta da Charles H. Moore. Incaricato di calcolare le traiettorie satellitari per un osservatorio, Moore cercava di sviluppare un tool che gli permettesse di semplificare il suo lavoro quotidiano. Dopo diversi anni di maturazione, questo tool divenne, nei primi anni Settanta, un linguaggio potente, elegante e coinvolgente.

Il punto di forza di Forth e' l'essere solo un nucleo che puo' essere facilmente modificato e migliorato in base alle proprie esigenze; simile ai blocchi Lego che costruiscono la forma che vogliamo. Il miglior argomento a favore di Forth e' il suo stesso utilizzo pertanto vi presentero' alcuni elementi del linguaggio. L'obiettivo non e' quello di fornire un corso di programmazione, quanto dare una panoramica delle possibilita' offerte dal linguaggio. Per un'iniziazione piu' completa, consiglio l'essenziale: **Starting Forth** di **Leo Brodie** disponibile gratuitamente online.

Ci sono implementazioni Forth per qualsiasi piattaforma. Che si tratti di un Commodore 64, di un Atari 800, di un PDP-11 o del vostro Linux 64 bit. Contrariamente a quanto

```
SCR # 0
  0 ******  fig-FORTH  MODEL  ******
  1
  2       Through the courtesy of
  3
  4
  5        FORTH INTEREST GROUP
  6          P. O. BOX 1105
  7        SAN CARLOS, CA. 94070
  8
  9         Implemented on the
  A           ATARI  800/400
  B                by
  C           Steve Calfee
  D             1/26/81
  E
  F          Copywrite 1981
 10
 11              RELEASE 1
 12        WITH COMPILER SECURITY
 13                AND
 14        VARIABLE LENGTH NAMES
 15
```

molti vorrebbero credere, Forth e' un linguaggio abbastanza semplice, ma per avvicinarsi ad esso dobbiamo lasciare da parte molti riflessi che abbiamo sviluppato con altri linguaggi. Forth e' un linguaggio compilato e interpretato. Quindi, un po' come in BASIC, possiamo testare ogni funzione o procedura (una parola in Forth) direttamente nell'interprete.
Ma e' qui che finisce il confronto, nessun GOTO, nessun numero di riga in Forth.

Forth e' un linguaggio molto diverso da Basic, Fortran, Pascal, C++ o Java. È piu' vicino a Lisp, APL, Prolog o Smalltalk. Anche se sembra un linguaggio funzionale, non possiamo definirlo in questo modo, non tutto e' funzione in Forth. È un linguaggio in cui la programmazione consiste nel costruire uno strumento software a partire da "primitive". Questi strumenti vengono poi utilizzati per crearne di nuovi, e cosi' via fino all'applicazione stessa.

Altre due caratteristiche fondamentali distinguono Forth dai linguaggi tradizionali. In primo luogo, l'uso intensivo dello stack, che evita di creare una moltitudine di variabili e costanti. In secondo luogo, il fatto che funziona in notazione polacca inversa (RPN). Quella notazione che rese famose le calcolatrici HP degli anni '70 e '80. Per esempio, se vogliamo addizionare 12 e 5 digiteremo nell'interprete :

12 5 + .

12 e 5 vengono poi introdotti nello stack, essendo il [+] una funzione (in Forth diremmo una parola) che prende gli ultimi due elementi dello stack per sommarli e restituisce il risultato nello stack.
Il punto [.] visualizza poi l'elemento in cima alla stack.
Questo puo' sembrare complesso, ma, come gli utenti della calcolatrice HP sanno, diventa rapidamente naturale e risparmia la necessita' di parentesi e di mettere in discussione la prioritizzazione delle operazioni. L'espressione 2 * (3 + 6) diventa 2 3 6 + *. L'addizione viene applicata al 3 ed al 6, il 9 viene poi inviato allo stack, la moltiplicazione viene poi applicata al 2 e al 9.
Forth definisce una grande varieta' di operatori speciali per manipolare i dati dello stack, per riorganizzarli (ROT, SWAP), per cancellare elementi (DROP), per generare elementi (DUP, OVER).

Per esempio, la seguente espressione calcola il quadrato di 10 usando la parola DUP che duplica il valore in cima alla stack e invia il risultato allo stack:

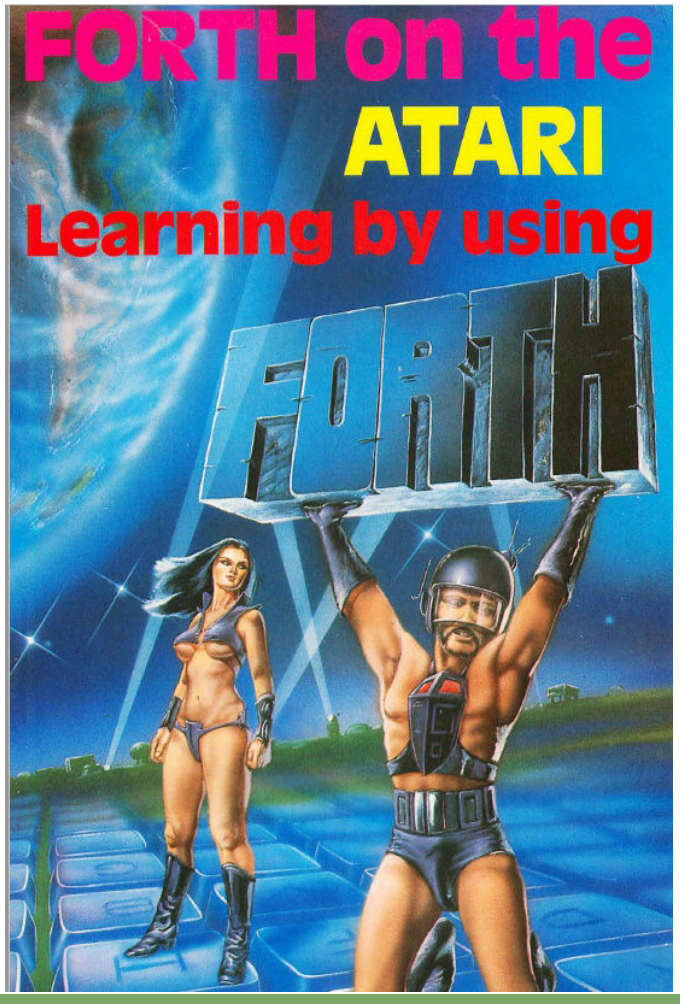
10 DUP *

Potremmo usare questo codice per creare un nuovo operatore chiamato SQUARE. Questo si fa dicendo al compilatore che vogliamo definire una nuova parola usando i due punti [:] seguiti dal nome della nostra parola. Indicheremo la fine della compilazione con il punto e virgola [;] .
Otterremo di conseguenza:

: SQUARE DUP * ;

Abbiamo qui definito una nuova parola che prendera' il primo elemento dello stack, lo duplichera' e lo moltiplichera' per se stesso.

Quindi, **4 SQUARE** inviera' 16 nello stack.
Perche' fermarsi qui?



Da questa nuova parola, posso crearne un'altra: CUBE
: CUBE DUP SQUARE * ;
4 CUBE inviera' 64 nello stack.

```
OK
: SQR DUP * ;

OK
4 SQR .
16
OK
: CUBE DUP SQR * ;

OK
4 CUBE .
64
OK
```

I programmatori Forth preferiscono generalmente definizioni brevi, e una definizione breve puo' essere sorprendentemente efficace. Qui l'algoritmo di Euclide per determinare il Massimo Comun Divisore (GCD in inglese) si inserisce in una singola linea.
: GCD BEGIN 2DUP MOD ROT DROP DUP 0 = UNTIL DROP . ;

```
OK
: GCD BEGIN 2DUP MOD ROT DROP DUP 0 =
  UNTIL DROP . ;

OK
343 280 GCD
7
OK

578 442 GCD
34
OK

4144 7696 GCD
592
OK
```

Naturalmente, potremmo criticare la mancanza di leggibilita' di questa linea di codice, ma nulla ci impedisce di documentare il nostro codice, che di solito e' fatto tra parentesi, principalmente indicando chiaramente l'effetto della parola sullo stack. Infatti, una linea cosi' oscura non dovrebbe mai essere trovata in un buon codice Forth.
Come esercizio, lasciamo al lettore il compito di analizzare questa linea di codice, ma si puo' vedere uno dei tanti modi per introdurre un ciclo condizionale in Forth con BEGIN e UNTIL.

La manipolazione dei dati non avviene solo utilizzando lo stack. Forth permette anche l'uso di variabili e costanti, ma anche qui il linguaggio offre una flessibilita' che si puo' trovare quasi esclusivamente nell'assembler.

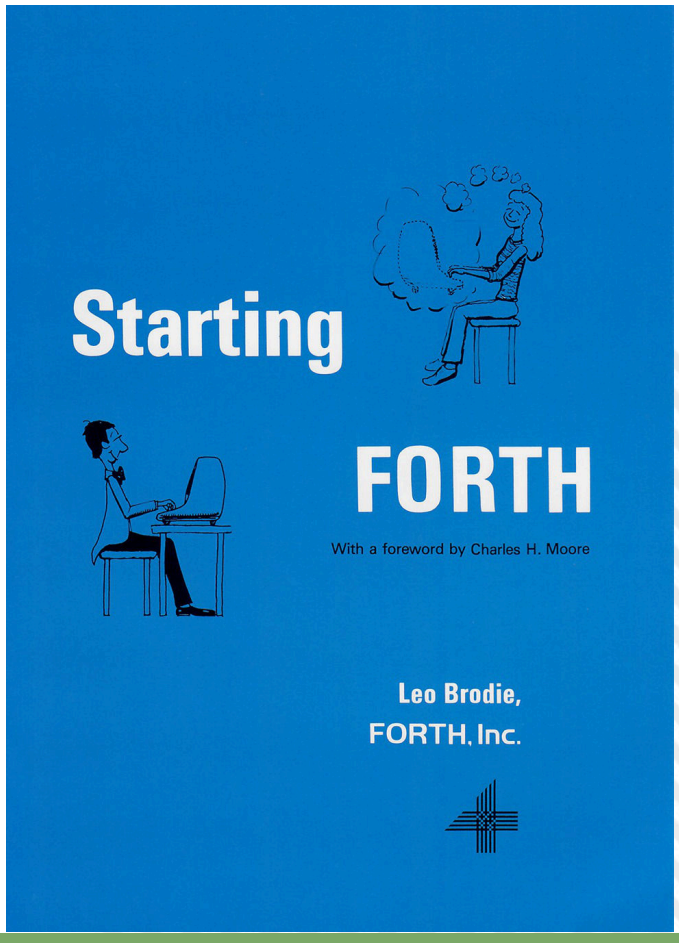
Per esempio dichiariamo una variabile (con la parola Forth VARIABLE)
VARIABILE myvariable
la parola [!] associa un valore a questa variabile.
36 myvariable !

È importante realizzare la distinzione tra l'indirizzo della variabile e il contenuto dell'indirizzo. Se digito il nome della mia variabile nell'interprete, quello che verra' inviato allo stack e' l'indirizzo della variabile.
Per ottenere il contenuto della variabile uso **@** (chiamato fetch in Forth). Cosi' [**myvariable @ .**] ritorna il contenuto che si trova all'indirizzo di memoria myvariable; in questo esempio 36.

Probabilmente ora avete un'idea di cosa intendevamo quando parlavamo di un toolbox. Sicuramente avrete anche un'idea del continente che la programmazione di Forth ci apre.

Forth non ha mai avuto la notorieta' di C, Pascal o BASIC. Il Jupiter Ace non ha avuto il successo commerciale che i suoi creatori avrebbero sperato, ma molti di coloro che hanno sperimentato Forth hanno sviluppato una dipendenza per questo linguaggio e la comunita' Forth e' ancora molto attiva.

Forth non sara' stata l'arma segreta dell'Atari, ma potrebbe diventare la vostra!



L'estensione MiniGrafik per il BASIC del Vic-20

di David La Monaca

Uno dei divertimenti più raffinati di noi geek e retrocomputeristi è quello di creare o ricercare e provare i cosiddetti "computer hack", particolari soluzioni software o pezzi di codice che, sfruttando alcune caratteristiche hardware, riescono a risolvere, tramite trucchi e stratagemmi, un determinato problema per velocizzare o semplificare un processo o per giungere ad implementare funzioni e applicazioni più o meno appariscenti, di solito qualcosa di inaspettato per una determinata macchina, date le sue limitazioni. Questa è la bellezza del software, la "magìa" che a volte riesce a superare i limiti intrinseci di un'architettura hardware. YouTube e alcuni siti specializzati nel raccogliere demo e intro sono pieni di questi hack. Un video a colori in tempo reale su Atari 800XL? Fatto.[1] Un programma di scacchi su ZX81 con 1K di RAM? Visto![2] Più di 16 colori in contemporanea su un immagine bitmap per C64? Visto anche questo![3] E Doom che gira su un Vic-20 inespanso? Bellissimo![4] Un brano .MOD che suona su Impulse Tracker e contemporaneamente esegue l'animazione Bad Apple sulle colonne delle voci audio? Che idea pazzesca![5]
Potrei continuare a lungo. L'elenco è molto fitto e potrebbe tranquillamente essere oggetto di un prossimo, sfizioso articolo per RMW. E talvolta, nella categoria degli hack finiscono singoli pezzi di codice scritti in assembly o altri linguaggi (persino in BASIC) che risolvono problemi specifici o costituiscono soluzioni utili ed efficaci. La creatività nello scrivere software, se scaturita da giuste e sapienti mani, può sfociare in piccoli o grandi capolavori. Spesso però soltanto gli esperti e i cultori dell'arte del coding possono apprezzare e riconoscere questi veri e propri colpi di genio, perché sono i frutti di una profonda conoscenza dell'hardware di un computer, di un singolo processore o di un chip audio o video, di un linguaggio o di un intero sistema.

MINIGRAFIK per Vic-20

Nella categoria degli hack allo stesso tempo sorprendenti e utili ricade un progetto di Michael Kircher, un ingegnere tedesco, universalmente riconosciuto come una sorta di guru del Commodore Vic-20, autore/coder di molti programmi e giochi e fra gli utenti più attivi di Denial, il forum più famoso tutto dedicato a questa macchina. Michael è anche l'autore e manutentore di una libreria grafica (e del relativo pixel editor ad alta risoluzione) che può tranquillamente essere definita il "top" per il piccolo Vic. Anzi, molto di più: Minigrafik, questo il nome dato alla libreria che estende il famigerato BASIC V2 di casa Commodore con alcuni comandi grafici, è un vero e proprio "software hack", che riesce a dotare il Vic-20 di primitive grafiche e comode istruzioni per riempire lo schermo 160x192 di tanti bei pixel colorati per disegnare immagini bitmap, tracciare grafici e funzioni 2D/3D, progettare giochi e molto altro ancora. Un editor a tutto schermo chiamato MiniPaint è il complemento pratico della libreria MiniGrafik. Esso permette di disegnare "a mano libera", selezionare colori e forme per realizzare sfondi e immagini bitmap da usare in giochi e applicazioni o da esportare in un formato riutilizzabile. Lo stesso MiniPaint è in realtà un'applicazione scritta in parte in BASIC sulla base di MiniGrafik, che ormai è diventata uno standard per la creazione di giochi e programmi BASIC o Assembly che fanno uso della modalità grafica ad alta o media risoluzione. Per estendere il BASIC del Vic-20 con Minigrafik bastano 8K di espansione RAM ed un dischetto o immagine D64 [MG]. Poi basta caricare la libreria e mandarla in esecuzione. Una volta lanciata, l'estensione si caricherà automaticamente in RAM, allocherà la memoria grafica e restituirà il controllo al BASIC cambiando il messaggio di apertura e aggiornando il numero di byte liberi (qualcuno in meno, ovviamente). In alternativa è possibile utilizzare un boot loader, che in sequenza carica Minigrafik, inizializza l'estensione senza tornare alla schermata di avvio e poi procede al caricamento di un programma client (applicazione o gioco) che necessita della libreria grafica. In questo modo, il metodo d'implementazione di Minigrafik risulta molto semplice: basta che l'immagine disco D64 contenga una copia della

Figura 1. Il logo Minigrafik generato con Minigrafik

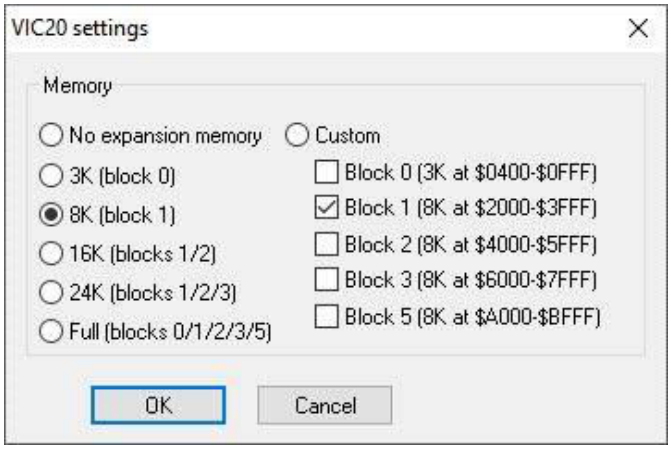


Figura 2. Espansione RAM necessaria per Minigrafik

Figura 3. La tabella colori del Vic-20 0-15

libreria e tutti i programmi e giochi che ne fanno uso potranno facilmente essere lanciati dopo il caricamento dell'estensione. Questo sistema modulare è molto più comodo rispetto al metodo di incorporare la libreria in ogni programma. Anche eventuali aggiornamenti dell'estensione si possono effettuare più rapidamente e senza sforzo.

L'estensione Minigrafik consiste di un singolo file .PRG che può essere caricato da disco con un semplice comando: LOAD"MINIGRAFIK",8. Dopo il RUN apparirà di nuovo il messaggio d'avvio del CBM BASIC. Come detto, il numero di byte disponibili per i vostri programmi BASIC sarà diminuito della quantità allocata per lo schermo bitmap e per l'estensione stessa, che in pratica aggiunge all'interprete BASIC ben 6 nuovi comandi custom ed una funzione. Tutte le nuove istruzioni sono precedute dal carattere '@' e devono essere utilizzate soltanto in modalità programma e non come comandi diretti dal prompt del BASIC. Questo perché l'output di testo normale può interferire con lo schermo ad alta risoluzione. Inoltre, quando si scrive codice, dopo un THEN di un costrutto IF, bisogna avere l'accortezza di utilizzare sempre il carattere due punti ':' prima di un comando Minigrafik, altrimenti il programma si fermerà con un fastidioso '? SYNTAX ERROR'.

Figura 4. Immagine generata con MiniPaint

I comandi aggiuntivi

I nuovi comandi che MG aggiunge al BASIC sono: @ON, @CLR, @RETURN, @SAVE e @LOAD. Completa la dotazione della libreria la funzione @() che restituisce lo stato ed il colore di un singolo pixel. Vediamone velocemente le caratteristiche e l'uso:

@ON inizializza il modo bitmap con risoluzione 160x192 pixel, ricentrando correttamente lo schermo sia per i VIC NTSC sia PAL.

@CLR pulisce lo schermo ad alta risoluzione. La RAM colore viene inizializzata al colore di primo piano.

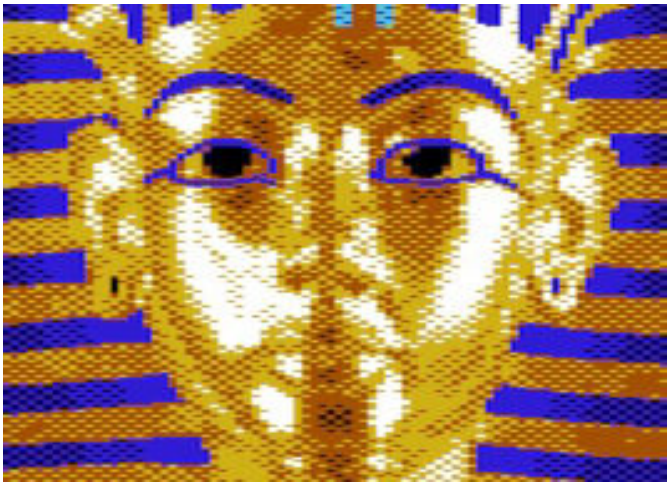


Figura 5. Immagine generata con MiniPaint

@RETURN torna al modo testo. Se si verifica un errore durante l'esecuzione di un programma, questo comando viene automaticamente eseguito prima di stampare a video il messaggio d'errore.

@<colore>,<x1>,<y1> [TO <x2>,<y2>] – disegna una linea con il colore indicato dalle coordinate di schermo x1,y1 alle coordinate x2,y2. Se la parte TO viene omessa, un singolo pixel alle coordinate x1,y1 viene plottato. I valori di x1 e x2 vanno da 0 a 159, mentre y1 e y2 da 0 a 191. L'origine dello schermo è fissata nell'angolo in alto a sinistra. Tutti gli argomenti possono essere inseriti come numeri, variabili o espressioni numeriche.

In media risoluzione (o modo multi-colore) la risoluzione viene dimezzata e quindi per tutte le coordinate x pari, x ed x+1 indicano lo stesso pixel. Il colore di primo piano (da 0 a 7) sono settati individualmente per ogni cella di 8x16 pixel. In questa modalità grafica due colori extra sono disponibili oltre a quello di sfondo (G) e di primo piano (F): il colore di bordo (B) e quello ausiliario (A). Il colore di sfondo (da 0 a 15), quello di bordo (da 0 a 7) e quello ausiliario (da 0 a 15) si applicano a tutto lo schermo. Per assegnare i colori sullo schermo nelle due modalità, occorre usare qualche comando POKE ben diretto:

POKE 36879,16*G+8+B (per impostare i colori di sfondo G e di bordo B)

POKE 36878,16*A (per impostare il colore ausiliario A)

Figura 6. Una schermata di Minipaint

POKE 646,8*M+F (per impostare il colore di primo piano [F] ed abilitare [M=1] o disabilitare [M=0] il modo multi-colore)

Ecco la tabella dei colori del VIC:

```
0 Black (Nero)          8 Arancione
1 White (Bianco)        9 Light Orange (Arancione ch.)
2 Red (Rosso)          10 Light Red (Rosa)
3 Cyan (Celeste)       11 Light Cyan (Celeste chiaro)
4 Purple (Viola)       12 Light Purple (Viola chiaro)
5 Green (Verde)        13 Light Green (Verde chiaro)
6 Blue (Blu)           14 Light Blue (Azzurro)
7 Yellow (Giallo)      15 Light Yello (Giallo chiaro)
```

@SAVE [<filename> [,<device>]] – salva lo schermo bitmap su un dispositivo (ad esempio su disco, 8). Il nome del file è opzionale nel caso di salvataggio su nastro. Il file viene salvato con un comando SYS iniziale che richiama una routine di visualizzazione della libreria grafica. La routine attende poi la pressione di un tasto e infine fa ripartire il VIC.

@LOAD [<filename> [,<device>]] – carica una schermata bitmap da un dispositivo (disco o nastro). Il comando senza argomenti carica il primo schermo bitmap dal nastro. Quando il caricamento è terminato, l'immagine viene automaticamente mostrata sullo schermo ed il programma in esecuzione continua con l'istruzione seguente. Non c'è attesa per la pressione di un tasto.

@(<x>,<y>) – L'unica funzione in dotazione restituisce il colore del pixel che si trova alle coordinate (x,y). L'argomento x varia da 0 a 159 mentre y va da 0 a 191 ed entrambi possono essere inseriti come numeri, variabili o espressioni numeriche. A seconda del modo grafico in uso (M=0 oppure M=1) la funzione restituisce valori da 0 fino a 3 (0 per pixel di sfondo, 1 per pixel in primo piano, 2 e 3 per un pixel di colore ausiliario).

