RetroMagazine

semplicemente retro

Numero 22 - Anno 4 - Aprile 2020 - WWW.RETROMAGAZINE.NET - Pubblicazione gratuita

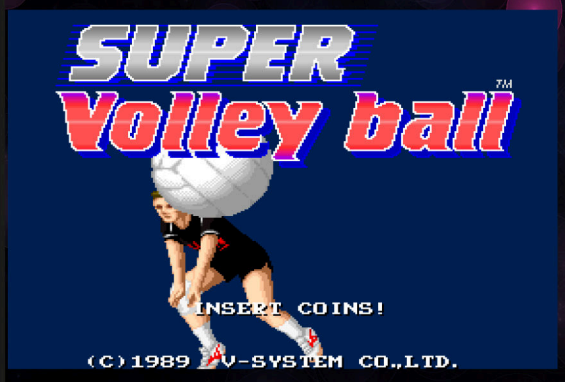
Intervista a Toru Iwatani

Ritorno al... passato

Windows 98

The Legend of Zelda (SNES)

THE STORY OF THOR (Megadrive)



SUPER VOLLEY (Arcade)

NVRCA - Non Visual Retro Computer Access

Introduzione ad ARexx – seconda parte

Grafici in 2D con tre righe di BASIC

LM80C Color Computer

Un computer autocostruito nel 2019 basato sullo Z80 - parte 2

Giappone 10ª puntata: Mrs Game & Watch

Nintendo Gamecube

Storia di un genio incompreso

Ma quanto siamo fortunati?

Probabilmente parlare di fortuna nel bel mezzo di una pandemia puo' suonare strano, ma noi vogliamo essere positivi o almeno provare a vivere la nostra passione come abbiamo sempre fatto.

Ne parlavo proprio una di queste sere in chat con Marco Pistorio, ma quanto siamo fortunati noi appassionati di retrocomputer e retrogamer? Quanti appassionati o collezionisti di altro materiale hanno la nostra stessa fortuna di veder realizzati i sogni che coltivavano da bambini/ragazzi?

Pensateci bene... Probabilmente il sogno di molti di noi quando eravamo adolescenti era quello di possedere una sala giochi in casa. Ebbene, gia' da diverso tempo e' un sogno facilmente realizzabile tramite il MAME.

Quanti di noi da piccoli avevano un computer e magari sognavano di possedere anche un altro modello? Ebbene anche questo oggigiorno e' facilmente realizzabile acquistando computer in mercatini reali o virtuali. E' vero che alcune di queste macchine hanno raggiunto prezzi relativamente alti, ma si tratta sempre di cifre che chiunque piu' o meno puo' permettersi.
Manca lo spazio in casa? La moglie/ragazza/convivente si lamenta per i troppi ferrivecchi in casa? Via di emulatori...
Con gli emulatori e' possibile ormai avere a portata di mano qualsiasi computer/console che il mercato abbia prodotto negli ultimi 40 anni. Parlo per esperienza personale; non ho un Amstrad CPC ma ho sempre sognato di possederlo e di smanettare con il suo parco software. Con l'emulatore posso provare tutto il software che voglio. E' vero, l'hardware originale è tutt'altra cosa, ma almeno posso placare la mia voglia in attesa di trovare l'occasione giusta.

Quanti di noi sbavavano di fronte alle copertine di videogiochi o alle mille cassettine che vedevamo nelle edicole? Ovviamente potevamo comprarne solo una parte e quindi rimaneva l'amaro in bocca per quello che era rimasto sullo scaffale. Adesso molti giochi sono abandoware e possono essere scaricati e giocati da tutti; per non parlare dei giochi da edicola che possono essere facilmente scaricati dal sito Edicola 8 bit.

Beh, convenite con me che siamo fortunati o no? Pensate se la nostra passione fossero state le auto d'epoca... Bellissime per l'amor di Dio ma decisamente inarrivabili in termini economici e di spazio richiesto... Cosa ne pensate? Vi sentite fortunati o no? E perche'? Fatecelo sapere scrivendo alla nostra mailbox redazionale.

Per chi fosse interessato/a a collaborare con la Redazione, può scrivere all'indirizzo: retromagazine.redazione@gmail.com. E adesso andate a leggervi la riflessione di Marco in quarta di copertina e godetevi questo numero almeno quanto noi ci siamo divertiti a crearlo!

Francesco Fiorentini

Sommario

Hanno collaborato nella realizzazione di questo numero di Retromagazine:

- Alberto Apostolo
- Alessandro Albano
- Andràs Vajda
- Carlo Nithaiah Del Mar Pirazzini
- Daniele Brahimi
- David La Monaca (Cercamon)
- Edoardo Ullo
- Francesco Fiorentini
- Gianluca Girelli
- Leonardo Milani
- Marco Fiaschi
- Marco Pistorio
- Michele Ugolini
- Starfox Mulder
- **Supporto grafico: Irene G. Valeri**
- **Copertina a cura di Flavio Soldani**

Nintendo Gamecube – Storia di un genio incompreso

di Carlo Nithaiah Del Mar Pirazzini

Figura 1 - Gamecube e joypad all'opera

Sono passati 19 anni dalla sua uscita sul mercato, ma il Gamecube (quarta console di casa Nintendo) rappresenta il migliore esempio di fare videogiochi della casa di Osaka. Usci in un momento di transito, nel momento in cui Sony presentava la sua popolare Playstation2 e Microsoft si buttava sul mercato ludico con la prima XBOX, il “cubetto” della grande N, dal desing incredibilmente compatto e “d’avanguardi” con la sua maniglia per trasportalo e i mini dischi come supporto ottico, aveva dalla sua dei capolavori semplicemente inarrivabili.
Rappresento anche una sorta di svolta assoluta. Fu la prima console di Nintendo ad utilizzare dischi ottici come supporto di memorizzazione principale. I suoi dischi, in formato simil-mini DVD, lo rendevano un sistema non progettato per riprodurre DVD di dimensioni standard o CD AUDIO (a differenza delle sue concorrenti contemporanee) portando il sistema a specializzarsi esclusivamente sui videogiochi. Era presente anche un supporto per gioco in rete, purtroppo in modo limitato a causa dei pochi (ma eccellenti titoli) tramite una scheda di rete a connessione a banda larga o con un adattatore per modem e poteva connettersi attraverso le 4 porte per i PAD con la contemporanea console portatile Game Boy Advance, tramite cavo apposito. Questo tipo di collegamento consentiva ai giocatori di acceder a funzionalità di gioco esclusive utilizzando appunto il portatile come secondo schermo e controller (alcuni titoli come Final Fantasy Christal Chronicles o Zelda Four Sword ne fanno uso eccellente).

Figura 2 - Luigi acchiappafantasi in Luigi's Mansion

Gamecube supportava il formato video composito per visualizzare i giochi sullo schermo; tuttavia, vi furono alcune differenze nei due modelli della console: i modelli prodotti nel periodo 2001–2003 avevano anche la possibilità di utilizzare cavi video a componenti, modalità in scansione progressiva e una seconda porta seriale; inoltre la targhetta sulla parte superiore della console con la scritta "Nintendo Gamecube" poteva essere rimossa. Questo modello divenne noto come "DOL–001".
Le particolarità appena menzionate vennero rimosse nel modello prodotto nel periodo 2004–2007; noto come "DOL–101". Nella versione più recente venne introdotto un firmware che disabilitò i trucchi Action Replay e il laser di lettura del disco venne migliorato. Questo stesso modello venne fornito con un'alimentatore elettrico da 48 watt per alimentare la console, mentre l'originale era tarato su una potenza di 46.

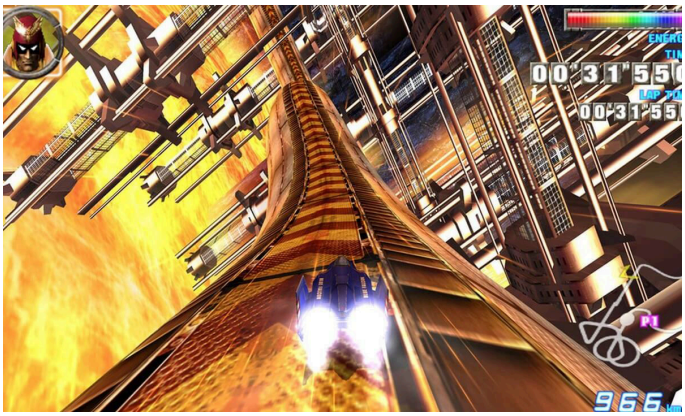


Figura 3 - FZERO GX

L’accoglienza della critica fu generalmente positiva. La console fu elogiata per il suo incredibile controller, per la vastissima libreria di software e i videogiochi di alta qualità, ma venne criticata per la sua strana estetica e per la mancanza di alcune funzionalità (visione dvd o cd musicali). Si piazzo con 21,74 milioni di unità vendute nel mondo prima che andasse fuori produzione nel 2007. Il suo successore, Wii, permetteva una retrocompatibilità con il software Gamecube fin dalla sua nascita.
Sul fronte del software sono stati pubblicati quasi 600 videogiochi. Tra i titoli di lancio della console figuravano Luigi’s Mansion, Super Smash Bros Melee e Pikmin. Tre titoli impressionanti sia per la loro realizzazione che per il loro gameplay. La console ha inoltre visto la partecipazione di numerose case di terze parti come Activision, Konami, Namco, Sega e Capcom con diverse esclusive di queste case per produrre giochi in esclusiva per la console.

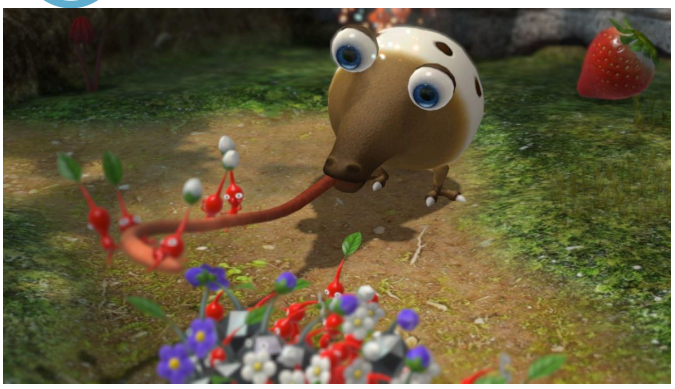


Figura 4 - Pikmin

Impossibile non citare titoli come Animal Crossing, la serie Metroid in formato fpg in prima persona, Super Mario Sunshine, Legend of Zelda: The Wind Waker, Mario Kart Double Dash, Paper Mario, la serie Mario Party e quella di Pokemon.

Tra i titoli di proprietà intellettuale Nintendo ma realizzati da terze parti non si possono non citare Star Fox: Assault che da sparatutto spaziale diventava un adventure game, oppure Donkey Konga che permetteva di usare due speciali controller a forma di tamburi per giocare.

Come non citare anche lo spettacolare F ZERO GX creato dal team Nintedo + Sega. Uno dei più bei giochi di guida futuristici anche ai giorni nostri.
Tra i titoli più interessanti di terze parti in esclusiva non possiamo non citare la serie Resident Evil con le due esclusive Resident Evil Zero e Resident Evil 4 di Capcom oppure l'incredile Metal Gear Solid: The Twin Snakes di Konami.

Ma allora cosa accadde? Perché non ebbe il successo sperato? La console poteva vantare un apparato a dir poco prodigioso e di gran lunga superiore alle due concorrenti Sony e Microsoft, eppure il bilancio finale parlerà di poco più di 20 milioni di unità e di una marea di rimpianti per Nintendo, incapace di spingere la sua console e capire per quale motivo non fosse mai scattata la scintilla col pubblico. Pubblicità sbagliata? Line up iniziale debole? Un Mario poco protagonista? Le cause potrebbero essere tante.

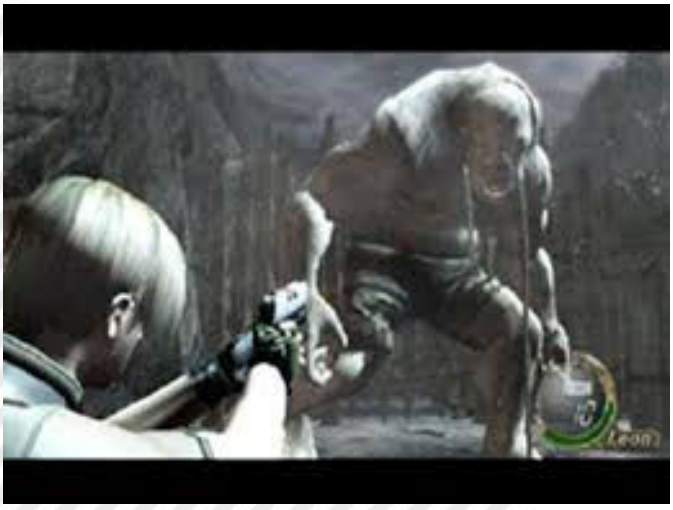


Figura 5 - Resident Evil 4

Ma il Gamecube permise a Nintendo di capire una lezione molto importante, una filosofia che anche oggi viene seguita: i competitor principali nel campo del progresso tecnologico primario e delle offerte per così dire mainstream ormai erano Sony e Microsoft e non era più la vecchia console war anni 90.

Occorse un notevole cambio di rotta e servì soprattutto offrire al pubblico qualcosa che non poteva essere assolutamente trovato con le altre console. La risposta, appunto, sarà Wii. La console per tutti.

Figura 6 - Mario in Super Mario Sunshine

CURIOSITÀ TECNICHE

Prima di chiudere, vediamo insieme qualche curiosità tecnica su Nintendo GameCube.

Central Processing Unit:
- IBM PowerPC Gekko-32 bit @485 MHz
- 64 Kbyte cache di primo livello
- 256 Kbyte cache di secondo livello
- Processo di fabbricazione a 180 nanometri
- Performance complessiva in MIPS: 1125 MIPS
- Performance complessiva: 1,9 GigaFLOPS

Graphics Processing Unit:
- ATI Flipper da 51 Milioni di transistor @162 MHz
- Rendering grafico di picco pari a oltre 20 milioni di poligoni (20,25 milioni di triangoli/ 405 mila vertici)
- Ombreggiatura a 5 Shaders di tipo non-unificato
- Fillrate in pixel da 648 megasamples al secondo
- Fillrate in texel da 648 megatexel al secondo
- Filtro lineare (bilineare e trilineare)
- 4 Render Output Units
- Performance complessiva dichiarata in virgola mobile: 8,6 GigaFLOPS

Memoria:
- 43 MByte RAM di tipo non-unificata
- 24 MByte RAM principale @324 MHz / Bandwidth @2,7 GB/s
- 3 MByte SRAM-embedded

LM80C Color Computer

Un computer autocostruito nel 2019 basato sullo Z80 - parte 2

di Leonardo Miliani

Dopo la prima infarinatura avuta nel precedente articolo pubblicato sul n. 21 della rivista, a cui vi rimando nel caso ve lo foste perso, in cui abbiamo visto com'è strutturato un computer e quali sono le componenti base che servono al suo funzionamento, in questa seconda parte analizzeremo come la CPU si interfaccia alle memorie ed ai dispositivi di Input/Output (I/O).

La comunicazione dei dati all'interno di un computer avviene tramite segnali elettrici digitali, che possono assumere un valore "alto" o "basso", ad indicare un valore di "1" o "0", rispettivamente. Questo tipo di segnale non viene usato soltanto per lo scambio di informazioni ma anche per l'attivazione o disattivazione di unità esterne alla CPU: siccome un computer che si rispetti è composto da più chip, bisogna inventare un sistema affinché il processore riesca a selezionare di volta in volta il chip con cui intende scambiare dati. Fortunatamente non dobbiamo inventare niente perché c'è chi già l'ha fatto per noi: si tratta di una famiglia di integrati notissima ed usata da un sacco di tempo, la famiglia 7400, presentata nel 1966 da Texas Instruments come versione economica (in package di tipo plastico) della famiglia 5400, introdotta 2 anni prima. Questa famiglia di integrati TTL racchiude chip tipo porte logiche AND, OR, NAND, decoder di indirizzi, flip-flop, buffer, inverter ed altro. Quando sentite dire che i computer degli anni '60 e '70 del XX secolo erano costruiti usando logiche TTL, sappiate che erano assemblati usando proprio decine, centinaia, di questi singoli integrati. IBM, Olivetti ed altri costruttori realizzavano interi computer con logiche TTL. Un famoso computer basato su logiche TTL fu il Datapoint 2200 di Computer Terminal Corporation (CTC): la sua CPU, invece che essere composta da un unico chip (era stato inizialmente chiesto sia ad Intel che a Texas Intruments di realizzare una CPU su un singolo integrato ma nessuna delle due compagnie riuscì a rispettare i tempi e le richieste tecniche di CTC) era composta da 100 logiche 7400 (figura 1). Menziono questo computer, costruito dal 1970 fino al 1979, perché lo schema logico della sua CPU è stata il seme dell'architettura x86 di Intel.

Figura 1: Datapoint 2200: la sua CPU era composta da 100 integrati della serie 7400 (foto: Wikimedia Commons)

Tornando alla nostra famiglia 7400, ho selezionato l'integrato 74HCT139, un decoder di indirizzi 2-a-4, vale a dire che con 2 ingressi si può selezionare una di 4 uscite disponibili. Questo è facile da capire ricorrendo alle potenze del 2: siccome un segnale digitale può assumere solo i valori di 0 e 1, ossia 2 distinti valori, 2 ingressi possono indirizzare 2^2 = 4 uscite. La sigla dell'integrato è composta da 3 parti: 74-HCT-139. "74" è la serie, "HCT" indica la famiglia e "139" identifica l'integrato. La sigla identificativa della famiglia è molto importante perché nel corso degli anni sono state prodotte numerose serie, ognuna compatibile con i livelli delle uscite dei transistor usati in quel periodo. Affinché un computer funzioni è importante usare integrati della giusta famiglia, i cui tipi e livelli di tensione siano compatibili con i chip principali del computer (CPU, chip audio e video, periferiche). Siccome tutti i suddetti chip presenti nell'LM80C sono stati selezionati di tipo CMOS (la tecnologia costruttiva dei transistor che li compongono), ho selezionato integrati della serie 7400 appartenenti alla famiglia HCT, sigla che indica integrati realizzati in tecnologia CMOS ad alta velocità e basso consumo elettrico. E' bene non mescolare famiglie diverse: quindi nel caso vogliate replicare il computer, ma anche acquistate la componenti per riparare il vostro, procuratevi logiche 7400 e integrati tutti dello stesso tipo, nel caso dell'LM80C di tipo CMOS (sono riconoscibili perché nella loro sigla hanno una "C", ad esempio Z84C0008PEG per uno Z80 in versione CMOS ad 8 MHz di frequenza massima). Per comodità, da ora in poi gli integrati della serie 7400 saranno indicati senza le lettere centrali, assumendo che essi sono tutti della famiglia HCT.

In figura 2 potete vedere il decoder con i collegamenti essenziali al suo funzionamento. Il pin "E" (per "Enable", abilitazione) è il piedino che serve ad abilitare, ossia selezionare, l'integrato; i pin A0 e A1 servono come ingressi di selezione e la combinazione dei segnali su di essi determina quale uscita, indicata dai piedini O0..O3, viene attivata. Faccio notare una cosa: potete vedere che su alcuni pin c'è un simbolo a forma di cerchio (in altri CAD o su altri profili di integrato potete trovare altri simboli, ad esempio un triangolo). Quel simbolo indica che il segnale è "attivo BASSO": ciò vuol dire che affinché l'integrato lo riconosca come attivo serve che sia a livello basso, o "0" per capirsi. Tornando con la mente al

precedente articolo ed alla piedinatura dello Z80, vi ricorderete come fra i pin di controllo del sistema della CPU ce n'erano 2 indicati con le sigle MREQ e IORQ: il primo è attivo quando la CPU vuole accedere alla memoria, ossia l'indirizzo presentato sul bus indirizzi è da intendersi destinato ad un'operazione di lettura o scrittura in memoria, mentre il secondo è attivo quando la CPU vuole eseguire un'operazione che coinvolga un chip periferico. Abbiamo visto che sia la famiglia di integrati Z80 sia gli integrati della serie 7400 riconoscono come segnali attivi quelli che hanno un valore basso: è facile quindi capire che a noi basta collegare il pin MREQ dello Z80 con il pin di abilitazione del 74139: quando questa linea andrà bassa, ossia lo Z80 chiederà l'accesso alla memoria, il livello basso su questo pin abiliterà il decoder 74139.

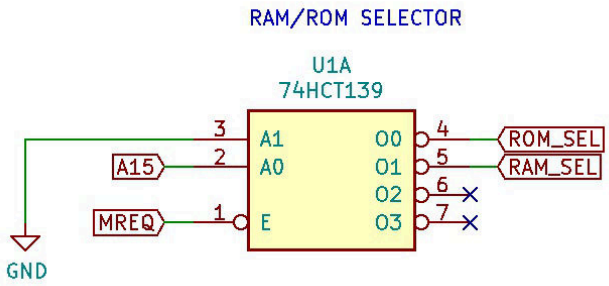


Figura 2: il 74139 usato come decoder degli indirizzi, per selezionare la RAM e la ROM

Se vi ricordate le specifiche del computer LM80C, la memoria è composta da 32 KB di ROM e 32 KB di RAM: la prima occupa lo spazio di indirizzi (indicato in esadecimale) da $0000 a $7FFF mentre la seconda da $8000 a $FFFF (questo perché lo Z80, dopo un reset, inizia ad eseguire le istruzioni a partire dall'indirizzo $0000 per cui la ROM deve occupare la parte bassa dello spazio di indirizzamento). Per indirizzare 32 KB di memoria, ossia 32.768 celle, occorrono 15 linee: difatti, 2^15 da proprio 32.768. Ora, visto che il bus indirizzi è composto da 16 linee è facile capire che possiamo usare la sedicesima linea come selettore del banco di memoria da 32 KB a cui vogliamo accedere. Difatti gli indirizzi $0000 e $7FFF, assegnati alla ROM, in binario sono rispettivamente 0000.0000.0000.0000 e 0111.1111.1111.1111; similarmente, gli indirizzi $8000 e $FFFF, assegnati alla RAM, sono in binario 1000.0000.0000.0000 e 1111.1111.1111.1111. È facile intuire perciò che il sedicesimo bit (evidenziato in grassetto) assume il valore 1 solo nel momento in cui accediamo ad un indirizzo appartenente alla metà superiore dello spazio indirizzi (ossia dalla cella $8000 compresa in poi) perciò faremo in modo che il pin A15 della CPU selezioni la ROM quando è a livello basso, e viceversa la RAM quando è a livello alto. Ciò è fatto collegandolo direttamente al pin A1 del 74139. Il pin A0, invece, lo colleghiamo direttamente a massa, affinché abbia sempre un valore fisso e la selezione avvenga esclusivamente con il cambio di stato dell'altro pin. Sui pin di input del 74139 potremo perciò avere solo 2 combinazioni: 00 oppure 01. Nel primo caso l'integrato abiliterà l'uscita O0, collegata al pin di abilitazione del chip di memoria ROM, mentre nel secondo l'uscita O1, collegata alla RAM. In figura 3 sono ripresi dallo schema elettrico i chip di RAM e ROM (per comodità affiancati) con le loro linee di abilitazione evidenziate in rosso. Siccome il decoder emette un segnale basso sull'uscita selezionata e, contemporaneamente, un segnale alto sull'altra, un solo chip per volta riceverà il segnale di abilitazione. A questo punto, i segnali sui pin "RD" ("read", lettura) e "WR" ("write", scrittura) dello Z80 saranno usati per indicare al chip di memoria selezionato l'operazione che la CPU intende effettuare. Va da sé che per la ROM (anzi, EEPROM), sarà possibile solo leggere: se vedete, il relativo pin "WR" è infatti connesso direttamente ai 5V, perché esso viene usato solo in fase di programmazione del firmware.

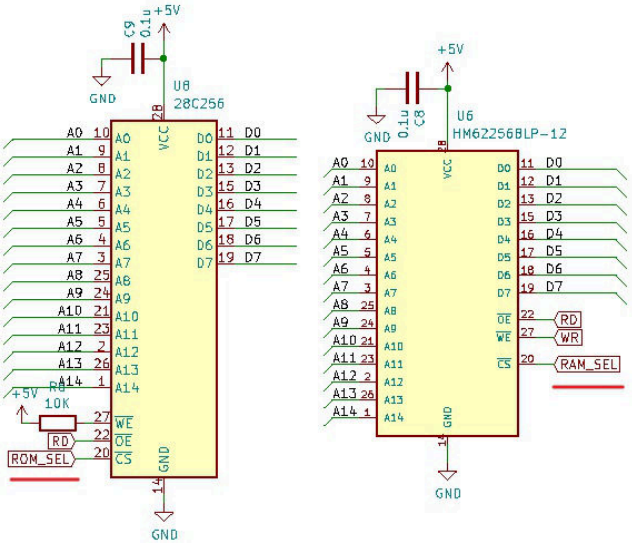


Figura 3: memorie ROM e RAM con le linee di selezione evidenziate

Abbiamo visto in azione il decoder di indirizzi per la selezione dei chip RAM e ROM da parte dello Z80. Come in questo caso, anche nel caso del dialogo con le unità periferiche entrano in ballo dei segnali per selezionare i vari chip ed altri per lo scambio dei dati. Torniamo per un attimo al nostro schema elettrico ed andiamo a vedere il decoder di I/O, un chip simile a quello usato per selezionare le memorie ma un pochino più complesso. Questo chip ha infatti 3 input grazie ai quali può selezionare fino a 8 unità indipendenti (infatti, 2^3=8): l'integrato si chiama 74HCT138. Perché non abbiano usato lo stesso integrato utilizzato per le memorie? Perché 4 sole uscite non ci bastavano, essendo le periferiche dell'LM80C ben 5 (con la possibilità in futuro di aumentare ulteriormente, espandendo il computer): abbiamo un chip per la comunicazione parallela, uno per quella seriale, un timer usato per generare segnali di clock e interrupt, ed infine il chip video e quello audio. Ognuno di questi integrati deve essere selezionato in modo univoco, altrimenti il computer non funziona.

In figura 4 potete vedere lo schema di collegamento del 74138. A differenza del 74139, visto in precedenza, il 74138 ha, oltre alle già menzionate 3 linee di ingresso, ben 3 linee di selezione dell'integrato stesso. Grazie a questa pletora di ingressi possiamo usare diversi 74138 ed attivarne uno solo quando una determinata combinazione di segnali si presenta sui loro ingressi di abilitazione. Ritornando al precedente articolo, dove ho analizzato i motivi che mi hanno spinto ad optare per lo Zilog Z80

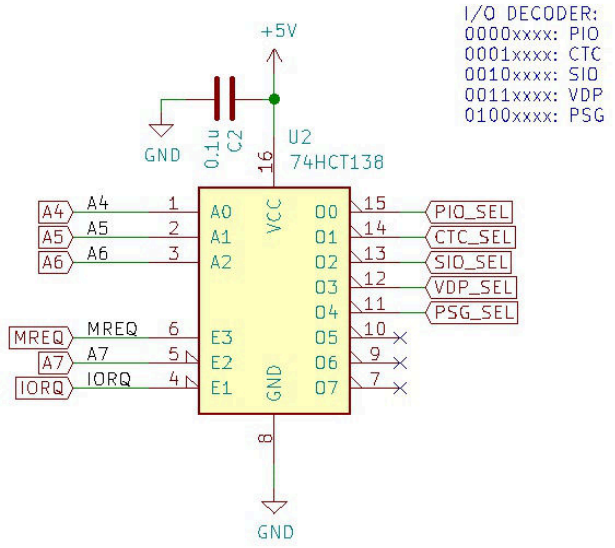


Figura 4: il 74138, usato come decoder per l'I/O

come CPU del mio computer, vi ricorderete che fra quelli a suo favore c'era la gestione di uno spazio di indirizzi di I/O separato dallo spazio di indirizzamento della memoria. Che significa questo? Spieghiamolo brevemente, facendo un confronto con una notissima CPU che non ha uno spazio di indirizzamento I/O separato, il 6502 (e derivati). Chi ha posseduto un computer Commodore si ricorderà che per accedere ai registri video o del suono si ricorreva a PEEK e POKE fatte a specifici indirizzi della memoria: in questo caso si dice che i registri sono mappati in memoria, indicando con questa locuzione il fatto che determinati indirizzi in realtà non sono più associati a celle di memoria ma servono a scambiare dati con i chip periferici: i progettisti del computer hanno costruito dei circuiti di decodifica simili a quelli esposti in questo articolo grazie a cui, quando la CPU seleziona una particolare combinazione di segnali sul bus indirizzi, l'hardware della macchina va a selezionare un determinato chip. In questo modo si sacrificano porzioni più o meno grandi della memoria, non potendo i blocchi indirizzi interessati essere più utilizzati per l'accesso alla memoria. Una CPU che invece gestisce 2 spazi di indirizzamento separati per la memoria e l'I/O permette di poter utilizzare tutto lo spazio di indirizzamento della memoria: lo Zilog Z80 indirizza in maniera diretta 64 KB, ed infatti l'LM80C non perde neanche uno di questi byte, avendo un accesso diretto a 65.536 singole celle di memoria. In più fornisce 256 indirizzi univoci per l'accesso ad altrettante unità di I/O (in realtà può indirizzarne 65.536, ma non voglio mettere troppa carne al fuoco, rimandando alla lettura del data sheet della CPU il lettore interessato). La discriminante la fanno i 2 semplici segnali elettrici già visti in precedenza: MREQ e IORQ. Il primo lo abbiamo usato per attivare il decoder delle memorie, e lo useremo anche ora, in congiunzione con il secondo, per attivare il nostro 74138. Perché usiamo entrambi i segnali, se per selezionare il 74139 abbiamo usato solo il segnale MREQ? Perché, come detto, il 74138 ha 3 ingressi di abilitazione dell'integrato, indicati in figura 4 con i pin denominati E1/E2/E3. La combinazione che il 74138 riconosce sui pin di "enable" è BASSO/BASSO/ALTO, rispettivamente per E1, E2 ed E3: ciò significa che affinché l'integrato capisca che vogliamo dialogare proprio con lui dobbiamo far sì che contemporaneamente arrivino sui 3 piedini indicati i segnali con i suddetti valori. Come si vede dallo schema, su E1 ed E3 arrivano rispettivamente i segnali dei pin IORQ e MREQ, emessi di valore opposto: IORQ è basso e MREQ è alto quando la CPU vuole accedere ad una periferica di I/O, mentre IORQ è alto e MREQ basso quando essa vuole accedere alla memoria. Alla luce di questo, è facile capire dal valore che assumono sui pin, che il 74138 si abilita quando riconosce i livelli della prima combinazione. Infine, il terzo pin E2 è collegato al pin A7 del bus indirizzi della CPU. Perché? Un aiuto ci viene dalla tabellina di decodifica presente nell'immagine: se osservate le combinazioni riportate, il pin A7, che è l'ottavo pin del bus indirizzi e quindi porta il valore del bit più alto del primo byte che compone l'indirizzo, assume sempre valore 0: ciò significa che tutte le periferiche il cui indirizzo va da $00 a $7F, in binario da 0000.0000 a 0111.1111, saranno gestite da questo 74138. Questo serve a semplificarci le cose nel caso in cui in futuro volessimo espandere le periferiche del computer ed assegnare ad un altro decoder 74138 il compito di gestire queste periferiche oppure se assegnassimo ad alcune di esse un indirizzo compreso nell'intervallo $80-$FF, in binario 1000.0000 – 1111.1111: qui potete vedere che l'ottavo bit, evidenziato in grassetto, assume sempre valore 1 (a differenza del precedente caso, dove assume sempre valore 0). Sistemati i pin di abilitazione, restano i pin di input che selezionano le uscite dell'integrato. Rifacendoci nuovamente alla tabellina, abbiamo collegato le 3 linee A0/A1/A2 alle linee A4/A5/A6 del bus indirizzi, così che la tripletta in ingresso selezioni il corretto chip: con 000 selezioniamo il PIO (periferica parallela), con 001 il CTC (il timer), con 010 il SIO (periferica seriale), con 011 il VDP (il chip video) e con 100 il PSG (chip sonoro). Abbiamo scelto proprio le linee A4/A5/A6 del bus indirizzi semplicemente perché tutte le periferiche contengono al loro interno più unità operative oppure perché necessitano di uno o più segnali per selezionare la modalità operativa, riservandoci questi bit come veri e propri selettori. Ad esempio, il PIO possiede 2 porte su cui si può operare in 2 modalità diverse, e lo stesso dicasi per il SIO; il CTC ha invece 4 canali interni, mentre il VDP ed il PSG usano un ulterore pin per indicare al chip se il valore indica la selezione un registro oppure se va interpretato come dato.

Bene, anche in questo articolo ho trattato tanti argomenti che, spero, siano risultati interessanti e vi abbiano aiutato nella comprensione di come i computer indirizzano i vari componenti interni. Nel prossimo articolo vedremo di affrontare nel dettaglio l'uso di qualche chip periferico.

Link utili

Pagina internet di riferimento del progetto:
https://www.leonardomiliani.com/en/lm80c/

Schemi elettrici e codice sorgente del firmware:
https://github.com/leomil72/LM80C

Pagina su Hackaday:
https://hackaday.io/project/165246-lm80c-color-computer

Intervista a Toru Iwatani

di Carlo Nithaiah Del Mar Pirazzini

Quarant'anni fa, nel maggio del 1980, uno sviluppatore giapponese di giochi Namco creò un gioco che vedeva una torta gialla semi-mangiata, navigare in un labirinto mentre cercava di mangiare tutte le pillole gialle e i frutti evitando quattro fantasmini colorati al suo inseguimento. Il nome del gioco, progettato e creato da un giovane impiegato di nome Toru Iwatani, era Pakkuman, o meglio, Puck Man.

Quando fu concesso in licenza negli Stati Uniti, il nome diventò Pac-Man ed entrò nella leggenda videoludica!

Da allora, il classico gioco arcade ha generato oltre una dozzina di sequel di giochi, alcuni dei quali hanno visto il coinvolgimento dello stesso Iwatani, un cartone animato e persino un film (anche se non riuscitissimo - ndN).

Abbiamo incontrato il professor Toru Iwatani grazie all'amico Takahiro Yoshioka da Tokio, per parlare del suo gioco di successo, dell'eredità che ha lasciato alle spalle, e per scoprire di più sui suoi progetti futuri.

Siamo qui con il Maestro Toru Iwatani e siamo pronti per fare due chiacchiere sulla sua vita, su PAC-MAN e sul futuro. Grazie Maestro per la disponibilità, partiamo dalle domande. Come mai lasciò l'industria dei videogiochi? Cosa accadde in seguito?

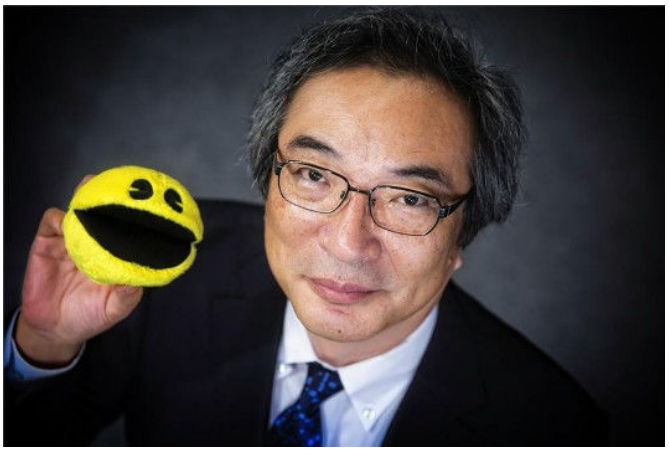
Grazie a voi e Forza Italia! Allora... ho lasciato l'industria dei videogiochi per lavorare in università e proseguire lo studio della teoria del gioco per il bene sociale, ovvero uno studio sull'utilità del momento ludico nella vita, nella socializzazione e nello sviluppo della persona. Questo studio prosegue anche grazie alla mia esperienza e conoscenza nella creazione di giochi per 30 anni. Attualmente sto insegnando agli studenti Teoria dei giochi, Pianificazione e sviluppo nel Tokyo Polytechnic University al Dipartimento di Applied Computer Science a Tokio. E vi dirò che lo faccio con una certa soddisfazione!

