World RetroMagazine
future days are back
Numero 27 Anno 4 - Dicembre 2020 - WWW.RETROMAGAZINE.NET - Pubblicazione gratuita

Merry Christmas 2020

da tutta la redazione





In questo numero

Giochi

Oltre 70 pagine !

Feste natalizie a 8 bit

Dopo un anno come questo, passato per lo più in casa o con la mascherina sul viso, può capitare di non avere molta voglia di festeggiare le vicine feste natalizie, se non per altro per il numero impressionante delle vittime di questo maledetto virus. Siamo a metà dicembre, ma non è la stessa atmosfera degli anni scorsi. Questa pandemia ci ha psicologicamente fiaccato, mentalmente sfinito. Se l'anno scorso, come sempre, l'orizzonte del futuro ci appariva sgombro e senza particolari nuvole minacciose, ora, nonostante l'ormai prossimo arrivo dei vaccini che dovrebbero segnare il "punto di ritorno" ad una presunta normalità, tutto ci appare come ammantato di un freddo telo trasparente da cui dobbiamo ancora fuggire. E non si tratta del tipico freddo dell'inverno che si annuncia, non è colpa della neve che ha già ricoperto alcune località in molte parti d'Europa. Tutto il pianeta è ferito, le nostre città ci appaiono spente, tutte le luci e le decorazioni delle nostre città ci sembrano meno festose, la neve caduta da qualche giorno non porta più quell'allegria e quella spensieratezza tipica delle vacanze di fine anno, le fredde temperature dell'inverno incipiente non hanno quel sapore che ci fa dire che nell'aria è arrivato il Natale.

È un po' come vivere in un mondo a 8-bit, ad una bassa risoluzione grafica, pochi sprite, pochi colori o addirittura in bianco e nero. Soltanto un anno fa il mondo come lo conosciamo si presentava come un potentissimo PC con CPU a 64-bit, multi-core, display 4k, GPU velocissima capace di visualizzare milioni di colori, colonna sonora full-stereo. E diciamolo, anche con poca anima. Le festività che stanno per arrivare ci mostrano invece una realtà più disadorna, austera, meno votata al consumismo sfrenato e meno sfavillante rispetto a quella cui siamo abituati.

Ma ricordate quanto divertimento e quanta fantasia accendeva in noi la realtà degli anni '80, quelli in cui i nostri amati 8-bit, con i loro 16 colori, chip sonori gracchianti e tastiere gommose, ci riempivano le giornate? Non era tutto già pronto e disponibile davanti ai nostri sensi. Mezz'ora per caricare un gioco da cassetta, dischetti spesso illeggibili dopo pochi utilizzi e joystick che si rompevano in continuazione. Era tutto più frugale e modesto e dovevamo usare molta immaginazione per credere che certi sprite o sfondi grafici avessero almeno una parvenza di realtà. Ricordate le copertine dei giochi? Immagini fantastiche che facevano correre la nostra fantasia, usate spesso per compensare la povertà grafica, sonora e dell'animazione del videogame incluso. Però ogni giorno era come una rivelazione, un passo avanti, una passione rinnovata dalla continua voglia di imparare cose nuove e di fare nuove scoperte.

E allora, in queste feste natalizie "a 8-bit" proviamo a ripartire tornando ad usare la nostra inventiva, la creatività e la passione che abbiamo dentro per accendere di nuovo i colori delle feste. Un modo per farlo è quello di godersi questo pazzesco numero 27 di RetroMagazine World, con oltre 70 pagine piene di sorprese, di novità e di esclusive.

Buone feste a tutti ed un augurio speciale per il prossimo anno. Che sia portatore di una vera rinascita e ci conduca ad una nuova epoca di solidarietà e amicizia. E così potremo festeggiare le prossime vacanze di Natale nella giusta atmosfera.

David La Monaca

SOMMARIO

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

- Alberto Apostolo
- Gianluca Girelli
- Michele Ugolini
- Carlo N. Del Mar Pirazzini
- Daniele Brahimi
- Flavio Soldani
- Francesco Fiorentini
- Attilio Capuozzo
- Marco Pistorio
- Leonardo Miliani
- David La Monaca
- Giovan Battista "giomba" Rolandi
- Antonino Porcino
- Phaze101
- Starfox Mulder
- Simone Battaglioni
- Hakim Rezki
- Roberto "Il Bardo" Del Mar Pirazzini
- Copertina a cura di Flavio Soldani

NINTENDO 64 - La rivoluzione tridimensionale di Mario

di Carlo N. Del Mar Pirazzini

Il Nintendo 64 è stata l'ultima console a cartucce della Nintendo, la console di quinta generazione nata come erede di quel Super Nintendo che rivoluzione il mondo dei videogames durante gli anni 90.

Il N64 fu prodotto da nintendo tra il 1996 e il 2002, usci per primo in Giappone e successivamente negli Stati Uniti (in settembre del 96). Sul mercato europeo arrivo nel marzo del 1997. Per molti bambini nati tra la fine degli 80 e l'inizio dei novanta rappresentò il regalo più ambito per Natale e per un certo periodo fu il rivale numero uno della Playstation 1, salvo poi arrivare secondo nella guerra delle console di quella generazione (annichilendo in vendite il Sega Saturn).

Lo sviluppo dei titoli per il nintendone terminò nel 2002 con la pubblicazione di The Legend of Zelda Wind Waker e Resident Evil 0, usciti anche in versione migliorara su GAME CUBE (abbiamo parlato del GAME CUBE su RMW n° 22 italiano, ndR).
Con 32,93 milioni di esemplari venduti in tutto il mondo è ad ogni modo un numero impressionante. Il Nintendo 64 segnò di fatto una vera e propria rivoluzione nel campo dei videogiochi, proponendo la prima vera esperienza tridimensionale di gioco con titoli di elevata qualità e tuttora autentiche pietre miliari del loro genere, ed ebbe riscontri positivi in tutto il mondo. Tuttavia, negli ultimi anni di vendita fu sorpassato dalla rivale PlayStation, che stabilì un record vendendo oltre 70 milioni di unità, ma con risultati nettamente superiori a quelli del Sega Saturn le cui vendite raggiunsero solo 9 milioni di unità.

La macchina si presenta come una scatola rettangolare di colore nero (ma durante gli anni uscirono anche edizioni speciali caratterizzate da colori diversi e anche trasparenti, ndR) con gli spigoli arrotondati e che presenta sul lato superiore una fessura per l'inserimento delle cartucce contenenti i giochi e sulla parte frontale 4 connettori per i gamepad, senza comprare periferiche esterne.

Funzione, come dicevamo, mediante cartucce che vengono direttamente inserite nella macchina. Questo sistema consente di immagazzinare meno dati rispetto ai CD-ROM, ma permette tempi di caricamento minimi, possibilità di salvataggio senza ausilio di memory card, presentando un sistema di riconoscimento delle cartucce tramite un circuito elettronico chiamato Checking Integrated Circuit affiancato da un bus di tipo Serial Peripheral Interface.
Nel periodo fu criticata per due cose. Il costo elevato dei giochi su cartuccia e appunto il non utilizzo dei CD ROM. Era un periodo storico dove il CD ROM rappresentava il futuro, costava decisamente poco ed era possibile utilizzare le console dotate di lettore cd rom anche come lettori dvd.
Presenta quattro uscite per i gamepad, in modo tale che si possa giocare con titoli multiplayer per quattro giocatori senza dover comprare supporti aggiuntivi come il Multitap di Sony. Sotto il gamepad è disponibile uno slot che consente di inserire periferiche addizionali. Davanti alla console è presente uno slot per inserire un nuovo banco di memoria RAM. A differenza del suo predecessore (il Super Nintendo), il Nintendo 64 non può fornire in uscita un segnale video RGB, e, nella versione europea, nemmeno S-Video. Questo significa che un N64 europeo ha come miglior segnale video possibile in uscita il video composito.
Sotto alla console è presente uno slot utilizzato per collegare la console con il Nintendo 64DD, espansione hardware uscita solo in Giappone.

La vera innovazione arriva attraverso il joypad, tra i fan chiamato **tricorno** a causa della sua particolare forma con 3 maniglie per afferrarlo, presenta:

Una leva analogica al centro del controller, con base ottagonale, sulla maniglia centrale.
Una croce digitale posta a sinistra del controller.
Sei tasti frontali, di cui due sono chiamati A e B (rispettivamente di colore blu e verde), gli altri quattro sono gialli, al centro dei quali vi è la lettera C e su ognuno di essi vi è una freccia direzionale (nel controller del Nintendo GameCube furono poi sostituiti da una seconda leva analogica, chiamata stick C).
3 tasti dorsali, di cui L ed R posti rispettivamente a sinistra e a destra, e Z è posto dietro la maniglia centrale del controller.
Tasto START al centro del controller.

Espansione posta nella parte alta del retro del controller, ideata per accogliere tre accessori opzionali aggiuntivi:
il Rumble Pak, che permetteva una funzione di vibrazione.
il Controller Pak, una memoria aggiuntiva adoperata da alcuni giochi, che permetteva il salvataggio di dati opzionali.

Fig. 1 - il Nintendo 64 ed il suo controller

il Transfer Pak, un accessorio che permetteva di trasferire alcuni dati di gioco tra il Nintendo 64 e il Nintendo Game Boy, la console portatile Nintendo. Fu poco supportata, dato che permetteva il trasferimento di foto scattate con un accessorio del Game Boy, la Game Boy Camera, e il trasferimento di dati fra i titoli Pokémon Stadium e Pokémon Stadium 2 con i tre titoli di Pokémon del Game Boy presenti all'epoca (Pokémon Rosso, Blu e Giallo).

Può essere usato in due configurazioni: nella prima lo si utilizza con le mani sulle maniglie esterne, trascurando la leva analogica e il tasto Z, nella seconda si predilige la maniglia centrale andando a sfruttare questi due elementi, trascurando la croce digitale e il tasto L. Comunque generalmente sono stati i giochi a "imporre" una delle due configurazioni di gioco. Il gioco The Legend of Zelda: Ocarina of Time introdusse una tecnica di gioco, attualmente usata nei giochi recenti, estremamente innovativa, che permetteva, ad esempio in un combattimento contro un nemico, di "agganciare" il bersaglio in questione, ovvero fare in modo che la telecamera lo inquadri automaticamente in ogni situazione di gioco, a prescindere dai movimenti effettuati dal personaggio protagonista e dal bersaglio stesso; essa è chiamata Z-targeting, proprio in funzione del fatto che quest'"agganciamento" si effettua tramite pressione del tasto Z.

Le altre due grossissime novità di questo controller sono state le introduzioni nel mercato videoludico di massa della leva analogica e la funzione di vibrazione (opzionale); data l'importanza di queste due nuove feature, in seguito anche Sony dotò la PlayStation di una revisione del proprio controller, realizzandone di fatto una seconda versione, il DualShock, che ora disponeva di due leve analogiche e della funzione di vibrazione integrata e autoalimentata.

PERIFERICHE

Durante il ciclo di vita del Nintendo 64 vennero prodotti diversi componenti aggiuntivi ufficiali per la console:
Controller Pak: una memoria da 256 kB che si inseriva sotto il pad, divisa in 123 "pagine". Successivamente Nintendo, accortasi della minima dimensione di questa scheda di memoria decise di crearne altre versioni dalla dimensione da 1 fino a 4 MB.
Expansion Pak: si tratta di un banco di 4 MB di RDRAM (Rambus DRAM) che va a affiancarsi al banco originale da 4 MB montato on-board, questo consente di raggiungere gli 8 MB di RAM, permettendo di utilizzare alte definizioni di gioco e molti altri miglioramenti. Sebbene sia supportato da diversi giochi, alcuni di questi possono funzionare solo

Fig. 2 - Nintendo 64 può contare di un parco titoli molto ampio con diverse esclusive Nintendo

Specifiche Tecniche

Central Processing Unit:
NEC VR4300-64 bit da 4,6 milioni di transistor
@93,75 MHz
24 kB cache di primo livello
Architettura MIPS / R4300i RISC
Capacità matematica totale della CPU: 93,0 milioni di operazioni al secondo
Processo di fabbricazione a 0,35 µm (micrometri)

Graphics Processing Unit:
Coprocessore Silicon Graphics-RPC Reality 64 bit @62,5 MHz
Unità vettoriale per interi a 8-bit
Rendering grafico di picco pari a 150.000 poligoni al secondo con tutte le elaborazioni[8], 600.000 poligoni monocromi al secondo.

Elaborazione:
Z-buffering
Anti-aliasing
Texture mapping
Filtro lineare (bilineare e trilineare)
Mip-mapping
Gouraud shading
Fillrate in pixel da 30 megasamples al secondo con Z-buffer
16.7 milioni di colori (32.768 sullo schermo)

Risoluzioni: 320 × 240 / 640 × 480 pixels
Performance GPU: 600 MegaFlops

Memoria:
4 MByte RDRAM
System Bandwidth @562,5 MB/secondo
Latenza di 640 nanosecondi

Sonoro:
16-bit stereo
100 canali PCM lineare (massimo 16-24 di elevata qualità). Ogni canale occupa per intero un ciclo della CPU
Frequenza di campionamento a 48,0 kHz
Formati supportati: MIDI, MP3, ADPCM e Tracker

Supporti di Memorizzazione:
Cartucce elettroniche da 4 a 64 MByte
Controller con stick analogico, funzione di vibrazione e ingresso per memory card (Controller Pak)

Input/Output:
4 porte per controller
Porta d'espansione per Expansion Pak

se l'Expansion Pak è stato installato sulla console, mentre altri necessitano dell'Expansion Pak per poter attivare alcune funzionalità extra del gioco.
Rumble Pak: è un accessorio che inserito nel gamepad vibra durante la partita in seguito a eventi generati nell'ambiente di gioco. Questo accessorio ormai è considerato uno standard disponibile per tutte le console delle ultime generazioni. Si tratta di fatto di un motorino alimentato da 2 pile ministilo.
Transfer Pak: è un accessorio che inserito nel controller permette di trasferire i dati di gioco tra il Nintendo 64 e il Game Boy, la console portatile Nintendo. Realmente, questo accessorio aveva una sola utilità, ossia quella di poter trasferire sul Nintendo 64 le immagini catturate con la Game Boy Camera ma in seguito fu adoperato nei giochi di Pokémon Stadium per passare dati dal Game Boy al Nintendo 64.

Il Nintendo 64 può contare di un parco titoli molto ampio con diverse esclusive Nintendo che sono state il vero motore della CONSOLE e che ha permesso di non cadere preda dello strapotere SONY (che imperversava in quel periodo).

Le note dolenti che non hanno permesso a Nintendo di surclassare il rivale Sony? Il prezzo non proprio accessibile delle cartucce, la politica di non abbassare mai i prezzi dei titoli più anziani e la mancanza di alcuni titoli di terze parti che approdarono in Sony (Final Fanasy 7 fu una vera Killer Application per Sony, ndR).

Ciò non toglie che furono tantissimi i giochi notevoli per questa console e qui sotto, visto che siamo sotto natale troverete la lista dei 5 fondamentali per questa console.

TITOLI FONDAMENTALI
MARIO 64 - La prima avventura in 3d di Mario e forse una delle avventure più amate dai giocatori. Si contende la palma con Super Mario World di miglior Mario di sempre. Un gioco molto grande e pieno di segreti. Da provare assolutamente.
Mario Kart 64 - Troverete la recensione in questo stesso numero ;).
The Legend of Zelda: Ocarina of Time/Majora Mask - Due JRPG differenti nello stile ma bellissimi! L'esperienza in 3d della saga sul Nintendo è meravigliosa. Forse i picchi più alti di gameplay della serie.
Goldeneye - FPS basato sulla serie 007. Fu rivoluzionario. Il titolo sviluppato da Rare - vendette più di 8 milioni di copie attestandosi come il titolo third party più venduto su Nintendo 64 - ma furono i livelli tridimensionali completamente esplorabili e i molteplici obiettivi da completare liberamente a trasmettere una sensazione di immersività mai provata prima in un videogioco.
Perfect Dark - Altro FPS seguito spirituale di Goldeneye. Prende ciò che di meglio c'è nel gioco Rare, espandendolo e aggiungendo una trama incredibile. Graficamente bellissimo per l'epoca.

Mario 64

The Legend of Zelda: Ocarina of Time


The Legend of Zelda: Majora Mask

Perfect Dark

UNO2IEC HOST

Una semplice interfaccia basata su Arduino UNO per collegare i Commodore 8-bit al vostro PC

di David La Monaca

UNO2IEC HOST è un progetto a basso costo, e dunque con qualche limitazione, nato per simulare la presenza di un drive 1541 collegato ai computer Commodore 8-bit dotati di una porta seriale DIN a 6 poli. Inizialmente il drive a dischetti CBM-1541 era stato progettato per diventare il degno compagno del C64, il computer più venduto nella storia della Commodore. All'inizio era un Floppy Disk Drive "single-sided" (utilizza un solo lato del floppy alla volta) con 170KB di spazio e dischi da 5"¼. Fin qui nulla di molto diverso da quello che c'era sul mercato all'epoca; il problema era che non disponeva di un'interfaccia vera e propria. Era collegato tramite la porta seriale (IEC) del C64, che utilizzava un controller di I/O (l'integrato MOS 6522) abbastanza veloce di suo, anche se presentava qualche bug quando si cercava di aumentare le performance. Ma a peggiorare le cose, visto che il progetto era in ritardo rispetto al lancio del C64, il boss della Commodore Jack Tramiel e il suo reparto marketing decisero che il 1541 doveva essere retro-compatibile con il VIC-20. Ciò costrinse Robert Russell (l'ingegnere progettista) a ridurre la velocità ancor più di quanto previsto. Il suo drive 1541, inizialmente disegnato per essere usato con un MOS 6526 su linea seriale ad alte prestazioni, poteva diventare il più veloce della sua epoca, ma alla fine si rivelò essere fra i più lenti, rispetto ai concorrenti a 8-bit degli anni '80. Più di due minuti per caricare 64KB di dati in memoria, una roba assurda. Ma la storia non finì in questo modo. In realtà, come molti sanno, il CBM-1541 è praticamente un computer completo perché dispone di una CPU 6502, ROM dedicata e di una piccola memoria RAM, in pratica manca solo il chip video! Così, nel tempo, alcuni programmatori hanno usato l'unità a dischi e il suo hardware come co-processore affidandogli alcuni compiti di calcolo o la gestione di turbo-loader per i loro programmi o i loro demo!

Realizzazione hardware

Andiamo subito al sodo. L'idea è quella di collegare una scheda Arduino UNO (o NANO) ad un C64 (poi vedremo che la compatibilità di ogni Commodore a 8-bit che abbia una porta seriale standard è totale) attraverso l'interfaccia IEC per simulare la presenza di un disk drive 1541. Il progetto UNO2IEC a cui faccio riferimento è quello di Lars Wadefalk [R1] che ho scoperto di recente anche se risale al 2013. Diviso in 3 fasi, il progetto richiede l'assemblaggio del cavo, la compilazione di un programma che farà da "disk image server" e la preparazione di una board Arduino come HOST tra il PC e il Commodore. Come detto, lo stesso progetto funziona su C16/Plus4, VIC-20, C64 e C128 e può anche essere usato con un FileBrowser che accetta comandi SD2IEC, mentre la compatibilità con cartucce FastLoader o JiffyDOS non è ancora stata implementata.

Partiamo dall'elenco dei componenti necessari per costruire il cavo:
- Una porzione di cavo a 5 fili
- 1 connettore maschio DIN a 6 poli
- 1 scheda Arduino UNO (o NANO, meno costoso)

Il montaggio è semplice: il connettore IEC ha 3 segnali di cui abbiamo bisogno (CLK, ATN, DATA) e uno opzionale

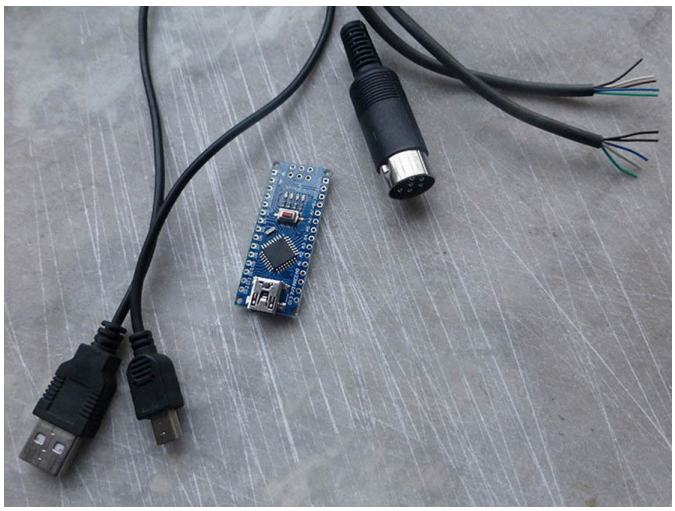


Fig. 1 - I componenti per realizzare il cavo

(RESET). Questi segnali possono essere collegati a qualsiasi pin digitali di Arduino.

Basta annotarli e poi modificarli opportunamente nell'interfaccia del programma server. Nel mio caso sono stati usati i pin digitali 6, 5, 4 e 3, anche per una questione di maggiore linearità e comprensione per l'utente finale.

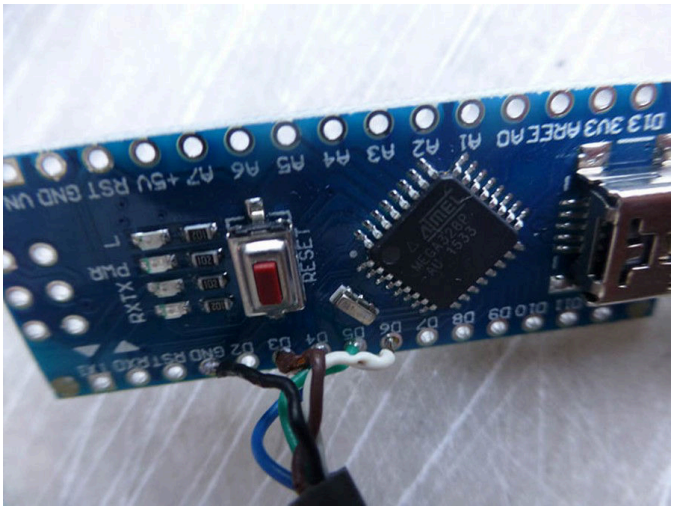


Fig. 2 - I fili saldati dal lato Arduino UNO

Per la connessione dei fili è stato usato questo schema (pin e colore):

IEC	SGN	COLORE	ARDUINO
5	DATA	Marrone	D3
3	ATN	Blu	D4
4	CLK	Verde	D5
6	RESET	Bianco	D6
2	GND	Nero	GND

Connettore femmina DIN 6 poli

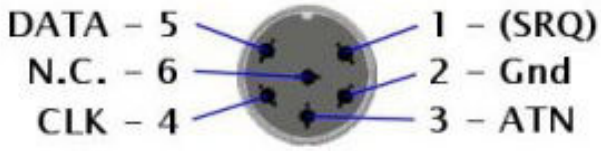


Fig. 3 - Il cavo UNO2IEC finito e pronto all'uso

Per quanto riguarda la programmazione della scheda Arduino occorre avere a disposizione l'ambiente di compilazione di Arduino (disponibile su www.arduino.cc). E' tutto molto semplice, basta seguire l'eccellente guida "Getting Started" del sito [R3].

Si tratta di decomprimere in una cartella del vostro sistema il progetto prelevato da [R2] e prendere in considerazione la cartella con il nome "uno2iec" che contiene i sorgenti necessari per compilare e effettuare l'upload sulla scheda Arduino del firmware necessario.

Aprite la cartella e fate doppio click sul file "uno2iec.ino": il file si aprirà nell'ambiente di compilazione di Arduino. Poi basta premere CTRL+U per dare il via alla compilazione ed al caricamento automatico sulla scheda precedentemente connessa al PC attraverso la porta mini-USB.

Installazione software

Il programma di controllo UNO2IEC HOST è un software open-source [R2] compilabile per le maggiori piattaforme:

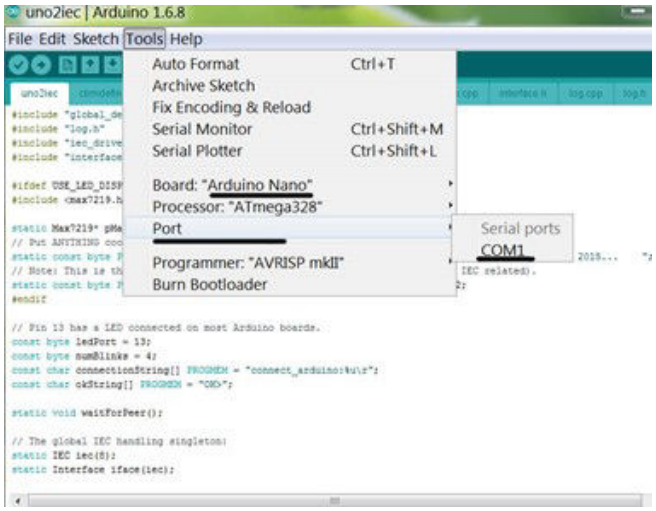


Fig. 4 - Compilazione e upload codice su Arduino NANO

MS Windows (da XP a W10), Mac OS X, Linux e RPI. Il pacchetto per Windows è disponibile già in forma binaria [R4], mentre per gli altri sistemi si possono compilare i sorgenti avendo l'accortezza di includere le librerie grafiche Qt per la gestione dell'interfaccia visuale del programma.

Il software UNO2IEC HOST è lo strumento con il quale interfacciamo il nostro computer a 8-bit Commodore con il PC attraverso l'utilizzo di una porta USB (sono supportate le porte USB 2.0 e 3.0). Il programma consente di selezionare l'immagine disco D64 che vogliamo utilizzare per caricare e salvare programmi, giocare e, in generale, trasferire dati da e per il nostro amato Commodore. Una volta effettuata la selezione dell'immagine disco è possibile caricare applicazioni e programmi presenti all'interno dei file D64/T64 usando i comandi standard (LOAD, LIST, RUN, DLOAD, DIRECTORY, ecc.) previsti dalle diverse implementazioni del BASIC sulle macchine Commodore a 8-bit.
L'installazione vera e propria del programma "server" è quanto di più semplice possa esserci. Per Windows, decompattate il pacchetto zip in una cartella di vostro gradimento e lanciate l'eseguibile "uno2iec.exe". Non c'è altro da fare. Lo stesso vale per gli altri sistemi: una volta effettuata la compilazione con le librerie Qt, per OS X troverete un file "uno2iec.app" e per Linux un eseguibile "uno2iec" nella cartella principale del pacchetto compresso, nessuna procedura d'installazione richiesta).

Configurazione porta COM

Prima di utilizzare il cavo UNO2IEC occorre configurare la porta seriale USB alla quale è fisicamente collegato il cavo. E prima di configurarla occorre... trovarla! Niente di complicato. Basta andare nella configurazione dei dispositivi Windows e scoprire quale porta COM il sistema operativo ha assegnato al cavo di comunicazione col C64. Nello specifico basta aprire il pannello di controllo (Start > Pannello di Controllo) e poi selezionare Gestione Dispositivi > Porte (COM e LPT). A questo punto nella lista si potrà scorgere la porta seriale riservata dal sistema operativo, ad es. USB-SERIAL CM340 (COM5).

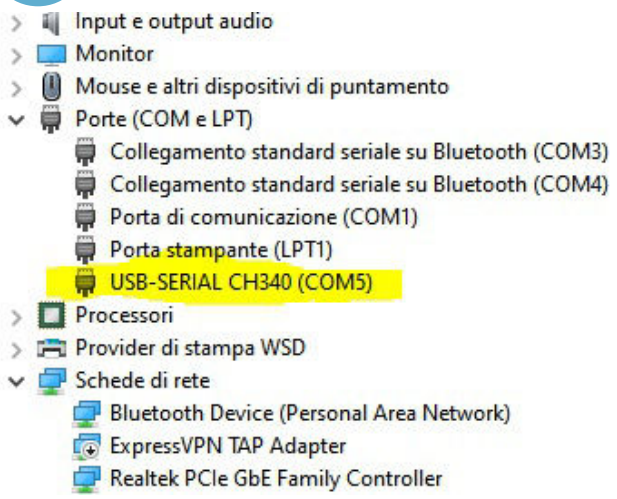


Fig. 5 - Rilevazione porta COM in Windows

Nel vostro caso potrebbe essere assegnata una porta COM differente (COM6, COM7, COM9, ecc.) a seconda della configurazione del PC che state usando. Segnatevi la porta COM che Windows ha selezionato per voi e procedete alla configurazione del software UNO2IEC.

Configurazione software UNO2IEC

OK, lanciate ora il software e impostate i parametri di funzionamento di base. Una volta aperto il programma selezionate l'interfaccia di visualizzazione tra quelle disponibili. Di default il tema è quello del Commodore 64, ma sono disponibili anche quelli del VIC-20, C128 (anche 80 colonne) e del PLUS/4. Dal menu Main selezionate Directory Listing Theme > Machine > Vic-20 | C64 | C128 | C128 (80 col) | Plus4. Questa impostazione non è altro che un modo per settare il tema della finestra di controllo del software quando si utilizzano I file D64. Nella finestra di controllo verranno visualizzati i contenuti del dischetto virtuale di volta in volta "montato", cioè selezionato.

Main > Directory Listing Theme > Machine > C64 (default)

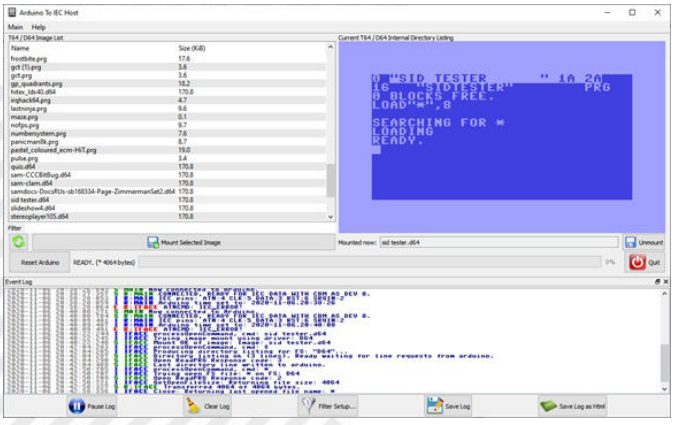


Fig. 6 - Il programma UNO2IEC in azione

Passiamo ora a configurare le impostazioni della porta di comunicazione che abbiamo rilevato dalla Gestione Dispositivi del Pannello di Controllo. Selezioniamo Main > Settings (Ctrl+O). Impostiamo la Porta COM concordemente a quanto trovato (nel nostro caso COM5) e lasciamo per ora la velocità di trasferimento dati (Serial Speed) a 57600 bit/sec.
Nella riga successiva selezioniamo i seguenti parametri:

Device Number: 8 | 9 | 10… [4-30] – corrisponde all'ID del drive utilizzabile
Reset Pin (optional):6 – non supportato da tutte le macchine Commodore
Clock Pin: 5, ATN Pin: 4, Data Pin: 3

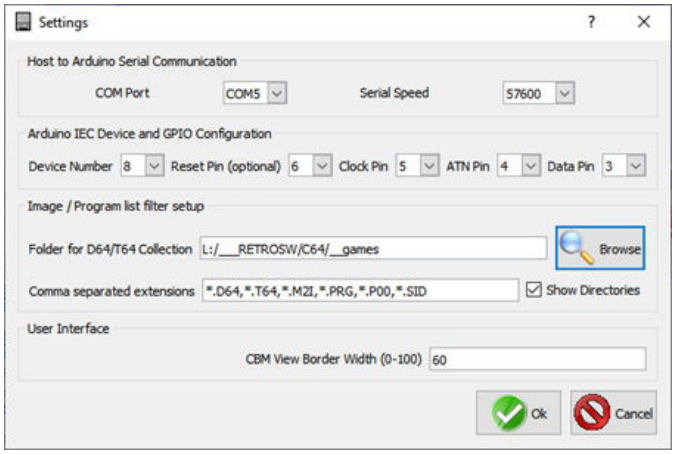


Fig. 7 - Impostazioni di comunicazione con la scheda Arduino

Tutti questi parametri potrebbero già essere a posto, ma in caso modificateli come sopra. In pratica diciamo al software UNO2IEC come sono stati configurati i pin della board Arduino UNO presente sul nostro cavo, per l'appunto in base alla funzione (pin di reset, il clock, il pin di dati, ecc.).

In seguito scegliamo la cartella che sul nostro sistema contiene i file D64, T64, PRG, ecc. In pratica il programma deve sapere dove si trova la vostra collezione di programmi e giochi per il vostro sistema. Ad es. Folder for D64/T64 Collection: C:\TEMP\RETROGAMES\C64

Utilizzo ordinario

Una volta confermato tutto con il pulsante OK, la parte sinistra della finestra principale del software UNO2IEC mostrerà l'elenco dei file (dischetti virtuali D64, tape T64, programmi singoli PRG, ecc.).

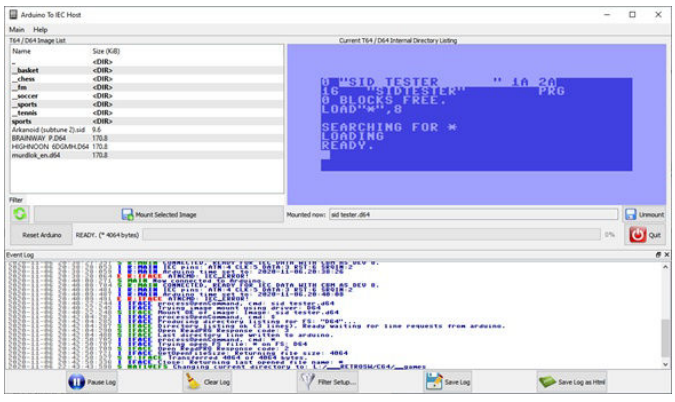


Fig. 8 - Selezione e mount di un file immagine

Per effettuare il mount di un file D64 o T64 basta selezionare il file con un doppio click oppure premere il pulsante Mount Selected Image. Nella finestra di controllo in alto a destra comparirà la directory del file immagine selezionato. A questo punto si può passare sulla tastiera del computer e digitare i comandi di caricamento dei programmi, esattamente come fareste con un drive 1541 collegato alla macchina. Quindi, ad esempio, per caricare la directory del dischetto virtuale digitare: LOAD"$",8 e premere Return. Per visualizzare la lista dei programmi basta un semplice LIST seguito da Return. Ovviamente l'ID del drive dipende da cosa avete selezionato nelle impostazioni del programma UNO2IEC. Per caricare un programma basta invece usare il comando LOAD"*",8 o LOAD"*",8,1 seguito da Return. Naturalmente vale la sintassi dei comandi per il drive 1541 esattamente come se aveste un drive fisico connesso alla porta seriale del vostro Commodore. Nel caso usiate computer che prevedono comandi avanzati per la gestione del drive seriale (come accade per C16, Plus/4 e C128), questi sono pienamente supportati.

Anche nel caso di mount di file T64, è sufficiente dare il comando LOAD"*",8 per caricare il programma contenuto nell'immagine del nastro. Se invece avete dei file PRG già pronti, occorre selezionare la cartella Windows che li contiene e passare poi alla tastiera del computer a 8-bit ed effettuarne il caricamento con LOAD"nomeprogramma",8. Ricordate che usare il carattere asterisco significa caricare l'ultimo file che il computer aveva caricato in precedenza. Ovviamente, nel caso in cui il computer sia stato appena acceso o resettato, con LOAD"*",8 si caricherà il primo programma presente nella directory del dischetto montato. Per la cronaca, il comando corretto per essere sicuri di caricare il primo programma dell'immagine disco è LOAD":*",8 (aggiungendo cioè il carattere dei due punti prima dell'asterisco).

Naturalmente la velocità di caricamento o di salvataggio è del tutto simile a quella di un drive fisico collegato alla seriale o a quella di altre interfacce SD2IEC, non possiamo aspettarci nulla di differente. Durante le fasi di caricamento o salvataggio dati, una comoda barretta di controllo ci aiuterà a capire a che punto siamo, ottenendo nel contempo anche un'idea del transfer rate corrente. La velocità di trasferimento può essere aumentata agendo sul parametro Serial Speed presente alla voce Main > Settings, ma è difficile spuntare una velocità maggiore anche a causa dei soli 2KB di RAM presenti sulla scheda Arduino (dei quali 512 byte sono usati come data buffer per caricare i settori dei dischi). Anche caricando la versione PRG del JiffyDOS prima di effettuare i necessari caricamenti oppure inserendo sul nostro Commodore una cartuccia FastLoader non si ottengono risultati migliori nel transfer rate, sebbene lo sviluppatore del progetto [R1] e altri utenti stiano tuttora effettuando dei test per migliorare sul fronte della velocità di trasferimento dati.

Una funzione utile del programma è quella di poter effettuare un reset della scheda Arduino, in caso di freezing dell'applicazione o della PCB. Se stiamo usando un C64 in cui la porta seriale supporta il pin di reset (i primi modelli avevano questa piedinatura, il C64c invece no), allora anche il nostro C64 verrà resettato: utile quando si vuole passare da un gioco all'altro o da un'immagine D64 all'altra.

Conclusioni

Prima dell'avvento della tecnologia USB le interfacce di collegamento tra computer Commodore e PC con porte seriali e parallele rappresentavano una soluzione comoda ed economica (ad es. 64HDD, cavi XA/XE, XUM 1541). Al giorno d'oggi l'interfaccia UNO2IEC non è probabilmente il modo più semplice ed immediato per utilizzare una memoria di massa "moderna" sui vecchi computer Commodore, ma di certo è fra le più economiche se non la più economica in assoluto. Il materiale per costruirne uno è davvero basilare, di facile reperibilità e poco costoso. La costruzione del cavetto e della stampa 3D di un box per proteggere la PCB può andare dagli 8 ai 12 euro, case incluso. Naturalmente le prestazioni di base sono più o meno quelle di un drive 1541 originale o di una delle tante interfacce SD2IEC in commercio, ma, senza scomodare il confronto con le interfacce più potenti ed elaborate come la Ultimate II+ o la Pi1541 (che garantiscono un livello di compatibilità con il drive originale intorno al 99%), per chi inizia o per chi non vuole spendere soldi su un drive originale e impelagarsi fra dischetti, pulizia e disallineamento delle testine, UNO2IEC HOST rappresenta senza dubbio una soluzione più che efficace.

Un ringraziamento particolare (e d'obbligo) va a **Carlo Piacentin**i, noto nel mondo del retrocomputing italiano, con il quale ho collaborato attivamente per questo piccolo progetto e che ha realizzato per me (e per se stesso) alcuni esemplari di cavi UNO2IEC di ottima fattura, trovando anche il tempo di realizzare una comoda custodia protettiva stampata in 3D. Ringrazio anche **Emmanuel Barraud** (aka **Klyde**) per il suo supporto e per aver anche lui costruito

Fig. 9 - Il cavo UNO2IEC con il suo case protettivo

alcune interfacce UNO2IEC HOST per sé e per alcuni amici retro-commodoristi.

Compilare il software Uno2IEC Host

La parte software della soluzione è sviluppata in C con interfaccia grafica QT ed è disponibile, oltre che per Windows, OS X e Linux, una versione per RaspberryPI. La compilazione non è complicata su nessuna piattaforma, tranne forse per Windows per la quale occorre scaricare una tonnellata di programmi tra librerie grafiche, compilatore e IDE adeguati. Per rendere la vita ancora più facile a tutti gli interessati abbiamo messo a disposizione il file eseguibile già pronto con la sua libreria QT statica per Windows. Per i sistemi Mac OS X, Linux e RPI, invece, potete seguire queste guide rapide.

Mac OS X

Per chi non lo sapesse, il kernel OS X è basato su BSD, più precisamente su di un kernel chiamato Darwin, che è libero e anche periodicamente aggiornato e modificato da un'attiva comunità. Poiché si tratta di un BSD quasi standard, un gruppo di programmatori ha portato/creato il gestore di pacchetti BSD per OS X; il progetto è chiamato Brew e si usa come Apt/Yum sulle distribuzioni BSD e Linux/Debian o Linux/CentOS. Useremo Brew per installare le librerie QT5 e poi procedere con la compilazione del progetto. Installare Brew e Qt5 è relativamente semplice ma è anche richiesto XCode da prelevare e installare dall'AppStore.
Dopo aver preparato tutto correttamente, scaricate il pacchetto "uno2iec-0.5.0.0.zip" da [R2], decomprimete il contenuto di questo file e verrà creata una cartella "uno2iec_src". Aprite un terminale, accedete alla cartella creata e digitare i comandi seguenti:

```
$ brew install qt5
$ qmake "CONFIG+=release staticlib".
$ make
```

Dopo diversi minuti e se tutto va bene, verrà creata una nuova cartella chiamata "release". All'interno di questa cartella ci sarà l'eseguibile "uno2iec.app", basta copiarlo nella cartella Applicazioni di OS X ed eseguirlo.
La compilazione sotto OS X è abbastanza lineare ma spesso dipende fortemente dal sistema in uso. Ho quindi accluso una versione binaria dell'app già compilata sotto VM Mojave [R5]. Scaricate il file "uno2iec_osx.zip" e apritelo, posizionando da qualche parte la cartella "uno2iec_osx". Un doppio click sul file uno2iec_host,app sarà sufficiente per far partire il programma. Dal menu principale basta poi scegliere "RP2IEC > Preferences" per impostare i parametri di funzionamento dell'interfaccia. Se il programma non dovesse partire, assicuratevi di aver installato la libreria Qt5 e aggiungete il comando:

```
$ ln -s /usr/local/opt/qt5 /usr/local/opt/qt55
```

Questo comando serve a superare un problema di percorso nella memorizzazione nel sistema della libreria grafica Qt5, essenziale per il corretto funzionamento del programma.

