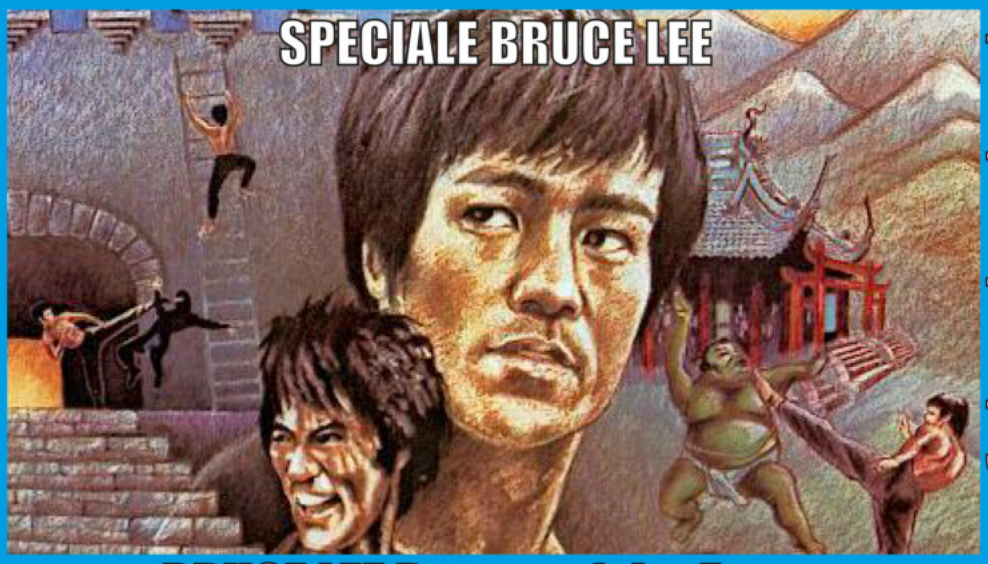
RetroMagazine

semplicemente retro

Numero 17 - Anno 3 - Settembre 2019 - WWW.RETROMAGAZINE.NET - Pubblicazione gratuita



BRUCE LEE Return of the Fury

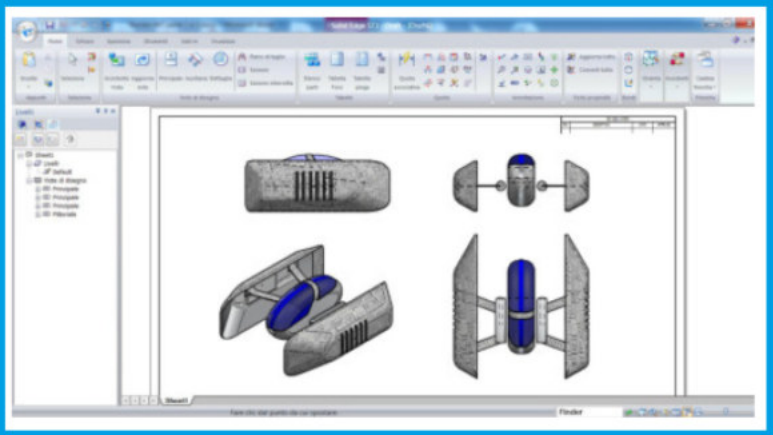
- IL FORMATO D64 (Seconda parte)
- Leggiamo i file .WAV sullo Z80NE
- I BUG NEI VIDEOGIOCHI RETRO'
- Retro Eventi -
- "Una intro...per iniziare" Assembly su C=64

PITFALL II The Lost Caverns

Dark Side of 8 Bit:
SORD M5 vs SEGA SC-3000

```
double dblTemp;
bool again = true;

while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    iLength = sInput.length();
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLength - 3] != '.') {
        again = true;
        continue;
    } while (++iN < iLength) {
        if (isdigit(sInput[iN])) {
```

Cenni di GAME CODING



NUCLEO 447 (C64)
DIARIO DI SVILUPPO PARTE 2 DI 2

! Oltre 60 pagine per rivivere l'età dell'oro dell'informatica !

A Settembre chi è esperto non viaggia mai scoperto!

Anche i proverbi, saggezza popolare italiana, sembrano ricordarci che a Settembre l'estate e' finita. Bentornati quindi cari lettori al nuovo numero di RetroMagazine, dopo la meritata sosta estiva. Mi auguro che abbiate trascorso ottime vacenze, durante le quali potreste aver avuto modo di prendere in man oi nostril amati retrocomputer per programmare, sperimentare o semplicemente giocare a quei giochi che tanto hanno riempito le giornate della nostra infanzia. Se lo avete fatto, vi invito a condividere con noi, come qualcuno gia' sta facendo, le emozioni che avete provato o le cose nuove che avete scoperto e che volete condividere con noi.
Noi di RetroMagazine, del canto nostro, abbiamo approfittato della pausa estiva per sperimentare nuove cose ma anche per stringere nuove collaborazioni con chi come noi si impegna per diffondere la passione per il retrocomputing.

AmigaGuru Gamer's blog
Come annunciato sulla nostra pagina, da qualche settimana abbiamo stretto una collaborazione con il blog gestito da Gianluca Girelli e Tony Aksnes. Il frutto della collaborazione si e' gia' tradotto in alcuni articoli di Gianluca pubblicati sulle pagine di RM e in alcune recensioni di RM tradotte in inglese e pubblicate sul blog. Che dire chi ben comincia e' a meta' dell'opera ed in questo caso non c'e' che da sperare che il proficuo interscambio duri il piu' a lungo possibile.

FRGCB – Finnish Retro Game Comparison Blog
Ho scoperto tardi questo blog gestito da FRGCB Dude, trovate il link piu' sotto nell'articolo dedicato, ma non ho certamente perso tempo nel contattare FRGCB Dude e chiedere il permesso di poter utilizzare parte del suo contenuto nella nostra rivista. Ovviamente non ci siamo limitati a riportare il contenuto sulla nostra rivista ma abbiamo tradotto il tutto in italiano a beneficio dei nostri lettori che non masticano l'idioma di Albione. E non solo…

Da cosa nasce cosa…
Dall'articolo di Bruce Lee pubblicato su FRGCB sono sorte alcune domande su quale fosse l'hardware per il quale il titolo fosse stato originariamente sviluppato. Siamo felici di poter pubblicare un contributo di Filippo Santellocco da cui si evince come la piattaforma originale sia stata l'Atari 8-bit. Colgo l'occasione per ringraziare Filippo per il suo articolo che ha voluto condividivere con noi.

Ma non finisce qui! Parliamo anche di confettura… Curiosi? Date un'occhiata all'ultima pagina per saperne di piu'.

Francesco Fiorentini

SOMMARIO

Hanno collaborato alla stesura di questo numero di RetroMagazine :

- Ermanno Betori
- Marco Pistorio
- Daniele Brahimi
- Antonino Porcino
- Gianluca Girelli
- The Dump Club 64 Team
- Starfox Mulder
- Adriano Avecone e Andrea Pastore
- David La Monaca (Cercamon)
- Filippo Santellocco
- Leonardo Vettori
- Francesco Fiorentini
- Alberto Apostolo
- Michele Ugolini
- Querino Ialongo
- FRGCB Dude
- Giorgio Balestrieri
- **Flavio Soldani - Realizzazione della copertina**

Dark Side of 8 Bit - SORD M5 vs SEGA SC-3000

di Ermanno Betori

Nei primissimi anni 80 vi fu un'esplosione di modelli di computer, in Italia si ricordano macchine dell'epoca quasi sempre i Commodore Vic20 e C64 oppure lo ZX Spectrum. Più raramente vengono menzionati Il TI99/4A oppure i sistemi MSX, ma sono stati quasi del tutto dimenticati degli ottimi computer giapponesi che vennero venduti in Europa ed in minima parte anche in Italia contemporaneamente al Commodore64 ed allo ZX Spectrum 48K.

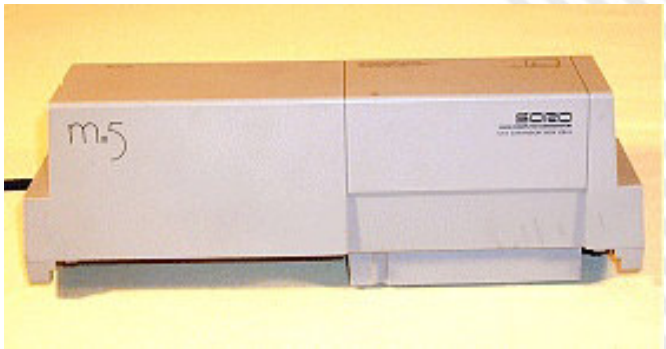
Tra loro ricordiamo il Sega SC-3000:

ed il Sord M5:

Entrambi i computer furono progettati usando la stessa componentistica di base, infatti hanno come cuore una CPU Z80A a 3.58 MHz, un chip video della Texas Instruments il TMS9929 (Pal) e come chip sonoro il SN76489AN. In pratica sono i discendenti del TI99/4A in quanto differiscono solo della cpu, uno Z80 (8bit) al posto di un TMS9900 (16bit) e sono gli zii del futuro sistema MSX in quanto tra i due sistemi cambia solo il chip audio; l'MSX monta infatti un General Instrument AY-3-8910.

Parliamo ora del **M5**; la ditta Sord venne fondata nel 1970 dall'allora ventisettenne Takayoshi Shiina. Era una ditta a conduzione familiare raffrontata ai colossi del settore dell'epoca ma si seppe ritagliare un suo mercato. La prima idea commerciale della ditta fu quella di entrare nel mercato business con macchine da ufficio (M23, M35), ma la domanda crescente di home computer, la convinse a proporre un sistema abbastanza simile allo ZX Spectrum, il Sord M5.
Questo computer fuori del Giappone vendette poco, specialmente in Europa dove trovò solo in Cecoslovacchia un mercato fiorente, ma stranamente riuscì a ritagliarsi un suo spazio, benché piccolo, in Inghilterra patria dei computer Sinclair. In Italia? Rumors dicono che fu usato in scuole nel Nord Italia ed in Toscana, ma ad oggi ancora non sono riuscito a contattare personalmente alcuno di questi pionieristici insegnanti.
Le caratteristiche tecniche di questa macchina sono interessanti: oltre ai tre principali chip sopra enunciati, troviamo un CTC (timer Counter) e nella versione base una EPROM da 8 Kb e una RAM da 4 Kb che si unisce ai 16Kb della video ram insita nel TMS9929. Le capacità video che ritroveremo su molti computer dell'epoca sono di 192x256 pixel per l'alta risoluzione e 24 righe per 40 colonne in modo testo, 16 colori e la possibilità di gestire fino a 32 sprite cosa all'epoca per nulla scontata. Il chip sonoro SN76489AN e' capace di generare 3 canali da 6 ottave più uno di rumore.
Le porte di I/O sono completate da una interfaccia centronics parallela, dall'uscita per registratore (DIN a 8 pin) e da due connettori per Joystick/Paddle di fabbricazione proprietaria. Il registratore a cassette (ne esiste una versione ufficiale della Sord), si interfaccia con il sistema alla velocità di 2 Kbit/sec alias 1200 boud. Altra caratteristica è la tastiera in gomma, come quella del Sega SC-3000, entrambe di buona qualità rispetto a quella dello ZX Spectrum.
Avendole provate entrambie, ritengo, anche se di poco, migliore quella del computer Sord, sia come manifattura che come design.

Il Sord M5 aveva una serie di periferiche dedicate che ne espandevano le funzionalità. Fra queste:

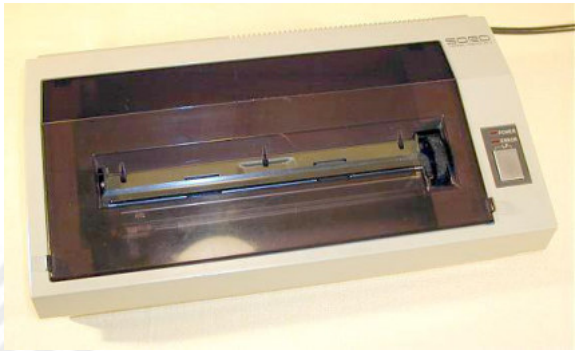


Expansion box (EB-5)

Joysticks (JS-5)/Joypads (JP-5) e Cartridge multiplexer (EC-5)

Parallel I/O cartridge (PI-5) e 32 KB RAM expansion (EM-5)

Tape interface (DR-5)



Thermal Printer (PT-5)

Floppy disk drive (FD-5)

Come vediamo il Sord M5 è un computer costruito in modo modulare, concetto molto in voga in quegli anni che da una parte permetteva un acquisto in base alle proprie necessità ma di contro non permetteva a chiunque di avere il computer completo e di rendersi conto delle reali possibilità, stessa problematica la ritroviamo sul TI99/4A, Il sega SC-3000, il Coleco Adam, il Mattel,ecc...

Tale situazione anche se limitata principalmente solo all'uso delle memorie di massa, la ritroviamo sullo ZX Spectrum, Commodore 64, in parte sullo Spectravideo 328 e sui sistemi MSX che sono successivi come idea progettuale. In pratica per avere una macchina funzionale un appassionato dovrebbe possedere oltre alla consolle M5 anche il cartridge expander, l'espansione di memoria ed ovviamente il registratore a cassette.

Come software sia il Sord M5 che il Sega SC-3000 nascono "nudi", cioè non hanno nessun linguaggio o sistema operativo precostruito ma necessitano di cartucce che lo contengano.

Il Sord ha tre tipi di basic:
- I=Initial, definito "Easy BASIC for Beginners";
- G=Graphics/Game, "Easy BASIC for Games";
- F=Scientific, "Floating Point".

I basic G e F per funzionare richiedono l'espansione di memoria da 32k. Ecco la ragione della necessità di possedere l'expander e l'espansione di memoria.

Il computer **SC-3000** ha invece avuto una genesi diversa; è il primo e unico computer ad essere stato disegnato e costruito dalla Sega. Nacque come evoluzione del Sega Game 1000, consolle sviluppata attorno alla cpu NEC 780C (clone del Zilog Z80), al chip video della Texas Instruments TMS9928A ed a quello audio SN76489. Il concetto di una macchina da giochi a cartucce era di grande interesse industriale e date le buone possibilità di guadagni con la vendita di video-giochi iniziarono a spuntare molti cloni, come l'Othello Multivision della Tsukada Original e il Telegames Personal Arcade (che poteva far girare sia i giochi del Colecovision che quelli SG-1000).

L'interesse del pubblico per gli home computer convinse Sega a seguire in parallelo un secondo progetto. Oltre alla console da giochi SG-1000 (e sue evoluzioni verso il Sega Master System), la società giapponese volle sfruttare la stessa architettura hardware per lanciare nel 1983 anche l'SC-3000, un computer di fascia bassa che era in grado di far girare i giochi della consolle SG-1000 ma anche altre applicazioni scritte in linguaggio macchina o BASIC.

Nonostante il costo decisamente superiore dell'SC-3000 rispetto all'SG-1000 ebbe un buon successo di

vendita come home computer per principianti, non a livello mondiale ma in specifici paesi tra i quali troviamo l'Italia. Per questo motivo furono vendute versioni localizzate del computer con la tastiera personalizzata per i seguenti paesi: Giappone, Australia, Nuova Zelanda, Francia, Italia and Finlandia.

Ad esclusione dell'Italia e della Francia ebbe anche una buona vendita come vero e proprio personal computer, grazie al modulo di espansione SF-7000 che integrava porta seriale, parallela, floppy drive ahimè da 3" (quelli rettangolari) e una espansione di memoria da 64k. Come detto precedentemente il computer senza nessuna cartuccia non funzionava, esattamente come la consolle SG-1000 la cui RAM a disposizione della CPU è di 1Kbyte, mentre nel SC-3000 è doppia.

Del linguaggio basic furono create tre versioni:
- BASIC Level II B (1983); espande la ram della cpu di ulteriori 2Kbyte;
- BASIC Level III A (1983); espande la ram della cpu di ulteriori 16Kbyte;
- BASIC Level III B (1983) e per finire il top di gamma che espande la ram della cpu di altri 32Kbyte;

Il basic che normalmente arrivò sul mercato italiano tramite il distributore Melchioni fu la cartuccia del BASIC Level III A e successivamente anche quella del BASIC Level III B che di norma era venduta insieme alla console.

Alla fine del 1983 erano state vendute a livello mondiale circa 120.000 unita, ricordo nel 1984 dei rumors che giravano in Italia tramite i rivenditori di computer che dicevano che la Sega aveva avuto un rallentamento nella consegna delle vendite per il SC-3000, dovuto al fatto che in Taiwan la fabbrica che produceva la maggior parte della componentistica dove si rifornivа la Sega aveva subito un incendio.

La cosa curiosa conosciuta molti anni dopo era relativa alla ditta Sord che nel 1983 ebbe una enorme espansione commerciale, e per questo exploit aveva ricevuto una ottima offerta di acquisizione da parte una grossa industria di computer. Dopo aver rifiutato tale proposta la Sord nel 1984 ebbe problemi nel rifornimento dei componenti ed altre problematiche logistiche, cosa che questa è confermata.. la domanda che nasce spontanea è se ambedue queste ditte subirono gli effetti di una guerra commerciale, infatti le tecniche giapponesi di dissuasione erano e sono "sottili"...

Mostriamo qui di seguito il Sega SC-3000H versione con tastiera meccanica completo dell'expander SF-7000.

Come vediamo la disposizione per l'uso effettivamente è quella mostrata in figura e la forzatura è dovuta al cavo di giunzione corto, ciò comporta uno scomodo uso cosa constatata personalmente.

Considerazioni finali: i computer sul lato costruttivo sono equivalenti con hardware molto ben curato. Come miglior design la lotta è dura, entrambi hanno le loro peculiarità, come software vince il Sega grazie alle molte cartucce create per la consolle SG-1000. Entrambi possono dare delle belle soddisfazioni, personalmente li consiglio per il collezionista esigente o per lo sviluppatore di retro-software che si vuole cimentare in sistemi non comuni ma non troppo. :-)

Una intro... per iniziare:
Programmazione Assembly su Commodore 64 - Terza puntata

di Marco Pistorio

Torniamo a parlare di rasterbars, provando ad approfondire i concetti che permettono di generare questo effetto grafico presente in tantissime "intro", un effetto che trovo da sempre affascinante.
Nei numeri precedenti 6 e 7 di RetroMagazine abbiamo cercato di sviluppare una semplice "intro" ottenendo due rasterbars orizzontali molto sfrangiate e quindi di impatto visivo modesto.
Il mio intento era quello di mostrarvi l'estrema facilità nell'ottenere tale effetto, ed ho volutamente sorvolato su uno degli aspetti "chiave" per realizzare un effetto più preciso, ovvero la temporizzazione, aspetto che proverò a coprire in questa terza puntata.

Ricordiamo che lo schermo video viene continuamente ridisegnato, con una frequenza di 50 Hz (ovvero 50 volte al secondo) oppure di 60 Hz (60 volte al secondo) in funzione del sistema video in uso (PAL oppure NTSC), riga per riga, indipendentemente dal fatto che l'immagine riprodotta nel frattempo sullo schermo cambi o meno.
Immaginiamo di trovarci in corrispondenza di una linea di schermo ben precisa, e di colorare con il rosso il bordo dello schermo e quello dello schermo , per poi colorare pochi istanti gli stessi elementi con il colore nero.
In funzione del fatto di trovarci già in un istante in cui il bordo è stato già disegnato, il risultato potrebbe essere quello di riuscire a colorare solo una parte della riga dove si trova il pennello ottico, perché la zona precedente della stessa riga è stata disegnata prima del nostro cambio di colore.
Analogamente, ciò potrebbe succedere anche mentre reimpostiamo il colore nero.
In conclusione, in base al preciso istante in cui interveniamo, potremmo riuscire a colorare solo una parte di una riga di schermo. Il risultato? Una linea sfrangiata.
Appare quindi evidente che tali interventi dovrebbero avvenire in istanti "calcolati" in base al tempo necessario al cannone elettronico per attraversare ciascuna riga.
Quanto tempo serve al cannone elettronico per attraversare una riga dello schermo video?
63 cicli di tempo macchina, nel sistema video PAL, oppure 65 cicli di tempo macchina nel sistema NTSC.
C'è anche un secondo problema che è necessario prendere in considerazione.
Le cosiddette "Bad Lines". Ogni 7 righe di schermo video, alla ottava quindi di ciascun gruppo di 8 righe, il processore recupera una riga di codici di carattere dalla memoria, perdendo ben 40 preziosi cicli di tempo macchina. Il tempo quindi a nostra disposizione per operare in queste righe di schermo scende a soli 23 cicli macchina (PAL) / 25 cicli macchina (NTSC).
Chi volesse approfondire l'argomento "BAD LINES" può leggere il documento

http://www.zimmers.net/cbmpics/cbm/c64/vic-ii.txt

che esamina questo e molti altri argomenti legati al chip grafico VIC2 in maniera esaustiva, con diagrammi temporali ed amenità varie ed eventuali, solo per veri uomini, duri e puri :)
Come temporizzare correttamente i cambi di colore per ottenere delle rasterbars definite e senza alcuna sfrangiatura?
Ricordo, quasi trent'anni fa, le difficoltà di trovare informazioni tecniche puntuali. Erano i tempi delle "crews", formate da hackers, crackers ed aspiranti tali, gruppi che era possibile frequentare solo dopo aver dato prova delle proprie qualità.
Non è stato facile neanche oggi, frugando in internet, trovare subito spunti chiari, ben commentati ed immediatamente usabili a proposito del raster stabile.
Oggi come allora, il raster stabile è una specie di Santo Graal.
Tuttavia essere in grado di "dominare" il raster apre le porte alla realizzazione, almeno potenzialmente, di molteplici applicazioni. Ne cito una a titolo di esempio, il "multiplexing degli sprites", che permette di superare agevolmente il limite degli 8 sprites contemporaneamente visibili sullo schermo del c64.

Aprendo la pagina presente al seguente indirizzo:

https://codebase64.org/doku.php?id=base:stable_raster_routine

trovai una routine molto interessante, che sembrava proprio fare al caso mio.
"Codice stabile" leggevo ad un certo punto. Pensai: "è fatta!".
La routine si basa sul principio del doppio IRQ, uno dei due metodi e probabilmente il più adoperato per raggiungere gli scopi di cui vi accennavo.
Il secondo metodo è quello di sfruttare un timer ottenuto tramite opportuna programmazione del chip CIA 6526.

Cercherò di spiegare il funzionamento del metodo del doppio IRQ nella maniera più semplice possibile, affinché risulti più semplice la comprensione del meccanismo adoperato, anche se non sarà una

spiegazione rigorosa.
E' necessario attendere esattamente 63 cicli macchina, nel caso il sistema video sia PAL, altrimenti 65 cicli macchina, se il sistema video è invece quello NTSC, affinché una riga dello schermo venga completamente disegnata.
Imposto una interruzione raster IRQ al raggiungimento di una ben precisa riga di schermo video da parte del cannone elettronico.
Appena viene servita faccio alcune cose, spendendo alcuni cicli macchina, ed imposto una seconda interruzione raster IRQ.
Quando viene servita la seconda richiesta di interruzione perdo ulteriori cicli macchina.
Nell'ultima parte della routine, carico nell'accumulatore il contenuto della locazione $d012 (Linea raster corrente) e confronto tale valore con quello letto subito dopo all'interno della stessa locazione $d012. Se i due valori corrispondono, l'istruzione BEQ successiva viene eseguita, impiegando 3 cicli macchina. Se invece il salto non viene effettuato in quanto i due valori confrontati differiscono, la stessa istruzione BEQ impiegherà solo 2 cicli macchina.
I tempi sono stati calcolati in maniera estremamente precisa, ed il codice eseguito subito dopo questa routine di stabilizzazione sarà eseguito ESATTAMENTE al passaggio del raster alla riga di schermo immediatamente successiva a quella dove è stata impostata la prima interruzione.

Questa routine copre solo una parte del problema.
E' necessaria una seconda routine, che si occupi di colorare efficacemente il bordo e lo schermo, che impieghi non più di altri 63 cicli macchina (se PAL), oppure di 65 cicli macchina (se NTSC). La stessa routine dovrebbe tener conto, in qualche modo, anche del fatto che ci si trovi o meno in una "bad line", ove i tempi a nostra disposizione si riducono, come già detto, a soli 23 cicli macchina (se PAL) oppure 25 cicli (se NTSC). Trovai infatti in rete un post di un appassionato che adoperava la summenzionata routine di stabilizzazione e "contava" i cicli macchina via via che colorava lo schermo video, ottenendo un risultato impreciso. Il bordo dello schermo risultava colorato in maniera diversa rispetto alla parte interna dello schermo.
Nei suggerimenti forniti in risposta a questo post ho trovato un esempio di codice sufficientemente valido per i nostri scopi, abbastanza completo, leggibile e di immediata applicazione, codice che si basa anche sulla routine di stabilizzazione già discussa. Il post è raggiungibile a questo indirizzo:

https://stackoverflow.com/questions/24375150/stable-raster-on-c64

La routine suggerita, che colora il bordo e lo schermo del C64, rispetta dei tempi ben precisi, e può agire correttamente anche se ci si dovesse trovare su una "bad line".

Mettendo insieme queste informazioni e rielaborandole sono riuscito nel mio intento di generare delle barre raster stabili, senza penare troppo. Segue il relativo codice assembly, secondo la sintassi KickAssembler. Se osserverete con attenzione tutte le righe, comprese quelle commentate, comprenderete come sia importante operare sempre con la massima precisione, rispettando i cicli macchina di ciascuna istruzione che dovrà essere eseguita nei tempi previsti, pena la comparsa di righe di colore sfrangiate, rallentamenti vari e/o crash del C64.
La demo è scaricabile in formato sorgente,in formato .prg ed in formato .d64 all'indirizzo:

https://github.com/marcus73/retromagazine_06

Il codice presentato funzionerà solo su C64 in modalità video europea (PAL). Non dovrebbe essere difficile, tuttavia, modificarlo per renderlo funzionante correttamente anche su C64 in modalità video NTSC.

Alla prossima, amici lettori.

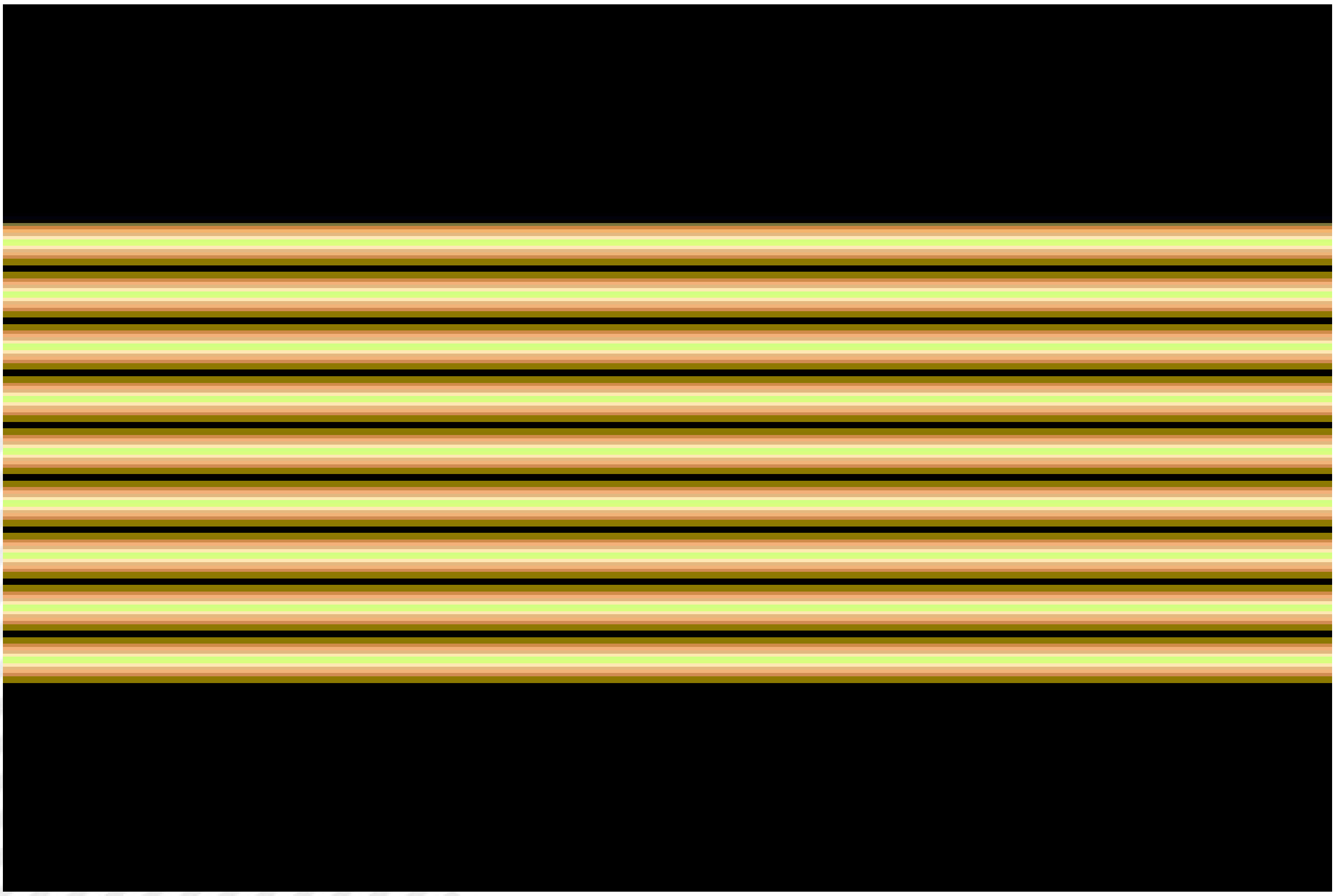
```
//////////////////////////////////////////////////////////////////////////////////
//
//      "STABLE RASTER"
//            REALIZZATA PER RETROMAGAZINE, SETTEMBRE 2019
//
//      VERSIONE PER C64 PAL, SINTASSI KICKASSEMBLER
//
//      RIFERIMENTI:
//                  https://stackoverflow.com/questions/24375150/stable-raster-on-c64
//                  https://codebase64.org/doku.php?id=base:stable_raster_routine
//
//////////////////////////////////////////////////////////////////////////////////

/*
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/


.pc = $0801 "Basic upstart"
:BasicUpstart(main)

           //
           // .byte $0b,$08,$0a,$00,$9e,$34,$30,$39,$36,$00
           //
           // Codice corrispondente alla linea BASIC: 10 SYS4096
           // per i compilatori che non creano automaticamente la linea per il lancio
           // del programma da BASIC.
           //

.const nr_col=143

.pc=$1000 "main"

main:
{
  jsr $ff81// SCINIT standard KERNAL function
                      // Initialize VIC; restore default input/output to keyboard/screen; clear screen; set PAL/NTSC switch and interrupt timer.

  sei                 // disabilita le interruzioni

  lda #$00            // inizializzazione routine suono
  tax                 //
  tay                 //
  jsr $4000           // address of inizialization subroutine

  lda #$35            // disabilita interprete BASIC e KERNAL
  sta $01             //

  lda #<int1          // prima interruzione, che farà eseguire
  ldy #>int1          // la routine int1
  sta $fffe           //
  sty $ffff           //

  lda #%0111111       //
  sta $dc0d           // Interrupt control and status register
                              // Bit #0: 1 = Enable interrupts generated by timer A underflow.
                              // Bit #1: 1 = Enable interrupts generated by timer B underflow.
                              // Bit #2: 1 = Enable TOD alarm interrupt.
                              // Bit #3: 1 = Enable interrupts generated by a byte having been received/sent via serial shift register.
                              // Bit #4: 1 = Enable interrupts generated by positive edge on FLAG pin.

  sta $dd0d           // Interrupt control and status register
```

```
                                // Bit #0: 1 = Enable non-maskable interrupts generated by timer A underflow.
                                // Bit #1: 1 = Enable non-maskable interrupts generated by timer B underflow.
                                // Bit #2: 1 = Enable TOD alarm non-maskable interrupt.
                                // Bit #3: 1 = Enable non-maskable interrupts generated by a byte having been received/sent via serial shift register.
                                // Bit #4: 1 = Enable non-maskable interrupts generated by positive edge on FLAG pin.

  lda #$01              //
  sta $d01a             // Interrupt control register
                                // Bit #0: 1 = Raster interrupt enabled.

  lda $dc0d             // reset per eventuali interrupts rilevati
  lda $dd0d             // reset per eventuali interrupts rilevati

  lda #$1b              // bit più significativo (#8) per la linea raster
  sta $d011             // dove generare l'interruzione azzerato

  lda #$01              //
  sta $d019             // ACK RASTER INTERRUPT

  lda start             // imposto la riga raster iniziale
  sta $d012             // per visualizzare le rasterbars (bits #0-#7)

  cli                   // riabilita le interruzioni

  jmp *
}

.pc=$1100 "routine_stabilize"

int1:
{
  pha
  txa
  pha
  tya
  pha

  :STABILIZE()

           ldx #nr_col

           // a delay to get to some cycle at the end of the raster-line, so we have time to execute both inc's on
           // each successive raster-line - in particular on the badlines before the VIC takes over the bus.
           .for (var i=0; i<28-2; i++) nop

           // just for illustrative purposes - not cool code :)
           .for (var i=0; i<nr_col; i++)
        {

             //inc $d020   // 6 cycles
             //inc $d021   // 6 cycles

             lda colors,x         // 4 cicli
             sta $d020            // 4 cicli
             sta $d021            // 4 cicli

            dex                   // 2 (tolto 1 nop sia da // badline che in // non-badline)

             .if ([i & %111] == 0)
             {
                 // badline

                 //nop
                 nop
                 nop
                 nop // 4*2=8 cycles

             }
           else
             {
                 // non-badline

                 //nop
```

```
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                nop // 24*2=48 cycles

                bit $ea     // 3 cycles
                            // = 63 cycles
            }
          }

  lda #$00
  sta $d020
  sta $d021

  lda #$f0              // riga raster per
  sta $d012             // la riproduzione del brano, verifica tasto spazio premuto etc.

  lda #<play            // la prossima interruzione manderà
  ldy #>play            // in esecuzione la routine
  sta $fffe             // play.
  sty $ffff             //

  lda #$01              //
  sta $d019             // ACK RASTER INTERRUPT

  jsr colora_barre

  pla
  tay
  pla
  tax
  pla

  rti
}

.pc=$2400 "macro STABILIZE"

.macro STABILIZE() {

  lda #<nextRasterLineIRQ
  sta $fffe
  lda #>nextRasterLineIRQ
  sta $ffff

  inc $d012

  lda #$01
  sta $d019

  tsx

  cli
```

```
  nop
  nop
  nop
  nop
  nop
  nop
  nop
  nop

nextRasterLineIRQ:
  txs

  ldx #$08
  dex
  bne *-1
  bit $00

  lda $d012
  cmp $d012

  beq *+2
}

.pc=$2500 "routine_colora_barre"
colora_barre:
{
            lda colors+nr_col               // memorizza colore più in alto del
            pha                             // vettore -colors- nello stack

            ldx #nr_col                     // copia i colori all'interno del
loop:                                       // vettore -colors- spostandoli via
            lda colors,x                    // via di 1 verso l'alto del
            sta colors+1,x                  // vettore
            dex                             //
            cpx #$ff                        //
            bne loop                        //

            pla                             // riprende il colore memorizzato nello
            sta colors                      // stack e lo copia nell'elemento più
                                            // in basso del vettore -colors-
            rts
}


.pc=$2600 "routine play"
play:
{
  pha
  txa
  pha
  tya
  pha

  jsr $4003           // address of play-sound subroutine

  lda $DC01           // verifica se viene premuto
  cmp #$ff            // il tasto SPAZIO.
  beq no_spazio       //

  lda #55             // ripristina BASIC ROM, KERNAL ROM
  sta $01             // ed I/O area

  jmp $FCE2           // warm reset

no_spazio:

  lda start           // imposto la riga raster iniziale
  sta $d012           // per visualizzare le rasterbars (bits #0-#7)

  lda #<int1          // prossima interruzione
  ldy #>int1          // eseguirà nuovamente la routine
  sta $fffe           // int1
  sty $ffff           //
```

```
  lda #$01          //
  sta $d019         // ACK RASTER INTERRUPT

  pla
  tay
  pla
  tax
  pla

  rti
}

.pc=$2800 "bars"
colors:
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00
.byte $00,$09,$09,$08,$0a,$0a,$07,$0d,$0d,$07,$0a,$0a,$08,$09,$09,$00

start:
  .byte 80

*=$4000 "music"
.import binary "resources/retromagfin.sid",126  //// rimuove headers non necessari
```



Rasterbars ottenute con il codice assembly presentato

Queste sono delle semplici intro che ho realizzato sulla falsariga di quanto vi ho presentato in questo articolo. La prima è sicuramente la più somigliante, con qualche elemento grafico in più. La scritta "RetroMagazine" comunque è stata già stata fornita all'interno del materiale dei precedenti articoli sull'argomento pubblicati nei numeri 6 e 7. Perchè non vi “mettete alla prova”, cercando di realizzarle per conto vostro?
Chi è già abile non avrà bisogno di questo esercizio. Se invece intendete diventarlo, fatelo... Se avete bisogno di una mano scrivete pure all'indirizzo email della redazione di RetroMagazine!

https://csdb.dk/release/?id=164022 (“RetroMagazine #1.1”)

https://csdb.dk/release/?id=164473 (“RetroMagazine #2”)

https://csdb.dk/release/?id=181050 (“RetroMagazine #4”)

I bug nei videogiochi retro

di Daniele Brahimi

Ultimamente ho visto spesso sui social network e altrove parlare dei cosidetti bug, errori di programmazione di alcuni videogiochi.
Si presentavano come difetti grandi e piccoli, alcuni con qualche quadratino colorato sullo schermo che andava ad intermittenza, altri invece che addirittura impedivano al giocatore di arrivare fino in fondo al gioco facendolo andare su tutte le furie dopo i soldoni spesi per esso, le ore trascorse davanti al televisore e la reputazione del titolo,magari una conversione tanto attesa. In alcuni video ne ho visti parecchi sia sul Commodore 64 che su altre macchine, praticamente sono ovunque!
Dopodiche' ho visto anche link e dibattitti sui gruppi e forum e con questo vorrei parlarvi di alcuni giochi che ho giocato personalmente che presentavano dei bug talmente evidenti da non lasciare nessuna alternativa al giocatore se non quella di giocare fino a quel punto il gioco ed accontentarsi... senza nemmeno avere la possibilità di un reclamo o un rimborso.

