RetroMagazine
semplicemente retro

Numero 15 - Anno 3 - Giugno 2019 - WWW.RETROMAGAZINE.NET - Pubblicazione gratuita

"Come far rinascere un cabinato..."

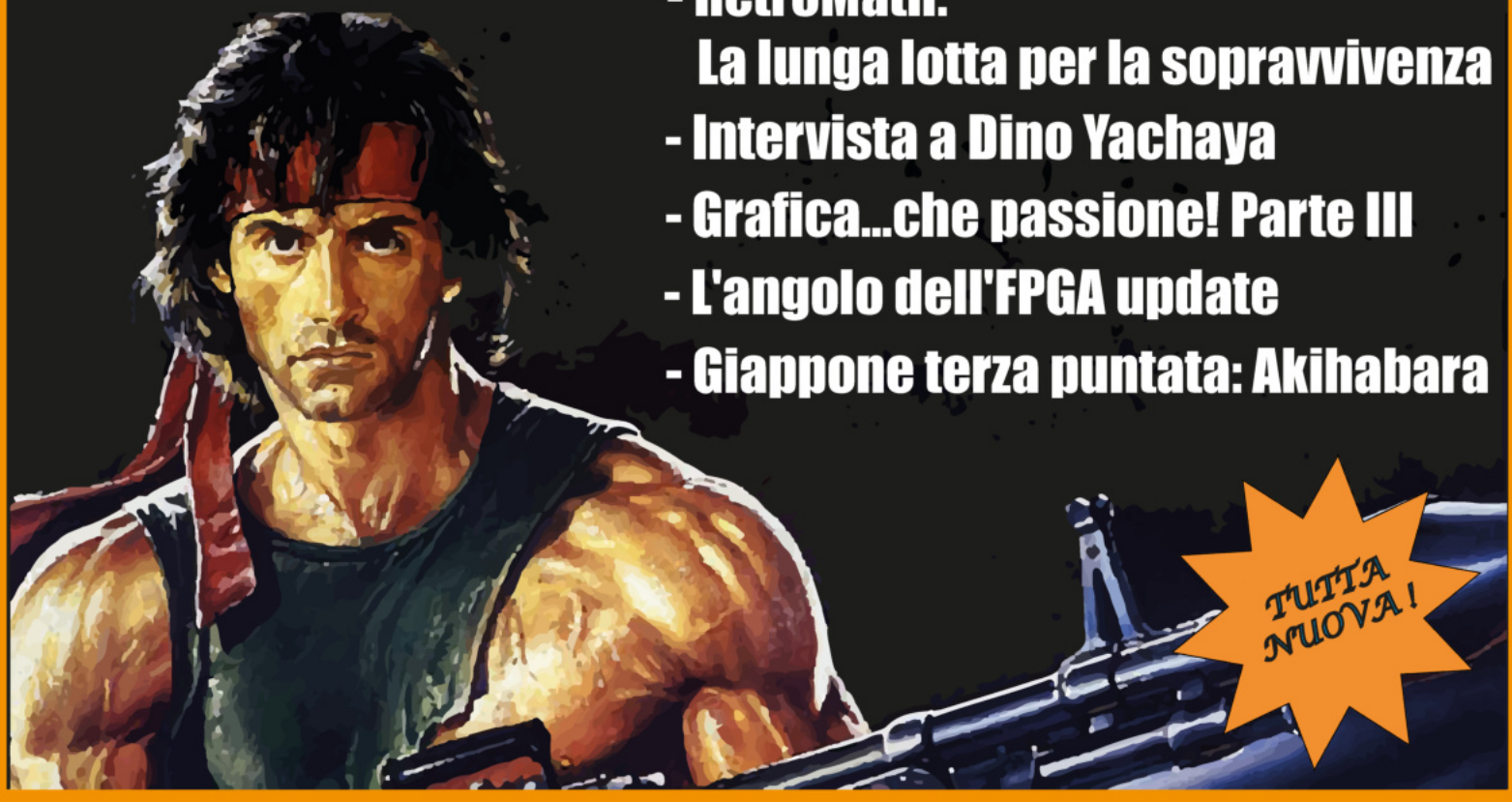
Pooyan BASIC Contest (Aprile-Maggio 2019)

Prince of Persia (AMIGA)

RAMBO First Blood Part II (C64)

....e ancora:

- Retroeventi giugno - luglio
- RetroMath: La lunga lotta per la sopravvivenza
- Intervista a Dino Yachaya
- Grafica...che passione! Parte III
- L'angolo dell'FPGA update
- Giappone terza puntata: Akihabara



RetroMagazine si rinnova!

Benvenuti cari lettori in questo nuovo numero di RetroMagazine; un numero speciale!

Un vecchio adagio calcistico dice letteralmente "squadra che vince non si cambia", ma noi di RM siamo piu' fedeli al motto Darwiniano "...la specie che sopravvive è quella che è in grado di adattarsi e di adeguarsi meglio ai cambiamenti dell'ambiente in cui si trova..." e quindi, per garantirci una lunga sopravvivenza, abbiamo deciso di cambiare la veste grafica e lo strumento con il quale assembliamo il nostro 'giornale'.

Da questo numero infatti, RetroMagazine verra' assemblato con **Scribus**, un software di impaginazione open source che, alla lunga, dovrebbe garantire maggiore flessibilita' con notevoli benefici nella gestione degli articoli e delle singole pagine. Non voglio stare a dilungarmi troppo su quelle che sono state le ragioni dietro la scelta di questo strumento, ma essendo RM una pubblicazione gratuita, l'adozione di un software open source ci e' sembrata ovviamente una soluzione ragionevole. Quello che invece vorrei sottolineare e' stato lo sforzo di un paio di membri della redazione per la realizzazione dei template. **Marco Pistorio** ed **Andrea Patrizio** hanno creato i modelli che attualmente stiamo utilizzando per impaginare con Scribus. Personalmente devo ammettere che il passaggio da Word, che conosco abbastanza bene e con cui ormai mi sentivo a mio agio per impaginare RM, a Scribus e' stato abbastanza traumatico. Si tratta in realta' di due software diversi, sia per destinazione che per filosofia e, senza i modelli messi a disposizione da Marco ed Andrea, probabilmente la mia curva di apprendimento sarebbe stata estremamente piu' lunga.

Questo quindi e' il primo numero con uno strumento completamente diverso dal solito ed i redattori tutti stanno cercando di familiarizzare con le nuove funzioni ed esplorando nuove possibilita'. Probabilmente questo numero potra' sembrare un po' grezzo, ma vi prometto che, come nostra abitudine, miglioreremo strada facendo! Se avete consigli da darci per abbellire la veste grafica in Scribus, fateci sentire la vostra voce e contattateci!

Francesco Fiorentini

SOMMARIO

Hanno collaborato alla stesura di questo numero di RetroMagazine :

- Roberto Lari
- Marco Petri
- Davide Fichera
- Arturo Dente
- Felice Nardella
- Alberto Apostolo
- Giuseppe Fedele
- Marco Pistorio
- Querino Ialongo
- Daniele Brahimi
- Francesco Fiorentini
- Adriano Avecone
- Ermanno Betori
- Starfox Mulder
- Michele Ugolini
- Federico Gori

Immagine di copertina realizzata da Flavio Soldani

Bare Metal C64 per Raspberry 2 e 3

...tranquilli, nessuna bara di metallo per i nostri amati Raspberry :-)

di Roberto Lari

In quest'articolo vi voglio parlare di un prodotto software molto particolare e che sta creando una piccola rivoluzione in un settore specifico, quello degli emulatori che girano su Raspberry (2 e 3B & 3B+).

Si tratta del BMC64.
"Cosa diavolo è il BMC64?" vi starete chiedendo. Ve lo spiego subito.
BMC64 è l'acronimo di Bare Metal C64, dove Bare Metal, tradotto alla buona dall'inglese significa "metallo grezzo" ed è un emulatore del nostro amato Commodore 64.

Tutto questo rumore per l'ennesimo emulatore del C64 direte voi. Si' ma c'è una grossissima differenza rispetto ai soliti emulatori disponibili per il Raspberry; mentre generalmente tutti gli altri si appoggiano ad un sistema operativo Linux, che fa da tramite tra l'emulatore del C64 e l'hardware vero e proprio, nei sistemi Bare Metal questo non avviene. E' l'emulatore del C64 stesso che funge da sistema operativo tra l'utente e l'hardware (grazie anche ad un set di librerie chiamate Circle), andando a risparmiare preziose risorse (come ram e cicli della cpu) e facendo si che perfino un non troppo potente Raspberry 2 (con cpu da 900mhz dual core) sia più che adeguato per garantire un'emulazione accurata del nostro amato C64.

Ecco spiegato il significato di "metallo grezzo" :-)
Un'altra cosa assolutamente degna di nota del BMC64 è il tempo di avviamento. Circa 4 secondi da spento a C64 avviato; in pratica quasi come il C64 reale!

Questa versione dell'emulatore è basata sul Vice 3.3 SDL2, l'ultima versione del famosissimo kit di emulatori Commodore 8 bit uscita a fine 2018 e che è, a mio parere, fatta davvero bene.

Già così sarebbe abbastanza, ma il meglio deve ancora venire.

Con una configurazione standard, con il nostro Raspberry sul tavolo collegato ad un monitor HDMI, un gamepad ed una tastiera USB, sarebbe già una bella cosa, ma forse non sufficiente ad esprimere il meglio di se!

Immaginate invece di avere il raspberry dentro la scocca di un vero C64 (biscottone o C64C), con collegati la tastiera del vero C64 e due joystick standard Atari DB9 tramite un'interfaccia (acquistabile online, nuova, senza problemi) chiamata Keyrah e con tanto di LED nella scocca funzionante. Inoltre, ciliegina sulla torta, il tutto collegato ad un monitor CRT tramite l'uscita audio/video composita (magari lo stesso monitor che utilizzate per il vero C64!). Terminate il tutto con una prolunga MicroSD - SD da mettere al di fuori della scocca per gestire facilmente la MicroSD ed un bel cavetto USB con interruttore per l'alimentazione. Otterrete qualcosa che a fatica si potrà distinguere dal C64 reale!
Altro aspetto davvero degno di nota è che non è necessario alcun kit di raffreddamento dotato di ventolina, quindi rumore zero. Personalmente comunque, consiglio sempre di installare sul Raspi un seppur minimo kit di dissipatori in rame per aumentarne la longevità.

Ma vediamo come funziona all'atto pratico questo emulatore: la prima cosa che ci domanderemo è quali siano i formati supportati. In questo caso tutti i più famosi ed utilizzati, ovvero i D64 & i G64 (copie 1:1 degli originali, protezioni incluse), le cartucce in formato CRT (accetta anche cartucce acceleratrici dei floppy disk come l'Epyx Fast Load e l'Action Replay con Fastload integrato oltre che quelle dei giochi, EasyFlash comprese), i benedetti TAP (ha un'emulazione del Datasette completa di tutto, counter compreso) ed anche i file PRG.

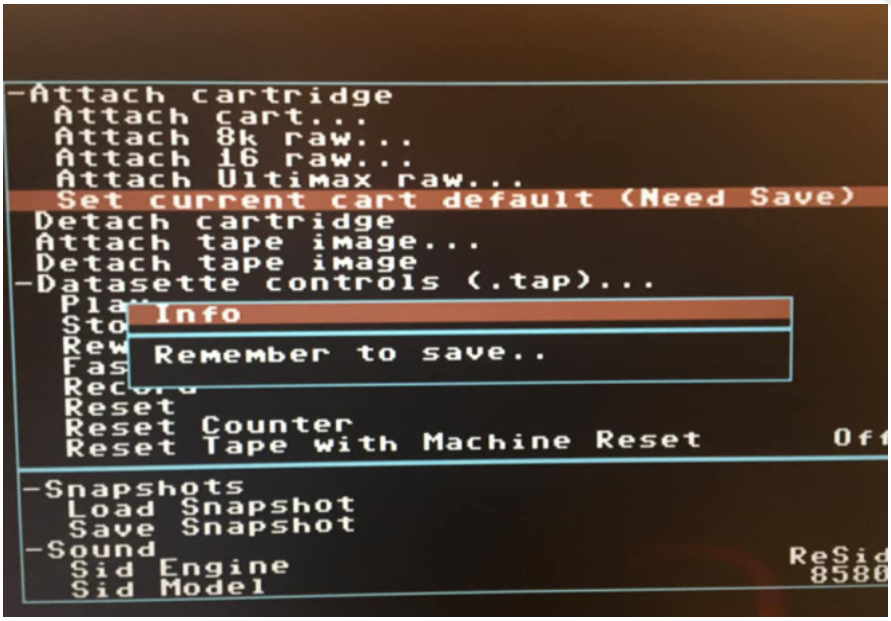
Inoltre, supportando l'emulazione dei drive più famosi e utilizzati dai

Un Commodore 64 che sembra vero ma non lo e'...

nostri amati 8 bit, come il 1541, il 1541-II, il 1571 e il 1581, potremo utilizzare anche i formati .D71 & .D81 (in lettura e scrittura). Viene emulato addirittura il rumore di caricamento dei floppy disk (tranne nel caso del drive 1581, rumore assente anche nel Vice per PC) e sono disponibili tutte le unità da 8 a 11.

Come comodità aggiuntive abbiamo il Warp per accelerare l'emulazione, lo swap delle 2 porte joystick (per non dover spostare continuamente il joystick da una porta all'altra come si faceva con il vero C64, a meno di non dover tenere due joystick collegati in pianta stabile) e la possibilità di mostrare / nascondere la status bar presente in basso. Status bar che ci mostra i led delle unità floppy drive che stiamo usando, lo stato del warp (si / no), lo stato delle porte joystick (normali o swappate) e lo stato del Datasette (play, stop, rew etc.etc.) con il relativo counter, grazie al quale potremo andare avanti e indietro nel nastro quando il gioco caricato da TAP nel caso giochi come Turrican, ce lo dovessero richiedere.

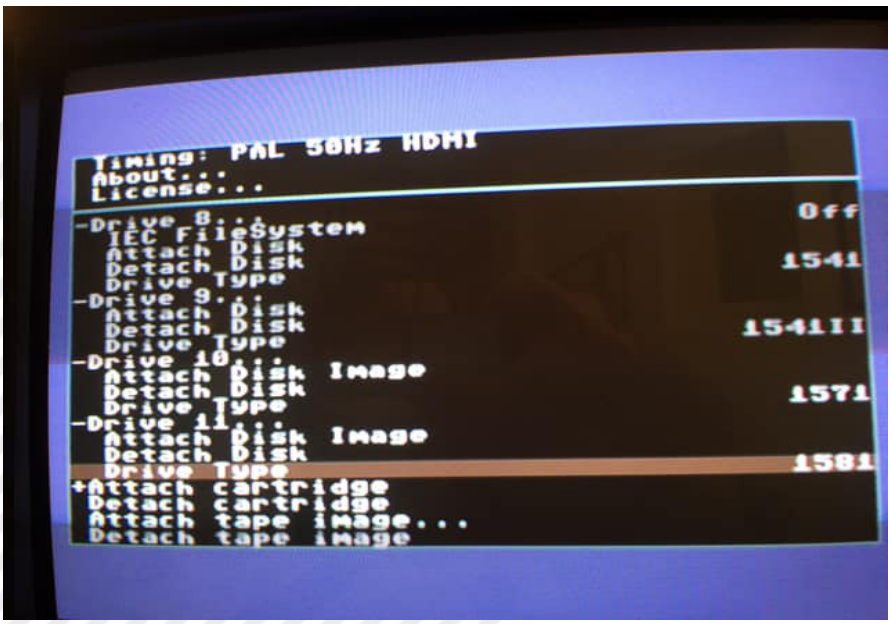
Al di fuori della configurazione da me indicata (Raspi + Keyrah + 1 o 2 joystick DB9), è possibile anche collegare 1 o 2 joystick DB9 usando



Parte del menu' con le opzioni disponibili

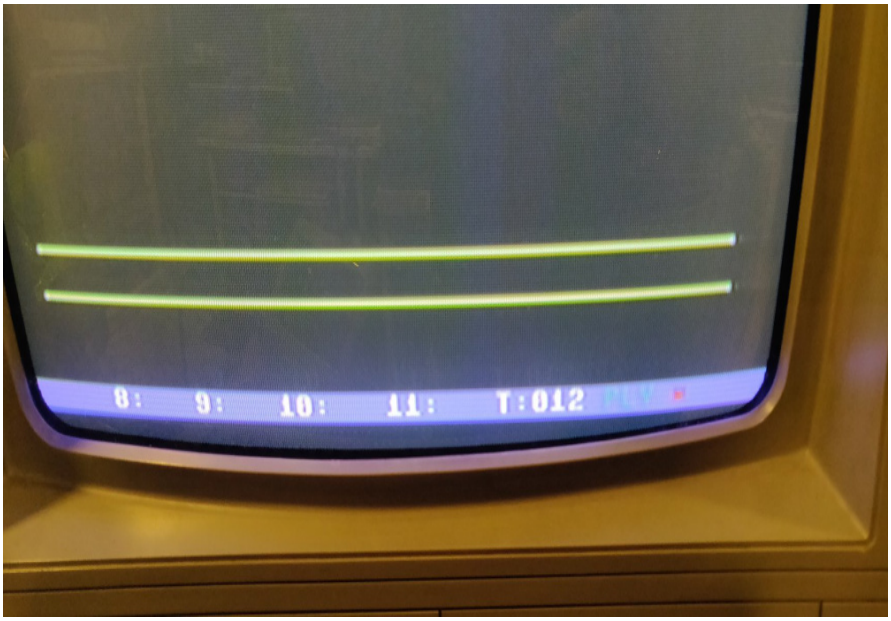
direttamente la piattina (chiamata GPIO) presente sul Raspi, ma è richiesta competenza tecnica e dimestichezza con il saldatore. In alternativa sta per essere resa disponibile (nel momento in cui vi sto scrivendo) un'interfaccia realizzata da amatori da mettere direttamente sul Raspberry per avere anche senza la Keyrah la presenza delle 2 porte DB9. In questo caso dovrete pero' trovare un'alternativa se vorrete comunque usare la tastiera del vero C64 e il led del case.

Se invece vi accontentate di usare dei gamepad o joystick usb, sappiate che sono completamente configurabili e che potrete assegnare ad alcuni dei loro pulsanti delle funzioni rapide senza dover avere sempre sotto mano la tastiera; E' possibile quindi gestire il WARP (on/off), l'apertura e la chiusura del menù, lo swap delle 2 porte joystick ed il comodissimo menù OSD TAP per gestire tutte le funzioni del Datasette come Play, Stop... E' persino possibile assegnare la direzione "SU" ad un pulsante di fuoco per usare il gamepad o il joystick USB come sulle console, ovvero con un pulsante di fuoco per sparare ed un altro per saltare (invece di muovere il joystick o gamepad verso l'alto). Piuttosto utile in giochi tipo Sam's Journey o Super Mario Bros 64.
Se si usano invece joystick DB9 in standard Atari, quindi con un solo pulsante di fuoco, e' possibile comunque assegnare queste funzioni rapide anche a delle combinazioni di tasti come C= + F1 o C= + F3 o C= + F5. Come sempre, per chi non ha la Keyrah, al posto del tasto C= (commodore) si puo' usare il tasto CTRL di sinistra.



Scelte possibili dei drive sulle varie unita' da 8 a 11

Nel momento in cui sto scrivendo l'articolo, grazie ad un suggerimento di un membro del gruppo Facebook dedicato a questo gioiellino di emulatore (vedi link in fondo all'articolo), si è scoperto che con l'aggiunta di un parametro particolare nel file config.txt è



La status bar e le sue funzioni

possibile variare l'aspect ratio dell'immagine per renderla il più simile possibile al C64 reale. Attenzione, è un parametro che va testato personalmente perche' su ogni modello di TV o monitor può dare risultati diversi.
Il parametro si chiama : framebuffer_aspect ed ecco alcuni dei valori che portano ad un'immagine più centrata e con meno bordi possibili (in base al tipo di schermo usato, una delle 3 opzioni suggerite potrebbe essere quella più adatta) :

framebuffer_aspect=0x00070009

framebuffer_aspect=0x00040005

framebuffer_aspect=0x00050006
(Questo è il parametro che sto usando sul mio 14" CRT in composito).

Fino adesso vi ho parlato dei punti forti del BMC64, purtroppo ci sono anche alcuni aspetti negativi da conoscere.
Essendo il Bare Metal un sistema senza Linux, non è in grado di riconoscere le periferiche USB di nessun tipo SE collegate a sistema già avviato, dovrete averle già connesse prima di accendere il Raspberry oppure sarete costretti a spegnere e riaccendere per averle attive e funzionanti.
Non potrete utilizzare nessun tipo di chiavetta per caricare i giochi perche' il BMC64 riconosce solo il contenuto della MicroSD. Tranquilli potrete fare tutte le sottocartelle che volete, tenendo ben a mente che non supporterà MicroSD superiori a 32GB e che la stessa dovrà essere formattata in FAT32 (niente EXFAT o NTFS).
Inoltre, essendo un'emulazione ultra accurata di un C64 e non un Linux con emulatore, non potrete usare la porta Ethernet o il Wifi per modificare il contenuto della microsd da remoto.

A questo punto, sarete curiosi di sapere come ottenere un sistema BMC64 funzionante. Vediamo quindi come realizzarlo velocemente con pochi semplici passaggi :

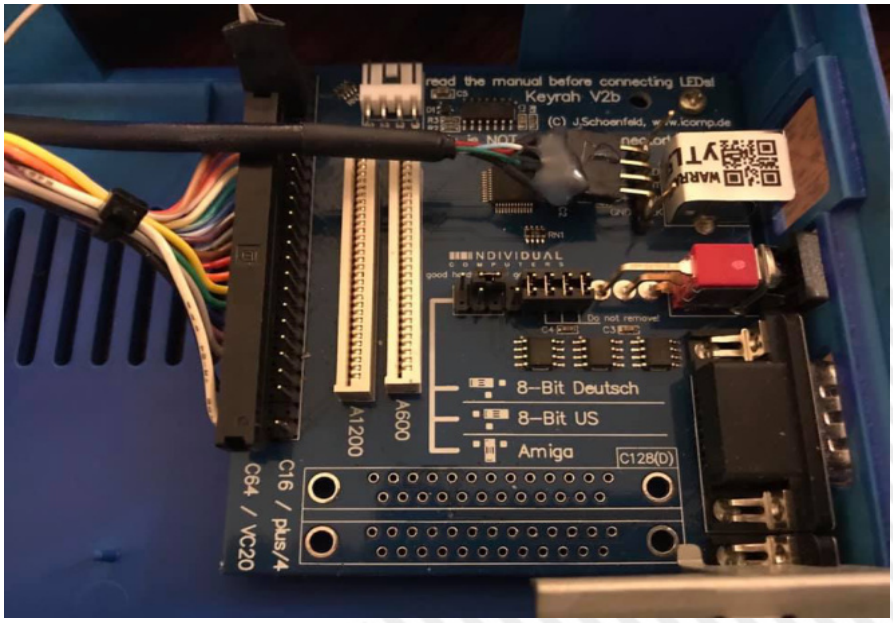
1) Scaricate l'ultima release disponibile dal sito ufficiale: http://accentual.com/bmc64/

2) formattate una microSD (massimo 32gb) in FAT32

3) decomprimete il file del punto 1 dentro la microSD del punto 2

4) aggiungete i seguenti file dentro la cartella C64 che vi troverete nella MicroSD (io li ho trovati dentro le sottocartelle del Vice del PC) :
- Basic (sottocartella C64 del Vice del PC)
- Chargen (sottocartella C64 del Vice del PC)
- Kernal (sottocartella C64 del Vice del PC)
- D1541II (sottocartella DRIVES del Vice del PC)
- Dos1541 (sottocartella DRIVES del Vice del PC) (se volete anche il 1541 primo modello)
- Dos1571 (sottocartella DRIVES del Vice del PC) (se volete anche il 1571)
- Dos1581 (sottocartella DRIVES del Vice del PC) (se volete anche il 1581)

5) aggiungete tutti i giochi che volete avendo cura di rispettare le



La Keyrah V2

sottocartelle già create :
- CARTS per le immagini .CRT
- DISKS per le immagini .D64, .G64, .D71 & .D81
- TAPES per le immagini TAP (attenzione che NON supporta i file .T64!)
- PRG per i file PRG

6) Nel caso di utilizzo di un monitor tramite uscita audio / video composita, aprite il file "CMDLINE.TXT" e modificate il parametro "machine_timing=pal" in "machine_timing=pal-composite", salvate ed uscite

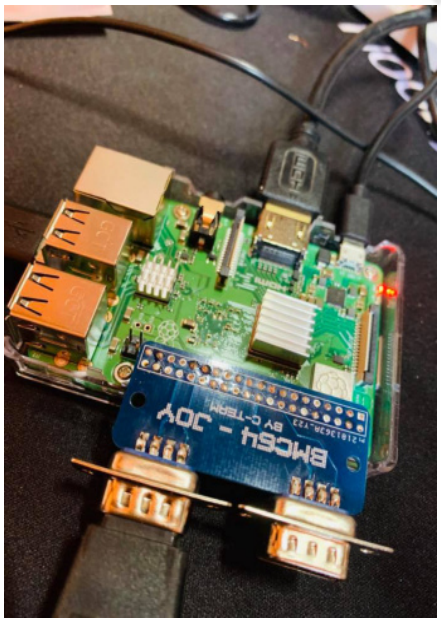
7) mettete la MicroSD nel Raspi (dopo aver fatto la rimozione sicura nel pc, mi raccomando!) e accendete!

Per caricare un file di un qualunque formato, sarà sufficiente premere F12 sulla tastiera USB (o i tasti C= + F7 nel caso di una Keyrah) per far apparire il menù dove abilitare tutte le funzioni dell'emulatore e montare le immagini nei drive o nel Datasette o caricare le cartucce.
E' possibile anche effettuare l'autostart (tramite la voce apposita nel menù) di un file in formato "PRG" e, anche se non dichiarato, è stato verificato che è possibile usare questa opzione per gli altri tipi di file (sono stati testati i formati CRT, D64 & TAP).

Per caricare piú velocemente i giochi e/o programmi memorizzati nei vari file immagine disco (D64, D71, D81 etc.etc.), possiamo montare una cartuccia Fastload (tipo la Epyx o la Action Replay) o cambiare al volo il kernal per usare il JiffyDos. Entrambe le opzioni sono salvabili per essere avviate in automatico, ma tenete presente che con il Jiffydos non potrete caricare i file TAP in quanto va ad occupare la zona di memoria della gestione del Datasette, esattamente come nel C64 reale. Da tenere a mente che alcune demo (e probabilmente anche qualche gioco), non accettano l'uso del Jiffydos e in alcuni casi nemmeno l'uso di altri drive oltre alla unitá standard 8.
Personalmente sto lasciando attiva di default la Action Replay 6.0 in quanto ha una compatibilitá maggiore della Epyx Fastload e non inibisce l'uso dei file TAP come fa invece il Jiffydos. Da una prova di caricamento da me effettuata e' risultato che il tempo di caricamento di un programma tramite la Fastload della Action Replay è all'incirca la metà rispetto allo stesso tramite il Jiffydos (che già di per se è molto più veloce rispetto ad un caricamento tradizionale).

Un paio di note sul piano tecnico. Il C64 reale utilizza una frequenza di

Un esempio di setup BMC64 * (Raffaele Merola)



Il BMC64 DB9JOY * (Fabrizio Lodi)

funzionamento sull'uscita video di 50,125hz. Il BMC64 in composito, esce esattamente sulla stessa identica frequenza, dandoci un'emulazione perfetta. La frequenza di uscita dell'HDMI invece è di 50hz tondi, con un piccolo scarto di 0,125 hz che per quanto poco, e' comunque presente.
La palette dei colori utilizzata è la PEPTO-PAL, a mio parere la più fedele a quella originale del Commodore 64.

Una piccola considerazione, siamo attorno alla fine di maggio 2018 ed è appena uscita la versione 1.8. e possiamo definirla come una versione molto avanzata (non ancora completissima di tutte le funzioni di cui è dotato il Vice 3.3 per PC da cui deriva, ma è già molto avanti). La cosa degna di nota è che, pur essendo il progetto BMC64 nato da pochissimi mesi e praticamente ancora in versione alpha appena 2 mesi fa (beta era già troppo da definire), lo sviluppo le nuove release sono molto frequenti. Lo sviluppatore e' molto attivo ed ascolta molto i suggerimenti degli utilizzatori e le segnalazioni di eventuali bug trovati (ha una specifica pagina GITHUB per poter ricevere queste segnalazioni). Personalmente lo trovo alquanto raro in uno sviluppatore oggigiorno, quindi i

miei complimenti più sinceri al creatore di BMC64! Tra le novità di questa release, abbiamo l'emulazione del mouse 1351 tramite un normalissimo mouse USB (per il Geos, ma non solo), l'ordinamento alfabetico dei file nelle cartelle, e la possibilità di implementare anche l'hard o il soft reset tramite gli hotkey per poter resettare il nostro emulatore in maniera molto agevole senza dover tutte le volte entrare nel menù (Per non rischiare reset accidentali ci chiederà comunque la conferma prima di effettuare il reset)
L'aggiornamento è operazione molto semplice, basta scaricare il nuovo file ZIP dal sito, scompattarlo direttamente nella nostra MicroSD avendo cura di sovrascrivere tutti i file, ed ecco la nuova release!

Lo sviluppatore ha inoltre appena rilasciato una versione molto preliminare del BMC128 e tra qualche settimana (probabilmente nel momento in cui leggerete questo articolo sarà già accaduto) dovrebbe rilasciare anche il BMVic20! Personalmente non vedo l'ora di poter mettere le mani su quest'ultimo, immaginate tutto il setup attuale del BMC64 (case, tastiera C64, Keyrah, joystick DB9 e monitor CRT) ma usato per il Vic20!

Per chi fosse interessato a saperne di più, esiste anche un gruppo Facebook creato inizialmente per il BMC64 ed in seguito modificato per accogliere altre versioni (che stanno uscendo o dovranno uscire) per conoscere e condividere tutto quello che riguarda questo tipo di emulatore che ha portato davvero ad un piccola rivoluzione nel mondo dei Raspberry

Conclusioni
Se qualcuno si stesse chiedendo se vale la pena farsi un proprio BMC64 oppure no ci sono a mio parere troppe variabili per poter dare una risposta secca tipo SI o NO…
Nel mio caso e' stato piuttosto semplice. Avevo già una Keyrah da tempo, avevo un Raspberry che mi avanzava nel cassetto ed un C64C con la scheda madre completamente guasta che conservo tutt'ora per i pezzi di ricambio. Avevo anche il cavo audio / video composito e mi avanzava pure una microsd per le prove, quindi ho potuto iniziare velocemente ad impostarlo ed a usarlo. Ho dovuto comprare solamente la prolunga microSD -> SD e il cavo di alimentazione USB con interruttore per maggiore comodità. Ma se non avete nulla di tutto questo, forse vale la pena valutare il costo complessivo della soluzione che vorrete adottare. Mi raccomando, nel caso decideste di metterlo dentro un case di un C64 biscottone o C64C, dovrete trovarne uno guasto se non già in vostro possesso. Mi auguro che in nessun caso andreste mai a rimuovere una scheda madre funzionante da un C64 per farlo diventare un C64 emulato.

Riferimenti :
Sito ufficiale del BMC64 (per info e download):
http://accentual.com/bmc64/

Gruppo Facebook dedicato al BMC64 & C.
https://bit.ly/2WcDd5Y

Dove acquistare la Keyrah V2:
https://www.vesalia.de/e_keyrahv2.htm

Per acquistare la memoria MicroSD, una prolunga MicroSD -> SD, il cavetto USB con interruttore, la prolunga del LED nel caso decideste di usare il case di un C64C invece di quello del C64 biscottone (basta una prolunga per ventolina dei pc maschio / femmina 3 pin) e il cavo video composito per Raspberry 2 e 3 :
https://www.amazon.it

Si ringraziano Raffaele Merola, Bob Rob & Fabrizio Lodi per la gentile concessione sull'uso di alcune loro fotografie

Adesso che sapete tutto o quasi di questo rivoluzionario tipo di emulatore, non mi resta che attendervi nel gruppo Facebook e augurarvi buon divertimento!

Altro esempi di setup BMC64 * (Bob Rob)

L'angolo dell'FPGA update

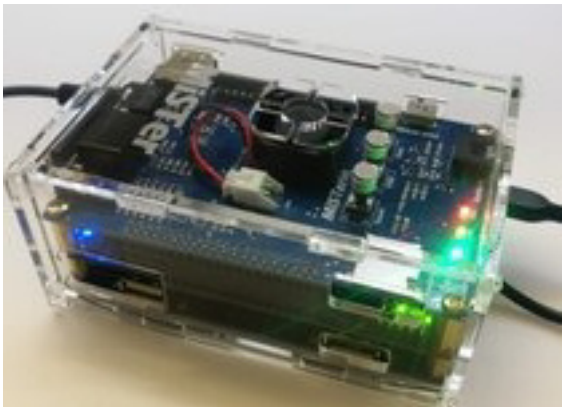
di Roberto Lari

Ciao a tutti,
nel numero 13 di Retromagazine vi avevo fatto scoprire e conoscere un mondo nuovo, ovvero i sistemi FPGA nella fattispecie quelli chiamati Mist & Mistica (se avete abitato sulla Luna fino adesso e vi siete persi quell'articolo, vi suggerisco di leggerlo prima di leggere l'update di oggi, lo trovate a partire dalla pagina 11, e dopo proseguite con questo articolo).

Sono passati 3 mesi da quando è stato scritto quell'articolo e ci sono un sacco di novità riguardo il mondo FPGA, ma andiamo con ordine:

Sono stati rilasciati uno o più firmware aggiornati, che migliorano il funzionamento globale.
Sono stati rilasciati diversi core aggiornati, in particolare (citiamo solo quelli più importanti) hanno aggiornato i core dell'MSX e del PC Engine facendoli diventare (finalmente) compatibili con la FAT32. Questo significa che, volendo, potremmo avere tutti i core, compresi questi 2 su una singola SD, invece di doverne usare altre (precedentemente dovevano essere inserite in schede formattate fat16 e soffrivano del limite dei 2 gb per SD)
E' stato implementato il supporto degli hard disk virtuali per alcuni sistemi tra i quali MSX, BBC Micro e Sinclair QL. Questi file in formato .VHD vanno messi nella root della nostra SD con dei nomi specifici per far si che il core li possa riconoscere e caricare all'avvio (MSX.VHD, QL.VHD & BBC.VHD)

Inoltre dopo aver chiesto a diversi sviluppatori di core FPGA di Mist & Mistica di implementare una funzione tanto cara a moltissimi di noi, che non si sa perché ancora mancasse, ecco finalmente l'implementazione della lettura dei file TAP per il nostro amato Commodore 64 (e in seguito anche per Vic 20 e C16 / Plus4) con tanto di tasto rapido sulla tastiera (per la precisione Pagina SU) per effettuare il Play e lo Stop della lettura del nastro virtuale. Inoltre, il nostro amico Antonino Porcino ha pure aggiunto la possibilità di leggere e scrivere (tramite i jack EAR & MIC presenti sulla Mistica stessa) da e sulle cassette magnetiche reali, usando un vero registratore a cassette. In alternativa, solo per leggere i file TAP, è possibile usare un simulatore come il MaxDuino (attenzione che attualmente il firmware del MaxDuino non permette di caricare i file TAP di nessun Commodore 8 bit).



Mister preassemblato di Manuel

Antonino Porcino ha scoperto ed in seguito acquistato la Mistica proprio grazie all'articolo precedente sugli FPGA ed ha già apportato (previa autorizzazione di chi gestisce la manutenzione dei core) diverse migliorie anche al core del C64 stesso e del Vic20. Non contento, sempre sul Vic 20 e sul C16 / Plus 4 ha migliorato il supporto all'uscita composita, poiché prima non era funzionante al 100%, ed è stato pure inserito tra gli sviluppatori ufficiali del core FPGA.

Un'altra cosa che ho scoperto da poco (si poteva fare anche prima ma non avevo mai verificato di persona) è la possibilità per alcune console di far avviare un gioco in automatico proprio come fanno le console che nascono con un gioco già in rom. Ad esempio il Sega Master System 1 con al suo interno Hang on, avvia il gioco non appena si accende la console senza inserire nessuna cartuccia. E' possibile fare la stessa cosa anche con il Mist / la Mistica. Basta copiare qualsiasi gioco nella root e rinominarlo SMS.ROM (per il core Sega Master System); in questo modo il gioco partirà in automatico all'avviamento del core. Forte non trovate?

Nel momento in cui sto scrivendo hanno apportato una modifica al core del C16 / Plus 4. Normalmente il core include anche la o le rom necessarie al funzionamento della macchina riprodotta; in questo caso le rom sono sono state rimosse dal core per avere più spazio per implementare funzioni aggiuntive. E' stato quindi aggiunto il supporto ai SID (sul vero C16 lo stesso SID del C64 è inseribile tramite una cartuccia e alcuni giochi e demo lo sfruttano). Ovviamente le rom a questo punto vanno aggiunte tramite un file apposito da copiarsi nella root della SD (in questo caso C16.ROM) che il sistema provvederà a caricare in automatico subito dopo il caricamento del core vero e proprio.

Il Mister

Nel precedente numero di RetroMagazine (il 14) invece il nostro amico Juri Fossaroli ci ha parlato di un prodotto simile ai nostri Mist & Mistica, non voglio dilungarmi sulle caratteristiche tecniche di cui vi ha già parlato lui (diciamo solo che è una versione più capiente in termini di spazio FPGA che permette di usare alcuni core che fisicamente non possono essere riprodotti Mist & Mistica in quanto troppo complessi come lo SNES, il Neogeo, lo Sharp X68000 o il 80486 da usare per Msdos , Windows 3.xx e perfino Windows 95 ad esempio), ma ci tengo a sottolineare una cosa che reputo alquanto importante, ovvero che i core che sono presenti in entrambi i sistemi (Mist /Mistica vs Mister) funzioneranno allo stesso identico modo, non andranno più velocemente o cose simili, però grazie alla maggiore dimensione del circuito FPGA, sul Mister è possibile implementare ulteriori funzioni aggiuntive, come per esempio hanno implementato l'uso del controller originale della Wii per simulare la pistola Zapper sul core del Nintendo NES per i giochi che la richiedono come Duck Hunt e tanti altri.

Infine per concludere, vi segnalo che il produttore della Mistica ovvero Manuel Fernández Higueras, offre anche un Mister già assemblato e pronto all'uso (questo anche grazie ad una SD da 8GB che fornisce in regalo con il software di funzionamento preinstallato), dotato di tutte le uscite video (HDMI sulla scheda madre DE10 Nano & VGA/RGB tramite addon), della SDRAM da 32mb aggiuntiva (da lui testata e certificata per funzionare fino a 167mhz, ben oltre la frequenza di funzionamento richiesta da tutti i core e che è fondamentale per il corretto funzionamento della maggior parte degli stessi), di un hub 7 porte USB, di un case trasparente dello stesso materiale di quello usato per la Mistica, e di una ventolina per i core più impegnativi (tipo il 486 per esempio), con un interruttore per decidere se e quando accenderla.

Inoltre ha anche lui il jack input EAR per permettere il caricamento da sorgenti esterni (registratore a cassette / Maxduino), esattamente come avviene con la Mistica.

La mancanza delle 2 porte DB9 (per me molto importanti per il feeling che trasmette l'uso dei veri joystick C64 / Amiga / Atari) non è più un problema in quanto esiste un adattatore da USB a DB9 che ho provato personalmente e che funziona benissimo senza lag (vedi link per l'acquisto nei riferimenti)

Altra cosa importante, per chi optasse per il Mister per usarlo connesso ad un classico monitor o tv HDMI, esiste la possibilità di settare l'immagine a 16:9 o 4:3 e ci sono moltissimi tipi di opzioni per simulare la visione di un CRT (come le scanline ad esempio) e sono combinabili tra loro, certo non sarà un'immagine identica ad un CRT vero e proprio ma è sicuramente meglio di niente!

Ultima ma non ultima, per ogni core vi è la possibilità di rimappare i tasti di qualsiasi controller usb vi colleghiate in tempo reale, opzione che ritengo decisamente utile.

Personalmente, se dovessi decidere di acquistarne uno, sicuramente opterei per questo modello, sul suo sito ufficiale ci sono tutte le caratteristiche tecniche, il prezzo completo di tutto l'insieme e le foto che ci mostrano il prodotto finito ed assemblato.

