RetroMagazine

semplicemente retro

Numero 14 - Anno 3 - Aprile 2019 - WWW.RETROMAGAZINE.NET - Pubblicazione gratuita

Giappone seconda puntata:
l'occidentale limbo dei quotidiani déjà-vu

Katana Soul: Samurai e Ninja in stile C=64

Emulare che passione: il MiSTer

Come calcolarne la data
e buona Pasqua da tutta la redazione

Le origini di un mito: Double Dragon!

RetroMagazine
Anno 3 - Numero 14

Hanno collaborato a questo numero:
- Giuseppe Fedele
- Juri Fossaroli
- Marco Petri
- Vintage Computer Club Italia
- Daniele Brahimi
- Alberto Apostolo
- Starfox Mulder
- Fabrizio Caruso
- The Ancient One (OldGamesItalia)
- Ermanno Betori
- Leonardo Giordani
- Robert Swiderski (FVB)
- Andras Vajda (8BRPI)
- Querino Ialongo
- Edoardo Ullo
- David La Monaca (aka Cercamon)
- Michele Ugolini
- Marco Pistorio
- Francesco Fiorentini

Immagine di copertina: Flavio Soldani

APRILE 2019 - WWW.RETROMAGAZINE.NET

IN EVIDENZA IN QUESTO NUMERO

I guerrieri della domenica

di Alberto Apostolo e Francesco Fiorentini

"Week-end warriors", "guerrieri della domenica" è l'appellativo con il quale i militari di carriera statunitensi chiamano quelli della Guardia Nazionale che compiono esercitazioni nei week-end mentre per il resto della settimana fanno vita da civili e vanno normalmente al lavoro.

Forse anche noi di RetroMagazine siamo considerati un po' dei "guerrieri della domenica" perché spendiamo parte del nostro tempo libero per documentarci e scrivere qualcosa che sia interessante anche per i lettori che scaricano liberamente RM.

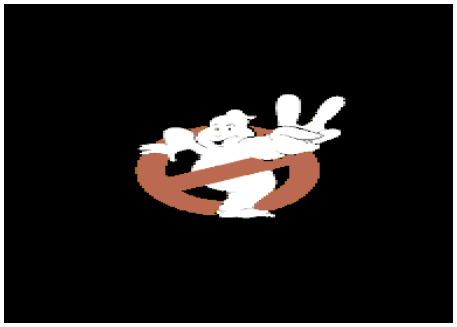
Sabato 23 e **domenica 24 marzo 2019,** ci siamo ritrovati a San Cesareo (Roma) per incontrarci finalmente di persona e non più soltanto tramite i social o la posta elettronica.

Non sono state le "Vacanze Romane" cantate dai Matia Bazar tanti anni fa, ma un intenso week-end e non c'era "l'oro e l'argento delle sale da tè" ma l'oro e l'argento delle idee di tutti noi che abbiamo condiviso per rendere RM una cosa migliore.

Stavo per dire "una cosa seria" ma RM lo è di già perché, nonostante sia e resterà una rivista amatoriale, per confezionarla ci mettiamo la serietà con cui pratichiamo le nostre rispettive professioni dal lunedì al venerdì (che non sono quelle di scrittore, grafico, editore, direttore di testata). Nei prossimi numeri, i lettori di RM avranno modo di apprezzare (e anche criticare, perché no?) gli sviluppi delle decisioni prese di comune accordo e che il nostro direttore Francesco ha attentamente raccolto.

Nella foto sono presenti: Francesco Fiorentini, Giorgio Balestrieri, Michele Ugolini, Ermanno Betori, Marco Pistorio, Alberto Apostolo, Querino Ialongo e Leonardo Vettori. Purtroppo manca Starfox Mulder che, nonostante fosse intervenuto all'evento, era gia' dovuto andare via per prendere il treno quando abbiamo scattato la foto (presente invece nella foto sotto; a voi scovarlo...).

In questo numero, che esce in coincidenza con la Pasqua, non ci siamo fatti mancare niente: HW, software, interviste e persino le uova. Non ci credete? Leggete l'articolo di fondo! ☺



La grafica hi-res con il C64

La seconda parte dell'articolo di **Marco Pistorio** su come generare grafica in alta risoluzione e multicolor con il Commodore 64.

Articolo a pagina 23

La sala giochi...

Un articolo tutto da leggere con l'autorevole punto di vista dello psicologo **Marco Petri,** che ha analizzato retroattivamente il valore sociale delle sale giochi.

Articolo a pagina 43

Il MiSTer

di Juri Fossaroli

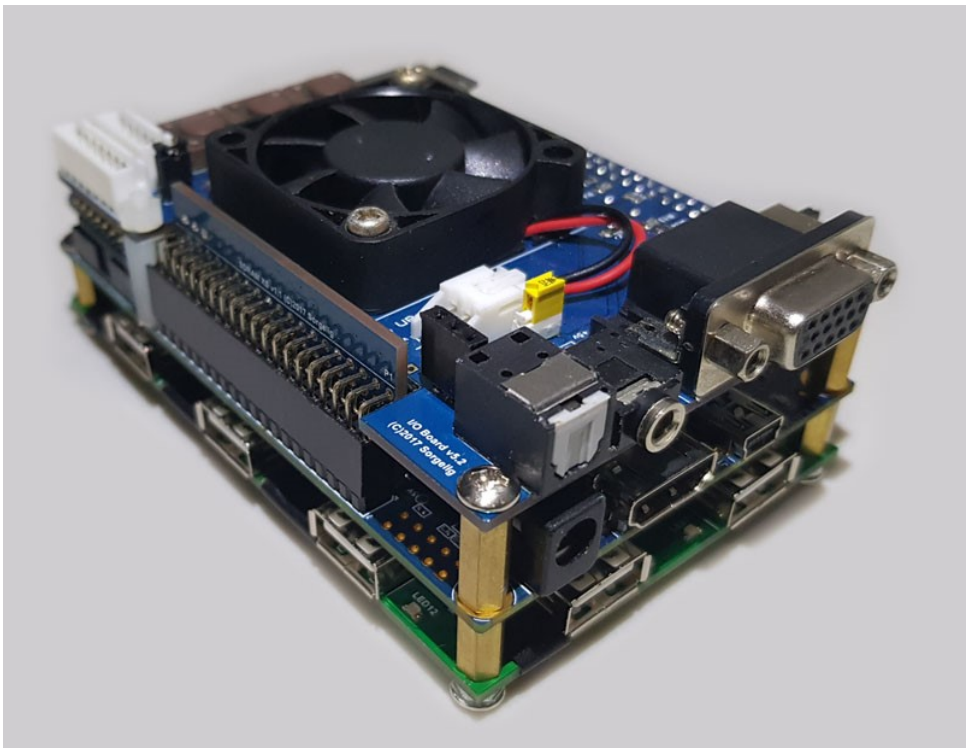
Continua la nostra avventura alla scoperta delle FPGA. Questa volta e' il turno di **Juri** di presentarci una macchina che sicuramente fara' venire l'acquolina in bocca a tanti retro-appassionati.

Personalmente, oltre a possedere gia' un MIST, sto seriamente prendendo in considerazione di acquistare anche questa, soprattutto dopo aver letto questo articolo. E voi? NdFF

Nello scorso numero il buon Roberto Lari vi ha parlato molto dettagliatamente dei sistemi Mist e Mistica, oggi io vi parlerò della "naturale" evoluzione di tali sistemi, signori, signore, vi presento il MiSTer.

Il prototipo del MiSTer

Ma cos'è esattamente il MiSTer e come nasce? Il MiSTer è un altro sistema basato su FPGA+ARM proprio come il Mist, esso nasce dalla mente di Alexey Melnikov (conosciuto come Sorgelig su Github) che, dopo aver creato e migliorato alcuni core presenti su Mist e dopo aver implementato l'uscita YPbPr per il Mist, ha preso contatto con il suo ideatore per progettare una sorta di Mist 2. In particolar modo era interessato ad implementare un'uscita HDMI da utilizzare con i moderni tv/monitor completamente assente sul vecchio sistema. Purtroppo ne è uscito un nulla di fatto ed Alexey dopo essersi rimboccato le maniche ha deciso di fare tutto da solo! Ha ideato da zero il sistema, scegliendo come base di sviluppo la scheda DE-10 Nano dell'azienda Terasic, da qua il nome MISTER (MISt + TERasic), ha inoltre pensato bene di rendere il tutto Open Source, infine creato una dettagliata documentazione ed importato i principali core del Mist! Questo



Il MiSTer in tutto il suo splendore!

ha attratto molti sviluppatori hardware e software che hanno iniziato a lavorare al progetto.

Il MiSTer si presenta principalmente come un sistema modulare, a differenza del Mist non è una board unica dentro un bel case metallico, ma è un insieme di schede spesso "a nudo" o racchiuso dentro un case stampato in 3D. Ma perché tutto ciò? Purtroppo per motivi sconosciuti (o magari conosciuti per qualcuno di voi più esperto di marketing) il solo chip FPGA utilizzato nella board di sviluppo Terasic DE-10 Nano costa da solo quasi il DOPPIO della board stessa! Non è incredibile?

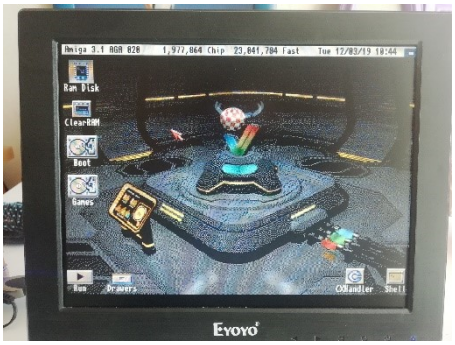
Già da sola la scheda base che include una memoria DDR3 da 1Gb, una scheda LAN, una porta micro USB, uno slot per MicroSD ed un' uscita HDMI, può eseguire molti core compreso il core AO486 che simula un computer 486 con tanto di scheda di rete e Soundblaster 2.0 in grado di far girare persino Windows 95. Ha inoltre a disposizione una porta mini usb utilizzabile come porta seriale.

I core che necessitano di una scheda di memoria SDRAM aggiuntiva da 32Mb sono: Amiga, Amstrad CPC, Archimedes, Colecovision, Commodore 64, Gameboy, Macintosh, MSX, Nes, Sinclair QL, Sega Master System, Super Nes, ZX Spectrum, Tsconf, X68000, Sam coupe, BK0011M ed Atari 800 (solo se si usano le cartuccie o una quantità di memoria maggiore di 320kb).

Avere la memoria aggiuntiva in un modulo separato può essere un vantaggio, nel momento in cui vi scrivo sta venendo sviluppato il core NeoGeo e potrebbe richiedere una memoria maggiore di 32Mb, in questo caso basterà solamente sostituire il solo modulo ram anziché l'intero sistema.

Il Core C64 connesso alla BBS

I più attenti si staranno chiedendo: ma la board base non ha 1Gb di memoria DDR3? Esatto ma c'è un però, le temporizzazioni di quella memoria non sono compatibili con i retro sistemi simulati, è necessaria una memoria più adatta anche se, per quanto riguarda il core NeoGeo, stanno cercando un sistema per sfruttare la memoria DDR3 integrata anziché sviluppare un ulteriore modulo SDRAM da 64Mb o 128Mb. Esiste anche un altro core Amiga (in fase di beta) che sfrutta 256Mb di ram DDR3 rendendola disponibile dentro l'ambiente Amiga.



Il core Amiga

Oltre alla scheda ram sono disponibili ulteriori schede aggiuntive, come la scheda IO che aggiunge una uscita VGA a identica a quella del Mist a 18bit che supporta gli stessi segnali, un' uscita audio stereo su jack 3.5mm, una uscita audio digitale Spdif, un connettore usb3 che funge da espansione, un secondo slot per schede MicroSD, una ventola, tre led e tre comodi pulsanti. Con l'uscita VGA di questa scheda si può usare lo stesso cavo video da VGA a SCART del Mist.

È anche disponibile una scheda RTC sfruttata al momento dai soli core Amiga, 486 ed Archimedes che mantiene l'orario anche a dispositivo spento. Infine è disponibile una scheda HUB USB a 7 porte (e' possibile collegare comunque un comune hub OTG USB, di quelli che vengono venduti per i cellulari).

Vi è un'ulteriore scheda chiamata AIO (All In One) che unisce in un unica scheda la memoria da 32Mb e la scheda IO, a cui manca pero il connettore di espansione.

Non vi sono porte DB9 al momento, si può ovviare utilizzando i vari convertitori DB9->USB in circolazione. È in sviluppo una ulteriore scheda che aggiungerà le porte DB9 più altre porte per vari retro controller.

Ma in definitiva, cosa fa? Se è una evoluzione del Mist avrà pure qualche asso nella manica giusto? Esatto!

L'FPGA da solo essendo 5,5 volte più "grande" è in grado di simulare core più complessi, come il già citato AO486 (non disponibile per Mist).

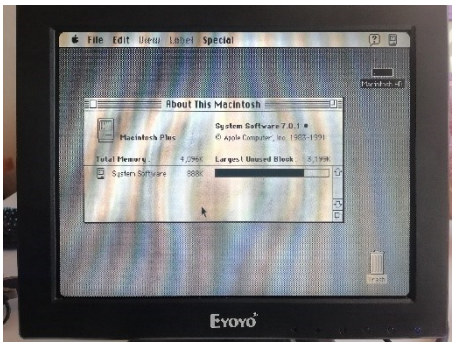
Il core AO486, prerogativa del MiSTer!

La parte ARM invece, a differenza del Mist che ha un firmware dedicato, è gestita da Linux. Mi sembra già di sentire un coro di "Noooooooo"! Molti di voi penseranno già ai grattacapi, reduci magari da brutte esperienze con Raspberry Pi od Odroid vari! Niente di tutto ciò! Il sistema Linux è ridotto davvero all'osso: si avvia in un paio di secondi, giusto il tempo necessario al TV/Monitor di agganciare il segnale HDMi e soprattutto, NON NECESSITA di alcuna manutenzione da parte dell'utente finale. Tutte le configurazioni necessarie avvengono utilizzando un file di testo (Mister.ini) proprio come nel Mist.

Le poche configurazioni che all'inizio del progetto necessitavano di essere "manipolate" entrando dentro al sistema, sono state spostate in comodissimi scripts, eseguibili direttamente dal menu principale!

Avere Linux a disposizione ha anche i suoi bei vantaggi, primo tra tutti il filesystem della MicroSD che da Fat32 passa ad exFAT, in questo modo non esistono più limiti di grandezza per i file contenuti, il Mist infatti non supporta file più grandi di 4Gb. Non è un problema per quanto riguarda i core più semplici in quanto i vari programmi o giochi non occupano più di qualche decina di kilobytes. Il problema viene fuori quando si utilizzano i file HDF per Amiga per esempio, non si possono avere harddisk più grandi di 4Gb. Questo problema non esiste più sul MiSTer!

La parte Linux inoltre implementa una emulazione di una Roland MT-32, collegabile virtualmente ad alcuni core tramite una connessione MIDI interna.. senza stendere nessun cavo!



Il core Macintosh

Viene emulato, sempre da Linux, un modem per poter collegare ad internet o alle bbs i vari core che lo supportano, sfruttando la porta ethernet e non solo! Sempre grazie a Linux sono supportate alcune "chiavette" wifi, ed è quindi possibile collegarsi ad internet senza utilizzare cavi. Al momento che scrivo sono supportate le seguenti chiavette wifi: "ASUS USB AC53 nano rev A1", "D-Link DWA-171 HWVer: A1" ed "Edimax EW-7811UN". Altre chiavette basate sugli stessi identici chipset dovrebbero funzionare ugualmente, io per esempio ne ho una cinese con chipset RTL8811AU che comprai a suo tempo per il mio Odroid XU4 e funziona molto bene.

Non solo, una volta connesso il MiSter alla wifi, e' possibile collegarsi tramite software FTP o direttamente tramite windows in quanto sono attivi i protocolli FTP e Samba.

Per i più audaci e per chi non ha paura di fare casini con Linux, è possibile inoltre accedere dall'interno utilizzando un core (Amiga o 486) con un software terminale, al sistema Linux, per poter fare tranquillamente "danni" senza usare nessun PC esterno. :D

Un'altra funzione davvero, ma davvero molto comoda, che sfrutta la connessione ad internet, è l'autoaggiornamento. Dal menù principale è possibile eseguire uno script che si collega ad internet e ricerca la presenza di core e firmware più recenti, li scarica e li installa da solo! Non c'è più quindi bisogno di

andare ogni volta a controllare se sono usciti aggiornamenti, scaricarli e copiarli sulla sd card.

Anche l'installazione è resa il più facile possibile, attraverso un piccolo programma di installazione, la MicroSD card viene preparata e caricata con tutto l'occorrente: Linux, lo "pseudo" firmware, ed il core Menu (con cui è possibile scaricare tutti gli altri core). A questo punto mancano solo i vari file richiesti dai core; giochi applicazioni e quant'altro.

Come ho scritto prima non ci sono le porte DB9 a cui poter collegare direttamente i vari retro Joystick, la scelta può essere più o meno opinabile, il Joystick può essere simulato da tastiera, sicuramente scomodo, si può collegare un Joypad usb o anche un Joypad wireless utilizzando il suo ricevitore ma si possono anche utilizzare quei convertitori da DB9 a USB presenti un po ovunque. Si può aspettare l'uscita della scheda LL CoolJoy che oltre alle porte DB9 a cui collegare i nostri amati Joystick ha pure le porte DB15 a cui collegare i controller NeoGeo. Ma esiste un'ulteriore possibilità; e cioè che qualcuno di voi si unisca alla squadra e sviluppi qualcosa per il MiSTer :)

Per quanto riguarda il video attraverso l'uscita HDMI c'è da dire che sono stati utilizzati diversi accorgimenti per evitare stuttering vari, un'opzione in particolare (vsync_adjust) permette di generare la stessa frequenza verticale originale del sistema simulato, anziché una frequenza standard. Alcuni vecchi sistemi, diciamo tutti praticamente, non generano una frequenza esatta a 50Hz o 60Hz, ma molto spesso sono leggermente differenti, personalmente ho provato questa opzione su un paio di Samsung (un vecchio tv al plasma, un monitor/tv SyncMaster T260HD) e su un economico monitor cinese Eyoyo da 10". Tutti quanti hanno funzionato perfettamente e gli scroll erano lisci e senza scatti. Invece, per quanto riguarda l'audio digitale attraverso l'HDMI, è possibile selezionare una codifica a 96khz/16bit per i tv che lo supportano e per una resa audio migliore, al posto della codifica standard a 48khz/16bit.

Principali caratteristiche del MiSTer:

*Intel Cyclone® V SoC 5CSEBA6U23I7 FPGA con 110,000LE e 5,570Kbit di Block RAM.

*ARM Cortex A9 dual-core CPU funzionante a 800MHz integrato all'interno dell'FPGA

*1GB (2x256Mx16) DDR3 SDRAM integrata nella board principale

*Uscita audio/video HDMI a 24bit

Schede aggiuntive principali:

*SDRAM con 256Mbit a 166Mhz (32Mb)

*I/O con VGA a 18bit (identica al Mist), TOSLink, connettore audio da 3.5mm, connettore di espansione, MicroSD secondaria, ventola, 3 led e 3 pulsanti

*RTC orologio con batteria tampone

*HUB 7 porte basato sul chip FE2.1 ed alimentato dalla board principale o esternamente

*AIO racchiude in un unica scheda le board I/O e SDRAM. Non è presente la porta di espansione

Links utili

Per il funzionamento e la configurazione dei vari core (che è praticamente identica a quella del Mist) vi rimando alla wiki ufficiale in inglese del progetto Mister, raggiungibile al seguente indirizzo:
https://github.com/MiSTer-devel/Main_MiSTer/wiki

C'è inoltre un' intera sezione su Atari Forum, in inglese, dedicata al Mister, con varie sottosezioni dedicate alle varie board ed ai core disponibili. Potete raggiungerla a questo indirizzo:
http://www.atari-forum.com/viewforum.php?f=117

Per i patiti di Facebook c'è ovviamente anche un gruppo ufficiale sempre in lingua inglese:
https://www.facebook.com/groups/1870135613286506/

Il menu' di auto-aggiornamento del MiSTer

RetroMath:

Algoritmi base per la localizzazione

di Giuseppe Fedele

La localizzazione di persone e beni è ormai diventata una esigenza fondamentale in diversi contesti: pubblicità mobile, sicurezza, ospedali, ecc. Ed infatti, i servizi basati sulla localizzazione stanno diventando una parte indispensabile nella vita di tutti i giorni.

Per le applicazioni outdoor, lo standard tecnologico ormai consolidato è il sistema Global Positioning System (GPS).

Il GPS è un sistema di navigazione globale basato su satelliti; tale tecnologia consente di ottenere un'informazione precisa circa la posizione e l'orario al quale un determinato ricevitore effettua la richiesta di localizzazione. Tutto ciò è possibile grazie ad una rete satellitare in funzione ad ogni ora del giorno e in ogni condizione climatica; la localizzazione pertanto è attendibile a patto che il ricevitore riesca a captare il segnale da almeno quattro satelliti distinti.

Il sistema GPS viene utilizzato principalmente per applicazioni basate sulla posizione esterna, perché la sua precisione è notevolmente ridotta in ambienti indoor a causa del deterioramento della ricezione dei segnali dai satelliti e degli effetti delle riflessioni multiple.

In questi ambienti è necessario disporre di altri strumenti che permettano di riprodurre dei segnali utili ai fini della localizzazione.

Le tecniche attualmente più utilizzate sfruttano tecnologie come: infrarossi (IR), Bluetooth, identificazione a radio frequenza, ultrasuoni, riconoscimento mediante tracciamento ottico, tecniche basate sulla stima della potenza del segnale (RSSI), ecc.

Scopo di questo articolo è quello di presentare e simulare alcuni semplici algoritmi utilizzati nell'ambito della localizzazione.

Un sistema di localizzazione è un sistema che riesce a determinare, in tempo reale, la posizione di qualcosa o qualcuno in uno spazio fisico. Un tipico scenario di tracciamento è costituito da una serie di sensori di riferimento, la cui posizione è nota nell'ambiente di interesse, e di un dispositivo mobile (uno smartphone o un PDA) la cui posizione è da identificare.

L'equivalente della rete di satelliti utilizzata dal sistema GPS per ottenere una stima della posizione di un oggetto in ambienti outdoor è costituita, in ambienti indoor, da una rete di sensori (Wireless Sensor Networks – WSN) posizionati opportunamente nell'ambiente di interesse. Per ogni sensore, che rappresenta un nodo di questa rete, può essere nota la sua posizione all'interno dell'ambiente. In questo caso i nodi di riferimento sono chiamati Anchors (ancore) e la rete viene chiamata Anchor Based contrapposta al caso Anchor Free in cui i nodi non conoscono la propria posizione.

Supponiamo quindi di essere in un ambiente in cui è presente una WSN e di essere dotati di un dispositivo capace di dialogare con i nodi della rete. Negli algoritmi di localizzazione sono identificabili tre fasi:

- Ranging: attraverso opportune misurazioni effettuate dal nostro dispositivo vengono stimate le distanze dai nodi della rete (raggiungibili dal dispositivo);
- Positioning: una volta determinate le distanze il dispositivo può determinare la posizione con una certa posizione;
- Refinement: l'accuratezza della posizione viene migliorata attraverso una procedura iterativa.

Per stimare la distanza tra il dispositivo mobile ed un nodo della rete possono essere usate diverse tecniche. La più intuitiva fa uso delle misurazioni temporali dei segnali radio. Queste misurazioni temporali sono quindi convertite in misure di distanza conoscendo la velocità dei segnali in gioco.

Se d_i è la distanza dalla i-sima ancora e c è la velocità di propagazione del segnale ($c = 299792.458\ Km/s$) allora il tempo di volo (**Time of Flight**) è definito da

$$T_{f,i} = \frac{d_i}{c}$$

La principale difficoltà dei sistemi basati sul tempo di volo è proprio la forte sensibilità della soluzione ottenuta rispetto alle misure temporali.

Per mitigare in parte questi errori si ricorre a questo espediente (Figura 1).

A trasmette il segnale al tempo t_1. Il segnale arriva a B al tempo t_2. B elabora il segnale e dopo un tempo T_d risponde ad A. In questo modo il tempo di volo viene calcolato come media dei due tempi di volo:

$$T_{f,i} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}$$

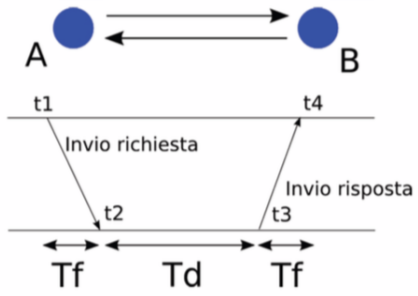


Figura 1. Doppio tempo di volo.

Un altro modo usato per stimare la distanza da un'ancora è valutare l'intensità del segnale ricevuto. Nel caso delle WSN si utilizza l'RSSI (**Received Signal Strenght Indicator**) che indica una stima della potenza del segnale ricevuto. Senza entrare nei dettagli (gli interessati possono fare riferimento a [1]), si può utilizzare la seguente relazione:

$$RSSI = -(10 \cdot n \cdot log_{10} d - P_r)$$

dove

- P_r è la potenza del segnale ricevuto espresso in dBm (decibel milliwatt $P[dBm] = 10 \cdot \log_{10}(P[W] \cdot 10^3)$) alla distanza di riferimento di un metro;
- n è la costante di propagazione del segnale.

Dalla relazione precedente si ricava la distanza come

$$d = 10^{\left(\frac{P_r - RSSI}{10 \cdot n}\right)}$$

In ambiente aperto i valori dei parametri sono $n = 2, P_r = -40dBm$, per cui la distanza ideale in funzione della potenza ricevuta è rappresentata dal grafico in Figura 2.

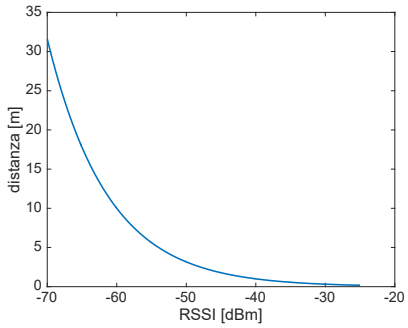


Figura 2. Distanza in funzione dell'RSSI $(\boldsymbol{n = 2, P_r = 10dBm})$.

Una volta stimate le distanze tra il dispositivo mobile e le ancore fisse, come facciamo a stimare la posizione del dispositivo nell'ambiente?

Descriviamo di seguito alcuni algoritmi di localizzazione molto semplici e facilmente implementabili.

Algoritmo Min-Max [2]

Questo algoritmo considera dei quadrati costruiti opportunamente per ogni ancora. Ogni punto di riferimento è al centro di un quadrato il cui lato è il doppio della distanza rilevata dal dispositivo mobile.

In Figura 3 i pallini di colore rosso, blu e arancione rappresentano le tre ancore presenti nell'ambiente, mentre il pallino verde il dispositivo mobile da localizzare. L'algoritmo calcola inizialmente le distanze tra il dispositivo mobile e le tre ancore (d_1, d_2, d_3) e procede costruendo un quadrato centrato su ogni ancora e di lato pari a $2 \cdot d_i, i = 1,2,3$. L'intersezione dei tre quadrati (quadrilatero nero in figura) fornisce una stima della posizione del dispositivo mobile.

Gli estremi del quadrilatero possono essere calcolati prendendo i massimi e i minimi di tutte le coordinate:

$$[\max(x_i - d_i), \max(y_i - d_i)] \times [\min(x_i + d_i), \min(y_i + d_i)]$$

dove (x_i, y_i) rappresenta la posizione dell'ancora *i-sima.*

E' ovvio aspettarsi che l'accuratezza della stima della posizione è tanto maggiore quanto minore è l'area del quadrilatero.

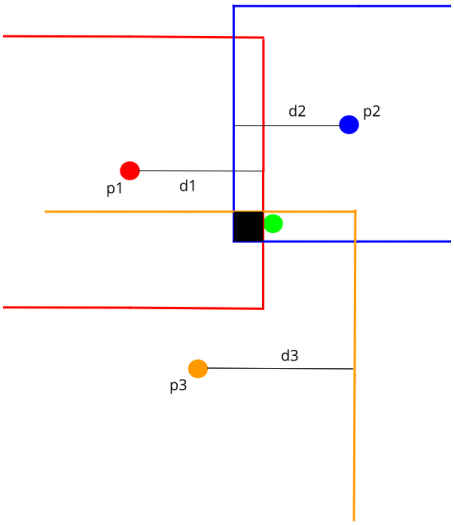


Figura 3. Algoritmo Min-Max. Il centro del quadrilatero formato dall'intersezione dei tre quadrati fornisce una stima della posizione del nodo mobile.

Algoritmo di Trilaterazione [3]

In questo caso la posizione del dispositivo mobile si ottiene come risultato di un sistema non lineare che dipende dalle distanze tra esso e le ancore di riferimento.

Se consideriamo l'ancora p_1 e la distanza d_1, allora la posizione del dispositivo mobile dovrà stare su un qualunque punto della circonferenza di centro p_1 e raggio d_1. Aggiungendo l'informazione relativa alla seconda ancora p_2 è quindi possibile definire un'altra circonferenza centrata in p_2 di raggio d_2 su cui la posizione del dispositivo mobile è vincolata a trovarsi. I due cerchi potrebbero intersecarsi in un solo punto (nel qual caso la posizione del dispositivo mobile sarebbe identificata), non intersecarsi proprio (e questo è un caso che per ora escludiamo) oppure intersecarsi in due punti (Figura 4). In questo caso la posizione cercata è ambigua potendo essere quella relativa ad una delle due intersezioni trovate. Per risolvere questa ambiguità è possibile considerare un'altra ancora p_3 che in questo caso consente di determinare univocamente la posizione cercata.

Considerando che l'equazione di una circonferenza di centro (x_i, y_i) e raggio d_i è:

$$(x - x_i)^2 + (y - y_i)^2 = d_i^2$$

allora l'intersezione si ottiene risolvendo il sistema non lineare

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 = d_2^2 \\ (x - x_3)^2 + (y - y_3)^2 = d_3^2 \end{cases}$$

che risolto rispetto a (x, y) fornisce la posizione del dispositivo mobile.

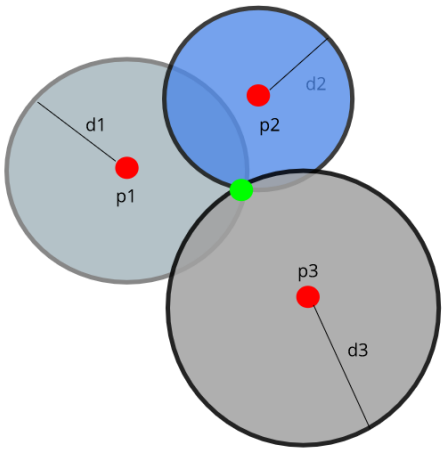


Figura 4. Posizione individuate mediante trilaterazione.

Notiamo che l'algoritmo potrebbe fornire un punto di intersezione non unico come in Figura 5.

Potrebbe inoltre accadere che due circonferenze non abbiano alcun punto di intersezione come in Figura 6.

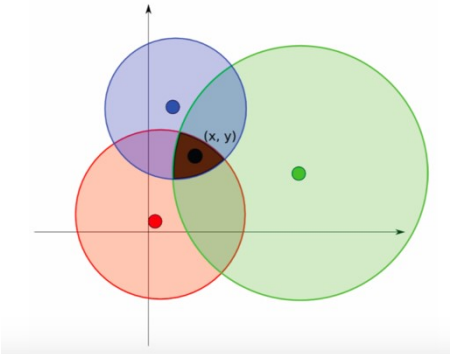


Figura 5. Soluzione non univoca.

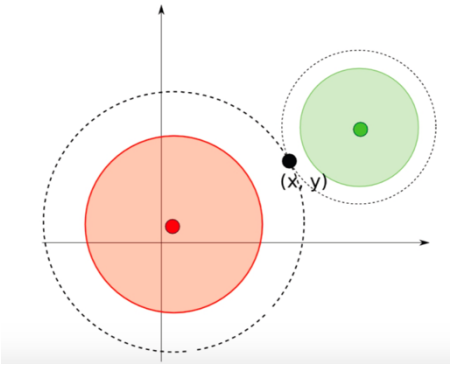


Figura 6. Nessuna intersezione.

Per risolvere questi inconvenienti si può procedere in questo modo:

- Si disegnano due circonferenze (relative a due ancore distinte)
- Se il punto di intersezione è unico allora si prende l'intersezione come posizione stimata

- Se le due circonferenze non hanno punti di intersezione si procede aumentando i raggi in modo proporzionale affinché ci sia una intersezione
- Se invece ci sono due intersezioni si prende quella a distanza minore dalla terza ancora.

Nel caso di tre soluzioni come in Figura 5 si può, ad esempio, considerare la media delle coordinate stimate.

L'algoritmo di trilaterazione può essere esteso al caso di n ancore. Questo caso, noto come **multilaterazione,** è utile poiché le distanza sono chiaramente soggette ad errori e quindi considerare più punti di riferimento consente di ridurre l'incertezza legata alla stima.

Il codice CBM riportato nel riquadro implementa su Commodore 128 i due algoritmi descritti.

Nel codice la matrice *A* di dimensioni $2 \times nA$ contiene le posizioni delle ancore (una per ciascuna riga). La variabile *MB* contiene invece la posizione del dispositivo mobile (che è ciò che l'algoritmo deve stimare). Le righe da 150 a 170 calcolano le distanze del dispositivo mobile da ciascuna ancora. Infine le righe da 190 a 240 e da 255 a 330 implementano gli algoritmi di trilaterazione e min-max, rispettivamente.

Le Figure 7 e 8 mostrano i risultati ottenuti.

Un compito che i più volenterosi possono svolgere è come trovare (se esistono) le intersezioni tra i tre cerchi e l'area di intersezione dei tre quadrati. Cosa succede se la distanza tra il dispositivo mobile e le ancore sono stimate con incertezza?

Buon lavoro e alla prossima avventura con RetroMath!!!

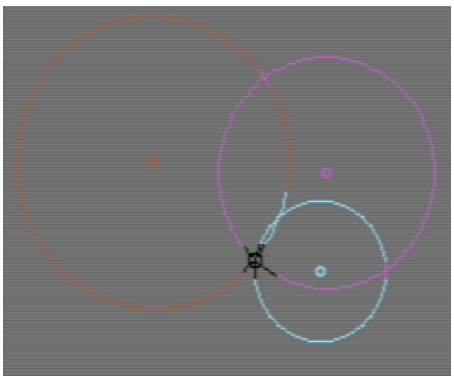


Figura 7. Risultati della trilaterazione in C128.

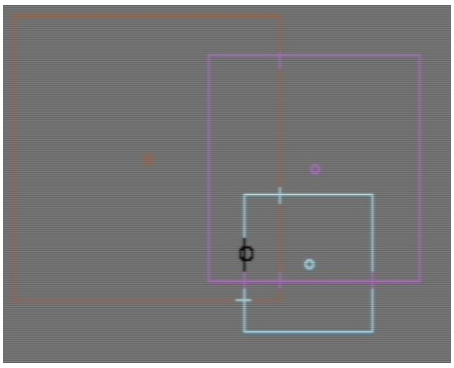


Figura 8. Risultati di Min-Max in C128.

Bibliografia:

[1] M. Werner. Indoor Location-Based Services, Springer.

[2] M. Srivastava A. Savvides, H. Park. The bits and flops of the n-hop multi-lateration primitive for node localization problems. IEEE Sensors Journal, 7:557–561, 2002.

[3] F.B. Elbahhar, A. Rivenq. New approach of indoor and outdoor localization systems. InTech, 2012.

```
10 rem------------------------------
20 rem Min-Max e Trilaterazione
30 rem Giuseppe Fedele
40 rem------------------------------
50 nA = 3 : rem numero di ancore
60 dim A(2,nA)
70 A(1,1) = 75  : A(2,1) = 90
80 A(1,2) = 157 : A(2,2) = 140
90 A(1,3) = 160 : A(2,3) = 95
100 rem dispositivo mobile
110 dim P(2)
120 MB(1) = 125 : MB(2) = 135
130 rem calcola distanze tra MB e A
140 dim d(nA)
150 for i=1 to nA
160 : d(i) = sqr((MB(1)-A(1,i))^2+(MB(2)-
A(2,i))^2)
170 next i
180 :
190 rem Trilaterazione
195 graphic 1,1
200 for i=1 to nA
205 : color 1,i+2
210 : circle 1,A(1,i),A(2,i),d(i),d(i)
218 : circle 1,A(1,i),A(2,i),2,2
220 next i
225 color 1,1
228 circle 1,MB(1),MB(2),3,3
230 get a$: if a$="" goto 230
240 graphic 0
250 :
255 rem Min-Max
260 graphic 1,1
270 for i=1 to nA
275 : color 1,i+2
280 : circle 1,A(1,i),A(2,i),2,2
285 : box 1,A(1,i)-d(i),A(2,i)-
d(i),A(1,i)+d(i),A(2,i)+d(i)
290 next i
300 color 1,1
310 circle 1,MB(1),MB(2),3,3
320 get a$: if a$="" goto 320
330 graphic 0
```

L'invenzione della calcolatrice tascabile

di Alberto Apostolo

Recentemente i mass-media di tutto il mondo hanno diffuso la notizia della scomparsa di Jerry Merryman, il quale è una delle tre persone a cui è attribuita l'invenzione della calcolatrice tascabile nel 1967 (le altre due sono Jack Kilby e James van Tassel).

Il gruppo di lavoro apparteneva alla sede di Dallas della Texas Instruments ed era diretto da Jack Kilby (8/11/1923-20/06/2005), il quale tracciò la via per i computer odierni inventando il circuito integrato (grazie al quale vinse il Premio Nobel nel 2000). Il prototipo costruito dal gruppo si trova allo Smithsonian Institution.

CalTech TI - immagine da viportal.com

La calcolatrice tascabile e' una cosa ormai data per scontata e che si trova ovunque (singolarmente oppure incorporata come software in computer e telefoni cellulari). Non avevo idea di chi fosse e allora mi sono messo a cercare in giro qualche informazione per onorare la sua memoria anche sulle pagine di RetroMagazine.

Jerry Merryman, nacque il 17 Giugno 1932 a Hearne una cittadina ferroviaria del Texas Centrale che si trova 150 miglia a Sud di Dallas.

La vita fu dura da subito. Sua madre morì dieci giorni dopo la sua nascita. Suo padre, un ferroviere, fu costretto a smettere di lavorare negli anni '50 per la cecità causata dal diabete [VP19]. Pressappoco dall'età di 11 anni divenne il radio-riparatore della città.

La moglie Phyllis racconta che con i pochi centesimi guadagnati andava al cinema nei pomeriggi e nelle sere. La Polizia sarebbe andata a farlo uscire perché aveva bisogno che riparasse le loro radio. Frequentò la Texas A&M University a College Station ma non terminò gli studi. Il suo lavoro consisteva nel lavorare al Dipartimento di Oceanografia e Meteorologia e trascorse molto tempo su una piattaforma petrolifera nel Golfo del Messico, misurando la forza dei venti degli uragani.

Cominciò a lavorare alla Texas Instruments nel 1963 per poi ritirarsi nel Gennaio 1994.

Jerry Merryman - immagine da Bing

La signora Merryman ha sostenuto che:-"Disse sempre che non gli importava nulla di essere famoso. Se i suoi amici pensavano che avesse fatto un buon lavoro, era contento".

Ai suoi amici e alla sua famiglia disse sempre che stava creando qualcosa. La figlia Melissa lo ricorda mentre fabbricava da solo un diapason per il loro pianoforte. Lei gli chiese come facesse a tirarlo fuori da quel pezzo di metallo e lui rispose:-"E' facile, basta togliere tutto ciò che non è un Fa-Diesis".