Figura 7. Immagine bitmap generata con MiniPaint

Programmi d'esempio

Invitandovi a scaricare e a provare il software MiniGrafik ed i suoi numerosi esempi inclusi nel dischetto di distribuzione, aggiungiamo, a corredo di questo piccolo articolo di presentazione, un paio di esempi d'utilizzo della libreria. Nel primo listato facciamo riferimento agli articoli già apparsi su RMW riguardanti il tracciamento di funzioni 2D mentre nel secondo presentiamo un gioco completo, un clone di Snake, scritto dallo stesso Michael Kircher.

Figura 8. Il semplice gioco clone di Snake

Entrambi i listati sono auto-esplicativi e facili da comprendere per chi mastica almeno un po' di BASIC. Esaminandoli potrete apprezzare come i pochi comandi della libreria MiniGrafik siano davvero ben congegnati e si integrino agilmente con il resto delle istruzioni BASIC V2, fornendo così uno strumento conciso ed efficace per scrivere programmi che facciano uso della modalità grafica. I risultati in termini di velocità di tracciamento di linee rette e curve, caricamento e salvataggio di schermate, controllo e flessibilità nella gestione dello schermo bitmap, sono perfettamente all'altezza delle aspettative e costituiscono ormai uno standard de facto, un punto di riferimento stabile per tutti i programmatori dell'amatissimo e indimenticato Vic-20.

-- Listato: mg-2d functions

```
10 rem vic-20 + minigrafik
20 rem 2d functions plot
30 rem ******************
120 sx=160
130 sy=191
140 hy=sy/2
150 printchr$(147)
160 print"    2d functions plot"
170 print
180 print"1.y=x*x*sin(1/x)"
```

```
190 print
200 print"2.y=x*sin(1/x)"
210 print
220 print"3.y=sqr(x*x+2)"
230 print
240 print"4.y=cos(x*exp(-x/5))"
250 print
260 print"5.y=6+2*x*x-x*x*x*x"
270 print
280 print"type in the number of"
290 print"the equation";
300 input n
310 ifn=1thendeffna(x)=x*x*sin(1/x)
320 ifn=2thendeffna(x)=x*sin(1/x)
330 ifn=3thendeffna(x)=sqr(x*x+2)
340 ifn=4thendeffna(x)=cos(x*exp(-x/5))
350 ifn=5thendeffna(x)=6+2*x*x-x*x*x*x
360 print
370 print"values of x range"
380 print
390 print"lowest value";
400 inputa
410 print
420 print"highest value";
430 inputb
440 print
450 ifa>=bthenprint"error-try again"
460 ifa>=bthengoto360
500 rem ***calculating range of y ***
510 print"calculating range of y"
520 c=(b-a)/100
530 m=1.0e-30
540 forx=atobstepc
550 ifx=0thengoto580
560 y=abs(fna(x))
570 ifm<ythenm=y
580 next x
590 print"ready to plot"
600 fori=1to1000
610 next i
620 printchr$(147)
630 gosub1010:rem prep screen
700 rem *** plotting
710 c=c/10:rem try c/5 or c/20
720 forx=atobstepc
730 ifx=0thengoto790
740 y=fna(x)
750 u=sx*(x-a)/(b-a)
760 v=hy+hy*y/m
770 ifv<0orv>sythengoto790
780 gosub1110:rem plot u,v
790 next x
800 rem *** ending
810 getg$:rem g$=inkey$
820 ifg$=""thengoto810
830 gosub1210:rem restore screen
840 printchr$(147)
850 print" another go? y or n"
860 getg$
870 ifg$<>"y"andg$<>"n"thengoto860
880 ifg$="y"then goto150
890 end:rem stop
1000 rem *** prepare hi-screen
1010 @on
1020 @clr
1030 :
1040 return
1100 rem *** plot function's dots
1110 @1,u,sy-v
1180 return
1200 rem *** restore screen
1210 @return:poke198,0
1220 return
```

-- Listato: mg-snake

```
1 REM *****
2 REM SNAKE
3 REM *****
10 DIMDX(3),DY(3):DX(0)=2:DY(1)=-
3:DX(2)=-
2:DY(3)=3:HX=80:HY=97:TX=HX:TY=HY:I=1
11
POKE36878,160:POKE36879,15:POKE646,13:@
ON:@CLR:@1,0,0TO0,191:@1,158,0TO158,191
12 FORY=0TO2:@1,0,YTO158,Y:@1,0,191-
YTO158,191-Y:NEXT:@2,HX,HY+1TOHX,HY-
1:GOSUB23:SC=0
13 GETA$
14 IFA$="X"THENI=0
15 IFA$=";"THENI=1
16 IFA$="Z"THENI=2
17 IFA$="/"THENI=3
18 @1,HX,HY+1:@I,HX,HY:@1,HX,HY-
1:HX=HX+DX(I):HY=HY+DY(I)
19 J=@(HX,HY-1):@2,HX,HY+1TOHX,HY-
1:IFJ=3THENJ=0:GOSUB23:GOTO21
20 K=@(TX,TY):@0,TX,TY+1TOTX,TY-
1:TX=TX+DX(K):TY=TY+DY(K)
21 IFJ=0THEN13
22 @RETURN:PRINT"SCORE:"SC:END
23
X=2*INT(RND(1)*80):Y=3*INT(RND(1)*64):I
F@(X,Y)<>0THEN23
24 @3,X,YTOX,Y+2:SC=SC+1:RETURN
```

Riferimenti

[1] Real time video su Atari 8-bit - https://www.youtube.com/watch?v=PAeYZWz15Ns
[2] ZX-81 1K Chess - https://www.youtube.com/watch?v=m0VAwqg9N0k
[3] Immagine HFLI su C64 - https://www.youtube.com/watch?v=SgM6KVdae3Y
[4] Doom su Vic-20 - https://github.com/Kweepa/vicdoom
[5] Bad Tracker - https://www.youtube.com/watch?v=SDvk3aL78fI
[MG] MiniGrafik Batch Suite - http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?t=5179
MiniGrafik download - https://dateipfa.de/.Public/denial/minigrafik/minigrafik.zip
MiniPaint - http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?t=5627
MiniPaint manual - https://dateipfa.de/.Public/denial/minigrafik/manual.zip

Alien Attack!

un nuovo gioco in Locomotive Basic

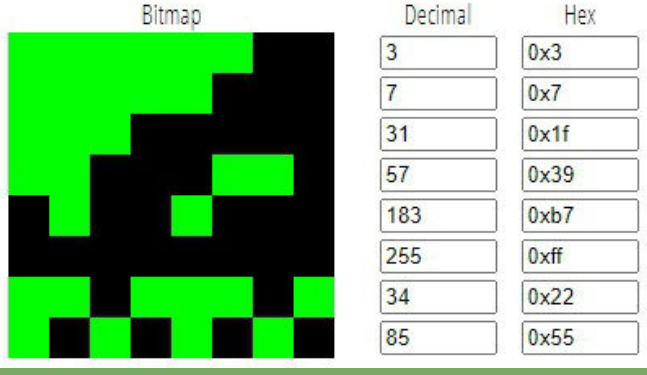
di Francesco Fiorentini

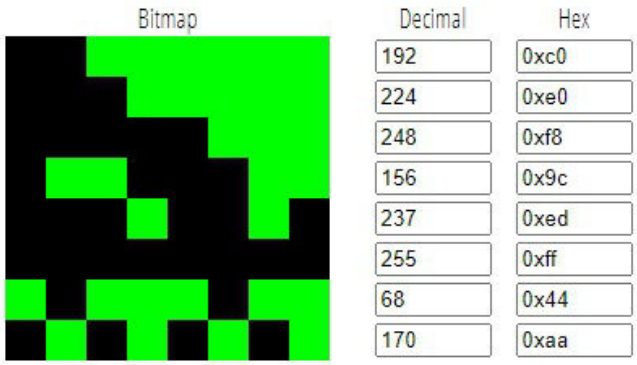
All'inizio del mese di agosto, sul gruppo **Retroprogramming Italia - RP Italia**, e' stato indetta una challenge con lo scopo di invitare i frequentatori a riprodurre un clone del gioco Air Attack. La sfida proposta dal gruppo, con l'intento di avvicinare sempre piu' persone alla retro-programmazione, era la seguente:

A) La challenge consiste nel riprodurre uno o anche più cloni del gioco "Air Attack", usando i seguenti linguaggi: il Basic, o il C, o l'ASM o anche una combinazione dei precedenti, come ad esempio Basic+ASM o C+ASM.
B) Tali giochi potranno essere programmati per qualsiasi computer a 8/16 bit (comprese le piattaforme 8088/8086).
C) I vostri lavori dovranno rispettare comunque, le seguenti categorie:
1) Categoria "Full BASIC"
2) Categoria "Full C"
3) Categoria "Full ASM"
4) Categoria "Mixed C" (C + ASM)
5) Categoria "Mixed BASIC" (Basic + ASM)

Inizialmente il termine di consegna dei lavori era stato fissato per la fine di agosto e, oberato dal lavoro e dagli impegni familiari, avevo inizialmente soprasseduto.
Verso la fine del mese di agosto noto pero' che la deadline e' stata posticipata al 24 di settembre e, spinto dalla curiosita', decido di scrivere due righe per animare un aeroplanino...

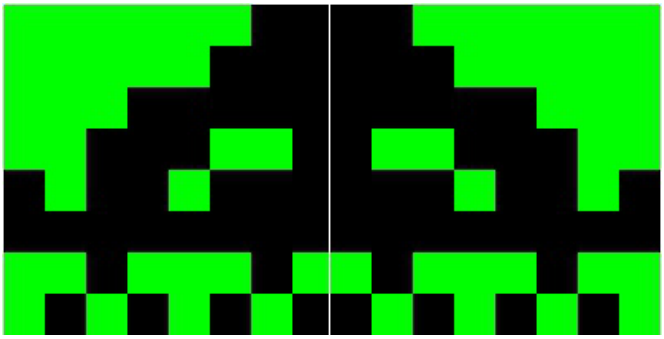
Ovviamente la mia scarsa propensione alla grafica non e' affatto d'aiuto per disegnare un aeroplano, decido quindi di puntare su qualcosa di piu' lineare. Idea! Posso sostituire l'aeroplano con un'astronave aliena; molto piu' semplice da realizzare.
Rimane comunque il problema che per realizzare un'astronave che sia minimamente credibile, un singolo carattere non e' sufficiente, ne servono almeno due.





Armato quindi di tutto il mio estro artistico, realizzo le due opere di cui sopra.

Vederle singolarmente, ingrandite nell'editor 8x8, non fanno una grossa figura, ma una volta affiancate devo ammettere che l'effetto e' credibile.



Avevo la mia astronave.
Adesso non restava altro da fare che animarla e vedere se il Locomotive Basic fosse in grado di gestire il movimento di due caratteri affiancati abbastanza velocemente (...spoiler: lo e' decisamente!).

Per fortuna l'animazione del protagonista del gioco Air Attack e' piuttosto semplice. Si tratta di muovere l'oggetto orizzontalmente, in una sola direzione e, una volta giunti al limite estremo dello schermo, ripartire dall'estremita' opposta soltanto un po' piu' in basso.

Si puo' quindi disegnare i due caratteri affiancati e spostarli uno alla volta di una posizione (verso destra o verso sinistra), avendo cura di ripulire la prima casella opposta al senso di marcia (piu' facile a farsi che a dirsi...).
Ripetendo questo processo per tutta la lunghezza dello schermo si avra' l'effetto di un oggetto unico che si muove da una parte all'altra del video.
Suggerisco caldamente a chi non ha mai tentato di realizzare un gioco, a provare ad animare un oggetto nello

schermo per capire quanto effettivamente sia semplice questa tecnica. Una volta fatto questo esperimento, avrete voglia di aggiungere qualche altro oggetto e poi ancora altro ed in men che non si dica avrete realizzato qualcosa che assomiglia molto ad un gioco. Provare per credere!

Bene, adesso che ho realizzato l'animazione del protagonista principale, usando lo stesso principio, ma in verticale, posso realizzare l'animazione della bomba.
Tra l'altro, l'animaziome della bomba si porta naturalmente dietro la distruzione dei palazzi, visto che il disegno della bomba andra' a sovrascrivere di volta in volta un piano del palazzo. Piu' semplice di cosi'!

Il disegno dello sfondo forse e' la cosa che mi ha portato via piu' tempo. Volevo realizzare qualcosa che fosse si' casuale, ma che al tempo stesso avesse una certa logica. Inizalmente avevo quindi puntato a limitare il numero di palazzi e la loro altezza disegnandoli pero' casualmente lungo tutta la lunghezza dello schermo... L'effetto era orribile! Piu' che una citta' sembravano una serie di scale messe a casaccio, senza nessuna logica.

Ci vuole un'idea migliore... Pensa Francesco, pensa!

Idea! E se disegnassi i palazzi partendo dal centro allargandomi verso destra e verso sinistra man mano che ne aggiungo uno?
Posso sempre limitarne il numero e l'altezza ed incrementare queste variabili man mano che si prosegue nel gioco, ma almeno l'impatto visivo dovrebbe essere migliore.
Provo e... EUREKA! Ho trovato la mia routine per il disegno dello sfondo.

L'effetto non e' niente male, soprattutto pensando che il tutto e' generato casualmente e dinamicamente.
Aggiungete pure che i piani dei palazzi e le loro sommita' sono anch'essi generati casualmenete scegliendo rispettivamente da un gruppo di 6 e 4 tile diversi e comprenderete come sia effettivamente difficile giocare una partita che sia uguale alla precedente.

Ho raggiunto il mio scopo, ho creato uno sfondo casuale che pero' ha una parvenza di credibilita' assomigliando ad una citta'.

Cosa rimaneva da fare? Ah si'! Il controllo di collisione dell'astronave con i palazzi ed il passaggio al livello successivo una volta che tutti i palazzi sono stati distrutti. Beh, non ci crederete, ma delle due, la seconda, era la sfida piu' complessa.

Il controllo delle collisioni in Locomotive Basic e' piuttosto banale. Questo linguaggio e' corredato della funzione COPYCHR$() che copia il carattere della posizione corrente in una variabile. E' intuitivo quindi come sia sufficiente posizionarsi con una LOCATE nella posizione che assumera' la nostra astronave e leggerne il contenuto. Se il carattere contenuto e' un piano del grattacielo o una sua sommita', significa che l'astronave e' entrata in collisione con un palazzo!

Per quanto riguarda invece intercettare la distruzione di tutti i palazzi ed il conseguente passaggio al livello successivo ho utilizzato un trucchetto. Avrei potuto memorizzare la posizione di ogni palazzo in un array e ripulirne la posizione ogni volta che la bomba era sulla sua verticale, ma poi avrei dovuto scorrere l'array per capire se tutti i palazzi erano stati distrutti...

Ho quindi optato per creare una stringa di testo contenente 40 volte il carattere 0. Ogni volta che disegno un palazzo inserisco un 1 nella posizione corrispondente, mentre imposto nuovamente il valore a 0 nella posizione in cui viene sganciata la bomba. Ogni volta che l'astronave ha raggiunto l'estremita' dello schermo controllo la variabile stringa e se e' formata da 40 volte 0, vuol dire che il livello e' ripulito e posso passare al successivo.

Adesso l'ossatura del gioco e' praticamente completa e posso cominciare a lavorare al suo arricchimento.

Disegno una prima immagine come introduzione, ma mi accorgo presto che manca di mordente. Il mio gioco si chiamera' **Alien Attack!** e la missione degli alieni e' quella di invadere la Terra... Ho bisogno quindi di qualcosa di piu' forte. L'immagine attuale, che vede un alieno faccia a faccia con un essere umano, e' proprio quella che cercavo.

Nel numero scorso vi ho descritto come importare immagini sull'Amstrad e come caricarle a video con il Basic. Ho usato quella stessa tecnica per visualizzare la schermata introduttiva e l'immagine del Game Over.

Ogni gioco che si rispetti deve invogliare il giocatore premiandolo con un punteggio e con la possibilita' di migliorare sempre di piu' sfidando se stesso.
Aggiungo quindi la gestione dei punti:
- ogni passaggio dell'astronave senza morire viene ricompensato con 50 punti
- completare il livello premia con 500 punti, piu' 100 punti per ogni riga rimanente prima di toccare terra
e la gestione dell'high score.

Per rendete il tutto piu' interessante, e dare la possibilita' di migliorare sempre un po' di piu', si vince una vita extra ogni 15000 punti (e multipli di 15000).

La tecnica di gioco e' ovvia, suggerisco comunque di mirare ai palazzi piu' alti in prima battuta e passare a quelli bassi successivamente...

Ah, dimenticavo...
Ho usato nuovamente il carattere Future Set che vi ho descritto nel numero 24.
Adesso posso dire che il gioco e' completo!

Il codice, sufficientemente commentato, lo trovate qua sotto, mentre l'immagine virtuale del dischetto (e le immagini) da caricare sul vostro emulatore Amstrad CPC preferito (io uso WinAPE), potete scaricarla da qui:
http://www.retromagazine.net/download/AlienAttack.zip