Cavo RGB SCART per Mistica

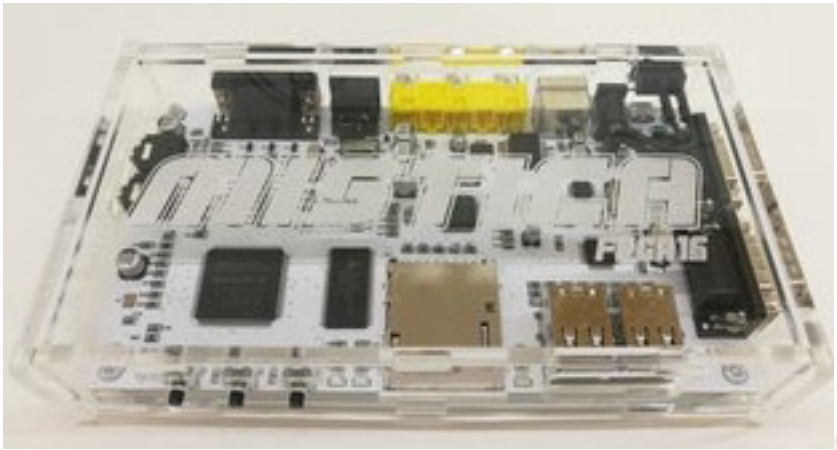
Riferimenti:

https://manuferhi.com/c/mistica-fpga16
(Mistica e cavo video RGB SCART di Manuel)

https://manuferhi.com/p/mister-fpga
(Mister pronto e preassemblato sempre di Manuel)

https://www.facebook.com/groups/295501721152384/
(Gruppo Facebook dedicato a questi apparecchi FPGA)

http://tiny.cc/1kbc7y
(link per adattatore da USB a DB9)



Mistica

"Come far rinascere un cabinato..."

di Marco Petri

Le righe successive, derivano dal mio personale manuale utente che ho dovuto redigere onde tenere traccia di tutte e caratteristiche del mio cabinato in caso di manutenzione. A onor del vero infatti, devo dire che configurarlo da zero, non è proprio una cosa semplicissima.
Mi sono accorto da tempo inoltre che non si trova in giro un vademecum univoco su come mettere su un sistema arcade dall'inizio alle fine, quindi ho ben pensato di farne uno io: in questo modo i lettori non dovranno andare a pescare chissà dove informazioni frammentate sparse per il web.
Sono doverose però delle premesse:

- Non volevo spendere soldi per un PC nuovo per far girare dei giochi di 30-40 anni fa;
- L' articolo si rifà al restauro di un cabinato d'epoca con CRT e non alla costruzione di uno ex-novo;
- L' hardware usato, se configurato bene, è più che sufficiente e si può rimediare a poco o addirittura gratis;
- Sono convinto che non serve avere l'ultima versione di Mame/ Groovymame. Mi trovavo già molto bene con la versione 0.140 del 2010 visto che faceva girare tutti i giochi di cui avevo bisogno. L'idea è quella di realizzare un sistema "embedded" che funziona, da conservare finchè dura. Della serie: non vi interessa se il bancomat che usate si regge su Win10 o XP o il basic del Commodore64: basta che funziona, no? Io sono di questo avviso, non me ne vogliate;
- Perdonate una certa trattazione "maccheronica" degli argomenti: non sono propriamente uno smanettone perchè preferisco passare più tempo a usare i mezzi piuttosto che a configurarli.

Premesso ciò, cominciamo dall'hardware:

LATO CABINATO:

Come detto, l'oggetto del desiderio (Fig. 1) è un cabinato generico Jamma (non lasciatevi ingannare dal logo del gioco Robotron 2084, non è un cabinato dedicato!) il cui anno di produzione non sono mai riuscito a definire. Tuttavia, considerando i materiali di costruzione, il font utilizzato per la scritta "game" sul davanti e la tipologia dei tasti, direi che

Fig.1 – Il cabinato dall'esterno.

potrebbe risalire al finire degli anni '80. In questo caso, se qualcuno avesse delle informazioni in più, sarei davvero grato se me ne desse notizia al nostro indirizzo mail.

Precisazione: le grafiche laterali, il neon e la grafica del marquee (quella illuminata dal neon in alto), sono state autoprodotte da me, sulla scorta di grafiche e loghi dell'epoca. Il cabinato originariamente, si presentava infatti tutto grigiastro, davvero bruttino, e il vetro del marquee aveva un vecchio adesivo tagliatl a metà e ingiallito.
Vedremo in seguito come fare tutto.
Altre caratteristiche sono le seguenti:

- ► monitor CRT, presumibilmente Videocolor, a 15 KHz;
- ► audio mono tramite unica cassa acustica centrale;
- ► Scheda di controllo del CRT MTC9000;
- ► Alimentatore Hantarex US300;
- ► Scheda di interfacciamento col PC: JammaASD *(http://www.arcadeitalia.net/ jammasd.html)*

LATO HARDWARE PC:

- ► AMD Sempron 2800 a 1,6 GHz;
- ► 2GB di Ram;
- ► HD 200 GB SATA;
- ► Scheda grafica ATI Radeon 7500 64 GB;
- ► Filtro rumore da applicare tra l'uscita audio del PC e l'ingresso audio della JammASD per ridurre il fruscio sulla cassa del cabinato che

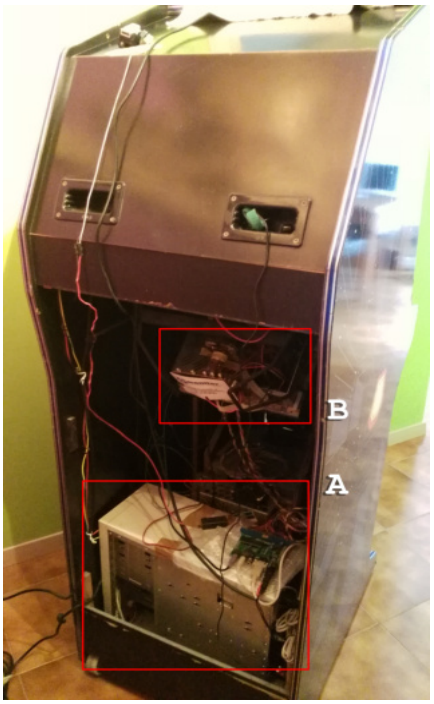


Fig.2 - Riquadro A: computer con scheda JammASD sopra; Riquadro B: retro del tubo catodico con scheda di controllo -

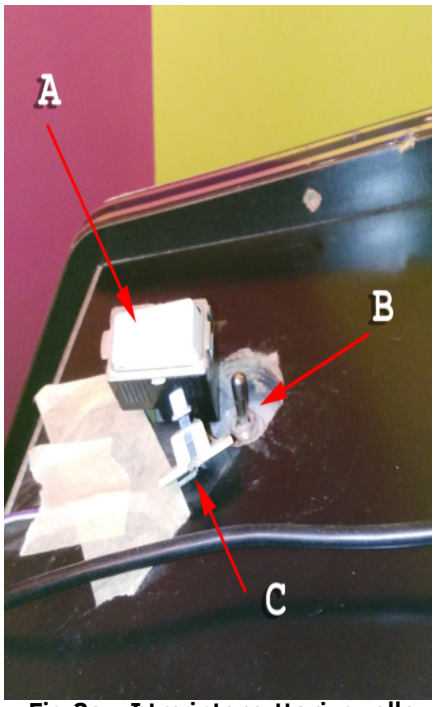


Fig.2a – I tre interruttori: quello generale A, quello del cabinato/ CRT B e quello del computer C

altrimenti sarebbe davvero troppo! (Non capisco da dove viene, qualche interferenza del cavolo...).

Come si nota, il PC non è certo tutta questa potenza ormai ma ha dimostrato di poter egregiamente eseguire tutti i giochi del periodo. Il case è un po' ingombrante, è vero, ma entra perfettamente anche se per poco.

Per quanto riguarda il lato "accensione", su internet ci sono degli schemi per accendere il tutto tramite un unico pulsante ma ho preferito mantenere separati i comandi (interruttore generale, del cabinato e del computer) per diversi motivi: 1) la scheda JammASD rimaneva sotto tensione con il led acceso; 2) cosa più importante: il computer ci mette un po' a trasformare il segnale video VGA nel segnale a 15 KHz accettato dal CRT. Durante questo frangente, ho ritenuto opportuno non rischiare di sottoporre il prezioso tubo catodico a un segnale video non adeguato (anche se la JammaASD dovrebbe fungere da filtro).

Quindi la procedura prudenziale che ho adottato è la seguente: accendo l'interruttore generale, poi quello del computer e qualche secondo dopo (quando sono sicuro che il segnale a 15 KHz è attivo) accendo il cabinato.

Per quanto riguarda il software:

♦ Mame/Groovymame 0.170 *(http://mamedev.org/oldrel.html*

https://drive.google.com/drive/folders/0B5iMjDor3P__aEFpcVNkVW5jbEE) ;

♦ CRT Emudriver 1.2b (Catalyst 6.5) for Windows XP-32 + VMMaker + Arcade_OSD 1.4b *(http://geedorah.com/eiusdemmodi/forum/viewtopic.php?id=65);*

♦ Mala Front-end 1.74 *(http://malafe.net/index.php?page=download);*

♦ DW Jukebox 3.4.1 *(http://dwjukebox.com/);*

♦ Windows XP Service Pack 3, 32 bit.

Scegliere un cabinato da restaurare: il problema del tubo catodico

Cosa è importante da vedere quando si compra un cabinato da restaurare? Credo che il componente più importante sia proprio il vecchio e caro CRT. Non mi ricordo quant imprechi ho gridato contro questa tecnologia dei monitor quando, all'epoca,

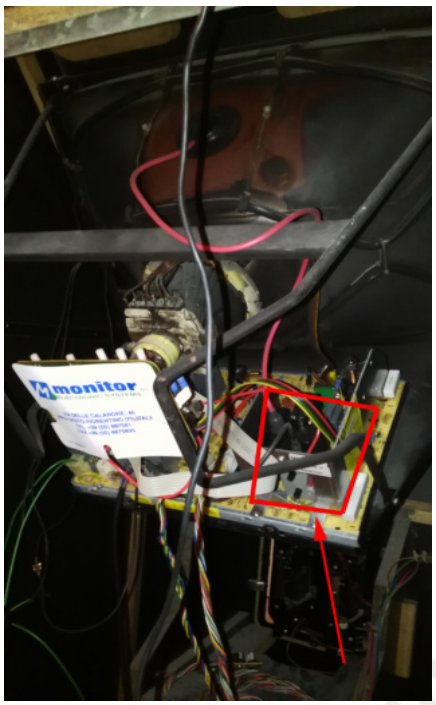


Fig.3 - Trasformatore di riga.

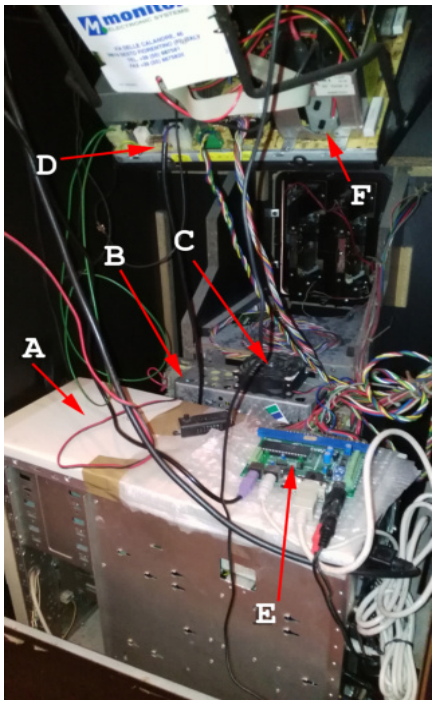


Fig.2b – A: Computer; B: Trasformatore Hantarex US300; C: Ventola del trasformatore; D: Scheda di controllo del CRT; E: Scheda di interfaccciamento JammASD; F: Trasformatore di riga.

dovevo traslocare. I CRT, lo sappiamo bene, erano ingombranti, affamati di corrente, scaldavano, emanavano radiazioni ma se volete godervi la piena esperienza dei giochi arcade di una volta, non potete farne a meno: la cromia e quella sorta di antialiasing naturale che permetteva il tubo catodico, ve la potete scordare con un moderno LCD, pur con tutti i filtri software attivati che volete.

Non voglio provocare chi è a favore di quest'ultimo, ma non me ne vogliate: mettere un LCD su un cabinato d'epoca è come mettere dei moderni cerchi in lega su un'auto degli anni '40! Un delitto. Oltretutto sta diventando sempre più difficile trovare degli LCD con formato 4:3. E poi volete mettere il "tunzzz", il classico ronzio del circuito di smagnetizzazione (DEGAUSS), che ci riporta direttamente a quei tempi?

Purtroppo, i CRT sono soggetti a esaurimento del cannone elettronico (catodo), di fatto il generatore del fascio di elettroni che disegna l'immagine sulla griglia dei fosfori. Una volta, quando si

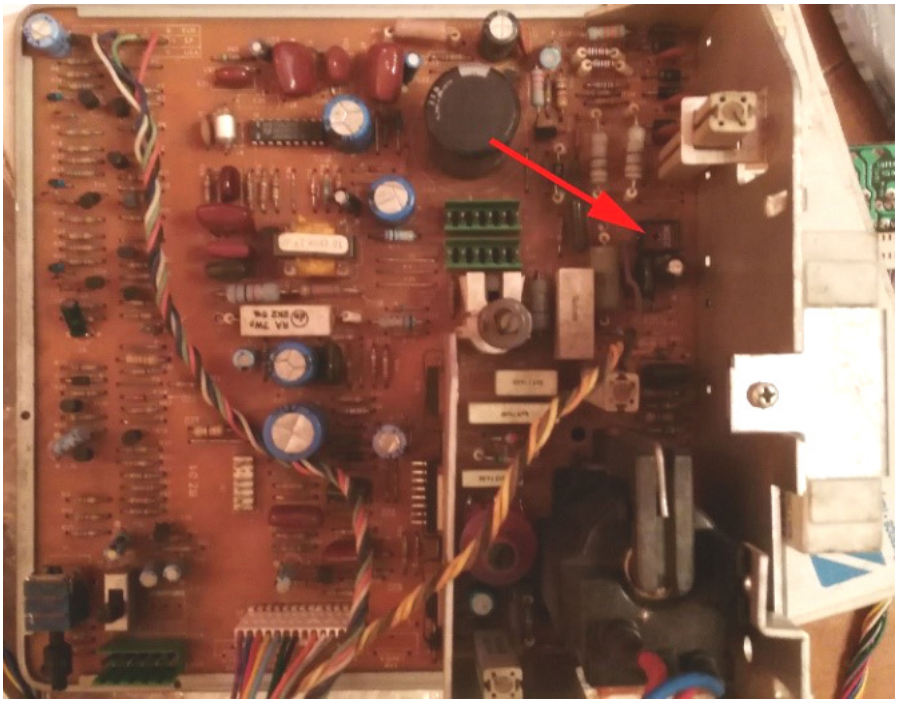


Fig.4 - Regolazione luminosità su scheda di controllo CRT, MTC 9000.

verificava una situazione simile, tanto era il costo di un buon CRT, che si preferiva rigenerarlo: si tagliava il collo del tubo di vetro, si sostituiva il catodo, si ripristinava il vuoto all'interno del tubo e si risaldava il vetro. Un lavorone che poteva essere fatto solo in fabbrica. Non esistendo più alcuna di queste, ormai quando un CRT si avvia verso l'esaurimento, non c'è più niente da fare.

Come vagliare dunque lo stato di CRT che si sta per comprare?

Fig.5 - Schedina esterna di controllo del CRT

Difficile definirlo. Una strada può essere quello di "sparare" la luminosità al massimo tramite il potenziometro che si trova sul trasformatore di riga (EAT) e che solitamente è contrassegnato dal nome "SCREEN", come indicato nelle Figg.2b e 3.

Questa è la luminosità di base che può ulteriormente essere regolata da un altro potenziometro si trova sulla scheda che contiene la logica di controllo del CRT. Purtroppo non c'è una posizione univoca. Nel caso della scheda MTC 9000 che ho trovato, la luminosità si poteva regolare direttamente tramite un cacciavite e il potenziometro indicato dalla freccia nella Fig.4:

mentre, nel caso delle più moderne schede di controllo della Semio/Monitor, i controlli della luminosità, contrasto, vertical shift etc., possono essere trovati su una comodissima schedina esterna attaccata tramite basetta come quella della Fig.5:

Attenzione: le schedine flottanti di controllo come quella dell'immagine precedente, possono introdurre delle interferenze per via della lunghezza dei fili non schermati.

Alla fine della discorso: cercate di individuare il controllo della luminosità sulla scheda che vi trovate davanti; unitamente al comando SCREEN di cui sopra, si dovrebbe poter ottenere uno schermo completamente bianco o quasi: se c'è qualche problema di esaurimento, probabilmente non ci riusciremo.

Altra questione riguardante il CRT: le macchie di colore. E' probabile che uno schermo a tubo catodico presenti delle macchie di colore, solitamente di colore magenta/fucsia/rossastro. In genere sono dovute a una non perfetta smagnetizzazione dello schermo, il che potrebbe significare che il circuito del DEGAUSS non va. A me è capitato quando ho testato uno schermo che non accendevo da molto tempo, staccato dalla logica di controllo.

Non dovrebbe dunque indicare un difetto permanente al tubo e comunque vi si può porre rimedio muovendo una calamita davanti

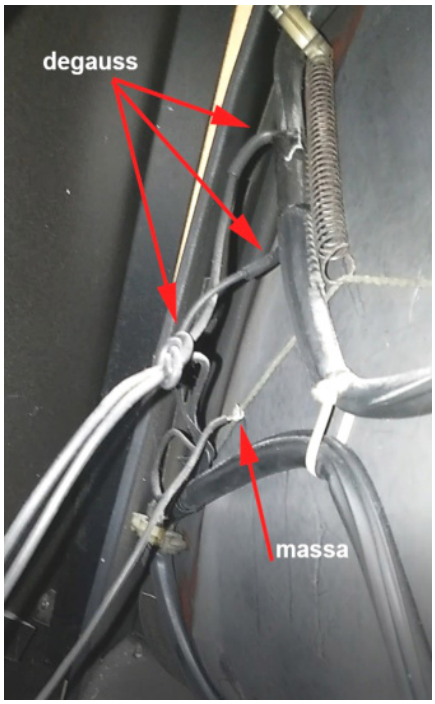


Fig.6 - Collegamento della gabbia di smagnetizzazione attorno al tubo catodico

allo schermo, tenendola a contatto, come se lo doveste "massaggiare".

L'ultima procedura da effettuare poi, è quella della taratura del colore e della luminosità. Per effettuarla, ho impiegato dei gradienti di colori e di toni di grigio come il vecchio monoscopio. Ovviamente il nero deve tendere al

nero più profondo e il bianco al bianco più brillante possibile. So che questo è modo a "occhiometro", ma non avevo certo voglia di spendere soldi per un colorimetro.
Tenete sempre conto, che una luminosità maggiore corrisponde a un uso più intenso del cannone elettronico quindi diminuisce la vita del CRT che comunque, se in buone condizioni, è di decine di anni (a seconda delle ore d'uso).
Detto ciò, proseguiamo con la parte software.

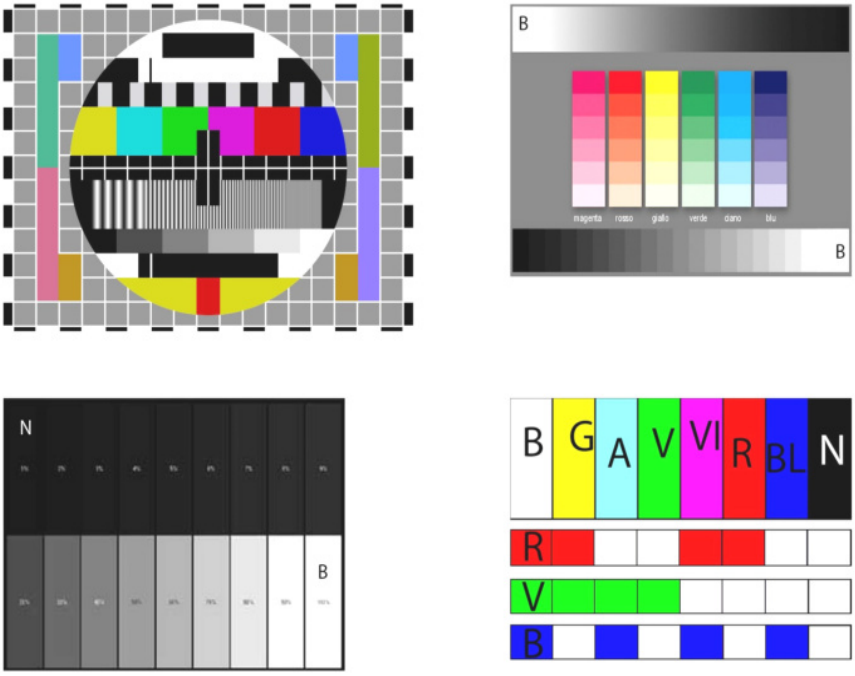


Fig.7a – Alcuni gradienti di colore e toni di grigio usati per la taratura del CRT.

Mame/Groovymame ed aggiornamento romset: Quale versione?

La mia esigenza era quella di provare il Groovymame, una versione modificata di Mame ottimizzata per l'uso coi CRT. Inoltre Groovymame, prima dell'avvio del gioco, indica sia la risoluzione/frequenza originale del gioco, sia quella scelta da lui stesso per aprirlo: in questo modo siamo sicuri di avere un'esperienza d'uso realmente "pixel perfect", ossia di poter giocare alla stessa risoluzione del gioco da bar.
In effetti, ho notato che rispetto a Mame normale, Groovymame riesce a scegliere meglio la frequenza e la risoluzione per ottenere un risultato visivo più simile all'originale.
Fino a poco tempo fa, ho usato il MameUI32 0.140 del 2010 che andava bene ma aveva cominciato a darmi dei problemi di audio in alcuni giochi quindi ho pensato di provare altro.
Bisogna dunque individuare la versione di Mame/Groovymame adatta al nostro hardware non solo a quello che vogliamo ottenere.
Nel mio caso non potevo puntare sulla versione 0.194 (o successive) per due motivi:
1) incompatibilità col front-end Mala (che però di seguito ho risolto, almeno con la versione 0.170);
2) incompatibilità della scheda grafica: dalla 0.171, Mame (e conseguentemente Groovymame) non supporta più il Direct Draw e quindi avrei dovuto cambiare la scheda grafica: non ci ho pensato neanche visto che quella che tutt'ora uso, va alla grande.
Ho dunque puntato su Mame/Groovymame 0.170 ma il mio romset (ossia la raccolta delle rom dei giochi) era adatta allo 0.140 che usavo precedentemente e quindi si imponeva un aggiornamento.
Dal sito Emuparadise mi sono scaricato i pacchetti di aggiornamento e ho seguito questo tutorial
https://forums.launchbox-app.com/topic/30220-how-to-update-your-mame-romset-to-a-newer-version-with-clrmamepro/

da usare con CLRMamePro *(https://mamedev.emulab.it/clrmamepro/).*

Questa è stata la fase più oscura dell'intero processo perché in effetti non so se il romset sia stato aggiornato in modo ottimale visto

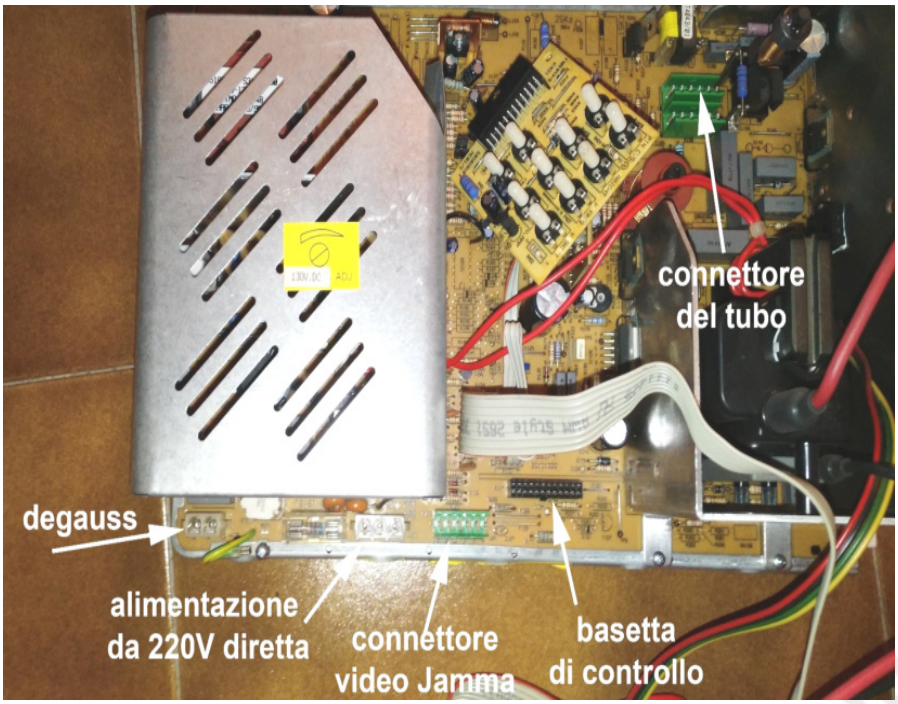


Fig.7 -Scheda di controllo CRT, Semio/Monitor

che diverse rom continuano a essere segnate in rosso da Mame. Tuttavia la maggior parte dei giochi funziona quindi sono soddisfatto così. Se poi ci dovesse essere qualcosa che non va nello specifico, mi scaricherò solo la rom di quel gioco che dà problemi e via la paura.
Un'altra verifica che si può fare è usare RomCenter
(http://www.romcenter.com/)

che dà una panoramica sul database di rom che usiamo, aiutandoci a correggere un gioco che magari non funziona per il fatto che possiede un nome sbagliato (seguite però dei tutorial come questo:
https://www.youtube.com/watch?v=1JtIh5u2Ko4).

Ok, dopo aver (bene o male) aggiornato il mio romset alla versione 0.170 ed aver scaricato sia Groovymame che Mame entrambi 0.170 a 32 bit, era il momento di preparare il PC.
Perché a 32 bit? Perché tanto sul mio PC ho 2 GB di ram e non avrei avuto vantaggi nell'uso dei 64 bit. Windows XP a 32 bit SP3 è leggero, stabile e soddisfa tutti i miei bisogni. Va solo alleggerito di alcune cose che riporto qua di seguito:

- Eliminare gli odiosi fumetti di avvertimento (del tipo: "questa risoluzione è troppo bassa" etc, etc)
(https://support.microsoft.com/it-it/help/307729/how-do-i-disable-balloon-tips-in-the-notification-area-in-windows-xp-o);
- *Eliminare il logo iniziale di caricamento*
(http://www.ottimizzazione-pc.it/togliere-logo-windows-avvio-del-computer/);
- Eliminare le info di chiusura
(https://www.ilsoftware.it/forum/viewtopic.php?t=58128);
- Eliminare la schermata di logon
(https://verytech.smartworld.it/come-eliminare-la-richiesta-di-nome-utente-e-password-allavvio-di-windows-xp-115020.html#steps_5);

```
"""""""""""""""""""""""""""""""""""""""""""
in
#
# CORE SEARCH PATH OPTIONS
#
rompath                 inserire il percorso della cartella roms ad es. C:
\mame\roms

in
#
# CORE MISC OPTIONS
#
skip_gameinfo            1 (disabilita i messaggi di eventuali
malfunzionamenti o problemi delle roms)

in
#
# CORE MKChamp OPTIONS
#
disable_nagscreen_patch   0 (disabilita il messaggio che per avanzare ci
chiede di digitare ok o Dx/Sx col joy)
disable_loading_patch     0 (disabilita la scritta loading quando lanciamo
una rom)

in
#
# OSD VIDEO OPTIONS
#
video                   ddraw (scegliamo di utilizzare direct draw che su
crt da la miglior resa)
"""""""""""""""""""""""""""""""""""""""""""
```

Riquadro 1 - Configurazione Mame.ini generato da groovymame

Ho anche tolto tutti i fronzoli grafici e riportato lo stile delle finestre di Windows 2000: basta andare in Pannello di Controllo>Sistema>Impostazioni (del riquadro Prestazioni) e scegliere "Regola impostazioni per ottenere le migliori prestazioni".
La schermata blu "WindowsXP" iniziale invece si può togliere andando su Pannello di Controllo>Account Utente e scegliere "Cambia modalità di accesso e disconnessione" e deselezionando la voce "Usa la schermata iniziale".

Installare i drivers della scheda grafica e generare le modelines

Ora è il momento di installare i CRT Emudriver per la scheda grafica che, tra l'altro, abiliteranno l'uscita video a 15 KHz pronta per essere visualizzata sul monitor CRT del cabinato.
Scarichiamo anche i CRT Tools (WMMaker e Arcade OSD), il primo serve per creare le modelines, cioè le risoluzioni tipiche dei vari giochi, mentre il secondo per l'affinamento e il test delle stesse qualora ci fossero dei problemi di sincronizzazione con il monitor del cabinato. Io ho scelto di rimanere alla versione 1.4b ma sono presenti anche quelli della versione 2.0. WMMaker2.0 ha introdotto una interfaccia utente ma per me era più che sufficiente la versione 1.4b.
Ora creiamo una cartella dove mettiamo il nostro Mame, il Groovymame e le rom dei giochi. Le rom ovviamente vanno nella cartella Roms; è utile poi creare anche una cartella ini dove andranno posizionati i file .ini di ogni gioco (spiegherò di seguito).
Nella cartella del Mame avremo ovviamente il mame.exe ma questo non ci interessa; quello che voglio è usare il Groovymame e quello che devo fare è metterlo nella stessa cartella del Mame solo che, per praticità e per distinguerlo dal Mame originale, lo rinomino in groovymame.exe .
A questo punto il mame.exe si può cancellare ma lo tengo per delle prove, casomai il groovymame dovesse darmi dei problemi.
Se nella cartella è presente un mame.ini, lo cancello perchè ne

creo uno a partire dal groovymame.exe stesso. Per fare questo apro il pannello dei comandi di Windows cliccando su Esegui (su WinXP) e scrivendo cmd. Questa operazione può essere fatta su anche su Win10, basta cercare e avviare il programma cmd.exe.
Portiamoci nella cartella di Mame/Groovymame e scriviamo il comando groovymame.exe –cc. In questo modo avremo creato un file mame.ini derivato da Groovymame. Ora lo possiamo modificare nel modo indicato qui di seguito (ovviamente per la versione 0.170!) *https://www.arcademania.eu/viewtopic.php?f=41&t=28825 .*

Riporto comunque le info necessarie. Fino a groovymame0.170, nel mame.ini generato da groovymame dovremo editare solo alcune linee
(vedi Riquadro 1)

Per i neofiti, ricordo che i file .ini possono essere editati tramite il comunissimo Notepad (Blocco Note) di Windows.
Da aggiungere che questa versione di mame è compilata con il supporto agli high score abilitato, dovremo ricordare solo di creare la cartella hi dove è l'eseguibile mame e di posizionare il file hiscore.dat relativo alla versione 170 accanto all'eseguibile mame.
Una volta modificato il file mame.ini, non rimane che creare le modelines tramite WMMaker ma bisogna modificare il file wmmaker.ini prima dell'operazione. Di seguito le voci da cambiare:
1 - MameExe: inseriamo il percorso di Groovymame;
2 - ListFromXML: attiviamola usando il parametro 1; questo permette di generare le modelines dalla lista dei giochi (mame.xml) supportati dalla nostra versione di Mame/Groovymame;
3 - GenerateXML: attiviamola usando il parametro 1; questa genera il file mame.xml dal quale saranno estratte le modeline come detto sopra;
4 - GenerateInis: attiviamola usando il parametro 1; questo genera i file .ini per ogni gioco che verranno registrati nella cartella definita alla voce IniPath;
5 - IniPath: inseriamo il percorso della cartella ini che abbiamo creato in precedenza; questo servirà per posizionare gli i file .ini per ogni gioco (spiego dopo perchè);
6 - MonitorType: io ho lasciato su "CUSTOM" visto che non so per certo il modello del mio monitor, so solo che probabilmente è un Hantarex ma quello che è certo è che è a 15KHz.
7 - TotalModes: a seconda dei driver grafici usati verrà creato un numero diverso di modelines come descritto nello stesso file di testo; io ho messo 200 per averne il massimo;
8 - XresMin_XML, YresMin_XML, XresMin_Custom: ho messo come minima risoluzione 224 perchè ci sono dei giochi che usano questa risoluzione così bassa;
9 - UpdateRegistry: di default dovrebbe stare già a 1; questo permette alle modelines generate di essere inserite nel registro di XP per poter essere usate in seguito.

Una volta fatti questi aggiustamenti, salviamo il file, apriamo il WMMaker.exe; verranno così generate le modelines e gli ini per i singoli giochi. Una volta fatto, potremo verificare la creazione delle varie modelines andando a vedere il file Modeline.txt che sarà presente nella stessa cartella di WMMaker.exe.
Riavviamo il PC.
Apriamo Arcade_OSD: qua sono elencate le modelines appena create che si potranno modificare e testare. Su questo non dico molto di più, un po' perchè non ne so molto, un po' perchè di massima bastano le modelines che ho creato (almeno per le mie esigenze). Dirò solo che selezionando (con la tastiera) Video Modes, verranno elencate appunto le modelines create da WMMaker. Selezionando una modeline la potremo testare e se ci sono tremolii o malfunzionamenti vari, possiamo scegliere Edit Modeline. Poi possiamo sbloccare la frequenza (Lock Freq) e tramite le frecce provare a variarla avanti o indietro finchè il tremolio non si estingue.
Lo so, il procedimento è un po' maccheronico detto così ma non so dirvi altro; se avete maggiori esigenze dovrete documentarvi meglio. Una volta fatti i necessari aggiustamenti, ricordatevi di salvare i cambiamenti.
Se dovreste aver bisogno di altre modelines, queste possono essere create e aggiunte tramite programmi come WinModelines *(http://www.geocities.ws/podernixie/htpc/modeline-en.html).*

Interesssante è anche questo calcolatore online di modelines: *https://arachnoid.com/modelines/index.html*

Anche qui per maggiori info vi invito ad approfondire altrove questo argomento.

Aggiustare la risoluzione assegnata, file mame.ini e file .ini nei giochi

Abbiamo visto le istruzioni sulla configurazione del file mame.ini a seconda delle varie versioni di Mame. Ricordo che questo file mame.ini in realtà non è quello creato a partire da Mame ma quello creato da Groovymame col comando groovymame.exe –cc .
Una volta sistemato il file mame.ini, teoricamente Groovymame dovrebbe fare di tutto per selezionare la migliore risoluzione e frequenza, quella cioè più simile possibile a quella originale del gioco, in base alle modelines generate usando WMMaker. Eppure nel mio caso, pur facendo un ottimo lavoro sulla frequenza, Groovymame ha preso delle cantonate: per esempio nel gioco Ghosts 'n Goblins che prevede originariamente una risoluzione 256x224@60, Groovymame mi assegnava una 256x240@60. Nulla di grave, ma se la modeline 256x224@60 è regolarmente presente nel sistema e funziona perfettamente (verificata con ArcadeOSD), perchè non usarla?

Inoltre mi dava dei problemi di tearing *(https://en.wikipedia.org/wiki/Screen_tearing).*

Come fare dunque per affinare le risoluzioni e avere più controllo sui parametri di ogni singolo gioco?
Ho scelto di agire sui file .ini di ogni singolo gioco, quelli cioè che abbiamo creato prima con WMMaker e che abbiamo sistemato nella cartella ini. In effetti io ho proprio messo un collegamento a questa cartella sul desktop di XP per interventi veloci.

```
## gng ##
## "Ghosts'n Goblins (World? set
1)" ##
## gng.c ##
screen0 \\.\DISPLAY1
resolution0  256x224@60
hwstretch 0
syncrefresh 0
```

Riquadro 2 - Esempio di file .ini di un gioco

La logica è che Mame/Groovymame vanno prima a vedere il file mame.ini principale e poi il file ini di ogni gioco quindi "l'ultima parola" spetta al file ini di ogni gioco. Ecco di seguito un esempio:
Modificare la risoluzione è facilissimo: basta inserirla a mano e salvare il file. Stop.
E per il problema del tearing? Ho risolto mettendo il valore 1 alla voce Syncrefresh (da quello che ho capito, questa cosa dovrebbe variare da monitor a monitor quindi per voi potrebbe non valere). Se la voce Syncrefresh non fosse presente (o se il file ini c'è ma è completamente vuoto, può capitare anche questo), si può copiare il testo di un altro file ini e/o aggiungere a mano la voce Syncrefresh. Si possono ignorare le voci con il # davanti che fungono solo da commento.
Ok,ok, questa non sarà la soluzione più elegante ma funziona bene e mi dà un buon controllo su ogni gioco. Serve solo un periodo di aggiustamento alla prima apertura del vostro gioco preferito qualora ci dovessero essere problemi.
Ora viene la parte forse più rognosa: la scelta del front-end.

Configurare il front-end Mala

Con MameUI32 0.140 usavo Mala Front-end 1.74. Per me è veramente il massimo visto che permette di operare tutta la configurazione tramite interfaccia grafica senza il fastidio di dover configurare a mano decine di file di testo .cfg o .ini; ha inoltre un jukebox MP3 integrato. Comunque, per chi preferisce, si può sempre modificare a mano tramite il semplice Notepad, i file di layout .mll che Mala produce.
Purtroppo sembra che il suo sviluppo si sia fermato qualche anno fa e, di base, non riconosce le ultime versioni di Mame; soprattutto ha dei problemi con il file mame.xml e quindi non riesce a creare l'elenco dei giochi da visualizzare.
Ma la soluzione c'è, almeno fino al Mame/Groovvymame 0.170 visto che lo 0.192 (e successivi) li ho esclusi a priori per via, come detto sopra, del mancato supporto al DirectDraw.
Il Mame/Groovymame 0.170 viene riconosciuto da Mala (sia dalla versione 1.74 che 1.82) ma c'è da risolvere il problema della compatibilità col file mame.xml che è stato prodotto da WMMaker quando abbiamo creato le modelines. Il file mame.xml può anche essere generato senza usare WMMaker aprendo il prompt dei comandi (cmd.exe) di Windows e usando il comando mame.exe -listxml > mame.xml .
Quindi o creiamo il file xml così o lo copiamo/spostiamo dalla cartella di WMMaker a quella di Mame/Groovymame. Ma ora bisogna fare una piccola modifica per renderlo compatibile con Mala: apriamo il file mame.xml col Notepad (abbiate pazienza, ci mette un bel po' ad aprirsi, se tutto è andato bene sarà un file di testo di circa 184 MB!) e tramite l'opzione Sostituisci sostituiamo la parola machine con game. Salviamo il tutto.
Ora Mala sarà perfettamente compatibile anche con la 0.170 (e credo anche con le più recenti versioni, vi invito a verificare!).
Estremamente facile poi creare una playlist di musiche di sottofondo da ascoltare durante la scelta del gioco nonchè dei vari effetti audio, come pure definire i tasti per velocizzare lo scrolling dei giochi (saltare di 10 in 10, saltare direttamente alla lettera successiva/precedente e così via).
Semplicissimo inoltre farsi il proprio layout tramite la facile utility MalaLayout, convertire un layout con MalaLayoutCoverter (non so però quali formati accetta) oppure personalizzare la propria lista di giochi principale e di preferiti con MalaGameList.
Un vero peccato che questo prodotto non sia più sviluppato, ma per ora funziona ancora alla grande a parte qualche lieve bug che comunque non ne pregiudicano l'utilizzo.
Non mi soffermerò sulla configurazione di Mala visto che risulta molto intuitiva. Vi ricorderò solo di:
1) Se ci chiede di creare un nuovo file mame.ini o mame.xml, diciamogli di no visto che ce l'abbiamo già e che tra l'altro sono personalizzati/modificati;
2) Groovymame comprende al suo interno l'utility SwitchRes che serve per assegnare meglio di Mame la risoluzione e la frequenza più simile all'originale; tuttavia il lato negativo è che questo farà aprire una finestra in stile DOS che si sovrapporrà al front-end: per risolvere, basta andare su Mala all'opzione DOS Windows State nella label MAME Config/Basic e scegliere l'opzione Hidden.
Anche se sul sito è scaricabile la versione 1.82 (sempre beta però), sono rimasto alla 1.74 (beta anch'essa) perchè la 1.82 è incompatibile col programma DW Jukebox. In effetti è quest'ultimo a essere veramente preistorico però è tanto carino e ormai ce l'ho configurato a puntino. Alla fine Mala 1.82 non ha cose particolari in più rispetto alla 1.74.

Fig.8 – Esempio di configurazione dei pulsanti dei cabinato tramite software della JammASD.

Altro Front-end?

Premetto: non voglio certo far scatenare una faida tra i sostenitori dei vari front-end ma solo esprimere una mia impressione di utilizzo. Quando ho visto che Mala dava dei problemi con le ultime versioni di Mame (prima di scoprire il modo per modificare il file mame.xml), ho provato il front-end AttractMode che vede tanti estimatori.

Che dirvi, non è affatto male ma non ha la facilità di configurazione che ha Mala: il più delle volte si deve agire direttamente sui file di testo .cfg o .ini il che, a mio avviso, lo rende un po' ostico e scomodo da configurare. Figuratevi che per dirgli di mostrare solo i giochi "working" (ossia quelli che non hanno problemi con Mame), si devono inserire dei comandi, atti a definire un filtro, che non avrei davvero saputo dove andare a prendere. Per fare un paragone, in Mala basta cliccare su una opzione dell'interfaccia per ottenere lo stesso risultato....