Il primo gioco in cui ne vidi uno durante la mia adolescenza, quando non sapevo nemmeno cosa volesse dire il termine bug, fu il tie in Robocop per Commodore 64; il gioco si presentava bene con quella bella scatola, manuale completo e giocabilità degna del successo del film. Ma con un errore non da poco in uno dei livelli che si trovavano oltre la metà del gioco: il quartiere OCP. Il livello erano pieno di quadratini multicolore che disorientavano completamente il giocatore ed era come percorrere un labirinto ad occhi chiusi con i più nemici che ti sparavano ovunque.

Di recente ho scoperto che quell'errore fu rimediato dai programmatori aggiungendo pochissimo tempo al livello precedente della raffineria in modo che quasi nessuno potesse arrivarci e vederlo. Al tempo stesso ne esisteva un altro, questa volta benigno, sempre in quel livello, ossia passare attraverso il doppio muro muovendo il joystick in varie direzioni arrivando così in tempo alla fine del livello e affrontando il non facile robot ED209 (altro bug?).

Un altro bug sotto i riflettori del quale mi è capitato di leggere, ma che non avevo mai visto di persona in quanto non ero mai arrivato fino a quel punto quando lo giocai fu Rastan. Un'altra conversione che si ispirò molto a Conan il barbaro e che tutti i retrogamer conoscono. C'era un punto in cui bisognava saltare un fiume con le corde appese al soffitto e la distanza non era proporzionata al salto, di conseguenza era impossibile raggiungere l'altra sponda.
Molti prima d'ora pensavano che si trattasse solo di prendere la mano al gioco e saltare con più precisione e prima o poi avrebbero meritatamente superato l'ostacolo. Invece nemmeno la mano più allenata lo avrebbe fatto.

Spesso mi sono domandato perché lasciare errori del genere? Sbagliare è umano ed anche persone preparate ed esperte come i programmatori possono farlo, ma perché non correggere queste errori grossolani? Perché ingannare migliaia di persone in questo modo e soprattutto deluderle così pur sapendolo?

Anche la conversione di Golden axe aveva una cosa simile anche se non so se la si può definire errore. IL gioco terminava un livello prima rispetto all'arcade, ma i programmatori corressero l'errore aggiungendo il livello mancante con annesso il boss finale.

Le mie non sono critiche né accuse e mai mi sognerei di farle su una rivista costruttiva e videoludica come questa, ma solo un invito a non deludere le aspettative dei fan che con tanto amore seguono, passano ore e ore davanti allo schermo e si fanno in quattro per comprare titoli tanto attesi e meravigliosi.

Il formato D64 - seconda parte

di Francesco Fiorentini

Bentrovati amici lettori, nel numero 14 di Retromagazine ho scritto un articolo dedicato al formato file D64 descrivendone la struttura e spiegando in dettaglio come accedere alle informazioni contenute.
Questo seconda parte dell'articolo vuole invece porre l'accento su alcune peculiarita' dei file D64, forse sconosciute ai piu' (ed anche a me fino a pochi giorni fa), ma che non mancheranno di destare l'interesse dei lettori più curiosi e "smanettoni".

Come abbiamo avuto modo di apprendere nella puntata precedente, la BAM o Block Availability Map è un po' il cuore di qualsiasi disco D64. La BAM è infatti il luogo dove, tra le altre informazioni, il DOS tiene traccia dei settori liberi e di quelli occupati.
Se vi siete persi l'articolo precedente vi suggerisco di recuperarlo e leggerlo perchè propedeutico a comprendere i concetti forniti in questo. In quell'articolo abbiamo avuto modo di capire come leggere i dati contenuti in un D64, interpretarli e visualizzarli. In questo nuovo articolo voglio invece soffermarmi su alcuni byte in particolare che, una volta modificati, possono essere utilizzati per celare e/o mascherare le informazioni contenute sul disco.

Sono riuscito a stuzzicare un po' la vostra curiosità?
Spero proprio di sì.

Premessa 1: per modificare le informazioni nei file D64 utilizzeremo uno strumento molto famoso tra i retroappassionati del Commodore 64, un programma chiamato **DirMaster**.
DirMaster rappresenta un po' il coltellino svizzero per accedere e modificare tutte le informazioni nei file D64. Se non lo avete installato sul vostro computer, vi invito a farlo scaricandolo da:
https://style64.org/release/dirmaster-v3.1.3-style
Se lo strumento non vi è familiare, non abbiate timore, l'interfaccia è veramente user-friendly e le funzioni abbastanza intuitive; inoltre cercherò di guidarvi passo passo con le modifiche che dovremo apportare.

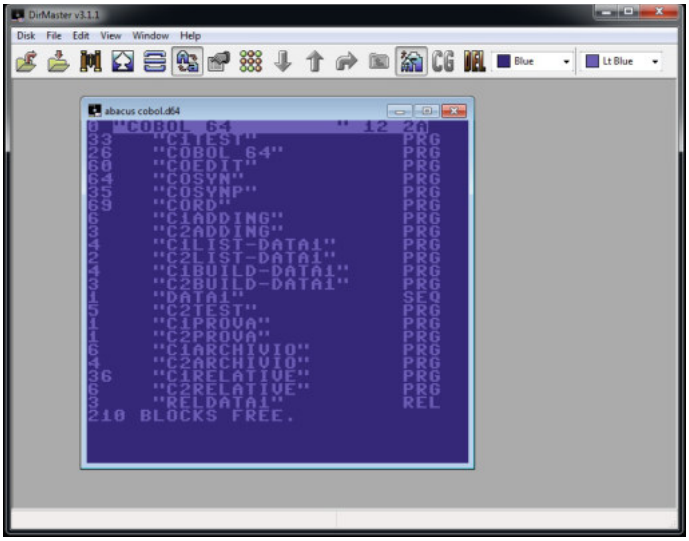
Premessa 2: tutte le modifiche che andremo ad effettuare sui nostri dischi saranno abbastanza invasive e potrebbero danneggiarli irreparabilmente. Suggerisco quindi, se vorrete ripetere le prove descritte nell'articolo, di utilizzare una copia ad-hoc di un disco, in modo da non rovinare l'originale. Personalmente utilizzerò ancora la copia del disco abacus cobol.d64 come fatto nel precedente articolo.
E' inoltre doveroso ribadire che l'autore dell'articolo e la redazione di RetroMagazine declinano ogni responsabilità su eventuali danni che potrete arrecare ai vostri dischi D64 nel cercare di ripetere i test proposti nell'articolo.

Alla buon ora... Vediamo di cominciare ricordandoci che i byte si contano partendo dal numero 0 (come Di Maio).

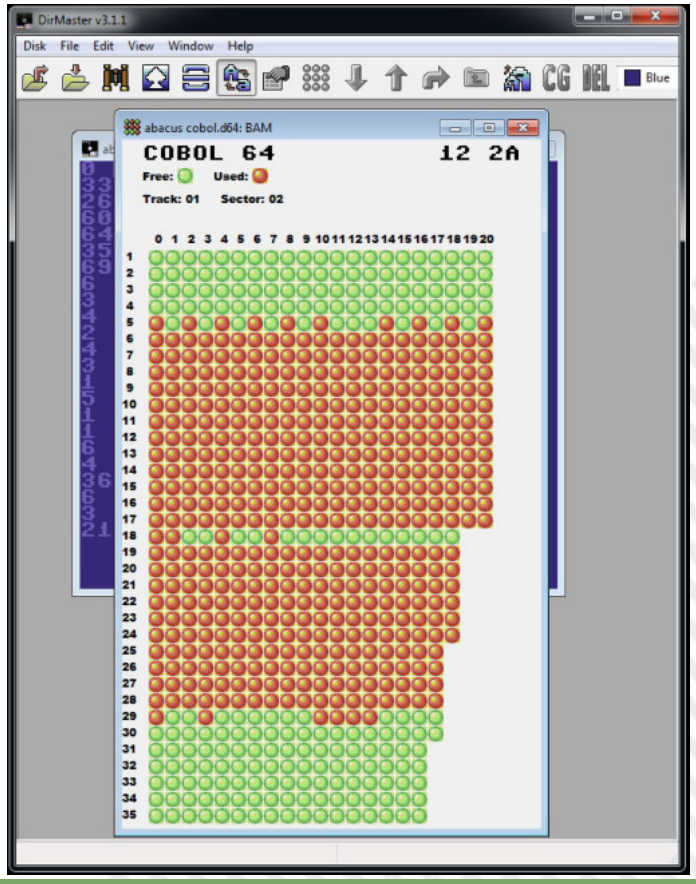
Il byte 02

Carichiamo un disco su DirMaster tramite Disk -> Open e poi selezionando il disco dal file system, oppure trascinando il disco prescelto (drag & drop) dentro la finestra del programma.

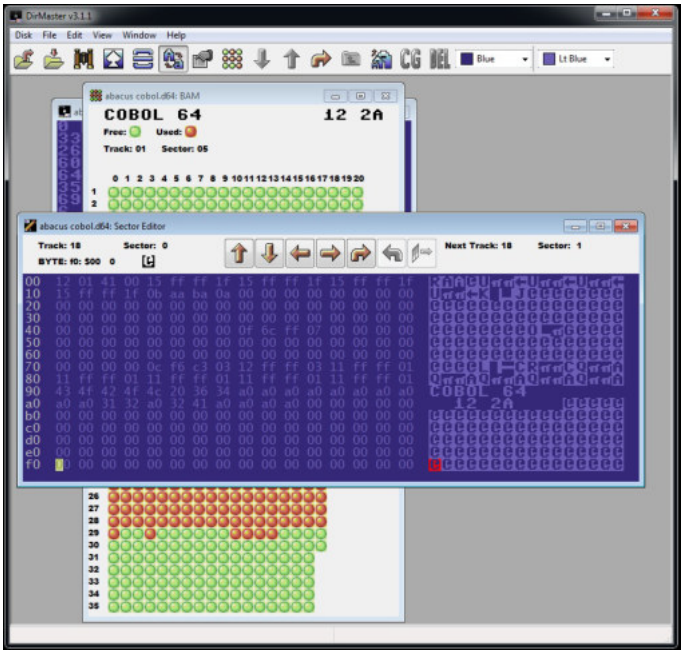
Una volta che il disco è stato letto e riconosciuto da DirMaster dovreste trovarvi di fronte a qualcosa di simile.



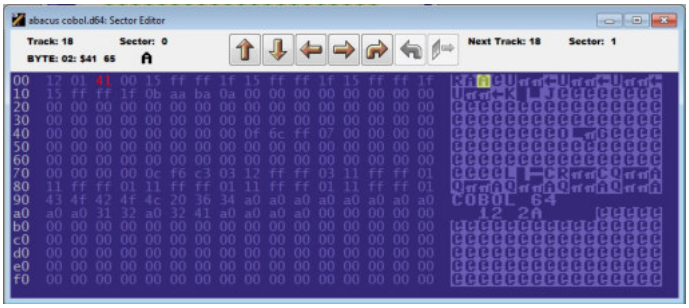
Adesso dal menù View, selezionate la voce BAM editor e dovreste vedere.



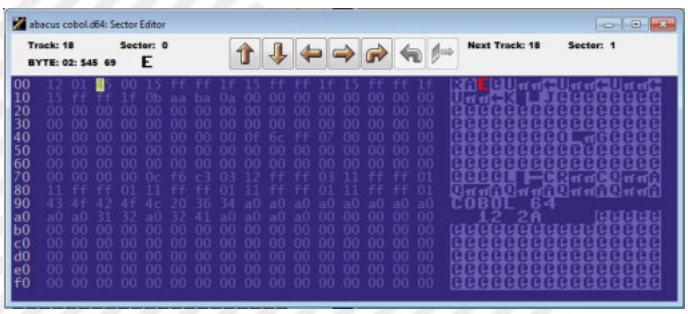
Come ormai sappiamo, la BAM si trova nella traccia 18 al settore 0; il valore che vogliamo modificare è il terzo byte della BAM, quindi fate click sul pallino immediatamente adiacente al numero 18. Dovreste accedere al Sector Editor, come si vede.



Se osserviamo attentamente i primi 3 byte della BAM, noteremo che il terzo byte (byte 02) contiene il valore hex 41 corrispondente alla lettera 'A'.



Questo byte, apparentemente senza significato, è invece estremamente importante per il DOS (Disk Operating System). Il valore del terzo byte deve essere per forza una 'A' oppure un errore 73, DOS MISMATCH,18, 00 verrà generato dal DOS. Questo byte viene infatti controllato ogni volta che un disco viene inserito nel disk drive.
Se questo byte viene invece modificato con il valore 'E' hex 45, il disco diventerà permanentemente write protected. Il valore 'E' infatti farà credere al DOS che il disco sia stato formattato con un drive non compatibile e renderà di fatto il disco impossibile da scrivere fino ad una nuova formattazione.
Proviamo a modificare il valore cliccando sul terzo byte e scrivendo il valore 45 al posto di 41.



Dopodichè dobbiamo salvare la modifica tramite Disk -> Save e proveremo ad utilizzare il disco con il VICE.
Una volta inserito il disco in VICE e listato la directory tutto ci sembrerà normale. Ma provando ad usare uno dei programmi od a salvare un banale listato basic sul disco.

```
3     "C2BUILD-DATA1"     PRG
1     "DATA1"             SEQ
5     "C2TEST"            PRG
1     "C1PROVA"           PRG
1     "C2PROVA"           PRG
6     "C1ARCHIVIO"        PRG
4     "C2ARCHIVIO"        PRG
36    "C1RELATIVE"        PRG
6     "C2RELATIVE"        PRG
3     "RELDATA1"          REL
209 BLOCKS FREE.
READY.
NEW

READY.
10 PRINT "PIPPO"
LIST

10 PRINT "PIPPO"
READY.
SAVE"PIPPO",8

SAVING PIPPO
READY.
```

Ci accorgeremo che il nostro file non verrà memorizzato...

```
0 "COBOL 64          " 12 2A
33    "C1TEST"            PRG
26    "COBOL 64"          PRG
60    "C0EDIT"            PRG
64    "COSYN"             PRG
35    "COSYNP"            PRG
69    "CORD"              PRG
6     "C1ADDING"          PRG
3     "C2ADDING"          PRG
4     "C1LIST-DATA1"      PRG
2     "C2LIST-DATA1"      PRG
4     "C1BUILD-DATA1"     PRG
3     "C2BUILD-DATA1"     PRG
1     "DATA1"             SEQ
5     "C2TEST"            PRG
1     "C1PROVA"           PRG
1     "C2PROVA"           PRG
6     "C1ARCHIVIO"        PRG
4     "C2ARCHIVIO"        PRG
36    "C1RELATIVE"        PRG
6     "C2RELATIVE"        PRG
3     "RELDATA1"          REL
208 BLOCKS FREE.
READY.
```

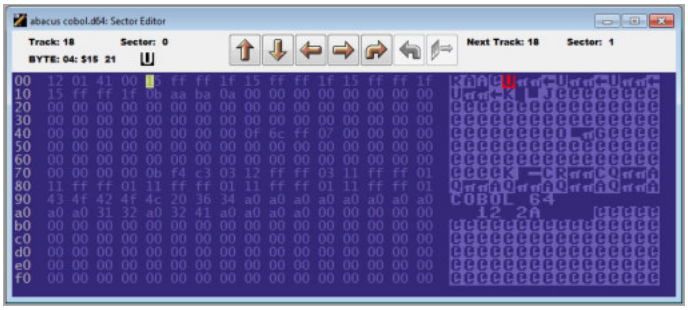
Quello che invece varierà è la dimensione dei blocchi disponibili (2 blocchi perche' ho fatto 2 prove), che però tornerà nuovamente al valore iniziale una volta rimosso il disco e riconnesso in VICE (provare per credere).
A scanso di equivoci ho modificato nuovamente il valore del terzo byte a hex 41 - 'A' e ripetuto ancora una volta la prova di salvataggio del file basic; questa volta il tutto ha avuto l'esito atteso.

```
33    "C1TEST"            PRG
26    "COBOL 64"          PRG
60    "C0EDIT"            PRG
64    "COSYN"             PRG
35    "COSYNP"            PRG
69    "CORD"              PRG
6     "C1ADDING"          PRG
3     "C2ADDING"          PRG
4     "C1LIST-DATA1"      PRG
2     "C2LIST-DATA1"      PRG
4     "C1BUILD-DATA1"     PRG
3     "C2BUILD-DATA1"     PRG
1     "DATA1"             SEQ
5     "C2TEST"            PRG
1     "C1PROVA"           PRG
1     "C2PROVA"           PRG
6     "C1ARCHIVIO"        PRG
4     "C2ARCHIVIO"        PRG
36    "C1RELATIVE"        PRG
6     "C2RELATIVE"        PRG
3     "RELDATA1"          REL
1     "PIPPO"             PRG
209 BLOCKS FREE.
READY.
```

Cambiando il dato del terzo byte ad altri valori, potremmo rendere il disco impossibile da leggere, da scrivere o entrambi i casi...
Questi valori però non sono documentati.

Il byte 04

Il quinto byte della BAM contiene il numero dei blocchi disponibili sulla traccia 1, in questo caso 21 (hex 15).



Modificando questo valore a hex FF (255), faremo credere al DOS che la traccia 1 abbia 255 blocchi liberi. Nel nostro caso il disco precedentemente aveva 209 blocchi liberi, dopo la modifica ne aggiungerà ben 234 (255 - 21) visualizzando quindi un totale di 443 (209 + 234) blocchi disponibili.

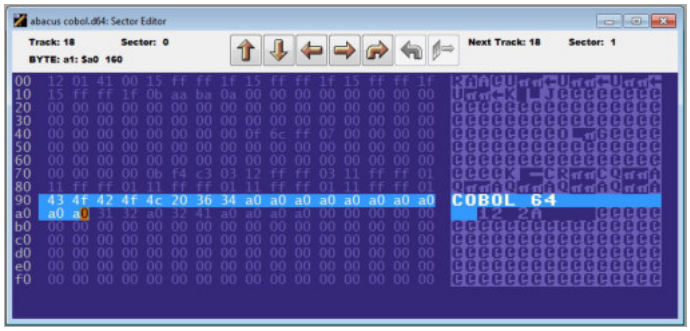
Modificando convenientemente questo valore per tutte le altre tracce del disco faremo credere al DOS di avere un superdisco!

Byte da 5 ad 7

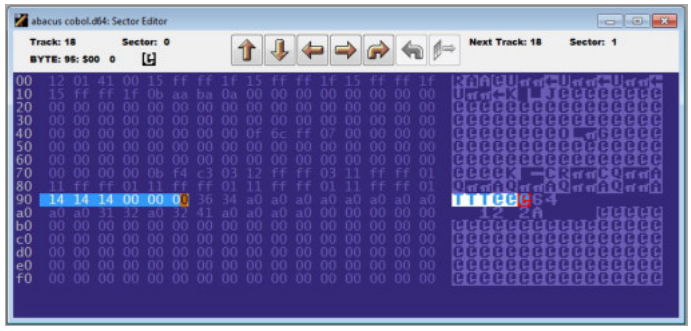
Come abbiamo visto nell'articolo precedente, i byte da 5 a 7 mostrano la rappresentazione "grafica" dei settori liberi o occupati in ogni traccia. Alterando questo valori il DOS non sarà più in grado di riconoscere quali settori sono occupati sul disco. Fortunatamente sarà possibile ripristinare i valori corretti procedendo ad una validazione del disco con il comando VALIDATING.

Byte da 144 a 161

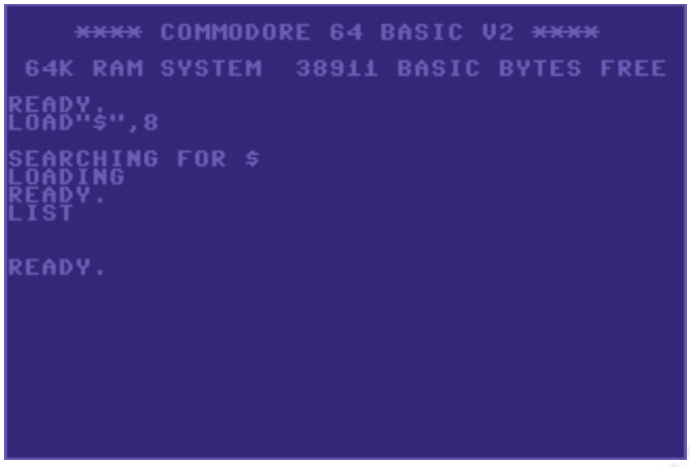
Anche questi byte dovrebbero esserci ormai familiari dall'articolo precedente, in quanto rappresentano il nome del disco.



Proviamo però a cambiare i primi sei byte del nome del disco con i valori: **14 14 14 00 00 00**



Otteremo come risultato che la directory del disco non potrà essere visualizzata!



Byte da 167 a 255

Questi byte sono liberi e la loro modifica non comporta nessun problema al DOS. Per questo motivo venivano utilizzati per memorizzare messaggi per quelli che provavano a piratare il software.

Alcune delle tecniche qui descritte venivano utilizzate per proteggere i dati nei dischi dalla copia. Ovviamente le tecniche appena mostrate sono molto blande e facilmente aggirabili, ma prossimamente vedremo come utilizzare i settori danneggiati dei dischi per prevenire le copie pirata. Appuntamento quindi ad uno dei prossimi articoli sul formato D64.

Leggiamo i file .WAV sullo Z80NE

di Antonino Porcino

Vi presento un'utility che ho scritto per convertire i file .WAV per il computer Z80 N.E. (Nuova Elettronica). Sarà un'ottima occasione per parlare di questo particolare computer italiano, ma anche di interfacce a cassette e di tecniche di elaborazione digitale dei segnali (DSP).

Lo Z80NE è un computer in kit di montaggio basato sul processore Z80 apparso sulla rivista Nuova Elettronica dal 1979 al 1985. Nel corso degli anni sulla rivista sono state pubblicate svariate periferiche per questo computer che gli appassionati si sono cimentati ad assemblare e fare funzionare; tra queste l'interfaccia per il registratore a cassette LX.385 trattata in questo articolo.

Tutti i dettagli sullo Z80NE li potete trovare online sul sito dedicato www.z80ne.com e sul gruppo Facebook "The Home of Z80 NE" dove si riuniscono gli estimatori di questo vintage. Inoltre, per chi ci si volesse cimentare, è possibile far girare il software dello Z80NE tramite il noto emulatore MESS (oggi MAME).

Il programma che ho scritto è una semplice utility da lanciare dal prompt dei comandi per convertire un file audio .WAV registrato sullo Z80NE in un file binario per PC. Per rendere l'utility completa, ho anche scritto l'operazione inversa che dato un file binario sul PC lo trasforma in un file .WAV, che sarà poi possibile caricare sullo Z80NE o sull'emulatore.

L'utility è open source e si trova all'indirizzo https://github.com/nippur72/z80ne-wav dove un README spiega come installarla ed utilizzarla. È scritta in JavaScript (node.js) ed è molto semplice e leggibile se volete dare un'occhiata al sorgente. Gira su Windows, Linux e Mac.

L'idea di scrivere l'utility è nata da una post sul gruppo Facebook riguardo la lettura dei file .WAV di questa macchina; in quel periodo stavo scrivendo una routine di turbo tape per un altro retro-computer (di cui vi parlerò in un altro articolo) per cui mi incuriosì quale formato nello specifico utilizzasse lo Z80NE.

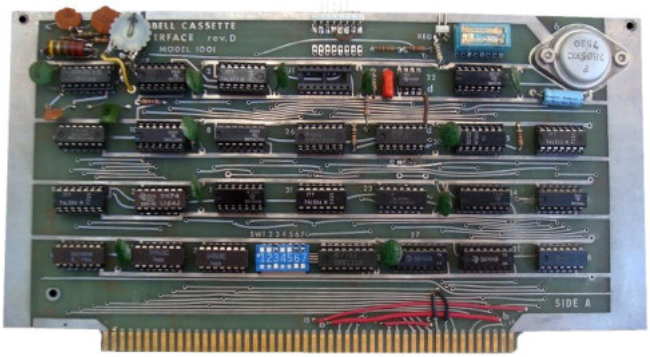
Leggendo la scheda tecnica del computer, appresi che la sua l'interfaccia a cassette LX.385 implementa il noto protocollo Kansas City Standard. La leggenda narra che nel lontano 1976, a seguito del proliferare dei formati per registratore a cassette, l'autorevole rivista americana Byte abbia sponsorizzato un simposio tenutosi nella città di Kansas City per definire uno standard che mettesse d'accordo i vari produttori.

Facciamo un tuffo nel passato, siamo a metà anni '70, un periodo di grandi cambiamento nel settore informatico. Non solo si stava assistendo al passaggio dal minicomputer al microcomputer, cioè ad un computer con la CPU tutta contenuta in un singolo chip, ma stava avvenendo un'altra transizione altrettanto importante: il passaggio dal nastro perforato al nastro magnetico. Infatti sino ad allora il nastro perforato era stato tra i metodi più usati per la memorizzazione permanente dei dati; fece la sua comparsa molto prima del computer con le telescriventi, le quali erano in grado di leggere e scrivere su nastro perforato i messaggi da trasmettere senza doverli ridigitare ogni volta. Le telescriventi a loro volta vennero poi collegate ai computer come principale periferica di I/O, e così il nastro perforato divenne automaticamente il supporto naturale per memorizzare i dati anche per i computer.

Il nastro magnetico rappresentava l'equivalente elettronico del nastro perforato di cui ne era l'evoluzione: più veloce, più pratico ed economico vista la diffusione sempre maggiore di audiocassette e registratori. Il problema era però che non c'era uno standard condiviso e non era possibile leggere un nastro di dati creato su un computer in un altro. Il Kansas City Standard tentò di porre rimedio a tutto ciò, senza però ahimè riuscirci in pieno.

Il KCS si diffuse maggiormente tra i microcomputer, dove l'interfaccia a cassetta consisteva in una scheda dedicata da inserire comodamente nel bus S-100.

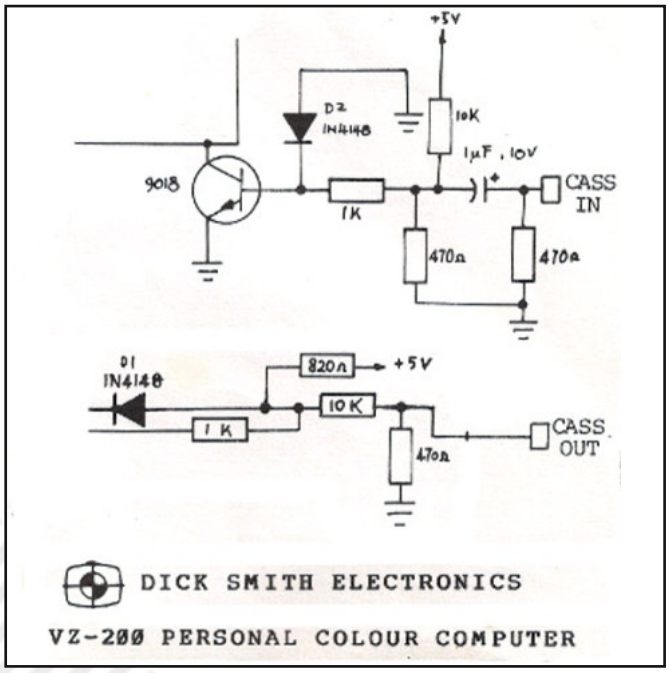
Diffusissima divenne l'interfaccia della Tarbell che implementava oltre al suo formato, anche il KCS.



La scheda Tarbell Tape Interface per bus S-100 (1976)

Quando successivamente i microcomputer cedettero il passo agli home, il discorso fu diverso: i produttori non volevano complesse interfacce dedicate, volevano piuttosto ridurre al minimo il costo di produzione del computer. Infatti in praticamente tutti gli home l'interfaccia a cassette si ridusse poi in un semplice circuito di pochi transistor.

In interfacce semplificate di questo tipo, l'audio del nastro viene convertito ad 1 bit e tutte le operazioni di codifica e decodifica sono fatte direttamente dalla CPU piuttosto che dall'interfaccia.



Esempio di interfaccia a cassette semplificata degli home computer: un transistor, due diodi e qualche altro componente (Dick Smith VZ200 / Laser 210).

Dovendo quindi utilizzare un circuito proprietario, i produttori degli home misero in dubbio l'efficienza del formato KCS: si resero conto infatti che questo ha alcune ridondanze che lo rendono sì più robusto ma solo a patto di utilizzare una costosa circuiteria; ma invece con la circuiteria semplificata i vantaggi si perdono e il formato risulta molto più lento di quanto sia effettivamente possibile. Motivo per il quale solo alcuni home lo implementarono, più che altro al solo scopo di poter vantare la compatibilità lo standard.

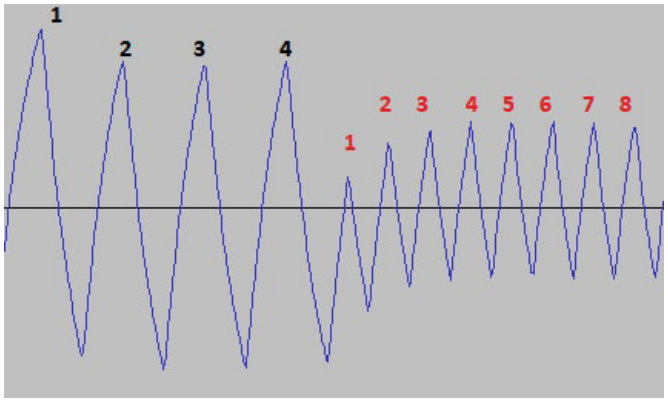
Potete leggere la storia completa del Kansas City Standard su Wikipedia https://it.wikipedia.org/wiki/Standard_Kansas_City dalla quale si apprende fra l'altro che anche il giovane Bill Gates vi partecipò in qualità di dipendente del MITS (Micro Instrumentation and Telemetry Systems).

Il Formato KCS

Ma vediamo i dettagli di questo famoso Kansas City Standard.

Si tratta di un protocollo ispirato alla tecnologia dell'epoca, infatti è molto simile a quello utilizzato da una radio-telescrivente o da uno dei primi modem FSK (ad esempio lo standard Bell 101). È un protocollo asincrono orientato ai caratteri piuttosto che ai file: ogni byte viene trasmesso separatamente come in una telescrivente, mettendo in conto anche eventuali pause tra un carattere ed il successivo (cosa che avveniva appunto nelle telescriventi durante la digitazione del messaggio). Anche il controllo è a livello del byte mediante bit di parità, mentre non sono previsti inizio/fine/nome del file o controlli tipo checksum (che devono essere implementati separatamente a livello di applicazione).

Il Kansas City Standard stabilisce che il bit "zero" sia trasmesso come 4 cicli di un tono a 1200 Hz, mentre il bit "uno" come 8 cicli di un tono a frequenza doppia, ossia 2400 Hz. Si tratta quindi di una modulazione FSK (frequency shifting key), ossia una modulazione di frequenza dove la frequenza è fatta variare esclusivamente tra due toni (1200 e 2400 Hz), in gergo chiamati mark e space.

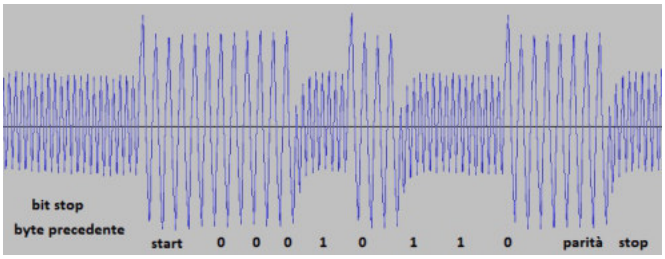


Il segnale audio di un bit "0" seguito da un bit "1" - a causa della particolare circuiteria adottata nella scheda LX.385 dello Z80NE, il segnale non è un'onda sinusoidale e neanche un'onda quadra, ma un'onda "quasi" triangolare.

I bit sono poi trasmessi in sequenza per formare un byte di 8 bit a cui viene aggiunto un bit di parità a scopo di controllo per poter capire se il byte sia stato ricevuto correttamente.

Il byte così composto è poi racchiuso tra un bit start sempre posto a "0" e due bit stop sempre posti ad "1". Il loro scopo è quello di fare in modo che successivamente sia possibile rilevare la transizione da stop a start, cioè tra la fine del byte precedente e l'inizio del byte successivo. Questo fa sì che il ricevitore sia in grado di sincronizzarsi col byte in arrivo anche se questo arriva in un momento leggermente diverso da quello previsto.

Una volta individuato l'inizio di un byte, è sufficiente leggere i bit di cui esso è composto ad intervalli regolari, poiché la loro temporizzazione è ben definita.



Un byte completo

Ma perché i bit di stop sono due anziché uno? Ciò si giustifica col fatto che nelle telescriventi, una volta che il carattere veniva ricevuto questo doveva essere stampato, e dunque il circuito ricevente non poteva essere disponibile durante la fase di stampa; questo perlomeno all'epoca delle telescriventi elettromeccaniche che non avevano un buffer -- una durata maggiore dello stop bit serviva a compensare il tempo usato per stampare il carattere ricevuto.

Riassumendo quindi il formato Kansas prevede:
n.1 bit start (sempre a "0")
n.8 bit di dati, ossia il byte da trasmettere, con il bit meno significativo trasmesso per primo
n.1 bit di parità (pari a "1" se i bit di dati a "1" sono in numero dispari, "0" altrimenti)
n.2 bit di stop (sempre a "1")

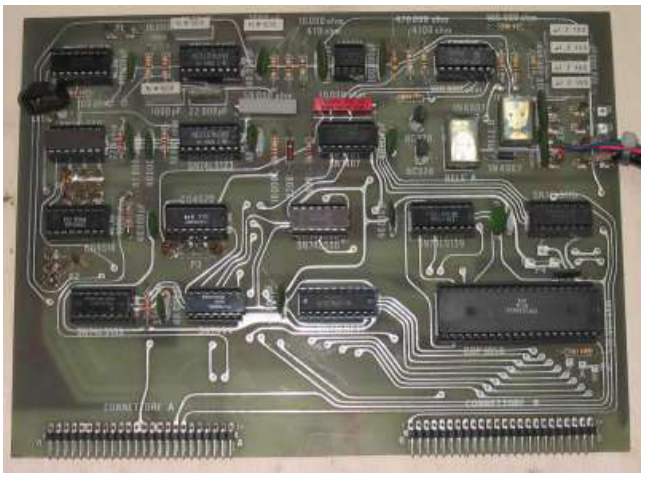
bit "0" trasmesso come 4 cicli di un tono a 1200 Hz.
bit "1" trasmesso come 8 cicli di un tono a 2400 Hz.
sincronizzazione del clock alla transizione da stop => start

Il KCS sullo Z80NE

Come nei microcomputer, lo Z80NE adotta una soluzione su scheda separata, questa però non è la classica interfaccia per bus S-100 poiché il computer dispone invece di un bus proprietario a 48 pin.

La scheda è denominata LX.385 ed è stata pubblicata nei numeri 70 e 71 della rivista Nuova Elettronica che potete leggere online. Nella rivista fra l'altro si trova una più o meno plausibile quanto dettagliata motivazione della scelta del KCS rispetto ad altri formati.
È possibile collegare l'interfaccia LX.385 a ben due registratori, così come è possibile scegliere il baudrate tra quelli previsti (300, 600 e 1200 baud). Per semplicità in questo articolo farò sempre riferimento alla velocità di base di 300 baud, poiché le altre sono ottenibili semplicemente moltiplicando per 2 o per 4 i valori.

Il cuore di decodifica dell'interfaccia è rappresentato da un circuito bistabile che è in grado di contare i passaggi per lo zero del segnale audio; dal numero di passaggi si ricava poi se trattasi di un bit "0" (4 passaggi) o "1" (8 passaggi).