Provatelo e fatemi sapere cosa ne pensate.
Buon divertimento!!!

```
10 REM **********************************
11 REM AlienAttack! by Francesco Fiorentini
12 REM SETTEMBRE 2020
13 REM **********************************
17 HISCORE=5000
18 GOSUB 10000: REM INTRO GRAFICA
19 GOSUB 9020: REM DEFINIZIONE CARATTERI
20 CLS

29 REM DEFINIZIONE CARATTERI
30 GOSUB 6000
39 REM NP=NUMERO PALAZZI INIZIALE - TH=TOP HIGH
40 NP=6:TH=6:LV=1:LF=3:PT=0:EXTRA=15000

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

1000 FOR R = 1 TO 23
1100 FOR I = 1 TO 42
1110 LOCATE I,R:CK$=COPYCHR$(#0):CKV=ASC(CK$)
1120 REM CONTROLLO COLLISIONE
1130 REM - VECCHIO CONTROLLO - IF CK$=PT$ OR CK$=PF$ THEN GOTO 3000
1140 IF CKV<>32 AND CKV<>254 AND CKV<>255 THEN GOTO 3000
1200 PRINT AF$
1220 IF I>1 THEN LOCATE I-1,R: PRINT AB$
1230 IF I>2 THEN LOCATE I-2,R: PRINT " "
1239 REM CONTROLLO DELLA BOMBA
```

```
1240 GOSUB 5000
1280 NEXT I
1290 PT=PT+50: IF PT>=EXTRA THEN GOTO 1291 ELSE GOTO 1298
1291 EXTRA=EXTRA+15000:LF=LF+1
1292 PEN 2:LOCATE 1,24: PRINT"LEVEL:"; LV ;"- LIVES:"; LF ; "- POINTS:"; PT:PEN 1
1293 GOTO 1299
1298 LOCATE 30,24:PEN 2: PRINT PT: PEN 1
1299 IF SF$="00000000000000000000000000000000000000000000" THEN PT=PT+(23-R)*100:GOTO 1310
1300 NEXT R
1301 REM LEVEL COMPLETED!!! MOVE TO NEXT ONE
1310 GOSUB 6500
1320 GOTO 70

3000 REM CRASH
3005 CRA$="***":PCRA$="   ":X=I-2:IF X<=0 THEN X=1:CRA$="**":PCRA$="  "
3010 LOCATE X,R: PRINT CRA$
3020 LF=LF-1
3030 IF LF=0 THEN GOTO 3500
3040 FOR T=1 TO 1000: NEXT T
3050 LOCATE X,R: PRINT PCRA$
3060 PEN 2:LOCATE 1,24: PRINT"LEVEL:"; LV ;"- LIVES:"; LF ; "- POINTS:"; PT
3061 PEN 1: LOCATE 10,25:PRINT "- HI-SCORE:"; HISCORE; "-"
3070 GOTO 99

3499 REM GAME OVER
3500 MODE 1
3501 LOAD "GOVER.SCR",&C000
3510 LOCATE 11,6: PRINT"----  GAME OVER  ----"
3511 LOCATE 15,8: PRINT"SCORE: "; PT
3525 FOR T=1 TO 1000: NEXT T
3530 LOCATE 11,10: PRINT"-- PLAY AGAIN? Y/N --": PEN 1
3531 IF PT>HISCORE THEN HISCORE=PT
3550 RE$ = INKEY$
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)
5010 IF INKEY(47)=0 AND BF=0 THEN GOTO 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 GOTO 5200 ELSE BR = BR+1
5030 IF BR = 24 THEN BR=0: BF=0: LOCATE BL,23: PRINT " ": MID$(SF$,BL,1)="0": GOTO 5200
5040 LOCATE BL,BR-1: PRINT " "
5190 LOCATE BL,BR: PRINT BO$
5200 RETURN

6000 REM DISEGNA ALIENO RIDEFINENDO IL CHAR(254 e 255)
6010 SYMBOL 254, 3, 7, 31, 57, 183, 255, 34, 85
6020 REM DISEGNA ALIENO RIDEFINENDO IL CHAR(255)
6030 SYMBOL 255,192, 224, 248, 156, 237, 255, 68, 170
6040 REM DISEGNA BOMBA RIDEFINENDO IL CHAR(253)
6050 SYMBOL 253,0, 36, 24, 60, 60, 60, 60, 24
6060 REM DISEGNA I PIANI DEL PALAZZO RIDEFINENDO IL CHAR (230,231,232,233,234,235)
6061 SYMBOL 230,255, 219, 255, 219, 255, 219, 255, 219
6062 SYMBOL 231,255, 251, 255, 223, 251, 255, 223, 255
6063 SYMBOL 232,255, 219, 255, 223, 255, 223, 255, 255
6064 SYMBOL 233,255, 255, 219, 255, 219, 255, 255, 255
6065 SYMBOL 234,255, 189, 255, 247, 255, 251, 191, 255
6066 SYMBOL 235,255, 223, 255, 247, 255, 255, 223, 255
6080 REM DISEGNA TOP1 DEL PALAZZO RIDEFINENDO IL CHAR (240) -
6081 SYMBOL 240, 0, 0, 0, 0, 24, 60, 90, 189
6082 REM DISEGNA TOP2 DEL PALAZZO RIDEFINENDO IL CHAR (241)
6083 SYMBOL 241, 129, 129, 129, 129, 195, 231, 255, 255
6084 REM DISEGNA TOP3 DEL PALAZZO RIDEFINENDO IL CHAR (242)
6085 SYMBOL 242, 24, 126, 126, 24, 24, 60, 126, 255
6086 REM DISEGNA TOP4 DEL PALAZZO RIDEFINENDO IL CHAR (243)
6087 SYMBOL 243, 24, 24, 24, 24, 24, 60, 126, 255
6189 REM AF$=ALIENFRONT - AB$=ALIENBACK - BO$=BOMB - PF$=PALACEFLOOR - PT$=PALACETOP
6190 AF$=CHR$(255):AB$=CHR$(254):BO$=CHR$(253):PF$=CHR$(230)
6200 RETURN
```

```
6499 REM GESTIONE DINAMICA DEI LIVELLI
6500 NP=NP+1:LV=LV+1:PT=PT+500:TH=TH+1
6501 IF PT>=EXTRA THEN EXTRA=EXTRA+15000:LF=LF+1
6504 IF TH>14 THEN TH=15:IF NP>19 THEN NP=19
6505 PEN 2:LOCATE 11,4: PRINT"--------------------"
6510 LOCATE 11,5: PRINT"-- GO TO LEVEL";LV;"----"
6520 LOCATE 11,6: PRINT"--------------------"
6525 FOR T=1 TO 1000: NEXT T
6530 LOCATE 11,7: PRINT"-- R U READY? Y/N ---"
6540 LOCATE 11,8: PRINT"--------------------":PEN 1
6550 RE$ = INKEY$
6560 IF RE$="Y" OR RE$="y" THEN GOTO 6570
6565 GOTO 6550
6570 REM FOR T=4 TO 8
6580 REM LOCATE 11,T: PRINT"                    "
6590 REM NEXT T
6591 CLS
6600 RETURN

7000 REM DISEGNA LO SFONDO
7001 REM AL=ALEATORIO - NP=NUMERO PALAZZI - P=COLORE DELLA PENNA
7002 SF$="00000000000000000000000000000000000000000"
7008 PEN 2:LOCATE 1,24: PRINT"LEVEL:"; LV ;"- LIVES:"; LF ; "- POINTS:"; PT
7009 PEN 1: LOCATE 10,25:PRINT "- HI-SCORE:"; HISCORE; "-"
7010 X=20-(NP/2): PEN 2
7020 FOR N=1 TO NP
7021   REM LOCATE 2,2: PRINT X
7022   REM CAMBIA IL TOP DEL PALAZZO CASUALMENTE X=INT(RND(1)*(high-low))+low
7025   A=INT((RND(1)*4)+1)
7026   PT$=CHR$(239 + A)
7030   HI=INT(RND(1)*(TH-2))+2
7031   OLDX=X
7041   MID$(SF$,X,1)="1"
7042   X=OLDX
7050   FOR I=HI-1 TO 0 STEP -1
7051     REM CAMBIA IL PIANO DEL PALAZZO CASUALMENTE X=INT(RND(1)*(high-low))+low
7052     A=INT((RND(1)*6)+1)
7053     PF$=CHR$(229 + A)
7063     rem P=INT((RND(1)*3)+1):PEN P
7065     IF I=HI-1 THEN GOTO 7066 ELSE GOTO 7070
7066     LOCATE X,23-I
7067     IF COPYCHR$(#0)<>PF$ THEN PRINT PT$: GOTO 7090
7070     LOCATE X,23-I: PRINT PF$
7090   NEXT I
7091 X=X+1
7100 NEXT N
7105 REM PEN 2: LOCATE 1,24: PRINT"LEVEL:"; LV ;"- LIVES:"; LF ; "- POINTS:"; PT: PEN 1
7110 PEN 1
7300 RETURN

9010 REM ******************************************
9011 REM * Future Set on Amstrad CPC
9012 REM * original code by Pete White
9013 REM * Popular Computing Weekly 713 August 1983
9014 REM *
9015 REM * Typed and corrected by
9016 REM * Francesco Fiorentini on June 2020
9017 REM * RetroMagazine World July 2020
9018 REM ******************************************
9020 SYMBOL AFTER 32
9030 REM Upper case chars
9040 SYMBOL 65,126,66,66,126,98,98,98,0
9050 SYMBOL 66,126,66,66,126,98,98,126,0
9060 SYMBOL 67,126,64,64,96,96,96,126,0
9070 SYMBOL 68,254,66,66,98,98,98,254,0
9080 SYMBOL 69,126,64,64, 120,96,96,126,0
9090 SYMBOL 70,126,64,64,120,96,96,96,0
9100 SYMBOL 71,126,64,64,102,98,98,126,0
9110 SYMBOL 72,66,66,66,126,98,98,98,0
9120 SYMBOL 73,60,16,16,24,24,24,60,0
9130 SYMBOL 74,126,8,8,24,24,24,120,0
9140 SYMBOL 75,68,68,68, 120,100,100,100,0
9150 SYMBOL 76,64,64,64,96,96,96, 126,0
9160 SYMBOL 77,126,74,74,98,98,98,98,0
9170 SYMBOL 78,98,82,74,102,98,98,98,0
```

```
9180 SYMBOL 79,126,66,66,98,98,98,126,0
9190 SYMBOL 80,126,66,66,126,96,96,96,0
9200 SYMBOL 81,126,66,66,98,98,106,126,4
9210 SYMBOL 82,126,66,66,126,106,100,98,0
9220 SYMBOL 83,126,64,64,126,6,6,126,0
9230 SYMBOL 84,126,16,16,24,24,24,24,0
9240 SYMBOL 85,66,66,66,98,98,98,126,0
9250 SYMBOL 86,66,66,66,66,66,36,24,0
9260 SYMBOL 87,66,66,66,98,106,106,126,0
9270 SYMBOL 88,102,102,36,24,36,102,102,0
9280 SYMBOL 89,66,66,126,16,24,24,24,0
9290 SYMBOL 90,126,4,8,16,32,64,126,0
9295 REM Lower case chars
9300 SYMBOL 97,0,0,126,6,126,70,126,0
9310 SYMBOL 98,96,96,96,126,98,98,126,0
9320 SYMBOL 99,0,0,126,96,96,96,126,0
9330 SYMBOL 100,6,6,6,126,70,70,126,0
9340 SYMBOL 101,0,0,126,98,126,96,126,0
9350 SYMBOL 102,60,48,48,120,48,48,48,0
9360 SYMBOL 103,0,0,126,70,70,126,6,126
9370 SYMBOL 104,96,96,96,126,98,98,98,0
9380 SYMBOL 105,24,0,24,24,24,24,24,0
9390 SYMBOL 106,6,0,6,6,6,6,6,126
9400 SYMBOL 107,96,96,102,108,120,108, 102,0
9410 SYMBOL 108,24,24,24,24,24,24,24,0
9420 SYMBOL 109,0,0,126,90,90,66,66,0
9430 SYMBOL 110,0,0,108,114,98,98,98,0
9440 SYMBOL 111,0,0,126,102,102,102,126,0
9450 SYMBOL 112,0,0,126,98,98,126,96,96
9460 SYMBOL 113,8,0,126,70,70,126,6,6
9470 SYMBOL 114,0,0,108,114,96,96,96,0
9480 SYMBOL 115,0,0,126,96,126,6,126,0
9490 SYMBOL 116,24,62,24,24,24,24,30,0
9500 SYMBOL 117,0,0,102,102,102,102,126,0
9510 SYMBOL 118,0,0,102,102,102,60,24,0
9520 SYMBOL 119,0,0,66,66,90,90,126,0
9530 SYMBOL 120,0,0,198,104,16,104,198,0
9540 SYMBOL 121,0,0,102,102,102,126,6,126
9550 SYMBOL 122,0,0,126,12,24,48,126,0
9555 REM Numbers
9560 SYMBOL 48,126,102,110,118,102,102,126,0
9570 SYMBOL 49,24,56,24,24,24,24,126,0
9580 SYMBOL 50,126,2,2,126,96,96,126,0
9590 SYMBOL 51,126,2,2,30,6,6,126,0
9600 SYMBOL 52,96,96,96,96,104,126,8,8
9610 SYMBOL 53,126,64,126,6,6,6,126,0
9620 SYMBOL 54,126,64,64,126,98,98,126,0
9630 SYMBOL 55,126,2,4,62,16,32,64,0
9640 SYMBOL 56,126,66,66,126,66,66,126,0
9650 SYMBOL 57,126,66,66,126,6,6,6,0
9680 SYMBOL 95,0,255,0,0,0,0,0,0
9690 RETURN

10000 REM INTRO GRAFICA
10005 MODE 1
10010 LOAD "ALIEN2.SCR",&C000
10015 PEN 2
10020 FOR I=1 TO 1000:NEXT I
10030 LOCATE 15,22:PRINT"ALIEN ATTACK!"
10040 LOCATE 8,24:PRINT "2020 - Francesco Fiorentini"
10045 PEN 1
10050 FOR I=1 TO 3000:NEXT I:CLS
10060 LOCATE 5, 5: PRINT "Our planet is dying."
10061 LOCATE 5, 6: PRINT "Our species is in danger."
10062 LOCATE 5, 7: PRINT "Our future is in danger."
10063 LOCATE 5, 8: PRINT "Our only purpose is survival."
10064 LOCATE 5, 9: PRINT "Whatever the cost..."
10065 LOCATE 5, 11: PRINT "We do not want to live together."
10066 LOCATE 5, 12: PRINT "We do not want to live together."
10067 LOCATE 5, 13: PRINT "We want the Earth!"
10068 LOCATE 5, 14: PRINT "Whatever the cost..."
10069 LOCATE 5, 16: PRINT "Whatever the cost!"
10070 LOCATE 5, 18: PRINT "This means WAR!"
10075 FOR I=1 TO 3000:NEXT I:
10080 RETURN
```

Introduzione ad ARexx – quinta parte

di Gianluca Girelli

GAME CODING CON AREXX - PARTE 2 di 2

Concludiamo in questo numero la carrellata sul linguaggio ARexx pubblicando la seconda ed ultima parte del tutorial su come implementare una avventura testuale. Ricordo che per meglio capire gli argomenti che verranno trattati consiglio, per chi non l'avesse fatto, di riprendere le parti precedenti pubblicate su RetroMagazine nei numeri 20, 22, 23 e 24, nonché l'articolo "Cenni di Game Coding" pubblicato sul numero 17.

Alla fine di questo tutorial avremo appreso tutte le basi per costruire un motore di gioco per avventure testuali: passare da una all'altra sarà "solo" una questione di cambiare l'ambientazione della storia (ad esempio western, piuttosto che cyberpunk)
e le interazione sugli oggetti che ne guidano il progresso.

1. GESTIRE GLI EVENTI

Nei numeri passati abbiamo imparato come costruire fisicamente il mondo di gioco (per semplicità abbiamo deciso di rappresentarlo come una matrice bidimensionale), come descriverne le locazioni (anch'esse raccolte all'interno di una seconda matrice, complementare alla prima), come navigare in tale mondo (gestione delle variabili di movimento) e come prepararci ad interagire con esso (implementazione di un semplice parser sintattico). Quello di cui ora abbiamo bisogno è capire come effettivamente gestire gli eventi (ovvero farli accadere) per poter proseguire nella storia. Supponiamo, ad esempio, di iniziare a giocare: siamo all'interno di un banco dei pegni dove in vetrina è esposto hardware pregiato, indispensabile per "vivere" nel mondo cyberpunk che stiamo costruendo.
La meccanica classica che sta dietro ad un'avventura testuale è: leggere (o rileggere tramite adeguato comando) la descrizione della locazione; esaminare il luogo nel dettaglio per capire se ci sono elementi da prelevare o con cui interagire; prendere (usare) detti elementi; se presi, esaminarli per scoprire eventuali informazioni nascoste. Tradotto in codice:

```
/*-------------------------*/
/*     GUARDA              */
/*-------------------------*/
Guarda:
  call Scenario(Pos)
  say
return
```

N.B. L'implementazione della routine Scenario() (che rimanda a video la descrizione della locazione attuale) è stata pubblicata sul numero precedente.

Come visto in precedenza, la penultima riga della descrizione di ogni locazione (contenuta nella precitata matrice) può contenere informazioni importanti, in questo caso:

Situazione.6.34='In una delle vetrine puoi vedere il tuo DECK "Ono-Sendai", modello UXB 7000.'

Prendiamo ora quindi l'oggetto:

```
/*-------------------------*/
/*     PRENDI              */
/*-------------------------*/
Prendi:
.....
.....
if Pos=34 & nome='DECK' then do
   og_num.1.1=1
   say 'Visto che avevi il ticket con te,
Shin ti rida'' il deck, ma non'
   say 'prima di averti "prelevato" 50
crediti!!'
   og_num.5.1=0
   og_num.6.1=0
   Situazione.6.34=''
end
if Pos~=34 & nome='DECK' & og_num.1.1=0 then
say 'Un biglietto che hai in tasca ti dice
che il DECK e'' da Shin.'
.....
.....
return
```

Per chiarezza riporto che:

	1	2	3	4	5	6	7	8	
1	game over description	**Panthers: no help ma software**	**Bassifondi**	**Body Bank: corpo, botta in testa. Targhetta dice Edison Carter**	**x**	**Residential 1: cadavere esploso, TV accesa**	**Residential 2: gente ipnotizzata, TV accesa**	game won description	1
2	vicolo: giornale	strada	strada	strada	strada	strada		**train station to spaceport – voli sospesi**	2
3	vicolo	**Arcade**	strada	**First Orbital Bank**	**x**	**Bank of Zurich: Flatline.**		senso/net	3
4	**Chatsubo: 50c, pawn ticket**		strada	dunking donuts, police jack- se ti connetti ti arrestano	tactical police, info su casi in residential 1 e 2	strada	strada	Net 23 – 4° piano – attacco finale	4
5	x	**Shin Pawn Shop: cyberdeck UXB**	downtown ovest	downtown est	strada	strada	Cray computing, 2° attacco- visione 3D del cyberspace	Net 23 – 3° piano -3° attacco	5
6	x	**Cheap Hotel: hacking software lev1**	strada	Amiga R&D, 1° attacco, blipvert project	Maas Biolabs, ripristino fisico per cyberspace	strada	Hosaka computing, nuove interfacce neurali	Net 23 – 2° piano	6
7	x	**Musabari Grill**	strada	strada	strada	piazza	strada	Net 23 – 1° piano	7
	1	2	3	4	5	6	7	8	

Figura 1 - La mappa di gioco

og_nome.1=deck
og_nome.5=crediti
og_nome.6=ticket

Inoltre, grazie alla potenza e alla flessibilità di ARexx nel gestire le variabili, abbiamo implementato un costrutto dove, se og_nome.1=deck allora: og_num.1.1=1 significa "deck preso", og_num.1.2=1 significa "deck esaminato" mentre og_num.1.3=1 significa "deck usato".

Senza entrare troppo nel dettaglio delle variabili, la storia inizia con il protagonista che possiede solo due oggetti: una ricevuta di un Banco dei Pegni ed un pò di spiccioli. Lo script prevede che se viene usato il comando "prendi deck" all'interno della locazione 34 (Negozio dei Pegni di Shin, confronta la mappa in figura) al protagonista venga restituito il suo "cyberspace deck" in cambio della ricevuta ed ovviamente dei soldi. Qualsiasi altro comando che usi la parola "deck" al di fuori della locazione 34 informerà il giocatore che l'oggetto sta da un'altra parte, sempre che il protagonista abbia ancora la ricevuta in tasca, altrimenti l'oggetto è evidentemente già stato preso. Infine, è importante notare come la routine Prendi() si occupi anche di azzerare la riga 6 all'interno della descrizione della locazione attuale (Situazione.6.34='') in tal modo rientrando da Shin (o usando il comando "Guarda") non verrà più menzionato il deck all'interno dello scaffale.

Ricevuto l'oggetto è quindi ora il caso di esaminarlo da vicino:

```
/*-------------------------*/
/*     ESAMINA             */
/*-------------------------*/
Esamina:
select
....
....
   when nome='DECK' & og_num.1.1 then
      do
       say 'E'' il tuo buon vecchio cyberspace
deck Ono-Sendai UXB 7000. Con questo puoi
collegarti ad un JACK.'
        og_num.1.2=1
```

```
      if regeneration=0 then say 'Ma non
sei ancora pronto.'
     end
....
....
otherwise say "Non hai questo oggetto con
te"; say; return
end
return
```

Di fatto ora abbiamo il deck (og_num.1.1=1, inizializzato nella routine "prendi") e lo abbiamo esaminato (og_num. 1.2=1, inizializzato nella routine "esamina"). Come in questo caso, a seconda dello stato di altre variabili, ci sono cose che possiamo già fare oppure no. Nel prosieguo della storia, una volta recuperato l'harware, il giocatore avrà bisogno di recuperare anche adeguato software che però, ancorchè combinato con il deck, non permetterà al protagonista di entrare nel cyberpazio finchè egli non sarà guarito da un precedente danno neurale (cioè sino a quando la variabile "regeneration" verrà inizializzata ad 1).

Supponiamo ora di aver reperito, dopo aver visitato alcuni ambienti, un dischetto contenente il software che ci serve. Abbiamo quindi bisogno di una routine che ci permetta di usarlo:

```
/*-------------------------*/
/*       USA               */
/*-------------------------*/
Usa:
....
....
if nome='DISCHETTO' & og_num.8.1 & og_num.
1.1 then do
     say
     say 'Hai inserito il dischetto nel deck
e caricato il software.'
     og_num.8.3=1
     end
else do
     if (nome='DISCHETTO' & og_num.8.1=0)
then say 'Non hai il DISCHETTO con te.'
     if (nome='DISCHETTO' & og_num.8.1 &
og_num.1.1=0) then say 'Non hai un DECK in
cui caricarlo.'
     end
if nome='DECK' & og_num.1.1 then say 'Il DECK
non puo'' essere usato direttamente. Devi
trovare un JACK.'
if nome='DECK' & og_num.1.1=0 then say 'Non
hai questo oggetto con te.'

if nome='JACK' then call Combat(Pos)
....
....
return
```

Come si intuisce, nel caso il dischetto sia in nostro possesso (og_num.8.1=1) esso verrà caricato nel deck (og_num. 8.3=1). Nel caso mancasse un elemento (il deck o il dischetto stesso) la procedura gestirà la relativa eccezione. Usare l'oggetto giusto nel posto giusto è la chiave delle avventure testuali pertanto la routine "usa" si occupa anche di invocare (qualora le condizioni al contorno lo consentano) la routine di "combattimento". Essa può non essere sempre necessaria e può assumere varie forme. Nel nostro caso si occupa di simulare un hacking al sistema semplicemente stampandone a video le fasi.
L'uso delle variabili "hack_1_disp" e "first_attack" ci aiuta a tenere traccia della situazione per monitorare la progressione degli eventi.
Altre variabili dalle analoghe sintassi e funzioni verranno inizializzate a mano a mano che il giocatore scopre nuovi oggetti che gli permetteranno di potenziare il suo equipaggiamento e di proseguire nella storia sino alla sua conclusione.

```
/*-------------------------*/
/*   COMBAT (hacking)      */
/*-------------------------*/
Combat:
select
   when Pos=44 & hack_1_disp then do          /
* Amiga R&D */
       say cancella
        say 'Inizio Hacking di Primo Livello'
       call Sleep()
       say
       say 'Accesso al CyberSpace. Violazione
in corso ....'
       say 'Trovati dati su: BLIPVERT PROJECT'
       say 'Gli esperimenti effettuati da
questa divisione Ricerca e Sviluppo confermano
la'
```

```
      say 'risposta positiva dell''individuo
ai segnali subliminali. Nascondendo la foto
di un'
      say 'prodotto all''interno di un
filmato, il soggetto provera'' un forte
impulso di procurarselo.'
     say 'Abbiamo la tecnologia per comprimere
il messaggio lungo tutto il segnale portante
della TV'
     say 'per mandare un messaggio subliminale
continuo.'
      say
'=========================================
==='
      say 'Hacking terminato. Uscita dal
CyberSpace effettuata.'
      first_attack=1
   end
....
....
 otherwise say 'Non sei ancora pronto.'
end
return
```

2. TENERE TRACCIA DELLA PROGRESSIONE DELLA STORIA

Il nostro lavoro è quasi concluso: l'unica cosa che ancora manca è un modo per tenere traccia dei progressi ottenuti dal giocatore in modo che possa avanzare nella storia a mano a mano che i compiti assegnati sono stati completati. Dal momento che questo software è stato sviluppato sostanzialmente per motivi didattici, la procedura di cui abbiamo bisogno sarà molto semplice. Il suo livello di complessità però può essere liberamente aumentato secondo le più diverse necessità. Vediamo un esempio:

```
/*-------------------------*/
/*    GAME PROGRESS        */
/*-------------------------*/
GameProgress:
 if regeneration & og_num.1.1 & og_num.8.3
then do
       hack_1_disp=1
       say '=========================='
       say 'Ora sei pronto per entrare nel
cyberspazio. L''hacking di primo livello e''
ora disponibile.'
       say '=========================='
end
 if first_attack & og_num.3.3 then do
       hack_2_disp=1
       say '=========================='
        say 'Ora sei pronto per entrare nel
cyberspazio 3D. L''hacking di secondo livello
e'' ora disponibile.'
       say '=========================='
end
....
....
return
```

GameProgress() lavora congiuntamente con altre routine del gioco e aggrega i risultati da loro prodotti per abilitare nuove opzioni permettendo così di avanzare nella storia. Abbiamo visto nel paragrafo precedente che il giocatore, dopo aver recuperato adeguato hardware (deck), software (dischetto) ed essere guarito dal danno neurale (regeneration), aveva raggiunto i requisiti minimi per progredire nel gioco. Ciò è stato ottenuto proprio grazie al primo IF-THEN di questa routine che, dopo aver verificato l'esistenza di tali requisiti, ha inizializzato la variabile "hack_1_disp". Se andate ora a rileggere il codice della routine Combat() vi apparirà chiaro il perchè alla fine del primo attacco sia stata inserita la variabile "first_attack": essa è uno dei prerequisiti del secondo livello. L'altro è il reperire ed usare un nuovo oggetto (nel nostro caso una nuova interfaccia neurale, og_num.3.3), cosa che ci abiliterà all'hacking successivo.

GameProgress() potrebbe fare molte altre cose, ad esempio aggiornare il sistema di punteggi (nel caso ci sia, come nelle vecchie avventure della Infocom) o assegnare "achievements" e "trofei" (come per XBox e PlayStation). All'aumentare della complessità e della lunghezza della nostra avventura potrebbe poi aver bisogno di aggiornare l'ambiente di gioco durante il run time. Il motivo è dovuto al fatto che il contenuto originale di tutte le variabili (ed in questo caso il contenuto integrale delle matrici che descrivono il mondo di gioco) deve sempre essere preservato. In caso diverso, successivamente al primo salvataggio tali informazioni andrebbero irrimediabilmente perse (poichè sovrascritte) e non ci sarebbe modo di giocare una seconda partita. Una soluzione al problema è espandere la matrice delle descrizioni in modo trasparente all'utente, aggiungendo le descrizioni aggiuntive in locazioni non raggiungibili dal

giocatore. In questo modo il software potrà caricare il mondo originale ad ogni partita preservandolo nel tempo, mentre GameProgress() conterrà istruzioni dedicate che modificheranno tale mondo solo durante il runtime. Nel nostro caso, ad esempio, la routine potrebbe contenere un'istruzione del tipo:

if og_nome.1=1 then Situazione.6.34='Lo scaffale ora è vuoto.'.

Poichè GameProgress() è invocata ad ogni ciclo siamo sicuri che essa rispecchierà fedelmente i nostri progressi senza però alterare il costrutto originale.