Linux / Raspberry PI

Per Linux le cose si fanno ancora più semplici! Usando una delle distribuzioni basate su Debian che offrono "apt" come gestore di pacchetti (potete usare qualsiasi distro, purché sappiate cosa fate), dobbiamo soltanto installare QT5 e le librerie di sviluppo e procedere con la compilazione. Nel mio caso ho fatto uso di una distribuzione LUbuntu 20.04, per la quale è stato necessario forzare l'installazione del modulo QtSerialPort. Anche il comando "qmake" era assente, ma è bastato dare un "sudo apt install qtchooser" per rimediare.
Scaricate il pacchetto "uno2iec_src.zip" dal sito [R2] – in particolare si tratta della versione 0.5.0.0 – e decomprimete il contenuto di questo file in una directory della vostra home directory. Verrà creata una cartella "uno2iec_src". Aprite ora un terminale, accedete alla cartella appena creata e digitate i seguenti comandi:

```
$ sudo apt install qt5-default
$ sudo apt install libqtSerialport5 libqtSerialport5-dev
$ qmake "CONFIG+=release staticlib".
$ make
```

Come per OSX, verrà creata una nuova cartella chiamata "release". All'interno di questa cartella ci sarà l'eseguibile "rpi2iec". Basta rinominarlo e copiarlo dove si vuole sul proprio Linux (es. in /usr/bin) e si avrà il programma server sempre pronto all'uso.
Del tutto analogo a quanto visto sopra il processo di compilazione per Raspberry PI (tutte le versioni) sul quale si raccomanda di utilizzare la distribuzione Raspbian.

Riferimenti

R1 – L'autore del progetto: https://github.com/Larswad
R2 – Il pacchetto dei sorgenti del progetto: https://github.com/Larswad/uno2iec/releases
R3 – Il compilatore per Arduino: https://www.arduino.cc/en/Main/Software
R4 – Il programma UNO2IEC HOST per Windows: https://www.retromagazine.net/download/uno2iec/uno2iec_host.zip
R5 – Il programma UNO2IEC HOST per Mac OS X: https://www.retromagazine.net/download/uno2iec/uno2iec_osx.zip

Introduzione al **MEGA65**

di Gianluca Girelli

Come questa rivista testimonia molto bene, i nostri amati retro-computer stanno vivendo una seconda (o forse terza) giovinezza. Uno dei motivi è probabilmente che nessuno di essi utilizzava parti mobili che che avrebbero potuto subire un guasto meccanico, quindi sono riusciti a resistere più o meno indenni al passare del tempo. Tuttavia, tutte le loro periferiche di archiviazione ne avevano, così negli ultimi anni molti produttori hanno esplorato modi per fare di nuovo un buon uso dei nostri vecchi compagni per mezzo di periferiche di archiviazione e navigazione di rete all'avanguardia.

La cosa buona di tali dispositivi (come la cartuccia Ultimate II, solo per citarne uno) è che ti permettono di mettere le mani sui vecchi computer, ma i loro condensatori ed altri componenti stanno comunque invecchiando e, un giorno, si romperanno. Grazie a molti "visionari" però, quali i ragazzi del MEGA (Museum of Electronic Games and Art), è stato avviato un nuovissimo progetto a 8 bit: costruire un computer basato sul C65, funzionante circa 50 volte più velocemente di un C64 pur rimanendo altamente compatibile. Design del C65, tastiera meccanica, uscita HD, supporto per scheda SD, Ethernet, memoria estesa e altre funzionalità aumenteranno a breve il nostro divertimento senza rovinare la sensazione a 8 bit.

Il MEGA65 è un progetto completamente open-source e, a differenza di altri computer basati su FPGA, si può esaminarne, modificarne e migliorarne l'implementazione. Questa implementazione FPGA si basa su una riproduzione della compatibilità su vasta scala del vecchio microchip MOS 6502, incluso l'OpCode illegale. Ciò significa che MEGA65 è progettato per offrire compatibilità sia con C64 che con C65 e, poiché è open-source, la sua compatibilità continuerà a migliorare nel tempo.

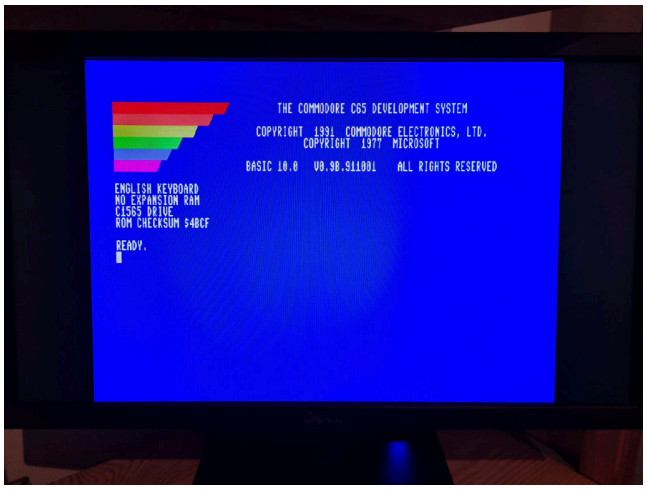
A differenza di altri prodotti là fuori, questo non ha lo scopo di farci fluttuare nella nostalgia: sebbene questo sia un risultato molto probabile e molto desiderabile, si può davvero tornare alla programmazione come ai vecchi tempi grazie a un'ampia libreria di linguaggi di programmazione facili da imparare, programmi di disegno e strumenti vari incentrati sul supporto alla nostra creatività.

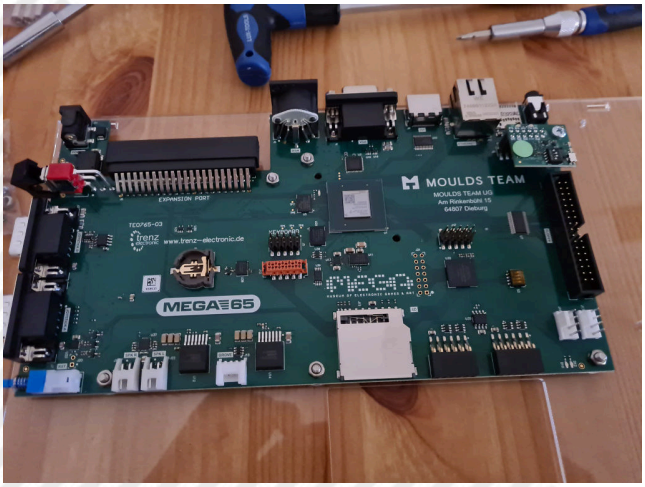
Il progetto è stato annunciato nel 2015 e avrebbe dovuto concludersi nel giro di pochi mesi. Ad oggi, una data di rilascio non è stata ancora annunciata, ma gli sviluppatori stanno davvero cercando di fare del loro meglio ed i risultati sono ora tangibili. Inoltre, è in fase di sviluppo una versione portatile del MEGA65 e le sue caratteristiche sono semplicemente uniche: un telefono cellulare doppio 4G (e aggiornabile 5G) con una durata della batteria senza precedenti (circa 1000 ore in standby), altoparlanti integrati stereo da 2W e tempo di avvio inferiore di 1 secondo.

Come ogni progetto di questo tipo, che si basa fortemente sul supporto diretto dei fan, anche questo ha avuto un momento di stop, proprio quando è stata lanciata la campagna di rilancio per l'acquisto dell'attrezzatura di stampa ad iniezione per costruire le scocche per il MEGA65. Una macchina del genere è molto costosa e anche se la comunità ha donato oltre 20K €, l'obiettivo di 66K sembrava quasi irraggiungibile, fino a... beh, fino al giorno in cui oltre 40K € sono stati raccolti in un fine settimana!

Dopo il successo della campagna di raccolta fondi, i kit di sviluppo sono diventati realtà nel giro di pochi mesi e ora sono stati inviati agli sviluppatori. Per gentile concessione di Stefan Vogt, creatore di avventure testuali multipiattaforma all'avanguardia, ora abbiamo la possibilità di dare uno sguardo ravvicinato a questa bellezza.

Le immagini di questo articolo mostrano la piastra inferiore e il drive, la scheda madre e la sua scocca, con il drive montato, insieme ai dettagli della scheda e della tastiera. Il devkit assemblato sembra davvero magnifico, e nonostante alcune persone pensino ancora che tutto questo sia solo un giocattolo costoso per geek e nerd (cosa di cui peraltro vado fiero) lasciatemi sottolineare che Tristam Island, l'ultima avventura testuale di Hugo Labrande uscita solo poche settimane fa, gira già su un MEGA65 e così sarà per Hibernated2 del già citato Stefan Vogt. Non posso chiudere questo articolo senza ringraziare Paul Gardner-Stephen e tutti gli altri di MEGA65.org per aver reso possibile tutto questo.





Un **codice operativo illegale**, chiamato anche **istruzione non documentata**, è un'istruzione per una CPU che non è menzionata in alcuna documentazione ufficiale rilasciata dal progettista o dal produttore della CPU, che tuttavia ha un effetto. I codici operativi illegali erano comuni sulle vecchie CPU progettate negli anni '70, come la tecnologia MOS 6502, Intel 8086 e Zilog Z80. Su questi vecchi processori, molti esistono come effetto collaterale del cablaggio dei transistor nella CPU e di solito combinano funzioni della CPU che non erano destinate ad essere combinate. Sui processori vecchi e moderni, ci sono anche istruzioni incluse intenzionalmente nel processore dal produttore, ma che non sono documentate in nessuna specifica ufficiale.

https://en.wikipedia.org/wiki/Illegal_opcode

MOS VIC

di Leonardo Miliani

In questo articolo andremo a scoprire il MOS VIC, il processore preposto alla gestione dell'immagine dei computer ad 8 bit Commodore VIC-20. Il VIC, acronimo che sta per Video Interface Chip, è un chip così importante per il VIC-20 da avergli dato anche il nome ma, per contro, le sue origini non sono comuni ad esso: difatti, il VIC è un chip nato prima del computer stesso.

Origini

Torniamo indietro al 1976. MOS Technology è un piccolo produttore di integrati fondato alla fine degli anni '60 del XX secolo da Allen-Bradley come fornitore secondario dei chip di Texas Instruments. Arriva a conoscere una certa fama nel 1975 quando, grazie al lavoro di diversi ex-progettisti del Motorola 6800, tra cui Chuck Peddle, che hanno lasciato l'azienda e sono stati assunti in blocco proprio da MOS Technology, mette in commercio il 6502, una CPU economica che riscuote subito un enorme successo commerciale. Motorola, però, denuncia MOS perché lo sviluppo del 6502 è a suo dire stato portato avanti con conoscenze e tecnologie "esportate" da Motorola dai suoi ex-dipendenti. Allen-Bradley, vista la malaparata (una causa con Motorola potrebbe rivelarsi molto costosa) e visto anche il calo che il fatturato del settore dei chip per calcolatrici sta registrando in quel periodo, vende le sue quote ai dirigenti dell'azienda e abbandona MOS Technology. Agli inizi del 1976 MOS patteggia e paga un'una-tantum a Motorola per chiudere la causa nonché si accorda per prendere le licenze dei chip periferici necessari al funzionamento del 6502.

Nonostante le buone vendite del 6502, le finanze di MOS non versano in buone acque in quel periodo. Alla fine del 1976, perciò, MOS accetta la proposta di acquisto fatta da Commodore, che in quel momento sta combattendo una guerra dei prezzi con gli altri produttori di calcolatrici elettroniche e cerca un produttore di chip per rendersi indipendente dal suo fornitore/rivale commerciale Texas Instruments.

Commodore cerca comunque di espandere il proprio mercato diversificando la sua offerta e, in quest'ottica, viene convinta da Peddle che le calcolatrici hanno ormai fatto il loro tempoo e che i computer domestici saranno il futuro. Peddle propone un sistema basato sul suo kit di sviluppo del 6502 chiamato KIM-1 che MOS offriva ai clienti del 6502: il progetto viene avviato ed il risultato finale è il PET, che Commodore mette in commercio nel 1977. Questo computer è simile ad un terminale, non offrendo la possibilità di gestire grafica raster ma avendo solo un set di caratteri semi-grafici con cui si possono disegnare le interfacce dei programmi.

In MOS Technology continuano le ricerche. Al Charpentier ha nel frattempo realizzato un chip grafico denominato Video Interface Chip, sigla 6560, destinato ad equipaggiare sistemi CRT a basso prezzo quali terminali di computer, sistemi biomedicali, o console da gioco domestiche, un altro settore dell'elettronica di consumo che in quel periodo sta prendendo campo. Il chip, però, non ha successo: MOS non riusce a piazzarlo presso nessuno dei suoi clienti. Contemporaneamente si cerca di migliorare il PET, Chuck Peddle e Bill Seiler hanno progettato un nuovo sistema denominato TOI (The Other Intellect), un computer per l'ufficio con un chip grafico siglato 6564 capace di visualizzare 80 colonne.

Il TOI resta a livello di prototipo perché il 6564 richiede le veloci ma molto costose (per l'epoca) RAM statiche (SRAM) per via dei suoi ristretti tempi di accesso alla memoria video; inoltre, le 80 colonne richiedono uno schermo speciale per essere visualizzate. Entrambi questi

fattori ne farebbero lievitare enormemente il prezzo di vendita e viene deciso di non portare avanti il suo sviluppo. Parallelamente è in fase di sviluppo anche un nuovo modello di PET capace di generare immagini a colori grazie ad un altro chip grafico siglato 6562 che, però, ha gli stessi problemi di natura economica perché anch'esso necessita delle memorie SRAM. Entrambi i progetti vengono accantonati.

Verso la fine degli anni '70 viene assunto un giovane ingegnere, Robert Yannes, che realizza a casa propria un prototipo di computer economico che chiama MicroPET. Insieme ad Al Charpentier e Charles Winterble, Yannes presenta questo prototipo a Jack Tramiel, capo di Commodore, che autorizza immediatamente il suo sviluppo. Il progetto "Vixen", che porterà al VIC-20, prende corpo. Per la grafica ed il suono viene deciso di usare il meglio di quanto fino ad allora sviluppato: viene ripreso l'originale 6560 a cui vengono aggiunti il più performante generatore sonoro del 6562 e la gestione del colore del 6564. Il nuovo VIC è nato.

Caratteristiche tecniche

Il VIC è un chip usato sia per la generazione dell'immagine che del suono. Per la sua programmazione usa 16 registri mappati in memoria, quindi visti dalla CPU e dal computer come normali celle di memoria, agli indirizzi $9000-$900F.

Il VIC possiede 14 linee di indirizzamento e può perciò accedere fino ad un massimo di 16 KB di memoria video, che divide in 3 aree principali: memoria video, memoria colore e memoria dei caratteri. Quest'ultima, nel caso dei VIC-20, è su ROM perché la mappa dei caratteri è predefinita ed è contenuta in 4 KB dove sono salvate quattro distinte mappe: 2 mappe normali, una con caratteri maiuscoli e semi-grafici e una con caratteri misti maiuscoli/minuscoli e semi-grafici, nonché le versioni invertite (ossia in "reverse") di entrambe.

A livello di schermo il VIC gestisce un'immagine le cui dimensioni possono essere impostate tramite i suoi registri fino ad un massimo di 248x232 pixel nei sistemi NTSC e 256x280 pixel in quelli PAL anche se il VIC-20 preimposta l'immagine a 176x184 pixel, corrispondenti rispettivamente a 22 caratteri in larghezza e 23 caratteri in altezza, per un totale di 506 celle di 8x8 pixel.

Questa ridotta risoluzione orizzontale, a causa del rapporto del segnale video di 4:3, si traduce in pixel dall'aspetto leggermente rettangolare, essendo più grandi in larghezza che in altezza (come si evince dalla figura 1). Il chip non permette di gestire grafica di tipo bitmap: ciò vuol dire che non è in grado di indirizzare i singoli pixel dell'immagine ma solo di usare i caratteri come unità grafica.

Fig. 1 - Le immagini generate dal VIC appaiono con pixel rettangolari

Per creare giochi con grafiche elaborate il VIC permette però di usare mappe personalizzate dando così al programmatore la possibilità di creare elementi grafici propri.

La memoria video corrisponde a tanti byte quanti sono quelli impostati per l'immagine corrente: nel caso del VIC-20, che riserva 512 byte per la memoria video, sono usati 506 byte per via della risoluzione dello schermo di 22x23 caratteri. Ogni byte contiene il codice del carattere da visualizzare, la cui matrice dei pixel viene recuperata dalla memoria caratteri, andando a "pescare" nella mappa attualmente attiva, mappa che può essere sia in ROM (per i caratteri predefiniti) oppure su RAM (nel caso si stia usando una mappa ridefinita).

Il colore dei singoli pixel è contenuto nella memoria colore ed è memorizzato in forma di nibble (4 bit), dove ogni nibble indica il codice del colore da usare. I colori sono in tutto 16 ma il chip ha alcune limitazioni. I colori principali dei caratteri e del bordo dell'immagine possono essere solo 8, scelti fra i seguenti: nero, bianco, rosso, ciano, viola, verde, blu e giallo. Per lo sfondo ci sono anche altri 8 colori detti ausiliari fra cui poter scegliere: arancio, arancio chiaro, rosa, ciano chiaro, viola chiaro, verde chiaro, blu chiaro e giallo chiaro.

Il VIC può operare in due modalità, con caratteri ad "alta risoluzione" oppure in "multicolore". La prima è la modalità standard: i caratteri sono larghi 8x8 pixel ed ogni bit in memoria corrisponde ad 1 pixel sullo schermo. La seconda gestisce caratteri con larghezza doppia di 16x8 bit dove 2 bit in memoria corrispondono ad 1 pixel sullo schermo e si attiva impostando il 4° bit del nibble del colore. In questa modalità si possono usare fino a 4 colori differenti, ed ogni colore è selezionato dal valore della coppia di bit.

Questa coppia non indica direttamente il colore ma da dove recuperarlo, fra: il colore primario, il colore del bordo, il colore dello sfondo, il colore ausiliario. Da notare che il chip ha 4 pin destinati esclusivamente al collegamento

diretto con la memoria colore per il recupero diretto del nibble del colore. Siccome la grafica multicolore appare con una risoluzione dimezzata per via del fatto che servono 2 bit per rappresentare un pixel (vedi figura 2) questa modalità non è stata molto utilizzata nei giochi.

Mappatura della memoria

Nonostante il chip video veda la memoria a lui dedicata come un unico blocco contiguo da 16 KB, nel computer in realtà essa viene indirizzata in maniera un po' particolare. La memoria dedicata alla mappa dei caratteri, che occupa 4 KB, è posta nell'area da $8000 a $8FFF. Siccome i caratteri gestibili dal VIC sono 128 e siccome ogni carattere è largo 8x8 bit, sono necessari 4 aree da 1.024 byte ciascuna per contenere i dati delle mappe integrate nel VIC-20. Come detto in precedenza, da $9000 a $900F sono mappati i 16 registri che gestiscono il VIC in ogni suo aspetto: impostazioni dei colori, risoluzione verticale e orizzontale, generazione dei suoni, ecc…

La memoria video è mobile. In un VIC-20 senza espansione RAM installata, essa occupa le celle da $1E00 a $1FFF (512 byte). Se è invece presente un'espansione allora la memoria video viene spostata nel blocco $1000-$11FF: questo perché la memoria dedicata ai programmi deve essere sempre contenuta in un blocco contiguo di RAM.

Anche la memoria colore viene posizionata in uno spazio di indirizzi che varia a seconda se il computer monta un'espansione o meno: nel primo caso viene posizionata nel blocco $9400-$95FF, mentre nel secondo nel blocco $9600-$97FF.

Pregi e difetti

Come detto in apertura, il VIC non gestisce solo l'immagine video. Per contenere il prezzo finale del computer i suoi progettisti hanno inserito al suo interno diverse funzionalità. Il chip è responsabile anche della generazione del suono: esso è dotato di 3 canali audio ad onda quadra e di un quarto canale per il rumore bianco. Il controllo del volume è, però, possibile solo a livello globale.

Il VIC integra due convertitori digitale/analogico grazie ai quali può leggere la posizione degli assi X e Y di un paddle. Può gestire anche una penna ottica. Presente anche l'accesso diretto alla memoria (DMA), per leggere e scrivere nella RAM in maniera indipendente.

Oltre che della grafica bitmap il chip difetta anche della circuiteria di refresh delle DRAM (RAM dinamiche), dato che è stato progettato per lavorare con le SRAM. Le DRAM sono un tipo di memoria molto in uso tra la fine degli anni '70 e la prima parte degli anni '80 perché più economico rispetto alle più veloci ma più costose SRAM: queste ultime,

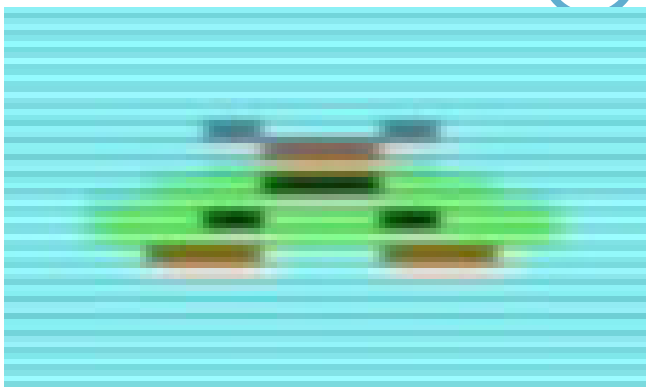


Fig. 2 - Eesempio di grafica multicolore dove due caratteri sono affiancati per creare un oggetto. Notare i pixel "a blocchi" con risoluzione orizzontale dimezzata.

però, a differenza delle DRAM, non necessitano di continui accessi ad intervalli regolari (il cosiddetto "refresh") per non perdere i dati memorizzati.

Il VIC non supporta gli sprite, presenti su altri chip grafici contemporanei come l'Atari TIA (Atari 2600) ed il TMS9918A (quest'ultimo è stato oggetto di un nostro precedente articolo), né un interrupt raster. Possiede comunque un registro che contiene la riga correntemente disegnata dal pennello video.

Eredità

Nonostante i suoi limiti, il VIC-20 riscosse un ottimo successo diventando il primo computer ad essere venduto in più di 1.000.000 di unità e superando il traguardo dei 2,5 milioni di esemplari commercializzati.

E parte del merito va soprattutto al suo VIC, grazie al quale il computer offriva ad un prezzo più che abbordabile capacità grafiche e sonore di un certo rilievo.

Il lavoro svolto sul VIC fu poi fatto fruttare degnamente. Da esso derivarono i due chip responsabili del successo del più venduto computer di sempre, il C64: il VIC-II ed il SID sono infatti diretti "discendenti" di quel primo, vincente, progetto.

Buone feste a tutti e al prossimo articolo.

Notizie Flash!

BASIC ON THE ZX SPECTRUM

(Alberto Apostolo)

Anja de Weerd, lettrice di RMW ENG, ha inserito un commento (Fig.1) all'articolo sulla conversione dei programmi da ZX81 a Spectrum, pubblicato su RMW 26 ITA / RMW 4 ENG, riguardante un piccolo gioco per ZX Spectrum (RMW 26 ITA Pag.7 Fig.7, RMW 4 ENG Pag.9 Fig.7).
Ringraziando per i complimenti, ho accolto le sue indicazioni che ho riportato nel programma in Fig.3.
Alla riga 75, il numero di "scroll" eseguiti è uguale al numero di apici (SymbolShift-7) meno 1.

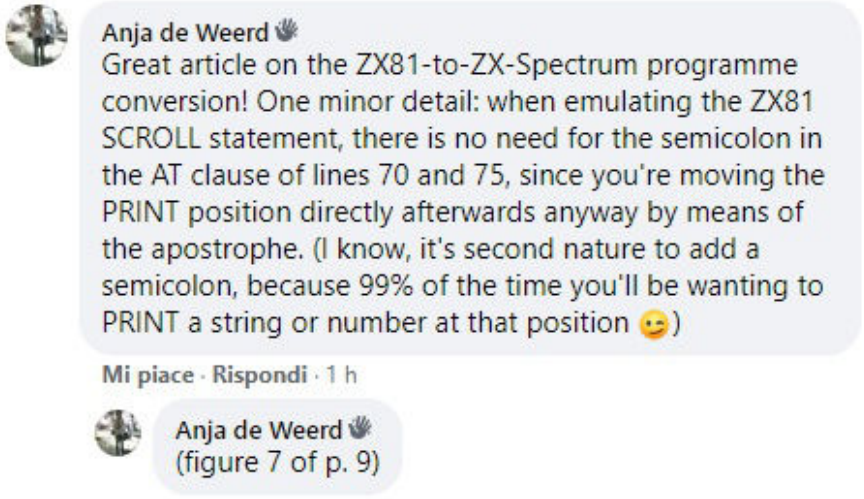


Fig. 1

N.d.R.: chi vuole saperne di più, può iscriversi al gruppo Facebook BASIC ON THE ZX SPECTRUM (Fig. 2) dove il commento è apparso.

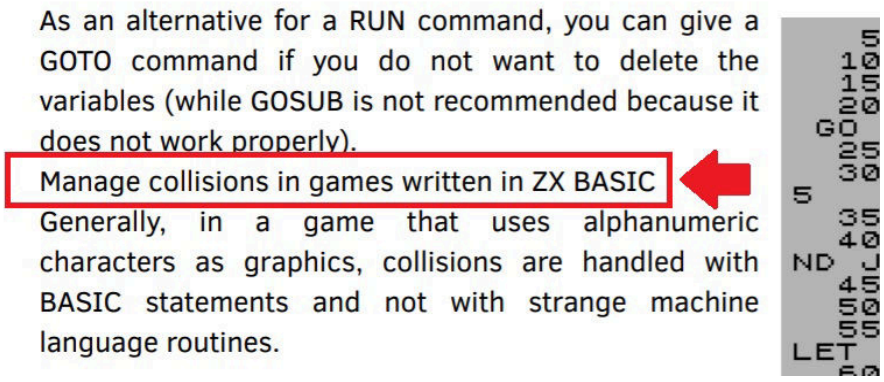
Fig. 2

```
  10 LET S=0: LET I=15
  15 PRINT AT 0,I;
  20 IF SCREEN$ (0,1+I)="W" THEN
 GO TO 90
  25 PRINT "■■■■"
  30 IF INKEY$<>"x" THEN GO TO 5
5
  35 FOR J=-8 TO 8
  40 PRINT AT 8-ABS J,I+1;("T" A
ND J<0)+(" " AND J>=0)
  45 NEXT J
  50 LET S=S-10
  55 LET I=I-(INKEY$="n" AND I>0
)+(INKEY$="m" AND I<29)
  60 PAUSE 5
  65 PRINT AT 14,RND*31;"W"
  70 POKE 23692,255
  75 PRINT AT 21,31;;;
  80 LET S=S+1: GO TO 15
  90 PRINT S
```

Fig. 3

ORRORI DI STAMPA

In RMW 4 ENG (pag. 9), uscito dopo Halloween, ci siamo lasciati sfuggire un "orrore" di stampa.
Il titolo di un paragrafo doveva essere evidenziato in grassetto (Fig.4). Ci scusiamo con i lettori.


As an alternative for a RUN command, you can give a GOTO command if you do not want to delete the variables (while GOSUB is not recommended because it does not work properly).

Manage collisions in games written in ZX BASIC

Generally, in a game that uses alphanumeric characters as graphics, collisions are handled with BASIC statements and not with strange machine language routines.

Fig. 4

UN "TRUCCO" PER ONE-LINERS SU ZX SPECTRUM

(Alberto Apostolo)

Se si digita una riga di programma troppo lunga, con tante istruzioni separate da ":", lo ZX Spectrum si "ribella" rallentando sempre di più la velocità del cursore.
Allora conviene partire dal fondo, inserendo le istruzioni a ritroso (il cursore trovandosi sempre all'inizio della riga, non sarà rallentato). Per non confondervi, preparate prima un elenco delle istruzioni da digitare sul vostro amato ZX.

Un clone di Snake per il Commodore64 su cartuccia

di Giovan Battista "giomba" Rolandi

Introduzione

Come molti dei lettori di queste pagine, anche io mi diletto a "collezionare" quelli che sono oggi chiamati "retrocomputer". Il motivo principale per cui lo faccio è per sperimentare il contatto puro tra hardware e software: mi piace infatti addentrarmi nei meandri di registri e locazioni di memoria, per poi vedere la macchina che obbedisce ad ogni singolo bit di programma che gli impartisco.

È in questo modo che, qualche anno fa, durante un periodo relativamente tranquillo, iniziai a programmare un piccolo gioco per il Commodore 64, un clone del recente e famoso Snake, reso celebre dai primi telefonini con schermo a matrice. Povero di fantasia, decisi di chiamarlo snake6502, in onore del celebre processore che si trova nel cuore di decine di retrocomputer.

Dopo averlo provato a lungo sull'emulatore VICE, e sulla macchina fisica su cassetta, in formato .prg, il naturale passo successivo è stato quello di portarlo su cartuccia, e per farlo ho dovuto affrontare delle nuove sfide che ho deciso di condividere coi lettori di questa rivista.

Spazio

Il Commodore64 è predisposto per utilizzare cartucce "standard" da 8KiB, anche se è possibile utilizzarne fino a 16KiB e, con l'aggiunta di hardware dedicato, anche oltre. Da parte mia, ho sempre cercato di essere parsimonioso nell'uso della memoria – certo potevo fare di più, ma con 4k di SID, 2k di caratteri custom, labirinti e codice, si fa comunque presto a riempire gli 8k disponibili per la configurazione base.

La memoria di ogni programma può essere suddivisa in sezioni, e concettualmente si riconoscono tre di esse: text, data e bss. La sezione text contiene il codice vero e proprio che definisce la logica di programma, le operazioni che devono essere compiute, le decisioni da prendere, i salti alle subroutine (e le subroutine stesse) e così via; la sezione data contiene invece dati già inizializzati, e possibilmente anche immutabili, come ad esempio una mappa caratteri, un labirinto, o una lookup table (cioè una tabella contenente dei valori precalcolati, per snellire i calcoli a tempo di esecuzione); la sezione bss infine contiene dati non inizializzati, ed è un'area che viene popolata direttamente durante l'esecuzione del programma.

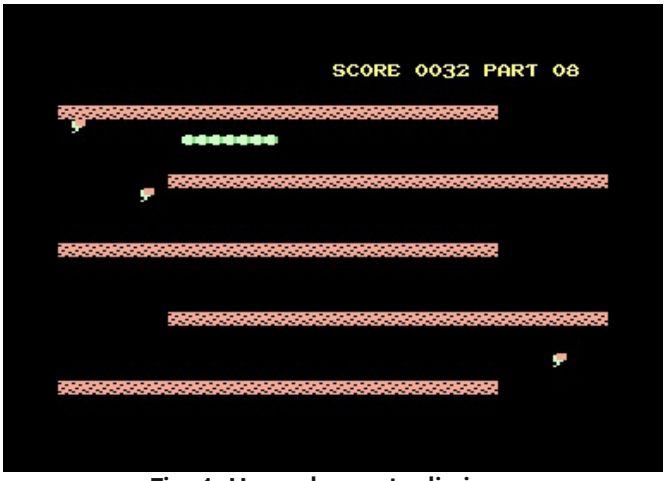


Fig. 1: Una schermata di gioco

Conoscendo queste basi, essenziali per chi si cimenta nella programmazione in assembly, si capisce fin da subito che certe sezioni del nostro programma, come la text e la data, che contengono il programma stesso e i labirinti, debbano necessariamente essere inserite all'interno della nostra ROM, mentre la bss, che invece conterrà, a tempo di esecuzione, le informazioni sullo stato del gioco, non ha motivo di occupare spazio nella nostra limitata ROM. Tra l'altro, il contenuto della ROM non può certamente essere cambiato durante l'esecuzione del gioco, a differenza del .prg che invece viene ricopiato in RAM al caricamento, quindi sprecare bss per metterla su ROM non ha alcun senso. Perciò, per esempio, le variabili per il punteggio, per localizzare la testa e la coda del serpente, e così via, non occuperanno nessuno spazio (se messe nella sezione giusta) se non un piccolo riferimento che l'assemblatore terrà a mente per noi sulla macchina di sviluppo.

Nella seguente tabella è visibile la mappa di memoria del gioco prima di iniziare la conversione: facendo un uso accorto delle sezioni, si vede, per esempio, come nessun riferimento nella zeropage venga allocato nel .prg, e come tutti i byte essenziali si concentrino in una piccola zona di memoria intorno all'indirizzo $1000.

Address	PRG	Description
$0000 - $0001	no	hardware
$0002 - $00FF	no	zero page pointers
$0100 - $07FF	no	*free ram*
$0800 - $0FFF	yes	initialized data segment + BASIC autostart
$1000 - $1FFF	yes	SID tune
$2000 - $27FF	yes	custom charset
$2800 - $xxxx	yes	program logic
$xxxx - $CCFF	no	*free ram*
$CD00 - $CDFF	no	data segment (not-initialized vars)
$CE00 - $CEFF	no	list X
$CF00 - $CFFF	no	list Y
$D000 - $DFFF	no	I/O
$E000 - $FFFF	no	Kernal

Tabella 1: Mappa della memoria

Conoscere il proprio assemblatore è fondamentale per organizzare il codice nella maniera più intelligente possibile. Ad esempio, con dasm si possono contrassegnare i segmenti inizializzati con SEG e quelli non inizializzati con SEG.U, mantenendo la comodità dei riferimenti, come si vede nell'estratto:

```
    SEG.U zeropageSegment
    org $02

; Generic src/dst copy pointers
srcPointer DS 2
dstPointer DS 2

; Music
    SEG sidSegment
    org $1000
    INCBIN "music.sid"

; Program
    SEG text
    lda (srcPointer),y
    sta (dstPointer),y

; Lists
    SEG.U listSegment
    org $ce00
listX DS 256
listY DS 256
```

Come si vede dalla mappa di memoria, l'aver usato questa accortezza ha permesso di tenere il programma in una regione di memoria piccola. Piccola, ma non troppo: a questo punto infatti il programma pesa 8252 byte, che sono comunque 60 byte di troppo per poter entrare nella ROM da 8192 byte, senza contare che si devono ancora aggiungere altri byte di programma per gestire l'avvio da cartuccia.

60 byte di troppo però sono allettanti, perché sì, la mappa di memoria risulta compatta, ma solo apparentemente. Infatti ci sono diverse locazioni di memoria che sono state scelte perché obbligate, e che hanno lasciato qualche manciata di byte di spazio vuoto qua e là.

Per esempio, l'autostart del BASIC, che serve per far partire il gioco non appena viene caricato, senza che l'utente debba battere RUN o, peggio, qualche astruso SYS, deve necessariamente trovarsi alla locazione di memoria $801, affinché possa essere eseguito automaticamente, mentre la musica per il SID è stata scritta per poter essere riprodotta solo se si trova alla locazione $1000. Allo stesso modo, tra SID e mappa caratteri, che deve trovarsi a $2000, avanzano una manciata di byte.

Come quantificare quanto spazio "groviera" è effettivamente vuoto?
Di nuovo, l'assemblatore è il nostro migliore amico per risolvere questo tipo di dubbi, in maniera automatica.

```
LASTINIT SET .
;
; ... codice ...
;
ECHO "file.asm @ ",LASTINIT,"len:",(. - LASTINIT)
```

Con dasm si può infatti suddividere il programma in vari file e inserire, all'inizio e alla fine di ognuno di esso, due linee come quelle riportate, in modo che, in fase di compilazione, venga mostrato lo spazio utilizzato da ogni pezzo di codice.
Si scopre quindi che, un buco qui e un buco là, questo lascia un centinaio di byte di spazio vuoto, difficile però da riempire: troppo poco e troppo frammentato per la mappa di caratteri, abbastanza per il codice (almeno all'inizio) ma limitante nel lungo periodo, e sicuramente predisponente a fare più spaghetti code del necessario.
Anche se lo spazio libero è apparentemente sufficiente, in realtà è quasi del tutto inutilizzabile a causa della frammentazione. Ma nonostante ciò, aggiungo quindi il codice per l'avvio da cartuccia (vedere più avanti), e eseguendo un po' "l'algoritmo dello zaino", il codice viene spezzettato e imbucato nei più stretti anfratti, e la dimensione del programma viene portata ad appena 8190 byte, cioè con un margine di appena 2 byte di memoria libera. È un (parziale) successo!

Avvio da cartuccia

Quando il Commodore64 si avvia, questo tenta di determinare, tramite una combinazione di hardware e software, se è presente una cartuccia inserita nell'apposita porta di espansione.
Dal punto di vista software, il Kernal controlla se, all'indirizzo $8004, è presente la stringa "CBM80" in PETSCII, e, se la trova, salta all'indirizzo indicato nelle locazioni $8000-$8001 (ossia la prima parola della cartuccia).

Quando il Kernal avvia la cartuccia, omette di chiamare le routine di inizializzazione dell'I/O, indispensabili per poter usare video, tastiera, joystick, memorie di massa e via discorrendo.
Come si vede dall'estratto di codice, quindi, è opportuno

richiamarle manualmente.

Prima di avviare il programma vero e proprio, poi, questo deve essere ricopiato nelle locazioni originali di memoria, perché alcune parti non possono risiedere altrove (es. la musica per il SID).

```
SEG cartridgeSegment
    org $8000

cartridge SUBROUTINE
    WORD .coldstart
    WORD .warmstart

    ; CBM80 in PETSCII
    ; (autostart signature)
    BYTE #$c3,#$c2,#$cd,#$38,#$30

.coldstart:
    sei
    stx $d016
    jsr $fda3
    jsr $fd50
    jsr $fd15
    jsr $ff5b
    cli

.warmstart:
    ; copy to original location
    jsr copy
    ; jump to program entry
    jmp start
```

La cartuccia quindi è pronta e funziona, ma a questo punto viene spontaneo domandarsi: ma tutto quel codice a spaghetti di prima, è una cosa fatta bene? E tutto questo lavoro, è servito a qualcosa? Il gioco entra appena appena in 8KiB: e se un domani dovesse essere ampliato?

Compressore RLE

Una delle tecniche per risparmiare spazio è la compressione dei dati, che sfrutta le ridondanze intrinseche di essi. Si pensi ad esempio al labirinto che deve percorrere il serpente: esso è realizzato per mezzo di numerose tessere vicine tra loro, tutte uguali e che formano dei motivi ripetitivi, come linee verticali e orizzontali. Uno degli algoritmi di compressione che possono essere impiegati per comprimere questo tipo di dati è il cosiddetto Run Length Encoding (RLE), e il concetto che sta alla base è piuttosto banale: perché memorizzare tante tessere X tutte uguali, una di seguito all'altra, quando si potrebbe semplicemente memorizzare un'informazione più compatta come "ci sono N tessere di tipo X in fila"? Una sequenza come "AAABBB" diventerebbe quindi "3A3B". Con questa semplice e banale osservazione, il labirinto di un intero livello, che in memoria occuperebbe 40x24 = 960 byte, può essere ridotto a poche decine di byte. Per esempio, nello snake6502, il labirinto "Training", dopo essere stato compresso con RLE, occupa appena 69 byte: un risparmio di memoria del 93%!

Questo tipo di compressione però era già stata applicata ai livello dello Snake6502 ben prima di pensare a fare la cartuccia, quindi gli 8190 byte sono già al netto della compressione. In sostanza sono sempre al punto di partenza.

Un problema della compressione RLE è che è troppo semplice ed è in grado di sfruttare solo ridondanze evidenti e grossolane. Come si è visto, funziona quindi molto bene per mappe e labirinti, ma è tremendamente inefficace contro il codice o le mappe di caratteri, che hanno ridondanze più sottili. Usare RLE sul codice potrebbe anzi rivelarsi altamente controproducente, aumentando la dimensione finale del programma anziché diminuirla.

Com'è possibile che ciò accada? Quando si comprimono dei dati, lo si fa per risparmiare memoria, ma a spese di aggiungere delle nuove strutture dati e una subroutine di decompressione, che vanno ad occupare spazio per cose che prima non c'erano: la speranza è che gli effetti benefici della compressione siano abbastanza grandi da andare a compensare la perdita dovuta all'aggiunta di questo overhead. Con la compressione RLE, questo compromesso non porterebbe vantaggi: serve perciò un algoritmo più furbo.

Compressore LZ

L'algoritmo LZ, così chiamato dal nome di Lempel e Ziv che lo hanno ideato, riconosce non solo quando un byte si ripete, ma anche quando si ripetono sequenze più lunghe. Ad esempio, se RLE non è in grado di comprimere in nessun modo la sequenza "ABABAB", LZ riconosce che "AB" è una sottostringa che si ripete 3 volte, e che quindi può concettualmente memorizzare qualcosa come "3AB". In pratica, l'algoritmo è più complesso e ne esistono diverse versioni.

L'idea di base di LZ è di vedere i dati da comprimere come un flusso di byte, e, man mano che lo si scorre, di riutilizzare le stringhe già incontrate puntandole indietro nel flusso. L'immagine seguente riporta un esempio abbastanza triviale in cui si vede come i 6 byte precedenti siano stati compressi in 4 facendo uso di una struttura speciale (offset;length), dove l'offset specifica la posizione della sottostringa originale, mentre length è naturalmente la

A	B	A	B	A	B
A	B	(-2;2)	(-4;2)		

Fig. 2: Una sequenza di byte prima e dopo la compressione con LZ

sua lunghezza.
Chiaramente questo è solo un esempio che necessita di essere approfondito in maniera più rigorosa, ma prima di farlo è bene soffermarsi su alcuni particolari.

Se la struttura (offset;length) ha una dimensione fissa, quanti bit si possono dedicare all'offset e quanti a length? Dedicare molti bit all'offset consentirebbe di recuperare stringhe di bit molto lontane spazialmente, a costo di limitare però la lunghezza di queste, e viceversa, dedicare molti bit alla length consentirebbe di recuperare stringhe molto lunghe, ma solo vicine.
Il numero di bit dedicati a offset e length può essere scelto in maniera tale da aver il miglior compromesso di compressione, che naturalmente è diverso a seconda dei dati che si vogliono comprimere, e spesso si scontra anche con i limiti tecnici dell'hardware: perciò, in sostanza, dipende dalla specifica implementazione.

La libreria liblzg implementa una versione dell'algoritmo LZ abbastanza semplice, tale che può essere facilmente portata su moderni sistemi embedded, ma anche su computer un po' retrò, come il Commodore64. Ne esiste infatti una versione "mini" per il processore Motorola 68000 e, naturalmente, per il MOS 6502. La libreria è corredata con un'utilità di compressione, scritta in linguaggio C, che può essere fatta girare su qualunque sistema operativo moderno, e che risulta particolarmente comoda per il cross-sviluppo di oggigiorno.