L'interfaccia LX.385 registratori a cassetta per Z80NE

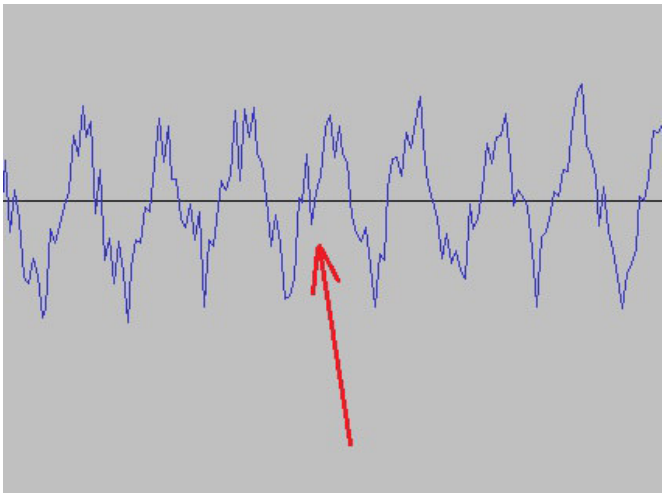
Decodifica della FSK

Come accennato, la demodulazione dei toni di un segnale FSK avviene semplicemente contando il numero di passaggi dell'onda per lo zero. Questa è una tecnica efficace ed economica che però presenta il limite che il segnale deve essere piuttosto pulito ed esente da rumore. In uno scenario di un registratore a cassetta questo può andar bene, poiché il segnale è l'unico ad essere presente sul nastro (oltre magari ad un po' di rumore statico). Se invece il segnale viene trasmesso attraverso un mezzo rumoroso, ad esempio un canale radio, tale decodifica mostra subito i suoi limiti: basta infatti una piccola interferenza per generare dei falsi positivi che inducono in errore l'algoritmo di decodifica.

Mediante tecniche di DSP, cioè di elaborazione digitale dei segnali, possiamo porre rimedio e spingere la decodifica anche fino a livelli di rumore molto elevati. Nelle mie non rigorose prove sono riuscito a leggere trasmissioni con livelli di S/N fino di circa 3dB, quando col metodo tradizionale sono richiesti livelli molto più elevati (sui 30 dB).

Ma in cosa consiste il DSP (digital signal processing)? Come suggerisce il nome si tratta di elaborare il segnale nella sua forma digitalizzata, ossia

campionata. I livelli di tensione audio diventano così una serie ben definita di numeri che rappresentano l'ampiezza del segnale audio originale. Con il DSP si applicano delle trasformazioni puramente matematiche derivate dalla teoria dei segnali.



Nella foto: il rumore causa un falso passaggio per lo zero dell'onda.

Tutto quello che facciamo con il DSP potremmo a rigore farlo in analogico, costruendo cioè dei circuiti con condensatori, resistenze ed induttori. Ed in realtà questo è quello che radio, TV e tutte le altre apparecchiature elettroniche facevano fino agli anni '80. Ma con l'avvento del digitale e l'aumentare della potenza di calcolo, è diventato più conveniente effettuare la controparte digitale mediante dei semplici calcoli matematici, che in buona sostanza si riducono ad addizioni e moltiplicazioni.

Senza scomodare le GPU moderne, adesso anche i microprocessori meno potenti sono in grado di effettuare un numero elevato di moltiplicazioni al secondo rendendo percorribile la strada del DSP per applicazioni che non siano molto esigenti. I processori moderni poi sono addirittura in grado di effettuare i calcoli direttamente in virgola mobile, evitando scomode conversioni da e verso i numeri interi cui facevano ricorso i primi processori DSP.

Guardando la figura sopra riportata, ci si rende conto come i bit pur se codificati come cicli di onde sinusoidali sono facilmente distinguibili anche ad occhio nudo: le onde "strette" sono quelle a frequenza più alta (2400 Hz), rappresentano quindi il bit 1; quelle più larghe il bit 0 (tono a 1200 Hz). Poiché vi è una certa separazione tra i due toni (di 1200 Hz), possiamo dividere i segnali dei due bit applicando due filtri passa banda.

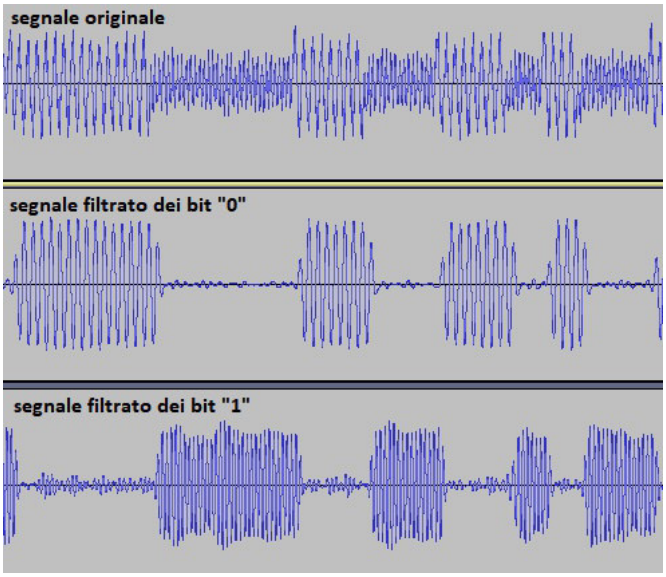
Un filtro passa banda è un filtro che lascia passare più o meno intatta la frequenza desiderata, annullando tutte le altre (ne abbassa il volume cioè). Un filtro digitale esegue il filtraggio semplicemente moltiplicando i campioni audio per i coefficienti numerici di cui è composto il filtro. Tali coefficienti numerici vengono preventivamente calcolati al momento della creazione del filtro, in base a degli algoritmi ricavati dalla teoria dei segnali. Come per magia, un filtro si riduce ad una serie, seppure lunga di semplici moltiplicazioni.

Tutto ovviamente avviene numericamente, e la qualità del risultato dipende dalle dimensioni del filtro digitale. Dimensioni che sono un compromesso tra velocità di risposta del filtro (filtri piccoli) e capacità di selezionare le frequenze con precisione chirurgica (filtri grandi). Nel programma utilizzo il filtro digitale di tipo FIR (filtro a risposta impulsiva infinita), impostando, a seguito di varie prove, a circa 80 coefficienti la grandezza del filtro.

In JavaScript esiste un'ottima quanto sconosciuta libreria chiamata "fili" che permette non solo di calcolare i coefficienti del filtro partendo dai suoi parametri, ma consente anche di "eseguire" il filtro su un vettore di campioni audio, restituendone l'audio filtrato. In pratica la libreria fa tutto il lavoro per noi sgravandoci dai complicati dettagli e dall'enorme teoria matematica che c'è dietro – molto comodamente per noi il filtro è una "black box".

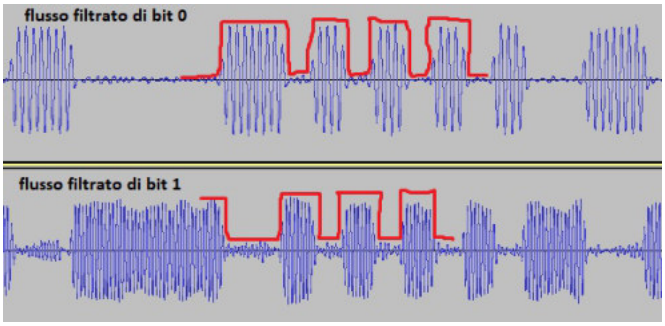
Creiamo quindi due filtri, uno a 1200 Hz per i bit "0" e uno a 2400 Hz per i bit "1". Imposto come larghezza di banda circa 2/3 del valore della frequenza centrale del filtro, ad esempio il filtro a 1200 Hz farà passare le frequenze da 1000 a 1400 Hz, in modo che ci sia una certa tolleranza nel caso la velocità di riproduzione del nastro non fosse precisa.

Dopo aver creato i due filtri, li applico sul segnale in ingresso: ottengo due nuovi segnali distinti, come mostrato in figura:



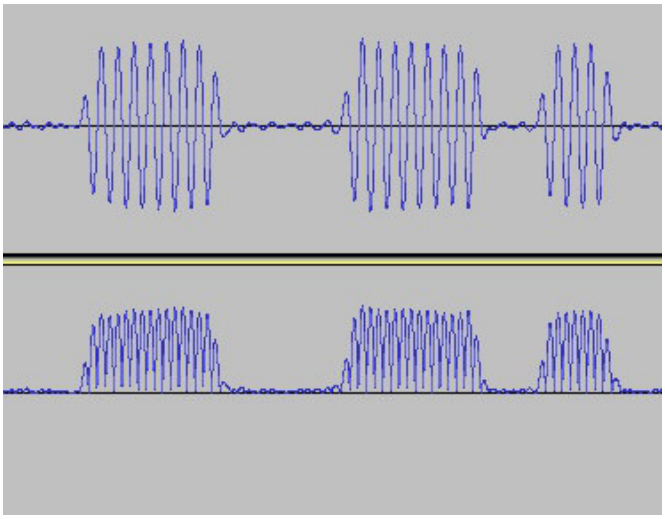
Nella figura ho utilizzato un segnale originale rumoroso in modo da mettere in risalto le capacità del sistema. Come si vede, i due filtri sono riusciti a

differenziare abbastanza bene i bit, risultando in due segnali pressoché disgiunti e sovrapponibili. Notate come quando è presente un certo bit, il suo complementare è a volume azzerato e viceversa.
Un'altra cosa che risalta all'attento osservatore, è che il segnale dei bit che stiamo cercando di estrarre è costituito dal "profilo" esterno delle due onde sinusoidali, che nella figura ho evidenziato in rosso:



Questo profilo infatti non è altro che il segnale a 300 bps dei bit "modulati": 1200 Hz diviso 4 cicli dà proprio 300 Hz (ossia 300 bps).

Ma come estrarre questo profilo? Innanzitutto nell'onda sinusoidale ci sarebbero in realtà due profili: quello sull'asse positivo e quello sull'asse negativo. Ce ne sbarazziamo di uno con un'operazione matematica, semplicemente invertendo il segno dell'onda quando questa è negativa. Si ottiene una cosa del tipo:

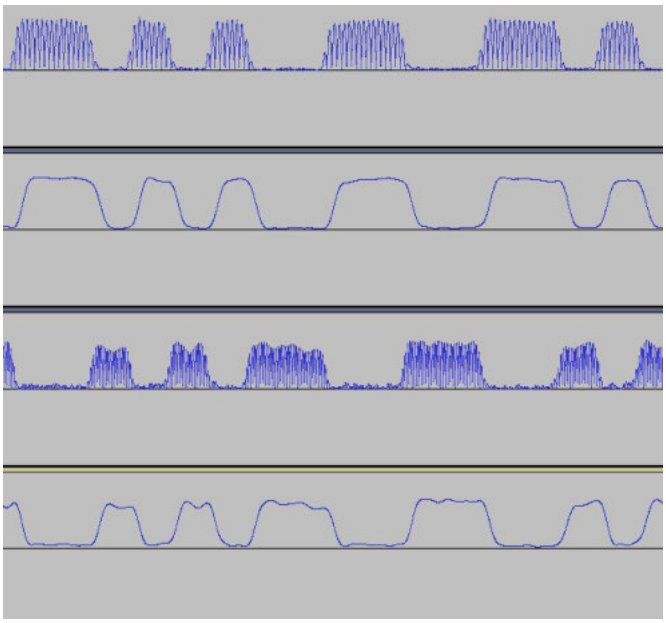


Gli appassionati di elettronica avranno notato la similarità con quanto si fa con gli alimentatori DC quando si deve rettificare la tensione AC con un ponte a diodi. Qui è lo stesso solo che lo facciamo senza usare i componenti elettronici, lo facciamo via software!

Riassumendo quindi: il segnale rettificato contiene dentro di se due onde distinte: un'onda ad alta frequenza (1200 Hz) che ormai non ci interessa più tanto, e l'onda del "profilo", a più a bassa frequenza (300 Hz) che è quella di nostro interesse.

Come facciamo ad estrarla? Ma con un altro filtro ovviamente!

Questa volta usiamo un filtro passa basso, che come dice lo stesso nome fa passare solo le frequenze al di sotto di un certo valore. Con la libreria creiamo quindi un filtro FIR passa basso a 300 Hz con 64 coefficienti il cui numero è stato stimato come dopo aver fatto vari tentativi.

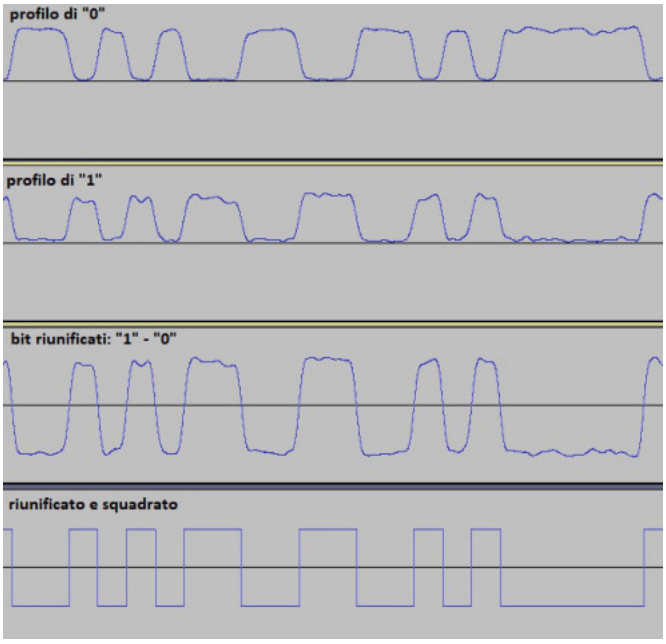


L'operazione filtraggio del profilo va svolta su entrambi i segnali rettificati, quello del bit "0" e quello del bit "1". Si ottengono due nuovi segnali di "profilo" che per comodità riunifichiamo in un unico segnale, semplicemente cambiando il segno al profilo del bit "0" e sommandolo al profilo di "1".

Infatti come detto sopra i due segnali sono complementari, quindi riunificandoli non si perde niente. In questo modo il segnale riunificato sarà "alto" in corrispondenza del bit "1" e "basso" in corrispondenza di "zero", che è molto più facile ed intuitivo da processare.

Facciamo inoltre un'ultima operazione finale sul segnale "riunificato": poiché a causa del rumore questo ha delle piccole variazioni di ampiezza che non ci interessano, lo rendiamo un'onda quadra perfetta, semplicemente guardando il segno "+" o "-" del segnale; in pratica lo quantizziamo a due livelli: se positivo diventa +1, se negativo diventa -1.

Come vediamo nell'immagine successiva, il segnale dei bit riunificato e squadrato adesso è bello pulito anche se il segnale originale conteneva dei rumori. È il segnale che sarà la base per la successiva decodifica.
In questo segnale, se facciamo un calcolo, otteniamo che un singolo bit è lungo 147 campioni audio (ottenuto come 44100 Hz di sample rate diviso per 300 bps). Questo vale per entrambi i bit "0" e "1" poiché come detto prima, la codifica Kansas è stata fatta in modo da avere la stessa lunghezza/durata per i singoli bit.



Passare dal segnale grezzo ai byte adesso è facile. Attenendoci alla codifica Kansas, dobbiamo per prima cosa individuare il passaggio dallo "stop bit" del byte precedente (o dell'inizio della registrazione se siamo all'inizio) allo "start bit" del byte successivo (o del primo byte nel file se siamo all'inizio). Lo facciamo semplicemente guardando i campioni ad uno ad uno, quando il precedente è +1 e il successivo è -1 allora avremo identificato una transizione, ossia l'inizio di un byte.

Immaginiamo di avere una testina di lettura virtuale (via software non è altro che un indice dentro l'array che contiene il segnale): adesso siamo con la testina nella zona di transizione del segnale da "alto" a "basso"; ci spostiamo quindi in avanti di 147 campioni saltando cosi il bit di "start", che non ci serve più, e ci spostiamo ancora in avanti di mezzo bit (73.5 campioni) per pescare il centro del primo bit.

Ci posizioniamo sul centro perché è la zona migliore dove leggere il valore. Se il segnale è troppo rumoroso, il bit considerato potrebbe essere un po' più lungo o più corto di 147 (pensate ad esempio ad un registratore non tarato che riproduce il nastro a velocità più elevata: i bit per forza di cose saranno più corti). Il centro è dunque la posizione più sicura. Da qui in poi gli altri bit si trovano andando avanti sempre di 147 campioni: prima i data bit, poi il bit di parità.

Arrivati ai due stop bit, li saltiamo andando avanti di 147 x 2. Basta ora solo ripetere il ciclo di lettura dall'inizio il quale tiene conto del fatto che il segnale di stop possa durare anche più di due bit (vista la natura asincrona del protocollo).

Una volta collezionati tutti i bytes (la fine è indicata con un segnale di stop di 15 secondi), basta scriverli in un file su disco ed il gioco è fatto!

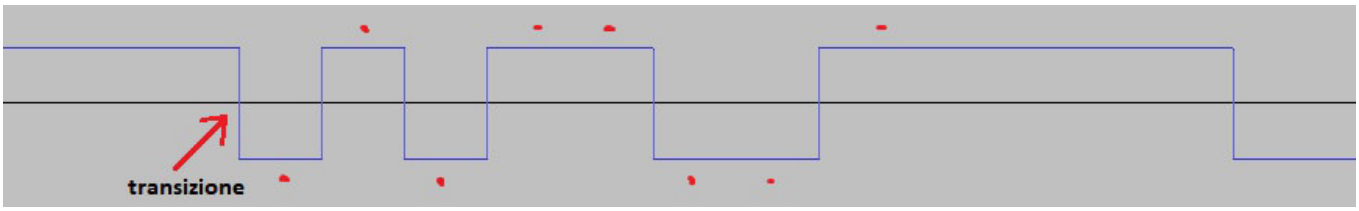


Immagine da https://www.1000bit.it

Cenni di GAME CODING

di Gianluca Girelli

Quando nel 1982 al compianto Jay Miner fu affidato il compito di sviluppare un nuovo computer, egli non sapeva che avrebbe contribuito a cambiare per sempre il mondo dell'informatica.
Incaricato da una cordata di dentisti di creare una semplice piattaforma di gioco da lanciare nel redditizio mondo dei videogames, ma fortunatamente dotato di un intuito e di una sensibilita' superiori, Miner seguì tutt'altra strada. Invece di disegnare un sistema chiuso, come tradizionalmente sono le console, partorì, con l'aiuto di molti altri valenti collaboratori, un sistema potente e liberamente espandibile che ancora oggi, dopo oltre trentanni, ci regala grandi soddisfazioni.

In qualche modo fedeli a questo principio, cioè a quello del connubio continuo tra il mondo del "gaming" e quello del "computing", si è quindi pensato di scrivere un articolo sul "game coding" (o "game programming" che dir si voglia) poichè, a parere dello scrivente, esso rappresenta la massima espressione dell'arte creativa, non solo in senso informatico [BOX].

All'interno di un gioco, infatti, possiamo trovare: grafica (2D, 3D o mista), suono (semplici effetti o parlato doppiato, localizzato e sicronizzato), intelligenza artificiale (più o meno "scriptata"), algoritmi di archiviazione, algoritmi di compressione dati, "middleware" di progettazione personaggi o livelli, player video per rivedere indipendentemente i filmati di intermezzo una volta sbloccati, player audio per la colonna sonora etc... Inoltre, se un gioco è stato sviluppato per il multiplayer online, è necessario anche programmare tutta la parte di gestione client/server, con particolare attenzione ai tempi di latenza in rete (per evidenti ragioni di sincronia gioco/giocatori) e magari anche al VoIP per la comunicazione continua tra giocatori impegnati in campagne in modalità cooperativa.

Chiaramente, non tutto il software videoludico è così complesso, ma molti di questi principi possono essere trovati in ogni gioco, anche in quello apparentemente piu' "banale". Le defunte avventure testuali, ad esempio, avevano bisogno per funzionare di un buon analizzatore sintattico mentre le avventure grafiche, eredi dirette di quelle testuali, necessitavano di una interfaccia utente potente ma allo stesso tempo semplice ed intuitiva, per evitare che il povero giocatore finisse bloccato e senza opzioni sin dalla prima "stanza" di gioco. Inoltre entrambi i tipi di gioco, cosi' come la loro successiva evoluzione (le avventure grafiche 3D), hanno bisogno di una adeguata struttura dati che possa riprodurre all'interno della memoria del nostro sistema computerizzato la complessità di un mondo di gioco tridimensionale ed il collegamento/interazione tra i vari ambienti che virtualmente lo compongono, nonche' con le "creature" che in esso vivono.

Per quanto sin qui detto, l'idea che sta dietro questo articolo è quella di dare dei cenni di teoria della programmazione dei giochi, che possano fornire al lettore una base fondante comune da applicare poi eventualmente allo studio di tutorial specifici (in particolare quelli su ARexx e Hollywood). In questo modo, con puro aprroccio andragogico (vedi elenco risorse) alla formazione, sarà più semplice imparare divertendosi.

Un primo sguardo al problema

Il componente centrale di ogni gioco, dal punto di vista della programmazione, è il cosidetto "main loop". Questo pezzo di codice permette al gioco di funzionare dolcemente a prescindere dall'input dell'utente (o dalla mancanza di tale input). La maggior parte dei software tradizionali risponde infatti solo in caso di input, mentre nulla viene fatto se questo input non c'è. Ad esempio, un word processor giustifica parole e testo mentre l'utente le sta digitando. In assenza di battute, l'elaboratore di testi rimane semplicemente in attesa. Altre funzioni, ad esempio una deframmentazione dell'hard disk, pur necessitando di parecchio tempo per di essere completate, vengono sempre iniziate solo quando l'utente "dice" al programma di fare qualcosa, ancorche' in modo differito nel tempo.

I giochi, d'altro canto, devono continuare ad operare a prescindere dall'input ricevuto, e ciò che permette loro di farlo è appunto il "game loop" il quale, in modo altamente semplificato e riportato in pseudocodice, potrebbe assomigliare a qualcosa di questo tipo:

```
inizio loop   ;continua finche' l'utente non decide di
uscire
     controlla l'input dell'utente
     gestisci l'intelligenza artificiale
     muovi i nemici
     gestisci le collisioni
     disegna la grafica
     gestisci il sonoro
fine loop
```

Il loop può essere rifinito e modificato durante il processo di sviluppo del gioco, tuttavia la maggior parte dei giochi è basata su una struttura concettuale di questo tipo.
È interessante notare però che durante la progettazione di dettaglio, il "game loop" può cambiare profondamente a seconda della piattaforma per la quale viene sviluppato.

I vecchi giochi per Amiga, così come i contemporanei giochi per MS-DOS e la maggior parte di quelli scritti per console, possono sfruttare le risorse di sistema in modo esclusivo e senza restrizioni. Invece, i giochi scritti per i moderni sistemi devono "girare" all'interno del limiti posti dal "process scheduler", che potrebbe avere altri programmi in esecuzione allo stesso tempo. Inoltre, il diffondersi dell'uso dell'hyper-threading e del multicore, che permette di disaccoppiare alcuni processi permettendo al gioco di funzionare senza scatti, deve tenere in considerazione il maggiore overhead imposto alla CPU. Ad esempio, il calcolo dell'intelligenza artificiale dei personaggi può avvenire indipendentemente dalla creazione in memoria della grafica di gioco, ma i due algoritmi devono poi convergere efficacemente durante il rendering della scena su schermo.

Un esempio di sviluppo

Dopo aver visto quindi le basi del problema introduciamo di seguito un possibile modello di sviluppo, derivato dall'esperienza dell'autore nella programmazione (sia hobbistica che indie) di videogame.
Nonostante questa esperienza si sia consolidata recentemente su linguaggi di scripting come ARexx e Hollywood (derivato direttamente dal linguaggio LUA), si è cercato di rimanere sempre fedeli al paradigma della "programmazione strutturata" (vedi elenco risorse) che rappresenta una garanzia di maggiore leggibilità e manutenibilità del codice. Inoltre, in riferimento al linguaggio LUA e derivati (Hollywood), questo approccio permette anche una veloce transizione verso il paradigma della programmazione orientata agli oggetti.

Nell'approccio moderno alla programmazione ogni programma si divide idealmente in 3 sezioni:

1. Sezione di costruzione delle strutture dati
2. Sezione di dichiarazione dei sottoprogrammi (procedure e/o funzioni)
3. Codice principale

Sezione 1: costruzione delle strutture

Questa sezione serve per costruire tutte le strutture dati necessarie al nostro codice per poter lavorare. Le tecniche usate per implementare questa parte del programma dipendono dal particolare linguaggio di programmazione usato. Ad esempio, se stiamo lavorando con "Hollywood", in questa sezione troveremo dei "pre-processor commands", che direttamente precaricano in memoria la grafica che ci servira' per costruire ed animare gli sprite. In linguaggi come il "C", invece, dovremmo prima creare l'oggetto "sprite" mediante la combinazione dei vari "costruttori di tipo" che il "C" ci mette a disposizione.

Pensando quindi ad Hollywood (vedi relativi articoli in RetroMagazine 16 e prossimamente nei successivi...), questa sezione potrebbe assumere la forma seguente:

```
@Includes ....
@BRUSHES ....
@SPRITES ....
@MUSIC ....
@SAMPLES ...
Const ....
```

Usando il linguaggio "C" (o simili), questa sezione conterrebbe invece primariamente "includes" e "defines", come si evince dall'esempio seguente che riporta parte di un "header" usato per la costruzione di una classe "sprite" su PlayStation2:

```
#ifndef __CSPRITE_H__
#define __CSPRITE_H__

#include <sps2lib.h>
#include <sps2tags.h>
#include <sps2regstructs.h>
#include <assert.h>
#include "PS2Defines.h"
#include "AddRegisters.h"
#include <math.h>

class CSprite
{
public:
    CSprite( float xPos, float yPos, int xTexture, int
yTexture, int widthTexture, int heightTexture);
    ~CSprite();

    void CSprite::setPos(float xPos, float yPos);
    void CSprite::setAngle(float Angle);
    void CSprite::addAngle(float Angle);
    void CSprite::setPosAll(int xPos1, int yPos1, int
xPos2, int yPos2, int xPos3, int yPos3, int xPos4, int
yPos4);
    void CSprite::setTexturePosST(float s1, float s2, float
t1, float t2);
    void CSprite::setTexturePosUV(int u1, int u2, int v1,
int v2);
    void CSprite::movePos(float xPos, float yPos);
    ......
    ......

private:
    int m_width, m_height;
    int m_xTexture, m_yTexture;
    float m_xPos, m_yPos;
    int m_xPos1, m_yPos1;
    int m_xPos2, m_yPos2;
    int m_xPos3, m_yPos3;
    int m_xPos4, m_yPos4;
    .......
    .......
    int m_iAnimationDirectionTexture;
    int m_iAnimationFramesInAnimation;
    int m_iAnimationDirectionOfFrame;
};

#endif
```

Sezione 2: dichiarazione delle procedure/funzioni

In (quasi) tutti i linguaggi di programmazione, le subroutine devo essere esplicitamente dichiarate prima di poterle utilizzare. Per alcuni linguaggi che supportano il"proto typing", in questa sezione si può trovare l'intestazione della funzione/procedura,

mentre la subroutine vera e propria potrebbe essere spostata dopo la dichiarazione delle variabili in uso o addirittura in file esterni. Questo modo di procedere è coerente con i paradigmi della programmazione ad "oggetti" e di quella "modulare".
In ogni caso vale il discorso precedentemente fatto: seguire un modello standardizzato facilita la scrittura del codice, la sua revisione e la sua manutenzione soprattutto nell'ottica della scalabilità. Ecco un esempio per questa sezione:

```
Function p_InitGameVars() ; Prototipo. Il codice viene
dichiarato piu' tardi o esternalizzato in un modulo

Function p_SetUpGameGfx() ; Prototipo. Vedi routine
precedente

Function p_InitGame() ; Dichiarazione esplicita.
    Cls
    p_InitGameVars()
    p_SetUpGameGfx()
    DisplaySprite(7, X, Y)
    DisplaySprite(1, sub_x, sub_y, f)
    DisplaySprite(6, #CENTER, #CENTER)
    Wait(100)
    RemoveSprite(6)
EndFunction
```

Nell'esempio, questa parte del codice si occupa di dichiarare ed inizializzare le variabili di gioco, costruire la grafica di base e di visualizzare una prima schermata introduttiva. È da notare come nei linguaggi più "puri" (come ad esempio il Pascal, ma anche tutta la famiglia del "C" e derivati), la dichiarazione delle variabili globali debba essere fatta all'interno della Sezione 1, poiché ogni variabile dichiarata internamente ad una subroutine diventa automaticamente una variabile privata.

[N.d.R: la sintassi usata per l'esempio di questa sezione è mutuata da Hollywood, che in realtà non ha capacità di gestire i prototipi di funzione. Nel linguaggio "C" invece i prototipi di funzione sono riconoscibili dalla parola chiave "void" a prefisso.]

Sezione 3: codice principale

In questa sezione si trova il main loop. Esso potrebbe assumere la forma vista in apertura di articolo (game loop "singolo") oppure essere strutturato in più loop, interni ed esterni. Quello più esterno potrebbe essere il "Main Menu", mentre al suo interno potrebbero trovare posto più strutture, una delle quali è sicuramente il "game loop" vero e proprio. Ecco un esempio:

```
Function p_MainMenu()
    ......
    p_PrepareMenu()
    Repeat
        .......
        If IsKeyDown("RETURN")
        Switch y
            Case 0:
                p_Game() ;ATTENZIONE. GAME LOOP!
            Case 1:
                P_Credits() ;routine crediti
            Case 2:
                p_DisplayControls() ;routine opzioni
            Case 3:
                p_CleanUp() ;esce dal programma e libera le
risorse utilizzate
            End
        EndSwitch
    EndIf
    ......
    WaitTimer(1, 1000/50)
    Forever
    If IsMusicPlaying(1) Then StopMusic(1)
EndFunction
```

Come si vede, all'interno di questo loop infinito che costituisce il menu principale del gioco, convivono più sottoprogrammi che svolgono varie funzioni, dalla selezione delle opzioni di gioco fino all'algoritmo di "garbage collection" di chiusura.
A seconda della scelta effettuata sarà poi possibile iniziare a giocare. Il "game loop" potrebbe essere scritto così:

```
Function p_Game()
; game main loop
    If IsMusicPlaying(2)=False Then PlayMusic(2) ;musica,
maestro!
    waskeydown=False
    gameover=False
    p_InitGame()  ;inizializza il gioco
    Repeat ;inizio loop
        If IsKeyDown("ESC") ;esci se premo "ESC"
            gameover=True
            waskeydown=True
        EndIf
        p_DisplayGameGfx() ;disegna la grafica
        p_ReadPlayerInput() ;leggi l'input del giocatore
        p_AnimatePlayer() ;anima il mio avatar
        p_MoveAndAnimateOpponent() ;calcola AI nemica e
muovi l'avversario
        WaitTimer(1, 40) ;temporizza a 25fps!
        Until gameover Or IsKeyDown("ESC") ;fine loop
        If IsMusicPlaying(2) Then StopMusic(2) ;sto
uscendo. musica off
            p_PrepareMenu() ;ritorna al menu principale
EndFunction
```

Ci sono molte osservazioni che potrebbero essere fatte a proposito di questa sezione, perchè questa parte del codice rappresenta veramente il cuore pulsante del gioco. La trattazione del solo game loop di solito occupa almeno un intero capitolo sui libri dedicati all'argomento. Tuttavia, la cosa forse piu' importante da notare, è che le porzioni di codice che si occupano di muovere il nostro avatar o che simulano l'intelligenza artificiale dell'avversario dovrebbero essere sempre e comunque disgiunte dal codice che invece sostiene l'interfaccia utente e che serve a leggere l'input.

I motivi sono sostanzialmente due:
1. non avendo una visione "diretta" di tutto quello che succede, l'AI nemica non può barare, dato che accede solo alle informazioni che deve effettivamente vedere per operare correttamente;
2. si ottiene la capacità di passare liberamente e continuamente tra giocatori controllati dall'AI e giocatori controllati da un avversario umano.

Il motivo per il quale nel nostro esempio le cose sono

diverse e concentrate in un'unica routine ("p_MoveAndAnimateOpponent()") è semplicemente perche' il codice da cui l'esempio è tratto era nato come demo tecnico che richiedeva solo una limitata interazione esterna.

Conclusioni

L'arte del game coding è così interessante e complessa che non può ovviamente essere discussa con un solo breve articolo.
Basta infatti fare un giro su internet (o anche in libreria, se non fosse che in Italia nessun produttore sembra essere interessato a questo argomento) per trovare decine di pubblicazioni che trattano di programmazione di giochi per tutte le piattaforme disponibili e con diverse finalità (computer, dispositivi mobili ctc).
Nonostante la loro natura variegata, tutti questi libri sono concordi su una cosa: che, a differenza degli altri software, la simulazione di gioco ha una vita propria e che in assenza di input dell'utente il programma di simulazione manderà presto o tardi qualche orripilante creatura a sfasciare la testa del vostro avatar. Ciò probabilmente vi motiverà abbastanza da premere qualche tasto e diventare quindi parte attiva del game loop.

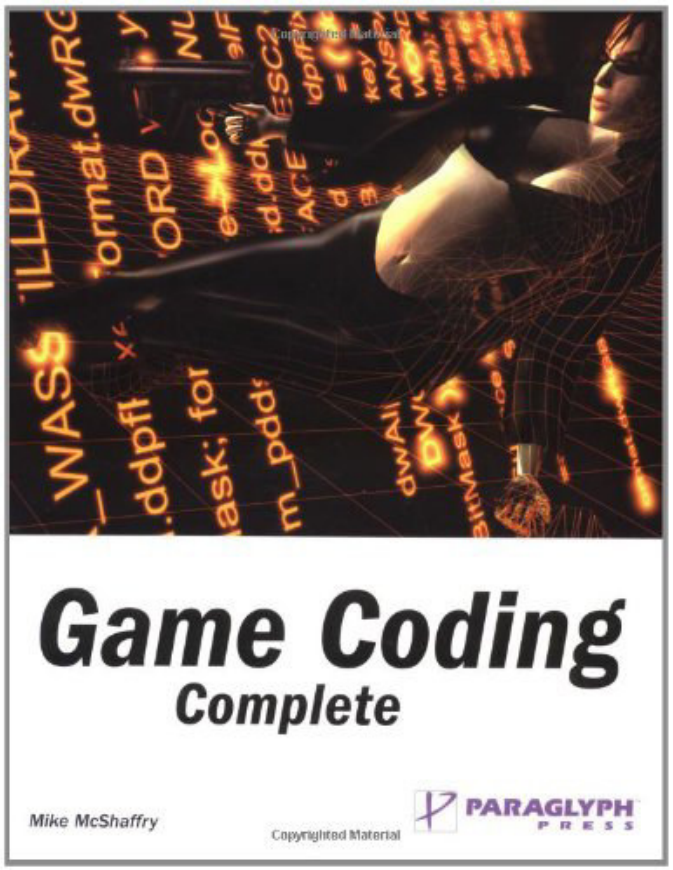
Nonostante questo loop possa assumere aspetti diversi, il modello pocanzi introdotto rappresenta un buon punto di partenza su cui basare i nostri successivi tutorial e può essere usato a prescindere dal linguaggio che verrà poi utilizzato per sviluppare effettivamente il progetto. Il suggerimento è quindi quello di sperimentare ed adattare questa struttura per trovare il modello che più si avvicina al vostro personale stile di programmazione.

Più il processo sarà schematizzato, ancorché non richiesto dal tipo di linguaggio in uso, più sarà facile l'eventuale cambio verso altri tipi di paradigmi di programmazione, come quella ad "oggetti" o addirittura quella "modulare".

Colgo l'occasione per ringraziare l'amico **Gianluca Girelli** e la redazione di **BITPLANE** per aver concesso a RetroMagazine la possibilità di pubblicare questo articolo tra le sue pagine.
Vi ricordo inoltre che potrete trovare altri contributi di Gianluca sul blog AmigaGuru che gestisce insieme a Tony Asknes: https://blog.amigaguru.com/

BIBLIOGRAFIA

The future was here - Jimmi Maher - The MIT Press
Game coding complete - Mike McShaffry - Paraglyph Press



ELENCO RISORSE

Introduzione ad Hollywood - prima parte:
http://www.retromagazine.net/getrm.php?id=16

Game programming:
http://en.wikipedia.org/wiki/Game_programming

Programmazione strutturata:
http://it.wikipedia.org/wiki/
Programmazione_strutturata

Programmazione orientata agli oggetti:
http://it.wikipedia.org/wiki/
Programmazione_orientata_agli_oggetti

Programmazione modulare:
http://it.wikipedia.org/wiki/
Programmazione_modulare

Andragogia:
http://it.wikipedia.org/wiki/Andragogia

The Dump Club 64

di The Dump Club 64 Team

Il Commodore 64 fu presentato la prima volta in anteprima mondiale il 6 giugno 1982 negli Stati Uniti e commercializzato un paio di mesi più tardi. Per questa gloriosa macchina, durante la sua vita commerciale, venne sviluppata una considerevole quantità di software. Prendendo come riferimento il più autorevole tentativo di catalogazione, il gamebase64 (http://www.gamebase64.com), soltanto per quanto riguarda i giochi possiamo contare oltre 27000 titoli.
Se poi volessimo aggiungere a questa cifra anche tutti gli applicativi senza ombra di dubbio supereremmo agevolmente la soglia dei 30000 titoli. Tra questi titoli 1% venne rilasciato su cartuccia mentre il restante 99% fu distribuito su nastro o supporto magnetico. Parliamo ovviamente di materiale ufficiale. Se invece volessimo considerare l'ambito dei supporti magnetici e nastri non ufficiali ad oggi, attraverso i file DAT di catalogazione, la somma raggiungerebbe la cifra di oltre 150.000 files per un totale di quasi 100 Gigabytes di informazioni.