Lei gioca ancora con i videogiochi? Cosa la attrae maggiormente?

Mi capita di giocare con i famosi giochi Arcade (da Sala Giochi) come i flipper e i pachinko. Mi rilassano e li adoro. Altrimenti prediligo i giochi per dispositivi portatili come gli smartphone e le console Nintendo. Sono per i giochi con regole semplici e con potenzialità di sviluppo durante il gioco. Non amo le cose "troppo realitistiche".

E in famiglia? I suoi figli sono videogiocatori?

Ho un figlio di 30 anni e una figlia di 26 anni. Giocano spesso con le loro console portatili o da casa. Non so molto altro. So che hanno una passione per Smash Bros di Nintendo.



A parte Pac-Man e altri personaggi che ha creato, qual è il suo personaggio di videogioco preferito e perché? C'è un gioco che non vede l'ora di giocare?

Mi piace il design semplice e carino dei fantasmi di Pac-Man. Mi piace anche il personaggio vermiforme di Tube in "Marble Madness" di ATARI. Mi piace la semplicità colorata dei giochi di Super Mario o dei prodotti col cavaliere Arthur. Quello che mi piace è entrare nel ruolo di simpatici personaggi a cui mi sento emotivamente legato nel gioco.

Sono passati 40 anni e ci sono milioni di gamers che giocano al tuo Pac-Man. Si aspettava che il gioco durasse così a lungo, e come si senti quando ne sente parlare e che i nuovi fan stanno ancora scoprendo Pac-Man dopo tutto questo tempo?

Il concetto originale di Pac-Man era il risultato del mio desiderio di creare un gioco che potesse piacere a tutti. Con in mente un pubblico femminile, volevo creare un gioco basato sul mangiare (motivo per cui il nome deriva dalla parola giapponese onomatopeica pakupaku, il suono che si fa quando si apre e si chiude la bocca mentre si mangia - ndN). Quando ci pensavo, ero in un ristorante e ho notato una pizza con una fetta mancante. Ho pensato: "Eccolo!" Questa è stata l'ispirazione ed è diventata la forma e il concetto generale di Pac-Man.

In quel periodo, le sale gioco erano sature di giochi in cui uccidere gli alieni era l'obiettivo principale. Molti di questi giochi avevano grandi concetti che erano divertenti da giocare, ma sentivo che nessuno di questi era ugualmente accessibile alle donne, ai bambini o alle coppie. All'epoca i giochi mancavano di varietà: questi tipi di giochi avevano un'immagine piuttosto brutale e un pubblico in gran parte maschile e adulto.

Volevo ravvivare i centri di divertimento del gioco portando sul palcoscenico giocatori femminili e coppie. Sono stato ispirato da molti dei giochi di Atari in questo senso; avevano alcuni concetti innovativi che mi hanno insegnato molto

sul design. Non avevo dubbi sul fatto che il concetto di Pac-Man avrebbe attratto le donne, anche se non passavo molto tempo a cercare la loro opinione sulle idee.
Quando finalmente ho mostrato la proposta al mio capo e ai miei colleghi, la risposta che ho ricevuto non è stata poi così travolgente. Avevo già creato altri tre giochi prima di Pac-Man, quindi ero già noto come game designer tra i miei colleghi. Di conseguenza, molte persone sono state in grado di guardare oltre questo diverso tipo di concetto e permettermi di esplorare le possibilità del gioco.

Cosa andò bene allora?

Quando stavamo realizzando Pac-Man 40 anni fa, non avevamo gli stessi limiti di budget o le scadenze che la maggior parte degli sviluppatori incontra oggi. Non avevamo pressioni estreme e in queto modo siamo riusciti a creare qualcosa di cui eravamo davvero sodisfatti. Era tutto molto stimolante! In questo modo siamo riusciti ad includere davvero tutto quello che volevamo nel prodotto finale!
La differenza la facemmo con il piccolo team. Ai giorni di oggi è più difficile con i team moderni, enormi e quasi impersonali, il nostro team era composto solo da cinque membri e in questo modo era facile controllare il flusso di lavoro. Non c'erano problemi di comunicazione né di chimica. Era impossibile (ride - ndN)! Perché si lavorava tutti in una grossa stanza.
Non possiamo però non includere la vera mossa vincente del gioco. La Semplicità! Quando mettevamo in evidenza gli elementi divertenti del gioco lo abbiamo fatto usando un approccio a prova di errore. Tutto doveva essere semplice, intuitivo e dinamico. La semplicità era la base del gioco e la sua grande fortuna.

Cosa ne pensa dei giochi di oggi?

Dipende da molti fattori. Ammiro il lavoro delle moderne case di produzione. Ammiro il lavoro di chi si occupa di muovere tutto quello che si vede oggi negli schermi. Ma io sono per le cose semplici e veloci. Il videogioco mi deve rilassare e divertire. Deve essere "amichevole" e "competitivo". Per questo prediligo il gioco su portatili e console.

Sono passati 40 anni da quando Pac-Man è nato. In tutti questi anni avrà sicuramente fatto un salto tra i ricordi e avrà sicuramente tratto delle conclusioni sul successo e sul suo lavoro.

Le conclusioni su Pac-Man? Mi portano felicità. Sono felice di come il mio personaggio abbia guadagnato una così grande popolarità in tutto il mondo. Nel 1980 sapevamo di avere un prodotto di alta qualità dopo averlo terminato e sapevamo che nulla potesse essere aggiunto o rimosso per renderlo migliore. Sono lieto che Pac-Man abbia guadagnato tale popolarità in tutto il mondo. Ma avevamo una paura incredibile. Non avevamo la minima idea di come sarebbe diventato un grande successo in tutto il mondo. Non avevamo previsto la travolgente reazione dal mercato estero (in America fu qualcosa di incredibile una vera Pac-Mania negli '80 - ndN). E sono felice che anche i giocatori di oggi conoscano Pac-Man e famiglia grazie alla loro semplicità e alla loro presenza su console di generazione in generazione. Insomma Pac è adorabile no? Semplice ed adorabile.

Siamo giunti alla fine dell'intervista e all'inizio della cena (questa intervista è stata realizzata durante una cena dove sia il maestro che Takahiro erano presenti - ndN). La ringraziamo per lo spazio e del tempo che ci ha concesso. Se vuole fare un saluto finale.

Un saluto alla felicità! Come sviluppatore di Pac-Man, sono stato in grado di conoscere molte persone che altrimenti non avrei avuto l'opportunità di incontrare. Queste persone sono tutte molto importanti per me e mi sento fortunato ad aver lavorato su un gioco così speciale. Sono stato fortunato a conoscere tantissimi giocatori e appassionati che mi hanno reso un "padre" orgoglioso. Questo mi ha reso felice e mi rende felice ogni giorno. Sono molto contento di come il mio lavoro sia diventato insegnamento per le giovani generazioni di game designer. Se ci pensiamo adesso è tutto partito da una pizza e da un puntino! Ironico.
Grazie a voi di avermi concesso il vostro spazio e un saluto speciale a tutta l'Italia, che ho nel cuore e che in questo momento difficile mi è molto vicina.
Forza Italia!

Ringraziamo il Maestro per l'intervista e il nostro amico Takahiro che si è prestato come redattore per conto nostro (Taka ti devo una birra quando sarai a Bologna - ndN)!

Il Maestro Iwatani ci ha promesso di farsi risentire nei prossimi numeri con alcune "novità".

Intervista a Luca Brocato - Una app per tutti "gli amanti della cassettina"

di Marco Pistorio

Ciao Luca, buongiorno! Da qualche giorno a questa parte circola in diversi gruppi Fb la notizia di una applicazione che stai realizzando, "PlayEdicola", un front-end per fruire in maniera più veloce ed immediata dei contenuti del noto sito www.edicola8bit.com, gestito da Giuseppe Di Lillo (Sovox).
Abbiamo deciso quindi di porti qualche domanda per conoscerti meglio e far conoscere ai nostri lettori il tuo progetto più nel dettaglio.

Raccontaci un pò di te Luca, e di come e quando è nata la tua 'retropassione' :)

Buongiorno a tutti voi. Ho 43 anni, sono sposato e ho due figlie. Premetto che la mia 'retropassione' non ha una nascita, è un evoluzione della mia passione per i computer che inizia all'età di 7 anni, nel lontano 1982, quando mio padre portò a casa il Commodore 64 (una macchina strabiliante). Cominciai subito a provare i listati del manuale. Poi grazie a Video Basic della Jackson ho veramente imparato a capire il funzionamento di un computer. Dal C64 passai ad Amiga sulla quale (essendo musicista) ho composto qualche mod con il protracker, e poi al PC dove ho scoperto la magia degli emulatori.

Come ti è venuto in mente di realizzare "PlayEdicola"? A chi si rivolge? Cosa permette di ottenere, in generale, questa applicazione?

Mentre sfogliavo il sito edicola8bit per rigiocare alcuni titoli delle famose cassette da edicola, mi sono accorto che impiegavo troppo tempo (sfogliare, cercare, scaricare, lanciare emulatore, selezionare il file...) e, alla fine, perdevo la voglia di giocare. Ci voleva qualcosa di immediato per valorizzare anche tutto il lavoro che ha fatto Giuseppe.
Al di là del fatto che le riviste pubblicavano giochi pirata, rappresentano oggi un qualcosa legato alla storia della vita di chi, come me, ha passato dei momenti spensierati, da soli o con fratelli, amici, compagni di scuola, cugini ec... Ricordi indelebili di ore in cui mio padre cercava di risolvere Agente Speciale (Impossible Mission), di pomeriggi passati con i compagni di scuola giocando alla coppa del mondo di Super Soccer (Microprose Soccer), di giornate intere cercando di finire Turrican... Visto che non si può viaggiare nel tempo, tutto questo possiamo riviverlo, in qualche modo.
PlayEdicola si rivolge appunto a chi è appassionato di retrogaming, per eliminare quella latenza che esiste tra pensare di giocare ad un vecchio gioco e giocarlo effettivamente.

Quali tecnologie hai impiegato per realizzare l'applicazione? Per quali motivi le hai scelte?

Quanto tempo è trascorso tra l'idea iniziale e la prima versione 'alpha' della applicazione?
A che punto del suo sviluppo ti trovi attualmente, esprimendolo con un numero tra 1 a 100?
Quanto tempo ritieni ci vorrà per vedere una prima versione stabile della applicazione?

Visto che ho cominciato a programmare con il vecchio Basic V2 del C64 e avendo avuto qualche esperienza con Visual Basic e Access di Microsoft, mi sono cimentato con questi ultimi. Non sono un programmatore professionista, mi definisco uno "smanettone", dato che non ho mai studiato.
L'idea è nata un paio di anni fa. La prima volta ho tentato, senza successo, cercando di catturare i link del sito. La seconda volta (settembre 2019) ho contattato Giuseppe e gli ho chiesto se voleva collaborare fornendomi il database del sito. Era da tempo che Giuseppe voleva creare un front-end ma non conoscendo la programmazione avrebbe dovuto affidarsi ad altre persone. Una volta ottenuto il database ho sviluppato il tutto.
La prima versione alpha ,o più precisamente una bozza, l'ho pubblicata il 15 Marzo 2020 nel gruppo Facebook di edicola8bit quindi sono passati 6 mesi. Teniamo conto che dedico pochissime ore a settimana per sviluppare.
Attualmente credo di trovarmi al punto 25 in una scala da 1 a 100. Non ho ancora idea di quanto tempo ci vorrà per avere una prima versione stabile.

Distribuirai l'applicazione gratuitamente o pensi a versioni successive che potranno essere distribuite a pagamento?

In verità ci ho pensato, e se dovessi distribuire l'applicazione a pagamento lo farei solo allo scopo di donare il ricavato a Giuseppe per l'enorme lavoro che ha svolto e che continua a svolgere. Ma penso che rimarrà gratuita con l'invito agli utilizzatori di contribuire al sito.

Trattandosi di un progetto che ha richiesto e richiederà ancora tempo e risorse, hai in mente delle strategie per finanziarlo? I nostri lettori potrebbero venirti incontro in tal senso e se si, come? In quale modo ritieni possano farlo?

Per il momento vado avanti con le mie forze, se qualcuno vuole contribuire si potrebbe pensare alle donazioni.

Prevedi di aggiungere nuove funzionalità all'applicazione? Ce ne sono alcune che hai già in mente? Se si, quali?

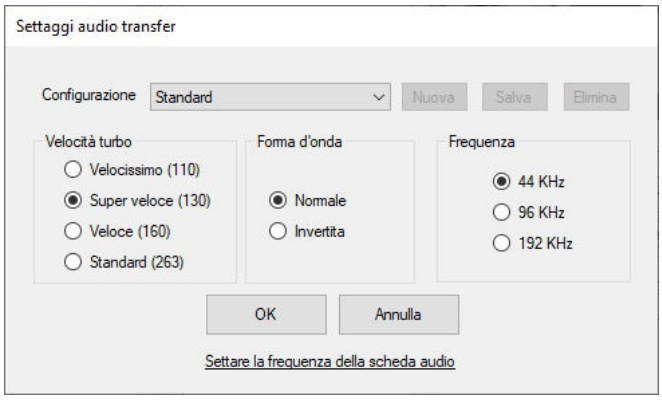
Si, innanzitutto implementare gli altri computer 8 bit di cui sono già presenti le relative riviste nel sito edicola8bit.

Ci sono altri progetti che conservi chiusi in un cassetto?

Si, i miei progetti musicali soprattutto.

Io e la redazione di RetroMagazine ti ringraziamo per aver risposto a queste domande, ci congratuliamo con te per il lavoro che hai già svolto e che abbiamo già avuto modo di valutare utilizzando la prima versione 'alpha' della tua applicazione e ci auguriamo che il progetto vada in porto nel minor tempo possibile e con la massima soddisfazione, sia per te che per tutti gli amici retroappassionati che seguono queste iniziative con estremo interesse.

Ciao e...in bocca al lupo, Luca!

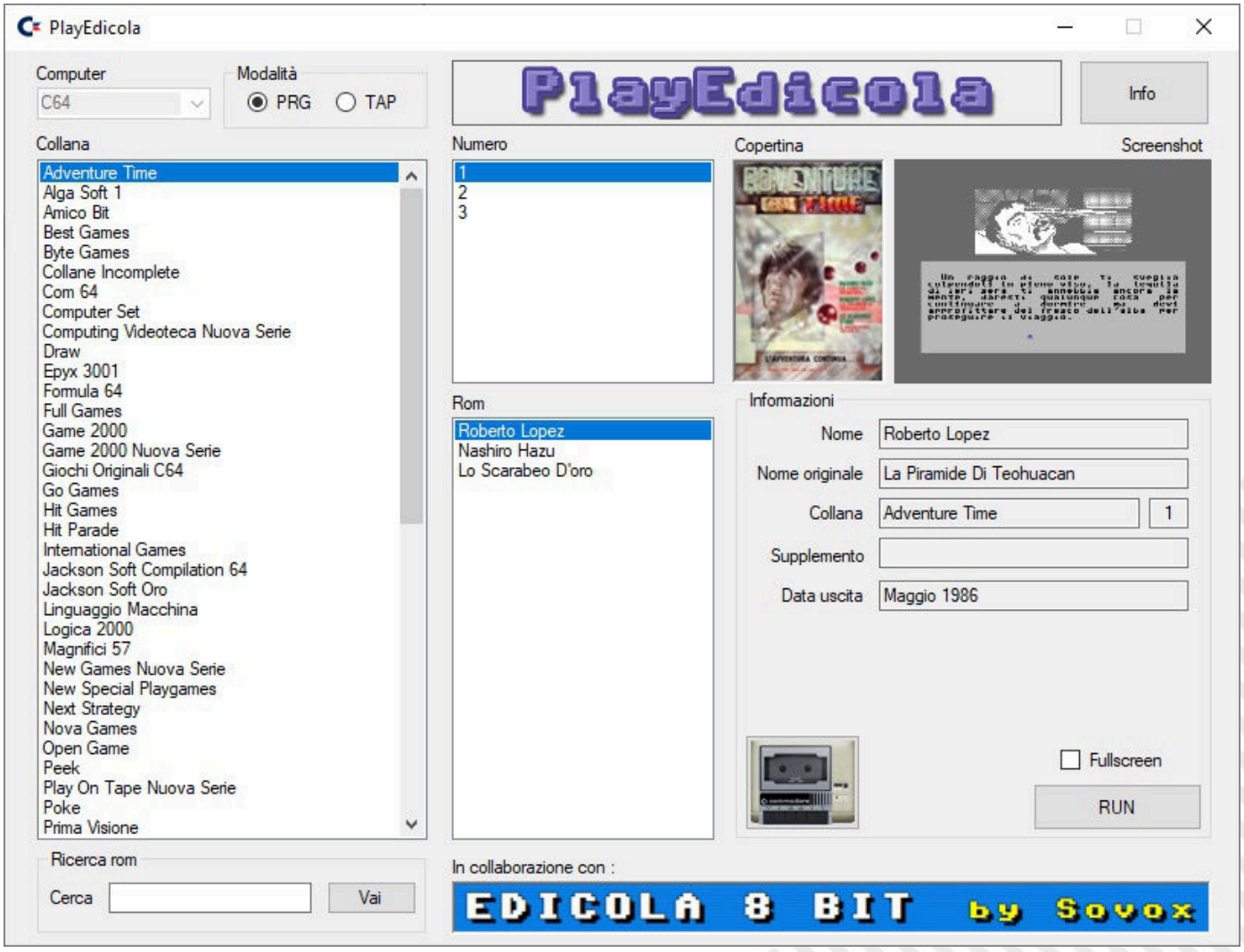


Finestra settaggi trasferiment programmi via audio

Riferimenti

• Pagina Fb del progetto:
https://www.facebook.com/playedicola/

• Sito Web Edicola 8 bit - Sovox
https://www.edicola8bit.com/default.php



Finestra principale del front-end (versione alpha)

Ritorno al... passato - prima tappa: Windows 98

di Marco Fiaschi

Era il lontano 2010 quando mi posi in maniera ripetitiva sempre lo stesso quesito: “Come posso utilizzare software di circa quindici anni fa al giorno d’oggi ?”.

Quasi per gioco, iniziai a cercare incessantemente la risposta a questa domanda e, dopo un’approfondita ricerca nell’ambito delle tecnologie informatiche del passato, sono giunto ad una conclusione. La risposta era semplice: virtualizzare i sistemi operativi.

Ricordo ancora con molto affetto il mio primissimo netbook sul quale iniziai gli “esperimenti” di virtualizzazione...con soli 2GB di RAM, un processore a 1.2GHz a 32bit non si poteva di certo ottenere una virtualizzazione estremamente performante. Cosi’ per un certo periodo fui costretto ad abbandonare questo sogno.

Dopo qualche anno passai ad un computer fisso, nettamente più potente del mio netbook con Vista sotto il “cofano”, e finalmente riuscii a sperimentare la virtualizzazione. Per chi non lo sapesse, il termine virtualizzazione indica la possibilità di astrarre e quindi emulare componenti hardware su un qualsiasi calcolatore, indipendentemente dalle sue capacità hardware e/o software. Detto questo, passiamo ai dettagli!

Con questo primo articolo inizieremo il viaggio nella storia dei sistemi operativi che, grazie a potenti software di emulazione, potremo riutilizzare sulle nostre moderne piattaforme. Come si evince nel titolo, cominciamo dal sistema operativo di casa Microsoft rilasciato nel lontano 25 giugno 1998. Windows 98 è stato un sistema dotato di kernel monolitico a 16 e 32 bit il cui supporto (purtroppo) è terminato l’11 luglio del 2006. Nato sotto il nome in codice Memphis, è entrato a far parte della famiglia di sistemi operativi denominata Windows 9x (a cui appartengono anche Windows 95 e Windows ME). Windows 98, come 95 o ME, fu sviluppato con un bootloader basato interamente su DOS (altro sistema operativo di Microsoft).

Windows 98 fu considerato “rivoluzionario” all’epoca, poiché fu il primo sistema operativo che supportava completamente le USB, includeva un nuovo file system ovvero il FAT32 (disponibile anche su Windows 95 tramite aggiornamento OSR2). Le prestazioni di questo sistema erano fenomenali. Grazie al supporto a FAT32, i dischi rigidi IDE potevano garantire velocità senza precedenti, utilizzando dati altamente compressi in modo da risparmiare spazio in MB.

Windows 98 oggi è considerato uno dei migliori sistemi operativi mai realizzati da Microsoft, godendo di una longevità che lo rende immortale. Curiosa è la scoperta avvenuta poco tempo fa riguardante l’aggiornamento della pagina di supporto di Windows 98, il che rende questo sistema operativo una vera e propria pietra miliare nella storia dell’informatica. E visto che anche voi sicuramente siete impazienti di poter riutilizzare il sistema operativo, vi lascio con una procedura dettagliata da seguire per poter emulare in tutta facilità Windows 98.

In rete si trovano diverse soluzioni che consentono di emulare sistemi operativi di qualsiasi architettura e di qualsiasi famiglia (Windows, Linux, MacOS, Sun Solaris ecc.), ma la scelta ricade sempre su VirtualBox o su VMWare. Per un utente “domestico” la scelta tra i due software è praticamente irrilevante poiché le necessità di emulazione sono ridotte ai minimi termini. Per queste guide faremo riferimento al software gratuito VirtualBox per una questione di migliore gestione del sistema da emulare (inserimento/rimozione di periferiche, schede di rete, schede audio, dischi rigidi ecc.).

Una volta installato e aperto VirtualBox, che ricordo essere disponibile anche per altri sistemi operativi oltre Windows, ci troveremo davanti un’interfaccia molto intuitiva. Clicchiamo su “Nuova” per poter creare una nuova macchina virtuale. Inseriamo poi il nome che vogliamo dare alla macchina, prestando attenzione a selezionare “Microsoft Windows” alla voce “Tipo” e “Windows 98” alla “Versione”.

Fatto ciò, VirtualBox ci consiglierà la quantità di memoria RAM (64MB) e lo spazio su disco (2GB) da assegnare al sistema da emulare. Se abbiamo a disposizione un calcolatore performante, possiamo decidere di alzare la memoria RAM a 512MB, accettando i 2GB di disco rigido suggeriti che dovrà essere del tipo VDI (VirtualBox Disk Image) allocato dinamicamente. Fatto questo, clicchiamo su Avvia e, nella successiva finestra, selezioniamo il lettore host da cui installare Windows 98, che potrà essere o il disco originale, o un file .ISO . Una volta avviato il disco (fisico o virtuale), ci troveremo davanti alla schermata di installazione del sistema operativo (foto 1).

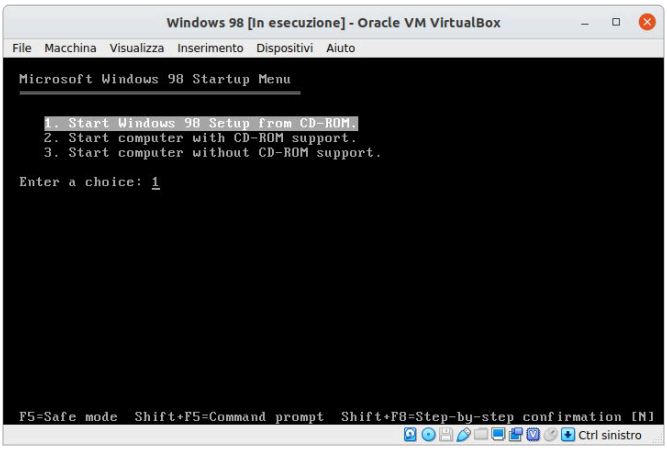


Figura 1

Selezioniamo la voce “Start Windows 98 Setup from CD-ROM”cliccando su Invio, diamo un altro Invio nella schermata successiva per continuare l’installazione (foto 2).

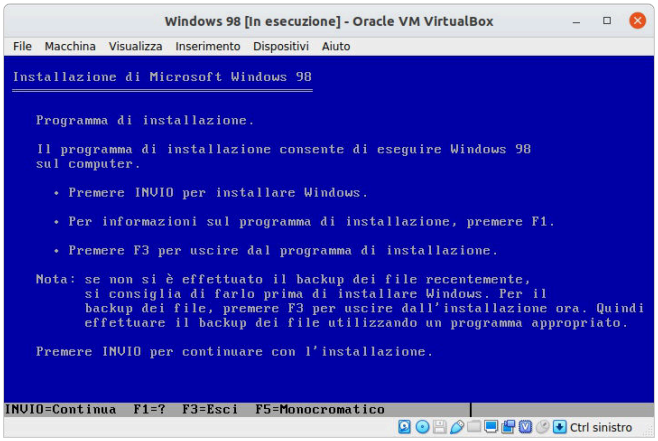


Figura 2

A questo punto va configurato lo spazio da allocare su disco (foto 3).

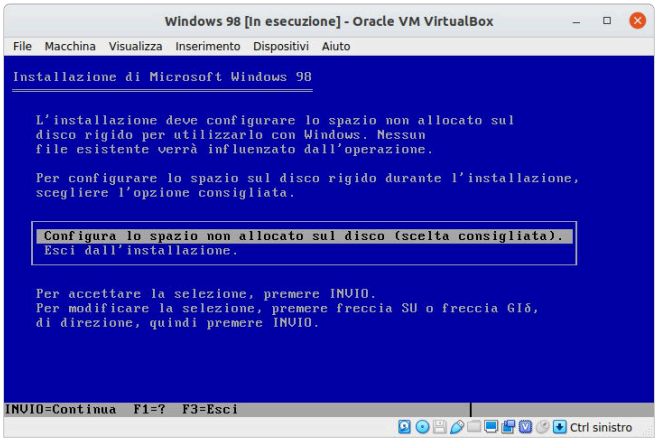


Figura 3

Quindi selezioniamo la voce “Configura lo spazio non allocato sul disco” dando Invio, attiviamo poi il supporto per i dischi grandi (foto 4), e attendiamo la formattazione dell’unità C: .
Una volta riavviato il sistema, ci ritroveremo davanti la

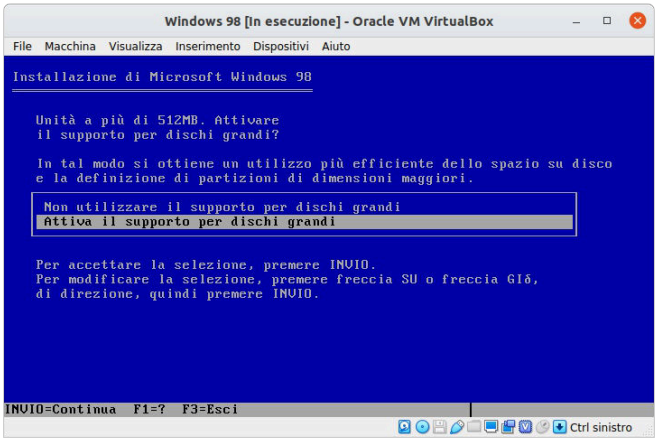


Figura 4

finestra di installazione “grafica” di Windows 98 (foto 5). Cliccando su Continua selezioniamo come directory “C:

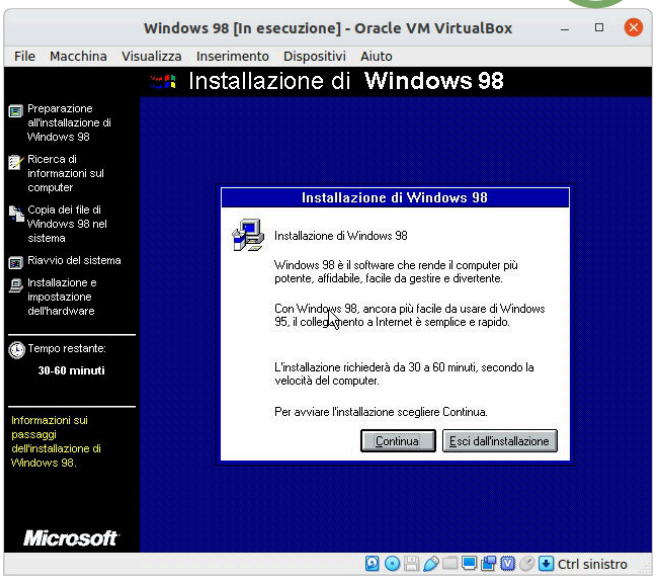


Figura 5

\\WINDOWS” e scegliamo una modalità d’installazione “Tipica” che, come ci suggerisce l’installer, è l’opzione consigliata per la maggior parte dei sistemi. Chi vuole personalizzare il setup, può farlo senza problemi, a patto che venga selezionata la voce “Personalizzata”.
Diamo successivamente un nome al computer e attendiamo l’installazione del sistema (foto 6).
Una volta riavviato il sistema, verremo catapultati nella

Figura 6

più totale nostalgia alla vista della schermata d’avvio di Windows 98 (foto 7), ma purtroppo l’installazione ancora non è terminata.

Nelle prossime schermate infatti dovremo selezionare il nome utente, accettare il contratto di licenza Microsoft, inserire il Product Key che potete trovare sulla copertina del manuale di Windows 98 e poi tirare un sospiro di sollievo quando verrà mostrata una schermata nella quale è scritto che Windows 98 ha salvato tutte le informazioni. Riavviato nuovamente il sistema, saranno rilevate in automatico tutto l’hardware non Plug & Play, dopodichè dovremo inserire la data e l’ora e il gioco è fatto!
Windows 98 si riavvierà automaticamente e, rullo di

Figura 7

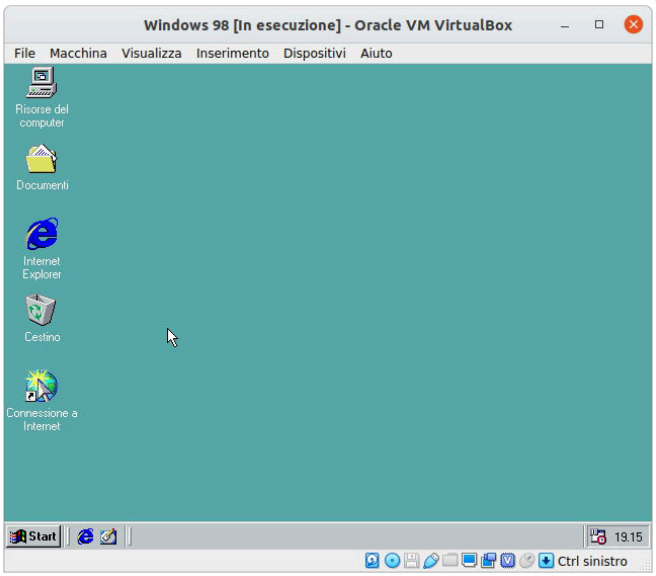


Figura 8

tamburi… il sistema è finalmente installato!

Ci ritroveremo infatti nel desktop di Windows (foto 8), accolti da un nostalgico suono d'avvio e da una schermata di benvenuto che mostra le caratteristiche e di funzionalità di Windows 98.
Da qui in poi possiamo cominciare ad installare tutti i programmi che vogliamo nella nostra macchina virtuale Windows 98 e cercando bene in rete, per questo sistema operativo, non c'è che l'imbarazzo della scelta.

Nella speranza che questo primo viaggio (NDR: l'ennesimo per RetroMagazine, il primo insieme a Marco) tra i sistemi operativi vi sia piaciuto, vi auguro un buon salto nel passato, con la speranza di avervi di nuovo a bordo per altri viaggi.

Buon divertimento!

Riferimenti

• Link per scaricare **VirtualBox**:
https://www.virtualbox.org/

• Link per scaricare **Windows 98 Second Edition**:
https://winworldpc.com/product/windows-98/98-second-edition

Nella stessa pagina si trovano anche altre versioni di questo sistema operativo

• Qui troverete senz'altro **Applicazioni** da installare:
https://winworldpc.com/library/applications

• E qui degli **Ambienti di Sviluppo**:
https://winworldpc.com/library/dev

• Se invece cercate un gioco in particolare, date un'occhiata a questa lista filtrando per anno:
https://www.myabandonware.com/browse/platform/windows/

Grafici in 2D con tre righe di BASIC

di Alberto Apostolo

Quando un programma genera una lista di risultati in forma numerica, può essere conveniente presentare un grafico generato dal computer.
Al giorno d'oggi esiste solo l'imbarazzo della scelta del software per realizzare il grafico di una funzione. Per esempio in Rete si può liberamente scaricare GNUPLOT.
Sulle riviste di Informatica degli anni '80 del XX secolo, il programmino BASIC per generare il grafico di una funzione era diventato un "tormentone" al pari di quello che calcolava i bioritmi.
Chi aveva un computer senza la grafica in alta risoluzione (e le opportune istruzioni BASIC per gestirla) doveva arrangiarsi stampando asterischi sullo schermo (di solito 21 righe per 32 colonne oppure 24 x 80).
I fortunati possessori di una stampante, potevano sfruttare righe di 80 colonne oppure di 132.
La tecnica per realizzare un grafico con il computer era semplicissima: un loop FOR-NEXT includeva una istruzione PRINT unita alla funzione TAB per la spaziatura dei caratteri. In Figura 1 è riportato un programma realizzato con il GWBASIC.
Le ordinate erano stampate sulla riga orizzontale mentre le ascisse progredivano in direzione opposta a quella del movimento della carta (scroll video) come nello schema di Figura 2.
Si ricorda che le caratteristiche della funzione TAB variano a seconda del dialetto BASIC. Occorre fare riferimento al manuale BASIC del proprio computer per non avere brutte sorprese durante la stesura dei programmi.

```
1010 FOR X=0 TO 6.3 STEP .25
1020 PRINT TAB(10+30*(SIN(X)+1));"*"
1030 NEXT X
```

```
                                        *
                                           *
                                              *
                                                 *
                                                    *
                                                       *
                                                       *
                                                    *
                                                 *
                                              *
                                           *
                                        *
                                     *
                                  *
                               *
                            *
                            *
                         *
                      *
                   *
                *
             *
          *
       *
    *
 *
 *
Ok
```

Figura 1

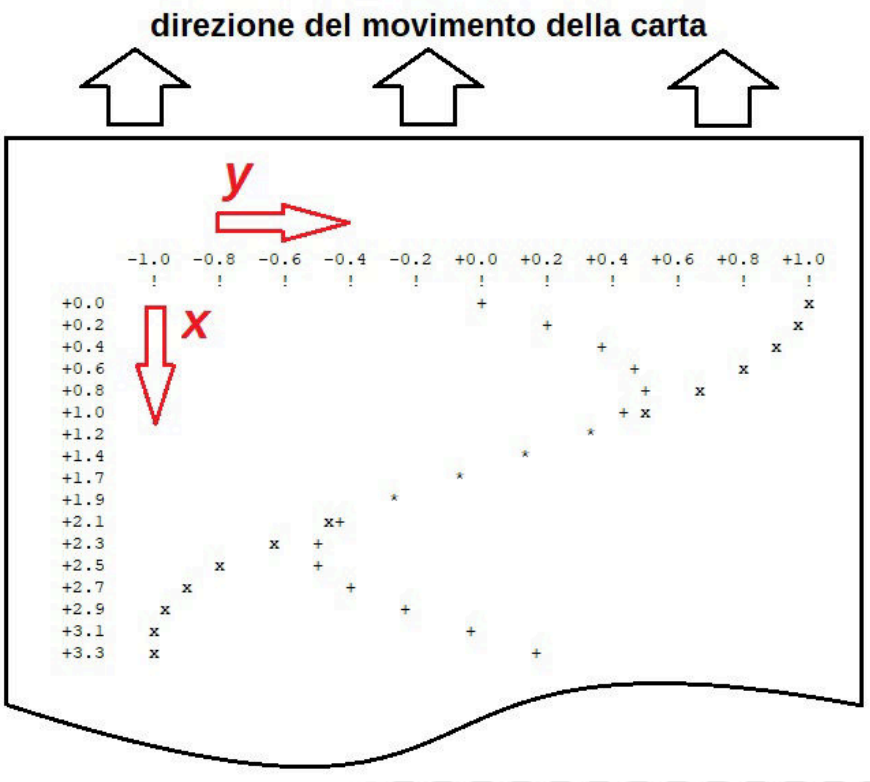


Figura 2

Il dimensionamento automatico (autoscaling)

Il programma visto in precedenza si può migliorare dimensionando automaticamente il grafico e

```
1000 DEF FNY1(X)=SIN(2*X)/2:DEF
FNY2(X)=COS(X)
1010 XMIN=0:XMAX=13:N=63:M=10:C=60
1020 YMIN=9.999999E+37:YMAX=-9.999999E+37
1030 DIM Y1(N),Y2(N)
1040 REM CALCOLA YMIN,YMAX
1050 FOR I=0 TO N
1060 X = (XMIN*(N-I)+XMAX*I)/N
1070 Y1(I) = FNY1(X):Y2(I) = FNY2(X)
1080 IF Y1(I)<YMIN THEN YMIN=Y1(I)
1090 IF Y2(I)<YMIN THEN YMIN=Y2(I)
1100 IF Y1(I)>YMAX THEN YMAX=Y1(I)
1110 IF Y2(I)>YMAX THEN YMAX=Y2(I)
1120 NEXT I
1130 A$=SPACE$(9)
1140 GOSUB 2010
1150 PRINT A$
1160 FOR I=0 TO N
1170 X = (XMIN*(N-I)+XMAX*I)/N
1180 TB1 = 10 + C*(Y1(I)-YMIN)/(YMAX-YMIN)
1190 TB2 = 10 + C*(Y2(I)-YMIN)/(YMAX-YMIN)
1200 PRINT USING"+##.#";X;
1210 IF ABS(TB1-TB2)<1 THEN PRINT
TAB(TB2);"*":GOTO 1240
1220 IF TB2<TB1 THEN PRINT
TAB(TB2);"x";TAB(TB1);"+":GOTO 1240
1230 PRINT TAB(TB1);"+";TAB(TB2);"x"
1240 NEXT I
1250 PRINT A$
1260 END
2010 FOR I=0 TO M
2020 Y=(YMIN*(M-I)+YMAX*I)/M
2030 PRINT TAB(7+C*I/M);USING"+##.#";Y;
2040 A$=A$+"!"+SPACE$(C/M-1)
2050 NEXT I
2060 PRINT
2070 RETURN
```

Figura 3

magari rappresentando due funzioni che si sovrappongono (come nel programma in GWBASIC di Figura 3).