"liblzg-mini", al fine di consentire il miglior compromesso possibile in fase di compressione, anziché impiegare una sola struttura (offset;length), ne impiega ben 4, e può usare quella più opportuna, a seconda di quanto è distante la stringa ridondante di byte o di quanto è lunga. Nella Figura 3, in cui ogni quadratino rappresenta un bit, sono riportate le quattro strutture: near copy, short copy, medium copy e distant copy, che vengono scelte alla bisogna.
Si noti come, per risparmiare quanti più preziosi bit possibili, in realtà essi non rappresentano direttamente offset e length come numeri binari, ma una loro variante. Per esempio, siccome gli offset da 0 a 23=8 possono essere agevolmente rappresentati tramite la struttura "near copy", la struttura "short copy", pur disponendo di 6 bit di offset, non rappresenta i valori da 0 a 26=64, ma quelli da 8 a 64+8=72. Analogamente, i 5 bit della length, che potrebbero rappresentare solo i valori da 0 a 25=32, vengono invece utilizzati per indicizzare una lookup table, che contiene una selezione di valori di lunghezze comuni, da 2 a 128. Tutti i dettagli sono consultabili direttamente nei commenti del codice sorgente.

Un ultimo problema che non è stato affrontato è quello della distinzione delle varie strutture dati. Per distinguerle in fase di decompressione viene apposto loro un prefisso, denominato in gergo marker: naturalmente ne esistono di 4 tipi, tanti quanti sono le strutture.
Tutti i byte che poi non sono marker né strutture, sono trattati come byte originali, non compressi, che fanno parte del flusso così come sono, e vengono denominati literal. Un campione di flusso di byte compressi appare quindi come una cosa del genere (vedi Figura 4).
Rimane da risolvere un ultimo problema: cosa succede se un marker è uguale ad un literal? Se il marker fosse $77 e si volesse invece rappresentare letteralmente un byte $77? In tal caso, liblzg risolve il problema considerando quel byte come un marker, ma ponendo il primo byte della corrispondente struttura al valore speciale $00, che sarebbe comunque inutilizzabile per altri scopi (offset = 0, length = 0). Usando quindi il marker più un pezzo di struttura, è possibile rappresentare anche quei literal uguali ai marker. In questi rari casi, però, un dato che in origine sarebbe stato grande 1 byte, compresso diventa di... 2 byte! È un compromesso che dobbiamo accettare, dati i vincoli tecnici.

Impiegando il compressore LZ per lo snake6502, la dimensione finale del programma passa dunque da 8190 ad appena 5614 byte, cioè ridotta al 69% dell'originale! Però, la routine di decompressione, non proprio banale,

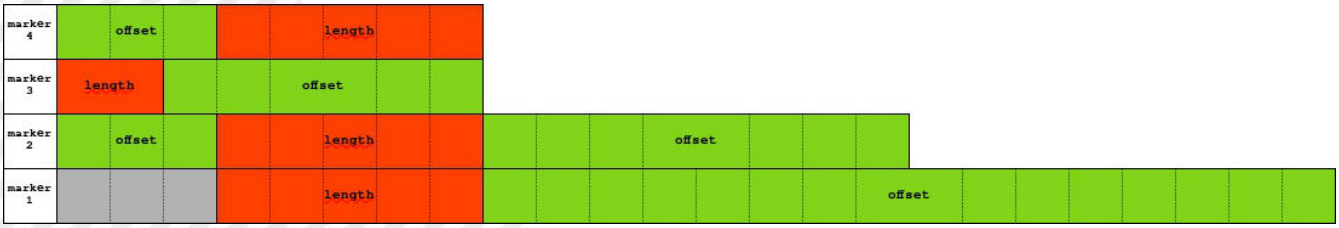


Fig. 3: Le 4 strutture dati utilizzate da liblzgmini

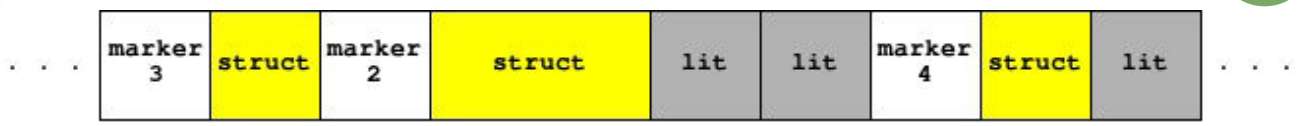


Fig. 4: Un esempio campione di flusso di byte compressi

deve comunque essere messa sulla cartuccia, non compressa, e occupa circa 400 byte. Nonostante questo, alla fine dei conti, la cartuccia pesa 6029 byte, e ci sono quindi ancora più di 2KiB liberi per eventuali espansioni future: un bel risultato! Ci siamo.

Impacchettamento

Il gioco viene quindi dapprima assemblato come al solito, eventualmente lasciando pure qualche spazietto vuoto qua e là per questioni di manutenibilità e leggibilità – tanto LZ si occuperà di ridurre al minimo lo spreco, dopodiché viene compresso con lzgmini nell'ambiente di sviluppo, diventando un bel file compatto snake.pak.lz. Questo file viene quindi concatenato alla routine di decompressione, e messo sulla cartuccia che, non appena avviata, lo scompatterà in memoria nella posizione originale, e ci permetterà finalmente di giocare.

Per rendere l'operazione di decompressione più scenografica, si può aggiungere alla routine una semplice istruzione sta $d020 da eseguire ad ogni ciclo, per ottenere qualche secondo di "loading bars".

Hardware Cart

Preparato il file binario con tutto questo cinema dentro, e estensivamente provato sull'emulatore, giunge infine l'ora di trasferirlo fisicamente su una cartuccia.

Il circuito da realizzare è piuttosto semplice, ma se – come nel caso del sottoscritto – non si è dei professionisti del settore, conviene affidarsi a qualche soluzione già pronta, per esempio alla Versa64Cart.

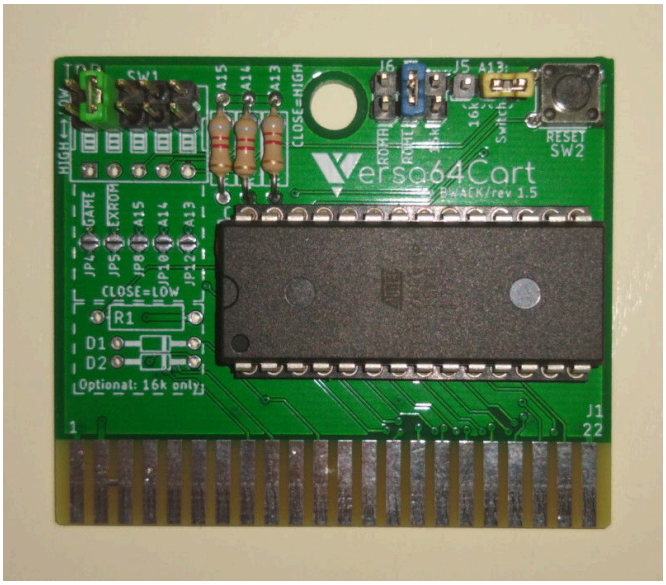


Fig. 5: La Versa64Cart con snake6502

La Versa64Cart è una scheda versatile che permette di realizzare cartucce da 8 e 16KiB per il Commodore64, sia in modalità standard che in modalità ultimax. Per una decina di Euro possono essere fatte stampare alcune copie di essa, presso una qualunque fabbrica di circuiti stampati, e per un'altra decina di Euro è possibile acquistare (in abbondanza) tutti i componenti che servono per saldarla a casa.

Tecnicamente servono soltanto:
- la scheda;
- una EPROM, o meglio ancora una EEPROM compatibile 27C64 (8KiB), 27C128 (16KiB), 27C256 (32KiB);
- un condensatore da 100nF;

In realtà, per godere della versatilità di configurazione, conviene procurarsi anche:
- un socket DIP "largo" a 28 pin, praticamente indispensabile se non si vuole saldare la ROM permanentemente;
- pulsante, comodo per il reset;
- DIP switch a 5 vie;
- resistenze (8-10 kΩ), pin header, e jumper assortiti.

Quando il Commodore64 viene acceso, questo determina il tipo di cartuccia attaccata tramite due segnali hardware, EXROM e GAME, attivi bassi, cioè quando sono portati a 0V. Affinché la cartuccia poi effettivamente risponda, quando interpellata dal Commodore, è necessario collegare il pin OE (Output Enable) della ROM al giusto segnale di ROM enable, che deve essere scelto tra ROML per le cartucce montate a $8000 e ROMH per quelle montate a $E000.

Per montare quindi la cartuccia dello snake6502, che è da 8KiB, modalità standard, indirizzo $8000, deve essere attivato il solo segnale EXROM (il secondo interruttore del DIP switch) e collegato l'OE della ROM al segnale ROML (jumper J6).

(Nell'immagine allegata viene usato un jumper in alternativa al DIP switch: tutte le configurazioni possibili sono elencate nel dettagliato manuale che è possibile scaricare online, insieme agli schemi circuitali)

Programmatore di EEPROM

L'ultimo problema da affrontare prima di poter effettivamente giocare è la scrittura della EPROM.

Anche se non è propriamente in linea con lo spirito del

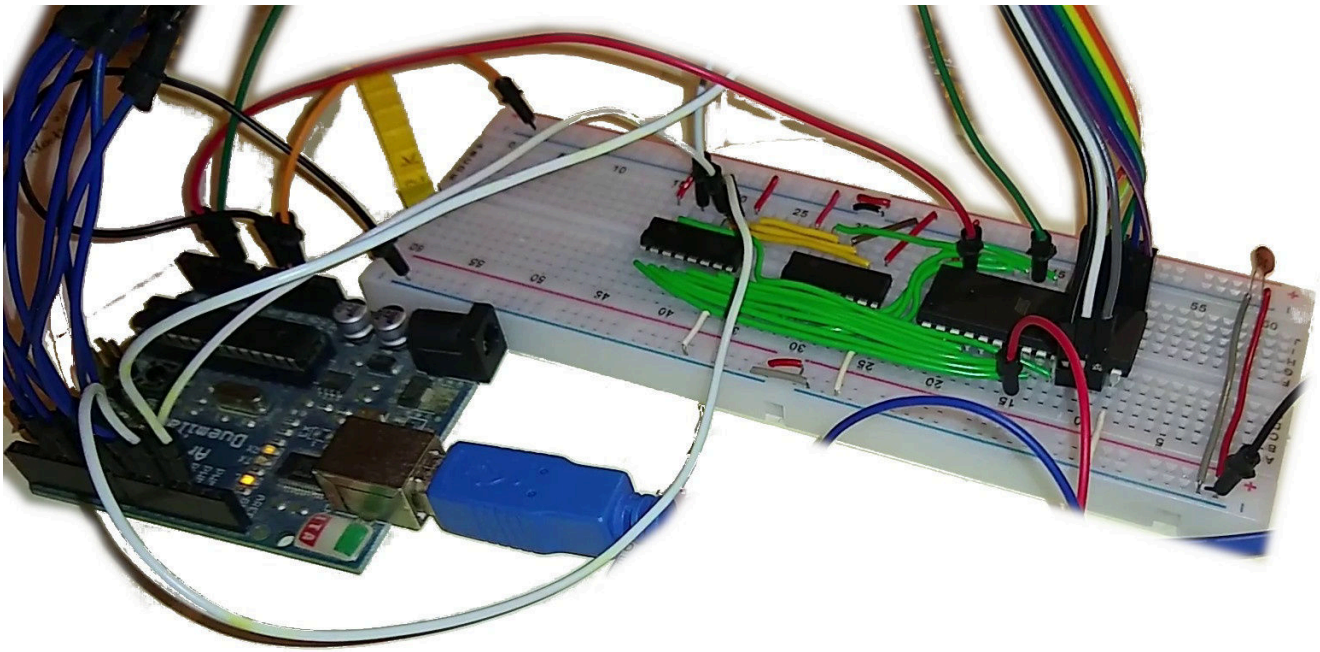


Fig. 6: Il programmatore di EEPROM artigianale costruito con Arduino

tempo, per semplicità conviene utilizzare una "moderna" EEPROM (Electrically Erasable Programmable ROM), così da poterla cancellare e riutilizzare comodamente in caso di errori, e anche per provare eventuali nuove versioni.
Le EEPROM 28Cxx (cioè 27Cxx compatibile) possono essere acquistate per pochi Euro, mentre i programmatori di EEPROM non sono costosissimi, ma, a livello amatoriale, può anche convenire utilizzare un più economico microcontrollore qualsiasi, come il famoso Arduino, specialmente nel caso in cui lo si possieda già, come il sottoscritto. Ben Eater è famoso nell'ambiente amatoriale per averne realizzato una versione didattica e riadattabile, perciò conviene ispirarsi al suo lavoro.

Un comune Arduino Duemilanove/Uno dispone di 20 piedini, ma per programmare la ROM ne servono di più:
13 piedini per gli indirizzi (213 = 8KiB)
8 piedini per i dati (8 bit = 1 byte)
2 piedini di controllo
2 piedini per la comunicazione seriale

I piedini per gli indirizzi, i più numerosi, sono fortunatamente necessari solo in output, perciò possono essere rimpiazzati agevolmente da un paio di shift register, ossia dei circuiti digitali i quali, passatagli una stringa di bit in serie, la mostrano in parallelo sui loro 8 piedini. Gli shift register della serie 595 possono essere collegati in cascata, permettendo di ottenere un numero arbitrariamente grande di piedini di output.
Il firmware per Arduino, ampliato, si occupa quindi di gestire la scrittura della EEPROM, selezionando gli indirizzi per la scrittura tramite gli shift register, leggendo il contenuto della cartuccia che gli viene passato dal computer di sviluppo tramite la porta seriale. Poiché i piedini per i dati sono direttamente collegati ad Arduino, poi, alla fine, è possibile rileggere la ROM per verificare l'integrità di quanto scritto.

Segue lo pseudocodice del programmatore di EEPROM:

```
/* Scrittura */
```

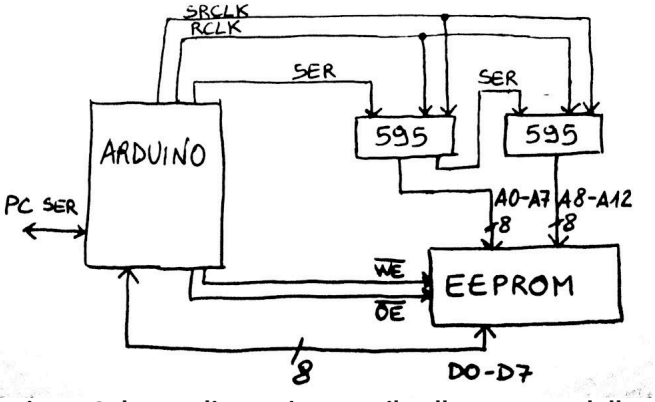


Fig. 7: Schema di massima per il collegamento della EEPROM all'Arduino

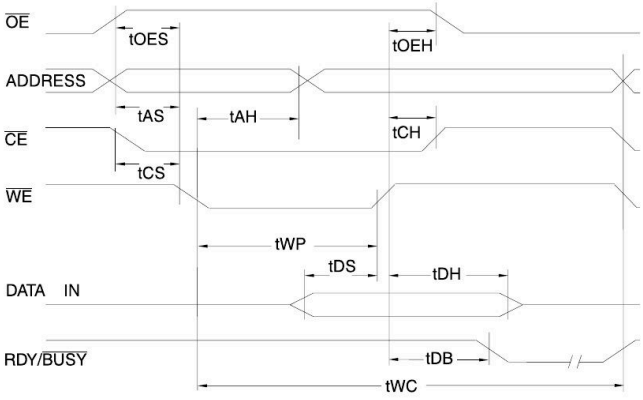


Fig. 8: Diagramma di temporizzazione per la scrittura

Fig. 9: Snake6502 eseguito su un Commodore 128D da cartuccia (in modalità 64)

```
indirizzo = 0
for indirizzo
     leggi dato da seriale
     scrivi indirizzo su shift-register (A0-A12)
     scrivi dato su piedini (D0-D7)
      comanda scrittura (WE) (HIGH → LOW → HIGH)
      indirizzo++
next

/* Lettura */
indirizzo = 0
for indirizzo
       scrivi indirizzo su shift-register (A0-A12)
        comanda output (OE) (HIGH → LOW → HIGH)
        leggi dato da piedini (D0-D7)
        scrivi dato su seriale
        indirizzo++
next
```

A questo punto non rimane che montare la EEPROM sulla Versa64Cart, inserirla nel Commodore64 e... giocare! ☺

In realtà la foto è su un Commodore 128D, ho sbagliato proprio sul finale!

Riferimenti

- dasm, https://dasm-assembler.github.io/
- EEPROM programmer by Ben Eater, https://github.com/beneater/eeprom-programmer
- liblzg, https://liblzg.bitsnbites.eu/
- snake6502, https://git.giomba.it/giomba/snake6502
- Versa64Cart, https://github.com/bwack/Versa64Cart
- VICE, https://vice-emu.sourceforge.io/

Programmare senza GOTO con lo ZX Spectrum

di Alberto Apostolo

Il BASIC cablato nella ROM del Sinclair ZX Spectrum non possiede i costrutti UNTIL, WHILE, IF-THEN-ELSE della programmazione strutturata.

Tuttavia, si possono ugualmente scrivere programmi strutturati, avvalendosi delle possibilità offerte dal BASIC Sinclair.

Chi vuole applicare su altre versioni BASIC le tecniche trattate in questo articolo, deve consultare i propri manuali di riferimento per verificare la fattibilità.

LE STRUTTURE ITERATIVE UNTIL E WHILE

Il BASIC Sinclair permette l'uso di espressioni condizionali che possono restituire il valore intero 1 (TRUE) o 0 (FALSE) da impiegare in normali espressioni aritmetiche.

Inoltre, è noto che il BASIC Sinclair consente di alterare i valori memorizzati nelle variabili di controllo di un ciclo FOR-NEXT all'interno del ciclo stesso.

Altri linguaggi non possiedono tale caratteristica (per esempio, il FORTRAN 77 e i cicli DO-CONTINUE).

In Fig. 1 è mostrata la conversione delle strutture UNTIL e WHILE in cicli FOR-NEXT del BASIC Sinclair.

Naturalmente, in caso di strutture annidate, è obbligatorio usare variabili di controllo con nomi diversi nei cicli FOR-NEXT.

LA STRUTTURA IF-THEN-ELSE

Ogni "Sinclairista" che si rispetti conosce bene l'espressione aritmetica in Fig.2, usata spesso nella programmazione dello ZX Spectrum per calcolare la posizione di caratteri lungo l'ascissa dello schermo spostati con i tasti 5 (sinistra) e 8 (destra).

Non è l'unica maniera di implementare una struttura IF-THEN-ELSE. Per esempio, si può ricorrere al comando GOSUB "calcolato" (Fig.3) oppure, in alternativa, utilizzare due cicli FOR-NEXT consecutivi (Fig.4). Sempre in Fig.4, notate la variabile A necessaria per "sincronizzare" i due cicli FOR-NEXT.

Anche qui, in caso di strutture annidate, è obbligatorio usare variabili di controllo con nomi diversi nei cicli FOR-NEXT.

```
UNTIL condition          FOR F = 0 TO 1
   [statements]             statements
                            F = condition
                         NEXT F

WHILE condition          FOR F = 1-(condition) TO 0
   [statements]             statements
                            F = -(condition)
                         NEXT F
```

Fig. 1

```
LET X = X - ( INKEY$="5" AND X > 0 )
          + ( INKEY$="8" AND X < 29)
```

Fig. 2

```
                          ... GOSUB 2000 -
                                    1000 * (cond)
IF cond                   ...
   THEN                    999 STOP
        statement_1_1     1000 statement_1_1
        ...                ... ...
        statement_1_n      ... statement_1_n
   ELSE                   1999 RETURN
        statement_2_1     2000 statement_2_1
        ...                ... ...
        statement_2_m      ... statement_2_m
END                       2999 RETURN
```

Fig. 3

```
REM IF
          LET A = cond
REM THEN
          FOR F = 1-A TO 0

             statement_1_1
             ...
             statement_1_n

          NEXT F
REM ELSE
          FOR F = A TO 0

              statement_2_1
              ...
              statement_2_m

          NEXT F
```

Fig. 4

UN GIOCO SCRITTO IN UNA SOLA RIGA DI BASIC

Il gioco riportato in Fig. 5 si può riscrivere senza GOTO (Fig. 6) e perfino in una sola riga di BASIC (Fig.7) come farebbe un vero "One-Liner" (chi sa scrivere programmi non banali in una sola riga di codice).

```
  10 LET S=0: LET I=15
  15 PRINT AT 0,I;
  20 IF SCREEN$ (0,1+I)="W" THEN
 GO TO 90
  25 PRINT "▀▀"
  30 IF INKEY$<>"x" THEN GO TO 5
5
  35 FOR J=-8 TO 8
  40 PRINT AT 8-ABS J,I+1;("T" A
ND J<0)+(" " AND J>=0)
  45 NEXT J
  50 LET S=S-10
  55 LET I=I-(INKEY$="n" AND I>0
)+(INKEY$="m" AND I<29)
  60 PAUSE 5
  65 PRINT AT 14,RND*31;"W"
  70 POKE 23692,255
  75 PRINT AT 21,31'''
  80 LET S=S+1: GO TO 15
  90 PRINT S
```

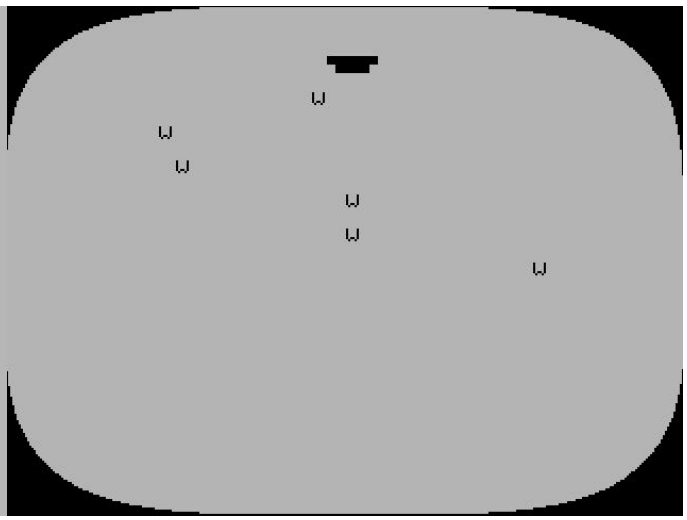


Fig. 5

```
 100 LET S=0: LET I=15
 110 PRINT AT 0,I;
 120 FOR F=0 TO 1
 130 PRINT "▀▀"
 140 FOR G=INKEY$<>"x" TO 0
 150 FOR J=-8 TO 8
 160 PRINT AT 8-ABS J,I+1;("T" A
ND J<0)+(" " AND J>=0)
 170 NEXT J
 180 LET S=S-10
 190 NEXT G
 200 LET I=I-(INKEY$="n" AND I>0
)+(INKEY$="m" AND I<29)
 210 PAUSE 5
 220 PRINT AT 14,RND*31;"W"
 230 POKE 23692,255
 240 PRINT AT 21,31'''
 250 LET S=S+1
 260 PRINT AT 0,I;
 270 LET F=SCREEN$ (0,1+I)="W"
 280 NEXT F
 290 PRINT S
```

Fig.6

```
1000 LET S=0: LET I=15: PRINT AT
 0,I;: FOR F=0 TO 1: PRINT "▀▀"
: FOR G=INKEY$<>"x" TO 0: FOR J=
-8 TO 8: PRINT AT 8-ABS J,I+1;("
T" AND J<0)+(" " AND J>=0): NEXT
 J: LET S=S-10: NEXT G: LET I=I-
(INKEY$="n" AND I>0)+(INKEY$="m"
 AND I<29): PAUSE 5: PRINT AT 14
,RND*31;"W": POKE 23692,255: PRI
NT AT 21,31''': LET S=S+1: PRINT
 AT 0,I;: LET F=SCREEN$ (0,1+I)=
"W": NEXT F: PRINT S
```

Fig.7

Conclusioni

E' stato realizzato ciò che a prima vista sembrava impossibile, eliminare l'uso del comando GOTO, grazie alla possibilità di modificare le variabili di controllo dei cicli FOR-NEXT e all'uso delle espressioni condizionali nel calcolo aritmetico.

Eliminare il GOTO facilita la scrittura di programmi più compatti, che addirittura possono essere contenuti in una sola riga di codice.

Infine, si può fare a meno dei comandi POKE alle variabili di sistema che puntano alle istruzioni da eseguire (NEWPPC 23618/23619 e NSPPC 23620).

Appendice

Nel 1985, anche io ho partecipato a un "One-Line" contest organizzato dalla rivista italiana "Sinclair Computer" [SC85].

Non c'erano premi da vincere ma solo la soddisfazione di vedere pubblicato il proprio programma (Fig.8).

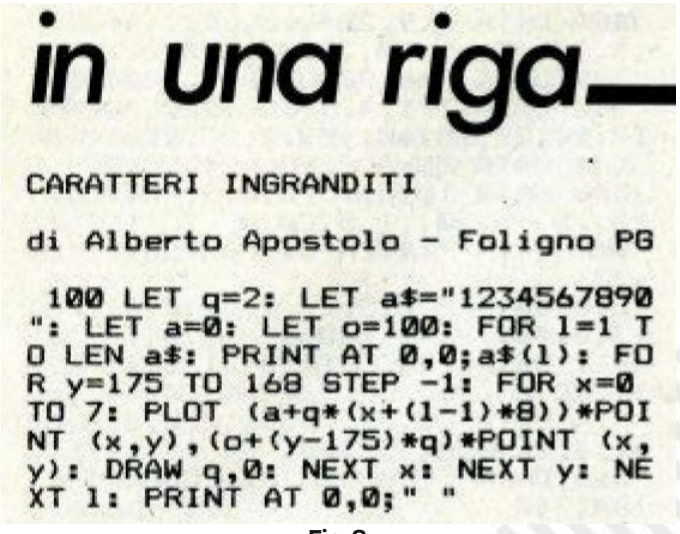


Fig.8

Bibliografia

[Bon83] R.Bonelli, "Alla scoperta dello ZX Spectrum", Gruppo Editoriale Jackson, 1983.

[SC85] AA.VV., "in una riga", Sinclair Computer n.15, Jul-Aug 1985, Pag.36. https://archive.org/details/Sinclair-Computer-15

Un bit di rarità

(rovistando qua e là)

I font Sinclair per creare adesivi personalizzati

di Alberto Apostolo

Cercando su Internet "sinclair computer logo" si trovano tante belle immagini (ufficiali e non) del logo Sinclair che possono essere scaricate e stampate su adesivi per decorare i nostri device oppure i contenitori di circuti autocostruiti. Ma oltre alle figure, sarebbe bello stampare scritte adesive con i font Sinclair.

I font usati all'interno dei computer Sinclair si possono scaricare gratuitamente in formato TrueType dal sito dafont.com (Fig. 1, [DF20a], [DF20b]). Altri siti permettono di scaricare versioni simili ma occorre pagare.

Il "Sinclair logo font" che compone la scritta "sinclair" sull'involucro si trova in [FS20] (Fig.2) ma per scaricarlo occorre registrarsi. Tuttavia si osserva che alcune lettere ("o","g","l","s","z") si confondono con i numeri 0,9,1,5,2.

Un problema simile si riscontra anche in [BA20] (Fig.3) dove le lettere "v" e "x" potevano essere curate meglio graficamente e le lettere "o","s","z" si confondono con 0,5,2 rispettivamente.

Un'altra versione degna di nota è quella presente nella rivista britannica Sync [SY20], usata nei n.2-6 del Vol.3 e raccolta in Fig. 4 (sono stato costretto ad aggiungere le lettere mancanti "b","q","x","v","z", disegnandole con Paint).

Non ho trovato materiale riguardo ai font usati nelle scritte "ZX 81",

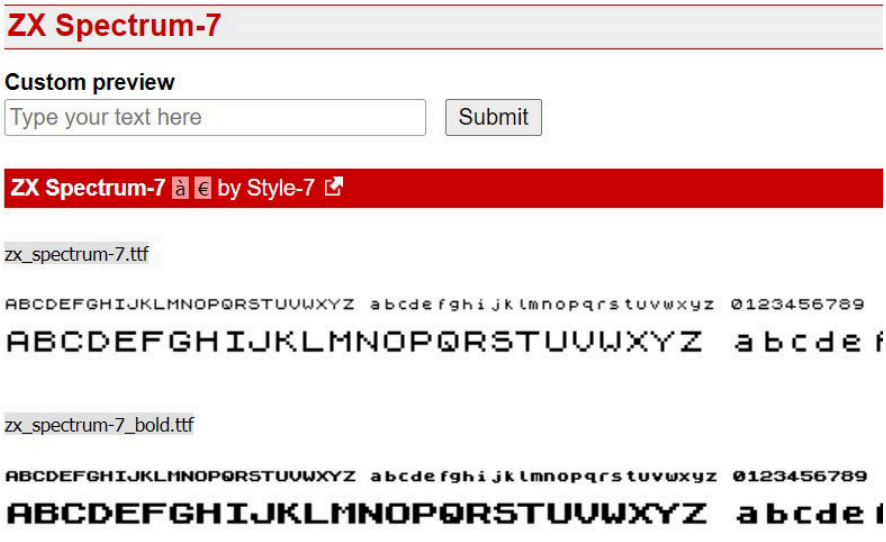


Fig. 1

Fig. 2

Download Bauhaus2015 here.

Oh and here's a font I just made based on the Sinclair logo! Use lower case only for best results. Named after Sinclair Research's founder, 'Uncle' Clive Sinclair. **Sir Clive** to you. Download here and I've also made a bold version called Sir Clive the Bold:

Fig. 3

"ZX Spectrum" e i comandi BASIC sulle tastiere ZX. Tuttavia si può rimediare in parte con il font Arial in grassetto nonostante alcune lettere (per es. "G") abbiano una forma molto diversa. In Fig. 5, il confronto tra il font Arial (a sinistra) e le vere scritte sui computer Sinclair (a destra).

Infine, cercando "sinclair QL font", mi sono imbattuto nel sito del mitico Dilwyn Jones. In [Jon20] si accede a una sezione aggiornata nel Marzo 2020, dove si possono scaricare font TrueType adatti per l'ambiente Windows e MacIntosh (Fig.6). Nella stessa pagina web vi sono anche link per scaricare versioni riferite a ZX81 e ZX Spectrum.

Fig. 4

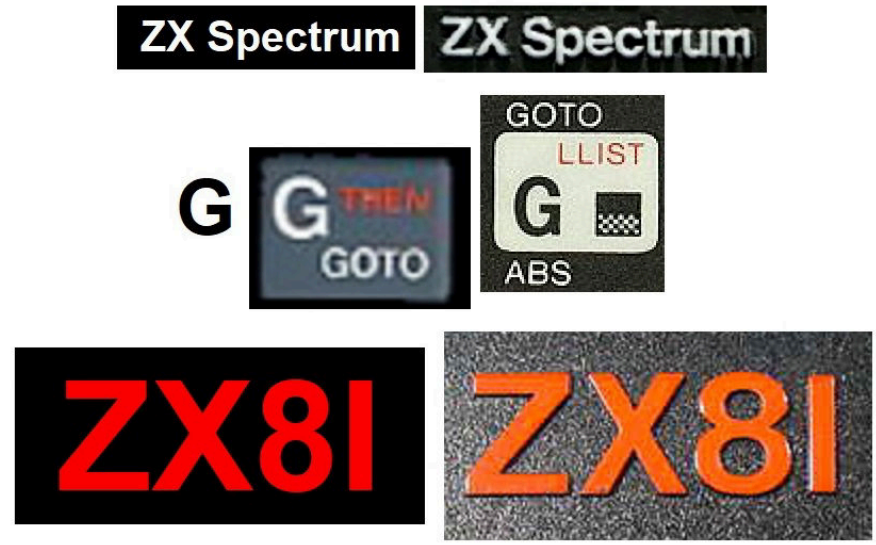


Fig. 5

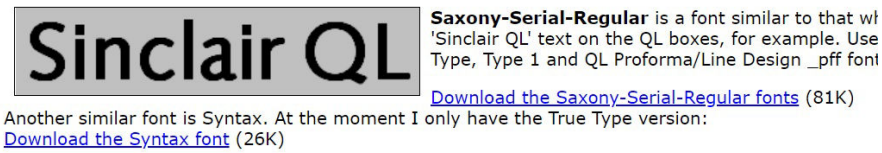


Fig. 6

Bibliografia

[BA20] (2020-10-24) retrieved from http://www.suppertime.co.uk/blogmywiki/2015/08/bauhaus2015-bitmap-font/

[DF20a] (2020-10-24) retrieved from https://www.dafont.com/zx-spectrum-7.font

[DF20b] (2020-10-24) retrieved from https://www.dafont.com/it/zx81.font

[FS20] (2020-10-24) retrieved from https://fontstruct.com/fontstructions/show/245226/sinclair_logo

[Jon20] (2020-10-24) retrieved from http://www.dilwyn.me.uk/fonts/

[SY20] (2020-10-24) retrieved from https://archive.org/details/syncmagazine

SymbOS - Windows su Amstrad CPC!

di Francesco Fiorentini

Incredibile! Semplicemente incredibile!

Non ho altre parole per descrivere questo software.
Che poi chiamarlo software è estremamente riduttivo; SymbOS e' un sistema operativo completo, con tanto di utilities, giochi, browser e strumenti di networking.
Detto cosi' oggigiorno non fa molta impressione, ma provate a pensare che questo sistemo operativo gira su macchine ad 8bit basate su processore Z80.
Eh già, SymbOS gira perfettamente su Amstrad CPC, MSX, Amstrad PCW ed Enterprise 64/128...
Come dicevo all'inizio: incredibile!

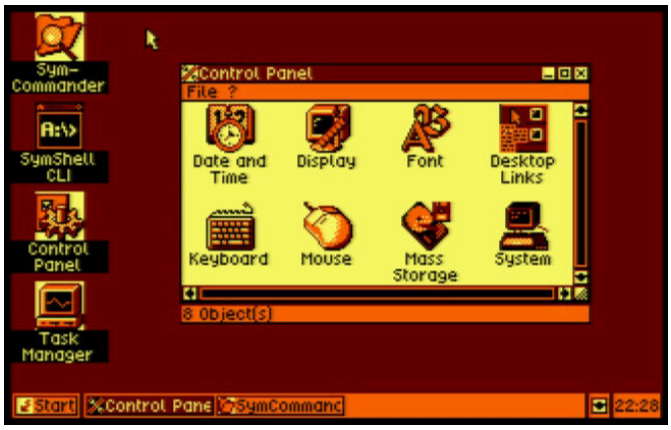


Fig. 1 - Non è Windows, ma SymbOS su Amstrad CPC

Un po' di storia

Negli anni 80 il GEOS pose delle solide basi per un sistema operativo grafico per il Commodore 64. Il S.O. sviluppato da Berkeley Softworks fu una vera e propria rivoluzione su macchine ad 8bit. Nonostante molti elementi fossero solo statici, era incredibile come una macchina ad 8bit con soli 64K di RAM, fosse in grado di far funzionare un sistema operativo grafico come il GEOS.
Ci furono molti tentativi di riprodurre lo stesso rilsulato anche su Amstrad CPC, ma per una ragione o per un'altra nessuno di questi fu in grado di avvicinarsi anche lontanamente a quanto sviluppato per il Commodore 64.

Fu così che, verso la fine del 2000, l'autore decise di realizzare un vero e proprio sistema operativo grafico per il CPC. D'altronde, molti CPC avevano di base 128K di memoria (il C64 solo 64K), una risoluzione di 320x200 con 4 colori (il C64 invece solo 2 colori per ogni area 8x8 alla risoluzione 320x200) e molti altri vantaggi rispetto al C64. Nacque cosi' l'idea del progetto SymbOS.

SymbOS significa "SYmbiosis Multitasking Based Operating System" ovvero Sistema Operativo Basato su SYmbiosis Multitasking. SymbOS sarebbe stato un vero e proprio sistema operativo moderno per Amstrad CPC, con real preemptive multitasking, gestione della memoria dinamica fino a 1024K ed un look and feel simile a MS Windows. In pratica avrebbe dovuto essere la dimostrazione di quello che sarebbe stato possibile realizzare su un CPC negli anni 80!
Lo sviluppo di SymbOS non è stato continuo, ma è proseguito dalla fine del 2000 sino almeno al 2017, quando la versione 3.0 di SymbOS per CPC & MSX & PCW & EP è stata rilasciato sul sito dell'autore. Da quel momento in poi non si hanno informazioni su eventuali altri aggiornamenti, ma la versione 3.0 è perfettamente funzionante e decisamente impressionante per cura, funzionalità e prestazioni!

SymbOS

SymbOS è disponibile per:
- per la serie Amstrad CPC 464/664/6128,
- per MSX1 (+V9990), MSX2, MSX2+ e MSX TurboR
- per tutti i modelli Amstrad PCW, 8xxx, 9xxx e la serie 10
- per le macchine Enterprise 64/128

Richiede un minimo di 128K ed un device di archiviazione di massa (floppy disk, IDE, SD...).

Fedele alla passione che ultimamente nutro per l'Amstrad CPC, ho voluto provare proprio questa versione.
Non possedendo pero' la macchina reale, il test è stato eseguito su WinAPE. All'inizio credevo di fare un torto all'autore, ma poi ho letto sul suo sito che anche lui, per preservare la macchine reali, sovente utilizza WinAPE per sviluppo e debugging e mi sono ricreduto.

Primo avvio...

Il manuale a corredo di SymbOS è molto dettagliato per quanto riguarda l'installazione della versione MSX, quanto invece e' piuttosto scarno di istruzioni rispetto al CPC.
Nel pacchetto Amstrad CPC troviamo 4 ROM e 3 dischetti.
Sebbene sia possibile eseguire SymbOS direttamente da dischetto, ho optato per un'installazione mista.
Avvieremo quindi il sistema operativo da ROM e poi

utilizzeremo il dischetto per l'esecuzione dei programmi non presenti in memoria.

Lanciamo quindi WInAPE e dal menu' **Settings -> General -> Memory**, andiamo a settare le opzioni come in Fig. 2. Selezionate **RAM 64K + 256K RAM Expansion** e **256K Silicon Disk**.

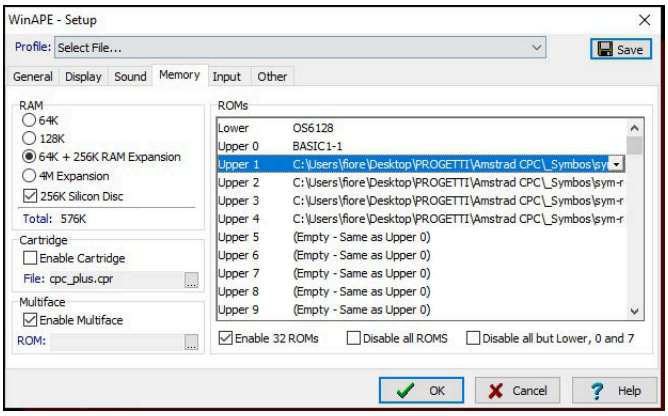


Fig. 2 - Settaggi RAM e ROM

Notate che ho inserito le **4 ROM** di SymbOS negli slot Upper1 fino ad Upper4:

- Upper1: sym-romA.rom

- Upper2: sym-romB.rom

- Upper3: sym-romC.rom

- Upper4: sym-romD.rom

Negli slot **Lower** ed **Upper0** ho inserito rispettivamente le ROM **OS6128** ed il **Basic 1-1**.

Assicuratevi che non ci siano floppy inseriti nei drive A: e B: e procedete ad un reset dell'emulatore: **Settings -> Reset**. Se tutto e' andato bene dovreste trovarvi di fronte ad una schermata come in Fig. 3.

Adesso, come specificato anche a video, basterà digitare **|Sym** o **|Symbos** per avviare il Sistema Operativo.

Fig. 3 - Avvio con le ROM di SymbOS

Fantastico! Un sistema operativo in tutto e per tutto simile a Windows è comparso sul vostro Amstrad CPC, vedi Fig.

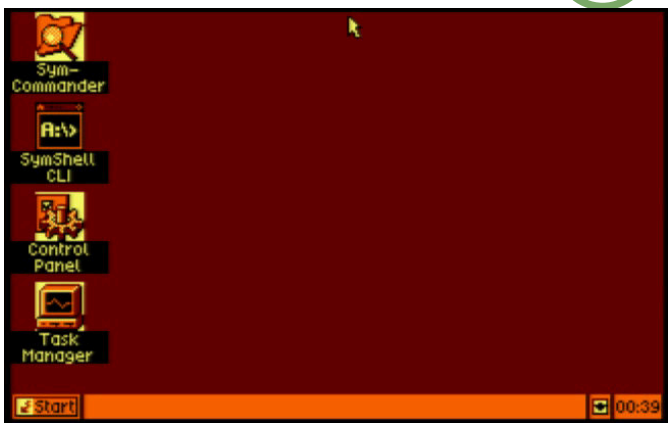


Fig. 4 - Il primo avvio di SymbOS su Amstrad CPC

4. La prima cosa che noterete è la freccia del mouse che, sorprendentemente, si muove fluida sul desktop. Come in Windows, bastera' un doppio click sulle icone per avviare il programma indicato ed un click destro del mouse per accedere alle impostazioni grafiche.

La seconda cosa che noterete è l'orologio in basso a destra. Anche questo perfettamente sincronizzato con l'orologio del computer ospite (Eh già, trovare il tempo per scrivere gli articoli di RMW tra gli impegni di lavoro ed una bimba di 18 mesi è un lusso che posso permettermi solo a notte tarda. :-) - NdR).

Proviamo quindi ad avviare il command prompt, che qui si chiama **SymShell CLI**. Doppio click sulla sua icona e... Un messagio di errore, vedi Fig. 5.

Cosa e' successo? Semplicemente dobbiamo indicare a SymbOS dove risiedono i programmi che vogliamo eseguire.

Fig. 5 - Avvio con le ROM di SymbOS

Inserite il disco **SymbOS-CPC-AppsStandard.dsk** (anche questo lo trovate all'interno dello zip che contiene le ROM) nel drive A con il comando **File** -> **Drive A:** -> **Insert Disc Image**. Dopodiche' dal menu' **Start** scegliete il comando **Run**... e nella finestra successiva cliccate sul pulsante **Browse...**. Si aprirà un'ulteriore finestra in cui dovrete sceglie un eseguibile qualsiasi, in questo caso va piu' che bene il file **Cmd.exe** (vedi Fig. 6), e confermate la scelta tramite i pulsanti **Open** ed **Ok**.