In tutto il mondo si calcola siano stati venduti, tra i vari modelli, circa 22 milioni di macchine della Commodore, ancor oggi molte di queste macchine funzionano perfettamente, discorso diverso per il software, i nastri e i dischetti hanno, purtroppo, buone possibilità di non essere più leggibili.
Come abbiamo accennato, insieme ai nastri il principale formato di distribuzione del software per il Commodore 64 sono stati i dischetti floppy da 5 pollici e ¼. Nel loro involucro di plastica morbido (floppy) risiedeva il disco sempre di plastica magnetizzato.
I dati binari del gioco o dell'applicativo erano immagazzinati attraverso il sistema della polarità opposta: una polarità per l'uno (1) e una polarità per lo zero (0). I dischetti per questo motivo andavano conservati lontani da fonti magnetiche che avrebbero potuto smagnetizzarli con la perdita irrecuperabile dei dati contenuti.

Purtroppo le proprietà magnetiche di un disco si affievoliscono nel tempo, i segnali diventano deboli e la loro lettura inevitabilmente diventa più difficile. Col passare del tempo, quindi, la lettura risulta incostante finché i dati non possono essere più recuperati se non parzialmente e con apparecchiature hardware molto costose: superato questo punto di non ritorno il decadimento si conclude con la perdita completa dei dati.

Tutto questo accade quando le celle dei bit, che perdono lentamente la loro polarità e le loro transizioni di flusso, non vengono più rilevate o rilevate erroneamente. Questo fenomeno chiamato **Bit rot** può essere causato sia da origini ambientali, naturali o semplicemente dalla corrosione magnetica del materiale del supporto. Alcune stime indicano che la durata dei floppy disk è compresa tra 10 e 30 anni anche se in realtà nessuno sa di preciso quanto possano durare ancora i floppy disk. Non esiste una scadenza: tra le tante variabili ci sono la qualità del supporto, l'utilizzo, il metodo di conservazione, i fattori ambientali, i repentini sbalzi di temperatura e di umidità, le muffe e infine il deterioramento del supporto stesso. Tutti questi fattori possono concorrere a consegnare irrimediabilmente i dati contenuti nei dischi ad un destino fatale. Certamente non furono progettati per durare così a lungo, e credo che all'epoca nessuno avesse preso in considerazione l'idea che venissero ancora utilizzati dopo oltre 30 anni.

Nella pratica dell'attività di chi trasferisce su file i dischetti (**dumping**) ancora molti floppy riescono ad essere letti ma il margine di sicurezza è comunque ampiamente superato ed è palese che non ci sia più molto tempo per rimandare la loro preservazione: tutti dovrebbero trasferire i dati dei loro floppy il prima possibile. Il decadimento dei supporti, il campo magnetico terrestre e le condizioni ambientali non sono le uniche minacce che affliggono questa grande massa di dati: molti dischetti vengono gettati o distrutti senza riconoscerne a tutti gli effetti una valenza culturale, artistica o di ingegno tra le opere umane.

Se ci pensate ogni libro, pubblicazione, disco musicale, film ed opera artistica viene, come è giusto che sia, archiviata dalle librerie nazionali di tutto il mondo mentre questa stessa cosa non accade per il software per il quale nessuno sembra preoccuparsi di preservarlo dalla scomparsa e di tramandarlo alle future generazioni. Tutto è affidato al lavoro di pochi appassionati che in giro per il mondo fanno del loro meglio per conservare e proteggere. Oltre l'indifferenza culturale spesso anche la speculazione diventa un ostacolo a quest'opera di salvaguardia.

THE DUMP CLUB 64

I dischetti tendono a diventare poco diffusi e c'è sempre qualcuno che, cercando di ricavarne un profitto esagerato, rende estremamente costoso e impraticabile il lavoro di preservazione. Abbiamo poi il problema dei collezionisti che privi di lungimiranza accumulano compulsivamente enormi quantità di materiale senza trasferirlo e senza condividerlo. Viene da sé che questo non riguarda solamente la sfera della generosità verso il prossimo di cui purtroppo alcune persone sembrano essere prive ma soprattutto l'idea di sicurezza e condivisione allargata dei files contro possibili perdite di dati o malfunzionamenti dell'hardware. Senza scordarsi che un dischetto non trasferito e impilato come un trofeo su uno scaffale in pochi anni diventerà illeggibile e inutilizzabile perdendo tutti i suoi dati e il suo valore economico e non potendo così poi essere ripristinato per l'assenza di una copia di backup trasferita.

Un discorso distintivo va fatto per quanto riguarda la preservazione del software originale e di quello pirata: il software originale ha bisogno di una cura e un'attenzione maggiore in quanto è molto meno diffuso, difficilmente reperibile e soprattutto difficile da trasferire rispetto a quello piratato. Contiene protezioni, firme digitali, grafiche e hiscore originali, di solito contenenti i nomi dei programmatori, che molto spesso nelle copie pirata vengono totalmente alterate se non eliminate del tutto. Dall'altro canto però anche le copie piratate hanno la loro dignità in quanto raccolgono l'ingegno dell'altra faccia della medaglia: l'hacking. Possiamo ritrovare l'eleganza dello sblocco di una protezione con il cracking, oppure l'intro di un distributore pirata, i cheat per rendere il gioco più facile, spesso decisivi in quei giochi che, per colpa di qualche bug, non possono essere giocati interamente, senza contare le schermate di caricamento alternative, le demo, i miglioramenti al gioco se non addirittura le espansioni dei livelli. Insomma è giusto preservare ogni singolo dischetto che sia originale o piratato perché è un pezzetto di storia che, oltre ai singoli dati, conserva anche un contesto storico culturale che merita di essere salvato e tramandato a tutti coloro che verranno.

The Dump Club 64 è un'iniziativa Italiana, dedicata principalmente al Commodore 64, e formata da un gruppo di appassionati di retrocomputing che si prefigge proprio questo intento: conservare, catalogare e condividere gratuitamente su file il software dei vecchi supporti magnetici e dei nastri del Commodore 64. Speriamo con il nostro modesto lavoro di riuscire a salvaguardare più materiale possibile, ritenendolo un piccolo patrimonio di valore storico e culturale. La nostra missione pertanto è preservare tutto il software, possibilmente anche quello Italiano molto meno diffuso, allegando i manuali originali o le schede del tempo.

Ci teniamo a precisare che facciamo tutto questo senza alcun scopo di lucro e gratuitamente. Non prendiamo soldi dai click alle nostre pagine e non prendiamo soldi dai banner pubblicitari che non abbiamo, anzi di soldi ne mettiamo tanti di tasca nostra per coprire le spese e per le apparecchiature, senza contare il tempo che impieghiamo. Facciamo il nostro lavoro perché crediamo sia utile farlo e, come abbiamo scritto in precedenza, non avrebbe valore se non fosse condiviso con gli altri: per questo motivo tutto il software che trasferiamo è interamente condiviso con la comunità degli appassionati di retrocomputing e con chiunque altro voglia usufruirne.

Il nostro sito è (www.dumpclub64.it) e oltre alla condivisione dei file, non disdegna recensioni su giochi e riviste dell'epoca dedicate al mitico home computer. Nella nostra pagina web sono già pronte otto sessioni di dumping piene di programmi e utility facilmente scaricabili. Se volete contribuire ed inviarci qualcosa andate nella sezione upload e caricate i vostri files: è tutto molto gradito e, poiché di lavoro ce n'è tanto, chiunque voglia collaborare con noi è fortemente ben accetto.

In questi giorni stiamo rifacendo la grafica del nostro sito e abbiamo in cantiere tante novità interessanti. Grazie all'ospitalità e la gentilezza di **RetroMagazine** avremo mensilmente tra le sue pagine una rubrica che tratterà principalmente le tematiche relative al trasferimento e alla preservazione dei dati. Siamo molto onorati di questa partecipazione e ci auguriamo di poter dare un valido contributo.

The Dump Club 64 Team saluta tutti i lettori di RetroMagazine!

A nome della redazione di RetroMagazine colgo l'occasione per porgere un caro ed affettuoso saluto a tutti i componenti del **The Dump Club Team** e per augurare loro i migliori auspici per il buon esito del progetto.

Sono certo che i contributi che verranno condivisi nei numeri successivi saranno di sicuro interesse per i nostri lettori e per tutti gli appassionati di retrocomputing in generale.

Invito inoltre i nostri lettori a dare un'occhiata al materiale ricercato dal The Dump Club 64. Magari qualcuno di Voi potrebbe dare un contributo significativo al progetto.

Francesco Fiorentini

Calcolare in multipla precisione - parte III

di Alberto Apostolo

In questa terza parte del percorso riguardante i calcoli in multipla precisione o precisione arbitraria, saranno presentati due programmi che applicano il metodo di Newton e Raphson per calcolare rispettivamente il reciproco e la radice quadrata di un numero in multipla precisione.
Nella parte I (RM 15) è stato introdotto il calcolo in multipla precisione (presentando alcune routine scritte in C) e nella parte II (RM 16) è stato descritto un programma in C per calcolare exp(x) sempre in multipla precisione.
Nel compilare i programmi presentati in questa serie di articoli, si ricorda di accertarsi che il vostro compilatore C sia almeno "C99 compliant" affinché sia possibile dichiarare variabili intere a 64 bit di tipo "long long int".

Il metodo di Newton-Raphson

Il metodo di Newton-Raphson o metodo NR è stato introdotto da Newton nel 1669 per risolvere equazioni polinomiali (senza un uso esplicito delle derivate) e poi generalizzato da Raphson pochi anni più tardi.
Il metodo NR consiste nel risolvere l'equazione f(x) = 0 costruendo una successione come riportato in figura 1 [Mul06].

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Figura 1

Se la funzione f ha derivata continua e se r è una radice singola (cioè f'(r) <> 0) allora la successione converge quadraticamente a r a patto che il valore di "innesco" x0 sia abbastanza vicino ad r (infatti non è garantita la convergenza globale, ovvero per qualunque x0).

Per calcolare il reciproco di un numero a si usa la funzione f(x) = 1/x - a ottenendo la successione di figura 2 che converge a 1/a.

$$x_{n+1} = x_n \cdot (2 - a \cdot x_n)$$

Figura 2

Nel calcolo della radice quadrata di solito si usa la funzione f(x) = x*x - a che conduce alla successione in figura 3, già conosicuta da Al-Kwarizmi, da Erone di Alessandria e dai Babilonesi (2000 anni prima di Erone).

$$x_{n+1} = \frac{1}{2}\left(x_n - \frac{a}{x_n}\right)$$

Figura 3

Ma è presente una "costosa" divisione che impone di usare una strategia diversa.
Se si sceglie f(x) = 1/(x*x) - a , si ottiene una successione che converge a 1/sqr(a) come in figura 4. In seguito per calcolare sqr(a) è sufficiente moltiplicare per a.

$$x_{n+1} = \frac{x_n}{2}(3 - a \cdot x_n^2)$$

Figura 4

La condizione di arresto delle iterazioni consiste nel verificare che la differenza in valore assoluto tra x(n+1) e x(n) sia inferiore ad un certo errore e. Per ulteriore sicurezza si può prevedere un numero massimo di iterazioni.
Una interessante proprietà del metodo NR è che si "auto-corregge": un piccolo errore nel calcolo di x(n) non cambia il valore del limite. Così è possibile cominciare le iterazioni con precisione molto piccola e raddoppiare la precisione ad ogni iterazione. Si può dimostrare con opportune ipotesi che la complessità computazionale dei metodi per calcolare la radice quadrata o il reciproco è la stessa della moltiplicazione'[Mul06].
Si ipotizza che esistano due costanti a e b (0 < a,b < 1) tali che il tempo di esecuzione M(n) di una moltiplicazione tra due numeri con n bit soddisfa M(an)<= bM(n) per n abbastanza grande. Questa ipotesi è soddisfatta dalla moltiplicazione lunga (cfr. RM 16), dal metodo di Karatsuba e dalle trasformazioni FFT (metodi non trattati in questa sede).

La routine Comp

La routine Comp.C effettua il confronto in valore assoluto tra due numeri in multipla precisione x e y. Se abs(x) > abs(y) restituirà un flag valorizzato a +1, se sono uguali 0, -1 se abs(x) < abs(y).

I due programmi MP_INVX e MP_SQRX

Il programma MP_INVX.C effettua calcolo del reciproco di un numero x in multipla precisione mentre il programma MP_SQRX.C effettua il calcolo della radice quadrata di un numero in multipla precisione.
Le imprecisioni eventuali che si possono rilevare nei calcoli sono dovute all'adozione della rappresentazione in virgola fissa.

La prossima volta...

Non perdete l'ultima puntata.

Bibliografia
[Mul06] J.M. Muller, "Elementary functions: algorithms and implementation" ,2a ed., Springer, 2006.

```
/*-------------------------------------------------------------------*/
/* Funzione: confronta due numeri x e y in multipla precisione       */
/*-------------------------------------------------------------------*/
void Comp(long beta,long l,MultPrec *p_x,MultPrec *p_y,long *p_cnf)
{     long i;
      *p_cnf = 0L;
      for (i = 1L;i <= l;i++) {
          if ((*p_x).cifra[i] > (*p_y).cifra[i]) {*p_cnf= 1L;}
          if ((*p_x).cifra[i] < (*p_y).cifra[i]) {*p_cnf=-1L;}
          if (*p_cnf != 0L) { break; }
      }
}
```

Comp.C

```
/*********************************************************************/
/* Programma: MP_INVX                                                */
/* Funzione : Calcolo reciproco di un numero a in multipla  precisione*/
/*********************************************************************/
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>       /* tastiera-video PC IBM amb. Windows MS-DOS */
#define LMAX         22L /* ATTENZIONE! IN C   RANGE = (0:LMAX- 1)    */
#define BASE 1000000000L /* base aritmetica in multipla precisione    */
#define NMAX      50000L /* numero max termini di una serie           */
/* Dichiarazione dei tipi di variabile strutturati                   */
typedef struct { long segno;              /* segno (-1, 0,+1)        */
                 long cifra[(LMAX+1)];    /* cifre in base beta      */

        } MultPrec;
/* Dichiarazione di funzione e subroutine                            */
int  LeggiTasto(void);
void Mulz(long,long,MultPrec *p_a,long,long *p_rp);
void Divz(long,long,MultPrec *p_a,long,long *p_rp);
void Suma(long,long,MultPrec *p_s,MultPrec *p_a,long *p_rp);
void Init(long,long,long,MultPrec *p_a,long,long *p_rp);
void DisplayMultPrec(long,long,MultPrec *p_a,char *s);
void Inpx(long beta,long l,long lpi,long lv,
                                  MultPrec *p_x,char *nomevbl);
void Mmpx(long beta,long l,long lpi,MultPrec *p_c
                        ,MultPrec *p_a,MultPrec *p_b,long *p_rp);
void Comp(long beta,long l,MultPrec *p_x,MultPrec *p_y,long *p_cnf);

/* Programma principale                                              */
int main(void)
{
      long beta = BASE;       /* base aritmetica multipla precisione */
      long l    = LMAX;       /* lunghezza totale vettore mult.prec. */
      long lpi  = 3L;         /* lunghezza parte intera              */
      long lv   = l-0;        /* numero elementi visualizzabili      */
      MultPrec  s,p,x,a,e,q;  /* variabili multipla precisione       */
      MultPrec  *p_s =&s;     /* puntatore var. mult.prec.           */
      MultPrec  *p_p =&p;     /* puntatore var. mult.prec.           */
      MultPrec  *p_x =&x;     /* puntatore var. mult.prec.           */
      MultPrec  *p_a =&a;     /* puntatore var. mult.prec.           */
      MultPrec  *p_e =&e;     /* puntatore var. mult.prec.           */
      MultPrec  *p_q =&q;     /* puntatore var. mult.prec.           */
      char      nomevbl[6]=""; /* nome variabile m.p. (per la stampa) */
      long rp,  *p_rp=&rp;    /* riporto e puntatore riporto         */
      long rd,  *p_rd=&rd;    /* resto   e puntatore resto           */
      long i;                 /* contatore (vbl. di lavoro)          */
      long ovf, *p_ovf=&ovf;  /* indicatore overflow e puntatore     */
      long iz = 0L;           /* posizione prima cifra != zero di x  */
      long ww;                /* variabile di lavoro                 */
      long cnf,*p_cnf=&cnf;   /* indicatore di confronto             */
/*                                                                   */
      _clrscr();              /* pulisci schermo                     */
      printf("Programma MP_INVX : INIZIO ELABORAZIONE\n");
/*                                                                   */

      strcpy(nomevbl,"a ");
      Inpx(beta,l,lpi,lv,p_a,nomevbl);

      Init(beta,l,lpi,p_e,0L,p_rp);
      e.cifra[l-lpi] = 1L;     /* inizializzazione variabile di err.  */

      if (a.segno == 0L) {
         printf("a = zero! Impossibile proseguire\n");
      } else {
         for (i = 1L ;i <= l;i++) {
             if (a.cifra[i] != 0L) {
                iz = i;
                ww = a.cifra[i];
                break;
             }
         }
         Init(beta,l,lpi,p_x,1L,p_rp);
         Divz(beta,l,p_x,ww,p_rd);
         if ( iz < lpi ) {
            for (i = iz;i < lpi;i++) {
                Divz(beta,l,p_x,beta,p_rd);
            }
         } else {
            for (i = lpi;i < iz;i++) {
                Mulz(beta,l,p_x,beta,p_rp);
                if (rp != 0L) {
                   printf("Overflow calcolo innesco x0");
                   ovf = 1L;
                   break;
                }
            }
         }
         if ( ovf != 1L ){
            for (i = 1L;i < NMAX;i++){
                p=x;
                q=x;
                Mmpx(beta,l,lpi,p_p,p_p,p_a,p_ovf);
                if (ovf != 0L){printf("Overflow 2");break;}
                Init(beta,l,lpi,p_s,2L,p_rp);
                p.segno = -p.segno;
                Suma(beta,l,p_s,p_p,p_rp);
                if (rp != 0L){printf("Overflow 3");break;}
                Mmpx(beta,l,lpi,p_x,p_x,p_s,p_ovf);
                if (ovf != 0L){printf("Overflow 4");break;}
                q.segno = -q.segno;
                Suma(beta,l,p_q,p_x,p_rp);
                Comp(beta,l,p_q,p_e,p_cnf);
                if (cnf == -1){break;}
            }
            strcpy(nomevbl,"1/a");
            DisplayMultPrec(lv,lpi,p_x,nomevbl);
            Mmpx(beta,l,lpi,p_x,p_x,p_a,p_ovf);
            strcpy(nomevbl,"a*1/a");
            DisplayMultPrec(lv,lpi,p_x,nomevbl);
         }
      }

      printf("\n\nTermini utilizzati = %5d",i);

/*                                                                   */
```

```
      printf("\n\nProgramma MP_INVX : FINE ELABORAZIONE\n\n");
      return 0;
}
/*---------------------------------------------------------------------*/
/* Funzione : visualizza numero in multipla precisione base 10^9       */
/*---------------------------------------------------------------------*/
void DisplayMultPrec(long lv,long lpi,MultPrec *p_a,char *nomevbl)
{     long i,j;
      printf("\n%-6s = ",nomevbl);
      j=lpi%10L;
      if ((*p_a).segno >  0L) {printf("+");}
      if ((*p_a).segno == 0L) {printf(" ");}
      if ((*p_a).segno <  0L) {printf("-");}
      if (j != 0L) {
         for ( i = j ; i < 10L ; i++ ) {printf("          ");} /*10bl */
      }
      for ( i = 1L ; i <= lv ; i++ ) {
         if (i == (lpi+1L)) {printf(".");} else {printf(" ");}
         printf("%9.9ld",(*p_a).cifra[i]);
         if ((i % 10L)==j) {printf("\n          ");}      /* 10 blank */
      }
}
/*---------------------------------------------------------------------*/
/* Funzione: moltiplicazione corta a=a*z in multipla precisione        */
/*---------------------------------------------------------------------*/
void Mulz(long beta,long l,MultPrec *p_a,long z,long *p_rp)
{     long i,izer,sz,az;
      long long int ipro,w;
      sz=(z > 0L)-(z < 0L);            /* segno di z                 */
      az=sz*z;                         /* valore assoluto di z       */
      izer=0L;
      *p_rp=0L;
      for (i = l;i >= 1L;i--) {
         ipro=(*p_a).cifra[i];         /*workaround bug ottim. PellesC*/
         ipro = (ipro * az) + *p_rp;
         *p_rp=ipro/beta;
         w = beta;                     /*workaround bug ottim. PellesC*/
         (*p_a).cifra[i] = ipro - (*p_rp * w);
         if ((*p_a).cifra[i] > 0L) {izer= 1L;}
      }
      (*p_a).segno=(*p_a).segno * sz * izer;
}
/*---------------------------------------------------------------------*/
/* Funzione: divisione corta a=a/z in multipla precisione              */
/*---------------------------------------------------------------------*/
void Divz(long beta,long l,MultPrec *p_a,long z,long *p_rd)
{     long i,izer,sz,az;
      long long int idiv,w;
      sz=(z > 0L)-(z < 0L);            /* segno di z                 */
      az=sz*z;                         /* valore assoluto di z       */
      izer=0L;
      *p_rd=0L;
      for (i = 1L;i <= l;i++) {
         idiv = beta;                  /*workaround bug ottim. PellesC*/
         idiv = (*p_rd * idiv) + (*p_a).cifra[i];
         (*p_a).cifra[i]=idiv / az;
         w = az;
         *p_rd = idiv - ( (*p_a).cifra[i] * w);
         if ((*p_a).cifra[i] > 0L) {izer= 1L;}
      }
      (*p_a).segno=(*p_a).segno * sz * izer;
}
/*---------------------------------------------------------------------*/
/* Funzione: somma algebrica s=s+a in multipla precisione              */
/*---------------------------------------------------------------------*/
void Suma(long beta,long l,MultPrec *p_s,MultPrec *p_a,long *p_rp)
{     long i,isom,izer=0L,icnf=0L,az;
      *p_rp=0;
      if ((*p_a).segno != 0L) {
         if ((*p_s).segno == 0L) {
            for (i = 1L;i <= l;i++) {(*p_s).cifra[i]=(*p_a).cifra[i];}
            (*p_s).segno = (*p_a).segno;
         } else {
             az = (*p_s).segno * (*p_a).segno;
             if (az == -1L) {
                for (i = 1L;i <= l;i++) {
                   if ((*p_s).cifra[i] > (*p_a).cifra[i]) {icnf= 1L;}
                   if ((*p_s).cifra[i] < (*p_a).cifra[i]) {icnf=-1L;}
                   if (icnf != 0L) {
                      (*p_s).segno = (*p_s).segno * icnf;
                      break;
                   }
                }
             }
             for (i = l;i >= 1L;i--) {
                if (az == -1L) {
                   if (icnf >= 0L) {
                      isom = *p_rp + (*p_s).cifra[i] - (*p_a).cifra[i];
                   } else {
                      isom = *p_rp + (*p_a).cifra[i] - (*p_s).cifra[i];
                   }
                } else {
                   isom = *p_rp + (*p_s).cifra[i] + (*p_a).cifra[i];
                }
                *p_rp = (isom >= beta) - (isom < 0L);
                (*p_s).cifra[i] = isom - *p_rp * beta;
                if ((*p_s).cifra[i] > 0L) {izer= 1L;}
             }
             (*p_s).segno=(*p_s).segno * izer;
         }
      }
}
/*---------------------------------------------------------------------*/
/* Funzione: inizializzazione con un numero z                          */
/*---------------------------------------------------------------------*/
void Init(long beta,long l,long lpi,MultPrec *p_a,long z,long *p_rp)
{     long i,irp;
      (*p_a).segno=(z > 0L)-(z < 0L);     /* segno di z               */
      for (i = 1L;i <= l; i++){ (*p_a).cifra[i] = 0L; }
      *p_rp= (*p_a).segno*z;              /* valore assoluto di z     */
      for (i = lpi;i >= 1L;i--) {
         irp=(*p_a).cifra[i]+*p_rp;
         *p_rp=irp / beta;
         (*p_a).cifra[i] = irp - (*p_rp * beta);
      }
}
/*---------------------------------------------------------------------*/
/* Funzione: digitazione di un numero in m.p.                          */
/*---------------------------------------------------------------------*/
void Inpx(long beta,long l,long lpi,long lv,MultPrec *p_x,char *nomevbl)
{     long i;                  /* indice (vbl di lavoro)              */
      long k;                  /* indice (vbl di lavoro)              */
      long m;                  /* indice (vbl di lavoro)              */
      long z=10L;              /* vbl di lavoro                       */
      long f=1L;               /* cifre in base beta                  */
      long flag_dec = 0L;      /* 1=punto decimale attivo, 0=disatt.  */
      long cont_dec = 0L;      /* conta cifre dopo il punto decimale  */
/*                                                                    */
      int kbint = 0;           /* memorizza tasto premuto             */
      long rrp=0L, *p_rrp=&rrp;/* riporto e puntatore riporto         */
      long rrd=0L, *p_rrd=&rrd;/* resto   e puntatore resto           */
      MultPrec  w;             /* variabile multipla precisione       */
      MultPrec  *p_w =&w;      /* puntatore var. mult.prec.           */

      while (z<beta){f++;z=z*10L;}
      Init(beta,l,lpi,p_w,0L,p_rrp);
      while((kbint != 'q') && (kbint != 'Q')){       /*q,Q per uscire */
          _clrscr();
          for (i = l;i >= 1L;i--) {
              if ((*p_w).cifra[i] > 0L) {
                 if (w.segno == 0L) {w.segno = +1;}
              }
          }
          DisplayMultPrec(lv,lpi,p_w,nomevbl);
          printf("\n\n       (Q,q)= esci, (C,c)= cancella una cifra");
          printf(", (.)= punto decimale, (-) cambia segno\n\n");
          kbint = LeggiTasto();
          if (kbint == '.') {flag_dec = 1;}
          if (kbint == '-') {w.segno = -w.segno;}
          if (flag_dec == 0) {
```

```
            if ((kbint >= '0') && (kbint <= '9')
                              && (w.cifra[1] < (beta/10L)) ) {
               Mulz(beta,l,p_w,10,p_rrp);
               w.cifra[lpi] = w.cifra[lpi] + kbint - 48;
            }
            if ((kbint == 'c') || (kbint == 'C')) {
               Divz(beta,lpi,p_w,10,p_rrd); /* si parte da lpi */
            }
         } else {
            if ((kbint >= '0') && (kbint <= '9')
                              && (cont_dec < ((lv-lpi)*f)) ) {
               cont_dec++;
               z = (cont_dec-1) / f;
               k = lpi + z + 1;
               m = f - (cont_dec-1) + z*f;
               z=1;
               for (i = 1L;i<m;i++) {z=z*10L;}
               w.cifra[k] = w.cifra[k] + (kbint - 48)*z;
            }
            if ((kbint == 'c') || (kbint == 'C')) {
               z = (cont_dec-1) / f;
               k = lpi + z + 1;
               m = f - (cont_dec-1) + z*f;
               z=10;
               for (i = 1L;i<m;i++) {z=z*10L;}
               w.cifra[k] = w.cifra[k] - (w.cifra[k] % z);
               cont_dec--;
               if (cont_dec == 0){flag_dec=0;}
            }
         }
      }
      (*p_x) = w;
}
/*---------------------------------------------------------------------*/
/* Funzione realizzata :                                               */
/* restituisce all'esterno il codice ASCII di un tasto premuto         */
/*---------------------------------------------------------------------*/
int LeggiTasto(void)
{
      int c = 0;                  /* carattere                         */
      while (_kbhit()) _getch();  /* finche' ci sono caratteri         */
                                  /* li legge e li ignora              */
      c = _getch();               /* legge un tasto                    */
      return c;
}
/*---------------------------------------------------------------------*/
/* Funzione: moltiplicazione tra numeri in m.p. c = a * b              */
/*---------------------------------------------------------------------*/
void Mmpx(long beta,long l,long lpi,MultPrec *p_c
                        ,MultPrec *p_a,MultPrec *p_b,long *p_ovf)
{
      long cifra[(2*l)+1];     /* buffer cifre in base beta            */
      long i,j,k;              /* indici (vbl di lavoro)               */
      long rr;                 /* vbl riporto                          */
      long izer = 0L;          /* flag = 1 se cifra<>0, 0 altrimenti   */
      long long int ipro;      /* vbl lavoro                           */

      for (i=(l+1L);i<=(2L*l)+1;i++){cifra[i]=0L;}
      *p_ovf=0L;
      for (i = l;i >= 1L;i--) {
         rr=0L;
         if ((*p_b).cifra[i]>0L) {
            for (j = l;j >= 1L;j--) {
               k=i+j;
               ipro=(*p_b).cifra[i];              /*workaround PellesC*/
               ipro=rr+cifra[k]+((*p_a).cifra[j] * ipro);
               rr=ipro/beta;
               cifra[k]=ipro % beta;
            }
         }
         cifra[i]=rr;
      }
      for (i = 1L;i <= lpi;i++) {
         if (cifra[i]>0L) {
            *p_ovf=1L;
            break;
         }
      }
      for (i = 1L;i <= l;i++) {
         (*p_c).cifra[i]=cifra[i+lpi];
         if ((*p_c).cifra[i]>0L) { izer=1L; }
      }
      (*p_c).segno=(*p_a).segno * (*p_b).segno * izer;
}
/*---------------------------------------------------------------------*/
/* Funzione: confronta due numeri x e y in multipla precisione         */
/*---------------------------------------------------------------------*/
void Comp(long beta,long l,MultPrec *p_x,MultPrec *p_y,long *p_cnf)
{     long i;
      *p_cnf = 0L;
      for (i = 1L;i <= l;i++) {
          if ((*p_x).cifra[i] > (*p_y).cifra[i]) {*p_cnf= 1L;}
          if ((*p_x).cifra[i] < (*p_y).cifra[i]) {*p_cnf=-1L;}
          if (*p_cnf != 0L) { break; }
      }
}
```

```
/*************************************************************************/
/* Programma: MP_SQRX                                                    */
/* Funzione : Calcolo radice quadrata di a in multipla  precisione       */
/*************************************************************************/
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include <conio.h>       /* tastiera-video PC IBM amb. Windows MS-DOS */
#define LMAX         22L /* ATTENZIONE! IN C   RANGE = (0:LMAX- 1)    */
#define BASE 1000000000L /* base aritmetica in multipla precisione    */
#define NMAX      50000L /* numero max termini di una serie           */
/* Dichiarazione dei tipi di variabile strutturati                     */
typedef struct { long segno;              /* segno (-1, 0,+1)        */
                 long cifra[(LMAX+1)];    /* cifre in base beta      */

        } MultPrec;
/* Dichiarazione di funzione e subroutine                              */
int  LeggiTasto(void);
void Mulz(long,long,MultPrec *p_a,long,long *p_rp);
void Divz(long,long,MultPrec *p_a,long,long *p_rp);
void Suma(long,long,MultPrec *p_s,MultPrec *p_a,long *p_rp);
void Init(long,long,long,MultPrec *p_a,long,long *p_rp);
void DisplayMultPrec(long,long,MultPrec *p_a,char *s);
void Inpx(long beta,long l,long lpi,long lv,
                                    MultPrec *p_x,char *nomevbl);
void Mmpx(long beta,long l,long lpi,MultPrec *p_c
                         ,MultPrec *p_a,MultPrec *p_b,long *p_rp);
void Comp(long beta,long l,MultPrec *p_x,MultPrec *p_y,long *p_cnf);

/* Programma principale                                                */
int main(void)
{
      long beta = BASE;        /* base aritmetica multipla precisione */
      long l    = LMAX;        /* lunghezza totale vettore mult.prec. */
      long lpi  = 3L;          /* lunghezza parte intera              */
      long lv   = l-0;         /* numero elementi visualizzabili      */
      MultPrec  s,p,x,a,e,q;   /* variabili multipla precisione       */
      MultPrec  *p_s =&s;      /* puntatore var. mult.prec.           */
      MultPrec  *p_p =&p;      /* puntatore var. mult.prec.           */
      MultPrec  *p_x =&x;      /* puntatore var. mult.prec.           */
      MultPrec  *p_a =&a;      /* puntatore var. mult.prec.           */
      MultPrec  *p_e =&e;      /* puntatore var. mult.prec.           */
      MultPrec  *p_q =&q;      /* puntatore var. mult.prec.           */
      char      nomevbl[6]=""; /* nome variabile m.p. (per la stampa) */
      long rp,  *p_rp=&rp;     /* riporto e puntatore riporto         */
      long rd,  *p_rd=&rd;     /* resto   e puntatore resto           */
      long i;                  /* contatore (vbl. di lavoro)          */
      long ovf, *p_ovf=&ovf;   /* indicatore overflow e puntatore     */
      long iz = 0L;            /* posizione prima cifra != zero di x  */
      long ww;                 /* variabile di lavoro                 */
      long cnf,*p_cnf=&cnf;    /* indicatore di confronto             */

/*                                                                     */
      _clrscr();               /* pulisci schermo                     */
      printf("Programma MP_SQRX : INIZIO ELABORAZIONE\n");
/*                                                                     */
      strcpy(nomevbl,"a ");
      Inpx(beta,l,lpi,lv,p_a,nomevbl);

      Init(beta,l,lpi,p_e,0L,p_rp);
      e.cifra[l-lpi] = 1L;     /* inizializzazione variabile di err.  */

      if (a.segno <= 0L) {
         printf("a <= zero! Impossibile proseguire\n");
      } else {
         for (i = 1L ;i <= l;i++) {
             if (a.cifra[i] != 0L) {
                iz = i;
                ww = a.cifra[i];
                break;
             }
         }
         if ( iz < lpi ) {
            if ((iz%2L)==0L) {ww=sqrt(ww)*sqrt(BASE);}
            Init(beta,l,lpi,p_x,1L,p_rp);
            Divz(beta,l,p_x,ww,p_rd);
            iz = (lpi-iz) / 2L;
            for (i = 1;i <=iz;i++) {
                Divz(beta,l,p_x,beta,p_rd);
            }
         } else {
            if ((iz%2L)==0L) {ww=sqrt(ww)*sqrt(BASE);}
            Init(beta,l,lpi,p_x,1L,p_rp);
            Divz(beta,l,p_x,ww,p_rd);
            iz = (iz-lpi+1) / 2L;
            for (i = 1;i <=iz;i++) {
                Mulz(beta,l,p_x,beta,p_rp);
                if (rp != 0L) {
                   printf("Overflow calcolo innesco x0");
                   ovf = 1L;
                   break;
                }
            }
         }
         if ( ovf != 1L ){
            for (i = 1L;i < 50L;i++){
                p=x;
                q=x;
                Mmpx(beta,l,lpi,p_p,p_p,p_x,p_ovf);
                if (ovf != 0L){printf("Overflow 2");break;}
                Mmpx(beta,l,lpi,p_p,p_p,p_a,p_ovf);
                if (ovf != 0L){printf("Overflow 3");break;}
                Init(beta,l,lpi,p_s,3L,p_rp);
                p.segno = -p.segno;
                Suma(beta,l,p_s,p_p,p_rp);
                if (rp != 0L){printf("Overflow 4");break;}
                Mmpx(beta,l,lpi,p_x,p_x,p_s,p_ovf);
                if (ovf != 0L){printf("Overflow 5");break;}
                Divz(beta,l,p_x,2L,p_rd);
                q.segno = -q.segno;
                Suma(beta,l,p_q,p_x,p_rp);
                Comp(beta,l,p_q,p_e,p_cnf);
                if (cnf == -1){break;}
            }
            s=x;
            strcpy(nomevbl,"1/sqr");
            DisplayMultPrec(lv,lpi,p_x,nomevbl);
            Mmpx(beta,l,lpi,p_x,p_x,p_a,p_ovf);
            strcpy(nomevbl,"sqr");
            DisplayMultPrec(lv,lpi,p_x,nomevbl);
            Mmpx(beta,l,lpi,p_x,p_x,p_s,p_ovf);
            strcpy(nomevbl,"sq/sq");
            DisplayMultPrec(lv,lpi,p_x,nomevbl);
         }
      }

/*     printf("\n\nTermini utilizzati = %5d",i); */

/*                                                                     */
      printf("\n\nProgramma MP_SQRX : FINE ELABORAZIONE\n\n");
      return 0;
}
```

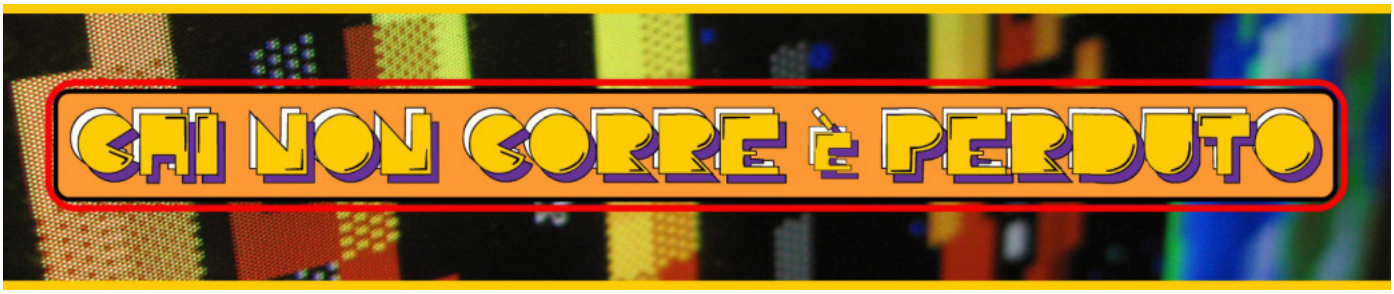
Attenzione! Il programma MP_SQRX.C ha solo il main() perché le routine sono le stesse del programma MP_INVX.C

```
a      = +                                                                       000000000 000000000 000000003
          .768585747 473838474 758585758 484848300 000000000 000000000 000000000 000000000 000000000 000000000
           000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000

       (Q,q)= esci, (C,c)= cancella una cifra, (.)= punto decimale, (-) cambia segno


1/a    = +                                                                       000000000 000000000 000000000
          .265351531 584579395 533947022 709510817 108920463 695904529 577477777 802034404 465956686 579354256
           941141597 740805278 801772104 743410577 283186139 117607628 031161580 043931403 393172780
a*1/a  = +                                                                       000000000 000000000 000000000
          .999999999 999999999 999999999 999999999 999999999 999999999 999999999 999999999 999999999 999999999
           999999999 999999999 999999999 999999999 999999999 999999999 999999999 999999999 999999998

Termini utilizzati =     9

Programma MP_INVX : FINE ELABORAZIONE


C:\Users\Utente\Desktop\C_Esercizi>



a      = +                                                                       000000000 000000000 000000000
          .000000000 000035553 452525363 525200000 000000000 000000000 000000000 000000000 000000000 000000000
           000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000

       (Q,q)= esci, (C,c)= cancella una cifra, (.)= punto decimale, (-) cambia segno


1/a    = +                                                                       000000000 000028126 663628141
          .561168376 209993629 973200101 293545652 316005098 242781522 480573022 731607278 576017503 597822459
           910343148 051177922 137283794 933484015 586192250 861781146 942691210 161215697 434588884
a*1/a  = +                                                                       000000000 000000000 000000001
          .000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000
           000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000

Termini utilizzati =     7

Programma MP_INVX : FINE ELABORAZIONE


C:\Users\Utente\Desktop\C_Esercizi>


a      = +                                                                       000000000 000000000 000000003
          .000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000
           000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000

       (Q,q)= esci, (C,c)= cancella una cifra, (.)= punto decimale, (-) cambia segno


1/sqr  = +                                                                       000000000 000000000 000000000
          .577350269 189625764 509148780 501957455 647601751 270126876 018602326 483977672 302933345 693715395
           585749525 225208713 805135567 676656648 364999650 826270551 837364791 216176031 077300769
sqr    = +                                                                       000000000 000000000 000000001
          .732050807 568877293 527446341 505872366 942805253 810380628 055806979 451933016 908800037 081146186
           757248575 675626141 415406703 029969945 094998952 478811655 512094373 648528093 231902307
sq/sq  = +                                                                       000000000 000000000 000000001
          .000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000
           000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000001

Programma MP_SQRX : FINE ELABORAZIONE


C:\Users\Utente\Desktop\C_Esercizi>
```

Dimostrazione d'uso dei programmi MP_INVX.C e MP_SQRX.C

Intervista a Simone Guidi

di Starfox Mulder



Starfox: Ciao Simo è un piacere poterti intervistare per RetroMagazine. Siccome ci conosciamo da anni stavolta la presentazione la faccio direttamente io: uomo di mare, scribacchino, padre. Arriva su un cargo battente bandiera liberiana e si installa nel posto più vicino al distributore di merendine. Prima di tutto però un grandissimo appassionato di Retrogaming che da anni tiene attivo il blog "Chi non corre è perduto". Cresciuto ad Atari e sale giochi, Simone Guidi appare oggi non solo come una delle persone più preparate ed entusiaste del settore ma sicuramente il più umile e simpatico. Ho dimenticato qualcosa?