3. INIZIALIZZAZIONE E MAIN LOOP

Dopo aver scomposto l'engine in tutte le sue parti ed averle realizzate una ad una, riporto di seguito l'intera sequenza di inizializzazione

e il main loop del gioco presa integralmente dal sorgente dell'avventura:

```
/*====================== GAME INIT
=======================*/
do until (risposta='NO' | risposta='N')

  call Livelli()          /* imposta descrizione
livelli */
  call Variabili()         /* inizializza
variabili di gioco */
  call Titoli()           /* routine di
presentazione */
  call Istruzioni()        /* visualizza le
istruzioni di gioco */
  call Scenario(Pos)       /* stampa situazione
e posizione */
  options prompt ">"

/* inizia main routine */

do until (frase='QUIT')
  say
  richiesta=random(1,4,time("E")*100)  /*
tra 1 e 4 compresi */
  say prompt.richiesta
  pull frase
  call Parser(frase)
  select
       when verbo='VAI' then call Vai(nome, Pos)
       when verbo='GUARDA' then call Guarda()
       when verbo='PRENDI' then call
Prendi(nome)
       when verbo='ESAMINA' then call
Esamina(nome)
        when verbo='USA' then call Usa(nome)
        when verbo='LISTA' then call Lista()
      when verbo='VOC' then call Vocabolario()
        when verbo='DEBUG' then call Debug()
        when verbo='VAR' then call Var(nome)
        when verbo='QUIT' then nop
        otherwise say 'non so che significa
' || verbo
  end
  call GameProgress()
  if ~tesoro_trovato then
numero_mosse=numero_mosse+1
end

  say 'un''altra partita?'
  pull risposta
end
/* fine main routine */

say cancella
say 'grazie per aver giocato'
say
say 'hai fatto ' numero_mosse 'mosse'; say
do j=1 to 7
  totale=totale+punteggio.j
end
say 'il tuo punteggio e'':' totale; say
if tesoro_trovato~=1 then say '....ma non
hai risolto il rebus del BlipVert Project!'
call Sleep()
call Crediti()
exit
/*======================== GAME FINISHED
=======================*/
```

Come si può notare, l'aver scomposto il problema nelle sue parti elementari permette di scrivere un main loop estremamente semplice da leggere e da manutere: dopo la fase di inizializzazione si entra nel loop di gioco, all'interno del quale il parser dirigerà le azioni chiamando le routine adeguate a seconda degli input dell'utente. Ad ogni ciclo GameProgress() gestisce i progressi del giocatore. All'uscita dal gioco verrà comunicato all'utente il suo punteggio e

verranno visualizzati i crediti.

4. ULTERIORI ELEMENTI PER IL CODICE DI GIOCO

Come detto in precedenza questo motore di gioco per avventure testuali fu sviluppato anni fa per motivi essenzialmente didattici. Per questa ragione manca di alcuni elementi che sarebbero invece indispensabili nel caso si voglia confezionare un prodotto quantomeno semi-professionale. La cosa più importante è decisamente la realizzazione delle routine di caricamento e salvataggio del gioco: è infatti impensabile pretendere che l'utente possa arrivare alla fine dell'avventura rapidamente e senza bisogno di pause intermedie. Un'altra cosa importante è poi la localizzazione del software: il sistema di matrici contenenti le descrizioni ed il parser andrebbero quindi sviluppati congiuntamente ad un insieme di variabili di supporto che permettano di evitare l'uso delle stringhe "hard-coded" e quindi, a titolo di esempio, la routine "esamina" andrebbe scritta così:

```
/*-------------------------*/
/*     ESAMINA             */
/*-------------------------*/
Esamina:
select
....
....
   when nome=var_deck & og_num.1.1 then
      do
        say stringa1
        og_num.1.2=1
        if regeneration=0 then say stringa2
      end
....
....
otherwise say stringa3; say; return
end
return
```

Dove "var_deck" contiene il nome dell'oggetto, mentre stringa1, stringa2 e stringa3 contengono le descrizioni delle azioni. Queste stringhe sono contenute nel file di localizzazione specifico per la lingua scelta dal giocatore dal menù principale pre gioco.

Infine, potremmo voler espandere il gioco con side quests, un sistema di punteggi o con un vocabolario più variegato che, unito ad un parser più raffinato, possa fornire all'utente un'esperienza più immersiva.

5. CONCLUSIONI

Il viaggio è stato lungo ma siamo finalmente arrivati alla nostra meta.
Spero che vi siate divertiti a leggere quanto io mi sono divertito a progettare il gioco ed a realizzare questa serie di articoli.
Il codice è troppo lungo per essere inserito nell'articolo ma l'intero sorgente è disponibile all'interno del file .zip pubblicato nell'area download del sito www.retromagazine.net, all'indirizzo contenuto all'interno del box immediatamente alla fine di questo articolo
Per giocarlo sarà sufficiente importarlo nel vostro emulatore preferito o, ancora meglio, sull'hardware reale se ne siete un fortunato possessore.
Qualsiasi versione di AmigaOS, a partire dalla versione 2, basterà allo scopo.

Link al materiale:

- https://www.retromagazine.net/download/ARexx5.zip

BIBLIOGRAFIA

- Mike Cowlishaw "The REXX Language: A Practical Approach to Programming" (1985) Prentice Hall. ISBN 0-13-780651-5.
- Chris Zamara, Nick Sullivan "Using Arexx on the Amiga" (1991) Abacus Software Inc. ISBN 1-55755-114-6.
- AmigaOS 2.0 Manuale di sistema
- http://it.wikipedia.org/wiki/Game_engine
- http://it.wikipedia.org/wiki/Porting

Utilizzo degli esempi

Come riportato nell'articolo sul num. 20, per utilizzare gli esempi bisogna salvare lo script in modalita' testo nel formato "nome_script.rexx".
Per lanciare lo script basta digitare da shell:
">rx nome_script.rexx"
o, piu' semplicemente:
">rx nome_script".

Cross-programmazione su Olivetti M20 in C

di Davide Bucci (traduzione ed adattamento di David La Monaca)

1. Introduzione

Il mese scorso abbiamo pubblicato un'introduzione all'Olivetti M20, una macchina piuttosto particolare del 1982 [1]. Abbiamo trascorso un po' di tempo a discutere le caratteristiche e le “stranezze” del PCOS (Professional Computer Operating System). Quest'articolo descrive invece come utilizzare un compilatore C relativamente moderno (nello specifico, una versione di GCC) per sviluppare software per l'M20.

Uno dei vantaggi di sfruttare i moderni computer e strumenti per programmare computer d'epoca è che ora abbiamo bellissimi editor di testo, eccellenti compilatori e linguaggi molto efficienti. Un purista potrebbe sostenere che, così facendo, la vera "esperienza di programmazione degli anni '80" va inevitabilmente a perdersi, ma il rovescio della medaglia è che ci sono innumerevoli vantaggi a compenso di questa perdita, specialmente quando si ha a che fare con progetti relativamente grandi. Negli ultimi mesi ho utilizzato questo approccio per sviluppare alcune avventure testuali. Sono stato in grado di sfruttare la portabilità del linguaggio C, puntando su sistemi diversi come il Sinclair ZX Spectrum e l'Olivetti M20 utilizzando quasi lo stesso codice sorgente.

In questo articolo descriverò come eseguire la cross-compilazione per l'M20 in C su un sistema operativo Unix-like come Linux o MacOSX. Non ho una macchina Windows a portata di mano, ma credo che per quel sistema operativo, programmi come MinGW o Cygwin possano tornare utili. Una strategia conveniente è quella di avere sulla macchina di sviluppo sia un compilatore che un emulatore in esecuzione per ogni piattaforma target. Gli strumenti necessari (z8k-pcos GCC, m20disk, MAME) devono essere scaricati e in alcuni casi compilati a partire dai sorgenti, quindi spero che non vi sentiate scoraggiati dall’uso di programmi come GNU make.

Questo articolo è organizzato come segue: inizierò descrivendo brevemente il compilatore e l'emulatore MAME per M20. Poi mostrerò come usarli per compilare ed eseguire alcuni esempi introduttivi. Infine discuterò come trasferire gli eseguibili sull'hardware reale ed eseguirli. Discuterò infine un esempio non banale (una piccola demo grafica) prima di trarre alcune conclusioni.

2. Il cross-compilatore C e l'emulatore

Molti personal computer degli anni '80 potevano essere programmati in uno dei tanti dialetti BASIC disponibili. L'Olivetti M20 non faceva eccezione e venne fornito con un interprete Microsoft ragionevolmente completo, chiamato BASIC-8000. Anche se il BASIC era un linguaggio semplice e facile da imparare, in alcune situazioni era dolorosamente lento. Inoltre, non era molto conveniente per operazioni di basso livello, non era efficiente per progetti di grandi dimensioni e fortemente limitato in molti settori. Ho iniziato a programmare con BASIC sul mio VIC-20 quando ero un bambino e l'ho usato per molti anni anche sul PC, ma non mi piace molto. Una suite di assembler per lo Z8001 era disponibile per l'M20, ma la gestione di grandi progetti in assembly è spesso noiosa, ingombrante e il codice non è portabile, anche se si possono scrivere programmi estremamente compatti ed efficienti.

Da un punto di vista moderno, il linguaggio C offre un buon compromesso tra velocità di esecuzione, facilità di sviluppo ed efficienza complessiva su macchine limitate, essendo un linguaggio compilato straordinariamente efficiente. Non descriverò qui i punti di forza e le insidie del linguaggio C (molte risorse e tutorial sono disponibili su Internet per questo), ma esistono moderni compilatori che si rivolgono a processori a 8 e 16 bit. Ad esempio il cc65 per il 6502, lo z88dk per lo Z80, ecc. Per l'Olivetti M20, molto lavoro è stato fatto in questa direzione da Christian Groessler nel corso di diversi anni. Christian ha creato una versione di GCC 2.9 per il processore Zilog Z8001 e PCOS, da un compilatore originariamente messo insieme nel 1998 dal gruppo eCos (allora parte di RedHat). Il suo lavoro includeva sia GNU binutils che newlib.

GCC 2.9 non supporta tutte le caratteristiche e le novità dei recenti compilatori standard per C e C++, ma è comunque uno strumento molto ben fatto, molto più potente dell'originale Microsoft BASIC disponibile sulla macchina. Christian distribuisce il compilatore insieme ai suoi sorgenti per molti sistemi Unix sul suo sito FTP [2], e ha scritto un articolo introduttivo, disponibile al link [3].

Una possibile strategia per installare il compilatore è quella di usare una delle distribuzioni binarie disponibili (Christian ha gentilmente preparato dei pacchetti pronti da usare per molte versioni di Un*x) oppure di compilarlo direttamente dai file sorgenti. Una volta fatto tutto, dovreste installare gli eseguibili in /usr/local/bin o assicurarvi che siano raggiungibili tramite il percorso corrente della shell. Se l'installazione è riuscita, digitando il seguente comando si dovrebbe ottenere la versione del compilatore, come segue:

```
$ z8k-pcos-gcc --version
2.9-ecosSWtools-990319-m20z8k-3
```

La suite del compilatore è composta da una collezione di

strumenti che appaiono subito familiari se si è abituati ad usare GCC. Ci sono versioni dedicate agli eseguibili COFF, ma non li useremo su M20. Gli strumenti dedicati a PCOS iniziano con il prefisso z8k-pcos.

Probabilmente, il modo più conveniente per lo sviluppo di software per un computer d'epoca è quello di avere, sulla macchina moderna di sviluppo, un efficiente compilatore abbinato a un buon emulatore. Il secondo strumento che useremo è quindi MAME, poiché a partire dalla versione v0.212, ha iniziato a supportare parzialmente l'M20. L'implementazione è ancora leggermente instabile, ma resta sufficientemente utile per testare rapidamente programmi brevi e semplici. Benjamin Eberhardt ha scritto un articolo molto interessante su come usare MAME per emulare un M20 [4].

MAME può essere scaricato da questo link [5] e molti degli sforzi fatti per emulare M20 sono stati fatti (ancora una volta) da Christian Groessler. Dopo il download, è necessaria una copia del codice ROM di avvio che si trova al link [6], così come un'immagine di un disco di avvio contenente PCOS, come quello presente negli archivi associati a questo articolo [7]. Una volta che MAME è installato sul vostro sistema e avete messo la ROM M20 nella directory corrente, può essere lanciato con un comando che ha la seguente struttura:

```
$ mame m20 -bios 0 -rompath . -flop1 <image1> -flop2 <image2> -window
```

Ora che gli strumenti principali di cui abbiamo bisogno sono pronti, nel prossimo paragrafo, scriveremo, compileremo ed eseguiremo alcuni semplici programmi in C sull'emulatore.

3 - Tre programmi 'Hello World'

Naturalmente, il primo programma che si può utilizzare per testare la toolchain del compilatore è il ben noto programma 'Hello World':

Figura 1: Output di Hello world in MAME

```
#include<stdio.h>

int main(int argc, char **argv)
{
    printf ("Hello World!");
    return 0;
}
```

Se chiamiamo questo file hello.c, il comando per compilarlo è:

```
$ z8k-pcos-gcc hello.c -o hello.cmd
```

Una sorpresa piuttosto spiacevole è che l'eseguibile è lungo ben 16211 byte. Se è piccolo per gli standard odierni, è invece relativamente grande per un computer del 1982 e questa dimensione non è accettabile per un programma così semplice. Dobbiamo rimediare a questo problema.

Il colpevole è la libreria standard e in particolare l'implementazione della funzione printf. Questa funzione offre capacità di formattazione molto flessibili, al prezzo di un codice sostanziale da includere nell'eseguibile. Vale la pena notare che, anche se il compilatore C supporta tipi a virgola mobile come double e float, l'attuale implementazione di scanf e printf non la gestisce. Per molti scopi pratici, tuttavia, se non si ha bisogno delle capacità di formattazione, printf può essere omesso del tutto. Un eseguibile più gestibile, lungo 9963 byte, può essere ottenuto a partire dal seguente codice:

```
#include<stdio.h>

int main(int argc, char **argv)
{
    fputs ("Hello World!", stdout);
    return 0;
}
```

Per ridurre ulteriormente le dimensioni del compilato, una tecnica interessante (che però rende il codice non portabile) è quella di sfruttare una chiamata diretta al sistema PCOS:

```
#include<sys/pcos.h>

int main(int argc, char **argv)
{
    _pcos_dstring("Hello World! \r");
    return 0;
}
```

Una volta compilato, questo codice produce un eseguibile di appena 2919 byte, certamente molto più in linea con quanto ci aspettavamo. Questa dimensione è ancora molto maggiore di quella ottenibile con un programma di assembly puro, ma si può considerare accettabile. Una

lista delle funzioni PCOS richiamabili da C si trova nell'intestazione di pcos.h, che segue da vicino la descrizione fatta da Olivetti nel manuale dedicato alla suite di linguaggi di assembly [8]. Le opzioni -Os e -O2 di gcc possono essere utilizzate e dicono al compilatore di ottimizzare il l'eseguibile rispettivamente in base al codice o alla velocità. In entrambi i casi, il semplice programma "Hello World!" produce un eseguibile di 2897 byte. Si noti nell'ultimo esempio l'uso di \r che corrisponde al carattere new line usato da PCOS al posto di \n.

Nella mia esperienza, è una buona pratica in C adottare una strategia modulare e tenere separate dal nucleo del programma le routine relative agli input e agli output. Quando si "porta" un programma relativamente grande, queste richiedono la maggiore quantità di lavoro. È lì che il codice non portabile (come la chiamata di sistema PCOS) può essere usato.

Se si vuole mescolare il codice Z8001 e C, o se si vuole usare il solo assembler z8001-pcos-as, ciò è assolutamente possibile. Il manuale del compilatore [9] include alcune istruzioni dettagliate su come farlo e contiene molti programmi di esempio. Se siete abituati all'assembly dello Z80, potreste trovare interessante imparare lo Z8001, in quanto doveva essere il successore a 16 bit dello Z80, sfruttando un paradigma di memoria segmentata e conservando un certo grado di compatibilità.

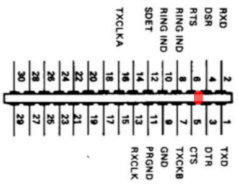
4 - Esecuzione di programmi in MAME

Per eseguire il programma Hello World descritto sopra, dobbiamo prima trasferirlo in un'immagine disco utilizzabile. MAME può leggere diversi tipi di immagini disco e il formato file più utile da usare con l'M20 ha l'estensione IMG (nelle vecchie versioni di MAME solo quelle in formato MFI possono essere scritte). Ci sono un certo numero di dettagli da considerare quando si creano immagini disco utilizzabili, a causa del formato head 0/track 0 che era diverso dal resto del disco. Come detto in precedenza, una buona immagine avviabile che può essere utilizzata con l'emulatore è il file pcos20.img che può essere scaricato da [7].

Avremo bisogno dell'utility m20floppy descritta in [10]. Scaricatela e compilatela con make, in modo da ottenere un eseguibile chiamato 'm20'. Una volta creato e installato nel vostro computer, per ottenere un'immagine disco chiamata hello.img, digitate:

```
$ m20 hello.img new
```

A proposito, questa utility supporta diversi comandi: lanciatela senza argomenti per ottenere una breve descrizione di ciascuno di essi. A questo punto, l'immagine del disco non è ancora utilizzabile, poiché m20floppy non crea il contenuto della head 0/track 0. Questo deve essere trasferito manualmente da un'immagine disco che li



Female DB9	M20 connector
2 (RXD)	1 (TXD)
3 (TXD)	2 (RXD)
6 (DSR)	3 (DTR)
4 (DTR)	4 (DSR)
7 (RTS)	5 (CTS)
8 (CTS)	6 (RTS)
5 (GND)	9 (GND)

Figura 2: Schema del cavo RS232

contiene. L'immagine disco example.img presente nello stesso archivio del disco PCOS può essere utilizzata per questo scopo:

$ dd conv=notrunc if=example.img of=hello.img bs=4096 count=1

Benjamin Eberhardt suggerisce in [4] un modo semplice per verificare se un'immagine disco contiene o no i dati corrispondenti alla head 0/track 0. Bisogna controllare i primi byte del file per vedere se sono diversi da zero. Se il comando dd ha avuto successo, ecco cosa si dovrebbe ottenere da un'immagine che può essere utilizzata con successo nell'emulatore:

$ hexdump hello.img |head -n 1
0000000 01 04 00 23 02 10 01 00 00 00 0a 00 c4 00 86 1e 00

ed ecco il risultato con un'immagine che non può essere utilizzata, in quanto mancano i dati della traccia 0:

$ hexdump bad.img |head -n 1
0000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Quando si dispone di un'immagine di un disco vuoto, è possibile copiare il risultato su un altro file, per evitare di dover ripetere il processo ogni volta. Possiamo quindi aggiungere il programma eseguibile all'immagine del disco:

$ m20 hello.img put hello.cmd

È possibile controllare il contenuto di un'immagine disco utilizzando il comando ls di m20floppy:

$ m20 hello.img ls
hello.cmd

Una volta che l'immagine del disco contiene l'eseguibile, lanceremo l'emulatore in una finestra, con un disco di sistema pcos20.mfi nel drive 0: e l'immagine hello.img nel drive 1:. Se entrambi i file sono disponibili nella directory corrente che contiene anche il file ROM m20.zip, il comando per lanciare MAME è il seguente:

$ mame m20 -bios 0 -rompath . -flop1 pcos20.img -flop2 hello.img -window

Se avete messo i file altrove, cambiate il loro percorso di conseguenza. Nell'emulatore, dopo che la macchina ha finito l'avvio, possiamo digitare 'hello' (o semplicemente 'he') e il programma Hello World dovrebbe essere eseguito, come mostrato in figura 1. Si può notare che non abbiamo dovuto cambiare l'unità corrente, poiché una delle stranezze di PCOS è che se un file non viene trovato nell'unità corrente, viene automaticamente scansionato anche l'altro. L'ultima unità scansionata diventa quella corrente. Se si hanno problemi con il layout della tastiera, è possibile attenuarli eseguendo 'sl' che permette di cambiare la lingua corrente. Il comando 'ps' salva la configurazione corrente dei PCOS e il salvataggio sarà permanente, poiché MAME può scrivere su file immagini nel formato img. Se si vuole avere descrizioni dei messaggi di errore più esplicite di un codice numerico, si può usare il comando 'ep', a spese di 1240 byte di RAM libera.

Lo stato attuale dell'emulazione MAME dell'M20 è che molte cose possono essere fatte, ma l'emulazione può essere instabile (un messaggio di avvertimento è infatti emesso da MAME in tal senso). L'emulatore è comunque prezioso per i test preliminari, in quanto il trasferimento di file su una macchina reale non è del tutto banale e richiede un po' di tempo e di fatica, come vedremo nel prossimo paragrafo.

5 - Trasferimento di file su un vero M20

Esistono diversi metodi disponibili per trasferire i file verso un vero Olivetti M20. Se si dispone di un computer MS-DOS con un drive per floppy disk da 360 KB, è possibile utilizzare le routine wrm20 e rdm20 di Dwight Elvey, come descritto in [11]. Ci sono limitazioni, principalmente a causa della particolare formattazione della head 0/track 0, che non è gestita da molti controller di dischi nel mondo dei PC. Di solito, un modo per aggirarle è quello di formattare un disco su M20 e scriverlo sul PC con gli strumenti di Dwight, che semplicemente saltano la traccia che non può essere scritta.

Nel mio caso, non possedendo un PC adatto, ho preferito realizzare un cavo null-modem RS232 per tentare il trasferimento dei dati con protocolli come XMODEM. La figura 2 mostra le connessioni del cavo. Ho rappresentato la numerazione dei pin in un connettore DB9 maschio come appaiono in questo modo sul lato a saldare del connettore femmina da utilizzare per il cavo. Sul lato "moderno", ho usato un'interfaccia USB-RS232 che ho acquistato molti anni fa, che funziona in modo affidabile con MacOSX. Ho scritto una piccola raccolta di utility in BASIC descritte in [12] che possono essere utilizzate per questo compito. Partire da zero può comportare la copia di un programma di ricezione XMODEM sulla M20 e poi utilizzarlo per trasferire gli strumenti più necessari. Invece di digitare direttamente il programma, una volta che l'M20 è collegato, si può reindirizzare l'ingresso e l'uscita del PCOS verso la porta RS232 con i seguenti comandi:

Figura 3: Trasferimento file in corso

```
pl ci
rs
sc com:,9600,none,0,8,half,off,256
ci 0,o,0
+Scom:, +Dcom:
```

I primi comandi caricano in memoria l'utility 'ci' e il driver RS232 e configurano l'M20 per una connessione a 9600 baud 8N1, senza eco né controllo XON/XOFF. Poi si apre una connessione seriale. Infine, l'ultimo comando reindirizza l'ingresso e l'uscita verso la RS232. Sul vostro moderno computer, se avete configurato correttamente il programma del terminale (io uso Minicom), dovreste vedere apparire sul vostro terminale il prompt PCOS che riproduce ciò che l'M20 scrive sullo schermo. Questo è un modo abbastanza comodo per usare l'M20, dato che potete controllare il computer a distanza. Potete per esempio lanciare l'interprete BASIC digitando 'ba' e copiare/incollare l'intero programma xreceive.bas. Per fare questo, dovreste prima configurare il vostro programma terminale per applicare un ritardo. Il BASIC non è abbastanza veloce per elaborare i dati spediti continuamente da un computer moderno e il risultato verrebbe stravolto dopo poche righe. Su Minicom, per esempio, digitate CTRL+A, poi T e impostate il "TX delay" su 10 ms. Si può salvare il file trasferito (come 'xmodem.bas'), poi riavviare la macchina e riemettere i primi quattro comandi visti sopra (poiché il reindirizzamento I/O non deve essere attivo per trasferire i file con XMODEM) e infine caricare ed eseguire 'xmodem.bas' all'interno di BASIC per trasferire i file. Ho inserito in [12] altri strumenti che possono essere utili.
La figura 3 mostra un trasferimento di file tra il mio MacBook Pro utilizzando Minicom e l'Olivetti M20, grazie ad un'interfaccia da USB a RS232 e al cavo che ho costruito. La figura 4 mostra il programma Hello World in esecuzione sulla mia macchina.