Dopo pochi istanti, il tempo di caricare il file da dischetto, dovreste trovarvi di fronte ad una finestra dall'aspetto

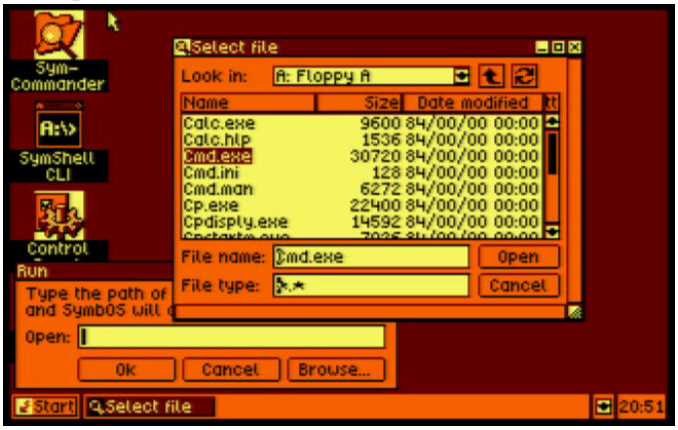


Fig. 6 - Avviamo l'eseguibile Cmd.exe

familiare. Sembra il command prompt del DOS, vedi Fig. 7. Ma le sorprese non sono finite qui. Provate a digitare il comando DIR. Ebbene si', la **SymShell CLI e' in tutto e per tutto compatibile con il prompt dei comandi di Windows.** Ho provato vari comandi quali: COPY, REN, DEL... Sono tutti perfettamente implementati.

E' possibile persino creare le directory con il comando **md**. Al successivo dir verranno visualizzate correttamente. Vedi

Fig. 7 - Il commad prompt di SymbOS

Fig. 8. Purtroppo, sebbene sia possibile entrare in una directory utilizzando il comando cd, la directory e' soltanto un mero artificio visivo, visto che non funziona come le directory vere e proprie del DOS; infatti eseguendo il comando

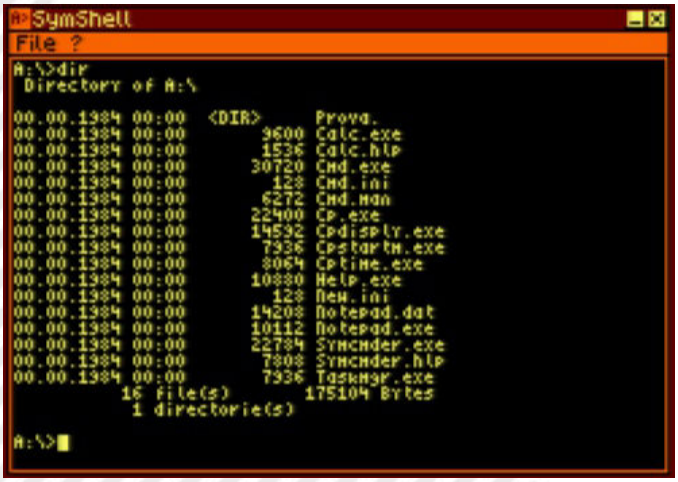


Fig. 8 - Il comando DIR; notate la directory 'Prova'

dir all'interno della directoy si vedra' anche il contenuto della root del disco. Sarebbe stato forse chiedere troppo.

Le meraviglie della SymShell CLI non sono pero' finite qui, tramite il menu' File e' possibile configurare l'aspetto della CLI. E le opzioni non sono certo poche, vedi Fig. 9. Possiamo scegliere la grandezza della shell: righe e colonne, il colore del testo, dello sfondo, del bordo e persino decidere se visualizzare la finestra in full screen.

Tutte le impostazioni settate verranno memorizzate nel file **cmd.ini** cosicche' ad ogni successivo riavvio della shell, la troveremo esattamente come l'avevamo configurata. Lo so

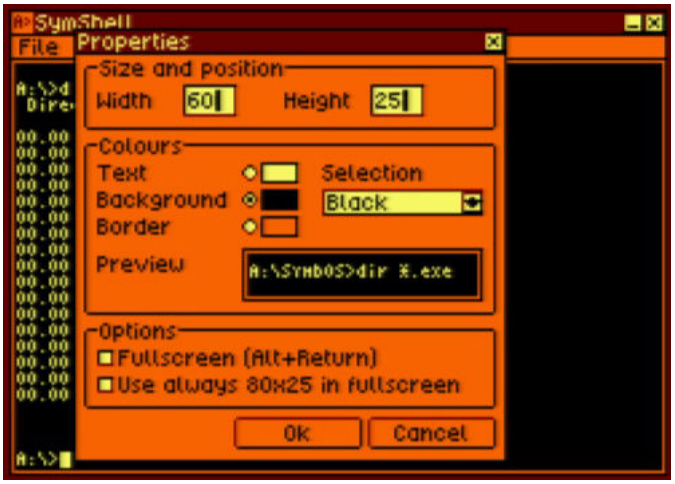


Fig. 9 - Le impostazioni della SymShell CLI

che mi ripeto, ma tutto cio' e' semplicemente incredibile.

Ovviamente le meraviglie non finiscono qui. Provate a fare doppio click sull'icona del **Control Panel**; sembra proprio di trovarsi di fronte al Pannello di Controllo di Windows 3.1 (vedi Fig. 10). La cosa soprendente inolte e' che tutte le icone del pannello di controllo sono funzionali alla configurazione di SymbOS; provate a cliccare su Mouse o Display e vi renderete immediatamente conto del numero incredibile di opzioni per configurare il sistema (Fig. 11). Interessante notare come la risoluzione grafica sia impostata

Fig. 10 - Il pannello di controllo ed il fuso orario...

di default a 320x200 a 4 colori ma che sia possibile anche impostarla a 640x200 a 2 colori. Proviamo quindi ad

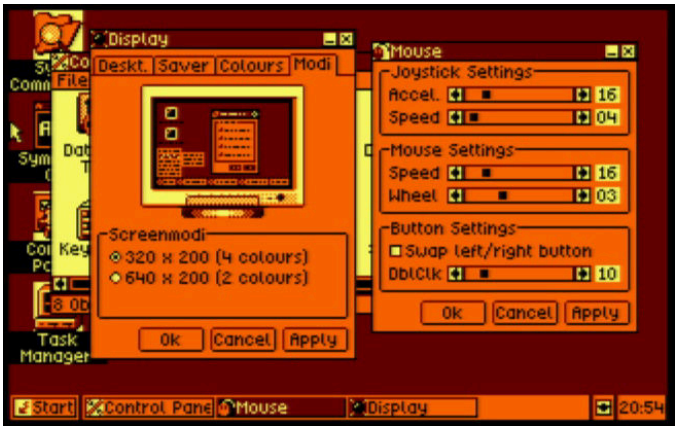


Fig. 11 - Opzioni di Display e Mouse

impostare questa risoluzione per vederne l'effetto. Lo spazio per le finestre aumenta sensibilmente ed e' possibile posizionarne anche quattro di fila, vedere Fig. 12, purtroppo i caratteri diventano meno leggibili ed i 2 soli colori a disposizione sono un limite, anche se non insormontabile. Terminiamo questa nostro primo viaggio all'interno di SymbOS parlando di **SymCommander**, un File Manager chiaramente ispirato al famoso Norton Commander. Come

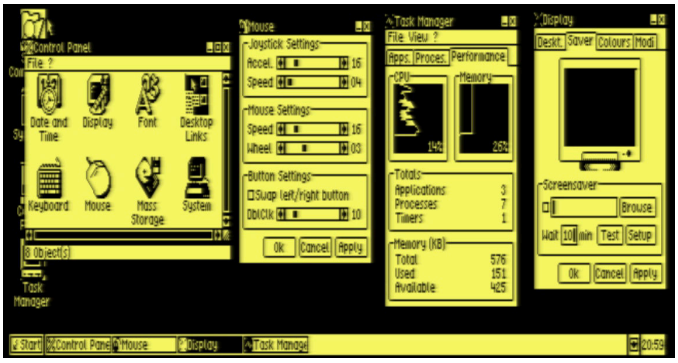


Fig. 12 - SymbOS impostato con risoluzione 640x200

potete vedere dalle immagini di Fig. 13 e 14 molte delle azioni piu' comuni da compiere su file e dischi, sono immediatamente raggiungibili tramite comodi pulsanti posti sotto le rappresentazioni grafiche delle directory.

Tutti gli altri comandi, e non sono pochi, sono raggiungibili dal comodo menu' dropdwon. Ovviamente anche SymCommander e' configurabile ad uso e consumo dell'utilizzatore. Il SymCommander e' accompagnato anche da un comodo file di help, richiamabile tramite il menu' ?. Da notare, in Fig. 14, come l'icona del SymCommander e del suo Help abbiano nella Task Bar due immagini differenti che consentono immediatamente di identificarli.

Conclusioni

Beh, cosa dire ancora? Il mio parere lo avrete intuito leggendo l'articolo. SymbOS e' un prodotto incredibile. Curato sin

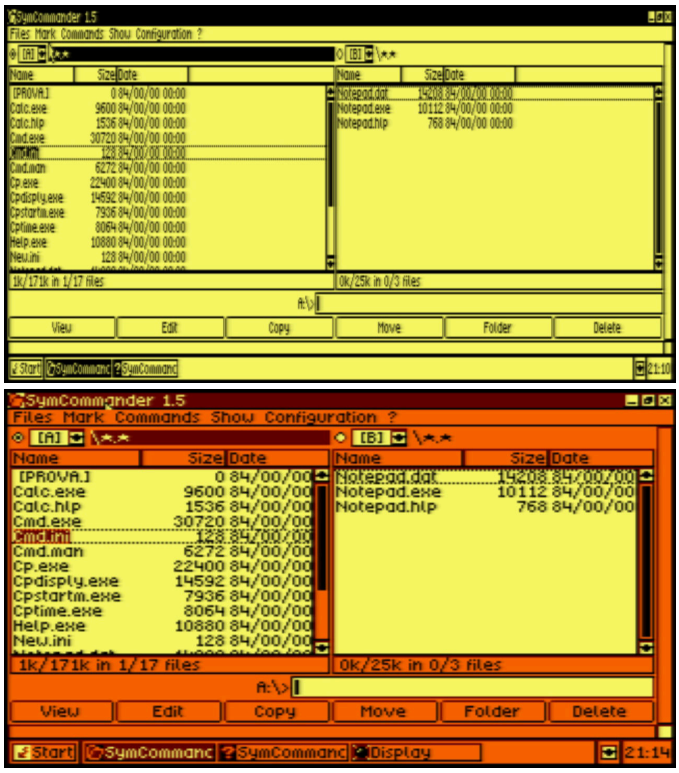


Fig. 13 e 14 - SymCommander a 640x200 e 320x200

nei minimi dettagli: provate a fare un doppio click sull'orologio nella task bar oppure a premere l'icona accanto ad esso (spoiler: e' la funzione mostra desktop); veloce ed affidabile (durante i miei test non ho mai riscontrato crash o comportamenti strani, solo un glitch nel SymCommander). E' possibile cambiare la risoluzione delle schermo al volo, non c'e' bisogno di riavviare o di caricare niente... Fosse stato disponibile sul mercato negli anni '80 l'Amstrad CPC avrebbe avuto sicuramente la sua killer application per il mercato office.

Ah, e tutto questo e' solo la punta dell'iceberg di SymbOS. Sul sito dell'autore troverete decine di altri programmi, giochi e persino utility per la connessione internet, che ovviamente non mancheremo di testare in successivi articoli di RMW. Il mio rammarico e' di non possedere una macchina reale per testarlo su HW originale, ma invito chi ne avesse ad inviarci foto, impressioni e magari una recensione. Ovviamente l'invito e' aperto a tutti gli hardware supportati, non solo all'Amstrad CPC.

SymbOS puo' essere scaricato da: http://www.symbos.de/

Provatelo, non ve ne pentirete!

Quando le collisioni degli sprite non collidono

di Attilio Capuozzo Fondatore RetroProgramming Italia – RP Italia

In merito alle Collisioni riguardanti gli Sprite, argomento trattato, peraltro, nella 3° puntata del Tutorial di Felice Nardella su come creare un Gioco in Full BASIC V2, è doveroso osservare che anche questi sono eventi che generano una richiesta di Interrupt Hardware di tipo IRQ come già visto precedentemente quando abbiamo accennato alle sorgenti di Interrupt del chip specializzato CIA #1.

Il chip grafico VIC-II, la GPU del C64, ha 4 sorgenti di richieste di Interrupt (Vedi FIG. 1) tra cui figurano, appunto, la Sprite to Sprite Collision e la Sprite to Data Collision.

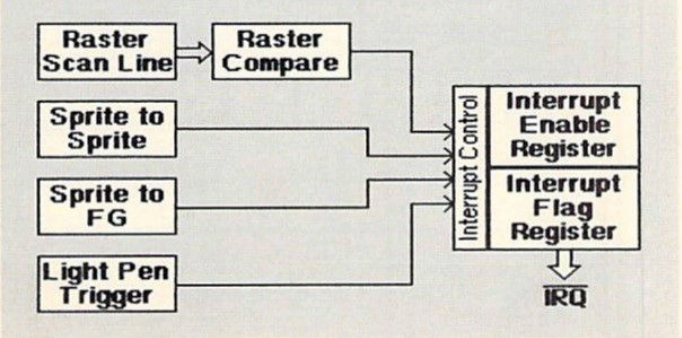


Fig. 1 - Sorgenti di Interrupt

Similmente ai 2 chip CIA, le sorgenti di Interrupt provenienti dal VIC-II sono controllate da 2 Registri Interni (Vedi FIG. 2): l'IRQ Mask Register (IRQMASK) mappato all'indirizzo 53274 ($D01A) e il VIC Interrupt Flag Register (VICIRQ) mappato all'indirizzo 53273 ($D019).

L'Interrupt Flag Register segnala quando ha origine un Interrupt dal VIC-II e indica quale sorgente ha generato la richiesta di Interrupt.

Risulteranno settati a 1 il bit 7 e i corrispondenti bit responsabili della richiesta di Interrupt.

A seguito del verificarsi di una richiesta di Interrupt, nell'Interrupt Flag Register risulterà settato un "latch" (una sorta di "blocco") in modo che non potranno più avere origine ulteriori richieste di Interrupt dalla medesima sorgente fino a che il latch non sarà azzerato scrivendo 1 nello specifico bit relativo, appunto, alla sorgente dell'Interrupt.

L'IRQ Mask Register (o Interrupt Enable Register) serve invece ad abilitare/disabilitare le 4 sorgenti di richieste di Interrupt del VIC-II.

E' necessario scrivere 1 o 0 in uno dei primi 4 bit del Registro 53274 ($D01A) per, rispettivamente, abilitare o disabilitare una delle sorgenti di Interrupt.

Quando si verifica una collisione tra 2 o più Sprite o tra uno Sprite e un Carattere o una parte del Bitmapped Screen entrano in gioco altri 2 Registri (vedi FIG. 3) che sono rispettivamente lo Sprite to Sprite Collision Register (SPSPCL) mappato all'indirizzo 53278 ($D01E) e lo Sprite

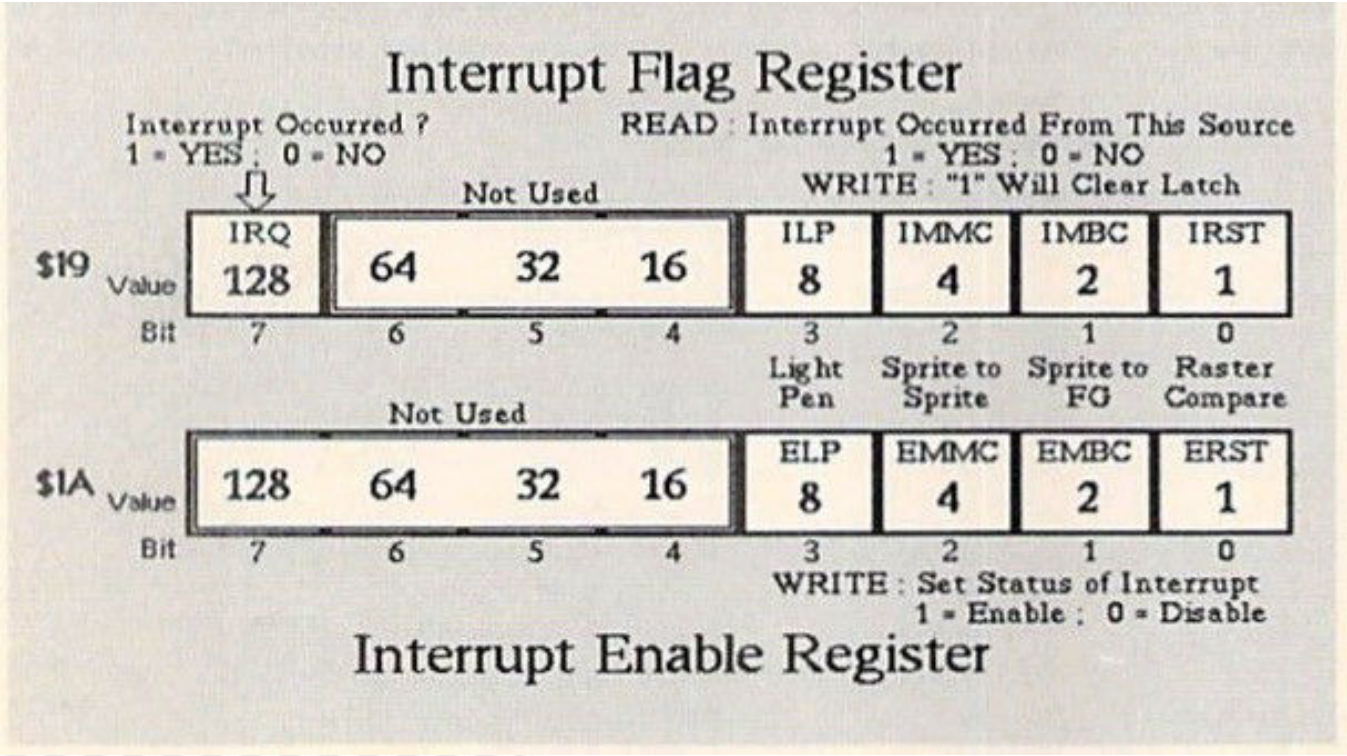


Fig. 2 - Interrupt Flag Register

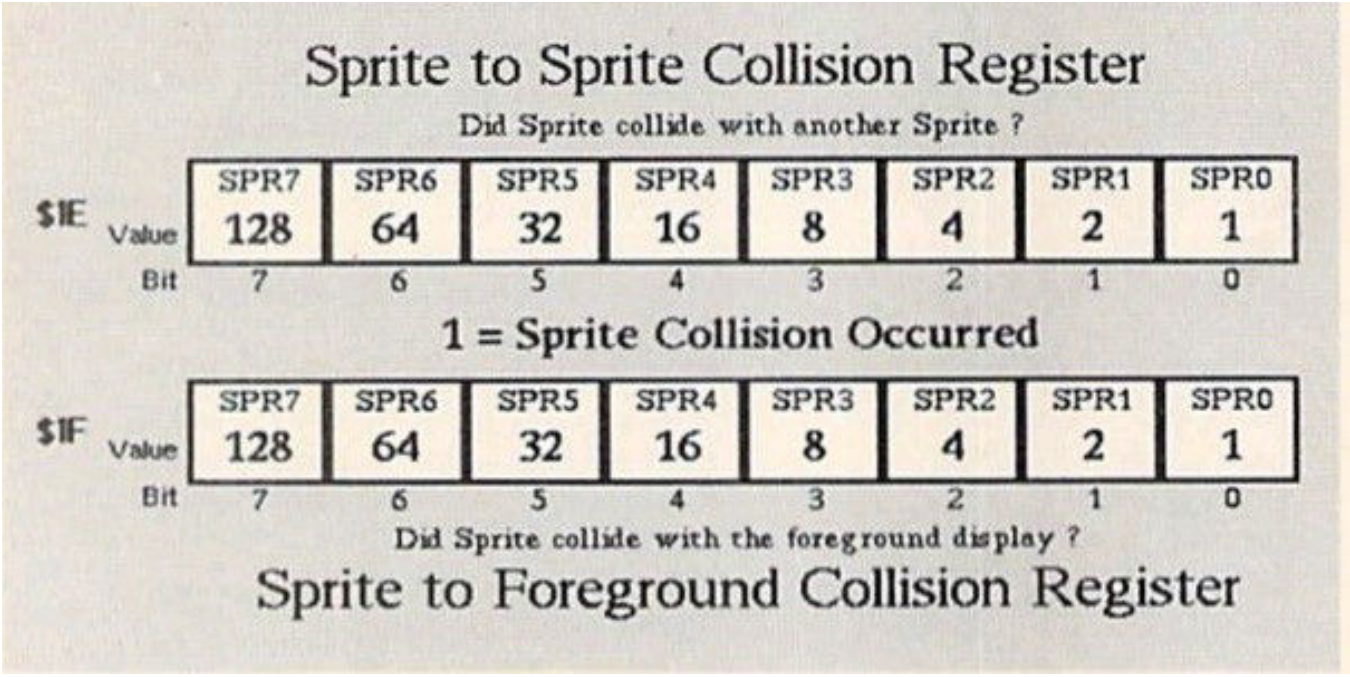


Fig. 3 - Sprite to Sprite Collision Register

to Data Collision Register (SPBGCL) mappato all'indirizzo 53279 ($D01F), quest'ultimo, nella letteratura tecnica, viene denominato anche Sprite to Foreground Collision Register o ancora Sprite to Background Collision Register ma noi preferiamo usare la prevalente denominazione proposta dalla C64 PROGRAMMER'S REFERENCE GUIDE che riteniamo essere più attinente alla natura della Collisione presa in esame e meno soggetta ad ambiguità interpretative da parte del lettore (come detto, Sprite to Data Collision Register).

Ogni singolo bit di entrambi i Registri di Collisione corrisponde a uno degli 8 possibili Sprite.

Tramite la lettura di questi Registri è possibile, dunque, verificare quali Sprite sono stati coinvolti in una Collisione in quanto i rispettivi bit risulteranno settati a 1.

La lettura dei Registri ne determinerà l'azzeramento del contenuto.

In merito alla Collisione è fondamentale rilevare che la stessa si verificherà solo quando la posizione di un pixel non trasparente dell'area rettangolare di uno Sprite (24x21 pixel nel caso di un Normal Size Sprite) o coinciderà con la posizione di un pixel non trasparente di un altro Sprite oppure quando incontrerà un pixel di Foreground di un elemento grafico dello Schermo (Character Block nel Text Mode o Cell nel Bitmap Mode).

Nel Modo Colore Standard dello Schermo, il pixel non trasparente (Foreground) è uguale a 1 mentre il pixel di Background equivale a 0.

Nel MultiColor Mode (MCM) i pixel di Foreground corrispondono alle combinazioni di bit "10" e "11" mentre tra quelli di Background è compreso, eccezionalmente, anche il bit pair "01" oltre alla combinazione di bit "00". Pertanto bisogna tenere bene a mente che il bit pair "01" NON genera Collisione.

Ricordiamo, per comodità del lettore, che il colore corrispondente alla coppia di bit "01" in uno Sprite Multicolor è settato dallo Sprite Multicolor Register 0 (SPMC0) mappato all'indirizzo 53285 ($D025).

Nel Multicolor Character Mode, invece, il colore può essere scelto tramite il Background Color 1 Register (BGCOL1) mappato all'indirizzo 53282 ($D022) e infine nel caso del Multicolor Bitmap Mode, il colore della coppia di bit "01" può essere impostato tramite il nibble alto della locazione della Screen Memory corrispondente alla Cell della Bitmap.

That's all folks!

Vi ricordo che potete raggiungere il gruppo **RetroProgramming Italia - RP Italia**:
https://www.facebook.com/groups/retroprogramming/

May the FORTH be with us - prima parte

di Francesco Fiorentini

Incuriosito dall'articolo di **Michel Jean** (vedi RMW 25 ITA e RMW 3 ENG) e memore di una chiacchierata con **Ermanno Betori** avvenuta qualche tempo fa, dove avevamo discusso della possibilitá di imparare il linguaggio FORTH, ho deciso di darmi una chance e cominciare a studiare questo inconsueto linguaggio.

Premessa

Premetto che attualmente non conosco il Forth e quindi non é mia pretesa insegnare questo linguaggio, quanto piuttosto creare una serie di articoli con alcune considerazioni personali che potrebbero tornare utili a chi deciderá di seguire questo stesso percorso.

Sul web si trovano decine di manuali, guide ed esempi di codice per qualsivoglia linguaggio ed il Forth non fa ovviamente eccezione; personalmente peró ho voluto arricchire la mia collezione cartacea acquistando il libro **FORTH Programming** di **Steven Vickers**. Questo libro é una ristampa del manuale del Jupiter ACE in occasione del 35simo anniversario (nel 2017) del lancio di questo computer avvenuto appunto nel 1982.

Gli esempi che proporró saranno quindi compatibili con la versione Forth del Jupiter ACE, anche se in teoria dovrebbe essere possibile eseguirli su altre macchine, il TI99/4A per esempio che ha diverse implementazioni di Forth, alcune delle quali veramente ricche di comandi.

Il Dizionario del Forth

Come abbiamo potuto leggere nell'ottimo articolo di Michel Jean, Forth é un linguaggio di programmazione piuttosto particolare. Cominciamo quindi a vedere insieme alcune di queste particolaritá. La prima e forse anche la piú evidente é il suo dizionario.

Provate ad eseguire questo comando:

```
vlist
```

Se lo avete eseguito sull'emulatore del Jupiter ACE, o meglio ancora su real hardware se ne siete tra i fortunati possessori, dovreste vedere il risultato di [Fig. 1].

Questa collezione di comandi é conosciuta come il Vocabolario del Forth. L'idea alla base del Forth é quella di espandere questa lista creando nuovi comandi a partire da quelli esistenti. Una volta che questi comandi aggiuntivi sono stati creati, andranno ad arricchire quello che viene identificato come il Dizionario del Forth.

Ovviamente questi nuovi comandi possono essere utilizzati a loro volta per crearne di nuovi. Potete quindi facilmente capire la potenzialitá del Forth rispetto ad altri linguaggi di programmazione.

Ma vediamo velocemente un esempio per capire quanto semplice sia arricchire il linguaggio di base con nuove funzionalita'.

Digitate quindi il comando:

```
cls
```

per ripulire lo schermo e poi digitate il seguente listato:

```
: hello
CR ." hello world"
;
```

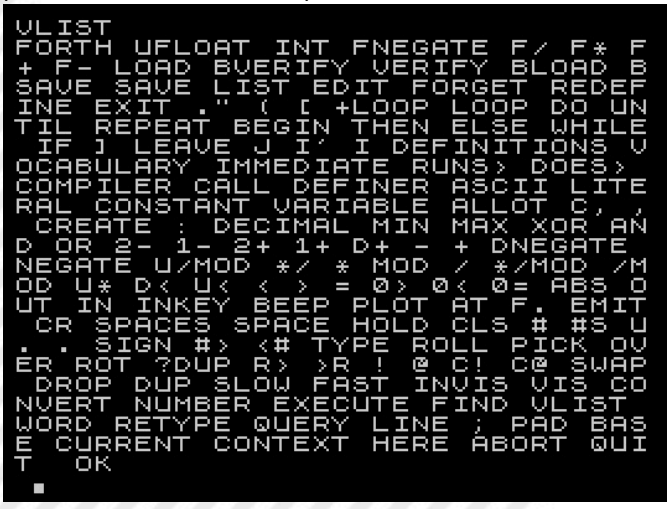


Fig. 1 - Il dizionario Forth del Jupiter ACE

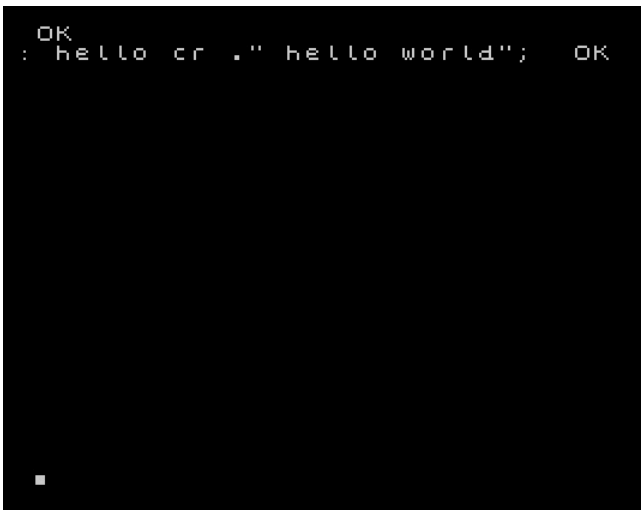


Fig. 2 - Il programma HELLO

Per il momento non preoccupiamoci dellle istruzioni, che analizzeremo in seguito, quanto del risultato. Fate attenzione a rispettare tutti gli spazi, anche quello tra il primo doppio apice e l'h di hello, altrimenti riceverete quasi sicuramente un errore.

Se avete fatto tutto correttamente dovreste trovarvi di fronte ad una schermata come quella in [Fig. 2]

Proviamo pure ad eseguire il nostro programmino digitando:

hello

Il risultato non é certo esaltante, ma abbiamo creato il nostro primo Hello World in Forth con successo:

```
 OK
: hello cr ." hello world";  OK

hello
hello world OK
```

Proviamo quindi ad eseguire nuovamente il comando VLIST. Come possiamo notare abbiamo creato un nuovo comando Forth che é stato aggiunto alla lista di quelli nativamente disponibili. Abbiamo quindi contribuito ad arricchire il Dizionario del nostro Forth.

```
vlist
HELLO FORTH UFLOAT INT FNEGATE F
/ F* F+ F- LOAD BVERIFY VERIFY B
LOAD BSAVE SAVE LIST EDIT FORGET
 REDEFINE EXIT ." ( [ +LOOP LOOP
 DO UNTIL REPEAT BEGIN THEN ELSE
 WHILE IF ] LEAVE J I' I DEFINIT
IONS VOCABULARY IMMEDIATE RUNS>
DOES> COMPILER CALL DEFINER ASCI
I LITERAL CONSTANT VARIABLE ALLO
T C, , CREATE : DECIMAL MIN MAX
XOR AND OR 2- 1- 2+ 1+ D+ - + DN
EGATE NEGATE U/MOD */ * MOD / */
MOD /MOD U* D< U< < > = 0> 0< 0=
 ABS OUT IN INKEY BEEP PLOT AT F
. EMIT CR SPACES SPACE HOLD CLS
# #S U. . SIGN #> <# TYPE ROLL P
ICK OVER ROT ?DUP R> >R ! @ C! C
@ SWAP DROP DUP SLOW FAST INVIS
VIS CONVERT NUMBER EXECUTE FIND
VLIST WORD RETYPE QUERY LINE ; P
AD BASE CURRENT CONTEXT HERE ABO
RT QUIT  OK
 ■
```

Ma non é certo finita qua. Come dicevamo il nostro comando HELLO é adesso parte integrante del dizionario del Forth e puó quindi essere usato per creare nuovi comandi.

Digitate il seguente listato:

```
: clshello
cls
hello
;
```

Se avete eseguito il tutto correttamente dovreste ricevere il messaggio OK alla fine. Provate quindi ad eseguire il nuovo comando: **clshello**

Esatto! Pulisce lo schermo e poi stampa a video il messaggio Hello World.

```
hello world OK

 ■
```

Anche in questo caso, se ripetiamo il comando VLIST, vedremo il nostro comando CLSHELLO all'inizio della lista dei comandi, seguito dal precedente HELLO.

I comandi : e ;

Come avrete probabilmente giá intuito, il comando : (due punti) serve per dichiarare la definizione di una nuova parola del Dizionario del Forth.

Il comando ; (punto e virgola) invece serve per comunicare al Forth che la definizione é terminata.

E se volessimo rivedere e correggere i nostri listati come possiamo fare?

I comandi LIST, EDIT e FORGET

Possiamo usare rispettivamente il comando LIST:

list hello

```
 OK
list hello
: HELLO
 CR ." hello world"
;
 OK
list clshello
: CLSHELLO
 CLS HELLO
;
 OK

 ■
```

ed il comando EDIT:

```
edit clshello
```

```
 OK
list hello
: HELLO
 CR ." hello world"
;
 OK
list clshello
: CLSHELLO
 CLS HELLO
;
 OK
edit clshello




 ?: CLSHELLO
  CLS HELLO
 ;
```

E se volessimo invece eliminare uno dei comandi appena creati? Niente di piú facile, basterá semplicementre dire a Forth di dimenticare il suddetto comando:

```
forget hello
```

```
forget hello vlist
FORTH UFLOAT INT FNEGATE F/ F* F
+ F- LOAD BVERIFY VERIFY BLOAD B
SAVE SAVE LIST EDIT FORGET REDEF
INE EXIT ." ( [ +LOOP LOOP DO UN
TIL REPEAT BEGIN THEN ELSE WHILE
 IF ] LEAVE J I' I DEFINITIONS V
OCABULARY IMMEDIATE RUNS> DOES>
COMPILER CALL DEFINER ASCII LITE
RAL CONSTANT VARIABLE ALLOT C, ,
 CREATE : DECIMAL MIN MAX XOR AN
D OR 2- 1- 2+ 1+ D+ - + DNEGATE
NEGATE U/MOD */ * MOD / */MOD /M
OD U* D< U< < > = 0> 0< 0= ABS O
UT IN INKEY BEEP PLOT AT F. EMIT
 CR SPACES SPACE HOLD CLS # #S U
. . SIGN #> <# TYPE ROLL PICK OV
ER ROT ?DUP R> >R ! @ C! C@ SWAP
 DROP DUP SLOW FAST INVIS VIS CO
NVERT NUMBER EXECUTE FIND VLIST
WORD RETYPE QUERY LINE ; PAD BAS
E CURRENT CONTEXT HERE ABORT QUI
T  OK
 ■
```

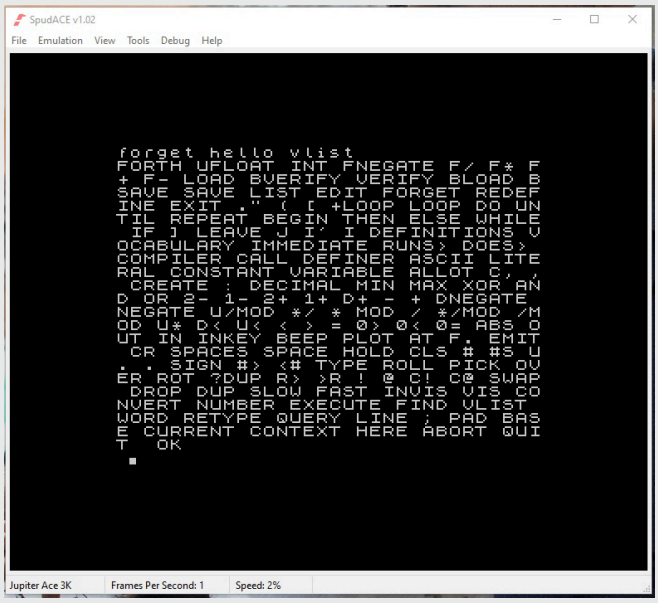
Da notare come il Forth sia cosí intelligente da dimenticare di conseguenza anche il comando CLSHELLO che era un derivato del comando HELLO.

Bene, siamo giunti al termine di questo nostro primo diario dedicato al FORTH. Mi auguro di aver stuzzicato un po' la vostra curiositá e che, come me, abbiate voglia di scoprire le peculiaritá di questo linguaggio che, soprattutto in Italia, é passato un po' in sordina negli anni d'oro degli 8 bit.

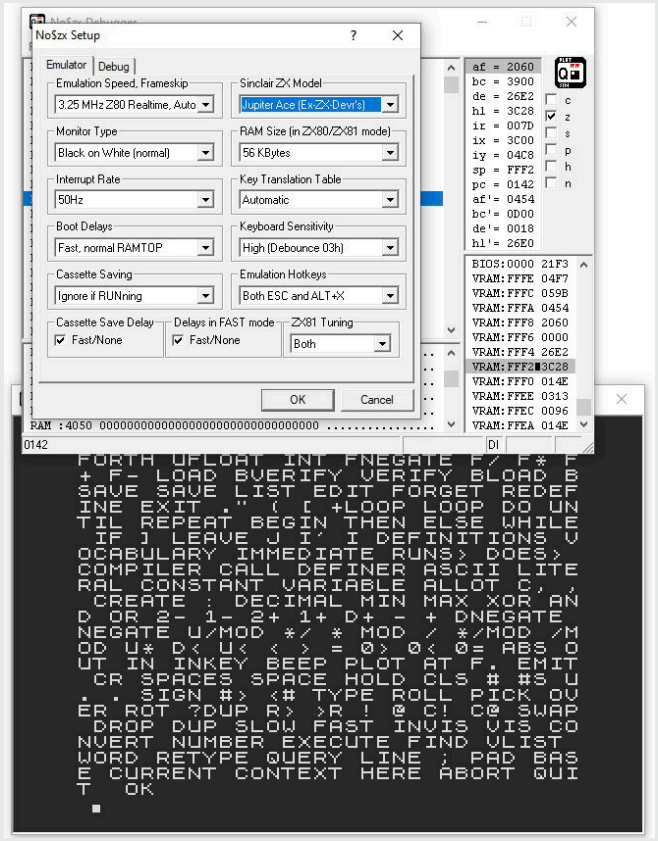
Prima di lasciarvi al box sull'emulatore del Jupiter ACE, voglio peró augurare a tutti Buone Feste.

Emulare il Jupiter ACE

Per emulare il Jupiter ACE, ed eseguire gli esempi illustrati, potete utilizzare l'emulatore **SpudACE**.



Oppure l'emulatore **no$zx**, avendo peró l'accortezza di scegliere il Jupiter ACE come **ZX Model** da emulare nel menú **Options** -> **Emulation Setup**.



Entrambi gli emulatori possono essere scaricati da: http://www.jupiter-ace.co.uk/emulators_win.html

Life: il gioco della vita

di Marco Pistorio

Vicini alla chiusura di questo numero di RetroMagazine World ci siamo resi conto che, rispetto ai numeri precedenti, mancava qualche riga di codice.

Per rimediare a questa mancanza abbiamo contattato Marco Pistorio che in men che non si dica ci ha fornito, non uno, ma ben due programmi in Basic V2 pronti da impaginare. La nostra scelta é caduta su questo algoritmo di LIFE, il gioco della vita.

Ai lettori il compito di analizzare il codice ed eventualmente modificarlo per aumentare il numero di generazioni.

Il Gioco della vita (Game of Life in inglese, noto anche solo come Life) è un automa cellulare sviluppato dal matematico inglese John Conway sul finire degli anni sessanta. Il Gioco della vita è l'esempio più famoso di automa cellulare: il suo scopo è quello di mostrare come comportamenti simili alla vita possano emergere da regole semplici e interazioni a molti corpi, principio che è alla base dell'ecobiologia, la quale si rifà anche alla teoria della complessità.
Da Wikipedia: https://it.wikipedia.org/wiki/Gioco_della_vita

```
100 rem gioco della vita
101 poke53280,0:poke53281,0:poke646,5
102 printchr$(147);
105 :
106 rem dimensiona matrici
107 dim x$(9,9),y$(9,9)
109 :
110 fd=1:gosub 4000
115 :
130 x$(4,4)="*":x$(4,5)="*":x$(4,6)="*"
133 x$(5,4)="*":x$(5,6)="*"
135 x$(6,4)="*":x$(6,5)="*":x$(6,6)="*"
199 :
200 for gn=1 to 7
202 :
205 gosub 3000
299 :
300 rem calcolo generaz. successiva
310 for j=2 to 8
315 :
320 for k=2 to 8
321 :
325 gosub 900
326 :
327 cr$=left$(x$(j,k),1)
330 if cr$=" " then 400
331 :
335 if sv<=1 then y$(j,k)=" "
336 if sv>=4 then y$(j,k)=" "
337 if sv>=2 and sv<=3 then y$(j,k)="*"
350 goto 500
399 :
400 if sv=3 then y$(j,k)="*"
499 :
500 next k,j
600 gosub 2000
610 fd=2:gosub 4000
699 :
700 next gn
800 end
899 :
900 rem controllo vicinanze cella
901 sv=0
902 if x$(j-1,k-1)="*" then sv=sv+1
903 if x$(j-1,k)  ="*" then sv=sv+1
904 if x$(j-1,k+1)="*" then sv=sv+1
905 if x$(j,k-1)  ="*" then sv=sv+1
906 if x$(j,k+1)  ="*" then sv=sv+1
907 if x$(j+1,k-1)="*" then sv=sv+1
908 if x$(j+1,k)  ="*" then sv=sv+1
909 if x$(j+1,k+1)="*" then sv=sv+1
910 return
999 :
2000 rem --- copia matrici ---
2001 for j=1 to 9
2002 for k=1 to 9
2003 x$(j,k)=y$(j,k)
2004 next k,j
2005 return
2999 :
3000 rem --- disegna matrice 9*9 ---
3001 :
3002 print"generazione:";gn
3005 :
3010 for j=1 to 9
3020 for k=1 to 9
3025 :
3030 print mid$(x$(j,k),1,1);
3035 :
3040 next k
3050 :
3060 print
3070 next j
3080 return
3999 :
4000 rem --- riempi matrici ---
4001 for j=1 to 9
4002 for k=1 to 9
4003 :
4004 if fd=1 then y$(j,k)=" ":x$(j,k)=" "
4005 if fd=2 then y$(j,k)=" "
4006 :
4007 next k,j
4008 return
```

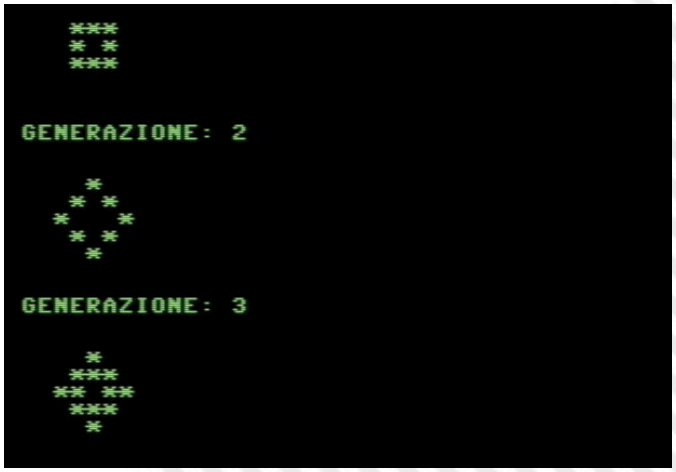


Fig. 1 - Output del programma

PETSCII BBS DAL BROWSER INTERNET

di Antonino Porcino

Nel numero 23 di RM Francesco Sblendorio ci ha illustrato il suo progetto "PETSCII-BBS Builder", un framework Java con cui è possibile costruire una BBS per i computer Commodore fruibile via internet piuttosto che attraverso la linea telefonica analogica come avveniva una volta.