Il suo amico ed ex-collega Gaynel Lockhart rammenta di un telescopio in calcestruzzo a casa sua con collegato un motore che permetteva di seguire un pianeta durante la notte. Un altro amico ed ex-collega, Vernon Porter, affermava:-"Ho un Dottorato di Ricerca in Scienze dei Materiali e ho conosciuto centinaia di scienziati, professori, vincitori di Premi Nobel ecc..

Jerry Merryman era il più brillante che avessi mai incontrato. Punto. Assolutamente, eccezionalmente brillante. Aveva una memoria incredibile e aveva una abilità nel mostrare formule, informazioni, su quasi tutti gli argomenti". Nonostante le sue realizzazioni, restò umile. "Non si sarebbe mai vantato di se stesso, mai", riferisce Melissa Merryman.

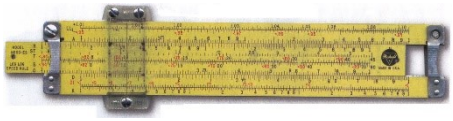
Jerry Merryman è deceduto il 27 Febbraio 2019 all'età di 86 anni per le conseguenze dovute a una insufficienza renale e cardiaca mentre era ricoverato all'Ospedale di Dallas, assistito dalla figlia Melissa e dalla figliastra Kim Ikovic (acquisita dopo seconde nozze nel 1993). Era già stato ricoverato nel Dicembre scorso dopo avere subito complicazioni in seguito a un intervento chirurgico per l'installazione di un pace-maker.

Un sogno antichissimo

A partire dalle civiltà più antiche furono sviluppati sistemi di numerazione e di calcolo. Allo stesso tempo furono compiuti tentativi per realizzare strumenti al fine di eseguire questi calcoli con minore fatica e maggiore accuratezza. Agli antichi Babilonesi è attribuita l'invenzione del primo abaco ("abakos" in greco significa "tavola") nel quale sassolini o frammenti di osso servivano da aiuto per il conteggio (dalla parola latina "calculus" che significa "sassolino" deriva il verbo "calcolare", n.d.A.).

Un passo importante nel calcolo meccanico fu compiuto agli inizi del secolo XVII da John Napier (un nobile scozzese e matematico). A

lui si deve il concetto di "logaritmo" e la compilazione di tavole numeriche cartacee per eseguire calcoli. Il concetto di logaritmo condusse alla invenzione del "regolo calcolatore" da parte del matematico inglese William Oughtred di Cambridge nel 1623. Il regolo calcolatore fu poi migliorato, introducendo l'uso di due componenti dette "fisso" e "scorrevole" aventi scale logaritmiche incise sopra. Con il regolo era possibile effettuare moltiplicazioni, divisioni e il calcolo di alcune funzioni matematiche (ottenendo risultati con almeno un paio di cifre significative di precisione).



Pickett - immagine da Google

Parallelamente in Germania, Wilhelm Schickard costruì un "orologio calcolatore" nel 1623. Sfortunatamente l'invenzione di Schickard fu ignorata fino alla sua riscoperta nel 1935. Assai più conosciuta è l'invenzione del matematico francese Blaise Pascal del 1642 (la "Pascalina").

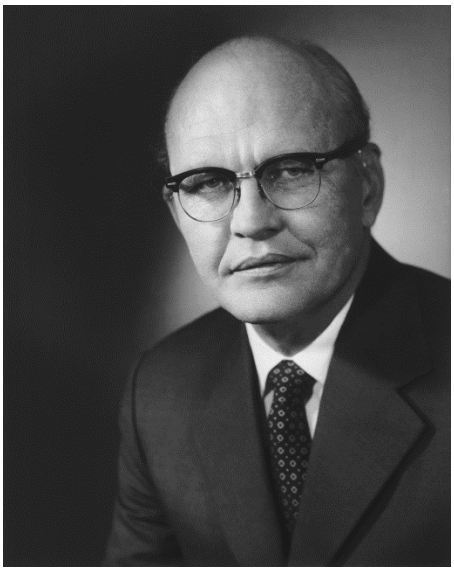
Pascalina - immagine da Google

Altre versioni di calcolatrici meccaniche furono realizzate nei secoli successivi. Una delle più note è quella realizzata dalla azienda tedesca Brunsviga, i cui modelli più aggiornati furono in uso negli uffici fino agli anni '60 del secolo XX.

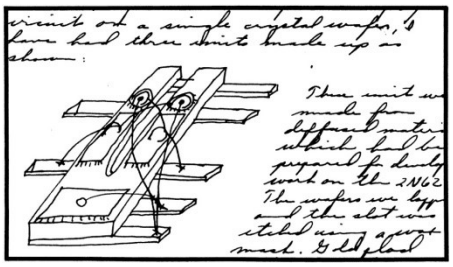
Brunsviga RK13 - immagine da Google

Ma il sogno di macchine sempre sempre più compatte e veloci continuò ad affascinare scienziati e matematici.

Quando Jack Kilby inventò il micro-chip al silicio (o circuito integrato) nel 1958, la realizzazione di quel sogno sembrò a portata di mano.



Jack Kilby - immagine da Google



Appunti di Kilby del 1958 - immagine da [Rei82]

Patrick Haggerty - immagine da ethw.org

Nel 1964 Patrick Haggerty (17/03/1914-01/10/1980, presidente della Texas Instruments) scrisse con lungimiranza che l'elettronica "integrata" avrebbe rimosso le "limitazioni" che determinavano le dimensioni dei dispositivi e riconobbe che con l'invenzione di Kilby era possibile la creazione di un calcolatore portatile che poteva stare sul palmo di una mano.

Un "coso" per rimpiazzare il regolo calcolatore

Patrick Haggerty incaricò Kilby di costituire un gruppo per realizzare una calcolatrice che sarebbe stata potente come quelle elettro-meccaniche, assai più ingombranti in uso all'epoca, ma allo stesso tempo abbastanza piccola da entrare nella tasca di una giacca. Kilby iniziò a lavorare al progetto nell' Ottobre del 1965. Verso la fine del 1965, Kilby (che era il capo di Merryman) presentò l'idea di una calcolatrice. Convocò alcune persone nel suo ufficio e disse: "we'd like to have some sort of computing device, perhaps to replace the slide rule. It would be nice if it were as small as this little book that I have in my hand." ("ci piacerebbe avere qualche tipo di macchina calcolatrice, che forse rimpiazzerà il regolo calcolatore. Sarebbe bello se fosse piccola come questo piccolo libro che tengo in mano", n.d.A.). Merryman nella sua modestia, pensava che stessero realizzando soltanto una calcolatrice e non immaginava che invece stavano gettando le basi della rivoluzione elettronica.

Secondo la testimonianza dell' amico ed ex-collega Ed Millis, Jerry Merryman progettò il circuito in tre giorni (e forse anche notti).

James van Tassel - immagine da Google

L'altro componente del gruppo, James van Tassel, si occupò della tastiera [Rei82]. Facendo uso di circuiti integrati di tipo Medium Scale Integration, nel 1967 venne costruito un prototipo che venne presentato ad Haggerty il 29 Marzo 1967. La calcolatrice era racchiusa in una scatola di alluminio di dimensioni 10.8 cm x 15.6 cm x 4.4 cm e pesava circa 1.4 kg ([Pet11][Kal10]).

A partire dal Settembre 1967 iniziarono le pratiche per ottenere un brevetto, brevetto che fu rilasciato dalle Autorità il 25 Giugno 1974 (U.S. Patent 3819921) [Seg19].

Una successiva investigazione (cfr. "Certain Portable Electronic Calculators, Inv. 337-TA-198" digitalizzato su Google Books) risale al Gennaio 1985.

Grazie allo sviluppo dei circuiti integrati di tipo Large Scale Integration (1969), il passo successivo fu quello di sviluppare una versione ancora più piccola. La Texas Instruments era partner della giapponese Canon, la quale produsse nel 1970 il modello Pocketronic. La Pocketronic pesava 0.8 kg e costava 395 Dollari, effettuava calcoli aritmetici e semplici operazioni algebriche, mostrando i risultati stampati su un nastro di carta termica invece di visualizzarli su un display [Kal10]. Tale modello fu disponibile per il mercato americano nel 1971 al prezzo di 345 Dollari.

In seguito altri costruttori riuscirono a mettere in commercio modelli di calcolatrici tascabili (per es. Sanyo, Sharp, Bowmar, Sinclair). Nel 1972 fu offerta al pubblico la Hewlett Packard HP 35, la prima calcolatrice scientifica tascabile.

11-HP35 - immagine Museo del Calcolatore "Laura Tellini" di Prato

Per un uso consapevole della calcolatrice

Prima della comparsa delle calcolatrici e dei computer i calcoli si eseguivano manualmente. Un piccolo aiuto era fornito dal regolo calcolatore e da tavole matematiche stampate su carta. Gli ingegneri, sapendo di fare calcoli approssimati, progettavano tutto con molta prudenza anche se ciò significava sprecare più materiale e ottenere apparecchiature con minori prestazioni [Sto06].

Con l'avvento delle calcolatrici tascabili, molte tecniche di calcolo (essenzialmente di natura pratica) sono diventate obsolete perché le calcolatrici eseguono lo stesso compito più rapidamente e con maggiore accuratezza. Tuttavia abbandonare queste tecniche ha rappresentatouna vera e propria perdita. Usare le tavole dei logaritmi e delle funzioni trigonometriche permetteva di acquisire familiarità con il loro comportamento. Usare scale e interpolazioni di valori intermedi non esplicitamente tabulati, costringeva gli utenti a mantenere una prontezza di ingegno a un livello a cui gli studenti di oggi non sono più abituati. Una volta che il problema è stato ridotto ad una applicazione della calcolatrice, lo studente diventa più passivo, impara meno e probabilmente è predisposto ad accettare acriticamente qualunque risultato della sua calcolatrice [Higo1]. Con una efficace sintesi J.F. Ganssle definisce questa situazione "computing without thinking" ovvero "calcolare senza pensare" [Ganoo].

Infatti spesso si dimentica che una calcolatrice riserva uno spazio di memoria finito per rappresentare un numero reale, effettuando una approssimazione mediante arrotondamento o troncamento. Ne consegue che in alcuni casi la "aritmetica di macchina" potrebbe inaspettatamente comportarsi in modo diverso dalla "aritmetica teorica" e l'ordine con cui si eseguono le operazioni con la calcolatrice potrebbe essere importante [Epp13]. I manuali universitari di Analisi Numerica contengono esempi di formule modificate in modo opportuno per impedire la perdita di cifre significative durante i calcoli intermedi.

Un esempio abbastanza semplice è quello che riguarda l'equazione di secondo grado (non vale usare la app "Calcolatrice" dello smartphone...):

$$x^2 - 2ax + b = 0$$

$$\text{quando } |b| \ll |a| \text{ conviene usare le formule}$$

$$\begin{cases} x_1 = a + segno(a)\sqrt{a^2 - b} \\ x_2 = \frac{b}{x_1} \end{cases}$$

$$\text{in alternativa a } x_{1,2} = a \pm \sqrt{a^2 - b}.$$

Non si può certamente rinunciare ai benefici delle innovazioni tecnologiche, ma da più parti si avverte l'esigenza di una didattica che sappia integrare l'uso delle nuove tecnologie con lo studio cosciente delle materie e con una ritrovata consapevolezza delle proprie risorse di memorizzazione, comprensione, concentrazione e attenzione. L'esperto di educazione Arvin Vohra ci ricorda che:- "Quello che alzare pesi fa per i muscoli, la Matematica lo fa per la mente" [Voho8].

Bibliografia

[AP19] Associated Press, https://www.nbcnews.com/news/us-news/jerry-merryman-brilliant-man-who-was-inventor-calculator-dies-ng79801 , consultato il 12 marzo 2019.

[Epp13] J.F. Epperson , "An Introduction to Numerical Methods and Analysis", John Wiley & Sons, 2013.

[Ganoo] J.G. Ganssle, "The art of designing embedded systems", 2a ed., Newnes, 2000.

[Higo1] P.M. Higgins, "Divertirsi con la matematica. Curiosità e stranezze dal mondo dei numeri", 2a ed., Edizioni Dedalo, 2001.

[Kal10] S.A. Kallen, "The Information Revolution", Greenhaven Publishing LLC, 2010.

[Pet11] H. Petroski, "An Engineer's Alphabet: Gleanings from the Softer Side of a Profession", Cambridge University Press, 2011.

[Rei82] T.R. Reid , "The Texas Edison", Texas Monthly, Luglio 1982.

[Seg19] D. Segal , "One Hundred Patents That Shaped the Modern World", Oxford University Press, 2019.

[Sto06] C. Stoll, "Quando il regolo dettava le regole", Le Scienze, n.455 luglio 2006, pp. 100–107.

[Voho8] A. Vohra, "The Equation for Excellence: How to Make Your Child Excel at Math", Roland Media Distribution, 2008.

[VP19] AA.VV. , https://www.viportal.co/jerry-merryman-co-inventor-of-the-pocket-calculator-dies-at-86/ , consultato il 12 marzo 2019.

[WWH19] AA.VV. , http://what-when-how.com/inventions/pocket-calculator-inventions/, consultato il 12 marzo 2019.

Il formato file D64

di Francesco Fiorentini

Se avete avuto modo di emulare un Commodore 64 vi sara' sicuramente capitato di caricare un gioco od un programma usando i vari formati compatibili con l'emulatore: uno di questi e' il formato D64. L'utilizzo del file D64 e' piuttosto semplice, prendiamo il file, lo diamo in pasto all'emulatore di turno e poi lo andiamo ad utilizzare come faremmo con un C64 reale con un dischetto inserito nel fido 1541. L'emulatore si fa carico di leggere il file D64 e trattarlo come se fosse un disco fisico vero e proprio.

Fin qui niente di particolare, ma se siete arrivati a leggere questo articolo, significa che come me vi siete fatti qualche domanda e volete capire come e' strutturato e come funziona un file D64. La cosa interessante inoltre e' che, essendo il file la rappresentazione in formato elettronico di un dischetto fisico (singola faccia utilizzabile su un Disk Drive 1541) del Commodore 64, riuscendo a comprendere la sua struttura, capiremo come funzionano anche i dischi reali del biscottone. Interessante vero?

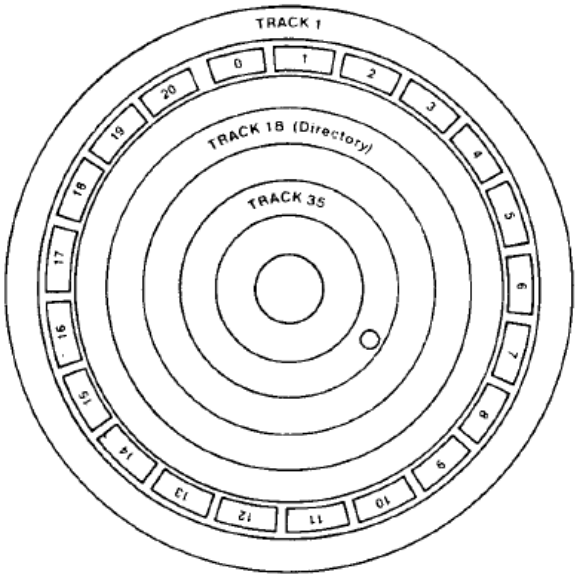


Fig. 1 - Rappresentazione grafica di un disco del 1541 (immagine tratta dal libro Inside Commodore DOS).

Partiamo dall'inizio: durante la formattazione il **DOS** (**Disk Operating System**) del 1541 divide il dischetto in tracce e settori dove poi immagazzinare le informazioni. In totale verranno create 35 tracce circolari partendo dall'esterno del disco, traccia numero 1, fino ad arrivare alla parte piu' interna dello stesso con la traccia numero 35.

Ogni traccia viene a sua volta suddivisa in un numero variabile di settori. Perche' questa differenza nel numero dei settori? Forse che il disco ruoti ad una velocita' diversa a seconda di dove e' posizionata la testina? Non proprio, il disco ruota sempre ad una velocita' costante di 300 rpm (giri al minuto), quello che varia, oltre alla superficie della traccia, e' il clock rate con il quale i dati vengono letti e scritti sul disco. Il disco e' quindi suddiviso in 4 zone con un diverso clock rate. Senza entrare in calcoli complicati con il rischio di appesantire eccessivamente questo articolo, diciamo che il risultato e' quello di generare un numero diverso di settori in queste quatto aree, come si vede nella tabella 1.

Zona	Traccia	Settori	Clock rate	Settori per zona
1	01 - 17	21	307,692 bits/sec	357
2	18 - 24	19	285,714 bits/sec	133
3	25 - 30	18	266,667 bits/sec	108
4	31 - 35	17	250,000 bits/sec	85

Tabella 1: zone, tracce, settori e clock rate

Da un rapido calcolo si evince che il numero totale di settori in un disco e' 683. Ogni settore, grande 256 byte conterra', oltre allo spazio per memorizzare le informazioni anche un blocco di intestazione che identifica il settore stesso.

Moltiplicando 683*256 otteniamo 174848 che, in teoria, dovrebbe essere la capacita' massima di un disco singola faccia utilizzabile su un Disk Drive 1541. Se dividiamo 174848 per 1024 (1KB) otteniamo 171KB che e' la dimensione esatta di un file D64 (come vedete nel file system). Ma aspetta, perche' hai detto in teoria? Perche' il DOS ha la necessita' di riservare dello spazio sul disco dove immagazzinare delle informazioni che serviranno (come vedremo successivamente) a tenere traccia di dove sono memorizzati i files sul disco, la directory e l'occupazione dello stesso.

Il DOS quindi si riserva un'intera traccia, la 18, riducendo cosi' lo spazio a disposizione dell'utente a 169984 byte, corrispondenti a 664 (683-19) tracce per 256 byte. Andiamo quindi a vedere che tipo di informazioni il DOS memorizza in questa traccia.

BAM - Block Availability Map

La BAM o Block Availability Map e' il luogo dove il DOS tiene traccia dei settori nei quali sono memorizzate le informazioni (utilizzati) e quali invece sono ancora a disposizione per memorizzare nuovi dati (liberi). La BAM e' memorizzata nel primo settore (0, lo zero) della traccia 18. Per comodita' ho riportato l'hexdump della prima parte (i primi 144) della BAM dell'immagine del disco **abacus cobol.d64** (attenzione questa e' l'immagine del mio disco, la vostra copia potrebbe essere differente).

```
00: 12 01 41 00 - 91396
01: 15 FF FF 1F - 91400
02: 15 FF FF 1F - 91404
03: 15 FF FF 1F - 91408
04: 15 FF FF 1F - 91412
05: 15 FF FF 1F - 91416
06: 15 FF FF 1F - 91420
07: 15 FF FF 1F - 91424
08: 03 00 02 0A - 91428
09: 00 00 00 00 - 91432
10: 00 00 00 00 - 91436
11: 00 00 00 00 - 91440
12: 00 00 00 00 - 91444
13: 00 00 00 00 - 91448
14: 00 00 00 00 - 91452
15: 00 00 00 00 - 91456
16: 00 00 00 00 - 91460
17: 00 00 00 00 - 91464
18: 10 EC FF 07 - 91468
```

```
19: 00 00 00 00 - 91472
20: 00 00 00 00 - 91476
21: 00 00 00 00 - 91480
22: 00 00 00 00 - 91484
23: 00 00 00 00 - 91488
24: 00 00 00 00 - 91492
25: 00 00 00 00 - 91496
26: 00 00 00 00 - 91500
27: 11 7F FF 03 - 91504
28: 12 FF FF 03 - 91508
29: 12 FF FF 03 - 91512
30: 12 FF FF 03 - 91516
31: 11 FF FF 01 - 91520
32: 11 FF FF 01 - 91524
33: 11 FF FF 01 - 91528
34: 11 FF FF 01 - 91532
35: 11 FF FF 01 - 91536
```

Per comodita' e per facilita' di lettura ho suddiviso l'hexdump a multipli di 4. Il numero a destra e' la posizione assoluta sul disco dell'ultimo byte.

I primi 4 byte `12 01 41 00` rappresentano rispettivamente i puntatori della traccia e del settore della prima directory (hex 12/01, dec 18/01), il carattere ASCII A (hex 41, dec 65) indicante il formato 1541 ed un byte inutilizzato (00). Cosa significa questa informazione? Che la catena delle directory inzia esattamente alla traccia 18, settore 1. Per il momento archiviamo questa informazione, ci servira' nel prosieguo dell'articolo.
I successivi 35 gruppi di 4 byte contengono invece la rappresentazione schematica dello spazio libero/allocato sul disco per ogni singola traccia. Il discorso comincia a farsi interessante e leggermente piu' complicato, quindi cerchero' di spiegare il tutto nel modo piu' semplice possibile.

Per comodita' utilizzero' le informazioni riguardanti la traccia 18, dove, anche in caso di disco vergine, qualcosa sara' sicuramente sempre memorizzato.

```
18: 10 EC FF 07
```

Il primo byte (hex 10, dec 16) indica il numero di settori liberi nella traccia, e fin qui tutto semplice. I successivi 3 byte invece riportano una mappatura dello stato di occupazione di tutti settori presenti nella traccia, in questo caso 19 (si veda nuovamente la tabella 1). Memorizzare le informazioni utilizzando 1 byte per settore non sarebbe stato efficiente, quindi si e' deciso di memorizzare l'informazione in binario: 0 occupato, 1 libero. Vediamo in dettaglio come.

Innanzitutto dobbiamo trasformare i nostri valori (EC FF 07) in binario, ottenendo rispettivamente:
EC = 11101100
FF = 11111111
07 = 0111

Se li mettiamo in fila otteniamo: 11101100 11111111 0111
Dobbiamo poi ricordarci che i bit sono memorizzati a partire da quello meno significativo, quindi per ottenere una rappresentazione grafica coerente dell'occupazione del disco dovremo girare ogni byte:
00110111 11111111 1110

I quattro byte quindi ci dicono che:

```
16 - 0011011111111111110
     12345678901234567890 - posizione
               1         2
```

Vi ricordo che 0 (zero) significa che il settore e' occupato ed 1 invece che e' libero. Proviamo a contare quindi i valori 1 presenti nella stringa di cui sopra. Ci accorgiamo che sono effettivamente 16 (come indicato dal primo byte), e che soltanto il primo, il secondo ed il quinto settore sono occupati. Fermi tutti, e lo zero alla posizione 20? Possiamo ignorarlo in quanto la traccia 18 contiene solo 19 settori.

Se analizziamo in dettaglio le 35 tracce dell'esempio precedente otteniamo la seguente rappresentazione dove O=settore libero, #=settore occupato:

```
TK | FS | Occupazione spazio disco
01 | 21 | OOOOOOOO OOOOOOOO OOOOO
02 | 21 | OOOOOOOO OOOOOOOO OOOOO
03 | 21 | OOOOOOOO OOOOOOOO OOOOO
04 | 21 | OOOOOOOO OOOOOOOO OOOOO
05 | 21 | OOOOOOOO OOOOOOOO OOOOO
06 | 21 | OOOOOOOO OOOOOOOO OOOOO
07 | 21 | OOOOOOOO OOOOOOOO OOOOO
08 | 03 | ######## #O###### #O#O#
09 | 00 | ######## ######## #####
10 | 00 | ######## ######## #####
11 | 00 | ######## ######## #####
12 | 00 | ######## ######## #####
13 | 00 | ######## ######## #####
14 | 00 | ######## ######## #####
15 | 00 | ######## ######## #####
16 | 00 | ######## ######## #####
17 | 00 | ######## ######## #####
18 | 16 | ##OO#OOO OOOOOOOO OOO
19 | 00 | ######## ######## ###
20 | 00 | ######## ######## ###
21 | 00 | ######## ######## ###
22 | 00 | ######## ######## ###
23 | 00 | ######## ######## ###
24 | 00 | ######## ######## ###
25 | 00 | ######## ######## ##
26 | 00 | ######## ######## ##
27 | 17 | OOOOOOO# OOOOOOOO OO
28 | 18 | OOOOOOOO OOOOOOOO OO
29 | 18 | OOOOOOOO OOOOOOOO OO
30 | 18 | OOOOOOOO OOOOOOOO OO
31 | 17 | OOOOOOOO OOOOOOOO O
32 | 17 | OOOOOOOO OOOOOOOO O
33 | 17 | OOOOOOOO OOOOOOOO O
34 | 17 | OOOOOOOO OOOOOOOO O
35 | 17 | OOOOOOOO OOOOOOOO O
```

Vediamo come e' strutturata quindi la prima parte della BAM:

Posizione	Byte	Contenuto
0-1	2	Traccia e settore della prima directory
2	1	Formato del disco (hex 45 per 1541)
3	1	Inutilizzato
4-7	4	Mappatura prima traccia
8-11	4	Mappatura seconda traccia
...	...	Mappature tracce successive
140-143	4	Mappatura trentacinquesima traccia

Tabella 2: prima parte della BAM

Terminata la BAM?

Quasi. In coda alla BAM si trovano altri 27 byte contenenti informazioni interessanti e piu' precisamente:

Posizione	Byte	Contenuto
144-159	16	Nome del disco, riempito con spazi
160-161	2	Spazi

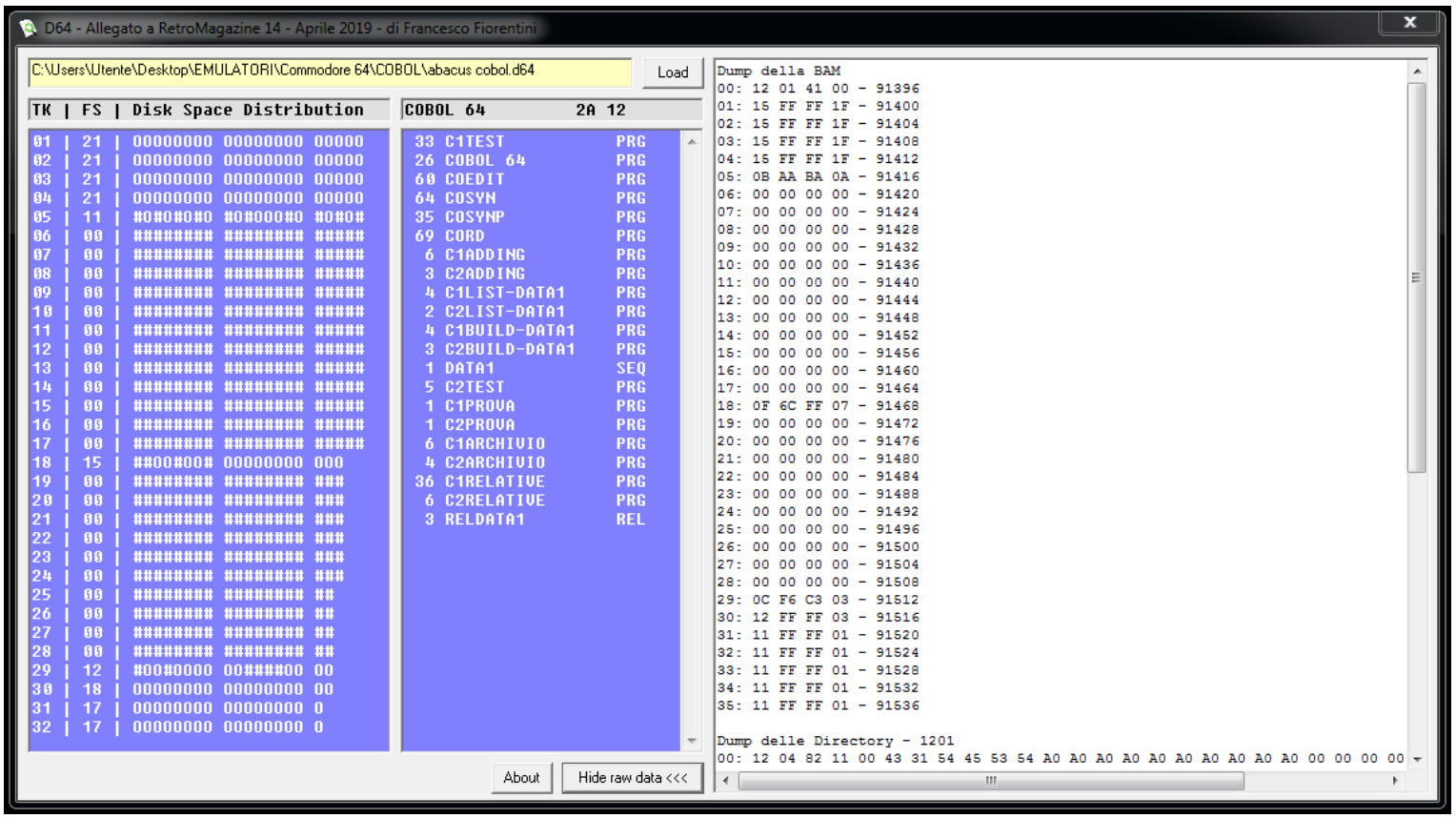


Fig. 2 Il programmino che ho realizzato in Visual Basic 5.0 per leggere un file D64.

162-163	2	ID del disco
164	1	Spazio
165-166	2	Versione del DOS e formato (2A)
167-170	4	Spazi

Tabella 3: seconda parte della BAM - nome del disco, ID, versione

Queste informazioni possono essere tranquillamente lette ed interpretate traducendo il valore Hex del byte nel corrispondente caratteri PETSCII (per comodita' nel programma allegato come esempio ho utilizzato l'ASCII che e' compatibile per buona parte ad esclusione di alcuni caratteri speciali).

La catene delle Directory

Terminata la lettura della BAM e' importante a questo punto stampare l'elenco dei file contenuti all'interno del dischetto. Dove sono memorizzate queste informazioni? Dato che il motore del Disk Drive 1541 gira a 300 rpm, per ottimizzare le prestazioni, il DOS distribuisce l'albero delle directory nella traccia 18 non in modo contiguo, ma con un intervallo pressoche' standard.

Generalmente le directory in un disco seguono questa distribuzione:
0 (BAM), 1, 4, 7, 10, 13, 16, 2, 5, 8, 11, 14, 17, 3, 6, 9, 12, 15, 18.

Ovviamente questa e' solo un'indicazione di massima, nelle mie prove mi sono imbattuto in molti dischi che rispecchiano questa sequenza, ma anche in alcuni in cui la distribuzione era diversa. Come fare quindi per non commettere errori? Semplice, basta seguire le indicazioni che ci vengono fornite dall'albero delle directory stesse.

L'unica informazione certa che conosciamo e' che il puntatore alla traccia ed al settore della prima directory si trova nelle prime due posizioni del primo settore della traccia 18, ricordate i primi quattro byte che abbiamo visto: **12 01** 41 00?

Bene questi ci dicono che la prima directory si trova nel **settore 01** (il secondo) della **traccia 18**. Da li' in poi ogni directory conterra' nei primi due byte i puntatori alla directory successiva; bastera' seguire le molliche di pane come Pollicino e non ci perderemo (sperando che questa volta non ci siano gli uccellini a mangiarle...), o almeno fino a quando non troveremo in questi due byte i valori (**hex 00/FF, dec 00/255**) che indicano che quella e' l'ultima directory del disco.

Posizione	Byte	Contenuto
1-2	2	Traccia e settore della prossima directory: 00/FF indicano che ci troviamo all'ultima occorrenza.
2-31	30	Primo file della directory
32-33	2	Inutilizzati
34-63	30	Secondo file della directory
64-65	2	Inutilizzati
66-95	30	Terzo file della directory
96-97	2	Inutilizzati
98-127	30	Quarto file della directory
128-129	2	Inutilizzati
130-159	30	Quinto file della directory
160-161	2	Inutilizzati
162-191	30	Sesto file della directory

192-193	2	Inutilizzati
194-223	30	Settimo file della directory
224-225	2	Inutilizzati
226-255	30	Ottavo file della directory

Tabella 4: organizzazione delle directory

Ora che abbiamo imparato a seguire la catena delle directory non ci resta altro da fare che leggere le informazioni che vi sono contenute e che riguarderanno ovviamente, il nome dei file, la loro tipologia, lo spazio che occupano sul disco e dove sono memorizzati.

I 30 byte che contengono le informazioni dei file nella directory sono descritti nel seguente modo:

Posizione	Byte	Contenuto
1	1	DOS File type (si veda tabella 6)
2-3	2	Traccia e settore dove si trova l'inizio del file
4-19	16	Nome del file (attenzione ai caratteri PETSCII
20-22	3	Inutilizzati ad eccezione per i file Relative
23-26	4	Sempre inutilizzati e quindi '00'
27-28	2	Riservati dal DOS per operazioni sul file
29-30	2	Dimensione del file in blocchi memorizzata in notazione LO-HI

Tabella 5: descrizione file nella directory

Tutto relativamente semplice, a parte il DOS File type di cui possiamo vedere la definzione nella tabella sottostante (tabella 6) e la dimensione del file in blocchi perche' deve essere calcolata tenendo conto che e' memorizzata in notazione LO-HI.

Cosa significa notazione LO-HI? Semplicemente che per calcolare l'occupazione del file dobbiamo tenere presente che il byte a sinistra e' il byte basso (lo-byte), mentre quello a destra e' il byte alto (hi-byte). Questa tipologia di memorizzazione permette di risparmiare spazio per memorizzare le informazioni e deve essere calcolata utilizzando la formula: dimensione = lo-byte + hi-byte * 256

Facciamo l'esempio con il primo file nella directory in **figura 2**.
I Valori sono 1E 00 e quindi utilizzando la formula diventeranno
Dimensione = 30 (1E) + 0 (00) * 256

Totale 30 blocchi utilizzati sul disco dal file C1 TEST.

HEX	DEC	Definzione
$80	128	DEL - File cancellato
$81	129	SEQ - File sequenziale
$82	130	PRG - Programma
$83	131	USR - File utente
$84	132	REL - File Relative
$00	0	Eliminato, non deve apparire
$01	1	SEQ - File sequenziale non chiuso
$02	2	PRG - Programma non chiuso
$03	3	USR - File utente non chiuso
$04	4	REL - File Relative non chiuso
$AO	160	DEL - File cancellato (sostituzione)
$A1	161	SEQ - File sequenziale (sostituzione)
$A2	162	PRG - Programma (sostituzione)
$A3	163	USR - File utente (sostituzione)
$A4	164	REL - File Relative (sostituzione) - infrequente
$CO	192	DEL - File cancellato (locked)
$C1	193	SEQ - File sequenziale (locked)
$C2	194	PRG - Programma (locked)
$C3	195	USR - File utente (locked)
$C4	196	REL - File Relative (locked)

Tabella 6: significato del primo byte (DOS file type)

Bene, come avrete certamente intuito, la traccia 18 del formato D64 e di conseguenza dei dischi del 1541, e' la traccia di partenza per accedere a tutte le informazioni contenute sui dischi. Come mai e' stata scelta la traccia 18 e non la 1 o la 35? Perche' e' la traccia che si trova esattamente a meta' del disco e quindi in questo modo si riesce ad ottimizzare le prestazioni di lettura e scrittura minimizzando il movimento della testina del disk drive. Molto ingegnoso.

Ritengo, in questo mio breve excursus, di aver fornito sufficienti informazioni su come creare una utility fatta in casa per leggere i dati memorizzati sui file D64. Se volete cimentarvi quindi non avete che da provare. Nel caso di dubbi o domande la redazione di RM e' sempre a vostra disposizione. Se invece volete dare un'occhiata al programmino (con tanto di sorgenti) che ho creato per leggere i file D64 potete trovarlo qui: http://www.retromagazine.net/getrm.php?id=d64 (**leggete il readme.txt allegato**).

Si tratta di un semplice programma in Visual Basic 5.0 sviluppato nei ritagli di tempo durante la stesura dell'articolo per assicurarmi che le infomazioni fornite fossero accurate. Visual Basic 5.0, rilasciato da Microsoft nel 1991, puo' essere scaricato da **Winworldpc.com**: https://winworldpc.com/product/microsoft-visual-bas/50

Funzionamento: il programma apre un file D64 in binario ed accede ad ogni singolo byte partendo dal primo della traccia 18 (il **91393**, ottenuto moltiplicando il numero dei settori della prima area, 657 per 256 ed aggiungendo 1). Da li' in poi ci sposteremo in avanti di un byte per volta, interpretandolo con le informazioni descritte nelle tabelle dell'articolo. L'intento del programma non e' la ricerca della prestazione, quanto quello di far comprendere in modo semplice ed intuitivo come leggere le informazioni contenute in un file D64.

Un prossimo articolo potrebbe riguardare la possibilita' di leggere ed esportare un file direttamente da un'immagine D64; magari un listato BASIC, dove dovremmo interpretare anche i token... (*date un'occhiata all'articolo successivo*). ☺

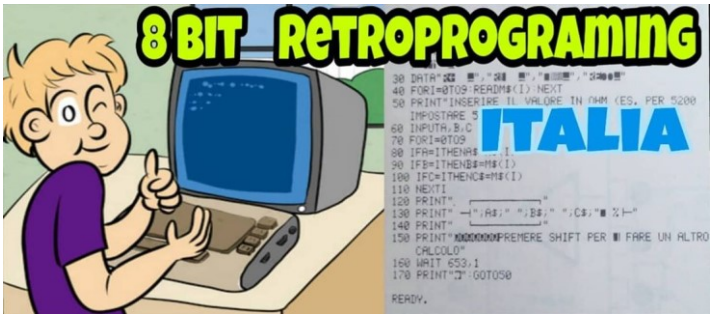
Buon divertimento!

Links Utili

Il libro **Inside Commodore DOS**, la bibbia del DOS del Commodore 64 e da cui ho tratto spunto per questo articolo, potete trovarlo qui:
https://www.pagetable.com/docs/Inside%20Commodore%20DOS.pdf

Mastercode: Assembler e BASIC Extender

di Andras Vajda (8 Bit RetroProgramming Italia)



Nel 1983 viene stampato un testo che sarà una pietra miliare nella storia del C64. Scritto a quattro mani da un programmatore BASIC professionista e un ingegnere elettronico esperto in Assembly 6502, propone non solo un completo assemblatore scritto interamente in BASIC V2, ma anche (come esempio applicativo) un vero e proprio BASIC Extender residente in RAM che implementa oltre una dozzina tra nuovi comandi e funzioni. Col presente articolo si propone un porting per CBM Prg Studio di tutto il codice Assembly relativo a tale Extender, aggiornato ed esteso con ulteriori comandi e comprensivo della correzione di un curioso ma fastidioso bug di progettazione dell'originale.

«*Programming is one of the most difficult branches of applied mathematics; the poorer mathematicians had better remain pure mathematicians.*» - (Edsger Dijkstra, 1930-2002)

1 Introduzione

Il testo di cui discutiamo [3] è probabilmente noto ai più nell'edizione italiana Jackson Libri del 1985 «Il linguaggio macchina del Commodore 64» [4]. La «strana coppia» di autori è costituita da David Lawrence, programmatore BASIC professionista ed autore di numerosi testi di informatica, e Mark England, ingegnere elettronico esperto in Assembly.

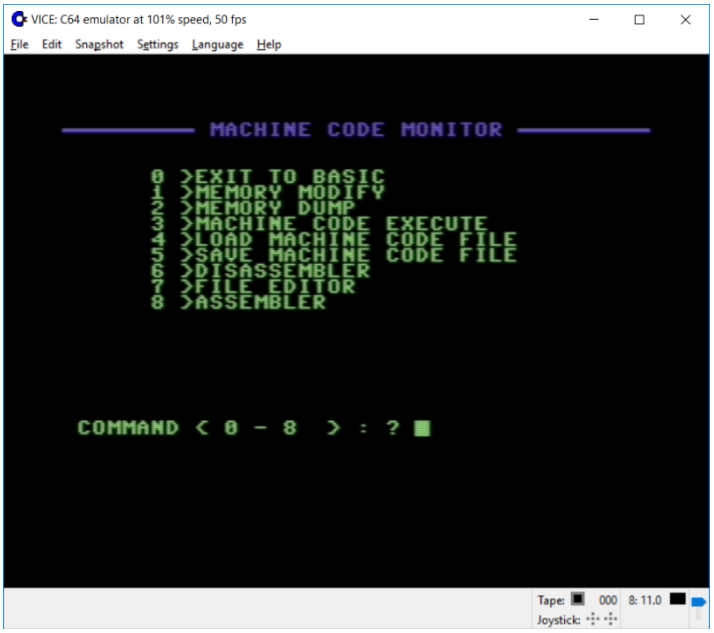
L'idea di un libro dall'approccio così innovativo, che ancora oggi rimane unico nello sterminato panorama della letteratura tecnica applicativa per C64, è venuta a Mark England: evidentemente stufo (come molti di noi) di studiare quei testi della prima ondata sull'Assembly dei PET che si limitavano a duplicare le informazioni su istruzioni, opcode e operandi già presenti nel datasheet della CPU 6510 proponendo poi, nel migliore dei casi, una fantasiosa silloge di brevi routine assembly di scarso o nullo interesse pratico.