Inizialmente sono fissati alcuni parametri come Xmin, Xmax, N intervalli per l'asse x, una risoluzione di C punti per l'asse y (con M valori da riportare su una scala graduata).

Nel loop di calcolo si memorizzano i valori assunti dalle due funzioni e si cercano i valori Ymin e Ymax che in seguito saranno utili per determinare l'autoscaling dell'asse y.

I punti della funzione 1 sono simboleggiati dai "+" , la funzione 2 dalle "x". Dove i valori saranno troppo "vicini" per essere rappresentati con due caratteri distinti si userà un unico carattere asterisco.

Un altro programma in GWBASIC è

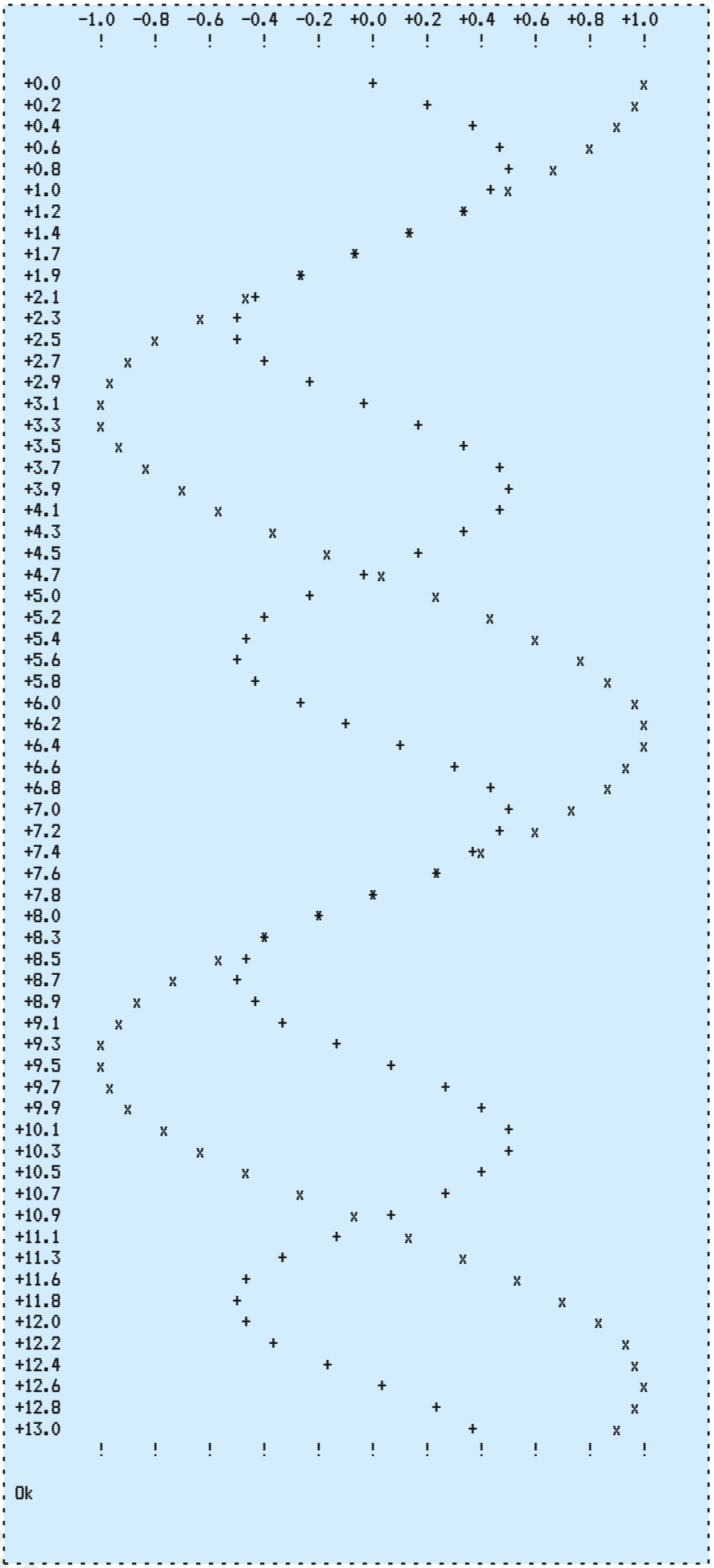


Figura 4 (output del programma in Figura 3)

```
1000 DEF FNX(T)=SIN(2*T)+COS(T)
1050 DEF FNY(T)=COS(2*T)+SIN(T)
1100 TMIN=0:TMAX=6.3
1125 N=63:M=6:R=40:C=60
1200 DIM X(N):DIM Y(N):DIM G$(R,C)
1240 XMIN=9.999999E+37:XMAX=-9.999999E+37
1250 FOR I=0 TO N
1300 T = TMIN + (TMAX-TMIN)*I/N
1350 X = FNX(T): Y = FNY(T)
1400 X(I)=X: Y(I)=Y
1450 IF X<XMIN THEN XMIN=X
1500 IF Y<YMIN THEN YMIN=Y
1550 IF X>XMAX THEN XMAX=X
1600 IF Y>YMAX THEN YMAX=Y
1650 NEXT I
1700 FOR I=0 TO N
1750 XC = C*(X(I)-XMIN)/(XMAX-XMIN)
1800 YC = R*(Y(I)-YMIN)/(YMAX-YMIN)
1850 G$(R-YC,XC)="*"
1900 NEXT I
1950 A$=SPACE$(9):GOSUB 2950:PRINT A$
2000 FOR I=0 TO R
2050 Y=YMIN + (YMAX-YMIN)*(R-I)/R
2100 PRINT USING "+##.###";Y;:PRINT " >";
2150 REM PROVA A METTERE ASSE Y
2200 IF XMIN>0 OR XMAX<0 THEN GOTO 2350
2250 XC = C*(-XMIN)/(XMAX-XMIN)
2300 IF G$(I,XC)<>"*" THEN G$(I,XC)="¦"
2400 FOR J=0 TO C
2450 REM PROVA A METTERE ASSE X
2500 IF YMIN>0 OR YMAX<0 THEN GOTO 2700
2550 YC = R - (R*(-YMIN)/(YMAX-YMIN))
2600 IF G$(YC,J)="¦" THEN G$(YC,J)="+"
2650 IF G$(YC,J)="" THEN G$(YC,J)="-"
2690 REM METTERE SPAZI NELLE PARTI VUOTE
2700 IF G$(I,J)="" THEN G$(I,J)=" "
2750 PRINT G$(I,J);
2800 NEXT J:PRINT "<":NEXT I
2850 PRINT A$:GOSUB 2950
2900 END
2950 FOR I=0 TO M
3000 X=(XMIN*(M-I)+XMAX*I)/M
3050 PRINT TAB(7+C*I/M);USING"+##.###";X;
3100 A$=A$+"¦"+SPACE$(C/M-1)
3150 NEXT I
3200 PRINT
3250 RETURN
```

Figura 5

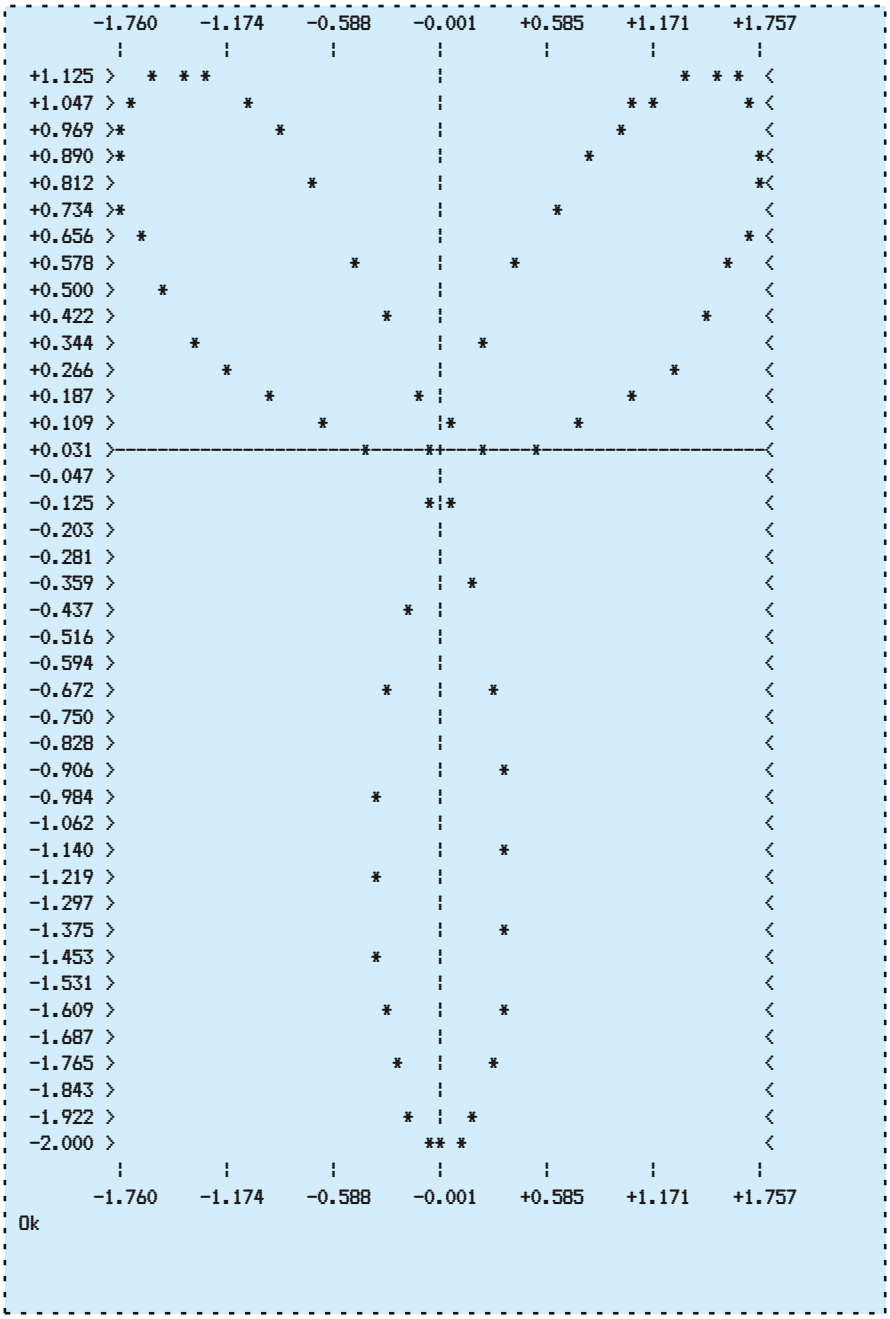


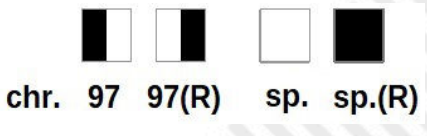
Figura 6 (una curva di Lissajous generata del programma a sinistra)

quello in Figura 5 nel quale si visualizza una funzione in forma parametrica dipendente dalla variabile T. Gli assi cartesiani sono orientati nel modo al quale siamo abituati, con le ascisse poste sull'asse orizzontale e le ordinate poste sull'asse verticale. A differenza del programma precedente i punti del grafico sono memorizzati all'interno di una matrice stringa G$ prima di essere stampati. La struttura del programma è un po' complicata dal fatto di voler includere nel grafico gli assi cartesiani se si trovano all'interno del quadrato

[Xmin,Xmax] X [Ymin,Ymax].

Grafica in media risoluzione

Sempre nei mitici anni '80, le parole "grafica in media risoluzione" indicavano spesso la grafica ottenuta posizionando in modo opportuno alcuni caratteri speciali disponibili su macchine come il Sinclair ZX81 (Figura 7).
Addirittura lo ZX81 comprendeva nel BASIC le istruzioni PLOT e UNPLOT (per marcare o cancellare un "punto") dando agli utenti una risoluzione 42 x 64 punti (esempio in Figura 8)
Ispirato dallo ZX81, avevo applicato a Scuola una cosa simile sul VIC 20 per "raddoppiare" la risoluzione dell'asse y quando dovevo stampare un grafico su carta con la Commodore 1515.
Utilizzavo in modo normale e in reverse il carattere 97 e lo spazio.



Duello in alta risoluzione tra Sinclair e Commodore

Lo ZX Spectrum (Figure 9 e 10) metteva a disposizione le istruzioni PLOT, DRAW e CIRCLE per la grafica in risoluzione 256 x 192 punti mentre sul VIC 20 (così come sul C64) occorreva armeggiare con POKE e PEEK.
Cercando in Rete, mi sono

Symbol	Code	How obtained	Symbol	Code	How obtained
	0	K or L SPACE		128	G SPACE
	1	G shifted 1		129	G shifted Q
	2	G shifted 2		130	G shifted W
	3	G shifted 7		131	G shifted 6
	4	G shifted 4		132	G shifted R
	5	G shifted 5		133	G shifted 8
	6	G shifted T		134	G shifted Y
	7	G shifted E		135	G shifted 3

Figura 7

```
10 LET X1=31
20 LET Y1=21
30 LET R1=10
40 LET R2=5
50 FOR T=0 TO 2*PI STEP .1
60 PLOT R1*SIN(T)+X1,R2*COS(T)+Y1
70 NEXT T
```

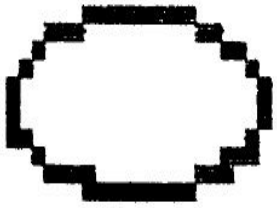


Figura 8

```
100 PRINT AT 0,10;"y=sin(x)"
110 PLOT 0,0: DRAW 0,175
120 PLOT 0,87: DRAW 255,0
130 FOR x=0 TO 255
140 PLOT x,SIN (x/25)*87+87
150 NEXT x
160>STOP

K
```

Figura 9

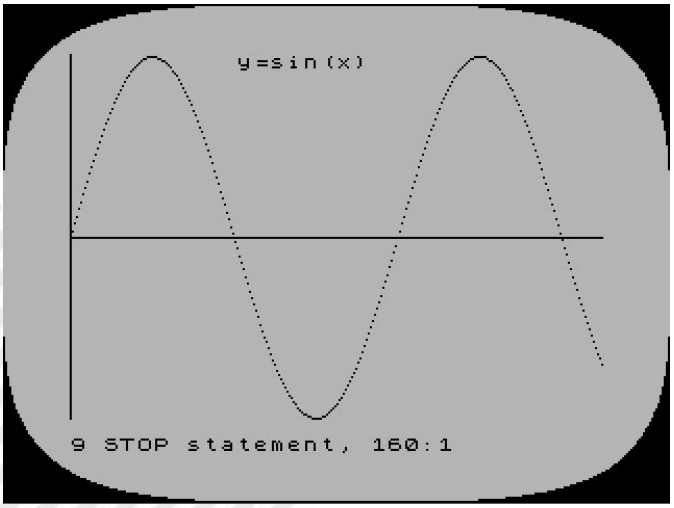
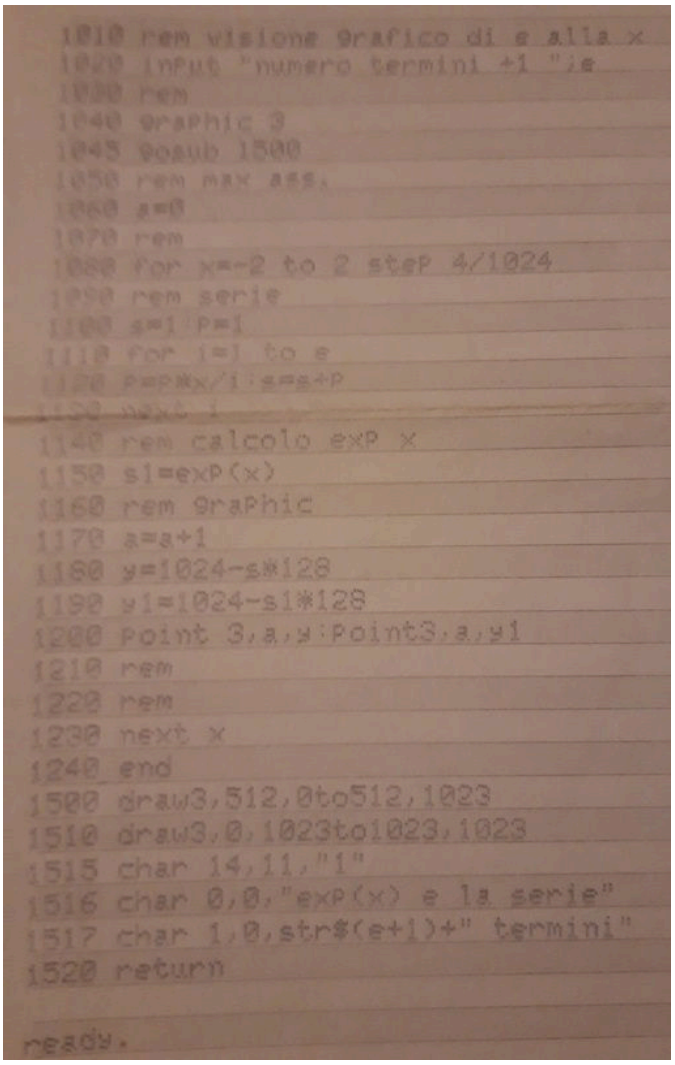


Figura 10



```
1010 rem visione grafico di e alla x
1020 input "numero termini +1 ";e
1030 rem
1040 graphic 3
1045 gosub 1500
1050 rem max ass.
1060 a=0
1070 rem
1080 for x=-2 to 2 step 4/1024
1090 rem serie
1100 s=1:p=1
1110 for i=1 to e
1120 p=p*x/i:s=s+p
1130 next i
1140 rem calcolo exp x
1150 s1=exp(x)
1160 rem graphic
1170 a=a+1
1180 y=1024-s*128
1190 y1=1024-s1*128
1200 point 3,a,y:point3,a,y1
1210 rem
1220 rem
1230 next x
1240 end
1500 draw3,512,0to512,1023
1510 draw3,0,1023to1023,1023
1515 char 14,11,"1"
1516 char 0,0,"exp(x) e la serie"
1517 char 1,0,str$(e+1)+" termini"
1520 return

ready.
```

Figura 11

imbattuto in un paio di deliziosi programmi per VIC 20 (espanso?) che ho copiato dal testo di Kosniowski (cfr. Bibliografia) e che potrebbero essere adattati anche per girare sul C64 (aggiustando POKE e PEEK). Non ho avuto la possibilità di verificare il funzionamento (e devo fidarmi) ma sembrano ben strutturati e alcune parti di codice potrebbero essere riutilizzate con profitto in altri programmi simili.

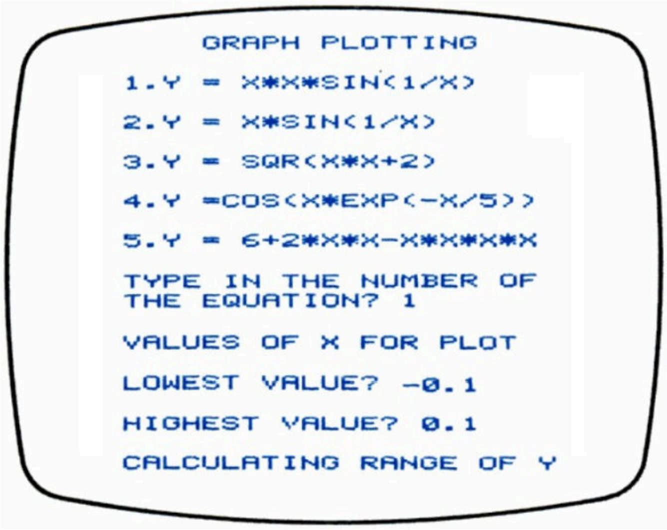
Chi non voleva impazzire con POKE e PEEK poteva ricorrere alla Super Expander Cartridge da mettere nello slot del VIC 20.

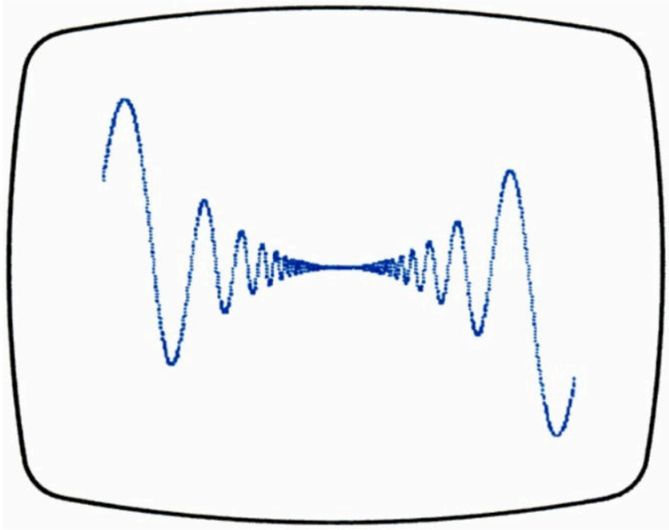
Un mio compagno di Scuola ne

aveva una che usava durante le ore di Laboratorio di Informatica. Anche il sottoscritto ne aveva usufruito per realizzare un piccolo programma che confrontava la funzione di libreria EXP(X) con un polinomio approssimante di Taylor della stessa funzione (Figura 11).

Conclusioni

Il prossimo articolo affronterà la rappresentazione grafica in 3D di funzioni matematiche z=f(x,y). Mi raccomando, non perdetelo!





```
10 REM                        ******************
20 REM                        *     VIC 20     *
30 REM                        * GRAPH PLOTTING *
40 REM                        *                *
50 REM                        ******************
60 REM
70 REM
100 REM %%%%%%%%%%%%%%%%%%%%%% SETTING UP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110 LET CS$=CHR$(147) : REM CODE FOR CLEAR SCREEN
120 LET SX=176 : REM SCREEN SIZE HORIZ
130 LET SY=160 : REM SCREEN SIZE VERT
140 LET HY=SY/2
145 GOSUB 1310 : REM EXTRA SETTINGS
150 PRINT CS$
160 PRINT "    GRAPH PLOTTING"
170 PRINT
180 PRINT "1.Y = X*X*SIN(1/X)"
190 PRINT
200 PRINT "2.Y = X*SIN(1/X)"
210 PRINT
220 PRINT "3.Y = SQR(X*X+2)"
230 PRINT
240 PRINT "4.Y =COS(X*EXP(-X/5))"
250 PRINT
260 PRINT "5.Y = 6+2*X*X-X*X*X*X"
270 PRINT
280 PRINT "TYPE IN THE NUMBER OF"
290 PRINT "THE EQUATION";
300 INPUT N
310 IF N=1 THEN DEF FNA(X)=X*X*SIN(1/X)
320 IF N=2 THEN DEF FNA(X)=X*SIN(1/X)
330 IF N=3 THEN DEF FNA(X)=SQR(X*X+2)
340 IF N=4 THEN DEF FNA(X)=COS(X*EXP(-X/5))
350 IF N=5 THEN DEF FNA(X)=6+2*X*X-X*X*X*X
360 PRINT
370 PRINT "VALUES OF X FOR PLOT"
```

SX and SY denote the number of points that can be plotted on the screen. Put in the values appropriate for your computer.

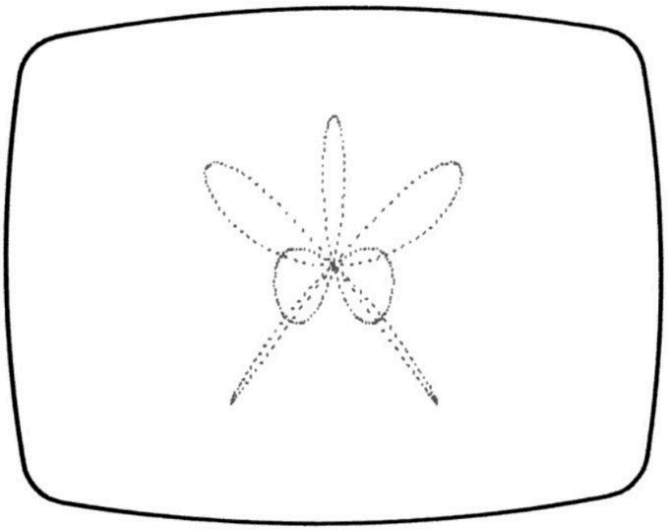
```
380 PRINT
390 PRINT "LOWEST VALUE";
400 INPUT A
410 PRINT
420 PRINT "HIGHEST VALUE";
430 INPUT B
440 PRINT
450 IF A>=B THEN PRINT "ERROR - TRY AGAIN"
460 IF A>=B THEN GOTO 360
500 REM %%%%%%%%%%%%%%%%%%%%%%% CALCULATING RANGE OF Y %%%%%%%%%%%%%%%%%%%%%%%
510 PRINT "CALCULATING RANGE OF Y"
520 LET C=(B-A)/100
530 LET M=1.0E-30
540 FOR X=A TO B STEP C
550 IF X=0 THEN GOTO 580
560 LET Y=ABS(FNA(X))
570 IF M<Y THEN LET M=Y
580 NEXT X
590 PRINT "READY FOR PLOTTING"
600 FOR I=1 TO 1000
610 NEXT I
620 PRINT CS$
630 GOSUB 1010 : REM PREPARE SCREEN
700 REM %%%%%%%%%%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
710 LET C=C/10 : REM TRY C/5 OR C/20
720 FOR X=A TO B STEP C
730 IF X=0 THEN GOTO 790
740 LET Y=FNA(X)
750 LET U=SX*(X-A)/(B-A)
760 LET V=HY+HY*Y/M
770 IF V<0 OR V>SY THEN GOTO 790
780 GOSUB 1110 : REM PLOT U,V
790 NEXT X
800 REM %%%%%%%%%%%%%%%%%%%%%%% ENDING AND ANOTHER GO %%%%%%%%%%%%%%%%%%%%%%%
810 GET G$ : REM LET G$=INKEY$
820 IF G$="" THEN GOTO 810
830 GOSUB 1210 : REM RESTORE SCREEN
840 PRINT CS$
850 PRINT " ANOTHER GO?  Y OR N"
860 GET G$ : REM LET G$=INKEY$
870 IF G$<>"Y" AND G$<>"N" THEN GOTO 860
880 IF G$="Y" THEN GOTO 150
890 END : REM STOP
1000 REM %%%%%%%%%%%%%%%%%%%%%%% PREPARE HI-RES SCREEN %%%%%%%%%%%%%%%%%%%%%%%
1010 POKE 36869,R8+12:POKE 36867,(PEEK(36867) AND 128) OR 21
1020 FOR I=RR TO SS:POKE I,0:NEXT I
1030 FOR I=0 TO 219:POKE PP+I,I-32*Q:POKE QQ+I,2:NEXT I
1040 RETURN
1100 REM %%%%%%%%%%%%%%%%%%%%%%% PLOT VIA POKE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1110 V=SY-V
1120 J=INT(U/8)
1130 L=INT(U-8*J)
1140 I=INT(V/16)
1150 K=INT(V-16*I)
1160 W=RR+I*352+J*16+K
1170 POKE W,PEEK(W) OR 2↑(7-L)
1180 RETURN
1200 REM %%%%%%%%%%%%%%%%%%%%%%% RESTORE SCREEN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1210 POKE 36869,R8:POKE 36867,R6:POKE 198,0
1220 RETURN
1300 REM %%%%%%%%%%%%%%%%%%%%%%% VIC 20 SETTINGS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1310 Q=PEEK(44)>=18:PP=7680+Q*3584:QQ=38400+Q*512
1320 IFQ=-1ANDPEEK(44)<32THEN PRINT "PROGRAM ABORTED - SEE APPENDIX":END
1330 RR=4096-Q*512:SS=RR+3583:R8=PEEK(36869):R6=PEEK(36867)
1340 RETURN
```

This part calculates the range of Y (approximately, to save time). The graph is then scaled to fill your screen.

This PLOTs the graph. A check is made to ensure that only the points that will fit on the screen are plotted.

Use the appropriate POKEs for your microcomputer.

```
      POLAR GRAPHICS

1.  R = 1

2.  R = SIN(2*Z)

3.  R = SIN(7*Z)

4.  R = 1+2*COS(Z)

5.  R = 1+COS(Z)

6.  R = 1+SIN(2*Z)

7.  R = 1+2*COS(2*Z)

TYPE IN THE NUMBER OF
THE EQUATION? 3

FOR STANDARD PLOT USE
A=1 AND B=1

VALUE OF A? 1

VALUE OF B? 3
```



```
10 REM                    ******************
20 REM                    *                *
30 REM                    * POLAR GRAPHICS *
40 REM                    *                *
50 REM                    ******************
60 REM
70 REM
100 REM %%%%%%%%%%%%%%%%%%%%%% SETTING UP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110 LET CS$=CHR$(147) : REM CODE FOR CLEAR SCREEN
120 LET SX=144 : REM SCREEN SIZE HORIZ
130 LET SY=176 : REM SCREEN SIZE VERT
140 LET RATIO=0.6 : REM TO MAKE HORIZ & VERT LINES SAME LENGTH
150 LET HY=SY/2
160 LET HX=SX/2
165 GOSUB 1310 : REM EXTRA VIC 20 SETTINGS
170 PRINT CS$ : REM CLS
180 PRINT "    POLAR GRAPHICS"
190 PRINT
200 PRINT "1. R = 1"
210 PRINT
220 PRINT "2. R = SIN(2*Z)"
230 PRINT
240 PRINT "3. R = SIN(7*Z)"
250 PRINT
260 PRINT "4. R = 1+2*COS(Z)"
270 PRINT
280 PRINT "5. R = 1+COS(Z)"
290 PRINT
300 PRINT "6. R = 1+SIN(2*Z)"
310 PRINT
320 PRINT "7. R = 1+2*COS(2*Z)"
330 PRINT
340 PRINT "TYPE IN THE NUMBER OF"
350 PRINT "THE EQUATION";
360 INPUT N
370 IF N=1 THEN DEF FNA(Z)=1
380 IF N=2 THEN DEF FNA(Z)=SIN(2*Z)
390 IF N=3 THEN DEF FNA(Z)=SIN(7*Z)
400 IF N=4 THEN DEF FNA(Z)=1+2*COS(Z)
410 IF N=5 THEN DEF FNA(Z)=1+COS(Z)
420 IF N=6 THEN DEF FNA(Z)=1+SIN(2*Z)
430 IF N=7 THEN DEF FNA(Z)=1+2*COS(2*Z)
440 PRINT
450 PRINT "FOR STANDARD PLOT USE A=1 AND B=1"
460 PRINT
470 PRINT "VALUE OF A";
480 INPUT A
490 PRINT
500 PRINT "VALUE OF B";
510 INPUT B
520 PRINT
```

SX and SY denote the number of points that can be plotted on the screen. Put in the values appropriate for your computer. The number RATIO is used to make a circle look like a circle. Try other values if Ø.6 does not work.

```
600 REM %%%%%%%%%%%%%%%%%%%%%%%% CALCULATING RANGE OF R %%%%%%%%%%%%%%%%%%%%%%%%
610 PRINT "CALCULATING RANGE OF R"
620 LET M=1.0E-30
630 FOR Z=0 TO 2*π STEP 0.1 : REM π=PI
640 LET R=ABS(FNA(Z))
650 IF M<R THEN LET M=R+0.1
660 NEXT Z
670 PRINT "READY FOR PLOTTING"
680 FOR I=1 TO 1000
690 NEXT I
700 PRINT CS$ : REM CLS
710 GOSUB 1010 : REM PREPARE SCREEN IF NECESSARY
800 REM %%%%%%%%%%%%%%%%%%%%%%%% PLOTTING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
810 FOR Z=0 TO 2*π STEP 0.01 : REM π=PI
820 LET R=FNA(Z)
830 LET U=HX+HY*RATIO*COS(A*Z)*R/M
840 IF U<0 OR U>SX THEN GOTO 880
850 LET V=HY+HY*SIN(B*Z)*R/M
860 IF V<0 OR V>SY THEN GOTO 880
870 GOSUB 1110 : REM PLOT U,V
880 NEXT Z
900 REM %%%%%%%%%%%%%%%%%%%%%%%% ENDING AND ANOTHER GO %%%%%%%%%%%%%%%%%%%%%%%%
910 GET G$ : REM LET G$=INKEY$
920 IF G$="" THEN GOTO 910
930 GOSUB 1210 : REM RESTORE SCREEN IF NECESSARY
940 PRINT CS$ : REM CLS
950 PRINT " ANOTHER GO?  Y OR N"
960 GET G$ : REM LET G$=INKEY$
970 IF G$<>"Y" AND G$<>"N" THEN GOTO 960
980 IF G$="Y" THEN GOTO 170
990 END : REM STOP
1000 REM %%%%%%%%%%%%%%%%%%%%%%%% PREPARE HI-RES SCREEN FOR VIC 20 %%%%%%%%%%%%
1010 POKE 36869,R8+12:POKE 36867,(PEEK(36867) AND 128) OR 25
1020 POKE 36866,R5-4:POKE 36864,R3+4
1030 FOR I=RR TO SS:POKE I,0:NEXT I
1040 FOR I=0 TO 219:POKE PP+I,I-32*Q:POKE QQ+I,2:NEXT I
1050 RETURN
1100 REM %%%%%%%%%%%%%%%%%%%%%%%% PLOT VIA POKE FOR VIC 20 %%%%%%%%%%%%%%%%%%%%
1110 V=SY-V
1120 J=INT(U/8)
1130 L=INT(U-8*J)
1140 I=INT(V/16)
1150 K=INT(V-16*I)
1160 W=RR+I*288+J*16+K
1170 POKE W,PEEK(W) OR 2↑(7-L)
1180 RETURN
1200 REM %%%%%%%%%%%%%%%%%%%%%%%% RESTORE SCREEN FOR VIC 20 %%%%%%%%%%%%%%%%%%%%
1210 POKE 36869,R8:POKE 36867,R6
1220 POKE 36866,R5:POKE 36864,R3:POKE 198,0
1230 RETURN
1300 REM %%%%%%%%%%%%%%%%%%%%%%%% VIC 20 SETTINGS %%%%%%%%%%%%%%%%%%%%%%%%%%%%
1310 Q=PEEK(44)>=18:PP=7680+Q*3584:QQ=38400+Q*512
1320 IFQ=-1ANDPEEK(44)<32 THEN PRINT"PROGRAM ABORTED - SEE APPENDIX":END
1330 RR=4096-Q*512:SS=RR+3583:R8=PEEK(36869):R6=PEEK(36867)
1340 R5=PEEK(36866):R3=PEEK(36864)
1350 RETURN
```

This part calculates the range of R (approximately, to save time). The graph is then scaled to fill your screen.