Creare layout in AttractMode inoltre è sicuramente una cosa meno intuitiva in quanto bisognerebbe conoscere il linguaggio di scripting Squirrel che genera i file .nut. Non dico che è una cosa impossibile, ma dovermi andare a studiare un linguaggio per poter organizzare un layout di un cabianto arcade, mi sembrava una follia. Inoltre sarà sempre più lento che utilizzare una utility basata su interfaccia utente.

Una soluzione in quest'ultimo caso però, potrebbe essere quella di creare il layout con MalaLayout e usare il file .mll generato da Mala con AttractMode, visto la compatibilità di quest'ultimo con il primo. Questo in effetti velocizza le cose.

Inoltre, come già detto, Mala ha un jukebox MP3 integrato che può essere davvero utile (anche se io ne ho usato un altro, il DW Jukebox).

Un'altra motivazione che mi ha fatto rimane su Mala 1.74, è anche il fatto che il programma per jukebox che ho usato io:
(http://dwjukebox.com/)

dava dei problemi di apertura con AttractMode.

Altri utenti invece si trovano bene con FEEL....

Insomma, provateli tutti e poi vedete voi quale preferite.

Configurare i tasti del cabinato

Non rimane ora che configurare i tasti del cabinato: il Player 1, il Player 2, la gettoniera (fedelissimo alle mitiche 200 Lire!!!): per farlo basta installare il semplicissimo software della JammASD. Niente da dire qui, è davvero tutto intuitivo e spiegato alla pagina web relativa. Nella Fig.8 trovate un esempio di come ho configurato i tasti del mio cabinato.

Assegnazione di azioni ai pulsanti tramite front-end Mala

Una volta configurati i tasti del cabinato tramite il software della JammaASD, è possibile assegnare diverse azioni a essi, singolarmente o combinandoli tra loro, tramite Mala. Ad esempio, potremmo dire al front-end che se usiamo la combinazione joystick giù+player 1, saltiamo alla lettera alfabetica successiva nell'elenco dei giochi. Oppure se premo il pulsante del cabinato associato alla lettera A, scorreremo la lista giochi di 10 in 10 verso l'alto:

Per saltare di lettera in lettera: Player1+ joy su/joy giù

Per saltare di 10 giochi in 10 giochi nella lista: A/Q

Selezionare un gioco: Player 1

Conclusioni

Non mi rimaneva che fare le

Fig.9 - DW Jukebox

Fig.10 – Marquee.

seguenti operazioni:

1) regolare i potenziometri del CRT per la geometria dello schermo, ossia riuscire a trovare un buon compromesso tra area visiva dei giochi verticali (1943, Legendary Wings, Galaga...) e quelli orizzontali che sono la stragrande maggioranza;

2) regolare i potenziometri per la luminosità e il contrasto;

3) regolare il volume dell'audio ottenendo il giusto compromesso tra gli slider audio di WinXP e quelli di Mala;

4) fare una copia-immagine di sicurezza dell'hard disk su un altro hd.

Aggiunger un jukebox ed altri emulatori

Il front-end Mala permette di aggiungere anche altri eseguibili, tra cui io ho usato il software per jukebox DW Jukebox. E' davvero molto vecchio e semplice ma l'ho preferito a quello incluso all'interno dello stesso Mala perchè permette una facile configurazione, ha delle grafiche simpatiche e presenta un salva schermo retro davvero sfizioso.

Vi dico subito che su software più recenti, non funziona proprio come accennato sopra. L'accortezza che si deve poi avere sui file mp3, è quella di avere i tag ordinati e con i relativi nomi delle canzoni e degli autori dato che DW Jukebox va a leggere proprio questi metadati per ordinare in elenco le canzoni.

Si possono poi aggiungere altri emulatori. Io provai a inserire, con successo anche se è una procedura piuttosto complicata, WinUAE per i giochi AMIGA e VICE per il C64.

L' aspetto esterno

Dulcis in fundo, non si può non pensare all'aspetto esterno. La parte più divertente è forse il marquee, ossia l'"insegna" in alto, dove ho avuto l'accortezza di aggiungere anche un neon.

Di marquee già pronti ve ne sono a bizzeffe, sia generici, sia quelli che riproducono gli originali degli arcade di allora (come il mio relativo a Robotron 2084 che ho scelto per il suo delizioso font computerizzato anni '80).

La cosa migliore è quella di scaricarli in formato vettoriale (EPS o AI), formati che si possono aprire in alta qualità sia in Photoshop che in Illustrator. Una volta che abbiamo definito la grandezza, lo possiamo portare a una copisteria per farlo stampare sullo stesso materiale utilizzato per le vetrofanie, al contrario, più o meno trasparente e adesivo in modo da appiccicarlo al vetro dall'interno.

Se poi si aggiunge un bel neon o una striscia led a luce calda, l'effetto è asicurato. Per aumentare il tono caldo, ho avvolto il neon dentro delle plastiche di colore giallognolo per ricreare l'effetto di luce a incandescenza tipico degli anni '80.

Avendo molti soldi da spendere, si potrebbe anche far serigrafare il vetro, proprio come si faceva con i marquee originali. C'è da dire tuttavia che la serigrafia è un processo molto costoso se fatto su un singolo pezzo; la vetrofania è un metodo molto più a buon mercato e comunque molto gradevole.

Per le grafiche laterali invece, potete cercare le voci "side art arcade" o "artwork arcade" su Google e anche qui se ne trovano a decine. La cosa più ostica in questo caso però, è far coincidere le dimensioni del cabinato a quelle della grafica. In tal caso io ho proceduto facendo una foto della sagoma del cabinato sovrapponendola tramite layer di Photoshop a un side art che avevo scaricato. Ritoccando e ridimensionando la grafica del side art, ho fatto combaciare i due layer (quello della foto e quello del side art) tirandone fuori una grafica che andava bene per il mio cabinato.

Fig.11 – Grafiche laterali.

Ho consegnato il file in copisteria la quale ha proceduto stampando il tutto su vinile (lucido o opaco) adesivo (mi pare sia costato intorno ai 30 euro a lato, stampato ovviamente con plotter a getto di inchiostro).

Successivamente, ho adagiato il cabinato di lato per terra, ho sovrapposto la stampa e con molta cura ho usato un taglierino per tagliare via l'eccesso di foglio (sì, perchè ovviamente il plotter produce una stampa rettangolare). Nelle copisterie più avanzate può essere anche scelto un taglio controllato da computer per un risultato certamente migliore e meno faticoso anche se credo che il file deve essere opportunamente preparato.

Una volta ritagliata la nostra grafica, la possiamo incollare dandoci però l'opportunità di posizionarla con precisione. Per farlo, si usa lo stesso trucchetto che si adopera per le vetrofanie: acqua e sapone spruzzata con uno spruzzino a mano direttamente sulla superficie dove incolleremo la grafica. Questo creerà un velo tra lo strato adesivo e la superficie il quale, prima dell'asciugatura, ci permetterà di spostare il foglio per correggere eventuali errori.
Srotoliamo il vinile adesivo stendendolo aiutandoci con una spatole o, come ho fatto io, con delle presine da cucina (!), avendo cura di eliminare eventuali bolle d'aria spingendole verso l'esterno. Se proprio non riusciamo a eliminarle, possiamo procederle a bucarle con uno spillo molto sottile per far uscire l'aria: sarà meglio un piccolo buco che una antiestetica bolla d'aria sottostante.
Una volta che il vinile è sistemato, la procedura è finita.

Considerazioni finali

Un lavorone? Eh si, un po'. Anche perchè vi assicuro che la messa a punto di tutto (hardware, software e grafiche esterne) richiede davvero molto tempo, senza contare eventuali interventi sul legno, cromature o gettoniera (che malgrado pulizia e fiumi di lubrificante, tende molto spesso a incepparsi). Però fa parte del gioco e vi assicuro che alla fine otterrete un invidiabile oggetto di arredo che può, tra l'altro, sostituire un anonimo stereo da soggiorno.
Buon divertimento!

Pooyan BASIC Contest (Aprile-Maggio 2019)

Gruppo facebook: **RetroProgramming Italia 8bit e oltre**

di Davide Fichera

Tra i mesi di Aprile e Maggio 2019 si è svolto all'interno del gruppo facebook *RetroProgramming Italia 8bit e oltre* (prima conosciuto come *8bit RetroProgramming Italia*) un contest di programmazione in linguaggio BASIC finalizzato allo sviluppo di un gioco, che ricalcasse le dinamiche del famoso gioco degli anni '80: **Pooyan**.

Essendo il contest aperto a tutti i sistemi a 8 bit usciti tra la fine degli anni '70 e per tutti gli anni '80, le versioni di BASIC utilizzabili per partecipare al contest erano davvero una moltitudine.

Per rendere il contest più interessante, e finalizzato a un qualche obiettivo, si sono voluti imporre dei limiti di spazio per ognuna delle 2 categorie proposte:

categoria 4K: in questa categoria potevano partecipare giochi il cui codice occupava uno spazio disco minore o uguale a 4Kbytes ed erano ammessi tutti i BASIC con rispettivi dialetti;
categoria 20-lines: in questa categoria partecipavano solo giochi il cui codice era scritto entro le 20 linee di codice e ogni linea era formata da max 80 caratteri (ammettendo le abbreviazioni). Solo i BASIC presenti in ROM nelle macchine originali o comunque quelli "ufficiali" erano ammessi in questa categoria.
Categoria FC: quest'ultima categoria include tutti quei giochi che non rientrano nelle due categorie precedenti e sono sempre giochi in BASIC ma senza limiti di spazio o dialetti BASIC utilizzati. I giochi presenti in questa categoria non sono giudicati.

Gli iscritti al gruppo hanno accolto con molto entusiasmo la proposta e alcuni di loro si sono dedicati immediatamente allo sviluppo del gioco. Le macchine scelte sono state diverse: Commodore 64, Commodore 16, Texas Instruments TI-99/4A e Olivetti Prodest PC-128, con un preferenza però verso le macchine della Commodore.

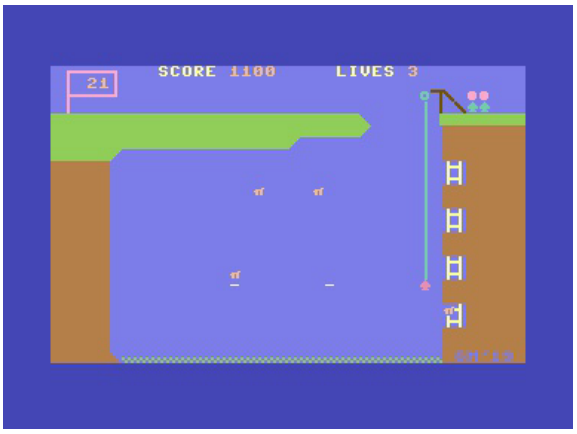
Una squadra di giudici all'interno del gruppo Staff di RetroProgramming Italia ha valutato tutti i giochi utilizzando come parametri di valutazione alcune caratteristiche quali:
- Aderenza al gameplay originale
- Qualità del codice
- Fluidità dell'azione
- Resa grafica
- Resa sonora
- Reattività dei controlli
- Documentazione

Abbiamo dato particolare rilievo alla documentazione del gioco, data la natura del gruppo, che è fondato principalmente sulla programmazione.
Nel periodo in cui il contest era attivo molti post riguardanti lo sviluppo dei giochi sono stati aperti sul gruppo, in cui si discutevano le diverse tecniche utilizzate per rendere al meglio uno o più aspetti del gioco e principalmente per ottimizzarne la fluidità dell'azione, data la lentezza instrinseca del BASIC. Tanta collaborazione quindi, diretta o indiretta, ha reso la competizione più avvincente e attiva.

Titoli che hanno partecipato al contest e relative schede:

Categoria 4K:

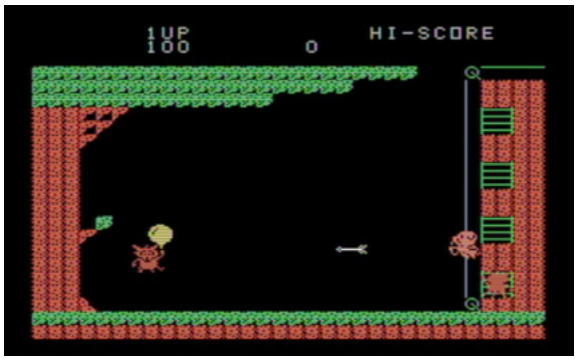
1) **Pooyan16-4**K (Giuseppe Mignogna, Commodore 16 – BASIC v3.5)
Link: https://drive.google.com/open?id=1u1RPmTk5CyWBQl0dr536KTopA65G-5qm



2) **Pooy4k** (Francesco Clementoni aka Arturo Dente, Commodore 64 – BASIC v2.0)
Link: https://drive.google.com/open?id=1jrSEbHXQamYVDvkcDZXz4oWp01wUZORb

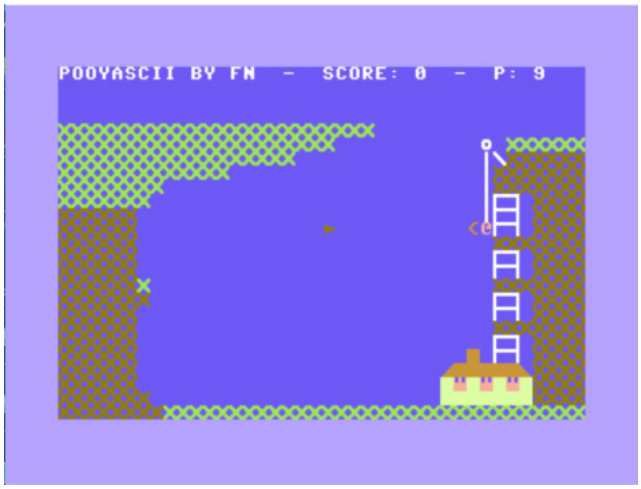
3) **Pooyan-4kb** (Felice Nardella, Commodore 64 – BASIC v2.0)
Link: https://drive.google.com/open?id=1KcWXHYPqZ4eiZVJ0_HXNrp-ksD_v7buL

4) **Ti YAN** (Francesco Ugga, Texas Instruments TI99/4A – TI BASIC)
Link: https://drive.google.com/open?id=1ViN2xb9W4edL5z9kiChM9NpCc_VFs9Bj
(questo è un po' difficile da fare funzionare perché bisogna importare il testo direttamente sull'emulatore)

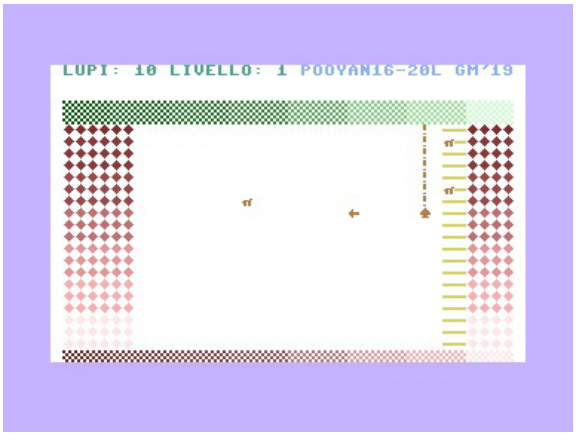


CATEGORIA 20-LINER:

1) **Pooyascii** (Felice Nardella, Commodore 64 – BASIC v2.0)
Link: https://drive.google.com/open?id=1nX81VCpUXUiUY_6tJXmRJ_So-dfuTg8H



2) **Pooyan16-20L** (Giuseppe Mignogna, Commodore 16 – BASIC v3.5)
Link: https://drive.google.com/open?id=1jRv4_olOLdBDI9c-9oUqP3TsgI8NEN3h



CATEGORIA FC (Fuori Contest)

1) **Mooyan** (Giuseppe Stassi – PC128/MO6)
Link: https://drive.google.com/open?id=16vQSvD0PmfbTjzreAQADJnrUKfLS_NQ9
(anche qui è un po' difficile usarlo perché bisogna incollare il testo direttamente sull'emulatore)

Tutto il materiale inerente il contest può essere facilmente trovato all'indirizzo:
https://drive.google.com/drive/u/0/folders/1sohhnCgLnwEedWYRIvJ8nTYEesqbRILg
Le cartelle includono per ogni concorrente tutti i docs, le immagini e i sorgenti dei loro giochi.

PREMIAZIONE

Per ogni contest bisogna proclamare dei vincitori, quindi sebbene tutti i lavori pervenuti siano meritevoli di lode, si deve procedere alla consegna virtuale delle coppe ai vincitori nelle due categorie in gara, per la categoria 4K:
Al **primo posto ex aequo Arturo Dente & Felice Nardella** al **terzo posto Francesco Gekido Ugga**

Mentre per il contest 20 Liner:
Giuseppe Mignona al primo posto
Felice Nardella al secondo posto.

POOY4K: ovvero Pooyan 4k

Cronaca della stesura del programma in basic e commento

di Arturo Dente

Il programma qui presentato rientra , per il contest omonimo , nella categoria 4k, con macchina target Commodore 64 e linguaggio Basic V2. Ide utilizzato: CBM Prg Studio.

In assenza di costrutti moderni, come chiamate a funzioni e multithread, e dati i limiti imposti dal contest (linguaggio e dimensioni) il gioco non rispecchia fedelmente l'originale, tuttavia si è cercato di mantenere quanto più gli elementi salienti e si è personalizzato qualcos'altro con un po' di creatività.
Il codice è strutturato essenzialmente in un ciclo principale che esegue gli opportuni gosub alle varie funzionalità (animazioni, collision detection, ecc), e a un insieme di variabili che – per lo più – servono a gestire gli stati in cui il gioco si trova di volta in volta.

```
4000rem main cycle
4010 gosub 6090:rem check joystick
4040gosub4380:rem sparo
4050gosub5030:rem entrata lupi
4060gosub 5220:rem inizio attacco
4065 gosub6030:rem padellate
4068gosub6280:rem fireball
4070 if m8 then goto 5510:rem gameover
4290goto4010
```

Ogni chiamata a una routine, infatti, non esaurisce il compito della stessa ma la porta avanti di uno step, in modo tale che al suo return può essere mandata avanti la successiva, e così via.

Il codice finale, poi, è stato ottimizzato usando le abbreviazioni fornite dal basic, per le quali il suddetto IDE fornisce lo strumento adatto per la conversione in massa dei comandi. I Rem sono stati poi eliminati tramite espressioni regolari su Notepad++ (rem.*$ per le righe che iniziano con rem, e :rem.*$ per i rem a fine riga) . Lo spazio guadagnato è stato così riallocato, diminuendo il numero di righe, e ottenendo infine un programma con dimensioni entro i limiti richiesti.

E' infine stato possibile inserire una sorta di colonna sonora all'inizio usando pochissimo spazio, come verrà illustrato nel commento del codice.

Il listato non ottimizzato

Di seguito, il listato del gioco non ottimizzato, verrà analizzato nei paragrafi successivi.

```
1gosub6230
10print"{clear}":lvl=3:v=53248:pa$="{black}F-{light
green}"
20dimwf$(3):rem vettori variabili interi1 e stringhe lupi
40i2=80:rem posizione iniziale y suino
50i3=8:rem delta y suino
60yf=(i2/8)-5:rem posizione iniziale freccia
62m4=0:rem paracadutisti scesi
70wf$(1)="{brown}  r{light green}":wf$(2)="{brown}
U8K{light green}":wf$(3)="{brown} K8U{light green}"
80m1=20:m2=190:m3=50:rem m1=x char lupo,m2=x sprite
lupo,m3=y sprite lupo
100 rem pokes per i colori in multicolor
110poke53281,6:poke53282,9:poke53283,9:poke53270,peek(5327
0)or16
140rem sprites
150pokev+21,255:pokev+37,10:pokev+38,1:pokev+39,0:pokev+28
,3
185poke2040,13:poke2041,14:poke2042,15:poke2043,11:rem
banks
190sz=832:rem Sprite block 13
200forx=0to62:ready:pokesz+x,y:nextx
205sz=896:forx=0to62:ready:pokesz+x,y:nextx
208forx=0to62:poke960+x,0:next:rem sprite freccia senza
read
209poke1000,255:pokev+41,0:pokev+4,0:pokev+5,0:rem sprite
freccia
210pokev,255:pokev+1,i2:rem posizione iniziale suino
220pokev+40,2:pokev+2,m2:pokev+3,0:rem colore e posizione
iniziale lupo
230forx=0to22:ready:poke728+x,y:next:pokev+6,0:pokev+7,0:p
okev+42,13:rem palla
240 hs=10*peek(727)
2100rem SCREEN 1 -
2120?spc(26)"     {reverse on}{light green}{169}{160}{127}
{reverse off}      ";
2130?spc(26)"    {reverse on}{169}{reverse off}{169} {127}
{reverse on}{127}{reverse off}     ";
2140?"{reverse on}{169}{reverse off}"+v$+"vvv      i
{127}{reverse on}{127}{reverse off}    ";
2150?v$+"{190}         i   hhhhhh";
2160?"vvvvvvvvvvvvvvvvv{190}"spc(15)"hhhhhhh";
2170?"v{reverse on}h{reverse off}vvvvvvv{190}{183}{183}
{183}{183}{183}{183}"spc(17)"hhhhhhh";
2180?" {reverse on}hh{reverse off}vvvvv{190}"spc(24)"{167}
{164}{164}{165}hhh";
2190?" {reverse on}hhh{reverse off}dv{reverse on}e{reverse
off}{190}"spc(25)"{167}{164}{164}{165}hhh";
2200?" vv{reverse on}h{reverse off}vh{reverse on}e{reverse
off}"spc(26)"{167}  {165}hhh";
2205fOt=1to3
2210?" vd{reverse on}h{reverse off}vhv"spc(26)"hhhhhhh";
2220fOz=1to2:?" vd{reverse on}h{reverse off}
vhv"spc(26)"{167}{164}{164}{165}hhh";:nE
```

```
2240?" Y{reverse on}hh{reverse off}vvv"spc(26)"{167}  {165}
hhh";:nE
2330?" vv{reverse on}h{reverse off}v6v"spc(26)"hhhhhhh";
2340?" vvvv6v"spc(26)"****hhh";
2350?" vvvvvvvVVVVVVVVV"+v$+"hhh{home}";
2355gosub 5400
2360goto4000
2500rem sprite suino
2510data0,0,0,0,60,0
2520data0,195,0,3,0,192
2530data3,20,192,12,85,192
2540data12,117,192,13,85,192
2550data13,85,192,12,85,192
2560data12,85,192,8,20,192
2570data44,101,192,170,149,192
2580data44,85,192,8,85,192
2590data15,215,192,3,255,0
2600data0,252,0,0,0,0
2610data0,0,0
2620 rem sprite lupo
2720 DATA 0,20,0,0,85,0
2730 DATA 1,215,64,7,125,208
2740 DATA 31,125,244,29,255,116
2750 DATA 125,85,125,85,0,85
2760 DATA 74,130,161,66,170,129
2770 DATA 18,170,132,18,235,132
2780 DATA 4,170,16,4,40,16
2790 DATA 2,150,128,10,150,160
2800 DATA 42,150,168,40,150,40
2810 DATA 0,150,0,0,170,0
2820 DATA 2,130,128
2825 rem palla
2830 DATA
0,255,0,3,0,192,4,255,32,8,0,16,8,255,16,4,0,32,3,0,192,0,
255,0
4000rem main cycle
4010 gosub 6090:rem check joystick
4040gosub4380:rem sparo
4050gosub5030:rem entrata lupi
4060gosub 5220:rem inizio attacco
4065 gosub6030:rem padellate
4068gosub6280:rem fireball
4070 if m8 then goto 5510:rem gameover
4290goto4010
4300rem suino up
4310ifi1=126thengoto4316
4314i4=-1:ifi7=0theni5=29:rem if here, fired
4316ifi7=0thenyf=(i2/8)-5
4319ifi2=80thenreturn
4322at$=" ":yy=(i2/8)-5:xx=30:gosub6000
4325i2=i2-i3:pokev+1,i2:rem muovi suino
4330return
4350rem suino down
4354ifi1=125thengoto4359
4356i4=-1:ifi7=0theni5=29:rem if here, fired
4359ifi7=0thenyf=(i2/8)-5
4360ifi2=216thenreturn
4362at$="{light green}i":yy=(i2/8)-6:xx=30:gosub6000
4365i2=i2+i3:pokev+1,i2:rem muovi suino
4370return
4380rem freccia
4390ifi4=0andi7=0thenreturn
4395if noti7 then pokes+1,16:pokes,195:pokes+4,129:rem
beep
4400i6=yf:i5=i5+15:i7=-1
4402 gosub 5270
4405ifi5>200 or m5theni5=260:i7=0:if
m5thenpokev+2,peek(v+2)-8:poke53277,2
4412 pokev+4,260-i5:pokev+5,(i6+5)*8
4415i4=0:if i7thenpokes+4,16:rem beep off
4420return
5030rem entrata lupi
5035ifi8<>-1thenreturn
5036b=notb
5037 fort=0to4
5038ifi9-(15-5*t)<0thengoto5060
5040 ?"{home}"spc(i9-(15-5*t))wf$(1)
5050?"{home}{down}"spc(i9-(15-5*t))wf$(2-b)
5060nextt
5070 i9=i9+1:ifi9>14theni8=1:i9=0
5210return
5220 rem inizio attacco
5225 if i8<>1 thenreturn
5228 if m3>50 then goto5245
5230 at$="   ":yy=0:xx=m1:gosub6000
5240 at$="   ":yy=1:xx=m1:gosub6000
5245 pokev+2,m2+3*sin(m3):pokev+3,m3
5246 poke53277,0
5247 gosub4380:rem sparo
5248 m3=m3+int(rnd(1)*3)+lvl:gosub6090:rem oscillazione e
vel.variabili+joystick
5249 if m3<210 then if notm5then goto 5254
5250 if m3>=210 then m7=m7+1:gosub5430
5251 m3=50:m1=m1-5:m2=m2-38:m4=m4+1:pokev+3,0:rem wolf
gone
5254 m5=0:ifm4=4 then gosub5450
5260 return
5270 rem collision detection
5280 fy=20+(i6+5)*8:fx=280-i5:m5=0
5285 if fx>m2theniffx<m2+24theniffy>m3theniffy<m3+28then
m5=-1:m6=m6+10
5288 if not m5 then return
5290 pokes+1,16:pokes,195:pokes+4,33:rem beep
5291 gosub 5400:pokes+4,16:rem update score and stop
beeping
5292 return
5400 rem update score
5410 at$="{white}hiscore:"+stR(hs)+" lvl:"+stR(lvl-2)+"
score: "+stR(m6)+"{light green}"
5415 yy=24:xx=6:gosub6000:reT
5420 rem lasciato x sicurezza return
5430 rem update the right wolf
5432 at$=wf$(1)+"{light green}":xx=32:yy=23-4*m7:gosub6000
5433 at$=wf$(2-b)+"{light green}":xx=32:yy=24-
4*m7:gosub6000
5434 fort=32to16step-1:pokes+1,t:pokes,
179+t:pokes+4,33:next:rem beep
5436 if m7=4 then m8=-1
5438 pokes+4,16
5440 return
5450 rem lvl up
5455 gosub5480:gosub5400
5470 return
5480 rem azzeramento parziale
5490 i8=0:i9=0
5500 lvl=lvl+1:m1=20:m2=190:m3=50:m5=0:m4=0:return
5510 rem end
5512 poke53271,1:fort=i2to225:pokev+1,t:nextt:poke53271,0
5515 pokes+1,16:pokes,195:pokes+4,33:rem beep
5520 ?"{clear}":pokev+21,0:?"{white}game
over":?:?"punteggio: "+str$(m6)
5525 if(m6>hs)tH?"{return}**nuovo
hiscore**{return}":poke727,m6/10
5530 ?"ancora? (s/n) ":pokes+4,16
5540getz$:if(z$)=""thengoto5540
5550 ifz$="n" then end
5560 run
6000 rem printat - at$,xx,yy
6010 poke780,0:poke781,yy:poke782,xx:sys65520:printat$;
6020 return
6030 rem padelle variabili
6040 if lvl<5 or m7=0thenreturn
6060 at$="  ":if int(rnd(1)*5)=3thenat$=pa$
6070 xx=30:yy=24-4*m7:gosub6000
6075 if at$="  " then return
6078 ifi2>=232-32*m7thenm8=-1
6080 return
6090rem check joystick
6095i1=peek(56320)
6100ifi7=0thenyf=(i2/8)-5
6110ifi1=126ori1=110thengosub4300
6120ifi1=125ori1=109thengosub4350
```

```
6130ifi1=111theni4=-1:
6135ifi7=0theni5=29
6140ifi8=0theni8=int(rnd(1)*2)-1
6150 return
6160 rem  music
6180 i=val(mid$(mu$,no,1))
6190
pokes+1,hi(i):pokes,lo(i):pokes+4,17:fort=1to100:next:
pokes+4,16:fort=1to50:next
6220 no=no+1:ifno>len(mu$)thenno=1
6222 return
6230 rem home
6231 s=54272:dim hi(4):dim lo(4):pokes,
15:mu$="1323132314342414243":no=1
6232
hi(1)=34:hi(2)=43:hi(3)=51:hi(4)=61:lo(1)=75:lo(2)=52:lo(3
)=97:lo(4)=126
6234 pokes+24,15:pokes+5,100:pokes+6,215:rem
volume,attack,sustain
6235 fort=stos+24:pokes,0:next:rem sid init
6238 fort=1to20:v$=v$+"V":next:rem variabile mattoni
ripetuti
6240 ?"{clear}":at$="{white}press a key"
6250 xx=15:yy=12:gosub6000:bb=rnd(-ti):xx=1:yy=1
6255 at$=" {red}p{green}o{white}o{yellow}y{purple}4{green}
k ":gosub6000:xx=xx+1:ifxx>100thengoto6240
6260 gets$:ifs$=""thengosub6180:goto6255
6270 return
6280 rem fireball
6290 ifm9=-1thengoto6400
6300 ifint(rnd(1)*8)=3thenm9=-1:n1=m3:n2=m2:return
6400
n2=n2+10:n1=n1+0.5:ifn2>=255orn1>=255thenpokev+7,0:pokev+6
,0:m9=0:return
6410 pokev+6,n2:pokev+7,n1:ifn1>i2-
12thenifn1<i2+12thenifn2>=242thenm8=-1
6420 return
```

Spiegazione e commento delle sezioni di codice

La schermata iniziale: inizializzazioni e musica

La prima riga (1gosub6230) rimanda alla 6230 dove viene impostata la variabile s come base per le locazioni di memoria relative al SID, il quale viene qui anche resettato e impostato a valori di volume, attack e sustain standard. Vengono anche definiti un array di 4 locazioni per l'alta frequenza e uno di altrettante locazioni per la bassa , in vista del successivo step di generazione di note, per il quale servirà anche la variabile mu$="4342414243".

La variabile V$ servirà in seguito per ridurre lo spazio necessario a stampare sequenze di "V" che compongono – in multicolor – buona parte dello sfondo di gioco.

La scritta scrollante POY4K verrà visualizzata attraverso la chiamata alla routine che si occupa di fare il "print at". Tale routine, che verrà utilizzata spesso, si baserà sempre sui valori di xx, yy e at$, rispettivamente posizione x, posizione y e variabile stringa da stampare. Non essendoci funzioni – come detto in prefazione – è comunque possibile simularle attraverso gosub e attraverso l'assegnazione di variabili dedicate solo a tale scopo, un gruppo di esse per ogni "funzione".

La musichetta nella schermata iniziale

La riga 6260, che attende la pressione di un tasto per iniziare il gioco, richiama ciclicamente la 6180 per l'esecuzione dell'ennesima nota del motivetto iniziale.

Le note utilizzate sono le 4 che compongono l'accordo di sol 7 (do, mi, sol, sib) e sono memorizzate, per l'alta e la bassa frequenza, nei rispettivi array hi() e lo(). La variabile mu$="1323132314342414243" viene così scandita ad ogni passaggio per recuperare l'indice degli array corrente, corrispondente alla nota da suonare, tramite i=val(mid$(mu$,no,1)).

Altre operazioni preliminari prima dell'inizio del gioco

Dopo la schermata di presentazione il codice ritorna alla riga 10, alla quale seguono una serie di inizializzazioni di variabili e di poke, il cui significato è ben comprensibile dai REM che corredano le varie istruzioni. Uniche segnalazioni degne di nota:

1) il disegno dello sfondo di gioco si avvale della variabile V$ già introdotta, nonché di un uso capillare della funzione SPC(n), che ha consentito di risparmiare parecchi bytes.

2) Lo sprite della freccia non è generato con dei read giacchè è quasi tutto 0, è sufficiente che tutti i bit della locazione 1000 sono impostati a 1 tramite il valore 255.

3) Infine, la variabile lvl, che sta per "livello", è inizializzata a 3 perché a valori più bassi fa partire il gioco troppo lentamente (si illustrerà in seguito in che modo condiziona la velocità e, quindi, la difficoltà), tuttavia per motivi estetici verrà mostrata al giocatore come lvl-2, in modo da partire da 1.

La riga di print che rende a video quest'ultima informazione, unitamente al punteggio, è contenuta nella sezione di codice che inizia a 5400 e che non fa altro che richiamare la già citata "print at" tramite gosub 6000, e viene eseguita una tantum prima del main cycle.

Inizio del gioco – analisi del ciclo principale

Alla riga 4000 inizia il ciclo principale, composto di 6 gosub e di un check sulla variabile m(8) per rilevare se si è in stato di gameover.

```
4000rem main cycle
4010 gosub 6090:rem check joystick
4040gosub4380:rem sparo
4050gosub5030:rem entrata lupi
4060gosub 5220:rem inizio attacco
4065 gosub6030:rem padellate
4068gosub6280:rem fireball
4070 if m8 then goto 5510:rem gameover
4290goto4010
```

Analizziamo i singoli gosub del main cycle

Check segnali del joystick (gosub 6090)

Il codice che viene eseguito in questa subroutine è il seguente:

```
6090rem check joystick
6095i1=peek(56320)
6100ifi7=0thenyf=(i2/8)-5
6110ifi1=126ori1=110thengosub4300
6120ifi1=125ori1=109thengosub4350
```

```
6130ifi1=111theni4=-1:
6135ifi7=0theni5=29
6140ifi8=0theni8=int(rnd(1)*2)-1
6150 return
```

Prima di commentarlo, occorre definire il senso delle variabili qui in gioco.
- La variabile i1 è utilizzata per leggere il valore della locazione 56320 (joystick porta 2).
- La variabile i2 memorizza la y dello sprite protagonista.
- La variabile i4 è pari a true (cioè -1) se si è premuto fire.
- La variabile i5 memorizza la x della freccia.
- La variabile i7 è pari a false (cioè 0) se la freccia non sta viaggiando.
- Infine, i8=-1 se è in atto l'animazione di entrata dei lupi, 0 se neutro, 1 se si entra in stato inizio attacco, 2 se è in uno stato idle

La riga 6100 aggiorna la posizione y della freccia solo se questa non sta viaggiando. Questo per evitare che, una volta sparata, continui a seguire i movimenti y dello sprite protagonista. Quando è sparata, infatti, la sua ordinata non deve più cambiare.
Le righe 6110 e 6120 richiamano le subroutine di spostamento dello sprite protagonista a seconda che si stia andando su o giù. Vengono rilevati i movimenti su e giù con e senza sparo, ecco perché entrambi le righe fanno il check sull'or di due valori.

Position	56321 Port 1	56320 Port 2	56321 Port 1 Fire pushed	56320 Port 2 Fire pushed
middle (no move)	255	127	239	111
up	254	126	238	110
down	253	125	237	109
left	251	123	235	107
right	247	119	231	103
up left	250	122	234	106
up right	246	118	230	102
down left	249	121	233	105
down right	245	117	229	101

La riga 6130 verifica se è stato premuto fire. In tal caso mette a true la variabile i4 (in modo che il programma sia informato di ciò anche in altri punti) , mantenendo così vivo l'evento anche al successivo variare del peek(56320)
La riga 6135 fa un check ridondante, nel senso che si ritroverà anche altrove, non è correlata allo scopo della routine, e verifica se la freccia – per qualche motivo- ha smesso di viaggiare, e in tal caso riporta la sua x al valore iniziale tramite i5=29.
La riga 6140 attraverso i8 verifica se ci si trova nello stato neutro, quello che precede l'uscita dei lupi. L'azione che segue è tale che se i8=0 (quindi stato neutro), o lo stato resta neutro oppure riparte l'animazione dei lupi (i8=-1). Poiché ciclicamente si tornerà in questa istruzione, prima o poi i nemici inizieranno la loro sequenza di entrata. Questo solo per dare un minimo di randomizzazione alla loro uscita che, comunque, avverrà quasi subito. Se si vuole un tempo maggiore, basta aumentare il ventaglio dei valori random restituiti.

Sparo della freccia (gosub 4380)

```
4380rem freccia
4390ifi4=0andi7=0thenreturn
4395if noti7 then pokes+1,16:pokes,195:pokes+4,129:rem
beep
4400i6=yf:i5=i5+15:i7=-1
4402 gosub 5270
4405ifi5>200 or m5theni5=260:i7=0:if
m5thenpokev+2,peek(v+2)-8:poke53277,2
4412 pokev+4,260-i5:pokev+5,(i6+5)*8
4415i4=0:if i7thenpokes+4,16:rem beep off
4420return
```

La prima cosa che viene verificata è se non sia in corso un lancio precedente, alla riga 4390.
Lo scoccare della freccia inizia quindi con il play del suono (riga 4395), l'aggiornamento della posizione y della stessa a partire da quella del suino, l'incremento della posizione x e infine lo switch di i7 a true (riga 4400), perché a questo punto la freccia sta viaggiando.
E infatti segue immediatamente il collision detection con il gosub 5270. La riga successiva controlla se la corsa del proiettile è terminata per vie "naturali" o perché è stato colpito un lupo (m5=true). In entrambi casi la freccia viene fatta sparire. Se il lupo è stato colpito (m5=true) allora lo allarga come a simularne l'esplosione, prima che venga fatto sparire.
La riga 4412 esegue i poke necessari per visualizzarla nella posizione corrente, la 4415 infine decreta che fire non è più in stato di "premuto" e termina il play del suono qualora esso sia stato iniziato in questo ciclo.
Riguardo al collision detection il codice è facilmente comprensibile. Ci limitiamo a dire che non si è approcciato il problema tramite opportuni peek perché era più difficile da gestire rispetto a un controllo su un intorno, scelta qui effettuata.

Movimento su e giù del suino protagonista

Come scritto più su, le righe 6110 e 6120 richiamano le subroutine di spostamento dello sprite protagonista (4300 e 4350)a seconda che si stia andando su o giù.

```
4300rem suino up
4310ifi1=126thengoto4316
4314i4=-1:ifi7=0theni5=29:rem if here, fired
4316ifi7=0thenyf=(i2/8)-5
4319ifi2=80thenreturn
4322at$=" ":yy=(i2/8)-5:xx=30:gosub6000
4325i2=i2-i3:pokev+1,i2:rem muovi suino
4330return
4350rem suino down
4354ifi1=125thengoto4359
4356i4=-1:ifi7=0theni5=29:rem if here, fired
4359ifi7=0thenyf=(i2/8)-5
4360ifi2=216thenreturn
4362at$="{light green}i":yy=(i2/8)-6:xx=30:gosub6000
4365i2=i2+i3:pokev+1,i2:rem muovi suino
```

4370return

Entrambi le subroutine di direzione verificano per prima cosa se si tratti di direzione + sparo o solo direzione. Il gioco, infatti, consente di sparare mentre ci si muove. In caso di direzione più sparo, viene comunque impostato lo stato "inizio sparo" (i4=-1), e successivamente viene verificato che la freccia non sia già precedentemente in viaggio (check su i7). Se la freccia deve partire, allora la sua x viene impostata a 29 e la sua y viene aggiornata in base alla posizione del suino. Infine, se il suino sta scendendo, viene disegnato un pezzo di corda sopra di lui tramite print at, se invece sta salendo viene cancellato l'ultimo pezzo allo stesso modo, per poi aggiornare la y dello sprite e disegnarlo.

Animazione di entrata dei lupi (gosub 5030)

```
5030rem entrata lupi
5035ifi8<>-1thenreturn
5036b=notb
5037 fort=0to4
5038ifi9-(15-5*t)<0thengoto5060
5040 ?"{home}"spc(i9-(15-5*t))wf$(1)
5050?"{home}{down}"spc(i9-(15-5*t))wf$(2-b)
5060nextt
5070 i9=i9+1:ifi9>14theni8=1:i9=0
5210return
```

Si tratta di una semplice animazione fatta coi caratteri in multicolor. Unica nota da segnalare: la parte inferiore dei lupi viene switchata ciclicamente tra due variabili stringa wf$(2) e wf$(3) tramite wf$(2-b) con b=notb ad ogni ciclo , in modo da far sembrare che stiano camminando.