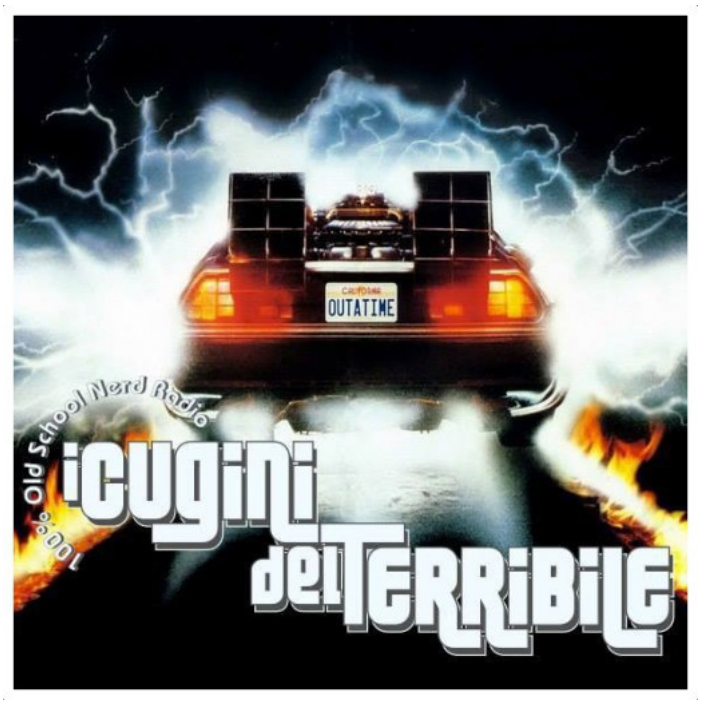
Simone: Ahah l'esagerazione fatta ad uomo. No, non hai dimenticato niente...anzi sì, hai scordato: appassionato di cinema, superdotato, lavoratore indefesso e grande figl de put, lup man...eccetera!

S.M.: Io mi avvicinai a questa passione circa 7 anni fa e la prima cosa che feci fu leggere avidamente ogni cosa presente in rete. Il tuo blog non solo era il più divertente ma anche il primo ad apparire cercando la parola "retrogaming". Hai una vagonata di followers da far invidia alla Ferragni o sei solo più bravo degli altri a foraggiare google per farti comparire nei primi risultati di ricerca?

Simone: Ti dirò che il blog è nato come valvola di sfogo, per puro caso, doveva essere un supporto ai libri che stavo scrivendo. La casa editrice che mi pubblicava riceveva dei fondi europei che implicavano l'informatizzazione dei suoi clienti. Mi obbligarono in pratica. Sta cosa decadde subito ed io dedicai il blog alle mie passioni nerd.
Per anni il blog aveva un impostazione standard e banalissima ma poi questa cosa cambiò quando mi accorsi che certa gente mi copiava gli articoli. E' accaduto due volte (l'articolo sui Rolling Stones contro i Verve e quello su Bakaroo Banzai) e da lì ho capito che o cominciavo ad apparire "professionale" o sto trend sarebbe continuato. Guarda caso avevo ragione: "l'abito fa il monaco". Una volta ripulita la grafica e seguendo un impaginazione più studiata tutto è cambiato. Al momento han smesso di copiarmi.

S.M.: Sul blog (che trovate all'indirizzo www.simoneguidi.info) parli di tanti argomenti legati a tutte le tue passioni ma un sondaggio che facesti un po' di tempo fa rivelò quanto gli argomenti più seguiti, se ben ricordo, fossero i film ed il retrogaming. Continuerai sulla linea che hai sempre tenuto o ci aspettano delle novità in futuro?

Simone: Novità, novità che sono già state messe in atto. Quando lanciai il sondaggio ero in un momento di stanchezza perché non riuscivo a capire quale fosse il genere di articoli che tirasse di più. Dal sondaggio in poi capii che avevo un pubblico da soddisfare e non potevo più permettermi di dissipare energie. Ho iniziato quindi a



concentrarmi su quei due argomenti ed ho visto che le cose sono migliorate. Dei due meglio il cinema perché almeno lì ci sono anche le interazioni, il retrogamer medio è più timido e commenta poco!

S.M.: Parliamo ora di podcasting, altro settore in cui spadroneggi. Chi segue la radio sa quanta differenza ci sia tra un programma che compare su una radio nazionale rispetto ad un podcast presente in rete, questi ultimi spesso privi di una colonna sonora o anche solo di un montaggio. Ore di persone che parlano e basta, in sostanza: due palle epiche. Tu hai invece uno stile molto curato ed intervallato da momenti leggeri, canzoni e musiche di sottofondo. Ci dici se ti sei ispirato a qualcuno nelle modalità di realizzazione dei Cugini del Terribile?

Simone: Premetto che sono un fortissimo ascoltatore della radio ed apprezzo particolarmente Radio Due e Radio Capital. Virgin Radio solo per la musica. Le mie due fonti di ispirazione più forti sono state: il podcast di "Mondo Nerd" di Lucarini e Macaluso, che prima girava su Radio Versilia (gran successo in podcast e pessimo in radio) e poi divenne il podcast ufficiale di Lega Nerd; e quella bellissima trasmissione radiofonica che fu Dispenser (su Radio Due) di Matteo Bordone e poi Costantino della Gherardesca. Dalla fusione di queste due trasmissioni ho creato il format dei Cugini del Terribile.

S.M.: Penso che in ogni genere di passione ce ne siano alcuni, ma nel mondo del retrogaming non è affatto raro trovarne in buon numero, quindi ti chiedo: cosa ne pensi di quella pletora di esperti o presunti tali che trattano tutti come se dovessero pendere dalle loro labbra?

Simone: C'è tantissima gente che genuinamente ha una passione e la coltiva per crescere in modo sano. Frequentano gli eventi, si pongono in modo umile verso chiunque e traggono il meglio da un hobby. Purtroppo ci sono anche un 20-30% di persone che non sono semplicemente appassionate di retrogaming: hanno solo quello. Capita che abbiano lavorato in quell'ambiente, scritto per riviste specializzate in tempi andati o in generale si trovino a vivere una vita di ricordi. Non considerano il fatto che nell'epoca di internet chiunque può apprendere con grande facilità ed usufruire di ogni contenuto. Alla fine si tratta di giochi, fatti all'epoca per divertire dei giovani. La materia andrebbe approcciata con quello spirito leggero che avevamo da giovani mentre magari costoro, invecchiando, hanno perso di vista il fatto che si parla di videogiochi.

S.M.: Ti ringrazio tanto per la disponibilità e invito tutti i nostri lettori a seguirti sul blog, su facebook e su spreaker.

Simone: Forse anche su Twitch...chissà. Grazie a te!

RIFERIMENTI:

www.simoneguidi.info

www.spreaker.com/user/dottorgonzo

www.facebook.com/icuginidelterribile/

BRUCE LEE
RETURN OF FURY

Anno: 2019
Codice: Ron J Fortier
Grafica: Kelly Day
Sonoro: John A Fitzpatrick
Piattaforma: C64
Genere: Azione

WELCOME TO
BRUCE LEE - RETURN OF FURY
35 years later, The Emperor had rebuilt the Forbidden World.
Enemies regrouped and came back even stronger.
There was no backing out of the challenge ahead.
Be like water, my friend, and you will be victorious.
The Legend has returned - The Adventure continues.

Ho voluto lasciare l'introduzione in inglese, come si legge nel sito Megastyle, perche' traducendola avrei corso il rischio di perdere in efficacia nel messaggio!

Bruce Lee - Return of Fury è la continuazione del gioco originale 'Bruce Lee' uscito per Commodore 64 negli anni '80. Stesso gameplay ma in un nuovo mondo, tutto da esplorare e sconfiggere. Ebbene sì, 35 anni dopo l'uscita di Bruce Lee ad opera di DataSoft nel 1984, siete pronti ad accettare la sfida ed impersonare di nuovo l'archetipo delle arti marziali per eccellenza? Ovviamente la risposta non può che essere una sola!

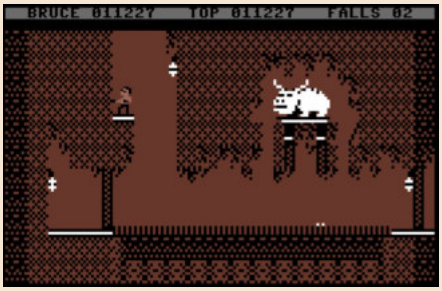
Partiamo subito dall'immagine di presentazione del gioco: meravigliosa! Perfetta per calarvi nella parte, con i colori cupi del Commodore 64 a dare forza ai muscoli tesi di Bruce. Vi garantisco che dopo averla vista vi verrà voglia di menare le mani, un po' lo stesso effetto che si ha dopo aver visto un film di arti marziali.

Lo scopo del gioco è quello di raccogliere tutte le lampade presenti in ogni schermata. Per raccoglierle Bruce può saltare da fermo, saltare correndo, arrampicarsi sulle corde ed addirittura lasciarsi cadere da grandi altezze. I movimenti di combattimento invece si limitano al pugno, con tanto di guanto nero ad enfatizzarne l'efficacia ed al calcio volante, vero marchio di fabbrica del nostro campione. All'occorrenza Bruce può

Via... si parte!

Chi e' il panzone verde? Shreck?

Il primo passaggio impegnativo



Sempre piu' difficile

abbassarsi per schivare i colpi di Yamo, abilissimo nei calci volanti, ma soprattutto per evitare i fendenti della spada del Ninja che sono ovviamente letali!

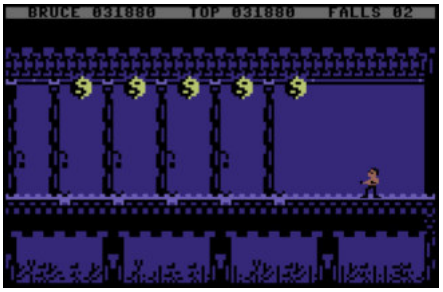
Se siete degli appassionati di Kung Fu Master e pensate che il gioco sia un mero colpire gli avversari o gli oggetti come nell'arcade della Irem, ne resterete delusi. Oppure piacevolmente sorpresi. Come detto prima, lo scopo del gioco è quello di collezionare le lampade e per farlo dovrete affrontare in ogni schermata dei piccoli enigmi. Alcune lampade sbloccano i passaggi ai livelli successivi e per raccoglierle dovrete evitare trabocchetti, lame, mine esplosive e chi più ne ha più ne metta. I combattimenti si riducono veramente all'osso, si fa decisamente prima a schivare i nemici che perdere tempo a combatterli.

Gli enigmi non sono mai complicati e guadagnare l'uscita è soltanto questione di tempo e di tempismo. Dopo il primo sguardo capirete immediatamente cosa fare, ma riuscire a farla è veramente un altro discorso.

Tecnicamente il gioco si compone di una serie di stanze a schermata statica. La grafica è essenziale, nello stile del Bruce Lee originale, ma sempre ad ottimi livelli. Nonostante tutto il gioco sia molto pixelloso, ci troviamo di fronte ad un vero capolavoro stilistico. Ogni schermata è differente dalle altre sia nello stile che nei colori utilizzati e dopo le prime due avrete una voglia pazza di vedere come sono tutte le altre. Vi garantisco che non re resterete mai delusi!

Il comparto sonoro si limita ad una musica di sottofondo durante il caricamento del gioco ed ad una simpatica musichetta nello schermo dei crediti.

Durante il gioco non c'è sonoro, solo degli effetti sonori che accompagnano i passi del nostro eroe e qualche altro effetto speciale. Molto efficace quello del calcio volante, un po' meno convincento quello dell'esplosione delle mine.
Chi legge le mie recensioni comunque sa che raramente apprezzo la musica nei giochi, se non particolarmente ben fatta, quindi personalmente apprezzo la scelta.

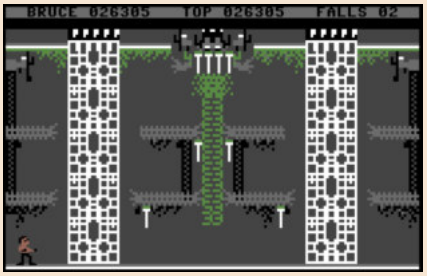


Quello che immediatamente colpisce del gioco è invece la giocabilità. È un gioco che 'prende'! Difficilmente riuscirete a terminarlo con 2 o 3 partite e vorrete continuare a giocarlo per vedere cosa c'è nella schermata successiva...
A differenza comunque di altri giochi simili, questo Bruce Lee ha quel qualcosa in più che ve lo farà rigiocare anche una volta completato. Fosse soltanto per far vedere a qualche vostro amico cosa ancora hanno da offire macchine a 8 bit nel 2019 se ben programmate.

Personalmente sono un appassionato di questo genere di giochi, che uniscono un minimo di ragionamento all'azione, quindi per me questo Bruce Lee - Return of Fury e' un gran bel gioco!

di **Francesco Fiorentini**

Per scaricare il gioco:
https://megastyle.itch.io/bruce-lee-return-of-fury



GIUDIZIO FINALE

» Giocabilità 90%

Personalmente sono un appassionato di questo genere di giochi, che uniscono un minimo di ragionamento all'azione, quindi per me questo Bruce Lee - Return of Fury e' un gran bel gioco!

» Longevità 80%

A differenza comunque di altri giochi simili questo ha quel qualcosa in piu' che ve lo fara' rigiocare anche una volta completato. Fosse soltanto per far vedere a qualche vostro amico cosa ancora hanno da offire macchine a 8 bit nel 2019.

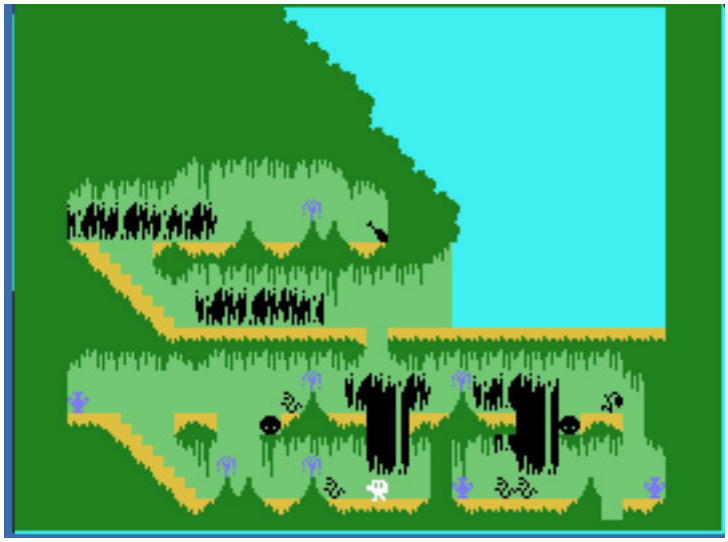
English version of this review on AmigaGuru Gamer's Blog: https://blog.amigaguru.com/?p=32211

MORPHY - SMART BALL

Morphy by Riccardo Tesio
Anno: 1984

Smart Ball by Alessandro Betori Anno 2011

Piattaforma: TI99/4A



In un numero precedente di RM abbiamo presentato tra i segreti del TI-Basic la possibilità di sfruttare gli sprite senza far uso dei moduli aggiuntivi quali l'Extedend Basic o la Mini-Memory. Qui mostriamo gli unici due programmi che hanno sfrutttato tale possibilità che sono Morphy e SmartBall.

Cosa abbiamno da dire su di loro? Innanzitutto che la scoperta tutta italiana di questa possibilità di usare gli sprite è stata fatta nel lontanto 1984 da Riccardo Tesio insieme al fratello Corrado grazie ad una loro lettura attenta del manuale TI-Internal dove è spiegato al bit tutto l'Hardware e il linguaggio GPL del TI99. Insieme crearono il game Morphy che fu pubblicato sulla rivista MC Computer numeri 34 e 35.

Se leggete gli articoli presenti su quella antica rivista noterete che l'approccio usato per creare il loader necessario ad abilitare gli sprite è differente da quello che ho esposto nel precedente articolo infatti quello presentato è una rivisitazione del boot loader ristudiato solamente nel 2017.

Motivo di questo apparente disinteresse è stato per quanto bizzarro possa essere, nella grande fiducia che gli utenti esteri (specialmente gli americani) avevano ed hanno nelle spegazioni tecniche sia software che hardware rilasciate per il TI99/4A dalla casa madre Texas Instruments.

Infatti la T.I. ha sempre spiegato che l'uso degli sprite non è possibile usando il TI-Basic in quanto non esistono dei comandi diretti che effettuano tale manipolazione degli stessi, tipo aumentare le dimensioni, definire i colori, aspetto, velocità automatica di spostamento ecc, tutte istruzioni presenti nelle cartucce XB o MM prima menzionate. Ma da una attenta lettura dei listati dei programmi qui proposti notiamo che il trucco di usare questi sprite è di trattarli come se fossero dei normali caratteri usando le istruzioni basic CALL CHAR, HCHAR, VCHAR, GCHAR.

Questo sul lato tecnico relativo alla programmazione.... ora passiamo in rassegna altri fattori su come

TABELLA PUNTI

GAME OVER

BOOT Loader Originale

```
Steps for TI-99/4A + Tape Recorder:
1. (Only first time) Enter Ti BASIC
and digit and RUN this program  (or
load BOOT file)...
100 CALL CLEAR
110 INPUT "REGISTRO(0-7),DATO(0-255)?
":R,D
120 A=18429-(256*R+D)
130 X$=CHR$(0)
140 OPEN #1:"CS1",OUTPUT,FIXED
150 PRINT #1:X$&X$&X$&X$&X$&X$&CHR$
(INT(A/256))&CHR$(A)
2. (Only first time) Run program and
insert 5,15 like input, then press
ENTER key.
3. (Only first time) Program run and
write (SAVE CS1) on tape a special
file...
4. (Only first time) Reset Ti99/4A.
5. Enter Ti-Basic and Load special
file from tape with OLD CS1.
6. RUN program and after few seconds,
screen shows black lines.
   Press any key (letter A,B ecc.)
and ENTER key: you get MEMORY FULL.
7. Write NEW and press ENTER key.
8. Load TI BASIC game MORPHY from
tape with OLD CS1 and RUN game.
Same process for game SMART BALL.
```

sono stati creati questi giochi. Iinfatti pur essendo totalmente differenti sia come grafica che come concept del gioco stesso, ambedue hanno dovuto ricercare un modo di usare al meglio questi minimal Sprite. Come?
Usando esattamente all'opposto lo Sprite.. su Morphy abbiamo che gli Sprite sono usati per visualizzare il nostro Eroe, il Boss finale ed le scritte di Game Over, su Smart Ball invece sono usati per lo sfondo, per la bandierina e specialmente per la pallina in modo da avere un effetto realistico del movimento.
Il primo problema nel creare un gioco usando il TI-Basic è che l'effetto di spostamento di un oggetto sullo schermo avviene non in modo continuo ma a salti di otto pixel per volta, questo modo standard di lavoro unito alla lentezza del TI-Basic porta ad un effetto di sfarfallio grafico e di ritardo nell'interazione dei comandi. Basta che tu gamer provi Morphy e vedrai che devi anticipare le mosse prima di vederle a video.. questo perchè la gestione del movimento dello Sprite è stata creata come se fosse un carattere, comportando certe volte un simpatico effetto teletrasporto dove l'eroe scompare per riapparire alla casella successiva.
I fratelli Tesio sapendo tale limitazione e grazie alla loro capacità grafica riuscirono a dotare il loro gioco con una grafica spettacolare, in quanto con un uso sapiente dei caratteri crearono il labirinto a scorrimento verticale, concetto già usato in altri giochi del TI99 vedi GROG Maze.
Di contro Alessandro Betori il creatore di Smart Ball nel creare il gioco di mini golf ha pensato di usare una grafica spartana ma di puntare molto sulla giocabilità. Per ovviare al fatto che simulando un movimento Carattere, Sprite, vi sono le limitazioni citate sopra che si evidenziano maggiormente quando vi è l'interazione con la tastiera, Alessandro pensò di creare un movimento automatico dello sprite senza che vi fosse nessuna interazione. Il problema era come fare ciò? Escogitò allora di usare il modus comandi di famosi giochi che sfruttano il calcolo matematico il cui capostipite come genere è il famoso gioco Artillery dove due nemici si combattono a colpi di cannone. Vedrai che, impostando i valori reativi al lancio della pallina come se fosse un alzo balistico, avrai il piacere di veder visualizzato con il TI Basic il movimento fluido dell'animazione.

di **Ermanno Betori**

Riferimenti WEB

Link Rivista Micro-Computer

http://www.mc-online.it/

Link dove effettuare il download di questi giochi, guardare sezione TI-Basic per Morphy e XB per Smart Ball

http://tigameshelf.net/

GIUDIZIO FINALE

» Giocabilità 80%

Bellissima grafica su Morphy ma il problema del ritardo sui controlli lo rende molto difficile, rigiocato dopo 30 anni un massacro. Rimane uno dei giochi migliori creati in TI-Basic.
Ball offre uno dei migliori giochi di Golf ma l'interazione obbligatoriamente "statica" lo penalizza.

» Longevità 85%

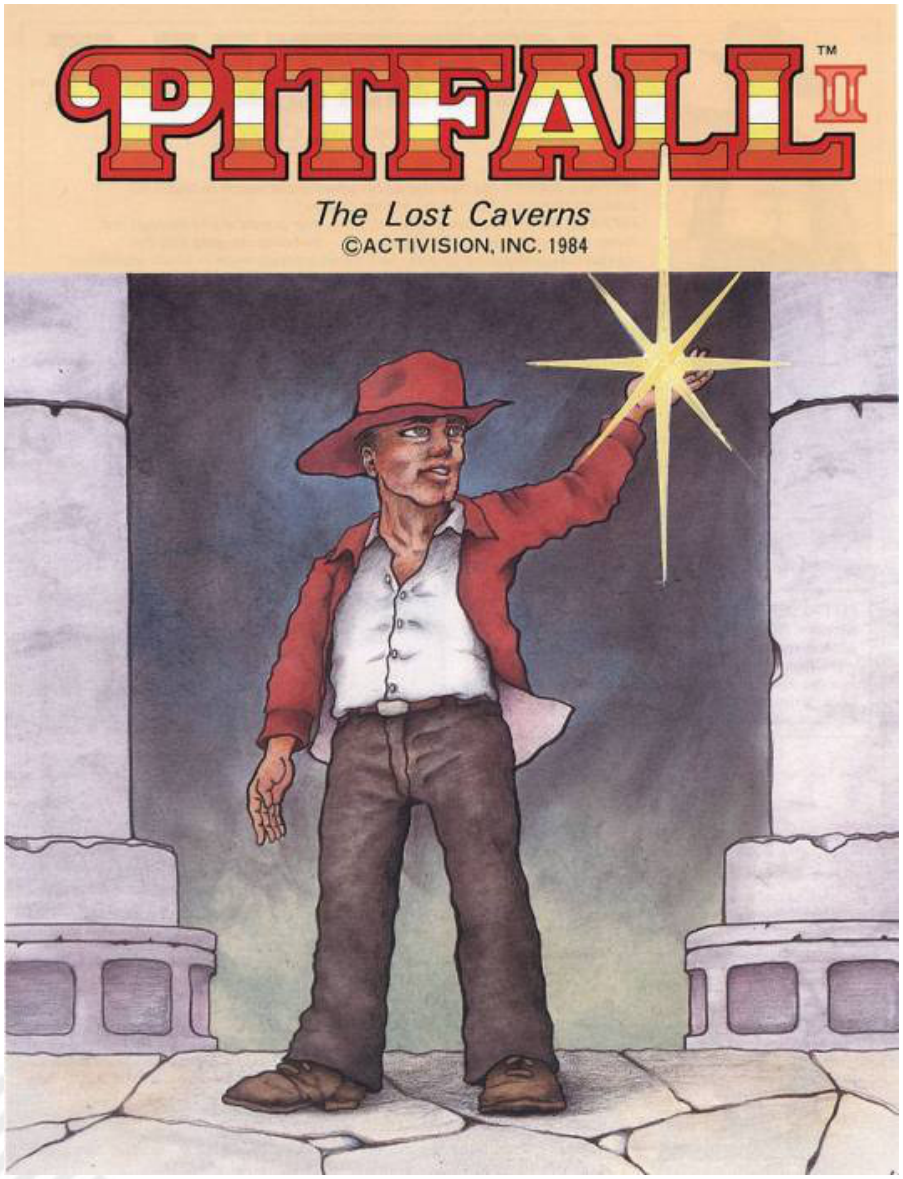
Dopo aver recuperato i tesori in Morphy difficilmente nell'immediato uno lo rigioca, comunque da al vero Gamer quel senso di vittoria cosa che manca a molti giochi moderni. Ball invece è molto diverte da giocare in più persone in quanto mette alla prova le loro capacità.

Visione Notturna :-)

PITFALL II
THE LOST CAVERNS

Anno: 1984
Publisher: Activision

Piattaforma: Multi
Genere: Azione



Pitfall II The Lost Caverns è un videogioco di genere platform pubblicato dalla Activision nel 1984 per computer e console casalinghi e riprogrammato dalla Sega nel 1985 per il mercato arcade.

La versione arcade parte dalle versioni home di Pitfall! e Pitfall II: Lost Caverns unendole in un ibrido dotato di veste grafica cartoonesca. Questa versione fu adattata per Sega SG-1000 e pubblicata solo in Giappone.

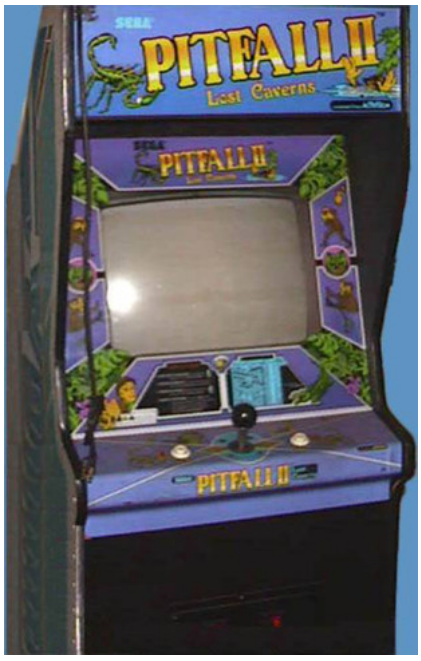
Nel gioco impersoneremo un avventuriero famoso in tutto il mondo e straordinario cacciatore di tesori, Pitfall Harry. Dovremo attraversare una rete dimenticata di foreste e caverne raccogliendo tesori per ottenere punti.

Questa versione, a differenza dell'originale, prevede la presenza

delle vite, così come di un timer a cui è possibile aggiungere 30 secondi raccogliendo un tesoro. Sono disponibili solo quattro livelli a progressione lineare, a differenza del Pitfall II in versione casalinga: per passare al livello successivo, dovremo completare un quadro raggiungendo una chiave.

Anche in questa nuova avventura, Pitfall Harry dovrà evitare pericoli come tronchi, pipistrelli, scorpioni, palle di fuoco e altro ancora, così come dondolarsi sulle liane per saltare paludi di fango, pozzi di lava, stagni infestati da coccodrilli e altre insidie. Sono presenti anche i famigerati fossi presidiati da coccodrilli, sulle cui teste dovremo saltare nel momento in cui la bocca delle belve sarà chiusa.

Harry è anche in grado di nuotare in corsi d'acqua sotterranei evitando letali anguille e capitoni. Harry deve anche evitare la caduta di frutti, stalattiti e altri oggetti. Le caverne dispongono di diversi piani collegati da scale sotto forma di labirinto. Vengono assegnati punti raccogliendo i tesori.



SCHEDA ARCADE

Hardware: Sega System 1. Game ID: 834-5627. CPU: Z80 a 3,6 Mhz. Processore audio: Z80 a 4 Mhz.
Chip audio: SN76496 a 4 Mhz, SN76496 a 2 Mhz. Orientamento dello schermo: orizzontale. Risoluzione video: 256 x 224 pixel. Aggiornamento video: 60,00 Hz. Palette: 1536 colori. Giocatori: 2. Controllo: joystick a 8 direzioni. Pulsanti: 1. Livelli: 4.

Curiosità

Riprogrammato e distribuito dalla Sega nel 1985. Il gioco è basato sui due titoli della serie creati da David Crane per la Activision su Atari 2600 e altre piattaforme. David Crane è uno dei più importanti game designer di sempre e un fondatore della Activision, la prima software house ad aver sviluppato giochi per hardware di terzi. Un gioco da tavola di Pitfall! fu realizzato nel 1983 dalla MB (Milton Bradley) con licenza ufficiale della Activision. La serie di Pitfall è stata trasformata in un cartone animato dalla Ruby-Spears Productions, prodotto da Joe Ruby e Ken Spears e trasmesso il 17 settembre 1983 nel programma "Saturday Supercade" sulla CBS.

Serie

1. Pitfall! (1982, Atari 2600).
2. Pitfall II - The Lost Caverns (1984, Atari 2600).
3. Pitfall II - The Lost Caverns (1985, Arcade).
4. Super Pitfall! (1987, Nintendo Famicom).
5. Pitfall - The Mayan Adventure (1991, Nintendo Super Famicom).
6. Pitfall 3D - Beyond The Jungle (1998, Sony PlayStation).
7. Pitfall - The Lost Expedition (2004, Sony PlayStation 2).

Conversioni

Console: Sega SG-1000.
Computer: Atari 800 (1983/1984, si tratta del gioco Activision e non della conversione arcade).

di **Adriano Avecone**
e **Andrea Pastore**

GIUDIZIO FINALE

» Giocabilità 95%

Rasenta la perfezione. Intuitivo, vario, controlli perfetti, gameplay reattivo e ricco di mille finezze.

» Longevità 96%

Difficilissimo da finire, ancor più arduo ottenere un punteggio di sorta. Mille segreti e tecniche segrete da padroneggiare. Un classico totale.

PHANTOMAS 2.0

Anno: 2018
Grafica: Jordi Sureda
Codice e Musica: Santi Ontañón
Piattaforma: Amstrad CPC
Genere: Azione

Dopo le meritate vacanze spese tra notti insonni, biberon, cambio di pannolini ed un solo articolo per il numero16, per questo numero 17 ho voluto decisamente fare le cose in grande: ben 3 articoli di cui 2 recensioni di giochi. Inoltre, ho voluto provare nuovamente ad uscire dalla mia zona di comfort per dedicarmi allla recensione di un gioco su una piattaforma a me poco familiare, l'Amstrad CPC.

Sui giochi degli anni 80 e 90 fiumi di inchiostro sono stati versati e possiamo facilmente trovare recensioni per quasi tutto il materiale prodotto (vedi le ottime schede pubblicate periodicamente da Arcadian). Quello che manca, a mio avviso, nel panorama ludico odierno delle macchine 8/16 bit, sono le recensioni dei giochi rilasciati dal 2000 in poi. Spesso si tratta di giochi creati da programmatori indipendenti, alcune volte di prodotti semi-commerciali dedicati ad un ristretto numero di appassionati, quello che pero' accomuna tutti questi lavori e' la passione e la cura con cui sono realizzati.

Phantomas 2.0 e' il rifacimento di un gioco pubblicato nel 1986 dalla Dinamic Software che, come quasi tutti i rifacimenti moderni, migliora decisamente la versione originale: in questo caso dal punto di vista grafico e della fluidita'. Phantomas 2.0, come l'originale, e' un gioco a piattaforme a schermata statica che ci immergera' in un mondo claustrofobico; ci ritroveremeo infatti nei panni di Phantomas, un androide venuto da non si sa dove, a dover vagare nel castello del Conte Dracula alla ricerca delle 3 armi necessarie per sconfiggere il malefico Vampiro. Va da se che le armi ricercate (croce, paletto e martello) sono ben nascoste all'interno del castello e trovarle non sara' affatto una passeggiata (ma com'e' che questi personaggi sprovveduti non si portano mai il necessario da casa? bah...).
A prima vista il gioco ricorda un

Schermata dei crediti

Mi ricorda qualcosa...

Che personaggi strani

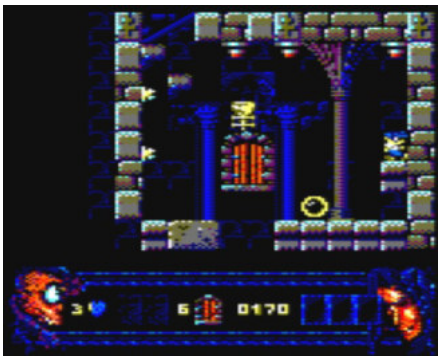
Sempre piu' strano

capolavoro caro a tutti i retrogiocatori, Rick Dangerous, ma le similitudini, purtroppo per Phanomas 2.0, si fermano solo alla prima impressione. Rick Dangerous vanta infatti una grafica pulita e brillante, mentre la grafica di Phantomas e' piuttosto 'impastata'. Attenzione, non sto dicendo che il gioco e' visivamente brutto, soltanto che, dopo aver ammirato la pulizia grafica di Dawn of Kernel in una delle miei precedenti recensioni per Amstrad, giocando con i colori si poteva fare certamente qualcosa di meglio anche qui.