Altro punto critico della maggior parte dei testi sull'Assembly della prima metà anni Ottanta: presupponevano che il lettore avesse facile e immediato accesso ad un ambiente assemblatore, cosa in realtà nient'affatto scontata all'epoca, visti anche i costi di licenza dei principali ambienti di sviluppo disponibili in quel periodo.

Nasce così l'idea di sviluppare un completo assemblatore scritto interamente in BASIC, che il lettore potrà digitare gradualmente come salutare esercizio, aiutato anche da un verificatore di checksum: e questa è solo la prima parte del divertimento, perché gli «esempi applicativi», invece di essere il solito inconcludente florilegio di scarne routine aritmetiche e di I/O, sono in realtà una completa applicazione che guida il lettore nei meandri dell'interprete BASIC e dell'interazione al più basso livello col Kernal, aggiungendo ben 15 nuovi comandi e funzioni al CBM BASIC V2.

2 Mastercode

La prima parte del testo è interamente occupata dai listati dell'Assembler Mastercode, il quale oggigiorno riveste unicamente interesse storiografico e didattico: digitarne il codice per intero, possibilmente su una macchina reale, costituirebbe un ottimo esercizio per molti neofiti. Si tratta di un ambiente modulare, basato su menu, fondamentalmente completo nell'impostazione ma molto limitato sia nella sintassi Assembly accettata, sia nelle prestazioni. Il suo pregio fondamentale è stato quello di conferire una totale autonomia e indipendenza al testo rispetto ai costosi ambienti commerciali all'epoca esistenti. Non occorre però neppure rimarcare che l'editor di memoria, il disassembler e il monitor risultano realmente utilizzabili solo in un ristretto novero di casi, dal momento che risiedono nella memoria dedicata al codice BASIC, e risentono di limitazioni drastiche sia rispetto ai monitor software dedicati, sia a maggior ragione rispetto a quelli su cartuccia, spesso perfino rilocabili e con caratteristiche decisamente avanzate. L'immagine seguente mostra l'aspetto originale del menu principale di Mastercode.



3 BASIC Extender

La programmazione di un BASIC Extender differisce profondamente dalla normale programmazione applicativa e richiede una solida consapevolezza di numerosi dettagli inerenti il codice in ROM (Kernal e BASIC). Una piena comprensione del codice richiederà, oltre allo studio del testo in questione, anche la consultazione di mappe di memoria e disassembly commentati come [1], [2], [5] e una conoscenza dei meccanismi fondamentali dell'interprete residente.

I comandi e le funzioni del CBM BASIC V2 constano di parole chiave o keyword, quelle che normalmente digitiamo nei programmi e al prompt interattivo, come LIST o PRINT. Codeste stringhe però non vengono

memorizzate letteralmente nel programma, per ovvi motivi di ottimizzazione in spazio: ciascuna di esse è invece sostituita da un codice univoco, rappresentato da un piccolo intero (un singolo byte) detto token. Tale valore numerico è caratterizzato dall'avere il bit più significativo posto a 1, per evitare ambiguità con altri elementi sintattici previsti dal linguaggio, come nomi di variabile o operatori. In fase di esecuzione, il token viene utilizzato per indicizzare correttamente la jump table o tabella dei vettori che contiene gli entry point delle routine assembly che implementano i singoli comandi BASIC; durante il listing il valore del token viene invece usato per un lookup inverso sulla tabella delle keyword, in modo che l'utente veda esattamente il comando che ha digitato in fase di immissione, rendendo del tutto trasparente il meccanismo di tokenizzazione o crunching.

Per gli usuali motivi di ottimizzazione in spazio (8kib di ROM erano pochi anche all'epoca), la tabella delle keyword è memorizzata usando uno dei tanti accorgimenti intelligenti dell'era Commodore: invece di sprecare un byte ponendolo a zero per ogni entry, rendendola così null-terminated o ASCII-zero che dir si voglia, il bit più significativo MSB dell'ultimo carattere è semplicemente posto a 1. Ciò, ovviamente, limita drasticamente ad ASCII-7 (in realtà PETSCII-7) il range dei caratteri effettivamente rappresentabili, il che tuttavia non costituisce affatto un problema per tale tipo di applicazione. Si tratta di una prassi talmente radicata che molti assemblatori offrono una apposita direttiva a tale scopo, come la .shift del Turbo Assembler.

Tenendo in mente quanto appena succintamente ricordato, i principali problemi che ci si trova ad affrontare nel progettare una espansione BASIC che rispetti il più possibile l'esistente sono i seguenti:

1) Il CBM BASIC risiede su una memoria di sola lettura (ROM o EPROM), tuttavia per rendere possibile l'interoperabilità con l'Extender occorre modificarne il codice in vari punti.

2) La tabella delle keyword standard ha dimensione limitata a 256 bytes, in quanto viene scandita entro l'interprete originale in modalità di indirizzamento indicizzato tramite il registro Y, ovviamente a 8 bit. Sfortunatamente, tale tabella risulta inutilizzabile ai fini di una espansione, in quanto già quasi totalmente occupata.

3) Occorre scrivere il codice Assembly per ciascuno dei nuovi comandi aggiunti all'Extender, in modo ovviamente compatibile con il firmware residente sulla macchina.

Il punto 1) (che su quasi ogni altro PET sarebbe l'epitaffio di ogni tentato progetto del genere) è pressoché insignificante nel nostro caso, dal momento che il C64 possiede infatti 64kib effettivi di memoria RAM contigua ed indiscriminatamente accessibile, alcuni segmenti della quale sono rimappati su ROM o port di I/O. Ma è possibile (e molto facile) copiare integralmente il contenuto della ROM nella RAM "sottostante", abilitando poi definitivamente quest'ultima e rendendo modificabile ogni byte copiato, ovviamente solo entro la sessione corrente.

La sola copia in RAM, tuttavia, non risolve il problema della tabella delle keyword e quello parallelo della corrispondente tabella dei vettori (come già ricordato, i puntatori alle singole routine corrispondenti a ciascuna keyword), la quale è parimenti quasi piena. Vi è poi un ulteriore problema: i token già occupati dalle keyword standard vanno dal 128 al 202, più 255 che rappresenta il pigreco. In considerazione di tutti questi vincoli, è strettamente necessario predisporre altre indipendenti tabelle, ed alterare parzialmente (seppure in modo minimo) il codice dell'interprete esistente, come già anticipato. Fortunatamente, l'intelligenza e la lungimiranza dei progettisti Commodore hanno fatto sì che buona parte del codice coinvolto sia revettorizzabile con estrema semplicità, in quanto puntato da variabili collocate entro i primi 2 kb di RAM, nell'area di sistema. In particolare, il vettore a 16 bit alla locazione 306h (ossia agli indirizzi 306 e 307 esadecimali) indirizza la routine PRTTOK, e risulta facilmente reindirizzabile per i nostri scopi. I token per le nuove keyword assumeranno valori da 203 in su, il che li rende immediatamente distinguibili da quelli standard.

Per le tabelle aggiuntive, la soluzione adottata dalla maggior parte dei BASIC Extender (incluso quello di cui discutiamo) consiste nell'uso di tabelle alternative indirizzate dinamicamente. Si deve partire dal presupposto che la routine di tokenizzazione («crunching» nella letteratura Commodore) del BASIC V2 non è particolarmente sofisticata, né tantomeno ottimizzata per le prestazioni: invece di ricorrere a strutture dati dedicate (come un albero binario) e algoritmi efficienti, si limita ad effettuare una banale ricerca lineare sulla tabella delle keyword usando come chiave la potenziale keyword attualmente in elaborazione. Un semplice contatore, inizializzato a monte del loop di scansione, viene incrementato ad ogni mancata corrispondenza, ed è proprio tale contatore che al termine della routine (salvo errori di sintassi) conterrà il valore del token corrispondente alla keyword esaminata nella linea BASIC corrente.

Oltre a questa struttura di base, decisamente elementare, vi sono poi nel codice dell'interprete residente varie euristiche per gestire le eccezioni e le singolarità del linguaggio: tra questi, il trattamento del separatore ':' che non è un token vero e proprio, ma viene gestito separatamente nel codice del cruncher/tokenizer.

La tabella aggiuntiva delle keyword viene indirizzata dinamicamente da una apposita routine CRUNCH nel codice di Lawrence & England, quando la keyword correntemente analizzata non ha trovato riscontro nella tabella originale. Di fatto, la scansione di ogni singola keyword e l'associazione con il relativo codice numerico (tokenizzazione) in una linea di codice seguono uno schema logico estremamente semplice e lineare:

- Il tokenizer confronta la stringa della potenziale keyword presente nella linea di codice correntemente analizzata con le keyword della tabella originale del BASIC V2 (con inizio alla locazione $A09E): END, FOR, NEXT, DATA, ...
- Se il confronto va a buon fine, si esce dal loop principale del tokenizer. A quel punto il numero contenuto alla locazione 10 (con l'aggiunta di un offset pari a 128, che per convenzione caratterizza i token BASIC) è il valore del token corrispondente al comando trovato nella linea di codice BASIC appena analizzata e viene quindi memorizzato assieme al resto della riga di codice.
- Ad ogni confronto fallito il token counter (contenuto nella locazione di pagina zero 10) viene incrementato e si passa alla keyword successiva in tabella.
- Se al termine del loop la stringa non ha trovato corrispondenza, la scansione della tabella delle keyword è giunta a fine e il token counter ha raggiunto il suo valore massimo per il BASIC standard, pari nel nostro caso a 202. Normalmente a questo punto si avrebbe un errore sintattico, ma in questo caso potrebbe ancora trattarsi di un nuovo comando. Una piccola modifica al codice ROM originale impone quindi di richiamare la routine CRUNCH sopra menzionata, la quale sostituisce dinamicamente nel tokenizer l'indirizzo di partenza della tabella delle keyword con il valore della label KEYWRD nel nostro sorgente (di norma $C000) e lo costringe ad eseguire un nuovo loop di scansione, ricominciando dal punto 1 ma con una differente tabella delle keyword e senza azzerare il token counter. In questo modo, per costruzione, è garantita l'assegnazione di token nell'intervallo riservato all'Extender, dal 203 in poi.

- Se uno dei confronti all'interno della tabella delle keyword estese va a buon fine, siamo in presenza di una keyword dell'Extender, che viene regolarmente tokenizzata. In caso contrario: SYNTAX ERROR.

In fase di esecuzione viene implementato un meccanismo analogo per la scelta della tabella dei vettori, in questo caso semplificato dalla immediata discriminazione tra gli intervalli di valori dei token standard ed estesi. La soluzione originariamente prevista da Lawrence e England prevedeva un totale di 15 nuove keyword, di cui tre funzioni, oltre a una coppia di istruzioni puramente rappresentative (FAST e SLOW) che fin dall'inizio l'autore del presente articolo non ha mai ritenuto opportuno implementare. La tabella seguente riassume sinteticamente le nuove funzionalità disponibili: poiché alcuni comandi hanno un numero elevato di parametri, per esigenze tipografiche si è scelto di ometterli quando potevano compromettere la leggibilità.

UNDEAD	Ripristina un programma BASIC dopo un (accidentale) comando NEW.
SUBEX	Elimina l'ultimo indirizzo di ritorno dallo stack.
RKILL	Compressione sorgente BASIC, elimina spazi e REM.
DOKE <addr>,<val>	Versione a 16 bit della POKE.
PLOT <row>,<col>	Posiziona il cursore alle coordinate specificate.
DELETE <inizio>,<fine>	Cancella intervalli di linee contigue da un sorgente BASIC.
BSAVE <...>	Salva un blocco di memoria su file.
BLOAD, BVERIFY <...>	Carica (verifica) un blocco di memoria da file.
MOVE <to>,<from>,<len>	Copia un blocco di memoria ad un nuovo indirizzo.
FILL <start>,<len>,<val>	Riempie un blocco di memoria con il valore dato.
RESTORE <line>	Versione estesa della RESTORE originale.
VARPTR(<var>)	Restituisce un puntatore alla variabile referenziata.
YPOS()	Complementare alla POS() del BASIC V2, restituisce la riga del cursore.
DEEK(<addr>)	Complementare a DOKE, è in pratica una PEEK() a 16 bit.

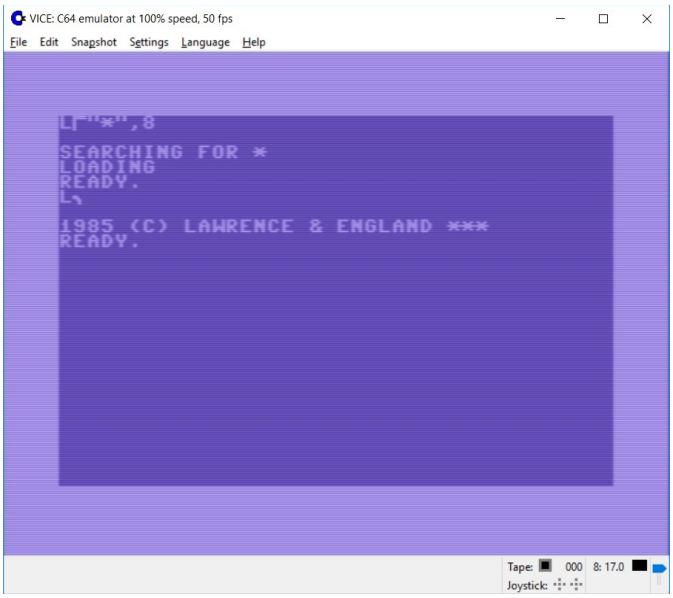
Nella implementazione originale era previsto, oltre al file binario prodotto da Mastercode e contenente il codice della vera e propria estensione BASIC, un loader scritto in BASIC V2 che si occupava del caricamento da disco (o nastro), della copia da ROM a RAM del BASIC V2 originale, della rilocazione dell'estensione a partire dall'indirizzo $Cooo e della revettorizzazione.

4 Porting ed estensione del progetto

Nel 1985, dopo avere pedissequamente digitato tutto il codice originale proposto dal testo, si è provveduto al porting dei sorgenti dell'Extender su Turbo Assembler, in un unico file. Tale lavoro mirava a superare le inerenti limitazioni sintattiche di Mastercode, rendendo più facile la manutenzione e l'espansione del progetto. Tutte le operazioni di caricamento e predisposizione dell'ambiente possono infatti essere centralizzate ed eseguite direttamente in Assembly senza il minimo problema, con un singolo eseguibile «mascherato» da programma BASIC ovvero caricato interamente da disco a partire dalla locazione di default $0801 e successivamente eseguito con un semplice comando RUN. Il codice di startup elaborato all'epoca è qui riportato in una moderna versione per CBM Prg Studio.

```
;************************************
; Codice "universale" per startup BASIC
;************************************
*=$0801
;------------------------------------
WORD BASEND ; Indirizzo della prossima linea
WORD 1985 ; Numero di linea -> anno di stesura...
BYTE $9E ; Token per "SYS"
TEXT "2102:" ; Indirizzo di avvio: $0836
BYTE $8F ; Token per "REM"
BYTE $22 ; Token per '"'
; Rendo illeggibile il listing con una serie di DEL ($14)
BYTE 20,20,20,20
BYTE 20,20,20,20
BYTE 20,20,20,20
; Solo la stringa seguente apparira' nel listato:
TEXT "(c) lawrence & england ***"
BYTE 0 ; Terminatore di linea
BASEND WORD 0 ; Terminatore di programma
;------------------------------------
;************************************
;** Inizio codice applicazione **
;************************************
```

Esaminando il codice il lettore più smaliziato noterà come, già in quel lontano 1985, l'allora giovane programmatore univa costantemente il divertimento all'apprendimento. Andando oltre la banale funzionalità di avvio da BASIC del codice Assembly, il listato con pochi bytes aggiuntivi risulta anche «protetto» da sguardi indiscreti (un programmatore minimamente esperto userà comunque un monitor con disassembler per esaminare a piacimento il codice dopo il caricamento) con una tecnica all'epoca universalmente diffusa per i software da edicola e commerciali. Se si esegue un LIST subito dopo il caricamento del codice, si ottiene unicamente quanto illustrato nello screenshot seguente:



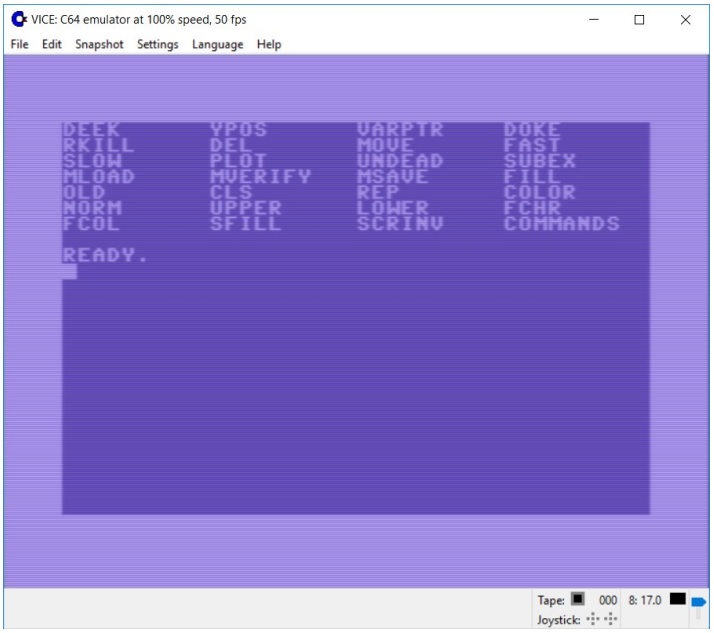
La sezione di codice successiva si occupa delle operazioni preliminari necessarie alla modifica del BASIC residente su ROM: copia il BASIC nella RAM sottostante e in seguito modifica alcuni vettori, reindirizzando alle nuove routine dell'espansione che ne permettono la coesistenza col BASIC originale.

```
;*************************************
; Espansione BASIC / RELEASE 2.0
;*************************************
;-------------------------------------
; Rilocazione codice in $C000
;-------------------------------------
START LDA #<LDR_END+1
STA MVSTART
LDA #>LDR_END+1
STA MVSTART+1
LDY #$00
STY MVDEST
LDA #$C0
STA MVDEST+1
LDX #LEN
MOVE1 LDA (MVSTART),Y
STA (MVDEST),Y
INY
BNE MOVE1
INC MVSTART+1
INC MVDEST+1
DEX
BNE MOVE1
;-------------------------------------
; Modifiche all'interprete BASIC C64:
; copia BASIC ROM in RAM e revectoring
;-------------------------------------
LDA $01 ; Seleziona la ROM BASIC
ORA #$01
STA $01
LDY #$00
; Non occorre modificare il registro $01 ad ogni lettura:
; l'hardware del C64 e' progettato per scrivere
; comunque su RAM, anche se viene selezionata la ROM.
BSTART LDA BASIC_START,Y
BDEST STA BASIC_START,Y
INY
BNE BSTART
INC BSTART+2
INC BDEST+2
LDX BDEST+2
CPX #$C0
BNE BSTART
MODIFY LDA $01 ; Seleziona la RAM in

BFFF
AND #$FE
STA $01
LDX #_JMP
STX $A7E1
LDA #<EXECUT
STA $A7E2
LDA #>EXECUT
STA $A7E3
STX $AFAD
LDA #<FUNEVL
STA $AFAE
LDA #>FUNEVL
STA $AFAF
STX $A604
LDA #<CRUNCH
STA $A605
LDA #>CRUNCH
STA $A606
LDA #<PRTTOK
STA $0306
LDA #>PRTTOK
STA $0307
LDA #<RESTORE-1
STA $A024
LDA #>RESTORE
STA $A025
...
LDR_END BYTE 0
```

L'estensione BASIC, nella nuova versione per Turbo Assembler, è poi stata arricchita con ulteriori nuovi comandi, di ispirazione didattica o mutuati da altre estensioni BASIC. Vale la pena di sottolineare nuovamente che questo progetto ha un valore precipuamente didattico e non presenta la benché minima velleità di porsi al livello di un Simon's BASIC o di qualsiasi altro Extender commerciale.

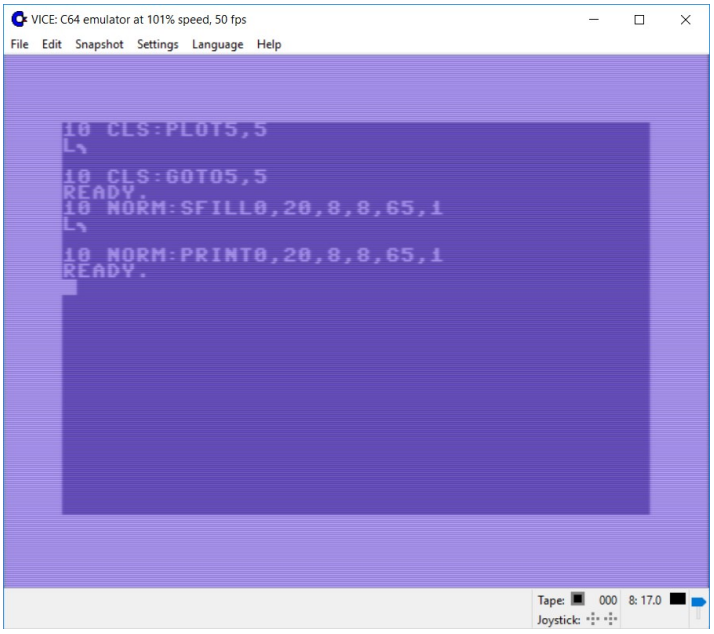
CLS	Cancella lo schermo.
COLOR <sfondo>,<bordo>,<testo>	Imposta i colori usando i codici CBM.
COMMANDS	Elenca a video tutti i nuovi comandi e funzioni dell'estensione.
FCHR <...>	Riempie con un carattere un'area rettangolare a video.
FCOL <...>	Riempie con il colore dato un'area rettangolare a video.
LOWER	Imposta il set di caratteri minuscolo.
NORM	Comoda scorciatoia per impostare colori e case di default.
REP	Imposta ciclicamente l'autorepeat della tastiera.
SCRINV <...>	Inverte sfondo e testo in un'area rettangolare a video.
SFILL <...>	Versione specializzata di FILL per la memoria video.
UPPER	Complementare a LOWER, imposta il set maiuscolo.

Si è inoltre introdotto un comando OLD, semplice e più sobrio sinonimo per UNDEAD. L'immagine seguente mostra il risultato dell'esecuzione di COMMANDS dopo il caricamento dell'estensione.



5 Un curioso bug

All'epoca l'aggiunta di nuovi comandi all'Extender rivelò presto un problema non previsto dagli autori, in quanto il bug apparentemente non era riproducibile nel contesto delle 14+3 keyword originali. L'immagine seguente mostra in modo immediato il problema:



Come si vede, esisteva un problema di errata tokenizzazione: il comando PLOT viene tokenizzato e quindi listato come GOTO, mentre la PRINT prende il posto di SFILL, e così via. Questo nel caso in cui sulla riga di programma siano presenti due o più nuove keyword.

In base ai meccanismi accennati al paragrafo §3, si possono evidentemente creare situazioni di concatenazione di nuovi comandi su una singola linea a causa di cui l'associazione tra token counter e indirizzo della tabella delle keyword non è più sincronizzata: ciò avviene perché esistono (rari) percorsi di esecuzione nel codice del tokenizer ROM che non transitano dal punto di iniezione in cui viene richiamato il codice aggiuntivo CRUNCH previsto dagli autori. Ci si trova quindi in una situazione incongruente in cui il token counter viene inizializzato a zero, ma nel codice del tokenizer viene ancora puntata la tabella delle keyword estese Coooh e non quella originale: ciò genera esattamente il tipo di errore evidenziato.

Effettuando tramite monitor un accurato reverse engineering del codice ROM interessato all'analisi di una linea di codice BASIC e alla tokenizzazione, si è quindi determinata la necessità di un ulteriore punto di iniezione nel codice del tokenizer stesso, in modo da garantire in qualsiasi caso la sincronizzazione tra l'inizializzazione del token counter (contenuto come abbiamo visto nella locazione di pagina zero 10) e l'indirizzo della tabella delle keyword originali alla locazione A09Eh. Il codice seguente, aggiunto al sorgente dell'estensione BASIC riveduta e corretta, risolve efficacemente anche tale bug.

```
;*************************************************
;** Il codice seguente non fa parte del testo ed
;** e' stato aggiunto per risolvere un curioso bug
;** del BASIC Extender. L'associazione originale tra
;** tabella delle nuove istruzioni e contatore usato
;** dal tokenizer BASIC non e' sufficientemente
;** robusta, perché esiste almeno un caso in cui
;** si crea una desincronizzazione, ad esempio
;** digitando una linea di programma come segue:
;** 10 CLS:SFILL 0,20,8,8,65,1
;** il secondo comando viene tokenizzato in modo
;** errato. Questo avviene, in breve, perche' gli
;** autori non hanno considerato tutti i possibili
;** percorsi di esecuzione nel tokenizer ROM,
;** cio' che di norma si attua effettuando una
;** analisi full code coverage con un tracer.
;** Per ovviare basta comunque un ulteriore punto
;** di iniezione, in corrispondenza della
;** inizializzazione del token counter. Si aggiunge
;** una brevissima routine che garantisce che
;** all'azzeramento del counter la tabella corrente
;** delle keyword sia quella originale del BASIC.
;*************************************************
STX $A5AE
LDA #<TOKSTR2
STA $A5AF
LDA #>TOKSTR2
STA $A5B0
...
;*************************************************
;** Routine aggiuntiva per eliminare il bug
;** nel tokenizer.
;*************************************************
TOKSTR2 JSR PUTREG
LDA #$A0
LDX #$9E
JSR TOKSTR
JSR GETREG
LDY #$00
STY $0B
JMP $A5B2
;*************************************************
```

6 Conclusioni

Si è brevemente presentata la storia del BASIC Extender didattico di Lawrence & England, accompagnata da una succinta descrizione di una possibile espansione di notevole interesse per l'apprendimento e lo studio dell'Assembly. Si è anche illustrata l'insorgenza di un bug progettuale del tutto imprevisto e la sua risoluzione tramite reverse engineering, sempre lecitamente consentito quando si tratti di garantire l'interoperabilità del codice.

Il codice Assembly allegato, predisposto per il moderno ambiente di cross-development CBM Prg Studio, viene messo a disposizione per lo studio e la sperimentazione, con l'augurio che molti lettori studiandolo si sentano invogliati a progettare ed implementare nuovi comandi per migliorare i propri skill in Assembly e comprendere meglio l'interazione col codice BASIC e Kernal presenti in ROM. Il codice è brevemente commentato in alcuni punti salienti e quasi tutti gli indirizzi di memoria significativi (originariamente tutti hardcoded, senza distinzioni) sono referenziati tramite label descrittive, ma ci si attende comunque che il lettore studi il sorgente avendo a disposizione almeno i testi elencati in bibliografia per una comprensione puntuale.

Riferimenti bibliografici:

[1] Dan Heeb, Vic and Commodore 64 Tool Kit: Basic (Compute!, 1984).
[2] Dan Heeb, Compute!'s Vic-20 and Commodore 64 Tool Kit: Kernal (Compute, 1985).
[3] David Lawrence and Mark England, Commodore 64 Machine Code Master: A Library of Machine Code Routines (Reston Pub Co, 1983).
[4] David Lawrence and Mark England, Il linguaggio macchina del Commodore 64. Con floppy disk (Jackson Libri, 1985).
[5] Sheldon Leemon, Mapping the Commodore 64 & 64C (Compute, 1987).

Link Utili:

Segnaliamo il sito https://sys64738.org, in modo che i lettori interessati possano scaricare direttamente lo ZIP con i sorgenti predisposti per CBM Prg Studio.

Il calcolo della data della Pasqua

di Alberto Apostolo

Nel IV secolo d.C. l'unità dell'Impero Romano era stata affidata alla Chiesa cristiana che abolì i riti pagani insieme al calendario romano, istituendo al loro posto un calendario basato su calcoli ecclesiastici e sulla commemorazione dei Santi. Uno degli scopi Concilio di Nicea (odierna Iznik in Turchia) tenuto nel 325 d.C., era quello di determinare un'unica data per celebrare la Pasqua da parte dei cristiani dell'Impero Romano d'Occidente e d'Oriente. Secondo il calendario ebraico tale data era fissata al quattordicesimo giorno del mese di Nisan, che iniziava con la comparsa della falce della Luna di primavera, quella più vicina all'equinozio di primavera.

Ma il calendario ebraico su base lunare non era preciso nel rispettare l'equinozio. Pertanto si faceva ricorso ad una approssimazione, fissando la data ad una domenica. Inoltre la Chiesa stabiliva che la Pasqua cristiana non dovesse cadere nella stessa data della Pasqua ebraica.

I computisti addetti ai calcoli del calendario cristiano ritenevano che il conflitto si potesse evitare scegliendo la prima domenica successiva dopo il plenilunio successivo all'equinozio di primavera. Tale problema divenne connesso al problema astronomico di stabilire il punto dell'equinozio di primavera dello Zodiaco [Ave93].

Essi ricavarono che la Pasqua cristiana poteva ricadere in qualsiasi giorno tra il 22 Marzo e il 25 Aprile e ci sarebbero voluti oltre cinque milioni di anni affinché la Pasqua cristiana coincidesse con quella ebraica. Avendo poi scoperto che era impossibile avere una formula matematica per determinare le date future, i computisti crearono una serie di tabelle di "epatte" (dal greco "epaktos" = "aggiunto", fonte www.treccani.it). Una particolareggiata trattazione di tali tabelle si trova sul testo del Cohn [Coh07].

Tuttavia il vero moto lunare era troppo complesso per formulare esattamente le tabelle e il calendario in uso fino a quel momento stava accumulando errori. Nel XVI secolo, la recessione dell'anno reale rispetto al calcolo artificiale era arrivata a 11 giorni e la Pasqua cadeva sempre più tardi nella stagione e quindi era sempre più calda.

Nel 1582 Papa Gregorio XIII convocò una commissione per riesaminare la questione della riforma del calendario.

Occorrevano due azioni. La prima, assicurare che in futuro la Pasqua arrivasse nel punto giusto dell'anno stagionale portando l'equinozio alla corretta collocazione nel ciclo annuale. La seconda, trovare un meccanismo per mantenerlo fisso.

Dopo lunghi dibattiti la questione fu risolta in modo drastico come avvenuto con la riforma di Giulio Cesare. Nel 1582 con la bolla papale "Inter Gravissimas" furono accolte le modifiche indicate dagli studi di Clavius e Lilius (nomi latini con cui sono maggiormente noti i due scienziati al servizio dello Stato Pontificio, vedi Appendice).

L'equinozio di primavera fu spostato dall'11 Marzo 1582 al 21 Marzo 1582 e si decretò che il giorno successivo al 4 Ottobre 1582 fosse il 15 Ottobre 1582 [Pap12]. Poi fu modificata la regola degli anni bisestili per gli anni divisibili per 100 che dovevano essere bisestili solo se erano anche divisibili per 400 (per es. 1600 e 2000 lo sono mentre non lo sono 1700, 1800, 1900). Con la riforma la durata media dell'anno era calcolata a 365,2425 giorni (circa 0.003 in più vero anno stagionale). Il ciclo annuale calcolato dall'uomo avrebbe preceduto le stagioni di un giorno in 3300 anni circa. La riforma gregoriana venne adottata subito dalle nazioni cattoliche e contrastata da quelle a prevalenza protestante e non considerata dalle altre culture non occidentali. La Gran Bretagna e le sue colonie adottarono il calendario solo a partire dal 1752 (eliminando 11 giorni dal computo). La Russia lo adottò dopo la Rivoluzione del 1917 (eliminando un numero maggiore di giorni).

Per migliorare ulteriormente il calendario gregoriano sono state suggerite nei secoli successivi diverse piccole riforme. Quella del Congresso Ortodosso Orientale di Costantinopoli (1923), affermava che gli anni divisibili per 900 dovessero essere bisestili solo se il resto fosse stato 200 o 600 (portando la discrepanza del calendario a un giorno ogni 44000 anni circa) [Ave93].

Con l'avvento dei computer e dei linguaggi di programmazione sono state realizzate diverse versioni del calcolo della Pasqua. Occorre però fare attenzione perché in alcuni testi non viene specificato il range di anni per i quali l'input è valido. Altra cosa a cui prestare altrettanta attenzione è la modalità di calcolo dei vari quozienti coinvolti nell'algoritmo e l'uso delle funzioni che calcolano la parte intera di un numero, verificando come sono definite senza fidarsi ciecamente di nomi "familiari" come INT(x) o simili perché ogni autore potrebbe avere introdotto una sua personale definizione (per esempio Cohn [Coh07]).

Chi non vuole calcolare ogni volta la data della Pasqua e di altre festività, può ricorrere a tabelle speciali che riportano calendari per un certo numero di anni.

In questo articolo viene presentato il programma "Pasqua" scritto in linguaggio C, ricavato dai testi di Redfern e Knuth e vale per anni y tali che 1582 < y < 4903. Nel programma si effettua la divisione con aritmetica intera e il quoziente viene troncato.

```
/*********************************************************/
/* Programma: Pasqua                                     */
/* Funzione : Calcolo data Pasqua [Red87],[Knu05] per 1582 < y < 4903 */
/*          ( le divisioni sono intere con troncamento )  */
/*********************************************************/
#include <stdio.h>
#include <conio.h>     /* tastiera-video PC IBM amb. Windows MS-DOS */
/* Programma principale                                  */
int main(void)
{
   long y;              /* anno in input                 */
   long c,g,x,z,d,e,n,month;   /* variabili di lavoro    */
```

```
/*                                          */
    _clrscr();              /* pulisci schermo          */
    printf("Programma Pasqua : INIZIO ELABORAZIONE\n");
/*                                          */
    for (y=2004;y<2024;y++) {
      g = y % 19 +1;        /* numero d'oro (ciclo di Metone,  */
                            /* astronomo ateniese V sec. b.C.) */
      c = y / 100 + 1;      /* calcolo del secolo       */
      x = (3*c)/4 - 12;     /* considera gli anni non bisestili*/
      z = (8*c +5)/25 - 5;  /* sincronizza con il moto lunare  */
      d = (5*y)/4 - x - 10; /* calcolo della Domenica   */
      e = (11*g + 20 + z - x) % 30;
      if (((e == 25) && (g > 11)) || (e == 24)) {
        e = e + 1;
      }                     /* calcolo epatta           */
      n = 44 - e;
      if (n < 21) {
        n = n + 30;
      }                     /* calcolo giorno di Luna piena   */
      n = n + 7 - ((d + n) % 7);/* avanza alla Domenica successiva */
      if (n > 31) {
        month = 4;          /* Pasqua nel mese di Aprile    */
        n = n - 31;
      } else {
        month = 3;          /* Pasqua nel mese di Marzo     */
      }
      printf("\n Il %2.2ld/%2.2ld/%4.4ld e' Pasqua!",n,month,y);
    }

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

Anche su Excel è possibile calcolare la data della Pasqua. E' necessario che il sistema usato per le date sia quello denominato 1900 (se è quello 1904 come nelle versioni pre-2011 allora va settato correttamente [EA17]). Dopo avere impostato l'anno nella cella A1, per calcolare la data della Pasqua si può usare la formula:

```
=VALUTA(("4/"&A1)/7+RESTO(19*RESTO(A1;19)-7;30)*14%;)*7-6
```

oppure

```
=ARROTONDA.DIFETTO(GIORNO(MINUTO(A1/38)/2+56)&"/5"&"/"&A1;7)-34
```

Appendice

Si ricorda che in base alla Pasqua sono calcolate le date delle seguenti celebrazioni della Chiesa Cristiana Cattolica [CEI07]:
1) Mercoledì delle Ceneri (inizio della Quaresima, 40 giorni prima di Pasqua)
2) Ascensione di Gesù Cristo (la prima Domenica 40 giorni dopo la Pasqua)
3) Pentecoste (Domenica, 50 giorni dopo la Pasqua)
4) Corpus Domini (seconda Domenica dopo Pentecoste)
5) Le settimane del Tempo Ordinario (prima della Quaresima e dopo la Pasqua).
Nota: le domeniche di Avvento sono calcolate in base al Natale che è sempre il 25 Dicembre.

Christoph Clau (o Christoph Klau), in latino Christophorus Clavius (Bamberg 1538 – Roma 1612) è stato un gesuita, preminente matematico e insegnante. Laureatosi a Coimbra (Portogallo) insegnò matematica e astronomia al Collegio Romano a partire dal 1564. Nel 1570 pubblicò "Commentarius in Sphaeram Joannis de Sacro Bosco" che influenzò scienziati come Tycho Brahe, Johannes Kepler e Galileo Galilei. Un commentario di Clavius sugli Elementi di Euclide (pubblicato nel 1574) fu studiato da René Descartes al Collegio di La Flèche [Nol15].

Alcune curiosità:

1) secondo alcuni, Christoph Clau potrebbe essere una semplice traduzione di Christoph Schlüssel (che in tedesco significa chiave, come clavis in latino) [Nol15].

2) uno dei più grandi crateri lunari da impatto è stato intitolato a Clavius [Wlaoo].

Luigi Lilio, in latino Aloysius Lilius (Cirò 1510 – Roma 1574), è stato un medico, astronomo e matematico italiano. Conseguita la laurea in Medicina a Napoli, si trasferì a Roma dove, con l'esperienza scientifica maturata a Napoli concepì e maturò la riforma del calendario (la cui differenza di 26 secondi rispetto all'anno solare corrisponde a circa 3 giorni ogni diecimila anni circa, 1 giorno eccedente ogni 3323 anni) [Pap12]. Si fa notare che Knuth lo considera napoletano perché all'epoca la Calabria (dove si trova Cirò) apparteneva al Regno di Napoli.

Clavius a sinistra e **Lilius** a destra - fonte Google

Bibliografia

[Ave93] A. Aveni, "Gli imperi del tempo. Calendari, orologi e culture", Vol. 33 di Storia e civiltà, EDIZIONI DEDALO, 1993.
[CEI07] Conferenza Episcopale Italiana, "Liturgia delle Ore", Libreria Editrice Vaticana, 2007.
[Coh07] M. Cohn , "The Mathematics of the Calendar", Lulu.com, 2007.
[EA17] AA.VV., "Formule e funzioni di Excel", Excel Academy, 2017.
[Knu05] D.E. Knuth, "The Art of Computer Programming", Vol. 1, Addison-Wesley Professional, 2005.
[Nol15] L. Nolan, "The Cambridge Descartes Lexicon", Cambridge University Press, 2015.
[Pap12] AA.VV., "I Papi della Memoria: La storia di alcuni grandi Pontefici che hanno segnato il cammino della Chiesa e dell'Umanità", Gangemi Editore spa, 2012.
[Red87] E.J. Redfern, "Introduction to PASCAL", Macmillan International Higher Education, 1987.
[Wlaoo] P. Wlasuk, "Observing the Moon", Springer Science & Business Media, 2000.

Grafica... Che passione!

Parte II: hi-res multicolor mode

di Marco Pistorio

Benvenuti a questa seconda parte dell'articolo "Grafica..che passione!". Stavolta ci occuperemo di convertire immagini a colori in schermate in alta risoluzione "multicolor mode" del C64.

Come ottenere tale risultato? Anche in questo caso effettueremo una scansione dell'immagine pixel per pixel, tenendo presente ovviamente che la risoluzione massima di una schermata hi-res multicolor mode per il C64 è di 160x200 punti. Pertanto l'immagine di partenza dovrà essere impostata alla stessa risoluzione, ovvero 160x200. Altro elemento da tenere in considerazione sono i colori. I migliori risultati di conversione si ottengono su immagini che adoperano pochi colori al loro interno. Ciò in quanto il C64, in questa particolare modalità grafica, permette di visualizzare fino a 4 colori per ciascuna cella di schermo, di dimensioni 4x8 pixel. Di questi 4 colori, uno sarà il colore di sfondo, comune a tutte le celle, mentre gli altri 3 potranno essere uno qualsiasi dei 16 colori disponibili nella palette del c64. Le celle che comporranno l'immagine saranno 1000, ovvero 40 in orizzontale e 25 in verticale. Ogni cella sarà composta da 4 pixel in orizzontale per 8 in verticale. Da qui, sviluppando i calcoli, avremo il numero totale delle celle, 40*25 ovvero 1000, numero analogo al numero totale delle celle a disposizione nella modalità "hires monocolor mode". Come mai, a parità di numero di celle, la risoluzione passa da 320x200 della modalità "hires monocolor mode" alla risoluzione 160x200 di questa modalità "hires multicolor mode"? E' semplice. Ciascun pixel in "hires multicolor mode" è "lungo" esattamente il doppio rispetto ai pixel della modalità monocromatica.
Come viene "calcolato" il colore di ciascun pixel? Questa è la parte più "intrigante" di questa modalità grafica.

Torniamo per qualche istante alla modalità hires monocolor mode. Qui i pixel vengono visualizzati esattamente per come sono stati definiti nell'area di memoria contenente la loro definizione, lunga 8000 bytes e colorati in base al valore contenuto in una seconda area di memoria, lunga 1000 bytes, che conterrà l'attributo del colore relativo a ciascuna delle 1000 celle 40x25 che compongono la schermata.

Ogni bit dell'immagine impostato al valore acceso "1" sarà colorato quindi in base a tale attributo colore. I bit spenti ovvero posti al valore logico "0" avranno il medesimo colore dello sfondo dello schermo, comune a tutte le celle. Nei files generati tramite il tool illustrato nello scorso numero di "RetroMagazine" ho fissato l'area di memoria contenente la mappa dei bit che descrivono l'immagine in alta risoluzione dalla locazione 8192 ($2000 in notazione esadecimale) alla locazione 16191 ($3f3f).

L'area di memoria contenente l'attributo colore per ciascuna delle mille celle di dimensioni 8x8 pixel è stata invece impostata a partire dalla locazione 1024 ($0400 in esadecimale) alla locazione 2023 ($07e7 in esadecimale). Quest'ultima area è riempita con il valore costante "1" che rende tutti i pixel accesi della schermata hi-res di colore bianco. E' possibile impostare tali aree a partire da diversi indirizzi. In rete è possibile ottenere tutte le informazioni necessarie a chi intendesse approfondire tali nozioni

Vediamo adesso cosa succede nel momento in cui intendiamo sfruttare la modalità hires multicolor mode. La risoluzione si dimezza, passando da 320x200 pixel a 160x200. La mappa dei bit che descrive l'immagine viene interpretata in maniera molto diversa da quanto accadeva nella precedente modalità. In pratica i bit non vengono più considerati singolarmente, bensì a coppie! Ciascuna di queste coppie di bit verrà visualizzata sullo schermo come un pixel lungo il doppio rispetto al pixel della precedente risoluzione. Il colore di tale pixel viene determinato in base alla seguente tabella:

Coppia	Colore
"00"	Colore di sfondo
"10"	Colore #1
"01"	Colore #2
"11"	Colore base

Viene adoperata l'area di memoria colore, dalla locazione 55296 ($D800 in esadecimale) alla locazione 56295 ($DBE7 in esadecimale) per contenere l'attributo di colore "colore base", comune a tutti i pixel di ciascuna cella 4x8 pixel che forma l'immagine.
L' area di memoria schermo, dalla locazione 1024 ($0400 in esadecimale) alla locazione 2023 ($07e7 in esadecimale), lunga anch'essa 1000 bytes, sarà destinata invece a contenere gli altri due attributi di colore (colore #1 e colore #2 in tabella), che saranno anch'essi comuni a tutti i pixel di ciascuna delle cella di 4x8 pixels che andranno a comporre l'immagine da visualizzare.
Il dato sarà memorizzato nella forma: **colore#1*16 + colore #2.**

Funzionamento del tool

Come dicevamo all'inizio di questa "chiacchierata", 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 nell'articolo sullo scorso numero non sono presenti meccanismi di controllo nè della presenza né della validità della immagine che viene elaborata.
A ciascun pixel verrà associato uno dei 16 colori presenti nella palette del C64.

Ma, partendo da un valore di colore espresso in notazione RGB, come fare per ottenere il colore corrispondente C64?

Ho risolto questo problema scrivendo due funzioni apposite, ovvero getclosestColorIndex() e ColorDistance(). Il loro funzionamento è abbastanza semplice.

Ottengo un valore numerico che mi esprime la "distanza" tra un qualsiasi colore rispetto al colore nero. Tale valore è ottenuto come la somma dei quadrati delle differenze tra le componenti Rosso, Verde e Blu del colore che intendo valutare rispetto alle stesse componenti del colore nero. Tale calcolo avviene dentro la funzione ColorDistance(), precisamente. Poi valuto le differenze tra tale valore che ho ottenuto e quello relativo ai 16 colori della tavolozza del C64, escludendo il nero. Il valore più basso che otterrò mi fornirà il colore che assocerò al pixel in esame, colore che rappresenterò mediante un indice intero che mi esprimerà appunto tale colore. Questo è il compito della funzione getClosestColorIndex().

Non intendo affermare che il metodo utilizzato sia infallibile. Non lo è, naturalmente. La distanza di colore può trovarsi in bilico tra due colori di palette diversi ed un colore può finire quindi per essere valutato (e poi rappresentato) con un colore totalmente diverso da quello di partenza. Esistono altri metodi e di certo non mancherà ai lettori più "creativi" impiegarne proficuamente uno più performante ☺

Dopo questa fase di "riconoscimento" del colore di ciascun pixel che forma l'immagine, che avviene invocando la procedura che ho chiamato pre_elab(), visualizzo il risultato ottenuto, ridisegnando l'immagine elaborata sfruttando i colori "riconosciuti", pixel per pixel. Tale lavoro è affidato alla procedura view_image().
Già in questa prima fase ci potremo rendere conto della "bontà" del riconoscimento effettuato.

A questo punto, la procedura calc_cellcolors() calcolerà quante volte ciascuno dei 16 potenziali colori è presente in ognuna delle 1000 celle 4x8 pixel che formano l'immagine da rappresentare. In questa fase forzo a 0 il numero di occorrenze relativo ai pixel che hanno lo stesso attributo colore di quello di background.

La chiamata alla procedura calc_cellcolors_normalized() popolerà un array di 1000 elementi x 3. Tale array conterrà i tre colori che verranno adoperati per ciascuna cella 4x8 pixel che comporrà l'immagine finale, rammentando che il quarto colore permesso, comune a tutte le celle, sarà quello di background. Con quale criterio sono stati ottenuti questi 3 colori?
Si tratta semplicemente dei 3 colori presenti più volte per ciascuna cella, cioè i 3 colori più diffusi. Più semplice di così... ☺
Gli altri colori eventualmente presenti in ciascuna cella non vengono quindi presi in considerazione in questa fase.

La procedura chiamata successivamente, make_data(), analizzerà l'immagine di partenza, confrontando il colore di ciascun pixel da mostrare con quello presente nell'array dei colori "normalizzati" ovvero "ridotti", popolato durante la chiamata alla precedente funzione. Se il colore del pixel dovesse corrispondere ad uno dei 3 colori memorizzati in tale array, il pixel verrebbe inserito nell'area di memoria contenente la mappa dei bit che rappresenta l'immagine, secondo una specifica codifica: **"11"** se il pixel risultasse del colore corrispondente al il primo colore più diffuso nella cella, **"10"** se il pixel risultasse del colore corrispondente al secondo colore più diffuso nella cella, **"01"** se il pixel risultasse del colore corrispondente al terzo colore più diffuso nella cella, **"00"** se il pixel risultasse dello stesso colore dello sfondo dello schermo. Nel caso infine il pixel risultasse di un colore non presente tra quelli disponibili nell'array dei 3 colori "normalizzati" relativamente alla cella in esame, e se il colore non fosse quello di background, verrebbe trattato come se si trattasse di un pixel con il colore più diffuso all'interno della cella in elaborazione. Ciò ovviamente crea una rappresentazione dell'immagine "degradata" in termini di colori, tanto più degradata quanto più si sia manifestata l'impossibilità di impostare per i vari pixel il loro corretto colore, colorandoli bensì del colore più diffuso all'interno della cella di appartenenza. Per ciascuna cella possiamo adoperare sempre e solo 4 colori, ovvero 3 più di 1 background, comune a tutte le celle che compongono la schermata.
Tali informazioni saranno via via memorizzate nel vettore mem_hires().
Dopo aver descritto la bitmap di ciascuna cella, ne calcoliamo il byte che colorerà i relativi pixel di tipo "10" e "01" con i due attributi di colore colore#1 e colore#2. Il calcolo sarà effettuato, cella per cella, secondo la formula già esposta colore#1*16+colore#2, ed il risultato verrà memorizzato nel vettore mem_screen(). E' evidente che gli attributi colore#1 e colore#2 saranno probabilmente diversi di cella in cella, perché in ciascuna cella la diffusione dei vari colori potrà essere diversa. Nel contempo, memorizziamo anche l'attributo del colore base, che colorerà i pixel nella cella di tipo "11", anch'esso determinato cella per cella. Anche per questo valore valgono le stesse considerazioni appena fatte. Può essere diverso da una cella ad un'altra, in quanto il colore più diffuso in una certa cella potrebbe differire dal colore più diffuso nella cella successiva. Questi valori che esprimono il colore base, cella per cella, verranno memorizzati nell'apposito vettore mem_colors().

Infine, invocando la funzione write_data(), registriamo i dati contenuti nei vettori mem_hires(), mem_screeen() e mem_colors() nei rispettivi files "hires.dat", "colors.dat" e "screen.dat."

Uso dei files generati dal tool

Come adoperare i files generati dal tool? Si può creare un file .d64 vuoto ed inserire al suo interno i files "hires.dat", "screen.dat" ed "hires.dat". Quindi basterà caricare e lanciare l'apposito caricatore in BASIC. Di seguito il codice relativo:

Programma BASIC per la visualizzazione della immagine grafica generata dal tool

```
1 poke53280,0:poke53281,0
10 poke 53272, peek(53272) or 8 :rem bitmap at 8192
20 poke 53265, peek(53265) or 32: rem bitmap on
30 poke 53270, 24: rem multicolor mode
40 if jk=0 then jk=1: load"hires.dat",8,1
50 if jk=1 then jk=2: load"screen.dat",8,1
55 if jk=2 then jk=3: load"colors.dat",8,1
60 get a$: if a$=""then60
70 poke 53272,21: poke53265,27
75 poke 53270,200
80 print chr$(147);
90 end
```

Chi di voi invece prediligesse l'assembly, ecco il relativo codice, con sintassi KickAssembler, che adempie al medesimo compito.

Codice Assembly per la visualizzazione della immagine grafica generata dal tool

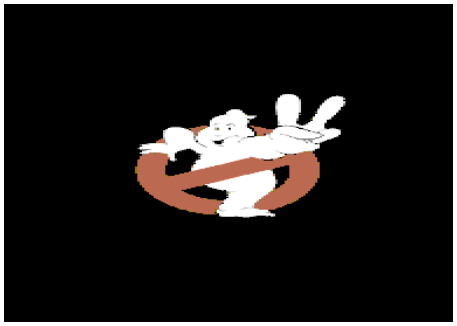
```
*= $1000

:BasicUpstart2(start)

start:
        lda #0
        sta $d020

        lda #1
        sta $d021

        lda $d018
        ora #$08
        sta $d018

        lda $d011
        ora #$20
        sta $d011

        lda #$18  //multicolor mode on
        sta $d016//

        // riempie la memoria schermo
        // (locazioni 1024-2023)
        // con i dati di -screen-

        ldx #$00
loop:
        lda $5000,x
        sta $0400,x

        lda $5100,x
        sta $0500,x

        lda $5200,x
        sta $0600,x

        lda $52E8,x
        sta $06E8,x
        inx         //Increment
accumulator until 256 bytes read
        bne loop

        //////////////////////////

        // riempie la memoria colore
        // (locazioni 55296-56295)
        // con i dati di -colors-

        ldx #$00
loop2:
        lda $4000,x
        sta $d800,x

        lda $4100,x
        sta $d900,x

        lda $4200,x
        sta $da00,x

        lda $42E8,x
        sta $daE8,x
        inx         //Increment
accumulator until 256 bytes read
        bne loop2

        jmp *

.pc=$2000-2
.import binary "hires.dat"

.pc=$4000-2
.import binary "colors.dat"

.pc=$5000-2
.import binary "screen.dat"
```

Conclusioni finali

Per lanciare il tool basterà fare doppio click sull'eseguibile "tool_bitmaps_4_c64_multicolors.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 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 ed insieme ad ulteriore materiale.
Il file da convertire dovrà chiamarsi, come già specificato, **immagine.bmp** e verranno prodotti dal tool tre files, **hires.dat, colors.dat** e **screen.dat**, che verranno sovrascritti di volta in volta se già presenti.
Il tool è stato pubblicato su GITHUB all'indirizzo di seguito specificato: https://github.com/marcus73/retromagazine_04
Ho ritenuto inutile esporlo anche nell'articolo. Potrete esaminare il codice sorgente del tool direttamente da voi. Il codice 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 3 files .dat potranno essere sfruttati, sia in BASIC che in assembly secondo quanto esposto.

In questa pagina potrete osservare alcune delle immagini che ho convertito e poi esportato da VICE a PC.
I risultati mi sembrano discreti, considerando oltretutto che trattasi di immagini visualizzate su C64! I risultati migliori li otterrete convertendo files bitmap con pochi colori. Suggerisco di lavorare con immagini fino a 16 colori, e con una densità di colore per cella non superiore a 4 colori presenti (1 è quello di background, comune a tutte le celle, ribadisco) .
Parleremo di conversioni ancora più accurate nella terza parte di questo articolo "Grafica, che passione!", sul prossimo numero di "RetroMagazine", dove scopriremo una particolare modalità grafica, denominata FLI.
Non mi resta che salutarvi, amici lettori. Alla prossima!



Esplorando l'Amiga - parte 6

di Leonardo Giordani

Inizializzazione della memoria

L'Amiga ha due tipi di memoria fisica. Il primo è sempre presente sulla scheda madre ed è tradizionalmente chiamato "Chip Memory". Il nome deriva dal fatto che sia la CPU che i chip custom hanno accesso ad essa, il che significa anche che tutti questi componenti condividono l'accesso. Questo rallenta la CPU quando sia essa che i chip custom accedono alla memoria, vanificando parzialmente il contributo che i chip custom danno all'architettura del sistema. I chip custom possono accedere alla memoria direttamente tramite DMA, quindi senza bloccare la CPU, ma la memoria stessa non può essere letta da più di un componente allo stesso momento.

Il secondo tipo di memoria è chiamata "Fast Memory" proprio perché la CPU ha un accesso esclusivo ad essa, che quindi darà performance migliori. È anche spesso chiamata "memoria espansa" in quanto viene aggiunta all'Amiga tramite schede di espansione.

Ad un certo punto durante il boot del sistema, Kickstart identifica il tipo di memoria installato sulla macchina e mette la dimensione della memoria espansa in `a4`. Se questo registro contiene un valore diverso da zero, quindi, Kickstart sa che una espansione di memoria è installata e configura la memoria in modo differente.

Il codice che inizializza la memoria è il seguente

```
00000380: 200c                  move.l  a4,d0
00000382: 6724                  beq.b   0x3a8
00000384: 41ee 024c             lea     0x24c(a6),a0
00000388: 43fa ffa8             lea     0x332(pc),a1
0000038c: 7400                  moveq   #0,d2
0000038e: 323c 0005             move.w  #0x5,d1
00000392: 200c                  move.l  a4,d0
00000394: 9088                  sub.l   a0,d0
00000396: 0480 0000 1800        subi.l  #0x1800,d0
0000039c: 6100 1688             bsr.w   0x1a26
000003a0: 41f8 0400             lea     0x400.w,a0
000003a4: 7000                  moveq   #0,d0
000003a6: 600a                  bra.b   0x3b2
000003a8: 41ee 024c             lea     0x24c(a6),a0
000003ac: 203c ffff e800        move.l  #-0x1800,d0
000003b2: 323c 0003             move.w  #0x3,d1
000003b6: 2448                  movea.l a0,a2
000003b8: 43fa ff6c             lea     0x326(pc),a1
000003bc: 74f6                  moveq   #-0xa,d2
000003be: d08b                  add.l   a3,d0
000003c0: 9088                  sub.l   a0,d0
000003c2: 6100 1662             bsr.w   0x1a26
000003c6: 224e                  movea.l a6,a1
000003c8: 6100 107e             bsr.w   0x1448
```

Come ho fatto negli articoli precedenti dividerò il codice in gruppi e rimpiazzerò alcuni indirizzi con etichette per rendere la routine più comprensibile


```
Check_expansion:
00000380: 200c                  move.l  a4,d0
00000382: 6724                  beq.b   Only_chip

Add_expansion:
00000384: 41ee 024c             lea     0x24c(a6),a0
00000388: 43fa ffa8             lea     0x332(pc),a1
0000038c: 7400                  moveq   #0,d2
0000038e: 323c 0005             move.w  #0x5,d1
00000392: 200c                  move.l  a4,d0
00000394: 9088                  sub.l   a0,d0
00000396: 0480 0000 1800        subi.l  #0x1800,d0
0000039c: 6100 1688             bsr.w   0x1a26
000003a0: 41f8 0400             lea     0x400.w,a0
000003a4: 7000                  moveq   #0,d0
000003a6: 600a                  bra.b   Add_chip

Only_chip:
000003a8: 41ee 024c             lea     0x24c(a6),a0
000003ac: 203c ffff e800        move.l  #-0x1800,d0

Add_chip:
000003b2: 323c 0003             move.w  #0x3,d1
000003b6: 2448                  movea.l a0,a2
000003b8: 43fa ff6c             lea     0x326(pc),a1
000003bc: 74f6                  moveq   #-0xa,d2
000003be: d08b                  add.l   a3,d0
000003c0: 9088                  sub.l   a0,d0
000003c2: 6100 1662             bsr.w   0x1a26
000003c6: 224e                  movea.l a6,a1
000003c8: 6100 107e             bsr.w   0x1448
```

I due indirizzi `0x326` e `0x332` utilizzati in questo codice contengono le stringhe `Chip Memory` e `Fast Memory` (zero-terminated)

```
; ###########################################################
; 'Chip Memory' string
```

```
00000326: 43 ; C
00000327: 68 ; h
00000328: 69 ; i
00000329: 70 ; p
0000032a: 20 ; SP
0000032b: 4d ; M
0000032c: 65 ; e
0000032d: 6d ; m
0000032e: 6f ; o
0000032f: 72 ; r
00000330: 79 ; y
00000331: 00 ; NUL

;############################################################
; 'Fast Memory' string

00000332: 46 ; F
00000333: 61 ; a
00000334: 73 ; s
00000335: 74 ; t
00000336: 20 ; SP
00000337: 4d ; M
00000338: 65 ; e
00000339: 6d ; m
0000033a: 6f ; o
0000033b: 72 ; r
0000033c: 79 ; y
0000033d: 00 ; NUL
```

Analizziamo il codice riga per riga

```
Check_expansion:
00000380: 200c                    move.l  a4,d0
00000382: 6724                    beq.b   Only_chip
```

La prima cosa che Kickstart fa è di controllare se il sopramenzionato registro `a4` contiene un valore diverso da zero, in modo da decidere se l'espansione di memoria è presente o meno. Se `a4` è zero il codice salta direttamente all'inizializzazione della chip memory.

```
Add_expansion:
00000384: 41ee 024c               lea     0x24c(a6),a0
00000388: 43fa ffa8               lea     0x332(pc),a1
```

Il codice poi carica due indirizzi effettivi, la prima posizione libera in memoria (`0x24c`) e la stringa `Fast Memory`. La ragione dietro il numero magico `0x24c` è spiegata più avanti in dettaglio. Lo scopo del codice è di chiamare la routine `AddMemList`, che aggiunge memoria alla memoria libera di sistema. Il prototipo della routine è il seguente

```
size = AddMemList(size, attributes, pri, base, name)
D0                D0    D1          D2   A0    A1
```

dove `size` è la dimensione della memoria in bytes, `attributes` contiene delle flag che specificano attributi della memoria stessa, `pri` è la priorità di questa zona di memoria, `base` è l'indirizzo base e `name` è il nome di questa specifica lista.

```
0000038c: 7400                    moveq   #0,d2
0000038e: 323c 0005               move.w  #0x5,d1
```

Il codice quindi prosegue preparando i registri per la chiamata di `AddMemList`. Dà alla memoria priorità o e imposta le flag degli attributi a `0x5`, ovvero `101`, che si traduce in `PUBLIC` e `FAST` (vedi `include_i/exec/memory.i`).

```
00000392: 200c                    move.l  a4,d0
00000394: 9088                    sub.l   a0,d0
00000396: 0480 0000 1800          subi.l  #0x1800,d0
0000039c: 6100 1688               bsr.w   0x1a26
```

L'indirizzo base della memoria espansa viene copiato in `do` (a mio parere un'inutile ripetizione di quanto il codice ha fatto 6 linee prima). Successivamente sottrae l'indirizzo della prima locazione libera (per tenere in conto eventuali dati già presenti in memoria) e la dimensione dello stack, che è stato inizializzato in precedenza da Kickstart a 6KiB (fisso). Dopo aver fatto questo il codice salta a `AddMemList` (`0x1a26`).

Quando la routine finisce, il codice prosegue con l'inizializzazione della chip memory. Questa deve venir inizializzata in due modi differenti a seconda della presenza o memo della memoria espansa, in quanto quest'ultima è preferibilmente utilizzata dalla CPU.

```
000003a0: 41f8 0400               lea     0x400.w,a0
000003a4: 7000                    moveq   #0,d0
000003a6: 600a                    bra.b   Add_chip

Only_chip:
000003a8: 41ee 024c               lea     0x24c(a6),a0
000003ac: 203c ffff e800          move.l  #-0x1800,d0
```

Se il test sulla presenza della memoria espansa fallisce, il codice salta direttamente a `0x03a8`. Se invece l'espansione è già stata installata la CPU esegue il codice a `0x03a0` e poi salta a `0x03b2` (Rinominato `Add_chip` qui).

```
000003a0: 41f8 0400               lea     0x400.w,a0
```

Quindi se c'è un'espansione di memoria, Exec sarà caricata lì, il che significa che sia la libreria che lo stack di sistema non sono nella chip memory. Possiamo quindi aggiungere alla memoria di sistema tutto lo spazio dopo `0x400` (maggiori dettagli su questo numero più avanti).

```
000003a4: 7000                    moveq   #0,d0
000003a6: 600a                    bra.b   Add_chip
```

Siccome lo stack è già stato creato in fast memory possiamo mettere un 0 in `do` e saltare al codice che aggiunge la memoria alle liste di sistema. Se l'espansione non è presente, invece, Exec è installata nella chip memory, per cui dobbiamo calcolare l'indirizzo base come abbiamo fatto per la fast memory.

```
Only_chip:
000003a8: 41ee 024c               lea     0x24c(a6),a0
000003ac: 203c ffff e800          move.l  #-0x1800,d0
```

Qui viene caricato l'indirizzo effettivo della prima locazione libera, `0x24c` byte dopo l'indirizzo ExecBase, e sottraiamo alla dimensione i 6 KiB di spazio per lo stack di sistema (negativo in quanto il codice lo somma alla dimensione totale).

```
Add_chip:
000003b2: 323c 0003               move.w  #0x3,d1
000003b6: 2448                    movea.l a0,a2
```

```
000003b8: 43fa ff6c                 lea     0x326(pc),a1
000003bc: 74f6                      moveq   #-0xa,d2
000003be: d08b                      add.l   a3,d0
000003c0: 9088                      sub.l   a0,d0
000003c2: 6100 1662                 bsr.w   0x1a26
```

La procedura seguita per la fast memory viene qui ripetuta. Gli attributi sono adesso `CHIP` e `PUBLIC` (`0x3`), la stringa a `0x326` è `Chip Memory`, e la priorità è -10 (`-0xa`). Il registro `a3` contiene già l'ultimo indirizzo della chip memory, per cui lo aggiungiamo a `d0` e sottraiamo la prima locazione libera calcolata precedentemente. Come prima, la routine chiama `AddMemList` a `0x1a26`.

```
000003c6: 224e                      movea.l a6,a1
000003c8: 6100 107e                 bsr.w   0x1448
```

L'ultima azione di questa parte di codice è di chiamare `AddLibrary` a `0x1448`. L'unico parametro che la routine richiede è l'indirizzo base della libreria da aggiungere (in `a1`), motivo per cui il codice copia `a6` in quel registro.

Il "numero magico" `0x24c`

Quando abbiamo discusso il modo in cui la memoria viene inizializzata abbiamo scoperto un "numero magico" che Kickstart usa per trovare la prima locazione libera in memoria

```
00000384: 41ee 024c                 lea     0x24c(a6),a0
```

Il primo indirizzo libero secondo questo codice è `0x24c` (588) byte dopo l'indirizzo base di Exec. La ragione dietro a questo numero è semplice: quando la libreria Exec viene installata le sue strutture usano esattamente 588 byte, pertanto questo è l'indirizzo del primo spazio libero.

È facile calcolare questo numero. Riporto qui una versione annotata della struttura `ExecBase` che ho già utilizzato nell'articolo precedente. La struttura è descritta nel file `include_i/exec/execbase.i`, e ho aggiunto l'indirizzo relativo di ogni campo. La prima colonna è l'indirizzo nella struttura `ExecBase`, mentre il secondo parte da `0x22`, che è la dimensione delle strutture `LN` e `LIB` che precedono `ExecBase`, e che ho descritto nel quarto articolo di questa serie.

```
0000 0022    UWORD   SoftVer
0002 0024    WORD    LowMemChkSum
0004 0026    ULONG   ChkBase
0008 002a    APTR    ColdCapture
000c 002e    APTR    CoolCapture
0010 0032    APTR    WarmCapture
0014 0036    APTR    SysStkUpper
0018 003a    APTR    SysStkLower
001c 003e    ULONG   MaxLocMem
0020 0042    APTR    DebugEntry
0024 0046    APTR    DebugData
0028 004a    APTR    AlertData
002c 004e    APTR    MaxExtMem

0030 0052    WORD    ChkSum


******* Interrupt Related
*****************************************
    LABEL   IntVects
0032 0054    STRUCT  IVTBE,IV_SIZE
003e 0060    STRUCT  IVDSKBLK,IV_SIZE
004a 006c    STRUCT  IVSOFTINT,IV_SIZE
0056 0078    STRUCT  IVPORTS,IV_SIZE
0062 0084    STRUCT  IVCOPER,IV_SIZE
006e 0090    STRUCT  IVVERTB,IV_SIZE
007a 009c    STRUCT  IVBLIT,IV_SIZE
0086 00a8    STRUCT  IVAUD0,IV_SIZE
0092 00b4    STRUCT  IVAUD1,IV_SIZE
009e 00c0    STRUCT  IVAUD2,IV_SIZE
00aa 00cc    STRUCT  IVAUD3,IV_SIZE
00b6 00d8    STRUCT  IVRBF,IV_SIZE
00c2 00e4    STRUCT  IVDSKSYNC,IV_SIZE
00ce 00f0    STRUCT  IVEXTER,IV_SIZE
00da 00fc    STRUCT  IVINTEN,IV_SIZE
00e6 0108    STRUCT  IVNMI,IV_SIZE


******* Dynamic System Variables
*************************************

00f2 0114    APTR    ThisTask

00f6 0118    ULONG   IdleCount
00fa 011c    ULONG   DispCount
00fe 0120    UWORD   Quantum
0100 0122    UWORD   Elapsed
0102 0124    UWORD   SysFlags
0104 0126    BYTE    IDNestCnt
0105 0127    BYTE    TDNestCnt

0106 0128    UWORD   AttnFlags

0108 012a    UWORD   AttnResched
010a 012c    APTR    ResModules
010e 0130    APTR    TaskTrapCode
0112 0134    APTR    TaskExceptCode
0116 0138    APTR    TaskExitCode
011a 013c    ULONG   TaskSigAlloc
011e 0140    UWORD   TaskTrapAlloc


******* System List Headers (private!)
*********************************

0120 0142    STRUCT  MemList,LH_SIZE
012e 0150    STRUCT  ResourceList,LH_SIZE
013c 015e    STRUCT  DeviceList,LH_SIZE
014a 016c    STRUCT  IntrList,LH_SIZE
0158 017a    STRUCT  LibList,LH_SIZE
0166 0188    STRUCT  PortList,LH_SIZE
0174 0196    STRUCT  TaskReady,LH_SIZE
0182 01a4    STRUCT  TaskWait,LH_SIZE
0190 01b2    STRUCT  SoftInts,SH_SIZE*5

01e0 0202    STRUCT  LastAlert,4*4

01f0 0212    UBYTE   VBlankFrequency
01f1 0213    UBYTE   PowerSupplyFrequency

01f2 0214    STRUCT  SemaphoreList,LH_SIZE
```

```
0200 0222    APTR    KickMemPtr
0204 0226    APTR    KickTagPtr
0208 022a    APTR    KickCheckSum

020c 022e    UBYTE   ExecBaseReserved[10];
0216 0238    UBYTE   ExecBaseNewReserved[20];
022a 024c    LABEL   SYSBASESIZE
```

Il file include presente sull'Amiga Developer CD riporta la struttura per una versione successiva di Exec, pertanto ha più campi di quelli mostrati qui. Questa versione della struttura può essere trovata in Amiga System Programmer's Guide, pagina 308. Potete facilmente trovare le definizioni di valori come `SH_SIZE` nei file include contenuti nella directory `include_i/exec/`.

Come si nota l'indirizzo finale è `0x24c`, che è esattamente l'indirizzo a cui inizia la memoria libera (si ricordi che `LABEL` è una macro e non un campo della struttura, pertanto non occupa spazio).

I "numeri magici" `0x676` e `0x400`

L'architettura del Motorola 68000 forza il progettista del sistema a riservare i primi 1024 byte (`0x400`) per i vettori delle eccezioni. La tabella che elenca tali vettori può essere trovata nel Programmer's Reference Manual, pagina B-2, e questa è la sorgente del numero magico che utilizziamo quando aggiungiamo la chip memory alle liste di sistema quando è presente un'espansione di memoria.

Si noti ad ogni modo che `0x400` non è l'indirizzo base di Exec, che invece risulta essere `0x676`. Come sappiamo, la libreria è preceduta dalla jump table, e siccome Exec esporta 105 funzioni la tabella sarà lunga `105*6 = 630` byte. Aggiungendo questi 630 byte ai primi 1024 riservati per i vettori delle eccezioni si ottiene 1654 (`0x676`), che per l'appunto è l'indirizzo base della libreria.

```
     +--------------------------------------+
     | First free address                   |
2242 +--------------------------------------+ 0x8c2 <-+
     | ExecBase structure                   |         |
1688 +--------------------------------------+ 0x698   |
     | LIB structure                        |         | Exec base
1668 +--------------------------------------+ 0x684   | structure
     | LN structure                         |         |
1654 +--------------------------------------+ 0x676 <-+
     | Jump vector #105                     |         |
     +--------------------------------------+         |
     | [...]                                |         |
     +--------------------------------------+         | Exec jump
     | Jump vector #2                       |         | vector table
1030 +--------------------------------------+ 0x406   |
     | Jump vector #1                       |         |
1024 +--------------------------------------+ 0x400 <-+
     | End of reserved space                |         |
     +--------------------------------------+         |
     | [...]                                |         |
  12 +--------------------------------------+ 0xc     | 1 Kilobyte
     | Vector #2                            |         | reserved by
   8 +--------------------------------------+ 0x8     | the M68k
     | Reset Initial Program Counter        |         | architecture
   4 +--------------------------------------+ 0x4     |
     | Reset Initial Interrupt Stack Pointer |         |
   0 +--------------------------------------+ 0x0   <-+
```

Nel prossimo articolo

È finalmente ora di mostrare la tabella completa dei vettori di Exec, visto che andremo ad analizzare tutte le funzioni definite in essa. Si discuterà quindi la struttura delle liste di memoria e la relazione con le linked lists. Infine, analizzerò il codice di `AddMemList` e mostrerò in dettaglio come le memorie chip e fast vengono aggiunte alla memoria libera di sistema.

Risorse

Microprocessor-based system design by Ricardo Gutierrez-Osuna ([slides](http://courses.cs.tamu.edu/rgutier/ceg411_f01/)), in particular [Lesson 9 - Exception processing] (http://courses.cs.tamu.edu/rgutier/ceg411_f01/l9.pdf)

Motorola M68000 Family Programmer's Reference Manual [PDF here](https://www.nxp.com/docs/en/reference-manual/M68000PRM.pdf)

Amiga System Programmers Guide, Abacus ([pdf here](https://archive.org/details/Amiga_System_Programmers_Guide_1988_Abacus))

Note for the English readers

If you are interested in the English version of this article, it can be found on Leonardo's blog at the URL: http://www.thedigitalcatonline.com/blog/2018/06/25/exploring-the-amiga-6/

C portabile e ottimizzato per gli 8-bit - parte 3

Terza parte: Tecniche avanzate per ottimizzare il codice C per 8-bit

di Fabrizio Caruso

Questa è la terza parte di una serie di tre articoli che descrivono tecniche per scrivere codice portabile e ottimizzato in ANSI C per tutti i sistemi 8-bit vintage, cioè computer, console, handheld, calcolatrici scientifiche e microcontrollori dalla fine degli anni 70 fino a metà degli anni 90.

L'articolo completo è disponibile on-line su:
https://github.com/Fabrizio-Caruso/8bitC/blob/master/8bitC.md

Consigliamo la lettura dei primi due articoli in cui abbiamo presentato i vari cross compilatori C, abbiamo dato alcune indicazioni su come scrivere codice C portabile e ottimizzato su tutte le architetture 8 bit.

Programmazione ad oggetti

Contrariamente a quello che si possa credere, la programmazione ad oggetti è possibile in ANSI C e può aiutarci a produrre codice più compatto in alcune situazioni. Esistono interi framework ad oggetti che usano ANSI C (es. Gnome è scritto usando GObject che è uno di questi framework).

Nel caso delle macchine 8-bit con vincoli di memoria molto forti, possiamo comunque implementare classi, polimorfismo ed ereditarietà in maniera molto efficiente.

Una trattazione dettagliata non è possibile in questo articolo e qui ci limitiamo a citare due strumenti fondamentali:

- puntatori a funzioni per ottenere metodi polimorfici, cioè il cui binding (e quindi comportamento) è dinamicamente definito a run-time. Si può evitare l'implementazione di una vtable se ci si limita a classi con un solo metodo polimorfico.
- puntatori a struct e composizione per implementare sotto-classi: dato uno struct A, si implementa una sua sotto-classe con uno struct B definito come uno struct il cui primo campo è A. Usando puntatori a tali struct , il C garantisce che gli offset di B siano gli stessi degli offset di A.

Esempio (preso da https://github.com/Fabrizio-Caruso/CROSS-CHASE/tree/master/src/chase)

Definiamo Item come un sotto-classe di Character a cui aggiungiamo delle variabili ed il metodo polimorfico _effect() :

```
struct CharacterStruct
{
        unsigned char _x;
        unsigned char _y;
        unsigned char _status;
        Image* _imagePtr;
};
typedef struct CharacterStruct Character;
...
struct ItemStruct
{
        Character _character;
        void (*_effect)(void);
        unsigned short _coolDown;
        unsigned char _blink;
};
typedef struct ItemStruct Item;
```

e poi potremo passare un puntatore a Item come se fosse un puntatore a Character (facendo un semplice cast):

```
Item *myItem;
void foo(Character * aCharacter);
...
foo((Character *)myItem);
```

Perché ci guadagniamo in termine di memoria? Perché sarà possibile trattare più oggetti con lo stesso codice e quindi risparmiamo memoria.

Uso avanzato della memoria

In molte architetture alcune aree della memoria RAM sono usate come buffer oppure sono dedicate a usi specifici come alcune modalità grafiche. Il mio consiglio è quindi di studiare le mappa della memoria di ogni sistema per trovare queste preziose aree.

Per esempio per il Vic 20:
http://www.zimmers.net/cbmpics/cbm/vic/memorymap.txt

In particolare consiglio di cercare:

- buffer della cassetta, della tastiera, della stampante, del disco, etc.
- memoria usata dal BASIC
- aree di memoria dedicate a modi grafici (che non si intendono usare)
- aree di memoria libere ma non contigue e che quindi non sarebbero parte del nostro binario

Queste aree di memoria potrebbero essere sfruttate dal nostro codice se nel nostro use-case non servono per il loro scopo originario (esempio: se non intendiamo caricare da cassetta dopo l'avvio del programma, possiamo usare il buffer della cassetta per metterci delle variabili da

usare dopo l'avvio potendolo comunque usare prima dell'avvio per caricare il nostro stesso programma da cassetta).

Esempi utili
In questa tabella diamo alcuni esempi utili per vari sistemi tra cui molti con poca memoria disponibile:

computer	descrizione	area
Commodore 16	tape buffer	$0333-03F2
Commodore 16	input buffer	$0200-0258
Commodore 64 & Vic 20	tape buffer	$033C-03FB
Commodore 64 & Vic 20	input buffer	$0200-0258
Commodore Pet	input buffer	$0200-0250
Commodore Pet	buffer	$033A-03F9
Galaksija	variable a-z	$2A00-2A68
Sinclair Spectrum 16K/48K	printer buffer	$5B00-5BFF
Mattel Aquarius	random number space	$381F-3844
Mattel Aquarius	input buffer	$3860-38A8
Oric	alternate charset	$B800-B7FF
Oric	grabable memory per modo hires	$9800-B3FF
Oric	Page 4	$0400-04FF
Sord M5	RAM for ROM routines (*)	$7000-73FF
TRS-80	RAM for ROM routines (*)	$4000-41FF
VZ200	printer buffer & misc	$7930-79AB
VZ200	BASIC line input buffer	$79E8-7A28

(*): Vari buffer e locazioni ausiliarie usate dalle routine in ROM. Per maggiori dettagli facciamo riferimento rispettivamente a: http://m5.arigato.cz/m5sysvar.html e http://www.trs-80.com/trs80-zaps-internals.htm.

In C standard potremmo solo assegnare le variabili puntatore e gli array su specifiche locazioni di memoria. Di seguito diamo un esempio di mappatura a partire da 0xC000 in cui abbiamo definito uno struct di tipo Character che occupa 5 byte, e abbiamo le seguenti variabili:

- player di tipo Character ,
- ghosts di tipo array di 8 Character (40=$28 byte)
- bombs di tipo array di 4 Character (20=$14 byte)