This PLOTs the graph. A check is made to ensure that only the points that will fit on the screen are plotted. Use PLOT or POKE as appropriate.

Use the appropriate POKEs for your microcomputer.

Bibliografia

[DC82a] T.Dwyer, M. Critchfield, "Il Basic e il personal computer uno:introduzione", Franco Muzzio & c. editore, 1982.

[DC82b] T.Dwyer, M. Critchfield, "Il Basic e il personal computer due:applicazioni", Franco Muzzio & c. editore, 1982.

[Got84] B.S. Gottfried, "Programmare in BASIC", ETAS Libri, 1984.

[JG82] M. James, S.M. Gee, "The Art of Programming the 1K ZX81", B. Babani Ltd, 1982.

[Kos83] C. Kosniowski, "Fun Mathematics on Your Microcomputer", Cambridge University Press, 1983.

Amstrad CPC Memory Display in Locomotive Basic

di Francesco Fiorentini

Continuo la mia avventura con il Locomotive Basic dell'Amstrad CPC, questa volta, dopo l'esperienza dei giochi in 10 righe, con un programma serio, una utility.

Mentre ero alla ricerca dell'ispirazione per un articolo per il numero 22 di RetroMagazine, mi sono imbattuto in un programma scritto diversi anni fa da AMSDOS, un utente del forum di CPC Wiki.
Come si evince dalla data scritta nei commenti nelle prime righe del codice, la versione originale del programma è stata scritta nel lontano 1997.

Dopo aver dato una rapida occhiata al codice mi sono subito reso conto che avrebbe fatto al caso mio e sarebbe stato un'ottima occasione anche per introdurre qualche nuovo concetto riguardante il Locomotive Basic.

Per chi fosse interessato, il codice originale si trova in questo thread sul forum di CPC Wiki:
https://www.cpcwiki.eu/forum/programming/memory-display/msg13921/#msg13921

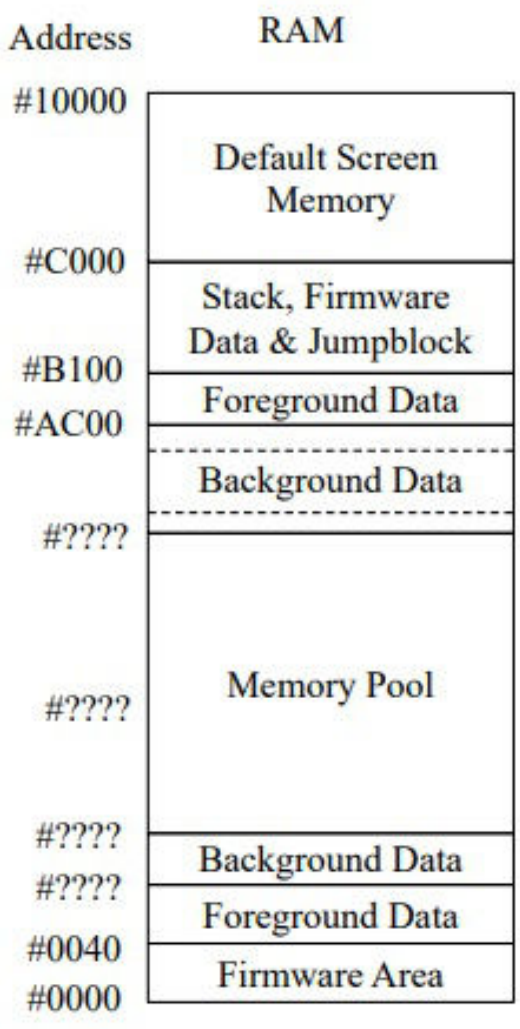


Figura 1: Memory Map dell'Amstrad CPC 464

Dopo aver copiato il codice sull'emulatore WinAPE ed aver dato il RUN, mi sono accorto che sì il programma funzionava perfettamente, ma anche che aveva bisogno di qualche aggiustatina per migliorarne la visibilità.

La versione originale infatti utilizzava la modalità grafica MODE 1, a 40 colonne, che personalmente reputo poco elegante, quindi la prima cosa che ho voluto fare è stata cambiarne l'aspetto grafico utilizzando la modalità MODE 2 ad 80 colonne.

Ovviamente questo cambiamente ha richiesto la modifica di tutto il resto dell'output video. In questo modo sono riuscito quindi a dividere lo schermo in due aree fisiche (in realtà sono 3, ma lo vedremo tra poco) ed a separare visivamente l'output del programma dall'area di interazione con l'utente.

Il comando WINDOW

Come ho già avuto modo di ribadire diverse volte, il Locomotive Basic è un linguaggio piuttosto potente ed il comando WINDOW ne è una chiara dimostrazione.
Vediamo qual è la sintassi di questo comando:

```
WINDOW[#‹stream expression›],[L],[R],[T],[B]
```

#STREAM: è possibile definire fino ad 8 fineste indipendenti che possono anche essere sovrapposte. La finestra numero 0 è la finestra principale dove messaggi di errore o di stato vengono visualizzati di default.
L = left column - colonna di sinistra
R = right column - colonna di destra (dipende dal modo grafico: 0 = 20 / 1 = 40 / 2 = 80)
T = top row - riga iniziale (sempre tra 1 to 25)
B = bottom row - riga finale (sempre tra 1 to 25)

La funzione WINDOW, come forse avrete già intuito, permette di creare una finestra virtuale all'interno dell'area video dove i comandi PRINT, INPUT... possono, tramite lo STREAM, interagire.

Come facilmente intuibile questo comando è molto potente perché tutte le istruzioni che vi verranno indirizzate tramite

Figura 2: Il programma in esecuzione dalla locazione 0000

lo STREAM, avranno effetto soltanto nella finestra indicata e vi rimarranno confinate. Per esempio, definendo una finestra di 10 righe, lo scrolling avverrà entro questo limite, senza impattare il resto dello schermo. Molto utile nel caso volessimo suddividere lo schermo in aree logiche.

La versione originale faceva già uso delle finestre per separare l'output del programma, vediamo quindi come sono state adattate nella nuova versione.

La prima finestra, composta da 3 righe ed identificata con lo stream #1, viene utilizzata per stampare a video l'intestazione del programma.
Si vedano le righe da 110 a 130.

La seconda finestra, composta invece da 22 righe ed indentificata dalla stream #2, viene utilizzata per stampare l'output del programma. Come potrete facilmente notare eseguendo il programma, lo scrolling della seconda finestra è totalmente seprato dalla prima che rimane invece fissa.
Si veda la riga 180.

La terza ed ultima finestra, identificata dallo stream #3 è invece interamente dedicata all'interazione con l'utente.
In questa finestra vengono richiesti i parametri da utilizzare per visualizzare la memoria (da quale indirizzo iniziare e quanti indirizzi visualizzare) e stampati i messaggi generati dal programma.

Si vedano le righe da 191 a 195.

Vorrei inoltre farvi notare come tutti i comandi che stampano a video, PRINT, INPUT... abbiano la possibilità di gestire lo stream dove redirigerne l'ouptut.

Il resto del programma

Il resto del programma è piuttosto semplice, in pratica tramite la funzione PEEK(n) andiamo a leggere il contenuto dell'indirizzo di memoria interessato e ne facciamo un dump in formato esadecimale (con annessa visualizzazione del carattere), decimale e binario.

Da notare che il Locomotive Basic ha anche a disposizione la funzione BIN$() che ci permette di risparmiare tempo nel convertire un intero in binario. Molto comodo.

BIN$ (<unsigned integer expression>,[<integer expression>])

Produce una stringa in formato binario rappresentante il valore di **<unsigned integer expression>** utilizzando il numero di caratteri binari come indicato da **<integer expression>** (in un range da 0 a 16).
Il valore <unsigned integer expression> deve essere compreso nel range: -32768 to 65535.

Un altra paio di particolarità, piuttosto semplici, da notare nel programma sono le righe 190 e 196.

Figura 3: Da #170 (368 decimale) inizia l'area dedicata al Basic

La riga 190 effettua un controllo sull'indirizzo di memoria da leggere, se questo dovesse eccedere il valore #FFFF (65535), il programma deve essere interrotto perché altrimenti genererebbe un errore. Il CPC 464 (oggetto di questo test) indirizza fino a 64K di RAM. Nel caso volessimo testare lo stesso programma sul CPC 6128, dovremmo modificare questa riga con il valore 131072 ovvero 128K. La riga 196 invece è fondamentale per ripristinare l'aspetto dell'interprete Basic prima della fine del programma. Da notare che ho intenzionalmente lasciato la modalità ad 80 colonne in quanto la trovo più piacevole per gli occhi.

Perché controllare il contenuto della memoria?

A questo punto però credo che vi starete facendo una domanda, perché controllare il contenuto della memoria? Beh, intanto perchè è figo... Ok, scherzo...

Attualmente questo genere di programmi sono piuttosto fini a se stessi, in quanto tutte le locazioni di memoria dei computer sono conosciute e mappate. In passato invece non era così e programmi del genere servivano agli smanettoni per capire come e dove le informazioni venissero memorizzate nella RAM del computer per poi poterne usufruire nei loro programmi.

Dove trovo adesso queste informazioni?

Oggigiorno possiamo trovare queste informazioni con una veloce ricerca su internet, mentre negli anni 80 e 90 era più difficile.

Il caso dell'Amstrad CPC comunque era già un'eccezione per quegli anni, in quanto gran parte della documentazione fu resa disponibile al momento del lancio.

In questo libro potete trovare la documentazione della memoria del CPC464:

http://cpctech.cpc-live.com/docs/manual/s968se02.pdf

Provate a vedere cosa trovate dalla locazione 368 decimale (170 esadecimale) in poi...

Buon divertimento!

NB: Quando copiate il programma fate attenzione agli spazi nella riga 120; 2 o più spazi potrebbero venire percepiti come 1...

```
10 '*********************************************
11 '*  Memory Display
20 '*  Public Domain - CP/M User
30 '*  Written by AMSDOS 27/4/1997
40 '*  Modified by Francesco Fiorentini 29/3/2020
42 '*********************************************
45 row=0
60 MODE 2:WINDOW 39,40,1,25:INK 3,2:INK 0,0:PAPER 3:CLS:BORDER 2
70 WINDOW#1,1,39,1,3:WINDOW#2,1,39,4,25:WINDOW#3,40,80,1,25:PAPER#1,0:INK 3,2
80 row=row+1:LOCATE#3,3,row:INPUT#3,"Enter Start Address ";start
90 row=row+1:LOCATE#3,3,row:INPUT#3,"Enter Length ";length
100 CLS#1:CLS#2:PAPER#2,0
101 REM WINDOW#1 è utilizzata per l'intestazione
110 PRINT
#1,CHR$(150);STRING$(6,CHR$(154));CHR$(158);STRING$(10,CHR$(154));CHR$(158);STRING$(6,CHR$(154));CHR$(158);STRING$(10,CHR$(154
));CHR$(156)
115 REM Attenzione agli spazi nella stringa sotto...
120 PRINT #1,CHR$(149);" Addr ";CHR$(149);" Hex Chr  ";CHR$(149);" Dec  ";CHR$(149)"  Binary  ";CHR$(149);CHR$(13)
130 PRINT
#1,CHR$(147);STRING$(6,CHR$(154));CHR$(155);STRING$(10,CHR$(154));CHR$(155);STRING$(6,CHR$(154));CHR$(155);STRING$(10,CHR$(154
))CHR$(153)
131 REM disassemblaggio delle memoria
140 length=length+start
150 FOR pr=start TO length
160 dis=PEEK(start)
170 char$=CHR$(1)+CHR$(dis)
179 REM WINDOW#2 è utilizzata per la stampa dei valori a video
180 PRINT #2,TAB(3);HEX$(start);TAB(11);HEX$(PEEK(pr),2);TAB(15);char$;TAB(20);PEEK(pr);TAB(28);BIN$(PEEK(pr),8)
190 start=start+1:if start > 65535 then goto 191 else NEXT pr
191 row=row+1:PAPER #2,0:LOCATE#3,3,row:PRINT#3,"Memory Map terminated."
192 row=row+1:PAPER #2,0:LOCATE#3,3,row:PRINT#3,"Continue to map? y/n"
193 answ$=INKEY$:IF answ$<>"Y" AND answ$<>"N" AND answ$<>"y" AND answ$<>"n" THEN 193
194 IF answ$ = "Y" or answ$ = "y" THEN GOTO 80
195 row=row+1:PAPER #2,0:LOCATE#3,3,row:PRINT#3,"Goodbye...":for i= 1 to 1000:next i
196 WINDOW 1,40,1,25:LOCATE 5,25:PAPER 0:MODE 2:BORDER 0:END
```

Listato del programma Memory Display

Memory Dump per Vic-20

di David La Monaca (Cercamon)

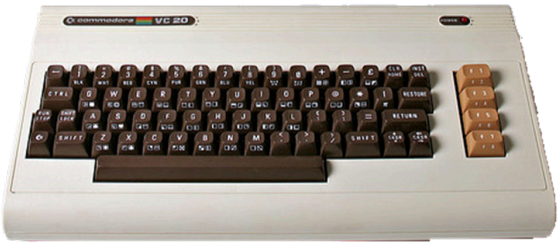
Un po' di storia (non guasta mai)

Non c'è dubbio su quale sia il computer di maggior successo della sventurata Commodore. A dispetto delle indubbie qualità tecniche della serie Amiga – il parere pressoché unanime è che, quando fu immessa nel mercato, fosse di 5-6 anni avanti alla concorrenza – e di altri modelli che ebbero una buona diffusione, come ad esempio la serie PET dedicata al mondo business, il Commodore 64 rappresenta la più grande affermazione sul mercato con milioni di esemplari venduti in tutto il mondo. Tralasciando e stendendo un velo più che pietoso sulle mirabolanti idee della casa madre che portarono alla produzione di Commodore 16, Plus/4 e, ahimé, Commodore 128 – una macchina strozzata sul mercato dalla stessa Commodore che quasi in contemporanea apriva il mercato a 16 bit con Amiga – non dobbiamo dimenticare il VIC-20, un home computer che nel suo piccolo – è il caso di dirlo – riscosse un ottimo successo in termini di unità vendute e di utilizzo videoludico fra il 1981 ed il 1983, una finestra temporale breve ma intensa, terminata con l'avvento del fratello maggiore, destinato a spazzare via ogni tipo di concorrenza.

I motivi sono presto detti: basso costo, CPU MOS 6502 a 1 MHz, chip grafico VIC-I dedicato a 16 colori, tastiera e case compatti e semi-professionali, cartucce gioco disponibili a costi accessibili, BASIC integrato, sonoro di qualità, collegamento diretto alla TV, ecc. In pratica, all'epoca, una perfetta via di mezzo tra console per videogiochi e computer programmabile. Molte le pecche riscontrabili nel progetto della macchina, col senno di poi, naturalmente: schermo a caratteri 23 righe x 22 colonne, una gestione della memoria RAM e ROM abbastanza fantasiosa, una sola porta joystick e soprattutto pochissima RAM a disposizione dell'utente, appena 3.5 KB nella versione base. Una miseria. Ma all'epoca i chip di memoria RAM erano fra i componenti più costosi in assoluto e Commodore, rivolgendosi al nascente mercato consumer casalingo, faceva del prezzo basso un elemento imprescindibile della sua politica commerciale.

Programmare su VIC-20

Come accennato, la programmazione in BASIC o in linguaggio macchina, almeno nella versione di serie, era tutt'altro che agevole, sia per l'esigua dimensione dello schermo sia per la cronica mancanza di memoria RAM disponibile per inserire il codice. Ma, come anche le carenze del C64 nel BASIC, portarono molti utenti a scoprire le meraviglie del linguaggio macchina, anche per il VIC-20 programmatori professionisti ed hobbysti furono costretti a giocare con la creatività tecnica e



Il Commodore ViC-20

riuscirono nella non facile impresa di produrre giochi e programmi più che rispettabili persino per la versione inespansa.

Con i suoi oltre 2 milioni e mezzo di esemplari venduti, non è difficilissimo ancor oggi, trovare un VIC-20 funzionante in molti paesi d'Europa, Italia compresa, dove ebbe una buona diffusione. Programmare su una macchina così limitata comporta, paradossalmente, qualche vantaggio: il livello di complessità dell'hardware è basso e si ha tutto sotto controllo; scrivere il codice diventa un ottimo esercizio di ottimizzazione; il debug risulta piuttosto semplice; veder girare il frutto delle proprie fatiche regala molta soddisfazione.

Perché proprio un programma Memory Dump?

Ed è proprio per queste ragioni che nelle mie scorribande nell'assembly del 6502/6510, che di tanto in tanto compio per (re)imparare a programmare e che purtroppo risultano sempre più frammentate nel tempo, ho voluto includere una piccola sfida con me stesso, scaturita da un post su Facebook, nel gruppo Commodore Vic-20 in lingua inglese. Uno dei membri del gruppo si era cimentato nella scrittura di un semplice programma BASIC per visualizzare in modo creativo il contenuto di porzioni della memoria. Nulla di che, ma lo screenshot che accompagnava il post attirò la mia attenzione per il fatto che il programma mostrava il contenuto di celle di memoria arrivando ad una semplice rappresentazione grafica dei bit di ciascun byte esaminato. Usando la tabella PETSCII del VIC-20 il programma mostrava un pallino vuoto per rappresentare un bit a zero, un pallino pieno per i bit a 1. Il programmatore, inoltre, si lamentava della scarsa velocità d'esecuzione del programma, chiedendo aiuto per ottimizzare la piccola routine BASIC che aveva messo insieme. Così mi sono detto: questo è un ottimo campo d'allenamento per ripassare un po' di aritmetica binaria, rappresentazione dei numeri in decimale ed esadecimale, conversioni da una base all'altra, ecc. E poi volevo vedere di quanto il linguaggio macchina poteva distanziare il BASIC in una semplice routine come questa. Il risultato è stato molto soddisfacente e soprattutto il divertimento nel programmare e nel vedere questo piccolo programma

girare su un vero VIC-20 è stato all'altezza delle aspettative.

Listato, commenti e curiosità

Nel Listato 1 troverete il codice completo commentato del programmino. Per chi mastica l'assembly per 6502/6510 non dovrebbe esserci alcun problema a seguire il flusso del codice. Per tutti gli altri, ecco di seguito qualche nota che spero torni utile.

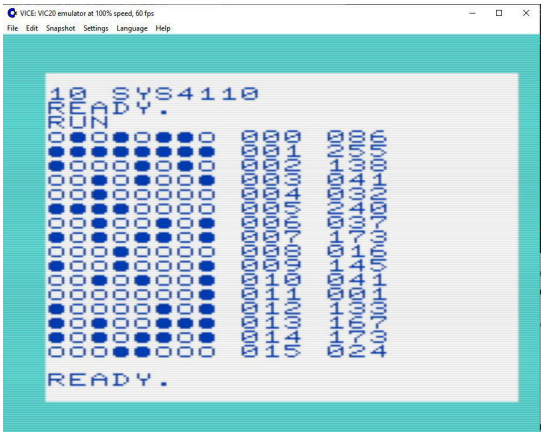
Per assemblare il codice ho usato CBM Prg Studio, ormai un punto di riferimento per la comunità retrocoding, insieme a Kick Assembler, C64 Studio e qualche altro compilatore con sintassi simil-Turbo Assembler. L'ho scelto per la semplicità d'uso e per la possibilità di automatizzare il processo di compilazione ed esecuzione del codice su emulatore (XVIC dell'arcinota suite di emulazione dei Commodore a 8-bit VICE).

Le etichette CHROUT e MEM_START, NUM_BYTES poste prima dell'inizio del codice servono a indirizzare la routine di stampa del KERNAL standard del VIC-20 (presente in $FFD2) e a inizializzare due variabili usate nel codice come prima locazione di memoria da esaminare e offset di byte da visualizzare. Nel listato (ad es. s'intende visualizzare le prime 50 locazioni memoria a partire dalla locazione $E000 del VIC-20).

Il programma è stato dotato di una riga BASIC di start ("10 SYS 4110") per semplificare l'esecuzione del programma. Basta un RUN dopo il caricamento standard del file PRG prodotto dall'assembler. La riga di BASIC viene posta alla locazione $1001 (4097), quella di default per i VIC-20 di serie, cioè senza espansione di memoria.

Alla locazione $100E (4110) inizia il codice vero e proprio del programma. La riga BASIC di lancio SYS 4110 esegue per l'appunto le istruzioni in linguaggio macchina a partire da questa locazione.

Il ciclo **dumpLoop** è quello principale del programma. Serve a esaminare via via tutte le locazioni di memoria a partire da MEM_START fino MEM_START + NUM_BYTES, prelevare il contenuto di ogni singolo byte e passarlo al ciclo più interno che effettua l'esame dei singoli bit da 0 a 7 di ciascun byte visualizzandone il valore in forma binaria e grafica.



Memory Dump in azione (1)

Come detto, pallino vuoto per lo 0, pallino pieno per l'1.

Il ciclo **bitLoop** è quello che consente al programma di esaminare ogni bit del byte che si sta esaminando e stamparne il valore sotto forma semi-grafica, usando i codici PETSCII di un pallino vuoto ($D7) o di un pallino pieno ($D1). Come effettuare il controllo di ogni singolo bit? Usiamo l'istruzione **ASL** che effettua uno spostamento a sinistra di tutti i bit del byte corrente contenuto nell'accumulatore A, facendo cadere, per così dire, il bit più a sinistra nel **bit di carry** del registro di stato. Esaminano poi il bit di carry con l'istruzione di salto condizionato **BCC** si ottiene il responso: uno 0 oppure un 1. Poi basta ripetere il processo per tutti e 7 i bit del byte in esame per visualizzarli e passare alla locazione di memoria successiva.

Dopo aver stampato il valore binario di una locazione in forma grafica, per rendere più leggibile il tutto, stampiamo in sequenza e sulla stessa riga il valore dell'offset e infine il valore contenuto nella locazione, entrambi in forma decimale.

Ma dal momento che entrambi questi valori sono in forma esadecimale nel momento in cui li andiamo a prelevare dalla memoria, allora usiamo una funzione di conversione (**convToStr** nel listato) che, sfruttando la sequenza della tabella PETSCII contenuta nella ROM dei caratteri del VIC-20, ci consente di fornire in ingresso un byte nell'accumulatore (A) e di ottenere in uscita le tre cifre (centinaia, decine ed unità) rispettivamente nei registri Y, X ed A stesso. Passando poi opportunamente e in ordine i valori di Y, X ed A alla routine di stampa di un singolo carattere CHROUT otterremo la stampa a video dei valori di offset e del contenuto della locazione di memoria.

Una curiosità: confrontando il carattere pallino vuoto ($D7 o 215 nella tabella PETSCII) del VIC-20 con l'analogo della ROM dei caratteri del C64, ho notato una differenza nel disegno della griglia 8x8. Il primo è un cerchietto mentre il secondo sembra più stretto, come un anello (vedi figure).

Conclusioni

Ovviamente questo articolo vuole soltanto mostrare quello che è poco più di un esercizio svolto in assembly e, ovviamente, suscettibile di ulteriori miglioramenti e di ottimizzazione del codice. È dedicato ai principianti e a chi si avvicina con curiosità e comprensibile circospezione al linguaggio assembly. Serve però anche a dimostrare come spesso sia necessario costruire da soli funzioni e comandi che nei linguaggi di alto livello siamo abituati a dare per scontati. Come spesso ricordato da tutti i manuali di assembly, la maggior fatica e il maggior tempo da concedere allo sviluppo del codice viene ricompensato da una velocità d'esecuzione che può arrivare ad essere anche 50 o 100 volte superiore a quella di un linguaggio interpretato. Inoltre, la possibilità di gestire

direttamente ogni singola locazione di memoria e di accedere all'hardware e alle sue funzioni fondamentali, oltre a fornire una fonte continua di spunti didattici, ripaga ampiamente il programmatore, anche in programmi semplici e tutto sommato lineari come questo. Ed è proprio grazie a questa piccola sfida che ho potuto parlare per la prima volta su RetroMagazine di questa affascinante e divertente macchina, perfetta per scopi di intrattenimento ed educativi.

Listato 1 – Memory Dump per VIC-20

```
; MEMORY DUMP per VIC-20
;
; Effettua un dump della memoria a partire da una locazione (MEM_START)
; prestabilita e per un certo numero di byte (offset NUM_BYTES)
; Per ogni byte mostra una rappresentazione grafica dei bit da 0 a 7
; con valore del byte e numero di offset in decimale
;
; esempio:
; bits      offset val
; 10010101  000    149
; 01100001  001    097
; 10010100  002    148
; 01101001  003    105
; 11001010  004    202
; ecc.
;

CHROUT    = $ffd2

MEM_START = $e000
NUM_BYTES = 50


              * = $1001       ; inizio BASIC 4097 - SYS call line 10 SYS 4110
              byte $0B,$10,$0A,$00,$9E,$34,$31,$31,$30,$00,$00,$00

              * = $100E       ; start 4110 = $100E

              ldx #0          ; registro X indice del byte corrente
dumpLoop      lda MEM_START,x ; carica in A il byte da analizzare

              sta valAcc      ; salva valore di A per stamparlo dopo i bit

              ldy #0          ; registro Y indice dei bit 0-7

bitLoop       asl             ; shift a SX dei bit - bit 7 nel carry
              pha             ; salva il valore corrente di A nello stack

              bcc printZero   ; se carry = 0 vai a stampare "0"

printOne      lda #$D1        ; carry = 1 - stampa pallino pieno o "1"
              jsr CHROUT
              jmp there       ; "1" stampato - salta al prossimo bit

printZero     lda #$D7        ; carattere pallino vuoto o "0"
              jsr CHROUT

there         pla             ; recupera il valore di A dallo stack

              iny             ; prossimo bit
              cpy #8          ; ultimo bit?
              bne bitLoop     ; no - analizza il prossimo bit

              lda #$20        ; stampa uno spazio
              jsr CHROUT
              txa             ; trasferisce X in A
```

```
                jsr convToStr   ; e lo stampa in decimale
                lda #$20        ; stampa uno spazio
                jsr CHROUT
                lda valAcc      ; stampa valore di A salvato in precedenza
                jsr convToStr   ; in formato decimale
                lda #$0d        ; stampa CR (13) dopo 8 bit
                jsr CHROUT      ; va a capo

                inx             ; prossimo byte
                cpx #NUM_BYTES+1 ; numero di byte da scandire finiti?
                bne dumpLoop    ; no - vai al prossimo byte
                rts             ; sì, termina il programma

; ---
; --- subroutine di conversione da byte (in A) a cifre decimali
; --- restituisce le centinaia in Y (0-9), le decine in X (0-9)
; --- e le unità in A (0-9)
; --- ad es.: Y=0, X=5, A=9 stampa 059, valore passato in A uguale a 59
; ---
convToStr       sta tmpAcc      ; salva A in locazione temporanea

                txa
                pha             ; salva X nello stack
                tya
                pha             ; salva Y nello stack

                lda tmpAcc      ; carica A con il valore salvato in temp
                ldy #$2f        ; routine di conversione
                ldx #$3a        ; fa uso della tabella ASCII/PETSCII
                sec
l0              iny
                sbc #$64        ; #100 - mette la cifra delle centinaia in Y
                bcs l0
l1              dex             ; mette la cifra delle decine in X
                adc #$0a        ; #10 - e quella delle unità in A
                bmi l1
                adc #$2f
                                ; a questo punto i registri Y, X ed A contengono
                                ; le centinaia (Y), le decine (X) e le unità (A)
                                ; del valore di A passato alla routine di conv.

                pha             ; stampa in sequenza Y, X ed A - prima salva A
                tya             ; trasferisce Y in A
                jsr CHROUT      ; e lo stampa
                txa             ; poi trasferisce X in A
                jsr CHROUT      ; e lo stampa
                pla             ; recupera A dallo stack
                jsr CHROUT      ; e lo stampa

                pla             ; recupera Y dallo stack
                tay
                pla             ; recupera X dallo stack
                tax

                rts             ; ritorna - fine subroutine convToStr

; ---
; --- locazioni di memoria di appoggio / scambio
; ---
valAcc          byte 0     ; valore dell'accumulatore/byte di memoria corrente
tmpAcc          byte 0     ; valore temporaneo di A
```

207	CF	O	
208	D0	P	
209	D1	Q	
210	D2	R	
211	D3	S	
212	D4	T	
213	D5	U	
214	D6	V	
215	D7	W	
216	D8	X	
217	D9	Y	

Particolare Petscii table C64

207	CF	79	
208	D0	80	
209	D1	81	
210	D2	82	
211	D3	83	
212	D4	84	
213	D5	85	
214	D6	86	
215	D7	87	
216	D8	88	
217	D9	89	

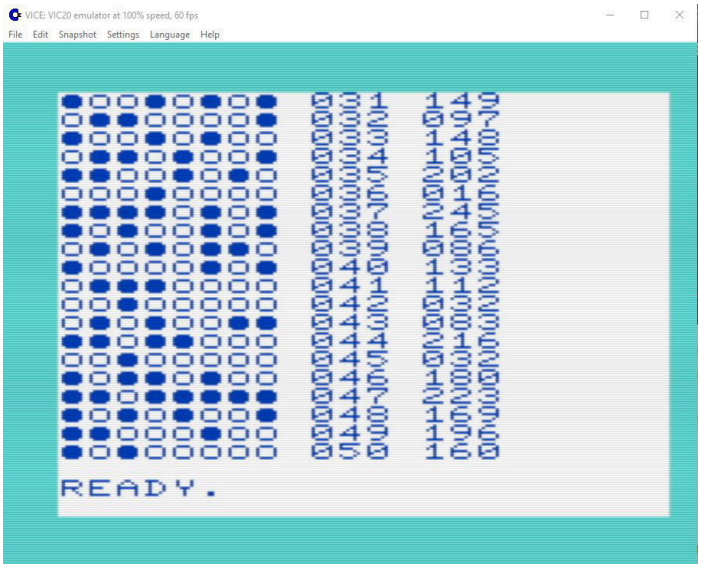
Particolare Petscii table Vic20

Riferimenti WEB

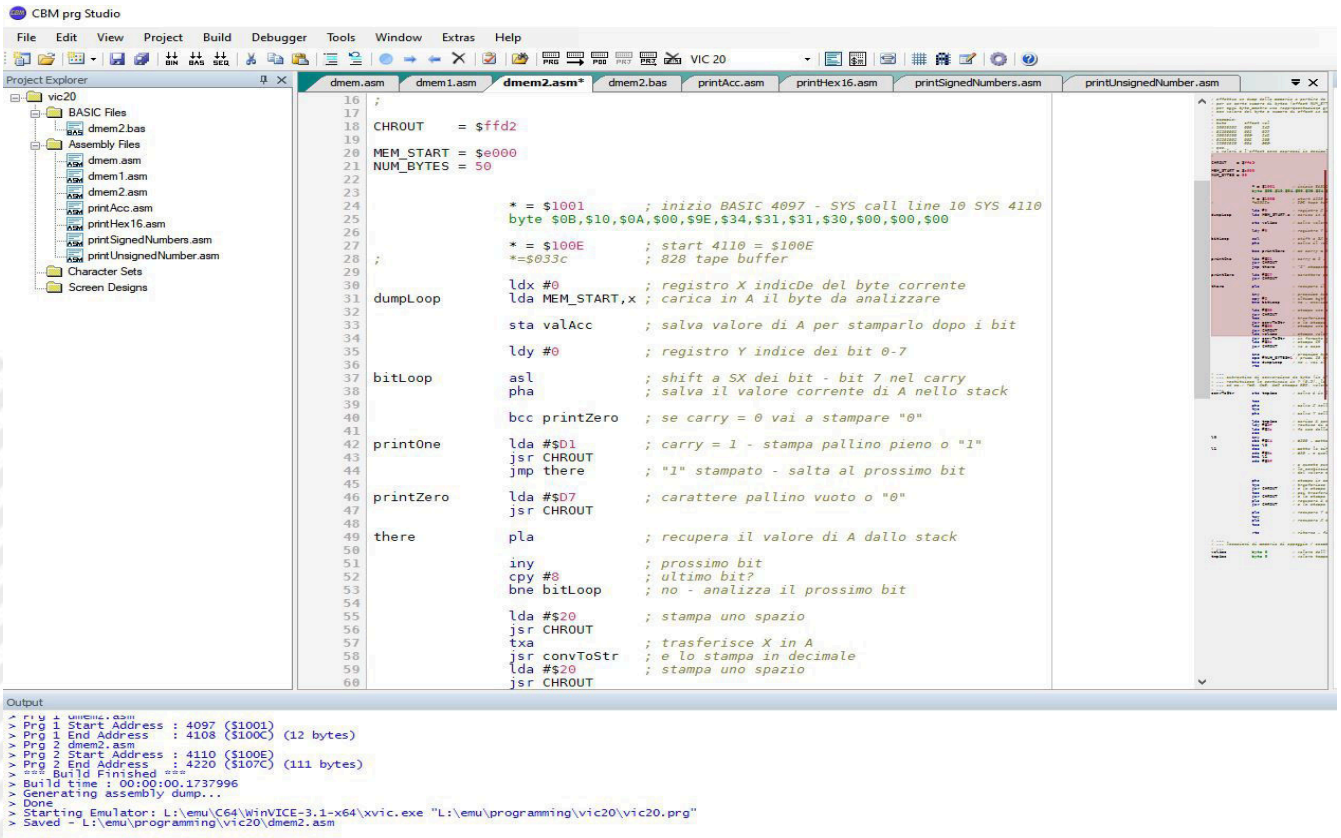
• CBM PRG Studio
https://www.ajordison.co.uk

• VICE
https://vice-emu.sourceforge.io

• VIC-20 Denial Forum
http://sleepingelephant.com/ipw-web/bulletin/bb/



Memory Dump in azione (2)



Il tool di sviluppo CBM Prg Studio in azione

Introduzione ad ARexx – seconda parte

di Gianluca Girelli

Questo articolo è comparso la prima volta sulle pagine di Bitplane nel Settembre del 2012.