Come è stato detto nell'intervista, il progetto ha avuto un discreto successo e il framework è stato utilizzato per creare altre BBS oltre a quella realizzata dallo stesso Francesco che attualmente gira all'indirizzo bbs.sblendorio.eu:6510.

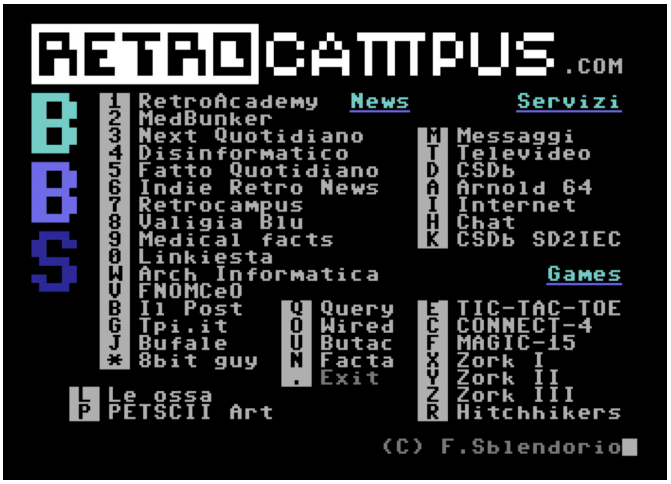


Fig. 1 - Il menu principale della BBS

Per collegare un Commodore 64 alla BBS è necessario un apposito modem da inserire nella user port, come ad esempio la scheda "KC64Wifi" di Pasquale De Luna [1]. Purtroppo però non molti possessori di C64 dispongono di un modem di questo tipo e ciò limita un po' la fruizione della stessa BBS da parte della maggioranza degli utenti.
Ragionando su questo problema era da un po' di tempo che mi frullava in testa l'idea di mettere in piedi un emulatore C64 e farlo collegare in qualche modo alla BBS di Francesco Sblendorio. In realtà questa cosa già la si potrebbe fare con il buon vecchio VICE, dopo averlo configurato correttamente; la cosa però non è proprio semplice per gli utenti non esperti. La mia idea invece, era di rendere disponibile un emulatore online richiamabile direttamente dal browser internet, che fosse quindi pronto per l'uso senza nessuna installazione o configurazione, permettendo così l'accesso alla BBS anche da parte dell'utente occasionale che volesse giusto curiosare un po'.
L'idea dell'emulatore nel browser mi ha sempre affascinato, per la facilità di accesso che ne deriva, ma anche perché usando il linguaggio JavaScript l'emulatore diventa modificabile a proprio piacimento aprendo infinite possibilità. Ad esempio è possibile ispezionare o modificare le locazioni di memoria "a cuore aperto", mentre cioè l'emulatore sta girando senza interrompere l'esecuzione.

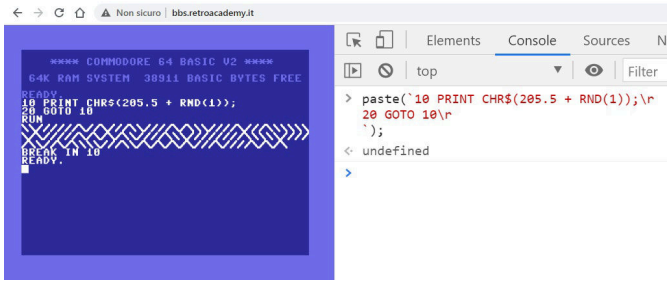


Fig. 2 - Esempio di interazione tra JavaScript e emulatore: JavaScript inietta codice BASIC nel C64 mediante la funzione "paste()"

Nel corso degli ultimi anni ho creato vari emulatori che girano nel browser (Laser 500, LM80C, ChildZ, General Processor e VIC20), principalmente per uso personale ad esempio per fare il debug dei miei programmi cross-compilati oppure per il reverse-engineering di alcuni dei computer di cui mi sono occupato.
Per la BBS però volevo non partire da zero, sia per il fatto che scrivere un emulatore di C64 è abbastanza complesso a causa dei suoi chip custom e sia perché il Commodore 64 è per fortuna uno dei computer più emulati che ci siano; quindi non aveva senso reinventare la ruota, era più conveniente guardarsi un po' in giro nel mondo open-source per vedere cosa ci fosse disponibile.

Il progetto "chips"

Dopo aver vagliato varie ipotesi, la mia scelta è andata sull'affascinante progetto "chips" di Andre Weissflog, che potete vedere in funzione sul sito "Tiny 8 bit Emulators" [2]. L'idea che sta dietro al progetto è veramente innovativa: piuttosto che realizzare un classico emulatore, l'autore ha pensato di simulare separatamente i singoli chip che compongono un computer sotto forma di files in linguaggio C, per poi collegarli insieme in maniera "virtuale". In questo modo si possono ricreare sistemi diversi semplicemente cambiando i chip di cui il sistema è composto.
L'emulazione avviene con precisione al singolo ciclo di clock (cycle-exact) e vengono emulati tutti i pin come nei

chip reali. L'implementazione è molto efficiente ed è tutta scritta in linguaggio C standard; ogni chip risiede in un file ".h" senza bisogno di altre dipendenze e può essere incluso nel proprio progetto, infatti è così che ho emulato lo Z80, il TMS-9929 e l'AY-3-8910 del computer LM80C [3].
La trovata geniale di Andre Weissflog è stata quella di collegare i chip tra di loro con un bus simulato a 64 pin; con le moderne architetture a 64 bit, questo bus può tranquillamente risiedere in una word della CPU ed essere scambiato nelle funzioni C all'interno di un normale registro del processore, garantendo così il massimo dell'efficienza. Ad esempio le linee del bus indirizzi A0-A15 risiedono nei bit 0-15 della word, mentre gli 8 bit del bus dati nei bit da 16 a 23. Gli altri segnali di controllo come RW, IRQ, NMI, ecc... risiedono nei bit successivi. Scrivere e leggere da questa word a 64 bit è un'operazione molto veloce poiché le moderne CPU possono spostare dati ed eseguire operazioni di shift nello stesso ciclo di clock.
Per usare uno di questi chip emulati, basta eseguire la relativa funzione denominata "tick()" che equivale a far avanzare il chip di un ciclo di clock. In questo step, vengono aggiornati i registri interni e i piedini di I/O così come farebbe un chip reale.

Nel progetto di Andre Weissflog sono stati implementati molti dei circuiti integrati dei computer a 8 bit (Z80, 6502, ecc). Trattandosi però di un progetto portato avanti da una singola persona, l'emulazione non è sempre perfetta e ci sono delle zone incomplete che l'autore non ha purtroppo avuto il tempo di realizzare. Si resta comunque stupiti di fronte all'enorme mole di lavoro svolto e sfogliare i sorgenti del progetto è un esercizio estremamente istruttivo per l'appassionato di retrocomputing.
Oltre ai singoli chip, nel progetto sono stati realizzati dei sistemi pronti per l'uso, con i circuiti già connessi che emulano quindi un computer completo (VIC20, C64, ZX Spectrum, Amstrad CPC, ecc...); ad esempio l'emulatore C64 risiede nel file "c64.h" al quale bisogna solamente collegare tastiera, audio e video -- che io nel mio progetto ho collegato al browser internet.

Vi è però un problema: il browser parla JavaScript mentre "chips" è scritto in C. Come far dialogare i due? E qui arriva in aiuto una recente tecnologia HTML chiamata WebAssembly. Si tratta essenzialmente di una macchina virtuale standardizzata che gira in una sandbox dentro il browser e che è in grado di eseguire in maniera molto efficiente il suo byte code. Tale byte code non è legato ad una specifica architettura di CPU, ma è generico senza nemmeno i registri (usa solamente lo stack); inoltre, per come è strutturato, si traduce molte bene in linguaggio macchina e le prestazioni sono vicine all'esecuzione del codice nativo. Il byte code ovviamente non viene scritto a mano, ma viene generato dai vari compilatori, nel mio caso da "emscripten" [10] il compilatore C/C++ nato specificatamente per il WebAssembly.

Con WebAssembly possiamo quindi eseguire dei programmi C all'interno del browser, anche un intero emulatore C64. A quest'ultimo vanno "collegati" tastiera, video e audio del browser attraverso l'apposito codice JavaScript. Lo schermo non è altro che un normale canvas HTML, mentre per il suono vi sono le "audio API" del browser, che sostanzialmente riproducono sugli altoparlanti il buffer di samples che l'emulatore fornisce ad intervalli regolari. Avendo in precedenza già scritto un emulatore per VIC-20 proprio con il progetto di Andre Weissflog, ne ho semplicemente riadattato il codice per il C64, sfruttando la similitudine tra questi due computer. In poco tempo ho tirato su un primo prototipo funzionante con il classico prompt "READY" e i 38911 bytes a disposizione dell'utente. Rispetto al progetto originale, ho apportato delle piccole modifiche, come ad esempio l'aggiunta del tasto RESTORE (che diventa "PGUP" sulla tastiera del PC) che non era implementato. Per far questo, dopo aver consultato lo schema elettrico del C64, ho visto che la linea del tasto RESTORE dalla matrice della tastiera va direttamente al pin NMI della CPU 6510, quindi è bastato aggiungere nel codice dell'emulatore le seguenti linee:

```
static uint64_t _c64_tick(c64_t* sys, uint64_t pins) {

    // RESTORE key
    if(sys->kbd.scanout_column_masks[8] & 1) {
        pins |= M6502_NMI;  /* RESTORE key is pressed */
    }
```

Dove "pins" è la word a 64 bit descritta pocanzi che contiene tutto il bus. Come si vede, per accendere il pin NMI è stato sufficiente fare un OR binario mediante l'operatore pipe "|" del C.

Il modem virtuale

Messo in piedi l'emulatore, come procedere per la BBS? La mia prima idea è stata quella di simulare uno dei vari modem disponibili al fine di utilizzare i programmi di comunicazione già esistenti. Ma mi sono reso subito conto dell'enorme sforzo richiesto, così ho optato per un'altra soluzione più sbrigativa: implementare un mio modem

virtuale. Alla fine non era richiesto un gran che: il modem doveva semplicemente ricevere ed inviare dei caratteri.

Poiché sul C64 non ci sono le porte di I/O (a differenza ad esempio dello Z80), il mio modem virtuale doveva essere mappato in memoria, come del resto già avviene per tutte le periferiche nel C64.

Nella funzione "c64_tick()" mi sono andato a trovare dove viene gestito l'I/O mappato in memoria e ho sottratto al SID lo spazio da $D7F0 a $D7FF riservandolo per la mia periferica virtuale:

```
else if (addr < 0xD7F0) {   // was D800
    /* SID (D400..D7FF) */
    sid_pins |= M6581_CS;
}
else if (addr < 0xD800) {
    /* MODEM (D7F0..D7FF) */
    modem_access = true;
}
else if (addr < 0xDC00) {
    /* read or write the special color Static-RAM bank (D800..DBFF) */
    color_ram_access = true;
}
```

Ad ogni lettura in questo range faccio eseguire la funzione JavaScript "modem_read()" e ad ogni scrittura richiamo "modem_write()", così da intercettare gli accessi al modem comodamente dal lato JavaScript:

```
else if (modem_access) {
    if (pins & M6502_RW) {
        /* modem read: chiama la funzione JavaScrip "modem_read(data)" */
        byte data = (byte) EM_ASM_INT({ return modem_read($0); }, addr);
        M6502_SET_DATA(pins, data);
    }
    else {
        /* modem write: chiama la funzione JavaScript "modem_write(addr,data) */
        uint8_t data = M6502_GET_DATA(pins);
        byte unused = (byte) EM_ASM_INT({ modem_write($0,$1); }, addr, data );
    }
}
```

Il mio modem virtuale è così strutturato:

```
// porte del modem virtuale
dim MODEM_DATA_OUT    as byte at $D7F2    ; caratteri da inviare
dim MODEM_DATA_IN     as byte at $D7F0    ; caratteri in arrivo
dim MODEM_DATA_REQ    as byte at $D7F3    ; 1 se ci sono caratteri in arrivo nel buffer
dim MODEM_ACK         as byte at $D7F1    ; porta per l'handshake
dim MODEM_CONNST      as byte at $D7F4    ; status del modem (0=connesso, <>0 disconnesso)
```

DATA_IN e DATA_OUT sono semplicemente le locazioni dove sono trasmessi e ricevuti i caratteri dal modem. DATA_REQ è un flag che indica se ci sono caratteri nel buffer di ricezione. CONNST ritorna lo stato connesso/non connesso alla BBS: lo uso per far cambiare il colore allo schermo da nero (connesso) a rosso (non connesso).

Per quanto riguarda ACK, questo permette una sorta di handshake con il modem per lo scambio dei caratteri in ricezione. Inizialmente non ci doveva essere alcun handshake perché i dati dal modem alla CPU dovevano transitare nel singolo ciclo di clock di lettura dell'I/O. Poi però questo sistema non si è dimostrato affidabile, talvolta alcuni caratteri venivano inspiegabilmente persi. Non ne ho capito esattamente il motivo, ma piuttosto che perdermi nei meandri di un infinito troubleshooting (considerato che c'è di mezzo un emulatore non molto testato), ho preferito risolvere velocemente con un brutale meccanismo di handshake: dopo che la CPU legge il byte in arrivo, questa manda sulla porta ACK prima il valore "0" e subito dopo "1"; in questo modo la transizione da 0 a 1 indica che il carattere è stato acquisito dalla CPU e il modem può procedere con il successivo.

Il programma terminale

```
; stampa il carattere a video
jsr CHROUT

; notifica al modem che il carattere è stato ricevuto
lda #0
sta MODEM_ACK
lda #1
sta MODEM_ACK
```

Fig. 3 - Il programma terminale fa handshake con il modem virtuale inviando i bytes "0" e "1" sulla porta ACK

Messo a punto il modem virtuale, mi serviva un programma di comunicazione da far girare sul C64 emulato, giacché nessuno di quelli esistenti poteva essere compatibile. Per leggere e stampare i caratteri a video, Francesco Sblendorio mi ha suggerito di utilizzare semplicemente le routine CHROUT ($FFD2) e GETIN ($FFE4) del kernal del C64, così come già avviene nel suo software per la scheda "Ultimate 64". Ho scritto quindi un piccolo programma assembly che essenzialmente fa un loop infinito tra lettura dei caratteri in arrivo e trasmissione al modem di quelli digitati da tastiera. Piuttosto che usare l'assembly nudo e crudo, ho utilizzato un macro linguaggio di mia invenzione che ormai da anni mi accompagna in tutti i miei progetti per 6502. Questo strumento (che ho chiamato "Asmproc" [4]) mi permette di scrivere un assembly leggibile mediante dei costrutti tipo IF-THEN, DO-LOOP ecc... evitando il ricorso ai classici branch-di-qua e branch-di-là che nel 6502 offuscano completamente la comprensibilità del codice.

Inizialmente il programma terminale della BBS era veramente

```
cursor_off:
    ldy $cc
    if zero then
        ldy #$01
        sty $cd
        do
            ldy $cf
        loop while not zero
    end if
    ldy #$ff
    sty $cc
    rts
```

Fig. 4 - esempio della programmazione assembly strutturata con Asmproc: l'implementazione della routine "cursor_off"; come si vede non ci sono né label né BEQ/BNE

una manciata di bytes, poi a poco a poco è stato ampliato fino ad arrivare a ben 165 bytes, con l'aggiunta del cursore lampeggiante, della campana del CHR$(7) e di altre piccole minuzie. Se vi incuriosisce, trovate il sorgente completo in fondo all'articolo.

Per il mio scopo della BBS, il programma terminale doveva partire in automatico all'avvio dell'emulatore; ma poiché sul C64, la memoria viene azzerata al boot, era necessario "iniettare" il programma in RAM solo dopo la fine dello stesso boot. Un trucco che ho escogitato è quello di stare in ascolto della locazione 204, che è quella che stabilisce se il cursore è presente sullo schermo oppure è spento; quando il cursore è acceso, vuol dire che il "READY" è apparso sullo schermo e quindi si può iniettare il programma in RAM per poi subito dopo infilare i caratteri "RUN" + RETURN nel buffer di tastiera, causandone l'esecuzione.

Il tunnel WebSocket

Finito il programma sul C64 che manda caratteri al modem virtuale, come fanno questi da qui ad arrivare alla BBS? In un mondo normale questo avverrebbe con una banale connessione TCP sulla porta 6510 che è quella dove gira la BBS. Ma invece no, c'è un ostacolo: per motivi di sicurezza informatica il browser non può aprire nessuna connessione TCP con il mondo esterno.

Per sopperire a tale mancanza negli ultimi anni è stato introdotto un nuovo standard chiamato WebSocket: è un protocollo con cui si possono veicolare delle connessioni simili ai socket TCP, incanalandole attraverso la normale porta 80 del servizio HTTP.

Però la BBS non accetta connessioni WebSocket, ma solo quelle TCP. Cosa fare dunque? L'idea che ho avuto è stata quella di realizzare un'apposita utility da far girare separatamente che accetti connessioni WebSocket dall'emulatore/browser e le inoltri verso la normale porta TCP della BBS. E' quello che in gergo si chiama tunnelling. Ho scritto quindi un'applicazione in JavaScript lato server, vale a dire in Node.js. Con molta fantasia l'ho chiamata "websocket-to-tcp"; come tutto il resto è open source e la potete trovare su Github [5]. L'utility può essere eseguita in locale ("localhost") lanciandola dal prompt dei comandi oppure più semplicemente si può usare quella che è stata installata direttamente sul server della BBS in modo tale che l'utente non debba eseguire alcuna installazione.
Vi è però un'ulteriore complicazione: poiché la connessione

```
$ wstcp -t bbs.sblendorio.eu -p 6510 -w 8080 -n bbs
Fri Dec 04 2020 18:40:23 GMT+0100 (GMT+01:00) Server is listening on port 8080 protocol "bbs"
Fri Dec 04 2020 18:41:17 GMT+0100 (GMT+01:00) connection accepted from https://nippur72.github.io
Fri Dec 04 2020 18:41:17 GMT+0100 (GMT+01:00) TCP connection established
Fri Dec 04 2020 18:41:17 GMT+0100 (GMT+01:00) TCP ready
Fri Dec 04 2020 18:41:40 GMT+0100 (GMT+01:00) TCP connection closed
Fri Dec 04 2020 18:41:40 GMT+0100 (GMT+01:00) peer ::1 disconnected
```

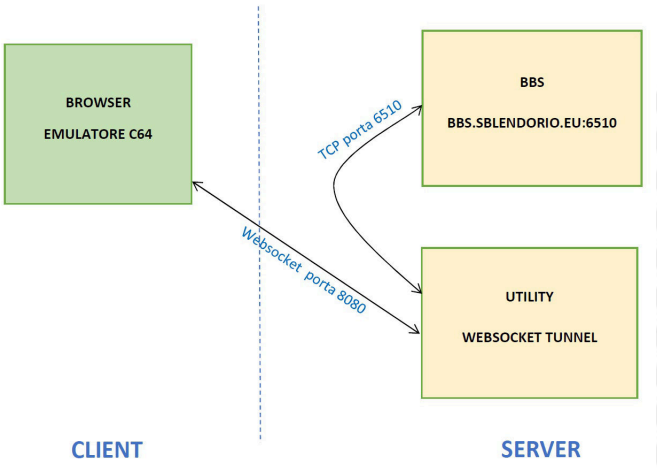
Fig. 5 - L'utility "wstcp" eseguita sul server della BBS crea un tunnel tra WebSocket e TCP

al WebSocket potrebbe originare da un emulatore che gira su un sito in HTTPS (HTTP Secure), si è reso necessario implementare anche questo tipo di connessione. Francesco Sblendorio ha quindi provveduto a certificare il sito della BBS con "letsencrypt" [6] (un'autorità che rilascia i certificati per HTTPS in maniera gratuita) e successivamente ha installato i relativi certificati sul server che adesso è in grado di stabilire connessioni criptate (le vostre chat sulla BBS sono quindi al sicuro dagli spioni!).

L'utility di tunnelling tra WebSocket e TCP è stata scritta in Node.js poiché ha il vantaggio di essere disponibile su tutti i S.O. (Window, Linux, Mac) e di avere delle librerie già pronte e facili da usare. Il suo funzionamento è molto semplice: si limita a creare un web server sulla porta specificata su cui fa transitare il traffico WebSocket restando in attesa di una connessione proveniente dal browser. A connessione avvenuta, apre una connessione TCP verso la BBS e scambia i dati tra i due. In tutto un centinaio di linee di codice.

Sul lato del browser invece, è il modem virtuale ad iniziare la comunicazione tramite WebSocket e a trasmettere alla CPU i caratteri da esso ricevuti.
La figura riassume il sistema di connessione tra browser e BBS:

Uso dell'emulatore



Non mostrato in figura vi è inoltre il Web server sulla porta 80 che fornisce la pagina HTML contenente l'emulatore quando l'utente naviga sull'indirizzo della BBS. Infatti potete aprire con un qualsiasi browser uno dei seguenti indirizzi:

http://bbs.retrocampus.com o http://retrocampus.com/bbs
http://bbs.retroacademy.it

Alternativamente è possibile accedere tramite il repository dell'emulatore su Github e mandare in esecuzione il programma del terminale con il parametro "load":
https://nippur72.github.io/c64-emu/?load=nippur72/terminal.prg

Si può anche usare il solo C64 senza BBS, omettendo "load":
https://nippur72.github.io/c64-emu
L'emulatore può anche essere incorporato all'interno di un'altra pagina web con il seguente codice HTML:

```
<iframe width="720" height="554" src="https://nippur72.github.io/c64-emu/?load=nippur72/terminal.prg"></iframe>
```

E' anche possibile collegarsi ad una BBS diversa da quella realizzata da Francesco Sblendorio. In questo caso bisognerà eseguire localmente il tunnel WebSocket, ad esempio:

```
$ wstcp -t particlesbbs.dyndns.org -p 6400 -w 8080 -n bbs
```

(apre un tunnel WebSocket verso la BBS "particlesbbs.dyndns.org:6400" tramite la porta 8080) per poi richiamare l'emulatore con il parametro "wstcp" indicando che il tunnel è attivo su localhost:8080:
https://nippur72.github.io/c64-emu/?wstcp=ws://localhost:8080&load=nippur72/terminal.prg

Consiglio di usare il browser Chrome, che è quello che risulta essere più veloce, permettendo l'esecuzione anche su computer non proprio recenti. Noterete infatti che la finestra del browser tenderà ad avere un carico sulla CPU piuttosto elevato; questo è dovuto al fatto che l'emulazione è di tipo cycle-exact, il che comporta che si devono simulare i quattro chip del C64 (CPU, VIC, SID e CIA) alla velocità di 1 MHz usando in un unico thread. Chrome è quello che riesce meglio in questa impresa, mentre tutti gli altri seguono a distanza.
Gli altri browser inoltre hanno dato problemi di incompatibilità, nello specifico con la funzione "Blob.arrayBuffer()": su Safari 13 per Mac tale funzione non è proprio implementata, mentre su Firefox 83 la stessa occasionalmente causava l'arrivo dei byte dal modem in ordine errato. Per risolvere ho applicato una patch trovata su internet che semplicemente reimplementa le funzione in JavaScript e la inserisce nella classe dell'oggetto "Blob" (essendo JavaScript un linguaggio dinamico si può fare questa cosa anche al run-time). La si trova nel file nel file "patch-arrayBuffer.js"

Considerazioni finali

In questo articolo ho illustrato come ho realizzato un emulatore per C64 che gira nel browser, dotandolo di un modem virtuale, con tanto di programma terminale, per poi farlo collegare alle BBS via internet, attraverso un tunnel WebSocket. Portare a termine questo progetto è stata una piacevole sfida che ha permesso di confrontarmi con alcune delle moderne tecnologie, applicandole al mondo del retro-computing. Spero che il risultato sia stato per voi interessante.

Links

[1] KC64Wifi di Pasquale De Luna: http://www.codingkoala.com/kc64wifi

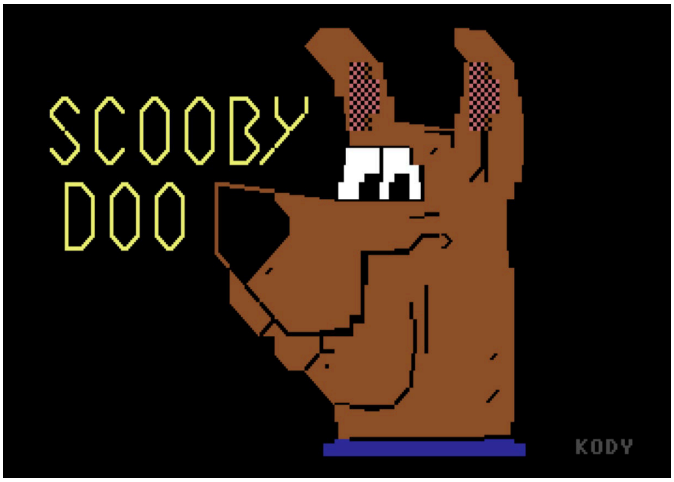


Fig. 6 - Una delle immagini dalla sezione PETSCII-art della BBS © Ivan KodydaKillah

[2] Tiny 8 bit Emulators: https://floooh.github.io/tiny8bit
[3] Emulatore LM80C: https://nippur72.github.io/lm80c-emu
[4] Asmproc: https://github.com/nippur72/asmproc
[5] WebSocket to TCP: https://github.com/nippur72/websocket-to-tcp
[6] Let's Encrypt: https://letsencrypt.org
[7] BBS http://bbs.retrocampus.com e http://bbs.retroacademy.it
[8] L'emulatore C64 online https://nippur72.github.io/c64-emu
[9] Sorgenti dell'emulatore C64-emu e del programma "terminal.prg" https://github.com/nippur72/c64-emu
[10] Compilatore Emscripten https://emscripten.org

-> Nella pagina successiva il sorgente di "**terminal.lm**".

```
   processor 6502

// routine del kernal
const GETIN  = $FFE4
const CHROUT = $FFD2

// porte del modem virtuale
dim MODEM_DATA_OUT    as byte at $D7F2    ; caratteri da inviare
dim MODEM_DATA_IN     as byte at $D7F0    ; caratteri in arrivo
dim MODEM_DATA_REQ    as byte at $D7F3    ; 1 se ci sono caratteri in arrivo nel buffer
dim MODEM_ACK         as byte at $D7F1    ; porta per l'handshake
dim MODEM_CONNST      as byte at $D7F4    ; status del modem (0=connesso, <>0 disconnesso)

    org 2049

basic start compact
   2020 sys {main}
basic end

main:
    lda #0   : sta 53280 : sta 53281 ; black screen
    lda #15  : sta 54296     ; max volume
    lda #14  : jsr CHROUT    ; lowercase
    lda #147 : jsr CHROUT    ; clr
    lda #5   : jsr CHROUT    ; white

terminal:
    ; controlla se ci sono caratteri da stampare nel buffer
    lda MODEM_DATA_REQ
    if a<>#0 then
        jsr cursor_off
        do
            ; esci da modalità QUOTE e INS
            ldx #0
            stx $D4
            stx $D8
            ; legge il carattere dal modem
            lda MODEM_DATA_IN
            ; se è CHR$(7) emette il suono della campana
            if a==#7 then jsr term_bell
            ; stampa il carattere a video
            jsr CHROUT

            ; notifica al modem che il carattere è stato ricevuto
            lda #0
            sta MODEM_ACK
            lda #1
            sta MODEM_ACK
            ; controlla se ci sono altri caratteri da stampare nel buffer
            lda MODEM_DATA_REQ
        loop while not zero
        jsr cursor_on
    end if

    ; legge un carattere da tastiera e lo manda al modem
    jsr GETIN
    if a<>#0 then sta MODEM_DATA_OUT

    ; aggiorna il colore dello schermo con lo status del modem (nero/rosso)
    lda MODEM_CONNST
    sta 53280
    sta 53281

    jmp terminal

; cursor on/off, term_bell (C) Francesco Sblendorio
; https://github.com/sblendorio/ultimateii-dos-lib/blob/master/src/samples/screen_utility.c

cursor_on:
    ldy #$00
    sty $cc
    rts

cursor_off:
    ldy $cc
    if zero then
        ldy #$01
        sty $cd
        do
            ldy $cf
        loop while not zero
    end if
    ldy #$ff
    sty $cc
    rts

term_bell:
    ldy #15  : sty $D418
    ldy #20  : sty $D401
    ldy #0   : sty $D405
    ldy #249 : sty $D406
    ldy #17  : sty $D404
    ldy #16  : sty $D404
    rts
```

Comporre musica per un retrogame

di Phaze101 (traduzione di David La Monaca)

Prendere il comando delle operazioni

La composizione di una musica per un gioco inizia con una richiesta, di solito attraverso un messaggio privato che arriva sulla nostra pagina [R1] o sul gruppo Facebook [R2], su Discord [R3] o sui recapiti che appaiono sul nostro sito web [R4]. A volte si tratta dello stesso programmatore del gioco oppure qualcuno del team che fa la richiesta: "Ciao, potresti comporre musica per il mio gioco per C64?" o anche, "Ho sentito la tua musica in un gioco, saresti interessato a fare musica per il nostro prossimo gioco su Amiga?" E se si tratta di qualcuno che già conosciamo, sarebbe qualcosa del tipo "Che ne dici di un po' di musica ed effetti sonori per il mio nuovo gioco MSX?" In molti casi, siamo noi a seguire da vicino alcuni progetti interessanti e sollecitare la richiesta offrendoci di fare musica per loro. Prima di accettare, facciamo un paio di controlli per capire quale sforzo sarebbe necessario (ad esempio, il numero di brani necessari per il gioco) per essere sicuri di poter rispettare le scadenze concordate. Cerchiamo anche di verificare se il progetto del gioco sia effettivamente realizzabile e che sia già in corso. Purtroppo in passato ci è capitato di impegnarci a fare musica per giochi che poi non sono mai stati pubblicati. E questo è sempre un peccato. Ovviamente alcune "richieste" riguardano i progetti interni al gruppo Phaze101 ma le trattiamo più o meno come qualsiasi altro progetto.

Trovare l'ispirazione giusta

Una volta deciso di salire a bordo di un progetto cerchiamo di ottenere quante più informazioni possibili sul gioco stesso. Video clip, schermate e trama, tutte queste cose ci aiutano a catturare l'atmosfera del gioco. A volte ci viene fornita una musica di esempio che si adatta alla sensazione del gioco, anche se questo non è sempre l'approccio migliore e potrebbe essere addirittura controproducente. Come già detto, il tema musicale viene creato appositamente in base alla richiesta. Crediamo che la cosa più importante per la musica del gioco sia quella di arricchire l'esperienza dell'utente, per far parte del "feeling" complessivo del videogame. Tirar fuori una melodia orecchiabile è fantastico, ma se non si adatta all'atmosfera del gioco potrebbe in realtà peggiorare l'esperienza generale del giocatore. Nella nostra musica cerchiamo di catturare l'essenza del gioco e questo funziona ancora meglio quando abbiamo mano libera per prendere la direzione che riteniamo sia la migliore. Facciamo anche domande specifiche sul progetto per saperne di più sui requisiti, i limiti e i vincoli. Questi elementi riguardano aspetti tecnici come il limite delle dimensioni dei file, la durata, il looping, il numero di canali da utilizzare e se la musica condividerà un canale con gli effetti sonori del gioco.

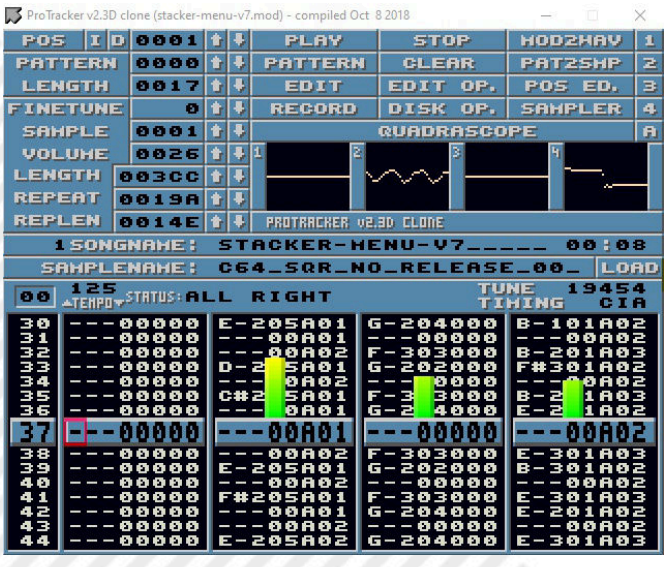


Fig. 1 - ProTracker Stacker 101

Comporre il tema musicale

Una volta che siamo pronti a partire, l'aspetto creativo è una delle parti più difficili dell'intero processo. Di solito si comincia con delle piccole clip fatte per trovare il feeling corretto sulle sonorità. Molte parti (e a volte intere melodie) vengono scartate (a volte il giorno dopo averle composte) per mantenere solo quelle che "suonano" bene. In questa fase all'interno dei membri di Phaze101 ci scambiamo molte opinioni da pari a pari. Quando crediamo di aver creato qualcosa di abbastanza buono per iniziare, chiediamo un feedback anche a chi ci ha commissionato il lavoro, tanto per essere certi che tutti siano in sintonia prima di passare ai passi successivi. Ottenere il suono giusto può rappresentare una sfida difficile. Naturalmente teniamo conto dei limiti del particolare chip sonoro da usare e cerchiamo di sfruttarli al meglio. Dopo tutto, è proprio questo il divertimento di fare musica su macchine retro! Per chi è abituato a comporre musica su piattaforme moderne, dove una singola nota o un campionamento può essere sufficiente a creare il fattore "wow" (o fattore

sorpresa) per l'intero brano, i limiti dei chip sonori di un tempo possono rappresentare una vera e propria frustrazione. Ad esempio, creare i campionamenti giusti su Amiga non è affatto facile, anche se questa piattaforma fornisce più spazio alla creatività. Tutto questo lavoro deve rientrare entro i limiti delle dimensioni del file o di memoria imposti dai requisiti del gioco.

Una volta che il suono è stato più o meno definito (anche se questo potrebbe essere costantemente ritoccato e a volte completamente modificato), si inizia a lavorare sul tema musicale vero e proprio da abbinare alle varie parti del gioco. A volte scrivere una bozza della melodia richiede solo poche ore di lavoro, a volte il progetto viene lasciato da parte per giorni o settimane, in attesa di una scintilla di ispirazione. Ciò è particolarmente vero se sono necessari più brani per lo stesso gioco, perché prendersi un po' di tempo libero dal progetto può significare ricaricare la mente con nuove idee. Quando arriva l'ispirazione, se un computer con un tracker non è a portata di mano, canticchiare la melodia e registrarla su un cellulare è una buona alternativa per assicurarsi che l'idea brillante venga catturata e non vada persa. Qualche volta provare prima una melodia su una tastiera di pianoforte o su una chitarra, aiuta a far scattare le idee che possono poi essere trasferite su un tracker in uno stato già maturo. A mano a mano che le cose progrediscono, cominciamo a provare le parti musicali con il gioco stesso. Tanto per cominciare, suonare la musica contemporaneamente alle videoclip di gioco esistenti (o agli screenshot) aiuta molto a capire come sarà il prodotto finito. Riproduciamo la musica più e più volte (soprattutto se la musica dovesse andare in loop più volte all'interno del gioco) per vedere se diventa troppo rapidamente fastidiosa. Questo ci aiuta a cercare di

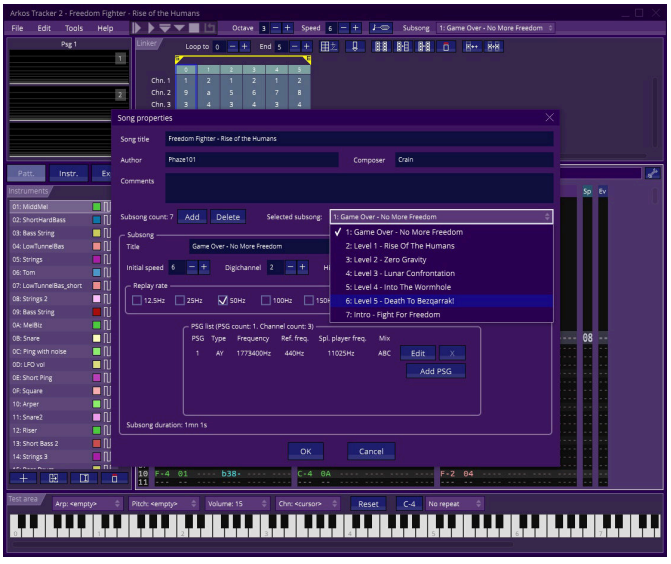


Fig. 3 - Arkos - Freedom Fighter

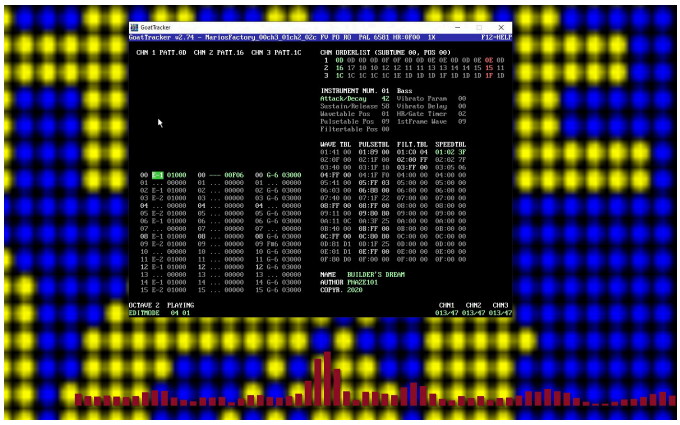


Fig. 2 - Goattracker - Mario Cement Factory

minimizzare l'effetto della ripetitività apportando le modifiche che non hanno un "peso" significativo sulla dimensione complessiva del file.

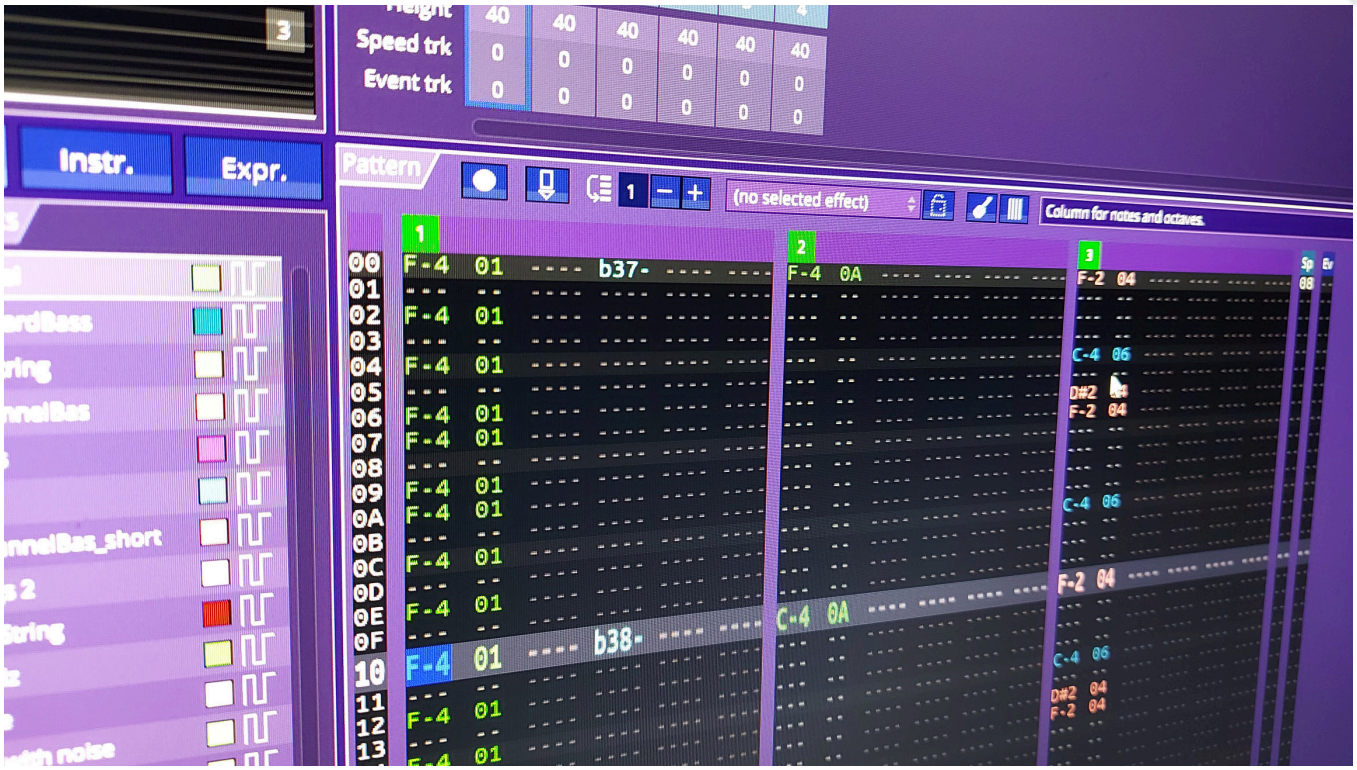
A questo punto iniziamo anche a concentrarci sugli effetti sonori, per assicurarci che tutto rientri nel feeling generale del gioco e che i livelli di volume siano corretti. Talvolta ci arriva la richiesta di comporre solo la musica, mentre gli effetti sonori sono realizzati da qualcun altro. Ci capita anche a partecipare a progetti in cui più tracce sono fatte da compositori diversi. Questo rende ancora più importante mantenere una visione olistica del prodotto e garantire che tutto l'insieme suoni come dovrebbe.

Mentre lavoriamo alle parti musicali e soprattutto quando sono per lo più prossime alla loro finalizzazione, facciamo in modo di ascoltare la musica su dispositivi diversi. Se tutto suona bene su cuffie normali a buon mercato o su cuffie di più alta qualità, su altoparlanti per monitor, altoparlanti esterni, telefoni cellulari e stereo della macchina... allora probabilmente suonerà bene sulla maggior parte dei dispositivi impiegati dagli utenti giocatori. In collaborazione con chi ci ha commissionato il lavoro, la musica viene anche ascoltata e valutata sia sugli emulatori sia sull'hardware originale. Per il C64/128 ci assicuriamo che suoni bene su entrambi i modelli di SID, MOS 6581 e 8580 e per questo evitiamo l'uso pesante dei filtri offerti dal SID stesso.