Ma veniamo a quello che personalmente ritengo la parte essenziale di un gioco, la giocabilita'. Phantomas 2.0 e' un gioco simpatico e divertente ma, almeno nel sottoscritto, non ha fatto scattare quella molla che avrebbe fatto si' di giocarlo fino al compimento della missione. Perche'? Bella domanda, cerchiamo di scoprirlo insieme... Le prime partite generalmente servono a capire la meccanica di un gioco, ma in Phantomas 2.0 oltre a prendere familiarita' con il gioco servono anche a capire cosa non fare. Embeh? Nulla di nuovo direte voi... E invece no! Nel primo quadro c'e' un ingresso sotterraneo che potreste essere tentati di prendere, ma che vi portera' inesorabilmente alla morte perche' non ha via di uscita, almeno inizialmente. Poco male, procediamo... In piu' di uno dei quadri successivi ci sono degli oggetti che dovrebbero essere raccolti ma che non possono ancora esserlo perche' devono essere attivati. Ancora, gli interruttori che servono ad aprire le finestre per far entrare la luce del sole nel maniero di Dracula sono veramente difficili da scovare o non li noterete fino a quando non ci sbatterete contro per caso... Piccole cose certo, che pero' minano l'immediatezza di questo titolo e contribusicono a mio avviso a non far innamorare il giocatore al gioco. I giocatori mordi e fuggi sono avvisati, ci troviamo di fronte ad un gioco carino che pero' non perdona nulla. Le vite a disposizione inizialmente sono poche e, anche se non e' difficile trovarne altre, dopo qualche partita in cui dobbiamo ricominciare sempre tutto da capo la tentazione di chiudere il gioco in un cassetto e' alta. Sicuramente un'occasione mancata, perche' con un qualche accorgimento in piu' il gioco sarebbe stato veramente interessante.

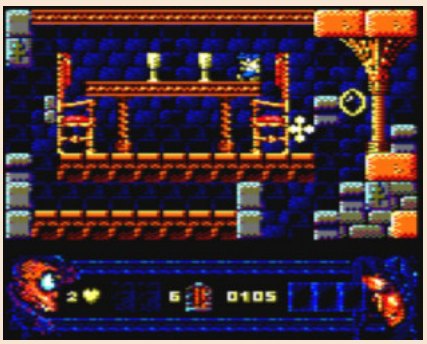
Non e' comunque una bocciatura, gli enigmi da risolvere e la grandezza del gioco sono tali da garantire al titolo una longevita' ed una profondita' invidiabili. Se siete quindi giocatori che amano le sfide ed i giochi difficili da addomesticare, vi trovate di fronte al prodotto che fa per voi. Se invece cercate un gioco immediato che vi tenga impegnati solo pochi minuti al giorno, passate oltre.

di **Francesco Fiorentini**



Per scaricare il gioco inclusi i sorgenti:
https://github.com/santiontanon/phantomas20cpc

Il gioco e' disponibile anche in versione fisica (attualmente esaurita):
http://www.matranet.net/boutique/cpc/phantomas20/phantomas20.php



GIUDIZIO FINALE

» Giocabilità 60%

Un gioco non immediato che potrebbe scoraggiare i giocatori con poco tempo a disposizione (...chi ne ha mai abbastanza?) che con qualche accorgimento in piu' sarebbe stato sicuramente un must.

» Longevità 80%

Peccato, perche'gli enigmi da risolvere e la grandezza del gioco sono tali da garantire al titolo una longevita' ed una profondita' invidiabili.
A quando Phantomas 3.0?

ARMY MOVES

Anno: 1986
Sviluppatori: Dinamic Software, Dinamic Multimedia
Piattaforma: Multi
Genere: Azione

E' passato un anno esatto da quando decisi di scommettere su me stesso scrivendo il primo articolo per RetroMagazine, incrociando le dita nella speranza che il mio primo articolo potesse piacere e risultare interessante. E così fu.
Le vacanze sono appena terminate e spero che voi tutti le abbiate passate bene, anche se di sicuro avrete sentito la mancanza dei giochi e dei vostri computer come la sentivo io quando andavo al mare lasciando inattivo il mio Commodore 64 per ben 20 giorni, senza nemmeno vedere le copertine dei giochi.
In questo Agosto afoso ho avuto il piacere di rigiocare seriamente un altro titolo che mi aveva colpito più per la colonna sonora del ponte sul fiume Kwai che per altro.

Si tratta dello sparatutto a scorrimento Army moves! Da molti fu accantonato e messo in particolar modo sulle cassette da edicola in quanto ritenuto un gioco mediocre e non facile da portare a termine. Ma io ho deciso di rivalutarlo e nonostante la difficoltà ho passato serate intere a giocarlo tra mille imprecazioni ed alla fine l'ho portato a termine! Il gioco come dicevo, è a scorrimento da entrambi i sensi tranne l'ultimo livello a schermate fisse stile labirinto. Ogni livello si svolge a bordo di una Jeep, un elicottero e i successivi a piedi sparando e saltando con precisione millimetrica. La difficoltà del gioco consiste appunto nello sparare in tempo e saltare al momento giusto per quanto riguarda i livelli a piedi e sulla Jeep, mentre con l'elicottero in volo basterà spostarsi come in un classico sparatutto. Basterà prendere la mano per superare i livelli in tranquillità e soprattutto mantenere la calma, non come ho fatto io. :-) eheheh

Il gioco si compone di un totale di sette intensi livelli; per fortuna alla fine del quarto vi verrà dato un codice che vi permetterà di cominciare ogni partita successiva dal quinto! In modo da agevolare e invogliare il giocatore nel proseguimento del gioco. Personalmente ho trovato gli ultimi livelli più scorrevoli e facili dei primi o forse e'soltanto una mia impressione dopo averci smanettato per ore e ore. Oppure i programmatori hanno deciso di invertire le cose, ovvero un gioco duro all'inizio ma facile alla fine; proprio la caratteristica che ha reso questo gioco singolare. L'ho trovata una soluzione migliore al classico aumenta il livello, aumenta la difficoltà. Anche se non ebbe molto successo, ci fu un seguito chiamato Navy moves, esattamente con la stessa difficoltà del primo ed il codice a metà gioco. Ho da poco scoperto che ci fu addirittura un terzo titolo uscito in piena epoca PC MS DOS ma fu una bolla di sapone perché era un titolo più appropriato per il biscottone e probabilmente aveva già dato abbastanza.
Se vorrete giocare Army Moves vi consiglio di non cercare soluzioni e di non guardare i game-play su youtube, cercate piuttosto di studiare i movimenti dei nemici e delle bombe, così supererete ogni livello con facilità. Avete a disposizione cinque o sei vite, a seconda delle versioni e per affrontare il gioco non sono poche, senza contare il fatto che le vincerete anche con il punteggio.

Con questo articolo voglio augurarvi un buon rientro dalle vacanze, buon lavoro o buon anno scolastico per voi o per i vostri figli...
E buon divertimento con Army Moves!

di **Daniele Brahimi**

Versione Commodore 64

GIUDIZIO FINALE

» Giocabilità 75%

Ci vuole 'mano' e soprattutto Precisione Millimetrica per padroneggiare bene il tutto.

» Longevità 80%

Arrivare in fondo al gioco non sara' una passeggiata.
Un po' piu' semplici sono i livelli con i mezzi quali la Jeep e l'elicottero.
Vi dara' filo da torcere.

Versione MS-DOS

HARRIER ATTACK!

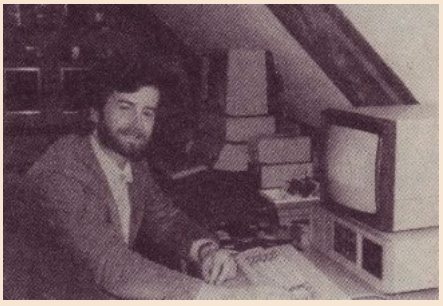
AKA: Bombedero!
Software House: Durell Software Ltd
Autori: Ronald Jeffs (Oric), Mike Richardson (ZX Spectrum/Amstrad CPC), John Parkinson (C64)
Anno: 1983
Genere: Shoot'Em Up
Piattaforma: Oric 16K

La software house che pubblicò questo classico dei computer a 8-bit (Durell Ltd) aprì i battenti non esattamente come una mega-azienda dalle grandi risorse finanziarie a disposizione: "Il capitale iniziale era di 100 sterline. Ho cominciato mettendo in piedi una linea di produzione per Oric. Ho preso un Oric-1, una TV in bianco e nero, qualche libro sul linguaggio macchina ed una stampante ad aghi Epson" – dichiarò Robert White in una delle sue prime interviste. "E la stampante fu un investimento a cui pensai e ripensai a lungo, prima di effettuarlo". Robert (con alle spalle una formazione da insegnante d'arte che aveva incrociato i computer nel suo primo lavoro) e sua moglie Veronica si trasferirono dall'Oxfordshire in un'abitazione vicino a Taunton, nel Somerset. "La casa era di proprietà di mia suocera, quindi in quella prima fase non ci dovevamo preoccupare di un mutuo da pagare. Così cominciai a scrivere un programma Assembler per il mio nuovo Oric. Il mio piano era quello di finire rapidamente l'Assembler, venderlo come un tool per programmatori e poi usarlo per scrivere Harrier Attack."

Ma presto si rese conto che doveva realizzare almeno quattro o cinque versioni del gioco che aveva in mente. Il mercato degli home computer nel 1983, nel Regno Unito e nel mondo, era molto variegato e multi-piattaforma. Bisognava raggiungere tutti gli appassionati scrivendo lo stesso gioco per i modelli più diffusi. Robert mise un annuncio nella zona alla ricerca di programmatori e fu così che Ron Jeffs e Mike Richardson si unirono all'avventura della nascente software house. Con la vendita di Harrier Attack! (che fece un po' di scalpore per via del suo scenario – era l'epoca della guerra tra Gran Bretagna e Argentina per la supremazia nelle isole Falkland/Malvinas) la Durell riuscì a trasferirsi dalla casa di Robert ai locali di una spaziosa mansarda di un vecchio edificio affacciato sul castello di Taunton. Mike Richardson aveva già completato metà di un gioco (Jungle Trouble) sul suo Spectrum quando rispose all'annuncio della Durell e fu assunto come programmatore freelance in quota assembly Z80.

Ron Jeffs si dedicò allo sviluppo di Harrier Attack su Oric 16K e in realtà nel tempo ne realizzò due versioni: una prima versione per Oric-1, cui seguì una versione che funzionava anche sul nuovo Oric Atmos. Non ci sono differenze sostanziali fra le due versioni, tranne che per alcuni aspetti grafici della cassetta e per il caricamento da nastro. La prima versione includeva il nastro con due registrazioni "fast" sul lato A ed una "slow" sul lato B, mentre la versione Oric-1/Atmos presentava solo una registrazione "fast" su entrambi i lati, probabilmente la seconda versione sfruttava le routine di caricamento che la nuova ROM dell'Atmos aveva aggiornato.

Spesso è Mike Richardson ad essere ricordato come l'autore del gioco, ma nella realtà, l'idea di Robert White venne concretizzata su Oric da Ron Jeffs e successivamente (o probabilmente almeno parzialmente in contemporanea) portata su ZX



Spectrum e Amstrad CPC da Mike, che impiegò circa due settimane e mezza per la versione Spectrum ed un paio di altre settimane per quella Amstrad CPC. Le due versioni Z80 non differiscono molto fra loro e riescono nell'intento di garantire una buona fluidità ed un'esperienza arcade degna di nota, considerando le limitazioni grafiche delle macchine. Le versioni di Richardson presentano qualche add-on come l'opzione di salvataggio e la hall of fame dei migliori punteggi ottenuti, oltre ad una grafica decisamente più avanzata rispetto a quella della versione Oric. La versione per ZX Spectrum divenne quella più popolare con oltre 17mila copie vendute contro le 10mila circa della versione Oric. Ed è forse per questa ragione che Richardson viene ricordato come autore del gioco, anche se, come abbiamo verificato dalle cronache dell'epoca, la prima implementazione avvenne sul 6502 dell'Oric-1. Fra tutte le versioni pubblicate la Durell vendette qualcosa come 250mila copie di Harrier Attack!

IL GIOCO

La grafica della versione Oric da me giocata presenta un ambiente di gioco abbastanza realistico nella grafica ma i movimenti di scrolling video orizzontale risultano tutt'altro che fluidi e piacevoli. Come la maggior parte dei giochi Durell il gioco funziona in modalità testo mediante l'uso esteso dei caratteri ridefiniti e questo garantisce che l'azione si svolga velocemente ma un po' a scatti. L'uso di uno scenario simile a quello delle isole Falkland fu in seguito negato da Robert White in un'intervista apparsa dopo l'uscita del gioco, ma in quel momento storico, la coincidenza, come detto, giocò a favore della popolarità del titolo sulle riviste e fra i videogamer inglesi. Il concetto alla base del gioco è piuttosto semplice ma coinvolge fin da subito il giocatore nella sfida di decollare col proprio jet da combattimento, sorvolare il territorio nemico evitando aerei, carri armati e navi nemiche e la barriera di fuoco anti-aerea messa in atto dall'avversario. Armati di missili aria-terra e bombe da sganciare sui vari obbiettivi sensibili del nemico, la missione è quella di durare il più a lungo in volo abbattendo le postazioni a terra e raggranellando punti preziosi fino ad arrivare agli edifici. L'obbiettivo di ogni missione è quello di rientrare sulla portaerei prima che finisca il carburante dopo aver triturato quanti più nemici possibili. Personalmente non sono mai stato molto bravo con gli arcade o gli sparatutto e l'uso della tastiera non aiuta (ho usato un Atmos originale per giocarlo e non è facile virare e sparare nello stesso momento). L'uso del joystick era però previsto per la versione Oric, a patto di possedere un'interfaccia ad hoc (Joystick Adapter) e, da quanto riportano alcuni siti web dedicati, 2 joystick, uno per virare, l'altro per sparare.

CONCLUSIONI

Si tratta di un classico fra i titoli retrogame, sicuramente da provare nelle varie versioni a 8 bit (la versione per C64 dispone di movimenti molto realistici del jet in volo). Anche su Oric, nonostante i problemi di caricamento e la responsività della tastiera (oltre all'incapacità cronica di chi scrive) ne viene fuori un'esperienza tutto sommato coinvolgente, degna di essere giocata sia su vecchio hardware che su emulatore. Ai suoi tempi (agli albori del videogioco da casa) era in linea con i titoli che uscivano (pensate a Scramble). Oggi mantiene una certa dose di attrazione, anche se la grafica e soprattutto il sonoro sono ampiamente superati. Esempio tipico di un gioco dalla struttura essenziale ma divertente.

di **David La Monaca (Cercamon)**

GIUDIZIO FINALE

» Giocabilità 68%

Per gli utenti Oric uno dei più giocabili di sempre.
Il gioco presenta una sua narrazione, anche se essenziale, che coinvolge fin da subito.

» Longevità 65%

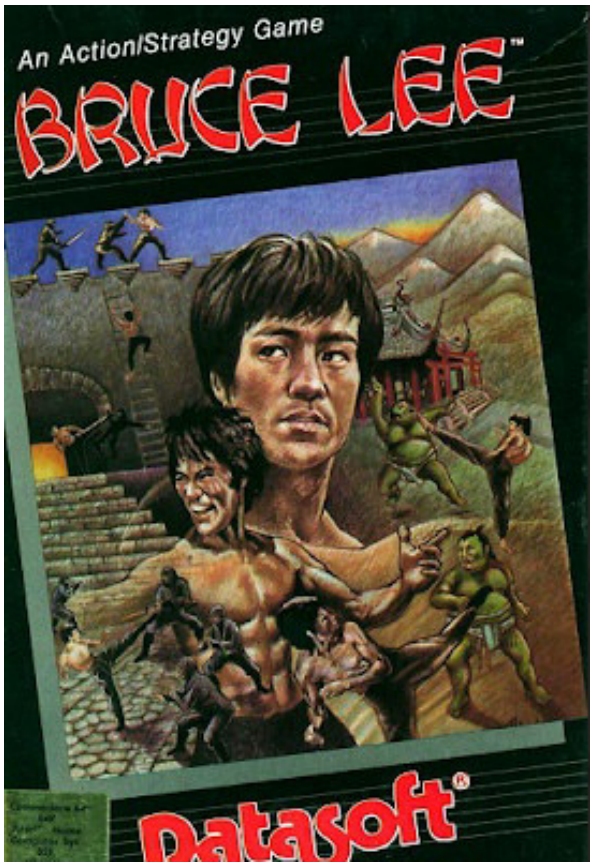
Un vero classico del genere.
Effetto nostalgia garantito.
Vi verrà voglia di rigiocarlo probabilmente di tanto in tanto.
Non proprio tutti i giorni.

BRUCE LEE

Articolo originale a cura di
FRGCB - Finnish Retro Game Comparison Blog

Bruce Lee (Datasoft Inc., 1984)
Versione originale di Ron J. Fortier e Kelly Day.

Pubblicato per:
Apple II, Atari 800, Commodore 64 and MS-DOS da Datasoft Inc. nel 1984
Amstrad CPC, BBC Micro and ZX Spectrum da U.S. Gold nel 1984
MSX da Comptiq nel 1985
Sharp MZ-800 da VSetin (?) nel 1988
NEC PC-8801 (e Sharp X-1?) da Comptiq nel 1989
Sega Master System da Kagesan nel 2015



Descizione del gioco
Bruce Lee e' un 'platform action-adventure game' che rompe alcuni schemi classici di questi generi; uno fra tutti, e' stato il primo gioco ad unire il genere platform con il picchiaduro.

La tua missione e' quella di guidare Bruce attraverso 20 schermi che dovrebbero rappresentare la torre dove vive un mago malvagio; anche se personalmente l'ho sempre considerata un castello. Durante il percorso ci saranno soltanto 2 nemici che ti affronteranno, cercando di mettere fine alla tua avventura: il Green Yamo ed il Ninja. Oltre a loro, ovviamente, ci sono anche diversi ostacoli sparsi un po' ovunque. Quando finalmente arriverari alla fine ed avrai sconfitto il malvagio Fire Wizard riceverai in premio: salute eterna, il segreto dell'immortalita', la principessa, le ali di Icaro, il punteggio piu' alto, i tuoi nonni morti saranno riportati in vita ed anche il tuo cane che fu arrotato dal vicino... e via discorrendo. Dopodiche' l'avventura ricomincera' da capo, soltanto con un livello di difficolta' maggiore. Bentornati nel 1984!
Il gioco offre anche la possibilita' di giocare in doppio con 2 differenti modalita'. In una giocherete a turni, competendo per il punteggio piu' alto, nell'altra il secondo giocatore controllera' il Green Yame cercando in tutti i modi di fermare Bruce.

Ho sempre considerato questo gioco uno dei piu' eleganti ed importanti lavori della storia dei videogiochi, accompagnato da numerosi elementiche raramente si riscontravano nei giochi di quel periodo. E' semplice da completarsi quando non si ha altro da fare e, a distanza di piu' di 30 anni, e' capace ancora di catturare l'attenzione.

Nel 1984 non molti giochi rappresentavano le arti marziali e molti giochi platform contemplavano l'uso di armi; quei pochi che lo facevano si limitavano ai fulmini e poco piu'. Bruce Lee e' stato probabilmente il primo platform adventure ad offrire qualcosa che assomigliasse alle arti marziali come mezzo per affrontare i nemici. Non ha importanza che non fosse esattamente realistico, si e' trattato comunque di una svolta.
Il gioco ti permette inoltre di utilizzare degli oggetti che, anche se non riconoscerai, imparerai ad usare correttamente per preseguire nel gioco. Imparerai a giocare d'astuzia attirando i nemici affinche' si uccidano da soli nelle trappole. Imparerai l'importanza del tempismo dei movimenti. Tutto questo e molto altro ma probabilmente non lo noterai nemmeno.

Caricamento
Ecco i tempi di caricamento di Bruce Lee su ogni macchina per cui fu rilasciato. Tempi calcolati in base alla versione originale del gioco.

ACORN BBC - Cassetta: 4 minuti 14 secondi
AMSTRAD CPC - Cassetta: 3 minuti 49 secondi
ATARI 8-BIT - Disco: 28 secondi
ATARI 8-BIT - Cassetta: 11 minuti 56 secondi
APPLE II - Disco: circa 30 secondi (su emulatore)
COMMODORE 64 - Disco: 2 minuti 54 secondi
COMMODORE 64 - Cassetta: 4 minuti 32 secondi

DOS - HD - immediato, eccetto il tempo di avviare il PC.
MSX - ROM - immediato, eccetto il tempo di avviare l'MSX.
MSX - Cassetta: 3 minuti 7 secondi (2400 baud) / 5 minuti 41 secondi (1200 baud)
NEC PC-8801 - Disco: circa 16 secondi (su emulatore)
SHARP MZ-800 - Cassetta: 3 minuti 8 secondi (1200 baud)
ZX SPECTRUM - Cassetta: 3 minuti 2 secondi

Come si puo' notare il vincitore e' la versione Cartridge dell'MSX, ma solo perche' il PC impiega piu' tempo ad avviarsi rispetto all'MSX.

Giocabilita'
La versione **ATARI** fu sviluppata contemporanemante alla versione del C64 quindi la giocabilita' non differisce di molto dall'originale. L'unica pecca e' che la versione Atari e' un po' troppo lenta per i miei gusti. Il gioco, gia' sufficientemente semplice di suo, diventa noioso se rallentato. Per il resto non possiamo lamentarci di nient'altro; tutto funziona come nella versione originale. Tutti i personaggi si muovono in modo appropriato negli ambienti di gioco, ogni ostacolo è programmato per muoversi in modo che tu possa facilmente seguire il suo ritmo e non è necessario arrampicarsi o correre inutilmente a meno che tu non voglia davvero farlo. E' soltanto un po' lenta.

Come ho gia' detto, la versione **C64** gira molto più velocemente di quella Atari e si adatta meglio al mio stile di gioco. Qui, tutto è in perfetto equilibrio. La corsa sembra frenetica come se le superfici fossero di marmo; sembra si' naturale, ma in modo un po' strano. I movimenti degli ostacoli sotto la superficie sono temporizzati in modo leggermente diverso rispetto alla versione Atari, ma queste sono praticamente le uniche differenze evidenti.

In quanto a giocabilita' la versione **MS-DOS** e' molto simile a quella del C64, fa solo un po' strano giocarla con un vecchio joystick analogico. I risultati migliori si ottengono giocandola con la tastiera, basta soltanto prenderci la mano.

La versione **SPECTRUM** ha piu' o meno la stessa velocita' della versione C64, ma poiché non è stata creata dal team originale, differisce un po' in alcuni aspetti grafici. Prima di tutto il gioco inizia da un punto diverso, anche se nella stessa schermata: niente di che, ma richiede qualche mossa aggiuntiva. In secondo luogo Bruce non si attacca facilmente ai bizzarri tappeti usati come ascensori e tuttora non ho ancora capito bene come fare. Terzo, Bruce cade un po' più velocemente, il che è abbastanza carino. La quarta e probabilmente la più significativa è che Green Yamo può salire le scale mentre nella versione originale non puo' farlo. Non so se questo particolate renda il gioco migliore, dipende dai gusti. Inoltre ci sono alcune piccole differenze nell'ambientazione e nel posizionamento degli ostacoli in alcuni schermi ma nulla che possa fare la differenza nel gameplay. Piccolezze per farti notare che non è esattamente lo stesso gioco, ma ci va abbastanza vicino.

La versione **AMSTRAD** a prima vista sembra un miscuglio tra la versione originale e la conversione dello Spectrum. Certamente ha elementi da entrambi: la grafica è quanto più vicino possibile all'originale, ma il gameplay è un miscuglio di tutto ciò che hai letto finora. Il tuo personaggio si muove in modo più simile alla versione Spectrum rispetto all'originale, ma Green Yamo sembra molto più lento che in qualsiasi altra versione. In alcuni punti, il terreno sembra essere un po 'incasinato e ti viene richiesto di fare più salti per spostarti ovunque. Non ne ho parlato prima, ma i combattimenti nelle versioni Spectrum e Amstrad sembrano un po' più rigidi e rapidi da eseguire rispetto all'originale. Questo rende i combattimenti un po' più difficili, perché devi avvicinarti di piu' ai nemici per ucciderli. Non molto agevole per i giocatori, soprattutto nelle fasi avanzate di gioco quando i nemici diventano più forti.
La versione MSX rientra in qualche modo a meta' fra tutte le altre. Il movimento del giocatore non è veloce come sul C64 ma nemmeno lento come su Atari e Amstrad. Tuttavia la versione MSX ha alcune strane caratteristiche nei movimenti che mi hanno un po' spiazzato. Il salto di Bruce inizia lentamente, come la sua velocità di corsa, ma a mezz'aria accelera improvvisamente. Rispetto alle altre versioni il tempo di recupero dei nemici è inesistente al primo livello di difficoltà. Inoltre, le tempistiche degli ostacoli posti sotto al terreno, hanno delle tempistiche molto diverse. E' sicuramente un po' più difficile come impostazione predefinita.

La versione **APPLE II** dal punto di vista del gameplay potrebbe essere considerata la più vicina alle versione Amstrad e MSX, ma i colori una vera pena per gli occhi. Non riesco proprio a guardarli per più di un paio di minuti. I controlli sembrano un po' a scatti anche se e' non esattamente la peggiore delle versioni.

La versione **BBC Micro** a prima vista è simile alla versione dello Spectrum anche se tutto il movimento del personaggio è innaturalmente rigido. Si muove tutto abbastanza velocemente, il che è un vantaggio, ma tutto il resto è così spento che fatico davvero a descriverlo correttamente. Dimenticavo, il Ninja o è completamente mancante, oppure è nero contro nero.

Per quanto riguarda la giocabilità la versione **NEC** non si discosta molto dagli originali. Il movimento del personaggio principale e' simile all'originale, mentre il ninja sembra essere stato rimpiazzato da un sostituto più lento. La grafica ci mette veramente del suo nel rendere il gioco il meno giocabile possibile con tutti gli sfarfalli e la pessima scelta dei colori; devi avere veramente un'alta resistenza per giocarlo per più di 5 minuti di seguito. Lo sfarfallio inoltre ha un certo effetto sul gameplay, rende di fatto piu' difficile combattere il Green Yamo perche' i personaggi tremolano come l'inferno mentre sono in movimento e il rilevamento delle collisioni nei combattimenti dipende essenzialmente dal punto di contatto degli sprite. Per riuscire bene devi anticipare leggerment gli attacchi. L'eccessivo sfarfallio tende comunque a rendere il tutto piu' difficile. Per rendere la versione NEC accettabile da un pusto di vista visivo prova l'antico trucco Spectrum: usa la modalita' scala di grigi.

La versione **MZ-800** è fondamentalmente un porting economico della versione SPECTRUM. La giocabilita' e' molto simile, soffre di alcuni gravi problemi di velocità

quando la quantità di sprite aumenta sullo schermo.
Tra le due versioni originali, C64 ed Atari, la versione C64 e' decisamente la migliore, non solo perché il gameplay è molto equilibrato e funziona come previsto dai programmatori ma perché è estremamente divertente giocare con la velocità con cui e' settata. La versione peggiore è indubbiamente quella BBC.

1. COMMODORE 64
2. IBM-PC COMPATIBLES
3. ATARI 8-BIT
4. APPLE][
5. MSX
6. AMSTRAD CPC
7. NEC PC-8801
8. ZX SPECTRUM
9. SHARP MZ-800
10. ACORN BBC MICRO

GRAFICA
Attenzione alle varie schermate: sono editate malamente e messe insieme con MS Paint. La maggior parte di esse servono solo a darvi un'idea di come appare il gioco sulle varie piattaforme.

SOPRA: Spectrum, Apple II - SOTTO: MS-DOS, BBC Micro

SOPRA: Atari, Spectrum - SOTTO: Commodore 64, MSX

La prima schermata che vi dà il benvenuto è di solito quella di caricamento, a meno che non si tratti della pagina di copyright/diritti d'autore, ma tralasciamo pure questo discorso. Le versioni Spectrum e Amstrad mostrano la rappresentazione più fedele della grafica di copertina della confezione, mentre le altre presentano un approccio più semplice e ravvicinato dell'immagine del protagonista stesso del gioco. Qui di seguito si possono vedere anche le schermate di caricamento del NEC, sul quale, in modo piuttosto stravagante, ne appaiono ben tre, ma di certo la cosa non mi fa impazzire. Direi che lo screenshot di caricamento dello Spectrum vince il primo premio, ma anche la versione per Apple II è ben realizzata.

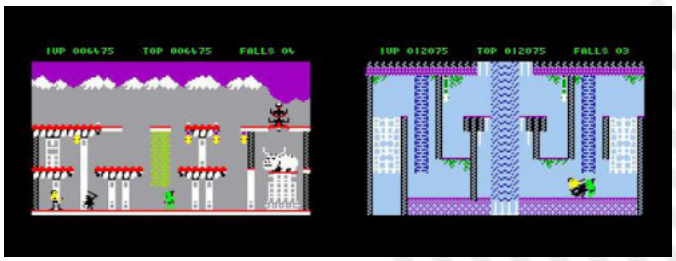
Schermate di caricamento su NEC PC-8801.

Purtroppo la schermata di caricamento non ha conseguenze sul punteggio finale assegnato a ciascuna versione per determinare quale sia la migliore. Quello che invece influisce è la quantità di nostalgia che suscita in ciascun retro-giocatore. =)

Atari 800

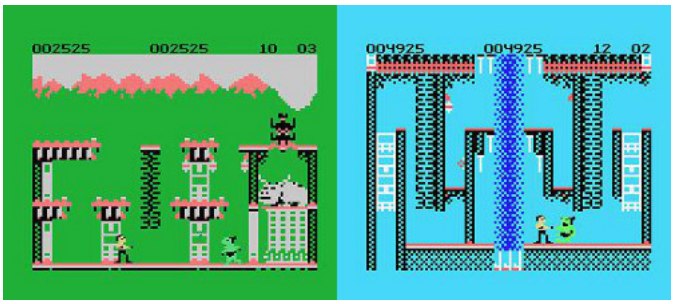
Commodore 64

ATARI/C64: grafica molto simile fra le due versioni, tranne che per la palette dei colori leggermente diversa e per il diverso font di sistema; in entrambi i casi si tratta solo di una mera questione di preferenze. Mi piace un po' di più il font di sistema Atari, ma per il resto a me sembra tutto praticamente identico. Ai miei occhi, però, la schermata finale sembra più gradevole sul C64.



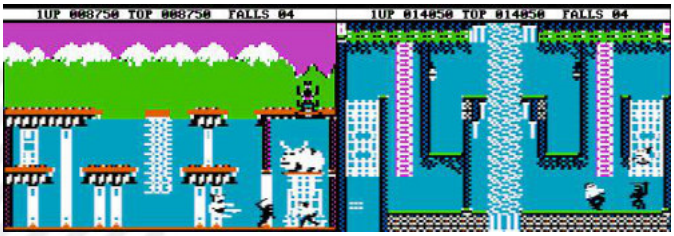
AMSTRAD CPC 464: La versione Amstrad sembra abbastanza simile agli altri originali, a parte qualche strana scelta di colori, ma invece di usare il font di sistema per la barra del punteggio i programmatori hanno deciso di creare un font apposito, che però appare un po' inadatto al gioco. Ad alcuni potrebbe anche piacere di più, ma io credo che in qualche modo dia meno risalto allo schermo in cui avviene l'azione. Non so perché, ma questa è solo la mia

opinione.

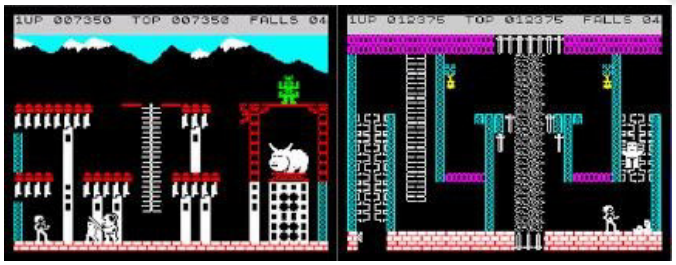


MSX: La grafica dell'MSX risulta un po' sfocata, soprattutto a causa delle dimensioni dello schermo, ma la scelta dei colori è semplicemente bizzarra. Lo sfondo, per esempio, è diverso in ogni area in cui potrei trovarmi a giocare: lo sfondo del palazzo è verde, la prima sezione sotterranea è color ciano (stranamente più luminoso che in precedenza), sotto è grigio chiaro (e questo in realtà è un po' meglio dell'arancione originale). Inoltre, a seconda del modello di MSX che viene utilizzato, il giocatore sperimenterà una quantità variabile di sfarfallamento dei personaggi. Ad esempio negli screenshot qui sopra anche il personaggio del ninja dovrebbe apparire. I colori dei personaggi sono abbastanza strani, ma tutto sommato inalterati rispetto all'originale.

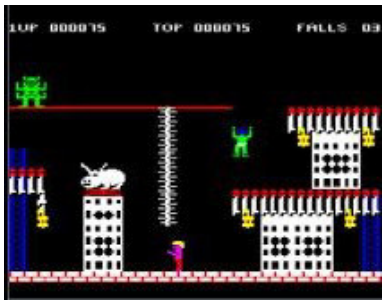
MS-DOS: L'inevitabile versione MS-DOS si presenta nell'orribile versione a 4 colori di default della grafica CGA (ciano, magenta, bianco e nero). Oltre ai colori, la versione per MS-DOS è perfettamente a posto e si avvicina in ogni altro aspetto alla qualità delle versioni Atari e C64. Per qualche strana ragione, la mia versione di Bruce Lee su MS-DOS sembrava avere 79 vite all'inizio del gioco, ma ciò non è molto importante…



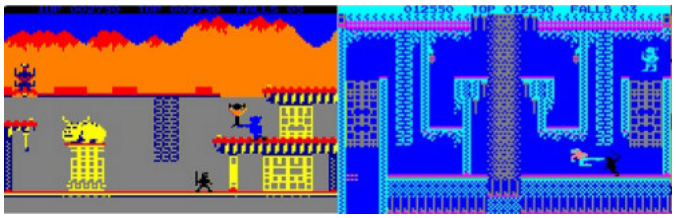
Apple II: Qui davvero la grafica comincia a diventare strana: la versione Apple II mostra un cielo completamente viola, montagne verdi e pareti blu nella prima area del gioco. Non viene certo coglia di giocare oltre la seconda stanza sotterranea. Tutti i personaggi sono in bianco e nero (o nel caso del ninja solo nero). L'animazione dei personaggi va un po' a scatti. Potrebbe trattarsi solo di un problema con l'emulatore, ma ho la sensazione che si tratti in realtà di una caratteristica riscontrabile solo su Apple II. Anche il font appare un po' fuori posto, ma suppongo che non ci si possa fare molto.



ZX Spectrum: La versione per Spectrum mostra una grafica molto prossima alla sgradevolezza, ma presenta alcune caratteristiche di riscatto. Anche se alcune delle strutture e delle decorazioni sembrano più interessanti e colorate, non si tratta in realtà di un vero miglioramento, perché ogni sfondo è nero e tutti i personaggi sono bianchi. Naturalmente questo serve a ridurre al minimo la possibilità di colour-clash, ma se c'è un personaggio chiamato Green Yamo, allora questo dovrebbe ragionevolmente essere verde, accidenti! Comunque, a parte le critiche, il gioco non è niente male quando lo si affronta – tutto è riconoscibile rispetto all'originale e tutti i personaggi si comportano in modo abbastanza preciso. Ma la sensazione generale è che questa versone sia un po' troppo lontano dall'originale per essere considerato accettabile.

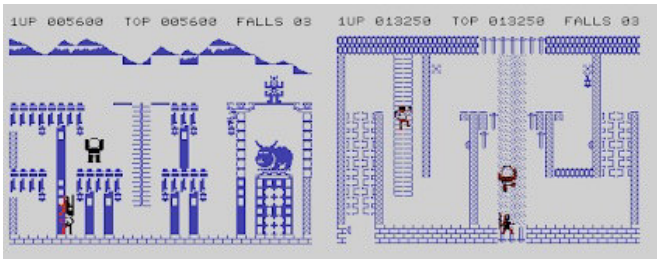


Acorn BBC Micro: Se state cercando qualcosa di peggio della versione Spectrum, non dovete andare oltre quella per Acorn BBC Micro. Lo schema dei colori è simile alla versione Spectrum in tutto e per tutto, ma i personaggi dei giocatori sono colorati – sono i colori sbagliati, d'accordo, ma sono pur sempre colorati. Beh, in effetti, è un po' peggiore della versione Spectrum perché i personaggi non fanno altro che sfarfallare, quindi i vostri occhi inizieranno a dolervi prima di arrivare al terzo schermo.