Dopo l'antipasto del tutorial precedente, pubblicato su numero 20 di RM e che ci ha portato ad esplorare le basi del linguaggio, iniziamo ad analizzare con maggiore profondita' la potenza dello scripting di ARexx. Come ampiamente illustrato, cio' che determino' il successo di REXX (e quindi di ARexx) fu il fatto di poter essere veramente un collante tra le piu' diverse applicazioni e, poiche' il tutto si fonda su efficaci comunicazioni inter-processo, e' necessario disporre di costrutti in grado di semplificare e formattare i flussi di input/output per arrivare ad una veloce ed efficiente validazione del testo. Ecco perche' la gestione delle stringhe, cioe' delle sequenze alfanumeriche di caratteri letti in input o prodotti in output, e' uno dei punti di forza di ARexx.
Ai fini della nostra trattazione intodurremo quindi ora alcune istruzioni del linguaggio relative alla gestione delle stringhe che ci permetteranno di creare un (assolutamente minimale) "parser sintattico" (vedi box).

Parser sintattico

In linguistica ed in informatica il termine "parsing" o, piu' formalmente, "analisi sintattica", e' il processo di analizzare una sequenza di simboli (tokens) o parole per determinarne la congruenza della struttura grammaticale in relazione ad una data "grammatica formale". Piu' nel particolare, un parser e' uno dei componenti di un "interprete" (o di un "compilatore") e serve per controllare la correttezza sintattica del linguaggio e per costruirne le strutture dati.

ARexx e le stringhe

L'assoluta importanza data alle stringhe dai creatori e dai primi utilizzatori del linguaggio traspare chiaramente dalla quantita' e dalla qualita' delle istruzioni utilizzabili per gestirle. In ARexx possiamo disporre, ad esempio, di costrutti in grado di:
- editare: "compress(str, [lista])" comprime la stringa rimuovendo da "str" i caratteri riportati nella "lista". Altre istruzioni si occupano di cancellare, inserire o sovrapporre stringhe o parti di esse;
- comparare: "compare(str1, str2)" determina la posizione a partire dalla quale le due stringhe differiscono. Altre istruzioni in questa "classe" controllano se una stringa e' una abbreviazione di un'altra o se una stringa e' contenuta in un'altra;
- formattare: estrarre una stringa da un'altra. Appartengono a questa categoria le istruzioni "left" e "right" riportate negli esempi seguenti;
- lavorare direttamente sulle parole: "words(str)" determina il numero di parole contenute in "str", "wordindex(str,n)" determina la posizione da cui parte l'n-esima parola in "str" mentre "word(str,n)" estrae la parola "n" dalla stringa "str".

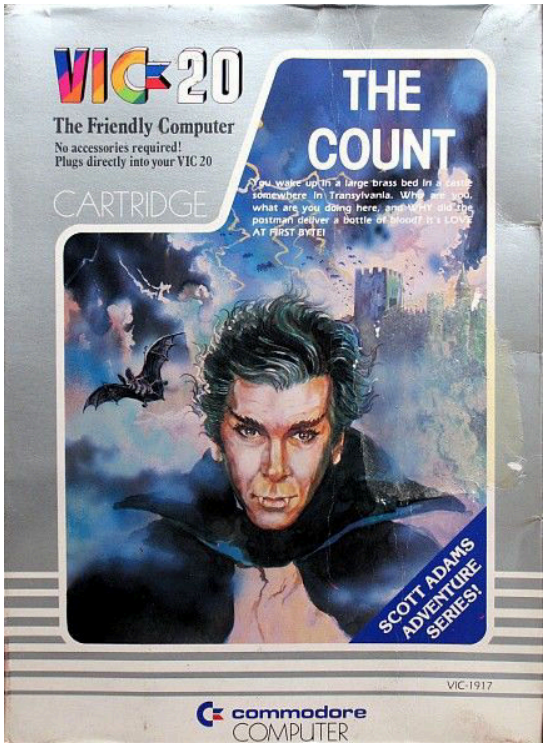


Figura 1: The Count per VIC20

Esiste poi un comando particolare,"parse", il quale dispone di tanti e tali argomenti da poter effettuare da solo molte delle funzioni appena accennate mediante l'uso di "modelli" (templates) definibili dall'utente. In tal modo si agevola tremendamente il consueto meccanismo che prevede la lettura della sequenza alfanumerica di caratteri e la sua successiva elaborazione e scomposizione nelle parti "significative" della frase.
In totale, il manuale "Using ARexx on the Amiga" citato nell'articolo precedente, riporta oltre trenta funzioni disponibili solo per gestire efficacemente le stringhe.

Cominciamo ora con alcuni esempi (istruzioni nel box) per chiarire meglio quanto appena espresso. In questi esempi supponiamo per semplicita' di dover elaborare una stringa che sappiamo gia' essere composta da due

Utilizzo degli esempi

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

parole (vedremo poi il perche').

```
 /* esempio 1 */
pull frase
verbo=left(frase,index(frase,' ')-1)
nome=right(frase,length(frase)-
index(frase,' '))
```

Nell'esempio dato, dopo aver acquisito in input la nostra stringa con l'istruzione "pull" ed averla memorizzata nella variabile "frase", la suddividiamo assegnando la prima parte della stringa ("left") alla variabile "verbo" e la seconda ("right") alla variabile "nome". La divisione viene fatta assumendo che le due parole siano separate da uno spazio, la cui posizione nella stringa viene determinata grazie all'istruzione "index". Se la stringa dunque e' "prendi corda", le nostre due variabili assumeranno i valori di:

`verbo="PRENDI" e nome="CORDA".`

Ricordiamo, infatti, che l'istruzione "pull" e' l'abbreviazione di "parse upper pull" e converte automaticamente la stringa in maiuscolo.
Lo stesso identico risultato puo' essere ottenuto in modo molto piu' semplice usando un "template":

```
/* esempio 2 */
parse upper pull verbo ' ' nome
```

dove il template e' costituito dalla sequenza: "variabile1 ' ' variabile2". Notare che le due variabili (nel codice chiamate direttamente "verbo" e "nome") sono separate da uno spazio racchiuso tra apici singoli che, in fase di esecuzione, si occupera' di rimuovere lo spazio che invece separa le due parole all'interno della stringa.

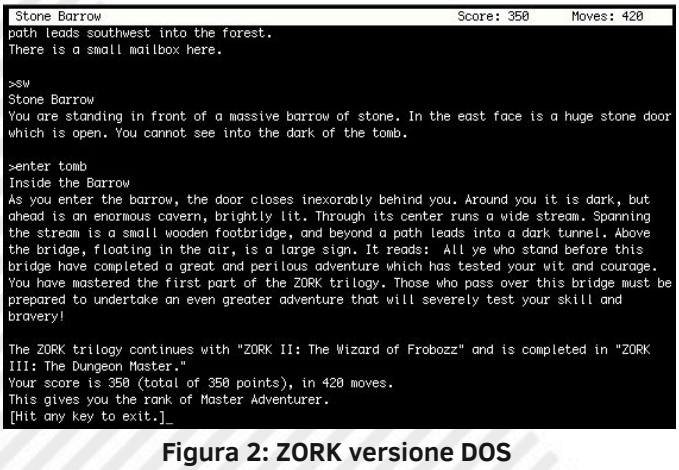


Figura 2: ZORK versione DOS

Un modo piu' sofisticato di estrarre gli spazi di troppo da una stringa senza usare la sequenza di singoli apici appena citata e' quello di combinare la parola chiave "parse" con l'argomento "var". Questo argomento specifica che la parola successiva a "var" e' una variabile e che tutte le altre parole sono invece il "template" da utilizzare per suddividere la stringa. Poiche' ad ogni parola estratta, tranne l'ultima rimanente, vengono rimossi gli spazi bianchi precedenti e seguenti la parola stessa, costruiamo il nostro modello facendo credere al programma che la stringa contenga una parola in piu' di quanto in realta' non sia.

```
/* esempio 3 */
pull frase
parse upper var frase verbo frase resto
nome = frase
```

Quello che succede in questo caso e' che la nostra frase viene letta in input ed inserita nella variabile "frase" ("pull frase"). Successivamente viene convertita in maiuscolo ("parse upper") e la prima parola viene estratta e assegnata alla variabile "verbo". Poiche' "frase" e' anche il secondo parametro del modello, essa conterrebbe ora tutto il resto della stringa, incluso lo spazio che separa le due parole. Inserendo una terza variabile (a perdere), anche alla seconda parola vengono rimossi gli spazi bianchi non voluti. Alla fine, per maggiore chiarezza, il contenuto di frase viene assegnato alla variabile "nome", anche se cio' non sarebbe necessario.

Vediamo ora altri esempi usando i modelli:

```
/* esempio 4 */
say "Inserisci nome ed eta', ad esempio:
Gianluca Girelli,50"
parse pull name "," age
say " Dici di essere" name"," age "anni di
eta'."
```

Come si vede lanciando il programma, quando ARexx e' in attesa di input nella console viene visualizzata una semplice riga vuota.
Spesso e' pero' piu' "user-friendly" visualizzare un "prompt" di richiesta (ad esempio ">") per indicare all'utente che deve digitare qualcosa. Cio' puo' essere fatto con l'istruzione OPTIONS PROMPT come segue:

```
/* esempio 5 */
options prompt ">"
say "Inserisci nome ed eta', ad esempio:
```

```
Gianluca Girelli,50"
parse pull name "," age
say " Dici di essere" name"," age "anni di
eta'."
```

Un semplice parser per giocare

A questo punto, a meno che non siate dei cultori dei linguaggi di programmazione fini a se stessi, starete cominciando ad annoiarvi e, poiche' il nostro motto continua ad essere "Remember when computing was fun?", perche' non mettere in pratica un po' di queste nozioni divertendoci? Personalmente sono sempre stato un grande appassionato di avventure testuali, sin da quando giocavo a "The Count" con il VIC20 di un amico. Come ricorderete, il bello di quei giochi era che raccontavano di mondi che prendevano forma direttamente nella nostra testa e il loro limite era solo la nostra immaginazione. "The Count" raccontava dell'esplorazione di un castello fantasma in cerca, appunto, del "Conte" (Dracula, ovviamente) e la "navigazione" all'interno del gioco veniva effettuata immettendo semplici frasi strutturate come nel primo esempio di questo articolo. Poi venne "Zork" della defunta Infocom e tutto il resto passo' in secondo piano [vedi foto], visto che gli analizzatori sintattici di Infocom riuscivano a fare il parsing di ben due righe di testo....

Senza volersi pero' montare la testa anche noi, grazie ad ARexx ed alle sue potenti istruzioni di gestione delle stringhe, possiamo scrivere il nostro piccolo parser e porre le basi di quello che alla fine sara' il nostro motore di gioco per avventure testuali!

Supponiamo quindi di aver la necessita' di impartire comandi al nostro avatar testuale e di aver deciso che questi comandi possano essere solo nella forma "verbo" o "verbo+nome". Il nostro parser dovra' dunque rispettare queste regole e si occupera' sostanzialmente di suddividere la stringa nelle sue componenti escludendo automaticamente le stringhe vuote (comando nullo!) e quelle contenenti piu' di due parole in quanto non congruenti con le regole della "grammatica" data.

Esaminiamo quindi il sottoprogramma seguente:

```
/*-------------------------*/
/* Parser sintattico       */
/*-------------------------*/
Parser:
if words(frase)=1 then
  select
     when frase='GUARDA' then do
               verbo='GUARDA'; nome=''
               end
     when frase='LISTA' then do
               verbo='LISTA'; nome=''
               end
     when frase='QUIT' then do
               verbo='QUIT'; nome=''
               end
     when frase='VOC' then do
               verbo='VOC'; nome=''
               end
     otherwise say"non capisco. riprova"
  end
  if words(frase)=2 then do
    verbo=left(frase,index(frase,' ')-1)
    nome=right(frase,length(frase)-
index(frase,' '))
  end
return
```

La subroutine viene indentificata da una "label" iniziale ("Parser:") che funge sia da nome del sottoprogramma, sia da indirizzo logico di riferimento per la chiamata da parte del programma principale e termina con l'istruzione "return".
La logica di questa routine rispecchia quanto detto precedentemente quindi:
- se la stringa contiene almeno una parola ("if words(frase)=1") il programma controlla se essa e' nella lista dei comandi singoli conosciuti. In questo caso il comando viene assegnato alla variabile "verbo", altrimenti viene restituito un messaggio di errore;
- se la stringa e' di due parole, si procede alla sua divisione come gia' illustrato in apertura di articolo.

Dobbiamo sostanzialmente notare due cose: innanzitutto, poiche' in ARexx le variabili sono globali, cioe' accessibili da chiunque in qualsiasi momento, nel caso la stringa sia composta da una sola parola la subroutine si occupa anche di azzerare la variabile "nome" in modo da non innescare comportamenti non previsti dalla nostra logica; inoltre, il parser potrebbe anche invocare direttamente le azioni da eseguire una volta decifrata la stringa, ma abbiamo scelto di mantenere le due cose concettualmente separate.

A questo punto il nostro analizzatore sintattico e' pronto ad essere usato all'interno del codice principale che potrebbe avere l'aspetto che segue e che, seppur non direttamente connesso alla trattazione delle stringhe, e' stato inserito per amore di chiarezza.

```
/* main code */
pull frase
call Parser(frase)
select
      when verbo='VAI' then call Vai(nome,
Pos)
      when verbo='GUARDA' then call Guarda()
      when verbo='PRENDI' then call
Prendi(nome)
      when verbo='ESAMINA' then call
Esamina(nome)
      when verbo='USA' then call Usa(nome)
```

```
        when verbo='LISTA' then call Lista()
        when verbo='VOC' then call Vocabolario()
        otherwise say 'non so che significa ' ||
verbo
end
```

Come si vede dal codice, una volta letta in input la stringa ed averla analizzata grazie al nostro parser, a seconda del valore della variabile "verbo" verra' invocata la subroutine corrispondende passando eventualmente come parametro il valole della variabile "nome".
Da notare che nel caso di "VAI" verra' passata alla routine di navigazione oltre al "dove" (es: VAI NORD) anche la posizione attuale (contenuta nella variabile "Pos") da riaggiornare.

Conclusioni

Il problema della formattazione dell'input/output e' vecchio quanto l'informatica stessa essendo intimamente connaturato con il motivo per cui i computer furono creati, cioe' aiutare a risolvere velocemente ed efficacemente i problemi.
Come si e' visto, con ARexx questo problema virtualmente non esiste dato il numero, la potenza e la flessibilita' delle istruzioni che in questo linguaggio si occupano delle stringhe.
Spero che questo articolo abbia dimostrato quanto lavorare con le stringhe possa essere anche divertente. Con il semplice uso di poche istruzioni abbiamo infatti creato il nostro primo parser, che potrebbe essere affinato e potenziato all'infinito.
Se questa lettura via ha fatto tornare la voglia di riesumare qualche vecchia avventura testuale, i link a fine pagine vi saranno molto utili. Non perdetevi comunque i prossimi numeri di RetroMagazine poiche' la sezione di "Programmazione con ARexx" sta per assumere le sembianze di "Game Coding con ARexx". In fondo in fondo ... do you remember when computing was fun?
Buon divertimento!

BIBLIOGRAFIA

• Per le avventure testuali di Scott Adams visita:
http://www.msadams.com/index.htm

• Per le avventure testuali della Infocom visita:
http://www.infocom-if.org/games/games.html

• Per "ZORK" (MS-DOS) visita il sito della Infocom:
http://www.infocom-if.org/downloads/downloads.html

• Per "ZORK" su Amiga visita:
http://www.lemonamiga.com/

• Per gli emulatori Amiga visita:
http://www.amigaforever.com/

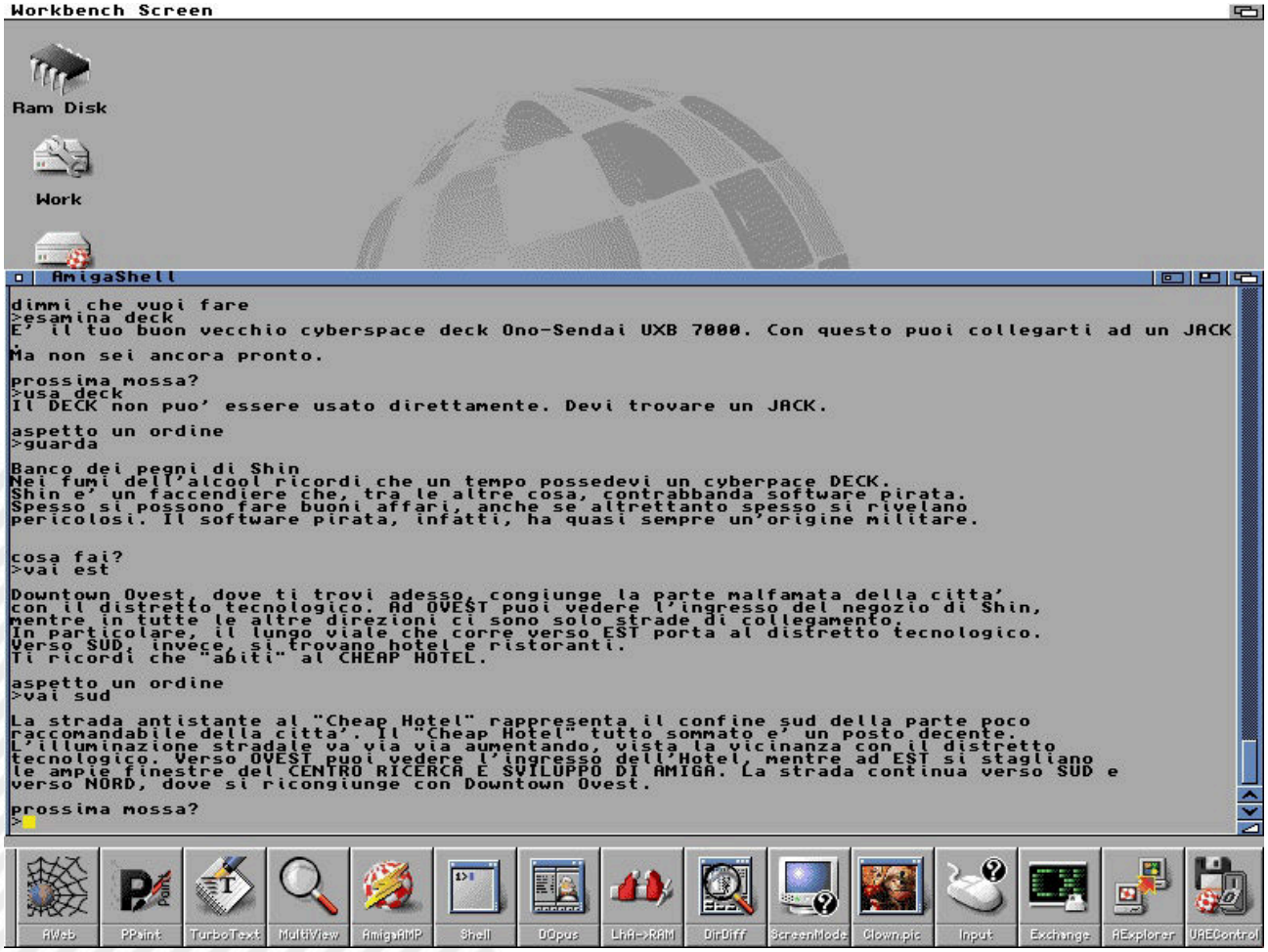


Figura 3: Cyber.rexx su AF12

COMAL: la strana storia di un linguaggio dimenticato - prima parte

di András Vajda

Continua la nostra collaborazione con il gruppo
Retro Programming Italia (RPI) by The Nerds

Sommario

Tra il 1973 e il 1974 il matematico danese Børge R. Christensen e il suo collega informatico Benedict Løfstedt definirono le specifiche della prima versione di COMAL (COMmon Algorithmic Language): un nuovo linguaggio che si impose rapidamente in tutte le scuole della Danimarca e successivamente in gran parte del Nord europeo grazie alle sue peculiari caratteristiche. Semplice come il BASIC ma strutturato come il Pascal, prettamente pensato per la didattica come Logo e Forth, finirà per essere utilizzato anche in campo applicativo per tutti gli anni Ottanta e oltre, grazie ad interpreti e compilatori disponibili per una vasta quantità di home e personal computer.

If you want more effective programmers, you will discover that they should not waste their time debugging: they should not introduce the bugs to start with.
(Edsger W. Dijkstra, 1930-2002)

Introduzione

Ripercorriamo sinteticamente la storia e le caratteristiche di un linguaggio che ha avuto il suo momento di gloria negli anni Ottanta, in ambito didattico ma anche applicativo. COMAL (acronimo di COMmon Algorithmic Language) nasce in Danimarca per una precisa esigenza didattica: l'impiego in corsi di alfabetizzazione informatica agli inizi degli anni Settanta, per i quali il BASIC era considerato inadeguato a causa della sua mancanza di struttura, ma al contempo il Pascal risultava eccessivamente avanzato e complesso. Durante gli anni Ottanta COMAL è stato implementato su due dozzine di home computer: particolare fortuna hanno avuto la versione interpretata 0.14 Public Domain su disco per Commodore 64, seguita dalla versione 2.0 su cartridge (molto più perfezionata, con una notevole quantità di memoria a disposizione dei programmi utente), e il compilatore per Atari con il quale sono stati prodotti anche numerosi applicativi commerciali, sia per usi domestici che burotici in area SOHO, distribuiti in tutta Europa.

Figura 1: Data General NOVA 1200

La genesi di COMAL

In una lunga e semiseria intervista rilasciata a "COMAL Today" e apparsa sul numero 25 (1989), il matematico danese Børge R. Christensen illustra la genesi del linguaggio e la sua a tratti rocambolesca implementazione su un Data General NOVA 1200 dotato unicamente di una unità a nastro carta perforato per la memorizzazione. Vogliamo qui riassumerne almeno i passaggi più salienti, pur rifuggendo dall'idea di proporre una pedissequa traduzione integrale che in quest'ambito divulgativo e colloquiale finirebbe per divenire uno sterile esercizio di acribia filologica, annoiando i lettori. Nel 1972 il college danese di Tønder (vicino al confine con la Germania) presso il quale il professor Christensen insegna matematica decide di dotarsi di uno dei primi calcolatori commerciali per applicazioni di ricerca e per la didattica. I primissimi corsi di informatica non sembrano tuttavia avere un particolare successo, e questo viene attribuito in particolare alla versione notevolmente fallata di Extended BASIC preinstallata sul calcolatore. Christensen lascia in realtà trasparire che anche la sua
limitata esperienza in materia ha il suo peso, rendendogli difficile la lettura dei programmi BASIC : per superare questi dubbi, si reca presso la vicina università di Årus, nota per avere un dipartimento di informatica piuttosto efficiente. Sfruttando la contingenza di dover preparare i problemi d'esame per il suo primo (piuttosto disastroso) corso di informatica, chiede una consulenza a Benedict Løfstedt che lavora appunto in tale università. La risposta di Løfstedt è perfettamente in linea col pensiero informatico dell'epoca: la colpa non è del docente né degli studenti, bensì del linguaggio. Si ricordi che in quell'epoca, tra gli anni Sessanta e Settanta, siamo ancora nel pieno di un dibattito molto acceso nella comunità informatica sulla contrapposizione tra programmazione strutturata e non, tra linguaggi come ALGOL da un lato e BASIC dall'altro, con numerosi punti di contrasto e posizioni fortemente

contrapposte, ad esempio, in merito all'uso della keyword GOTO: dibattito dal quale, oltre a fondamentali articoli come quelli di Edsger W. Dijkstra1 [Dij87, Dij82] e accorati interventi di altri padri nobili dell'informatica come C. A. R. (Tony) Hoare (1934-), Niklaus Wirth (1934-) e Donald E. Knuth (1938-), scaturisce tra l'altro nel 1966 anche un fondamentale teorema di estensione della tesi di Church-Turing sulla computabilità, dovuto agli italiani Corrado Böhm (1923-2017) e Giuseppe Jacopini (1936-2001) [BJ66], il quale sancisce che qualsiasi algoritmo computabile secondo Church-Turing può essere implementato in un linguaggio minimale dotato unicamente di costrutti sequenziali, selezioni e iterazioni (cicli), il che quindi esclude la stretta necessarietà di uno statement come GOTO.

In breve, Christensen viene prontamente indirizzato ad un libro seminale di Niklaus Wirth [Wir73] che è una pietra miliare nella storia dell'informatica applicativa e ne riceve una vera e propria illuminazione. Tuttavia, pur affascinato dal Pascal e dalla programmazione imperativa strutturata, si rende presto conto che la difficoltà di utilizzo per i suoi allievi, nella fascia d'età del college, sarebbe comunque eccessiva. Decide quindi di mantenere un ambiente interattivo e un linguaggio interpretato, ma durante tutto il 1973 in una lunga serie di scambi epistolari col collega Løfstedt costruisce la definizione di un linguaggio sostanzialmente nuovo, ibridando la struttura del Pascal con l'intuitività del BASIC e gettando le basi per quello che diventerà lo standard COMAL 75.

Se la genesi della definizione risulta relativamente semplice, la prima implementazione del linguaggio si rivela ben presto assai più ardua. Dopo avere invano tentato di rivolgersi ad alcune aziende locali, Christensen incarica due dei suoi migliori studenti di aiutarlo a codificare in Assembly sul NOVA 1200 l'intera implementazione dell'ambiente interattivo e dell'interprete, ed è qui che il

Figura 2: Esemplari di nastro carta perforato, larghezza 5 (giallo) e 8 (rosa) fori.

suo racconto assume un tono vagamente epico, sebbene sempre intriso di autoironia. La pressoché totale mancanza di esperienza dell'improvvisato team, nonostante la consulenza intermittente di Løfstedt, e la fissazione su alcune caratteristiche didatticamente utili ma onerose dal punto di vista computazionale, come i nomi di variabile estesi a ben (sic!) 8 caratteri rispetto ai due consentiti dal BASIC, provocano un accumulo di ritardi. Un primo prototipo risulta effettivamente operativo il 5 agosto 1974, ma una versione realmente utilizzabile viene completata solo nel febbraio 1975. Christensen indulge anche nel colorire il racconto sottolineando come uno dei due studenti prescelti, incaricato in particolare del debugging, avesse all'epoca il poco edificante vizio dell'alcool, risultando spesso fuori combattimento per lunghi periodi a causa di sbornie colossali e ritardando così ulteriormente lo sviluppo.

Altro aspetto pionieristico dell'intera vicenda è sicuramente l'imbarazzante limitatezza dei mezzi a disposizione, decisamente primitivi anche per l'epoca: soprattutto il terrificante nastro carta perforato, in mancanza di qualsivoglia mezzo di memorizzazione magnetico nell'installazione Data General in uso, ma anche il costo complessivo dell'intera operazione, che si attesta su un budget di soli 300 dollari. Tuttavia, proprio l'universale diffusione nelle scuole danesi di ogni ordine e grado di macchine NOVA identiche o simili a quella utilizzata per lo sviluppo risulta essere il fattore strategico che decreta una immediata diffusione di COMAL nel mondo della didattica: in meno di un anno la maggioranza degli istituti sul territorio nazionale ha abbandonato il BASIC dell'installazione di default in favore del nuovo linguaggio, anche se in molti casi il caricamento del nastro carta originale con una telescrivente con lettore a 10 caratteri/secondo richiede poco più di un'ora di tempo. Christensen si compiace nell'attribuire tale successo principalmente alla presenza di nomi di variabile di lunghezza relativamente significativa, come risulta dalle interviste agli studenti, e ovviamente alla possibilità di usare costrutti strutturati. Come terza motivazione viene citata la possibilità di attribuire un nome alle subroutines, in realtà implementata fin dai primissimi linguaggi di alto livello (COBOL in primis) ma all'epoca inesistente nei BASIC tradizionali. Tutto ciò sottolinea come COMAL 75 nasce dal traumatico incontro con le limitazioni di un BASIC decisamente primitivo e quindi ne costituisce un tentativo di superamento, anche in termini prestazionali, pur rimanendo nell'ambito di un linguaggio interpretato. Accorgimenti come l'uso estensivo di jump tables dinamiche, puntatori e link diretti alle procedure nascono esplicitamente per superare meccanismi

realmente arretrati e farraginosi come la ricerca sequenziale effettuata dall'interprete nella vera e propria lista linkata semplice costituita dall'insieme delle linee BASIC in caso di GOSUB, che nel decennio successivo saranno progressivamente abbandonati anche nella maggioranza degli home BASIC interpretati.

Si noti che, a questo punto della sua storia, il linguaggio è ancora caratterizzato da un impronunciabile nome danese e probabilmente, sotto tale egida e legato a doppio filo com'era con una serie di macchine non particolarmente diffuse su scala globale e con un mezzo di distribuzione poco pratico come il nastro perforato, non avrebbemai superato i confini del regno di Danimarca. In realtà è lo stesso Christensen, secondo la sua narrazione, a ideare il nome COMmon Algorithmic Language per banale analogia con l'ALGOL (ALGOrithmtic Language), all'epoca uno dei protagonisti di maggior rilievo nel campo della programmazione strutturata: progettato e standardizzato accuratamente, tanto da essere usato (nella versione ALGOL 60 in particolare) come linguaggio ufficiale per l'espressione di algoritmi negli articoli scientifici per oltre tre decenni. Anche la metasintassi conosciuta come Forma di Backus Naur (BNF), ben nota a qualunque informatico che abbia compiuto un minimo di studi istituzionali, è stata in origine concepita specificamente per descrivere la sintassi dei programmi in ALGOL: al pioniere John Backus (1924-2007) si deve l'ideazione di tale formalismo per ALGOL 58, perfezionato poi da Peter Naur (1928-2016) con ALGOL 60, come magistralmente illustrato dall'ineffabile Donald Knuth [Knu64]. Nella pratica l'unica aggiunta alla BNF (standardizzata dalla ISO/IEC 14977:1996, rev. 2018) che non sia riconducibile direttamente ad ALGOL è semplicemente l'uso delle parentesti quadre [ ] per delimitare i parametri opzionali: introdotto in realtà pochi anni dopo ALGOL 60 con la definizione del linguaggio PL/I di IBM e poi diffusosi universalmente anche nelle sinossi informali di comandi e funzioni d'ogni genere, dai comuni linguaggi di programmazione ai batch, alle shell Unix, fino all'help online di singoli programmi applicativi, eccetera.

Piccolo inciso: un aspetto decisamente saliente della BNF e delle sue estensioni (come EBNF o le specializzazioni come TBNF, ABNF o RBNF) è l'espressività. Essa risulta suffciente a descrivere anche la meta-metasintassi stessa di BNF in modo molto conciso, autoreferenziale e ricorsivo, come riportiamo di seguito da un classico esempio presente nella maggioranza dei testi, abbreviato per mere ragioni tipografiche. Risulta poi intuitivo come ad esempio in EBNF, grazie all'uso delle espressioni regolari, l'autodefinizione risulti ancora più concisa a livello di <letter>, <digit>, <symbol>.

```
<syntax> ::= <rule> | <rule>
<syntax> <rule> ::= <opt-whitespace> "<" <rule-
name> ">"
<opt-whitespace>    "::="    <opt-whi    tespace>
<expression> <line-end>
<opt-whitespace> ::= " " <opt-whi tespace> | ""
<expression> ::= <list> | <list> <opt-whi tespace>
" | " <opt-whitespace> <expression>
<lline-end> ::= <opt-whi tespace> <EOL> | <lline-
end> <lline-end>
<list> ::= <term> | <term> <opt-whitespace> <list>
<term> ::= <literal> | "<" <rule -name> ">"
<literal> ::= '" ' <text1> '" ' | " '" <text2> " '"
<text1> ::= "" | <charac t e r1> <text1>
<text2> ::= ' ' | <charac t e r2> <text2>
<character> ::= <letter> | <digit> | <symbol>
<letter> ::= "A" | "B" | "C" | "D" | "E" | "F" |
"G" | "H" | " I " | "J" | . . .
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" |
"6" | "7" | "8" | "9"
<symbol> ::= " | " | " " | " ! " | "#" | "$" | "%"
| "&" | "(" | ")" | . . .
<character1> ::= <character> | " '"
<character2> ::= <character> | '" '
<rule-name> ::= <letter> | <rule-name> <rule-char>
<rule-char> ::= <letter> | <digit> | "-"
```

Una volta ideato un nome convincente per il nuovo linguaggio, dopo la presentazione e la rapida diffusione di COMAL 75 negli istituti scolastici danesi, il professor Christensen si dedica alla stesura della documentazione del progetto, creando così le basi per quello che sarà il primo libro di testo del linguaggio [Chr82], originariamente pubblicato in danese qualche anno dopo la prima versione del linguaggio, nel 1978. Nel frattempo però i minicomputer NOVA si avviano rapidamente verso l'obsolescenza, a causa dell'invasione del mercato da parte dei primi PET: il mero timore di poter essere dantescamente rispedito ad insegnare qualche BASIC, così ci racconta coloritamente il simpatico matematico danese, è suffciente a spingere il suo gruppo di lavoro a perfezionare una ulteriore versione del linguaggio, più completa ed efciente, destinata alle piattaforme emergenti. Così il gruppo di lavoro si amplia, unendo i suoi sforzi ad un team in un altro college e nel giro di un paio d'anni si arriva alla versione COMAL 80, così denominata sia in base al fattore cronologico (1980 è l'anno di presentazione), sia perché sviluppata prevalentemente su una macchina basata sullo Zilog Z80. Nel maggio 1982 viene ratificato lo standard internazionale denominato

COMAL KERNAL, poi sottoposto a riapprovazione 1983. In quegli stessi anni, a maggior gloria del nazionalismo danese, una società locale ha iniziato a progettare e realizzare computer specificamente dedicati alla didattica, grazie soprattutto all'uso di un moderno bus che rende facile l'espansione della macchina e anche la sperimentazione elettronica negli istituti tecnici: ovviamente tale macchina è dotata di COMAL. Al linguaggio però in questo momento mancano ancora molte delle caratteristiche che lo hanno reso famoso nei due decenni successivi: grafica e sprites, suono, un editor avanzato, un ambiente completo che consenta di gestire direttamente nastri, dischi e altre memorie di massa. Tuttavia, si è già ricavato una solida posizione di predominio nell'ambito didattico, tanto che a partire dai primi anni Ottanta nessun istituto danese accetta forniture di calcolatori che non abbiano COMAL a bordo.
Nel 1982 il danese Mogens Kjær, dietro consiglio di Christensen e sulla base dello standard COMAL KERNAL, inizia a sviluppare quella che sarà la famosa e diffusa serie di versioni 0.1x Public Domain per Commodore PET e successivamente per C64, avviando assieme ad un gruppo di pionieri ed entusiasti (Jens Erik Jensen, Helge Lassen e Lars Laursen) una società denominata UniComal ApS, che di fatto succede al team originale nello sviluppo del linguaggio e nel porting verso altre piattaforme. L'ambiente di lavoro viene espanso e perfezionato per sostituire completamente quello del BASIC V2 e il relativo DOS Wedge, dando la possibilità di gestire con semplici e intuitivi comandi i file su nastro e disco, le stampanti e altre periferiche. Vengono aggiunti i moduli binari, denominati "packages", che sono librerie di routine in codice macchina richiamabili direttamente da COMAL. Pochi anni dopo nasce anche una versione sviluppata appositamente per gli Amstrad, successivamente anche un compilatore per Atari e una lunga serie di porting per altri home computer, inclusa una versione avanzata di COMAL 80 su cartridge (la 2.0) per C64 commercializzata nel 1985 che resterà in auge a lungo, praticamente per l'intera storia di tale home computer. Viene presto sviluppata anche una versione con compilatore per i potentissimi VAX-11, evento più unico che raro nella storia dei linguaggi didattici nati su home computer. Negli anni Ottanta e Novanta il linguaggio, nelle sue varie versioni, viene utilizzato per la didattica in numerosi paesi anglofoni del Nord Europa, ma anche per svariate applicazioni commerciali, in particolar modo su C64 con apposita cartridge e soprattutto su Atari e VAX nella versione compilata. Chiudiamo questa succinta cronistoria con una doverosa citazione di Børge R. Christensen: "OMAL is first of all for people who are not professional... to make it possible for people to program computers, even if they were not programming people.". Come talora avviene, si tratta però di un linguaggio talmente ben concepito da attrarre immediatamente e per lungo tempo a venire l'attenzione di molti professionisti, che data l'epoca pionieristica della sua diffusione non ha rischiato di diventare un Santo Graal di pasticcioni e improvvisatori, come è invece è tristemente accaduto alla svolta del millennio con linguaggi concepiti nominalmente con le medesime intenzioni. Inoltre è indiscutibile merito di questo linguaggio l'aver abbreviato il percorso verso il conseguimento di solide competenze di problem solving e logica della programmazione per intere generazioni di studenti in gran parte del sistema scolastico europeo anglofono.