```
Character *ghosts = 0xC000;
Character *bombs = 0xC000+$28;
Character *player = 0xC000+$28+$14;
```

Questa soluzione generica con puntatori non sempre produce il codice ottimale perché obbliga a fare diverse deferenziazioni e comunque crea delle variabili puntatore (ognuna delle quali dovrebbe occupare 2 byte) che il compilatore potrebbe comunque allocare in memoria.

Non esiste un modo standard per dire al compilatore di mettere qualunque tipo di variabile in una specifica area di memoria. I compilatori di CC65 e Z88DK invece prevedono una sintassi per permetterci di fare questo e guadagnare diverse centinaia o migliaia di byte preziosi. Vari esempi sono presenti in: https://github.com/Fabrizio-Caruso/CROSS-CHASE/tree/master/src/cross_lib/memory

In particolare bisogna creare un file Assembly .s (con CC65) o .asm (con Z88DK) da linkare al nostro eseguibile in cui assegnamo un indirizzo ad ogni nome di variabile a cui aggiungiamo un prefisso underscore.

Sintassi CC65 (esempio Vic 20)

```
.export _ghosts;
_ghosts = $33c
.export _bombs;
_bombs = _ghosts + $28
.export _player;
_player = _bombs + $14
```

Sintassi Z88DK (esempio Galaksija)

```
PUBLIC _ghosts, _bombs, _player
defc _ghosts = 0x2A00
defc _bombs = _ghosts + $28
defc _player = _bombs + $14
```

CMOC mette a disposizione l'opzione --data=<indirizzo> che permette di allocare tutte le variabili globali scrivibili a partire da un indirizzo dato.

La documentazione di ACK non dice nulla a riguardo. Potremo comunque definire i tipi puntatore e gli array nelle zone di memoria libera.

Struttura ottimale del binario

Se il nostro programma prevede dei dati in una definita area di memoria, sarebbe meglio metterli direttamente nel binario che verrà copiato in memoria durante il caricamento. Se questi dati sono invece nel codice, saremo costretti a scrivere del codice che li copia nell'area di memoria in cui sono previsti. Il caso più comune è forse quello degli sprites e dei caratteri/tiles ridefiniti. Questo sarà possibile solo in sistemi in cui i dati stanno nella stessa memoria RAM a cui accede direttamente il processore come per esempio molti sistemi che prevedono video memory mapped.

[CC65] Istruiamo il linker

Ogni compilatore mette a disposizioni strumenti diversi per definire la struttura del binario e quindi permetterci di costruirlo in maniera che i dati siano caricati in una determinata zona di memoria durante il load del programma senza uso di codice aggiuntivo. In particolare su CC65 si può usare il file .cfg di configurazione del linker che descrive la struttura del binario che vogliamo produrre. Il linker di CC65 non è semplicissimo da configurare ed una sua descrizione andrebbe oltre lo scopo di questo articolo. Una descrizione dettagliata è presente su: https://cc65.github.io/doc/ld65.html. Il mio consiglio è di leggere il manuale e di modificare i file di default .cfg già presenti in CC65 al fine di adattarli al proprio use-case.

Exomizer ci aiuta (anche) in questo caso

In alcuni casi se la nostra grafica deve trovarsi in un'area molto lontana dal codice, e vogliamo creare un unico binario, avremo un binario enorme e con un "buco". Questo è il caso per esempio del C64 in cui la grafica per caratteri e sprites può trovarsi lontana dal codice. In questo caso io suggerisco di usare exomizer sul risultato finale: https://bitbucket.org/magli143/exomizer/wiki/Home.

[Z88DK] Appmake fa (quasi) tutto per noi

Z88DK fa molto di più e il suo potente tool appmake costruisce dei binari nel formato richiesto. Z88DK consente comunque all'utente di definire sezioni di memoria e di definire il "packaging" del binario ma non è semplice. Questo argomento è trattato in dettaglio in https://github.com/z88dk/z88dk/issues/860.

Compilazione ottimizzata

Non tratteremo in modo esaustivo le opzioni di compilazione dei cross-compilatori e consigliamo di fare riferimento ai loro rispettivi manuali per dettagli avanzati. Qui daremo una lista delle opzioni per compilare codice ottimizzato su ognuno dei compilatori che stiamo trattando.

Ottimizzazione "aggressiva"

Le seguenti opzioni applicano il massimo delle ottimizzazioni per produrre codice veloce e soprattutto compatto:

Architettura	Compilatore	Opzioni
Intel 8080 ACK -O6	ACK	-o6
Zilog Z80	SCCZ80 (Z88DK)	-o3
Zilog Z80000	ZSDCC (Z88DK)	-compiler=sdcc -SO3 --max-alloc-node20
MOS 6502	CC65	-O -Cl
Motorola 6809	CMOC	-o2

Velocità vs Memoria

In generale su molti target 8-bit il problema maggiore è la presenza di poca memoria per codice e dati. In generale il codice ottimizzato per la velocità sarà sia compatto che veloce ma non sempre le due cose andranno assieme.

In alcuni altri casi l'obiettivo principale può essere la velocità anche a discapito della memoria.

Alcuni compilatori mettono a disposizioni delle opzioni per specificare la propria preferenza tra velocità e memoria:

Architettura	Compilatore	Opzioni	Ottimizzazione
Zilog Z80	ZSDCC (Z88DK)	compiler=sdcc --opt-code-size	Memoria
Zilog Z80	SCCZ80 (Z88DK)	--opt-code-speed	Velocita'
MOS 6502	CC65	-Oi , -Os	Velocita'

Problemi noti

- CC65: -Cl impedisce la ricorsione
- ZSDCC: ha dei bug a prescindere dalle opzioni e ne ha altri presenti con -SO3 in assenza di --max-alloc-node20000.

Per ridurre i tempi di compilazione di ZSDCC (a volte lunghissimi) e i suoi bug, consigliamo di usare SCCZ80 (eventualmente con opzione -O3) durante la fase di sviluppo. ZSDCC andrebbe provato solo alla fine.

Evitare il linking di codice inutile

I compilatori che trattiamo non sempre saranno capaci di eliminare il codice non usato. Dobbiamo quindi evitare di includere codice non utile per essere sicuri che non finisca nel binario prodotto. Possiamo fare ancora meglio con alcuni dei nostri compilatori, istruendoli a non includere alcune librerie standard o persino alcune loro parti se siamo sicuri di non doverle usare.

Evitare la standard lib

Evitare nel proprio codice la libraria standard nei casi in cui avrebbe senso, può ridurre la taglia del codice in maniera considerevole. Questa regola è generale ma è particolarmente valida quando si usa ACK per produrre un binario per CP/M-80. In questo caso consiglio di usare esclusivamente getchar() e putchar(c) quando è possibile.

[z88dk] Pragmas per non generare codice

Z88DK mette a disposizione una serie di pragma per istruire il compilatore a non generare codice inutile.

Per esempio:

#pragma printf = "%c %u"
includerà solo i convertitori per %c e %u escludendo tutto il codice per gli altri;

#pragma-define:CRT_INITIALIZE_BSS=0
non genera codice per l'inizializzazione dell'area di memoria BSS;

#pragma output CRT_ON_EXIT = 0x10001
il programma non fa nulla alla sua uscita (non gestisce il ritorno al BASIC);

#pragma output CLIB_MALLOC_HEAP_SIZE = 0
elimina lo heap della memoria dinamica (nessuna malloc possibile);

#pragma output CLIB_STDIO_HEAP_SIZE = 0
elimina lo heap di stdio (non gestisce l'apertura di file).

Alcuni esempi sono in https://github.com/Fabrizio-Caruso/CROSS-CHASE/blob/master/src/cross_lib/cfg/z88dk.

Usare le routine presenti in ROM

La stragrande maggioranza dei sistemi 8-bit (quasi tutti i computer) prevede molte utili routine nelle ROM. E' quindi importante conoscerle. Per usarle esplicitamente dovremo scrivere del codice Assembly da richiamare da C. Il modo d'uso dell'Assembly assieme al C può avvenire in modo in line (codice Assembly integrato all'interno di funzioni C) oppure con file separati da linkare al C ed è diverso in ogni dev-kit. Per i dettagli consigliamo di leggere i manuali dei vari dev-kit.

Questo è molto importante per i sistemi che non sono (ancora) supportati dai compilatori e per i quali bisogna scrivere da zero tutte le routine per l'input/output.

Esempio (preso da https://github.com/Fabrizio-Caruso/CROSS-CHASE/blob/master/src/cross_lib/display/display_macros.c)

Per il display di caratteri sullo schermo per i Thomson M05, M06 e Olivetti Prodest PC128 (sistemi non supportati da nessun compilatore) piuttosto che scrivere una routine da zero possiamo affidarci ad una routine Assembly presente nella ROM:

```
void PUTCH(unsigned char ch)
{
  asm
  {
     ldb ch
     swi
     .byte 2
  }
}
```

Le librerie spesso lo già fanno per noi

Fortunatamente spesso potremo usare le routine della ROM implicitamente senza fare alcuna fatica perché le librerie di supporto ai target dei nostri dev-kit lo fanno già per noi. Usare una routine della ROM ci fa risparmiare codice ma può imporci dei vincoli perché per esempio potrebbero non fare esattamente quello che vogliamo oppure usano alcune aree della RAM (buffer) che noi potremmo volere usare in modo diverso.

BASCK: scoviamo le librerie della ROM

Se non riusciamo a trovare informazioni sulle routine della ROM come per esempio le loro entry points perché, per esempio stiamo sviluppando per un sistema poco documentato, possiamo usare il tool BASCK (https://github.com/z88dk/z88dk/blob/master/support/basck/basck.c, sviluppato da Stefano Bodrato) che viene distribuito con Z88DK. BASCK prende in input i file delle ROM di sistemi basati su Z80 e 6502 e cercando vari pattern, trova le routine e i loro indirizzi. Usare queste routine non è sempre facile ma in alcuni casi è banale.

Esempio:

1. Dobbiamo lanciare BASCK passandogli la ROM e leggere il suo output. Per esempio se cerchiamo la routine PRINT nella ROM, filtriamo nell'output la stringa "PRS" (in Unix useremo il comando "grep")

```
> basck -map romfile.rom |grep PRS
PRS = $AAAA ; Create string entry and print it
```

Otterremo in questo modo l'indirizzo della routine del BASIC per stampare dei caratteri.

2. A questo punto potremo scrivere del codice in Assembly o C per usarla:

```
extern void rom_prs(char * str) __z88dk_fastcall @0xAAAA;
main() {
rom_prs ("Hello WORLD !");
while (1){};
}
```

Sfruttare i chip grafici

Come visto nelle sezioni precedenti, anche se programmiamo in C non dobbiamo dimenticare l'hardware specifico per il quale stiamo scrivendo del codice. Conoscere l'hardware può aiutarci a scrivere codice molto più compatto e/o più veloce. In particolare la conoscenza del chip grafico può aiutarci a risparmiare tanta ram.

Esempio (Chip TI VDP come il TMS9918A presente su MSX, Spectravideo, Memotech MTX, Sord M5, etc.).
I sistemi basati su questo chip prevedono una modalità video testuale (Mode 1) in cui il colore del carattere è implicitamente dato dal codice del carattere. Se usiamo questo speciale modo video, sarà quindi sufficiente un singolo byte per definire il carattere ed il suo colore con un notevole risparmio in termini di memoria. I computer Atari 8-bit prevedono un modo grafico testuale analogo (graphics mode 1+16, Antic mode 6).

Esempio (Chip VIC del Commodore Vic 20)
Il Commodore Vic 20 è un caso veramente speciale perché prevede dei limiti hardware considerevoli (RAM totale: 5k, RAM disponibile per il codice: 3,5K) ma anche dei trucchi per superarli almeno in parte.

La caratteristica più sorprendente è che il chip grafico VIC può mappare una parte dei caratteri in RAM lasciandone metà definiti dalla ROM. Se bastano n (<=64) caratteri ridefiniti possiamo mapparne in RAM solo 64 con POKE(0x9005,0xFF);. Ne potremo usare anche meno di 64 lasciando il resto per il codice ma mantenendo in aggiunta 64 caratteri standard.

Inoltre è possibile in alcuni casi fare uso della memoria video dedicata a cui accedono alcuni chip grafici (come il TI VDP, MOS VDC del C128, etc.) per altri scopi. Il costo computazionale sarebbe comunque notevole perché l'accesso su questa memoria sarebbe indiretto.

Con questo articolo termina la serie di 3 articoli redatta da **Fabrizio Caruso** che RetroMagazine ha avuto l'onore di ospitare tra le sue pagine.

Ogni altra fatica editoriale che Fabrizio vorra' proporci, trovera' spazio tra le nostre pagine visto l'alto contenuto didattico e l'interesse suscitato dai suoi scritti.

Il team di RetroMagazine

RetroGiochiAmo: Jeep Command

di Daniele Brahimi

Dopo aver trascorso ben due mesi in mezzo a rovine azteche rischiando la vita ad ogni passo, godiamoci il meritato momento di relax su una bella Jeep.

Non parlo degli odierni Suv da figli di papà che infestano le nostre città ormai da diversi anni, bensì di una modesta Jeep militare in grado di sparare, effettuare salti ed evitare bombe.

Dai mitici avventurieri dei numeri scorsi ho voluto passare ad uno dei primi sparatutto su Commodore 64 ed il gioco di cui voglio parlarvi questo mese è uno dei primi giochi che ho fatto girare sul mio beneamato Commodore 64: Jeep Command!

Come vi ho anticipato poco fa, avremo il controllo di una Jeep il cui compito è quello di saltare buche, sparare ed evitare bombe. Tutto ciò potrebbe sembrare semplice ad un primo impatto ma vi assicuro che sarà molto più insidioso di quanto si possa pensare.

Inizialmente il percorso sarà scorrevole, pochi nemici e pochi ostacoli, ma andando avanti giustamente aumenterà la difficoltà ed il terreno diventerà anche più interessante dal punto di vista grafico e di gameplay.

Con un po' di allenamento e prendendoci la mano sono sicuro che riuscirete ad arrivare fino in fondo al gioco anche perche' più proseguirete e più avrete la possibilità di vincere vite extra... Già quelle a disposizione sin dall'inizio non sono poche. Il segreto sta quasi tutto nel saper padroneggiare la velocità del mezzo, regolabile.

La cosa che mi ha motivato a giocare e rigiocare questo gioco con grande entusiasmo

ed ovviamente riportarlo sul numero di questo mese è stata la musica! Molto bella e rilassante oltre che motivante... E poi diciamoci la verità: Quanti di noi hanno comprato ed amato i giochi grazie soprattutto alla musica presente? Essa sarà stata uno dei punti di forza dei programmatori per invogliare il giocatore e rapirlo, ma essa meriterebbe non un articolo, bensì un capitolo a parte.

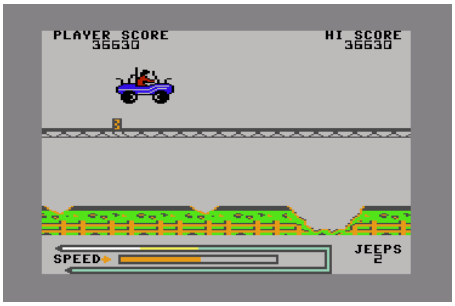
Altro punto di forza di questo gioco è il caricamento che non vi ruberà molto tempo come invece fecero molti altri giochi (con tutto rispetto).

Il titolo è stato presente anche in molte cassette da edicola e non si dovrebbe fare fatica a trovarlo, così ve lo potrete gustare!

La durata media del gioco? Preferisco non rovinarvi la sorpresa visto che tra poco è pasqua, se volete saperlo giocatelo, amatelo e gustatevelo!

Il gioco ha avuto anche un seguito parecchio simile al primo e presto o tardi arriveremo anche da lui, questo è poco ma sicuro.

Con questo articolo voglio augurare una felice Pasqua a voi tutti retrogamer che seguite la rivista sin dal primo numero mettendoci evidentemente passione (si sente eccome) ed anche a tutto lo staff che spero avrò modo di conoscere di persona.



Ogni mese sono felicissimo di riportare alla luce un gioco dimenticato e spesso da molti sottovalutato all'epoca dei fatti.

Ci leggiamo al prossimo articolo!

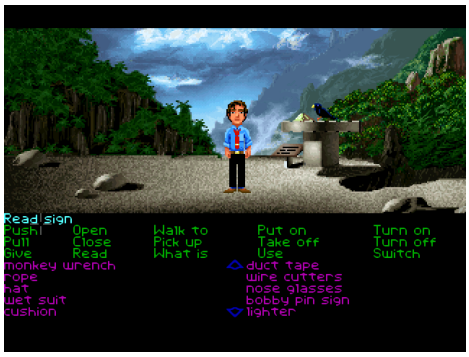
Zak McKracken and the Alien Mindbenders

di The Ancient One

"Solo tu puoi fermare la stupidemia"

Zak McKracken è una pietra miliare nelle avventure grafiche ed è un gioco sotto molti aspetti tutt'oggi senza eguali.

Zak McKracken, insieme a Maniac Mansion, rivoluzionarono il classico game play delle avventure grafiche, riducendo al minimo le possibilità di morire e introducendo una rivoluzionaria interfaccia utente (il celebre SCUMM - Script Creation Utility for Maniac Mansion - di Ron Gilbert) che permette al giocatore di concentrarsi sul gioco anziché sulla ricerca di sinonimi compresi dal gioco. Per capire lo spessore di questa rivoluzione, basti pensare che fino ad allora le avventure grafiche erano mosse da parser identici a quelli delle avventure testuali. Esempi lampanti sono i primi capitoli delle grandi saghe della Sierra (King's Quest in primis)! Uscito dopo Maniac Mansion, Zak ne sfrutta il motore grafico e l'interfaccia, ma non per questo è privo di una sua forte e personalissima carica innovativa, seppur meno appariscente di quella del predecessore.



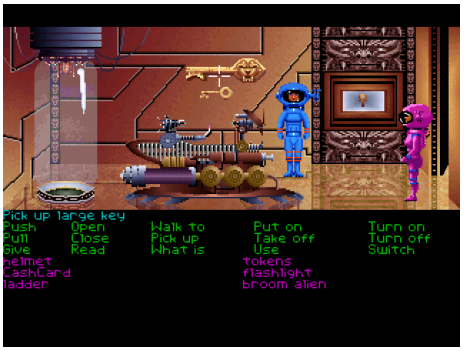
Il motore grafico e l'interfaccia è identica a quella di Maniac Mansion. Se oggi può apparire poco user-friendly, bisogna dire che ci si abitua rapidamente.

Le caratteristiche "notevoli" per l'epoca di questo motore grafico sono, fra le tante:

- la gestione di più personaggi protagonisti (che in Zak, a differenza di Maniac Mansion, sono predeterminati e non entrano in gioco dall'inizio).
- la possibilità di avere lo scorrimento dello sfondo,
- locazioni buie o illuminate,
- modifica di particolari oggetti dello sfondo in base alle azioni del giocatore (ad es. una finestra che si apre e si chiude).

Tecnicamente il gioco è ormai superato, anche se l'originale stile "fumettoso" della grafica non lo rende affatto sgradevole agli occhi, grazie anche alla chiarezza e semplicità del tratto.

Gli effetti sonori e le musiche sono limitati al minimo, anche se lo "Zak McKracken Theme" (informazioni sul midi) può suscitare tutt'oggi grandi emozioni a qualsiasi oldgamer!



Da un punto di vista tecnico, è interessante notare che Zak adotta un codice, poi abbandonato dagli altri giochi della Lucasfilm, che permetteva di modificare lo sprite del protagonista, consentendo così al giocatore di far indossare a Zak e agli altri personaggi qualsiasi indumento egli desiderasse. Questo, oltre che creare una nuova tipologia di enigmi (ad es. indossare il cappello e gli occhiali col nasone per non farsi riconoscere), è estremamente divertente e, senza dubbio, innovativo.

E' opportuno precisare che anche altri giochi della LucasFilm (e non solo, si pensi a Hand of Fate della Westwood) vedevano il personaggio indossare abiti diversi nel corso del gioco, ma permettevano di farlo solo in determinati momenti della storia, senza che il giocatore avesse su questo alcun controllo.

Un'ultima innovazione interessante apportata all'engine di Maniac Mansion è il ridimensionamento fuori schermo (precursore del ridimensionamento "scalare" introdotto dalla Lucas con Loom). Scelta tecnica che ben si adatta al "sense of wonder" che pervade l'intero gioco. Celebre questa immagine del tempio di Marte:

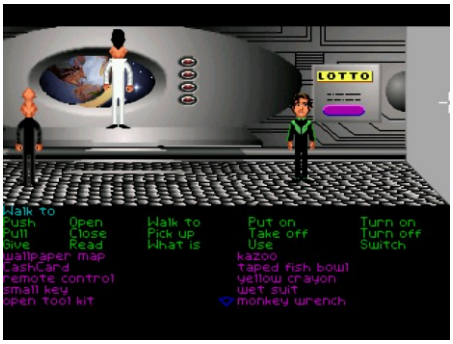
Come accennato, i quattro protagonisti del gioco sono predeterminati e in questo modo è precluso a Zak uno dei più grandi pregi di Maniac Mansion: la possibilità di avanzare e concludere l'avventura in modi diversi a seconda dei personaggi scelti all'inizio del gioco.

Questo sembrerebbe togliere dei punti all'interattività di Zak, ma in realtà questa mancanza è abbondantemente compensata da una diversa impostazione del gioco. Innanzitutto Zak si contraddistingue da Maniac Mansion:

- La vastissima (forse a tutt'oggi non uguagliata) libertà d'azione lasciata al giocatore, che fin dall'inizio del gioco ha accesso alla maggior parte delle numerosissime locazioni.
- Per il maggior numero di locazioni e per la varietà delle stesse (Zak si troverà a esplorare tutti i luoghi mitici del nostro pianeta e andrà persino su Marte, finendo per

svelare tutti i grandi misteri della terra!).

- Per un diverso approccio da tenere nel risolvere gli enigmi. Infatti uno degli aspetti principali di Zak è che tutti gli oggetti devono essere usati in base alle loro caratteristiche fisiche e così, per risolvere un determinato enigma, posso essere usati più oggetti realisticamente adatti allo scopo (ad es. per svegliare il conducente dell'autobus all'inizio del gioco è possibile utilizzare una varietà di oggetti: o per fare rumore (il kazoo o la chitarra), o per battere sul finestrino (tutti gli oggetti "duri", come il pane secco o il coltello).
- Per la simpatica possibilità di variare l'interfaccia utente, per cui in certe occasioni (ad es. quando prendete il controllo dello scoiattolo) cambiano i verbi che è possibile utilizzare. Purtroppo però questa possibilità viene sfruttata solo raramente...



Rimane invece, come in Maniac Mansion, il concetto per cui certe azioni possono essere compiute solo da certi personaggi. Mantenendo così tutta un'interessante tipologia di enigmi.

Al game play di Maniac Mansion è stato poi aggiunto un altro elemento che ho trovato molto interessante: i labirinti. Molto spesso infatti certe locazioni chiave sono precedute da dei lunghi labirinti (es.. i corridoi di un tempio o una jungla). Quasi tutti sono molto difficili da completare se non si capisce la "sequenza" che ci permette di districarci all'interno.

Per finire, Zak ha una serie di pregi che non è facile descrivere a parole:

- Il "sense of wonder" dinanzi ai grandi misteri della terra (dalle piramidi, fino al faccione di marte)
- Il sottile senso dell'umorismo, reso praticamente senza scene animate o dialoghi, ma affidato all'atmosfera. Indimenticabile il sorriso di Zak...
- La bellissima sensazione di "aver viaggiato" in lungo ed in largo che si prova dopo aver completato l'avventura.

Per tutte queste caratteristiche Zak McKracken ha reinventato (insieme a Maniac Mansion) il genere della avventure grafiche ed è ancora oggi un'avventura validissima, anche se il giocatore dovrà mettere in conto l'interfaccia non ancora perfetta del gioco e la grafica ridotta al minimo. Nonché l'assenza di dialoghi.

Non ha il carisma ed il fascino di Monkey Island, ma - strettamente parlando - come avventura grafica (libertà d'azione e struttura degli enigmi) gli è probabilmente superiore.

SOPHIA

GIUDIZIO SUL GIOCO

GIOCABILITA'

90%

Immediato e coinvolgente come pochi titoli sullo spectrum. Alcuni passaggi potrebbero apparire particolarmente frustranti all'inizio ma una volta imparato come superarli ci accorgeremo che l'unico avversario del gioco è la nostra lentezza di riflessi. Una volta imparato a giocare, Sophia premia sempre chi si intrattiene con lei.

LONGEVITA'

90%

Per trionfare dovrete memorizzarvi tutte le posizioni degli oggetti e dei mostri per poi abituarvi a passare rapidamente da un arma all'altra dato che le creature cambiano di tipologia e i teschi non sono sempre nello stesso posto. Difficile come un gioco dell'epoca, coinvolgente come un titolo che ha fatto tesoro di decenni d'esperienza.

Sophia

Alessandro Grussu – Anno 2017 – Piattaforma: Zx Spectrum

La Storia

"L'Impero è una vasta entità politica che si estende su numerose terre, popolate da diverse genti. Per anni pace e giustizia sono state assicurate dal dominio dell'attuale Imperatore. Ma adesso, l'Impero è sotto una grave minaccia: Yojar, uno stregone malvagio arrivato da un'altra dimensione, lo ha invaso. Non solo ha occupato tre regioni principali con i suoi scherani, ma vi ha anche disseminato dei simboli di potere a forma di teschi giganti, che fanno da diffusori dell'influenza del loro creatore; se rimangono sul posto per un tempo abbastanza lungo, permetteranno a Yojar di entrare nella nostra dimensione, e allora sarà impossibile fermarlo..." così recita la trama principale e potete immaginare come si evolverà. L'impero, abbastanza incapace da non poter nulla contro il potere di Yojar ma abbastanza umile da sapere a chi rivolgersi, chiede aiuto alle maghe, uniche detentrici del potere magico. Una di loro si fa avanti volontaria ed inizia così la missione di Sophia per sconfiggere il malvagio stregone.

Il gioco

Sophia è stato rilasciato gratuitamente da Alessandro Grussu sul suo sito internet (http://www.alessandrogrussu.it/Sophia.html) ed è giocabile su Zx Spectrum 128k. Si presenta come un platform in 2D a schermate fisse in cui impersoneremo la giovane maga intenta a raccogliere i teschi giganti (simboli di potere) disseminati nei vari regni. Ogni teschio raccolto andrà poi portato nell'apposito calderone della strega per essere distrutto ed una volta raccolti tutti e quattro i teschi presenti nel livello si potrà passare a quello successivo. Quattro stage in tutto, con l'ultimo che si risolve nel confronto col boss principale del gioco.

Arrivare al cospetto di Yojar sarà però un impresa di quelle davvero toste. I livelli sono disseminati di trappole e di nemici che si rigenerano a più riprese. Oltre ai teschi del potere, Sophia dovrà anche trovare le tre chiavi, anche loro sparse lungo i livelli, capaci di farla accedere a zone altrimenti chiuse da appositi cancelli, il tutto stando attenta alle munizioni consumate.

La nostra potrà utilizzare due tipi di magia: fulmine e sfera di energia, entrambe possedute in numero limitato e singolarmente più efficaci su mostri di un certo tipo piuttosto che di un altro (a noi scoprire quale arma si rivelerà più efficace su ogni specifico mostro). Le vite a nostra disposizione sono cinque e dulcis in fundo avremo la possibilità, una sola volta per livello, di utilizzare un incantesimo difensivo capace di renderci invulnerabili per alcuni secondi. Tutta questa carenza di risorse può però venir compensata facendo acquisti. In ogni stage è infatti presente una dimora di strega retta su zampe di gallina (Baba Yaga docet) in cui entrare a fare acquisti. Ogni mostro ucciso è una moneta e con esse si possono ricaricare le magie consumate (compresa quella difensiva) e addirittura acquistare una vita extra...ma solo una per livello.

I controlli e le opzioni

Sophia si controlla nativamente da tastiera: Q per volare, A per cambiare magia offensiva, O per andare a sinistra, P per andare a destra, M per utilizzare una magia d'attacco ed infine H per la magia difensiva. Tutti questi tasti possono però venir cambiati nel menu opzioni presente nella schermata principale del gioco, così come scegliere di giocare con il joystick (Kempston o Sinclair) e soprattutto accedere con una password all'ultimo livello raggiunto. Già, perché alla fine di ogni stage completato con successo ci verrà rilasciato un codice di colori con cui potremo passare direttamente al livello successivo senza dover ogni volta ricominciare dall'inizio. Aspettate a dire che è un vantaggio da principianti, il gioco è davvero tosto e benedirete questa possibilità.

Conclusioni

Sophia è difficile ma non proibitivo. Entusiasmante sin da subito e capace di rapirti partita dopo partita fino a che non lo si avrà completato. Per finirlo ci vuole concentrazione e moltissima esperienza (altrimenti detta: si muore più che in dark souls). A livello tecnico è fantastico e lo Speccy viene spremuto per benino sia sul fronte grafico che su quello sonoro, grazie alle sue musiche sempre gradevoli e concitate. La scena di giochi per ZX Spectrum è più viva che mai anche oggi e Alessandro Grussu si dimostra con grande competenza un eccellenza in materia.

Curiosità

Il gioco è stato tradotto in 8 lingue, tra cui ovviamente l'italiano.Ufficialmente venne rilasciato per Halloween.

di Starfox Mulder

TI-SCRAMBLE

GIUDIZIO SUL GIOCO

GIOCABILITA'

95%

E' il gioco del bar dove io in gioventù ci spesi un capitale compreso di interessi , solo che ora finalmente è gratis ☺.

LONGEVITA'

Eterno

Per trionfare dovrete memorizzarvi tutte le posizioni E' il primo inimitabile sparatutto orizzontale, che altro dire?

TI-Scramble

Rasmus MoustGaard – Anno 2013 – Piattaforma: TI99/4A

La Konami nel 1981 sviluppò un game arcade dedicato esclusivamente per le sale giochi, dal titolo Scramble. Tale nome sembrerebbe che non fu scelto casualmente in quanto in ambito militare lo scramble (o scrambling) è un termine che definisce l'atto di far decollare un aereo per intercettare e identificare un veicolo sconosciuto, e il gioco infatti rappresenta una astronave che deve evitare/distruggere una serie di nemici sia aerei che terrestri. Tale gioco è stato una pietra miliare nel campo dei videogiochi in quanto fu il primo sparatutto a scorrimento orizzontatale con reale scorrimento grafico, multipli livelli distinti e l'innovativo rifornimento di carburante! Niente più astronavi a propulsione atomica si usa ora la normale benzina ☺.

Il gioco ebbe un successo straordinario e dopo Berzerk fu come numero di cabinati venduti il secondo nella storia del mercato statunitense. Venne poi convertito per Vectrex e Tomy Tutor e dopo molti anni comparve anche per console più moderne vedi game boy nel 2002 e notizia odierna nelle raccolte Konami Arcade Classics e Konami Arcade Collection in arrivo su PlayStation 4, Xbox One, PC e Nintendo Switch.

Vi furono negli anni 80 un mare di cloni non autorizzati che usavano lo stesso nome o molto similare (SKRAMBLE) per quasi tutti i computer 8 bit dell'epoca che andavano dal Vic20/C64 al MSX/ZX Spectrum ecc... con risultanti non sempre fedeli all'originale. Fra i computer che all'epoca non ebbero nessuna software house che si cimentò nella conversione di questo gioco famoso furono il TI99/4A l'Atari e l'Amstrad CPC.

Questa lacuna è stata colmata grazie alle produzioni homebrew create negli ultimi anni dove abbiamo dei porting fedeli all'originale.

Ecco l'ordine temporale dei porting effettuato su questi sistemi:

Sistema	Anno
Atari 7800	2012
TI99/4A	2013
Atari 2600	2015/16
Atari 800	2018
Amstrad CPC & GX 4000	2019

TI Scramble che è il porting di Scramble sul computer TI99 merita una piccola menzione ad uso degli utenti del TI... Tale porting oltre che estremamente fedele all'originale è particolare in quanto è stato creato in pochissimo tempo da **Rasmus MoustGaard,** un programmatore Danese che fino all'anno prima era completamente all'oscuro dei computer creati dalla Texas Instruments e del TI99/4A!

Lui si presentò come un fulmine a ciel sereno al pubblico degli utenti Texas, presentando in un anno tre giochi che erano: Titanium derivato come concetto da Uridium, TI Scramble derivato da Scramble e qui presentato e Road Hunter che fu creato prendendo come spunto Spy Hunter.

Tutti questi giochi oltre ad essere stati programmati interamente in assembler, sfruttano al massimo il computer ed hanno creato de facto un nuovo standard delle tecniche di programmazione sul TI in quanto ogni gioco ha superato dei limiti di programmazione che si credevano invalicabili. Ad oggi Rasmus è uno dei più prolifici programmatori di videogiochi sul TI99 portando su questa macchina dei giochi che sono delle vere e proprie perle e saranno oggetto di trattazione sui prossimi numeri.

di Ermanno Betori

PARSEC

GIUDIZIO SUL GIOCO

GIOCABILITA'

95%

Gioco che ha una perfetta rispondenza delle coincidenze nelle collisioni, movimenti fluidi sia della navicella spaziale che dei nemici, una giocabilità ottima che si esprime però non con i Joystick originali Texas che erano di scarsa qualità, ma tramite adattatore con quelli creati per gli altri 8 bit vedi ad esempio il famoso Albatross.

LONGEVITA'

SV

Dopo aver superato come record personale il milione di punti, dopo oltre 30 anni di possesso una partita ancora ci scappa. A mia conoscenza i maestri, nonché devoti di questo gioco, sono i fratelli Matthew e Jason Doucette, proprietari della software house Xona Games, che sono riusciti ad arrivare oltre il milione e mezzo di punti (livello 31) usando l'emulatore classic99 finché Windows non gli andò in crash con conseguente reset del gioco.

Parsec

Jim Dramis e Paul Urbanus – Anno 1982 – Piattaforma: TI99/4A

Questo Shoot' em up spaziale creato per il TI99/4A nacque nel 1982 dalle fertili menti di due sviluppatori di videogames Jim Dramis e Paul Urbanus (Jim Dramis programmò sempre per il TI99 anche Car Wars suo primo videogioco e Munchman).

Doveva essere nelle intenzioni la loro risposta al videogioco Scramble, infatti vi ritroviamo molti elementi in comune quali la necessità del rifornimento del carburante, multi-livelli distinti, scrolling orizzontale dello sfondo, ma con un approccio e implementazioni diverse. Come sparo venne sostituito il singolo pixel che rappresentava il proiettile (vedi Scramble) con una linea che simulava concettualmente il colpo di energia di un laser, inoltre venne ideato il concetto di surriscaldamento della navicella che avveniva se si usava troppo il laser con conseguente autodistruzione, pertanto niente sparo fisso a ripetizione, mentre i nemici ovviamente non hanno tale limitazione ;-).

Altra diversificazione rispetto a Scramble che aveva tra i tanti nemici delle meteore e missili, è la presenza di navicelle nemiche kamikaze, una astronave che diventa invisibile, un satellite killer, un tunnel per il rifornimento e la presenza di un campo di asteroidi (Guerre Stellari, l'Impero Colpisce Ancora, Star Trek "Astronave Klingon" vi dice qualcosa :-) ??)

Come qualità fu all'epoca il top, infatti divenne la killer-application del TI99/4A come miglior gioco e ci sono voluti alcuni decenni per avere altri videogiochi che lo superassero sia come concept che come tecnica di programmazione. Questo videogioco fu il primo ad usare il modo grafico BitMap, ad avere tramite il modulo Speech Synthesizer una voce femminile che ti avvisava l'arrivo dei nemici o degli asteroidi, il superamento del livello, il rifornimento ecc.., e una idea di pseudo-sfondo parallasse che si concretizzerà dopo oltre 30 anni nel demo Light-Year creato dal programmatore Rasmus Mousgaard con la collaborazione per la musica di Mike Brent (alias Tursi) e RushJET. Inoltre il gioco fu venduto su cartuccia e riusciva a funzionare perfettamente sulla sola consolle del TI99 che aveva a disposizione oltre alla VRAM del processore video, solo 256 byte di RAM dedicata alla CPU!!

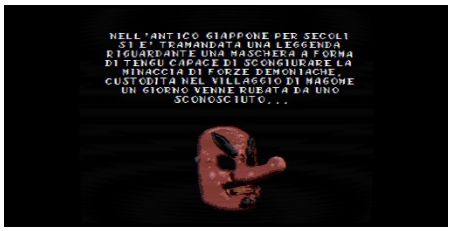
Nel gioco i programmatori misero molti Easter-Egg; sullo sfondo che scrolla oltre alla scritta Parsec compaiono le lettere Jed e Urb che sono riferite ai loro nomi e cognomi, come anche i nomi delle navicelle nemiche Urbite e Dramite, mentre il nome della navicella invisibile Bynite (la più infame...) prende il nome da Don Bynum capo della divisione sviluppo del TI99/4A (chissà come mai... LOL).

di Ermanno Betori

Curiosita'

Qui sopra vediamo uno snapshot del demo Light–Year, creato con palese riferimento a Parsec.

KATANA SOUL



GIUDIZIO SUL GIOCO

GIOCABILITA'

80%

Il gameplay di Katana Soul è uno dei punti forti di questo titolo firmato da Seep. Il gioco è immediato ma offre una sfida davvero ardua soprattutto in modalità C64 che, di fatto, raddoppia la difficoltà con nemici più resistenti e potenti in grado di togliere molta energia ad ogni singolo attacco. Il titolo è molto grazioso e rievoca perfettamente lo stile Commodore 64 grazie ad una palette di colori azzeccata ed a un level design efficace. Buonissimo il sonoro che accompagna l'azione.

LONGEVITA'

75%

Il gioco non è esattamente lunghissimo. I più bravi possono terminarlo in mezz'ora ma Katana Soul difficilmente perdona e quindi per arrivare a questa eccellenza bisogna fare più tentativi. Terminato il gioco ci si può cimentare nella modalità Survival che punta ad aumentare il senso di rigiocabilità.

Katana Soul

Seep – Anno 2018 – Piattaforma: PC

È già strano, ma al tempo stesso divertente, scrivere di Commodore 64 nel 2019. E parlare di un gioco che è stato realizzato per Pc ma che ripercorre abbastanza fedelmente lo stile del famoso biscottone ad 8 bit è comunque stimolante.

Katana Soul, uscito lo scorso 25 marzo su Steam per Pc, è un action adventure realizzato da Seep (una piccolissima software house italiana indipendente formata dai fratelli Sergio ed Enrico Giansoldati avvezza a questo genere di titoli), che rievoca i classici sul C64 di fine anni '80 ed inizio anni '90. Lo fa con una grande verve grazie, lo diciamo subito, ad un lavoro serio e meticoloso per la "ricostruzione" di un titolo simil C64. Katana Soul è, infatti, un gioco molto semplice che elimina tutto il superfluo che il buon Commodore 64 non avrebbe – all'epoca – potuto offrire visti non solo gli evidenti limiti hardware della macchina ma anche dei tool di sviluppo. Ora le cose vanno diversamente perché si assistono a lavori incredibili ma questa è un'altra storia.

Giusto ricordare che il team sia stato coadiuvato da Andrea Baroni, autore delle musiche, e da Oscar Celestini che ha realizzato l'ottimo artwork di presentazione che fa da "copertina" virtuale al gioco. I fratelli Giansoldati si sono concentrati sul gameplay e su una buonissima resa grafica che omaggia al meglio le grandi produzioni del passato in grado di scrivere storia degli home computer.

Giocando a Katana Soul ci si immerge in atmosfere passate, e per molti versi irripetibili, grazie ad una sfida "nuda e cruda" che anche a difficoltà normale mette a dura prova gli utenti perché pretende una grande attenzione. Si deve "semplicemente" andare avanti lungo diverse ambientazioni popolate da creature ostili e sempre più forti. Bene: non sarà la classica passeggiata di salute ma un vero e proprio calvario soprattutto per chi è abituato agli standard odierni che prevedono aiuti costanti. Qui è tutto diverso. Non ci sono salvataggi: quando si muore per l'ultima volta (si hanno 4 vite) la partita finisce e si deve ricominciare dall'inizio con un nuovo tentativo. Gli stessi sviluppatori hanno descritto la loro opera come "un gioco breve ma dalla grande sfida". Parole confortate da un gameplay che rievoca fedelmente, soprattutto in modalità C64, quelle che erano le effettive difficoltà dei titoli dell'epoca. O forse è meglio dire di un'era videoludica che ha forgiato tanti giocatori. Il nostro samurai, armato soltanto di katana e guidato dai nostri riflessi, dovrà farsi strada in ambientazioni tipicamente orientaleggianti. Il nostro eroe dovrà affrontare spiriti, demoni e creature tratti dal folklore tradizionale giapponese dell'era Sengoku. Si dovranno scoprire tutte le storie dietro la potenza della maschera tengu e togliere la maledizione demoniaca.

Dal punto di vista tecnico, il titolo si fa apprezzare per una buonissima rievocazione di quello che il C64 poteva offrire. Qualche immagine di presentazione (e finale) molto ben realizzata, una grafica adeguata con personaggi animati in modo congruo a ella potenzialità del mezzo che non disdegna, comunque, qualche trovata ad effetto come gli effetti meteo ed una buona varietà nelle ambientazioni, soprattutto quelle aperte che si alternano molto bene a quelle interne. Interessanti anche alcuni nemici su una rosa di oltre 30 tipologie di mostri diversi.

Ogni tanto avremo l'opportunità di trovare alcuni bonus come quelli che ricaricano la barra d'energia o che ci offrono temporanea immunità permettendo anche di lanciare sfere di energia molto utili ed a tratti quasi indispensabili benché sia possibile andare avanti anche con il solo ausilio della nostra fedele katana. Troveremo anche la fiamma vitale che aggiungerà una vita. Quello che colpisce è comunque l'alto grado di sfida. Superare il prologo per chi non è allenato sarà già un successo ma andando più avanti le difficoltà si moltiplicheranno. Non contenti, gli sviluppatori hanno pure aggiunto la difficoltà C64 e la modalità Survival. Quest'ultima si sbloccherà dopo aver terminato il gioco la prima volta. Opportuno, infine, segnalare l'ottimo rapporto qualità/prezzo: Katana Soul costa 2,39 euro. Li vale oggettivamente tutti.

di Edoardo Ullo

DOUBLE DRAGON

Double Dragon

Taito - Anno 1987 - Piattaforma Arcade

Inizia tutto così...

La mitica prima scena del gioco in cui i cattivi rapiscono la bella Marion

Le ultime fatiche

Dopo aver affrontato tutti i nostri nemici eccoci al cospetto del boss finale

Un seguito degno di nota

Nel 1992 viene rilasciato per lo snes The return of Double Dragon che merita di essere giocato

GIUDIZIO SUL GIOCO

GIOCABILITA'

90%

LONGEVITA'

100%

Nell'epoca d'oro delle sale giochi, ti accorgevi dell'arrivo di un grande titolo dalla ressa che si faceva attorno al suo cabinato, e questo è quello che è successo anche la prima volta che ho visto *Double Dragon*. Mi ricordo infatti la fila di ragazzi incuriositi da quel gioco che presentava tantissime novità sotto ogni punto di vista e che sarebbe diventato poi una pietra miliare del suo genere. Ma andiamo con ordine.

Sviluppato dalla Technos Japan per la Taito, *Double Dragon* fa la sua comparsa nel 1987 proponendosi come primo titolo beat'em up, genere che avrà in futuro un grandissimo successo. Già dal titolo gli sviluppatori volevano far capire l'importanza della cooperazione e infatti i protagonisti sono Billy e Jimmy Lee, due gemelli che devono attraversare tutta la città per liberare la bella Marion rapita da una banda di criminali.

I due protagonisti sono identici sia negli sprites (tranne il colore della divisa) e sia nelle mosse e combo possibili.

Durante il loro cammino si imbattono nei vai membri della banda rivale, ognuno con proprie caratteristiche sia di combattimento che fisiche. Ed è proprio questa una delle principali novità di *Double Dragon* che colpisce immediatamente, poiché non dobbiamo aspettare la fine del livello per imbatterci nel classico boss, ma per strada incontriamo continuamente nemici molto più grandi e forzuti di noi.

Seconda scelta vincente di questo titolo è la possibilità di combattere sia a mani nude e sia raccogliendo armi lasciate per strada dai nostri avversari. Ci possiamo così armare di fruste, mazze da baseball, coltelli e persino barili o sassi da lanciare contro i nostri nemici. Ma la vera caratteristica entusiasmante di *Double Dragon* è rappresentata dalle tecniche di combattimento che si possono attuare. Innanzitutto abbiamo ben tre pulsanti, uno dedicato al salto, uno ai pugni e un terzo ai calci. Ma combinando questi tasti possiamo realizzare mosse speciali come calci volanti, calci all'indietro e le mitiche gomitate che rappresentano una tecnica vincente per abbattere ogni cattivo. Dando poi due tocchi veloci allo stick si può usare anche una terribile testata pronta a stendere il malcapitato. Anche il contatto con il nostro avversario è particolare perché possiamo scegliere di atterralo con colpi singoli o bloccarlo con colpi continui e poi lanciarlo in aria.

A questo gameplay ricchissimo di novità bisogna aggiungere anche una grafica curatissima per quel periodo, con gli scenari ricchi di particolari e colori. Anche gli sprites dei protagonisti sono vincenti con un abbigliamento tipico delle bande di strada degli anni settanta e ottanta fatto di pantaloni strappati e semplici gillet per far risaltare muscoli e cicatrici. È piuttosto evidente infatti in *Double Dragon* il richiamo a *I guerrieri della notte*, film cult che tanto ha segnato la nostra infanzia.

Come se questo non bastasse, gli sviluppatori hanno riservato una bella sorpresa nel finale. Infatti se si termina il gioco in modalità cooperativa bisogna combattere tra di noi in uno scontro fratricida per conquistare il cuore della bella Marion.

Double Dragon ha avuto in sala giochi due seguiti rimasti però lontani dal successo del primo capitolo. Se infatti *Double Dragon II* mantiene lo stesso impianto grafico ma fallisce nel tentativo di innovare le tecniche di combattimento, il terzo seguito si distacca in tutto dai suoi predecessori e per questo motivo non ha mai entusiasmato i numerosi fan della saga.

Diverse poi le conversioni per le varie console casalinghe. Tra queste è doveroso ricordare l'ottimo adattamento per il Sega Megadrive e il tentativo, per lo Snes, di rilanciare la serie con *The return of Double Dragon* che, pur discostandosi dall'originale per grafica e gameplay, merita di essere giocato.

Cercando di sfruttare l'effetto nostalgia, la stessa Tecnnos Japan ha lanciato poi nel 1995 per la Neo Geo un *Double Dragon* in versione picchiaduro ad incontri. Il titolo però, seppur pregevole, è rimasto un pò anonimo, schiacciato dai pilasti del genere come *Street Fighter* e *Mortal Kombat*.

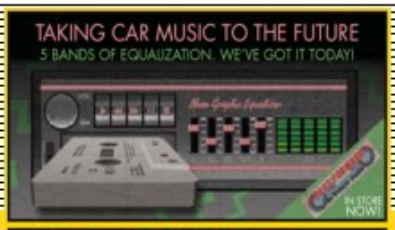
Querino Ialongo

THE SPRINGFIELD CITY Inquirer

Issue 10 Vol. 1

CITY ORIENTATION

SPECIAL SUPPLEMENT



Grand Theft Retro Magazine special

Brewed by IU4ASH

Y EST HOSPITAL

V ADS TEAMED U

A UTODELUXE FAMILY CO.

N GYMNICKS

E ASTERRAY

H ARDEE GOLF

T HE BOMB FAM

N. BNB

I NTERCITY #1 MART

O RCHESTRA ROYAL

J. R. WEST PATH

Un pomeriggio in sala giochi

di Marco Petri - Grafico 3D e Psicologo

UN POMERIGGIO-TIPO DELL'EPOCA

Ore 16/16,30 di un pomeriggio tra il 1987 e il 1990, finiti i compiti, una veloce merenda e via, si usciva fuori tra le strade di quel quartiere che sembrava un doposcuola a cielo aperto tanti erano i marmocchi che giocavano per strada.

Immagine da Google

Cuffie del Walkman sulle orecchie (dopo aver riavvolto con una matita o una penna Bic la musicassetta per risparmiare le preziosissime batterie non ricaricabili), "Bad" di Michael Jackson e "Papa don't preach" di Madonna a tutta birra e via all' appuntamento sotto casa dell'amichetto del cuore; sì, quello che ci faceva aspettare 20 minuti prima di scendere!

Una passeggiata per la via principale del quartiere, la via Tiburtina, e poi in SALA GIOCHI per un bel doppio all'arcade preferito di turno, proprio quello che dopo avervi sottratto decine di pezzi da 200 lire, vi concedeva l'onore di arrivare all'ultimo quadro e, ancora più raramente, di finirlo attirando l'attenzione di decine di altri ragazzini che si raduvano attorno al cabinato di gioco rendendoci l'eroe del momento e non solo!

Quanti di voi tra i 35 e i 45 anni (ma anche 50) si riconoscono in questa descrizione di un pomeriggio-tipo posizionato tra i primi anni '80 e i primi anni '90?

LE VITA NELLE SALE GIOCHI

Prima della stesura di questo articolo, mi è stato detto che un lettore di RetroMagazine ha chiesto come si viveva in sala giochi in quei tempi e cosa rappresentava per noi. Eccovi dunque un resoconto/ricordo dal punto di vista di un grafico 3D e psicologo quarantaduenne quale attualmente sono.

Queste righe, come mia abitudine, non hanno la pretesa di essere un "verbo divino" ma solo di stimolare la riflessione e la consapevolezza di quelle emozioni per poter considerare come sono cambiati i costumi della nostra società e magari considerare alla fine se davvero si è andati verso un miglioramento.

L'epoca d'oro delle sale giochi dura circa un decennio andando, come detto, dai primi anni '80 ai primi anni '90 con punte massime raggiunte tra l'85 e il '90.

Purtroppo non posso fornire dati statistici ma a giudicare dal numero delle sale giochi presenti nel mio quartiere di Roma (Portonaccio-Monti Tiburtini) e in quelli adiacenti (Pietralata, Casal Bruciato) e considerando il declino subìto negli anni successivi, direi che è una valutazione verosimile.

Le sale giochi, comunemente chiamate tra noi ragazzini dell'epoca "bische", rappresentavano dei veri e propri aggregatori sociali. In sala giochi si potevano ritrovare gli amichetti compagni di scuola, visto che all'epoca (non so se sia ancora così) gli istituti scolastici davano la precedenza ai residenti della zona.

La "bisca" quindi fungeva certamente da collante sociale perchè permetteva in tutta libertà, senza le costrizioni scolastiche, di poter sviluppare le amicizie con chi si riteneva più opportuno. La sala giochi era dunque un punto di riferimento per i giovanissimi del quartiere al pari della scuola e quando si è concentrati in un unico punto con la stessa passione e lo stesso obiettivo, bhè, la ricetta per creare un ambiente dove ci si relaziona è pronta!

Ma qual era questo obiettivo? Forse all'epoca antecedente, quella dei flipper, consisteva nel più alto punteggio che si riusciva a fare, il famoso "Hi Score". Non che non fosse ancora presente qualche flipper ai miei tempi ma l' Hi Score non era più ambito da nessuno; piuttosto si cercava di raggiungere il quadro più avanzato possibile e magari, perchè no, di finire il gioco.

Immagine da Google

E quando questo succedeva, bhè, era un vero e proprio evento! La folla attorno al cabinato diventava via via più numerosa quanto più si avanzava nel gioco, era una vera e propria legge di proporzionalità diretta! Una folla di amici, conoscenti e/o sconosciuti, rimanevano con gli occhi incollati verso quei quadri che non avevano mai visto, affascinati da tanta novità ma anche dalla speranza di scovare qualche trucchetto che permettesse anche a loro di arrivare tanto lontano nel gioco.

Tu che eri perfettamente concentrato sull'azione, madido di sudore (soprattutto nei mesi estivi) e ti accorgevi della folla che si era radunata attorno a te solo quando, esalata l'ultima vita del personaggio di turno, tornavi alla realtà lasciando i comandi del videogioco.

Nel frattempo, come in ogni comunità che si rispetti, la notizia del tuo record aveva praticamente fatto il giro del quartiere.

Era tutto un fatto di "cannoncini" (le famose vite che si avevano a disposizione, mai capito perchè venissero chiamate così!). Vigeva poi la regola condivisa anche se non palesata, che valeva considerare finito il gioco con un solo gettone o moneta da 200 lire che fosse.

Perchè tutta questa meraviglia? Perchè quegli arcade erano, mediamente, dannatamente difficili! Era anche logico in fin dei conti: dovevano garantire una longevità di gioco particolarmente alta per sottrarre il maggior numero di soldi ai giocatori. Quando infatti se ne raggiungeva la conclusione, bhè, era come

aver raggiunto la meta di un viaggio: finiva tutto, l'impresa passava alla storia (e non sto esagerando, dentro me conservo ancora il netto ricordo degli arcade finiti!) e si passava a un altro gioco.

Ma torniamo all'aspetto psicosociale in relazione a quello tecnico. Perchè le sale giochi hanno avuto così tanto successo? Prima di tutto, non si era mai visto niente di simile prima di allora e si sa, i giovani soprattutto sono attirati dalle novità. Sì, le console erano cominciate ad apparire anni prima, sul finire degli anni '70, e home computer come il Commodore64, lo Spectrum e l'MSX erano già presenti sul mercato. Anzi, se si pensa che nell' 87 era già disponibile l'Amiga 500, ci si può chiedere perchè la gente spendesse soldi, pur magari avendo già un computer a casa, negli arcade da bar.

Le ipotesi di risposte sono molteplici:

1) La qualità dei giochi: l'arcade da bar rappresentava il non-plus-ultra della tecnica di quel periodo. La grafica, la velocità, il suono, la possibilità di giocare in doppio, i tempi di caricamento nulli..... Le console e gli home computer, non potevano certo competere. L'arcade rappresentava il limite massimo verso il quale venivano paragonate le conversioni dei titoli che le software house facevano per le macchine casalinghe. E quasi sempre questi ultimi ne uscivano sconfitti! Coi computer a 16 bit quali Amiga e Atari ST, le cose migliorarono un po' e in alcuni casi, ricordo conversioni più che decenti; tuttavia gli investimenti e il numero di persone coinvolte nello sviluppo di un gioco "home" non erano paragonabili a quelli tipici del mondo arcade da bar.

Una sola eccezione, che scoprii solo grazie a internet molto tempo dopo, ha visto delle conversioni pressocchè perfette: erano quelle effettuate per il computer Sharp X68000 di cui all'epoca tuttavia, noi comuni mortali ne ignoravamo addirittura l'esistenza.

Fatto sta che chi, come il sottoscritto, viveva di scuola e sala giochi, usciva quasi sempre frustrato dal confronto tra i giochi del C64 e quelli da sala giochi: uno dei motivi che mi spinse a comprare l'Amiga 500.

2) Un meccanismo neuro-psicologico alla base del successo del gioco in genere (anche di quello "home"), è il soddisfacimento del "sistema della ricompensa", ossia "quel gruppo di strutture neurali responsabili della motivazione, dell'apprendimento associativo, e delle emozioni positive, in particolare quelle che coinvolgono il piacere come componente fondamentale (ad esempio, gioia, euforia ed estasi)" (Fonte: Wikipedia). E direi che di gioia ed euforia nel riuscire a finire un gioco ce ne era parecchia.

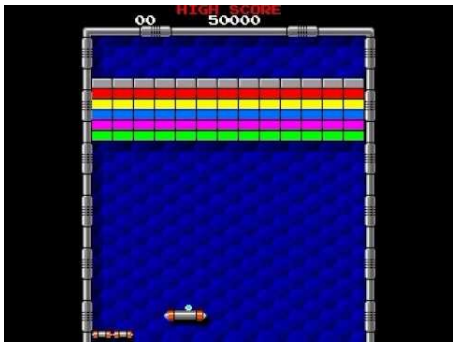
Bene, immaginatevi ora questo meccanismo ma condiviso con altre persone coetanee: questo soddisfacimento trova conferma nell'altro individuo e si rafforza ulteriormente.

3) Altro meccanismo era quello del conformismo e dell'identità sociale. Il primo concetto lo potremmo riassumere con la frase "mamma, vanno tutti in sala giochi, ci vado anche io" mentre il secondo con la frase "mi identifico e mi riconosco nei valori e negli obiettivi del gruppo di coetanei che va in sala giochi"; questo è un processo importante dello sviluppo che concorre alla definizione dell'Io e quindi della sensazione di essere, e di far parte di qualcosa, della persona. Non possiamo dimenticarci infatti che la sala giochi di quartiere in fin dei conti contribuiva sicuramente a far sentire meno isolati molti ragazzini.

4) Ancora, nelle sale giochi l'individuo imparava le strategie di problem-solving condiviso, ossia delle azioni da intraprendere per superare una certa difficoltà del videogioco. Se si considera poi che spesso questo avveniva con persone del tutto nuove, non ci risulta difficile comprendere come potesse essere un ambito che favorisse le relazioni sociali.

Da non dimenticare poi un aspetto secondario non meno importante: una volta finiti i (pochi) soldi a disposizione, per il fatto che ormai si stava fuori casa, non rimaneva altro da fare che prendere un pallone a andare a giocare per strada! Credo infatti di poter affermare di appartenere all'ultima generazione che si è sbucciata le ginocchia giocando per le strade di un quartiere di Roma. Tutto questo contribuiva non poco al proseguimento e consolidamento di quella socialità che magari era iniziata proprio in sala giochi.

Ultimo, ma non meno importante, devo affermare che il fenomeno delle sale giochi ha contribuito non poco a cementificare il rapporto con mio padre.



Arkanoid

Sì, perchè spesso era lui che mi ci accompagnava (quando ero più piccolo) e con questa scusa, si metteva a giocare anche lui (era un drago ad Arkanoid che riusciva a finire con una sola moneta da 200 lire!). Spesso giocavamo insieme, lui guidava la racchetta e io sparavo quando questa si trasformava dopo aver preso la pillola rossa con la L sopra. Oppure mi ricordo le partite a Moon Patrol, dove io guidavo e sparavo e lui mi aiutava a saltare le fosse create dalle bombe lanciate dalle navicelle nemiche.

Moon Patrol

Andare a trovare nuove sale giochi diventava poi anche un pretesto per farmi conoscere Roma, oppure per fare delle passeggiate la domenica con la bicicletta: "dove andiamo papà?", e lui "mha, vediamo di raggiungere quella sala giochi laggiù, facciamo una partita e torniamo".

ERANO TUTTE ROSE E FIORI?

Bhè, siamo onesti: si sa che troppo spesso i ricordi sono ammantati di un velo di esagerazione, in senso positivo o negativo. Queste "bische" avevano solo riscontri positivi? Non proprio. Uno dei difetti ad esempio, è che spesso alcune di essere

diventavano punti di riferimento anche di figuri non proprio raccomandabili.

In quegli anni andava molto l'eroina (a testimoniarlo erano le numerose siringhe che si potevano trovare sui marciapiedi di certe vie dove magari si giocava a pallone) e si sapeva che nel retrobottega di alcune bische era meglio non andare. Là dietro c'erano i biliardi, non ci interessavano certi giochi, roba da grandi. E devo dire, in tutta onestà, che rispettando questa regola non è mai successo, nè a me nè ai miei amichetti, nessun episodio spiacevole. Ma non tutte le sale giochi erano ovviamente sottoposte a questo compromesso, non tutte cioè avevano un lato malfamato, per carità.

Altro lato distorto dell'intera faccenda: ogni tanto giungeva una notizia di qualche ragazzo, magari più grandicello, che aveva osato prendere dei soldi di nascosto ai genitori per andarseli a spendere in sala giochi. Oppure quelli un po' più bulletti che ti tampinavano perchè volevano scroccare una moneta/gettone. Però faceva parte del gioco: si imparava a trattare questi personaggi non proprio positivi, si imparava in parole povere, a vivere socialmente.

...E POI IL DECLINO

Ma come tutte le cose, quell'era doveva finire. Un successo, devo dire, piuttosto effimero se si considera che è durato pressapoco solo una decina di anni e molti dei coetanei che entrano in casa mia, neanche riconoscono cosa sia il cabinato d'epoca posto in bella vista nel mio soggiorno. Sembra quasi che ci sia stato un reset della memoria collettiva in questo senso.

Le cause del declino?

Credo siano state diverse:

1) disinteresse nei confronti dei generi di giochi tipici arcade (spara-tutto, platform, qualche gioco di guida);

2) aumento delle capacità grafiche dei personal computer e delle console che permettevano ormai livelli tecnici pari se non superiori (tipo la Playstation 1 del 1994 e poi a seguire le schede acceleratrici 3fdx prima e Nvidia dopo che portarono i giochi con visuale in prima persona a livelli sempre più realistici).