NEC: Da qualche parte in mezzo ai due precedenti si può collocare la versione per NEC. Sostanzialmente il gioco per NEC appare abbastanza simile all'originale ma lo schema cromatico è stato probabilmente ideato da una persona estremamente malvagia.
Giù nella prima area da affrontare le montagne sullo sfondo sono orrendamente tinte d'arancione e rosso. Per lo meno il personaggio di Bruce Lee è colorato di giallo ma il suo busto sembra galleggiare sulle montagne. Il nostro protagonista salta e corre in modo mirabilmente veloce, proprio come dovrebbe e anche il ninja è nero, una buona cosa. Il Green Yamo è sorprendentemente

reso in nero e blu e ciò lo fa apparire come un'indescrivibile massa scura gelatinosa. C'è poi il tremolio, già menzionato in precedenza - questo rende la versione NEC quasi altrettanto terribile quanto la versione BBC. Quando si arriva alla prima zona sotterranea il gioco si trasforma improvvisamente con un'orribile grafica in stile CGA: ciano, magenta, bianco e nero. Dato che ci sono così tanti oggetti animati attorno, il movimento del protagonista diventa dolorosamente lento e questo lo trasforma in un Bruce di colore bianco-ciano. Il personaggio di Green Yamo sembra almeno più verde ora nel suo aspetto ciano/ciano-interlacciato ed il ninja è ancora nero. Per il resto si tratta più o meno del gioco che dovrebbe essere, solo decisamente lento e doloroso per gli occhi. La terza sezione, tuttavia, è assolutamente puro inferno. Letteralmente. Ci sono diverse tonalità di colore per rendere al meglio il fuoco e lo zolfo, tutto per la gioia dei vostri occhi e non si capisce cosa stia succedendo, perché le sfumature di colore sono così vicine l'una all'altra che bisogna davvero concentrarsi su quello che si muove sullo schermo. Se non altro le piattaforme principali sono colorate di nero, ma gli oggetti in movimento sono intrisi di rosa su sfondo arancione. Inguardabile. Si tratta di qualcosa di semplicemente malvagio e da qui in avanti



proprio non si riesce ad andare oltre.
SHARP MZ-800: Non sono sicuro che l'assenza di variazioni di colore sia una valida alternativa ad avere colori orrendi ma la versione SHARP MZ-800 presenta solo una grafica in blu, nero e rosso su sfondo grigio chiaro. A rendere il mio lavoro più facile è il fatto che per il resto questa versione ha esattamente lo stesso aspetto della versione Spectrum.

Tirando le somme, la versione Atari sembra la migliore, ma è solo una questione di gusto - i primi 4 della classifica qui sotto vanno benissimo e se riuscite a liberarvi dai vincoli dello schema cromatico originale, anche la versione Spectrum non è poi così male.

1. ATARI 8-BIT
2. COMMODORE 64
3. AMSTRAD CPC
4. MSX
5. MS-DOS / IBM-PC COMPATIBILI
6. APPLE][
7. ZX SPECTRUM
8. SHARP MZ-800
9. NEC PC-8801
10. ACORN BBC MICRO

SONORO
Stranamente nella versione Atari la parte sonora appare programmata in modo un po' troppo casuale coi suoi beep e boop sparsi qua e là. Non c'è nessuna sigla o brano tematico da sentire all'inizio per l'utente Atari e al suo posto bisogna sopportare 11 minuti di beep durante il caricamento da nastro, prima di ascoltare un po' più di boop casuali durante il gioco. Non esattamente quello che speravo, specialmente dopo aver ascoltato la colonna sonora della versione C64 ed aver visto la versione Atari solo una volta quando ero molto giovane, che mi aveva lasciato una forte impressione positiva.

Il C64 con il suo chip SID fa esattamente quello che ci si aspetta: creare un'atmosfera sonora che vi trascina nel gioco momento dopo momento. Anche se non è l'unico del gruppo di macchine in esame che raggiunge questo risultato, di certo alza l'asticella della qualità con un numero pazzesco di effetti sonori melodici, la migliore resa del brano tematico di Bruce Lee e persino un mix di inni nazionali e altra roba niente male nella versione cassetta. Non c'è bisogno di ascoltare musica durante il gioco visto che gli effetti sono già abbastanza musicali.
La versione più vicina alla pura maestria del C64 è sorprendentemente la versione MS-DOS. Persino con un beeper scadente, si possono ascoltare un sacco di suoni che non cercano affatto di imitare i suoni del C64, ma c'è qualcosa di decisamente particolare, un fascino unico ed un certo spessore nel sonoro di questa versione. Anche la sigla è molto vicina all'originale. Pollice su.
La versione Apple II cerca di imitare quella MS-DOS, ma risulta per lo più fastidiosa e produce alcuni timbri sonori allucinanti che vi faranno desiderare di uccidere il gatto del vostro vicino di casa.

Le versioni Amstrad ed MSX rappresentano qui la terra di mezzo e nel complesso sono abbastanza legate alla versione Atari. Niente di così accattivante, ma abbastanza buono per darvi la sensazione di giocare a Bruce Lee. A differenza della versione MSX, almeno la versione Amstrad presenta il tema musicale integrale.
In modo sorprendente, la versione NEC presenta un insieme molto variegato di suoni, cosa che lo colloca a metà classifica. Solo quando Yamo fa il suo ingresso sullo schermo, il suono emesso sembra un grande rutto invece dello "YAWP" che è il suo marchio di fabbrica o come lo si vuole definire.

Anche la versione Spectrum include il tema del gioco in qualche modo integro, ma la colonna sonora è così confusa che è quasi irriconoscibile. Gli effetti sonori che si possono notare sono principalmente i suoni durante l'azione del gioco, beep casuali quando si raccolgono gli oggetti che brillano sospesi ed effetti relativi alla morte dei personaggi. I suoni relativi al combattimento sono completamente ignorati qui. La versione SHARP MZ-800 presenta suoni dello stesso livello, per ragioni abbastanza ovvie.
Il BBC Micro si prende tutta la torta nella competizione dei suoni meno suggestivi. In realtà tutto si riduce a nient'altro che alcuni ticchettii di due toni (camminare e colpire il nemico) e poi la melodia della morte dei personaggi, che non si adatta molto bene al resto.
Il chiaro vincitore di questa categoria è quindi il C64.

1. COMMODORE 64
2. MS-DOS / IBM-PC COMPATIBILI
3. AMSTRAD CPC

4. ATARI 8-BIT
5. NEC PC-8801
6. MSX
7. APPLE][
8. ZX SPECTRUM / SHARP MZ-800
9. ACORN BBC MICRO

CONCLUSIONI

Tutto sommato, tenendo conto anche della disponibilità della macchina scelta nel paese in cui si vive e naturalmente il suo prezzo, si dev'essere pazzi a scegliere un computer sulla base di un singolo gioco. Ma se si dovesse scegliere in base ad un solo gioco, beh, si potrebbe fare molto peggio che farlo attraverso la bontà di un gioco come Bruce Lee, dal momento che stiamo parlando di un vero classico.

Ricordo di aver provato questo gioco su 4 diverse macchine in gioventù: prima la versione Atari, poi Spectrum, poi C64 e infine quella per Amstrad CPC. Già allora, la versione C64 appariva chiaramente la migliore, proprio per via della colonna sonora. Ma per effettuare un confronto migliore nel 2013, ben 30 anni dopo la sua uscita, ho pensato che sarebbe stato giusto almeno testarlo su tutte le piattaforme per cui era stato pubblicato e rendere noti i risultati sul blog.

Sebbene solo da un punto di vista matematico, ecco il risultato finale:

1. COMMODORE 64 - Giocabilità 10, Grafica 9, Sonoro 9 = TOTALE 28.
2. ATARI 8-BIT - Giocabilità 8, Grafica 10, Sonoro 6 = TOTALE 24.
3. MS-DOS / IBM-PC COMPATIBILI - Giocabilità 9, Grafica 6, Sonoro 8 = TOTALE 23.
4. AMSTRAD CPC - Giocabilità 5, Grafica 8, Sonoro 7 = TOTALE 20.
5. MSX - Giocabilità 6, Grafica 7, Sonoro 4 = TOTALE 17.
6. APPLE II - Giocabilità 7, Grafica 5, Sonoro 3 = TOTALE 15.
7. NEC PC-8801 - Giocabilità 4, Grafica 2, Sonoro 5 = TOTALE 11.
8. ZX SPECTRUM - Giocabilità 3, Grafica 4, Sonoro 2 = TOTALE 9.
9. SHARP MZ-800 - Giocabilità 2, Grafica 3, Sonoro 2 = TOTALE 7.
10. ACORN BBC MICRO - Giocabilità 1, Grafica 1, Sonoro 1 = TOTALE 3.

Naturalmente si tratta sempre di una questione di opinioni e di preferenze di ciascuno, dello stile di gioco, del fattore nostalgia e di tutto il resto. Anche dopo aver effettuato tutti i conteggi, le mie versioni preferite di Bruce Lee restano quelle del C64, Atari e Spectrum. Proprio perché è solo aritmetica, non vi aspettate che rappresenti la verità. Questo è il mio punto di vista, ma in un certo senso spero che rispecchi anche l'opinione dell'ideatore originale. =)

Nota: Potreste essere venuti a conoscenza di almeno un altro remake di Bruce Lee, quello per Sega Master System. Una piccola bella sorpresa giunta nel 2015, sicuramente degna di essere vista e testata, anche se non è completamente fedele all'originale.

Schermate del remake per Sega Master System uscito nel 2015.

Un'altra bella sorpresa è arrivata a tutti i fan di Bruce Lee nel 2013, quando Bruno R. Marcos ha pubblicato il suo sequel non ufficiale, ma pur sempre un vero e proprio sequel, semplicemente intitolato Bruce Lee II. Il gioco presenta un'area molto più ampia del precedente in cui muoversi, un numero maggiore di avversari contro cui cavarsela e di enigmatici rompicapo. Presenta inoltre modalità grafiche autentiche e ben realizzate per i fan di C64 e Amstrad. Per noi gente comune, il gioco sembrava enorme e complesso abbastanza da non poter essere implementato su un C64 reale o sull'Amstrad, ma – Udite! Udite! – il lavoro di conversione per C64, cominciato a metà del 2014 da Jonas Hultén, ha visto la luce nel 2015! Il gioco è disponibile adesso sul suo sito web dedicato in vari formati diversi. Non posso che consigliarvi caldamente di scaricarlo e provarlo! Ne vale la pena.

Quella che state leggendo e' una traduzione autorizzata dell'articolo originale pubblicato da **FRGCB Dude** sul suo blog **FRGCB** all'indirizzo: http://frgcb.blogspot.fi/2013/08/bruce-lee-datasoft-inc-1984.html

Colgo l'occasione per ringraziare pubblicamente **FRGCB Dude** per aver concesso a RetroMagazine l'autorizzazione a pubblicare parte del suo lavoro.

Traduzione a cura di: **David La Monaca (Cercamon)** e **Francesco Fiorentini**

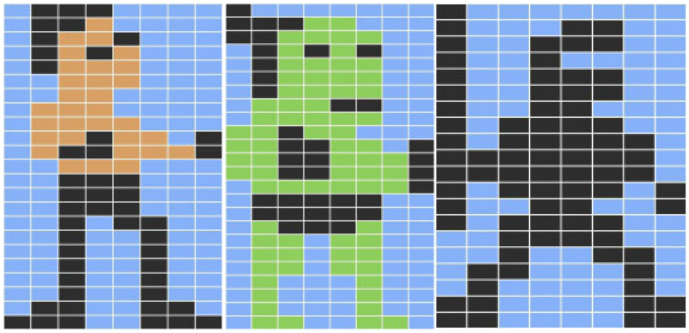
BRUCE LEE

UN GIOCO “SU MISURA” PER ATARI 8-BIT

Anche se il programmatore Ron J. Fortier non lo avesse dichiarato (vedi Milne R., Bruce Lee, in Retro Gamer, Imagine Publishing Ltd, n. 145, agosto 2015, p. 40-43), sarebbe evidente che progettò Bruce Lee (1984) per le macchine del compianto Jay Miner, commercializzate da Atari a partire dal 1979.

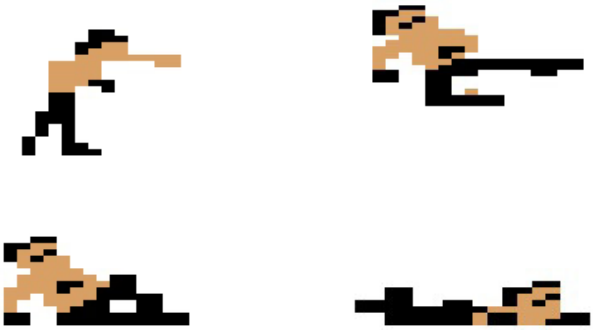
Infatti un occhio abituato alle modalità grafiche di questi computer riconosce immediatamente che la schermata del gioco è composta da una riga di caratteri con risoluzione 320x200 pixel seguita da 11 righe di caratteri con la peculiare risoluzione 160x100.

Ma l’aspetto che più fa capire quanto il gioco sia stato confezionato “su misura” sono gli sprite. Gli Atari dispongono di 4 sprite (denominati “players”) monocolore da 8 pixel orizzontali per 256 pixel verticali e di 4 sprite (“denominati “missiles”) monocolore da 2 pixel orizzontali per 256 pixel verticali; questi ultimi possono essere automaticamente raggruppati andando a formare un quinto sprite monocolore da 8 pixel orizzontali per 256 pixel verticali.



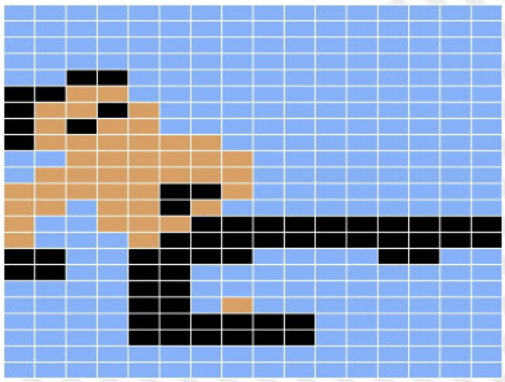
Ebbene, il gioco usa esattamente i cinque sprite disponibili e la loro larghezza: il personaggio di Bruce Lee è composto da due sprite sovrapposti, uno color carne e uno nero, Green Yamo da due sprite sovrapposti, uno verde e uno nero, il ninja dal rimanente sprite nero.

Ma se gli sprite Atari hanno una risoluzione orizzontale di soli 8 pixel, in che modo è stato possibile realizzare i frame più larghi dei due personaggi principali, come ad esempio i seguenti?

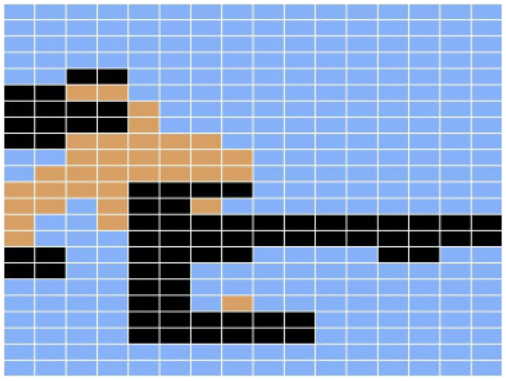


Come la console Atari VCS, anche i computer Atari possono automaticamente raddoppiare o quadruplicare le dimensioni orizzontali dei pixel degli sprite.

Raddoppiare semplicemente le dimensioni avrebbe prodotto un risultato sgradevole. Allora Fortier e il grafico Kelly Day decisero di raddoppiare le dimensioni di uno solo dei due sprite, lasciando invariate quelle dell’altro. Nel caso del calcio volante, le dimensioni dello sprite di colore nero vengono raddoppiate e la perdita di risoluzione viene mascherata dallo sprite color carne sovrapposto, con il seguente ottimo risultato finale.



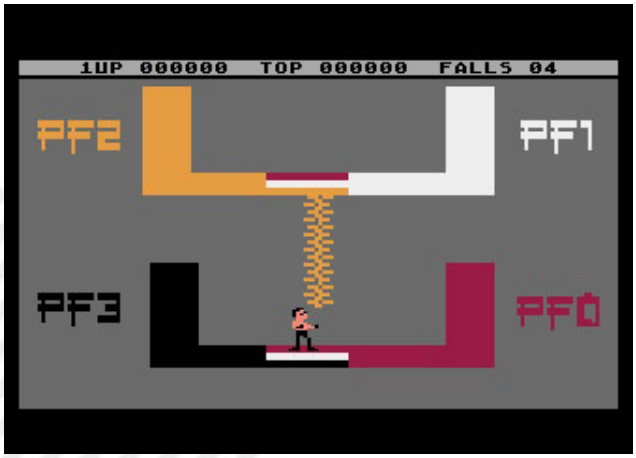
Posizionando lo sprite di colore nero davanti a quello color carne, si raggiungerebbe il seguente risultato, che fa capire come i pixel dello sprite nero siano di dimensione doppia.



Questa tecnica non era nuova; fu utilizzata nel gioco Popeye del 1983, quando il protagonista sferra il micidiale pugno. In questo caso gli sprite sono sempre due, ma è pure attivata l'opzione che fa sì che i pixel sovrapposti assumano un terzo colore, risultato di un OR tra i due colori.

All'inizio abbiamo detto che per lo sfondo è stato impiegato un modo grafico con caratteri dalla particolare risoluzione 160x100 pixel invece della "tradizionale" risoluzione 160x200; come mai? Per risparmiare memoria, visto che fu stabilito che il gioco dovesse occupare al massimo di 32KB.

Un'ultima annotazione tecnica. Il motore del gioco si affida alle collisioni degli sprite con i diversi registri colore: registro 0 (PF0) - pavimento, registro 1 (PF1) – verificare l'evento, registro 2 (PF2) – scala, registro 3 (PF3) – ostacolo.



Dopo la versione per Atari, Fortier programmò quella per Commodore 64; la definisce una sorta di emulazione dell'originale, nel senso che volle il più possibile rimanergli fedele, non solo a livello grafico e di gameplay ma anche di codice. Ad esempio, come spiegato da Vidar "dmx" Bang (Bruce Lee – Return of Fury), invece di usare tre sprite multicolore, Fortier impiegò cinque sprite monocolore. Questa scelta e la maggior velocità del gioco rendono le animazioni meno fluide.

Una domanda alla quale è difficile rispondere è come mai sul C64 la carnagione di Bruce Lee sia gialla. Non si tratta di un problema di palette perché tra i sedici colori della macchina Commodore ci sarebbe un beige. Qualcuno avanza l'ipotesi che in realtà venga indossata la famosa jumpsuit del film Game of Death del 1979 (successivamente usata da Uma Thurman in Kill Bill Volume 1 del 2003). Però la jumpsuit del film è completamente gialla, non ha i pantaloni neri. Inoltre nella copertina del gioco e nelle pubblicità dell'epoca Bruce Lee è ritratto a torso nudo con i pantaloni neri, come nel gioco per Atari.

Infine, visto che torso e viso sono fatti con lo stesso sprite, la faccia risulta irrealisticamente gialla. A tal proposito, c'è chi azzarda che il giallo sia stato scelto di proposito vista l'origine asiatica del protagonista. Il programmatore di Bruce Lee – Return of Fury ha accontentato tutti permettendo la scelta del colore.

Ricordo con affetto Bruce Lee (ho ancora la cassetta dell'epoca). Non sono il solo perché credo sia difficile trovare qualcuno che non conosca il gioco Datasoft e non l'abbia apprezzato.

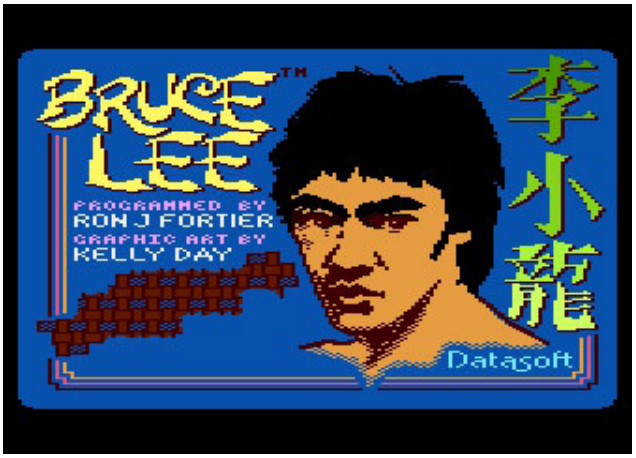
Bruce Lee, miglioramenti grafici e homebrew per Atari 8-bit

Tra i tanti estimatori di Bruce Lee c'è Konstantinos "TIX" Giamalidis, un ingegnere informatico che in passato ha lavorato come artista animatore. La sua abilità è emersa a partire dal 2018 con nuovi sprite per giochi quali ad esempio H.E.R.O., Miner 2049er, Moon Patrol, Pitfall II e Pole Position. Quest'anno ha provato a migliorare lo sprite del protagonista, finendo poi per modificare anche quelli di Green Yamo e del Ninja. Una bella impresa, considerato che si tratta di ben 38 frame diversi, che potete apprezzare nella pagina successiva, sotto quelli originali.

L'impresa è stata complicata dal fatto che alcuni frame sono composti da due sprite sovrapposti, uno dei quali con le dimensioni orizzontali raddoppiate (vedi l'articolo "Un gioco su misura per Atari 8 bit").

Inoltre è stato necessario l'intervento di Fantômas sul codice del gioco per risolvere alcuni problemi di rilevamento delle collisioni e per poter disporre di un frame esclusivamente dedicato alla "morte" del protagonista, visto che nel gioco originale questo frame viene usato anche in altri frangenti.

Ciliegina sulla torta, la sostituzione della schermata iniziale da quattro colori con una versione da 15 colori, realizzata nel 2010 da MrFish.



È possibile scaricare l'ultima versione del lavoro di TIX in questo thread:
https://atariage.com/forums/topic/292947-bruce-lee-retweaked-sprites/

La qualità dei nuovi sprite non è sfuggita a Vidar "dmx" Bang, l'autore del recente homebrew per Commodore 64 Bruce Lee: Return of Fury, che li integrerà in una prossima versione del suo gioco in formato cartuccia.
TIX è instancabile. Oltre ad avere realizzato quasi un centinaio di frame per un gioco che chi vi scrive sta preparando, adesso si sta occupando di Popeye e di GoSub, un nuovo gioco per Atari 7800.

Nel febbraio di quest'anno ho annunciato su AtariAge l'imminente rilascio di Bruce Lee - Return of Fury per C64, ricordando anche l'uscita di Bruce Lee II per PC nel 2013 e per C64 nel 2015.

La notizia e il desiderio di molti utenti di vedere girare questi straordinari giochi anche sulle macchine dove Bruce Lee è nato, hanno ispirato e motivato Fantômas e Ute, che si sono messi al lavoro per esplorare la fattibilità del progetto. Il primo passo è stato quello di effettuare un completo reverse-engineering di Bruce Lee, a seguito del quale è stato possibile rilasciare un editor di livelli (Figura 1).

Considerate le conoscenze acquisite, è probabile che venga realizzato un vero e proprio "Bruce Lee Construction Kit" che permetterà di disegnare livelli con funzionalità aggiuntive rispetto al gioco originale.

Al momento i due programmatori non hanno ancora deciso cosa fare per quanto riguarda Bruce Lee - Return of Fury e Bruce Lee II. Forse il primo verrà convertito oppure sarà realizzato un gioco ex novo. Meno facile che veda la luce Bruce Lee II, tecnicamente più complesso per gli Atari.

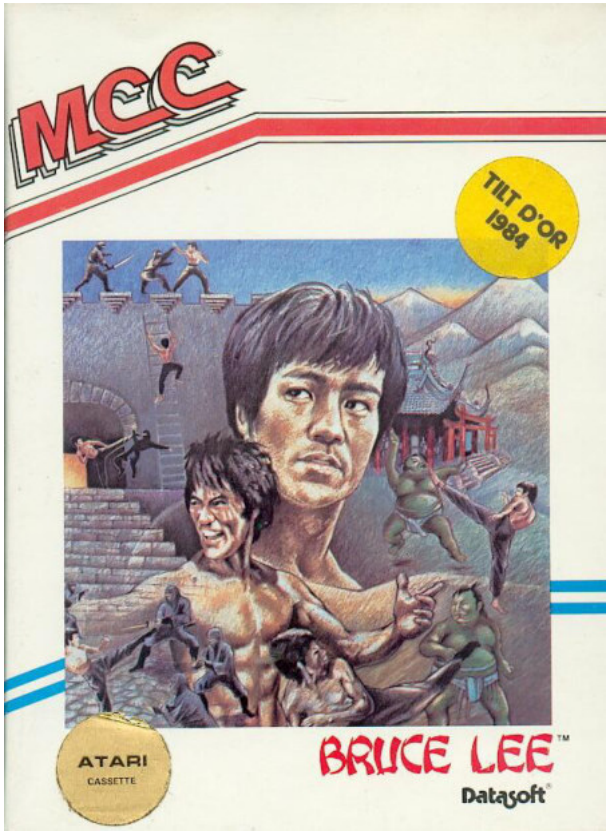
Per una eventuale schermata iniziale, ecco un'immagine da me convertita da 160x240 pixel (overscan verticale completo) e 28 colori.

Potete seguire gli sviluppi in questo thread: https://atariage.com/forums/topic/288392-two-bruce-lee-sequels/

Dimenticavo, Yaron Nir e Vladimir "popmilo" Jankovic stanno lavorando al gioco Sumo Adventure, con protagonista Green Yamo.

Credo che il lottatore di sumo verde si meriti proprio un gioco a lui dedicato, siete d'accordo?

Filippo "Philsan" Santellocco
www.santellocco.com/atari



Risorse

Milne R., Bruce Lee, in Retro Gamer, Imagine Publishing Ltd, n. 145, agosto 2015, p. 40-43

https://atariage.com/forums/topic/292947-bruce-lee-retweaked-sprites
https://atariage.com/forums/topic/288392-two-bruce-lee-sequels/

L'autore ringrazia Konstantinos "TIX" Giamalidis, Fantômas, Vidar "dmx" Bang e Yaron Nir per le informazioni fornite.

www.santellocco.com/atari
https://www.facebook.com/groups/atariworld/
https://atariage.com/forums/topic/176545-topic-for-newbies/
www.santellocco.com/museo

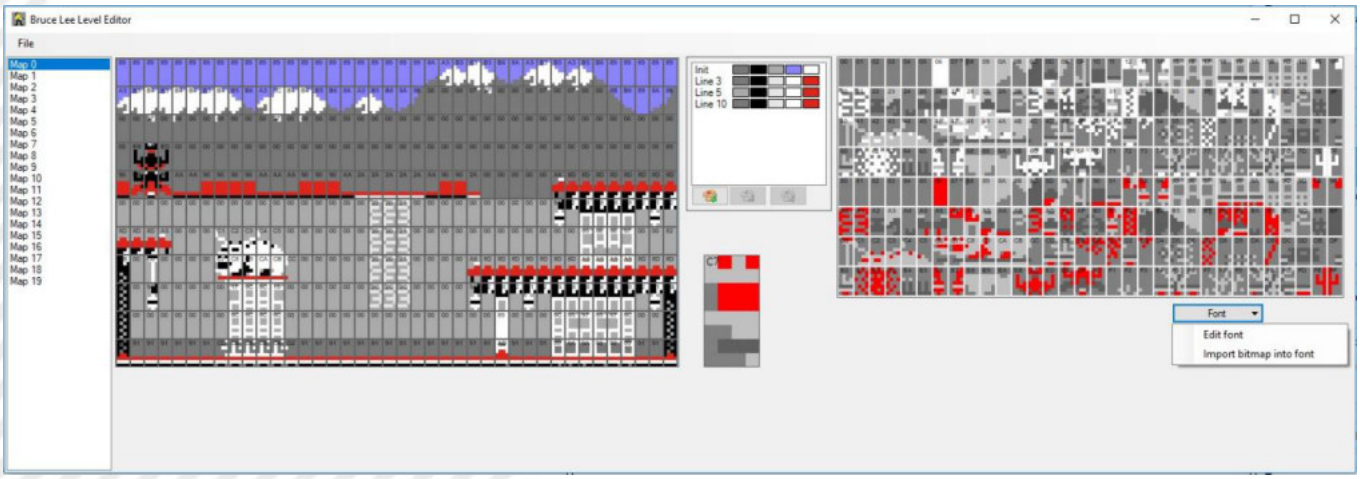
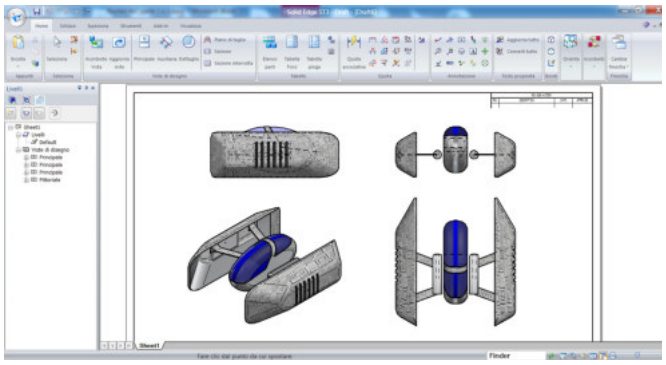


Figura 1 - L'editor di livelli rilasciato in seguito al reverse engineering di Bruce Lee

NUCLEO 447 – DIARIO DI SVILUPPO PARTE 2 DI 2

a cura di Associazione Firenze Vintage Bit Onlus (Leonardo Vettori)

il diario di sviluppo di Nucleo 447 iniziato nel numero 16.



22/10/2018 – Antonio Savona e l'importanza dei colori

Ho avuto un intero fine settimana per sperimentare le variazioni di grafica con il Seuck. Di tutte le prove fatte non me n'è piaciuta nemmeno una.
Forse non ho dato il massimo, ma sicuramente uno dei punti deboli del Seuck è che i blocchi sono troppo grossi , 5 x 5 caratteri, e che quando viene associato un unico colore al blocco è difficile ottenere una variazione di colori nella mappa.
Vi faccio un esempio per farvi capire meglio:
I miei colori fissi sono in ordine di importanza, 1 nero, 2 grigio scuro, 3 grigio chiaro, il 4 è il colore custom.
Se io voglio passare da un paesaggio blu a un paesaggio verde con una certa armonia dovrei:

1) Creare il blocco 1 di colore blu
2) Creare il blocco 2 di colore blu che sfuma sul grigio
3) Creare il blocco 3 di colore grigio che sfuma sul verde
4) Creare il blocco 4 di colore verde

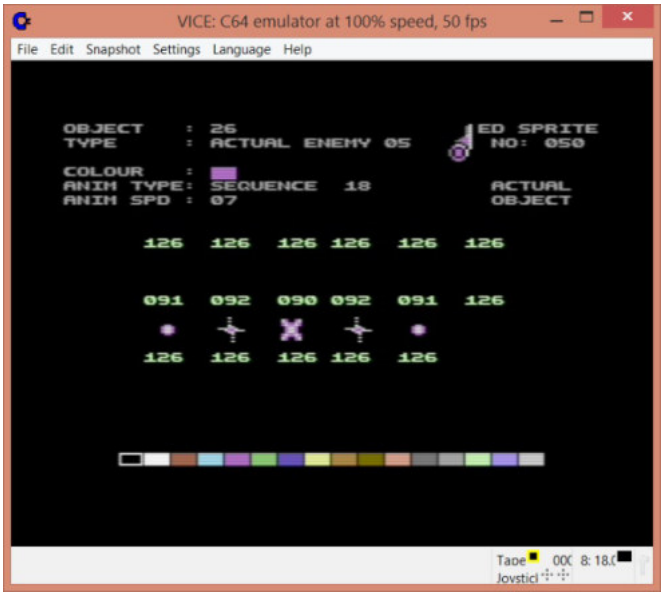
Per ottenere questa variazione dovrei sacrificare diversi blocchi e io li ho già utilizzati quasi tutti, mannaggia della miseria. Alla fine decido di dividere i 4 mondi con alcuni intermezzi di texture tecnologiche, come se queste barriere metalliche servissero per dividere un mondo dall'altro. Inoltre credo che quando inserirò gli sprite animati di colore diverso dai fondali, questi mondi quasi monocromatici prenderanno un po' di "vita" e il giocatore si concentrerà più sull'azione che sul resto. La mappa di gioco sarà composta come segue:

1) 100 blocchi del mondo di cristallo e mostro finale di primo livello
2) Intermezzo tecnologico
3) 100 blocchi del mondo vegetale e mostro finale di secondo livello
4) Intermezzo tecnologico
5) 100 blocchi del mondo di rocce e mostro finale di terzo livello
6) Intermezzo tecnologico
7) 100 di mondo tecnologico con richiami agli altri 3 mondi e il mostro finale.

Preso dalla necessità di sentirmi dire bravo da uno dei guru della programmazione, spedisco la mappa a Antonio Savona che ho conosciuto a Maggio allo Svilupparty a Bologna e gli chiedo cosa ne pensa. Mi risponde dicendo che la grafica gli piace moltissimo ma scrive anche che: una cosa che non mi ha convinto al 100%, ma questo può essere un dettaglio facilmente migliorabile, è la messa insieme delle tiles. Lo stacco in alcune aree è molto netto. Questa chiaramente è una limitazione del Seuck, però si può mascherare meglio (forse). Poi mi consiglia di andare a vedere Aviator dove sembra abbiano fatto miracoli.
Ha ragione. Mi ha scritto esattamente quello che sospettavo anch'io. Cado in una profonda depressione e decido di analizzare il gameplay di Aviator mentre mi mangio i fagioli alla picchiapò come Bud Spencer e Terence Hill. Aviator non mi fa impazzire come gioco ma tra una salsiccia e l'altra capisco come smussare il problema della monocromaticità.

La soluzione consiste nel dedicare ad ognuno dei mondi un'animazione speciale di colore diverso al colore del mondo sperando che gli dia "un po' di vita". Per esempio nel mondo dei cristalli ci saranno dei luccichii.

Nel mondo delle piante le foglie si muoveranno un pochino. (che eliminerò nelle versioni siccessive)
Nel mondo delle rocce ci saranno gli aculei che ho disegnato (poi cambierò idea in seguito). Nel mondo tecnologico ci metterò degli ingranaggi e delle scosse elettriche. Speriamo non mi costino troppi sprite. Dovrei cavarmela con 16 sprite e 5 o 6 animazioni.



03/11/2018

1237 sono i blocchi di memoria rimasti per i percorsi dei nemici. Ho completato il primo mondo e i 2 intermezzi tecnologici. Ho praticamente consumato 2/3 della memoria disponibile e ho coperto solo 1/3 del gioco. Un paio di calcoli ben assestati mi indicano che è inutile continuare con questo sistema perché non coprirò mai l'intera mappa di gioco a meno che non cambi qualcosa. Cancellare tutte le traiottorie dei nemici e ricominciare da capo non mi disturba anche perché il gameplay che è venuto fuori non mi piace. E' scialbo, blando e senza senso. Dopo ore di meditazione sul divano e di intensa autocritica penso che la soluzione migliore sia quella di allungare il brodo.
Allungare il brodo non suona bene ma qualche volta non è una cattiva soluzione.
Ecco la mia scaletta di lavoro:

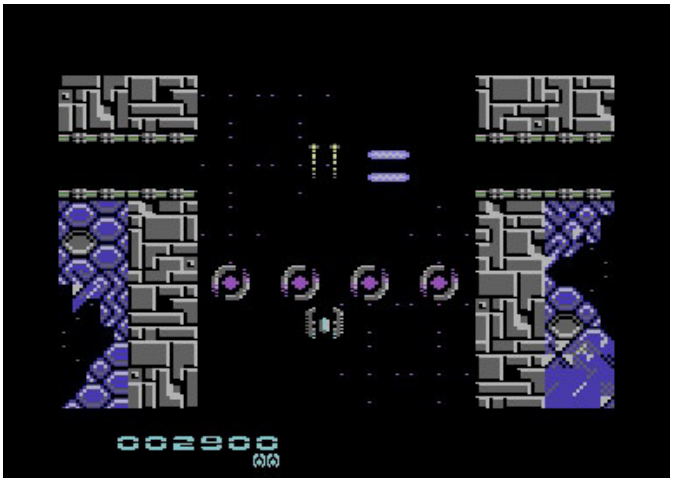
1) Togliere buona parte del primo mondo che non mi convince
2) Aggiungere 3 aree "STILL" nell'ultimo mondo, cioè le schermate dove il gioco si ferma per qualche secondo.
3) Utilizzare la funzione JOINT. Si quella funziona che mi faceva abbastanza schifo sin dall'inizio.

Nel frattempo continuo a modificare la mappa e a renderla più colorata e ne sono soddisfatto. Mescolando le tiles di diversi stili e aggiungendo sprites di colore diverso dovrei smussare quella cromaticità che non mi piaceva. Ho anche deciso di disegnare dei fossili. Danno l'idea di un pianeta vivente che ha una certa storia.