Attacco paracadutisti (gosub 5220)

```
5220 rem inizio attacco
5225 if i8<>1 thenreturn
5228 if m3>50 then goto5245
5230 at$="   ":yy=0:xx=m1:gosub6000
5240 at$="   ":yy=1:xx=m1:gosub6000
5245 pokev+2,m2+3*sin(m3):pokev+3,m3
5246 poke53277,0
5247 gosub4380:rem sparo
5248 m3=m3+int(rnd(1)*3)+lvl:gosub6090:rem oscillazione e
vel.variabili+joystick
5249 if m3<210 then if notm5then goto 5254
5250 if m3>=210 then m7=m7+1:gosub5430
5251 m3=50:m1=m1-5:m2=m2-38:m4=m4+1:pokev+3,0:rem wolf
gone
5254 m5=0:ifm4=4 then gosub5450
5260 return
```

Dopo aver verificato, tramite i8, se ci troviamo nel posto giusto, controlla se l'ascissa del lupo paracadutista è all'inizio, e in caso sovrascrive degli spazi per cancellare il lupo non paracadutista.
Dopodiché si tratta di far scendere lo sprite secondo lo stato attuale delle variabili m2 e m3 che ne descrivono le coordinate. Per dare un effetto di ondulazione si è introdotto un delta x tramite la funzione seno, dandole come argomento la y, in modo tale che all'aumentare della discesa il suddetto delta si muove tra -1 e 1. Viene poi moltiplicato per 3 per rendere più apprezzabile lo scostamento.

```
pokev+2,m2+3*sin(m3):pokev+3,m3
```

All'interno della routine viene anche eseguito un ulteriore controllo dello stato-sparo rispetto a quello del ciclo principale (riga 5247 gosub 4380), onde evitare lag di rilevazione dello stesso, date le diverse righe che interessano la sezione attuale. Similmente la riga successiva ripete il polling del joystick, dopo aver incrementato la y del paracadutista (variabile m3) di un delta casuale e in funzione del livello di gioco:
m3=m3+int(rnd(1)*3)+lvl
Successivamente, la riga 5249 verifica se il paracadutista non abbia ancora raggiunto terra e non sia stato colpito. Se è così, salta le successive e va all'ultima prima del return, che esegue un controllo su quanti paracadutisti sono partiti. Nel caso siano 4, allora il livello viene incrementato.
Se invece ci troviamo alla riga 5250, allora il lupo ha raggiunto il terreno e, tramite gosub 5430, viene disegnato a destra dello schermo tramite gli stessi caratteri che lo definiscono nell'entrata in scena all'inizio di ogni livello. La riga successiva reinizializza alcune variabili e cancella lo sprite del paracadutista, mentre – infine – la 5254 controlla se i lupi scesi sono quattro, e in tal caso fa avanzare di livello tramite gosub 5450, che esegue a sua volta due gosub, uno per l'azzeramento parziale delle variabili, e uno per la stampa a video del livello aggiornato. Ovviamente, questa ridondanza è mantenuta qui solo per facilità di lettura, ma nel codice minimizzato è eliminata, portando i due gosub direttamente nella chiamata del primo.

"Padellate" dai lupi che sono giunti a destinazione

```
6030 rem padelle variabili
6040 if lvl<5 or m7=0thenreturn
6060 at$="  ":if int(rnd(1)*5)=3thenat$=pa$
6070 xx=30:yy=24-4*m7:gosub6000
6075 if at$="  " then return
6078 ifi2>=232-32*m7thenm8=-1
6080 return
```

Il tempo è tiranno, per cui i lupi si portano avanti con il lavoro già con la padella in mano. Se riescono ad assestare un colpo sul suino sono già pronti a cucinarlo.
Tutto ciò avviene da lvl>=5 (ovvero livello 3 per il giocatore), e non avviene se non esiste alcun lupo a destra, come suggerisce la riga 6040.
La "padella" non è altro che la stringa pa$, che viene visualizzata tramite "print at" in momenti random (il consueto gosub 6000).La posizione y della padella è funzione del numero di lupi presenti a destra (m7). Il "collision detection" è molto semplice e controlla - a "padellata in corso" - semplicemente se la y del suino è tale che la padella lo colpisca o che colpisca la corda sopra di lui.

Hadoken! (gosub 6280)

```
6280 rem fireball
6290 ifm9=-1thengoto6400
```

```
6300 ifint(rnd(1)*8)=3thenm9=-1:n1=m3:n2=m2:return
6400
n2=n2+10:n1=n1+0.5:ifn2>=255orn1>=255thenpokev+7,0:pokev+6
,0:m9=0:return
6410 pokev+6,n2:pokev+7,n1:ifn1>i2-
12thenifn1<i2+12thenifn2>=242thenm8=-1
6420 return
```

I paracadutisti, stufi di aspettare il proprio turno, hanno acquisito da Street Fighter la sfera energetica.
Dopo aver controllato se è già in corso un lancio di bolla energetica (e nel caso, return), il codice decide a random se effettuarne uno oppure no (riga 6300), inizializzando in quest'ultimo caso le variabili deputate a tale azione.
Il resto è normale amministrazione: incremento variabili x e y (quest'ultima ha un leggero degrado mano mano che procede, tramite n1=n1+0.5) , pokes per visualizzare lo sprite e semplice collision detection basato su intorno della posizione del suino.

Game over (goto 5510)
L'ultima riga del ciclo principale controlla lo stato della variabile m8, la quale –se è stata impostata a true da qualche altra sezione del codice, a seguito di eventi "mortiferi" – rimanda alla riga 5510, dove inizia la fine di tutto:

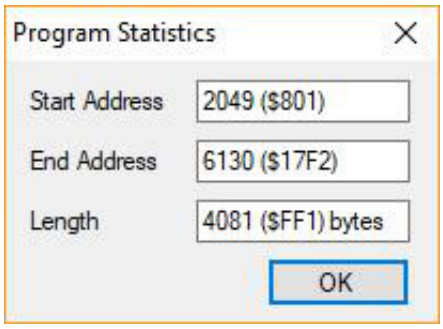
```
5510 rem end
5512 poke53271,1:fort=i2to225:pokev+1,t:nextt:poke53271,0
5515 pokes+1,16:pokes,195:pokes+4,33:rem beep
5520 ?"{clear}":pokev+21,0:?"{white}game
over":?:?"punteggio: "+str$(m6)
5525 if(m6>hs)tH?"{return}**nuovo
hiscore**{return}":poke727,m6/10
5530 ?"ancora? (s/n) ":pokes+4,16
5540getz$:if(z$)=""thengoto5540
5550 ifz$="n" then end
5560 run
```

La riga 5512 esegue un'animazione dove il suino protagonista cade dalla fune, e per effetto della gravità si "allunga" (poke53271,1, rimesso a 0 a fine animazione).
Viene poi eseguito un beep di riscontro sonoro, per poi pulire lo schermo (anche da sprites), e mostrare il punteggio (variabile m6).
Una nota importante riguarda la memorizzazione dell' Highest Score. Per risparmiare spazio prezioso, si è deciso di far ripartire il gioco eseguendo Run (che cancella lo stack). Così facendo, però, si azzerano tutte le variabili, compresa quella ipoteticamente dedicata al punteggio più alto. Per rimediare a ciò, si è deciso di registrare un decimo del punteggio nella locazione 727 (un decimo, perché arrivare a 255 punti non è difficile, 2550 forse è impossibile).
Tale locazione è un punto esterno ai bordi della fireball. Era possibile sicuramente trovare una locazione libera e non impegnata, tuttavia l'idea di avere una sorta di side effect sulla palla energetica, dipendentemente dal punteggio più alto precedente – mi piaceva.
E se il giocatore totalizza più di 2550 punti? Il programma va in errore, e il suino si desta e scopre di essere parte di un sistema informatico dove tutto avviene nella sua immaginazione tramite una rete che connette il suo cervello a un mondo virtuale, chiamato Suimatrix.

Appendice
Le variabili più importanti e il loro ruolo
Poiché il basic V2 del commodore 64 recepisce solo i primi due caratteri di una variabile come "chiavi" , onde evitare di incorrere in errori di omonimie indesiderate (ad esempio lupoapiedi$ e lupoparacadutista$ sono la stessa cosa) si è mantenuta fin da subito una lista delle variabili utilizzate, avendo scelto di usare delle serie alfanumeriche composte da lettera+numero[1-9].
Queste sono le loro definizioni:

```
rem i1=joystick
rem i2= y suino
rem i3=dy suino
rem i4=-1 se premuto fire,0 se no
rem i5=x freccia
rem i6=y freccia
rem i7=-1 se la freccia sta viaggiando,else 0
rem i8=-1 se in atto entrata in scena lupi,0 se neutro, 1
se entra in stato inizio attacco,2 stato idle
rem i9=margine sinistro entrata lupi
rem m1=x char lupo,m2=x sprite lupo,m3=y sprite lupo
rem m4 contatore paracadutisti scesi
rem m5 colpito lupo,m6 punteggio,m7 lupi arrivati a casa
rem m8=gameover se true
rem m9=-1 se iniziato lancio palla
rem n1=y palla, n2=x palla
```

Il codice minimizzato
Il codice minimizzato, ottenuto in parte tramite automatismi (funzione "compress" di CBM prg studio e eliminazione dei Rem tramite regex, come detto all'inizio) e in parte riorganizzando i "buchi" liberatisi per accorpare quanto più possibile pezzi di codice, è il seguente, e "pesa" 4081 bytes, come da immagine.




```
10goS6230:?"{clear}":lvl=3:v=53248:pa$="{black}F-{light
green}":dIwf$(3):i2=80
20i3=8:yf=(i2/8)-5:m4=0:wf$(1)="{brown}  r{light
green}":wf$(2)="{brown} U8K{light green}":wf$(3)="{brown}
K8U{light green}"
80m1=20:m2=190:m3=50:po53281,6:po53282,9:po53283,9:po53270
,pE(53270)or16
150pOv+21,255:pOv+37,10:pOv+38,1:pOv+39,0:pOv+28,3:pO2040,
13:pO2041,14:pO2042,15
185pO2043,11:sz=832:fOx=0to62:rEy:pOsz+x,y:nEx:sz=896:fOx=
0to62:rEy:pOsz+x,y:nEx
208fOx=0to62:pO960+x,
0:nE:pO1000,255:pOv+41,0:pOv+4,0:pOv+5,0:pOv,255:pOv+1,i2
220pOv+40,2:pOv+2,m2:pOv+3,0:fOx=0to22:rEy:pO728+x,y:nE:pO
v+6,0:pOv+7,0:pOv+42,13
```

```
2120?sP26)"     {reverse on}{light green}{169}{160}{127}
{reverse off}      ";
2130?sP26)"    {reverse on}{169}{reverse off}{169} {127}
{reverse on}{127}{reverse off}     ";
2140?"{reverse on}{169}{reverse off}"+v$+"VVV      i
{127}{reverse on}{127}{reverse off}    ";
2150?v$+"{190}         i   hhhhhh";
2160?"VVvvvvvvvvvvvvvvv{190}"sP15)"hhhhhhh";
2170?"v{reverse on}h{reverse off}vVvvvvV{190}{183}{183}
{183}{183}{183}{183}"sP17)"hhhhhhh";
2180?" {reverse on}hh{reverse off}vvvvV{190}"sP24)"{167}
{164}{164}{165}hhh";
2190?" {reverse on}hhh{reverse off}dv{reverse on}e{reverse
off}{190}"sP25)"{167}{164}{164}{165}hhh";
2200?" vv{reverse on}h{reverse off}vh{reverse on}e{reverse
off}"sP26)"{167}  {165}hhh";
2205fOt=1to3
2210?" vd{reverse on}h{reverse off}vhv"sP26)"hhhhhhh";
2220fOz=1to2:?" vd{reverse on}h{reverse off}vhv"sP26)"{167}
{164}{164}{165}hhh";:nE
2240?" Y{reverse on}hh{reverse off}vvv"sP26)"{167}  {165}
hhh";:nE
2330?" vv{reverse on}h{reverse off}v6v"sP26)"hhhhhhh";
2340?" vvvv6v"sP26)"****hhh";
2350?" vvvvvvVVVVVVVVV"+v$
+"hhh{home}";:hs=10*pE(727):goS5400
2510dA0,0,0,0,60,0,0,195,0,3,0,192,3,20,192,12,85,192,12,1
17,192,13,85,192
2550dA13,85,192,12,85,192,12,85,192,8,20,192,44,101,192,17
0,149,192
2580dA44,85,192,8,85,192,15,215,192,3,255,0,0,252,0,0,0,0,
0,0,0,0,20,0,0,85,0
2610dA1,215,64,7,125,208,31,125,244,29,255,116,125,85,125,
85,0,85
2760dA74,130,161,66,170,129,18,170,132,18,235,132,4,170,16
,4,40,16
2790dA2,150,128,10,150,160,42,150,168,40,150,40,0,150,0,0,
170,0
2820dA2,130,128,0,255,0,3,0,192,4,0,32,8,0,16,8,0,16,4,0,3
2,3,0,192,0,255,0
4010goS6090:goS4380:goS5030:goS5220:goS6030:goS6280:ifm8tH
gO5510
4290gO4010
4300ifi1=126tHgO4316
4314i4=-1:ifi7=0tHi5=29
4316ifi7=0tHyf=(i2/8)-5
4319ifi2=80tHreT
4322at$=" ":yy=(i2/8)-5:xx=30:goS6000:i2=i2-
i3:pOv+1,i2:reT
4350ifi1=125tHgO4359
4356i4=-1:ifi7=0tHi5=29
4359ifi7=0tHyf=(i2/8)-5
4360ifi2=216tHreT
4362at$="{light green}i":yy=(i2/8)-
6:xx=30:goS6000:i2=i2+i3:pOv+1,i2:reT
4380ifi4=0aNi7=0tHreT
4395ifnOi7tHpOs+1,16:pOs,195:pOs+4,129
4400i6=yf:i5=i5+15:i7=-1:goS5270
4405ifi5>200rm5tHi5=260:i7=0:ifm5tHpOv+2,pE(v+2)-
8:pO53277,2
4412pOv+4,260-i5:pOv+5,(i6+5)*8:i4=0:ifi7tHpOs+4,16
4420reT
5030ifi8<>-1tHreT
5036b=nOb:fOt=0to4:ifi9-(15-5*t)<0tHgO5060
5040?"{home}"sPi9-(15-5*t))wf$(1):?"{home}{down}"sPi9-(15-
5*t))wf$(2-b)
5060nEt
5070i9=i9+1:ifi9>14tHi8=1:i9=0
5210reT
5220ifi8<>1tHreT
5228ifm3>50tHgO5245
5230at$="   ":yy=0:xx=m1:goS6000:yy=1:goS6000
5245pOv+2,m2+3*sI(m3):pOv+3,m3:pOv+29,0:goS4380
5248m3=m3+int(rN(1)*3)+lvl:goS6090
5249ifm3<210tHifnOm5tHgO5254
5250ifm3>=210tHm7=m7+1:goS5430
5251m3=50:m1=m1-5:m2=m2-38:m4=m4+1:pOv+3,0
5254m5=0:ifm4=4tHgoS5480:goS5400
5260reT
5270fy=20+(i6+5)*8:fx=280-i5:m5=0
5285iffx>m2tHiffx<m2+24tHiffy>m3tHiffy<m3+28tHm5=-
1:m6=m6+10
5288ifnOm5tHreT
5290pOs+1,16:pOs,195:pOs+4,33:goS5400:pOs+4,16:reT
5400at$="{white}hiscore:"+stR(hs)+" lvl:"+stR(lvl-2)+"
score: "+stR(m6)+"{light green}"
5415yy=24:xx=6:goS6000:reT
5430at$=wf$(1)+"{light green}":xx=32:yy=23-4*m7:goS6000
5433at$=wf$(2-b)+"{light green}":yy=24-4*m7:goS6000
5434fOt=32to16stE-1:pOs+1,t:pOs,
179+t:pOs+4,33:nE:ifm7=4tHm8=-1
5438pOs+4,16:reT
5480i8=0:i9=0:lvl=lvl+1:m1=20:m2=190:m3=50:m5=0:m4=0:reT
5510pO53271,1:fOt=i2to225:pOv+1,t:nEt:pO53271,0:pOs+1,16:p
Os,195:pOs+4,33
5520?"{clear}":pOv+21,0:?"{white}game over":?:?"punteggio:
"+stR(m6)
5525if(m6>hs)tH?"{return}**new
hiscore**{return}":pO727,m6/10
5530?"ancora? (s/n) ":pOs+4,16
5540gEz$:if(z$)=""tHgO5540
5550ifz$="n"tHeN
5560rU
6000pO780,0:pO781,yy:pO782,xx:sY65520:?at$;:reT
6030iflvl<5orm7=0tHreT
6060at$="  ":ifint(rN(1)*5)=3tHat$=pa$
6070xx=30:yy=24-4*m7:goS6000:ifat$="  "tHreT
6078ifi2>=232-32*m7tHm8=-1
6080reT
6090i1=pE(56320)
6100ifi7=0tHyf=(i2/8)-5
6110ifi1=126ori1=110tHgoS4300
6120ifi1=125ori1=109tHgoS4350
6130ifi1=111tHi4=-1:
6135ifi7=0tHi5=29
6140ifi8=0tHi8=int(rN(1)*2)-1
6150reT
6180i=vA(mI(mu$,no,1))
6190pOs+1,hi(i):pOs,lo(i):pOs+4,17:fOt=1to100:nE:pOs+4,16:
fOt=1to50:nE
6220no=no+1:ifno>len(mu$)tHno=1
6222reT
6230s=54272:dIhi(4):dIlo(4):pOs,
15:mu$="1323132314342414243":no=1
6232hi(1)=34:hi(2)=43:hi(3)=51:hi(4)=61:lo(1)=75:lo(2)=52:
lo(3)=97:lo(4)=126
6234pOs+24,15:pOs+5,100:pOs+6,215:fOt=stos+24:pOs,
0:nE:fOt=1to20:v$=v$+"V":nE
6240?"{clear}":at$="{white}press a key"
6250xx=15:yy=12:goS6000:bb=rN(-ti):xx=1:yy=1
6255at$=" {red}p{green}o{white}o{yellow}y{purple}4{green}k
":goS6000:xx=xx+1:ifxx>100tHgO6240
6260gEs$:ifs$=""tHgoS6180:gO6255
6270reT
6280ifm9=-1tHgO6400
6300ifint(rN(1)*8)=3tHm9=-1:n1=m3:n2=m2:reT
6400n2=n2+10:n1=n1+0.5:ifn2>=255orn1>=255tHpOv+7,0:pOv+6,0
:m9=0:reT
6410pOv+6,n2:pOv+7,n1:ifn1>i2-
12tHifn1<i2+12tHifn2>242tHm8=-1
6420reT
```

Pooyan-4kb

Commento al gioco Pooyan-4kb_Nardella_Felice.bas

di Felice Nardella

Protagonista del gioco è il maialino che deve impedire ai lupi di arrivare al rifugio, in cui ci sono gli altri porcellini da preservare. Pertanto i lupi devono essere abbattuti dalle frecce, lanciate dal maialino, prima che arrivino a terra. Occorre anche evitare di farsi colpire dalle pigne lanciate dai lupi.

Si gioca col Joystick in porta 2.
Leva in alto: sposta il carrello col maialino, in alto.
Leva in basso: sposta il carrello col maialino, in basso.
Pulsante Fire: lancia la freccia (è possibile anche tenere sempre premuto il pulsante Fire se si vuole che le frecce partano di continuo, ma si può lanciare soltanto una freccia alla volta).
Se prima di totalizzare 1000 punti si perdono tutte le vite (P), il gioco si ferma e viene visualizzata la schermata che dà la possibilità di rigiocare o di uscire dal gioco.
Se si riesce a totalizzare 1000 punti, scatta il livello 2, che è più difficile del primo, perché viene raddoppiata la velocità di discesa dei lupi; in compenso le vite del maialino aumenteranno di 20.
Nel momento in cui si arriva a totalizzare 2000 punti, prima che le vite del maialino finiscano, si vince la partita e il gioco termina, offrendo la possibilità di giocare di nuovo o di uscire dal gioco.

Spiegazione del codice:

Riga 1:
v=53248:
pone nella variabile v il valore 53248 (indirizzo base del Video Interface Chip II per la creazione degli sprite);
si=54272:
pone nella variabile si il valore 54272 (locazione di partenza del Sound Interface Device);
z=56320:
pone nella variabile z il valore 56320 (locazione contenente il valore di posizione del joystick in porta2);
e=58640:
pone nella variabile e il valore 58640 (si veda più avanti a proposito del posizionamento del cursore);
vi=2040:
pone nella variabile vi il valore 2040 (locazione di partenza del puntatore allo sprite, che indica dove si trova lo sprite in memoria);
h=830:
pone nella variabile h il valore 830 (una delle locazioni di buffer del Datassette, che viene utilizzata per immagazzinare in maniera “permanente”, cioè finché non si spegne il C64 oppure finché non si usa il Datassette, il valore massimo dei punti “HI SCORE” accumulati);
l=211:
pone nella variabile l il valore 211 (valore legato al posizionamento del cursore: l per indicare la colonna ed l+3 per indicare la riga);
u=.9:
pone il valore .9 nella variabile u;
k=53280:
pone il valore 53280 (locazione colore bordo) nella variabile k;

Riga 2:
x=215:
pone in x il valore 215 (coordinata x di partenza del maialino)
y=110:
pone in y il valore 110 (coordinata y di partenza del maialino)
p=20:
pone in p il valore 20 (numero di maialini disponibili)
r=8:
pone in r il valore 8 (riga della corda)
y1=77:
pone in y1 il valore 77 (coordinata y di partenza del lupo 1)
x2=30:
pone in x2 il valore 30 (coordinata x di partenza del lupo 2)
y2=y1:
pone in y2 il valore di y1 (coordinata y di partenza del lupo 2 = coordinata y di partenza del lupo 1)
dy=8:
pone in dy il valore 8 (incremento della posizione y del maialino)
i=5:
pone in i il valore 5 (incremento/decremento della posizione y dei lupi durante il movimento)
t=10:
pone in t il valore 10 (incremento/decremento della posizione x dei lupi durante il movimento)
a=15:
pone in a il valore 15 (incremento dele posizioni x o y delle pigne durante il lancio)
hi=peek(h):
pone in hi il valore della locazione 830 (in cui viene immesso il punteggio massimo)

Riga 3:
gosub61:
rimanda alla subroutine che visualizza il title screen;
print"{clear}":

cancella lo schermo;
pokek,6:
colora bordo in blu (6)
pokek+1,6:
colora lo sfondo in blu (6)
gosub70:
rimanda alla subroutine che disegna lo sfondo.

Riga 4:
pokel+3,1:pokel,34:syse:print"{reverse on}{cyan}"hi*10:
imposta la posizione della riga e della colonna e stampa in quella posizione (riga 1, colonna 34) il valore hi*10=punteggio massimo

Riga 5:
forn=12800to13054:readq:poken,q:next:
pone nella zona di memoria compresa tra i byte 12800 e 13054 (cioè lo spazio per 4 sprites da 63 byte ognuno) i dati scritti nelle linee DATA (dalla riga 88 a 100), che rappresentano i 4 sprites

Riga 6:
pokevi,200:
inserisce nel blocco 200 (12800/64) i dati dello sprite 0 (maialino);
pokevi+1,201:
inserisce nel blocco 201 i dati dello sprite 1 (lupo 1);
pokevi+2,201:
inserisce nel blocco 201 i dati dello sprite 2 (lupo 2) che sono identici al lupo 1;
pokevi+3,202:
inserisce nel blocco 202 i dati dello sprite 3 (freccia);
pokevi+4,203:
inserisce nel blocco 203 i dati dello sprite 4 (pigna);

Riga 7:
pokev+21,7:
attiva e visualizza i primi tre sprites (maialino, lupi 1 e 2). Poiché 20 =1 attiva lo sprite 0, 21 =2 attiva lo sprite 1 e 22 = 4 attiva lo sprite 2, la loro somma (1+2+4=7) li attiva tutti e tre.
pokev+28,31:
Attiva lo sprite multicolor per tutti gli sprites (0, 1, 2, 3, 4). Infatti 20 + 21 +22 +23 + 24 = 31;
pokev+37,0:
imposta il colore multicolor 0 = nero (0);
pokev+38,8:
imposta il colore multicolor 1 = arancio (8);

Riga 8:
pokev+39,10:
imposta il colore dello sprite 0 = rosso chiaro (10);
pokev+40,2:
imposta il colore dello sprite 1 = rosso (2);
pokev+41,9:
imposta il colore dello sprite 2 = marrone (9);
pokev+42,9:
imposta il colore dello sprite 3 = marrone (9).

Riga 9:
pokesi+6,240:
imposta il Sustain-Release
pokesi+5,0:
imposta l'Attack-Decay
pokesi+3,8:
imposta il pulse wave duty cycle voice 1 high byte
pokesi+4,65:
imposta la forma d'onda voce 1

Riga 10:
j=peek(z):
inserisce in j il valore il valore di posizione del joystick in porta2
f=jand16:
controlla il bit 4 (24=16) del byte letto in z per verificare la pressione del tasto Fire
iff=0theniffs=0thenfs=1:
pone il flag di controllo per l'avvio della freccia (fs) =1 se f=0 (cioè se è premuto il tasto fire) e se il flag stesso è = 0, per evitare che parta un'altra freccia quando, ad esempio, si preme in successione veloce il pulsante fire, o lo si tiene sempre premuto.
xf=216:yf=y:gosub53:
imposta le coordinate iniziali xf e yf della freccia e rimanda alla subroutine in 53 per l'effetto sonoro

Riga 11:
iffsthengosub36:
se il flag fs è <> 0 rimanda alla subroutine in 36 che fa partire la freccia

Riga 12:
ifj=125orj=109theny=y+dy:gosub68:print"{yellow}{98}":r=r+1:
se viene abbassata la leva del joystick viene abbassato il carrello col maialino; si rimanda alla routine che controlla riga e colonna del cursore e si stampa il petscii 98 per far allungare la corda.

Riga 13:
ifj=126orj=110theny=y-dy:gosub68:print" ":r=r-1:
se viene alzata la leva del joystick viene alzato il carrello col maialino; si rimanda alla routine che controlla riga e colonna del cursore e si stampa uno spazio per far accorciare (cancellare) la corda.

Riga 14:
14ifr<8thenr=8:
si controlla l'altezza massima della corda

Riga 15:
15ifr>18thenr=18:
Si controlla la lunghezza massima della corda

Riga 16:
16ify>190theny=190:
si controlla il livello più basso del carrello

Riga 17:
17ify<110theny=110:
si controlla il livello più alto del carrello

Riga 18:
pokev,x:
viene impostata la coordinata x della posizione del carrello del maialino
pokev+1,y:
viene impostata la coordinata y della posizione del carrello del maialino
pokev+2,x1:
viene impostata la coordinata x della posizione del lupo 1
pokev+3,y1:
viene impostata la coordinata y della posizione del lupo 1
x1=x1+t:
viene incrementata la posizione orizzontale verso destra del lupo 1
ifx1>70theny1=y1+i:
viene controllato che, al raggiungimento della posizione orizzontale > 70, venga incrementato il valore della posizione verticale verso il basso del lupo 1
x1=x1-t:
viene decrementata la posizione orizzontale del lupo 1 (il lupo cadrà verso il basso rimanendo sulla stessa posizione verticale)

Riga 19:
19ifrnd(1)>uthenifx1=70thenifps=0thenps=1:xp=x1:yp=y1:gosub53:
si fa in modo che venga attivato il flag ps per far sì che non ci sia già un'altra pigna lanciata in aria, quando il numero casuale generato rnd(1) supera il valore di u; si pongono le coordinate xp e yp iniziali della pigna uguali a quelle del lupo 1; si rimanda alla subroutine che fa emettere un effetto sonoro (identico al lancio della freccia)

Riga 20:
20ify1=87thengosub56:
quando il lupo 1 comincia a scendere viene emesso un suono (riga 56)

Riga 21:
21ifpsthengosub57:
se il flag ps <> 0 viene lanciata dal lupo 1 una pigna di coordinate xp e yp identiche a quelle del lupo 1; si rimanda alla subroutine che controlla la pigna.

Riga 22:
22ify1>213thenx1=x1+t:y1=213:
si fa in modo che quando il lupo 1 in discesa arriva a terra, si incrementa la coordinata x e si lascia fissa la coordinata y per farlo avanzare verso destra.

Riga 23:
23ifx1>255thengosub55:gosub43:x1=0:y1=77:
quando il lupo 1 raggiunge la coordinata x > 255 si rimanda prima alla subroutine in 55 e poi alla subroutine in 43; si pongono le coordinate x e y del lupo 1 ai valori iniziali

Riga 24:
24pokev+4,x2:
viene impostata la coordinata x della posizione del lupo 2
pokev+5,y2:
viene impostata la coordinata y della posizione del lupo 2
x2=x2+t:
viene incrementata la posizione orizzontale verso destra del lupo 2
ifx2>130theny2=y2+i:
viene controllato che al raggiungimento della posizione orizzontale > 130, venga incrementato il valore della posizione verticale verso il basso del lupo 2
x2=x2-t:
viene decrementata la posizione orizzontale del lupo 2 (il lupo cadrà verso il basso rimanendo sulla stessa posizione verticale)

Riga 25:
ifrnd(2)>uthenifx2=130thenifps=0thenps=1:xp=x2:yp=y2:gosub53:
si fa in modo che venga attivato il flag ps per far sì che non ci sia già un'altra pigna lanciata in aria, quando il numero casuale generato rnd(2) supera il valore di u; si pongono le coordinate xp e yp iniziali della pigna uguali a quelle del lupo 2; si rimanda alla subroutine che fa emettere un effetto sonoro (identico al lancio della freccia)

Riga 26:
ify2=87thengosub56:
quando il lupo 2 comincia a scendere viene emesso un suono (riga56)

Riga27:
ifpsthengosub57:
se il flag ps <> 0 viene lanciata dal lupo 2 una pigna di coordinate xp e yp, identiche a quelle del lupo 2; si rimanda alla subroutine che controlla la pigna.

Riga 28:
ify2>213thenx2=x2+t:y2=213:
si fa in modo che quando il lupo 2 in discesa arriva a terra, si incrementa la coordinata x e si lascia fissa la coordinata y per farlo avanzare verso destra.

Riga 29:
ifx2>255thengosub55:gosub43:x2=30:y2=77:
quando il lupo 2 raggiunge la coordinata x > 255 si rimanda prima alla subroutine in 55 e poi alla subroutine in 43; si pongono le coordinate x e y del lupo 2 ai valori iniziali

Riga 30:
d=peek(v+30):
legge il valore dell'indirizzo di collisione fra sprite (53278) e lo pone in d
ifd=10thengosub60:gosub41:
verifica la collisione tra la freccia e il lupo 1; se c'è collisione, rimanda alle subroutine in 60 e poi in 41

Riga 31:
ifd=12or(d>26andd<31)thengosub60:gosub42:
verifica la collisione tra la freccia e il lupo 2; se c'è collisione, rimanda alle subroutine in 60 e poi in 42

Riga 32:
ifd=17thenxp=x2:yp=y2:pokev+21,7:gosub55:gosub43:
verifica la collisione tra la pigna e il maiale; se c'è collisione, riporta le coordinate x e y della pigna a quelle iniziali, disattiva e quindi fa sparire la pigna e rimanda alle subroutine in 55 e poi in 43

Riga 33:
ifsc=100thenifg=0thenu=.
8:i=t:p=p+20:g=1:pokel+3,1:pokel,27:syse:print"{red}2":
se si raggiungono i 1000 punti, aumenta la frequenza di lancio delle pigne; aumenta la velocità di discesa dei lupi; le vite del maialino aumentano di 20; viene stampato il 2 di colore rosso a fianco alla sritta LVL:); si fa in modo che ciò accada una volta sola (ovvero solo quando g=0)

Riga 34:
ifp=0then45:
Quando le vite del maialino finiscono, esce dal loop e va alla riga 45

Riga 35:
goto10:
Ricomincia il loop dalla riga 10

Riga 36:
xf=xf-25:
Fa diminuire di 25 il valore di xf
pokev+6,xf:
Fa muovere la freccia verso sinistra lungo l'asse x
pokev+7,yf:
Comunica la coordinata y della freccia che è fissa
pokev+21,15:
Attiva lo sprite della freccia e quindi la visualizza
ifxf<44thenfs=0:pokev+21,7:
Se la coordinata x della freccia risulta inferiore a 44 (quindi se arriva all'albero), riporta il flag fs a 0 e fa sparire la freccia

Riga 37:
return:
fine della subroutine

Riga 38:
sc=sc+1:
incrementa il valore di sc (score)
fs=0:
riporta il flag fs a 0
pokev+21,7:
rimangono attivi solo gli sprites dei maialini e dei lupi
ifsc=200then52:
se si raggiunge il punteggio, ritenuto come massimo, di 2000, salta alla riga 52

Riga 39:
vh=hi:ifsc>hithenhi=sc:pokeh,hi:
viene messa nella variabile hi il punteggio massimo e lo si scrive nella locazione 830 del buffer del Datassette

Riga 40:
pokel+3,1:pokel,16:syse:print"{cyan}"sc*10:
sposta il cursore alla posizione di coordinate (16,1) e scrive il punteggio
gosub54:
rimanda alla subroutine in 54
return:
fine della subroutine

Riga 41:
forb=0to20:pokev+2,0:pokev+3,0:pokev+2,x1:pokev+3,y1:next:x1=0:y1=77:return:
subroutine che fa lampeggiare il lupo 1 in caso di collisione con la freccia e riporta il lupo 1 alla posizione iniziale.

Riga 42:
forb=0to20:pokev+4,0:pokev+5,0:pokev+4,x2:pokev+5,y2:next:x2=15:y2=77:return:
subroutine che fa lampeggiare il lupo 2 in caso di collisione con la freccia e riporta il lupo 2 alla posizione iniziale.

Riga 43:
forb=0to20:pokev,
0:pokev+1,0:pokev,x:pokev+1,y:next:p=p-

1:c=4:ifp<10thenc=5:
subroutine che fa lampeggiare il maialino in caso di collisione con la pigna o nel caso in cui i lupi raggiungono la porta della casetta; decrementa di 1 le vite; decide la posizione della colonna in cui scrivere il numero delle vite a seconda del numero di cifre del valore di p.

Riga 44:
pokel+3,1:pokel,c:syse:print"{reverse on}{cyan}"p:return:
scrive il valore di p in posizione (c,1)

Riga 45:
gosub61:gosub69:print"{reverse on}game over":
rimanda alla subroutine in 61 e poi in 69; fa apparire la scritta

Riga 46:
ifsc>vhthenprinttab(13)"{down}{reverse on}new record!!!":
controlla se il punteggio supera il record precedente; fa apparire la scritta alla colonna 13 e si sposta di una riga sotto.

Riga 47:
print"{down} play again? (y/n)":
fa apparire il messaggio una riga sotto

Riga 48:
geta$:ifa$=""then48:
attende la pressione di un tasto

Riga 49:
ifa$="y"thenrun:
se si preme il tasto y, ricomincia il programma dall'inizio.

Riga 50:
ifa$<>"n"then48:
La pressione di un carattere diverso da "n", riporta alla riga 48

Riga 51:
print"{clear}by f. nardella":end:
Se si preme "n" viene visualizzato il messaggio e viene terminato il programma.

Riga 52:
gosub61:gosub69:print"{reverse on}you win!!":goto47:
rimanda alla subroutine in 61 e poi in 69; scrive il messaggio e salta alla riga 47

Riga 53:
pokes+24,15:form=10to20:pokes+1,m:next:pokes+1,0:pokes+24,0:return:
riproduce il suono al lancio della freccia o della pigna: inserisce il valore 15 al registro del volume (vol. massimo); inserisce i valori da 10 a 20 cambiando la frequenza del suono; riporta la frequenza alta a 0; riporta il volume a 0

Riga 54:
pokes+24,15:form=60to40step-1:pokes+1,m:next:pokes+1,0:pokes+24,0:return:
riproduce il suono quando la freccia colpisce un lupo: inserisce il valore 15 al registro del volume (vol. massimo); inserisce i valori decrescenti da 60 a 40 cambiando la frequenza del suono; riporta la frequenza alta a 0; riporta il volume a 0

Riga 55:
pokes+24,15:form=30to10step-1:pokes+1,m:next:pokes+1,0:pokes+24,0:return:
riproduce il suono quando il maialino perde una vita: inserisce il valore 15 al registro del volume (vol. massimo); inserisce i valori decrescenti da 30 a 10 cambiando la frequenza del suono; riporta la frequenza alta a 0; riporta il volume a 0

Riga 56:
pokes+24,15:form=5to30:pokes+1,m:next:pokes+1,0:pokes+24,0:return:
riproduce il suono quando i lupi cominciano a scendere dall'albero: inserisce il valore 15 al registro del volume (vol. massimo); inserisce i valori crescenti da 5 a 30 cambiando la frequenza del suono; riporta la frequenza alta a 0; riporta il volume a 0

Riga 57:
xp=xp+15:yp=yp-5:pokev+8,xp:pokev+9,yp:pokev+21,23:ifxp>155then yp=yp+15:
controlla il movimento della pigna (lo muove verso destra e verso l'alto); attiva lo sprite della pigna; quando la pigna supera la coordinata x = 155 la pigna devia verso il basso.

Riga 58-59:
ifxp>210thenps=0:xp=x2:yp=y2:pokev+21,7:
return:
disattiva lo sprite della pigna quando supera la coordinata x = 210; riporta alla posizione iniziale la pigna; disattiva il flag ps.

Riga 60:
pokev+21,7:d=peek(v+30):gosub38:return:
rimangono attivi solo gli sprite del maialino e dei due lupi; rimanda alla subroutine alla linea 38

Riga 61:
print"{clear}{yellow}":forn=0to9:pokev+n,0:next:pokek,0:pokek+1,0:
cancella lo schermo e colora di giallo il cursore; riporta a 0,0 le coordinate di tutti gli sprites (quindi

spariscono dal video); colora di nero bordo e sfondo.