Un primo sguardo a COMAL 80 per C64

La fortunata serie di release per Commodore avviata dal COMAL 0.11 rilasciato da UniComal ApS nel 1982 è culminata con una versione di larghissima diffusione, la 0.14, alla quale si riferiscono anche numerosi manuali. Tale versione Public Domain, caricata da floppy o da nastro, non lasciava tuttavia molto spazio disponibile per i programmi utente (circa 9:900 bytes sul C64): in ultima analisi, ciò non costituiva in realtà un problema per il principale utilizzo del linguaggio, quello didattico e formativo. Dopo una versione intermedia 1.2 destinata principalmente al Commodore PET 8096, compare la versione 2.0, meglio nota come COMAL 80 per C64 prodotta nel 1984 e distribuita su cartuccia CBM nei primi mesi del 1985. Tale versione, disponibile anche su schede con ROM associate ad espansioni di memoria per i PET, garantiva sul C64 30:714 bytes di spazio per i programmi utente, più del triplo delle versioni precedenti. Essa risultava notevolmente più potente e completa delle prime release, ed è stata utilizzata con successo nell'arco di due decenni non solo per la didattica, ma anche per numerosi programmi commerciali.
L'ambiente interattivo sostituiva completamente l'originale del C64, rendendo disponibili potenti comandi per la gestione di dischi e nastri, la stampa, le periferiche su porta utente e seriali, librerie grafiche, perfino un sottosistema LOGO Turtle, editor di programma fullscreen e molto altro sotto forma di moduli binari abilitabili a piacimento.
Risulta inoltre molto semplice espandere il linguaggio con nuove keyword create dall'utente e con moduli binari scritti in Assembly. In questa sede preferiamo però evitare di iniziare con una sterile, pedissequa elencazione delle caratteristiche del linguaggio e delle sue peculiarità sintattiche (tutti aspetti peraltro ampiamente sviscerati nella manualistica indicata in bibliografia), lasciando invece "parlare" degli esempi concreti di codifica come primo primo assaggio della potenza e della reale modernità di COMAL 80. Senza voler appesantire la trattazione con puntuali citazioni dai testi di riferimento fondamentali in

materia come [Seb15, FW08], si preferisce che il lettore valuti intuitivamente, hands-on, la leggibilità e la manutenibilità del linguaggio come caratteristiche oggettive di software engineering e language design. Per cominciare proponiamo quindi un brevissimo programma che in sole 23 LOC genera esaustivamente tutte le permutazioni dei simboli PETSCII immessi in input in una stringa arbitraria (lunghezza massima 20 caratteri, comunque sconsigliata anche ai più temerari, sia pure su emulatori con warp mode). Il programma è tratto direttamente dal disco di esempi che accompagnava la cartridge COMAL 80.

```
0010 // SAVE "permute"
0020 //
0030 // (c) 1984 by UniComal ApS .
0040 //
0050 PAGE
0060 DIM a$ OF 20
0070
0080 PROC permute (a$,k)
0090 IF k<=1 THEN
0100 PRINT a$
0110 ELSE
0120 permute (a$,k-1)
0130 FOR i :=1 TO k-1 DO
0140 permute (a$ (:i-1)+a$ (k)+a$ (i+1:k-
1)+a$ (i)+a$(k+1:),k-1)
0150 ENDFOR i
0160 ENDIF
0170 ENDPROC permute
0180
0190 INPUT "Permute: ": a$
0200 PRINT
0210 permute (a$,LEN(a$))
0220
0230 END "End"
```

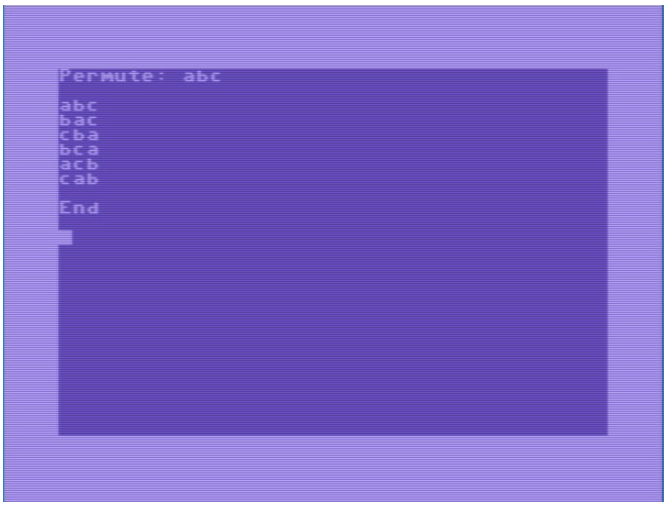


Figura 3: Risultato del programma "permute"

Balzano immediatamente agli occhi la sintesi, l'intuitività e l'assoluta chiarezza della struttura logica del programma, tipiche della programmazione strutturata quanto antitetiche ai primitivi BASIC dei PET. I numeri di linea vengono generati automaticamente dall'editor e sono allineati per default a 4 cifre, aumentando notevolmente la leggibilità. Il programma non fa uso di nomi lunghi di variabile, in quest'occasione. Si nota immediatamente l'uso di un classico algoritmo ricorsivo, sul quale non ci soffermeremo se non per sottolineare come il parametro stringa passato di volta in volta alla funzione sia in realtà l'unione di sottostringhe ricavate dinamicamente con "modernissime" espressioni di slicing, che molti studenti e practitioners credono nate con Python negli anni Novanta o con linguaggi anche più recenti, ma in realtà supportate da oltre mezzo secolo dai benemeriti FORTRAN, APL, ALGOL, Ada e perfino da qualche home BASIC dei primi anni Ottanta, come ad esempio il Sinclair BASIC. Si notino inoltre i commenti che qualcuno potrebbe essere tentato di definire "in stile C++", in realtà altra "modernità" degli anni Settanta: alla quale ovviamente Bjarne Stroustrup nei primi anni Ottanta può essersi liberamente ispirato, a margine del notevole sforzo di concepire il suo C with classes come evoluzione e superamento delle possibilità di SIMULA.

Fine prima parte, continua e finisce nel prossimo numero.

Edsger W. Dijkstra (1930-2002)

Fisico teorico per formazione, è stato uno dei più influenti informatici teorici dell'ultimo mezzo secolo. Pioniere e anticipatore dell'uso dei metodi formali di specifica e verifica, è ben noto in ambito informatico generalista anche come autore di alcuni dei più taglienti aforismi in merito a taluni linguaggi di programmazione in voga all'epoca del dibattito di cui discutiamo, ancor oggi citati sovente, sebbene quasi sempre a sproposito, fuori dal loro precipuo contesto storico.

NVRCA - Non Visual Retro Computer Access

di Alessandro Albano in collaborazione con David La Monaca (Cercamon)

Premessa

NVDA su Windows, Voiceover su iOS e OSX, Talkback su Android e Orca su Linux, sono tutti software per le tecnologie assistive vocali, rivolte a persone con disabilità visiva. Tecnicamente si chiamano screen reader.
Tramite questi strumenti, è possibile utilizzare a pieno, tutte le funzionalità messe a disposizione dal PC in uso.

Questi software fanno uso di un sintetizzatore vocale per vocalizzare il testo presente sullo schermo. Non vi sono restrizioni di funzionamento nei casi in cui il testo sia contenuto in oggetti messi a disposizione del sistema operativo in uso: finestre, icone, menù, documenti di testo o prompt dei comandi / terminale.

Tali strumenti, non funzionano invece in caso si desideri utilizzare software che fanno uso di grafica tramite le tecniche di rendering 2D e 3D. Cito ad esempio il caso di un videogame. Come appare ovvio, in questo caso, lo screen reader, non è in grado di descrivere le scene o le schermate e non è nemmeno in grado di vocalizzare il solo testo presente a video, nemmeno quello dei menù, in quanto prodotto tramite rendering grafico.

Unica soluzione possibile è che chi sviluppa il software, ad esempio un videogioco, inserisca funzionalità vocali ad-hoc al suo interno. Cosa non impossibile, ma certamente non adatta a tutti i prodotti concepiti con la grafica al centro dello sviluppo software. Questo limite esclude le persone con disabilità da tutta una serie di prodotti software 2D e 3D, dai videogiochi fino ai software per il disegno CAD, solo per citarne alcuni.

Figura 1: Display in braille

L'idea di NVRCA

Negli anni 80/90 ero vedente. Come tanti ho posseduto anche io alcuni home computer e ne ricordo molti modelli a casa degli amici dell'epoca. Sono rimasto innamorato dei miei MSX 1, Commodore 64 e Commodore Amiga 500 Plus. Questa passione mi è servita per avvicinarmi al mondo della programmazione, per farla poi diventare la mia professione a partire dal 2000. Ho ripreso in mano da poco questa mia passione, con la differenza che ora sono non vedente e si presenta il problema dell'impossibilità di utilizzo di queste fantastiche macchine.

Così, mi è frullata una (spero geniale) idea nella mente: "Perché non sviluppare uno screen reader per i retro-computer?". Naturalmente, le limitazioni di cui ho parlato nel precedente paragrafo rimarrebbero anche sui retro-computer. Penso quindi che sarebbe fantastico avere uno screen reader da utilizzare nella console BASIC, all'avvio di macchine come il C64 o l'MSX. In questo modo si avrebbe il controllo della macchina e la si potrebbe programmare. In aggiunta, penso che potrebbero essere sviluppati alcuni giochi con il supporto vocale, ad esempio delle avventure testuali o IF (Interactive Fiction).

Come implementare NVRCA

Dopo alcune ricerche, che possono essere comunque estese e migliorate, ho individuato, come macchina di partenza il Commodore 64. Principalmente per due motivi: Il primo è la disponibilità di un sintetizzatore vocale software, il S.A.M. (Software Automatic Mouth), in lingua inglese e in italiano (comparso in edicola tanti anni fa a cura della redazione della storica rivista Commodore Computer Club e chiamato “La Voce”). Il secondo motivo è la quantità di memoria RAM presente sul C64. Per non parlare poi dell’ampia diffusione nel mondo di questa macchina.

Il S.A.M. si presta anche particolarmente bene perché, una volta caricato in memoria, lascia ampio spazio al BASIC V2 nativo della macchina per poter scrivere programmi e giochi in BASIC o Linguaggio Macchina in grado di usare i comandi aggiuntivi messi a disposizione dal software di sintesi vocale. Il tutto in maniera piuttosto semplice e immediata.

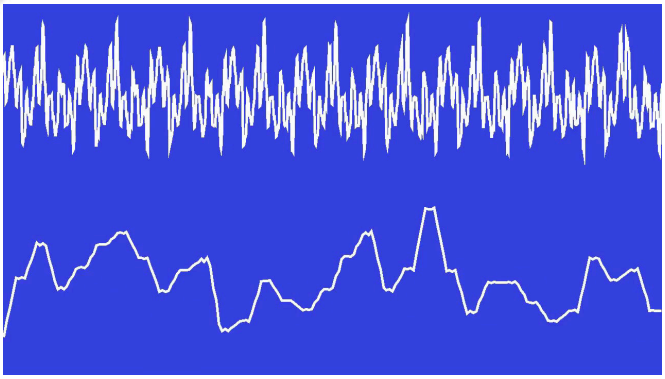


Figura 2: Speech synthesis data

Funzionalità base di NVRCA

Gli screen reader per i PC odierni sopra citati sono provvisti di numerosissime funzionalità. Per NVRCA provo a definirne alcune di base:

- lettura dei caratteri digitati sulla tastiera. Ad esempio mentre si digita nella console o si inserisce un comando in una avventura testuale, il sintetizzatore pronuncia il carattere premuto;
- pronuncia automatica del testo stampato a video dalla console o dall'avventura;
- (opzionale) in digitazione, ammesso che venga pronunciata ogni lettera premuta, quando si sta scrivendo una parola seguita dalla barra spaziatrice, il sintetizzatore deve leggere la parola completa appena digitata a conferma della bontà della digitazione;
- con uno shortcut da tastiera impartire il comando di leggere la riga corrente;
- con uno shortcut da tastiera impartire il comando di lettura dell'intero schermo di testo corrente;
- ove possibile, inserire dei piccoli feedback uditivi, minimali, quando non serve un feedback della sintesi vocale, ma magari è presente un'operazione in corso, per segnalare all'utente che sta accadendo qualcosa sullo schermo (caricamento e salvataggio dati, elaborazioni in corso, ecc.).

Alcune note tecniche

Naturalmente, all'avvio del retro-computer, il sintetizzatore vocale non sarà presente in memoria. A meno che non si utilizzi una cartuccia o qualcosa del genere. Per questa ragione ci vorrebbe una procedura automatica di avvio del software o l'utente dovrà digitare "alla cieca" almeno il primo comando di LOAD del software NVRCA. È necessario, quindi, che il programma NVRCA integri il sintetizzatore vocale o che questo venga modificato ed esteso con le funzionalità che ho elencato sopra per diventare un vero sistema di accessibilità. In alternativa, bisognerebbe caricare prima il sintetizzatore e successivamente caricare NVRCA. Insomma, l'automazione completa del primo caricamento necessario sarebbe davvero il massimo.
Superato questo primo ostacolo, bisogna capire come

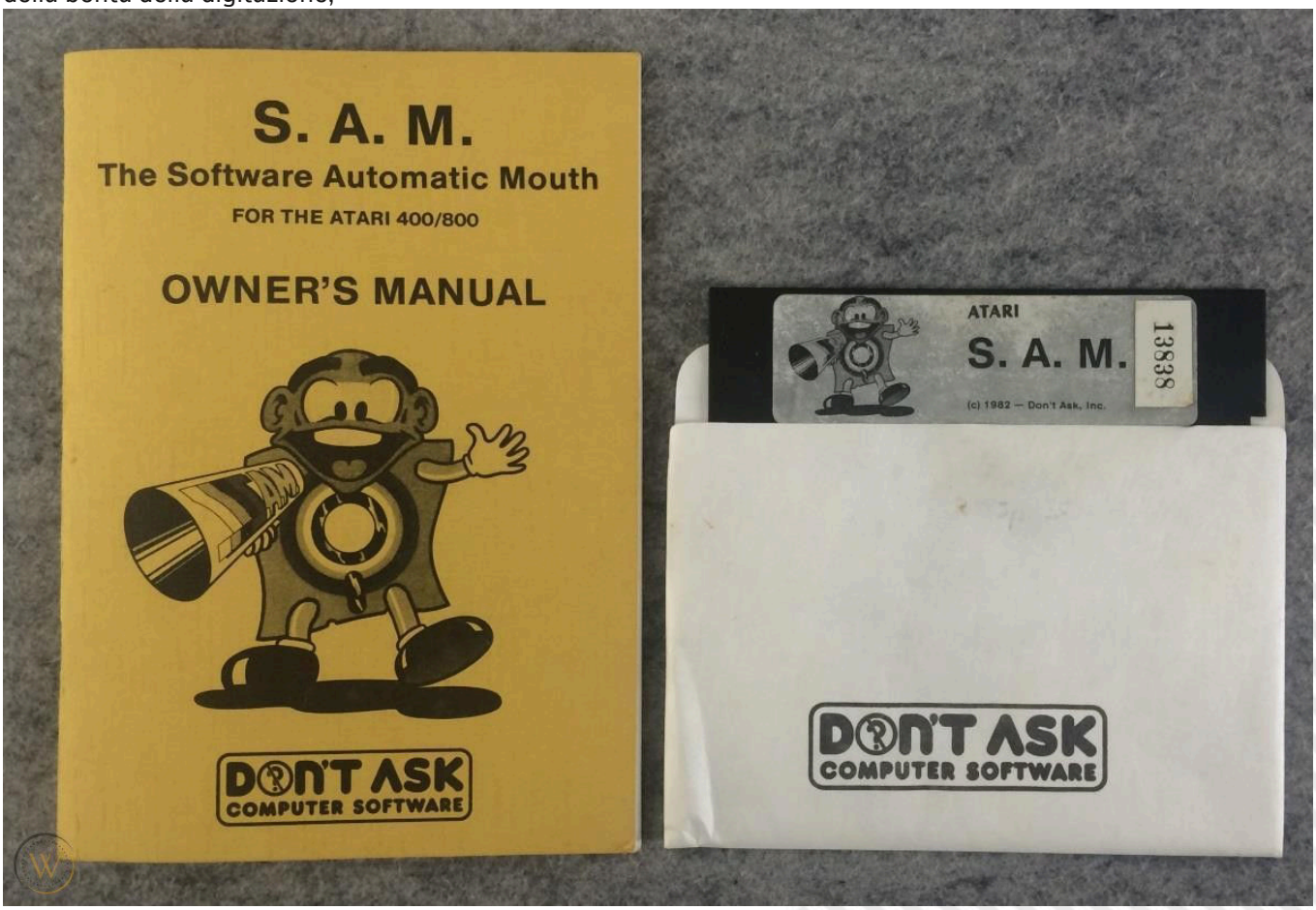


Figura 3: S.A.M. The Software Automatic Mouth per Atari 400/800, floppy e manuale utente

caricare in memoria NVRCA in modo che rimanga in ascolto dei comandi dell'utente e della digitazione dei caratteri/comandi. Naturalmente, per quanto riguarda la gestione dello schermo, il retro-computer deve continuare a funzionare nel medesimo modo di prima. Se si digita un tasto esso deve essere stampato a video oltre che vocalizzato e così via. Deve essere possibile programmare in BASIC o eseguire comandi macchina e udire vocalmente gli output a video del sistema.

Piano di lavoro

Sono un programmatore Java con 20 anni di esperienza, ma non ne ho con la programmazione in BASIC o Assembly per queste retro-macchine che implica la gestione della memoria, dei chip audio e video, dell'input da tastiera, ecc. Ho letto diversi numeri della rivista che ospita questo mio contributo, ho consultato il manuale sul BASIC del C64 e installato diversi software per Windows per programmare i retro-computer. Mi piacerebbe avere un confronto sulla mia idea, sulla sua fattibilità e sulla sua complessità. Se realizzabile, mi piacerebbe svilupparlo a più mani, sfruttando così questa idea per entrare nel vivo della programmazione sui retro-computer. Potremmo così allargare gli orizzonti di questo fantastico e affascinante mondo anche ai non vedenti. Sostengo poi da tempo che i retro-computer siano adatti alla formazione e all'apprendimento dell'informatica e dell'elettronica per i giovani. Chiunque sia interessato o voglia contribuire con un'idea o con codice, può contattarmi ad uno dei miei recapiti.

Riferimenti e link

• Linkedin:
https://it.linkedin.com/in/alessandro-albano-57b015122

• Blog:
www.alessandroalbano.it

• Pagina Facebook:
https://www.facebook.com/alessandroalbanoit/

• Twitter:
https://twitter.com/sharkboyto

• S.A.M. Sintetizzatore vocale per C64:
https://csdb.dk/release/?id=42843

• La Voce (versione con fonemi in italiano):
http://ready64.org/download/scheda_download.php?id_download=69 (La Voce n. 1)

http://ready64.org/download/scheda_download.php?id_download=34 (La Voce n. 2)

http://ready64.org/download/scheda_download.php?id_download=4 (La Voce n. 3)

Figura 4: Tastiera Braille bluetooth

Giappone 10^puntata:
Mrs Game & Watch

di Michele Ugolini

Cari lettori, stiamo tutti vivendo un difficile periodo per colpa del Covid19, spero di sollevarvi un pò il morale con questo articolo pieno di simpatici aneddoti.
Mentre il tempo si sta dilatando, lo spazio dentro casa sembra restringersi: le leggi della Relatività si stanno prendendo gioco di noi, in maniera particolarmente amara. Dobbiamo resistere, solo così potremo uscire vittoriosi da questo tremendo lockdown: collaboriamo, tutti uniti ed altrettanto distanti.
Con l'abbondante tempo libero mi sono cimentato in due attività: ho tradotto svariati testi giapponesi ed ho ricostruito quasi tutte le rocambolesche avventure, curiosità, aneddoti ed interviste riguardo il concepimento del Game&Watch.

Il padre di questo gioco lo abbiamo conosciuto nello scorso capitolo.
Mi raccomando, prima di continuare, se non lo avete già fatto, leggete l'articolo dello scorso numero di RM: "Mr Game&Watch". (cfr. figura 1)
Ora dobbiamo svelare i dati riguardo alla madre e ricostruire il percorso del bizzarro concepimento.
Si parlerà addirittura di proiettili. Iniziamo!

Come è nata l'idea dei Game&Watch? Una storia assolutamente assurda, però altrettanto reale. Prendiamo una serata, particolarmente noiosa, rientrando a casa dal lavoro.
Iniziamo a giocherellare con la calcolatrice, annoiati, poi improvvisamente il nostro corpo viene lanciato alla velocità di 300Km/h.
Nel frattempo lo spettatore di questa scena inverosimile sente scorrere un brivido lungo la schiena, immediatamente gli si accenderà la famosa lampadina sopra la testa, l'icona tipica degli inventori!
Vi sembrerà incredibile, ma ho realmente elencato in due sole frasi: modalità e luogo del concepimento, nonché madre e padre del nostro amato Game&Watch.

Mrs Game&Watch, quindi la noia è realmente la madre di questi gioielli?
Non è troppo azzardato supporlo.
Luogo del concepimento? Considerando i 300Km/h, stavo parlando di un treno Shinkansen.
A tal proposito, mi raccomando cari lettori, leggete il numero di RM relativo al mio articolo sui treni: “Il salvadanaio di Targa”. Così potrete comprendere appieno quanto l'elemento "treno giapponese" sia visceralmente interconnesso alla loro vita quotidiana, in così numerosi aspetti psicosociali che diviene ben presto un cardine insostituibile del loro vissuto. Questo discorso è particolarmente valido per l'amato Game&Watch.
Orario del concepimento? Tarda serata. Il tutto è avvenuto nell'orario di uscita dal lavoro, quando gli stressatissimi business man giapponesi rientrano a casa. Gente psicologicamente devastata: il lavoro in Giappone è una questione di massima serietà, al pari della vita stessa.
Dettagli scabrosi riguardo il concepimento? Beh, il

Figura 1: Game & Watch

tutto è avvenuto picchiettando una calcolatrice ad LCD.
Nome del padre, o meglio, del creativo? Ovviamente il nostro amatissimo Gunpei Yokoi che osservava una scena bizzarra.

Bene, cari lettori, ho formulato questo improbabile concepimento, attraverso una chiave di lettura ironica. Eppure il famoso "fondo di verità" non è un mero fondo, ogni dettaglio è reale, ecco a voi la spiegazione, senza mistificazioni.

La buffa spiegazione poc'anzi descritta è realmente avvenuta a bordo di un treno "proiettile": i Bullet train in Giappone si chiamano "Shinkansen".
Mentre il sig. Yokoi, una sera, viaggiava per affari a bordo di un treno proiettile, notò un signore accanto a lui che armeggiava con una calcolatrice LCD. Yokoi osservò, affascinato, mentre l'uomo premeva i pulsanti, con aria annoiata. All'improvviso, Yokoi si chiese se i pendolari stanchi, che cercavano di passare il tempo, potessero essere interessati a un dispositivo di gioco portatile.

Così nacque nella sua testa una sfocata idea del futuro giocattolo. Non l'oggetto in sè, ovviamente l'immagine non poteva essere già plasmata, ancora mancavano troppi ingredienti, però in quel momento era stata concepita l'idea! Sappiamo tutti quanto sia complessa la strada di una fragile intuizione: prima o poi l'idea dovrà scontrarsi con i problemi pratici della realtà.

Siamo a conoscenza che Game&Watch è stato partorito da Nintendo. Quindi, come ha fatto una normalissima persona come Mr "G", Gunpei Yokoi, a varcare le porte produttive della grande "N", cioè Nintendo?
La sua collaborazione con Nintendo è nata nel 1965.
Appena laureato alla Doshisha University, fu rapidamente assunto, proprio nel 1965 da Nintendo.
La ditta all'epoca produceva giochi da tavolo e giocattoli vari.
Il compito iniziale di Yokoi fu la manutenzione delle macchine che producevano le famose carte da gioco giapponesi Hanafuda.
Nel 1966, il grande capo di Nintendo, Hiroshi Yamauchi, visitò la fabbrica in cui lavorava Gunpei e notò un giocattolo che era stato ideato e costruito da lui stesso per intrattenersi durante il suo tempo libero. Yamauchi gli ordinò di sviluppare proprio quel giocattolo e di crearne un prodotto da immettere nel mercato entro Natale. Si trattava di un piccolo braccio estensibile che fu commercializzato con il nome di "Ultra Hand".
Fu un successo enorme. (cfr. figura 2)

Da quel preciso momento storico, numerosi fortunati eventi fecero proseguire Gunpei agilmente, nel famoso percorso ad ostacoli.
In quegli anni, nella grande Nintendo, ciascun membro dello staff preposto alla realizzazione di un progetto, non godeva assolutamente di alcuna mansione specifica: questo aspetto donò grande fortuna a Mr "G" nel suo immediato futuro!
Per esempio, quando un gruppo di ingegneri veniva assegnato alla creazione di un oggetto, all'interno del gruppo stesso non sussisteva una divisione netta tra programmatori, progettisti e sviluppatori di hardware.
Ognuno di loro, da bravo tuttofare, si alternava nella programmazione del software, partecipava anche

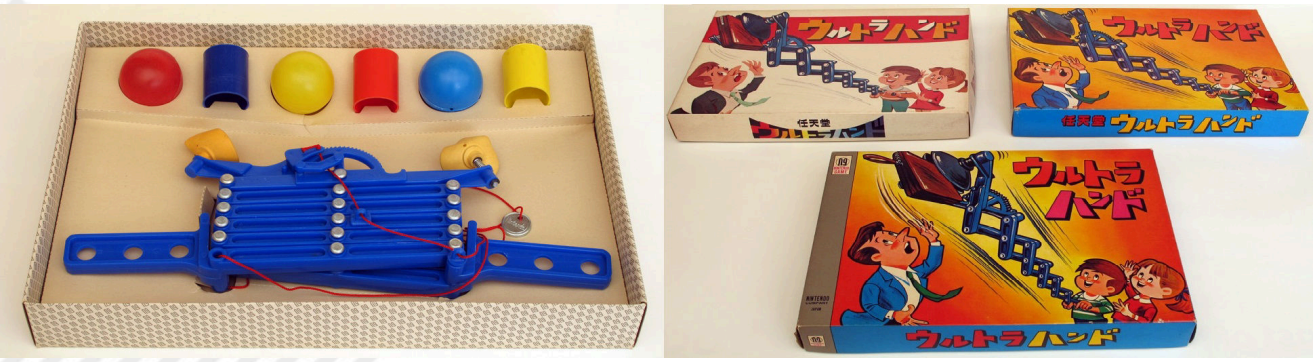


Figura 2: il gioco Ultra Hand

alle riunioni per discutere di nuove idee, proponendo innovazioni e revisioni, occupandosi anche di hardware, svolgendo lavori manuali, addirittura registrando spot pubblicitari.

[...] Bypassando momentaneamente l'arco temporale che sto narrando, vorrei raccontarvi un aneddoto curioso. Al momento di lanciare sul mercato il nostro amato Game&Watch , lo staff aveva creato un geniale spot pubblicitario tramite un siparietto realmente divertente. Si erano nascosti sotto una scatola, collegata ad un cavo, ed interpretavano il gioco. La parte superiore della scatola era illuminata, sopra c'era una console Game & Watch e sembrava che il protagonista della pubblicità ci stesse giocando. [...]

Ritorniamo sui nostri passi, da questo divertente siparietto, al reale ingresso nella grande "N", c'è ancora tanta strada da narrare.
Proprio la strada, infatti, a pari merito del treno, fu di grande aiuto in questo progetto.
Similmente alla situazione osservata da Mr “G” nello Shinkanen, questa volta la strada ha magicamente mescolato tre incredibili ingredienti:
1) Un bel giorno, da bravo tuttofare, il nostro amato Mr "G", ha dovuto temporaneamente sostituire l'autista ufficiale del capo della grande "N".
2) Proprio qualche giorno prima, Mr "G" aveva assistito alla famosa scena dentro lo Shinkansen.
3) Il capo della grande "N", proprio quella mattina, aveva in programma una riunione di lavoro: doveva incontrare un suo collega ed amico, il capo della Sharp, Akira Saeki. Scopo dell'incontro? Doveva aiutarlo a risolvere un problema nel reparto di produzione dei pannelli LCD: data la mole enorme di calcolatrici tascabili vendute, il mercato ormai era saturo. Quindi il capo di Sharp stava cercando nuove idee per aumentare il mercato di questi pannelli a cristalli liquidi.

Il capo della grande “N” non era ancora a conoscenza del problema del suo amico.
Durante il viaggio, però, il timido pilota, Gunpei, gli rivolse la parola. "Direttore, mi perdoni per una cosa di poco conto... scusi il disturbo, ho recentemente assistito in treno ad una situazione che... e allora mi chiedevo se si potesse creare qualcosa per... e avrei pensato che si potrebbe fare questa cosa che...".

Quanto proficuo fu questo "brodo primordiale" di elementi magici, così divinamente predisposti e bendisposti sul tavolo del destino, per far immediatamente allestire la sala parto del futuro Game&Watch.
Il capo della grande "N" raccontò la grande idea del piccolo Mr "G" al capo della grande "S".
Entrambi intuirono immediatamente la genialità di Gunpei.
Quindi Mr "G" fu rapidamente convocato per la pianificazione del progetto.
In figura 3 allego alcuni disegni del progetto di "Game & Watch: Chef". E' il quarto titolo Game & Watch della serie Wide Screen. Un cuoco deve mantenere il cibo costantemente in aria lanciandolo con una padella. Uscito in Giappone nel settembre del 1981. (cfr. figura 3)

Ovviamente anche questi due big dell’industria utilizzarono al meglio le proprie idee e soprattutto l'esperienza pratica.

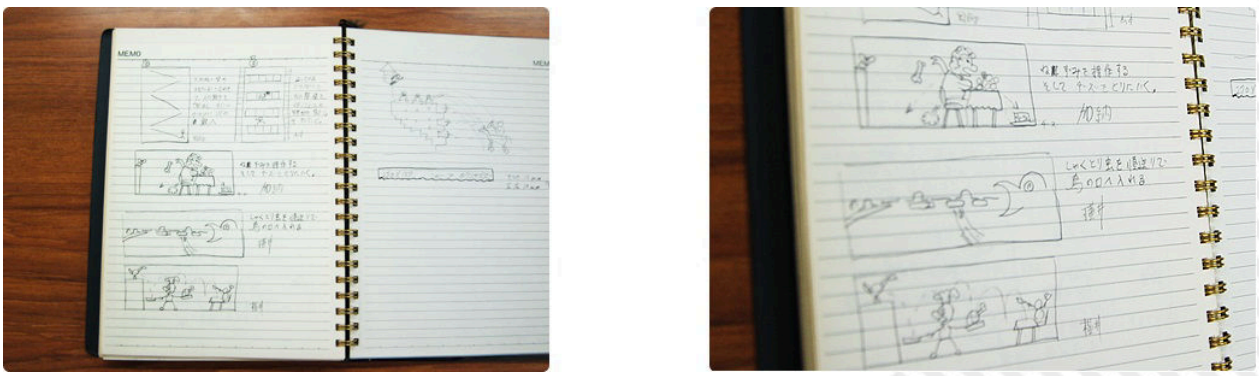


Figura 3: appunti per il gioco Game & Watch: Chef

Stabilirono quindi svariate regole per armonizzare il progetto.
Punto primo: doveva essere tascabile con dimensioni simili alle calcolatrici tascabili di quel tempo.
Punto secondo: doveva essere sia valido come un giocattolo, che utile come un orologio.
Punto terzo: doveva essere di immediata comprensione, utilizzabile senza alcun manuale allegato.
Punto quarto: non doveva generare stress, bensì doveva toglierlo.
Punto quinto: la vendita era destinata unicamente alla categoria dei business man.
Punto sesto: l'utilizzo doveva essere ergonomico, comodo, con tasti facilmente raggiungibili dai pollici.
Punto settimo, sicuramente il più importante: doveva essere facilmente occultabile nelle mani.
Cari lettori. Si, avete letto bene. Occultabile. Game&Watch è nato anche grazie a questa peculiarità sociale, tutta nipponica!

Stiamo parlando di un giocattolo designato per un pubblico adulto: uomini di affari giapponesi che rientravano stanchi dal lavoro, la sera, seduti in treno, in una carrozza ultra affollata, con tante persone geometricamente e strettamente sedute a sinistra, destra, davanti e diritte in piedi sopra di sé.
In Giappone, in quegli anni, l'aspetto ludico per un uomo di affari era una questione pruriginosa, nel senso che, un uomo adulto che giocava, purtroppo, creava abbastanza imbarazzo fra i presenti. L'immagine di affidabilità e serietà per un uomo di affari era di vitale importanza. Sì, certamente anche oggi è estremamente importante. Soprattutto al tempo, tuttavia, un uomo di affari non doveva assolutamente mordicchiare un dolcetto o maneggiare un giocattolo. Non dico che avesse dovuto entrare in casa da "uomo duro" con le scarpe piene di fango, gettando la cacciagione insanguinata sul tatami, però sto anche dicendo che avrebbe destato scandalo entrando in casa con in mano un... giocattolo!

Il Game&Watch, perciò, è stato plasmato anche attraverso questo obbligo sociale.
Perciò il serioso business man, per non scalfire il proprio imperturbabile decoro, poteva estrarre dal taschino il proprio giocattolo solo senza la presenza di vicini curiosi, pronti a sbirciare il peccaminoso giocattolo in mano a quel deprecabile uomo adulto.
Se poi all'improvviso fosse arrivato il controllore, oppure qualcuno seduto nuovamente a fianco, beh, l'uomo di affari lo avrebbe agevolmente silenziato, occultato tra le mani e reinserito nel taschino, esattamente come se si fosse trattato di una normale calcolatrice tascabile.