6 - Un esempio non banale: accesso alla memoria per la grafica

Naturalmente la programmazione in C offre molte possibilità e il codice della lista 1 mostra due funzioni e una macro

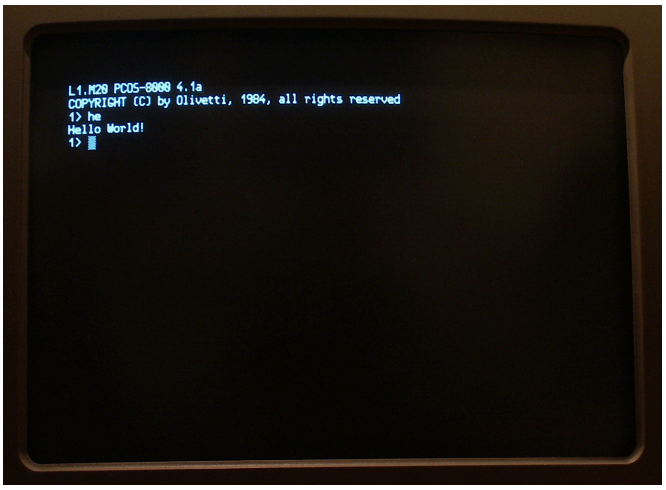


Figura 4: Hello World lanciato su un vero M20

che può essere utilizzata per disegnare sullo schermo accedendo direttamente alla memoria (su una macchina in B/N):

- La funzione "scrclear" cancella lo schermo (non c'è differenza tra la modalità grafica e quella testuale sull'M20, lo schermo visualizza sempre il modo grafico).
- La macro 'PSET_M' disegna un pixel su una griglia di 512x256.
- La funzione 'line' disegna un segmento con l'algoritmo di Bresenham [13].

Il risultato può essere visto nella figura 5. Naturalmente tale implementazione può essere migliorata, ma serve a dare un'idea dell'espressività del linguaggio C. Se si sente davvero il bisogno di sporcarsi le mani, il manuale gcc [9] descrive in dettaglio l'integrazione del codice C con l'assembly Z8001, prendendo ad esempio diverse versioni della funzione 'scrclr'. Se avete già familiarità con il linguaggio assembly dello Z80, potreste in effetti trovarvi a vostro agio con lo Z8001. Tra gli strumenti che vengono forniti con il compilatore GCC, l'assembler z8k-pcos-as è abbastanza potente e comodo.

Conclusione

In questo articolo abbiamo brevemente descritto come cross-programmare l'Olivetti M20, concentrandoci sul linguaggio C. Dopo una breve introduzione, abbiamo discusso gli strumenti che abbiamo scelto per il compito, vale a dire una versione speciale di GCC su misura per il processore Z8001 e il sistema operativo PCOS, nonché l'emulatore MAME.

Abbiamo poi introdotto il classico programma Hello World e abbiamo visto come ridurre le dimensioni dell'eseguibile prodotto dal compilatore. Abbiamo discusso come eseguirlo nell'emulatore e come trasferire i file su una macchina reale. Abbiamo finito la nostra discussione presentando un esempio di accesso diretto alla memoria dello schermo. Il manuale del compilatore [9] scritto da Chris vale sicuramente la pena di essere letto se si vuole andare oltre quanto descritto in questo articolo.

A proposito, quasi dimenticavo! Quest'articolo (come quello che avete letto il mese scorso) è stato scritto interamente usando Oliword sul mio Olivetti M20. I file di testo sono stati poi trasferiti utilizzando la porta RS232 su un moderno MacBook Pro, dove è stata fatta la modifica finale.

Tutto il codice sorgente discusso nell'articolo può essere scaricato all'indirizzo [7]. L'archivio contiene le immagini su disco degli esempi discussi e la versione Olivetti M20 (sono disponibili per molti computer a 8 e 16 bit di due giochi di avventura di testo da me sviluppati: "The Queen's Footsteps" e "Two Days To The Race").

Ringraziamenti

Vorrei ringraziare Christian Groessler per gli straordinari strumenti, l'impegno costante per l'M20, così come per le innumerevoli discussioni fruttuose che abbiamo avuto negli ultimi quindici anni. Per quanto riguarda l'emulatore MAME, non avrei mai potuto emulare un M20 senza l'aiuto di Benjamin Eberhardt, al quale vorrei esprimere la mia gratitudine.

Questo documento sarebbe stato probabilmente imbarazzante da leggere senza la gentile e attenta correzione di bozze di Chris Carter. I restanti errori sono miei.

BIBLIOGRAFIA E RIFERIMENTI

[1] D. Bucci "The Olivetti M20 and the history of a website" RMW #24-IT, June 2020 / RMW #02-EN, July 2020.
[2] C. Groessler, personal FTP site: ftp.groessler.org
[3] C. Groessler, D. Bucci "Cross-programming for the Olivetti M20 using GCC": http://www.z80ne.com/m20/index.php?argument=sections/download/z8kgcc/z8kgcc.inc
[4] B. Eberhardt, "Emulating the M20 with MAME": www.z80ne.com/m20/index.php?argument=sections/tech/mame_m20.inc
[5] MAME official website: https://www.mamedev.org
[6] Olivetti M20 ROMs available at https://wowroms.com/en/roms/mame/olivetti-l1-m20/89051.html
[7] https://www.retromagazine.net/download/m20inC_sources_and_disk_images.zip
[8] Olivetti "M20 Assembler language user guide," release 2.0, March 1983
[9] C. Groessler, "GCC Z8001 user manual," 2009: http://www.z80ne.com/m20/sections/download/z8kgcc/z8kgcc.pdf
[10] C. Groessler, "Manipulate disk images": http://www.z80ne.com/m20/index.php?argument=sections/transfer/imagehandle/imagehandle.inc

[11] D. Elvey "How to read and write disk images for the M20 system": http://www.z80ne.com/m20/index.php?argument=sections/transfer/imagereadwrite/imagereadwrite.inc
[12] D. Bucci "Transferring files using a RS232 connection": http://www.z80ne.com/m20/index.php?argument=sections/transfer/serial/serial.inc
[13] N. Johnson, "Advanced Graphics in C," ed. Osborne, McGraw-Hill 1987. http://www.z80ne.com/m20/index.php?argument=sections/download/z8kgcc/z8kgcc.inc

Listato 1: Codice C per l'accesso diretto alla memoria video

```
/* Segment #3: begin of video RAM for a B/W
monitor machine */
unsigned short *screen = (unsigned short
*)0x3000000;

#define SCREEN_WIDTH    512
#define SCREEN_HEIGHT   256
#define SCREEN_SIZE     (SCREEN_WIDTH / 16 *
SCREEN_HEIGHT)   /* words */
#define ABS(a) ((a)>0 ? (a):(-a))
#define MAX(a,b) (((a)>(b))? (a):(b))
#define SIGN(a) ((a)>0 ? 1 : ((a)==0 ? 0 :
(-1)))
#define TRUE -1
#define FALSE 0

/*  Fills screen mem with a defined word. */
void fillscr(unsigned short p)
{
    unsigned short *s;
    for (s=screen; s < screen+SCREEN_SIZE;
++s)
        *s = p;
}

/*  Just turn on a pixel by accessing
directly to the video RAM. */
#define PSET_M(x,y) *(screen + (((y)<<5) |
((x)>>4)))|=1<<(15-((x) & 0x000F))

/*  Plot a line using Bresenham algorithm.
    from N.Johnson, "Advanced Graphics in C"
    ed. Osborne, McGraw-Hill 1987. */
void line(unsigned short x1, unsigned short
y1,
    unsigned short x2, unsigned short y2)
{
    short dx=x2-x1, dy=y2-y1;
    short ix=ABS(dx), iy=ABS(dy);
    short inc=MAX(ix, iy), plotx=x1,
ploty=y1, i, plot;
    short x=0, y=0;
    PSET_M(plotx,ploty);  /* Plot the first
pixel */
    for(i=0; i<=inc; ++i) {
        x += ix;
        y += iy;
        plot=FALSE;
        if (x>inc) {
            plot=TRUE;
            x-=inc;
            plotx+=SIGN(dx);
        }
        if (y>inc) {
            plot=TRUE;
            y-=inc;
            ploty+=SIGN(dy);
        }
        if (plot)
            PSET_M(plotx,ploty);
    }
}

int main(int argc, char **argv)
{
    int i;
    fillscr(0);
    for (i=0; i<512; i+=10) {
        line(0,0,i,128);
        line(0,255,i,128);
        line(511,255,i,128);
        line(511,0,i,128);
    }
    return 0;
}
```

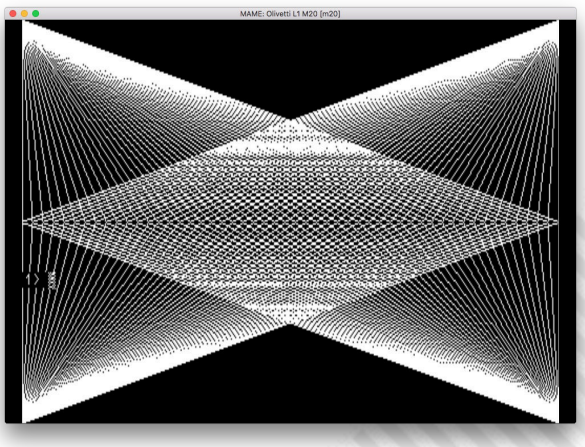


Figura 5: Il risultato prodotto dal listato 1

Giappone 13^puntata:
Nintendo G&W, sfida all'immortalità

di Carlo N. Del Mar Pirazzini, Giorgio Balestrieri, Michele Ugolini

Cari lettori, benvenuti in questa recensione speciale di RMW25.

Ecco a voi la scaletta degli argomenti trattati:
A) Introduzione
B) La collaborazione di Giorgio Balestrieri con MADrigal (aka Luca Antignano)
C) Mario's Cement Factory porting
D) GIG Tiger returns
E) Cuphead porting
F) Esistono altri simulatori ed emulatori?
G) Nintendo G&W 35th, sfida all'immortalità
H) Conclusioni

Parleremo di passato, presente e futuro dei G&W. Parleremo dell'immortalità di questi prodigi elettronici, nati dal genio giapponese e alimentati da numerosi appassionati che, a livello planetario, continuano a investire energia e denaro in questi oggetti sacri. Questa volta parleremo in due: l'articolo sarà steso da noi due redattori: Michele Ugolini e Carlo N. Del Mar Pirazzini.
Un terzo redattore di RMW , in questa occasione, verrà intervistato: le luci saranno puntate sul nostro Giorgio Balestrieri.

A) INTRODUZIONE
Michele:
Parlare di Nintendo in Giappone significa parlare delle solide basi costruttive di un loro edificio. Basi fondate su cemento armato. Strutture che poggiano su una base robusta, sebbene siano isolate da essa. Costruzioni che salgono al cielo utilizzando materiali sintetici altamente tecnologici, antisismici, made in Japan, dotati di un determinato anno di nascita ed un prestabilito anno di demolizione. Studi di materiali che ogni anno migliorano la propria tecnologia, reinventandosi, dopo aver assorbito i precedenti modelli matematici. Tremenda questa cosa dal punto di vista italiano: in Italia si riparano frequentemente le medesime abitazioni "in saecula saeculorum", spesso con le medesime tecniche ed i medesimi materiali degli anni passati. Perchè mi sto soffermando su questi punti? Il motivo è semplice, le dinamiche costruttive nipponiche rispecchiano un medesimo "modus operandi" anche sui G&W.
Da buon curioso ho sempre sbirciato e sono sempre rimasto affascinato dai cantieri edili nipponici, poiché ho avuto il "piacere" di sorbirmi tante scosse sismiche. Infatti spesso, trovandomi a svariati piani di altezza di un edifico, affacciandomi ai finestroni, rimanevo ammutolito, spaventato e meravigliato nel vedere ondeggiare i grattacieli tra loro, non come esili fili d'erba, ma come imponenti sfide mentali che contrastavano, in diretta, le possenti regole fisiche della Natura. Altrettanto solide e perfezioniste sembrano rivelarsi le basi costruttive dei G&W. In profondità i G&W sono ancorati alle proprie origini sacre, "ineluttabili" proprio come l'anima dei giapponesi. In superficie subiscono lo stress evolutivo di grafiche imponenti (vedi Playstation e Xbox, cfr. figura 1), incassando colpi su colpi , oscillando per colpa dei terremoti del marketing

Figura 1

moderno. Scosse su scosse, senza piegarsi al fallimento. Incredibile.
Sono sempre rimasto affascinato, oltre che dai loro edifici, anche dalla "stilizzata anima immortale" dei G&W, così imponente nella propria funzionalità, dotata di una tale sfrontata sicurezza che, a schiena diritta, può sfidare le possenti ed inesorabili leggi del Tempo! Difatti non è stata definita la data di demolizione, o meglio, di abbandono, di questo ambizioso ed imperturbabile progetto di ciclica reinvenzione dei G&W. Pertanto, sembra proprio essere una sfida all'immortalità: ecco spiegato il titolo di questa recensione. In questo articolo, con la collaborazione di Carlo N. Del Mar Pirazzini e Giorgio Balestrieri, verranno trattati vari temi inerenti ai Game&Watch: Discuteremo del solido sviluppo che ad oggi, 2020, sembra dirigersi seriamente verso il futuro.

Carlo N. Del Mar Pirazzini:
I primordiali anni 80, anni ruggenti. Nintendo rilasciava quella serie di "scacciapensieri" conosciuti come Game & Watch. Giochi ed orologio.
Io adoravo Mario e Donkey Kong. Adoravo queste scatoline magiche che ci facevano sognare mondi, avventure, sfide...
Una scatola temporale che sarà sempre nei miei ricordi.
Questi miei ricordi si sono risvegliati grazie alla passione di tanti fan. Oggi ho analizzato due port appena usciti, Mario's Cement Factory convertito per C64 e l'omaggio a Cuphead.
La scatola temporale è stata aperta!!
Buon viaggio.

B) LA COLLABORAZIONE DI GIORGIO BALESTRIERI CON MADRIGAL (AKA LUCA ANTIGNANO)
Michele:
RMW: "Giorgio, benvenuto in questa intervista di RMW25, oggi gli onori vanno a te che hai collaborato con il geniale Luca Antignano per il porting dei G&W. Come è nata la vostra collaborazione?"
GB: "Ciao Michele e grazie per avermi dato la possibilità di provare l'esperienza di essere "dall'altra parte" in una intervista di RMW. La collaborazione con Luca è iniziata grosso modo nei primi mesi del 2017, quando mi venne la voglia di rimettermi a giocare con i G&W che avevano allietato le mie giornate da adolescente (ossia 30-35 anni fa). Conoscevo già da prima il folle progetto di Luca, ma non l'avevo mai provato seriamente prima di quei fatidici mesi quando la nostalgia prese il sopravvento. A quel tempo usavo Linux, nella forma di Ubuntu Mate, come principale SO già da qualche anno ma il pacchetto di emulatori MADrigal era distribuito solo per sistemi Windows. Dopo qualche prova, scoprii che potevo farli girare senza difficoltà anche su Linux utilizzando Wine e scrissi a Luca per dagli un feedback dei miei esperimenti. Chiacchierando con lui, scoprii che esisteva un core G&W per libretro, creato da Andre Leiradella e rilasciato come opensource, che potevo utilizzare per far girare i simulatori in maniera nativa sotto Linux. Da qui l'idea di tentare di renderli disponibili anche per RetroPie, la distribuzione principe per il retrogaming su sistemi SOC e non solo."

RMW: "In quali porting ti sei cimentato? Quale asset software e hardware hai adottato e soprattutto quali problemi o bug hai dovuto affrontare?"

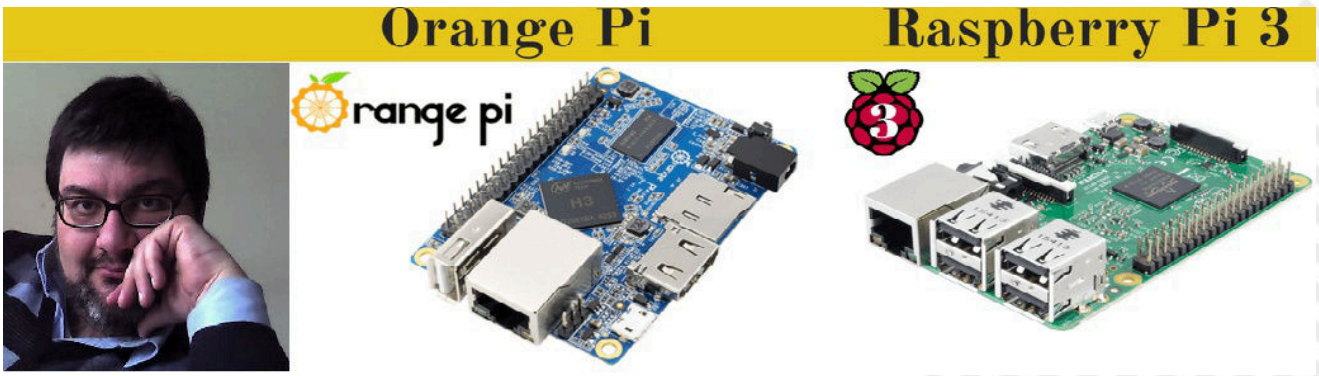


Figura 2

GB: "In realtà, non ho effettuato nessun porting, il lavoro vero di trasposizione l'aveva già fatto Andre, io mi sono solo limitato ad incorporarlo su RetroPie e, già che c'ero, su OrangePie (la versione di Retropie per SOC Orange). Andre aveva creato il core G&W da includere in libretro, il principale emulatore multisistema di RetroPie, ed un transpiler per convertire i sorgenti in Delphi dei giochi di Luca in LUA che, una volta compilati, potevano essere visti come "rom" da libretro e fatti girare grazie al core G&W. Come ho già detto, il mio contributo è stato semplicemente quello di impacchettare tutto in maniera che potesse essere installato e fruito tramite RetroPie."

RMW: "C'è stato qualche aspetto curioso o degno di nota che hai notato durante il tuo lavoro? Differenze minime oppure importanti tra le due schede Orange/Raspberry?" (cfr. figura 2)
GB: "No, in realtà ottenere una versione funzionante dei simulatori è stato un processo abbastanza lineare su entrambe le piattaforme. Sia RetroPie che OrangePie sono basate su Debian Linux, per cui una volta recuperati i sorgenti di libretro-gw da GitHub, compilarli è stato piuttosto semplice; il massimo della difficoltà è stato individuare le librerie necessarie al processo, ma assolutamente nulla di complicato, Andre ha fatto un ottimo lavoro. Tra l'altro, nelle versioni odierne di RetroPie la compilazione del core G&W non è più necessaria poiché di default questo core è già disponibile nella distribuzione ma all'epoca era inclusa solo una versione piuttosto vecchia e per far girare dignitosamente il pacchetto completo di Luca occorreva compilare a mano il tutto. L'idea dietro il mio pacchetto per Retro/OrangePie in effetti nasce proprio da qui, ossia dalla voglia di rendere disponibile il parco di G&W simulati da Luca velocemente e facilmente per tutti gli utenti di queste distribuzioni per il retrogaming. Per semplificare al massimo l'installazione, riuscii a creare un singolo file eseguibile di qualche decina di mega che bastava copiare sulla macchina di destinazione, lanciare ed attendere che finisse il suo lavoro. A quel punto ci si ritrovava con un core G&W aggiornato, tutti i simulatori installati e tutti gli snapshot già disponibili, non restava altro che giocare. Parlo al passato perché l'ultima volta che mi sono occupato di aggiornare il mio pacchetto di installazione è stato nel 2019; probabilmente dovrei dargli un'occhiata e vedere se occorre ritoccare qualcosa. Luca nel frattempo ha smesso di produrre simulatori ma libretro è andato avanti, forse c'è qualche rettifica da apportare."

RMW: "Hai qualche aneddoto o curiosità inerente al mondo dei G&W? Ne collezioni e se si, possiedi qualche rarità o stai cacciando qualche G&W in particolare (magari qualche lettore potrebbe aiutarti nella caccia)?"
GB: "Un aneddoto si, in cui sono coinvolti sia i G&W che me stesso. Ricordo che all'età di 16 o 17 anni ero in spiaggia con i miei genitori e mi ero portato un G&W per passare il tempo sotto l'ombrellone. Era uno di quelli a schermata multipla, ossia quel tipo di G&W che erano in grado di mostrare quadri di gioco diversi a seconda del livello, accendendo e spegnendo in maniera opportuna alcuni elementi grafici della schermata. Questo in particolare aveva un piccolo pirata come protagonista (cfr. figura 3) che in una prima fase di gioco doveva trasportare una serie di bombe dalla nave pirata ad una caverna

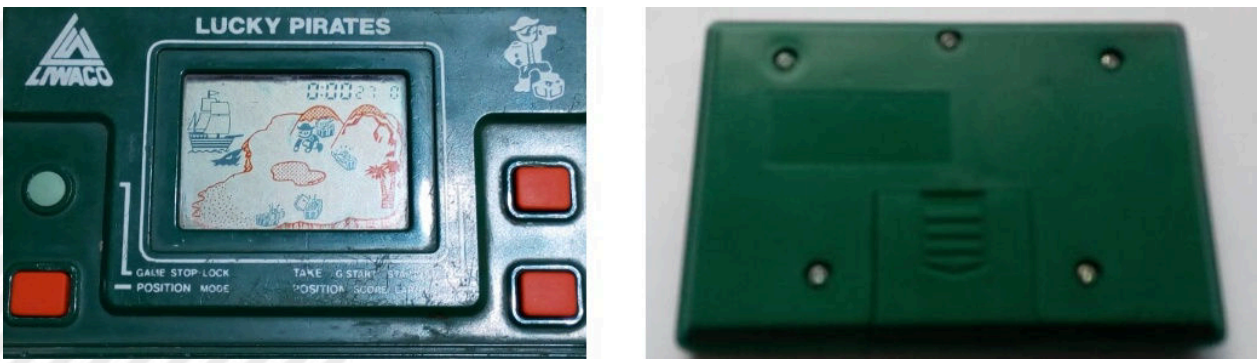


Figura 3

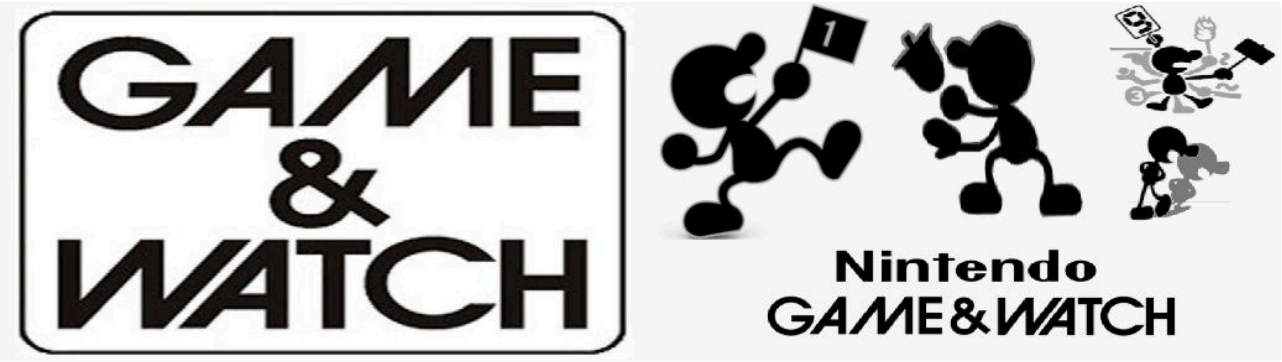


Figura 4

percorrendo lo schermo da sinistra verso destra e nella seconda caricare i tesori trafugati dalla caverna sulla nave viaggiando da destra a sinistra, mostrando una schermata diversa per ogni fase grazie al "trucco" menzionato prima.