Riga 62:
ifothen67:
quando la subroutine viene eseguita dalla seconda volta in poi, salta alla riga 67

Riga 63:
printtab(15)"{reverse on}pooyan-4kb":
scrive il messaggio con 15 spazi verso destra

Riga 64:
printtab(10)"{down*2}use joystick in port2":
scrive il messaggio con 10 spazi verso destra e due verso il basso

Riga 65:
printtab(11)"{down}press fire to start":o=1:
scrive il messaggio con 11 spazi verso destra e uno verso il basso; pone o = 1

Riga 66:
j=peek(z):ifj<>111then66:
return:
controlla il joystick; se viene premuto il pulsante Fire, salta alla linea 66

Riga 68:
pokel+3,r:pokel,25:syse:return:
sposta il cursore alla riga r, colonna 25

Riga 69:
pokel+3,10:pokel,15:syse:return:
sposta il cursore alla riga 10 colonna 15

Riga 70-92:
70print" {reverse on}{cyan}UCCCCCI{reverse off}
{reverse on}UCCCCCCCCI";
71print"{reverse off} {reverse on}Bp:{160}
20B{reverse off} score: 0 lvl: 1 {reverse on}Bhi:{160}
B";
72print"{reverse off} {reverse on}JCCCCCK{reverse
off} {reverse on}JCCCCCCCCK";
73print"{down}{reverse off} {pink}Q Q"
74print" {reverse on}{cyan}UI{reverse off}{yellow}
DD{pink}D{reverse on} {reverse off}D{reverse on} "
75print"{reverse on}{green} {reverse off}{169}
{reverse on}{cyan}JK{reverse off} {reverse on}
{green} "
76print"{reverse on} {reverse off}{169} {yellow}B
{cyan}M{reverse on}{brown} # "
77print"{reverse on}{green} {reverse off}{169}
{reverse on}{brown} #{160}{160} "
78print"{reverse on}{green} {reverse off}{169}
{reverse on}{brown} #{160} {reverse off}{green}
{185}"
79print"{reverse on}{brown} {160}{reverse off}
{reverse on} #{160} {reverse off}{169}"
80print"{reverse on} {160}{reverse off} {green}{185}
{reverse on}{brown} #{160} "
81print"{reverse on} {160}{reverse off} {127}
{reverse on} #{160} "
82print"{reverse on} {160}{reverse off} {reverse on}
"
83print"{reverse on} {160}{reverse off} {reverse on}
"

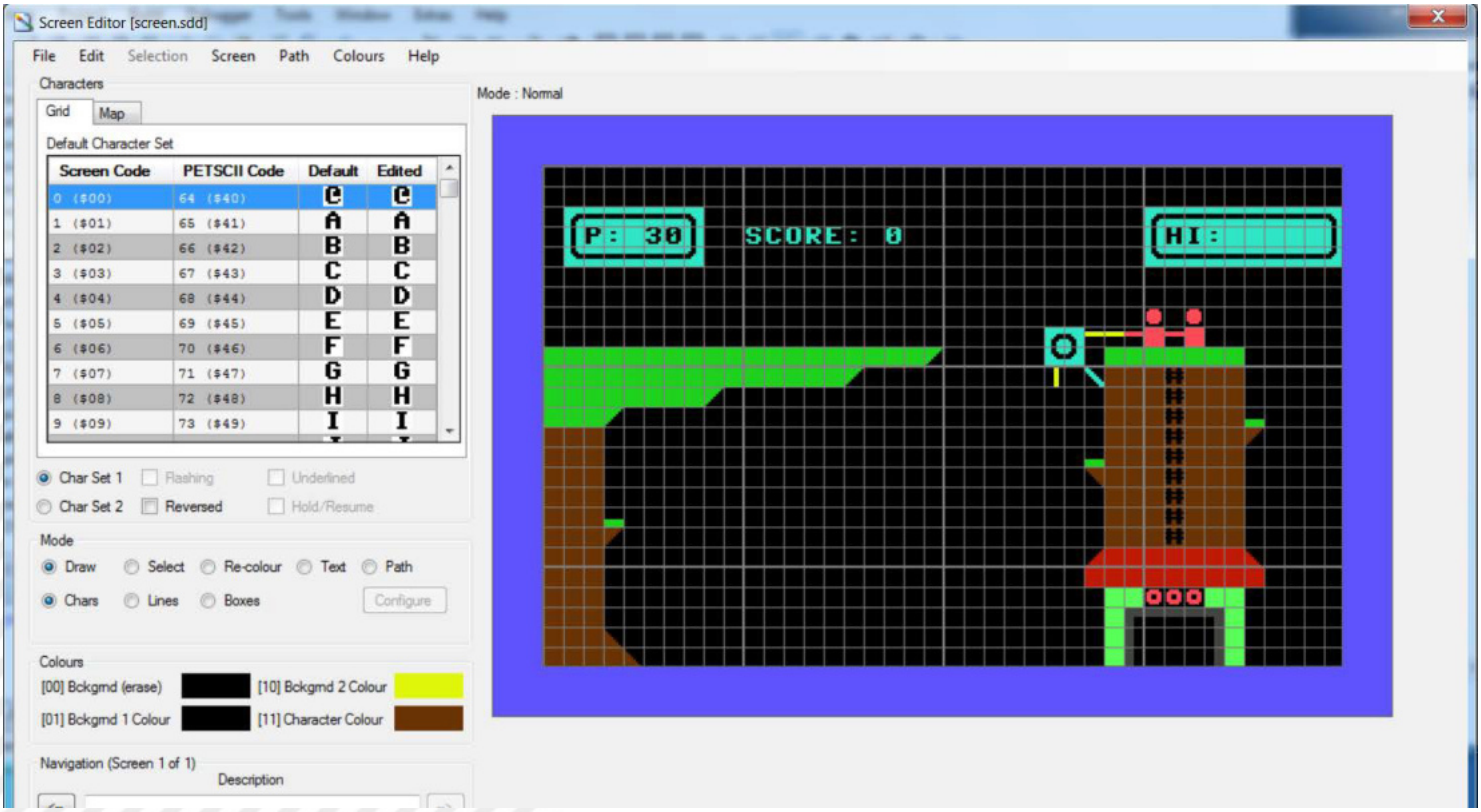
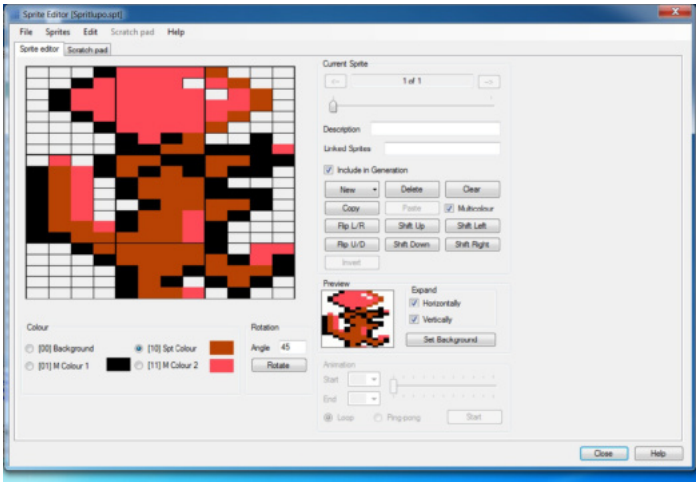
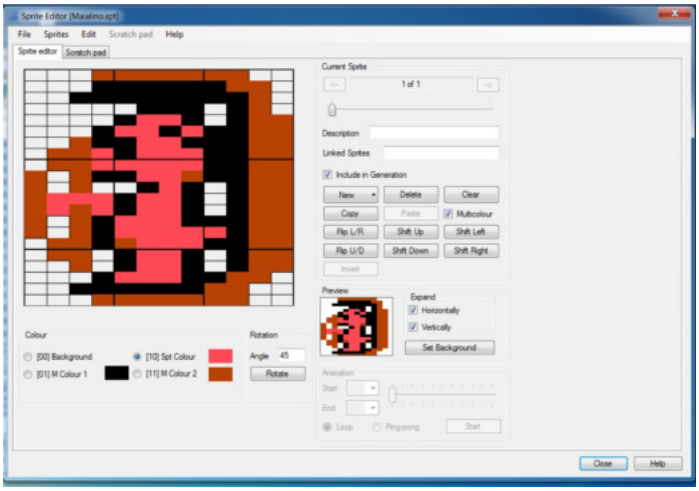


Figura 1 - Lo sfondo realizzato con lo Screen Editor

```
84print"{reverse on} {160}{reverse off}{green}{185}
{reverse on}{brown} # "
85print"{reverse on} {160}{reverse off}{169}
{reverse on} # "
86print"{reverse on} {160}{reverse off} {reverse on}
{red}{169} {127}"
87print"{reverse on}{brown} {160}{reverse off}
{reverse on}{red} "
88print"{reverse on}{brown} {160}{reverse off}
{reverse on}{light green} {reverse off}{pink}
WWW{reverse on}{light green} "
89print"{reverse on}{brown} {160}{reverse off}
{reverse on}{light green} {dark gray}{172}{162}
{162}{162}{187}{light green} "
90print"{reverse on}{brown} {160}{127}{reverse
off} {reverse on}{light green} {reverse off}{dark gray}
{161} {reverse on}{161}{light green} "
91print"{reverse on}{brown} {160}{160}{127}
{reverse off} {reverse on}{light green} {reverse off}
{dark gray}{161} {reverse on}{161}{light green} "
92print"{reverse on}{gray} pooyan by fn ":return
```

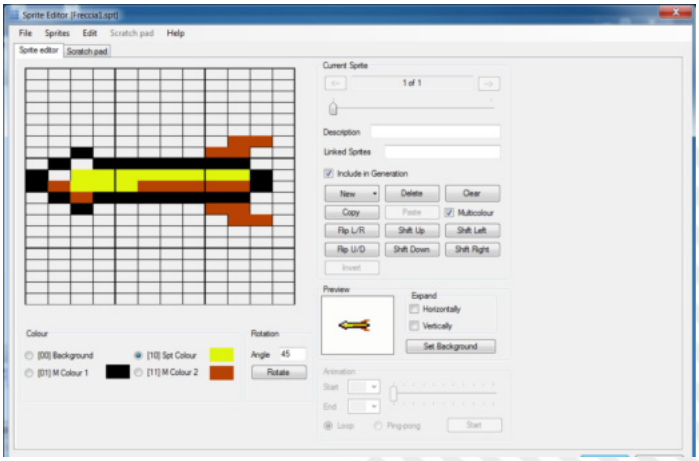


Disegnano lo sfondo su schermo, attraverso le istruzioni PRINT; vengono stampati su schermo i petscii che danno origine al disegno di sfondo. Le righe sono state ottenute e generate tramite il software di sviluppo CBM Prg Studio v3.13.0 (http://www.ajordison.co.uk/index.html).
Come si vede in figura 1 lo sfondo è stato sviluppato tramite lo Screen Editor del software stesso:



Righe 93-102:

```
93data3,255,240,1,85,124,5,85,92,,21,28,,105,31,1,
166,95
94data13,105,95,62,106,95,62,170,95,205,107,95,2
04,25,31,234,105,31
95data238,105,31,205,169,95,205,170,159,61,234,
95,60,105,31,12,105,31
96data1,169,124,5,85,240,3,255,240,,1,255,128,7,2
52,224,31,255,56,31,255,248,7
97data255,224,1,255,128,,118,4,,90,87,49,165,165,
109,153,148,108,106,64,108,106,84
98data108,26,164,108,107,144,107,107,64,90,171,
64,6,170,79,1,170,157,,105,169,
99data100,100,,107,16,,,,,,,,,,,,,,,,,,,,,,,,,,60,4,,240,17,8
5,80,74,170
100data164,122,191,244,29,85,80,4,,240,,,60,,,,,,,,,,
,,,,,,,,,,20,,
101data105,,1,150,64,6,105,144,9,215,96,7,125,20
8,13,215,112,3,125,192,1
102data215,64,3,125,192,,215,,,125,,,20,,,,,,,,,,,,,,,,,,,,
,,,,,
```



Informazioni lette da READ Q alla linea 15, che definiscono gli sprites (maialino, freccia, lupo e pigna). Sono stati ottenuti tramite lo sprite editor del software CBM Prg Studio, come si vede nelle figure qui a lato.

Calcolare in multipla precisione-parte I

di Alberto Apostolo

Questo articolo rappresenta la prima puntata di una serie di pubblicazioni sul tema dei calcoli in multipla precisione o precisione arbitraria. Nel numero 13 di RetroMagazine sono stati presentati due programmi scritti in C per calcolare abbastanza velocemente pi greco (e il suo reciproco) con 2000 cifre decimali ed è stato accennato alle esigenze che spingono a estendere le capacità di calcolo di un computer (cfr. "RetroMath: Ritorna la sfida del pi greco", RM 13).

I tipi di dato intero e virgola mobile (floating-point) disponibili su quasi tutti i linguaggi programmativi possono rivelarsi insufficienti quando è richiesto calcolare in modo estremamente preciso le soluzioni di alcuni problemi di Fisica e Chimica oppure quando occorre elaborare numeri di enorme grandezza. Ad esempio, l'algoritmo crittografico RSA ha bisogno di chiavi costituite da numeri di almeno 1000 cifre per resistere agli attacchi degli analisti crittografi ([SD06], [Ski09]).

La strategia più comune per superare tali restrizioni è quella di costruire una aritmetica in multipla precisione, adoperando lunghe stringhe di byte che generalizzano la rappresentazione dei numeri in virgola mobile con mantissa ed esponente [MBD18].

Nel 1976 Brent e Salamin scrissero la libreria m.p. in FORTRAN (un insieme di funzioni e utility per gestire questo nuovo tipo di dati). Negli anni successivi furono pubblicati i lavori di Bailey (MPFUN e ARPREC), Cohen (PARI), Zimmermann (MPFR e MPFI) e Granlund (GNU MP, noto anche come GMP) [Mul06].

Per scopi ludico-didattici tenteremo di realizzare una nostra "libreria" di routine "fatte in casa" dove i numeri in m.p. sono vettori di numeri interi, con un approccio diverso per programmare con una certa semplicità e immediatezza.

Invece in ambito professionale, è sempre cosa buona e giusta affidarsi a librerie di software già certificato e garantito senza dover "inventare la ruota" ogni volta. Per questo motivo, nella sezione dei "Link utili", in fondo all'articolo, sono elencati indirizzi dove è possibile scaricare alcune librerie m.p. citate in precedenza.

Multipla precisione con virgola fissa

Fissato un numero intero B maggiore o uguale a 2, sia x un numero reale costituito un numero finito di cifre x_k tali che

$$0 \leqslant x_k < B \ , \ k = -m,\dots,0,1,\dots,n-2$$

$$x_{n-1} > 0 \ .$$

La notazione in Figura 1 (convenzionalmente adottata) si dice *rappresentazione posizionale di x in base B*.

$$x_B = (-1)^s [x_{n-1}\dots x_1 x_0 \bullet x_{-1} x_{-2} \dots x_{-m}] =$$

$$= (-1)^s \left(\sum_{k=-m}^{n-1} x_k B^k \right)$$

Figura 1

Il punto che separa la parte intera dalla parte frazionaria è il *punto decimale* se B=10, *punto binario* se B=2. Il numero x è *positivo* se s=0,*negativo* se s=1. I numeri con rappresentazione finita di cifre costituiscono una approssimazione dei numeri reali [QSS14].

A causa della precisione finita della parte frazionaria, un numero reale x in base B si può troncare o arrotondare a m cifre.

Il *troncamento* (chopping) consiste nell'ignorare le cifre successive a x_{-m}. L'*arrotondamento* (rounding o round-off) consiste nel conservare tutte le cifre fino a $x_{-(m+1)}$ e poi procedere al calcolo della cifra X di posto -m con le regole:

$$X = x_{-m} \qquad se \ x_{-(m+1)} < B/2$$

$$X = x_{-m} + 1 \qquad se \ x_{-(m+1)} \geqslant B/2 \ .$$

Lavorando in base 10, al fine di ridurre gli errori sistematici, il meccanismo di arrotondamento si può raffinare mediante le regole [DF11]:

1) se la prima cifra da eliminare è 0,1,2,3,4 allora la cifra meno significativa resta inalterata (arr. per difetto),

2) se la prima cifra da eliminare è 6,7,8,9 oppure 5 seguito da almeno una cifra diversa da 0, allora la cifra meno significativa rimasta viene maggiorata di una unità (arr. per eccesso)

3) se la prima cifra da eliminare è 5 seguito solo da zeri, allora la cifra meno significativa resta inalterata quando è pari e maggiorata di una unità quando è dispari (regola del numero pari).

Nella rappresentazione detta in virgola fissa (fixed point), un calcolatore assegna n cifre per la parte intera e m cifre per la parte frazionaria.

Un numero reale x in base B può essere approssimato da un numero in virgola fissa se si rispetta la relazione $abs(x) < (B^n - 1)$. In caso contrario si dice che si è verificato un *overflow* o *traboccamento*.

L'uso della virgola fissa limita enormemente il valore minimo e massimo dei numeri rappresentabili su un calcolatore, a meno di non impiegare molte locazioni di memoria [QSS14]. Per questo esiste la rappresentazione dei numeri in virgola mobile con mantissa ed esponente, molto simile alla notazione scientifica delle grandezze fisiche $G = M * 10^Z$ dove $0 < M < 10$ e Z è numero intero con segno [DR05].

In generale la rappresentazione in virgola fissa si trova nel COBOL (un linguaggio per applicazioni commerciali) dove i numeri spesso esprimono somme di denaro o percentuali, mentre nei linguaggi per applicazioni scientifiche (come

il FORTRAN) i numeri in virgola mobile esprimono la parte scalare di grandezze fisiche.

Un modo per implementare un numero in precisione multipla è quello di memorizzare le cifre da cui è composto in un vettore di numeri interi. Se si sceglie la virgola fissa accetteremo tutte le limitazioni legate a tale rappresentazione ma la programmazione si semplificherà molto (ad esempio, sarà facile da visualizzare e cade l'obbligo di "normalizzare" il risultato alla fine di ogni operazione). Un ulteriore alleggerimento deriverà dalla adozione della regola del troncamento al posto di quella dell'arrotondamento.

Con il linguaggio C, una possibile realizzazione di un tipo di dato in multipla precisione sarà quella riportata nel Listato 1.

Il segno memorizzato a parte in una variabile “long” avrà valore +1 se la variabile m.p. è maggiore di zero, -1 se minore di zero, altrimenti zero.

Nel vettore non si userà la posizione zero al fine di facilitare la traduzione in altri linguaggi dove il range degli indici di un vettore è [1:LMAX]. Inoltre è opportuno osservare che in una variabile di tipo "long" (costituita da 32 bit) si può memorizzare più di una cifra e utilizzare una base $B=10^N$. Tuttavia l'aritmetica degli interi a 32 bit pone dei limiti a N. Per non "sballare" i calcoli a 32 bit, vale la limitazione $10^{2N} < 2^{31}$ che porta a scegliere N=4 e dunque B=10000.

L'idea di usare una base $B=10^N$ è diffusa presso numerosi autori. Ad esempio, in Rete si trova una riproduzione di "Aritmetica a precisione multipla" di Cristiano Teodoro, un brillantissimo articolo pubblicato nel 1986 su MC Microcomputer [Teo86].

Al giorno d’oggi sono a disposizione portatili a 64 bit e nel frattempo sono comparsi compilatori C freeware che aderiscono allo standard denominato C99, il quale offre variabili intere di tipo "long long

```
#define LMAX     500L

typedef struct { long segno;              /* segno (-1, 0,+1)        */
                 long cifra[(LMAX+1)];    /* cifre base scelta       */

       } MultPrec;
```

Listato 1

```
/*-------------------------------------------------------------------------*/
/* Funzione: inizializzazione con un numero z                              */
/* a    (numero m.p. dove:                                                 */
/*                          ° a.segno è intero,                            */
/*                          ° a.cifra[] vettore di interi)                 */
/* l    (num. int. elementi vettore a, partendo da 1)                      */
/* lpi  (num. int. elementi parte intera di a)                             */
/* beta (num. int. base aritmetica in m.p.)                                */
/* z    (num. int. con segno)                                              */
/* rp   (num. int. che contiene il riporto)                                */
/* i    (num. int. di lavoro)                                              */
/* irp  (num. int. di lavoro)                                              */
/*-------------------------------------------------------------------------*/
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);
     }
}
```

Init.C

```
/*-------------------------------------------------------------------------*/
/* Funzione: moltiplicazione corta a=a*z in multipla precisione            */
/* a    (numero m.p. dove:                                                 */
/*                          ° a.segno è intero,                            */
/*                          ° a.cifra[] vettore di interi)                 */
/* l    (num. int. elementi vettore a, partendo da 1)                      */
/* lpi  (num. int. elementi parte intera di a)                             */
/* beta (num. int. base aritmetica in m.p.)                                */
/* z    (num. int. con segno)                                              */
/* rp   (num. int. che contiene il riporto)                                */
/* sz   (num. int. che contiene il segno di z)                             */
/* az   (num. int. che contiene z in valore assoluto)                      */
/* i    (num. int. di lavoro)                                              */
/* izer (num. int. = 1 se almeno una "cifra" di a è > 0, 0 altrimenti)*/
/* ipro (num. int. di lavoro a 64 bit)                                     */
/* w    (num. int. di lavoro a 64 bit)                                     */
/*-------------------------------------------------------------------------*/
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  variabili 64 bit */
        ipro = (ipro * az) + *p_rp;
        *p_rp=ipro/beta;
        w = beta;                       /*workaround  variabili 64 bit */
        (*p_a).cifra[i] = ipro - (*p_rp * w);
        if ((*p_a).cifra[i] > 0L) {izer= 1L;}
     }
     (*p_a).segno=(*p_a).segno * sz * izer;
}
```

Mulz.C

int" a 64 bit. Allora ho deciso mettere un po' di "peperoncino", modificando le routine presentate nel numero 13 di RM con l'utilizzo di una variabile "buffer" a 64 bit e il passaggio alla base $B=10^9$ (cosa che massimizza il numero di cifre memorizzate in un intero "long").
Lo standard C99 si rifersce allo standard ISO/IEC 9899:1999. In seguito è stato rimpiazzato dallo standard C11 (ISO/IEC 9899:2011) e ancora dal C17/C18 (entrambi indicano lo standard ISO/IEC 9899:2018, fonte: Wikipedia).
Di conseguenza, il glorioso ma vetusto TURBO C 2.01 della Borland (su Pentium IV) è stato sostituito dal "Pelles C", installato sul mio piccolo netbook a 64 bit (per il download e altre informazioni, vedi Appendice).

Inizializzare una variabile m.p.

La routine Init.C assegna un numero intero z (compreso tra -2^{31} e 2^{31}) a una variabile *A* in multipla precisione. In pratica si cerca di "spalmare" il valore assoluto dell'intero z sui vari elementi del vettore delle "cifre" di A. Funziona anche nel caso in cui la variabile A in multipla precisione non abbia parte intera. In questo caso, il segno di z si assegna al segno di A ma poi si ha un "overflow" perché il valore assoluto di z resta assegnato alla variabile di riporto rp.

Somma algebrica tra variabili m.p.

Nella routine Suma.C la modalità di calcolo della somma algebrica "s=s+a" in multipla precisione è esattamente quella che eseguirebbe manualmente un operatore umano, tenendo conto delle casistiche elencate nel Listato 2. La somma algebrica avviene effettuando non più di due scansioni dei vettori delle cifre (una per l'eventuale confronto tra s ed a e l'altra per la somma algebrica). Pertanto si dice che la somma ha complessità computazionale lineare.

```
 1) se s <> 0 e a = 0 allora s resta invariato
 2) se s = 0 e a <> 0 allora a = s
 3) se s<>0 ed a<>0 hanno segno diverso allora
    3.1) se abs(s) maggiore di abs(a) allora
                              ° sottrazione "cifre" di a a quelle di s
                              ° il segno di s resta invariato
    3.2) se abs(s) minore di abs(a) allora
                              ° sottrazione "cifre" di s a quelle di a
                              ° il segno di s diventa il segno di a
 4) se s<>0 ed a<>0 hanno lo stesso segno allora
                         ° si sommano alle "cifre" di s quelle di a
                         ° il segno di s resta invariato
```

Listato 2

```
/*--------------------------------------------------------------------*/
/* Funzione: somma algebrica s=s+a in multipla precisione             */
/* s    (numero m.p. dove:                                            */
/*                          ° s.segno è intero,                       */
/*                          ° s.cifra[] vettore di interi)            */
/* a    (numero m.p. dove:                                            */
/*                          ° a.segno è intero,                       */
/*                          ° a.cifra[] vettore di interi)            */
/* l    (num. int. elementi vettore a, partendo da 1)                 */
/* beta (num. int. base aritmetica in m.p.)                           */
/* icnf (num. int. = 1 se abs(s)>abs(a), -1 se abs(s)<abs(a)          */
/*                                    ,  0 altrimenti       )        */
/* izer (num. int. = 1 se almeno una "cifra" di a è > 0, 0 altrimenti)*/
/* rp   (num. int. che contiene il riporto o il prestito)             */
/* i    (num. int. di lavoro)                                         */
/* isom (num. int. di lavoro)                                         */
/* sgns (num. int. di lavoro, memorizza il nuovo segno di s)          */
/*--------------------------------------------------------------------*/
void Suma(long beta,long l,MultPrec *p_s,MultPrec *p_a,long *p_rp)
{    long i,isom,izer=0L,icnf=0L,sgns;
     *p_rp=0L;
     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 {
            sgns=(*p_s).segno;
            if ((*p_s).segno != (*p_a).segno) {
               for (i = 1L;i <= l;i++) {
                  if ((*p_s).cifra[i] > (*p_a).cifra[i]) {
                     icnf= 1L;
                     break;
                  }
                  if ((*p_s).cifra[i] < (*p_a).cifra[i]) {
                     icnf=-1L;
                     sgns = (*p_a).segno;
                     break;
                  }
               }
            }
            for (i = l;i >= 1L;i--) {
               if ((*p_s).segno != (*p_a).segno) {
                  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=sgns * izer;
        }
     }
}
```

Suma.C

Moltiplicazione corta variabili m.p.

In alcuni testi, la moltiplicazione in base 10 di un numero intero per un numero intero a una cifra sola è chiamata moltiplicazione corta (per esempio 5672 * 8 = 45376). Nel caso di una base 10N, un numero z a "una cifra sola" è tale che 0 <= z < 10N. Nel nostro caso è possibile moltiplicare per numeri z tali che -10N <= z <= +10N effettuando una sola scansione del vettore delle "cifre" del numero in m.p. (la moltiplicazione corta ha complessità computazionale lineare).
Quando z = 10^x (per 1 <= x <= 9) la routine Mulz ha l'effetto di uno shift verso sinistra di x cifre in base 10 del numero a in multipla precisione (con attenzione al valore del riporto rp per testare eventuali "overflow").
Nella routine Mulz.C si trova un "workaround" sull'uso delle variabili intere a 64 bit discusso in Appendice.

```
/*-------------------------------------------------------------------*/
/* Funzione: divisione corta a=a/z in multipla precisione            */
*/ a    (numero m.p. dove:                                           */
*/                        ° a.segno è intero,                        */
*/                        ° a.cifra[] vettore di interi)             */
*/ l    (num. int. elementi vettore a, partendo da 1)                */
*/ lpi  (num. int. elementi parte intera di a)                       */
*/ beta (num. int. base aritmetica in m.p.)                          */
*/ z    (num. int. con segno)                                        */
*/ rd   (num. int. che contiene il resto)                            */
*/ sz   (num. int. che contiene il segno di z)                       */
*/ az   (num. int. che contiene z in valore assoluto)                */
*/ i    (num. int. di lavoro)                                        */
*/ izer (num. int. = 1 se almeno una "cifra" di a è > 0, 0 altrimenti)*/
*/ idiv (num. int. di lavoro a 64 bit)                               */
*/ w    (num. int. di lavoro a 64 bit)                               */
/*-------------------------------------------------------------------*/
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  variabili 64 bit */
        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;
}
```

Divz.C

Divisione corta variabili m.p.

Analogamente alla moltiplicazione, la divisione in base 10 a una cifra sola è chiamata divisione corta (per esempio 45376 / 8 = 5672). Nel caso di una base 10^N, un numero z a "una cifra sola" è tale che 0 <= z < 10N. Nella routine Divz.C è possibile dividere per numeri z diversi da zero tali che -10N <= z <= +10N effettuando una sola scansione del vettore delle "cifre" del numero in m.p. (anche la divisione corta ha complessità computazionale lineare).
Quando z = 10^x (per 1 <= x <= 9) la routine Divz ha l'effetto di uno shift verso destra di x cifre in base 10 del numero a in multipla precisione.
A differenza delle routine precedenti che sono esatte, con la divisione corta si possono perdere cifre significative con un errore $err\_div <= Beta^{-K}$ dove K è il numero di celle riservate alla parte frazionaria e Beta la base scelta.
Anche in Divz si applica un "workaround" sulle variabili a 64 bit.

```
/*-------------------------------------------------------------------*/
/* Funzione : visualizza numero in multipla precisione base 10^9     */
*/ a    (numero m.p. dove:                                           */
*/                        ° a.segno è intero,                        */
*/                        ° a.cifra[] vettore di interi)             */
*/ lv   (num. int. elementi da visualizzare, partendo da 1)          */
*/ lpi  (num. int. elementi parte intera di a)                       */
*/ vbl  (stringa che identifica la variabile da visualizzare)        */
*/ i    (num. int. di lavoro)                                        */
*/ j    (num. int. di lavoro)                                        */
/*-------------------------------------------------------------------*/
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 */
      }
}
```

DisplayMultPrec.C

```
Programma MP_TEST1 : INIZIO ELABORAZIONE

ps     = +                    000000000 000000000 000000000 000000000 000000000 000000000 000000000 001805300
           .001261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261
            261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261
            261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261
            261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261 261261261
            261261261 261261261 261261261 261261261 261261261 261261261 261261261

Programma MP_TEST1 : FINE ELABORAZIONE


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

VISUALIZZARE UNA VARIABILE M.P.

La routine DisplayMultPrec permette di visualizzare un numero in multipla precisione e una stringa di sei caratteri dove si assegna un nome (di solito il nome della variabile m.p. dichiarata dall'utente). Uno dei parametri è lv, contenente quante sono le "cifre" visualizzabili al fine di non mostrare le ultime affette da errori di calcolo dovuti al troncamento e alla discretizzazione degli algoritmi utilizzati.

UN PROGRAMMA DIMOSTRATIVO

Il programma MP_TEST1.C (di cui per brevità si darà solo il main alla fine dell'articolo) è un esempio per applicare le routine illustrate in precedenza.

APPENDICE

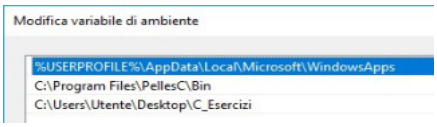
Il "Pelles C" è freeware (vedi sez. "Link utili") ed è facilissimo da installare (consultare le indicazioni relative al sistema operativo della propria macchina).
Quando l'installazione è terminata, coloro che lavorano in ambiente Windows devono lanciare il file comandi povars64 in C:\Program Files\PellesC\Bin che serve per settare le variabili di ambiente su una macchina a 64 bit). Se non si vuole utilizzare l'IDE messo a disposizione e lanciare il compilatore dal Prompt del DOS da

```
/*************************************************************************/
/* Programma: MP_TEST1                                                   */
/* Funzione : Programma dimostrativo                                     */
/*************************************************************************/
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>       /* tastiera-video PC IBM amb. Windows MS-DOS */
#define LMAX        57L  /* 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                              */
void Init(long,long,long,MultPrec *p_a,long,long *p_rp);
void Suma(long,long,MultPrec *p_s,MultPrec *p_a,long *p_rp);
void Mulz(long,long,MultPrec *p_a,long,long *p_rp);
void Divz(long,long,MultPrec *p_a,long,long *p_rp);
void DisplayMultPrec(long,long,MultPrec *p_a,char *s);
/* Programma principale                                                */
int main(void)
{
      long beta = BASE;        /* base aritmetica multipla precisione */
      long l    = LMAX;        /* lunghezza totale vettore mult.prec. */
      long lpi  =  8L;         /* lunghezza parte intera              */
      long lv   = l-2L;        /* numero elementi visualizzabili      */
      MultPrec  a,b,s;         /* variabili multipla precisione       */
      MultPrec  *p_a =&a;      /* puntatore var. mult.prec.           */
      MultPrec  *p_b =&b;      /* puntatore var. mult.prec.           */
      MultPrec  *p_s =&s;      /* 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           */
/*                                                                    */
      _clrscr();               /* pulisci schermo                     */
      printf("Programma MP_TEST1 : INIZIO ELABORAZIONE\n");
/*                                                                    */
      Init(beta,l,lpi,p_a,1L,p_rp);
      b=a;                              /* esempio assegnazione m.p.  */
      Divz(beta,l,p_b,555L,p_rd);
      Divz(beta,l,p_b,1000L,p_rd);      /* shift a destra             */
      Init(beta,l,lpi,p_s,2579L,p_rp);
      Suma(beta,l,p_s,p_b,p_rp);
      Mulz(beta,l,p_s,100L,p_rd);       /* shift a sinistra           */
      Mulz(beta,l,p_s,7L,p_rd);
/*                                                                    */
      strcpy(nomevbl,"ps ");
      DisplayMultPrec(lv,lpi,p_s,nomevbl);
/*                                                                    */
      printf("\n\nProgramma MP_TEST1 : FINE ELABORAZIONE\n\n");
      return 0;
}
```

Main del programma MP_TEST1.C

una cartella utente, occorre aggiungere alla variabile ambiente PATH la cartella C:\Program Files\PellesC\Bin riservata al compilatore e un'altra per i codici sorgente.
La variabile di ambiente PATH si modifica dal Panello di Controllo (su Internet si trovano diversi tutorial che spiegano come fare).




```
/* programma n_fatt per testare il
compilatore */
#include<stdio.h>
#include<conio.h>

long long int fac(int x) {
        if (x<2)
                return 1;
        int i;
        long long int t;
        t=x;
        for (i=2;i<x;i++)
                t*=i;
        return t;
}

int main(void) {
        long long x = fac(20);
        _clrscr();
        printf("%lld\n", x);
        return 0;
}
```

Il comando di compilazione-linkaggio è semplicemente "cc <nomefile>.c" (per aggiungere altre opzioni, consultare la documentazione riguardante il compilatore).
Il programma di prova n_fatt.c calcola 20!= 2432902008176640000. Se il risultato è diverso allora controllare se il compilatore sia stato settato correttamente.

ATTENZIONE! Durante le prove delle routine che usavano variabili a 64 bit, ho riscontrato un fastidioso fenomeno del quale fornisco un esempio.
Se a = 777777 , b = 888888 , c = 333333 sono "long" e w è "long long int", l'assegnazione "w = a * b + c;" viene comunque calcolata a 32 bit e va in overflow (nonostante w sia a 64 bit).
Infatti w dovrebbe assumere il valore 691356975309 e invece assume valore -132759347.
Per fare in modo che il compilatore sia in grado di dimensionare opportunamente lo spazio di memoria dei risultati intermedi di calcolo, ho adottato il seguente workaround:
1) in "w = espressione;" verificare quale è il tipo di dato con sizeof() maggiore tra w , le variabili presenti nella espressione e le operazioni che potrebbero richiedere maggiore spazio per non andare in overflow,
2) spezzare in due parti l'assegnazione usando una variabile u di lavoro dichiarata usando il tipo più "spazioso",
3) usare l'assegnazione "u = x;" dove x è una variabile appartenente alla espressione dove siamo certi che sarà usata per il primo calcolo intermedio,
4) modificare "w = espressione" (mettendo u al posto di x, ed eventualmente le parentesi, per essere certi che il parsing avvenga come desiderato).
Nel nostro caso, tutto funziona se si spezza l'espressione in "u = a;" e poi "w = u * b + c;" dove u è "long long int".
Invece non funziona se si scrive "u = c;" e "w = u + a * b;" (l'overflow si materializzerà ancora a causa del prodotto "a * b" gestito con il tipo “long” ed elaborato per primo grazie alla regola delle precedenze nelle operazioni).
Sospettando un bug, ho chiesto spiegazioni all'autore del compilatore assicurandomi che è tutto regolare.
Infine (per chi usa i sistemi con Windows) fate attenzione agli aggiornamenti degli strumenti di rimozione malware che potrebbero cancellare file come POCC.EXE o POLINK.EXE, costringendo a dovere reinstallare il compilatore.

Link utili

High-Precision Software Directory: *https://www.davidhbailey.com/dhbsoftware/*
GNU MP: *https://gmplib.org*
MPFR: *https://www.mpfr.org*

Pelles C: *www.smorgasbordet.com/pellesc/*

Bibliografia

[DF11] G. Dalba, P. Fornasini, "Esercizi di Fisica: Meccanica e Termodinamica", Springer Science & Business Media, 2011.
[DR05] M. Dapor, M. Ropele, "Elaborazione dei dati sperimentali", Springer, 2005.
[MBD18] J.M. Muller, N. Brunie, F. de Dinechin et alt., "Handbook of Floating-Point Arithmetic", Birkhäuser, 2018.
[Mul06] J.M. Muller, "Elementary functions: algorithms and implementation", II ed. , Springer, 2006.
[QSS14] A. Quarteroni, R. Sacco, F. Saleri, P. Gervasio, "Matematica Numerica", IV ed., Springer Science & Business Media, 2014.
[SD06] T. St Denis, "Implementing Cryptographic Multiple Precision Arithmetic", Elsevier, 2006.
[Ski09] S.S. Skiena, "The Algorithm Design Manual", Springer Science & Business Media, 2009.
[Teo86] C. Teodoro, "Aritmetica a precisione multipla", MC Microcomputer n.56 (Ottobre 1986), pagg.154-159, *https://issuu.com/adpware/docs/mc056*

RetroMath: la lunga lotta per la sopravvivenza

di Giuseppe Fedele

La demografia è la scienza che studia quantitativamente i fenomeni che riguardano la popolazione.
Questi fenomeni sono oggi più che mai importanti per una serie di motivi, primo tra tutti l'incremento della popolazione mondiale che, in un mondo con risorse limitate, costituisce uno dei problemi con cui i leader mondiali devono scontrarsi. Le statistiche suggeriscono che nel 2100 la popolazione mondiale raggiungerà quota 11,2 miliardi di abitanti, un numero enorme, che per molti studiosi sarà correlato ad una grave crisi che riguarderà sicuramente la scarsità di risorse soprattutto energetiche e idriche e in secondo luogo la salvaguardia dell'ambiente e degli ecosistemi sempre più a rischio a causa del numero crescente di industrie, automobili, allevamenti e consumi domestici.
Al di là degli scenari apocalittici vogliamo approfondire in questo articolo alcuni modelli matematici di previsione utilizzati in passato per descrivere le variazioni della numerosità di una popolazione.

Modello di Malthus: la teoria delle catastrofi

Thomas Robert Malthus, economista e demografo inglese (13-febbraio 1766, 29-dicembre 1834) propose nel 1798 un modello matematico dell'evoluzione di una popolazione in presenza di risorse illimitate e in assenza di predatori o antagonisti per l'utilizzo delle risorse.

Sotto queste ipotesi, i fattori di evoluzione sono essenzialmente il tasso di natalità λ e il tasso di mortalità μ che indicano il numero di individui nati e morti per unità di tempo, rispettivamente. Indichiamo con Δt un intervallo di tempo. Siamo interessati a capire con quale velocità varia la popolazione nell'intervallo considerato. Indicando il numero di individui all'istante t con N(t), avremo che dopo un intervallo Δt, cioè all'istante t+Δt, il numero di individui diventerà N(t+Δt).
La velocità di variazione della popolazione (tasso di crescita) sarà data dalla differenza tra la popolazione al tempo t+Δt e quella al tempo t, divisa per l'intervallo temporale Δt, ovvero

$$\frac{\Delta N}{\Delta t}=\frac{N(t+\Delta t)-N(t)}{\Delta t}$$

Questo tasso di crescita sarà quindi legato al tasso di natalità e mortalità:

$$\frac{N(t+\Delta t)-N(t)}{\Delta t}=(\lambda-\mu)N(t)$$

Supponiamo adesso che N(t) vari con continuità il che permette di far tendere Δt a zero. In questo caso

$$\lim_{\Delta t\to 0}\frac{N(t+\Delta t)-N(t)}{\Delta t}=\dot{N}(t)=\epsilon N(t)$$

dove ε=λ-μ è il potenziale biologico della popolazione.
La soluzione dell'equazione che descrive il numero di individui al tempo t è dunque

$$N(t)=N(t_0)e^{\epsilon(t-t_0)}$$

se la popolazione all'istante t_0 è composta da $N(t_0)$ individui. E' evidente che la popolazione cresce esponenzialmente se ε>0, ovvero se i nati superano i morti, in caso contrario la popolazione tende ad estinguersi (Figura 1).

Nelle sue Leçons sur la théorie mathématique de la lutte pour la vie [1] del 1931 lo scienziato italiano Vito Volterra riportava una serie di modelli matematici da lui sviluppati per descrivere alcuni fenomeni osservati in biologia tra i quali il famoso modello di interazione predatore-preda in cui due popolazioni, l'una predatrice e l'altra preda, si trovano a vivere nello stesso ambiente.
Perché non proviamo a simulare il suo modello su un Commodore 128!

La tabella 1 mostra i dati della popolazione degli USA (espressa in milioni di individui) nel periodo 1790-1950 (ad intervallo di dieci) [2]. Vediamo come è possibile scegliere i parametri del modello di Malthus in modo da approssimare i dati reali.

Dalla soluzione dell'equazione di Malthus è semplice ricavare che il potenziale biologico della popolazione è pari a

$$\epsilon = \frac{\ln(N(t)) - \ln(N(t_0))}{t - t_0}$$

Dalla tabella il valore di $N(t_0)$ con t_0=1790 è di 3929; scegliendo t=1800, si ha:

$$\epsilon = \frac{\ln(5308) - \ln(3929)}{10} = 0.0301$$

La Figura 2 mostra l'andamento della popolazione stimata usando il modello di Malthus.Come si evince la legge funziona bene nel periodo 1790-1850, questo spinse Malthus a ritenerla valida sulla base dei dati a disposizione ai suoi tempi. Dopo quegli anni il modello cresce troppo velocemente e i dati previsti si discostano sensibilmente dai dati reali.

Essendo la crescita della popolazione esponenziale secondo Malthus, e in presenza di risorse naturali limitate, il progressivo aumento della popolazione implica che povertà e fame nel mondo sono destinate a crescere sempre più fino a raggiungere esiti catastrofici.

Modello di Verhulst: la crescita logistica

Abbiamo visto come il modello di Malthus produca delle previsioni di crescita della popolazione poco realistiche per periodi di tempo lunghi. In particolare il modello non tiene conto di alcuni meccanismi che determinano variazioni sulle condizioni di vita: risorse limitate, inquinamento prodotto dagli individui, energie spese, aumento della predazione, ecc. Questi meccanismi implicano una trasformazione sulle condizioni di vita in accordo al principio noto come effetto logistico secondo cui ad alte densità, un aumento della popolazione produce una diminuzione di fertilità ed un aumento di mortalità.

Questo si esprime matematicamente considerando i tassi di natalità e mortalità variabili a seconda del numero di individui

$$\lambda(N) = \lambda - \acute{\lambda} N$$

$$\mu(N) = \mu + \acute{\mu} N$$

con $\acute{\lambda}, \acute{\mu} > 0$

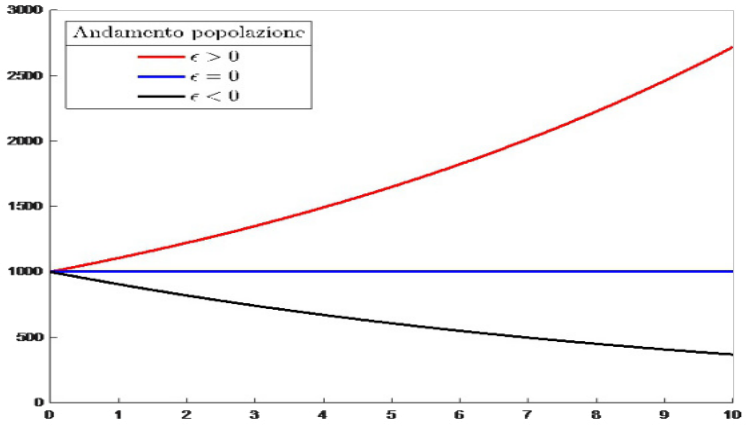


Figura 1. Andamento della popolazione con modello Malthusiano.

Anno	Pop	Anno	Pop	Anno	Pop
1790	3929	1800	5308	1810	7240
1820	9638	1830	12866	1840	17069
1850	23192	1860	31443	1870	38558
1880	50156	1890	62948	1900	75995
1910	91972	1920	105711	1930	122775
1940	131669	1950	150697		

Tabella 1. Dati popolazione USA (in mil.di individui) periodo 1790-1950

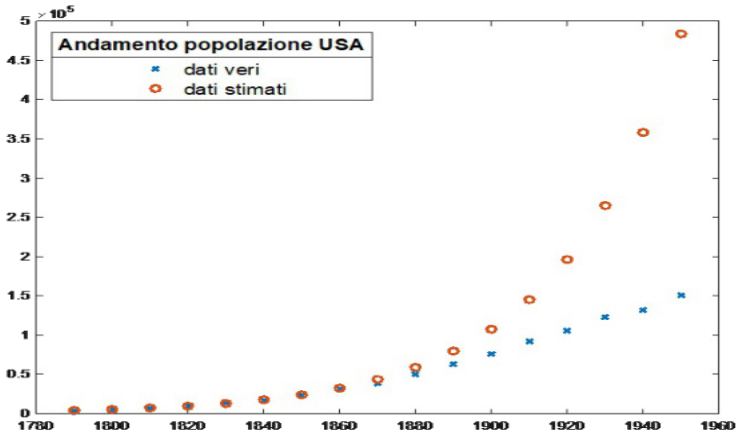


Figura 2. Stima Malthusiana della popolazione USA (1790-1950).

In questo caso l'equazione che descrive l'andamento della popolazione è

$$\dot{N}(t) = \left(\lambda - \mu - \left(\acute{\lambda} + \acute{\mu}\right) N(t)\right) N(t)$$

$$\rightarrow (\lambda - \mu)\left(1 - \frac{\acute{\lambda} + \acute{\mu}}{\lambda - \mu} N(t)\right) N(t)$$

$$\rightarrow \gamma\left(1 - \frac{N(t)}{k}\right) N(t)$$

dove

$$\gamma = \lambda - \mu$$

si definisce **potenziale biologico intrinseco** della popolazione e

$$k=\frac{\lambda-\mu}{\lambda+\mu}$$

rappresenta la capacità portante dell'ambiente. Il modello descritto prende il nome dal matematico e statistico di origine belga Pierre Francois Verhulst (28-ottobre 1804, 15-febbraio 1849) che lo propose nel 1838.
La soluzione dell'equazione e quindi l'andamento della popolazione nel tempo è

$$N(t)=\frac{kN(t_0)}{N(t_0)+(k-N(t_0))e^{-\gamma t}}$$

Nel caso in cui il parametro γ è positivo allora

$$\lim_{t\to\infty} N(t)=k$$

cioè la popolazione tende asintoticamente alla capacità portante dell'ambiente qualunque sia la condizione iniziale $N(t_0)$ (Figura 3). L'equazione logistica rappresenta quindi un modello di crescita esponenziale ma circoscritta; questo significa che la crescita frena perché le risorse dell'ambiente (che hanno favorito appunto tale crescita) sono limitate sia che si tratti di spazio fisico che di alimenti.
E' interessante notare che, nel caso di risorse illimitate ovvero se la capacità portante dell'ambiente tende all'infinito ($k\to\infty$) la crescita logistica converge alla crescita malthusiana

$$\lim_{k\to\infty} N(t)=N(t_0)e^{\gamma t}$$

Modello preda-predatore: Lotka-Volterra

Il modello di Lotka-Volterra prende in considerazione due specie interagenti: predatori e prede. Questo modello fu inspirato da un problema sollevato dal biologo Umberto D'Ancona (genero di Volterra) relativo ad alcune indagini statistiche condotte tra il 1905 e il 1925 su una fluttuazione della percentuale di pesci selaci (pesci predatori) sul totale del pesce pescato nei porti dell'Alto Adriatico.

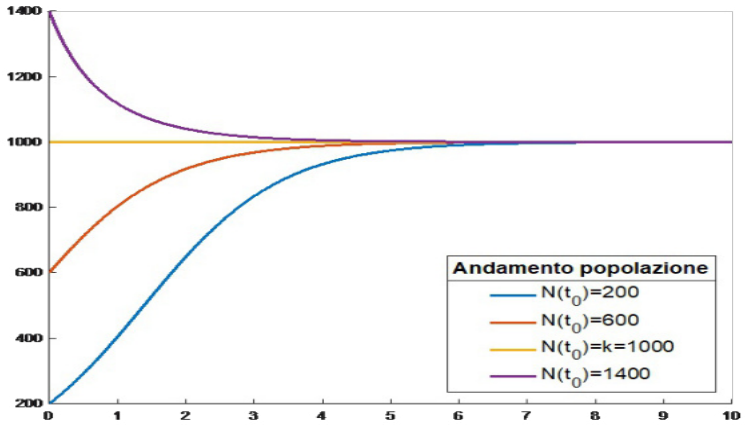


Figura 3. Andamento della popolazione con modello di crescita logistica

D'Ancona aveva ipotizzato che queste fluttuazioni fossero dovute alla diminuzione dell'attività di pesca durante la prima guerra mondiale che aveva ristabilito un nuovo equilibrio naturale in favore dei pesci predatori. La presenza di questa specie tende invece a diminuire tra il 1919 e il 1923 a causa della intensificazione delle attività di pesca.
Volterra risolse il problema generalizzandolo al caso della convivenza fra prede e predatori. Indicando con x(t) la popolazione delle prede e con y(t) quella dei predatori, il modello di evoluzione proposto è il seguente:

$$\dot{x}(t)=(a-by(t))x(t)$$
$$\dot{y}(t)=(cx(t)-d)y(t)$$

con a,b,c,d costanti reali positive. Notiamo come l'andamento delle prede, in assenza dei predatori y(t)=0 per ogni istante di tempo t, è malthusiano

$$\dot{x}(t)=ax(t)$$

Le prede possono infatti riprodursi poiché dispongono del cibo (plancton, ecc.) in quantità praticamente illimitata. Viceversa i predatori, in assenza di prede, non possono che tendere all'estinzione

$$\dot{y}(t)=-dy(t)$$

Gli altri termini presenti nel modello rappresentano i termini di interazione tra le due specie. In particolare la crescita dei predatori sarà proporzionale a quanto cibo è presente nell'ambiente (numero di prede) oltre che al numero dei predatori stessi (cx(t)y(t)). In modo analogo la presenza di predatori provocherà una diminuzione del numero di prede (by(t)x(t)).

Il sistema di Lotka-Volterra di equazioni differenziali, sebbene di due sole equazioni in due incognite e quindi apparentemente molto semplice, non può in realtà essere risolto con metodi analitici. E' possibile però risolvere il problema per via numerica grazie ad algoritmi implementati su calcolatore. Questo è quello che faremo tra un attimo utilizzando un retrocomputer!!!
Per poter implementare il modello descritto è necessario discretizzare le equazioni in modo da far sparire i termini legati alle derivate [3]. A tal proposito possiamo considerare l'approssimazione della derivata:

$$x(t+\Delta t)=x(t)+\dot{x}(t)\Delta t$$

Analogamente per la popolazione dei predatori:

$$y(t+\Delta t)=y(t)+\dot{y}(t)\Delta t$$

Sostituendo avremo che:

$x(t+\Delta t)=\Delta t(a-by(t))x(t)+x(t)$
$y(t+\Delta t)=\Delta t(cx(t)-d)y(t)+y(t)$

Questo sistema si risolve quindi con un algoritmo che calcoli il valore delle funzioni x e y all'istante t+Δt utilizzando i loro valori all'istante precedente t.
Indicando con n e n+1 gli istanti t e t+Δt, avremo che

$$x_{(n+1)}=\Delta t(a-by_n)\,x_n+x_n$$
$$y_{(n+1)}=\Delta t(cx_n-d)\,y_n+y_n$$

Il codice nel riquadro grigio contiene l'implementazione dell'algoritmo su Commodore 128. La simulazione del modello avviene per circa 76 secondi ((3*319-1)*dt). L'andamento delle due specie interagenti sono separate in tre sottografici di durata 25 secondi circa (Figura 4).

Figura 4. Simulazione Lotka-Volterra su C128.