Inutile dirlo, tanto rapidamente avvenne il parto dopo il concepimento, altrettanto rapidamente si spostò il target di età degli acquirenti.
Gli uomini d'affari vennero ben presto battuti, in termini di vendite, dai più giovani.
Bambini e ragazzi di tutte le età dovevano assolutamente possedere un Game&Watch: fu moda, un boom rapidissimo ed incontrollabile.
Il nome Game&Watch divenne rapidamente meno carico di interesse a livello di "Watch" e più carico di importanza nel "Game". Vennero create tantissime varianti di gameplay. (cfr. figure 4 , 5)
Nello scorso articolo ho elencato alcune tipologie tra le più vendute al mondo.
Avvenne tutto così rapidamente che, sempre in quegli anni, mentre il boom di vendite dilagava, nel medesimo staff di Mr “G” nasceva il problema per Masayuki Uemura nella realizzazione del Game Boy. In realtà, grazie alle incredibili vendite del Game&Watch, il progetto Game Boy andava a gonfie vele, ma il prezzo per la realizzazione era momentaneamente troppo oneroso, perciò venne in aiuto il nostro amato

Figura 4: bozzetti di gioco per Game & Watch

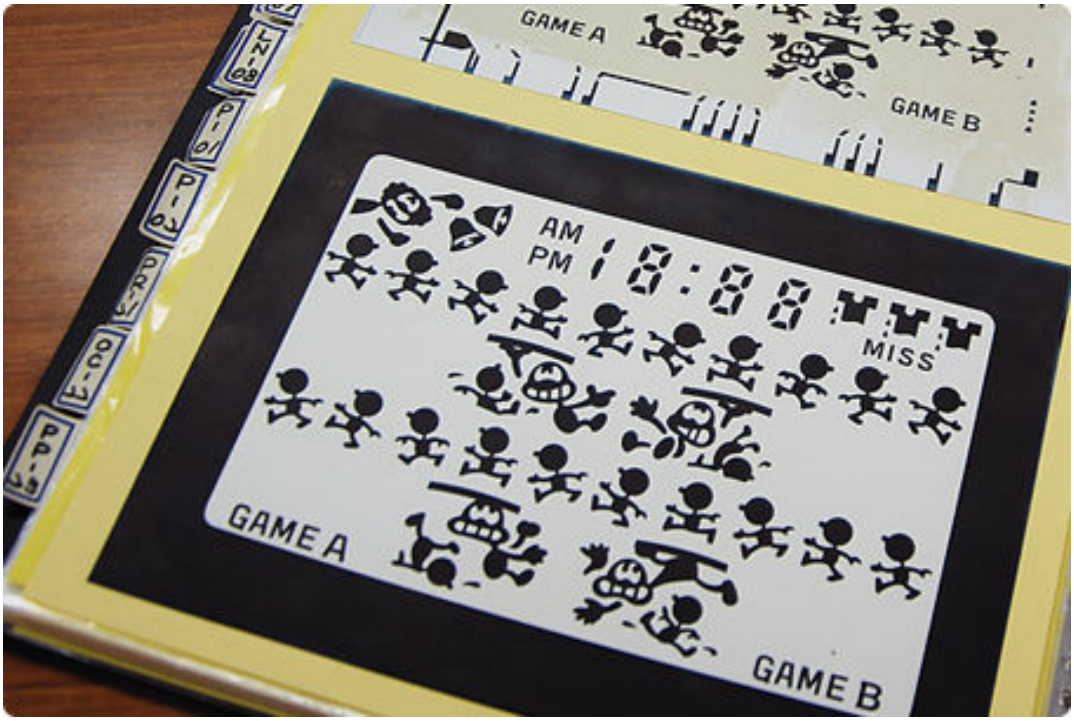


Figura 5: bozzetti di gioco per Game & Watch

Gunpei. Cari lettori, questa è tutta un'altra storia. Ho solo accennato alla nascita del Game Boy giusto per farvi capire che improvvisamente in Giappone il boom dei Game&Watch aveva aperto le infernali porte di un vorticoso ed incontrollabile ambiente elettronico! Ci fu anche un problema di vendite ed esportazione poiché rapidamente l'oggetto divenne mito. Ad oggi suppongo che non sia più un semplice “mito”, penso abbia raggiunto lo status di "oggetto immortale".

La strada fu complice per far nascere Game&Watch e sempre la strada portò via Mr "G": morì in un incidente stradale nel 1997. Non dobbiamo essere tristi, ogni volta che armeggiamo con un Game&Watch, oppure un suo clone, sapremo apprezzare meglio la ricchezza che vive in questo oggetto, così perfetto, minimale, dotato di una stilizzata anima immortale.

Bene cari lettori, nel prossimo numero aggiungerò altri dettagli riguardo i nostri amati Game&Watch ed inizierò a spolverare il discorso accennato nella scorsa puntata: la resurrezione dalle ceneri. (cfr. figura 6)
Per ora è tutto, a presto!

Figura 6: Game & Watch - Ball

VISIBLE SOLAR SYSTEM

Anno: 1983
Sviluppatore: Commodore
Editore: Commodore
Piattaforma: Commodore 64
Genere: Educativo

In breve: un viaggio virtuale nel sistema solare, a bordo di un'astronave.

Sei il comandante di un'astronave in viaggio attraverso il Sistema Solare. La tua nave ha uno spazio di manovra di oltre 1 miliardo di miglia ed è fornita di un sistema computerizzato per aiutarti a portare a casa nuove ed eccitanti scoperte.

Si apre così il manuale di Visible Solar System; un incipit piuttosto ambizioso, come si conviene ad un programma di esplorazione spaziale. Vediamo se le premesse sono state mantenute o meno.

NAVIGAZIONE

Dopo un'introduzione piuttosto scarna (Fig. 1), ci troviamo di fronte al sistema di navigazione dell'astronave (Fig. 2): piuttosto limitato nonostante la premessa iniziale del manuale.
Dal centro dello schermo ad uscire, sono visibili le orbite di Mercurio, Venere, Terra, Marte e Giove. I numeri ad essi associati indicano la loro distanza dal sole. La posizione della nostra astronave è indicata dall'icona rossa. Sulla sinistra dello schermo è indicata l'altitudine della nostra astronave sopra l'orbita dei pianeti (in milioni di miglia). L'astronave è inoltre dotata di una telecamera che vi permetterà di vedere l'orbita ed il movimento dei pianeti in 3D come li vedremmo dall'astronave.

Premete il tasto 'V' per vedere la rappresentazione tridimensionale dei pianeti Mercurio, Venere, Terra, Marte e Giove come se foste seduti sull'astronave (Fig. 3).

Premete il tasto 'O' per tronare di nuovo al centro di controllo dell'astronave (Fig. 2).

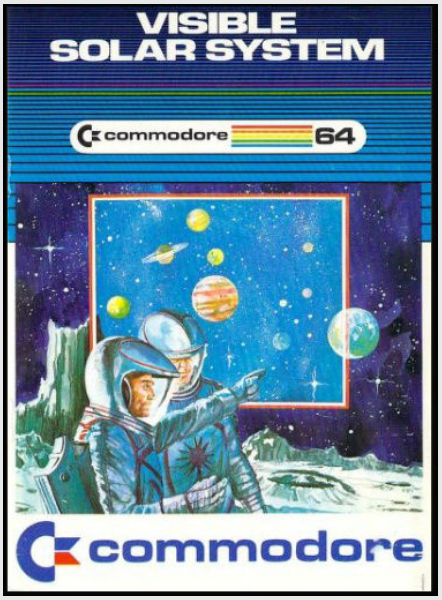
Una volta tornati al centro di controllo, potrete spostare l'astronave e la telecamera utilizzando i tasti:
CRSR → Sposta il target a destra
CRSR ↓ Sposta il target indietro
← Sposta il target a sinistra
↑ Sposta il target avanti
G Sposta l'astronave al target
U Sposta l'astronave in alto
D Sposta l'astronave in basso
, Sposta la telecamera in alto
. Sposta la telecamera in basso

Una volta decisa la posizione dell'astronave, premete il tasto 'V' per vedere la rappresentazione tridimensionale dei pianeti. (NDR - la prima volta che ho visto la rappresentazione in fig. 3 avevo scambiato Giove per la Morte Nera... :-)).

ESPLORAZIONE DEI PIANETI

I pianeti Terra, Marte, Giove e Saturno possono essere esplorati da vicino premendo il tasto 'P'.
Per spostarsi da un pianeta all'altro, basta premere nuovamente il tasto 'P' ed il pianeta successivo verrà visualizzato con la sua scheda personalizzata.

Si vedano le immagini in Fig. 4 per Marte, Fig. 5 per Giove, Fig. 6 per Saturno e Fig. 7 per la Terra. Da notare che nella visualizzazione della Terra è ben visibile anche la Luna. Ognuna di queste schede è corredata da una serie di informazioni tra le quali:
- la distanza dell'orbita del pianeta dal sole, indicata dall'asterisco blu in cima allo schermo
- ORBIT: indica la distanza dell'orbita dal sole (in miglia)
- RADIUS: indica il raggio del pianeta, anche questo in miglia
- YEAR: indica la durata dell'anno in giorni tessrestri



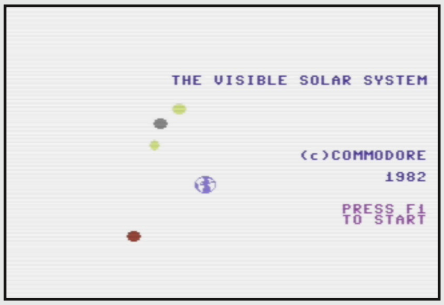


Fig. 1 - Introduzione

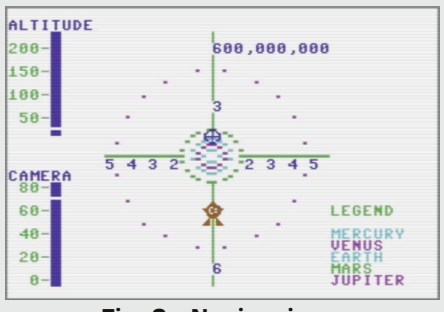


Fig. 2 - Navigazione

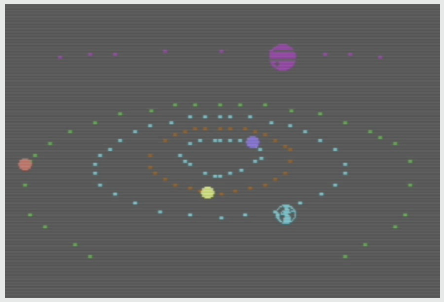


Fig. 3 - Vista dall'astronave

- DAY: indica la durata del giorno in ore terrestri
- MOONS: indica il numero di satelliti del pianeta

Ovviamente la forza di gravità è differente su ognuno di questi pianeti, quindi WEIGHT indica quale sarebbe il tuo peso sul pianeta assumendo che il tuo peso sulla Terra sia 160 pounds.
AGE invece indica la tua età sul pianeta assumendo che la tua età sia 24 anni terrestri.

COMPUTER PLANETARY
Se vuoi vedere più informazioni riguardo i sei pianeti più prossimi al sole, Mercurio, Venere, Terra, Marte, Giove o Saturno, puoi utilizzare il Computer Planetary. Premendo i numeri dall'1 al 6 vedremo le schede relative rispettivamente a questi 6 pianeti (Fig. 8 e 9); le schede contengono queste informazioni:
- in cima allo schermo è visibile il nome del pianeta e la sua distanza dal sole in miglia
- RADIUS in miles: è la distanza dalla superficie del pianeta al suo centro. Il raggio di Giove e Saturno è misurato dalla punta estrema del loro denso strato di nubi.
- YEAR in Earth Days: è il tempo che impiega il pianeta a percorrere un'orbita completa attorno al sole. In giorni terrestri.
- DAY in Earth Hour: è il tempo di rotazione del pianeta sul suo asse. In ore terrestri.
- MOONS: Il numero di satelliti che orbitano attorno al pianeta.
- ORBIT in Miles perSecond: è la velocità del pianeta nella sua orbita. In miglia al secondo.
- ESCAPE velocity in Miles per Second: è la misura della velocità necessaria ad un missile per uscire dall'orbita del pianeta. In miglia al secondo.
- TO ATM. in - degrees Farenheit: è la temperatura del pianeta alla cima delle sue nubi. In gradi Farenheit.
- WARM ATM. in + degrees Farenheit: è la massima temperatura raggiungibile nell'atmosfera del pianeta. In gradi Farenheit.

ASTROCALC
Il software è inoltre corredato di un calcolatore astronomico in grado di comparare i pianeti. Premendo il tasto 'A', è possibile visualizzare nella colonna di destra delle schede del Computer Planetary, una serie di informazioni riguardanti una comparazione tra il pianeta nella scheda ed uno di riferimento.

Assumendo di essere per esempio sulla scheda del pianeta Terra e volerla comparare con Marte, dovremo premere il tasto A fin quando ASTROCALC MARS non apparirà nella colonna di destra.
A questo punto scopriremo che il raggio di Marte è la metà di quello della Terra, mentre un anno marziano è invece 2 volte quello terrestre.

Il manuale si completa inoltre con un tour virtuale dei pianeti, fornendo alcune informazioni addizionali non contenute nel software.

GIUDIZIO
Ovviamente questo tipo di software risente pesantemente del passare degli anni e ne esce sicuramente sconfitto. Un software come questo non può che far sorridere ai giorni nostri, abituati ad ottenere foto ad alta definizione, animazioni e giga e giga di informazioni riguardanti ogni più remoto oggetto astrale con una banale ricerca su Internet. Probabilmente nel 1983 era un oggetto di sicuro valore didattico, ma al giorno d'oggi invece ha valore soltanto come preservazione del software. Vale comunque la pena provarlo.

REPERIBILITA'
Il software può essere scaricato in formato cartridge qui:
http://www.elysium.filety.pl/tools/cartridges/pack.2/Visiblss.zip

Il manuale in inglese, invece si può reperire su Archive.org:
https://archive.org/details/Visible_Solar_System_1982_Commodore_a/mode/2up

A questo punto non mi resta che augurarvi buona retro-navigazione nel sistema solare.

di **Francesco Fiorentini**

Fig. 4 - Marte

Fig. 5 - Giove

Fig. 6 - Saturno

Fig. 7 - Terra, con la Luna a destra

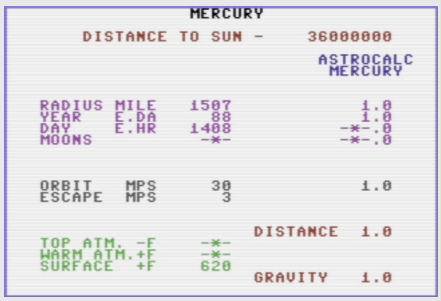


Fig. 8 - Informazioni su Mercurio

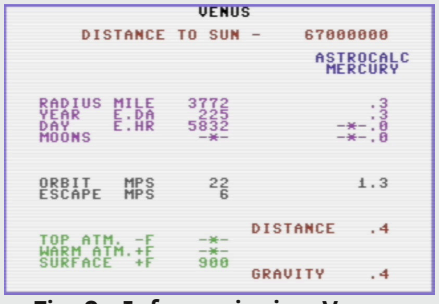


Fig. 9 - Informazioni su Venere

ATARI 80 CLASSIC GAMES IN ONE!

Un diamante da collezione direttamente dal passato!

Nel momento in cui sto scrivendo l'articolo, sono passati esattamente 15 giorni da quel 3 marzo in cui in televisione venne annunciato di rimanere in casa per scongiurare il più possibile la diffusione del Covid-19. Milioni di italiani si sono ritrovati a dover trascorrere intere giornate a casa, senza saper come intrattenere il tempo che sembra non passare mai.
Tra le tante attività c'è senza ombra di dubbio quella della pulizia della propria abitazione che, molto spesso, può rivelare sorprese inaspettate. E' esattamente il caso mio! Era una domenica pomeriggio quando decisi di dare una bella spolverata in cantina. Chi l'avrebbe mai detto che questa pulizia sarebbe diventata una vera e propria caccia al tesoro? Io non di certo. Tra i vari scatoloni e la nostalgia che mi assaliva nel ritrovare oggetti appartenenti al mio passato, spiccò una custodia di un videogioco che non ricordavo nemmeno possedere. Si trattava del gioco per pc Atari 80 Classic Games In One.

Anni fa era un appuntamento fisso comprare i cereali al supermercato per potersi accaparrare una copia dei dischetti che si trovavano all'interno. Impossibile dimenticarli! All'epoca probabilmente non era comprensibile l'importanza di possedere quei titoli intramontabili. Oggi invece hanno un valore inestimabile per chi, come me, ama collezionare oggetti appartenenti al proprio passato.
Ricordo ancora il momento in cui vidi questo videogioco all'interno della scatola dei cereali. Avendo solo 6 anni all'epoca, mi trovai letteralmente spiazzato nel trovarmi davanti linee e quadratini colorati a 8 bit che si muovevano ed emettevano suoni strani. Fu questo il motivo per cui quel dischetto fu destinato a rimanere chiuso in un cassetto. Oggi da buon collezionista, ritrovare quella collezione di classici arcade è un evento che non ha parole per essere descritto.

Per chi non lo conoscesse, la collezione Atari 80 Classic Games in One, è appunto una selezione di 80 giochi arcade definiti "Classici", o ancora meglio "Leggende" del mondo videoludico. Contiene titoli come 3D Tic Tac Toe, Pong, Crystal Castles (la mia droga :-D), Centipede, Asteroid, Battlezone, Missile Command, Star Raiders e tanti altri.
La collezione prevede addirittura titoli per l'Atari 2600 che, tramite un'apposita emulazione, permettono all'utente di poterli avviare e giocare su qualsiasi piattaforma Windows.

Nel momento in cui sto scrivendo l'articolo posso dirvi con certezza che questo disco, rilasciato nel lontano 2003 per Windows 98 e XP, funziona perfettamente su Windows 10 (semplicemente fantastico no?)... E perché non eseguire una virtualizzazione di un Windows 98 giusto per completare l'effetto retrò?

di **Marco Fiaschi**

FIX-IT FELIX JR

Editore: SEGA
Sviluppatore: TOBI KOMI
Piattaforma: Sega Genesis/Megadrive
Genere: Puzzle game

Nel 2012 uscì nelle sale il film d'animazione "Ralph Spaccatutto" ambientato in un mondo dove i videogiochi "vivono" oltre il semplice momento ludico. Un buon prodotto di Disney.

Come sicuramente saprete nel gioco era presente FIX-IT FELIX, dove guidavamo Felix aggiustatutto nella riparazione di un palazzo messo a soqquadro dal gigantesco e burbero Ralph.
Le richieste dei fan del film e degli accaniti giocatori fecerò si che Sega producesse una versione fisica del "finto videogioco".
Ed ecco che salta fuori questo FIX-IT FELIX JR, dove non faremo altro che vestire la tuta di Felix e aiutarlo a riparare i danni di Ralph allo stabile.

Il gioco è piuttosto semplice ed intuitivo. E' a quadro singolo, dove muoveremo Felix tra cornicioni e balconi e, premendo il tasto A, ripareremo le finestre danneggiate da Ralph.
Riparate tutte le finestre del quadro si raggiungerà quello successivo, fino a toccare il piano superiore dove butteremo giù il bestione arrabbiato.

Un gioco davvero carino, nato proprio con l'intenzione di essere retro e di mantenere quel fascino retro presente anche nel film.
Semplice e ben realizzato, il gioco risulta ripetitivo andando avanti anche se il livello di difficoltà cambia leggermente le cose.

E' possibile acquistarlo in versione fisica su Amazon, oppure scaricarlo dal web e giocarlo su emulatore o caricandolo su un sd inserirlo in un everdrive.

Carino, giocabile... rètro.

di **Carlo Nithaiah Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 80%

Un prodotto veloce, semplice da apprendere e dotato di una buona difficoltà.
Si vive molto l'atmosfera presente nel film, anche se in realtà del film non è presente nulla tranne i due protagonisti.

» Longevità 60%

Il gioco presenta un buon livello di difficoltà ma a lungo andare diventa ripetitivo.
Peccato.

THE LEGEND OF ZELDA
A LINK TO THE PAST

Anno: 1991
Sviluppatore: NINTENDO
Editore: NINTENDO
Piattaforma: Super Nintendo
Genere: ACTION/GDR
Disponibile anche per Gameboy Advance, servizio online di Switch - romhack localizzata in italiano disponibile sul web.

Opera Magna o Magnum opus è un'espressione latina che significa "grande opera".
Viene utilizzata soprattutto in riferimento alla più celebre, grande e importante opera di uno scrittore, un artista o un compositore ed in questo caso di un videogioco.
Sono tre le grandi opere su Super Nintendo, assieme a Super Mario World e Super Metroid c'è appunto questo The Legend of Zelda: A link to the Past, terzo capitolo della serie. Primo ed unico sulla console a 16 bit. Capolavoro di ben 29 anni fa, fresco come se fosse uscito oggi stesso.
Arrivando dritti al punto, il capitolo di Zelda a 16-Bit è impeccabile in ogni sua singola, minima componente.

Così sorprendente che stupisce ancora oggi per una cura nel level design che va oltre il maniacale, offrendo dungeons così articolati che ancora oggi brillano di luce scintillante.
Stupisce per la vastità del mondo di gioco principale, così vasto e pieno di segreti, che trasmette una sensazione di esplorazione al limite dell'indescrivibile, tanto grande da poter rivaleggiare con i modelli degli attuali gdr open world.

Nulla è lasciato al caso, nulla!
Ne' uno scenario, un enigma, un combattimento. Tutto risulta perfettamente incastrato nel tessuto del gioco concepito da Shigeru Miyamoto, oliato alla perfezione da funzionare dal primo livello all'ultimo scontro.

E colpisce forte l'enorme numero di strumenti da raccogliere e utilizzare, lascia senza parole soprattutto il fatto che ognuno di essi, anche quello apparentemente più insignificante come può essere un retino per catturare farfalle, risulta fondamentale per procedere nell'avventura, pronti ad affrontare ogni ostilità e trappola grazie proprio agli indispensabili vantaggi regalati da ciascun elemento del nostro inventario.

A conferma di quanto già detto... nulla, assolutamente nulla è stato lasciato al caso e che invece il giocatore deve capire come sfruttare appieno tutto ciò che il gioco ci mette a nostra disposizione.
Girare per Hyrule dà soddisfazione,

esplorare il paesaggio, i villaggi, i dungeon e arrivare alle complesse Boss fight è una "vera libidine" come proverete la stessa cosa guardando la stupenda estetica e l'atmosfera. Il contrasto tra Hyrule e il Dark world colpisce profondamente.

Colpisce ancora più forte l'immenso accompagnamento musicale, perfetto in ogni contesto grazie a un'orchestrazione sublime che lascia senza parole. Ciliegina sulla torta, una veste grafica che colpisce forte con il suo stile apparentemente semplice, ma nei fatti realizzato con il cuore. Un piccolo cartone animato, con una visuale dall'alto e animazioni strepitose.

Ma allora non c'è nulla fuori posto? Nulla di sbagliato? Errori? No, mi spiace non ce ne sono.
Tutto funziona in maniera impeccabile, ed eventuali difetti sono solo di puro gusto personale. Forse si può obiettare con la storia del gioco. La solita basica storia della principessa da salvare (beh... qui in realtà sono 7 le principessa da salvare), ma anche qui si potrebbe rispondere.
E' un gioco d'avventura col prode eroe e già questo è perfetto in questo genere di giochi.

Questo Zelda deve essere giocato una volta nella vita. Rappresenta il "VIDEOGIOCO" (infatti nel 2006 Entertainment Weekly lo inserì al primo posto nella lista dei 100 più grandi videogiochi della storia e rappresenta una testimonianza di come Nintendo abbia dato tutto ciò che aveva da dare nel realizzare un'opera, senza risparmiarsi, conquistando la luna, le stelle, il creato entrando nella leggenda!

di **Carlo Nithaiah Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 99%

Si fa tutto con la croce direzionale e 4 tasti. E si riesce a fare davvero di tutto. Link combatte, spara, usa il retino, suona un'ocarina, martella, corre, infilza, fa magie.
Fa tutte queste cose in un mondo diviso in due ma con una marea di segreti. Così intuitivo e giocabile... Perfetto!

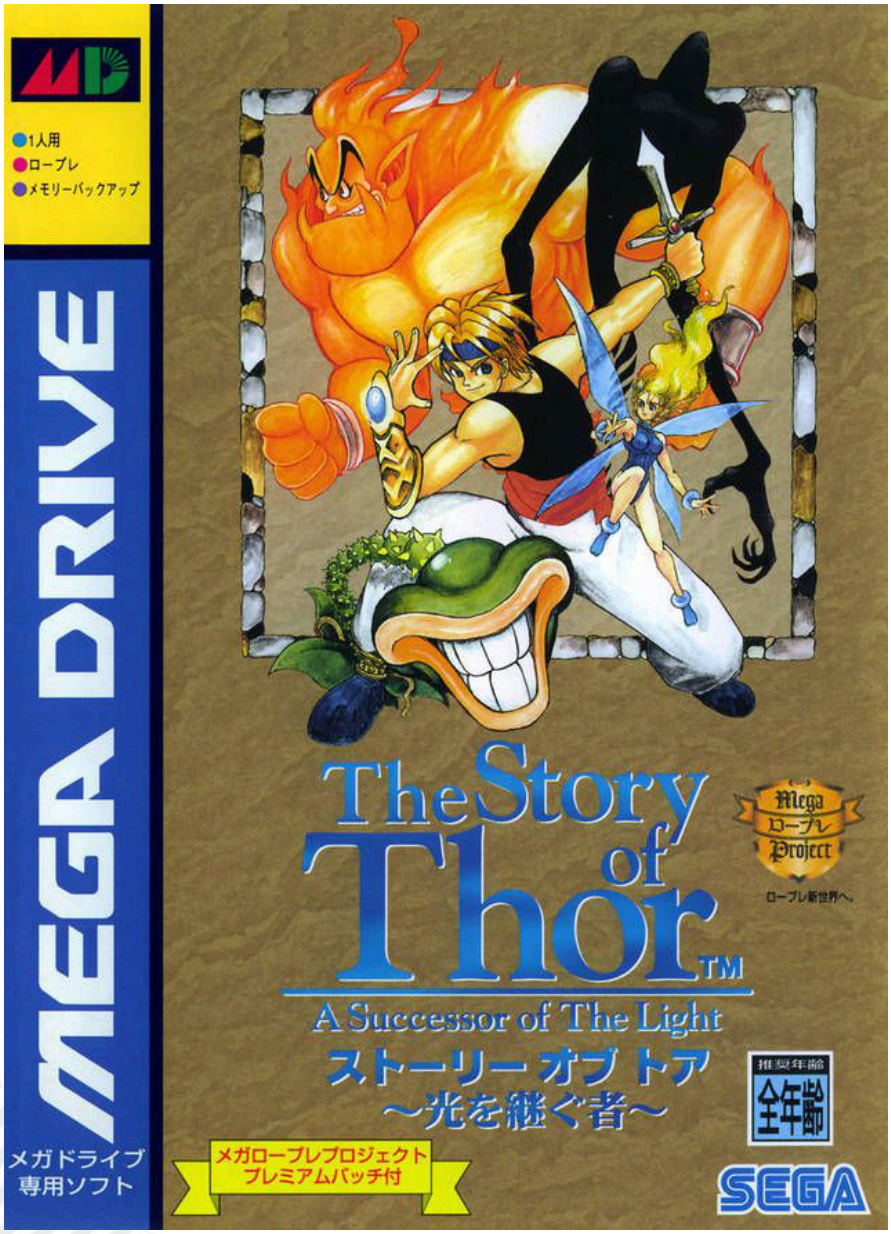
» Longevità 99%

Sono passati 29 anni dalla sua uscita. Nel 1991 ci stupivamo di grafica, sonoro e di come questo gioco ti incollasse al televisore lasciandoti senza fiato per il pathos.
Nel 2020, in piena emergenza covid, Zelda su Snes ha ancora questo potere e sembra non avere tempo. Immortale

THE STORY OF THOR

A SUCCESSOR OF THE LIGHT

Sviluppatore: Ancient
Editore: SEGA
Piattaforma: Sega Megadrive/ Genesis - Rilasciato anche per Virtual Console, Steam e App Android.
Genere: Action RPG



NOTTI D'ORIENTE...

Nel 1994/95, in piena "terza età" dei 16 bit, la Sega pubblico per il su Sega Megadrive questo gioco, chiamato in America "Beyond Oasis" e da noi (Europa) "The Story of Thor".
Sulla scia di Soleil e dei meravigliosi Secret of Mana e Chrono Trigger per Snes, il mercato degli rpg d'azione sul 16 bit Sega era in ottima saluta.

Il gioco è il classico esempio di questo genere che unisce i combattimento in tempo reale ad alcuni momenti di esplorazione e di risoluzione degli enigmi.

Il nostro protagonista è il principe Ali, che a causa di una serie di fortunati eventi si imbatte in uno strano bracciale d'oro situato all'interno di una grotta. Una volta indossato ecco che appare dinnanzi a lui un Jiin, che racconta l'antica leggenda della guerra tra due stregoni, uno dei quali possedeva

questo bracciale e controllava i quattro potenti spiriti della Terra. Contrapposto a lui vi era un malvagio visir possessore di un bracciale identico ma d'argento, che attraverso i poteri magici degli elementali della terra voleva creare il caos per conquistare tutte le terre conosciute. Ben presto il nostro Ali si troverà quindi a dover utilizzare i poteri del bracciale e a dover affrontare il malvagio avversario per salvare il regno di Oasis e tutti i suoi abitanti... P.S. Se mi chiedete dove sia Thor... si tratta di una pessima localizzazione.

Cosa ci troviamo di fronte? Una bellissima ambientazione ricca di particolari e caratterizzata da una grafica di impatto e molto pulita. Ci troviamo come nel caso del già recensito Soleil (vedi Retromagazine nr 21 nd N) ad un ottimo uso delle potenzialità della macchina Sega.
Sempre sul comparto tecnico non posso non citare la colonna sonora di quel genio di Yuzo Koshiro autore già delle colonne sonore di Streets of Rage per Megadrive e di Actraiser per Snes (e di un'altra camionata di titoli). Solo la colonna sonora vale l'esperienza. Incredibile!

Meccaniche per giocare. Se premiamo il pulsante start accederemo ad un menù dove potremo settare l'arma da fare utilizzare ad Ali, controllare i vari oggetti e ripristinare il vostro status. E' inoltre presente una mappa in rilievo che risulta fondamentale per sapere sempre dove andare. Se all'inizio il controllo risulta un po macchinoso, ma se verrà settato bene attraverso l'uso ad esempio del pad a 6 pulsanti, in brevissimo tempo utilizzeremo i tasti e il D-pad per eseguire le numerose mosse e combinazioni (in tipico stile Zelda A Link to The Past per Snes).

Le ore di gioco davanti a noi sono circa una decina e con l'avanzare dei numerosi livelli verrete a conoscenza delle numerose abilità e soprattutto potrete trovare e comandare i sopracitati quattro Jiin, i quali saranno fondamentali per la risoluzioni di alcuni enigmi.
Come in ogni RPG che si rispetti saranno numerosi i dungeon che vi troverete ad esplorare e che si concluderanno con dei boss davvero ottimamente realizzati e carismatici, i quali vi daranno parecchio filo da torcere. Notevole è infatti il character design, generalmente ben oltre la media per colori e animazioni di molti altri giochi del periodo.

Ma tiriamo le somme. Come è invecchiato in questi anni?
Bene, decisamente bene dal punto di vista tecnico e da quello della giocabilità. Gli action rpg, come già scrivevo lo scorso numero, sono quasi in via di estinzione e ben venga il retrogaming a riscoprire questo genere. Penso che se comincerete una partita, vorrete finirlo a tutti i costi. A questo proposito vi posso anticipare che a fine gioco vi verrà assegnato un rango in base ai mostri uccisi e al tempo impiegato per terminarlo. Più giocate e cercate enigmi e segreti e più il punteggio crescerà.

Peccato non esista una localizzazione in italiano (anche per i più piccoli) se non qualche hackrom tradotta non perfettamente.

Passo e chiudo.

di **Carlo Nithaiah Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 80%

Abbastanza lungo per essere finito senza deviazioni, moooolto lungo se viene voglia di trovare TUTTI gli Enigmi.

» Longevità 90%

Un gioco assolutamente da provare. Bella grafica, bel sonoro, buona giocabilità, discreta longevità, bello stile... Che cosa volete di più?

THE SHADOW OVER HAWKSMILL

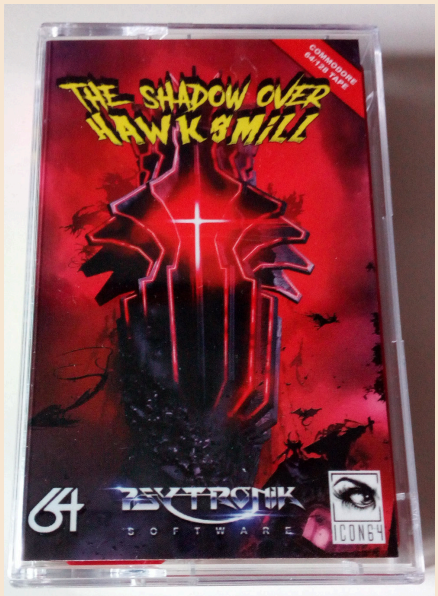
Publisher: Psytronik
Anno: 2020
Piattaforma: Commodore64
Genere: Action/ Adventure

Dalla mia ultima recensione d'un gioco per "Biscottone" ne è passata di acqua sotto i ponti, e confesso di essere sempre entusiasta quando riesco a recensire qualcosa di nuovo, che i più non conoscono. Per carità, sempre valido parlare di qualche classico per presentarlo ad un pubblico nuovo, ma volete mettere la soddisfazione di poter dire "questo è uscito ora!"?
Come per le migliori sorprese, un pacchetto arrivò a casa mia inaspettato il primo d'Aprile e, non ricordandomi neppure di averlo ordinato (erano passati due mesi), hanno cominciato a brillarmi gli occhi nel momento in cui l'ho aperto ricordandomi il giorno in cui ci investii i pochi euro della cassettina.

La Psytronik è una software house inglese che produce titoli per commodore 64, zx spectrum, amiga e tante altre macchine del passato. Questa volta ha voluto puntare su uno dei miei scrittori preferiti: H.P. Lovecraft.
La trama recita: strani avvenimenti stanno accadendo nella piccola città di Hawksmill, nel nord est dell'Inghilterra. L'intera popolazione è svanita dall'oggi al domani! Ad instillarti il sospetto che le forze oscure ne siano responsabili è stata la lettura, nel corso degli scorsi anni, del più grande, celebre ed antico tomo occulto: il Necromonicon. Che sia questo l'evento preannunciato nell'antico manoscritto? Sei determinato a scoprire cosa si celi dietro l'ombra calatasi su Hawksmill.

E' il 1947 ed il nostro eroe si cala in un action-adventure 2D splendidamente disegnato. Con il joystick si fa tutto e se le direzioni destra/sinistra ci permettono di muoverci, sarà cliccando verso l'alto che potremo saltare o salire le scale. La direzione in basso ha invece una moltitudine di funzioni: abbassarci, raccogliere oggetti (sempre celati dietro casse, barili o forzieri), entrare nelle porte, scendere le scale e, combinato con il tasto azione, darci la possibilità di scorrere l'inventario.
Già, perché ovviamente il tasto azione è uno solo e dovremo prima selezionare l'oggetto impugnato (mai più di uno per volta, per necessità di controller) e poi usarlo. Se si tratta di un frutto ci cureremo, se sarà una chiave o un oggetto li utilizzeremo, se infine sarà un arma questa sparerà ma attenzione: tenendo premuto il pulsante azione infatti l'arma continuerà a sparare ma il nostro personaggio rimarrà bloccato sul

Copertina



posto, permettendoci di direzionare lo sparo con il joystick in alto o in basso.
Tutto molto facile da apprendere, tutto adatto a dare profondità all'esperienza di gioco.

L'avventura non è affatto corta, con 60 schermate da esplorare (backtracking a più non posso dato che la maggior parte degli edifici cittadini sono inizialmente bloccati) ed un tasso di mortalità altissimo. Non si salva ovviamente e perdere l'ultimo punto ferita equivale a Game Over, Caput, si ricomincia dall'inizio.
Per fortuna una volta presa la mano ed imparata la posizione dei nemici sapremo sempre dove sparare e quali punti "checkpoint" tenere a mente per ricaricare almeno la scorta di proiettili.
Survival Horror? Quasi, sicuramente sparare a caso non sarà una buona idea se ci saremo addentrati troppo in profondità nelle grotte e la distanza dalle casse di munizioni inizierà a farsi significativa.

Sul fronte tecnico siamo ad un livello altissimo, sia per le grafiche che per gli effetti sonori. Per quanto riguarda le musiche invece avrebbero potuto far di meglio dato che si limitano a fare da sottofondo neutro. Nulla che rimanga in mente.

In definitiva un gioco d'atmosfera, ottimamente realizzato e divertente. Punitivo come i giochi di una volta ma sviluppato con la consapevolezza moderna. Consigliatissimo!