5/11/2018 – La terribile funzione Joint

Le intenzioni dei programmatori del Seuck erano buone.
La funzione "Joint" era stata pensata per incollare più sprite insieme e creare dei nemici più grossi, di solito boss di fine livello. Ma c'era un problema. Ogni volta che uno degli sprite che componeva il boss finale veniva distrutto, quello stesso sprite spariva ma rimanevano attaccati tutti gli altri. In pratica non moriva il boss intero, ma spariva un pezzettino per volta e l'effetto finale era abbastanza infelice e innaturale. Per questo motivo ho disegnato i boss di fine livello utilizzando i blocchi degli sfondi. Nonostante tutto la funzione "Joint" ha il lato positivo di farmi risparmiare memoria. Ogni volta che sposto un nemico dall'alto verso il basso consumo circa 17 unità di memoria. Se ci sono 4 nemici allora il conto diventa 17 x 4=68. Se invece uso il comando "joint",

cioè “uniscilo” io consumo 7 unità per i nemici dal secondo in poi, quindi 17 unità per il primo nemico , poi 7 per il secondo, 7 per il terzo e 7 per il quarto. Il totale con la soluzione “joint” sarà 38. 38 e’ meglio di 68. Ho risparmiato 30 unità.
Poiché il comando Joint non è stato pensato per gestire le formazioni, il comando Joint non controlla che gli sprite siano veramente attaccati tra di loro e quindi i nemici possono essere distanti l’uno dall’altro.
Ho provato diverse formazioni nemiche, a “X” a “V” ma il gioco mi va sempre in palla perché il “joint” vuole vedere tutti gli sprite sullo schermo e quindi per il momento l’unica formazione che riesco a fare è quando tutti i nemici sono allineati in orizzontale, ma è sempre meglio di niente. Non mi preoccupo molto per la memoria adesso.



Nei prossimi schemi i nemici avranno la capacità di sparare e quindi avrò bisogno di meno sprite e meno memoria. Dovrei farcela a riempire tutto il gioco, speriamo di riuscire a fare una cosa bella.

15/11/2018 – l’importanza dei punti di riferimento.

Sono abbastanza soddisfatto della mappa che ho disegnato fino ad oggi ma ho scoperto che non è funzionale con gli attacchi nemici. Alcune navicelle passano sopra i blocchi di cristallo o di roccia e questo è un errore che va contro le regole che mi sono dato. Vi ricordo che fin dall’inizio ho deciso che tutto quello che non era nero poteva distruggere il giocatore e di conseguenza anche i nemici.
Devo ridisegnare la mappa, forse non tutta, ma quella necessaria per permettere alle navicelle nemiche di passare attraverso la vegetazione, le rocce, i cristalli, proprio come farebbe il giocatore. Ho bisogno di avere dei punti di riferimento e quindi li disegno sullo sfondo nero. Toglierò questi punti alla fine.
Ridisegnare gli attacchi nemici consiste nel:

1) Aggiungere nemico
2) Controllare la memoria a disposizione.
3) Modificare la mappa affinché l’attacco nemico abbia un senso e non passi sopra lo sfondo
4) Modificare il blocco della mappa per renderlo più carino.
5) Controllare il lavoro fatto e se va bene allora salvare, per carità salvare!!!
6) Ripartire dallo step1

Alla fine mi ci affeziono a quei puntini e decido di tenerli, non tutti ma una buona parte.

Inoltre capisco che scopo del mio lavoro è quello di permettere a tutti i giocatori di “ammirare” la grafica dei 4 livelli che ho disegnato. Se devo sbagliare con la difficoltà allora preferisco sbagliare nel renderlo più facile invece che più difficile.
Uno dei commenti che riceverò in seguito è che non si percepisce la curva della difficoltà, risulta quasi costante dall’inizio alla fine, ma questa è un’altra storia.

01/01/2019 Rychard Blayliss

Blayliss ha aggiornato il suo sito e il SEUCK CONTEST 2019 ha inizio. Sono 2 mesi che lavoro agli attacchi nemici, al debug, ai mostri di fine livello e a rendere la mappa più interessante e colorata.

Ho salvato il gioco circa 200 volte.
Ho modificato la mappa circa 200 volte.
Ho cambiato gli sprite circa 200 volte.

Questo gioco è ben lontano dal gioco originale che avevo disegnato 30 anni fa, ma sono contento. Ho disegnato qualcosa di meglio. Sono abbastanza

soddisfatto perché ho utilizzato:

1) tutta la memoria per gli attacchi
2) tutti i 128 blocchi per la grafica
3) quasi tutti gli sprite a disposizione
4) tutti gli oggetti,
5) quasi tutti i 256 caratteri
6) tutta la lunghezza della mappa

Adesso sono stanco, annoiato e non mi diverto più.
Ad ogni miglioramento del gioco penso sempre ad altri progetti su altri tool (come il Graphic Adventure Creator o il TIC-80) e capisco che è arrivato il momento chiudere questo lavoro. L'ho fatto bene? L'ho fatto male? L'ho fatto. Anche se il contest scade il 1 Giugno 2019, e quindi ci sono altri 5 mesi di tempo per poterlo migliorare, mi sono ripromesso che spedirò il gioco a Blayliss entro la fine del mese di Gennaio.
Ho tempo 30 giorni per trovare degli effetti speciali un po' particolari e se non li trovo... pazienza, me ne farò una ragione. Io sono un grafico non un musicista.

22/01/2019 – Mega boss finale

Contatto Richard Blayliss e gli mando il gioco. E' molto gentile e disponibile ed è ben contento di ricevere la prima entry del Contest. Sarà stato un errore di battitura dovuto alla stanchezza ma subito dopo aver spedito il gioco mi rendo conto di aver sbagliato il nome nella intro. Invece di scrivere 477 ho scritto 447 ma poco importa a questo punto. Per quanto riguarda il megaboss finale ho trovato una soluzione e abbastanza "kubrickiana". Non so se molti l'apprezzaranno ma almeno è originale.
Gli scrivo il PLOT in inglese che dice:

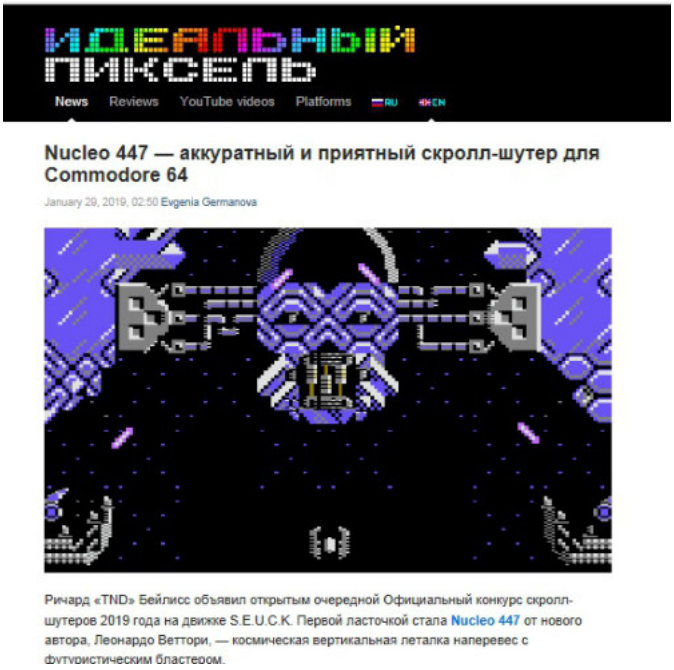
A message arrives from outher space for all living beings across the universe.
A crazy scientist decrypted it. It says something like "Warning! Asteroid out of control wandering! It contains the origins of the most dangerous weapon ever known! Enter the asteroid, find and destroy the weapon before it will destroy your planet!"
Nucleo 447 is the name of that asteroid and it is now floating through the solar system. All spaceships sent inside it never came back. Now it is your turn to try to save the universe.
There are 4 worlds: Crystal world, Green world, Rock world and the technologic world. After completing a world, you will be facing a final boss, which should be defeated in order to move on to the next zone.

Mi prepara una bella intro con musica lenta e abbastanza misteriosa proprio come gliela avevo chiesta e 3 giorni dopo il gioco è scaricabile dal suo sito.

29/01/2019 La mia prima recensione

Nucleo 447 — аккуратный и приятный скролл-шутер для Commodore 64

Il mio gioco piace in Russia e viene definito "un gioco pulito e piacevole". Improvvisamente mi rendo conto di provare grande ammirazione per il popolo russo.



02/02/2019 Nucleo 447 in 3D

Preso dall'entusiasmo della bella recensione decido di contattare Oliver Meyer Chi è Oliver Meyer? E' lo sviluppatore dell'emulatore YACE64. Ho sempre avuto un debole per questo emulatore perché, è in grado di trasformare un gioco 2D in uno 3D. Siccome tutti i giochi Seuck utilizzano lo stesso motore grafico, con le giuste impostazioni e qualche regola di base, è possibile rendere tutti i giochi Seuck un pochino più interessanti del solito. Anche un semplice giochino con una grafica mediocre ha la possibilità di sembrare più divertente se trasformato in 3D. Contatto immediatamente Oliver Meyer. Lui si dimostra subito interessato all'idea e gli chiedo di usare il mio gioco come test.
Dopo una serie di aggiustamenti vari, Oliver riesce a dare una profondità esagerata a Nucleo 447 e lo trasforma quasi in un TPS. (Third Person Shooter). Il gioco è praticamente ingiocabile ma l'esperimento è riuscito.

23/03/2019 Retromagazine

La mia avventura con Nucleo 447 finisce alla prima riunione di Retromagazine.
Si complimentano con me per il gioco ma qualcuno mi

fa subito notare che non si capisce il finale e ha ragione.

Non volendo uccidere un “feto” spaziale, perché politicamente scorretto anche se spaziale, ho preteso di scriverci sopra un plot e farlo diventare “la più pericolosa arma dell’universo” per lanciare una specie di messaggio ambientalista (l’umanità è l’arma più pericolosa dell’intero universo), ma avendo terminato tutta la memoria disponibile non sono stato in grado nemmeno di scrivere un “THE END” nella schermata finale. La scritta “THE END” può essere realizzata con un solo sprite ma che mi occupa un oggetto del Seuck e qualche unità di memoria.
Poiché:

1) non ho più oggetti disponibili
2) per liberare un po’ di memoria dovrei togliere un nemico da qualche parte
3) Non ne avevo voglia.

Mi sono ripromesso che se vinco il contest faccio una bella schermata finale che Richard Bayliss inserirà in qualche modo. Il problema è che il classico “giocatore Seuck” continua a sparargli (nel dubbio si distrugge sempre tutto) e non capisce perché il feto spaziale non muore. Semplicemente non muore perché non è il mostro di fine gioco ma la risposta a questo viaggio.
Evidentemente Stanley Kubrick, in 2001 Odissea nello spazio, è stato più bravo di me a costruire la scena finale. Me ne farò una ragione.

Bene, non resta che invitarvi a scaricare il gioco dal link indicato nel riquadro che trovate alla fine della pagina ed augurarvi buon divertimento.

Esempio di Pinball in 2D trasformato in 3D con Yace64.

Link utili:
Nucleo 447 dal sito di Richard Blayliss
http://tnd64.unikat.sk/
Seuck_Compo_2019.html#Nucleo447

Dalla Russia con amore: una recensione di Nucleo 447
http://tnd64.unikat.sk/
Seuck_Compo_2019.html#Nucleo447

YACE64
http://www.yace64.com/

Nucleo 447 in (pseudo) 3D
https://www.youtube.com/watch?v=hrP0OfYunJY

L’Associazione Firenze Vintage Bit Onlus vi ricorda il prossimo appuntamento: **Z80 - Prima parte**
Visto l'entusiasmo del precedente incontro sul tema del microprocessore Z80 svoltosi l'anno scorso, Walter Pugi ci propone di approfondire la conoscenza di questo microprocessore a 8 bit di casa Zilog, in due incontri consecutivi. In questo primo incontro analizzeremo la struttura interna del microprocessore, le operazioni logiche sui registri, i registri indice, le interrupt, le porte INPUT/OUTPUT e molto altro ancora.
Vi aspettiamo **Giovedì 26 Settembre** con **Z80 – Prima parte** con **Walter Pugi**.

Giappone 5^puntata: Saldi, Retrogaming o Fukubukuro ?

di Michele Ugolini

Cari lettori, il caldo sta terminando, le ferie sono finite quasi per tutti, siamo rientrati in pieno servizio presso i nostri doveri, l'esuberante Giappone si sta inesorabilmente allontanando dal picco del volume turistico estivo e spero tanto che la caccia nipponica al retrogaming sia stata fruttuosa.
Invece, se non siete ancora partiti per le ferie, sappiate che si sta avvicinando un periodo particolarmente interessante per questa straordinaria nazione.

E' ormai tempo di saldi!

E' veramente divertentissimo il Giappone in tempo di saldi, altrochè "cyber monday" o "black friday", troverete il "judgment day": l'euforia nazionale sarà alle stelle, cercherò di coinvolgervi in una recensione altrettanto effervescente e spiritosa, cosicché quando vi troverete in queste situazioni aliene ricorderete le mie parole, spero, con un sorriso.

Siete reduci dal viaggio in Tokyo o qualche altra città maggiore? Avete notato quanti Otaku locali rovistavano nei vari reparti, presso le catene, non solo del retrogaming, ma sopratutto dell'usato elettronico? Avete anche notato il loro ammontare di "touch and go" che venivano reiterati, fino allo sfinimento, presso queste mega ditte? Nei miei primi viaggi, questo strano fluire di personaggi stravaganti non mi destava particolare curiosità.

Dopo svariate visite, però, il loro ectoplasmatico, silenzioso e composto via-vai, non poteva proprio più rimanere eluso dalla mia attenzione.
Complice la loro indole educata ed apparentemente recessiva, che in realtà sfocia spesso nella patologia sociale del personaggio passivo/aggressivo, complice la loro caccia serrata, dove ogni Otaku comprende benissimo che la scaffalatura X del negozio Y appartenente alla catena Z sarà rifornita il giorno K... ho appreso anche io alcune logiche dalla loro metodica.
In realtà cerco sempre di mediare, tramite un costante dialogo con la mia coscienza, un certo distaccamento dalla loro prossemica.

Vi invito caldamente a seguire questo consiglio, altrimenti potreste finire nel pericolosissimo vortice della caccia al retrogaming inerente ai titoli "loro", ricercati e supervalutati in terreno "loro", dovendovi infine scontrare in futuro contro le "loro" regole.
Vi rammento anche che, rileggendo le scorse recensioni riguardo il Giappone, potrete rinfrescare una sana allergia che potrà distanziarvi dal loro mondo reale regnato da Otaku surreali.

Vi ricordo che il loro mondo è saturo di situazioni patologiche, dove i semplici sostantivi Otaku e Hikikomori sono entrati nello scibile italiano da pochi anni.
In realtà in Giappone rappresentano una piaga difficilmente reversibile, germinata probabilmente negli anni settanta, sapientemente sminuita dalla sinistra coscienza del marketing nipponico.

Vi chiederete come mai ogni volta vi metto in guardia dal loro mondo.
Probabilmente penserete che sono rimasto traumatizzato da alcune loro meccaniche interne.
Si, avete indovinato, la caccia al retrogaming, nonché il Giappone in generale, mi hanno donato tante emozioni positive e probabilmente altrettante negative.
Perciò memore dei miei scontri, inizialmente persi pietosamente, vorrei farvi apprendere alcuni punti importanti per evitare future perdite di tempo, pazienza, soldi, etc..

Dovrete assecondare le loro regole, non scontratevi mai direttamente, il loro meccanismo non vi permetterà un approccio rustico: non vi infiltrerete senza un piano.
Fidatevi, loro sono ottimi ingranaggi nel sistema al quale appartengono, non ci sarà spazio per noi che sbarchiamo su questo territorio alieno, oltretutto già minati dalla stanchezza del viaggio: saremo sudati, con un fuso orario che ci sta allucinando insieme alle accecanti luminarie onnipresenti, il tutto farcito da profumi esotici e rumori assordanti profusi da questo fluente meccanismo ordinato e sincronizzato.

Purtroppo la caccia al retrogaming in Giappone è una cosa seria, estremamente redditizia, in questo caso è il pianeta Terra che ruota attorno al loro asse videoludico, infinitamente ricco di pericoli e complesse situazioni sociali borderline.

Questo problema non sussiste se visiterete il Paese del Sol Levante una o due volte nella vostra vita, là è tutto perfettamente organizzato per mostrarsi ai vostri occhi attraverso il massimo grado di tripudio sensoriale.

I problemi nascono se amate il retrogaming e deciderete di aprire la vostra stagione di caccia mirando ad una collezione che godrà la propria solenne eternità

dentro la famosa vetrina anti proiettile, illuminata, pressurizzata e quindi più sacra, che risiede nella vostra abitazione.
Stavo parlando dei saldi? Lasciate ogni speranza voi che proverete, da lontano, ad entrare...
(cfr. figura 1 e 2).

Figura 1

Primo consiglio: c'è saldo e saldo.

Le tipologie di saldi, inerenti alle caratteristiche settoriali del loro marketing interno, nascono per esigenza, ovviamente, della loro domanda interna, ergo, più è apparentemente strana la richiesta di un determinato servizio (o merce), altrettanto saranno apparentemente strane le regole generate per quel determinato elemento. Queste regole non rispetteranno i classici vincoli stagionali o dettami profusi dalla moda o anniversari di una particolare uscita di un prodotto oppure ancora ricambi obbligati dalla scadenza di un determinato servizio.

Quindi, se vorrete approcciarvi alla caccia di un determinato elemento, dovrete studiare attraverso i loro social, quali situazioni migliori si possono creare per ottimizzare il vostro "appostamento di caccia".

Per esempio, da qualche anno sto cercando di completare un settore di prodotti della Square Enix, quasi quotidianamente devo reperire informazioni gentilmente tradotte dai miei "infiltrati" per poter prendere il biglietto aereo che mi permetterà di raggiungere al momento giusto, il negozio giusto che accetterà la mia prenotazione attraverso le allucinanti regole del preorder dettate dalla ditta medesima.

Vi rimando alla lettura dell'articolo inerente alla loro machiavellica concezione, non di acquisto, bensì di un misero sorteggio per il preorder di un prodotto da collezione.
Tutto rasenta la pazzia, almeno dal nostro punto di vista.
Temete forse che io mi stia burlando di voi?
Documentatevi e sorprendetevi!

Figura 2

Secondo consiglio: Fukubukuro, violenza manifesta, ma buste chiuse!

Attenzione agli specchietti per le allodole. Prima o poi vi capiterà di osservare, nei negozi con prodotti in saldo, delle misteriose buste colorate, appariscenti, voluminose, pesanti, tutte in fila, sigillate all'inverosimile, contenenti chissà quale rarità, o molto più probabilmente, tutti gli scarti di un magazzino che proprio non sapeva più dove rifilare quei prodotti così maneggiati, rovistati, impolverati, provenienti dagli scaffali di qualche distretto così visitato che neppure la "Signora Noia" in persona avrebbe resistito ad uno sbadiglio nel visionarli.
Non fate i miei errori mi raccomando.

L'enorme busta rossa (cfr. figura 3) potrebbe contenere un cuscino con stampato Donkey Kong, così maneggiato da tante mani, che la scimmia apparirebbe meno marrone delle ditate impresse sulla

Figura 3

stoffa!

I veri saldi, cari lettori, sono quelli rappresentati da folle di casalinghe e studenti ammassati (ma pur sempre in fila composta ed ordinata), davanti alla porta d'ingresso di specifici grandi magazzini, almeno da alcuni giorni. Lì si trovano gli oggetti dei desideri a prezzi ottimi. Il problema è che i nostri giorni di ferie scorrono alla velocità della luce e nessuno di noi vorrà mai sostare in fila, neppure per qualche ora, in attesa dell'apertura di quei precisi negozi. Oltretutto quando si aprono le porte, si scatena l'inferno e solo loro hanno studiato al millimetro i vari percorsi interni per raggiungere l'oggetto del desiderio, mentre noi, non avremo idea di dove il fiume umano potrà trascinarci (cfr. figura 4)!
Temete forse che io mi stia burlando di voi?
Documentatevi e sorprendetevi!

Terzo consiglio: PS4 Idols games ed i loro vicini di casa delle Stark Industries.

Siccome i loro oggetti del desiderio non subiscono la mera regolamentazione di un prodotto standard, sarà improbabile che riuscirete nell'acquisto di un loro videogioco per PS4 inerente ad un fenomeno interno, per esempio degli Idols. Pertanto rimediare "The Idolmaster: Platinum Stars" originale, della Bandai Namco, in versione platinum, deluxe o uranium edition, diventerà una vera impresa anche per loro. Per noi sarà più facile addentrarci nei saldi dei prodotti che non rappresentano una meta delle loro mode.

Ad esempio, chi tra noi non ha mai desiderato possedere un mini reattore atomico per soddisfare le proprie esigenze energetiche domestiche? Pensate che possedere un tale giocattolo sia illegale in Giappone? Pensate forse che unicamente Tony Stark e Sheldon Cooper li abbiano progettati ed utilizzati?

Ragazzi miei, si sta concludendo il 2019, guardate la fotografia della prossima pagina (cfr. figura 5) : la casalinga giapponese in copertina veste la classica moda anni settanta, ottanta.
Se a quel tempo poteva lei, perché mai il giapponese medio del 2020 non dovrebbe godere di siffatto innocente e silenzioso elettrodomestico che gli fornirà per decenni, abbondanti KiloWatt di energia elettrica e sopratutto una goduriosa infinità di acqua bollente gratis?

Certo, l'investimento iniziale è consistente, pertanto informatevi bene in dogana onde evitare le solite noiosissime e scontate scartoffie burocratiche. Vi starete domandando come funziona il vuoto a rendere o a perdere di questo giocattolino. Nessun problema, al numero di telefono che vedete sulla rivista troverete massima cordialità, pronta ad abbracciarvi nel frizzante mondo del nucleare! Temete forse che io mi stia burlando di voi?
Documentatevi e sorprendetevi!

Quarto consiglio: non fidatevi assolutamente di ciò che vi dico, mai!

Si esatto, è una condizione molto importate. Documentatevi prima di partire, leggete le recensioni e studiate i video sul web attinenti all'argomento da voi scelto.
Non fidatevi della mia modalità spiritosa nel descrivere le follie nipponiche.

Ormai non so più bene neppure io a quale dei due mondi appartengo: sono compromesso!

Per esempio, una fetta consistente di visitatori proviene dall'Australia. Ci sono anche tantissimi americani. Entrambi sono ottimi recensori. Cinesi e russi non sono così frequenti.

I coreani si mescolano tra i giapponesi in maniera a noi poco smascherabile, oltretutto attingere informazioni dalla loro lingua è complesso per via della loro grafia e lingua altaica. Estrapolare informazioni dai video in lingua cinese o dall'alfabeto cirillico è altrettanto frustrante.

Gli europei rappresentano una fetta ridotta del turismo nipponico.
Però non sarà impossibile trascorrere un ipotetico capodanno nell'affollatissimo quartiere di Shibuya ed ascoltare i vostri vicini

mentre chiacchierano proprio nel vostro dialetto locale. Si, mi è successo, poi purtroppo è anche arrivata una moltitudine di polizia che ci ha imposto di sgomberare il famoso attraversamento pedonale di Shibuya proprio mezz'ora prima delle 00:00, un abbattimento morale rapidamente risollevato lungo le voluminose arterie collaterali del distretto, per fortuna!

Temete forse che io mi stia burlando di voi? Documentatevi e sorprendetevi!

Bene cari lettori, spero che la curiosità vi porterà ad ottimizzare la programmazione delle vostre eventuali ferie autunnali, così vicine alle loro date dei saldi.

Spero anche di aver generato in voi un sorriso, così potrete utilizzarlo sfuggendo agilmente da qualche "situazione imbottigliata" descritta poco sopra.

Un abbraccio e mi raccomando, fatemi sapere l'argomento che vorrete leggere nel prossimo numero.

Figura 5

Figura 4

Un autunno di RetroEventi!

Con la fine dell'estate ad aspettarci non c'è soltanto il ritorno al solito tram tram quotidiano ma per fortuna, per noi instancabili appassionati di retrocomputing e retrogames, anche tanti eventi sparsi un pò in tutta Italia. Ecco perciò per voi una lista, rigorosamente in ordine cronologico, di appuntamenti imperdibili.

Dal 2013 a San Giorgio Canavese, in provincia di Torino, si tiene uno dei principali eventi legati ai mattoncini LEGO in Italia: il **San Giorgio Canavese Brick Expo**!
Due giorni all'anno, oltre mille metri quadrati di esposizione di creazioni originali realizzate con i mitici mattoncini.
E dal 2017 l'evento si è evoluto, triplicando addirittura lo spazio a disposizione!
Oltre all'esposizione di LEGO, è stato inaugurato un nuovo padiglione dedicato al modellismo statico e dinamico e uno dedicato al mondo del fumetto e tutto ciò che gli gira intorno.
Molto interessante l'area retrogames curata da Massimo Chiaffredo e Mauro Cipolla da sempre appassionati di videogame e di console della loro giovinezza che hanno conservato e hanno nel tempo affiancato altri preziosi tesori scovati in mercatini, cantine, solai e ovunque si possano trovare perle coperte dalla polvere. Hanno affinato anche la loro perizia tecnica, così da riuscire a riparare e rimettere in funzione molte di queste vecchie glorie che sembravano destinate all'oblio.
Per non parlare della loro cultura enciclopedica al riguardo: chiedetegli notizie di qualunque oscuro videogioco uscito solo a Osaka nel '75 e ve ne sapranno raccontare vita, morte e miracoli!
Il successo dell'anno scorso ha sbalordito tutti. La coda di bambini che volevano giocare con Super Mario o Sonic è stata una sorpresa incredibile, nonchè il segno che quei vecchi e quasi artigianali videogiochi, se confrontati con le meraviglie tecniche moderne, hanno ancora qualcosa da dire, anche alle nuove generazioni.
Quest'anno perciò si replica con NES, Sega Mega Drive, Intellivision, Playstation e tanto altro: che siate nostalgici che vogliono riprovare le emozioni dei vecchi tempi, giovani curiosi di scavare nell' "archeologia" del vostro passatempo preferito o esperti appassionati del settore, il San Giorgio Brick Expo è il posto che fa per voi.
L'appuntamento è per il 21 e il 22 Settembre 2019 al castello di San Giorgio Canavese!

Il **Carmacomics**, giunto alla 5° edizione consecutiva, ospita autori, artisti, doppiatori e fumettisti come Ugo Verdi e Pabus (Road to Armageddon), Sayuki Nanaji e Andrea Kha (Tidal Lock), AlbHey (Bao Publishing), Elena Romanello (Scrittrice), Monica Tomaino (Scrittrice Fantasy), Renato Novara (Doppiatore di Ted Mosby in How Met Your Mother, voce di Ezio Auditore in Assassin's Creed di Rufy in One Peace e di Sonic The Hedgehog...)
Non mancherà, come ogni anno, la presenza di Doctor Game, esperto di retrogame e amatissimo su youtube grazie ai suoi video ironici, taglienti e pieni di contenuto storico.
Anche l'area Games sarà attiva con il team di Videogames Generation, con postazioni dedicate al retrogame old school (con console vere) ed emulazione, con ampio spazio anche a postazioni musicali (con un vero controller per Beatmania e non solo), non mancherà un cabinato su scheda Neo Geo MVS originale e aree dedicate alle ultime novità videoludiche, anche indipendenti.
Durante i giorni di fiera, come da tradizione, si svolgeranno i tornei di Fifa e del Videogames Generation Championship dedicato al retrogame.
Grazie a Dimensione Arcana ci sarà anche un'area dedicata ai giochi di società, e ai tornei di Magic nello spazio adibito da Magic Fun.
Nella domenica si svolgerà anche la gara cosplay sul palco con giudici dal mondo del Cosplay e diversi premi di categoria.
L'appuntamento è per il 4 -5 - 6 Ottobre a Carmagnola in Provincia di Torino, in piazza Antichi Bastioni con ingresso rigorosamente gratuito.
Ultima novità di quest'anno sarà l'area esterna dedicata allo street food (Nigiri e Tempura inclusi)

Dopo il successo nell'edizione di Milano e Verona, il 20 Ottobre tornano a Roma, a grandissima richiesta, gli amici del **Coin Up**.
CoinUp Italia è la prima mostra mercato italiana dedicata al mondo del retrogaming, board games e videogiochi usati, organizzato da appassionati del settore stanchi di vedere i propri vecchi giochi svenduti e svalutati.
Oggi l'evento è diventato un punto di riferimento per tutte quelle persone, private e non solo, che cercano un'occasione per comprare, vendere e scambiare i propri retrogaming, board games e videogiochi usati.
L'evento inizierà alle ore 12:00 e si chiuderà alle ore 19:00. Il biglietto da acquistare

RETROEVENTI SETTEMBRE - OTTOBRE 2019

SAN GIORGIO BRICK EXPO
S.Giorgio Canavese
21-22 Settembre

CARMACOMICS
Carmagnola (TO)
4-5-6 Ottobre

all'entrata è di 5,00€. I minori di 12 anni entrano gratis se accompagnati da un adulto con biglietto, inoltre all'interno dell'area troverai bibite e panini.
Se poi volete battere la folla potete approfittare degli ingressi VIP ad accesso anticipato a numero chiuso, che ti permettono di entrare in esclusiva dalle ore 11:00.
Vuoi partecipare come espositore e vivere l'emozione della vendita? Bene, essere protagonista al CoinUp è molto semplice ed emozionante, manda una mail a coinupitalia@gmail.com o scrivi nella pagina dei contatti e prenota il tuo spazio.
Con un contributo di 30€ hai diritto a:
- il tuo spazio espositivo della lunghezza di 2 metri lineari compreso di tavolo e due sedie
- due pass espositore.
Così potrai esporre i tuoi giochi per l'intero evento e se disponi di molto materiale puoi prenotare anche più tavoli.
Se invece ti serve meno spazio con un contributo di 20€ hai diritto a:
- il tuo spazio espositivo della lunghezza di 1 metro lineare compreso di tavolo e una sedia
- un pass espositore.
L'evento si svolgerà completamente al coperto e quest'anno vedrà anche la presenza di noi di Retromagazine con la partecipazione del sottoscritto, di Giorgio Balestrieri e di Ermanno Betori. Insomma è vietato mancare!

Il 5 Ottobre a Quiliano in provincia di Savona si svolgerà **Archeologia informatica e Videogiochi**.
L'idea è di percorrere un po' la storia dell'informatica sia documentandola e sia nella pratica utilizzando una serie di emulatori oggi disponibili su PC.
L'attenzione sarà verso emulatori liberi ed in particolare su quelli che girano nativamente o mediante wine su GNU+Linux (si accennerà anche ad esempio a WinUAE).
Inoltre sarà certamente presente il Raspberry come base per il retrocomputing,
considerando anche l'eventuale possibilità di sostituirlo a schede originali non funzionanti all'interno dei vecchi case.
Non verrà trascurato comunque l'aspetto "legale" che riguarda le ROM e videogiochi.
Inoltre sarà presentato Amiga Forever come opportunità legale di emulare l'Amiga.
E non saranno certo dimenticati progetti come AROS e HAIKU.
In ultimo, ma non ultimo, le avventure testuali in Inform dell'amico Marco Vallarino (del quale abbiamo già parlato qui su retromagazine tempo fa)
Si tratterà fondamentalmente di un evento frontale anche se ci sarà adisposizione del pubblico dei PC e Raspberry con emulatori funzionanti sui quali potranno provare giochi (e sistemi operativi) e qualche home computer "originale" (purtroppo non sempre funzionati).
L'evento si terrà presso la biblioteca civica A. Aonzo e a questo appuntamento ne seguirà un altro dedicato al Retrocomputing e alla Didattica.

Infine vi segnaliamo, il 26 Ottobre a Milano, **Once Upon a Sprite - Retrochiacchiere e dintorni**.
Giunto ormai alla quarta edizione, l'evento è dedicato al retroprogramming con l'intento di ripercorrere gli anni '80 e '90 attraverso l'esperienza e la viva voce di chi in quel periodo è stato coinvolto nello sviluppo di videogiochi per le più diverse piattaforme.
Anche se il retroprogramming è l'elemento principale della manifestazione, non è il solo tema trattato; nei talk previsti per la giornata si esploreranno anche le meraviglie della filiera produttiva di giochi e software di quegli anni: dallo sviluppo alla pubblicazione fino alla distribuzione, sia tramite i canali leciti che, soprattutto in Italia, quelli illeciti, toccando argomenti come le demo e le cracking scene ed il fenomeno della pirateria in generale.

Tra gli eventi in programma segnaliamo:
- Nascita e sviluppo del mercato videoludico italiano a cura di Carlo Santagostino
- Tavola rotonda: editoria videoludica italiana, con i protagonisti di VIdeogiochi, Zzap, TGM e K
- Scena Hacking/Phreaking su Amiga, a cura di Stefania Calcagno
- Storia dei gruppi hacker europei (Amiga/PC) a cura di Antonio Mazzanti (Randall Flagg di Razor 1911)
- Space ARDUINvaders - Reverse engineering della board di Space Invaders e reimplementazione su Arduino Nano a cura di Giuliano C. Peritore, Vittorio Signorelli, Maurizio Damiani Chersoni
- Scrivere un emulatore di Commodore 64 (e farci girare i giochi di Jeff Minter) a cura di Valerio Lupi (Xoanino)
L'evento si avvarrà anche del supporto di Archeologia Informatica, Retroprograming Italia, Retrocampus e ovviamente di RetroMagazine, che sarà presente nella figura del nostro David La Monaca.

di **Querino Ialongo**

COIN UP!
Roma 20 Ottobre

ARCHEOLOGIA INFORMATICA E VIDEOGIOCHI
Quiliano 5 Ottobre

ONCE UPON A SPRITE
26 Ottobre Milano

Avventure, avventurieri e sviluppatori

Siamo ormai quasi ad ottobre, l'attività lavorativa e scolastica è ripresa a pieno ritmo da un po' ed i salutari effetti delle vacanze iniziano a mostrare già qualche cenno di cedimento. Occorre dunque correre ai ripari, impiegare quel po' di tempo libero che ci resta per rievocare i ricordi della scorsa estate, dei momenti passati in giro per le località teatro del nostro riposo estivo e delle avventure, piccole o grandi che siano state, che abbiamo vissuto. Ecco, avventure, è questa la parola chiave. La loro rappresentazione su computer e dispositivi per il gaming di varia natura ha da sempre avuto un suo fascino e chi ci segue da qualche tempo sa quanto noi di RetroMagazine siamo affezionati a questo tema. Un gioco d'avventura, che sia grafico o testuale è perfetto per donarci anche un solo breve momento di evasione dalla realtà e garantirci una piccola pausa ristoratrice, il tutto con il solo impiego della nostra fantasia e della nostra materia grigia. Potremmo recuperare un classico del passato certo, oppure procurarci uno dei tanti titoli che ancora oggi vengono sviluppati ed apprezzati, ma quest'anno, quest'inverno, possiamo fare di meglio. Possiamo scriverne una, farla giocare ad un'intera comunità di appassionati ed addirittura essere premiati, se la nostra opera stimolerà l'interesse di un pubblico sufficientemente vasto.

Come già nel 2018, i nostri amici di Old Games Italia hanno organizzato anche quest'anno il concorso "Marmellata d'avventura", dedicato a chi ha sempre avuto nel cassetto un'avventura testuale da scrivere ma non ha mai avuto l'occasione o lo stimolo giusto per farlo. Forte del successo di autori e giocatori della scorsa edizione, il regolamento del contest è stato ampliato per dare maggior stimolo agli autori, che dovranno curare sia la qualità narrativa dei contenuti e la complessità della trama, che esplorare ed ampliare l'aspetto "tecnico" di questa tipologia di giochi. Ai fini dell'ammissione alla competizione, le opere dovranno ricadere comunque nella sola categoria delle avventure testuali, quale che sia la loro declinazione, ed essere inedite. Per quanto riguarda lo sviluppo invece, non ci sono limiti: qualsiasi linguaggio di programmazione, tool di sviluppo e persino piattaforma è concesso. Per gli autori che hanno partecipato l'anno scorso è disponibile una sezione "fuori concorso", dove è possibile presentare un nuovo capitolo delle proprie avventure già pubblicate o riproporre un gioco in versione corretta e migliorata, senza alcun obbligo di attinenza ai temi della competizione. Noi di RetroMagazine, sia per le ragioni espresse più volte sulle pagine della rivista che per gli aspetti di sviluppo coinvolti, non possiamo essere meno che entusiasti di un evento del genere e vi invitiamo calorosamente a partecipare, in veste di autori, di giocatori o perchè no, di entrambi.

Per maggiori informazioni, potete fare riferimento alla pagina ufficiale di "Marmellata d'avventura" su:

http://www.oldgamesitalia.net/forum/index.php?showtopic=26866

Per questo mese è tutto, il numero 17 di RetroMagazine si chiude qui, arrivederci tra un mese (più o meno, siamo sempre una rivista aperiodica) ed in bocca al lupo a chi vorrà partecipare all'evento di OGI. E, ovviamente, buon divertimento a chi giocherà le avventure in gara.

Giorgio Balestrieri

Disclaimers

RetroMagazine
Anno 3 - Numero 17

Direttore Responsabile
Francesco Fiorentini

Vice Direttore
Marco Pistorio

Settembre 2019