```
10 rem Modello di Lotka-Volterra
20 rem-------------------------------
100 rem parametri
110 a=0.25 : b=0.5
120 c=0.5  : d=0.6
130 rem popolazioni iniziali normalizzate
140 x0=0.5 : y0=0.1
145 rem dt, numero di steps
150 dt=0.08 : NN=319*3
160 dim x(NN), y(NN)
180 x(1)=x0 : y(1)=y0
190 for n=1 to NN-1
200 : x(n+1)=dt*(a-b*y(n))*x(n)+x(n)
210 : y(n+1)=dt*(c*x(n)-d)*y(n)+y(n)
220 next n
1000 rem disegna popolazioni
1010 graphic 1,1
1012 char 1,5,1,"modello di lotka-volterra:"
1013 char 1,5,2,"a="+str$(a)+" b="+str$(b)+" c="+str$(c)+" d="+str$(d)
1014 d=60
1015 draw 1,0,d to 319,d
1018 draw 1,0,2*d to 319,2*d
1019 draw 1,0,3*d to 319,3*d
1020 for k=1 to 3
1025 : for n=1 to 319
1035 :   draw 1,n,d*k-x((k-1)*319+n)*10
1042 :   draw 1,n,d*k-y((k-1)*319+n)*10
1046 : next n
1054 next k
1057 getkey a$
1060 graphic 0
```

Bibliografia

[1] V. Volterra, Leçons sur la théorie mathématique de la lutte pour la vie. Bull. Amer. Math. Soc 42 (1936): 304-305.

[2] G. Israel, Modelli Matematici: Introduzione alla matematica applicata, Franco Muzzio Ed., 2002.

[3] E. Salinelli, F. Tomarelli, Modelli Dinamici Discreti. Springer, 2014.

Grafica...che passione!
Parte III, hi-res multicolor mode con FLI

di Marco Pistorio

Ed eccoci giunti a questa terza parte dell'articolo "Grafica..che passione!". Anche stavolta ci occuperemo di convertire immagini a colori in schermate in alta risoluzione "multicolor mode" del C64, ma con l'ausilio di una tecnica denominata FLI.

Come funziona? Cercherò di spiegarlo nella maniera più semplice possibile, lasciando come sempre ai più volenterosi l'onere di approfondire la questione per conto proprio.
Vi anticipo intanto che per visualizzare schermate FLI non sarà possibile impiegare il BASIC. Perché? Perché per far funzionare questa tecnica serve agire a tempo di raster, e ciò non è possibile farlo via BASIC.
Cerchiamo ora di comprendere come funziona FLI. Abbiamo già visto come nel Commodore 64 vengono rappresentate le schermate grafiche in alta risoluzione ed in particolare quelle a colori, nelle due precedenti parti di questo articolo, pubblicate nei numeri scorsi di "RetroMagazine".
Abbiamo visto che la trama dell'immagine bitmap viene interpretata a coppie di bit, ed in base a tali coppie viene attributo a ciascun pixel dell'immagine uno dei 4 colori disponibili per ciascuna cella di 4x8 pixel, ottenendo una immagine con risoluzione 160x200 pixel in questa modalità hires multicolor mode.
Ad ogni coppia di bit "10" e "01" viene assegnato, come già illustrato nella parte II di questo articolo, un colore che è ottenuto attraverso la parte alta e bassa del relativo byte che si trova nella memoria di schermo, ed ogni cella di 4x8 pixel è legata ad uno di questi byte. Per ogni schermata video hires servirà un'area di 1000 di questi bytes per gestire questi due attributi di colore, che abbiamo chiamato in precedenza Colore#1 e Colore#2 . Ne consegue che per ciascuna cella di 4x8 pixel sono disponibili due diversi colori, più un terzo che è il cosiddetto Colore base, che viene assegnato alla coppia di bit "11" ed ottenuto mediante l'uso di un ulteriore byte che, tutti insieme, diventano altri 1000 bytes della memoria cosiddetta "colore" ed infine un quarto colore, assegnato alla coppia "00" che è il colore di sfondo, comune ovviamente per tutte le cellette di 4x8 pixel che compongono l'immagine che visualizziamo sullo schermo ed il valore di questo colore è memorizzato nella apposita locazione di memoria $D021.
Come intervenire in questo meccanismo, peraltro già illustrato nella seconda parte di questo articolo, per ottenere più colori a disposizione in ogni cella?

Qui entra in scena FLI (acronimo di Flexible Line Interpretation) che ragiona in questa maniera.
Invece di ottenere le informazioni di colore per ciascuna linea di carattere, come avviene normalmente, le otteniamo per ciascuna linea di pixel, scrivendo opportunamente in una particolare locazione di memoria, ovvero la locazione $D011, nota anche come screen control register #1, nell'esatto momento in cui la linea di pixels è stata visualizzata sul nostro schermo.
Immediatamente prima però, cambiamo il contenuto del puntatore all'area di memoria di schermo video debitamente.
A questo punto il VIC-II leggerà nuovamente le informazioni relative al colore, informazioni che potranno essere anche diverse da quelle lette ed impiegate per la per la linea di pixel già sullo schermo.
Sarà possibile rifare questo "trucchetto" per ciascuna linea di pixel a schermo.

Troverete, insieme al tool che esamina una immagine e genera tutte queste informazioni per voi (memoria colore, 8 memorie di schermo e mappa hi-res) anche una routine assembly che metterà insieme tutte quesle informazioni e che vi mosterà sul vostro C64 l'immagine risultante, e potrete valutare da voi i vantaggi di questa tecnica "FLI" rispetto a quanto si possa ottenere normalmente in hires multicolor mode con il c64

Svantaggi di questa tecnica? Purtroppo ne esistono, non posso negarlo ahinoi.
Innanzitutto, appena si lavora sulla locazione $D011 cercando di ingannare il VIC-II costringendolo a rileggere nuovamente le informazioni colore, si manifesta il cosiddetto "FLI-Bug" nella parte sinistra dello schermo. Una modesta porzione di questa area di schermo si colorerà, indipendentemente dalle coppie di bit da rappresentare lì, riducendo di fatto l'area di schermo video impiegabile, area che ho "limitato" nel tool forzando un colore nero fisso in corrispondenza di questa area comunque non utilizzabile. Un secondo svantaggio è l'occupazione di memoria, che risulta maggiore in base al fatto che per descrivere i nuovi colori è necessario impegnare nuove aree di memoria schermo.
Infine, l'immagine deve essere "spennellata" continuamente, a tempo di raster, leggendo ed impostando linea per linea i giusti colori, e questo comporta ovviamente un costo in termini di elaborazione, costo che non si avrebbe visualizzando una immagine hires in multicolor mode tradizionale.

Parliamo ora dei vantaggi, che ci sono naturalmente!
Osservate le due immagini di una

stella stilizzata che si trovano nella pagina successiva. Noterete subito che la prima risulta diversa, più "degradata" rispetto alla seconda. Il motivo è presto detto. L'immagine contiene diverse linee con colori diversi ravvicinate. Nella normale modalità hires multicolor mode abbiamo la possibilità di impostare fino a 4 colori per ciascuna cella di 4x8 pixel, uno dei quali è comune per tutte le celle, come già sapete.
Quindi, in tutte quelle celle 4x8 dove ci sono più di 4 colori da impostare non è possibile colorare tutti questi pixel ma soltanto una parte, mentre i restanti vengono impostati al colore di fondo, il nero. Il risultato è quindi modesto.
Nella foto successiva abbiamo invece la stella ottenuta mediante FLI. Adesso il computo dei colori funziona così. Un colore è quello di fondo, comune a tutte le celle 4x8, uno è quello ottenuto dalla memoria colore, che deve essere lo stesso per ciascuna cella 4x8 e poi abbiamo Colore#1 e Colore #2, che possono essere scelti tra uno qualsiasi dei 16 colori della tavolozza del C64 e possono essere diversi riga per riga, per ogni cella 4x8. Ciò implica che si hanno a disposizione molti più colori dei classici 4 per cella! Ecco il motivo per cui la stella riprodotta nella seconda figura risulta più gradevole graficamente, e con colori più completi. Ovviamente l'immagine di esempio non è stata scelta a caso ma costruita appositamente sottolineare le differenze tra la modalità hires multicolor mode classica e quella FLI.Qualcuno considera questa modalità FLI una modalità "illegale" perché non ufficiale, perché frutto dello sfruttamento di un "bug" del VIC-II. Tuttavia, apprendere della esistenza di un bug e superarlo conseguendone un vantaggio non mi sembra una cosa malvagia, anzi. Solo grazie allo studio attento del VIC-II è stato possibile ottenere FLI e diverse altre modalità "illegali" che sottolineano sia la qualità dell'hardware del C64 che era evidentemente possibile sfruttare ulteriormente, e la bravura di coloro che hanno scovato tali difetti e li hanno superati in maniera proficua. C'è addirittura chi riesce a "camuffare" talmente bene l'area del "FLI-Bug" (area che io lascio vuota) che non ci si accorge neanche della esistenza di tale area. Più bravi di così :)

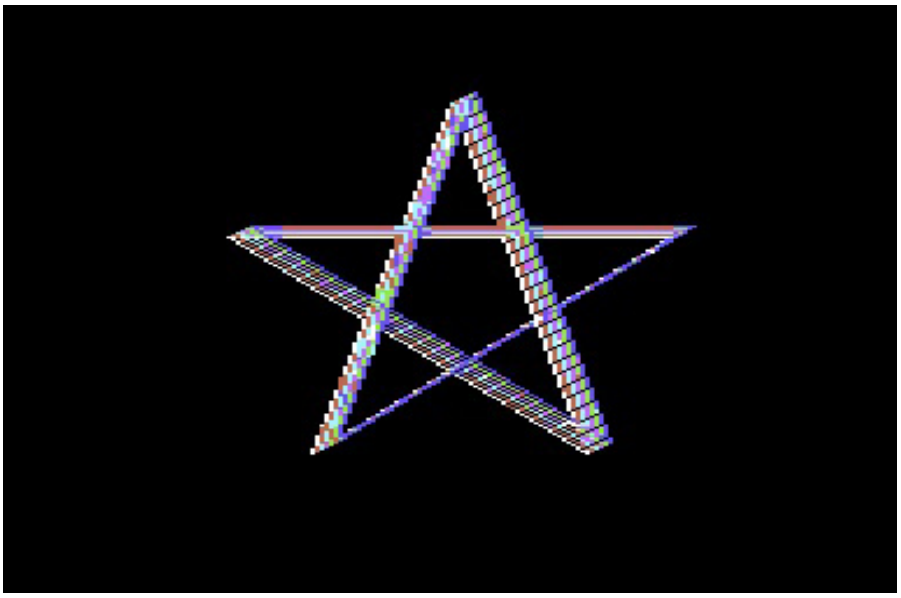


Figura 1 - Stella (NO FLI)

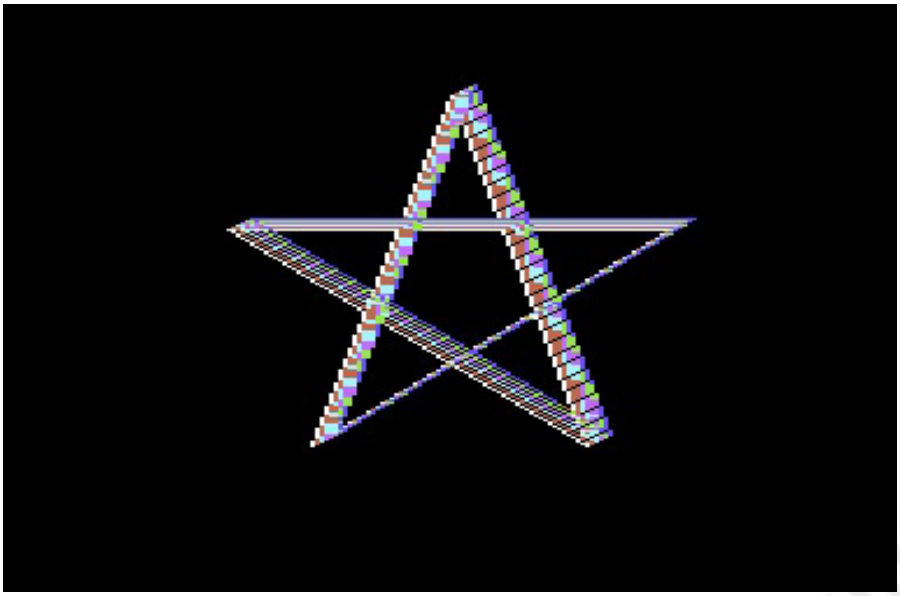


Figura 2 - Stella (FLI)

- ***Funzionamento del tool***

Il tool è molto simile a quelli già visti, sebbene sia stato modificato per gestire la complessità della analisi dei colori secondo la logica FLI. Il tool inizia a scansire l'immagine da convertire, che dovrà chiamarsi sempre "immagine.bmp" e dovrà trovarsi nella stessa cartella contenente l'eseguibile del tool. Per gli stessi motivi descritti nelle due precedenti parti di questo articolo non sono presenti meccanismi di controllo nè della presenza né della validità della immagine che viene elaborata.Il codice è abbastanza analogo a quello già presentato, in particolare nella parte precedente di questo articolo, presente nello scorso numero di "RetroMagazine" quindi eviterò di commentare analiticamente il funzionamento e lo scopo delle varie routines all'interno del tool, rimandandovi alla lettura del precedente articolo per uno studio aporofondito del codice dello stesso.

A ciascun pixel verrà associato uno dei 16 colori presenti nella palette del C64.

- ***Uso dei files generati dal tool***

Come adoperare i files generati dal tool? Purtroppo, a fronte delle peculiarità di FLI non è possibile la riproduzione dell'immagine tramite BASIC alternativa alla routine in assembly, ma resta valida solo quest'ultima opzione.
Di seguito quindi riporto il codice assembly necessario. Allo scopo ho sfruttato il codice che ho trovato a questo indirizzo web:
http://codebase64.org/doku.php?id=base:fli_displayer
che ho appena modificato per adattarlo all'impiego dei files prodotti dal tool e che può essere compilato direttamente mediante KickAssembler, e che troverete pubblicato su GITHUB, oltrechè pubblicato di seguito.

- ***Conclusioni finali***

Per lanciare il tool basterà fare doppio click sull'eseguibile "tool_bitmaps_4_c64_multicolors_FLI.exe".
Se è tutto ok, pochi secondi dopo l'avvio del tool comparirà, all'interno di una piccola finestra, il contenuto della immagine bitmap elaborata, pronta per essere data

```
//////////////////////////////////////////////////
// Routine assembly con sintassi KickAssembly
// per la visualizzazione di immagini hires
// multicolor mode + FLI
//////////////////////////////////////////////////

.label tab18   = $0e00
.label tab11   = $0f00

        *= $1000

:BasicUpstart2(start)

        jmp start
irq0:   pha
                lda $d019
        sta $d019
        inc $d012
        lda #<irq1
        sta $fffe      // set up 2nd IRQ to get a stable
IRQ
        cli

       // Following here: A bunch of NOPs which allow
the 2nd IRQ
       // to be triggered with either 0 or 1 clock
cycle delay
       // resulting in an "almost" stable IRQ.

        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
irq1:
ntsc1:  lda #$ea       // modified to NOP NOP on NTSC
        lda #$08
        sta $d018      // setup first color RAM address
early
        lda #$38
        sta $d011      // setup first DMA access early
        pla
        pla
        pla
       lda $d019
        sta $d019
        lda #$2d
        sta $d012
        lda #<irq0
        sta $fffe      // switch IRQ back to first
stabilizer IRQ
        lda $d012
        cmp $d012      // stabilize last jittering cycle
        beq delay      // if equal, 2 cycles delay. else
3 cycles delay
delay:
       stx savex+1

        ldx #$0d
wait:   dex
        bne wait

ntsc2:  lda #$ea       // modified to NOP NOP on NTSC
ntsc3:  lda #$ea       // modified to NOP NOP on NTSC

       // Following here is the main FLI loop which
forces the VIC-II to read
       // new color data each rasterline. The loop is
exactly 23 clock cycles
       // long so together with 40 cycles of color DMA
this will result in
       // 63 clock cycles which is exactly the length
of a PAL C64 rasterline.

       nop
       nop
l0:
        lda tab18+1,x
        sta $d018      // set new color RAM address
        lda tab11+1,x
        sta $d011      // force new color DMA
        inx            // FLI bug $D800 color = 8
(orange)
        cpx #199       // last rasterline?
ntsc4:  bne l0         // branches to l0-1 on NTSC for 2
extra cycles per rasterline

       lda #$70
       sta $d011      // open upper/lower border

       //lda $d016
       //eor #$01       // IFLI: 1 hires pixel shift
every 2nd frame
       //sta $d016
       //lda $dd00
       //eor #$02       // IFLI: flip between banks
$4000 and $C000 every frame
       //sta $dd00

savex:  ldx #$00
       pla
nmi:   rti
```

```
start:
        sei
        lda #$35
        sta $01         // disable all ROMs
        lda #$7f
        sta $dc0d       // no timer IRQs
        lda $dc0d       // clear timer IRQ flags

        lda #$2b
        sta $d011
        lda #$2d
        sta $d012

        lda #<nmi
        sta $fffa
        lda #>nmi
        sta $fffb       // dummy NMI to avoid crashing
due to RESTORE
        lda #<irq0
        sta $fffe
        lda #>irq0
        sta $ffff
        lda #$01
        sta $d01a       // enable raster IRQs

        jsr initgfx
        jsr inittables
        jsr ntscfix

        lda $d019
        dec $d019       // clear raster IRQ flag
        cli


        ldx #00
        lda #$00
ss:
        sta $3C00,x
        dex
        bne ss

        jmp *           // that's it, no more action
needed

initgfx:
        lda #$00
        sta $d015       // disable sprites
        sta $d020       // border color black

                lda #$00        // background color
        sta $d021

                lda #$ff
                sta $7fff       // upper/lower border
black

        lda #$18
        sta $d016
        lda #$08
        sta $d018
        lda #$96        // VIC bank $4000-$7FFF
        sta $dd00

        ldy #$04
        ldx #$00
l1:     lda fli,x
        sta $d800,x     // copy color RAM data
        inx
        bne l1
        inc l1+2
        inc l1+5
        dey
        bne l1
        rts
```

```
inittables:
        ldx #$00
l2:     txa
        asl
        asl
        asl
        asl
        and #$70        // color RAMs at $4000
        ora #$08        // bitmap data at $6000
        sta tab18,x     // calculate $D018 table
        txa
        and #$07
        ora #$38        // bitmap
        sta tab11,x     // calculate $D011 table
        inx
        bne l2
        rts

ntscfix:
        bit $d011
        bmi *-3
        bit $d011       // wait for rasterline 256
        bpl *-3
        lda #$00
test:   cmp $d012
        bcs nt
        lda $d012       // get rasterline low byte
nt:     bit $d011
        bmi test
        cmp #$20        // PAL: $37, NTSC: $05 or $06
        bcs pal

        lda #$ea
        sta ntsc1
        sta ntsc2
        sta ntsc3
        dec ntsc4+1
pal:    rts

       // link a demo picture
        *= $3c00-2
fli:

.pc=$3C00+$0400*0
.import c64 "colors.dat"

.pc=$3C00+$0400*1
.import c64 "screen0.dat"

.pc=$3C00+$0400*2
.import c64 "screen1.dat"

.pc=$3C00+$0400*3
.import c64 "screen2.dat"

.pc=$3C00+$0400*4
.import c64 "screen3.dat"

.pc=$3C00+$0400*5
.import c64 "screen4.dat"

.pc=$3C00+$0400*6
.import c64 "screen5.dat"

.pc=$3C00+$0400*7
.import c64 "screen6.dat"

.pc=$3C00+$0400*8
.import c64 "screen7.dat"