Mi sono sempre chiesto tuttavia come mai non si sia conservata una nicchia di mercato. Ormai le sale giochi sono quasi del tutto scomparse e se esistono è veramente difficile trovarvi dei cabinati.

In effetti, credevo che si sarebbe verificato quello che è avvenuto con gli e-book: si sono diffusi ma non hanno del tutto soppiantato i libri tradizionali; credevo dunque che, al pari delle odiose slot machine apparse in seguito, i cabinati arcade avrebbero continuato ad esistere tra i giovani. In fin dei conti, l'arcade è una tipologia di gioco che ha sempre un certo ascendente. Eppure non fu così.

E poi arrivò quel geniaccio di Nicola Salmoria col suo MAME nel 1997 che permise la realizzazione del sogno di qualsiasi ragazzino degli anni '80: poter giocare i titoli da sala giochi sul proprio home computer! Tuttavia ormai era troppo tardi: non era certo la stessa cosa che stare in una vera sala giochi.

E ORA? E' VERAMENTE TUTTO MIGLIORE?

L'avvento di internet poi ha contribuito a cambiare i nostri costumi: "perchè uscire di casa quando posso giocare in condivisione non solo con i ragazzi della mia scuola e del mio quartiere ma con chiunque nel mondo?". Se da una parte questo enorme incremento tecnologico ha portato degli indubbi benefici, non ho paura di fare la figura del bigotto affermando che, per contro, ha certamente contribuito alla sedentarietà e soprattutto all'isolamento fisico delle nuove generazioni. Ottimo parlare, magari in inglese, con qualcuno che vive in Australia; un po' meno relegare al solo approccio virtuale quella che prima sarebbe potuta essere una relazione umana, reale, viva, fatta non solo di scambi testuali ma anche e soprattutto di comunicazione non verbale visiva.

Certo, non voglio ora dire che non ci siano altri ambiti dove, attualmente, un ragazzino possa sviluppare gli stessi concetti, ci mancherebbe. Vi sono le scuole calcio, le palestre, le piscine, i doposcuola, i circoli estivi...

Ma ci sono diversi "però": prima un ragazzino di 10 anni gridava alla mamma "Ma', io scendo in sala giochi", andava all'isolato a fianco e la mamma era sicura che lo avrebbe trovato lì. Ora la vedo difficile che lo stesso ragazzino possa incollarsi una borsa da calcio di 15 Kg e andare alla scuola calcio che magari dista 2-3 quartieri dal suo. Sì è persa quella che potrei definire, con un neologismo, la "quartierosità" del luogo in cui si vive.

Altro aspetto, che grava sulle famiglie, è il lato economico. Prima, qualche ora di intrattenimento e di ambito sociale (se si considera anche il post-sala giochi), poteva costare una manciata di monete da 200 lire; quanto costa invece oggi una palestra, una piscina o una scuola calcio?

Posso inoltre osservare che si è persa inoltre quella spontaneità (il ragazzino che dice alla mamma che esce e che si reca in sala giochi) che garantiva lo sviluppo dell'indipendenza della persona. Oggi vedo questi ragazzini che vanno sì in palestra, piscina, etc, ma sono in molti casi accompagnati, magari con la macchina, obbligati forse anche dal fatto che la mamma li ha "incastrati" nel frequentare determinati ambiti perchè ormai ha pagato la retta.

COSA RIMANE OGGI?

Per fortuna, come già detto, grazie al sommo MAME (e diversi add-on e versioni modicate) è possibile ricrearsi (o restaurare) un cabinato dell'epoca, una cosa che vedremo in un futuro articolo.

Per il resto, quelle esperienze e quelle emozioni risuoneranno per sempre nei cuori e nelle menti di chi quelle emozioni, dinamiche e atmosfere tipiche delle sala giochi di quartiere le ha vissute dal vivo.

Giappone seconda puntata: l'occidentale limbo dei quotidiani déjà-vu

di Michele Ugolini

"Oh no! Scoppia tutto , proprio ora che andava tutto bene..."

"Ecco lo sapevo! Arriva un altro mostro..."

"Aiuto! Ma chi sono questi? Cosa vogliono da noi?"

Cari lettori, queste frasi gettate quasi apparentemente senza senso, vi suggeriscono qualche indizio?

No, non mi sto burlando di nessuno, non sono neppure i primi segni di qualche disordine neurologico: fidatevi, è tutto "causale", nulla è "casuale"!

In questa puntata vorrei portare la vostra attenzione sulle ferree regole della creazione multimediale nipponica, ritmata, scandita, efficace nei propri stereotipi e programmi, finalizzati ad attirarci per poi renderci parte dell'ingranaggio.

Per chi non ha letto la mia prima recensione del Retrogaming nel mondo nipponico, in RM13 , vi invito a tale lettura, sarà un aperitivo particolarmente bitter. Leggetelo, non fatevi convincere dalle mie osservazioni, ragionate con il vostro intuito, poi proseguite in questa seconda recensione.

In questa lettura, ci addentriamo nella complessa dinamica dei déjà vu occidentali, sapientemente gestiti dai nostri amici nipponici, presenti in situazioni incredibilmente frequenti. Dinamiche estremamente complesse e così ben studiate che sembrano a volte provenire dal mondo della moda. Non è forse proprio l'abito di alta moda ad essere intriso di bellezza a colpo d'occhio nonché frutto dello studio approfondito di diversi fattori, sin dalle origini della propria creazione?

Vorrei parlare delle dinamiche nipponiche, tanto frequenti da determinare in noi un immediato riconoscimento della loro matrice, appunto a colpo d'occhio. Più precisamente parlerò della costellazione di elementi da loro usati, per attuare l'imprinting sui loro seguaci, nel più profondo senso del significato: si, sto parlando di noi!

In questa seconda parte, infatti, saranno commentate e non giudicate, alcune loro basi psicologiche che attuano, che hanno attuato e che per l'eternità attueranno su di noi, a partire dalla nostra gioventù, per realizzare in noi alcuni automatismi: il discernimento e la conseguente valutazione positiva, oppure negativa, degli elementi del nostro vissuto che quotidianamente ci circondano.

Cosa c'entra con il Retrogaming? Assolutamente tutto, perché ad oggi, noi occidentali, coccolati e viziati dalla loro cultura, nonché dai loro prodigi multimediali, non sappiamo di non conoscere diversi elementi delle loro dinamiche condizionanti, tanto che, per assurdo, potremmo giudicare negativa una dinamica della nostra società, "a pelle", "a colpo d'occhio", senza comprendere realmente le regole che hanno generato tale giudizio così automatico.

Sembra che io stia disquisendo il tutto con un approccio nebbioso e misterioso. In realtà i dubbi si dissiperanno mano a mano che smonterò il sapiente ingranaggio nipponico. Potremo comprendere meglio perché una nazione così lontana da noi abbia realmente mutato il nostro scibile, la nostra cultura ed i nostri odierni giudizi, su tutto ciò che ci circonda e ci appartiene!

La sconfitta compatta, ma non delle intenzioni

Il primo passo per comprendere queste dinamiche implica una breve digressione storica. Siamo nell'immediata fine del secondo conflitto mondiale. L'esuberanza nipponica aveva giocato le proprie carte in maniera piuttosto discutibile.

Ricordiamo la loro dichiarazione di guerra a Cina, invasione in Corea, espansionismo asiatico in Thailandia, Filippine, Myanmar, intenzioni di guerra contro Unione Sovietica per via dell'alleanza tedesca ed infine la tanto nefasta provocazione a Pearl Harbour.

L'esercito nipponico, e purtroppo la popolazione innocente, avevano tristemente e duramente appreso da due bombe atomiche, sganciate a tre giorni di distanza, quanta discrepanza esistesse riguardo le proprie intenzioni rispetto alle proprie possibilità. Il Giappone perse la guerra. Il Giappone non perse le proprie intenzioni. La popolazione nipponica si ritrovava con le spalle al muro ed era ormai inevitabilmente uscita dal profondo isolamento nazionale per via dell'irreparabile ferita atomica subìta. La loro comunità viveva un'esperienza toccante di infinita impotenza. Differentemente, noi occidentali, abbiamo vissuto una logorante esperienza di guerra fratricida, specialmente noi italiani, tra fazioni partigiane e fasciste. Il dolore di questo massacro sarà per sempre compagno del nostro scibile. I giapponesi, invece, come hanno metabolizzato la loro sconfitta? La sconfitta è avvenuta con estrema violenza, ma senza un fratricida logorio interno, quindi in maniera compatta. Probabilmente il principale ingrediente della loro perfetta integrazione meccanica al sistema, è stato forgiato a fuoco (e radiazioni) dal dolore della sconfitta, avvenuta in maniera così abominevole che li ha portati a stringersi e compattarsi uniformemente per ottenere l'unica possibilità di dominio sul mondo, forse anche per vendetta, ma soprattutto per un continuum della loro ideologia suprematista, attraverso la strada funzionalmente più vincente: l'intelligenza! Non la forza, ma l'onnipotente intelligenza.

Difficilmente si potrebbe immaginare che la bomba atomica, similmente all'Araba Fenice, abbia portato ad una evoluzione proficua. Eppure è proprio così. Chiaramente a modo loro.

AniMatrix nel loro secondo Rinascimento

Il DNA giapponese aveva immediatamente recepito che la mera forza nulla poteva rispetto all'elevazione intellettuale. Quindi il Giappone doveva cambiare. Completamente. In tutto. Proviamo ad assaporare per un attimo il nostro gioco Taito, Sega, Naomi, Snk, Nintendo preferito nella nostra infanzia. Cerchiamo di ricordare la ritmica scandita del gameplay. Ricorderemo sicuramente un ambiente di gioco difficile all'inizio, però costruttivo, che portava alla soluzione del gioco tramite piccoli passi, senza mai indietreggiare, una ritmica solida, ben scandita, precisa , tanto che "prendendo la mano" con un poco di esperienza , ci donava una forma di abilità quasi scontata nel proseguire.

Adesso catapultiamoci nel loro periodo post medioevale, poiché una buona dose della creazione di questa ritmica vede la luce dopo il loro periodo Edo (il nostro Medioevo). Difatti già nel periodo Meiji, che si concluse nei primi del 1900, il rigore assoluto, la precisione e soprattutto la puntualità erano diventati parte ineludibile del sistema. Dopo la potenza distruttiva subìta nella guerra , il loro DNA doveva mutare ad una ancor più affinata evoluzione di questi elementi innati, pur di riportare le loro intenzioni di dominio su tutto il mondo. Questa è la spiegazione della loro bandiera utilizzata in guerra (Hinomaru) che vediamo sempre nei videogiochi, specialmente nel Retrogaming. I raggi rossi del paese del "Sol Levante" si irradiavano e illuminavano "tentacolarmente" il resto del mondo. In molti giochi del passato possiamo notare questo irraggiamento sul globo. (cfr. fig. 1)

Figura 1

Questa costante è osservabile soprattutto nei loro titoli di guerra, fateci caso, è frequentissima. Dopo le ceneri, la resurrezione: studiarono con profonda precisione come attuare questa irradiazione. Non sto usando la parola "irraggiamento" con leggerezza, voglio proprio indurre al significato atomico con conseguente vendetta sull'Occidente attraverso il metodo del Contrappasso. Capirono immediatamente che le loro innate doti di precisione, calcolo matematico, estro creativo, puntualità ed integrazione sincronizzata al sistema interno potevano creare qualcosa di buono.

Ecco allora che si gettarono a capofitto nell'elettronica, calcolatrici, stampanti, computer, processori. Anche Jack Tramiel dovette fare i conti con questi inarrestabili produttori di prodigi elettronici. Poi entrarono nella motorizzazione: Toyota, Honda, Subaru, Mazda, Suzuki, Yamaha, Daihatsu, etc..

Poi unirono la motorizzazione meccanica all'elettronica e nacquero: Sanyo, Panasonic, Hitachi, National, Toshiba, etc..

Nel frattempo la loro inarrestabile evoluzione studiò il resto del mondo che arrancava e faticava nel rimodellare la propria identità nazionale. In questo preciso istante iniziava il loro Secondo Rinascimento, come spiegato in AniMatrix.

Qui entrarono, con perfetta simmetria temporale, nella nostra adolescenza. Vennero alla luce: Nintendo, Sony, Snk, NeoGeo, Sega, Taito, Naomi, Microcabin, Bandai, Enix, DataEast ed un altro centinaio di ditte apparentemente a noi meno conosciute. Forgiarono e delinearono in noi i confini di tutto ciò che stavamo apprendendo dal mondo, interferirono con i nostri sogni e il discernimento del giusto dallo sbagliato, creando attorno a noi il "loro" mondo.

Figura 2

Focalizzare o disconoscere?

Prendiamo come esempio "Nausicaa della valle del vento", di Hayao Miyazaki , capolavoro riconosciuto a livello planetario. Questo film di animazione era una trappola perfetta. La qualità grafica per quell'epoca era talmente evoluta che aveva fatto innamorare

Figura 3

e stupire, sia i ragazzi, che gli adulti dei primi anni 80. La strategia del film probabilmente riassumeva la loro ideologia di dominio mondiale? (cfr. fig. 2). Il film iniziava narrando un mondo distrutto dall'inquinamento, con piccole culle di umanità sopravvissute dalla distruzione fratricida. Forse si alludeva al mondo occidentale? L'umanità aveva desertificato il pianeta, trasformato il mare in acido. Le foreste lussureggianti, colme di vegetazione fantascientifica, erano divenute tossiche tanto che emanavano spore velenose per l'essere umano. In queste foreste vivevano in simbiosi animali fantastici, particolarmente feroci verso l'essere umano, poiché nelle epoche passate erano stati nemici mortali. Tutto era in equilibrio, una bellezza grafica da lasciare a bocca aperta. Gli animali che abitavano le foreste erano disegnati con una tale eleganza ed armonia di movimenti che difficilmente si potevano definire nemici.

Il film forse portava a riflettere sul vero significato di giusto e sbagliato? Insinuava soprattutto il "dubbio dentro il cuore" riguardo il giudicare una cosa corretta o errata? Amico o nemico? Le meravigliose foreste velenose erano nostre nemiche? Il mare acido, così popolato da esseri viventi mai visti, era realmente mortale? Le spore che piovevano dal cielo come neve, che ricoprivano le foreste e creavano fantasiosi disegni sospinte dalle correnti degli animali volanti, erano realmente pericolose? L'essere umano che aveva distrutto la natura e decimato i propri fratelli e che soprattutto dalla stessa natura era stato esiliato, per sopravvivenza, in piccole tribù ai confini delle foreste tossiche, era realmente l'essere vivente più simbiotico del regno animale?

La medesima ritmica... i medesimi elementi scenografici... gli stessi temi, li possiamo analizzare anche in: "Una tomba per le lucciole", film d'animazione, 1988 tratto dall'omonimo racconto semi-autobiografico di Akiyuki Nosaka. Le lucciole infuocate che cadevano su tutto il Giappone non erano altro che le scintille ardenti delle città che bruciavano per via dei bombardamenti incendiari dell'aviazione americana (cfr.fig. 3).

Non erano più le lucciole primaverili che volavano nei loro ambienti naturali magnificamente disegnati all'inizio del film. Nell'ambiente dei videogiochi si ripete la medesima metodica, tutto è in equilibrio, esattamente come un ipotetico foglio di carta bianca della nostra infanzia, pronto per essere scritto dalle esperienze della vita, con mille quesiti, tutti da guadagnare attraverso gli esempi della vita. Con l'avvento di questa meccanica nipponica non c'è nemmeno bisogno di uscire da casa per guadagnarsi l'esperienza di giusto o sbagliato commettendo errori nella vita: tutto era già comodamente servito sul piatto di argento, davanti alla televisione.

Oltretutto i nostri genitori erano inconsciamente contenti di questo "celato lavaggio del cervello", pur di salvare le nostre ginocchia rovinate dai giochi in strada! Ricordando questa scrittura del nostro foglio bianco tramite l'analisi dei valori odierni, non vi suggerisce un retrogusto di pacata, ma subdola, violenza ed una delicata, ma cinica, tortura?

Proseguiamo. Dopo questi erosivi dubbi, la meravigliosa animazione del film ci contagiava nel sostenere la causa dell'equilibrio pacifico tra i due regni, quello degli animali mostruosi nella foresta tossica e quello dei mostri umani che avevano distrutto il pianeta.

Allo stesso tempo la trama del film ci induceva a perorare questo equilibrio, coccolandoci attraverso una animazione ricca di elementi esplorativi, animali fantastici, tribù che vivevano con pochi elementi naturali rimasti fortunatamente non mortali per l'essere umano. La trama e gli ideali ci fornivano quindi le prime scritture del foglio bianco descritto prima, rispondeva con estrema comodità alle prime domande della nostra infanzia ed ammaliava i nostri genitori, più adulti, inculcando in loro il dubbio sulla ragione della guerra e soprattutto sul concetto di giusta vittoria o dolorosa sconfitta. Il tutto adornato di meraviglia grafica impensabile negli anni 80 occidentali.

Questo è l'imprinting. Diveniamo parte del sistema. Lo giudichiamo perfetto, stabile, meraviglioso, tanto da doverlo proteggere da qualsiasi elemento che perturberà un tale livello di elevazione.

Ecco perché all'improvviso il Giappone deve creare il giustificato odio per il nemico che, similmente ai bombardamenti americani, pioveva orridamente dal cielo. Con una dolorosissima lacerazione della nostra anima, il Giappone perseguiva il suo intento di mostrare quanto il mondo occidentale sporcava e distruggeva la loro onnipotente ed indiscussa bellezza.

Ecco che arrivano dall'alto i mostri, il fuoco, la morte, la distruzione, la disperazione. Ora può apparire più chiaro il progetto giapponese di procedere senza pietà nel torturare e graffiare il nostro cuore ormai innamorato e fidelizzato al loro sistema. Ecco che noi stessi, facenti ormai parte del loro ingranaggio, ci ritroviamo complici nel combattere il nemico.

Ecco che loro, invece, con cinica lucidità, sono funzionalmente perfetti nel dettagliare gli elementi chiave che ci permetteranno di riconoscere il nemico da combattere. Semplice imprinting.

E' realmente difficile catalogare i videogiochi del passato privi di questi tre elementi: un nemico che arriva dall'alto e mette tutto a ferro e fuoco, oppure un grande fungo nucleare, oppure una forza sconosciuta sovradimensionata che perturba con sproporzionata forza il quieto vivere. Più che stereotipi, questi sono programmi calcolati con fredda coscienza dal sistema nipponico, in maniera pressoché ubiquitaria, sia nel loro mondo multimediale che videoludico, sia nel passato che nel presente e nel futuro.

Perché questa pratica così cinica? Perché questa necessità di perfezione assoluta nei loro ingranaggi socioeconomici? Perché ci innamoriamo dei loro videogiochi moderni, amiamo e abbiamo amato i loro prodotti videoludici passati, tanto da farli diventare "parte di noi"? Infine, perché ad oggi non riusciamo più a focalizzare questi elementi condizionanti per poi estrapolarli dai loro prodigi multimediali?

La risposta è apparentemente semplice, complice del fatto che ormai siamo stati allenati alla miopia, programmata a puntino per eliminare in noi gli elementi chiave che ci permetteranno di disconoscere il lavaggio del cervello subìto. Analizziamo allora la multifattorialità di questa miopia programmata.

Volere è potere attraverso un'overdose nella distruzione di massa

Personalmente parlando trovo che questa miopia programmata è difficile da metabolizzare una volta che il tarlo del sospetto inizia a scavare in noi. Il Giappone ci ha quindi donato, oltre che le coccole, anche una grande cattiveria gratuita?

Questa cattiveria gratuita è simile a quella che potremmo indurre in un Otaku giapponese, lettore di manga, magari chiedendogli "tu leggi prima il testo dentro il fumetto o guardi l'immagine dentro il riquadro?". Perché i giapponesi hanno deciso senza alcuna ombra di dubbio di adottare questa punizione, con tale fermezza, così cinicamente nella nostra infanzia?

Figura 4

Le radici della risposta sono storiche. Il Giappone poco prima della resa, era quasi completamente distrutto, dopo i bombardamenti che non avevano risparmiato quasi nessuna struttura. Si salvava appena l'antica capitale religiosa di Kyoto ed alcuni lembi delle campagne. Dall'alto piovevano bombe incendiarie che incenerivano interi paesi costruiti in legno, piovevano bombe esplosive sui laboratori dell'arsenale nipponico, pioveva cenere nera in continuazione, vista la potente ed indistinta combustione di tutto quanto. I giapponesi erano stremati.

Allora, perché è stata utilizzata per ben due volte l'atomica? Forse per testare sul campo e sulla vita in carne ed ossa gli immondi effetti distruttivi? Classico esibizionismo americano per far capire al resto del mondo "chi comanda" realmente? Dare il colpo finale all'esercito nipponico che mai e poi mai avrebbe dichiarato resa fino all'ultimo uomo sopravvissuto sulla più sperduta isola dell'oceano? Si, certo, la storia è verissima, per trent'anni questo signore continuò la guerra che in realtà era finita: https://it.wikipedia.org/wiki/Hiroo_Onoda.

Anche nel film "Chi trova un amico trova un tesoro" di Bud Spencer e Terence Hill si scherza riguardo a questo buffo aneddoto.

La triste risposta è questa: la bomba atomica è stata usata per il semplice fatto che gli americani "potevano". Quindi i giapponesi hanno attuato questa vendetta culturale poiché nella propria coscienza collettiva "dovevano e potevano". Spieghiamo meglio questo concetto.

Stupire inutilmente oppure inorridire con scenografica esagerazione?

Vogliamo parlare del fragoroso fungo atomico di Yattaman, Calendar Men, Neon Genesis EVAs, Gundam, Mazinga, Cyborg 009? (cfr. fig.4) Vi consiglio anche di rivedere il cinquantesimo episodio dell'Uomo Tigre (Ricordo di Hiroshima) e naturalmente leggere il potente ed emotivamente dilaniante Gen di Hiroshima. Parlando di Daitan3, vogliamo ricordare il famoso Attacco solare? Sembra o non sembra una potenza distruttiva, titanicamente sproporzionata, rispetto al nemico da distruggere? Perché veniva utilizzata? Semplice, era parte del proprio corredo, quindi poteva. Vogliamo parlare del tuono spaziale di Goldrake?

Non sarebbero servite queste armi di esagerata potenza distruttiva per eliminare il nemico. Sarebbero bastate le comuni armi in dotazione per sconfiggere l'avversario senza l'ausilio di armi così spettacolari, irraggianti, esplosive, scenografiche e sproporzionate, che ci ricordano così tanto lo squilibrio di similitudine tra un ordigno nucleare esplodente di 25 Kilotoni ed il paesino costiero di Nagasaki, costruito in legno, con rari edifici in muratura. Questo concetto di mancato equilibrio doveva naturalmente

Figura 5

creare uno shock in noi, ormai ingranati nel loro sistema, per spingerci a difenderli dai nemici ed infine perfezionare la focalizzazione sulla sconfitta degli odiati usurpatori.

Questi concetti di profondo disequilibrio, sproporzionato, superfluo, scenografico, distruttivo ora vi sembrano essere tutti gli ingredienti base del Retrogaming nipponico? Probabilmente si, magari in alcune forme più celate o più evidenti, dipende dal contesto settoriale del marketing. Questi ingredienti, personalmente parlando, li ho riscontrati come target di tutto il business destinato all'Occidente, soprattutto nei prodotti passati, dove l'imprinting era fondamentale per attirare i futuri proseliti. Noi. Ho detto "prodotti Passati"? Prima parlavo di Presente. Vogliamo fare un esempio che integri il Futuro? Nulla di più facile, la fusione di passato, presente e futuro è una costante in tutti i loro prodotti. Pertanto cito solo il titolo di un film di animazione, semplicemente meraviglioso, assolutamente da vedere, da far rabbrividire per tutta la sua durata: "E' questo il tuo nome?", "Kimi no na wa", meglio conosciuto come "Your name", di Makoto Shinkai, anno 2016! (cfr. fig.5).

La divulgazione occidentale: conclusione e perfezionamento di elusione ed identità!

Riassumendo, nel nostro amato mondo del Retrogaming, ci sono tutte queste tappe dell'identità nipponica, una denuncia sociale sapientemente celata, nascosta, data in pasto a noi occidentali, che rispecchia la loro solida base di esperienza vissuta.

Esattamente come al tempo di Raffaello, Michelangelo e Da Vinci: si dipingeva in chiave criptica sia il bello che soprattutto il cattivo tempo che accompagnava la situazione socioeconomica (e tantopiù religiosa) di tali epoche.

E' tutto abbastanza intuitivo ora. Analizziamo dunque le ultime "apparenti" discrepanze nipponiche che amalgamano tutte le logiche descritte sopra. Il Giappone aveva perso per colpa dell'atomica made in USA... però oggi l'ombrello atomico di difesa della patria è garantito proprio dagli... americani! Il Giappone è un colosso dell'elettronica di precisione... però prima del nucleare era un paese visionario e guerrafondaio dotato di un apparato bellico carente in tutti i punti. Il Giappone vive proprio grazie all'energia atomica... proprio l'energia che gli ha portato morte ed indicibili sofferenze dopo l'irradiazione. Il Giappone nel 2012 crea la New Tokyo Tower alta 634m , seconda in altezza solo al Burj Khalifa... però è anche il paese con i terremoti più violenti al mondo. (cfr. fig. 6).

Ecco perché i giapponesi vivono la vita appieno, in maniera folle ed esuberante. Ecco perché si godono lo stipendio e fanno girare l'economia così vorticosamente: nel loro DNA sanno che tutti questi elementi apparentemente contrapposti non sono un "ossimoro" ma una costante della propria esistenza, una realtà che potrebbe finire da un momento all'altro, come in Hiroshima e Nagasaki, o come a Fukushima (120.000 anime spazzate via dal mare in pochi minuti) oppure potrebbe arrivare il tanto temuto e preventivato terremoto "Big One" a Tokyo e demolire mezza metropoli.

Niente da fare, tutto prosegue, similmente alla canzone di Vasco: "tutto un equilibrio sopra la follia". Anche noi, ora, dopo aver analizzato tutto questo meccanismo, viviamo una sorta di identità nucleare ed alterità nucleare? Ci identifichiamo o distacchiamo da questi elementi? O forse anche noi, inconsciamente, da decenni, viviamo il conflitto di tale coabitazione? Se realmente amiamo i loro prodotti multimediali, abbiamo subìto anche noi il dolore di una perfezione nipponica graffiata e deturpata da un nemico invasore e distruttore. Siamo anche noi vittime di questa immonda fase storica? Odiamo anche noi ciò che li ha resi tali dopo le due atomiche? Amiamo anche noi ciò che li ha resi tali dopo le due atomiche?

Concordo con il professore Toshio Miyake dell'Università Ca' Foscari di Venezia, docente del modulo "Società giapponese contemporanea" che riassume il concetto delle due bombe atomiche con queste profonde parole : "[...] il nucleare diventa un simulacro postmoderno e globalizzato del

Figura 6

Giappone contemporaneo, inserendosi in una fase molto importante che riguarda anche l'attualità [...]". Forse ora, dopo aver letto ciò, siamo meno condizionati ma ben consci e serenamente lucidi, avendo compreso meglio le dinamiche dell'imprinting attuato già nella nostra gioventù? Vi ricordo che questi sono solo alcuni sparuti elementi che hanno forgiato il Retrogaming. Chiedetemi e vi porterò altre analisi, anche perché riuscire a descrivere tutte le loro evolutissime e concatenate dinamiche socioeconomiche è alquanto complesso e bisogna procedere per settori.

Concludo con una riflessione personale. Mi piacerebbe tantissimo ascoltare le vostre critiche (costruttive), i vostri punti di vista e la presa di coscienza nell'analisi dei mille déjà-vu trattati.

In realtà vi confesso che sarei ancor più felice di trovare davanti a me le più acerrime critiche provenienti dal ramo negazionista, sarebbe veramente magico se si potesse esaudire tale ideologia del "non accaduto".

Sarebbe bellissimo se si potesse realizzare questo ramo di pensiero, così potrei dimenticare tutto ciò che ho visto nelle città di Nagasaki ed Hiroshima, i musei, gli orologi che per l'eternità mostreranno l'ora dell'esplosione, i filmati, le bottiglie di vetro fuse in un blocco unico, i possenti cancelli di ferro liquefatti e gocciolanti come burro fuso e poi rappreso, fotografie di tutti i tipi di ustione e liquefazione di corpi e organi, proiezioni murali di bianche ombre umane su edifici anneriti, etc... (cfr. fig. 7, 8).

Figura 7

Un sentito grazie al Giappone, sia per il bene, sia per il male che ci ha donato in maniera più o meno celata. Fatemi sapere cosa vorrete sapere nel prossimo episodio. A presto!

Approfondimenti

1) "Gen di Hiroshima" (Hadashi no Gen: semi-autobiografico di Keiji Nakazawa) del 1976.

Attenzione alla visione. Il film mostra anche la terrificante procedura di scioglimento e demolizione dei corpi umani aggrediti dalle radiazioni nucleari. L'esposizione dei corpi entra in una distopica ed irreale visione di corpi vaganti in preda a dolori lancinanti che attendono solo di morire. Mostra il dramma del desiderio di morire quanto prima poiché le irreparabili e dolorosissime ferite sono una piccola parte della paurosa combustione interna ad opera dei cocenti ed invisibili raggi gamma. Mostra tutta la debolezza dell'essere umano, la sua permeabilità agli agenti atomici esterni senza poter fare nulla se non interrogarsi su quale sarà la prossima fase.

Figura 8

2) L'amatissimo film "Nausicaä della Valle del vento (Kaze no tani no Naushika)" è un film d'animazione del 1984 scritto e diretto da Hayao Miyazaki. Mostra tutte le tappe del condizionamento psicologico trattate sopra. Adatto per tutto il pubblico.

3) "Una tomba per le lucciole (Hotaru no haka)" è un film d'animazione giapponese del 1988. E' un film di struggente demolizione emotiva, che mostra perfettamente tutti gli argomenti trattati, dove gli insetti (appunto le lucciole) del titolo sono paragonati alle scintille sospinte dal vento divampate dalle città in fiamme dopo i bombardamenti. Bisogna vederlo solo se emotivamente pronti... al peggio.

Sono tutti film toccanti, colossi e testimonianza storica eterna del Giappone, assolutamente da conoscere per poter meglio comprendere la matrice delle creazioni videoludiche e multimediali nipponiche.

Buona visione!

Intervista a **Mauro Corbetta**

Intervista a cura di David La Monaca (aka Cercamon)

Siamo ad aprile 2019 e sinceramente 30-35 anni fa, quando il mercato dei videogiochi viveva un periodo d'oro con un numero ingente di titoli pubblicati per arcade, console e home computer, non avrei mai pensato che a così grande distanza di tempo ci sarebbe stata una tale vivacità attorno al retrocomputing. L'evoluzione dell'hardware, per molti di noi che l'hanno vissuta passo dopo passo a partire dai primi anni '80, è sempre stata costante ed in un certo senso *inesorabile*. Molte macchine promettenti in origine sono finite, più o meno rapidamente, nel dimenticatoio "generale", ma sono rimaste pur sempre vive e vegete nella memoria dei singoli appassionati. L'evoluzione del software di intrattenimento ha seguito da vicino un percorso parallelo ed integrato a quello del progresso dell'hardware ed oggi assistiamo ad un momento di rinnovata passione per tutto il mondo del cosiddetto *retrogaming*, sia da parte di coloro che negli anni '80 e '90 hanno trascorso ore di spensieratezza e di divertimento videoludico, sia da parte delle nuove generazioni di videogiocatori, che, spesso saturi di proposte moderne fatte da super-grafica ed animazioni ma scarsa interattività, guardano indietro con favore ai videogiochi di un tempo, dove gli elementi di giocabilità, longevità ed originalità prevalevano sui "muscoli" di silicio e cicli di clock degli odierni *kolossal* che girano su PC, PS4 e XBOX.

Retrocomputing e retrogaming stanno quindi vivendo una seconda giovinezza un po' in tutto il mondo e l'Italia è forse fra i paesi in cui il fenomeno è maggiormente in crescita, soprattutto grazie all'opera di alcuni appassionati che da anni e instancabilmente promuovono eventi, attività ed iniziative volte a riportare i fan di un tempo alle loro vecchie passioni e a preservare e divulgare fra i giovani la storia dei videogiochi. In questo numero di RM siamo molto lieti di intervistare Mauro Corbetta, che è fra coloro che nel nostro paese, si sono impegnati e s'impegnano in prima persona e con grande energia per conservare e diffondere la memoria e la cultura del videogame.

Mauro Corbetta al suo 'smartphone'

L'intervista

DLM. Salve Mauro e grazie per aver accettato l'intervista. Sappiamo che sei conosciuto nell'ambito del retrogaming, soprattutto grazie alle tante iniziative che nel corso degli ultimi anni hai portato avanti. Ma ti invito lo stesso a tracciare un tuo breve profilo a beneficio di coloro che non hanno ancora un'idea precisa di te e del tuo operato nel mondo videoludico.

MC. Ciao a tutti. Nel mondo del retrogaming essenzialmente molti mi identificano (ed io stesso mi identifico) con il progetto RetroEdicola Videoludica. Si tratta di un'esperienza molto articolata che nasce una trentina di anni di anni fa, inizialmente con l'intento di preservare e indicizzare ogni rivista videoludica pubblicata nelle edicole. Se pensiamo al periodo dei primi anni '90 e alla mole spropositata di riviste che uscivano, era quasi impossibile seguire tutto, eppure, grazie all'aiuto di tanti amici motivati dalla stessa passione non solo siamo riusciti nell'intento, ma abbiamo anche ricostruito la storia di tante avventure editoriali. Lo step successivo avviene in tempi più recenti: nel 2009, constatando il lento declino delle riviste (iniziavano a uscirne pochissime) e con più tempo libero a disposizione, ci siamo chiesti se era possibile, in termini di costi e qualità, effettuarne la digitalizzazione. Dopo circa un anno di tentativi e dopo aver contattato diversi autori ed editori, nel 2010 mettiamo su la prima versione del sito, che permetteva soltanto di scaricare il file PDF di alcune riviste. Ricordo con piacere l'elevato successo dell'iniziativa e nonostante non fossimo certo i primi, la qualità e la professionalità del nostro operato ci mise subito in luce. I primi tempi sono stati serratissimi e ci hanno permesso di conoscere tantissime persone ed espandere le nostre collaborazioni su tutta la penisola. Nei successivi 8 anni di attività ci siamo mossi sia nelle scansioni documentali - ad oggi siamo arrivati ad oltre tremila riviste digitalizzate - sia nel colmare i buchi della nostra emeroteca con una serie di pubblicazioni che non arrivava in tutta Italia e della cui esistenza onestamente non avevamo neanche idea, arrivando così ad un totale di oltre cinquemila pezzi. Le cose fatte sono tantissime, senza contare i viaggi per tutta la penisola per recuperare riviste, per partecipare ad eventi, per conoscere nuovi collaboratori o per intervistare autori ed editori. Personalmente sono arrivato a percorrere anche più di quattromila km in un anno. Tutti ci siamo dati da un gran da fare, insomma!

Lo step finale, o meglio, quello che ci porta alla situazione odierna, avviene nel 2007: a fronte di tanti collaboratori, per proteggere il nostro operato e per cercare un giorno nuove leve che speriamo proseguiranno il nostro operato (gli anni iniziano a essere tantini ;-)), abbiamo trasformato il progetto iniziale in un'Associazione Culturale e, contemporaneamente, ci siamo fusi con "RetroGaming Bergamo", una realtà locale che da diversi anni organizza eventi retrogaming nella bergamasca. Così abbiamo comprato un locale in città che fungesse da sede ed oggi funzione sia come centro per la consultazione della nostra emeroteca e per lo studio della storia del videogioco, sia come luogo d'incontro in cui settimanalmente organizziamo eventi di vario tipo, dalle serate *arcade* su proiettore agli incontri con autori e

protagonisti del mondo videoludico, fino agli workshop a tema.

Spero di non essermi dilungato troppo con questa risposta, ma RetroEdicola è davvero una grande *scatola*, a seconda di come si apre, si può accedere ai tanti aspetti legati al videogioco e più in generale alla cultura pop. Cosa ci aspetta il futuro? Da inizio di quest'anno, dopo 9 anni di pubblicazioni amatoriali, siamo diventati anche editori e a breve presenteremo il nostro primo libro. Invece di puntare su numeri monografici e pubblicazioni dirette sull'argomento "videogames", cosa che abbiamo già approfondito nel corso degli anni, abbiamo cercato di far emergere la nostra passione e ciò che ci sta più a cuore: "Retrovisioni" sarà una raccolta di racconti brevi (8 come i bit dei primi home computer) che ruotano attorno alla tematica Videogioco e Cultura Pop, scritti da professionisti e non. Il calderone di idee in seno all'Associazione è poi sempre sul fuoco ed ogni giorno siamo pronti per una nuova sfida ;-).

DLM. Come ti definiresti per dare subito un'idea della tua passione e dei tuoi progetti a coloro che non ti conoscono?

MC. Sono il classico vecchio brontolone, purista e pignolo, ed essendo della classe 1977, per me retrogaming (una parola che secondo me sta sempre più perdendo significato) va dall'Atari VCS fino al Dreamcast. Ciò che è seguito, è un mondo che capisco poco e ancor meno mi attira. C'è da dire anche che da quando sono Presidente dell'Associazione, come è giusto che sia in un ambiente sociale, così come io dò un input ai giovani di provare computer e console dei miei tempi, allo stesso modo loro mi spronano a provare le "cose moderne". E anche se la cosa non mi esalta, di sicuro mi aiuta a tenere un'apertura mentale più ampia. Sempre in ambiente associativo negli ultimi tempi ho iniziato ad apprezzare alcuni emulatori (che mi permettono di mostrare e provare velocemente più giochi in ambienti dove spesso non si ha la stessa tranquillità di casa), ma fondamentalmente, e nella sfera privata, per me il videogioco è sempre legato all'hardware reale e originale. Nei miei progetti collettivi offro spesso ai miei collaboratori il consiglio di mettere sempre amore in quel che si fa, di farlo con stile e di rispettare sempre se stessi ed il prossimo. Tutte qualità che ci hanno permesso di arrivare fino a questo punto a testa alta. E poi penso che si tratti di un ottimo consiglio da mettere in pratica anche nella vita di tutti i giorni.

Il logo dell'Associazione RetroEdicola

DLM. Qual è la tua esperienza nella comunicazione e nel giornalismo specializzato?

MC. Più che un giornalista sono un grafico. Discendo da una famiglia di tipografi ed ho studiato DTP (Desktop Publishing) sui primi Mac. Ho assistito al passaggio dall'analogico al digitale e questo spesso ha condizionato la mia scelta delle riviste. Non per niente ho iniziato ad apprezzare e collezionare quelle riviste videoludiche dai primi anni '90, come la gloriosa C+VG, che aveva un'impostazione decisamente moderna per l'epoca, se confrontata con le spartane pubblicazioni precedenti. In ambito giornalistico non professionale ho scritto su diversi blog e siti specializzati per molti anni. In tempi recenti anche per il sito dell'Eco di Bergamo (anche se a causa del mio poco tempo libero si tratta di una collaborazione saltuaria), mentre sulla cara e vecchia carta, fatte salve le mie auto-pubblicazioni (tutte disponibili su RetroEdicola), ho scritto sulla rivista Retrogame, sempre in ambito divulgativo di riviste di videogiochi.

DLM. Sei da sempre un utente "spinto" di computer per fini ludici o professionali? Qual è la stata la molla che inizialmente ti ha condotto sulla strada dell'informatica e qual è stata la tua prima esperienza con un computer?

MC. Come ho accennato prima, ho iniziato a usare i computer per uso professionale, in ambito grafico. Ricordo che iniziammo con un vecchio *mainframe* e l'operazione d'impaginazione si effettuava (scomodamente) usando una grande tavoletta grafica ed una trackball. Il primo sistema che posso dire davvero "mio" è stato invece un Mac Classic, che usavo sia per studio che per lavoro. Ho dei ricordi bellissimi delle ore passate su quella scatoletta ;-).

Sul piano *home* invece è stato mio nonno a mettermi tra le mani un Commodore 64 quando avevo solo 8 anni, spinto dalla sua passione per la fantascienza e per Asimov. Inizialmente lo accendevo solo ogni tanto, ma poi, crescendo, il C64 ha tirato fuori il videogiocatore che è in me. Penso che ancora oggi sia la mia macchina videoludica preferita.

DLM. Quando hai preso in considerazione di occuparti della parte di divulgazione e di comunicazione nel mondo del retrocomputing?

MC. Non credo che per me e per quelli della mia età o poco più vecchi, ci sia un vero e proprio concetto di *retrogaming*. Semplicemente gioco da una vita (letteralmente) e la cosa è diventata abbastanza naturale: conoscendo gente più giovane mi è sempre piaciuto raccontare della mia passione, così come scrivere speciali e articoli monografici. Ultimamente con la nostra associazione siamo stati chiamati per presentare i videogiochi nelle scuole e devo confessare che si tratta di un'esperienza bellissima che ci ha spinto anche a ricostruire come sono nati i primi videogiochi della storia.

DLM. Vuoi ricordare ai nostri lettori quali altri progetti hai curato personalmente in passato e quali di questi sono tuttora in vita con successo?

MC. Il progetto è sempre solo uno, ma come ho detto all'inizio ha mille sfaccettature e progetti interni. Il progetto originale ha cambiato nome da quando lo pensai per la prima volta, ha cambiato forma quando è diventato web, è cresciuto quando ha assunto una dimensione fisica. Ed è ormai un'entità a sé stante, una sorta di filosofia che si modella con la nostra passione e ci definisce come gruppo. Ed è costruita per sopravvivere quando noi non ci saremo. Associazione

Culturale RetroEdicola Videoludica, segnatevela! ;-)

DLM. I tuoi progetti e le tue iniziative denotano una grande creatività e un'attenzione speciale per le persone in modo da riunirle attorno ad una passione comune. Secondo te, da dove derivano queste qualità? Dalle esperienze dirette avute in gioventù? Dagli stimoli (anche di oggi) da parte di amici? Dalla passione per i videogames?

MC. E' una bella domanda. Per me il bello dei videogiochi è proprio il lato *social* che ne deriva. Ogni videogioco giocato in due o più persone è un'esperienza unica: quando ci sediamo di fronte a un videogioco siamo tutti uguali, non ci sono più pregiudizi, bianco o nero, ricco o povero. Ho provato anche a giocare con persone che parlavano solo una lingua straniera che non conosco e ci siamo divertiti in egual misura! Il lato sociale, quello che oggi purtroppo spesso manca (e non intendo a causa dei social network o cose simili), il rapporto fisico con le persone è quello che sto cercando di ricreare nel nostro Club: sembra una banalità, ma viviamo sempre più in una società che ci relega chiusi in casa, interconnessi solo virtualmente. E a me francamente non sta bene.

RetroEdicola è anche socialità

DLM. Qual è stato il tuo sistema preferito fra quelli di un tempo (dagli home computer in avanti) e perché?

MC. Ho avuto la fortuna di aver posseduto tutti i sistemi videoludici che desideravo. Tendenzialmente non ho preferenze a livello hardware, basta che il sistema abbia titoli validi. Se invece devo rispondere con il cuore, allora la musica cambia. Come accennato, il C64 è stato per me molto importante, ho poi giocato molto con Amiga (prima 500 e poi 1200). Un altro capitolo importante della mia carriera videoludica si è consumata sui PC, in particolare con i *LAN party*: prima con il classico Doom e poi molti altri fino a Quake 3 Arena, le sfide in rete sono state una mia fissa. Il piacere di ritrovarsi in luoghi disparati con il PC sottobraccio e montare il tutto alla buona per spararci (virtualmente) è una cosa che per lungo tempo mi è mancata (i giochi moderni non prevedono quasi più questa modalità). Non a caso è stata la prima iniziativa che abbiamo portato nella nostra sede.

Tornando ai miei sistemi preferiti, non posso non citare le console, Super Nintendo e Megadrive in primis, una battaglia che si è consumata a suon di giochi fantastici. Sono stati gli anni più esaltanti per leggere le riviste videoludiche specializzate: ogni mese tantissimi titoli validi tra cui scegliere. Menzione d'onore infine per i sistemi Arcade: ho avuto la fortuna di aver avuto in casa un cabinato fin da giovanissimo. La caccia alle schede Jamma e alle cartucce NeoGeo MVS quando venivano dismesse è una cosa che mi ha sempre divertito molto. Gli anni passano ma un sistema arcade per me è il top se si parla di videogiochi. Le ultime console con cui mi sono confrontato sono state Dreamcast e Xbox. Soprattutto sul DC mi sono davvero divertito, un connubio perfetto tra passato e futuro. Da lì in avanti le console ed i giochi per PC sono diventati (per me) troppo complicati e troppo lunghi da giocare (invecchiando il tempo manca sempre di più). A seguire tutta quelle porcherie dei DLC, dello streaming e via dicendo. Sto recuperando in parte questi giochi, frequentando il Club, ma continuano a non darmi le stesse emozioni dei videogiochi classici.

DLM. RetroEdicola Videoludica è forse il progetto più conosciuto fra quelli che porti avanti e di cui sei ideatore e supervisore. Ma qual è stata la prima rivista per computer che hai acquistato in edicola?

MC. Mi devo grattare un po' la testa per cercare di ricordare. Ho comprato così tante riviste in vita mia che a volte mi confondo: in ogni caso, prima di C+VG (Computer+Videogiochi edita da Jackson), non mi attiravano troppo (a livello di impaginazione). Ho però preso saltuariamente un po' di tutto. Credo che la prima rivista che ho acquistato sia stata Zzap! – di cui conservo un piacevole ricordo.

La sede del Club a Bergamo

DLM. Qual è stato invece il primo gioco che ricordi di aver giocato sul tuo primo computer? E giochi ancora ai vecchi giochi per home computer o preferisci le moderne console e le produzioni "kolossal" di oggi per il tuo tempo libero?

MC. Gioco e ho sempre giocato, soprattutto in compagnia, ai videogiochi classici ed ora che abbiamo una sede che promuove la storia del videogioco, più che mai ho ripreso a videogiocare. Ovviamente riesco a spaziare spesso tra presente e passato, ma i videogiochi moderni non sono fatti per me. Il mio primo videogioco? Domanda difficile. Sicuramente sarà stato qualcosa uscito su cassetta pirata per Commodore 64 :-D. Scherzi a parte, non ricordo proprio quale sia stato il titolo. Dopo un inizio un po' titubante ho macinato e comprato tanti di quei titoli che ricordarsi del primo mi viene difficile.

DLM. Qual è l'iniziativa più coinvolgente, non necessariamente lontana nel tempo, alla quale hai lavorato? E quale quella più intricata e più complicata da far nascere?

MC. L'ultimo step di RetroEdicola! La trasformazione in Associazione Culturale è stata un passaggio che ha richiesto molto lavoro e anni di organizzazione. Premesso che inizialmente non avevo intenzione di fondarne una per conto mio, sia per il numero elevato già presente sul nostro territorio, sia perché essere presidente di una associazione è comunque un compito impegnativo e richiede carisma e forza, ho passato diversi anni a cercare una casa per il progetto altrove. Ma alla fine ho rinunciato perché ci sono molte associazioni che usano questa forma giuridica per fini lucrativi o perché non ne condividevo i valori morali o, ancora, perché non presentavano la qualità e la serietà che io cercavo. So che posso apparire pedante o troppo esigente, ma dopo una vita passata nel