Beh, il punto è che dopo un po' ero talmente concentrato sul gioco che letteralmente dimenticai dov'ero e quello che c'era intorno a me, tanto ero preso dall'azione.

Dopo un bel po' di tempo (credo un paio d'ore), riuscii a battere il gioco, accumulando tanti punti da azzerare il segnapunti e mi fermai.

Riguadagnata la cognizione della realtà, mi accorsi che praticamente tutti i vicini di ombrellone mi stavano osservando con espressioni tra il curioso e lo stupito: seguendo il ritmo di gioco, sempre più veloce, avevo assunto un'assurda posa sulla sdraio ed avevo preso a muovere mani e dita in maniera frenetica, a scatti velocissimi, cosa che attirò l'attenzione (e la preoccupazione) di quelli che mi stavano intorno. Oggi un comportamento del genere non desta meraviglia, visto il rapporto a volte patologico che si ha con gli smartphone, ma all'epoca rischiai seriamente che qualcuno mi tirasse una secchiata d'acqua per farmi "riprendere"...

Per quanto riguarda la collezione dei G&W, si ne ho qualcuno, sono un collezionista di console e le ho grossomodo tutte dalla Channel F alla PlayStation 4, ma non colleziono G&W, quelli che posseggo li ho presi solo per variare un po' il parco macchine. Sono orientato praticamente solo alle console sia perché alcune di loro le so programmare, sia perché è grazie ai videogiochi che ho scoperto cosa volevo fare nella vita.

La mia prima console è stata una Atari 2600 su cui ho passato, come molti della mia generazione, una quantità indegna di ore a giocare.

Poi, da buon pre-adolescente, mi venne voglia di capire com'erano fatti i videogiochi, il che mi portò qualche anno dopo a scoprire il meraviglioso mondo della programmazione, la cui arte è poi diventata il mio mestiere."

Bene. Ringraziamo il nostro Giorgio per il divertente aneddoto, la preziosa intervista e l'importante lavoro svolto, donato alla comunità dei G&W (cfr. figura 4 e 5). Adesso la tabella di marcia prosegue con altre curiose e golose tappe.

Figura 5

C) MARIO'S CEMENT FACTORY PORTING
Carlo N. Del Mar Pirazzini:
Mario'S Cement Factory (cfr. figura 6)
Editore: hayesmaker64
Anno: 2020
Genere: Game & Watch/Platform
Piattaforma: Commodore 64

Iconico. Difficile trovare una parola diversa per Mario e per i Game & Watch.
Questa conversione si basa sul gioco originale, pubblicato nel 1983 sviluppato per i Game & Watch di Nintendo sotto la guida di quel genio di Gunpei Yokoi.
Convertito per Gameboy, Gameboy Advance e anche per Nintendo DSi è considerato come uno dei giochi più strani del franchise di Mario.
Il giocatore controlla Mario al lavoro in una fabbrica di cemento, dove dovrà riempire delle betoniere con il cemento che cade dalla catena di montaggio.
Dovrà farlo velocemente e senza farle straripare sotto gli occhi del datore di lavoro. Per farlo dovrà attivare le leve su tre livelli che permetteranno la loro discesa nel camion.
Questo port per C64 è stato realizzato in Assembly e devo ammettere che mantiene lo stesso spirito frenetico del gioco per il portatile.
La curva di difficoltà è sempre graduale e si lascia giocare piacevolmente.
Veloce e ben strutturato, sono sicuro che regalerà una buona dose di divertimento a chi cerca un gioco veloce da caricare per rilassarsi.
Unica pecca è ovviamente la ripetitività che influisce sulla longevità globale, ma è da considerare un ottimo port.

Giudizio finale

Giocabilità: 80%
Incarna lo spirito dei Game & Watch e lo infila in un C64. Ben sviluppato.
Longevità: 65%
Ha lo spirito dai Game & Watch.. vi deve tenere incollati per pochi minuti. Altrimenti alla lunga, stanca.

D) GIG TIGER RETURNS
Michele:
Cari lettori, ci sono grandi novità. I Gig Tiger, i videogiochi portatili che hanno fatto la storia degli anni '90, stanno per tornare.

Ecco la notizia in anteprima rilasciata da "The Verge":
https://www.theverge.com/2020/2/19/21136607/hasbro-tiger-electronics-lcd-handheld-games-xmen-sonic-transformers

Hasbro ha annunciato che saranno disponibili negli Stati Uniti a partire dalla prossima estate 2020. Stop! Fermi tutti! Siamo già nel 2020 ed è autunno! Questa freschissima notizia infatti è datata 20 / Febbraio / 2020.
Ancora non si parlava di questo maledetto Covid19. Poi tutto è caduto nell'oblio.
I Tiger Electronics, videogiochi portatili diventati celebri negli anni '90 e conosciuti da noi con il nome di Gig Tiger, purtroppo non sono arrivati nell'estate 2020. L'iniziativa promossa da Hasbro aveva previsto quattro giochi: The Little Mermaid (La Sirenetta), Transformers Generation 2, X-Men Project X e Sonic

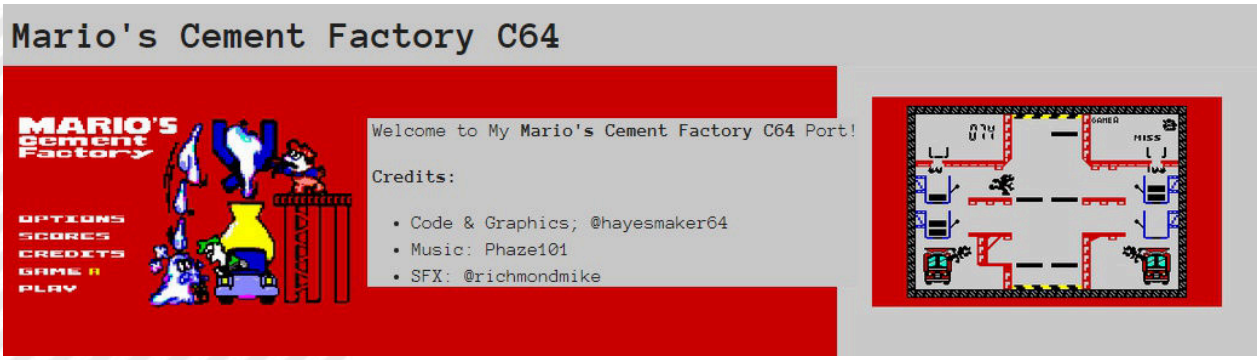


Figura 6

the Hedgehog 3. Erano prenotabili da GameStop negli Stati Uniti al prezzo di 14,99 dollari l'uno. Probabilmente in Europa il prezzo sarebbe stato 13,9 Euro. Al tempo i gloriosi GIG Tiger degli anni novanta costavano ventimila lire. Il prezzo del remake perciò sembrava ottimo.
Non erano state rilasciate informazioni riguardo l'espansione di vendita in Europa, le notizie ufficiali sarebbero state unicamente fornite alla fiera del giocattolo di New York. Come disse Morpheus in Matrix: "Al destino, come sappiamo, non manca il senso dell'ironia". Gli aggiornamenti di questi giochi finirono ben presto sopraffatti dalle notizie tragiche che tutti noi conosciamo.
In questo link le informazioni ufficiali ad oggi non aggiornate. (cfr. figura 7). Ad oggi reperibili. Ad oggi congelate:
https://nypost.com/2020/02/19/hasbro-bringing-back-beloved-1990s-toy-tiger-electronics-hand-held-games/

La Tiger Electronics aveva dato vita, tanti anni fa, anche al "Furby", la console "Game com", il robottino interattivo "2-XL", basato su audiocassetta, il "R-Zone" ed infine , appunto, questa linea di handheld LCD, simile ai famosi G&W della Nintendo. I Tiger erano delle console piccolissime con un display Lcd monocromatico ed una scocca robusta: sembravano indistruttibili. Ogni console era dotata di un solo videogame con annesso mini speaker per musica ed effetti. Ergonomicamente erano ben progettati, la solidità era indiscussa, l'audio era accettabile, il prezzo era competitivo. Nintendo aveva trovato un degno rivale in questo preciso settore. La presenza dei GIG Tiger, in quei tempi, avrà destato scompiglio nella casa della grande "N"? La mia opinione è: no. Nintendo era fiera di farsi scopiazzare, difatti nella casa della grande "N" aleggiava ovviamente una filosofia giapponese che recitava: il marketing richiama unicamente ulteriore marketing.
Le versioni dei Tiger del tempo erano numerose: Street Fighter, Double Dragon, Golden Axe, Castlevania, Robocop, Hook, Spider-Man, Batman, Flash, Biancaneve e i sette nani, La Sirenetta ed addirittura un gioco dedicato a Michael Jordan! Dai primi anni ottanta al 2003, anno della dismissione, sono stati creati ben 184 giochi, consultabili nel catalogo del sito Handheld Museum, vi allego il link:
https://www.handheldmuseum.com/Tiger/index.html

Tiger Electronics è stata fondata nel 1978 ed ha prodotto i suoi LCD in maniera indipendente fino al 1998, anno in cui è entrata nella grande famiglia Hasbro. Oggi, nel 2020, proprio Hasbro ha intuito la diffusa nostalgia per questi oggetti sacri (almeno per noi collezionisti). Ecco qua, finalmente nel 2020 nasce la Hasbro Tiger LCD. Purtroppo è nata in un momento storico molto complesso. Speriamo tutti che si troverà una soluzione efficace contro questa maledetta epidemia, così potranno ripartire tutte le produzioni mondiali. Non disperiamo, tutto è stato rallentato, i danni sono ingenti, le perdite ed il dolore sono immani, ma non tutto è perso. Nel frattempo, per ingannare il tempo, vi metto al corrente di una ghiotta curiosità: molti GIG Tiger sono perfettamente emulati grazie a Jason Scott che con il suo team ha reso disponibile "Handheld History", un archivio sul web che permette di emulare sul nostro browser circa 60 di quei titoli.
Il progetto ogni tanto viene aggiornato con nuovi giochi, molti dei quali sembrano essere perfetti come

Figura 7

quelli originali. Vi allego il link:
https://archive.org/details/handheldhistory

E) CUPHEAD PORTING
Carlo N. Del Mar Pirazzini:
Cuphead: Game and Watch Edition
Editore: indipendente
Sviluppatore: Simon Delavenne, At0mium, Heelio
Genere: Platform
Piattaforme: Web

Un tributo ad un gioiello che si rivela esso stesso una gemma preziosa. (cfr. figura 8)
Questa versione web di Cuphead si rivela una piacevole sorpresa per tutti gli estimatori dei G&W e del titolo sviluppato dalla MDHR.
Un gioco sviluppato con l'engine Unity WebGl, che ricrea sul vostro schermo il tipico G&W del passato ma vi mette alla guida del simpatico Cuphead alle prese con una terribile pianta carnivora che vi lancia pennuti da saltare e, nello stesso momento, cercherà di catturarvi con i suoi rami.
Ben realizzato, con pochi comandi (destra, sinistra, su più un tasto adibito per lo start del gioco) e soprattutto molto giocabile.
Che dire, forse un po monotono, ma di sicuro effetto.
Un omaggio ad uno dei migliori giochi run 'n gun degli ultimi 10 anni.

Giudizio finale

Giocabilità: 85%
Semplice e lineare. Ben sviluppato.
Longevità: 65%

Il livello di difficoltà cresce e purtroppo anche la voglia di provare qualcos'altro. Ma divertente.

F) ESISTONO ALTRI SIMULATORI ED EMULATORI?
Michele:
Nello scorso numero abbiamo parlato del geniale Luca Antignano (aka MADrigal).
Avete visitato tutti i suoi preziosi link e controllato tutto il materiale dei siti che ci ha fornito?
E' un lavoro straordinariamente voluminoso vero? Incredibilmente nel web c'è ancora materiale di emulazione e simulazione dei G&W pronto a soddisfare ulteriori nostre esigenze.

Mame probabilmente è la piattaforma più conosciuta, easy to use, alla portata di tutti. Forse è la scelta più comoda in assoluto. Merita un capitolo specifico. Quindi probabilmente ne parlerò in futuro, quando sarà creata una recensione dedicata, mettendo a confronto Mame insieme alle particolarità dei G&W come porting non ufficiali, cloni, rarità, etc..
In questo paragrafo parlerò unicamente della possibilità di giocare con un G&W/Handheld tramite piattaforme indipendenti da MAME.
Ovviamente non tutte le possibilità: questo fantastico mondo dei G&W è mutevole, alcuni progetti nascono, altri periscono, altri vengono fusi, è straordinario pensare che questi piccoli oggetti possano godere di una tale salute ed un ottima lungimiranza nel marketing! (cfr. figura 9)

Possiamo partire da Retropie/RetroArch abbondantemente presenti nel web.
Per esempio lr-gw. E' un simulatore e non un emulatore. Ciò significa che i giochi con cui è

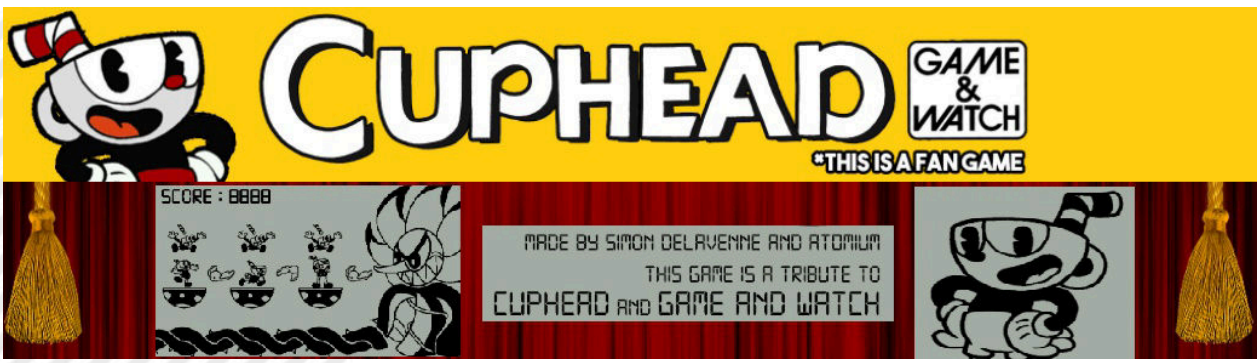


Figura 8

possibile giocare non sono in realtà i giochi originali, ma ricreazioni dei giochi.Ecco il link:
https://retropie.org.uk/docs/Game-%26-Watch/
Qui potrete trovare l'emulatore:
https://github.com/libretro/gw-libretro

Qui troverete le spiegazioni per l'emulazione:
https://youtu.be/DzbsfCC77IQ

Vogliamo giocare con i G&W tramite Java? Nulla di più immediato. Troviamo a disposizione LCDgame.js, è una libreria JavaScript open source che attualmente supporta rappresentazioni autentiche di Donkey Kong II e Mario Bros.
Ecco a voi il link:
http://bdrgames.nl/lcdgames/

Essendo tutto opensource potrete facilmente collaborare nel bellissimo nonchè snello progetto:
https://github.com/BdR76/lcdgame.js

Passiamo ad un altro progetto molto interessante: HQ. Ecco il link:
http://www.emulator3000.org/hq.htm

Handheld Quake, simulatore di giochi portatili sovietici e stranieri.
Sinceramente trovo veramente grazioso questo progetto poiché utilizzabile sia con Linux che con Windows. La lista dei giochi russi è questa: Mysteries of the ocean, Just you wait!, Merry cook, Explorers of space, Autoslalom, Merry arithmetics, Space flight, Fisher tom-cat, Hockey, Fowling, Space bridge, Rhythm.
La lista dei giochi Nintendo simulata tramite questa piattaforma è la seguente: Chef, Egg, Octopus, Fire, Mickey Mouse.
Provatelo. Vi piacerà. Un applauso agli sviluppatori. Complimenti vivissimi!
Non dimentichiamo di parlare di "Pica Pic", di Hippopotamus.
Sono G&W giocabili unicamente online, sfruttando Adobe Flash.
I creatori sono polacchi ed hanno dato vita ad un sito veramente divertente e immediato per un gameplay veloce! Ecco a voi il link:
http://www.pica-pic.com/

Il loro vecchio sito era questo:
http://www.hipopotamstudio.pl/

Vogliamo parlare di Android?
Su Google play ogni tanto si aggiungono G&W/ Handheld completamente nuovi. Basta digitare "Alice game & watch" e salta fuori uno dei G&W che più adoro! Il creatore "datsuryoku_k" di Kanagawa, Giappone, ne ha sviluppati tanti ed alcuni sono veramente divertenti. A voi il piacere della navigazione. Alcuni sviluppatori sono stati bravissimi, altri decisamente discutibili.
Altri creatori da citare? Sicuramente: AviSoft, Pixelegend, Lapigames, Mascot1039, Yanmania, Million Rabbit Studio, Short2Games, etc..
Più navigherete nei link dei suggerimenti e maggiormente vi allontanerete dal concetto purista dei G&W... decidete voi quando fermare il vostro viaggio verso... beh... i meandri oscuri del pessimo gusto!

Vogliamo parlare di Zophar? Ecco a voi il link:
https://www.zophar.net/gw.html

Figura 9

Troverete numerosi giochi pronti al download. Funzionano con Windows. La lista è golosa: Armor attack, Ball, Balloon fight, Banana sbang, Camelot, Circus circus, Donkey angler, Donkey kong, Donkey kong circus, Donkey kong 2, Donkey kong jr, Dungeons and dragons.

Infine, attenzione ad alcuni link non funzionanti e progetti abbandonati. Per esempio:
http://www.handheld.remakes.org/

Può bastare? Sono abbastanza sicuro che non ho elencato tutte le possibilità di gameplay di questi incredibili G&W. Sono altrettanto sicuro che nella prossima recensione ci sarà altro materiale che scoverò. Aiutatemi nella caccia, cari lettori!

G) NINTENDO G&W, SFIDA ALL'IMMORTALITA'
Michele:
Proprio mentre stavo stendendo questo articolo, ecco comparire, nel pomeriggio del 3 Settembre 2020, questo annuncio dalla Nintendo: "La classica console che ha rivoluzionato il mondo dei videogiochi fa il suo ritorno! In occasione del 35° anniversario di Super Mario Bros., non farti sfuggire Game & Watch: Super Mario Bros".
Mi sembrava incredibile, stavo scrivendo un articolo intitolato "G&W, sfida all'immortalità" e Nintendo svelava la notizia di un remake ufficiale dei G&W, nonchè altri giochi, per la ricorrenza dei 35 anni dalla nascita di Super Mario Bros. Queste si che sono reali proiezioni verso il futuro!
Ecco alcuni link:
https://www.nintendo.it/Notizie/2020/settembre/Guarda-subito-il-Super-Mario-Bros-35th-Anniversary-Direct--1836547.html
https://youtu.be/a8DJpeCy8CQ

La pubblicità recita : "Supera voragini, calpesta i Goomba ed entra nei tubi proprio come ti ricordi, ma con comandi ancora più precisi grazie alla pulsantiera + della console. Gioca da solo o alternati con un amico per un po' di sana competizione!".
Poi ancora : "Cerchi una sfida maggiore? Allora non perderti Super Mario Bros.: The Lost Levels! Se invece preferisci un'esperienza più rilassata, Ball (con un tocco speciale di Super Mario) fa al caso tuo!".
Infine conclude : "L'orologio digitale incluso può riprodurre 35 animazioni diverse, che includono anche alcuni degli amici e dei nemici di Mario! Tieni d'occhio la console anche quando non sei impegnato a salvare la Principessa Peach."

Esatto, è proprio così, Nintendo festeggerà il proprio brand più famoso: Super Mario Bros. (cfr. figura 10)
Il 13 Novembre uscirà un G&W ufficiale, esclusivo, limitato, dedicato a Mario. Insieme al gioco, nel G&W saranno presenti alcuni livelli "The lost levels" che ricordiamo, in Giappone sono usciti con il titolo "Super Mario Bros 2" ed infine sarà presente la versione classica di Ball con il viso di Mario al posto di "Mr. G&W".
Troveremo il nostro buffo Mario-giocoliere pronto ad intercettare le palline che cadono dall'alto. Sarà presente anche l'orologio. Un orologio tutto speciale a quanto pare.
Una nota cara a tutti noi collezionisti. Avete presente i pad del FamiCom (il NES giapponese per intederci)? Bene, questo G&W avrà forma e colorazione di quei pad! Sembra che saranno presenti anche diverse

Figura 10

personalizzazioni software, setup grafici e soprattutto degli adorabili easter eggs, tutti da scoprire come al tempo abbiamo scovato tutti i tubi sotterranei del nostro Mario.
Nintendo ha ufficializzato l'uscita per il 13 Novembre. Il prezzo sembra essere di circa 50€. Sicuramente il preorder diverrà un luogo caldo di battaglia, quindi, occhi aperti cari lettori. Oltretutto sembra che lo stile FamiCom sarà quello ufficialmente venduto in Asia, però ricordiamoci che alcune versioni dei Nintendo Mini passati, possedevano scocche e giochi differenti tra loro in base al luogo di vendita: Asia piuttosto che America piuttosto che Europa. Sono proprio curioso di conoscere la mole di vendita di questo remake. (cfr. figura 11)
Restiamo sintonizzati su queste notizie e teniamo bene a mente le regole del gioco attorno alle quali, tra pochissimo tempo, ruoteranno milioni di dollari:

Sony: Playstation 5,
Microsoft: Xbox Series X,
Nintendo: Game&Watch.

C'è poco da ridere. In casa della grande "N" non si fa nulla a caso. Ad ogni modo, solo il futuro potrà raccontarci il passato e soprattutto l'esito della sfida dei G&W... all'immortalità!

H) CONCLUSIONI

Michele:

Abbiamo parlato tantissimo. Il mondo G&W è talmente vasto che sicuramente, a breve, arriveranno ulteriori golose notizie. Ne sono certo. Oltretutto, per noi collezionisti, viste le recenti news, è un periodo concitato.
Invece, scommetto che per Nintendo è un momento di relax: si gode i propri introiti prima di annunciare nuove console ultra potenti, pronte a sfidare Sony e Microsoft con elegante leggerezza. I G&W sfidano l'immortalità? Non c'è dubbio.
Vinceranno anche questa sfida? Chissà!

Carlo N. Del Mar Pirazzini:

Rispolverate questi oggetti, installate o provate i giochi che abbiamo recensito qui. E' un tuffo nel passato e un salto nei ricordi bellissimo.. e soprattutto una sfida contro voi stessi!! Siete ancora in grado di battere il vostro io di 40 anni fa??

Figura 11

SENSIBLE WORLD OF SOCCER 2020

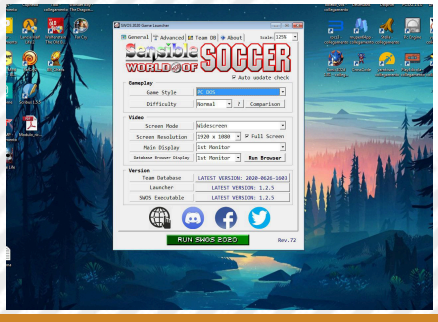
Anno: 2020
Editore: FREEWARE – SENSIBLESOCCER.DE
Sviluppatore: Sensible Software (originale) – Diversi sviluppatori nel corso degli anni
Genere: Sportivo
Piattaforma: PC/AMIGA

Il gioco più bello del mondo? Il calcio? No, Sensible Soccer!