.pc=$6000
.import c64 "hires.dat"
```

in pasto al nostro fido "biscottone".
Nel caso in cui invece la finestra comparisse vuota, molto probabilmente il file .bmp fornito non è stato trovato oppure non contiene le corrette informazioni colore.
Anche stavolta ho predisposto un file bitmap di prova, che potrete elaborare.
Tale file sarà a vostra disposizione insieme al tool con i sorgenti ed insieme ad ulteriore materiale sul repository GITHUB all'indirizzo che segue:

https://github.com/marcus73/retromagazine_05

Il file da convertire dovrà chiamarsi, come già specificato, immagine.bmp e verranno prodotti dal tool ben 10 files, che si chiameranno rispettivamente: hires.dat, colors.dat, screen0.dat, screen1.dat, screen2.dat, screen3.dat, screen4.dat, screen5.dat, screen6.dat e screen7.dat che verranno sovrascritti di volta in volta se già presenti.

Vale anche qui la considerazione fatta nelle precedenti due parti di questo articolo.
Il codice presentato NON INTENDE essere un modello di perfetta programmazione. Vuole essere uno spunto, un punto di partenza per chi di voi intenda approfondire tali argomenti.
I 10 files .dat potranno essere sfruttati in assembly secondo quanto già esposto.

Spero che questa "carrellata" di tools, che spaziano su diverse modalità video del C64 siano stati di vostro gradimento ed abbiano stuzzicato in voi l'intenzione di approfondire e far "vostri" tutti i vari concetti illustrati. Non mi resta che salutarvi e ringraziarvi per aver letto anche quest'ultima parte degli articoli "Grafica, che passione!". Fateci sapere se vi sono piaciuti, mi raccomando. Non è escluso che, anche in funzione di vostre segnalazioni, si possa tornare ad approfondire qualcuno degli argomenti trattati, aggiungendo altra "carne al fuoco", perché no?

Ciao a tutti, amici lettori!

Un altro esempio di schermata FLI molto colorata!

ROC'N ROPE

Publisher: Konami
Anno: 1983
Piattaforma: arcade
Genere: Platform

Nei mitici anni 80 noi appassionati di videogames potevamo sfogare le nostre passione sia nelle sale giochi e sia nei bar di quartiere che avevano spesso un angolo o un'intera sala dedicata a flipper e cabinati vari.
Proprio in uno di questi bar, mentre mio padre consumava la sua birra chiacchierando con gli amici, conobbi in una sera d'estate Roc'n Rope, che ancora oggi rimane uno dei mie titoli preferiti.
Prodotto dalla mitica Konami nel 1983, Roc'n Rope è un platform classico a schermata fissa, stretto parente di Mario Bros e Donkey Kong.
In questo titolo vestiamo i panni di un simpatico esploratore che deve scalare la vetta di una montagna per cercare di raccogliere le piume di una mitica fenice d'oro.
A nostra disposizione abbiamo un arpione con corda che usiamo per salire i vari livelli delle montagne e una piccola torcia che può bloccare i nostri nemici per alcuni secondi quando ci sono troppo vicini.
A proposito dei nemici, in questo gioco dovremmo vedercela contro uomini cavernicoli e dinosauri che sono pronti ad inseguirci anche sulla corda del nostro arpione, mentre terribili uccelli preistorici, di tanto in tanto, ci lanciano contro enormi massi.
Lungo il cammino possiamo raccogliere piume d'orate perse dalla fenice che aumenteranno il nostro score e la classica pillola in stile pacman che ci permette, per pochi istanti, di divorare i nostri avversari.
I livelli sono soltanto quattro: una montagna, un castello di ghiaccio, una caverna piena di passagi mobili e ancora una montagna con una cascata al centro pronta a travolgerci. Per questo motivo, con un pò di pratica, Roc'n Rope non è impossibile da terminare, ma questo non deve farci pensare che siamo davanti ad un titolo minore.
Anzi a mio avviso il gioco è geniale nel suo gameplay ed è sempre piacevole, appena si ha un pò di tempo libero, fare una partita mordi e fuggi.

di Querino Ialongo

GIUDIZIO FINALE

» Giocabilità 90%

Roc'n Rope ha un ottimo gameplay ed è davvero piacevole muovere il nostro esploratore tra piattaforme ed incredibili arrampicate

» Longevità 80%

Nonostante questo titolo abbia più di 35 anni ed è costituito di soli quatro stages io lo trovo davvero piacevole da giocare ancora oggi. Classico gioco per partite "mordi e fuggi" dal divertimento assicurato.

RAMBO FIRST BLOOD PART II

Publisher: Ocean
Anno: 1984
Piattaforma: Commodore 64
Genere: Sparatutto

Il nostro John Rambo si trova collocato in un ambiente tropicale. Scelta da manuale :)

Dopo anni o meglio dopo un intera generazione passata ormai, torno ad occuparmi di un genere di videogiochi in cui durante la mia infanzia ero un vero e proprio specialista in quanto la mia sala giochi casalinga, ossia i tie-in.

Ho usato questo termine tecnico che, in confidenza ho scoperto anch'io da poco, visto che durante la mia infanzia badavo poco ai termini, manuali e roba da informatici fatti e finiti; I tie in sono precisamente i titoli dei giochi tratti dai film ed all'epoca ne fecero parecchi per computer e console, visti i successoni di molti film anni 80-90, così ho deciso di ripescare un bel gioco (tratto ovviamente da un bel film) di quegli anni di un eroe ed attore sempreverde: Rambo First Blood part II.

I giochi tratti dai film di rambo uscirono per parecchi computer e console, tutt'ora stanno uscendo per le console odierne (e come dargli torto?) Però questo titolo l'ho trovato singolare in quanto l'ho scoperto ultimamente e lo avevo visto solo di sfuggita negli anni d'oro di Commodore.

Partiamo subito dall'inizio: Il nostro protagonista, stimato ed amato John Rambo, si trova in un campo aperto con una lieve vegetazione ed un ampio campo di prigionia, armato con pugnale, frecce, fucile che recupererà strada facendo e le sue migliori amiche frecce infuocate, proprio come nel film.

Egli dovrà farsi strada fino al cuore del campo di prigionia, liberando i prigionieri e raggiungendo l'elicottero al campo base più avanti, dopodichè sarà diretto verso un altro campo di prigionia con il medesimo compito.

Per quanto risulti breve il gioco, l'ho trovato intenso e coinvolgente dal punto di vista musicale, già lo scorso numero avevo citato il fatto che Commodore fosse un vero Juke box di splendide musiche e mi meraviglio del fatto che non sia mai uscita una compilation musicale dedicata ad esso.

Qui la mia memoria va ad un altro famoso sparatutto per C64, "Commando", lo ricordate?

Durante la prima partita i nemici potranno risultare ben addestrati e numerosi soprattutto nel primo livello ed anche i più braci potrebbero perdere la partita per pura sfortuna dato che abbiamo un'unica vita con una barra energetica che non tarda molto ad esaurirsi, dopotutto Rambo è pur sempre un essere umano con dei limiti, quindi per arrivare al primo casolare contenente i prigionieri, consiglio di aggirare il perimetro del campo per poi trovare subito l'entrata, sparare e sparare, soprattutto alle piccole basi per poi sparare a quella che farà uscire i prigionieri.
Dopo sarete liberi di andare avanti tra la folta vegetazione fino a raggiungere l'elicottero nel quale ci sarà una breve sequenza giocabile con esso nel secondo livello fino a raggiungere l'atterraggio del secondo campo.

Il terzo ed ultimo livello sarà una battaglia aerea in elicottero che vedrà impegnati voi ed un elicottero nemico che farà di tutto per non farvi arrivare in fondo al livello dove raggiungerete l'atterraggio.

Il combattimento finale contro l'elicottero l'ho trovato parecchio rognoso e mi ci è voluto un po' prima di abbatterlo (per poi vederlo resuscitato!) ed andare avantifino alla fine. Il mio consiglio è quello di fare un giro intorno ad esso, evitare i suoi micidiali colpi e poi sparargli per poi proseguire oltre, così lo potrete seminare per un po' di tempo accorciando le distanze, infine abbatterlo definitivamente. Se non lo avete ancora fatto (ne dubito) Guardate tutti i film di Rambo e giocate a questo titolo, ne vale la pena questa maratona che ho fatto io anche se la nostalgia ahimè è sempre dietro l'angolo e credo che ormai faccia parte di noi retrogamer.

Termino questo articolo salutandovi tutti ed augurandovi un buon inizio estate e buon divertimento con i nostri intramontabili titoloni!

di Daniele Brahimi

RIFERIMENTI WEB

https://it.wikipedia.org/wiki/Rambo:_First_Blood_Part_II_(videogioco_1985)

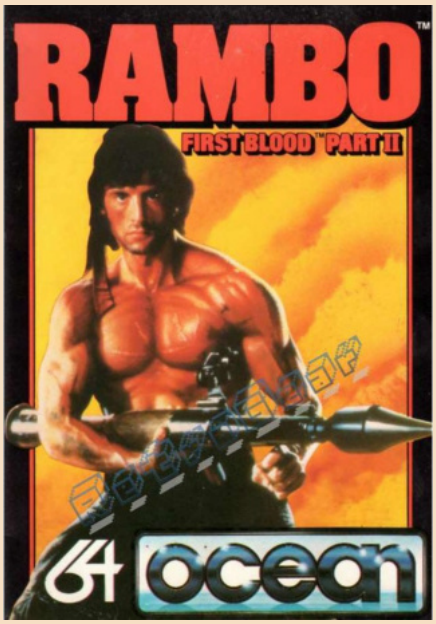
GIUDIZIO FINALE

» Giocabilità 85%

E' abbastanza semplice capire come procedere all'interno del gioco. Interfaccia utente intuitiva e personaggio. Per gli amanti del genere sparatutto è un buono titolo!

» Longevità 65%

Dopo aver completato il gioco, i veri "duri" lo continueranno a giocare con piacere.



PRINCE OF PERSIA

Publisher: Broderbund
Anno: 1990
Piattaforma: Amiga
Genere: Platform

Prince of Persia è una vera e propria pietra miliare nel mondo dei videogiochi platform, tanto che sfido a trovare tra i nostri lettori qualcuno che non ne abbia mai sentito parlare.

La trama è piuttosto semplice; approfittando dell'assenza del sultano, il malvagio visir Jaffar che mira al trono del regno, rapisce la principessa e le impone un ultimatum: ha un ora di tempo per acconsentire a sposarlo, dopodichè verrà uccisa... Forse un po' severo, ma dal suo punto di vista sicuramente coerente!

Il giocatore impersona l'eroe di turno, il vero amore della principessa, che dovra' arrivare fino alla stanza dove è rinchiusa la sua amata per cercare di salvarla. Ovviamente, per raggiungere la prigione dorata della fanciulla, dovra' attraversare tutto il palazzo di Jaffar che, in fin dei conti, è un vero e proprio labirinto disseminato di trappole e controllato da pericolose guardie.

Passata la scarna presentazione, peraltro ben disegnata e colorata, si entra subito nel vivo dell'azione. Dal punto di vista grafico il gioco si presenta piuttosto bene, con una grafica sobria e funzionale, ma che non fa certo gridare al miracolo, che ci accompagna all'interno del labirintico palazzo di Jaffar. La grafica quindi, pur essendo ben disegnata ed in linea con gli standard del tempo, non e' il punto di forza di Prince of Persia. Non appena proveremo a muovere il nostro personaggio invece, la sua animazione ci lascera' sicuramente a bocca aperta.

Il nostro eroe infatti si muove, corre, salta, si abbassa e penzola nel vuoto con una fluidità ed una leggerezza di movimenti come non si era ancora avuto modo di vedere prima dell'uscita di questo titolo. Il gioco che forse si avvicinava ad una simile fluidità di movimenti dello sprite principale era Impossible Mission, ma qui, credetemi siamo veramente ad un altro livello. Si racconta infatti che Jordan Mechner, l'autore principale

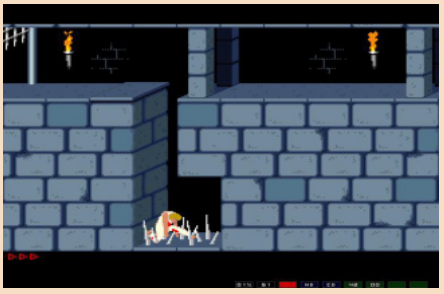
Decisione sofferta

La principessa ha un ora di tempo per decidere se sposare il perfido Jaffar o morire...

Comincia tutto da qui

Il palazzo di Jaffar è un vero e proprio labirinto. Salvare la principessa non sarà facile.

Morire non è affatto difficile



Vedere morire il nostro alter ego virtuale non farà felice nessuno, ma un po' di sangue virtuale male non fa...

del gioco, sia stato a lavorare per giorni all'animazione dell'eroe, studiando i movimenti di suo fratello che, andava combinato più o meno come la versione povera del principe, con una tuta dai colori simili a quelli degli sprite ed una "mappina" (in napoletano, uno straccio) sulle spalle per simulare il mantello di alcuni personaggi. Conciato in siffatto modo cercava di rompersi le corna saltando ostacoli e scavalcando muretti, mentre il fratello riprendeva il tutto. In giro ci sono frammenti video di queste prodezze, molto comiche per la verità! Ringrazio Giorgio Balestrieri per la citazione.

Il sonoro invece merita un discorso a parte. Non ci sono musiche che accompagnano l'azione, ma gli effetti sonori sono all'altezza della situazione. La scelta di non mettere una colonna sonora, a mio modesto avviso è azzeccata, in quanto spesso e volentieri il sottofondo musicale rischia alla lunga di diventare monotono e talvolta infastidire il giocatore invece di deliziarlo.

Gli effetti sonori di Prince of Persia invece sono sì essenziali e minimalisti, ma sicuramente azzeccati ed utili per il proseguimento del gioco. Basti pensare infatti al rumore delle porte che, fuori dalla portata visiva del giocatore, si aprono e si chiudono quando passiamo sopra una piattaforma mobile. Il rumore minaccioso delle lame che fuoriescono dal pavimento invece e' un vero e proprio tocco di classe, per non parlare del rumore attutito dei passi del nostro eroe... Tutti elementi che, se ben combinati come in Prince of Persia, contribuiscono a calare il giocatore nell'atmosfera claustrofobica del palazzo di Jaffar. Proprio come era nelle intenzioni dei programmatori!

La giocabilità è ai massimi livelli, in pratica ci troviamo di fronte ad un antesignano di Tomb Raider in 2 dimensioni ma con un limite di tempo realistico. Il gioco infatti richiede di portare a termine la missione e liberare la principessa in un ora di tempo reale... Nel caso non riuscissimo a terminare il gioco nell'intervallo concesso, dovremo ricominciare tutto da capo. Attenzione quindi ai trabocchetti ed alle guardie perche ogni volta che verremo uccisi perderemo del tempo prezioso; tempo che la nostra principessa non ha…

Per quanto riguarda la difficoltà, il gioco è ben calibrato. Il primo livello e' semplice e permette una rapida curva di apprendimento delle dinamiche di gioco, ma proseguendo l'avventura, gli enigmi si faranno sempre più complicati, le trappole piu' insidiose e le guardie piu' agguerrite. In poche parole il gioco è coinvolgente sin dall'inizio al punto da tenere il giocatore incollato allo schermo almeno fino a quando non avrà terminato il suo compito e liberato la principessa prigioniera.

Il tallone d'Achille di questo titolo e' probabilmente la longevita'. E' vero infatti che non smetterete di giocarlo sino a quando non lo avrete terminato, ma una volta presa la mano riuscirete a finirlo in breve tempo. A questo punto, difficilmente lo rigiocherete se non dopo qualche anno. Dalla sua uscita ne sono passati quasi trenta, sara' l'occasione buona per rigiocarlo? :-)

di Francesco Fiorentini

Anche Adriano Avecone ha postato nel suo gruppo Arcadian una recensione di Prince of Persia con diverse informazioni aggiuntive.

Sviluppatore: Jordan Mechner. Publisher: Brøderbund. Anno: 1989-1990. Musiche: Francis Mechner (musica), Tom Rettig (effetti sonori), Mark Cooksey (NES), Tommy Tallarico (Game Boy).
Piattaforme: NEC PC-9801, MS-DOS, Amiga, Atari ST, Sharp X68000, Amstrad CPC, SAM Coupé, PC Engine, Gameboy, FM Towns, Master System, SNES, Sega CD, NES, Apple Macintosh, Game Gear, Megadrive, Gameboy Color, iOS, Nintendo 3DS, Wii. Non ufficiali: Enterprise 128, Electronika BK-0011M, ATM Turbo, ZX Spectrum, HP48/GX, TI-89, TI-92, Commodore Plus/4 (Demo), Commodore 64, Linux, Microsoft Windows, Roku (Streaming Box and Smart TV), BBC Master.

Prince of Persia è un videogioco di genere platform/azione del 1989 pubblicato dalla Brøderbund e sviluppato da Jordan Mechner inizialmente per Apple II e in seguito convertito per moltissimi sistemi.

In Prince of Persia dovremo controllare un protagonista senza nome in una serie di dungeon con l'obiettivo di sconfiggere il Grand Visir Jaffar e salvare la principessa da questi imprigionata. Proprio come in Karateka, il primo gioco di Mechner, Prince of Persia utilizza una tecnica di animazione in rotoscoping per ricreare movimenti fluidi e realistici. Per questo gioco è stato utilizzato un video in cui il fratello dello sviluppatore eseguiva una serie di acrobazie, oltre a e film come The Adventures of Robin Hood. Il gioco ha ottenuto un eccellente riscontro a livello critico ma vendite non adeguate alla qualità del prodotto, che ha comunque riscosso un buon successo grazie all'elevato numero delle conversioni realizzate. Prince of Persia ha ispirato numerosi giochi, da Another World fino a Tomb Raider.
Il gioco ha ottenuto due sequel, Prince of Persia 2: The Shadow and the Flame e Prince of Persia 3D, e due reboot, Prince of Persia: The Sands of Time (2003), a sua volta oggetto di tre sequel, e il nuovo Prince of Persia del 2008.

Il gioco è ambientato nell'antica Persia: mentre il sultano è all'estero per portare avanti una

campagna bellica, il suo visir e mago Jaffar usurpa il potere del suo sovrano. L'unico ostacolo per la conquista del trono è la figlia del sultano, rinchiusa da Jaffar in una torre per obbligarla a diventare sua moglie. In caso di rifiuto, la ragazza verrà uccisa dopo 60 minuti (120 nella versione SNES). Il protagonista, amato dalla principessa, viene rinchiuso nelle segrete del palazzo, da cui dovrà fuggire in modo da liberare la donzella e sconfiggere Jaffar. Oltre alle guardie, trappole e segrete presenti nelle prigioni, dovremo anche affrontare una copia del protagonista evocata da uno specchio magico. Il gioco è costituito da dodici livelli, anche se alcune versioni per console dispongono di un maggior numero di quadri.

Lo sviluppo del gioco è iniziato nel 1985, anno in cui Mechner si è laureato alla Yale University. Lo sviluppatore aveva già realizzato Karateka per la Brøderbund. Tale publisher aveva chiesto al creativo di realizzare un sequel del suo primo gioco, ma non contrastò il desiderio di Mechner di creare un gioco originale. Per creare il sistema di combattimento con la spada, Mechner ha usato il duello finale tra Errol Flynn e Basil Rathbone del film Adventures of Robin Hood. Anche se all'epoca il rotoscoping veniva considerato una tecnica pionieristica, Mechner scelse di adottare questo metodo per colmare le proprie lacune in termini di disegno e animazione.

Un altro importante elemento di originalità del gioco era la possibilità di combattere con le spade e non con armi da fuoco o lotta fisica, come spesso avveniva nei videogiochi del periodo. La Arsys Software e la Riverhillsoft hanno ottimizzato la grafica e l'aspetto del principe nel realizzare le versioni per console e home compute, aggiungendo il classico turbante e il gilet, che furono ripresi anche dalla versione Macintosh. La versione FM Towns, anch'essa realizzata dalla Riverhillsoft, aggiunge una colonna sonora in formato CD audio. La versione Gameboy è stato il primo gioco a utilizzare le musiche di Tommy Tallarico, all'epoca un tester della Virgin Interactive che si offrì di realizzare la musica del gioco in via del tutto gratuita.

Grazie alle attività di ingegneria inversa degli appassionati è stato possibile documentare dettagliatamente i formati dei file della versione MS-DOS, creando persino editor di livelli che hanno consentito di realizzare oltre sessanta mod. Il 17 aprile 2012, Jordan Mechner ha creato un repository GitHub contenente il codice sorgente della versione originale per Apple II di Prince of Persia, con tanto di documenti tecnici.

Di particolare rilevanza è la recente conversione del gioco per Commodore 64, eseguita da Andreas Varga (Mr. SID) il 16 ottobre 2011. Il gioco è basato sul codice originale per Apple II del 1989 e funziona su un C64/128 reale solo usando una cartuccia EasyFlash, Ultimate 1541 II/II+ o qualsiasi altro sistema compatibile con le cartucce in formato Easyflash. La conversione C64 comprende tutti i 13 livelli del gioco originale, offre un'elevata velocità, controlli precisi e fluidi, effetti sonori efficaci e un risultato in generale eccellente. La risoluzione degli sprite non è elevatissima, ma è compensata da ottime animazioni. Per superare gli occasionali rallentamenti del gioco è possibile utilizzare il gioco su Commodore 128 in modalità 64 in modo da sfruttare la maggiore velocità della CPU (2,04 MHz).
Il manuale è integrato nel gioco ed è possibile visualizzarlo prima di iniziare l'avventura.

di Adriano Avecone

RIFERIMENTI WEB

https://it.wikipedia.org/wiki/Prince_of_Persia

GIUDIZIO FINALE

» Giocabilità 90%

Prince of Persia è una pietra miliare del divertimento videoludico. Se ce ne fosse ancora bisogno e' la palese dimostrazione che non serve una grafica da capogiro per creare un capolavoro.

» Longevità 90%

Sono passati quasi 30 anni dalla sua uscita, ma quando un gioco è fatto bene esce indenne anche dalla prova del tempo.
Riprendetelo in mano e come me vi accorgerete che e' ancora bello come un tempo. Forse di piu'. :-)

Ed anche questa è fatta

Le guardie dei primi livelli sono facili da sopraffare, ma andando avanti col gioco la loro abilità con la spada cresce.

SUPER DEMON ATTACK

Publisher: Imagic
Anno: 1982
Piattaforma: TI99/4A
Genere: Sparatutto

Demon Attack è uno sparatutto creato da Rob Fulop per l'Atari 2600 e pubblicato dalla Imagic nel 1982.

Fu creato basandosi sul concept dello sparatutto arcade del 1979 Galaxian, anche se ricordava da vicino, specialmente nei primi livelli, l'arcade del 1980 Phoenix.

Tali somiglianze, effettivamente molto forti, scatenarono una causa da parte di Atari, Inc., che aveva acquistato i diritti legali su tale videogioco. Dalla diatriba legale, da cui la Imagic si chiamò fuori, contribuì a far si' che Demon Attack diventasse il gioco più venduto nel 1983.
Tale notorietà portò alla nascita di molte conversioni, che furono create per le maggiori console dell'epoca, quali ad esempio l'Intellivision e l'Odyssey e sui più famosi computer.
Ricordiamo versioni per la famiglia Atari a 8 bit (Atari 2600, 5200, 800, 800XL, 130XE), il VIC-20 ed il Commodore 64, i PC IBM (come booster), il TRS-80 Color Computer, l'IBM PCjr ed il TI99.

La conversione che fu creata per il TI99/4A merita una menzione a parte ed e' l'oggetto di questa recensione.

Secondo quandto riportato dall'ex dipendente della Texas Instruments Bill Barneia, questo porting della versione dell'Atari 2600 fu rinominato in Super Demon Attack per ragioni di marketing. Sembrava una evoluzione dell'originale ma in realta' non era così!
La logica che fu usata era che, aggiungendo "Super" al titolo del gioco, il tutto sarebbe sembrato un seguito e quindi un miglioramento rispetto alla versione originale del gioco Atari 2600.

L'idea di cambiare il titolo aggiungendo l'aggettivo "Super" venne concepita in ritardo vedendo il gioco finito (migliore grafica, sonoro, effetti vocali ecc.. rispetto a quella per l'atari 2600) e non e' esente da errori derivati dalla fretta di attuare il cambio.

Indovinate che livello e'...

Le immagini sopra e sotto questo riquadro si riferiscono alla versione Atari 2600. Niente a che vedere con la conversione del TI99/4A.

Mostro di fine livello Atari 2600

Versione TI99/4A

La grafica della versione Ti99/4A e' decisamente su un altro pianeta, come si evince anche da questa immagine. Spero abbiate colto il doppio senso. :-)

Infatti nella schermata di selezione del gioco sul TI99 si legge Demon Attack, così come nella schermata del titolo del gioco.
Inoltre, un manuale per Demon Attack (senza "Super" nel titolo) poteva essere trovato nel catalogo di giochi del 1983 pubblicato dalla TI e nel catalogo Triton della primavera del 1984.

Tutte queste incongruenze sollevano la domanda se qualche cartuccia o manuale del gioco siano stati venduti con il nome Demon Attack anziché “Super”.

Negli anni che seguirono la sua uscita vi furono speculazioni sul fatto che si trattasse di un porting dell'originale o di un vero sequel, ma a distanza di 36 anni possiamo affermare con certezza che questo titolo è davvero la conversione del famoso Demon Attack!

Passando finalmente ad analizzare la versione per il TI99 ci accorgiamo che presenta molte particolarità interessanti, specialmente a livello di programmazione.

Notiamo immediatamente un utilizzo sapiente del sintetizzatore vocale, degli Sprite in modo multicolore e per quanto riguarda lo sfondo che sembra in alta risoluzione. Il tutto condito da una alta giocabilità che, dopo Parsec, ha posto questo gioco per decenni tra le prime cinque game-killer application per il TI99/4A.

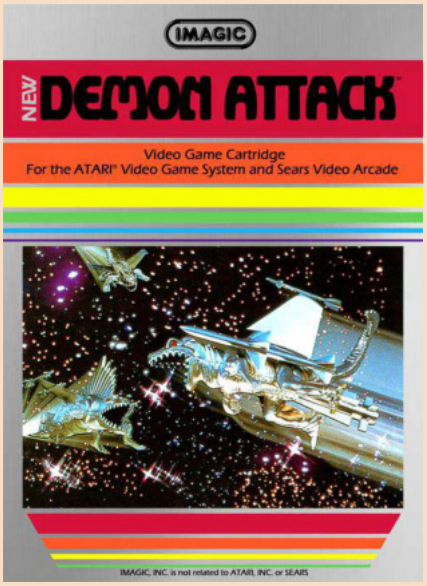
La versione originale di Super Demon Attack per il TI99 ha addirittura degli effetti vocali che riproducono durante la partita una voce sintetizzata che commenta lo stile di gioco del giocatore.
Questo che possiamo definire a ragion veduta un valore aggiunto, fu successivamente eliminato per ragioni sconosciute dalla versione su cartuccia. Questa limitazione tolse molto del feeling che questo gioco emanava.

La versione beta con presente l’effetto voce è sopravvissuta per oltre 30 anni solo in formato floppy disk grazie allo scambio tra user group americani ed europei. E' solamente in questi ultimi anni che è stata convertita su cartuccia.

Possiamo oggi affermare senza timore di smentita che la Imagic creò per il Ti99 una delle migliori conversioni di Demon Attack.
Una pietra miliare che ogni appassionato gamer dovrebbe possedere.

Naturalmente la versione con la voce sintetizzata ;-).

di Ermanno Betori



GIUDIZIO FINALE

» Giocabilità 90%

Demon Attack come già detto è stato per quasi 30 anni una pietra di paragone di come doveva essere concepito un arcade sul TI99, controlli molto reattivi, grafica suoni ed effetti vocali che all’epoca fecero sognare gli utenti. Cosa altro aggiungere?

» Longevità 85%

Dopo aver distrutto il demone e finito il gioco per due volte la difficoltà aumenta esponenzialmente perciò dal terzo livello in poi solo i veri gamer riescono ad andare avanti creando agli utenti normali un poco di sconforto... Io sono riuscito solo una volta a ricominciare il gioco per la quarta volta. ;-)

Mostri di fine livello

Intervista a Dino Yachaya

di Starfox Mulder

Starfox Mulder: Ciao Dino è un piacere poterti intervistare per RetroMagazine. Ti andrebbe di presentarti ai nostri lettori? Chi sei e cosa fai assieme ad Inty Home per tenere viva la fiamma della nostra passione comune?

Dino Yachaya: Io sono prima di tutto un collezionista, colleziono ogni tipi di console dalla prima generazione a oggi ma ovviamente la mia preferita è intellivision, la prima che ho avuto da piccolo alla età di 8 anni. Intellivision mi è sempre rimasta nel cuore. Negli ultimi anni avevo avuto sempre l'idea di sviluppare qualcosa di nuovo per questa console. Conoscendo Roberto (vroby), ottimo programmatore, siamo riusciti a sviluppare nuovi giochi. Roberto è il mio socio in inty-home, proveniente da diversi software, aveva già sviluppato per Commodore 64 e Amiga. Prima di conoscere me quasi non conosceva intellivision. Inty-home nasce per produrre giochi homebrew per intellivision, siamo gli unici in Italia e penso in Europa. Siamo un team molto piccolo: io, Roberto e da poco (visti i buoni risultati ottenuti) anche Marco e Lamberto i quali ci aiuteranno nello sviluppo del prossimo gioco.

S.M.: Quo Vadis è considerabile un action adventure con elementi platform che si combinano all'esplorazione. Quali strumenti, quali risorse e quali tecnologie sono state impiegate per realizzare questo progetto e quanto tempo ha richiesto complessivamente il suo sviluppo? Localizzazione in latino (a proposito complimenti per l'originalità), realizzazione delle cartucce fisiche, overlay, come avete affrontato tali fasi? Puoi parlarci anche degli strumenti di sviluppo impiegati?

D.Y.: Innanzitutto voglio sottolineare che facciamo quasi tutto "in casa", un homebrew nel vero senso della parola. La programmazione è fatta in assembler tramite compilatori inty-basic, le cartucce (shell o involucri di plastica) le abbiamo disegnate noi e le produciamo realizzandole tramite stampanti 3D , quindi le facciamo con il colore che ci piace di più (bianco per Quo Vadis) e che le grafiche che vogliamo, istruzioni, adesivi e overlay disegnati da me e stampati in tipografia sempre qui a Pavia. Solo le board (parti interne elettroniche alle cartucce) vengono comprate esternamente in Canada in quanto non siamo ancora in grado di realizzare un circuito stampato ma contiamo di arrivare a farlo. Tutti gli homebrew del mondo hanno lo stesso fornitore in Canada al momento ma noi vorremmo essere i primi anche a fare le board. Come tempistiche di realizzazione Quo Vadis è stato piuttosto veloce. Roberto in principio lo aveva iniziato come test per provare il movimento a 8 direzioni, mai fatto prima su intellivision, ma una volta in sviluppo ci siamo accorti che il titolo poteva essere valido e quindi lo abbiamo terminato. Il latino è venuto fuori in quanto il gioco è ambientato nel periodo dei templari e ci sembrava appropriato e originale.
A conti fatti ci son voluti circa 6 mesi dalla prima bozza alla messa in vendita.

S.M.: Il gioco è esaurito in poco

più di due mesi, pensate di fare delle ristampe in futuro?

D.Y.: Il gioco è stato esaurito per il 50% in preordine e il resto in una settimana dall'uscita. No, niente ristampe , la nostra politica rimane di fare sempre 100 copie in serie limitata, così è stato anche per il precedente gioco (Gyruss) e così sarà per il prossimo. Vorremmo distinguerci anche così: pochi pezzi ma di qualità!

S.M.: State realizzando altri progetti originali per Intellivision? Ho letto bene che c'è in cantiere un gioco ispirato alla serie TV "Spazio 1999"?

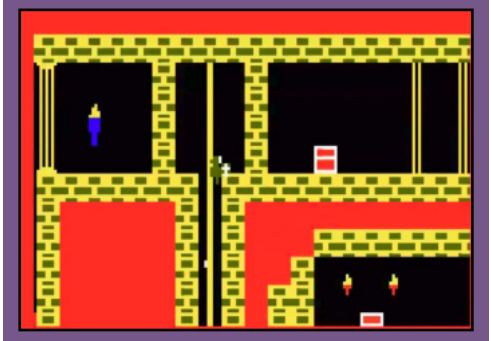
D.Y.: Sì, dovrebbe essere il prossimo gioco. Spazio 1999 sarà un gioco rivoluzionario per intellivision poiché per la prima volta un gioco sarà realizzato su due cartucce. Vorremmo pubblicarlo per il 13 settembre, quando ricorre il ventennale del "distaccamento della base lunare Alfa dalla terra" nel telefilm originale. Così ci è stato anche chiesto dal fan club di "Spazio 1999", anche se non sarà facile, settembre è vicino. Inoltre in "Spazio 1999" useremo degli accorgimenti tecnici mai utilizzati prima su intellivision, proveremo a portare la console oltre i suoi limiti.

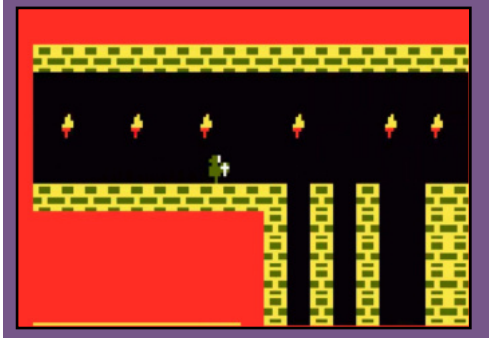
S.M.: Cosa ne pensi del nuovo Intellivision Amico?

D.Y.: È un progetto molto ambizioso ma a mio avviso più rivolto al mercato attuale che agli appassionati di Retrogaming come noi. Fino ad oggi si è visto poco in merito, quindi è difficile giudicarlo; sicuramente ritrovare il marchio intellivision è affascinante. Io avrei voluto più una console tipo l'annunciata (e mai realizzata) intellivision 3 degli 80es.

S.M.: Ti ringrazio tanto per la disponibilità e i dettagli delle risposte, ci hai lanciato qualche scoop niente male ed ora l'attesa di Settembre si farà decisamente più carica di aspettative. Visto che ne ho l'occasione ti ringrazio pure per il lavoro che tu, Roberto ed il team di Inty home state facendo. Finchè ci saranno appassionati come voi nessuna console morirà mai!

D.Y.: Grazie a voi!





Giappone 3^puntata: Akihabara

di Michele Ugolini

Complicato, si, se dovessi riassumere l'intero articolo con un unico aggettivo, proporrei l'aggettivo "complicato" poiché in questa lettura introduco in maniera piuttosto allucinante e spero coerente, ciò che potrete vivere in questo magico luogo per godere di una moltitudine di prodotti del gaming, retrogaming e di ciò che appartiene al mondo elettronico giapponese passato, presente e, discorso soprattutto attinente al Giappone: futuro!

Più precisamente vi descriverò una tipologia di mercato multimediale ed elettronico che all'apparenza è stato trasfigurato a puntino, con sembianze allettanti al mondo occidentale, ovviamente completamente ignorabili all'interno del loro sistema.

Cercherò di esporvi un universo finalizzato, appena usciti dalla metro e gettati in pasto alla folla delle strade di Akihabara, ad orientare il naso all'insù, aprire la bocca, esclamare la propria meraviglia, congelarsi nel centro delle loro vie principali, dichiarare stato di emergenza a tutte le ghiandole sudoripare del corpo e produrre adrenalina per tutta la giornata. Il primo tuffo dentro Akihabara, lo confesso, è entusiasmante, si assapora un mix di emozioni frizzanti e contrastanti tra le quali l'amaro di non aver potuto godere di tutto ciò anni e anni prima di quel momento ma... alla fine della giornata e soprattutto alla seconda alla terza o alla decima visita... scapperete in mezzo alle migliaia di persone calibrando il vostro tragitto in modalità tipicamente giapponese: con la mano alla fronte proiettata longitudinalmente innanzi al vostro percorso che mostra alla folla la vostra direzione e soprattutto la vostra fretta di spostarvi e di seminare le migliaia di persone che vi circondano per poi proiettarvi al prossimo gruppo di migliaia di persone che, ovviamente, aspettano soltanto voi!

Ma non dovevo parlare di Akihabara? Perchè vi sto parlando di un piano di fuga?

Si scusatemi, ma è ancora fresco in me il ricordo del mio recente acquisto, dovevo comperare degli Hyoushi-Ghi e l'edificio di strumenti musicali era distante poche centinaia di metri dalla metropolitana, eppure, sfidando la sorte per stupida presunzione vantandomi di essere un discreto conoscitore del Giappone, per spostarmi a piedi da un punto A ad un punto B nell'orario di punta serale di un sabato sera.. ci ho impiegato più di un ora!

Quindi ritorniamo a noi. Cercando tramite Wikipedia la parola Akihabara compariranno eloquenti, serene, confortanti ed esaustive righe di descrizione di questo sovraffollato quartiere di Tokyo, chiamato anche “Electric Town” ed “Akiba” per gli autoctoni: "[...] Akihabara è famosa soprattutto per la sua grande concentrazione di negozi che vendono tutti i tipi di apparecchi elettronici, anime, videogiochi. Probabilmente è la più vasta area di vendita del mondo per beni elettronici e computer, inclusi oggetti nuovi e usati. Il materiale nuovo si può trovare nei negozi della strada principale, mentre l'usato di tutti i tipi (software, hardware e molto, molto altro) si può trovare nelle vie sul retro. Parti di ricambio nuove per gli amanti del fai-da-te sono facilmente disponibili in molti negozi. Attrezzi, parti elettriche, cavi, videocamere miniaturizzate ed altro si possono trovare nei minuscoli corridoi vicino alla stazione. I turisti stranieri tendono a visitare i grossi negozi vicino alla stazione, mentre i giapponesi locali sanno dove ottenere una varietà di scelta e di prezzi migliore allontanandosi dalla stazione. Oltre che per gli appassionati di elettronica è anche una miniera per gli appassionati di anime: qui si trovano numerosi negozi dove si possono acquistare anime, manga, dojinshi, OST, action figure, costumi per il cosplay e locali tematici come i maid café. Come per i negozi di elettronica si può andare da negozi multipiano (come il famoso Mandarake) a piccoli negozi specializzati che occupano una sola stanza [...]".

Ciò che Wikipedia non racconta è "cosa" e soprattutto "chi" vi aspetta, nel più stretto senso della parola: qui si aspettano gli occidentali.

A differenza degli USA che, non sempre, bensì spesso, lasciano dei dubbi di proprio vuoto interiore, tanta pubblicità e poco prodotto, gigantesche scenografie per offrire un prodotto senza anima e senza storia.. qui in Giappone troverete ogni tipo di mercato, quello assordante e vuoto dentro, per arricchire il sistema attorno a voi, fino a quello della contemplazione, dove vi potrete allenare al vuoto interiore per arricchire la vostra stessa anima.

Tutto è adrenalinico, pieno di spettacoli, effetti luminosi, gigantesche insegne super luminose, effetti laser ovunque, musiche assordanti provenire da qualsiasi negozio, sorridenti e chiassose ragazzine vestite in cosplay da cameriera (tipo Maid Cafè) pronte a consegnarvi pubblicità ed offerte della loro ditta, grattacieli infiniti in altezza e quantità che vi circondano e aspettano per offrire prezzi stracciati ed occasioni. Il tutto si dispiega lungo le strade cosparse di automobili dalle articolate e costosissime modifiche in "modding" e "tuning" dove farle ballare sugli ammortizzatori telecomandati a ritmo di musica assordante proveniente

dall'abitacolo è la prassi più scontata in assoluto! La soglia di attenzione della spesa viene azzerata, la paura di tornare a casa a mani vuote è già morta quando siete scesi all'aeroporto, l'ansia di farsi sfuggire tutte quelle mirabolanti occasioni che vi circondano vi farà sudare acido fino alla sera, uscire gli occhi dalle orbite, proverete senso di nausea, crampi addominali, secchezza delle fauci, sarete stremati, è così pieno di occasioni, prezzi favolosi, ultimo modello qui, lì, anche là, moltitudine di gente, ovunque, davanti, dietro, negozi, luci, rumori, aiuto!

Bene, era mio intento vaccinarvi, ora avete un po' di dubbi, ansia e forse pure timore.

Adesso raccogliamo le informazioni e pianifichiamo precisamente cosa fare e cosa evitare in un luogo della Terra dove è un ottima idea non improvvisare ed è infinitamente cosa buona programmare ogni singolo spostamento sia con i mezzi pubblici che soprattutto a piedi. Sia per spostarsi da un punto A ad un punto B, sia soprattutto per ritornare al punto A.

Vi invito anche alla lettura dei miei due articoli precedenti sul Giappone, così il lessico specializzato a tale ambiente vi sembrerà più amichevole, altrimenti parole come "anime" , "otaku" e soprattutto "figure" potrebbero mostrarsi poco digeribili.

Figura 1

Un ultimo monito, dal momento che questo articolo sarà letto, quindi Giugno 2019, a causa dell'elevato metabolismo produttivo sfornato dall'instancabile macchina nipponica, le informazioni diventeranno obsolete nel giro di poco tempo, i prodotti elencati nella pagina probabilmente saranno riassortiti nel giro di poche settimane, alcune ditte subiranno fusioni o si separeranno, la fisionomia di una strada nel giorno che la osserverete sarà graficamente modificata in brevissimo tempo, l'unica costante saranno le arterie stradali con il loro composto traffico cittadino. Quindi vi invito ad aggiornare le vostre informazioni relativamente al mese che visiterete il Giappone e che vi adopererete alla caccia di retrogaming.

Cosa si può fare ad Akihabara, soprattutto per noi amanti del retrogaming? Tutto e sicuramente anche ciò che la nostra fantasia non potrebbe osare. Parliamone.

Figura 2

- ***Akiba info***

http://www.akiba-information.jp/html/akibainfo/about-info.htm (cfr. figura 1)

Un buon punto di partenza, troverete massima cortesia al banco informazioni, massima folla davanti e dietro di voi e non osate saltare la fila o far entrare un vostro amico arrivato in un secondo momento: in Giappone non si fa. Ecco a voi intanto un pratico Pdf del quartiere. Guardatelo con calma, se pianificate una giornata di caccia al retrogaming fatelo con estrema attenzione calcolando spostamenti e ritorni per muovervi agevolmente tramite treni e metro. Preferibilmente, non scegliete orari di punta.

http://www.akiba-information.jp/html/multi/DownLoad/townmap-e.pdf

Figura 3

• ***Super Mario Kart***

Quante notte insonni avete passato disperandovi e inzuppando il pigiama di sudore per non aver mai guidato nella realtà il Kart di Super Mario ? Magari vestiti da Hello Kitty. Come vi capisco.

Fatelo, dovete mettervi in gioco poiché difficilmente si potrebbe presentare una seconda occasione nella vostra vita.

https://akihabara.maricar.jp/ (cfr. figura 2)

Dovrete essere muniti unicamente di un certificato di guida internazionale, recarvi alla ditta descritta nel link, pagare l'assicurazione e perfezionare il contratto scegliendo il tempo di noleggio, un ora , tre ore oppure tutta la giornata. Potrete scegliere un costume tra i mille proposti, montare l'action cam sul casco, farvi fotografare da milioni di persone lungo le vie.

Ovviamente non dovrete investire nessuno nè creare situazioni pericolose.

Potrete viaggiare incredibilmente fino ai 60km/h e parcheggiare nei normali posti delle automobili pagando normalmente la sosta. Ogni carenza di educazione sarà immediatamente e sonoramente punita con una tale efficienza (e rapidità) da lasciare anch'essa a bocca aperta: sarete sorpresi dal grado di educazione della Polizia durante l'elevazione della multa.

• ***Yodobashi camera Akiba***

Impossibile non raggiungerlo. Non lo state trovando? In realtà avete passeggiato attorno al suo perimetro già svariate volte, senza comprendere dove si entra.

Questo ipermercato della tecnologia non ha un ingresso, bensì centinaia, per raggiungerlo è molto semplice, voi entrate da qualsiasi ingresso dell'edificio otticamente più monumentale che vi appare a destra dall'uscita della metro poi, una volta dentro, dirigetevi verso le aree più rifornite del materiale che state cercando, lì attraverso la modalità "subfolder" provate ad individuare nei vari corridoi, pieni zeppi di materiale, ciò che stavate cercando. Non c'è bisogno che vi dica che troverete, non solo ogni prodigio elettronico appena lanciato sul mercato nipponico, ma soprattutto l'intera genealogia storica dei predecessori di quel prodotto, regola specialmente valida per il retrogaming, robotica, fumettistica, etc..

https://aroundakiba.tv/stories/yodobashi (cfr. figura 3)

Definirlo negozio è paradossale, si potrebbe definire una gradevole collina artificiale sovra illuminata.

E' probabilmente l'edificio più imponente di Akihabara destinato alla vendita di tutto ciò che la vostra esistenza non potrà più farne a meno dopo la prima visita.

Fidatevi, non fermatevi a curiosare i primi oggetti sotto i vostri occhi, andate oltre, infarinatevi leggermente su tutto ciò che esiste lungo i suoi nove piani e poi decidete quale piano visitare, altrimenti ci metterete tutta la giornata a visitare un piano e la durata della vostra vacanza terminerà in quello stesso giorno , stabilendo appena entrati, un piccolo accampamento indiano in qualche corridoio del primo piano!

Agli ingressi c'è un distributore di mappe dell'edificio (come in quasi tutti gli edifici voluminosi) che vi mostrerà le tipologie di articoli presenti nei vari piani. Non conviene descrivere con precisione tutte le tipologie dei prodotti anche perché quasi tutti gli scaffali, settimanalmente, vengono riorganizzati da capo a piedi.

Sommariamente, ad oggi, nei rispettivi piani, si può elencare la presenza di:

·1f Cell Phones, Computers, Tablet Accessories, Memory Counter, Apple Store

·2f PC Parts, PC Software, PC Accessories, PC Books, Printers, Fax Machines and Phones, Digital Writing Goods, Suitcases

·3f Film Cameras, Digital Cameras, Video Cameras, Watches, Brand Goods, Film, Camera Accessories, Cosmetics, Watch Repair, Souvenir Corner, Tourist Corner

·4f Audio Visual Goods

·5f Heating and Cooling, Kitchen Appliances, Appliances, Emergency Lighting/Goods, Hearing Aids

·6f Games, Toys, Instruments, Bicycles, Music, Adult Software

·7f Various Shops (Bookstore, Music Store, Glasses, Formal Wear, Handbags, etc)

·8f Food Court and Game Corner

·9f Golf Store, Driving Range, Batting Café

Arrivati all'ottavo piano sarete adrenalinici, euforici, disidratati ed affamati quindi gli sterminati bar e ristoranti sapranno cosa

necessitate, infine al nono piano c'è il classico grande campo da golf che potrete utilizzare unicamente se nel portafoglio sarà rimasto un abbondante ottimismo!

• ***Kotobukiya***

http://en.kotobukiya.co.jp/kotobukiya-akihabara-store-floor-guide/ (cfr. figura 4)

Cosa non si trova qua dentro?

Al primo piano troverete il paradiso retro e odierno di: Monster Hunter (Official Hunter's Shop), Yotsuba&! (Yotsuba&! and Danboard Official Store), Nintendo, Dragon Quest, Final Fantasy, Star Wars, Marvel, DC Comics, novelty snacks, virtual Youtubers, Ghibli.