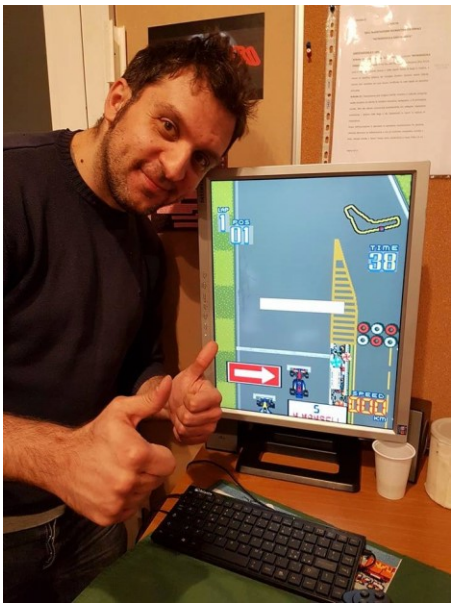
perseguire un obbiettivo, non potevo accontentarmi di niente di meno da quello che mi ero prefisso. Alla fine quando stavo per rinunciare mi sono imbattuto nei ragazzi di RetroGaming Bergamo, che promuovevano serate per locali a tema retrogaming: nonostante la giovane età, la loro passione e l'energia con cui portavano avanti il progetto mi hanno piacevolmente sorpreso. Dopo un paio d'anni passati a sondare la zona mi sono deciso a comprare un locale per fare base e ci siamo uniti per proporre un'offerta completa: noi di RetroEdicola ci occupiamo del lato informatico e storico, loro di tutto quello che riguarda il videogioco e l'organizzazione degli eventi. Siamo aperti da due anni (anniversario il mese prossimo) e ad oggi non potrei essere più che contento della direzione intrapresa. Nonostante le difficoltà, siamo ormai una macchina rodata, che organizza eventi ogni fine settimana e che propone un luogo dove ritrovarsi e giocare tutti i giorni, con quasi 100 soci tesserati paganti e non è poco! Il Club è anche un'officina di idee, un posto dove scambiarsi opinioni guardandosi negli occhi e non fa distinzioni d'età o di vedute. Ogni contributo è utile per la crescita sia dell'associazione sia personale di ogni associato.

DLM. Quanto è importante secondo te preservare la memoria di un tempo e di un mondo come quello degli anni '80 e '90 creatosi attorno all'avvento di console e home computer sul mercato nazionale ed internazionale?

MC. I videogiochi sono parte della storia dell'umanità. E tutte le storie meritano il nostro rispetto. La preservazione della memoria storica non è solo importante, è un dovere di cui noi videogiocatori dobbiamo necessariamente farci carico.

DLM. Quali sono state le difficoltà principali che hai incontrato assieme ai tuoi collaboratori nello sviluppo di progetti complessi e che si sviluppano nel tempo, come RetroEdicola o il Club ad esso collegato?

MC. A parte i problemi tecnici e di tempo individuali, il grande problema sui progetti a lungo termine sono sempre le persone. Suscitare e mantenere interesse nelle persone che partecipano al progetto e riuscire a trasmettere energia a tutti nel tempo non è affatto facile. E' un fatto naturale: si parte sempre con slancio, ma poi per mantenere una costanza nell'impegno non basta la passione, soprattutto sulla lunga distanza. Serve responsabilità e persino sacrificio. Significa mettersi lì a lavorare per il progetto anche quando non se ne ha voglia, anche se questo comporta problemi personali. Dall'altro lato, non è facile trovare collaboratori validi, poi bisogna saperli valorizzare, fare squadra ma anche avere le forza di allontanarli quando non funzionano più. E' un fatto ciclico, ma è l'unica strada percorribile: non si può pensare di mandare avanti un progetto a lungo termine se si pensa di far tutto da soli. Allo stesso modo non è possibile far crescere il progetto se non si trovano nuove menti e soprattutto con idee diverse: questo perché un'eccessiva specializzazione porta secondo me ad una lenta morte del progetto.



Gabriele Ferri, gamer e collaboratore attivo di RetroEdicola

DLM. RetroEdicola Club organizza periodicamente eventi a tema interessantissimi per tutti coloro che sono appassionati (e per i nostalgici) di giochi per console e computer ma non solo. A volte il target si sposta su personaggi o artisti, oppure su fumetti o film o ancora su piattaforme di gioco meno usate fino agli incontri più tecnici in cui si parla di sistemi operativi o di programmazione. Quali sono le scelte che ti portano, con i tuoi collaboratori, in una direzione o nell'altra? Le richieste degli associati, un piano programmato atto a coprire i tanti fenomeni del passato o altro ancora?

MC. Premesso che l'associazione oltre a essere finalizzata alla Cultura Videoludica è anche impegnata alla divulgazione della Cultura Pop (-olare), che spesso per sua natura si interseca con quella dei videogiochi. Gli eventi spaziano così tanto sia per richiesta dei soci (ovviamente), sia per ampliare le conoscenze di tutti e cercare di aprire la mente. Oltre al fatto che siamo videogiocatori, ci piace divertirci in tutto quello che facciamo. Il calendario viene discusso da tutto il direttivo, poi viene proposto ai soci. Non cerchiamo di coprire tutto (cosa impossibile) ma puntiamo piuttosto ad un equilibrio tra studio e gioco e qualche volta semplicemente dobbiamo sottostare alle date che ci propongono gli invitati.

Il modello tipico è: ad inizio mese un torneo, poi invitiamo qualche personaggio (nel panorama del videogioco, del fumetto, del cinema o altro), alcuni giorni dedicati ai LanParty (sia su PC sia su console) e chiudiamo con un'*arcade night* (si gioca sul proiettore a turno mentre il presentatore racconta la storia del videogioco stesso). Il martedì è dedicato al circolo letterario (sì, scriviamo molto e abbiamo anche una fanzine e, come anticipato, siamo anche editori), il mercoledì è il giorno del cineforum (titoli scelti dai soci su percorsi tematici), giovedì facciamo i giochi in scatola (un mix tra passato e proposte moderne) e il venerdì si chiude con un aperitivo e serial TV (sempre tra passato e presente). Direi che non ci annoiamo mai ;-).

DLM. Molti considerano ZZap!, The Games Machine o K, le riviste di riferimento di un'epoca d'oro degli home computer, almeno da un punto di vista dei videogames. Come mai, secondo te il panorama in edicola è radicalmente cambiato da circa 15-20 anni fino a raggiungere un livello francamente desolante negli ultimi anni? Colpa solo di Internet e dei contenuti che propone (YouTube, Facebook e forum) in alternativa per i videogiocatori che vogliono informarsi sulle ultime novità?

MC. Questo è un argomento che da solo richiederebbe un libro intero e si tratta di un problema che riguarda in generale la nostra società, come è cambiata e come è cambiato il nostro modo di vivere. La carta probabilmente non morirà mai, ma sarà sempre più relegata ad approfondimenti e ad uso di chi vuole davvero sapere e studiare a fondo un argomento. Siamo giunti a questo prima di tutto per necessità: le informazioni su carta viaggiano lente, la carta costa molto e richiede tanto spazio. Sono tre fattori che remano contro, perché noi abbiamo sempre più fame di novità (in un mondo videoludico che ne ha sempre di meno). Soldi ne girano sicuramente di meno (è un dato di fatto) e lo spazio soprattutto per noi veterani è un fatto cruciale. D'altra parte assistiamo giorno dopo giorno ad un impoverimento culturale generale: si legge sempre di meno, si guardano più le immagini e le fotografie e ci si affida sempre più al sentito dire e ai falsi profeti. Ecco allora che un post di due righe su Facebook vale più di due pagine di recensione. E i tanti, troppi Youtuber da due soldi diventano dei riferimenti.

DLM. Cosa pensi del mondo del retrocomputing/retrogaming e del relativo successo che questo fenomeno sta vivendo ultimamente, anche in relazione ai diversi *remake* di giochi pubblicati e alla commercializzazione di nuovo hardware come le console ed i computer "mini", emulatori, nuovi accessori e periferiche per i vecchi sistemi?

MC. *Retro-qualcosa* sono tutte parole che, più passa il tempo, più mi danno l'orticaria e perdono sempre più di un qualunque significato. Penso che in prima istanza siamo davanti ad un fenomeno di tendenza. C'è sicuramente una larga fetta di giovani che vogliono tornare indietro e sapere o che, annoiati dal panorama moderno, cercano più semplicità e più giochi validi, tutte cose che solo il nostro passato può darci.

Questo fa sicuramente comodo a chi deve vendere e ha poche idee, anche se i più intelligenti usano le versioni mini di console e computer per far conosce ai giovani l'origine dei loro videogiochi (basti pensare che chi oggi ha 18 anni non ha mai visto neanche la prima Playstation). In generale, però, come avviene per il cinema, mancano le idee innovative e si scava nel passato. Io francamente sono nauseato da queste cose, spesso spacciate a cifre mostruose e realizzate grossolanamente o con dentro squallidi emulatori, quando con pochi soldi si può recuperare un qualunque sistema originale, spesso spendendo meno. Anche in ambiente Club non ne sentiamo la necessità. Qualche volta usiamo l'emulazione, ma sempre in misura limitata. Ed è quello che dovrebbero fare tutti: quando si prende coscienza che questi vecchi sistemi hanno il potere di farci divertire, perché non spendere un po' di tempo per recuperarli e giocare come si deve? Poi, certo, ognuno può fare quello che vuole, ma questo è il mio pensiero.

La t-shirt ufficiale di RetroEdicola

DLM. Se potessi tornare indietro nel tempo, c'è una cosa (o più cose) che cambieresti o faresti in modo diverso in uno dei tuoi progetti?

MC. Sono una persona decisa, non mi piacciono le persone che tentennano. Quando prendo una decisione è quella e non ho mai pensato di cambiare qualcosa di RetroEdicola. Diciamo che se ci fosse una fantomatica macchina del tempo, con il senno di poi la userei per venire a conoscenza di tutte quelle riviste che non sapevo esistessero (soprattutto perché non arrivavano nelle nostre zone) e le comprerei tutte! ;-)

DLM. I formati digitali che rientrano nella categoria e-book (come quelli compatibili con Kindle e altri lettori hardware o software) rappresentano secondo te una reale strada percorribile a livello di business e di mercato per la pubblicazione di riviste specializzate in console/computer games?

MC. Direi proprio di no. Finita la novità, gli stessi e-book si sono drasticamente ridimensionati, relegati ad una piccola fetta di utenti. Ancora meno potrebbe avere successo una rivista che ha molto colore e non ha un'impaginazione regolare. Il bello di sfogliare le riviste è di poter soffermarci su quel che ci piace o ci attira. Non è solo leggere da pagina uno in avanti, da destra a sinistra ma proprio come in maniera *random*. Questo è impossibile su qualunque sistema elettronico, che ci vincola e richiede un sacco di azioni per muoversi persino nella stessa pagina. Manca insomma la visione globale che solo un oggetto reale può fornire. Magari potrebbe funzionare per pubblicazioni autoprodotte o senza esigenze di business, ma in ambito di mercato non sono e non possono essere minimamente prese in considerazione.

DLM. Come hai impostato il lavoro attorno a RetroEdicola.com, un progetto ambizioso ma anche complicato da portare avanti? Come si svolge il processo di recupero, digitalizzazione e pubblicazione di un numero tipo?

MC. Servono molte persone per portare alla luce un documento PDF di qualità RetroEdicola: il recupero per fortuna è la parte più semplice. Avendo passato una vita a svuotare le rimanenze delle edicole e una decina d'anni in giro per il paese a recuperare riviste dai privati, abbiamo tutta l'esperienza per affermare che è molto difficile che qualcosa ci sfugga.

Il secondo step è quello più lungo: la digitalizzazione. Usiamo scanner Epson A3 per portarli in digitale: sono molto costosi e mediamente ne scassiamo uno all'anno, tanto è il lavoro a cui sono sottoposti. Usiamo il formato A3 perché molte riviste non avevano una dimensione standard e un A4 è spesso insufficiente. Ogni rivista viene scansionata a pagina singola, il che vuol dire togliere le spille o scollare le pagine. Diciamo che mediamente il processo va da un'ora a due per ogni rivista. E siamo solo in due a scansionare: io e Gabriele Ferri.

Infine viene la parte di editing, che è molto delicata: serve ripristinare la rivista, ma non farla nuova, deve essere leggermente corretta cromaticamente per adattarsi ai monitor, ma non deve stravolgerla. Il nostro guru è Simone Battaglioni, che utilizza

Photoshop ad altissimo livello e tramite le *action* riesce a tirare fuori una rivista completa in un paio d'ore. In pratica si riesce a pubblicare in digitale una rivista al giorno. Senza considerare tutto il lavoro necessario per il sito web, che per l'indicizzazione e ora anche per il Club (organizzazione, recupero giochi, manutenzione e altro), porta via altre 4 ore giornaliere.

A volte è passione, altre volte è lavoro, altre volte è sacrificio, ma i risultati si vedono e siamo tutti consapevoli che stiamo facendo un grande lavoro, che verrà apprezzato soprattutto dalle generazioni future, che per forza di cose non potranno più contare sugli originali (ormai introvabili o reperibili solo a prezzi stratosferici).

DLM. Cosa pensi del panorama odierno dei videogiochi in generale? Secondo te, qual è la direzione verso la quale sta andando il mercato internazionale videoludico? C'è ormai spazio soltanto per le grandi produzioni o il grande pubblico può ancora essere coinvolto con la creatività e la giocabilità di un titolo sviluppato da produzioni più piccole o addirittura *homebrew*?

MC. Lo chiedi a uno che è troppo vecchio e a cui il mercato non si rivolge più. Oltretutto sono abbastanza chiuso verso le tecnologie del futuro. Vivo in un mondo che capisco sempre di meno e che sempre meno mi piace. Aborro l'idea dello streaming e del digitale, non mi piacciono le produzioni videoludiche faraoniche, ma in generale non mi piace neanche il cinema moderno e la musica che ascolto. Vedo da un lato sempre meno idee accompagnata da sempre più ignoranza generale. Se oggi si va in questa direzione di declino, la "colpa" per me è da imputare al pubblico, che ormai compra e "consuma" ogni schifezza pubblicizzata esaltandola come se si trattasse di un capolavoro. Ovviamente non dico che sia tutto da buttare. Nella povertà generale, qualche novità di tanto in tanto salta fuori, ma, ripeto, non è il tempo che mi si addice. E forse neanche mi interessa, anche perché ho talmente tanti giochi classici che ancora devo provare e altrettanti che ho dimenticato che mi aspettano. Queste "cose moderne" le lascio ai giovani, alla fine è un problema loro. E se a loro sta bene così, chi sono io per dire no?

Conclusioni

Ringraziamo Mauro per aver riflettuto insieme a noi sul momento particolare che stiamo vivendo nel mondo del retrogaming e dell'editoria legata a questo settore. L'energia che Mauro ed i suoi collaboratori e amici dell'Associazione profondono senza sosta da tanti anni a Bergamo, in Lombardia e in tutta Italia tramite le molte iniziative editoriali e quelle promosse sul web, lasciano ben sperare affinché la storia e la cultura del videogioco non vadano perdute.

La cura e la passione, che tutti i collaboratori dell'Associazione RetroEdicola guidati da Mauro mettono nell'organizzazione dei progetti e nelle costanti attività quotidiane, traspaiono soprattutto nella grande qualità dei contenuti digitali pubblicati ad uso di tutti e degli eventi organizzati. A tutto questo l'incessante lavoro dell'Associazione aggiunge una buona dose di principi fermi come quello di tenere in grande considerazione l'aspetto sociale ed umano che la cultura videoludica comporta. La promozione del retrogaming e tutte le attività svolte per preservare la memoria ed il ricordo attivo e vitale del videogioco fanno di RetroEdicola il più ambizioso progetto italiano e sicuramente fra quelli più produttivi in Europa. Auguriamo quindi lunga e prospera vita all'Associazione di cui Mauro è presidente e punta di diamante. Il futuro (ed il passato) del videogame con loro è al sicuro!

Riferimenti

Associazione Culturale RetroEdicola Videoludica
Facebook - https://www.facebook.com/RetroedicolaClub/
Twitter - https://twitter.com/RetroEdicola
Web - http://www.retroedicola.com/

Vintage Computer Festival Italia 2019

Roma 27/28 Aprile presso Complesso Ex Cartiera Latina

a cura del Vintage Computer Club Italia

Promuovere e divulgare la storia dell'informatica personale è l'obiettivo del **Vintage Computer Festival Italia 2019** (www.vintagecomputerclubitalia.it), che si terrà a **Roma il 27 e 28 aprile presso il Complesso Ex Cartiera Latina** (via Appia Antica 42): un evento a ingresso gratuito che, oltre a presentare una grande esposizione di computer storici dalle origini agli anni '90, offre importanti momenti di divulgazione, ospiti internazionali, grandi personaggi della storia del computer ed eventi speciali da non perdere.

Organizzato dall'Associazione Culturale Vintage Computer Club Italia con la partecipazione di Magnetic Media Network Spa, questo festival è un appuntamento imperdibile per tutti coloro che amano il computer e le sue evoluzioni, dagli anni '60 ad oggi: professionisti, studenti, appassionati, famiglie, collezionisti, giornalisti, curiosi e nativi digitali potranno vedere e toccare con mano i pezzi più importanti della storia del personal computer, per la maggior parte ancora funzionanti, e potranno rivivere quegli anni fantastici di scommesse e pionierismo.

Programma 2019: ospiti e appuntamenti da non perdere

La sala conferenze, che conta 150 posti a sedere, ospiterà un ricco programma di interventi, testimonianze e momenti di discussione aperti a tutti.

In questa edizione si parlerà in particolar modo di come il Personal Computer a impattato il mondo della scuola cambiando il modo di insegnare e apprendere.

Si inizierà al sabato mattina con la conferenza di Liza Loop, che è stata pioniera in questo avendo capito già dai primi anni 70 che il PC avrebbe modificato la vita di studenti e insegnanti e si è sempre impegnata in questo settore con consulenze per Apple, Atari, TRS.

Si continuerà con il racconto delle esperienze di Jan-Daniel Nicoud che nel suo laboratorio al Politecnico di Losanna, mise su un gruppo che si rivelò fonte di intuizioni e invenzioni

Tema analogo si affronterà nella tavola rotonda patrocinata da Rai Scuola, nella quale personaggi di spicco della didattica italiana affronteranno questo tema attualizzandolo.

VINTAGE COMPUTER CLUB ITALIA

La domenica sarà invece dedicata a temi più vari iniziando dalla prima conferenza nella quale Silvio Henin di AICA illustrerà il progetto di salvaguardia del patrimonio informatico italiano.

Momento storico per gli appassionati di Gaming sarà invece la conferenza di Allan Alcorn che ripercorrerà le sue esperienze negli anni 70 sia in Atari, dal Pong al VCS, sia come consulente per altre aziende come Apple Computer.

Il pomeriggio si aprirà con una tavola rotonda nella quale alcuni imprenditori dell'epoca e studiosi del fenomeno, illustreranno come nacque e come evolse la produzione di Personal Computer in Italia tra la fine degli anni '70 e gli anni '80.

Chiuderà la giornata, e l'evento, una dimostrazione dell'Alpha Syntauri, sistema musicale basato su Apple IIe e hardware/software Syntauri, di Mauro Sabbione, tastierista storico dei Matia Bazar negli anni '81-84 che, all'avanguardia, segno' proprio con l'elettronica quegli anni del famoso gruppo italiano.

A seguire Sabbione terrà un concerto solo-piano che ripercorrerà con un sincronismo tra musica e video, la musica dei Matia Bazar di quegli anni.

L'area espositiva

L'ampia area espositiva di circa 800 mq, divisa in due grandi sale, conta la presenza di diverse associazioni italiane che si occupano di recuperare, preservare e promuovere conoscenze e materiale informatico. I visitatori potranno vedere e provare tutti i pezzi più importanti che hanno fatto la storia dell'informatica dal 1965 agli inizi del 1990.

La sala Nagasawa conterrà alcune postazioni espositive. Da un lato una veloce escursione della storia della nascita e evoluzione del PC dalla Olivetti P101 ai tre computer che aprirono il mercato cioè Apple II, Commodore PET e TRS 80, dall'altro i computer che hanno contrassegnato la produzione italiana tra il 70 e 80, dalle schede Z80 NE, ai computer basati su processore Z80 e sistema CP/M, agli Olivetti M20 e M24 ai compatibili / cloni di AppleII e IBM.

Nella stessa sala ci sarà una postazione presso la quale verrà esposto un progetto, ancora non completo, per la realizzazione di una replica del computer di bordo delle missioni Apollo (il c.d. DSKY) che portarono l'uomo sulla Luna.

Due postazioni con macchine esposte e installazioni multimediali illustreranno il sistema musicale Alpha Syntauri e lo Smaky, personal computer progettato da Jan-Daniel Nicoud a metà degli anni 70.

Gran parte della sala sarà allestita poi a Gaming Zone con console, videogiochi e arcade degli albori dell'informatica videoludica e dei giochi da sala.

Nella sala Appia si svolgerà invece il raduno/esposizione degli appassionati/collezionisti. Qui i visitatori potranno destreggiarsi tra i computer che hanno segnato gli anni del pionierismo, dai Commodore ai Sinclair, dagli Apple agli Atari e molti altri.

All'evento parteciperà anche BasicNet spa (Kappa®, Robe di Kappa®, K-Way®, Superga®, Briko®, Jesus® Jeans, Sabelt) che, sabato alle ore 12.30, con il suo laboratorio 8-bit Lab, effettuerà, in sala conferenze, l'annuale accensione dell'Apple- 1, primo computer costruito da Steve Jobs e Steve Wozniak, nel 1976. La rarissima motherboard è di proprietà dell'imprenditore e collezionista Marco Boglione.

Il Vintage Computer Festival Italia è l'unica manifestazione fuori dagli USA che gode del patrocinio della Vintage Computer

Federation ed è annoverato tra i Vintage Computer Festival ufficialmente riconosciuti. Gode inoltre del patrocinio di:

- Ministero dell'Istruzione dell'Università e della Ricerca (ancora in definizione)
- Regione Lazio
- Assessorato alla Crescita Culturale del Comune di Roma

Media Partner:

- Rai Scuola
- RTR99
- PC Professionale

Il VCFI è organizzato in collaborazione con MMN (http://mmn.it/)

Sponsor:

- Unidata (www.unidata.it)
- Altemica (http://www.altemica.com)
- AeP (http://www.aep-italia.it)
- S&H (http://www.seh.it)

L'evento in breve
Cosa: Vintage Computer Festival Italia 2019 (ingresso gratuito)
Quando: 27 aprile e 28 aprile
Dove: Roma, Complesso Ex Cartiera Latina (via Appia Antica 42)

Programma completo

Sabato 27 aprile

- Ore 10 cerimonia di apertura presso la sala conferenze.
- Ore 10:15 apertura Gaming Zone con **Allan Alcorn** - Sala Riunioni
- 10:30 **Liza Loop** "*How the computer has changed teaching and learning*" - sala conferenze
- 12.30 Accensione annuale Apple-1, 1976 by 8-bit Lab (Claudio Parmigiani, Gabriele Seleri, Cecilia Botta) di BasicNet Spa
- 14:00 **Jean-Daniel Nicoud** "*Pioneering work on "micros" in the 70's, at the EPF-Lausanne (École polytechnique fédérale de Lausanne)*" - sala conferenze
- 16:00 Digital World (Rai Scuola) Tavola Rotonda "*Come il Personal Computer ha cambiato l'insegnamento e l'apprendimento*": Moderatore **Corrado Giustozzi**
- 19:00 Chiusura

Domenica 28 aprile

- 9:00 Apertura
- 9:30 Silvio Henin (AICA) – sala conferenze
- 10:30 **Allan Alcorn** - sala conferenze
- 12:00 **Corrado Giustozzi** intervista **JD Nicoud**
- 14:30 Tavola rotonda "La produzione italiana a inizio anni 80" moderatore **Maurizio Sorrentino**
- 17:00 **Mauro Sabbione** tastierista dei Matia Bazar '81-'84: "*Albori di musica elettronica con l'Alpha Syntauri*" a seguire **concerto "*Tango a Berlino Parigi Londra*"**
- 19:00 chiusura manifestazione

Incontrali:

- Sabato 10:30 **Allan Alcorn** incontra gli appassionati presso la Gaming Zone
- Domenica 10:00 **JD Nicoud** incontra gli appassionati alla postazione Smaky
- Domenica 16:30 **Allan Alcorn** incontra gli appassionati presso la Gaming Zone

Orari

Esposizione e conferenze

- Sabato 27 aprile apertura ore 10.00 - chiusura ore 19.00
- Domenica 28 aprile apertura ore 9.00 - chiusura ore 19.00

Raduno/esposizione appassionati e collezionisti

- Sabato 27 aprile apertura ore 14.00 - chiusura ore 19.00
- Domenica 28 aprile apertura ore 9.00 - chiusura ore 14.00

Alcune immagini dall'edizione del 2018:

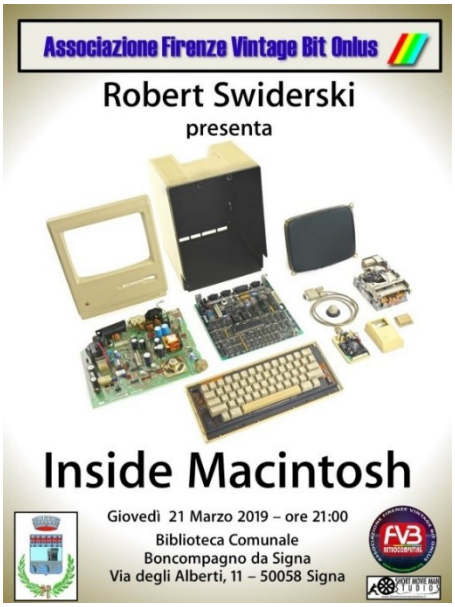
INSIDE MACINTOSH

a cura di Associazione Firenze Vintage Bit Onlus (Robert Swiderski)

L'anno 1984, per molte persone, è stato un anno come tanti altri. Per noi, che seguiamo quasi quotidianamente lo sviluppo dell'informatica e le novità che arrivano, quell'anno è stato veramente un anno rivoluzionario. Nel 1984 entra nelle nostre case un computer, anzi il computer che da ora in poi chiamerò semplicemente il Mac.

Cosa c'è di così speciale che aveva questa nuova macchina?

A questo proposito vorrei presentarvi ciò che ho condiviso con i miei amici, durante l'ultimo incontro, del Firenze Vintage Bit. Il tema dell'incontro era molto accattivante anche se per i poco esperti poteva sembrare noioso: "Inside Macintosh".



Per affrontare questo argomento vogliamo chiederci all'inizio cosa ci interessa in particolare: "Cosa può fare e come" oppure cosa nasconde dentro di sé questa macchina? Penso che tutte e due domande sono di grande interesse almeno per noi.

Nonostante il Macintosh abbia un sistema operativo di oltre 35 anni, l'interfaccia grafica, il mouse, la tastiera e lo schermo lo rendono straordinariamente facile da utilizzare. Un rapido sguardo al pannello di controllo del Macintosh 128K rivela una grafica incredibilmente moderna. Le "metafore" di Susan Kare in forma di una tartaruga o la lepre per indicare le regolazioni della velocità del mouse, le icone del sistema operativo o tanto altro sono la dimostrazione che siamo entrati in un'altra epoca, quella che io personalmente definisco un'epoca del futuro.

Vi ricordo che siamo nel 1984. Prima di quest'anno per comunicare col computer dovevi imparare ad usare tanti termini, procedure come i comandi di un sistema operativo. Da ora in poi non è l'uomo che deve farsi capire ma il computer che deve cercare di capire ciò che gli dice l'uomo. E questo avviene con il Mac. Alcuni potrebbero, in questo momento, pensare al Xerox, alla Lisa ecc. È vero, ma Il Mac, ce l'ha fatto gustare e entrando nelle nostre case a rivoluzionato il modo di usare il computer.

Rilasciato nel gennaio 1984 semplicemente come Apple Macintosh, dopo l'uscita del Mac 512K nel mese di settembre sempre dello stesso anno, il Macintosh viene chiamato Macintosh 128K. Attenti, perché il Macintosh 128 non è la stessa Macchina che viene presentata a Gennaio. Dal punto di vista tecnico erano due computer diversi. La scheda madre del 128 viene completamente ri - progettata per poter, al suo interno, accogliere sia 128KB, sia quella 512KB. Questo ci fa capire che, prima di Novembre 1984, le macchine originali portano il nome solamente "Macintosh".

Come è nata questa macchina?

Se hai tanti soldi puoi costruire tutto. In una rivista si legge: "Gli ingegneri hardware amano dire che qualsiasi cosa può essere progettata se si è disposti a pagare il prezzo di componenti ad alto costo e di enormi alimentatori". Nel nostro caso nessuno credeva che fosse possibile montare un computer su due piccoli circuiti inseriti in una leggera, compatta custodia. Dietro a questo "miracolo" sta un uomo geniale di nome Burrell Smith. Aveva ridisegnato quattro volte la scheda digitale fino a ottenere quella che tutti conoscono: il Mac del 1984.

Burrell Smith

La prima è stata progettata nel 1979 sulle indicazioni di Jef Raskin. All'interno si trovava il microprocessore Motorola 8909E, 64K di memoria e un display in bianco e nero di 256x256 pixel. Jef Raskin non voleva superare il costo di 500 dollari per il suo gioiello. Purtroppo il microprocessore 6809 era limitato dal 16 bit ciò costituiva un grande suo limite.

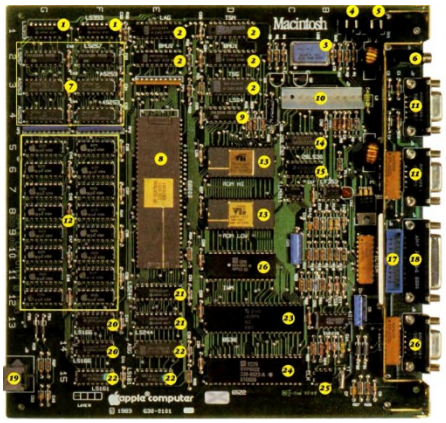
La seconda nasce nel 1980 grazie a una nuova idea di adattare 68000 a un bus di memoria a 8 bit, applicando un circuito di "BUS TRANSFORMER" costruito con chip PAL. Così è nato nel 1980 il nuovo Macintosh basato su 68000, 8 MHZ, 64KB di RAM e un display a 384x256 bit. Era il 60% più veloce della Lisa.

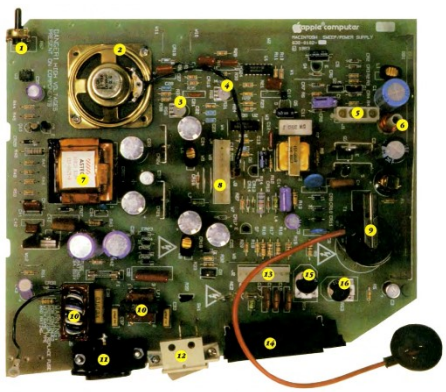
Il terzo progetto della scheda digitale è del 1981. Un nuovo chip chiamato SCC che supportava una rete locale integrata,

rendendo AppleTalk possibile senza hardware aggiuntivo, oltre a fornire porte seriali, spinse Burrell a riprogettare tutto, addirittura aggiungendo una seconda fila di chip RAM, raddoppiando la memoria, per arrivare a 128 KB di memoria necessari.

La quarta versione, siamo nel 1982, viene fuori quando Burrell si rende conto che il software, progettato nel frattempo, non sarebbe mai entrato in 64 KB di memoria. Era necessario un bus di 16 bit. La memoria aggiuntiva gli consentiva di raddoppiare la risoluzione dello schermo (512x 342 pixel invece di 384 x 256).

L'ultima versione, quella che venne presentata nel 1984, era basata sul suo progetto precedente sfruttando chip PAL, un Bus di memoria a 16 bit, un display 512 x 342 pixel, alcune funzionalità inventate da Lui in collaborazione con VLSI Design. Lo Zilog 8530 SCC (serial Communications controller) (controller di comunicazioni seriali), forniva la trasmissione dei dati sincrona e asincrona fino a 230,4 K bit al secondo utilizzando un formato di dati auto-clock e fino a 1 megabit al secondo utilizzando un orologio esterno.





Cosa fa di questo computer, il Computer di seconda generazione?

La semplicità del Mac dipende dal suo basso numero di chip: contiene circa 50 IC (circuiti integrati), che ne riducono le dimensioni fisiche e il prezzo e ne aumentano l'affidabilità. Mac riduce il numero dei suoi chip combinando le funzioni di molti chip standard con otto matrici di logica programmabile (PAL). Funziona ad una velocità di clock più elevata, 7,83 MHz rispetto ai 5 MHz di Lisa per non parlare di IBM 4,7 MHz con microprocessore 8088. Usa la sua memoria in modo più efficiente perché i suoi programmi e subroutine sono codificati in linguaggio assembly del 68000. Ha eliminato le schede aggiuntive e utilizza un bus seriale ad alta velocità che implementa ciò che Apple chiama "slot virtuali" "virtual slots". Gli slot virtuali hanno diversi vantaggi rispetto agli slot per periferiche hardware convenzionali. Riducono i potenziali problemi inerenti a qualsiasi connessione meccanica aggiunta (un connettore di interfaccia seriale ha meno pin di una tipica scheda di interfaccia). Riducono l'RFI (interferenze in radiofrequenza) (radio-frequency interference).

Purtroppo, ma è stata una scelta inevitabile, consente di attivare solo un programma alla volta per far sì che una singola applicazione potesse utilizzare più finestre. Questo non era male come sembra perché permetteva tagliare e incollare il materiale da un documento all'altro prima di caricare il secondo documento che doveva sostituire il primo. Oggi non ci rendiamo conto che questo era un passo nel futuro.

Uno sguardo ravvicinato all'hardware del Mac dimostra che Il Mac è il computer più sofisticato mai offerto nella sua fascia di prezzo, di quel epoca, più potente e più veloce di molte macchine di allora.

Descrizione

L'unità principale del Macintosh è composta da otto parti in più un esterno mouse e tastiera per un totale di 10 parti.

All'interno di Macintosh, hardware e software lavorano insieme per fornire un sistema in grado di supportare la grafica ad alte prestazioni, le periferiche integrate e i canali di comunicazione verso il mondo esterno.

Ad esempio, i pixel visualizzati sul display del Mac sono quadrati (non rettangolari, come negli altri computer); questo semplifica enormemente il software che disegna piazze e cerchi, ridimensiona il testo e la grafica e la stampa delle immagini sullo schermo.

Il cuore dell'elettronica digitale Macintosh è il processore MC68000 e la sua memoria (sia RAM che ROM). Un chip che elabora da uno a due milioni di istruzioni ogni secondo.

La memoria ROM si collega direttamente al bus dati del sistema e viene utilizzata solo dal 68000. Gran parte del codice time-critical del sistema, come i programmi grafici di basso livello, le routine del sistema operativo e le routine dell'interfaccia utente, risiedono nella ROM "The User Interface Toolbox". Poiché i bus dati e indirizzi ROM sono utilizzati esclusivamente dal 68000, la ROM è sempre accessibile alla massima velocità del processore di 7,83 MHz; Il comportamento delle routine ROM è completamente determinato dal contenuto del disco inserito in esso, ovvero i progettisti di software possono utilizzare le routine ROM per creare un'applicazione Macintosh "standard", oppure possono scrivere il proprio codice per creare un'applicazione che si comporta come preferiscono.

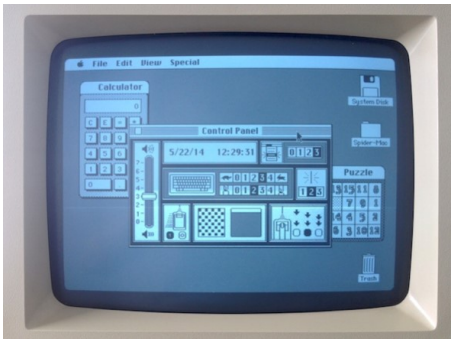
Il Mac ha 128 KB di memoria; La RAM del Mac è composta da due banchi da 8 chip ciascuno: 128K di RAM su 16 chip. La memoria non poteva essere ampliata, ma i programmi potevano usare il codice ROM.

Il mouse, la tastiera e l'orologio in tempo reale di Macintosh veniva gestito da un altro chip:

l'adattatore di interfaccia versatile 6522 (versatile interface adapter).

Il controller del disco Macintosh è un singolo componente LSI (large-scale integration)(integrazione su larga scala) denominato chip IWM ("macchina integrata Woz"). Il dispositivo, un'integrazione a un chip del controller del disco originariamente progettato da Steve Wozniak per Apple II, gestisce i dati a 500 K bit al secondo. Per controllare la velocità del motore del disco, un modulatore della larghezza dell'impulso posizionato sulla scheda digitale consente al disco di muoversi a una delle possibili velocità del motore del disco. Questo cambiamento consente al Mac di registrare sul disco con una densità lineare costante (il che significa che è possibile inserire più dati sulle tracce più esterne), in contrapposizione all'approccio basato sulla densità radiale costante utilizzato dalla maggior parte dei commutatori (che mette la stessa quantità di dati su ogni traccia indipendentemente dalla posizione). Per comprendere questa novità basta pensare ai floppy disk della HP (prodotti anch'essi dalla Sony) che memorizzavano su 70 tracce di dati per 135 tpi (tracce per pollice) ottenendo in risultato 270 Kb invece di 80 tracce per 135 tipi e quindi 400 KB. La maggior parte dei computer utilizzava un chip controller del disco invece del processore per controllare l'unità. Il Mac (come Apple II) usava il suo processore per controllare direttamente l'unità che ruotava sotto il controllo del software tra 390 e 600 rpm (giri al minuto) e trasferiva i dati alla velocità di 489,6 K bit al secondo. L'unità può memorizzare 400K byte su un disco da 3,5 pollici a lato singolo anche se il Mac è progettato per essere in grado di utilizzare unità a doppia faccia per ottenere 800K byte per disco.

Il Mac ha un monitor da 9 pollici che visualizza un'immagine a 60.15 Hz. La risoluzione dell'immagine video è 80 pixel per pollice, quindi lo schermo complessivo è 512 per 342 pixel. Il display del Mac ha solo una modalità: grafica. I caratteri sono gestiti dall'unità Quickdraw. È un pacchetto grafico che è il cuore del Macintosh. Bill Atkinson, il suo creatore, l'ha riscritto molte volte e ridotto da un programma Pascal da 160K byte a un pacchetto da 24K byte di codice 68000 altamente ottimizzato. Quickdraw è molto veloce, ad esempio può stampare sullo schermo più di 7000 caratteri al secondo. La velocità di visualizzazione video è di '60,15 Hz, come è scritto sopra, ed è generata internamente invece di essere derivata dalla corrente di linea. Ciò consente al Mac di essere utilizzato senza modifiche in paesi con corrente di linea a 50 Hz.



... e per concludere.

Il Mac è una macchina rivoluzionaria, costruita bene e solida, oggi lenta e limitata, ma quello che doveva fare lo fa bene, nonostante 35 anni già passati, anche nei tempi nostri.

La nostra serata si è conclusa con la presentazione di alcune caratteristiche di questo "gioiello" toccandolo, guardandolo dentro "Inside" e rivivendo i vecchi tempi.

Bibliografia

Revolution in The Valley: The Insanely Great Story of How the Mac Was Made

Varie riviste degli anni '8': Applicando i primi numeri, Apple Soft, MCmicrocomputer n.28, ecc...

L'**Associazione Firenze Vintage Bit Onlus** vi ricorda i prossimi due appuntamenti:

Vi aspettiamo **Giovedi 18 Aprile** con "**MALEDETTI NASTRI**" di **Federico Gori**...

E **Giovedi 23 Maggio** con "**Commodore CDTV**" di **Luca Cusani**:

RETROEVENTI APRILE-MAGGIO 2019

RETROEVENTI

eventi aprile-maggio 2019

PISTOIA NON SOLO COMICS

A Pistoia dal 27 al 28 Aprile

VICOMIX

A Vicenza dal 27 al 28 Aprile

COINUP

A Milano il 26 Maggio

Da questo numero di retromagazine inauguriamo una nuova rubrica dedicata agli eventi legati al retrocomputing e ai retrogames che vengono proposti da organizzazioni e associazioni varie.
Siamo certi che questa rubrica sarà molto gradita e utile per i nostri lettori e sicuramente darà a noi di retromagazine la possibilità di entrare in contatto e di condividere la nostra passione con importanti realtà che operano sul territorio.

Il primo evento che vi vogliamo segnalare è il **Pistoia non solo comics & videogames** giunto alla sua quarta edizione e diventato ormai un appuntamento importante per gli appassionati.
La manifestazione si svolgerà il 27 e 28 Aprile a Pistoia in via Sandro Pertini nel centro fiere "La Cattedrale".
Fulcro e cuore pulsante della fiera sarà il padiglione *comics*, guarnito dalla presenza di autori e fumettisti che hanno dato vita a leggende come Diabolik e agli eroi Marvel, inoltre presenti scuole di fumetto, workshop esclusivi e spazio per provare i migliori giochi a tavolo e tornei di carte.
Puntuale anche quest'anno l'affermata gara *cosplay* che avrà un intrattenimento musicale live corredata da un raduno a tema One Piece.
Ma soprattutto vi consigliamo di visitare l'importante padiglione *videogames* con oltre 200 postazioni gratuite, area Nintendo con tornei SmashBros, realtà virtuale, simulatori di guida e tornei di Fifa e Street Fighter!
Poi per tutti gli appassionati di Fortnite tantissime postazioni gratuite per sfidarvi al gioco del momento, tornei a premi a cura di Gamers Arena e Meet&Greet con gli CaptainBlazer.
Ma il cuore pulsante di questo padiglione sarà l'area retrogames, con una vera e propria sala giochi anni '80, con 100 postazioni per rivivere le emozioni delle sfide a Street Fighters, PacMan, Metal Slug e molti altri!
Per l'occasione saranno disponibili anche cabinati speciali a tema per un modo tutto nuovo di giocare ai vecchi classici. Più di 50 postazioni interattive per rivivere la storia del videogioco dalle primissime console Pong fino agli anni 2000.
Infine sarà presente anche un' area tematica Starwars dove si verrà trasportati in una galassia lontana, mentre sul palco principale si alterneranno poi esibizioni di danza, JapanCooking show, spettacoli di magia e molto altro.

Sempre il 27 e 28 Aprile ma a Vicenza si svolgerà il **Vicomix, Fiera del Fumetto**. Anche questa manifestazione è ormai una tappa fissa e molto attesa dai fan ed è giunta addirittura alla sua quinta edizione.
Oltre alle aree dedicate ai *comix* e al *cosplay*, al vicomix ci si potrà tuffare in una incredibile area *retrogames* mai vista prima con i cabinati e flipper originali della sala giochi marchiati Nintendo, Bally, Atari, Gottlieb, Taito, Capcom.
inoltre ci si potrà sfidare sulle console originali come Atari, Nintendo NES, Master System, MegaDrive, Sega Saturn, Sony PSOne, Super Nintendo, Nintendo 64, fino alle Playstation 2 e Xbox.
Ecco anche alcuni titoli che potrete giocare gratuitamente: Moon Patrol, Pac Man, Donkey Kong, Double Dragon, Final Fight, Tetris, Galaga, Track & Field, Tron, Space Ace, Super Sprint, Gauntlet, Hard Driving, Popeye, Mario Bros, Q Bert, Metal Slug, Super Mario, Duck Hunt, Sega Rally, X-Men Vs. Street Figher, Mortal Kombat, Mario Kart, Operation Wolf e tanti altri capolavori che hanno fatto la storia dei videogames compreso il mitico Dragon's Lair che per la prima volta potrà essere giocato dal pubblico sul megaschermo con il videogioco del 1983, basato su filmati a cartone animato realizzati da Don Bluth Animation che sbancò le sale giochi e portò al successo la categoria dei giochi su laser disc!

Il retrogames a Vicomix Fiera del Fumetto a Vicenza è più vivo che mai! E ricordate, non serve il gettone per giocare!

L'ultimo evento che vi segnaliamo, ma solo per ordine cronologico, è il **CoinUp** che si terrà a Sesto San Giovanni Milano il 26 Maggio presso lo spazio Mil, in via Luigi Granelli 1.
CoinUp Italia è davvero unico perché è la prima mostra mercato italiana dedicata al mondo del retrogaming, board games e videogiochi usati, organizzato da appassionati del settore stanchi di vedere i propri vecchi giochi svenduti e svalutati. L'intenzione è diventare un punto di riferimento per tutte quelle persone private e non solo che cercano un evento per comprare, vendere e scambiare i propri retrogames, board games e videogiochi usati.
L'evento inizierà alle ore 12:00 e si chiuderà alle ore 17:00. Il biglietto da acquistare all'entrata è di soli 3,00€. I minori di 12 anni entrano gratis se accompagnati da un adulto con biglietto.
Vuoi battere la folla? Puoi approfittare degli ingressi VIP ad accesso anticipato a numero chiuso, che ti permettono di entrare in esclusiva dalle ore 11:00.
L'evento si svolgerà completamente all'interno.
Coinup sta diventando una bella realtà itinerante che oltre alla prossima tappa di Milano, vanta già la presenza a Cerea e diverse volte a Roma.

Oltre a questi grandi eventi, in Italia esistono numerose realtà indipendenti che si occupano con tanta passione di retrogaming e retrocomputing. Associazioni aperte al pubblico e organizzatrici di eventi specifici.

La passione per console e computer di un tempo viene coltivata un po' ovunque lungo lo stivale ed il compito di questa rubrica sarà, con calma, raccogliere tutte le realtà che vorranno farne parte. Eccovi il primo elenco delle associazioni che si sono interfacciate con noi ancor prima della nascita di questa rubrica:

LOMBARDIA

RetroEdicola Videoludica Club, via Gabriele Rosa 18c, Bergamo (BG)
http://www.retroedicola.com/index.php?pid=eventi
Aperto dal lunedì al venerdì dopo le 20 e nei weekend secondo calendario. Contattare prima di passare e per richieste di apertura pomeridiane chiamare il 371.1523053 o 348.496.2652

FRIULI VENEZIA GIULIA

Naonian Retrogaming Society, Via Generale Antonio Cantore, 49, 33170 Pordenone (PN)
https://naonianretrogamingsociety.wordpress.com/
Aperti tutti i venerdì e tutti i lunedì dalle 20.30 alle 24:00. Sono previste aperture straordinarie che verranno comunicate tramite la pagina Facebook.
Per contatti: pnretrogaming@gmail.com

TOSCANA

Firenze Vintage Bit, Via della croce 62, San Mauro a Signa (FI)
www.retrocompute.org
Prossimo evento: "Commodore CDTV" - Giovedì 23 Maggio 2019
Per contatti: info@retrocomputer.org

MARCHE

Bitelloni Club, via Mercato 12,Gradara (PU)
www.ibitelloni.com
Aperto Giovedì 18 Aprile, Giovedì 9 Maggio e Venerdì 24 Maggio, poi chiusura estiva.
Per contatti: gdr80@hotmail.it , 3277036134

PUGLIA

Apuleia Retrocomputing, Via Capaldi n.60 – 70125 Bari (Ba)
www.apuliaretrocomputing.it/
Per notizie su giorni ed orari di apertura telefonare allo 0808807143

rubrica a cura di Querino Ialongo e Starfox organizzatori degli eventi segnalati.

Attenzione

Quello che vedete è un esempio di quello che sarà la rubrica in futuro dopo aver ricevuto tutte le vostre segnalazioni.
Le realtà incluse, infatti, sono gestite da amici con cui eravamo in contatto già da tempo. Dal prossimo numero anche la vostra associazione sarà inclusa se ci contatterete. Non ci sono costi di alcun tipo e speriamo anzi di fare un servizio utile a tutti gli appassionati d'Italia.

Easter Eggs…

di Francesco Fiorentini

Come si evince dai miei profili FB e Linkedin, attualmente vivo e lavoro in Olanda dove, come in altri paesi nordici esiste da sempre una tradizione riguardante la decorazione delle uova di Pasqua e la loro la ricerca una volta nascoste. Quasi tutti qui onorano la tradizione ed anche la mia azienda non e' voluta essere da meno organizzando proprio in questi giorni una ricerca delle uova con tanto di premio cioccolatoso finale (che tra l'altro insieme ad una collega ho vinto…).

Incuriosito dalla tradizione ho provato quindi a documentarmi per comprenderne la nascita, chiedendo anche a colleghi ed amici, ma il vero motivo della ricerca delle uova si perde nella notte dei tempi. Sono quindi ricorso al fido Google, ma devo dire che anche qui la ricerca ha prodotto si' diversi risultati, senza pero' concordare su quale sia la reale motivazione.

Un collega, mi ha suggerito una motivazione religiosa: anticamente per la Quaresima c'era divieto non solo di mangiare carne, ma anche i derivati della carne, e pertanto uova, latte, latticini, formaggi… Per questo motivo, essendo le uova un cibo povero e facilmente reperibile nelle case dei contadini, c'era la necessita' di nasconderle in modo da 'aiutare' tutti i componenti della famiglia a rispettare la penitenza quaresimale.

Google mi ha invece suggerito una storiella meno pratica, ma non per questo meno affascinante. Sembra infatti che il dispettoso coniglio pasquale, una volta impossessatosi delle uova di pasqua, le abbia nascoste dentro la campana di un campanile. Quando la campana e' stata suonata, le uova sono state sparpagliate in tutto il villaggio e quindi da qui la necessita' di cercarle per riunirle tutte…

A questo punto appare chiaro il motivo che ha suggerito il nome Easter Egg per identificare i messaggi o i contenuti, di solito di natura bizzarra, nascosti all'interno di un software. La motivazione che ha spinto la creazione di quello che viene considerato il primo Easter Egg e' invece ben nota. Si racconta che negli anni '70 all'Atari i programmatori fossero super controllati e fosse loro impedito anche firmare i propri lavori. Per questo motivo, Warren Robinett ha nascosto nel gioco Adventure del 1979 le proprie iniziali in una stanza segreta, dando cosi' vita al primo Easter Egg di cui si ha conoscenza.

RetroMagazine non e' voluta essere da meno ed ha nascosto all'interno di questo numero un Easter Egg. Trovarlo non vi sara' difficile, ma il primo che ci inviera' una mail all'indirizzo retromagazine.redazione@gmail.com con la soluzione corretta, si aggiudichera' un simpatico gadget di RetroMagazine.

In bocca al coniglio pasquale!

Ringraziamenti

Chiudo, come mio solito, con i **ringraziamenti** a tutti **i gruppi Facebook**, ai siti **OldGamesItalia** ed **IlVideoGioco.com** che ci aiutano a condividere la rivista ad ogni uscita e con un ringraziamento particolare a **Vincenzo Scarpa** che sta riservando nel suo ottimo sito **EmuWiki** uno spazio dedicato a RetroMagazine.

Arrivederci al prossimo numero!

Disclaimer

RetroMagazine
Anno 3 - Numero 14

Direttore Responsabile
Francesco Fiorentini

Vice Direttore
Marco Pistorio

Aprile 2019