Gli attrezzi del mestiere

Lo strumento principale di composizione è ovviamente un buon tracker. Abbiamo lavorato con molti tracker diversi nel corso degli anni, ma quando si fa musica per gli altri una delle preoccupazioni principali è quella di utilizzare prodotti che offrono una buona routine di riproduzione sonora stabile che può essere facilmente integrata dal gioco che intendiamo musicare.

La nostra preferenza per il Commodore 64 va a Goattracker



perché crediamo che sia attualmente uno dei migliori tracker in circolazione per questa piattaforma.

Usiamo Arkos Tracker per fare musica per il chip AY di MSX, ZX Spectrum e CPC. Possiede un'interfaccia moderna ed è molto facile da usare rispetto anche ad altri tracker.

Quando si tratta di Amiga, il tracker prescelto è Pro Tracker (incluso il clone Windows di Olav Sørensen).
Per Amiga disponiamo anche dell'aiuto di Renoise su Windows con vari strumenti VST per generare campioni, e strumenti come Audacity per elaborarli. Ogni tanto ci capita di utilizzare altri strumenti software e siamo sempre felici di sperimentare e provare nuovi tool.

I progetti in corso

Il 2020 è stato un anno piuttosto impegnativo, perché a parte i nostri progetti in corso abbiamo realizzato la musica di diversi giochi, e molti altri sono ancora in corso di lavorazione.

- “Mario Cement Factory” di Hayesmaker, un porting del gioco portatile Game & Watch retro per il Commodore 64 (già recensito in RMW #25-IT).
- “Freedom Fighter - Rise of the Humans” di Giuseppe Ettore Pintus (aka Geppo) per MSX, pubblicato nel luglio 2020 (già recensito in RMW #25-IT).
- “Neptune Lander Elite” di C64_Mark, una variante del classico gioco lander per Commodore 64, prossimo alla pubblicazione.
- “Pik'n'Mix” di Shallan50K per Commodore 64, in uscita per la fine dell’anno.

Scrivere il finale

Quando tutto è pronto dal punto di vista musicale, a volte ci vogliono comunque settimane o mesi prima che il gioco venga pubblicato ufficialmente. Quando viene rilasciato, ci preoccupiamo di supportare il gioco e l'autore mettendo un video sul nostro canale YouTube [R5] o promuovendo il gioco sui nostri canali social media. È sempre un onore e un privilegio per noi essere invitati a lavorare a progetti e sentirci orgogliosi di condividere il nostro lavoro in ogni modo possibile.

Riferimenti

R1 – Phaze101 FB Page: https://www.facebook.com/Phaze101Games
R2 – Phaze101 FB Group: https://www.facebook.com/groups/Phaze101
R3 – Phaze101 Discord: https://discord.gg/tXnRe4XYV7
R4 – Phaze101 Web Site: https://phaze101.com
R5 – Phaze101 YouTube: https://www.youtube.com/c/Phaze101

Giappone 15^puntata:
Oh no! More G&W!

di Michele Ugolini

"Oh no! More lemmings!" recitava il titolo Psygnosis che abbiamo giocato negli anni 90. "Oh no! More G&W!" recita il titolo di questo numero di RMW visto che questi prodigi elettronici stanno tornando nel mercato odierno come i famosi Gremlins.

Finalmente è uscito il G&W originale Nintendo per il 35° anniversario della grande "N". L'avete già messo in vetrina? Quanti minuti lo avete utilizzato? Problemi alla vista, oppure tutto bene?

A prescindere da queste domande provocatorie rimane un oggetto sacro per noi collezionisti. La sensazione dei tasti in gomma è identica ai tasti originali del G&W di 40 anni fa.
Il display, il design, la disposizione grafica, sono tutti elementi semplicemente adorabili. A differenza delle vecchie batterie a pastiglia, questa volta la batteria è inserita internamente, riscuotendo apprezzamenti e malumori dal web: quando la batteria sarà esausta , purtroppo, sarà da estrarre per evitare rigonfiamenti o dispersione del contenuto.

Altri malumori dal web?
Il prezzo è stato abbondantemente criticato, inoltre, cosa non da poco, in questa versione del G&W non è presente uno stand per tenerlo verticale nella nostra vetrina.
Uno stand in cartoncino era invece presente nel remake del G&W "Ball" del 2010 che avevo recensito alcune puntate fa, un prodotto cartaceo economico ma robusto, funzionale e soprattutto marchiato "G&W". Peccato, dovremo acquistare uno stand anonimo dal web!

Infine l'ultimo malumore proveniente da internet: la sveglia. Difatti è presente un orologio con diversi wallpaper, però non è presente una sveglia che possa tirarci giù dal letto al mattino, magari con la suoneria di Mario.
Grande peccato.

Pregi? Si, ovviamente tantissimi: stiamo giocando ad un prodotto ufficiale Nintendo e soprattutto dal web sono già arrivati numerosi tutorial, che ovviamente sia RMW che io sconsigliamo fortemente di cercare/visionare/applicare, tramite i quali si può modificare l'avvio di Mario e far girare Doom.
Probabilmente in futuro si potranno inserire altri programmi e giochi... ma la sacralità di un oggetto ufficialmente rilasciato da Nintendo svanirà!
Sappiate comunque che numerosi temerari sono già riusciti nella riprogrammazione.
Doom è un titolo epico ma la grafica degli anni 90, unita ad un display così ridotto, genera diversi problemi alla vista: sfido chiunque a non lamentarsi a causa di disturbi alla vista dopo mezz'ora di gameplay.
D'altro canto, il gusto di far girare svariati giochi/programmi in questo G&W, denota l'alto ingegno dei nostri amici più temerari. (cfr. figura 1)

Come sappiamo Nintendo è famosa per generare fermento nelle ditte dei suoi rivali. Rivali non nel vero senso della parola dato che il marketing in Giappone crea unicamente altro marketing ed ogni ditta si inserisce negli introiti attraverso una propria intelligente fetta di servizi in stile "subfolder".

Fig. 1 - Doom on Nintendo Game & Watch

Ed ecco la notizia giunta da alcuni giorni sul web: Capcom lancerà sul mercato, tramite Amazon Japan, "Retro Station".
Sarà un mini cabinato, dal particolarissimo stile vintage.
Personalmente giudico il design molto provocante, nel senso che ancora non riesco a decidere se lo amerò o non lo digerirò! L'unica certezza è che non rimarrà inosservato.

Ad Osaka non si scherza, Capcom inserirà dieci giochi, potremo giocare a Street Fighter (II, II Champion Edition, Super Street Fighter II, Super Street Fighter II Turbo, Super Puzzle Fighter II Turbo) e Mega man (The Power Battle, 2 The Power Fighters, X, Soccer, Man & Bass).
Le dimensioni sono: 329 mm x 280 mm x 315 mm. Il peso sarà rilevante: 2.1 kg. (cfr. figura 2)
Troveremo un display e speaker stereo posizionati frontalmente, dal web si intuisce che l'audio godrà di ottima qualità.
In Giappone sarà possibile acquistarlo al prezzo di 21.780 yen, pari a circa 176 dei nostri euro. Dovremo aspettare l'import occidentale che subirà le varie maggiorazioni doganali.

Questa volta il prezzo è alto e l'oggetto è molto particolare, riusciranno i nostri eroi, della Capcom, a stare al passo con le abili strategie del mercato della Nintendo?
Non è una domanda inutile, dobbiamo infatti ricordare il limitato successo riscosso dalla mini consolle Capcom Arcade, varata nell'Aprile del 2019: le unità vendute non hanno suscitato particolari malumori presso Nintendo, la partita questa volta è aperta e le variabili sono azzerate, solo il futuro potrà raccontarci il passato di questa "operazione nostalgia"... ancora oggi perpetrata dai colossi nipponici.

Vogliamo parlare infine dei Gig Tiger finalmente arrivati anche sul mercato italiano?
Tre versioni al momento molto apprezzate: Marvel X-Men ProjectX , Sonic the hedgehog 3, Transformers generation 2. I prezzi rilevati dal web hanno già iniziato la loro salita e discesa. Abbiamo affrontato un Black friday anomalo, un Cyber monday intorpidito e ci aspetta un Natale ben diverso dal solito, la società purtroppo si è dovuta interfacciare con un nemico insidioso e silenzioso. Anche le strategie di mercato hanno dovuto affrontare nuove variabili, alcune insospettabilmente fruttuose, altre incredibilmente fallimentari.

Le vendite dei GIG Tiger, sembra, stiano andando bene, sul sito camelcamelcamel.it si potranno tenere controllate le curve dei prezzi.

https://it.camelcamelcamel.com/

Una notizia inaspettata proviene recentemente dalla SEGA, la stessa SEGA che ha concesso alla Hasbro di produrre un GIG Tiger in onore al nostro amato Sonic.
La triste notizia è che, sebbene la pandemia abbia portato affari d'oro al mondo videoludico, il divieto di assembramenti ha stroncato il settore delle "sala giochi".
Sega Sammy Holdings, settore famosissimo in

Fig. 2 - Capcom Retro Station

Giappone per curare il reparto sala giochi , cabinati e mondo arcade ha dichiarato di aver subìto un calo significativo del fatturato.
Shukan Famitsū è la testata videoludica più importante e rispettata in Giappone, contiene soprattutto recensioni e notizie dal mondo dei videogiochi e recentemente ha pubblicato la dolorosa notizia della SEGA: la vendita dell'85% delle quote a GENDA INC.

Tutti noi speriamo di poter giocare in futuro ai magnifici cabinati giapponesi dentro Tokyo, ancora marchiati SEGA, ma il famosissimo SEGA building di Akihabara sparirà dalla affollatissima via dell'electric town.
Sarà felice Laox?
O forse Bic Camera?
Non credo!
Penso che anche la palazzina dietro la Sega, che ospita Super Potato, sarà tristissima.
Chissà che fine faranno il "SEGA Akihabara Building" e il vicino "SEGA VR area"?
(cfr. figura 3)

Il Giappone è famoso per reinventarsi e per rinascere dai propri problemi e sconfitte. Non è assolutamente improbabile che una ditta concorrente (non nemica, ma solo posizionata in una fetta congruente dentro il medesimo settore del marketing) tenda la mano al 15% della SEGA.
Oppure sarà Microsoft a tendere la mano? Se Microsoft decidesse di unire anche SEGA, dopo i coraggiosi matrimoni con Bethesda, Ninja Theory, Playground Games, Obsidian, vorrebbe dire soltanto che il "monopolio PlayStation" dovrebbe rivedere l'intera organizzazione interna di casa Sony ed operare azioni più chirurgiche nel proprio immediato futuro!
Vi allego intanto un link per una piacevole lettura riguardo il futuro mondo videoludico:

https://www.everyeye.it/notizie/microsoft-sega-matrimonio-s-ha-fare-nuovo-indizio-greenberg-483922.html

Ultima indiscrezione: nel 2021 ci saranno importanti novità per quanto riguarda Sonic.
Lo ripeto da svariati numeri di RMW: il mondo G&W non è alla fine nè tantomeno il mondo del retrogame.
E' ovvio che anche SEGA si stia trasformando, l'epidemia permetterà all'abilità nipponica di inventare qualcosa di geniale, per alimentare un marketing post pandemia completamente nuovo e rimodernato.
Non lo sono forse anche le stesse vie delle città a distanza di un anno?
Non troveremo mai negozi, insegne, indicazioni stradali ed edifici identici al precedente anno nel quale avevamo visitato il Giappone.

E' tutto. Rimanete sintonizzati, una poderosa rinascita è alle porte nipponiche: un ritorno scoppiettante per foraggiare noi fedeli consumatori e collezionisti.

A presto, a tutti voi i migliori auguri di Buone Feste!

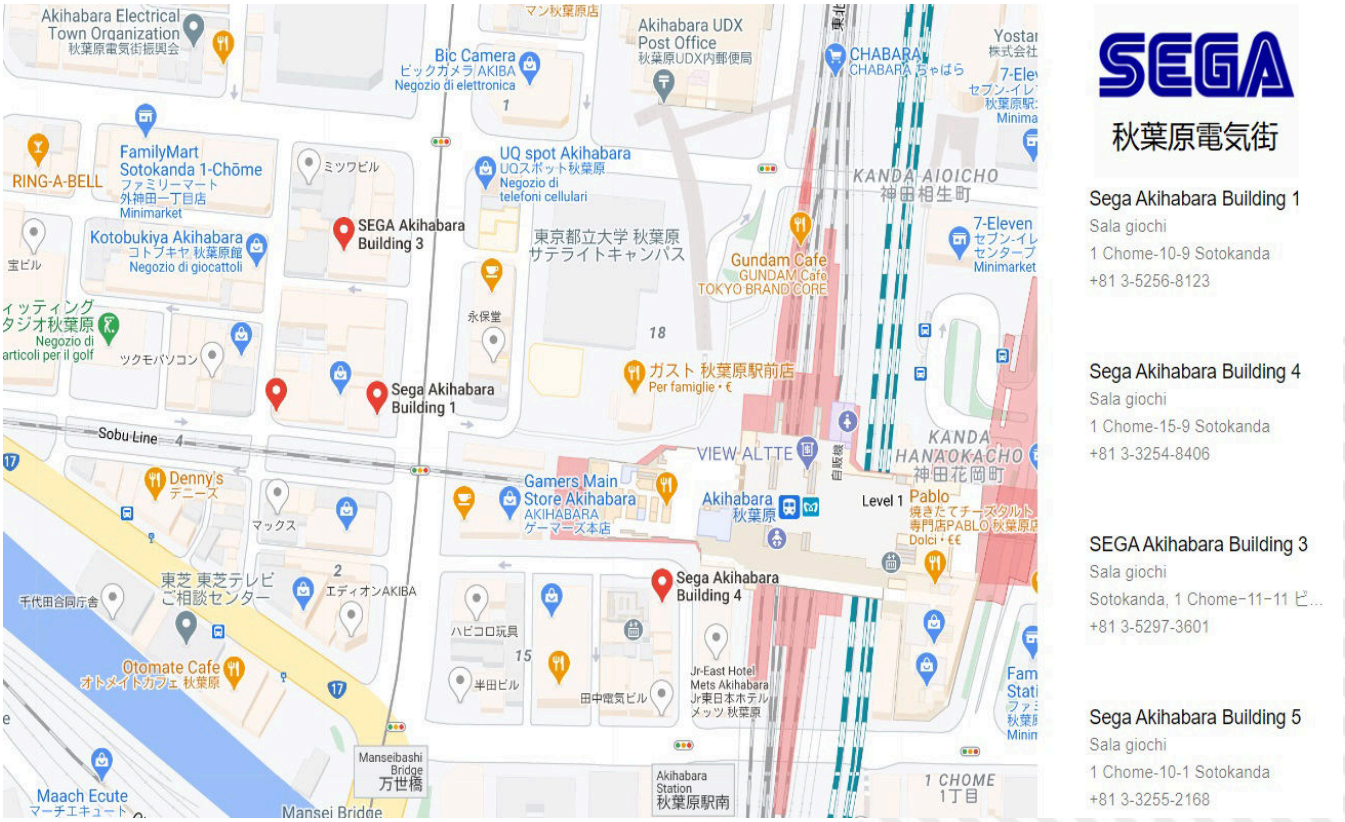


Fig. 3 - Tokyo quartiere Akihabara

Intervista esclusiva a Randall Flagg

Phreaker/cracker/hacker della scena DOS/Windows internazionale negli anni '90 e 2000

di David La Monaca

1 - Intro

Chi si ricorda di DOTT, ossia Day Of The Tentacle? Sicuramente molti dei nostri lettori ne avranno un bel ricordo. Era un adventure "point & click" della LucasFilm Games, noto anche come Maniac Mansion II, basato sull'altrettanto noto motore di sviluppo SCUMM. Ciò che non è altrettanto noto al pubblico di tanti retrogamer è l'esistenza di una piccola storia sul crack di DOTT, che vogliamo raccontarvi qui di seguito, una storia che parla della competizione fra gruppi della cracking scene degli anni '90. E la competizione si svolgeva tutta sulla fama, sulla rispettabilità e sulle capacità tecniche dei vari membri della scena. Chi per primo riusciva a rimuovere o superare una protezione da videogame otteneva notorietà e rispetto, oltre al non trascurabile diritto ad "infamare" o sbeffeggiare gli altri gruppi nei file di testo .NFO che accompagnavano i crack.

Nel caso di DOTT, la storia andò più o meno così: pare che quelli di Hybrid non avessero le abilità necessarie, così a un certo punto hanno rinunciato. Il gioco non riuscivano a craccarlo nemmeno quelli degli Swedes. Addirittura arrivarono a dichiararlo "incraccabile", forse per guadagnare un po' di tempo sugli altri gruppi, scoraggiandoli dal partecipare alla competizione e cercare di essere comunque i primi, continuando a lavorare alacremente sul crack. Ma questo era uno dei trucchi ben conosciuto da tutti nella scena e considerato unanimamente poco corretto.

Così, alle 9:40 del 12 giugno 1993, un membro di Hybrid scrisse questo messaggio: "Ok ragazzi, ecco qui il primo dischetto di DOTT! Quindi smettetela di dire cazzate.. Eh sì, esatto, è una rottura di coglioni da crackare, così adesso abbiamo dato a tutti quanti una buona possibilità di farcela,

Fig. 1 - Randall Flagg aka Antonio Mazzanti

credo! C'è probabilmente anche un'altra protezione da copia dopo quella stronzata della protezione della BATTERIA SPECIALE, quindi cercate di capirci qualcosa! - Hoson/Hybrid 1993 (incazzato, stanco, tosto, pesante, metallaro). P.S. Almeno siamo abbastanza uomini da ammettere le nostre sconfitte! P.P.S. Ci stiamo ancora lavorando!"

Il team Razor 1911, divisione europea (naturalmente!), alle 7:33 dello stesso giorno (12 giugno), ora europea, ricevette la seguente nota, scritta da un cracker che agiva sotto il nickname Randall Flagg: "Day Of The Tentacle - Maniac Mansion II - Cracked by Randall Flagg! Nota: --- per uso pubblico ---"

Già, tutto chiaro, quindi, almeno così sembrerebbe. Ma indovinate un po'? Esatto, la corsa non era ancora finita! Altri rappresentanti della scena, provenienti dal mitico gruppo The Dream Team, e più precisamente Dr. Detergent/TDT [R.I.P. :-(], pare che avessero già risolto la partita alle 13:48 dell'11 giugno. Nel loro file NFO si legge: "Eravamo concentrati sul gioco e sapevamo che ce l'avremmo fatta, ma non QUANDO! Finalmente ora, la vostra squadra di cracking preferita ce l'ha fatta di nuovo! Ed è venuto fuori un CRACK bello pulito! TDT non ha niente a che vedere con la beneficenza e pubblicare una crack patch sarebbe soltanto come regalare il rilascio ad un altro gruppo! Dopo aver installato il gioco, assicuratevi di copiare il file DOTTCRK.COM e di lanciarlo OGNI VOLTA prima di cominciare a giocare! Come avete letto dall'NFO di Hybrid, hanno detto a tutti che NON era stato craccato e che quindi tutti potevano craccarlo e arrivare per primi, ma TDT ha battuto tutti sul tempo! AVVISO PER I RILASCI DI HYBRID! Non disturbatevi a scaricarli, se non siete dei cracker voi stessi.. hehe"
Quindi la questione qui è DOVE è stato craccato il gioco (considerando solo Razor 1911 e TDT), scoprire il fuso orario corretto e stabilire CHI è arrivato per primo alla mèta!

Per capire quale fosse il livello della competizione fra i diversi gruppi, bisogna dare un'occhiata al file .NFO di un'altra squadra. Stavolta si trattava dei GENESIS*PROJECT, una parte del leggendario gruppo che operava nella scena del C64, che orgogliosamente continuava ad occupare nella scena un posto di rilievo. Il 12 giugno 1993, alle 19:03, Snacky/G*P scrisse: "Beh, di sicuro abbiamo avuto qualche problema all'interno di GENESIS e anche con alcuni sfigati che si sono professati membri di G*P, ma ora sono davvero orgoglioso di presentarvi una crack-patch per questo fantastico gioco... Mi ci sono volute alcune ore per metterla a punto e spero che a tutti voi piaccia! Ehi, voi di TDT, che vi succede...? Bella release eehhhh, un altro dischetto n. 6 da diffondere in giro per una crack-patch

che NON funziona... hahahah! Ehi, voi di Hybrid, che combinate...? Prima di arrendervi e dire che questo gioco NON può essere craccato, la prossima volta aspettate qualche altro giorno ed ecco qua! Assicuratevi di scaricare i nuovi programmi trainer e le nuove release del vostro gruppo di intrattenimento preferito del 1993!"

Quindi, anche se l'orario rivela ogni cosa, questa sembrava essere la crack-patch DEFINITIVA, non una qualunque release bacata. Conoscendo le capacità di Snacky nessuno ne dubitava, ma confrontando l'orario del file di patch, Randall Flagg/Razor1911 risultava il primo ad essere arrivato al traguardo, seguito poi dagli altri. Ma non bisogna dimenticare che i cracker stavano anche testando a fondo il gioco, controllando in questo modo la protezione. Dopo tutto, si trattava di un gioco che utilizzava una protezione abbastanza sofisticata che si attivava durante lo svolgimento del gioco, procedendo lungo la trama. Questo importante fatto sembra superare la questione dell'orario di rilascio per stabilire chi davvero fosse giunto per primo alla sprotezione COMPLETA del gioco.

Quindi potremmo concludere che G*P avesse pienamente vinto la gara. Ma c'è un problema: perché nel file crack di G*P chiamato GENESIS.GP (il crack consiste in due piccoli file, DOTTCRGP.COM e GENESIS.GP) si può leggere chiaramente: "RAZOR/Maniac Mansion II". Forse non significa nulla, oppure sì? L'unico che davvero potrebbe aiutarci a dirimere la questione sarebbe lo stesso Snacky, un cracker della scena C64 davvero abile e rispettabile, uno che non ha mai rubato una crack: ci avrebbe messo molto più tempo che craccare un gioco tutto da solo. Sfortunatamente non abbiamo una testimonianza diretta di Snacky, ma abbiamo trovato un commento di Randall Flagg datato 2016 su un thread dedicato a questo crack: "Ehi... posso soltanto dire che è stato molto divertente da craccare!"

Alcuni sostengono che all'epoca Snacky fosse un membro sia di G*P sia di Razor... Ma l'ultima parola su tutta la storia ce la fornisce lo stesso Randall Flagg: "Posso soltanto aggiungere che nel mio crack file ho anche cambiato il testo in cui Dr. Fred fa scattare la protezione chiedendo gli ingredienti della super batteria (il testo ASCII era mascherato tramite XOR come per altri giochi LucasFilm da un singolo byte, forse 0x69h se ricordo bene). Il crack avrebbe cancellato completamente quella scena con Dr. Fred, lasciando passare il giocatore senza richiedere alcunché. Ma lo faceva anche nel caso in cui in seguito il giocatore avesse guardato i piani della batteria. Niente di così complesso. Se non mi sbaglio, altri 2 o 3 gruppi annunciarono il loro crack, anche 2-3 giorni dopo e poi sono stati annientati praticamente ovunque. La versione RAZOR è l'unica che si può ancora trovare in giro, persino su file torrent. :)".

Conclusione: il crack di Randall Flagg è stato il primo ad apparire cronologicamente ed era anche già COMPLETO!

Fig. 2 - Antonio nel 1993 al telefono (da notare il fido modem US Robotics HST sulla credenza, la caotica postazione PC e le bacchette da batteria)

Andiamo allora ad incontrare l'autore di questo ed altri innumerevoli crack di giochi DOS/Windows degli anni '90 e 2000. I lettori italiani di RMW andranno fieri del fatto che uno dei più autorevoli e abili cracker di tutta la cracking scene sia stato l'italianissimo Randall Flagg, al secolo Antonio Mazzanti.

2 - .NFO

DLM – Ciao Antonio e grazie per aver accettato il nostro invito. Come abbiamo cercato di spiegare nell'introduzione, sei un personaggio poco conosciuto al di fuori di una certa cerchia di tecnici, hacker e appassionati. Eppure molti di coloro che stanno leggendo ora quest'intervista devono a te la chance di aver potuto giocare a quei titoli per PC DOS e Windows degli anni '90 che presentavano una protezione particolarmente articolata e ostica da superare. Oltre al già citato DOTT, basti pensare alla serie di avventure grafiche della LucasFilm Monkey Island e Indiana Jones. Ti chiediamo allora di presentarti brevemente e di darci qualche notizia su di te.

Grazie a te per l'invito. Non sono un tipo che ama tanto parlare o scrivere di sè. Quando mi si chiede di raccontare di me stesso o delle mie esperienze in modo formale spesso declino. Preferisco una bella chiacchierata in amicizia, come stiamo facendo io e te adesso a voce, anche se in due città diverse. Sono nato a Firenze nel 1972, ho trascorso una bella infanzia ed ho avuto una mamma molto forte e intelligente. Tutto normale con la scuola dell'obbligo, poi alle superiori ho scelto Ragioneria per Programmatori, una scelta più dettata dal desiderio di trovare nella scuola un modo per imparare ad usare i computer. Ovviamente qualcosa ho imparato, ma se proprio devo esser sincero, devo ammettere che in pratica ho fatto tutto da solo. Guidato soprattutto dalla mia innata curiosità, ho sempre voluto sapere cosa c'era dietro ai giochi e al software che vedevo girare sui computer che via via ho posseduto. Mi sono sempre interessato (e lo faccio ancora) sia di hardware e di elettronica, sia di software e di programmazione. Dopo

la maturità, mi sono iscritto anche all'università, ma non l'ho mai finita, mi sono sempre sentito troppo avanti, eheheh.

DLM – Sei sempre stato un appassionato di computer fin da bambino? Che cosa ti ha attirato verso il magico mondo digitale?

Non posso dire di essere un "appassionato" neppure oggi. Da ragazzino niente elettronica e informatica fino a quando arrivò in casa uno ZX Spectrum, il mio primo computer, comprato dalla mamma lungimirante. Ovviamente cercai subito i giochi e poi cominciai pian piano a sperimentare con programmini in BASIC, il mio primo programma fu un programma per stampare le tabelline. Il ciclo FOR fu per me illuminante e da lì in poi compresi che quel mondo era per me. M'interesso e lavoro con l'elettronica solo da una decina d'anni e faccio reversing praticamente da sempre. All'inizio fui attirato dalle intro che apparivano sulle copie pirata dei dischetti per C64.

DLM – Dicci di più sul tuo primo computer? Lo possiedi ancora o ne hai presi molti altri nel tempo?

Mia madre aveva una visione chiara del futuro dei suoi figli e per me aveva pensato alle due "I": inglese e informatica. Così da ragazzo feci un lungo soggiorno studio in Inghilterra che mi servì moltissimo in seguito per tutte le mie avventure digitali e per la mia vita professionale. Un giorno arrivò in casa grazie a lei uno ZX Spectrum 16KB, completo di registratore a cassette Philips. Non l'ho più ma con il tempo ho praticamente riacquistato tutti i retro computer che in me riuscivano a suscitare un po' di nostalgia, eheh.

DLM – Quando hai cominciato ad interessarti al dietro le quinte di programmi e giochi? Ovvero, quando hai iniziato a scoprire il linguaggio macchina e l'assembly?

Quando uscì Maniac Mansion per C64, mi ci appassionai ma, procedendo nell'avventura a un certo punto la mia copia si piantava. Ero bloccato e non potevo sapere cosa accadeva nel resto del gioco. Così presi lo stesso gioco per PC. Avevo circa 15 anni e scoprii che tramite un semplice debugger potevo vedere cosa faceva un programma o un gioco caricato in memoria. Alla scuola superiore avevo avuto esperienza con il COBOL e con il modulo a runtime. Avevo imparato la differenza tra programma interpretato e programma eseguibile. Così, con il Turbo Debugger riuscii a trovare l'istruzione JMP che mi serviva per bypassare la protezione. Ricordo che quel momento fu particolarmente eccitante. Da lì in avanti m'interessai anche agli altri giochi della Lucasfilm, fino ad arrivare a The Secret of Monkey Island e poi The Day Of The Tentacle, ma a quel punto ero già passato ad uno strumento più sofisticato e completo di debugging come Soft-ICE.

Il mio primo vero crack fu per Motocross, un gioco per DOS, per il quale, ricordo, mi bastò piazzare in modo strategico un salto incondizionato (in pratica un solo byte). Il crack di DOTT, invece, fu fra i più complessi da affrontare. Fra l'altro la protezione era di due tipi: una all'inizio e l'altra durante una delle fasi dell'avventura. Di sicuro non bastò utilizzare qualche opcode di jump ben assestato, DOTT faceva uso di una stack machine da aggirare con una serie di opcode push/pop e di comparazione. Non a caso solo pochi gruppi si cimentarono nel trovare il crack ed io fui il primo a postarlo in giro.

DLM – Ai tempi non c'era Internet, al massimo qualche lentissimo collegamento via modem alle BBS. Come hai imparato, in concreto, a destreggiarti con la programmazione?

Sono un completo autodidatta. A 14 anni raccolsi i miei risparmi ed andai alla Libreria Universitaria di Via San Gallo a Firenze e ordinai due libri sull'assembly 8086/8088 e 80386, ovviamente in inglese. Sono stati la mia porta d'ingresso per la programmazione a basso livello e per comprendere le routine di protezione che incontravo su giochi e programmi per DOS.

DLM – Oltre al mondo dei computer a 8 e 16 bit, sappiamo che sei sempre stato attratto dal mondo delle telecomunicazioni. Quando è nata questa passione che ti a spinto a curiosare nelle reti telefoniche?

Dopo esser entrato nei RAZOR 1911, cominciai a esplorare il mondo telematico e l'obbiettivo era quello di riuscire a fare telefonate gratis ovunque per poter comunicare rapidamente con gli altri membri del gruppo e per collegarmi alle BBS di mezzo mondo. Si trattava di lambiccarsi il cervello e spremersi le meningi per scoprire i metodi più impensabili e creativi per attaccare sistemi e reti telefoniche. Naturalmente non posso andare troppo nel dettaglio. ;-)

DLM – Ad un certo punto sei approdato alla scena cracking

Fig. 3 - In primo piano Sector 9, membro dei Razor 1911 a cena con il nostro Randall Flagg (a sinistra sullo sfondo)

aderendo a gruppi della scena italiana e internazionale. Fra gli altri ricordiamo Dead Memory, Razor 1911, Eclipse e Hybrid. Nomi che a molti di noi ricordano pomeriggi spensierati passati a giocare titoli pazzeschi e a volte veri capolavori. Come ti è successo di entrare a far parte di questo universo, facendolo diventare un po' il tuo stile di vita?

All'inizio degli anni '90 lavoravo per un negozio di computer della mia città e allo SMAU di Milano conobbi dei ragazzi che facevano parte dei Dead Memory che m'introdussero alla scena cracking italiana (in pratica le poche produzioni di giochi nostrani riguardavano i titoli Simulmondo e altre due-tre software house). Cercavano dei cracker, così iniziai a craccare giochi in italiano per loro. Alcuni di questi titoli come 3D World Tennis, uscito nel novembre del 1992, potevano anche essere interessanti per altri gruppi esteri. Quindi cominciai ad accedere alle BBS e ad esplorare quel nuovo mondo uploadando i crack dei giochi italiani. Un giorno feci l'upload del crack dell'adventure Indiana Jones and the Fate of Atlantis. Mi notò il leader dei Razor (aka The Renegade Chemist) e, dal momento che i giochi Lucasfilm erano considerati molto complessi da sproteggere, mi contattò e mi offrì di far parte del loro team.

DLM – Puoi parlarci dell'atmosfera respirata in quei giorni, quando ti dividevi fra phreaking e cracking? Com'eri organizzato dal punto di vista tecnico e logistico?

L'organizzazione era in fondo molto semplice: quando un gioco necessitava di un crack, il supplier (colui che forniva l'originale) chiamava tramite bluebox, carta di credito o altro metodo il primo cracker disponibile, il quale procedeva a scaricare il gioco direttamente dal supplier. Poi cercava di craccare il titolo e se ci riusciva redigeva un file .NFO (un file di testo) con le note necessarie per la distribuzione. Il supplier chiamava i cosiddetti "courier" (i distributori) ed il gioco craccato finiva rapidamente su tutte le nostre BBS. Da queste, altri courier freelance lo uploadavano sulle BBS di altri gruppi. Gli HQ (Head Quarters) degli altri gruppi avevano la priorità, in questo modo potevano mettere il "sigillo" sulla release. In tutti i passaggi si comunicava principalmente al telefono o con messaggi sull'HQ.

DLM – Tornando alla tua formazione, com'è stata la tua esperienza universitaria? Riuscivi a studiare, lavorare e dedicarti alle attività di phreaking/cracking?

Beh sì, dopo il diploma ho frequentato per tre anni l'Università, prima a Pisa e poi a Firenze, ma sinceramente i corsi mi stavano un po' stretti. Mi sembrava che parlassero di un mondo ormai passato. Tutto quello che ho imparato, l'ho imparato sul campo, sperimentando e stando in contatto con altri cracker della scena, leggendo libri e consultando documentazione tecnica trovata sulle BBS e poi su Internet.

DLM – La nostra rivista si occupa prevalentemente di retrocomputing e retrogaming. Ricordi qual è stato il primo videogame che hai giocato? Ti piace ancora utilizzare l'hardware fisico di un tempo?

Fig. 4 - De Gregorio di OUAS (sx) e Mazzanti (dx) in una foto del 2018 con un noto programmatore/designer della Lucasfilm Games

Non ricordo con precisione il nome del primo gioco che caricai e giocai su home computer. Di certo dev'essersi trattato di un gioco per ZX Spectrum. Ricordo un clone di Pong, che mia madre aveva acquistato. Oggi utilizzo spesso Amiga (i modelli 1000, 500, 600, 2000 e 1200 sono tutti nella mia collezione) e anche molti degli 8 e 16 bit degli anni '80. E ancora oggi mi diverto a craccare giochi MS-DOS e Windows utilizzando hardware reale (un PC 386SX ed un Pentium 200 MHz).

DLM – Sappiamo anche che ti dedichi a progetti di elettronica per professione ed effettui anche riparazioni di vecchie macchine per amici e appassionati. Dopo tanti anni ti diverte ancora mettere le mani su schede e motherboard di tanti anni fa?

Certo! L'idea di fondo nasce dalla voglia di rendere omaggio a chi quelle macchine le ha progettate e realizzate! Vederle ancora oggi all'opera dà i brividi! Spesso riparo i vecchi home computer che trovo in giro per poi regalarli ad amici e conoscenti. Oppure riparo quelli degli amici che mi chiedono aiuto.

DLM – Puoi raccontarci un aneddoto particolarmente avvincente fra tutti quelli che sicuramente hai nella tua memoria che riguarda le attività di phreaking (sempre che si possa raccontare)?

Eheheh... questa è una domanda che mi fa scaturire mille ricordi e mille storie nella memoria. Storie troppo lunghe da raccontare in questa sede, ma non escludo di portarne qualcuna al prossimo OUAS di Roma, come già accaduto a fine ottobre all'OUAS 2019 di Milano.

DLM – E qualche storia che invece riguarda la tua attività di cracking?

Una storia interessante da raccontare sarebbe quella che riguarda un noto programmatore/designer e sceneggiatore della Lucasfilm Games, che durante un incontro di qualche anno fa mi perdonò per aver craccato i suoi giochi (vedi foto del messaggio che mi è stato inviato). Per altri episodi invito tutti a guardare il mio intervento all'ultimo OUAS

2019, recentemente pubblicato su Youtube [R5]. Ne sentirete delle belle!

DLM – Come sceglievi i titoli da craccare? Era una tua libera decisione oppure dipendeva soltanto da cosa procuravano i supplier?

Beh, l'organizzazione dei gruppi era in realtà abbastanza rigorosa. Tutto dipendeva da una chiara gerarchia. Il cracker con la reputazione migliore del gruppo veniva consultato per primo, poi venivano chiamati gli altri. Ma spesso dipendeva anche dal nome della software house o del publisher del gioco da craccare. Un nuovo gioco Lucasfilm o Sierra andava sempre al miglior cracker del gruppo. Una questione di prestigio e di rispetto.

DLM – Hai mai partecipato al coding di una intro o di un keygen di un gioco o programma craccato dal tuo gruppo?

Ho lavorato a diversi keygen, ad esempio quello per il gioco 1:1 via LAN che controllava i serial number dei giochi connessi fra loro. Per quanto riguarda le intro, non ricordo di aver mai scritto una singola riga di codice. A volte mi capita ancor oggi di scrivere dei keygen, ma ormai solo per puro divertimento e un po' d'accademia. E ovviamente non li divulgo.

DLM – Quando ti trovavi di fronte ad una protezione, qual era la motivazione più forte che provavi nel tentare l'impresa?

L'obbiettivo era sempre quello di essere il numero uno, il più bravo, il più abile della scena e portare prestigio al gruppo di cui facevo parte. Perché quando raggiungevi certi livelli, ottenevi automaticamente il rispetto degli altri e potevi persino permetterti di prenderti beffe degli altri gruppi, mandare loro saluti ironici sui file .NFO e addirittura in qualche caso prendere per il culo i cosiddetti "lamer cracker".

DLM – Ricordi quali tool software utilizzavi all'inizio, quando muovevi i primi passi nel cracking? Come sono cambiati nel tempo e quali sono quelli attuali e moderni?

All'inizio usavo il Turbo Debugger della Borland. Poi sono passato a Soft-ICE DOS/WIN, quindi Sourcer e CodeView. Non disdegnavo neppure l'uso del semplice debug tool incluso nel DOS. Poi imparai a compilare con il TASM (Turbo Assembler) e come editor usavo spesso HexView. Al giorno d'oggi, uso IDA/GHIRA come decompilatori e OllyDBG e X64Debug come debugger.

DLM – Sempre rimanendo nel novero della legalità, hai qualche storia da raccontarci di qualche impresa puramente di hackeraggio cui hai partecipato?

Ce ne sono diverse. Una delle più interessanti è quella delle centraline System75, apparati telefonici in uso negli Stati Uniti che contenevano dei modem a 1200/2400 bps. A questi apparati elettronici ci si poteva connettere e hackerare

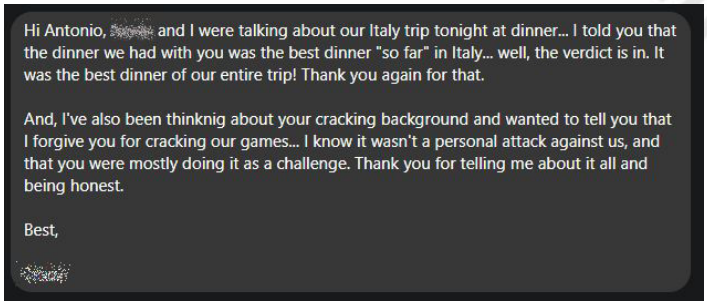


Fig. 5 - Il messaggio di perdono ricevuto dal designer Lucasfilm

(qualche volta persino con pwd standard). Se l'hacking riusciva, si potevano addirittura creare dei nuovi numeri telefonici da usare a piacimento. Si potevano anche creare dei PBX da utilizzare per chiamare altri numeri (tipicamente erano usati per effettuare chiamate long distance). In pratica si potevano effettuare chiamate locali (gratuite) e poi rimbalzare sui PBX per telefonare a numeri nazionali o esteri. La cosa divertente e davvero cool era che si potevano chiamare anche i numeri 700 per creare delle vere e proprie conference call ad accesso riservato (mediante PIN). Alla fine è così che si organizzavano le audioconferenze gratuite per coordinarci all'interno del gruppo o per chiamare gratis ovunque. La qualità delle chiamate era spesso molto buona e usavamo i System75 anche per accedere alle BBS di tutto il mondo.

DLM – Come detto nell'introduzione, di tanto in tanto partecipi ad eventi per portare e preservare la tua esperienza nella scena phreaking/craking/hacking. Noi di RMW abbiamo potuto apprezzare il tuo intervento dello scorso ottobre a Milano, soprattutto per le curiosità e per i succosissimi aneddoti che hai raccontato. L'anno precedente, a OUAS 2018 tenutosi a Roma, hai addirittura tenuto uno workshop aperto a tutti in cui hai dato informazioni più tecniche e hai sfidato gli intervenuti a craccare uno specifico gioco. Come ti è venuta quest'idea e com'è andata?

Mi è stato chiesto di partecipare a OUAS 2017 per effettuare un intervento classico ma io ho proposto una vera e propria live cracking session. In pratica, ho mostrato la sprotezione "in diretta" di un gioco (nel dettaglio si trattava di Prehistorik 2 di Eric Zmiro). In quell'occasione mi sono armato di un vecchio portatile 386 e svolgendo tutto dal vivo, inclusi i problemi d'interfacciamento della porta VGA con lo schermo ed il proiettore, ho eseguito un'analisi operativa e la realizzazione del programma crack.

Abbiamo ripetuto l'esperienza all'evento del 2018, nel quale la cracking session si è trasformata in un vero e proprio workshop a cui hanno aderito un buon numero di partecipanti. Le sessioni del workshop riguardavano i crack di Zak McKracken e di Maniac Mansion mentre l'esercizio effettivo per i partecipanti si è svolto sulla realizzazione di un crack di Loom, usando semplicemente un debugger adeguato. Credo che il workshop del 2018 sia stato un evento unico in Italia e forse anche in Europa. I risvolti pratici e gli elementi tecnici toccati durante le sessioni dai tanti partecipanti, a giudicare dai feedback ricevuti, si sono rivelati molto

interessanti ed estremamente formativi.

DLM – Torniamo al cracking duro e puro. C'è mai stata una protezione che davvero non sei riuscito a superare? E quella che, dopo averla superata, proprio ti ha stupito per la genialità di chi l'aveva ideata?

No, non ricordo di essermi mai imbattuto in qualcosa di incraccabile. A volte ci voleva più tempo per superare le protezioni costruite dai programmatori o da veri e propri esperti di protezione assunti dalle software house, ma ne sono sempre venuto a capo. :-)

Fra le protezioni più difficili da aggirare devo menzionare quelle derivate dal Protection Kit di Eric Zmiro, autore di diversi giochi per PC/DOS (fra cui Prehistorik 2). Quello di Zmiro è stato senza dubbio il miglior sistema di protezione che io abbia mai visto. Ricordo che all'epoca mi diede molto filo da torcere e scrissi un completo unwrapper per cercare di superarlo e ci sono ancora alcune parti del suo kit che sono rimaste irrisolte. Tempo fa sono stato in contatto con Eric e, anche se è rimasto sempre fermo nel suo odio contro tutti i cracker del mondo, mi ha addirittura spedito una copia originale del suo software.