Al secondo piano: Tales Series Shop, THE IDOLM@STER, THE IDOLM@STER SideM, Fate, ATLUS, Azur Lane, White Cat Project, My Hero Academia , Kyoto Animation/Free, Love Live!, Ensemble Stars, Sword Art Online, Cardcaptor Sakura, Sailor Moon, CyberConnect2, Arc System Works, Kotobukiya es series.
Al terzo: Frame Arms, Frame Arms Girl, Hexa Gear, Megami Device, Cu-poche, Girls und Panzer, Kotobukiya plastic model kits, M.S.G, Kotobukiya figures, HMM Zoids, hobby supplies, display cases.
Retro collezionisti parliamoci chiaro, è il nostro paradiso e forse il nostro inferno, ne uscirete devastati psicologicamente ed economicamente. Entrate a vostro rischio e pericolo solo se disponete di un budget notevole.

Figura 4

• ***Tsukumo Robot Kingdom***

Avete deciso che i droni che svolazzano nel parco davanti casa devono subire una fine simile ai personaggi di GTA?
Terminator prima edizione è sempre stato il vostro sogno nel cassetto, per poter eliminare tutte le Sarah Jeanette Connor che vi infastidiscono nella giornata?
Volete costruire il braccio robotico pasticcione, assistente di laboratorio di Tony Stark, direttamente sulla vostra scrivania e comandargli di spaventare il vostro cane?
Siete nel posto giusto.
Retrogaming e robotica fusi insieme allo stato puro, il regno della robotica dagli albori ad oggi, dove l'occidentale diffusione dell'adorabile Raspberry ammutolisce al cospetto di questi prodotti, contemplando in silenzio l'infinità di gioielli di fronte a sè.
https://robot.tsukumo.co.jp/ (cfr. figura 5)

Qui potrete realmente godere di una delle anime elettroniche del Giappone: la precisione della meccanica unita al controllo software e alla nutrita schiera di seguaci di questo o quell'altro software che organizzano settimanalmente campionati studenteschi e incontri di RobotWars. La sovrabbondanza di linguaggi macchina e soluzioni tecnologiche relative alla robotica sono qui. Asimo di Honda vi sembrerà meno lontano. L'innovativo Honda Assist Walking Device vi farà assaporare la virtuosa quotidianità della loro esistenza.

Figura 5

Figura 7

• ***Mandarake Complex***

Avete sempre desiderato leggere Ken il Guerriero in giapponese?
Avete sempre sognato di sfogliare i fumetti di Goku ed appurare che l'ultima pagina del manga coincide con la prima pagina della lettura?
Avete sempre fantasticato sull'idea di apprendere in quanto poco tempo si possa svuotare un portafoglio pieno zeppo di Yen?
Siete nel posto giusto.

https://earth.mandarake.co.jp/shop/ (cfr. figura 6)

Qui c'è tutto riguardo l'usato, probabilmente è tra i negozi più grandi al mondo. Se non trovate qualcosa qui, state tranquilli che vi resteranno solamente poche altre migliaia di occasioni, per reperire ciò state cercando, ovviamente dentro Akihabara!
Giochi, videogiochi, manga, libri, anime, bambole, merce hentai, Cd, Dvd, comics, etc..
Non fare una passeggiata dentro Mandarake significa diventare facile preda di convulsioni notturne per il rimpianto di non esserci mai entrati almeno una volta nella propria vita.

• ***Sega Arcade Building***

Nelle vie principali troverete spesso i negozi della Sega, rifornitissimi di videogiochi, tutta la loro storia videolutica è onorata qui dentro, dagli albori al futuro, su più piani, sempre pieni di gente di tutte le età.
https://segaretro.org/Sega_Akihabara (cfr. figura 7)

Ricordiamo quando le Slot, una decina di anni fa in Italia, avevano sostituito molti cabinati arcade delle nostre amate sala giochi.
Gli stessi cabinati arcade della nostra infanzia li ritroverete qui, originali, perfettamente funzionanti, una partita costerà a partire da 100Yen, troverete anche una quantità sterminata di videogiochi Sega che non sono mai sbarcati in Occidente.
Soprattutto non dovete perdervi le famose postazioni Sega di guida delle automobili e le capsule sferiche chiuse e motorizzate con proiezione olografica interna oppure a realtà aumentata internamente all'abitacolo. Fate una partita in LAN fra voi turisti del vostro gruppetto, non vi consiglio di sfidare in LAN i vostri vicini sconosciuti poichè potrebbero essere degli Otaku autoctoni: le capsule e le sfere hanno prezzi maggiori e i giapponesi sono estremamente agguerriti, tipo Goku, che non vedeva l'ora di trovare qualcuno più forte di lui. Questo è il posto giusto per loro, non per voi!

• ***Taito's Hey***

A due minuti da Akihabara JR Station troverete questo sibillino negozio della Taito, con tutte le sue leggende passate e odierne, costellato di centinaia di cabinati per videogiochi mai visti in occidente e soprattutto...
centinaia di macchinette col braccio meccanico che deve acchiappare il peluche ed altri oggetti! (Figura 8). Non l'ho considerato come una meta da Retrogamer ma... Taito è nel cuore di tutti noi e quindi... anche io ho provato ad acchiappare (inutilmente) un peluche a forma di astronave nemica di Space Invaders!
https://www.taito.co.jp/ (cfr. figura 8)

• ***Tokyo Leisure Land***

Volete entrare nel paradiso dimenticato delle sala giochi della nostra infanzia?
Eccovi accontentati, ma attenzione, l'area dedicata alla ricostruzione tipica per noi occidentali sarà frequentata da noi occidentali, mentre le mille postazioni nipponiche saranno occupate dai giapponesi.
Perchè? Vi sono due motivi, il primo è una questione lessicale: una volta

Figura 6

inserito il gettone, il linguaggio di default nei cabinati (a noi sconosciuti) sarà unicamente il Giapponese. Il secondo motivo è che non conosciamo la loro infinita distribuzione di personaggi, saghe, Idols, inseriti come videogame dentro le macchine, quindi senza trama potremo comandare il personaggio in maniera più debole e meno attinente.

Vi faccio un esempio realmente buffo che mi è accaduto qualche anno fa davanti a questo tipo di sala giochi.

La vera sala giochi giapponese è quanto di più simile possa esistere ad una discoteca nostra, con all'interno centinaia e centinaia di postazioni da gioco simili a "All we dance" oppure "Taiko no Tatsujin" e altri giochi musicali similari dove l'abilità sta nel seguire il ritmo e la coordinazione degli arti del corpo umano fornendo al momento giusto l'input alla macchina attraverso device predisposti all'acquisizione.

Non potevo credere ai miei occhi, un ometto sulla cinquantina, magro, filiforme, bassa statura, in divisa da lavoro, con la ventiquattrore appoggiata a terra e giacca piegata sopra... stava saltellando come una "Carla Fracci", leggiadro nonchè imbizzarrito, giocando ad un cabinato simile a “Dance Revolution" però molto più articolato e ricco di effetti luminosi e musicali.

Figura 8

Al video del cabinato c'era il personaggio principale a me sconosciuto (comunque una Idol giapponese) che doveva eseguire una serie di passi complicatissimi e rapidissimi e... quest'ometto li eseguiva con una tale precisione da congelare la mia attenzione davanti alla vetrina.

I miei occhi non riuscivano a credere ciò che vedevano: l'abilità di questo uomo adulto che giocava con tale agilità e convinzione ad un cabinato per ragazzine. Lui si che ce la poteva fare ed era sicuramente un vincente nella vita reale visto che sfogava con così tanta energia la massacrante giornata lavorativa, ballando come una ragazzina Idol virtuale, che probabilmente lo rappresentava.

Ho iniziato a registrarlo con la mia telecamera. Lui, anziché essere indispettito era ancora più fiero e dava ancor più il meglio di sé. Una danza frenetica dove mani e piedi dovevano dare gli input alla macchina nel momento e nel posto giusto.

Dietro di me iniziavano a fermarsi altri turisti stupiti dalla situazione e anch'essi iniziavano a videoregistrare.

Poi si sono aggiunte altre ed altre persone, anche autoctoni giapponesi, nel frattempo quest'uomo continuava a dare il meglio di sè. Ho ancora il video, ogni tanto lo guardo e magicamente svaniscono molti pensieri inutili, ammirando una realtà così lontana dall’ambiente quotidiano. Morale della favola , questo signore ha completato l'intero gioco con un gettone, con punteggio 100%. Alchè la macchina gli regala un piccolo spettacolino di effetti luminosi e musica a tutto volume per ricompensarlo dello sforzo generato... da venti minuti di danza folle.

E' improvvisamente scrosciato uno spontaneo ed unanime fragoroso applauso da tutta la folla, un

Figura 9

Figura 10

applauso così rincuorante dopo tutta quella fatica che l'ometto ci ha anche donato un inchino. Era fiero! Era riuscito! Aveva vinto e addirittura aveva conquistato i nostri cuori! Il suo sorriso e le lacrime lo stavano illuminando d'immenso. Finito il lungo applauso e gli inchini, infila la giacca, sistema la cravatta, prende la ventiquattrore e se ne va tra la felicità sua e lo sbigottimento del pubblico. Era diventato un “Idol”? No, per carità! Proprio no! Non sono neppure in grado di descrivere con precisione il fenomeno di questi artisti musicali e danzanti chiamati “Idol”, è una creazione loro, interna, molto intima che non è possibile associare ad un occidentale “Influencer” o ad uno “Youtuber” o “Streamer”, è tutto molto più complicato.

Se volete provare queste esperienze così tipicamente nipponiche, dove l'impossibile e soprattutto l'improbabile prendono vita ogni momento, entrate in questa specie di sala giochi giapponesi, non entrate nelle nostalgiche ma limitate emulazioni occidentali. Tuffatevi nella follia.

Qui per esempio:
http://llakihabara.sakura.ne.jp/intro.html (cfr. figura 9)

a soli 3 minuti camminando dalla stazione JR. Hours: 10:00-24:55 (altra follia giapponese, gli orari, questo locale per esempio chiude alle 24:55, è una effimera espressione che mira ad allungare le 24 ore della giornata portandole a 25, pertanto non stupitevi se troverete questo "bug", ovviamente 24:55 corrisponde all'una meno cinque minuti della notte).

• ***Gachapon Hall***

Che adorabili i distributori di palline tipo “sorpresa dell’Ovetto Kinder” che riempivano le sala giochi della nostra infanzia!

Ma dove sono spariti tutti? C'erano distributori di chewingum, giocattoli miniaturizzati, pupazzetti di tutti i tipi...

Eccoli qui:
https://grapee.jp/en/94733 (cfr. figura 10)

3-15-5 Sotokanda, Chiyoda 104-0061, Tokyo , 11AM - 8PM (10PM on Fri, Sat and 7PM on Sun)

In quasi tutti i negozi di elettronica in Akihabara troverete decine e decine di questi distributori allineati lungo file a ridosso degli edifici e se pensate che in Giappone siano moda passata vi sbagliate di grosso. A parte che troverete molto di ciò che c'era nella nostra infanzia, ma soprattutto troverete materiale che in Occidente ce lo sogniamo. Vi ricordate il fenomeno degli Exogini? Era nato tutto qua dal loro eroe Kinnikuman. Poi hanno creato tutti questi pupazzetti con varie mescole di plastica morbida con consistenza simile alle gomme da cancellare. Non solo mescole diverse ma ovviamente tante colorazioni e svariate serie limitate. Questi distributori hanno un pubblico costante da mattina a sera e vedrete sempre un omino che riempie i distributori. Se potrete stare vicino a questi distributori qualche minuto provate ad affinare l'occhio ai cacciatori/collezionisti di queste pallette, sono molto frequenti, anche perché non dovete credere che in tutta la via ci siano troppi distributori del medesimo articolo, eppure giuro, conterete migliaia e migliaia di questi Gacha- (cioè il suono

Figura 11.1

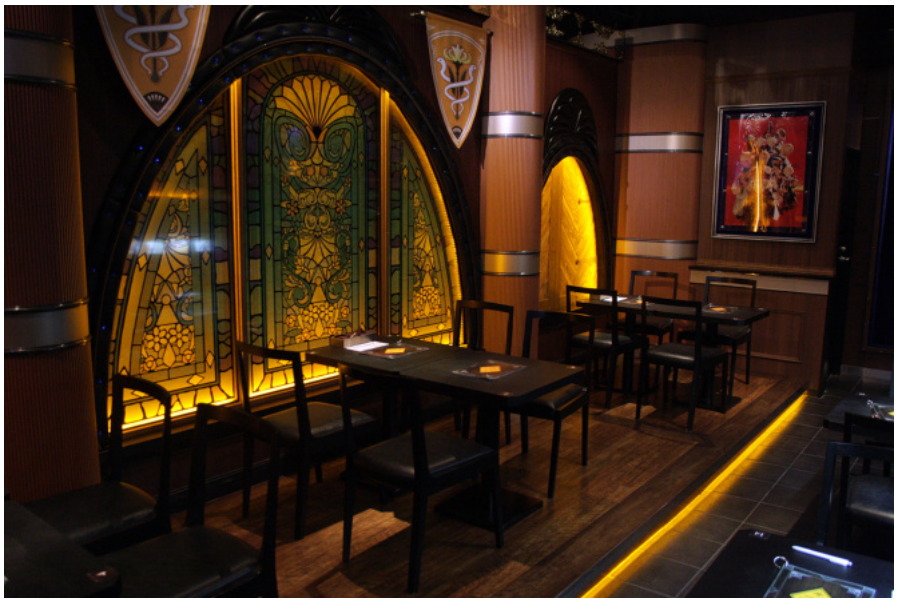


Figura 11.2

causato dalla manovella del distributore automatico) -Pon (il suono della capsula giocattolo nell'atto di cadere nel recipiente).

• ***Gundam Cafè oppure Final Fantasy Erzora Cafè' ?***

Amanti del retrogaming, dovete prendervi un caffè qui:
http://g-cafe.jp/en (cfr. figura 11.1)

oppure qui:
https://www.pasela.co.jp/paselabo_shop/ff_eorzea/ (cfr. figura 11.2)

Non servono descrizioni, non vi accenno nulla, tutto da scoprire.

• ***Donki***

No , no, mi spiace, l'assonanza a Donkey è fuorviante, in realtà Donki è il diminutivo di Don Quihotte.
No , no, mi spiace, nei negozi di Don Quihotte non vendono mulini a vento e non vedrete all'interno nè demoni nè eserciti di arabi.
In questi negozi vige una filosofia simile al "brainstorming".
http://www.donki.com/en/store/shop_detail.php?shop_id=98 (cfr. figura 12)

A tre minuti dalla Akihabara JR Station.
Potrete trovare di tutto, nel 90% delle probabilità troverete tutte cose inutili, la tipica cianfrusaglia, però magari proprio in mezzo a quegli "acchiappa polvere" potrebbero trovarsi oggetti da collezione super valutati in Occidente.
In un Donki in giro ad Osaka per esempio ho trovato l'edizione limitata di "Nausicaa della valle del vento", a poco più di 4000Yen, sigillata, come fondo di magazzino, un prezzo ridicolo per un indiscusso capolavoro cinematografico di Hayao Miyazaki (maestro di animazione riconosciuto a livello mondiale).
Qui ho anche trovato statuette di Goku a prezzi ridicoli solo per il fatto che erano usate e fuori moda poichè non raffiguravano l'ultima serie in produzione di Dragon Ball Super.

Figura 12

• ***Yellow Submarine Hobbybase***
http://www.yellowsubmarine.co.jp/ (cfr. figura 13)

E' inevitabile, dovete visitare anche questa ditta se siete collezionisti di card gaming, giochi da tavolo e mille altre cose non elettroniche, che neppure la nostra fantasia potrebbe lontanamente immaginare. Esistono talmente tanti giochi che non conosciamo, nonchè tutta la loro storia fino ad oggi, che si potrebbe passare l'intera vacanza in un unico negozio come questo. Vi assicuro che in poco spazio sono immagazzinati prodotti dalle cifre astronomiche e scaffali pieni di offerte strappalacrime per noi moderati e poco esigenti collezionisti occidentali! Vedrete la follia nipponica in mezzo a questi scaffali: vedrete girare senza sosta gli Otaku giapponesi e non, a tutte le ore, girovagare e tornare, contrattare selvaggiamente (pratica maleducata in Giappone sia per il ribasso quanto per una eventuale mancia), scambiare, etc.
Dimenticavo, prenotate con largo anticipo qualche duello di card game poichè il posto ai tavoli, sebbene ampio, è sempre oberato, infine ricordate queste parole: siete nel cuore della follia del collezionismo e dello scambio,

dovete adeguarvi ai ritmi folli.

• ***Super Potato***
http://superpotatoakiba.jp/ (cfr. figura 14)

E' uno dei cuori più potenti e riforniti del Giappone per la vendita di videogames dell'intera storia multimediale, noi retrogamer potremmo svenire all'ingresso, trovando nelle vetrine l'oggetto che abbiamo digitato milioni di volte su Google, Ebay ed Amazon in Italia.
Non voglio farvi desistere dall'entrare, ma sappiate che qui non si regala nulla, i veri oggetti da collezione sono sconosciuti a noi occidentali, quindi troverete delle statuette limitate a prezzi impensabili e troverete anche le sfere del drago di Dragonball a poche manciate di Yen. Dipende cosa cercate, se cercate qualcosa di collezionabile giapponese, state molto attenti alla spesa.
Se invece cercate qualcosa di collezionabile occidentale troverete molto materiale usato e riutilizzato talmente tante volte che è caduto nel loro dimenticatoio.
Vi auguro buona ricerca, io solitamente ci metto un intero pomeriggio a rovistare nei loro schedari di cartucce e dischi, scelgo sempre una giornata dove arrivano i rifornimenti, possibilmente lontana dal weekend, in orari che la gente è a scuola e in ufficio mentre io, in vacanza e senza problemi di orario, ho più tempo libero.

• ***Radio Center, Radio Kaikan***
Ad un solo minuto dalla solita stazione della JR.
Anche voi siete "OM"/"HAM" come me? Allora 73!
Dovete passare qui:
https://www.japanistry.com/radio-center/ (cfr. figura 15.1)

troverete di tutto e soprattutto ciò che non speravate di scovare. Se non avete individuato qualcosa che cercate, non preoccupatevi, l'oggetto desiderato sicuramente è presente nel loro catalogo ed arriverà in negozio entro il giorno seguente.
Siamo nel cuore elettronico di Tokyo, qui i ritmi sono estremi, il rifornimento è sacro ed immediato. Provare per credere.
Ho comprato un tasto verticale telegrafico alla ridicola cifra equivalente di venti Euro.
Icom e Yaesu sono i padroni di casa e le loro bande di lavoro spesso non coincidono con le nostre, però tranquilli che i kit di modding non sono vietati in Giappone visto che li utilizzerete in Italia (ovviamente sono vietati in Italia, se li applicherete una volta arrivati a casa dal Giappone).
Attenzione, la Dogana potrebbe chiedervi cosa sono quelle strane voluminose radio che trasportate, quindi consiglio sempre ai miei amici OM di portare dietro nominativo internazionale e fotocopia della licenza.
Poco più distante , anche Radio Kaikan potrebbe riservare delle sorprese (non sempre del mondo OM):
http://www.akihabara-radiokaikan.co.jp/ (cfr. figura 15.2)

Figura 13

• **Magical Trip**
http://bit.ly/2FAkAF9 (cfr. figura 16)

La vostra vacanza in Giappone si protrae per più di due settimane e

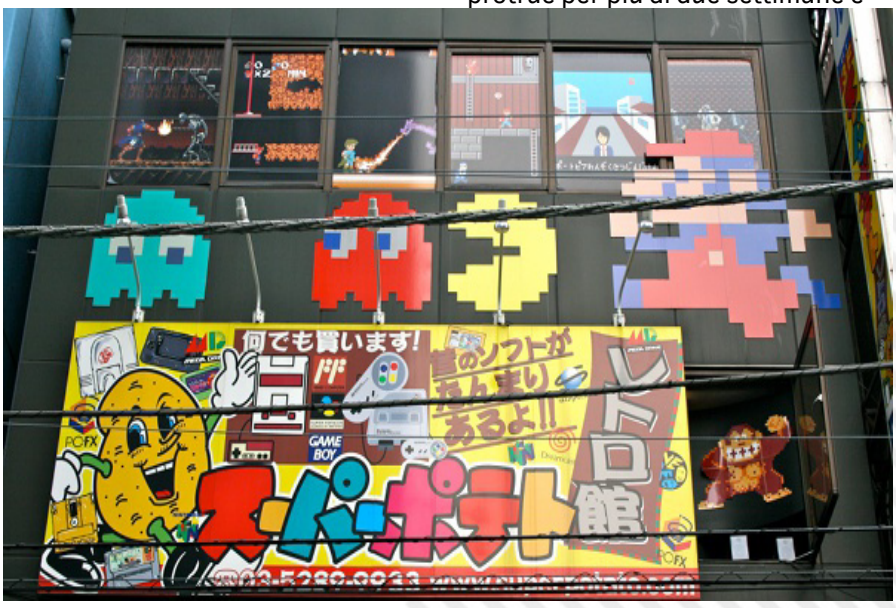


Figura 14

Figura 15.1

soffrite di notti insonni poichè state perdendo dell'inutile tempo, dormendo in un luogo così adrenalinico? Come vi capisco! Svegliatevi immediatamente ed uscite per strada, l'illuminazione notturna di Tokyo è probabilmente superiore a quella di una giornata nuvolosa e i servizi funzionano 24 ore su 24, magari siete in preda a qualche allucinazione data dall'insonnia ed è il momento giusto per prenotare un tour organizzato dalla "Magical trip" e svolgere per una intera mezza giornata il famoso "Anime and Gaming Adventure Tour in Akihabara". Potrete infarinavi sulle migliaia di mega-ditte che potranno offrirvi ogni tipo di articoli dal passato ad oggi. Un "must" per noi retrogamer.

• ***Janpara***
http://www.janpara.co.jp (cfr. figura 17)

Amanti Apple, dovete visitare questa ditta, troverete di tutto e di più, oggettistica retro e all'ultimo grido.
Riparazioni, cellulari di ogni epoca e da collezione.
Ricordatevi che lo standard di frequenze nipponiche non coincide obbligatoriamente con le specifiche europee, quindi informatevi bene, oltretutto le schede Sim ricaricabili sono rare e malviste in Giappone (a causa della legislazione riguardo la sicurezza interna). Ho sempre trovato molta difficoltà nella telefonia vocale. Fortunatamente ho sempre riscontrato molta facilità nella navigazione dati tramite il noleggio dei Pocket Wifi, ritirabili anche in aeroporto.

• ***Tamashii Nations Tokyo***
https://tamashii.jp/tokyo/ (cfr. figura 18)

Collezionisti di figure, qui siete nel posto giusto, Gundam e Mazinga sono le icone più rappresentate.
Vi consiglio di entrare in questi luoghi meno frequentati da noi retrogamer poichè in alcune scaffalature remote, all'interno di queste mega-ditte, potrebbe comparire insospettabilmente e magicamente una cartuccia originale di Pac Man a prezzi ridicoli.
Effettivamente a me è accaduto: Arkanoid per Super Famicom al prezzo di 100Yen! Forse sono riuscito a trovarlo perchè al mercato nipponico è venuto completamente a mancare il ricordo di un passato così remoto ed i collezionisti moderni non cercano più titoli così preistorici. Oltretutto questo mercato arcaico è stato ignorato anche dal turista occidentale, che appunto non è riuscito a raggiungere questi luoghi meno visitabili senza una adeguata conoscenza del territorio: ora conoscete anche queste perle di speranza!

• ***Tokyo Anime Center***
https://animecenter.jp/en/ (cfr. figura 19)

Anime di tutti i tipi dal passato ad oggi, raggiungerlo è molto semplice seguendo le istruzioni direttamente dal sito.
Ichigaya Station (Yurakucho Line) : 1-minute walk from Exit 6
Ichigaya Station (Namboku Line) : 1-minute walk from Exit 6
JR Ichigaya Station : 5-minute walk
Ichigaya Station (Toei Shinjuku Line) : 6-minute walk from Exit 1

Figura 15.2

Figura 16

Non ricordo con precisione di essere stato qui, probabilmente in quell'ipotetica mia visita avevo raggiunto la soglia di allucinazione nipponica dopo le troppe visite nei negozi dedicati al collezionismo di anime.
Ad ogni modo, non stupitevi se troverete il 99% di materiale sconosciuto, dovete considerare che il mercato interno è già saturo di vendite, pertanto agli editori giapponesi non interessa gettare in pasto a noi occidentali qualcosa che non potremmo apprezzare per divergenze culturali o livelli inadeguati di maturazione socioeconomica, con conseguente rischio di limitato profitto.

• ***Sofmap, Yamada Denki, Laox***

Ecco a voi altre tre mega-ditte disseminate in numerosi palazzi dentro Akihabara che meritano di essere visitate per il discorso appena affrontato, qui prende vita la maggior probabilità di reperire materiale da retrogamer: sono luoghi insospettabili e poco frequentati dai giapponesi autoctoni e soprattutto poco attraenti a noi occidentali.
Sono delle catene di potenti ditte che offrono di tutto, la parola chiave ovviamente è "materiale elettrico", quindi giochi, videogiochi e qualsiasi cosa sia alimentata da elettricità. Troverete vari palazzi dedicati a ciascuna di queste mega-ditte, basta guardare in alto e salire dal piano terra tramite le scale mobili.
Come sempre, a ciascun ingresso, troverete la mappa cartacea dei prodotti interni disseminati ai vari piani.
Prendetela e infarinatevi superficialmente sui prodotti, per poi iniziare la caccia in maniera approfondita.

• ***Akihabara Gamers Main Shop, Akky***

Queste due ditte non sono titaniche, ma vi sfido ad affittare mensilmente una manciata di metri quadri di attività commerciale in questi luoghi folli: ogni centimetro è importantissimo, fidatevi, è anche saturo di prodotti stipati all'inverosimile. Se queste ditte stanno sopravvivendo da anni significa che stanno realmente lavorando, quindi meritano una chance di visita poichè potrebbe essere proprio qui il gioco che in Occidente ha prezzi improponibili e che state cercando da una vita.
Merita particolare attenzione Gamers:
https://www.gamers.co.jp/shop/2434/

Però non fatevi trarre in inganno, non è il paradiso dei gamers e soprattutto non è il luogo più privilegiato di occasioni da prendere al volo, più di metà della merce rappresenta i soliti articoli "civetta" che avrete visto in centinaia di altri negozietti più o meno anonimi lungo le vie principali. Ad ogni modo entrambi possono donarvi speranza nella ricerca, forse anche più delle mega-ditte super rifornite ed ovviamente super setacciate da noi collezionisti.

• ***Hamada***

http://www.hamada-dk.com/

Qui si parla del paradiso per i collezionisti di hardware datato, troverete le schede video all'ultimo grido ed una montagna di schede per le quali l'ultimo grido è stato urlato tanti anni or sono prima della detonazione di un condensatore! Buona caccia, attenzione alla polvere (quanto odio le ventole di raffreddamento non pulite) e alle saldature appuntite dello stagno (che avranno punto migliaia di mani).
Come sempre vi suggerisco di entrare qui dentro perchè magari l'adattatore Sharp, Commodore, Jamma che avete sempre cercato da cannibalizzare per un vostro progetto è proprio qui.

Figura 17

Figura 18

CONCLUSIONI

Non aggiungo altro in questo articolo, cari amici, non voglio spaventarvi, ho solo accennato cosa potrete trovare in Akihabara, solamente dal punto di vista di un retrogamer.

In realtà c'è infinitamente di più in base alla tipologia di acquirente che rappresentate. Ci sono servizi per tutto ciò che volete, dove la realtà supera di gran lunga la fantasia occidentale. Comunque, se visiterete tutti questi negozi descritti sopra, non vi basteranno due settimane e probabilmente troverete noie doganali per l'ammontare di prodotti acquistati. Per i miei colleghi "OM", vi consiglio di informarvi bene sulle frequenze di lavoro degli apparati radio venduti in Giappone e la discutibile utilità riguardo la modifica che apporterete invalidando purtroppo la garanzia.

Vi ricordo anche che Tokyo offre tanti servizi considerati improbabili ed impossibili in Occidente, quindi conviene provare almeno una volta nella vita, viceversa l'Occidente spesso non ha alcuna idea della sterminata reperibilità di qualsivoglia prodotto e componenti elettronico presenti dentro Tokyo e soprattutto in Akihabara, Shinjuku, Roppongi, Shibuya, Asakusa, etc...

Vi ho parlato solo del quartiere di Akihabara, tra i più interessanti relativamente all'elettronica, ma ovviamente, sebbene estremamente assortito, non è l'unico, inoltre detiene il maggior grado di setacciamento dei prodotti da parte di noi retrogamer provenienti da tutto il mondo. Prossimamente parlerò anche di altri quartieri dell'elettronica che vendono prodotti dissimili da Akihabara.

Sicuramente questo quartiere non è uno tra i più belli da fotografare e non rappresenta neppure la priorità del turista occidentale medio. Ogni quartiere ha le proprie specifiche di marketing, dalla moda più esclusiva fino agli oggetti più pacchiani disseminati lungo quartieri anonimi e mediocri.

Nella prossima puntata vi parlerò di due prodigiose ditte dell'usato BookOff e HardOff, lontane dai centri turistici, dove potrete organizzare un viaggio in Giappone unicamente per girovagare in queste due "catene della speranza" disseminate lungo tutto il Giappone.

Infine, scherzosamente, ma non troppo... nel frattempo vi consiglio di frequentare un corso accelerato di filosofia ed acquisire la capacità di manipolare la mente come uno Jedi: vi sarà discretamente utile quando alla dogana sarete pescati a campione dalla Polizia Doganale... visto che il detector di densitometria e soprattutto il metal-detector inizieranno ad urlare a squarciagola, scoppiettando tra mille scintille ed i doganieri vi chiameranno a rapporto strofinando le mani... fiutando un salato verbale!

A presto amici.

Figura 19

SFORMATI VIDEO 2: MALEDETTI NASTRI

a cura di Associazione Firenze Vintage Bit Onlus (Federico Gori)

Il progresso tecnologico, come ben sappiamo, non si limita al solo campo dell'informatica, ma copre una vastissima gamma di argomenti. *Uno dei settori più interessanti da riscoprire è quello legato allo sviluppo dei vari formati di riproduzione audio-video, che da sempre stimola la creatività e l'impegno di un gran numero di addetti ai lavori.
Oggi siamo ormai abituati alle piattaforme di streaming, dove possiamo scegliere un film da un catalogo predefinito, con tutta la libertà (ma anche i limiti) del caso, oppure affidarci all'alta qualità del Blu-ray o a quella ancora buona dell'inossidabile DVD. Un tempo però le cose erano ben diverse. Ci sono voluti decenni per avere la possibilità di vedere materiale registrato sul nostro televisore di casa e le cose non sono andate sempre bene per chi ha provato ad imporsi sul mercato.

Durante la prima serata, tenutasi giovedì 23 Febbraio 2018, abbiamo approfondito alcuni formati a disco; nello specifico il celebre Laserdisc, il molto meno celebre CED ed infine il Video CD. Ognuno di questi ha caratteristiche particolari, ma nessuno è riuscito ad intaccare il predominio sul mercato delle VHS.
Come ben sappiamo infatti, la videocassetta ha letteralmente spopolato, diventando per più di un ventennio il formato preferito dal pubblico, sia per la registrazione casalinga che per la visione di film su nastro. Presentata dalla JVC nel 1976, fin da subito ha saputo imporsi come standard, grazie alla possibilità di registrare programmi e film in televisione, in modo da poterli poi riguardare a piacimento. Successivamente, con l'avvento della vendita dei film su nastro, il mercato si è ulteriormente allargato, facendo crescere in maniera esponenziale tutto l'ambiente (basta pensare alla creazione dei videonoleggi, nonché negozi specifici per la vendita).
Ma la VHS era quindi l'unica concorrente su nastro? Assolutamente no, come infatti abbiamo visto nella seconda serata di "Sformati Video", che si è tenuta Giovedì 18 Aprile 2019.
Chi ha vissuto quel periodo sa bene che ci fu una lotta senza quartiere per il predominio sul mercato. Sappiamo già chi è il vincitore, ma non conosciamo ancora il nome del rivale; si tratta del Betamax della Sony. Creato nel 1975, il Betamax è stato il primo vero tentativo di poter fornire un formato affidabile per la registrazione casalinga. Dopo alcuni esperimenti di inizio anni '70 (tra cui il VCR di Philips), Sony scende in campo con la sua proposta e presenta un formato ben costruito e affidabile, con l'intenzione di farlo diventare lo standard. Il nome "beta" ha un doppio scopo: è infatti sia la parola giapponese utilizzata per descrivere il modo in cui i segnali vengono registrati sul nastro, sia un rimando alla lettera greca, la cui forma ricorda la posizione che assume il nastro all'interno del videoregistratore. La parte finale del nome, "max", viene invece aggiunta per dargli un tono di grandezza.

Pur arrivando prima sul mercato, il Betamax vede subito una difficile strada davanti a sé. Gli stretti controlli imposti da Sony per la produzione dei videoregistratori infatti, portano altre grandi compagnie a prendere strade autonome, compresa JVC, che poi svilupperà la videocassetta.
Lo scontro tra Betamax e VHS ha inizio subito dopo l'arrivo sul mercato di quest'ultima. La lotta per i primi anni è senza esclusione

di colpi. Sony può contare su una qualità video maggiore, una superiore fedeltà audio ed un'immagine più stabile, nonché una qualità di costruzione dei videoregistratori impareggiabile. Tutto questo però si scontra con il costo dei lettori stessi, molto più alto della concorrenza, nonché la durata esigua del nastro, che all'inizio non supera un'ora di registrazione (contro le due del VHS). Sony inizierà ad affidare la costruzione dei videoregistratori a terze parti solo all'inizio degli anni '80 e creerà nello stesso periodo cassette molto più capienti, ma la lotta a quel punto sarà già stata decisa.

Il colosso giapponese cercherà in ogni modo di migliorare e variare la tecnologia Beta, offrendo nel 1983 l'audio stereo hi-fi, creando telecamere che utilizzano le cassette per registrare (cosa che verrà poi fatta anche con il VHS) e creando formati successivi di alta qualità come SuperBeta (1985) ed ED Beta (1988), senza tuttavia recuperare mai il terreno perso. A tutto questo si aggiunge anche la beffa di un lunghissimo processo in America, soprannominato "Caso Betamax", intentato a Sony da Universal Studios, con altre case cinematografiche a supporto (tra cui la stessa Walt Disney). Essendo il primo formato di videoregistrazione ad uscire, infatti, è malvisto dato che permette di registrare film dalla televisione in maniera totalmente gratuita. La causa ha inizio nel 1976 e si concluderà definitivamente con la vittoria di Sony solo nel 1984.

Già dalla fine degli anni '80 il Betamax uscirà di scena, in quanto la stessa Sony si concentrerà sulla produzione di videoregistratori VHS. C'è però da dire che l'ultimo lettore è stato prodotto dalla casa giapponese nel 2002 e che i nastri sono stati prodotti dalla ufficialmente fino al 2016 (altre compagnie li producono ancora), per cui in realtà una piccola parte del mercato è rimasta comunque fedele al progetto Sony. Leggende urbane raccontano anche che nella scelta della VHS come formato principale ci sia stata anche la grande influenza del mondo della pornografia, anche se ad oggi non è del tutto certo (in quanto i film per adulti sono comunque presenti, anche se in quantità minore, su Betamax).

Dopo aver parlato delle caratteristiche chiave di formato e lettori, la conferenza si sposta sul Video 2000 di Philips, un formato alternativo dalle caratteristiche tecniche molto avanzate, che ebbe la sfortuna di arrivare sul mercato solo nel 1980 (e solo in Europa), quando ormai la lotta tra VHS e Betamax era al culmine. Il V2000 nasce come evoluzione del VCR, messo in commercio da Philips all'inizio degli anni '70 e presenta una cassetta leggermente più grande di una VHS, con due lati speculari (come le musicassette).
Questo permette una impareggiabile durata di registrazione, che nel tempo arriva fino ad otto ore (quattro per lato) e addirittura sedici in modalità long play. Il sistema di puntamento inoltre risulta particolarmente efficiente, grazie ad un particolare dispositivo piezoelettrico che segue la traccia durante la lettura. Questo comporta

una stabilità perfetta dell'immagine anche durante la pausa o la lettura veloce, cosa impossibile con una normale VHS. Il V2000 prevede inoltre, su alcuni modelli, la possibilità di riavvolgere il nastro fino ad un punto preciso impostato tramite numero sul contatore.
Nonostante i grandi traguardi tecnici, purtroppo, il formato non riesce ad imporsi in Europa, complice il ritardo con cui arriva sul mercato. Già nel 1986 Philips dichiara che si concentrerà sui videoregistratori VHS ed il Video 2000 uscirà di scena definitivamente nel 1989. Dopo questa escursione nel "terzo formato in una guerra a due" (come fu definito il V2000), la conferenza si conclude con un approfondimento sul sistema D-Theater.
Nonostante il successo incredibile del VHS, infatti, JVC cerca sin dalla fine degli anni '80 di creare formati a nastro più performanti. Il primo risultato è l'S-VHS del 1987, che porta la risoluzione da 240 linee della videocassetta a 420, ma che riceve una diffusione limitata.
Il passo successivo è il Digital VHS del 1998, sviluppato in compartecipazione con Hitachi, Matsushita e Philips, che permette una registrazione digitale in alta definizione su nastro, con una qualità impareggiabile per l'epoca. I nastri possono contenere una quantità di dati che varia da 25 a ben 50 GB.
I costi sono comunque molto alti e il tutto rimane esclusivo appannaggio dell'ambiente professionale.
Nel 2002, in America e Giappone, viene presentato il D-Theater, ovvero un sistema D-VHS che prevede la vendita di film in alta definizione su nastro, da collegare al televisore con la presa component.
Anche se la cassetta rimane una D-VHS, i film registrati in questo formato non possono essere letti da un normale lettore Digital VHS, ma necessitano di una linea apposita. I film sono registrati in 720p e 1080i e offrono tracce audio multiple con supporto per Dolby Digital e DTS.
Nonostante l'alta qualità audio video e il supporto di case di produzione come 20th Century Fox, Dreamworks ed Universal Pictures, il formato ha vita breve e la produzione si interrompe alla fine del 2004, quando già tutto il mondo si sta preparando all'avvento di un formato su disco in alta definizione.
Anche questo porterà ad una guerra tra supporti (HD-DVD vs Blu-ray), di cui parleremo nella terza serata di "Sformati Video", prevista ad inizio 2020, a cui vi invitiamo fin da ora.

L'Associazione Firenze Vintage Bit Onlus vi ricorda il prossimo appuntamento:

VIDEOGIOCHIAMO CON SEGA MASTER SYSTEM

La prima metà dell'anno si sta per concludere e la nostra associazione ha deciso di organizzare una serata rilassante dedicata interamente al Sega Master System. A differenza delle precedenti conferenze, però, stavolta lasceremo la parola ai controller. Ci sfideremo su vari titoli della celebre console ad 8 bit, festeggiando insieme l'arrivo dell'estate. La serata si terrà nell'attuale sede dell'associazione, in via della Croce 62 a San Mauro a Signa (FI). Ad attendervi troverete classici come Double Dragon, Gangster Town e Super Monaco GP, oltre a risate e divertimento.

Vi aspettiamo Giovedì 13 Giugno con "VIDEOGIOCHIAMO CON SEGA MASTER SYSTEM" con Federico Gori.

Tu chiamala se vuoi... RetroEstate

di Querino Ialongo

Nonostante questo tempo ci fa pensare ancora all'inverno, ormai la stagione estiva è alla porte e si moltiplicano, per noi appassionati di retrogames e retrocomputing, eventi, fiere e mercatini un pò in tutta Italia. In questo numero vi parliamo dei maggiori eventi del mese di giugno e luglio.

Il primo importante appuntamento che vi segnaliamo è lo **UPLAY** che si terrà a Pisa il 15 e 16 giugno, giunto addirittura alla sua dodicesima edizione.
Uplay Pisa è un innovativo festival sulle tematiche del videogioco, anime, manga, cosplay e cultura giapponese.
L'evento punta alla qualità più assoluta del prodotto offerto al pubblico con iniziative nuove e originali, che per certi versi lo portano a paragonare ad un "contenitore" di eventi e spettacoli davvero unici.
Uplay Pisa è quindi un mondo a parte per un pubblico esigente e di grande qualità. Il festival è organizzato da Uplay Eventi con l'aiuto dell'Associazione Culturale CUSplay PISA.
La parte commerciale e gli spettacoli si terranno all'interno del Palazzo dei congressi che si trova in via Giacomo Matteotti 1, in una zona molto centrale e di facile raggiungimento con tutti i mezzi pubblici o privati.
Inoltre è stato realizzato PRESS START; una nuova manifestazione completamente dedicata al retrogaming con postazioni di gioco, collezionismo, info point con la storia delle console e dei principali giochi e store retrogames completamente a disposizione del pubblico!
Un evento grandioso che nasce come fratello di Uplay Pisa e si svilupperà nel corso dei prossimi anni. In via eccezionale infatti l'ingresso a questa manifestazione sarà con un contributo libero e senza costi fissi e tutto l'eventuale ricavato verrà destinato dai ragazzi dell'organizzazione per lo sviluppo ed il potenziamento delle future edizioni!
Apice del festival sarà il concerto della mitica Cristina d'Avena che dopo 3 anni tornerà a Uplay Pisa con un nuovo grandissimo spettacolo il 16 giugno alle ore 17.
Dopo la performance della regina delle sigle dei cartoni animati, autografi e foto ricordo con tutto il pubblico!

Il secondo imperdibile evento che vi segnaliamo è **CASALE COMICS & GAMES** che si terrà a Casale Monferrato sempre il 15 e 16 giugno.

L'evento, giunto alla sua quarta edizione, è una grande fiera dedicata al fumetto, al gioco e al cosplay in cui i partecipanti si sentiranno sempre al centro.
Decine di stand, tantissimi ospiti, torrioni e sotterranei ricchi di giochi e sfide da affrontare, dall'area videogames ai workshop, dai concerti ai contest, dalle esibizioni agli incontri con gli ospiti. In particolare nell'area comics potrete non solo accaparrarvi le ultime

RETROEVENTI GIUGNO - LUGLIO 2019

UPLAY
PISA 15-16 GIUGNO

CASALE COMICS & GAMES
CASALE MONFERRATO
15-16 GIUGNO

novità, ma anche incontrare i vostri artisti preferiti, partecipare ai workshop e magari anche competere con altri aspiranti disegnatori in uno dei tanti contest a premi proposti.
Molto cospicua l'area gioco dove ci sarà la quarte edizione de LA ROCCA IMPLACABILE, spazio dedicato ai giochi da tavolo, gdr e cardgames vari con decine di stand con giochi in anteprima, demo, ospiti, tornei e dimostrazioni dal vivo, insomma un appuntamento assolutamente imperdibile per gli amanti del genere.
Inoltre grazie allo strepitoso lavoro dei ragazzi di VG Perspective, ci saranno più di cinquanta postazioni dedicate al mondo del videogioco tradizionale, con un occhio dedicato alle console di nuova generazione, tornei, sfide e uno spettacolare allestimento con Super Mario su un gamepad gigante, Donkey Kong

all'interno di una botte e tantissime altre sorprese!
Il 16 giugno torna al Parco Sterza di Sestri Levante il **PILA COMICS & GAMES**, l'evento dedicato principalmente al mondo del fumetto.
Comics, decine di stand espositivi, dimostrazioni di boardgames e wargames tridimensionali, giochi di ruolo, concerti, spettacoli, approfondimenti, truccabimbi e molto altro in questa intensa giornata.
Numerose saranno le novità tra le quali gli interventi di Carlo Chendi e Giovanni Talami. Inoltre i concerti de "I Ragazzi tra le stelle" (concerto a tema Disney) e gli "EXPEDOS"(nota cover band) senza dimenticare che per tutta la durata dell'evento ci saranno i contest per la miglior cover band e per i migliori cosplayers, entrambi nominati da una giuria d'eccezione e qualificata. Ricca anche l'area dedicata ai retrogames sia con diversi stand destinati all'esposizione di cabinati e sia con uno spazio dedicato all'organizzazione di tornei.
Ultimo evento che vi segnaliamo in ordine di tempo è il **BURTOMICS**

COMICS AND GAMES che si terrà il 12, 13 e 14 luglio nello splendido scenario medievale di Finalborgo, uno dei tre nuclei urbani formanti l'abitato di Finale Ligure, comune italiano in provincia di Savona.
Il festival è diventato ormai un appuntamento fisso e il suo cuore sarà un'area dedicata ai comics con workshop e stand espositivi.
Non mancherà la tradizionale gara cosplay, mentre grazie all'associazione culturale Sintesi e i Bagni Nautico Finale Ligure ci saranno a disposizione dei cosplayer splendidi photoset gratuiti e professionali.
Presente anche un'area retro games dedicata ai giochi anni 80/90 con uno sguardo attento ai titoli giapponesi più famosi.
Saranno poi esposte diverse console vintage (tra le quali nes, supernes, gameboy, atari, commodore), simulatori e alcune postazione per fortnite, uno dei titoli più amati del momento.
Inoltre gli organizzatori hanno promesso che dal prossimo anno sarà prevista un'intera area dedicata anche ai cabinati.

SE VOLETE UNA RECENSIONE PER IL VOSTRO EVENTO CONTATTATECI:

RETROMAGAZINE.REDAZIONE@GMAIL.COM

PILA COMICS & GAMES
SESTRI LEVANTE 16 GIUGNO

BURTOMICS
COMICS AND GAMES
FINALE LIGURE 12-13-14 LUGLIO

E non finisce qui...

di Marco Pistorio

E siamo giunti così alla fine di questo numero 15 di RetroMagazine, numero che vi abbiamo presentato con un look tutto nuovo e che si compone di ben 78 pagine che spaziano, come di solito, su molteplici argomenti.

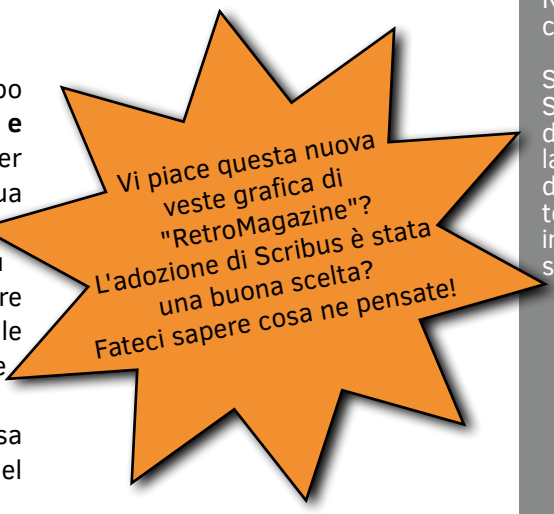
Tutto ciò con l'evidente intento di offrirvi una rivista più accattivante dal punto di vista visivo ma anche, anzi soprattutto, ricca di contenuti. Fateci sapere se questo numero vi è piaciuto.
Diteci cosa vi piacerebbe trovare che ancora non trattiamo.
RetroMagazine potrà crescere grazie anche al feedback di voi lettori, feedback che non dovrà mai venir meno.
Comunicateci le vostre sensazioni, le vostre idee, e fatevi avanti anche se intendete collaborare in qualsiasi forma con noi, anche saltuariamente. C'è tanto da fare, garantito!

Ringraziamo gli amici del gruppo "**Retroprogramming Italia 8 bit e oltre - Associazione culturale**" per la sempre più solida e proficua collaborazione con la nostra fanzine e ci auguriamo, anche su questo fronte, che si riesca ad essere sempre più sinergici, in maniera tale da poter fare insieme sempre più e sempre meglio.
L'idea del Basic Contest è qualcosa che effettivamente mancava nel panorama del retrocomputing italiano e la professionalità e la dedizione con i quali il contest viene gestito mensilmente sono encomiabili. La risposta dell'utenza del gruppo è stata all'altezza della situazione e noi come rivista ufficiale siamo lieti di ospitare tra le nostre pagine il risultato di tanto impegno.

Un altro sincero ringraziamento va a tutti i gruppi su Facebook che ci permettono di pubblicizzare le uscite della fanzine.

Restate sintonizzati, amici lettori! Un saluto a tutti voi dalla redazione di RetroMagazine, in attesa dell'uscita del prossimo numero della vostra "retrorivista" preferita, prima di mettere i remi in barca e concederci una meritata pausa estiva.

Ciao a tutti!



Disclaimers

RetroMagazine
Anno 3 - Numero 15

Direttore Responsabile
Francesco Fiorentini

Vice Direttore
Marco Pistorio

Giugno 2019