Grafica eccellente
E' un piacere per gli occhi, sia gli scenari che i mostri di ispirazione lovecraftiana. Vi sfido a trovare gli Shoggoth.

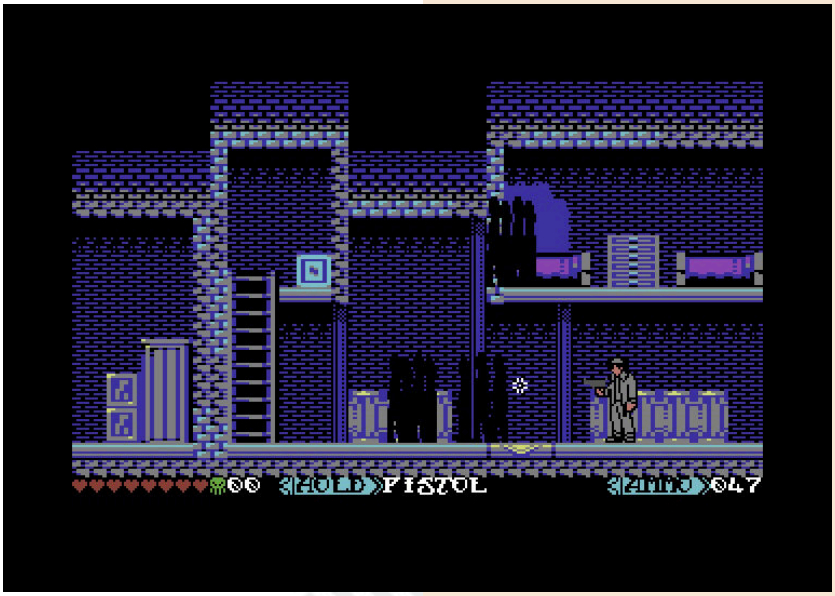
Packaging di pregio
Io ho investito "solo" nella versione standard economica e me ne sono pentito per tutto il ben di Dio che i ragazzi della Psytronik hanno inserito in quella deluxe. Sono comunque fierissimo della mia cassettina.

di **Starfox Mulder**

GIUDIZIO FINALE

» Giocabilità 85%
Facile de apprendere e stimolante nell'esplorazione proposta. Con un joystick ad un solo tasto non so come si potesse fare di meglio.

» Longevità 80%
Punitivo ma sempre affrontabile. Finirlo non richiederà solo una mezz'oretta quando sarete esperti, figuriamoci la prima volta.



ETERNAL DARKNESS: SANITY'S REQUIEM

Sviluppatore: Silicon Knights
Editore: NINTENDO
Piattaforma: Nintendo GameCube
Genere: Azione/Survival Horror Psicologico

"Deep into that darkness peering, long I stood there, wondering, fearing, doubting..." Edgar A. Poe.

Si apre con questa citazione (incompleta) del racconto "The Raven" del poeta Edgar A. Poe l'introduzione di Eternal Darkness: Sanity's Requiem per il Nintendo Gamecube.

Potevamo cominciare recensendo titoli allegri come Super Mario Sunshine o Paper Mario... oppure Zelda o Mario Kart Double Dash, ma sarebbe stato troppo semplice. Troppo! Cominciamo a dare uno sguardo al parco giochi della macchina Nintendo partendo proprio da questo survival horror creato dalle sapienti mani dei Silicon Knights per Nintendo stessa. Precisiamo subito che questo Eternal Darkness non può essere considerato un survival horror classico e ne può essere paragonato a Resident Evil. Eternal darkness può essere inserito in un nuovo genere coniato dagli stessi sviluppatori, ossia horror psicologico: raramente il giocatore sobbalzerà dalla sedia per un improvviso spavento, ma giocando si avrà sempre addosso una terribile (ma piacevole) sensazione di inquietudine, che sfocerà poi in spavento/ stupore con le allucinazioni vera novità di questo gioco.

Una sensazione che richiama i racconti del già citato Poe, di H.P Lovecraft, Arthur Bloch e di quella cerchia di scrittori che raccontavano di orrori differenti, spaziali e mostruosamente inconcepibili.

Partiamo dalla storia... Molto prima dell'arrivo della razza umana il nostro pianeta era abitato da un'altra specie priva di ogni vicolo fisico o naturale. Il passare del tempo, l'avanzata dei ghiacci, il movimento delle placche continentali e altre ragioni a noi sconosciute, hanno determinato l'esilio degli Antichi, che ora giacciono prigionieri in attesa del tempo propizio per il loro ritorno.

Divorati dalla brama di potere e dal fascino degli antichi, alcune sette si adoperano per il loro ritorno, portando così l'umanità a morte certa.
Ma non tutto è perduto, il fato del genere umano è nelle mani di alcuni prescelti, uomini comuni, non eroi, che non potranno però fuggire al loro destino, e dovranno farsi forza per combattere una guerra segreta all'ombra della incosciente razza umana.

Tra i prescelti da oltre duemila anni c'è un cognome che sembra spesso ripetersi quello della famiglia Roivas, che trova in Alexandra l'ultima della dinastia. Ed è proprio lei l'ultima ignara prescelta, ad iniziare le indagini suo malgrado, dopo l'incidente capitato al nonno, trovato dalla polizia nella sua abitazione totalmente privo della testa. Alex prende il primo aereo per Rhode Island, e giura di non lasciare l'appartamento del nonno finche non avrà fatto luce sulla sua orribile morte. Proprio nella casa incominciano le nostre prime esplorazioni nei panni di Alex che ci porteranno in breve tempo al ritrovamento in una stanza segreta di un grosso tomo relegato in pelle umana, il Libro delle Tenebre.

Se la protagonista femminile è la giovane Alexandra, il vero protagonista maschile è proprio il Libro delle Tenebre. Tra le sue pagine si sviluppa il fulcro di tutta la vicenda, è suddiviso in dodici capitoli, che abbracciano 2000 anni di storia, che ci porteranno ad impersonare appunto 12 differenti personaggi.

Il giovane centurione romano Pius Augustus impegnato a combattere in Persia, Ellia schiava dell'imperatore cambogiano, il nobile Karim, il frate (Paul) con il suo capitolo ambientato durante l'inquisizione, il templare Antony, l'architetto veneziano Roberto

Bianchi, l'archeologo Edward e in periodo più vicini a noi il reporter Peter (durante la prima guerra mondiale) il pompiere Michael (1991) e poi numerosi membri della famiglia Roivas, Maximilian (medico) Edward (psichiatra) e la stessa Alex.

I personaggi sono tutti ottimamente caratterizzati, diversi per caratteristiche fisiche e psicologiche che influenzeranno notevolmente il loro controllo.
Per esempio i personaggi dalla corporatura più robusta e quelli più anziani, si stancano se corrono troppo, e necessitano quindi di fermarsi per qualche secondo per recuperare le forze, e quelli più giovani e meglio dotati fisicamente saranno sotto questo punto di vista ben più resistenti.

Le differenze sussistono anche da un punto di vista psicologico: tipi come Edward (lo psichiatria), troppo ancorati alla realtà avranno in caso di visioni incredibili e fuori dal normale un decremento della loro sanità mentale molto elevato, altri come Paul il frate e quindi uomo di chiesa avranno nella loro fede un sostegno psicologico quasi inesauribile.

Il gioco fu inizialmente pensato per una sua uscita su Nintendo 64, ma visto i tempi "biblici" degli sviluppatori si penso di adattarlo per una sua uscita su Gamecube.

La sua resa grafica ovviamente risente di alcune limitazioni dello sviluppo precendente, ma è comune ben realizzata.

La grafica, comunque va migliorando col procedere del gioco, riuscendo talvolta anche a stupire per effetti di luce (legati soprattutto all'uso della magia), e per l'illuminazione in tempo reale tramite torce.
Buona anche la qualità dei filmati in computer grafica, presenti in buona misura, ma mai troppo lunghi o invadenti, che contribuiscono ancor più ad un'immedesimazione (e in alcuni casi vi farà sobbalzare dalla sedia).

Per il sonoro è stato fatto un lavoro quasi maniacale; musiche adattissime (imperdibile nel capitolo ambientato nel medioevo i sublimi canti gregoriani), rumori di sottofondo davvero in tema (urla, lamenti) e un doppiaggio eccezionale.

Quest'ultimo tutto in inglese (ma vanta comunque sottotitoli perfettamente tradotti in italiano), è eseguito da doppiatori professionisti, il parlato è perfettamente sincronizzato con il labiale dei personaggi, è presenta addirittura un breve dialogo tutto in latino.

L'esplorazione del gioco vi porterà via diverse ore di gioco (circa una ventina), ma per poter gustare il vero finale del gioco (che vi assicuro è incredibile) bisognerà rigiocarlo per tre volte di fila e seguire i piccoli ma notevoli cambiamenti di trama.
Insomma è una bellissima avventura adulta che vi terrà incollati allo schermo e al vostro pad.

Consigliatissimo per chi è dotato del Cubo Nintendo o chi vuole giocarlo su emulatore Dolphin. Purtroppo non è mai stata prevista una versione per le numerose console virtuali ne sui sistemi Wii e WiiU ne sulla Switch.

di **Carlo Nithaiah Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 80%

Non per deboli di cuore! Non per i mostri o lo splatter, ma per quella forte sensazione di claustrofobia, disagio e "sanità mentale" che vedrete scorrere sullo schermo. I comandi sono all'inizio da apprendere bene, ma alla lunga vi ci ritroverete.

» Longevità 90%

Venti ore di gioco non sembrano tante, ma per capire il vero senso del gioco bisogna ripeterlo per tre volte. Inoltre si ha la possibilità di giocare con tanti personaggi differenti e tutti ottimamente caratterizzati. Geniale.

SUPER VOLLEY

Sviluppatore: V-System/Data East
Anno: 1989
Piattaforma: Arcade
Genere: Simulazione sportiva

Non sono tanti i videogiochi dedicati alla pallavolo indoor ad essere ricordati.
Probabilmente, Super Volleyball – oltre ad essere uno dei primi titoli dedicato a questo sport meraviglioso – è uno dei migliori giochi dedicati di sempre.
Giusto fare un distinguo netto per evitare equivochi: la pallavolo indoor è quella disciplina che si gioca 6 contro 6 nelle palestre e nei palazzetti dello sport che in Italia i bambini degli anni '70 ed '80 hanno imparato a conoscere grazie a cartoni animati famosi quali Mimi e la nazionale di pallavolo e successivamente con Mila e Shiro.
Un prodotto atipico e comunque diverso dai numerosi videogiochi rivolti al beach volley (due contro due su piaggia) che imperversavano anche in quegli anni e negli anni '90 nelle sale e nei sistemi casalinghi di tutto il mondo.

Sviluppato dalla giapponese V-System, Super Volleyball venne poi lanciato in sala giochi distribuito dalla stessa software house giapponese in tutto il mondo e negli Usa da Data East.

Andiamo subito al sodo: la chiave del successo sta nella immediatezza del gameplay unitamente alla sua profondità nonché alla sua curva di difficoltà non troppo ardua ma neppure – avanti nella competizione – troppo facile.

Il compito non è facile: portare il Giappone a vincere la medaglia d'oro ai mondiali (che per inciso nella realtà si disputarono l'anno successivo e videro il primo dei tre trionfi iridati consecutivi della nazionale Azzurra).

Per farlo bisogna battere tutte le avversarie. Le sfide sono degli scenari che ci portano nelle fasi finali dell'ultimo set, ovvero al tie-break e quindi senza cambio palla.

La scalata all'oro comincia con Cuba, poi Cina, Stati Uniti, URSS (all'epoca era così) per poi entrare nella fase finale. A questo punto, già sicuri del bronzo, si riaffrontano gli Stati Uniti in semifinale e l'Unione Sovietica nella finalissima. Se si perde con gli Usa si conquista il terzo gradino più basso del podio. Se si va in finale, è chiaro il risultato minimo sia l'argento o l'iride in caso di successo.

Più avanti si va, più difficili sono gli avversari e più ampio è il distacco da recuperare: dall'11-11 con Cuba all'8-11 con l'Urss nel torneo di qualificazione. Nelle due partite finali, invece, si parte dal punteggio di parità: 8-8. Altra condizione per vincere: bisogna farlo entro 2 minuti e 30 secondi. Che arcade sarebbe sennò?

Il gameplay è piuttosto intuitivo e sfrutta piuttosto bene l'inquadratura da sinistra a destra a scorrimento laterale. Si parte dalla battuta ed a seconda della posizione dello stick e del pulsante si effettuano diversi tipi di servizi: di sicurezza, normale, o al salto.
Ci sono anche tre servizi speciali: una battuta al salto decisamente aggressiva ed in grado di andare a punto almeno la metà delle volte nelle prime due-tre partite, una a pallonetto che fa addirittura volare la palla oltre l'inquadratura della camera con un bell'effetto grafico di profondità del soffitto del palazzetto, ed il servizio Fantasma – che chi vi scrive chiama a foglia morta – che fa certamente punto nei primi tre incontri e che si può fare soltanto una volta. Quest'ultimo è molto spettacolare a livello visivo con la sfera che sembra moltiplicarsi. Generalmente questo è il colpo finale per ovvi motivi tattici. Ma non funziona contro l'URSS.

La fase del gameplay durante la partita prevede l'utilizzo di un giocatore in ricezione, poi dell'alzatore che può –

sempre a seconda dei nostri comandi – dettare diversi tipi di battuta. Può fare la classica veloce, molto efficace quasi sempre, o di aprire allo schiacciatore che viene dalla seconda linea (spettacolare quanto pericolosa perché se murati non c'è nessuno a ricevere la sfera) o fare la finta per mandare fuori fase il muro avversario o, ancora si può tentare l'attacco in seconda (sfruttando l'effetto a sorpresa) o ancora tentare il pallonetto più o meno velenoso.

Anche in fase di ricezione si possono fare molte mosse: c'è il tuffo sia in avanti che indietro ma anche il muro. Purtroppo, può anche capitare che questo sfiori la palla e questa voli via senza possibilità di essere rigiocata. Ma se fatto con tempismo può essere un attacco micidiale.

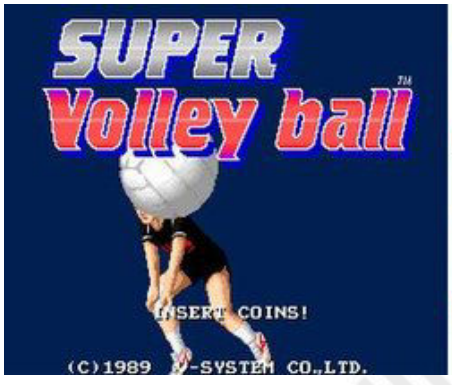
Come dicevamo, il gameplay è piuttosto interessante perché permette tante possibilità nonostante un relativo numero di mosse. Perché ogni tanto anche l'intelligenza artificiale fa degli errori ed è tutto piuttosto divertente.

A rendere Super Volleyball memorabile anche la particolare grafica molto cartoonesca con la presentazione del sestetto dei nostri eroi giapponesi e poi un riquadro col ritratto del giocatore che batte, ed un altro dedicato a chi fa punto. Il campo, i giocatori ed il palazzetto dello sport sono ben disegnati. Belle e complete anche le animazioni di atleti in campo e del pallone. Non mancano anche alcune divertenti schermate di intermezzo, soprattutto in caso di sconfitta.

Ulteriore ciliegina sulla torta è rappresentata dal comparto sonoro. Le varie musichette di presentazione sono così orecchiabili che era davvero complicato togliersele dalla mente. Riuscitissima quella di presentazione ma anche quella durante la partita. Inoltre, ad ogni attacco riuscito c'era un commento dedicato: "Nice Block" (Bel muro), "Nice Spike" (Bella schiacciata). Poi due frasi che però non siamo riusciti a capire quando si eseguivano le battute a pallonetto ed a foglia morta e le frasi minimali dell'arbitro che segnalava quando il pallone toccava, o usciva (perché sia gli attacchi che le battute potevano anche andare oltre le righe del campo) o se veniva toccata dal muro.

Insomma, per essere un titolo arcade, Super Volleyball, ha regalato davvero tanto agli appassionati di sport e pallavolo grazie ad un mix riuscitissimo che pur oggettivamente non eccellendo in nessun comparto, è stato in grado di trovare un posto nella storia. Forse anche più dei due sequel che migliorarono ed ampliarono notevolmente la summa tecnica e la varietà del gameplay e parliamo di Power Spike I e II giunti nel 1991 e nel 1994.

di **Edoardo Ullo**



GIUDIZIO FINALE

» Giocabilità 85%

Titolo adatto a tutti e con una buona varietà di colpi. Non tantissimi ma neppure pochi con una difficoltà spalmata e comunque equilibrata senza troppi momenti impossibili.

» Longevità 75%

Solo sei partita da vincere ma stiamo parlando di un gioco arcade. Peccato si possa scegliere soltanto un team.

TEHKAN WORLD CUP

Sviluppatore: Tehkan, adesso Tecmo
Anno: 1985
Piattaforma: Arcade
Genere: Simulazione sportiva

Questo numero sembra dedicato alle simulazioni sportive arcade. Dopo SuperVolleyball, uno dei giochi arcade preferiti di Edoardo, è il mio turno di parlarvi del mio gioco arcade preferito in assoluto.

A questo gioco sono legati i miei ricordi più belli da sala giochi e, grazie a questo gioco, ho conosciuto degli amici che, dopo quasi 35 anni, tuttora frequento.

Ma cominciamo dall'inizio; siamo tra il 1985 ed il 1986, frequento le scuole medie ed Ivan è ormai un ex compagno delle elementari, ma come nella migliore delle tradizioni continuiamo a frequentarci. I genitori di Ivan gestiscono un bar e collegato a questo vi è una delle più grandi sale giochi della mia città (Poggibonsi) di quel periodo. Ovviamente l'occasione è troppo ghiotta per non approfittarne, così la loro sala giochi è il nostro punto di ritrovo e di aggregazione. In quella sala giochi, che conteneva circa una ventina di giochi, ho scoperto molti degli arcade a cui sono per ovvie ragioni molto affezionato. Uno di questi è Tehkan World Cup!

Ricordo ancora la prima volta che lo vidi; era uno dei primi arcade cocktail mode che vedevo, quindi attrasse immediatamente la mia attenzione. Un gioco del calcio a volo d'uccello, con una grafica meravigliosa e dei colori sfavillanti! La sua musica poi era molto alta, quindi riecheggiava poderosa nella sala. Non tardai molto ad inserire le 300 lire richieste (ebbene sì, rispetto agli altri giochi che funzionavano con 200 lire questo accettava monete da 100) ed a fare la mia prima partita. La ricordo ancora come se fosse ieri; in un modo o nell'altro riuscii a passare di misura le prime due squadre, ma alla terza persi inesorabilmente.
Poco male, avevo tutto il tempo per imparare le meccaniche di gioco e completare l'opera. Non tardai molto a capire come segnare alle prime squadre ed arrivare quasi sempre in finale. Da lì in poi, a vincere il campionato del mondo, il passo fu abbastanza breve.

In singolo
In breve il gioco in singolo si compone di 7 match della durata di 1:30 minuti ciascuno. Affronteremo nell'ordine le seguenti 6 squadre: Scozia, Unione Sovietica, Brasile, Germania Est, Francia ed Uruguay per scontrarci in finale la temibile Germania Ovest.
Nel mio immaginario di ragazzino le squadre erano altre però; per esempio la quarta era l'Italia, la quinta la Francia e la sesta l'Argentina... Ma questo ovviamente cambia poco al fine del gioco.

Non sono previsti falli e le possibilità di tiro si limitano al tiro secco ed al pallonetto. Entrambi possono essere effettuati in verticale, orizzontale e diagonale. Non c'è possibilità di passaggio se non in una di queste direzioni.

Segnare alle prime tre squadre è abbastanza semplice ed abbiamo a disposizione diverse possibilità. La più ovvia è il tiro in diagonale, con cui possiamo trafiggere tutte le squadre, quello che cambia per le prime tre squade è la distanza di tiro. Sono riuscito a segnare in diagonale tirando quasi dalla riga del fallo laterale. Inoltre, se siamo abbastanza vicini alla porta, possiamo segnare anche con il pallonetto. Provateci, l'effetto della palla che si alza e si insacca all'incrocio è visivamente gratificante. Mi raccomando dovete essere molto vicini alla porta perchè la palla tende ad alzarsi molto velocemente. Vi ricordo che per fare il pallonetto dovete premere leggermente il pulsante di tiro.

Alle prime tre squadre inoltre è

possibile segnare tirando dritto per dritto a fil palo, tra palo e portiere. Con le prime due è possibile segnare così persino da fuori area.

Mi sembra di ricordare che sia possibile segnare alle prime due avversarie anche con il pallonetto in fase discendente; in questo caso dovete trovare la piazzola giusta, oltre la metà campo, perchè il pallonetto è molto lungo.

Gli altri modi per segnare vanno bene con tutte le squadre. Il tiro in diagonale è efficace contro tutti i gli avversari controllati dal computer, ma in finale dovrete essere veramente vicini alla porta per sorprendere il portiere avversario. Cosa peraltro non semplicissima visto il numero di avversari che cercheranno di soffiarvi il pallone. Inoltre potete segnare a tutte le squadre effettuando un cross laterale verso l'area di rigore per l'attaccante accorrente. Se sarete fortunati l'attaccante si troverà qualche passo avanti rispetto alla palla e si produrrà quindi in una fantastica rovesciata, molto scenografica, altrimenti segnerà di piede, di testa oppure in mezza rovesciata.

Il sonoro è essenziale, tranne la musica dell'introduzione che è decisamente accattivante; ancora oggi mi sorprendo a fischiettarne il motivetto!

Tutto bello? Quasi, perchè dei difetti ci sono.
Man mano che si prosegue nel gioco, le squadre controllate dal computer tendono a giocare come i 'pulcini' (categoria dei bambini); vanno tutti dietro alla palla, o meglio al portatore di palla avversario. Inoltre, la velocità dei giocatori controllati dal computer, da un certo punto in poi, è più elevata di quelli della nostra squadra, obbligandoci a continui cambi di direzione per evitare le scivolate avversarie. Come se non bastasse i giocatori avversari sono in grado di effettuare colpi a noi proibiti, come per esempio tiri con velocità controllata o diagonali a noi impedite... Difetti che comunque a mio avviso non minano la grande giocabilità di questo gioco.
Forse la pecca che potremmo ascrivergli è di essere un po' troppo facile e dopo poche partite chiunque sarà in grado di arrivare in finale e vincere.

Va inoltre ricordato un bug che affligge soltanto l'ultima partita: alcune volte il giocatore con la palla, controllato del computer, si dimentica del pallone e continua a correre come se niente fosse. Il problema è che il tempo va avanti senza che noi abbiamo la possibiltà di fare alcunchè, mentre la palla rimane immobile da qualche parte. Molto fastidioso... Possiamo però rimediare tentando di recuperare la sfera utilizzando il radar e cercando 'a naso' nella zona di campo dove questa è stata smarrita. A volte funziona!

In 'doppio'
Ma vi ricordate che all'inizio della recensione vi ho detto che grazie a questo gioco ho fatto diverse amicizie? Come? Non tardammo molto a provare a fare un doppio io ed Ivan. Con 600 lire potevamo giocare in due per ben 3:00 minuti. Ed è proprio nel doppio che questo gioco da il meglio di sè! Niente colpi proibiti, niente avversari con velocità maggiore rispetto alla nostra, niente trucchetti... no, no, i trucchetti ci sono eccome nel doppio. Ci sono alcuni punti del campo da cui poter tirare senza che il nostro avversario abbia la possibiltà di parare il tiro. Mi ricordo anche della possibilità di tirare addosso ad un giocatore avversario a centrocampo con la palla che andava inesorabilmente ad insaccarsi senza che il portiere potesse intervenire...

Nonostante i trucchetti, che cercammo di limitare con un apposito regolamento, cominciammo a sfidarci in estenuanti partite per cercare di capire chi fosse il migliore, finchè altri avventori della sala giochi non tardarono ad unirsi a noi per misurare la loro capacità.
Ideammo anche una speciale classifica a punti chiamata AGP - Associazione Giocatori Professionisti, di cui appendemmo il cartellone con i nostri nomi, la posizione ed il punteggio nella parete sopra il gioco. La sala giochi era del bar dei genitori di Ivan, potevamo farlo. :-)

Tra i ragazzi che si unirono a questa speciale competizione, ricordo tra gli altri Fabio ed Antonio. Come vi ho detto all'inizio, siamo tuttora amici e tutto è cominciato proprio così, nella sala giochi di un bar, di fronte ad un gioco del calcio.

Grazie Tehkan World Cup!

di **Francesco Fiorentini**

GIUDIZIO FINALE

» Giocabilità 85%
Il più vario gioco del calcio arcade, con una discreta possibilità di giocate e di realizzazioni. Grafica accativante e musichetta d'introduzione deliziosa.

» Longevità 75%
Solo sette partita da vincere, ma stiamo parlando di un gioco arcade. Aggiungere 15 punti per il doppio!

ASTERIX AND THE MAGIC CAULDRON

Anno: 1986
Sviluppatore: Melbourne House

Piattaforma: C64
Genere: Avventura

Proseguiamo e non ci fermiamo con la nostra marcia dopo circa un mese di quarantena e si spera che all'uscita della rivista ci sarà un netto calo di contagiati e attività verso il nuovo avviamento senza ripercussioni economiche.

Voglio dedicare questo articolo non solo a voi costretti a casa a combattere contro la noia, ma anche ad Albert Uderzo, creatore di Asterix e venuto a mancare da poco, ringraziandolo doverosamente per averci dato tanto; più di tutto i suoi simpatici personaggi.

Visto il successo dei fumetti, serie animate e gadget, non poteva mancare un videogame per i vari home computer di allora. Alla morte del creatore mi sono subito posto la domanda se fosse esistito un gioco anche per il Commodore 64, così eccomi accontentato!
Asterix and the magic cauldron, il titolo uscito nel 1986 per Commodore 64, ZX Spectrum ed Amstrad CPC oltre a varie versione per console.

Il giocatore controlla Asterix in varie ambientazioni come la foresta, gli accampamenti e perfino Roma. Il nostro compito è quello di trovare tutti i sei pezzi del calderone magico che servirà a Panoramix per creare la pozione necessaria per avere la meglio sui centurioni romani.
Il gioco si svolge su ambientazioni a schermata fissa multidirezionali e in stile labirinto in cui ci saremo noi, il fidato Obelix (anche se la sua presenza è solo a scopo decorativo) ed i nemici come i cinghiali ed i centurioni romani.

Oltre a non perdere energia e vite, bisognerà fare attenzione che la barra della fame, contrassegnata da un pollo arrosto, non vada esaurita, altrimenti si perderà una vita. Consiglio quindi di cercare anche i polli in giro per i livelli oppure di ingaggiare i combattimenti contro i i cinghiali.

Nelle schermate inoltre si troveranno anche delle chiavi per accedere a zone chiuse che saranno però sempre ben sorvegliate dai centurioni pronti a combattere!

Il sistema di combattimento mi ha ricordato i giochi di ruolo attuali, anche se non sono arrivato a battere più di due centurioni alla volta senza perdere vite. Forse una parte della difficoltà del gioco sta proprio in questo, a parte le strade che cambiano in continuazione rendendo quindi molto difficile la ricerca dei pezzi del calderone.

Per fortuna avremo una fiala di pozione magica che ci renderà invincibili e in grado di fare a fettine più centurioni possibile, peccato che si può utilizzare una sola volta per partita, quindi siate saggi e parsimoniosi! Usarne una per battere due cinghiali… Ehm.

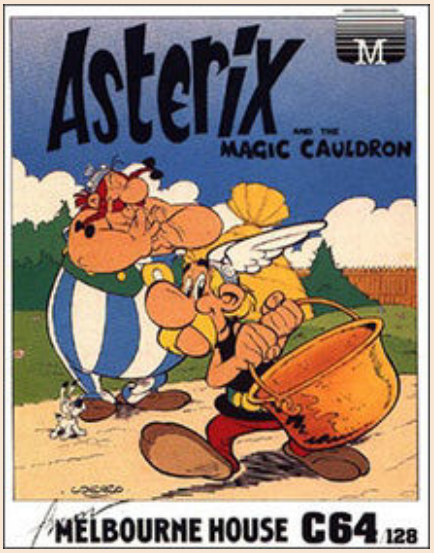
Graficamente non mi ha colpito molto anche se devo dire che non mi aspettavo molto di più, d'altronde la grafica era quella e si era felici con poco negli anni 80; poi Asterix è Asterix!

L'unico difetto, se così vogliamo dire, a parte la difficoltà dei combattimenti è il caricamento un po' lunghetto tra una schermata e l'altra… Ma almeno allunga un po' il tempo di gioco in queste giornate da reclusi.

Se proprio non riuscite ad andare avanti e siete all'esasperazione, andate a vedervi il gameplay su qualche sito, per capire l'ubicazione dei primi due o tre pezzi del calderone.

Ancora una volta, vi auguro una serena rinascita, tante cose belle vedrete che andrà tutto bene!

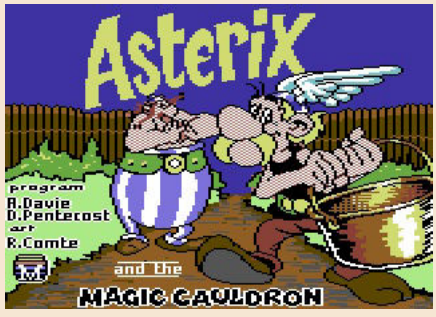
di **Daniele Brahimi**



GIUDIZIO FINALE

» Giocabilità 65%
Combattimenti non facili, ma bella l'esplorazione.

» Longevità 75%
Difficile al punto giusto, ma caricamenti molto lenti, oserei dire biblici...



SWOOP

Sviluppatore: David Elliott
Distributore: MicroPower
Anno: 1982
Piattaforma: BBC Micro, Acorn Electron, C64
Genere: Shoot'em up

Ho giocato di recente a questo titolo dopo diversi anni, e ne approfitto quindi per recensirlo e farlo conoscere ai lettori di RetroMagazine.
Non è stato semplice ritrovarlo dopo tutti questi anni perchè lo ricordavo registrato all'interno una cassetta color bianco panna con un titolo "Eagles", cassetta quest'ultima che ricordavo fosse oltretutto originale.
Purtroppo, con una certo disappunto, non riuscivo a trovare alcuna traccia di questo gioco partendo da questi elementi che ricordavo.
Finalmente, lanciando un titolo presente nel Gamebase64, ovvero il gioco **Swoop**, rivedo la familiare schermata racchiusa in una cornice colorata che scorre lungo il perimetro dello schermo, accompagnata da una musica di sottofondo che è un discreto arrangiamento della celeberrima "Fuga in D minore" di Bach.
Si tratta di un gioco che uscì inizialmente nel 1982 per micro BBC, nel 1983 per Acorn Electron ed infine nel 1984 per C64 e fu uno dei primi giochi che giocai sul mio biscottone, e la versione che recensisco è proprio quest'ultima.
Veniamo ora a descrivere il gioco in maniera più dettagliata.

Si tratta essenzialmente di uno dei tanti cloni esistenti del famoso titolo **Galaxian**. Gli alieni invasori assumono la forma di uomini uccello che sbattono le ali. Swoop differisce dagli altri cloni di Galaxian in quanto questi "Birdmen" deporranno uova esplosive se raggiungono con successo il fondo dello schermo.
Se le uova vengono toccate dalla navetta del giocatore viene persa una vita, limitando quindi efficacemente il movimento della navicella del giocatore fino a alla loro scomparsa, trascorso un certo periodo di tempo prestabilito, ed aggiungono pertanto un grado di difficoltà in più.

La dinamica del gioco è estremamente semplice ed intuitiva, come molti dei giochi di quell'epoca, e si sviluppa senza particolari sorprese.
Gli elementi da tenere in considerazione durante il gioco sono i seguenti:
- come già detto , è necessario evitare che i "birdmen" raggiungano il bordo dello schermo perchè quando ciò avviene appariranno le famigerate uova esplosive che non devono essere toccate, pena la perdita di una vita;
- gli attacchi dei "birdmen" si fanno sempre più efficaci e veloci via via che si aumenta di livello, come avviene di consueto;
- A colore differente corrisponde una modalità di attacco differente, in perfetto stile "Galaxian".

Non è assolutamente facile superare il terzo/quarto livello senza "aiutini", chiamiamoli così.
La versione presente nel Gamebase vi consente di giocare anche in modalità "cheat", non perdendo vite durante il gioco, non preoccupatevi :)
Scoprirete così che alla fine del 31-mo livello il gioco inizia nuovamente dall'inizio. Non è stato previsto quindi un vero e proprio finale, peccato.

E' un gioco veloce, che ha una dinamica di gioco intuitiva, semplice. Non sfrutta certamente tutte le potenzialità dell'hardware del C64, come del resto molti dei primi giochi che uscirono all'epoca, tuttavia è un gioco che caratterizza bene il genere ed il livello di giocabilità di un gioco di quegli anni.
E' un gioco che probabilmente piacerà provare sia a chi lo ha conosciuto in passato sia agli estimatori del genere "Galaxian" che potrebbero apprezzarlo anche per il grado di difficoltà in più introdotte dalle famigerate "uova" deposte a fondo schermo, insomma è un clone di "Galaxian" più tosto del solito. Un saluto a tutti!

di **Marco Pistorio**

GIUDIZIO FINALE

» Giocabilità 90%

Semplice, veloce, intuitivo, colorato ma non troppo, consigliato per chi ha amato "Galaxian" e ne apprezza un suo clone "tosto"

» Longevità 70%

Lo sforzo di far meglio, di progredire di livello, di accumulare un punteggio migliore, di perfezionare i movimenti, gli attacchi e la difesa vi farà giocare qualche partita in più. La mancanza di un vero e proprio finale è un fattore che ha il suo peso.

"Chi vuol esser lieto, sia, di doman non c'è certezza"

(Canzona di Bacco, Lorenzo De' Medici, a.d. 1490)

In questi giorni riflettevo un po'.

E' probabile che il famigerato COVID-19, che è entrato prepotentemente nelle cronache di questi giorni, sia una delle concause che mi ha portato a queste riflessioni.

Ci si affanna a raccogliere ed a mettere da parte per il futuro tutto quello che riteniamo più prezioso.
Vecchie console di gioco, nuove riedizioni delle stesse, retrocomputer, periferiche varie e via discorrendo.
Ma anche intere raccolte di giochi, collections, TOSEC per le più disparate piattaforme, sia in formato digitale che fisico a volte.

Tanto per intenderci, il solo Gamebase64 consta di 26.900 titoli nella versione più attuale, la versione 16.
Se provassimo 1 titolo al al giorno, sapete quanto impiegheremmo per provarli tutti? Circa 73 anni!
Oppure ancora, se volessimo impiegare un solo anno per giocarli tutti, ne dovremmo giocare circa 73 per ogni singolo giorno dell'anno!

Non sarebbe più lungimirante da parte nostra cercare di godere giorno per giorno di quello che abbiamo, invece di limitarci a mettere tutto da parte per un ipotetico domani?

Altro elemento da tenere in conto: è saggio pensare che i nostri riflessi, la nostra mano, i nostri occhi, tra 20 o 30 anni si comportino come si comportano oggi?

La risposta a queste domande spetta a voi, cari amici nonchè retroappassionati lettori.

La risposta che mi sono dato io la trovate nei versi, che trovo molto saggi, tratti da un famoso componimento poetico di diversi secoli fa.

Un saluto da parte mia e di tutta la redazione di RetroMagazine, insieme al messaggio di speranza che è diventato una specie di mantra di questi giorni difficili: "Andrà tutto bene"

Marco Pistorio

Disclaimers

RetroMagazine
Anno 4 - Numero 22

Direttore Responsabile
Francesco Fiorentini

Vice Direttore
Marco Pistorio

'APRILE 2020'