DLM – Ti capita ancora di sederti al PC e provare a crackare giochi moderni? Oppure di dedicarti al re-cracking di vecchi titoli, per completarli o per renderli ancora più semplici per i giocatori?

Sì, mi capita spesso, anche se il tempo è sempre poco per dedicarsi a queste attività. Ricevo ancora tante richieste da amici su Facebook o dai vari gruppi MS-DOS. Craccare giochi moderni su PC o altre piattaforme o console? No, grazie, mi rifiuto categoricamente.

DLM – Nel tempo hai sicuramente accumulato una mole impressionante di informazioni, tecniche, know-how, ecc. Hai mai pensato che tutta questa miniera di informazioni andrebbe in qualche modo preservata? Senza rimanere impigliati nelle maglie della legge (anche se almeno in Italia dovresti godere di qualche articolo relativo alla prescrizione...), c'è modo secondo te di "digitalizzare" tutte queste informazioni?

Facendo le dovute proporzioni, paragonerei la mia abilità ed i miei skill accumulati nel tempo a quelli di un chirurgo o di un artigiano. E' difficile se non impossibile descrivere le capacità e le tecniche che si mettono in atto per superare o rimuovere la protezione di un gioco o di un programma. Ancora più difficile mettere per iscritto cose come l'intuito, la creatività o la capacità d'osservazione quando si usa un editor esadecimale o si legge il codice assembly ricavato da un debugger. Se questo fosse possibile allora si potrebbe pensare di digitalizzare la fantasia o la creatività umana, cosa che al momento trovo alquanto complicato. :-)

E' stato interessante tenere un workshop all'OUAS 2018 di Roma. In quel contesto si possono mostrare alcuni principi e metodi di base. Altre info sono difficili da trasmettere se non con la frequentazione assidua o un contatto diretto con coloro che ambiscono ad imparare. L'hacking non è esattamente una materia di studio. Sì, le tecniche possono essere classificate, ma per quello va bene uno qualunque dei volumi dedicati a queste attività.

DLM - Quali sono secondo te i riferimenti must-have presenti sulla Rete e non, che ogni buon cracker o phreaker dovrebbe consultare o tenere nella sua personale raccolta?

La Rete è piena di testi e libri che parlando di hacking e in alcuni casi anche di cracking e phreaking. Sono materie complesse e articolate, in realtà, impossibili secondo me da racchiudere e inscatolare in libri e tutorial. Io consiglio sempre a chi vuole avvicinarsi a questo mondo, di procurarsi il libro "Programming the Intel 80386" [R6] e cominciare a leggere.

3 - Outro

DLM - Un'ultima domanda: come mai hai scelto "Randall Flagg" come tuo nome di battaglia nella cracking scene?

Sono sempre stato un appassionato lettore dei libri di Stephen King. All'epoca "L'ombra dello scorpione" mi colpì molto ed uno dei suoi personaggi più oscuri, cattivi e spietati, poi diventato sinonimo del male assoluto in molti libri successivi, era proprio...

Ci sarebbe ancora moltissimo da chiedere, tantissimo da ricordare e forse non basterebbe una rubrica fissa su RMW piena di aneddoti e curiosità con tutte le storie che R.F. potrebbe ancora raccontarci. Ma per il momento ci fermiamo qui, ringraziando di cuore Antonio Mazzanti per la sua disponibilità e per la sua simpatia. E' stato davvero divertente chiacchierare e ricordare assieme a lui i vecchi tempi della scena cracking anni '90. Non dimenticate di visionare il video dell'intervento di Antonio all'ultimo Once Upon A Sprite 2019 [R5]. Ne vale la pena.

Riferimenti

[R1] RAZOR 1911 official website - https://razor1911.com/
[R2] LucasFilm Games - https://www.lucasfilm.com
[R3] Snacky e Rubicon: https://ready64.org/articoli/leggi/idart/102/rubicon-rivelato
[R4] Eric Zmiro's Best Protection Kit: http://fabulousfurlough.blogspot.com/2008/08/eric-zmiros-best-protection-kit.html
[R5] OUAS 2019: https://www.youtube.com/watch?v=wec5tt9OPN0
[R6] Programming the Intel 80386: https://archive.org/details/programmingintel00smit

The Real Ghostbusters Arcade Fangame

Intervista a **Giancarlo Schiano** e **anteprima** del gioco

Hot Topic!!!

di Carlo N. Del Mar Pirazzini

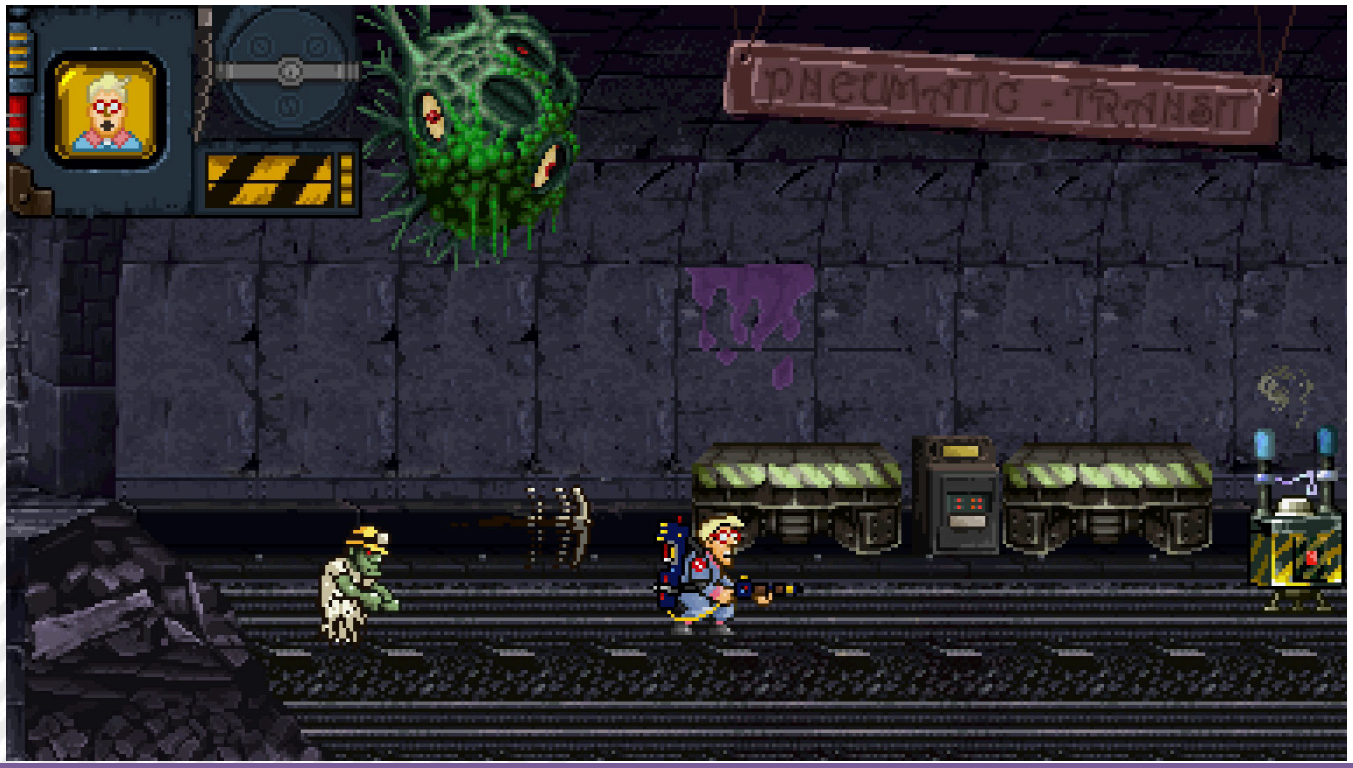
"La cosa è una forza dominante del mondo sotterraneo a capo di una legione di demoni malvagi. Lo scopo della cosa è di invadere e conquistare il mondo dei viventi."
Dottor Egon Spengler

Qualche giorno fa mi appare una notifica su Fb che mi lascia di stucco. Nella pagina italiana dedicata ai Ghostbusters si parla della futura uscita di "The Real Ghostbusters arcade fangame".
Mi precipito per vedere di cosa si tratta e lo stupore si tramuta in "Devo provare questo gioco!!!!!".
Un mix tra Metal Slug e la bellissima serie televisiva della Columbia Pictures. Eccezionale.
Visiono le foto presenti e un piccolo video e poi scrivo subito agli sviluppatori.
Ed eccoci qui con una nuova intervista con Giancarlo Schiano, un pezzo del cuore di questo gioco in via di sviluppo (l'altra parte del cuore scoprirete chi è, ndN).
Una persona squisita con cui si può parlare di tutto, ma soprattutto un grande fan degli acchiappa-fantasmi e della serie tv.
Vi lascio alle domande e alle foto del gioco.

Nith - Ciao, presentati e presenta il tuo progetto ai nostri lettori. Chi sei e da come è nato lo sviluppo di questo gioco.
Giancarlo - Prima di tutto grazie per questa intervista che per me è un onore, un piacere fare con voi e presentare il progetto ai vostri lettori. Mi chiamo Giancarlo, ho 33 anni, abito a Ravenna e da sempre, da quando li ho conosciuti, da che ho memoria sono un fan pazzo dei The Real Ghostbusters, più in generale del mondo degli acchiappa fantasmi ma in particolare ho seguito con grande passione la saga televisiva degli eroi di New York. Mi hanno sempre fatto compagnia nei pomeriggi da ragazzino, mi facevano sognare tantissimo perché espandevano il mondo degli acchiappa fantasmi al di là del film, dando avventure nuove e soprattutto ti davano forse qualche insegnamento, perché per noi ragazzi dell'epoca i cartoni animati erano un veicolo d'istruzione. E per me tutto questo è stato il motivo del perché ho deciso di creare il videogioco.

Nith - Da cosa è partita la scintilla di creare un arcade games dei GB sullo stile di Metal Slug?
Giancarlo - La scelta stilistica di unire un cult game come Metal Slug a The Real Ghostbusters è nata dal fatto che tanti anni fa, durante i miei dodici anni d'età passai l'estate con un mio cuginetto e lui inventò una scusa incredibile per attirare la mia attenzione e passione per la serie tv. Si inventò di avere un programma con il quale era possibile modificare i personaggi e i nemici di un gioco a tuo piacimento. Io già mi immaginavo di creare Egon, di sparare con il flusso contro i fantasmi, contro gli zombi, le mummie… poi, ovviamente, si rivelò una "balla" colossale e questo mi ha lasciato negli anni un piccolo/grande vuoto dentro. Da fan mi ha lasciato questo vuoto e l'ho voluto, anni dopo, colmare io stesso. Ovviamente noi abbiamo preso ispirazione dal design, ma abbiamo ricreato interamente tutto quello che vedrete nel gioco finito.

Nith - Con che sistema state sviluppando il gioco. In



quanti siete nel team di sviluppo.
Giancarlo - Voglio risponderti che siamo in due. Soltanto in due. Due persone che da due anni si dedicano nel loro tempo libero alla creazione di questo gioco. Due persone ovvero, Morgana Zaltron che è la mia compagna ed io. Lei è lo scripter che crea i codici ed io sono il designer, occupandomi della grafica in totale. Stiamo sviluppando il gioco con Game Maker Studio 2. Ringrazio sempre Morgana per il lavoro che stiamo portando a termine. Il gioco allo stato attuale è tra l'80 e il 90% della realizzazione finale. Stiamo lavorando sull'ottimizzazione finale dei codici.

Nith - che piattaforme pensate di prendere in considerazione e che tipo di target di giocatori avete in mente (ovvero il gioco avrà un'impronta arcade vecchio stile o sarà più simile ai prodotti moderni in termine di giocabilità)
Giancarlo - Il gioco sarà interamente "VECCHIO STILE". Avete presente il primo Metal Slug ad inizio anni 90 dove spendevi un capitale per poterlo finire?? Bene sarà esattamente così. Non ci saranno effetti speciali, non ci sarà nulla della dinamica dei giochi attuali. E' tutto puramente in stile "retro-arcade". Classico! In alcuni punti anche "macchinoso nel giocarlo", ma è il bello degli arcade storici, dove avevi delle difficoltà intrinseche durante la partita che non ti rendevano facile la vita e quello creava la sfida. La voglia di vedere ancora di più, di migliorarsi. Questa è stata la nostra idea, il nostro imprinting per creare il gioco.
Per il momento non abbiamo in mente di farlo su altre piattaforme (ricordiamo il gioco girerà su sistemi Windows, ndN) perché ancora dobbiamo chiudere il gioco su Pc. Quindi il nostro primo passo è chiuderlo e renderlo GRATUITO per tutti. Poi se riusciamo a convertirlo e portarlo su altre piattaforme e device per noi sarà un plus che ci riserviamo di fare ma che al momento non garantiamo.

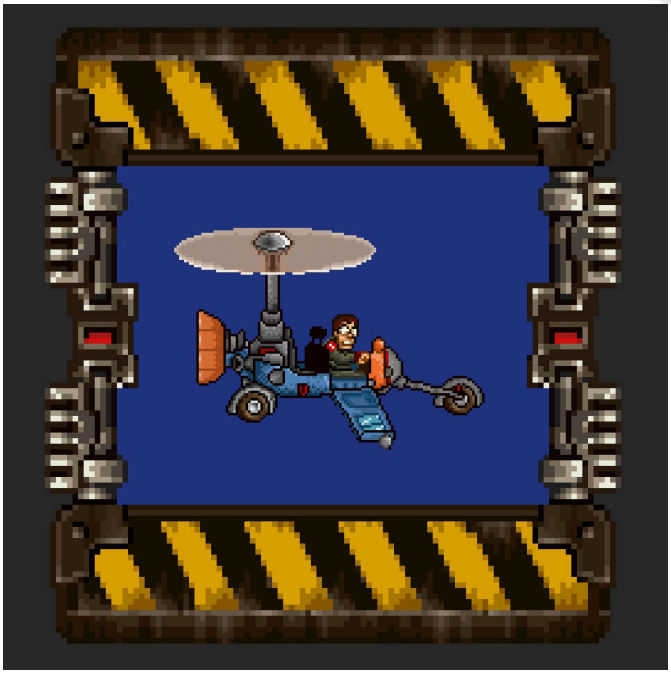
Nith -- Previsioni per l'uscità? Che piattaforme di divulgazione volete utilizzare?
Giancarlo - In realtà non penso che noi spenderemo neanche un centesimo per fare pubblicità! Mi spiego meglio. E' un gioco creato da dei fan, io e la mia compagna, per dei fan. Noi ci auguriamo che il fandom legato alla serie sia così propositivo da fare lui stesso da mezzo di divulgazione, da pubblicità per questo progetto. Noi non chiediamo nulla. Lo doniamo. Stiamo donando assieme al gioco tre anni della nostra vita e della nostra passione per questo gioco e per The Real Ghostbusters senza avere nulla in cambio se non dei fan entusiasti. E non c'è nulla di meglio di un fan entusiasta come mezzo di divulgazione! Per quanto riguarda l'uscita, ci auguriamo di farlo uscire completo per metà del 2021. Considera che ci stiamo lavorando dal 2018 e per noi sta diventando una necessità chiuderlo. E per chiuderlo intendo renderlo un prodotto giocabile quanto prima possibile.

Nith - Raccontaci il gioco in se. Ovvero il gameplay.
Giancarlo - Come ti ho detto ricorda Metal Slug, quindi immaginatevi il gioco con tematiche The Real Ghostbusters. L'intero gioco è un gigantesco cameo di ogni cosa vista all'interno della serie tv e di tutti i giocattoli della Kenner che uscirono a seguito della serie tv. Inoltre abbiamo nascosto numerosi Easter Egg di film che anno fatto la storia del cinema degli anni 80/90. Lascio a voi libera l'immaginazione. Per concludere la questione Gameplay, ti svelo qualcosa... faccio un po' di spoiler. Ci saranno 5 livelli più un extra selezionabile dal menù chiamato ECU (ovvero l'ecto contenitore) che servirà ai giocatori per visitare al suo interno tutte le entità che hanno catturato all'interno del gioco. Una specie di Wall of Fame dove il personaggio potrà "interagire" con le entità che avrà sconfitto all'interno del gioco (e non solo, ndN), proprio come succede in una delle puntate del cartone animato, quella dedicata al racconto di Natale di E. Scrooge di Charles Dickens (a questo punto dell'intervista Giancarlo sfodera la sua conoscenza di The Real Ghostbusters raccontandomi perfettamente l'episodio. Questo ragazzo è una vera enciclopedia degli acchiappafantasmi! NdN). Ogni livello ha un titolo differente. Ogni livello è dedicato

ad un tema particolare.
Il primo è dedicato ad un nemico particolare di Egon, i fan capiranno e saranno entusiasti! Il secondo è dedicato alla Ecto 1, sullo sfondo di New York city. Qui troveremo i nemici della serie animata e dei giocattoli Kenner.
Il terzo livello dedicato alla Ecto 2, ovvero in volo nei cieli di New York.
Il quarto livello sarà dedicato ad una villa infestata e ci porterà nelle fogne di New York, già viste nel film Ghostbusters 2. E ci ritroveremo alla fine di questo livello ad affrontare ben quattro avversari all'interno della dimensione fantasma!
Il quinto ed ultimo livello… il livello finale. Ma non aggiungo altro.

Nith -Avete una campagna per potervi aiutare nel supporto?
Giancarlo - Il problema qui è molto gravoso. Sappiamo che stiamo andando ad interagire con un marchi registrati. Un marchio che appartiene alla Sony e a Ghost corps (società che detiene il controllo del marchio e del franchise ad esso legato, NdN) e noi non possiamo chiedere fondi o un crowfunding per sostene il progetto, ma possiamo lasciar aperto a donazioni l'opportunità che ogni fans può fare per supportare il progetto. Ma è solo una donazione che andrebbe ad aiutare per sostenere le spese vive per lo sviluppo. In questo senso non abbiamo e non voglio creare una campagna in questo senso. Vogliamo che i nostri fans siano allietati dal prodotto e dal suo essere gratuito. Sperando che riporti i fans ad un momento felice, visto il periodo terribile che stiamo vivendo.

Nith - Grazie delle risposte. Se vuoi aggiungere qualcosa, sentiti libero di dire quello che vuoi.
Giancarlo - Intanto voglio ringraziare Morgana, senza di lei il progetto non sarebbe potuto andare avanti. Voglio ringraziare i fans, veri supporter del progetto. In meno di 4 giorni abbiamo avuto quasi 2000 followers, una cosa inaspettata e questo riprova il fatto che il passa parola è più importante della pubblicità roboante a pagamento.
Ringrazio voi di Retromagazine World che ci avete dato lo spazio per parlarne, perché per il sottoscritto è bellissimo condividere gratuitamente una cosa che ha fatto parte della mia infanzia e che, in parte, mi ha reso quello che sono oggi.
Detto questo concludo con una mia riflessione personale: Visto il seguito che un piccolo progetto come questo ha avuto e sta avendo su tanti appassionate della serie e non appassionati mi viene da dire che non sono importanti i budget, non sono importanti i soldi investiti ma lo sono il cuore e l'amore che la gente ci mette. Vedo troppo spesso giochi con altissimi livelli qualitativi a livello grafico ma con una povertà di trama e gameplay abissale (e questa è pura verità, da stampare su tavole immortali. NdN). Ritrovare uno stile di gioco vecchissimo, per certi versi macchinoso, forse lontano anni luce da quello che si trova oggi nelle nostre console e nei nostri computer ma comunque così apprezzato ci fa capire che, alla fine non è la grafica, ma l'amore, la passione, di metterci ogni goccia di fatica per rendere tutto perfetto.



Ad esempio il gioco di Ghostbusters uscito nel 2012, ha avuto un grandissimo successo perché si vedeva una continuità, un amore spasmodico per il marchio e per i personaggi. Nulla fu lasciato al caso.
L'opportunità su questo brand, su The Real Ghostbusters sono infinite, ci auguriamo noi fan, io per primo, che venga ripreso. Che si possa espandere e che possa avere lo stesso impatto che ha dato a noi in passato alle nuove generazioni.

Concludo e ti ringrazio. Voglio fare un piccolo annuncio, oltre a questo progetto ludico, abbiamo anche il film "THE REAL GHOSTBUSTERS – IL FILM", purtroppo fermo causa COVID, ma siamo pronti per ripartire e portarlo avanti. Anche in questo caso troverete tutte le informazioni nella pagina **The Real Ghostbusters Live Action**. Io partecipo attivamente nel ruolo di Egon. Ancora grazie a tutti voi e chissà, se i fan apprezzeranno, in futuro il progetto potrebbe avere anche un seguito. Farlo diventare un lavoro? Per ora è difficile, ma chissà!

Ringraziamo Giancarlo e Morgana per il lavoro che stanno svolgendo e per questa esauriente intervista.

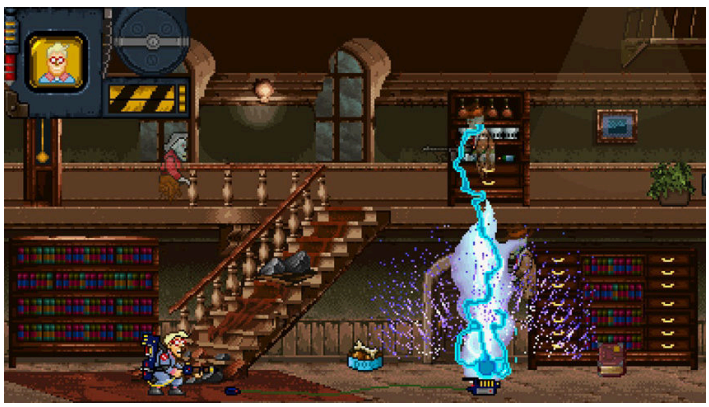
Vi invitiamo a seguire il progetto su facebook alla pagina **The Real Ghostbusters Arcade fangame** e vi lasciamo alla bellissime immagini del gioco.
Personalmente non vedo l'ora di mettere le mani sul gioco e testarlo in quanto fan dei film e della serie tv.
Vi lascio con una citazione sempre della serie televisiva.

"Ricordatevi se voi non avete paura, lui non può farvi del male."
Winston Zeddmore


ECTO-1
F.D.N.Y.
AMBULANCE


ECTO-1


CITY OF DAMNED



Metamorphosis (ZX Spectrum 48kb/128kb) - PREVIEW

L'Evoluzione fa un passo avanti

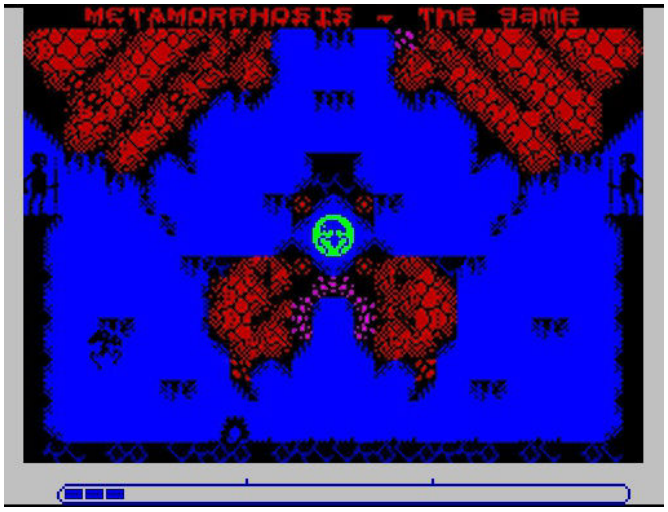
di Starfox Mulder

Non capita spesso di parlare di retrogaming e poter avere delle novità in esclusiva. Se un hardware è retro, per definizione, è fuori produzione e raramente riceve release di nuovi videogames. Sebbene poi alcune macchine godano di una rinnovata giovinezza, siamo abituati a provare i nuovi giochi una volta terminati e pubblicati. Questa volta si fa un'eccezione.

Leonardo Vettori (grafica e gameplay) è uno di noi, del Team di RM, e non è nuovo al mondo retroludico anche come sviluppatore. Vi ricordate Nucleo 447/448? Recuperatevi i numeri 16 e 17 della rivista (tanto son sempre in archivio scaricabili gratuitamente). Coadiuvato da **Kees Van Oss** come programmatore, **Pedro Pimenta** e **Mario Fanciulli** alle musiche, il team delle meraviglie si è lanciato sull'idea di un nuovo platform per **ZX Spectrum**, dalla grafica eccezionale e il concept innovativo.

In Metamorphosis tutto gira intorno al concetto di evoluzione. Inizieremo il gioco con un essere dalle sembianze simili ad un aracnide ed una barra di energia molto limitata.

Potremo spostarci nelle due direzioni (destra e sinistra), saltare o colpire i nemici con un particolare attacco a sputo. Lo scopo non è ucciderli, ma farli regredire allo stato di esistenza precedente, indebolendoli di colpo in colpo finché non saranno degenerati a vermi e saremo in grado di mangiarli (rigenerando così la nostra energy bar). Inizialmente il numero massimo di punti energia a nostra



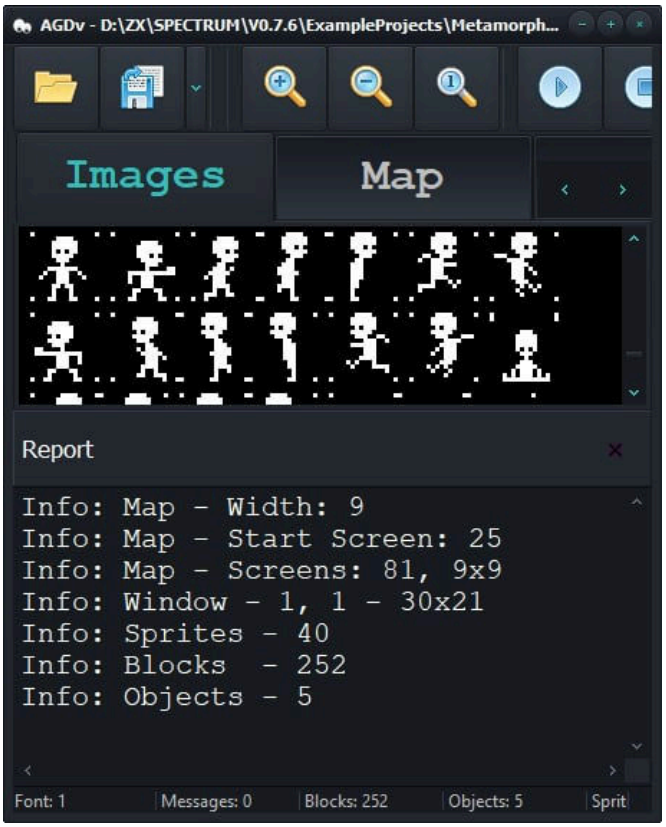
disposizione sarà molto limitato ma con l'avanzare del gioco potremo recuperare degli oggetti che, se posti nel raggio di energia presente nella stanza centrale, ci aumenteranno il valore massimo della barra di energia, oltre ad avvinarci alla conclusione dello stage. Passata una certa soglia ecco che evolveremo ad un mix umano-aracnide, "e questa non è nemmeno la mia forma finale" (avrebbe detto un noto Villain), consci però che ogni colpo subito potrebbe portarci a regredire alla forma precedente.

Perchè evolvere?

Innanzi tutto perché è lo scopo del gioco arrivare a diventare dei bambini veri (oggi mi han preso bene le citazioni) ma

in termini di puro gameplay l'evoluzione ci porterà a dei vantaggi concreti che non sto a descrivervi perché...ho giocato (e finito) solo la versione Demo.

Nella versione di Metamorphosis messa a mia disposizione e terminata su real hardware si può concludere solo il primo dei tre livelli che saranno poi presenti nella versione finale, quindi il mio giudizio è concretamente limitato a quello che vale "livello tutorial". I nemici sono programmati intelligentemente e non si limitano ad attaccarci ma si combattono anche tra loro, alle volte evolvendo sotto i nostri occhi dopo aver mangiato un avversario e diventando di rimando più minacciosi.



Questo elemento non ricordo di averlo mai visto introdotto in un gioco per Spectrum e mi ha fatto un enorme piacere. L'impatto grafico poi, come accennato in precedenza, è tra le cose più incredibili che mi sia mai capitato di vedere sulla macchina in questione. Le musiche sono un eccellente accompagnamento e per finire il gameplay non mostra il fianco a nessuna concreta critica...salvo forse quella di non essere un sistema di gioco originale, ma come si faccia a creare qualcosa di rivoluzionario per una macchina che ha sul groppone 38 anni me lo dovete spiegare.

Nelle sue dinamiche collaudate, Metamorphosis da comunque il meglio, portandoci ad alternare scenari esplorativi a stage di puro combattimento, sbloccabili con chiavi apposite da scovare lungo i livelli. Tutto con schermate fisse, tutto capace di girare perfettamente su un 48Kb (ma se volete le musiche vi serve un 128Kb).

Che dire quindi in conclusione?

Leonardo è un amico e mi ha chiesto di essere spietato, ma visto il prodotto che mi è stato messo a disposizione è davvero complesso trovargli un difetto. Troppo facile? Sì, palesemente, ma si tratta del primo stage, sono verosimilmente certo alzeranno l'asticella della competizione nei prossimi due.

Quello che di certo non cambierà è l'effetto sense of wonder dovuto ad un'ambientazione originale e curata, capace di calarci in un delirio primitivo-gygeriano come non se ne erano mai visti.

A mani bassi il gioco che mi ha creato più attesa (A.k.a. Hype) dell'ultimo periodo.

LOOM

Anno: 1990
Editore: Lucasfilm
Sviluppatore: Lucasfilm
Genere: Avventura grafica
Piattaforma: Amiga

"Un Viaggio Musicale"

Articolo originale Simone Battaglioni, rieditato per RMW da Gianluca Girelli

Loom è uno storico videogioco della Lucasfilm Games uscito negli anni 90 su molte piattaforme. La recensione si riferisce alla versione Amiga.

Sinossi
Con un atmosfera cupa da fantasy medievale, la storia di Loom racconta il viaggio di un ragazzo appartenente alla "Gilda dei Tessitori" alla scoperta di sé stesso e di cosa è accaduto alla sua gente. Si ritroverà ben presto coinvolto, suo malgrado, nel salvare il destino del regno.

Svolgimento
Nel gioco interpretiamo Bobbin Threadbare, e il nostro viaggio inizia apprendendo le prime rudimentali nozioni sul come applicare l'arte della tessitura. Una volta lasciata l'isola di Loom e approdati al continente incontreremo le altre Gilde, tra cui quelle di Vetrai, Pastori, Fabbri...

Di alcune sentiremo soltanto il nome, altre invece ci coinvolgeranno maggiormente. Faremo anche brutti incontri tra cui un Drago femmina poco focosa... Scopriremo svariati segreti nascosti ed esploreremo città. Per finire fermeremo il Male e il suo malefico piano.

Contenuto delle varie edizioni
Come si usava negli anni 90, Loom veniva venduto in scatola cartonata con raffigurazioni e stile medievale fantasy, con al suo interno il "Book of Patterns" dove annotare le varie note da poter ripetere nel gioco(spiegazione più avanti).

Tali sequenze di note non erano sempre le stesse per poter aumentare la longevità del titolo.

Il gioco era protetto contro la copia con dei codici da inserire in momenti predefiniti, tratti dal manuale, non fotocopiabili in quanto scritti in blu con righe rosse...

Un Audio-Drama su musicassetta (in inglese) raccontava l'antefatto del gioco.

Inoltre, era accluso un foglietto con le informazioni tecniche in base alla versione acquistata.

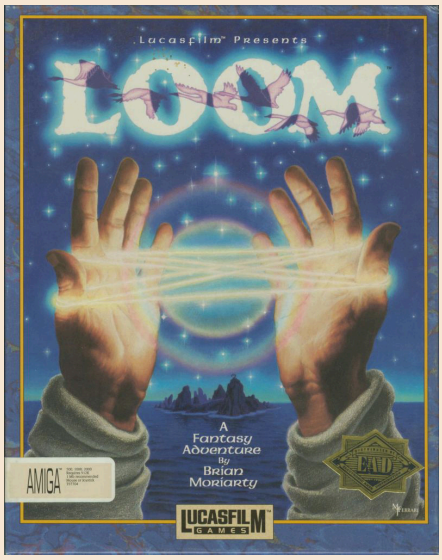
Spiegazione tecnica
Loom è stato il quarto gioco a usare il motore di gioco SCUMM scritto da Brian Moriarty, con la colonna sonora sulle musiche di Tchaikovsky e i fondali grafici realizzati da Mark Ferrari.

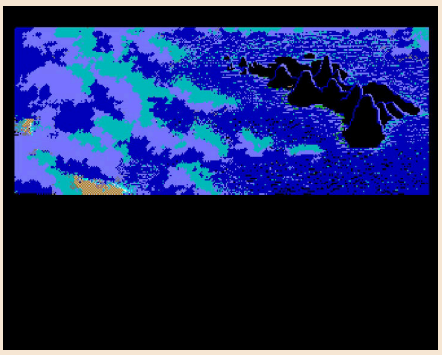
Sviluppato per primo su PC EGA fu poi portato su diverse piattaforme tra cui Amiga, Atari-ST, Macintosh.
Loom è un'avventura grafica, ma diversa da tutte le altre: mentre nelle altre si utilizzano le azioni come Aprire, Chiudere, Parlare e via discorrendo, qui si utilizzano le note per creare magie dette "Trame" con cui si possono compiere azioni tipo: Tingere, Aprire, Riempire ecc.

L'arrangiamento musicale ben si adatta all'atmosfera fantasy, con musiche che si ripetono nei momenti giusti. Tale meccanica, che oggigiorno sembra scontata, è in realtà piuttosto difficile da realizzare con efficacia, e la frustrazione di non avere musica sincronizzata col gioco portò in seguito i programmatori della Lucasfilm Games a sviluppare il sistema iMuse (Iteractive Music Streaming Engine).

Come aggiunta, gli effetti sonori uditi durante le varie location accompagnano il giocatore senza fargli dimenticare dove si trova in quel momento.

Il comparto grafico sviluppato su PC





EGA sfrutta sapientemente il dithering per poter quindi simulare a schermo un maggior numero di colori, per cui fino a quando non uscì la versione VGA in 256 colori con tanto di doppiaggio su CD, versione EGA fu mantenuta in tutte le piattaforme precedentemente menzionate.

Come ultime versioni uscirono quella per FM-Towns e PC-Engine TG16.
A discapito di quello che si può pensare, Loom ebbe un discreto successo per l'epoca: Brian Moriarty dichiarò che era intenzionato a scrivere una trilogia in cui il secondo capitolo avrebbe dovuto chiamarsi "Forge" e il terzo "The Fold". In seguito accantonò l'idea per dedicarsi ad altri progetti e non ci fu nessun altro che pensò o se la sentì di proseguire.

Conclusioni Finali
Loom è davvero un videogioco particolare, non troppo enigmatico o complesso, ma che sa dosare con cura e invogliare il giocatore alla scoperta di quello che sa offrire. Ogni passo che si compie viene vissuto come un traguardo personale. Quando uscì all'epoca fu innovativo e molto apprezzato, tanto da essere ancora oggi considerato un Cult tra tutti i nerd...

Curiosità
E' da sottolineare il fatto che Loom uscì in Inglese, Francese, Spagnolo e Tedesco. Nel corso degli anni svariati utenti si sono cimentati nella traduzione Italiana del videogioco.
L'ultima uscita in ordine di tempo è a opera di Simone "SimonPPC" Battaglioni, redattore del presente articolo, che ha cercato di mantenere inalterata l'opera originale dell'autore.

Nel momento in cui scriviamo era al lavoro sulla versione VGA, ma ci ha confidato che sta pensando di rimettere mano alle precendenti versioni da lui adattate e cioè PC EGA, Amiga, Atari ST e Macintosh.

Come ciliegina sulla torta, tra le varie foto incluse in questa recensione potete ammirare la tomba della madre di Bobbin, che essendo parte grafica ha richiesto parecchio tempo per capire come modificarla.

Simone è però riuscito anche in questa impresa grazie alla collaborazione con un utente straniero.

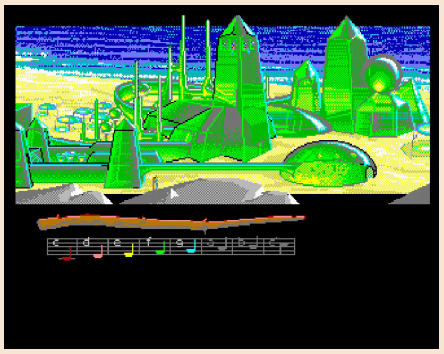
Prima di lasciarvi, vogliamo proporvi questo video del longplay di Loom fatto da Simone Battaglioni insieme agli amici di Retro Edicola Videoludica: https://www.youtube.com/watch?v=DmSyp_0ETV0

di **Gianluca Girelli e Simone Battaglioni**

GIUDIZIO FINALE

» Giocabilità 85%
Grazie ai tre livelli di difficoltà è ben bilanciato e gli enigmi non sono così complessi da non essere superati.
Una delle più belle avventure mai realizzate.

» Longevità 90%
Non vi stancherete di giocarlo per scoprire se avete lasciato indietro qualche cosa o solo per il gusto di riascoltare le melodie.



HOLIDAY LEMMINGS

Data di pubblicazione: 1991-1994
Piattaforma: Commodore Amiga (e moltissime altre)
Genere: puzzle/platform
Versione recensita: Holiday Lemmings

Se oggi la qualità grafica di un titolo nextgen si misura dalla complessità poligonale e dalla presenza massiccia di ray tracing quando il Commodore Amiga ancora dominava il mondo dei video giochi i giocatori più esigenti misuravano la qualità artistica di un gioco dalla qualità della pixel art e la serie Lemmings dava davvero il meglio di sè visto che i roditori in questione erano stati sapientemente realizzati dalla software house DMA Design utilizzando proprio una manciata di minuscoli pixel! Lemmings era quel tipo di puzzle che ti mettevi a giocare magari dopo una furiosa battaglia spaziale a Project X o dopo un intero gran premio al poligonale Grand Prix di Geoff Crammond perdendo le ore nel complicato e delizioso tentativo di portare la maggior parte dei simpatici esserini verso l'uscita di una serie di livelli sempre più complicati e impegnativi. Come molti di voi sapranno Lemmings è una serie di videogiochi che ha saputo anche in tempi recenti intrattenere milioni di giocatori anche su tablet e smartphone ma nata all'inizio degli anni '90 proprio sulla ammiraglia di casa Commodore.

I livelli di gioco erano per la tribù di Lemmings una immensa fatality nel senso che se il giocatore non avesse dato loro le giuste abilità nel momento più opportuno sarebbero potuti restare lì a camminare avanti e indietro all'infinito oppure sarebbero morti cadendo in massa da un'altezza che il loro esile fisico non avrebbe potuto sopportare. Insomma in Lemming gli spiaccicamenti erano all'ordine del giorno!
L'interfaccia di gioco era composta da un livello a tempo a scorrimento multidirezionale, da una piccola mappa e da una serie di icone che indicavano le abilità disponibili variabili di livello in livello. Il giocatore doveva valutare la situazione in base alla morfologia del livello e alle abilità disponibili per completare la missione. In genere l'obbiettivo era far giungere una certa percentuale di Lemmings al portale di uscita ma questa percentuale poteva essere impostata dai programmatori anche al 100%!
Una impresa non sempre di facile completamento visto che l'intelligenza artificiale dei simpatici esserini si limitava a farli camminare costantemente in una certa direzione invertendola in caso di contatto con ostacoli non superabili con la loro normale camminata.

Utilizzando però le abilità cliccando il Lemming giusto al momento giusto (cosa non sempre facile data la loro dimensione) potevamo far compiere loro azioni anche piuttosto complesse come per esempio costruire ponti, scavare il suolo e le rocce, utilizzare ombrelli come paracadute, bloccare l'avanzata degli altri membri della tribù (in modo da poter procedere senza troppe perdite) o addirittura suicidarsi!
Già, in Lemmings a volte anche il sacrificio era indispensabile per il bene di tutta la tribù ed ecco come un divertente puzzle game che nella sua versione Holiday ci deliziava con deliziosi scenari natalizi (ma non perdetevi i capitoli principali) aveva saputo conquistarsi una posizione speciale nel cuore di moltissimi giocatori di quegli anni e con le nuove vesioni mobile ancora oggi! Oh No!

di **Flavio Soldani**

GIUDIZIO FINALE

» Giocabilità 90%
Lemmings è strategia, puzzle e rompicapo allo stato puro. Se vi lascerete conquistare non lo mollerete tanto presto !

» Longevità 96%
I primi e forse più appaganti capitoli nati su Amiga sono disponibili ancora oggi pronti da emulare su qualsiasi tipo di piattaforma comprese console a 8 e 16 bit. Disponibili anche versioni riadattate per piattaforme mobile.

WEIRD DREAMS

Anno: 1989
Editore: Rainbird
Sviluppatore: Rainbird
Genere: Horror
Piattaforma: Commodore 64

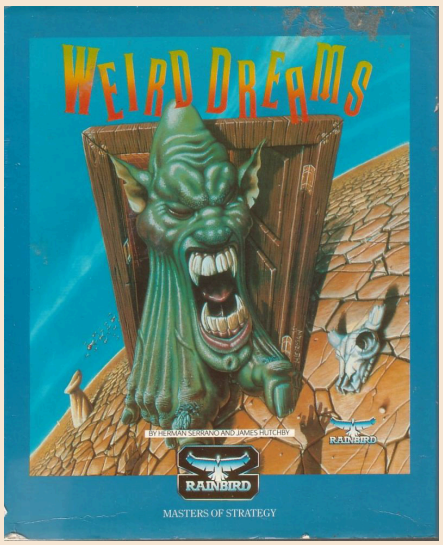
Cari sognatori, a chi di voi succede al risveglio di cercare di ricordarsi in maniera dettagliata i sogni appena fatti? E di sicuro molti di voi, me compreso, sono poi andati alla ricerca sui siti di interpretazione dei sogni per avere qualche idea dei simboli onirici e del significato generale, soprattutto se i sogni erano in realtà veri incubi. Uno dei sogni più ricorrenti, che almeno una volta nella vita capita a tutti, è quello in cui volate sopra i cieli della nostra città come un uccellino. Weird Dreams ci permetterà di rivivere alcuni di questi strani sogni davanti allo schermo del PC o dello smartphone. Il protagonista dal pigiama a scacchi di questo gioco, infatti, si ritroverà a vivere in questi bizzarri prodotti della mente a causa del sonno indotto da un'anestesia subita nella sala operatoria di un ospedale.