Fifa 2020 e PES sono solo successori di quello che è il re assoluto dei giochi di calcio su computer e console.
Uscito originariamente su Amiga e successivamente su ATARI ST e PC DOS, Sensible Soccer cambiò il modo di giocare a calcio sulle nostre piattaforme ludiche.

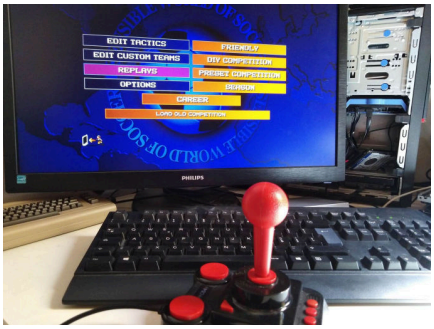
È forse il gioco più convertito ed imitato del genere e, nonostante abbia sul groppone un bel po' di anni ha una fan base davvero impressionante. Così impressionate da renderlo immortale. Negli anni è stato continuamente aggiornato e rifinito, giocato e rigiocato da tutti gli appassionati.

In questo sciagurato 2020 i ragazzi di sensiblesoccer.de escono con questa versione "definitiva".
Sensible World of Soccer 2020 è un'applicazione per Windows realizzata disassemblando la versione Dos del



gioco Sensible World of Soccer 96/97. Ingrandita e ampliata, aggiornata con tutti i giocatori del mercato attuale. Immenso è il data base dei giocatori e delle nazioni selezionabili (2400 team e 26.000 giocatori).

Cosa hanno aggiunto in più. Un supporto ai più moderni controller usb, nuovo sistema di salvataggio, possibilità di giocare in multiplayer e persino un sistema di aggiornamento dei db squadre e giocatori gratuito.
Il gioco non è cambiato. È sempre quel bellissimo prodotto che uscì nel 94. Veloce, dinamico e tutto basato sul controllo della palla "dinamico" del giocatore.



Giocabilissimo e dalla longevità eterna, soprattutto in questa versione sempre aggiornata e aggiornabile.
Dulcis in fundo, la versione è disponibile anche per Amiga (A600 o A1200 con 4 MB DI FASTRAM, esoso ma ci può stare).

Cercatelo e scaricatelo (lo trovate in offerta su GOG a poco meno di 2 euro). Il calcio non è mai stato così divertente.

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 95%

Controlli perfetti e livello graduale di difficoltà. Ottima IA delle squadre. Imponenti gli aggiornamenti e la possibilità di rimanere sempre aggiornati nel futuro.

» Longevità ETERNO

Non posso utilizzare numeri. Sensible Soccer è andato oltre il tempo e la storia e possiede una fanbase davvero incredibile. Sempre aggiornato, sempre longevo… quindi ETERNO.

RUFF 'N' TUMBLE

Editore: Renegade
Sviluppatore: Wunderkind
Anno: 1994
Genere: Platform/Shoot'em up
Piattaforma: Amiga/CD32

La perfezione non è essere perfetti, ma tendere continuamente ad essa.
- JOHANN GOTTLIEB FICHTE

Signori miei, qui ci siamo vicini. Vicinissimi alla perfezione in un videogioco e il tutto dentro ad un disco da 1,44 Mb (quando ancora il mondo era semplice vero?, NdN).

Nel 1994 Ruff'n'Tumble sul mercato Amiga, un mercato sconvolto dal fallimento della Commodore, imponendosi come uno dei migliori titoli di quel periodo e forse di tutti i tempi.
Al giorno d'oggi il gioco non solo conferma le qualità che dimostrava allora, ma ne sfodara moltissime che in quel periodo passarono inosservate, ma che al giorno d'oggi valgono di essere rispolverate.

Il gioco è sviluppato su quattro mondi (divisi in quattro sezioni) ed è un gioco di piattaforme con una forte vena da shoot'em up, oggi si potrebbe classificare tra un Run n Gun e un Metroidvania. Sviluppata dal duo conosciuto come Wunderkind (Robin Levy e Jason Perkins) ci attira subito per la sua realizzazione tecnica e ci trasporta in un gioco dalle meccaniche furiose ma al contempo complesse e variegate.
Il protagonista è un bambino paffuto di nome Ruff Rogers, che per terminare i livelli, deve raccogliere un numero (indicato da appositi contatori in alto sullo schermo) di biglie rosse, verdi e blue. Senza di queste non può accedere al livello successivo. E, nel mentre, far saltare in aria tutto quello che gli si mette davanti!
Il momento di "distruzione" è possibile grazie a cinque differenti tipologie di armi che vanno dalla semplice mitragliatrice al più potente laser, passando per il lanciarazzi o il devastante lanciafiamme. Una delle innovazioni più grandi del gioco sta appunto nella gestione dell'arsenale: l'armamento ha munizioni infinite ma non dura in eterno. Un'apposita barra decresce in maniera rapida col passare dei secondi indipendente dai colpi esplosi. Una volta giunta alla fine si ritorna al dignitoso ma non "godurioso" sparo classico. Per evitare la situazione l'unica cosa che il nostro ciccioso

eroe può fare è raccogliere i numerosi power up presenti nei livelli. Può sembrare una limitazione ma in realtà rende l'azione frenetica e la curva di divertimento sempre al massimo, permettendovi di esplorare il gioco in ogni anfratto possibile.
Notevole è anche la conformazione dei livelli, la disposizione e il comportamento dei nemici. Studiato apposta per costringere il giocatore ad affrontare il percorso in velocità, ma permettendogli anche di esplorare le mappe (che sono davvero vaste) e di fronteggiare insidie con un minimo di tattica.
L'intensità di azione diviene quindi mozziafiato, dispensando un divertimento in quantità industriale!
E la curva di difficoltà? Ruff'n'Tumble è severo ma giusto. Mai scorretto e permette sempre di proseguire offrendo, grazie ad un ottimo sistema di controllo, varie modalità per farlo.
Ed è il sistema di controllo il vero must sul quale i Wunderkind hanno costruito questo gioiello. Pur con un solo tasto e nonostante l'inerzia eccessiva, la cura con il quale è stato studiato ci permette un controllo pressoche' totale su Ruff, che corre, salta, nuota, si arrampica e blasta il tutto con precisione e naturalezza; cosa rarissima in moltissimi altri prodotti simili di quei tempi (e anche in molti giochi moderni). Il risultato? Una giocabilità eccellente, esaltata anche da una colonna sonora di impatto!
Concludo parlando della realizzazione grafica. Pulito, pieno di colori ed effetti luce (notevoli quelli di passaggio tra luce e buio), animazioni sempre perfette e una cura del dettaglio impressionanti. Questo era possibile quando i programmatori venivano lasciati lavorare in pace e con passione e questo era quello che sapevano tirar fuori dalla Amiga.
Ruff'n'Tumble, signori miei, è il canto del cigno del Commodore Amiga. Un canto magnifico che si avvicina alla perfezione e dimostra come quella macchina fosse capace di andare oltre alla cattiva gestione dei suoi manager. Capolavoro!

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 95%

Livelli ben studiati e immensi, ottimo sistema di controllo, severo ma giusto e immediatamente giocabile.

» Longevità 85%

Un ottimo prodotto che vi terrà impegnati per moltissimo tempo e che dimostra che era possibile in un dischetto da 1,44. Non potete perdervelo ora.

GOLDEN AXE WARRIOR

Anno: 1991
Editore: Sega
Sviluppatore: Sega
Genere: Action Rpg
Piattaforma: Sega Master System

Tra la fine degli anni 80 e l'inizio dei 90, la SEGA sentiva una forte pressione davanti allo strapotere in campo ludico casalingo di Nintendo. I marchi più famosi della casa di Kyoto erano Super Mario e Legend of Zelda.
Se per quanto riguarada Mario bisognerà attendere ancora un po (il 1991 con l'arrivo di Sonic), Sega decide di sviluppare un concorrente del noto Rpg Zelda sulla piattaforma ad 8 Bit chiamata Sega Master system. Tra il 1989 e il 1990, forte del brand di Golde Axe e della sua leva dei suoi appassionati crea questo Golde Axe Warrior.
In quegli anni passo un po in sordina, bistrattato dalle riviste del settore o quasi, ma devo ammettere che si tratta di un buonissimo prodotto.
Se si ha familiarità con Zelda, Golden Axe Warrior sarà come sentirsi a casa. La trama è classica, giocheremo nei panni dell'eroe, venuto per distruggere Death Adder (nemico storico della saga Golden Axe) e per porre fine al suo regno di terrore nel mondo di Firewood. Insomma, la trama è simile a Zelda e agli altri milioni di giochi d'avventura.
Per arrivare dal cattivo di turno dobbiamo raccogliere nove cristalli nascosti in nove dungeon differenti sparsi in tutta Firewood. Un ulteriore vantaggio per proseguire nel gioco è la pletora di segreti nascosti, mini giochi, citt da esplorare, side quest da risolvere e negozi. Ci sono tantissime zone da esplorare e da individuare e, letteralementem, un segreto in ogni mondo che andremo a visitare. In alcuni punti saremo in grado di esplorare zone utilizzando oggetti come canoe, mini imbarcazioni o altro che ci permetteranno di raggiungere zone difficilmente accessibili senza. Questo mancava al primo Legend of Zelda, un mondo enorme da esplorare liberamente. Ogni mondo è ben caratterizzato e specifico e inoltre c'è una bella sezione corposa dedicata alle magie. Come in Zelda per Nes anche qui potremo salvare i progressi di gioco.
Insomma è un bel gioco, classico ma avvincente.
Graficamente era anni luce rispetto al rivale su Nintendo. Molto curata la grafica e ben dettagliata. Con un ottimo uso delle palette del 8 bit di Sega. Molto fluido e ben animato. Bella anche l'introduzione d'apertura piuttosto epica. Insomma un bello sfoggio delle capacità grafiche del Sega Master System.
Molto scarso il sonoro, la musica non è negativa di per se ma alla lunga è snervante e ripetitiva e mi è parsa inadatta in alcuni punti di gioco. Belle le musiche dei Boss e ottimi gli effetti sonori in generale.

Grafica

Ecco una delle aree in cui il gioco brilla davvero rispetto a Zelda, ha una grafica semplicemente meravigliosa. Alcuni di essi non sono molto innovativi (i boss di scambio di palette mi infastidiscono sempre e alcuni

potrebbero aver usato qualche dettaglio in più), ma sono comunque meravigliosi. Tanti colori e movimento fluido, è ben fatto. La storia di apertura è decente, ma la schermata del titolo è una vera e propria vetrina di ciò che l'SMS può fare.

Suono e musica

Qui troviamo il gioco un po 'carente. Una delle cose più memorabili di Zelda era la musica, che si adattava perfettamente. Sebbene la musica in Golden Axe Warrior non sia necessariamente negativa, diventa semplicemente un po 'fastidiosa e ripetitiva e talvolta non si adatta davvero. La musica dei dungeons avrebbe potuto essere più inquietante, anche se la musica dei boss è bella. Gli effetti sonori, tuttavia, sono buoni, molto meglio di quello che troverai in Zelda.

I controlli all'inizio sono un po 'traballanti, il personaggio ha la tendenza a spostarsi un po' nella direzione sbagliata fino a quando non ti abitui, ma non è un grosso problema. Alcuni nemici richiedo una precisione maggiore e putroppo la collissione non è delle migliori e va ad influenzare il gameplay nel suo complesso. Tuttavia la configurazione molto semplice e l'interfaccia con elementi diversi non sono male. Potrebbe essere stata una buona idea avere due slot separati, uno per un'arma e uno per uno strumento / magia, ma dal momento che non hai mai davvero bisogno di usarne due contemporaneamente, non cambia più di tanto.

Il grado di sfida del gioco ci terrà impegnati parecchie ore e non ci annoierà.

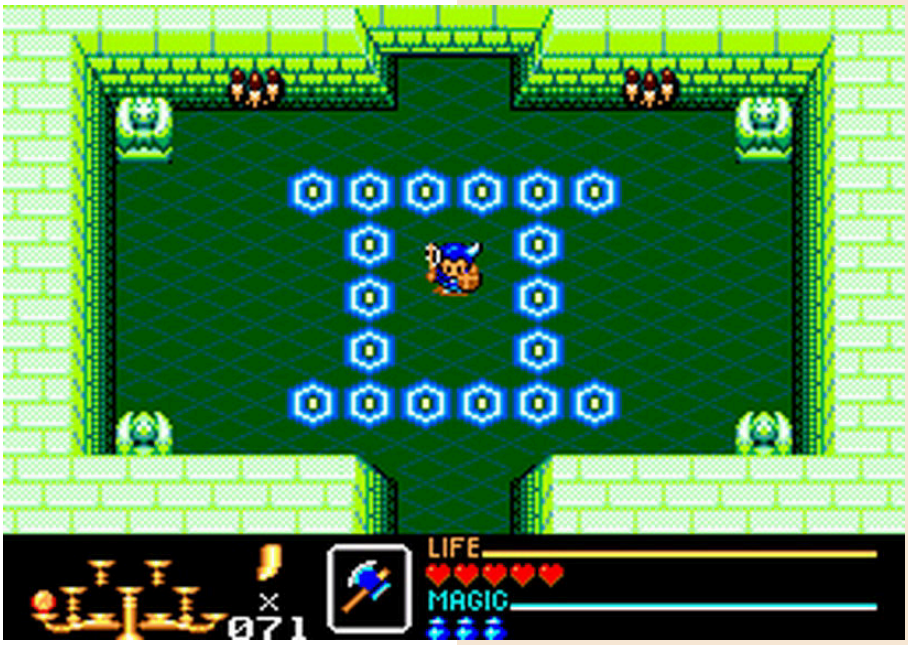
Devo ammettere che è un buon gioco! Molto buono! Uno dei pochi ARpg per SMS che mi è piaciuto giocare fino in fondo. Sebbene sia molto simile alla saga di Zelda è davvero ben fatto tecnicamente e presenta un buon grado di sfida. Non migliora il genere ma si lascia giocare con piacere. Peccato che Sega non abbia continuato a sviluppare questa serie di Spin off anche su altre console perché il fascino del mondo di Golden Axe è sempre notevole. Provatelo e giocateci.
La cartuccia la trovate a prezzi "scandalosi" e il mio consiglio è di provare a fare ricerche o caricare la rom su un everdrive o su emulatore. Non esiste la localizzazione in italiano, ma non influisce più di tanto.

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 70%
Comandi non sempre perfetti non aiutano a prendere la manualità immediata al gioco. Peccato perché il sistema di gioco e il doppio slot per armi e magia non sono per niente male.

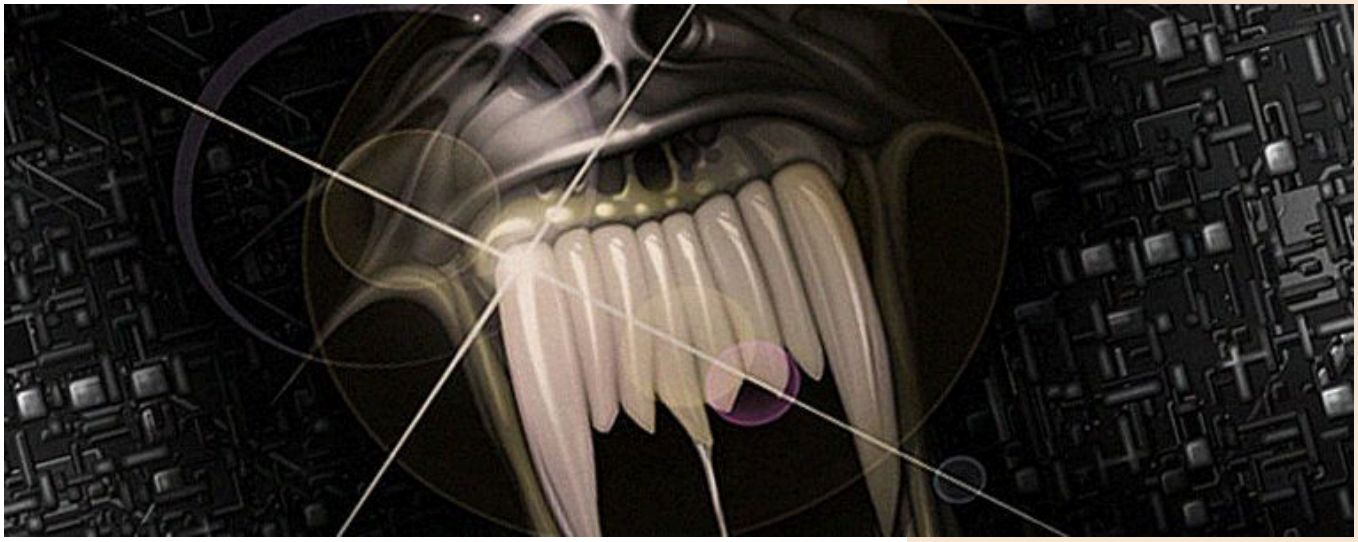
» Longevità 90%
Tante sidequest, tanti oggetti da recuperare e segreti da svelare il tutto corredato da una meravigliosa grafica vi terranno impegnati per un bel po.



ALIEN BREED

Piattaforma: Commodore Amiga
Genere: Azione/Sparatutto

Data di pubblicazione: 1991
Versione recensita: Alien Breed Special Edition 92 – Amiga ECS



Essere teen negli anni '80 non significava solo scoprire giorno dopo giorno computer e coin-op sempre più innovativi ma anche assistere alla nascita di veri e propri capisaldi e capolavori del cinema e della musica.

E magari mentre copiavamo da MC Microcomputer l'ultimo listato per il nostro Vic20 o Commodore 64 e mentre alla radio ascoltavamo You Shook me All night long degli AC/DC su Italia 1 stavano per dare uno degli ultimi film di fantascienza usciti nei due/tre anni precedenti.
Ovviamente se volevamo viaggiare nel tempo guardavamo Ritorno al Futuro ma quando avevamo bisogno di un po' di sano terrore e adrenalina le scelte non mancavano davvero.

La saga di Alien, in tempi recenti evolutasi in Prometheus e Covenant riuscì a portare sui nostri schermi il terrore più puro proveniente dallo spazio sotto forma di creature imponenti, infide e letali disegnate dal genio di Giger, animate dal nostro Rambaldi con una nemesi, la giovane Sigourney Weaver, determinata e sexy allo stesso tempo, in grado di capire prima del resto dell'equipaggio che quelle bestie trovate su di una remota colonia stellare terrestre non sarebbero mai dovute arrivare sulla terra.

E mentre le tecnologie sul finire degli anni '80 passano dagli 8 ai 16 bit la voglia di rivivere quell'epopea cinematografica sotto forma di videogioco diventava sempre più forte ma solo con l'avvento dei mai dimenticato computer Amiga di casa Commodore un team di programmazione inglese riuscì a regalare al popolo dei videogiocatori un gioco in grado di ricreare la tensione vista nelle pellicole cinematografiche.

Team 17 per Amiga era sinonimo di qualità e (spesso) anche di difficoltà ma nel caso di Alien Breed era chiaro che non avrebbe potuto essere una passeggiata fuggire da una base spaziale infestata da Xenomorphi senza perdere una tonnellata di vite.

Il gioco non aveva la licenza originale ma quello che c'era bastava a ricreare sui nostri monitor l'adrenalina della caccia a fottuti insetti giganti con acido al posto del sangue.

Le missioni erano piuttosto semplici: raggiungi l'uscita, raggiungi l'uscita entro un tot numero di secondi dopo aver distrutto alcuni reattori ma la struttura labirintica dei livelli lo era notevolmente meno.

I corridoi erano spesso bloccati da porte ad attivazione elettronica, servivano chiavi per aprirle e sparse nei livelli non ce n'erano mai abbastanza. Era quindi necessario accedere ad alcuni terminali situati in specifiche posizioni così da accedere al computer centrale della base ovvero il mai dimenticato Intex System dove utilizzando i crediti recuperati si potevano acquistare chiavi, potenziamenti, nuove armi, munizioni e medikit indispensabili per salvarci la pelle dai continui ed incessanti attacchi alieni.

La serie Alien Breed ricevette numerosi seguiti: Alien Breed 2 The horror continues e Alien Breed Tower Assault dove la grafica bidimensionale del gioco fu portata all'ennesima potenza con l'aiuto del chipset AGA montato sugli ultimi modelli di Amiga (1200 e 4000) più ben due capitoli in 3D che cercavano di spingere l'architettura Amiga oltre i suoi limiti (richiedendo però schede di espansione aggiuntive per potersi godere il tutto a schermo intero con un framerate decente).

In tempi più recenti l'ancora attivo Team 17 che ricordo fu l'autore di altri titoli memorabili come la saga di Worms su Amiga,PC e Console oltre che lo sparatutto Project X, il platform Superfrog e i picchiaduro Body Blows cercò di riesumare dal passato il brand con nuovi episodi in 3D su PC e Console senza però risultare incisiva come per i primi capitoli 2D su Amiga.

Degli Alien Breed originali non mancano anche dei remake pubblicati su Playstation 4 e PS Vita anche se comunque, personalmente, consiglio di rispolverare la saga originale in emulazione Amiga su PC con WinUae per capire davvero cosa si provava allora, agli inizi degli anni '90, quando i fottuti alieni uscivano dalle pareti delle nostre camere da letto!

di **Flavio Soldani**

GIUDIZIO FINALE

» Giocabilità 92%

Finalmente un gioco dove sopravvivere ad orde di alieni in grado di ricreare la tensione del blockbuster cinematografico Il primo capitolo di una serie non ufficiale in grado di regalare al giocatore giocabilità e soddisfazioni con un pizzico di strategia che non guastava.

» Longevità 96%

Il gioco non era lunghissimo ma vista la difficoltà arrivare alla fine era comunque un'impresa non da poco. Consigliamo la versione Special Edition 92 che rendeva il gioco leggermente più facile. Se per longevità intendiamo anche la durata nel tempo direi che anche dopo 30 anni una sana partita ad Alien Breed in versione emulata è soddisfacente ancora oggi!

FRANTIC FREDDIE

Anno: 1983
Sviluppatore: Commercial Data Systems

Piattaforma: Commodore 64
Genere: Platform

Dopo un lungo e afoso Agosto passato interamente a Milano, sudando sotto la mascherina e alla ricerca di qualche titolo passato un po' inosservato ma carino, ho fatto l'ennesima scoperta grazie soprattutto alle cassette da edicola. Spesso e volentieri comprendevano piccoli titoli quasi introvabili in versione originale, ma che meritavano qualche riflettore vista la giocabilità e l'intrattenimento che ci davano durante i noiosi pomeriggi estivi e festivi ed anche durante le serate interamente dedicate alla nostra postazione videoludica.

Questa volta è il turno di un tecnico dei telefoni, Frantic Freddie! Gioco che si presenta come il classico a piattaforme bidimensionali su più livelli contraddistinti da scale, in questo caso pali del telefono su cui salire e scendere per evitare nemici e raccogliere gli oggetti essenziali per terminare il livello e tanti altri bonus che incrementano il punteggio.

Detto così potrebbe essere un gioco semplice di quelli che intrattengono poche decine di minuti e invece la difficoltà non manca di certo! Essa era dettata dal fatto che le scale non potevano essere salite o scese da entrambi i lati, in più potevano bloccare il passaggio per andare oltre, mettendoci in un vicolo cieco ad aspettare solo la morte.

Riassumendo, per completare ciascun livello bisogna recuperare tutte le pentole luccicanti sparse per il livello ed evitare i mostri chiamati Greeblies che ad ogni livello cambiano aspetto (ma la cattiveria rimane sempre uguale). Ogni tot di livelli superati vedremo anche dei simpatici stacchetti, molto tipici dei platoform anni 80, che danno quel pizzico necessario in più di intrattenimento e ricompensano la nostra non poca fatica.