Tutto comincia in una schermata piuttosto confusa, dove è difficile anche capire cosa si deve fare. Una volta capito il meccanismo di gioco, ci ritroveremo in tante altre schermate fisse con tanti nuovi mondi, tra cui un luna park e molti nemici ed enigmi da affrontare, come api giganti e bambine armate di coltelli da cucina. Il mondo dei sogni non è solo singolare ma anche pieno zeppo di misteri. E questo gioco lo rappresenta quasi alla perfezione, grazie alle strane ma bellissime ambientazioni. Naturalmente la risoluzione degli enigmi è essenziale per procedere nel gioco. La giocabilità presenta un sistema un po' lento e la risposta ai comandi tarda un po' ad arrivare soprattutto quando bisogna voltarsi per fuggire da un nemico. Una delle motivazioni più forti per proseguire a giocare (e per portarlo a termine), nonostante la difficoltà intrinseca del gameplay, è la qualità del tema musicale, diverso per ogni schermata. L'uso del chip sonoro è davvero magistrale e meriterebbe la pubblicazione della colonna sonora!

Il gioco non ottenne molta eco sulla stampa quando uscì, anche se a dire il vero, sulle riviste specializzate fu giudicato con buone recensioni. Probabilmente passò un po' in sordina a causa del concomitante successo ottenuto dai tanti smash-hits dell'epoca.
Di conseguenza sparì velocemente dalle classifiche e dai riflettori della stampa e del passaparola. Inoltre pochi negozi lo avevano sugli scaffali o in vetrina. Weird Dreams fu poi convertito anche per i computer a 16 bit, riscuotendo molto più successo grazie alle maggiori capacità grafiche e sonore di macchine come Atari ST e Amiga. Ma in tutte le versioni, oltre alla colonna sonora, sono gli enigmi proposti ad invogliarci al caricamento e a procedere attraverso le varie schermate di gioco. Nella versione C64, consiglio vivamente la versione su disco per maggiore comodità e per evitare caricamenti continui, dato che non faticheremo a perdere le cinque vite a disposizione (difficile averla vinta contro un'ape gigantesca e un'inquietante bambina armata di coltello da cucina).

Dopo la perdita di una vita si potrà vedere una scena della situazione fuori dal sogno, ossia i chirurghi che ci stanno operando. I loro sguardi non promettere nulla di buono circa l'esito dell'intervento. Sono forse loro gli autori di tutto questo estraniamento che per noi significa vagare nell'incubo del gioco? Non nego che ad alcuni il gioco potrebbe risultare un po' inquietante data la suggestione e l'atmosfera generale, ma vale comunque e sempre la pena di rispolverarlo, farci un giro e conservarlo nella vostra collezione come una reliquia. Riservategli le dovute attenzioni, come ho fatto io in questo articolo, poiché secondo me meritava e merita qualcosa in più in termini di apprezzamento da parte dei giocatori.
Come sempre auguro a tutti voi di passare delle serene feste natalizie nonostante queste limitazioni inevitabili a cui siamo costretti da quasi un anno ormai. Consoliamoci con RetroMagazine World ed i nostri amati retrogame!

di **Daniele Brahimi**



GIUDIZIO FINALE

» Giocabilità 60%
I comandi appaiono un po' lenti e ritardano l'azione e la nostra reazione ai pericoli.

» Longevità 75%
La colonna sonora e gli enigmi contribuiscono ad allungare la vita del gioco.

MIGHTY FINAL FIGHT

Anno: 1983
Editore: Nintendo
Sviluppatore: Capcom
Genere: Beat em up
Piattaforma: Nintendo Nes

Final Fight è un classico arcade amato dalla comunità videoludica da quando è uscito nelle sale giochi nel 1989.
È stato portato su tantissime piattaforme tra cui lo SNES e ha anche ricevuto due sequel sul sistema, ma forse non tutti sanno che è stato anche convertito nel 1993 su NES?

Nel luglio di quell'anno infatti Capcom pubblico Mighty Final Fight sul 8 bit Nintendo, una versione più "spensierata" di Final Fight con un colorato sitle chibi super deformed.
E' un porting classico dell'arcade originale e segue la stessa trama. Tutto si svolge a Metro città, metropoli invasa dalla ciminilità a causa della Mad Gear Gang.

Il leader della banda criminale si è innamorato di Jessica, figlia del sindaco di Metro City, Haggar.
Il sindaco Haggar riceve una telefonata nel mezzo di una sessione di allenamento con Cody, il fidanzato di Jessica e il compagno di allenamento di Cody, Guy. Il cattivo dall'altra parte della linea informa Haggar che Jessica è stata rapita e sarà costretta a sposare il leader della Mad Gear Gang. Senza un momento da perdere, Haggar, Cody e Guy entrano in azione e si preparano ad affrontare la Mad Gear Gang una volta per tutte per riportare Jessica a casa sana e salva.

Come ci si potrebbe aspettare, Mighty Final Fight è un picchiaduro a scorrimento laterale. A differenza di FINAL FIGHT classico, questa versione è "only one player".

L'assenza del secondo giocatore purtroppo fa perdere al gioco il titolo di più grande picchiaduro a 8 bit. Anche in questa versione possiamo selezionare tutti e tre gli eroi principali Cody, Guy e Haggar. Tutti hanno le loro abilità uniche e una serie di mosse differenti.

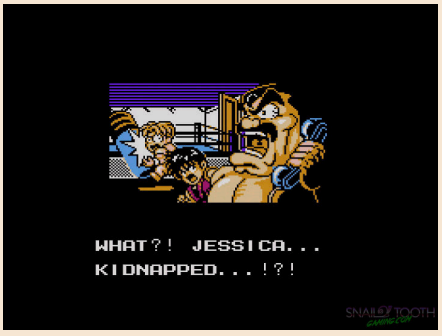
Questa versione per NES si distingue anche dagli altri giochi della serie introducendo un sistema di livelli per la progressione del personaggio. Il giocatore riceve punti esperienza per aver sconfitto un nemico e la quantità di esperienza che ottieni dipende dalla mossa con cui finisci il tuo avversario, le mosse più complicate ti fanno guadagnare più punti.

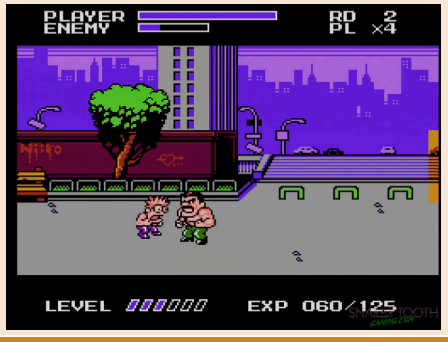
Una volta guadagnata abbastanza esperienza, il tuo personaggio salirà di livello e sbloccherà nuove tecniche di combattimento. Il gioco è abbastanza generoso da dare al giocatore 6 vite per partita. Ci sono anche fasi bonus durante il gioco in cui puoi guadagnare vite extra, potenziamenti della salute, armi specifiche del giocatore e persino continui extra. Anche con tutte queste vite e power-up, il gioco non è una passeggiata nel parco.

Ha una discreta quantità di sfide, quanto basta per tenerti impegnato ma non abbastanza per finire nella stessa categoria di giochi come Ninja Gaiden. All'inizio può sembrare troppo difficile, ma l'eccellente controllo è facilmente padroneggiato con la pratica e ti dà la possibilità di vedere il gioco fino alla fine, proprio come ho fatto nel video qui sotto. Nel complesso, il gameplay è eccellente. È fluido, reattivo, soddisfacente e, soprattutto, è molto divertente.

La grafica di questo gioco è superba. Un ottimo esempio di ciò di cui è capace il NES. L'animazione dello sprite sono alcune delle mie preferite di sempre sul piccolo Nintendo. Adoro il modo in cui reagiscono i nemici all'essere colpiti e il modo in cui i loro volti esprimono lo shock dopo essere stati presi a pugni.

Gli sfondi sono ben dettagliati e colorati, e mi piace la direzione artistica Super Deformed che gli sviluppatori hanno





deciso di adottare per questo titolo.

Unico appunto è il farfallio lampeggiante di cui soffrono gli sprites. L'unità di elaborazione delle immagini nel NES è in grado di visualizzare solo otto sprite per linea di scansione alla volta. I modelli dei personaggi sono spesso costituiti da più sprite e quando sullo schermo sono presenti troppi sprite contemporaneamente, l'hardware deve dare la priorità a ciò che deve essere mostrato.

Ciò si traduce nello sfarfallio degli sprite che è una caratteristica di tanti fantastici giochi NES. Questo accade spesso in Mighty Final Fight.

Mi ha infastidito un po 'quando sono entrato in questo gioco per la prima volta, ma dopo averlo giocato per un po' ho smesso di notarlo e mi ha lasciato quasi del tutto la mente perché mi stavo godendo così tanto il gameplay.

Gli sprite tremolanti sono meglio che vedere il NES rallentare e sbuffare, un altro evento comune nei giochi NES, e Mighty Final Fight non rallenta mai.

Concludendo, Mighty Final Fight è un fantastico picchiaduro che viene ingiustamente trascurato e spesso dimenticato. Penso che ciò sia dovuto al suo rilascio tardivo su NES.

A questo punto lo SNES era già uscito da due anni e Final Fight 2 stava per essere rilasciato su quel sistema. Oggi, il rilascio tardivo significa che è raro e costoso nella scena del collezionismo di videogiochi retrò.

Fortunatamente la versione Famicom è molto più economica e non contiene cambiamenti importanti oltre alla lingua nelle scene tagliate.

Consiglio di seguire quella strada se volete giocare a questo gioco con una cartuccia originale. Nel complesso, ho difficoltà a pensare a qualcosa di negativo su Mighty Final Fight.

È un po 'corto e non ha una modalità cooperativa per due giocatori, ma è comunque un'esperienza per giocatore singolo molto divertente e stimolante. Alcuni boss vengono ripetuti e gli sprite lampeggiano molto, ma il gameplay, il controllo, la colonna sonora e l'arte sono così buoni che ti dimentichi di questi difetti.

Consiglio vivamente questo gioco sia ai fan occasionali del NES che ai collezionisti più accaniti e ci sono modi in cui puoi giocare a questo gioco senza spendere i tuoi sudati dollari per una copia della cartuccia NES. Oppure di testarlo su emulatore o su Switch tramite lo shop virtuale.

Ci sono molte opzioni per uscire e riportare la pace a Metro City.

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 90%

Un gameplay eccezionale con un sistema di avanzamento del personaggio innovativo per questo tipo di giochi. Peccato per la mancanza del secondo giocatore contemporaneamente.

» Longevità 75%

Divertente, ben sviluppato e con una buona difficoltà durante il gioco. Non è lunghissimo.

MARIO KART 64

Anno: 1996
Editore: Nintendo
Sviluppatore: Nintendo
Genere: Guida
Piattaforma: Nintendo 64

Nel mondo di Mario, tutto è fantasia; niente può essere reale, niente può avere senso.
Ma che succede quando i personaggi del mondo di Mario vengono messi al posto di guida?

Mario Kart 64 è un sequel eccezionale di Super Mario Kart (SNES).

Il Super Mario Kart originale conteneva quattro modalità di gioco: Mario GP, Modalità Versus, Modalità Battaglia e Prova a tempo - disponibili solo per 1-2 giocatori.

Su N64 Mario Kart fa un grande salto in avanti nel suo gameplay. Ora, da 1 a 4 giocatori possono sfidarsi su tutte le nuove scintillanti piste. Ben 16 piste in modalità Grand prix divise in quattro coppe: Mushroom Cup, Flower Cup, Star Cup e Special Cup.

Come nel primo bellissimo gioco per il Super Nintendo (un capolavoro assoluto di gameplay dopo quasi 30 anni, ndr) i giocatori potranno sfidarsi in modalità 50 cc, 100 cc e nella furiosa e velocissima categoria 150cc. La difficoltà ovviamente si basa su queste classi di velocità, più è bassa e più è semplice e, ovviamente, più è alta e più la sfida sarà complicata.

Ovviamente non mancano gli avversari di gioco, 8 personaggi tutti con le proprie caratteristiche personali (da Toad leggero e veloce ma fragile fino a Bowser, pesante e distruttivo), non mancano i power up e le armi da poter lanciare per farsi strada.

Nella modalità GP a 2 giocatori, potranno unire le forze o competere contro altri sei piloti. La modalità "Versus" invece presenta in modo brutale la sfida tra 2-4 giocatori sulle piste del gioco. Senza coppe o premi. Semplice brutalità tra rivali.

Sono presenti come dicevamo 8 personaggi giocabili: Mario, Luigi, Principessa Peach, Toad, Wario, Yoshi, Donkey Kong e Bowser. Classificati come pesi leggeri, medi e massimi.

Pertanto ogni classe di peso (come dicevamo qualche riga sopra) ha una forza e una debolezza e saremo costretti a scegliere con un po' di pratica il personaggio più adatto a noi.

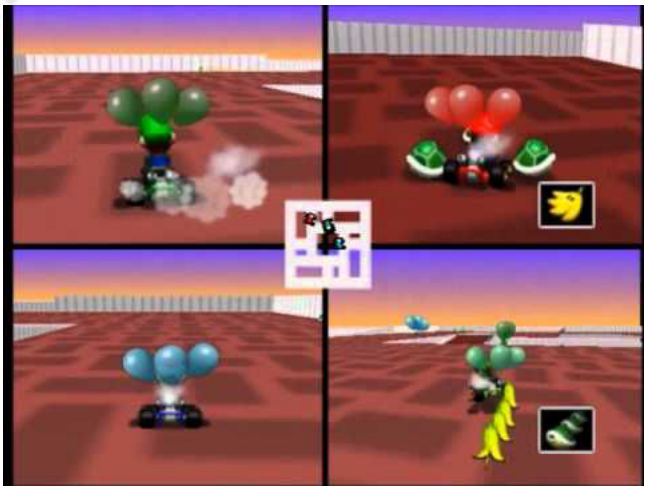
Io prediligo ad esempio Toad e Yoshi, ma il primo soffre per la stazze e il secondo per la maneggevolezza.

Sono presenti le classiche armi del primo Mario Kart come la buccia di banana, il guscio verde e il guscio rosso, il time ghost che ci permette di sparire momentaneamente, la stella.

A queste si aggiungono le banane multiple e il set di gusci, ovvero tre gusci che ci gireranno attorno mentre corriamo e ci permetteranno di avere più proiettili rispetto al normale.

E' presente anche il finto blocco power up che, se usato correttamente, può diventare devastante e molto fastidioso.

Il vero gusto del gioco però è nella modalità Battle, disponibile per 2-4 giocatori contemporaneamente, che



consente di giocare una partita in quattro arene diverse.

In questa modalità ogni giocatore riceve tre palloncini. Lo scopo è quello di evitare di farsi distruggere questi palloncini dall'avversario cercando di farli saltare a lui.

Questa modalità è forse la parte più divertente e distruttiva del gioco. Capace di tenerci incollati allo schermo e di far crollare amori o amicizie visto l'alto tasso di sfida.

Il salto generazionale tra il Super Nintendo e il Nintendo 64 è ben visibile nella grafica e nel suo sviluppo. Un ambiente 3D molto dettagliato dove gli sprite prendono più "coraggio" nei movimenti, le piste diventano più colorate e particolareggiate e sono presenti numerosi effetti grafici.

Un bellissimo e coloratissimo mondo fantasy in 3d in puro stile "NINTENDO". Certo non sfrutta completamente l'hardware della macchina come altri giochi (Mario 64 o Zelda), ma è un bel vedere e gira magnificamente.

Personalizzata anche la colonna sonora e gli effetti sonori. Ogni personaggio emette suoni o urla differenti.

I controlli del gioco sono perfettamente bilanciati e mai frustranti. Si accelera con "A" e si frena con "B". Il tasto "Z" seleziona l'oggetto e lo spara, i tasti dorsali "L" e "R" fanno saltare il personaggio.

Il pad "C" cambia le visualizzazioni, la levetta analogica viene utilizzata per la guida. Controlli predefiniti comodo e convenienti, ma è sempre possibile customizzarli.
In conclusione Mario Kart 64 è l'ideale per giocare contro i tuoi amici.

E' il tipo gioco da inserire nel periodo natalizio quando (DPCM permettendo) i parenti in casa si annoiano.

Accendiamo il Nintendo 64 e ci lanciamo in una sfida a 4 senza precedenti.

Un gameplay avvincente e una struttura di gioco in perfetto stile nintendo rendono il gioco in multiplayer la parte più bella del gioco.

Vi starete chiedendo perché non ho scritto qualche riga o una recensione anche per il primo Super Mario Kart.

Perché reputo questa versione il punto di svolta della serie.
L'evoluzione che ha permesso a brand di continuare anche nelle sue future incarnazioni.

Un diamante grazie appunto alla versione multiplayer.

Mario Kart 64 è un gioco ben fatto, divertente, chiassoso e quasi perfetto. Anche dopo tutti questi anni.

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 95%
Numerose modalità di gioco sia in singolo che in doppio, sistema di difficoltà calibrato con le classi dei go kart, controlli perfetti.

» Longevità 95%
Come si può dare un voto ad un gioco che ha fatto del multiplayer su schermo il suo punto di forza. Fino a quattro giocatori in modalità gp è divertente, in VS è violento e grezzo ma è con la Battle che diventa leggendario! Immortale!.

NEW GAME!!!

WIZ QUEST FOR THE MAGIC LANTERN

Anno: 2020
Sviluppatore: Mutation Software
Genere: Platform
Piattaforma: Amiga (solo chipset AGA)

Wiz Quest for the Magic Lantern è un nuovo gioco platform d'azione 2D a scorrimento laterale per Commodore Amiga. In Wiz, prendi il controllo dell'ultimo mago anziano e devi trovare la Lanterna Magica prima che sia troppo tardi. Non sei disarmato nella tua ricerca poiché hai a disposizione il tuo libro di magia e pozioni.
Avrai bisogno di ogni trucco per completare questa ricerca poiché il mondo davanti a te è pericoloso.
Un inizio classico per un gioco nuovo. Buon platform d'azione vecchio stile Quando pensiamo ai giochi platform d'azione in 2D, probabilmente pensiamo a Mario e compagnia. Si lo faccio anch'io.
Il franchise Nintendo è sinonimo del genere e per molteplici buone ragioni. Cosa dovrebbe rendere questo nuovo gioco per Amiga diverso dai classici? Wiz Quest for the Magic Lantern ha quel tocco di fascino in più che manca alla maggior parte dei giochi 2D.
Ad esempio il set di animazioni di movimento del nostro protagonista, ad esempio prendiamo l'animazione a piedi, con fotogrammi extra per naso e cappello. Entrambi si muovono ad ogni passo apportando un po' di realismo al gioco, ma non solo tutto il set delle animazioni è davvero ben curato.
Il diavolo è nei dettagli come dice un vecchio proverbio. Per fortuna, Wiz Quest for the Magic Lantern presta loro attenzione.
Gli sviluppatori hanno reso sviluppato il gioco in due modalità. Quella classica in versione fisica che presenta manuale, scatola e dischi (e diversi extra, ndr) e la versione in digital download per tutti quelli che non hanno la fortuna di possedere un Amiga 1200 o 4000 in forma fisica.
Tralasciando le specifiche e le caratteristiche, quello che abbiamo è un buon platform solido.
Il breve tempo di sviluppo e il fatto che questo sia il primo (di molti) nuovi giochi per Amiga di Mutation significa che non c'è molto da spingere su ciò di cui l'hardware è capace. Wiz non è un gioco veloce e di contrazione in alcun modo. Non c'è un timer, quindi puoi dedicare il tuo tempo a passare attraverso i livelli. Ed è un gioco che non permette gli errori, è davvero un prodotto "vecchia scuola".

Unico dubbio. E se non fosse per la bellissima grafica AGA, sono sicuro che il gioco girerebbe facilmente su un 68000 Amiga. Questo non vuol dire che Wiz sia un brutto gioco, in quanto non lo è, ma forse ottimizzando meglio il codice si sarebbe potuto godere anche con processori meno performatii dei 68020 o superiori. È un platform carino, simpatico e insolito con scorrimento fluido e che merita di essere in qualsiasi collezione di proprietari di Amiga.
Come dicevo il livelli non sono tantissimi, ma sono tutti piuttosto difficili da portare a termine senza l'attenzione necessaria. Questo vi darà diverse ore di gioco.

Il verdetto non può che essere positivo. Costa solo 9,99 dollari!
Lo potete ordinare qui:
http://softwareamusements.com/MutationSoftware/index.html
di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 75%
Il protagonista sarà anche tenero e delicato, i mostri simpatici.. ma il gioco è difficilotto e richiede pratica (è molto pixel perfect). Ben sviluppato il sistema di controllo e la fisica dei salti.

» Longevità 80%
Un bel 8 pieno come voto! Il grado di sfida è alto e vi farà passare diverse ore davanti allo schermo.

NEW GAME!!!

ZETA WING

Anno: 2020
Editore: Protovision
Sviluppatore: Sarah Jane Ivory
Genere: Shoot em up
Piattaforma: Commodore 64

Un perfetto gioco arcade. Questo è Zeta Wing.

Uno spara-tutto verticale tecnicamente perfetto, fluido, con tonnellate di parallasse, senza rallentamenti e dal game play avvincente.

Un prodotto sviluppato con tanto amore nei confronti della scena commodore e per amore dei suoi giocatori.

La trama è semplice. Ripulire il pianeta con la nostra astronave attraverso 7 livelli con la possibilità di migliorare il nostro sistema di armamento avanzando nei livelli.

L'avanzamento avviene con 10 potenziamenti del nostro sistema di difesa e, gaudio e tripudio, anche se si muore non si rimane a secco come in molti shoot em up, ma si ritorna indietro di un potenziamento. Questo perché il gioco è si ben bilanciato, ma è dannatamente frenetico e pieno di sfide.

In tutto questo aggiungiamoci una musica d'atmosfera per il genere, tre livelli di difficoltà ed il salvataggio del punteggio sul disco.

Un gioco degno dei migliori shoot em up sulla macchina Commodore, ma degno anche dei bellissimi giochi arcade dello stesso genere.

Bello e anche poco costoso. Solo 3,99 dollari attraverso il sito dell'autrice.

Non lasciatevelo sfuggire.

Il 2020 è un anno tribolato ma sul versante Commodore 64 ha visto una scena ricca e piena di piccoli gioielli.

Bello

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 90%
Un perfetto gioco arcade! Bilanciato, giocabile, fluido.

» Longevità 90%
Vi impegnerà per diverso tempo e il grado di sfida è ben strutturato.

SUPER MARIO 64

Anno: 1996
Editore: Nintendo
Genere: Plaform
Piattaforma: Nintendo 64

Nel 1996 con il prepotente ingresso del "poligono" nel mondo dei videogiochi tutti gli sviluppatori del tempo si lanciarono nel produrre qualsiasi cosa utilizzando motori grafici tridimensionali per le più svariate macchine del tempo.

Un'impresa difficile e non sempre riuscita. Il salto generazionale non era ancora stato somatizzato completamente dai programmatori e, molto spesso, i risultati erano "strani" e deludenti.

Anche il Nintendo 64 vide il cambiamento della saga principale della compagnia approdare con una nuova veste tridimensionale, ma grazie alle sapienti mani e idee di Shigeru Miyamoto questo salto dal 2d al 3d fu azzeccatissimo.

E così arrivò un nuovo capitolo di Mario anche su Nintendo 64 e rivoluzionò il mondo dei platform influenzandolo per le generazioni future.

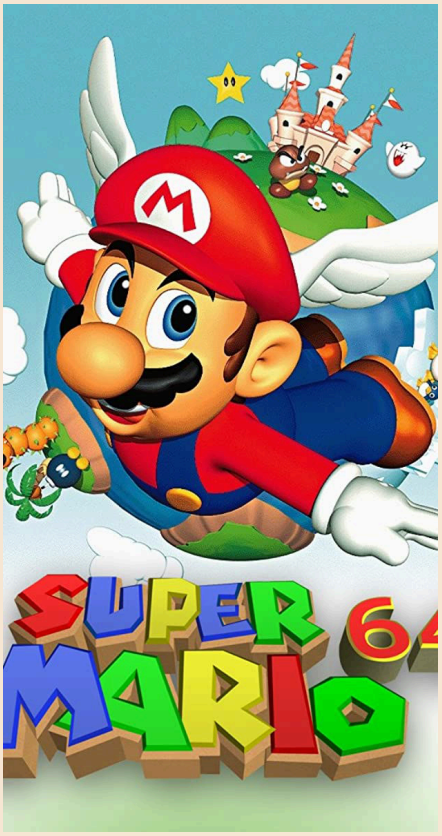
La Nintendo aveva fatto centro ancora una volta, creando un perfetto equilibrio tra sistema di controllo, grafica, giocabilità e senso di esplorazione che prima di allora non si era ancora visto. Questo fece si Super Mario 64 un autentico capolavoro riconosciuto universalmente.

Quando ero piccolo impugnando il "TRICORNUTO" pad e prendendo il controllo del baffuto idraulico, riuscivo a percepire la sensazione che questa sua nuova avventura avrebbe aperto chissà quali porte nel futuro dei videogiochi. Ed è stato davvero così. Con i 35 anni dalla nascita della saga dell'idraulico italo-americano, Nintendo ha deciso di fare riuscire tre titoli per la Switch, tra questi ovviamente c'è Super Mario 64.

Ma visto che in questo numero Nith ha parlato proprio della console originaria (nella sezione Hardware, ndN) parlerò di questa versione. Super Mario 64 è un piacere tuttora, la ricerca delle 120 stelle può ancora diventare un'ossessione, anzi fosse per me ne avrei aggiunte ancora di più.

Agli occhi moderni del classico videogiocatore, potrà risultare macchinosa la videocamera che vi segue (in realtà io la trovavo un po' fastidiosa già nel 96, ndN) e la grafica un po' "retrò", ma ciò nonostante dal punto di vista tecnico SM64 si difende ancora piuttosto bene e sfrutta a dovere l'hardware del Nintendo 64.

Il sistema di controllo è perfetto. Mario si comanda che è un piacere e davvero siamo liberi di fare quello che ci pare. Questo ha cambiato l'approccio alla



saga anche nei capitoli successivi come Sunshine per Game Cube e i bellissimi Galaxy per Wii.

Per tutti coloro che volessero tuffarsi in uno dei videogiochi più importanti e divertenti di sempre consiglio caldamente di recuperarlo.

Sia per il vostro N64 in versione fisica, sia su emulatore, sia in versione Rom per Everdrive o anche nella bella raccolta su Switch.

Provatelo, di giochi belli o molto belli ce ne sono tanti, ma sono pochi i titoli capaci di creare una nuova strada per il futuro. Super Mario 64 creò la via per il futuro. Un capolavoro di Miyamoto che fu il trampolino di lancio per l'ultima console a cartucce della storia.

Prima di lasciarvi spiego i miei voti. Se ci fosse un globale darei un secco 95% perché è l'insieme. I voti che ho espresso sono una "mia voglia", ovvero oggi come oggi vorrei molto di più, vorrei più stelle, più livelli, più Mario.

di **Hakim Rezki**

GIUDIZIO FINALE

» Giocabilità 90%
Un piccolo miracolo di gioco, con un sistema di controllo incredibile. Ancora oggi è un piacere giocarci.

» Longevità 85%
120 Stelle sono tante. Ma ne volevo di più.

FINAL FANTASY VII

Anno: 1997
Editore: Sony
Sviluppatore: Square Soft
Genere: JRPG
Piattaforma: Playstation 1/PC

Nel mondo dei videogiochi, la storia e l'ambientazione di Final Fantasy VII sono state elevate al regno del mito. Pubblicato per la prima volta nel 1997 su Sony PlayStation, ha inaugurato una nuova era dei giochi di ruolo giapponesi e, nel frattempo, la sua storia dove l'eroe Cloud Strife e una banda disordinata di idealisti e tirapiedi combattono contro l'accattivante e malvagio Sephiroth per per il bene del pianeta, è diventata più grande della vita per molti giocatori.

Cloud, Sephiroth, la città fluttuante di Midgar: questi sono significanti nel mondo dei giochi che vanno molto più in là del gioco da cui provengono ed esistono con potere al di fuori del loro contesto.

Final Fantasy VII è stato la killer application assoluta per la macchina Sony e forse l'esperienza visiva più abbagliante per quella console e per le console di quel periodo.

I filmati di alta qualità si fondevano perfettamente con la grafica del gioco e con le animazioni per creare il mondo sorprendentemente realistico del gioco.

Ma quello che colpiva di più del gioco era la trama che questa estetica aiutava a tessere.

La commovente trama di Final Fantasy VII è influenzata da alcune delle più grandi opere di film e letteratura di fantascienza, tra cui Dune di Frank Herbert, Frankenstein di Mary Shelley e persino Godzilla.

Proprio quest'anno è uscito, sull'onda di innumerevoli operazioni simili, il Remake per le console di nuova generazione. Un remake rifatto graficamente e curatissimo, che cambia di pochissimo le meccaniche di gioco e l'andamento della trama.
Ma concentriamoci su questa primordiale versione del 1997.

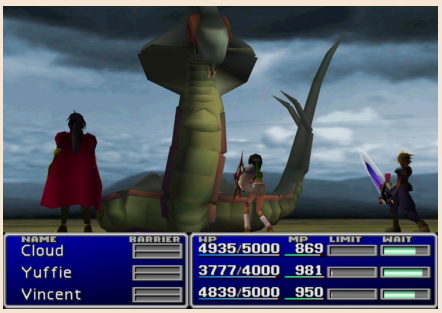
Come per i precedenti e bellissimi episodi apparsi sulle console Nintendo, questo episodio si presentó al grande pubblico che stava imparando ad apprezzare i videogiochi. Un pubblico ben più vasto di quello che aveva vissuto la precedente console wars anni 90, che presentava al suo interno neofiti letteralmente folgorati dal sistema di gioco ben strutturato e dalla narrativa perfetta.

Incollati per ore allo schermo, emozionati dalle storie. Pensate ho visto persino piangere ragazzi davanti allo svolgersi della storia.

Ma dopo tutti questi anni cosa possiamo dire di questa versione? E' ancora bella?

Risente dell'età esteticamente, ma quando riaprirete il cofanetto e riaccenderete la prima Playstation la magia sarà di nuovo li con voi.
Parola di bardo.

di **Roberto "Il Bardo" Del Mar Pirazzini**



GIUDIZIO FINALE

» Giocabilità 90%
Sistema di controllo intuitivo e godibilissimo. La narrazione perfetta vi terrà incollati allo schermo.

» Longevità 95%
Tre cd rom di enigmi, combattimenti e storie. Non dico altro.

NEW GAME!!!

SYDNEY HUNTER AND THE CAVERNS OF DEATH

Anno: 2020
Editore: CollectorVision
Sviluppatore: CollectorVision
Genere: Puzzle/Platform
Piattaforma: Super Nintendo

L'anno scorso, ho giocato con gusto a Sydney Hunter and the Curse of the Mayan per Switch e l'ho adorato, quindi immagina la mia sorpresa quando ho scoperto che il suo predecessore è effettivamente disponibile come gioco Super Nintendo!
Una volta che ci ho messo le mani sopra, sono stato immediatamente riportato alla metà degli anni '90, quando i giochi SNES dominavano la console wars. Meravigliosi anni 90!

Partiamo dal packaging. La scatola è bellissima, completa di fantastiche copertine di Joe Simko (Joe Simko è un illustratore di New York City che ha contribuito alle carte collezionabili di Topps Garbage Pail Kids e Wacky Packages, ndNth) ed è effettivamente migliore delle scatole del gioco "vecchio stile" perché è realizzata con un materiale durevole invece che con un cartone fragile. All'interno, la cartuccia del gioco si trova in un supporto di plastica trasparente che, ancora una volta, è probabilmente migliore della classica confezione del gioco SNES. Infine, il manuale che contiene informazioni preziose e, naturalmente, una pagina in cui scrivere le tue password. È davvero un pacchetto straordinariamente ben fatto.

Allora, per quanto riguarda il gioco vero e proprio? Sydney Hunter è un geologo in missione per esplorare il Monte Fato. Presto rimane intrappolato nella montagna e l'obiettivo è guidarlo verso la libertà. Lo fai correndo, saltando e arrampicandoti attraverso 12 complicati livelli labirintici mentre lanci il tuo boomerang alle creature terribili che risiedono nelle caverne.
Molte stanze sono avvolte dall'oscurità, ma fortunatamente Sydney ha una torcia che può impugnare e che illumina la stanza. Tuttavia, non può correre con la torcia, quindi sei costretto a memorizzare dove si trovano i nemici e i pericoli. La formula risultante è piuttosto impegnativa e mette alla prova la tua capacità di avanzare lentamente attraverso le stanze.

Una cosa che questo gioco fa eccezionalmente bene è offrire scenografie che premiano l'esplorazione visto che ci sono molti tesori nascosti ovunque. Dovrai esplorare per progredire poiché la maggior parte delle fasi include blocchi che non si aprono a meno che tu non posizioni determinati oggetti sui rispettivi piedistalli. A volte, la rimozione degli oggetti fa salire gradualmente la lava, quindi dovrai riportare l'oggetto rapidamente sul suo piedistallo, altrimenti Sydney potrebbe essere solo il prossimo sacrificio per le divinità dell'isola.

A livello grafico, le animazioni di Sydney sono molto belle. Guardarlo oscillare sulle corde e correre e saltare in giro è un piacere. Tuttavia, gli ambienti non cambiano mai veramente, quindi abituati a vedere molti contorni di roccia blu, lava rossa e viti verdi. Tra i lati positivi, la musica è ben fatta e fornisce alcuni brani tribali mescolati con sintetizzatori old school mentre esplori.

In un periodo particolare come questo Sydney è un bel gioco con cui rilassarsi e perdersi tra le caverne. Lo trovate sul sito https://collectorvision.com/ assieme ad altri titoli interessanti per molti sistemi (Amiga, Intellivision, Coleco, Snes...) al costo di 40 dollari in versione completa o 25 dollari loose.

Ne vale la pena.
Dimenticavo. La saga di Sydney Hunter è presente in diverse piattaforme di gioco: la trovate anche su Sega Master System, Coleco Vision, Intellivision e Nes. Sono tutti giochi molto diverti che sfruttando a dovere le potenzialità delle macchine su cui girano. Parola del bardo!

di **Roberto "il Bardo" Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 80%
Confezione di prima qualità, perfetta per i collezionisti di giochi retrò. Gameplay impegnativo ma semplice. Buona musica e belle animazioni.

» Longevità 65%
Abbastanza breve senza molta rigiocabilità. Ambienti invariati durante l'avventura.

RISTAR

Anno: 1995
Editore: Sega
Sviluppatore: Sega
Genere: Platform
Piattaforma: Sega Megadrive

"*Noi siamo figli delle stelle*
Figli della notte che ci gira intorno
Noi siamo figli delle stelle
Non ci fermeremo mai per niente al mondo"

Alan Sorrenti – Figli delle Stelle.

Non potevo non fare riferimento a questa canzone italiana dei primi anni 80 per raccontare la storia di Ristar, figlio delle stelle e eroe di questo platform su Megadrive.

Uno degli ultimi platform prodotti prima della dismissione della console.

E questo nostro "figlio delle stelle" ha l'arduo compito di salvare la galassia di Valdi dai piani del malefico alieno Greedy. Certo non è l'eroe più forte ne quello più tecnologico, ma è quello più eroico!Il nostro eroe non dispone di nessun super spin o super salto ma di un paio di braccia estensibili con le quali potrà afferrare oggetti, aggrapparsi alle scale, salire nei posti più impensabili e, ovviamente menare i nemici.

Il gioco si presenta come il classico platform a scorrimento bidimensionale, simile a Super Mario o a Sonic (in questo caso per via dello stesso team di sviluppo, ovvero il Sonic Team, ndr).

Le braccia estendibili di Ristar sono utilizzate come mezzo principale per attaccare i nemici; estendendo le braccia, afferrando il nemico e spingendosi verso di loro in un movimento di " colpo di testa " per sconfiggerlo. Lo stesso movimento consente anche di aprire scrigni del tesoro contenenti vari oggetti o di colpire parti diverse dell'ambiente, come abbattere alberi. Inoltre, le sue braccia elastiche possono essere utilizzate anche per afferrare e / o lanciare oggetti.

Oltre ad attaccare, le braccia di Ristar sono anche usate come metodo per proiettarlo attraverso i livelli. Molte strutture simili a pali sono presenti per far oscillare Ristar da un lato all'altro, attraverso spazi vuoti o per salire o scendere verticalmente dalle piattaforme.

Ristar è anche in grado di aggrapparsi a nemici e oggetti a mezz'aria e dondolarsi su di essi. Inoltre, le "Maniglie a stella" sono collocate nei livelli, in cui il giocatore deve far afferrare Ristar e usare lo slancio per farlo oscillare in un cerchio di 360 gradi .

Lasciar andare lo lancia in una determinata direzione, a seconda del momento del rilascio. Se si guadagna abbastanza slancio, le scintille appaiono dietro Ristar e lui esegue una mossa chiamata "Meteor Strike", che lo rende invincibile e in grado di sconfiggere qualsiasi nemico toccandolo.

Quando si perde abbastanza slancio, di solito pochi secondi, il volo cessa e lui ricade a terra tornando al suo stato normale, anche se questo può essere esteso rimbalzando contro pareti e soffitti durante il volo.

Ogni livello termina con una speciale "Maniglia a stella", che viene utilizzata per avviare Ristar fino alla fine del livello. I punti bonus vengono assegnati in base all'altitudine di Ristar quando si vola fuori dallo schermo, in modo simile a come finiscono i livelli in Super Mario Bros.

Il gioco arrivo un po' in sordina, in una periodo particolare quello del cambio generazionale tra console.

Il team di sviluppo ha sviluppato un prodotto davvero eccellente dal punto di visto grafico. Coloratissimo e superbamente animato, questo gioco è una vera gioia degli occhi.

Meno eccellente è il comparto sonoro, ma godibile.

Innovativo nel game play ma forse meno intuitivo rispetto a Sonic. Macchinoso nel manovrare i movimenti del personaggio.
Sono presenti sei pianeti da ripulire, in ognuno dei quali vi sono due livelli, con un miniboss nel primo e uno vero e proprio nel secondo. Una volta completati i pianeti si arriva allo scontro finale col perfido Greedy.

Impresa non impossibile e quindi la longevità finale è nella media.

Consiglio finale, recuperate in qualche modo questo prodotto di nicchia e giocatelo.
Passerete qualche oretta rilassandovi … tra le stelle!

di **Carlo N. Del Mar Pirazzini**

GIUDIZIO FINALE

» Giocabilità 80%
E' un prodotto del Sonic Team e quindi è ben sviluppato sotto tutti i punti di vista, ma non è intuitivo come la serie Sonic. Ci vuole un po' di pratica per gestire al meglio le abilità del personaggio.

» Longevità 80%
Sei mondi da esplorare e una longevità tipica di questo genere di giochi.

Diamo i numeri!

No, non voglio parlarvi dei numeri del COVID, a cui ormai siamo tutti tristemente abituati; nell'ultimo numero dell'anno, nell'ultimo articolo dell'ultima pagina, vorrei condividere con voi lettori un po' di numeri e fatti che hanno definito questo ed i primi 4 anni di RetroMagazine World.

Quest'anno che si sta avviando alla conclusione, a parte tutte le difficoltà che lo hanno caratterizzato, ha visto la nostra rivista trasformarsi profondamente: RetroMagazine ha cambiato nome, diventando **RetroMagazine World**.
Un cambiamento che si è reso necessario perché abbiamo voluto aprirci al mondo pubblicando anche la versione in inglese.
Mantenere due pubblicazioni in due lingue differenti, richiede tempo ed uno sforzo notevole da parte di tutta la redazione, per questo motivo avrete notato un leggero diradamento delle uscite... Ma non preoccupatevi, siamo sempre vivi e vegeti. Oggi più che mai, in barba al COVID!

Chi segue la nostra **Pagina Facebook** (https://www.facebook.com/RetroMagazine-World-2005584959715273) inoltre, si sarà indubbiamente accorto del cambio di passo che ha compiuto quest'anno: ogni settimana pubblichiamo decine di post, curiosità e preview dei nuovi giochi in sviluppo. Merito di questo deciso cambiamento va a **Nithaiah** che instancabilmente si sta prodigando per trovare informazioni da condividere con tutti voi.

Ah, per chi non lo avesse notato, abbiamo aperto anche un canale su **Twitter** (https://twitter.com/RetromagazineW). Se vi trovate più a vostro agio su quel social, seguiteci anche lì.

Ma non volevi dare un po' di numeri? Certo! Eccoli che arrivano!
In questi 4 anni abbiamo pubblicato 27 numeri in italiano (compreso quello che tenete tra le mani) e 5 numeri in inglese. Sono stati pubblicati in queste pagine più di 480 articoli di cui 190 recensioni di giochi e più di 100 articoli dedicati alla programmazione. 16 Sono state le rubriche RetroMath e 24 le interviste. Abbiamo inoltre toccato più di 15 linguaggi di programmazione e ben più di 70 computer differenti.
Ovviamente queste statistiche non sono fini a se stesse, ma ci hanno permesso di scoprire che abbiamo dato poca visibilità alle macchine Atari e soprattutto all'Atari ST. Non era certamente nostra intenzione snobbare questo computer; purtroppo nessuno in redazione ne è un vero esperto e non abbiamo mai ricevuto contributi esterni. Pubblichiamo quindi un appello:
Siete esperti di Atari 8-bit ed Atari ST e volete collaborare con noi?
Fatevi avanti, le porte delle redazione sono aperte!

Ah, la copertina di questo numero natalizio è speciale ed è un chiaro omaggio ad una famosa rivista italiana. Vediamo chi indovina sia la rivista sia il numero che ha ispirato il nostro **Flavio**.

Prima di lasciarVi voglio augurarVi a nome di tutta la redazione di RetroMagazine World, Buone Feste con l'augurio che, il 2021, possa vedere presto terminare l'emergenza COVID perché vogliamo tornare di persona ai nostri retroeventi!

Francesco Fiorentini

RetroMagazine World
Anno 4 - Numero 27 - DICEMBRE 2020

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