Il livello musicale è molto buono grazie a musiche di brani di successo e popolari come Boogie Fever, Don't Bring me down ecc. Forse rischio di essere ripetitivo, ma il biscottone aveva la più bella musica secondo me e mi meraviglio di come possano avere ottenuto tutti quei brani di successo e convertirli in otto bit!

Per chi è amante ed esperto del genere platform, troverà sicuramente un po' di originalità in questo gioco o forse era meglio dire limitazioni?

Come ogni gioco che si rispetti, una volta presa la mano sarà divertimento puro! I giochi quasi impossibili da finire si contavano sulle dita di una mano e adesso non sto qui ad elencare i titoli visto che li avrete sicuramente notati e giocati all'infinito con una speranza che vive tuttora.

La mia speranza invece è che tutti voi abbiato passato bene queste vacanze un po' particolari e diverse dal solito; ma non diversa è stata la nostra attenzione e passione verso il retrogaming che ci ha tenuto compagnia non solo durante i mesi di lockdwon, bensì anche sotto l'ombrellone, chalet di montagna ecc.

Prima di salutarvi vi raccomando di pronunciare questa frase ogni volta che giocherete a questo e tutti gli altri giochi: Nulla è impossibile!

di **Daniele Brahimi**

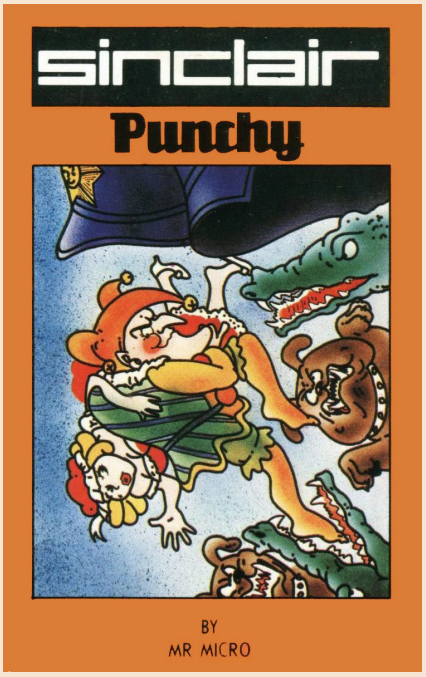
GIUDIZIO FINALE

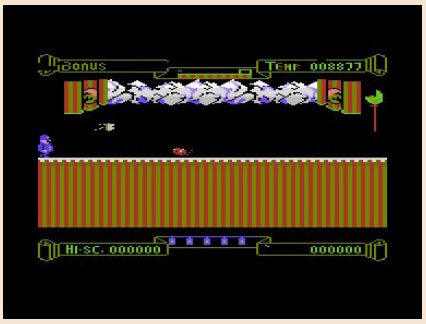
» Giocabilità 70%
Diifcoltoso al punto giusto, ma divertente.

» Longevità 75%
Non lungo, ma fonte di compagnia!

PUNCHY

Anno: 1983
Sviluppatore: Mr.Micro
Piattaforma: Zx Spectrum, Commodore 64, Commodore 16, Commodore VIC-20, Amstrad CPC, MSX, Tatung Einstein
Genere: Platform





Prendi il controllo di Bobby il poliziotto in un paesaggio da incubo.

Mentre le nuvole rotolano minacciose sopra la tua testa, il tuo obiettivo è attraversare quindici schermi per raggiungere lo stand di Punch e Judy, dove Judy è stata rinchiusa dal cattivo di turno, Mr. Punch.

Ovviamente, Mr. Punch ha posto vari ostacoli sulla strada di Bobby: ci sono buche da scavalcare con una precisione "pixel perfect", pomodori volanti e torte di crema pasticcera, tappeti magici da cavalcare e, naturalmente, lo stesso Mr. Punch, che è più che pronto ad infilzarti con la sua spada per impedirti di andare avanti nel tuo percorso di gioco e completarlo.

Ho giocato sia la versione su C64 che quella su C16 e, nonostante il C64 abbia dalla sua parte una dotazione di RAM superiore ed un comparto audio/video migliore rispetto al suo fratello minore, ritengo che su quest'ultimo la resa sia migliore, il gioco risulti più grazioso e giocabile.

Nel C64 la scelta dei colori non mi è sembrata azzeccata e si perde qualcosa anche in termini di giocabilità.

Nel C16 invcece la grafica è gradevole, i colori non sono eccessivi ed il personaggio di Bobby è ben disegnato, anche se si nota a volte durante il gioco qualche difetto cromatico noto come "colour-clash", tipico anche dei giochi Spectrum.

Il suono è funzionale e c'è una simpatica melodia suonata come ricompensa per aver raggiunto la fine di ogni schermata.

I controlli sono semplici e rispondono in modo fluido e la difficoltà, proseguendo nei vari schermi di gioco, cresce nel modo giusto.
Ovviamente la difficoltà è tanto maggiore quanto più ci si addentra nel gioco.

A tratti il gioco può risultare frustrante, in particolare quando i salti devono essere cronometrati con estrema precisione per evitare i detriti volanti, ed esattamente nel punto corretto sullo schermo.

C'è anche un limite di tempo per completare ogni schermata, ma il gioco è comunque molto giocabile, e con un po' di pratica può essere completato, anche se Bobby non può mai riposare - proprio mentre riesce a salvare Judy, Mr. Punch arriva e la spinge via di nuovo, e il gioco torna all'inizio!

Con i suoi pregi e difetti, lo considero comunque un buon gioco, ed è tra l'altro uno dei primi che ho giocato nei miei "verdi anni", pertanto mi sento particolarmente legato a questo gioco anche per questo motivo.

Un saluto a tutti e...alla prossima!

di **Marco Pistorio**

GIUDIZIO FINALE

» Giocabilità 90%
Gioco fluido, con un grado di difficoltà crescente ben calibrato. E' il gioco ideale per chi possiede un buon occhio, veloci riflessi e la giusta dose di pazienza.

» Longevità 70%
E' difficile probabilmente che lo si rigiocherà molte volte, a causa del fatto che il gioco non finisca mai ma riprenda sempre dall'inizio .

TINY BOBBLE

Anno: 2020
Sviluppatore: Abyss

Genere: Platform
Piattaforma: Amiga

Il gruppo chiamato Abyss in questi anni ci ha regalato una serie di giochi remake di grandi classici, tutti chiamati Tiny.

Un lavoro notevole che ha sempre riscosso presso gli utenti molto successo.

In questa calda estate "anomala" hanno fatto uscire Tiny Bobble, conversione "arcade perfect" (così la definiscono gli sviluppatori) per Amiga.

L'originale versione uscita negli anni 80, pur gradevole e d'effetto, mancava di similitudine con la macchina da sala ed era un port della versione Atari ST, proprio per questo il collettivo Abyss ha messo mano al codice di gioco originale e ha tirato fuori questa versione.
Prima della versione definitiva, hanno messo sul mercato (gratuitamente) diverse beta version che facevano ben sperare.

Cosa hanno modificato o migliorato? Ecco alcuni miglioramenti rispetto al gioco originale per Amiga:
- Necessita solo di un singolo file "minuscolo" da 178kb (l'originale necessitava di un disco pieno)
- 50 fps (invece di 25 fps)
- 32 colori (invece di 16)
- 150 power up (invece di 40)
- Altezza dello schermo originale di 224 pixel (era di soli 200 pixel)
- Quasi tutte le animazioni sprite (aveva solo il 20% circa di animazioni)
- Schermata di avanzamento (non era disponibile)
- Animazione dei Big Enemies ogni 16° livello (non disponibile)
- Animazione del Player Two Join (non disponibile)
- Immagini del Big Score (non era disponibile)
- Schermo esteso animato (non era animato)
- Schermata Pozione animata (non animata)
- Introduzione animata (non era animata)
- Boss fight animata (era un'immagine fissa)
- Finali multipli (aveva solo un finale)

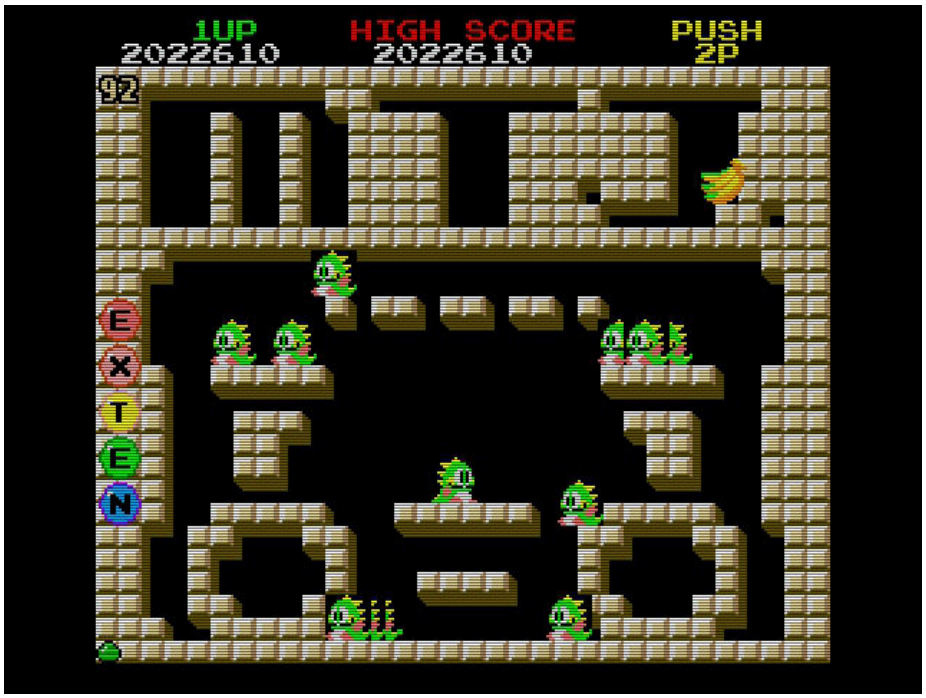
Tutte premesse mantenute.
Qualitativamente è un bel vedere e anche dal punto di vista audio abbiamo una bella conversione e una qualità audio di tutto rispetto.

Tutta la codifica è stata eseguita con Amiga C/C++ Compile ed è visionabile sul sito degli sviluppatori.

Però… E si', c'è un però...

Non è realmente arcade perfect.

E' molto bello, molto simile, ben sviluppato ma è eccessivamente facile. Semplificato.



GIUDIZIO FINALE

» Giocabilità 90%

E' un'esperienza quasi perfetta.
Giocabile come il più classico dei Bubble Bobble ma in alcuni punti non proprio "perfetto" come promesso.

» Longevità 90%

E' sempre Bubble Bobble, ma è molto più semplice e con molte lacune (MANCANO le stanze segrete!! ndN).

Alcuni power up e bonus sono assenti o in posizioni non corrette (e si lo so, sono pignolo) e mancano le stanze segrete (vera sfida di resistenza del gioco originale).

Peccato davvero, perché lo sviluppo è davvero quasi perfetto e totalmente simile all'originale.

Criticabile inoltre la scelta di far uscire la versione definitiva una settimana dopo l'ultima beta. Avrei perso qualche settimana di più.

Attenzione non è un fallimento. E' sempre un bellissimo Bubble Bobble.

Il gioco è disponibile in formato ADF per tutti gli Amiga con almeno 1 mb di ram, gira perfettamente anche emulato su WinUAE e con i Core sul MisterFPGA.

di **Carlo N. Del Mar Pirazzini**

BLACK THORNE

Publisher: Interplay
Anno: 1995
Piattaforma: Snes
Genere: Platform

La mia esperienza di videogiocatore è nata nelle sale giochi degli anni 80 e primi anni 90. A casa mia invece ho sempre preferito i computer e così sono passato dal mitico commodore 64 direttamente al mio primo pc 386. In pratica ho saltato completamente la fase delle console, in particolare quelle a 16 bit.

Ma proprio grazie ai computer ho scoperto il fenomeno dell'emulazione e questo mi ha permesso, e mi permette ancora oggi, di poter giocare gemme del passato.

Tra questi titoli oggi vi voglio parlare del mio preferito, che purtroppo non ha avuto quella consacrazione che invece meritava pienamente, sto parlando di Black Thorne versione super nintendo.

Il titolo è stato sviluppato dalla Blizzard per Interplay nel 1994 (pc e mac) e 1995 (snes e sega 32x) e già questo è un ottimo biglietto da visita. Ma aldilà delle sue nobili origini, Black Thorne presenta un gameplay originale ed una giocabilità incredibile ancora oggi.

Cominciamo con un po' di storia. In questo gioco vestiamo i panni di Kyle, erede al trono di un regno che è stato conquistato dal perfido Sarlac che, grazie ad un esercito di orchi, ha ridotto il popolo degli Androthi alla schivitù. Partendo dalle miniere dell'usurpatore, dobbiamo risalire i vari livelli fino allo scontro finale con lo stesso Sarlac e riportare l'ordine nel regno.

Il gioco è un platform in perfetto stile Prince of Persia e Another World, con movenze e dinamiche che fanno il verso anche al bellissimo Flashback. Perciò il nostro scopo è quello di superare ogni schermata muovendoci tra scale, ponti, piattaforme, buche da saltare e mostri da uccidere.

Il nostro eroe è armato di un potente fucile a pompa che si potenzia superando i livelli e inoltre può raccogliere e inserire nel suo inventario granate, chiavi, bombe radiocomandate e pozioni curative.

Per superare ogni schermata dobbiamo sempre ragionare bene sulla mossa giusta da fare e inoltre spesso ci capiterà di risolvere indovinelli o trovare passaggi nascosti.

Durante il nostro cammino incontreremo orchi e mostri con i quali dobbiamo ingacciare duelli fino all'ultimo colpo di fucile. Inoltre possiamo interagire anche con prigionieri Androthi che ci danno informazioni importanti o ci lasciano oggetti da usare nella nostra avventura.

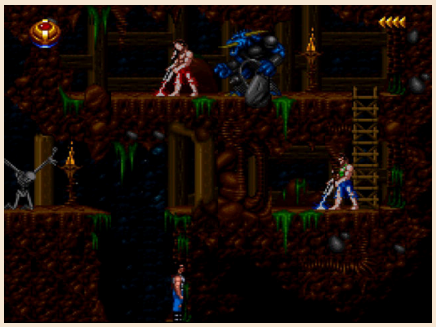
Arrivati nella sala del trono, per sconfiggere il perfido Sarlac dobbiamo combinare al meglio le nostre mosse e cioè salti, schivamenti e rotolamenti, altrimenti non avremo mai la meglio sul mostro.

Blach Thorne ha una grafica stupenda, con sprites dettagliati e colorati. Ottimo anche il comparto sonoro con musiche ed effetti che contribuiscono a creare la giusta atmosfera.

Black Thorne è sicuramente rude e violento, come dimostra lo spargimento di sangue continuo o la possibilità di poter anche sparare e uccidere i prigionieri. Questo purtroppo contribuì alle critiche verso questo titolo e influì anche sulla mancata creazione di un suo seguito, che invece ci stava assolutamente.

Ciò nonostante vi consiglio davvero di recuperare questo capolavoro che non troverete mai tra le classifiche dei migliori giochi del super nintendo, ma che invece offre un esperienza di gioco davvero
unica.

di **Querino Ialongo**



GIUDIZIO FINALE

» Giocabilità 90%

Black Thorne porta la giocabilità di Prince of Persia e Flashback al massimo livello. Se vi piace il genere lo amerete alla follia ancora oggi.

» Longevità 90%

Ogni salto, ogni movimento, ogni sparo va pensato e calcolato. Questo contibuisce a rendere il gioco molto longevo. Alla fine vi accorgerete che per terminarlo vi occorreranno diverse ore di gioco.

ANGOLO OSCURITA'
STURMWIND

Publisher: Duranik
Anno: 2013
Piattaforma: Dreamcast
Genere: Shoot'em up

Se c'è qualcosa che gli appassionati di computer e console hanno imparato, è che l'obsolescenza prima o poi arriva. Potrebbe essere tra un anno, cinque o dieci, ma alla fine un sistema viene inevitabilmente lasciato in panchina, mentre altre macchine più potenti e versatili ne prendono il posto. Ma cosa succede a coloro che escono dal mercato? Sono destinati a sparire nella memoria, ricordati solo da un gruppo di nostalgici appassionati?

Non necessariamente.

Ci sono persone che ancora credono nel passato e lo supportano attivamente; ecco quindi che console e computer "storici" vengono arricchiti da nuove uscite annuali, sviluppate con passione e dedizione da veri e propri eroi romantici.
Uno degli esempi più interessanti, in tal senso, è Sturmwind, uno sparatutto a scorrimento orizzontale creato dal team tedesco Duranik.

Nato in origine su Atari Jaguar nel lontano 1997, con il titolo provvisorio "Native", il gioco è stato poi spostato prima su Nuon, una delle console più oscure di sempre, trovando infine casa su Dreamcast. Uscito sotto etichetta Red Spot Games nel 2013, Sturmwind è la dimostrazione di come bravura e testardaggine possano vincere la sfida del tempo.

Dopo un coreografico filmato introduttivo, veniamo introdotti al menù principale, attraverso cui possiamo entrare nel vivo del gioco. Ci aspettano sedici livelli di azione al cardiopalma, suddivisi in sette mondi, con un centinaio di nemici diversi ad attenderci ed una ventina di giganteschi boss da affrontare. Considerando la media di durata di uno sparatutto di questo tipo, siamo su livelli assolutamente ragguardevoli.
Iniziamo dall'impianto estetico: graficamente Sturmwind fa cadere la mascella a terra.

I programmatori hanno spremuto la Dreamcast come un limone ed il risultato si vede eccome.

Il motore grafico presenta elementi bidimensionali e tridimensionali perfettamente fusi insieme, fondali interattivi, effetti particellari, riflessi, simulazione di liquidi e tonnellate di nemici contemporaneamente su schermo.

I livelli sono sorprendentemente vari, sia per ambientazioni che per tematiche, e sono contraddistinti da una fluidità ineccepibile, che si mantiene costante anche quando incontriamo i giganteschi boss, animati in maniera splendida. Vedere muovere il tutto su schermo fa una certa impressione, soprattutto quando la nostra cara Dreamcast è attaccata in Vga ad un monitor CRT.

Anche il reparto audio si difende benissimo, con una colonna sonora di ottima qualità (come ci si aspetta dai compositori tedeschi), sempre calzante e integrata nell'azione. Anche gli effetti sono ben realizzati e roboanti al punto giusto.

Uno degli aspetti più interessanti di Sturmwind è di non esser stato pensato per un pubblico di soli smanettoni. Anche chi non ha grande esperienza nel genere si troverà perfettamente a proprio agio, grazie alla possibilità di scegliere il livello di difficoltà, nonché di poter salvare i progressi ad ogni nuovo capitolo raggiunto. In questo modo la sfida è più accessibile e rende il gioco davvero per tutti. Sono supportati tutti i tipi di controller, dal pad standard agli arcade stick, la VMU e persino il Rumble Pack (che è anche configurabile).

Oltre alla modalità normale è presente anche quella arcade, dove dobbiamo superare sei livelli senza continuare, e tutta una serie di extra come i trofei (simili a quelli dei giochi moderni), che aprono una sezione dedicata a disegni preparatori, sketch e modelli.
E' presente anche la possibilità di poter inserire il proprio punteggio sul sito ufficiale del gioco.

Quindi che dire? Sturmwind è un vero e proprio atto d'amore verso Dreamcast, che nessun appassionato di shooter a scorrimento dovrebbe perdersi. Se poi siete in vena di spendere e volete cercare la limited edition, ci troverete anche un cd con la colonna sonora e uno dei boss (una piovra gigante) in una tenerissima versione peluche. Per chi fosse sprovvisto della console di casa Sega, potete comunque recuperare la recente versione EX, che è disponibile su PC, Xbox One e Switch.

Sturmwind si rivela quindi un esempio da seguire e dimostra ancora una volta che le nostre amate macchine possono continuare a stupirci, in barba alle logiche commerciali ed al principio dell'obsolescenza. Il passato è passato, ma può rivivere sempre, l'importante è non smettere di crederci.

Alla prossima!

di **Federico "Arzak1" Gori**

GIUDIZIO FINALE

» Giocabilità 90%

Grazie ad una varietà notevole di ambientazioni e situazioni, nonché a un livello di difficoltà scalabile, il gioco si rivela molto immediato e divertente, risultando adatto sia ai neofiti che agli appassionati del genere.

» Longevità 85%

La quantità di livelli, modalità ed opzioni presenti, rendono il gioco molto più longevo della media del genere, regalandoci un'esperienza completa e duratura.

Il Basic che visse due volte. Anzi di più!

Il porting è il processo di trasposizione, a volte anche con modifiche, di un componente software, volto a consentirne l'uso in un ambiente di esecuzione diverso da quello originale (fonte Wikipedia).
A proposito del linguaggio Basic, non si parla però di porting quanto di conversione, o adattamento, in base al "dialetto" utilizzato.

Quanti di noi in passato hanno provato ad adattare codice Basic, specifico per un computer, per poterlo utilizzare su un altro? Oggi come oggi, dato che abbiamo tutto il software a portata di mano, è una pratica quasi perduta, ma in passato era una vera e propria esigenza. Pensate a quando compravate le riviste e trovavate i listati in Basic per una moltitudine di macchine ma voi ne possedevate solo una... quanti programmi, soprattutto giochi, sprecati. Perché quindi non provare ad adattare un listato, nato per girare su un'altra macchina, alla nostra? Chi si è mai cimentato in questa impresa si sarà reso conto della difficoltà. Lungi dall'essere complesso come un porting reale, dove l'adattamento del codice macchina la fa da padrone, anche la conversione Basic non scherza affatto.

Il Basic, a differenza di altri linguaggi di programmazione, ha visto proliferare una miriade di dialetti a volte configurabili come veri e propri linguaggi distinti. Date un'occhiata a questa lista giusto per farvi un'idea di massima:
https://en.wikipedia.org/wiki/List_of_BASIC_dialects
E la lista, per quanto estesa, è lungi dall'essere completa.
Proviamo, per esempio, a pensare a tutte le estensioni Basic che negli anni sono nate per potenziare il Basic V2 del Commodore 64; in questa lista se ne trovano soltanto alcune...

Tutta questa lunga introduzione per dirvi che personalmente mi sto mettendo all'opera per trasferire alcuni listati Basic nati per piattaforme specifiche e 'portarli' in altre.

Perché tutto questo sforzo? A che pro?

Prima di tutto perché è divertente ed educativo. Viste le differenze sostanziali di molti dialetti, a volte è necessario riscrivere la logica del programma per trasferirlo. In questo modo mi assicurero' di aver ben compreso il codice e soprattutto di avere dimestichezza con il dialetto di destinazione.
Secondariamente perché mi piace dare una seconda (o terza, quarta...) vita a codice che altrimenti finirebbe dimenticato.
Entrambe le ragioni si sposano perfettamente con la mission della nostra rivista.
Inoltre questo esercizio getta le basi per un progetto di RMW che è ancora in erba, ma che contiamo di far crescere al più presto.
Vi invito inoltre a provare questo stesso esercizio, soprattutto se siete programmatori alle prime armi. Vi garantisco che vi si aprirà un mondo.

Volete saperne di più? Continuate a seguirci, ne vedrete delle belle!

Francesco Fiorentini

RetroMagazine World
Anno 4 - Numero 25 - SETTEMBRE 2020

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